Graph based Synthesis for Low power Combinational logic with a maximal/minimal disjoint Function Set

ABSTRACT

This paper discusses a new, systematic approach to the synthesis of a 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 adjacency matrix*. Then the binary value corresponding to the candidate pair is correlated with another proposed *binary information matrix* to enable synthesis. 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 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, compatible with standard synthesis methods. Circuit level simulations demonstrate mean savings in power, gate and literal counts by 11.7%, 29.4% and 4.8% respectively, over a wide range of examples compared with other existing techniques.

Categories and Subject Descriptors

B.6.1 [**Design Styles**]: Combinational logic

General Terms

Design, Theory, Performance

Keywords

AOI logic, AND-XOR expressions, Boolean distance, Low power

1. INTRODUCTION

Logic synthesis has matured as a field and is used in every major digital IC design worldwide [7]. 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 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 [7], In general, however, the reduction is around 5%-15% [9]. 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 can be grouped 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, number of gates (N_G) and literals (N_L) required for its implementation. Of course, a comparison based on N_G is only approximate since different gate types are of different, (although roughly comparable) sizes.

The technique has its roots in the rudiments of graph and network theory. It enables best minimum solutions even for logic networks composed of elements exhibiting strong or complete adjacency, using a variation of the proposed heuristic, as will be seen in section 5.1.2. 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.

The remainder 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 background to some of the important classes of AND-XOR expressions, such as Reed-Muller (RM), Generalized RM (GRM) and Pseudo-Kronecker RM (PKRM) forms. In Section 3, the inherent nature and some fundamental properties governing Boolean functions exhibiting only maximal and exact bit-wise complementary support are mentioned. Section 4 discusses some essential graph theoretic principles. Section 5 describes the proposed graph structure and *weighted adjacency matrix* formulation with an example. Minimization heuristics for some problem categories are also described. Besides, the mode of direct synthesis by correlation of

GLSVLSI '07, March 11-13, 2007, Stresa-Lago Maggiore, Italy.

the binary data with another proposed information matrix is also dealt with in this section for minterms/maxterms as well. Section 6 briefly highlights the simulation mechanism, examples and also tabulates the results obtained for the different cases. Comparison between the different synthesis procedures in terms of P and N_G is also graphically illustrated in this section. Finally, we make the concluding remarks in Section 7.

2. CONVENTIONAL SYNTHESIS

Conventional logic design is usually based on AND-OR logic and OR-AND logic. We have used AOI logic in common with the usual library descriptions in the technology domain; this will be referred to as S*tandard 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.

XOR 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. 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. Finally, circuits containing XOR gate types have excellent design-for-test properties. Such is their significance that even some earlier FPGA styles had incorporated 2-input XOR gate in their basic granularity blocks, for example the Cli 6006 from Concurrent Logic Inc.

Various classes exist in AND-XOR expressions involving only AND and XOR gate types. This is because any arbitrary logic function can be purely realized using only AND and XOR logic gates. For example, the RM, GRM and PKRM forms extensively rely upon such gates for implementation. Slight modifications of the Shannon expansion (1) are made for the AND-XOR logic to derive Davio expansions (2), (3) and given by [10] as,

$$F = [(x \textbf{ and } F_x) \textbf{ xor } (x' \textbf{ and } F_{x'})] \quad (1)$$

$$F = [F_{x'} \textbf{ xor } \{x \textbf{ and } (F_x \textbf{ xor } F_{x'})\}] \quad (2)$$

$$F = [F_x \textbf{ xor } \{x' \textbf{ and } (F_x \textbf{ xor } F_{x'})\}] \quad (3)$$

Recursive application of the above tree expansions results in various RM trees [10]. 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 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 PKRM structure is generated. Importantly, in all these structures only two kinds of gates (AND and XOR) are used for circuit realizations. Since obtaining minimal expressions for RM, GRM and PKRM forms is by itself a separate extensive procedure and due to the necessity for a reasonable comparison with those of our proposed realizations, we have resorted to factoring those forms with elementary minimization rules in line with [3]. The objective of factorization is to represent a Boolean function in a logically equivalent factored form but with a minimum number of literals. We have opted for algebraic factorization as it is simple and requires much less CPU execution time compared to Boolean factorization. This weak operation has enabled us to greatly reduce the literal count as well as the gate count needed for such realizations, while still preserving the *integrity* of those structures. This technique is also justified in the sense that we are primarily concerned with AND-XOR logic formats. Hence, the corresponding factorized expressions would henceforth be identified as *f-RM*, *f-GRM* and *f-PKRM* expansions respectively.

3. BACKGROUND AND PRELIMINARIES

3.1 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 $\{m_i\}$ refer 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 $O(n)$.

3.1.1 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.

3.1.2 Property 2

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

3.1.3 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 $j = 0,1,\ldots,2^{(n-1)}-1$; 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 $m = 1,2,\ldots,(n+1)$.

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

4. REVIEW OF GRAPH THEORY

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 [8]. Further details regarding the pictorial descriptions and properties of all graphs and other network terminologies can be found in [8].

5. PROPOSED GRAPH & SYNTHESIS

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 [8]. The decimal equivalent of each binary minterm/maxterm shall correspond to a unique *vertex label* or identity for the proposed network. 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* $\neq \{\ \}$, the inclusion of its elements in the ON-set or OFF-set is dictated by the optimality of the best minimal solution that is predicted.

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 adjacency list representation.

5.1 Weighted Adjacency Matrix specification

A directed multigraph G with a non-zero finite vertex set can in general be represented in matrix form by an *Adjacency matrix*. For our problem specification, we opt for a slightly modified form of the latter, designated a *Weighted Adjacency Matrix (WAM)*. The *order* of an Adjacency matrix would be '$n \times n$'. The added feature of a weighted adjacency matrix over the conventional one would be that each matrix element representing the presence of a directed edge from one vertex to another, a_{ij}, is also multiplied by the decimal equivalent of the corresponding Boolean distance between the two vertices (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 end vertices. In order to preserve the structural property of an Adjacency matrix and to make it appropriate for our problem specification, we introduce *n(n-1)/2* extra edges, i.e., an edge between two end vertices is considered to be bidirectional. So, for the proposed structure, if $|V| = n$, then $|E| = n(n-1)$. Hence the proposed binary network is an extension of the complete graph functionality and this leads to a *dense network* with number of edges of *O[n(n-1)]*. The typical structure of a four terminal logic network is shown in Fig. 1. Here $W^{e(x,y)}$) represents the *weight of an edge* 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. It is also clear from the proposed graph that the in-degree/out-degree of each vertex would be *O(n)* and that the minimum eccentricity (*radius*) and maximum eccentricity (*diameter*) of each of the graph vertices would also be equal to *unity*. The *weighted adjacency matrix* of Fig. 1 would be as follows.

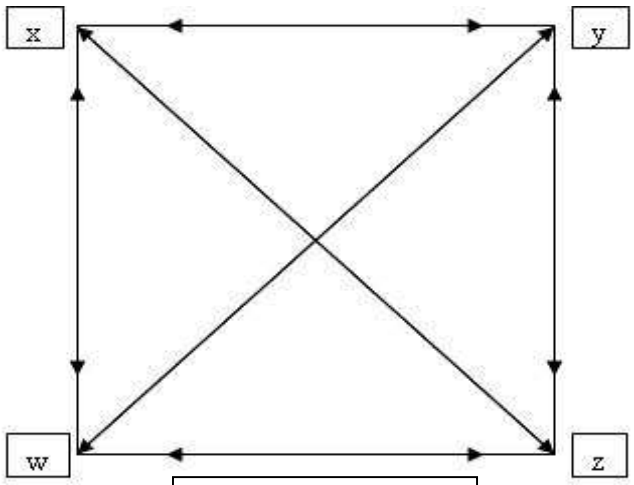

Figure 1. An example 4-terminal Boolean network

$$\begin{array}{c} \\ w \\ x \\ y \\ z \end{array} \begin{array}{c} \begin{array}{cccc} w & x & y & z \end{array} \\ \begin{bmatrix} 0 & 1\times HD(w,x) & 1\times HD(w,y) & 1\times HD(w,z) \\ 1\times HD(x,w) & 0 & 1\times HD(x,y) & 1\times HD(x,z) \\ 1\times HD(y,w) & 1\times HD(y,x) & 0 & 1\times HD(y,z) \\ 1\times HD(z,w) & 1\times HD(z,x) & 1\times HD(z,y) & 0 \end{bmatrix} \end{array}$$

From the above representation, it can be inferred that the matrix is *non-negative*, *square* and *symmetric* which implies that $|a_{ij}| = |a_{ji}|$. Here '*i*' and '*j*' assume row and column indices respectively and vice-versa.

5.1.1 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 $|E(K_n)|=n(n-1)$. A *weighted adjacency matrix* is then formed as shown above. For *Max* $|a_{ij}|=Max\ |a_{ji}|=HD[O(n)]$, group the vertices corresponding to the i^{th} and j^{th} rows or those corresponding to the i^{th} and j^{th} columns. The above procedure is continued until all the row entries or column entries in the matrix are checked. Then the binary information corresponding to the term-pair undergoes a literal matching with the information 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 most suitable for effectively synthesizing widely used practical digital circuits of any order, which contain such functionality, and include parity generator and checker circuits, with even number of input literals.

5.1.2 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 $|E(K_n)|=n(n-1)$ is sketched. A compact *weighted adjacency matrix* is framed. For $|a_{ij}| = |a_{ji}| = HD[O(unity)]$, combine those vertices corresponding to the i^{th} and j^{th} rows or those corresponding to the i^{th} and j^{th} columns.

Another *weighted adjacency 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 proposed 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 process 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 K-Map, Quine-McCluskey method or by a standard two-level logic minimizer such as Espresso-Exact. 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 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.

5.1.3 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 and is beyond the scope of this work. However, amongst the functions that can be classified in this category, we have been able to achieve good simplification, albeit with mixed performance results.

5.1.4 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 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 information matrix specification for maxterms as is evident from equations (8) and (9). For minimization 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 opposite 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 network would have a similar binary tree construction.

5.2 Synthesis with Binary Information Matrix

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 paves the way for systematic grouping and reduction of pair-wise; maximally disjoint binary 2-tuples. This is possible by a binary information matrix for a 2-tuple fully disjoint ON-set pair described by equations (6) and (7) as follows,

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

$$= [\ (x_{n-1}\ \textbf{xor}\ x_{n-2})\ \textbf{and}\ldots\ldots\ldots\textbf{and}\ (x_1\ \textbf{xnor}\ x_0)\ ] \quad (6)$$

$$= [\ (x_{n-1}\ \textbf{xnor}\ x_{n-2})\ \textbf{or}\ldots\ldots\ldots\textbf{or}\ (x_1\ \textbf{xor}\ x_0)\ ]' \quad (7)$$

A binary information matrix for a 2-tuple disjoint OFF-set pair would be described by equations (8) and (9), given below.

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

$$= [\ (x_{n-1}\ \textbf{xnor}\ x_{n-2})\ \textbf{or}\ldots\ldots\ldots\ldots\textbf{or}\ (x_0\ \textbf{xnor}\ x_{n-1})\ ] \quad (8)$$

$$= [\ (x_{n-1}\ \textbf{xor}\ x_{n-2})\ \textbf{and}\ldots\ldots\ldots\ldots\textbf{and}\ (x_0\ \textbf{xor}\ x_{n-1})\ ]' \quad (9)$$

It is clear that equations (6) and (7) and similarly equations (8) and (9) 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 information matrix has an order of '$2 \times n$'. 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.

6. SIMULATION METHOD & RESULTS

About 40 arbitrary multiple input, single output non-regenerative logic functions, representing all possible *problem cases* described above, were considered for simulation studies. Table 2 in the appendix lists some of the examples; for reasons of brevity only a representative sample have been included. The MOS transistor descriptions corresponding to the synthesized gate level netlist were simulated as the back-end using Mentor Graphics tools for a 0.35 micron TSMC CMOS process technology. The estimation of power consumption of circuits based on a finite input vector set is through tagged probabilistic simulation scheme [5]. The reason for the choice of this scheme is due to the minimal associated error component. The simulation results obtained for the target technology validate our proposition and arguments for majority of samples, detailed in Table 1. The graphical comparison plots are depicted by Figs. 2 and 3. Since the solutions obtained by our method are comparable with those of the traditional ones and industry-standard tools, we have not considered any problem-sets for *case 2*.

Table 1. Comparison of design metrics for different synthesis styles

Logic Function ID	Standard form		f-RM form		f-GRM form		f-PKRM form		Proposed form	
	P (nW)	N_G	P (nW)	N_G	P (nW)	N_G	P (nW)	N_G	P (nW)	N_G
Case 1 - LF1[4]	10.78	9	13.46	5	13.46	5	13.46	5	9.67	3
Case 1 - LF2[4]	14.44	9	5.7	3	6.44	5	5.7	5	5.7	3
Case 3 - LF3[4]	9.65	6	10.4	5	12.16	7	9.65	7	8.15	3
Case 1 - LF4[4]	20.14	13	11.38	4	14.95	4	11.38	4	11.25	3
Case 3 - LF5[4]	7.76	6	14.79	6	13.81	6	13.81	6	9.67	3
Case 3 - LF6[4]	11.18	7	8.64	5	7.8	6	8.64	5	5.99	3
Case 1 - LF7[5]	16.49	10	16.01	8	18.75	9	18.75	9	9.82	4
Case 3 - LF8[5]	16.67	9	10.12	4	10.12	4	10.12	4	8.23	3
Case 1 - LF9[5]	7.76	6	14.79	6	13.81	6	13.81	6	9.67	3
Case 3 - LF10[5]	29.09	13	16.15	8	17.23	8	23.39	11	12.27	5
Case 3 - LF11[5]	10.73	14	13.46	5	13.46	5	13.46	5	9.66	3
Case 3 - LF12[5]	12.51	9	14.5	5	20.57	7	23.6	9	8.15	3
Case 3 - LF13[5]	12.59	9	16.81	6	14.59	5	21.79	9	8.19	3
Case 1 - LF14[5]	29.24	15	8.03	3	12.23	5	12.23	5	8.03	3
Case 3 - LF15[5]	12.75	7	13.82	8	14.14	8	14.14	8	10.56	6
Case 3 - LF16[5]	19.48	11	13.09	6	13.09	6	13.09	6	13.11	6
Case 1 - LF17[5]	13.38	8	16.57	10	16.57	10	16.57	10	13.05	7
Case 1 - LF18[5]	14.49	9	5.7	3	6.44	5	6.44	3	5.7	3
Case 1 - LF19[5]	26.42	15	11.38	4	14.5	4	14.5	4	9.48	3
Case 3 - LF20[5]	15.74	8	11.72	7	11.72	7	11.72	7	15.72	8
Case 1 - LF21[6]	18.15	11	22.34	12	25.95	14	25.95	14	16.34	5
Case 1 - LF22[6]	25.82	15	20.62	5	19.75	5	20.62	5	17.52	4
Case 3 - LF23[6]	15.69	10	11.73	7	14.32	8	14.32	8	8.25	4
Case 3 - LF24[6]	15.68	9	13.42	7	14.14	8	14.14	8	9.89	4
Case 3 - LF25[6]	16.98	10	17.88	8	23.81	9	23.81	9	12.22	4

LFM^n; LF – Logic Function, M – Function identity, n – Number of input literals

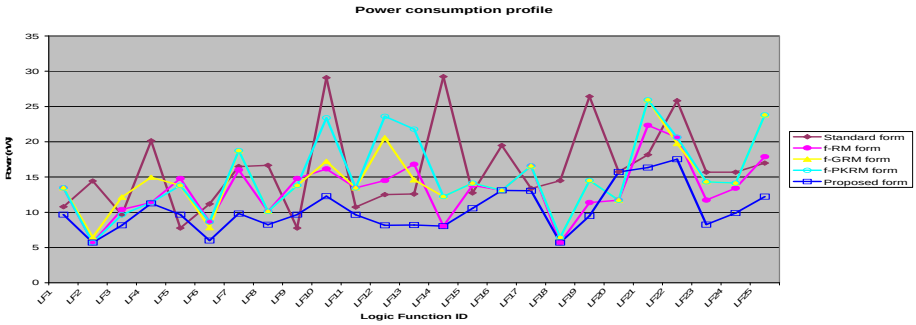

Figure 2. Power consumption comparison for different synthesis methods

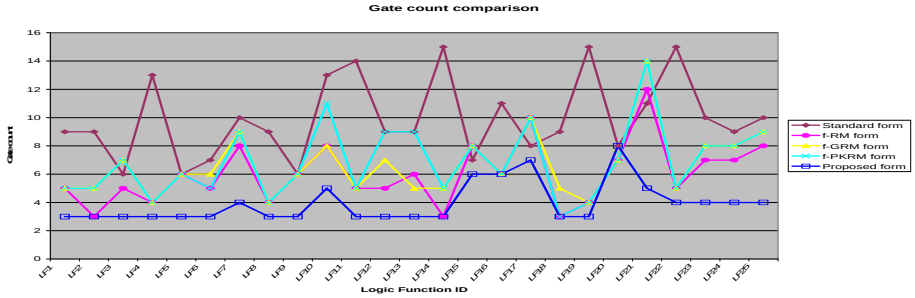

Figure 3. Gate count comparison for various synthesis schemes

Let us consider the output truth vector of a four-variable Boolean function given by $[0,1,1,0,0,0,0,0,0,0,0,1,1,0]^T$, which falls under *case 1*. The *power consumption metric* found out by simulation with a technology target, for the solution obtained based on the standard form is 16.7 nW. Among the different Reed-Muller forms, f-PKRM has the lowest value of 10.1 nW and our proposed form has a value of 8.2 nW respectively.

For the truth vector of a four-variable logic function, given by $[0,0,0,0,1,1,1,0,1,1,0,0,0,0,0]^T$ (belonging to *case 3*), the standard, f-GRM and proposed forms have power consumption values of 14.4 nW, 13.3 nW and 10.7 nW respectively.

For *case 4*, we consider the truth vector of a 4-input logic function as, $[1,1,1,1,0,1,1,0,0,1,1,0,1,1,1,1]^T$. In this case, the total power metric was found for the standard, f-PKRM and proposed forms respectively as 16.0 nW, 10.0 nW and 8.1 nW.

From the detailed simulation results mentioned in Table 2, we have considered the best minimum power amongst those of the other conventional methods and compared it with our method. The percentage savings/extra expenditure incurred in each case was then averaged to arrive at the net savings for all the examples considered. The same procedure has been followed with respect to gate count and literal count.

7. CONCLUSIONS

The computational complexity associated with logic simplification has posed challenges since the early 60's [2]. Exact solutions for some central problems have been derived only within the last few years and others remain open. This paper has addressed an important issue of practical relevance, by means of introducing a novel approach to logic reduction for certain classes 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 significant works, such as [4] and [6], as it does not expect any intuitive/complex analytical solutions, but rather enables 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 enabling hand-crafted solutions for smaller degree problems.

Algorithmic logic synthesis is usually carried out in two stages, the independent stage where the given Boolean equations are minimized with no regard to physical properties and the dependent stage where mapping to a physical cell library is done. At the logic level, although a number of logic types exist within the large family of XOR Sum of Products expressions [10], we have selected only three of them (RM, GRM and PKRM forms), owing to a relatively narrower solution choice and reduced computational cost required to arrive at a best minima. The importance of our contribution is substantiated by improved results, in quantitative comparison with the traditional ones, at both the technology-independent and dependent phases. Further work is carried out to complete the automation of the synthesis process to result in an integrated EDA tool for XOR and/or XNOR dominated switching circuits.

8. REFERENCES

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

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

[3] Unni Narayanan, and Liu, C.L. Low Power logic synthesis for XOR based circuits. *Proc. of the IEEE/ACM ICCAD* (San Jose, California, USA, Nov. 9-13, 1997), 570-574.

[4] Bernasconi, A. Fast Three-level logic Minimization based on Autosymmetry. *Proceedings of the 39th ACM/IEEE DAC* (New Orleans, Louisiana, USA, June 10-14, 2002), 425-430.

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

[6] Valentina Ciriani. Logic Minimization using Exclusive OR Gates. *Proceedings of the 38th ACM/IEEE DAC* (Las Vegas, Nevada, USA, June 18-22, 2001), 115-120.

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

[8] Dieter Jungnickel. *Graphs, Networks and Algorithms.* Springer-Verlag, 2nd Edition, 2005.

[9] Farzad Nekoogar, and Faranak Nekoogar. *From ASICs to SOCs: A Practical Approach.* Prentice-Hall, 2003.

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

APPENDIX

Table 2. Description of Boolean functions

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