A Power Optimization Method Considering Glitch Reduction by Gate Sizing

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 power optimization method considering glitch reduction by gate sizing. 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. In the optimization method, we improve the accuracy of statistical glitch estimation method and device a gate sizing algorithm that utilizes perturbations for escaping a bad local solution. The effect of our method is verified experimentally using 12 benchmark circuits with a 0.5 μm standard cell library. Gate sizing reduces the number of glitch transitions by 38.2 % on average and by 63.4 % maximum. This results in the reduction of total transitions by 12.8 % on average. When the circuits are optimized for power without delay constraints, the power dissipation is reduced by 7.4 % on average and by 15.7 % maximum further from the minimum-sized circuits.

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(steady-state transition) and a spurious transition(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 signal propagation characteristics(delay)[3]. If we properly optimize timing characteristics such that the number of glitches is minimized, and if the area(power) cost for the optimization is small, we can expect that the power cost is well overcompensated and overall power dissipation is reduced by the glitch reduction.

Gate sizing is an effective method for delay optimization and many solutions are proposed such as Refs.[4, 5, 6]. Gate sizing has been utilized not only for delay optimization but also for power optimization[7, 8, 9, 10]. The main idea of previous approaches for power reduction is to optimize the amount of capacitive load[7, 8] or the amount of capacitive load and short-circuit current[9, 10] based on the transition activity information obtained beforehand. The transition activity, however, is affected by the sizing operation, which is not considered in the optimization. Although Ref.[10] proposes to update the transition information a few times during the optimization, it is not enough to fully consider the sensitivity of glitch activity with respect to timing modification cased by a sizing operation. None of the previous approaches explicitly optimize the number of transitions for power reduction. In this paper, we propose a power optimization method considering glitch reduction by gate sizing. Our method utilizes the sensitivity for reducing power consumed by glitches.

Our optimization method consists of two techniques; a statistical estimation method of glitch activities and an optimization algorithm for gate sizing. 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[11]. The propagating glitches, however, are not considered in the method[11], and therefore we have developed a statistical estimation method. The optimization algorithm needs hill-climbing ability or perturbation process because the power optimization is an ill-behaved problem. A simple greedy algorithm can easily get trapped in a bad local optimal solution. Simulated annealing, however, requires much computational costs. We therefore devise an optimization algorithm which has the ability to escape from a bad local solution while keeping small computational costs.

The target of our optimization method is a CMOS combinational circuit designed in a synchronous design style. This paper is organized as follows. Section 2 discusses the glitch estimation method based on a statistical approach. Section 3 explains the optimization algorithm of gate sizing for reducing power dissipation. Section 4 shows some experimental results of our method. Finally Section 5 concludes the discussion.

2 Glitch Estimation Based on a Statistical Approach

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 which are generated by steady-state (non-glitch) transitions.

propagating glitches the glitches which are generated by the glitch transitions propagating from fan-in gates.

The first component corresponds to newborn glitches at gate outputs produced by non-glitch transitions of their input signals. Second component corresponds to the glitches produced by glitch transitions of their input signals. Second component, in other words, represents the glitches which are generated previously at a gate in the fan-in direction and propagate through the gate. Hereafter we use the term ‘‘generated glitch‘‘ to refer to the first component and the term ‘‘propagating glitch‘‘ to the second component.

As for the estimation of generated glitches, a statistical approach is proposed by Lim and Soma[11]. However, the effect of propagating glitches are 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 before they are suppressed. Therefore the effect of the propagating glitches cannot be neglected. In this section, we first briefly explain the statistical method for the estimation of generated glitches[11]. We then propose an estimation method for propagating glitches. With these two methods, we are ready to optimize the circuit for reducing glitches.

ISLPED 98, August 10-12, 1998, Monterey, CA USA
© 2000 ACM ISBN 1-58113-059-7/98/08…$5.00

2.1 Definitions

We define the primary input signal $x[n]$, a synchronized discrete-time logic signal as

$$x[n] = x(nT) = x(t)|_{t=nT}, \tag{1}$$

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[12].

$$P(x) = \lim_{k\to\infty} \frac{1}{k} \sum_{n=1}^{k} x[n], \tag{2}$$

$$D(x) = \lim_{k\to\infty} \frac{1}{kT} \sum_{n=1}^{k} |x[n] - x[n-1]|_{x[0]=x_0}. \tag{3}$$

The switching probabilities $P^{00}(x)$, $P^{01}(x)$, $P^{10}(x)$, $P^{11}(x)$ are the probabilities that the signal of gate x changes as $0 \to 0$, $0 \to 1$, $1 \to 0$, $1 \to 1$, respectively. These probabilities have the following relations.

$$P^{00}(x) + P^{01}(x) + P^{10}(x) + P^{11}(x) = 1, \tag{4}$$

$$P^{01}(x) = P^{10}(x) = \frac{D(x)}{2}, \tag{5}$$

$$P^{11}(x) + P^{10}(x) = P(x). \tag{6}$$

Transition rate $R(x)$ is defined as

$$R(x) = \lim_{t\to\infty} \frac{n_x(t)}{t}, \tag{7}$$

where $n_x(t)$ is the number of transitions of $x(t)$ between a time interval of length t. The total power dissipation PW, including short-circuit power dissipation, is represented as follows.

$$PW = \frac{1}{2} \sum_{i}^{n} PW_{table} R(i), \tag{8}$$

where n is the number of gates and PW_{table} is the energy that is consumed when the output changes. The values of PW_{table} are given by look-up tables which includes the power dissipated by the short-circuit current. 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 use Eq.(8) as the object function of power optimization.

2.2 Estimation of Generated Glitches

We briefly explain the estimation method for generated glitches based on a statistical approach[11]. The condition for glitch generation is to hold the following two conditions simultaneously.

Condition 1 The input pattern ω_k is the pattern which can cause glitches.

Condition 2 The interval time ζ between successive transitions at different inputs is larger than the delay τ.

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 ω_k occurs. The propagation probability $P_{prop}(\omega_k)$ is the probability that the input pattern ω_k satisfies Condition 2, and can be represented as follows:

$$P_{prop}(\omega_k) = \int\int_{A_k} f(\alpha) f(\beta) d\alpha d\beta, \tag{9}$$

where α and β are the arrival times of the respective signals in ω_k, f is the distribution function which represents the number of transitions as a function of arrival time, and A_k is the area which satisfies Condition 2 in the $\alpha - \beta$ space(Example, Fig. 1).

As for the distribution function f, it is known that the distribution becomes normal($N(m, \sigma)$) when the number of paths is sufficiently large[11]. It however needs much computing resource to decide m and σ using a full path search algorithm. We therefore assume that the distribution is uniform thereby avoiding a lengthy computation of f. The simplified distribution function f is represented as follows:

$$f(t) = \frac{1}{\alpha_{max} - \alpha_{min}} \cdot \{U(t - \alpha_{min}) - U(t - \alpha_{max})\}, \tag{10}$$

where α_{max} is the latest arrival time and α_{min} is the fastest arrival time. We will experimentally demonstrate that the simplified distribution function f is a reasonable approximation in Section4.

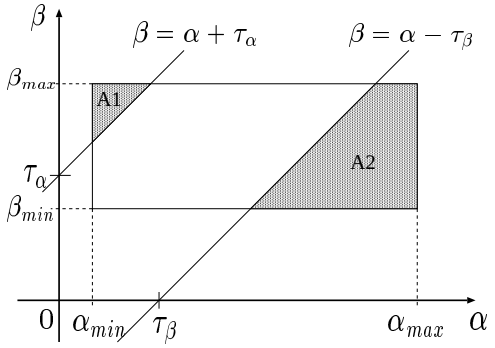


Figure 1: Surface integral area of the distribution function f.
Parameters α_{min} (β_{min}) and α_{max}(β_{max}) represent the earliest and the latest arrival times respectively.
Parameter τ_α(τ_β) represents the delay time of signal α(β).

Using P_{patt} and P_{prop}, generated glitch rate $R_{gen}(i)$ is represented as follows.

$$R_{gen}(i) = f_{clk} \cdot \sum_{k} \{P_{prop}(\omega_k) \cdot P_{patt}(\omega_k)\}. \tag{11}$$

2.3 Estimation of Propagating Glitches

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

$$R_{prop}(x) = \lim_{t\to\infty} \frac{n_{prop\_x}(t)}{t}, \tag{12}$$

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}, f_{clk} as follows:

$$R(x) = f_{clk} \cdot D(x) + 2 \cdot \{R_{gen}(x) + R_{prop}(x)\}. \tag{13}$$

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 sufficient time interval between the input transitions, the following equation holds at any gates[12].

$$R(y) = \sum_{i=1}^{n} P(\frac{\partial y}{\partial x_i}) R(x_i), \tag{14}$$

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.

$$R_{prop}(y) = \sum_{i=1}^{n} P(\frac{\partial y}{\partial x_i}) \cdot \{R_{gen}(x_i) + R_{prop}(x_i)\}. \quad (15)$$

In the case of 2-input AND gate, Eq. (15) is represented as follows.

$$\begin{aligned} R_{prop}(y) &= P(b) \cdot \{R_{gen}(a) + R_{prop}(a)\} \\ &\quad + P(a) \cdot \{R_{gen}(b) + R_{prop}(b)\}. \end{aligned} \quad (16)$$

Using Eq. (6), Eq. (16) is transformed to:

$$\begin{aligned} R_{prop}(y) = &\{P^{11}(b) + P^{10}(b)\} \cdot \{R_{gen}(a) + R_{prop}(a)\} \\ &+ \{P^{11}(a) + P^{10}(a)\} \cdot \{R_{gen}(b) + R_{prop}(b)\}. \end{aligned} \quad (17)$$

Equation (15) assumes that there is sufficient time interval between the transitions, so this equation overestimates 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.2). If the input B retains high, the glitch propagates through the gate. If the input B keeps low, the glitch never propagate through the gate. But if there is a transition at the input B, glitch propagation through the gate depends on the timing of the transitions. In order to take the lower bound of the estimation, we neglect the timing-dependent glitch propagation. Therefore the estimation of the propagating glitch rate becomes:

$$\begin{aligned} \min\{R_{prop}(y)\} = &P^{11}(b) \cdot \{R_{gen}(a) + R_{prop}(a)\} \\ &+ P^{11}(a) \cdot \{R_{gen}(b) + R_{prop}(b)\}. \end{aligned} \quad (18)$$

The above equation is obtained by setting P^{10} in Eq. (17) to be zero. Similar discussion can be made for other kinds of gates.

We can therefore calculate the lower bound of the propagating glitch rate R_{prop} from Eq. (15) as:

$$R_{prop}(y) = \sum_{i=1}^{n} \{R_{gen}(x_i) + R_{prop}(x_i)\} \cdot P(\frac{\partial y}{\partial x_i})\Big|_{P^{10}=P^{01}=0}. \quad (19)$$

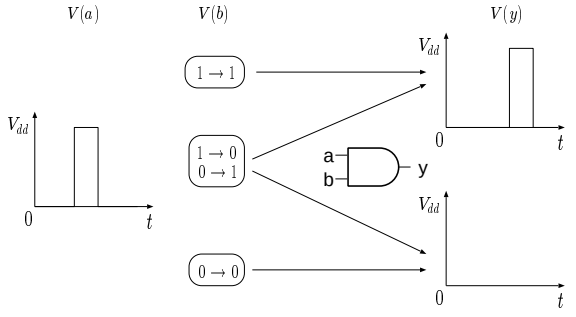


Figure 2: The condition which allows a glitch propagating through a 2-input AND gate.

3 Optimization Algorithm for Gate Sizing

Given the estimation of glitch transitions together with functional transitions, we now have a good measure of overall power dissipation. We execute discrete (cell-based) gate sizing for power optimization of a CMOS combinational circuit using the estimation method. Gate sizing does not affect functional transitions, whereas it does affect glitch transitions considerably, which will be shown later in our experiment. Resizing of a gate may generate a glitch which propagates throughout the circuit, it may be immediately blocked by the next gate. Therefore, power optimization by gate sizing is an ill-behaved problem. A simple greedy algorithm or numerical optimization may easily be trapped in a bad local optimal solution. Hill-climbing or perturbation, such as simulated annealing, is necessary to escape from a bad sub-optimal solution. Simulated annealing, however, requires numerous calculations of objective functions (power estimation in this case), and hence the computing cost is prohibitively high. We therefore develop the following heuristic algorithm which has both the merit of rapid convergence and the ability to get out of a bad local solution.

Process 1: At each gate, evaluate the sensitivity of the objective function by resizing the gate.

Process 2: Select gates according to the sensitivity and resize them. The number of resized gate is at most Max_Change. If there are no gates which has the sensitivity of reducing the object function, the optimization procedure finishes.

Process 3: If the iteration count goes over a pre-defined value $Max\_Iteration$, the optimization procedure finishes. Reduce Max_Change by a factor of $Reduce\_Rate$ and go back to Process 1.

In the case of power optimization, the object function is Eq. (8). As Eq. (8) includes short-circuit power dissipation, the power optimization considering overall power dissipation can be executed. We evaluate the sensitivity of the objective function both for sizing-up and sizing-down operations (Process 1). Here we define that the positive direction in sensitivity implies the direction of improvement in the objective function. We resize gates in Process 2. If the number of gates with positive sensitivity exceeds Max_Change, Max_Change gates are resized according to the sensitivity from the highest. If the number of gates with positive sensitivity is less than Max_Change, all positive gates are resized. Since we resize at most Max_Change gates at a time, there is no guarantee that the overall resizing results in the improvement of the objective function. 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, we can expect to reach to a good solution quickly. Tuning the parameters, $Max\_Iteration$, Max_Change, and $Reduce\_Rate$ can adjust

the amount of perturbations and convergence speed. Consequently we can control the computation time and quality of the solution.

In the case of power optimization under delay constraints, we first optimize the circuit for satisfying the delay constraints. The delay optimization is executed by our algorithm using the delay time as the object function. Then the circuit is optimized for reducing power dissipation. The procedure of optimization under delay constraints is the same with the procedure without delay constraints except for the following two points. In Process 1, we calculate the sensitivity only when the sizing does not violate delay constraints. In Process 2, we resize at most Max_Change gates at once, so there is a possibility that timing violation occurs. If timing violation occurs, the circuit is delay-optimized until the delay constraint is satisfied in Process 2.

4 Experimental Results

In this section, we first verify the accuracy of our glitch estimation method experimentally. Next we evaluate the effectiveness of our optimization algorithm compared with other algorithms. Finally we show the experimental result of power optimization and demonstrate that reducing glitch by gate sizing is an effective approach for power reduction. 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. The target library is a 0.5 μm standard cell library which is used for actual fabrication. The library includes basic and complex gates. Buffer and inverter have six varieties in the driving strength and other gates have three varieties. We adopt a wire load model that the wire load is calculated as a linear function of fanout. Coefficients of the linear function are statistically extracted from a detailed routed layout of C7552 circuit. The transition density D and signal probability P at each gate are calculated by logic simulation. Input patterns are randomly generated with a signal probability of 0.5. The number of applied patterns is 1000, which is the adequate number for the power estimation at circuit level[2]. The cycle time of the input patterns is 100ns, which is a sufficient time for all benchmark circuits to finish the behavior. The constants $Max\_Iteration$, $Reduce\_Rate$ and initial Max_Change are set to 50, 0.90, 0.4×(number of gates), respectively.

First we examine the validity of the simplified uniform distribution function f which is used for generated glitch estimation. In order to access the effect of the distribution function, we investigate the normalized surface integral area of f for P_{prop} computation. In the case of Fig. 1, the area is calculated as $\frac{\mathrm{A1+A2}}{(\alpha_{\max}-\alpha\min)\cdot(\beta_{\max}-\beta_{\min})}$. We compute all the normalized surface integral areas for every input pattern ω_k at all gates in C3540 circuit. The distribution of the normalized surface integral area is shown in Fig. 3. We can see that the most of the area is either close to zero or one. The area close to zero means that the fastest and the latest arrival times are so close that there is little possibility of glitch generation. The area close to one means that the fastest and the latest arrival times are so apart that glitch generation is most likely to occur. The area below 0.05 and above 0.95 occupy 71 % of the total cases. Other circuits also have the same feature and the average percentage of this area is 65.7% over 12 benchmark circuits. In these cases, the shape of the distribution function f has little effect on P_{prop}. Therefore we can observe that the assumption of the distribution function being uniform is a reasonable approximation.

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 [11] except for the simplified calculation of f function).

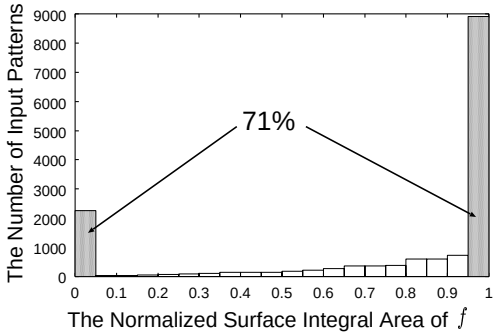
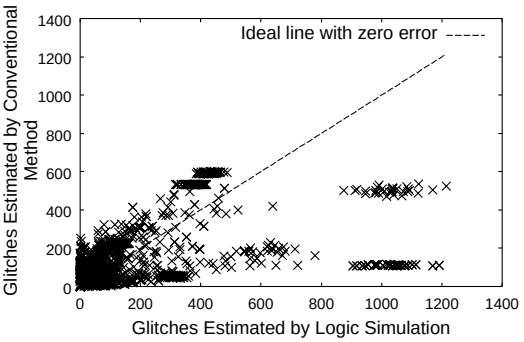


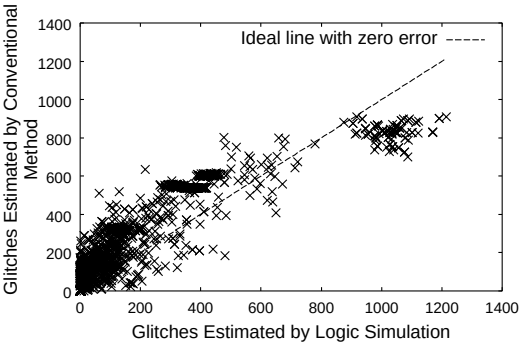
Figure 3: The distribution of the normalized surface integral area of f (C3540).

Proposed Method Both generated and propagating glitches are estimated.

Fig. 4 shows the accuracy comparison of glitch estimation 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 0.90, whereas the coefficient of the conventional method is 0.48 in des circuit. The average correlation coefficients of the proposed method over 12 benchmark circuits are 0.81 and the coefficients of the conventional method is 0.59.



Conventional Method(correlation coefficient 0.48)



Proposed Method(correlation coefficient 0.90)

Figure 4: Accuracy comparison of glitches between conventional and proposed method (des).

Next we examine the effectiveness of our optimization algorithm. We compare the proposed optimization algorithm with a simple greedy algorithm and the simulated annealing method. The simple greedy algorithm calculates the sensitivity for all gates and resize a single gate with the largest sensitivity. After resizing the gate, the sensitivity of each gate is recomputed. If there are no gates which reduces the object function, the optimization loop finishes. The simple greedy algorithm is the same with the proposed algorithm in the case that Max_Change, $Reduce\_Rate$ and $Max\_Iteration$ are set to 1, 1.0 and ∞ respectively. The simulated annealing method is implemented as follows. A reconfiguration(move) is to select a gate randomly and resizing the gate to a size which is randomly decided. As for annealing schedule, we hold temperature T constant during $100\times$(number of gates) reconfigurations or $10\times$(number of gates) successful reconfigurations. The temperature is decreased by the factor of 0.90. Table 1 shows the comparison of the optimization algorithms. The experiment is carried out using Eq.(8) as the object function. The column ''Reduction'' represents the percentage of the power reduction from the minimum-sized circuits estimated by our glitch estimation method and Eq.(8). The column ''Time'' indicates CPU times for the optimization on a SUN Ultra2. In ex5p circuit, the greedy algorithm is trapped into a bad local solution and hence the reduction remains 18.7%, whereas the simulated annealing and proposed methods achieve more than 30 % reduction. The proposed algorithm reduces the power dissipation by 13.0% on average, whereas the greedy algorithm reduces by 11.1%. Also the CPU time spent for the proposed method is 55% of that for the greedy algorithm on average. Compared with the simulated annealing, the proposed algorithm can find a solution close to that of the simulated annealing, while spending only 0.4% of the CPU time on average.

Table 1: Comparison of Optimization Algorithms.

Circuit	Greedy		S.A.		Proposed		NO. of Gates
	Reduction(%)	Time (s)	Reduction(%)	Time (s)	Reduction(%)	Time (s)	
C3540	10.0	97	10.2	8956	10.0	60	525
C5315	15.2	279	15.9	24148	15.7	111	721
apex3	12.5	156	15.2	47139	14.2	105	750
pair	4.1	94	4.1	16789	4.1	75	841
alu4	14.4	424	15.0	42168	14.4	206	889
ex5p	18.7	544	32.3	65792	31.2	175	986
C7552	6.1	216	7.5	41258	7.5	163	1098
i10	6.7	551	8.2	88864	7.7	212	1165
misex3	15.1	335	17.0	34075	16.2	158	1165
apex2	13.3	550	15.4	77869	14.7	254	1254
seq	14.1	697	16.8	85496	16.1	325	1374
des	3.2	810	4.7	98524	4.0	662	1608
average	11.1	-	13.5	-	13.0	-	-

Next we will examine to what extent we can reduce glitches by gate sizing. In this case, the objective function is the sum of transition rate as $Object = \sum_i^n R(i)$, where n is the number of gates. Initial circuits consist of the minimum-sized gates. Before and after minimizing the number of transitions, we examine the number of transitions by logic simulation. Figure 5 shows the percentage of the number of transitions compared to the total transitions of the minimum-sized circuits. The white bars correspond to functional transitions. The sum of gray and black bars represents the glitch transitions in the minimum-sized circuits. The glitch transitions of the optimized circuits correspond to the gray bars. Hence the black bars represents the amount of glitch reduction by gate sizing. Glitch transitions occupy 51.3 % of total transitions maximum and 33.8% on average. Timing optimization by gate sizing can reduce glitch transitions by 63.4 % maximum and 38.2 % on average. This reduction directly leads to the decrease in the total transition counts. The average reduction of the total transitions by gate sizing is 12.8 %.

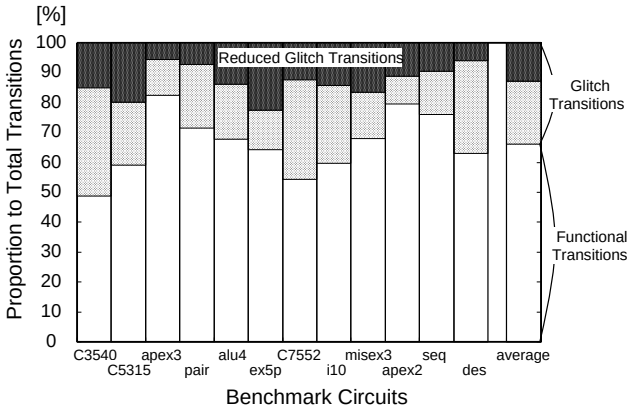


Figure 5: Toggle Reduction.

Now we will show the result of power optimization. The objective function is Eq. (8) which represents dynamic power dissipation including short-circuit power dissipation. We can optimize circuits considering overall power dissipation. First, we execute power optimization without any delay constraints. Initial circuits consist of the minimum-sized gates. The result is summarized in Table 2. Power dissipation is calculated based on the transition counts obtained by logic simulation. Column ''Area Increase'' indicates the increase of total amount of cell area. ''Toggle Reduction'' shows the reduction of transitions which consists of both functional and glitch transitions. Value ''Power Reduction'' is different from that of Table 1 because the value in Table 1 is based on our estimation of glitch activities whereas the value in Table 2 is obtained by logic simulation. From the table, we can see that the proposed method reduces the total number of transitions by 10.7 % with the expense of 6.8 % area increase on average, which results in the power reduction of 7.4 % on average and 15.7 % maximum further from the minimum-sized circuits. It is notable that the delay is also reduced in all circuits, although we do not include delay in the objective function nor constraints. The percentage of delay reduction is 21.5% on average. Glitch reduction has an aspect of path balancing. In this experiment, we use the minimum-sized circuits as starting points, and hence the path balancing is enforced by reducing longer path delays, which leads to the reduction of the critical path delay.

Finally 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. The circuit is optimized in the following three methods.

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

Conventional Power Optimization 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.

After the optimization, power dissipation is evaluated by logic simulation. The power-delay trade-off curve of each method is shown in Fig. 6. The minimum-sized circuit is located near the

Table 2: Power reduction with no delay constraints.

Circuit	Power Reduction (%)	Delay Reduction (%)	Area Increase (%)	Toggle Reduction (%)
C3540	10.1	12.5	5.1	11.3
C5315	13.5	11.8	7.0	17.1
apex3	0.0	32.6	8.2	6.2
pair	2.8	22.8	3.9	3.1
alu4	8.3	34.6	3.9	11.1
ex5p	15.7	23.0	26.0	25.7
C7552	8.9	19.6	3.2	9.9
i10	8.8	17.0	5.4	10.7
misex3	10.3	14.5	5.2	14.5
apex2	4.8	24.1	5.2	8.8
seq	4.9	34.8	4.9	8.2
des	0.8	10.5	3.4	2.3
average	7.4	21.5	6.8	10.7

top right corner of the figure. Achievable delay times by the three methods are the same. The fastest circuits by the three methods have 8.9 nsec 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 minimum-sized circuit, while the power dissipation is almost the same as that of the minimum-sized circuit. Corresponding increase in capacitive load is compensated by the reduction of glitch activity which is a by-product of the delay optimization. Explicitly exploiting the possibility of glitch reduction, the proposed method further reduces the power dissipation by more than 11% except for the case with the tightest constraint(8.9ns). We can see that the gate sizing considering glitch reduction is an effective method for power reduction.

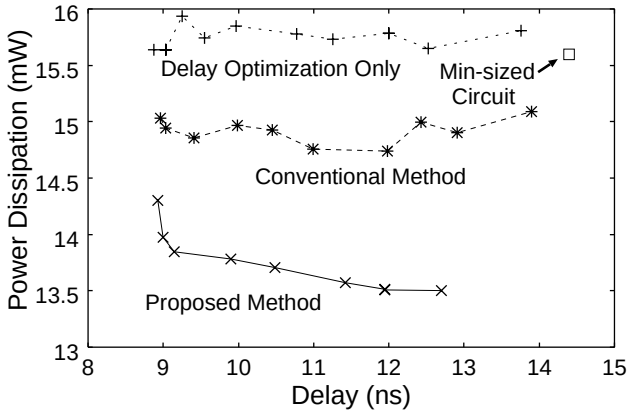


Figure 6: Power-Delay Trade-Off Curve (C5315).

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. The effect of our method is experimentally verified using 12 benchmark circuits with a 0.5 μm standard cell library. By gate sizing, glitch transitions are reduced by 38.2 % on average, which results in the reduction of total transitions by 12.8 %. The power dissipation is reduced by 7.4 % on average and by 15.7 % maximum from the minimum-sized circuits.

References

[1] Amelia Shen, Abhijit Ghosh, Srinivas Devadas and Kurt Keutzer,'' On Average Power Dissipation and Random Pattern Testability of CMOS Combinational Logic Networks,'' In Proceedings of the 1992 IEEE/ACM International Conference on Computer-Aided Design, pp.402-407, 1992.

[2] Daniel Brand and Chandu Visweswariah,'' Inaccuracies in Power Estimation during Logic Synthesis,'' In Proceedings of the 1996 IEEE/ACM International Conference on Computer-Aided Design, pp.388-394, 1996.

[3] Farid N. Najm and Michael Y. Zhang,'' Extreme Delay Sensitivity and the Worst-Case Switching Activity in VLSI Circuits,'' In Proceedings of the 32nd IEEE/ACM Design Automation Conference, pp.623-627, 1995.

[4] J. P. Fishburn and A. E. Dunlop,'' TILOS: A Posynomial Programming Approach to Transistor Sizing,'' In Proceedings of the 1985 IEEE/ACM International Conference on Computer-Aided Design, pp.326-328, 1985.

[5] M. R. C. M. Berkelaar and J. A. G. Jess,'' Gate Sizing in MOS Digital Circuits with Linear Programming,'' In Proceedings of European Design Automation Conference, pp.217-221, 1990.

[6] Guangqiu Chen, Hidetoshi Onodera and Keikichi Tamaru,'' An Iterative Gate Sizing Approach with Accurate Delay Evaluation,'' In Proceedings of the 1995 IEEE/ACM International Conference on Computer-Aided Design, pp.422-427, 1995.

[7] Yutaka Tamiya and Yusuke Matsunaga,'' LP based Cell Selection with Constraints of Timing, Area, and Power Consumption,'' In Proceedings of the 1994 IEEE/ACM International Conference on Computer-Aided Design, pp.378-381, 1994.

[8] How-Rern Lin and TingTing Hwang,'' Power Reduction by Gate Sizing with Path-Oriented Slack Calculation,'' In Proceedings of the 1st Asia-Pacific Design Automation Conference, pp.7-12, 1995.

[9] Manjit Borah, Robert Michael Owens and Mary Jane Irwin,'' Transistor Sizing for Minimizing Power Consumption of CMOS Circuits under Delay Constraint,'' In Proceedings of the 1995 International Symposium on Low Power Design, pp.167-172, 1995.

[10] Sachin S. Sapatnekar and Weitong Chuang,'' Power vs. Delay in Gate Sizing: Conflicting Objectives?,'' In Proceedings of the 1995 IEEE/ACM International Conference on Computer-Aided Design, pp.463-466, 1995.

[11] Yong Je Lim and Mani Soma,'' Statistical Estimation of Delay-Dependent Switching Activities in Embedded CMOS Combinational Circuits,'' IEEE Transaction on Very Large Scale Integration Systems, Vol. 5, No. 3, pp.309-319, September 1997.

[12] F. N. Najm,'' Transition Density, a Stochastic Measure of Activity in Digital Circuits,'' In Proceedings of the 28th IEEE/ACM Design Automation Conference, pp.644-649, 1991.