Post-Layout Logic Duplication for Synthesis of Domino Circuits with Complex Gates

Aiqun Cao
Synopsys, Inc.
Mountain View, CA
aiqun@synopsys.com

Ruibing Lu
Synopsys, Inc.
Mountain View, CA
ruibing@synopsys.com

Cheng-Kok Koh
Purdue University
West Lafayette, IN
chengkok@ecn.purdue.edu

ABSTRACT

Logic duplication to resolve the logic reconvergent paths problem encountered in Domino logic synthesis is expensive in terms of area and power. In this paper, we propose a combined logic duplication minimization and technology mapping scheme for Domino circuits with complex gates. The logic duplication is performed as a post-layout step as the duplication cost is minimized based on accurate timing information. Experimental results show significant improvements in area, power, and delay.

1. INTRODUCTION

Domino logic has been extensively used in circuitry that demands high speed. In order to implement a binate logic network by a Domino circuit, which can only implement non-inverting logic, the binate logic network has to be transformed into a unate one by pushing inverters (bubble pushing) towards primary inputs. However, due to the existence of logic reconvergent paths in the logic network, some inverters are inevitably trapped within the reconvergent paths during the bubble pushing. Moreover, the inverters may also be trapped at other intermediate fan-out nets due to inappropriate output phase assignment.

A common approach to eliminate the trapped inverters is to duplicate the fan-in cones of the intermediate nets where the inverters are trapped so that the trapped inverters can be pushed to the primary inputs. The logic duplication cost can be as expensive as duplicating the whole logic network.

In [6, 8], output phase assignment algorithms were proposed to minimize the logic duplication cost incurred by the elimination of inverters trapped *outside* logic reconvergent paths. The inverters trapped *inside* the reconvergent paths were treated as non-optimizable and logic duplication due to these inverters was imperative.

To reduce the duplication cost caused by the inverters trapped *inside* the logic reconvergent, a mixed static-Domino circuit with the static logic circuit trailing the Domino logic circuit was proposed in [3]. The transitive fan-outs of the inverters were synthesized into static logic so that the inverters were allowed inside the reconvergent paths. In [2], the trapped inverters were allowed inside the reconvergent paths if the fan-out gate of the inverter is an AND gate, whose inputs obey a certain timing constraint [5], called the *Early-Late Delay Difference Bound (ELDDB)* constraint in [2]. The minimization of logic duplication, which relied on accurate timing information, was performed as a post-layout step. Neither [3] nor [2] considered the inverters trapped *outside* the reconvergent paths.

In [3, 6, 2, 8], simple Domino gates were considered. Complex Domino gates, which were more desirable because of their high performance and low area overhead, were considered in [4, 7]. However, the starting point for the technology mapping approaches proposed in [4, 7] was a unate network of two-input AND-OR gates. In other words, logic duplication and technology mapping were performed independently in [4, 7].

In this paper, we combine technology mapping with logic duplication minimization; the logic network is mapped into complex Domino gates in such a way that the duplication cost is minimized. We also minimize the logic duplication cost due to inverters *inside* and *outside* the reconvergent paths. This can reduce the duplication cost more substantially than all previous works.

As the logic duplication cost is minimized based on a more general ELDDB constraint, we perform the logic duplication as a post-layout step when accurate timing information is available. Experimental results show that further reductions on the area, power, and delay can be achieved when compared with [2].

Our contributions in this paper are the following:

1. We combine the technology mapping to complex Domino gates step with logic duplication minimization.
2. We optimize the logic duplication cost by considering inverters trapped *inside* and *outside* the reconvergent paths at the same time.
3. A new post-layout expansion algorithm is proposed to to perform minimal logic duplication to further reduce the area, power, and power of the circuits. Runtime is also reduced substantially.

2. TECHNOLOGY MAPPING AND DUPLICATION MINIMIZATION FOR COMPLEX DOMINO GATES

ASPDAC 2005, January 18-21, 2005, Shanghai, China.

In this section, we extend the ELDDB constraint in [2] to complex Domino gates. Then we combine the technology mapping and logic duplication minimization together for the synthesis of Domino circuits with complex gates. The logic duplication is minimized by generating complex candidate gates through technology mapping. Mapping to complex candidate gates is different than mapping to other complex gates. Moreover, we consider the inverters trapped both inside and outside the reconvergent paths in order to minimize the logic duplication. Note that our primary objective is to minimize the layout area. Consequently, the power of the circuits can also be optimized. As a result of complex gates mapping, the logic depth is reduced, resulting in better delay.

2.1 Definition

We give some definitions first. Given a pair of reconvergent paths with an inverter trapped inside (see Fig. 1(a)), the two paths are called the two branches of the reconvergent paths. The common starting node of the two branches is called the divergent node (node D in Fig. 1(a)) and the common ending node the convergent node (node C in Fig. 1(a)). All other nodes in the reconvergent paths are intermediate nodes. The fan-in cone of the reconvergent paths consists of the fan-in cone of the divergent node and the divergent node itself.

2.2 ELDDB constraint for complex Domino gates

[2] proposed the ELDDB constraint for 2-input AND Domino gate in order to allow a trapped inverter to be an input to the 2-input AND gate. It requires that the inverting input (output of the inverter) always arrives earlier than the non-inverting input of the AND gate. Here, we generalize the ELDDB constraint to complex Domino gates. In general, every input to a complex gate has a timing window in which its transition may take place in the evaluation phase. Fig. 1(c) for example, shows the timing window of $\overline{D}$, U, and Y. Each timing window is defined by an *early edge* (or the earliest transition) and a *late edge* (or the latest transition). We use $D^l_{1\to 0}$ to denote the *latest late edge* among all inverting inputs (making a 1→0 transition) along a path from the precharged dynamic node (node P in Fig. 1(b)) to the ground. Similarly, we use $D^e_{0\to 1}$ to denote the *latest early edge* among all non-inverting inputs (making a 0→1 transition) along the same path. Then, the false discharge of the dynamic node of a complex gate due to an inverting input making a 1→0 transition can be avoided if Eqn. (1) holds for all paths from the dynamic node to the ground:

$$D^e_{0\to 1} - D^l_{1\to 0} \geq B, \tag{1}$$

where $B > 0$ is the delay difference bound. B was called the *Early-Late Delay Difference Bound* (ELDDB) in [2]. In other words, a Domino gate can operate correctly as long as the latest input transition in each pull-down path is a 0→1 transition. A complex gate that can accept an input that makes a 1→0 transition is a *candidate gate*.

2.3 Candidate primitive

In order to minimize logic duplication by the use of complex candidate gates, we have to take mapping into consideration when we perform logic duplication minimization. The starting point of our approach is an optimized logic network that is decomposed into a two-input NAND-NOR directed acyclic graph (DAG). For Domino circuit synthesis, the NAND-NOR DAG is typically transformed into an AND-OR DAG with bubble pushing. Here, we transform the NAND-NOR DAG into an "AND-OR-INV" DAG. During the bubble pushing process, some inverters are allowed to be trapped inside certain reconvergent paths if candidate gates can be found along the reconvergent paths. We refer to the AND/OR gates as primitives.

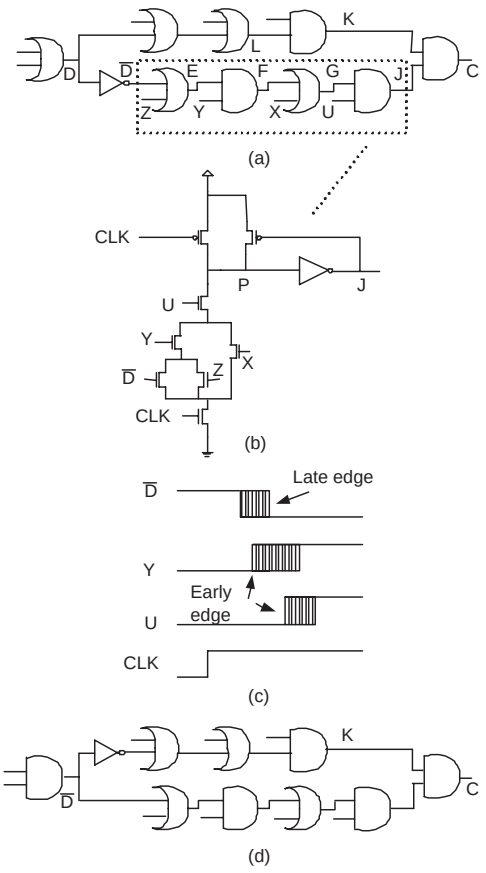

Figure 1: ELDDB constraint of complex candidate gate.

For each intermediate primitive along a reconvergent branch, we refer to the fan-in that is along the branch as an internal fan-in. The other fan-in (if there is one) is called an external fan-in of the intermediate node. Note that the two inputs of gate C in Fig. 1(a) are internal fan-ins as well as external fan-ins, depending on which reconvergent branch is being considered.

Consider the mapping of the bottom branch of the reconvergent path in Fig. 1(a) into a complex gate. As we cannot map a trapped inverter into a complex Domino gate, the output of the inverter can only be an input to a complex gate. The mapped complex gate M is shown in Fig. 1(b). For the complex gate M to be a candidate gate, either U or Y has to make a 0→1 transition later than $\overline{D}$ making a 1→0 transition. U and Y are the external fan-ins of the intermediate primitives J and F, respectively.

In order to find a complex candidate gate, we have to find a *candidate primitive* in the subgraph of the "AND-

OR-INV" DAG that coalesces into a candidate gate. A candidate primitive should have the following attributes:

1. It must be an intermediate AND gate in reconvergent paths whose transitive fan-ins include the trapped inverter.

2. The delay difference between its external fan-in and the output of the inverter is always equal to or larger than B, the ELDDB.

J and F in Fig. 1(a) are candidate primitives if they satisfy the second attribute. We may find other candidate primitive if the trapped inverter can be relocated within the reconvergent paths by bubble pushing. In Figure 1(d), for example, if the fan-ins of K satisfy the ELDDB constraint, K is also a candidate primitive after relocating the trapped inverter.

For simplicity, we assume that the trapped inverter in the reconvergent paths is at the fan-out of the divergent node as that provides the earliest possible late edge for a $1 \rightarrow 0$ transition made by the inverter. We also consider only one inverting input for each candidate gate.

As we need only one candidate primitive to be mapped into each candidate gate, when there are more than one candidate primitive in a pair of reconvergent paths, we always pick the one that provides the safest margin. In other words, we pick the one whose external fan-in is the last to arrive. Note that some reconvergent paths may not contain a candidate primitive no matter where the inverter is relocated. The fan-in cones of these reconvergent paths have to be duplicated later in order to eliminate the trapped inverters.

2.4 Logic duplication minimization

Suppose we have the timing information at the logic level. Although the candidate primitives for each reconvergent paths can be identified independently, different reconvergent paths may require different output phase assignments in order to obtain their candidate primitives during the bubble pushing. Fig. 2 shows an example of that. $C1$ and $C2$ are two convergent nodes of different reconvergent paths and $PO1$ and $PO2$ are primary outputs. Two candidate primitives in these two different reconvergent paths require conflicting phase assignments of $PO1$ and $PO2$. Therefore, these two reconvergent paths are *incompatible* with each other and one of them has to duplicate its fan-in cone. Obviously, we pick the one whose fan-in cone is smaller to duplicate.

The problem of logic duplication minimization can be formulated as that of finding the least weight vertex cover of an incompatibility graph $G(V, E)$ as in [2]. We construct G by representing each pair of reconvergent paths with a vertex in V whose weight equals the corresponding duplication cost. An edge in E connects two vertices if the represented two pairs of reconvergent paths are incompatible to each other. After that, the same branch and bound algorithm as in [2] is applied.

Thus far, we have been concerned with inverters trapped inside reconvergent paths. However, there may be inverters trapped outside reconvergent paths. An example of a sub-network with a trapped inverter outside reconvergent paths is shown in Fig. 3. We refer to such a sub-network as pseudo reconvergent paths, and $D1$ and $D2$ pseudo divergent nodes. Note that no matter how different phases are assigned to $PO1$ and $PO2$, the inverter remains trapped. Therefore, the

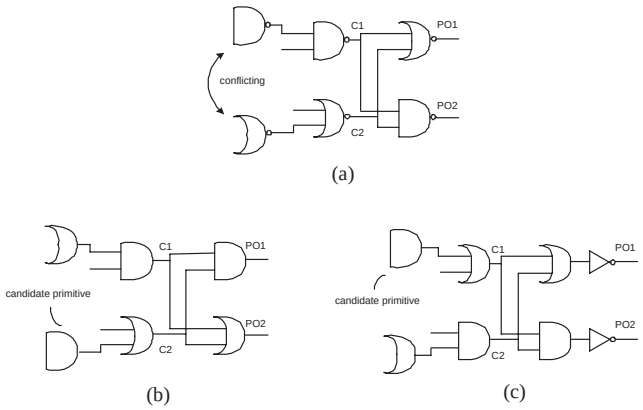

Figure 2: Conflicts among candidate primitives in different reconvergent paths.

approach proposed in [6] and [8] to deal with the inverters trapped outside the reconvergent paths cannot eliminate the inverter here without duplicating $D1$'s or $D2$'s fan-in cone.

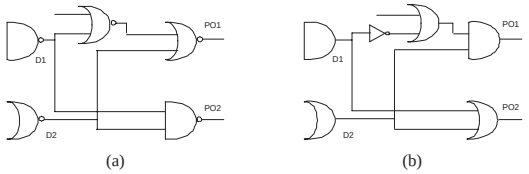

Figure 3: Pseudo reconvergent paths.

We observe that the pseudo reconvergent paths can be treated as the regular reconvergent paths. The trapped inverter can be relocated to be a direct fan-out of either of the pseudo divergent nodes. Then we have reduced the problem to that logic duplication minimization for regular reconvergent paths. Therefore, in order to minimize the logic duplication cost, the least cost vertex cover formulation can be extended by using an incompatibility graph that includes both reconvergent paths and pseudo reconvergent paths.

We use Elmore delay model to get estimated delay information. We obtain the optimal phase assignment of the primary outputs and identify the candidate primitives simultaneously. We do not duplicate the fan-in cones of the reconvergent paths and pseudo reconvergent paths that have candidate primitives. For the rest, we perform logic duplication of their fan-in cones to remove the trapped inverters. The resulting "AND-OR-INV" DAG is then partitioned into trees and the technology mapping is applied.

2.5 Mapping to complex candidate gates

We partition the DAG into a forest of trees whose roots are primary outputs or divergent nodes (including pseudo ones). The DAG is not broken at multi-fanout nodes as that would result in very small trees, in which case the advantages of having complex Domino gates would be negated. Moreover, this partitioning method allows multi-fanout nodes to be mapped into multi-output Domino gates. We apply the similar dynamic programming mapping approach as in [7], but deal with the mapping of the reconvergent paths with trapped inverters in a different fashion. [7] have reconvergent paths but no inverters trapped inside; hence, the recon-

vergent paths were broken at the divergent nodes and were treated the same as the rest.

As we can see in Fig. 1(a), a trapped inverter does not have to be one of the direct fan-ins of the candidate primitive. We have to make sure that the candidate primitive is mapped into the same complex gate one of whose inputs is the output of the inverter. Assume that the limits on the width and height a complex Domino gate are W and H. We additionally impose the following constraints on the candidate primitives in order to facilitate the dynamic programming based mapping:

1. It has to be within the width and height limits W and H from the output of the inverter.
2. The transitive fan-ins of the candidate primitive along the branch of the reconvergent paths do not include any multi-fanout nodes.

During the bottom-up fashion dynamic programming mapping approach, any optimal sub-solutions that do not map the candidate primitive into the candidate gate will be discarded. Note that the objective is to minimize the total number of transistors. As the optimal solutions of the dynamic programming tend to map as many primitives into a complex gate as possible, the disregard of mapping fewer number of primitives into a complex gate is unlikely to affect the optimality of the dynamic programming.

3. POST-LAYOUT OPTIMIZATION

As no accurate timing information can be obtained, the real candidates that satisfy the ELDDB constraints cannot be determined at the logic level. Therefore, the circuit cannot be finalized at the logic level if we want to apply the techniques presented in the previous section. Therefore, the logic duplication is performed as a post-layout step and a post-layout optimization scheme is presented, as shown in Fig. 4. At the logic level, as a preprocessing step, we use estimated delay values to predict the candidate primitives (see Section 2.4). The synthesis scheme proposed in Section 2.4 is applied to produce maximum (weighted) candidate primitives. The dynamic programming mapping approach presented in Section 2.5 is then applied to map candidate primitives into candidate gates. Accurate timing information obtained after post-layout simulation will be used to verify the ELDDB constraints of the candidate gates.

In this section, we propose a new post-layout technique for logic duplication minimization. In contrast to [2], which started from a circuit with redundant fan-in cones for all reconvergent paths with trapped inverters, and iteratively removed them and performed layout compaction, we start from a minimal circuit that is possibly non-functional and perform logic duplication to make it functional. Hence, we expand the layout instead of compacting it. At the logic level, we do not perform duplication for the candidates that are obtained based on the estimated timing information. After we obtain the layout of the minimal circuit, we perform timing analysis on those candidates to see if they truly satisfy the timing constraints. If certain candidates do not, we have to duplicate the corresponding fan-in cones by inserting appropriate cells into the layout.

In contrast to [2] where redundant fan-in cones have to be eliminated one at a time, here we can perform a batch insertion of fan-in cones that correspond to candidate gates

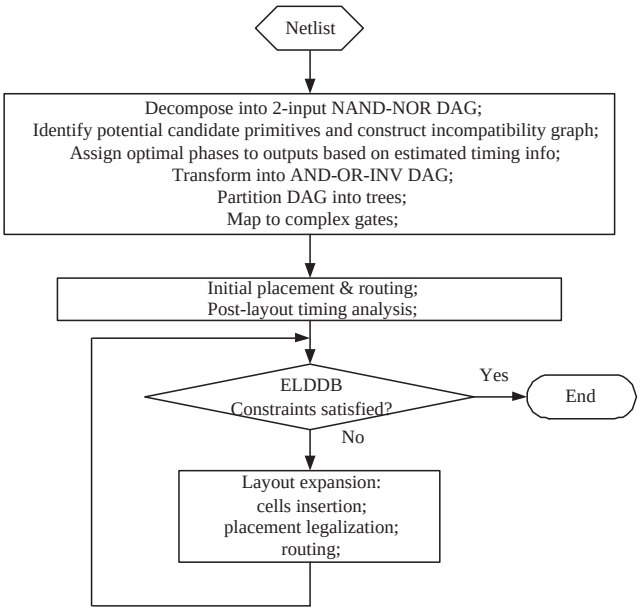

Figure 4: Algorithm flow.

whose timing constraints are not satisfied. The layout is expanded to accommodate these additional cells. However, it is possible that new timing violations may emerge after the insertion of fan-in cones and layout expansion. Therefore, the batch insertion may have to be performed iteratively. The whole flow is shown Fig. 4.

3.1 Layout expansion

In the layout expansion step, we strive to minimize the effects of layout expansion on the delays of most signals, especially those critical to meeting the timing constraints.

The details are as follows: Consider a candidate that violates the timing constraint. Let Δ denote the original fan-in cone of the reconvergent paths containing that candidate. Δ consists of complex gate cells $C_i (1 \leq i \leq n)$. We denote the location of cell C_i by its center (x_i, y_i) in the original layout. Let Δ' be the duplicated fan-in cone, which has the same number of cells as Δ, that will be inserted. Each cell C_i' in Δ' is the *dual* of the cell C_i in Δ; a parallel connection of transistors in the pull-down network of C_i corresponds to a series connection of the transistors in the pull-down network of C_i', and vice versa.

In order to minimize the changes in the delays of most signals, we observe that the best way to insert Δ' is to place each cell C_i' in Δ' on the location of its dual cell C_i in Δ, if we allow overlapping of cells in the layout. The two branches of the reconvergent paths that were both driven by Δ are now driven by Δ and Δ' separately. By placing C_i' on the location of C_i, the wire length of the nets along the reconvergent paths remains unchanged. Therefore, it is likely that the delays of signals along the reconvergent paths as well as the delays of the remaining signals change little.

However, such a placement is illegal; the layout has to be expanded in order to remove the overlapping of cells. As long as the topology of the circuit, i.e., the relative positions of cells remain unchanged after layout expansion, the timing constraints for the remaining candidates are likely to remain satisfied for the following reason. Although the

expansion of the layout causes longer wire length and longer signal delays, the delays are likely to increase proportionally. Hence, the ELDDB constraints are likely to remain met for the remaining candidates.

Assume that the area of the original layout L is A and the sum of the area of the cells inserted is A_{in}. The new layout L' has area $A' = A + A_{in}$ after expansion. Here, we assume that the aspect ratios of L and L' are the same.

The layout expansion step starts with using a slicing tree structure (for slicing floorplan) to encode the relative positions of the cells in the layout. To construct the slicing tree T, we first cut/bisect L into two partitions. Assume that the first cut direction is vertical. All cells whose centers are in the left partition form the left child of the root of the slicing tree and those in the right partition form the right child. We recursively make cuts on the left and right children and grow T until each leaf of T contains only a few cells. The cut directions are chosen based on the aspect ratio of each cut region. Next, we insert the duplicated cells into T based on the locations of their dual cells.

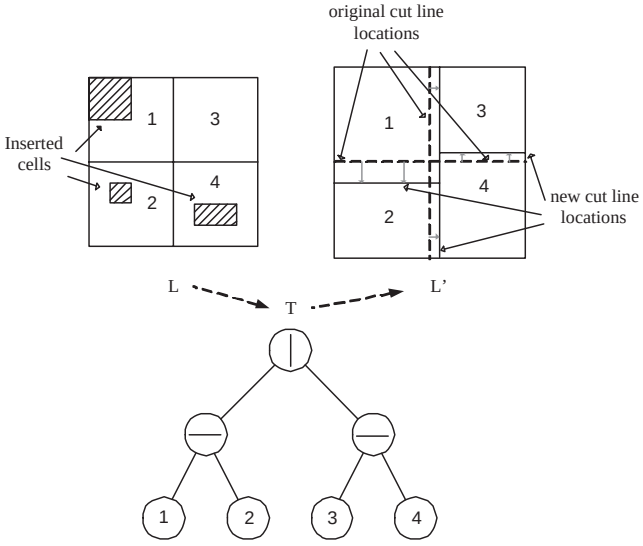

Figure 5: Illustration of layout expansion.

After that, we use this updated slicing tree T to place the original and new cells into L'. Consider the left and right children of the root of T. Although the cut was a bisector of L, it should not be a bisector of L' as the total cell area of the left child A'_l is not equal to that of the right child A'_r ($A'_l + A'_r = A'$) due to the inserted cells. Therefore, we adjust the position of the vertical cut in L' such that the area of the left partition is equal to A'_l. The positions of cut lines in the descendants of A'_l and A'_r are similarly adjusted through a top-down traversal of T. Effectively, each region in the slicing tree has an area that can accommodate the cells without overlapping. Fig. 5 gives a simple illustration of the expansion.

A dynamic programming based row-by-row legalization proposed in [1] is adopted to legalize the placement obtained after layout expansion. Here, the objective is to minimize the displacement for all cells after the placement legalization. Higher priority are given to cells whose timing behaviors are crucial to meeting the ELDDB constraints in order to minimize the displacement on these cells. The relative positions of the cells remain almost unchanged after the legalization step. After that, we perform routing, followed by parasitic extraction and timing analysis. If there are candidates that do not satisfy the timing constraints, the layout expansion step is repeated as shown in the flow in Fig. 4.

4. EXPERIMENTAL RESULTS

We implement the algorithms presented in Sections 2 and 3 in C++. The flow presented in Fig. 4 is run on ten ISCAS benchmark circuits. The benchmark circuits are first optimized with SIS before being fed into the flow as inputs. The experiments run on Sun Ultra 10 workstation with 256Megabytes memory. Initial placement and routing are performed by *QPLACE* and *WROUTE*. NanoSim and PathMill are used for post-layout simulations. A PERL script automates the flow with LEF/DEF data formats transferring layout data.

Virtuoso Layout Synthesizer is used to mimic an on-the-fly cell generator to generate the cell layouts for complex Domino gates under $0.35\mu m$ technology. The width and height limits W and H for the complex Domino gates are both set to 3. The ELDDB B is specified to be $2ns$.

The experimental results of running the flow in Fig. 4 are reported in Table 1 (listed along the row 'Complex-optimal-expansion'). For comparison, we perform a full logic duplication as in [4, 7], followed by the dynamic programming mapping approach (listed along the row 'Complex-duplication'). We can see that we reduce the area by 30% and power by 34% when compared with the full logic duplication approach.

To demonstrate the effectiveness of the presented techniques at logic level (Section 2) and physical level (Section 3) independently, we also run other sets of experiments. The first set of experiments has the optimal output phase assignment replaced by a random output phase assignment in the flow ('Complex-random-expansion'). Obviously, that would result in a fewer number of candidates and thus more logic duplication. As we can see, our optimal output phase assignment has 22% lower area and 18% lower power than the random output phase assignment.

To show the improvement due to the layout expansion algorithm, we use the iterative elimination and compaction ('IEC') algorithm in [2] ('Complex-optimal-IEC') to perform the post-layout logic optimization in the second set of experiments. First, the pre-layout logic optimization technique outlined in Section 2 is applied. Then, logic duplication is performed for all candidates so that 'IEC' can be applied. It shows that only one circuit (C6288) has area and power marginally worse when comparing 'Complex-optimal-expansion' with 'Complex-optimal-IEC'. The average improvements in area and power due to the layout expansion algorithm are 8% and 7%, respectively. Moreover, 'Complex-optimal-expansion' reduces the run time substantially to 38% of that of 'Complex-optimal-IEC' as the number of post-layout iterations is reduced due to batch insertion.

Although our flow does not attempt to optimize speed performance directly, the reduction in critical path delay of these benchmark circuit by our flow is quite impressive. 'Complex-optimal-expansion' reduces the critical path delay by 16%, 9%, and 4% when compared with 'Complex-duplication', 'Complex-random-expansion', and 'Complex-optimal-IEC', respectively. These improvements are achieved mainly because of the smaller logic depth and physical lay-

	Circuit	C432	C499	C880	C1355	C1908	C2670	C3540	C5315	C6288	C7552	ratio
Simple-duplication [2]	Area (x1000μm^2)	72.1	205.9	145.9	207.4	257	428.2	547.1	749.3	942.2	1190.5	1.19
	Power (mW)	40	102.4	93.2	112.7	114.7	186.3	225.4	258.8	271.3	396.1	1.13
	delay (ns)	18.8	15.6	15.1	16.9	21	18.8	30.4	27.9	31.9	25.2	1.32
Simple-IEC [2]	Area (x1000μm^2)	58.2	168.4	119.1	160.8	209.3	337	448.9	605	839.5	931.8	0.96
	Power (mW)	35	82.9	69.2	90.4	85.5	151.3	166.2	189.6	212.5	328.5	0.89
	delay (ns)	16.6	14	16	14.2	17	16.9	29.1	25.5	29.4	22.5	1.2
	runtime (m)	39	61	50	85	77	170	156	381	322	405	1.18
Simple-expansion	Area (x1000μm^2)	51.1	142.5	100.8	143.1	178.2	289.2	400.9	549.8	787.1	881.6	0.85
	Power (mW)	32.1	70.3	58.8	78.7	76.9	131.7	150	174.8	189.1	288.2	0.78
	delay (ns)	15.9	13.2	15	13.3	16.3	15.6	27.8	23.1	28.2	20.1	1.12
	run time (m)	25	34	28	38	29	52	60	167	244	298	0.6
Complex-duplication	Area (x1000μm^2)	60.5	174.4	135.9	177.1	216.4	349.1	416.5	636.1	849.7	946.8	1
	Power (mW)	36.8	87.8	80.9	100.5	102.3	168.7	197.2	221.3	242.9	359.1	1
	delay (ns)	11.4	10.7	11.4	12.3	16.7	15.1	23.9	21.9	27.5	20.8	1
Complex-random-expansion	Area (x1000μm^2)	50.3	154.4	113.6	168	207.8	346.3	398.3	584.8	766.6	890.7	0.92
	Power (mW)	33.2	75.2	65.5	86.2	84.5	136.8	160.4	180.8	201.5	302.3	0.84
	delay (ns)	10.8	10.1	10.9	11.4	15.9	14	22.6	19.1	25.6	18.9	0.93
Complex-optimal-IEC	Area (x1000μm^2)	43.9	132.8	98.8	149.5	170.7	281	316.3	477.4	688.5	760.5	0.78
	Power (mW)	30.7	65	52.5	70.9	71.1	124.1	139.2	165.6	173.8	269.4	0.73
	delay (ns)	10.3	9.7	10.1	10.9	15.2	13.3	20.9	18.7	24.1	18.1	0.88
	runtime (m)	34	48	40	66	61	128	145	297	414	360	1
Complex-optimal-expansion	Area (x1000μm^2)	38.5	111.2	82.6	132.3	166.5	234.2	311.3	420.6	716.7	686.3	0.70
	Power (mW)	28.2	59.6	45.6	63.4	67.2	105.8	133.2	145	181.1	228.2	0.66
	delay (ns)	10	9.1	9.8	10.2	13.8	12.1	21.4	17.5	25.1	16.4	0.84
	runtime (m)	16	21	18	26	20	44	48	96	127	159	0.38

Table 1: Comparisons of experimental results

out area. However, there are two circuits (C6288 and C3540) with slightly longer critical path delays when comparing 'Complex-optimal-expansion' with 'Complex-optimal-IEC'. We believe these results can be further strengthened if the layout expansion algorithm and the dynamic programming mapping algorithm are both extended to minimize delay directly.

The layout expansion algorithm is also applied on Domino circuits with simple gates ('Simple-expansion'). The results from [2] are also included. As we can see, the improvements of our approach on the Domino circuits with simple gates are even larger, validating the effectiveness of the new post-layout optimization approach. The results listed along the rows 'Simple-duplication' and 'Simple-IEC' correspond to the methods of applying full logic duplication and 'IEC' on simple gates, respectively.

For other previous work on Domino logic synthesis, [6] reported 14% area reduction and [8] had similar results. In [3], gate area was reduced by 25% whereas the delay increased by 3%. All these results were based on gate level implementations.

5. CONCLUSION

In this paper, we combine the technology mapping with logic duplication minimization for Domino circuits with complex gates. An optimal output phase assignment algorithm is proposed to minimize logic duplication due to inverters trapped in reconvergent paths and pseudo reconvergent paths. Moreover, we propose a post-layout expansion algorithm to perform minimal logic duplication. Experimental results show significant improvements in area, power, and delay of the Domino circuits.

6. REFERENCES

[1] A. Agnihotri, M. C. Yildiz, A. Khatkhate, A. Mathur, S. Ono, and P. H. Madden. Fractional cut: improved recursive bisection placement. In *Proc. Int. Conf. on Computer Aided Design*, pages 307–310, Nov. 2003.

[2] A. Cao and C.-K. Koh. Post-layout logic optimization of Domino circuits. In *Proc. Design Automation Conf*, pages 820–825, June 2004.

[3] K. W. Kim, T. Kim, C. L. Liu, and S. M. Kang. Domino logic synthesis based on implication graph. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 21(2):232–240, Feb. 2002.

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

[5] R. Puri. Design issues in mixed Static-Domino circuit implementations. In *Proc. IEEE Int. Conf. on Computer Design*, pages 270–275, Oct. 1998.

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

[7] M. Zhao and S. S. Sapatnekar. Technology mapping for domino logic. In *Proc. Int. Conf. on Computer Aided Design*, pages 248–251, Nov. 1998.

[8] M. Zhao and S. S. Sapatnekar. Dual-monotonic domino gate mapping and optimal output phase assignment of domino logic. In *Proc. IEEE Int. Symp. on Circuits and Systems*, pages 309–312, May 2000.