

SYNTHESIS OF READ-ONCE DIGITAL HARDWARE WITH REDUCED ENERGY DELAY PRODUCT

Padmanabhan Balasubramanian^a and Siddhagunta Theja^b

^aSchool of Computer Science, The University of Manchester, Manchester, LAN, UK
padmanab@cs.man.ac.uk

^bSasken Communication Technologies Ltd., Bangalore, Karnataka, INDIA
sthejar@sasken.com

ABSTRACT

This paper presents a low power driven synthesis framework for the unique class of non-regenerative Boolean Read-Once Functions (BROF). A two-pronged approach is adopted, where the satisfiability of the functionality is first ensured at the logic level on the basis of the proposed 'hybrid synthesis method'. The resulting circuitry is contrasted with the reduced disjunctive normal form (DNF), resulting from standard two-level synthesis tool, ESPRESSO and conjunctive normal form (CNF) expression obtained via, the conventional Tabulation method. The gate level schematics are then translated into MOS transistor descriptions via, static CMOS and stacked CMOS implementation styles. Leakage Control Transistors (LCTs) are also inserted between the pull-up and pull-down network nodes, so as to minimize the overall power consumption of the digital logic circuits designed. Furthermore, the effect of transistor re-ordering on the delay of the resulting CMOS digital designs is also investigated. The adopted synthesis procedures are all evaluated based on a common Energy Delay Product (EDP) metric. The SPICE simulation results obtained for a 350nm TSMC CMOS process are promising, as it reports 41.5% savings in EDP, 10.5% reduction in power and 17.7% decrease in delay for the proposed method, on an average, over the best of conventional methods.

Keywords: Read-Once Function (ROF), Low power design, EDP, static CMOS, stacked CMOS.

1. INTRODUCTION

Minimization of power consumption of CMOS based circuit designs is generally possible at five levels (Chandrakasan et al., 1995) viz. technology, circuit, logic, architecture and algorithm. Hence the search for the optimal solution must include, at each level of abstraction, 'a design improvement loop'. Obviously, collecting the feedback on the impact of the different choices on a level-by-level basis reduces the time required to arrive at a design decision. In this context, this issue is dealt with in this paper at the logic and circuit levels for an important class of logic functions, namely the Boolean Read-Once functions (BROF). BROFs, a special family of monotone Boolean functions, have interesting special properties and account for a large percentage of functions, which arise in real circuit applications. They have also generated much interest in the field of computational learning theory (Golombic et al., 2001).

Although algorithms exist, that obtain exact minimum sum-of-products (SOP) or disjunctive normal form (DNF) expressions in two-level logic for a large set of functions including BROFs (Coudert, 1994), the majority of practical systems use heuristic logic minimization algorithms. These produce irredundant sum-of-products expressions that are not necessarily optimal. Here, optimality criteria are defined by the critical design metrics, which qualify as the figure-of-merit for the circuits implemented. For example, MINI (Hong et al., 1974),

PRESTO (Svoboda and White, 1979), ESPRESSO (Brayton et al., 1984), and others (Nguyen et al., 1987) (Rudell and Sangiovanni-Vincentelli, 1987) (McGeer et al., 1993) even produce non-optimal irredundant DNFs. Due to restriction on the fan-in of the library cells used for actual circuit realizations; typical practical implementations of a logic function utilize a multilevel network of logic elements. Even an industry standard synthesis tool would construct a multilevel Boolean network corresponding to an RTL (register-transfer level) description of a design. Next, this network is optimized using several technology-independent techniques. Finally, technology-dependent optimization transforms the technology-independent circuit into a network of library gates in a given technology. The simple cost estimate of total literal count of the factored form of the logic function at the technology-independent stage is replaced by more concrete, implementation-driven estimates after technology mapping. Mapping is constrained by factors such as the available gates (logic functions) in the technology library, the drive sizes for each gate and the power, delay and area characteristics of each gate.

Multi-level binary logic circuits are generally represented in an easy and efficient manner by directed acyclic graphs (DAG). Rooted Binary Directed Acyclic Graphs (Rooted BDAG) usually represent combinational logic circuits in a compact multi-level form. A DAG can be unfolded in such a way that multiple fan-out nodes do not exist, except for the circuit inputs. Each internal node is labeled with a Boolean operator (AND or OR). Henceforth, the primitives used for logic level realization shall be classified under an Atomic Operator set (AO) (Devadas et al., 1994), which would comprise AND gate, OR gate and an inverter as elements, whose fan-in and fan-out are described by a 2-tuple as (2,1), (2,1) and (1,1) respectively. The definition of the atomic operator set also makes it sufficiently clear that complementary inputs are not assumed to be present, as is the general case with cell based IC designs. Although double-rail inputs are usually available for designs targeting programmable logic devices, for standard cell based designs; single-rail inputs are preferred, as they tend to consume lesser hardware area overhead.

Three different gate level netlist information for a given logic functionality are obtained from the technology-independent synthesis stage, two of them (CNF and DNF) (Kohavi, 1999) based on conventional techniques (McGeer et al., 1993) (McCluskey, 1956). It is well known that, traditionally, DNF and CNFs are generally implemented using NAND logic and NOR logic respectively. The realizations then undergo transformations according to the proposed heuristic to result in logic implementations with lesser transition activity. The two-level NAND and NOR digital circuits and the transformed logic functions are then transformed into multi-level logic, concomitant with equivalent DAG representations and described using static CMOS and stacked CMOS logic styles. With a view to reduce the leakage power component for ultra sub-micron ranges, LCTs (Hanchate and Ranganathan, 2004) are then introduced between the pull-up and pull-down network nodes. The effect of input reordering on the delay parameter of the three different implementations has been subsequently determined at the gate level. To validate our proposition, EDP has been chosen as the quantitative figure-of-merit (Guyot and Abou-Samra, 1998). The simulation results demonstrate that the application of transformations to the minimized functionalities resulting from standard synthesis tools enables low power design.

The remainder of this paper is organized as follows. Section 2 gives background information about the terminologies used. Section 3 describes the motivation for this work. We also discuss a quick estimation of transition count here. Sections 4 and 5 discuss the logic level and circuit level optimization methodologies adopted. Section 6 gives the proposed algorithm. Section 7 cites an illustrative example. Section 8 depicts the various functionalities considered and also the simulation results obtained. Section 9 highlights the conclusions of this research.

2. PRELIMINARIES

2.1. Boolean Read-Once Function (BROF)

A Boolean function, F , is called a read-once function (ROF), if it has a factored form in which each variable appears exactly once. Hence the complexity of a read-once formula is the least, as it has no variable repeated. The reason that ROFs are also known as non-repeatable tree functions is that its parse tree has no variable repeated (Golumbic et al., 2001).

2.2. Description of Sum term (Product term)

A new terminology is proposed, namely the description set of a Boolean term (sum term or product term). The description set of a sum term [product term], shall be represented by the notation $D(S_i)$ [$D(P_i)$]. $D(S_i)$ specifies the set of all literals in their actual form, that the particular sum term S_i is dependent upon for its evaluation to a logic value of '0' and $D(P_i)$ indicates the set of all literals in their respective form, that a product term P_i depends upon for its evaluation to a logic value of '1'.

For e.g. If $F = a'bd + b'cd'$, where $P_1 = a'bd$ and $P_2 = b'cd'$, then $D(P_1) = (a',b,d)$ and $D(P_2) = (b',c,d')$. Here P_1 and P_2 are BROFs. The above argument holds good for sum terms as well. If $G = (a+b+c') \cdot (d'+e+f)$, where $S_1 = (a+b+c')$ and $S_2 = (d'+e+f)$, then $D(S_1) = (a,b,c')$ and $D(S_2) = (d',e,f)$. In this case, S_1 , S_2 and G are all BROFs.

2.3. Sum of Disjoint Products (SoDP)

A logic function in its simplest and reduced form can be represented as a SoDP, if and only if (1) is satisfied. Here '+' stands for the Boolean OR operator. It is given as:

$$F = P_j + P_k; \text{ such that } \{D(P_j)\} \cap \{D(P_k)\} = \{ \} \quad (1)$$

2.4. Product of Disjoint Sums (PoDS)

A logic function in its minimized form could be identified as PoDS, if and only if (2) is satisfied. Here '•' stands for the Boolean AND operator. Hence:

$$F = S_j \cdot S_k; \text{ such that } \{D(S_j)\} \cap \{D(S_k)\} = \{ \} \quad (2)$$

3. MOTIVATION FOR THE WORK

For well-designed sub-micron digital CMOS circuits, the dominant source of power consumption is due to the charging and discharging of the node capacitances and is computed as,

$$P = 0.5 \cdot \alpha \cdot C_L \cdot V_{dd}^2 \cdot f_{clk} \quad (3)$$

where ' α ' is the switching activity, ' C_L ' is the load capacitance (modeled as the sum of intrinsic capacitance of the device, extrinsic load capacitance and net capacitance) at the output node (Gallant and Al-Khalili, 1999), ' V_{dd} ' is the supply voltage and ' f_{clk} ' is the clock frequency.

It is clear from the above equation that reduction in switching activity leads to minimization of power consumption. It has been our observation, after extensive studies and validation that inverters associated with primary inputs tend to experience the greatest transition activity, which results in increased power dissipation. This is much more dominant in the case of both SoDP and PoDS functions, whose sub-functions are BROFs, where input literals predominantly appear in complementary form. Hence our novel proposition is to do away with primary input inverters, although this may lead to introduction of inverting buffers at the intermediate outputs, which generally exhibits minimal activity. Our preliminary inference based on computation of the transition count using the following novel formulation has been the inspiration for this work. SPICE based simulation results further rationalized our understanding. Though quite a few logic minimization schemes especially targeting low power realizations have existed in literature; to our knowledge, this has been the first research attempt in this direction, especially targeting the special class of BROFs.

3.1. Classification of Logic primitives

The entire class of basic logic gates (library cells) have been grouped, primarily on the basis of distribution of elements between their 'ON' and 'OFF' sets as follows. However, the following classification holds good only for a uniform distribution of input patterns.

3.1.1. Singleton logic gate

A logic gate, 'Gk', is referred to as "Singleton", if it has either a singleton ON set or a singleton OFF set. The AND, OR, NAND and NOR gates functionality can be grouped under this

category. The AND and NOR gates have a singleton ON set, while OR and NAND gates have a singleton OFF set.

3.1.2. Equivalent logic gate

A logic gate, 'Gk' is said to be "Equivalent", if it comprises of an equivalent distribution of elements in both its ON and OFF sets. The NOT, XOR and XNOR gates functionality belong to this group.

3.2. Theoretical estimation of upper bound of Transition Count (TC)

Let 'N₀' and 'N₁', represent the finite number of minterms/maxterms, which result in logic '0' and logic '1' in the output column of the truth table of a logic gate, say Gk. In other words, they represent the cardinality of the OFF set (Gk_{OFF}) and ON set (Gk_{ON}) of the logic gate. Let 'N_s' indicate the total number of unique states. Then we have,

$$N_s = 2^n = (N_0 + N_1) \quad (4)$$

Whenever, there is a state change from Gk_{ON} to Gk_{OFF} or vice-versa, it is construed as a transition. Therefore, the transition activity is,

$$TC \text{ Gk} = 2 \times (N_0 + N_1 - 1) \quad (5)$$

Since either N₀ = 1 or N₁ = 1 in case of Singleton gates (AND, NAND, OR and NOR gates), we can easily conclude that,

$$TC \text{ Gk} = 2 \times N_0 \times N_1 = 2 \times |Gk_{ON}| \times |Gk_{OFF}| \quad (6)$$

Relying on the principle of mathematical induction, the upper bound of the switching activity for the case of Equivalent gates (NOT, XOR and XNOR gates) is determined as,

$$TC \text{ Gk} = (N_0 + N_1) [(N_0 + N_1) - ((N_0 + N_1) / 2)] \quad (7)$$

$$\text{Hence, } TC \text{ Gk} = ((N_0 + N_1)^2 / 2) \quad (8)$$

As N₀ = N₁ for the class of equivalent gates, the above equations can be simplified as,

$$TC \text{ Gk} = ((N_0)^2 / 2) = ((N_1)^2 / 2) = 2 \times N_0 \times N_1 \quad (9)$$

Alternately, it is given as,

$$TC \text{ Gk} = 2 \times N_0 \times N_1 = 2 \times |Gk_{ON}| \times |Gk_{OFF}| \quad (10)$$

3.3. Simplifying the computation of TC for gates of two-level and three-level logic circuits, with uniform input distribution

The computation of TC (indicative of switching activity) using (6) or (10) seems to be apply for all the gates in a logic network irrespective of the level, in which they appear in a circuit.

However, for two-level logic circuits, the computational complexity associated with the estimation of TC for an individual gate can be greatly reduced if its evaluation would purely depend upon a relationship between the number of primary inputs and the number of actual inputs to a gate. This will enable a quicker evaluation of the TC, as a theoretical measure for two-level logic circuits. Otherwise, the enumeration of Gk_{ON} and Gk_{OFF} for each gate in a logic design may take a longer time for initial estimation of TC. This may become cumbersome for larger designs.

When inverters are required to obtain complementary versions of the primary input signals, as is the case with standard cell-based IC designs, the traditional two-level logic would essentially be transformed into three-level logic. In this case, (6) or (10) could be used to determine the transition count of each of the primary input inverters or by using (12) to arrive at a quick estimate without having to compute the cardinality of Gk_{ON} and Gk_{OFF} for each of them. For the gates present in the first level of a two-level logic circuit (when there are no primary input

inverters) or in the intermediate level, i.e. in the second level of a three-level logic circuit, where the first level is defined by those of the inverters associated with primary inputs, equations mentioned in section 3.3.2 would be helpful in quickly evaluating the transition count of all the individual gates. However, it should be noted that these mathematical formulations would only hold good for the case, when the inputs are uniformly distributed and uncorrelated.

3.3.1. Computing the upper TC bound for the primary input inverters in a three-level circuit

Let 'NI' be the number of primary inputs, which form the basic support of an arbitrary logic function, F and 'NG' be the number of inputs to a gate. Then the upper bound of the transition count, TC of the inverting gates (occupying the first-level) of a three-level logic circuit would be given by,

$$TC Gk = N_s \times (2^{NI-1}) = [2^{NI} \times (2^{NI-1})] \quad (11)$$

Hence, it obviously follows that,

$$TC Gk = 2^{(2NI-1)} = (N_s^2)/2 \quad (12)$$

3.3.2. Computing the upper TC limit for the first-level gates of a two-level logic circuit and the intermediate level gates of a three-level logic circuit

Let us now evaluate the maximum TC limit for the gates present in the first-level in case of a two-level logic circuit and the gates present in the intermediate in case of a three-level binary logic circuit.

The transition count, TC of such a gate could be generalized based on the principle of mathematical induction as follows.

$$TC Gk = 2 \times [(2^{NI-NG}) \times (2^{NI} - 2^{NI-NG})] \quad (13)$$

On further simplification, we get,

$$TC Gk = 2^{(2NI-2NG+1)} \times (2^{NG} - 1) \quad (14)$$

If the number of inputs to a gate in the first level of a two-level logic circuit or at the intermediate level in a three-level logic circuit is the same as that of the number of primary inputs of the corresponding logic function, then (14) would be reduced as follows,

$$TC Gk = 2 \times (2^{NG} - 1) \quad (15)$$

It should be noted that (15) only represents a special case of (14). The above equations were derived methodically in a systematic manner based on the principle of mathematical induction. It could be noticed in this context that (12), (14) and (15) are dependent only upon 'NI' and 'NG'. Therefore, this enables an easier and faster calculation of the transition count at a gate output node, depending upon its level in an initial realization.

3.3.3. Sample cases

Let us consider two functions to explain the computation of TC for a Boolean specification in two-levels and three-levels. It should be borne in mind that (14) could be used to directly calculate TC for gates present in the first level of a two-level logic circuit or the gates present at the intermediate level of a three-level logic circuit, provided that the inputs exhibit uniform distribution and uncorrelated; similar condition for (12). For inputs exhibiting correlation, (6) or (10) would still hold good for all the gates, irrespective of their presence at any level of any type of circuit (two-level or three-level); however, the computational cost would become higher.

3.3.3.1. Case 1 (Two-level logic)

Let us consider an arbitrary 4 variable logic function, given by,

$$Y = (y_1+y_2) \cdot (y_1+y_3+y_4) \quad (16)$$

Let us have the assumption that all the primary inputs exhibit a uniform distribution and are uncorrelated. Therefore, $P(a=0) = P(b=0) = P(c=0) = P(d=0) = 0.5$. A similar probability value holds good for the inputs present in a logical HIGH state.

The sum terms, (y_1+y_2) and $(y_1+y_3+y_4)$ would be realized in the first-level of a two-level logic circuit. Here, primary input inverters are not necessary and so the AND gate necessary for conjunction of the two sum terms would comprise the second level of the logic realization.

In the first logic level, the OR gate needed to realize the logical sum, (y_1+y_2) , would have $|GK_{ON}| = 12$ and $|GK_{OFF}| = 4$. Hence as per (10), TC of this AND gate would be 96. The number of inputs for the OR gate needed to realize the above product (NG) is 2 and the number of primary inputs for the function Y (NI) is 4. Substituting these values in (14), we obtain a similar transition count of 96. For the sum term, $(y_1+y_3+y_4)$, $|GK_{ON}| = 14$ and $|GK_{OFF}| = 2$. Substituting these values in (10), TC for this OR-ing would be 56. This matches with the TC computed using (14), wherein NI = 4 and NG = 3. The integer measure of the transition count for Y, considering a two-level logic realization is then 262.

3.3.3.2. Case 2 (Three-level logic)

Let us consider a 4 variable BROF, given by,

$$Z = a'b' + cd \quad (17)$$

Let us assume that all the inputs are random inputs (for e.g. white noise) which are not correlated. This means that $P(a=1) = P(b=1) = P(c=1) = P(d=1) = 0.5$. Let us also assume a similar probability value for the complementary state of the input binary signals.

The product terms, $a'b'$ and cd would be realized in the second and first logic levels of a three-level logic circuit respectively. The inverters used to obtain the negation of the input signals 'a' and 'b' would constitute the first level and the OR gate needed for disjunction of the two product terms in order to realize F would form the third level of the logic implementation.

Here, in the first logic level, the AND gate needed to realize the logical product, cd , would have $|GK_{ON}| = 4$ and $|GK_{OFF}| = 12$. Hence as per (10), TC of this AND gate would be 96. The number of inputs for the AND gate needed to realize the above product is given as, NG = 2 and the number of primary inputs for the function F is given by, NI = 4. Substituting these values in (14), we obtain a similar transition count of 96. This would be the same for the product term, $a'b'$ as well. Hence, the integer measure of the transition count for Z, considering a three-level logic realization would then be 574, with input inverters alone accounting for 45% of the total.

In the light of the above discussions, we also make it succinctly clear that computation of TC could at the best serve as a useful forecasting measure at the technology-independent stage and it may not be able to predict the actual power savings that is likely to result after the technology-mapping phase in all cases; nor does it forecast any improvement/degradation in speed metric evaluated after the physical implementation stage, as it is independent of device parameters.

4. LOGIC LEVEL OPTIMIZATION

The complexity of a combinational logic circuit designed, roughly speaking, depends on the number of its constituent gates and that of a gate depends upon its number of input literals. For practical applications, circuits with minimum number of gates are preferred, as they tend to have a direct bearing on the hardware implementation and area overhead, power consumption and performance. The conventional methods of implementing a reduced SOP form by NAND logic and a minimal POS form by NOR logic (Kohavi, 1999) is contrasted with the proposed method, designated as '*hybrid synthesis method*', which favours a wise combination of NAND, NOR and NOT gate primitives, from a low power perspective. Our method, besides ensuring a minimum area solution, for a majority of the samples, has also enabled improved EDP metric, substantiated by the simulation results obtained.

Apart from affecting a minimum power solution, the proposed technique has also resulted in considerable improvement in delay, which is justified by the average delay parameter obtained for the samples considered in this paper, shown in Fig. 1 and Fig. 2 for the case of static and stacked CMOS circuits with embedded leakage control transistors respectively.

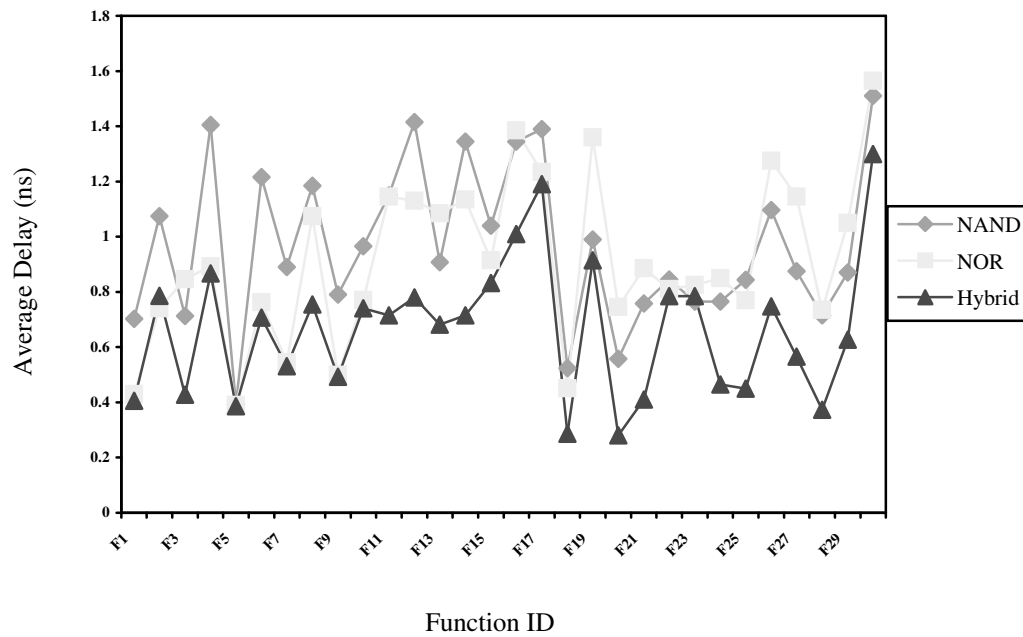


Figure 1. Comparison of average delay based on static CMOS circuits with LCTs

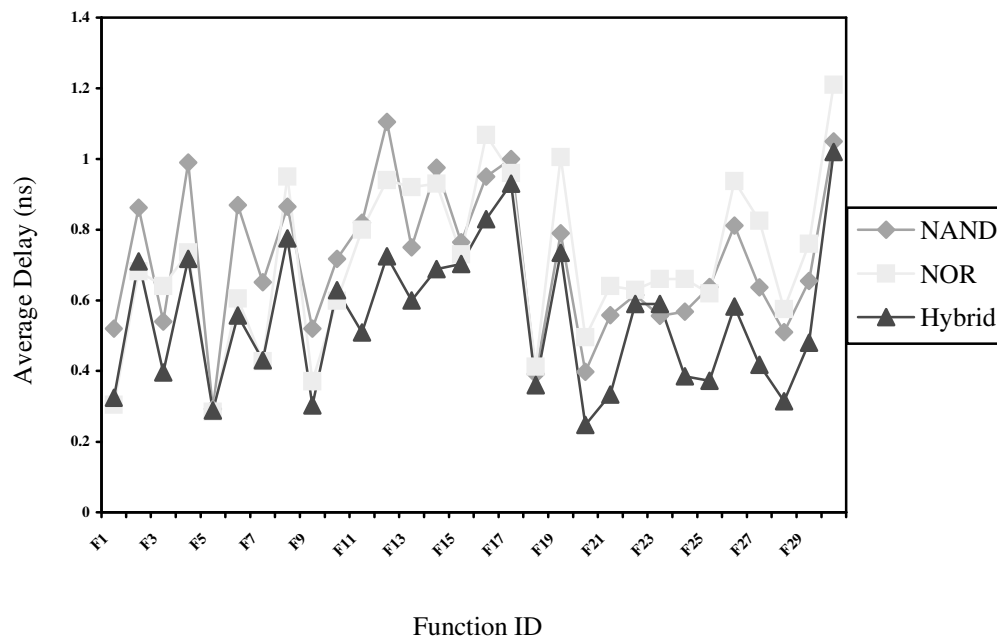


Figure 2. Comparison of average delay based on stacked CMOS circuits with LCTs

5. CIRCUIT LEVEL MEASURES

A couple of circuit level optimization approaches have been introduced to study their significance in minimizing the EDP, whilst satisfying the requisite Boolean functionality. The effect of transistor reordering in reducing delay without transistor sizing, formulated as a delay

optimization algorithm in (Carlson and Lee, 1995) has been included in our analysis. The re-ordering scheme followed for static CMOS and stacked CMOS NAND gates with LCTs is top critical re-ordering; while for static CMOS and stacked CMOS NOR gates with LCTs, bottom critical re-ordering is adopted. The simulation based reasoning is highlighted in Table 1.

Table 1. Effect of Input re-ordering on primitive gate metrics

Type of Re-ordering	$T_{(Avg.)}$ (in ns)	Power dissipation (in Watts)	EDP ($\times 10^{-30}$ Js)
2-input NAND gate			
Static-CMOS with LCTs			
Top critical re-ordering	0.032	1.4012 nW	1.4348
Bottom critical re-ordering	0.035	1.7030 nW	2.0862
Stacked-CMOS with LCTs			
Top critical re-ordering	0.0425	1.4012 nW	2.5309
Bottom critical re-ordering	0.065	1.7030 nW	7.1952
2-input NOR gate			
Static-CMOS with LCTs			
Bottom critical re-ordering	0.11	67.4143 pW	0.8157
Top critical re-ordering	0.11	109.0497 pW	1.3195
Stacked-CMOS with LCTs			
Bottom critical re-ordering	0.07	67.4143 pW	0.3692
Top critical re-ordering	0.12	109.0423 pW	1.5441

Though (Hanchate and Ranganathan, 2004) advocate the suitability of LCTs for static CMOS logic, its adaptability with stacked CMOS implementation style has been investigated for the first time in this work, as the latter facilitates effective wave-pipelined circuit designs (W.J. Kim and Y.B. Kim, 2003). Besides, static CMOS logic ensures a lower EDP value, which corroborates the findings cited in (Hanchate and Ranganathan, 2004). The usefulness of LCTs in reducing leakage power consumption, which is a dominant source in sub-nanometer designs, has been included in the static CMOS realizations, although their impact in reducing the average power dissipation has been ascertained in this paper, as the anomaly between the different power consumption components will be actually reflected only in the total power consumption. The other reason being the necessity to compute EDP.

6. PROPOSED ALGORITHM

A heuristic has been proposed to effect optimization at the technology-independent synthesis phase to reduce the transition activity at all the gate output nodes. It is given below.

Algorithm:

Start: The input and output descriptions of a given logic function are in standard PLA format

- Step 1: Minimize the given Boolean function, F , using standard two-level synthesis, such that $F = F_n \cup F_{n+1} \cup F_{n+2} \cup \dots F_t$; $\cup \Rightarrow$ '+' (Boolean OR) or '.' (Boolean AND); where $F_n, F_{n+1}, F_{n+2}, \dots, F_t$ are all sub-functions describing the original functionality F ;
- Step 2: Compute TC using equation (6) or equation (10) for F and for $F_n, F_{n+1}, F_{n+2}, \dots, F_t$ as well;
- Step 3: For every literal mapping ($z \rightarrow z'$) associated with $D(F_n)$,
- Step 4: Compute TC for such literal transformation (TC_z) as 2^{2n-1} , where $n = \text{ls}[F]$;
- Step 5: Total TC = $TC(F) + \sum_{r=0}^t [TC(F_{n+r})] + \sum_{z=0}^p TC_z$; for 't' gates and 'p' input inverters
- Step 6: If $s[F_n] = \{x_i\} \cup \{x_j'\}$; ($s[F]$ = support of F); and the conditionality $i \cap j = \phi$ holds well,
- Step 7: For every $x_k \in \{x_i\}$ and $x_l \in \{x_j'\}$, $x_k \cap x_l = \phi$;
likewise, for every $x_m \in \{x_j'\}$ and $x_n \in \{x_i\}$, $x_m \cap x_n = \phi$; also $\{x_i\} \cap \{x_j'\} = \{ \}$
- Step 8: For $\{x_j'\}$, apply De-Morgan's theorem, thereby,
- Step 9: For say, $x_o', x_p', x_q' \in \{x_j'\}$, perform the following transformations
 $(x_o' \wedge x_p' \wedge x_q') \Leftrightarrow (x_o \vee x_p \vee x_q)'$ and $(x_o' \vee x_p' \vee x_q') \Leftrightarrow (x_o \wedge x_p \wedge x_q)'$
- Step 10: Transform $(x_o \wedge x_p' \wedge x_q')$ as $(x_o \wedge (x_p \vee x_q)')$;
similarly $(x_o \vee x_p' \vee x_q')$ $\Leftrightarrow (x_o \vee (x_p \wedge x_q)')$; $\wedge \Rightarrow$ (logical AND), $\vee \Rightarrow$ (logical OR)
- Step 11: Continue steps 10 and 11, till all terms in the minimized version of F are exhausted, else
- Step 12: Compute total TC as, $TC = TC(F) + \sum [TC(\text{transformed sub-functions})] + \sum_{z=0}^p TC_z$
- Step 13: A power optimal solution would have been obtained
- End;

7. ILLUSTRATIVE EXAMPLE

The following example illustrates the low power methodology of the proposed algorithm. The sequencing of operations leading to a lower transition activity solution is described using this example by correlating with the subsequent steps of the proposed heuristic. Consider the compressed ON-set description of a 7-variable Boolean function, given by, $Y_{ON} = \{(100011x), (011xxx0), (xxx100x)\}$. Here 'x' accounts for a variable permutation.

The reduced DNF derived after step 1 (henceforth a 'step' mentioning would implicitly refer to that of the above algorithm) is given by,

$$Y_{ON} = [ab'c'd'ef + a'bcg' + de'f'] \quad (11)$$

Using (6) and (10), we compute the respective TC (as in step 2) for the read-once Boolean sub-functions ($ab'c'd'ef$), ($a'bcg'$) and ($de'f'$) as 504, 1920 and 3584, while the OR-ing of the sub-functions is associated with a TC of 5150. An input literal translation (step 3), such as for the mapping ($a \rightarrow a'$) accounts for a TC (as per step 4) of 8192. This is applicable for all individual literal complementations. Hence Y_{ON} is found to have an overall TC (step 5) of 68502.

The transformed minimized functionality, obtained on the basis of the proposed hybrid technique (as highlighted in step 10), using De-Morgan’s laws (corresponding to steps 8 and 9) is:

$$Y_{ON} = [(b+c+d)'aef + (a+g)'bc + (e+f)'d] \quad (12)$$

The corresponding TC for the read-once sub-functions (b+c+d)', ((b+c+d)'aef), (a+g)', (a+g)'bc, (e+f)' and ((e+f)'d) are determined as 3584, 504, 6144, 1920, 6144 and 3584 using equation (6) respectively, while the combined OR-ing of the above sub-functions contributes to a TC of 5150. Hence, in this case, Y_{ON} has a total TC of 27030, computer as per step 12 of the algorithm. For this case, the percentage reduction in switching activity, ‘α’ and actual total power consumption has been 30.27% and 58.26% respectively.

8. SIMULATION MECHANISM AND RESULTS

A significant number of non-regenerative digital circuits (incorporating BROFs) have been considered for optimization studies to validate our proposition and some of them have been mentioned in Table 2.

Table 2. Logic function description
(F_I^J; I – Function identity, J – Order of the function in terms of input variables)

Function ID	Compressed Function description
F ₁ ⁹	(11xxxxxx)(xxxxx11)(xxxxxxx1) - PoDS
F ₂ ⁶	(001xxx)(xxx001) – PoDS
F ₃ ⁸	(11xxxxxx)(xx00xxxx (xxxxxx00) – PoDS
F ₄ ⁵	(000xx)(xxx11) – SoDP and PoDS
F ₅ ⁶	(1x00xx)(x100xx)(xxxx00) – PoDS
F ₆ ¹¹	(000xxxxxxxx)(xxxx000xxxx)(xxxxxxx1x11) - PoDS
F ₇ ¹⁴	(11xxxxxxxxxxxxxxxx)(xxxxxxxx00xxxx)(xxxxxxxxxxx000) - PoDS
F ₈ ⁸	(0001xxxx)(xxxx0011) – PoDS
F ₉ ⁶	(001xxx)(xxx11x)(xxxxx1) – PoDS
F ₁₀ ⁴	(00xx) (xx11)' – SoDP and PoDS
F ₁₁ ¹²	(11xxxxxxxxxx)(xx00xxxxxxxx)(xxxxxx00xxxx)(xxxxxxx00xx)(xxxxxxxxxxx00) - PoDS
F ₁₂ ¹²	(001xxxxxxxxxx)(xxx001xxxxxxxx)(xxxxxx001xxx)(xxxxxxxxxxx001) - PoDS
F ₁₃ ⁹	(0xxx01xxxx)+(xxxxxx0x0) – SoDP
F ₁₄ ¹¹	(000xxxxxxxx)+(xxx000xxxx)+(xxxxxxxx001) - SoDP
F ₁₅ ⁸	(0x0x0xxx)+(xxxxx001) – SoDP
F ₁₆ ¹⁴	(0000xxxxxxxx)+(xxxx0000xxxx)+(xxxxxxxxxxx0001) - SoDP
F ₁₇ ⁹	(00000xxxx)+(xxxxx00x1) – SoDP
F ₁₈ ⁶	(0xxxx)+(x1xxxx)+(xx00x1) – SoDP
F ₁₉ ¹⁰	(00xxxxxxxx)+(xx00xxxxxxxx)+(xxxx00xxxx)+(xxxxxxx11xx)+(xxxxxxxx11) - SoDP
F ₂₀ ¹⁴	(0xxx01xxxxxxxx)+(xxxxxx0x0xxxx)+(xxxxxxxxxxx00) – SoDP
F ₂₁ ⁷	(010xxxx)+(x01xxx0) – SoDP
F ₂₂ ⁴	(000x)+(0x10) – SoDP
F ₂₃ ⁸	(xxx000xx)+(xxxx1x00) – SoDP
F ₂₄ ⁴	(0x01)+(x010) – SoDP
F ₂₅ ⁴	(0x0x)+(00xx)+(x010) – SoDP
F ₂₆ ⁴	(010x)+(1xx1)+(x010) – SoDP
F ₂₇ ⁵	(100xx)+(x0x00)+(0100x) – SoDP
F ₂₈ ⁵	(x00xx)+(0xx0x)+(x1100)
F ₂₉ ⁵	f1=(x00x0)+(111xx) – SoDP f2=(x00x0)+(0x10x) - SoDP f3=(01x0x)+(x1x11) - SoDP
F ₃₀ ⁶	(000xxx)+(x1x000)+(10111x)+(11100x)+(1x1011) - SoDP

The estimation of total power consumption is on the basis of the well-established tagged probabilistic simulation scheme (Ding et al., 1998), with a static input signal probability of 0.5. The simulation results obtained for an input frequency of 100MHz with a supply of 3.3V for a 0.35 micron CMOS process at an ambient temperature of 27° C, using Mentor Graphics tools on a Linux platform have been listed in Tables 3 and 4. Industry standard BSIM3 device models (The MOSIS Service, 2006) have been used for transistor descriptions.

Table 3. Physical power consumption comparison profile (in nW)

Function ID	Stacked CMOS with LCTs			Static CMOS with LCTs		
	NAND logic	NOR logic	Hybrid logic	NAND logic	NOR logic	Hybrid logic
F ₁ ⁹	6.4634	10.3448	6.1499	6.7561	9.6436	6.6522
F ₂ ⁶	15.7155	13.0980	11.4689	14.1700	12.3335	10.3942
F ₃ ⁸	12.0940	8.4327	6.6511	11.1592	7.9283	6.4153
F ₄ ⁵	7.1840	5.6122	4.8278	7.1635	5.2220	5.2777
F ₅ ⁶	12.0731	6.9808	6.8555	10.9503	6.5543	6.5817
F ₆ ¹¹	14.9084	15.5296	10.7309	14.0694	14.5604	10.4508
F ₇ ¹⁴	12.7472	10.2446	8.3748	11.7292	9.6240	8.2525
F ₈ ⁸	19.7953	19.7932	15.0524	17.9344	18.5937	14.2014
F ₉ ⁶	12.7449	14.8008	10.3060	11.6196	13.9099	9.4979
F ₁₀ ⁴	7.5946	8.4655	3.7247	6.8174	7.9228	3.5306
F ₁₁ ¹²	8.1984	21.3019	13.6331	16.4564	20.1128	12.9824
F ₁₂ ¹²	14.3379	12.9733	22.6368	30.2476	27.7453	20.5759
F ₁₃ ⁹	12.1800	9.8758	9.7410	10.9164	9.2969	8.7872
F ₁₄ ¹¹	15.3210	13.8684	8.5929	13.8328	13.0154	7.2029
F ₁₅ ⁸	9.3206	8.7253	8.7765	8.5347	8.1826	8.3032
F ₁₆ ¹⁴	11.7779	19.1791	17.6794	23.2007	17.9775	16.5564
F ₁₇ ⁹	13.4886	12.3074	7.8800	12.1577	11.5318	7.3734
F ₁₈ ⁶	12.1793	10.2029	8.0298	10.9082	9.6602	7.7581
F ₁₉ ¹⁰	15.4724	23.4498	14.9533	15.1340	22.0628	14.9469
F ₂₀ ¹⁴	16.2599	14.7588	12.8124	14.6725	13.9115	11.7486
F ₂₁ ⁷	13.6878	11.7968	9.7618	12.6784	11.0600	8.8076
F ₂₂ ⁴	13.7087	13.1396	12.6592	12.8873	12.3665	12.1185
F ₂₃ ⁸	15.0867	11.7294	12.6592	14.5691	11.0342	12.1185
F ₂₄ ⁴	13.6878	11.7968	9.7618	12.6784	11.0600	8.8076
F ₂₅ ⁴	12.2252	13.3905	12.5085	11.8982	12.6208	12.0511
F ₂₆ ⁴	15.6309	17.8180	12.0297	14.8838	15.9964	10.9179
F ₂₇ ⁵	21.4807	20.3456	20.7647	21.1837	19.1078	19.5735
F ₂₈ ⁵	14.4931	18.3155	13.0526	14.1963	17.2355	12.5536
F ₂₉ ⁵	8.0960	10.4363	12.0295	16.5957	21.2753	12.3468
F ₃₀ ⁶	16.0123	20.5043	15.3431	15.2713	19.5333	14.7428

Fig. 3 shows a graphical comparison of EDP profile for the reduced functions, based on the different synthesis procedures and implemented using static CMOS style with LCTs

introduced. Fig. 4 depicts a visual comparison of EDP metric for the simplified functions based on different synthesis methods and implemented using stacked CMOS style with LCTs introduced. Although in practice, NOR logic is not preferred over NAND logic for effecting better optimization results, the simulation results obtained indicate a slight reverse trend. The reason for this is attributable to the nature of literal description of the BROFs (sub-functions of the given logic function) and choice of a constrained atomic operator set. From the simulation results mentioned in Tables 3 and 4, the best minimum value for the design metric corresponding to NAND/NOR logic has been compared with that of the result obtained by hybrid logic for each case and then averaged to compute the mean savings in power and EDP.

Table 4. Transistor level EDP metric comparison profile (in 10^{-27} Js)

Function ID	Stacked CMOS with LCTs			Static CMOS with LCTs		
	NAND logic	NOR logic	Hybrid logic	NAND logic	NOR logic	Hybrid logic
F ₁ ⁹	3.189	1.192	1.008	1.826	0.897	0.688
F ₂ ⁶	18.161	7.182	7.076	10.541	5.745	5.239
F ₃ ⁸	6.139	6.035	1.215	3.254	3.247	1.006
F ₄ ⁵	14.181	4.465	3.624	7.021	2.828	2.717
F ₅ ⁶	1.869	1.061	1.021	0.936	0.532	0.545
F ₆ ¹¹	22.044	9.029	5.356	10.649	5.329	3.248
F ₇ ¹⁴	10.097	3.042	2.356	4.971	1.758	1.525
F ₈ ⁸	27.797	22.873	8.580	13.418	16.780	8.529
F ₉ ⁶	7.954	3.701	2.499	3.141	1.914	0.869
F ₁₀ ⁴	7.072	5.032	2.045	3.509	2.838	1.396
F ₁₁ ¹²	10.842	27.927	6.969	11.065	12.872	3.363
F ₁₂ ¹²	28.707	16.565	13.772	36.933	24.515	10.815
F ₁₃ ⁹	10.031	11.626	4.524	6.141	7.868	3.163
F ₁₄ ¹¹	27.716	17.865	4.392	13.149	11.257	3.409
F ₁₅ ⁸	10.081	7.305	6.082	4.994	4.361	4.076
F ₁₆ ¹⁴	27.306	36.789	18.034	20.938	20.486	11.405
F ₁₇ ⁹	26.061	18.771	11.158	12.157	10.627	6.377
F ₁₈ ⁶	3.344	2.075	0.659	1.671	1.639	1.005
F ₁₉ ¹⁰	15.164	43.372	12.519	9.445	22.283	8.074
F ₂₀ ¹⁴	5.053	8.192	1.004	2.318	3.408	0.719
F ₂₁ ⁷	7.854	9.239	1.641	3.947	4.530	0.982
F ₂₂ ⁴	9.788	8.621	7.801	4.874	4.908	4.218
F ₂₃ ⁸	8.829	7.983	7.801	4.503	4.806	4.218
F ₂₄ ⁴	8.011	8.523	2.106	4.083	4.817	1.305
F ₂₅ ⁴	8.708	7.939	2.533	4.835	4.851	1.672
F ₂₆ ⁴	18.776	28.965	6.721	9.801	14.059	3.704
F ₂₇ ⁵	16.446	26.673	6.628	8.846	13.005	3.411
F ₂₈ ⁵	7.409	9.894	1.811	3.692	5.698	1.245
F ₂₉ ⁵	6.127	11.506	4.841	7.119	12.288	2.844
F ₃₀ ⁶	36.510	50.219	25.929	16.836	28.598	15.338

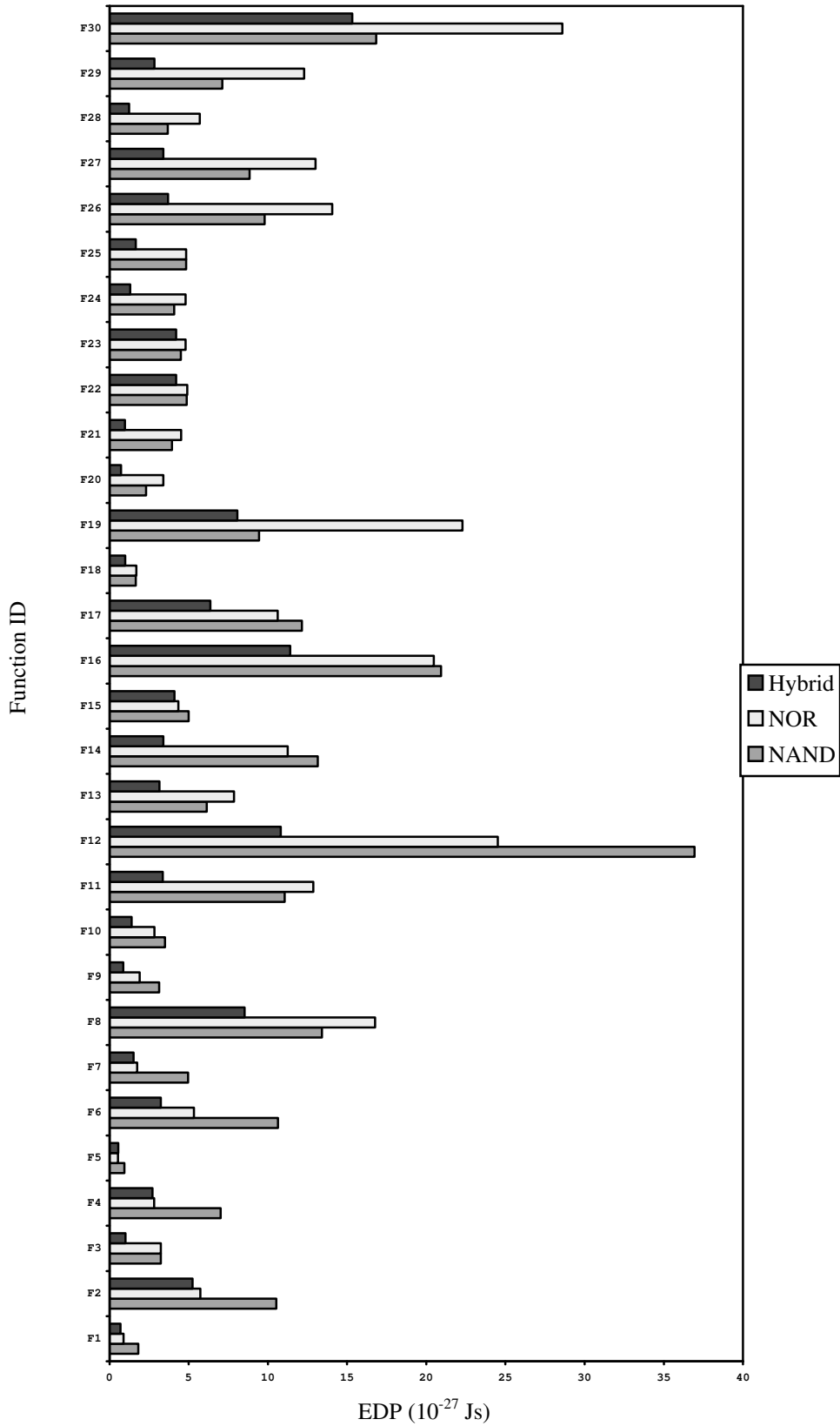


Figure 3. EDP comparison profile based on static CMOS logic with LCTs

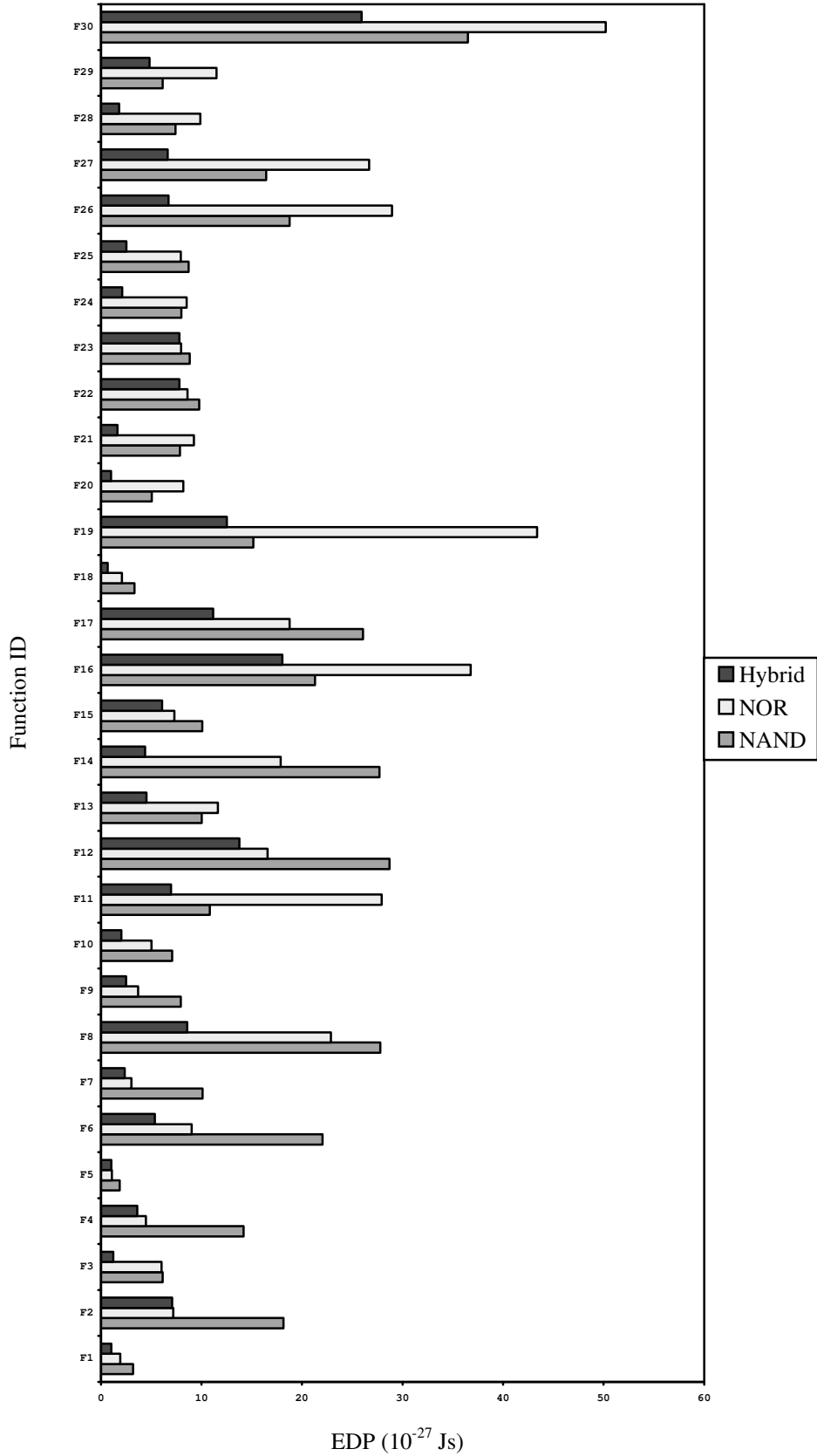


Figure 4. EDP comparison profile based on stacked CMOS logic with LCTs

9. CONCLUSIONS

Energy Delay Product has emerged as an important and useful performance metric to determine the quality or goodness of any integrated circuit design (Horowitz et al., 1994 and Hodges et al., 2003). A way to improve this metric is to avoid wasting energy – avoid causing node transitions that are not needed. This paper first discusses a novel synthesis scheme for lesser-associated transition count and switching activity (technology-independent optimization cycle), while preserving the desired logic functionality for Boolean read-once functions. Secondly this paper presents the validation of the proposition by mapping the gate level netlist information onto standard device models (technology-dependent optimization phase), based on two design styles, namely static and stacked CMOS logic. Variable re-ordering for minimizing the delay component has also been taken into consideration.

Leakage control transistors have also been embedded between the appropriate pull-up and pull-down network nodes to enable optimization in average power consumption by keeping the leakage component to a minimum. Since dynamic power accounts for 80% of the total power consumption in the sub-micron range and leakage power contributes to only around 10%-15%, the use of LCTs will have a profound impact on reducing the average power dissipation in the sub-nanometer domain, where leakage power eclipses dynamic power. Hence, the proposal would prove to be beneficial for ultra deep submicron technology nodes such as 90nm or below.

From the simulation results, we find that the savings in EDP and power consumption are to the tune of 42.36% and 14.35% respectively for the proposed implementation using static CMOS logic with leakage control transistors, in comparison with those of the best results obtained by conventional methods. For realizations with stacked CMOS logic along with leakage control transistors, the corresponding minimization in energy-delay product and power dissipation are 42.19% and 4.49% respectively, based on a similar comparison. The delay improvement has been around 15.57% for static CMOS and 20.72% for stacked CMOS implementation styles, over traditional approaches.

The proposed synthesis method would be suitable for problems with large number of variables as well. The essential prime cubes are obtained in the first stage using standard logic synthesis tools. The irredundant prime implicants (implicates) then undergo logic-preserving transformations, as per the proposed algorithm, suitable for effecting designs with lower activity rates. It should be understood in this context that off-the-shelf synthesis tools may at the best yield a minimized solution, but does not guarantee power efficient solutions. The low power methodology described in this work would be useful for ensuring optimality of a wide variety of logic functions (expressed in sum-of-disjoint products or product-of-disjoint sums form) and would carry significance even from a pedagogical point of view. However, for certain types of read-once function, namely positive Achilles' heel function and pure horn Achilles' heel function; the proposed technique might not be suitable for effecting optimization with respect to the Boolean equations underpinning it. This is owing to their basic definition and inherent property. Hence this appears to be a problem, for which a low power driven strategy could indeed specifically be formulated at the circuit level.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable comments which helped in enhancing the quality of the manuscript. This work was performed when the authors were affiliated with the School of Electrical Sciences, Vellore Institute of Technology (University and IET, UK Accredited), Vellore, TN, India.

REFERENCES

- Alain Guyot and Selim Abou-Samra. (1998) "Low power CMOS digital design", *Proc. of International Conference on Microelectronics*, 6-13.
- Brayton, R.K., Hachtel, G.D., McMullen, C.T. and Sangiovanni-Vincentelli, A.L., (1984) *Logic Minimization Algorithms for VLSI Synthesis*, Kluwer Academic, Boston.
- Carlson, B.S. and Suh Juch Lee. (1995) "Delay optimization of digital CMOS VLSI circuits by Transistor Reordering", *IEEE Trans. on CAD of Integrated Circuits and Systems*, **14**(10), 1183-1192.

- Chandrakasan, A.P. and Broderon, R.W. (1995) "Minimizing power consumption in digital CMOS circuits", *Proceedings of the IEEE*, **83**(4), 498-523.
- Chih –Shun Ding, Chi Ying Tsui and Massoud Pedram. (1998) "Gate-level power estimation using tagged probabilistic simulation", *IEEE Transactions on CAD of Integrated Circuits and Systems*, **17**(11), 1099-1107.
- Coudert, O. (1994) "Two-level logic minimization: an overview", *Integration – The VLSI Journal*, **17**(2), 97-140.
- Hodges D.A., Resve Saleh and Jackson, H.G., (2003) *Analysis and Design of Digital Integrated Circuits*, McGraw-Hill, NY.
- Hong, S.J., Cain, R.G. and Ostapko, D.L., (1974) "MINI: A heuristic approach for logic minimization", *IBM Journal of Research and Development*, **18**(5), 443-458.
- Horowitz, M., Indermaur, T. and R.Gonzalez, R. (1994) "Low-power digital design", *Proc. IEEE Symposium on Low Power Electronics*, 8-11.
- Martin C.Golumbic, Aviad Mintz and Udi Rotics. (2001) "Factoring and Recognition of Read-Once Functions using Cographs and Normality", *Proc. of ACM/IEEE Design Automation Conference*, 109-114.
- McCluskey, E.J., (1956) "Minimization of Boolean Functions", *Bell Systems Technical Journal*, **35**(5), 1417-1444.
- McGeer, P.C., Sanghavi, J.V., Brayton, R.K. and Sangiovanni-Vincentelli, A.L., (1993) "ESPRESSO-SIGNATURE: a new exact minimizer for logic functions", *IEEE Transactions on VLSI Systems*, **1**(4), 432-440.
- Michael Gallant and Dhamin Al-Khalili, (1999) "Synthesis of low power CMOS circuits using hybrid topologies", *Integration – The VLSI Journal*, **27**(2), 143-163.
- Narender Hanchate and Nagarajan Ranganathan, (2004) "LECTOR: A technique for leakage reduction in CMOS circuits", *IEEE Transactions on VLSI Systems*, **12**(2), 196-205.
- Nguyen, K., Perkowski, M. and Goldstein, N. (1987) "Palmini-Fats Boolean minimizer for Personal Computers", *Proc. of ACM/IEEE Design Automation Conference*, 615-621.
- Rudell, R. and Sangiovanni-Vincentelli, A.L., (1987) "Multiple-valued minimization for PLA optimization", *IEEE Transactions on Computer Aided Design*, **6**(5), 727-750.
- Srinivas Devadas, Abhijit Ghosh and Kurt Kuetzer., (1994) *Logic Synthesis*, McGraw-Hill Series on Computer Engineering, USA.
- Svoboda, A. and White, D.E., (1979) *Advanced Logical Circuit Design Techniques*, Garland Press, New York.
- The MOSIS Service <http://www.mosis.org/Technical/Testdata/tsmc-035-prm.html> (accessed June 10, 2006).
- Woo Jin Kim and Yong Bin Kim. (2003) "Automating wave-pipelined circuit design", *IEEE Design and Test of Computers*, **20**(6), 51-58.
- Zvi Kohavi., (1999) *Switching and Finite Automata Theory*, Tata McGraw Hill, USA.

Received: August 9th 2006

Accepted in final format: May 20th 2007 after three revisions.

About the authors:

Padmanabhan *Balasubramanian* completed his B.E degree in Electronics and Communication Engineering discipline from University of Madras, TN, India in 1998 and his M.Tech in VLSI System from National Institute of Technology, Tiruchirappalli, TN, India in 2005. His research interests are in asynchronous design, logic synthesis for low power and timing optimization issues. He is currently working towards his PhD degree in the School of Computer Science at The University of Manchester, UK. He can be reached at padmanab@cs.man.ac.uk

Siddhagunta *Theja* completed her B.E in Electrical and Electronics Engineering from University of Madras, TN, India in 2004 and her M.Tech in VLSI Design from Vellore Institute of Technology (University and IET, UK Accredited), Vellore, TN, India in 2006. She is at present a Design Engineer with Sasken Communication Technologies Ltd., Bangalore, India and is currently working on LEC for the Sunburst project. She can be reached at sthejar@sasken.com