Optimal Low Power XOR Gate Decomposition

Hai Zhou[1], D. F. Wong[2]
[1] Advanced Technology Group, Synopsys Inc., Mountain View, CA 94043
[2] Department of Computer Sciences, University of Texas, Austin, TX 78712

Abstract

With the remarkable growth of portable application and the increasing frequency and integration density, power is being given comparable weight to speed and area in IC designs. For the problem of low power decomposition of an XOR gate, if the implementation technology is static CMOS logic, previous research gave an $O(n \log n)$ time algorithm which assumes that the inputs have both polarities available. But that approach can not be used in dynamic logic. In this paper, we analyze the properties of optimal XOR decompositions in dynamic logic. Based on these optimality properties, we design an optimal algorithm to solve the low power XOR decomposition problem in dynamic logic. We also point out that the previous solution for static logic is not optimal, and give an optimal algorithm which does not even change the input polarities.

1 Introduction

In the past, the major concerns of VLSI design were performance, area and reliability; power considerations were of only secondary importance. However, this has begun to change recently, and power is being given comparable weight to speed and area. Two factors mainly contribute to this trend. On one hand is the remarkable success and growth of portable applications which demand high-speed computation and complex functionality with low power consumption. Without low-power design techniques, future portable devices will suffer from either a very short battery life or a very heavy battery pack. On the other hand, power dissipation in high-end products rises with increased integration density and circuit speed. The resulting heat will increase the packaging and cooling costs and shorten the life of ICs. Therefore, low-power techniques are also needed.

In current technology, power dissipation in digital CMOS circuits is dominated by the *dynamic dissipation*, which is mainly the charging and discharging of the load capacitances [4]. In a circuit consisting of n gates, it can be modeled as

$$P = 0.5V_{dd}^2 f_{clk} \sum_{i=1}^{n} C_L^i E_{sw}^i$$

where V_{dd} is the supply voltage, f_{clk} is the clock frequency, C_L^i is the load capacitance at the output of gate i, and E_{sw}^i (referred to as *switching activity*) is the expected number of gate i transitions per clock cycle. As we can see, V_{dd} and f_{clk} are fixed by the technology, but C_L^i and E_{sw}^i can be controlled in design process, usually E_{sw}^i in logic synthesis and C_L^i in physical design.

Most previous efforts in power reduction have focused on reducing power consumption in AND/OR or NAND/NOR based circuits [8, 10, 11, 12]. However, there are major classes of circuits which have sizable AND/OR representations, but have very compact AND/XOR representations [6, 9]. Standard examples are arithmetic functions such as adders, multipliers, and error-correcting codes [5, 9]. Recently, there has been some success in achieving area reduction by employing optimization techniques specifically targeted towards initial AND/XOR representations in the well known Fixed Polarity Reed Muller (FPRM) form [9].

For the low power AND/OR gate decomposition, if the implementation technology is dynamic logic, it can be solved by Huffman's algorithm in $O(n \log n)$ time [10]. But it is more complicated in static logic. The present best exact algorithm for static logic runs in $O(n2^n)$ time and a very good heuristic algorithm can be derived from it [12]. For low power XOR gate decomposition, to the best of our acknowledgment, the only result in the literature is an algorithm by Narayanan and Liu [1]. Their algorithm is for static logic and requires that each input has both polarities available. But, as we will show later, it is not optimal.

This paper has two contributions to the low power XOR gate decomposition. The first is an optimal algorithm for dynamic logic. For this problem, we first study the structure of an optimal decomposition tree. This is given by a set of necessary properties an optimal tree must have. Then, based on these properties, we design an algorithm to construct an optimal decomposition tree. The time complexity of the algorithm is $O(n \log n)$.

The second contribution is another optimal algorithm for low power XOR gate decomposition in static logic. We find that Narayanan and Liu's algorithm [1], which requires both polarities for inputs, is **not** optimal. Based on the properties of decompositions in static logic, we design an $O(n \log n)$ time optimal algorithm which does not even require both polarities available for the inputs.

The rest of the paper is organized as follows. In section 2, we define the low-power XOR gate decomposition problems, both in dynamic logic and static logic. Section 3 presents the solution for dynamic logic. Section 4 points out the problems in Narayanan and Liu's algorithm [1] and presents the optimal algorithm for static logic. Finally, some experimental results are reported in Section 5. Due to page limit, most of the proofs are omitted.

2 Problem formulation

Technology decomposition is a step before technology mapping which decomposes a network with multiple input gates into a network with only two input gates. In XOR gate decomposition, we need to decompose a multi input XOR gate into a tree of two input XOR gates.

We treat each signal in a circuit as a random variable and define the *signal probability* of a signal x as the probability of x being 1. For simplification, in the sequel, we use the same symbol to represent both a signal and its probability. Its meaning can be distinguished from the context. As in [1, 8, 10, 11, 12], we assume the *zero delay model* where gate delays are assumed to be zero and thus signal transitions due to glitching are ignored. We also assume the inputs of the XOR gate are uncorrelated (*spatial independent*) and the present input signal value is independent of those in the past (*temporal independent*). Under these assumptions, given input signal probabilities and a decomposition tree, the probabilities of internal signals can be computed as follows. Start from the primary inputs, for each gate $z = x$ XOR y, the probability $z = x + y - 2xy$. For example, Figure 1 shows one XOR gate decomposition and the signal probabilities of all nodes.

Given the probability, the switching activity of a signal in a circuit really depends on the implementation logic style. If implemented in dynamic logic styles, in each clock cycle, a gate output is first pre-charged to 0 (pre-charged to 1), and then evaluated to the result value. Therefore, if the output probability is x, the switching activity will be

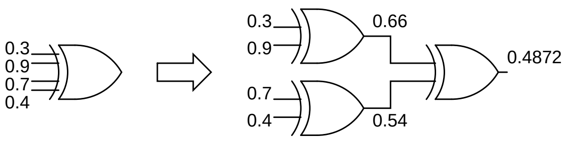

Figure 1: XOR decomposition

$2x$ $(2(1-x))$, where one x (one $1-x$) comes from the pre-charging and one x (one $1-x$) comes from the evaluation.

In static logic, under the temporal independence assumption, the switching activity E_{sw} of signal x can be written as

$$\begin{aligned} E_{sw}(x) &= Pr[x: 0 \to 1] + Pr[x: 1 \to 0] \\ &= Pr[x=0]Pr[x=1] + Pr[x=1]Pr[x=0] \\ &= 2Pr[x=1]Pr[x=0] \\ &= 2x(1-x) \end{aligned}$$

Plotting the switching activity as a function of signal probability both in dynamic logic and static logic, we have Figure 2. As we can see, if considered only in term of switching activity, static logic is always better than dynamic logics. But we can not come to the same conclusion in terms of power dissipation, since dynamic logics usually have smaller load capacitances.

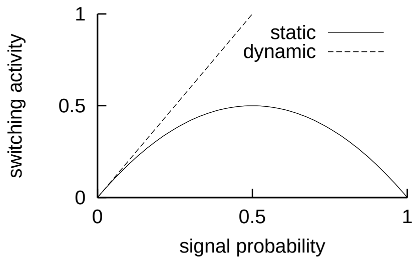

Figure 2: Relations between switching activity and signal probability

Based on the above discussion, the problem of low power XOR gate decomposition in dynamic logic and static logic, respectively, can be defined as follows.

Low power XOR *gate decomposition problem – Dynamic logic*: Given an n-input XOR gate with inputs $I = \{s_1, s_2, \ldots, s_n\}$, construct a tree $T = (V, E)$ of 2-input XOR gates with $s_1, s_2, \ldots, s_n$ as its leaves such that

$$E_{sw}^{dynamic}(T) = \sum_{v \in V-I} 2v$$

is minimized.

Low power XOR *gate decomposition problem – Static logic*: Given an n-input XOR gate with inputs $I = \{s_1, s_2, \ldots, s_n\}$, construct a tree $T = (V, E)$ of 2-input XOR gates with $s_1, s_2, \ldots, s_n$ as its leaves such that

$$E_{sw}^{static}(T) = \sum_{v \in V-I} 2v(1-v)$$

is minimized.

3 Decomposition for dynamic logic

In this section, we solve the low power XOR gate decomposition in dynamic logic. First, the structures of optimal decompositions are analyzed and the optimality properties are given in Section 3.1. Then, based on these properties, an optimal algorithm is designed and analyzed in Section 3.2.

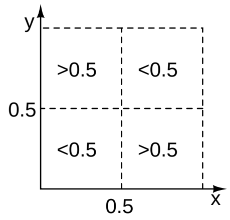

Figure 3: Output probability as function of input probabilities

3.1 Optimality properties

Given two input signals with probabilities x and y, the probability of the output of an XOR gate is given as $z = f(x, y) = x + y - 2xy$. The partial derivatives of f are given as follows.

$$\frac{\partial f}{\partial x} = 1 - 2y, \qquad \frac{\partial f}{\partial y} = 1 - 2x$$

This means that, when $x < 0.5$ ($y < 0.5$), the function f is strictly increasing with y (x); when $x > 0.5$ ($y > 0.5$), it is strictly decreasing with y (x). Combined with the fact that $f(0.5, y) = f(x, 0.5) = 0.5$, the values of f can be characterized by Figure 3.

The truth table of the operation XOR is shown in Table 1. Comparing Figure 3 with Table 1, we have the following lemma.

Table 1: Truth table of XOR

XOR	0	1
0	0	1
1	1	0

Lemma 1 *The probability of a node is equal to* 0.5 *if and only if the subtree rooted at the node has at least one node whose probability is* 0.5*; otherwise, the probability is greater than* 0.5 *if and only if any subtree rooted at the node has odd numbers of leaves whose probabilities are greater than* 0.5.

Based on the above lemma, we have the following theorem.

Theorem 1 *In an optimal* XOR *gate decomposition, except the root, no internal node could have probability greater than* 0.5.

This means that, except at the root, inputs of probabilities greater than 0.5 can not be combined with probabilities smaller than 0.5. We can also prove that, no of them, except one, can be combined with probability 0.5. This is stated formally as the following theorem.

Theorem 2 *In an optimal* XOR *gate decomposition, there is at most one input of probability greater than* 0.5 *which can be combined with probability* 0.5.

Combining the results from Theorem 1 and Theorem 2, we have the following result.

Corollary 2.1 *In an optimal* XOR *gate decomposition, all inputs of probabilities greater than* 0.5 *must be combined pairwisely, except one when there are odd numbers of them, which is either a child of the root or combined with a probability of* 0.5.

By combining inputs of probabilities greater than 0.5 pairwisely, we get signals whose probabilities are smaller than 0.5. When there are odd numbers of them, the one left can be taken care of easily. Therefore, we are facing a problem of low power XOR gate decomposition where no input has probability greater than 0.5. We will show that this problem can be solved by Huffman's algorithm. That is, each time, the two signals with the smallest probabilities are combined until there is only one signal. We call the tree constructed by Huffman's algorithm Huffman tree.

If we use $w(T) = [v_1, v_2, \cdots, v_n]$ to represent the non-decreasing sequence of probabilities on internal nodes of a tree T, then that means, for Huffman tree T, we have

$$\sum_{i=1}^{n} w(T)_i \leq \sum_{i=1}^{n} w(S)_i$$

for any tree S. In fact, we can prove a much stronger result which will also be used in Section 4. First, we will define a binary relation "$\preceq$" on probability sequences.

Definition 1 *We say $w(T) \preceq w(S)$ if for all h such that $1 \leq h \leq n$, we have*

$$\sum_{i=1}^{h} w(T)_i \leq \sum_{i=1}^{h} w(S)_i.$$

As we can check, the relation $\preceq$ is reflexive, anti-symmetric, and transitive, thus is a partial order. The following theorem shows that the Huffman tree has the minimum probability sequence with respect to $\preceq$.

Theorem 3 *If no input probability is greater than 0.5 and T is a Huffman tree, we have $w(T) \preceq w(S)$ for any tree S.*

Now the only remaining question is how we can pair-up the input probabilities which are greater than 0.5. As we already know, when $x > 0.5$, the function $f(x, y) = x + y - 2xy$ is monotonically deceasing with y. Thus, combining the largest two probabilities will give the smallest probability. But this will leave the remaining probabilities larger. However, as we can prove in the next theorem, it actually gives the optimal solution.

Theorem 4 *In an optimal* XOR *gate decomposition, the probabilities greater than 0.5 are paired up from largest to smallest.*

3.2 Algorithm

Based on the properties shown in the previous section, we have the following algorithm to solve the low power XOR gate decomposition problem in dynamic logic. First, sort the input probabilities which are greater than 0.5. Then, combine them pairwisely from the largest to the smallest. Finally, iteratively combine the two signals with the smallest probabilities until there is one signal.

As we already know, a signal with probability 0.5 is special since it will always produce probability 0.5 no matter which signal it combines with. Thus, if we consider a subtree whose internal nodes all have probability 0.5, we can change it to any tree topology without increasing the switching activities. This gives us the flexibility to optimize other parameters without compromising on power dissipation.

One of such possibilities is the delay optimization. It works as follows. When the combination iterations in the above algorithm come to a point when there are at most one probability which are smaller than 0.5, the process is stopped. The arrival time of each current signal is then computed. And the remaining combinations are based on arrival times to give a root with minimum arrival time. That is, the two signals with the smallest arrival times are combined iteratively. But we must take care of one special case when none of the two earliest signals has probability of 0.5 (this could happen when one is smaller than 0.5 and the other is greater than 0.5). In this case, the earliest signal will instead be combined with the third earliest signal which has probability 0.5.

4 Decomposition for static logic

In the previous section, we solved the lower power XOR gate decomposition in dynamic logic. In this section, we will solve the problem in static logic.

Narayanan and Liu [1] formulated the problem of lower power XOR gate decomposition in static logic and provided the following algorithm to solve it: *Let k be the number of primary inputs to the* XOR *tree with probabilities greater than 0.5. If k is even, replace all k inputs by their respective complements. If k is odd, replace any $k-1$ inputs by their respective complements. Then simply use the Huffman algorithm to build the optimal tree.*

We observe that the above algorithm is **not** optimal. This can be shown by a simple example in Figure 4. The problem has three inputs: $0.1, 0.5$, and 0.9. Since there is only one signal whose probability is greater than 0.5, by Narayanan and Liu's algorithm, no signal will be changed to its complement. Thus, as shown in Figure 4(a), probabilities 0.1 and 0.5 are first combined, and then combined with 0.9. The switching activities on the two internal nodes is $2 \cdot 0.5(1-0.5)+2 \cdot 0.5(1-0.5) = 1$. But if we first combine 0.1 and 0.9, then 0.5, as shown in Figure 4(b), the corresponding switching activities are $2 \cdot 0.82(1 - 0.82) + 2 \cdot 0.5(1 - 0.5) = 0.6476$, which is smaller.

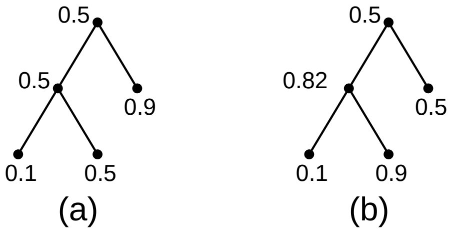

Figure 4: (a) Solution given by Narayanan and Liu (b) A solution with smaller switching activities

To design an optimal algorithm for the low power XOR gate decomposition in static logic, we need to analyze the properties of the decompositions. We start with the following observation by Narayanan and Liu [1].

Theorem 5 (Narayanan and Liu) *Suppose we have an* XOR *tree T consisting of n inputs. If we replace any subset of the n inputs by their complement, the switching activities in the tree remain unchanged.*

This result comes from two facts. One is the property of XOR gate: complementing one fanin will complement the output. The other is the property of static logic: a signal has the same switching activity as its complement.

Therefore, we will first solve the decomposition problem for a special case when no input probability is greater than 0.5. As in [3], we give the following definition.

Definition 2 *A function $G : U^n \to R$ is* Schur concave *if*

$$(x_i - x_j)(\frac{\partial G}{\partial x_i} - \frac{\partial G}{\partial x_j}) \leq 0 \textit{ for all } x_i, x_j \in U, 1 \leq i, j \leq n.$$

Schur [7] and Ostrowski [2] gave the following result which characterizes the relation between a function being Schur concave and the result of applying it on a sequence of weights.

Theorem 6 (Schur and Ostrowski) *Suppose G is increasing, then $G(\mathbf{a}) \leq G(\mathbf{b})$ for all sequences $\mathbf{a} \preceq \mathbf{b}$ if and only if G is Schur concave.*

In fact, we can easily check that the cost function $E_{sw}^{static}(T) = \sum_{v \in T} 2v(1-v)$ for XOR gate decomposition in static logic is Schur

Table 2: Experimental results in static logic

#in	MinHuffman			MaxHuffman			Narayanan-Liu			Ours
	#bad	MaxR	AvgR	#bad	MaxR	AvgR	#bad	MaxR	AvgR	#bad
3	736	81.74%	7.98%	750	75.70%	7.89%	384	81.74%	4.36%	0
4	871	93.11%	10.29%	993	117.04%	10.96%	436	64.23%	3.85%	0
5	938	93.41%	9.99%	1000	100.23%	11.65%	473	43.89%	3.48%	0
6	967	104.65%	10.97%	1000	91.26%	12.15%	486	38.11%	3.77%	0
7	988	59.71%	10.33%	1000	107.18%	12.24%	500	26.47%	3.13%	0
8	995	105.01%	10.26%	1000	72.15%	11.99%	496	24.58%	3.03%	0
9	997	57.61%	10.10%	1000	76.03%	11.72%	464	18.24%	2.66%	0
10	998	53.65%	10.30%	1000	70.08%	11.75%	524	17.20%	2.90%	0
11	999	56.84%	9.49%	1000	52.00%	11.19%	481	14.80%	2.38%	0
12	1000	53.32%	9.43%	1000	50.99%	11.25%	484	13.57%	2.33%	0

concave and increasing when no probability is greater than 0.5. Therefore, by combining Theorem 3 and Theorem 6, we have the following result.

Theorem 7 *If no input probability is greater than* 0.5*, the Huffman algorithm gives an optimal solution to the low power* XOR *gate decomposition in static logic.*

For the general problem with arbitrary input probabilities, we also have an optimal algorithm which does not even require both polarities for the inputs. It is stated as the following theorem.

Theorem 8 *For each input probability p, use* $\min\{p, 1-p\}$ *as its weight. Then the Huffman algorithm with the merging function* $M(x, y) = x + y - 2xy$ *gives an optimal* XOR *gate decomposition in static logic.*

Proof: Let T be the decomposition given in the theorem and T' be the same decomposition except the inputs whose probabilities are greater than 0.5 are changed to their complements. From the construction, we know that T' is a Huffman tree. If T is not optimal, then there exists a decomposition S such that $E_{sw}^{static}(S) < E_{sw}^{static}(T)$. We can derive S' from S by changing the inputs of probabilities greater than 0.5 to their complements. From Theorem 5, we know that $E_{sw}^{static}(T') = E_{sw}^{static}(T)$ and $E_{sw}^{static}(S') = E_{sw}^{static}(S)$. Thus, $E_{sw}^{static}(S') < E_{sw}^{static}(T')$. This is a contradiction to Theorem 7. $\square$

Same as in dynamic logic, the property of probability 0.5 can be used to optimize other parameters such as delay without compromising on power dissipation. In this case, we will stop when there is at most one non-0.5 probability and use the Huffman algorithm on arrival times to minimize the arrival time of the root.

5 Experimental results

We implement the two optimal algorithms for low power XOR gate decomposition in dynamic logic and static logic. Both algorithms have the time complexity of $O(n \log n)$. In fact, when running on a PC with 166MHz CPU and 64M memory, decomposing 1000 gates each with 100 inputs only takes about 4 seconds.

In order to measure the difference between the optimal algorithms and other simple heuristics, we also implement the Huffman algorithm. It comes in two versions. In one version which we called *MinHuffman*, two signals with the smallest probabilities are combined iteratively. In other version which we called *MaxHuffman*, on the contrary, two signals with the largest probabilities are combined iteratively. In static logic, we also compare with the algorithm given by Narayanan and Liu [1].

On each input size from 3 to 12, we randomly generated 1000 sets of input probabilities. We run *MinHuffman*, *MaxHuffman*, Narayanan and Liu's algorithm, and the two optimal algorithms on each of them. For each heuristic algorithm, the number of non-optimal solutions is counted. Given a problem I, let $Opt(I)$ represent the optimal solution, we use the ratio

$$R = \frac{S(I) - Opt(I)}{Opt(I)}$$

to measure the non-optimality of solution $S(I)$. For each heuristic algorithm, among all 1000 problems of the same input size, the maximum ratio *MaxR* and the average ratio *AvgR* are computed. The results in static logic are given in Table 2, where "#bad" represents the number of non-optimal solutions. In dynamic logic, *MinHuffman* and *MaxHuffman* heuristics have similar bad performances.

References

[1] U. Narayanan and C. L. Liu. Low power logic synthesis for XOR based circuits. In *ICCAD*, pages 570–574, 1997.

[2] A. Ostrowski. Sur quelques applications des fonctions convexes et concaves au sens de I. Schur (offert en homage á P. Montel). *J. Math. Pures Appl.*, 31:253–292, 1952. (In French).

[3] D. S. Parker. Conditions for optimality of the Huffman algorithm. *siam-computing*, 9(3):470–489, August 1980.

[4] M. Pedram. Power minimization in IC design: Principles and application. *ACM TODAES*, January 1996.

[5] I. S. Reed. A class of multiple-error-correcting codes and the decoding scheme. *IRE Transactions on Information Theory*, PGIT-4:38–49, 1954.

[6] T. Sasao. *Logic Synthesis and Optimization*. Kluwer Academic Publishers, 1983.

[7] I. Schur. Über eine klass von mittelbildungen mit anwendungen auf die determinantentheorie. *Sitzungsber. Berl. Math. Ges.*, 22:9–20, 1923. (In German).

[8] Vivek Tiwari, Pranav Ashar, and Sharad Malik. Technology mapping for low power. In *DAC*, 1993.

[9] C.-C. Tsai and M. Marek-Sadowska. Multilevel logic synthesis for arithmetic functions. In *DAC*, pages 68–73, 1996.

[10] C.-Y. Tsui, M. Pedram, and A. M. Despain. Technology decomposition and mapping targeting low power dissipation. In *DAC*, 1993.

[11] C.-Y. Tsui, M. Pedram, and A. M. Despain. Power efficient technology decomposition and mapping under an extended power consumption model. *IEEE TCAD*, pages 1110–1122, 1994.

[12] H. Zhou and D. F. Wong. An exact gate decomposition algorithm for low-power technology mapping. In *ICCAD*, San Jose, CA, 1997.