Clock Skew Scheduling for Timing Speculation

Rong Ye†‡, Feng Yuan†, Hai Zhou§ and Qiang Xu†‡

†CUhk REliable Computing Laboratory (CURE)
Department of Computer Science & Engineering
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
‡Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
Email: {rye,fyuan,qxu}@cse.cuhk.edu.hk

§Department of Electrical Engineering and Computer Science
Northwestern University, USA
Email: {haizhou}@northwestern.edu

***Abstract*—By assigning intentional clock arrival times to the sequential elements in a circuit, clock skew scheduling (CSS) techniques can be utilized to improve IC performance. Existing CSS solutions work in a conservative manner that guarantees "always correct" computation, and hence their effectiveness is greatly challenged by the ever-increasing process variation effects. By allowing infrequent timing errors and recovering from them with minor performance impact, timing speculation techniques such as Razor have gained wide interests from both academia and industry. In this work, we formulate the clock skew scheduling problem for circuits equipped with timing speculation capability and propose a novel CSS algorithm based on gradient-descent method. Experimental results on various benchmark circuits demonstrate the effectiveness of our proposed methodology.**

I. Introduction

Clock skew scheduling (CSS), which treats clock skew as a manageable resource instead of design liability, is an effective technique to improve IC performance, by assigning intentional clock arrival times to internal flip-flops (FFs). Earlier works in this domain (e.g., [1–3, 5]) focused on clock period reduction to maximize IC timing performance. With technology scaling, process variation has become a serious concern for circuit design and the earlier performance-driven CSS solutions may result in low manufacturing yield due to timing uncertainty caused by process variation. Consequently, various yield-driven CSS techniques (e.g., [4, 6, 7]) were developed to maximize timing yield under a certain clock period.

Despite the different design objectives and optimization methods, a common design constraint of existing CSS techniques is that they need to guarantee the timing correctness of the circuit after skew adjustment, even in the worst case scenario. The ever-increasing process variation effects hence pose serious challenges for these techniques since a large timing guard band has to be reserved to tolerate timing uncertainty, leading to rather limited performance improvement room for CSS techniques.

By allowing infrequent timing errors and achieving timing error resilience with error detection and correction techniques, timing speculation techniques such as Razor [8] enable highly energy-efficient "better than worst-case" designs, and hence have attracted lots of attention from both academia and industry [9–14]. For circuits equipped with timing speculation capability, since there is no need to guarantee "always correct" operation in such designs, we can afford to have more aggressive skew optimization strategies to improve circuit performance without necessarily reserving timing guard bands. Recently, a post-silicon clock skew tuning framework [15] has been proposed to manipulate timing slacks of different FFs according to collected timing error information at runtime. However, how to conduct pre-silicon clock skew scheduling at design stage for such timing speculated designs, to the best of our knowledge, has not been studied in the literature yet. The pre-silicon clock skew scheduling work investigated in this paper can be easily combined with the post-silicon clock skew tuning work.

Motivated by the above, in this paper, we first develop a general formulation of CSS problem for circuits with timing speculation capability, wherein timing error rate and its corresponding impact are explicitly considered. We then propose a novel clock skew scheduling algorithm based on gradient-descent method (GDM) to tackle this problem. As shown in experimental results on various benchmark circuits, our proposed technique is able to significantly reduce the overall timing error rate of the circuit, thus dramatically improving its throughput.

The remainder of this paper is organized as follows. In Section II, we present the preliminaries and related works. The problem formulation for CSS with timing speculation explicitly considered and the corresponding GDM-based skew scheduling algorithm are then detailed in Section III and Section IV, respectively. Next, Section V presents our experimental results on various benchmark circuits. Finally, Section VI concludes this paper.

II. Preliminaries and Motivation

A. Background on Clock Skew Scheduling

A synchronous circuit (see Fig. 1 for a simple example) with edge-triggered storage elements (e.g., flip-flops) can usually be modeled as a directed graph $G(V, E)$ as depicted in Fig. 2. In Fig. 1, the squares represent FFs and the circles represent combinational logics. In the timing constraint graph as shown in Fig. 2, each node $(v_i \in V)$ represents a FF and each arc $(e_{ij} \in E)$ represents the longest/shortest path from FF_i to FF_j.

Let s_i be the clock arrival time of v_i, and D_{ij} and d_{ij} be the maximum and minimum path delays of e_{ij} respectively, the setup-time and hold-time constraints of traditional CSS problem without timing speculation are as below,

$$\begin{aligned} s_i - s_j &\le T_{cp} - T_{setup} - D_{ij} \quad , \\ s_i - s_j &\ge T_{hold} - d_{ij} \quad , \end{aligned} \tag{1}$$

where T_{cp} is the clock period, and T_{setup} and T_{hold} are the setup time and the hold time of FFs, respectively. In order to solve this CSS problem using a graph-theoretic approach, these timing constraints

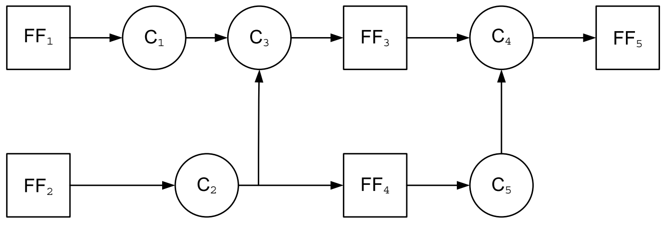

Fig. 1. A simple example of synchronous digital circuit.

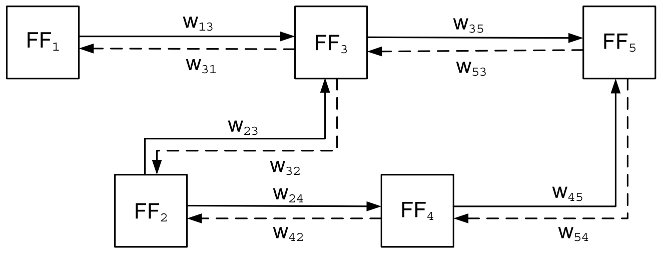

Fig. 2. Timing constraint graph of circuit example.

would usually be presented in timing constraint graph (see Fig. 2) by replacing the hold time constraint of the path from FF_i to FF_j with edge weight $w_{ji} = -(T_{hold} - d_{ij})$ and replacing the corresponding setup time constraint with edge weight $w_{ij} = T_{cp} - T_{setup} - D_{ij}$. The solid directed lines in Fig. 2 imply the setup time relation and the dashed directed lines imply the hold time relation (refer to [18] for more background about CSS).

B. Related Work

Properly assigning clock arrival times to FFs, known as clock skew scheduling, can help solve the problem that the maximum achievable operating frequency is limited by the maximum datapath delay in the circuit. There has been significant effort to explore CSS to improve performance. Generally speaking, clock skew scheduling can be classified into two categories by different optimization objectives: performance-driven ones [1, 2] to achieve the highest operational frequency and timing yield-driven ones [3–7] to maximize yield under a given clock period.

Some early works [1, 2, 5] laid a foundation by formulating the CSS problem and solving it optimally. The author of [16] argued that these optimal solutions do not achieve the lower bound of clock period, since the hold time constraints often limit the feasible clock period, and proposed delay insertion into the logic network as a post-processing step to solve the hold time violations and help improve the feasible timing schedule.

However, even after delay insertion, it is known that the minimum clock period of a synchronous circuit achievable through CSS is still limited by the uncertainties of the data-propagation times on local data paths, caused by the ever-increasing process variations. Considering this variability of critical path delays, some prior works (e.g., [1, 2]) allocated a safety margin with both upper and lower bounds to each feasible region of clock skews in advance to minimize clock period. Intuitively, since it seems more reasonable to target a skew close at the middle of skew feasible region, the authors of [3] proposed an original formulation of this problem and solved it using quadratic programming to minimize the total least square of skews. In [4], the authors optimized clock slacks using an incremental slack distribution method to tolerate process variation. In [5], the authors modeled the variations on critical path delays with a single variable and transformed the yield optimization problem to a minimum mean cycle problem to guarantee safety margins. In [6], the authors modeled the problem as a minimum cost-to-time ratio problem by introducing variance of path delay distribution into the feasible skew region to consider the statistical difference of critical path delays. Recently in [7], the authors argued that all prior works cannot handle non-Gaussian critical path delays, and hence proposed a formulation of yield-driven clock skew scheduling technique under non-Gaussian variations.

The above works all try to conduct CSS with the guarantee that there would never be timing errors to occur. With this serious promise, the traditional CSS has to reserve a rather large timing guard band, which limits the benefits brought by conducting CSS.

C. Timing Speculation

Circuit-level timing speculation technique, being able to detect timing errors at online stage, react to the error quickly and recover from it by rolling back to a known-good pre-error state, has become one of the most promising solutions to deal with the ever-increasing static and dynamic variation effects with technology scaling. With timing speculation, timing error is no longer the evil that has to be avoided at the cost of timing guard band. It provides designers the opportunity to trade off error rate with operating clock period.

Various techniques have been presented for online timing error detection. Without loss of generality, let us consider the representative *Razor* flip-flop [8] to demonstrate how timing error detectors work. A Razor-FF contains a main flip-flop, a shadow latch and some additional control logic, to detect timing errors. The main flip-flop latches the output signal at the clock edge with possible timing error, while the shadow latch guarantees to receive the correct value, by latching the signal a fraction of a cycle later. Consequently, when the shadow latch and the main FF values do not agree, a timing error is detected. To make use of timing speculation technique, it is necessary to replace all critical FFs that are driven by speed-paths (i.e., critical or near-critical paths) of the circuit with Razor-FFs (or other timing speculators).

D. Motivation

As discussed above, we formulate the CSS problem targeting circuits equipped with timing speculation capability and present a novel CSS algorithm based on gradient-descent method to maximize circuit performance in this work. The proposed technique is motivated by the observation that a large design guard band needs to be reserved for the traditional clock skew optimization techniques, due to the increasing process variation. With timing speculation, we do not need to guarantee "always correct" operation any longer, which dramatically increases the flexibility and improvement room of clock skew optimization techniques. Meanwhile, CSS can manipulate the timing budgets of different FFs, so that those FFs with larger sensitization probability can be better taken care of by allocating more timing slacks. In other words, the combination of these two techniques can naturally complement each other to improve system performance, which motivates this work to consider timing speculation explicitly into CSS.

III. Problem Formulation

The CSS problem formulation for circuits with timing speculation capability is quite different from traditional one (e.g., [7]), because our optimization objective is not the absolute clock period and we need to differentiate the contributions of various circuit paths to the system's overall timing error rate. We detail our problem formulation as follows.

Generally speaking, for circuits with timing speculation capability, we can roll back the system once timing error is detected and then lower system frequency for a short while to re-compute the result in the failure cycle. Similar to [14], we can trade off timing error rate with operating clock period to minimize equivalent clock period T_{ecp} according to the following optimization objective,

$$\min_{\{T_{cp},\vec{s}\}} T_{ecp} = (1 + error(T_{cp}, \vec{s}) \cdot penalty) \cdot T_{cp} \quad , \tag{2}$$

where T_{cp} is operating clock period, $\vec{s}$ is the vector for skew setting, $error(T_{cp}, \vec{s})$ is error cycle rate function with regard to T_{cp} and $\vec{s}$, indicating the probability for timing error to occur, and $penalty$ is the penalty due to error occurring. This penalty includes both the cycles of wasted execution that must be discarded when an error occurs and the time spent on checkpointing and re-execution.

To achieve above optimization objective, if a certain T_{cp} is given, the problem in Eq. 2 can be developed as

$$\min_{\{\vec{s}\}} error(\vec{s}) = 1 - \prod_{j=1}^{|V|} (1 - error_j(\vec{s})) \quad , \tag{3}$$

where $error_j(\vec{s})$ is error probability of FF_j during one clock cycle. By solving above optimization problem, we can finally select the best operating clock period T_{cp} to minimize the equivalent clock period as indicated in Eq. 2.

To calculate $error_j(\vec{s})$, the error probability of each FF, we need to know the path sensitization probability, which can be acquired by performing functional simulation of the circuit with representative workloads. Note that, we only need to record the sensitized delay information of FFs during this simulation process, but not all the sensitized path delays.

Besides, considering process variation, the path delay becomes a random variable [7], therefore we have

$$\begin{aligned} error_j(\vec{s}) &= \sum_{i \in P_j} \sum_k N_{ijk} \cdot Pr\{s_i - s_j \geq \widetilde{W}_{ijk}\} \quad , \\ \widetilde{W}_{ijk} &\triangleq T_{cp} - T_{setup} - \widetilde{D}_{ijk} \quad , \end{aligned} \tag{4}$$

where N_{ijk} is the probability for the k^{th} path from FF_i to FF_j to be sensitized, $\widetilde{D}_{ijk}$ is the corresponding path delay variable of the k^{th} path, s_i is skew value of FF_i, $\widetilde{W}_{ijk}$ denotes a random variable as defined in Eq. 4, and P_j is the set of FFs that serve as the beginners of the paths from FF_i to FF_j. Note that, since the problem of hold time constraints can be solved by delay padding [16], we only consider the setup time constraints in this work. $Pr\{s_i - s_j \geq \widetilde{W}_{ijk}\}$ implies the probability for timing error to occur if the setup time constraint as specified in Eq. 1 is violated.

By defining the Cumulative Distribution Function (CDF) of $\widetilde{W}_{ijk}$ as

$$F_{ijk}(x) = Pr\{x \geq \widetilde{W}_{ijk}\} \quad , \tag{5}$$

where $x = s_i - s_j$, we have

$$error_j(\vec{s}) = \sum_{i \in P_j} \sum_k N_{ijk} \cdot F_{ijk}(s_i - s_j) \quad . \tag{6}$$

Note that, if we consider the path delay between FF pair as a variable following a certain distribution, we can have a general form for error probability calculation using an integral form, instead of the summation in Eq. 6,

$$\begin{aligned} error_j &= \sum_{i \in P_j} \int_{-\infty}^{+\infty} \mathbf{N}_{ij}(x) \cdot \mathbf{F}_{ij}(s_i - s_j, x) dx \quad , \\ \mathbf{F}_{ij}(s_i - s_j, x) &= Pr\{s_i - s_j > T_{cp} - T_{setup} - x\} \quad , \end{aligned} \tag{7}$$

where $\mathbf{N}_{ij}(x)$ is the probability function with regard to sensitized path delay x.

Finally, based on Eq. 3 and Eq. 6, we can formulate the CSS problem with timing speculation as an optimization problem as below,

$$\min_{\{\vec{s}\}} error(\vec{s}) = 1 - \prod_{j=1}^{|V|} (1 - \sum_{i \in P_j} \sum_k N_{ijk} \cdot F_{ijk}(s_i - s_j)) \quad . \tag{8}$$

The optimization targeting this objective is expected to reduce the error cycle rate of overall system so that it can improve circuit throughput.

IV. GDM-Based Skew Scheduling Algorithm

In this work, we develop a novel technique based on gradient-descent method (GDM) to solve the CSS problem with timing speculation. The simplest version of GDM (refer to [17] for details) is usually formulated as the following unconstrained optimization problem:

$$\min_{\{\vec{s}\}} f(\vec{s}), \vec{s} \in \mathbb{R}^n \quad , \tag{9}$$

where $f(\vec{s})$ is a scalar objective function, $\mathbb{R}^n$ is the n-dimensional real Euclidean space, and $\vec{s}$ is a vector of n real components, $\{s_i | 1 \leq i \leq n\}$. It is assumed that $f(\vec{s})$ is differentiable so that the gradient vector $\triangledown f(\vec{s})$ exists everywhere in $\mathbb{R}^n$. The solution of above problem is denoted as $\vec{s}^*$.

GDM is a first-order optimization algorithm that use the gradient vector $\triangledown f(\vec{x})$ to determine the search direction for each iteration. The simplest and most famous GDM algorithm is the *method of steepest descent*, which takes steps proportional to the negative/positive of the gradient (or, the approximate gradient) of the function at current iteration to minimize/maximize $f(\vec{s})$.

The targeted CSS problem with timing speculation in this work is how to determine clock skew setting and hence allocate timing slack as resource to the most critical paths to reduce error cycle rate of overall system and improve system throughput. This is strongly relevant to both sensitization probability and expected error probability for each timing critical/sub-critical path. From this viewpoint, GDM is just applicable to our targeted problem. How to apply GDM in targeted CSS problem will be detailed in the following.

A. Proposed Optimization Metric

From Eq. 2, we can see that the system error cycle rate must be reduced and kept within a small number to achieve a higher throughput, otherwise the penalty caused by error occurrence would greatly reduce the benefit of clock period decrease and even worsen the throughput.

Based on this observation, we can simplify the original optimization objective described in Eq. 8 by assuming $error_j(\vec{s})$, the error probability for each FF, is so small that there would not be more than one error occurring at the same time. Therefore, after expanding Eq. 3 we can ignore the high order terms and obtain the following approximation,

$$error(\vec{s}) = 1 - \prod_{j=1}^{|V|} (1 - error_j(\vec{s})) \approx \sum_{j=1}^{|V|} error_j(\vec{s}) \quad . \tag{10}$$

This simplified optimization objective is used in our GDM-based skew scheduling algorithm as a metric to guide the reduction of system error cycle rate. After substituting Eq. 6 into Eq. 10, the proposed optimization metric is written as

$$error(\vec{s}) = \sum_{j=1}^{|V|} \sum_{i \in P_j} \sum_{k} N_{ijk} \cdot F_{ijk}(s_i - s_j) \quad . \tag{11}$$

B. Skew Constraint

In practice, the assigned skew values should be constrained in a specified range with a lower bound and an upper bound (e.g., $[s_{lower}, s_{upper}]$). This constraint requirement is beyond the basic unconstrained formulation of GDM in Eq. 9. In order to tackle this problem, we can use such a function,

$$S(x_i) = s_{lower} + (s_{upper} - s_{lower}) \cdot (1 - \frac{1}{1 + e^{\mu x_i}}) \quad , \tag{12}$$

to constrain the assigned skew values within specified range $[s_{lower}, s_{upper}]$ all the time. Here, μ is a parameter to control the function shape. As a result, by substituting Eq. 12 into Eq. 11, we obtain the optimization objective written as

$$error(\vec{x}) = \sum_{j=1}^{|V|} \sum_{i \in P_j} \sum_{k} N_{ijk} \cdot F_{ijk}(S(x_i) - S(x_j)) \quad , \tag{13}$$

where $\vec{x}$ is the new vector variable. Obviously, once $\vec{x}^*$ is determined, the skew value $\vec{s}^*$ can be easily calculated according to Eq. 12. The transformation from a constrained problem to an unconstrained version makes GDM applicable to targeted CSS problem.

C. Proposed Skew Scheduling Algorithm

After approximating optimization metric in Section IV-A and applying the transformation in Section IV-B, it is now ready for GDM to determine the clock skew setting.

We compute the gradient of objective function. For $\forall\ \ell \in \{1, 2, \cdots, n\}$,

$$\begin{aligned}\frac{\partial error(\vec{x})}{\partial S(x_\ell)} &= -\sum_{i \in P_\ell} \sum_{k} N_{i\ell k} \cdot F'_{i\ell k}(S(x_i) - S(x_\ell)) \\ &\quad + \sum_{\ell \in P_j} \sum_{k} N_{\ell jk} \cdot F'_{\ell jk}(S(x_\ell) - S(x_j)) \quad ,\end{aligned} \tag{14}$$

$$\frac{dS(x_\ell)}{dx_\ell} = (s_{upper} - s_{lower}) \frac{\mu e^{\mu x_\ell}}{(1 + e^{\mu x_\ell})^2} \quad , \tag{15}$$

where $F'_{ijk}(\cdot)$ is the probability density function. Based on Eq. 14 and Eq. 15, we can finally get the gradient according to

$$\frac{\partial error(\vec{x})}{\partial x_\ell} = \frac{\partial error(\vec{x})}{\partial S(x_\ell)} \cdot \frac{dS(x_\ell)}{dx_\ell} \quad . \tag{16}$$

Since we would like to minimize the error cycle rate, we update the parameters using the negative of the computed gradient at each iteration as below,

$$x_\ell^{new} = x_\ell - \eta \frac{\partial error(\vec{x})}{\partial x_\ell} \quad , \tag{17}$$

where η is the learning rate. By updating the parameter $\vec{x}$ periodically until the procedure converges, we can obtain the skew setting $\vec{s}^* = S(\vec{x}^*)$ according to the definition in Eq. 12.

The algorithm flow of proposed CSS technique is described in Fig. 3. First of all, we initialize the variables $\vec{x}$ as same as the case with *zero*-skews. Then, we repeat the procedure (see Line 3 $\sim$ Line 14) to compute the gradient and update parameters until the convergence criterion is satisfied.

```
#    $f(\vec{x})$, the objective function for optimization
#    $\epsilon_g$ and $\epsilon_{\vec{x}}$, convergence tolerances
1.   Initialize $\vec{x}$
2.   REPEAT for each iteration
3.       IF $\| \nabla f(\vec{x}) \| < \epsilon_g$
4.           Set $\vec{x}^* = \vec{x}$
5.           Break
6.       ELSE
7.           Set $\vec{g} = -\nabla f(\vec{x})$
8.       FOR the iteration $i$ from 1 to $|V|$
9.           Update parameters $x_i^{new} = x_i - \eta \cdot g_i$
10.      IF $\| \vec{x}^{new} - \vec{x} \| < \epsilon_{\vec{x}}$
11.          Set $\vec{x}^* = \vec{x}^{new}$
12.          Break
13.      ELSE
14.          Go for next iteration
```

Fig. 3. Proposed GDM-based skew scheduling algorithm.

V. Experimental Results

A. Experimental Setup

To evaluate the effectiveness of proposed CSS method, we conduct experiments on several large ISCAS'89 and IWLS'05 benchmarks. We synthesize these circuits and obtain timing information using Synopsys EDA tools. To take process variation effect into consideration, we perform Monte Carlo simulation to inject gate-level delay variation following Gaussian distribution with standard deviation equal to 8%. We conduct simulation with random inputs in our experiments and the simulation is performed with 100,000 cycles. All the experiments are conducted on a *2.8GHz* PC with *4GB* RAM.

For reasonable comparison, we provide a reference via offline timing analysis with false paths excluded according to [19], denoted as $CSS_{nonrazor}$. We apply timing speculation directly without CSS and denote this solution as CSS_{noncss}. That means, we assign all the FFs with *zero*-skew in CSS_{noncss}. A yield-driven CSS work [7] is used as the baseline solution for comparison and denoted as $CSS_{baseline}$. Our proposed CSS technique is denoted as $CSS_{proposed}$. The three solutions mentioned above with timing speculation (CSS_{noncss}, $CSS_{baseline}$, and $CSS_{proposed}$) select the operating clock period of circuits according to Eq. 2 to minimize the equivalent clock period T_{ecp} which considers the penalty for error occurring. Note that, for fair comparison, in $CSS_{baseline}$ we apply the algorithm in [7] to determine skew setting and then find out the optimal clock period since timing speculation capability is equipped. The comparison between $CSS_{baseline}$ and $CSS_{proposed}$ can reflect the benefit of explicitly combining CSS with timing speculation, since $CSS_{proposed}$ consider timing speculation capability in the optimization procedure of CSS technique explicitly.

Usually, which FF to be equipped as Razor-FF lies on its timing pressure. Without considering CSS, a simple scheme is to equip all the FFs, whose maximum delays are larger than β of clock period (e.g., $\beta = 80\%$), as Razor-FFs. Because pre-silicon CSS discussed in this paper is performed offline at design stage and its effect is just manipulating timing budgets between FFs, we can simply set up those FFs, whose timing slacks are less than $(1 - \beta)$ of clock period after conducting CSS, as Razor-FFs. In our experiments, the timing threshold β is set to be 80%, and the hardware cost for equipping each Razor-FF is assumed to be 10 gates. The *penalty* in Eq. 2 is assumed to be 10 clock cycles according to [14], and we sweep the operating clock cycle to select the one with best performance.

Bench.	TG#	TFF#	CSS_{noncss}		$CSS_{baseline}$		$CSS_{proposed}$		Runtime (s)
			$RFF\#$	Cost(%)	$RFF\#$	Cost(%)	$RFF\#$	Cost(%)	
s1494	680	6	2	2.94	2	2.94	3	4.41	0.17
s5378	3042	179	22	7.23	18	5.92	32	10.52	1.20
s9234	5866	228	22	3.75	25	4.26	18	3.07	1.15
s13207	8803	638	10	1.14	10	1.14	10	1.14	2.06
s15850	10470	597	89	8.50	89	8.50	81	7.74	3.33
s35932	18148	1728	144	7.93	144	7.93	144	7.93	13.76
s38584	21021	1426	168	6.90	180	7.39	182	7.48	16.30
s38417	24341	1564	87	4.14	106	5.23	108	5.14	10.29
wb_conmax	73233	3316	657	8.97	699	9.54	698	9.53	23.93
des_perf	154204	9105	668	4.33	887	5.75	870	5.64	46.83
ethernet	157841	10752	1553	9.84	1602	10.15	1584	10.04	34.87
AVERAGE				5.97		6.25		6.60	

TG#: total gate count; TFF#: total FF count; RFF#: Razor-FF count;
Cost: hardware cost ratio for equipping Razor-FFs.

TABLE I
EXPERIMENTAL RESULTS ON HARDWARE COST AND ALGORITHM RUNTIME.

Bench.	$CSS_{nonrazor}$	CSS_{noncss}			$CSS_{baseline}$			$CSS_{proposed}$				
	$T_{ecp}(ns)$	$T_{ecp}(ns)$	$\sigma(ns)$	$\Delta_1(\%)$	$T_{ecp}(ns)$	$\sigma(ns)$	$\Delta_2(\%)$	$T_{ecp}(ns)$	$\sigma(ns)$	$\Delta_3(\%)$	$\Delta_4(\%)$	$\Delta_5(\%)$
s1494	2.72	2.38	0.068	-12.63	2.29	0.067	-3.73	2.19	0.070	-4.32	-7.89	-19.52
s5378	1.86	1.58	0.017	-15.21	1.36	0.024	-13.45	1.35	0.024	-1.13	-14.43	-27.44
s9234	4.58	4.05	0.086	-11.61	3.84	0.068	-5.16	3.39	0.064	-11.65	-16.21	-25.94
s13207	5.57	4.12	0.080	-25.97	4.09	0.081	-0.72	4.05	0.048	-1.07	-1.78	-27.29
s15850	6.22	5.28	0.035	-15.12	4.75	0.046	-9.95	4.55	0.063	-4.38	-13.89	-26.91
s35932	3.68	3.57	0.070	-2.99	3.53	0.071	-1.03	3.49	0.073	-1.13	-2.15	-5.07
s38584	6.96	6.39	0.145	-8.10	6.21	0.129	-2.87	6.07	0.106	-2.29	-5.09	-12.78
s38417	6.12	5.73	0.065	-6.48	5.17	0.062	-9.76	4.57	0.042	-11.56	-20.20	-25.36
wb_conmax	6.43	5.97	0.078	-7.20	5.53	0.063	-7.28	4.91	0.107	-11.21	-17.68	-23.60
des_perf	13.70	13.44	0.340	-1.90	12.23	0.309	-9.00	10.88	0.275	-10.99	-19.00	-20.54
ethernet	11.28	10.53	0.055	-6.65	10.28	0.050	-2.35	9.96	0.057	-3.11	-5.39	-11.68
AVERAGE				-10.35			-5.94			-5.71	-11.25	-20.56

T_{ecp}: mean equivalent clock period considering error penalty;
σ: standard deviation of equivalent clock period T_{ecp};
Δ_1: T_{ecp} difference ratio between CSS_{noncss} and $CSS_{nonrazor}$;
Δ_2: T_{ecp} difference ratio between $CSS_{baseline}$ and CSS_{noncss};
Δ_3: T_{ecp} difference ratio between $CSS_{proposed}$ and $CSS_{baseline}$;
Δ_4: T_{ecp} difference ratio between $CSS_{proposed}$ and CSS_{noncss};
Δ_5: T_{ecp} difference ratio between $CSS_{proposed}$ and $CSS_{nonrazor}$.

TABLE II
EXPERIMENTAL RESULTS ON EQUIVALENT CLOCK PERIOD T_{ecp}.

B. Results and Discussion

First of all, we report the results on hardware cost and algorithm runtime of proposed CSS technique in Table I. As can be seen, the average hardware cost for CSS_{noncss}, $CSS_{baseline}$ and $CSS_{proposed}$ to enable timing speculation are 5.97%, 6.25% and 6.60%, respectively. Although the hardware cost after conducting CSS seems a little higher than CSS_{noncss}, the additional cost is still within an acceptable range. The runtime for proposed GDM-based CSS technique is listed in Column 10 of Table I.

Secondly, we perform Monte Carlo simulation to produce 100 sample chips with different variation patterns for each benchmark. In Table II, we report the equivalent clock period T_{ecp} and its standard deviation σ for CSS_{noncss}, $CSS_{baseline}$, and $CSS_{proposed}$, respectively. This clock period T_{ecp} has taken both the selected operating clock period and the penalty for error occurring into account. As can be seen, CSS_{noncss} can achieve 10.35% reduction in T_{ecp} in average when compared with $CSS_{nonrazor}$. This improvement reflects the efficacy of Razor technique itself without CSS, which is not the main contribution of this work. After applying $CSS_{baseline}$, a yield-driven CSS proposed in [7], another 5.94% improvement can be achieved in average, which shows that conducting CSS is beneficial for timing speculation. Compared with $CSS_{baseline}$, $CSS_{proposed}$ can obtain 5.71% further improvement. This reflects the efficacy to consider timing speculation in CSS optimization procedure explicitly. The improvement for $CSS_{proposed}$ over CSS_{noncss} and $CSS_{nonrazor}$ are demonstrated in Column 12-13 of Table II.

Finally, to get more details of proposed CSS technique, we take $s38417$, the largest circuit in ISCAS'89, as an example in the following. In Fig. 4, we show the results with variation effects after performing Monte Carlo simulation. The corresponding mean of T_{ecp} and standard deviation σ are depicted in Fig. 4. Moreover, in Fig. 5, we show the trends of both proposed optimization metric in Eq. 11 and the error cycle rate obtained by timing simulation with respect to operating clock period. Proposed CSS technique always selects the operating clock period with reasonable error cycle rate to minimize equivalent clock period (or, maximize throughput). In Fig. 6, the trends of proposed optimization metric and error cycle rate with respect to GDM iteration number under the optimal operating clock period are plotted out. This figure shows the details of proposed GDM-based skew scheduling algorithm to optimize proposed metric and hence reduce the error cycle rate. The similar trend of these two curves in Fig. 6 proves the effectiveness of proposed optimization algorithm and metric.

VI. CONCLUSION

Clock skew scheduling has been exploited as an effective technique to improve timing performance and yield of ICs and attracted lots of research interest. However, with the ever-increasing process variations, a large timing guard band has to be reserved to tolerate

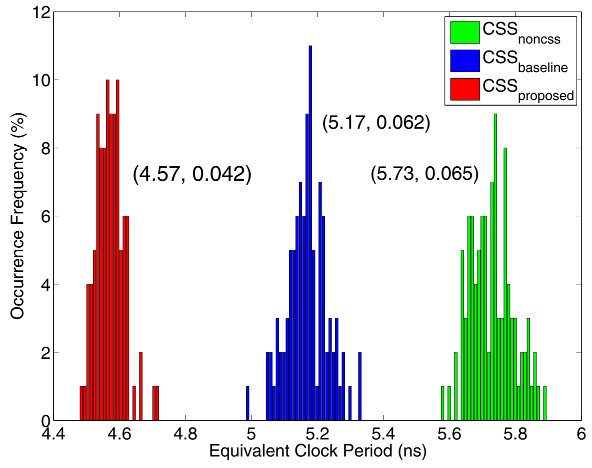

Fig. 4. Experimental results to show variation effects.

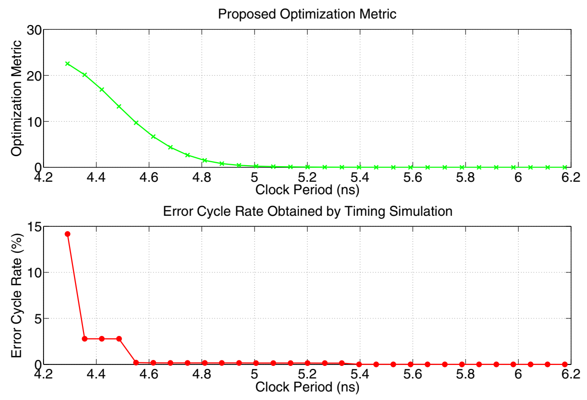

Fig. 5. Experimental results wrt. clock period.

variations and guarantee "always correct" operations. In this work, we formulate the CSS problem for circuits equipped with timing speculation capability, and propose a GDM-based skew scheduling algorithm to determine skew setting effectively. Experimental results on benchmark circuits demonstrate that the proposed CSS technique is able to significantly enhance circuit performance.

VII. Acknowledgement

This work was supported in part by the General Research Fund CUHK417807 and CUHK418111 from Hong Kong SAR Research Grants Council (RGC), in part by a grant N_CUHK417/08 from the NSFC/RGC Joint Research Scheme, and in part by NSF under CCF-1115550.

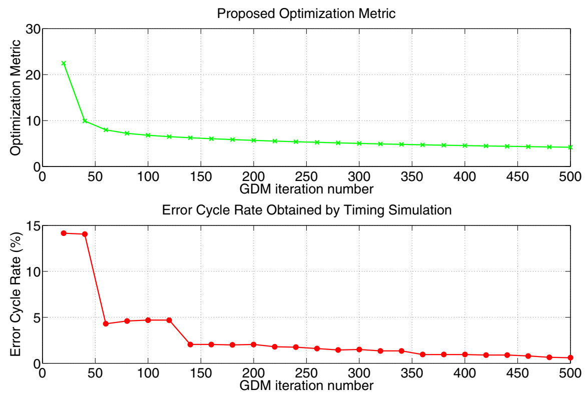

Fig. 6. Experimental results wrt. GDM iterations.

References

[1] J. P. Fishburn. Clock skew optimization. In *IEEE Transactions on Computers*, vol.39, pp.945-951, July 1990.

[2] R. B. Deokar and S. S. Sapatnekar. A graph-theoretic approach to clock skew optimization. In *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 407-410, 1994.

[3] I. S. Kourtev and E. G. Friedman. Clock skew scheduling for improved reliability via quadratic programming. In *Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 239-243, 1999.

[4] X. Wei, Y. Cai, and X. Hong. Clock skew scheduling under process variation. In *Proc. IEEE International Symposium on Quality Electronic Design (ISQED)*, pp. 237-242, 2006.

[5] C. Albrecht, B. Korte, J. Schietke, and J. Vygen. Cycle time and slack optimization for VLSI-chips. In *Proc. IEEE/ACM International Conference on Computer-aided Design (ICCAD)*, pp. 232-238, 1999.

[6] J. L. Tsai, D. H. Baik, C. C. P. Chen, and K. K. Saluja Yield-driven, false-path-aware clock skew scheduling. In *IEEE Design & Test of Computers*, vol. 22, no. 3, pp. 214-222, 2005.

[7] Y. Wang, et al. Timing yield driven clock skew scheduling considering non-Gaussian distributions of critical path delays. In *Proc. IEEE/ACM Design Automation Conference (DAC)*, pp. 223-226, 2008.

[8] D. Ernst, et al. Razor: A low-power pipeline based on circuit-level timing speculation. In *Proc. IEEE/ACM International Symposium on Microarchitecture (Micro)*, pp. 7-18, 2003.

[9] S. R. Sarangi, et al. VARIUS: A model of process variation and resulting timnig errors for microarchitectus. In *IEEE Transactions on Semiconductor Manufacturing*, vol. 21, pp. 3-13, Feb. 2008.

[10] S. Sarangi, B. Greskamp, A. Tiwari, and J. Torrellas. EVAL: Utilizing processors with variation-induced timing errors. In *Proc. IEEE/ACM International Symposium on Microarchitecture (Miro)*, pp. 423-434, 2008.

[11] Y. Liu, F. Yuan and Q. Xu. Re-synthesis for cost-efficient circuit-level timing speculation. In *Proc. ACM/IEEE Design Automation Conference (DAC)*, pp. 158-163, 2011.

[12] K. Bowman, et al. Circuit techniques for dynamic variation tolerance. In *Proc. ACM/IEEE Design Automation Conference (DAC)*, pp. 4-7, 2009.

[13] T. Austin and V. Bertacco. Deployment of better than worst-case design: solutions and needs. In *Proc. IEEE International Conference on Computer Design (ICCD)*, pp. 550-558, 2005.

[14] M. de Kruijf, S. Nomura, and K. Sankaralingam. A unified model for timing speculation: Evaluating the impact of technology scaling, CMOS design style, and fault recovery mechanism. In *Proc. International Conference on Dependable Systems and Networks (DSN)*, pp. 487-496, 2010.

[15] R. Ye, F. Yuan and Q. Xu. Online clock skew tuning for timing speculation. In *Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 442-447, 2011.

[16] S. Huang, C. Cheng, C. Chang, and Y. Nieh. Clock period minimization with minimum delay insertion. In *Proc. IEEE/ACM Design Automation Conference*, pp. 970-975, 2007.

[17] J. A. Snyman. Practical mathematical optimization: An introduction to basic optimization theory and classical and new gradient-based algorithms. *Springer*, 2005.

[18] I. S. Kourtev, B. Taskin, and E. G. Friedman. Timing optimization through clock skew scheduling. *Springer*, 2009.

[19] F. Yuan and Q. Xu. On timing-independent false path identification. In *Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 532-535, Nov. 2010.