# A Practical Gate Resizing Technique Considering Glitch Reduction for Low Power Design

Masanori Hashimoto, Hidetoshi Onodera and Keikichi Tamaru
Department of Communications and Computer Engineering,Kyoto University
E-mail: hasimoto@tamaru.kuee.kyoto-u.ac.jp

## Abstract

We propose a method for power optimization that considers glitch reduction by gate sizing based on the statistical estimation of glitch transitions. Our method reduces not only the amount of capacitive and short-circuit power consumption but also the power dissipated by glitches which has not been exploited previously. The effect of our method is verified experimentally using 8 benchmark circuits with a 0.6 $\mu$m standard cell library. Our method reduces the power dissipation from the minimum-sized circuits further by 9.8% on average and 23.0% maximum. We also verify that our method is effective under manufacturing variation.

## 1 Introduction

The dynamic power dissipation, which is the dominant source of power dissipation, is directly related to the number of signal transitions in a circuit. A signal transition can be classified into two categories; a functional transition and a glitch. It is well known that glitches occupy a considerable amount in the signal transitions of a circuit. Reference[1] indicates that the glitch power dissipation accounts for 20% to 70%, and Ref.[2] says 7% to 43%. Also glitches are extremely sensitive to delay characteristics[3]. Therefore glitch reduction by optimizing delay characteristics is a reasonable approach for power reduction.

We propose a power optimization method considering glitch reduction by gate sizing. Conventional approaches for power reduction optimize the amount of capacitive load[4, 6] or the amount of capacitive load and short-circuit current[5, 7] based on the transition activity information obtained beforehand. None of the conventional approaches explicitly optimize the number of transitions for power reduction. Our method reduces not only the amount of capacitive and short-circuit power consumption but also the power dissipated by glitches explicitly.

Our optimization method consists of two techniques; a statistical estimation method of glitch activities and an optimization algorithm for gate resizing. For the estimation of glitch activities, we classify glitches into two classes; generated glitches and propagating glitches. As for the generated glitches, we adopt a statistical estimation method proposed by Lim and Soma[8]. The propagating glitches, however, are not considered in their method, and therefore we have devised a statistical estimation method. The optimization algorithm has been designed to have the ability of escaping from a bad local solution while keeping small computational costs.

A preliminary version of this method is reported in [9], where the effect of manufacturing variability is not taken into account. In real circuits, there exist statistical perturbations of circuit parameters such as skew fluctuations and deviations in gate delay, which may affect glitch activities and thereby cannot be neglected. Also, not all glitches have full-swing transitions. Treating all glitches as full-swing transitions may cause an excessive overestimation of glitch power dissipation. In this paper, we propose a practical power optimization method considering actual phenomena, such as skew fluctuations and partial-swing transitions.

The circuit under optimization is a CMOS combinational circuit designed in a synchronous design style. This paper is organized as follows. Section 2 discusses the statistical glitch estimation method considering propagating glitches, skew fluctuations and partial-swing transitions. Section 3 explains the optimization algorithm of gate resizing. Section 4 shows some experimental results. Finally Section 5 concludes the discussion.

## 2 Statistical Glitch Estimation

In this section, we explain an estimation method for glitch activities based on a statistical approach. Glitches can be separated into the following two components.

**generated glitches:** the glitches that are generated by functional (non-glitch) transitions.

**propagating glitches:** the glitches that are generated previously at a gate in the fan-in direction and propagate through the gate.

As for the generated glitches, a statistical estimation method is proposed by Lim and Soma[8]. However, the effect of propagating glitches is not taken into account. Some part of the generated glitches may be immediately blocked by the fan-out gates. Other part, however, will propagate through the circuit until they are suppressed or reach to primary outputs. Therefore the effect of the propagating glitches cannot be neglected.

The voltage swing of glitches is not always $V_{DD}$. The energy dissipated by charging and discharging the load capacitance is proportional to the voltage swing. Treating all glitches as full-swing transitions cause an overestimation of the power dissipated by glitches. Therefore we improve the estimation method of the generated glitches[8] such that the power dissipated by partial-swing transitions can be considered.

In real circuits, there exist uncertainties in delay characteristics, which may spoil the effect of power optimization. For example, after a clock distribution tree is designed, the skew time at each flip-flop(latch) can be estimated. However, the estimated skew time has some errors. Also, the skew time fluctuates owing to the statistical variation of the transistor characteristics and the wire capacitance. We therefore contrive the estimation method that can consider skew fluctuations. This consideration increases the tolerance of glitch reduction to actual phenomena in real circuits.

### 2.1 Preparations

We define the primary input signal $x[n] = x(t)|_{t=nT}$, where $n$ is an integer and $T$ is the period of the system clock. The signal probability $P(x)$ and the transition density $D(x)$ are defined as follows[10].

$$P(x) = \lim_{k\to\infty} \frac{1}{k} \sum_{n=1}^{k} x[n], \qquad (1)$$

$$D(x) = \lim_{k \to \infty} \frac{1}{k} \sum_{n=1}^{k} |x[n] - x[n-1]|_{x[0]=x_0}. \quad (2)$$

The switching probabilities $P^{00}(x)$, $P^{01}(x)$, $P^{10}(x)$ and $P^{11}(x)$ are the probabilities that the signal of gate $x$ changes as $0 \to 0$, $0 \to 1$, $1 \to 0$, $1 \to 1$, respectively[9]. Transition rate $R(x)$ is defined as

$$R(x) = \lim_{t \to \infty} \frac{n_x(t)}{t}, \quad (3)$$

where $n_x(t)$ is the number of transitions of $x(t)$ between a time interval of length $t$.

In order to consider short-circuit power dissipation, we utilize a power estimation method based on look-up tables. In this method, the total power dissipation $PW$, including short-circuit power dissipation, is represented as follows.

$$PW = \frac{1}{2} \sum_{i}^{n} PW_{table}(i) R(i), \quad (4)$$

where $n$ is the number of gates and $PW_{table}(i)$ is the energy that is consumed at the gate $i$ when the output changes. The values of $PW_{table}(i)$ are given by look-up tables. The look-up tables are two-dimension tables with load capacitance and input transition time as variables and they are characterized beforehand by circuit simulation.

We derive path delays using a static timing calculation method. A delay time at each gate is calculated based on two dimensional look-up tables with capacitive load and slew as parameters.

### 2.2 Previous Work on Generated Glitch

First, we briefly explain the estimation method for generated glitches[8]. The condition for glitch generation is to hold the following two conditions simultaneously(Fig. 1).

**Condition 1:** The input pattern $\omega_k$ is the pattern that can cause glitches.

**Condition 2:** The interval time $\zeta$ between successive transitions at different inputs is larger than the delay $\tau$.

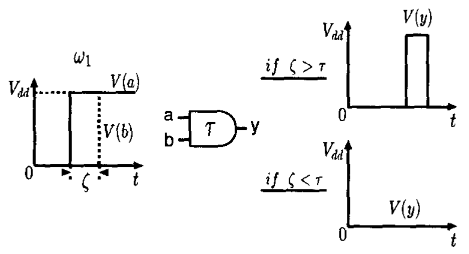


Figure 1: An input pattern and condition for glitch generation in a 2-input AND gate.

We calculate the probability satisfying Condition 1 and the probability satisfying Condition 2 separately. The pattern probability $P_{patt}(\omega_k)$ is the probability that the input pattern $\omega_k$ occurs. The generation probability $P_{gen}(\omega_k)$ is the probability that the input pattern $\omega_k$ satisfies Condition 2, and can be represented as follows:

$$P_{gen}(\omega_k) = \int\int_{A_k} f(\alpha) f(\beta) d\alpha d\beta, \quad (5)$$

where $\alpha$ and $\beta$ are the arrival times of the respective signals in $\omega_k$, $f$ is the distribution function that represents the number of transitions as a function of arrival time. $A_k$ is the area that satisfies Condition 2 in the $\alpha - \beta$ space(Example, Fig. 2). Using $P_{patt}$ and $P_{gen}$, generated glitch rate $R_{gen}(i)$ is represented as follows.

$$R_{gen}(i) = f_{clk} \cdot \sum_{k} \{P_{gen}(\omega_k) \cdot P_{patt}(\omega_k)\}. \quad (6)$$

The rigorous derivation of the distribution function $f$ requires two processes. The first process is to search all paths and calculate the delay of each path. The second process is to evaluate the activating probability of each path. The complexities of these processes are practically infeasible. We therefore propose to use an uniform distribution. The uniform distribution function $f$ is represented as follows:

$$f(t) = \frac{1}{\alpha_{max} - \alpha_{min}} \cdot \{U(t - \alpha_{min}) - U(t - \alpha_{max})\}, \quad (7)$$

where $\alpha_{max}$ is the latest arrival time, $\alpha_{min}$ is the fastest arrival time, and $U$ is the step function.

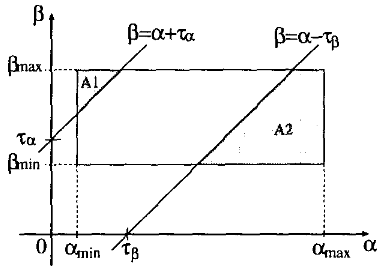


Figure 2: Surface integral area of the distribution function $f$. Parameters $\alpha_{min}(\beta_{min})$ and $\alpha_{max}(\beta_{max})$ represent the earliest and the latest arrival times respectively. Parameter $\tau_\alpha(\tau_\beta)$ represents the delay time of signal $\alpha(\beta)$.

### 2.3 Propagating Glitch

We define the propagating glitch rate $R_{prop}(x)$ as follows:

$$R_{prop}(x) = \lim_{t \to \infty} \frac{n_{prop\_x}(t)}{t}, \quad (8)$$

where $n_{prop\_x}(t)$ is the number of propagating glitches at the gate $x$ between a time interval of length $t$. From the definitions, total transition rate $R$ can be represented using $D$, $R_{gen}$, $R_{prop}$ and $f_{clk}$ as follows:

$$R(x) = f_{clk} \cdot D(x) + 2 \cdot \{R_{gen}(x) + R_{prop}(x)\}. \quad (9)$$

The multiplication factor of two in the second term comes from that a single glitch causes two transitions.

Now, we explain an estimation method of the propagating glitch rate $R_{prop}$. If the inputs of a gate have no correlation with each other and there is a sufficient time interval between the input transitions, the following equation holds at any gates[10].

$$R(y) = \sum_{i=1}^{n} P(\frac{\partial y}{\partial x_i}) R(x_i), \quad (10)$$

where $x_i$ is the $i$-th input of the gate, $y$ is the output and $n$ is the total number of inputs. From the definition of $R_{prop}$, if the glitches at the inputs have no correlation and have sufficient time interval between the transitions, $R_{prop}$ can be represented as follows.

$$R_{prop}(y) = \sum_{i=1}^{n} P(\frac{\partial y}{\partial x_i}) \cdot \{R_{gen}(x_i) + R_{prop}(x_i)\}. \quad (11)$$

Equation (11) assumes that there is a sufficient time interval between the transitions, so this equation may overestimate propagating glitches. There is a possibility that the overestimation of propagating glitches at each gate causes an excessive overestimation along with the signal propagation. Therefore we should estimate the lower bound of propagating glitches. Let us consider the situation that a glitch comes from the input a in a 2-input AND gate (Fig.3). If the input b retains high, the glitch propagates through the gate. If the input b keeps low, the glitch never propagates through the gate. But if there is a transition at the input b, glitch propagation through the gate depends on the timing of the transition. In order to take the lower bound of the estimation, we neglect the timing-dependent glitch propagation. Therefore the propagating glitch rate is represented as follows.

$$R_{prop}(y) = \sum_{i=1}^{n}\{R_{gen}(x_i) + R_{prop}(x_i)\} \cdot P(\frac{\partial y}{\partial x_i})\Big|_{P^{10}=P^{01}=0}. \quad (12)$$

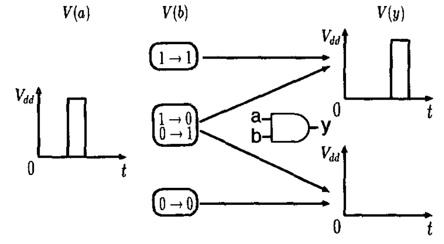


Figure 3: The condition that allows a glitch propagating through a 2-input AND gate.

## 2.4 Partial-Swing Transitions

The energy dissipated by charging and discharging the load capacitance $C$ is proportional to the voltage swing. When the voltage swing is $V_{DD}/2$, the dissipated energy, which is represented as $C \cdot \frac{V_{DD}}{2} \cdot V_{DD}$, is the half of the energy of a full-swing transition. Treating a partial-swing transition as a full-swing transition causes an overestimation of the energy dissipated by glitches. Therefore we propose an approach such that a partial-swing transition is converted into an equivalent fraction of a full-swing transition based on the dissipated energy. For example, a transition that the voltage swing is $V_{DD}/2$ is regarded as 0.5 transition.

Fig. 4 shows the relationship between the voltage swing $V_{SW}$ and the difference of the arrival time$\gamma$(=$\alpha - \beta$ or $\beta - \alpha$) in a 2-input NAND gate. We examine the relationship under two output load conditions by circuit simulation, and approximate it as a linear function:

$$V_{SW} = \begin{cases} \frac{V_{DD}}{2\tau}\gamma & 0 \le \gamma \le 2\tau \\ V_{DD} & \gamma > 2\tau \end{cases}, \quad (13)$$

where $\tau$ represents the delay of the gate. Similarly, in the other gates, such as multi-stage gates, we examine the relationship between $V_{SW}$ and $\tau$, and approximate it as a linear function.

Using this conversion, we can improve Eq. (5) as follows.

$$P_{gen}(\omega_k) = \int\int f(\alpha)f(\beta)h(\alpha,\beta)d\alpha d\beta, \quad (14)$$

$$h(\alpha,\beta) = \frac{V_{SW}(\alpha,\beta)}{V_{DD}}. \quad (15)$$

When the distribution function $f$ is uniform, $h(\alpha,\beta)$ of Eq. (15) can be transformed as follows.

$$h(\alpha,\beta) = \begin{cases} U(\alpha-\beta-\tau_\beta') & 0 \le \alpha-\beta \\ U(\beta-\alpha-\tau_\alpha') & 0 \le \beta-\alpha \end{cases}, \quad (16)$$

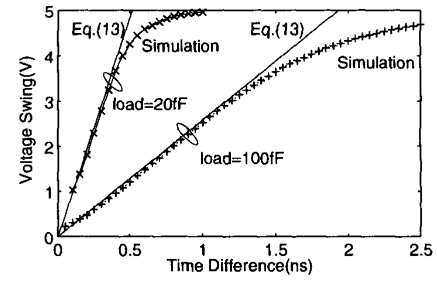


Figure 4: Relationship between the swing voltage and the difference of the arrival time in 2-input NAND gate.

where $\tau'$ is derived from the following equation.

$$\int U(\gamma - \tau')d\gamma = \int h(\gamma)d\gamma. \quad (17)$$

In the case of Eq.(13), $\tau_\alpha'(\tau_\beta')$ becomes $\tau_\alpha(\tau_\beta)$.

Using Eqs.(7) and (16), Eq.(14) can be transformed as follows(Fig.5).

$$\begin{aligned} P_{gen}(\omega_k) &= \int\int_{A_k'} f(\alpha)f(\beta)d\alpha d\beta & (18)\\ &= \frac{\text{area}(A_k')}{(\alpha_{max}-\alpha_{min})(\beta_{max}-\beta_{min})}, & (19)\end{aligned}$$

where area($A_k'$) represents the shaded area in Fig. 5.

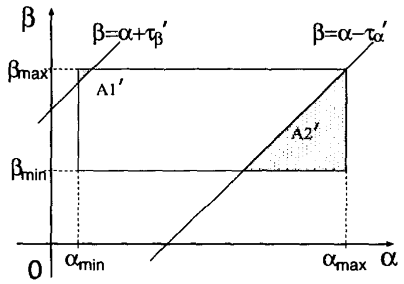


Figure 5: Surface integral area of the distribution function $f$ considering partial-swing transitions.

## 2.5 Skew Fluctuation

After a clock distribution tree is designed, the skew time at each flip-flop(latch) can be estimated. However, the estimated skew time has certain amount of estimation errors. Also, the skew time varies due to manufacturing variability. We therefore should consider skew fluctuation in glitch estimation. We assume that the distribution of the skew time is normal($\mu$, $\sigma$) and $\mu$ is the estimated skew time. The skew at each primary input appears as the skew in the arrival time at the input of each gate. The distribution of the skew is well approximated by normal[11]. Hence we approximate $P_{gen}(\omega_k)$ under skew fluctuation as a weighted average over five sampling points.

$$\begin{aligned} P_{gen}(\omega_k) &= 0.404\int\int_{A_k'} f(\alpha)f(\beta)d\alpha d\beta & (20)\\ &+ \sum 0.149\int\int_{A_k'} f(\alpha\pm\sigma)f(\beta\pm\sigma)d\alpha d\beta \end{aligned}$$

## 3 Optimization Algorithm for Power Reduction

We execute discrete (cell-based) gate sizing for power reduction of a CMOS combinational circuit using the proposed estimation method. We develop a heuristic algorithm that has both the merit of rapid convergence and the ability to get out of a bad local solution. In this section, we explain the algorithm under delay constraints. A flow-chart of the algorithm is shown in Fig. 6.

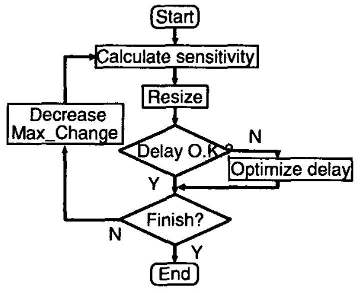


Figure 6: The optimization algorithm under delay constraints.

**Calculate sensitivity:** At each gate, we evaluate the sensitivity of the objective function Eq.(4) both for sizing-up and sizing-down operations. If a sizing operation violates delay constraints or the signal transition time exceeds a pre-defined value, we don't calculate the sensitivity and eliminate the operation from candidates.

**Resize:** We select gates according to the sensitivity and resize them. The number of the gates resized simultaneously is at most *Max_Change*.

**Delay O.K.?:** There is a possibility that a timing violation occurs because we resize at most *Max_Change* gates at once. We judge whether the delay constraints are satisfied or not.

**Optimize delay:** The circuit is optimized by a similar algorithm to this power optimization using delay sensitivity with respect to sizing operations until the delay constraints are satisfied.

**Finish?:** If the iteration count goes over a pre-defined value *Max_Iteration*, or if no gates are resized, the optimization procedure finishes.

**Decrease** *Max_Change*: We reduce *Max_Change* by a factor of *Reduce_Rate*.

Since we resize at most *Max_Change* gates at a time, there is no guarantee for improvement. The evaluated sensitivity for each gate is only valid for single resizing of the corresponding gate. This simultaneous resizing is regarded as a perturbation to the circuit. The amount of perturbation is reduced as the number of *Max_Change* is decreased through the iteration.

In the beginning of the optimization, i.e., when *Max_Change* is large, many gates are resized simultaneously. In this case, the amount of perturbation is large, and solution space is expected to be explored globally. Parameter *Max_Change* is gradually reduced at the rate of *Reduce_Rate*, and the amount of perturbation decreases. The gradual reduction of *Max_Change* has a similar role to the temperature reduction in simulated annealing. The ratio of reduction can control the speed of convergence and the search area of solutions. At the final stage, *Max_Change* becomes small and this algorithm behaves like a greedy algorithm. A greedy algorithm is suitable for finding a local optimal solution, which merit is exploited in our algorithm at the final stage. With the help of the perturbation and the greediness, our algorithm can find a solution close to that of the simulated annealing in less than one hundredth of CPU time[9].

## 4 Experimental Results

In this section, we show some experimental results. First, we verify the accuracy of our glitch estimation method. Next, we demonstrate power optimization results and verify the effectiveness of our method. Finally, we show that our method can reduce glitches under the fluctuation of skew times and wire capacitances.

The circuits used for the experiments are taken from ISCAS85 and LGSynth93 benchmark sets(See Table 1). These circuits are synthesized and mapped by a commercial logic synthesis tool[12] such that the area is minimized under the delay constraints of 10ns. The target library is a standard cell library used for actual fabrication in a 0.6 $\mu$m process with three metal layers. The library includes basic and complex gates. Buffer and Inverter have six varieties in the driving strength and other gates have three varieties. The transition density $D$ and signal probability $P$ at each gate are calculated by logic simulation. The power dissipation is evaluated by a commercial transistor-level power simulator[13]. Input patterns are randomly generated with a signal probability of 0.5. The number of applied patterns is 100, which is the adequate number for the power estimation at circuit level[2]. The cycle time of the input patterns is 100ns. The constants *Max_Iteration*, *Reduce_Rate* and initial *Max_Change* are set to 50, 0.90, 0.4$\times$(number of gates), respectively. The objective function is Eq. (4) which represents dynamic power dissipation including short-circuit power dissipation.

### 4.1 Glitch Estimation

Now we examine the accuracy of our glitch estimation method. We estimate the number of glitch transitions at every node in a circuit and compare it to the value obtained by logic simulation. We estimate the glitch transitions in the following two ways.

**Conventional Method:** Only generated glitches are estimated (equivalent to [8] except for the simplified calculation of $f$ function).

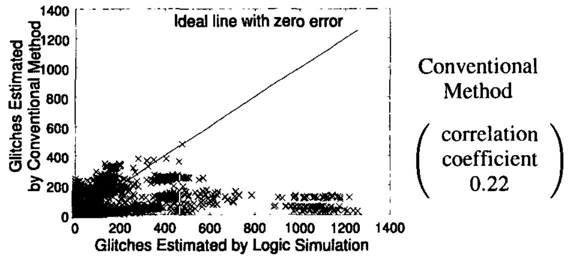
**Proposed Method:** Both generated and propagating glitches are estimated.

Fig. 7 compares the accuracy of the number of estimated glitches between the conventional method and the proposed method in des circuit. The horizontal axis represents the number of glitches estimated by logic simulation. The vertical axis represents the number of glitches estimated by the conventional method or the proposed method. The correlation coefficient is calculated between simulated values and estimated values. The correlation coefficient of the proposed method is 0.79, whereas the coefficient of the conventional method is 0.22 in des circuit. The average correlation coefficients of the proposed method over 8 benchmark circuits are 0.78 and the coefficients of the conventional method is 0.33.

Table 1: Accuracy of proposed power estimation method.

| Circuit | Simulation | | Conv. | Proposed | | |
|---|---|---|---|---|---|---|
| | Power (mW) | Time (s) | Error (%) | Error (%) | Time (s) | #gates |
| C3540 | 21.3 | 149 | -45.5 | -19.2 | 0.04 | 959 |
| alu4 | 17.2 | 105 | -9.3 | 5.2 | 0.07 | 1386 |
| pair | 18.4 | 159 | -21.2 | -9.8 | 0.06 | 1399 |
| misex3 | 13.3 | 216 | -12.0 | 9.8 | 0.06 | 1416 |
| C5315 | 36.3 | 527 | -40.2 | -18.5 | 0.08 | 1574 |
| i10 | 31.5 | 273 | -40.3 | -19.7 | 0.11 | 2075 |
| C7552 | 60.5 | 713 | -47.7 | -22.8 | 0.11 | 2114 |
| des | 42.6 | 672 | -27.0 | -6.6 | 0.19 | 2793 |
| average$^{\dagger}$ | - | - | 30.4 | 14.0 | - | - |

average$^{\dagger}$: the average over the absolute amount of each error.



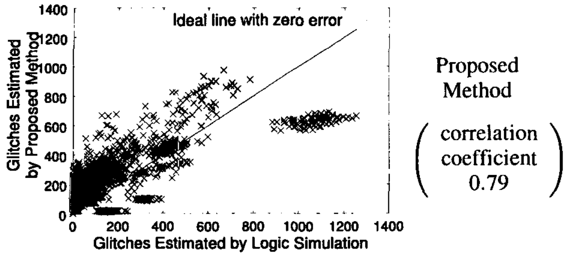


Figure 7: Accuracy comparison of glitches between conventional and proposed method (des).

We examine the accuracy of the estimated power dissipation. We estimate the power dissipation using the proposed glitch estimation method that can consider propagating glitches and partial-swing transitions. Table 1 shows the result of power estimation. The column "Power" under "Simulation" represents the power dissipation evaluated by a transistor-level power simulator. The column "Error" under "Proposed(Conv.)" represents the estimation error of the proposed(conventional) method. The column "Time" represents the CPU time for power estimation on a SUN Ultra2. The average error of the proposed method is 14.0%, whereas the error of the conventional method that ignores the propagating glitches is more than 30%. The CPU time required for the proposed method is more than 1000 times shorter than that for a transistor-level power simulator, which enables to use the estimation method inside the optimization loop considering glitch reduction.

## 4.2 Power Optimization

Here, we show the result of power optimization. First, we optimize power dissipation without delay constraints. The initial circuits used in this experiment consist of the min-sized gates. The min-sized circuits means that the overall capacitive load is minimum. Table 2 shows the result of the power optimization. The power dissipation before/after optimization is evaluated by a transistor-level power simulator. The column "Power(Delay) Reduction" represents the reduction of the power(delay) from the min-sized circuit. The column "Area Increase" shows the increase of the total cell area from the initial circuit. Our method increases the area by 4.5% on average, but the power dissipation is reduced by 9.8% because of glitch reduction. This means that the power dissipation of the circuits with the minimum active area is not minimum. It is notable that the delay is also reduced in all circuits, although the delay is not included in the objective function nor the constraints. The reduction of delay is 26.1% on average. Glitch reduction has an aspect of path balancing. In this experiment, we use the minimum-sized circuits as starting points. The path balancing is enforced by reducing longer path delays, which leads to the reduction of the critical path delay.

Next we present the result of power optimization under delay constraints and compare the result with those of conventional methods. We optimize the circuit C5315 under a variety of delay constraints and measured power dissipation using a transistor-level power simulator. The circuit is optimized in the following three methods.

Table 2: Power optimization under no delay constraints.

| Circuit | Power Reduction(%) | Delay Reduction(%) | Area Increase(%) |
|---|---|---|---|
| C3540 | 8.0 | 12.6 | 4.2 |
| alu4 | 14.8 | 36.9 | 4.0 |
| pair | 4.3 | 26.4 | 3.0 |
| misex3 | 7.5 | 26.3 | 5.6 |
| C5315 | 11.5 | 10.8 | 4.3 |
| i10 | 5.2 | 22.6 | 6.1 |
| C7552 | 23.0 | 45.6 | 3.1 |
| des | 3.9 | 27.9 | 5.3 |
| average | 9.8 | 26.1 | 4.5 |

**Delay Optimization:** optimize delay only and do not care about power dissipation.

**Conventional Method:** optimize power dissipation based on the transition information of the initial circuit throughout the optimization process.

**Proposed Method:** optimize power dissipation by the proposed method.

The power-delay trade-off curve of each method is shown in Fig. 8. The initial and the min-sized circuits are located near the top right corner of the figure. The initial circuit is the circuit after logic synthesis. Achievable delay times by the three methods are the same. The fastest circuits by the three methods have 5.6ns delay time. However the power dissipation is different and, as expected, the proposed method provides the lowest. Because the reduction of the delay time and path balancing lie in the same direction, it is seen that delay reduction does not increase power dissipation so much. Indeed, the fastest circuit obtained by the delay optimization method has the total cell area 13 % larger than that of the initial circuit, while the power dissipation is almost the same as that of the initial circuit. Corresponding increase in capacitive load is compensated by the reduction of glitch activity which is a by-product of the delay optimization. The conventional method which assumes constant glitch activities throughout the optimization process does not work well. It is because the glitch activities are changing in the optimization process. In order to reach good solutions, we have to consider the fact that glitches are affected by gate resizing seriously. Explicitly exploiting the possibility of glitch reduction, the proposed method further reduces the power dissipation. We can see that the gate sizing considering glitch reduction is an effective method for power reduction.

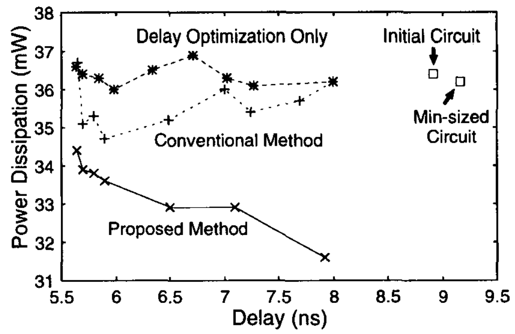


Figure 8: Power-delay trade-off curve (C5315).

### 4.3 Tolerance to Skew Fluctuation and Wire Capacitance Variation

In actual circuits, there are various factors that change delay characteristics, such as skew fluctuations, variations in transistor characteristics and wire capacitances. We examine the tolerance of our method to uncertainties in delay characteristics.

First, we evaluate power dissipation under skew fluctuations. The skew time at each primary input is assumed to fluctuates according to the normal distribution(0, $\sigma$). We optimize power dissipation by the following two methods.

**Sizing(A):** optimization that does not consider skew fluctuations, i.e. only the first term in Eq. (20) is considered.

**Sizing(B):** optimization that considers skew fluctuations(Sec. 2.5).

We generate 100 sets of skew patterns for $3\sigma$ of skew fluctuation being 0.5ns and 1.0ns. In this fabrication process, the delay time of a single inverter with fanout loading three is 0.1ns. Fig. 9 shows the relationship between the amount of power reduction and skew fluctuations for C5315 circuit. We can see that our method can reduce power dissipation under skew fluctuation. The consideration of the skew fluctuation results in the increase in the worst value of the power reduction by 1%. In the case of $3\sigma = 0.5$ns, Sizing(B) is much effective than Sizing(A). We guess the reason such that the consideration for skew fluctuation compensates not only skew fluctuations but also the error in delay calculation as a by-product.

Because of manufacturing variability, wire capacitance fluctuates. Also wire load estimation contains a certain amount of error. Therefore the gate delay has some amount of uncertainty. We evaluate power dissipation under wire load fluctuations. The fluctuation of wire capacitance is assumed to have a normal distribution(0, $\sigma$). We generate 100 sets of wire load and evaluate power dissipation. The ratio of total gate capacitance and the total wire capacitance is about 1:2 in this circuit. The relationship between power reduction and the amount of wire capacitance fluctuations for C5315 circuit is shown in Fig.10. The average reduction at each $3\sigma$ value is almost the same. Even in the worst case of $3\sigma = 40\%$, the power dissipation is reduced by 10.9%. We can see that our method is effective under uncertainties in delay characteristics that exist in fabricated circuits.

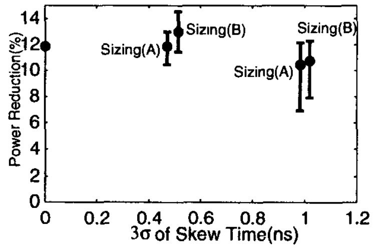


Figure 9: Power reduction under skew fluctuations (C5315).
Sizing(A): does not consider skew fluctuations.
Sizing(B): considers skew fluctuations.

## 5 Conclusion

We propose a power optimization method by gate sizing. Our method optimizes not only the amount of capacitive load and short-circuit current but also the number of glitch transitions. We devise a statistical glitch estimation method that can consider propagating glitches, partial-swing transitions and skew fluctuation. Our gate resizing algorithm has both the merit of rapid convergence and the ability to get out of a bad local solution. The effect of our method is experimentally verified using 8 benchmark circuits with a 0.6 $\mu$m standard cell library. The power dissipation is reduced

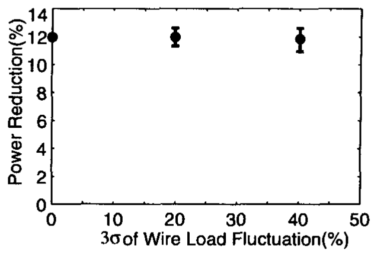


Figure 10: Power reduction under wire capacitance fluctuations (C5315).

from the minimum-sized circuits by 9.8 % on average and by 23.0 % maximum. We observe that the conventional method, which assumes that glitches do not change by gate resizing, does not achieve sufficient power reduction. On the other hand, our method can reduce power dissipation further guided by the glitch estimation method. We also verify that our method is effective under manufacturing variability such as skew time fluctuation and wire capacitance variation.

## Acknowledgments

This work is supported in part by Semiconductor Technology Academic Research Center (STARC).

## References

[1] A. Shen, A. Ghosh, S. Devadas and K. Keutzer, "On average power dissipation and random pattern testability of CMOS combinational logic networks," *Proc. ICCAD*, pp. 402--407, 1992.

[2] D. Brand and C. Visweswariah, "Inaccuracies in power estimation during logic synthesis," *Proc. ICCAD*, pp. 388--394, 1996.

[3] F. N. Najm and M. Y. Zhang, "Extreme delay sensitivity and the worst-case switching activity in VLSI circuits," *Proc. DAC*, pp. 623--627, 1995.

[4] Y. Tamiya and Y. Matsunaga, "LP based cell selection with constraints of timing, area, and power consumption," *Proc. ICCAD*, pp. 378--381, 1994.

[5] M. Borah, R. M. Owens, and M. J. Irwin, "Transistor sizing for minimizing power consumption of CMOS circuits under delay constraint," *Proc. ISLPD*, pp. 167--172, 1995.

[6] H.-R. Lin and T. T. Hwang, "Power reduction by gate sizing with path-oriented slack calculation," *Proc. ASP-DAC*, pp. 7--12, 1995.

[7] S. S. Sapatnekar and W. Chuang, "Power vs. delay in gate sizing: Conflicting objectives?," *Proc. ICCAD*, pp. 463--466, 1995.

[8] Y. Je Lim and M. Soma, "Statistical estimation of delay-dependent switching activities in embedded CMOS combinational circuits," *IEEE Trans. on VLSI Systems*, vol. 5, no. 3, pp. 309--319, September 1997.

[9] M. Hashimoto, H. Onodera, and K. Tamaru, "A power optimization method considering glitch reduction by gate sizing," *Proc. ISLPED*, pp. 221--226, 1998.

[10] F. N. Najm, "Transition density, a stochastic measure of activity in digital circuits," *Proc. DAC*, pp. 644--649, 1991.

[11] M. Berkelaar, "Statistical delay calculation, a linear time method," *Proc. TAU*, pp. 15--24, 1997.

[12] Synopsys Inc., *Desigin Compiler Reference Manual*, 1998.

[13] Synopsys Inc., *PowerMill Reference Manual*, 1998.