False Path and Clock Scheduling Based Yield-Aware Gate Sizing[*]

Jeng-Liang Tsai[†], DongHyun Baik[†], Charlie Chung-Ping Chen[‡] and Kewal K. Saluja[†]

[†] University of Wisconsin-Madison
Department of Electrical and Computer Engineering
1415 Engineering Drive
Madison, WI 53706
{*jltsai@cae, dbaik@ece, saluja@engr*}*.wisc.edu*

[‡] National Taiwan University
Graduate Institute of Electronics Engineering &
Department of Electrical Engineering
Taipei 106, Taiwan
cchen@cc.ee.ntu.edu.tw

Abstract

Timing margin (slack) needs to be carefully managed to ensure a satisfactory timing yield. We propose a new design flow that combines a false-path-aware gate sizing and a statistical-timing-driven clock scheduling algorithms to maximize timing yield. Our gate sizing algorithm preserves the true path lengths that may otherwise be altered by the traditional gate sizing algorithms due to the presence of false paths. The slack is then distributed to each path according to its path delay uncertainty to maximize the timing yield. Experimental results show that our flow achieves significant timing yield improvements ($> 20\%$) than a traditional flow for a subset of the benchmark circuits with little or negligible area penalty.

1. Introduction

Device parameter variations can alter the expected performance of each device in a circuit, thus affecting the timing yield substantially. In the past, yield issues were usually addressed by improving the manufacturing processes. However, this approach is no longer sufficient in deep submicron technologies. New design methodologies must be able to incorporate statistical timing analysis to address the yield issues [1]. Although statistical timing analysis techniques have been studied for more than a decade [2], relatively little research has utilized the statistical timing information for designs. In fact it is not even clear how seriously the traditional circuit design techniques impact the yield and how to combine the statistical timing information into current design flows to improve the timing yield. Below we describe the qualitative impacts of gate sizing and clock scheduling on the yield, before delving into the details of this paper.

[*] This work was partially funded by National Science Foundation under grants CCR-0093309 & CCR-0204468 and National Science Council of Taiwan, R.O.C. under grant NSC 92-2218-E-002-030.

Conventional gate sizing for area and power minimization can have negative impact on yield as follows. Conventional sizing methods minimize the area and power consumption of a circuit by reducing the gate sizes while satisfying the path delay constraints. As a result, the number of combinational paths, that are close to the critical delay, increases. This, in turn, changes the path delay distributions and increase the cumulative timing violation probability.

More recently, clock scheduling has been proposed as a common performance optimization technique for sequential circuits [3, 4]. However, higher clock frequency also results in more timing critical paths and higher probability of timing failure. Several attempts [5, 6] were made to improve the timing yield by finding a clock schedule that minimizes the number of paths with small slacks. None of these works take statistical timing information into consideration and their results are not verified against any yield model.

It is well known that a large portion of the structural paths in a common circuit are functionally unsensitizable (false) paths [7]. Previous works on gate sizing and clock scheduling also do not consider the impacts of false paths and their results may not be optimal in terms of yield.

The rest of the paper is organized as follows. In section 2, we propose a high level design flow that combines a statistical-timing enhanced clock scheduling algorithm and a new false-path-aware gate sizing algorithm. The gate sizing algorithm is formulated and the heuristics for applying it to sequential circuits are proposed in section 3. In section 4, we propose a statistical-timing-driven clock scheduling algorithm. Section 5 shows the benefits of the proposed design flow for area and yield on ISCAS89 and ITC99 benchmark circuits, and section 6 concludes this work.

2. High Level Design Flow

A typical design flow of a sequential circuit consists of three stages, namely architecture definition, design implementation, and design optimization. In the first stage, the architecture is defined and each functional unit is allocated

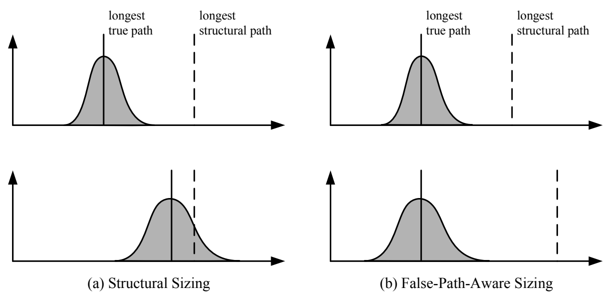
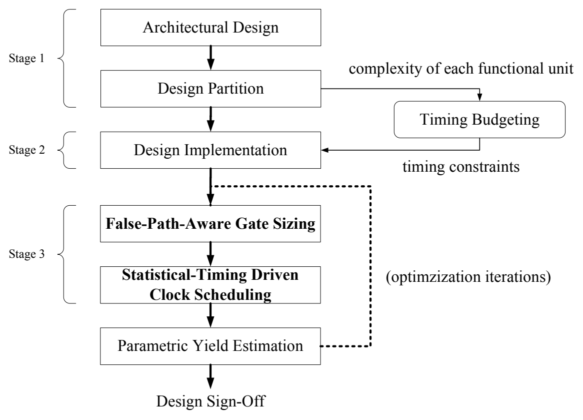


Figure 1. Path delay distributions before and after (a) traditional gate sizing, and (b) false-path-aware gate sizing.



Stage 1
Architectural Design
Design Partition
complexity of each functional unit
Timing Budgeting
Stage 2
Design Implementation
timing constraints
Stage 3
False-Path-Aware Gate Sizing
Statistical-Timing Driven Clock Scheduling
(optimzization iterations)
Parametric Yield Estimation
Design Sign-Off

Figure 2. High level design flow

a timing budget based on its complexity. A clock schedule is determined based on the timing budgets of all units. In the second stage, an initial design of each unit is completed while keeping in mind that it meets its timing budget. In the third stage, area and power optimization techniques, such as gate sizing, are performed on each unit without increasing its longest path delay.

In the traditional design flow, often only structural paths are paid attention to in stages 2 and 3. However, gate sizing without considering false paths can have serious impact on performance and timing yield. Figure 1(a) shows the path delay distribution for a circuit before and after the gate sizing in a conventional design flow. When the longest true path is shorter than the longest structural path, a traditional gate sizing algorithm will size down the gates (or skip repeater insertions) on the true paths to reduce area and power. This can cause the true path delay distribution to shift to the right. For high performance designs this will cause a significant performance degradation and will require serious manual ramifications to fix the timing problems. However, false-path-aware gate sizing will not increase the longest true path delay as illustrated in figure 1(b). Thus, preventing the costly engineering-change-orders (ECOs). Therefore, false paths must be identified and considered early on in the gate sizing step, rather than during the timing analysis step.

Also, traditional design flows do not utilize the statistical timing information to determine the clock schedule. To maximize the timing yield, the clock schedule should be determined based on the statistical timing information. In particular the slacks should be distributed to each path according to its delay uncertainty. Since gate sizing changes the path delay distributions, an optimization method for sequential circuits must perform statistical timing analysis and clock scheduling in each gate sizing iteration for yield estimation. However, false-path-aware statistical timing analysis and clock scheduling can be a bottle neck if used in the inner loop of the optimization.

The design flow proposed in Figure 2 breaks the optimization into two separate steps. The differences between our flow and a traditional flow are that we use a false-path-aware gate sizing algorithm, and the clock schedule is determined after gate sizing is completed thus avoiding costly iterations of the processes. Details of false-path-aware gate sizing and statistical-timing-driven clock scheduling are provided in sections 3 and 4, respectively.

3. False-Path-Aware Gate Sizing

In this section, we describe the false-path-aware gate sizing method for sequential circuits. Our formulation is based on the work by Chen et al. [8], which utilizes the Lagrangian relaxation method. Heuristics to apply false-path-aware gate sizing on sequential circuits are also proposed. Finally, we describe a linearized formulation that is used in this paper for false path aware gate sizing.

3.1. Problem formulation

Let d_i be the delay of gate i and assume the gate size is inversely proportional to the gate delay, the false-path-aware gate sizing problem can be formulated as follows.

$$(\mathcal{GS}) \quad min. \quad \sum_{i=1}^{n} \frac{1}{d_i} \tag{1}$$

$$s.t. \quad \sum_{i \in p} d_i \leq A_0 \quad \forall p \in P - F \tag{2}$$

$$L_i \leq d_i \leq U_i \quad i = 1 \dots n \tag{3}$$

Where A_0 is the target delay, P is a set of all possible paths, and F is a set of all false paths. The constraint (3) limits the lower and upper bound of the gate delays. By solving the above optimization problem, the gate delay d_i will be increased until it reaches U_i, or the delay of some path passing through gate i reaches A_0.

Consider the circuit in figure 3 with two false paths, $< 8, 7, 5, 3, 2, 1 >$ and $< 9, 7, 5, 3, 2, 1 >$. Let $L_i = 1, i = 1 \dots 7$, and $A_0 = 5$, a conventional sizing method will assign a delay of 1 to all the gates on $< 7, 5, 3, 2, 1 >$. However, if the two false paths are identified, we can remove the

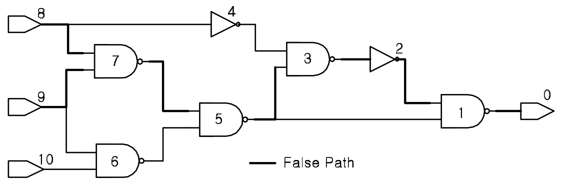


Figure 3. Circuit with false path

corresponding path delay constraints from (2). Note that after removing the constraints on false paths, gate 7 can have longer delay (smaller gate size) since the path $< 7, 5, 1 >$ is the only sensitizable path through gate 7.

False path can be identified by sensitization criteria [9]. To verify the sensitizability of a given path, transitions on the target paths are assigned and maximal implication is performed on the circuit. If the on-path signal has transition from non-controlling values to controlling values, all the off path signals(side inputs) for that node must have non-controlling value. If maximal implication of such assignment has conflict, the target path is considered as false path since the on-path signal cannot be propagated.

3.2. Gate sizing for sequential circuits

In a sequential circuit with clock scheduling, the maximum feasible path delay for the paths between a pair of flip-flops depends on the clock schedule. Moreover, the total number of path delay constraints for a large sequential circuit can be prohibitive. To address these two issues, we propose two heuristic based solutions that lead to optimal or near optimal solution. We also discuss implementation problem and the solution adopted by us in this section.

3.2.1. Pair-wise sizing We provide a divide-and-conquer heuristic to apply $\mathcal{GS}$ to sequential circuits. First, $\mathcal{GS}$ is applied to each pair of FFs separately. For each pair of FFs, the fan-out cone of PI and the fan-in cone of PO are extracted, and only the gates that are in the cone intersection are targeted for sizing. This approach is desired especially when the timing budget between each pair of FFs are different due to clock scheduling. Since a gate can belong to the cone intersections of multiple FF pairs, more than one gate delay could be assigned for the same gate. In this case, the minimum delay will be assigned to avoid creating paths with setup time violations. Although this heuristic is sub-optimal, experimental results show that it still generate relatively good results.

3.2.2. Longest K-path based sizing From the experiments we find that most of the benchmark circuits have reasonable number of true paths. For these circuits, $\mathcal{GS}$ can be directly applied for gate sizing. However, some pairs of FFs do have exponential number of true paths in their cone intersections. In those cases, only the K longest true paths are extracted. Since this does not guarantee that the setup time constraints for all the true paths will be satisfied, the delay of the longest true path after sizing is checked for setup time violation. If any timing violation exists, K is increased and the circuit is re-sized.

3.2.3. Linearized implementation for gate sizing Thus, sizing implementation has four levels of computational complexities: 1) determination of false paths, 2) sizing considering all possible pairs, 3) sizing while meeting constraints for all possible true paths, and 4) non-linearity of the optimization problem. While implementing the sizing method in this paper, we identify false paths using the sensitization criteria and maximum implications described in section 3.1. We indeed consider all possible pairs of FFs for gate sizing but we limit the value of K (longest true paths) to be less than 1000. We find this value of K to be sufficient to generate feasible sizings and keep the problem size small. Finally, instead of solving the non-linear optimization problem $\mathcal{GS}$ that minimizes $\sum_{i=1}^{n} \frac{1}{d_i}$, we convert it to linear optimization that maximized the function $\sum_{i=1}^{n} d_i$. Note that this change is made only to contain the computational complexity.

4. Statistical Timing and Clock Scheduling

Clock scheduling determines the slacks of each path and hence the timing yield. To improve the timing yield, slacks should be assigned to each path according to its timing uncertainty. To achieve this, we first need to obtain the statistical longest and shortest path delay information. Since both gate sizing and false paths can affect the path delay distributions, it is necessary to perform false-path-aware statistical timing analysis after gate sizing.

The longest (shortest) path delay distribution is largely determined by the first few structural longest (shortest) true paths. Therefore, we combine the path sensitizability algorithm in section 3.1 with a depth-first path enumeration algorithm to efficiently find the K-longest (shortest) true paths. Our path enumeration algorithm starts from PI to PO and uses the expected longest (shortest) delay from a node to PO to guide the depth-first-search. A dynamic delay path tree (DPT) is used to record the sensitizable and un-sensitizable partial path information. Once a partial path is found un-sensitizable, all its descendants will not be enumerated. Therefore, our algorithm enumerates the true paths according to their path delays without explicitly listing all structural paths. The path information is then fed to a path-based statistical timer to obtain the mean values $(\mu(\cdot))$ and standard deviations $(\sigma(\cdot))$ of the longest and shortest path delays, D_{ij} and d_{ij}, between FF_i and FF_j. With the statistical information, the constraint formulation is as follows. Let T_i and T_j be the clock arrival times of FF_i and FF_j, respectively, and let CP be the clock period. Further, let the hold-time and setup-time slacks be λ_h and λ_s. The the hold-

Circuits	Traditional sizing + Prop			F.-P.-A. sizing + fp-Prop		
	Yield	Area	Ratio	Yield	Area	Ratio
b05s	**0.319**	617	71.5%	**0.999**	663	76.7%
b06	0.557	41	95.6%	0.557	41	95.6%
b07s	0.675	306	84.6%	0.677	306	84.6%
b10	0.628	145	93.4%	0.628	145	93.4%
b11s	0.716	339	77.6%	**0.958**	359	82.1%
b12	0.659	808	89.4%	**0.762**	815	90.1%
b13s	**0.269**	237	89.0%	0.269	237	89.0%
s1488	**0.512**	555	84.9%	0.512	555	84.9%
s5378	**0.429**	2716	97.7%	0.429	2716	97.7%
s9234.1	**0.493**	4837	86.4%	0.493	4837	86.4%
s15850.1	0.922	9086	93.0%	0.946	9190	94.0%
s35932	**0.440**	12929	80.5%	**0.923**	13118	81.7%
s38417	**0.554**	18765	84.6%	**0.811**	18789	84.7%
s38584.1	0.849	17672	91.8%	0.849	17677	91.8%

Table 1. Comparisons on timing yield.

time constraint is:

$$T_i + \mu(d_{ij}) > T_j + \sigma(d_{ij})\lambda_h. \qquad (4)$$

The setup-time constraint is:

$$T_i + \mu(D_{ij}) + \sigma(D_{ij})\lambda_s < T_j + CP. \qquad (5)$$

Note in the above the hold-time and setup-time slacks, λ_h and λ_s, are normalized with the path delay uncertainties, and these should be maximized to maximize the timing yield. After optimization, paths with larger path delay uncertainties will obtain larger absolute slacks ($\sigma \cdot \lambda$). The constraints (4) (5) are linear and the optimization problem can be solved using an efficient algorithm [6] with slight modifications. We refer to the traditional clock scheduling scheme ($\sigma \triangleq 1$) as *Prop* and our scheme as *fp-Prop*.

5. Experimental Results

We applied our design flow on two sequential benchmark suites: ISCAS89 and ITC99. Timing yields are obtained by Monte Carlo simulations that consider only true path delays to reflect the true timing yields. Instead of modeling process variations, such as channel length and threshold voltage variations, separately, we assume they all manifest themselves into gate delay variations, and all logic gates have an initial gate delay as a Gaussian distribution, (μ, σ), of $(1, 0.15)$. Path delay uncertainties are approximated by $\sqrt{n}\,\sigma$, where n is the number of gates on a path.

Table 1 shows the results using traditional gate sizing followed by *Prop* clock scheduling, and using false-path-aware gate sizing and *fp-Prop* clock scheduling. In this table all columns are self explanatory except the columns marked "Ratio" which provide the ratios of circuit area after gate sizing as compared to the circuit area before gate sizing. We can draw the following three major conclusions:

1. The design flow proposed by us in this paper achieves significantly better timing yields ($> 20\%$) than the traditional flow for five benchmark circuits with less than 5% area overhead.
2. Traditional gate sizing has serious (adverse) timing yield impact on the circuits b05s, s35932 and s38417, due to the effects illustrated in figure 1. Our new design flow only has little or no adverse impact on the timing yields of these circuits.
3. For b13s, s1488, s5378 and s9234.1, both the traditional flow and our flow cause significant timing yield degradations. This can be improved by performing the gate sizing and clock scheduling iteratively.

6. Conclusion

We propose to improve the timing yields by performing false-path-aware gate sizing and statistical-timing-driven clock scheduling. Experimental results demonstrate the versatility of our algorithms. Our design flow can also be used in an iterative optimization framework for further timing yield improvements.

References

[1] Chandu Visweswariah. Death, taxes and failing chips. In *Proceedings of the 40th conference on Design automation*, pages 343–347, 2003.

[2] Yutaka Deguchi, Nagisa Ishiura, and Shuzo Yajima. Probabilistic CTSS: analysis of timing error probability in asynchronous logic circuits. In *Proceedings of the 28th conference on ACM/IEEE design automation conference*, pages 650–655, 1991.

[3] John P. Fishburn. Clock skew optimization. *IEEE Transactions on Computers*, 39(7):945–951, July 1990.

[4] Rahul B. Deokar and Sachin S. Sapatnekar. A graph-theoretic approach to clock skew optimization. In *Proceedings of the 1994 IEEE International Symposium on Circuits and Systems*, volume 1, pages 407–410, May 1995.

[5] Ivan S. Kourtev and Eby G. Friedman. Clock skew scheduling for improved reliability via quadratic programming. In *Proceedings of the 1999 IEEE/ACM international conference on Computer-aided design*, pages 239–243, 1999.

[6] Stephan Held, Bernhard Korte, Jens Maberg, Matthias Ringe, and J. Vygen. Clock scheduling and clocktree construction for high performance asics. In *Proceedings of the 2003 IEEE/ACM international conference on Computer-aided design*, pages 232–239, 2003.

[7] Karl Fuchs, Franz Fink, and Michael H. Schulz. DYNAMITE: An efficient automatic test pattern generation system for path delay faults. *IEEE Transactions on Computer-aided design*, 10(10):1323–1335, Oct 1991.

[8] Chung-Ping Chen, Chris C. N. Chu, and D. F. Wong. Fast and exact simultaneous gate and wire sizing by lagrangian relaxation. In *Proceedings of the 1998 IEEE/ACM international conference on Computer-aided design*, pages 617–624, 1998.

[9] Seiji Kajihara, Kozo Kinoshita, Irith Pomeranz, and Sudhakar M. Reddy. A method for identifying robust dependent and functionally unsensitizable paths. In *10th International Conference on VLSI Design*, pages 82–87, Jan 1996.