OPTIMAL UNATE DECOMPOSITION METHOD FOR SYNTHESIS OF MIXED CMOS VLSI CIRCUITS

Sai Praveen Kadiyala and Debasis Samanta

School of Information Technology, Indian Institute of Technology, Kharagpur

ABSTRACT

Static CMOS logic style is often the choice of designers for synthesizing low power circuits. This style is robust in terms of noise integrity however, it offers less speed. Domino logic style, as an alternative is often found in critical paths of various large scale high performance circuits. Yet, due to high switching activity they are not suitable for synthesis of low power circuits. To achieve both power and speed benefits, we propose a method of designing circuit using mixed CMOS logic style, taking advantages of both static and Domino logic styles. For a given circuit, we extract the unate and binate components using a unate decomposition algorithm. These are optimized such that the resulting circuit is optimum in terms of power, area and delay. To do this, a multi-objective genetic algorithm is employed. The optimized unate and binate blocks are mapped using Domino and static cell libraries, respectively. Testing the efficacy of our approach with ISCAS85 and MCNC89 benchmark circuits showed an improvement of 25% in delay and 22% in transistor count with 12% more power dissipation compared to circuits with only static CMOS logic. Thus, mixed CMOS circuits are promising in high speed and area constraint applications.

KEYWORDS

Domino Logic, High Performance Architecture, Boolean Decomposition, Unate function, Low Power Circuit

1. INTRODUCTION

Recent trends show that there is a steep rise in the usage of battery operated handheld gadgets [1], [2], [3], [4]. This is posing increasing demands for devices operating at low power and high speed [5], [6]. With custom made chips coming into focus, the designers are pushing more and more functionalities on a single chip [7], [8], [9]. In fact, designers are now pushing billions of transistors in a single chip [10]. This increase the density of the chip and further give rise to problems like thermal variations, process variations, packaging, cooling issues etc. [11], [12], [13], [14]. This necessitates the synthesis of circuits with low power dissipation, without compromise in speed.

Static CMOS logic style is often the choice of designers for designing low power circuits. This is because it is simple to fabricate, has good input/output decoupling and with lower switching activity. Circuits with this logic style are robust in nature and have good noise margins. Pass transistor logic (PTL), another style of static logic family, also finds good application in small scale designs. This logic is known for its low area overhead and ease for implementation. A number of attempts have been made to synthesize circuits using PTL [15], [16], [17], [18], [19]. Though static CMOS is used in low power circuits, it has inherent drawbacks. This style requires double the device count compared to other logic styles. The presence of bulky PMOS transistors in the charging path makes this logic style slow. Although, PTL offers less area and reduced noise margins, it has voltage degradation problem due to threshold voltage offset. As a consequence

DOI : 10.5121/vlsic.2016.7201

this logic is unable to provide robust circuits [18]. As an alternative to static CMOS logic, the dynamic CMOS logic is known for circuit synthesis [20], [21], [22]. It is popular in high performance devices, especially in realizing critical paths for high speed microprocessors [23]. This style requires only half the number of transistors compared to static CMOS logic style. This logic works on the basis of precharge and evaluation phases [24], [25] under a control of clock. Domino logic and NORA logic are two main logic styles that belong to this family. Dynamic CMOS logic, however, has some inherent limitations like charge sharing, clock skew etc. [26], [27]. Cascading this logic for realizing daily life circuits leads to errors in the output, because of intermediate output degradation. Better performance in terms of area and speed of dynamic logic are often checked by the factors like reduction in robustness, increase in switching activity and hence active power dissipation [23]. Moreover, Domino logic realizes only noninverting logic. Hence, the logic functions that are to be realized must be unate in nature. Otherwise, for binate function, it needs the duplication for both type of output.

Of late, to exploit advantages of more than one logic styles, 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. The two logic styles can be combined so that the circuits with mixed logic are advantageous with respect to both power and speed. However, there are some issues to be addressed in order to realize such a mixed circuit. Domino logic is inherently monotone, and can realize only unate functions. Hence it is needed to decompose the given circuit into unate and binate components. The unate component can be realized using Domino logic and binate component can be realized using static CMOS logic. Another problem that can arise is the synchronization between various components. Since the outputs of Domino logic go to the inputs of static logic and vice versa, care should be taken that there should not be any racing at the interface. This requires a careful timing analysis which includes redefining of set-up and hold times, designing of latches etc.

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 the previous methods mainly focused on decomposing the circuit but nowhere emphasis was given on realization using mixed static-domino. Also, a comparative study of various techniques is very much needed. In order to address the above issues, in this paper, we present mixing of both static and Domino logic style. Our principle is to judiciously mix static and domino logic styles to gain in terms of power and speed simultaneously.

The rest of the paper is organized as follows. Some basic concepts related to the current topic are presented in Section 2. Section 3 describes our proposed methodology in designing the mixed CMOS circuits. The experimental results and comparison with other existing techniques are given in Section 4. Finally Section 5 concludes the paper.

2. BASIC CONCEPTS

Proposed methodology for designing of mixed CMOS circuits involves realizing unate functions. In this section, we present few basic terminologies which we refer to in our discussion.

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 $2n$ states.

Weight of a state: Weight of the state is the number of '1's the state has in its binary representation.

Unate function: A Boolean function is said to be unate if each of its input variables exist in either true or compliment form but not both. A Boolean function which is not unate is said to be binate.

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

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 of Hamming distance 1.

Type of influence: A state pair having Hamming distance 1 represents the transition of a particular variable from $0 \rightarrow 1$ or $1 \rightarrow 0$. It can result in change of the output. Depending on the transition in the output $0 \rightarrow 1$, $1 \rightarrow 0$ or no transition, the influence of the state pair is decided as positive, negative and neutral. 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.

Variable of influence: 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.

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

3. PROPOSED METHODOLOGY

In this section, we present our methodology for synthesizing mixed CMOS circuits. An overview of our methodology is shown in Fig. 1.

Our overall approach consists of the following operations.

Suppose, given a circuit C_{init} whose Boolean function is represented by f(X). X is the input vector, where $X = (x_1, x_2, x_3, \ldots, x_{n-1}, x_n)$.

Initial Unate Decomposition (IUD): A unate decomposition method, denoted as UD, which decomposes C_{init} into two different logical set $U(X)$ and $B(X)$. That is, $UD(f(X)) \rightarrow \{U(X), B(X)\}$ such that $U(X) \cup B(X) = f(X)$ and $U(X) \cap B(X) = \phi$, where $\cap$, $\cup$ are the disjunction and conjunction operators of set theory respectively.

Optimizied Unate Decomposition (OUD): An optimization of a given unate-binate decomposition of Boolean function is performed, where Opt is the optimization operator and U_{opt}, B_{opt} are the logical set that result after optimization.

In other words, $Opt(U(X),B(X)) \rightarrow \{U_{opt}(X),B_{opt}(X)\}$ is a decomposition optimized with respect to some objectives.

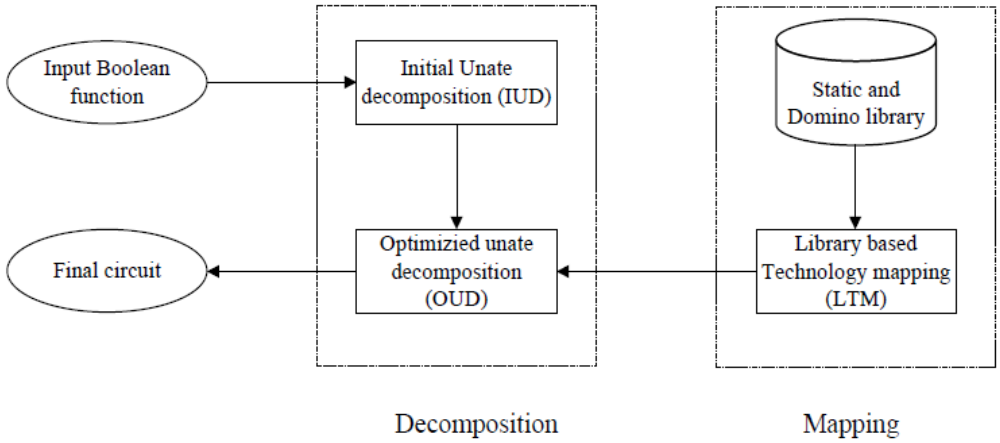


Figure1. Overview of our methodology

Library based Technology Mapping (LTM): Suppose, L_{stat}, L_{dyn} are the static and dynamic libraries used for mapping U_{opt}, B_{opt} set, respectively. *Map* is the operator for mapping and if C_{final} is the resulting final circuit, then

$$Map(U_{opt}, B_{opt}, L_{stat}, L_{dyn}) \rightarrow C_{final}$$

Next, we describe the above mentioned operations in detail with illustration.

2.1. Initial Unate Decomposition (IUD)

We propose an algorithm which takes a completely defined Boolean function as input and decomposes it into unate and binate parts. A flowchart of our algorithm is shown in Fig. 2. The notations used in the flowchart are mentioned in the following.

We explain various steps in our algorithm with the help of an example. To start with, we consider the following Boolean function as an input.

$$f(x)= \overline{x_3}\,\overline{x_1} + \overline{x_2}\,\overline{x_1}\,\overline{x_0} + x_2\,\overline{x_1}\,x_0 + x_3x_2x_1\,\overline{x_0} \quad (1)$$

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

Step 1: In this step, we extract the onset (OS) states of the Boolean function. These are {0, 1, 4, 5, 8, 11, 14}. Realizing the OS elements is equivalent to realizing the original Boolean function.

Step 2: Here, we construct a POSET for the extracted OS (see Fig. 3). The considered example has states with 5 different weights 0, 1, 2, 3, 4. The elements which belong to the onset are represented using grey ovals and the remaining are represented with white ovals. The decimal value of the elements is shown adjacent to the ovals.

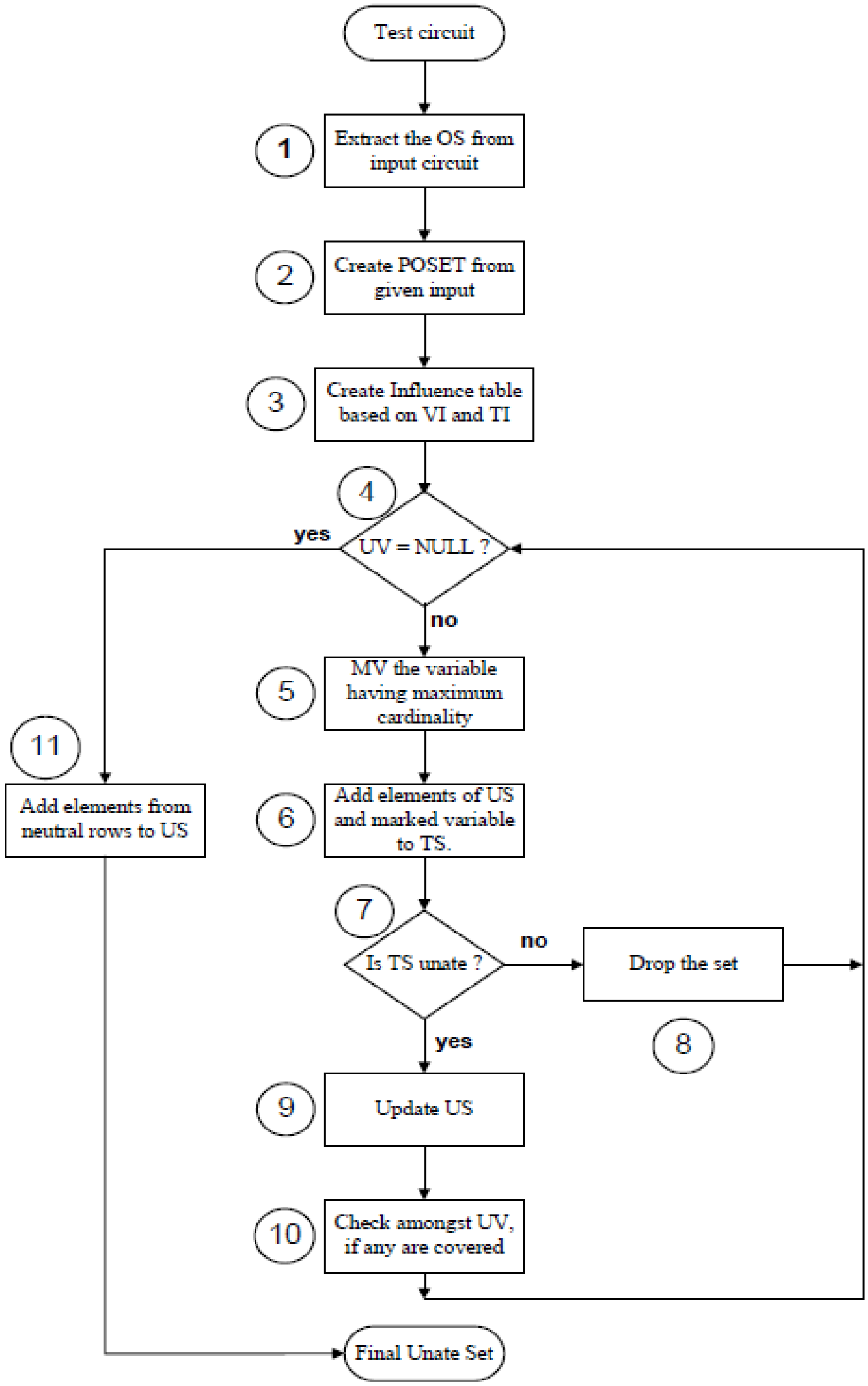


Figure2. Flowchart representation of proposed algorithm

Table 1. Influence Table for $f = \sum (0, 1, 4, 5, 8, 11, 14)$

Type of Influence	X_0	X_1	X_2	X_3
+	10-11 8-9	9-11 12-14	10-14	3-11 6-14
-	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 1-5 9-13 2-6	0-8 2-10

Step 3: We categorize each state pair of the POSET based on its VI and TI into an Influence table (for example, Influence table for the considered $f(x)$ is shown in Table.1). If we consider the pair 1−9, the variable which is changing is $x3$ 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 the possible pairs, 32 in this case, are categorized.

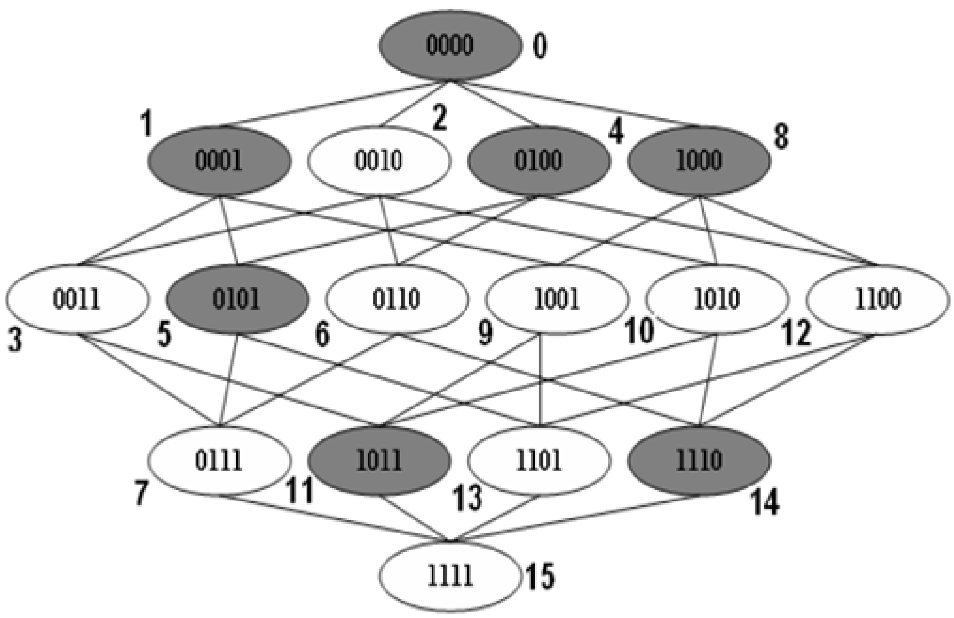


Figure3. Partially ordered set for the considered example

Step 4: This step checks for any existing unmarked variables. Initially, all the variables of the influence table are unmarked. At the beginning of each iteration a particular variable is marked.

Step 5-6: We define cardinality of a set as 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 this case the maximum cardinality set has VI as x_1, TI as − and cardinality of 5. The elements are 0, 1, 2, 3, 4, 5, 6, 7, 8, 10 and these form the initial US. MV 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 our current US spans any other set present in the 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. Its VI is marked and its 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. 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 our running example, state 15 is added from the neutral influence, giving the final unate set.

With reference to our example, we get 14 states in the maximum unateset {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 the elements of OS are categorized into either unate or binate set. Both the sets together correspond to the realization of Boolean function $f(X)$. Thus we conclude our Initial Unate Decomposition (IUD) as follows

$$f(X) = U(X) \cup B(X) \tag{2}$$

Where, $U(X) = \sum (0, 1, 4, 5, 8)$, the unate set and $B(X) = \sum (11, 14)$, the binate set. After performing the IUD for the given circuit, the realization will be as shown in Fig. 4. Next, we perform optimization of the obtained sets for overall better performance of the circuit, which is discussed in the following sub section.

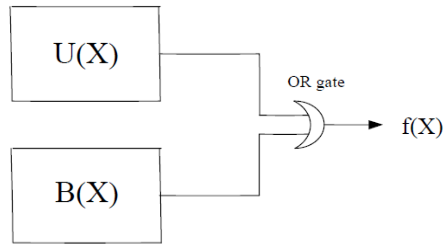


Figure4. Mixed CMOS design after IUD

3.2. Optimization of Unate Decomposition (IUD)

In this section, first we state the need for optimizing IUD. Next, we state the problem of optimizing an IUD, clearly defining the objectives, constraints and design parameters in it. Finally, we suggest a multi objective Genetic Algorithm based approach to solve the COPT 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 which are 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. We formally define the COPT problem in the following. We refer the below mentioned notations in our definition.

Notations used in COPT problem definition			
Boolean function	$f(x) = \{x_1, x_2, \dots x_n\}$	Static and dynamic libraries	$(L_{stat}; L_{dyn})$
Onset of function	$OS = \{i_1; i_2; \dots i_J\}$	Power, area, delay of static block	$P_{stat};A_{stat};D_{stat}$
Number of elements in OS	J	Power, area, delay of dynamic block	$P_{dyn}; A_{dyn}; D_{dyn}$
Number of Elements in U(X)	I	Target values for power, area and delay	$(P_0;A_0;D_0)$
Number of Elements in B(X)	$J\text{-}I$	Operators for mapping and optimization	$MAP; Opt$

For a given Boolean function $f(X)$, $\{U(X), B(X)\}$ 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

$$IUD(U(X),B(X)) \rightarrow OUD(U_{opt}(X),B_{opt}(X))$$

We consider following three objective functions to 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 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 circuits.

We define a decomposition $U_{opt}(X)$, $B_{opt}(X)$ as the OUD, if it satisfies the following.

Given IUD $\{U(X), B(X)\}$ of a logic $f(X)$:

$$OUD\{U_{opt}(X), B_{opt}(X)\} = \text{minimize } [P = f_p(U(\mathrm{X}), B(\mathrm{X})),\ A = f_a(U(\mathrm{X}), B(\mathrm{X})),\ D = f_d(U(\mathrm{X}), B(\mathrm{X}))\ ]$$

subject to $Uopt(X) \subseteq U(X)$, $B(X) \subseteq Bopt(X)$, $Uopt(X) \cup Bopt(X)=U(X) \cup B(X)=OS$, and $P \leq P0$, $A \leq A0$, $D \leq D0$, for some constraints $P0$, $A0$ and $D0$.

Below, we propose a GA based approach to solve the COPT problem. Various steps involved in that process are described below.

GA-based approach: We follow a non-dominated sorting genetic algorithm (NSGA-II) [28] to solve the COPT problem. A detailed framework of the NSGA-II is shown in Fig. 5. We have followed binary encoding to define a chromosome in GA.

Given a Boolean function of n input variables, we encode the corresponding OS states with an n bits binary representation. We define a representation for all the states in 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 US of $f(X)$.

Structure of chromosome: The chromosome structure looks as shown in Fig. 5, It is a set of representations for the I elements present in the Onset.

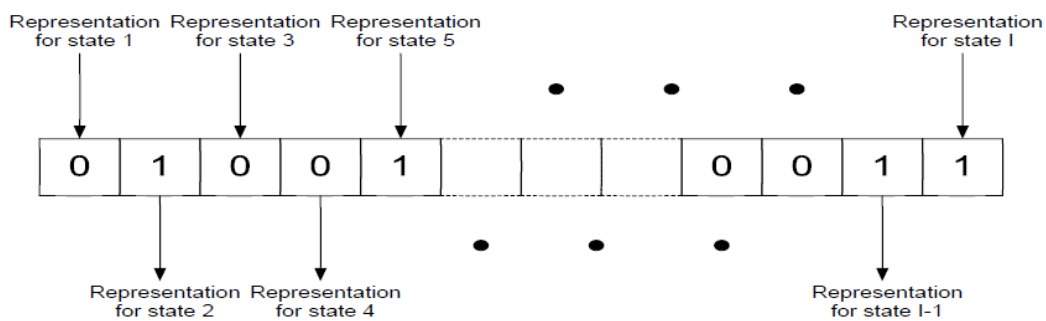


Figure5. Structure of chromosome for unate set having I elements

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. An instance of our genetic algorithm based optimization is shown in 12 Fig. 6. Each of them can be either 0 or 1. Np number of such candidates are chosen for initial parent population (see Fig. 7). On this initial population, we perform a two point crossover technique [28]. Later we mutate the population with a probability pm. By doing so, we generate Nc number of population which is used in selection of candidates for next generation.

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 $f_i^p(P_1)$, $f_i^a(A_1)$, $f_i^d(D_1)$ for a given candidate i as shown in Fig. 6. In the same way, fitness values are computed for all the candidates in the combined parent and child population.

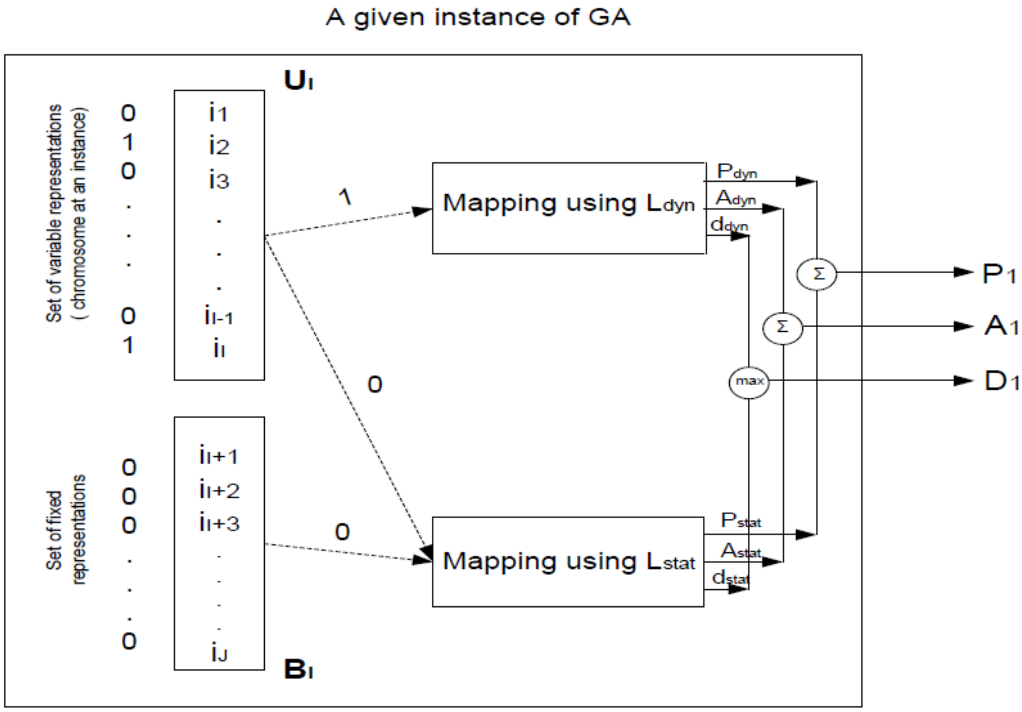


Figure6. Overview of our GA operation

Using the obtained fitness values we perform a non-dominated sorting of entire parent and child population. According to Coello et al. [29], we consider a vector $U = (U_1, U_2, \ldots, U_k)$ dominates a vector $V = (V_1, V_2, \ldots, V_k)$, denoted by $U \prec V$, iff U is partially less than V, i.e. $\forall i \in \{1, 2, \ldots, k\}$, $u_i \leq v_i \wedge \exists\, i \in \{1, 2, \ldots, k\} : u_i < v_i$. 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.

All the, hence obtained candidates are assigned a non domination rank(i_{rank}) as stated in [28]. 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 Euclidian distance between two candidates x and y belonging to the population.

For an individual i the crowding distance is computed as shown below,

$$i_{distance} = 1/2(|d_{ik}| + |d_{ij}|) \quad (3)$$

where j, k are the nearest neighbours to i on either side, along the non domination level.

Like this for all the individuals in the population, their respective crowding distance ($i_{distance}$) are computed. Using these two properties of an individual the crowd comparison is performed. If α is the crowd comparison operator, as stated in [28] we define

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

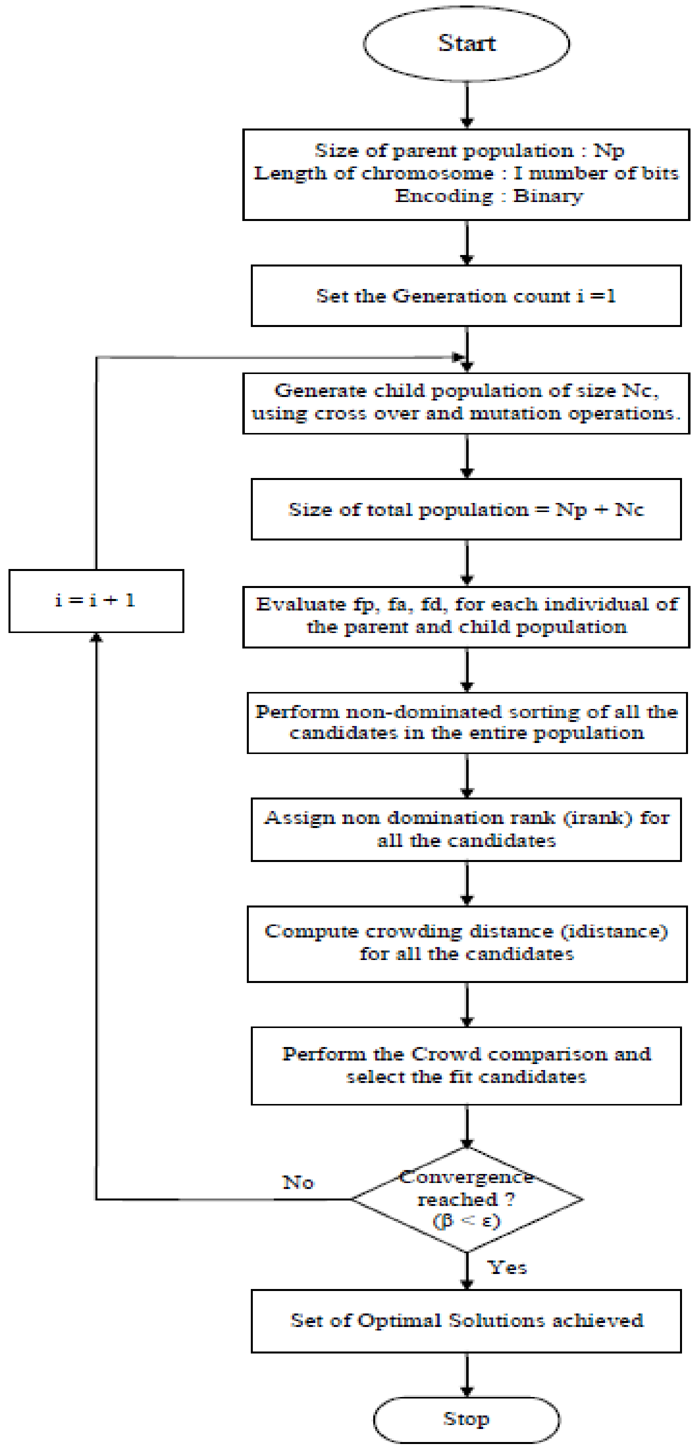


Figure7. NSGA-II framework to solve COPT problem

which 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 i.e. having more $i_{distance}$. As shown in Fig. 5, we used crowd comparison operator [28] to select the solutions towards a uniformly spread-out Pareto-optimal front.

As suggested by Deb et al. in [28] we choose, β which defines the diversity of the obtained Pareto optimal set, as a criteria for terminating our GA. After obtaining the optimal Pareto front we compute Nadir point (*nadir*) for the front, in order to choose the best individual [28]. For our *COPT* problem, the 3 coordinates of Nadir point namely $\{X'_{nadir}, Y'_{nadir}, Z'_{nadir}\}$in the state space are expressed as follows

$$X'_{nadir} = max(fp(X)),\ Y'_{nadir} = max(fd(X)), Z'_{nadir} = max(fa(X)) \quad (5)$$

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 the candidates of 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 sets are used in realizing our optimized mixed CMOS circuit.

4. EXPERIMENTS AND EXPERIMENTAL RESULTS

In this section, we present details on various experiments conducted to substantiate the efficacy of our proposed approach. 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 obtained results with those of the existing techniques.

4.1. 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. The parameters power, area and delay are chosen as metrics for evaluating the performances of various approaches. To perform mapping of the .pla 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. 2. The t_{LH} and t_{HL} give the respective rise and fall delays obtained from simulations performed using $0.065\mu m$ CMOS process, 1.1V, 27^oC. Process variation effects become significant in sub45nm range. Hence they are not addressed in our work. Though some standard functions may appear in both the libraries their respective parameter values differ.

Berkeley SIS tool, Version 1.3 [30] is used for mapping and various pre-processing 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 the circuit. Since we are performing library based mapping, we used the print delay-m library command which includes a library based delay model. The dynamic and leakage power for the components are 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. For the particular supply voltage, in a similar fashion leakage power is also estimated by the tool. 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.

Table 2. List of some of the cells present in the library

Cell	Static Cell Library			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

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 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, consistent with the literature on NSGA II [28]. The diversity operator β as mentioned in [28] is observed over 5 successive generations. The constant ϵ which measures the change in β over successive generations, is chosen to be 0.001.

4.2. 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 flavour 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.

Table 3. Considered benchmarks from ISCAS85 and MCNC89

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

4.4. Experimental Results

The performance of our IUD algorithm with some MCNC89 benchmark circuits is shown in Table. 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.

Table 4. Performance of IUD

Name of the Circuit	I/O	OS (state count)	U (state count)	B (state count)	CPU time (seconds)
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
t481	16/1	7632	5834	1798	85
table3	14/14	3933	2632	1301	380
tcnh0	17/16	5312	3691	1621	1634

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 analyse 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 pre-processing technique which uses SIS tool. In case of multi-output circuits, we have considered each output separately.

Next, we consider two standard ISCAS benchmark circuits C880.pla and C1908.pla. The power, area and delay are normalized with respect to the static CMOS realization. Obtained normalized values are shown below in Fig. 8. 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. 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 affect the critical path and hence reduce the delay. Power and area of the realizing all unate states in domino style is 50% and 60% more than 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 for C880.pla, is achieved when 60% of unate nodes are realized using Domino logic (shown with a marker in Fig. 8.a). In a similar fashion, analysis is done for the circuit C1908.pla, shown in Fig. 8.b. The optimum, in this case is achieved when 70% of unate nodes are realized using Domino logic.

Experimental results on ISCAS85 and MCNC89 benchmark suites using mixed CMOS (IUD) and optimized mixed CMOS (OUD) are shown in Table. 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 the table 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. Below we mention the performance of various other existing approaches.

Table 5. Performance of our decomposition algorithm

Name of the Circuit	I/O	Mixed CMOS (IUD)			Optimized Mixed CMOS (OUD)		
		Power(uW)	Area(tr. count)	delay(ns)	Power(uW)	Area(tr. count)	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	60/26	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

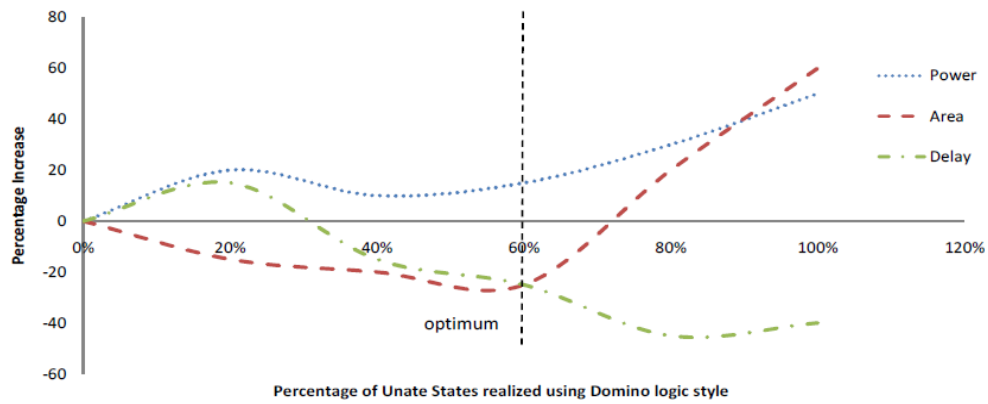
Table 6. Comparative study of various approaches

Name of the Circuit	I/O	Static approach (SIS)			Two level approach			Prasad's approach			Jacob's approach		
		P(uW)	A(tr.count)	D(ns)	P(uW)	A(tr.count)	D(ns)	P(uW)	A(tr.count)	D(ns)	P(uW)	A(tr.count)	D(ns)
b1	3/4	17.7	44	58	31.4	77	37	20.8	55	45	21.4	49	66
ex5	8/63	215.1	4234	355.8	3743	7692	192.4	235.4	4123	303.4	246.4	4126	376.4
9sym	9/1	501	940	423.3	898	1732	280.4	482.3	863	394.6	632.3	1053	413.2
x3	135/99	2783	4217	250.7	3982.1	6841	140.3	2943	4162	242.4	2945.3	2931	243.2
C880	60/26	2017.7	766	21.4	2823.1	1225	12.84	2439.4	920	18.3	2312.4	854	24.1
C1908	33/25	2242.7	1071	41	3251.5	1714	24.6	2851.4	1189	35.7	2489.4	1234	38.3
C2670	233/140	3680.7	1325	31	4984.1	2120	18.6	4390.7	1562	26.4	3895.4	1543	37.4
C5315	178/123	9015.6	3004	39.8	13720.3	4806	23.9	10547.1	2983	29.3	10213.7	2885	44.6

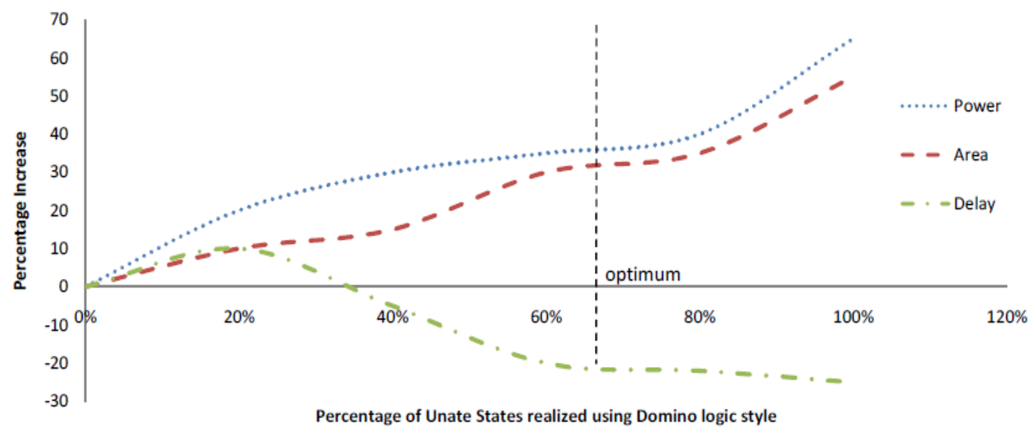
Comparison of various existing methods is mentioned in Table. 6. Static CMOS realization [30], Two-level decomposition [10], Prasad's [31] and Jacob's [32] approaches are compared. From Table. 6, it can be seen that the Two-level based approach gives the minimum delay amongst the existing techniques. 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 [10]. This approach is 22% faster than optimized mixed CMOS, 10% faster than simple mixed CMOS (from Table. 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 (mentioned in Table. 6), 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. On the other hand, our optimized mixed CMOS approach outperforms this technique both in terms of power and delay.

Results of synthesis using Prasad's approach [31] are mentioned in columns 9 to 12 of Table. 6. Our approach has clearly outperformed the Prasad's approach both in terms of area and delay. Our approach has showed 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 *C*5315.*pla*, is shown in Fig. 9.



(a) C880.pla



(b) C1908.pla

Figure8. Percentage increase of power, area and delay using IUD realization over static CMOS realization

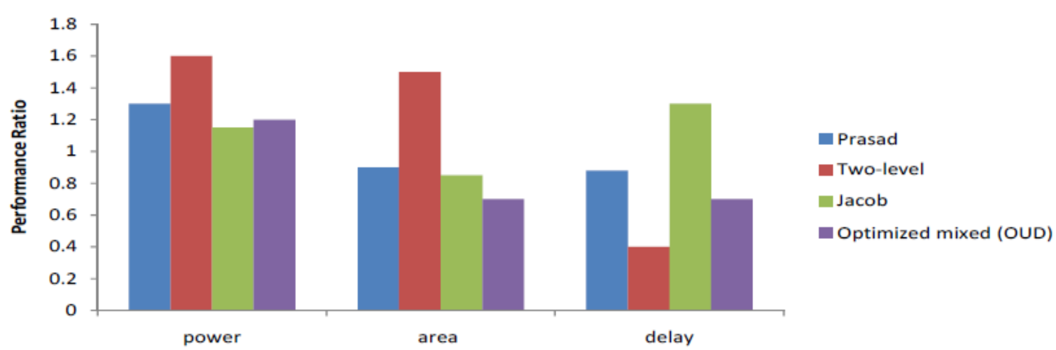


Figure9. Performance ratio of various approaches with respect to static CMOS realization for C5315.pla

5. CONCLUSION

An approach to realize mixed static Domino circuit is proposed in this work. 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 to obtain better power, area and delay. Given a circuit, first we perform an initial unate decomposition (IUD) and later we optimize it for obtaining power and speed efficiency. Also, the proposed approach yields lower transistor count compared to static CMOS logic style. Mixed CMOS circuit is 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. In this work, we only considered circuit decomposition. However, designing of clock which is a highly active signal in the circuit, is yet to be addressed. Next research aims to address the problem of reducing power dissipation caused by clock signal.

REFERENCES

[1] Massimo, A. (2012): "Ultra-Low Power VLSI Circuit Design Demystified and Explained: A Tutorial", IEEE Trans. on Circuits and Systems, Vol. 59, No. 1, pp. 3-29.

[2] Ramakrishnan, V., Sujan, K.M., Poras, T.B. (2013): "NEM Relay-Based Sequential Logic Circuits for Low-Power Design", IEEE Trans. on Nanotechnology, Vol. 12, No. 3, pp. 386-398.

[3] Chen, S.L., Wang, J.G. (2013): "VLSI Implementation of Low-power Cost efficient Lossless ECG Encoder Design for Wireless Healthcare Monitoring Application", Electronics Letters, Vol. 49, No. 2, pp. 91-93

[4] Choi, J.W., Lee, J., Min, B.G., Park, J. (2010): "Energy Efficient Hardware Architecture of LU Triangularization for MIMO Receiver", IEEE Trans. on Circuits and Systems- II, Vol. 57, No. 8, pp. 632-636.

[5] Azcona, C., Calvo, B., Medrano, N., Celma, S. (2013): "CMOS quasi-digital Temperature Sensor for Battery Operated Systems", Electronics Letters, Vol. 49, No. 21, pp. 1338-1340.

[6] Sai Praveen, K., et al. (2015): "Perceptually Guided Inexact DSP Design for Power, Area Efficient Hearing Aid", Proc. of Biomedical Circuits and Systems, pp. 1-4.

[7] Xiang, Z., Ahmed, L. (2010): "A Multilayer Nanophotonic Interconnection Network for On-Chip Many-core Communications'. Proc. of Design Automation Conference, pp. 156-161.

[8] Shin, J.L., Huang, D., Bruce, P., et al. (2011): "A 40 nm 16-Core 128-Thread SPARC SoC Processor", IEEE Journal of Solid-State Circuits, Vol. 46, No. 1, pp. 131-144.

[9] Wendel, D.F., Kalla, R., Warnock, J.(2011): "POWER7, a Highly Parallel, Scalable Multi-Core High End Server Processor", IEEE Journal of Solid-State Circuits, Vol. 46, No. 1, pp. 145-161.

[10] Debasis, S., Nishant, S., Ajit, P. (2002): "Synthesis of High Performance Low Power Dynamic CMOS Circuits". Proc. 15th Int. Conf. on VLSI Design, pp. 99-104.

[11] Mohammad, H.M., Reza, F.M., Akbar, D., et al. (2013): "A Universal Method for Designing Low-Power Carbon Nanotube FETbased Multiple-Valued Logic Circuits", IET Comput. Digit. Tech, Vol. 7, No. 4, pp. 167-181.

[12] Javid, J., Mohab, A. (2008): "Statistical Thermal Profile Considering Process Variations: Analysis and Applications", IEEE Trans. on Computer Aided Integrated Circuits and Systems, Vol. 27, No. 6, pp. 1027-1040.

[13] Jieyi, L., Ja, C.K., Seda, O.M., Yehea, I. (2010): "SACTA: A Self-Adjusting Clock Tree Architecture for Adapting to Thermal-Induced Delay Variation", IEEE Trans. on Very Large Scale Integrated (VLSI) Systems, Vol. 18, No. 9, pp. 1323-1336.

[14] Bing, S., Yufu, Z., Ankur, S. (2013): "Dynamic Thermal Management under Soft Thermal Constraints", IEEE Trans. on Very Large Scale Integrated (VLSI) Systems, Vol. 21, No. 11, pp. 2045-2054.

[15] Yin, T.H., Jin-Fa, L. (2012): "Low Voltage and Low Power Divide-By-2/3 Counter Design Using Pass Transistor Logic Circuit Technique", IEEE Trans. on Very Large Scale Integrated (VLSI) Systems, Vol. 20, No. 9, pp. 1738-1742.

[16] Mariano, A.H., Monico, L.A. (2011): "CMOS Full-Adders for Energy-Efficient Arithmetic Applications", IEEE Trans. on Very Large Scale Integrated (VLSI) Systems, Vol. 19, No. 4, pp. 718-721.

[17] Tsung, T.L., Louis, P.A., Matthew, D.P., Jan, M.R. (2009): "Asynchronous Computing in Sense Amplifier-Based Pass Transistor Logic", IEEE Trans. on Very Large Scale Integrated (VLSI) Systems, Vol. 17, No. 7, pp. 105-115.

[18] Ilham, H., Denis, F., Ian OConnor, Jean-Didier, L. (2010): "ULPFA: A New Efficient Design of a Power-Aware Full Adder", IEEE Trans. on Circuits and Systems, Vol. 57, No. 8, pp. 2066-2074.

[19] Dejan, M., Cheng, C.W., Louis, P.A., Tsung-Te, L., Rabaey, J.M. (2010): "Ultralow-Power Design in Near-Threshold Region", Proceedings of IEEE, Vol. 98, No. 2, pp. 237-252.

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

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

[22] Sai Praveen, K., Debasis, S. (2014): "Clock-Gating Approach to Low Power Domino Circuit Synthesis", International Journal of Computers and Applications, Vol. 36, No. 4, pp. 140-147.

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

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

[25] Sai Praveen, K., Debasis, S. (2015): "On-the-fly Mapping for Synthesizing Dynamic Domino Circuits", Proc. of International Conference on VLSI Design, pp. 458-463.

[26] Yu-Shun, W., Min-Han, H., Chia-Ming, L. (2011): "A 1.2V 6.4GHz 181ps 64-bit CD Domino Adder with DLL Measurement Technique'. Proc. of International Symposium on Circuits and Systems (ISCAS), pp. 1423-1426.

[27] Sai Praveen, K., Debasis, S. (2014): "CRDOM: Cell Reordering Based Domino On-the-fly Mapping ", International Journal of VLSI Design & Communication Systems, Vol. 5, No. 4, pp. 13-18.

[28] Kalyanmoy Deb. (2002): "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II", IEEE Trans Evolutionary Comp, Vol. 6, No. 2, pp. 182-197.

[29] Coello Coello, C.A. (2000): "An Updated Survey of GA Based Multi-objective Optimization Techniques", ACM Computing Surveys, Vol. 32, No. 2, pp. 109-143.

[30] Brayton, R.K. (1992), "SIS: A System for Sequential Circuit Synthesis". University of California Berkely, 1992.

[31] Mukul, R.P., Krikpatrick,D., Brayton,R.,K. (1997): "Sangiovanni-Vincentelli AL. Domino Logic Synthesis and Technology Mapping". Int. Workshop on Logic Synthesis.

[32] James, J., Alan, M.: "Unate Decomposition of Boolean Functions". Proc. Int. Workshop on Logic Synthesis, pp. 66-71.

AUTHORS

Sai Praveen Kadiyala received his B.Tech. degree in Electrical Engineering and PhD in Low Power VLSI from Indian Institute of Technology in 2008, 2015 respectively. Currently, he is working as a research fellow in Nanyang Technological University, Singapore. His research interests include VLSI Design, Low power VLSI, Inexact Architecture, Machine learning for Cyber Security.

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