Retiming Sequential Circuits for Low Power

José Monteiro, Srinivas Devadas
Department of EECS
MIT, Cambridge, MA

Abhijit Ghosh
Mitsubishi Electric Research Laboratories
Sunnyvale, CA

Abstract

Switching activity is the primary cause of power dissipation in CMOS combinational and sequential circuits. We give a method of estimating power in pipelined sequential CMOS circuits that accurately models the correlation between the vectors applied to the combinational logic of the circuit.

We explore the implications of the observation that the switching activity at flip-flop outputs in a synchronous sequential circuit can be significantly less than the activity at the flip-flop inputs. We present a retiming method that targets the power dissipation of a sequential circuit.

1 Introduction

For many consumer electronic applications low average power dissipation is desirable and for certain special applications low power dissipation is of critical importance. For applications such as personal communication systems like hand-held mobile telephones, low-power dissipation may be the tightest constraint in the design. More generally, with the increasing scale of integration, we believe that power dissipation will assume greater importance, especially in multi-chip modules where heat dissipation is one of the biggest problems.

The average power dissipation of a circuit, like its area or speed, may be significantly improved by changing the architecture or the technology of the circuit [1]. But once these architectural or technological improvements have been made, it is the switching of the logic that will ultimately determine its power dissipation.

Methods for the power estimation of logic-level combinational [7] and sequential [3] circuits have been presented previously. In this paper, we augment the methods of [3] to obtain a more accurate estimation method that is applicable to pipelined sequential circuits. We assume that the reader is familiar with the techniques described in [3].

Traditionally, logic synthesis has been applied to improve the area or speed of a circuit. In [8], a new cost function for combinational logic synthesis targeting low power was presented. Methods that lowered power dissipation by restructuring the combinational logic of a circuit were developed. A method to speed up a sequential circuit using retiming and subsequently lowering power dissipation (and increasing delay) by scaling down the power supply voltage was presented in [2]. In this paper, we explore the application of retiming techniques to modify switching activities on internal wires of a circuit and demonstrate the impact of these techniques on average power dissipation.

2 Power Estimation

A common model for a sequential circuit is shown in Figure 1. We assume that power is dissipated only when the input vector to the circuit changes. We will denote the vector pair applied to the combinational logic as $< V0,\ Vt >$. $V0$ and Vt have a primary input part and a present-state part. $V0$ is denoted $I0@P0$ and Vt is denoted $It@Pt$, where $I0$ and It correspond to the primary input parts, and $P0$ and Pt correspond to the present-state parts.

One can ignore the feedback corresponding to the next-state lines and present-state lines and estimate the power dissipated by the combinational logic using the method of [3]. However, this strategy is a relatively crude approximation because of two reasons. Firstly, it assumes that the vector pairs applied to the combinational logic are uncorrelated. However, a vector pair $< V0,\ Vt >$ will have the property that Pt is the state

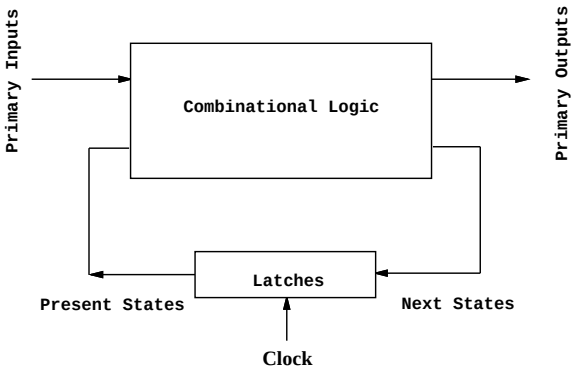

Figure 1: A General Synchronous Sequential Circuit

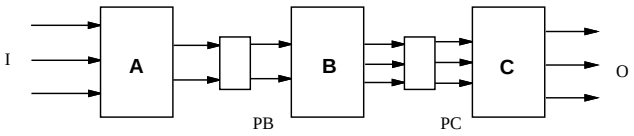

Figure 2: A k Pipeline

produced by $I0$ when the machine is in state $P0$. This correlation is ignored in combinational analysis. Further, the machine may be in different states (different $P0$'s) with different probabilities. Combinational analysis will assume a uniform probability for all the states.

2.1 Pipelines

Many sequential circuits, such as pipelines, can be acyclic. They correspond to blocks of combinational logic separated by flip-flops. An example of a 2-stage pipeline that is an acyclic sequential circuit is given in Figure 2. I corresponds to the primary inputs to the circuit, O the primary outputs, and PB and PC the present-state lines that are inputs to blocks B and C, respectively.

It is possible to estimate the power dissipated by acyclic circuits that are k-pipelines, *i.e.* those that have exactly k flip-flops on each path from primary inputs to primary outputs, without making any assumptions about the probabilities of the present-state lines. This is because such circuits are k-definite [4], *i.e.* their state and outputs are a function of primary inputs that occurred at most k clock cycles ago.

Consider the circuit of Figure 3. The symbolic simulation equations corresponding to the switching activities of logic gates in blocks A, B and C are assumed to have been computed using the method of [3]. The symbolic simulation equations for block A receive inputs from $I0_i$ and It_i, since block A receives inputs from I alone. The symbolic simulation equations for block B receive inputs from $PB0_j$ and PBt_j, and to model the relationship between PB and I, we generate $PB0_j$ from $I0_i$ and the PBt_j from It_i. Similarly, the symbolic simulation equations for block C receive inputs from the $PC0_k$ and PCt_k and to model the relationship between PC and I we generate $PC0_k$ from $I0_i$ and the PCt_k from It_i.

The decomposition of Figure 3 implies that the gate output switching activity can be determined given only the vector pair $< I0, It >$ for the primary inputs. Therefore, to compute gate output transition probabilities, we only require the transition probabilities for the primary inputs. This use of the next-state logic generates Boolean equations which model the relationship between the state of the circuit and the previously applied input vectors.

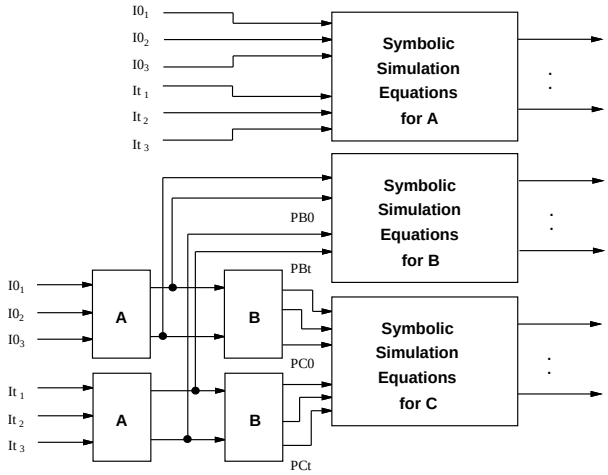

Figure 3: Taking k Levels of Correlation Into Account

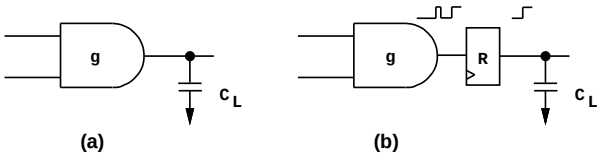

Figure 4: Adding a Flip-Flop to a Circuit

3 Positioning of Flip-Flops

We begin with some interesting observations that relate the positioning of flip-flops in a sequential circuit to the power dissipation of the circuit.

Consider the circuit of Figure 4(a). If the average switching activity (during a clock cycle) at the output of gate g is E_g and the load capacitance is C_L, then the power dissipated by the circuit is proportional to $E_g \cdot C_L$. Now consider the situation when a flip-flop R is added to the output of g, as illustrated in Figure 4(b). The power dissipated by the circuit is now proportional to $E_g \cdot C_R + E_R \cdot C_L$, where E_g is as before, C_R is the capacitance at the input of the flip-flop, and E_R is the average switching activity at the flip-flop output. The main observation here is that $E_R < E_g$, since the flip-flop output will make at most one transition at the beginning of the cycle. For example, the gate g may glitch and make three transitions as shown in the figure, but the flip-flop output will make at most one transition when the clock is asserted. This implies that is possible that $E_g \cdot C_R + E_R \cdot C_L$ is less than $E_g \cdot C_L$ if both E_g and C_L are high. Thus, the addition of flip-flops to a circuit may actually decrease power dissipation. Since adding flip-flops to a circuit is a common way to improve the performance of a circuit by pipelining it, it is worthwhile investigating the ramifications of this observation.

Next, consider the more complex scenario of altering the position of a flip-flop in a sequential circuit. Consider the circuit of Figure 5(a). The power dissipated by this circuit is proportional to $E_0 \cdot C_R + E_1 \cdot C_{L1} + E_2 \cdot C_{L2}$. Similarly, the power dissipated by the circuit of Figure 5(b) is proportional to $E_0 \cdot C_{L1} + E'_1 \cdot C_R + E'_2 \cdot C_{L2}$. Again, one circuit may have a lesser power dissipation than the other. Due to glitching, E'_1 may be greater than E_1 but by the same token E'_2 may be less than E_2. The capacitances of the logic blocks and the flip-flops along with the switching activities will determine which of the circuits is more desirable from a power standpoint. The circuits may also have differing performance.

We utilize the above observations in a heuristic retiming strategy that targets power dissipation as its primary cost function.

4 Retiming for Low Power

Retiming algorithms that minimize clock periods [5, 6] rely on the fact that delay varies linearly under retiming. Unfortunately that is not so with switching activity. The retiming of a single node can dramatically change the switching activity in a circuit and it is very difficult to predict what this change will be. Further, estimating the switching activity is itself a computationally expensive task.

The algorithm we propose for reducing power dissipation in a pipelined circuit heuristically selects the set of nodes which, by having a flip-flop placed at their outputs, lead to the minimization of switching activity in the network. Nodes are selected based on the amount of glitching that is present at their outputs and on the probability that this glitching propagates through their transitive fanouts.

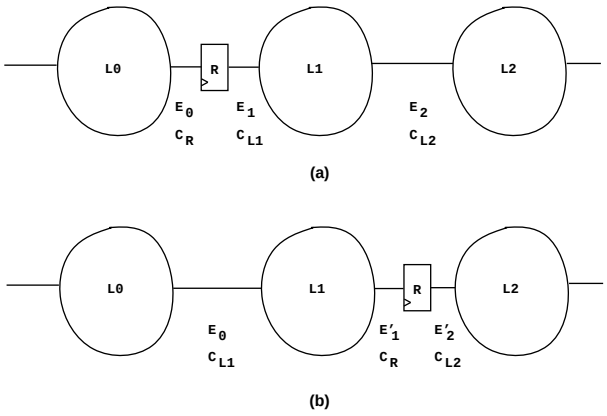

Figure 5: Moving a Flip-Flop in a Circuit

4.1 Cost Function

We start by estimating the average switching activity of the combinational network (ignoring the flip-flops), both with zero delay (E_{zeroD}) and actual delay (E_{genD}) for each gate, thus obtaining the amount of glitching (E_{glitch}) at each gate by taking the difference of the expected number of transitions in these two cases ($E_{glitch} = E_{genD} - E_{zeroD}$).

We then evaluate the probability that a transition at each gate propagates through its transitive fanout. For each gate j in the transitive fanout of node i we calculate the probability of having a transition at node j caused by a transition at gate i (*sensitivity* of gate j relative to gate i, $s_{j,i}$):

$$s_{j,i} = \frac{P(i\updownarrow \wedge j\updownarrow)}{P(i\updownarrow)}$$

where $P(i\updownarrow)$ is the probability of a transition at node i, calculated using the methods of [3].

The value of $P(i\updownarrow \wedge j\updownarrow)$ can be calculated by first calculating the primary input conditions under which a transition at i triggers a transition at j. This can be calculated using the zero delay power estimation methods in [3].

Since the objective is to reduce power, we weight these sensitivities with the capacitive load of the corresponding node. So the measure of the amount of power dissipation that is reduced by placing a flip-flop at the output of a node i is:

$$power\_red(i) = E_{glitch}(i) \times (C_i + \sum_{j}^{fanout_i} (s_{j,i} \times C_j))$$

The transitive fanout of a node might contain a very large number of nodes, so we restrict the number of levels of transitive fanout that are taken into account. This not only reduces computation time, but also can increase the accuracy since glitching can be filtered out by the inertial delay of combinational logic.

One other factor that can significantly contribute to power dissipation is the number of flip-flops in the network. We try to minimize this number by giving higher weights to nodes with larger number of inputs ($n_i(i)$) and outputs ($n_o(i)$). A flip-flop placed at one of these nodes will be in a greater number of paths, reducing the total number of flip-flops needed. Therefore, our final cost function is:

$$weight(i) = power\_red(i) \times (n_i(i) + n_o(i))$$

4.2 Verifying a Given Clock Period

Although we aim at the circuit that dissipates the least possible power, we might also want to set a constraint on performance by specifying the desired clock cycle of the retimed circuit.

In the retiming algorithm we will be selecting the nodes that should have a flip-flop placed at the output. We restrict this selection to the nodes that still allow the retimed circuit to be clocked with the given clock period. Since the algorithm works with pipelines, this is accomplished simply by discarding nodes that have a path longer than the desired clock period, both from any primary input or to any primary output.

4.3 Retiming Constraints

The objective is to select the nodes (from those not excluded in the previous phase) with the highest weights. The constaint for node selection is that the number of nodes that share any input-output path should not surpass a given value (which is the number of flip-flop stages in the pipeline). The set having the highest sum of weights over the nodes belonging to the set is chosen.

We restrict our algorithm to place one stage of flip-flops at a time. The reason for this is that, if we allowed two stages, the algorithm could select a node i and one of its immediate fanout nodes j for a set. Choosing i will eliminate most of the glitching present at j, possibly changing significantly the weight of j, and this new weight of j difficult to predict. Thus, for pipelines with more than one stage, we apply our algorithm iteratively.

So the goal is to find the set of nodes with no more than one node per input-output path and with the highest sum of weights. Our algorithm uses a binary tree search over all the nodes, keeping record of the best set so far. For large networks, we limit the search to the most promising nodes.

First we check for pairwise compatibility, *i.e.* for each pair of nodes we check if there is one input-output path to which they both belong. This greatly simplifies the test at each level of the binary tree as we just verify if the node corresponding to this level is *incompatible* with any other node previously selected.

4.4 Executing the Retiming

Initially we position the flip-flops at the primary inputs of the network. To place a flip-flop at the output of a node in the selected set, we recursively perform backward retiming on the node, *i.e.* we add a flip-flop at its output and remove a flip-flop at each input. This operation is repeated with nodes that have *negative* flip-flops at their output due to previous retimings. Eventually we reach the primary inputs where flip-flops are present, thereby ending the recursion.

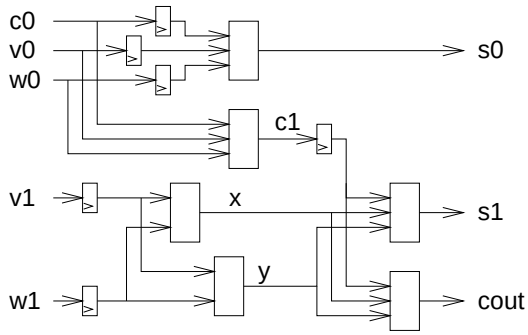

Figure 6: Circuit with the Nodes in the Selected Set Retimed.

Once we have placed flip-flops at the output of all the nodes in the set, there are typically some flip-flops that can still be moved without disturbing the flip-flops already placed. These are flip-flops on paths that do not contain any node in the selected set. For instance, consider the circuit in Figure 6 which has been through the first phase of retiming, where the only node in the selected set was node `c1`.

The first observation is that although node `c1` was retimed (and has a flip-flop at its output as was the objective), `s0` was not. Thus the flip-flops at the inputs `c0`, `v0` and `w0` were not removed. In this case it is obvious that it is preferable to retime node `s0` so that we reduce the number of flip-flops in the circuit (one at the output of `s0` instead of three at the inputs).

The second observation is that the flip-flops at inputs `v1` and `w1` were also not touched. Nodes `x` and `y` can be retimed and this would reduce the levels of combinational logic in the circuit from two to one. Note that retiming `x` and `y` will make `s1` and `cout` retimable, but we do not allow it since that would remove the flip-flop from the output of `c1`.

Thus, in the last phase of the algorithm we go through the network, from primary inputs to primary outputs, performing a backward retiming on retimable nodes so that:

- The maximum delay is lower than the desired clock period.
- The number of flip-flops is reduced.
- This retiming operation does not disturb the flip-flops placed at the output of the nodes in the selected subset.

5 Experimental Results

In Table 1, power estimation results for several pipelined sequential circuits are summarized. For each

circuit, the number of stages in the pipeline, the number of flip-flops (ff), estimated power using the method of [3] (METHOD-I) and the estimated power using the method pictorially described in Figure 3 (METHOD-II) are given. The CPU times for power estimation using the two different techniques on a DEC 3000 Model 500 AXP Workstation are also given. The different pipelined implementations with a varying number of stages for each circuit were obtained by adding flip-flops to the inputs to the circuit and retiming the circuit for minimum delay. A uniform frequency of 20 MHz was assumed for computing the power dissipation for all the circuits.

Given a delay model, the method of Figure 3 exactly computes the average switching activity for a pipelined circuit taking into account the correlation between the flip-flops. The method of [3] and other power estimation methods are restricted to assuming default values of 0.5

EX	stages/ff	METHOD-I		METHOD-II	
		Power	time	Power	time
cla_16	0/0	1100	12	1100	13
	1/48	2497	5	2389	8
	2/91	3740	4	3509	7
	3/131	4953	3	4632	6
rpl_16	0/0	1419	23	1419	26
	1/33	2302	8	2303	12
	2/65	3198	5	3172	8
	3/98	4132	4	4025	8
cbp_16	0/0	1815	111	1815	117
	1/38	2842	29	2748	45
	2/78	3931	9	3657	20
	3/115	4989	7	4569	12
cbp_32	0/0	3687	1772	3687	1890
	1/74	5751	894	5590	904
	2/152	7912	588	7459	608
	3/223	9944	446	9234	453
mult4	0/0	689	3	689	3
	1/14	1218	1	900	2
	2/27	1558	1	1152	1
	3/43	2054	1	1503	1
mult6	0/0	2658	151	2658	155
	1/29	3707	262	2803	55
	2/53	4166	7	3156	32
	3/76	4710	3	3581	155
mult8	0/0	6621	207	6621	207
	1/46	8051	110	6104	120
	2/87	8798	80	6690	84
	3/136	9743	70	7404	77

Table 1: Power Estimation for Sequential Circuits

EX	Retime-Delay			Retime-Power		
	ff	delay	power	ff	delay	power
cla_16	48	12	2389	43	12	2147
rpl_16	33	18	2303	32	32	2074
cbp_16	38	22	2748	34	42	2388
cbp_32	74	42	5590	61	71	4725
mult4	14	5	900	11	7	853
mult6	29	8	2803	22	11	2596
mult8	46	11	6104	37	15	5834

Table 2: Retiming for Low Power Without Any Timing Constraints

for the switching activities of the flip-flop outputs. They also cannot take into account the correlation between the flip-flops. This results in erroneous power values.

Next we present results obtained by using the retiming method of Section 4 that directly targets power dissipation. The delay and power dissipated by circuits retimed for minimum delay, and the delay and power dissipated by circuits retimed for minimum power *without any timing constraints* are given in Table 2. We were able to achieve significant reductions in power for some of the circuits by a judicious placement of registers using the strategies described in Section 4. However, the maximum delay of some of the retimed circuits for low power is close to the delay of the original circuit. So retiming for low power disregarding timing might give poor results is terms of performance.

In Table 3 we present the results obtained for the same circuits but now adding the constraint of minimum delay. We give results both for 1-stage pipelines and 3-stage pipelines. The latter was obtained by applying the algorithm of section 4 first to the original circuit and then to each of the two combinational parts of the retimed circuit.

We first note that the power dissipated by the pipelined circuits obtained by retiming for low power disregarding timing or by retiming for low power with a minimum delay constraint are very close. Thus it is possible to achieve a important gains in power dissipation without losing in performance. For example `rlp_16` placing a delay constraint results in a slightly better power dissipation due to the heuristic nature of the algorithms used.

Secondly observe that, even though we are using an iterative strategy for the 3-stage pipelined circuits, the gain in power is greater for these circuits. This means that even greater savings could be obtained if our algorithm is extended to build k-stages pipelines in one pass, by taking into account in the cost function of a

EX	st	Delay	Retime-Delay		Retime-Power	
			ff	power	ff	power
cla_16	1	12	48	2389	44	2181
	3	6	131	4632	126	4280
rpl_16	1	18	33	2303	31	2039
	3	9	98	4025	99	3698
cbp_16	1	22	38	2748	32	2407
	3	11	115	4569	105	4125
cbp_32	1	42	74	5590	59	4871
	3	21	223	9234	172	7725
mult4	1	5	14	900	13	860
	3	3	43	1503	38	1378
mult6	1	8	29	2803	26	2660
	3	4	76	3581	78	3563
mult8	1	11	46	6104	43	6003
	3	6	136	7404	128	6975

Table 3: Retiming for Low Power and Minimum Delay

node the reduction of glitching caused by the selection of another node that shares a common path(s).

6 Acknowledgements

This research was supported in part by the Defense Advanced Research Projects Agency under contract N00014-91-J-1698 and in part by a NSF Young Investigator Award with matching funds from Mitsubishi and IBM Corporation.

References

[1] A. Chandrakasan, T. Sheng, and R. W. Brodersen. Low Power CMOS Digital Design. In *Journal of Solid State Circuits*, pages 473–484, April 1992.

[2] P. Duncan, S. Swamy, and R. Jain. Low-Power DSP Circuit Design Using Retimed Maximally Parallel Architectures. In *Proceedings of the* 1^{st} *Symposium on Integrated Systems*, pages 266–275, March 1993.

[3] A. Ghosh, S. Devadas, K. Keutzer, and J. White. Estimation of Average Switching Activity in Combinational and Sequential Circuits. In *Proceedings of the* 29^{th} *Design Automation Conference*, pages 253–259, June 1992.

[4] Z. Kohavi. *Switching and Finite Automata Theory.* Computer Science Press, 1978.

[5] C. E. Leiserson, F. M. Rose, and J. B. Saxe. Optimizing Synchronous Circuitry by Retiming. In *Proceedings of* 3^{rd} *CalTech Conference on VLSI*, pages 23–36, March 1983.

[6] Giovanni De Micheli. Synchronous Logic Synthesis. *IEEE Transactions on Computer Aided Design*, 10(1), January 1991.

[7] F. Najm. Transition Density, A Stochastic Measure of Activity in Digital Circuits. In *Proceedings of the* 28^{th} *Design Automation Conference*, pages 644–649, June 1991.

[8] A. Shen, S. Devadas, A. Ghosh, and K. Keutzer. On Average Power Dissipation and Random Pattern Testability of Combinational Logic Circuits. In *Proceedings of the Int'l Conference on Computer-Aided Design*, pages 402–407, November 1992.