Low Power Multiplexer Decomposition

Unni Narayanan*
Dept. of Computer Science
University of Illinois
Urbana, IL 61801

Hon Wai Leong
Dept. of Information and Computer Science
National University of Singapore
Singapore 119260

Ki-Seok Chung * C. L. Liu *
Dept. of Computer Science
University of Illinois
Urbana, IL 61801

Abstract

The advent of portable digital devices such as laptop personal computers has made low power circuit design an increasingly important research area. Recently, low power decomposition for simple logic gates such as AND *and* OR *has been extensively researched. However, the problem of MUX decomposition to minimize power dissipation has not been addressed. In this paper, we study the problem of low power multiplexer (MUX) decomposition. MUX decomposition is the procedure of transforming an n-to-one MUX into an equivalent tree of two-to-one MUXes. We propose a formulation for the minimum power MUX decomposition problem based on the common CMOS pass transistor implementation of a MUX. Given the occurrence probabilities of the data signals and their on probabilities, we analyze the power dissipation of our MUX implementation and give a general method for computing the power dissipation of a MUX tree decomposition. We then present several algorithms which efficiently generate minimum power MUX decompositions. We demonstrate the effectiveness of our algorithms with experimental results.*

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, portable electronic devices have limited battery life, and so the circuitry in these devices 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.

In this research, we address the issue of low power decomposition of multiplexers (MUXes). The problem of MUX decomposition is that of converting an n-to-1 MUX into a logically equivalent MUX tree of 2-to-1 MUXes. For example, Figure 1 shows two possible MUX tree decompositions of an 8-to-1 MUX. The data

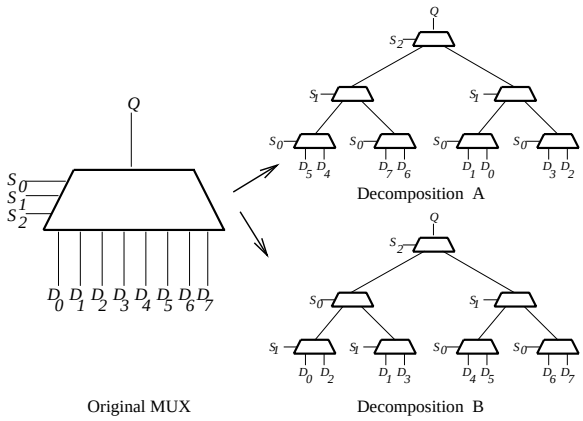

Figure 1: Different MUX decompositions result in different power dissipations

signals of the MUX are denoted by $D_0, D_1, \ldots, D_7$ while the selection signals of the MUX are denoted by S_0, S_1, S_2. We use the convention that the combination $(S_2S_1S_0)$ of selection signals that will be used to "select" the data signal D_j where j is the decimal equivalent of $(S_2S_1S_0)$. We shall refer to this combination as the *encoding* for D_j. For example, the combination (101) will be used to select D_5 and so the encoding for D_5 is (101).

A *balanced MUX decomposition* is a MUX decomposition in which all the path lengths from the root MUX to the data signals are equal. Note that both the decompositions in Figure 1 are balanced. A *uniform MUX decomposition* is a MUX decomposition in which all of the (2-to-1) MUXes in the same level of the MUX tree use the *same* selection signal. In Figure

*Supported in part by the National Science Foundation (NSF) under grant MIP-9222408.

1, Decomposition A is an example of a uniform decomposition. A *nonuniform MUX decomposition* is a MUX decomposition in which the (2-to-1) MUXes in a given level of the MUX tree can use *different* selection signal. Decomposition B in Figure 1 is an example of a nonuniform decomposition.

In any MUX decomposition, the *power consumed* is given by the sum of the power consumed in all the MUXes in the tree. In general, different decompositions leads to different power consumptions. We define a *minimum power MUX decomposition* to be a MUX decomposition in which the power consumption is minimized. We use a model of implementation of a 2-to-1 MUX where the power consumed is proportional to the switching activity and is expressed by the following formula:

$$\frac{1}{2}CV_{dd}^2 \cdot f \qquad (1)$$

where C is the output capacitance of the gate, V_{dd} is the supply voltage, and f is the frequency of transitions at the output of the gate.

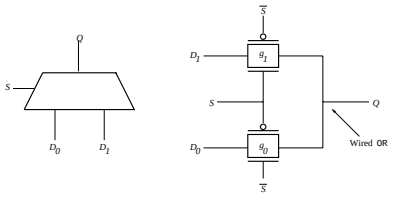

Figure 2: A 2-to-1 MUX and its corresponding pass transistor implementation.

Both the straightforward two-level AND/OR implementation of a 2-to-1 MUX and the commonly used pass transistor implementation of a 2-to-1 MUX illustrated in Figure 2 dissipate power according to Equation 1. Observe that if the on probability of the output of a MUX M is P_M then under the zero-delay model the power dissipation of the MUX is proportional to $2P_M(1-P_M)$.

In this paper we solve the following problem: Given an arbitrary n-to-1 MUX M with a fixed encoding for the data signals, determine a minimum power balanced MUX decomposition of M, i.e. a balanced decomposition of the MUX M into a MUX tree consisting of 2-to-1 MUXes. We study both uniform and nonuniform low power decompositions.

The general idea of MUX decomposition and its importance were presented in [7, 11] and most recently, in [12] where MUX decomposition was studied in relation to delay minimization. In particular, Thakur *et al.* states in [12] that most circuits contain large numbers of MUXes. Consequently, they argue that overall delay in a circuit can be decreased by optimizing the delay (via suitable decomposition) in these MUXes embedded in the circuit. However, this previous research on MUX decomposition did not consider the issue of optimization of the power dissipation. The related problem of logic decomposition of simple gates such as AND and OR for power reduction has been widely studied in [14, 8, 10, 13]. However, none of this work addresses the decomposition problem of more complex logic gates such as MUXes.

This paper is organized as follows: In Section 2 we analyze the power dissipation of 2-to-1 MUXes in MUX trees. In Section 3 we propose three original heuristic algorithms for efficiently performing low power MUX decomposition. In Section 4 we describe how we handle incomplete MUXes in our algorithms. In Section 5 we present experimental results demonstrating the effectiveness of the various approaches. In Section 6 we summarize our results and discuss possible future directions of this research.

2 Power Dissipation in a MUX Tree

In any MUX tree, the total power dissipation is given by the sum of the power dissipation in the individual MUXes in the tree. We know that the power dissipation in the individual MUXes can be computed from the on probabilities of the their outputs. For a 2-to-1 MUX, this can be done quite easily. However, for MUX trees, the computation of the on probabilities of an "internal" MUX in the tree is quite complicated. We first require some basic terminology.

2.1 Basic Terminology

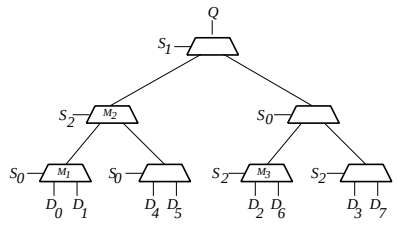

Figure 3: M_2 is an *internal* MUX. M_2 and M_3 are not internal MUXes.

Consider the MUX tree shown in Figure 3. We define a *leaf MUX* to be a MUX at the bottom-most level of the tree, where both the MUX inputs correspond to some given data signals. The remaining MUXes in the tree are called *internal MUXes*. In Figure 3, M_1 is a leaf MUX, while M_2 is an internal MUX. For any MUX M, we define the *subtree at M* to mean the subtree with root M. We define the data signals set of M, denoted $\mathcal{D}(M)$ to be the set of data signals to the MUXes in the subtree at M. We define the selection

signals set of M, denoted $\mathcal{S}(M)$ to be the set of selection signals to the MUXes in the subtree at M. For example, in Figure 3 $\mathcal{D}(M_1) = \{D_0, D_1\}$, $\mathcal{S}(M_1) = \{S_0\}$ and $\mathcal{D}(M_2) = \{D_0, D_1, D_4, D_5\}$, $\mathcal{S}(M_2) = \{S_0, S_2\}$. The probability that a data signal D_i is high is denoted by d_i. The probability that a selection signal S_i is high is denoted by s_i.

2.1.1 Leaf 2-to-1 MUXes

For a leaf MUXes, we have the following theorem:

Theorem 2.1 *For a leaf 2-to-1 MUX M with data signals D_0 and D_1 and selection signal S, the on probability of the output is given by $P_M = \overline{s}d_0 + sd_1 = (1-s)d_0 + sd_1$.*

Hence, the power dissipation of a leaf 2-1 MUX is given by $2(\overline{s}d_0 + sd_1)(1 - (\overline{s}d_0 + sd_1))$.

2.1.2 Internal 2-to-1 MUXes

For an internal MUX M, the on probability of M is a function of $\mathcal{D}(M)$ and of $\mathcal{S}(M)$. We first illustrate this with an example. Consider the internal MUX M_2 in Figure 3 with $\mathcal{D}(M_2) = \{D_0, D_1, D_4, D_5\}$, and $\mathcal{S}(M_2) = \{S_0, S_2\}$. The output of M_2 is a 1 whenever the "selected" data signal from $\mathcal{D}(M_2)$ is a 1. Note that selection is done via the selection signals in the set $\mathcal{S}(M_2)$. By enumerating the cases, it is easy to see that $P_{M_2} = \text{Prob}[(S_2 = 0) \wedge (S_0 = 0)] \cdot d_0 + \text{Prob}[(S_2 = 0) \wedge (S_0 = 1)] \cdot d_1 + \text{Prob}[(S_2 = 1) \wedge (S_0 = 0)] \cdot d_4 + \text{Prob}[(S_2 = 1) \wedge (S_0 = 1)] \cdot d_5$

Note that the encoding for D_5 is $E(D_5) = (\underline{1}0\underline{1})$ and D_5 is selected when $(S_2 = 1) \wedge (S_0 = 1)$. Similarly, (The value of S_1 is immaterial in this case since $S_1 \notin \mathcal{S}(M_2)$.)

In general, for any internal MUX M, each data signal $D_j \in \mathcal{D}(M)$ with encoding $E(D_j) = (e_{k-1} \ldots e_1 e_0)$ will be selected at M when $S_i = e_i$ for each selection signal $S_i \in \mathcal{S}(M)$. More formally we have the following definition:

Definition 2.1 *Let D_j be a fanin of a MUX M. Then let $\mathcal{SP}(D_j, M)$ be an assignment of values to the selection signals in $\mathcal{S}(M)$ where $S_i = e_i$ for all $S_i \in \mathcal{S}(M)$. We call $\mathcal{SP}(D_j, M)$ the* selection pattern *for D_j with respect to the MUX M, or equivalently, with respect to $\mathcal{S}(M)$.*

From Definition 2.1 we get the equality: $Prob(\mathcal{SP}(D_j, M)) = Prob\left(\forall_{S_i \in \mathcal{S}(M)}(S_i = e_i)\right)$ We say that $Prob(\mathcal{SP}(D_j, M))$ is the probability of occurrence of $\mathcal{SP}(D_j, M)$. Then the on probability P_M of any internal MUX M is given by the following:

Theorem 2.2 *For any MUX M, the on probability P_M is given by*

$$P_M = \sum_{\forall D_j \in \mathcal{D}(M)} Prob(\mathcal{SP}(D_j, M)) \cdot d_j \qquad (2)$$

Proof: Omitted for brevity. □

We have the following important result that follows from Theorem 2.2.

Theorem 2.3 (Independence Theorem) *The on probability P_M of any internal MUX M is independent of the order of the selection signals in the subtrees of M.*

Proof: From Theorem 2.2 we see that P_M is independent of the order of the selection signals in the subtree. □

To compute $Prob(\mathcal{SP}(D_j, M))$ we use the occurrence probability of a data signal which is defined as follows:

Definition 2.2 *Given an n-to-1 MUX with data signals $D_0, D_1, \ldots, D_{n-1}$ let $P(D_j)$ denote the probability that signal D_j is selected as the output of the n-to-1 MUX, We call $P(D_j)$ the occurrence probability of data signal D_j.*

Given the occurrence probabilities of the data signal we can compute the on probabilities of any MUX in the MUX tree without making a spatial independence assumption by using the principle of inclusion and exclusion [6]. We omit the details for the sake of brevity.

3 Heuristic Algorithms

We present three efficient heuristics for determining a balanced minimum power MUX decomposition: BOTTOM UP, TOP DOWN, and HYBRID.

3.1 Bottom Up

In the BOTTOM-UP heuristic algorithm we construct a uniform MUX decomposition from the leaves to the root. Conceptually each signal corresponds to the vertex of a boolean cube. For example the data signals of the MUX in Figure 1 can be mapped to the three dimensional cube in Figure 4. Each weighted edge in the boolean cube corresponds to the power dissipated by the MUX which combines the data signals located at the ends of the edge. In Figure 4 , the edge weight α would be the power dissipated by the 2-to-1 MUX which combines data signals D_0 and D_4 using selector signal S_2. Next, BOTTOM-UP greedily determines for each dimension of the cube the sum of the weights. The algorithm then merges vertices along the dimension of the cube that has the smallest sum of weights. For example, suppose for the MUX that Initially the

cheapest dimension to merge vertices is along dimension three. Then the cube will now shrink to a square. This corresponds to merging data signals into MUXes with selection signal S_2.

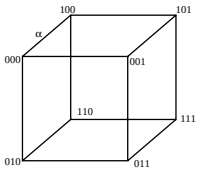

Figure 4: Boolean cube corresponding to 8-to-1 MUX.

BOTTOM-UP recursively calls itself on the output signals of the MUXes merged in the previous step. It terminates when there is just one output signal. Figure 5 shows the progress of the heuristic.

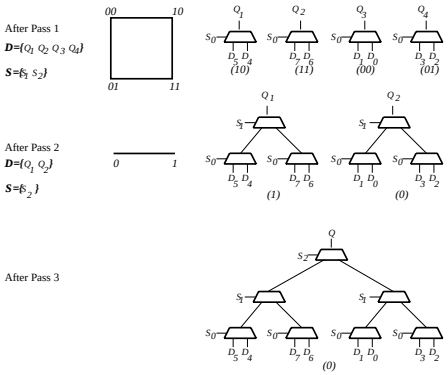

Figure 5: Progress of BOTTOM UP heuristic.

3.2 Top Down

In the TOP-DOWN heuristic algorithm we construct the MUX decomposition from the root to the leaves. Given a MUX M, we define a *partition of M along S* to be the partitioning of the data signals of M along the selection signal S. We can partition M along S_0,S_1, or S_2 as shown in Figure 6. For each of these partitions, we can compute the power dissipation at the two induced MUXes M_1 and M_2 (see Figure 6) *directly* using Theorem 2.2 because of the Independence Theorem. (The Independence Theorem also say that the power dissipation at the root MUX is the same for all these partitions.)

The basic idea in TOP DOWN is to partition M along all possible selection signals and choose the selection signal S^* that results in the minimum power dissipation in the pair of MUXes M_1 and M_2. For this example, choosing $S^* = S_2$ results in the minimum power dissipation. Hence TOP-DOWN partitions the signals

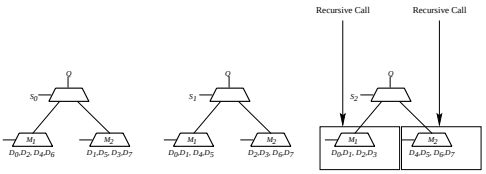

Figure 6: First pass of TOP DOWN heuristic.

along S_2 and recursively calls itself on each of the subtrees rooted at M_1 and M_2. The recursion terminates when there are just two data signals remaining.

3.3 Hybrid

In the BOTTOM-UP algorithm, we attempt to minimize overall power by greedily combining signals from the leaves of the tree to the root of the tree. The advantage of this approach is that at each stage of the algorithm we minimize the power of half the remaining MUXes in the tree. The disadvantage of BOTTOM-UP is that we only consider uniform solutions. The advantage of the TOP-DOWN algorithm is that we consider nonuniform solutions. The disadvantage is, of course, that we are only minimizing the overall power of two MUXes at a time. In the HYBRID heuristic algorithm we combine the best attributes of both the BOTTOM-UP and TOP-DOWN approaches.

HYBRID works by calling the BOTTOM-UP algorithm to generate an initial decomposition of the tree. The selection signal that ends up at the root of the tree is then used to partition the signals into two subtrees. HYBRID then recursively calls itself on each of the subtrees. We trace the algorithm in Figure 7. Initially a bottom up pass is performed and yields S_2 as the root. In the next step we would partition the tree at the root

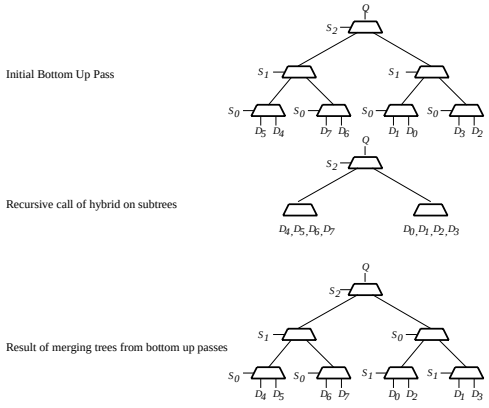

Figure 7: Progress of HYBRID heuristic.

and recursively call HYBRID on signals D_0, D_1, D_2, D_3

and on signals D_4, D_5, D_6, D_7. HYBRID calls BOTTOM UP on both of the subtrees and a nonuniform decomposition is obtained.

4 Incomplete MUXes

We now describe how to handle the general case of an m-to-1 MUX where m is not a power of 2 – we call these *incomplete MUXes*. The basic idea is to treat an incomplete m-to-one MUX as a complete n-to-one MUX where n is the smallest power of 2 greater than m, by assigning the unused (or "dummy") data signals with on probability of 0 (and occurrence probability of 0 by definition). Suppose we have a 4-to-1 MUX in which D_1 is an unused input signal. We denote a MUX with data signal D_1 as a "virtual" MUX which consumes 0 power. This has two effects. First, by Theorem 2.2 D_1 will not contribute to the on probability of the output of any MUX in the tree. Second, the virtual MUX does not add any power consumption to the tree. The net result is that the balanced tree with the virtual MUX is identical to the unbalanced tree without the virtual MUX and without the unused signal.

The results and algorithms in this paper then extend to incomplete MUXes with slight modification.

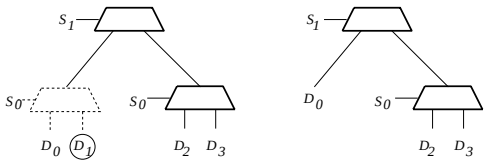

Figure 8: One the left is a balanced tree with a virtual MUX. On the right is the corresponding non-balanced tree. Both trees are equivalent in terms of power consumption.

5 Experimental Results

We have implemented all the MUX decomposition algorithms presented in this paper, namely, Bottom-Up for uniform decomposition, Hybrid and Top-Down for non-uniform decomposition, and Branch-and-Bound for both. The implementations are in C++ running on a SUN Sparc 20 workstation. In addition, we have also incorporated the extension to handle incomplete MUXes in all of these implementations.

Note that in general, to obtain the occurrence probabilities of the data signals requires observing the dynamic nature of these data signals and the MUX output over the execution of the circuit. Since there are, as yet, no benchmark data on these, we have resorted to generating the occurrence probabilities for our test cases.

In our experiments, we allow the on probabilities of the data signals d_j's to vary uniformly over a range $0.5(1-\beta)$ where the parameter β $(0 \leq \beta < 1)$ is used to control the variation in the on probability d_j's. A similar method is used to control the variation in the occurrence probabilities of the data signals, $P(D_j)$'s. These are generated to vary uniformly over the range $\overline{p}(1-\alpha)$ where the parameter α $(0 \leq \alpha < 1)$ controls the variation in the occurrence probabilities $P(D_j)$'s. We decompose MUXes with different sizes with n ranging from 8 to 32. For each size n, we generate data sets from various class $\mathcal{M}(n, \alpha, \beta)$ of MUXes of size n. For incomplete MUXes, we also generated the subclass $M_1(n, \alpha, \beta, \gamma)$ where the additional parameter γ $(0 <= \gamma < 0.5)$ is the percentage of don't care input signals.

MUX	n	α	β	γ	BOTTOM-UP	Optimal	Worst
M8.2	8	0.45	0.5	0	3.214*	3.214	3.486
M16.2	16	0.45	0.5	0	7.108*	7.108	7.429
M32.2	32	0.45	0.5	0	14.869*	14.869	15.331
M8.3	8	0.2	0.8	0	2.746*	2.746	3.459
M16.3	16	0.2	0.8	0	5.927*	5.927	7.225
M32.3	32	0.2	0.8	0	13.110*	13.110	14.739
MI8.2	8	0.5	0.5	0.2	1.531*	1.531	1.778
MI16.2	16	0.5	0.5	0.2	4.398	4.388	4.894
MI32.2	32	0.5	0.5	0.2	9.042	8.872	9.418
MI8.3	8	0.5	0.5	0.3	1.299*	1.299	1.475
MI16.3	16	0.5	0.5	0.3	3.433*	3.433	3.897
MI32.3	32	0.5	0.5	0.3	8.142	7.845	8.660
MI8.4	8	0.5	0.5	0.4	1.312*	1.312	1.450
MI16.4	16	0.5	0.5	0.4	2.940*	2.940	3.235
MI32.4	32	0.5	0.5	0.4	4.863	4.792	5.409

Table 1: Uniform decomposition results.

MUX	HYBRID	TOP-DOWN	Optimal	Worst
M8.2	3.212*	3.212*	3.212	3.486
M16.2	6.992*	6.992*	6.992	7.37
M32.2	14.79*	15.058	14.784	15.456
M8.3	2.743*	2.743*	2.743	3.459
M16.3	5.809*	6.002	5.809	7.250
M32.3	13.084	13.376	13.053	15.036
MI8.2	1.509*	1.706	1.509	1.778
MI16.2	4.345	4.659	4.317	4.931
MI32.2	8.729	9.177	8.530	9.631
MI8.3	1.299*	1.362	1.299	1.475
MI16.3	3.439	3.826	3.424	3.917
MI32.3	7.883	8.443	7.580	8.792
MI8.4	1.312*	1.450	1.312	1.450
MI16.4	2.737	3.085	2.686	3.359
MI32.4	4.741	4.863	4.624	5.504

Table 2: Nonuniform decomposition results

In the tables, an $*$ denotes that the algorithm returned the optimal decomposition. The columns designated as optimal in Table 1 and Table 2 contain the optimal uniform and optimal nonuniform MUX

decompositions respectively. The optimal decompositions were obtained by using a branch and bound algorithm that takes advantage of Theorem 2.3 to prune the search space. Although, the branch and bound runs significantly faster than exhaustive search it is still too slow to be of practical use in a synthesis system.

Table 1 contains the total power consumed in the BOTTOM-UP decomposition, the optimal uniform decomposition, and the worst case uniform decomposition. There are two major points that should be noted. First, in general, regardless of the values of α,β, or γ in every case BOTTOM-UP returns a result that is an optimal or near optimal uniform decomposition. Second, we see that the potential power savings between an optimal uniform decomposition and the worst uniform decomposition can be as high as 34%.

Table 2 contains the results for non-uniform decompositions. The algorithms were run on the same MUXes with the same parameters as in Table 1 In general, HYBRID performs optimally or near optimally in all cases. It is interesting to note that TOP-DOWN generally performs worse than HYBRID and TOP-DOWN performs especially worse in decomposition of incomplete MUXes. This reflects the fact that careful pairings of "dummy" signals at the bottom levels of the tree result in greater power savings than partitioning all data signals at the root.

6 Conclusion

In this paper we have solved the problem of determining the minimum power balanced MUX decompositions. The MUX power model that we used has a physical basis in terms of the commonly accepted pass transistor implementation. We used occurrence probabilities to compute the power dissipation of 2-to-1 MUXes in the MUX tree. We presented one algorithm for generating low power uniform MUX decompositions and two algorithms for generating low power nonuniform MUX decompositions. Our experimental results have shown that in general the heuristics perform quite close to optimal. We have also demonstrated that the difference in power between the best uniform decomposition and the best nonuniform decomposition is minimal, but the difference between the best decomposition and the worst decomposition can be as much as 34%.

Our ongoing research addresses the natural generalization of this problem, namely determining the minimum power decomposition when no encoding is specified for the data signals. It should be clear that, in general, the encoding of the data signals affects the power dissipation in a MUX tree. This is a direct consequence of Theorem 2.2. Hence there are two problems that need to be solved. The first is the problem of determining a fixed length encoding of the data signals such that power is minimized in the MUX tree. The second is the problem of determining a variable length encoding of the data signals such that the power is minimized in a MUX tree.

References

[1] R. K. Brayton, G. D. Hachtel, and A. L. Sangiovanii-Vincentelli. Multilevel logic synthesis. *Proceedings of the IEEE*, 78(2):264–300, February 1990.

[2] Anantha P. Chandrakasan and Robert W. Broderson. *Low Power Digital CMOS Design*. Kluwer Academic Publishers, 1995.

[3] A. A. Ghosh, S. Devadas, K. Keutzer, and J. White. Estimation of average switching activity in combinational and sequentail circuits. In *Design Automation Conference*, pages 68–73, 1992.

[4] D. A. Huffman. A method for the construction of minimum redundancy codes. In *Proceedings of the IRE*, volume 40, pages 1098–1101, September 1952.

[5] Eric Lehman, Yosinori Watanabe, Joel Grodstein, and Heather Harkness. Logic decomposition during technology mapping. In *International Conference on Compuer-Aided Design*, pages 264–271, 1995.

[6] C. L. Liu. *Introduction To Combinatorial Mathematics*. McGraw-Hill, Inc., 1968.

[7] R. Murgai, R. K. Brayton, and A. L. Sangiovanni-Vincentelli. An improved synthesis algorithm for multiplexer-based pgas. In *Design Automation Conference*, pages 380–386, 1992.

[8] Rajeev Murgai, Robert K. Brayton, and Alberto Sangiovanni-Vincentelli. Decomposition of logic functions for minimum transition activity. In *Proceedings of European Design and Test Conference*, pages 404–410, 1995.

[9] F. Najm. Transition density, a stochastic measure of activity in digital circuits. In *International Conference on Compuer-Aided Design*, pages 644–649, June 1991.

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

[11] R. L. Rudell. *Logic Synthesis for VLSI Design*. PhD thesis, U. C. Berkely, April 1989.

[12] Shashidhar Thakur, D. F. Wong, and Shankar Krishnamoorthy. Delay minimal decomposition of multiplexers in technology mapping. In *Design Automation Conference*, 1996.

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

[14] 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.

[15] Sarma B. K. Vrudhula and Hong-Yu Xie. Techniques for cmos power estimation and logic synthesis for low power. In *International Workshop on Low Power Design*, pages 21–26, 1994.

[16] Neil H. E. Weste and Kamran Eshraghian. *Principles of CMOS VLSI Design*. Addison-Wesley Publishing Company, 1993.