Synthesis of High Performance Low Power Dynamic CMOS Circuits

Debasis Samanta[†], Nishant Sinha[‡] and Ajit Pal[†]

[†]Department of Computer Science & Engineering
Indian Institute of Technology Kharagpur
West Bengal, India 721302
apal@cse.iitkgp.ernet.in

[‡]Department of Electrical & Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
nishants@ece.cmu.edu

Abstract

This paper presents a novel approach for the synthesis of dynamic CMOS circuits using Domino and Nora styles. As these logic styles can implement only non-inverting logic, conventional logic design approaches cannot be used for Domino/Nora logic synthesis. To overcome this problem, we have used a new concept called unate decomposition of Boolean functions. The unate decomposition expresses a general Boolean function in terms of a minimum number of positive and negative unate functions, which can be readily mapped to a two-level network of Domino/Nora logic circuit. To deal with functions of very large number of variables, a function is first decomposed into sub-functions of not more than 15 variables. Unate decomposition is efficiently performed for each of these sub-functions independently. However, two-level Domino/Nora realization for these functions are quite often not suitable for the realization of practical VLSI circuits having reasonable delay, because of the large number of series/parallel MOS transistors. To overcome this limitation, we have performed multilevel decomposition of each sub-function. The netlist produced by the multilevel decomposition directly maps (on-the-fly) to Domino/Nora cells. In order to analyze the circuits synthesized by our approach, we have estimated the delay and power of the circuits based on the models presented in the paper. Our result is then compared with the static CMOS circuits synthesized by standard SIS tool. Our approach has been found to achieve better results with regard to area, delay and power consumption compared to the existing approaches. It is envisaged that the synthesized Domino/Nora circuits will be suitable for realizing high-performance and low power circuits.

1. Introduction

By combining the advantages of full-complementary CMOS circuits and pseudo-nMOS circuits, dynamic CMOS circuits are realized to provide a number of useful properties, such as lower transistor-count, higher speed, glitch-free operation, and lower switching threshold. These properties make dynamic circuits very attractive for the realization of high-performance systems. However, when several dynamic stages are cascaded, clock-skew problem arises. To overcome the clock-skew problem, Domino [1] or Nora [2] logic styles have been invented. The Domino logic circuits suffer from the limitation that it can implement only non-inverting logic functions. This necessitates the realization of the complements of internal signals through separate cones of logic using complements of the primary inputs resulting in significant area overhead. Therefore, the conventional synthesis paradigms used for static CMOS, which implicitly assume that the complement of any internal node is available using a single inverter cannot directly be applied to the synthesis of Domino (or Nora) CMOS circuits. This has motivated researchers in recent years to develop synthesis process taking into consideration the constraints and flexibilities of Domino logic [3], [4].

Most of the existing works start with a multilevel circuit in terms of standard gates realized using some standard tool, such as SIS. Then the circuit is manipulated to convert it into an unate circuit. This is followed by a technology-mapping step, which transforms the circuit in terms of Domino logic gates. We have adopted a radically different approach. Our starting point is the functional description of a Boolean function given in PLA/BLIF format. As our objective is to deal with digital circuits with large number of input variables, we *partition* the given circuit graph in terms of sub-graphs, each having not more than 15 input variables. This is done, so that the u*nate decomposition*, which is the second step of our algorithm, can be performed 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. For functions of more than 15 variables, it becomes computationally infeasible. 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 Nora network. 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, we perform *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. While doing the decomposition step, cells are created *on-the-fly* using the sub-functions satisfying the constraints, thereby performing the crucial step of *technology mapping*. Instead of using standard cell-library, this on-the-fly cell generation approach helps to realize circuits with smaller number of levels having smaller delay. We have developed suitable models to estimate delay and power of the synthesized circuits.

Efficacy in our approach has been tested with ISCAS and MCNC benchmark circuits having large number of inputs (as many as 233) and outputs (as many as 140). Our result is compared with that of the full complementary static CMOS circuits synthesized by the standard SIS tool. The synthesized Domino and Nora circuits are found to be superior in performance in terms of all the three important parameters; area, delay and power, compared to their static CMOS counterpart. This makes the Domino and Nora logic circuits suitable for high-performance and low-power applications.

Basic approach of various steps of our algorithm, the partitioning, unate decomposition, technology mapping, delay estimation and power estimation techniques are introduced in Sec. 2. Various algorithms based on these approaches are given in Sec. 3. Section 4 presents the implementation details and experimental results. Finally, conclusions and future works are given in Sec. 5.

2. Basic Approach

Our synthesis process is visualized in Fig. 1. Basic approach of various steps of our algorithm is presented in the following subsections.

2.1 Partitioning

As we mentioned in the previous section, we partition a large function into smaller sub-functions having not more than 15 input variables, and then process them independently. For this purpose, we have created a graph corresponding to a given circuit in BLIF format. The partitioning is done in two phases. In the first phase, the entire graph is traversed in a breadth-first fashion and based on the maximum size (number of inputs) of a partition, the circuit graph is partitioned into smaller sub-graphs corresponding to smaller sub-circuits. The first phase partitions the input graph into variable-size partitions. We usually end up with partitions having too few nodes, which can be merged with other partitions to form a partition with reasonably larger number of inputs. The second phase allows us to merge or refine those partitions. We move a node from its current position to the new one, if the number of edges connecting it to the new partition is more than the number of edges connecting to the earlier one. This is similar to the refinement phase in the standard Fiduccia-Mattheyses (FM) bi-partitioning algorithm. We have observed that, by adopting this "divide-and-conquer" approach it is possible to handle VLSI circuits involving hundreds of input variables. The partitioning algorithm is presented in Sec. 3.1.

2.2 Unate Decomposition

To overcome the limitation due to only non-inverting logic, we have proposed a novel logic synthesis technique for the realization of Domino or Nora circuits, based on two-level unate decomposition. Our approach is to decompose any given Boolean function in terms of sub-functions, which are either positive or negative unate. Based on this decomposition, any function f can be expressed as a product of k terms,

$$f = \sum_{i=1}^{k} N_i \cdot P_i$$

where N_i and P_i denote negative and positive unate functions, respectively. This decomposition of f leads directly to a two-level realization having a generic structure shown in Fig. 2(a) and 2(b) for Domino and Nora logic, respectively. The first level of this network consists of a number of blocks equal to the number of partitions (factors) in the decomposition; each block consists of a precharged n-block corresponding to each P_i. The outputs of these blocks are applied to the transistors in the second level, each of which is in series with an n or p sub-block corresponding to each N_i in the decomposition. If a factor consists of a single unate function (rather than the product of two unate functions), either the n-block in the first level or an n or p sub block in the second level will be absent. It has been shown [5] that the unate decomposition partitions a general Boolean function in terms of a minimum number of unate sub-functions. As a consequence, the two-level realization based on this unate decomposition realizes a function with minimum number of gates.

2.3 Multilevel Decomposition

Two-level dynamic CMOS realization for functions of large number of variables using the unate decomposition approach outlined in the previous subsection is not suitable for practical implementation because of a large number of transistors in series and parallel in each gate. To realize circuits with reasonable delay, it in necessary to impose constraint on the *length* and *width* of a practical gate. By *length* we mean the maximum number of

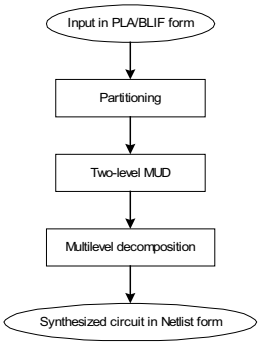


Fig. 1 Synthesis steps

series transistors in the gate. When the number of transistors in series increases (this is the case when a product term consists of a large number of variables), on-resistance of the gate increases and thereby affecting the delay of the gate. Increase in (width) the number of transistors in parallel (this is the case when large number of product terms are there in the function) proportionally increases the drain capacitance of the gate, which in turn increases the delay time as well as the dynamic power consumption. As the delay increases rapidly with the number of transistors in series, this number is restricted to 4 to get reasonable delay. Similarly, the number of transistors in parallel is restricted to 6 in order to check the delay as well as dynamic power consumption due to load capacitance.

To satisfy the *length* and *width* constraints, we propose another decomposition step, where each node in the two-level decomposed circuit is further broken down into sub-structures and intermediate nodes till each node satisfies the *length* and *width* constraints. This decomposition is termed as *multilevel decomposition*. Multilevel decomposition can be achieved very efficiently by computing kernels in a network [6]. Kernel based multilevel decomposition is an algebraic manipulation of logic functions as illustrated in the following example.

Example 1: Consider the function f_1,

$f_1 = abe\bar{g} + abfg + ab\bar{e}g + ace\bar{g} + acfg + ac\bar{e}g + de\bar{g} + dfg + d\bar{e}g$,

which has 33 literals. This can be written in factored form using only 9 literals as given below.

$$f_1 = (a(b + c) + d)(e\bar{g} + g(f + \bar{e}))$$

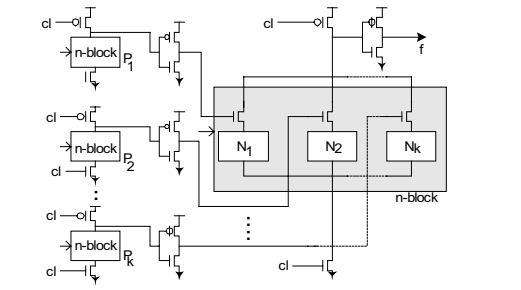
On decomposing the same function into smaller gates, as required during the technology mapping, the transistor count increases, but only slightly. For example, the following decomposition of f has a total literal count of 12 instead of 33.

$$H = a(b+c)+d,\ J = g(f+\bar{e}),\ I = e\bar{g}+J$$
$$f_1 = H \cdot I$$

Factored form helps us to find common sub-functions to several other functions and thus improves sharing. This not only reduces the complexity of the network but also leads to significant savings on transistor-count.

2.4 Technology Mapping

A conventional way of mapping the netlist into a circuit is to use a standard cell library. It is known that the quality of a circuit is greatly influenced by the number of gates in the library. Now, defining a standard cell library is another implementation issue. In our work, we have proposed the technology mapping with *on-the-fly* cell generation, which does not use any standard cell library. In the netlist obtained from the multilevel decomposition, each node in the list represents a logic function of an arbitrary gate to be



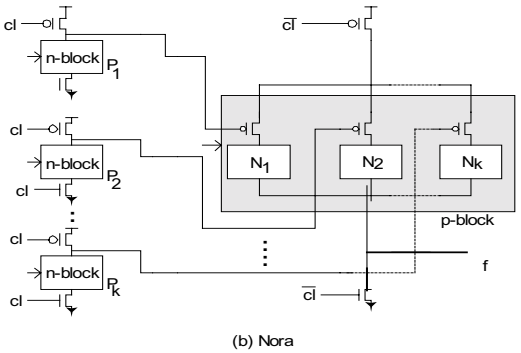


Fig. 2 Generic structure of 2-level dynamic Domino/Nora circuits

realized. However, a node can be treated as a cell only when it satisfies the *length* and *width* constraints. If a node has the values of *length* and *width* more than the upper limits specified, then cell-based multilevel decomposition is performed in order to satisfy the *length* and *width* constraints. The cell-based multilevel decomposition mechanism is illustrated with the help of the following example.

Example 2: Consider the function

$f_2 = x_1x_2x_3x_4x_{12} + x_6x_7x_8 + x_1x_9 + x_2x_7 + x_4x_8x_9 + x_2x_5x_6x_9x_{10}x_{11} + x_1x_8x_{11}$

Further assume that the *length* and *width* constraints for our technology are 3 and 4, respectively. After the decomposition satisfying the *length* and *width* constraints, the netlist becomes:

$A = x_1x_2x_3 \qquad B = x_2x_5x_6 \qquad C = x_9x_{10}x_{11}$

$D = x_6x_7x_8 + x_1x_9 + x_2x_7 + x_4x_8x_9$

$f_2 = A.x_4x_{12} + D + B.C + x_1x_8x_{11}$

The objective of the technology mapping with cell-based decomposition is to decompose a node so that it can be realized with cells of permissible size. At the same time, in order to ensure lesser number of levels, we are to create intermediate nodes in terms of primary inputs as far as possible. In practice, the multilevel decomposition and technology mapping steps are combined into single a cell-based decomposition step.

2.5 Delay Estimation

In order to calculate the delay of a dynamic CMOS circuit, we propose an estimation model to calculate the delay of a dynamic CMOS cell. Let us assume that the logic of a cell is represented by

$X = x_{11}x_{12}x_{13}..x_{1n_1} + x_{21}x_{22}x_{23}..x_{2n_2} + ... + x_{m1}x_{m2}x_{m3}..x_{mn_m}$

The structure of such a dynamic cell is shown in Fig. 3. Let us assume the following:

m = *width* of the cell i.e. the number of parallel paths

n = *length* of the cell i.e. the number of transistors in the longest chain

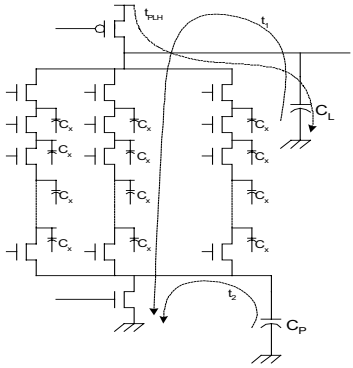


Fig. 3 Structure of a dynamic cell

R_p = On-resistance of pMOS transistor, where

$$R_p = \frac{V_{dd}}{k_p(V_{dd} - V_{Tp})^{\alpha}} + \frac{V_{dd}}{k_p\left[2(V_{dd} - V_{Tp})V_{dd} - \frac{V_{dd}^2}{2}\right]} \quad (1)$$

R_n = On-resistance of nMOS transistor, where

$$R_n = \frac{V_{dd}}{k_n(V_{dd} - V_{Tn})^{\alpha}} + \frac{V_{dd}}{k_n\left[2(V_{dd} - V_{Tn})V_{dd} - \frac{V_{dd}^2}{2}\right]} \quad (2)$$

Here, k_p, k_n are the gain factor of pMOS and nMOS transistors, respectively and V_{Tp} and V_{Tn} are the threshold voltage of pMOS and nMOS transistors, respectively. The constant α is 2 and 1.3 for long channel and short channel MOSFETs, respectively.

C_{dp} = drain capacitance of pMOS
C_{dn} = drain capacitance of nMOS
C_{gp} = gate capacitance of pMOS
C_{gn} = gate capacitance of nMOS
C_{int} = interconnect capacitance.

.

The delay of dynamic CMOS cell then can be given as

$$t_d = \frac{t_{PLH} + t_{PHL}}{2} \quad (3)$$

where, t_{PLH} = charging time of the load capacitance C_L

and t_{PHL} = discharging time of the pull-down network.

Now, the charging time of the load capacitance is given by

$$t_{PLH} = 0.69.R_p.C_L \quad (4)$$

where, the load capacitance C_L can be expressed as

$$C_L = C_{dp} + m.C_{dn} + C_{\text{int}} + (C_{gp} + C_{gn}) \quad (5)$$

Similarly, the discharge time of the pull-down network is given by

$$t_{PHL} = t_1 + t_2$$

where, t_1 = discharge time of C_L through the longest path in the pull-down network, and t_2 = discharge time of C_P. Thus,

$$t_{PHL} = 0.69\left[(n+1).C_L.R_x + \sum_{i=1}^{n-1}(i+1).2C_x.R_x + R_n.C_P\right] \quad (6)$$

Here, $R_x = R_n$ and $C_x = C_{dn}$, if the pull-down network is a network of nMOS transistors and $R_x = R_p$ and $C_x = C_{dp}$, if the pull-down network is of pMOS transistors, and $C_P = m.C_x + C_{dn}$.

2.6 Power Estimation

As switching power is the most dominant source of power dissipation in dynamic CMOS circuits, we have considered only the switching power in our power estimation. For a given dynamic cell there are several points where power dissipation occurs. Figure

4 shows a dynamic cell and various power consuming points in it. Here, P_1 = power to charge and discharge C_{L1}, P_2 = power to charge and discharge C_P, P_3 = power to charge and discharge all internal capacitances, and P_4 = power at inverter output i.e. at C_{L2}.

Switching power at C_{L1}

Switching power at C_{L1} occurs due to charging (precharge) and discharging (evaluation) the load capacitance C_{L1} . The switching power at C_{L1} can be given as

$$P_1 = E^0(X).C_{L1}.V_{dd}^2 \cdot f \quad (7)$$

here, $E^0(X)$ being the probability that the output of logic X is 0.

In the above expressions, the load capacitance C_{L1} can be calculated as follows:

$$C_{L1} = C_{dp} + m.C_{dn} + C_{\text{int}} + \left(C_{gp} + C_{gn}\right) \quad (8)$$

The switching activity of a logic X can be calculated as below:

Let the Boolean function X is in SOP forms with n product terms as

$$X = X_1 + X_2 + X_3 + ... + X_n$$

Here, a product term X_i is given by,

$$X_i = x_{i1} \cdot x_{i2} \cdot x_{i3} \cdot ...x_{im}$$

Now, $E^0(X_i) = 1 - p^1_{x_{i1}} \cdot p^1_{x_{i2}} \cdot p^1_{x_{i3}} \cdot ...p^1_{x_{im}}$ (9)

where, $p^1_{x_{ij}}$ indicates the probability of input signal x_{ij} being at '1'. With this, we have,

$$E^0(X) = E^0(X_1) \cdot E^0(X_2) \cdot E^0(X_3) \cdot ...E^0(X_n) \quad (10)$$

$$E^1(X) = 1 - E^0(X) \quad (11)$$

Probability of a transition at output is

$$E^{1\to 0}(X) = E^{0\to 1}(X) = E^0(X) \cdot E^1(X) \quad (12)$$

Switching power at C_P

It can be noted that the capacitance at node 2 (Fig. 4) will be charged and discharged at every clock cycles. Hence, it is independent on switching activity. It can be expressed as given below:

$$P_2 = C_P.V_{dd}^2 \cdot f$$

Internal switching power

There is a source of power consumption due to charge and discharge the internal node capacitances inside the logic cell. The total dynamic power consumed internally is :

$$P_3 = \left(\sum_{i=1}^{m} \sum_{j=1}^{\#Trans.\ inp\ ath\ i-1} \alpha_{ij} \cdot 2.C_x \right) \left(V_{dd} - V_{Tx}\right) V_{dd} \cdot f \quad (13)$$

where, $V_{Tx} = V_{Tp}$ if the transistor is in p-network else it is V_{Tn}, and α_{ij} is the switching activity of the internal node x_{ij}.

Switching power consumption at static CMOS buffer node C_{L2} can be expresses as given below:

$$P_4 = \left(E^1(X) \cdot E^0(X)\right) C_{L2}.V_{dd}^2 \cdot f \quad (14)$$

where, $C_{L2} = C_{dp} + C_{dn} + f_{out}(C_{gp} + C_{gn})$, f_{out} being the *fanout* of the cell under consideration.

As the Nora circuits are inverter free, this component is not present for the Nora logic circuits.

3. Algorithms

Our synthesis procedure comprises with the following steps:

1. Partitioning a BLIF circuit into subcircuits
2. Minimum unate decomposition of each subcircuit
3. Multilevel cell-based decomposition for technology mapping

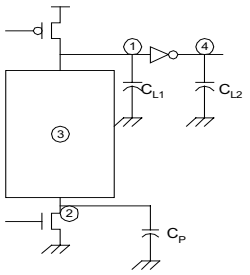


Fig. 4 Various power consuming nodes in a dynamic cell

The algorithms implementing these tasks are given in the following sub-sections.

3.1 Partitioning

With the partitioning, we are to decompose a graph (which is in *BLIF* form) into a number of partitions so that the number of inputs of a partition does not exceed a certain upper limit. We have proposed the algorithm *K_PARTITION* for partitioning any *BLIF* circuit into k partitions, $k \geq 1$ is given as below.

Algorithm K_PARTITION

Input: An acyclic graph G for a input *BLIF* circuit.

Output: k partitions of $G = \{P_1, P_2, ..., P_k\}$, where $k \geq 1$

Step 1. *Creation of initial partitions*

Let the number of inputs in each partition be less than the some upper limit, say δ.
Push all outputs node in a queue Q;
$i = 0$; $\#P_i = 0$; // $\#P_i$ is the number of primary input
While (Q is not empty) *do*
Pop a node n from Q;
i++;
Create_Partition (n, P_i);
EndWhile

Create_Partition(n, Pi)

Push n and all its successors in *BFS* fashion in q;
While ($\#P_i \leq \delta$ or q is not empty) *do*
Pop a node m from q;
If (inclusion of m does not exceed $\#P_i$ by δ) *then*
Add m into P_i
$\#P_i$ += number of inputs in m;
Else
Break the loop;
EndIf
EndWhile
While (q is not empty) *do*
Pop a node x from q;
Push x into Q;
EndWhile

Step 2. *Refinement of partitions*

In this step, we are to refine the partitioned graph so that total number of partitions can be reduced.
For all nodes $n \in G$
For all adjacent partitions of n
Find a partition P with maximum edges of n in it;
Move n from its current partition to destination partition P Such that $\#P \leq \delta$

3.2 Minimum Unate Decomposition

Based on the theory of two-level unate decomposition presented in [5], algorithm for minimum unate decomposition which decomposes a given function in terms of minimum numbers of unate functions is outlined below:

Algorithm MUD

Input: A Boolean function *f* as a sum of product form.

Output: $f = \overline{G}_0 \cdot G_1 + \overline{G}_2 \cdot G_3 + \overline{G}_4 \cdot G_5 + \cdots$, where $\overline{G}_i$, i = 0, 2, 4, … are the *positive unate* functions and G_i, i =1, 3, 5, … are the *negative unate* functions.

Step 1. *Tabularize the Boolean function f.*
Tabularize the given function f based on its true and false sets and the weight of the minterms.

Step 2. *Obtain the Initial Unate Decomposition (IUD).*
Obtain a decomposition table, so that no minterm covers any minterms in left columns.

Step 3. *Obtain the Minimal Unate Decomposition (MUD) table.*
Update *IUD* table, so that no minterm covers any minterms in right columns.

Step 4. *Obtain two-level decomposed form of*

$$f = \overline{G}_0 \cdot G_1 + \overline{G}_2 \cdot G_3 + \overline{G}_4 \cdot G_5 + \cdots$$

Example 3: Let us consider the case of a Boolean function f_3.

$$f_3 = \sum(3,5,7,8,11,12,14,15)$$

Two-level minimum unate decomposed form of f_3 can be obtained as:

$$\overline{G_0} = x_3 x_1 + x_4$$
$$G_1 = \overline{x}_4 + \overline{x}_3 \overline{x}_1$$
$$\overline{G}_2 = x_2 x_1 + x_4 x_3 x_2$$
$$G_3 = 1$$

3.3 Algorithm for Multilevel Unate Decomposition

Following is the algorithm to define the multilevel decomposition of a unate function:

Algorithm MLUD

Input: UDN = a set of netlist after unate decompsition of the entire circuit ($f_1, f_2, \ldots f_n$)

Output: MDN = a set of netlist after multilevel unate decomposition.

Step 1. *Remove duplicated unate blocks from the list of blocks.*

Let $UDN = \{G_{1\_0}, G_{1\_1}, \ldots G_{1\_x}, G_{2\_0}, G_{2\_1} \ldots G_{2\_y}, \ldots G_{n\_0}, G_{n\_1}, \ldots G_{n\_z}\}$

If $\exists (G_{i\_k} \text{ and } G_{j\_l}) \in UDN \mid G_{i\_k} \in f_i \text{ and } G_{j\_l} \in f_j \text{ and } G_{i\_k} = G_{j\_l}$ *then*

Remove G_{j_l} from the netlist *UDN*

Update the function definition

Endif

Step 2. *Decompose each unate block in multilevel.*

$MDN = \phi$

Forall $G_{i\_j} \in UDN$ *do*

Simplify the node G_{i_j}

Let $L = \{G_{i\_j}\}$

N = *Decompose(L)*

Forall $n \in N$ *do*

If !*Constraint*(n) *then*

n' = *Split*(n)

Upgrade N with n'

Endif

Endfor

$MDN = MDN \cup N$

EndFor

Procedures those are referred in *Algorithm MLUD* is defined as below:

Decompose(L)

Step 1. *Computation of all kernels*
Compute all kernels K of netlist L

Step 2. *Selection of best decomposition*

$P_{min} = \infty$
$D_{min} = L$
If $\#K > 0$ *then*
Forall $k \in K$ do
If Constraint(k) *then*
L' = *Factored*(L, k)
D = *Decompose*(L')
P = *EstimatePower*(D)
If ($P < P_{min}$) *then*
$D_{min} = D$
Endif
Endif
Endfor
Endif
Return(D_{min})

Split(n)

Let λ = length constraint and ω = width constraint

Step 1. *Split a product term in n to satisfy length constraint*
For each $p \in n$ *do*
If Length(p) > λ *then*
Break p with minimum possible intermediates
Endif
Endfor

Step 2. *Split n to satisfy width constraint*
If Width(n) > ω then
Break n with minimum number of intermediates;
Each intermediate is as far as possible in terms of primary inputs.
Endif

4. Implementation and Experimental Results

We have implemented our entire synthesis method on *SUN Ultra Sparc 10* systems and using C++ programming embedding with STL and GTL of ATT. We have tested our implementation with ISCAS benchmarks. Value of transistor's parameters are extracted using BSIM4 model and for 0.18μ technology as a particular instance.

The synthesis of dynamic CMOS circuits starts with partitioning an input circuit in BLIF format resulting smaller sub-circuits. Each sub-circuit is then transformed into PLA form (in terms of minterms) and *disjoint decomposition* is carried out to contain non-overlapping minterms only. After preprocessing the input, our tool decomposes a Boolean function into a set of positive and negative unate sub-functions based on the algorithm MUD. It should be noted that, algorithm MUD can handle a single logic function at a time, whereas a circuit is in general multi-output in nature. This problem has been sorted out by extracting one function at a time in a partition and then obtaining its unate decomposed version (in the form of a netlist) and storing the result in a temporary file. This process is to be repeated for all functions in the partition and finally giving the netlist for a partition. Our tool then performs cell-based multilevel decomposition for nodes in the netlist in order to satisfy the *length* and *width* constraints. After this multilevel decomposition is over, our tool produces final netlist of synthesized circuit.

In order to realize static CMOS circuits, we have used the Berkeley SIS tools. The netlist is optimized with the *script.rugged* command in SIS. Technology mapping is performed using *44-2.genlib* and with the option of minimum area. The result of the static CMOS circuits for ISCAS benchmarks is shown in Table 1.

Results of Domino and Nora circuits for ISCAS benchmarks are shown in Table 2. During the multilevel decomposition, for the constraints of a cell, we have chosen *length* = 4 and *width* = 6. In order to compare the power consumption for three classes of circuits, operating frequency for a given circuit is chosen as the minimum frequency among the circuits with three different techniques. For power estimation, we have considered only the switching power consumption, because the static power consumption is usually two orders of magnitude smaller in comparison to the switching power.

Experimental results show that dynamic circuits are faster, Domino realizations provide 44% lesser delay compared to their static counterparts. However, the delay in Nora circuits is slightly higher than Domino circuits. Transistor counts (a measure of area) of both the Domino and Nora circuits are on the average 35% and 44% less, respectively compared to the static CMOS circuits. Domino circuits consume 63% and Nora circuits require 58% lesser energy than the static CMOS circuits, which are superior to the existing results.

6. Conclusions and Future Works

In this paper, we have proposed a novel logic design approach for the synthesis of large dynamic CMOS circuits. As Domino/Nora logic styles can implement only non-inverting logic, we have overcome this problem by using unate decomposition. This also makes Nora circuits are completely inverter free. 'On-the fly' cell generation approach adopted in the synthesis process helps to realize circuits with lesser number of levels. The synthesized circuits are found to be superior in terms of area, delay and energy requirement compared to their static counterparts and better than the reported results.

In our work, we have concentrated only on the dynamic power consumption, which is the most dominant component in case of dynamic circuits. But, for battery-operated hand held portable VLSI circuits based on deep sub-micron technology, leakage power is another important source of power consumption, which should not be ignored. Also we have not taken into consideration charge sharing and charge leakage problems, which may be overcome by incorporating weak pMOS devices in the pull-up network. In the Domino logic style, the inverter may be replaced by more complex static gates [7]. We propose to extend our work in these directions.

Table 1: Static CMOS Circuits

Bench marks	PI/PO	f in MHz	Delay (ns)	#Transistor	Level	Power (μW)	Energy (fJ)
C432	36/7	125	7.98	592	24	466.36	3721.55
C499	41/32	190	5.25	1880	21	1410.21	7403.60
C880	60/26	192	5.19	1412	34	1193.19	6192.65
C1355	41/32	163	6.11	1880	20	1318.66	8057.01
C1908	33/25	149	6.67	1756	34	1331.38	8880.30
C2670	233/140	145	6.89	3453	19	1872.61	12902.28
C3540	50/22	92	10.79	3878	38	1487.14	16046.24
C5315	178/123	116	8.59	6058	42	3413.30	29320.25
C6288	32/33	107	9.32	11222	35	4061.67	37854.76
C7552	207/108	145	6.86	8214	38	6262.47	42960.54
				40345			

Table 2: Dynamic CMOS Circuits

Bench mark	f in MHz	Domino Circuits							Nora Circuits						
		#Transistor	#Level	Delay (ns)	% qt	Power (μW)	Energy (fJ)	% qE	#Transistor	#Level	Delay (ns)	% qt	Power (μW)	Energy (fJ)	% qE
C432	125	575	10	3.08	61.4	419.62	1292.4	65.3	479	7	3.31	58.5	438.29	1450.7	61.0
C499	190	1707	11	3.87	26.3	927.84	3590.7	51.5	1601	8	4.14	21.1	951.07	3937.4	46.8
C880	192	961	14	4.04	22.1	568.33	2296.0	62.9	801	10	4.32	16.8	607.19	2623.1	57.6
C1355	163	1183	14	3.16	48.3	734.05	2319.6	71.2	1044	10	3.38	44.7	787.14	2660.5	66.9
C1908	149	1498	16	3.63	45.6	977.62	3548.8	60.0	1404	11	3.88	41.8	1023.1	3969.7	55.3
C2670	145	2538	16	3.71	46.1	1390.5	5158.9	60.0	2474	11	3.84	44.3	1566.0	6013.4	53.4
C3540	92	3584	25	4.69	56.5	1070.6	5021.2	68.7	3309	18	4.42	59.0	1113.2	4920.5	69.3
C5315	116	5733	30	6.29	26.8	2026.5	12746.8	56.5	4587	22	6.74	21.5	2201.6	14839.2	49.4
C6288	107	2699	12	3.58	61.6	3307.3	11840.8	68.7	2159	8	3.74	59.9	3518.2	13157.9	65.2
C7552	145	5824	18	3.78	44.9	4090.4	15461.9	64.0	4932	12	3.91	43.0	4524.1	17689.2	58.8
		26302			*43.9%*	*31.5%*		*62.9%*	*22790*			*41.1%*	*26.9%*		*58.4%*

Acknowledgement

This work was supported by Intel Corporation, USA, grant #7310. We are thankfull to James W. Tschanz of Intel Corporation for his valuable comments and suggestions.

References

1. R. H. Krambeck, C. M. Lee and H. S. Law, "*High-Speed Compact Circuits with CMOS*", in IEEE Journal of Solid State Circuits, pp. 614-619, Vol. SC-17, No. 3, June 1982.
2. N. F. Goncalves and H. J. De Man, "*NORA: A Race free Dynamic CMOS Technique for Pipelined Logic Structures*", in IEEE Journal of Solid State Circuits, pp. 261-266, Vol. 18, No. 3, June 1983.
3. M. R. Prasad, D. Kirkpatrick and R. K. Brayton, "*Domino Logic Synthesis and Technology Mapping*", in International Workshop on Logic Synthesis, May 1987.
4. Min Zhao and Sachin S. Sapatnekar, "*Technology Mapping for Domino Logic*", in IEEE/ACM Proc. of Design Automation Conference, pp. 248-251, 1998.
5. A. Pal and A. Mukherjee, "*Synthesis of Two-level Dynamic CMOS Circuits*", in IEEE Proc. of International Workshop on Logic Synthesis, May 1999.
6. R. K. Brayton, R. L. Rudell and A. L. Sangiovanni-Vincentelli, "*MIS: A Multiple-Level Logic Optimization System*", in IEEE Trans. on Computer Aided Design, pp. 1062-1081, Vol. 6, No. 6, 1987.
7. Tyler Thorp Gin Yee and Carl Sechan., "*Domino Logic Synthesis Using Complex Static Gates*", in IEEE/ACM Proc. of International Conference on Computer Aided Design, pp. 242-247, 1998.