Implication Graph based Domino Logic Synthesis*

Ki-Wook Kim† C. L. Liu‡ and Sung-Mo Kang†

Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign
Department of Computer Science, National Tsing Hua University

Abstract

In this paper, we present a new approach to the problem of inverter elimination in domino logic synthesis. A small piece of static CMOS logic is introduced to the circuit to avoid significant area penalty resulting from duplication. To maximize the domino logic part and to minimize the static CMOS logic part, a generalized ATPG based logic transformation is proposed to eliminate or relocate a target inverter. Based on the new concept of *dominating set of mandatory assignment (DSMA)* and the corresponding implication graph, we propose algorithms to identify a minimum candidate set for a target inverter. Experimental results show that logic transformation based on implication graph can reduce transistor counts by 25% and power delay product by 25% on average.

1 Introduction

Dynamic logics such as domino logic [8] can be a substitute for static CMOS logic in high performance control logic as well as in data path design. Such substitutions could lead to reduction in delay as well as silicon area. Despite the benefits provided by domino logic circuits, the monotonic property of domino logic imposes substantial limitations on the synthesis of an inverting logic. The monotonic property originates from the intrinsic operation of the domino logic which allows at most one low-to-high transition at the output buffer in the evaluation phase.

To overcome such limitations, attempts have been made to minimize inverters by the method of output phase assignment [5]. However, some inverters cannot be swept away this way, because they are trapped in a reconvergent loop as illustrated in Figure 1. Most of the conventional techniques to eliminate trapped inverters are based on duplication of subcircuits. There have been efforts to minimize the area of duplicated logic [12], or the power consumption [9] by way of output phase assignment. Satisfiability don't-care optimization was used in [11]. There have been approaches to implementing an inverting logic in circuit level, such as dual rail logic [4], clock-delayed domino logic [14], and clock-and-data precharged dynamic circuit [10].

In essence, duplication-based inverter elimination approaches are extensions of the bubble pushing algorithm. Every transitive fanin gate of a trapped inverter is duplicated in the network, and then inverters are pushed towards primary inputs by applying DeMorgan's theorem.

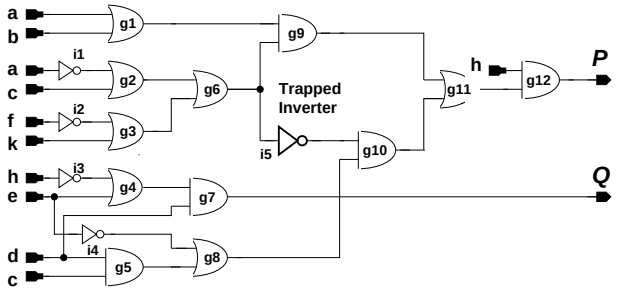

Figure 1: A combinational circuit with a trapped inverter.

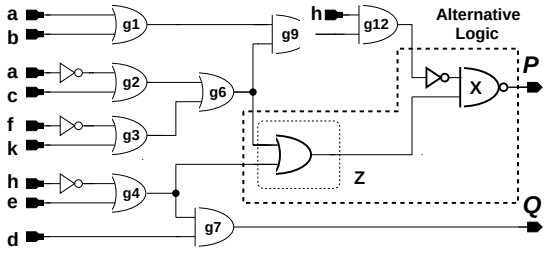

Figure 2: Logic transformation-based inverter elimination or relocation.

Area penalty is obviously significant in duplication-based inverter elimination. Along with duplicated transistors, considerable amount of interconnecting wires is required. Thus, the routing and placement problems become more complex, which in turn result in performance degradation.

To overcome such significant penalties caused by duplication, we allow some inverters to be included in a part of the logic network. The monotonic property of domino logic prohibits the implementation of the transitive fanout subcircuit of an inverter in traditional domino logic. Because static CMOS logic allows both low-to-high and high-to-low transitions, the transitive fanout subcircuit of an inverter can be synthesized with static CMOS logic. In our approach, the circuit is composed of a dominant *domino logic portion* followed by a small *static CMOS logic portion.* Small piece of static CMOS logic in the circuit is to avoid significant area penalty due to duplication.

Figure 2 shows the resynthesized network when our proposed transformation technique is applied to the circuit in Figure 1. All the gates are implemented with domino logic except gate X which will be implemented with static CMOS logic. Clearly, the number of transistors is significantly reduced compared to duplication-based inverter removal.

Our ultimate goal is to maximize the advantages of domino logic and at the same time to minimize the size of the compensating static CMOS logic. Logic transformation enables us either to eliminate

*Supported in part by National Science Foundation (USA) under grant NSF MIP 96 12184 and by National Science Council (ROC) under grant 88-2215-E-007-026.

†Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, IL 61801

‡Department of Computer Science, National Tsing Hua University, Hsin-Chu, Taiwan 30043

0-7803-5832-X /99/$10.00 ©1999 IEEE.

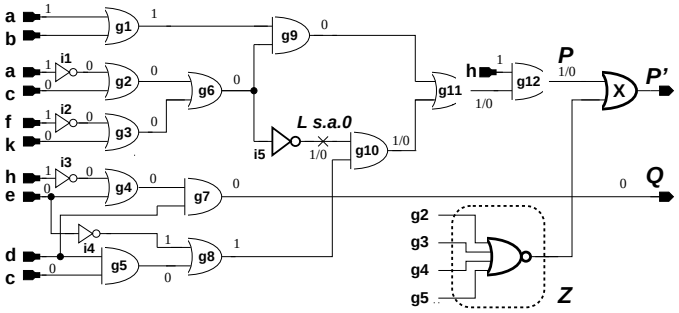

Figure 3: A set of mandatory assignments to generate a single stuck-at-0 fault at the signal L, and to propagate the fault to a signal P. The signal L can be removed by adding a redundant signal Z and a gate X.

the inverters, or to relocate the inverters as close to the primary outputs as possible so that the CMOS subcircuit is kept small.

For elimination or relocation of inverters, we propose a generalized automatic test pattern generation (ATPG) based logic resynthesis technique which will be presented in section 2. Experimental results and conclusion follow in section 3 and section 4.

2 Inverter Elimination using ATPG-based Logic Transformation

2.1 Problem Formulation

Without loss of generality, we assume that a logic network is a combinational circuit consisting of 2-input AND gates, 2-input OR gates, and inverters. The complementary signal of each primary input is assumed to be available.

Logic transformation enables us to restructure a circuit such that an inverter output signal can be replaced by an alternative signal. ATPG techniques can be used to identify redundant signals as candidates for substitution by performing fault excitation and propagation [1, 2].

The transformation is carried out as follows. For a *target* signal L, generate (or identify) an alternative signal Z (called a *candidate*) from a set of existing signals (called a *candidate set*), such that L and Z are reciprocally redundant with respect to the *observing* signal P. In other words, L is redundant because of Z, and Z is redundant because of L. We introduce a new gate X with the observing signal P and the alternative signal Z as inputs, then remove the target signal L and the corresponding gates.

It is important to note that the alternative signal Z is not unique. For the circuit in Figure 1, $Z = (\overline{g4} \cdot \overline{g6})$ is a candidate for L, resulting in the circuit in Figure 2. Meanwhile, as shown in Figure 3, another implementation of $Z = (\overline{g2} \cdot \overline{g3} \cdot \overline{g4} \cdot \overline{g5})$ can also replace L without modifying the functionality of the circuit in Figure 1.

Most of the conventional ATPG-based logic transformation approaches have focused on single line rewiring. In [1], identification of multiple signals as a candidate set is carried out.

We introduce the notion of a minimum candidate set, and propose systematic approaches to identifying minimum candidate sets in the following sections. Our new approaches generalize traditional ATPG-based logic transformation.

2.2 Definitions and Notations

Throughout this paper, L denotes the output signal of the target inverter to be removed. P denotes a signal where the *stuck-at-v* (briefly, *s-a-v*) fault at L will be observed as illustrated in Figure 3. Without loss of generality, we assume that there is no inverting logic in the path between L and P. X denotes an additional gate the controlling value of which is complementary to the fault value, namely $\overline{v}$. Z denotes an alternative signal of L, and Z_{IN} denotes all signals used to generate Z. The inputs to the gate X are P and Z, and the output signal of the gate X is denoted P' which represents the same function as P after removing the target signal L.

A *set of mandatory assignments* $SMA(L, P, v)$ of signal L is a set of ordered pairs $< x, V(x) >$, where x is a signal and $V(x)$ is an assignment of values to x. $SMA(L, P, v)$ is a necessary and sufficient condition for the generation of a single *s-a-v* fault at signal L, and to propagate the fault to the signal P through a path from L. For convenience, we use the representation $x \in SMA(L, P, v)$ if an ordered pair $< x, V(x) >$ is in $SMA(L, P, v)$.

A *dominating set of mandatory assignments* $DSMA(L, P, v)$ of signal L is a subset of $SMA(L, P, v)$ such that $DSMA(L, P, v)$ sets P to $\overline{v}$.

Let us define the notions of redundancy and candidate set formally.

Signal L is said to be *redundant* with respect to signal P if there is no consistent $SMA(L, P, v)$. Signal Z is a *candidate* for signal L if $< Z, \overline{v} > \in SMA(L, P, v)$, and $< P, \overline{v} > \in SMA(Z, Z, v)$. The set of signals for generating a candidate Z, i.e. Z_{IN}, is a *candidate set* for L.

$SMA(L, P, v)$ makes candidate Z have $\overline{v}$ by definition. $< Z, \overline{v} >$ keeps the *s-a-v* fault at L from propagating to P' because $\overline{v}$ is the controlling value of X. This means that there is no consistent test for a *s-a-v* fault at L. Therefore, L becomes redundant because of Z. Meanwhile, in order to assure the functional equivalence after replacing L with Z, Z must be redundant because of L. For this purpose, when testing Z under the $SMA(Z, Z, v)$ condition, we assign to P the controlling value of X, which keeps the *s-a-v* fault at Z from propagating to P'. In other words, Z is forced to be redundant. Therefore, if Z is a candidate for L, L and Z are reciprocally redundant.

2.3 Minimum Candidate Set

Now we are ready to describe the property of a minimum candidate set. The proof of the following lemma and theorems are shown in [7].

Lemma 2.1 *Let Z_{IN} be a minimum candidate set for L. If Z_{IN} exists, then Z_{IN} does not contain any variable $x \notin SMA(L, P, v)$.*

Lemma 2.1 connotes that a minimum candidate set must be chosen from $SMA(L, P, v)$. In other words, the search space for a minimum candidate set is restricted to SMA.

Lemma 2.2 *A signal L can be removed from the network by introducing a gate X with controlling value $\overline{v}$, and a signal Z such that $DSMA(L, P, v) \subseteq SMA(Z, Z, v)$, and all tests compatible with $DSMA(L, P, v)$ imply that Z is equal to $\overline{v}$.*

Intuitively, Lemma 2.2 means that being a member of $DSMA$ is a necessary condition to be a candidate in a candidate set. Based on Lemma 2.1 and Lemma 2.2, we prove the following main theorem.

Theorem 2.1 *A minimum candidate set for L is a minimum $DSMA(L, P, v)$.*

Thus the minimum candidate set problem is translated into that of identifying a minimum $DSMA(L, P, v)$ from $SMA(L, P, v)$. In other words, the goal is now to choose a minimum set of variables and associated values which assert that the output P will have the controlling value of X. We propose two algorithms based on the notion of implication graph defined in the following section.

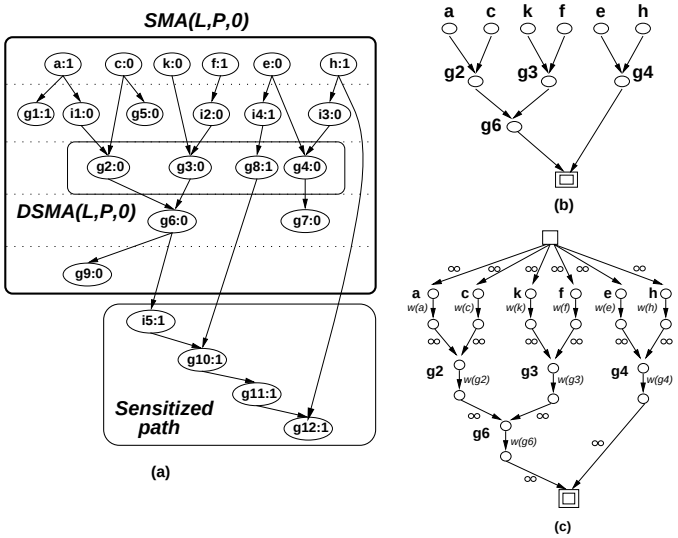

Figure 4: (a) An implication graph corresponding to the $SMA(L,P,0)$; (b) simplified implication graph with bridge nodes taken into account; (c) modified implication graph for min-cut algorithm.

2.4 Minimum Candidate Set Algorithms based on *SMA* Implication Graph

To determine a minimum $DSMA$, we use (1) the number of transistors for generating an alternative signal Z and (2) signal propagation delay as figures of merit in choosing variables from the SMA. In other words, we choose the candidate set with minimum cardinality first. If there is a tie, an early signal is preferred.

2.4.1 *SMA* Implication Graph

A SMA implication graph (in brief, implication graph) $G = (V, E)$ is a directed acyclic graph consisting of a node set V and an edge set E. The implication graph represents an SMA and the implication relationships among the ordered pairs. It is constructed as follows. Each ordered pair in the SMA corresponds to a node in the implication graph. If a gate v_i is connected to a gate v_j, and the assigned value of v_i in SMA implies the assigned value of v_j (denoted $v_i \Rightarrow v_j$), then there is a directed edge (v_i, v_j) in the graph. All the gates in the sensitized path between the target signal and the observing signal are collapsed into one sink node. Then, any component connected to the sink node constitutes an implication graph.

By definition, a $DSMA$ is represented in an implication graph as follows. Let V_D be a set of nodes corresponding to a $DSMA$. Then, there exists at least one path including a subset of V_D between each source node and the sink node.

To reduce computational complexity, we simplify the implication graph in the following way:

Node Collapsing If a node v_t has a single incident edge (v_f, v_t) from a node v_f, then the node v_t can be eliminated, and all the edges emanating from v_t will be adjusted to emanate from the node v_f. This simplification is based on the following theorem:

Theorem 2.2 *Let A be a subset of the dominating set of mandatory assignments, i.e. $A \subseteq DSMA(L, P, v)$. If all ordered pairs in A are implied by a set of ordered pairs $B \subseteq SMA(L, P, v)$, then $(DSMA(L, P, v) - A) \cup B$ is also a $DSMA(L, P, v)$.*

Node collapsing helps to improve the signal propagation delay while keeping the transistor counts unchanged.

Redundant Edge Elimination Redundant edges that represent transitive implication can be removed from the implication graph based on the following theorem:

Theorem 2.3 *Suppose the assigned variables A and B imply C and C implies A and B $((A \wedge B) \Leftrightarrow C)$. If A and C imply D $((A \wedge C) \Rightarrow D)$, then C implies D $(C \Rightarrow D)$.*

2.4.2 Optimal Algorithm for Minimum Candidate Set

In general, the minimum $DSMA$ problem is transformed into the problem of identifying a minimum set of nodes which covers all source nodes. The set covering problem is known to be NP-hard [3].

In the special case where the implication graph is a tree with a node set V and an edge set E, the optimal solution can be obtained by the min-cut algorithm. The algorithm computes the cut of minimum weight with time complexity $O(\mid V \mid \cdot \mid E \mid + \mid V \mid^2 \cdot log \mid V \mid)$ [13].

For this purpose, we exploit a node-splitting technique, which is used to find a min-cut for nodes rather than edges. Each node is divided into two nodes that are connected by an edge which will be weighted by a finite value. Other edges are assigned to infinity. Then the min-cut in the transformed graph gives an optimal solution.

2.4.3 Heuristic Algorithm for Minimum Candidate Set

The general problem of finding a minimum candidate set becomes more complex when the graph contains nodes with multiple outgoing edges representing multiple fanouts in the circuit as well as nodes which have no path to the sink node. (These nodes are called *bridge nodes*).

To apply the min-cut algorithm, we use the following heuristics to change the graph into a tree.

Multiple Outgoing Edges In some cases, a node with multiple outgoing edges precludes the possibility of obtaining a better candidate set. To remove such limitation, we use the following heuristics. First, we remove all outgoing edges except one incident with a node at the lowest level. This heuristic is used to put a preference on the earliest fanout node. Secondly, if the sink nodes of all outgoing edges are located at the same level, choose the edge incident with the node with the largest number of incident edges. (It corresponds to the greedy approximation for the set covering problem.)

Bridge Nodes Bridge nodes provide opportunities for obtaining better candidate sets. To take advantage of bridge nodes, we use the following heuristic. First, all connected bridge nodes are collapsed into one supernode v_b. An edge (v_f, v_t) will be replaced by an edge (v_b, v_t), if there is at least one outgoing edge from v_f to one of the bridge nodes.

Example 2.1*:* Simplifying the implication graph in Figure 4(a) and applying heuristics for a bridge node ($g4$) lead to the implication graph in Figure 4(b). In order to apply the min-cut algorithm, the implication graph in Figure 4(b) is modified to become that in Figure 4(c) by node-splitting. The weighting function $w(\cdot)$ will be properly specified according to the objective of logic transformation such as switching activity for low power. The min-cut algorithm produces the minimum candidate set $\{g4, g6\}$ for unity weight function for the implication graph in Figure 4(c). □

3 Experimental Results

The described methods have been implemented. First, each circuit in the benchmark circuit set was optimized using SIS, and inverters are minimized based on the technique proposed in [5]. Then

Table 1: Results for ISCAS-85 circuits

Circuit	Inv. count	Duplication			IG Optimal Algorithm[1]					IG Heuristic Algorithm[2]				
		Area	Delay	Power	Area	Delay	Power	CPU[3]	CMOS[4]	Area	Delay	Power	CPU	CMOS
C432	9	1150	82.7	22.6	741	77.9	14.5	9.0	0.07	766	78.0	14.5	1.2	0.07
C880	36	1760	78.6	38.2	1423	86.4	26.9	34.8	0.12	1457	89.4	27.1	3.8	0.13
C1355	112	2564	45.2	48.1	2072	44.5	40.7	90.2	0.11	2090	45.1	39.5	7.1	0.11
C1908	107	2471	72.9	51.3	2029	80.3	49.7	117.0	0.10	2316	80.9	51.3	6.8	0.12
C2670	94	3707	59.6	87.4	2780	61.6	60.0	157.4	0.07	2991	62.3	63.8	10.5	0.07
C3540	139	7063	100.4	156.3	5061	100.8	106.9	275.9	0.03	5211	101.1	105.3	18.7	0.05
C5315	250	7389	72.9	208.4	5221	76.8	122.5	704.5	0.18	5303	77.9	134.0	29.3	0.24
C6288	887	15155	232.1	546.6	12275	240.8	415.6	4017.6	0.24	12551	242.6	401.2	76.2	0.26
C7552	402	10192	182.7	311.8	7542	195.0	228.0	1591.7	0.10	7722	203.7	231.6	40.8	0.13
Ratio	·	1.0	1.0	1.0	**0.75**	**1.03**	**0.73**	·	**0.10**	**0.78**	**1.05**	**0.73**	·	**0.13**

[1] Implication graph based optimal algorithm

[2] Implication graph based heuristic algorithm

[3] Computation time in sec

[4] The ratio of the transistor counts for static CMOS logic part to the transistor counts for the whole circuit

trapped inverters are eliminated using duplication and our algorithms. SMA of the target signal is obtained by using the technique in [15]. Area and delay are measured using SIS after technology mapping. Power consumed by the circuit is measured using FIT power estimation tool [6], whose accuracy of the power estimation is within 5% from HSPICE.

Table 1 compares the results of inverter elimination by duplication, our implication graph (IG) based optimal algorithm and heuristic algorithm for the ISCAS-85 circuits. In comparison with the duplication approach, the IG optimal algorithm reduced the area by 25%, while the delay increases by about 3% on the average. We note that the power reduction is significant. The circuits consume 27% less power. Comparing the performance of the IG heuristic algorithm with that of the IG optimal algorithm, we note that computations are carried out within reasonable time without significant degradation of area, delay, and power.

The column denoted "CMOS" corresponds to the number of transistors in the static CMOS logic divided by the number of transistors in the whole circuit. On the average, 10% of the transistors used in the static CMOS logic. We note that a larger number of trapped inverters causes increases in the size of the static CMOS logic as in the circuit C6288.

4 Conclusions

This work was motivated by the observation that duplication-based inverter elimination causes expensive area and power penalty. To overcome such penalties, we proposed a hybrid circuit containing both domino logic and compensating static CMOS logic.

To minimize the CMOS logic and to maximize the benefits of domino logic, we proposed a general ATPG-based approaches for the elimination of inverters. Our new algorithms enable us to map the set of mandatory assignments into the implication graph, and to identify minimum candidate sets for inverter replacement from the implication graph.

Our experimental results show that area is reduced by 25% on the average. The transformed circuits also show a 25% reduction in power delay product compared with the duplicated circuits.

Note that our approach to identifying minimum candidate set is also useful for logic resynthesis. Equipped with the proper weighting function, such as switching activity, this technique can determine an optimal solution for low power in a reasonable amount of computational time.

Acknowledgments

We are indebted to Prof. Janak H. Patel and Dr. Ilker Hamzaoglu for providing the automatic test pattern generation tool. Also we express our gratitude to Ki-Seok Chung in Synopsys for his constructive comments.

References

[1] S. C. Chang, M. Marek-Sadowska, and K. T. Cheng. Perturb and simplify: multilevel boolean network optimizer. *IEEE Trans. Computer-Aided Design*, 15(12):1494–1504, November 1996.

[2] L. A. Entrena and K. T. Cheng. Sequential logic optimization by redundancy addition and removal. In *Proc. IEEE/ACM Int. Conf. Computer Aided Design*, pages 310–315, 1993.

[3] M. R. Garey and D. S. Johnson. *Computers and Intractability : A Guide to the Theory of NP-Completeness.* Freeman, 1979.

[4] L. G. Heller, W. R. Griffin, J. W. Davis, and N. G. Thoma. Cascode voltage switch logic : A differential CMOS logic family. In *Proc. IEEE Int. Solid-State Circuits Conf.*, pages 16–17, 1984.

[5] A. Jain and R. K. Bryant. Inverter minimization in multi-level logic networks. In *Proc. IEEE/ACM Int. Conf. Computer Aided Design*, pages 462–465, 1993.

[6] K. W. Kim, Ting Ting Hwang, C. L. Liu, and S. M. Kang. Logic transformation for low power synthesis. In *Proc. Design, Automation and Test In Europe*, pages 158–162, 1999.

[7] K. W. Kim, C. L. Liu, and S. M. Kang. Implication graph based domino logic synthesis. Technical Report UILU-ENG-99-2206 (DAC 72), University of Illinois at Urbana-Champaign, 1999.

[8] R. H. Krambeck, C. M. Lee, and H. S. Law. High speed compact circuits with CMOS. *IEEE J. Solid State Circuits*, SC-17(3):614–619, June 1982.

[9] P. Patra and Unni Narayanan. Automated phase assignment for the synthesis of low power domino circuits. In *Proc. ACM/IEEE Design Automation Conf.*, pages 379–384, 1999.

[10] Johnny Pihl and Einar J. Aas. Logic synthesis with the CDPD circuit technique. In *Proc. Int. Symp. Circuits and Systems*, May 1996.

[11] M. Prasad, D. Kirkpatrick, R. Brayton, and A. Sangiovanni-Vincentelli. Domino logic synthesis and technology mapping. In *Proc. Int. Workshop Logic Synthesis*, May 1997.

[12] R. Puri, A. Bjorksten, and T. E. Rosser. Logic optimization by output phase assignment in dynamic logic synthesis. In *Proc. IEEE/ACM Int. Conf. Computer Aided Design*, pages 2–8, 1996.

[13] M. Stoer and F. Wagner. A simple min cut algorithm. *Algorithms - ESA'94, LNCS 855*, pages 141–147, 1994.

[14] Gin Yee and Carl Sechen. Clock-delayed domino for adder and combinational logic design. In *Proc. IEEE/ACM Int. Conf. Computer Design*, pages 332–337, October 1996.

[15] J. K. Zhao, E. M. Rudnick, and J. H. Patel. Static logic implication with application to redundancy identification. In *Proc. IEEE VLSI Test Symp.*, April 1997.