Dual-monotonic domino gate mapping and optimal output phase assignment of domino logic *

Min Zhao Sachin S. Sapatnekar
Department of Electrical and Computer Engineering
University of Minnesota, 200 Union Street SE, Minneapolis MN 55455, USA.
contact: sachin@ece.umn.edu

Abstract

In this paper, two problems on domino logic synthesis are addressed. A mapping method that maps the complementary logic cones independently when AND/OR logic is to be implemented, and together using dual-monotonic gates in the case of XOR/XNOR logic, is proposed. The results show up to 28.9% improvement in area and always show the same or better performance in delay over existing approaches. Then, a 0-1 integer programming formulation is provided for the output phase assignment problem for domino logic. It considers the cost difference between two polarities and enables a standard linear programming package to be used to solve the problem. The results show up to 41.0% improvement in area.

1 Introduction

Domino logic is one of the most effective circuit configurations for implementing high speed logic designs. A major problem in domino logic synthesis is related to its non-inverting property. This feature necessitates logic duplication of the nodes of an input network for which both polarities of a logic signal are required, as illustrated in Figure 1.

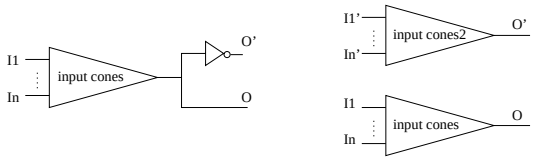

Figure 1: Logic duplication in domino logic synthesis

The manner in which the logic duplication problem is handled is the main factor that influences the quality of the domino logic synthesizer. In this paper, two issues related to logic duplication in domino logic are addressed. The first is the issue of mapping the duplicated logic using dual-monotonic gate mapping and the second is the problem of reducing the duplication cost, for which a 0-1 programming formulation of the output phase assignment problem is presented.

2 Dual-monotonic gate mapping

2.1 Dual-monotonic logic

A dual-monotonic gate is a merged gate that generates both negative and positive polarities of a logic signal [1]. A typical dual-monotonic two-input XOR gate is shown as Figure 2. The logic used to implement $a \cdot b$ and $\overline{a} \cdot b$ share the same transistor, D. Due to the complementary relations between a and $\overline{a}$, a sneak path between $O1$ and $O2$ can be prevented. While dual-monotonic implementations of XOR logic use a smaller number of transistors than duplicated gates, this is not true for all logic functions. Indeed, implementations of most common dual-monotonic gates do not share as many transistors as the XOR gate.

*This work was supported by the Semiconductor Research Corporation under contract 99-TJ-692

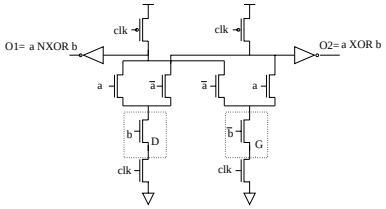

Figure 2: An example of a dual-monotonic gate

2.2 Previous work and motivation

A domino gate implementation of an input network often requires the synthesis of both positive and negative signals at many nodes due to the unateness requirements; this is known as dual-rail logic. A logic function and its complement can be built as two separate gates or as one merged gate (a dual-monotonic gate).

Recently, several papers on domino logic synthesis have appeared [2–4]. All of them follow the synthesis methodology proposed in [2] in which dual-rail logic cones are independently mapped. This approach has the advantage of maintaining the mapping flexibility of each polarity. However, the presence of an XOR function and its recovergence property will decompose the input network into very small mapping trees, which causes a large area and delay cost for tree-by-tree technology mapping. On the other hand, dual-monotonic XOR is a widely-used configuration in manually designed domino networks; its application to synthesis of domino logic can effectively solve the above problem. Therefore, we propose a mapping method that can make use of the advantages of both cases by mapping dual-rail AND/OR logic independently with standard domino gates, and mapping XOR/XNOR logic with merged dual-monotonic gate.

2.3 Dual-monotonic mapping algorithm

The method consists of the following steps:

Step 1 Generate a DAG network of two-input AND/OR gates for the given circuit.

Step 2 Recognize the XOR/XNOR logic inside the DAG by pattern matching. If both positive and negative polarity of XOR/XNOR logic are required, the XOR/XNOR logic networks are collapsed into one XOR/XNOR node to be implemented as a dual monotonic gate.

Step 3 Perform the technology mapping on the AND/OR/XOR/XNOR subject network, mapping AND/OR nodes to the standard domino gates and XOR/XNOR nodes to various dual-monotonic gates.

The last step of the dual-monotonic mapping algorithm is technology mapping on an AND/OR/XOR/XNOR network. During technology mapping, all possible matchings are enumerated at each network node. While the traditional mapping patterns can be applied to AND/OR nodes, the matching patterns available to XOR/XNOR

nodes need to be considered. One single XOR/XNOR node can be mapped to a XOR dual-monotonic gate. Moreover, the flexible configurations of dual-monotonic gates enable the exploration of more matching patterns at XOR/XNOR nodes. For example, the matching pattern of Figure 3(a) in the network corresponds to a three input XOR gate as Figure 3(b).

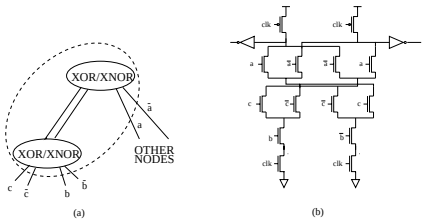

Figure 3: Matching pattern 1: three input XOR gate

The matching pattern of Figure 4(a) corresponds to the dual-monotonic gate as Figure 4(b). It is obtained by replacing the transistors D and G in Figure 2 by NMOS subnetworks.

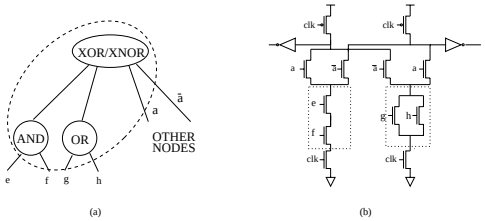

Figure 4: Matching pattern 2: arbitrary AND/OR/XOR logic

In this work, we incorporate the above issues in the parameterized mapper in [4], but any other mapper may easily be adapted for this purpose.

3 Output phase assignment using 0-1 ILP

3.1 Previous work and motivation

The output phase assignment problem was first addressed and solved in [5]. The problem was defined as follows: Given a combinational logic network and all primary inputs in the true and complemented form, choose an optimal phase (i.e., polarity) assignment for the primary outputs so as to require minimal logic duplication for obtaining inverter free logic.

In [5], given an input network, the network is divided into the region that must be duplicated and region whose duplication cost can be minimized by output phase assignment, referred to as the optimizable logic region. The fanout nets in the optimizable logic region are called candidate nets. The problem of finding optimal output phase assignment to minimize the duplication of optimized logic region was formulated as 2SAT unate covering problem and solved by using binary decision diagrams.

Our solution to the output phase assignment problem improves [5] in the following aspects.

In our mapper, the output assignment problem is formulated into a 0-1 integer programming problem and a standard linear programming package can be used to solve the problem.

The objective of our phase assignment problem is to minimize the implementation cost of the mapped network instead of minimizing logic duplication in the unmapped network, as in [5]. From our observation, the output phase assignment accomplishes the objective of reducing implementation cost through several factors. Reducing logic duplication is one such factor, and the cost difference between the implementations of positive and negative polarity logic is another important consideration of output phase assignment. Suppose W and H are the constraints on the width(maximal number of parallel chains) and height(maximal number of series chains) of the NMOS pull-down network of domino gate, respectively. Due to the absence of a complementary PMOS network in domino gate, domino gates usually have large W with limited H, which causes the cost difference between implementations of positive and negative polarity.

For example, given the constraints that $H = 2$ and $W = 4$, the logic network of Fig 5(a) will be mapped to three domino gates while its complement in Fig 5(b) requires only one domino gate.

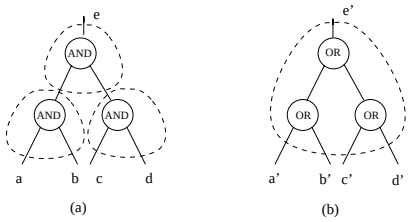

Figure 5: The cost difference of positive and negative polarity

3.2 Algorithm outline

The algorithm can be outlined as follows:

1. Decompose the network into a disjoint set of trees and obtain the area cost estimation of each tree by counting number of nodes in the tree or performing tree-by-tree mapping on the network. The area cost of a tree is assigned to the root node of the tree, u, as the weight $C(u)$.
2. Find the optimizable logic region using the method of [5]; build a DAG from the multiple fanout nodes of the optimizable logic region.
3. Write out the 0-1 integer linear programming formulation and solve it.

3.3 0-1 ILP for minimal duplication cost

Given an input network, a DAG $G(V,E)$ can be built as follows. Each vertex $v \in V$ corresponds to a multiple fanout or a primary output node in the optimizable logic region of input network. If vertex u is a literal node of a tree rooted at vertex v in the original input network, there is a corresponding edge $e_{u,v} \in E$ in DAG.

In the DAG, several constants are assigned to an edge or a node from the original input network. These include:

- $O(u,v)$, which represents the inversion polarity between vertex u and vertex v. If there is an even number of inverters from vertex u to vertex v in the input network, $O(u,v)$ is 0; otherwise, $O(u,v)$ is 1.
- $C(u)$, which represents the area cost of the tree whose root is at vertex u in the input network.
- $k(u)$, a constant whose value is twice the number of fanout of vertex u.

The $\{0,\ 1\}$ integer variables that will be included in the 0-1 programming formulation include:

- $r(u) = 1$ if there is an inverter moving from the inputs of node u towards the outputs of node u; else 0.

• $x(u,v) = 1$ if there is an inverter on the edge $e_{u,v}$ after the outputphase assignment; else 0.

• $y(u,v)$ is a dummy variable that transforms a condition statement into a linear constraint.

• $q(u) = 1$ if the fan-in tree of node i needs to be duplicated; else 0.

The weight of an edge $e_{u,v}$ after output phase assignment, denoted by $w(u,v)$ is given by

$$w(u,v) = O(u,v) + r(u) - r(v) \tag{1}$$

Since $O(u,v)$, $r(u)$, $r(v) \in \{0,1\}$, $w(u,v)$ takes a value in the set $\{-1,0,1,2\}$. If $w(u,v) = 0$ or 2, it represents the absence of an inverter between node u and node v. If $w(u,v) = -1$ or 1, then there is one inverter between node u and node v. Therefore, this may be captured by the condition

$$x(u,v) = \begin{cases} 0 & w(u,v) \in \{0,2\} \\ 1 & w(u,v) \in \{-1,1\} \end{cases} \tag{2}$$

The above condition may be rewritten as a linear equation.

$$w(u,v) + x(u,v) = 2 \times y(u,v) \tag{3}$$

where $y(u,v) \in \{0,1\}$ is a dummy variable introduced to transform a condition statement into a linear constraint.

Combining the equations (1) and (3), we have

$$2 \times y(u,v) - x(u,v) = O(u,v) + r(u) - r(v) \tag{4}$$

If there is an inverter at any position in the fanout cone of a node u, the node will have to be duplicated to ensure the unateness property for domino logic. This condition can be given by the linear formula

$$q(u) \times k(u) - \sum_{i \in dir-succ(u)} (x(u,i) + q(i)) \geq 0 \tag{5}$$

It can be illustrated as Figure 6. Here, $i \in dir-succ(u)$ implies that there is an edge from node u to node i in the DAG $G(V,E)$ defined in Section 3.3. The constant $k(i)$ was defined earlier and can easily be verified to always be larger than $\sum_{i \in dir-suc(u)(u)} (x(u,i) + q(i))$. This formula implies that if any of its successors needs to be duplicated, or if there is one inverter at the output edges of node u, node u will have to be duplicated as $q(u)$ is forced to be 1; otherwise the objective function will force $q(u)$ to 0.

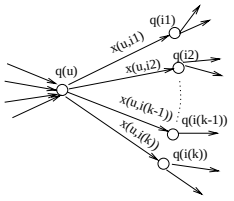

Figure 6: 0-1 programming constraints

Therefore, the output phase assignment problem can be formulated as the 0-1 integer linear programming problem:

$$\begin{gathered} \text{minimize } \textstyle\sum_{u \in V} C(u) \times q(u) \\ \text{subject to} \\ 2 \times y(u,v) - x(u,v) = O(u,v) + r(u) - r(v) \quad \forall e_{u,v} \in E \\ q(u) \times k(u) - \textstyle\sum_{i \in dir-succ(u)} (x(u,i) + q(i)) \geq 0 \quad \forall u \in V \\ q(u), r(u) \in \{0,1\} \qquad \forall u \in V \\ y(u,v), x(u,v) \in \{0,1\} \qquad \forall e_{u,v} \in E \end{gathered}$$

3.4 0-1 ILP for minimal implementation cost

This section modifies the 0-1 formulation of Section 3.3. to incorporate the cost difference between positive and negative polarity implementations to obtain the minimal implementation cost.

In addition to the notation defined in 3.3, the symbols that will be used include:

$Z(u)$ is a constant that represents the cost difference between optimal implementation of each of the two polarities. If the implementation cost of positive[negative] polarity of the node u is $p(u)[n(u)]$, then the value of $Z(u) = p(u) - n(u)$.

We introduce $t(u)$ as a variable that help to maximize $r(u)$. Its value can be expressed as follows.

$$t(u) = \begin{cases} 0 & q(u) = 1 \\ r(u) & q(u) = 0 \end{cases} \tag{6}$$

Similarly, we also define a variable $s(u)$to minimize $r(u)$ as

$$s(u) = \begin{cases} 0 & q(u) = 1 \\ 1 - r(u) & q(u) = 0 \end{cases} \tag{7}$$

There are three possibilities for each node after output phase assignment. A node will either be duplicated, implemented in positive polarity, or in negative polarity. In the case of no duplication at node u, if $Z(u) > 0$, it costs less to implement the node with negative polarity; if $Z(u) < 0$, it is better to implement the node u with positive polarity. Hence, using the previous definitions of $q(u)$ and $r(u)$, the problem can be expressed as

$$\begin{gathered} \text{if } q(u) = 0 \textbf{ and } Z(u) > 0, \text{maximize } r(u); \\ \text{if } q(u) = 0 \textbf{ and } Z(u) < 0, \text{minimize } r(u). \end{gathered}$$

The first statement can be expressed by linear equations

$$\begin{array}{ll} \text{minimize} & -Z(u) \times t(u) \\ \text{subject to} & t(u) \leq 1 - q(u) \\ & t(u) \leq r(u) \\ & q(u), r(u), t(u) \in \{0,1\}, \qquad \text{if } Z(u) > 0, \end{array}$$

When $q(u) = 1$, $t(u)$ is forced to 0. Hence, there is no limit on $r(u)$ and the duplication cost is given by $C(u) \times q(u)$. When $q(u) = 0$, $r(u)$ is forced to be as large as possible since the negative value of $-Z(u)$. When polarity preferences of two nodes conflict with each other, $Z(i)$ is the weight that decides which node should win. Similarly, the second statement can be captured by

$$\begin{array}{ll} \text{minimize} & Z(u) \times s(u) \\ \text{subject to} & s(u) \leq 1 - q(u) \\ & s(u) \leq 1 - r(u) \\ & q(u), r(u), s(u) \in \{0,1\}, \qquad \text{if } Z(u) < 0 \end{array}$$

Combining with the linear equations of section 3.3, the 0-1 ILP formulation of the output phase assignment problem for cost minimization can be rewritten as

$$\text{minimize} \sum_{u \in V} C(u) \times q(u) - \sum_{Z(u)>0} Z(u) \times t(u) + \sum_{Z(u)<0} Z(u) \times s(u)$$

subject to

$$\begin{aligned}
2 \times y(u,v) - x(u,v) &= O(u,v) + r(u) - r(v) \quad \forall e_{u,v} \in E \\
q(u) \times k(u) - \sum_{i \in dir-succ(u)} (x(u,i) + q(i)) &\geq 0 \quad \forall u \in V \\
t(u) \leq 1 - q(u), \quad t(u) &\leq r(u) \qquad \forall Z(u) > 0 \\
s(u) \leq 1 - q(u), \quad s(u) &\leq 1 - r(u) \qquad \forall Z(u) < 0 \\
q(u), r(u), t(u), s(u) &\in \{0,1\} \qquad \forall u \in V \\
y(u,v), x(u,v) &\in \{0,1\} \qquad \forall e_{u,v} \in E
\end{aligned}$$

4 Experimental results

The above algorithms are implemented using C++ and incorporated into the framework of the parameterized mapper of [4]. The experiments were executed on the LGSynth91 multi-level combinational circuit sets. All of the input circuits are optimized with *script.rugged* of SIS and targeted to the area minimization. In the result tables, the results are presented in terms of area and delay, where the area is estimated as the transistor count while the delay is estimated by a coarse measure that counts the number of gate levels.

4.1 Dual-monotonic gate mapping

All of results in Table 1 were obtained from domino gates with the limits of $W = 4, H = 4$, where W and H is as defined in Section 3.1. A comparison of our dual-monotonic gate mapping results with [4] are shown in Table 1. The second column shows the results obtained from the basic parameterized mapping algorithm of [4]. Column 3 contains the results obtained from the dual-monotonic mapping algorithm of section 2 while Column 4 shows the corresponding area reduction. Column 5 reports the number of XOR/XNOR gates detected in the circuits. The execution time of both algorithms are almost same and are less than 10 seconds for all circuits.

From the results, we can see that in the circuits with significant XOR substructures, the dual-monotonic gate mapping is quite effective. The presence of dual-monotonic gates can help to reduce the gate level of the input circuits in some case.

Table 1: Dual-monotonic gate mapping

Circuits	Basic area/delay	Dual-mono area/delay	Reduct %	#XOR
b9	254/5	254/5	0 %	0
c8	279/6	279/6	0 %	0
count	287/9	287/9	0%	0
i6	761/3	761/3	0 %	0
C880	1163/20	1051/20	9.6 %	16
C1355	1824/9	1360/7	25.4 %	72
C1908	1978/18	1588/14	19.7 %	50
C2670	1944/12	1729/12	13.7 %	40
C3540	4527/23	4241/20	6.3 %	38
C6288	13702/71	10629/57	28.9 %	419
C7552	7919/18	6613/16	16.6 %	213
rot	1774/10	1772/10	0.1 %	1
dalu	2347/13	2338/13	0.4%	3
k2	2884/16	2884/16	0 %	0
des	9945/10	9843/10	1.0 %	51

4.2 Output phase assignment

Table 2 shows the efficacy of our 0-1 programming output phase assignment algorithms. To demonstrate the influence of cost difference between two polarities to total implementation cost, the domino gates were restricted to $W = 8$ and $H = 2$.

Column 2 contains the number of primary output in the circuits. Column 3 lists the results when the output phase assignment are not used. Column 4 shows the results when 0-1 formulation of 3.3 is applied. Column 5 shows the results obtained by incremental 0-1 programming formulation in 3.4 while Column 6 shows the corresponding area reduction. We use the linear program solver *lp_solve_2.3* [6] to solve the 0-1 integer linear programming formulas. All of the benchmarks can solved in under one minute with no or minor simplification of 0-1 ILP except for the 0-1 programming problem for Circuit k2, which is too large to yield the result in reasonable time.

It was observed that in some circuits such as dalu, the cost reduction by output phase assignment mainly comes from the duplication cost reduction; in some circuits such as i6 and b9, the implementation cost difference between two polarities becomes the most significant consideration of output phase assignment. In some case, the two objectives of output phase assignment can be contradictory to each other. In Circuit x1, the optimization for duplication cost minimization causes a cost increase due to the increased cost of implementing the reversed polarity and increases the total implementation cost.

Table 2: Output phase assignment using a 0-1 ILP

Circuits	#po	no-ass area	opt-ass1 area	opt-ass2 area	reduction %
b9	21	391	389	311	20.5%
c8	18	407	364	330	18.9%
i6	67	1298	1281	766	41.0%
C1355	32	2064	2064	2064	0.0%
C2670	140	2647	2624	2414	8.8%
C6288	32	14257	14257	14252	0.0%
C7552	108	9069	9069	8904	1.8%
rot	107	2332	2296	2196	5.8%
dalu	16	3020	2752	2745	9.1%
k2	45	3974	-	-	-
des	245	13130	13130	11710	10.8%
apex7	37	833	791	736	11.6%
frg1	3	362	362	362	0%
x1	35	861	927	850	1.3%
x3	99	2381	2360	2100	11.8%

5 Conclusion

In this paper, we have explored a new domino logic mapping method and proposed a solution to the output phase assignment problem. It was shown that both methods can reduce the area of the mapped circuits significantly with very limited computation overhead. By replacing the area cost of logic trees with power cost, we expect our 0-1 ILP of output phase assignment can serve the power minimization objective proposed in [7] similarly.

References

[1] T. Williams, "Dynamic logic: Clocked and asynchronous." Tutorial notes at the ISSCC, 1996.

[2] M. R. Prasad, D. Kirkpatrick, and R. K. Brayton, "Domino logic synthesis and technology mapping," in *IWLS*, 1997.

[3] T. Thorp, G. Yee, and C. Sechen, "Domino logic synthesis using complex static gates," in *Proc. ICCAD*, pp. 242–247, 1998.

[4] M. Zhao and S. S. Sapatnekar, "Technology mapping for domino logic," in *Proc. ICCAD*, pp. 248–251, 1998.

[5] R. Puri, A. Bjorksten, and T. E. Rosser, "Logic optimization by output phase assignment in dynamic logic synthesis," in *Proc. ICCAD*, pp. 2–8, 1996.

[6] M. Berkelaar, "LP SOLVE 2.3 Users' Manual," 1998.

[7] P. Patra and U. Narayanan, "Automated phase assignment for the synthesis of low power domino circuits," in *Proc. ICCAD*, pp. 379–384, 1999.