Implication-Based Gate-level Synthesis for Low-Power

Topics: Technology-Independent, Combinational Logic Synthesis and Optimization

Abstract

The paper presents a new logic optimization method of multi-level combinational CMOS circuits, which minimizes area and power. Present methods to reduce power on logic circuits apply functional methods like logic factorization on the Boolean networks. The method described here uses Boolean transformations that exploit implications at the gate-level based on both controllability and observability relationships. These transformations are novel and have not been explored for low power synthesis. It is observed that the average switching activity of lines in a circuit is the primary cause for power consumption. Restructuring the circuit based on the switching activity can significantly reduce the average power consumption in a circuit. New transformations are presented here, along with a new synthesis method to optimize both area and power. Our methods can also be applied as a pre-processing step before technology mapping. Experimental results show circuits synthesized by our method consume less power with a lesser area than those synthesized by SIS and HANNIBAL.

1 Introduction

With the increasing emphasis of portability and mobility, new constraints on VLSI design have evolved. Power consumption, therefore, becomes an important issue, battery size increasing with power consumption, given the same amount of time for which the devices are operated. Moreover, power consumption limits, as well, the number of gates that can be put on VLSI chips because of factors like packaging cost and stability. Hence, power consumption reduction has gained much importance in today's digital circuits. Power consumption can be reduced at various stages of the design process; power optimization at the logic design stage has considerably less investment cost since it does not effect the IC fabrication process.

This paper presents a logic synthesis tool which operates on unmapped, multi-level combinational circuits. The goals of our synthesis process are optimization of both area, and power consumption. The structural transformations presented in this paper are, therefore, tailored for both power and area. Present tools which optimize combinational circuits for low power consumption concern themselves with circuit representations like Boolean networks, factored forms, etc. The procedure described here possesses the added advantage of working with a level which is much closer to the physical design of the circuits. Also, a power estimation technique is used which is applicable on the gate-level design. Such a cost function, combined with the estimate of the area, is used during the synthesis procedure. Transformations presented here can be classified under two classes.

- Transformations for area optimization.
- Transformations for reduction of power consumption.

Transformations of the first type can be found in [2, 3]. New Boolean transformations for reducing power consumption are introduced. A synthesis method is also proposed which optimizes area as well as power. However, these techniques have potential applications in delay optimization as well.

The paper is organized as follows. The next Section provides a short review of those previous logic design approches to reduce power consumption. Section 3 introduces our power estimation method for a gate-level design. Structural transformations used in our methodology are set out in Section 4. New structural transformations are also introduced, aimed at reducing the average switching activity in the circuit. A description of the proposed algorithm and the heuristics is given in Section 5. Experimental results, where our circuits are compared against those synthesized by SIS and HANNIBAL, are analyzed in Section 6. Section 7 offers the conclusions.

2 Previous Work

The context here has focused on technology-independent power optimization for multi-level circuits. All of these methods assume a *Boolean Network* representation of the circuit. All such synthesis methods contain two essential components – power estimation, and optimization guided by the cost function.

Power Estimation Model: In a digital CMOS circuit, dynamic power consumption is the dominant source of power consumption. The amount of energy dissipated in a CMOS circuit [13, 18] is approximately equal to the energy required to charge or discharge the load capacitances at various nodes in the Boolean network. If the circuit is controlled by a global clock, then the estimated average power dissipated is given by

$$P = \sum_{\forall \text{nodes } i} (0.5 \times C_{l_i} \times V_{dd}^2) \times E_i / T_c \tag{1}$$

where

E_i is the *switching activity* at the node i.

C_{l_i} is the load capacitance at the output of the node.

T_c is the global clock period.

Because the values of V_{dd} and T_c are constant, the power consumed will be proportional to the switching activity and the load capacitance. Hence, for optimization, the following cost function can be used:

$$P = \sum_{\forall \text{nodes } i} p_i(1 - p_i) \times C_{l_i} \tag{2}$$

where

p_i is the signal probability of node i and

C_{l_i} is the load capacitance at node i

The product $p_i(1 - p_i)$ is called the *switching activity* (denoted as transition density in [13]), E_i of node i. The power is directly proportional to the value of the total switched capacitance in the circuit. Throughout the remainder of this paper, the term *switched capacitance* will be used instead of the total power. The switched capacitance will be used in the cost function during circuit optimization. It should be noted that both the power consumed due to short circuit power dissipation, as well as power dissipation due to glitching are not taken into account in this estimation.

In all previous methods, the switching activity is determined either using BDDs [13], or by random pattern simulation using a large number of vectors. The switched capacitance value, as estimated in [18] and [14], is discussed here.

Definition 2.1: Given a node n_i and fanin node n_j, factored load $FL(n_j, n_i)$ is defined as the number of times the variable n_j is used (in positive or negative form) in the factored expression of node n_i [19]. For example, if a node d is expressed as $d = a(b+c)+bc$, we have $FL(a, d) = 1$, $FL(b, d) = 2$ and $FL(c, d) = 2$. □

Given a Boolean network, [18] calculates the switched capacitance at each node, n_i. The expression of the power being consumed at a node n_i [18] is given as

$$P_1(n_i) = (c \times \ m \times \sum_{n_j \in cubes(n_i)} E_j) + (\sum_{n_k \in fanins(n_i)} E_k) \tag{3}$$

Here, c is the fanout load of node n_i, m is the number of product terms in the cover, E_j is the switching activity of the jth cube in the cover and E_k is the switching activity of the kth local input to n_i.

In [19] the power is calculated in a similar way, but uses a more accurate estimation of the load value at the node. The expression used in [19] for a node n_i is given as

$$P_2(n_i) = \sum_{n_j \in fanins(n_i)} FL(n_j, n_i) \times \ E_j + \sum_{n_k \in fanouts(n_i)} FL(n_i, n_k) \times \ E_i \tag{4}$$

Both the above cases use the estimation of power consumption as the cost function during synthesis. Other methods to estimate power can be found in [16, 15, 17].

Optimization Procedure: Optimization methods shown in [18, 19] are based on functional methods using algebraic factorization. Logic Factorization [19] was guided by the 'power cost' of kernels extracted during optimization, rather than using area cost of the kernels, as done in SIS[8]. Computation of kernel sets for all nodes is followed by extracting the set of kernel intersections. These kernels are considered for optimization, based on their 'power cost'. Optimization using don't-cares [14, 18] is done by examining the power cost of compatible don't-cares in a Boolean network. Two-level logic optimization was also performed, based on calculation of appropriate prime implicants [20], to be used for obtaining better power performance.

3 Proposed Power Estimation

To our knowledge, the proposed method is the first procedure at the gate-level for power optimization. The following formulates a power estimation method which treats each primitive gate as a node in the network with *zero-delay*. Our goal here is to provide relative estimates for guiding the optimization. Therefore, the model provides a useful basis for estimating dynamic power consumption.

Given a multi-level gate-description of a circuit, the proposed power estimation method, first estimates the signal probabilities at each line in the circuit. Signal probabilities are used to calculate the switching activity at each line. The *switched capacitance* in the circuit is calculated based on equation (2). The signal probabilities at each line are determined by random-pattern simulation. Random-pattern simulation can also take into account particular signal probabilities assigned at the primary inputs of the gate. Load capacitance is derived from the nature of the gate, its fanins and its fanouts.

Capacitance Calculation:

The load capacitance calculation is related to the implementation of the gate as a standard CMOS cell. Load capacitance can be adjusted with any other implementation that will be used for technology mapping. The total load capacitance at the output of a CMOS gate [12] is the sum of:

1. gate capacitance (of other inputs connected to the output of the gate) - C_g.

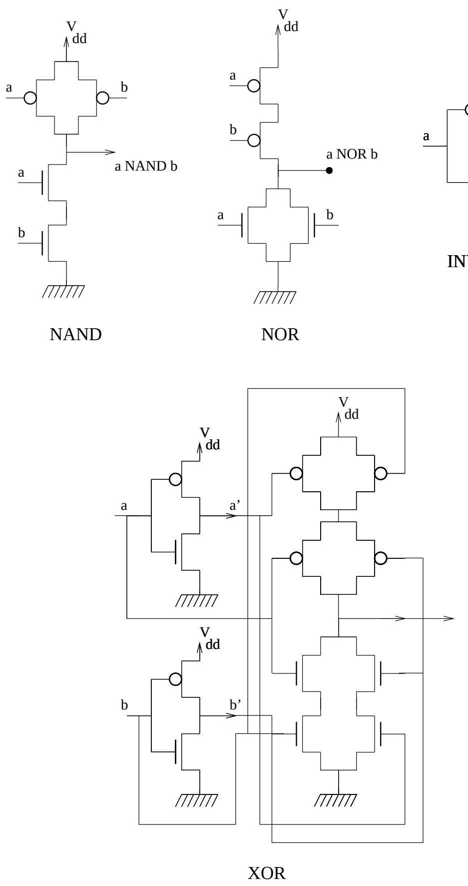

Figure 1: Gate-level implementations some gates considered during power estimation.

2. diffusion capacitance (of the drain regions connected to the output) - C_{ds}.

3. routing capacitance.

In the gate-level description of a circuit, only the first two components can be estimated. In our procedure we estimate the C_g and C_{ds}, assuming standard CMOS implementations. Some of the implementations considered during the synthesis process is shown in Figure 1. This might be an assumption regarding the implementation technology but the values of load capacitance can be calculated similarly for other implementations, as well. Some of the implementations used in our tool are shown in Figure 1. Given a NAND gate with f_{in} fanins and f_{out} fanouts, the value of the capacitance, C_l can be estimated as follows.

$$C_l = (f_{in} + 1) \times C_{ds} + 2 \times f_{out} \times C_g \tag{5}$$

It is assumed that the values of the two capacitances are the same for both *n type* and *p type* transistors.

The power estimated using the above model is used in the cost function during synthesis. It has been observed that our power calculation has trends similar to those of SIS.

4 Structural Transformations

The proposed synthesis is based on structural logic transformations to the gate-level design. This section defines the framework of the optimization process, reviews the existing structural logic transformations and proposes new transformations aimed at reducing dynamic power consumption. Transformations use certain controllability and observability implications among lines in the circuit. Because these transformations can cover a wide range of Boolean logic transformations, they are denoted as *Implication-based Boolean*, IB transformations.

4.1 Preliminaries

Assume a combinational circuit, C, is given with n primary inputs and m primary outputs. The combinational circuit consists of primary gates like AND, OR, NOT, NAND, NOR and EX-OR gates. All gates in the circuit have a unique label and their output signals, y_i, realize Boolean functions $y_i(\vec{x})$: $B_2^n \rightarrow B_2$ with $B_2 = \{0{,}1\}$, where the variable $x_1, ..x_n$ corresponds to the primary input signals of the circuit C. Following the usual representation of a combinational circuit as a directed acyclic graph (DAG), a signal f lies in the *transitive fanout* of y if and only if there exists a directed path from y to f. Furthermore, we assume that there are no *external* don't cares; the function of the combinational network $C(\vec{x})$:$B_2^n \rightarrow B_2^m$ with $B_2 = \{0{,}1\}$ is completely specified.

Two combinational networks, C and C', are called *equivalent*, denoted $C = C'$, if they implement the same function $C(\vec{x})$:$B_2^n \rightarrow B_2^m$ with $B_2 = \{0{,}1\}$. They are called *structurally identical* if there exists a one-to-one mapping between C and C', such that for every node y_i in C, there is a y_i' in C' and vice-versa, where y_i and y_i' implement the same function. A transformation from a network C to another equivalent network C' is possible by replacing the node y in C by a node y' with an equivalent function. A function at node y can also be replaced by some non-equivalent function y' if this does not change the function $C(\vec{x})$:$B_2^n \rightarrow B_2^m$ of the logic network as a whole. Such functions are called *permissible functions* [6].

4.2 Gate-level IB Transformations - Review

IB transformations, and their derivatives, have been applied on gate-level circuits either for minimizing area in technology-independent gate designs [6, 4, 5, 2], minimizing area after technology mapping [7], or even to enhance testability [3]. IB transformations were chosen to drive our synthesis process because of two principle reasons :

- They are simple and the cost of each transformation can be estimated with low time overhead.
- These transformations preserve the functionality of the circuit and therefore, there is no need to verify for equivalence after transformation, unlike [5].
- The expansions based on IB transformations can cover a wide variety of logic transformations [2].

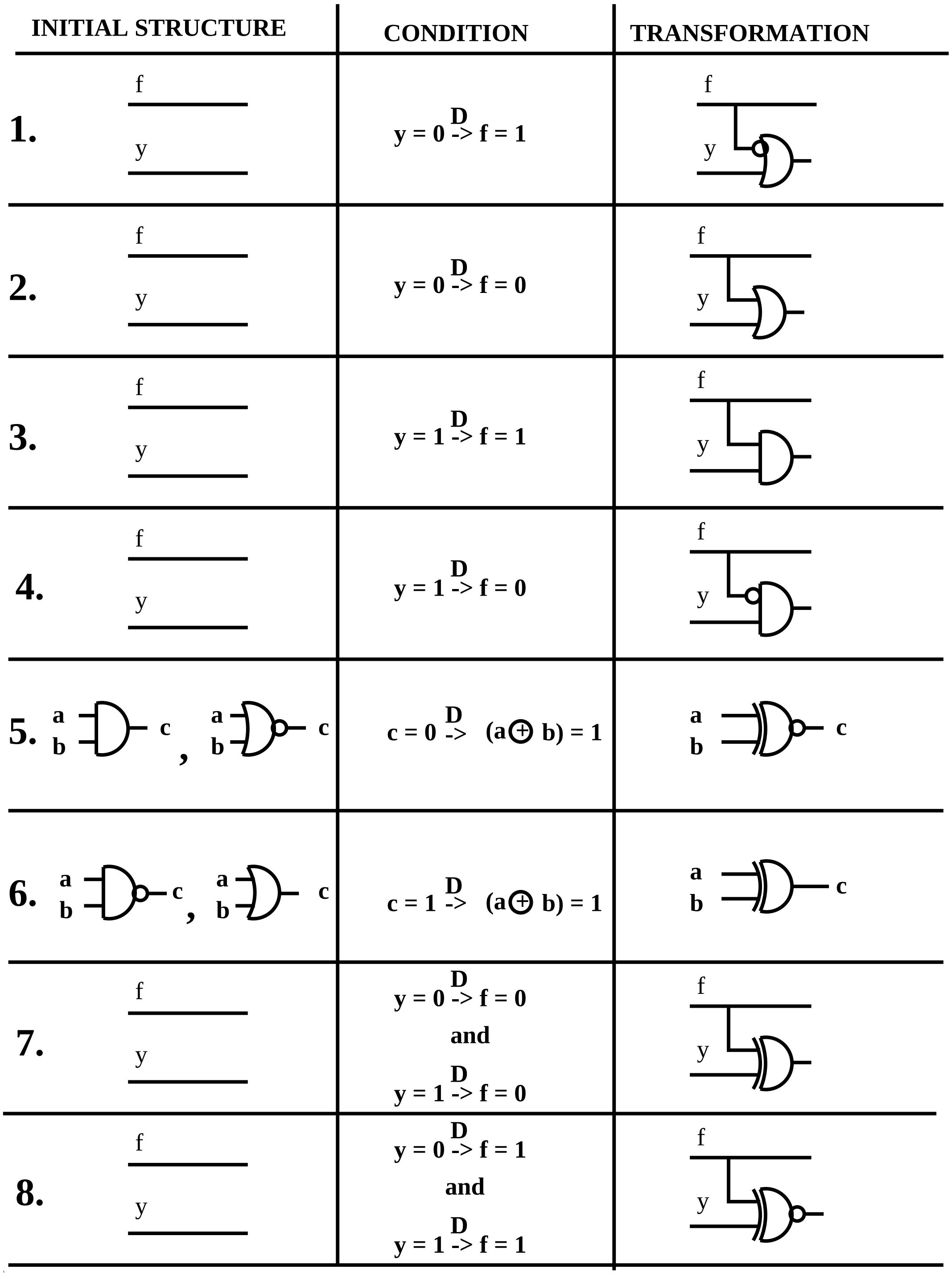

Figure 2: Structural Transformations.

Three types of implications involved to make these transformations are defined:

Definition 4.1: Given two lines in a circuit, y and f, if the line f takes value V, ($V \in \{0,1\}$), whenever, y has U, ($U \in \{0,1\}$), then there is a controllability implication between $y{=}U$ and $f{=}V$, denoted by $y = U \rightarrow f = V$.

Definition 4.2: In a circuit having y and f, if the line f is V, ($V \in \{0,1\}$), whenever the value U, ($U \in \{0,1\}$), at y is observed at any of the primary outputs, then there is an observability implication between $y{=}U$ and $f{=}V$, denoted by $y = U \overset{O}{\rightarrow} f = V$.

Definition 4.3: For two lines, y and f, there exists a testability implication (referred as D-implication in [2, 3]), denoted by $y = U \overset{D}{\rightarrow} f = V$ if

(a) $y = U \rightarrow f = V$ *or*

(b) $y = U \overset{O}{\rightarrow} f = V$.

The transformations use *Shannon's expansion* to make gate-level connections, based on the above implications. Transformations applied using *controllability implications* exploit the controllability don't cares in the network, whereas those with *observability implications* exploit the observability don't cares [9]. For an arbitrary Boolean function, $f(\vec{x})$:$B_2^n \rightarrow B_2$ and $y(\vec{x})$:$B_2^n \rightarrow B_2$ with $B_2 = \{0,1\}$, the following equation holds:

$$y(\vec{x}) = f(\vec{x}).y(\vec{x}) \mid_{f(\vec{x})=1} + \bar{f}(\vec{x}).y(\vec{x}) \mid_{f(\vec{x})=0} \tag{6}$$

Here, we use the following notation, $y = f.y \mid_1 + \bar{f}.y \mid_0$ to represent the above equation. The methodology suggested can be viewed as a two-step process :

1. Transformation: $y = f.y + \bar{f}.y$

2. Reduction: Redundancy elimination.

The first step is equivalent to performing a *division* [9]. Combining it with redundancy elimination results in an expansion that can also be seen as special *transductions* (transformation and reduction) [6]. The method to identify divisors is based on *indirect implications* [1]. If a value assignment at a node y allows us to imply a unique value assignment at node f, then equation 6 can be simplified. The node f must not be in the transitive fanout of y, to ensure that the circuit remains combinational after the transformation.

Different types of transformations at the first step can result in different kinds of logic operations. Eight types of such transformations, *Transformations 1 - 8*, have been presented in Figure2 These transformations were proposed in [2, 4, 5, 3] and are being used in the proposed synthesis scheme. These transformations alter the structure of the circuit without changing the functionality of the circuit. The first four transformations introduce AND, OR and NOT gates in the circuit, whereas the next four transformations introduce EX-OR or EX-NOR gates in the circuit. Each transformation is followed by

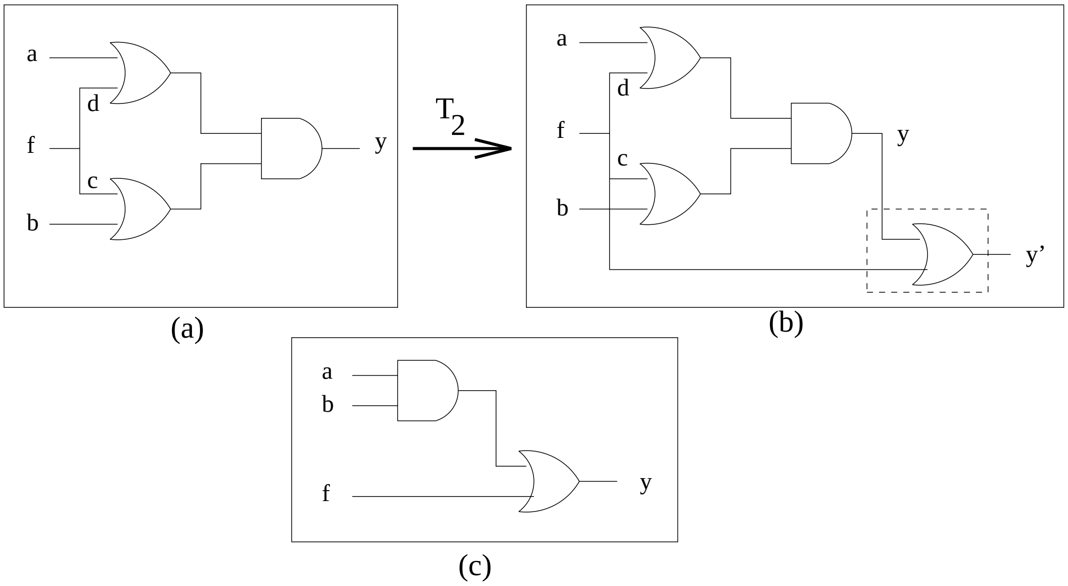

Figure 3: An example of Transformation 2 followed by redundancy elimination.

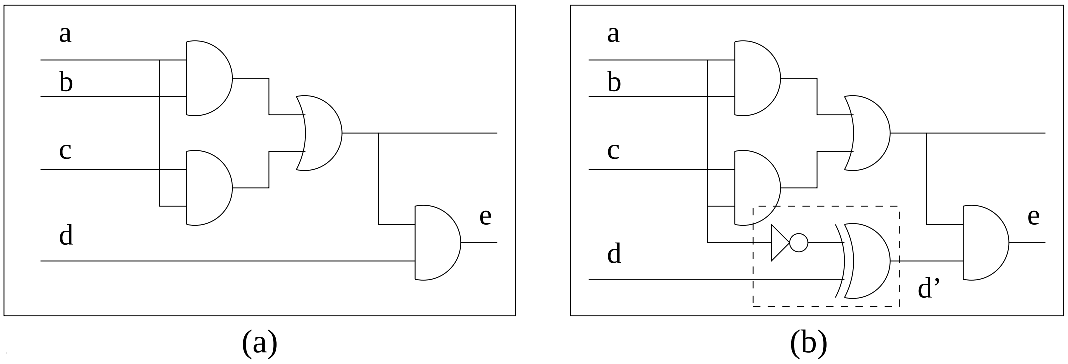

Figure 4: An example of Transformation 8.

redundancy elimination, which may result in a circuit of reduced cost. In the context of power optimization, these transformations can be viewed as a two step process, where the addition of some active area (transformation) is followed by removal of active area in another part of the circuit (redundancy removal).

Example 4.1: Take the circuit shown in Fig. 3(a) as an example. It can be observed that $y = 0 \Rightarrow f = 0$ and transformation T_2 can be applied on the circuit. The resultant circuit is shown in Fig. 3(b). ATPG processing on the resultant circuit would determine lines 'c' and 'd' to exist as redundant s-a-1 lines and hence, can be eliminated. The final circuit after redundancy elimination is shown in Fig. 3(c).$\Box$.

Example 4.2: The circuit shown in Fig. 4 illustrates an example for *Transformation 8*. The conditions required to test d stuck-at 0 as well as d stuck-at 1 include the assignment ($a = 1$). Thus, from the definition of D-implications, we get $d = 0 \xrightarrow{D} a = 1$ and $d = 1 \xrightarrow{D} a = 1$ as true. The resultant circuit is shown in Fig.4(b).$\Box$

Conditions to be used in the transformations are determined by Recursive Learning. This is accomplished by two routines: *make_all_implications()* and *fault_propagation_learning()*, as given in [1]. Any other method to identify precise implications can be used to make efficient transformations. The implication-based synthesis procedure only looks at *indirect implications* as the promising candidates for optimization.

4.3 New Transformations for Reducing Power Consumption

This section includes proposed transformations which tend to reduce the *switched capacitance* in the overall circuit. This may, sometimes, increase the area of the circuit, while inherently aiming to reduce the power consumption in the circuit. During synthesis for low power, it is desirable that most lines in the circuit have a signal probability, either near 0 or near 1. A line with a signal probability near 0.5, has a high switching activity and consumes high dynamic power. Transformations proposed here try to reduce the switching activity of lines in the circuit, especially those with high load capacitance.

A. Power Transformations

It is observed that for all previous structural transformations, the second step involves all redundancies in the circuit. However, some lines removed here may have contributed to maintaining a lesser *switched capacitance* in the circuit. These lines, if preserved, may contribute towards a low power consumption. The two steps involved in *Power Transformations* are:

1. Making connections as in Transformations 1-8.
2. *Selective redundancy* elimination.

During redundancy removal, identify two types of redundancies are identified. The first type, *redundancy_type A*, are those lines whose removal will reduce the switching activity in the circuit. These type of lines will be outputs of gates, whose removal is followed by removal of the gate inputs. Lines with signal probability, in the vicinity of 0.5, also will be good candidates for removal. The other type, *redundancy_type B*, are those lines whose removal does not reduce the power consumption of the circuit. A line which is a fanout line from a gate and has a low switching activity can be retained, as it does not consume much power. This line may, on the contrary, aid in suppressing the activity of the gate at its output. *Selective redundancy* elimination removes only *redundancy_type A* lines and retains *redundancy_type B* lines. After more transformations, the *redundancy_type B* lines may not remain redundant in the circuit. However, at the end of the synthesis process, all redundancies are removed to make the circuit irredundant. The following example illustrates the power transformation concept.

Example 4.3: Consider the circuit shown in Fig 5. Let line a be introduced as part of a transformation in Step 1. As a result, say, two other lines, *1* and b become redundant. Removing line *1* causes removal of the OR gate feeding it. This, in turn, can remove more lines, including lines *2* and *3*. Here, *1* is a

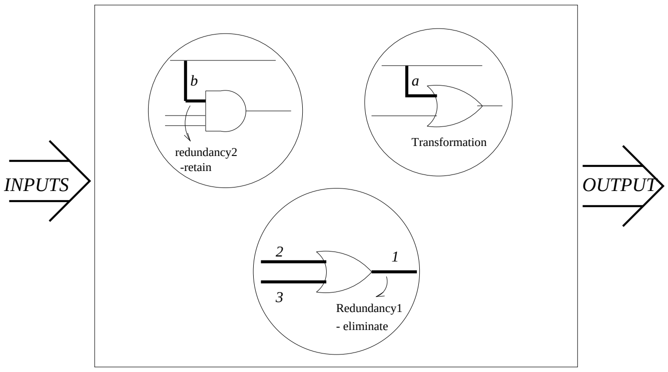

Figure 5: Selective redundancy elimination

redundancy_type B line as this would considerably reduce the area and power. On the other hand, removal of line b removes only one connection, thus reducing the total capacitance by one gate capacitance C_g. Line b might actually help dampen the switching activity at the output of the gate that it feeds. Hence, if the switching activity of line b is very small, line b can be identified as a *redundancy_type B*. Any other gate-level synthesis tool will remove the *redundancy_type B* lines, resulting in a higher power consuming circuit.□

B. Transformations Based on Trend

As mentioned, transition activity in the circuit can be minimized by having more lines which have controllabilities in a lopsided fashion (close to 0 or 1). Transformations made during optimization might improve the signal probability on some lines while worsening the same on other lines. Also, successive transformations might attempt to change the signal probabilities of the same line in opposite directions. Hence, certain trends should be established on the bad lines: the process of making connections should push the signal probability of the bad lines in a fixed direction. During a transformation, there will be some lines whose signal probabilities will be changing in the preferable direction and some whose signal probabilities will be changing in the opposite direction. A transformation is only made when most of the lines weighted by their load capacitances change their signal probabilities in a favorable direction. During this transformation, a cost parameter is calculated based on the trend of the signal probability values. If the signal probabilities of m lines are changed in the favorable directions and n of them move in the opposite direction, the cost calculated is, therefore, given by:

$$Trend\_cost = \sum_{\forall m} (\Delta p)_i \times fanout - \sum_{\forall n} (\Delta p)_i \times fanout \qquad (7)$$

where $(\Delta p)_i$ is the change in signal probability at the line being considered.

The transformation can be described as a four step process.

1. Making connections, as in Transformations 1-8.

2. Redundancy removal.

3. Calculate *Trend_cost*.

4. Reject transformation if *Trend_cost* is non-positive.

The transformations for trend have to be applied consecutively to make a considerable reduction in the switching capacitance of the circuit.

C. Dampening Switching Activity in Gates

The signal probabilities of certain lines in the circuit can be changed by pulling in appropriate lines at the fanins of their gates. This requires that the lines pulled not alter the functionality of the circuit and that the gates have skewed signal probability after the transformation. It has been observed that transformations based on *observability implications* can pull in lines at a gate that can change its switching activity and yet maintain the functionality. *Controllability implications* would not be able to change the signal probability of the gate to which the line is pulled in. Though redundancy elimination following the transformation may change the signal probability of the gate, *observability implications* make an immediate impact on the switching activity. This type of transformation can be used to target specific lines with high switching activity and is very powerful. The following steps outline this technique.

1. Find a line y with high switching activity and high load capacitance.

2. Find all lines which have observability-based implications with y.

3. Choose the best line to dampen y's switching activity.

4. Make the connection.

Example 4.4: In Fig. 6, we consider an OR gate with signal probability 0.64 trying to dampen its switching activity. It can be observed that pulling in line c into the OR gate would change the signal probability of line l from *0.64* to *0.71*, contributing in a lower power consumption of the circuit. Such a transformation is possible due to the condition $l = 1 \overset{O}{\rightarrow} c = 1$. However, the load capacitance at line c has now increased, due to the connection which should be compensated for

by the dampening of the switching activity at l.□

D. Reducing EX-OR Gates

EX-OR gates are prone to high switching activity because EX-OR gates are not biased towards a particular value, and the output of an EX-OR gate is sensitive to all its inputs. Thus, transformations which reduce the number of EX-OR gates in a circuit will reduce the power consumption of a circuit. Don't-care conditions at the inputs to the EX-OR gates and at the outputs of the EX-OR gates can be

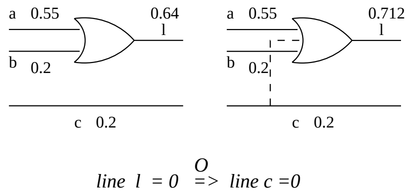

Figure 6: Dampening activity of lines

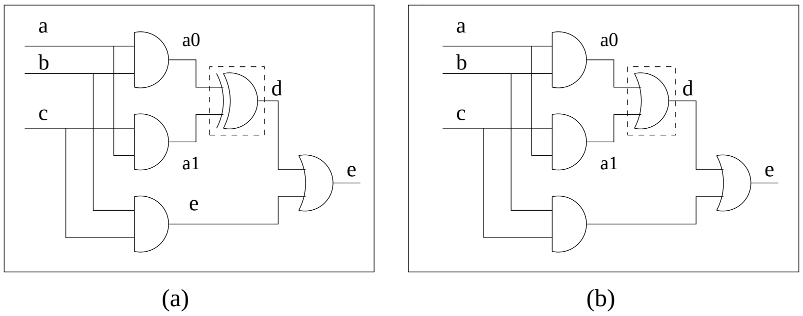

Figure 7: Changing EX-OR gates to an OR gate.

used to transform them into other types of gates [11, 9]. These don't-cares can be determined either using an ATPG tool, or by identifying the observability don't cares. The following two theorems can be applied to transform EX-OR (or EX-NOR) gates to other 2-input gates.

Four simple rules are outlined to convert an EX-OR gate to a two-input gate, based on observability don't cares.

Theorem 4.1: Let y be a node feeding from an EX-OR gate with two inputs, a_0 and a_1. The necessary conditions and the permissible functions are as follows:

1. If ($a_0 = 1$, $a_1 = 1$) is observability don't care, then $y = a_0$ OR a_1 is permissible.

2. If ($a_0 = 0$, $a_1 = 0$) is an observability don't care, then $y = a_0$ NAND a_1 is permissible.

3. If ($a_0 = 1$, $a_1 = 0$) is an observability don't care, then $y = \bar{a}_0$ AND a_1 is permissible.

4. If ($a_0 = 0$, $a_1 = 1$) is an observability don't care, then $y = a_0$ AND $\bar{a}_1$ is permissible.

Proof: The permissible functions can be derived from the truth table of the EX-OR gate.□

Example 4.4: The circuit shown in Fig.7(a) can be represented as ($ab \oplus bc+ca$). At node d, the condition for a change from 0 to 1, to be observed at e, is $\{(a_1,a_0)=(0,0)\}$. So $\{(a_1,a_0)=(1,1)\}$ is an observability don't care for the EX-OR gate. This condition can allow the EX-OR gate at node d to be transformed to an OR gate, yielding the circuit in Fig.7(b).□

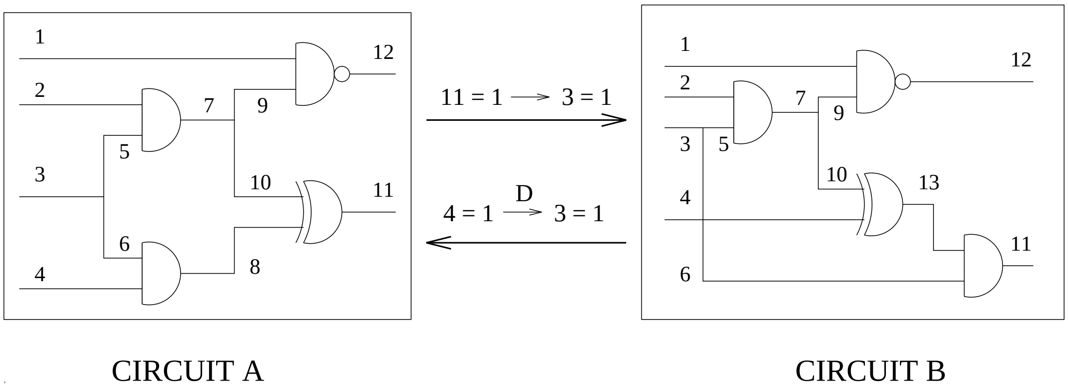

Figure 8: Transformation making a difference in power consumption.

5 Synthesis for Low Power

This Section describe the flow of the optimization process with the primary objective of reducing power consumption and a secondary objective of reducing area. All the transformations mentioned in previous sections are used, guided by a cost function in terms of the switched capacitance in the circuit. The method is greedy in nature. However, we use techniques to 'shake' the resultant circuit structure to come out of a 'local minima'. The two main features include a synthesis guidance procedure based on the cost value, and a method for efficient detection of the switched capacitance in the circuit. The switched capacitance is measured based on the proposed model. However, the power consumed by the circuit has to be estimated at every iteration to provide a proper guidance to the synthesis process.

It is observed that synthesis with area optimization (or other goal) does not necessarily result in a circuit with minimum power. A transformation accepted due to a lesser area can be bad in terms of power consumption. As an example, in Fig 8, we show two small circuits A and B. The two circuits can be obtained from one another during optimization, by means of the implications shown. Assuming the inputs having a signal probability of 0.5, the power estimated in circuit A is less than in circuit B. Note that both the circuits have the same area. A synthesis method aiming at a lesser area may transform circuit A to B, as in HANNIBAL [2]. However, during power optimization, circuit A is a better choice; a synthesis procedure encounters a large number of such situations, where a correct decision leads to a better circuit.

5.1 Incremental Estimation of Switching Activity

As the synthesis method is guided by the switched capacitance of the circuit, the incremental estimation of the switching activity should have a low time overhead. After each transformation, the change in the signal probabilities of the lines are estimated by updating only those signal probabilities which are affected by the transformation. These updated signal probabilities are used to calculate the switching activity of the circuit. Detailed description of the incremental signal probability calculation can be found

```
     Optimization Algorithm For Low Power(Iter)
     Begin
1.   For i = 1 to Iter do
2.     random_pattern_simulate(circuit) /* Estimate signal probabilities */;
3.     power_estimate(circuit) /* Using proposed model */;
4.     if i is even then
5.       For all lines in the circuit do
6.         Make Transformations 1 - 8;
7.         selective_red_eliminate(circuit);
7.         Remove unnecessary EX-OR gates;
7.      else
8.       For all lines in the circuit do
9.         Make Transformations for Trend;
10.    For lines with high switching activity do
11.      Dampen signal probabilities;
12.  Redundancy elimination; /* Final step */
     End
```

Figure 9: Algorithm for Power Optimization

in [21]. The proposed method uses random-pattern simulation for better signal probability estimation after every predetermined number of *incremental updates* on the signal probability values.

5.2 Synthesis Algorithm

There can be a number of algorithms based on the proposed transformations to synthesize a multi-level circuit for a given cost function. One such algorithm is outlined here. Also outlined are some of the primary heuristics used in our implementation. The cost function chosen places a primary emphasis on power and a secondary emphasis on area.

The outline of a procedure is presented in Figure 9. The algorithm starts with signal probability estimation of the circuit using random pattern simulation of the circuit. The number of patterns used here depends on the size of the circuit. The signal probability calculations also take into account any particular signal probability distribution at the inputs of the circuits. The signal probabilities are updated after every transformation. The cost of the synthesis scheme is the *switching capacitance*, determined in Step 2. The algorithm then makes a certain number of predetermined iterations through the circuit making various types of transformations. In this particular algorithm, the different types of transformations have been scheduled as shown in Fig 9. Finally, a redundancy elimination pass is made through the circuit in order to produce an irredundant circuit.

Besides the new transformations, certain heuristics have also been used, special to synthesizing circuits for minimizing power. They are as follows:

Circuit Name	Original			Proposed			Proposed/Hannibal		
	2-i/p GE	Delay	Power	2-i/p GE	Delay	Power	2-i/p GE	Delay	Power
c432	175	18	1077.0	103	18	683.8(541.5)	1.00	0.78	0.83(0.93)
c499	340	17	2036.2	333	19	1997.5(1577.6)	1.00	1.00	1.00(1.00)
c880	320	24	2302.9	285	28	1891.6(1487.1)	0.98	0.96	0.86(0.97)
c1355	496	24	3684.3	348	22	2219.9(1711.7)	0.90	0.91	0.57(0.78)
c1908	529	37	4262.6	323	28	2083.7(1751.8)	0.90	0.90	0.70(0.86)
c2670	779	30	5830.9	508	29	3538.0(3338.1)	0.98	1.30	0.80(1.01)
c3540	1113	45	6830.6	836	41	4572.9(4240.8)	0.96	1.00	0.89(0.94)
c5315	1746	47	14845.6	1344	47	10205.8(8810.4)	0.98	1.00	0.82(0.95)
c6288	2384	-	23684.9	1873	-	21186.8(10439.4)	1.00	-	0.99(1.00)
c7552	2367	40	21678.4	1281	86	10089.0(7425.1)	0.99	0.89	0.76(0.92)

Table 1: Comparison of Power with Original and HANNIBAL.

1. Choosing lines to dampen switching activity at a gate: There can be several lines which can be pulled in to dampen the switching activity of a gate. The best line depends on the nature of the gate. For dampening the ouput of an AND or NAND gate, a line with signal probability close to 0.0 looks ideal. Similarly, for an OR or NOR gate, a line with signal probability close to 1.0 appears to be the best. Also, one should be careful in dampening an AND or NAND (OR or NOR) gate whose signal probability is greater (less) than 0.5.

2. Transformations to remove EX-OR gates: Transformations to remove EX-OR gates for power reduction should immediately follow transformations introducing EX-OR and selective redundancy removal.

3. Cost function variation: In the synthesis process, area optimization is also allowed in restrictive cases where it does increase the power consumption of the circuit.

4. Higher learning dampen high switching activity gates: The synthesis scheme also spends some extra effort in determining implications to dampen some lines in the circuit which have high high switching activity.

5. Moving out of a local minima: We also devote time during synthesis where we allow the area and the switch capacitance of the circuit to rise. For example, transformations based on the trend of the switching activity of the circuit disregard the area and the effect of the load capacitances in some iterations. Also certain types of transformations allow the area to increase.

Some of these heuristics are embedded within the new transformations, whereas the others are realized in the flow of the synthesis process.

Circuit Name	Proposed		Proposed/SIS 1.1_RUG		Proposed/SIS 1.1_ALG	
	2-i/p GE	Power	2-i/p GE	Power	2-i/p GE	Power
c432	103	683.8	0.85	0.77	0.65	0.68
c499	333	1997.5	0.86	0.62	0.86	0.62
c880	285	1891.6	0.95	0.85	0.89	0.80
c1355	348	2219.9	0.91	0.71	0.76	0.71
c1908	323	2083.7	0.92	0.80	0.89	0.83
c2670	508	3538.0	0.95	0.84	0.88	0.82
c3540	836	4572.9	0.94	0.92	0.92	0.82
c5315	1344	10205.8	1.05	0.88	0.97	0.84
c6288	1873	21186.8	0.86	0.94	0.87	1.11
c7552	1281	10089.0	0.75	0.65	0.71	0.60

Table 2: Comparison of Power with SIS 1.1 - script.rugged and script.algebraic.

Circuit Name	Pre-map Area		Pre-map Power		Post-map Area		Post-map Power	
	Prop/ALG	[19]/ALG	Prop/ALG	[19]/ALG	Prop/ALG	[19]/ALG	Prop/ALG	[19]/ALG
c432	0.68	1.01	0.62	0.96	0.77	0.90	0.62	0.92
c880	0.80	0.99	0.81	0.91	0.97	1.00	0.98	0.92
c1355	0.71	1.00	0.67	0.94	0.81	0.98	0.70	0.91
c1908	0.83	1.01	0.96	1.01	0.97	0.90	0.99	0.94
rot	0.90	1.01	0.91	1.01	0.89	1.00	0.87	0.99
b9	0.77	1.28	0.81	1.08	0.83	0.99	0.75	0.99
t481	0.04	0.99	0.11	0.98	0.04	0.07	0.06	0.78

Table 3: Ratio of power consumption for both mapped and unmapped logic.

6 Experimental Results

The algorithm outlined in the previous Section was implemented to demonstrate the effectiveness of the transformations to minimize power and area. The circuits were power-optimized based on a input signal probability distribution as obtained from [22]. The tool was applied on the ISCAS85 [10] circuits and some additional circuits available from SIS. The circuits were compared to those synthesized by SIS-1.2 [9, 8] and HANNIBAL [2].

The resultant circuits were compared based on the power estimated, the area and also the 'longest sensitizable' path. The power is estimated based in the 'zero-delay' model. We present comparison in the power consumptions as estimated by our model as well as by SIS-1.2. To estimate the area, we have used the number of *2-input gates*, denoted as 2-i/p GE (equivalent to half the *number of connections* in [2]) to evaluate the resultant area. A two-input EX-OR gate is counted as 2.5 two-input gates. Though the synthesis tool did use path delay in the cost function, the 'longest sensitizable' path lengths of the resultant circuits were measured [23]. The path length is counted in terms of the number of levels in the unmapped circuit. The number of levels of an EX-OR gate is assumed to be two.

6.1 Comparison with HANNIBAL

HANNIBAL [2] optimizes circuits on the basis of area only. The synthesis method described here tries to minimize power keeping the area as low as possible. The results as displayed in Table 1 compare the performance on the ISCAS85 benchmark circuits. The power estimation shown in the Table is the power measured by the proposed model. The power corresponding to the brackets in column 7 and 10 is based on the power estimated by SIS-1.2. It can be observed that for 8 out of the 10 circuits the proposed tool generates a circuit with lesser power. The average reduction of power as estimation by the proposed model is 18%. However, the average reduction when measured using SIS-1.2 is 6.7%. This is because the proposed model is used as the cost function during synthesis. An accurate power estimation model can further reduce the power consumption in the resulting circuits. Also there is very small area penalty incurred, compared to HANNIBAL, during power optimization. The path delay is found to increase only in 1 out of the 9 circuits.

6.2 Comparison with SIS-1.2

The area and the power of the resultant circuits are compared with those by SIS-1.2 in Table 2. For SIS-1.2, the scripts considered were *script.rugged* and *script.algebraic*. Both of these scripts optimize the area of the circuit using optimization techniques based on the boolean network. The proposed method optimizes both the area and the power of the circuit at the gate-level. The area shown in Table is measured by 2-i/p gate equivalents, and the power is measured, based on the proposed model. In almost all the circuits, both the area and power estimated is found to be considerably less. For *script.rugged*, the average reduction in area and power is 10% and 20% respectively, whereas for the the *script.algebraic*, the area and power reduction is 16% and 22% respectively.

6.3 Comparison after Technology Mapping

The proposed synthesis method is technology-independent, the output being an unmapped gate-level circuit. However, optimization for power is only useful if the effect can be carried even after *technology mapping.* Table 3 presents the comparison of certain mapped and unmapped circuits. Gate-level circuits produced by the proposed optimized tool and by SIS-1.2 (script.algebraic) are considered. During technology mapping, the circuits are mapped to the library *stdcell2_2.genlib.* The power, as measured by SIS-1.2 is reported in the Table; as shown, there is considerable reduction in power as well as area in the mapped circuit.

The circuits implemented in [19] were not available yet for comparison. However, in Table 3 we show the amount of savings achieved by [19] over SIS-1.2(script.algebraic), as reported in the paper. Though the method proposed in [22] uses a different technology mapping library, some approximate comparison remain valid. The power measured after technology mapping for the proposed tool is reduced by 29%, compared to 7.85% as reported in [19].

7 Conclusions

A technology-independent multi-level synthesis method to optimize area and powerhas been presented. Based on the gate-level representation of the circuit, this method uses structural transformations. Several new structural transformations have been proposed to reduce power, as well as a providing a new synthesis methodology to guide the circuit towards less power and area. Experimental results demonstrate that the proposed transformations possess the strength to synthesize circuits with a lesser cost function, compared to HANNIBAL, SIS-1.2 and [19]. Furthermore, these transformations can also be extended to sequential circuits, and circuits after technology mapping.

References

[1] W. Kunz, D.K. Pradhan, "Recursive Learning: A New Implication Technique for Efficient Solution to CAD Problems - Test, Verification and Optimization," IEEE Trans. on CAD, Sept., 1994.

[2] W. Kunz, P.R. Menon, "Multi Level Logic Optimization by Implication Analysis," ICCAD 1994.

[3] M. Chatterjee, D. Pradhan and W. Kunz, "LOT: Logic optimization with testability - New Transformations using recursive learning," ICCAD 1995.

[4] K.T. Cheng, L.A. Entrena,"Multi-level Logic Optimization by Redundancy Addition and Removal," EDAC, pp. 373-377, 1993.

[5] S.C. Chang, M. Marek-Sadowska,"Perturb and Simplify: Multi-Level Boolean Network Optimizer," ICCAD, pp. 2-5, 1994.

[6] S. Muroga *et. al*, "The Transduction Method - Design of Logic Networks Based on Permissible Functions," IEEE Trans Compt., Oct 1989.

[7] B. Rohfleisch, F. Berglez, "Introduction of Permissible Bridges with Application to Logic Optimization after Technology Mapping," Proc. EDAC/ETC/EUROASIC, 1994.

[8] R.K. Brayton, *et. al*, "MIS: Multi-level Interactive Logic Optimization System," IEEE Trans on CAD, pp. 1062-1081, Nov., 1987.

[9] G. De. Micheli, "Synthesis and Optimization of Digital Circuits," McGraw-Hill, Inc. 1994.

[10] F. Brglez, H. Fujiwara , "A Neural Netlist of ten Combinational Benchmark Circuits and a Target Translator in FORTRAN,", Proc. Intl. Symp. on Circuits and Systems, June 1985.

[11] T. Sasao (ed.) *Logic Synthesis and Optimization*, Kluwer Academic Publishers, 1993.

[12] N. Weste, K. Eshraghian, "Principles of CMOS VLSI design - A systems perspective," Addison Wesley Publishing Co.

[13] F. Najm, "Transition Density: A new measure of Activity in digital circuits," IEEE Transactions on Computer-Aided design of Integrated Circuits and Systems, Feb 1993.

[14] S. Iman, M. Pedram, "Multi-level network optimization for low power," ICCAD, 1994.

[15] H. Mehta, *et. al*, "Accurate Estimation of Combinational Circuit Activity," pp. 618-622, DAC 1995.

[16] S. Devadas, S. Malik, "A survey of optimization techniques targetting low power VLSI circuits," pp. 242-247, DAC 1995.

[17] A. Ghosh, *et. al*, "Estimation of average switching activity in combinational an d sequential circuits," pp. 253-259, DAC 1992.

[18] A. Shen, et.al, "On average power dissipation and random pattern testability of CMOS combinational logic networks," ICCAD, 1992.

[19] S. Iman, M. Pedram, "Logic extraction and decomposition for low power," DAC, 1995.

[20] S. Iman, M. Pedram, "Two level logic minimization for low power," ICCAD, 1995.

[21] Authors, Technical Report on the paper.

[22] S. Iman, *Private Communication*, Dept. of Electrical Engg., University of Southern California.

[23] J. Bell, *Private Communication*, Dept. of Computer Science, Texas A&M University.