Logic Optimization by Output Phase Assignment in Dynamic Logic Synthesis

Ruchir Puri
IBM Thomas J. Watson Research Center
Yorktown Heights, NY - 10598

Andrew Bjorksten Thomas E. Rosser
IBM Corporation, 11400 Burnet Road
Austin, TX - 78758

Abstract

Domino logic is one of the most popular dynamic circuit configurations for implementing high-performance logic designs. Since domino logic is inherently non-inverting, it presents a fundamental constraint of implementing logic functions without any intermediate inversions. Removal of intermediate inverters requires logic duplication for generating both the negative and positive signal phases, which results in significant area overhead. This area overhead can be substantially reduced by selecting an optimal output phase assignment, which results in a minimum logic duplication penalty for obtaining inverter-free logic. In this paper, we present this previously unaddressed problem of output phase assignment for minimum area duplication in dynamic logic synthesis. We give both optimal and heuristic algorithms for minimizing logic duplication.

1 Introduction

The use of dynamic logic in high-performance microprocessor design is an efficient way of increasing circuit speed and reducing area. Domino logic [6] allows a single clock to precharge and evaluate a cascade of dynamic logic blocks and requires incorporating a static CMOS inverting buffer at the output of each dynamic logic gate as shown in Figure 1. Inspite of various area and speed advantages, the inherently non-inverting nature of domino gates requires the implementation of logic network without inverters. This *inverter-free logic constraint is a fundamental constraint for implementing logic functions with domino gates.*

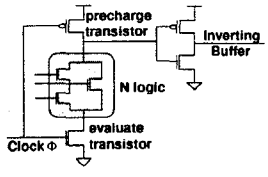

Figure 1: Basic domino CMOS gate.

CMOS static logic is always synthesized using the flexibility of manipulating inverters in the logic network. The inverter-free constraint in domino logic design limits this flexibility. This constraint implies that all the logic inversions should be performed at the clock phase boundaries, i.e., at the primary inputs or primary outputs where the inverters can be absorbed in registers. Thus the first step in domino logic synthesis is to make the logic inverter-free. A straight forward approach is to convert the technology independent logic into AND, OR, and NOT gates only. Subsequently, starting at the primary outputs, the inverters can be propagated back towards the inputs by applying simple De Morgan's laws. If an inverter is trapped[1] at the fanout of a gate G, then gate G is duplicated for implementing both positive and negative phases and the inverter is pushed backward. Pushing the inverters back towards primary inputs is guaranteed to restrict the increase in the size of the circuit to at most twice the original size and not increase the number of logic levels [8]. This procedure transforms the given logic network into an inverter-free logic network with inverters at its primary inputs only.

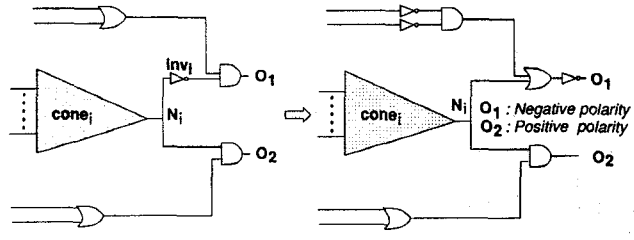

Figure 2: Output phase assignment for eliminating trapped intermediate inverters.

In general, the duplication penalty for removing the trapped inverters all the way to the primary inputs can be quite heavy in terms of circuit area. For example, we have observed a 10% to 80% area overhead in microprocessor design partitions if all the trapped inverters are removed by pushing them to the primary inputs. In addition, this area overhead may also result in a substantial power dissipation penalty. One of the solutions is to propagate some of the inverters forward towards primary outputs thereby avoiding the logic duplication. Since some inverters can be propagated forward and others can be propagated backward, the process may be very complex in a practical design and the choices exponential. The process of propagating an inverter forward is equivalent to chosing an implementation phase (i.e., polarity) of the outputs that eliminates this inverter. This is illustrated in Figure 2, where inv_i is trapped at fanout net N_i. Propagating inv_i back towards primary inputs will require duplication of fanin logic cone $cone_i$. This duplication can be avoided by propagating inv_i forward towards primary outputs which is simply equivalent to implementing primary output O_1 in nega-

[1] An inverter is said to be *trapped* at a fanout net N_i, if it cannot be propagated back towards primary inputs without duplicating the logic gate that is feeding the fanout net N_i.

tive phase. Since there are 2^n possible phase assignments for implementing n primary outputs, it is non-trivial to find an optimal output phase assignment for minimum logic duplication.

To the best of our knowledge, the problem of logic optimization by output phase assignment in dynamic logic synthesis has not been addressed before. A variation of this problem was alluded to by Brayton et al. in [1]. In [2], they addressed the problem of output phase assignment for obtaining minimum single ended domino trees, i.e., minimum number of inverters trapped at fanout nets. They viewed this problem as being similar to optimal output phase assignment problem for PLAs [9][10] and gave a solution using a branch and bound algorithm. We show that the inverter minimization problem is a special case of the more general *minimal logic duplication problem* and can be formulated as a simple case of the unate covering problem.

In the following section we formulate the problem of output phase assignment for minimum area duplication in domino logic synthesis and give both optimal and heuristic algorithms for minimizing logic duplication.

2 Inverter-free Logic Synthesis

As described above, the problem of minimum logic duplication for obtaining inverter-free domino logic is reduced to chosing an optimal output phase assignment such that the logic duplication penalty is minimized.

Problem Definition: *Given a combinational logic network, the output phase optimization problem in dynamic logic synthesis is to chose an optimal phase (i.e., polarity) assignment for the primary outputs so as to require minimal logic duplication*[2] *for obtaining inverter-free logic.*

In the most general case, every fanout net in the logic network is a potential candidate where an inverter may get trapped if proper output phase assignment is not chosen. In practice, many of the inverters that are trapped at intermediate fanout nets cannot be eliminated by selecting a phase assignment of the primary outputs. In the following Section, we analyze the inverters which cannot be eliminated by output phase optimization.

2.1 Inverters trapped at Reconvergent Fanouts

Logic networks that are fanout-free can also be made inverter-free without any duplication penalty by simply pushing all the inverters back towards primary inputs. In such a case, the inverters cannot get trapped at any intermediate fanout net and no logic duplication is required. In comparison, logic networks that are not fanout-free may require logic duplication to remove

[2] *Optimal output phase assignment solution for minimal logic duplication* implies that the total area of technology independent logic gates that must be duplicated to obtain an inverter-free logic must be minimum, i.e., there is no other output phase assignment solution that can yield a better logic duplication area.

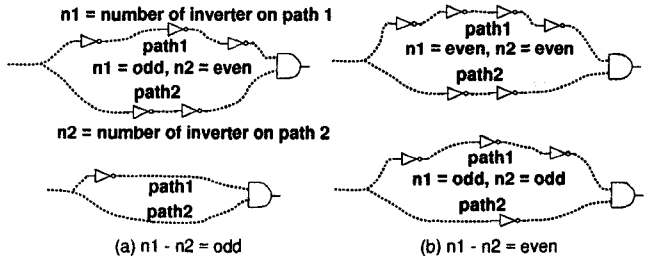

Figure 3: Inverters trapped in reconvergent fanouts

trapped inverters. In general, fanout nets can be divided into two categories: reconvergent fanout nets and non-reconvergent fanout nets. A reconvergent fanout is the root of an undirected closed loop of gates as shown in Figure 4 by highlighted lines. In the following, we analyze reconvergent fanout nets.

Let n_1 be the number of inverters on $path_1$ of the reconvergent loop and let n_2 be the number of inverters on $path_2$ of the reconvergent loop, as shown in Figure 3. There are two possible scenarios which will determine whether duplicating logic in the fanin cone of reconvergent fanout net is essential, i.e., if the difference in number of inverters on $path_1$ and $path_2$, i.e., $n_1 - n_2$ is: (a) an *odd* number (as shown in Figure 3(a)) and (b) an *even* number (as shown in Figure 3(b)). In the following, we analyze these cases in detail.

Case 1: $n_1 - n_2$ is *odd* implies that $path_1$ has an odd number of inverters and $path_2$ has an even number of inverters or vice-versa. If $path_1$ has an odd number of inverters, then they can be reduced to only one inverter by propagating them forward/backward. Similarly, an even number of inversions in $path_2$ can be eliminated by combining them. Thus we will always have one inverter trapped in one branch of the reconvergent loop. This inverter can only be made to toggle between the two paths but cannot be pushed out of the reconvergent loop. This is illustrated in Figure 3(a).

Case 2: $n_1 - n_2$ is *even* implies that either both $path_1$ and $path_2$ have an even number of inverters or both of them have an odd number of inverters. If there are an odd number of inverters in both $path_1$ and $path_2$, then they can be reduced to only one inverter in each path by propagating them forward/backward. This leaves one inverter in each path which can be propagated back through the fanout net. Thus the condition where one inverter is trapped in one branch of the reconvergent loop is avoided. If there are an even number of inverters in both $path_1$ and $path_2$, then they can be eliminated from each path by combining them. This leaves no inverter in either $path_1$ or $path_2$. Thus in the case of $n_1 - n_2$ is *even*, an inverter can never get trapped at the reconvergent fanout. This is illustrated in Figure 3(b).

As described above in Case 1, an inverter will be trapped in the reconvergent fanout loop only if the difference in the number of inverters between two branches of the reconvergent fanout loop is an *odd* number. To

eliminate this inverter the logic fanin cone rooted at the reconvergent fanout net must be duplicated. Thus the logic in fanin cones of all the reconvergent fanout nets with trapped inverters must be duplicated. This defines a bipartition on the logic network : logic that must be duplicated for removing inverters, i.e., *duplicated logic region* and logic that can be minimized for logic duplication by an optimal output phase assignment, i.e., *optimizable logic region.* Thus all the nets in the duplicated logic region need not be considered for minimizing logic duplication, since the logic in their fanin cones has already been duplicated. It is obvious that an inverter can never get trapped in a inverter-free reconvergent fanout loop. Thus inverter-free reconvergent fanouts need not be considered for logic optimization by output phase assignment. Only non-reconvergent fanout nets in the optimizable logic region must be considered for minimizing logic duplication. These non-reconvergent fanout nets are called *candidate nets* which are used in determining the output phase assignment.

The above analysis significantly prunes the number of fanout nets that must be considered for an optimal output phase assignment for minimizing logic duplication. Finding candidate nets requires the knowledge of all the reconvergent fanouts in the logic network, which in itself may be very complex. In the following Section, we describe an efficient procedure that determines the candidate nets for output phase optimization by a simple traversal through the logic network from primary outputs to primary inputs.

2.2 Determining the candidate nets

Since the fanout nets are the only possible candidates where inverters may get trapped, every fanout net defines an output phase assignment for eliminating the trapped inverter or for avoiding an inverter being trapped. The output phase assignment corresponding to every fanout net can be determined by a simple traversal from primary outputs towards primary inputs as follows.

In the given combinational logic network with m primary outputs $O_1, O_2, \ldots, O_m$, we associate a phase assignment vector $\{v_{i_1}, v_{i_2}, \ldots, v_{i_m}\}$ with every fanout net N_i in the logic network, where v_{i_j} represents the phase of a primary output O_j required to eliminate the trapped inverter or to avoid an inverter being trapped at fanout net N_i. Let P and N represent the *positive* phase and the *negative* phase of a primary output respectively. Net N_i defines the phase assignment of primary output O_j (i.e., the value of v_{i_j}) if O_j is in the fanout cone of net N_i. If primary output O_j is not in the fanout cone of net N_i, then net N_i is not affected by this output and the corresponding phase value v_{i_j} is assigned a don't care value (indicated by a $-$). If O_j is in the fanout cone of N_i and every path from N_i to O_j contains an even number of inverters, then corresponding phase assignment v_{i_j} is defined as *positive* (P). Similarly, if every path from O_j to N_i contains an odd number of inverters then corresponding phase assignment v_{i_j} is defined as *negative* (N). If one path from N_i to O_j contains an even number of inverters and another path contains an odd number of inverters, then there will be two conflicting values of v_{i_j}, i.e., positive due to the path with even number of inverters and negative due to the path with odd number of inverters. As shown in Figure 3(a), this condition characterizes a reconvergent fanout with a trapped inverter and requires the duplication of logic cone rooted at fanout net N_i. Thus in such a case, none of the nets in the fanin cone of net N_i are considered for minimizing logic duplication.

Initially, for each primary output O_j, the output phase assignment of corresponding net N_i is initialized as $\{v_{i_1} = -, v_{i_2} = -, \ldots, v_{i_j} = P, \ldots, v_{i_m} = -\}$. To determine the logic duplication boundary, i.e., to find the optimizable logic region, we start from primary outputs and traverse towards primary inputs while propagating phase assignments as follows. In the case of an AND/OR gate, the input pins of the gate are given the same phase assignment as its output. In the case of a NOT gate, the input pin of the gate is given a complementary phase assignment as its output. A fanout net N_i will receive a phase assignment from every logic gate it feeds. The phase assignment for a net N_i can be obtained by combining the phase assignment of all its fanouts. A positive or a negative phase assignment when combined with a don't care phase assignment will yield a positive or negative phase assignment, respectively. If a fanout net N_i receives conflicting phase assignment for an output O_j, i.e., negative and positive both, then it is root of a reconvergent fanout loop with a trapped inverter (as described in Section 2.1). Thus fanout net N_i is a net on the logic duplication boundary. This implies that all the nets in the fanin cone of net N_i can be eliminated from consideration for minimizing logic duplication and we can avoid traversing further from net N_i. In this way, through a simple traversal from primary outputs towards primary inputs, we can determine the duplicated logic region and the optimizable logic region. All the fanout nets in the optimizable logic region are possible candidates for minimizing logic duplication. Since the optimizable logic region does not contain any reconvergent fanout nets with a trapped inverter, fanout nets in this region can only be non-reconvergent; or reconvergent without a trapped inverter. As explained in Section 2.1, an inverter can never get trapped in a reconvergent fanout loop which is inverter-free. This eliminates the reconvergent fanout nets without a trapped inverter from consideration for minimizing logic duplication. Thus only non-reconvergent fanout nets in optimizable logic region are candidates for minimizing logic duplication. A fanout net N_i in optimizable logic region is non-reconvergent if it receives any two disjoint phase assignments from its fanout gates.

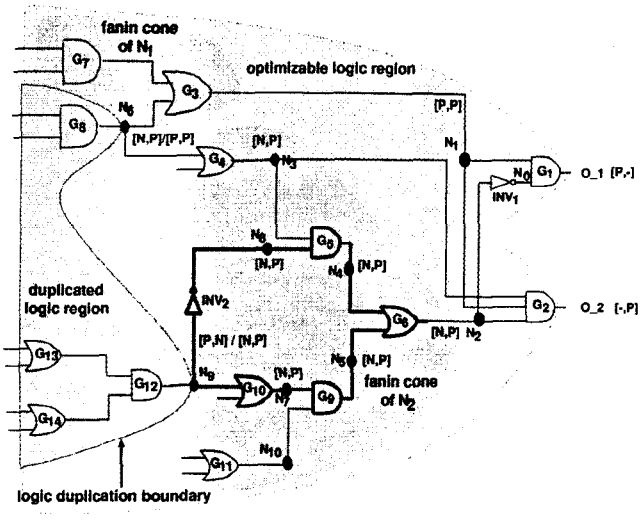

Figure 4: Output phase assignment: An example.

Definition : Two phase assignments $\{v_{i_1}, v_{i_2}, \ldots, v_{i_m}\}$ and $\{v'_{i_1}, v'_{i_2}, \ldots, v'_{i_m}\}$ are said to be *disjoint* if and only if for every output O_j at least one of the corresponding phase assignment values v_{i_j} and v'_{i_j} is a don't care, i.e., $\forall j$ *either* $v_{i_j} = -$ *or* $v'_{i_j} = -$.

For example, phase assignments $v_i = \{--PN-\}$ and $v'_i = \{-P--N\}$ are *disjoint*. In comparison, phase assignments $v_i = \{--PN-\}$ and $v'_i = \{-PP-N\}$ are *not disjoint* since for output O_3 both the phase assignments are defined, i.e., $v_{i_3} = v'_{i_3} = P$. The candidate nets for minimizing logic duplication, i.e., non-reconvergent fanout nets in the optimizable logic region can be determined simply by traversing all the fanout nets in the optimizable logic region and checking them for disjoint phase assignments as defined above. In the following, this procedure is explained with a simple example shown in Figure 4.

Example : After the initial step of propagating inverters towards primary inputs without any logic duplication, two inverters INV_1 and INV_2 are trapped at fanout nets N_2 and N_9 respectively, as shown in example logic network of Figure 4. To find the duplicated logic region and the optimizable logic region, the phase assignments are initialized at the primary outputs and they are propagated towards primary inputs. Since logic gates $G_1, G_2, \ldots, G_{14}$ are of type AND/OR, their input pins will receive the same phase assignment as their output. The input pin of an inverter will receive the complementary phase assignment as its input, e.g., inverter INV_1 output net $N_0's$ phase assignment $[P, -]$ will be propagated as $[N, -]$ to its input N_2. Thus fanout net N_2 receives an assignment $[N, -]$ from INV_1 and an assignment of $[-, P]$ from gate G_2. These two assignments are combined to yield the phase assignment of fanout net N_2, i.e., $[N, P]$. Similarly, the phase assignment of fanout net N_3 can also be obtained as $[N, P]$. Fanout net N_9 receives a phase assignment of $[P, N]$ from INV_2 and a phase assignment of $[N, P]$ from gate G_{10}. Since these two assignments are conflicting, net N_9 is a reconvergent fanout with a trapped inverter, as shown by highlighted lines in Figure 4. This implies that INV_2 cannot be removed and the fanin cone of N_9, i.e., G_{12}, G_{13}, and G_{14}, must be duplicated for removing inverter INV_2. Thus fanout net N_9 is a net at the logic duplication boundary and may not be considered as a candidate for minimizing logic duplication by output phase assignment to obtain inverter free logic. Similarly, fanout net N_6 receives conflicting assignments $[P, P]$ from gates G_3 and $[N, P]$ from gate G_4. Thus fanout net N_6 is root of a reconvergent fanout loop $\{N_6, N_1, O_1, N_2, N_4, N_3, N_6\}$ with a trapped inverter INV_1. As shown in Figure 4, both nets N_6 and N_9 define the duplicated logic region with gates G_8, G_{12}, G_{13}, and G_{14}. As a result, only fanout nets N_1, N_2, N_3 in the optimizable logic region may be considered for minimizing logic duplication by chosing an optimal output phase assignment. Among these nets, all the phase assignments that fanout net N_3 receives from its fanout gates are non-disjoint, i.e., assignment $[-, P]$ from gate G_2 and assignment $[N, P]$ from gate G_5 are not disjoint. Thus fanout net N_3 is a reconvergent fanout without a trapped inverter and an inverter can never be trapped inside this reconvergent loop by chosing any output phase assignment. This implies that net N_3 can also be eliminated as a candidate net for minimizing logic duplication. This yields only two fanout nets N_1 and N_2 that are candidates for selecting an optimal output phase assignment.

The candidate nets obtained above can be utilized for selecting an output phase assignment for minimal logic duplication to obtain inverter free logic. In the following section, we formulate this problem as a graph problem and solve it using boolean satisfiability framework.

2.3 Minimizing logic duplication

Every candidate fanout net in the optimizable logic region defines a phase assignment for primary outputs that will eliminate a trapped inverter or avoid an inverter being trapped. In general these assignments may be conflicting with each other, i.e., chosing a phase assignment enforced by candidate net N_i may avoid an inverter being trapped at N_i but may trap an inverter at another candidate net N_j. The conflicting nature of output phase assignments enforced by two candidate nets is defined using the *incompatibility* constraint as follows.

Definition : Candidate nets N_i and N_j are said to be *incompatible* if they define at least one conflicting primary output phase assignment otherwise they are said to be *compatible*.

For example in Figure 4, candidate net N_1 defines an output phase assignment $[P, P]$ to avoid an inverter being trapped. Similarly, candidate net N_2 defines an output phase assignment $[N, P]$ for eliminating a trapped inverter INV_1. Since these assignments define conflicting positive and negative phase values for primary out-

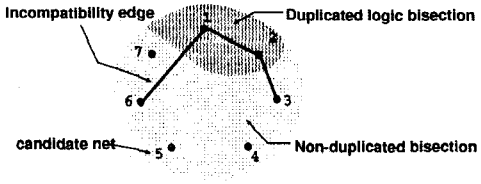

Figure 5: Incompatibility graph.

put O_1, candidate nets N_1 and N_2 are *incompatible.*

The incompatibility constraint implies that at least one of the two incompatible nets must have a trapped inverter. If candidate nets N_i and N_j are incompatible, then chosing the phase assignment[3] enforced by N_i will trap an inverter at N_j and vice-versa. Thus the logic fanin cone of one of the candidate nets must be duplicated. The tradeoff between duplicating logic fanin cone of one net over another depends on the associated area duplication penalty. The penalty for duplicating a logic cone can be computed by adding the technology independent area of its logic gates. Since the logic gates in duplicated logic region have already been duplicated, only logic gates in the optimizable logic region are considered for calculating duplication penalty. Using the incompatibility constraint, the output phase assignment problem for minimum logic duplication to obtain inverter free logic can be restated as follows.

Problem : *Given a set of optimizable candidate nets* $\{N_1, N_2, \ldots, N_n\}$ *and their incompatibility constraints, find a phase assignment to the primary outputs that yields minimum logic area duplication penalty for obtaining inverter free logic.*

We can formulate this problem using a graph model and solve it using boolean satisfiability approach described as follows.

2.3.1 Graph Theoretic Formulation

We represent the candidate nets and the incompatibility relationships in a *incompatibility graph* as shown in Figure 5. A candidate net N_i is represented by a node i in the graph. Two nodes i and j are connected by an undirected edge if their corresponding candidate nets N_i and N_j are incompatible. In Figure 5 there are seven nodes $1, 2, \ldots, 7$ corresponding to seven candidate nets $N_1, N_2, \ldots, N_7$ respectively. Assume that these candidate nets require following output phase assignments, i.e., $N_1 : [PP-P-]$; $N_2 : [N-P--]$; $N_3 : [P-PP-]$; $N_4 : [--PP-]$; $N_5 : [--P-N]$; $N_6 : [-NP-N]$; and $N_7 : [---PN]$. Among these nets, the output phase assignments of candidate nets N_1 and N_6 are incompatible, i.e., only one of them can be chosen to avoid the duplication penalty of its fanin logic cone. This is represented by an incompatible edge between corresponding nodes 1 and 6 in the graph shown in Figure 5. Similarly, N_1 ,N_2 and N_2, N_3 are incompatible. Since the logic fanin cones of two incompatible nets may be overlapping in general, the duplication cost associated with a candidate net may not be independent of the other candidate nets. This constrains us from assigning independent weights for duplication penalty to individual nodes in the graph.

As shown in Figure 5, the solution of *output phase assignment problem* on the incompatibility graph corresponds to dividing the incompatibility graph into two partitions such that no two nodes connected by an incompatible edge are in the bisection corresponding to non-duplicated logic and the area penalty cost associated with the duplicated logic region is minimum. Since the output phase assignments of candidate nets that do not have any incompatibility constraint can be combined without any output phase conflict, the corresponding nodes in the incompatibility graph are assigned to non-duplicated logic bisection. For example in the incompatibility graph of Figure 5, nets N_4, N_5, and N_7 are not associated with any incompatibility edge. Thus their output phase assignments, i.e., $N_4 : [--PP-]$, $N_5 : [--P-N]$, and $N_7 : [---PN]$ can be combined without any conflict to yield an output phase assignment of $[--PPN]$. This does not require duplication of fanin cones of N_4, N_5, and N_7.. Thus the corresponding incompatibility graph nodes 4, 5, and 7 are assigned to the non-duplicated bisection in Figure 5. The remaining candidate nets (i.e., graph nodes) that must be assigned to duplicated logic bisection for minimum duplication penalty can be obtained by representing the incompatibility constraints using boolean clauses as follows.

2.3.2 Boolean Constraint Satisfaction

We associate a boolean variable x_i with every node i in the graph. The boolean assignments to variable x_i are defined as follows. An $TRUE$ assignment to x_i implies that the fanin cone of candidate net N_i corresponding to node i is chosen for logic duplication and $x_i := FALSE$ implies that the fanin cone of candidate net N_i corresponding to node i is *not* chosen for logic duplication. The incompatible edge between two graph nodes i and j can represented by a simple two literal unate boolean clause $(x_i + x_j)$ which implies that at least one variable among x_i and x_j must be $TRUE$ to satisfy this clause. This is equivalent to duplicating the fanin cone of at least one corresponding candidate net N_i and N_j thereby satisfying their incompatibility relationship. The boolean constraint formula derived from the incompatibility graph will represent a simple unate 2-SAT formula. The fanin logic cone of every candidate net N_i whose corresponding boolean variable x_i receives a $FALSE$ assignment will not be duplicated. Thus the

[3] If the output phase assignments corresponding to nets N_i and N_j are incompatible and the compliment of net N_i's output phase assignment is compatible with net N_j's output phase assignment or vice-versa, then it is essential to consider the compliment phase assignments of N_i and N_j. A complete description of this complimentary phase assignment framework is given in [7].

corresponding node, node i in the incompatibility graph, is assigned to the non-duplicated bisection.

In Figure 5 the incompatibility graph can be represented by a simple boolean constraint formula F given as : $F = (x_1 + x_2)(x_1 + x_6)(x_2 + x_3)$. Since our goal is to minimize the logic duplication penalty, a satisfiable assignment that results in minimum logic duplication is chosen. Subsequently, every candidate net N_i corresponding to boolean variable x_i with $FALSE$ value is assigned to non-duplicated logic bisection in the incompatibility graph. For example, if assignment $x_1 = 1, x_2 = 1, x_3 = 0, x_6 = 0$ is chosen, then incompatibility graph nodes 3 and 6 corresponding to $FALSE$ boolean variables x_3 and x_6 respectively are assigned to the non-duplicated logic bisection (shown in Figure 5). The phase assignment of candidate nets corresponding to nodes in the non-duplicated logic bisection of incompatibility graph can be combined to yield the optimal solution. This implies that the phase assignments of candidate nets N_3, N_4, N_5, N_6, and N_7 (corresponding to non-duplicated bisection nodes $3, 4, 5, 6$, and 7 is for an optimal output phase assignment of $[PNPPN]$. The process of chosing an optimal boolean assignment among various satisfiable assignments is better explained using the example of Figure 4 as follows.

Consider the logic network shown in Figure 4. As already discussed in Section 2.2, this logic network has only two candidate nets N_1 and N_2 and they are incompatible. This can be formulated by the constraint formula $F = (x_1 + x_2)$. Since fanin cone of candidate net N_1 requires less duplication penalty (i.e., gates G_3 and G_7) than the fanin logic cone of candidate net N_2 (i.e., gates $G_4, G_5, G_6, G_9, G_{10}$, and G_{11}), boolean assignment $\{x_1 = 1, x_2 = 0\}$ is chosen as the final solution which implies an output phase assignment of $[N, P]$, i.e., primary output O_1 is implemented with a negative phase and primary output O_2 is implemented with a positive phase.

A *heuristic solution* to the output phase assignment problem for minimum logic duplication can be obtained as follows. Assigning a $TRUE$ value to a variable x_i that satisfies maximum number of clauses in the incompatibility constraint formula F. Remove these satisfied clauses from the constraint formula. Iteratively repeat these steps until all the clauses in the constraint formula F are satisfied. In the logic network, fanin cones of candidate nets corresponding to boolean variables with $TRUE$ value must be duplicated since they will have trapped inverters. Every candidate net N_i corresponding to boolean variable x_i with $FALSE$ value is assigned to non-duplicated logic bisection in the incompatibility graph. The phase assignment of primary outputs can be obtained by combining the phase assignments corresponding to the non-duplicated candidate nets, (i.e., the nodes in the non-duplicated logic bisection of incompatibility graph).

An *optimal solution* to the minimum logic duplication problem can be obtained as follows. Derive a binary decision diagram (BDD) from the incompatibility constraint formula F, which succinctly represents all of its possible solutions. In this BDD, every path from the root node to the constant 1 node represents a satisfiable assignment to the boolean constraint formula F. The area duplication penalty of this satisfiable assignment can be calculated by adding the technology independent area of all the gates in the fanin cone of every net N_i corresponding to every variable x_i with $TRUE$ assignment. With the duplication area penalty of the heuristic solution (obtained as described above) as the initial upper bound, traverse the BDD using a simple branch and bound algorithm [3]. During traversal, if the partial variable assignment at any BDD node (the BDD path from the root node) yields more logic duplication penalty than the upper bound, then prune that node and backtrack. The upper bound is updated whenever a complete satisfiable assignment solution with less logic duplication penalty is obtained. The branch and bound algorithm will yield an optimal satisfiable assignment to the constraint formula F, which corresponds to the minimum logic duplication penalty. This satisfiable assignment can be translated into the optimal output phase assignment solution as described above in the case of a heuristic solution.

After finding an output phase assignment as described above, two inverters are inserted at every primary output with a negative phase assignment. one of the two inverters for every primary output is pushed back all the way to the primary inputs of the logic network. For example, in the logic network shown in Figure 4(a), we obtained an optimal output phase assignment $[NP]$. Thus two inverters are inserted at output O_1 which received a negative phase assignment. One of these inverters is pushed back towards the primary inputs, thereby eliminating the trapped inverter INV_1 and requiring the duplication of optimizable logic region gates G_3 and G_7 in the optimizable logic region. Thus, to make the logic inverter-free with output phase assignment, duplication of gates $G_3, G_7, G_8, G_{12}, G_{13}$, and G_{14} is required. In comparison, without any output phase assignment, the logic duplication of gates $G_6, G_5, G_4, G_9, G_{10}, G_{11}, G_8, G_{12}, G_{13}$, and G_{14} would have been essential to push inverters INV_1 and INV_2 to primary inputs.

In some dynamic logic synthesis scenarios (e.g., when designers have the flexibility of absorbing inverters in mid-cycle latches), it may be required to achieve a different design goal of minimizing the number of inverters trapped at the intermediate fanouts. Inverter minimization is a well researched problem in static logic synthesis [4][5]. In dynamic logic circuits, the output phase assignment problem for obtaining minimum number of inverters was solved using a branch and bound algorithm in [2].

Table 1: Experimental results.

Designs					Output Phase Assignment Results			
Name	No. of primary inputs	No. of primary outputs	Gate count	No. of fanout nets	Area in Icells[4] when all the trapped inverters are pushed to primary inputs (all outputs have positive polarity)	Area in Icells after output phase optimization to minimize logic duplication	% reduction in area due to output phase assignment	No. of negative polarity outputs
ex1	89	151	4215	743	9167	8346	8.7%	4
ex2	51	28	907	166	3773	2281	39.5%	6
ex3	41	53	905	168	3422	3422	–	–
ex4	27	36	181	46	540	410	24.0%	10
ex5	32	6	451	93	1184	1184	–	–

The problem of obtaining minimum number of inverters is a special case of the more general minimal duplication problem discussed in this paper. The minimum number of inverters correspond to maximum number of nodes in the non-duplicated bisection of the incompatibility graph, i.e., minimum number of trapped inverters. This problem can be reduced to a simple case of the general unate covering problem on a unate 2SAT (only two literal clauses) boolean formula as follows.

Find a satisfiable assignment to the unate 2SAT incompatibility constraint formula F such that the number of variables (x_j) with FALSE assignment are maximum.

A solution to this simplified 2SAT unate covering problem will yield minimum number of trapped inverters at intermediate fanout nets in the logic network.

3 Experimental Results

This algorithm has been implemented within the Booledozer synthesis framework. Experimental results using the BDD algorithm on our internal designs, i.e., microprocessor design partitions are given in Table 1. Since inverters may get trapped at fanout nets in the logic network, the number of fanout nets in a design gives a measure of output phase assignment problem complexity. The output phase assignment is very effective in minimizing logic duplication penalty in the designs having non-reconvergent fanouts with trapped inverters close to primary outputs. Table 1 shows that in some designs significant area savings can be obtained, e.g., $ex2$ and $ex4$. The computing time required for the logic duplication optimization stage was negligible compared to the dynamic logic technology mapping and timing optimization stages in the dynamic synthesis scenarios. As discussed in Section 2.1, the fanin logic cones of reconvergent nets with a trapped inverters must be duplicated and the inverters trapped in the remaining logic (i.e., the optimizable logic region) may be eliminated by an output phase assignment to minimize logic duplication. In many designs, the optimizable logic region does not contain any trapped inverters. Such designs do not offer any chance of minimizing logic duplication penalty. Our experiments revealed that our designs $ex3$, $ex5$, and some public benchmarks (e.g., C880, C3540, C7552 etc.) belonged to such a category.

[4] An *Icell* is a basic unit of area.

4 Conclusion

In this paper, we addressed the problem of output phase assignment for minimum logic duplication in inverter-free dynamic logic synthesis. The problem was formulated in graph theoretic terms and both optimal and heuristic algorithms were presented. Experimental results indicate that in some cases substantial area savings can be obtained.

Acknowledgements

We would like to thank Daniel Brand and David Hathaway for valuable discussions. Reinaldo Bergamaschi, David Kung, Lakshi Reddy, Leon Stok, and Louise Trevillyan provided many useful comments.

References

[1] R. K. Brayton, C. L. Chen, C. T. McMullen, R. H. J. M. Otten, and Y. J. Yamour. Automated Implementation of Switching Functions as Dynamic CMOS Circuits. In *IEEE Custom Integrated Circuits Conference*, pages 346–350, 1984.

[2] R. K. Brayton, C. L. Chen, and Y. J. Yamour. Method for Optimizing Logic for Single-Ended Domino Logic Circuits. *IBM Technical Disclosure BulletinYO883-0364*, 27(7B):4398–4401, December 1984.

[3] S. Even. *Graph Algorithms*. Comp. Sc. Press, 1979.

[4] J. W. Goetz and D. J. Hathaway. Method and Apparatus for Optimizing a Logic Network. *US Patent number 5,282,147*, January 1994.

[5] A. Jain and R. E. Bryant. Inverter Minimization in Multi-Level Logic Networks. In *IEEE/ACM International Conference on CAD*, pages 462–465, 1993.

[6] R. H. Krambeck, C. M. Lee, and H. S. Law. High Speed Compact Circuits with CMOS. *IEEE Journal of Solid State Cicuits*, SC-17(3):614–619, June 1982.

[7] R. Puri, A. Bjorksten, and T. E. Rosser. Logic Optimization by Output Phase Assignment in Dynamic Logic Synthesis. *IBM Research Report*, April 1996.

[8] S. M. Reddy. Complete Test Sets for Logic Functions. *IEEE Transaction on Computers*, C-22(11):1016–1020, November 1973.

[9] T. Sasao. Input Variable Assignment and Output Phase Optimization of PLAs. *IEEE Transactions on Computers*, 33(10):879–894, October 1984.

[10] C. L. Wey and T. Y. Chang. An Efficient Output Phase Assignment for PLA Minimization. *IEEE Transactions on CAD*, 9(1):1–7, January 1990.