Logical Effort Based Technology Mapping*

Shrirang K. Karandikar
Dept. of Electrical and Computer Engineering
University Of Minnesota
srirang@ece.umn.edu

Sachin S. Sapatnekar
Dept. of Electrical and Computer Engineering
University Of Minnesota
sachin@ece.umn.edu

Abstract

We propose a new approach to library-based technology mapping, based on the method of logical effort. Our algorithm is close to optimal for fanout-free circuits, and is extended to solve the load-distribution problem for circuits with fanout. On average, benchmark circuits mapped using our approach are 25.39% faster than the solutions obtained from SIS.

1 INTRODUCTION

The technology mapping step of synthesis binds a technology independent logic level description of a circuit to a library of gates in the target technology. A number of algorithms have been proposed for this step, such as tree-mapping [1] and DAG-mapping [2], using load-dependent delay models [3], constant delay models [4, 5] as well as using logical effort [6]. High-performance designs use rich libraries, with multiple instances of each cell, with varying delay, area and drive capabilities. Technology mapping, therefore, is not simply identifying the best cells to be used to implement some logic, but also the best instance of the selected cells.

In this paper, we apply logical effort [7, 8] to the problem of *minimum-delay technology mapping*. Our approach has two advantages over previous methods. First, the size of each gate in the solution is implicitly determined, and does not have to be considered during matching. Second, the delay model is inherently load-dependent, and there is no need to enumerate solutions for all possible load values, as is traditionally done [3]. This makes our approach faster than current algorithms for fanout-free circuits.

We also formulate and solve the *load-distribution* problem (described in Section 2), which occurs in the case of circuits with multiple fanouts. In [9], this problem was addressed in the context of sizing a mapped circuit. We use the approach presented there to guide the technology mapping algorithm at multiple fanout points in the circuit, leading to mapped circuits that have better performance than solutions obtained by previous methods.

2 BACKGROUND

2.1 Logical Effort

Logical effort [7, 8] has been widely used in a variety of application domains [5, 10, 11, 12] as well as in industry standard EDA synthesis tools [13, 14]. Using logical effort, the delay of a gate with input capacitance c_i is modeled by a linear function of the load c_l as:

$$D = g \times \frac{c_l}{c_i} + p \tag{1}$$

where g is the logical effort, $\frac{c_l}{c_i}$ is called the electrical effort and p is the parasitic delay of the gate.

*This work was supproted by SRC under contract 2001-TJ-884 and NSF under award CCR-0098117.

As shown in [8], the above equation can be extended to estimate the minimum delay, $\hat{D}$, of a *path* of logic as

$$\hat{D} = NF^{\frac{1}{N}} + P = N(GH)^{\frac{1}{N}} + P \tag{2}$$

where $F = GH$ is the path effort, P is the path parasitic delay and N is the number of gates on the path under consideration. The path logical effort, G, and path electrical effort, H, are obtained as the product of the gate logical and electrical efforts. Equivalently, H is the ratio of the output to input capacitances of the path. The minimum delay described by Equation (2) is obtained by distributing the path effort F equally to each gate on the path, if the parasitic delay P is ignored. Note that Equation (2) is only applicable to paths that have single-fanout gates.

2.2 The Load Distribution Problem

A two-step dynamic-programming algorithm for technology mapping based on tree covering was proposed in [1], and has served as the basis of later technology mapping algorithms. In the *matching* step, matches for all gates are generated, and the optimum match at each gate is stored as the solution for that gate, and in the *covering* step, the solution for the entire circuit is generated by an output-to-input traversal. Later approaches [3, 4, 5] improve on [1] by using more refined delay models that take into account the delay dependence of load, and the effect of multiple gate sizes. However, they do not address the load-distribution problem, described below.

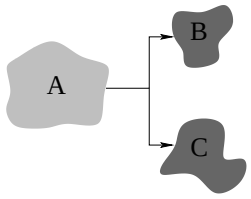

Figure 1. Technology Mapping at Multiple Fanouts

In a tree-mapping scenario, consider the situation shown in Figure 1, where some logic A has two fanouts, B and C, which eventually drive primary outputs (POs). Assume that A, B and C are fanout-free regions. The optimal solution for A depends directly on the load being driven, which, in this case, is the input capacitance of B and C. There are two situations that have to be considered:

Interactions between A **and its outputs** Assigning a larger input capacitance to B and C makes them faster, at the cost of increasing the load on A, and slowing it down, and vice versa. What is the optimum value of capacitance that should be assigned to the output of A, so that the delay of the *entire* circuit is minimized?

Interactions between B **and** C If these two fanout-free regions have completely different delays to the POs of the

circuit, we would like the critical branch to have a larger input capacitance. Conversely, if B and C have similar delays, they should have the same input capacitance. Thus, if we determine the optimal load that A should be driving, what is the best distribution of this capacitance to each fanout?

We refer to these two problems together as the *load-distribution problem*. Given a load at a multiple fanout point in the circuit, current algorithms can determine the best mapping for the logic up to that point. However, this load is typically estimated using heuristics, and since the mapped solution depends directly on the load being driven, wrong estimates can lead to sub-optimal solutions.

3 MAPPING USING LOGICAL EFFORT

In this section, we show how we can use logical effort to guide the selection of matches when mapping a circuit to a target library. We first show how fanout-free circuits can be mapped using logical effort, followed by our approach for multiple fanouts, where we provide a solution to the load-distribution problem. We finally summarize our overall approach for general circuits.

3.1 Mapping Fanout-Free Circuits

As is done in the traditional approach, we evaluate all matches at each gate in the subject graph. However, the cost we minimize is the cumulative path logical effort, G. First consider a simple path, with each gate having one fanin. As mentioned before, the path electrical effort, H, in Equation (2), can be calculated as the ratio of output to input capacitances of the path. i.e., if the electrical effort of a path is known, its delay can be calculated using Equation (2), *without knowing the sizes of each gate on the path*. If the path has a fixed number of stages, then for a given path electrical effort, the minimum delay over all possible implementations is obtained by the implementation that minimizes the path logical effort, G.

We now allow any number of stages for the implementation, and keep track of the optimal solution for each path length. Hence, we obtain a set of solutions at the PO, each of which implement the logic using a different path length. We can use Equation (2) to determine which of these gives us the minimal delay. Once the values of G, H and N that minimize the delay have been determined, the corresponding gate sizes can be calculated as described in [8].

We can now generalize this approach to circuits with gates having multiple fanins. A nice property of the logical effort formulation is that for paths of the same length, the path with maximum delay is also the one with maximum path logical effort, G. Hence, we can use the accumulated values of G at each input of a match to determine which input is the critical one. The cost of each match is defined to be the product of the logical effort of the match, and the *maximum* of the costs of its inputs. As before, the delay depends on the length of the path, N. We therefore record solutions for all values of path length at each gate, and at the POs, the best delay over all N can be selected, and the corresponding solution recovered. Thus, we trade off the traditional approach of calculating and storing solutions for all possible load values at each gate [3], with generating solutions for different values of path length, which is small in practice.

The pseudo-code of our dynamic-programming based approach is presented in Algorithm 1. For all legal values of lengths, each gate t keeps track of the accumulated product of logical efforts G_t, and the corresponding matches. G_t is indexed by the length of the path at the inputs of the match at gate t, plus 1 for the match at t itself. Assume that we are considering the match of a library pattern m at gate t, which has logical effort g_m and parasitic delay p_m, and that the length of the path from the PI to t is n. The cumulative logical effort of length n at input i of the match is $G_i[n]$. We select the maximum of this value over all inputs, and take its product with g_m, to obtain the cumulative logical effort at the output of t for a path of length $n+1$. Finally, the match and the cumulative parasitic delay $P_t[n+1]$ corresponding to the selected $G_t[n+1]$ are also stored.

Algorithm 1 `Mapping for Fanout-Free Regions`

```
// initialize
for each primary input (PI) p do
    G_p[0] = 1
end for
// Phase I: Matching
for each gate t in topological order do
    set G_t[n] = +∞ for all n
    // M_t is the set of all matches at t
    for each m ∈ M_t, with logical effort g_m do
        // I is the set of inputs to m
        // calculate cumulative effort G_t[n+1] from
        // the inputs, corresponding to distance n
        if g_m × max_{i∈I} G_i[n] < G_t[n+1] then
            G_t[n+1] = g_m × max_{i∈I} G_i[n]
            P_t[n+1] = p_m + P_i[n]
        end if
    end for
end for
// Phase II: Selecting Solution
at the PO, select the combination of G, H and N that minimizes delay
// Phase III: Covering
select matches in a traversal from the PO to PIs, sizing the matches appropriately
```

The optimality of Algorithm 1 is based on the following lemma (proof omitted due to space restrictions):

Lemma 1. *Selecting solutions of a gate t based on the input with maximum path logical effort, and sizing this solution based on the path effort hence determined, does not adversely affect the non-critical inputs of t.*

In Algorithm 1, the value of the cumulative effort for a match at a circuit node is calculated based on the previously stored optimal values at its inputs. Naturally, the value of cumulative effort at a node will be minimum only if the value at its inputs is minimum. This optimal substructure property of our formulation, along with Lemma 1, leads to an optimally mapped solution for the entire circuit[1].

3.2 A Solution to the Load Distribution Problem

We now address the general case of a circuit with multiple fanouts,

[1]The above discussion assumes that each input of a match has the same logical effort and input capacitance. Extensions to Algorithm 1 and a stronger version of Lemma 1 that can handle the general case are omitted due to space restrictions.

and present a solution to the load distribution problem. We treat the circuit as a collection of fanout-free regions. In this case, the critical input of a fanout-free region is not well defined, since the path having the maximum delay through the region may not lie on the critical path of the circuit. We therefore use a modified version of Algorithm 1, where instead of storing only one value of $G_t[n]$ at gate t, we store $G_{s_i \to t}[n]$, where $s_1, s_2, \ldots, s_k$ are the inputs of a fanout-free region ending in t (all of s_i and t are in the fanout-free region).

We now propose a solution to the load-distribution problem, by extending techniques developed in [9]. The circuit is initially divided into fanout-free regions, and matches for each of the fanout-free regions are generated as described above. The Delay-C_{in} curve of s_i, $D_{s_i \to PO}$, is the minimum delay of the critical path from s_i to some PO, for different values of input capacitance. The critical path may span multiple fanout-free regions of the circuit, and $D_{s_i \to PO}$ implicitly stores the optimal values of load at each multiple fanout point, as well as the optimal distribution of this load to each fanout.

The Delay-C_{in} curve of s_i is calculated as follows. At a PO, the Delay-C_{in} curve consists of a single delay value of zero, for the fixed load being driven. The circuit is traversed from the POs to the PIs, and therefore the Delay-C_{in} curves of each fanout of t are known. The load that t has to drive is the sum of the input capacitances of each of the fanouts. Since the Delay-C_{in} curves of each fanout have been calculated, for any fanout F_j, if we select a particular input capacitance, we immediately know the minimum delay of the critical path from F_j to a PO. The minimum delay of the critical path from s_i for some value of input capacitance $c_{in_{s_i}}$ to a PO is composed of the minimum delay of the path within the fanout-free region (i.e., the path from s_i to t) and the maximum delay from any fanout of t to a PO. Say we have some selection of input capacitances of each fanout of t, and since matching is complete, we can select the logical effort $G_{s_i \to t}$, electrical effort $H_{s_i \to t}$ and path length $N_{s_i \to t}$ that minimize the delay of the path from s_i to t. Adding to this value the maximum delay to the POs of any fanout F_j gives us the required critical path delay *for that selection of input capacitances of fanouts*. Repeating this for every combination of input capacitances of the fanouts and selecting the minimum delay thus obtained gives us $D_{s_i \to PO}$. Thus, by considering all combinations of fanout capacitances, we directly address the load-distribution problem.

Algorithm 2 shows how the Delay-C_{in} curve of input s_i of a fanout-free region terminating in t is calculated. Given an electrical effort, $H = \frac{C_L}{c_{in_{s_i}}}$, `Calculate` $DCurve_{s_i}$ calculates the best delay of a fanout-free region from all the solutions of different lengths that have been generated. This is used by `Calculate` $D_{s_i \to PO}$ to determine the best load, and the best distribution of this load to all fanouts, for the given input capacitance.

Consider the circuit shown in figure 1, and assume that B and C drive fixed loads at POs. For different input capacitances of B and C, we can calculate the minimum achievable delay, thereby obtaining the Delay-C_{in} curves at their inputs. At fanout-free region A, we need to consider all combinations of input capacitances of B and C. Each such combination is one possible value of load for A. For a particular load and input capacitance, we can calculate the minimum delay in A using `Calculate` $DCurve$. Combining this with the maximum of delays to POs through B and C gives us a possible point on the Delay-C_{in} curve of the input of A.

Algorithm 2 `Calculating the Delay-`C_{in}` Curves`

```
Calculate DCurve_{s_i}[C_L][c_{in_{s_i}}]
// s_i is the input, t is the output of the path
for all values of path length n do
    temp = n [ G_{s_i→t}[n] × C_L / c_{in_{s_i}} ]^{1/n} + P_{s_i→t}[n]
    if temp < DCurve_{s_i}[C_L][c_{in_{s_i}}] then
        DCurve_{s_i}[C_L][c_{in_{s_i}}] = temp
    end if
end for
```

```
Calculate D_{s_i→PO}[c_{in_{s_i}}]
// t has l outputs, F_0, F_1 ... F_l
for every combination of c_{in_j} of all fanouts F_j do
    if the selected combination is not redundant then
        C_L = Σ_{j=1}^{l} c_{in_{F_j}}
        Calculate DCurve_{s_i}[C_L][c_{in_{s_i}}]
        temp = DCurve[C_L][c_{in_{s_i}}] + max_{j=1...l} D_{F_j→PO}[c_{in_{F_j}}]
        if temp < D_{s_i→PO}[c_{in_{s_i}}] then
            D_{s_i→PO}[c_{in_{s_i}}] = temp
        end if
    end if
end for
```

Although the total number of combinations of $c_{in_{F_j}}$ is large ($O(\prod |c_{in_{F_j}}|)$, where $|c_{in_{F_j}}|$ is the number of possible values of input capacitance of F_j), it is shown in [9] that the number of combinations that actually have to be considered is much smaller, and is equal to $O(\sum |c_{in_{F_j}}|)$. The remaining combinations are redundant, and there is no loss of information by ignoring these.

This dynamic programming algorithm addresses both the components of the load-distribution problem. First, the *globally* optimal output and input capacitance for each fanout-free region is determined. Second, the best distribution of the output load into the input capacitances of the fanout-free regions being driven is determined.

3.3 Summary of Our Approach

The complete approach for logical effort based technology mapping addressing the load-distribution problem, called MELT (Technology Mapping using Logical Effort: the order of letters are suggestive of the multiple input-output-input traversals of the circuit required by our approach) is presented in Algorithm 3. After the first three steps, which have been described previously, we have Delay-C_{in} curves at the PIs of the circuit. At each PI, the load that minimizes the maximum delay to any PO is selected. The PIs are are processed in decreasing order of this delay. A forward traversal from the PIs using the selected loads fixes the input and output capacitances and the lengths of each fanout-free region. This information, in turn can be used to select the matches of the optimal solution.

In Algorithm 3, there are two issues that restrict the optimality of the final solution. First, the processing of each input of a fanout-free region is carried out independent of other inputs of this region. The solutions generated by different inputs may contradict each other. Second, circuits in general have reconvergent fanouts. The interaction between multiple, overlapping reconvergent paths is difficult

Algorithm 3 `MELT: Technology Mapping using Logical Effort`

Divide the circuit into fanout-free regions

PI→ PO Traversal: generate matches for each fanout-free region using Algorithm 1, storing optimal matches for each input of the fanout-free region

PO→ PI Traversal: calculate Delay-C_{in} curves for each input to the fanout-free region using Algorithm 2

PI→ PO Traversal: select the optimal electrical effort for each fanout-free region, and the corresponding lengths

Covering: use the assigned output and input capacitances to generate the corresponding optimal covers for each fanout-free region

to analyze efficiently. For both these cases, we use the heuristic of assuming that all paths are independent, and make the best choice available. The loss of optimality is acceptable when compared with the alternative of calculating the exact solution.

4 RESULTS

In order to validate our approach, we used MELT to map the ISCAS combinational benchmark circuits. These results were compared with SIS [15]. The library used for SIS was generated by calibrating INV, 2-, 3- and 4-input NAND and NOR, AOI- and OAI- 21, 211, 22, 221, 222, 31, 32, 33, XOR and XNOR gates on a 0.1μ technology using the Berkeley Predictive Technology Model[2] [16]. Multiple sizes of each gate were generated, for a total library size of approximately 400 elements. These gates were also calibrated in order to obtain the logical effort and parasitic delays, which constitute the library used by our algorithm, with 25 elements, one for each gate type. Once gate sizes for the mapped circuit are calculated using MELT, they are normalized to actual sizes available in the library.

The results obtained are as shown in Table 1. The first column lists the benchmark circuit. The next two, under the title SIS show the best delay obtained for each circuit using the command `map -n 1` in SIS, and the corresponding running time, T, in seconds. The performance of MELT for the same circuits is as shown. On average, our algorithm generates circuits that are 25.39% faster than those obtained using SIS. Interestingly, MELT also has an area improvement of 7.84%. During the covering step, the load at multiple fanout points is accurately known, and is usually higher than that estimated by SIS. Complex gates have better delay characteristics at higher loads, as compared to the equivalent using simple gates, and consequently MELT makes greater use of complex gates. Since complex gates tend to occupy less area than the equivalent circuit composed of simple gates, we observe an overall area improvement.

Table 1. Technology Mapping: SIS Vs. MELT

Circuit	SIS		MELT		% delay
	Delay(*ps*)	T(s)	Delay(*ps*)	T(s)	improv.
C17	79.91	0.02	65.72	0.02	17.76
C432	1611.36	0.65	795.93	1.96	50.60
C499	622.59	1.16	609.17	1.20	2.16
C880	656.54	1.07	643.43	1.21	2.00
C1355	686.39	1.50	678.19	2.35	1.19
C1908	1037.01	1.74	868.42	2.31	16.26
C2670	1885.50	2.52	868.82	3.93	53.92
C3540	2148.30	3.64	1218.21	6.20	43.29
C5315	1624.71	6.01	971.36	6.40	40.21
C6288	3020.81	6.87	2893.31	8.91	4.22
C7552	2098.08	7.35	1097.69	10.15	47.68
Average					25.39

4 REFERENCES

[1] K. Keutzer. DAGON: Technology Binding and Local Optimization by DAG Matching. In *Proc. IEEE/ACM DAC*, 341–347, 1987.

[2] Y. Kukimoto *et al*. Delay-Optimal Technology Mapping by DAG Covering. In *Proc. IEEE/ACM DAC*, 348–351, 1998.

[3] H. J. Touati *et al*. Performance-Oriented Technology Mapping. In *Proc. 6th MIT Conf. on Adv. Res. in VLSI*, 79–97, 1990.

[4] J. Grodstein *et al*. A Delay Model for Logic Synthesis of Continuously-Sized Networks. In *Proc. IEEE/ACM ICCAD*, 458–462, 1995.

[5] L. Stok *et al*. Wavefront Technology Mapping. In *Proc. DATE*, 531–536, 1999.

[6] B. Hu *et al*. Gain-Based Technology Mapping for Discrete-Size Cell Libraries. In *Proc. IEEE/ACM DAC*, 574–579, 2003.

[7] R. F. Sproull and I. E. Sutherland. Theory of Logical Effort: Designing for Speed on the Back of an Envelope. In *IEEE Adv. Res. in VLSI*, 1–16, 1991.

[8] I. Sutherland *et al*. *Logical Effort: Designing Fast CMOS Circuits*. Morgan Kaufmann, San Fransisco, CA, 1999.

[9] S. K. Karandikar and S. S. Sapatnekar. Fast Comparisons of Circuit Implementations. In *Proc. DATE*, 910–915, 2004.

[10] F. Beeftink *et al*. Gate-Size Selection for Standard Cell Libraries. In *Proc. IEEE/ACM ICCAD*, 545–550, 1998.

[11] W. Donath *et al*. Transformational Placement and Synthesis. In *Proc. DATE*, 194–201, 2000.

[12] K. Sulimma *et al*. Improving Placement under the Constant Delay Model. In *Proc. DATE*, 677–682, 2002.

[13] L. Stok *et al*. BooleDozer: Logic Synthesis for ASICs. *IBM Journal of Res. & Dev.*, 40(4), 407–430, 1996.

[14] Magma Design Automation. Gain Based Synthesis: Speeding RTL to Silicon, 2002.

[15] E. M. Sentovich *et al*. SIS: A System for Sequential Circuit Synthesis. Technical Report UCB/ERL M92/41, Electronics Research Laboratory, Dept. of EECS, University of California, Berkeley, May 1992.

[16] Y. Cao *et al*. New Paradigm of Predictive MOSFET and Interconnect Modeling for Early Circuit Design. In *Proc. IEEE CICC*, 201–204, 2000.

[2] Available from http://www-device.eecs.berkeley.edu/~ptm.