A Statistical Approach to the Timing-Yield Optimization of Pipeline Circuits

Chin-Hsiung Hsu[†], Szu-Jui Chou[†], Jie-Hong R. Jiang[§†], and Yao-Wen Chang[§†]

Department of Electrical Engineering[§]/Graduate Institute of Electronics Engineering[†]
National Taiwan University, Taipei 10617, Taiwan
{arious, rerechou}@eda.ee.ntu.edu.tw; {jhjiang, ywchang}@cc.ee.ntu.edu.tw

Abstract. The continuous miniaturization of semiconductor devices imposes serious threats to design robustness against process variations and environmental fluctuations. Modern circuit designs may suffer from design uncertainties, unpredictable in the design phase or even after manufacturing. This paper presents an optimization technique to make pipeline circuits robust against delay variations and thus maximize timing yield. By trading larger flip-flops for smaller latches, the proposed approach can be used as a post-synthesis or post-layout optimization tool, allowing accurate timing information to be available. Experimental results show an average of 31% timing yield improvement for pipeline circuits. They suggest that our method is promising for high-speed designs and is capable of tolerating clock variations.

1 Introduction

As the semiconductor fabrication technology advances to the sub-100nm feature size regime, sensitivities of IC designs to process variations and environmental fluctuations are ever-increasing. To maintain design robustness against these uncertainties, it becomes more and more apparent that traditional design methodologies need to be modified and consider variations at the early stage of a design flow since not all process variations can be diminished with technology advances after all.

In recent years, statistical approaches to circuit analysis and optimization have been revolutionizing the EDA community. They are mostly centered around delay and power issues, the two main concerns affected by design uncertainties. In this paper, we focus on the timing issue. Traditional approaches to timing optimization were based on worst-case analysis. For instance, any gate delay under a certain operation condition may be set as a deterministic value fixed at the 3σ point in statistics to ensure enough margin tolerating variations. However, worst-case analysis is too conservative especially for more and more stringent design constraints in timing. Furthermore, when designs become more sensitive to process variations, it is harder to make design safe under worst-case variations. Due to the inadequacy of traditional worst-case analysis, the need of statistical analysis emerges, and has attracted intensive research efforts. Statistical optimization is the next step as statistical analysis is getting mature.

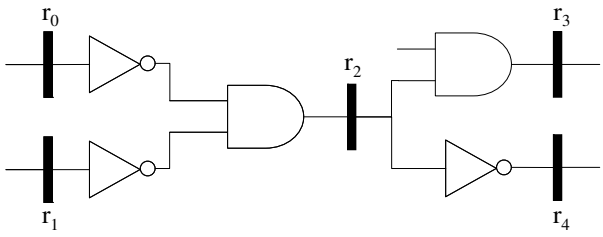

Fig. 1. A motivating example for timing yield improvement by replacing DFFs with latches.

Based on statistical timing analysis, most existing statistical optimization approaches focused on gate sizing, e.g., [4–6, 11], and clock skew scheduling, e.g., [1, 7, 10, 14]. Rather, we propose a new statistical optimization methodology, which is orthogonal and complementary to gate sizing and can possibly be combined with clock scheduling for further improvement. We take advantage of the transparency property of level-sensitive latches for tolerating delay uncertainties. In fact, there were prior efforts focusing on the tradeoff between flip-flops and latches in other optimization context. For instance, flip-flops may be replaced with latches to optimize storage [16] or power [8]. However, to the best of our knowledge, there was no work done in the context of optimizing timing yield in the statistical domain. Consider Figure 1 for a motivating example. In the circuit, assume the delays (in nanoseconds) of an AND gate and a NOT gate, and a wire are in normal distributions $N(5, 1)$, $N(3, 1)$, and $N(0, 0)$, respectively. (That is, we neglect the wire delay and assume that the AND- and NOT-gate delays are of mean values 5 and 3, respectively, and are of the same variance 1.) Suppose the clock period is 8ns. By Monte Carlo simulation, the timing yield of the circuit with all positive-edge triggered D-type flip-flop (D-FF) registers is 33.19%; after replacing r_2 with an active-high latch, the yield increases to 93.02%. A nearly 60% improvement is achieved by replacing a D-FF with a latch. Note that in this replacement the number of pipeline stages remains unchanged.

Given a design with edge-triggered D-FF implementation of state-holding elements (i.e. registers), we substitute level-sensitive latches for D-FFs such that timing yield is maximally improved. In addition, this substitution also enhances the tolerance to clock skew uncertainty as was known in the timing community. Based on dynamic programming, we devise an optimal algorithm for pipelined circuits, and generalize it for arbitrary sequential circuits. The proposed method can be used for pre-layout optimization under a statistical model of design uncertainties. Moreover, because latches are of smaller sizes compared with D-FFs, the substitution is possible without affecting nearby circuit structures and thus can be performed even after physical design. Thereby, accurate timing information may be used. In contrast, yield improvement by gate sizing may invalidate

prior physical design when devices are sized up, and thus may suffer from the design closure problem.

Why is latch substitution challenging? Firstly, statistical timing analysis for latch-based design is itself tricky compared with those for combinational designs and D-FF based sequential designs [3]. Secondly, aside from the timing analysis issue, for optimization there are an exponential number of register configurations to be explored. Essentially, each register can be of type $\mathcal{D}$ (standing for a D-FF), $\mathcal{H}$ (an active-high latch), or $\mathcal{L}$ (an active-low latch). Thus, for a design with n registers, there are 3^n possible configurations, each of them requiring the above analysis to determine its timing yield. Despite these challenges, there exist effective approaches to the latch substitution problem. We organize our explanation as follows. Section 2 gives some preliminaries of our models and the underlying timing analysis. Section 3 analyzes the effect of substituting latches for D-FFs, and formalizes our optimization objectives. Section 4 presents our algorithms, which are evaluated with experimental results in Section 5. Finally, concluding remarks are given in Section 6.

2 Preliminaries

2.1 Statistical Timing Models and Analysis

To simplify our exposition, in our discussion we shall assume that gates are the main delay sources. However, wire delays as well can be taken into account straightforwardly. Using the model of [15], global and local variations as well as correlations can be handled. By statistical static timing analysis (SSTA), the input-to-output delay distributions of a combinational block in a sequential circuit can be obtained. Thus we may compute the *longest combinational-path delay distribution* $\Delta(r_i, r_j)$ (resp. *shortest combinational-path delay distribution* $\delta(r_i, r_j)$) from register r_i to register r_j by Gaussian-approximating max [2] (resp. min) and sum operations over Gaussian random variables.[1] While $\delta(r_i, r_j)$ is immaterial in combinational timing analysis, it is crucial in analyzing sequential circuits involving latches. Note that $\Delta(r_i, r_j)$ (similarly $\delta(r_i, r_j)$) is not a distribution for some single fixed path, rather it may probabilistically correspond to different paths.

2.2 Timing Yield of Sequential Circuits

Let $T = T_H + T_L$ be the clock period with high interval T_H and low interval T_L. Given a design with some target operation speed, its timing yield is the probability that no violation occurs with respect to timing constraints, see e.g.

[1] For circuits with pure D-FF registers, analyzing register-to-register delays may seem far from necessary. In fact, computing the longest delay of every combinational block is enough. However, for circuits containing latches, computing register-to-register delays is necessary due to the transparency of latches making combinational blocks not well separable for timing analysis.

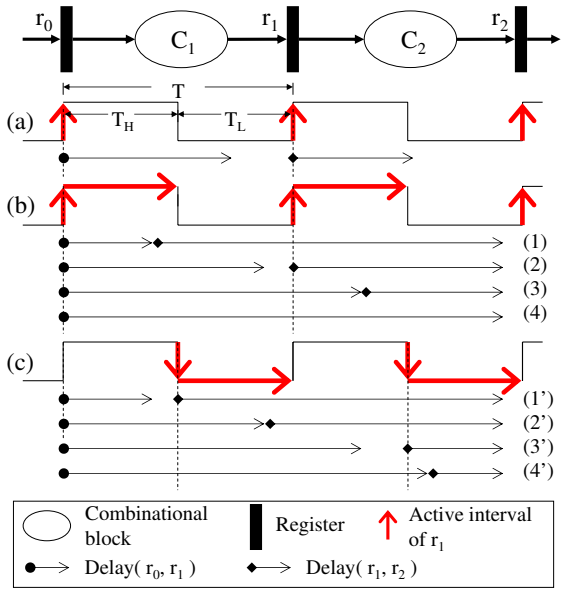

Fig. 2. A single-path pipelined circuit and timing diagrams. (a) $type(r_0) = type(r_1) = type(r_2) = \mathcal{D}$; (b) $type(r_1) = \mathcal{H}$ and $type(r_0) = type(r_2) = \mathcal{D}$; (c) $type(r_1) = \mathcal{L}$ and $type(r_0) = type(r_2) = \mathcal{D}$

.

[3]. In the simplest case, when a circuit is implemented with D-FFs for all of its registers, its timing yield is the probability

$$\Pr[\bigwedge_{(r_i,r_j)} (\Delta(r_i, r_j) \leq T)], \tag{1}$$

for any register pair (r_i, r_j) with a combinational path from r_i to r_j. For example in Figure 2 (a), where registers r_0, r_1, r_2 are of type $\mathcal{D}$, then the yield is

$$\Pr[(\Delta(r_0, r_1) \leq T) \wedge (\Delta(r_1, r_2) \leq T)]. \tag{2}$$

3 Timing Yield and Register Configuration

3.1 Timing Yield Changed by Latch Replacement

We study the effects of substituting latches for D-FFs. To begin with, consider the single-path pipelined circuit of Figure 2. Intuitively, an active-high latch can tolerate longer delay of its fan-in combinational block than a D-FF. If the type of r_1 is changed to $\mathcal{H}$ as shown in Figure 2 (b), the longest delay of combinational block C_1 can exceed T. For a circuit to operate without any timing violation, essentially four cases need to be analyzed depending on $\Delta(r_0, r_1)$:

case 1 $0 \leq \Delta(r_0, r_1) < T_H$: The signal of C_1 arrives r_1 within the active interval and can directly pass to C_2; so $T < \Delta(r_0, r_1) + \Delta(r_1, r_2) \leq 2T$ must hold. In addition, C_2 must satisfy $T < \delta(r_0, r_1) + \delta(r_1, r_2) \leq 2T$ for r_2 to latch the right value.

case 2 $T_H \leq \Delta(r_0, r_1) < T$: The signal of C_1 arrives r_1 before r_1 is turned on; so it must wait until r_1 is active again at T. C_2 must satisfy $\Delta(r_1, r_2) \leq T$. In addition, C_1 must satisfy $\delta(r_0, r_1) > T_H$; otherwise the earliest and latest signals of C_1 arrive C_2 in different clock cycles.

case 3 $T \leq \Delta(r_0, r_1) < T + T_H$: The delay of C_1 is in the active interval of r_1 and can directly pass to C_2; so $T < \Delta(r_0, r_1) + \Delta(r_1, r_2) \leq 2T$ must hold. Also, $\delta(r_0, r_1) > T_H$ must hold for the same reason as case 2.

case 4 $T + T_H \leq \Delta(r_0, r_1) < 2T$: The signal of C_1 cannot pass through r_1 in $2T$; so this case is forbidden.

Although case 1 incurs no timing violation in this example, it is problematic if r_1 has a designated initial value (which will be erased) or r_1 fans out to a primary output since then the number of pipeline stages seen from the output is different. We exclude it from our yield calculation and consider only legal cases 2 and 3. For these two cases, the delay between r_0 and r_1 is restricted to $T_H \leq \Delta(r_0, r_1) < T + T_H$ and $\delta(r_0, r_1) > T_H$, while the delay between r_1 and r_2 is restricted to $\max\{\Delta(r_0, r_1), T\} + \Delta(r_1, r_2) \leq 2T$, where $\max\{\Delta(r_0, r_1), T\}$ equals T in case 2 and $\Delta(r_0, r_1)$ in case 3, respectively. Thus, the yield equals

$$
\begin{aligned}
&\Pr[\text{case 2}] + \Pr[\text{case 3}] \\
&= \Pr[(T_H \leq \Delta(r_0, r_1) < T) \wedge (\delta(r_0, r_1) > T_H) \wedge (\Delta(r_1, r_2) \leq T)] + \\
&\quad \Pr[(T \leq \Delta(r_0, r_1) < T + T_H) \wedge (\delta(r_0, r_1) > T_H) \wedge \\
&\quad (T < \Delta(r_0, r_1) + \Delta(r_1, r_2) \leq 2T)] \qquad (3) \\
&= \Pr[(T_H \leq \Delta(r_0, r_1) < T + T_H) \wedge (\delta(r_0, r_1) > T_H) \wedge \\
&\quad (\max\{\Delta(r_0, r_1), T\} + \Delta(r_1, r_2) \leq 2T)]. \qquad (4)
\end{aligned}
$$

In contrast, if the type of r_1 is changed to $\mathcal{L}$ as shown in Figure 2 (c), four cases similar to the above ones need to be analyzed depending on $\Delta(r_0, r_1)$, which we omit due to limited space. The analysis forms the basis of our yield calculation. It can be extended to the analysis of pipeline circuits since every pair of adjacent registers can be transform into a circuit as in Figure 2.

In computing the timing yield of a pipeline circuit, the timing constraints of a combinational block depend on the types of its preceding registers, which leads to complex computation especially for latches. Due to the transparency of latches, delay distributions need to be propagated across latches. For example, $\Delta(r_0, r_1)$ is needed in Equation (4) in calculating the yield between registers r_1 and r_2. (For D-FF based designs, there is no need to propagate distribution across register boundaries since the output of a D-FF has zero arrival time.) To resolve this complication, we shift the delay distribution of a combinational block to make the equations for the three types of registers identical. That is, we modify the delay distribution of a register input and pass it as a slack to

the fan-out blocks. Thereby we may propagate probability distributions across latches. Precisely speaking, for active-high latches, by defining

$$\Delta_{\text{shift}}(r_0, r_1) \equiv \Delta(r_0, r_1) - T_H, \delta_{\text{shift}}(r_0, r_1) \equiv \delta(r_0, r_1) - T_H,$$
$$\Delta_{\text{shift}}(r_1, r_2) \equiv \max\{\Delta(r_0, r_1) - T, 0\} + \Delta(r_1, r_2), \text{ and} \delta_{\text{shift}}(r_1, r_2) \equiv \delta(r_1, r_2),$$

Equation (4) can be rewritten as

$$\begin{aligned} &\Pr[\text{case 2}] + \Pr[\text{case 3}] \\ &= \Pr[(\Delta_{\text{shift}}(r_0, r_1) < T) \wedge (\delta_{\text{shift}}(r_0, r_1) > 0) \wedge \\ &\quad (\Delta_{\text{shift}}(r_1, r_2) < T) \wedge (\delta_{\text{shift}}(r_1, r_2) > 0)]. \end{aligned} \tag{5}$$

For active-low latches, similar rewriting is also available, which we omit due to limited space. For D-FFs, on the other hand, no shifting is needed.

With the above distribution shifted, we make all longest delay constraints compared with T and shortest delay constraints compared with 0 as in Equations (5). Finally, for any register pair r_i and r_j connected by a combinational block under analysis, we perform the max operation over $\{\Delta_{\text{shift}}(r_i, r_j)\}$ and min operation over $\{\delta_{\text{shift}}(r_i, r_j)\}$, and obtain the probability of the combinational block without timing violation by

$$\Pr[(\max\{\Delta_{\text{shift}}(r_i, r_j)\} < T) \wedge (\min\{\delta_{\text{shift}}(r_i, r_j)\} > 0)].$$

3.2 Problem Formulation

Definition 1. *Let R be a nonempty set of registers of a sequential circuit. A* register configuration *of R is a total function $\rho : R \to \{\mathcal{D}, \mathcal{H}, \mathcal{L}\}$.*

D-FFs are the most common implementation of state-holding elements of sequential circuits due to their simple edge-triggered timing constraints. We assume that a given design is in D-FF implementation initially. By changing the initial register configuration, a circuit can be made more insensitive to timing variations while maintaining its behavior. Essentially, pipeline stages should not be changed before and after modifying register configurations. Therefore, no two latches of the same type can be connected by a combinational path. Furthermore, even two latches of different types cannot be connected by a combinational path because the number of pipeline stages will decrease if the total number of registers cannot increase. Hence we require that the fan-in and fan-out registers of a latch have to be of type $\mathcal{D}$. (Note that a positive-edge triggered D-FF can be decomposed into an active-low latch followed by an active-high latch. So it is possible to maintain pipeline stages by increasing the register count, which we disallow in this paper.)

The optimization problem can be stated as follows.

Yield optimization problem: Given a sequential circuit with $\rho(r) = \mathcal{D}$, for any register r, and the distributions of its gate and wire delays, find the register configuration such that timing yield is maximally improved subject to the above replacement criterion.

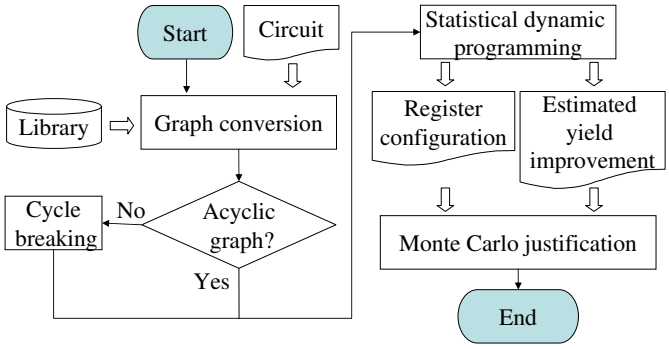

Fig. 3. The flowchart of statistical latch replacement.

4 Statistical Latch Replacement

4.1 Optimization Flow Overview

The flow of our algorithm is shown in Figure 3. Firstly, the input circuit is abstracted and converted to a register dependency graph with statistical timing models and analysis to abstract essential timing information. Secondly, all cycles of the register dependency graph are made acyclic with respect to a chosen minimal feedback vertex set. Thirdly, the resultant acyclic graph is levelized in topological order from inputs to outputs. Fourthly, our statistical dynamic programming algorithm is conducted forwardly over the levelized acyclic graph. The optimal configuration can then be derived by tracing backward from outputs to inputs. Finally, Monte Carlo simulation can optionally be applied to justify the yield improvement.

4.2 Statistical Dynamic Programming

We abstract a given input circuit C with a *register dependency graph* $G = (V, E)$, where a vertex $v_i \in V$ represents a register r_i in C and there is an directed edge $(v_i, v_j) \in E$ if and only if there is a combinational path from r_i to r_j in C. Also, register-to-register distributions $\Delta(r_i, r_j)$ and $\delta(r_i, r_j)$ are computed according to the delay distributions of C, and is associated to its corresponding edge $(v_i, v_j) \in E$. If a circuit has feedback, there will be cycles in the converted graph. In order to levelize the register dependency graph, we break all cycles by finding a minimal feedback vertex set (FVS) [9]. After making a register dependency graph acyclic, we levelize it in a topological order such that each vertex is labelled with the longest distance from an input vertex. Given an levelized acyclic register dependency graph, we derive a register configuration with maximal timing yield by the statistical dynamic programming algorithm outlined in Figure 4.

```
Algorithm: StatisticalDynamicProgramming
Input:    levelized register dependency graph
          G = (V, E) and delay distributions on E
Output:   optimal register configuration for yield
begin
 01 set level-1 registers to D-FFs with local yield 1
 02   ℓ := LevelCount(G)
 03   for i = 2, ..., ℓ
 04     let R_i be the set of registers at level-i
 05     for every register configuration α of R_i
 06       compute the highest local yield Y_α of α
             subject to the configurations of R_{i-1}
             and their local yields
 07       record the config. of R_{i-1} responsible for Y_α
 08   set R_ℓ to the config. β_ℓ of all D-FFs
 09   for i := ℓ − 1, ℓ − 2, ..., 2
 10     set R_i to the config. β_i responsible for β_{i+1}
 11   return β's
end
```

Fig. 4. The Statistical Dynamic Programming Algorithm.

We add artificial D-FFs at the primary inputs and outputs when converting a circuit to a register dependency graph. Hence we set level-1 and level-ℓ registers to be of type $\mathcal{D}$, where ℓ is the number of levels in the levelized register dependency graph. In addition, we define the *local yield* of a register to be the accumulated yield computed forward from level-1 registers, each having local yield 1. The statistical dynamic programming algorithm computes and stores the optimal configurations and the corresponding local yields in a forward direction based on the timing analysis introduced in Section 3.

Take a single-path pipelined circuit as an example. The statistical dynamic programming algorithm proceeds in two phases as shown in Figure 5 (a) and (b). In the first phase, three configurations $\{\mathcal{D}, \mathcal{H}, \mathcal{L}\}$ are considered for each register in a forward direction. Since we require that the fan-in and fan-out registers of a latch need to have type $\mathcal{D}$, only a subset of two consecutive configurations need to be considered as indicated by the arrows of Figure 5 (a). At each level, the maximal local yield is kept for each configuration of $\{\mathcal{D}, \mathcal{H}, \mathcal{L}\}$. Once the final level is reached, the algorithm enters the second phase. It extracts the optimal configuration for each register backward.

For a register dependency graph with large pipeline widths, the above algorithm becomes inefficient (in fact, exponential complexity in the pipeline width) since it considers all possible configurations for registers at each level. We alleviate this problem by greedily optimizing one register at a time without considering

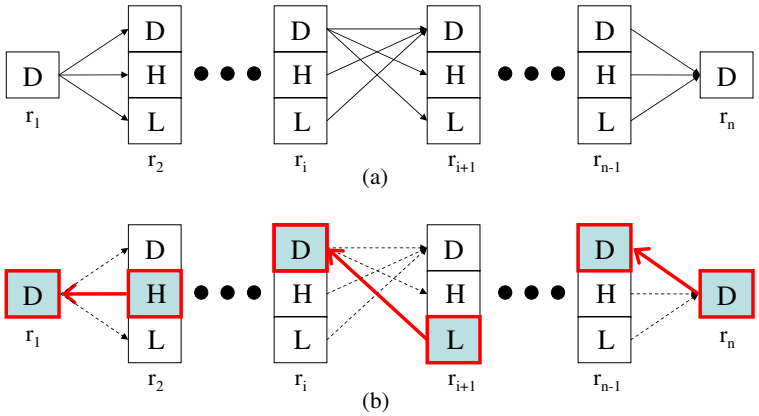

Fig. 5. Statistical dynamic programming for a single-path pipelined circuit optimization. (a) Forward yield calculation. Only feasible edges are shown. (b) Backward tracing the optimal configuration.

the configurations of other registers at the same level. Thus, we may need to handle the consistency problem for conflicting register type assignments. Note that because we only consider one register at a time, the result may differ from the global optimum. It is a tradeoff between optimality and efficiency.

5 Experimental Results

The proposed algorithm is implemented in C++ codes. The experiments were conducted on a Linux machine with Pentium IV 3.2GHz CPU and 3GB memory. Two sets of circuits are used: pipeline circuits and general sequential circuits all from ISCAS benchmark suites. The pipeline circuits were generated from combinational circuits by adding 4-stage pipelines. For a given circuit, under the SIS [13] environment, technology mapping was conducted to obtain delay information and then minimum-period retiming was performed (thus registers were relocated evenly over the circuit). In addition, the circuits were synthesized to balance long and short combinational paths. (Note that, in high-speed and/or low-power designs, long and short paths tend to be balanced. For instance, performance-driven logic optimization and power optimization with dual threshold voltage assignments tend to balance long and short delays. Thus, design trends meet our timing requirements.) All delay variations are in normal distribution with 10–20% deviation.

Table 1 shows the results for 10% and 20% delay deviations. Columns 1, 2, and 3 show the circuits, numbers of pipeline stages, and numbers of registers, respectively. The clock periods are shown in the 4th column, where the clock period of a circuit is determined by imposing the timing yield of the circuit with all D-FF registers to fall between 60–65%. The numbers of D-FFs replaced by

Table 1. ISCAS benchmark circuits with 10% and 20% delay deviations.

Circuit	# of stages	Total reg.	Clock period		Replaced reg.		Original yield (%)		Final yield (%)		Impv. (%)		CPU time (s)	
			10%	20%	10%	20%	10%	20%	10%	20%	10%	20%	10%	20%
ISCAS85	Pipeline circuits with clock minimization													
c432	5	214	8.13	8.58	18	28	64.4	63.2	100.0	97.2	35.6	34.0	0.21	0.20
c499	5	186	8.65	9.37	13	8	65.0	62.2	100.0	100.0	35.0	37.8	0.11	0.11
c880	5	242	7.36	7.74	14	16	61.5	62.7	67.0	98.7	5.5	36.0	0.14	0.13
c1355	5	218	9.42	10.18	9	10	60.3	62.3	100.0	99.8	39.7	37.5	0.16	0.16
c1908	5	240	13.44	14.26	19	19	62.5	64.0	100.0	98.1	37.5	34.1	0.19	0.19
c3540	5	278	11.14	11.96	78	62	64.0	62.3	94.1	93.9	30.1	31.6	0.42	0.40
c5315	5	867	11.88	12.60	0	0	60.1	61.7	60.1	61.7	0.0	0.0	0.61	0.63
c7552	5	879	11.26	12.12	56	69	63.7	63.5	99.6	99.9	35.9	36.4	0.71	0.68
Average											27.41	30.93	0.284	0.313
ISCAS89	Sequential circuits													
s1196	-	18	50.24	53.54	3	4	62.9	59.7	67.2	62.4	4.3	2.7	0.04	0.05
s5378	-	179	47.79	52.98	10	10	65.2	61.1	71.9	65.2	6.7	4.1	0.44	0.45
s9234	-	211	108.57	118.86	8	8	54.7	57.8	56.0	59.3	1.3	1.5	0.90	0.89
Average											4.10	2.77	0.460	0.463

level-sensitive latches are shown in the fifth column. Columns 6, 7, and 8 list the original, final, and improved timing yields, respectively. The yields are are justified with Monte Carlo simulation. The reported CPU times in the ninth column are without counting the Monte Carlo simulation. Each of Columns 4–9 is divided further into two sub-columns for 10% and 20% delay deviations.

As can be see, the improvements are consistent above 30% for all of the pipeline circuits, except for circuits `c880` and `c5315`. For `c880` in the 10%-deviation case, some inaccuracy in timing analysis causes inadequate latch replacements and degrades the yield improvement. For `c5315`, a similar reason causes inadequate latch replacements, which are later cancelled by the justification of Monte Carlo simulation. This problem can be overcome by using more accurate SSTA tools. Nevertheless, the average improvements of pipeline circuits are 27% and 31% for deviations 10% and 20%, respectively. It suggests that our approach to yield improvement is robust against the changes of delay deviation. It is interesting to note that the numbers of replaced registers for 20%-deviation cases are in general larger than those for 10%-deviation ones as shown in Column 5. It suggests the importance of latch replacements for increased delay deviations. On the other hand, for cyclic sequential circuits, such as `s1196`, `s5378`, and `s9234`, our approach only yields mild improvements. It is understandable because the register dependency graphs of these circuits are close to complete graphs, which makes latch replacement almost impossible.

To see the relation between the clock period and yield improvement, we conduct another experiment over circuit `c1355`. The result is plotted in Figure 6. As can be seen, by reducing the clock period, the yield of the original design with all D-FFs tends to vanish very quickly from 100% to 0% whereas that of the optimized version remains high and stable for another 1 unit of delay. (The glitch in the figure is due to different optimal register configurations for different clock periods.) The result tends to suggest that our latch replacement algorithm is *robust against clock variation*, and *suitable for high-speed designs*. Hence our approaches are promising for yield improvement in the current trend of high-speed designs.

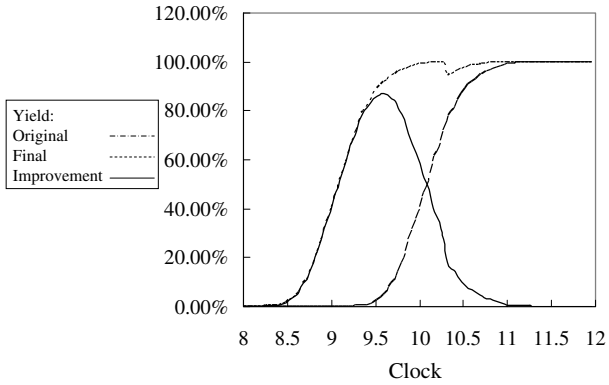

Fig. 6. Experimental results. Yield vs. clock period for circuit `c1355`.

6 Conclusions and Future Work

Based on statistical timing analysis, we have proposed an algorithm to optimize the timing yield of a sequential circuit. Experimental results show that, by substituting latches for D-FFs, timing yield can be improved about 31% on average for pipelined circuits. In addition, the results suggest that latch replacement tends to tolerate clock variations. Complementary to other design-for-yield methodologies like gate sizing and clock skew scheduling, our technique may be combined with these techniques for further improvement. Since most circuits use D-FFs for register implementation, our approach may be widely applicable to standard designs. Since replacing D-FFs with latches incurs no area penalty, the proposed algorithm can be used for not only pre-layout but also post-layout optimization, where accurate timing information is available.

For future work, since our approach only yields mild timing yield improvements to cyclic sequential circuits, some work needs to be done to overcome this limitation. On the other hand, we may consider multiple-phased clocking scheme, which may lead to further yield improvements. Also, setup-time and hold-time constraints may be added in our framework.

Acknowledgments

This work was supported in part by NSC grants 94-2218-E-002-083, 95-2221-E-002-432, and 95-2218-E-002-064-MY3.

References

1. C. Albrecht, B. Korte, J. Schietke, and J. Vygen. Cycle time and slack optimization for VLSI-chips. In *Proc. ICCAD*, pp. 232-238, 1999.

2. C. E. Clark. The greatest of a finite set of random variables. *Operations Research*, vol. 9, no. 2, pp. 145-162, 1961.
3. C.-T. Chao, L.-C. Wang, K.-T. Cheng, and S. Kundu. Static statistical timing analysis for latch-based pipeline designs. In *Proc. ICCAD*, 2004.
4. S.-H. Choi, B. Paul, and K. Roy. Novel sizing algorithm for yield improvement under process variation in nanometer technology. In *Proc. DAC*, 2004.
5. K. Chopra, S. Shah, A. Srivastava, D. Blaauw, and D. Sylvester. Parametric yield maximization using gate sizing based on efficient statistical power and delay gradient computation. In *Proc. ICCAD*, 2005.
6. M. Guthaus, N. Venkateswaran, C. Visweswariah, and V. Zolotov. Gate sizing using incremental parameterized statistical timing analysis. In *Proc. ICCAD*, 2005
7. A. Hurst and R. Brayton. Computing clock skew schedules under normal process variation. In *Proc. IWLS*, 2005.
8. K. Lalgudi and M. Papaefthymiou. Fixed-phase retiming for low power design. In *Proc. ISLPED*, 1996.
9. H.-M. Lin and J.-Y. Jou. On computing the minimum feedback vertex set of a directed graph by contraction operations. *IEEE Trans. on CAD*, vol. 19, no. 3, 2000.
10. J. Neves and E. Friedman. Optimal clock skew scheduling tolerant to process variabtions. In *Proc. DAC*, pp. 623-628, 1996.
11. S. Raj, S. Vrudhula, and J. Wang. A methodology to improve timing yield in the presence of process variations. In *Proc. DAC*, pp. 448-453, 2004.
12. K. Sakallah, T. Mudge, and O. Olukotun. checkT_c and minT_c: Timing verification and optimal clocking of synchronous digital circuits. In *Proc. ICCAD*, pp. 552-555, 1990.
13. E.M. Sentovish et al. SIS: a system for sequential circuit synthesis. *Technical Report UCB/ERL M92/41*, UC Berkeley, 1992.
14. J.-L. Tsai, D. Baik, C.-P. Chen. and K. Saluja. A yield improvement methodology using pre- and post-silicon statistical clock scheduling. In *Proc. ICCAD*, pp.611-618, 2004.
15. C. Vishweswariah, K. Ravindran, K. Kalafala, S. Walker, and S. Narayan. First-order incremental block-based statistical timing analysis. In *Proc. DAC*, pp. 331-226, 2004.
16. T.-Y. Wu and Y.-L. Lin. Storage optimization by replacing some flip-flops with latches. In *Proc. DAC*, 1996.