A New Method of Identifying Critical Paths for Performance Optimization

Shiang-Tang Huang Tai-Ming Parng

Jyuo-Min Shyu

Dept. of Electrical Engineering
National Taiwan University
Taipei, Taiwan, R.O.C

Computer & Communication Research Labs.
Industrial Technology Research Institute
Hsinchu, Taiwan, R.O.C

Abstract

This paper describes a new method of identifying the critical paths for performance optimization. We show by example that, during performance optimization, some false paths may become sensitizable, selection of critical paths therefore need to consider some false paths besides sensitizable paths. These critical paths are called candidate paths in this paper. We have developed new techniques for extracting candidate paths, and an effective graph model for representing and manipulating the huge number of candidate paths. The speed and memory utilization of the proposed method are demonstrated using ISCAS benchmark circuits.

1 Introduction

Performance optimization (assuming fixed circuit topology) is an iterative process involving repeated timing analysis and critical path selection. However, due to the existence of *false paths*, i.e., paths which can never be activated by input vectors, critical paths identified by using static timing analysis techniques [1, 2, 3, 4, 5, 6] may consist of *sensitizable paths* (non-false paths) and false paths, resulting in inefficient optimization process.

Many researchers have proposed techniques to deal with the path sensitizability problem [7, 8, 9]. These techniques, while powerful in identifying sensitizable paths, do not consider false paths which may become sensitizable during optimization. A performance optimizer using these techniques therefore need to call the sensitizability analysis routine iteratively to identify the critical paths for optimization purposes. The high CPU time cost of sensitizability analysis slows down the efficiency of performance optimization.

We have observed that reporting some false paths as critical paths can avoid iterative sensitizability analysis and thus increase the efficiency of optimization. Consider the example circuit in Fig. 1. Assume that the rising/falling delays of G_1 , G_2 , and G_3 are 2, 1, and 1, respectively. It is easy to show that when A is logic 1 at time 0, D will be set to logic 0 at time 2; when A is logic 0 at time 0, D will be set to logic 1 at

rising/falling delays of G_1 are 2.
rising/falling delays of G_2 and G_3 are 1.

Figure 1: Example Circuit.

time 3. Obviously, the paths $P_1 = A-(G_2)-\overline{C}-(G_3)-\overline{D}$ and $P_2 = \overline{A}-(G_1)-B-(G_3)-D$ are sensitizable, and the paths $P_3 = A-(G_1)-\overline{B}-(G_3)-\overline{D}$ and $P_4 = \overline{A}-(G_2)-C-(G_3)-D$ are false. Note that since the maximal delay for D to be logic 1 is $\min(\text{delay of } P_1, \text{delay of } P_3)$, to improve the speed, we only need to consider P_1 ; P_3 can be ignored. However, since the maximal delays for D to be logic 1 is $\max(\text{delay of } P_2, \text{delay of } P_4)$, to improve the speed, we need to consider both P_2 and P_4 . To summarize, during optimization, we only need to iteratively calculate the delays of P_1 , P_2 , and P_4 (not the sensitizabilities of these paths), and the CPU time cost can therefore be significantly reduced.

Based on the above observation, in this paper we consider performance optimization (on a fixed circuit topology) as a selection process to choose, among a set of topologically and functionally identical circuits, a "faster" one. We then propose a new concept for critical paths, called *candidate paths*, to include both sensitizable paths and some false paths which may become sensitizable and dominate the circuit delays during optimization. After investigating the conditions for a false path to become sensitizable, we propose two rules for judging if a false path is a candidate path or not. Since the number of candidate paths may be too large to manipulate in practice, we propose a new graph model, called *Delay Expansion Graph (DEG)*, to represent the candidate paths (with delay information incorporated in the graph). Algorithms to extract candidate paths with delay values larger than a given threshold (or the first K long candidate paths) are

[†]This work was supported by the National Science Council R.O.C under Grant NSC 81-0404-E002-110.

also presented.

The contents of this paper are as follows. Section 2 gives the definitions of paths and sensitizable paths, while the definition of a candidate path set is presented in Section 3. The rules for extracting the candidate paths is presented in Section 4. Section 5 details the algorithm to construct a DEG. Section 6 gives the methods for selecting critical paths from a DEG. The experimental results on ISCAS benchmarks are given in Section 7. Finally, Section 8 summarizes the presentation.

2 Paths and Sensitizable Paths

Definition [Physical Node]: The input ports, output ports, and internal nodes (nets) of a logic circuit are referred to as *physical nodes*.

Definition [Logical Node]: For each *physical node* N , there are two corresponding *logical nodes*, N and $\bar{N}$. N models the dynamic behaviors of N in logic state 1, while $\bar{N}$ models N in logic state 0. When N is unknown, N and $\bar{N}$ are defined to be 0. When N is logic 1, N is logic 1 and $\bar{N}$ is logic 0. When N is logic 0, N is logic 0 and $\bar{N}$ is logic 1.

Definition [Path]: A path $P = n_0-(G_1)-n_1-(G_2)-n_2-\dots-(G_k)-n_k$ (or $\bigcup_{i=1}^k n_{i-1}-(G_i)-n_i$) is defined to be a finite alternating sequence of logic gates ($G_1, G_2, \dots, G_k$) and logical nodes ($n_0, n_1, \dots, n_k$), where n_{i-1} and n_i are the input and output nodes of G_i respectively, and the transition of n_{i-1} will cause n_i to switch if the other inputs of G_i are properly set.

The delay of $n_{i-1}-(G_i)-n_i$ in L , denoted as $PD(n_{i-1}-(G_i)-n_i, L)$, is the time to propagate signals from n_{i-1} to n_i . Since $PD(n_{i-1}-(G_i)-n_i, L)$ can be calculated, we can assume $PD(n_{i-1}-(G_i)-n_i, L)$ is available. Thus the delay of a path $P = n_0-(G_1)-n_1-(G_2)-n_2-\dots-(G_k)-n_k$ in L , denoted as $PD(P, L)$, can be calculated by the following formula:

$$PD(P, L) = \sum_{i=1}^k PD(n_{i-1}-(G_i)-n_i, L).$$

Definition [Subpath]: A path P_1 is a *subpath* of another path P_2 if (1) all gates and nodes on P_1 are also on P_2 , and (2) the order in which the gates and nodes appear on P_1 is the same as the order in which they appear on P_2 .

Definition [Controlling and Non-Controlling Node]: For a logic gate G , its input logical nodes are partitioned into *controlling* and *non-controlling nodes*. The following table lists the controlling and non-controlling nodes of some primitive gates.

Gate	Non-Controlling Nodes	Controlling Nodes
AND	$A_1, \dots, A_n$	$A_1, \dots, A_n$
NAND	$A_1, \dots, A_n$	$A_1, \dots, A_n$
OR	$A_1, \dots, A_n$	$A_1, \dots, A_n$
NOR	$A_1, \dots, A_n$	$A_1, \dots, A_n$
INV	A	A
BUFFER	A	A

where $A_i(\bar{A}_i)$ is the input node of G .

Note that (1) the partition on INV and BUFFER is for consistency consideration and (2) one of the two output logical nodes of G is the logic-OR of the controlling nodes; while the other is the logic-AND of the non-controlling nodes.

Definition [Active Time]: The *active time* of a logical node n in a circuit L under an input vector I , denoted as $AT(n, I, L)$, is defined to be the time when n becomes and stays in logic 1. $AT(n, I, L)$ is calculated according to the following steps:

1. If the value of n is 0, $AT(n, I, L)$ is ∞ .
2. If n is a primary input node, $AT(n, I, L) = 0$.
3. If the value of n is the logic-OR of the controlling nodes of G ,

$$AT(n, I, L) = \min_{i=1}^m (AT(a_i, I, L) + PD(a_i-(G)-n, L))$$
where a_i is the input controlling node of G .
4. If the value of n is the logic-AND of the non-controlling nodes of G ,

$$AT(n, I, L) = \max_{i=1}^m (AT(a_i, I, L) + PD(a_i-(G)-n, L))$$
where a_i is the input non-controlling node of G .

Definition [Sensitizable Path]: Given a circuit L , a path $P = n_0-(G_1)-n_1-(G_2)-n_2-\dots-(G_k)-n_k$ is a *sensitizable path* if and only if there exists at least one input vector I such that (1) for all n_j on P , the value of n_j is logic 1, and (2) for all n_j on P , $AT(n_j, I, L) = \sum_{i=1}^j PD(n_{i-1}-(G_i)-n_i, L)$.

Definition [False Path]: A path which is not a *sensitizable path* is a *false path*. In other words, a path $P = n_0-(G_1)-n_1-(G_2)-n_2-\dots-(G_k)-n_k$ is a false path if and only if for any input vector I (1) there exists a node n_j on P and the value of n_j is logic 0, or (2) there exists a node n_j on P and $AT(n_j, I, L) \neq \sum_{i=1}^j PD(n_{i-1}-(G_i)-n_i, L)$.

3 Candidate Path Set

An optimization process on a circuit L could be viewed as a selection process choosing a "faster" circuit among a set of topologically and functionally identical circuits. This set of circuits is referred to as the *circuit family* of L , denoted as $Family(L)$.

Definition [Maximal Delays of a Logical Node and a Circuit]: The maximal delay of a circuit L , denoted as $MaxD(L)$, is the delay of the longest sensitizable path in L , while the maximal delay of a logical node n in L , denoted as $MaxND(n, L)$, is the delay of the longest sensitizable path ending at n . Obviously,

$$MaxD(L) = \max_{n \in L} MaxND(n, L)$$

Note that the computation of $MaxND(n, L)$ can either be found in our previous work on **Timed Boolean Algebra** [10], or can be performed by using the active times of n :

$$MaxND(n, L) = \max_{I \in V(I)} AT(n, I, L)$$

where $V(I) = \{I \mid \text{value of } n \text{ under } I \text{ is logic 1}\}$.

Definition [Candidate Path Set]: A candidate path set of a logic circuit L , denoted as $CPS(L)$, is a set of paths such that

$$\forall L', L' \in \text{Family}(L), \exists P, P \in CPS(L), \\ \text{MaxD}(L') \leq PD(P, L')$$

4 Extracting the Candidate Path Set

In this section, we discuss what false paths can be ignored during the optimization process. By removing these paths a candidate path set is obtained.

Firstly, we investigate the conditions for a false path to become sensitizable. Assume the circuit being optimized is L , and the circuit after optimization is L' , $L' \in \text{Family}(L)$. Let path $P = n_0-(G_1)-n_1-(G_2)-n_2-\dots-(G_k)-n_k$ be a false path in L and a sensitizable path in L' . Based on the definitions of sensitizable and false paths, there exists at least one input vector I such that

1. for all n_j on P , the value of n_j is logic 1,
2. there exists n_j on P and $\sum_{i=1}^j PD(n_{i-1}-(G_i)-n_i, L) \neq AT(n_j, I, L)$
3. for all n_j on P , $\sum_{i=1}^j PD(n_{i-1}-(G_i)-n_i, L') = AT(n_j, I, L')$

Without loss of generality, we can assume $j = k$ and the subpath $n_0-(G_1)-n_1-\dots-(G_{k-1})-n_{k-1}$ of P is sensitizable under input vector I , $\sum_{i=1}^{k-1} PD(n_{i-1}-(G_i)-n_i, L) = AT(n_{k-1}, I, L)$. Thus, $\sum_{i=1}^k PD(n_{i-1}-(G_i)-n_i, L) = PD(P, L)$ and $PD(P, L) \neq AT(n_k, I, L)$. Now, consider the following two cases:

Case 1. $PD(P, L) < AT(n_k, I, L)$

From the formula for calculating active times, we find n_{k-1} must be a non-controlling node of gate G_k and the formula for calculating $AT(n_k, I, L)$ is of the form $\text{maz}(\dots)$. Let P' be a sensitizable path ending at n_k in L , $PD(P', L) = AT(n_k, I, L)$. Since P is sensitizable in L' , $PD(P, L') = AT(n_k, I, L')$ and the formula for $AT(n_k, I, L')$ is also of the form $\text{maz}(\dots)$, we get $PD(P', L') \leq PD(P, L') = AT(n_k, I, L')$. This means P may become sensitizable and dominate the delay of n_k . So, P should not be ignored during optimization.

Case 2. $PD(P, L) > AT(n_k, I, L)$

In the similar way, we find and the formula for calculating $AT(n_k, I, L)$ is of the form $\text{min}(\dots)$. Let P' be a sensitizable path ending at n_k in L , $PD(P', L) = AT(n_k, I, L)$. Since P is sensitizable in L' , $PD(P, L') = AT(n_k, I, L')$, and the formula for calculating $AT(n_k, I, L')$ is of the form $\text{min}(\dots)$, we get $PD(P, L') \leq PD(P', L')$. This means P will meet the constraint if P' meets the constraint. So, P can be ignored during optimization.

Figure 2: A sample circuit.

From the above discussions, it is easily concluded that a false path $P = n_0-(G_1)-n_1-(G_2)-n_2-\dots-(G_k)-n_k$ in a circuit L can be ignored, if for any input vector I there exists a node n_j such that (1) the value of n_j is logic 0, i.e. $AT(n_j, I, L) = \infty$ or (2) $\sum_{i=1}^j PD(n_{i-1}-(G_i)-n_i, L) > AT(n_j, I, L)$. By examining this rule in detail, we find that it can be rewritten as follows:

[Rule 1]: A path $P = n_0-(G_1)-n_1-(G_2)-n_2-\dots-(G_k)-n_k$ in a circuit L is a non-candidate path if there exists a node n_j on P such that $\sum_{i=1}^j PD(n_{i-1}-(G_i)-n_i, L) > \text{MaxND}(n_j, L)$.

Moreover, we find that the candidate path set $CPS(L)$ of a multiple outputs circuit L can be further reduced by the following rule:

[Rule 2 :] If a path $P_1 \in CPS(L)$ is a subpath of another path $P_2 \in CPS(L)$, P_1 can be removed from $CPS(L)$, because if P_2 meets the constraint, P_1 does too.

5 Modeling the Candidate Paths

[Delay Expansion Graph] : A delay expansion graph (DEG) is a generalization of a circuit. The nodes and arcs in a DEG are called *delay expansion nodes* and *delay expansion arcs* respectively. The creation of delay expansion nodes, delay expansion arcs, and the DEG from a circuit can be proceeded as follows.

[Create the Delay Expansion Nodes] : Given a node n , if the number of possible delay values of n is k , n is expanded to k different nodes. Let the k possible delays of n be $d_1, d_2, \dots$ and d_k . The k delay expansion nodes are denoted as $n^{d_1}, n^{d_2}, \dots$ and n^{d_k} , where n^{d_i} represents n with expansion delay value d_i .

[Create the Delay Expansion Arcs] : Let n_1 and n_2 be the input and output logical nodes of a gate, and d be the delay to propagate signal from n_1 to n_2 . For each delay expansion node n_1^i , if n_2^{i+d} exists, a delay expansion arc from n_1^i to n_2^{i+d} is created.

Without removing the non-candidate paths, DEG can be used to model the set of all paths in a circuit. Consider the circuit in Figure 2. The rising delay of G_2 is 2 and the other delays are 1. The DEG which models the set of all paths is shown in Figure 3.

On the other hand, with the non-candidate paths removed, DEG can be used to model all candidate paths. Now, we apply the two rules described in Sec-

Figure 3: The DEG of the logic circuit in Figure 2. '*' denotes an output node.

tion 4 to modify the DEG and thus create a DEG which models all candidate paths.

1. **[Remove the unnecessary delay expansion nodes]** : Rule 1 is to remove the path $P = \bigcup_{i=1}^k n_{i-1}-(G_i)-n_i$, if there exists any node n_i on P such that $\sum_{j=1}^i PD(n_{j-1}-(G_j)-n_j, L) > MaxND(n_i, L)$.
Let $t_i = \sum_{j=1}^i PD(n_{j-1}-(G_j)-n_j, L)$, for $1 \leq i \leq k$, then this rule is to remove all the paths $P' = \bigcup_{i=1}^k n_{i-1}^{t_i}-(G_i)-n_i^{t_i}$ in the DEG, if there is any t_i such that $t_i > MaxND(n_i, L)$. Moreover, if $t_i > MaxND(n_i, L)$, then all the paths passing through $n_i^{t_i}$ have to be removed. So $n_i^{t_i}$ can be removed from the DEG.
2. **[Re-mark the output nodes which have succeeding node(s) as internal nodes]**: Rule 2 is to remove the path P_1 which is a subpath of another path P_2 in the candidate path set. Now, let P'_1 and P'_2 be the corresponding paths of P_1 and P_2 in the DEG respectively. P'_1 is also a subpath of P'_2 . Assuming the end node of P'_1 is $n_k^{t_k}$, there will be one or more succeeding delay expansion nodes after $n_k^{t_k}$ and thus all paths terminating at $n_k^{t_k}$ can be removed. To accomplish this task, we can simply re-mark $n_k^{t_k}$ as an internal node.

Again, we use the logic circuit L in Figure 2 as an example. The actual maximal delays of the logical nodes are shown below:

$$\begin{aligned} MaxND(D, L) &= 1 & MaxND(\bar{D}, L) &= 1 \\ MaxND(E, L) &= 3 & MaxND(\bar{E}, L) &= 2 \\ MaxND(F, L) &= 4 & MaxND(\bar{F}, L) &= 2 \\ MaxND(G, L) &= 2 & MaxND(\bar{G}, L) &= 3 \end{aligned}$$

The DEG which models all the logical paths is shown in Figure 3.

The algorithm to construct a DEG is very simple. Firstly, the event-driven technique commonly used in logic simulation is used to automatically traverse the paths and to get each node's possible delays. The required delay expansion nodes and delay expansion arcs can then be created.

6 Path Selection from a DEG

A DEG can be considered as a graph, where the values on the arcs represent the delays to propagate signals from nodes to nodes. (Note that the gates' delay values can be calculated by the optimizer.)

1. To extract the candidate paths with delay values larger than a given threshold T , the algorithm proposed in [5] can be used.
2. To obtain the first K long candidate paths, the algorithm proposed in [6] can be used.
3. To compute the delay of the longest candidate paths, the block-oriented algorithm [1] can be used.

7 Experimental Results

All the proposed techniques have been implemented based on a recently improved version of our previous work on **Timed Boolean Algebra**. The effectiveness and efficiency of the method have been tested by ISCAS benchmark circuits on SUN SPARCstation2.

Table 1 shows the number of paths, the number of candidate paths, and the reduction percentage (reflecting the savings in considering candidate paths instead of all the paths). Note that although the reduction percentage is not high for some circuits, the actual computational savings are very significant. To see this, consider C5315 as an example where the reduction percentage is only 3.9%. The paths and candidate paths with delays larger than 44 (assuming a unit delay model) are the long paths and long candidate paths which are of interest for optimization purpose. Figure 5 shows that the savings in the computational cost is indeed significant.

Table 2 shows the CPU times to construct DEGs and memory utilization for the benchmark tests. We see that all the circuits except C6288 are finished in two CPU seconds. For all cases, memory space required is less than 6M bytes.

8 Conclusion

In this paper, to avoid computing the redundant path sensitizabilities, a new definition of critical paths, called candidate paths, is proposed, and an effective model, called **Delay Expansion Graph**, is proposed

Figure 4: Apply the two rules to the DEG of Figure 3. "*" denotes an output node.

Table 1: No. of paths and candidate paths

CKT	No. of Paths (1)	No. of Candidate Paths (2)	Reduction % 1-(2)/(1)
C1908	1458114	418494	71.30%
C2670	1359768	892470	34.37%
C3540	57353342	52834624	7.788%
C5315	2682610	2578198	3.893%
C6288	1.979 E20	1.007 E20	49.09%
C7552	1452986	1397355	3.819%

Figure 5: Long paths and long candidate paths.

Table 2: Performance and Memory Utilization.

CKT	CPU (seconds)	No. of Nodes in DEG	Memory
C1980	0.50	16224	612
C2670	0.50	12511	561
C3540	0.95	27631	1013
C5315	1.28	31788	1256
C6288	5.23	179673	5548
C7552	1.69	49892	2034

to represent and manipulate the huge number of candidate paths. The experimental results on the ISCAS benchmark circuits show the effectiveness and the efficiency of our proposed algorithm for creating DEGs. In the future, we will try to incorporate more rules towards finding a minimal candidate path set.

References

- [1] R. B. Hitchcock et al, "Timing Analysis of Computer Hardware," *IBM. J. Res. Develop.*, Vol. 26, No. 1, Jan. 1982.
- [2] Lionel C. Bening et al, "Developments in Logic Network Path Delay Analysis," *Proc. 19th Design Automation Conf.*, 1982.
- [3] Norman P. Jouppi, "Timing Analysis for nMOS VLSI," *Proc. 20th Design Automation Conf.*, 1983.
- [4] Norman P. Jouppi, "Timing Analysis and Performance Improvement of MOS VLSI Designs," *IEEE Trans. Computer-Aided Design*, vol. CAD-6, No. 4, Jul. 1987.
- [5] H.C. Yen, S. Ghanta, and H.C. Du, "A Path Selection Algorithm for Timing Analysis," *Proc. 25th Design Automation Conf.*, 1988.
- [6] Steve H.C. Yen, David H.C. Du, and S. Ghanta, "Efficient Algorithms for Extracting the K Most Critical Paths in Timing Analysis," *Proc. 26th Design Automation Conf.*, 1989.
- [7] David H.C. Du, Steve H.C. Yen, and S. Ghanta, "On the General False Path Problem in Timing Analysis," *Proc. 26th Design Automation Conf.*, 1989.
- [8] Patrick C. McGeer and Robert K. Brayton, "Efficient Algorithms for Computing the Longest Viable Path in a Combinational Network," *Proc. 26th Design Automation Conf.*, 1989.
- [9] Hsi-Chuan Chen, David, and H.C. Du, "Path Sensitization in Critical Path Problem," *IEEE Intl. Conf. Computer-Aided Design*, 1991.
- [10] Shiang-Tang Huang, Tai-Ming Parng, and Jyuo-Min Shyu, "A New Approach to Solving False Path Problem in Timing Analysis," *IEEE Intl. Conf. Computer-Aided Design*, 1991.