SYNTHESIS OF LOW POWER HIGH PERFORMANCE MIXED CMOS VLSI CIRCUITS

Kadiyala Saipraveen

SYNTHESIS OF LOW POWER HIGH PERFORMANCE MIXED CMOS VLSI CIRCUITS

Thesis submitted to the
Indian Institute of Technology Kharagpur
for award of the degree

of

Doctor of Philosophy

by

Kadiyala Saipraveen

Under the guidance of

Dr. Debasis Samanta

School of Information Technology
Indian Institute of Technology Kharagpur
Kharagpur - 721 302, India
January 2015

CERTIFICATE OF APPROVAL

24/07/2015

Certified that the thesis entitled **Synthesis of Low Power High Performance Mixed CMOS VLSI Circuits** submitted by **Kadiyala Saipraveen** to the Indian Institute of Technology, Kharagpur, for the award of the degree Doctor of Philosophy has been accepted by the external examiners and that the student has successfully defended the thesis in the viva-voce examination held today.

(Member of DSC) (Member of DSC) (Member of DSC)

(Member of DSC) (Member of DSC) (Member of DSC)

(Supervisor)

(Internal Examiner) (Chairman)

CERTIFICATE

This is to certify that the thesis entitled **Synthesis of Low Power High Performance Mixed CMOS VLSI Circuits**, submitted by **Kadiyala Saipraveen** to Indian Institute of Technology Kharagpur, is a record of bona fide research work under my supervision and I consider it worthy of consideration for the award of the degree of Doctor of Philosophy of the Institute.

Date: 24/07/2015

Dr. Debasis Samanta
Associate Professor
School of Information Technology
Indian Institute of Technology Kharagpur
Kharagpur - 721 302, India

DECLARATION

I certify that

a. The work contained in the thesis is original and has been done by myself under the general supervision of my supervisor.

b. The work has not been submitted to any other Institute for any degree or diploma.

c. I have followed the guidelines provided by the Institute in writing the thesis.

d. I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

e. Whenever I have used materials (data, theoretical analysis, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.

f. Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Kadiyala Saipraveen

BIOGRAPHY

Kadiyala Saipraveen is currently a Ph.D. research scholar in the School of Information Technology at Indian Institute of Technology Kharagpur, India. In his research he is focusing on developing mixed CMOS circuits using Boolean Decomposition, Mapping and Power minimization techniques. He received his B.Tech. degree in Electrical Engineering from Indian Institute of Technology in 2008. His current research interests include VLSI Design, Low power VLSI, High Performance architecture, Design Automation.

Dedicated to
Sri Sri Radha Govindji & Srila Prabhupada
ISKCON, Kolkata

ACKNOWLEDGMENT

It gives me immense pleasure to write this part of my dissertation. From the initial concept to final product, the advices of many people have been involved for giving the shape of this dissertation.

First and foremost, I would like to express my deepest gratitude and sincere thanks to my advisor, Dr. Debasis Samanta for his inspiration and able guidance all throughout the course of my PhD Program. Without the help, encouragement and patient support I received from my guide, this thesis would never have materialized. His guidance and support is far beyond duty. I feel privileged to have got this opportunity to work with him.

I would like to thank Prof. Rajib Mall, Head of SIT and my faculty members of SIT for extending me all the possible facilities to carry out the research work. I also wish to thank all of my departmental academic committee members for their valuable suggestions during my seminars. I am also grateful to all members of School of Information Technology.

I am grateful to Mr. Anirban, Lab-in-charge, AVLSI Lab; for helping me by providing the necessary access to various tools which facilitated my research. It is great opportunity to me carry out my research work with advanced CAD and EDA tools. I would like to give my thanks to Nijwmwary, Anusua, Shripad, Sayan, Pradipta, Soumalya and all my colleagues who constantly supported me without any hesitation and encouraged me during the course of my work.

I am grateful to my parents and in-laws for their constant encouragement. I would like to give my special thanks to my wife Sirisha, for their sacrifice and support. Finally, I thank my spiritual master *His Holiness Radhanath Swami Maharaj* whose teachings and example are guiding my consciousness at every step.

Kadiyala Saipraveen
School of Information Technology,
Indian Institute of Technology, Kharagpur

List of Symbols and Abbreviations

List of Symbols

$\alpha_{p_2}^{p_1}$	Area ratio of Process p_1 to p_2
$\beta^{p_1} p_2$	Delay ratio of Process p_1 to p_2
A_{pen}	Area Penalty
A_0	Target Area
$B(X)$	Binate Component
$Comb\_Oper$	Gate combination operation
D_0	Target Delay
D_{sav}	Delay Savings
d_{xy}	Eculidean distance between two points x and y
$f(X)$	Arbitrary Boolean function
f_a	Area of Boolean function$f$
f_d	Delay of Boolean function f
f_p	Power dissipation of Boolean function f
GF_Oper	Gate formation operation
H	Height of Domino cell
HL	Hard Limit on Repository
$ITER$	Maximum number of Iterations
L_{Domino}	Domino Cell Library
L_{stat}	Static Cell Library
P_0	Target Power

ps	Pattern Set
REP	Repository
RS	Re-ordering Set
SL	Soft Limit on Repository
$U(X)$	Unate Component
W	Width of Domino cell

List of Abbreviations

AMOSA	Archive Multi Objective Simulated Annealing
BDD	Binary Decision Diagram
COPT	Circuit Optimization
CPL	Complementary Pass transistor Logic
CROPT	Cell Re-Ordering Optimization
DAG	Directed Acyclic Graph
DCVSL	Differential Cascode Voltage Switch Logic
ECAT	Error Corrector and Address Translator
ELDDB	Early Late Delay Difference Bound
FCG	Favorable Clock Gating
ISCAS	International Symposium on Circuits And Systems
ISMA	Index based Subgraph Matching Algorithm
IUD	Initial Unate Decomposition
LCT	Link Cardinality Table
MCNC	Micro Electronics Center of North Carolina
MOPSO	Multi Objective Particle Swarm Optimization
NSGA	Non Dominated Sorting Genetic Algorithm
OS	On Set

OUD	Optimum Unate Decomposition
PACTS	Power Aware clock Tree Synthesis
PBE	BiPolar Effect
PDN	Pull Down Network
POSET	Partially Ordered Set
PSACTS	Power Slew Aware clock Tree Synthesis
PSOPT	Pattern Set Optimization
PTL	Pass Transistor Logic
PUN	Pull Up Network
SOI	Silicon on Insulator
ZBDD	Zero suppressed Binary Decision Diagram

Abstract

Of late, there is a steep rise in the usage of handheld gadgets and high speed applications. VLSI designers often choose static CMOS logic style for low power applications. This logic style provides low power dissipation and is free from signal noise integrity issues. However, designs based on this logic style often are slow and cannot be used in high performance circuits. On the other hand designs based on Domino logic style yield high performance and occupy less area. Yet, they have more power dissipation compared to their static CMOS counterparts. As a practice, designers during circuit synthesis, mix more than one logic style judiciously to obtain the advantages of each logic style. Carefully designing a mixed static Domino CMOS circuit can tap the advantages of both static and Domino logic styles overcoming their own short comings.

We propose a methodology based on unate decomposition to realize a mixed static Domino circuit. We present an algorithm for decomposing a Boolean circuit into unate and binate sub blocks. Using an Influence table approach, the decomposition algorithm obtains the maximum unate set, containing states that can realize a Domino block. Later we attempt to find an optimum part of the unate set to be realized using Domino logic and the rest to be realized using static logic.

Next, we present a novel on-the-fly mapping technique to map the obtained Domino block. We follow a node by node incremental mapping approach combining the nodes based on their functional properties. This is done till the restrictions on height and width of the individual cells are reached. Then by re-ordering of cells selectively, we try to gain advantage in terms of delay and minimizing area penalty. We use a two-objective optimization method to find the optimum set of re-ordering cells.

Finally, we propose a power aware clock gating approach for the Domino blocks of a circuit which reduces the dynamic power dissipation of the blocks. We obtain a set of favorable gate patterns which can yield power savings, when clock gating is applied. These gate patterns, in groups are used to match the circuit using sub graph matching algorithm. We try to find an optimum pattern set which gives maximum power savings with a minimum penalty on area. Our proposed methodology is implemented and tested on the standard ISCAS and MCNC benchmarks. Comparative study with the existing works shows that our approach offered 15% improvement in power, 12% improvement in area and 19% improvement in delay.

Keywords: Unate decomposition, mixed CMOS synthesis, cell-reordering, on-the-fly mapping, sub graph matching, clock gating, low power design

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Of late, there is a steep rise in the usage of battery operated handheld gadgets. The demands for devices operating at low power and high speed are ever growing. With custom made chips coming into focus, designers are trying to realize many functionalities on a single chip. In fact, designers are now pushing billions of transistors on a single chip to realize a wide variety of applications. This increases the density of the chip and further give rise to problems like thermal variations, process variations, packaging, cooling issues etc. Architectural level methods for reducing power dissipation often try to scale supply voltages, operate various modules of a chip at different supply voltages etc. Transistor level methods focus on reducing the threshold limits, scaling down the device size etc. At present designers are also focusing on methods that work at logic level by obtaining novel styles that will reduce power dissipation and improve performance of the circuit. This dissemination is related to logic level synthesis of VLSI CMOS circuits.

1.1 Context of Research

There is a drastic increase in the on chip transistor density over the last decade. As per the prediction of Moore's law, there will be an exponential growth in the number of transistors that can be realized on a single die with respect to time [1]. As the chip complexity is doubling every two years, the law remains strong [2]. The trends in transistor count on microprocessor chip, over the past four decades are shown in Fig. 1.1 [3]. The constant rise in the transistor count proves the aptness of Moore's law. Case study of microprocessor

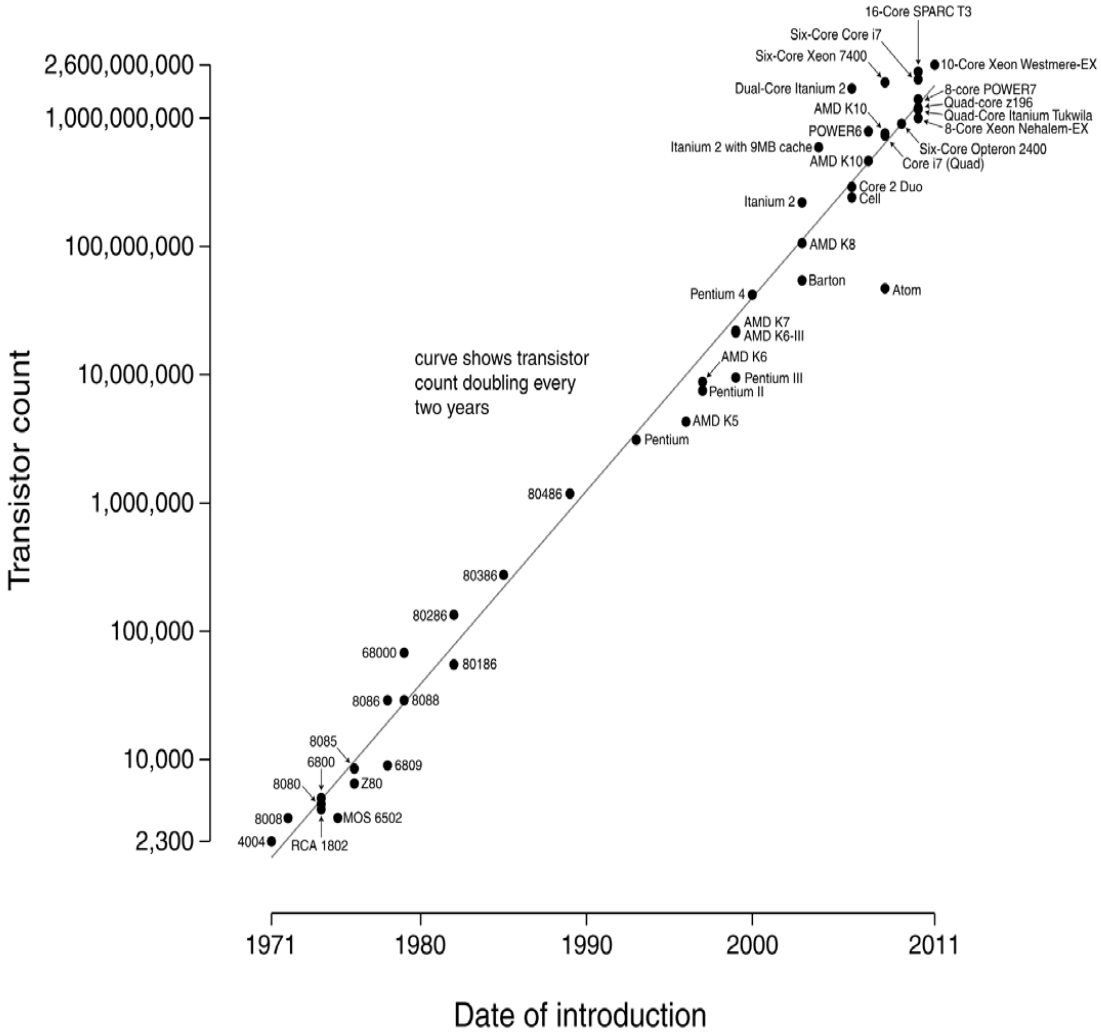


Figure 1.1: Trends of transistor count in microprocessor chips [3]

reveals the steep rise in complexity and performance aspects [4]. From its origin during 1970s and till date the clock frequencies increased from 0.1 MHz to 1 Gigahertz scale [5]. The transistor count to be mounted on chip started from a modest 2000, now crossing billion mark. In early days, each and every transistor is carefully placed and the design is manually optimized and fitted into the environment as is the case for Intel 4004 microprocessor [6]. This is not feasible when millions of transistors are to be integrated, since time to market must be minimized for success of any such component [7]. This led to the evolution of number of design automation tools which handle various aspects of circuit design at a very large scale.

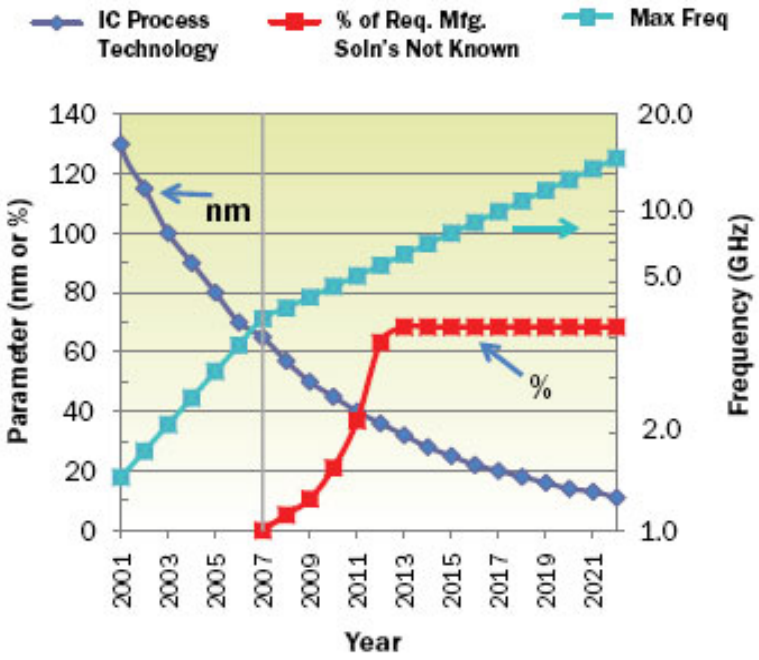


Figure 1.2: Trends of technology and frequency of microprocessor chips [11]

Performance of the circuits can be improved by scaling of channel. Yet, it increases power-density more than expected [8],[4]. Higher levels of integration are often driven by the motive of cost minimization. However, low cost technological breakthroughs to keep improving power savings are getting very rare. Modern system-on-chip (SOC) demands for more power [9], [10]. In both logic and memory, with decrease in device size, static power is growing really fast and so is dynamic power. The trends of the device size and clock frequency are shown in Fig 1.2 [11]. Red symbols indicate the percentage of the set of lithographic requirements, for which there were no known manufacturable solutions. These trends predict a drastic increase of power density inside the chips with a simultaneous increase in speed. Power dissipation is the main constraint when it comes to portability [12]. There is an increase in demand for more features and extended battery life at a lower cost. Each new process beginning with 120nm node, records higher dynamic and leakage current density with a minimum improvement in speed [13]. This requires high levels of silicon integration in advanced processes, but advanced processes have inherently higher leakage current [14]. So there is a need to focus on reducing the overall power dissipation by taking all the above facts into consideration.

Table 1.1: Approaches at various hierarchical levels

S.No	Hierarchical level	Approach
1	System	Partitioning, power down
2	Algorithm	Complexity, concurrency, regularity
3	Architecture	Parallelism, pipelining, redundancy, data encoding
4	Circuit Logic	Logic styles, energy recovery, transistor sizing
5	Technology	Threshold reduction, multi-threshold devices

Power minimization and speed improvement approaches can be performed at various levels of digital design hierarchy. As an alternative for individualized approach, designers are developing circuits using a hierarchical fashion [15], [16] where automation can be easily introduced. This hierarchical approach in designing digital circuits has given an advantage compared to the analog circuits and helps in very large scale design. Various abstraction levels in the digital design flow are system level, module level, register transfer level (RTL), gate level and transistor level, respectively. As we go from system level to device level the amount of abstraction keep on decreasing [17]. Approaches like *power down* and *partitioning* address the power minimization problem at a system level. The later helps in executing the modules sequentially thus minimizing power. Exploiting the concept of regularity is done at algorithmic level, which significantly reduces the excess computations [18]. To save the clock cycle and restrict extra logic, the concepts of encoding data, pipelining and parallelism are often used [19]. These techniques improve the circuit performance at architecture level. At a circuit level, performance improvement is obtained by using different logic styles as per the requirement, approaches like clock gating, energy recovery etc. [20]. Methods like dual V_T and threshold reduction are applicable to minimize power dissipation at technology level [17]. A summary of various approaches valid for different abstraction levels is shown Table 1.1. This thesis primarily deals with the task of performance improvement at a logic level [21]. We focus on designing novel logic styles which would yield low power and high performance for circuits. We chose to work at logic level, because improvements at this level can be effectively clubbed with the gains that are obtained at other abstraction levels.

In the following section, we briefly describe the factors that have motivated us to carry out research in this particular area.

1.2 Motivation of the Work

A given logic function can be implemented using various logic styles. The emphasis on a particular logic style depends on the application in which the design is to be used. Handheld devices and battery operated systems aim for energy saving designs. High performance applications need designs that have faster switching speeds. In this section, we briefly describe various logic styles that are used in circuit synthesis.

A complementary CMOS logic style is a combination of a *pull-up network* (PUN) and *pull-down network* (PDN). All inputs to a gate are distributed to both PUN and PDN networks [22]. An example of 2-input NAND gate realized using complementary CMOS is shown in Fig. 1.3 (a). These PDN and PUN blocks are constructed in a complementary fashion such that at a given instance only one of the paths remains active [23]. In other words, it can be stated that the output node always remains at a *low impedance node* in steady state.

Ratioed logic style tries to realize a given Boolean function using a less number of transistors, however at the cost of more power dissipation and less robustness [24]. In

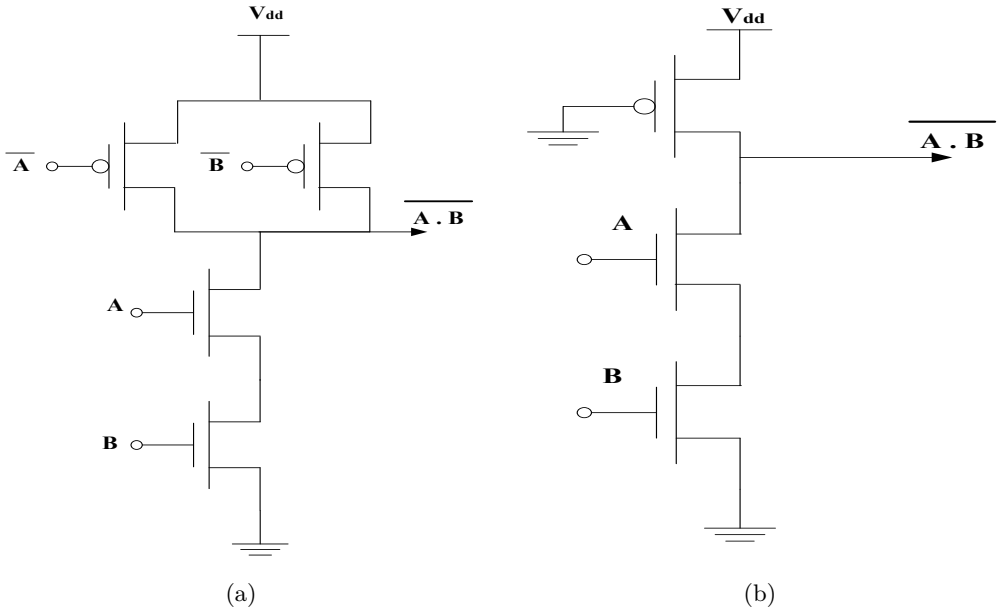
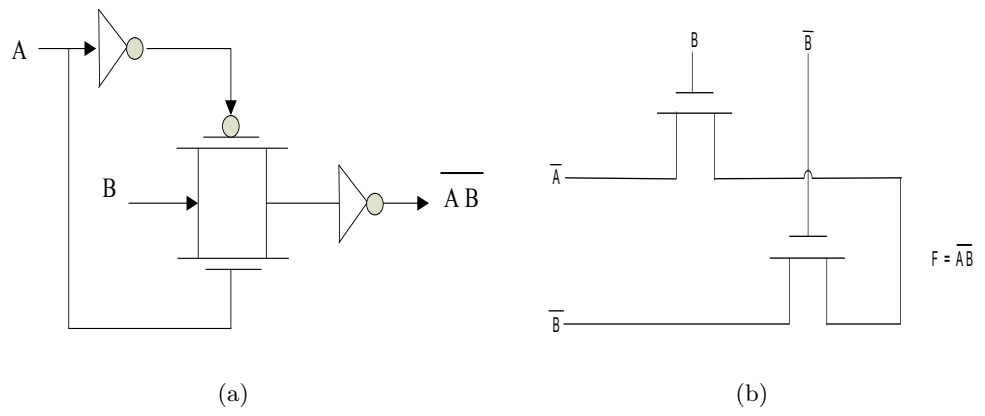


Figure 1.3: Realization of 2-input NAND using (a) Complementary CMOS (b) Pseudo NMOS logic

ratioed logic style, the pull-up network is replaced by an unconditional load which raises the output to '1' when the suitable inputs arrive [25]. A gate designed using this logic style consists of an active pull-down network and a simple load device on contrary to dual networks which have both pull-up and pull-down networks as active blocks [26]. Often the simple load device can be a grounded PMOS transistor. In such cases, the logic style is termed as *pseudo NMOS* logic style. A 2-input NAND gate realization using pseudo NMOS logic style is shown in Fig. 1.3 (b).

In principle, a transmission gate is made up of two field effect transistors, which is in contrast to the traditional discrete field effect transistors. Here, the substrate terminal (bulk) is connected internally to the source terminal [27]. The two transistors, an n-channel MOSFET and a p-channel MOSFET are connected in parallel with this and the drain and source terminals of the two transistors are connected together [10]. Their gate terminals are connected to each other via a NOT gate (inverter) to form the control terminal [28]. Implementation of a 2-input NAND gate using this logic style is shown in Fig. 1.4 (a).

As an alternative to complementary CMOS logic, *pass transistor logic* tries to provide a much simpler logic style by reducing the number of transistors [29]. Here the primary inputs drive not only the gates but also the source-drain terminals. Implementation of 2-input NAND gate using pass transistor logic (PTL) style is shown in Fig. 1.4 (b). The reduced number of transistors lead to low load capacitance and high propagation delay [30]. Yet, this logic style faces the disadvantage of voltage degradation [31].



(a) (b)

Figure 1.4: Realization of 2-input NAND using (a) Transmission gate (b) Pass transistor logic

In order to gain in terms of performance, designers often implement *complementary pass transistor logic* style. A complementary pass transistor logic (CPL) gate consists of two NMOS logic networks (one for each signal rail), two small pull-up PMOS transistors for swing restoration, and two output inverters for the complementary output signals [32]. The AND/NAND gate realization using CPL is shown in Fig. 1.5. The availability of both polarities of every signal helps in realizing the circuit efficiently with small number of transistors [33].

Differential cascode voltage switch (DCVS) also has an inherent self-testing property which can provide coverage of both stuck-at and dynamic faults [34], [35]. A further attraction of DCVS circuits is the fact that they can be readily designed using straightforward procedures based on Karnaugh maps (K-maps) and tabular methods [36]. A 2-input NAND gate realization using DCVS logic style is shown in Fig. 1.6.

The push-pull logic style (PPL) consists of two parts, a pass-transistor logic network for evaluating the logical function and a push-pull level restoring circuit [37]. A 2-input NAND gate realization using push pull logic style is shown in Fig. 1.7. The complementary pass and control variables are connected to the drain and gate of the pass-transistors in the logic network to implement the logic functions AND and NAND, respectively. And for level restoration, the drain of the PMOS is connected to the output of the n-channel network to push the degraded 'high' signal to the supply voltage while the drain of the NMOS is connected to that of the p-channel network to pull down the degraded 'low' signal to the

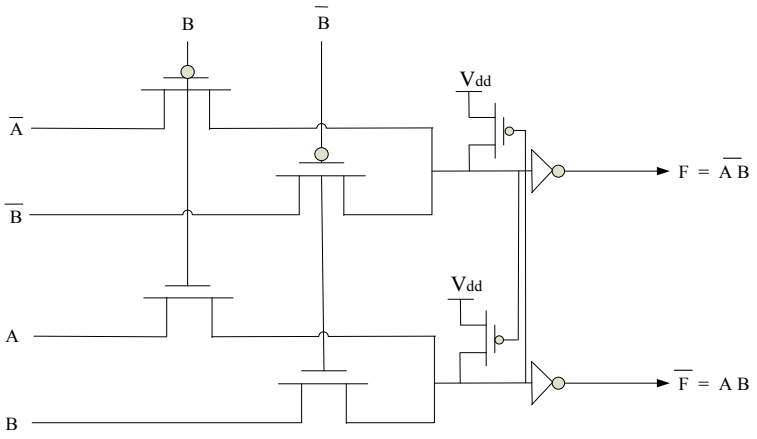


Figure 1.5: Realization of 2-input NAND using complementary pass transistor logic style

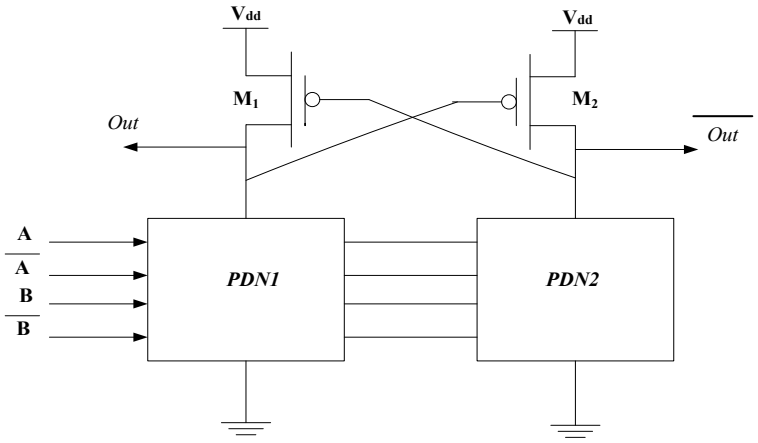


Figure 1.6: Realization of 2-input NAND using differential cascode voltage switch logic style

ground with their gates cross-coupled together [38], [39].

Complementary CMOS style requires 2N number of transistors for N fan-in gate. Various other logic styles which are described focused on reducing the number of devices. As an alternative logic style called *dynamic logic* which obtains similar result without any static power dissipation [40]. An additional *clock* input is needed for this logic style and is based on precharge and conditional discharge phases [41]. The basic topology of this style consists of a precharge and evaluate transistors operated by the *clock* signal, a pull down network as in the case of complementary CMOS style [42]. Clock *CLK* signal drives the two major phases *precharge* and *evaluation* of the dynamic circuit [43].

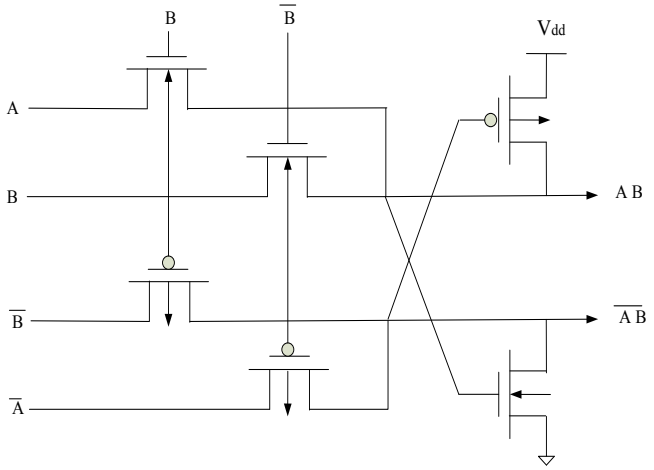


Figure 1.7: Realization of 2-input NAND using push pull logic style

A general Domino logic module consists of a pull down network (dynamic block) made of n-type transistors followed by a simple inverter [44]. A 2-input AND gate realization using Domino logic style is shown in Fig. 1.8. During the precharge phase, the output of the dynamic block is charged up to V_{dd} and the output of the inverter becomes 0. During the evaluation phase the output of dynamic block conditionally reaches to 0, and the inverter output can at the most make a single transition from 0 to 1 [45]. Since, all outputs are assured single transition from 0 to 1 during evaluation, the outputs of these gates can be safely connected as inputs to other gates. The presence of a static inverter at the output gives additional noise immunity, since the fanout is driven by a low impedance output [46], [47].

A different style of cascading dynamic gates is presented in *np-CMOS* logic style which uses alternatively both NMOS and PMOS logic blocks [48]. As shown in Fig. 1.9, in the P-tree PMOS device are used to build the PUN network including the PMOS evaluation transistor. Here, the NMOS precharge transistor drives the output low during precharge [49]. The output conditionally evaluates and makes a transition 0 to 1 accordingly [50].

In Table 1.2, we summarize the performance of different logic styles using some common parameters like power dissipation, delay, transistor count and robustness. Each parameter is classified into various categories namely low, medium, high and very high. These are a comparison against the performance of standard complementary CMOS logic.

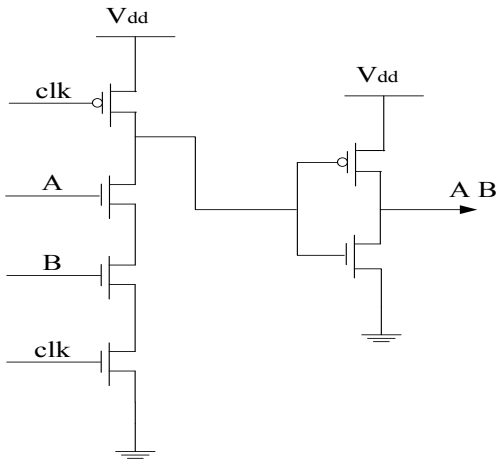


Figure 1.8: 2-input AND gate realization using Domino logic style

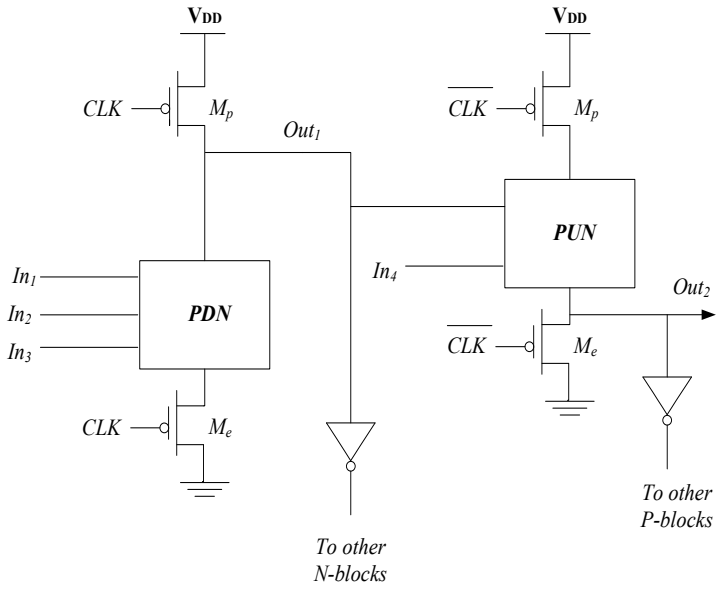


Figure 1.9: A general topology of *np-CMOS* logic style

concepts of *rise time*, *fall time* is introduced. These times are defined by the time gap between 10% and 90% points of the signal during transition. The rise and fall time of a signal are determined by the gate that is driving the signal and the load presented to it.

Table 1.2: Performance of various logic styles

Logic style	Power dissipation	Delay	Transistor count	Robustness
Ratioed logic	high	high	medium	medium
PTL	medium	medium	low	low
Transmission gate	low	medium	medium	high
CPL	high	low	high	low
Push pull logic	high	medium	high	high
Differential cascode voltage switch logic	high	medium	very high	high
Dynamic logic	medium	low	low	low
np CMOS logic	high	low	high	medium

1.2.1 Advantages and Disadvantages of Static CMOS logic style

In this section, we present in brief about a major logic style that is used in the low power design industry, that is the complementary CMOS logic style. For the sake of convenience, we refer the complementary CMOS logic style as static CMOS style in the rest of the thesis. Static CMOS logic style is often used for designing circuits where power dissipation is to be

minimized. This is because, the logic style is simple to fabricate. Since it has both NMOS and PMOS along the rail to rail path this style offers good input/output decoupling. Due to the absence of clock signal the circuits designed with this style have less switching activity. Circuits with this logic style are robust in nature and have good noise margins. However, static CMOS logic is slower because it uses bulky PMOS transistors in its charging path [51], [52].

1.2.2 Advantages and Disadvantages of Domino logic style

On the other hand, Domino logic style is widely used in custom circuit design, especially in high performance oriented circuits. In addition to the benefit with respect to speed, this logic style offers smaller area and ensures glitch free operations [44], [53]. This logic style runs 1.5 - 2 times faster than static CMOS logic because these gates present much lower input capacitance for the same output current and a lower switching threshold [45]. The Domino logic style, which has an additional inverter at the output overcomes these problems of cascading and charge leakage issues which are common in dynamic CMOS logic [47]. Yet, there are difficulties in designing and verifying this class of circuits. Domino logic circuits can implement only non inverting logic [46]. Also, this logic style suffers from signal noise integrity issues. A mutual performance comparison of static CMOS and Domino logic style are presented in Table 1.3.

1.2.3 Mixed CMOS logic style

Of late, to exploit advantages of more than one logic style, designers are using mixed logic style to synthesize digital circuits. Static CMOS logic has a clear advantage in terms of power and Domino logic has advantage in terms of speed and area. In order to claim the combined advantages of both logic, attempts have been made to judiciously mix both the logic styles and synthesize the circuits.

Table 1.3: Performance of Static vs. Domino logic

Logic style	Power dissipation	Delay	Transistor count	Robustness
Static CMOS logic	high	medium	low	high
Domino logic	medium	low	medium	medium

Some approaches exist to decompose circuits into unate and binate components. Bubble pushing algorithm technique attempts to realize a complete unate circuit from a given arbitrary Boolean circuit [54]. This method focused on converting any given circuit into a unate form by applying De Morgan's laws on the constituent nodes. These laws are applied on the internal gates starting from primary outputs going till primary inputs. During the traversal, the approach tries to make every node it encounters into a unate node. Each input of a node is considered for determining its unateness. If a node is binate with respect to a variable, a corresponding new variable is created, which is the complement of the original variable. The new variable is substituted in place of the old, making the node unate. While performing this method, there arise lots of trapped inverters within the circuit (example shown in Fig. 1.10 (a)). making the resulting circuit a unate binate scenario [55]. In continuation to the above technique, a candidate block based replacement technique, of the trapped inverters is proposed for performing the mixed static domino synthesis.

A two-level based decomposition technique also attempts to convert the entire circuit into unate [56], [57] (example shown in Fig. 1.10 (c)). This approach followed a drastically different method. It starts with description of a Boolean function given in PLA format. The large scale circuits are partitioned into sub graphs, each having not more than 15-input variables. As this unate decomposition step is based on (minterm) canonical representation of Boolean functions, its complexity increases exponentially with the number of input variables. In the unate decomposition step, each sub-function is expressed in terms of only positive and negative unate functions, which directly maps to a two-level Domino or no-race (NORA) networks. However, direct realization of this two-level network leads to MOS networks with large number of series/parallel transistors in each cell. To overcome this problem, they performed a multilevel decomposition of each unate sub-function. The major challenge with this technique is, it results in a huge number of extra logic which nullifies the advantages of using mixed logic.

Lastly, to realize mixed CMOS circuits, there exists a BDD based decomposition method [58]. This is a technique for decomposing incompletely specified Boolean functions into unate, binate sub blocks. This approach aimed at improving the quality of circuit by restructuring the netlist. Various steps in the approach included *cover minimization, phase assignment, selection of largest unate component* etc. The initial cube cover is processed

as long as its cardinality reaches under certain limit. Since, the initial cover is binate, extraction of such unate cube subsets is possible. While finding the largest unate subset a greedy computation is used. The cube subset which can be unate with the largest number of cubes in the cover is identified. This method took a cube cover and return the cubes that have largest size. The whole extraction procedure is applied iteratively. It continues till the cover being processed reaches certain limit. The last block thus obtained during the iterative process may end up being binate. Realization of a sample circuit using this process is shown in Fig. 1.10 (b).

1.2.4 Issues and challenges with static Domino mixed logic

Synthesis of Boolean functions using more than one logic styles is a complex issue. Domino logic style can realize only unate functions and static logic style can realize binate functions. Judicious decomposition of Boolean functions into unate and binate sub blocks is a prime necessity for our research. The logic blocks thus obtained must be mapped to appropriate gates. Taking into consideration of individual logic styles different mapping techniques must be adopted. Clock signal which plays a significant role in the functioning of Domino block must be designed carefully. Further, designing a low power clocking approach is a major issue to deal with.

Many approaches have been proposed by researchers to realize mixed logic styles. These include mixing of static and PTL (Pass transistor logic), complex static Domino gate approach and compound Domino approach [8], [13] . These approaches attempted to partition the circuits into individual block and map with respective logic style. Defining an efficient means of partitioning such that the obtained blocks result in an optimum circuit is a big challenge. Various blocks in mixed CMOS logic need a mapping technique to map them. Especially, mapping of Domino block can be done by library free mapping [59]. Many approaches for mapping, mentioned in literature focused on reducing the redundancy and area overhead [60], [61], [62]. Managing the delay along the critical path is always a challenging issue. The mixed logic circuits are expected to perform better than the individual logic styles. Clock gating technique which improves power savings is often employed in sequential circuits [63], [64]. Works involved with clock gating for Domino circuits have focused on bubble pushing based methods for obtaining unate set [65]. Hence,

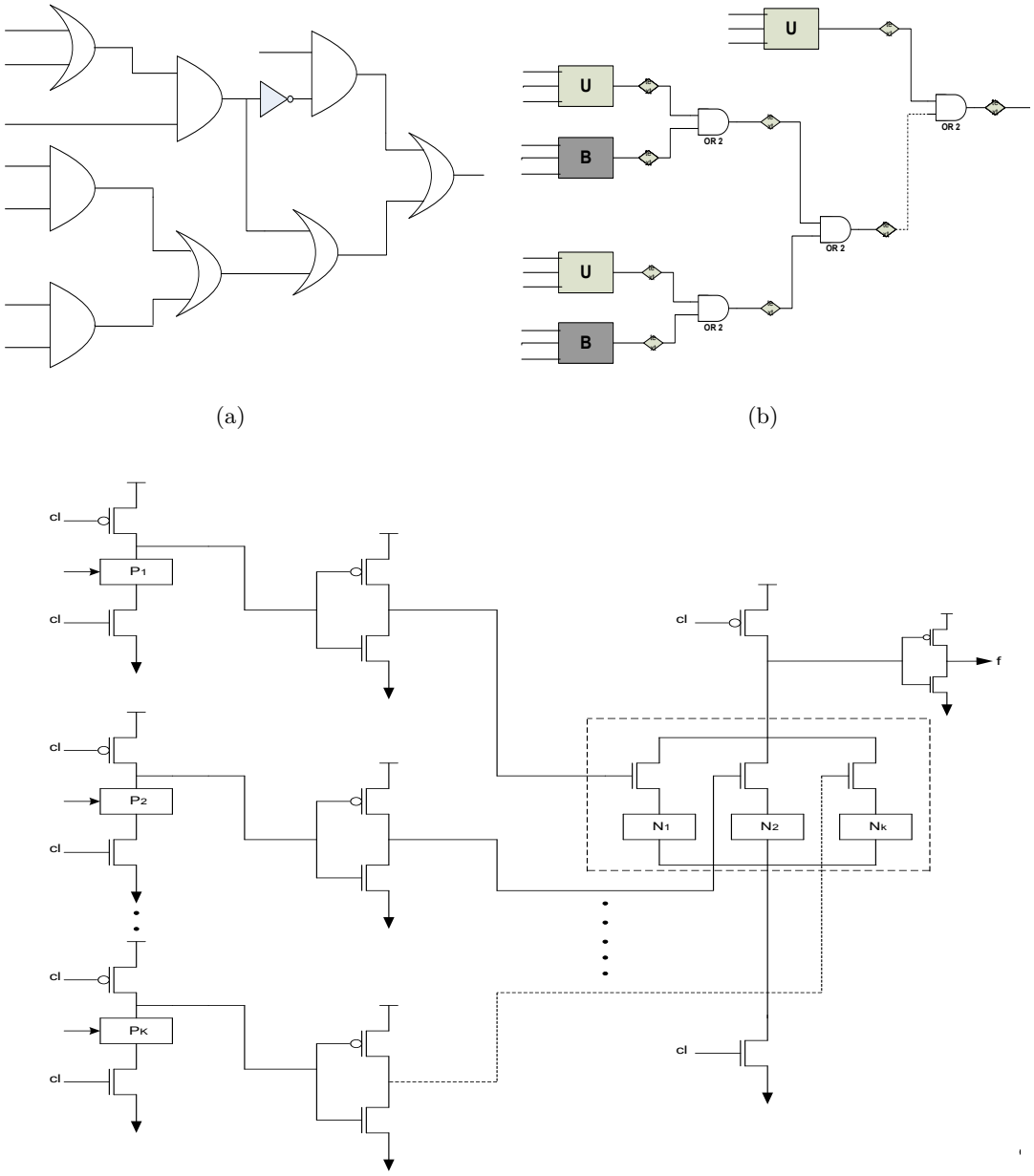


Figure 1.10: (a.) Bubble pushing method, (b.) BDD based method, (c.) Two-level method

an accurate clock gating technique which is based on novel methods of unate decomposition and bubble pushing is desirable.

Few works have been done in the field of decomposing circuits into unate and binate sub blocks. Works related to the issue of mapping Domino logic on-the-fly also exist in the

literature. Our survey in the field of clock design for Domino logic reports various works addressing issues that exist in the literature. However, a number of limitations exist in these work, some of them are mentioned below.

1. In the works reported on decomposing Boolean functions using various techniques [57], [54], emphasis was never laid on improving simultaneously speed and power of the overall circuit. Some methods focused on decomposing the circuit into various block but nowhere emphasis was laid on realizing circuit using mixed static-Domino logic [58]. Also, a comparative study of various techniques is very much needed.

2. Various approaches on library free mapping reported in the literature, begin with a NAND based directed acyclic graph (DAG) network [66], [67], [68]. None of the literature considered unate circuits as a base for their approaches. Works which adopted a parameterized library mapping [69], [70] 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-tune cells along critical path for obtaining high performance. The flexibility offered by Domino logic style in designing the individual cells needs to be investigated. Also there is a scope for re-ordering the cells along critical path, which further minimizes delay. Fine tuning these cells along the critical path, without increasing their individual transistor count is in fact another challenging task.

3. Many of the researchers in the field of clock gating focused on sequential circuits only [63], [64], [71]. Since the outputs of a combinational block solely depends on its inputs, the same technique for clock gating cannot hold true for sequential and combinational circuits simultaneously. Majority of these works focused on minimizing the routing length of clock, addressing the slew constraints [72], [63] etc. Hence, there is a need to propose a method which attempts to reduce the redundant switching of gates in Domino circuits. Constant charging and discharging of Domino blocks with every rise in clock pulse motivates researchers to implement a gating technique which can reduce the redundant switching activity. Simultaneously, reducing the redundant switching and keeping a control on logic overhead is going to be a challenging task.

Our work is motivated by the need to address the above stated issues. Based on this motivation, we focus on synthesizing mixed static Domino CMOS circuits. In the following section, we present the major objectives of our research work.

1.3 Objectives of the Thesis

We set the following objectives in our research work.

1. Our main aim is to present a complete methodology for designing a mixed static Domino CMOS logic circuit which consumes low power and gives high performance.

2. Static and Domino logic styles better implement binate and unate logics, respectively. Most often circuits are binate in nature. Hence, our first objective is to devise a decomposition algorithm which decomposes a given Boolean logic into unate and binate sub blocks. We also aim to find an optimum decomposition such that the performance of overall circuit is optimum.

3. The obtained unate and binate blocks can be mapped using Domino and static logic styles, respectively. The Domino logic style would be mapped using an on-the-fly mapping technique. Re-ordering cells along the critical path would significantly improve the delay of a particular circuit. Our next objective is to formulate an on-the-fly mapping approach based on cell re-ordering such that delay of the circuit is improved at a minimum area penalty.

4. Clock is the most active signal driving the Domino block. During the operation of the circuit, some portions of the Domino block remain redundant for significant amount of time. Hence, our objective would be to identify such possible redundant portions and systematically block them using a clock gating technique. We aim to use sub graph matching techniques to meet this objective.

1.4 Organization of the Thesis

The rest of this thesis is organized into chapters as follows.

Chapter 2 provides a literature survey of various existing works related to area of research. In our research, we focus on topics like unate decomposition of Boolean functions, on-the-fly mapping of Domino circuits and clock gating techniques. We survey the works on these topics.

Chapter 3 presents our proposed approach to decompose the Boolean logics using a unate decomposition algorithm. We first discuss few concepts and definitions which we use in describing our methodology. Also, we postulate few lemmas which act as basis in our approach. Next, we present the unate decomposition algorithm which is used to decompose a circuit into unate and binate blocks. Further, we perform an optimization of these blocks so that overall power, area and delay of the circuits are optimum. We prove that our optimization problem is NP-complete. Finally, we validate our approach with standard benchmark circuits and present the experimental results along with a comparative study.

Chapter 4 presents a method for on-the-fly mapping of Domino circuits using a cell re-ordering technique. We first state few definitions and lemmas used to describe our methodology. Next, we present an approach to map Domino circuits on-the-fly. Further, we perform fine tuning of delay path by selective re-ordering of cells. We also present an approach for optimum cell-reordering. We present the experimental results we have obtained along with a comparative study with existing approaches.

Chapter 5 presents a pattern matching based clock gating approach for the Domino block of mixed CMOS circuit. First we present some definitions which form the basis of our approach. Next, we provide a method to obtain the *favorable gate patterns*. We, present an approach based on pattern recognition for mapping the gate patterns to the original circuit. Finally, we try to minimize the clock gating overhead, keeping power savings maximum using an optimization technique. We present the results obtained using the experimental studies we have conducted and compare them with the existing related work.

Chapter 6 concludes the thesis with critical analysis of our work. We also report the important contributions made by our work. Finally, we discuss the possible future extension to our work.

Chapter 2

Survey of Existing Work

In this chapter, we present a survey of the literature which are related to our research work. This chapter is organized as follows. In Section 2.1 various approaches existing for unate decomposition of Boolean functions are reviewed. Section 2.2 presents the work related to mapping of Domino cells using various techniques. Various approaches for performing clock gating are reported in Section 2.3. Finally, we summarize the chapter in Section 2.4.

2.1 Unate Decomposition of Boolean Functions

A number of work exist in the literature on decomposing a Boolean function with respect to various criteria [34, 73–85]. In the following we present some key work related to unate decomposition of Boolean functions.

The approach of bubble pushing algorithm is well known for realizing unate networks. Prasad et al. in his work [54] follow the bubble pushing method in order to convert a given arbitrary Boolean circuit into a positive unate circuit. This approach is a generalized version of De Morgan's laws. In this approach, the traversal of the circuit is done from output nodes to input nodes. During the traversal, the approach tries to make every node it encounters into a unate node. Each input of a node is considered for determining its unateness. If a node is binate with respect to a variable, a corresponding new variable is created, which is the complement of the original variable. The new variable is substituted in place of the old, making the node unate. Such a unate circuit will have some internal nodes which are complement of some other internal nodes and some primary inputs which

are complement of some other primary inputs. This information is used further in *don't care optimization step*. The relationship between the primary input and its complement is characterized interms of satisfiability don't cares (SDCs). SDCs are set of outputs of a node whose corresponding input vectors may never occur at the input of the node [6]. These fall in the category of internal don't care set. Such SDC relationships act as don't cares for each output which help in further simplification of the node. During this process some of the complements cannot be pushed to the primary outputs. This is because application of Demorgan's laws will change the type of node and also its inputs. However those inputs may not be exclusively serving one particular node. Hence the inputs to other nodes get affected making the back propagation of complements troublesome. This situation is also termed as *trapped inverter* problem. The entire unate circuit can be realized using Domino logic and the trapped inverters must be realized using static logic.

Samanta et al. [57] developed a two-level decomposition method for realizing a given circuit using pure unate logic. Their approach follows a drastically different method. It starts with description of a Boolean function given in PLA format. The large scale circuits are partitioned into sub graphs, each having not more than 15-input variables. Hence partition is done, in order to carry out the unate decomposition efficiently. As this unate decomposition step is based on (minterm) canonical representation of Boolean functions, its complexity increases exponentially with the number of input variables. In the unate decomposition step, each sub-function is expressed in terms of only positive and negative unate functions, which directly maps to a two-level Domino or no-race (NORA) networks. However, direct realization of this two-level network leads to MOS networks with large number of series/parallel transistors in each cell. To overcome this problem, they performed a multilevel decomposition of each unate sub-function. The multilevel decomposition step produces final netlist of the synthesized network satisfying the length and width constraints required for realizing high-performance circuits.

A binary decision diagram (BDD) based approach for decomposing incompletely specified Boolean functions into unate, binate sub blocks is presented by Jacob et al. in [58]. This approach aimed at improving the quality of circuit by restructuring the netlist. Various steps in the approach included *cover minimization, phase assignment, selection of largest unate component* etc. During the cover minimization process both the internal

and external don't cares are taken into consideration. The initial cube cover is processed as long as its cardinality reaches under certain limit. Simultaneously selection of a set of cubes is done, which when considered form a unate cover. Since, the initial cover is binate, extraction of such unate cube subsets is possible. While finding the largest unate subset a greedy computation is used. The cube subset which can be unate with the largest number of cubes in the cover is identified. The less number of literals a cube has, the more is the chance of being part of unate cube. Selection of the largest cubes is done based on zero suppressed binary decision diagram (ZBDD). This method took a cube cover and return the cubes that have largest size. The whole extraction procedure is applied iteratively. It continues till the cover being processed reaches certain limit. The last block thus obtained during the iterative process may end up being binate. The cardinality of the last extracted component tends to be small compared to the rest of the obtained blocks.

A candidate block based approach, which is a further refinement of bubble pushing algorithm for unate decomposition of Boolean function is presented by Parmar et al. in [55]. This approach aimed to realize the static and Domino blocks of the decomposed Boolean function such that overall performance of the circuit can be improved. In the first part of their work, they considered the timing constraints that are to be observed by the Domino and static input signals. They have redefined the setup and hold constraints seeing from the perspective of both Domino and static gates. Both the signals coming from Domino gates and static gates, going to as input to Domino gate are kept under this purview. Setup constraints are framed in a way such that before the end of the evaluation cycle the correct outputs are to be evaluated. Also, it is taken into consideration that the data input is to be pre-charged before beginning of the evaluation of the next cycle. Another constraint was framed that the signal must be held till the outputs are settled. For static input signals it is verified that the signal must be glitch free and the output must be available before the evaluation phase is started. After defining the timing constraints the work focused on balancing the path delays such that the inputs can arrive at same time. This is done by introducing additional pass transistor logic elements. The work followed the bubble pushing algorithm [54] to make the circuit unate. However, in order to address the trapped inverters that arise out of implementing the algorithm, they used a candidate block based approach. An AND2 gate cascaded with an inverter is considered as a candidate

block. Other topologies having trapped inverters rechanged to this particular topology using De Morgan's transformations. It is ensured that one of the input of the candidate block come from the static inverter of Domino gate while the other comes from the dynamic gate of the Domino gate. In this way, the trapped inverters are successfully pushed to the primary ports. Since the candidate blocks receive inputs from various sources the delay balancing elements are added accordingly such that the timing issue will not arise. The work also discussed steps which lead to further optimization of candidate blocks, if present consequently. However, such optimization is not possible in presence of an intermediary fanout.

In Table 2.1, we summarize various unate decomposition methods we have discussed and present their key features.

2.2 Technology Mapping Techniques for Boolean Functions

A number of work exist in the literature on efficient mapping of Boolean functions using cell libraries [86–94]. Often these libraries are predefined, yet in some cases, the libraries are updated on-the-fly. The advantages of using Domino logic can be better tapped by using an on-the-fly mapping technique. In the following, we present some important work which are based on the on-the-fly mapping.

A method for mapping the circuit into parameterized library is proposed in work Zhao et al. [69], which exploits the special features of the Domino logic. The starting point of the parameterized tree covering algorithm is 2-input AND/OR trees. At each node of the tree the optimal sub solutions for various height (maximum number of transistors in

Table 2.1: Works related to unate decomposition of Boolean logic

Method	Decomposition method	Key features
Prasad et al. [54]	Bubble pushing	Don't care optimization, De Morgan's laws
Parmar et al. [55]	Trapped inverter elimination	Identifying candidate blocks, Defining timing constraints
Jacob et al. [58]	BDD and ZBDD	Cover minimization, unate block extraction
Samanta et al. [57]	Two-level realization	Positive, negative unate blocks, pure unate

series), width (maximum number of transistor chains in parallel) combinations are stored. The cost of accumulated fan-in cones along with the current Domino segment are stored at each and every node. In order to compute the height and width of a sub solution at a parent node, the individual children node are used. Since the Domino n-block consists of series and parallel chains, only AND and OR type of nodes are considered. A mapping algorithm is defined for the nodes, which computed the overall cost of mapping procedure. They have derived the cost functions based on the area cost model and the delay cost model. A complexity analysis of this parameterized mapping is also presented in the work. Later, this work extended to a directed acyclic graph (DAG), which allows the overlap of two mapped cells. A correct cost estimation step is performed in this case so as to mitigate the penalty imposed due to overlapping of mapped logic. Further in this work, a method for mapping the dual-rail logic cones for AND/OR/XOR/XNOR is suggested. It is done based on a dual monotone mapping algorithm, where a signal and its complement are treated as two separate signals. An analysis of the output phase assignment problem and its influence on the implementation cost is also presented in this work. The phase assignment problem is characterized as a 0-1 integer linear programming problem and a linear programming package was used to handle this pre-processing step.

A technique for mapping a random logic Domino gate network is presented in [95]. In contrast to the regular bulk CMOS, the implementation technology used in this work is silicon on insulator (SOI). Various methods, used in order to reduce the *parasitic bipolar effect* (PBE) are discussed in this work. PBE is a common factor in SOI devices and could lead to erroneous output. An algorithm aiming at mapping Domino logic network, eliminating PBE is elaborated in this work. Transformation of the transistor stacks, re-ordering of gate structure to reduce susceptibility to PBE are also done. Insertion of PMOS pre-discharge transistors helped to further reduce the PBE effect. In order to obtain the unate network for Domino mapping, authors have followed a simple bubble pushing algorithm. The trapped inverters are addressed by performing the logic duplication. The complexity of mapping algorithm is thoroughly reduced in order to cope for the loss of details occurred due to restructuring. Potential discharge transistors and presence of parallel branch at the bottom of cell structure is monitored at every step of the mapping algorithm. The processing of nodes present in the circuit is done in a topological fashion so that the

inputs to the current node are already processed. The algorithm minimized the mapping cost by minimizing the number of transistors, including the pre-discharge transistors. Various steps carried through out the work ensured that the body voltage of the SOI device is low and the PBE is not triggered. The affect of timing hysteresis caused due to variation in body voltage is considerably reduced by the proposed approach. In order to further reduce the PBE, circuit timing behavior is made predictable by narrowing the range of permissible voltages for the body.

A mapping and synthesis scheme for Domino logic circuits is presented by Cao et al. [96] which aims at minimizing the duplication cost of internal inverter with some timing constraints. Also, the mapping technique uses realization of complex Domino gates. The work explored the possibility of retaining the trapped inverters, in reconvergent paths of Domino logic, by introducing an early late delay difference bound (ELDDB). They have defined this as difference between the *latest late edge* and *latest early edge* of the inputs which helps in preventing the false discharge of the dynamic nodes. Initially the logic duplication minimization is considered for circuits made of simple gates. The duplication cost is reduced by transforming all the convergent paths into AND gates with the help of bubble pushing. This avoidance of logic duplication cost lead to incompatible phase assignment and hence they framed an output phase assignment problem. Different approaches were suggested in this work for mapping the complex Domino gates and complex Domino candidate gates. They have chosen to minimize the layout area as their major objective. Authors have used "AND-OR-INV" DAG representation of the circuit in order to find a candidate primitive which later transforms into a candidate gate. They have also formulated the attributes of the candidate gate which are mainly dependent on the transitive fan-ins and delay difference between gates (i.e. ELDDB). A dynamic programming based approach is used for mapping the complex gates on re convergent paths. Additional care about transitive fan-ins is taken care such that ELDDB constraint will not be critical for the complex candidate gate. Finally the robustness of Domino gates is improved by performing logic optimization as post-layout step.

A bin packaging based mapping algorithm is presented by Yoshikawa et al. [59], which aims in reducing both levels of the Domino circuit and its complexity. They aimed to decrease the total number of transistors used in the design so as to decrease the complexity

of cell layout. During their mapping approach care is taken such that the noise immunity of the Domino circuit is maintained. As a first step trees are extracted from the input circuit. This is done by decomposing the tree from the input side by the process of fan-in ordering. During this step two fan-in nodes are connected to the new generated two input node which is having less number of levels. This process is repeated until the entire tree consists of two input nodes. Next, this the bin packaging algorithm is carried out from the input side. A constraint based packing of nodes having a large number of levels is done at each step of the algorithm. The authors proved that the complexity of the Domino primitive cell increases only if the number of levels decrease. In order to simplify the cell layout further, complexity reduction method is proposed in this work. Initially all the levels of the critical paths are estimated. This is done while keeping the gates on the critical paths fixed. After this the trees from the non-critical paths are selected and are mapped under the defined constraints. The mapping is taken into consideration only if the number of levels in the critical paths are not increased in the revised circuit. This helped in decreasing the complexity of non-critical paths without imposing penalty on the critical path.

A library free mapping method for DAG description of circuit is suggested by Marques et al. [66]. This work combined minimizing the total number of series transistors along with a wave font mapping technique. Since the configuration of the transistors to be used are obtained by a cell generation tool, the algorithm is inherently library free. The proposed library less wave front mapping algorithm uses a matching and covering routines. A two input decomposition of Boolean function is considered. The matching generation window relies on the width of the wave front that is used, where as the covering function is dependent on the lower bound on the number of serial transistors. In order to improve the speed of the covering algorithm, the work used routines similar to those mentioned in *Espresso-Signature*. The reduction in the number of series transistors lead to the decrease in logical effort of the cells. This in turn helped in decreasing the delay of the circuit and meeting the mapping objectives. The proposed method is applicable to both static CMOS, as well as Domino logic based circuits. In case of static CMOS this method aimed at having smaller pull-up block than the pull down block. Changing the polarity of the inputs is employed in order to exchange the logic between the pull-up and pull-down blocks. The matching algorithm

is constrained by the number of variables or the number of literals in consideration. Since the match generation process is in with in the wave width, it performed its search across the fan outs. In order to save on the area aspect the authors have suggested to limit the matching function to fan-out free region.

With an aim to reduce the area-delay product of a circuit, a library free mapping approach based on computing logic effort is presented by Pullerits et al. [70]. The computation of the logical effort helped in choosing the minimum delay architecture regardless of the technology process used. The final covering is made independent of the initial covering by performing a combination of structural and Boolean matching. The entire procedure is followed in three steps namely cell generation, matching and covering. Using the maximum possible serial and parallel transistors as inputs the cell generation tool will generate various combinations of gates. A hash table is also generated based on indexing, for quick look up of these gates. In order to optimally match the graphs a combination of structural traversing and Boolean matching is performed. The independence of the mapped circuit from the initial decomposition is obtained due to Boolean matching. Traversal of the tree is done from the root to the leaf node and from left to the right child in order to obtain the matching gates. During the traversal, at each node all possible matches are explored from that point to its primary inputs. The covering problem is addressed with aim to reduce the delay and silicon area. The minimization of delay in the covering step is achieved by careful selection which reduces the electrical effort and parasitic capacitances. A recursive tree traversal algorithm is employed for covering problem which traverses from leaf node and terminate at root node. Each match iterated is fixed in place at the current node being analyzed. The completion of the recursive call, leads to the best match for that node, and best critical path delay from the inputs to the subject graph.

An approach for library free mapping based on KL cuts (K input, L output circuit cut) is presented by Martinello et al. [67]. This primarily targets mapping of multioutput cells which significantly reduce the circuit area. FPGAs that have multioutput logic under test (LUT) are also targeted in this work. A covering algorithm is presented in this work which tries to map an entire circuit using KL cuts. The algorithm used a greedy approach for finding the potential local maxima of mapping. The covering of the cuts is carried out

from inputs to outputs. At each iteration the largest possible solution is chosen. Also, the number of KL cuts are eliminated from the solution space. The iterations are repeated until the circuit is fully covered. The work, defined two effort levels in defining the spread of KL cuts. The high effort level leads to the presence of all the KL-cuts in the solution space. Using the low effort level only the global and local KL cuts are made available to the covering algorithm. It is observed that a good fraction of KL-cuts are multiple output in nature. Also, it is found that, two single output cuts can be implemented by a single multiple output LUT. However this can be done as long as the sum of the number of inputs of these cuts is no larger than the LUT number of inputs. Most of the single output KL cuts found have few inputs, which allows its combination leading to a high utilization of multiple output LUTs. An approach for computing KL cuts with an unbounded K is also discussed in this work.

A mapping technique MIXSyn is presented by Amaru et al. [68], which targets circuits that are dominated by mixed XOR, AND/OR gates. This approach aimed to realize smaller and faster devices namely application specific integrated chips (ASICS). The method followed a two-step procedure. Initially, a two step optimization enabled a selective and distinct manipulation of AND/OR - XOR intensive portions is performed. The optimization quality of the procedure is improved by adding the external don't cares of the conditions. Later, an area-oriented library free mapping is presented in the work. This aimed to reduce the overall circuit area with a minimum computational effort. In order to support both AND/OR and XOR functions, a subject graph is used with an enhanced base function. The mapping algorithm included three major steps namely pre-decomposition gate-assignment and gate building. During the first step, a subject graph a 2-bounded DAG, into a forest of trees. The generated trees have subset of primary inputs at the leaves and nodes representing a logic function form the base function set. Once, the forest of the trees are generated a tree covering method is implemented which generates subtree with at most m inputs (m being maximum gate fan-in). Internal inverters are propagated to the leaves and standard rules for cell construction are implemented. Finally, the Boolean binary sub trees that have and additional one to one correspondence with logic gates are generated using the greedy algorithm. The tree decomposition is driven in way that it resulted in a set of minimum

number of gates.

In the Table 2.2, we summarize various works on the library-free mapping and present their key features that we have discussed .

Table 2.2: Works Related to Library Free Mapping

Work	Mapping technique	Key features
Sapatnekar et al. [69]	Parameterized library mapping	Area, delay cost functions, dual-monotone mapping, output phase assignment
Karandikar et al. [95]	Mapping for SOI devices	Addressing PBE, logic restructuring
Cao et al. [96]	Complex domino gate	Minimizing logic duplication, identifying re-convergent paths, timing constraints (ELDDB)
Yoshikawa et al. [59]	Bin packaging method	Fan-in re-ordering, reducing layout complexity, minimizing levels of Domino cells
Marques et al. [66]	Library-free wave front mapping	Applicable for both static and Domino logic, minimizing number of serial transistors, reducing structural bias
Pullerits et al. [70]	Library-free mapping	Estimation of logical effort, electrical effort, recursive tree traversal
Martinello et al. [67]	KL cuts	Multiple Output cells, Area minimization, effort level
Luca et al. [68]	Library-free Mapping	Logic manipulation, external don't care, gate assignment, building

2.3 Various Clock Gating Approaches

Clock gating is an important technique often used in minimizing power dissipation of Boolean functions [14,97–106]. Circuits using Domino logic heavily depend on this approach in minimizing their overall power dissipation. Below we present some major works related to the research area of clock gating.

A deterministic clock gating (DCG) methodology as an alternative for pipeline balancing (PLB) is presented by Li et al. [107]. A key observation that for many of the stages in a modern pipeline, a circuit block's usage in a specific cycle in the near future is deterministically known a few cycles ahead of time, lead to the formulation of proposed approach. The DCG method gave no performance loss and no loss opportunity since the

usage of block is well known in advance. The finer granularity of the DCG technique (few cycles) offered a lot of flexibility in implementing the gating. The proposed DCG has not taken any heuristics into account, while performing gating. The work analyzed various possible opportunities to implement DCG in a super scalar architecture. The implementation of DCG is done on execution units, pipeline latches, D-cache word-line decoder and result bus driver. The selection logic in a conventional issue queue not only selects instructions to be issued based on execution unit availability, but also matches instructions to execution unit. The proposed approach leverages the selection logic, and provides information about the execution units that will remain unused and which are later clock-gated. The pipeline latches are clock gated at the end of rename, register read, execute, memory and write back stages. In case of "rename", the number of clock-gated latches in any cycle can be determined from the decode stage in the previous cycle. Similar to the gating of pipeline latches, D-cache word line decoders are clock-gated using the load/store issue information. Here, clock gating can be implemented directly to the result bus drivers. In case of result bus, when it is not used in a particular clock cycle, clock gate signal prevents C_L from switching, and reduces power. In this fashion the DCG procedure clock gates many unused modules by determining their behavior few cycles ahead of time.

A fine grain clock gating of dynamic logic circuits at circuit level granularity is presented by Nilanjan et al. [65]. The proposed technique also improved switching power by preventing redundant computations. The approach is further extended to Domino/skewed logic styles based on Shannon expansion. It dynamically identified idle parts of logic and applied clock gating to them to reduce power in the active mode of operation. The approach performed Shannon decomposition of a Boolean function and identified at an instant only one cofactor performs useful computations. The other co-factors perform redundant computations. Using a chosen variable, the AND gates used for clock gating the co-factors of the Boolean function are controlled. While doing this both the redundant computations and clock power are saved. It is ensured that all the operations are carried out in active mode of the circuit. The procedure is extended hierarchically for multiple levels of expansion, satisfying the area, timing constraints. Since the logic between the co-factors is shared and not gated, it kept on through out. The choice of the control variable is guided by the objective of minimizing total power in active mode. With a target of maximizing the logic in gate cofactors the

control variable is chosen. This minimized the shared logic which is active throughout the process and cannot be clock-gated. The control variable selection method is also easily extended to multi-output circuits by choosing a common control variable for all outputs at each level of expansion.

A method for evaluating the field programmable gate array (FPGA) clock network architectures having an in built clock gating compatibility is presented by Safeen et al. [108]. This approach also presents a flexible routing algorithm that can operate at various gating granularities. Gating at time, various device regions having various clock loads is also done in this work. Initially some novel clock gating architectures are presented in this work. They aim to make some minor hardware changes, whereby a subset of clock signals can be controlled by an enable signal. A broad range of clock gating architectures with various levels of granularity, within clock distribution frameworks that resemble in commercial chips are considered in this work. Different main gating architectures discussed in the work are : namely, no clock gating, region, column based coarse and fine grain clock gating (FG_Region, FG_Coulmn, CG_Region, CG_Coulmn). The gating options are analogous: FG_Region permits gating where clock signals enter sub-regions; FG_COLUMN permits gating where the horizontal half-spines in sub-regions connect to vertical quarter-spines. The vice-versa is defined for CG_Region, CG_column. For a given placement, the clock power term is computed as the sum of the power consumed by each clock signal in the design. The structured nature of the clock network helped to rapidly compute a clock routing during placement, making the estimation of clock capacitance and power easier. For the region-based clock enable architecture, gating is only available at the entry points of regions and therefore, the gated clocks used on columns of a region must also be routed on horizontal spines in the region. The added flexibility provided by gating capability at column entry simplifies the clock power computation. The clock routing and clock power saving are separately estimated for region based and column based architectures. All performed computations and comparisons of clock power consumption are done relative to a baseline architecture.

A novel approach for low power gated clock tree design is presented by Shen et al. [71]. It aimed to use fully both the logical and physical information between registers.

Various steps of the procedure include register placement, gated clock tree construction and incremental placement. As a first part, the register clustering and pulling the registers closely is implemented. This is done in order to reduce the wire length and clock tree power. This can affect the wire length of some other signal and area of the net. Hence, a net weighting method is used which took into account of both slack and the net's switching activity. Next, reduction of the redundant gating logics is performed with an improved algorithm that implements gate insertion and zero skew clock routing simultaneously. This algorithm works without assuming a gate is inserted before each leaf (or register) at the initial stage. With the help of bottom up node merging a zero skew clock routing is also discussed. The Elmore delay model is used as reference in carrying out the skew merging. Placement issues arise, since the gating logic is inserted in the gated clock tree after the placement. In order to carry out the incremental placement, authors have used a design database which uses the information of gating logic and gated clock tree. It is observed that the insertion of the gating logics for each internal node in the clock tree prevented any changes on the clock net and the clock branching. The trade-off between clock tree and signal nets are thoroughly analyzed in this work.

A technique to enlarge the gating cover by gating more number of devices is presented in Lin et al. [63]. This technique uses an interpolation method in a satisfiability (SAT) based algorithm. The clock gating signals are chosen as SAT proofs. An approach on modeling the clock gating as a sat problem is presented. The CRAIG interpolation technique [64], which is used to construct revised gating signals is elaborated in this work. The clock gating algorithm followed a three step process. First, valid gating candidates are identified. A net-register pair is a valid gating candidate, if the net's function can be formally proved as a gating condition of the register. A logic simulation on the circuit under test is done in order to filter out the invalid net-register pairs. Next, a satisfiability check is performed on the remaining net-register pairs, by the SAT engine. After this based on interpolation techniques various types of gating candidates are found (the *Itype*, *Gtype*). Always the interpolants and the additional logic are verified, so that the timing constraints are not violated. Finally, the power savings for each valid gating candidate is computed. The gating signals that gate most number of registers, with highest power savings are identified

and are selected in order to maximize power savings. The actual power saving took the overlap affect into consideration, since different signals can gate a set of registers.

A review of various existing clock gating techniques is presented by Khaturia et al. in [109]. The work also analyzed these techniques by observing the simulations of each of these techniques. A two-input AND gate is used as clock gating logic where on of its input is the clock signal and the other is a control signal meant to control the output. The simulation observations demonstrated that the hazards at the enable can pass on to the gated clock. For actions that are to be performed on the positive edge of the global clock a NOR based clock gating technique is used. However, the similar problem of hazards and glitches arise in this type of gating and lead to erroneous outputs. The hazards at the enable signal are avoided by using a latch-based AND clock gating technique where, the hazards of the enable are blocked by the latch. However, the delay occurred due to insertion of additional logic must be taken into account during timing verification and the affect of glitch is not nullified. The same is the case with latch-based NOR clock gating where the hazards problem is overcome but the problem due to glitch persists. A multiplexer (MUX) based clock gating is discussed where a MUX is used close to an open feed back loop around a basic D-type flip-flop under the control of an enable signal. Though the resulting circuit is robust and follows the rules of synchronous design, an additional MUX per bit lead to increase in power dissipation. Finally, the work proposed a new gated clock generation circuit based on a negative and positive latch. It is ensured that the controlling device's clock is off even when the target device's clock is on/off. This way power is saved by avoiding unnecessary switching. In this technique, both glitches and hazards problem is resolved.

In order to reduce the dynamic power usage, two novel approaches, power aware and power slew aware gated clock tree synthesis (PACTS, PSACTS) approaches are presented by Lu et al. in his work [72]. These synthesizers are proposed with a zero skew based on Elmore delay model. In this work, the clock tree is simultaneously constructed with the inclusion of clock gates thus avoiding the slew changes. The method begins with construction of a binary clock tree in a bottom up course. The standard slew constraints are considered in the PACTS synthesizer. A simultaneous gate/buffer insertion method is proposed in order to reduce power and construct a binary tree topology. A nearest neighbor

selection method is used in construction of the tree topology. This lead to the effective control of the enable signals and the wire length. The activity of the internal nodes are simultaneously monitored and are updated. The run time of the program is drastically low since reduction in complexity. The PSACTS synthesizer included a stricter constraint that included a hard limit on clock slew. A slew-oriented lookup table is introduced, to provide the information of driving ability during the buffer and gate insertion. The complexity analysis of the proposed work is thoroughly done by performing activity computation, transition probability computation, instruction stream input and neighbor updating. The experimental results also show that the slew rate limitation is satisfied with a small clock skew from SPICE estimation. The gate insertion method applied in PACTS is consistent with the result of the pairing cost, which can further enhance the performance of the topology generation.

An approach for synthesizing clock gating conditions automatically is presented by Hurst et al. [64], which attempts to minimize the net list perturbation. The proposed method is both timing and physical aware. Also, this method is scalable, utilizes simulation and satisfiability tests and avoided the need of symbolic representation. The approach modeled the circuit as a hypergraph whose nodes are either single bit registers are single output combinational logic nodes. The multiple clock domain case is addressed by treating each register group separately. The gating algorithm aimed at finding incomplete gating conditions for each registers as a disjunction of literals. After that, the candidate identification step is carried out in which a set of candidate literals are identified in terms of gating signals for each register. Later, these are narrowed by following timing, physical and structural constraints. Next, the candidate pruning step is carried out where each literal is ensured that it is consistent with the gating condition. If any literal is found violating the condition, they are immediately removed from consideration. Later a satisfiability solver is used in order to prove the correctness of each of these candidates using an incremental mode. The learned clauses in the SAT problem are stored in order to speed up the future runs. The problem now is viewed as the weighted maximum set cover problem, where the weight of each element set is exactly its net contribution. A greedy addition heuristic is used in order to solve the problem. After selecting the subset of candidate sets, each of these is used to drive a clock gate and produce a single gated clock signal. Finally, as a

Table 2.3: Works related to clock gating of Boolean logic

Method	Gating technique	Key features
Hai Li et al. [107]	Deterministic gating	Pipeline latch, execution units, result bus
Nilanjan et al. [65]	Fine grain gating	Shannon expansion, shared logic, selection of control variable
Safeen et al. [108]	Gating for FPGA	Region, column based architecture, coarse grain, fine grain
Shen et al. [71]	Activity and register aware gated clock tree design	Gated clock tree, incremental placement, zero skew
Lin et al. [63]	SAT based clock gating	CRAIG interpolation, gating candidate, power optimization, SAT proof
Khaturia et al. [109]	gated clock generation circuit	Overcoming hazards and glitches, MUX based, latch based clock Gating
Lu et al. [72]	PACTS, PSACTS	Slew rate constraint, concurrent gate insertion, slew table construction
Hurst et al. [64]	Automatic generation of gating signal	Candidate pruning, candidate Proving, maximum gating condition, don't care based optimization

part of post gating optimization, observability don't cares are used in order to minimize the logic implementation of the next state of the function.

In Table 2.3, we summarize various works and their key features on the clock gating that we have discussed.

2.4 Conclusion

Though works have been reported on decomposing Boolean functions using various techniques, major emphasis was never given on improving simultaneously speed and power of the overall circuit. All methods mainly focused on decomposing the circuit but nowhere emphasis was given on realization using mixed static-Domino. Nevertheless, a number of works have been reported in the literature on library-free mapping, most of them begin with a *nand* based DAG network. None of the work 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. Many works have been reported in the literature on implementing clock gating. Nevertheless, they focused on sequential circuits only. Since the outputs of a combinational block solely depends on its inputs, the same technique for clock gating cannot hold true for sequential and combinational circuits simultaneously. Majority of these works focused on minimizing the routing length of clock, addressing the slew constraints etc. Hence, there is a need to propose a method which attempts to reduce the redundant switching of gates in Domino circuits. Constant switching of Domino blocks with every rise in clock pulse motivates us to implement a gating technique which can reduce the redundant switching activity. Simultaneously reducing the redundant switching and keeping a control on logic overhead is yet to be addressed.

Chapter 3

Decomposition of Boolean Logics

An initial study reveals that for a given logic pure Domino circuit is not advantageous in terms of area, power and delay. This is so because, Domino logic style is inherently monotone and yield better result only when the entire logic is realizable with non-inverted gates [8]. In other words, to get the benefit of Domino logic style, better if we apply it to unate part only. This necessitates to decompose a logic into optimum unate and binate parts. This chapter addresses how an initial decomposition of a Boolean logic can be obtained and then optimization of the decomposition.

In this work, we propose an influence based unate decomposition algorithm which decomposes a given circuit into a set of unate and binate components. Later using an optimization technique we balance these two blocks for the optimum performance of overall circuit. Our objective of optimization is to judiciously mix static and Domino logic styles in the same circuit to gain in terms of power and speed simultaneously.

The rest of the chapter is organized as follows. Some basic concepts related to the topic are presented in Section 3.1. Section 3.2 describes our proposed methodology of designing the mixed CMOS Boolean logics. The experimental results and comparison with existing techniques are given in Section 3.3. Finally Section 3.4 concludes the chapter.

3.1 Basic Concepts and Definitions

In this section, we present few basic terminologies which we refer to in our discussion.

Definition 3.1: *State of a function*: For a given Boolean function, the input at a given instant forms its state. For example, a Boolean function having n input variables has 2^n states.

Definition 3.2: *Weight of a state*: Weight of a state is the number of '1's the state has in its binary representation. For example, the binary representation of state 5 is 0101 and its corresponding weight is 2.

Definition 3.3: *Unate function*: A Boolean function is said to be *unate* if it is either *positive unate* or *negative unate* in each of its input variables but not both. A Boolean function which is not *unate* is said to be *binate*. For example, $f_1 = x_1\overline{x_2} + \overline{x_3}x_4$ is unate, where as $f_2 = x_1\overline{x_2} + \overline{x_1}x_4$ is binate.

Definition 3.4: *Positive unate function* : A Boolean function f is said to be *positive unate* in a variable x_i *iff*, $f_{\overline{x_i}} \subseteq f_{x_i}$. For example, the function $x_1x_2 + \overline{x_3}x_4x_5$ is positive unate in x_1, x_2, x_4, x_5.

Definition 3.5: *Negative unate function* : A Boolean function f is said to be *negative unate* in a variable x_i *iff*, $f_{x_i} \subseteq f_{\overline{x_i}}$. For example, the function $x_1x_2 + \overline{x_3}x_4x_5$ is negative unate in x_3.

A Boolean function $f : B^n \rightarrow B$ with n input variables is said to be *completely specified* when the response of the function to all it's 2^n input states is specified.

Definition 3.6: *Partially ordered set (POSET)*: For an n variable Boolean function, *POSET* gives a weight based ordering of all 2^n states. Further, all state pairs having a Hamming distance 1 are connected together. For example, states 4(100) and 5(101) have a Hamming distance 1.

Definition 3.7: *State pair*: A state pair is defined as follows. Two states which have a single transition of a variable, that is, having a Hamming distance 1, form a state pair. For example states 10(1010) − 14(1110), 5(101) − 7(111), 13(1101) − 15(1111) form state pairs

with Hamming distance 1.

Lemma 3.1: For a n variable Boolean function, there are $n+1$ levels in it's *POSET*.

Proof: Each level of *POSET* corresponds to a particular weight of the state. For a state of n variable function there are $n+1$ weights possible. Hence, the number of levels in *POSET* will be $n+1$. For example, a Boolean function of four variables has 16 states. It has one state with weight '0', four states with weight '1', six states with weight '2', four states with weight '3' and one state with weight '4'. These states with five different weights form five different levels in *POSET*.

Definition 3.8: *Type of influence (TI)*: A state pair having Hamming distance 1 represents transition of a particular variable from $0 \rightarrow 1$ or $1 \rightarrow 0$. It can result change in output. Depending on the transitions in output namely $0 \rightarrow 1$, $1 \rightarrow 0$ or no transition, the influence of the state pair is decided as *positive* ($0 \rightarrow 1$), *negative* ($1 \rightarrow 0$) and *neutral* (no transition). This is called *type of influence (TI)*. For example, output of state 4(100) is '1' and output of state 5(101) is '0', then the state pair $4-5$ has a negative influence.

Definition 3.9: *Variable of influence (VI)* : For a given state pair the variable which is causing the transition from one state to another is called *variable of influence (VI)*. For example, VI of state pair $4(100)-5(101)$ is x_0, considering the input variables as x_2, x_1, x_0, respectively.

Definition 3.10: *Conflict state pair*: Two state pairs having same variable of influence but different type of influences, become conflict state pair. For example, 3-11 may have positive influence and 1-9 may have negative influence and both belong to same variable of influence. Thus, they are conflict state pairs to each other.

Lemma 3.2: A variable can include a state pair having either positive or negative influence but not both, if they form a unate set.

Proof: If we choose a state pair of a particular variable with a type of influence (either positive or negative), then the state pair having the opposite influence is corresponding to the complement of the variable. For a function to be unate, all variables of the function must be present either in their true or complemented form, but not both. Hence, state pairs having same type of influence only must be chosen.

3.2 Proposed Methodology

Given a Boolean logic, we are to realize a CMOS circuit, combining both static and dynamic CMOS logic styles. In this section, we present our proposed approach to realize static dynamic mixed CMOS circuits. An overview of our approach is shown in Fig. 3.1.

Our approach can be stated as follows. Suppose, given a circuit C_{init}, whose Boolean function is represented by $f(\mathbf{X})$. $\mathbf{X}$ is the input vector, where $\mathbf{X} = (x_1, x_2, x_3, \ldots, x_{n-1}, x_n)$. Our approach consists of the following tasks.

Initial unate decomposition (IUD): We decompose the Boolean function $f(X) = U(X) \cup B(X)$, where $U(X)$ is a unate function and $B(X)$ is a binate function. We call this decomposition as initial unate decomposition (IUD). $U(X)$ is meant for realizing the circuit with Domino logic where as $B(X)$ is for static logic style.

Optimium unate decomposition (OUD): An optimization of a given unate-binate set is performed in this step. After the initial unate decomposition, we obtain unate and binate sets. Judicious mapping of unate set using Domino logic style, and the rest using static logic style would result in an optimum performance of the circuit. We try to obtain the optimum combination in this step.

Library based technology mapping (LTM): We use L_{stat}, L_{dyn} the static and dynamic gate libraries for mapping the binate and unate sets, respectively. The mapping of the sets using cell libraries also helps us in estimating the power, area and delay of the circuits so realized. In the following, we describe the above mentioned steps in details.

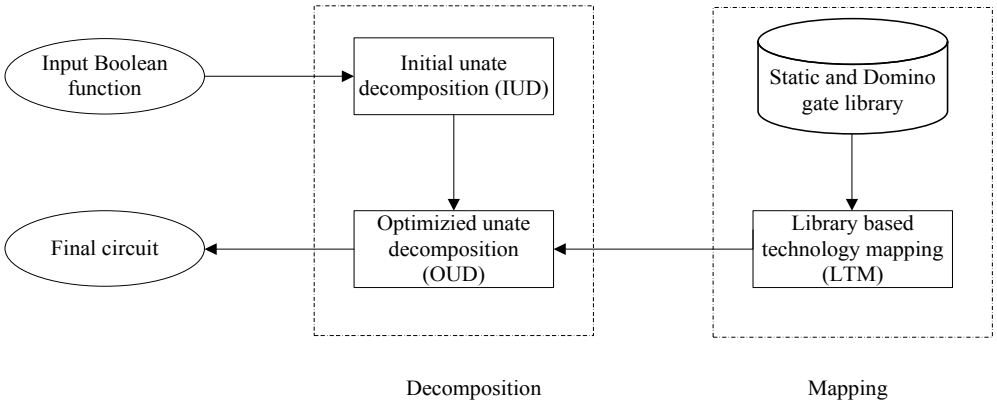


Figure 3.1: Overview of our proposed approach

3.2.1 Initial unate decomposition

We propose an algorithm which takes a completely defined Boolean function as input and decomposes it into unate and binate parts. We term the algorithm as IUD. The various steps of IUD algorithm are shown in the form of a flowchart in Fig. 3.2. The notations used in the flowchart are mentioned below.

In the following, we explain various steps in IUD algorithm with the help of examples. To start with, we consider the following Boolean function as an input.

$$f(x) = \bar{x_3}\bar{x_1} + \bar{x_2}\bar{x_1}\bar{x_0} + x_3\bar{x_2}x_1x_0 + x_3x_2x_1\bar{x_0} \tag{3.1}$$

where the number of input variables $n = 4$.

Step 1: In this step, we obtain the *onset* (**OS**) states of the Boolean function $f(x)$. For example, the **OS** of $f(x)$ is $\{0, 1, 4, 5, 8, 11, 14\}$. Note that realizing the **OS** elements is equivalent to realizing the original Boolean function.

Step 2: We construct a POSET for the **OS**. For the example **OS**, the POSET is shown in Fig. 3.3. This POSET has states with 5 different weights 0, 1, 2, 3, 4. Hence, there are five levels in the POSET (Lemma 1). The elements which belong to the onset are shown using grey ovals and the remaining are in white ovals. The decimal value of the elements is shown adjacent to the ovals.

Notations used in IUD algorithm		
Onset of a Boolean function	:	OS
Unateset at a given instance of algorithm	:	US
Temporary set which stores elements	:	TS
Unmarked variables	:	UV
Marked variables	:	MV
Variable of influence	:	VI
Type of influence	:	TI

Step 3: We categorize each state pair of the *POSET* based on it's **VI** and **TI** into a table called *Influence table*. For example, *Influence table* for $f(x)$ is shown in Table. 3.1. For

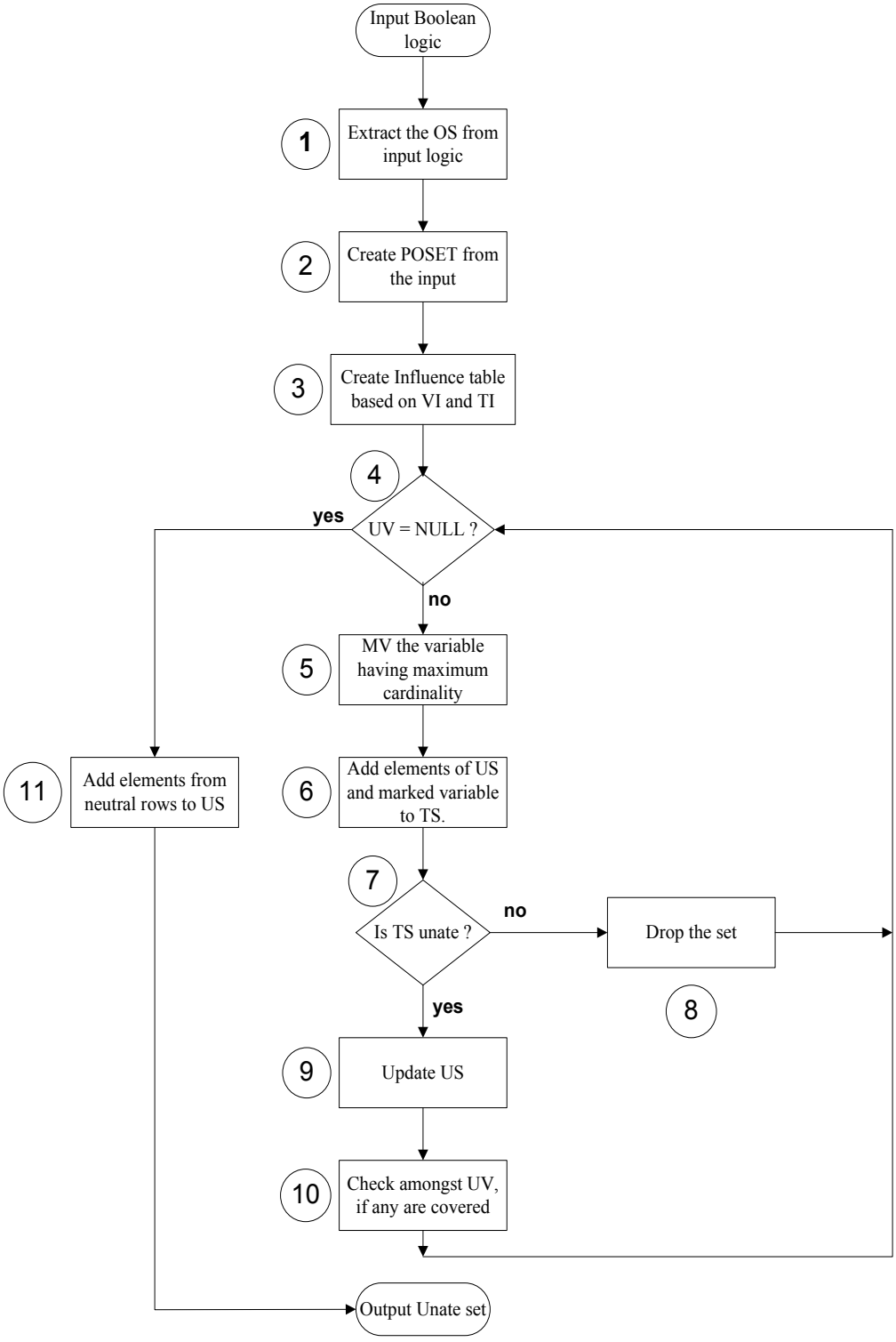


Figure 3.2: Flowchart of IUD algorithm

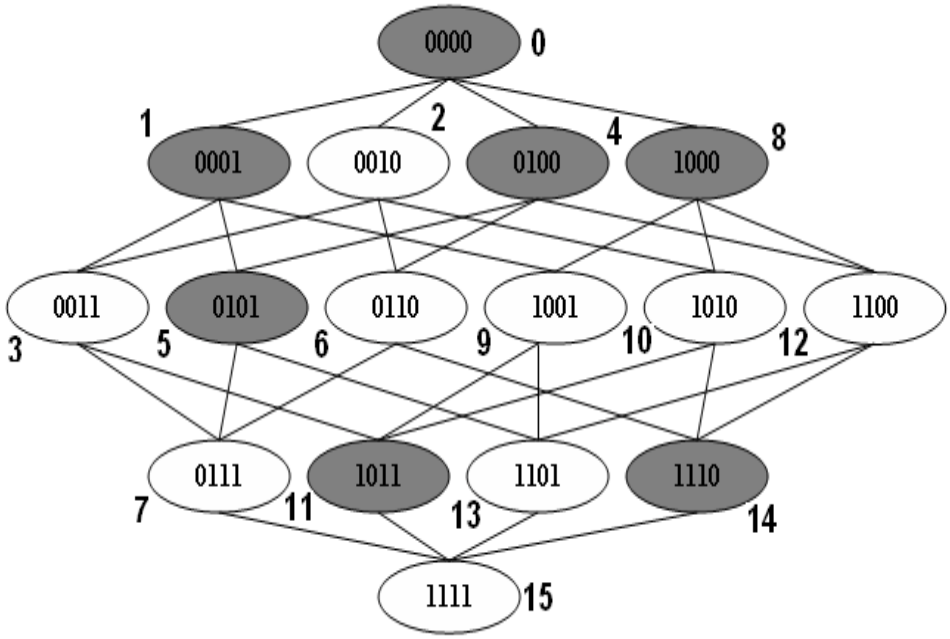


Figure 3.3: POSET of $f(x) = \Sigma(0, 1, 4, 5, 8, 11, 14)$

example, the pair $1 - 9$, the variable which is changing is x_3 and the value of the output is changing from *high* to *low*. Hence, the pair is placed in the row of negative influence and under x_3 column. Like this, all pairs, 32 in this case, are categorized.

Step 4: In this step, we check if there exist any unmarked variables. Initially, all variables in Influence table are unmarked. At the beginning of each iteration a particular variable is marked.

Step 5-6: Cardinality of each set is obtained by counting the number of state pairs belonging to a particular **VI** and **TI**. Initially, the unate set (**US**) and temp set (**TS**) are empty. To begin with, we choose the largest cardinality set of state pairs and include them in our **US**. The corresponding **VI** is marked. In our running example, the maximum cardinality set has **VI** as x_1, **TI** as $-$ and cardinality is 5. The elements are 0, 1, 2, 3, 4, 5, 6, 7, 8, 10 and these form the initial **US**. We mark the variable x_1.

Step 7: Checking for unateness is done in this step. There will not be any conflict state pairs since this is the first set included. Hence, the set is unate in itself.

Step 9: Since the set is unate the **US** is updated to {0, 1, 2, 3, 4, 5, 6, 7, 8, 10}.

Step 10: In this step, we check whether the current **US** spans any other set present the

Table 3.1: Influence table for $f = \Sigma(0, 1, 4, 5, 8, 11, 14)$

Type of influence	xo	x1	x2	x3
$+$	10-11	9-11 12-14	10-14	3-11 6-14
$-$	8-9 14-15	0-2 8-10 1-3 5-7 4-6	8-12 11-15	1-9 4-12 5-13
N	0-1 6-7 2-3 12-13 4-5	13-15	0-4 3-7 7-15 9-13 2-6	0-8 2-10

Influence table. Our current **US** doesn't span any other set. Hence, we continue with our iteration.

Repeat steps 4, 5, 6, 7, 9, 10: Next, we have to choose amongst the remaining **UV**, a set with highest cardinality. With our running example, it is x_3 with negative influence, a cardinality of 3 and state-pairs 1-9, 4-12, 5-13. It's **VI** is marked and it's elements are now added to **TS** along with current **US**. The current **TS** is $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13\}$. In this case, there are no conflict state pairs and hence states 9, 12, 13 are included to the current **US**.

Repeat Steps 4, 5, 6, 7 : We consider variable x_2 and mark it. The states 11, 14 get included in our **TS**.

Step 8: Including states 11, 14 to **TS** which belong to x_2 variable, form conflict pairs (3-11, 6-14) for the already existing negative influence pairs of x_3. This violates the unateness of the current **US**, as per Lemma 1. Hence, the set is dropped. In a similar fashion, the sets under variable x_0 are also dropped.

Step 11: Since there are no unmarked variables left, we proceed to add states from neutral influence set. In the running example, state 15 is added from the neutral influence, giving the final unate set.

In the present example, we have got 14 states in the maximum unate set {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 15}. From the final **US**, we choose our **OS** states. They are 0, 1, 4, 5, 8. These belong to the unate part $U(X)$ of **OS**. The remaining two states {11, 14} form

the binate part $B(X)$ of **OS**. Hence, all elements of **OS** are categorized into either unate or binate set. Both sets together correspond to the realization of considered Boolean function $f(X)$. Thus, we conclude our initial unate decomposition (IUD) as follows,

$$f(X) = U(X) \cup B(X) \tag{3.2}$$

where, the unate set $U(X) = \Sigma(0, 1, 4, 5, 8)$ and the binate set $B(X) = \Sigma(11, 14)$. Here ends the first step of our proposed methodology as shown in Fig. 3.2. Next, we perform optimization of the obtained sets for better overall performance of the circuit. It is mentioned in the following subsection.

3.2.2 Optimization of unate decomposition

In this section, first we state the need for optimizing *IUD*. Then, we state the problem of optimizing an *IUD*, clearly defining the objectives, constraints and design parameters in it. Before proceeding to solve the optimization problem, we prove that the problem is NP-complete. Finally, we suggest a multi-objective Genetic Algorithm based approach to solve the problem.

After obtaining the unate and binate set of a function, it is observed that unate sets are usually large compared to their binate counterparts. As a consequence, mixed CMOS realization may result in circuits heavily biased with Domino logic, which may not be optimum in terms of power, area and delay. In fact, we have the flexibility to choose how much portion of a unate set is to be realized using Domino logic. The remaining part of the unate set along with entire binate set can be realized using static CMOS logic such that final circuit is optimum in terms of power, area and delay. Therefore, there should be a judicial choice to achieve the optimum realization given an *IUD*. We call this problem as COPT (circuit optimization). We formally define the COPT problem in the following. We refer the following notations in our definition.

For a given Boolean function $f(X)$, $\{U(\mathbf{X}), B(\mathbf{X})\}$ are the two sets which are obtained after *IUD*. Our objective is to move some elements from $U(X)$ to $B(X)$ resulting a new decomposition. We call it as optimum unate decomposition *(OUD)*, that is {U(X), B(X)} $\xrightarrow{OUD}$ $U_{opt}(X), B_{opt}(X)$. We consider the following three objective functions to

Notations used in COPT problem		
Boolean function	:	$f(\mathbf{X}) = \{x_1, x_2, \ldots, x_n\}$
Static and dynamic libraries	:	(L_{stat}, L_{dyn})
Onset of the function $f(X)$	:	$OS = \{i_1, i_2, \ldots, i_J\}$
Power, area, delay of static block	:	$P_{stat}, A_{stat}, D_{stat}$
Power, area, delay of dynamic block	:	$P_{dyn}, A_{dyn}, D_{dyn}$
Number of elements in OS	:	J
Number of elements in $U(X)$	:	I
Number of elements in $B(X)$	:	$J - I$
Target values for power, area and delay	:	(P^0, A^0, D^0)
Operators for mapping and optimization	:	MAP, Opt

judge the optimality of *OUD*. Suppose, $\{U_k(X), B_k(X)\}$ denote any decomposition. Then $f_p(U_k(X), B_k(X))$ denotes the power requirement to realize the logic $U_k(X), B_k(X)$ into static Domino mixed VLSI circuit. Similarly, $f_a(U_k(X), B_k(X))$ and $f_d(U_k(X), B_k(X))$ denote the estimation of area and delay, respectively to realize mixed static-Domino VLSI circuits.

We define a decomposition $U_{opt}(X), B_{opt}(X)$ as the *OUD*, if it satisfies the following. Given IUD of a logic f(X) = {U(X),B(X)} :,

$$
\begin{aligned}
OUD\{U_{opt}(X), B_{opt}(X)\} = & minimize[P = f_p(U(\mathbf{X}), B(\mathbf{X}))], \\
& minimize[A = f_a(U(\mathbf{X}), B(\mathbf{X}))], \\
& minimize[D = f_d(U(\mathbf{X}), B(\mathbf{X}))], \\
subject\ to\ & U_{opt}(X) \subseteq U(X), B(X) \subseteq B_{opt}(X), \\
& U_{opt}(X) \cup B_{opt}(X) = U(X) \cup B(X) = OS \\
and\ & P \leq P_0, A \leq A_0, D \leq D_0 \\
for\ some\ constants\ & P_0, A_0, D_0
\end{aligned}
$$

NP completeness of COPT:

Below, we prove that COPT problem is NP-complete. To prove this, we consider the following steps.

1. We show that COPT $\in$ NP.

2. We reduce a standard NP-complete problem L_{SAT} to COPT in polynomial time. That is, $L_{SAT} \preceq_p COPT$ [110].

For a given Boolean function $f(X)$ having OS as *onset*, say $\{U_{opt}(X), B_{opt}(X), P_{opt}, A_{opt}, D_{opt}\}$ is a certificate for COPT. We can easily check whether or not $U_{opt}(X) \subseteq U(X)$, $B(X) \subseteq B_{opt}(X)$, $U_{opt}(X) \cup B_{opt}(X) = U(X) \cup B(X) = OS$, and $(P_{opt}, A_{opt}, D_{opt}) \leq (P_0, A_0, D_0)$. These verifications can be done in a polynomial time. Hence, COPT $\in$ NP.

We choose the *0-1 Knapsack* problem as an NP-complete problem [110]. We prove that *0-1 Knapsack* problem is $\preceq_p$ COPT.

The *0-1 Knapsack* problem can be expressed as

$$C = Maximize \sum_{i=1}^{N} x_i C_i, \;\; where \;\; x_i = [0, 1] \tag{3.3}$$

$$Subject\ to \sum_{i=1}^{N} x_i W_i \leq W_0 \tag{3.4}$$

Here, W_i, C_i are the weight and cost of the i^{th} item and W_0 is the limit on weight of all n items put together. Let us consider J, I, P^x, A^x, d^x be an instance of COPT. The total number of states is represented by J, number of unate states is represented by I, and P^x, A^x, d^x are it's power, area and delay. Let an instance of *Knapsack* problem be N, C, W where the total number of available objects, overall cost and weight of picked up items are represented by N, C and W, respectively. COPT aims to choose, say m states out of the I unate states to implement using Domino logic. The rest $I - m$ states of unate set along with $J - I$ states of binate set will be implemented using static logic. The choosing of m elements is governed by minimizing power, area and delay of the circuit. There is a one-to-one correspondence between *0-1 Knapsack* and COPT problem. In other words, minimization of $\{P, A, D\}$ can be mapped onto the minimization of $\frac{1}{C}$ in *0-1 Knapsack* problem. Similarly, constraints in both optimization problems has one-to-one correspondence. Let us consider a reduction function g which reduces an instance of COPT to an instance of *Knapsack*. We can say that if $m \in I$, $g(m) \in N$, that is, if we can choose m states out of I unate states, then we can choose $g(m)$ number of items from N items in case of *0-1 Knapsack* problem such that the conditions are met. Since the reduction can be done linearly, we then say *0-1 Knapsack* problem is $\preceq_p$ COPT. Hence, COPT problem is NP-complete.

We propose a GA based approach to solve the COPT problem, which is stated in the

following.

GA-based approach: We follow a *non-dominated sorting genetic algorithm (NSGA-II)* [111] to solve the COPT problem. A detailed framework of the NSGA-II is shown in Fig. 3.4. We have followed a binary encoding to define a chromosome in GA.

Given a Boolean function of an n input variables, we encode the corresponding **OS** states in an n bit binary representation. We define a representation for all the states in a unate set. A 0 representation shows that the state to be realized using static logic and 1 representation shows the state to be represented using Domino logic. The length of the chromosome is exactly the same as the number of states in OS of $f(X)$.

Since, there are I number of states in the $U(X)$, a valid chromosome has just I number of 1s or 0s representations and remaining $J-I$ states belonging to $B(X)$ have 0 representation each. To decide a candidate of initial parent population, randomly we choose I binary bits corresponding to I states of the unate set. Each of them can be either 0 or 1. N_p number of such candidates are chosen for initial parent population (see Fig. 3.4). On this initial population, we perform a two point crossover technique [111]. Later we mutate the population with a probability p_m. By doing so we generate N_c number of population which is used in selection process.

After obtaining both parent and child, we evaluate the fitness values for each candidate i belonging to the population. Given a candidate belonging to the population, we can group the states which are to be realized using static logic and Domino logic separately. We obtain the f_p^i, f_a^i, f_d^i for the candidate i. In the same way, fitness values are computed for all candidates in the combined parent and child population.

Using the calculated fitness values, we perform the non-dominated sorting of entire parent and child population. According to Coello et al. [112], we consider a vector $\overline{U} = (U_1, U_2, \ldots, U_k)$ dominates a vector $\overline{V} = (V_1, V_2, \ldots, V_k)$, denoted by $\overline{U} \prec \overline{V}$, iff $\overline{U}$ is partially less than $\overline{V}$, that is $\forall i \in \{1, 2, \ldots, k\}$, $u_i \leq v_i \wedge \exists i \in \{1, 2, \ldots, k\} : u_i < v_i$. For our COPT problem, we consider the two vectors $\overline{U}, \overline{V}$ as $i = \{f_p^i, f_d^i, f_a^i\}$ and $j = \{f_p^j, f_d^j, f_a^j\}$, both having 3 objectives. Based on the above definition, we obtain the non-dominated candidates from the population.

After obtaining the non-dominating candidates, we sort them according to various levels

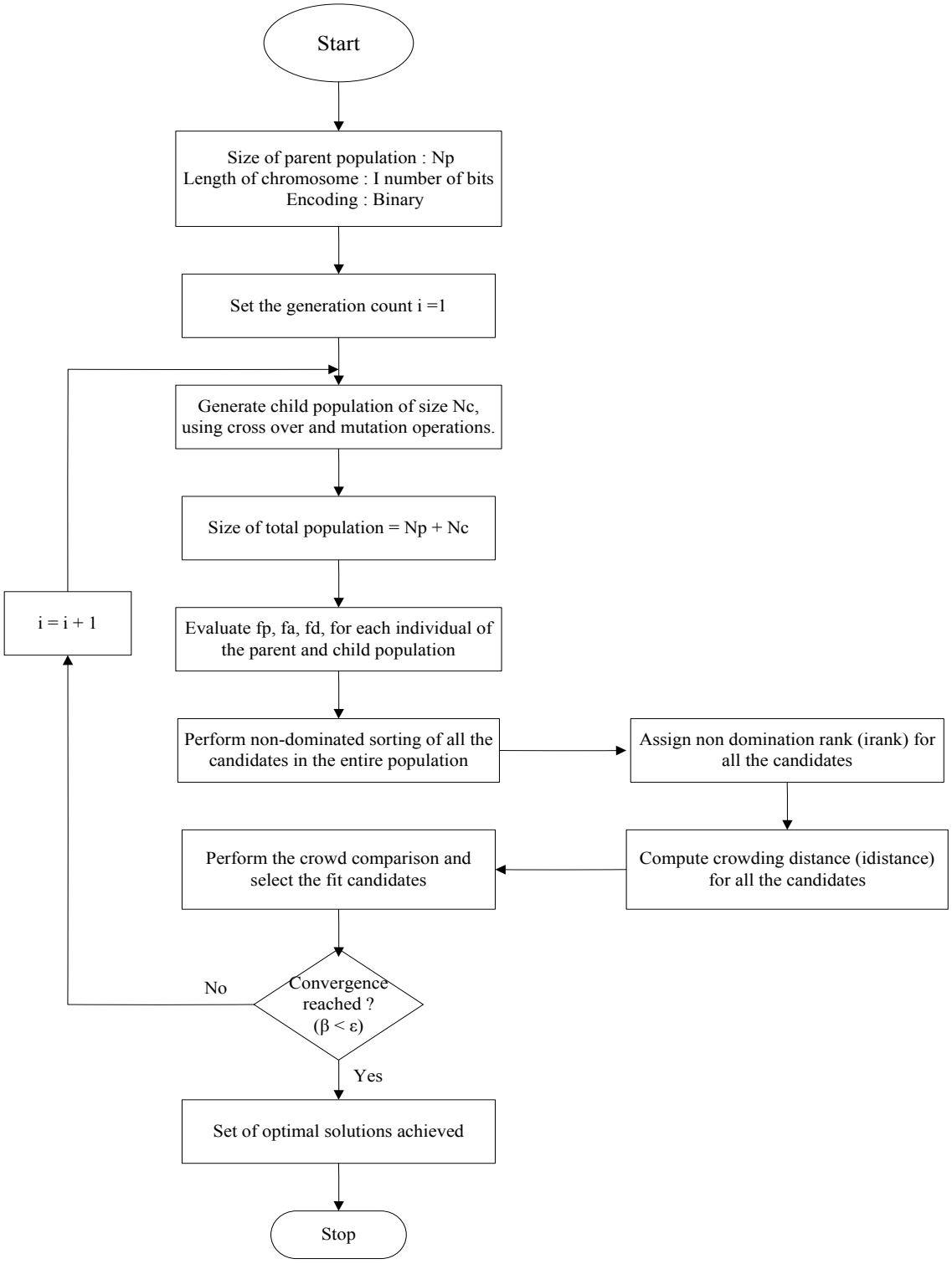


Figure 3.4: NSGA II framework to solve COPT problem

of domination. All candidates in a particular level have f_p, f_d, f_a values which are not dominated by remaining candidates of the same level. A candidate belonging to higher domination level dominates all candidates in its lower domination levels.

We assign ranks to all candidates which are in various non-domination levels. The candidates in higher non-domination level are given a lower rank. As we go down the domination levels, rank of the candidates in a particular domination level increases. Rank one is given to all the candidates which dominate the candidates of all other levels. Like this a non-domination rank(i_{rank}) is assigned to all candidates of the entire population.

After assigning i_{rank}, we compute the crowding distance($i_{distance}$) for each candidate in a given level. This is done by measuring the average distance of two nearest candidates on either side of i along that particular non-domination level. We use the notation d_{xy}, which represents the Euclidean distance between two candidates x and y belonging to the population. It is computed as shown below:

$$d_{xy} = \sqrt[2]{(f_p(x) - f_p(y))^2 + (f_a(x) - f_a(y))^2 + (f_d(x) - f_d(y))^2} \tag{3.5}$$

For an individual i the crowding distance is computed as shown below, where j, k are the nearest neighbors to i on either side, along the non domination level.

$$i_{distance} = \frac{1}{2}(d_{ik} + d_{ij}) \tag{3.6}$$

Like this for all individuals in the population, their respective crowding distance($i_{distance}$) are computed. Using these two measures of an individual the crowd comparison is performed. If α is the crowd comparison operator, as stated in [111] we define

$i \alpha j$, if ($i_{rank} < j_{rank}$) or if (($i_{rank} = j_{rank}$) and ($i_{distance} > j_{distance}$))

It means, if two solutions are belonging to different non-domination ranks, then we select the solution with lower rank. If both solutions are having the same rank, we select a solution which is in less crowded region, that is, having more $i_{distance}$. As shown in Fig. 3.4, we used crowd comparison operator [111] to select the solutions towards a uniformly spread-out Pareto-optimal front.

As suggested by Deb et al. [111], we choose, β which defines the diversity of the obtained

Pareto optimal set, as a criteria for terminating our GA. This parameter is observed over five successive generations. If the change in the value of this parameter is less than ϵ (a predefined constant [111]), then the algorithm is stopped, because the desired diversity is obtained.

After obtaining the optimal Pareto front we compute Nadir point ($nadir$) for the front, in order to choose the best individual. For our $COPT$ problem, the 3 co-ordinates of Nadir point namely $\{X'_{nadir}, Y'_{nadir}, Z'_{nadir}\}$ in the state space are expressed as follows

$$X'_{nadir} = max(f_p(X)), Y'_{nadir} = max(f_d(X)), Z'_{nadir} = max(f_a(X)) \tag{3.7}$$

where, $max(f_p(X)), max(f_d(X)), max(f_a(X))$, are the maximum values of power, area and delay obtained in that particular level. Using this $nadir$ we compute $d_{i,nadir}$ for all candidates in the Pareto front. We choose the candidate which has the minimum $d_{i,nadir}$, as the most fitting candidate.

Using this candidate we obtain the final $\{U_{opt}, B_{opt}\}$. These set are used to realize an optimal mixed CMOS circuit.

3.3 Experiments and Experimental Results

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

3.3.1 Objectives

We validate our decomposition algorithm on a set of standard benchmark circuits. We estimate the performance of the circuit with reference to IUD and OUD. Also, we compare the circuits with OUD, with only static and only Domino logic. Finally, we compare our approach with the existing approaches to mixed CMOS VLSI circuit realization.

3.3.2 Experimental setup

The decomposition 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. Power, area and delay are chosen as metrics for evaluating the performance of the circuits. To perform mapping of .pla files (benchmark files) we have developed a set of static and Domino cell libraries. While developing libraries, we have included a set of standard cells which form building blocks of any circuit. Some of these cells along with their respective transistor count are mentioned in Table. II. The t_{LH} and t_{HL} give the respective rise and fall delays obtained from simulations performed in CMOS 180nm process, 1.8V, 27^oC. Though some standard functions may appear in both the libraries their respective parameter values differ.

Berkeley SIS tool, Version 1.3 [113] is used for mapping and various preprocessing of circuits. The logic descriptions of unate, binate components along with static and Domino libraries are used by the SIS tool while performing the mapping. The *map -m* command is executed for mapping a circuit. Since we are performing library based mapping, we used the *print_delay -m library* command which uses a library based delay model. The dynamic power for the component is estimated by using *power_estimate -f* command. The tool takes an estimate of the activity, C_g, C_d and computes the dynamic power for a given V_{dd} and f. We have used 1.8 V as supply voltage (V_{dd}) and clock frequency(f) is set to 20MHz. For overall power, area, the sum of individual power, area of the components are considered. For overall delay, the one which has more delay amongst static and Domino block is considered.

For performing the optimization we used the '*ga_optimtool*' available in MATLAB software, Version 8.1a. We have written a script file *circuit_optimum.m* which defines the functions that are to be optimized. For evaluating power, area and delay for a given

Table 3.2: List of some of the cells present in the libraries.

Name of cell	Static cell library			Name of cell	Domino cell library		
	(transistor) count)	(t_{LH} in ps)	(t_{HL} in ps)		(transistor count)	(t_{LH} in ps)	(t_{HL} in ps)
inverter	2	1.12	1.4	inverter	-	-	-
2-nand	4	2.35	2.02	domino 2-and	6	1.32	1.67
3-nand	6	2.16	2.94	domino 3-and	7	2.46	3.12
2-nor	4	1.45	1.92	domino 2-or	6	1.7	1.2
3-nor	6	2.1	2.8	domino 3-or	7	2.1	1.85

Table 3.3: Considered benchmarks from ISCAS85 and MCNC89

Circuit	Circuit function	Input lines	Output lines
b1	–	3	4
ex5	–	8	63
9sym	–	9	1
x3	–	135	99
C880	ALU and Control	60	25
C1908	ECAT	33	25
C2670	ALU and Control	233	140
C5315	ALU and Select	178	123

member of population SIS tool is invoked through the script file. The tool runs for 50 generations as most of the test cases converged much before that. We have considered crossover probability(p_c) as 0.9. The mutation probability(p_m) is taken to be 0.15, as recommended in several literature on NSGA II [111]. The diversity operator β as mentioned in [111] is observed over 5 successive generations. The constant ϵ which measures the change in β over successive generations, is chosen to be 0.001.

3.3.3 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*). Also, 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 233 input 140 output variable ALU and Control described using (*C2670.blif*). Characteristics of some of the benchmark circuits considered are shown in Table. 3.3.

3.3.4 Experimental results

The performance of our IUD algorithm with some MCNC89 benchmark circuits is shown in Table. 3.4. The number of states in the onset *(OS)*, the number of states in the unate set *(U)* and binate set *(B)* after decomposition are mentioned in columns 3 to 5. The last column mentions the CPU runtime required for carrying out the decomposition in each case. The computing time for the decomposition algorithm increases rapidly with the increase in

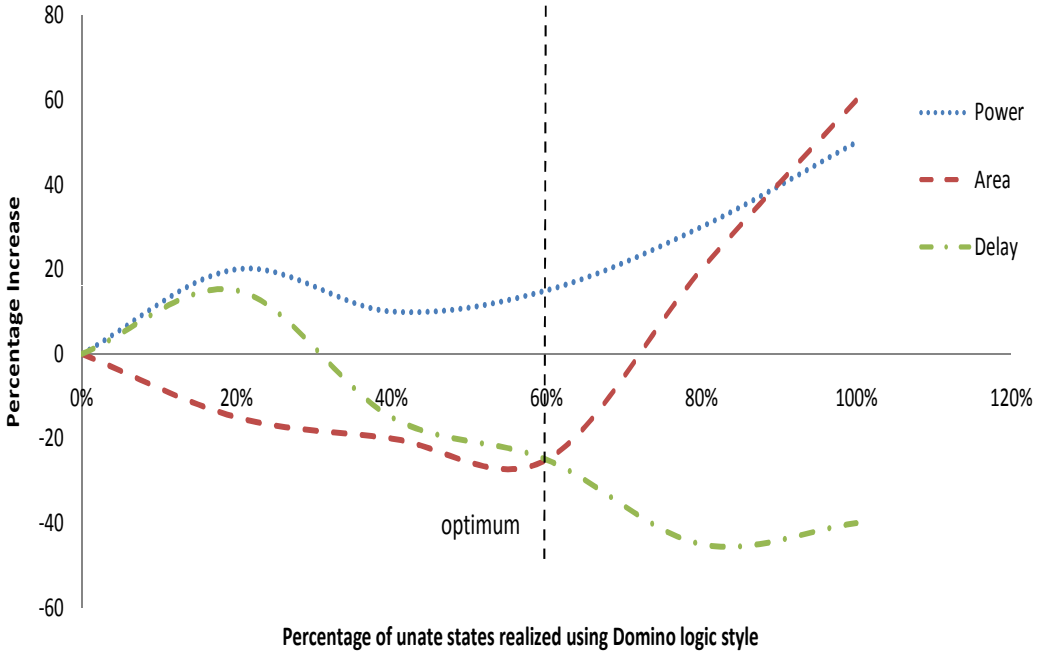
number of input variables. For three functions we have observed the respective run times. The circuits *9sym*, *t481* and *tcnh0* with 9, 16 and 17 input variables required 2, 85 and 1634 seconds, respectively. The possible reason for this can be the exponential rise in memory and power requirements of the CPU. It is clear from the fact that number of instances we have to analyze for an N variable function is 2^N. Hence, we applied the algorithm directly, only for circuits having input variables less than 19. To handle the circuits having input variables above 19, we adopted a standard preprocessing technique which uses *SIS* tool. In case of multi-output circuits, we have considered each output separately.

With reference to two standard ISCAS benchmark circuits namely C880.pla and C1908.pla, the realization of both the static and Domino circuits are shown in Fig. 3.5. The respective power, area and delay are normalized with respect to the static CMOS realization. Obtained normalized values are shown in Fig. 3.5 (a), 3.5 (b), as function of percentage of unate states realized using Domino logic style.

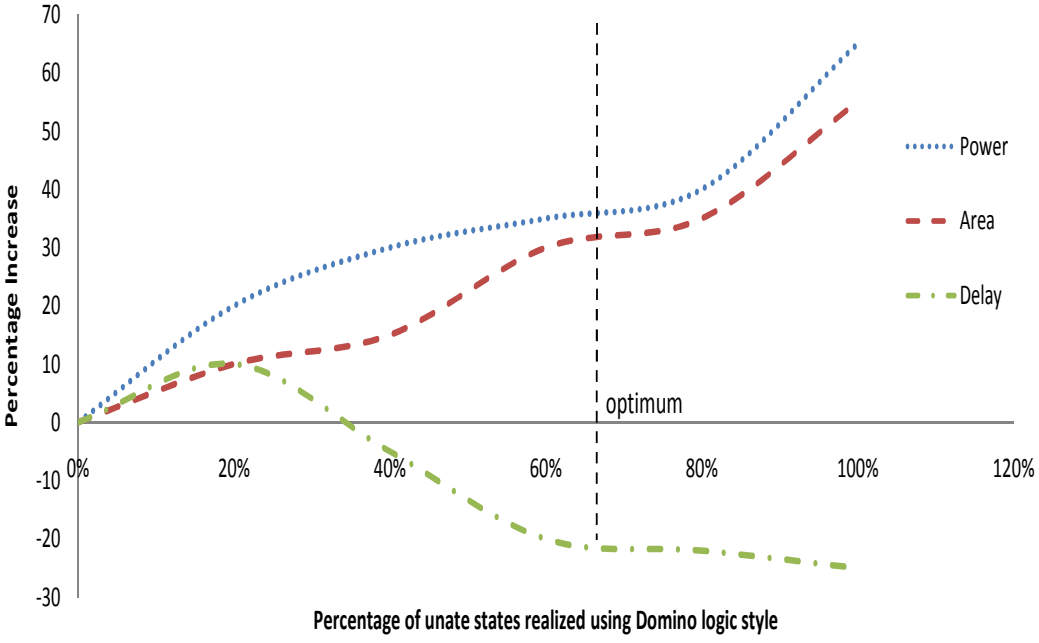
We have computed the power, area, delay values of the mixed CMOS realization at various stages. These values range from realizing the entire unate states using static logic to realizing all of them using Domino logic. The percentage increase with respect to corresponding static realization is plotted at each stage. The value at 0% Domino realization of unate set corresponds to pure static realization. Hence, the percentage increase at this point is shown as 0%. As the percentage of Domino realization of unate states increases there is an initial increase in the delay followed by gradual decline. This can be accounted by the fact that the initial Domino nodes may not be in the critical path. When their number is increased they significantly effect the critical path and hence reduce the delay. Power and area of realizing all unate states in Domino style is 50% and 60% more than

Table 3.4: Performance of IUD

Circuit	I/O	State Count			CPU time (seconds)
		OS	U	B	
5xp1	7/10	640	505	135	2
9sym	9/1	238	173	65	2
cm151a	12/2	4100	3234	866	5
alu2	10/6	5120	3953	1167	10
C1908	60/25	10402	7321	3081	500
C880	33/25	13023	8415	6608	732



(a)



(b)

Figure 3.5: Percentage increase of power, area and delay using IUD realization over static CMOS realization for (a). C880.pla, (b). C1908.pla

Table 3.5: Performance of our decomposition algorithms

Circuit	I/O	Mixed CMOS (IUD)			Optimized Mixed CMOS (OUD)		
		Power(uW)	Area	delay(ns)	Power(uW)	Area	delay(ns)
b1	3/4	22.3	68	41	18.4	57	46.4
ex5	8/63	298.5	2912	236.3	269.3	2635	290.3
9sym	9/1	329.7	650	260	288.3	592	285.4
x3	135/99	3231.4	2985	170.6	3002.1	2654	185.4
C880	60/25	2648.3	643	13.8	2483.7	613	16.2
C1908	33/25	2895.8	650	29.7	2693.4	535	33.5
C2670	233/140	4485.3	870	21.6	4132.6	759	24.3
C5315	178/123	12835.3	2592	27.4	10533.7	2314	32.3

the corresponding static realization. This can be supported by the fact that Domino logic style has higher switching and hence higher dynamic power dissipation. Also in order to realize pure Domino we have to follow two-level procedure which result in huge number of transistors. The possible optimum is achieved when 60% of unate nodes are realized using Domino logic (shown with a marker in Fig. 3.5 (a) and 3.5 (b)). The optimum, in the case of C1908.pla is achieved when 70% of unate nodes are realized using Domino logic.

Experimental results on ISCAS'85 and MCNC'89 benchmark suites using mixed CMOS (IUD) and optimized mixed CMOS (OUD) are shown in Table 3.5. The power, area and delay values for the circuit realization before and after optimizing are mentioned in this table. We can clearly see from Table 3.5, that the optimization we carried out resulted in a significant savings in terms of power and area. For the circuit C2670, the savings in power and area after optimization is 8.5% and 13%, respectively. However, the optimization process imposed an average penalty of 7% on delay when compared to simple mixed CMOS design.

Comparison of our approach with related methods is mentioned in Tables 3.6, 3.7, 3.8. Static CMOS realization [113], Two-level decomposition [57] are compared with Prasad's

Table 3.6: Comparative study of various approaches w.r.t power dissipation

Name of the approach	Power dissipation of circuit in uW							
	b1	ex5	9sym	x3	C880	C1908	C2670	C5315
static approach [113]	17.7	215.1	501	2783	2017.7	2242.7	3680.7	9015.6
Two-level approach [57]	31.4	3743	898	3982.1	2823.1	3251.5	4984.1	13720.3
Prasad's approach [54]	20.8	235.4	482.3	2943	2439.4	2851.4	4390.7	10547.1
Jacob's approach [58]	21.4	246.4	632.3	2945.3	2312.4	2489.4	3895.4	10213.7
Kim's approach [53]	22.3	253.6	712.9	3431.3	2613.4	2501.3	3920.4	11321.6

Table 3.7: Comparative study of various approaches w.r.t area

Name of the approach	Area of circuit in number of transistors							
	b1	ex5	9sym	x3	C880	C1908	C2670	C5315
static approach [113]	44	4234	940	4217	766	1071	1325	3004
Two-level approach [57]	77	7692	1732	6841	1225	1714	2120	4806
Prasad's approach [54]	55	4123	863	4162	920	1189	1562	2983
Jacob's approach [58]	49	4126	1053	2931	854	1234	1543	2885
Kim's approach [53]	45	4089	1001	2741	814	1139	1346	2685

Table 3.8: Comparative study of various approaches w.r.t delay

Name of the approach	Delay of circuit in ns							
	b1	ex5	9sym	x3	C880	C1908	C2670	C5315
static approach [113]	58	355.8	423.3	250.7	21.4	41	31	39.8
Two-level approach [57]	37	192.4	280.4	140.3	12.84	24.6	18.6	23.9
Prasad's approach [54]	45	303.4	394.6	242.4	18.3	35.7	26.4	29.3
Jacob's approach [58]	66	376.4	413.2	243.2	24.1	38.3	37.4	44.6
Kim's approach [53]	69	408.9	431.2	277.3	27.4	42.3	40.5	47.8

[54], Jacob's [58] and Kim's [53] approaches. From these tables it can be seen that the Two-level approach gives the minimum delay amongst the existing techniques. This can be accounted from the fact that it employs a pure Domino logic style throughout its design. Domino logic are faster than their static counterparts [113]. This approach is 22% faster than optimized mixed CMOS, 10% faster than simple mixed CMOS (from Table 3.5). However, this particular approach requires the maximum number of transistors compared to other existing approaches. This is a potential drawback of this approach. The Jacob's approach

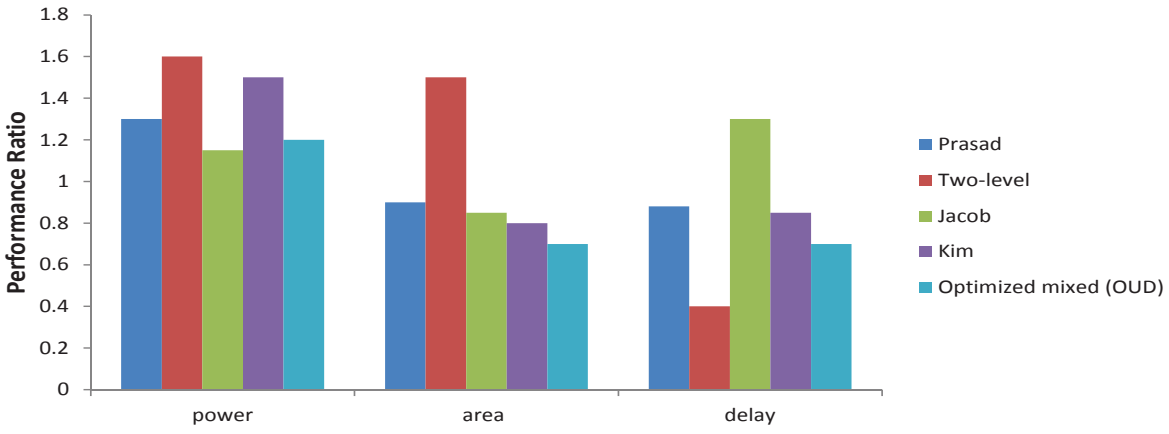


Figure 3.6: Performance ratio of various approaches with respect to Static CMOS realization for C5315.pla

(mentioned in Table 3.6, 3.7, 3.8), consumed significantly less power and area than the two level approach. This is possible because there are some static blocks in the final design using this approach. They account for low power and even give rise to more delay. Kim's approach (mentioned in Table 3.6, 3.7, 3.8), though suffered with huge power dissipation compared to Prasad's and Jacob's approach, it consumed less area and offered lesser delay compared to them. This can be supported by the fact that Kim's approach moved closer to realization of Domino logic based circuits when compared to Prasad's and Jacob's. However, our optimized mixed CMOS approach outperforms this technique both in terms of power and delay.

Results of synthesis using Prasad's approach [54] are mentioned in 9 to 12 of Table 3.6. Our approach has clearly outperformed the Prasad's approach both in terms of area and delay. Our approach has shown 40% reduction in area and 12% reduction in delay as against 1% penalty in power dissipation. This is possibly because Prasad's approach results in the presence of trapped inverters within the circuit which can be realized using static CMOS logic style only. Performance ratio of various approaches against corresponding static CMOS realization, for an ISCAS bench mark $C5315.pla$, is shown in Fig. 3.6.

3.4 Conclusion

In general, Domino CMOS circuits consume require less number of transistors and offer minimum delay. In this work, our mixed CMOS design has shown advantage in terms of speed and area when compared to circuits with only static logic style. The NSGA based circuit optimization results in further improvement of results with respect to power dissipation.

Chapter 4

Technology Mapping for Domino Logic

Technology mapping is an essential step that captures the behavior of the circuits with the help of various gates [60], [61], [62]. In the last chapter, after decomposition we have obtained a static and Domino logic block. Mapping the Domino logic can be done in different ways. In this chapter, we propose a method to map the Domino logic block so that the overall delay is minimum.

Many works reported in literature begin with a NAND based directed acyclic graph (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 investigate in this direction. Also there is a scope for re-ordering the cells along critical path, which further minimizes delay. Fine tuning these cells along the critical path, without increasing their individual transistor count is in fact a challenging task.

Keeping the above scope in view, in this chapter, we propose a cell re-ordering based Domino on-the-fly mapping. We call this as CRDOM approach. First, we convert a 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 further minimizes delay. Finally, we choose an optimum cell re-ordering set along the critical path, where delay and area penalty are minimized by using a two objectives optimization problem.

The rest of the chapter is organized as follows. Section 4.1 describes some basic concepts, which will be referred in our approach. Our proposed methodology for realizing cell re-ordering based Domino on-the-fly mapping is elaborated in Section 4.2. Experiments and the experimental results obtained along with a comparative analysis are presented in Section 4.3. Finally, Section 4.4 concludes the chapter.

4.1 Basic Concepts and Definitions

In the following, we present some basic concepts which we use in our discussion.

Definition 4.1 *Node:* A gate (cell) in a given circuit is often refereed as a node in the graphical representation of the circuit. A gate equivalent of a node in a circuit graph is shown in Fig. 4.1.

Definition 4.2 *Circuit graph:* A graphical representation of Boolean circuit, where various gates form various nodes of the graph. In Fig. 4.1, a single node graph is shown.

Definition 4.3 *Width of node:* Width of a node is defined as the number of parallel chains of transistors present in the node. For the node shown in Fig. 4.1, the width is 3.

Definition 4.4 *Height of node:* Height of a node is defined as the maximum number of serial transistors present in a single chain in the node. For the node shown in Fig. 4.1, the height is 3.

Definition 4.5 *Leaf node:* All the primary inputs to a circuit form leaf nodes in the graphical representation of then circuit.

Definition 4.6 *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.

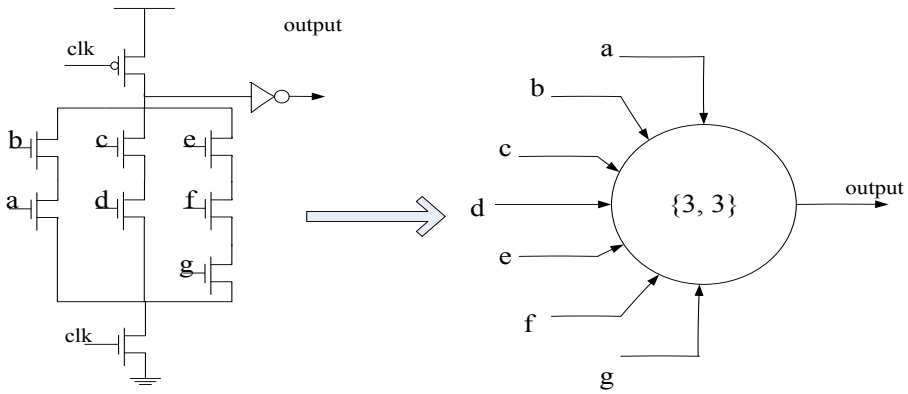


Figure 4.1: A dynamic cell and its nodal representation

Definition 4.7 *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.

Definition 4.8 *Original cell:* A cell belonging to a circuit before re-ordering is called the original cell.

Definition 4.9 *Re-ordering cell:* The functionally equivalent cell which replaces an original cell along the critical path after re-ordering is called re-ordering cell.

Definition 4.10 *Functional equivalence:* During the mapping procedure, two different cells C_1, C_2 may have different architectures, yet they may realize the same function. If such two cells exist, then we say that cells C_1 and C_2 are functionally equivalent.

Definition 4.11 *Equivalence table:* Equivalence table (ET) keeps a database of all possible cells (under the constraints height, width) and their corresponding functionally equivalent cell. It also stores additional information such as, improvement in delay, penalty on area etc. when re-ordering is done.

Lemma 4.1: A minimum of 5 input gate must be used to gain mapping 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 static CMOS logic style requires $2N$ number of transistors. In order to gain advantage using Domino style, Eqn. 4.1 must be satisfied. Hence, the number of inputs to the circuit must be greater than 4.

$$N + 4 < 2N \tag{4.1}$$

Lemma 4.2: The delay of a particular node increases with the increase in height of the node.

Proof: The resistance of individual NMOS transistors increases, 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. Hence, delay of a node is proportional to the height of the node [57].

4.2 Proposed Methodology

We call our proposed approach to Map Domino cells as CRDOM an acronym of Cell Re-ordering based Domino On-the-fly Mapping. An overview of our proposed approach CRDOM is shown in Fig. 4.2. Our approach consists of the following.

Given a unate block C_{init} which is completely unate in nature. Let L_{dom} be the library of various gates that are required to map the initial logic.

Raw_Map: An on-the-fly mapping step, denoted as Raw_Map which takes the circuit C_{init} containing a set of gates $g_1, g_2, \ldots, g_n$ as input and gives C_{Raw_Map} as the resulting

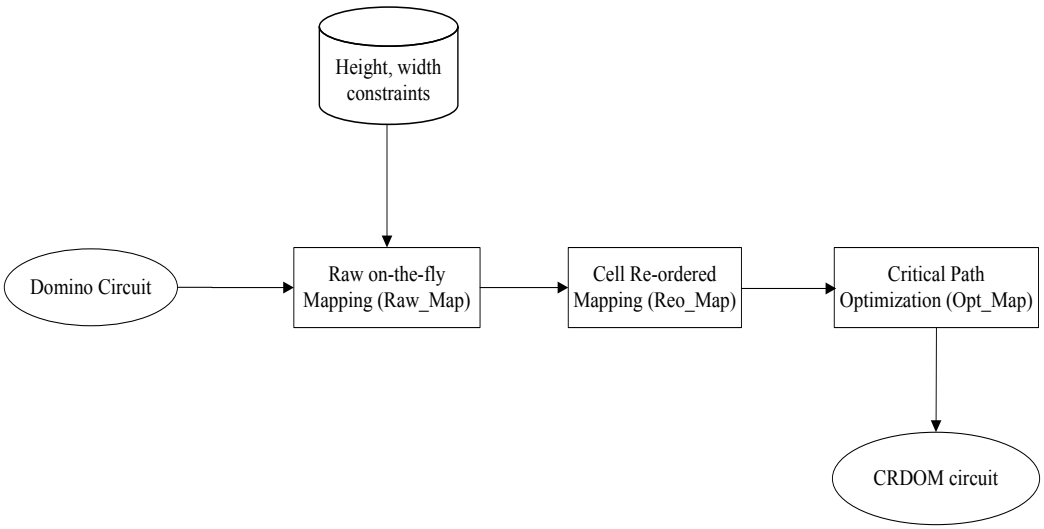


Figure 4.2: Overview of CRDOM approach

circuit which consist another set of gates $G_1, G_2, \ldots, G_k$, which obey certain height, width constraints. We represent this as

$Raw\_Map\{C_{init}(g_1, g_2, \ldots, g_n)\} \rightarrow C_{Raw\_Map}(G_1, G_2, \ldots, G_k)$

Reo_Map: A 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 with respect to delay. If not, this step replaces the gates with the corresponding functionally equivalent gates which are lower bound with respect to delay. We define this as $Reo\_Map\{C_{Raw\_Map}(G_1, G_2, \ldots, G_l)\} \rightarrow C_{Reo\_Map}(G_1^*, G_2^*, \ldots, G_m^*)$, where $(G_1, G_2, \ldots, G_l)$ are gates obtained after Raw_Map step and $(G_1^*, G_2^*, \ldots, G_m^*)$ 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. Suppose, $Opt\_Map\{C_{Reo\_Map}(G_1^*, G_2^*, \ldots, G_m^*)\} \rightarrow C_{Opt\_Map}(G_1^o, G_2^o, \ldots, G_m^o)$, where $(G_1^o, G_2^o, \ldots, G_m^o)$ are the final gates present along the critical path of the circuit.

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

4.2.1 Raw mapping

We propose an algorithm which maps an arbitrary library based Domino circuit to an on-the-fly Domino circuit using a node mapping algorithm. A flowchart of the algorithm is shown in Fig. 4.3. Various notations used in the flowchart are presented in the following.

Before explaining the algorithm, we define the following operations which we refer to in our discussion. Given a node N_k, let $H(N_k), W(N_k)$ denote the height and width of the node respectively. Given two numbers a, b, we define functions SUM, MAX where

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

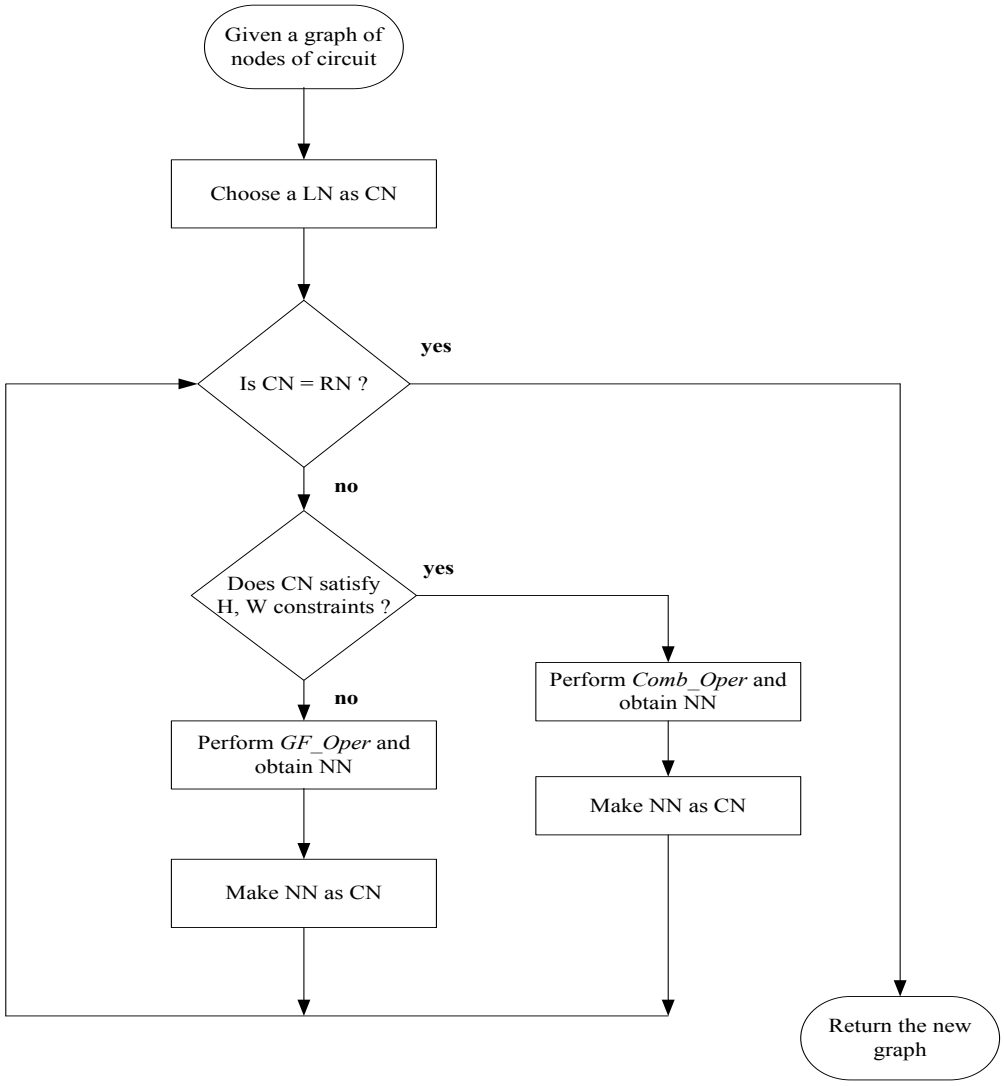


Figure 4.3: Flowchart of the node mapping algorithm

$SUM(a, b)$ gives the sum value of a and b, $MAX(a, b)$ returns the maximum amongst a and b as output.

Comb_Oper: In this operation, two or more nodes belonging to a circuit graph are combined to form a new node. Suppose, nodes N_1 and N_2 are combined to a new node N_{new}. These are the following cases:

- If the nature of combination is AND, then $H(N_{new})$ is $SUM(H(N_1), H(N_2))$ and $W(N_{new})$ is $MAX(W(N_1), W(N_2))$.

- If the nature of combination is OR, then $H(N_{new})$ is $MAX(H(N_1), H(N_2))$ and $W(N_{new})$ is $SUM(W(N_1), W(N_2))$.

GF_Oper: A mapping procedure takes place under certain height (H), width (W) constraints. Whenever, the constraints exceed threshold values, there a gate is formed there itself. The output of this gate is continued with a new node which starts with H=1, W=1. This node will carry forward the function of formed gate. Such an 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, let us consider the Boolean equation mentioned in Eqn. 4.2.

$$f(x) = \{(a+b)cde + (ed + bc + a + c)\} \tag{4.2}$$

where a, b, c, d, e, d are the primary inputs to the circuit. The graphical representation of Eqn. 4 is shown in Fig. 4.4 (a). The initial mapping is done using a 2-input, 3-input AND (•), OR (+) Domino gates (these gates belong to the library L_{dom} in this case).

Suppose, the height and width limits for mapping are set to $H_{Max} = 4$, $W_{Max} = 4$. Initially, all leaf nodes have a dimension {1,1}. Hence, all of them satisfy the dimension (height, width) limits. Next, the node 4 which has inputs a and b is considered as current node (CN). Performing *Comb_Oper* on this node will lead to a node of dimension {1,2}. A similar operation is performed on nodes 5, 6, 7, 8 which results in formation of new nodes of dimension {2,1}, {2,1}, {2,1}, {1,2} respectively. These are shown in Fig. 4.4 (b). Since, these dimensions are within the maximum defined limits the *Comb_Oper* is further applied to nodes 2, 3, respectively. After *Comb_Oper* is performed to node 2, a new node '9'is formed with dimensions {4,2} and inputs a, b, c, e. Similarly, a new node '10' is formed by applying *Comb_Oper* on node 3. It also has dimensions {2,4} and inputs a, b, c, d, e (shown in Fig. 4.4 (c)). Performing of *Comb_Oper* to nodes 9, 10 will result in exceeding the limits of H, W. Hence, the *GF_Oper* as mentioned in flowchart (shown in Fig. 3), is performed on nodes 9, 10. In the final mapped circuit, we have nodes 1, 9, 10 having dimensions {1,2}, {4,2}, {2,4}, respectively.

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

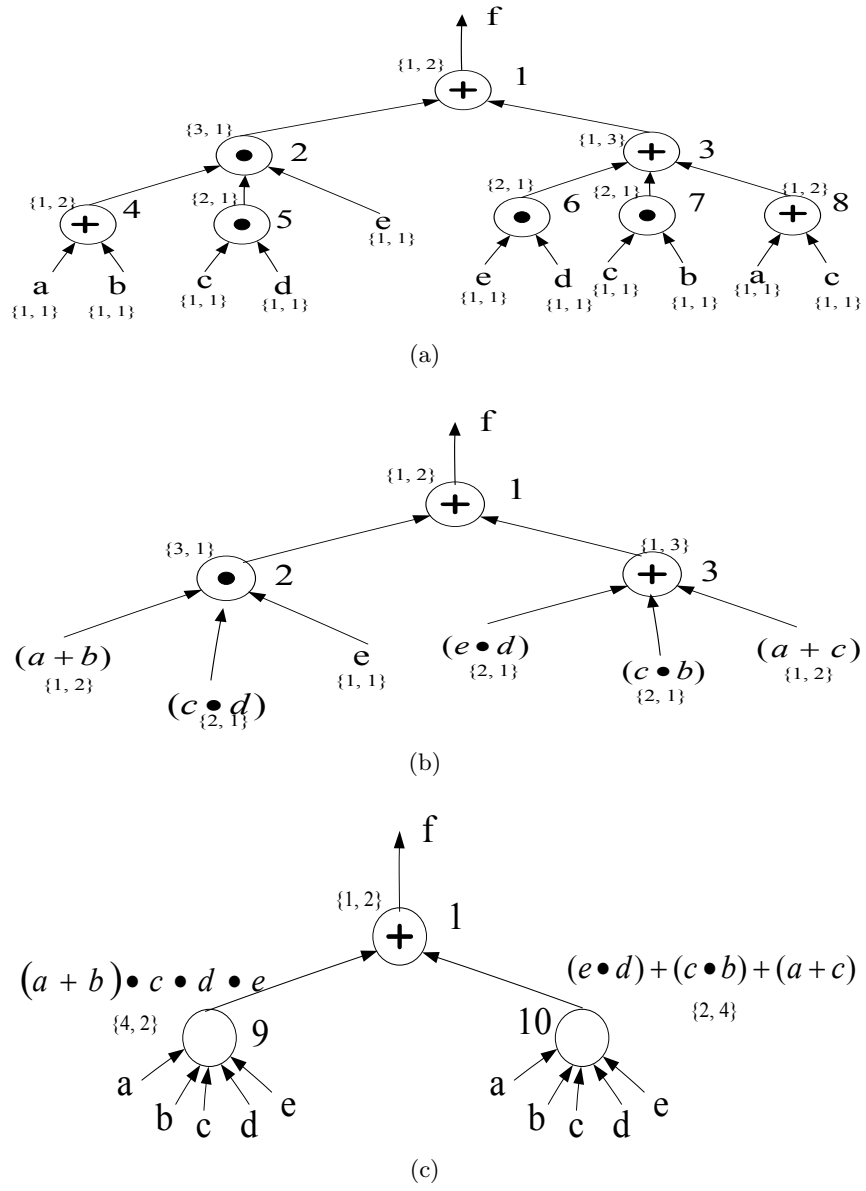


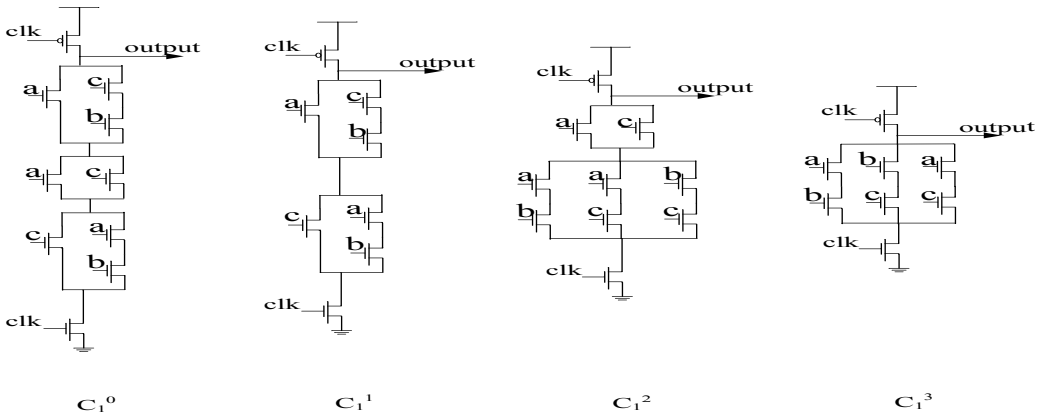
Figure 4.4: Example of node mapping algorithm (a) Initial stage (b) Intermediary stage (c) Final stage

of nodes (cells) along the critical path. It is mentioned in the following section.

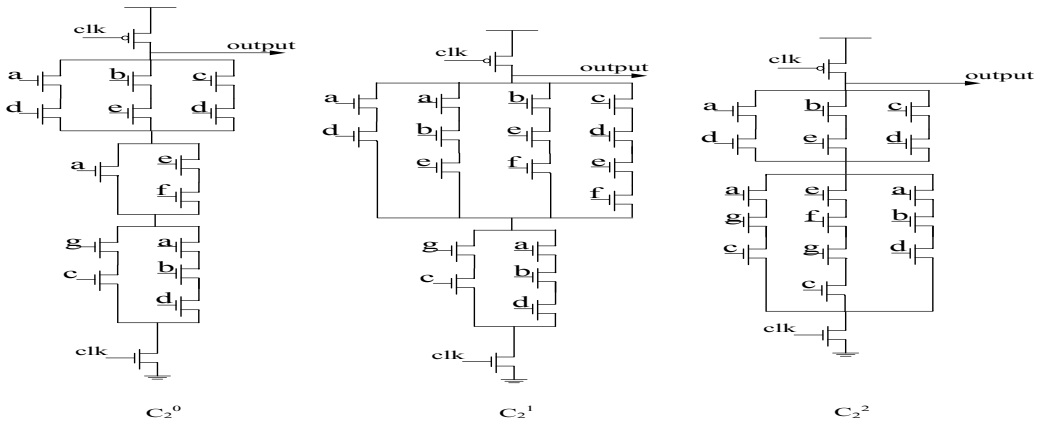
4.2.2 Re-ordered mapping

In this section, we discuss a cell re-ordering based mapping to obtain a circuit with minimum delay.

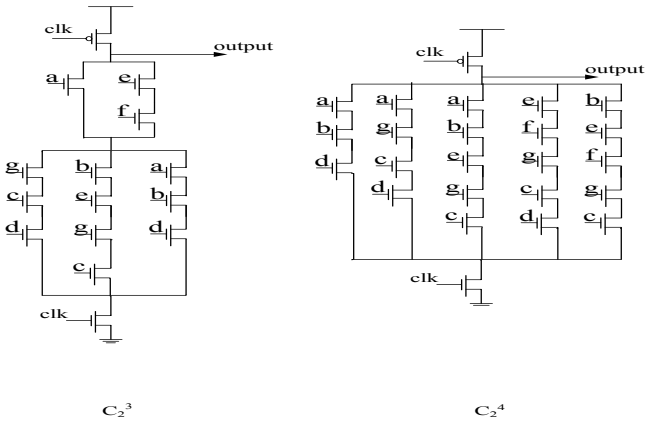
Domino cells obtained with raw mapping may have height and width constraints, can



(a)



(b)



(c)

Figure 4.5: Re-ordering cases for cells (a.) C_1 , (b.) & (c.) C_2

realize a wide range of functions. For example we consider two different functions mentioned in Eqn. 4.3, 4.4 to be realized in single cells respectively.

$$f = (a + bc)(c + a)(ab + c) \tag{4.3}$$

$$f = (ad + be + cd)(a + ef)(gc + abd) \tag{4.4}$$

Two different cells C_1^0, C_2^0 shown in Fig 4.5 (a), 4.5 (b) represent their initial realizations. Three other similar equivalence cases are shown in Fig. 4.5 (a) for cell C_1^0 and four equivalence cases are shown in Fig. 4.5 (b) and 4.5 (c) for cell C_2^0. All such equivalence case for cells C_1^0 & C_2^0 along with amount of delay advantage and area penalty is stored in an Equivalence Table (defined earlier). The Equivalence Table (ET) for cells shown in Fig. 4.5 (a), 4.5 (b), 4.5 (c) is shown in Table 4.1. Only the re-ordering cells which give a positive delay advantage w.r.t original cells are considered.

Using the ET, the *Reo_Map* step is performed on circuit shown in Fig. 4.6. 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.

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.

4.2.3 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 propose a two objective Particle Swarm based approach to solve the cell re-ordering optimization 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 re-ordering cells 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.

Table 4.1: Example of an equivalence table (ET) for two different original cells (Delay measured as number of levels, area as number of transistors)

Original cell name	Original function	Re-ordered cell name	Re-ordered function	$Delay^1$	$Area^2$
C_1^0	(a + bc)(c + a)(ab + c)	—	—	5	8
C_1^0	(a + bc)(c + a)(ab + c)	C_1^1	(a + bc)(ab + c)	4	6
C_1^0	(a + bc)(c + a)(ab + c)	C_1^2	(a + c)(ab + ac + bc)	3	8
C_1^0	(a + bc)(c + a)(ab + c)	C_1^3	ab + ac + bc	2	6
C_2^0	(ad + be + cd)(a + ef)(gc + abd)	—	—	7	14
C_2^0	(ad + be + cd)(a + ef)(gc + abd)	C_2^1	(ad + abe + bef + cdef)(gc + abd)	7	17
C_2^0	(ad + be + cd)(a + ef)(gc + abd)	C_2^2	(ad + be + cd)(agc + efgc + abd)	6	16
C_2^0	(ad + be + cd)(a + ef)(gc + abd)	C_2^3	(a + ef)(gcd + begc + abd)	6	13
C_2^0	(ad + be + cd)(a + ef)(gc + abd)	C_2^4	(abd + acdg + abceg + cdefg + bcefg)	5	21

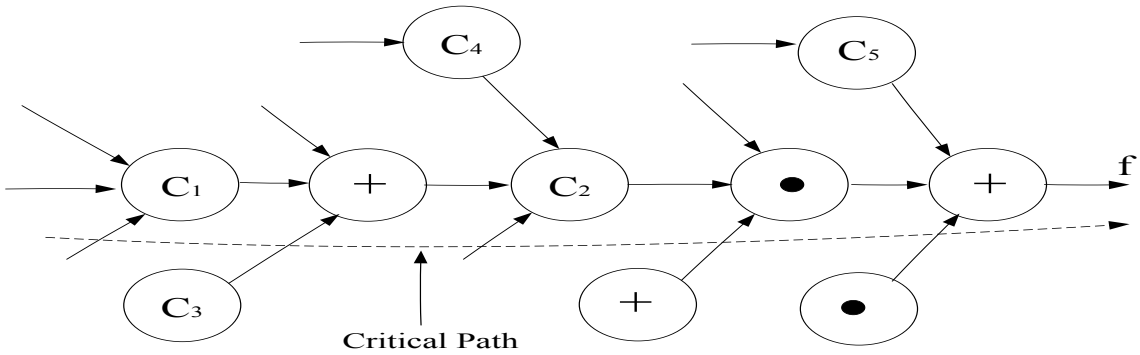


Figure 4.6: Cells C_1, C_2 along the critical path suitable for re-ordering

Notations used in solving CROPT		
RS	:	Re-ordering set
REP	:	Repository
W	:	Damping co-efficient [114]
r_1	:	cognitive coefficient [114]
r_2	:	social coefficient [114]
K	:	number of cells along critical path
$LBEST$	:	Local best particle in current iteration
$GBEST$	:	Global best particle over all occurred iterations
X_i^j	:	Position vector for particle i in j^{th} iteration
V_i^j	:	Velocity vector for particle i in j^{th} iteration
MAX	:	Population size
$ITER$	:	Limit on number of iterations

For a given initial re-ordering RS_{init}, let $\{C_1^{R_1}, C_2^{R_2}, \ldots, C_i^{R_i}\}$ be the i number of re-ordering cells for a number of *original cells* along the critical path of a circuit C_{init}. These are obtained from the *Reo_Map* step. The value of R_i can range from 0 to the maximum number of equivalence cases possible for cell i. Our objective is to find an optimum re-ordering set (RS_{opt}) resulting in a particular re-ordering of critical path. That is

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

We consider the following two objective functions to judge the optimality of RS. Suppose an arbitrary re-ordering set be $RS_k = \{C_1^{R_1}, C_2^{R_2}, \ldots, C_k^{R_k}\}$. Let $f_d(C_{init}, RS_k)$ denotes the delay advantage (D_{sav}) obtained by re-ordering the circuit with RS_k. Similarly, $f_a(C_{init}, RS_k)$ denotes the estimation of area penalty (A_{pen}) incurred while re-ordering. We aim to model our CROPT as a 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($C_{init}, C_1^{R_1}, C_2^{R_2}, \ldots, C_i^{R_i}$):

$$
\begin{aligned}
Opt\_Reo\ (C_{init}, RS_{opt}) \ &= minimize\ \{A_{pen} = f_a(C_{init}, RS_k)\}, \\
&\quad minimize\ \{InD_{sav} = (1 + f_d(C_{init}, RS_k))^{-1}\}, \\
&\quad subject\ to\ InD_{sav} \leq D_0,\ A_{pen} \leq A_0,\ \{RS_k\} \subseteq \{C_1^{R_1}, C_2^{R_2}, \ldots, C_i^{R_i}\}, \\
&\quad for\ \ given\ constraints\ D_0,\ A_0\ [115] \\
&\quad and\ R_i\ ranges\ from\ 0\ to\ the\ maximum\ number \\
&\quad of\ \ equivalence\ cases\ for\ cell\ C_i
\end{aligned}
$$

PSO optimization: We follow a Particle Swarm multi objective optimization algorithm (MOPSO) [114] 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 multiobjective optimization approaches [115], [116], [117]. Also this procedure has less dependence on the set of initial points when compared with other Domino based approaches [114], [118], [119]. 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 re-ordering set (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. 4.5 [115]. The velocity vector of i^{th} particle for $(j+1)^{th}$ iteration is computed using the particle's position vector in j^{th} iteration and the local best particle (LBEST), global best particle (GBEST) computed at j^{th} iteration. The values of r_1, r_2 are taken as mentioned in [114].

$$V_i^{j+1} = W \times V_i^j + r_1 \times (LBEST[j] - X_i^j) + r_2 \times (GBEST[j] - X_i^j) \tag{4.5}$$

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 for j^{th} iteration involves fitness sharing [114] and Roultewheel based selection, as explained in [115]. After the new velocities for the RSs are computed the new positions for each RS are also computed using Eqn. 4.6. 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.

$$X_i^{j+1} = X_i^j + V_i^j \tag{4.6}$$

At each iteration, non dominated solutions go into the repository (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 continues till the limit on the iterations ($ITER$) is reached. Our optimization procedure terminates then.

The REP maintains a historical record of all the nondominated RSs. To decide a certain RS should be added to the REP or not, the nondominated 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 nondominated RSs, we select one RS based on nadir point computation [111]. This will give us the RS_{opt}, that is 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. 4.6, 20 number of distinct RSs are possible. After performing CROPT we obtained the pareto front as $\{C_1^3, C_2^4\}$, $\{C_1^3, C_2^3\}$. Out of both we choose first set as optimal solution based on Nadir point computation [111].

4.3 Experiments and Experimental Results

In this section, we present details of the experiments we have conducted to substantiate the efficacy of our proposed approach. First, we mention the objectives of our experiments. Next, the experimental setup, which we have used while implementing our proposed method and the results obtained. We also mention the benchmark circuits that we have considered in our experiments. Finally, we compare our results with the related work.

4.3.1 Objectives

Objective of our approach is to compare of the proposed node mapping approach with the standard library based mapping approaches. Our next objective is to estimate the number of possible re-ordering cases with respect to height, width constraints of Domino cell and the effect of carrying out cell re-ordering along the critical path. Finally, to assess the advantages of optimum re-ordering approach in comparison to raw mapping and re-ordered mapping approaches. Also, we aim to compare other mapping approaches with different steps in our proposed methods namely, Raw_Map, Reo_Map and Opt_Map.

4.3.2 Experimental setup

All cells generated during the *Raw_Map* step belong to the Domino logic style. In order to track the connectivity of various gates present in the initial *Dominonetlist*, we used the Berkeley SIS tool, Version 1.3 [113]. The delay savings and area penalty of various Domino cells are computed using simulations performed in $0.18\mu m$ CMOS process, 1.8V, 27^oC. 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 Core2 Duo(2.8 GHz) processor. For applying the node mapping algorithm, we restrict the dimensions of cells, height and width to be 4 and 6, respectively. We increase 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 gate formation operation (*GF_Oper*).

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 and width constraints the number of possible re-ordering sets are shown in Fig. 4.8.

For performing the optimization we coded the MOPSO [115] 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 $N = 4 \times k$ 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 [114]. Also in order to choose the optimal solution from the REP [115], we have used the approach of computing Nadir Point as mentioned in [111]. We aimed to validate our approaches on some standard benchmark circuits. In addition to circuits mentioned in Table 3.3, we have considered some practical circuits like Hans-Carlson adder (HC2.blif), Sparse Kogge Stone Adder (S2-KS2.blif).

4.3.3 Experimental results

Before discussing the experimental results we define two parameters *area ratio* (α), *delay ratio* (β) which we refer to subsequently in this section. The *area ratio* denoted by $\alpha_{p_2}^{p_1}$, gives the ratio of area occupied by a circuit designed using process p_1 to area occupied by the same circuit designed using process p_2. Here, area is measured in number of transistors needed for design. Similarly, the *delay ratio* denoted by $\beta_{p_2}^{p_1}$, gives the ratio of delay produced by a circuit designed using process p_1 to delay produced by the same circuit designed using process p_2.

In our experiments, first we run our node mapping algorithm on the set of benchmark circuits we considered. The overall area (transistor count) and delay obtained using the Raw_Map step (RM) for each circuit is mentioned in columns 4, 5 of Table 4.3, respectively.

Table 4.2: Delay of the cells in ns with various height and width

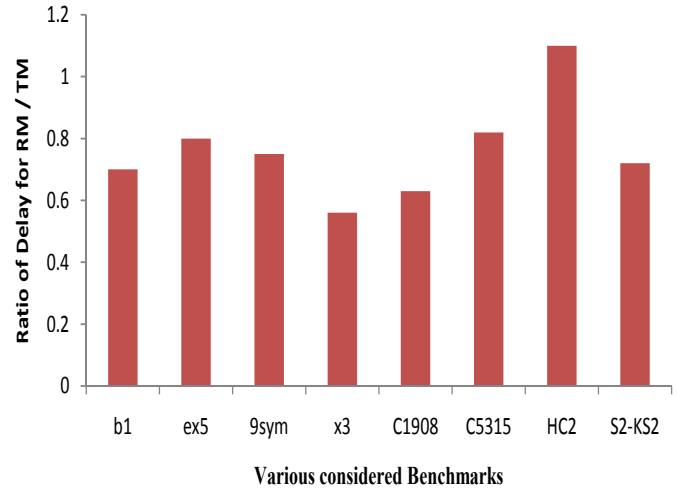
		Width of the cell					
Height of the cell		1	2	3	4	5	6
	1	1.08	1.32	1.56	1.8	1.98	2.11
	2	1.23	1.44	1.67	1.89	2.1	2.34
	3	1.67	1.89	2.13	2.45	2.79	3.12
	4	2.15	2.52	2.81	3.11	3.45	3.81

We also present the area, delay results obtained by implementing a library based technology mapping approach [113], in columns 2, 3, respectively. We can clearly observe that the approach raw mapping performs better both in terms of area and delay. This is possible because the technology mapping 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 4.1). The raw 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 raw mapping approach gave savings of 28% on delay and 19% on area when compared to the result obtained using technology mapping approach. The ratio of area and delay of various circuits obtained using raw mapping approach to that of the values obtained using technology mapping approach namely $\alpha_{TM}^{RM}, \beta_{TM}^{RM}$ are shown in Fig. 4.7(a) and Fig. 4.7(b), respectively.

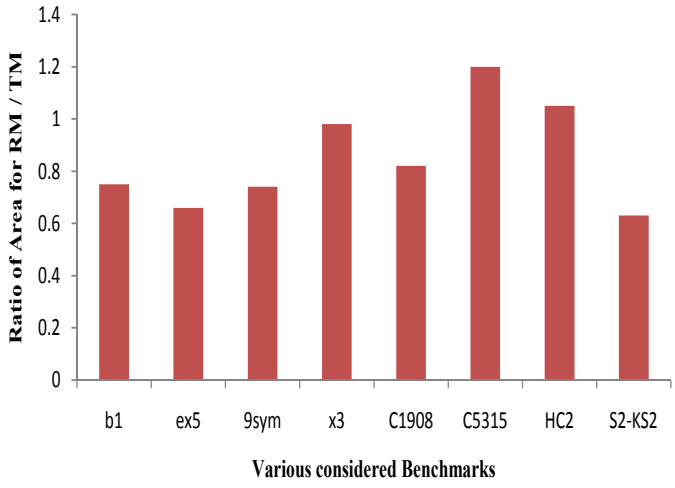
After the *Raw_Map step* is over, we carried out the *Reo_Map* step on various benchmarks. For the given height, width constraints of cell, we derived various possible *original cell* structures and obtained the corresponding re-ordering cell (if exists) for each structure. In Fig. 4.8, we show the number 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 dimension of a cell increases, there is an increase in the number of re-ordering cells. This is due to the fact that, higher dimension limits 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. 4.8, that for a given height, width constraints of cells, there exists a bound on the number of possible re-ordering cells.

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. 4.4 respectively. The approaches *Binpack* [59] and *CCMAP* [96] are also applied on the considered benchmarks and their respective area, delay measures are shown in columns 2 to 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



(a)



(b)

Figure 4.7: (a) Delay comparison of RM/TM, (b) Area comparison of RM/TM

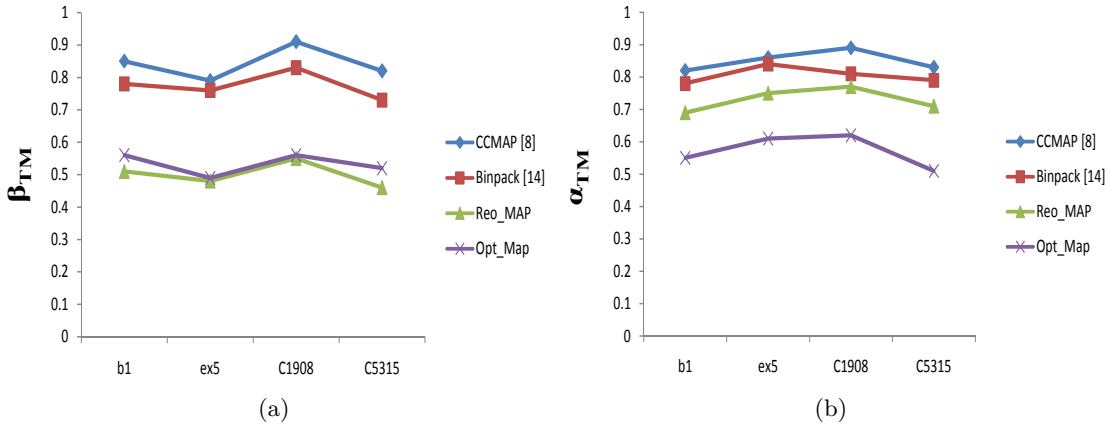


Figure 4.8: For circuit C5315.pla (a) Delay comparison (b) Area comparison

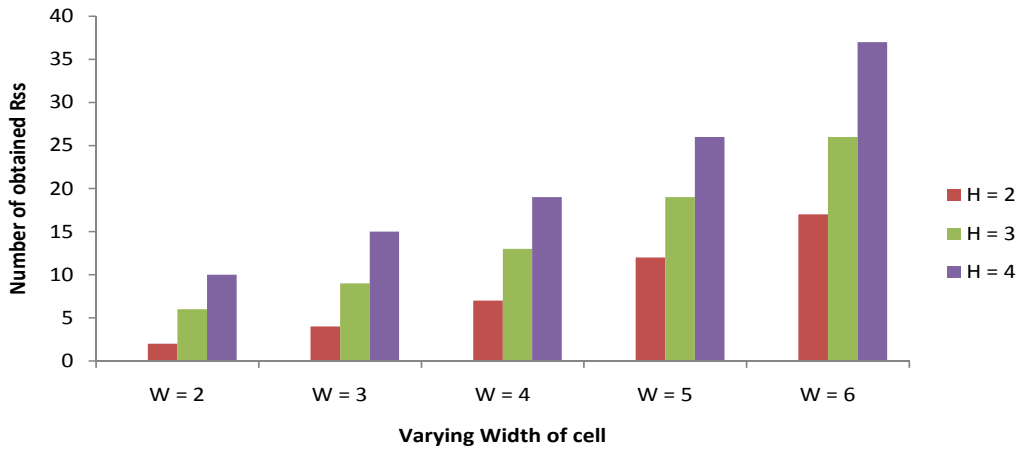


Figure 4.9: Re-ordered set distribution for various dimension cells

critical path. This lead to a improvement in area by 14% with a minimal penalty on delay.

Table 4.3: 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	2816	27.4
HC2	3843	45.7	3395	42.35
S2-KS2	3574	39.8	2252	28.65

Table 4.4: Performance evaluation of various approaches

Circuit	CCMap [96]		Binpack [59]		Reo_Map		Opt_Map	
Name	Area (count)	Delay (ns)	Area (count)	Delay (ns)	Area (count)	Delay (ns)	Area (count)	Delay (ns)
b1	43	38.25	47	35.1	48	22.9	35	25.2
ex5	3215	257.8	3463	236.6	3628	154.7	2638	169.9
9sym	673	335.4	725	307.8	759	201.2	552	220.9
x3	3246	206.1	3496	189.1	3662	123.6	2663	135.7
C1908	927	30.35	999	27.9	1046	18.2	761	19.9
C5315	2327	24.95	2505	22.8	2625	14.9	1909	16.4
HC2	2997	38.85	3266	35.65	3381	23.3	2459	25.6
S2-KS2	2787	33.83	3038	31.1	3145	20.3	2287	22.3

For a particular ISCAS benchmark circuit C1908.pla, w.r.t procedure TM, the delay ratio β_{TM} and the area ratio α_{TM} for various procedures are shown in Fig. 4.9 (a) and 4.9 (b) respectively.

4.4 Conclusion

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 and width flexibility offered by cells is exploited. This enables us to provide an optimum mapped Domino circuit. Such a circuit will be optimum in terms of both critical path delay and area penalty. Also, the proposed mapping approach gives significant delay savings compared to the library based mapping of Domino circuits. Cell re-ordering 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 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.

Chapter 5

Clock Gating for Low Power

In Chapter 4, we focused on improving the delay of Domino block with the help of on-the-fly mapping technique. In this chapter, we focus on reducing the power dissipation of the Domino blocks using clock gating. Clock gating is an effective technique in blocking the charging and discharging of redundant logic in a given circuit [107], [65]. Many works have been reported in recent literature related to the implementation of clock gating. However, majority of them focused on sequential circuits only. Since the outputs of a combinational block solely depends on its inputs, the same technique for clock gating cannot hold true for sequential and combinational circuits simultaneously. Further, many of these works focused on minimizing the routing length of clock, addressing the slew constraints etc. In fact, a method to reduce the redundant switching of gates in Domino circuits needs to be investigated. Constant switching of Domino blocks with every rise in clock pulse is a power hungry activity, which can be alleviated as a clock gating approach, implementing a gating technique. Simultaneously reducing the redundant switching and keeping a control on logic overhead appears to be a challenging task.

Keeping the above scope in view, in this chapter, we propose a clock gating scheme based on pattern recognition technique for Domino circuits. First, we generate a list of gate patterns which can undergo clock gating from the Domino cell library. From these, we identify a list of *favorable gate patterns (FGP)*, that is, patterns that can yield a positive power savings when clock gating is applied. Next, we use an index based sub graph matching algorithm (ISMA) which maps the obtained FGPs to the Domino block of a given circuit.

Finally, we convert our problem into a two objective optimization problem which tries to maximize the overall power savings, with a minimum penalty on area overhead. How we address the issues that arise in identifying the FGPs, mapping the FGPs to the circuit and finding an optimum set of patterns is presented in this chapter.

The rest of the chapter is organized as follows. Some key definitions and lemmas that are needed for describing our work are mentioned in Section 5.1. Our proposed methodology for implementing an optimum clock gating for Domino circuit is described in Section 5.2. The experimental results and comparison with the existing techniques are given in Section 5.3. Finally, Section 5.4 concludes the chapter.

5.1 Some Basic Concepts and Definitions

Our proposed methodology for designing optimum clock gating for Domino circuits involve finding *favorable gate patterns (FGPs)* and a *subgraph matching algorithm*. In this section, we present some basic terminologies which we refer to in our discussion.

Definition 5.1 *Node :* Given a graphical representation of Boolean circuit, each gate of the circuit is called as a node of the corresponding graph.

Definition 5.2 *Link :* Given a graphical representation of Boolean circuit, an interconnection between two nodes (gates) of the graph (circuit) is called as a link of the corresponding graph.

Definition 5.3 *Type of link :* A link to a node can be of two types. An incoming link defined by type '**a**' and an outgoing link defined by type '**A**'. For example, in Fig. 5.1, we can see that link types for node 1 are '**A**','**A**', node 2 are '**a**','**A**' and node 3 are '**a**','**a**', respectively.

Definition 5.4 *Type of node :* Given a graphical representation of a Boolean circuit, each gate of circuit corresponds to a particular node in the graph. Each node belongs to a particular type, based on the functionality of corresponding gate. For example, in Fig. 5.1, type of nodes 1 and 2 is 2-input AND (2A) and type of node 3 is 2-input OR (2O).

Definition 5.5 *Link cardinality table (LCT):* Each link of the graph is associated with a

node. This node may belong to various types depending on the functionality it realizes. The *LCT* contains, for each type of link, number of nodes which are present along with their respective cardinalities. For example, for the graph shown in Fig. 5.1, considering type **a** link, 2A has 1 node and 2O has 1 node.

Definition 5.6 *Favorable gate pattern (FGP):* Adaption of clock gating minimizes redundant switching of gates thereby reducing dynamic power ($red(P_{dyn})$) dissipation but needs additional logic which consumes additional power (P_{addl}). Those gate patterns which can effectively save power, that is having $red(P_{dyn}) - P_{addl} > 0$, are called *favorable gate patterns (FGPs).*

Definition 5.7 *Priority node (PN):* While implementing the subgraph matching algorithm, we match the nodes of subgraph (FGP) to nodes of network (circuit) one by one. At a given instance, the node belonging to the subgraph that is going to be matched is termed as *priority node (PN).* Using the *LCT*, the node which has the least number of possible network nodes, is chosen as *PN.*

Lemma 5.1: A minimum of 3 levels must be present in the gate pattern to undergo clock gating.

Proof: In order to block the redundant switching, that is, to implement clock gating for a gate at level I, the nature of gate at level $I + 1$ must be known. The gating logic that is to be designed for level I should draw the input from the level $I - 1$. Hence, minimum 3 levels should exist in the gate pattern, in order to implement clock gating.

Lemma 5.2: If an AND gate at level I has one of its input turning low (at level $I - 1$), the gates leading to other input (at level $I - 1$) of AND gate need not to be switched.

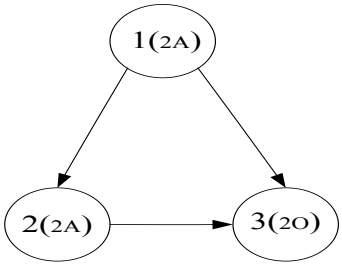


Figure 5.1: An example network

Proof: The output of AND gate is true (1) when all of its input are true. In case one of the inputs turns to be false (0), independent of other input the output of AND gate is false. Hence, for this particular interval the gates leading to other input need not to be switched and can save power.

5.2 Proposed Methodology

In this section, we present our methodology for performing optimum clock gating of Domino circuits. An overview of our proposed methodology is shown in Fig. 5.2. Our overall approach consists of the following operations.

Suppose, given a circuit C_{init} which is completely unate in nature. Let L_{dom} be the Domino cell library used for mapping. We brief the different tasks in our approach as follows.

Pattern generation (Pat_Gen): A pattern generation step, denoted as Pat_Gen, which tries to generate a set of *favorable gate patterns (FGPs)* from a given L_{dom}, such that each individual gate pattern (p_i) is associated with power savings (P_{sav}), area penalty (A_{pen}) with it, simultaneously satisfying the constraint ($P_{sav} > 0$). We can define this as $Pat\_Gen\{L_{dom}\} \rightarrow \{p_1, p_2, \ldots, p_k\}$ where $\{p_1, p_2, \ldots, p_k\}$ are in the set of FGPs.

Pattern matching (Pat_Match): A subgraph matching operation, where the individual

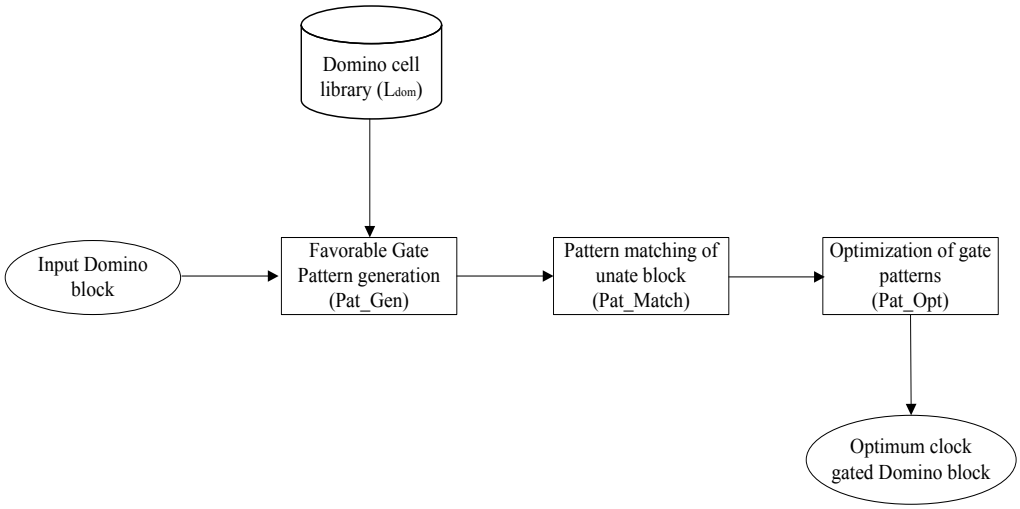


Figure 5.2: Overview of optimum clock gating of Domino circuit

FGPs (subgraphs) are matched to the given circuit (C_{init}) using an *index based subgraph matching algorithm (ISMA)*. We can formalize as, $Pat\_match\{C_{init}, p_i\} \rightarrow \{P^i_{sav}, A^i_{pen}\}$ where p_i is a considered FGP, P^i_{sav}, A^i_{pen} are the obtained power savings and area penalty after matching is performed.

Pattern optimization (Pat_Opt): An optimum set of patterns is explored, which when used, can obtain maximum power savings with a minimum area penalty. Pat_Opt is the optimization operator, $p_1, p_2, \ldots, p_l$ are the l optimum patterns and $\{P^{opt}_{sav}, A^{opt}_{pen}\}$ are the resulting power savings and area penalty after optimization.

In other words, $Pat\_Opt\{U_{opt}, p_1, p_2, \ldots, p_k\} \rightarrow \{p_1, p_2, \ldots, p_l, P^{opt}_{sav}, A^{opt}_{pen}\}$ is an optimum pattern matching with respect to some objectives.

In the following sub sections, we discuss the above mentioned tasks in details.

5.2.1 Pattern generation (Pat_Gen)

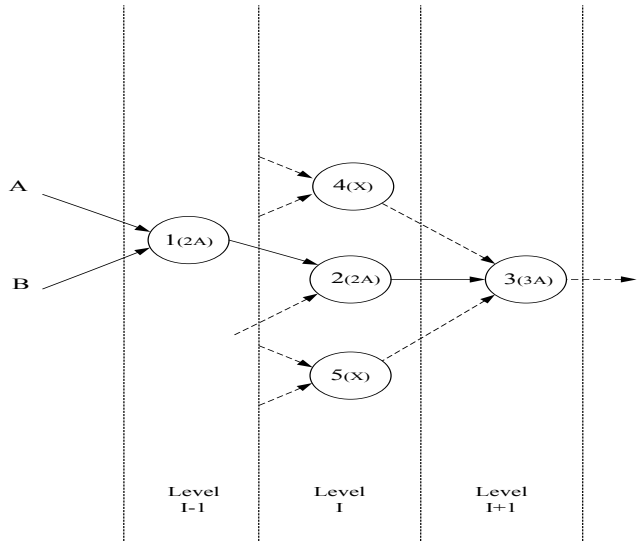
We propose a method for generating patterns that takes a Domino cell library as input and generates FGPs which can implement clock gating and have some fixed specifications. Also, we describe models that we have used in obtaining the power savings (P_{sav}) and area penalty (A_{pen}) for a given FGP.

To start with, we consider an example which has the following gates present in its cell library L_{dom}. The notations we use in extracting FGPs are mentioned below.

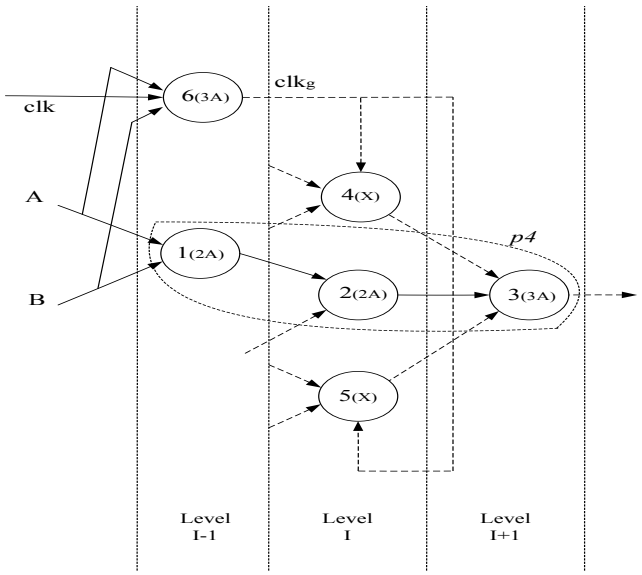
To be members of a given pattern, we select the gates from the Domino cell library L_{dom}. Suppose, the L_{dom} consists of 2 input OR, AND; 3 input OR, AND gates denoted as 2O, 2A, 3O, 3A, respectively.

According to Lemma 5.1, a minimum of three levels are required in a FGP. Hence, we start framing FGPs with a basic minimum of a single gate in each level of the pattern.

Notations used in extracting FGPs	
L_{dom}	Domino cell library
2A	2-input Domino AND gate
3A	3-input Domino AND gate
2O	2-input Domino OR gate
3O	3-input Domino OR gate



(a)



(b)

Figure 5.3: A Scenario of (a) without clock gating, (b) with clock gating using FGP p_4

For the considered cell library, we formulated a possible pattern as shown in Fig. 5.3 (b). Non-clock gated version of this circuit appears in Fig. 5.3 (a). The pattern p_4 (shown in fig. 5.3 (b)) has three levels $(I+1, I, I-1)$ with one gate in each level. For the part of a circuit shown in Fig. 5.3, in order to clock-gate the gates at level I, there is an AND gate (3A) at level $I+1$, which satisfies Lemma 5.2. Next, the condition when one of the input

gates at level I goes low, the remaining gates present at level I, which are providing inputs to the 3A gate, can be clock gated. These are marked as 'X', (see Fig. 5.3). Depending on the input to the gate at $I-1$, (in this case A and B to 2A gate), the clock-gating logic is devised. Here, if either A=0 or B=0 happens, then all the gates (i.e. 4 and 5) would be low. Hence, as shown in Fig. 5.3, we choose the 3A gate for generating the gated clock for both "X" gates.

Estimation of power (P_{sav}): Here, we present a model, which we have adopted to estimate the power savings, obtained by introducing clock gating. We assume that the switching power (P_{dyn}) of gate 'X' (shown in Fig. 5.3), linearly depends on the clock frequency (f_{clk}), as stated in Eqn. 5.1.

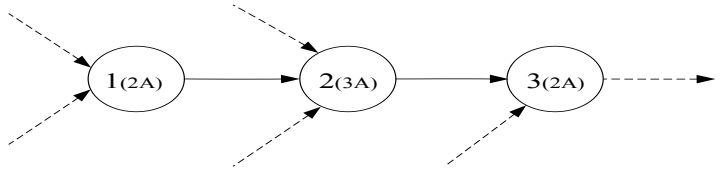
$$P_{dyn} = \frac{1}{2} C_L V_{dd}^2 f_{clk} \tag{5.1}$$

where C_L is the load capacitance, V_{dd} is the supply voltage and f_{clk} is the switching frequency. Assuming that the probability of inputs A, B to be '0' or '1' is $p_A(0)$=$p_B(0)$= 0.5 each, then the probability of gated clock to be '0' denoted by, $p_{clk_g}(0) = 1 - p_A(1)p_B(1)$ $= 1 - 0.25 = 0.75$. This implies that 75% of power savings can be obtained in X gates. This comes at the cost of one additional 3A static gate (gate 6, in Fig. 5.3) which is used as clock gating logic, that consumes additional power (P_{addl}). Hence, the overall power savings can be obtained by adding the power savings of 'X' gates and reducing the extra power penalty caused by clock gating logic. It is formulated in Eqn. 5.2.

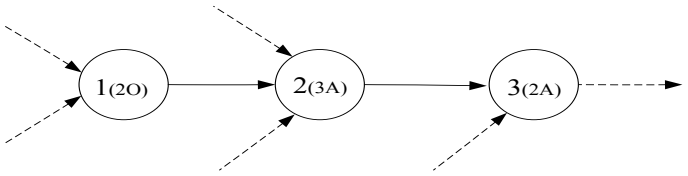
$$P_{sav} = P_{sav}(X_1) + P_{sav}(X_2) - P_{addl} \tag{5.2}$$

Estimation of area (A_{pen}): In our work, to estimate the area overhead caused due to insertion of clock gating, we consider only the gate area. We estimate the penalty in terms of number of transistors. For the considered pattern p_1 (as shown in Fig. 5.3), we employ a 3A static gate as gating logic which needs 8 transistors. Hence, $A_{pen} = 8$, for the considered p_4.

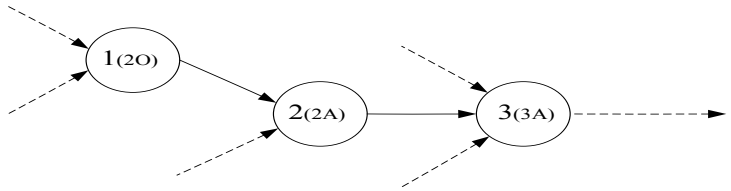
After obtaining the patterns, we consider only those patterns whose $P_{sav} > 0$. From the previously defined library L_{dom}, we got four different FGPs, as shown in Fig. 5.4. These form the set of *favorable gate patterns (FGPs)*. Next, we try to identify the presence of



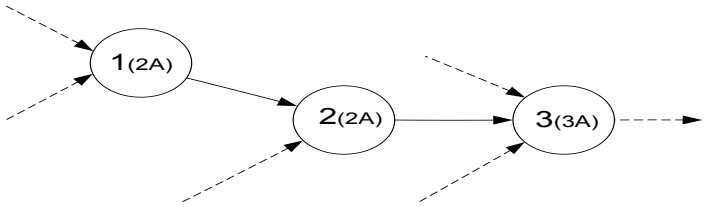
(a) p_1



(b) p_2



(c) p_3



(d) p_4

Figure 5.4: Various possible FGPs for considered L_{dom}

various FGPs in the considered Domino circuit. The detail procedure is mentioned in the following sub section.

5.2.2 Pattern matching (Pat_Match)

After obtaining a set of FGPs which can implement clock gating, we further try to find the presence of these FGPs in a given Domino circuit. To do this we follow an index based subgraph matching algorithm (ISMA) proposed in [120], which finds the presence of a desired sub graph in a given graph. In the following, we describe ISMA algorithm with

the help of an example. The notations used in the algorithm are mentioned below.

To start with, we consider a circuit with 13 Domino gates (as shown in Fig. 5.5), as the graph for which gate patterns are to be matched. The gate pattern p_4 obtained using Pat_gen is used as FGP for matching the circuit.

Various steps in ISMA are shown as the flowchart in Fig. 5.6.

Step 1: In this step, we extract the link cardinality table (LCT) (Definition 5.5) for all link types present in the circuit. These are of types **A** and **a** for our current example.

Step 2: Here, we check for unmarked nodes in the FGP. Initially, all nodes are unmarked in the FGP.

Step 3: We use the LCT of the circuit, to select the *priority node (PN)* from the FGP. For the current example, the LCT is shown in Table. 5.1. From Table. 5.1, we can infer that the gate of type 2O, is having least cardinality. Yet, this type of node doesn't exist in the considered FGP (p_4). Hence we go for the gate which has the next highest cardinality (3 in this case).

Notations used in ISMA algorithm		
LCT	Link cardinality table	// Used for selecting PN
PN	Priority node	// Node that is selected for matching

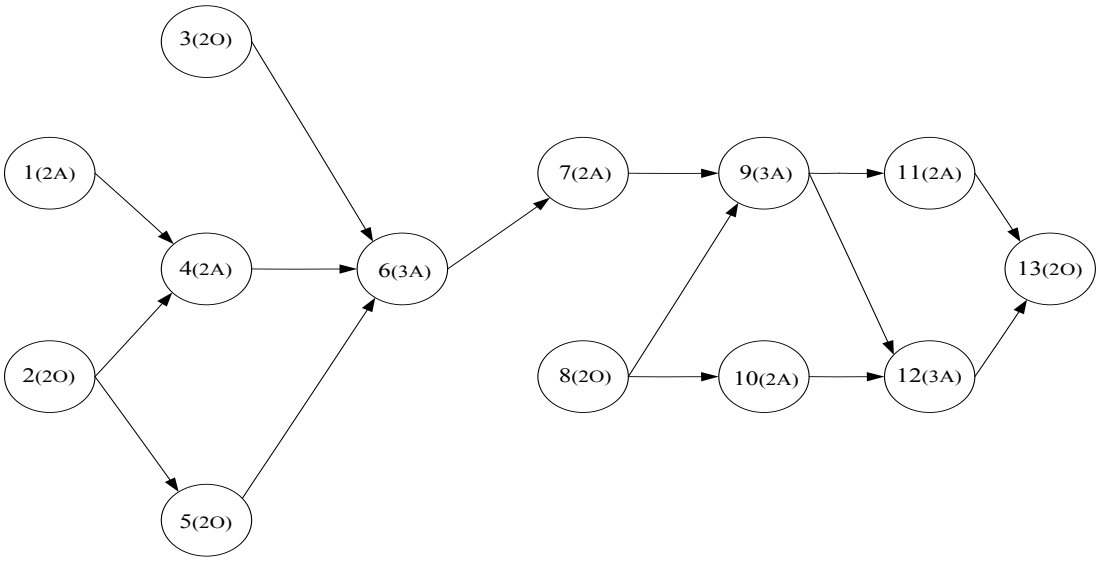


Figure 5.5: Graphical representation a considered Domino circuit

Table 5.1: Link Cardinality Table for circuit graph shown in Fig. 5.5

S. No	Type of link	Type of gate	Gate number	Cardinality
1	A	3A	6, 9, 12	3
		2A	4, 7, 10, 11	4
		2O	2, 3, 5, 8	4
2	a	3A	6, 9, 12	3
		2A	4, 7, 10, 11	4
		2O	5, 13	2

This matched with gate type 3A, having either an incoming link or an outgoing link. We can see that node 3 of FGP, which is a 3A type has the minimum incoming links, that is of type **a**. Hence, the current *PN* will be node 3. We also mark this node.

Step 4: This step checks for the node of type 3A having a link type **a** in the circuit. It matches to node numbers 6, 9, 12 of the circuit. All nodes are equally favorable. We randomly choose circuit node 6 and proceed further. The remaining possible nodes will be considered, if and only if, FGP can't be mapped in this iteration.

Step 5: Since a match exists (FGP node 3 is matched to circuit node 6), we return back to Step 2.

Repeat Steps 2, 3, 4: We still find that FGP nodes 1, 2 are unmarked. To find *PN* for these nodes, we explore the neighboring circuit nodes of 6 from which an incoming link comes to 6. These happen to be type **a** links from circuit nodes 3, 4, 5. Out of these only circuit node 4 is of similar type to FGP node 2. Hence, current *PN* will be FGP node 2. We mark node 2 of FGP and match it to network node 4.

Repeat Steps 5, 2, 3, 4: Since match exists we repeat once again steps 2, 3, 4. In a similar fashion, we get circuit node 1 as the matching node for FGP node 1. As in Step 3, we mark this FGP node. Final checking happens in Step 2. Since all nodes are marked, final matched circuit will be returned.

Step 5: If at all a particular node of FGP has no matching node in the circuit, we say FGP cannot be matched with the circuit. Hence, the algorithm will terminate at that point and state that no match is found. In our example, the final matched circuit, using the given FGP is shown in Fig. 5.7.

If a situation exists, where more than a single match occurs (e.g. nodes 6, 9, 12 in this case), we randomly choose one match (say '6' in this case) and proceed with further steps

of our algorithm. If the entire FGP can't be mapped (successfully mapped in this case), then we consider the other possible matches one after another and apply ISMA on them. If we have to match a set of patterns to the circuit, we first sort the patterns based on their individual power savings. Then we apply our *ISMA* algorithm repeatedly, choosing the patterns in the decreasing order of power savings. This helps us in reducing the complexity of search tree of ISMA. The search tree for the considered example is show in Fig. 5.8.

Next, we perform an optimization on the set of FGPs, to find a best set of FGPs that can cover the entire circuit, yielding maximum power savings with a minimum area penalty.

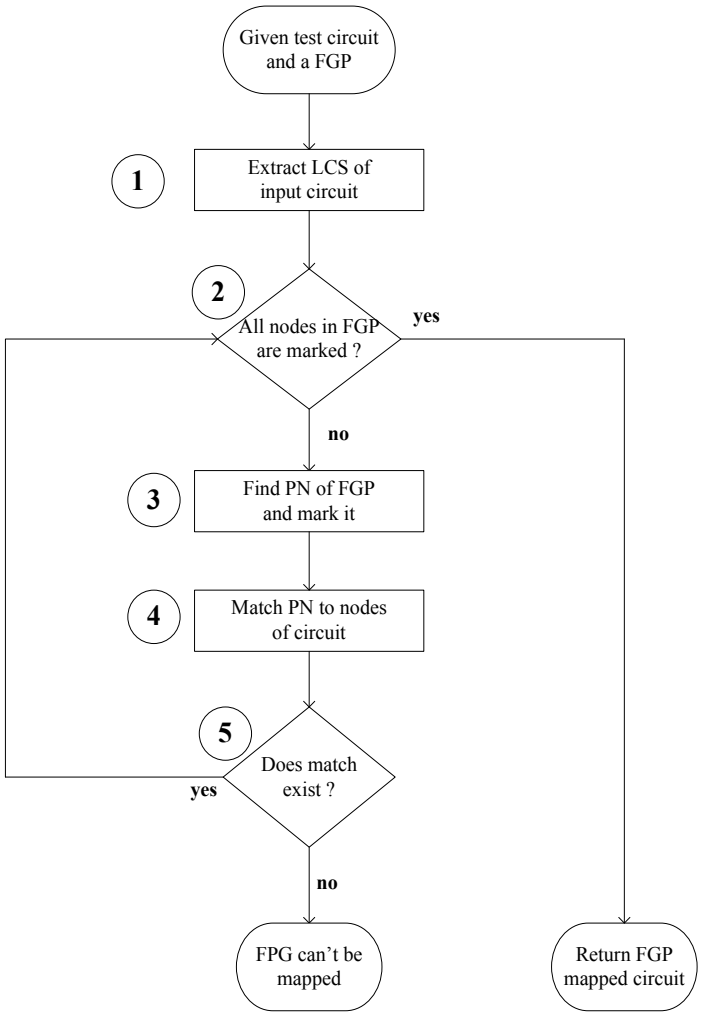


Figure 5.6: Flowchart representation of ISMA algorithm

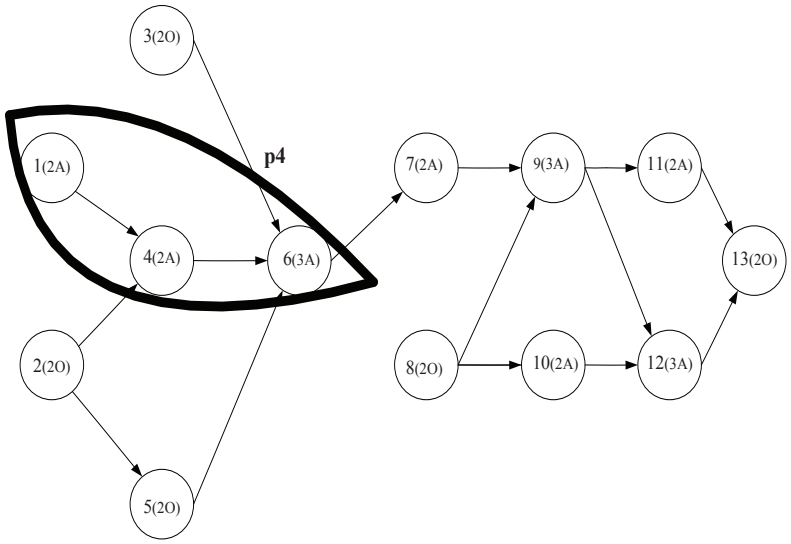


Figure 5.7: FGP mapped circuit

It is discussed in the following sub section.

5.2.3 Pattern Optimization (Pat_Opt)

In this section, first we state the need for finding an optimum set of FGPs used for matching a Domino circuit. Next, we state the problem of optimizing *Pat_Match*, clearly defining the objectives, constraints and design parameters in it. Finally, we suggest a two objective Genetic algorithm based approach to solve the pattern set optimization problem (PSOPT).

After performing mapping of the circuit with a particular FGP, it is observed that using different set of FGPs, termed as pattern set (ps) give different power savings and area penalty (various cases are shown in Fig. 5.9). As a consequence, the mapping of

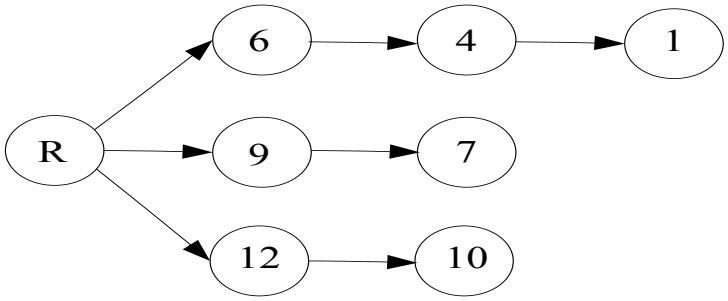
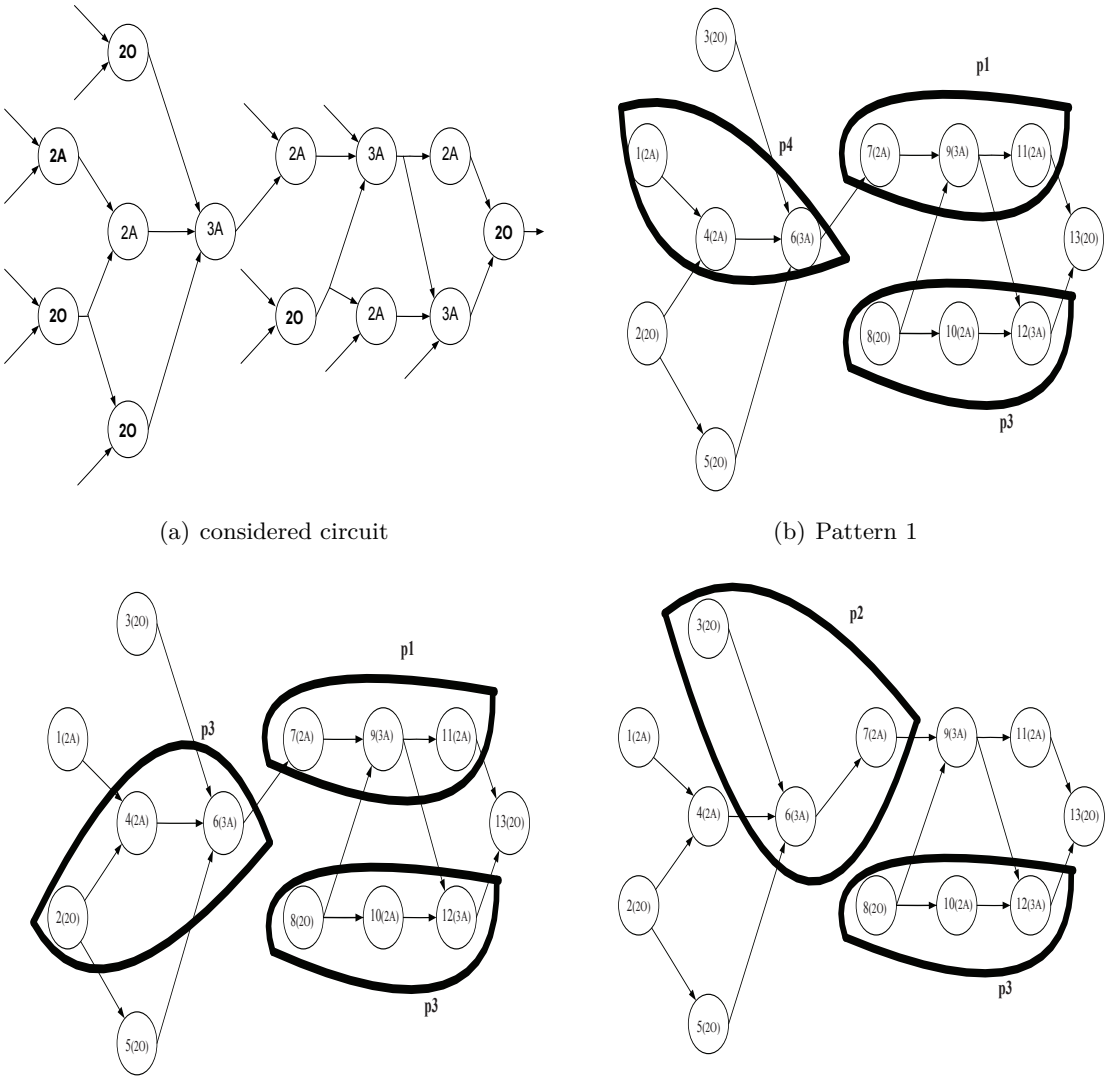


Figure 5.8: Search tree for the considered example



(a) considered circuit

(b) Pattern 1

(c) Pattern 2

(d) Pattern 3

Figure 5.9: An example test circuit and possible matching

Domino circuit with a particular ps may not be optimum in terms of power savings and area penalty. In fact, we have the flexibility to choose a ps that can cover the entire Domino circuit such that the final mapping is optimum in terms of power savings and area penalty. Therefore, a judicial choice must be made, in order to achieve optimum realization for a given *Pat_Match*. We call this problem as pattern set optimization problem (PSOPT). We formally define the PSOPT problem in the following. We refer the following notations in our optimization.

Notations used in solving PSOPT		
ps	:	FGP set
α	:	cooling rate
temp	:	temperature of the iteration
T_{max}	:	maximum allowed temperature
T_{min}	:	minimum possible temperature
iter	:	maximum possible iterations
	:	at a given temperature
HL	:	hard limit for clustering
SL	:	soft limit for clustering
i	:	total number of FGPs
Archive	:	stored set of pss

For a given circuit C_{init}, let $\{p_1, p_2, \ldots, p_i\}$ are the i number of FGPs obtained after *Pat_gen* step. Our objective is to find an optimum pattern (ps_{opt}) resulting a optimum gating of the circuit. We call it as pattern optimization (PSOPT), that is

$$\text{Pat\_Match } (C_{init}, p_1, p_2, \ldots, p_i) \rightarrow \text{Pat\_Opt } (\text{C}_{init}, \text{ps}_{opt}).$$

We consider the following two objective functions to judge the optimality of ps. Suppose, an arbitrary pattern set be $ps_k = \{p_1, p_2, \ldots, p_k\}$. Let $f_p(C_{init}, ps_k)$ denote the power savings obtained by matching the circuit with ps_k. Similarly, $f_a(C_{init}, ps_k)$ denote the estimation of area penalty incurred while performing the matching. We aim to model our PSOPT as a minimization problem. Hence, we define another function P_{sav}^{-1} which is the inverse of P_{sav} and is defined as $P_{sav}^{-1} = (1 + P_{sav})^{-1}$

We define the operation to find an optimum pattern set $\{ps_{opt}\}$ as the Pat_Opt, if it satisfies the following.

Input: Given Pat_Map $(C_{init}, p_1, p_2, \ldots, p_i)$:

$$\begin{aligned} Pat\_Opt\ (C_{init}, ps_{opt}) = & minimize\{A_{pen} = f_a(C_{init}, ps_k)\}, \\ & minimize\{P_{sav}^{-1} = (1 + f_p(C_{init}, ps_k))^{-1}\}, \\ subject\ to \quad & P_{sav}^{-1} \leq P_0,\ A_{pen} \leq A_0,\ \{ps_k\} \subseteq \{ps_1, p_2, \ldots, p_i\}, \\ for\ some\ constraints \quad & P_0, A_0 \end{aligned}$$

SA based optimization: We follow an *Archived Multi Objective Simulated Annealing (AMOSA) [121]* approach to solve the PSOPT problem. A brief explanation of fitting our PSOPT problem into the framework of the AMOSA procedure is mentioned below. We have chosen the AMOSA procedure because it is accurate and computationally efficient with time complexity O(MNlog(N)) compared to other multiobjective optimization approaches [122], [123]. Also, this procedure doesn't discard an improbable candidate straight away as is done in other domination based approaches [124], [125]. Various steps followed in the AMOSA procedure are mentioned below.

We have followed a binary encoding to define a chromosome in AMOSA. As shown in Fig. 5.10, first we form an *Archive* of pss. To do this, we select $\gamma \times SL$ number of pss ($0 < \gamma < 1$), in the *Archive*. Later, by using a simple hill climbing technique, we refine these pss, accepting a ps if only it dominates the previous one. It is continued over sufficient number of iterations such that the pss are non-dominating to each other. In case the number of pss exceed *HL* clustering is applied to reduce the number of pss to *HL*. Using the clustering technique, preserves the diversity of solutions in the *Archive* [121]. The size of cluster is allowed up to $SL > HL$ and later grouped into *HL* size clusters [121]. This helps in reducing the clustering calls. The clustering technique namely Single linkage algorithm [126] is used for clustering.

In AMOSA procedure, we used the concept of amount of domination in computing the acceptance probability of a new set of ps. Given two pattern sets ps_1 and ps_2, the amount of domination is defined as,

$$\Delta dom_{ps_1,ps_2} = \frac{f_p(C_{init},ps_1)-f_p(C_{init},ps_2)}{R_1} \times \frac{f_a(C_{init},ps_1)-f_a(C_{init},ps_2)}{R_2}$$

where, R_1 and R_2 are the ranges of power and area objectives, respectively. These ranges are computed from the solutions present in the *Archive* along with the new and current pss used for computing it [121]. This amount of domination is used for computing the acceptance probability of a new set of FGP.

At the beginning of AMOSA process, one of the ps from the archive is randomly selected as an initial solution at $temp = T_{max}$. The current ps is perturbed to generate a new solution, representing a new ps. The domination status of new ps is checked with the

current ps and pss present in the *Archive*. It gives rise to the following three cases:

Case I : If the current *ps* dominates the new ps and $k(k \geq 0)$ pss from the *Archive* dominate the new ps, then we follow *selection procedure I*, as mentioned [121]. The new ps is selected as the current ps with

$$probability = \frac{1}{1 + exp(\Delta dom_{avg} \times temp)}, \tag{5.3}$$

where $\Delta dom_{avg} = ((\sum_{i=1}^{k} \Delta dom_{i,new-ps}) + \Delta dom_{current-ps,new-ps})$.

Case II : If the current ps and the new ps are non-dominating with each other, then we follow *selection procedure II*. The new ps is compared with the pss in *Archive*. Based on whether it is dominated fully, dominated by some pss or completely non dominated the new ps is either accepted with a *probability* (as mentioned in Eqn. 5.3), or made as the current ps or stored in *Archive* replacing all other existing pss.

Case III : Finally, if new ps dominates the current ps, *selection procedure III* is followed. The new ps is checked with other pss in the *Archive*. If k pss from the *Archive* dominate the new ps then, the ps from the *Archive* which amounts for minimum difference of domination is selected as current ps with a

$$probability = \frac{1}{1 + exp(-\Delta dom_{min})} \tag{5.4}$$

where (Δdom = minimum of difference of domination amounts between new ps and pss present in the *Archive*). If the new ps dominates k pss of the *Archive*, then the new ps is made as the current ps and all the k existing pss are removed from the *Archive*.

The above process is repeated *iter* number of times at each *temp*. Based on the cooling rate α, temperature is reduced to $\alpha \times temp$ at every stage, till the minimum temperature (T_{min}) is attained. After the process is terminated, the pss present in *Archive* form the non-dominated solutions. From these, we choose the best ps based on *Nadir point* computation as mentioned in [111]. This forms the optimum pattern set (ps_{opt}), which yields maximum power savings with minimum area penalty, when clock gating is applied.

5.3 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.

5.3.1 Objectives

We validate our mapping approach on a set of benchmark circuits. Initially, we present the proportion of FGP in the total possible gate pattern for a given cell library. Also, we present the variation of number of FGPs with increase in number of gate levels in a pattern. Next, we estimate the performance of circuits with reference to the operations Pat_Gen, Pat_Match and Pat_Opt. Also, we compare the performance of Pat_Match approach with the no clock gating approach. Finally, we compare our approach with the existing approaches in realizing clock gated Domino logic circuits.

5.3.2 Experimental setup

The FGPs generated according to the Pat_Gen algorithm are completely based on the Domino cell library L_{dom} which is used for mapping a Domino circuit. In order to track the connectivity of various gates present in the Domino netlist, we used the Berkely SIS tool, Version 1.3 [113]. The power savings of Domino gates and area penalty of the gating logic are computed using simulations performed in $0.18\mu m$ CMOS process, 1.8V, 27^oC. Since, area penalty is computed in terms of number of transistors, we got the transistor count of the gating logic from L_{stat} which is used to map the binate block of mixed CMOS circuit. We use static logic gates only for generating clock gating signals.

The ISMA algorithm is implemented 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 ISMA algorithm, we restrict the size of FGP to 3 levels only, since it is the minimum number of levels required to implement clock gating. We

mapped the available FGPs to this connectivity graph one after another, giving top priority to the ones which have high power savings. Overall power savings and area penalty is computed by summing up the individual power savings and area penalty of individual FGPs. For the FGP's shown in Fig. 5.4, the P_{sav} and A_{pen} are reported in Table 5.2. For the sake of simplicity, in Table 5.2, the power savings are mentioned in terms of power dissipation of standard 3-input Domino AND gate P_{3A}.

For performing the optimization, we implement AMOSA procedure in C programming language and compiled using GCC compiler. Experiments are performed in a similar environment as used for implementing ISMA algorithm. T_{max} and T_{min} are set to be $20,000^oC$ and 1^oC respectively [122]. The *cooling rate* (α) is chosen to be 0.5. Maximum number of iterations at a given temperature is set to be 25. HL and SL in our experiment are chosen as 5% and 10% of total population [121]. We aimed to validate our approaches on some standard benchmark circuits. In addition to circuits mentioned in Table 3.3, we have considered some daily life circuits like Hans-Carlson adder (HC2.blif), Sparse Kogge Stone Adder (S2-KS2.blif).

5.3.3 Experimental results

For the considered example of Section 5.2, an estimate of total number of possible gate patterns (PGPs) and number of FGPs is shown in Fig. 5.10. For the example, we enlisted four gate patterns for which clock gating can be applied. For each pattern various gating architectures are possible. Only some of them yield benefits in terms of power. Out of the 20 possible gating architectures only four of them result in positive power savings. Hence, for this case the FGP are 20% of the total possible gate patterns. We have also computed the possible gating architectures keeping number of gate levels 3, 4, 5 (shown in Fig. 5.11).

Table 5.2: P_{sav} and A_{pen} for FGPs shown in Fig. 5.4.

S. No	FGP name	Power savings (quantified)	Area penalty (transistor count)
1	p_1	$0.75(P_{3A})$	8
2	p_2	$0.25(P_{3A})$	12
3	p_3	$0.25(P_{3A})$	12
4	p_4	$0.75(P_{3A})$	8

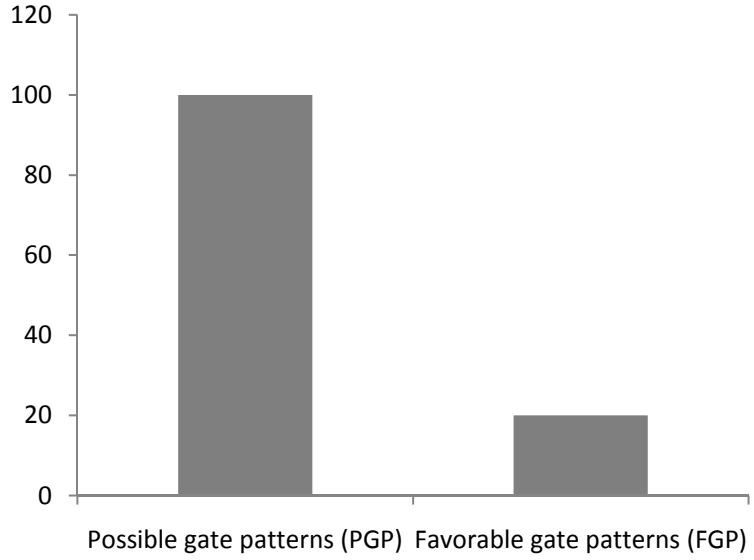


Figure 5.10: Percentage of favorable gate Patterns compared to total possible gate patterns obtained for the considered example

We can see that there is a steep rise in the possible gating architectures. This signifies the exponential rise in the number of gating architectures with increase in number of gate levels in the pattern.

We have carried out various operations mentioned in our methodology, on the chosen set of benchmarks. Respective power dissipation of the circuit in uW and total area in

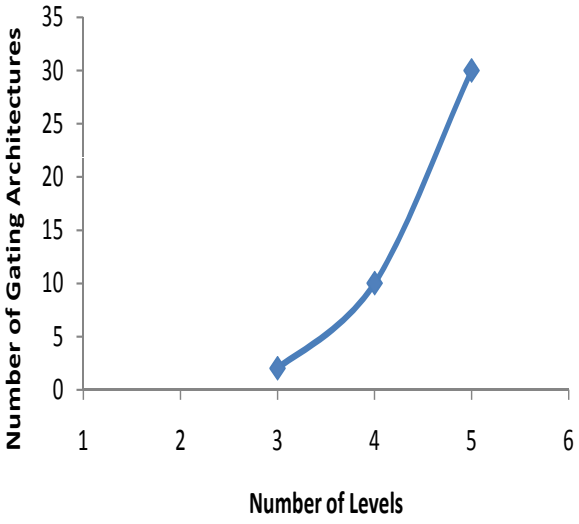


Figure 5.11: Number of possible gating architectures with increase in gate levels

terms of transistor count are shown in Table 5.3. We have also computed the results using approaches mentioned in [83] and [65]. These are shown in columns 2 to 5 of Table 5.3. Columns 6 to 9 mention the power and area results obtained from initial pattern matching (Pat_Match) and optimum pattern (Pat_Opt) matching approach. For a particular test case *ex5.pla* the power consumed without clock gating approach of Subirats et al. [83], is more than that consumed by Banerjee et al. [65]. The similar trends follow for other test cases too. This supports the motivation behind implementing clock gating logic. Note that for the same test case approach [65] needed 91 transistors which is 40% more than the non-clock gating approach.

Analysis of Table. 5.3 shows that *Pat_Match* operation produced 35% power savings with an area penalty of 20%. This is further improved, after applying *AMOSA* based optimization. The high area penalty for approach [65], could be ascertained to the fact that, it excessively used 2-input AND/OR gates. Also, usage of Bubble pushing algorithm in approach [65], lead to high logic duplication and hence affected the obtained power savings. Pattern recognition based clock gating approach has shown 15% better power savings, since it had less gating logic. Each pattern is considered, if only it can yield power savings. The optimization process further offered 20% improvement in power savings and 8% area penalty. For a particular case, C5315.pla we have computed the percentage increase of power savings and area penalty for various approaches like *Pat_Match*, *Pat_Opt* and FCG [65] with respect to work done in [83]. These results are shown in Fig. 5.12. In the same figure, we have also presented comparison of our work with the approaches mentioned in Safeeen et al. [108], Hurst et al. [64]. Observation of these results show that, Safeen's approach

Table 5.3: Comparison of power dissipation and area penalty

Circuit Name	OUD [83]		FCG [65]		Initial matching (Pat_Match)		Optimum Matching (Pat_opt)	
	P (uW)	A(tr.count)	P (uW)	A(tr.count)	P (uW)	A(tr.count)	P (uW)	A(tr.count)
b1	18.4	57	32.4	91	12.4	72	10.1	68
ex5	269.3	2635	392.5	3982	208.5	2917	176.3	2732
9sym	288.3	592	433.7	793	210.3	663	180.4	621
x3	3002.1	2654	4890.8	3847	2204.4	2931	1897.4	2812
C1908	2693.4	535	3934.1	925	1832.3	596	1539.7	557
C5315	12835.3	2314	16530.4	3682	8342.4	2602	6324.5	2499
HC2	14742.8	2754	19832.6	4133	9632.3	3032	8013.2	2892
S2-KS2	15331.5	2532	21632.4	3989	9934.4	2734	8543.4	2681

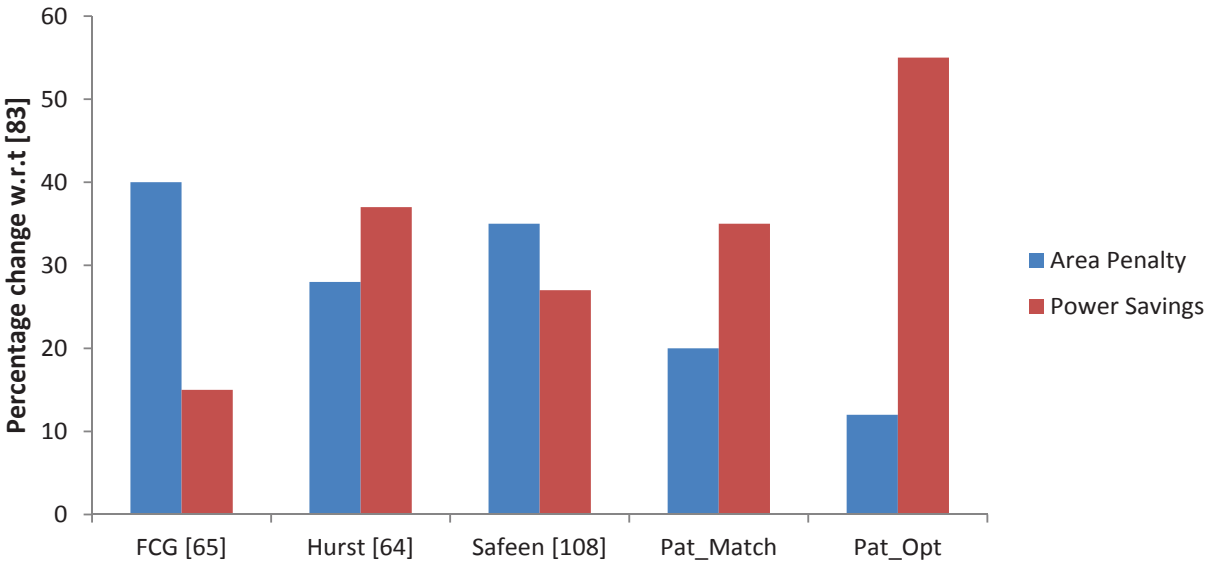


Figure 5.12: Percentage change of power savings and area compared to non clock gated approach of [83] for the circuit C5315.pla

produced more area penalty, as the approach is oriented specifically for FPGAs (which includes both coarse grain and fine garing clock gating). However, Hurst et al. approach gave more power savings than Safeen's approach as it involved finding of maximum gating condition obtained from pruning of gating candidates.

5.4 Conclusion

A pattern recognition based clock gating for mixed static Domino logic circuits is proposed in this work. In order to implement clock gating for dynamic circuits, additional logic and routing is required. If not taken into consideration, it leads to increase in area and power dissipation of the overall circuit. This work proposes to obtain an optimum clock gated circuit. Such a circuit will be optimum in terms of both power dissipation and area. Also the proposed clock gating approach gives significant power savings compared to the non clock gated circuits. Pattern recognition based approach for clock gating is comparable with other clock gating techniques reported elsewhere. We may conclude that our proposed clock gating approach especially suits for low power applications like hand held gadgets, rechargeable devices etc.

Chapter 6

Conclusion and Future Research

In this thesis, we have proposed an approach to synthesize VLSI circuits using mixed static Domino CMOS logic style. The main objective of our research is to synthesize Boolean circuits targeting low power dissipation and offering high performance. In this Chapter, we present the various contributions of this thesis. A critical review of our research achievements vis-a-vis objectives is presented in Section 6.2. Finally, in Section 6.3, we present the scope for future work and directions for further extensions of our research.

6.1 Contribution of this Thesis

The main contributions of our research work can be summarized as follows.

Unate Decomposition: The first contribution of this thesis is an approach to decompose a Boolean logic suitable for realization of a mixed static Domino circuit. In order to realize a circuit using Domino logic, it must be completely unate. However, complete unate circuit is impractical. This work proposes an approach to obtain an optimum unate binate circuit. Such a circuit can be synthesized reducing power, area and delay. Given a circuit, we perform an initial unate decomposition (IUD) which we optimize to reduce power dissipation and delay.

On-the-fly Mapping: Our next contribution in this dissemination is a cell re-ordering based on-the-fly mapping approach for Domino logic circuits. In order to map Domino logic cells the height, width flexibility of cells has been investigated, which in turn leads to decrease in area, delay of the overall circuit. 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 an initial on-the-fly mapping of Domino logic using a node mapping algorithm. Later, we perform the re-ordering of the cells along the critical path so that the delay can be minimized. Finally, we find an optimum set of re-ordering of cells 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.

Clock Gating: A pattern recognition based clock gating for Domino logic circuits is the third contribution of our work. In order to implement clock gating for dynamic circuits, additional logic and routing is required. If proper care is not taken, it leads to increase in area and power dissipation of the overall circuit. This work proposes to obtain an optimum clock gated circuit. Such a circuit will be optimum in terms of both power dissipation and area penalty. Given a Domino block, first we generate the favorable gate patterns from the Domino cell library and map the circuit using the available favorable gate patterns. Later we find the optimum set of patterns that can yield maximum power savings with a minimum penalty on area.

6.2 Significance of our Research

The proposed decomposition based approach presented in Chapter 3, yields lower transistor count compared to static CMOS logic style. Mixed CMOS circuits are comparable with only dynamic and only static realizations according to works reported elsewhere. We may conclude that mixed CMOS circuit is suitable for low power and high speed applications such as mobile and handheld digital gadgets etc.

The cell re-ordering based mapping approach presented in Chapter 4, offered a significant delay advantage compared to the standard Domino mapping technique. Also, the area penalty imposed while cell re-ordering is kept as minimal as possible. Especially, this approach makes the design suitable for high performance applications.

Significant power savings were recorded by using the pattern recognition based clock gating approach, presented in Chapter 5. The clock signal, which is highly active in the Domino block is thoroughly dealt with in this chapter. This made the overall design

comparable to other low power approaches reported in the literature. This encourages the design to be used in various application of handheld gadgets.

6.3 Future Scope of Work

This work however leaves a number of issues opens and problems to address giving a scope for further extension of this research. We mention few such issues.

The decomposition methods can be further generalized for incompletely specified Boolean functions.

Also, various other approaches for multi objective optimization and choosing of the best candidate from the Pareto optimal front can be explored.

The raw mapping approach considered the gates to be of "and", "or" natures only. Further functionalities can be considered in the initial mapped circuit and new set of rules can be derived accordingly for combination operations. Concepts like average case delay in terms of incompletely defined Boolean functions and usage of logical effort in estimating the delay can also be employed.

Our cell mapping strategies attempted to optimize area and delay in the proposed technique. We can also study the effects on parameters like power delay product and energy delay product so that circuits with low power and high performance can be designed.

The clock gating logic is optimized using our proposed approach in terms of switching power and area. However, the gating logic significantly affects the delay of the circuit. Hence performance parameter can also be included in the analysis.

Further, precise models based on deterministic clock gating can be explored which will help in pipelining of circuits.

Bibliography

[1] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, "Near-threshold Computing: Reclaiming Moore's Law through Energy Efficient Integrated Circuits," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 253–266, 2010.

[2] J. Wu, Y.-L. Shen, K. Reinhardt, H. Szu, and B. Dong, "A Nanotechnology Enhancement to Moore's Law," *Applied Computational Intelligence and Soft Computing*, vol. 2, pp. 2–15, 2013.

[3] "Moore's law ," http://en.wikipedia.org/wiki/Moore's_law, accessed: 2015-01-26.

[4] N. H. Khan, S. M. Alam, and S. Hassoun, "Power Delivery Design for 3-D ICs Using Different Through-Silicon Via (TSV) Technologies," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 4, pp. 647–658, 2011.

[5] S.-M. Kang and Y. Leblebici, *CMOS Digital Integrated Circuits*. Tata McGraw-Hill, 2003.

[6] J. M. Rabaey, A. P. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits*. Prentice Hall, 2002.

[7] A. P. Chandrakasan, W. J. Bowhill, and F. Fox, *Design of High Performance Microprocessor Circuits*. Wiley-IEEE, 2000.

[8] V. G. Oklobdzija, B. R. Zeydel, H. Dao, S. Mathew, and R. Krishnamurthy, "Energy-delay Estimation Technique for High Performance Microprocessor VLSI Adders," in *Proceedings of 16th IEEE Symposium on Computer Arithmetic*, 2003, pp. 272–279.

[9] M. Alioto, G. Palumbo, and M. Pennisi, "Understanding the Effect of Process Variations on the Delay of Static and Domino Logic," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 18, no. 5, pp. 697–710, 2010.

[10] F. Carbognani, F. Buergin, N. Felber, H. Kaeslin, and W. Fichtner, "Transmission Gates Combined with Level-restoring CMOS Gates Reduce Glitches in Low-power

Low-frequency Multipliers," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 16, no. 7, pp. 830–836, 2008.

[11] B. Guenin, "When Moore is Less: Exploring the 3rd Dimension in IC Packaging," *Electr. Cool*, vol. 15, no. 1, pp. 24–27, 2009.

[12] "Low power VLSI chip design: Circuit Design Techniques ," http://www.eeherald.com/section/design-guide/Low-Power-VLSI-Design.html, accessed: 2014-12-25.

[13] B. R. Zeydel, D. Baran, and V. G. Oklobdzija, "Energy-efficient Design Methodologies: High-performance VLSI Adders," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 6, pp. 1220–1233, 2010.

[14] S. Wimer and I. Koren, "Design Flow for Flip-Flop Grouping in Data-Driven Clock Gating," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2014.

[15] R. L. Geiger, P. E. Allen, and N. R. Strader, *VLSI Design Techniques for Analog and Digital Circuits*. McGraw-Hill, 1990.

[16] L.-T. Wang, C.-W. Wu, and X. Wen, *VLSI Test Principles and Architectures: Design for Testability*. Academic Press, 2006.

[17] N. H. Weste and K. Eshraghian, *Principles of CMOS VLSI design*. Addison-Wesley Reading, MA, 1993.

[18] W. Wolf, *Modern VLSI Design: System-On-Chip Design*. Pearson Education, 2002.

[19] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*. Newnes, 2013.

[20] M. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. Springer, 2000.

[21] G. Martin, B. Bailey, and A. Piziali, *ESL Design and Verification: A Prescription for Electronic System Level Methodology*. Morgan Kaufmann, 2010.

[22] M. Margala and N. G. Durdle, "Noncomplementary BiCMOS Logic and CMOS Logic for Low-Voltage, Low-Power Operation-A Comparative Study," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 10, pp. 1580–1585, 1998.

[23] Y. Berg, "Static Ultra Low Voltage CMOS Logic," in *Proceedings of IEEE NORCHIP*, 2009, pp. 1–4.

[24] R. Kumar and V. K. Pandey, "Low Power Combinational Circuit Based on Pseudo NMOS Logic," *International Journal of Enhanced Research in Science Technology and Engineering*, vol. 3, no. 3, pp. 452–457, 2014.

[25] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS Digital Design," *IEICE Transactions on Electronics*, vol. 75, no. 4, pp. 371–382, 1992.

[26] N. H. Weste and D. M. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*. Pearson Education India, 2005.

[27] S. A. Mondal, S. Talapatra, and H. Rahaman, "Analysis, Modeling and Optimization of Transmission Gate Delay," in *3rd Asia Symposium on Quality Electronic Design*, 2011, pp. 246–253.

[28] V. M. Srivastava, R. Patel, H. Parashar, and G. Singh, "Reduction in Parasitic Capacitances for Transmission Gate with the help of CPL," in *International Conference on Recent Trends in Information Telecommunication and Computing*, 2010, pp. 218–220.

[29] P. K. Meher, S.-F. Hsiao, C.-S. Wen, and M.-Y. Tsai, "Low-Cost Design of Serial-Parallel Multipliers Over GF (2ˆ m) Using Hybrid Pass-Transistor Logic (PTL) and CMOS Logic," in *Proceedings of the International Symposium on Electronic System Design*, 2010, pp. 131–134.

[30] P. Buch, A. Narayan, A. R. Newton, and A. Sangiovanni-Vincentelli, "Logic Synthesis for Large Pass Transistor Circuits," in *Proceedings of International Conference on Computer Aided Design*, 1997, pp. 663–670.

[31] K. Yano, Y. Sasaki, K. Rikino, and K. Seki, "Top-down Pass-transistor Logic Design," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 6, pp. 792–803, 1996.

[32] L. Gao, "High performance Complementary Pass Transistor Logic Full Adder," in *Proceedings of the International Conference on Electronic and Mechanical Engineering and Information Technology*, 2011, pp. 4306–4309.

[33] C. P. Kumar, S. Tripathy, and R. Tripathi, "High Performance Sequential Circuits with Adiabatic Complementary Pass Transistor Logic (ACPL)," in *Proceedings of IEEE Region 10 Conference*, 2009, pp. 1–4.

[34] Y.-T. Lai, M. Pedram, and S. B. Vrudhula, "BDD Based Decomposition of Logic Functions With Application to FPGA Synthesis," in *Proceedings of the* 30^{th} *International Design Automation Conference*, 1993, pp. 642–647.

[35] D. Suvakovic and C. Salama, "Two Phase Non-Overlapping Clock Adiabatic Differential Cascode Voltage Switch Logic," in *Proceedings of International Solid-State Circuits Conference*, 2000, pp. 364–365.

[36] K. M. Chu and D. L. Pulfrey, "A Comparison of CMOS Circuit Techniques: Differential Cascode Voltage Switch Logic versus Conventional Logic," *IEEE Journal of Solid-State Circuits*, vol. 22, no. 4, pp. 528–532, 1987.

[37] W.-H. Paik, H.-J. Ki, and S.-W. Kim, "Push-pull Pass-transistor Logic Family for Low Voltage and Low Power," in *22nd European Solid-State Circuits Conference*, 1996, pp. 116–119.

[38] P. H.-J. KI, WOO-HYUN and S.-W. KIM, "Low Power Logic Design using Push-pull Pass-Transistor Logics," *International Journal of Electronics*, vol. 84, no. 5, pp. 467–478, 1998.

[39] T. Kuroda and T. Sakurai, "Overview of Low-power ULSI Circuit Techniques," *IEICE Transactions on Electronics*, vol. 78, no. 4, pp. 334–344, 1995.

[40] M. H. Anis, M. W. Allam, and M. I. Elmasry, "Energy-efficient Noise-tolerant Dynamic Styles for Scaled-down CMOS and MTCMOS Technologies," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 10, no. 2, pp. 71–78, 2002.

[41] L. Ding and P. Mazumder, "On Circuit Techniques to Improve Noise Immunity of CMOS Dynamic Logic," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 12, no. 9, pp. 910–925, 2004.

[42] M. W. Allam, M. H. Anis, and M. I. Elmasry, "High-speed Dynamic Logic Styles for Scaled-down CMOS and MTCMOS Technologies," in *Proceedings of the International Symposium on Low Power Electronics and Design*, 2000, pp. 155–160.

[43] J. Yuan and C. Svensson, "High-speed CMOS Circuit Technique," *IEEE Journal of Solid-State Circuits*, vol. 24, no. 1, pp. 62–70, 1989.

[44] C. Efstathiou, Z. Owda, and Y. Tsiatouhas, "New High-Speed Multioutput Carry Look-Ahead Adders," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 60, no. 10, pp. 667–671, 2013.

[45] A. Guar and H. Mahmoodi, "Impact of Technology Scaling on Performance of Domino Logic in Nano-scale CMOS," in *20th International Conference on VLSI and System-on-Chip*, 2012, pp. 295–298.

[46] G. Palumbo, M. Pennisi, and M. Alioto, "A Simple Circuit Approach to Reduce Delay Variations in Domino Logic Gates," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 10, pp. 2292–2300, 2012.

[47] A. Peiravi and M. Asyaei, "Current-comparison-based Domino: New Low-leakage High-speed Domino Circuit for Wide Fan-in Gates," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 21, no. 5, pp. 934–943, 2013.

[48] Y. Sun and V. Kursun, "Low-power and Compact NP Dynamic CMOS Adder with 16nm Carbon Nanotube Transistors," in *IEEE International Symposium on Circuits and Systems*, 2013, pp. 2119–2122.

[49] S. M. Mahmood and Y. Berg, "Ultra-low Voltage and High Speed NP Domino Carry Propagation Chain," in *IEEE Faible Tension Faible Consommation*, 2013, pp. 1–4.

[50] Y. Sun and V. Kursun, "Carbon Nanotubes Blowing New Life Into NP Dynamic CMOS Circuits," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 2, pp. 420–428, 2014.

[51] M. Aigner, S. Mangard, R. Menicocci, N. Olivieri, G. Scotti, and A. Trifiletti, "A Novel CMOS Logic Style with Data Independent Power Consumption," in *IEEE International Symposium on Circuits and Systems*, 2005, pp. 1066–1069.

[52] M. Anis, M. Allam, and M. Elmasry, "Impact of Technology Scaling on CMOS Logic Styles," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 49, no. 8, pp. 577–588, 2002.

[53] K.-W. Kim, C. L. Liu, and S.-M. Kang, "Implication Graph Based Domino Logic Synthesis," in *Proceedings of the International Conference on Computer-Aided Design*, 1999, pp. 111–114.

[54] M. R. Prasad, D. Kirkpatrick, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Domino Logic Synthesis and Technology Mapping," in *Proceedings of International Workshop on Logic Synthesis*, 1997.

[55] D. M. Parmar, M. Sarma, and D. Samanta, "A Novel Approach to Domino Circuit Synthesis," in *Proceedings of 20th International Conference on VLSI Design*, 2007, pp. 401–406.

[56] A. Pal and A. Mukherjee, "Synthesis of Two-level Dynamic CMOS Circuits," in *IEEE Computer Society Workshop VLSI*, 1999, pp. 82–92.

[57] D. Samanta, A. Pal, and N. Sinha, "Synthesis of High Performance Low Power Dynamic CMOS Circuits," in *Proceedings of Asia and South Pacific Design Automation Conference*, 2002, pp. 99–105.

[58] J. Jacob and A. Mishchenko, "Unate Decomposition of Boolean Functions," in *Proceedings of International Workshop in Logic Synthesis*, 2001, pp. 66–71.

[59] K. Yoshikawa, S. Inui, Y. Hagihara, Y. Nakamura, and T. Yoshimura, "Domino Logic Synthesis System and its Applications," *Journal of Circuits, Systems, and Computers*, vol. 15, no. 2, pp. 277–287, 2006.

[60] A. Mishchenko, S. Cho, S. Chatterjee, and R. Brayton, "Combinational and Sequential Mapping with Priority Cuts," in *Proceedings of International Conference on Computer Aided Design*, 2007, pp. 354–361.

[61] S. Jang, B. Chan, K. Chung, and A. Mishchenko, "WireMap: FPGA Technology Mapping for Improved Routability," in *Proceedings of the 16th International ACM/SIGDA Symposium on Field Programmable Gate Arrays*, 2008, pp. 47–55.

[62] A. Mishchenko, S. Chatterjee, and R. K. Brayton, "Improvements to Technology Mapping for LUT-based FPGAs," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, vol. 26, no. 2, pp. 240–253, 2007.

[63] T.-H. Lin and C.-Y. R. Huang, "Using Sat Based Craig Interpolation to Enlarge Clock Gating Functions," in *Proceedings of ACM Design Automation Conference*, 2011, pp. 621–626.

[64] A. P. Hurst, "Automatic Synthesis of Clock Gating Logic with Controlled Netlist Perturbation," in *Proceedings of the 45th ACM Design Automation Conference*, 2008, pp. 654–657.

[65] N. Banerjee, K. Roy, H. Mahmoodi, and S. Bhunia, "Low Power Synthesis of Dynamic Logic Circuits using Fine-grained Clock Gating," in *Proceedings of Design, Automation and Test in Europe*, 2006, pp. 862–867.

[66] F. S. Marques, L. Rosa Jr, R. P. Ribas, S. S. Sapatnekar, and A. I. Reis, "DAG based Library-free Technology Mapping," in *Proceedings of 17th ACM Great Lakes symposium on VLSI*, 2007, pp. 293–298.

[67] O. Martinello Jr, F. S. Marques, R. P. Ribas, and A. I. Reis, "KL-cuts: A New Approach for Logic Synthesis Targeting Multiple Output Blocks," in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2010, pp. 777–782.

[68] L. Amarú, P.-E. Gaillardon, and G. De Micheli, "MIXSyn: An Efficient Logic Synthesis Methodology for Mixed XOR-AND/OR Dominated Circuits," in *Proceedings of Asia and South Pacific Design Automation Conference*, 2013, pp. 133–138.

[69] Z. Min and S. S. Sapatnekar, "Technology Mapping Algorithms for Domino Logic," *ACM Transactions on Design Automation of Electronic Systems*, vol. 7, no. 2, pp. 306–335, 2002.

[70] M. Pullerits and A. Kabbani, "Library-free Synthesis for Area-Delay Minimization," in *International Conference on Microelectronics*, 2008, pp. 187–191.

[71] W. Shen, Y. Cai, X. Hong, and J. Hu, "An Effective Gated Clock Tree Design based on Activity and Register Aware Placement," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 18, no. 12, pp. 1639–1648, 2010.

[72] J. Lu, W.-K. Chow, and C.-W. Sham, "Fast-Power and Slew-Aware Gated Clock Tree Synthesis," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 20, no. 11, pp. 2094–2103, 2012.

[73] B. Wurth, U. Schlichtmann, K. Eckl, and K. J. Antreich, "Functional Multiple-output Decomposition with Application to Technology Mapping for Lookup Table-based FPGAs," *ACM Transactions on Design Automation of Electronic Systems*, vol. 4, no. 3, pp. 313–350, 1999.

[74] P. Kalla, Z. Zeng, M. J. Ciesielski, and C. Huang, "A BDD-based Satisfiability Infrastructure Using the Unate Recursive Paradigm," in *Proceedings of Design, Automation and Test in Europe*, 2000, pp. 232–236.

[75] J. Cartadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, E. Pastor, and A. Yakovlev, "Decomposition and Technology Mapping of Speed-Independent Circuits using Boolean Relations," *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, vol. 18, no. 9, pp. 1221–1236, 1999.

[76] A. Mishchenko and T. Sasao, "Large-scale SOP Minimization Using Decomposition and Functional Properties," in *Proceedings of the 40th Annual Design Automation Conference*, 2003, pp. 149–154.

[77] A. K. Palaniswamy and S. Tragoudas, "Improved Threshold Logic Synthesis Using Implicant-Implicit Algorithms," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 10, no. 3, p. 21, 2014.

[78] T. Luba, J. Kalinowski, and K. Jasiński, "PLATO: A CAD Tool for Logic Synthesis Based on Decomposition," in *Proceedings of the Conference on European Design Automation*, 1991, pp. 65–69.

[79] A. Mishchenko, B. Steinbach, and M. Perkowski, "An Algorithm for Bi-decomposition of Logic Functions," in *Proceedings of the 38th Annual Design Automation Conference*, 2001, pp. 103–108.

[80] L. Amarú, P.-E. Gaillardon, and G. De Micheli, "BDS-MAJ: A BDD-based Logic Synthesis Tool Exploiting Majority Logic Decomposition," in *Proceedings of the 50th Annual Design Automation Conference*, 2013, p. 47.

[81] G. Chen and J. Cong, "Simultaneous Logic Decomposition with Technology Mapping in FPGA Designs," in *Proceedings of International Symposium on Field programmable Gate Arrays*, 2001, pp. 48–55.

[82] M. Zhao and S. S. Sapatnekar, "Timing-driven Partitioning for Two-phase Domino and Mixed Static/Domino Implementations," in *Proceedings of the IEEE/ACM International Conference on Computer Aided Design*, 1999, pp. 107–110.

[83] J. L. Subirats, J. M. Jerez, and L. Franco, "A New Decomposition Algorithm for Threshold Synthesis and Generalization of Boolean Functions," *IEEE Transactions on Circuits and systems I*, vol. 55, no. 10, pp. 3188–3196, 2008.

[84] L. Franco, J. L. Subirats, and J. M. Jerez, "MaxSet: An Algorithm for Finding a Good Approximation for the Largest Linearly Separable Set," in *Proceeding of Artificial Neural Networks*, 2007, pp. 648–656.

[85] L. Franco, J. L. Subirats, M. Anthony, and J. M. Jerez, "A New Constructive Approach for Creating all Linearly Separable (threshold) Functions," in *International Joint Conference on Neural Networks*. IEEE, 2006, pp. 4791–4796.

[86] M. Zhao and S. S. Sapatnekar, "Technology Mapping for Domino Logic," in *Proceedings of IEEE/ACM International Conference on Computer Aided Design*, 1998, pp. 248–251.

[87] S. Bobba, M. De Marchi, Y. Leblebici, and G. De Micheli, "Physical Synthesis onto a Sea-of-Tiles with Double-gate Silicon Nano Wire Transistors," in *Proceedings of ACM Design Automation Conference*, 2012, pp. 42–47.

[88] T. S. Czajkowski and S. D. Brown, "Functionally Linear Decomposition and Synthesis of Logic Circuits for FPGAs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 12, pp. 2236–2249, 2008.

[89] J. Ciric and C. Sechen, "Efficient Canonical Form for Boolean Matching of Complex Functions in Large Libraries," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 5, pp. 535–544, 2003.

[90] M. Lefebvre and C. Liem, "Cell Generator-Based Technology Mapping by Constructive Tree-Matching and Dynamic Covering," *VLSI Design*, vol. 3, no. 1, pp. 1–12, 1995.

[91] W.-c. Chou, P. A. Beerel, R. Ginosar, R. Kol, C. J. Myers, S. Rotem, K. Stevens, and K. Y. Yun, "Average-case Optimized Technology Mapping of One-hot Domino Circuits," in *Proceedings of International Symposium on Advanced Research in Asynchronous Circuits and Systems*, 1998, pp. 80–91.

[92] J. Xue, D. Al-Khalili, and C. N. Rozon, "Tree-based Transistor Topology Extraction Algorithm for Library-free Logic Synthesis," in *Proceedings of IEEE International Conference on Semiconductor Electronics*, 2004, pp. 5–10.

[93] H.-Y. Song and R. I. Bahar, "Power, Delay, and Area Constrained Synthesis for Mixed Domino/Static Logic Optimization," Citeseer, Tech. Rep., 2000.

[94] T. B. Diplomacao and A. I. Reis, "Domino Logic Library Design and Library Synthesis," Citeseer, Tech. Rep., 2011.

[95] S. K. Karandikar and S. S. Sapatnekar, "Technology Mapping for SOI Domino Logic Incorporating Solutions for the Parasitic Bipolar Effect," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 11, no. 6, pp. 1094–1105, 2003.

[96] A. Cao, R. Lu, C. Li, and C.-K. Koh, "Postlayout Optimization for Synthesis of Domino Circuits," *ACM Transactions on Design Automation of Electronic Systems*, vol. 11, no. 4, pp. 797–821, 2006.

[97] S. Farah and M. Bayoumi, "A Comprehensive Operand-aware Dynamic Clock Gating Scheme for Low-power Domino Logic," in *Proceedings of IEEE 26th International SOC Conference*, 2013, pp. 349–354.

[98] L. Raja, K. Thanushkodi, and T. Hemalatha, "Comparitive Analysis of Various Low Power Clock Gating Design for ALU," in *Proceedings of IEEE International Conference on Electronics and Communication Systems*, 2014, pp. 1–5.

[99] S. Wimer and A. Albahari, "A Look-Ahead Clock Gating Based on Auto-Gated Flip-Flops," *IEEE Transactions on Circuits and Systems I*, vol. 61, no. 5, pp. 1465–1472, 2014.

[100] S. Wimer and I. Koren, "The Optimal Fan-out of Clock Network for Power Minimization by Adaptive Gating," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 20, no. 10, pp. 1772–1780, 2012.

[101] L. Li and K. Choi, "Activity-driven Optimised Bus-specific-clock-gating for Ultra-low-power Smart Space Applications," *IET communications*, vol. 5, no. 17, pp. 2501–2508, 2011.

[102] A. G. Strollo, E. Napoli, and D. De Caro, "New Clock-gating Techniques for Low-power Flip-Flops," in *Proceedings of International Symposium on Low Power Electronics and Design*, 2000, pp. 114–119.

[103] R. Fraer, G. Kamhi, and M. K. Mhameed, "A New Paradigm for Synthesis and Propagation of Clock Gating Conditions," in *Proceedings of ACM/IEEE Design Automation Conference*, 2008, pp. 658–663.

[104] R. Bhutada and Y. Manoli, "Complex Clock Gating with Integrated Clock Gating Logic Cell," in *Proceedings of International Conference on Design and Technology of Integrated Systems in Nanoscale Era*, 2007, pp. 164–169.

[105] X. Man and S. Kimura, "Comparison of Optimized Multi-Stage Clock gating with Structural Gating Approach," in *Proceedings of IEEE Region 10 Conference TENCON*, 2011, pp. 651–656.

[106] P. Babighian, L. Benini, and E. Macii, "A Scalable Algorithm for RTL Insertion of Gated Clocks based on ODCs Computation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 1, pp. 29–42, 2005.

[107] H. Li, S. Bhunia, Y. Chen, K. Roy, and T. Vijaykumar, "DCG: Deterministic Clock-gating for Low-power Microprocessor Design," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 12, no. 3, pp. 245–254, 2004.

[108] S. Huda, M. Mallick, and J. H. Anderson, "Clock Gating Architectures for FPGA Power Reduction," in *Proceedings of International Conference on Field Programmable Logic and Applications*, 2009, pp. 112–118.

[109] J. Kathuria, M. Ayoubkhan, and A. Noor, "A Review of Clock Gating Techniques," *MIT International Journal of Electronics and Communication Engineering*, vol. 1, no. 2, pp. 106–114, 2011.

[110] V. V. Vazirani, *Approximation Algorithms*. Springer, 2001.

[111] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[112] C. A. Coello, "An Updated Survey of GA-based Multiobjective Optimization Techniques," *ACM Computing Surveys*, vol. 32, no. 2, pp. 109–143, 2000.

[113] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Sequential Circuit Design using Synthesis and Optimization," in *Proceedings of Computer Design: VLSI in Computers and Processors*, 1992, pp. 328–333.

[114] K. E. Parsopoulos and M. N. Vrahatis, "Particle Swarm Optimization Method in Multiobjective Problems," in *Proceedings of the ACM Symposium on Applied Computing*, 2002, pp. 603–607.

[115] C. A. C. Coello, G. T. Pulido, and M. S. Lechuga, "Handling Multiple Objectives with Particle Swarm Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 256–279, 2004.

[116] K. B. Lee and J. H. Kim, "Multiobjective Particle Swarm Optimization with Preference Based Sort and its Application to Path following Footstep Optimization for Humanoid Robots," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 6, pp. 755–766, 2013.

[117] B. Y. Qu, P. N. Suganthan, and S. Das, "A Distance Based Locally Informed Particle Swarm Model for Multimodal Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 3, pp. 387–402, 2013.

[118] Y. Zheng, H. Ling, J. Xue, and S. Chen, "Population Classification in Fire Evacuation: A Multiobjective Particle Swarm Optimization Approach," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 1, pp. 70–81, 2014.

[119] M. Hu, T. Wu, and J. D. Weir, "An Adaptive Particle Swarm Optimization with Multiple Adaptive Methods," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 5, pp. 705–720, 2013.

[120] S. Demeyer, T. Michoel, J. Fostier, P. Audenaert, M. Pickavet, and P. Demeester, "The Index-based Subgraph Matching Algorithm (ISMA): Fast Subgraph Enumeration in Large Networks using Optimized Search Trees," *PloS one*, vol. 8, no. 4, pp. 161–183, 2013.

[121] S. Bandyopadhyay, S. Saha, U. Maulik, and K. Deb, "A Simulated Annealing-based Multiobjective Optimization Algorithm: AMOSA," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 3, pp. 269–283, 2008.

[122] S. Bandyopadhyay, "Multiobjective Simulated Annealing for Fuzzy Clustering with Stability and Validity," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 41, no. 5, pp. 682–691, 2011.

[123] J. Chen, W. Zhu, and M. Ali, "A Hybrid Simulated Annealing Algorithm for Nonslicing VLSI Floorplanning," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 41, no. 4, pp. 544–553, 2011.

[124] R. Precup, R.-C. David, E. M. Petriu, S. Preitl, and M. Radac, "Fuzzy Control Systems with Reduced Parametric Sensitivity Based on Simulated Annealing," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 8, pp. 3049–3061, 2012.

[125] F. J. Rodriguez, C. Garcia-Martinez, and M. Lozano, "Hybrid Metaheuristics Based on Evolutionary Algorithms and Simulated Annealing: Taxonomy, Comparison, and Synergy Test," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 6, pp. 787–800, 2012.

[126] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data.* Prentice-Hall, Inc., 1988.

Publications out of this Thesis

Published

- K Sai Praveen and Debasis Samanta, "CRDOM : Cell Re-ordering Based Domino On-the-fly Mapping", *International Journal of VLSI Communication Systems*, vol. 5, no. 4, 2014.
- K. Sai Praveen, and D. Samanta, "Pattern Recognition Based Clock Gating for Domino Logic Circuits", *International Journal of Computers and Applications*, vol. 34, no. 8, 2014.
- K Sai Praveen and Debasis Samanta, "On-the-fly Mapping for Synthesizing Dynamic Domino Circuits", 28^{th} International Conference on VLSI Design, 2015.
- K. Sai Praveen and D. Samanta, Power Aware Domino Circuit Synthesis Using Optimal Clock Gating, Advances in Electronics, Hindawi (Accepted).

Communicated

- K. Sai Praveen, and D. Samanta, "Optimum Unate Decomposition Method for Synthesis of Mixed CMOS VLSI Circuits", Circuits Devices and Systems, IET (under review).