

# Multi-level Logic Optimization for Low Power using Local Logic Transformations

Qi Wang                      Sarma B. K. Vrudhula

Center for Low Power Electronics

Department of Electrical and Computer Engineering

University of Arizona

Tucson, AZ 85709

## Abstract

*In this paper we present an efficient technique to reduce the switching activity in a CMOS combinational logic network based on local logic transformations. These transformations consist of adding redundant connections or gates so as to reduce the switching activity. Simple and efficient procedures, based on logic implication, for identifying the sources and targets of the redundant connections are presented. Additionally, procedures that permit the designer to trade-off power and delay after the transformations are described. Results of experiments on the MCNC benchmark circuits are given. The results indicate that significant reduction of the switching activities of a CMOS combinational circuit can be achieved with a very low area overhead and low computational cost.*

## 1 Introduction

Power consumption has become an important optimization metric in the design of microelectronic systems. The dominant component of power dissipation in CMOS logic is due to charging and discharging capacitances. The average power dissipated in charging or discharging the load capacitance is given by

$$P = \frac{1}{2} \times C_{load} \times \frac{V_{dd}^2}{T_{cycle}} \times E(\text{switching}). \quad (1)$$

$C_{load}$  is the load capacitance of the gate,  $T_{cycle}$  is the clock cycle time,  $E(\text{switching})$  is the expected number of signal transitions per cycle and  $V_{dd}$  is the power supply voltage. Hence the power consumption of a CMOS combinational network can be determined by summation of the average switching activity of each node weighted by its load capacitance. Apart from reducing the parasitic capacitances by careful layout, one other way to reduce the power consumed by a logic gate is to reduce the switching activity.

Several methods to reduce the switching activity in CMOS combinational circuits have been proposed. In [4, 7] the don't care sets of internal nodes of the combinational network are used to minimize the weighted average switching activity for each node. In [9], a modified version of the ESPRESSO [5] heuristic is used to search for a cover of cubes that have

reduced switching activity. In [8, 16], the problem of two-level logic optimization for low power is addressed. In [16], the authors show how to balance the potentially conflicting requirements of minimizing power and area. Under the assumption of equal signal probabilities, an algorithm is presented which minimizes the amount of overlap among cubes in a cover while attempting to maximize the sizes of the cubes. In [8] the authors extend the idea further and solve the problem of optimizing a two-level combinational circuit for low power without the assumption of equal input signal probabilities.

All of the works described above depend on the computation of prime implicants or local don't care sets of internal nodes. This may be very time consuming. In [14], power reduction is achieved by identifying a subnetwork (guarded subnetwork) that can be disabled. The disabling of a subnetwork is achieved by adding a latch to each input of the guarded subnetwork. More recently, in [6] the authors proposed a structural transformation approach based on the concept of permissible functions. Due to their use of ATPG techniques, the computation cost may be high. However, they report an average of 26% power reduction.

The rest of this paper is organized as follows. In Section 2.1, the power estimation model [17] used in this paper is introduced. The ideas of adding redundant connections and how they may reduce the switching activity are explained in Section 3. In Sections 3.1, 3.2 methods for identifying the target gates based on single-source implication are described. To increase the size of potential solutions, target selection based on multiple-source implication is presented in 3.4. In Section 4 has a brief description of how power and delay can be traded off after the logic transformations. In Section 5, enhancements to the basic methods are described. Finally, in Section 6 experimental results on MCNC benchmark circuits are given.

## 2 Preliminaries

### 2.1 Power Estimation Model

Assuming that  $V_{dd}$  and  $T_{cycle}$  are fixed for a given circuit, an estimate of the total power consumption of a CMOS combinational circuit (based on Equation 1),

is given by

$$P_{total} = a \times \sum_{\forall \text{ gates } i} C_{load}(i) \times E(i), \quad (2)$$

where  $a$  is a constant under the assumption, and  $C_{load}(i)$  and  $E(i)$  denote the load capacitance and the expected number of logic transitions (expected switching activity) at gate  $i$  respectively. For brevity, in the following we will refer the term  $C_{load}(i) \times E(i)$  as the *switching capacitance* of gate  $i$ .

In [17], an efficient technique for estimating the average number of signal transitions in CMOS logic circuits under the zero delay model is presented. In this method, a logic signal  $x$  is modeled as a two state, discrete time Markov chain, which is characterized by two parameters,  $\lambda_x$  and  $\mu_x$ , where  $\mu_x$  is the 0→1 transition probability and  $\lambda_x$  is the 1→0 transition probability. Let  $H_x$  ( $L_x$ ) be the random variable that represents the length of a 1-run (0-run) on a logic signal  $x$ . It is a simple matter to show that  $H_x$  and  $L_x$  are geometrically distributed with expectations given by:

$$E(H_x) = \frac{1}{\lambda_x} \quad \text{and} \quad E(L_x) = \frac{1}{\mu_x}, \quad (3)$$

Furthermore, the expected number of signal transitions (per unit time) is given by:

$$E(x) = \frac{2}{\frac{1}{\lambda_x} + \frac{1}{\mu_x}} \quad (4)$$

For each type of gate, the parameters of the output signal can be determined by the parameters of the gate's input signal. For example, the parameters,  $\lambda_f$  and  $\mu_f$ , of the output of an  $n$ -input *and* gate are given by

$$\lambda_f = 1 - \prod_{i=1}^n (1 - \lambda_i) \quad (5)$$

$$\mu_f = \frac{\lambda_f \prod_{i=1}^n \mu_i}{\prod_{i=1}^n (\lambda_i + \mu_i) - \prod_{i=1}^n \mu_i} \quad (6)$$

Similar formulas for other logic gates are derived. These formulas, which are exact in the absence of reconvergent fanout, allow local propagation of these parameters and result in a very fast method for estimating expected signal activity. In the presence of reconvergent fanout the parameters can be computed using BDDs [17]. We use this technique while performing the logic transformations to efficiently re-evaluate the switching activities at each node.

### 3 Overview of Transformations

Consider the circuit shown in Figure 1(a). The corresponding table on the right shows the parameters,  $\lambda$  and  $\mu$ , and  $E$ , the expected number of signal transitions per unit time, associated with each signal. Pseudo random binary waveforms with the given input transition probabilities were applied to the circuit. Input  $c$  has relatively few transitions and since  $\mu(c)$  is

much smaller than  $\lambda(c)$ , the binary waveform on  $c$  has long runs of zeros (see Equation 3). Suppose  $c$  and  $g_1$  are selected as the source and target of a new connection  $\ell_1$ . This connection is redundant and hence the function of the new circuit remains the same. Note that by adding  $\ell_1$ ,  $E(g_1)$  reduces from 0.49 to 0.078. Additionally, with the addition of  $\ell_1$ , both the connections from  $c$  to  $g_3$  and  $g_4$  become redundant (e.g., s-a-1 faults on these two lines are undetectable). After removing these new redundancies, the circuit shown in Figure 1b is obtained. This transformation reduced the total switching capacitance from 5.265 to 4.181; a 21% reduction. (In this example, we assume the input load capacitances of all gates are the same, e.g. unit 1. This is generally true for the small primitive gates as shown in the example.)

The circuit in Figure 1(a) was also simulated using waveforms with different input parameters. These parameters and the expected signal transitions are shown in the table corresponding to Figure 1(c). This time gate  $g_3$  was selected as the source. The binary waveform associated with the output of gate  $g_3$  indicates long runs of ones. Gate  $g_5$  is selected as the target. The addition of the (redundant) connection from  $g_3$  to  $g_5$  (line  $\ell_2$ ) results in a 26.4% reduction in the total switching capacitance, which is due to the significant switching activity reduction at  $g_5$  and  $g_6$ . Note that in this case, although the output load at  $g_3$  and the area of the circuit is increased, we achieve a better structure in terms of the power consumption. However, if the connection  $\ell_2$  was selected with the input parameters shown in Figure 1a, the reduction in switching activity would only be 3.4%, which is primarily due to the removal of gate  $g_4$ . Considering the overhead of the transformation and the estimation error, this may not be desirable. The above example highlights two important points: (1) local logic transformations can be very effective for re-synthesis when the objective is to reduce the switching activity, and (2) the effectiveness of the transformations depends strongly on the input waveforms.

### 3.1 Selecting a Source

The logic transformations consist of adding redundant connections from the output of a low activity gate  $g_s$  (source) to a high activity gate  $g_t$  (target) so that the local function at the gate  $g_t$  is changed and as a result, its activity is reduced. Hence a good candidate for the source gate will be a gate with low activity. Thus sources are selected by first estimating switching activity at each gate. This can be done by (1) generating random binary waveforms on the inputs with given transition probabilities, performing logic simulation and then computing the average number of signal transitions on each signal, or by (2) propagating the parameters as done in [17]. The gates are sorted in increasing order of switching activity and the *low activity* gates are selected in turn as potential sources for the redundant connections.

### 3.2 Selecting a Target by Single-Source Implication

After selecting a source gate, a target gate is sought such that the connection from the source to target is redundant. This can be achieved by simple implication. Let  $TFO(g)$  ( $TFI(g)$ ) denote the the *transitive fanin* (*transitive fanout*) set of gate  $g$ .

**Theorem 1** *Let  $g_i$  and  $g_j$  be two logic gates and  $c_j$  be the controlling value of gate  $g_j$ . If  $g_i = v$  ( $v \in \{0, 1\}$ ) implies that  $g_j$  is unobservable, where  $g_j \notin TFI(g_i)$ , and  $v = c_j(\bar{c}_j)$  then the non-inverted (inverted) connection from  $g_i$  to  $g_j$  is redundant and the local function at  $g_j$  will be changed by adding this redundant wire.*

As an example, consider the circuit shown in Figure 2. Gate  $g_1$  is a gate with a very low switching activity. Suppose we imply a logic 0 on the output of gate  $g_1$  (the value chosen for forward implication is explained below). After forward implication, we perform backward propagation of *unobservabilities* [10]. Since one of the inputs to  $g_4$  is its controlling value, the other input (output of  $g_3$ ) is unobservable. This is indicated by a  $u$ . Propagating  $u$  as far back (toward to the primary inputs) as possible shows that gate  $g_2$  is unobservable. The connection from  $g_1$  to the input of  $g_2$  is redundant. This is easily seen since a s-a-1 fault on the added connection is undetectable. Note also that both  $g_2$  and  $g_3$  have high switching activities before the redundant connection was added and the significant reduction in switching activity is due to the new connection.

In the backward propagation of the unobservabilities, if a fanin of a gate  $g$  has a controlling value of  $g$  (assigned during the implication phase), then all the other fanins of  $g$  are marked as unobservable. If a gate is unobservable, all its inputs are marked as unobservable. The complicated case is the back-propagation of the unobservabilities at a gate with multiple fanouts. In general, a gate with multiple fanouts may observable even when all its fanouts are unobservable [10]. This involves the multiple stuck-at fault identification. However, this does not pose any difficulties in this application since the new untestable fault is always a branch fault, i.e., a single stuck-at fault. Hence, a gate with multiple fanouts is marked unobservable if all its fanouts are unobservable.

Not all redundant connections will change the function at the target gate. Figure 3 shows a simple example taken from [3]. In this example,  $g_1 = 1$  implies  $\bar{a} = 0$ . Hence the inverted connection from  $g_1$  to  $g_2$  is redundant. But this connection will not change the function realized by  $g_2$ , and the switching activity at  $g_2$  will remain the same. Therefore such a connection may not be useful for reducing the switching activity.

### 3.3 Selecting the Implication value

Recall that a target gate is found by first performing a logic implication on the source gate  $g_s$ . If a logic 1(0) is implied at  $g_s$  and a target gate  $g_t$  is found, then

the s-a-0(1) fault at the connection from  $g_s$  to  $g_t$  will be undetectable and the connection will be redundant. However in practice, given a source gate, not both a logic 1 and a logic 0 implication will be useful. For example, suppose a source gate  $g_s$  has a low signal activity and a much smaller value of  $\lambda$  than the value of  $\mu$ . This means that the binary sequences at  $g_s$  will *typically* have long runs of ones. According to the Theorem 1 if a logic 0 is implied at the output of  $g_s$  and the target gate  $g_t$  is found to be unobservable, then connection from  $g_s$  to  $g_t$  will be redundant. Furthermore, if the  $g_t$  is an AND/NAND gate, a non-inverted connection can be added without changing the function at the primary outputs. However, because  $g_s$  has long runs of ones, and since 1 is not the controlling value of  $g_t$ , the added connection will have little effect on the switching activity at  $g_t$ . Similarly, if the  $g_t$  is an OR/NOR gate, an inverted connection can be added without changing the function at the primary outputs. Because the output of  $g_s$  is inverted, the input to  $g_t$  will have long runs of zeros. Again, this will not have any significant effect on the switching activity at  $g_t$ . Thus if the binary sequence at  $g_s$  has low switching activity and has longer runs of zeros (ones) than the runs of ones (zeros), a logic value of 0 (1) will be implied at the output of  $g_s$ . Figure 4 summarizes the basic steps involved in finding a target using single source implication.

### 3.4 Multiple Source Implication: A Covering Problem

The backward propagation of the unobservability in the single-source implication algorithm will stop at a gate with multiple fanout or at a primary input. To create more potential redundant connections, the unobservabilities should be propagated as far back as possible. If some but not all of the fanouts of a gate  $g$  are unobservable, then the backward propagation of the unobservability cannot be continued to the inputs of  $g$ . Multiple-source implication attempts to make all the fanouts of a multiple fanout gate  $g$  unobservable by finding an *optimal* subset of sources found during the single-source implication procedure that will collectively make  $g$  unobservable.

**Definition 1** *Let  $g_t$  be a gate with fanout  $r$ . For each fanout branch  $k$  of  $g_t$ , let  $I_k(g_t) = \{g_j^{v_j}\}$ , where  $g_j$  is a source gate such that  $g_j = v_j \Rightarrow$  at least one fanout of  $g_t$  is unobservable.  $I_k(g_t)$  is called the implication set for fanout branch  $k$  of gate  $g_t$ .*

Suppose that the backward propagation of unobservabilities stopped at gate  $g_t$  and could not proceed to its inputs. In this case,  $\bigcap_{j=1}^r I_j(g_t) = \emptyset$  since otherwise gate  $g_t$  would have been marked as unobservable by the single-source implication procedure and multiple-source implication would not be needed to propagate the unobservability to its inputs.

**Theorem 2** Let  $g_t$  be a target gate with fanout  $r$  and  $C = \{g_{i_1}^{v_{i_1}}, \dots, g_{i_j}^{v_{i_j}}\}$  be a cover of the sets  $I_1(g_t), \dots, I_r(g_t)$ . Let  $g^*$  be a gate such that:

- (1)  $g'_{i_1}, \dots, g'_{i_j}$  are its inputs, where  $g'_{i_k} = v'_{i_k} \oplus g_{i_k}$ , for  $k = 1, \dots, j$ .
- (2) If  $g_t$  is an (AND or NAND)/(OR or NOR) gate,  $g^*$  is an NAND/AND gate.

Then the connection from  $g^*$  to  $g_t$  is redundant and the local function at  $g_t$  will be changed.

As an example, consider the circuit shown in Figure 5.  $I_1(g_t) = \{g_1^0, g_2^1\}$ ,  $I_2(g_t) = \{g_3^1, g_4^1\}$ , and  $I_3(g_t) = \{g_3^1\}$ . Note that  $g_3^1$  represents the condition that if  $g_3 = 1$ , then fanout branches 2 and 3 of  $g_t$  will be unobservable. A cover of the sets is  $C = \{g_1^0, g_3^1\}$ . A new gate  $g_n$  is added with  $\overline{g_1}$  and  $g_3$  as its inputs. To test the  $s-a-0$  fault on the connection from  $g_n$  to  $g_t$ ,  $g_1$  must be 0 and  $g_3$  must be 1. However, from Figure 5(a),  $g_1 = 0$  and  $g_3 = 1$  will result in all the fanouts of  $g_t$  to be unobservable. Hence the fault is unobservable at the primary outputs. That is the connection from  $g_n$  to  $g_t$  is redundant. Furthermore, the local function at  $g_t$  will be changed.

It is easy to see that the size of the cover determines the number of inputs of the newly created gate. To minimize the number of inputs to the new gate, we need to find a *minimum cardinality cover* of the implication sets of the fanouts of the target gate. Note that the minimum cardinality cover need not be the best in terms of switching activity reduction. For example,  $C = \{g_2^1, g_3^1\}$  is also a cover, and may result in a greater reduction in switching activity. The algorithm of the multiple-source implication is shown in the Figure 6.

### 3.5 Redundancy Removal

The addition of redundant connections may make one or more irredundant connections redundant [2, 3]. For this reason the final step in the procedure is redundancy removal. Clearly, the redundant connections that were added to reduce the switching activity are not removed. The redundancy removal procedure used is presented in [10]. This algorithm also uses logic implication and is easily implemented using the routines used in the single-source and multiple-source implication algorithm.

## 4 Power and Delay Trade-Offs

The biggest concern regarding logic “re-synthesis” (i.e., post technology mapping) for low power may be its effect on circuit performance or delay. In this section, we describe a useful approach to carry out power-delay tradeoffs after the transformations. Since the logic transformations can be carried out after technology mapping, any delay increase in the transformed circuit is due to the added redundant connections.

This makes it easier to tradeoff power for delay or visa versa. If the transformed circuit violates the timing constraints, it is therefore possible to make the circuit satisfy the timing constraints by removing some of the redundant connections.

The advantages of carrying out the tradeoff between power and delay after the transformation are summarized as follows: (1) It may provide a means to escape from a local optimal solution. For example, the addition of a connection may violate the timing constraints, but this addition may result in other connections becoming redundant, and whose removal results in a decrease in power consumption and/or decrease in the delay of the circuit. (2) Separating the transformation and timing analysis makes the whole process very efficient. If the power-delay tradeoff is done during the transformation, then timing analysis has to be carried out repeatedly. This will be very time consuming. (3) The accuracy of the power-delay tradeoff depends strongly on the underlying timing model in use. Since the timing analysis is not performed too often, we may choose a very accurate timing (e.g., a model that considers false paths). As a result, a more efficient and accurate tradeoff analysis between power and delay is achieved.

A drawback of the above approach is that if the original circuit is drastically changed, then it may not be possible to satisfy the timing constraints by removing some connections after the transformation. One way to circumvent this problem is to start from a transformation with very loose timing constraints, e.g. a limit on the increase in the number of levels of the circuit. This will keep most of the changes in the circuit local and therefore less significant changes over the whole circuit structure. Following this, power-delay tradeoffs with much more stringent timing constraints can be performed. The experimental results demonstrate the effectiveness of this approach.

## 5 Implementation

### 5.1 Enhancements to Single-Source Implication

The result after single-source implication is a set of logic gates (targets) which are unobservable. However, not all the logic gates will be good candidates for the target connection. The following *rules* are incorporated in the single-source implication procedure.

1. If a chain of unobservable gates is found, only the gate at the start of the chain (closest to the primary inputs) is selected as the target gate. This is because any reduction in the switching activity will be propagated along the chain. This will tend to reduce the need for additional connections.
2. A threshold on the difference between the level of the source and target gate is included to control the increase in the number of levels.
3. Selecting a source gate based solely on its switching activity often results in an unacceptably large increase in the delay. For this reason, gates are

first sorted by their level and within each level, they are sorted in increasing order of switching activity. The source gate selection proceeds level by level. This results in a much smaller delay penalty.

## 5.2 Enhancements to Multiple-Source Implication

Multiple-source implication results in the addition of redundant logic. The complexity of this logic is determined by the size of the cover of the implication sets. The size of the minimum covering represents the minimum number of inputs of the new gate whose output will be then connected to the target gate. Recall that all the elements of the minimum size cover are the gates with low signal activities, because they are obtained during the single-source implication step. However, the individual low activity gates may result in a high activity signal at the output of the added gate. This tends to get worse with increasing size of the cover. For this reason, the backward propagation of unobservabilities takes place only for gates with fewer than a given number (user specified) of fanouts. In the experiments reported in this paper, this number was set to 5. As a result, the set covering problem can be solved exactly.

## 6 Experiment Results

The techniques presented in this paper were implemented on a Sparc4. The starting point is a technology mapped circuit using the *mcnc.gentlib* library option of the SIS package [12]. To obtain more potential candidates for adding connections, only 2-input NAND/NOR gates and inverters were used to build the network. The procedures described in this paper were tested on a number of commonly used benchmark circuits. The first few columns of Tables 1 and 2 show a summary of the circuits. The circuit sizes range from a few gates to two thousand gates.

The power consumption is computed using Equation 2. The switching activity for each node in a circuit was obtained by logic simulation. The load capacitance was obtained from the library. The input vectors were obtained in the following way. First, for each primary input, a random signal probability value was generated. Then a independent pseudo-random binary waveform for each input with the given signal probability was generated. Finally, the  $\lambda$  and  $\mu$  parameters for each primary input were extracted from the binary sequence. This constitutes one set of experimental input vectors. For each circuit 10 such sets of data were generated, i.e. for each circuit ten experiments with different random input patterns were performed. The area of the circuit is computed by summing up the gate area values provided by the library. The delay of the circuit was taken to be the maximum of the arrival times taken over all primary outputs. This was obtained by static timing analysis. The delay information of each gate was taken from the library.

To examine the efficacy of carrying out tradeoff analysis after the logic transformations, ten experiments, each under three different constraints, were conducted. The first, referred to as *loose* timing constraints, imposed a limit on the increase in the number of levels allowed as a result of the transformation and no other timing constraint after the transformation was imposed. The second and third constraints imposed a limit of 5% and 1% increase on the delay of the transformed circuit. The results for all these are shown in Tables 1 and 2.

In Table 1, the averages over the ten experiments for each circuit are shown. The overall average power reduction is about 13.7% with a 8.8% increase in delay. When a limit of 5% increase in delay is imposed on the transformed circuit, and power is traded off for delay, a 12.1% increase in delay and a 3.10% increase in area is obtained. When a limit of 1% is imposed on the increase in delay, an average 0.9% increase in delay, and a 2.9% increase in area with a 11.6% reduction in power was obtained. The efficacy of the proposed mechanism for trading off power for delay is demonstrated and is significant. While the increase in delay was reduced from 8.8% to 0.9% on the average, the reduction in power changed only slightly from 13.7% to 11.6%.

Table 2 shows the maximum power reduction over the ten experiments. Again a efficacy of trading off power and delay is demonstrated. An average of 38.3% maximum reduction in power is achieved with a 15.4% increase in delay under the *loose* timing constraint. After carrying out the power-delay tradeoff, a 31% reduction in power reduction with a 0.2% increase in delay can be achieved. Among all the experiments, the best case was a 67% reduction in power with a 3.9% decrease in delay. This improvement in the delay was due to the addition and deletion of the redundant connections.

Note that in both tables, there are results where the timing constraints are not satisfied. In the current implementation, only the removal of redundant connections is considered as a means of trading off power for delay. Many other transformations can also be incorporated to satisfy the time constraints. Finally, the delay information was obtained by performing static timing analysis which does not account for false paths. Therefore the actual delay may be less than the values shown in the tables.

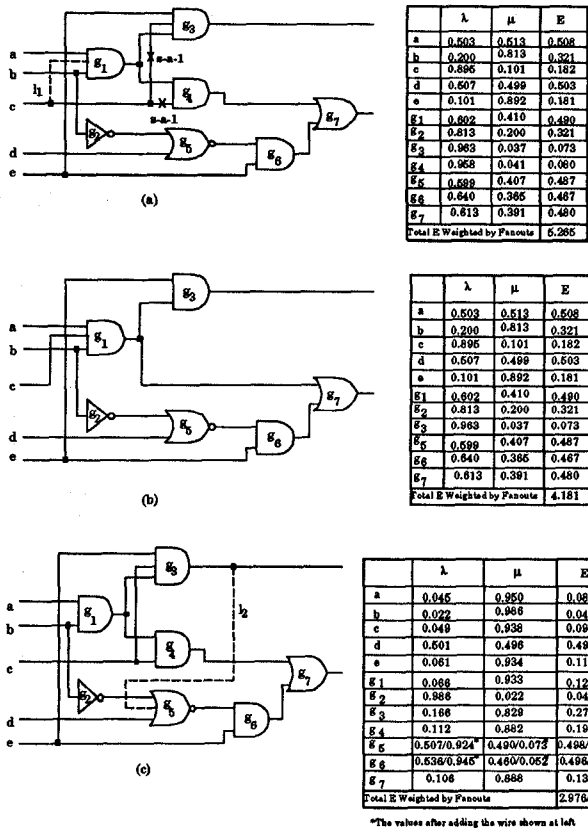


Figure 1: An example of re-wiring to reduce switching activity

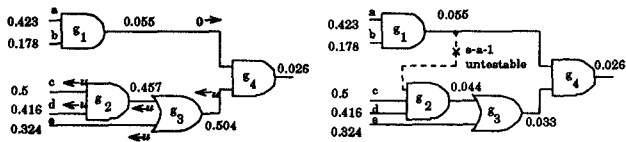


Figure 2: Single Source Implication

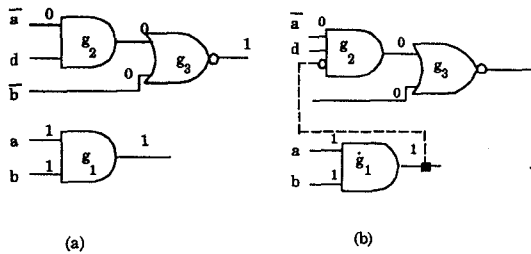


Figure 3: Useless redundant connections

```

procedure ssi () {
/* $R_s$  is the set of gates implied unobservable.*/
1.  $R_s = \emptyset$ ;
2. sort gates in increasing order of  $E$ ;
3. if ( $S$  is not empty) {
4. take a gate  $g_s$  with the smallest  $E$ ;
5. select a logic value  $v$  to imply;
6. forward/backward implication of  $v$  at  $g_s$ ;
7. back propagation of implied unobservabilities;
8. if (no unobservable gate can be found) goto 2;
9. else add found unobservable gates to  $R_s$ ; }
10. else stop;
12. return  $R_s$ ; }

```

Figure 4: Single-source implication algorithm.

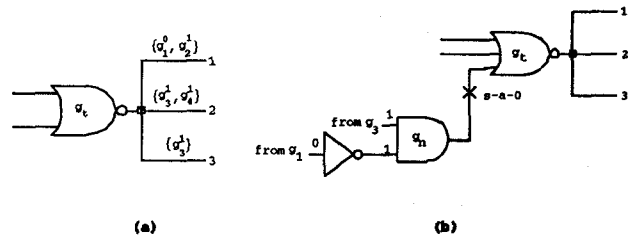


Figure 5: An example of Multiple Source Implication

```

procedure msi () {
/* $R_m$  is the set of gates implied unobservable.*/
1.  $R_m = \emptyset$ 
2. for (all gates  $g_i$  with fanout > 1 are not unobservable by single source implication) {
3. find a minimum cover of the implication sets  $I_k$ ;
4. forward/backward implication the logic values;
5. Backward propagation of unobservabilities;
6. add newly found unobservable gates to the  $R_m$ . }
7. return  $R_m$ ; }

```

Figure 6: Multiple-source implication

D.I = % increase in delay, A.I = % increase in area (gates), P.I = % decrease in switching activity

Circuit	PIs	POs	Gates	No Delay Constraint			5% Delay Constraint			1% Delay Constraint			Cpu
				D.I	A.I	P.I	D.I	A.I	P.I	D.I	A.I	P.I	
cm138a	135	99	970	2.28	6.72	20.18	2.28	6.72	20.18	2.28	6.72	20.18	2
cm85a	6	8	44	1.73	4.73	17.02	1.55	3.64	17.16	0.93	3.36	16.64	2
cu	11	3	70	22.66	10.30	28.07	3.63	1.82	22.16	-0.60	0.53	17.23	4
dalu	75	16	1152	10.07	9.20	11.84	2.30	3.35	9.57	1.95	3.34	9.56	103
duke2	22	29	578	4.58	6.33	21.10	1.53	3.54	18.01	0.50	3.46	18.04	75
frg2	143	139	1054	8.82	5.00	23.54	2.59	4.00	23.15	1.94	4.00	23.13	95
k2	45	42	1472	3.70	4.10	11.27	0.34	2.27	10.57	0.34	2.27	10.57	429
misex2	25	18	139	4.15	7.82	14.67	-2.76	3.10	13.55	-3.20	3.05	13.49	9
misex3	14	14	824	9.79	6.63	8.97	0.79	3.29	7.67	0.02	3.25	6.56	95.8
mux	21	1	63	5.87	4.29	12.60	0.00	1.24	4.93	0.00	1.24	4.93	2
pair	173	137	1967	2.59	1.29	2.43	1.01	0.67	2.34	0.35	0.49	2.33	138
pcler8	27	17	125	5.52	7.65	14.77	3.01	6.94	14.58	3.01	6.94	14.58	7
pm1	16	13	69	8.38	4.85	10.25	1.02	0.69	9.51	0.37	0.40	9.69	3
rot	135	103	814	2.95	1.78	3.43	1.89	1.09	3.26	0.14	0.97	3.13	38
sao2	10	4	162	5.7	5.79	11.57	0.83	3.48	10.77	0.03	3.30	10.64	8
t481	16	1	959	11.91	4.07	8.41	3.01	2.21	7.12	0.75	2.11	6.49	142
term1	34	10	202	19.67	9.82	26.86	31.4	7.31	21.27	0.11	6.92	19.79	9
vda	17	37	824	2.76	1.22	4.42	0.27	0.79	4.28	0.27	0.79	4.28	104
x1	51	35	418	14.08	5.14	10.30	3.02	2.59	9.25	2.36	2.51	9.14	22
x3	135	99	979	29.10	5.25	12.99	6.26	3.27	12.22	5.85	3.26	12.21	48
AVG	55.6	41.3	644.3	8.82	5.60	13.73	1.79	3.10	12.08	0.87	2.95	11.63	65

Table 1: Summary of the Average Power Reduction

D.I = % increase in delay, A.I = % increase in area (gates), P.I = % decrease in switching activity

Circuit	PIs	POs	Gates	No Delay Constraint			5% Delay Constraint			1% Delay Constraint			Cpu
				D.I	A.I	P.I	D.I	A.I	P.I	D.I	A.I	P.I	
cm138a	135	99	970	-3.91	9.38	67.43	-3.91	9.38	67.43	-3.91	9.38	67.43	3
cm85a	6	8	44	-1.22	8.18	66.88	-1.22	8.18	66.88	-1.22	8.18	66.88	3
cu	11	3	70	37.79	17.42	74.10	4.50	5.30	58.50	0.00	3.03	37.35	4
dalu	75	16	1152	7.71	10.58	21.02	1.79	4.04	20.54	1.79	4.04	20.54	107
duke2	22	29	578	7.30	16.46	53.89	1.17	6.12	51.00	0.37	6.01	50.76	84
frg2	143	139	1054	13.00	10.79	55.98	4.80	9.51	52.98	3.20	9.45	52.97	105
k2	45	42	1472	0.35	3.70	20.10	0.35	3.70	20.10	0.35	3.70	20.10	428
misex2	25	18	139	11.32	7.41	23.99	-7.27	3.70	21.83	-7.27	3.70	21.83	10
misex3	14	14	824	27.74	14.95	29.28	4.60	7.33	23.62	0.00	7.11	22.92	120
mux	21	1	63	28.34	19.05	54.26	0.00	3.81	8.70	0.00	3.81	8.70	5
pair	173	137	1967	13.28	1.96	4.33	3.61	1.14	3.84	0.00	0.98	3.63	135
pcler8	27	17	125	13.64	20.92	60.32	10.23	20.41	59.23	10.23	20.41	59.23	8
pm1	16	13	69	14.08	10.89	19.91	0.00	2.97	16.56	0.00	2.97	16.56	3
rot	135	103	814	5.48	21.3	5.61	2.84	1.29	5.00	-1.43	1.22	4.96	47
sao2	10	4	162	10.11	18.32	45.22	4.04	17.22	40.87	0.00	16.48	39.88	9
t481	16	1	959	35.39	13.46	27.21	4.96	3.03	11.96	0.65	2.90	11.74	167
term1	34	10	202	27.36	16.92	57.55	3.42	13.29	40.77	0.85	12.99	40.77	8
vda	17	37	824	0.00	1.49	7.94	0.00	1.49	7.94	0.00	1.49	7.94	116
x1	51	35	418	17.98	12.63	16.57	1.38	3.91	14.68	1.38	3.91	14.68	31
x3	135	99	979	42.70	15.33	53.66	1.66	11.98	52.87	-0.48	11.92	52.87	81
AVG	55.6	41.3	644.3	15.42	11.60	38.26	1.85	6.89	32.27	0.23	6.68	31.09	74

Table 2: Summary of the Maximum Power Reduction

## 7 Conclusions

In this paper we proposed an implication based logic resynthesis approach for low power application. The experimental results show that significant power reduction can be achieved with a low area and computational cost. We also described how tradeoffs between power and delay can be carried out after the transformations. These methods provide a great degree of freedom to include a variety of optimization criteria and to examine tradeoffs among them.

Work to significantly improve these methods is currently in progress. This includes (1) improvement of the post-transformation power delay trade-off by using fanout trees, signal buffering, and other techniques, (2) solving the set covering problem with different optimization criteria other than minimum cardinality, e.g., finding a cover that will result in the least switching activity, (3) extension of the methods to include more complex gates, e.g. AOI gates.

## 8 Acknowledgements

This work was supported by a grant from the Motorola Semiconductor Products Sector. We gratefully acknowledge their support. In addition, we wish to thank Dr. Gary Yeap and Mr. Hong-Yu Xie with the Advanced Design Technology, at Motorola in Tempe, Arizona and Dr. Shantanu Ganguly at Motorola (Somerset) in Austin, Texas, for their invaluable support on this work.

## References

- [1] A.P. Chandrakasan, S.S. Scheng, and R.W. Broderon. "Low Power CMOS Digital Design." *IEEE Journal of Solid State Circuits*, 27(4), April 1992, pp. 473-483.
- [2] S.C. Chang, M. M. Sadowska. "Perturb and Simplify: Multi-Level Boolean Network Optimizer." *Proceedings of ICCAD*, 1994, pp.2-5
- [3] K.T. Cheng, L. Entrena. "Multi-Level Logic Optimization By Redundancy Addition and Removal." *Proceedings of ICCAD*, 1993, pp. 373-377.
- [4] A. Shen, S. Devadas, A. Ghosh, and K. Keutzer. "On Average Power Dissipation and Random Pattern Testability of Combinational Logic Circuits." *Proceedings of ICCAD*, 1992, pp. 402-407.
- [5] R.K. Brayton, G.D. Hachtel, C.T. McMullen, and A. Sangiovanni-Vincentelli. *Logic Minimization Algorithms for VLSI Synthesis*. Boston, Massachusetts: Kluwer Academic Publisher, 1984.
- [6] B. Rohfleisch, A. Kolbl, and B. Wurth "Reducing Power Dissipation after Technology Mapping by Structural Transformations" *Proceedings of DAC*, 1996, pp.
- [7] S. Iman, M. Pedram. "Multi-Level Network Optimization for Low Power." *Proceedings of ICCAD*, 1994, pp. 372-377.
- [8] S. Iman, M. Pedram. "Two-Level Logic Minimization for Low Power." *Proceedings of ICCAD*, 1995, pp. 433-438.
- [9] R. I. Bahar, F. Somenzi. "Boolean Techniques for Low Power Driven Re-Synthesis." *Proceedings of ICCAD*, 1995, pp. 428-432.
- [10] M.A. Iyer, M. Abramovici. "Low-Cost Redundancy Identification for Combinational Circuits." *Proceedings of the 7th International Conference on VLSI Design*, 1994, pp.315-318.
- [11] R.S. Martin, J.P. Knight "Power-Profiler: Optimizing ASICs Power Consumption at the Behavioral Level" *Proceedings of DAC*, 1995, pp. 42-47.
- [12] E. Sentovich, K. Singh, C. Moon, H. Savoj, R. Brayton, and A. Sangiovanni-Vincentelli. "Sequential circuit Design Using Synthesis and Optimization." *Proc. of ICCD*, Oct. 1992. pp. 328-333.
- [13] F. N. Najam. "A Survey of Power Estimation Techniques in VLSI Circuits." *IEEE Trans. on VLSI Systems*, December 1994, pp. 446-454.
- [14] V. Tiwari, S. Malik, P. Ashar. "Guarded Evaluation: Pushing Power Management to Logic Synthesis/Design" *Proceedings of International Symposium on Low Power Design*, 1995, pp. 221-226.
- [15] C. Tsui, M. Pedram, and A. Despain. "Technology Decomposition and Mapping Targeting Lower Power Dissipation." *Proceedings of IEEE 30th DAC*, p. 68-73, 1993
- [16] S. B. K. Vrudhula, H. Y. Xie. "Techniques for CMOS Power Estimation and Logic Synthesis for Low Power." *Proceedings of International Workshop on Low Power Design*, 1994, pp. 21-26.
- [17] S. B. K. Vrudhula, H. Y. Xie. "A Fast and Accurate Technique for Estimating Signal Activity in CMOS Logic Circuits." *Technical Report CENG-95-109*, Dept. of ECE, University of Arizona, 1994.