Low Power Multiplexer Decomposition

Unni Narayanan*  Hon Wai Leong  Ki-Seok Chung *  C. L. Liu *
Dept. of Computer Science  Dept. of Information and Computer Science  Dept. of Computer Science
University of Illinois  National University of Singapore  University of Illinois
Urbana, IL 61801  Singapore 119260  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 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 $(S_2 S_1 S_0)$ of selection signals that will be used to “select” the data signal $D_j$ where $j$ is the decimal equivalent of $(S_2 S_1 S_0)$. 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 1, Different MUX decompositions result in different power dissipations.
1. Decomposition A is an example of a uniform decomposition. A \textit{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 \textit{power consumed} is given by the sum of the power consumed in all the MUXes in the tree. In general, different decompositions lead to different power consumptions. We define a \textit{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:

\[
\frac{1}{2}CV_{dd}^2 \cdot f
\]

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.

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 \textit{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

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 \textit{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 \(D(M)\) 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 MUX, 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 \( S_i \), the on probability of the output is given by \( P_M = \text{Prob}(D_0 + sd_i) = (1 - s)d_0 + sd_i \).

Hence, the power dissipation of a leaf 2-1 MUX is given by \( 2(\text{Prob}(D_0 + sd_i))(1 - (\text{Prob}(D_0 + sd_i)) \).

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 \( M_3 \) in Figure 3 with \( \mathcal{D}(M_3) = \{ D_0, D_1, D_4, D_5 \} \), and \( \mathcal{S}(M_3) = \{ S_0, S_2 \} \). The output of \( M_3 \) is a 1 whenever the “selected” data signal from \( \mathcal{D}(M) \) is a 1. Note that selection is done via the selection signals in the set \( \mathcal{S}(M) \). By enumerating the cases, it is easy to see that \( P_M = \text{Prob}(S_2 = 0) \cdot d_0 + \text{Prob}(S_2 = 0) \cdot (S_0 = 1) \cdot d_1 + \text{Prob}(S_2 = 1) \cdot (S_0 = 0) \cdot d_4 + \text{Prob}(S_2 = 1) \cdot (S_0 = 1) \cdot d_5 \).

Note that the encoding for \( D_5 \) is \( E(D_5) = (101) \) and \( D_3 \) is selected when \( (S_2 = 1) \land (S_0 = 1) \). Similarly, \( (\text{The value of } S_1 \text{ is immaterial in this case since } S_1 \notin \mathcal{S}(M_3)) \).

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: \( \text{Prob}(\mathcal{SP}(D_j, M)) = \text{Prob}(\forall S_i \in \mathcal{S}(M) (S_i = e_i)) \).

We say that \( \text{Prob}(\mathcal{SP}(D_j, M)) \) 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

\[
P_M = \sum_{D_j \in \mathcal{D}(M)} \text{Prob}(\mathcal{SP}(D_j, M)) \cdot d_j
\]

**Proof:** Omitted for brevity. \( \square \)

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. \( \square \)

To compute \( \text{Prob}(\mathcal{SP}(D_j, M)) \) 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 \( \alpha \) 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.](image)

### 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 along $S_2$ and recursively calls itself on each of the sub-trees 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.](image)
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.

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 $\beta$ ($0 < \beta < 1$) 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 $[1 - \alpha]$ where the parameter $\alpha$ ($0 < \alpha < 1$) 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 $M(n, \alpha, \beta)$ of MUXes of size $n$. For incomplete MUXes, we also generated the subclass $M_1(n, \alpha, \beta, \gamma)$ where the additional parameter $\gamma$ ($0 \leq \gamma < 0.5$) is the percentage of don't care input signals.

### Table 1: Uniform decomposition results.

<table>
<thead>
<tr>
<th>MUX</th>
<th>Hybrid</th>
<th>Bottom-Up</th>
<th>Optimal</th>
<th>Worst</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_4$</td>
<td>3.217*</td>
<td>3.217*</td>
<td>3.217</td>
<td>3.486</td>
</tr>
<tr>
<td>$M_8$</td>
<td>6.992*</td>
<td>6.992*</td>
<td>6.992</td>
<td>7.37</td>
</tr>
<tr>
<td>$M_{16}$</td>
<td>14.75*</td>
<td>15.058</td>
<td>14.784</td>
<td>15.46</td>
</tr>
<tr>
<td>$M_8$</td>
<td>2.743*</td>
<td>2.743*</td>
<td>2.743</td>
<td>3.459</td>
</tr>
<tr>
<td>$M_{16}$</td>
<td>5.809</td>
<td>6.001</td>
<td>5.809</td>
<td>7.39</td>
</tr>
<tr>
<td>$M_{32}$</td>
<td>13.084</td>
<td>13.376</td>
<td>13.083</td>
<td>15.036</td>
</tr>
<tr>
<td>$M_{16}$</td>
<td>2.743*</td>
<td>2.743*</td>
<td>2.743</td>
<td>3.459</td>
</tr>
<tr>
<td>$M_{32}$</td>
<td>4.345</td>
<td>4.659</td>
<td>4.317</td>
<td>4.931</td>
</tr>
<tr>
<td>$M_{64}$</td>
<td>8.729</td>
<td>9.177</td>
<td>8.530</td>
<td>9.631</td>
</tr>
<tr>
<td>$M_{64}$</td>
<td>1.509*</td>
<td>1.706</td>
<td>1.509</td>
<td>1.778</td>
</tr>
<tr>
<td>$M_{128}$</td>
<td>4.345</td>
<td>4.659</td>
<td>4.317</td>
<td>4.931</td>
</tr>
<tr>
<td>$M_{256}$</td>
<td>8.729</td>
<td>9.177</td>
<td>8.530</td>
<td>9.631</td>
</tr>
<tr>
<td>$M_{128}$</td>
<td>1.509*</td>
<td>1.706</td>
<td>1.509</td>
<td>1.778</td>
</tr>
<tr>
<td>$M_{512}$</td>
<td>4.345</td>
<td>4.659</td>
<td>4.317</td>
<td>4.931</td>
</tr>
</tbody>
</table>

### 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 $\alpha, \beta$, or $\gamma$ 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 the 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