CRDOM: CELL RE-ORDERING BASED DOMINO ON-THE-FLY MAPPING

Sai Praveen Kadiyala and Debasis Samanta

School of Information Technology, Indian Institute of Technology, Kharagpur, India

ABSTRACT

This Domino logic is often the choice for designing high speed CMOS circuits. Often VLSI designers choose library based approaches to perform technology mapping of large scale circuits involving static CMOS logic style. Cells designed using Domino logic style have the flexibility to accommodate wide range of functions in them. Hence, there is a scope to adopt a library free synthesis approach for circuits designed using Domino logic. In this work, we present an approach for mapping a domino logic circuit using an On-the-fly technique. First, we present a node mapping algorithm which maps a given Domino logic netlist using On-the-fly technique. Next, using an Equivalence Table, we re-order the cells along the critical path for delay, area benefit. Finally, we find an optimum re-ordering set which can obtain maximum delay savings for a minimum area penalty. We have tested the efficacy of our approach with a set of standard benchmark circuits. Our proposed mapping approach (CRDOM) obtained 21% improvement in area and 17% improvement in delay compared to existing work.

KEYWORDS

Domino logic, on the fly mapping, library free synthesis, cell re-ordering, critical path

1. INTRODUCTION

Of late, Domino logic style is employed in designing various high performance circuits [1], [2], [3], [4], [5], [6], [7], [8]. Traditionally technology mapping procedures rely on predefined standard cell libraries [9], [10], [11], [12]. Designers even choose to go for cell generator based techniques for mapping complex static gates [13], [14], [15]. Significant attempts are made to decompose a circuit into forest of trees at single fanout points, there by following a tree by tree mapping approach [16], [17]. However, these approaches are not beneficial for circuits involving Domino gates [18], [19]. The single fanout trees will have additional clock transistors which will nullify the advantages offered by Domino cells [20], [21]. The ability of Domino cells to realize complex functionalities and multi outputs in a single cell yields advantages in terms of area and delay [22]. Hence there is a necessity to synthesize circuits using large Domino cells such that resulting circuit has less area and yields high performance.

Early attempts using library free mapping are mentioned in works [9], [22]. These works used a parameterized library mapping which imposes constraints on height and width of the Domino cells. Work [9], mentioned an approach based on formation of complex candidate gate. This approach tried to perform mapping minimizing logic duplication. A mapping technique focused

DOI : 10.5121/vlsic.2014.5402

on minimizing logical effort is presented in [14]. This approach tried to reduce the delay of the overall circuit by reducing the number of series transistors along the critical path. A similar work is presented in [23], which tried to minimize the delay based on logic effort regulation. A combination of structural and Boolean matching is performed in this approach. In [16], a mapping approach for incompletely specified functions is presented. Here, the average case delay is minimized by computing lower and upper bounds of delay for the overall circuit. A virtual mapping technique for large circuits is presented in [19]. This work aimed at forming large gates by grouping without any constraints. Later splitting of the gates is done by minimizing gate count and stages of the gate. An enhanced technology mapping algorithm is proposed in [20], which focused on dividing the circuit into k-input, l-output cuts. A transistor reordering based mapping approach is suggested in [21], which is applied to SOI Domino logic addressing bipolar effects. This approach includes adding extra PMOS discharge transistors. Work [24] presents a bin packaging algorithm for library free mapping. This approach simplified the level and cell complexity. A multilevel cell based mapping approach is proposed in [25]. It aimed at decomposing large circuits into not more than 15 variables and performs a kernel based multi level mapping. A technique for area oriented library free mapping, taking full advantage of complex gates is presented in [26]. It involved various steps like pre decomposition, gate assignment and gate building.

Parameterized library mapping mentioned in works [9], [22], did not emphasize the basis for choosing the height and width constraints of the Domino cell. Also the bubble pushing algorithm used for converting the given binate circuit into unate, is not area efficient and leads to huge logic duplication. A significant amount of logic duplication arises in work [13]. This hampers the advantages of forming complex candidate gates. Though delay minimization based on logical effort is presented in [14], [23], they focused on circuits in general and gave little emphasis on Domino logic based circuits. The technology mapping procedure for incompletely specified circuits mentioned in [16] considers nand based DAG covers as the basis. Since Domino logic realizes only non inverting logic this approach cannot be valid. Works [17],[18] and [19] which consider formation of complex candidate gates for mapping domino cells, have not emphasized on managing the critical path of the circuit. Also light was not thrown on the levels of cell to be designed. Approach mentioned in [20] has not addressed the problem at transistor level. Work mentioned in [21], can be applied to only SOI devices and not suitable for Domino circuits in general. Mapping techniques presented in [24] and [26] though emphasized on minimizing the cell complexity by reducing the levels, there was hardly any focus on minimizing overall circuit delay.

Though many works are reported in the literature on library free mapping most of them begin with a nand based DAG network. None of the literature considered unate circuits as a base for their approaches. The works which adopted a parameterized library mapping did not focus on managing critical path. Hence, there is a necessity for designing a mapping technique which takes care of realizing large functionalities in a single cell and simultaneously fine-tunes cells along critical path for obtaining high performance. The flexibility offered by Domino logic style in designing the individual cells motivates us to dig in this direction. Also there is a scope for
re-ordering the cells along critical path, thus formed during the mapping which can further minimize delay. Fine tuning these cells along the critical path, without increasing their individual transistor count is a challenging task.

Keeping the above scope in view, in our work we try to present a Cell Re-ordering based Domino On-the-fly Mapping (CRDOM) approach. First we convert given unate network into a netlist of large Domino cells using a node mapping algorithm. Next, we try to re-order the cells along critical path thus formed, which can further minimize delay. Finally, we choose an optimum cell re-ordering set for the critical path, where delay and area penalty are minimized by using a two objective optimization problem. How we resolve the issues that arise in cell mapping, cell re-ordering and finding optimum cell re-ordering set are presented in this work.

The rest of the paper is organized as follows. Section 2 describes some basic definitions and derivations, establishes certain Lemmas that will be used further in our approach. Our proposed methodology for realizing Cell Re-ordering based Domino On-the-fly mapping is elaborated in Section 3. The setup, benchmarks we chose for conducting the experiments and the experimental results obtained along with a comparative analysis are presented in Section 4. Finally, Section 5 concludes the paper.

2. BASIC CONCEPTS

Our proposed methodology for designing Cell Re-ordering based Domino On-the-fly Mapping (CRDOM), involves dynamic mapping of cells (on-the-fly), constructing Equivalence Tables, cell re-ordering and Optimization of Critical Path. In the following we present some basic concepts which we use in our further discussion.

Node: A gate (cell) in a given circuit is often refereed as node in the graphical representation of the particular circuit. A gate equivalent of a node in a circuit graph is shown in Fig. 1.

Circuit Graph: It presents a graphical representation of Boolean circuit, where gates of the circuit form various nodes of the graph. In Fig. 1, a single node graph is shown.

Width of node: Width of a node is defined by the number of parallel transistor chains present in the node. For the node shown in Fig. 1, the width is 3.

Height of node: Height of a node is defined by the maximum number of serial transistors present in a single chain in the node. For the node shown in Fig. 1, the height is 3.

Leaf Node: The primary inputs to a given circuit form the leaf nodes in the graphical representation of the considered circuit.

Gate formation Node: Every node during the mapping must obey certain width and height constraints. If the mapping algorithm exceeds these constraints for a particular node, a Gate formation node occurs there and a new node is started from the next.

Cell Reordering: Delay of a particular node is dependent on the architecture of the node. Restructuring of the transistors in a particular cell (node), is often done to minimize its delay. This is called Cell Reordering.

Original Cell: The cell belonging to the circuit before re-ordering is called the original cell.

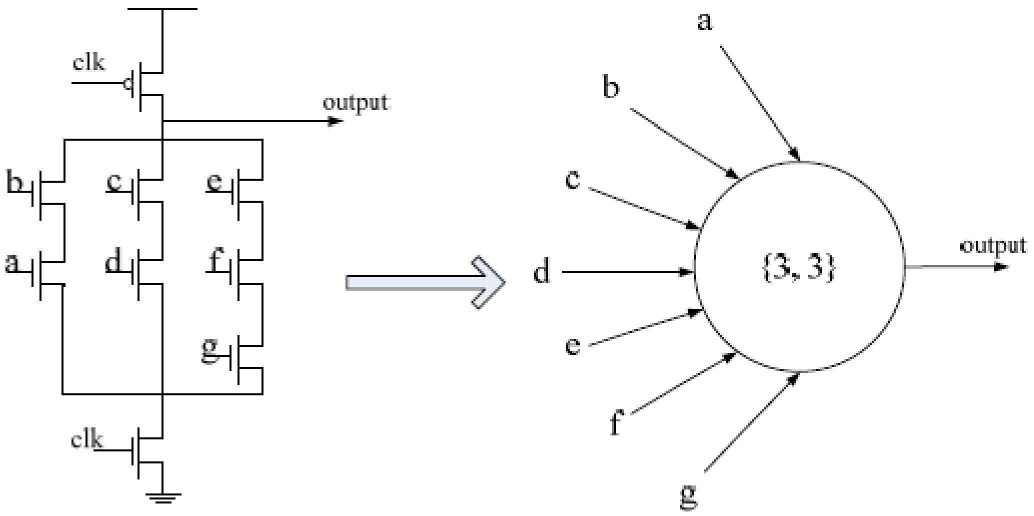


Figure1. A dynamic cell and its nodal representation

Re-ordering Cell: This is a functionally equivalent cell which replaces an original cell along the critical path after re-ordering is called re-ordering cell.

Functional Equivalence: During the mapping procedure, two different cells C1;C2 may have different architecture, yet they may realize the same function. In this scenario, we can say that functional equivalence exists between cells C1 and C2.

Equivalence Table: The Equivalence Table (ET) keeps a database of all the possible cells (under height, width constraints) and their corresponding functionally equivalent cell. It also stores additional information like, the improvement in delay, penalty on area .etc when reordering is done.

Lemma 1: A minimum of 5 input gate must be used to gain Domino advantage.

Proof: Every domino gate requires 4 number of additional transistors in the form of inverter, precharge and evaluate transistors. For an N input cell the Static CMOS logic style requires 2N number of transistors. In order to gain advantage using Domino style, Eqn. 1 must be satisfied. Hence, the number of inputs to the circuit must be greater than 4.

$$N + 4 < 2N \tag{1}$$

Lemma 2: The delay of a particular node increases with the increase in height of the node.

Proof: The resistance of individual nmos transistors get added if they are present in series. This increases the total resistance, the discharging current has to go through and hence there is an increase in delay. This shows that delay of a node is directly proportional to the height of the node.

3. PROPOSED METHODOLOGY

An overview of our proposed methodology of Cell Re-ordering based Domino On-the-fly Mapping (CRDOM) is shown in Fig. 2.

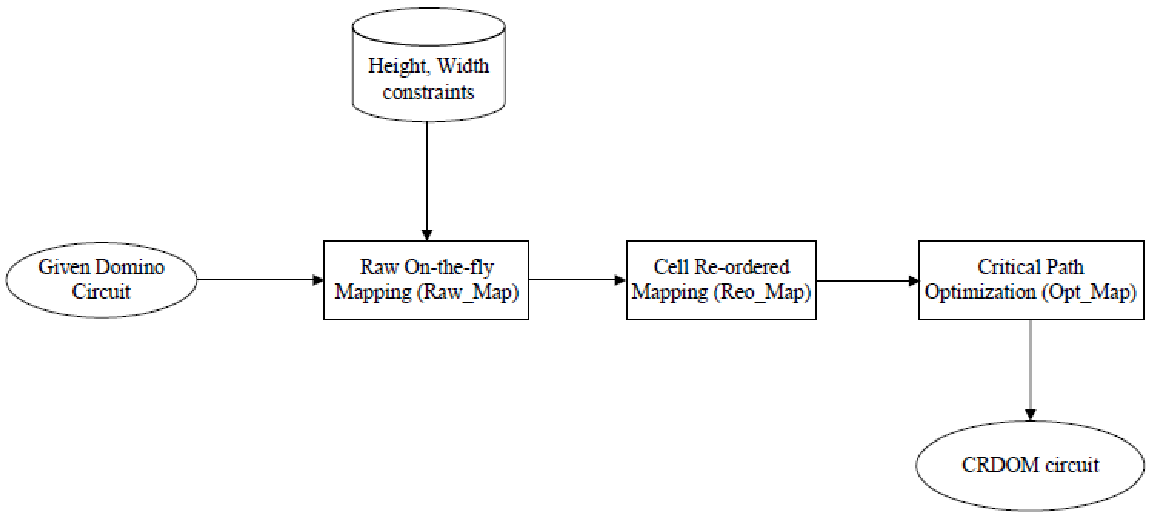

Height, Width constraints
Given Domino Circuit
Raw On-the-fly Mapping (Raw_Map)
Cell Re-ordered Mapping (Reo_Map)
Critical Path Optimization (Opt_Map)
CRDOM circuit

Figure2. Overview of CRDOM approach

Our overall approach consists of the following operation:

Given a circuit C_{init} which is completely unate in nature. Let L_{dom} be the library of various gates that has done the initial mapping of the circuit.

Raw Map: A On-the-fly mapping step, denoted as Raw_Map which takes the circuit Cinit containing a set of gates g_1, g_2,… g_n as input and gives C_{Raw_Map} as the resulting circuit which consist another set of gates G1,G2, … Gk, which obey certain height, width constraints. We represent this as $Raw\_Map\{C_{init}(g_1,g_2, \dots gn)\} \rightarrow C_{Raw\_Map}(G_1,G_2, \dots G_k)$

Reo Map: A Boolean reordering step which takes input as an On-the-fly mapped circuit and verifies whether or not, each gate present along the critical path is at its lower bound realization w.r.to delay. If not, this step replaces the gates with the corresponding functionally equivalent gates which are lower bound w.r.to delay. We define this as

$Reo\_Map\{C_{Raw\_Map}(G_1,G_2, \dots G_l)\} \rightarrow C_{Reo\_Map}(G^*_1,G^*_2, \dots G^*_m)$, where $(G_1,G_2, \dots G_l)$ are gates obtained after Raw Map step and $(G^*_1,G^*_2, \dots G^*_m)$ are gates obtained after Reo Map step.

Opt Map: Finally an optimum set of re-ordering cells is found done which optimizes the critical path delay of the circuit and the area penalty obtained. Say

$Opt\_Map\{C_{Reo\_Map}(G^*_1, G^*_2, \dots G^*_m)\} \rightarrow C_{opt\_Map}(G^0_1, G^0_2, \dots G^0_m)$. where $(G^0_1, G^0_2, \dots G^0_m)$ are the final gates present along the critical path of the circuit.

A detailed description of the above steps with illustrations is presented in the following.

3.1. Raw Mapping

In this section we present an algorithm which maps an arbitrary library based Domino circuit to an On-thefly Domino circuit using a node mapping algorithm. A flow chart of the algorithm is shown in Fig. 3. Various notations used in the flowchart are presented in the following.

Notations used in Node Mapping Algorithm			
RN	Root Node	NN	New Node
CN	Current Node	LN	Leaf Node
GF_Oper	Gate Formation Operation	*Comb_Oper*	Combination Operation

Before explaining the algorithm, we define the following operations which we use in our mapping procedure.

Comb_Oper: In this operation the two or more nodes belonging to a circuit graph are combined to form a new node. If the nature of operation of the node is AND the height of the new node will be the summation of individual height of all the nodes and the width of the node is the maximum of all the nodes. The vice versa holds true for OR operation nodes. The details are presented in Eqn. 2 and Eqn. 3.

$$N_3 = AND(N_1,N_2) \rightarrow H(N_3) = H(N_1) + H(N_2),\ W(N_3) = Max(W(N_1),\ W(N_2)) \qquad (2)$$

$$N_3 = OR(N_1,N_2) \rightarrow W(N_3) = W(N_1) + W(N_2),\ H(N_3) = Max(H(N_1),\ H(N_2)) \qquad (3)$$

where N_3 is a new node formed after combining N_1 and N_2.

GF_Oper: The mapping procedure happens under certain H, W constraints. Whenever, the constraints are exceeded, there a gate is formed and further a new node starts with H=1, W=1. This node will carry forward the output of formed gate. Hence, this operation is called Gate formation Operation (*GF_Oper*).

In the following we explain various steps listed in the flowchart with the help of an example. To start with we consider the Boolean equation mentioned in Eqn. 4

$$f(x) = (a + bc)\ (c + bd) \qquad (4)$$

where a, b, c, d are the primary inputs to the circuit. The graphical representation of Eqn. 4 is also shown in Satge.1 of Fig.4. The initial mapping is already done using a 2-input AND, OR Domino gates

In our algorithm the height, width limits for mapping are set to $H_{Max} = 4$, $W_{Max} = 6$. As shown in Stage 1 of Fig.4, initially all the leaf node (LN) s have a dimension {1,1}. Hence all of them satisfy the dimension (height, width) limits. Next, as mentioned in flowchart (Fig. 3), the node 4 which has inputs b and c is considered as current node (CN). Performing Comb Oper on this node will lead to a node of dimension {2,1}. A similar operation is performed on node 5 which results in formation of a new node (NN) of dimension {2,1}. These are shown as Stage. 2 of Fig. 4. Since, these dimensions are within the maximum defined limits the Comb_Oper is further applied on nodes 2, 3 respectively. After Comb_Oper is performed on node 2, a NN ‘6’is formed with dimensions {2,2} and inputs a,b,c. Similarly a NN '7' is formed by applying Comb Oper on node 3. It also has dimensions {2,2} and inputs b,c,d (shown in Stage. 3 of Fig. 5). Since both nodes 6, 7 satisfy the height, width constraints we futher perform the Comb Oper on nodes 1, 6, 7. This will result in node 8 (as shown in stage 4 of Fig. 3). It has dimensions {4,2} and inputs a, b, c, d. Since the root node (RN) '1' is covered, our algorithm terminates at this stage, here giving output as a graph containing a single node '8'. The *GF_Oper* mentioned in flowchart (shown in Fig. 3), was carried out at last in the current example. This is because of the fact that none of the nodes have exceeded the predefined limit for dimensions.

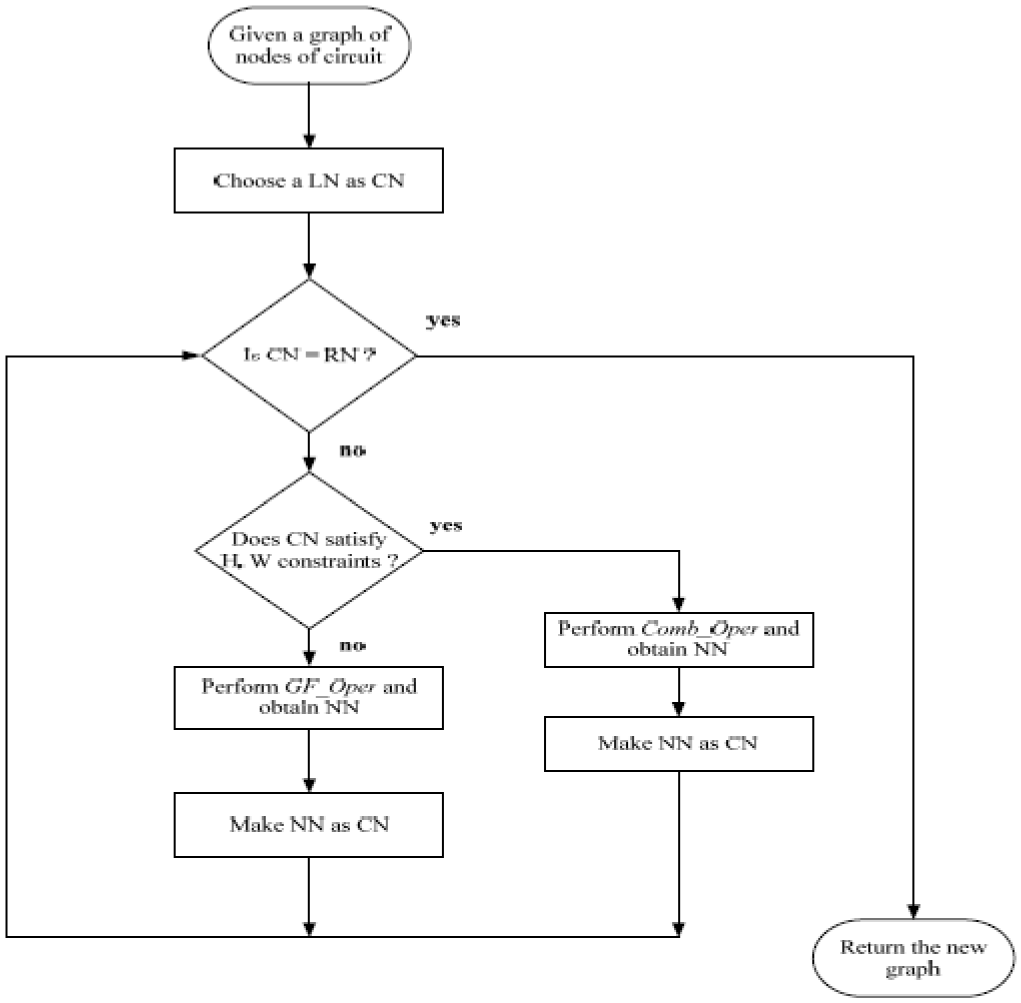

Given a graph of nodes of circuit
Choose a LN as CN
Is CN = RN ?
yes
no
Does CN satisfy H, W constraints ?
yes
no
Perform *Comb_Oper* and obtain NN
Make NN as CN
Perform *GF_Oper* and obtain NN
Make NN as CN
Return the new graph

Figure 3. Flowchart of Node Mapping Algorithm

However the above process obtains a raw On-the-fly mapping of Domino circuits. We can get better performance of the nodes by following a fine tuning technique namely re-ordering of nodes (cells) along the critical path. It is mentioned in the following section.

3.1. Re-ordered Mapping

In this section we first present the need for cell re-ordering for minimum Delay. Later we present an approach for obtaining minimum delay mapped circuit.

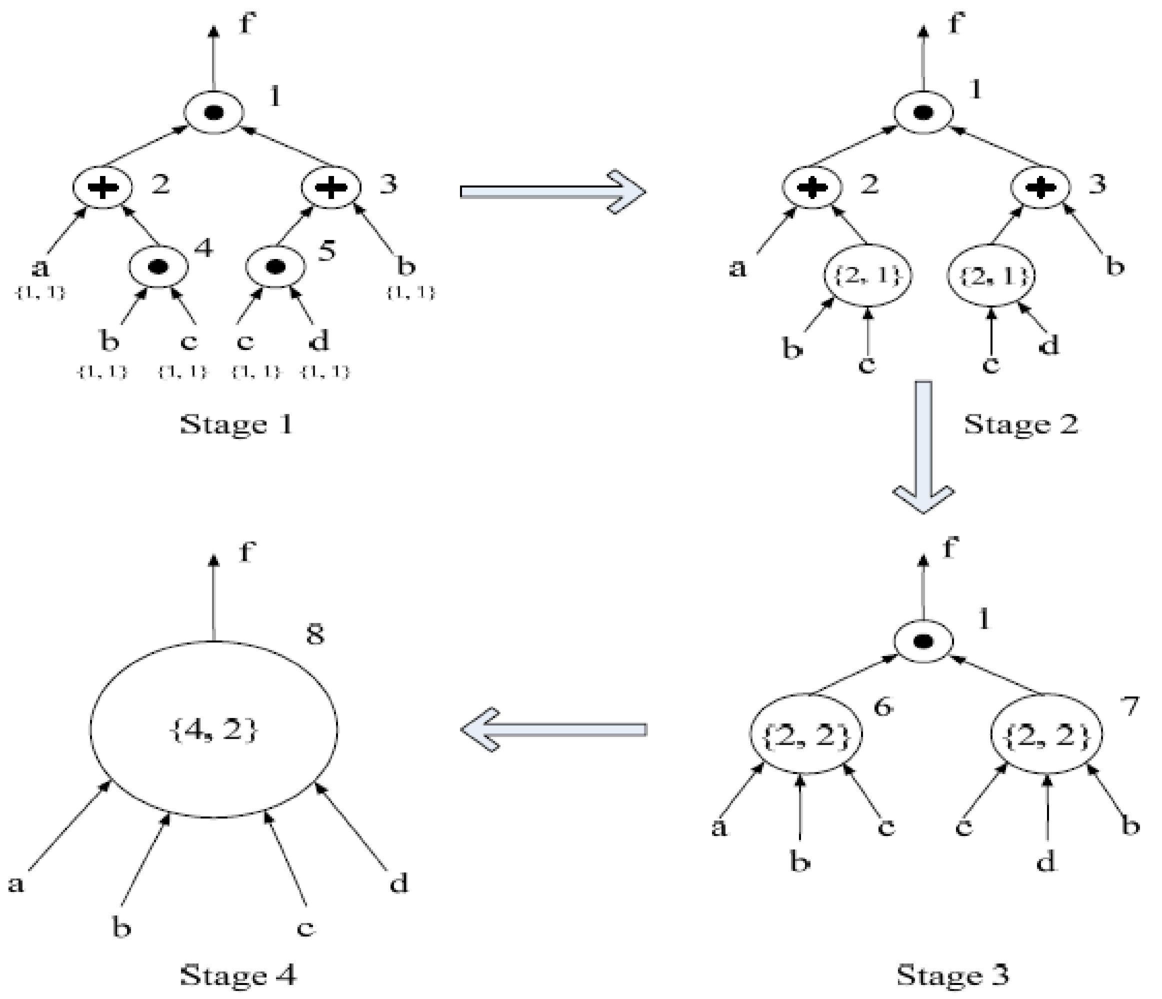


Figure 4. Example of node mapping algorithm

Any Domino cell having a height, width constraints has the capacity to realize wide range of functions. The higher the dimension of the Domino cell, the wider range of functions it can accommodate. Often, due to reasons like factoring, elimination, substitution the number of logic terms to be realized change drastically [27], [28], though the equivalent function to be realized remains the same. These transformations have an effect on the dimensions of the node. From lemma 2, these will have an effect on the delay of the cell. Since there is a change in the logic terms to be realized, there will be a difference in the number of transistors utilized. For example, the two circuits shown in Fig 5.a have different heights and number of transistors under utilization. Yet, both of them realize the same function. Three other similar Equivalence cases are shown in Fig. 5. For, various such cells the best possible re-ordered cell, along with amount of delay advantage and area penalty can be stored in table called Equivalence Table. For the circuit shown in fig. 6, Equivalence Table (ET) containing four original cells is shown in Table 1. Only the re-ordering cells which give a positive delay advantage are considered. If more than one re-ordering cell exist for a given original cell, then the cell which gives better delay advantage is chosen as the final re-ordering cell.

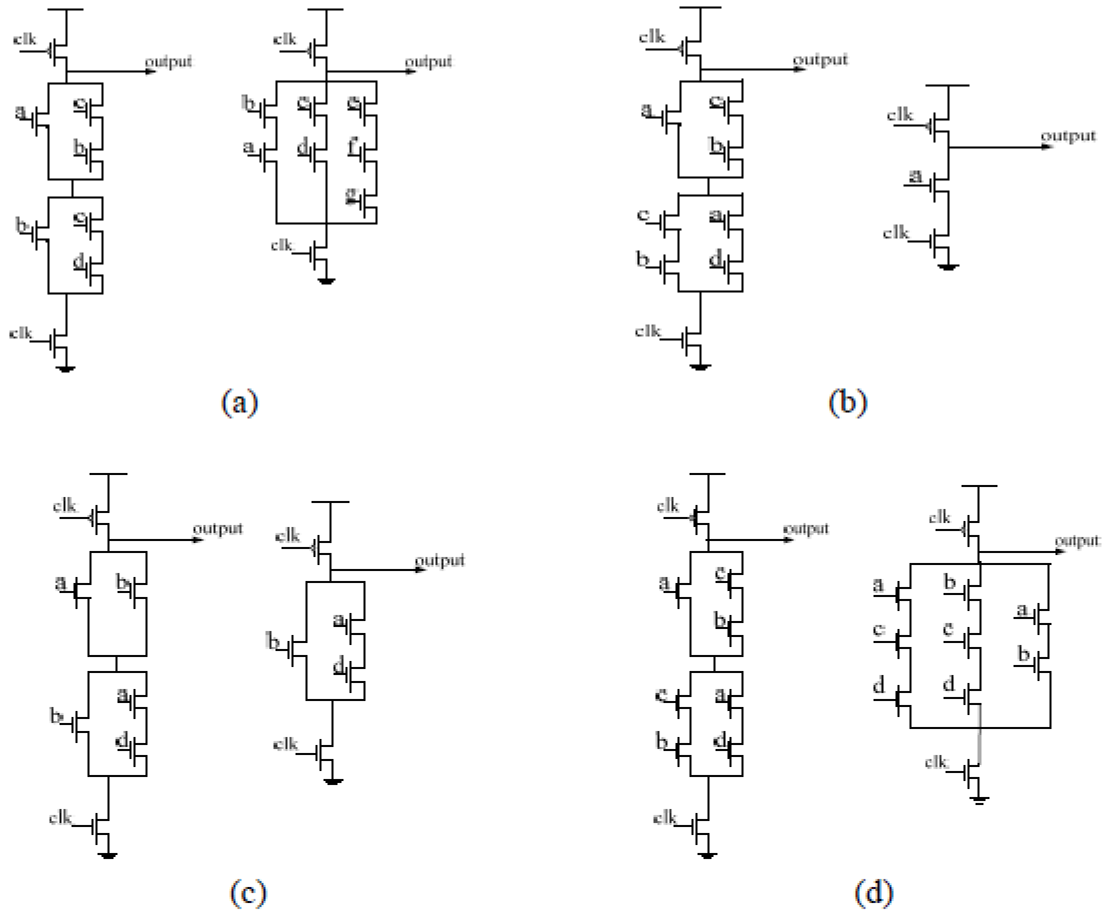


Figure 5. Re-ordering cases for cells (a.) C_1, (b.) C_2, (c.) C_3, (d.) C_4

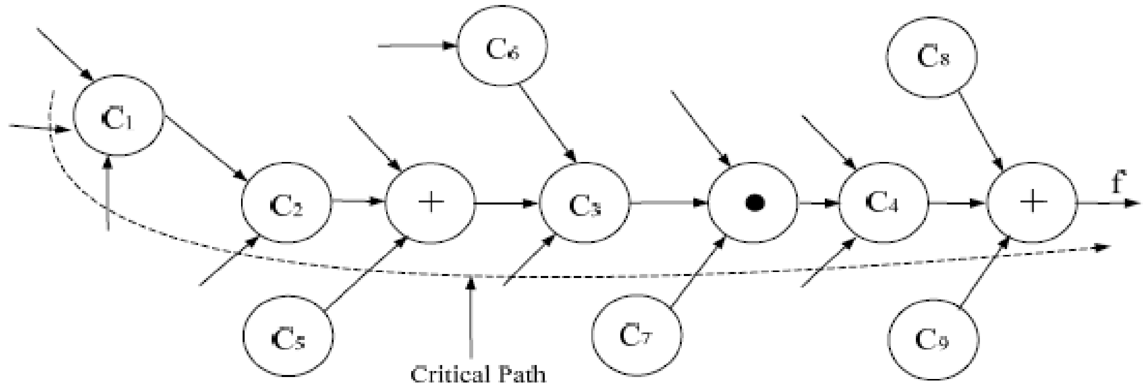
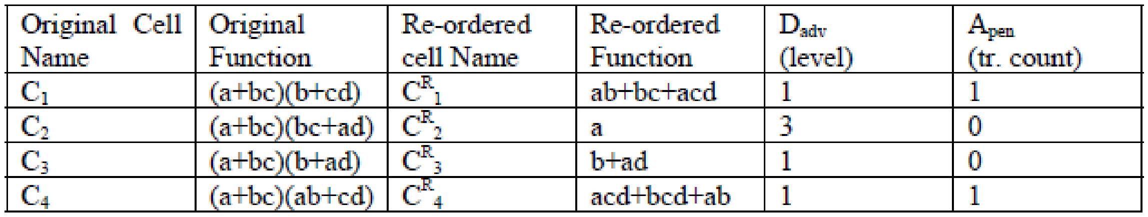


Figure 6. An example circuit with cells C_1, C_2, C_3, C_4 along critical path

Using the ET shown in Table. 1, the Reo Map step is performed on circuit shown in Fig. 6. In order to obtain the overall delay advantage we focus only on the cells existing along the critical path. These cells are cross checked with the cells present in the ET and replaced if Equivalence exists. This approach is repeated for all the cells along the critical path such that the circuit's final delay is minimized. Alternatively, an area oriented re-ordering can be performed, where the Equivalence for all the cells (present in both critical and non-critical paths) is checked for, and replaced if area advantage exists. This is beyond the objective of current work.

Table 1. Equivalence Table (ET) for the circuit shown in Fig. 6



Original Cell Name	Original Function	Re-ordered cell Name	Re-ordered Function	D_{adv} (level)	A_{pen} (tr. count)
C_1	(a+bc)(b+cd)	C^R_1	ab+bc+acd	1	1
C_2	(a+bc)(bc+ad)	C^R_2	a	3	0
C_3	(a+bc)(b+ad)	C^R_3	b+ad	1	0
C_4	(a+bc)(ab+cd)	C^R_4	acd+bcd+ab	1	1

Depending on H, W constraints a wide variety of nodes can be formed during the Raw Map step. This increase in diversity of nodes proportionately increases the number of cells present in the Equivalence Table. We can conclude that, for a given circuit with mapping constraints the size of an Equivalence Table (ET) is finite.

In the following section we further describe a combined delay, area aware optimization of the above mentioned mapping approach.

3.3. Critical Path Optimization

In this section, first we state the need for finding an optimum re-ordering of cells along the critical path. Next, we state the optimization problem clearly defining objectives, constraints and design parameters in it. Finally, we suggest a two objective Particle Swarm based approach to solve the Cell Re-ordering Optimization (CROPT) problem.

After performing the re-ordering of cells along the critical path, it is observed that replacing different sets of cells give different delay advantage and area penalty. As a consequence, arbitrarily reordering various cells along the critical path may not lead to optimum results with respect to delay advantage and area penalty. In fact we have a flexibility to choose a set of cells to be reordered such that the final re-ordering will be optimum in terms of delay and area. Therefore, a judicial choice must be made, in order to achieve optimum realization for a given Reo Map. We call this problem as Cell Re-ordering Optimization problem (CROPT). We formally define the CROPT problem in the following. We refer the following notations in our optimization.

For a given RS_{init} let $\{C^R{}_1, C^R{}_2, \ldots C^R{}_i\}$ be the *i* number of re-ordering cells for a *i* number of original cells along the critical path of a circuit $C_{init.}$ These are obtained from the Reo_Map step. Our objective is to find an optimum re-ordering set (RS_{opt}) resulting in a particular re-ordering of critical path. We call this as Cell Re-ordering Optimization (CROPT), that is

$$Reo\_Map(RS_{init})\ Opt\_Reo(RS_{opt})$$

Notations used in solving CROPT			
RS	Re-ordering set	REP	Repository
W	Damping coefficient	R1	Cognitive coefficient
R2	Social coefficient	K	Number of cells along critical path
LBEST	Local best particle in current iteration	GBEST	Global best particle over all occurred iterations
X_j^i	Position vector for particle I in j^th iteration	V_j^i	Velocity vector for particle I in jth iteration
MAX	Population Size	ITER	Limit on number of iterations

We consider the following two objective functions to judge the optimality of RS. Suppose an arbitrary re-ordering set be $RS_k = \{C^R_1, C^R_2, ... C^R_k\}$. Let $f_d(C_{init}, RS_k)$ denote the delay advantage (D_{sav}) obtained by re-ordering the circuit with RS_k. Similarly $f_a(C_{init}, RS_k)$ denote the estimation of area penalty (A_{pen}) incurred while re-ordering. We aim to model our CROPT as minimization problem. Hence, we define another function InD_{sav} which is the inverse of D_{sav} and is defined as $InD_{sav} = (1 + D_{sav})^{-1}$

We define an operation to find the optimum re-ordering set $\{RS_{opt}\}$ as the Opt Reo, if it satisfies the following.

Given Reo_Map $(C_{init}, C^R_1, C^R_2, \dots C^R_i)$: Opt_Reo (C_{init}, RS_{opt}) = minimize $\{A_{pen} = f_a(C_{init}, RS_k)$, $InD_{sav} = (1 + f_d(C_{init}, RS_k))^{-1}\}$, subject to $InD_{sav} \leq D_0$, $A_{pen} \leq A_0$, $\{RS_k\}$ $\{ C^R_1, C^R_2, \dots C^R_i \}$ for some constraints D_0, A_0.

PSO Based Optimization: We follow a Particle Swarm Multi Objective Optimization algorithm (MOPSO) approach to solve the CROPT problem. A brief explanation of fitting our problem in the MOPSO framework is presented in the following. We have chosen the MOPSO procedure, since it has got a high speed of convergence compared to other multi objective optimization approaches [29]. Also this procedure has less dependence on the set of initial points when compared with other domino based approaches [30]. Various steps followed in the MOPSO procedure are mentioned below.

We have defined the limit of population as MAX. For each particle in the population we have initialized the position vectors, the RS. At the beginning of the optimization all the particles are assigned zero velocity. Using respective position vectors, the fitness values of InD, A are computed. The positions of the particles that represent the non-dominated vectors are stored in the repository. We generate a two dimensional hypercube for each particle corresponding to each objective function to be optimized. The local best for each particle is also stored in the memory for each iteration. The velocity vectors for the particles are updated in each iteration using Eqn. 5. The velocity vector of *ith* particle for (j+1)*th* iteration is computed using the particle's position vector in *jth* iteration and the LBEST, GBEST computed at *jth* iteration.

$$V^{j+1}_i = WXV^j_i + R_1X(LBEST[j] - X^j_i) + R_2X(GBEST[j] - X^j_i) \qquad (5)$$

The LBEST of the population is computed in every single iteration. The GBEST is obtained for the overall number of iterations took place till current instance. Velocity of a RS is computed for each of its dimension, in this case InD, Area separately. The procedure for selecting GBEST[j] for jth iteration involves fitness sharing and Roultewheel based selection, as explained in [29]. After the new velocities for the RSs are computed the new positions for each RS are also computed using Eqn. 6. If the respective RSs go beyond the search space then either they are stuck to the relevant boundary or the velocity is multiplied by (-1) such that they go in opposite direction. Again the RSs are evaluated using both the objective functions.

$$X^{j+1}{}_{i} = X^{j}{}_{i} + V^{j}{}_{i} \qquad (6)$$

At each iteration, non dominated solutions go into the *REP* and the dominated *RSs* are removed. Whenever the *REP* size gets full, the *RSs* located in the less populated areas are given preference over the highly crowded RSs. This is done in order to obtain well distributed pareto fronts. In this way the iterations of the process continue till the limit *ITER* is reached. Our Optimization procedure terminates here.

The *REP* maintains a historical record of all the non dominated *RSs*. To decide a certain *RS* should be added to the *REP* or not, the non dominated *RSs* found at each iteration in the primary population of are compared (on a one-per-one basis) with respect to the contents of the *REP* which, at the beginning of the search will be empty. If the *REP* is empty, then the current RS is accepted. If this new *RS* is dominated by an individual within the *REP*, then such a RS is automatically discarded. Otherwise, if none of the RSs contained in the *REP* dominates the RS wishing to enter, then such a *RS* is stored in the *REP*. If there are RSs in the REP that are dominated by the new *RS*, then such RSs are removed from the *REP*. If *REP* is full, then we start selecting less crowded *RSs* to maintain good spread in the solution set. Finally, from the repository of non dominated *RSs*, we select one RS based on nadir point computation [31]. This will give us the RS_{opt}, i.e. the exact cells that are to be re-ordered along the critical path, so that we can get the maximum benefit on delay and at a minimum area penalty. For the example circuit shown in Fig. 6, 24 number of distinct RS are possible. After performing CROPT we obtained the pareto front as { $C^{R}{}_{1}$, $C^{R}{}_{2}$, $C^{R}{}_{3}$ }, { $C^{R}{}_{2}$, $C^{R}{}_{3}$, $C^{R}{}_{4}$ }. Out of both we choose first set as optimal solution based on Nadir point computation.

In the following section we describe the various experiments we have carried out to verify the efficacy of our proposed approach.

4. EXPERIMENTS AND EXPERIMENTAL RESULTS

In this section, we present details on various experiments conducted to substantiate the efficacy of our proposed approach. First, we mention the various objectives for which the experiments are carried out. Then we describe the experimental setup, which we have used while implementing our proposed method and results obtained. We also mention the benchmarks that we have considered for carrying out the experiments. Finally, we present a comparative study of obtained results with those of existing techniques.

4.1. Objectives

We validate our On-the-fly mapping approach on a set of benchmark circuits. Initially, we present the variation in delay of domino cells with respect to changes in their height and width constraints. Also, we present a comparison of the adopted node mapping approach with the standard library based mapping approaches. Next, we estimate the number of possible reordering cases with respect to height, width constraints of Domino cell and also measure the effect of carrying out cell re-ordering along the critical path. Finally, we present the advantages of optimum re-ordering approach in comparison to raw mapping and re-ordered mapping approaches. Also we present the comparative study of various other mapping approaches with different steps of our proposed methods namely, Raw_Map, Reo_Map and Opt_Map.

4.2. Experimental Setup

All the cells generated in during the Raw Map step belong to the Domino logic style. In order to track the connectivity of various gates present in the initial Domino net list, we used the Berkeley SIS tool, Version 1.3 [32]. The delay savings and area penalty of various Domino cells are computed using simulations performed in 0.18_m CMOS process, 1.8V, 27^{o}C. Area penalty is computed in terms of number of transistors, and delay in order of nanoseconds.

The node mapping algorithm is written in C programming language and compiled using GCC compiler. Experiments are performed on Linux platform with an Intel Core2Duo(2.8 GHz) processor. For applying the node mapping algorithm we restricted the dimensions of cells, height and width to be 4 and 6 respectively. We increased the height and width of the cells step by step and whenever the constraints have crossed the limit, we obtained a new cell by performing Gateformation operation (GF_Oper). An analysis of delay for cells having various height, width are mentioned in Table 2. These results are obtained using 180nm CMOS process.

The overall delay savings (D_{sav}) and area penalty (A_{pen}) is computed by summing up the delay savings and area penalty of individual re-ordered cells. For the various height, width constraints the number of possible re-ordering sets are shown in Fig. 8.

For performing the optimization we coded the MOPSO procedure in C programming language and compiled using GCC compiler. Experiments are performed in a similar environment as used for implementing Node Mapping algorithm. The population size is taken as N = 4 X k where k is the number of cells present along the critical path. We set the *ITER* value to 2000, where convergence is expected to happen as mentioned in [30]. Also in order to choose the optimal solution from the *REP*, we have used the approach of computing Nadir Point as mentioned in [31].

Table 2. Delay of cells with various height and width (in ns)

		Width of the cell					
Height of the cell		1	2	3	4	5	6
	1	1.08	1.32	1.56	1.98	1.98	2.11
	2	1.23	1.44	1.67	1.89	2.1	2.34
	3	1.67	1.89	2.13	2.45	2.79	3.12
	4	2.15	2.52	2.81	3.11	3.45	3.81

4.3. Benchmark Circuits

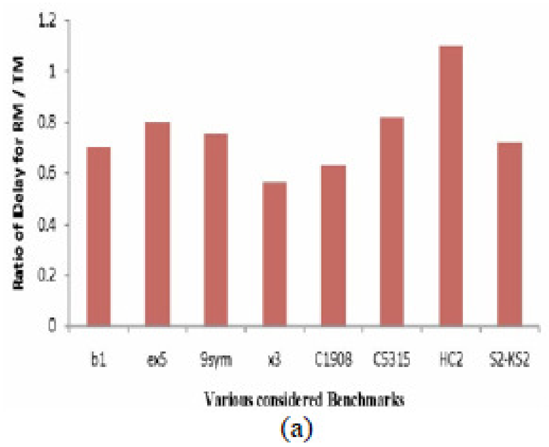
We aimed to test our approach with benchmark circuits having wide range of input variables. Hence, we considered MCNC'89 circuits which start with a 3 input logic (b1.pla) and range up to 135 input logic (x3.pla). We chose ISCAS'85 benchmark circuits as they add industrial flavor to our work. These circuits span from a 33 input and 25 output Error Corrector and Translator circuit (C1908.blif) to a 178 input 123 output variable ALU and Select described using (C5315.blif). Also, we have considered some daily life circuits like Hans-Carlson adder (HC2.blif), Sparse Kogge Stone Adder (S2-KS2.blif). Characteristics of some of the benchmark circuits considered are shown in Table 3.

Table 3. Considered benchmarks from ISCAS85 and MCNC89

Circuit Name	Circuit Function	Input Lines	Output lines
b1	---	3	4
ex5	---	8	63
9sym	---	9	1
x3	---	135	99
C1908	ECAT	33	25
C5315	ALU and Select	178	123
HC2	Hans-Carlson Adder	129	65
S2-KS2	Sparse Kogge Stone Adder	129	65

4.4. Experimental Results

In our experiments, first we run our node mapping algorithm on a set of benchmark circuits mentioned in Table. 3. The overall area (transistor count) and delay (in ns) obtained using the Raw Map step (RM) for each circuit is mentioned in columns 4,5 of Table 4 respectively. We also present the area, delay results obtained by implementing a library based technology mapping approach (TM) [32], in columns 2,3 respectively. We can clearly observe that the approach RM performs better both in terms of area and delay. This is possible because the TM approach repeatedly used 2-input, 3-input AND/Or gates which are present in the library. These gates can't exploit the Domino advantage because of less number of inputs (Lemma 1). The RM mapping required less number of levels, since the *Comb_Oper* of cells will merge many small cells into cells of high dimension. On an average the RM approach gave savings of 28% on delay and 19% on area when compared to the result obtained using TM approach. The ratio of delay and area of various circuits obtained using RM approach to that of the values obtained using TM approach are shown in Fig. 7(a) and Fig. 7(b) respectively.



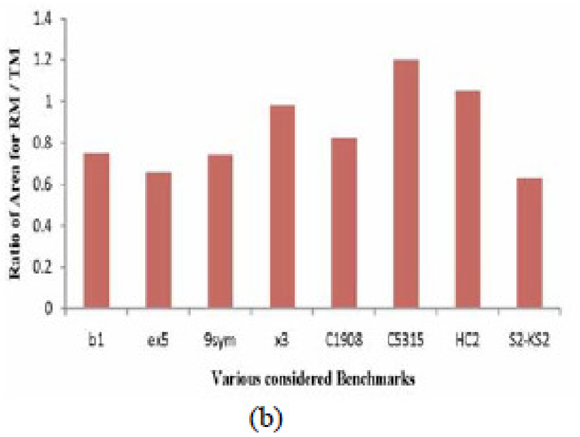


Figure 7. (a) Delay comparison of RM/TM (b) Area comparison of RM/TM

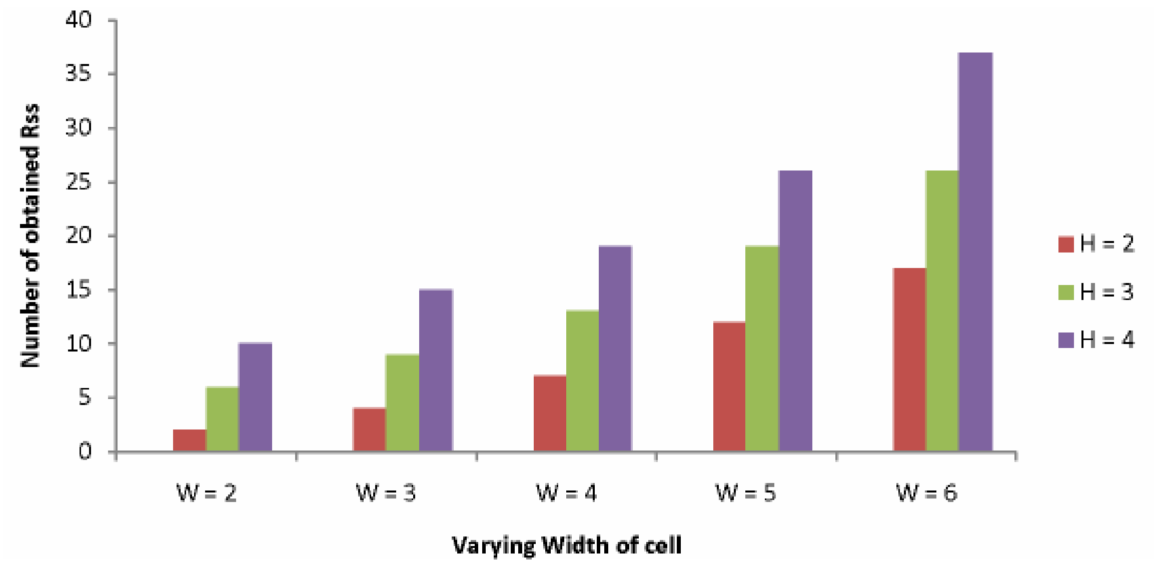


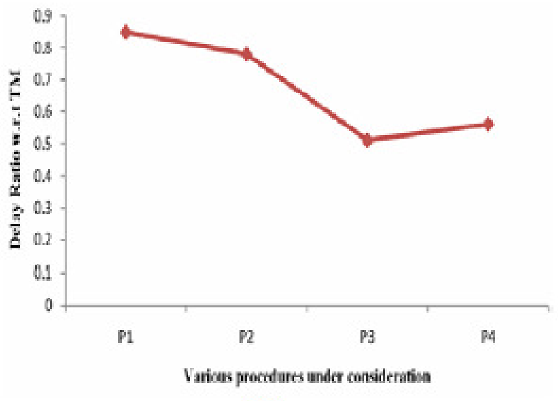
Figure 8. Re-ordered set distribution for cells of various dimensions

After the Raw_Map step is over, we carried out the Reo_Map step on various benchmarks. For the given height, width constraints of cell, we derived various possible original cell structures and obtained the corresponding re-ordering cell (if exists) for each structure. In Fig. 8, we show the number of possible re-ordering cells for various cases; height ranging from 2 to 4 and width ranging from 2-6. It is clearly known that for case H=1, W=1 no re-ordering is possible and hence not shown in the plot. It can be observed from the plot that, as the cell dimension increase, there is an increase in the number of re-ordering cells. This is due to the fact that, higher dimension limits of the cell result in formation of wide range of original cell. Also, such cells can be further reduced using Boolean reduction techniques. Using these cells we build the Equivalence Table (ET) and used it to carryout the Reo Map step. It can be concluded from Fig.8, that for a given height, width constraints of the cell, there exists bound on the number of possible re-ordering cells.

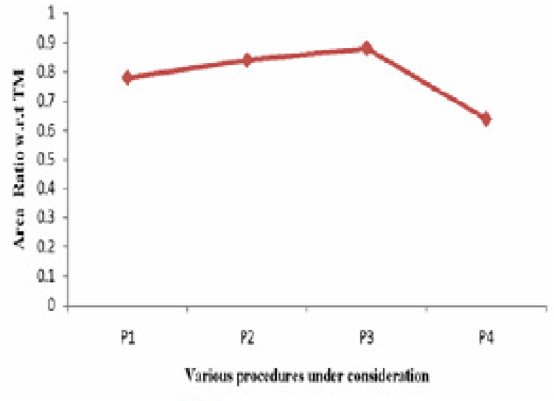
Table 4. Comparison of Technology Mapping with Raw Mapping

Circuit Name	Technology Mapping		Raw Mapping	
	Area (count)	Delay (ns)	Area (count)	Delay (ns)
b1	55	45	41	31.5
ex5	4123	303.4	2721	242.7
9sym	863	394.4	638	295.9
x3	4162	242.4	4078	135.7
C1908	1189	35.7	974	22.49
C5315	2983	29.3	3579	32.23
HC2	3843	45.7	4035	49.35
S2-KS2	3574	39.8	2252	28.65

After doing the re-ordering we have computed the revised delay and area values for all the benchmarks. These are presented in columns 6, 7 of Table. 5 respectively. The approaches *Binpack* [24] and *CCMAP* [13] are also applied on the considered benchmarks and their respective area, delay measures are shown in columns 2-5 respectively. It is clear from the analysis that approach Binpack gives better results in terms of both area and delay compared to approach in *CCMAP* because of heavy logic duplication deficit which occurs in the later approach. Also, *CCMAP* approach is based on dynamic programming approach which takes large time to converge. Yet both the approaches Binpack, *CCMAP* are super ceded by the results of delay obtained using Reo Map by 13%, 17% respectively. This is clear from the fact that the cell re-ordering along the critical path further reduces delay compared to Raw_Map step and better than *Binpack*, *CCMAP* because of the significant reduction in the number of transistors along critical path. Since re-ordering constitute penalty on area we see a 6% increase in area for Reo_Map compared to *Binpack* approach. This penalty is overcome further by Opt Map step. It selectively re-orders the cells along critical path. This lead to a improvement in area by 14%
with a minimal penalty on delay. For a particular ISCAS benchmark circuit C1908.pla, the percentage improvement compared to TM process for delay and area are shown in Fig. 9(a) and 9(b) respectively.



(a)



(b)

Figure 9. For circuit C5315.pla (a) Delay comparison (b) Area comparison over P1 (CCM, [13]), P2 (Binpack, [24]), P3 (Reo_Map), P4 (Opt_Map) approaches normalized to TM

Table 5. Performance evaluation of various approaches

Circuit Name	CCMap [13]		Binpack [24]		Reo_Map		Opt_Map	
	Area (count)	Delay (ns)	Area (count)	Delay (ns)	Area (count)	Delay (ns)	Area (count)	Delay (ns)
b1	55	45	41	31.5	48	22.9	35	25.2
ex5	4123	303.4	2721	242.7	3628	154.7	2638	169.9
9sym	863	394.4	638	295.9	759	201.2	552	220.9
x3	4162	242.4	4078	135.7	3662	123.6	2663	135.7
C1908	1189	35.7	974	22.49	1046	18.2	761	19.9
C5315	2983	29.3	3579	32.23	2625	14.9	1909	16.4
HC2	3843	45.7	4035	49.35	3381	23.3	2459	25.6
S2-KS2	3574	39.8	2252	28.65	3145	20.3	2287	22.3

5. CONCLUSIONS

A cell re-ordering based On-the-fly mapping for Domino logic circuits is proposed in this work. In order to map Domino logic cells the height, width flexibility of cells can be exploited. Otherwise, it leads to increase in area, delay of the overall circuit minimizing advantages obtained using Domino logic. This work proposes to obtain an optimum mapped Domino circuit. Such a circuit will be optimum in terms of both critical path delay and area penalty. Given a Domino block, first we perform (Raw_map), the initial On-the-fly mapping of Domino logic using a node mapping algorithm. Later we perform (Reo_map) the re-ordering of the cells along the critical path so that the delay can be minimized. Finally, we find an optimum set of reordering of cells (RS_opt) that can yield maximum delay savings with a minimum penalty on area. Also the proposed mapping approach gives significant delay savings compared to the library based mapping of Domino circuits. Cell reordering based approach for mapping is comparable with other mapping techniques reported elsewhere. We may conclude that our On the-fly mapping approach especially suits for high speed applications like ALUs, Processor chips etc. However, aspects like considering multiple re-ordering cells for given original cell, area, power oriented mapping are not fully exploited in this work and are yet to be investigated. Our, future research aims to modify the Equivalence Table (ET) and design a power, area aware mapping procedure.

REFERENCES

[1] Massimo, A., Gaetano, P., Melita, P.,(2010) "Understanding the Effect of Process Variations on the Delay of Static and Domino Logic", IEEE Trans. on Very Large Scale Integrated (VLSI) Systems, Vol. 18, No. 5, pp 697-710.

[2] Ali, P., Mohammad, A., (2013) "Current-Comparison-Based Domino: New Low-Leakage High-Speed Domino Circuit for Wide Fan-In Gates", IEEE Trans. on Very Large Scale Integrated (VLSI) Systems, Vol. 21, No. 5, pp 934-943.

[3] Namin, A.H., Leboeuf, K., Muscedere, R., Wu, H., Ahmadi, M., (2010) "High-Speed Hardware Implementation of a Serial-in Parallel-out Finite Field Multiplier Using Reordered Normal basis", IET Circuits Devices Syst., Vol. 4, No. 2, pp 168-179.

[4] Zeydel, B.R., Baran, D., Oklobdzija, V.G., (2010)"Energy-efficient Design Methodologies: High Performance VLSI Adders", IEEE Journal of Solid-State Circuits, Vol. 45, No. 6, pp 1220-1233

[5] Seid, H.R., Hanpei, K., Kaustav, B., (2009) "High-Speed Low-Power FinFET Based Domino Logic". Proc. of ASP-DAC, pp 829-834.

[6] PreetiSudha, M., Mahapatra, K.K., (2013) “A Low-Power Circuit Technique for Domino CMOS Logic”, Proc. of International Conference on Intelligent Sys. and Signal Processing, pp 256-261.

[7] Pandey, A.K., Mishra, R. A., Nagaria, R.K., (2012) “Low Power Dynamic Buffer Circuits”, Int. J. VLSI design & Communication Systems, Vol. 3, No. 5, pp 53-65.

[8] Wairya, S., Nagaria, R.K., Sudarsan Tiwari, (2011) “New Design Methodologies for High-Speed Mixed- Mode CMOS Full Adder Circuits”, Int. J. VLSI design & Communication Systems, Vol. 2, No. 2, pp 78-98.

[9] Zhao, M., Sapatnekar., S.S., (1998) “Technology Mapping for Domino Logic”. Proc. of International Conference on Computer- Aided Design, pp 248-251.

[10] Bobba, S., (2012) “Physical Synthesis Onto a Sea-of-Tiles with Double-gate Silicon Nanowire Transistors”. Proc. of Design Automation Conference, pp 42-47.

[11] Czajkowski, T.S., Brown, S.D., (2008) “ Functionally Linear Decomposition and Synthesis for FPGAs”, IEEE Trans. on Computer Aided Design, Vol. 27, No. 12, pp 2236-2249.

[12] Ciric, J., Sechen, C., (2003) “Efficient Canonical Form for Boolean Matching of Complex Functions in Large Libraries”, IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems, Vol. 22, No. 5, pp 535-544.

[13] Cao, A., Ruibing, L., Chen, L., Cheng-Kok, K., (2006) “ Postlayout Optimization for Synthesis of Domino Circuits”, Trans. On Design Automation of Electronic Systems, Vol. 11, No. 4, pp 797-821.

[14] Marques, F.S., Rosa, L.S., Ribas, R.P., Sapatnekar, S.S., Reis, A.I., (2007) “DAG Based Library-Free Technology Mapping”. Proc. of Great Lake Symposium on VLSI Circuits, pp 293-298.

[15] Martin, L., Cliff, L., (1995) “Cell Generator-Based Technology Mapping by Constructive Tree-Matching and Dynamic Covering”, Journal of VLSI Design, Vol. 3, No. 1, pp. 1-12

[16] Chou, W., Peter, A.B., Ran, G. et al., (1998) “Average Case Optimized Technology Mapping of One Hot Domino Circuits”. Proc. of International Symposium on Advanced Research in Asynchronous Circuits and Systems, pp. 80-91.

[17] Jingyue, X., Dhamin, A., Rozon, C.N., (2004) “Tree-based Transistor Topology Extraction Algorithm for Library-free Logic Synthesis”. Proc. of International Conference on Semiconductor Electronics.

[18] Song, H.Y., Bahar, R.I., (2008), “Power, Delay, and Area Constrained Synthesis for Mixed Domino/Static Logic Optimization”, PhD dissertation.

[19] Diplomacao, T.B., Reis, A.I., (2011) “Domino Logic Library Design and Library Synthesis”, PhD dissertation.

[20] Martinello, O., Marques, F.S., Ribas, R.P., Reis, A.I., (2010) “KL-Cuts: A New Approach for Logic Synthesis Targeting Multiple Output Blocks”. Proc. of Design Automation Test in Europe, pp 777-782.

[21] Shrirang, K.K, Sachin, S.S., (2001) “Technology Mapping for SOI Domino Logic Incorporating Solutions for the Parasitic Bipolar Effect”. Proc. of Design Automation Conference, 2001, pp. 1094-1105

[22] Zhao, M., Sachin, S.S., (2002) “Technology Mapping Algorithms for Domino Logic”, Trans. on Design Automation of Electronic Systems, Vol. 11, No. 4, pp 797-821.

[23] Pullerits, M., Kabbani A., (2008) “Library-Free Synthesis for Area-Delay Minimization”. Proc. of International Conference on Microelectronics.

[24] Yoshikawa, K., Inui, S., Hagihara, Y., Nakamura, Y., Yoshimura, T. (2006) “Domino Logic Synthesis System and its Applications”, Journal of Circuits, Systems and Computers, Vol. 15, No. 2, pp 277-287.

[25] Samanta, D., Pal, A., (2003) “Synthesis of Mixed CMOS VLSI Circuits”, PhD dissertation.

[26] Luca, A., Pierre-Emmanuel, G., Micheli, G.D.: ’MIXSyn: An Efficient Logic Synthesis Methodology for Mixed XORAND/ OR Dominated Circuits’. Proc. of ASP-Design Automation Conference, 2013, pp.133-138.

[27] Roy, R., Bhattacharya, D., Boppana, V., (2005) “Transistor-level Optimization of Digital Designs With Flex Cells”, IEEE Computer, Vol. 38, No. 2, pp 53-61.

[28] Schneider, F., Ribas, R., Sapatnekar, S., and Reis, A., (2005) “Exact Lower bound for the Number of Switches in Series to Implement a Combinational Logic Cell”. Proc. of International Conference on Computer Design, pp 357-362.

[29] Coello Coello, C.A., Gregorio, T.P., Maximino, S.L., (2004), "Handling Multiple Objectives With Particle Swarm Optimization", IEEE Trans. on Evolutionary Computing, Vol. 8, No. 3, pp 256-279.
[30] Parsopoulos, K.E., Vrahatis, M.N., (2002) "Particle Swarm Optimization Method in Multiobjective Problems". Proc. Of Symposium on Applied Computing, pp 603-607.
[31] Kalyanmoy Deb., (2002) "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II", IEEE Trans. Evolutionary Comp, Vol. 6, No. 2, pp 182-197.
[32] Brayton, R.K., (1992) "SIS: A System for Sequential Circuit Synthesis". University of California Berkely, June

AUTHORS

Sai Praveen Kadiyala received his B.Tech. degree in Electrical Engineering from Indian Institute of Technology in 2008. Presently, he is working toward the Ph.D. degree at the School of Information Technology, Indian Institute of Technology, Kharagpur. His research interests include VLSI Design, Low power VLSI, High Performance architecture, Design Automation.

Debasis Samanta (A'95–S'02–M'02) received his B.Tech. degree in computer science and engineering from Calcutta University, M. Tech. degree in computer science and engineering from Jadavpur University, and Ph.D. degree in computer science and engineering from Indian Institute of Technology, Kharagpur. He is currently an Associate Professor in the School of Information Technology, Indian Institute of Technology, Kharagpur. His research interests include low power VLSI, biometric processing, and software testing.