The Advantages of Latch-Based Design Under Process Variation

Aaron P. Hurst, Robert K. Brayton
University of California, Berkeley
Berkeley, CA
{ahurst, brayton}@eecs.berkeley.edu

ABSTRACT

Latch-based design may offer power and area savings, but its theoretical performance is no better than that of register-based design after clock scheduling is applied. Under the traditional deterministic timing model, the optimal period in both latch- and register-based designs is limited by the maximum mean delay of any cycle in the circuit. However, when process variation is considered, latch-based design is often dramatically more variation-tolerant, resulting in a better yield and/or allowing a more aggressive clocking than the equivalent design with registers. This effect can not be observed using traditional deterministic timing analysis and requires a probabilistic timing model to quantify. We analyze several benchmark circuits, and demonstrate that manufacturing a latch-based design will result in 4 times fewer failures than the equivalent design using registers. Finally, we consider of the problem of how to schedule the clocks of latches to further maximize the yield.

Keywords

Clock Scheduling, Register, Latch, Design-for-Yield, Statistical Timing Analysis.

1. INTRODUCTION

The synchronization and state storage in a sequential design is typically implemented with one of two devices: edge-triggered registers or level-sensitive latches. An ideal edge-triggered element will propagate the value from its input to its output only at the instant during each clock cycle when the clock input is rising (or falling, depending on the implementation) and maintain the its output value at all other times. An ideal level-sensitive device will propagate the value from its input to its output whenever the clock input is high (or low) and maintain the its output value at all other times. The latch is *transparent* when input can propagate directly to the output and *closed* otherwise.

At the cost of additional timing verification effort, latch-based design can offer area and power savings over register-based design. When driven by a single global clock, latches are also often able to support a higher frequency of operation. The transparent mode of a latch allows a limited amount of delay balancing between adjacent combinational paths, possibly overcoming the period limitation imposed by the single longest path. Registers are always limited by the single longest path.

This performance advantage over register-based design seems to disappear when clock skew scheduling is considered. Clock skew scheduling [1] optimally balances the combinational path delays by applying different non-zero skews on the clock inputs of each register or latch, often greatly improving performance. With the application of clock scheduling, the optimal clock period in both latch- and register-based designs is identical, assuming that any hold violations are corrected. In both cases, the performance is limited by the maximum of the sum of delay along any loop in the circuit divided by the number of registers or latches along that loop: the maximum mean cycle time.

Process variation is implicitly accounted for in most deterministic timing models by considering a set of corner cases where all components behave antagonistically in some manner. It has been recognized that this is overly pessimistic, resulting in costly over-design. [2] The introduction of process variation-aware timing has had success in addressing this problem, and the representation of timing variation as a the linear vector of independent normally-varying components has been particularly successful. [3,4] We employ this model to capture the effect of manufacturing variations on delay.

While clock scheduling can be used to optimally balance a single set of delay values, process variation may result in one of an infinite number of timing possibilities for every manufactured instance. One clock schedule can not optimally balance all possible timings, and some of these variations that fail under this schedule could have possibly been corrected with another. Design-time optimizations, such as clock-skew scheduling, can only be used to choose the best solution for the largest number of devices.

Latch-based circuits transcend this limitation. Instead of fixing a single balance point, the timing of a latch allows delays to be balanced anywhere within the transparent window. This allows a larger number of possible variations to meet timing. In the deterministic model this offered no

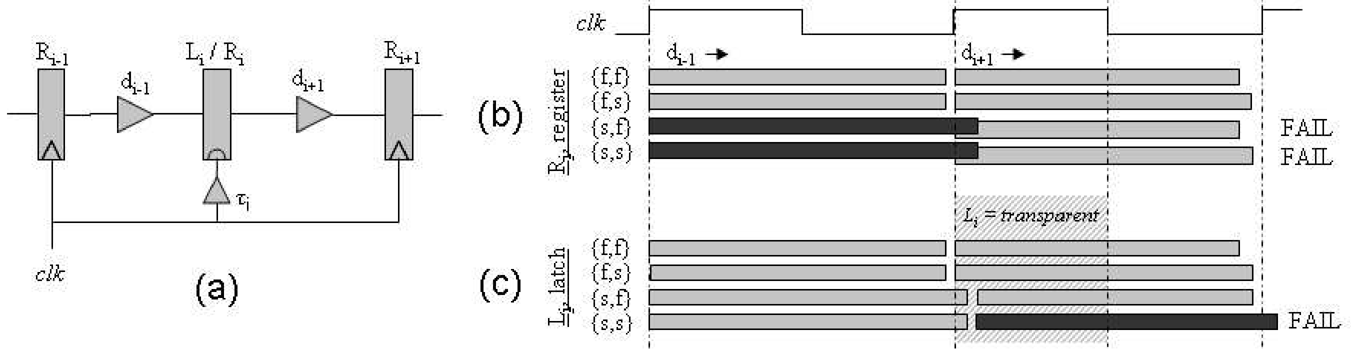

Figure 1: A simple example illustrating the advantages of a latch under process variation. The circuit structure (a), in any manufactured instance, is equally likely to have either a fast ("f") or slow ("s") version of the two delay elements. If the middle sequential element (L_i / R_i) is a register, out of the four possible timing cases, more fail to meet timing (b) than if the middle sequential element were a latch (c). The failing delays are solid black.

advantage, with only a single worst-case timing; in the statistical timing model this translates into an increase in yield, or if used to clock the latches more aggressively, an increase in performance. Latches hold an advantage over registers. The greater maximum yield of latch-based design is illustrated using a simple example and then demonstrated on several benchmark circuits.

We also introduce the problem of *statistical latch scheduling*, which seeks to optimize the timing of the latch clocks to maximize the yield. A variant of the problem was described in [5] and solved by scheduling latches so that the worst-case timing meets even more restrictive timing constraints, but this is done blind to the details of the actual process variations and amounts to over-design. We present a fully variation-aware formulation of the problem and offer some simplifications and approximates. Using a heuristic, we demonstrate how to greatly increase the yield and/or performance of a latch-based design over its register-based equivalent.

2. MOTIVATING EXAMPLE

To illustrate the potentially greater variation-tolerance of latches, a contrived example circuit is shown in Figure 1a. There are three sequential elements: two registers, R_{i-1} and R_{i+1}, and R_i/L_i, which may either be a register or a latch. There are also two delay elements: d_{i-1} and d_{i+1}. For the circuit to operate correctly at the desired clock period *T*, the setup and hold timing constraints must be satisfied at each of the sequential elements. Ignoring other types of failures, the yield is the fraction of manufactured instances that successfully meet all timing constraints.

The example is subject to the following simple manufacturing variation. Each of the two delay elements, d_{i-1} and d_{i+1}, comes independently in one of two varieties: fast or slow, each of equal probability.

$$d_{i-1} = \begin{cases} T & \text{"}fast\text{"} \quad \mathcal{P} = 50\% \\ 1.2T & \text{"}slow\text{"} \quad \mathcal{P} = 50\% \end{cases}$$
$$d_{i+1} = \begin{cases} 0.8T & \text{"}fast\text{"} \quad \mathcal{P} = 50\% \\ T & \text{"}slow\text{"} \quad \mathcal{P} = 50\% \end{cases} \tag{1}$$

Now consider the case where the middle sequential element is implemented as a register, R_i. The resulting timings of the four possible manufactured cases are illustrated in Figure 1b. It is clear that half of them fail to meet the setup constraint on R_i, resulting in a yield of 0.5. Note that no intentional skewing of R_i's clock (by inserting a delay, $\tau_i \neq 0$) will improve the yield. Even though the designer could use skewing to correct for the violations in either the "fast, slow" or the "slow, fast" instances, he can not simultaneous correct for both.

If the middle sequential element had been implemented as a latch, L_i, the timings of the four possible cases would have been different, as illustrated in Figure 1c. Here the transparency of the latch allowed the complementary variations in both the "fast, slow" and "slow, fast" cases to balance each other on an instance-by-instance basis. The yield of the latch-based design is 0.75, a result that is unachievable using a register.

3. STATISTICAL LATCH SCHEDULING

The SMO formulation of latch timing [6] provides a clear and concise description of the behavior and operational constraints of latches. We extend this formulation with a more general description of clock timing but also restrict the use of multi-cycle combinational paths.

Let there exist a periodic global reference signal *t* with period *T*. All latch timing quantities in cycle *n* are defined relative to the *n*th global signal, t_n. For each latch L_i, two variables are used to describe its clocking: $\tau_i^{\uparrow}$ the start of the transparent window, and $\tau_i^{\downarrow}$, the end of the transparent window. There is also a setup and hold time parameter associated with each L_i. Figure 2 illustrates these values.

The ordering of the events naturally constrains that $\tau_i^{\downarrow} > \tau_i^{\uparrow}$, and because the latch clock is T periodic, $\tau_i^{\downarrow} - \tau_i^{\uparrow} \leq T$. Note that there is no restriction that either $\tau_i < \mathrm{T}$; latches may be skewed by multiple periods relative to the global reference signal and to each other.

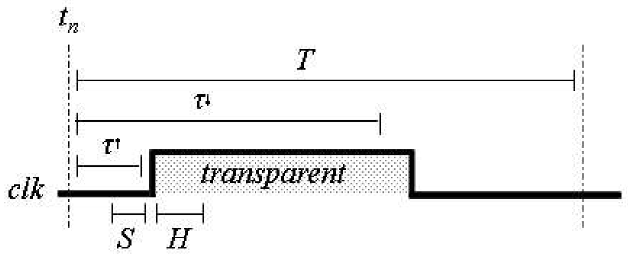

Figure 2: Relevant latch timing variables. All quantities are defined relative to an arbitrary global reference signal *t* with period *T*. The latch becomes transparent at time $\tau^{\uparrow}$ and closed at $\tau^{\downarrow}$. Correct functional operation requires that the input signal is stable for *S* and *H*.

The latches sit amidst combinational logic. If there exists a combinational path from latch L_j to L_i, then let $d_{j\to i}$ be the shortest path delay and $D_{j\to i}$ be the longest path delay. These delays are random variables of the form described in section 1. The path delays are extracted from the netlist using a block-based statistical timing analysis method similar to [3,4]. While the resulting distributions of delay are only approximations of the actual quantities, it has been demonstrated that this technique is quite accurate for typical circuits.

3.1 Complete Statistical Latch Scheduling

Let a_i be the earliest arrival time at the input of latch L_i, and A_i be the latest arrival time. The earliest time that the output of latch L_i will depart is z_i, and the latest time is Z_i. All four of these quantities are random variables.

$$\begin{aligned} a_i &= \min_{\forall j\to i}\left(z_j + d_{j\to i} - T\right) \\ z_i &= \max\left(a_i, \tau_i^{\uparrow}\right) \\ A_i &= \max_{\forall j\to i}\left(Z_j + D_{j\to i} - T\right) \\ Z_i &= \max\left(A_i, \tau_i^{\uparrow}\right) \end{aligned} \tag{2}$$

For the latch to operate correctly, both its setup and hold timing constraints must be met. The yield of the design, Equation 3, is the probability that all setup and hold timing constraints are simultaneously satisfied.

$$\mathcal{P}\left\{\forall_i\left(A_i + S_i \leq \tau_i^{\downarrow} \quad \wedge \quad a_i - H_i \geq \tau_i^{\downarrow} - T\right)\right\} \tag{3}$$

The complete latch scheduling problem is to maximize the yield by choosing all $\{\tau_i^{\uparrow}, \tau_i^{\downarrow}\}$.

The evaluation of Equation 3 for a particular set of variables is also known to the *statistical latch verification problem.* This is discussed in more detail in section 3.3

3.2 Single-Duty Statistical Latch Scheduling

Besides being a difficult optimization problem, a complete latch schedule is of little practical interest because of the difficulties in its physical implementation. In the register-based clock skew scheduling problem, one of edge of the clock needs to be scheduled. This can is typically implemented in an efficient manner by inserting delays on the clock inputs and shifting the phase of the clock. However, the complete latch schedule dictates the scheduling of both edges, for which a phase shift is not sufficient. The duty cycle (fraction of time that the clock signal is high) must also be adjusted. Compared to a simple delay element, the circuitry required to locally adjust the duty cycle is tremendously costly.

We propose a simplification to the complete latch scheduling problem by requiring that all clocks are have the same duty cycle. The resulting schedule becomes feasible to implement with current clock generation technology, requiring only local delay insertion and the generation of a global clock with a specific duty cycle, for which many techniques have been proposed. [8-10] The optimization problem is also simplified. Instead of two variables per latch, we need only choose a set of skews, one for each latch L_i, and the global duty cycle, δ. The mapping from the complete latch scheduling problem to the single duty latch scheduling problem is accomplished using the following substitution.

$$\forall_i \begin{pmatrix} \tau_i^{\downarrow} = \tau_i^{skew} \\ \tau_i^{\uparrow} = \tau_i^{skew} + \delta T \end{pmatrix} \tag{4}$$

Therefore, the single-duty latch scheduling problem is to maximize the yield, Equation 3, by choosing each τ^{skew} and a single global duty cycle δ.

The possibility remains of generating and distributing a finite number of clock signals with various duty cycles. The costs of benefits of this approach require further study, including the difficulty of the associated clock scheduling problem. We leave this for future work.

3.2.1 $\delta \to 0$

The case of $\delta \to 0$ warrants special attention. While the common implementation of a latch would not function correctly under such operating conditions, momentarily consider a device that can successfully latch the state with such a clock input. With a delta-length transparency window, its output would track its input at exactly one moment per cycle. This sequential behavior is identical to that of an edge-triggered register, and the timing of a latch-based design clocked with $\delta = 0$ is indeed indistinguishable from the timing of an equivalent register-based design. From an abstract timing view, registers are but a special case of latches.

3.2.2 Heuristic Solution

We generate a single-duty latch schedule in two phases. First, the duty cycle is fixed at a particular value and the set of clock skews, τ^{skew}, is chosen to optimize the objective. Fixing $\delta = 0$ is particularly convenient, as this skew schedule can be generated optimally, under a deterministic timing model, or using a heuristic variation-aware approach [11]. Next, with the τ^{skew} values fixed, the duty cycle, δ, is chosen to maximize the yield.

4. STATISTICAL LATCH VERIFICATION

Efficiently and accurately evaluating the objective, Equation 3, is central to the optimization problem. In the statistical latch scheduling problem, this is particularly difficult for two reasons. All of the relevant timing quantities are random variables, each with an associated distribution. Even if the delay distributions ($d_{j\rightarrow i}$ and $D_{j\rightarrow i}$) are normal, the arrival and departure times will not be; the maximum and minimum of a set of normal variables is not itself normal. There exists a normal approximation for the maximum and minimum operators [14], but this introduces approximation error. Also, if there are any cycles in the circuit, the definitions in Equation 2 become recursive. The two common solutions are to iterate until a fixpoint is reached [6,7] or to construct a structural graph and search for negative cycles. [12]

Chen and Zhou [13] have proposed a solution to the statistical latch verification problem that addresses these, using a mixed approach of approximated analytic analysis and random simulation. They random select and analyze acyclic sections of a structural timing graph, combining the results to calculate the composite yield of the entire design.

As with most random systems, a well-designed Monte Carlo simulation can also be valuable in characterizing the behavior of random quantities. The Monte Carlo version of statistical latch timing verification involves repeating one of the deterministic verification algorithms over a large number of random samples of the process variations. The fraction of successful timings is the resulting yield. The runtimes of this approach quickly become unattractively long, even for designs of moderate size. It is especially impractical to depend on this technique for use inside an optimization loop.

4.1 Iterative Analytical Approximation

Szymanski and Shenoy [7] prove that the latch arrival times can be computed by iteratively applying the operations of Equation 2 and that the arrival times will converge to a fixpoint. Chen and Zhou [13] initially dismiss a statistical version of this iterative method because of the problems of correlating variations across multiple iterations. This concern is warranted, but this problem also affects reconvergent structures in combinational timing analysis, but it appears not to significantly affect the result there.

In the statistical version of the iterative algorithm, the true distribution of a latch arrival time can be shown to converge because every set of delays will converge; the distribution is simply the set of solutions weighted by the total probability of the delays that give rise to that solution value. In the actual implementation, however, the distributions are repeatedly approximated as normal random variables, and it is not clear that the approximation error of [14] will not continue to accumulate and prevent convergence. For all of the examples tried, this appears not to be the case; the opposite appears to be true. The latch arrival time distributions converge quite quickly and remain constant, as illustrated in Figure 3.

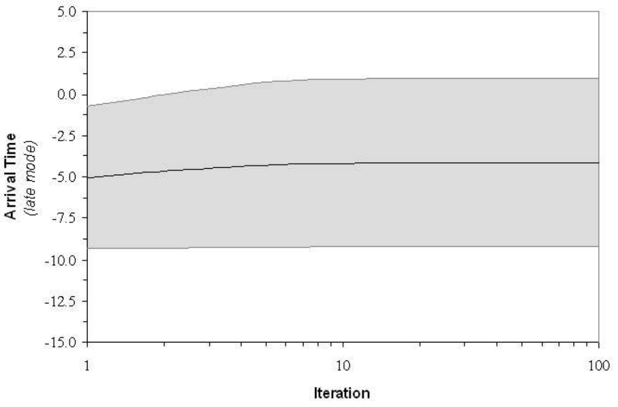

Figure 3: The statistical late mode arrival time (A_i) at an arbitrary register in the s1196 benchmark is plotted over the course of 1000 iterations of the statistical latch timing algorithm with δ =15%. The mean arrival time is the central line, with one standard deviation shaded on either side. The statistical distribution quickly converges to and remains fixed at its final distribution.

Table 1 shows a runtime comparison between the Monte Carlo simulation and the iterative analytical approximation, on both the full netlist and the graph of extracted path delays. While the initial results are promising, a full evaluation of the accuracy of the iterative analytic approximation is currently in progress.

5. ANALYSIS

The statistical latch scheduling heuristic was applied to 38 publicly available benchmarks, and selected results are presented. Despite the long runtime, all yield data are computed using a 10,000 iteration Monte Carlo simulation to provide an accurate and unapproximated characterization of latch timing under process variation.

Yield data are presented using the MSBF (mean successes between failure) metric, computed using Equation 5. The approximately Gaussian nature of yield curves does lend itself well to linear comparisons of raw data and often tempts evaluations on a irrelevant portion of the yield curve. A 50% improvement in yield from 0.50 to 0.75 is more numerically significant than a 2% improvement in yield form 0.97 to 0.99, even though the latter is more practically meaningful and typically more of an achievement. Furthermore, the economics of testing and product support associate significant cost with the rate of *failure*, not the yield. A direct comparison between the relative rates of failure better captures these properties.

$$MSBF = \frac{yield}{1 - yield} \tag{5}$$

Figure 4 plots the MSBF versus the duty cycle of an example benchmark design. As the duty cycle is increased, the design becomes increasing tolerant to process variation, resulting in dramatically fewer timing failures. The best latch scheduled design (at a duty cycle of about 15%) fails about 4 times less frequently than the same clock scheduled register-based design. Recall that the register-based behaves identically to the latch-based design operating at $\delta = 0$, the leftmost point in Figure 4.

Figure 4 also plots the fraction of designs that fail because of setup violations and/or hold violations for each value of δ. As the transparent window of the latch is widened, the circuit becomes more increasingly tolerant to setup violations, as more complementary longest path variations are balanced by the transparency of the latch. Depending on the tightness of the hold constraints in the design, further increases in the duty cycle begin to cause hold time failures that offset any further benefit from decreasing fraction of setup violations.

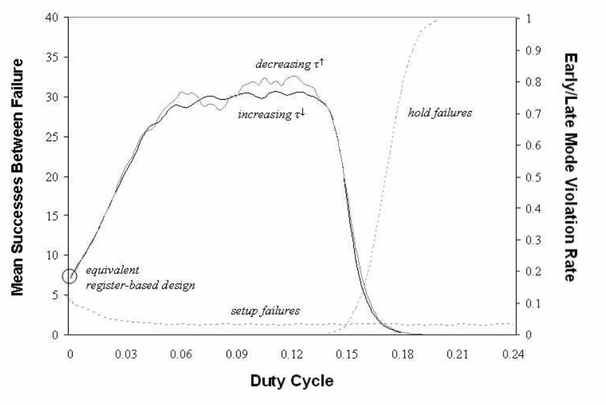

Figure 4: The MSBF of the latch-based s1488 design at different duty cycles. The duty cycle is adjusted by both increasing $\tau_i^{\downarrow}$ and decreasing $\tau_i^{\uparrow}$, and both methods are shown. The rate of setup and hold failures is overlaid. It's clear that the latch scheduling can greatly improve the yield of this design.

5.1 Hold Time Correction

In the deterministic case, a hold time violation can be corrected by inserting delays along the shortest path. If there is more than a single path, this additional delay will be added off of the critical path and not will affect the longest-path timing. In the statistical case, because the tightness of the hold time constraints limit the degree to which the duty cycle can be increased and the longest-paths balanced, a similar approach can be used to improve the efficacy of latch scheduling. However, one effect of the normal delay model is that all possible paths have a non-zero probability of being critical in some manufactured instance. Adding delay to any path—including ones that are likely to be short—increases the longest path delay. Any delay added to ease a hold time constraint should provide more benefit by enabling additional balancing with a longer duty cycle than it costs through direct increases in the longest path delay.

Figure 5 shows the results of this technique applied to two circuits. In both examples, delay has been added to bring the mean of the shortest path delay up to half of the period. It is clear that this is significantly more beneficial on the s1196 benchmark, allowing the latch scheduling to improve the failure rate by about 2.5 times more than was possible in the uncorrected design. The relative tightness of the setup and hold constraints in a design greatly affects the efficacy of latch scheduling.

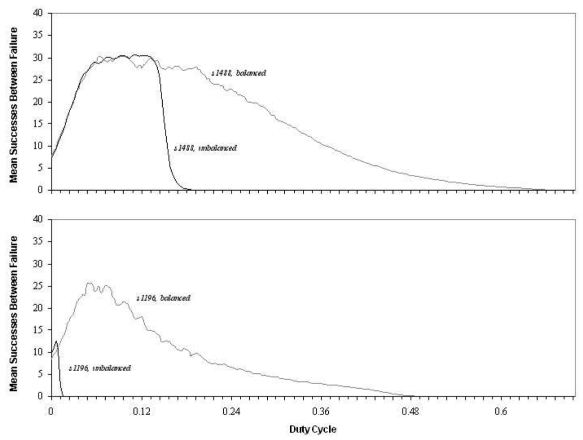

Figure 5: The MSBF of two latch-based variants of each of the s1488 (top) and s1196 (bottom) designs at different duty cycles. The dark lines show the yield curve vs. duty curve of the original circuit; the lighter shows the shortest-path balanced version. The maximum achievable yield is not improved for the s1488 design but is significantly improved for s1196, which was clearly more constrained by a tight hold time constraint.

Table 1: Runtime Results, Arbitrarily Selected Benchmarks

Benchmark	**# Latches**	**# Delay Paths**	**Netlist, M.C.** 1000 Iterations	**Netlist, Iter.** change < 0.01%	**Netlist, Iter.** change < 0.01%	**Graph, M.C.** 1000 Iterations	**Graph, Iter.** change < 0.01%
			Runtime (secs)	Num Iterations	Runtime (secs)	Runtime (secs)	Runtime (secs)
s27	4	14	0.155	6	0.108	0.014	0.001
s208	9	53	0.600	6	0.092	0.028	0.001
s400	22	133	1.44	6	0.126	0.054	0.003
s1196	19	57	7.93	32	0.475	0.038	0.003
s1488	7	49	8.33	8	0.235	0.028	0.001
s15850_opt	514	12151	52.9	5	1.156	7.928	0.166
s38417_opt	1565	33611	170	2	7.56	23.3	0.368
b12_opt	122	1585	18.7	3	0.465	0.78	0.015
b15	450	63670	106.0	4	1.83	24.9	0.585
elliptic3	1123	160893	85.4	2	4.70	59.7	1.62
tseng	386	5289	34.6	2	1.10	2.33	0.046
dsip	225	1309	27.7	2	0.71	0.54	0.012

6. CONCLUSIONS

While the optimal clock periods of latch- and register-based designs after clock scheduling appear identical under a traditional deterministic timing model, latches posses much more desirable properties when subjected to any process variation. A statistical analysis of their behavior in some common benchmarks shows that the number of failures can be reduced by 4 times. We discuss some methods for approaching this analysis and optimization in an efficient manner.

7. ACKNOWLEDGMENTS

This work was funded in part by the MARCO Focus Center for Circuit & System Solutions (C2S2, www.c2s2.org), under contract 2003-CT-888.

8. REFERENCES

[1] J. P. Fishburn, "Clock skew optimization*," IEEE Trans. on Computing*, vol. 39, pp. 945-951, July 1990.

[2] M. Orshansky, L. Milor, C. Pinhong, K. Keutzer, H. Chenming, "Impact of spatial intrachip gate length variability on the performance of high-speed digital circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 21, no. 5, pp. 544-553, May 2002.

[3] A. Devgan and C. Kashyap, "Block-based static timing analysis with uncertainty," in *Proceedings of the ICCAD*, pp. 607-614, 2003.

[4] H. Chang and S. S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single pert-like traversal," in *Proceedings of the ICCAD*, pp. 621-626, 2003.

[5] J. L. Neves and E. G. Friedman. "Optimal clock skew scheduling tolerant to process variations", in *Proceedings of the DAC*, pages 623--628, June 1996.

[6] K. Sakallah, T. Mudge, and O. Olukotun, "Timing verification and optimal clocking of synchronous digital circuits," in *Proceedings of the ICCAD*, pp. 552-555, 1990.

[7] T. G. Szymanski and N. Shenoy, "Verifying clock schedules," in *Proceedings of the ICCAD*, 1992.

[8] F. Mu and C. Svensson, "Pulsewidth control loop in high-speed CMOS clock buffers," in *IEEE Journal of Solid-State Circuit*, vol. 35, no. 2, pp. 134-141, Feb 2000.

[9] K.-H. Cheng, C.-W. Su, C.-L. Wu and Y.-L. Lo, "A phase-locked pulsewidth control loop with programmable duty cycle," in *Proceedings of IEEE Asia-Pacific Conference on Advanced System Integrated Circuits*, Aug. 2004.

[10] Hwang-Cherng Chow, "Duty cycle control circuit and applications to frequency dividers ," in *Proceedings of ICECS*, 1999.

[11] A.P. Hurst and R.K. Brayton, "Computing Clock Skew Schedules Under Normal Process Variation," in *Proceedings of the IWLS*, 2005.

[12] N. V. Shenoy and R. K. Brayton, "Graph algorithms for clock schedule optimization," in *Proc. of the ICCAD*, 1992.

[13] R. Chen and H. Zhou, "Clock schedule verification under process variations," in *Proceedings of the ICCAD*, November 2004.

[14] C. Clark, "The greatest of a set of random variables," *Operations Research*, 1961.

[15] W. H. Press, W. T. Vetterling, S. A. Teukolsky, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 2 ed., 1992.