

Post-Mapping Transformations for Low-Power Synthesis

RAJENDRAN PANDA* and FARID N. NAJM

Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801

We propose a logic synthesis system that includes power optimization *after* technology mapping. Our approach is unique in that our post-mapping logic transformations take into account information on circuit delay, capacitance, arrival times, glitches, etc., to provide much better accuracy than previously proposed technology-independent power optimization methods. By changing connections in a mapped circuit, we achieve power improvements up to 13% in case of area- or delay-optimized circuits, with reductions also in area and delay. We show that by applying the proposed technique on circuits that are already restructured for lower switching activity using the technique presented in [11], total power savings up to 59% in case of area-optimized circuits and 38% in case of delay-optimized circuits are achievable. The post-mapping transformations are based on the *transition density* model of circuit switching activity and the concept of *permissible logic functions*. The techniques presented here are applicable equally well to both synchronous and asynchronous circuits. The power measurements are done under a general delay model.

Keywords: Logic synthesis, low power, VLSI cad, transduction, don't care optimization

1. INTRODUCTION

The high device count and clock frequency of modern ICs has made power dissipation of VLSI chips a major consideration during chip design. Hence the need for low-power logic synthesis techniques to aid in the automatic design of low-power chips. While it is important to attack the synthesis problem at all levels, in this paper we focus on the *logic synthesis* phase and propose a new algorithm for optimization in a post-technology mapping step that achieves lower power circuits on a variety of test cases.

In the popular CMOS technology, the power dissipated by logic gates is almost entirely due to the logic transitions. Thus the power-dissipation depends on the *switching activity* inside the circuit, which in turn depends on the *input switching pattern* and the *specific implementation* of the circuit. This complicates the low-power synthesis problem, since (i) exact input signals may not be known during the design phase and (ii) the traditional synthesis techniques of optimizing a technology independent circuit model become inadequate.

To get around the input pattern-dependence problem, one can use probabilities to describe the

*Corresponding author.

set of *all possible logic signals*. Specifically, we will use a measure of switching activity, called the *transition density* [1], that can be efficiently evaluated without requiring *exact* information about the primary input signals. In order to automatically find a low-power implementation of a circuit design, we propose a new power-reducing logic transformation to be applied after the technology-mapping stage of logic synthesis. This transformation is proposed to augment any power optimization effort that is carried out at the technology-independent phases of logic synthesis. In particular, we demonstrate the effectiveness of our post-mapping procedure on the circuits that are restructured for minimum switching activity at the *technology decomposition* stage.

The transition density at a node x in the circuit is the *average number of logic transitions per second* at that node, denoted by $D(x)$. If $D(x_i)$ is the transition density at the output x_i of the i th logic gate, then the average power consumed by the circuit is:

$$P_{av} = \sum_{i=1}^n \frac{1}{2} V_{dd}^2 C_i D(x_i)$$

where C_i is the total capacitance at the output of *gate_i*.

In order to lower the circuit power, one would like to automatically design the circuit in such a way as to either reduce n , or to reduce the values $C_i D(x_i)$. Reducing n is equivalent to the traditional synthesis concern of reducing area. Reducing the $C_i D(x_i)$ terms, however, is strictly a power concern and requires new synthesis algorithms. Even though gate area might be a measure of C_i , minimizing area does not necessarily minimize the power because it is the *cross-terms* $C_i D(x_i)$ that have to be minimized, and not only the C_i terms.

As we will review in the next section, most existing low-power synthesis techniques optimize a technology-independent model of the circuit in which the gates are assumed to have zero-delay. This, we will show, is inadequate for low-power synthesis. Our approach is unique in that we take

the circuit technology into account and make detailed considerations of delay, capacitance, rise/fall times, glitches, etc., in our procedure to minimize the $C_i D(x_i)$ terms. Thus, we take into account the so-called *glitch power*, which is an important feature of our approach and differentiates this work from others. Briefly, the glitch power is a component of the power dissipation that is due to spurious unplanned-for transitions inside a logic circuit. These transitions, or *glitches*, arise from the unequal delays of paths that the signals take. Throughout this work, we will assume that we are dealing with a combinational circuit block which is embedded in a synchronous sequential circuit. For more details on glitch power and other related issues of power estimation, the reader is referred to [2].

We will make use of the concept of permissible functions [3], which we briefly review in the appendix. In Section 2 we give a short review of previous work in this area and in Section 3 an overview of our system. Details of the proposed algorithm are given in Section 4 and the experimental results are reported in Section 5. Summary and conclusions are given in Section 6.

2. BACKGROUND

The work in [4] considers several strategies at the technological, architectural and circuit design levels, that can be adopted for optimizing power in VLSI circuits. While reduction of power supply voltage and power-down strategies are already common in industry, additional power reduction can be obtained at the logic optimization and technology-binding phases of logic synthesis.

There has been much recent work on logic synthesis for low power. Unfortunately, most of the proposed techniques have dealt with a *technology-independent* model of the logic circuit, based on the use of a *zero-delay* timing model for logic gates. We briefly cite some of this previous work and then give a simple example to illustrate that optimization with no regard to delay is not

adequate for low-power synthesis. In contrast to these techniques, our optimization procedure includes transformations that are applied after technology mapping, and makes use of the circuit delay information.

The work in [5] considers the switching activity contribution of kernel intersections while extracting new nodes as factors of other nodes. It uses transition density [1] to evaluate the node power. Another approach [6] uses selective collapsing of nodes on non-critical paths, timing optimization and node minimization, as strategies to optimize power during the logic optimization phase. The work in [7] presents a technique to minimize the *zero-delay* power by changing the Boolean function at the nodes. The problem of power minimization at the technology-binding phase (technology decomposition and technology mapping) of logic synthesis has also been actively addressed [8–11].

The power computed under a zero-delay assumption (or simply, zero-delay power) is not an appropriate guiding measure for low-power optimization. This is due to the fact that an optimization step that improves zero-delay power does not provide a guarantee about the improvement of the total power. To prove this point, we show in Table I below two different implementations of a reasonably-sized benchmark circuit, *vda*. The second implementation dissipates 35.4% more power than the first implementation, despite the fact that its zero-delay power is 2.8% less and area is 1.9% less. Both these implementations had the same optimization route up-to and including technology mapping, but differed in their post-mapping optimization.

Finally, some techniques have been proposed for technology-dependent optimization, along the lines of transistor/gate sizing. Thus [12] proposes

resizing of gates on non-critical paths, taking false paths into account. Resizing is of course a viable optimization strategy, but we will show that there is scope for more optimization by applying logic transformations that make use of the delay and functionality information.

3. SYSTEM OVERVIEW

A schematic overview of our low-power logic synthesis system, called *LogicPower* is shown in Figure 1. The solid lines show the current implementation.

The main synthesis flow is the vertical downward path on the right. The three blocks of LOGIC MINIMIZATION, TECHNOLOGY DECOMPOSITION and TECHNOLOGY MAPPING represent the traditional SIS synthesis flow [13]. We have found scope for optimization in the technology decomposition step, as reported in [11]. The technology decomposition algorithm uses transition density as a measure of internal switching activity, and the density values are computed, in a separate block, as shown in the figure, using [1] and [14]. We have also added a post-mapping block, which is the subject of this paper, labeled GATE-LEVEL TRANSFORMATIONS in the figure. Post-mapping transformations have the potential for high-accuracy because they use information on the circuit delay, glitches, etc. These post-mapping transformations are unique to our approach.

In the remainder of this paper, we will discuss *cost functions* that make use of the delay information and are useful inside the optimization loop to determine the favorable moves. In addition to that, our algorithm also uses an accurate measure of power dissipation and the glitching component of power, based on [15], to guide the transforma-

TABLE I Example of optimization leading to favorable zero-delay power but worse total power

Implementation	#Gates	#Connections	Area	ZDPower	Total Power	Delay
A	469	948	727552	412.993	536.083	17.476 nS
B	459	933	713632	401.472	725.661	19.450 nS

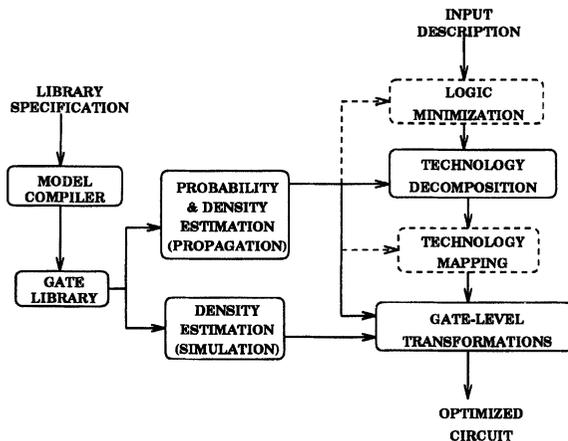


FIGURE 1 Overview of logicpower.

tions and also to verify the final results. The power estimation uses a general delay gate level model and is accurate to within a user-specified error bound, with user-specified confidence. It takes into account fanout capacitance, rise/fall times, inertial delay and propagation delay.

The LOGIC MINIMIZATION and TECHNOLOGY MAPPING remain the same as in the MIS/SIS synthesis system.

4. POST-MAPPING TRANSFORMATIONS

We shall first explain the motivation for doing transformations after the technology mapping stage. Consider the SIS-style of optimizing a technology-independent network and later mapping it to a set of gates in a given technology, with the objective of minimizing the area/speed/power. Such an optimization process may give a circuit implementation that possesses one or both of the following undesirable features:

- (i) The switching consists of excessive glitches.
- (ii) Some nodes with very high switching activity are driving large loads.

Both of the above situations are undesirable for power and thus a post-mapping step that can

rectify the above situations becomes useful. Besides, the performance of any power optimization procedure depends heavily on the accuracy of power estimation and the availability of accurate information about the various circuit parameters that affect the power. These parameters are the gate capacitances, the amount of switching at the nodes, the glitching component of switching activity of the nodes, the arrival times of signals, and finally the propagation and inertial delays of the gates. As all these factors can be accurately assessed after the technology mapping step, it follows that power optimization can be done very effectively at the post-mapping stage.

Another favorable characteristic of post-mapping optimization is its ability to augment any power optimization done at the previous stages of synthesis. In fact, we will show through experimental results that the proposed post-mapping transformations work very well with and augment the savings obtained through another power optimization technique [11], applied at the pre-mapping stage, thus resulting in large combined reduction in the power consumption. While the technology-independent optimizations are subject to the risk of being partly undone by the restructuring of the circuit during the subsequent steps of optimization, the proposed technology-dependent post-mapping procedure suffers much less from such a risk.

The basic idea behind our post-mapping transformation is to change the connections configuration of the network such that:

- (i) Inputs of gates are driven by signals with less transition density, instead of by signals with higher transition density.
- (ii) The glitches at the output of gates are reduced.
- (iii) Any gate or connection that becomes redundant in the process of achieving the above is removed.

This reconfiguration is achieved by substituting the fanout connections of gates with connections

from other gates, if such substitutions will decrease the total power dissipation in the network, provided the Boolean functions at the nodes remain correct. To determine which connections can substitute which, we use the concept of Permissible Functions [3].

To illustrate how this works, we refer to Figure 2. Consider the fanout branch of gate 1 that is driving one of the inputs of gate 2 (the *nand* gate). There may be other nodes in the circuit (other than *f*) which can be used to drive that same input of gate 2, without altering the functionality of the circuit. The Boolean functions at these other nodes are said to be *permissible* for this fanout connection. Thus, we could disconnect the input of gate 2 from gate 1 and instead connect it to some other gate in the circuit. Of course, this is a useful transformation only when it leads to lower power. As a side note, in the course of applying this transformation, if both the fanouts of gate 1 get substituted by connections from other nodes, then gate 1, and possibly some more gates in its input cone, can be eliminated.

This type of transformation has been referred to as *generalized gate substitution* and has been used with the objective of eliminating gates and/or connections [16]. We use this transformation, but in a different manner, for reducing power.

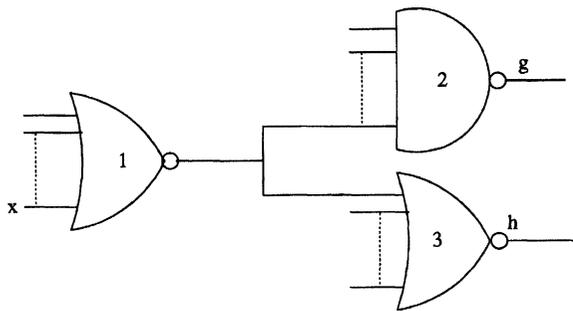


FIGURE 2 Example to illustrate the use of permissible functions.

In the course of the transformation, we do not want the Boolean function at node *g* (in the illustration) to change, since that would alter the Boolean functions, and hence the zero-delay activity, of the nodes in its transitive fanout, and would have an unpredictable global effect on the circuit power. So we restrict the set of candidate substitutions for a connection to be such that the output function of the gate driven by this connection does not change. We ensure this by using the *Locally-derived Set of Permissible Functions* [17] for the connection.

Originally, LSPFs were used in (re)synthesis [17], with the objective of speeding up the calculation of permissible functions, at the cost of sacrificing the optimality. The restriction that the immediate output function should not change is responsible for reducing the don't care space of the permissible functions to a smaller subset. In our approach, we take advantage of this restriction, as that is exactly what we wish to achieve, viz., preserving the Boolean functions at all the nodes in the network, so that the zero-delay activity of the nodes is well under control.

In order to substitute a connection c_{ij} from gate g_i to gate g_j , the procedure finds all the candidate nodes, g_k which satisfy the following requirements:

- (i) $F(g_k) \in LSPF(c_{ij})$
- (ii) Maximum arrival time of signal at g_k is less than minimum arrival time of signal at g_i , or Level(g_k) is less than or equal to Level(g_i).

Condition (i) ensures that the Boolean functions at the other nodes would remain the same and thus the *zero-delay* power, which is usually the major component of the *total* power, would remain unaffected at these other nodes. Condition (ii) ensures that no feed-back loop is created, if c_{ij} is substituted by a connection from g_k . Condition (ii) also helps to keep the overall delay affected less after the substitution. Among the various candidates (including g_i itself) the one that would result in maximum power reduction is chosen to do the substitution. The strength of our procedure lies in

the detailed consideration of the various parameters related to the modification of total power due to this substitution. We consider four contributions to the power saving, when a connection c_{ij} from gate g_i to gate g_j can be substituted by a connection from node, g_k . They are described below.

4.1. Power Cost Function

We use a three-part cost function. If x is a candidate node for substituting an input of a gate, Y , then the *power cost* of using x to drive Y is given as:

$$\begin{aligned} \text{PowerCost}(x, Y) = & \text{LoadingCost}(x, Y) \\ & + \text{GlitchingCost}_1(x, Y) \\ & + \text{GlitchingCost}_2(x, Y) \end{aligned}$$

The *LoadingCost* term in the above considers the transition density of x and the amount of load on x due to the input capacitance of Y . The *GlitchingCost* terms consider the effect on total power due to the glitches at x , and due to the glitches at the other inputs (i.e., side inputs) of Y , respectively. The following four considerations explain how these three cost factors are calculated.

4.1.1. Consideration of Loading

In Figure 3, substituting the connection from z to Y by a connection from x is attractive, if $D(x) <$

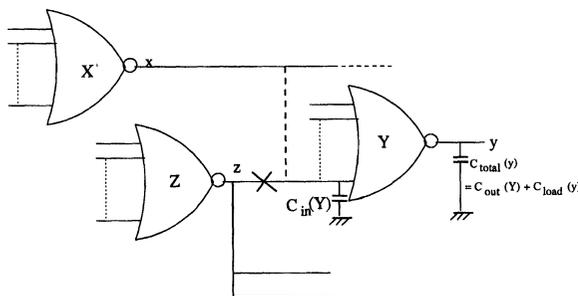


FIGURE 3 Considering transition density for connection substitution.

$D(z)$. If this substitution is done, the capacitance load of the input pin of gate Y , $C_{in}(Y)$ will get shifted to node x . So, we take the first component of the power cost as the transition density of x , weighted by the additional capacitance it will have to drive, if the substitution takes place. That is:

$$\text{LoadingCost}(x, Y) = D(x) \times C_{in}(Y)$$

4.1.2. Consideration of Arrival Times with Glitches

Here we associate a cost with the amount of glitches present in a signal, as some or all of these glitches can pass through the gate to which this signal is made an input, causing power dissipation at that gate's output. We call this cost as *GlitchingCost*₁. If two or more signals transition at the inputs of a gate at the same time or within a duration separated by the inertial delay of the gate, then at most only one transition will occur at the output of the gate. This suggests that what is crucial in passing the glitches of the input signals through a gate is the non-simultaneity in the switching of these input signals. To evaluate the various candidate signals for substituting an input of a gate, we devise below a heuristic measure of non-simultaneity of switching of the candidate with respect to the side inputs to the gate.

DEFINITION 2 (Activity Window) The *activity window* of a signal arriving at the input of a gate is the time window bounded by the minimum and the maximum arrival times of the signal at the input of that gate. It is the window within which all the transitions of the signal at the input of that gate can occur.

DEFINITION 3 (Critical Windows) The *critical windows* of a signal, x , with respect to a signal, s , both arriving at the input of a gate, are those segments of the activity window of x that are *not* overlapping with the activity window of s .

Note that the total duration of the critical windows of a signal with respect to another signal

provides a heuristic measure on the non-simultaneity in the switching of these two signals at the input of the gate. We use this measure to ascertain approximately what proportion of the glitches in a signal will pass through a gate to which this signal is made an input.

Referring to Figure 3, the glitching cost of x , w.r.t the gate, Y , is taken as the amount of glitches in x , weighted by a non-simultaneity factor of x , $\alpha(x)$, w.r.t. the other inputs of Y and the capacitance, $C_{total}(y)$, which the resulting output glitches have to drive.

Thus, the $GlitchingCost_1$ of x driving gate Y is given by the equation:

$$GlitchingCost_1(x, Y) = \alpha(x)[D(x) - D_0(x)]C_{total}(y)$$

where $C_{total}(y)$ is the sum of the output capacitance of gate, Y , and its load, and, $D(x)$ and $D_0(x)$ are the *total* and *zero-delay* transition densities of x , and thus their difference is the amount of glitches in x .

The non-simultaneity factor, $\alpha(x)$, is the average over all the side inputs to Y of the relative durations when x will not switch simultaneously with the side input, and is calculated as:

$$\alpha(x) = \frac{1}{|\text{inputs}(Y)|} \sum_{\substack{s \in \text{inputs}(Y) \\ s \neq x}} \left(\frac{\text{Total Duration of Critical Windows of } x \text{ w. r. t. } s}{\text{Duration of Activity Window of } x} \right)$$

Figure 4 illustrates the *rationale* behind this cost function. Of the two candidate signals x_1 and x_2 in the figure, note that $D_{x_2} > D_{x_1}$. Though the loading cost considerations would justify the choice of x_1 over x_2 , it *may be* worthwhile, however, to prefer candidate x_2 , to x_1 , for a large proportion of the glitches in x_1 occur outside the *activity windows* of other input signals, while most of the switching of x_2 occurs simultaneously with some of the other inputs.

Note that, in the above, we have not considered the fact that the gate might be desensitized to an

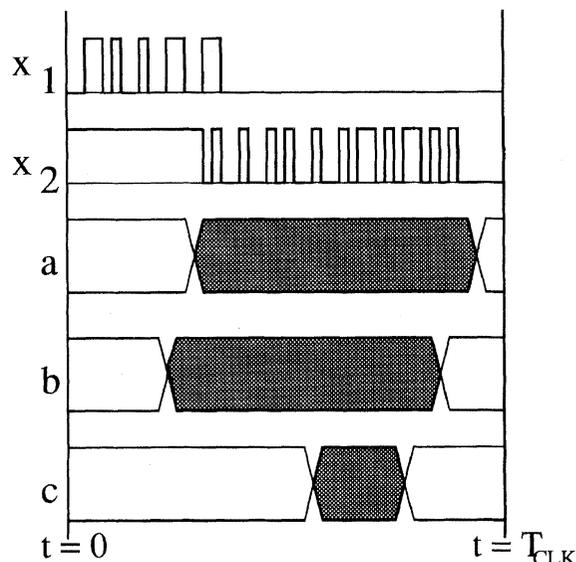


FIGURE 4 Considering non-simultaneity in switching for connection substitution.

input transition. We account for that factor in the third part of our power cost function, viz., $GlitchingCost_2$.

4.1.3. Consideration of Signal Probabilities with Glitches

As the connection substitutions are chosen with the constraint that the Boolean functions of the gates in the network (and hence their zero-delay activity) should not change, the differences in the signal probabilities of the candidate inputs to a gate would have *no* effect on the zero-delay power at the gate's output. However, they can have a pronounced effect on propagating the glitches of the side inputs to the gate's output. If the signal probability of a candidate input is such that it sensitizes the gate to the transitions at the other inputs for *unduly* long durations, then the glitch power at the output of the gate can become very large, making this candidate unsuitable for substitution.

As an example, and referring to Figure 2 again, it is preferable to replace the fanout connection

driving the *nand* gate by a signal (from the candidate set) that has the least signal probability (of logic ONE). This is because a low probability at one input of a NAND/AND gate inhibits to a greater extent the glitching transitions at its other inputs from going through. Likewise, it is desirable to replace the other fanout connection (driving the *nor* gate) by a candidate having the largest probability. In general, this strategy applies to any logic gate, simple or complex, and aims at reducing the Boolean difference probabilities, which contribute to the propagation of transitions in the circuit [1].

This notion leads to the third cost factor, viz., the *GlitchingCost*₂.

$$\begin{aligned} \text{GlitchingCost}_2(x, Y) &= |P(x) - \hat{P}(x)| \\ &\quad \times [D(y) - D_0(y)] \times C_{\text{total}}(y) \end{aligned}$$

where x is the input signal under evaluation, y is the output node of gate, Y , whose input connection is under substitution, $D(y)$ and $D_0(y)$ are the *total* and *zero-delay* transition densities of the output node, $C_{\text{total}}(y)$ is the total capacitance of the output node, $P(x)$ is the signal probability of x and $\hat{P}(x)$ is the *desired* probability for the input terminal of x . The desired probability is the probability of a signal that will completely desensitize the gate to the transitions of the other inputs. Its value is 0 or 1, depending on the type of the gate (*nand*, *nor*, *aoi* etc.) and the input pin. If the gate is an *xor/xnor*, then this term of the power cost is not considered, as in such a case the gate remains sensitive to input transitions all the time. Note that, in the above expression, we take the output glitches to approximate the sum of glitches at the input nodes other than x .

4.1.4. Consideration of Gate Eliminations

Although we carry out the connection changes on an already optimized circuit, we may not want to rule out the possibility of residual redundancies at the post-mapping stage. Even if the original circuit were to be irredundant, we would want the

connection substitutions to take the circuit from one irredundant form to another, preferably to one having a lower number of gates. Since a gate can be eliminated only after substituting *all* its fanouts, it is desirable that we make the power cost function sensitive to the opportunities of eliminating a gate. We do so by considering an opportunity cost in the power cost function, while evaluating the candidate inputs.

If the redundancy of a connection is such that its removal changes the Boolean function at some of its successor nodes, then the total *zero-delay* power of the network, which is the major component of the *total* power of the network, can change in an unpredictable manner. However, if the removal does not cause change of Boolean functions at any node, then we are at least assured of a reduction in the total zero-delay power of the network. As this is exactly what we guarantee by using LSPF, then removal of redundancies in this way is *most likely* to reduce the total power consumed in the circuit. So is the case also when a gate is eliminated by substituting all its fanouts through the use of LSPF of these fanouts.

Consider Figure 2 again. Suppose we were able to replace one of the fanouts of *gate 1* with an attractive alternative. If the second fanout connection also has an alternative (though not attractive), it means that *gate 1* is *eliminable* in the circuit. However, it can not be eliminated before the other fanout is substituted. To help the process of eliminating *gate 1*, we make *every* fanout substitution of this gate more attractive, by considering a larger capacitance load in the *LoadingCost* term of the cost function. As a heuristic, we include the drain capacitance of *gate 1* in the *LoadingCost* term to reflect the opportunity cost of eliminating *gate 1*.

4.2. Algorithm

The procedure of changing the connections configuration is applied iteratively to the network, until the total power of the network shows no further improvement. Each iteration attempts to

substitute the fanout connections, beginning from the primary output nodes and sweeping towards the primary inputs. The capacitances at the nodes, the arrival times and the level information are calculated only once during each iteration. The transition densities at the nodes are estimated through statistical simulation [15] at the beginning of each iteration. The signal probabilities of the nodes are calculated just once at the beginning of the procedure, using the BDDs of the global functions at the nodes. As no new nodes are added and no Boolean function in the network is changed during the iterations, the probability values calculated at the beginning hold good throughout the procedure. The algorithm is given in Figure 5.

5. EXPERIMENTAL RESULTS

For evaluation of our above algorithm, we started with circuits that were already well optimized using the *rugged.script* of the SIS optimizer. For a number of MCNC benchmark circuits, two sets of circuit implementations were generated: *reference* circuits, and *study* circuits. The *reference* circuits were obtained using the balanced-tree technology

decomposition of SIS and later mapping for optimum area (or delay), also using the SIS mapper. The SIS *lib2.genlib* library was used for mapping. In the *study* circuits, the technology decomposition was done using the algorithm given in [11]. Then, these circuits were mapped, also for area (or delay) in the same manner as the *reference* circuits, using the SIS mapper. The transformation procedure in Section 4.2 was later applied to these circuits. The *reference* and *study* circuits were then simulated for power estimation, using the statistical power estimator, MED [15]. Circuits were simulated with input signals of probability 0.5 and transition density of 0.5 transitions/clock. The simulation was done till the total power of the circuit converged to an error bound of 2% with a confidence of 95%.

Table II pertains to the circuits that were mapped for optimum area and Table III to the circuits that were mapped for minimum delay. Column 2 of the tables shows the relative size of the circuits, using the number of gates in the 2-input nand/nor implementation. Column 3 is the power consumption of the *reference* circuits in $\mu\text{W}/\text{MHz}$. Column 4 shows the percentage reduction in power of the circuit, as a result of application of our low-power technology decomposition algorithm [15]. Similarly, *further* reduction in power, delay and area achieved by the post-mapping transformation are shown in Columns 5, 6 and 7. Column 8 shows the percentage of connections in the circuit that were substituted using the algorithm presented above. Finally, column 9 shows the total power reduction. A negative number in these tables means an increase in area/delay/power. Note that only the power reductions given under column 5 are the result of application of the proposed post-mapping transformations. The total power reduction figures in column 9 and the low-power technology decomposition results of [11] are presented here to demonstrate that the proposed post-mapping transformations work well with our earlier proposed technique [11], preserving the lower-activity circuit structure achieved at the decomposition phase.

Algorithm. (Post-Mapping Transformation)

```

Calculate signal probability (P(i)) of every node i
Set MinTotalPower :=  $\infty$ 
while (T) {
  Levelize Network
  Calculate MinArrival and MaxArrival for every node
  Simulate density & power at all nodes using [15]
  if(NetworkPower  $\geq$  MinTotalPower) EXIT
  Set MinTotalPower := NetworkPower
  Accept current network as the best
  foreach node (i)
    foreach fanout connection ( $C_{ik}$ ) of i
      Calculate LSPF of  $C_{ik}$ 
      MinCost( $C_{ik}$ ) := PowerCost(i,k)
      Set BestCandidate( $C_{ik}$ ) := i
      foreach node (j) such that Level(j)  $\leq$  Level(i)
        or MaxArrival(j) < MinArrival(i)
          foreach fanout connection ( $C_{jk}$ ) of j
            if  $F(j) \in$  LSPF( $C_{ik}$ ) and PowerCost(j,k)  $\leq$  MinCost( $C_{ik}$ )
              set MinCost( $C_{ik}$ ) := PowerCost(j,k)
              set BestCandidate( $C_{ik}$ ) = j
  Replace fanout connections of i by respective best candidates
}

```

FIGURE 5 Post-mapping transformation algorithm.

TABLE II Evaluation of post-mapping transformations (for min. area mapped circuits)

Circuit Name	Size #Gates	Reference Power μ W/MHz	% Reduction due to				% connection changes	Total Power Reduction %
			Decomp. Pwr	Post-Mapping Pwr	Dly	Area		
apex7	249	6.650	6.1	0.2	9.4	2.1	6.3	
example2	296	12.314	5.7	0.1	-5.8	0.0	5.8	
x4	352	11.424	1.9	0.6	-0.7	0.3	2.5	
x1	368	13.533	11.2	2.5	9.1	1.9	13.7	
alu2	383	19.622	11.1	8.1	27.7	3.4	19.2	
vda	500	24.440	-3.9	2.9	0.5	0.7	-1.0	
i9	539	18.259	15.5	11.3	22.4	0.9	26.8	
alu4	739	37.022	13.3	4.4	11.1	3.0	17.7	
x3	923	30.459	2.8	12.1	21.2	5.9	14.9	
t481	978	23.760	9.6	7.6	4.5	5.1	17.2	
i8	1100	57.630	57.9	0.9	1.6	0.8	58.8	
Average			13.2	4.8	9.2	2.2	15.4	

TABLE III Evaluation of post-mapping transformations (for min. delay mapped circuits)

Circuit Name	Size #Gates	Reference Power μ W/MHz	% Reduction due to				% connection changes	Total Power Reduction %
			Decomp. Pwr	Post-Mapping Pwr	Dly	Area		
apex7	249	12.994	8.5	1.3	-1.0	0.0	9.8	
example2	296	14.167	-9.3	11.1	0.6	2.8	1.8	
x4	352	17.147	-2.1	0.3	2.0	0.3	-1.8	
x1	368	23.136	7.7	4.8	-1.7	2.6	12.5	
alu2	383	25.743	3.3	12.7	17.8	7.8	16.0	
vda	500	29.876	10.3	2.5	7.8	2.3	12.8	
i9	539	42.602	9.8	7.4	-7.4	0.9	17.2	
alu4	739	53.215	17.5	5.0	9.8	3.2	22.5	
x3	923	52.319	-1.9	12.4	-27.2	7.5	10.5	
t481	978	27.603	1.8	11.4	6.9	6.6	13.2	
i8	1100	76.115	28.7	9.6	0.8	4.4	38.3	
Average			6.5	7.1	0.8	3.5	13.0	

The post-mapping transformations achieved power reduction of 4.8% and 7.1% (on average) for the area-mapped and delay-mapped circuits respectively, over and above the power reduction achieved by our low-power technology decomposition algorithm. The area and delay of the circuits also decreased substantially as a result of the post-mapping transformations. The potential for power reduction by this transformation can be very high in case of larger circuits, as indicated by the examples, x3, t481 and i9. As the number of connections grow quadratically to the number of nodes in the circuit, more opportunities for substitution and more candidates for each substitution become possible in larger circuits.

The constraint of preserving the Boolean function at every node after the connections are substituted may appear too restrictive. However, the data under column 8 shows that, on an average, as much as 6.1% of the circuit connections were substituted.

Finally, the effectiveness of the post-mapping transformations in handling both the zero-delay power, as well as the glitch power, has been brought out in Table IV. Column 2 of the table shows the power dissipation due to glitches, as a percentage of the total dynamic power. Columns 3 and 4 show the percentage reduction in zero-delay power and the percentage reduction in glitching power respectively. Column 5 shows the percentage reduction in area, due to the elimination of

TABLE IV Detailed analysis of post-mapping transformation results (for min. delay mapped circuits)

Circuit Name	Glitch Power (as %Total Pwr)	%Reduction in zd Power	%Reduction in Glitch Power	%Reduction in Area
apex7	11.8	0.0	11.0	0.0
example2	17.4	8.4	23.9	2.8
x4	15.7	0.5	-0.8	0.3
x1	11.6	3.5	14.7	2.6
alu2	27.2	10.3	19.1	7.8
vda	23.0	3.2	0.2	2.3
i9	28.8	4.1	15.6	0.9
alu4	33.8	5.6	3.8	3.2
x3	19.5	11.8	14.9	7.5
t481	18.5	9.8	18.5	6.6
i8	25.2	12.9	0.0	4.4
Average	21.1	6.4	11.0	3.5

gates by connection changes. The figures given in that table are for delay mapped circuits.

We can make the following observations from the table:

- (i) It can be seen that the glitching power constitutes a considerable fraction of the total power. It is as high as 34% in some circuits (*alu4*). This reiterates the point that ignoring the glitching power in a synthesis system for power can seriously affect the optimality of the results.
- (ii) Column 3 of Table IV shows an average reduction of 6.4% in zero-delay power, as against only 3.4% reduction in area (column 5). This was made possible by considering the *LoadingCost* of the candidate substitutions and shifting the loads from highly transitioning nodes to nodes with low transition density.
- (iii) Similarly, a comparison of column 4 with column 5 shows the effectiveness of the glitching cost considerations made by the algorithm. There are several examples in the table (*apex7*, *example2*, *x1*, *alu2* and *t481*) for which the reduction in glitching power is much higher than the reduction in area. For instance, *example2* recorded a 23.9% reduction in glitch power as against only a 2.8% reduction in area. This implies that the bulk of reduction in glitch power was realized by the reduction of the total glitching activity in

the circuit, than by the reduction of the capacitances in the circuit.

- (iv) There are also few examples (*x4* and *i8*) for which the algorithm had no or negative effect on glitches. This is possibly caused by the inexactness of the heuristic measure that considers the non-simultaneity in the switching of signals, viz., the *critical windows*, and also due to the presence of false paths that artificially expand the activity windows of signals.

As the transformations are based on locally-derived don't cares that can be calculated very fast, and moreover, since our use of BDD structures for the Boolean functions and the permissible functions permit efficient checking of candidate functions w.r.t. the permissible functions (as these involve implication checks), our post-mapping algorithm is quite fast. The post-mapping algorithm completed in 2 to 5 iterations in all the circuits that we tested. The largest circuit, viz., *i8* took 953 and 873 CPU seconds for the area and delay mapped circuits respectively, on a Sun Sparc-10 workstation.

6. SUMMARY AND CONCLUSIONS

We have presented a methodology for low-power synthesis, wherein we perform change of connec-

tions configuration in a post-mapping phase. We have shown the effectiveness of the algorithm through test results on a number of benchmark circuits. The procedure performs very well in reducing both the zero-delay and the glitching components of power. We have also demonstrated that the procedure works well in conjunction with another circuit restructuring technique applied at the technology-decomposition phase, to result in large combined power reductions.

The idea of making detailed considerations of *transition density*, *signal probability*, *arrival times* and *glitching transitions* of a signal to minimize the circuit power are *unique* to our approach and this approach can be extended also to other post-mapping transformations, like resynthesis and merging of gates. We are currently working on such extensions.

Acknowledgement

This work was supported by the National Science Foundation (NSF) under grant MIP-9308426.

References

- [1] Najm, F., "Transition density: A new measure of activity in digital circuits", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 310–323 (Feb. 1993).
- [2] Najm, F., "A survey of power estimation techniques in VLSI circuits", *IEEE Transactions on VLSI Systems*, pp. 446–455 (Dec. 1994).
- [3] Muroga, S., Kambayashi, Y., Lai, H. C. and Culliney, J. N., "The Transduction Method – Design of Logic Networks Based on Permissible Functions", *IEEE Transactions on Computers*, **38**, pp. 1404–1424 (Oct. 1989).
- [4] Chandrakasan, A. P., Sheng, S. and Brodersen, R. W., "Low-Power CMOS Digital Design", *IEEE Journal of Solid-State Circuits*, **27**(4), pp. 473–484 (April 1992).
- [5] Roy, K. and Prasad, S. (1992). "SYCLOP: Synthesis of CMOS logic for low power applications", *IEEE International Conference on Computer Design*, pp. 464–467.
- [6] Shen, A., Ghosh, A., Devadas, S. and Keutzer, K. (1992). "On average power dissipation and random pattern testability of CMOS combinational logic networks", *IEEE/ACM International Conference on Computer-Aided Design*, pp. 402–407.
- [7] Iman, S. and Pedram, M. (1994). "Multi-level Network Optimization for Low Power", *IEEE/ACM International Conference on Computer-Aided Design*, pp. 372–377.
- [8] Tsui, C.-Y., Pedram, M. and Despain, A., "Technology Decomposition and Mapping Targeting Low Power Dissipation", *30th ACM/IEEE Design Automation Conference*, pp. 68–73 (June 1993).
- [9] Tiwari, V., Ashar, P. and Malik, S. (1993). "Technology Mapping for Low Power", *Proc. 30th ACM/IEEE Design Automation Conference*, pp. 74–79, Dallas, TX, June 14–18.
- [10] Lin, B. and de Man, H. (1993). "Low-Power Driven Technology mapping under Timing Constraints", *International Workshop on Logic Synthesis*, pp. 9a-1–9a-16.
- [11] Panda R. and Najm, F. N., "Technology Decomposition for Low-Power Synthesis", *IEEE 1995 Custom Integrated Circuits Conference*, pp. 627–630 (May 1995).
- [12] Bahar, R. I., Hachtel, G. D., Macii, E. and Somenzi, F. (1994). "A Symbolic Method to Reduce Power Consumption of Circuits Containing False Paths", *IEEE/ACM International Conference on Computer-Aided Design*, pp. 368–371.
- [13] Brayton, R., Hachtel, G. D. and Sangiovanni-Vincentelli, A. L., "Multilevel Logic Synthesis", *Proc. of the IEEE*, **78**(2), pp. 264–300 (February 1990).
- [14] Najm, F., "Low-pass filter for computing the transition density in digital circuits", *IEEE Transactions on Computer-Aided Design*, **13**(9), pp. 1123–1131 (September 1994).
- [15] Xakellis, M. and Najm, F. (1994). "Statistical Estimation of the Switching Activity in Digital Circuits", *31st ACM/IEEE Design Automation Conference*.
- [16] Xiang, X. (1990). "Multilevel Logic Network Synthesis System, Sylon-Xtrans and Read-Only Memory Minimization Procedure, MINROM", Ph.D. Thesis, University of Illinois at Urbana-Champaign.
- [17] Limqueco, J. C. (1992). "Logic Optimization of MOS Networks", Ph.D. Thesis, University of Illinois at Urbana-Champaign.

APPENDIX

Permissible Functions

We briefly review the concept of a permissible function [3], which is at the core of our transformations.

DEFINITION 1 (Permissible Function) A function f is said to be a permissible function of a gate or connection if replacing the function realized at the output of the gate or at the connection by the function f does not cause any primary output of the network to realize an incorrect function.

The Set of Permissible Functions (SPF) at a node or connection can be represented using any two of the ON-, OFF-, and the DC-Sets of vertices for that node/connection. In this work, we will use the

on-, and off-sets of a node in a BDD structure to represent the permissible functions:

$$\text{SPF}(x) = [G^{\text{ON}}(x), G^{\text{OFF}}(x)]$$

where the $G^{\text{ON}}(x)$ and $G^{\text{OFF}}(x)$ are the on-set and off-set of x respectively.

In our post-mapping transformations, we use a subset of the maximal set of permissible functions, known as the *locally-derived set of permissible functions* [17], abbreviated as LSPF. The LSPF has a property that implementing any of the functions in the LSPF of a node or connection, at that node or connection, will not change the Boolean function of the nodes lying in the transitive fanouts of that node or connection. The LSPF is particularly attractive for post-mapping transformations, as we have shown.

If x is an input connection to a gate whose Boolean function in terms of its immediate inputs is y , then it can be shown that the LSPF of the connection x with respect to its output y is:

$$\text{LSPF}(x) = \left[F(x) \frac{\partial y}{\partial x}, \bar{F}(x) \frac{\partial y}{\partial x} \right]$$

where x is the Boolean variable of the connection x , $F(x)$ is the Boolean function of x and $\partial y / \partial x$ is the Boolean Difference of y w.r.t. x , defined as:

$$\frac{\partial y}{\partial x} = y|_{x=0} \oplus y|_{x=1}$$

where \oplus is the exclusive-or function.

Authors' Biographies

Rajendran Panda received a Ph.D., degree in computer science from the University of Illinois at Urbana-Champaign in 1996. He worked with the Coordinated Science Lab., of the university, researching on computer-aided design methods

and synthesis techniques for low-power VLSI circuits. He is currently with Motorola Inc. in Austin, Texas. Earlier, from 1978 to 1991, he served the Bharat Heavy Electricals Ltd., Bangalore, India, specializing in electrical system design, plant/facilities engineering, and project engineering and management. Rajendran received a B.E. (Hons.) degree in Electrical and electronics engineering from Madurai University, India, in 1978 and an L.L.B. degree from Bangalore University, India, in 1983.

Farid N. Najm received the B.E., degree (with distinction) in electrical engineering from the American University of Beirut (AUB) in 1983, and the M.S., and Ph.D., degrees in electrical and computer engineering from the University of Illinois at Urbana-Champaign in 1986 and 1989, respectively.

He worked with the General Swedish Electric Company (ASEA) in Västerås, Sweden, in 1982, and was a teaching assistant at AUB in 1983. He later worked as Electronics Engineer with AUB from 1983 to 1984 and held a visiting position with the University of Warwick, England, in 1984. While at the University of Illinois, 1985–1989, he was a research assistant with the Coordinated Science Laboratory, and worked for a year with the VLSI Design Lab., at Texas Instruments Inc., in Dallas, Texas. In July 1989, he joined Texas Instruments as Member of Technical Staff with the Semiconductor Process and Design Center. In August 1992, he became an Assistant Professor with the Electrical and Computer Engineering Department at the University of Illinois at Urbana-Champaign. He is currently Associate Professor Dr. Najm received the IEEE Transactions on CAD Best Paper Award in 1992 and the NSF Research Initiation Award in 1993. His research interests are in the general area of CAD tool development for VLSI circuits, including power estimation, low-power design, reliability prediction, synthesis of low-power and reliable VLSI, timing analysis, test generation, and circuit and timing simulation.



Hindawi

Submit your manuscripts at
<http://www.hindawi.com>

