Delay Prediction for Technology-Independent Logic Equations

Paul T. Gutwin* Patrick C. McGeer and Robert K. Brayton
EECS Department
University of California, Berkeley
Berkeley, CA 94720

Abstract
We present a new technology independent delay model. This model assumes that the technology mapper will attempt modest local restructuring of the network. It models the restructuring by producing a staggered network for each gate based on the arrival times of the fanin signals. The delay of the network is calculated using this network, and when compared to the delay reported by technology mapping, is found to be accurate and efficient.

1 Introduction
Combinational logic circuits are typically represented as a directed acyclic graph. Each of the nodes in the graph has an associated boolean function, and each of the edges in the graph represent signals in the circuit. When the nodes in the graph represent arbitrary boolean functions, the network is considered to be technology independent.

Delay modeling at the technology dependent phase is the process of assigning delay values from library elements to the nodes in the network. However, at the technology independent stage, the structure of the implemented network is not known, and delay prediction is much harder.

Technology independent delay prediction of a node must depend on some assumptions about the structure of the network that will implement that node. The simplest assumption is that all paths through a node should have equal delay, and is known as balanced delay. If all paths through the node have equal delay, the output arrival time is controlled by the latest arriving input signal. We could obtain a performance improvement if the implementation of this node allowed the latest arriving signal to have a smaller delay to the output than the other input signals. This is what performance enhancement techniques and technology mapping procedures attempt to do. This paper deals with the more realistic question of predicting the unbalanced delay of a node.

2 Previous Work
One method for providing an accurate technology independent delay model uses a technology mapper to map each node to a temporary network. The delay of this temporary network is reported as the delay of the node [2].

This model is deficient in two ways. First, it creates a network where each of the paths through the network has equal delay. Secondly, it can be time consuming, since it creates a second network using a technology mapper.

Another model proposed by Wallace and Chandrasekhar [9] predicts the delay of a technology independent network without mapping the circuit. Their model is based on the assumption that the technology mapper will implement all nodes as balanced trees of gates. They define the term complexity to be the number of terms times the maximum number of literals in any term of a sum-of-products expression. The delay of a node is

\[ d = a_0 + a_1 \ln c + a_2 \ln f \]  

where \( c \) is the complexity of the node, \( f \) is the number of fanout of the node, and \( a_0, a_1 \) and \( a_2 \) are technology constants.

The Wallace and Chandrasekhar model more accurately predicts the balanced delay through a node without creating a subnetwork, and thus has a speed advantage over the mapped delay model. However, this model fails to accurately predict the delay of a circuit since current technology mapping techniques produce unbalanced trees of gates.

3 Proposed Delay Model

3.1 Model Overview
Modern technology mappers attempt to optimize the delay and area of circuits [6, 8]. The problem of accurate technology-independent delay modeling is therefore the problem of rapidly and accurately modeling the performance-oriented technology mapper. Simply running the mapper is too expensive, especially for the delay optimization mappers. Rather, we would like to abstract the effect of the techniques used by such mappers. These techniques fall into three categories: fanout optimization, gate selection and performance-enhancing techniques. All of these, however, may be subsumed by arguing that the general effect of these strategies is the same as applying a technology-independent speedup technique on local areas of the network, as we will show later.

Performance-enhancing techniques are in general socialist: they increase the delay of early arriving signals in order to reduce the delay on late arriving signals. These transformations produce a network of logic as depicted in figure 1. In this staggered network, the inputs are grouped in descending order with respect to arrival times: the inputs in
factor transformation. We therefore examine the cofactor transformation in detail.

The cofactors have been proposed, in [1, 4, 5, 7] they all have been shown to be variations on the general theme of the Shannon cofactor transformation. We therefore examine the cofactor transformation in detail.

Given a function \( f \) and some late arriving signal \( x \), the Shannon cofactor of \( f \) is given as \( f = x f_x + \overline{x} f_{\overline{x}} \).

This leads to the following physical realization of \( f \): The cofactors \( f_x \) and \( f_{\overline{x}} \) are formed, and then selected by a multiplexer controlled by \( x \). Clearly, this is a circuit where there is minimum delay between \( x \) and the output.

Consider what would happen if there were two late arriving signals \( x \) and \( y \). We would write \( f = xy f_{xy} + x \overline{y} f_{xy} + \overline{x}y f_{\overline{x}y} + \overline{x} \overline{y} f_{\overline{x}y} \), and attempt to achieve an efficient representation of the 4:1 multiplexer implied. When the Shannon expansions are completed, the resulting circuit from the cofactor transformation takes on the appearance of the tree in figure 2. This tree, called the Shannon cofactor tree, has leaves labeled 0 and 1, a root equivalent to the function output, and nodes equivalent to the function variables.

Since our ultimate goal is to predict the effects of this type of transformation, we can make the following observation. If we were able to find the delay of each of the functional blocks depicted in figure 1, we could use this information to calculate the unbalanced delay of the entire function as a result of the transformation.

The Shannon tree in figure 2 represents our example function where each of the \( F_i \)'s represent the Shannon cofactor of the signals in group \( G_i \) of the function \( F \). Referring again to figure 1, we note that since each of the signals in group \( G_i \) arrive at roughly the same time, the optimum delay for each of the signals in \( G_i \) through \( F_i \) would be equal as well. Making any one of these delays smaller than the rest would have the consequence of making some other path through \( F_i \) longer, and this new longer path would become the critical path through the circuit.

We now have a framework for calculating the unbalanced delay of a node:

1. The input signals are partitioned into groups based on their relative arrival times.
2. Derive the equivalent network of \( F_i \) by performing the cofactor of the node function \( F \) over the group \( G_i \).
3. Calculate the balanced delay of each of the functional blocks \( F_i \).

The total delay for \( F \) is given as the critical path through the resulting network.

Given a partition of the input signals, we need to compute the delay through each functional block \( F_i \). We can estimate this if we know the number of signals entering the functional block. We know the size of the group of signals in \( G_i \), but we do not know the size of the group of signals that will fan out of \( F_i \) into \( F_i \).

One approach to this problem is to examine the Shannon tree, since this has a direct physical interpretation, and each of the edges in the tree (graph) represent a wire in a viable technology implementation of the function. However, the Shannon tree suffers from a serious flaw: distinct cofactors in this tree may have multiple representations. Thus, we can not get an accurate estimate of the number of distinct cofactor terms.

Our solution to this problem is to use the Reduced Ordered Boolean Decision Diagram (BDD). A BDD can be viewed as a consolidated version of a Shannon tree, where identical subtrees are merged. Each distinct cofactor of the function is represented by a single partial path through the BDD, and hence represents the minimal number of connections needed in a circuit based on the supplied variable ordering. This property can be seen by comparing our example Shannon tree (figure 2) with figure 3. We use the BDD to give us an estimate on the number of signals communicating between functional blocks \( F_i \) and \( F_j \).

It is well known that the ordering of variables used when building a BDD can have a dramatic effect on its size. Hence, this would affect the number of edges in the graph, and subsequently our estimate of the number of signal communications between groups. However, we are only concerned with the number of signal wires between \( F_i \) and \( F_j \). The following theorem assures us that we need only be concerned with the content of the groups, and the group relative order. Any variable ordering consistent with the partitioning of the input signals will suffice.

Theorem 3.1 Let \( G_1 < G_2 < \ldots < G_n \) be a partition of the input variables defining a partial order of the input variables. Then the number of wires between function blocks \( F_i \) and \( F_{i-1} \) is independent of the order of the variables in each group \( G_i \), and is equal to the number of BDD nodes in group \( G_{i-1} \) which are terminus points for edges.
Note that the first step of our delay calculation framework outlined on page 2 is quite difficult. The development of the staggered network depends on the grouping of the input signals. Since we cannot know the exact grouping that might be used by the technology mapper, we can only try to estimate it. Finding the optimal grouping appears to be very difficult and we resort to the grouping heuristic presented in the next section.

4 Algorithm

4.1 Outline
The algorithm we use for calculating the unbalanced delay of a node follows the framework developed in section 3.1. The algorithm has four steps: First, the input signals are arranged in increasing order of arrival time, earliest to latest. Next, a BDD is created using the arrival time ordering. Third, the input signals are partitioned into groups using the BDD. Lastly, the BDD is used to calculate the size of the group of signals communicating between functional blocks, and the delay through each of the functional blocks. The total delay for the node is calculated using the delay of each of the functional blocks.

4.2 Grouping Heuristic
The object of the grouping heuristics is to form groups of input signals such that the delay through the entire node is minimal. The heuristic maintains a simple invariant: the output of a functional block should never arrive later than a fan-in input signal to the next group.

The grouping heuristic begins by placing the earliest arriving signal in a group by itself, and all the other input signals in a temporary group. We then consider the earliest arriving signal in this temporary group. Say this is the \( n \)th signal. The arrival time of the \( n \)th signal is compared with the arrival time of the output of the functional block which has as its input the \( n+1 \)th signal. If the arrival time of the \( n \)th signal is later than the output of the functional block, a new group is formed with the \( n \)th signal. This new group (and the output of its associated functional block) becomes the functional block to which we will subsequently compare the \( n+1 \)th signal. If the arrival time of the \( n \)th signal is earlier than the output of the functional block, the signal is placed in the group feeding the functional block.

5 Results
We compare the results of our technology independent delay model to the delays obtained from running the technology mapping system within MIS.

Our example circuits are from the MCNC benchmark suite. An algebraic script was run on each of the examples as well as decomposition with quick factoring before running either the mapper or our model. The MCNC LIB2 technology data was used for mapping the circuits. The scripts used for our results are shown in figure 4. The technology mapper used fanout optimization (-F), buffer removal (-A), and a tree mapper which optimizes for delay (-n 1). The print-delay command prints out the arrival times (-a) of the latest signal (-p 1) using the cofactor model (-m tdc) or the mapped circuit model (-m library).
We used the following parameters for our technology independent delay equation: $a_0 = 0.60$ and $a_1 = 0.75$. These parameters were chosen so as to minimize the average error for the circuits in the MCNC suite for the LIB8 technology library. Table 1 shows our results from circuits which had a mapped delay of greater than 10ns. The smaller examples are omitted for brevity, but their average error is 11.34% which is consistent with the average error for the large examples.

We report the delay of the mapped circuit (Mapped), the delay predicted by the Timing Driven Cofactor (TDC) model, and the CPU times in seconds for an IBM RISC System/6000. The mean error of the circuits with delay of at least 10ns is 11.04% shown in table 1.

We notice some anomalies: alupla and rd73-hdl have very similar technology mapped delays, but the error for each circuit from the TDC model is 27.41% and -14.68% respectively. We believe this is due to the influence of four effects:

- The technology mapper has the ability to optimize across node boundaries, while the TDC model only considers the delay for each node in isolation.
- The technology mapper uses fanout buffering which the TDC model does not consider.
- The accuracy of the model is dependent on the arrival time at each of the nodes. Small errors in the arrival times of nodes near the primary inputs of the circuit can have large effects on the overall delay predicted for the entire circuit.
- We model a potentially useful logic optimization transformation which would result in lower delays. This suggests that for some circuits, e.g. rd73-hdl and rd53-hdl, this type of structuring would speed up the circuit.

### Table 1: Comparison of Delay Predictions and Actual Delay

<table>
<thead>
<tr>
<th>Example</th>
<th>TDC Delay</th>
<th>CPU</th>
<th>Mapped Delay</th>
<th>CPU</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>5xpl-hdl</td>
<td>12.44</td>
<td>0.13</td>
<td>13.08</td>
<td>12.06</td>
<td>-4.92</td>
</tr>
<tr>
<td>rd54-hdl</td>
<td>12.90</td>
<td>0.15</td>
<td>15.19</td>
<td>13.51</td>
<td>-15.10</td>
</tr>
<tr>
<td>duc2</td>
<td>10.38</td>
<td>1.62</td>
<td>10.60</td>
<td>66.87</td>
<td>-1.06</td>
</tr>
<tr>
<td>misex3</td>
<td>17.68</td>
<td>5.52</td>
<td>17.32</td>
<td>126.00</td>
<td>0.91</td>
</tr>
<tr>
<td>rd84</td>
<td>12.44</td>
<td>1.37</td>
<td>12.70</td>
<td>62.16</td>
<td>-2.09</td>
</tr>
<tr>
<td>9sym-hdl</td>
<td>16.20</td>
<td>0.21</td>
<td>18.55</td>
<td>18.14</td>
<td>-13.66</td>
</tr>
<tr>
<td>misexsc</td>
<td>13.33</td>
<td>22.23</td>
<td>11.17</td>
<td>114.62</td>
<td>19.36</td>
</tr>
<tr>
<td>sao2-hdl</td>
<td>20.60</td>
<td>0.47</td>
<td>21.33</td>
<td>30.58</td>
<td>-3.44</td>
</tr>
<tr>
<td>9sym</td>
<td>13.53</td>
<td>11.96</td>
<td>10.51</td>
<td>59.27</td>
<td>28.78</td>
</tr>
<tr>
<td>fsl1m-hdl</td>
<td>12.44</td>
<td>0.12</td>
<td>13.54</td>
<td>11.22</td>
<td>-8.16</td>
</tr>
<tr>
<td>rd53-hdl</td>
<td>8.10</td>
<td>0.08</td>
<td>10.21</td>
<td>6.91</td>
<td>-20.63</td>
</tr>
<tr>
<td>9symml</td>
<td>11.15</td>
<td>5.67</td>
<td>10.40</td>
<td>56.00</td>
<td>7.16</td>
</tr>
<tr>
<td>alupla</td>
<td>17.09</td>
<td>0.42</td>
<td>13.41</td>
<td>21.67</td>
<td>27.41</td>
</tr>
<tr>
<td>rd73-hdl</td>
<td>11.55</td>
<td>0.11</td>
<td>13.54</td>
<td>10.79</td>
<td>-14.68</td>
</tr>
<tr>
<td>zdml-hdl</td>
<td>10.20</td>
<td>0.08</td>
<td>10.62</td>
<td>7.48</td>
<td>-3.90</td>
</tr>
</tbody>
</table>

6 Conclusion

The results show that the TDC model is approximately one order of magnitude faster than the technology mapper, which allows its interactive use in technology independent performance optimization algorithms. Considering that the TDC model is trying to predict the delay of a delay optimizing technology mapper, the error bound of ±20% is acceptable in many applications.

The TDC model predicts in several cases a delay significantly better than that obtained by the technology mapper (rd84-hdl, rd53-hdl, rd73-hdl). This may indicate that additional performance improvements can be obtained by using a logic restructuring technique similar to the cofactor operation used in the TDC mode.

Use of the model will follow two threads. First, there is application of our local cofactor approach in technology independent performance optimization algorithms such as speedup [7]. The second is in technology mapping, where more accurate predictions of delays may allow for better mapping strategies.

### References