A Fast Fanout Optimization Algorithm for Near-Continuous Buffer Libraries

David S. Kung
IBM T. J. Watson Research Center
33-115, P. O. Box 218
Yorktown Heights, NY 10598 USA
914 945 3183
kung@watson.ibm.com

1. ABSTRACT

This paper presents a gain-based fanout optimization algorithm for near-continuous buffer libraries. A near-continuous buffer library contains many buffers in a wide range of discrete sizes and each buffer of a specific type satisfies a size-independent delay equation. The new fanout algorithm is derived from an optimal algorithm to a special fanout optimization problem for continuous libraries. The gain-based technique constructs fanout trees which have better timing at similar area cost. Since no combinatorial search over buffer sizes or fanout tree topologies is used, our execution time is up to 1000 times faster when compared to conventional fanout algorithms.

1.1 Keywords

Logic synthesis, gate-sizing, fanout optimization.

2. INTRODUCTION

Fanout optimization is the construction of a buffer tree to drive the large fanouts so that timing and capacitance violations are reduced to a minimum. Fanout optimization is a very effective logic transform but is typically the most time-consuming transform in technology-dependent timing optimization. It could be responsible for 70% of the slack improvement after an initial rough gate-sizing but might incur up to 50% of the time spent in timing optimization. Therefore finding a fast and effective fanout optimization algorithm deserves a high priority.

Unfortunately, the fanout optimization problem for libraries with discrete sizes has been proven to be NP-complete[1]. The majority of the previous work on this subject[2][3] relies on heuristics which assume some template for the buffer tree and then use some intelligent combinatorial search to select a good buffer size for each tree node and a good sink arrangement. These algorithms are polynomial in the number of sinks, the number of stages in the buffer tree and the number of buffers available in the library. As a result the runtime can be quite prohibitive when the number of sinks and the number of available buffers are large.

Reference [3] made the important observation that using a buffer library with almost continuous sizes greatly simplifies the fanout optimization problem. Armed with this insight, the strategy of this paper is as follows. Fanout optimization is first analyzed using a continuous buffer library. Since non-inverting buffers are usually constructed out of inverters, only inverters are assumed to be in the buffer library. *From now on, buffers and inverters are synonymous*. A continuous buffer library contains buffers which are available in continuous sizes and each buffer of a given type obeys a size-independent linear delay equation[5]

$$delay = p + l \cdot g \qquad (1)$$

where g (referred to as gain) is defined as $load/C_{in}$. p and l are independent of the size of the inverter and C_{in} is the input pin capacitance of the inverter. *In this paper, size is used interchangeably with input pin capacitance*. Using the concept of gain, a class of fanout optimization problem with a specific optimization objective is shown to be exactly solvable. A practical fanout optimization algorithm referred to as the gain-based algorithm is then derived from this optimal algorithm. The gain-based algorithm is applicable to near-continuous buffer libraries in addition to continuous buffer libraries. A near- continuous buffer library contains many buffers in a wide range of discrete sizes and each buffer of a given type obeys delay equation (1). Near-continuous buffer libraries can be realistically designed [6] and are gaining popularity as the buffer libraries of choice for high performance designs.

3. THE FANOUT PROBLEM

The inputs to the fanout problem are as follows:

- a fanout region which consists of a source and a set of N sinks, P sinks with positive polarity and Q sinks with negative polarity. Each sink, s_i, has a given polarity and drives a load, L_i. A timing constraint is imposed on each sink so that the delay from the source to sink, s_i, is less than or equal to D_i.
- a set of inverters with a continuous range of sizes.

The objective is to construct a fanout tree which presents the minimum load to the source such that the timing constraint at each sink is observed. This is in contrast to the standard optimization criterion which minimizes the area of the fanout tree so a discussion of the relevance of this alternative optimization criterion is in order. In high performance designs where delay optimization is critical, there is a tendency towards adopting a gain-based synthesis methodology[6]. We would like to argue that the minimum source load criterion is a reasonable objective within the gain-based framework.

In gain-based synthesis, library cells and gates in the network are parametrized by gain instead of size. The delay of a gate is a function of its gain and is independent of the load that it drives. Therefore it is relatively easy to find a gain

1-58113-049-x-98/0006/$3.50

assignment to satisfy the timing constraints. However, primary inputs have maximum load limits and each gate has a maximum size limit. These size constraints are difficult to satisfy. The size computation of a gate requires a propagation of the primary output loads through its fanout cone. The size of each gate in the fanin cone of the source gate (the gate that drives the source of the fanout region) is directly proportional to the source load of the fanout region. Minimizing the source load increases the chance of satisfying these size constraints.
This paper focuses on the minimum source load criterion and "optimal" in this paper carries the narrow sense of being optimal with respect to this criterion.

4. FANOUT TREES

To set the stage for theoretical analysis of fanout trees, several pertinent concepts will be defined.

Given two fanout trees T_1 and T_2, a sink s_i of T_1 is equivalent to a sink s_j of T_2 iff s_i and s_j have the same polarity, $L_i = L_j$, and the delay from the source of T_1 to s_i is equal to the delay from the source of T_2 to s_j.

T_1 is equivalent to T_2 iff

- $P_1 = P_2$ and $Q_1 = Q_2$,
- there exists a 1-1 correspondence between the sinks of T_1 and the sinks of T_2 under the equivalence relation,
- the load at the source of T_1 is equal to the load at the source of T_2.

The following lemma is an immediate consequence

Lemma 1

If T_1 is equivalent to T_2 and T_1 is an optimal solution to a fanout optimization problem, then T_2 is also an optimal solution to the fanout optimization problem.

Two important transformations on fanout trees, merge and split, are defined as illustrated by Fig. 1 where all the inverters have the same gain and the sinks of net5 is the union of the sinks of net2 and net3.

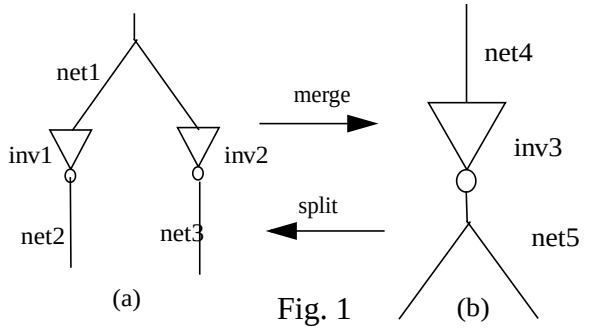

Fig. 1

Lemma 2

If T_2 is derived from T_1 by either the merge or the split transform, T_1 and T_2 are equivalent.

By repeated applications of the split transform, any fanout tree can be transformed into a tree for which each inverter has only one fanout. Fig. 2(a) is a fanout tree with 4 sinks of positive polarity and 1 sink of negative polarity and Fig. 2(b) is the fanout-free equivalent. Such a tree will be referred to as a *fanout-free tree* and is important for the following reason.

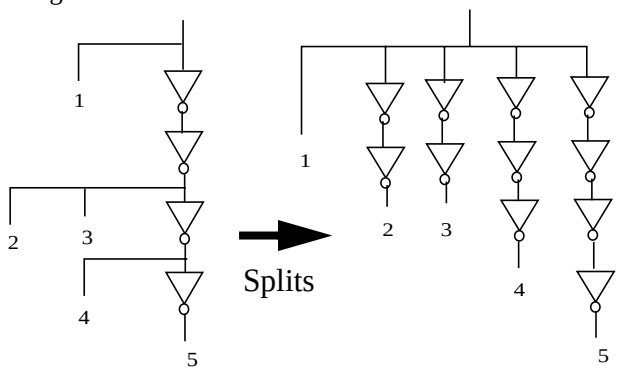

Fig. 2

Theorem 1

There exists an optimal fanout-free tree for a given fanout optimization problem.

Theorem 1 states that there is no loss of optimality by restricting the search to fanout-free trees. The next section focuses on finding the optimal fanout-free tree for a given fanout optimization problem.

5. FANOUT-FREE CHAINS

The fanout-free tree with N sinks is simply N fanout-free chains connected at the source. To find the optimal solution, it is sufficient to find the optimal solution to the following problem for each fanout-free chain:

Given a source, a sink with load L and a given polarity, and a source to sink delay constraint D, find the number of inverters, m, and their input pin capacitances, c_i, such that the input pin capacitance of the inverter connected to the source is minimized.

Let g_i be the gain of the i^{th} inverter on the chain with the first inverter being the one connected to the source, the problem is to minimize the capacitance

$$c_1 = L / \prod_1^m g_i \tag{2}$$

under the constraint

$$\sum_1^m (p + l g_i) \le D \tag{3}$$

At first glance, there seems to be no simple solution to this problem since there are $m+1$ variables (including m itself) to solve for. However using the inequality of the arithmetic and geometric means the following theorem can be derived.

Theorem 2

c_1 is minimized when $g_1 = g_2 = \ldots = g_m = g$

Armed with theorem 2, the optimal fanout-free chain problem is reduced to finding the g and m which minimize

$$c_1 = L g^{-m} \tag{4}$$

with the equality constraint

$$m = D / (p + lg) \tag{5}$$

Substituting (5) into (4), c_1 becomes

$$L g^{D/(p+lg)} \tag{6}$$

The minimum of c_1 occurs when g satisfies the following equation

$$\ln g = 1 + p/(lg) \quad (7)$$

Let the solution to (7) be γ and $\mu = D/(p + l\gamma)$. That means the optimal fanout-free chain has μ inverters and each inverter has gain γ. The size of each inverter can be easily derived from the sink load L and γ.

There is a caveat to this solution because μ is most likely non-integral. In reality a feasible solution for m must be an integer and must be even for positive polarity sinks and must be odd for negative polarity sinks. The procedure to find the optimal fanout-free chain is modified as follows:

- calculate the optimal γ and μ (could be fractional) as above,
- round off μ to the two nearest feasible integers accounting for polarity
- if $\mu < 1$ and sink polarity is negative the only feasible integer is 1, in which case the optimal fanout-free chain contains 1 inverter with gain $(D-p)/l$, otherwise
- let the 2 integers be μ_1 and μ_2, so the respective gains are $\gamma_1 = (D-\mu_1 p)/l$ and $\gamma_2 = (D-\mu_2 p)/l$
- assume $m = \mu_1$ and $g = \gamma_1$ is the solution with a smaller c_1
- the optimal fanout-free chain has μ_1 inverters each with gain γ_1
- the size of each inverter is computed from L and γ_1.

This fanout-free chain is optimal because c_1 as a function of m, $L\left(\frac{D-mp}{lm}\right)^{-m}$, is convex for $0 \le m \le D/p$. The optimal fanout-free tree is obtained by finding the optimal solution for each of its fanout-free chains. The optimal gain value for each chain could be different. The optimal fanout-free tree is unacceptable as a buffer tree in practice because the first inverter of each chain might have a unrealistically small input pin capacitance. The next section discusses how to derive a practical fanout tree from this optimal solution.

6. THE FANOUT ALGORITHM

In this section we utilize the theoretical results from the previous sections to design a practical fanout optimization algorithm. In practice, inverter libraries do not offer continuous sizes and there are lower and upper bounds to the input pin capacitances. The fanout tree topology sought for is a tree with internal fanouts so that the sizes of the inverters fall into a realistic range. We learn from section 4 that if the gains assigned to each chain of the fanout-free tree are the same then the merge operation can be applied to create a desirable topology. Moreover, the optimal gain for each chain in the fanout-free tree is most likely not very different from γ because the absolute magnitude of the round-off in m is less than 2 and most of the times less than 1. Therefore a common gain close to γ assigned to each chain is likely to result in a near-optimal fanout-free tree. This observation leads to the concept of a *uniform gain fanout-free tree*.

The uniform gain fanout-free tree with respect to a fanout problem input and a uniform gain g is constructed as follows:

- for each sink in the fanout problem
  - compute μ by substituting g into (5)
  - round off μ to the biggest integer m less than μ accounting for polarity and such that the delay constraint for that chain is observed.
  - construct an inverter chain with m inverters each with gain g

The fanout optimization algorithm proceeds as follows:

- compute the solution, γ, to (7) given p and l from the inverter library,
- let G be a set of g's in the neighborhood of γ, e.g. if $\gamma = 3.4$ choose G = {3.0, 3.2, 3.4, 3.6, 3.8}
- for each g in G, construct a fanout tree using the following 5 steps
  - construct a uniform gain fanout-free tree with respect to the problem input and the uniform gain g
  - perform the merge operation iteratively on the uniform gain fanout-free tree until no merging is possible
  - from g and the sink loads compute the size of each inverter in the tree
  - locate the inverter in the library whose size is closest to the computed size
  - replicate inverters if the maximum inverter size available is exceeded
- choose the best fanout tree among the ones constructed

The gain-based algorithm is distinctly different from prior arts in that once a gain is selected, no combinatorial search is required in calculating the buffer sizes and the sink arrangement. The complexity of the algorithm is analyzed as follows. For each sink (5) is solved, from which the number of buffers for the tree is determined. Both operations are simple arithmetic operations and performing them on all the sinks requires complexity *O(N)*. Then each buffer in the fanout tree is selected by matching the closest size, incurring another *O(N*ln(K))* (K is the number of inverters in the near-continuous inverter library) since in the worst case the number of buffers is proportional to the number of sinks. In reality the complexity is likely to be *O(N)+O(ln(K))*, since the number of buffers is approximately equal to the number of levels of buffering.

7. EXPERIMENTS

The fanout optimization algorithm is applied to 19 fanout cases and compared with the fanout correction transform in BooleDozer which uses LT trees[2] as templates and combinatorial search to find good size assignment and sink arrangement. Each testcase consists of a fanout region with the primary input pin being the source and the primary output pins being the sinks. This is preferred over performing fanout optimization on logic networks, in order to isolate the effect of fanout optimization. The fanout testcases are specified in a similar fashion as the fanout problem inputs discussed in section 3. Testcase (*Q, P*) refers to a fanout with *P* positive sinks and *Q* negative sinks. The first 10 testcases are randomly generated examples designed to stretch the limits of fanout optimization algorithms. The arrival time at the source is set to 0.0 while the sink required time is a random distribution over a prescribed range. The sink loads are also a random distribution over the range of input pin capacitances of the cells available in the library. The remaining 9

testcases are fanout regions extracted from industrial high performance designs. The library is discrete and has 30 inverters with varying sizes. For a fixed input slew, the linear delay equation (1) is a very good approximation for each inverter in the library.

The results are listed in Table 1. “slack” is the slack at the source of the fanout tree, “area’ is the area of the entire fanout tree and “cpu” is the run time on a 200MHz PowerPc workstation. The new fanout optimization algorithm is better in timing across the board with only a 1% increase in overall area. However, the biggest advantage is the runtime performance. Conventional combinatorial methods consume enormous computation time when the number of buffers becomes large (> 10) and when the number of sinks becomes large. The gain-based algorithm is more than 1000 times faster in such cases.

	combinatorial			**gain-based**		
model	slack	area	cpu	slack	area	cpu
(5,5)	-0.01	36	13.8	0.01	32	0.12
(12,16)	0.05	60	45.8	0.05	48	0.14
(32,8)	-0.04	48	33.0	-0.01	86	0.20
(16,64)	-0.08	194	547	-0.08	172	0.40
(8,128)	0.0	266	1931	0.04	282	0.63
(28,34)	-0.06	120	189	-0.06	124	0.32
(56,44)	-0.35	210	280	-0.17	216	0.53
(12,5)	-0.04	32	5.53	-0.03	46	0.16
(7,31)	0.04	84	113	0.04	74	0.21
(18,47)	-0.03	140	272	-0.01	124	0.29
(12,20)	-0.24	28	22.6	-0.23	28	0.31
(15,23)	-0.31	34	24.1	-0.26	30	0.39
(16,17)	-0.16	28	23.2	-0.14	28	0.43
(16,18)	-0.11	28	23.4	-0.09	30	0.47
(20,17)	-0.19	26	20.0	-0.16	30	0.29
(21,12)	0.09	22	11.0	0.12	32	0.41
(25,21)	-0.37	30	32.6	-0.32	38	0.52
(6,28)	-0.52	48	27.2	-0.50	24	0.38
(8,24)	-0.45	36	22.3	-0.42	24	0.31

Table 1

8. CONCLUSION

This paper has presented a very fast and effective fanout tree construction algorithm for a near-continuous buffer library. The algorithm is applicable to buffer libraries with discrete sizes as long as there is a wide range of buffer sizes and the size independent delay equation (1) is obeyed. Since fanout optimization is arguably the most important transform in timing optimization, the quality of logic synthesis can be substantially enhanced by this new algorithm. This leads to the issue of how a buffer library should be designed. Many buffer libraries are designed with only several (less than 5) buffers for each buffer type and buffers are constructed out of multi-stage inverters to increase the drive capability. These buffer designs make sense when fanout optimization algorithms in logic synthesis consist of combinatorial heuristics whose run time are unacceptable if the number of buffers is large. With the new algorithm, it is beneficial to design buffer libraries in the near-continuous style and allow synthesis the extra degrees of freedom to construct good fanout trees.

9. ACKNOWLEDGMENT

The author would like to thank L. Trevillyan, D. Brand, L. Stok and R. Puri for reading the manuscript.

10. REFERENCES

[1] C. L. Berman, J. L. Carter and K. F. Day, “The Fanout Problem: From Theory to Practice”, Advanced Research in VLSI: Proceedings of the 1989 Decennial Caltech Conference, C. L. Seitz, editor, MIT Press, 1989, pp. 69-99.

[2] K. J. Singh and A. Sangiovanni-Vincentelli, “A Heuristic Algorithm for the Fanout Problem”, Proceedings of the 27th DAC, 1990, pp. 357-360.

[3] K. Kodandapani, J. Grodstein, A. Domic and H. Touati, “A Simple Algorithm for Fanout Optimization using High-Performance Buffer Libraries”, Proceedings of ICCAD, 1993, pp. 466-471.

[4] L. Stok et. al. “BooleDozer: Logic synthesis for ASICs”, IBM Journal of Research and Development, vol. 40, no. 4, July 1996, pp. 407-430.

[5] I. Sutherland and R. Sproul, “The Theory of Logical Effort: Designing for Speed on the Back of an Envelope”, Advanced Research in VLSI, Univ. of Calif. Santa Cruz, 1991.

[6] K. Shepard et. al. “Design Methodology for the S/390 Parallel Enterprise Server G4 microprocessors”, IBM J. Res. Development, vol. 41 (1997), pp. 515-547.