Wire Reconnections Based on Implication Flow Graph

Shih-Chieh Chang, Zhong-Zhen Wu, and He-Zhe Yu
National Chung-Cheng University, Chia-Yi, Taiwan R.O.C.

Abstract

Global Flow Optimization (GFO) can perform the fanout/ fanin wire re-connections by modeling the problem of the wire re-connections by a flow graph and then solving the problem using the maxflow-mincut algorithm on the flow graph. However, the flow graph cannot fully characterize the wire re-connections which causes GFO to lose optimality on several obvious cases. In addition, we find that the fanin re-connection can have more optimization power than the fanout re-connection but requires more sophisticated modeling. In this paper, we re-formulate the problem of the fanout/fanin re-connections by a new graph called the implication flow graph. We show that the problem of wire re-connections on the implication flow graph is NP complete and also propose an efficient heuristic on the new graph. Our experimental results are very exciting.

1 Introduction

Unlike most ATPG algorithms [2][3][5][6] which add only **one** redundant wire and remove other redundancies at one iteration, Global Flow Optimization (GFO) [1] and dual Global Flow Optimization [4] can simultaneously add and remove many redundant wires at a time. Let *s* be a target node under consideration. The GFO technique [1] attempts to re-connect the immediate fanouts of node *s* to the inputs of other nodes, and the dual GFO technique [4] re-connects the immediate fanins of node *s* from the outputs of other nodes. We can consider (dual) GFO as a way of the (fanin) fanout re-connection of node *s*. The basic idea of GFO introduces the concept of "controlling sets" to broadcast the signal of node *s* toward Primary Outputs (POs) or Primary Inputs (PIs). It observes that if certain conditions in the controlling sets are met, wires can be simultaneously added and removed. Then, the problem of minimizing wire re-connections is transformed to that of network flow whose optimum solution can be solved by the maxflow minimal cut algorithm. For example, in Fig. 1, let us consider to re-connect the fanouts of node *s*. First the GFO derives a flow graph in Fig. 2. Since nodes $\{n_8, n_9, n_{18}\}$ form a cutset in the flow graph, GFO claims that the fanouts of node *s* can be re-connected to nodes $\{n_8, n_9, n_{18}\}$. The new circuit with fanout re-connections is shown in Fig. 3 where the number of fanouts of node *s* is reduced from 6 to 3.

In comparison to other rewiring techniques, GFO does have some speed advantage. At each iteration in GFO, the algorithm performs only one ATPG operation to derive the summary information of the controlling sets, and then uses a flow graph to determine the addition and removal of wires simultaneously. On the other hand, other rewiring techniques require to preform redundancy checks for wires which are to be added or removed. In addition, the GFO's flow graph model can provide a better-structured view for simultaneous re-connections of multiple wires than other rewiring techniques.

However, despite the above GFO's advantages, its performance is inferior to both the traditional algebraic types of optimization and most ATPG synthesis techniques. First, we observe that the original GFO may lose optimality on several obvious cases. In fact, even in their demonstrated example in [1] (also in Fig. 1), there exists a better solution than the one in the paper. The optimality is lost because during the transformation to a flow graph, some optimization freedom is not well characterized. To preserve the optimality, we transform the optimization problem to a new graph called the implication flow graph. Compared to the original flow graph, the implication flow graph contains a new type of node whose fanins can be selected from one of several candidates. We further show that finding an optimal fanout/fanin re-connection in the implication flow graph is NP complete.

In practice, we find that the fanin re-connection is often more effective than the fanout re-connection. However, the dual GFO technique [4] only considers to re-connect the fanins of node *s* from nodes inside the input cone of node *s*. The restriction leads to the loss of optimality because there are usually more opportunities to re-connect from nodes outside the input cone of node *s* than inside the cone. In this paper, we show the fanin re-connection requires much more sophisticated modeling than the one shown in [4]. By carefully modeling the fanin re-connection and solving the problem in the implication flow graph, we obtain much larger solution space than the original GFO algorithm. Our experimental results are very exciting.

2 Global Flow Revisit

In this section, we briefly explain the fanout GFO technique. The fanin GFO is similar to the fanout GFO. Before we discuss the fanout GFO technique, we would like to have some definitions. The *mandatory assignments* (MA) are the value assignments to nodes required for a test to exist and must be satisfied by any test vector. The process of computing these mandatory assignments and checking their consistency is referred to as *implication.*

Without loss of generality, let us assume that a circuit consists of only NOR gates and consider to re-connect the immediate fanouts of node *s* to the inputs of other nodes. The algorithm can be divided into three phases. In the first phase, GFO gathers the information of the "forcing sets" for node s. The forcing set F_{10} (F_{11}) is the set of nodes with MA of 0 (1) after setting and implying $s=1$. For example, in Fig. 1, if node s is set to 1, we have the forcing sets $F_{10}=\{n_1, n_2, n_3, n_4, n_8, n_9, n_{12}, n_{13}, n_{18}\}$ with forcing value 0 (MA of 0) and $F_{11}=\{s, n_5, n_6, n_7, n_{10}, n_{11}, n_{19}\}$ with forcing value 1 (MA of 1). It can be easily shown that any new connection from s to a node in F_{10} is redundant. For example, wires s->n_9 and s->n_{13} are redundant.

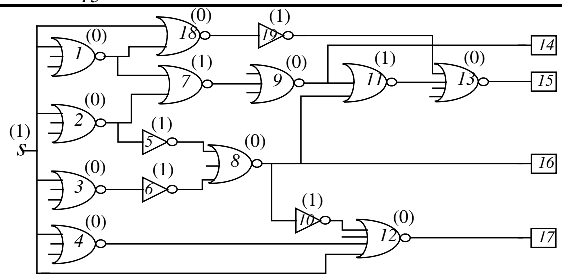

Fig. 1 Example of global flow optimization.

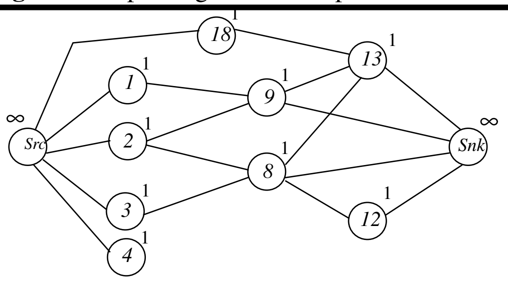

Fig. 2 Flow graph for the circuit in Fig. 1.

A node is said to be a *frontier* node if from this node to any primary output, there exists a path which does not contain (or be blocked by) a node in F_{10} or F_{11}. A frontier node is the far-most node close to PO which MA $s=1$ can be broadcasted to. In Fig. 1, nodes $\{n_8, n_9, n_{12}, n_{13}\}$ are frontier nodes. The GFO algorithm observes that any re-connection is legal as long as after re-connections, the MAs of the frontier nodes are not changed. Then, in the second phase, GFO builds a flow graph by some rules. Without going into details, we illustrate how the graph is built by the example in Fig. 1. First, in the flow graph (shown in Fig. 2), the algorithm adds a source node (*Src),* representing node s and a sink node *(Snk)*. Then, for each node n_i in $F_{10}=\{n_1, n_2, n_3, n_4, n_8, n_9, n_{12}, n_{13}, n_{18}\}$, it adds a corresponding node N_i, $\{N_1, N_2, N_3, N_4, N_8, N_9, N_{12}, N_{13}, N_{18}\}$ with weight 1 in the flow graph. Edges are then added with some rules [1]. Edges $\{N_8$->*sink*, N_9->*sink*, N_{12}->*sink*, N_{13}->*sink*$\}$ are first added from all frontier nodes to the sink node. According to the rules in [1], edges which are added to the flow graph must satisfy the following condition. If two input edges $\{I$->M, J->$M\}$ of node M are added, it must guarantee that the MA of m can be implied from the MA of nodes i and j. For example, the fanin edges N_1->N_9 and N_2->N_9 are added because MA $n_9=0$ can be implied from $n_1=0$ and $n_2=0$. Also, edge N_{18}->N_{13} is added because MA $n_{13}=0$ can be implied by $n_{18}=0$.

The flow graph by the rules in [1] has the property that if the source node is re-connected to the corresponding nodes in a cutset, the circuit functionality is unchanged. For example, in Fig. 2, nodes $\{N_8, N_9, N_{18}\}$ form a cutset of the flow graph. Therefore, node s can be re-connected to the corresponding nodes $\{n_8, n_9, n_{18}\}$ without changing the circuit functionality. Therefore, in GFO, the problem of finding the minimum number of nodes to which s fanouts, can be solved by finding a minimum cutset in the flow graph. For example, re-connect the fanouts of s to the minimum cutset $\{n_8, n_9, n_{18}\}$ can reduce the fanouts from 6 to 3, where the new circuit is shown in Fig. 3.

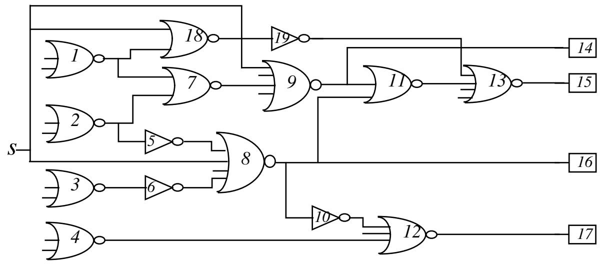

Fig. 3 Result of fanout re-connections for the circuit in Fig. 1

1. Add a *Src* node representing the node s and a sink node *Snk*.
2. Add a corresponding node for a node with implication. If a node has a controlling value, the corresponding node in IFG has an IMP_OR type with weight 1. If a node has a non-controlling value, the corresponding node has an IMP_AND type with weight infinite.
3. Edges are added as follows. First, the IMP_AND node is connected to all its corresponding input nodes. An IMP_OR node is connected to all its corresponding nodes with controlling value to the node. Add the edges from all the frontier nodes to the sink node.

3 Wire Re-connections by the Implication Flow Graph

It is not mentioned in GFO [1][4] whether the algorithm is optimal. In fact, there exists a better solution for the example in Fig. 1. In this section, we first show that GFO loses optimality in its transformation to a flow graph. Then, we propose a new graph called implication flow graph to preserve the optimality. We further show that the fanout (fanin) re-connection problem on the new graph is NP complete.

During the construction of a flow graph, the fanin edges of a node are added if the MA of the node can be implied by the MAs in its fanins. However, there are actually two types of implication. One is the "AND" condition of implication. For example, in Fig. 2, the MA of $n_9=0$ is implied by both its fanin MAs $n_1=0$ AND $n_2=0$. Another is the "OR" condition of implication. For example, the MA of $n_8=0$ can be implied by either MA $n_2=0$ OR MA $n_3=0$. The problem of using a flow graph is its inability to model the OR condition of implication. To capture the OR condition, we build a new graph called the Implication Flow Graph (IFG). The new graph is built as follows. In the IFG, we construct two types of nodes, one of which is the "*IMP_OR*" type to model the "OR" condition and the other is the "*IMP_AND*" type to model the "AND" condition.

The algorithm to construct an IFG is in Fig. 4. (The IFG for the circuit in Fig. 1 is shown in Fig. 5(a).) In the first step of Fig. 4, we add a source and a sink node. Then, in the second step, the corresponding nodes to all nodes with MAs are added to the IFG while in GFO only nodes in F_{10} are considered. If a node has a controlling value, the corresponding node in the IFG has an IMP_OR type with weight 1 such as nodes $\{N_8, N_{13}\}$ in Fig. 5(a). If a node has a non-controlling value, the corresponding node has an IMP_AND type with weight infinite such as $\{N_7, N_{11}\}$ in Fig. 5(a). Then in the third step, edges are added. An IMP_AND node is connected from all its inputs such as $\{N_1\text{->}N_7, N_2\text{->}N_7\}$ in Fig. 5(a). An IMP_OR node is connected from all its inputs with a controlling value such as $\{N_2\text{->}N_8, N_3\text{->}N_8\}$ in Fig. 5(a). Finally, edges are added from the frontier nodes to the sink node.

After building the IFG, we construct a new graph called a "*degenerated*" IFG by removing all but one fanin edges for each IMP_OR node. Note that after removal, all IMP_OR nodes in a degenerated IFG can have only one fanin. Note that the MA of an IMP_OR node can be implied by just one fanin MA. Then, similar to the fanout GFO, any cutset in a degenerated IFG can form a solution for the fanout re-connection. Note that since only IMP_OR nodes are assigned weight 1, a cutset can contain only IMP_OR nodes.

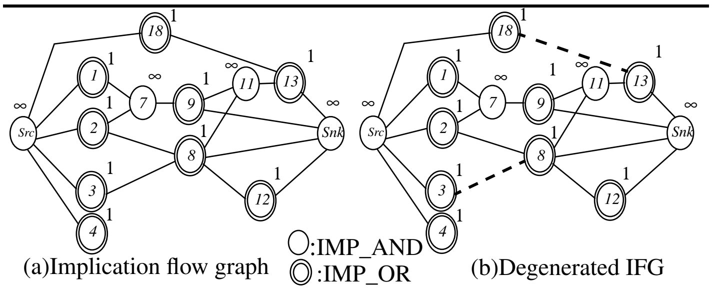

Fig. 5 Implication flow graph and degenerated IFG for Fig. 1.

Theorem 1: The problem of finding the minimum fanout re-connection in an IFG is NP complete.

For the same example in Fig. 1, since there are two IMP_OR nodes $\{N_8, N_{13}\}$ with two fanin edges so there are 4 degenerated IFGs. A degenerated IFG is shown in Fig. 5(b) by removing the dotted edges. The degenerated IFG has a cutset $\{N_8, N_9\}$. Therefore, we can re-connect the fanouts of s to nodes $\{n_8, n_9\}$, resulting a better solution than [1]. Later, we discuss our heuristic to obtain a "good" degenerated IFG having a small cutset.

4 Fanin re-connections by fanin IFG

The objective of the fanin re-connection is to re-connect the fanins of a node from outputs of other nodes. The dual GFO algorithm only considers those nodes in the fanin cone for re-connections. However, in many cases, the fanin re-connection can only be achieved from nodes outside the fanin cone. For example, consider the circuit in Fig. 6. There is no fanin re-connection of node s from nodes inside its fanin cone but we can re-connect node s from nodes $\{n_{11}, n_{12}\}$ outside the fanin cone, and remove the original fanin wires $\{n_8\text{->}s, n_9\text{->}s, n_{10}\text{->}s\}$. In addition of restricting nodes in the fanin cone, the problem of observability is not considered in the dual GFO.

We now re-formulate the problem of the fanin re-connection as follows. Let a node s be an AND gate under consideration. The procedure consists of three steps. First, we obtain the observability MAs *ObvMa(s)* for node s, which are the assignments must be satisfied for node s to be observable at POs. For any fanin re-connection, the MA set, *ObvMa*(s) should be preserved. For example, we have *ObvMa*(s) = $\{n_7=1\}$ in Fig. 6. Then, we assign and propagate MA $s=1$. Let the additional MAs derived by setting $s=1$ under the observability condition be *ActMa(s=1)*. In the example, we have *ActMa*($s=1$)= $\{s=1, n_8=1, n_9=1, n_{10}=1, n_1=1, n_2=1, n_3=1, n_4=1, n_5=1, n_6=1, n_{11}=1, n_{12}=0, n_{13}=1\}$. Similar to the definition of a frontier node in the fanout GFO, a node in *ActMa*($s=1$) is defined to be a *fanin frontier* node if from any PI to this node, there exists a path which does not contain (or be blocked by) a node with an MA. The fanin frontier nodes are the far-most nodes (close to PI) where MA $s=1$ can be broadcasted. Moreover, any fanin re-connection must guarantee that MAs of the fanin frontiers are the same. In the example, fanin frontiers are $\{n_1, n_2, n_3, n_4, n_5, n_6\}$.

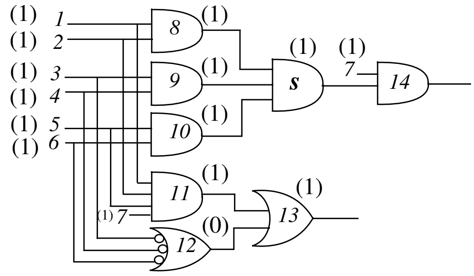

Fig. 6 Example of fanin GFO.

In the second step, we build a fanin IFG from the summary information of *ObvMa(s)* and *ActMa(s=1)*. The procedure is in Fig. 7. Because it is possible to have MAs propagating forward and backward, the ways of adding edges, assigning weights and the types (either IMP_AND or IMP_OR) are different from the fanout IFG. We have enumerated all possible cases of implication and derived a set of rules in TABLE 1. Suppose an AND node n_i has an MA of *val*. The purpose of a rule is to describe how other MAs can be implied from $n_i=val$. Consider Rule 1. If MA $n_i=1$ is a non-controlling value, MA $n_i=1$ can imply all its fanin MAs. Edges are added from N_i to its inputs such as edges $\{N_{12}\text{->}N_3, N_{12}\text{->}N_4, N_{12}\text{->}N_6\}$ in Fig. 8(a). Consider another rule Rule 3, which has one controlling fanin n_k and one (the other) non-controlling fanin n_j. The MA of $n_k=0$ can be implied from both $n_j=1$ and $n_i=0$ so edges are added from $N_i\text{->}N_k$ and $N_j\text{->}N_k$. For example in Fig. 6, node n_{13} satisfies the condition of Rule 3 so edges $\{N_{13}\text{->}N_{11}, N_{12}\text{->}N_{11}\}$ in Fig. 8(a) are added. We then connect all the frontier nodes to the sink node *Snk* and also the source node *Src* to those nodes which don't have any input. For example, in Fig. 8(a), the edges are added from the source node to nodes $\{N_8, N_9, N_{10}, N_{12}, N_{13}\}$. Finally, the type of a node is assigned by the following. If the MA of a node can be implied by several of its fanouts/fanins, it is assigned the IMP_OR type. This is also characterized by Rule 4. For example in Fig. 6, node $n_1=1$ can be implied by $n_8=1$ and $n_{11}=1$, so node N_1 in Fig. 8(a) is assigned the IMP_OR type. All others are assigned the IMP_AND type. The IFG for the circuit in Fig. 6 is shown in Fig. 8(a).

1. Add a *Src* node representing node *s* and a sink node *Snk*.
2. Add a corresponding node for a node in *ActMa(s=1)*.
3. The edges, the type, and the weight assignment of a node is shown in TABLE 1. Edges are added from a frontier node to *Snk* and from *Src* to all nodes without inputs.
4. A node is assigned the IMP_OR type if the MA can be implied by several others. Otherwise, it is assigned IMP_AND.

Fig. 7 Procedure to build a fanin Implication Flow Graph.

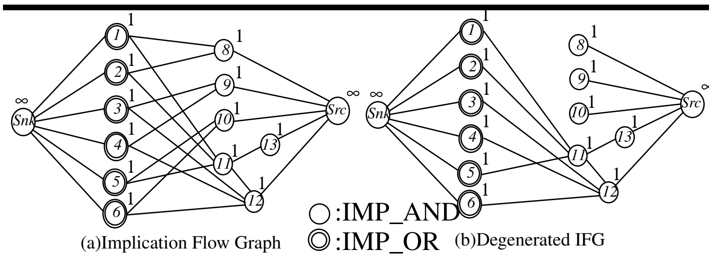

Fig. 8 Implication Flow Graph and Degenerated IFG for the circuit in Fig. 6.

Because there are 6 IMP_OR nodes with 2 input edges in the fanin IFG of Fig. 8(a), we can derive 2^6 degenerated IFGs. One degenerated IFG is shown in Fig. 8(b), which has a cutset $\{N_{12}, N_{13}\}$. Therefore, one can re-connect the fanins of node *s* from nodes $\{n_{12}, n_{13}\}$ without changing the functionality.

5 A heuristic to obtain an efficient degenerated IFG

Obviously, the size of a cutset depends on how a degenerated IFG is obtained. In order to obtain a "good" degenerated IFG with a small size cutset, we need to select one input edge and remove all others for each IMP_OR node. Our basic idea comes from the following observation. Let an immediate fanout of the source node *Src* be *SrcFanout*. Note that the set of all *SrcFanouts* can form a cutset. If after edge removal, all frontier nodes "depend on" (are in the transitive fanouts of) as few as SrcFanouts, a degenerated IFG can have a small cutset. In Fig. 8(a). if we remove all the edges from $\{N_8\text{->}N_1, N_8\text{->}N_2, N_9\text{->}N_3, N_9\text{->}N_4, N_{10}\text{->}N_5, N_{10}\text{->}N_6\}$, the frontier nodes $\{N_1, N_2, N_3, N_4, N_5, N_6\}$ depend on $\{N_{12}, N_{13}\}$. Our heuristic first assigns a cost for each SrcFanout, the value of which calculates the number of frontier nodes which are in its transitive fanouts. Then, an input edge of an IMP_OR node is selected if the corresponding input node has large cost values of SrcFanouts in its transitive fanins. In this way, we can obtain a good degenerated IFG. We also use some heuristic to break cycles in a fanin IFG. Once a good degenerated IFG is obtained, we can then perform the fanin re-connection by finding a cutset in the degenerated IFG.

6 Experimental results

We have implemented the fanin/fanout GFO algorithms based on the fanout/fanin IFGs and preformed experiments on ISCAS and MCNC benchmark circuits. Each circuit is first optimized by the SIS (algebraic) script, and then decomposed into AND/OR gates. Then, we perform further area minimization by GFO [1] and our IFG based algorithm. The results are shown in TABLE 2. For example, circuit alu4 has 1239 literals after SIS optimization. After GFO, the circuit has 1231 literals while after our optimization, the circuit has 1160 literals. On the average, we obtain around 8% improvement in comparison to traditional GFO. It can be clearly found that our IFG based algorithm can explore much more optimization power than GFO.

7 Conclusion

In this paper, we have formulated the fanin/fanout re-connection problem by a graph called implication flow graph. With reasonable running time, significant improvements on area optimization can be achieved by our proposed methods.

8 References

[1] C. L. Berman and L. H. Trevillyan. "Global Flow Optimization in Automatic Logic Design," IEEE Trans. CAD 10, pp. 557-564, May 1991.

[2] S. C. Chang, M. Marek-Sadowska, and K. T. Cheng, "Perturb and Simplify: Multi-Level Boolean Network Optimizer", *IEEE Transaction on Computer Aided Design*, Vol. 15, pp. 1494-1504, Nov 1996.

[3] K. T. Cheng and L. A. Entrena, "Multi-Level Logic Optimization by Redundancy Addition and Removal," in *Proc. European Conference On Design Automation*, pp.373-377, Feb. 1993.

[4] R. Damiano and L. Berman, "Dual Global Flow", Proc. IEEE Int. Conf. on Computer Design, pp. 49-53, Oct. 1991.

[5] U. Glaser and K.T. Cheng, "Logic Optimization by an Improved Sequential Redundancy Addition and Removel", in Proc. of ASP-DAC, pp.235-240, Sept. 1995.

[6] W. Kunz and D. K. Pradhan, "Multi-Level Logic Optimization by Implication Analysis", *Digest Int. Conf. on Computer Aided Design*, pp. 6-13, Nov. 1994.

TABLE 1:Rules for adding edges and assigning weight.

R. #	The condition of a rule	wt.	Edge addition rules
1	n_j MA=1, n_k MA=1, n_i MA=1	1	N_i (1) → N_j (1), N_i (1) → N_k (1)
2	MA=0, n_j, n_k, n_i MA=0	∞	N_i (0) → N_j (0)
3	MA=1 n_j, n_k MA=0, n_i MA=0	1	N_j (1) → N_k (0), N_i (0) → N_k (0)
4	n_i MA=0(1), n_j MA=0(1), n_k MA=0(1)	1	N_j (0(1)) → N_i (0(1)), N_k (0(1)) → N_i (0(1))

TABLE 2:Area comparison between GFO and ours.

circuit	SIS lits(SOP)	GFO lits(SOP)	ours (IFG) lits(SOP)	time (sec)
alu4	1239	1231	1160	9.30u
C432	316	313	227	0.32u
C6288	4296	4265	3831	14.15u
cps	1468	1468	1395	18.68u
dalu	2041	2017	1914	12.61u
example2	482	478	443	1.10u
i8	1585	1570	1448	12.6u
mult32a	694	694	570	1.08u
sbc	1164	1148	1081	2.34u
t481	1105	1105	969	7.52u
term1	352	351	320	0.34u
ttt2	305	305	290	0.30u
x1	432	430	407	0.34u
Total 41		1	0.92	