World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:1, No:8, 2007

Matrix Based Synthesis of EXOR dominated Combinational Logic for Low Power

Padmanabhan Balasubramanian, and C. Hari Narayanan

***Abstract*—** This paper discusses a new, systematic approach to the synthesis of a NP-hard class of non-regenerative Boolean networks, described by $F_{ON}[F_{OFF}]=\{m_i\}[\{M_i\}]$, where for every $m_j[M_j]\in\{m_i\}[\{M_i\}]$, there exists another $m_k[M_k]\in\{m_i\}[\{M_i\}]$, such that their Hamming distance $HD(m_j, m_k)=HD(Mj, M_k)=O(n)$, (where 'n' represents the number of distinct primary inputs). The method automatically ensures exact minimization for certain important self-dual functions with 2^{n-1} points in its one-set. The elements meant for grouping are determined from a newly proposed weighted incidence matrix. Then the binary value corresponding to the candidate pair is correlated with the proposed binary value matrix to enable direct synthesis. We recommend algebraic factorization operations as a post processing step to enable reduction in literal count. The algorithm can be implemented in any high level language and achieves best cost optimization for the problem dealt with, irrespective of the number of inputs. For other cases, the method is iterated to subsequently reduce it to a problem of O(n-1), O(n-2),…. and then solved. In addition, it leads to optimal results for problems exhibiting higher degree of adjacency, with a different interpretation of the heuristic, and the results are comparable with other methods.

In terms of literal cost, at the technology independent stage, the circuits synthesized using our algorithm enabled net savings over AOI (AND-OR-Invert) logic, AND-EXOR logic (EXOR Sum-of-Products or ESOP forms) and AND-OR-EXOR logic by 45.57%, 41.78% and 41.78% respectively for the various problems.

Circuit level simulations were performed for a wide variety of case studies at 3.3V and 2.5V supply to validate the performance of the proposed method and the quality of the resulting synthesized circuits at two different voltage corners. Power estimation was carried out for a 0.35micron TSMC CMOS process technology. In comparison with AOI logic, the proposed method enabled mean savings in power by 42.46%. With respect to AND-EXOR logic, the proposed method yielded power savings to the tune of 31.88%, while in comparison with AND-OR-EXOR level networks; average power savings of 33.23% was obtained.

***Keywords*—**AOI logic, ESOP, AND-OR-EXOR, Incidence matrix, Hamming distance.

I. Introduction

LOGIC synthesis has matured as a field and is used in every major digital IC design worldwide [4]. Despite a wealth of research results and pioneering commercial tools, some significant problems remain open owing to its inherent computational complexity [2]. Logic synthesis forms an important part of the design cycle for a digital circuit/system. Its actual significance should also be understood in the context of the increasingly important requirement to minimize power dissipation. The low power design challenge is one that requires abstraction, modeling and optimization at all levels of design hierarchy. The considerations range from the technology being used for the implementation, circuit and logic topologies, digital architectures and even the algorithms being implemented [1]. This articulates the fact that the power component should be considered early during the logic synthesis phase as well. Gate-level optimization may achieve power savings: in some specific cases more than 50% reduction in power, without loss of performance, may be achieved [4]. Also gate-level optimizations are relatively low cost in terms of design-effort compared with other techniques.

This paper presents a novel and versatile synthesis method incorporating available gate library types. It is primarily aimed at combinatorial logic networks, whose ON/OFF set exhibits maximal and complementary pair-wise disjointness. In other words, the function set contains canonical terms, which differ based on their Hamming distance, (HD), which is O(n). The proposed technique guarantees best results for the above problem definition, where the optimality of the solution obtained is quantitatively evaluated in terms of total power consumption (P) of the circuit realized, and number of literals (N_L) required for its implementation. Of course, a comparison based on N_L is suitable not only for standard cell-based designs but also for FPGA technology targets. This is because literals tend to correlate quite well with circuit area in custom IC designs and they determine the number of look-up tables (LUTs) needed for realization of the functionality with FPGA.

The technique has its roots in the rudiments of graph and network theory. It enables best minimum solutions even for logic networks composed of terms exhibiting strong or complete adjacency, using a variation of the proposed heuristic, as will be seen in section 5. Although our analysis for some problem cases with function sets exhibiting adjacency as well as non-adjacency between their elements have yielded satisfactory experimental results, still an exhaustive analysis is deemed necessary to gain a complete insight about the effectiveness of our proposition for such function classes, which promises scope for further work in this direction. This phenomenon is also due to the NP-hard enumeration of such possible functions. The strategy considers the issue of output phase optimization as well.

Padmanabhan Balasubramanian is with the School of Computer Science, The University of Manchester, Manchester, MAN M13 9PL UK (phone: +44-161-275 6294; e-mail: spbalan04@gmail.com, padmanab@cs.man.ac.uk).

C. Hari Narayanan is with the School of Electrical Sciences, Vellore Institute of Technology (University and IET, UK Accredited), Vellore, Vellore – 632 014 India (e-mail: harinara_21@yahoo.com).

The rest of this paper is organized as follows. Section 2 provides concise background information pertaining to some traditional synthesis approaches available in literature, such as AND-OR-Invert (AOI) logic and also a briefing about some of the important classes in the large AND-EXOR logic family. We also provide a reference to AND-OR-EXOR three level networks. In Section 3, the inherent nature and fundamental properties governing Boolean functions exhibiting only maximal and exact bit-wise complementary support are mentioned anew. Section 4 discusses some essential graph theoretic principles. Section 5 describes the complete graph structure and weighted incidence matrix formulation with an illustration. Minimization heuristics for all problem categories are also described. Besides, the mode of direct synthesis by correlation of the binary data with another proposed binary value matrix is also dealt with in this section for minterms/maxterms as well. Section 6 briefly highlights the simulation mechanism, and gives details regarding the cases considered for simulation studies and subsequent validation. Comparison between the different synthesis procedures in terms of power and N_L is also graphically illustrated in this section. Finally, we conclude in Section 7.

II. Conventional synthesis methods

Conventional logic design is usually based on AND-OR logic and OR-AND logic. We have used AND-OR-Invert logic in common with the usual library descriptions in the technology domain; this will be referred to as AOI form in this paper. We have adopted single rail circuits for our implementation. Although double-rail inputs are available in the general case for designs targeting programmable logic devices, for standard cell based IC designs, the circuit inputs are generally single-rail. This is because single-rail circuits tend to have fewer interconnections and thereby less hardware area overhead.

EXOR based designs have certain well-known advantages over the above classical realization methods. Firstly, they pave way for a more concise expression for many basic arithmetic functions, thereby reducing the complexity of the networks. Secondly, many practical digital circuits used in the fields of coding theory, linear system, telecommunication and arithmetic coding contain basic functionality which are inherently mod-2 sum [6]. Finally, circuits containing EXOR gate types have excellent design-for-test properties [7] [8]. Such is their significance that even some earlier FPGA styles had incorporated 2-input EXOR gate in their basic granularity blocks, for example Cli 6006 from Concurrent Logic Inc.

Various classes exist in AND-EXOR expressions involving only AND and EXOR gate types [6]. This is because any arbitrary logic function can be purely realized using only AND and EXOR logic gates. For example, the RM, GRM and PSDKRO form extensively rely upon such gates for implementation. Slight modification of the Shannon expansion to suit Galois field of order 2 (1) are made for the AND-EXOR logic to derive Davio expansions (2), (3) and given by,

$$F = [(x \cdot F_x) \oplus (x' \cdot F_x')] \quad (1)$$

$$F = [F_x' \oplus \{x \cdot (F_x \oplus F_x')\}] \quad (2)$$

$$F = [F_x \oplus \{x' \cdot (F_x \oplus F_x')\}] \quad (3)$$

In (1), (2) and (3), the symbols '$\cdot$' and '$\oplus$' stand for AND and EXOR operators respectively, while '+' and '$\odot$' would imply logical OR and EXNOR operators.

Recursive application of the above tree expansions results in various RM trees [6]. If only the positive Davio expansion (2) is used repeatedly for variable expansion with some fixed order of expansion of variables, a compact RM tree is generated. A GRM tree is created when a choice exists between positive Davio expansion (2) and negative Davio expansion (3) for each variable. If equations (1), (2) and (3) are used along with the choice of equations (2) and (3) in each sub-tree, the PSDKRO structure is generated. Importantly, in all these structures only two kinds of gates (AND and EXOR) are used for circuit realizations.

Amongst the AND-EXOR expressions, Exclusive-OR-sum-of-products expression (ESOP) is the most general logical expansion of interest [6]. ESOP is basically a logical expression that combines arbitrary products terms by EXOR-ing. For an n-variable function, there are at most 3^{tn} different ESOPs with t products. Although, no efficient minimization algorithm is known for more than five variables and it is still open [6], heuristics have been formulated which obtain near minimum or exact minimum ESOP forms [9] [10] [11] [12].

An AND-OR-EXOR network, where the output EXOR gate has only two inputs, is one of the simplest three-level architecture [13] [14]. In other words, here, the logic function is represented as an EXOR sum of two sum-of-products expressions. This network realizes an EXOR of two sum-of-products expressions, where the products associated with a sum term are mutually disjoint. This network is suitable for implementing arithmetic functions and also for realization of many random functions. Since this configuration is found in the basic macro cell architecture of CPLDs, this form gained significance. Also the upper bound on the number of products in this representation is shown to be $5(2^{n-4})$, which is 37.5% smaller than the upper bound on the number of products in the conventional sum-of-product expression of the logic function (which is 2^{n-1}) and 16.67% smaller than the maximum bound on the number of product terms in the AND-EXOR expansion, in which both complemented and uncomplemented forms of an input variable are used ($3(2^{n-3})$) [17].

III. Function definition and properties

We will now highlight the basic definition of NP-class of logic functions and list their important properties anew.

A. Function definition

Let Z be a logic function with its support and ON-set defined as,

$$s[Z] = \{x_{n-1}, x_{n-2}, \ldots, x_0\} \quad (4)$$

$$Z_{ON} = \{m_i\};\ 0 \le i \le (p-1), \text{ where } 0 \le p \le 2^{|s[Z]|} \quad (5)$$

where, Z_{ON} stands for the ON-set of the function Z and $\{m_i\}$ refers to the set of all minterms. Now, for every m_j, $m_k \in \{m_i\}$, that exists, the Boolean distance between the two binary tuples is given by HD (m_j, m_k) and is O(n).

1) Property 1

For the binary 2-tuple (m_j, m_k), whose HD is O(n), it naturally follows that ($m_j \cap m_k$) = { } and the converse is also true.

2) Property 2

The upper bound for the number of exact bit-wise complementary pairs in the function set would be $2^{n-1}/2^{|s[Z]|-1}$.

3) Property 3

For the upper bound of 2^{n-1} disjoint binary 2-tuples, the bit-wise complementary pair is obtained by grouping an element in a finite set comprising 2^{n-1} elements with the absolute value of the element being 'j', such that j = 0,1,….,$2^{(n-1)}$-1; with an unique element of another finite set of a similar cardinality, whose absoluteness is described by 'k', such that k = 2^n-m; where m = 1,2,….,(n+1).

The above definition and formulation of properties also hold good for the case of functions specified in terms of their OFF-set (i.e. in terms of their maxterms).

IV. Graph Theory Fundamentals

A graph 'G' is a pair, G = (V, E), consisting of a finite set $V \neq \phi$ and a set E of two-element subsets of V. In graph theory, infinite graphs are also studied, however here; we restrict ourselves to the finite case. The elements of V are called vertices. An element, e = {a, b} of E is called an edge with end vertices 'a' and 'b'. We say that 'a' and 'b' are incident with 'e' and that 'a' and 'b' are adjacent or neighbours of each other, and write e = ab [5]. Further details regarding the pictorial descriptions and properties of all graphs and other network terminologies can be found in [5].

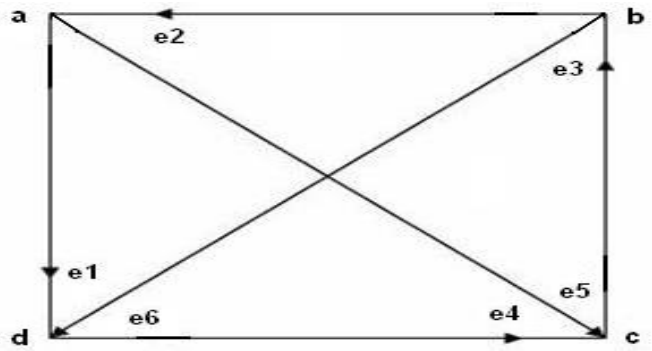
V. Weighted Incidence matrix, Binary value matrices and Synthesis mechanism

In our case, the problem under consideration is represented in a novel way on the lines of a complete graph specification. The complete graph, K_n is a network with |V|=n and |E|=(n(n-1))/2. In other words, the empty graph, K_n' is a graph with |K_n'|=n and |E(K_n')|=0, where K_n' is the complementary version of K_n [5]. The decimal equivalent of each binary cube shall correspond to a unique vertex label of the complete graph. So, the total number of minterms [maxterms] in the ON-set [OFF-set) of a completely specified logic function shall account for the number of vertices in the graph. As far as incompletely specified logic functions are concerned, whose DC-set ≠ { }, the inclusion of its elements in the ON-set or OFF-set is dictated by the optimality of the best minimal solution that could be predicted and hence all the DC cubes are initially considered as candidates.

The algorithm for such a directed graph (digraph) can be implemented by means of any high-level language and run on a computer with even an incidence list representation.

A. Weighted Incidence Matrix specification

A directed multigraph G with a non-zero finite vertex set can in general be represented in matrix form by an Incidence matrix [5]. For our problem specification, we opt for a slightly modified form of the latter, designated as Weighted Incidence Matrix (WIM). The order of an incidence matrix would be 'n×(n(n-1)/2)'. The added feature of a weighted incidence matrix over the conventional one would be that each matrix element representing the presence of a directed edge from one vertex to another, is also multiplied by the decimal equivalent of the corresponding Boolean distance between the two vertices or binary cubes (to be read as an entry corresponding to a row 'i' and a column 'j'), HD (i, j). In other words, every edge of this strongly connected graph will be associated with a weight, which is the Boolean distance between its head and tail vertices. Hence the proposed binary network is modeled on the basis of a complete graph structure. The typical structure of a four terminal logic network is shown in Fig. 1.



$$HD(a,b) = W^{e(a,b)} = W^{e(b,a)}$$
$$HD(a,c) = W^{e(a,c)} = W^{e(c,a)}$$
$$HD(a,d) = W^{e(a,d)} = W^{e(d,a)}$$
$$HD(b,c) = W^{e(b,c)} = W^{e(c,b)}$$
$$HD(b,d) = W^{e(b,d)} = W^{e(d,b)}$$
$$HD(c,d) = W^{e(c,d)} = W^{e(d,c)}$$

Fig. 1 An example 4-terminal binary network

Here $W^{e(x,y)}$ represents the weight of an edge, 'e' with its head at vertex 'y' and tail at vertex 'x' and is equal to the decimal equivalent of the binary distance between those two vertices. The directions of the edges, e1 to e6 are arbitrary. The weighted incidence matrix of Fig. 1 would be as follows.

	e1	e2	e3	e4	e5	e6
a	1×HD(a,d)	-1×HD(a,b)	0	0	1×HD(a,c)	0
b	0	1×HD(a,b)	-1×HD(b,c)	0	0	1×HD(b,d)
c	0	0	1×HD(b,c)	-1×HD(c,d)	-1×HD(a,c)	0
d	-1×HD(a,d)	0	0	1×HD(c,d)	0	-1×HD(b,d)

From the above matrix specification, it can be inferred that

$$\sum_{i=1}^{n_c} m_{i1} = \sum_{i=1}^{n_c} m_{i2} = \sum_{i=1}^{n_c} m_{i3} = \sum_{i=1}^{n_c} m_{i4} = 0 \quad (6)$$

Here 'm_{ij}' indicates a matrix element with indices 'i' and 'j' representing the particular row and column. Also, the number of entries in each row of the above matrix would be (n-1).

1) Case 1: Exact grouping procedure for logic functions with actual complementary ON-set/OFF-set

Given a Boolean function, Z, whose ON-set, $Z_{ON} = \{m_i\}$, such that $|Z_{ON}|$ is even, then for every $m_j \in \{m_i\}$, there definitely exists another $m_k \in \{m_i\}$, where $m_j \cap m_k = \phi$, then $HD(m_j, m_k) = n$. Also, there does not exist any $m_l \in \{m_i\}$, such that $HD(m_j, m_l) = HD(m_k, m_l) = O(unity)$. It has been observed that the number of functions belonging to this category grows exponentially with 'n' and hence the proposed technique would carry much significance for higher values of 'n'. A digraph is then drawn with $|V(K_n)|=n$ and $|E(K_n)|=(n(n-1)/2)$. A weighted incidence matrix is then formed as shown above. For $\max|m_{ij}|=\max|m_{kj}|=HD[O(n)]$, group the graph vertices or binary cubes corresponding to the i^{th} and k^{th} rows. The above procedure is continued until all the row entries in the matrix are checked, so that none is left ungrouped with any of the other vertices of the complete graph. Then the binary information corresponding to the term-pair undergoes a literal matching with the value matrix mentioned in the next sub-section, so as to enable direct logic synthesis. The resulting factors are subjected to a weak factorization heuristic to yield the best and minimal irredundant solution. The above routine holds good for maxterm dependent functions also and takes the least polynomial time since single matrix iteration is sufficient to obtain terms, suitable for pairing.

The above procedure is also ideal for certain self-dual logic functions which contain exactly 2^{n-1} elements in its one-set. A self-dual function can be generally described as follows. Let Z be a completely specified self-dual Boolean function, so that $Z_{ON}=\{m_i\}$ and $Z_{OFF}=\{M_j\}$; such that $|Z_{ON}|=|Z_{OFF}|$ and $i \neq j$. Let $M_k \in \{M_j\}$. Then the equality relation based on the indices, $\{i\}$ **xor** $k \in \{j\}$, would hold well. If Z is an incompletely specified function, then the values for don't cares are assigned such that the above relation is satisfied and the cardinality of either of the main functional sets (ON/OFF-set) would be (n/2). The method illustrated is suitable for effectively synthesizing practical digital circuits of any order, which contain such functionality and examples include parity generator and checker circuits, with even number of input literals. For odd inputs, the problem is first reduced in dimension to O(n-1) and then subsequently solved.

2) Case 2: Exact two-level solution for function elements exhibiting complete adjacency

For a logic function, F, whose ON-set is $F_{ON} = \{m_a\}$, then for every $m_b \in \{m_a\}$, there exists atleast one $m_c \in \{m_a\}$, such that $m_b \cap m_c \neq \{\ \}$ or $HD(m_b, m_c)<O(n)$. Hence the minimum binary distance is unity and the maximum value is (n-1). Then, similarly, a directed graph with $|V(K_n)|=n$ and $|E(K_n)|=(n(n-1)/2)$ is sketched. A weighted incidence matrix is framed. For $|m_{ij}| = |m_{kj}| = HD[O(unity)]$, combine those vertices corresponding to the i^{th} and k^{th} rows. Another weighted incidence matrix is framed for the vertex pair entries resulting from the previous matrix. This would constitute the second iteration. The shared literal would have to be neglected with a don't care occupying that variable position. For the problem under consideration represented by this present matrix, the notion of adjacency for binary values would be governed by the following: 1&0, 0&1, 0&d, d&0, 1&d, d&1 are understood as non-adjacent (whose HD=1), whereas 0&0 and 1&1 are considered adjacent (whose HD=0). Here'd' refers to a don't care term. In the new matrix, the presence of a unity element is to be checked. If found, the above procedure is continued, otherwise all the canonical terms would have been already co-joined to result in a minimum expression in standard disjunctive normal form. The vertex combinations would imply the reduced products and they have to be logically summed to obtain the above form. It has been found that the results obtained by this method match those obtained by standard synthesis solutions such as Quine-McCluskey method or a standard two-level logic minimizer such as Espresso. The number of iterations in this case would depend upon the degree of grouping attained in the initial stages and as such this method covers the issue of output phase optimization and is also applicable for functions appearing in canonical product forms. An added advantage in this method is that it can be used to solve for both the normal phase as well as for the inverted phase of the function in parallel and then compare them based on the number of literals and realizable gates needed, in conformity with the directed acyclic graph specification of a logic circuit.

3) Case 3: NP-hard function enumeration problem

Given a Boolean function, Z, whose ON-set, $Z_{ON} = \{m_i\}$, such that $|Z_{ON}|$ is odd, then for every $m_j \in \{m_i\}$, there exists an element, $m_k \in \{m_i\}$, where $m_j \cap m_k = \phi$, then $HD(m_j, m_k) = n$. In addition, there also exists atleast one more element, $m_p \in \{m_i\}$, where $1 < HD[m_p, \{m_i\}] < O(n-1)$ holds good. To quantify the number of functions belonging to this class is by itself an NP-hard problem as there are numerous functions belonging to this category and is beyond the scope of this work. Among the functions that we have considered, belonging to this category, we have achieved good simplification, enabling power optimized solutions.

4) Case 4: Considering output phase optimization

Similar to conventional PLA type output phase optimization, for the problem type described in the previous cases, binary networks can be drawn for both the normal phase of the function (F) as well as for the complementary phase (F'). Then their corresponding minimum solutions can be compared in terms of the number of irredundant prime implicants (implicates) and/or literals to decide on the best choice of function polarity.

If a logic function is described in terms of its OFF-set (with or without don't cares), by default, it is translated into an ON-set problem to be solved and the solution is inverted. This is necessitated due to the presence of an extra implicant required for a minimal OFF-set solution compared with the latter. In turn, this is attributed to the nature of the binary value matrix specification for maxterms as is evident from (8) and (9).

To find a minimum solution for problem definitions associated with case 3, the technology independent heuristic would make a final choice based on the literal count of the resulting solutions corresponding to the normal and complementary function polarities. However, a comparison on the basis of realizable gates is also possible, if the circuit can be standardized to accommodate only gates available in a library. This seems to be a reasonable proposition as the directed acyclic graph representation for a combinatorial switching circuit would have a similar binary network.

B. Synthesis with Binary Value Matrices

The proposed synthesis technique, although primarily meant for effecting cost optimization by way of reducing the number of gates and literals needed for implementation, achieves the objective of minimizing the power cost of the Boolean network realized. Although power optimization is a consequence of the area-centric approach, it is a desirable outcome. The method proposed in this paper mainly paves way for efficient grouping and subsequent reduction of pair-wise; maximally disjoint binary 2-tuples. This is possible by a binary value matrix for a 2-tuple fully disjoint ON-set pair described by equations (7) and (8) as follows,

$$\begin{array}{c} \\ m_j \\ m_k \end{array} \begin{array}{c} \begin{array}{cccccc} x_{n-1} & x_{n-2} & \cdots\cdots\cdots & x_1 & x_0 \end{array} \\ \begin{bmatrix} 0 & 1 & \cdots\cdots\cdots & 1 & 1 \\ 1 & 0 & \cdots\cdots\cdots & 0 & 0 \end{bmatrix} \end{array}$$

$$= [\,(x_{n-1} \oplus x_{n-2}) \cdot \ldots\ldots\ldots\ldots\ldots \cdot (x_1 \odot x_0)\,] \quad (7)$$

$$= [\,(x_{n-1} \odot x_{n-2}) + \ldots\ldots\ldots\ldots\ldots + (x_1 \oplus x_0)\,]' \quad (8)$$

A binary value matrix for a 2-tuple disjoint OFF-set pair would be described by (9) and (10), as given below.

$$\begin{array}{c} \\ M_j \\ M_k \end{array} \begin{array}{c} \begin{array}{cccccc} x_{n-1} & x_{n-2} & \cdots\cdots\cdots & x_1 & x_0 \end{array} \\ \begin{bmatrix} 0 & 1 & \cdots\cdots\cdots & 1 & 1 \\ 1 & 0 & \cdots\cdots\cdots & 0 & 0 \end{bmatrix} \end{array}$$

$$= [\,(x_{n-1} \odot x_{n-2}) + \ldots\ldots\ldots\ldots\ldots + (x_0 \odot x_{n-1})\,] \quad (9)$$

$$= [\,(x_{n-1} \oplus x_{n-2}) \cdot \ldots\ldots\ldots\ldots\ldots \cdot (x_0 \oplus x_{n-1})\,]' \quad (10)$$

It is clear that equations (7) and (8) and similarly equations (9) and (10) are equivalent duals of each other. The rows in the above matrices correspond to the canonical binary 2-tuple pair and the columns represent the support of the function. Hence a basic binary value matrix is of order '2×n'. The intersection of a row index with a column index is assigned a binary 1(0), if the variable associated with the minterm (maxterm) is present in normal form and 0(1) otherwise.

For problems belonging to case 1, these matrices yield reduced, irredundant terms. For e.g. if $F=\sum m(6,9)$, then an ESOP minimization heuristic would yield a minimized expression of (a'bcd') ⊕ (ab'c'd); a two-level logic minimizer would obtain the reduced solution as (a'bcd') + (ab'c'd); whereas the binary value matrix would enable a minimized solution as (a ⊕ b)·(b ⊕ c)·(c ⊕ d), thus reducing the number of literals needed for realization by 25%. An AND-OR-EXOR network is not feasible for the above function in normal form.

In general, the terms hence obtained would be subject to weak factorization operations, so that the final solution would be the best solution possible. For case 2, however these matrices are not warranted. For case 3, these matrices are reasonably helpful in obtaining reduced forms. However binary value matrices of O(< n) would also be considered for synthesis purpose. Again we apply traditional algebraic factorization operations so as to eliminate redundancy in the final minimized expression. The concept of output phase optimization would prove to be useful for such problems, where the number of cubes in SOP would be exponential, whereas they would be linear in ESOP forms.

VI. Simulation Mechanism and Results

Let us consider a simple example benchmark, n168 [16], belonging to the NP-class of functions, to illustrate the significance of the proposed method.

The truth table specification for the above function is given by $[0,1,1,0,1,0,0,1,1,0,0,1,0,1,1,0]^T$. We now list the minimized expressions corresponding to AOI form, AND-EXOR form, AND-OR-EXOR form and proposed form. These are given by (11), (12), (13) and (14) respectively.

$$\begin{aligned} \text{AOI} = {} & (a'b'c'd) + (a'b'cd') + (a'bc'd') + (a'bcd) \\ & + (ab'c'd') + (ab'cd) + (abc'd) + (abcd') \end{aligned} \quad (11)$$

$$\text{AND-EXOR} = d' \oplus (a'b'c') \oplus (a'bc) \oplus (abc') \oplus (ab'c) \quad (12)$$

$$\text{AND-OR-EXOR} = d' \oplus (a'b'c' + abc') \oplus (a'bc + ab'c) \quad (13)$$

$$\text{Proposed form} = (a \oplus b) \odot (c \odot d) \quad (14)$$

From the above equations, we find that the literal costs for the four different forms in order are 32, 13, 13 and 4 respectively. This makes it clear that the proposed forms enables optimization in literal cost by 69.23% over the best of the remaining expressions. It is also clear that (14) would result in direct usage of library cells, found in a standard cell library. After translation of the above equations into a multi-level format by means of conversion to a directed acyclic graph specification with nodes exhibiting a fan-in of only 2, and with gate sharing, we find that the proposed form enables

savings in device count as well.

With respect to the number of transistors required for physical implementation of the expressions corresponding to (11), (12), (13) and (14), we find that they would be 146, 104, 92 and 36 respectively, for implementation with static CMOS logic style. Hence, our synthesis methodology effects savings in device count by 60.87% over the best of other methods.

Many non-regenerative logic functions, representing all possible problem cases (excepting case 2 as the solutions obtained by the proposed method are the same with respect to other methods) described above, were considered for simulation studies. Table 1 lists the description of the functionality. The MOS transistor descriptions corresponding to the synthesized gate level netlist of the different forms for each logic function were simulated as the back-end using Mentor Graphics ELDO for a 0.35 micron TSMC CMOS process technology. The estimation of power consumption of circuits based on a representative vector set, which includes input patterns with varying probabilities, is through tagged probabilistic simulation scheme [3], for two different supply voltages, viz. 3.3V and 2.5V. The practical results obtained for the target technology validate our proposition and arguments. The graphical comparison plots for the three different sources are depicted by Figures 2 and 3. The literal count comparison is depicted by Fig. 4.

TABLE I
LOGIC FUNCTION SPECIFICATION

Logic Function ID	Canonical ON-set specification of the Boolean function
$LF1^4$	{0,4,5,6,7,8,9,10,13,15}
$LF2^4$	{0,1,3,6,7,8,10,13}
$LF3^4$	{0,3,12,15}
$LF4^4$	{5,6,9,10}
$LF5^4$	{0,3,5,6,9,10,12,15}
$LF6^4$	{1,2,4,7,8,11,13,14}
$LF7^4$	{5,6,7,9,10,11}
$LF8^4$	{1,2,13,14}
$LF9^4$	{0,1,2,5,10,13,14,15}
$LF10^4$	{0,1,2,4,11,13,14,15}
$LF11^4$	{0,1,4,5,6,7,10,15}
$LF12^5$	{8,11,20,23}
$LF13^5$	{0,1,6,7,10,11,12,13,18,19,20,21,24,25,30,31}
$LF14^5$	{1,2,5,6,25,26,29,30}
$LF15^5$	{0,3,12,15,16,19,28,31}
$LF16^5$	{0,3,4,7,10,11,12,13,14,15,16,17,18,19,20,21, 22,23,24,27,28,31}
$LF17^5$	{1,2,5,6,16,18,25,26,29,30}
$LF18^5$	{5,10,21,26,28,29}
$LF19^5$	{8,9,10,11,21,22}
$LF20^5$	{1,2,5,6,16,18,25,26,29,30}
$LF21^5$	{1,2,9,10,13,14,17,18,20,21,29,30}
$LF22^5$	{4,5,10,11,19,20,21,23,26,27}
$LF23^5$	{0,4,9,11,12,14,17,19,20,22}
$LF24^6$	{17,18,21,22,41,42,45,46}
$LF25^6$	{7,8,55,56}
$LF26^6$	{3,4,11,12,51,52,59,60}
$LF27^6$	{3,4,11,12,16,24,32,40,51,52,59,60}
$LF28^6$	{0,4,8,12,13,14,17,18,45,46,49,50}
$LF29^6$	{0,9,18,27,36,45,54,63}
$LF30^6$	{17,18,29,30,52,53,54,55,56,59}

LFM^n: LF – Logic Function, M – Function ID, n – number of inputs

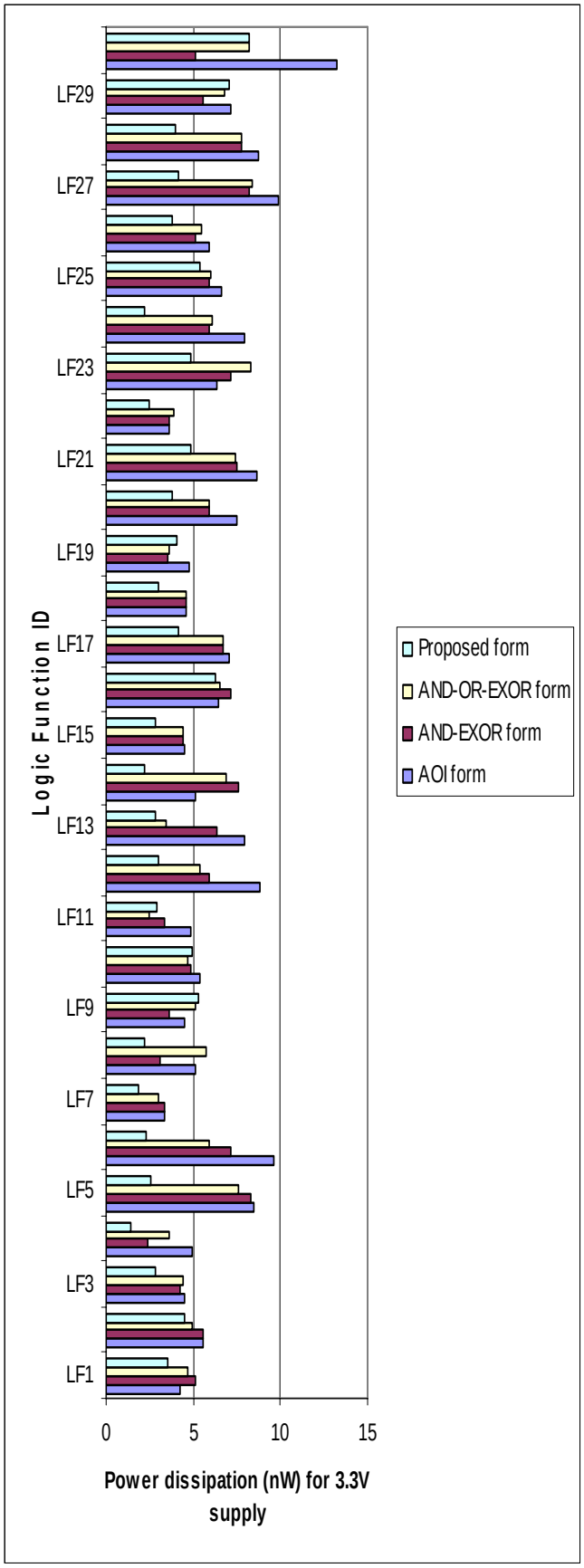


Fig. 2 Average power consumption at 3.3V supply

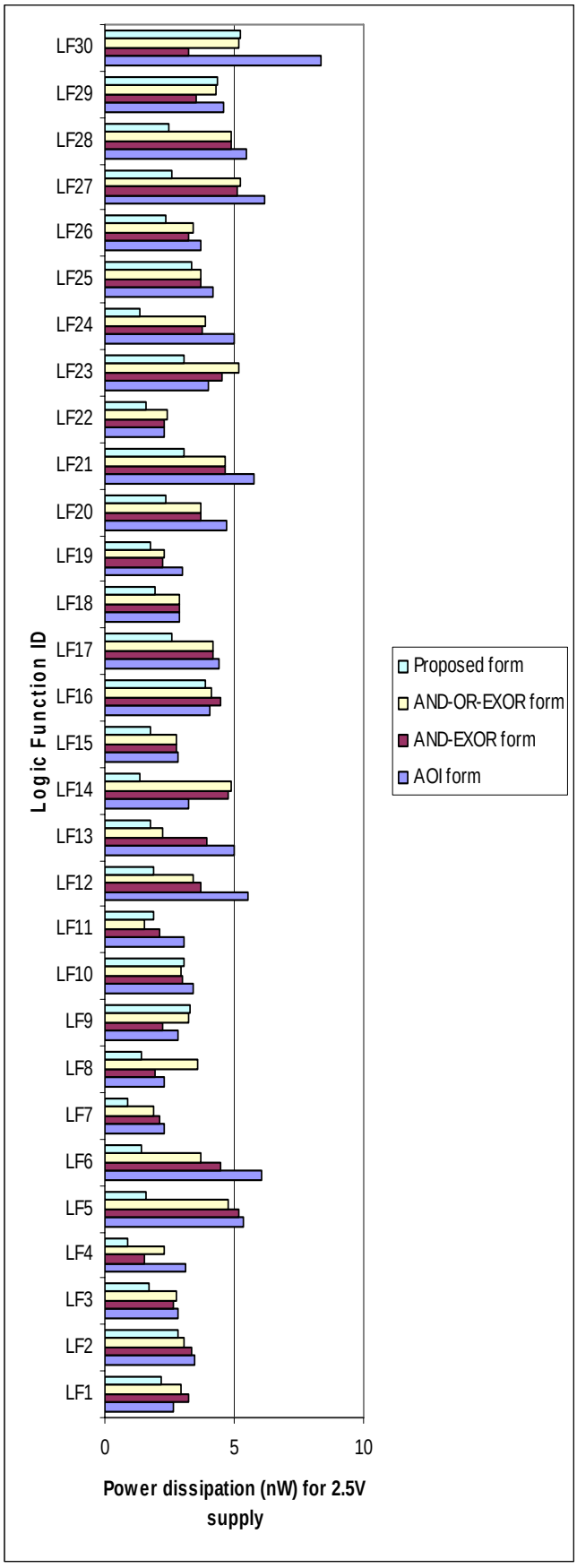


Fig. 3 Average power dissipation at 2.5V supply

VII. Conclusion

This paper has addressed an important issue of practical relevance, by means of introducing a novel approach to logic reduction particularly for a certain NP hard class of Boolean functions, based on graph theory, and suitable for easier implementation with a high-level language. Our approach is greatly simplified in comparison with those of existing methods by enabling a systematic reduction procedure. The other advantage is that it also minimizes functions not exhibiting auto-symmetry. This adds to the pedagogical value of this research work and is ideal for obtaining reduced solutions, even manually, for logic functions with fewer inputs, dominated by EXOR/EXNOR logic functionality.

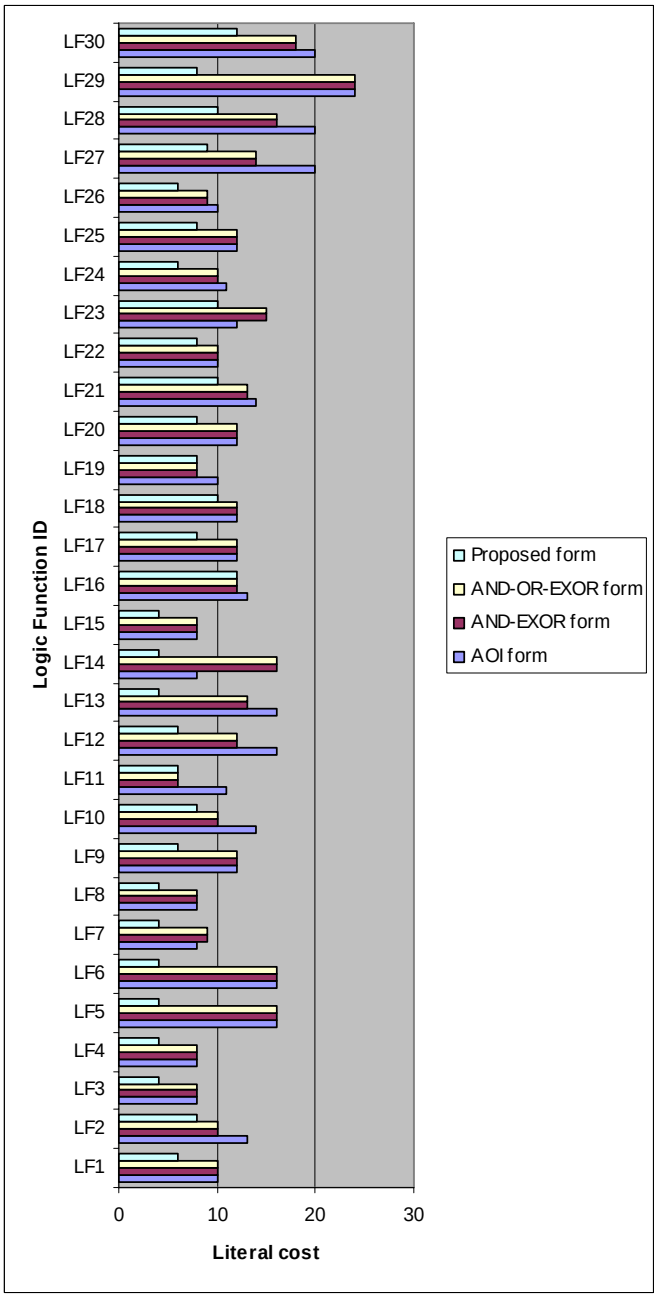


Fig. 4 Literal count comparison for various forms

At the logic level, although a number of logic types exist within the large family of AND-EXOR logic, we have selected the most general ESOP forms. We have also considered the conventional AOI logic and the AND-OR-EXOR three level networks. A multilevel logic synthesis approach has been followed throughout this work, as it usually produces the most cost effective realization of logic functions [15]. Also fan-in considerations with respect to the target technology necessitated a multilevel approach. The significance of our contribution is substantiated by improved results, in quantitative comparison with the traditional ones, at both technology-independent (in terms of literal cost) and dependent phases (with respect to power estimation) around two voltage corners, compatible with a 0.35µm TSMC CMOS process technology.

The proposed form consistently outperforms the other forms, as it acknowledges the presence of all library cells, such as EXOR, EXNOR, NAND, NOR and NOT, instead of relying upon restricted gate functionality. At 3.3V supply, it achieves mean power savings of 41.99%, 31.26% and 32.4% over multi-level AOI, AND-EXOR and AND-OR-EXOR forms respectively; while at 2.5V supply, it enables corresponding average savings of 42.92%, 32.49% and 34.06%. However, it should also be noted that the proposed method considers only the usage of library cells with a maximum fan-in of 2. Hence the method is technology-aware and is particularly useful for ASIC based designs. The method can be exploited to result in much more compact and reduced level circuits if FPGAs are the target technology, wherein fan-in does not result in a profound variation in speed. Therefore with FPGA as target, the heuristic would enable savings in terms of area and resources utilized (for e.g. LUTs in a FPGA) as this is evident from the overall reduction in literal count (43.04%) for a multi-level synthesis approach.

REFERENCES

[1] A.P. Chandrakasan, and R.W. Brodersen, "Minimizing power consumption in digital CMOS circuits," *Proc. of the IEEE*, vol. 83(4), April 1995, pp. 498-523.

[2] Christopher Umans, et. al., "Complexity of two-level logic minimization," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 25(7), July 2006, pp. 1230-1246.

[3] C. –S. Ding, et al., "Gate-level power estimation using tagged probabilistic simulation," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 17(11), November 1998, pp. 1099-1107.

[4] Sasan Iman, and Massoud Pedram. *Logic Synthesis for Low Power VLSI Designs.* New York: Springer Publishing, 1998.

[5] Dieter Jungnickel. *Graphs, Networks and Algorithms.* New York: Springer-Verlag, 2nd Edition, 2005.

[6] Tsutomu Sasao. *Switching Theory for Logic Synthesis*. Kluwer Academic Publishers, 1999.

[7] U. Kalay, M. Perkowski, and D. Hall, "A minimal universal test set for self test of EXOR-Sum-of-Products circuits," *IEEE Trans. on Computers*, vol. 49(3), March 1999, pp. 267-276.

[8] H. Rahaman, et al., "Testing of stuck-open faults in generalized Reed-Muller and EXOR sum-of-products CMOS circuits," *IEE Proc. on Computers and Digital Techniques*, vol.151(1), January 2004, pp. 83-91.

[9] Tsutomu Sasao, "EXMIN2: A simplification algorithm for Exclusive-OR-Sum-of-Products expressions for multiple-valued-input two-valued-output functions," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 12(5), May 1993, pp. 621-632.

[10] N. Song, and M. Perkowski, "Minimization of Exclusive Sum of Products Expressions for Multi-Output Multiple-Valued Input, Incompletely Specified Functions", *IEEE Trans. on CAD of integrated Circuits and Systems*, vol. 15(4), April 1996, pp. 385-395.

[11] Alan Mishchenko, and Marek Perkowski, "Fast heuristic minimization of Exclusive-Sums-of-Products", *Proc. of 5th International Reed-Muller Workshop*, 2001, pp. 242-250.

[12] Stergios Stergiou, and George Papakonstantinou, "Exact minimization of ESOP expressions with less than eight product terms," *Journal of Circuits, Systems and Computers*, vol. 13(1), February 2004, pp. 1-15.

[13] D. Debnath, and T. Sasao, "A heuristic algorithm to design AND-OR-EXOR three-level networks", *Proc. of ASP-DAC*, 1998, pp. 69-74.

[14] E.V. Dubrova, D.M. Miller, and J.C. Muzio, "AOXMIN-MV: a heuristic algorithm for AND-OR-XOR minimization," *Proc. of 4th International Reed-Muller Workshop*, 1999, pp. 37-53.

[15] Giovanni De Micheli, *Synthesis and optimization of digital circuits*. New York: McGraw Hill, 1st Edition, 1994.

[16] M.A. Harrison, *Introduction to Switching and Automata Theory*. Mc-Graw Hill, 1965.

[17] E.V. Dubrova, D.M. Miller, and J.C. Muzio, "Upper bound on number of products in AND-OR-XOR expansion of logic functions," *IET Journal of Electronics Letters*, vol. 31(7), March 1995, pp. 541-542.

Padmanabhan Balasubramanian completed his B.E degree in Electronics and Communication Engineering 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. He was earlier Lecturer in the School of Electrical Sciences at Vellore Institute of Technology (University and IET, UK Accredited), Vellore, TN, India. He is working towards his PhD in the School of Computer Science at The University of Manchester, Lancashire, UK. His research interests are in logic synthesis for low power, asynchronous design; CMOS based design and timing optimization issues.

C. Hari Narayanan received his B.Tech in Electronics and Communication Engineering from School of Engineering, Cochin University of Science and Technology, Cochin, Kerala, India in 2004. He is currently pursuing his final year M.Tech in VLSI Design from Vellore Institute of Technology (University and IET, UK Accredited), Vellore, TN, India. His interests are in software programming and digital design.