Improved Logic Optimization Using Global Flow Analysis Extended Abstract

Leonard Berman and Louise Trevillyan

IBM T. J. Watson Research Center
Yorktown Heights, New York 10598

Abstract

This paper is concerned with techniques for automatically reducing circuit size and improving testability. In an earlier paper [2], we introduced a new method for circuit optimization based on ideas of global flow analysis. In this paper, we describe two extensions to the method. The first is a basic improvement in the primary result on which the earlier optimization was based, the second extends the applicability of the method to "conditional" optimizations as well. Together these enhancements result in improved performance for the original algorithm, as well as the ability to handle designer specified "don't cares" and redundancy removal uniformly in the framework of a graph based synthesis system such as LSS.

Overview

There are two basic approaches to multi-level design: 1) the algebraic/boolean approach [6] , and 2) the graph-based approach [4]. Although systems generally incorporate ideas from both approaches, it is fair to say that the boolean/algebraic method typically represents a function as a directed acyclic graph (dag) *whose nodes compute arbitrary functions* and performs optimizations using factoring and 2-level minimization on the nodes. On the other hand, the graph based approach represents a function as a dag *whose nodes compute simple functions* and performs optimizations using graph manipulation and data flow algorithms. This paper describes a significant enhancement to an optimization which belongs to the conceptual and algorithmic framework of the graph based synthesis approach. We note that there has been progress towards applying these ideas in the framework of algebraic synthesis[7].

In [2] the authors introduced a new circuit optimization which used ideas from global flow analysis to optimize a circuit by first computing circuit summary information and then using this information via the MIN/CUT algorithm to reduce the number of connections in a circuit. This paper describes improvements to this method. There are two main contributions. The first is an improvement of the main theorem of [2] which guaranteed that the optimization was legal, i.e. that it left the function of the circuit unchanged. The second contribution is a refinement of the method that allows "conditional optimizations"[1] to be handled uniformly. Together, the extensions reported here result in a significant conceptual, as well as practical, simplification to the logic optimization phase of LSS. This phase, which until recently included more than a dozen local transformations, can now be described entirely in terms of two primitive processes: global flow methods for boolean logic optimization, and factoring to satisfy fan-in/fan-out constraints.

The global flow method is described in detail in [2, 3]. At a high level it can be described as follows. Iteratively, for each net the procedure determines how the terminals of the current net can be rearranged and chooses a legal rearrangement for implementation. These legal rearrangements are determined assuming the net carries the controlling value[2] and reflect the boolean nature of the operators (as opposed to the algebraic nature of factoring). These possible rearrangements are determined using summary information which consists of assertions concerning the state of other nets in the circuit. The correctness of the rearrangements is guaranteed by Theorem 1 of [2] which states that rearrangements of the terminals of a net that leaves the "FRONTIER" unchanged are legal. (These concepts will be defined precisely later.)

Example:We use the circuit in figure 1 to illustrate throughout. When the procedure considers signal "i", it performs deductions under the assumption that **i=1** since 1 is the controlling value for NOR gates. It derives the assertions: **n=0, m=0, j=0**[3]**, p=1, r=0, s=0, q=0.**[4] It then determines that FRONTIER(i)={**q=0, r=0**}[5]. The algorithm then changes the terminals of "i" to produce the circuit shown in figure 3.

Since the optimizations involve rearranging the terminals of the net under consideration, it is important to differentiate between assertions which are independent of the precise connections of the current net and those which depend on where this net goes. Our first result addresses this distinction. In our earlier paper, this distinction was captured to some extent in the definition of the frontier. In this paper, this distinction is refined through the idea of an assertion concerning the circuit being "FREE". The earlier theorem is extended to show that any rearrangement which leaves the non-"FREE" part of the frontier unchanged is legal. This results in significant connection reduction in some examples. Note that in the above example, the connection of "i" at the source of s is not necessary because the assertion **q=0** is independent of

[1] These conditional optimizations include connection reduction optimizations which use designer specified output "don't care" information [5] (p.235-236), as well as redundancy removal optimizations.

[2] For a network of NORs, the value '1' is the "controlling value" since it controls the output of the logic gate, and blocks the ffect of other inputs.

[3] Since $j=1 \Rightarrow i=0$ and therefore $i=1 \Rightarrow j=0$.

[4] These correspond to $C_{10}(i) = \{n,m,j,r,s,q\}$ and $C_{11}(i) = \{p\}$ in the notation of [2].

[5] This corresponds to FRONTIER(i)={q,r} in the notation of [2].

where "i" is connected. It is this type of information which is captured in the notion of a "FREE" assertion.

The second enhancement to the method results from an improvement in the formalism which permits us to reason about individual terminals rather than entire nets. This does not require any fundamental change to the techniques, but it does permit "conditional" optimizations to be accomadated in the same theoretical framework. These conditional optimizations include connection reduction optimizations[6] which utilize designer specified output "don't care" information (this has not been possible before in graph-based synthesis systems) and redundancy removal optimizations.[7]

Summary of Technical Material

Terminology and Foundations

Circuits are represented as dags. For simplicity, we assume the nodes of the dag to be NORs, INPUTs, or OUTPUTs. We use terms gate, node, signal, and terminal in the standard way. Since we assume each gate has a single output, we will also identify the net with the node which is its source. In what follows $\mathscr{C}$ is an arbitrary circuit satisfying the above constraints.

Definition:A $=\{(c_i,v_i) \mid c_i$'s are terminals in $\mathscr{C}$ and v_i 's are associated values$\}$ is called a ***partial assignment of*** $\mathscr{C}$.

We can think of a partial assignment as a set of assertions (or deductions) about the state of the circuit. For example, if $(c,1) \in$ A, we may say that A contains the assertion **c=1**, or equivalently that A asserts that terminal c has value 1. We will write partial assignments as sets of pairs or sets of assertions interchangeably. Note that a partial assignment, A, determines a set of possible input values, i.e. all input choices which result in the assertions of A being true. We shall refer to such inputs as ***input assignments of A***.

Definition: For any partial assignment A of $\mathscr{C}$, let $A^{\mathscr{C}}$ denote the partial assignment consisting of all assertions implied by A. This is called the ***closure of A***.

Although the results described here are independent of the method used to derive these assertions, for the sake of clarity in this abstract, we use the following recurrences which we introduced in [2] as our deductive system. (Let $X(s) = \{\textit{inputs of source of s}\}$.)

$$\begin{aligned}
C_{10}(A) = {} & \{(s,0) \mid \exists y \in X(s)[(y,1) \in C_{11}(A)]\} \cup A \\
& \cup \{(s,0) \mid \exists (y,1) \in C_{11}(A)[s \in X(y)]\} \\
& \cup \{(s,0) \mid \exists x[(x,1) \in A,(x,0) \in C_{10}(\{(s,1)\})]\} \\
& \cup \{(s,0) \mid \exists c \text{ on same net as s},(c,0) \in C_{10}(A)\} \\
C_{11}(A) = {} & \{(s,1) \mid \exists y,(y,0) \in C_{10}(A), s \in X(y), \\
& \qquad \forall t \in X(y)[t \neq s \Rightarrow (t,0) \in C_{10}(A)]\} \\
& \cup \{(s,1) \mid \forall y \in X(s)[(y,0) \in C_{10}(A)]\} \cup A \\
& \cup \{(s,1) \mid \exists x[(x,1) \in A,(x,0) \in C_{00}(\{(s,0)\})]\} \\
& \cup \{(s,1) \mid \exists c \text{ on same net as s},(c,1) \in C_{11}(A)\}
\end{aligned}$$

Similarly for C_{00}, C_{01}.

The use of this deductive method results in the closure, $A^{\mathscr{C}}$, being the least fixed point of these recurrences. We note that Hachtel [9] uses related deductive methods while Brayton [7] takes a different approach.

The definition of partial assignments permits different values to be associated with different terminals of a net. Such a partial assignment corresponds to inconsistent assertions about the state of the circuit. Inconsistent assignments are ruled out through the notion of a compatible assignment.

Definition:A partial assignment A is said to be ***compatible in*** $\mathscr{C}$ if whenever (c_j,v_j) and $(c_i,v_i) \in A^{\mathscr{C}}$, and c_j and c_i refer to terminals of the same net, then $v_j = v_i$.

The following notation will be useful. For any compatible A, we let $\mathscr{C}\backslash$A be a circuit identical to $\mathscr{C}$ but with all terminals of A removed, and each terminal of A which is the input to a gate replaced by a connection to a new primary input. Also let B(A) = $\{(c,v) \mid (c,v) \in A^{\mathscr{C}}$ and there is no path in $\mathscr{C}$ from any terminal in A to any terminal on the same net as c$\}$.

Observe that B(A) contains assertions which are dependent on how the signalin A are computed but independent of how they are used. This is true because any assignment to inputs that results in the terminal-value pairs of A, must also result in the pairs contained in B(A). In fact, for any input assignment of A, if $\mathscr{C}$ is changed by rearranging terminals in A (subject to some restrictions), the assertions of B(A) still hold. This is because no path exists from A to B(A) and so there can be no direct dependence of these assertions on the terminals in A. We illustrate this with the following.

Example:Consider the circuit from figure 1, and assume that the partial assignment, A, under consideration sets all terminals of signal "i" to 1. We see that $A^{\mathscr{C}}$={(n,0), (m,0), (j,0), (p,1), (r,0), (s,0), (q,0)} with each signal-value pair replaced by all the appropriate terminal-value pairs. We also see that B(A) contains the assertions **n=0, m=0, j=0, p=1**, again with signals replaced by the appropriate terminals.

This independence is important and leads us naturally to our main definition.

Definition:FREE(A) = $B(A)^{\mathscr{C}\backslash A}$.

Example:If we continue the earlier example, we see that FREE(A) contains **q=0** as well as the assertions **n=0, m=0, j=0, p=1** which were also in B(A).

We see intuitively that the assertions in FREE(A) are consistent with $A^{\mathscr{C}}$ and also independent of A. Consistency is established by:

Lemma: If A is compatible then

$$\{c \mid \{(c,0),(c,1)\} \subset \{FREE(A) \cup A^{\mathscr{C}}\}\} = \phi.$$

Proof: From the definition of the closure operator, we see that for any set of assertions X and partial assignment A, $X^{\mathscr{C}\backslash A} \subseteq X^{\mathscr{C}}$ and that $B(A)^{\mathscr{C}} \subseteq A^{\mathscr{C}}$. Combining these observations shows that $FREE(A) \subseteq A^{\mathscr{C}}$, and since A is compatible, the result follows.■

Intuitively, the independence of FREE(A) follows from the construction of $\mathscr{C}\backslash$A. By replacing connections of terminals of A which are inputs of gates by connections of new primary inputs, we prevent the establishment of any assertions which depend n the precise terminals of A.

Another idea which we need is that of the frontier of a signal in a set of nodes. Intuitively, the frontier is the subset of nodes closest

[6] Connection reduction optimizations are those logic optimizations which leave the function unchanged at the most forward changed nodes.

[7] Redundancy removal optimizations are those where the function may be changed at the most forward changed node; however, the change can not be seen at any output because of the circuit structure.

to the outputs. More formally, Given a signal i in $\mathscr{C}$ and a partial assignment, A, we define the **frontier of i in A, $\mathscr{F}$**(A,i), as the set of nets, j, for which:

- $(j,0) \in A^{\mathscr{C}}$,
- there is a path $j \rightarrow j_1 \rightarrow j_2 ... \rightarrow OUTPUT$ such that for no j_i is $j_i \in A^{\mathscr{C}}$,
- j is reachable in the circuit from i.

If we say frontier of a net, we are referring to the partial assignment which assigns '1' to all terminals of that net.

The importance of the sets $\mathscr{F}$(A,i) and FREE(A) is illustrated by the following theorem.

Main Theorem: Let i be a net in $\mathscr{C}$ and A any partial assignment which sets i to 1. Let $\mathscr{C}'$ be identical to $\mathscr{C}$ except that terminals of i in A may be missing and additional terminals of i may be present. Assume that none of the connections of i in $\mathscr{C}'$ are to nodes in FREE(A). If in the two circuits, the sets $\{x \mid x \in \mathscr{F}(A,i) \wedge x \notin FREE(A)\}$. are identical, then the two circuits compute the same function.

This theorem is very similar to the main theorem of [2]. In fact, it is established by showing that in this case, the hypotheses of our earlier theorem hold. In our earlier work, we could guarantee that a rearrangement was legal only if the frontier sets in the two circuits were identical; while to apply this theorem, we require only that the non-"FREE" part of the frontier sets be identical. Since connections must be added to maintain the equivalence of these two sets, we see that our new result can, in principle, result in smaller circuits. We have found this to be true in practice as well.

Example:Continuing the example, we saw earlier that $\mathscr{F}$(A,i)=**{q=0,r=0}**. If we combine this with the value for FREE(A) computed above we see that $\{x \mid x \in \mathscr{F}(A,i) \wedge x \notin FREE(A)\}$ =**{r=0}**. The above theorem then guarantees that the circuit shown in figure 2 is equivalent to that shown in figure 1. Note that in figure 2 signal i has only 2 terminals. As mentioned earlier, our previous method which did not make use of "FREE" connections would result in the circuit shown in figure 3 in which net "i" has three terminals. We realize that this is a simple example which could be handled by other less sophisticated methods. It is meant only to illustrate the improvement in our new optimization.

As presented here, the method appears to be computationally intensive since to compute the "FREE" connections, we must perform deductions for a new circuit. However, we note that a good approximation, to the deductions embodied in the recurrences shown in the appendix, can be computed on a signal by signal basis. This is done by a straightforward use of data flow propagation techniques [1, 10] which never computes information that will be invalidated before it is needed. The running time of this procedure is proportional to the product of the size of the controlling sets and the average fan-in of the circuit. These techniques enable us to compute the "FREE" connections with minimal extra cost during the same graph traversal.

Experimental Results

We performed a number of experiments to evaluate the impact of our main theorem on the effectiveness of global flow optimization. We did this by creating two programs, one of which used our main theorem and one of which relied on Theorem 1 of [2]. Both programs utilized the theorems through the artifact of derived graphs. The method used to construct the derived graphs was somewhat different from that described in [2]; however, identical methods were used in both programs. In addition, because of the computational constraints alluded to above, we did not compute the entire fixed point of the recurrences; rather, we weakened the recurrences by dropping the term corresponding to the contrapositive throughout the experiment. When computing the results based on [2], we weakened the recurrences even more by including only the term corresponding to forward propagation. The result of these two approximations can only exaggerate the possible benefit due to our main theorem, and therefore, our results should be considered as an upper bound on the usefulness of this idea. We feel strongly that other experiments are needed to evaluate the benefit which might be gained by utilizing the contrapositive or even using the entire F_{ij} sets[2].

Our experiments began with logic which had been run through the high-level and and/or level optimizations of LSS and then translated to NORs. (See [4, 8] for details about these optimizations.) We then ran one of the two programs: FREEOPT, which utilized our main theorem, or WEAKOPT, which was based on our earlier methods. These were followed by some "tidying up" programs which propagate constants, remove common sub-expressions, eliminated double negations, etc.. We did not perform fan-in or fan-out correction.

Our most surprising result was that in one of the 15 cases for which comparisons were run, the size of the NOR level circuit produced by WEAKOPT was smaller than that produced by FREEOPT. This must be due to the interaction of successive applications of the optimization and suggests that the effect of signal ordering in the sequence of applications is important. Currently, we treat high fan-out signals first. Other than this, results were in accordance with our expectations. In many cases the two methods were identical. In those where there was a difference, there was a wide spread in effect; the improvement varied from 25% to 100%[8]. This large variance is not too surprising; it suggests that if the style of specification is such that the main theorem applies, it may apply frequently.

Other Applications

As mentioned earlier, the refinement in our formalism extends the applicability of our method to conditional optimizations. This permits us to make use of "don't care" conditions directly. We do this by adding a new function, which recognizes the appropriate "care" set, to the circuit and a new pair setting the output of this function to '1' to partial assignments used for optimization. The resulting optimizations will be valid on all inputs in the "care" set. Both output don't care's and redundancy removal can be accommodated in this framework.

Summary and Conclusion

In this paper we describe two enhancements to our earlier global flow algorithm for connection reduction. We show how to make better use of all types of "don't care" information, and we show how to apply our methods to redundancy removal optimizations. From a practical standpoint, these enhancements result in superior performance for the algorithm. From a conceptual standpoint, they unify a wide variety of optimizations which have been part of the Logic Synthesis System.

[8] The percentage improvement was computed as the ratio of the number of connections removed by each program.

The work presented suggests a number of avenues for further research:

1. Develop an incremental or on-line algorithm which maintains the full controlling sets.
2. Determine the effect of using the forcing sets or the full controlling sets in global flow. (See [9] for a beginning.)
3. Investigate the effect of signal ordering on global flow.

Acknowledgements

We would like to thank Larry Carter and Andrea LaPaugh, without whom this might have been written but could never have been read.

References

1. F. E. Allen and J. Cocke, "A Program Data Flow Analysis Procedure," *CACM*, vol. 19, no. 3, pp. 137-147, March 1976.
2. L. Berman and L. Trevillyan, "A Global Approach to Circuit Size Reduction," *Advanced Research in VLSI, 5th MIT Conference*, pp. 203-214, Cambridge, MA: MIT Press, March 28-30 1988.
3. L. Berman, L. Trevillyan, and D. Brand, "Applications of Global Flow Analysis in Logic Synthesis," *Proceeding of 1988 Int. Symp. on Circuits and Systems*, Helsinki, Finland, June 7-9 1988.
4. D. Brand, "Logic Synthesis," *Design Systems for VLSI Circuits: Logic Synthesis and Silicon Compilation*, pp. 301-326, Martinus Nijhoff, 1987.
5. R. K. Brayton, "Algorithms for Multi-Level Logic Synthesis and Optimization," *Design Systems for VLSI Circuits: Logic Synthesis and Silicon Compilation*, pp. 197-248, Martinus Nijhoff, 1987.
6. R. K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A. R. Wang, "MIS: A Multiple-Level Logic Optimization System," *IEEE Trans. on CAD*, vol. CAD-6, no. 6, November 1987.
7. R. K. Brayton, E. M. Sentovich, and F. Somenzi, "Don't Cares and Global Flow Analysis of Boolean Networks," *Proceedings of the ICCAD*, November 1988.
8. J. A. Darringer, W. H. Joyner, Jr., C. L. Berman, and L. Trevillyan, "Logic Synthesis Through Local Transformation," *IBM Journal of Research and Development*, vol. 25, no. 4, pp. 272-280, July 1981.
9. G. Hachtel, R. Jacoby, P. Moceynas, and C. Morrison, "Performance Enhancements in BOLD using "Implications"," *Proceedings of the ICCAD*, November 1988.
10. L. Trevillyan, W. Joyner, Jr., and C. L. Berman, "Global Flow Analysis in Automatic Logic Design," *IEEE Trans. on Computers*, vol. C-25, no. 1, January 1986.

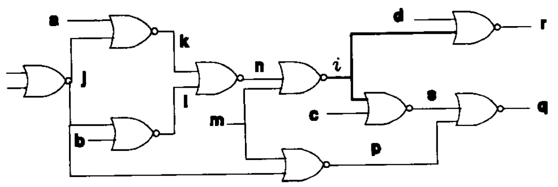

Figure 1

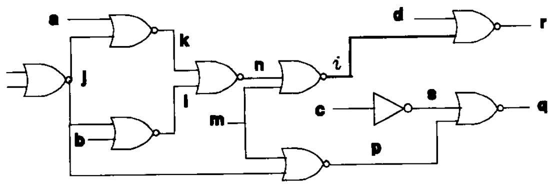

Figure 2

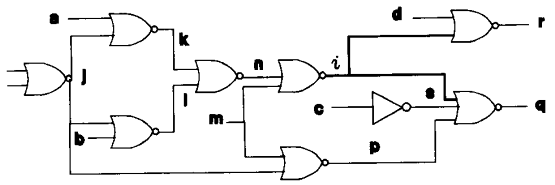

Figure 3