International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:2, No:5, 2008

Library Aware Power Conscious Realization of Complementary Boolean Functions

Padmanabhan Balasubramanian, and Cemal Ardil

***Abstract*—**In this paper, we consider the problem of logic simplification for a special class of logic functions, namely complementary Boolean functions (CBF), targeting low power implementation using static CMOS logic style. The functions are uniquely characterized by the presence of terms, where for a canonical binary 2-tuple, $D(m_j) \cup D(m_k) = \{ \}$ and therefore, we have $| D(m_j) \cup D(m_k) | = 0$ [19]. Similarly, $D(M_j) \cup D(M_k) = \{ \}$ and hence $| D(M_j) \cup D(M_k) | = 0$. Here, '$m_k$' and '$M_k$' represent a minterm and maxterm respectively. We compare the circuits minimized with our proposed method with those corresponding to factored Reed-Muller (f-RM) form, factored Pseudo Kronecker Reed-Muller (f-PKRM) form, and factored Generalized Reed-Muller (f-GRM) form.

We have opted for algebraic factorization of the Reed-Muller (RM) form and its different variants, using the factorization rules of [1], as it is simple and requires much less CPU execution time compared to Boolean factorization operations. This technique has enabled us to greatly reduce the literal count as well as the gate count needed for such RM realizations, which are generally prone to consuming more cells and subsequently more power consumption. However, this leads to a drawback in terms of the design-for-test attribute associated with the various RM forms. Though we still preserve the definition of those forms viz. realizing such functionality with only select types of logic gates (AND gate and XOR gate), the structural integrity of the logic levels is not preserved. This would consequently alter the testability properties of such circuits i.e. it may increase/decrease/maintain the same number of test input vectors needed for their exhaustive testability, subsequently affecting their generalized test vector computation.

We do not consider the issue of design-for-testability here, but, instead focus on the power consumption of the final logic implementation, after realization with a conventional CMOS process technology (0.35 micron TSMC process). The quality of the resulting circuits evaluated on the basis of an established cost metric viz., power consumption, demonstrate average savings by 26.79% for the samples considered in this work, besides reduction in number of gates and input literals by 39.66% and 12.98% respectively, in comparison with other factored RM forms.

***Keywords*—** Reed-Muller forms, Logic function, Hamming distance, Algebraic factorization, Low power design.

I. Introduction

THE low power design challenge is one that requires abstraction, modeling and optimizations at all levels of design hierarchy; including the system, algorithmic, architectural, logic, circuit and process levels. Perhaps the driving factor for designing low power circuits in the recent times has been the remarkable success and growth of the important class of portable and personal computing devices, high-end wireless communication systems etc., which demand complex functionality, high throughput, low weight and long operation times before the battery is recharged. In these applications, average power consumption is a critical design constraint. Combining optimizations at all the levels results in orders of magnitude of power reduction [2]. Such an impressive reduction in circuit power will however be possible only if optimization flows and techniques at each level of design hierarchy are developed [24].

Logic synthesis has matured as a field to be universally accepted and is used in every major IC design and production house worldwide. A wealth of research results and a few pioneering commercial tools for low power logic synthesis have appeared in the last couple of years. Logic synthesis is an important part of the design cycle for a digital integrated circuit. This implies that in order to minimize power effectively, power component should be considered during the logic synthesis phase. In certain cases, gate level optimization results in more than 50% power reduction without sacrificing the circuit speed [2]. This approach promises to be very successful and useful, since the investment to reduce power by design is relatively small in comparison to other techniques and also because it is relatively untapped in potential.

This paper presents a power optimization technique at the logic level, which is ultimately expected to minimize power consumption of the transistor level realization of the logic implementation by a new methodology, and also by taking advantage of existing techniques like logic restructuring for minimizing area. Logic restructuring techniques include common sub-function extraction and factorization. In this paper, algebraic factorization is widely considered and power optimization after logic extraction for a unique class of functions, namely CBF is addressed as follows.

Problem statement: Given a Boolean function comprising a complementary ON/OFF set of elements (exhibiting bit-wise or position-wise negation), find a reduced and possibly minimal cover for the network, such that the final size and power cost of the network are reduced. Indeed, the power cost of the network is theoretically modeled as the summation of power cost at all the primary gate input, intermediate gate output and primary gate output nodes.

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).

Cemal Ardil is with the National Academy of Aviation, Baku, Azerbaijan (e-mail: cemalardil@gmail.com).

The aim of this work is to exploit the "regularity" inherent in complementary Boolean functions (CBF), in order to reduce the time needed for its logical synthesis to obtain a power optimal solution. In light of the above, this work addresses the issue of how to derive a minimized decomposition of complementary logic functions by predominant usage of XOR and/or XNOR gates, consistent with a standard cell library specification and also the means to develop an integrated framework for reducing such functions, so as to reduce the literal count and/or gate area of the final design. In this context, it should be noted that though power optimization is achieved by an area-centric approach.

As an initial theoretical measure to forecast the possibility of power savings that is likely to be attained; transition count [3], as an integer measure, is used as a parameter to represent the switching activity at all the gate output nodes of all the logic levels. However, we observe that this integer metric does not correspond to/reflect the actual power consumption or savings that may be attained by one synthesis method over another. But overall, this parameter serves only as a useful indicator of possible power efficient implementations, which is validated by the problem cases considered in this work.

The remainder of this paper is organized as follows. Section 2 classifies the complementary Boolean functions into two types and also lists their associated conditionalities. Section 3 provides concise background information pertaining to ESOP and also sheds light on different factored RM forms. Section 4 presents the details regarding the methodology followed to group the terms and also explains the synthesis procedure. A specific problem case is also considered to serve as an illustration for the proposed synthesis technique. The simulation mechanism and results obtained for the different factored RM forms and that corresponding to the proposed method for a considerable number of case studies, realized using static CMOS logic style, is highlighted in section 5. Also graphical plots for the cost metrics pertaining to different realizations are included in this section. Lastly, we conclude and cite scope for further work in section 6.

II. Complementary Boolean Function

Let F be a Boolean function with support of F [4], defined as, $s[F] = \{x_{n-1}, x_{n-2},\ldots,x_0\}$. The support set basically lists all the variables that influence a function output. We now classify CBF into two types: CBF specified in terms of minterms (m-CBF) and a CBF defined by its maxterms (M-CBF). The details follow.

A. Proposition 1: m-CBF

A logic function is defined as a minterm-based CBF (m-CBF), if and only if equation (2) is satisfied.

$$F_{ON} = \{m_i\};\ 0 \le i \le (n-1), \text{ such that } 0 \le n \le 2^{|s[F]|} \quad (1)$$

where, for every $m_j \in \{m_i\}$, there certainly exists a $m_k \in \{m_i\}$, such that the binary 2-tuple (m_j, m_k) is described as pair-wise disjoint or bit-wise complementary. For this, the following condition has to be satisfied, given by,

$$D(m_j) \cup D(m_k) = \{\ \} \text{ and } |\ D(m_j) \cup D(m_k)\ | = 0 \quad (2)$$

where, $D(m_j)$ and $D(m_k)$ represent the description set [19] of minterms 'm_j' and 'm_k' respectively.

B. Proposition 2: M-CBF

A logic function is understood to be a maxterm-based CBF (M-CBF), if and only if equation (4) holds well.

$$F_{OFF} = \{M_i\};\ 0 \le i \le (n-1), \text{ where } 0 \le n \le 2^{|s[F]|} \quad (3)$$

where, for each $M_j \in \{M_i\}$, there certainly exists a $M_k \in \{M_i\}$, in which case the binary 2-tuple (M_j, M_k) is identified to be pair-wise disjoint or bit-wise complementary. Alternatively, the following conditionality would hold good.

$$D(M_j) \cup D(M_k) = \{\ \} \text{ and } |\ D(M_j) \cup D(M_k)\ | = 0 \quad (4)$$

The conditionalities (2) and (4) are elucidated in section 4.

III. ESOP and Factored Reed-Muller forms

Traditional logic design is usually based on two-level SoP (Sum of Product terms) and PoS (Product of Sum terms) forms. Infact, two-level SoPs have been widely used to represent and manipulate Boolean functions. Exact and heuristic SoP minimization has traditionally attracted attention of researchers over several years, because in many applications, it is important to have as compact a SoP representation as possible. [5] and [6] indicate the earliest research and a recent research work undertaken in this context. But two-level logic is of less significance in a VLSI design environment and is clearly library unaware, due to fan-in restrictions (generally limited to three inputs for generic atomic operators) imposed on the gates in a standard cell library. Hence, multi-level logic seems to the practical design alternative and this is beneficial for enhanced performance even though there is a trade-off with logic depth [7].

XOR based designs, on the other hand, 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-testability properties.

Exclusive Sum-of-Products (ESOP) have been introduced by I. I. Zhegalkin in 1927 [8] [9] and later independently rediscovered by S.M. Reed [10] and D.E. Muller [11]. A systematic classification of various families of Reed-Muller (Zhegalkin) expressions has been given in [12]. In simple terminology, an ESOP is an Exclusive-OR of zero or more cubes, where a cube is a product term composed of literals using Boolean AND operation and a literal is a Boolean variable appearing in positive or negative polarity. An ESOP

is reduced if it does not contain identical cubes. An ESOP is minimal if all cube pairs have distance 2 or more. An ESOP is exact minimum if it contains the minimum number of cubes among all ESOPs representing the given function. The following propositions are proved using the basic property of an XOR operation.

A. Lemma 1

Two identical cubes (Hamming distance – 0 cubes) can be added to any ESOP without changing the functionality represented by them.

B. Lemma 2

The XOR of two cubes that exhibit unity Hamming distance can be represented by a single cube.

Inspite of their inherent advantages, XOR gate and Exclusive Sum-of-Products (ESOP) minimization, have been underestimated for the following reasons: the XOR gate, as implemented in the widely used standard cell libraries, is almost two times larger and slower compared to equivalent fan-in NAND and NOR gates. Relatively few algorithms use ESOPs for internal representation of Boolean functions. Finally, the exact minimization of ESOPs is practical only for arbitrary functions of five input variables and special classes of functions up to ten variables [13], while heuristic minimization is based on search algorithms [25] and is computationally more expansive than SoP minimization. Although, for long, it has been conjectured that ESOPs require fewer products than sum-of-products expressions (SoPs) by experiments using randomly generated functions; for e.g. an ESOP requires only 't' products to represent a parity function of 'n' variables, while the SoP requires $2^{(t-1)}$; this is not always the case. In case of Achilles' heel functions with input file specification $(2^k, 2)$ for k = 0,1,…,2r; the number of products to represent this function would be 'r' for SoPs and (2^r-1) for ESOPs. Despite these individual comparison cases, ESOPs are found to be result in concise expressions than SoPs for arithmetic functions. For e.g. the sufficient number of products to represent an n-bit adder would be $[2^{(n+1)} - 1]$ for ESOPs, $[2^{(n+1)} + n - 2]$ for other classes of AND-XOR expressions and $[6.2^n – 4n – 5]$ for SoPs [17].

However, the XOR gate and ESOPs continue to play an important role in logic synthesis, design-for-test and other areas of computer technology due to these important reasons: for many Boolean functions, the number of cubes in minimal ESOPs is less than the number of cubes in minimal SOPs. Since the XOR gate has excellent testability properties, as mentioned above, it subsequently leads to efficient methods for automatic test pattern generation [14]. Also, it is clear from the previous discussion, that for many types of practical circuits, even the selective use of XOR gates in logic synthesis yields better implementations in terms of both area and delay [15] [16]; although, in the paper, we study the effectiveness of XOR based designs mainly from a power perspective.

Various classes exist in ESOP expansions 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 expressions form only a subset of the ESOP form. Slight modifications of the Shannon expansion in GF (2) are made to obtain Davio-1 (5), Davio-2 (6) (positive Davio expansion) and Davio-3 (7) (negative Davio expansion) axioms, which correspond to AND-XOR logic [17] [18].

$$F = [(x \cdot F_x) \oplus (x' \cdot F_x')] \quad (5)$$

$$F = [F_x' \oplus \{x \cdot (F_x \oplus F_x')\}] \quad (6)$$

$$F = [F_x \oplus \{x' \cdot (F_x \oplus F_x')\}] \quad (7)$$

In (5), (6) and (7), the symbols '$\cdot$' and '$\oplus$' stand for AND and XOR operators respectively, while '+' and '$\odot$' would henceforth correspond to logical OR and XNOR operations. 'x' is the decision variable, 'F_x' is the positive residue (positive cofactor) of the function F and 'F_x'' is the negative residue (negative cofactor) of the function F.

Recursive application of the above tree expansions results in various RM trees [18]. 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, apart from NOTs as necessary.

Since obtaining minimal expressions for RM, GRM and PKRM forms is by itself a separate extensive procedure and even then it would contain many XOR operators; owing to the necessity for a reasonable comparison with that of our proposed realization, we have resorted to factoring those forms with useful minimization rules stated in [1]. We also make it clear that we consider only RM, PKRM and GRM forms with fixed polarity in this work. Though in PKRM and GRM expressions, both true and complemented literals can appear for the same variable, in case of standard cell based IC design (CBIC), this assumption would not hold good.

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 operation has enabled us to greatly reduce the literal count as well as the gate count needed for such realizations, while still preserving the definition of those structures. This technique is also justified in the sense that we are primarily concerned with reduced and irredundant AND-XOR logic formats. Hence, the corresponding factorized expressions are identified as f-RM, f-GRM and f-PKRM expansions respectively.

IV. Grouping Methodology and Synthesis Procedure

In [19], we introduced a new terminology, namely the description set of a Boolean term and the reader is referred to the same for details. In short, the description set of a canonical product is the set of all primary input literals that a minterm is dependent upon, in their respective polarities, for its evaluation to a logical HIGH state. Similarly, the description set of a canonical sum is the set of all primary input literals that a particular maxterm is dependent upon, in their respective polarities, for its evaluation to a logical LOW state. We now first discuss how the terms (minterms or maxterms) are grouped based on their description set and set union operation [19]; then proceed with the explanation of the proposed synthesis procedure.

Let 'm_1' and 'm_2' be two minterms, specified by their respective description sets as, $D(m_1) = \{y_{n-1}, y_{n-2}, \ldots, y_0\}$ and $D(m_2) = \{y_{n-1}, y_{n-2}, \ldots, y_0\}$, where '$y_k$' assumes either of the logical states of '0' or '1', and k = (n-1), (n-2),….,0. If $[y_k \varepsilon D(m_1)] \neq [y_k \varepsilon D(m_2)]$, for all 'k', then we have,

$$D(m_1) \cup D(m_2) = \{\ \} \text{ and } | D(m_1) \cup D(m_2) | = 0 \quad (8)$$

Now we arrive at the conclusion that 'm_1' and 'm_2' are suitable candidates for grouping. The above illustration is suitable for maxterms as well. This grouping mechanism is suitable for all CBF. Care must be taken to perform only as many distinct and minimum number of grouping operations, as deemed necessary, and no terms are to be left un-checked and also that each grouping operation would be performed on only two distinct standard sum (product) terms.

Now we proceed with a two-bit comparison at a time, starting with the most significant input literal and its next significant variable, until comparison between the least significant variable and its predecessor. In such a case, if 'y_{n-1}' and 'y_{n-2}' of $D(m_1)$ [likewise $D(m_2)$] have the same Boolean values, then they could be combined by a logical inclusive operation as $(y_{n-1} \odot y_{n-2})$. On the other hand, if they are different, they could be combined by a logical exclusive operation as $(y_{n-1} \oplus y_{n-2})$. It should be remembered in this context that the above logical operations correspond only to the two minterms 'm_1' and 'm_2' considered. We then logically AND all such combined variables. We could also have an alternative solution in terms of logical OR-ing of all such combined variables. For this, the logical inclusive and logical exclusive operators have to be swapped and the resulting expression is to be complemented. In a similar manner, we could proceed with other sets-of-two minterms, which satisfy (8). Hence, it becomes clear that two expressions could be obtained; one corresponding to a logical conjunctive form and another corresponding to a logical disjunctive form. However, in general, we prefer the former solution rather than the latter, as NAND gates tend to be faster than NOR gates, when realized in static CMOS logic style [20].

For a Boolean function, F, which is not strictly CBF, its ON-set could be described by, $F_{ON} = \{m_i\}$, where $\{m_i\}$ represents the set of all distinct minterms constituting the ON-set of the function F. Then, for every $m_j \in \{m_i\}$, there exists an element, $m_k \in \{m_i\}$, where $D(m_j) \cup D(m_k) = \{\ \}$, and (8) is satisfied. In addition, there also exists atleast one more element, $m_p \in \{m_i\}$, where the inequality, $1 < | D(m_j) \cup D(m_k) | \leq O(n-1)$ is satisfied. In this case, 'm_p' could be combined with any of $\{m_i\}$, such that $| D(m_p) \cup$ any of $D[\{m_i\}] |$ is a minimum. This means that one or more variables would be common between the description sets of 'm_p' and $\{m_i\}$. So the above steps detailed in the previous paragraph may be followed for the remaining variables and the shared variables could be AND-ed with the resulting expressions.

Once all the minterms have been checked, we then resort to algebraic factorization operations at this stage, by applying lemmas 1 and 2 as stated in the previous section and other elementary algebraic factorizations on the various logical disjunctions/conjunctions obtained, which results in a minimal and irredundant solution (where the minimality criterion is measured in terms of literal count and gate count). This synthesis procedure is significant, in the sense that XOR and XNOR gates with a fan-in of only 2 are used for realization. In case of other Boolean gates (viz. AND or OR), we can go for a cascade network, in order to adhere to the technology cell library specifications. Hence the synthesis mechanism is clearly library aware as evident from the above discussion.

However, for the problem comprising maxterms, we adopt a similar but slightly different methodology. The initial set-of-two maxterms comparison should be extended to accommodate an extra comparison between the most significant input variable and the least significant input variable. This would give rise to an extra term in the resulting expression. In order to avoid this, the maxterms-defined problem may be considered as a minterm-defined problem by applying the concept of output phase optimization. This is considered to be a wise and pragmatic approach for these types of problems. Secondly, though the logical inclusive and exclusive operations used for combining input variables would be the same; the expression obtained by logical disjunction would be negated instead of inverting that obtained by logical conjunction, as in the previous case.

For a logic function, specified in terms of its maxterms, which is strictly not CBF, a similar grouping mechanism has to be followed as detailed for that of a minterm specified problem. The only difference being that the variables which are common are to be OR-ed with the resulting expressions.

A. Example

Let the ON-set of a 5-variable logic function be described by,

$$F_{ON}(a,b,c,d,e) = m_8 + m_{11} + m_{20} + m_{23} \quad (9)$$

For this problem, m_8 and m_{23}; similarly m_{11} and m_{20} satisfy (8). The gate-level implementations of the final solutions corresponding to the various forms viz. f-RM, f-GRM, f-PKRM and proposed forms are shown in figures 1, 2, and 3

respectively. Their corresponding Boolean relations are given by (9), (10), (11) and (12).

$$F_{f\text{-}RM} = (d \oplus e') \cdot [(bc') \oplus \{a(b \oplus c)\}] \quad (10)$$

$$F_{f\text{-}GRM} = (d \oplus e') \cdot [(b'c) \oplus \{a'(b \oplus c)\}] \quad (11)$$

$$F_{f\text{-}PKRM} = (d \oplus e') \cdot [(b'c) \oplus \{a'(b \oplus c)\}] \quad (12)$$

$$F_{Proposed} = (a \oplus b) \cdot (b \oplus c) \cdot (d \odot e) \quad (13)$$

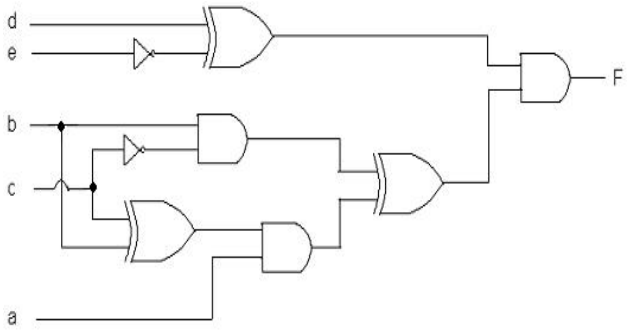

Fig. 1 Factored RM form realization

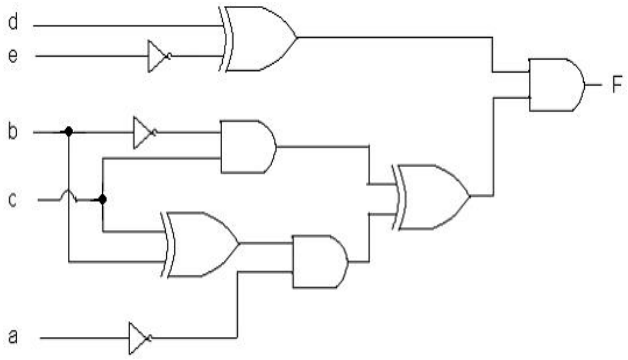

Fig. 2 Factored GRM and PKRM forms implementation

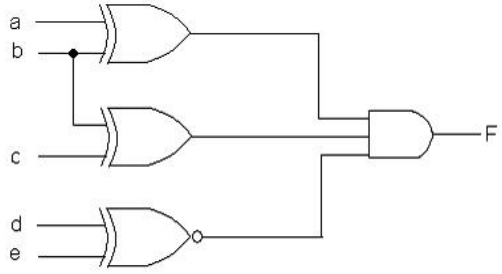

Fig. 3 Synthesized function based on proposed method

A comparison between the various realization schemes in terms of technology-independent and technology-dependent parameters is shown in Table I. From the following table, we make some important inferences. Firstly, transition count [3], as an integer measure of switching activity at all the gate output nodes for a uniform input distribution, forecasts that the proposed form is likely to garner savings in power consumption over the other forms. The percentage savings obtained for the proposed form over the other forms is indicated within brackets in the corresponding columns (in all the Tables). This has been done for the other parameters as well. Also, transition count as well as power consumption of both the f-GRM and f-PKRM forms turns out to be the same.

TABLE I
COMPARISON BETWEEN VARIOUS REALIZATIONS

Realization	Transition count	N_G	N_L	Power (nW)
f-RM form	3040 (42.11%)	8 (50%)	7 (14.28%)	16.001 (38.64%)
f-GRM form	3552 (50.45%)	9 (55.56%)	7 (14.28%)	18.573 (47.13%)
f-PKRM form	3552 (50.45%)	9 (55.56%)	7 (14.28%)	18.573 (47.13%)
Proposed	1760	4	6	9.819

N_G – Number of gates; N_L – Number of input literals

However, it should be noted that computation of transition count would be a cumbersome task for non-uniformly distributed input patterns (i.e. that which may exhibit spatial, temporal or spatio-temporal correlations). Also, it may not indicate the actual savings in power consumption that is likely to be achieved since it is independent of technology parameters and depends only on statistics of the primary input signals. This is also evident from the above table.

V. Simulation Mechanism and Results

For a given Boolean function, minimized expressions and equivalent gate level realizations were obtained based on the different synthesis schemes viz., f-RM, f-GRM, f-PKRM and proposed. The gate level netlists were then realized using static CMOS logic style, by library matching with the cells normally available in a standard cell library. The gate level primitives have been described in terms of a MOS transistor schematic, using Mentor Graphics tools (DA - Design Architect) [21]. In this full-custom approach, the length dimension of all the transistors is dictated by a technology specification viz., 0.35 micron TSMC CMOS process, with a gain factor (β_p/β_n) of 2.5. The SPICE netlist for the transistor level description corresponding to an industry-standard BSIM3 device model was obtained at the back-end and executed using ELDO circuit simulator. The simulation waveforms have been observed using XELGA waveform viewer tool. The above simulation procedure has been followed for all the gate-level netlists, corresponding to the respective synthesis schemes.

A number of multiple-input, single-output non-regenerative logic functions (most of them CBF and few not-strict CBF) have been taken for analysis purpose and they are listed in Table 6 (made available as an appendix).

Table 2 gives the transition count computed for a uniform input distribution for the minimized gate-level solutions obtained according to the different synthesis methods. Tables 3 and 4 summarize the gate count and literal count, for the case studies considered as per the different realizations. Power

dissipation values obtained for the various samples considered, corresponding to different synthesis schemes have been listed in Table 5. The estimation of average power component of the circuits is through tagged probabilistic simulation scheme [22]. The reason for choosing this method is that the error component in this method is minimal, and to improve its efficiency, only tagged waveforms at the circuit inputs have been computed. The total power consumption value obtained for a representative input pattern (assuming uniformly distributed and uncorrelated input sequences) was then summed up in each case to estimate the mean savings in power dissipation for the proposed method over those corresponding to other methods, on an overall basis. A graphical comparison of the proposed scheme with those of other methods is depicted by figures 4, 5, 6 and 7.

TABLE II
EVALUATION OF TRANSITION COUNT FOR VARIOUS REALIZATIONS

Function ID	f-RM form	f-GRM form	f-PKRM form	Proposed form
$F1^4$	608	608	608	352
$F2^4$	480	736	480	352
$F3^4$	352	608	608	352
$F4^4$	568	824	824	352
$F5^4$	480	480	480	352
$F6^4$	512	512	512	384
$F7^4$	696	696	696	352
$F8^4$	440	440	440	344
$F9^4$	492	830	830	408
$F10^4$	12666	1024	1024	600
$F11^5$	3040	3552	3552	1760
$F12^5$	1920	1920	1920	1408
$F13^5$	2688	2688	2688	1408
$F14^5$	3296	3520	4228	2496
$F15^5$	2432	2432	2432	1408
$F16^5$	2360	3328	3712	1408
$F17^5$	2816	2360	3712	1408
$F18^5$	2816	2360	3712	1408
$F19^5$	1536	2560	2560	1536
$F20^5$	3376	3376	3376	2480
$F21^5$	2480	2480	2480	1968
$F22^5$	4368	4368	4368	2704
$F23^5$	1408	2432	1408	1408
$F24^5$	2048	2048	2048	1536
$F25^5$	2808	2808	2808	3320
$F26^6$	22336	22336	22336	8672
$F27^6$	9088	9088	9088	7040
$F28^6$	11648	13696	13696	7040
$F29^6$	11648	13696	13696	7040
$F30^6$	13696	15744	15744	7040
Total (% increase)	125102 (45.38%)	12355 (44.69%)	126066 (45.79%)	68336

TABLE III
COMPARISON IN TERMS OF GATES AVAILABLE IN THE CELL LIBRARY

Function ID	f-RM form	f-GRM form	f-PKRM form	Proposed form
$F1^4$	5	5	5	3
$F2^4$	4	6	4	3
$F3^4$	3	5	5	3
$F4^4$	5	7	7	3
$F5^4$	4	4	4	3
$F6^4$	4	4	4	3
$F7^4$	6	6	6	3
$F8^4$	5	6	5	3
$F9^4$	8	7	7	5
$F10^4$	12	10	10	5
$F11^5$	8	9	9	4
$F12^5$	4	4	4	3
$F13^5$	6	6	6	3
$F14^5$	8	8	11	5
$F15^5$	5	5	5	3
$F16^5$	5	7	9	3
$F17^5$	6	5	9	3
$F18^5$	6	5	9	3
$F19^5$	3	5	5	3
$F20^5$	8	8	8	6
$F21^5$	6	6	6	6
$F22^5$	10	10	10	7
$F23^5$	3	5	3	3
$F24^5$	4	4	4	3
$F25^5$	7	7	7	8
$F26^6$	12	14	14	5
$F27^6$	5	5	5	4
$F28^6$	7	8	8	4
$F29^6$	7	8	8	4
$F30^6$	8	9	9	4
Total (% increase)	184 (35.87%)	198 (40.40%)	206 (42.72%)	118

TABLE IV
LITERAL COST COMPARISON FOR DIFFERENT SYNTHESIS SCHEMES

Function ID	f-RM form	f-GRM form	f-PKRM form	Proposed form
$F1^4$	4	4	4	4
$F2^4$	4	4	4	4
$F3^4$	4	4	4	4
$F4^4$	5	5	5	4
$F5^4$	4	4	4	4
$F6^4$	4	4	4	4
$F7^4$	5	5	5	4
$F8^4$	5	4	5	4
$F9^4$	7	8	8	7

$F10^4$	12	10	10	6
$F11^5$	7	7	7	6
$F12^5$	4	4	4	4
$F13^5$	5	5	5	4
$F14^5$	8	7	11	6
$F15^5$	4	4	4	4
$F16^5$	4	6	7	4
$F17^5$	6	4	7	4
$F18^5$	6	4	7	4
$F19^5$	4	4	4	4
$F20^5$	9	8	8	8
$F21^5$	8	8	8	8
$F22^5$	10	10	10	10
$F23^5$	4	4	4	4
$F24^5$	4	4	4	4
$F25^5$	8	8	8	9
$F26^6$	12	12	12	8
$F27^6$	6	6	6	6
$F28^6$	7	7	7	6
$F29^6$	7	7	7	6
$F30^6$	7	7	7	6
Total (% increase)	184 (13.04%)	178 (10.11%)	190 (15.79%)	160

TABLE V
POWER DISSIPATION FOR VARIOUS SCHEMES (IN NANOW)

Function ID	f-RM form	f-GRM form	f-PKRM form	Proposed form
$F1^4$	13.459	13.460	13.460	9.669
$F2^4$	10.049	10.776	10.049	8.142
$F3^4$	5.701	6. 444	5.701	5.701
$F4^4$	10.399	12.159	9.646	8.145
$F5^4$	10.123	10.849	10.123	8.234
$F6^4$	11.384	14.947	11.384	11.253
$F7^4$	14.793	13.810	13.810	9.668
$F8^4$	8.635	7.801	8.635	5.992
$F9^4$	7.553	13.294	8.943	10.678
$F10^4$	14.621	18.210	18.210	15.171
$F11^5$	16.001	18.753	18.753	9.819
$F12^5$	10.123	10.123	10.123	8.234
$F13^5$	14.793	13.810	13.810	9.668
$F14^5$	16.154	17.229	23.390	12.274
$F15^5$	13.459	13.461	13.459	9.664
$F16^5$	14.497	20.574	23.604	8.145
$F17^5$	16.180	14.592	21.797	8.187
$F18^5$	16.810	14.592	21.797	8.187
$F19^5$	8.025	12.228	12.228	8.025
$F20^5$	13.816	14.136	14.136	10.556
$F21^5$	13.089	13.089	13.089	13.108
$F22^5$	16.565	16.565	16.565	13.054
$F23^5$	5.701	6.444	6.444	5.701
$F24^5$	11.384	14.496	14.496	9.481
$F25^5$	11.717	11.717	11.717	15.719
$F26^6$	22.341	25.953	25.953	16.342
$F27^6$	20.623	19.745	20.623	17.519
$F28^6$	11.729	14.316	14.316	8.254
$F29^6$	13.415	14.142	14.142	9.894
$F30^6$	17.877	23.805	23.805	12.223
Total (% increase)	391.016 (21.56%)	425.076 (27.85%)	444.208 (30.95%)	306.707

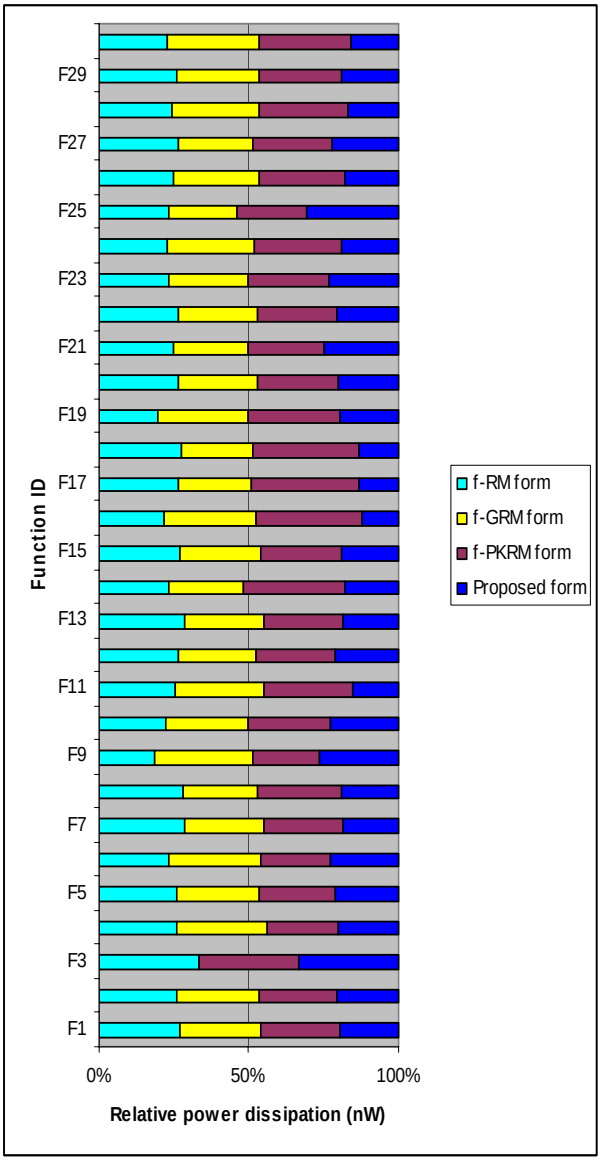

Fig. 4 Relative power consumption comparison of different methods

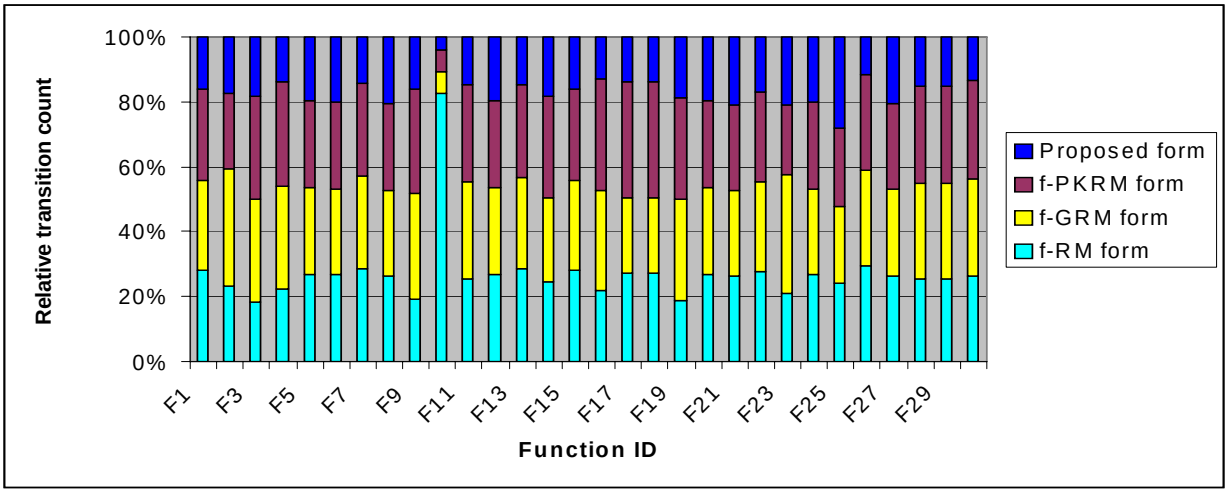

Fig. 5 Relative transition count pertaining to different methods

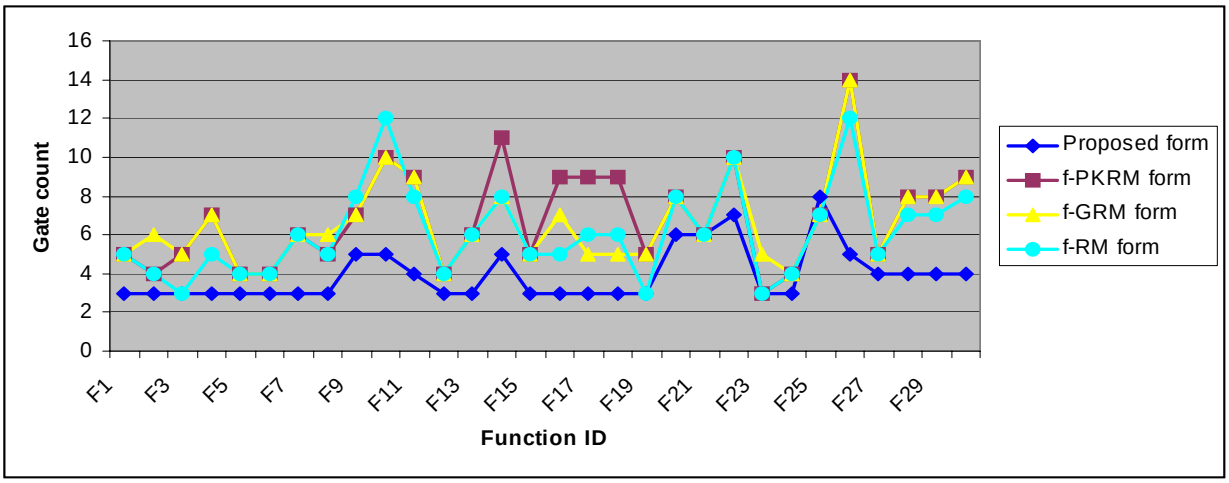

Fig. 6 Number of gates for different realizations

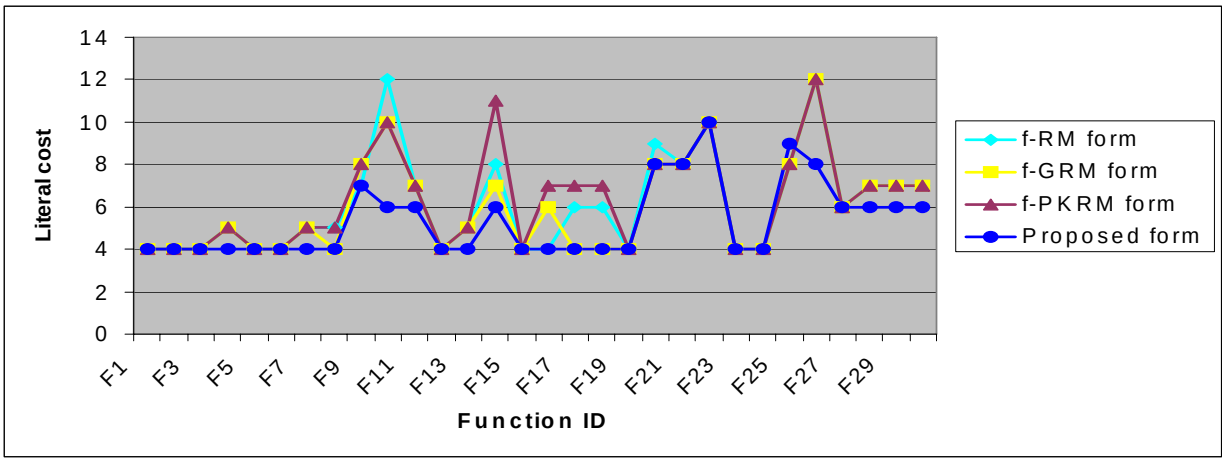

Fig. 7 Number of input literals required for the various implementations

VI. Conclusion and Scope for Future Work

The computational complexity associated with logic minimization has posed challenges since the beginning of the field in the early 60's; indeed solutions for some central questions, which remained elusive, have been obtained only within the last few years and others remain open [23]. This work addresses an important issue of practical relevance. A better technique of logic minimization and implementation, especially for a unique class of Boolean functions viz., CBF is discussed in this paper. We witnessed that the proposed systematic formulation for the logic simplification problem has also resulted in better solutions for some logic functions which are not strictly CBF. The significance of our contribution is substantiated by improvement achieved for a crucial design metric, in comparison with other realizations.

On the basis of computation of transition count, as an integer measure of switching activity, the proposed grouping methodology and synthesis mechanism predicted mean savings in power consumption over the other factored RM forms by 45.3%; while the simulation results enabled optimization in power consumption by 26.8%. We have also been successful in achieving decrease in gate count by 39.7% and reduction in the number of input literals required for implementation by around 12.9%.

For a number of 4-variable functions requiring 't' products [17], the average number of products for fixed polarity RM (FPRM) form, Kronecker RM form (KRM), SoP, PKRM form and ESOP are found to be 5.50, 4.73, 4.13, 3.84 and 3.66 respectively; while for a number of 5-variable functions requiring the same number of products [17], the average number of the products with respect to KRM, PKRM and ESOP is found to be 10.066, 6.976 and 6.162. Hence it becomes clear that among the AND-XOR type logical expressions, ESOP is the most general class, and requires the fewest products to represent given functions. Hence it would be worth pegging ESOPs as candidates for comparison with that of our proposed forms in terms of the critical design metric of power consumption. Also, we could consider analyzing multiple input and multiple output logic architectures to study the beneficial effects of sharing between the terms. A strategy to estimate the advantages of this proposed synthesis technique for other CMOS based realization styles is also in the pipeline. Another important step would be to develop a framework for the proposed synthesis scheme on the lines of a decision diagram structure.

Acknowledgment

The authors wish to thank Mrs. Sirisha Yellapragada for her help with the simulations and artwork.

References

[1] U. Narayanan, and C.L. Liu, "Low power logic synthesis for XOR based circuits", *Proc. of IEEE/ACM International Conf. on Computer Aided Design*, pp. 570-574, 1997.

[2] Sasan Iman, and Massoud Pedram, *Logic Synthesis for Low Power VLSI designs*, Springer-Verlag Publishing, Berlin Heidelberg, 1998.

[3] P. Balasubramanian, R. Chinnadurai, and M.R. Lakshmi Narayana, "Minimization of Dynamic Power Consumption in Digital CMOS Circuits by Logic Level Optimization", *WSEAS Trans. on Circuits and Systems*, vol. 4(4), pp. 257-266, April 2005.

[4] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev, *Logic synthesis of Asynchronous controllers and Interfaces*, Springer Series in Advanced Microelectronics, Springer-Verlag, Berlin Heidelberg, 2002.

[5] K. Nguyen, M. Perkowski, and N. Goldstein, "Palmini-Fats Boolean minimizer for Personal Computers", *Proc. of ACM/IEEE Design Automation Conference*, pp. 615-621, 1987.

[6] S. Kahramanli, and S. Tosun, "A novel essential prime implicant identification method for exact direct cover logic minimization", *Proc. of 2006 International Conference on Computer Design*, pp. 10-16, 2006.

[7] Giovanni De Micheli, *Synthesis and Optimization of Digital Circuits*, Mc-Graw Hill, New York, 1994.

[8] I.I. Zhegalkin, "O tekhnike vychisleniy predlozheniy v simvolicheskoy logike" (About a Technique of Computation of Expressions in Symbolic Logic), *Mat. Sb*., vol. 34, pp. 9-28, 1927.

[9] I.I. Zhegalkin, "Arifmetizatsiya simvolicheskoy logiki" (Arythmetization of Symbolic Logic), *Mat. Sb*., vol. 35, pp. 311-377, 1928.

[10] S.M. Reed, "A class of multiple-error-correcting codes and their decoding scheme", *IRE Trans. on Information Theory*, vol. PGIT-4, pp. 38-49, 1954.

[11] D.E. Muller, "Application of Boolean algebra to switching circuit design and to error detection", *IRE Trans. On Electron. And Comp*., vol. EC-3, pp. 6-12, 1954.

[12] D.H. Green, "Families of Reed-Muller Canonical forms", *International Journal of Electronics*, vol. 70(2), pp. 259-280, February 1991.

[13] T. Sasao, "An exact minimization of AND-EXOR expressions using BDDs", *Proc. of IFIP WG 10.5 Workshop on Applications of the Reed-Muller Expansion in Circuit Design*, pp. 91-98, 1993.

[14] 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), pp. 267-276, March 1999.

[15] C. Yang, M. Ciesielski, and V. Singhal, "BDS: A BDD-based logic optimization system", *Proc. of 37th ACM/IEEE Design Automation Conference*, pp. 92-97, 2000.

[16] A. Mishchenko, B. Steinbach, and M. Perkowski, "An algorithm for bi-decomposition of logic functions", *Proc. of 38th ACM/IEEE Design Automation Conference*, pp. 103-108, 2001.

[17] T. Sasao, *Logic Synthesis and Optimization*, Kluwer Academic Publishers, Massachusetts (USA), 1993.

[18] S. Chattopadhyay, S. Roy, and P.P. Chaudhuri, "KGPMIN: An Efficient Multilevel Multioutput AND-OR-XOR Minimizer", *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 16(3), pp. 257-265, March 1997.

[19] P. Balasubramanian, and C. Ardil, "Compact Binary Tree Representation of Logic Function with Enhanced Throughput", *International Journal of Computer, Information, and Systems Science, and Engineering*, vol. 1(2), pp. 90-96, 2007.

[20] B. Zeidman, "An Introduction to Application Specific Integrated Circuits", Tutorial: *Proc. of Embedded Systems Conference*, USA, 1999.

[21] Available: http://www.mentor.com

[22] C.-S. Ding, C.-Y. Tsui, and M. Pedram, "Gate-level Power Estimation using Tagged Probabilistic Simulation", *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 17(11), pp. 1099-1107, November 1998.

[23] Christopher Umans, Tiziano Villa, and Alberto L. Sangiovanni Vincentelli, "Complexity of Two-Level Logic Minimization", *IEEE Trans. On CAD of Integrated Circuits and Systems*, vol. 25(7), pp. 1230-1246, July 2006.

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

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

APPENDIX

TABLE VI
FUNCTION IDENTITY AND SPECIFICATION

Function ID	ON-set logic description
$F1^4$	{0,3,12,15}
$F2^4$	{4,7,8,11}
$F3^4$	{5,6,9,10}
$F4^4$	{6,7,8,9}
$F5^4$	{1,2,13,14}
$F6^4$	{0,3,5,6,9,10,12,15}
$F7^4$	{0,7,8,15}
$F8^4$	{5,6,7,9,10,11}
$F9^4$	{5,6,7,9,10}
$F10^4$	{0,3,7,8,12,15}
$F11^5$	{8,11,20,23}
$F12^5$	{1,2,5,6,25,26,29,30}
$F13^5$	{0,1,2,3,28,29,30,31}
$F14^5$	{1,2,9,10,13,14,17,18,20,21,29,30}
$F15^5$	{0,3,12,15,16,19,28,31}
$F16^5$	{0,1,2,3,5,6,9,10,12,13,14,15,16,17,18,19,21,22,25,26,28,29,30,31}
$F17^5$	{0,1,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,27,28,31}
$F18^5$	{0,3,4,7,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,27,28,31}
$F19^5$	{2,3,4,5,8,9,14,15,16,17,22,23,26,27,28,29}
$F20^5$	{4,5,10,11,20,21,26,27}
$F21^5$	{1,2,5,6,19,23,25,26,29,30}
$F22^5$	{5,10,21,26,28,29}
$F23^5$	{9,11,12,14,17,19,20,22}
$F24^5$	{0,1,6,7,10,11,12,13,18,19,20,21,24,25,30,31}
$F25^5$	{8,9,10,11,21,22}
$F26^6$	{7,8,55,56}
$F27^6$	{17,18,29,30,52,55,56,59}
$F28^6$	{17,18,21,22,41,42,45,46}
$F29^6$	{13,14,17,18,45,46,49,50}
$F30^6$	{3,4,11,12,51,52,59,60}

FX^n; F – Boolean Function, X – Identity, n – Number of primary inputs