Technology-dependent Transformations For Low-power Synthesis

Rajendran Panda, Advanced Design Technologies, Motorola, Inc., Austin, TX

Farid N. Najm, Coordinated Science Lab., University of Illinois at Urbana-Champaign, Urbana, IL

Abstract

We propose a methodology for applying gate-level logic transformations to optimize power in digital circuits. Statistically simulated[14] switching information, gate delays, signal arrival patterns, and signal probabilities are considered in reducing the switching activity-capacitance products. Power reduction up to 45.4% (average 12.4%) is achieved, with considerable improvements in area and delay, in pre-optimized benchmarks. Also the effect of transformations on the random pattern testability of the circuits is studied.

1. Introduction

Power reduction has become an important objective at every level of digital circuit design. The problem of power optimization has been considered at the system[12] and architecture[2] levels. In this paper, we address the problem at the logic synthesis level. Like the classical area and delay optimization approaches, power optimization also has been attempted through *technology-dependent* and *technology-independent* methods. The technology-independent methods have targeted algebraic *node simplification*[4, 11] and *node optimization*[5]. These methods use a 'power cost' of Boolean expression associated with a node, based on the fanout and input/output switching activities of the node.

The difficulty in optimizing power in a technology-independent manner is that :

(1) accurate delays, necessary for accurate power estimation, are unavailable until the network is mapped to a technology.

(2) the coarse granularity of network representation (fewer nodes, with large fan-in/fan-out) limits the accuracy and effectiveness of the cost function. Besides, subsequent restructuring could drastically affect earlier optimization efforts unpredictably.

In view of this, we propose our power optimization transformations on *technology-mapped* networks. Earlier proposed technology-dependent methods ([10] and [13]) ignored the effect of delays on power. In contrast, we give a detailed consideration of delay effects to effectively control glitch power, an elusive component of total power that has been neglected hitherto.

The paper is organized as follows. Section 2 discusses the power and timing models used in our work. The proposed transformations are presented in section 3. In section 4, experimental results are presented and a summary conclusion is given.

2. Power and Delay Models

The methodology described here targets static CMOS technology, and optimizes the *average* power dissipation resulting from the *dynamic* (*switching*) component, which is the dominant component. Thus, we attempt to minimize the expression :

$$P_{avg} = \frac{1}{2} V_{dd}^2 \sum_{gate_i} C(x_i) D(x_i) \tag{1}$$

where x_i refers to a node. $C(x_i)$ is the total capacitance at x_i, and $D(x_i)$ is the average switching activity of x_i.

As we should be able to consider the delay effects on switching, and hence on power, it is imperative that the switching measure is based on a *general* delay model. Unfortunately, much of the proposed power optimization methods have chosen to use a *zero-delay activity* measure, based on static probabilities.

Zero-delay activity of a gate refers to the transitions at its output in a hypothetical situation when the gate has no delays, and when all the inputs to the gate arrive simultaneously. It, thus, only provides a *lower bound* on the activity. The total activity, however, includes glitching transitions that occur due to the gate inputs arriving at different time instances within a clock period, and due to multiple input transitions. These are further modified by the propagation and inertial delays of the gates.

Our delay model makes use of user-specified values of *propagation* delays (t_{pdlh} and t_{pdhl}) for the gate modules, and rise and fall *transition* times (t_r and t_f) for their output nodes, all of which are assumed to refer to *unloaded* modules, with *instantaneous* (zero rise/fall times) transitions at the inputs. These values are then scaled to take into account the loading at the output, and the non-zero rise/fall times at the inputs, as described in [8].

We use *transition density*, proposed in [7], as the general switching measure. Through the statistical simulation-based estimation procedure, described in [14], we obtain accurately the zero-delay and glitching components of the switching, using the general (scalable) delay model described above.

While calculating the delays, using the above delay model, and the power, using eq. (1), a suitable estimate of the capacitances due to interconnects, C_{wire}, needs to be used. The example results presented here do not include this, though estimates based on suitable wire-load models of a technology can be used, and these values refined through an iterative process of synthesis and extraction.

DAC 97, Anaheim, California

(c) 1997 ACM 0-89791-920-3/97/06 ..$3.50

3. Technology-dependent Transformations

3.1. Transformations

A circuit synthesized for optimal area and/or delay, without regard to power, is likely to have sub-structures that are highly power-dissipative, resulting from situations such as :

[1] The Boolean function at the node causing excessive switching. The relation between (delay-decoupled) transition density at a node, y, and the Boolean functionality, $y = f(x_1, x_2, ..., x_n)$, is as given by the relation[7] :

$$D(y) = \sum_{i=1}^{n} P\left(\frac{\partial y}{\partial x_i}\right) D(x_i) \qquad (2)$$

In the above, $P\left(\frac{\partial y}{\partial x_i}\right)$ is the probability of the *Boolean difference* of y with respect to x_i. Further,

$$\frac{\partial y}{\partial x} \triangleq y|_{x=1} \oplus y|_{x=0} \qquad (3)$$

where $\oplus$ denotes the exclusive-or operation.

[2] Highly active nodes driving large fan-outs.

[3] The arrival patterns and probabilities of the inputs to gates are such as to cause excessive glitches at the output.

The freedom available in the selection of Boolean functions at the nodes, and hence the signals to drive the gates, presents opportunities to transform these highly power-dissipative sub-structures into normal ones. We deploy this freedom in restructuring, using *permissible function sets*[6] at the nodes, which correspond to the extent of Boolean *don't cares* available at a node. The full (external and implicit) set of don't cares at a node u_i is given as :

$$DC(u_i) = \bigwedge_{u_j \in IS(u_i)} \left(\overline{\frac{\partial u_j}{\partial u_i}} + DC(u_j)\right) \qquad (4)$$

where $IS(u_i)$ refers to the set of immediate successors of u_i, i.e. the outputs of gates driven by u_i.

Muroga, et. al,[6] proposed a *transduction* procedure, wherein several gate-level transformations are iteratively applied to improve the traditional objectives of area and delay of a network. We apply systematically a subset of these transformations, viz. *gate substitution*, *gate elimination*, *re-wiring* and *inverter addition*, so that the overall activity-capacitance products at the nodes are reduced.

Figure 1 illustrates re-wiring of inputs to gates driven by a gate (u_i). Supposing u_i is highly power dissipative, the re-wiring eases off some or all of the fan-out from u_i, and moves them to other nodes of less activity. Often, gate u_i can be rendered redundant by a sequence of re-wiring, as in Figure 1(a), resulting in larger power reduction. The possible elimination of several other gates in the fan-in cone of u_i makes such gate elimination attractive even when the switching rate of u_i is low. Gate substitution is a special case of gate elimination, where every fan-out of a node (u_i) is transferred to another node (u_j).

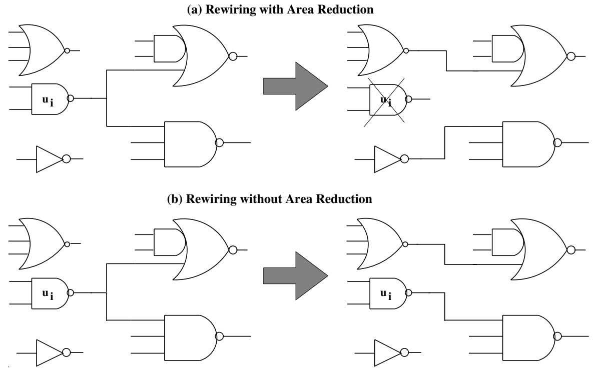


Figure 1. Example of Re-wiring.

The quality of optimization through wire/gate substitutions can be improved significantly by generating more choices for substitution. This is achieved by adding to the network nodes with new Boolean functions. By adding inverters to existing nodes, we create new functions for use in the transformation. In this case, the resulting power reduction is required to pay for the cost of the inverter also.

3.2. Cost Function

The global effects of a local transformation, such as rewiring, on area or delay of a circuit can be easily evaluated. However, this is not the case with power. We illustrate this with the help of Figure 2. Suppose u_i is a candidate input to u_j. As a result of a change at the input of u_j, the Boolean functions of u_j and of other gates (u_k) in its fan-out cone, may change. Based on eq.(2), and the modified signal behavior at u_j, the activity at every u_k in the fan-out region can change drastically, estimating which will require re-simulating the fan-out cone of u_j. This is computationally prohibitive, especially when several candidates (u_i) are to be evaluated for each transformation.

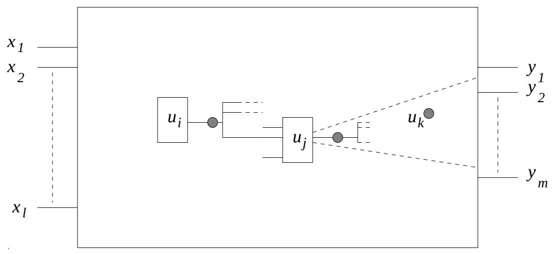


Figure 2. Local change affecting power globally.

We overcome this difficulty by :

(i) imposing certain restrictions on the Boolean functions that can be used for substitution, so that the change in power in the fan-out cone can be safely neglected. This aspect is dealt with in Section 3.4.

(ii) using an analytical expression as a cost function, taking into account the density, probability, glitch and arrival time information of the substitute, to evaluate (predict) the effect of the transformation on the total power.

In what follows, we explain the cost function for gate substitution, where gate u_i is substituted by gate u_j. The cost functions for other transformations can be similarly derived. We consider 4 components of the cost.

[i] Power change due to transfer of loads from u_i to u_j :

$$S_1(u_i, u_j) = (D(u_i) - D(u_j)) \sum_{v \in FO(u_i)} C_g(v) \qquad (5)$$

where $FO(u_i)$ are the gates in the fan-out of u_i and $C_g(v)$ is the gate (input) capacitance of gate v.

[ii] Power reduction due to elimination of u_i, and possibly several other gates in its fan-in cone :

$$S_2(u_i,u_j) = D(u_i)C_d(u_i) + \sum_{v \in FI(u_i)} D(v)C_g(u_i) + \sum_{v \in R} \Big(D(v)C_d(v) + \sum_{w \in FI(v)} D(w)C_g(v) \Big) \quad (6)$$

where R is the set of gates in the fan-in cone of u_i that are removable when u_i is removed, and $C_d(v)$ is the drain capacitance of gate v.

While S_1 and S_2 are easy to evaluate, other components, discussed below, pose difficulties.

[iii] The change in glitch power of fan-out gates of u_i, due to differences in the arrival times and signal probabilities of u_i and u_j. This is very expensive to determine exactly, as it would require simulation of the fan-out gates of u_i, for each candidate substitute u_j. Instead, we use a two-part heuristic measure, based on the arrival times and probabilities of u_i and u_j. Thus,

$$S_3(u_i, u_j) = S_{3,prob}(u_i, u_j) + S_{3,arrival}(u_i, u_j) \quad (7)$$

$S_{3,prob}$ and $S_{3,arrival}$ are explained in Section 3.3. The change in the arrival time of u_j, due to the newly added fan-out, is usually not drastic, and so its effect on the glitches in the successors of u_j can be ignored.

[iv] Power change at the transitive fan-out nodes of u_i :

$$S_4(u_i, u_j) = \sum_{v \in TFO(u_i)} C_{total}(v)\ \Delta D(v) \quad (8)$$

where C_{total} is the total capacitance and ΔD is the change in the switching activity due to the transformation. Determination of S_4 is prohibitively expensive as said before. Assuming the changes in glitch power of these nodes are negligible, S_4 can be safely ignored if it can be ensured that the zero-delay power at the transitive fan-out nodes either remains the same or decreases after the transformation. We, in fact, ensure this by constraining the set of permissible functions that can be used for the transformation. This is further explained in Section 3.4.

3.3. Glitch Control

3.3.1. Effect of Probabilities

Figure 3 shows two implementations of a function, and their input/output transitions. Implementation **(b)** generates excessive glitches as a result of the output NOR remaining sensitive longer to the transitions in signal a. The lower probability of signal $a'b'$ staying at logic ONE is undesirable in this case. We, hence, use a notion of *desired probability* of x w.r.t. u, denoted $\hat{P}(x, u)$, which is the probability of x that de-sensitizes a gate most to the transitions in u, given the probabilities of other inputs. The *probability cost* of x w.r.t. u is given as :

$$c_{prob}(x,u) = C_{total}(u) \times \Big(\sum_{\substack{v \in inputs(u) \\ v \neq x}} \left| P(x) - \hat{P}(x, v) \right| \times [D(v) - D_0(v)] \Big) \quad (9)$$

where $D(v) - D_0(v)$ is the amount of glitches at v. The deviation in an input's probability from its desired value (weighted by input glitches and output load) is used to penalize unfavorable substitutions. In the above, we use *signal probabilities* of the nodes, determined from the global functions of the nodes. As we store the Boolean functions in BDDs, the calculation of signal probabilities is easily done through a *scan* operation on the BDD.

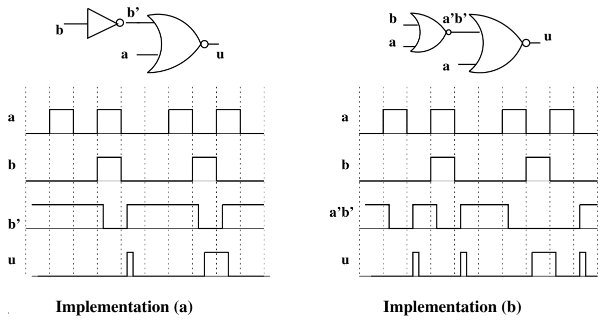


Figure 3. Effect of Probability on Glitching

The change in the glitches in the fan-out gates of u_i due to substitution by u_j is then :

$$S_{3,prob}(u_i, u_j) = \sum_{u \in FO(u_i)} (c_{prob}(u_i, u) - c_{prob}(u_j, u)) \quad (10)$$

3.3.2. Effect of Arrival Times

A glitch can occur at the output of a gate if two inputs transition non-simultaneously. To control glitches, we penalize substituting inputs with signals that switch outside the switching windows of other inputs to the gate.

Definition. Activity window *of a signal arriving at the input of a gate is the time interval 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.*

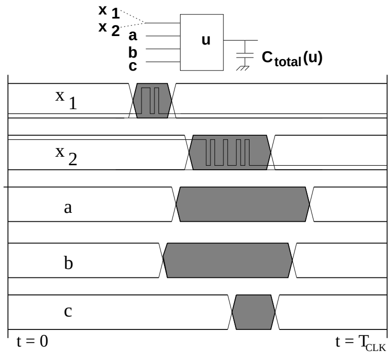


Figure 4. Considering Non-simultaneity in Switching of Inputs

Definition. Critical windows *of a signal, x, with respect to a signal, y, are those segments of the activity window of x that are not overlapping with the activity window of y.*

Definition. Non-simultaneity factor *of x w.r.t. y is the ratio of the total width of the critical windows of x, w.r.t. y, to the width of the activity window of x.*

Given that x is an input to a gate u, the contribution of x toward output glitches of u is taken as the amount of glitches in x, weighted by the non-simultaneity factor of x w.r.t. the other inputs of u. Thus, a cost measure based on arrival times is heuristically taken as:

$$c_{arrival}(x, u) = \alpha(x)\,[D(x) - D_0(x)]\,C_{total}(u) \qquad (11)$$

where $C_{total}(u)$ is the total capacitance at the output of u, and $D(x)$ and $D_0(x)$ are the total and zero-delay transition densities of x. α is the average non-simultaneity factor of x w.r.t. other inputs of u. Thus, $S_{3,arrival}(u_i, u_j)$ is given as :

$$S_{3,arrival}(u_i, u_j) \;=\; \sum_{u \in FO(u_i)} (c_{arrival}(u_i, u) - c_{arrival}(u_j, u)) \qquad (12)$$

We illustrate the above approach through Fig. 4. Suppose there are two possible inputs, x_1 and x_2, for a substitution, and the activity windows of other inputs, a, b, and c are as shown. It is seen that, though x_2 has more glitches than x_1, it can be expected to cause less glitching at the output as its activity window overlaps well with those of other signals.

Note that, in the above, we are not considering the fact that the gate may be de-sensitized to an input glitch. Actually, this is accounted for in our consideration of the probability effects, in Section 3.3.1.

In the above, the activity windows of signals are computed from the statistical minimum and maximum arrival times of signals, as recorded during the power estimation simulation run. The statistical timing provides more realistic activity windows than static-timed min. and max. arrival times, as the false paths are eliminated, and moreover the timing of most active paths that matter for power are considered.

3.4. Permissible Functions For Power

The maximum set of permissible functions[6] provide the most freedom in carrying out a logic transformation. However, this allows the Boolean functions of nodes in the transformed network to change arbitrarily, making the S_4 term above very expensive to determine. We identify two subsets of the maximum sets of permissible functions that eliminate the need to compute S_4.

3.4.1. Locally-derived Set of Permissible Functions

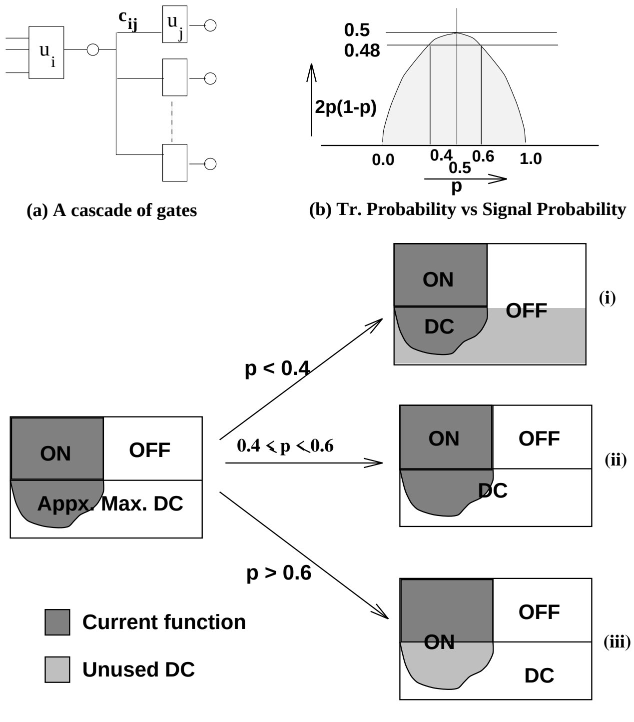
Consider a cascade of gates, shown in Figure 5(a). The LSPF of the node u_i is derived under the constraint that the Boolean functions of its immediate successors (u_j's) should not change when the current function at u_i is substituted with a permissible function. This essentially amounts to *not* considering the don't cares of the immediate successors, and so a smaller set of permissible functions is obtained. However, since Boolean functions of immediate successors, and hence that of other successors do not change, the zero-delay power in the fan-out cone of u_i does not change when a transformation is applied to u_i using this set. As the change in the glitch power at the successor nodes (but not immediate successors) is negligible, the contribution of S_4 can be ignored altogether. The change in both the zero-delay and glitch power of immediate successors are considered by cost terms, S_1 and S_3.

3.4.2. Filtered Set of Permissible Functions

If the values of a signal in two consecutive clock cycles are not correlated, then the transition probability (i.e. the transition density minus glitches) of the signal can be given by the relation :

$$D_0 \;=\; 2p(1-p) \qquad (13)$$

where p is the probability of the signal being HIGH. This relation is shown graphically in Fig. 5(b). Note that the switching activity is very high at probabilities close to 0.5, with no significant variation in the range 0.4 – 0.6. When considering the effect of a transformation on the transition densities of successor nodes, but not immediate successors, the change in glitches are negligible, as said before. Thus, the transformations need be concerned primarily with the zero-delay transitions of these nodes. As a result, we can use the above probability-density relation to guide the transformations.



(c) Filtering of don't cares

Figure 5. Permissible Functions for Power

The idea is to let the Boolean functions of successors of u_i change such that their probabilities move further away from 0.5, as a result of a transformation at u_i. This is achieved by trimming the don't cares of the successors (u_j) as shown in Figure 5(c), before propagating them for the calculation of the don't cares of u_i.

In Figure 5, a set of permissible functions is represented as a unique partitioning of the Boolean space into three disjoint sets - ON, OFF and DC. The shaded set covering the entire ON set and a part of the DC set represents the current Boolean function. Suppose the signal probability of an immediate successor (u_j) is more than 0.6, then the current function at u_j is specified as the ON-set of u_j, so that any new function at u_j, as a result of a transformation at u_i, can only have higher signal probability than the current function. In calculating the don't care of u_i, the don't cares of u_j filtered in the above manner is used. Likewise, the OFF-set is expanded in the case of probability being less than 0.4. When the probability is in the range 0.4 - 0.6, no trimming is done, so that larger don't care sets are available. The better optimality obtainable with larger don't cares more than outweigh the risk of slightly increasing the power at a node, should its probability change from 0.4 to 0.5.

[5] originally proposed a similar procedure to filter the don't cares, but the procedure described above usually re-

TABLE 1. Performance Results of Proposed Optimization

Circuit	Size			Reference	Glitch Pwr.	Percentage Reduction in					Run Time
Name	#i	#o	#nodes	Power	(%Total)	Power	Delay	Area	ZDPwr.	Glitch Pwr.	cpu sec.
z4ml(A)	7	4	27	42.4	8.1	10.94	2.90	5.10	10.7	14.0	9.3
z4ml(D)			44	74.9	11.3	19.70	17.06	9.40	18.5	29.4	12.4
cm163(A)	16	5	33	68.4	18.2	3.44	18.99	3.45	1.9	10.3	7.9
cm163(D)			42	78.6	18.4	32.08	40.43	14.77	23.3	71.3	18.6
apex7(A)	49	37	133	227.5	8.3	1.23	-1.56	0.86	1.2	1.7	289.1
apex7(D)			184	335.8	8.2	0.59	0.00	-0.16	0.7	-0.2	297.4
ex2(A)	85	66	250	387.3	12.5	11.92	1.67	3.73	9.4	29.3	135.7
ex2(D)			250	387.3	12.5	11.92	1.67	3.73	9.4	29.3	169.5
x4(A)	94	71	225	314.7	12.7	1.99	0.00	0.40	1.4	6.0	213.6
x4(D)			261	408.1	11.8	0.86	8.08	-0.22	0.2	6.1	246.7
x1(A)	51	35	195	328.9	10.4	2.93	7.00	-0.29	3.1	1.1	566.3
x1(D)			270	552.2	8.9	2.18	0.00	1.51	2.0	4.1	614.6
i3(A)	132	6	160	636.1	5.7	30.66	0.00	28.57	28.7	63.4	52.1
i3(D)			312	915.1	8.7	45.44	0.00	32.14	42.1	80.6	68.5
alu2(A)	10	6	196	395.4	29.1	12.93	34.72	1.65	12.2	14.7	747.9
alu2(D)			278	506.0	25.3	15.73	31.71	7.91	14.7	18.9	870.7
C432(A)	36	7	142	78.6	29.2	11.70	20.28	5.44	6.1	25.2	918.8
C432(D)			187	92.6	20.9	25.89	25.36	20.97	19.8	49.0	913.2
vda(A)	17	39	376	568.1	29.6	0.48	-1.88	-0.16	1.8	-2.6	1631.6
vda(D)			469	596.3	20.1	4.97	3.13	1.28	4.1	8.5	1201.3
i9(A)	88	63	297	346.8	31.0	10.83	28.68	0.62	3.5	27.1	770.2
i9(D)			374	784.8	23.3	7.60	26.86	0.72	3.8	19.9	745.9
alu4(A)	14	8	372	746.3	32.0	5.44	12.84	1.08	5.3	5.8	4149.3
alu4(D)			511	941.5	31.5	11.14	19.41	1.43	6.4	21.4	5423.4
x3(A)	135	99	458	796.1	15.8	15.87	3.03	5.81	13.1	30.4	2019.8
x3(D)			657	1290.0	15.5	16.58	-0.82	7.03	14.1	30.1	3063.5
t481(A)	16	1	480	484.0	19.7	18.63	10.96	6.87	18.4	19.4	7431.4
t481(D)			611	582.2	17.4	21.75	19.39	9.92	19.9	30.4	2286.5
i8(A)	133	81	527	700.6	19.5	4.17	0.41	0.61	0.8	18.0	3395.5
i8(D)			740	1033.7	17.3	13.27	-0.26	4.82	14.5	7.7	4536.4

sults in larger don't care sets. Moreover, since only a part of the don't cares of successors is rejected, the filtered maximum sets are considerably larger than the LSPF discussed in the previous section when the number of fan-outs is small, but only slightly larger when the fan-out is large.

3.5. Transformation Procedure

The transformations are attempted at every node in an iterative manner, starting from the primary outputs, and proceeding toward the primary inputs. A node is optimized only after all its successors have been attempted, to minimize the need to recalculate the don't cares, which are otherwise expensive to do. This strategy, combined with the trimming of the permissible functions as explained in Section 3.4, eliminates the need to re-simulate the circuit often for power estimation.

Before each iteration, the capacitances, global Boolean functions, signal arrival times and probabilities are calculated for all the nodes. The transition densities and glitches at the nodes are estimated through statistical simulation[14], and the statistical minimum and maximum arrival times are stored.

As a gate is visited, the set of permissible functions of the gate's output, and also of its input connections are calculated, and candidate substitutes for them are then identified from the nodes in the network. The candidate with the best power cost is then used to substitute the gate or its input, provided that no feed-back is formed and that the speed of the circuit is not degraded. The latter condition is checked using the static timing information calculated at the beginning of the iteration.

The procedure is repeated until either no more useful transformation is identified, or no more reduction in power is achieved.

4. Results

Table 1 shows the performance of the proposed technique applied to a subset of pre-optimized MCNC benchmarks. The pre-optimization consisted of an initial node minimization using SIS[1], followed by a minimal-activity decomposition described in [9], and later a mapping to a library(*lib2.genlib*). Except $C432$, all circuits were minimized using full set of don't cares. The mapping was done for both minimal area and minimal delay, and thus two sets of reference circuits (marked (A) or (D) in column 1) were obtained.

The proposed optimization was then applied using filtered set of permissible functions (section 3.4.2). The power estimation was done with random input vectors having signal probability of 0.5, and 0.5 transitions per clock. Since the transformations can shift loads from internal nodes to primary inputs, and vice-versa, for a fair comparison, the total power values reported in the table include also the power required to switch the primary inputs, although this power will be dissipated outside the circuit being optimized.

An average power reduction of 15.3% (9.5%) in delay-mapped (area-mapped) circuits, with significant improvements in the case of decoder($C432$), adder (*z4ml*) and alu(*alu2*) circuits, was achievable. The area of the circuits further decreased by 7.7% (4.3%) on average as a result of eliminating gates, and the speed improved by 12.8% (9.2%). This latter part was, in fact, achieved by respecting the delay constraints during the transformations.

The high glitch contents (column 6) justify our detailed treatment of glitches. The effectiveness of shifting of fanouts, elimination of gates, and glitch control, are brought out in columns 9 – 11. A comparison of columns 11 and 9 shows the effectiveness of the glitching considerations (cost term S_3). In almost all examples, the reduction in glitches is much higher than the reduction in area. For instance, *example2(D)* recorded a 29.3% reduction in glitch power against only 3.7% reduction in area, implying that the bulk of reduction in glitch power came from the reduction in glitching activity, rather than reduction in capacitances (as indicated by the reduction in area). In some circuits with very regular structure (e.g. *i3* and *C432*), very high reduction of glitches was possible.

Similarly, the larger reduction in the zero-delay power (column 10) in comparison to area reduction is attributed to the shifting of fan-out loads from high-activity nodes to low-activity nodes, and to the transformation of high-activity Boolean functions into low-activity ones. In few examples (e.g. *apex7* and *vda*) the transformations had a slight negative effect on glitches, possibly due to the inexactness of our heuristic measure for glitch control.

TABLE 2. Effect of Optimization on Testability

Circuit Name	Before optimization				After optimization			
	Total Faults	Redun. Faults	RT-covered Faults	%	Total Faults	Redun. Faults	RT-covered Faults	%
cm163(A)	190	4	172	92.47	186	5	169	93.37
cm163(D)	184	8	158	89.77	174	2	164	95.35
apex7(A)	724	1	699	96.68	725	1	698	96.41
apex7(D)	723	1	686	95.01	723	1	676	93.63
ex2(A)	956	0	912	95.40	940	6	886	94.86
ex2(D)	956	0	881	92.15	940	6	901	96.47
x4(A)	1121	0	1103	98.39	1112	0	1077	96.85
x4(D)	1235	0	1203	97.41	1234	0	1204	97.57
x1(A)	1025	0	861	84.00	1023	6	837	82.30
x1(D)	957	0	801	83.70	949	0	797	83.98
i3(A)	526	0	406	77.19	334	0	231	69.16
i3(D)	574	0	393	68.47	334	0	260	77.84
alu2(A)	1009	11	980	98.20	979	20	941	98.12
alu2(D)	1013	10	974	97.11	970	12	937	97.81
vda(A)	1205	0	1077	89.38	1205	2	1037	86.20
vda(D)	1214	0	1077	88.71	1212	2	1076	88.93
i9(A)	1823	0	1819	99.78	1815	3	1806	99.67
i9(D)	1724	0	1716	99.54	1716	0	1711	99.71
x3(A)	2622	52	2405	93.58	2506	17	2333	93.73
x3(D)	2670	35	2506	95.10	2506	13	2342	93.94
t481(A)	2061	209	1096	59.18	1934	168	1036	58.66
t481(D)	2080	203	1118	59.56	1934	147	1050	58.76
i8(A)	3409	21	3347	98.79	3378	21	3303	98.39
i8(D)	3390	19	3335	98.93	3262	20	3196	98.58

As the transformations alter the node probabilities, which relate to the testability measures, viz. controllability and observability[3], it is of interest to study the effect of the transformations on the random pattern testability of the circuit. A comparison of the number of total, and redundant faults, and the random test coverage of testable faults is given in Table 2.

We observed only very marginal decrease in the test coverage in about 60% of the cases. But in many cases, (e.g. *i3(D), cm163a(D)*), the coverage, in fact, improved appreciably. The number of redundant faults also decreased in most cases due to elimination of more redundancies. This is highly noticeable in the case of *t481*. There are, however, few cases where the optimization introduced new redundant faults. This is due to the transformations rendering some connections redundant, but which redundancies are not being subsequently exposed due to the use of subsets of the maximum don't cares for optimization.

In conclusion, the proposed method provides a solution for power optimization, as an 'add-on' to an existing design flow centered around traditional area/delay optimization. Based on technology-dependent restructuring, it provides much accuracy, and better confidence level for the results achieved, which the designers very much desire.

REFERENCES

[1] Brayton, et. al. Multilevel logic synthesis. *Proc. of the IEEE*, 1990.

[2] Chandrakasan, et. al. Optimizing power using transformations. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 1995.

[3] Goldstein. Controllability/observability analysis of digital circuits. *IEEE Trans. on Circuits and Systems*, CAS-26(9):685–693, 1979.

[4] Iman, et. al. Logic extraction and factorization for low power. *DAC-95*.

[5] Iman, et. al. Multi-level network optimization for low power. *ICCAD-94*.

[6] Muroga, et. al. The transduction method – design of logic networks based on permissible functions. *IEEE Trans. on Computers*, 1989.

[7] Najm. Transition Density : A new measure of activity in digital circuits. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 1993.

[8] Panda. *Synthesis Techniques for VLSI Low-Power Circuits.* PhD thesis, University of Illinois at Urbana-Champaign, 1996.

[9] Panda, et. al. Technology decomposition for low-power synthesis. *CICC*, pages 627–630, 1995.

[10] Rohfleisch, et. al. Reducing power dissipation after technology mapping by structural transformations. *DAC-96*.

[11] Roy, et. al. SYCLOP: Synthesis of CMOS logic for low-power applications.

[12] Tiwari, et. al. Compilation techniques for low energy: An overview. *Symposium on Low Power Electronics*.

[13] Ueda, et. al. Low power design and its testability. *Proc. Fourth Asian Test Symposium, India*, 1995.

[14] Xakellis, et. al. Statistical estimation of the switching activity in digital circuits. *DAC*, pages 728–733, 1994.