Low Power Logic Synthesis for XOR Based Circuits *

Unni Narayanan C. L. Liu

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

Abstract

An abundance of research efforts in low power logic synthesis have so far been focused on AND/OR *or* NAND/NOR *based logic. A typical approach is to first generate an initial multi-level* AND/OR *or* NAND/NOR *representation of a boolean function. Next, the representation is optimized in terms of power. However, there are major classes of circuits such as arithmetic functions which have sizable* AND/OR *representations but have very compact* AND/XOR *representations. For these functions* AND/OR *based optimization approach often yields poor results. In this paper, we put forth a paradigm for low power logic synthesis based on* AND/XOR *representations of boolean functions. Specifically, we propose transforming a boolean function into a Fixed Polarity Reed Muller form that allows us to efficiently synthesize* XOR *trees and* AND *trees with provably minimum switching activity. Preliminary experimental results show that we attain good power savings with negligible area overhead and often area reduction when compared to conventional* AND/XOR *based synthesis methods and the Berkeley SIS system.*

1 Introduction

The advent of portable digital devices such as laptop personal computers has made low power circuit design an increasingly important research area. For example, laptop computers have limited battery life, and so the circuitry in the computer must be designed to dissipate as little power as possible without sacrificing performance in terms of speed. Additionally, high power consumption increases the cost of handling heat dissipation and diminishes the reliability of today's increasingly complex circuits with higher transistor counts and faster clock rates. It is well known that in CMOS technology a large portion of power dissipation on chip is due to dynamic power consumption at the gates which is computed according to the formula:

$$\sum_{i=1}^{N} \frac{1}{2} C_i V_{dd}^2 f_i$$

where C_i is the output capacitance of the ith gate, V_{dd} is the supply voltage, f_i is the frequency of transitions at the output of the ith gate, and N is the total number of gates on the chip. Clearly, a reduction in f_i will lead to a corresponding reduction in the total power consumption of the circuit. We can compute f_i as $2P_i(1-P_i)$ where P_i is the probability that the output of the ith gate is high.

Till now most efforts in power reduction have focused on reducing the power consumption in AND/OR or NAND/NOR based circuits. The typical approach is to first generate an initial AND/OR or NAND/NOR representation of a boolean function. Next, the representation is optimized in terms of switching activity through a variety of well known techniques [5]. For example, past research has encompassed low power technology decomposition of AND and OR gates [4, 12, 3] as well as low power technology mapping of combinational circuits implemented in terms of AND and OR gates [9]. However, there are major classes of circuits which have sizable AND/OR representations, but have very compact AND/XOR representations [11, 8]. Arithmetic functions such as adders, multipliers, and error correcting codes are some standard examples [7, 11]. For these functions AND/OR based optimizations often yield poor results. 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 [11]. In this paper, we propose techniques for low power logic synthesis of AND/XOR based circuits. We are able to efficiently synthesize a power optimal multi-level representation of a circuit in the sense that each XOR tree, AND tree, and OR tree in the circuit consumes prov-

*This research was supported in part by the National Science Foundation (NSF) under grant MIP-96-12184 and a research grant from IBM.

ably minimum power under the zero-delay model with the temporal and spatial independence assumptions. We believe that this approach will become increasingly important in the synthesis of a large class of boolean functions because the area cost of XOR gates implemented in CMOS technology is decreasing [13].

An FPRM form for a boolean function f is simply a representation of f in which each variable of the function has been expanded with either the positive Davio expansion or the negative Davio expansion [1, 11]. An FPRM representation of f is an XOR of products representation in which each variable is either complemented or uncomplemented throughout the whole expression. For example, consider the function:

$$f = \overline{x_1}\,x_2 \oplus \overline{x_1}\,x_3 \oplus \overline{x_1}\,x_9 \oplus x_4\,x_5\,x_6 \oplus x_7 \oplus x_8 \quad (1)$$

Observe that x_1 appears only in complemented form and x_2 appears only in uncomplemented form. The same is true for the remaining variables in the expression. We can associate with f an nine bit polarity vector $(0, 1, 1, 1, 1, 1, 1, 1, 1)$ which describes whether a particular variable is complemented or not. The ith entry in the polarity vector is 0 if x_i appears in complemented form and is 1 if x_i appears in uncomplemented form. An alternative representation of the function f in (1) is :

$$f = \overline{x_1}\,\overline{x_2} \oplus \overline{x_1}\,\overline{x_3} \oplus \overline{x_1}\,x_9 \oplus x_4\,x_5\,x_6 \oplus \overline{x_7} \oplus \overline{x_8} \quad (2)$$

In this case the corresponding polarity vector is $(0, 0, 0, 1, 1, 1, 0, 0, 1)$. It should be noted that given any boolean function and any polarity vector we can *always* efficiently retrieve the corresponding FPRM form from Ordered Function Decision Diagrams (OFDD) which are based upon the Davio expansions [11, 10, 2]. However, the size of the FPRM representations for the same function may be different. Figure 1 shows the respective multi-level implementations, A and B, corresponding to the FPRM forms in (1) and (2). When we take into consideration power consumption, the two FPRM representations differ greatly. For example, suppose that the on probabilities for the primary inputs are: $P_{x_1} = .90, P_{x_2} = .99, P_{x_3} = .99, P_{x_4} = .40, P_{x_5} = .10, P_{x_6} = .20, P_{x_7} = .90, P_{x_8} = .50$, and $P_{x_9} = .50$. If we realistically assume that we have both the primary inputs and complements of the primary inputs available to us and we account for internal correlations then the expected number of transitions or power consumption for Synthesis A is 2.6. The expected number of transitions for Synthesis B is 1.3. Hence, there is a 50% difference in the power consumption between the two multi-level realizations.

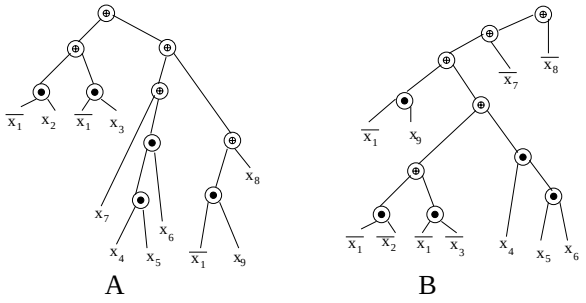

Figure 1: Respective multi-level implementations of Expression 1 and Expression 2

In this paper we study the relationship between the polarity vector, structure, and power consumption of XOR based logic and put forth a paradigm for low power synthesis of such circuits. In Section 2 we present our main theoretical results on low power logic synthesis based upon FPRM forms. In Section 3 we illustrate how the effects of the low power synthesis are preserved after factorization and reduction rules are applied to reduce the area of the circuit. In Section 4 we present experimental results. In Section 5 we summarize our results and present future directions for this research.

2 Theoretical Results

In this section we demonstrate how to choose a polarity vector (and hence the corresponding FPRM form) such that we can efficiently synthesize every XOR tree, AND tree, and OR tree in the circuit with provably minimum power dissipation. Suppose we have an XOR gate g with inputs a and b. Then the on probability at the output of the gate denoted by P_g is $P_a + P_b - 2P_aP_b$ where P_a and P_b are the on probabilities of a and b. The power consumed by g is $2P_g(1-P_g)$. If we assume that both the primary inputs and their complements are available then there are some interesting consequences:

Theorem 2.1 *Suppose we have an* XOR *tree T consisting of n inputs. If we replace any subset of the n inputs by their complement the power dissipation in the tree remains unchanged under the zero delay model.*

Observe that if we complement an even number of inputs to an XOR tree then the functionality of the tree

is preserved. We define the notion of a power vector as follows:

Definition 2.1 *Given a set S of n data signals $s_1, \ldots, s_n$ we define the power vector $V_s = V_{s_1}, \ldots, V_{s_n}$ to be an n-bit vector such that $V_{s_i} = 0$ if $P_{s_i} \leq 1/2$ and $V_{s_i} = 1$ if $P_{s_i} > 1/2$.*

Definition 2.2 *A power vector $V_s = (V_{s_1}, \ldots, V_{s_n})$ is said to be low signal biased (LSB) if at most one V_{s_i} is equal to 1. Similarly, V_s is said to be high signal based (HSB) if at most one V_{s_i} is equal to 0. We call an* XOR *tree whose primary inputs have an LSB power vector an LSB* XOR *tree. Similarly, we call an* XOR *tree whose primary inputs have an HSB power vector an HSB* XOR *tree.*

XOR trees have the following interesting property:

Theorem 2.2 *Suppose we have an* XOR *tree T that implements a boolean function f and consumes power P with respect to an arbitrary power vector V. Then, we can always transform T into a tree T' that realizes f with power vector V' such that $P' \leq P$ and V' is LSB or HSB.*

Theorem 2.2 means that we never increase the power dissipation in a tree by transforming it into a new tree whose inputs have an LSB power vector. The result is significant because of the following theorem:

Theorem 2.3 (LSB/HSB Optimality Theorem) *If we have n data signals which have an LSB power vector then the Huffman algorithm synthesizes an* XOR *tree that consumes the minimum power under the zero delay model. Furthermore, if the n data signals have an HSB power vector then the Anti-Huffman algorithm synthesizes an* XOR *tree that consumes the minimum power under the zero delay model.*

Theorem 2.2 together with Theorem 2.3 suggest an efficient polynomial time algorithm for synthesizing a power minimum XOR tree given arbitrary input on probabilities. The algorithm is as follows: Let k be the number of primary inputs to the XOR tree with probabilities greater than half. If k is even replace all k inputs by their respective complements. If k is odd then replace any $k-1$ inputs by their respective complements. Then simply use the Huffman algorithm to build the optimal tree. For AND and OR trees we have the following theorems that are analogous to Theorem 2.3 for XOR trees:

Theorem 2.4 *If the on probabilities of the primary inputs to an* AND *tree are less than half then the Huffman algorithm synthesizes the minimum power tree. Similarly if the off probabilities of the primary inputs to an* OR *tree are less than half then the Huffman algorithm synthesizes the minimum power tree.*

Theorem 2.5 *If the product of the on probabilities of the primary inputs to an* AND *tree are greater than half then the Anti-Huffman algorithm synthesizes the minimum power tree. Similarly if the product of the off probabilities of the primary inputs to an* OR *tree are greater than half then the Anti-Huffman algorithm synthesizes the minimum power tree.*

Hence, if we choose a polarity vector such that all the primary inputs have on probabilities less than $1/2$, then we can directly apply Theorems 2.3-2.5 to efficiently synthesize every XOR tree, AND tree, and OR tree in the circuit with provably minimum power dissipation.

3 Factorization and Reduction Rules

In practice we want to reduce the size of a circuit to satisfy area constraints. Area reduction can be achieved through the application of factorization rules such as: (1) $a\,b \oplus a\,c \oplus a\,d \ldots = a(b \oplus c \oplus d \oplus \ldots)$ (2) $a\,b + a\,c + a\,d \ldots = a(b + c + d + \ldots)$, and reduction rules such as: (1) $a\,b \oplus a = a\,\overline{b}$, (2) $a\,b \oplus a\,c \oplus a\,b\,c = a(b + c)$, (3) $a\,b \oplus \overline{b} = a + \overline{b}$.

We observe that factorization does not change the polarities of the primary inputs, and so we can still directly apply Theorems 2.3-2.5 to efficiently synthesize power optimal trees. For example, Figure 3 contains the factored realizations of Expressions 1 and 2. Synthesis B, the power optimal realization, consumes 40% less power than Synthesis A. The reduction rules, however, may change the polarity of some of the primary inputs. In this case, we employ an efficient branch and bound algorithm that makes use of Theorems 2.3-2.5 to prune the search space in order to synthesize power optimal trees.

4 Experimental Results

Our algorithms have been implemented in C++ and executed on a Sparc 1 workstation. We used the PUMA OKFDD package from the University of Freiburg to generate the initial FPRM forms. In the tables the parameter α is the percentage of on probabilities of inputs that lie between $[0, .5 - \beta]$ and $[.5 + \beta, 1]$. The power measurements were conducted using the power estimation tool in SIS 1.2 assuming a 20MHZ clock. The circuits that we used came from the MCNC benchmark suite. Finally, all of the power measurements were of *mapped* circuits using the well-known MCNC generic library (mcnc.genlib).

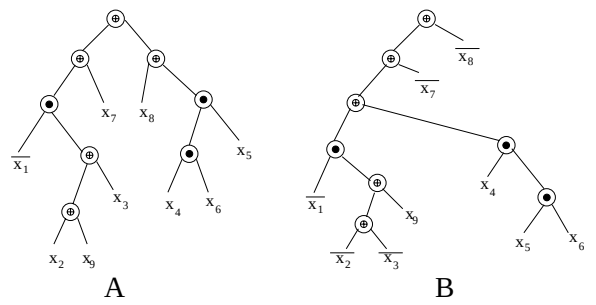

Figure 2: Respective factored multi-level implementations of Expression 1 and Expression 2

In Table 1, we compare the results of the balanced tree synthesis algorithm (denoted with a "B") from [11] using the the area minimization heuristic described in [10] to generate the initial FPRM form against the results of our algorithm which is referred to as the non-balanced synthesis (denoted with an "N"). First we see that our algorithm produced results that differ negligibly in area from the best known area minimization heuristic. In fact, overall, our realizations have a slightly smaller area. Second, we achieve an average 18% power reduction from the best known XOR based synthesis technique. Additionally, circuits such as `f51m` and `pcle` highlight the fact that the power savings is due to the optimal synthesis of the trees and the polarity of the variables and not a fortuitous initial FPRM form that has fewer literals.

For the sake of completeness, in Table 2 we compare the performance of SIS (by using the AND/OR based commands: `collapse` and `resub` and denoted with an "S".) against that of our algorithm. We selected four circuits which realize arithmetic functions as our benchmark circuits. On the average we have a 37% area savings and a 53% power savings. In fact, the best known SIS scripts (*boolean, rugged,* and *algebraic*) perform very poorly against XOR based techniques in terms of area and power for arithmetic circuits. The reader is referred to [11] for a detailed comparison between arithmetic circuits synthesized from the SIS scripts and arithmetic circuits synthesisized from FPRM forms.

5 Conclusion

In this paper we have presented a low power synthesis paradigm for XOR based logic. We have taken advantage of special properties of the FPRM form and XOR gates to develop an efficient optimal low power synthesis algorithm for XOR trees, AND trees, and OR trees. There are two natural extensions of this work. The first would be to modify the mapping phase to preserve the effects of the low power synthesis. The special properties of XOR gates and FPRM forms might yield additional opportunities for power savings beyond those presented in [9]. Second, the reduction and factorization rules have to be studied further in terms of their effect on power consumption of XOR based logic.

Acknowledgement

The authors would like to thank Chien-Chung Tsai of Mentor Graphics for his valuable suggestions.

References

[1] M. Davio, J. P. Deschamps, and A. Thayse. *Discrete and Switching Functions*. McGraw-Hill, Inc., 1978.

[2] R. Dreschsler, A. Sarabi, M. Theobald, B. Becker, and M. A. Perkowski. Efficient representation and manipulation of switching functions based on ordered kronecker functional decision diagrams. In *Design Automation Conference*, 1994.

[3] Unni Narayanan, Hon Wai Leong, Ki-Seok Chung, and C. L. Liu. Low power multiplexer decomposition. In *International Symposium on Lower Power Electronics and Design*, 1997.

[4] R. Panda and F. Najm. Technology decomposition for low-power synthesis. In *IEEE Custom Integrated Circutis Conference*, pages 627–630, 1995.

[5] R. Panda and F. Najm. Technology-dependent transformations for low-power synthesis. In *Design Automation Conference*, pages 650–655, 1997.

[6] D. S. Parker. Conditions for optimality of the huffman algorithm. *SIAM Journal of Computing*, 9(3):470–489, August 1980.

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

[8] Tsutomu Sasao, editor. *Logic Synthesis and Optimization*. Kluwer Academic Publishers, 1983.

[9] V. Tiwari, P. Ashar, and S. Malik. Technology mapping for low power. In *Design Automation Conference*, pages 74–79, June 1993.

[10] Chien-Chung Tsai and Malgorzata Marek-Sadowska. Minimization of fixed-polarity and/xor canonical networks. In *IEE Proceedings of Comput. Digital Technology*, volume 141, 1994.

[11] Chien-Chung Tsai and Malgorzata Marek-Sadowska. Multilevel logic synthesis for arithmetic functions. In *Design Automation Conference*, 1996.

[12] Chi-Ying Tsui, Massoud Pedram, and Alvin M. Despain. Technology decomposition and mapping targeting low power dissipation. In *Desgin Automation Conference*, pages 68–73, 1993.

[13] Neil H. E. Weste and Kamran Eshraghian. *Principles of CMOS VLSI Design: A Systems Perspective*. Addison-Wesley, 1993.

Circuit	PI	PO	α	β	Area			Power		
					Lit-B	Lit-N	% L Imp	Power-B	Power-N	% Pow. Imp
9symml	9	1	0.00	0.00	430	470	-9.30	1213.0	1206.2	-0.57
			0.50	0.25	430	534	-24.10	1103.2	1189.7	-7.84
			1.00	0.25	430	418	2.80	916.5	621.5	32.19
b1	3	4	0.00	0.00	25	20	20.00	61.9	68.4	-10.50
			0.50	0.25	25	20	20.00	39.7	35.3	11.08
			1.00	0.25	25	20	20.00	34.6	28.5	17.63
CM150	21	1	0.00	0.00	379	274	27.70	606.0	370.5	38.60
			0.50	0.25	379	321	15.30	612.6	407.9	33.41
			1.00	0.25	379	360	5.00	323.4	156.5	51.60
CM42	4	10	0.00	0.00	139	147	-5.70	308.0	311.8	-1.23
			0.50	0.25	139	118	15.10	263.2	190.5	27.69
			1.00	0.25	139	126	9.30	81.9	81.3	0.73
CM85	11	3	0.00	0.00	840	892	-0.60	2248.6	2257.9	-0.41
			0.50	0.25	840	503	40.10	2076.2	929.7	55.22
			1.00	0.25	840	531	36.80	1072.0	426.0	60.26
f51m	8	8	0.00	0.00	300	300	0.00	813.2	798.2	1.84
			0.50	0.25	300	290	3.33	703.3	534.8	23.96
			1.00	0.25	300	317	-5.60	497.8	347.2	30.25
pcle	19	9	0.00	0.00	250	266	-6.40	622.2	561.2	9.80
			0.50	0.25	250	495	-98.00	668.6	828.3	-23.90
			1.00	0.25	250	415	-66.00	365.2	310.0	15.12
t481	16	1	0.00	0.00	140	123	12.10	360.4	315.3	12.51
			0.50	0.25	140	135	3.60	268.0	262.3	2.10
			1.00	0.25	140	116	17.14	193.3	102.8	46.81
Average			0.00	0.00			4.72			6.30
			0.50	0.25			-3.08			15.20
			1.00	0.25			2.43			31.80
Overall Average							1.36			17.77

Table 1: Comparison between balanced and nonbalanced approaches where the minimum cubes heuristic was used for the balanced approach and the LSB power vector was used for the nonbalanced approach.

Circuit	PI	PO	α	β	Area			Power		
					Lit-S	Lit-N	% L Imp	Power-S	Power-N	% Pow. Imp
9symml	9	1	0.00	0.00	739	470	36.40	2647.0	1206.2	54.30
			0.50	0.25	739	534	27.70	1746.4	1103.2	36.83
			1.00	0.25	739	418	43.40	1228.4	621.5	49.40
f51m	8	8	0.00	0.00	445	300	32.50	1731.4	798.2	53.90
			0.50	0.25	445	290	34.80	1118.4	534.8	52.18
			1.00	0.25	445	317	28.80	743.2	347.2	53.30
pcle	19	9	0.00	0.00	335	266	20.60	1131.4	561.2	50.40
			0.50	0.25	335	495	-47.80	764.4	828.3	-8.35
			1.00	0.25	335	415	-23.90	323.4	310.0	4.10
t481	16	1	0.00	0.00	6847	123	98.20	19322.7	315.3	98.40
			0.50	0.25	6847	135	98.00	13441.7	262.3	98.00
			1.00	0.25	6847	116	98.21	7588.0	7588.0	98.60
Average			0.00	0.00			46.93			64.25
			0.50	0.25			28.18			44.67
			1.00	0.25			33.63			51.35
Overall Average							37.24			53.42

Table 2: Comparison between SIS and the nonbalanced approach using the LSB power vector.