Low Power Logic Synthesis under a General Delay Model*

Unni Narayanan
Design Technology
Intel Corporation
Santa Clara, California

Peichen Pan
Design Technology
Intel Corporation
Portland, Oregon

C. L. Liu
Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan ROC

Abstract

Till now most efforts in low power logic synthesis have concentrated on minimizing the total switching activity of a circuit under a zero delay model. This simplification ignores the effects of glitch transitions which may contribute as much as 30% of the total power consumption of a circuit. Hence, low power logic synthesis techniques which optimize power under a zero delay model are often not successful in attaining "real" power savings as measured under a more accurate general delay model. In practice, to accurately estimate the switching activity in a circuit under a general delay model can be computationally expensive. Hence, to repeatedly call accurate but slow power estimation tools to direct the synthesis flow is not a viable approach in the design of low power synthesis tools. In this paper we take advantage of a fast method for estimating the total switching activity in a circuit under a general delay model to synthesize low power circuits. Specifically, we use the approximation as a basis for algorithms that solve two problems: (1) low power technology decomposition of gates under a general delay model (2) low power retiming of sequential circuits under a general delay model.

1 Introduction

The advent of portable digital devices such as laptop personal computers has made low power circuit design an increasingly important research area. For example, laptop computers have limited battery life, and so the circuitry in the computer must be designed to dissipate as little power as possible without sacrificing performance in terms of speed. Additionally, high power consumption increases the cost of handling heat dissipation and diminishes the reliability of today's increasingly complex circuits with higher transistor counts and faster clock rates. It has been shown in [2] that in CMOS technology a large portion of power dissipation on chip is due to dynamic power consumption at the gates which is computed according to the formula: $\sum_{i=1}^{N} 1/2 \cdot C_i V_{dd}^2 f_i$ where C_i is the output capacitance of the ith gate, V_{dd} is the supply voltage, f_i is the number of transitions at the output of the ith gate, and N is the total number of gates on the chip. Clearly a reduction in f_i will lead to a corresponding reduction in the total power consumption of the circuit.

Till now most efforts in low power logic synthesis have focused on minimizing f_i for each of the gates under a zero delay model [9, 10, 13]. However, in reality gates have a delay which may result in glitch transitions that increase the total power consumption in the circuit. Glitch transitions are transitions that take place at the output of a gate g prior to the output attaining its steady state value. As an example, consider the circuit in Figure 1.

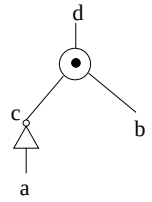

Figure 1: Glitching properties of a simple circuit

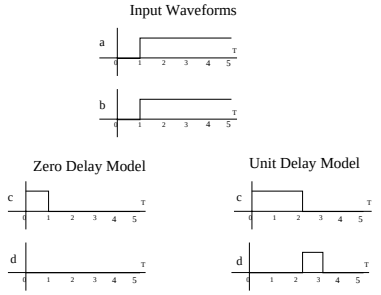

Figure 2: Glitch transitions that are not accounted for in a zero delay model.

Suppose we apply inputs to a and b as specified by the timing diagram in Figure 2. We see that under a unit delay

*This research was supported in part by the National Science Foundation (NSF) under grant MIP-96-12184 and a research grant from IBM.

model the output of the AND gate d experiences a $0 \rightarrow 1$ transition at time 2 and experiences a $1 \rightarrow 0$ transition at time 3 prior to attaining its steady state value of 0. Under a zero delay model these two transitions would not be counted. Thus if P_a, the probability a is 1, is .9 and P_b, the probability b is 1, is .1, then under the zero delay model the expected number of transitions at d is .0378, but under the unit delay model the expected number of transitions at d is .0540 or a 43% difference in expected switching activity. In an arbitrary circuit glitching power can account for as much as 30% of the total power dissipation [7]. Hence most low power logic synthesis techniques which attempt to minimize power under a zero delay model are often not successful in attaining "real" power savings as measured under a general delay model. Therefore, it is important to develop synthesis techniques which directly minimize power under a general delay model. Furthermore, in [1] the authors note that glitch power is one of the few sources of power dissipation that can be estimated accurately prior to the physical design step. Thus, savings in terms of glitch power during logic synthesis can have a significant impact on the final circuit.

In practice, to accurately estimate the switching activity in a circuit under a general delay model can be computationally expensive. Hence, to repeatedly call accurate but slow power estimation tools to direct the synthesis flow is not a viable approach in the the design of low power synthesis tools. In this paper we present a fast method for estimating the switching activity under a general delay model. While this only an estimate of the total power consumption in a circuit, we show that it is accurate enough to guide in the synthesis of low power circuits. To demonstrate the effectiveness of our estimate, we use the approximation as a basis for algorithms that solve two problems: (1) low power technology decomposition of simple gates under a general delay model (2) low power retiming of sequential circuits under a general delay model. For both problems our algorithms attain good power savings when the resulting circuits are measured under a general delay model by the power estimation tool in Berkeley SIS. In Section 2 we present our power estimation method which is a simplification of a more general scheme presented in [3]. In Section 3 we present the algorithm and experimental results for low power technology decomposition of simple gates. In Section 4 we present the algorithm and experimental results for the low power retiming of sequential circuits. Finally, in Section 5 we summarize our work and present future directions for this research.

2 Switching Activity Under a General Delay Model

First we briefly review two of the more popular power estimation methods which attempt to account for glitching transitions and we explain why these methods are not of practical value during logic synthesis. We will then present our estimation method and study some of its properties.

2.1 Review of Power Estimation Methods

Other than outright simulation (which is very costly) there are two basic approaches to gate level power estimation in practice today: (1) the transition density method (2) symbolic simulation. In the transition density method there is the concept of the transition density for a gate g denoted by $D(g)$. The quantity $D(g)$ is the expected number of transitions at the output of gate g in unit time. In a very important paper in [8], Najm showed that $D(g)$ for a gate g with n uncorrelated inputs $x_1, \ldots, x_n$ can be computed as: $D(g) = \sum_{i=1}^{n} \Pr(\partial f/\partial x_i) D(x_i)$ where f is the boolean function realized at the output of gate g with $\partial f/\partial x_i = f_{x_i} \oplus f_{\overline{x_i}}$ and where f_{x_i} and $f_{\overline{x_i}}$ are respectively the positive and negative cofactors of f with respect to x_i. The transition density approach has the nice property of efficiently and accurately propagating glitches that appear at the primary inputs of a circuit throughout the entire circuit. However, the disadvantage of the transition density approach is that it estimates power under a zero delay model. Hence, the transition density formulation accounts for glitches that are propagated from the primary inputs of the circuit, but does not account for glitches that are generated from within the circuit. In practice, this is a serious limitation.

In symbolic simulation a symbolic network is constructed from the original network and models the behavior of the original network at two time instants [7]. For each primary input i in the original network there are two primary inputs, i_{t_1} and i_{t_2}, in the symbolic network which correspond to the values of i in the original network at time instants t_1 and t_2. Each primary output in the symbolic network is denoted by $E_{g,t}$ and has the property that $E_{g,t}$ is 1 if and only if the gate g in the original network experiences a transition at time instant t. The probability that $E_{g,t} = 1$ is the probability of a transition at the output of g in the original network at time instant t. Thus, $\sum_{\forall g, \forall t} \Pr(E_{g,t} = 1)$ in the symbolic network is the total expected number of transitions in the original network. Symbolic simulation computes the expected number of transitions under the general delay model completely taking into account spatial correlation (reconvergent fanouts). The problem is that symbolic simulation requires a costly BDD evaluation for each primary output of the symbolic network because the symbolic network may contain internal correlations that were never present in the original network.

2.2 A Simplified Glitching Model

We note that a gate g in a circuit experiences a transition at its output only if one of its inputs changes. This motivates the following definition and theorem:

Definition 2.1 *The path length from a primary input i to a gate g is the number of gates on the path from i to g.*

Theorem 2.1 *Suppose a set of inputs are applied to the primary inputs of a circuit at time t. Then under the unit delay model, the set of times a gate g experiences a transition is $\mathcal{T}_g = \{t_i | t_i = t + i \ \ \forall i \in L\}$ where L is the set containing the distinct path lengths from g to its primary inputs.*

Let $P_{g,t}(0 \rightarrow 1)$,$P_{g,t}(1 \rightarrow 0)$, $P_{g,t}(0 \rightarrow 0)$, and $P_{g,t}(1 \rightarrow 1)$ denote respectively the probabilities of $0 \rightarrow 1$, $1 \rightarrow 0$, $0 \rightarrow 0$, and $1 \rightarrow 1$ transitions at the output of gate g at time t. Then we can define the total power consumption in a circuit as follows:

Definition 2.2 *Let $\mathcal{G}$ denote the set of all gates in a circuit. Let $\mathcal{T}_g$ denote the set of possible transition times of gate g as specified by Theorem 2.1. Then the total power consumption in a circuit under a unit delay model is :*

$$\sum_{\forall g \in \mathcal{G}, \forall t \in \mathcal{T}_g} P_{g,t}(0 \rightarrow 1) + P_{g,t}(1 \rightarrow 0)$$

Thus, in order to compute the power consumption in a circuit we need to compute $P_{g,t}(0 \rightarrow 1)$ and $P_{g,t}(1 \rightarrow 0)$ for

each gate g. The precise formula for computing the transition probabilities for a specific gate depends on the functionality of the gate. First we require some basic terminology. Let $x, x', y, y' \in \{0, 1\}$. Let $\mathcal{S}$ denote any subset of fanins of a k input gate. Let $P(\mathcal{S}(x))$ denote the probability that *all* the signals in $\mathcal{S}$ have a value of x. Let $P(\mathcal{S}[(x \rightarrow y), (x' \rightarrow y')])$ denote the probability that *at least* one signal in $\mathcal{S}$ makes an $x \rightarrow y$ transition and *all* the remaining signals make $x' \rightarrow y'$ transitions. Let $\mathcal{S}_t$ denote the set of signals that change at time t. Finally, let $\mathcal{S}_{\overline{t}}$ denote the set of signals that do not change at time t. Then we have the following theorem for AND, OR, and NOT gates:

Theorem 2.2 *Let g be and* AND *gate. We have:*

1. $P_{g,t}(0 \rightarrow 1) = P(\mathcal{S}_{\overline{t-1}}(1)) \cdot P(\mathcal{S}_{t-1}[(0 \rightarrow 1), (x \rightarrow 1)])$
2. $P_{g,t}(1 \rightarrow 0) = P(\mathcal{S}_{\overline{t-1}}(1)) \cdot P(\mathcal{S}_{t-1}[(1 \rightarrow 0), (1 \rightarrow x)])$
3. $P_{g,t}(1 \rightarrow 1) = P(\mathcal{S}_{\overline{t-1}}(1)) \cdot P(\mathcal{S}_{t-1}(1 \rightarrow 1))$
4. $P_{g,t}(0 \rightarrow 0) = 1 - P_{g,t}(0 \rightarrow 1) - P_{g,t}(1 \rightarrow 0) - P_{g,t}(1 \rightarrow 1)$

If g is an OR *gate then we have:*

1. $P_{g,t}(0 \rightarrow 1) = P(\mathcal{S}_{\overline{t-1}}(0)) \cdot P(\mathcal{S}_{t-1}[(0 \rightarrow 1), (0 \rightarrow x)])$
2. $P_{g,t}(1 \rightarrow 0) = P(\mathcal{S}_{\overline{t-1}}(0)) \cdot P(\mathcal{S}_{t-1}[(1 \rightarrow 0), (x \rightarrow 0)])$
3. $P_{g,t}(1 \rightarrow 1) = 1 - P_{g,t}(0 \rightarrow 1) - P_{g,t}(1 \rightarrow 0) - P_{g,t}(0 \rightarrow 0)$
4. $P_{g,t}(0 \rightarrow 0) = P(\mathcal{S}_{\overline{t-1}}(0)) \cdot P(\mathcal{S}_{t-1}(0 \rightarrow 0))$

If g is a NOT *gate then we have:*

1. $P_{g,t}(0 \rightarrow 1) = P(\mathcal{S}_{t-1}(1 \rightarrow 0))$
2. $P_{g,t}(1 \rightarrow 0) = P(\mathcal{S}_{t-1}(0 \rightarrow 1))$
3. $P_{g,t}(1 \rightarrow 1) = P(\mathcal{S}_{t-1}(0 \rightarrow 0))$
4. $P_{g,t}(0 \rightarrow 0) = P(\mathcal{S}_{t-1}(1 \rightarrow 1))$

Theorems 2.1-2.2 suggest an efficient algorithm for estimating total switching activity in a circuit: (1) In topological order for each gate apply Theorem 2.2 to compute the probabilities of $0 \rightarrow 1$, $1 \rightarrow 0$, $0 \rightarrow 0$, and $1 \rightarrow 1$ transitions for each time instant t that the output of the gate can change (the set of time instants can be computed using Theorem 2.1). (2) Compute the total switching activity as specified by Definition 2.2.

We make several observations. First, the algorithm exactly computes the expected number of transitions under a unit delay model if the circuit has no reconvergent fanouts (this is the same value that would be computed by symbolic simulation). Second, the algorithm can easily be modified to estimate transitions under a general delay model by using accurate delay information to determine the exact transition times of fanins to a gate. We note that this algorithm turns outs to be a special case of a more general approach proposed in [3]. In that work, the authors modify the Parker-McCluskey method [11] to compute switching activities under a general delay model. Their algorithm takes as a parameter the number of levels in the transitive fanin of each gate that should be considered to account for spatial correlations due to reconvergent fanouts. We observe that our approach and their approach are identical when their algorithm backtracks just one level in the transitive fanin of the gate. In [3] they report that this estimate is within 20% of the actual switching activity for benchmark circuits which have internal correlations.

3 Low Power Technology Decomposition

The technology decomposition problem is the problem of transforming a gate with many fanins into an equivalent tree representation where each gate has a small number (say two) fanins. Different decompositions vary in terms of their total switching activities. We first review previous work in the area of low power technology decomposition. After this we present our algorithm for decomposition under a general delay model. Finally we will present experimental results.

3.1 Previous Results in Low Power Decomposition

Most work in the area of low power technology decomposition has assumed a zero delay model. The most important results are found in [9, 15] where the authors show that for static CMOS logic, the Huffman algorithm synthesizes the zero delay power optimal decomposition when the on probabilities of the fanins are all less than or equal to 0.5. Similarly, the authors show that the Anti-Huffman algorithm synthesizes the zero delay power optimal decomposition when the product of the on probabilities are greater than 0.5. However, in reality the best decomposition depends on the structure of the transitive fanins to the gate as well as the on probabilities of the fanins. Thus, the optimal zero delay result does not hold under a general delay model. This is illustrated in Figure 3.

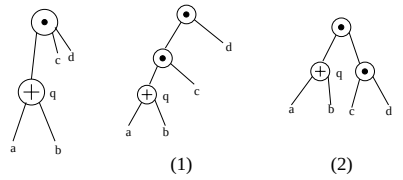

Figure 3: Problems with optimal zero delay decomposition

Suppose we wish to decompose the AND gate in Figure 3 and the on probabilities to the primary inputs are $P_a = .28$, $P_b = .28$, $P_c = .50$, and $P_d = .50$. We observe that the on probability at the output q of the OR gate is .48. Hence, the optimal zero delay algorithm will decompose the AND gate as in (1). But, under a unit delay model (with a 20 MHZ clock and Vdd = 5V) we find that the AND tree in (1) dissipates 4.64 microwatts and the AND tree in (2) dissipates 3.39 microwatts - nearly a 27% difference in power. Even if we include the power of the OR gate we find that (1) dissipates 8.34 microwatts and (2) dissipates 7.09 microwatts which is still a 15% difference in total power consumption. Hence, we see that the optimal zero delay decomposition algorithm can be very suboptimal in practice.

3.2 A Technology Decomposition Algorithm

We propose a greedy approach for decomposing an n input gate. The algorithm works as follows:

1. Enumerate the gates in topological order.
2. For each gate in the ordering do the following:
   (a) For each pair of signals use Theorem 2.2 to compute the power that would be dissipated by a two input gate with the pair as its fanins.

(b) Combine the two signals which consume the least power. Remove the two signals from the set of signals. Add the output signal of the new gate to the set of signals.

(c) While there is more than one signal remaining in the set of signals go to Step (a).

This is a straightforward greedy heuristic with the important twist that we use Theorem 2.2 to greedily determine the best signal pairings in the decomposition. For both examples in previous section, this heuristic returns the optimal decomposition. Finally, we note that for a combinational circuit the topological ordering guarantees that the decomposition of a gate never affects the glitching properties of a gate that has already been decomposed.

3.3 Experimental Results

Table 1 contains experimental results for technology decomposition in which we compare the power dissipated by circuits synthesized with the balanced approach (denoted by a B) against those synthesized with our heuristic (denoted by an H). For the experiments the primary inputs had on probabilities of .5. We used the power estimation tool in SIS (assuming a 20 MHZ clock and Vdd=5V) and parameters extracted from the technology library mcnc.genlib to measure the power under a general delay model. Under the general delay model we obtain an average power savings of 11.5%. This savings might appear to be small, but the reader is reminded that this is "real" power savings which includes glitching power. We note that our current implementation of the heuristic uses a unit delay model when it greedily decomposes a gate, however, the algorithm can easily be modified to decompose gates based upon the arrival times from a general delay model. We would expect the results to improve with such an enhanced implementation.

Name	PI	PO	Size	B	N	% Red.
cm138	6	8	26	127.9	111.8	12.6
cu	14	11	76	441.4	411.3	6.8
decod	5	16	50	260.9	250.2	4.1
duke2	22	29	885	3733.7	2946.8	21.1
example2	86	66	342	1808.3	1675.0	7.4
frg1	29	3	789	3969.0	3349.4	8.7
misex3c	14	14	1348	6667.1	5643.2	15.4
pcler8	27	17	78	316.1	277.4	12.2
ttt2	25	21	652	3695.0	3355.7	9.2
vg2	25	8	796	3947.1	3248.4	17.7
Average						11.5

Table 1: Comparison of decomposition heuristics under a general delay model as measured by SIS

4 Low Power Retiming of Sequential Circuits

Retiming is a procedure that was originally proposed by Leisersen and Saxe in [5] to reduce the cycle time of sequential circuits. Informally, retiming can be described as a repositioning of latches in a circuit that preserves the functionality of a circuit.

From Theorem 2.1, we know that in a combinational circuit the number of times a gate experiences a transition at its output is related to the number of distinct path lengths from the primary inputs of the circuit to the gate. For a sequential circuit the number of times that a gate experiences a transition is related to the number of distinct path lengths from the nearest level of latches in the transitive fanin of the gate. This is because the latches are controlled by a global clock which prevent the propagation of glitches into the transitive fanout of the latch. Retiming repositions the latches in a circuit consequently affects the number of times each gate may experience a transition. Consider the example in Figure 4. The numbers beside the gates are the number of distinct path lengths to each gate from the nearest level of latches in the transitive fanin of the gate. A forward retiming of g_2 has many implications. First, g_1 and g_2 become more imbalanced and will tend to glitch more. It is also very important to note that although the glitches at g_2 will not be propagated to g_3 and beyond, the retiming has made g_3 more imbalanced. Hence, g_3 and its transitive fanout may actually glitch *more* because of the retiming.

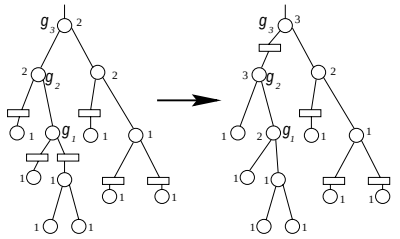

Figure 4: Retiming affects power dissipation in a circuit

4.1 Previous Results in Low Power Retiming

There are two previous results on low power retiming. In [6], the authors conduct an empirical study of power dissipation when varying levels of pipelining are added to a circuit. They observe that by adding more levels of pipelining to a sequential circuit more gates are likely to have balanced paths and so there is a power reduction. They also note that the power reduction is eventually offset by the power consumed by additional latches required for deep pipelining. In [7], the authors propose an algorithm for pipeline insertion in a sequential circuit. Since this is the only known algorithm for low power retiming, it warrants close inspection. The first step of their algorithm is to perform a detailed power estimation of all the gates in the network. Each gate is then weighted by three factors: (1) The amount of glitching activity at the gate (the difference between the switching activity under a general delay model and the zero delay model) (2) The probability that a transition at a gate results in transitions in the gate's transitive fanout (at most two levels ahead) (3) The number of fanouts of the gate. Latches are then placed in a greedy fashion in front of the gates based upon their weights. In an attempt to reduce the total latch count, the final step of their algorithm is to forward retime gates which are not on the same paths as the gates that were greedily retimed. The approach in [7] has three limitations: (1) Their algorithm requires a costly simulation for each level of pipelining that is inserted. (2) As was previously illustrated in Figure 4, placing a latch in front of a gate may prevent the propagation of a glitch at the cost of the generation of a new glitch. (3) Their algorithm may introduce large latch counts.

4.2 Estimation of Power in Sequential Circuits

An ongoing and considerably difficult research problem is that of estimating power in a sequential circuit [7]. Before we present our low power retiming algorithm we refine our estimation procedure in Section 2 to handle sequential circuits:

1. We approximate the total power consumed as $\alpha \cdot S + \beta \cdot K$ where S is our estimate of the switching activity, K is the total latch count, and α and β are user specified weights.
2. For a latch L on a cycle we set the transition probabilities to be: $P_{L,t_1}(0 \rightarrow 1) = .25$, $P_{L,t_1}(1 \rightarrow 0) = .25$, $P_{L,t_1}(0 \rightarrow 0) = .25$, and $P_{L,t_1}(1 \rightarrow 1) = .25$ where t_1 is the time instant that a new set of inputs is applied to the primary inputs in the circuit.

4.3 A Low Power Retiming Algorithm

We have several insights from the analysis of the pipeline insertion algorithm from [7]. First, retiming an individual gate can have a significant impact on the total switching activity in the circuit. Thus, any retiming based upon the switching activity of individual gates is not sufficient to reduce the total power. Second, the latch count can have a significant effect on the power consumption of the circuit. However, we note here that retiming a circuit to minimize the total number of latches (a problem that has already been solved in [5]) is not tantamount to minimizing power because the cost in terms of latch power may vary with different technologies and, of course, the switching activity in the combinational portions of the circuit will have a significant impact on the total power. These observations lead to the following algorithm:

1. Enumerate the gates in topological order.
2. For each gate g in the ordering:
   (a) Backward retime g.
   (b) If g is retimed then estimate the power as described in Section 4.2.
      i. If the power increases then restore the latches to their original positions.
      ii. If the power decreases then anchor the latches behind g.

As with the technology decomposition algorithm we process the gates in topological order. Hence, we will never move latches from the fanins of gates that have already been retimed.

4.4 Experimental Results

We present three tables of experimental results. Since there are not any known low power retiming algorithms for sequential circuits, we use a greedy heuristic similar to the pipelining approach in [7] as a point of comparison. The greedy heuristic backward retimes gates based upon their glitching activities (the difference between the estimated unit delay power and zero delay power). In the greedy heuristic, if a gate is retimed the latches behind the gate are then anchored and the gate cannot be retimed further. $H(100,1)$ denotes our approach in which each transition is assigned a weight of 100 and each latch is assigned a weights of 1. The columns designated as *Constrained* indicate that for both the greedy heuristic and our approach all the retimings were limited to within at most 5 levels of logic. *Unconstrained* indicates that the retimings were not limited. L denotes the total number of latches in the retimed circuit. T denotes the total power (switching activity and latch power). C denotes the combinational power exclusively. In Table 2 we consider the total power consumed by the circuit. We estimated the power using the power estimation tool in SIS (assuming $V_{dd} = 5V$ and a 20MHZ clock) under a general delay model with delay parameters extracted from mcnc.genlib and mcnc_latch.genlib. All of the primary inputs had on probabilities of .5. In Table 3 we consider just the combinational power, and the measurements are organized in an identical fashion to Table 2. Finally, in Table 4, we report the total power, and the measurements are organized in an identical fashion to Table 2.

We immediately see from Table 2, that under a general delay model we attain an average 49% power savings over the greedy approach. A close inspection reveals that the circuits, retimed by our algorithm have far fewer latches than the greedy approach. Thus, a large portion of the power savings is due to the difference in the number of latches. However, when we just the consider the combinational power the results are quite interesting. From Table 3 we see that we attain about an average 7% power savings over the greedy heuristic in terms of pure switching power. Furthermore, for some circuits such as s420 our heuristic achieves as much as a 28% reduction in combinational power. This indicates that the heuristic does indeed reduce the overall switching activity. This is especially significant because the results in [6] indicate that an increased number of latches in a circuit achieve greater balance and consequently reduce switching activity due to glitches. But our results show that a **judicious placement of fewer latches can actually attain better overall balance than a greedy placement of more latches in a circuit**.

Finally, we note that in reality the power consumed by a single latch is considerably greater than the power consumed by a single transition. Thus, if we use more realistic weight assignments for the transitions and latches, we see that even further power reduction is possible. In Table 4, we show compare three variants of our heuristic allowing unconstrained retiming. We see that the more realistic weight assignment in $H(1,100)$ garners an additional 7% power reduction.

Benchmark Description				Unconstrained		
				$H(100,1)$	$H(1,1)$	$H(1,100)$
Name	PI	PO	S	T	T	T
mm4a	7	4	299	1028.4	1040.2	1065.4
s208	10	1	125	306	333.3	300
s298	3	6	195	551.8	559.6	530.6
s344	9	11	211	839.8	852	850.1
s349	9	11	218	854.9	874.3	859.4
s386	7	7	203	717.6	700.9	702.9
s400	3	6	260	930.3	418.5	407.2
s420	18	1	261	385.9	390.2	363.1
s641	35	23	206	850.9	849.9	845.3
Average				718.40	668.77	658.22

Table 4: Comparison of three versions of our heurstic under a general delay model with SIS.

Benchmark Description				Constrained					Unconstrained				
				Greedy		$H(100,1)$		%	Greedy		$H(100,1)$		%
Name	PI	PO	S	L	R	L	T	Sav.	L	T	L	T	Sav.
mm4a	7	4	299	16	1069.1	12	1014.6	5.0977	14	1057.2	12	1028.4	2.7242
s208	10	1	125	25	636.6	8	316.4	50.298	32	1038.9	8	306	70.546
s298	3	6	195	97	1939.4	18	698.8	63.968	70	1505.1	14	551.8	63.338
s344	9	11	211	48	1689.4	15	835.2	50.562	48	1684	15	839.8	50.131
s349	9	11	218	49	1724.6	15	857.8	50.261	49	1728.3	15	854.9	50.535
s386	7	7	203	86	2049.9	6	730.3	64.374	86	2025.3	6	717.6	64.568
s400	3	6	260	120	1689.7	65	869.7	48.529	95	1232.2	65	930.3	24.501
s420	18	1	261	46	931.1	16	394	57.684	62	1730.9	16	385.9	77.705
s641	35	23	206	74	1676	19	836.3	50.101	63	1504.2	19	850.9	43.432
Average								48.99					49.72

Table 2: Total power of greedy and $H(100,1)$ under a general delay model with SIS.

Benchmark Description				Constrained					Unconstrained				
				Greedy		$H(100,1)$		%	Greedy		$H(100,1)$		%
Name	PI	PO	S	L	C	L	C	Sav.	L	C	L	C	Sav.
mm4a	7	4	299	16	841.6	12	895.7	-6.4282	14	903.9	12	873.4	3.3743
s208	10	1	125	25	232.1	8	253.0	-9.0047	32	299.1	8	247.0	17.419
s298	3	6	195	97	393.9	18	442.4	-12.313	70	374.8	14	362.4	3.3084
s344	9	11	211	48	608.0	15	544.9	10.378	48	598.4	15	553.9	7.4365
s349	9	11	218	49	619.6	15	570.2	7.9729	49	626.4	15	542.4	13.41
s386	7	7	203	86	695.8	6	674.1	3.1187	86	696.0	6	660.7	5.0718
s400	3	6	260	120	295.9	65	286.8	3.0754	95	301.6	65	344.7	-14.29
s420	18	1	261	46	344.0	16	310.8	9.6512	62	443.7	16	319.4	28.014
s641	35	23	206	74	670.0	19	686.6	-2.4776	63	660.3	19	678.0	-2.6806
Average								0.44					6.78

Table 3: Combinational power of greedy and $H(100,1)$ under a general delay model with SIS.

5 Conclusion

In this paper we used an estimate of the general delay switching activity to synthesize low power circuits. This estimate was the basis of algorithms that addressed two problems: (1) Low power technology decomposition of simple gates under a general delay model (2) Low power retiming of sequential circuits under a general delay model. For both problems, we attained good power savings when measuring the power under an accurate delay model. There are other difficult problems in low power synthesis. For example, low power technology mapping under a general delay model remains a challenge (since mapping often undoes premapping optimizations). Prior work in low power mapping has always been based upon a zero delay model. We believe that approaches similar to those presented in this paper could yield new insights into problems such as technology mapping.

References

[1] Daniel Brand and Chandu Visweswariah. Inaccuracies in power estimation during logic synthesis. In *International Conference on Computer-Aided Design*, pages 388–384, 1996.

[2] Anantha P. Chandrakasan and Robert W. Broderson. *Low Power Digital CMOS Design*. Kluwer Academic Publishers, 1995.

[3] Jose C. Costa, Jose C. Monteiro, and Srinivas Devadas. Switching activity estimation using limited depth reconvergent path analysis. In *International Symposium on low power electronics and design*, pages 184–189, 1997.

[4] Eric Lehman, Yosinori Watanabe, Joel Grodstein, and Heather Harkness. Logic decomposition during technology mapping. In *International Conference on Compuer-Aided Design*, pages 264–271, 1995.

[5] Charles E. Leiserson and James B. Saxe. Retiming synchronous circuitry. *Algorithmica*, 6:5–35, 1991.

[6] Jeroen Leitjen, Jef van Meerbergen, and Jochen Jess. Analysis and reduction of glitches in synchronous network. Received: 1996.

[7] Jose Monteiro and Srinivas Devadas. *Computer-Aided Design Techniques for Low Power Sequential Logic Circuits*. Kluwer Academic Publishers, 1997.

[8] F. Najm. Transition density, a stochastic measure of activity in digital circuits. In *International Conference on Compuer-Aided Design*, pages 644–649, June 1991.

[9] U. Narayanan, Hon Wai Leong, Ki-Seok Chung, and C. L. Liu. Low power multiplexer decomposition. In *International Symposium on low power electronics and design*, pages 269–274, 1997.

[10] U. Narayanan and C. L. Liu. Low power logic synthesis for xor based circuits. In *International Conference on Computer-Aided Design*, 1997.

[11] K. Parker and E. McCluskey. Probabilistic treatment of general combinational networks. *IEEE Transactions on Electronic Computers*, C-24(6):668–670, 1975.

[12] G. I. Stamoulis. A monte-carlo approach for the accurate and efficient estimation of average transition probabilities in sequential logic circuits. In *IEEE Custom Integrated Circuits Conference*, pages 221–224, 1996.

[13] Sarma B. K. Vrudhula and Hong-Yu Xie. Techniques for cmos power estimation and logic synthesis for low power. In *International Workshop on Low Power Design*, pages 21–26, 1994.

[14] Neil H. E. Weste and Kamran Eshraghian. *Principles of CMOS VLSI Design: A Systems Perspective*. Addison-Wesley, 1993.

[15] Hai Zhou and D. F. Wong. An exact gate decomposition algorithm for low-power technology mapping. In *International Conference on Computer-Aided Design*, 1997.