Retiming-Based Logic Synthesis for Low-Power

Yu-Lung Hsu
Institute of Computer Science
National Chung-Hsing University
Taichung 402, Taiwan, ROC
+886-4-2840497
paul@cs.nchu.edu.tw

Sying-Jyan Wang
Institute of Computer Science
National Chung-Hsing University
Taichung 402, Taiwan, ROC
+886-4-2840497
sjwang@cs.nchu.edu.tw

ABSTRACT
Power management has become a great concern in VLSI design in recent years. In this paper, we consider the logic level design technique for low power applications. We present a retiming-based optimization method, in which part of the circuit is selected and moved so that it produces logic signals one clock cycle before they are actually applied. If these values can solely determine the output logic level, then the other part of the circuit can be turned-off to save power. We explore acceptable retimed circuit structures, in which circuit function is not changed. An algorithm is proposed to select the optimal logic block to be retimed. We experiment the low-power circuit structure with some MCNC benchmark circuits, and results indicate an improvement over previous methods. Our method achieves a significant reduction in switching activity, and the reduction can be more than 70% in some case. The required area overhead is very small.

Categories and Subject Descriptors
B.6.3 [**Logic Design**]: Design Aids – *automatic synthesis, optimization, switching theory.*

General Terms
Algorithms, Design.

Keywords
Low-power, retiming, logic design, switching actvity.

1. INTRODUCTION
Low power design has attracted tremendous attention in recent years. The advance in technology makes it possible to put more and more devices in the same silicon area while at the same time pushes the clock rate even higher. Low power design is thus necessary to reduce the packaging and cooling costs as well as prolong the life span of integrated circuits (ICs). The second source of requirement for low power design comes from mobile applications, in which the lifetime of a battery can be extended if the power is reduced.

ISLPED'02, August 12-14, 2002, Monterey, California, USA.

The technology-independent low-power design strategy reduces power consumption through a refined design process. An obvious method to reduce the power consumption is to shut down part of a circuit when it is not in operation condition. Since the dynamic power dissipation in a VLSI is usually introduced by signal transitions in the circuit, many studies have also been carried out to minimize the average power dissipation by reducing switching activities [1] of a given logic circuit. The minimization can be achieved at technology mapping phase [2] and logic design phase [3]-[6]. The major logic design methods for low power include techniques to eliminate glitches [3],[4], to reduce switching activity in normal computation [5],[6].

In this paper, we present a retiming-based technique for low power design. This method tries to move a part of the circuit to an earlier stage, so that signal transitions in some nets can be halted while the circuit function is still correct. We experiment the proposed method, and the results show that our method can efficiently reduce dynamic power consumption in a circuit.

2. PRELIMINARIES
CMOS is currently the dominant technology in digital VLSI. Two components contribute to the power dissipation in CMOS circuits. The static dissipation is due to leakage current, while dynamic power dissipation is due to switching transient current as well as charging and discharging of load capacitances.

Since the amount of leakage current is usually small, the major source of power dissipation in CMOS circuits is the dynamic power dissipation. Dynamic power dissipation appears only when a CMOS gate switches from one stable state to another. In the dynamic power dissipation, the component due to the charging and discharging of load capacitance is usually the dominant factor. Thus the average dynamic dissipation of a CMOS gate is:

$$P_{avg} = \frac{1}{2} \times C_L \times V_{DD}^2 \times f_p \times N \qquad (1)$$

where C_L is the load capacitance, V_{DD} is the power supply voltage, f_p is the clock frequency, and N is the average number of gate output transition, or switching activity, in a clock cycle [3]. Thus, the power consumption can be reduced if one can reduce the switching activity of a given logic circuit without changing its function.

The focus of low-power logic design is to eliminate unnecessary switching activities in a logic circuit. One source of such activity comes from glitches, which is due to delay in components. Under zero-delay, a logic gate should make at most one signal transition in each clock cycle. However, the delay in signal propagation may

create multiple output transitions in a cycle, and this is known as glitches. Glitches can be reduced by retiming or gate freezing [3],[4]. On the other hand, the major source of switching activity comes from normal circuit function; so much better results can be achieved by trying to suppress signal transitions during normal operation [5]. One approach is to synthesize a precomputation logic [5], and normal circuit operation is suspended whenever output function can be solely decided by the precomputation logic. Another frequently applied technique is the asynchronous design (e.g. [6]), in which signal makes transition only when necessary.

3. LOW-POWER CIRCUIT RETIMING

3.1 Basic Idea

One obvious way to reduce dynamic power consumption during normal circuit operation is to eliminate signal transitions that are not necessary to get the correct output signal. This can be done through the notation of *Observability Don't Care* (ODC) [7] sets. The basic idea is illustrated in Figure 1.

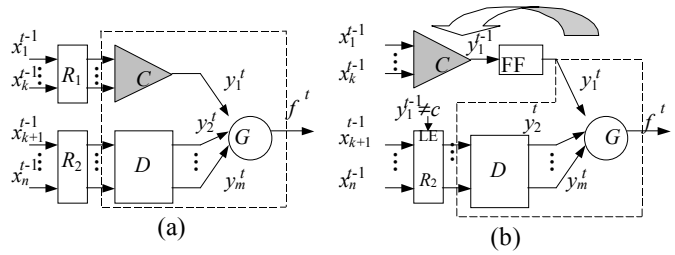

Figure 1. (a) Original Circuit, (b) retimed circuit.

Consider a single output function $f(x_1,\ldots,x_n)$ whose Boolean network is shown in Figure 1(a). Let the node that produces the output function be G, and assume node G has m inputs $y_1,\ldots, y_m$. Let that the *controlling value* of node G be c, which means the output is determined whenever one of the inputs has the value c on it. For instance, the controlling value of AND and NAND gates is 0, while for OR and NOR gates the controlling value is 1. When a node has a controlling value on any one of its inputs, the logic signals on the other inputs are not observable and thus can be set to any value. Since our goal is to achieve lower power consumption, a feasible way is to refrain those unobservable signals from changing so that switching activity can be reduced.

Since our method involves the sequential activity in a circuit, we shall use the following notation to describe timing behavior on a given net.

Definition 1: The logic value on net y at time t is denoted as y^t.

In Figure 1, if $y_1^t=c$ (c is the controlling value of node G), the signals on y_2^t to y_m^t are irrelevant and thus all the switching activities in logic block D can be stopped to reduce power in cycle t. An easy way to accomplish this goal is to disable register R_2 from loading new values at cycle t. Since there are no signal transitions in R_2, all signals in logic block D will not change. A retimed circuit that achieves this goal is shown in Figure 1(b). We move cone C such that it produces signal y_1 a cycle earlier, and the result is stored in a flip-flop (FF), which will be applied to node G at the next cycle. Assume $y_1^{t-1}= c$, which is the controlling value of node G. In this case, f^t will be solely decided by y_1^t, which implies that we do not have to load register R_2 in cycle t. Hence, in Figure 1(b), register R_2 will be load enabled (LE) only if $y_1^{t-1}\neq c$.

Note that, in this method, we need to introduce one extra flip-flop to store the output of retimed cone C. On the other hand, if the inputs of C are not used anywhere else, the corresponding registers can be removed. For example, register R_1 in Figure 1(a) is removed in Figure 1(b) since $x_1,\ldots,x_k$ are used only by logic cone C.

The same net may be able to disable two different logic blocks since it can be assigned with two different logic values. For example, consider the circuit shown in Figure 2. If the logic value on net y is 0, logic block D_0 can be turned-off to save power, while logic block D_1 can be halted when 1 is assigned to net y.

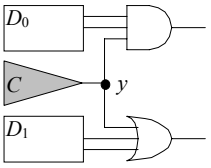

Figure 2. CP and DCNS.

In order to facilitate the discussion, we shall use the following terms in the paper. Let N be the set of nets in a circuit.

Definition 2: A *controlling point* is a 2-tuple (y, c), where $y\in N$ and $c\in\{0,1\}$. For a given controlling point p=(y, c), the *controlling cone*, or *C-cone* for simplicity, is the logic cone whose output is net y, and the set of nets in this cone is denoted as $CNS(p)$.

Definition 3: The set of nets whose switching activity can be halted by a controlling point p is referred to as the *Don't-Care Net Set* for p, and it is denoted as $DCNS(p)$ for simplicity.

For example, there are two controlling points for net y in Figure 2: p_0=(y,0) and p_1= (y,1). The C-cone for either controlling point is the logic cone C. According to the above definitions, $DCNS(p_0)$ contains the nets in logic block D_0, and $DCNS(p_1)$ is the set of nets in D_1.

3.2 Synchronization of Retimed Logic Logic

When a cone C is retimed, it may create a synchronization problem if some internal nets in C have fanout branches that are applied to nodes external to C. The scenario is shown in Figure 3(a). Assume that we will retime cone C for low power. Let the output of cone C be y_i, and assume a net y_j in cone C has fanout branches. In the retimed circuit (Figure 3(b)), both y_i and y_j are computed a cycle earlier than they are actually used. Therefore, both signals have to be buffered in registers so that timing requirement will not be violated.

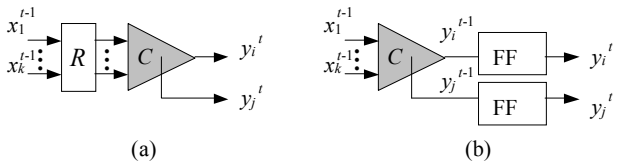

Figure 3. Signal synchronization in retimed logic: (a) original circuit, (b) retime circuit.

3.3 First Page Copyright Notice

In Figure 1, all inputs to logic block D are applied by a register that can be disabled from loading new values. However, it may not be possible to disable all of them in some case. The exceptional condition is illustrated in Figure 4. Assume the controlling value of node G is c, and thus $p=(y_i,c)$ is a controlling point. Suppose net y_j in logic block D has fanout branches connected to at least a node that is not in D. In this case, register R_2 should be loaded with new values in each cycle; otherwise, y_j may have incorrect value on it. In other words, net y_j and those nets in its input cone do not belong to $DCNS(p)$.

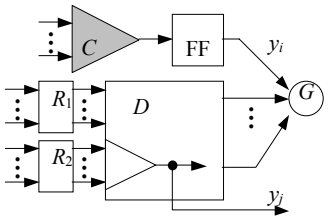

Figure 4. Selecting load-enabled registers.

4. ALGORITHM

In this section, we present our algorithm to select the best controlling cones to be retimed. In the discussion, we consider a Boolean network, in which the set of all nets in is N. Assume $|N|=n$, where $|N|$ is the cardinality of N. Let the set of all controlling points in the circuit be CP, so the size of CP is $2n$.

4.1 Estimating Reduced Switching Activities

Our goal is to find a subset $P \in CP$, so that the reduced switching activity will be maximized if the C-cones correspond to P are retimed. Therefore, the first thing we need to do is to find a way to estimate the saved switching activities due to the retiming of a given set of controlling points.

For a net y, the probability of signal transition on y in any cycle, which is denoted as $PS(y)$, is:

$$\begin{aligned} PS(y) &= \Pr\{\text{a transition on } y \text{ at time } t\} \\ &= \Pr\{y^{t-1} \oplus y^t = 1\} \\ &= \Pr\{((y^{t-1}=0) \wedge (y^t=1)) \vee ((y^{t-1}=1) \wedge (y^t=0))\} \\ &= 2 \times \Pr\{y=0\} \times \Pr\{y=1\} \end{aligned} \quad (2)$$

If $p=(y, c)$ is selected as a controlling point, the expected amount of saved switching activity in the combinational circuit, which is denoted as $ESC(p)$, can be calculated as follows:

$$ESC(p) = \Pr\{y=c\} \times \sum_{z \in DCNS(p)} PS(z) \quad (3)$$

The expected value of saved switching activity due to a set of controlling points P, $ESC(P)$, is more complex. Let $P=\{p_1, p_2\}$, and assume $CNS(p_1) \cap DCNS(p_2) \neq \varnothing$. In this case Eq. (3) is not a correct estimation for p_2, since a part of $DCNS(p_2)$ has been retimed and the switching activity in that part is not eliminated. The correct $ESC(P)$ can be computed as follows. Let $P = \{ p_1, p_2, \ldots, p_k\}$, where $p_i = (y_i, c_i)$ for $1 \le i \le k$ and $k \le n$. Thus, we have:

$$CNS(P) = \bigcup_{i=1}^{k} CNS(p_i) \quad (4)$$

Rewrite Eq. (3) as:

$$ESC(p_i) = \Pr\{y_i = c_i\} \times \sum_{z \in \{DCNS(p_i) - CNS(P)\}} PS(z) \quad (5)$$

In Eq. (5), "–" is the set difference operation. Therefore,

$$ESC(P) = \sum_{i=1}^{k} ESC(p_i) \quad (6)$$

Finally, if some registers can be removed due to the retiming process (e.g., R_1 in Figure 1(a) is removed after retiming), more switching activity can be saved. Let the number of nodes inside a flip-flop be n_f, and the number of flip-flops that can be removed be r. The probability of signal transition in a flip-flop is 0.5 assuming that the input is random. Therefore, the expected value of reduced switching activity in the whole circuit is:

$$ES(P) = \sum_{i=1}^{k} ESC(p_i) + r \times 0.5 \times n_f \quad (7)$$

We assume $n_f = 10$ in this paper. The following lemma shows how to select controlling points so that maximal reduction in average switching activity can be achieved.

Lemma 1: Let P be a set of controlling points. P achieves maximal reduction in average switching activity if and only if the condition is satisfied:

$$\forall P_i \subseteq CP \text{ and } P_i \neq P, ES(P_i) \le ES(P)$$

4.2 Selecting Controlling Points

According to Lemma 1, we may have to consider all subsets of CP to decide the best retiming scheme. Obviously the complexity is too high, as there are up to $O(2^{2n})$ sets to be considered. However, in reality, the size of the optimal set of controlling points is relatively small. We apply the following algorithm to find the circuit to be retimed.

Algorithm: Finding the optimal C-cones to be retimed.

Input: A netlist N and a given k.

Output: A subset of N to be retimed.

```
1.  MaxSaving←0, BestSolution←∅;
2.  Construct the set of all controlling points CP;
3.  for (i=1; i<= k; i++) {
4.     Find all valid subset P⊆CP, |P|=k;
5.     for (each valid P) {
6.        Find DCNS(P) and CNS(P);
7.        Check removed registers;
8.        if (ES(P) > MaxSaving) {
9.           MaxSaving← ES(P);
10.          BestSolution←P;
          }
       }
    }
11. Report CNS(P);
```

The complexity of this algorithm is $O(n^k)$. However, our experimental results, which will be presented in the next section, show that in most case the size of the optimal set is usually 1 or 2. Therefore, a small k should be sufficient for this algorithm.

5. EXPERIMENTAL RESULTS

We experiment the proposed method with 10 MCNC combinational benchmark circuits. These circuits are also used in

[5] so that a comparison is possible. We did not experiment with two-level circuits, since the structure of two-level circuits make it difficult to apply the proposed method. The statistics of the benchmarks and experimental results are given in Table I. For each circuit, the number of inputs (I), outputs (O), and literal counts (Literals) are listed. For the proposed method, the number of controlling points (CP), the number of extra flip-flops required (Extra FF), the number of load-enabled flip-flops vs. the number of inputs (#LEFF/#I), and the percentage reduction in switching activity (% Reduction (S.A.)) are given in the four corresponding columns. The results in [5] are also shown for comparison. The method presented in [5] synthesizes a precomputation logic to disable selected registers, so it requires extra combinational logic but no extra flip-flops are required.

The reduction in switching activity is obtained by logic simulation. For each circuit we generate 1000 random patterns and feed them into the circuits, and the signal transitions in all internal nodes are calculated under zero-delay model. The results are consistent if the number of applied random patterns is greater than 100. As we can see in Table I, our method gives better results in 8 out of the 10 experimented circuits, while in the rest two circuits (cmb and cordic) our method gives inferior results. It can be seen that in these two circuits the number of load-enabled flip-flops is relatively small. Actually, the performance of our method strongly depends on the circuit structure.

6. ACKNOWLEDGMENTS

We presented a retiming-based design methodology targeted for low-power application. This method is applicable to sequential circuits, in which part of the combinational logic can be moved before the registers so that some part can be disabled to save power. Experimental results show that the proposed method can efficiently reduce dynamic power consumption.

The efficiency of this method depends on the circuit structure. Since the structures of the benchmark circuits we used are fixed, it may restrict the effectiveness of the proposed method. If the circuits can be resynthesized, we shall be able to extract better controlling cones to achieve maximal dynamic power reduction.

7. REFERENCES

[1] F. Najm, "Transition density, a stochastic measure of activity in digital circuits," in *Proc. 28th Design Automation Conf.*, pp. 644-649, June 1991.

[2] C. Tsui, M. Pedram, and A. Despain, "Technology decomposition and mapping targeting low power dissipation," in *Proc. 30th Design Automation Conf.*, pp. 68-73, 1993.

[3] J. Monteiro, S. Devadas, and A. Ghosh, "Retiming sequential circuits for low power," in *Proc. Int'l Conf. Computer-Aided Design*, pp. 384-402, Nov. 1993.

[4] L. Benini, G. de Micheli, G.; E. Macii, M. Poncino, R. Scarsi, "Glitch power minimization by selective gate freezing," *IEEE Trans. VLSI Systems*, vol. 8, no.3, pp. 287-298, June 2000.

[5] M. Alidina, J. Monteiro, S. Devadas, A. Ghosh, and M. Papaefthymiou, "Precomputation-based sequential logic optimization for low power," in *IEEE Trans. on VLSI System*, vol. 2, no. 4, pp. 426-436, Dec. 1994.

[6] A. Mota, J. Monteiro, and A. Oliveira, "Power optimization of combinational modules using self-timed precomputation," in *Proc. IEEE Int'l Symp. Circuits and Systems*, vol. 2 , pp. 17 –20, 1998.

[7] M. Damiani, G. De Micheli, "Observability don't care sets and boolean relations," in *Proc. IEEE Int'l Conf. Computer-Aided Design*, pp. 502-505, 1990.

Table I: Experimental results.

CIRCUIT NAME	Circuit Statistics			Proposed Method				Precomputation[5]	
	I	O	Literals	CP	Extra FF	#LEFF/#I	%Reduction (S.A.)	Extra Literals	%Reduction (Power)
cht	47	36	167	1	0	46/47	30.3%	1	16%
cm138	6	8	35	1	1	5/6	72.4%	3	47%
cm150	21	1	61	1	0	20/21	36.7%	1	23%
cmb	1	4	62	1	1	7/16	22.6%	10	43%
cordic	23	2	194	1	1	17/23	29.3%	18	39%
majority	5	1	12	1	0	4/5	41%	1	19%
mux	21	1	54	1	0	20/21	52.2%	0	22%
pcle	19	9	71	2	1	16/19	60.4%	3	30%
pcler8	27	17	95	2	1	16/27	41.2%	3	38%
unreg	36	16	144	1	0	19/36	28.4%	2	18%