Thirty Six Years of EXOR Logic Synthesis: Memoir

Tsutomu Sasao
Department of Computer Science
Meiji University, Kawasaki 214-8571, Japan

***Abstract*—This paper reviews the author's research on EXOR logic synthesis for the past 36 years. Topics include, AND-EXOR minimization, AND-OR-EXOR minimization, index generation functions, linear decomposition of index generation functions, and synthesis of multiple-output linear circuits.**

I. AND-EXOR Logic Circuit

A. Philipp W. Besslich

In the fall of 1983, at the library of Osaka University, I encountered a paper of Philipp W. Besslich from the University of Bremen, Germany [4]. In that paper, he detected properties of logic functions using spectrum transformation to simplify logical expressions with many variables. Since I was interested in the benchmark functions used in his paper, I wrote him a letter with a list of questions. After that, we began to exchange our knowledge.

About one year later, he sent me a letter that he wanted to visit Japan to work with me by a fellowship sponsored by the German Academic Exchange Service (DAAD) and the Japan Society for the Promotion of Science (JSPS). Soon, he successfully obtained the grant.

When he visited Osaka University in 1986, he brought a Pascal program that simplified AND-EXOR expressions. The algorithm first obtained the spectrum of the logic function to detect features of the function, and then simplified the AND-EXOR expression. At that time, I was not interested in AND-EXOR expressions, and also was ignorant about the spectral transformation of logic functions. However, after some discussions with him, I noticed that a simplification method for AND-OR expressions (MINI) can also be applied to AND-EXOR expressions. In a few days, I developed a much faster and more effective program than Besslich's.

In addition to the logic minimizer, Besslich was interested in the benchmark functions for AND-EXOR circuits. He proposed symmetric functions $SB(n,k)$ represented by EXORing all products of k variables out of n variables. For example, when $n=4$ and $k=2$, $SB(4,2)$ consists of $\binom{4}{2}=6$ logical products:

$$SB(4,2) = x_1x_2 \oplus x_1x_3 \oplus x_1x_4 \oplus x_2x_3 \oplus x_2x_4 \oplus x_3x_4.$$

Let $\tau(SB(n,k))$ be the minimum number of products in an AND-EXOR expression (ESOP) to realize an $SB(n,k)$ function. For example, a minimum ESOP for $SB(4,2)$[1] is

$$SB(4,2) = \bar{x}_1\bar{x}_2x_4 \oplus x_1x_2\bar{x}_4 \oplus x_2x_3 \oplus x_1x_3 \oplus \bar{x}_3x_4,$$

Thus $\tau(SB(4,2))=5$. Since

$$SB(n,k) = SB(n-1,k) \oplus x_nSB(n-1,k-1),$$

we have the relation:

$$\tau(SB(n,k)) \le \tau(SB(n-1,k)) + \tau(SB(n-1,k-1)).$$

TABLE 1.1
Numbers of products to realize $SB(n,k)$ functions

	k								
n	0	1	2	3	4	5	6	7	8
1	1	1							
2	1	2	1						
3	1	3	3	1					
4	1	4	5	4	1				
5	1	5	8	8	5	1			
6	1	6	11	12	11	6	1		
7	1	7	15	19	19	15	7	1	
8	1	8	20	30	30	30	20	8	1

With my new AND-EXOR minimizer, we obtained the number of products needed to represent $SB(n,k)$ functions by AND-EXOR expressions for different values of n and k. Table 1.1[2] shows the number of products to realize $SB(n,k)$ functions by an ESOP, which is obtained by EXMIN2[62].

Also, I minimized AND-EXOR expressions for all the representative classes of four variable functions.We found that AND-EXOR expressions (ESOPs) require many fewer products than AND-OR expressions (SOPs). We published this result as an IEEE TC paper [57], that is one of the most cited papers among our publications. Although Besslich stayed in Japan only for one month, the outcome was remarkable.

As for the $SB(n,k)$ function, many years later, I realized that the function was considered by Yasuo Komamiya in 1959. Yasuo Komamiya attended the Japanese multiple-valued logic workshop held in Shikanoshima Island in July 1990, where Norio Koda presented our work on the simplification of AND-EXOR expressions. After that, Komamiya sent me his book and papers. In fact, the book and paper contain the theory on $SB(n,k)$ functions. Unfortunately, I realized it after

[1]Even for this simple function, an exact minimum ESOP is difficult to derive by humans. In fact, Even-Kohavi-Paz falsely claimed that this function required 6 products in their IEEE TEC paper [17].

[2]For some entries, minimality are not proved yet.

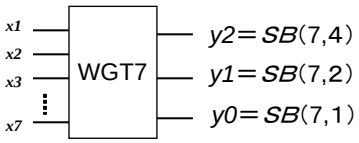

Fig. 1.1. WGT7.

Komamiya passed away[3]. His theory [31] is best illustrated by the following:

Example 1.1: $WGT7$ is a **bit counting circuit** [70] of 7 inputs. It has $x_1, x_2, \ldots, x_7$ as inputs and y_2, y_1, y_0 as outputs (Fig. 1.1). Then,

$$\sum_{i=1}^{7} x_i = 2^2 y_2 + 2^1 y_1 + 2^0 y_0,$$

where $+$ denotes an integer addition and concatenation is integer multiplication. Komamiya's theorem states that

$$\begin{aligned} y_2 &= SB(7,4) = \bigoplus_{i<j<k<l} x_i x_j x_k x_l, \\ y_1 &= SB(7,2) = \bigoplus_{i<j} x_i x_j, \\ y_0 &= SB(7,1) = \bigoplus_{i=1}^{7} x_i. \end{aligned}$$

$WGT7$ is also denoted by $rd73$, and is often used as a benchmark function of logic synthesis. ■

After Besslich returned to Germany, I developed an iterative AND-EXOR minimizer called EXMIN. This can treat multiple-valued input logic functions, and can simplify AND-EXOR PLAs with input decoders. With this program, I designed various benchmark functions, and showed that AND-EXOR networks can be much simpler than corresponding AND-OR networks for arithmetic circuits. Table 1.2 shows the number of products to realize certain arithmetic functions[4], which is obtained by EXMIN2, an updated version of [62].

TABLE 1.2
NUMBERS OF PRODUCTS TO REALIZE ARITHMETIC FUNCTIONS

	AND-OR		AND-EXOR	
	1bit	2bit	1bit	2bit
ADR4	75	17	31	11
MLP4	121	86	62	49
WGT8	255	54	59	24

B. Reed-Muller Workshop

In April of 1988, I moved to the newly founded campus of Kyushu Institute of Technology (KIT) in Iizuka City, Fukuoka, Japan. As for the simplification of AND-EXOR expressions, Norio Koda of Tokuyama Technical College spent a sabbatical year in our group. We published several papers on the upper and lower bounds on the number of products in AND-EXOR expressions (ESOPs). After that, he became a Ph. D. student of mine, and in 1996, he received his Ph.D. degree.

The impact of 1990 IEEE TC paper with Besslich [57] was rather high. In 1993, Wolfgang Rosenstiel from Germany informed me that he was organizing a Reed-Muller workshop in Hamburg. So, I attended the workshop with Norio Koda. The attendance at the Reed-Muller 1993 workshop was small, but many key persons on decision diagrams and EXOR logic gathered.

At the end of the workshop, we decided that the second workshop would be held in 1995 in Makuhari, Chiba, Japan. The organizers were myself and Masahiro Fujita[5] of Fujitsu America Laboratory. We published a monograph [66] in 1996, in addition to the workshop proceedings. So, we invited authors whose papers become be chapters of the book. The workshop site was provided by Fujitsu, and I obtained several grants to support the travel expenses for overseas attendees. We had many participants, and the workshop was quite successful.

The 9th workshop was held in 2009 in Naha, Okinawa, Japan. In this time, we published a monograph on Boolean function [74], in addition to the proceedings. The 11th workshop was held in 2013 in Toyama, Japan. In this time, we published a monograph on ZDDs [82].

II. AND-OR-EXOR Logic Circuit

In CMOS realizations, a two-input EXOR gate requires at least 6 transistors (Fig. 2.1), while a two-input NAND gate requires only four transistors. Also, the EXOR gate is slower than the NAND gate. For this reason, EXOR logic circuits are not so popular in industry. To overcome this disadvantage, I developed the architecture shown in Fig. 2.2.

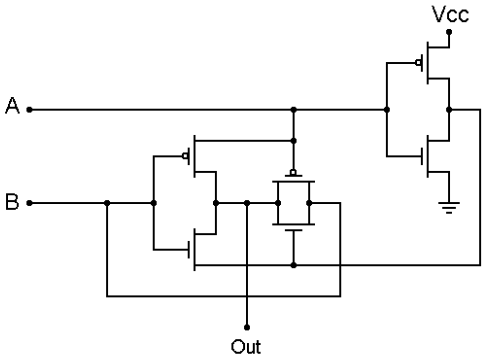

Fig. 2.1. CMOS Realization of an EXOR gate

In this circuit, the function is represented by

$$f = F_1 \oplus F_2,$$

where F_1 and F_2 are realized by SOPs. Note that only one two-input EXOR gate is required in the output, but its logical

[3]Since Komamiya's work was not well known, Radomir S. Stankovic and I wrote papers [96], [98], [99] to promote Komamiya's work.

[4]For some entries, minimality are not proved yet.

[5]Currently, University of Tokyo.

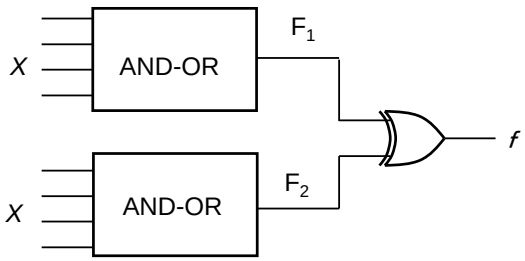

Fig. 2.2. AND-OR-EXOR Realization[65]

power is amazing. Consider the function:

$$f = (x_1 \vee x_2)(x_3 \vee x_4)(x_5 \vee x_6)(x_7 \vee x_8).$$

When f is implemented by an AND-OR circuit, $2^4 = 16$ products are necessary. However, if we implement $\bar{f}$, the complement of f, then only four products is necessary as follows:

$$\bar{f} = \bar{x}_1\bar{x}_2 \vee \bar{x}_3\bar{x}_4 \vee \bar{x}_5\bar{x}_6 \vee \bar{x}_7\bar{x}_8.$$

Thus, f is represented as

$$f = (\bar{x}_1\bar{x}_2 \vee \bar{x}_3\bar{x}_4 \vee \bar{x}_5\bar{x}_6 \vee \bar{x}_7\bar{x}_8) \oplus 1.$$

In Fig. 2.2, realize $F_1 = \bar{f}$ and $F_2 = 1$. Then, we need only $4+1 = 5$ products[6]. In general, F_2 can be any logic function.

Table 2.1 compares the number of 4-variable functions that require t products [65]. AND-OR circuits (SOP) require up to 8 products, and require on the average, 4.13 products; AND-EXOR circuits (ESOP) require up to 6 products, and require on the average, 3.66 products; and, AND-OR-EXOR requires up to 5 products, and require on the average, 3.62 products.

Also, AND-OR-EXOR circuits efficiently realizes adders. Table 2.2 compares the number of products to realize n bit adders. AND-OR-EXOR with 2-bit decoders requires only $n+1$ EXOR gates, are as efficient as AND-EXOR with 2-bit decoders.

In 1998, Debatosh Debnath[7] from Bangladesh obtained his Ph. D. degree on the research of *AND-OR-EXOR three-level logic circuits*.

TABLE 2.1
NUMBER OF 4-VARIABLE FUNCTIONS THAT REQUIRES t PRODUCTS[65]

t	AND-OR	AND-EXOR	AND-OR-EXOR
0	1	1	1
1	81	81	81
2	1804	2268	2316
3	13472	21744	22896
4	28904	37530	37634
5	17032	3888	2608
6	3704	24	0
7	512	0	0
8	26	0	0
m	4.13	3.66	3.62

[6]This is a kind of *output phase optimization* [70].
[7]Currently, Oakland University.

TABLE 2.2
NUMBER OF PRODUCTS TO REALIZE n-BIT ADDERS[65]

Architecture	# of products	$n = 4$
AND-OR	$6 \cdot 2^n - 4n - 5$	75
AND-EXOR	$2^{n+1} - 1$	31
AND-OR with 2-bit decoders	$n^2 + 1$	17
AND-EXOR with 2-bit decoders	$(n^2 + n + 2)/2$	11
AND-OR-EXOR with 2-bit decoders	$(n^2 + n + 2)/2$	11

TABLE 3.1
EXAMPLE OF A REGISTERED VECTOR TABLE

Registered Vector						Index
x_1	x_2	x_3	x_4	x_5	x_6	f
0	0	0	0	1	0	1
0	1	0	0	1	0	2
0	0	1	0	1	0	3
0	0	1	1	1	0	4
0	0	0	0	0	1	5

III. INDEX GENERATION FUNCTION

In 2002, the Cluster Project (the first stage) of the MEXT (Ministry of Education, Culture, Sports, Science and Technology of Japan) started. In the MEXT Cluster Project, we received a large amount of research funds for many years. The final goal was to develop a commercial product by doing joint research with industry. At the end of the first stage of the Cluster Project, Masayuki Chiba of YAMAHA Corporation showed me a concept of **index generator** as the key device in the network hardware.

An **index generation function** is an integer valued function:

$$\{0,1\}^n \to \{0,1,\ldots,k\}.$$

For example, in the case of $n = 6$ and $k = 5$, the function maps $k = 5$ different two-valued vectors into $k = 5$ distinct integers as shown by the example in Table 3.1.

An index generation function can be directly implemented by a Content Addressable Memory (CAM), but CAMs dissipate much power and are expensive.

Thus, I invented a better realization than CAM for this function. The **Index Generation Unit** (IGU) shown in Fig. 3.1 can realize index generation functions quite efficiently. The design problem of an IGU can be formulated as a minimization problem of the variables for an incompletely specified function. This method is quite efficient, and we can easily implement a practical pattern matching network by using an FPGA and memories. An IGU can easily implement a circuit for $k > 10^6$. Furthermore, if we use a linear decomposition, we can drastically reduce the size of the memory. With this idea, we published many papers. Especially, there are interesting mathematical problems, and it became a fruitful research endeavor.

In 2011, I published a monograph [75] that summarizes index generation functions.

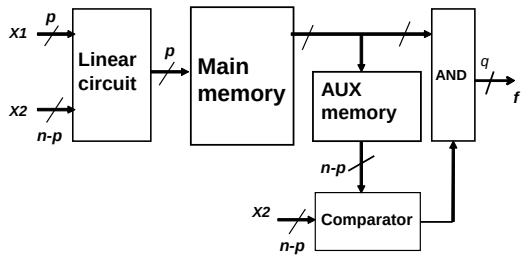

Fig. 3.1. Index Generation Unit (IGU).

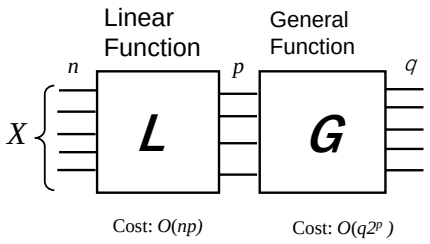

Fig. 4.1. Linear Decomposition

IV. Linear Decomposition

In March 2013, I retired from Kyushu Institute of Technology, Fukuoka, Japan, and from April 2013, become a professor of Meiji University, Kanagawa, Japan. Mochinori Goto [20], and Yasuo Komamiya [31], who developed the Japanese first relay computer in 1952 [99], were professors there. Although there were no Ph.D students, some M.S. students are very good, and with the help of Yukihiro Iguchi, I could continue my research work. Fortunately, I could obtain research grants from JSPS all the years at Meiji. The decline of Japanese LSI industry reduced students working for logic synthesis. So, I sought new applications: logic synthesis for pattern matching and data mining.

A **linear decomposition**[8] shown in Fig. 4.1 was first developed by Nechiporuk in 1958 [46], [97]. In the linear decomposition, L realizes linear functions, while G realizes an arbitary function.

Definition 4.1: [23], [87]. An **affine transformation** of the input variables $x_1, x_2, \ldots, x_n$ is defined as

$$\begin{aligned} y_1 &= c_{10} \oplus c_{11}x_1 \oplus c_{12}x_2 \oplus c_{13}x_3 \oplus \ldots \oplus c_{1n}x_n, \\ y_2 &= c_{20} \oplus c_{21}x_1 \oplus c_{22}x_2 \oplus c_{23}x_3 \oplus \ldots \oplus c_{2n}x_n, \\ &\vdots \\ y_n &= c_{n0} \oplus c_{n1}x_1 \oplus c_{n2}x_2 \oplus c_{n3}x_3 \oplus \ldots \oplus c_{nn}x_n, \end{aligned}$$

where $c_{ij} \in \{0, 1\}$. Two logic functions f and g are **A-equivalent** (affine-equivalent), denoted by $f \overset{A}{\sim} g$ if g is obtained from f by an **invertible affine transformation** of the input variables. When $c_{i0} = 0$ for all $i = 1, \ldots, n$, then the transformation is **linear**.

[8]Linear transform and affine transform are often confused in switching theory [70]. Nechiporuk considered *affine* transformation, but called it *linear*. Functions y_i in Definition 4.1 are affine. When $c_{i0} = 0$, they are linear.

TABLE 4.1
Numbers of Equivalence Classes of Logic Functions [87], [23]

Class	$n=1$	$n=2$	$n=3$	$n=4$	$n=5$
ALL	4	16	256	65,536	4,294,967,296
L	4	8	20	92	2,744
A	3	5	10	32	382

TABLE 4.2
Number of Variables to Represent m-out-of-20 Code to Binary Converter: Best Solutions ($n = 20$) [86].

		t : Max compound degree					
m	k	1	2	3	4	5	6
1	20	19	13	10	8	7	6
2	190	19	14	12	10	9	9
3	1140	19	16	13	12	12	*11
4	4845	19	16	15	15	15	15

* denotes optimal solution proved by theorem.

Nechiporuk enumerated the **affine equivalence classes** of logic functions for up to $n = 5$ variables. Also, he obtained the minimum circuit for each representative function for $n = 4$. Assume that linear transformation is freely available. Then, one can select the most economical circuit (function), among the logic functions in the equivalence class. Table 4.1 compares the number of different equivalence classes. L denotes the linear transformation, and A denotes the affine transformation. The numbers of affine equivalence classes are denoted by A, and they are 32 for $n = 4$, and 382 for $n = 5$.

Nechiporuk used linear decomposition to realize **completely specified logic functions**. On the other hand, I used linear decomposition to reduce the number of variables for **incompletely specified index generation functions** [72]. I also assume that the circuit is **programmable**. Thus, L is implemented by EXOR gates, multiplexers and registers [75], while G is implemented by a memory. The cost is measured by the number of bits to store the functions. Thus, the cost for L is $O(np)$, while the cost for G is $O(q2^p)$, where $q \leq p \leq n$ and $q = \lceil \log_2 k \rceil$. In this case, the minimization of the number of the variables p for the memory is the key issue of the design. After spending more than 10 years, we had the following:

Theorem 4.1: [89] Any incompletely specified index generation function with k registered vectors, can be represented with at most $p = \lceil 2 \log_2 k \rceil - 2$ variables, when a linear decomposition is used.

The number of variables to represent an m-out-of-20 code to binary number converter is investigated for different values of compound degrees t, and for different values of m. The original number of variables is $n = 20$. The number of registered vectors is $k = \binom{20}{m}$, and the function requires at least $q = \lceil \log_2(k+1) \rceil$ variables. Table 4.2 compares the results[9]. When the compound degree is one ($t = 1$), all the converters required 19 variables. For $m = 1, 2$ and 3, with the increase of the compound degree t, the necessary number

[9]For some entries, minimality are not proved yet.

of variables decreased. The entry with * denotes an optimum solution proved by the lower bound $\lceil \log_2(k+1) \rceil$.

As shown in Table 4.2, linear decompositions of index generation functions are quite effective.

V. Multiple-output Linear Circuits

The index generation unit (IGU) contains a linear part. So, I considered a design method for linear circuits consisting of fan-in limited EXOR gates [88].

Example 5.1: Consider the following linear functions:

$$\begin{aligned} y_1 &= x_1 \oplus x_2 \oplus x_3 \oplus x_4 \\ y_2 &= x_1 \oplus x_2 \oplus x_3 \oplus x_5 \\ y_3 &= x_1 \oplus x_2 \oplus x_3 \oplus x_6 \\ y_4 &= x_1 \oplus x_2 \oplus x_3 \oplus x_7 \end{aligned}$$

The **straightforward representation** requires $4 \times 4 = 16$ literals. However, by extracting the **common clause** $u = x_1 \oplus x_2 \oplus x_3$, we have the following **extracted representation**:

$$\begin{aligned} u &= x_1 \oplus x_2 \oplus x_3 \\ y_1 &= u \oplus x_4 \\ y_2 &= u \oplus x_5 \\ y_3 &= u \oplus x_6 \\ y_4 &= u \oplus x_7 \end{aligned}$$

Note that the extracted representation requires $3 + 4 \times 2 = 11$ literals. When the fan-in limitation of EXOR gates is $p = 3$, the straightforward realization (Fig. 5.1) requires 8 gates, instead of 4 gates, since to implement a 4-input EXOR gate, we need two EXOR gates. On the other hand, the extracted realization (Fig. 5.2) requires only 5 gates. ■

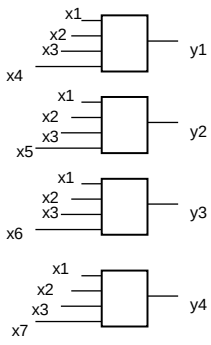

Fig. 5.1. Straightforward Realization

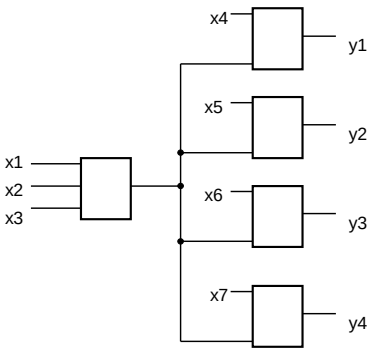

Fig. 5.2. Extracted Realization

An n-input m-output linear circuit can be designed using a **characteristic function** of two variables, where the first variable takes n values, while the second variable takes m values.

Fig. 5.3 shows the map for the straightforward realization, while Fig. 5.4 shows the map for the extracted realization. Each loop corresponds to an EXOR gate. It shows the input variables and output connections. For example, the top loop in Fig. 5.3 corresponds to the top gate in Fig. 5.1. The inputs of the gate are x_1, x_2, x_3 and x_4, and the output is connected to y_1. The large loop in Fig. 5.4 corresponds to the extracted gate in Fig. 5.2. The inputs of the gate are x_1, x_2 and x_3, and the output is connected to y_1, y_2, y_3 and y_4.

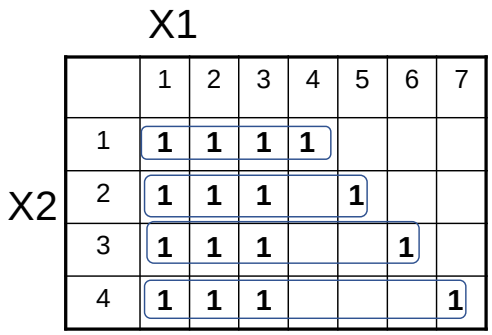

Fig. 5.3. Map for Straightforward Realization

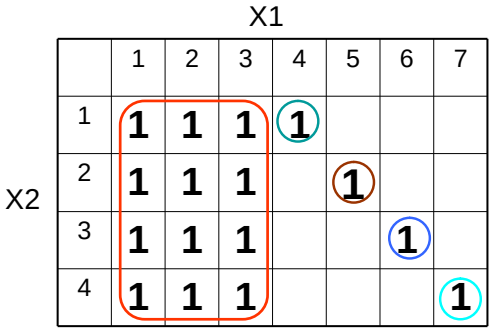

Fig. 5.4. Map for Extracted Realization.

Example 5.2: Consider the function that appears in Example 5.1. Fig. 5.5 shows the map for the non-disjoint EXOR cover. In this map, the columns 4 and 5 are interchanged. Note that the cells covered by two loops cancel each other. Thus, this map represents the same characteristic functions as Figs. 5.3 and 5.4. However, we have a different formulas:

$$\begin{aligned} y_2 &= x_1 \oplus x_2 \oplus x_3 \oplus x_5 \\ y_1 &= y_2 \oplus x_4 \oplus x_5 \\ y_3 &= y_2 \oplus x_5 \oplus x_6 \\ y_4 &= y_2 \oplus x_5 \oplus x_7 \end{aligned}$$

In this case, the number of the connections is 13, which is larger than that of the extracted solution shown in Fig. 5.4. However, when the fan-in limitation is $p = 4$, it requires only four EXOR gates, as shown in Fig. 5.6. ■

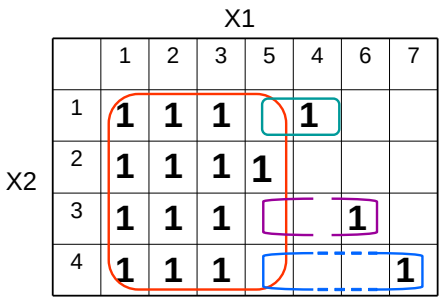

Fig. 5.5. Map for non-disjoint EXOR cover-based realization.

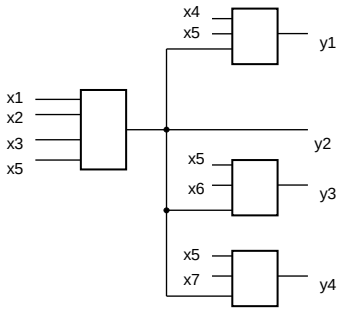

Fig. 5.6. Non-disjoint EXOR cover-based realization.

As shown in the examples, the design of a linear circuit can be done on a map of a multi-valued two-variable characteristic function. A multi-valued ESOP minimizer EXMIN2[62] was used to find these solutions. Viva EXOR logic synthesis!

VI. Concluding Remarks

This paper briefly summarizes the authors research on EXOR logic synthesis for the past 36 years. Part of the material in this paper was taken from [70], [79], [86], [88].

1) AND-EXORs (ESOPs) realize symmetric functions and adders more efficiently than AND-ORs (SOPs). An arbitrary symmetric functions of $n = 2r$ variables can be represented by an ESOP with at most $2 \cdot 3^{r-1}$ products, while an SOP requires up to 2^{n-1} products.
2) Heuristic ESOP minimizers are available, however an exact minimum ESOP is hard to obtain. Unlike SOP, lower bounds on the number of products in ESOP are hard to obtain. Thus, the minimality of solutions of some of the entries in Tables 1.1 and 1.2 are still not proved yet.
3) In CMOS technology, an EXOR gate is more expensive than a NAND (NOR) gate. However, in FPGAs, the costs are the same.
4) AND-OR-EXOR is promising. One method is to represent a function f by an ESOP, and partition the products into r groups so that each group is represented as a disjoint SOP [60], and represent f by

$$f = F_1 \oplus F_2 \oplus \cdots \oplus F_r.$$

5) Linear decomposition is useful for index generation functions. Heuristic minimizers for index generation functions are available. However, exact minimum solutions are still hard to obtain. Minimality of solutions of some of entries in Table 4.2 are still not proved yet.
6) A multiple-output linear circuit can be design by its characteristic function. However, the exact minimum circuit is hard to derive.
7) Logic synthesis using linear decomposition requires the knowledge of affine equivalence classes.
8) EXOR logic synthesis is more complicated than AND-OR logic synthesis. To develop a new tool, knowledge of group theory and spectral transform are helpful. To find a useful theorem, many years are necessary.

Acknowledgments

Mitch Thornton invited me for the talk, and motivated to write this paper. Jon T. Butler carefully read the drafts many times, and improved presentation. This research is partly supported by Grant in Aid for Scientific Research of the Japan Society for the Promotion of Science (JSPS).

Bibliographical Notes

Classification of AND-EXORs [10], [21], [35], [57], [61], [67]; PPRMs [31], [37], [53], [108]; FPRM minimization [5], [6], [105]; PSDKRO minimization [63]; GRM minimization [11], [68], [69]; ESOP heuristic minimization [8], [17], [18], [24], [34], [54], [60], [62], [56], [94]; ESOP exact minimization [25], [26], [30], [35], [49], [50], [64], [100]; other AND-EXOR expressions [107]; EXOR multi-level logic networks [101]; AND-OR-EXOR networks [12], [13], [14], [15], [65]; AND-EXOR test [51], [52], [69], [55], [93]; EXOR-AND-OR expressions [9], [91], [33], [3]; multi-valued EXORs [16]; multi-output linear circuit [7], [19], [88]; minimization of variables for index generation functions by linear decomposition [1], [2], [38], [39], [40], [41], [42], [71], [72], [73], [76], [77], [78], [80], [81], [83], [84], [85], [85], [89], [90]; equivalence class [22], [23], [32], [47], [87]; implementation of IGUs [43], [44], [45]; and spectral techniques [27], [28], [101], [95], [102], [104], [106].

Some publication names are abbreviated as follows: **IEEE** for the Institute of Electrical and Electronics Engineers, **IRE** for The Institute for Radio Engineers, **IEE** for The Institute of Electrical Engineers (United Kingdom), **IEICE** for The Institute of Electronics, Information and Communication Engineers (Japan), **TX** for Transactions on X, **(E)C** for (Electronic) Computers, **CAD** for Computer-Aided Design of Integrated Circuits and Systems, **IT** for Information Theory, **CE** for Communication and Electronics, **CS** for Circuits and Systems, **DAC** for ACM/IEEE Design Automation Conference, **ISMVL** for IEEE International Symposium on Multiple-Valued Logic, **ICCAD** for IEEE International Conference on Computer Aided Design, **ICCD** for IEEE International Conference on Computer Design, **ISCAS** for IEEE International Symposium on Circuits and Systems, **IWLS** for International Workshop

on Logic and Synthesis. **ASPDAC** for Asia-Pacific Design Automation Conference, **RM** for Reed-Muller Workshop.

REFERENCES

[1] J. Astola, P. Astola, R. Stankovic and I. Tabus, "An algebraic approach to reducing the number of variables of incompletely defined discrete functions," *ISMVL*, Sapporo, Japan, pp. 107-112, May 2016. Also, in *Journal of Multiple-Valued Logic and Soft Computing*, Vol. 31, No. 3, 2018, pp. 239-253.

[2] J. T. Astola, P. Astola, R. S. Stankovic, and I. Tabus, "Algebraic and combinatorial methods for reducing the number of variables of partially defined discrete functions," *ISMVL*, Novi Sad, Serbia, pp. 167-172, May 2017.

[3] A. Bernasconi, V. Ciriani, F. Luccio, and L. Pagli, "Three-level logic minimization based on function regularities," *IEEETCAD*, Vol.22, No. 8, pp.1005-1016, Aug. 2003.

[4] Ph. W. Besslich and P. Pichlbauer, "Fast transform procedure for the generation of near minimal covers of Boolean functions," *IEE Proc.*, Vol. 128, Part E, No. 6, pp. 250-254, Nov. 1981.

[5] Ph. W. Besslich, "Efficient computer method for ExOR logic design," *IEE Proc.*, Vol. 130, Part E, pp. 203-206, Nov. 1983.

[6] G. Bioul, M. Davio, and J. P. Deschamps, "Minimization of ring-sum expansions of Boolean functions," *Philips Res. Rpts.*, Vol. 28, pp. 17-36, 1973.

[7] J. Boyar1, P. Matthews, and R. Peralta, "On the shortest linear straight-line program for computing linear forms," MFCS 2008, LNCS 5162, pp. 168-179, 2008.

[8] D. Brand and T. Sasao, "Minimization of AND-EXOR expressions using rewriting rules," *IEEE TC*, Vol. 42, No. 5, pp. 568-576, May 1993.

[9] V. Ciriani,"Synthesis of SPP three-level logic networks using affine spaces," *IEEE TCAD*, Vol. 22, No. 10, pp. 1310-1323, Oct. 2003.

[10] M. Davio, J-P Deschamps, and A. Thayse, *Discrete and Switching Functions*, McGraw-Hill International, 1978.

[11] D. Debnath and T. Sasao, "GRMIN2: A heuristic simplification algorithm for generalized Reed-Muller expressions," *IEE Proceedings-Computers and Digital Techniques*, Vol. 143, No. 6, pp. 376-384, Nov. 1996.

[12] D. Debnath and T. Sasao, "Minimization of AND-OR-EXOR three-level networks with AND gate sharing," *IEICE Trans. Information and Systems*, Vol. E80-D, No. 10, pp. 1001-1008, Oct. 1997.

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

[14] E. V. Dubrova, D. M. Miller, and J. C. Muzio, "Upper bounds on the number of products in AND-OR-EXOR expansion of logic functions," *Electronics Letters*, Vol. 31, No. 7, pp. 541-542, March 1995.

[15] E. Dubrova, D. M. Miller, and J. Muzio, "AOXMIN-MV: A heuristic algorithm for AND-OR-XOR minimization,"*RM-1999*,

[16] G. Dueck and D. M. Miller, "A 4-valued PLA using the MOD SUM," *ISMVL*, pp. 232-240, May 1986.

[17] S. Even, I. Kohavi, and A. Paz, "On minimal modulo-2 sum of products for switching functions," *IEEE TEC*, Vol. EC-16, pp. 671-674, Oct. 1967.

[18] H. Fleisher, M. Tarvel, and J. Yeager, "A computer algorithm for minimizing Reed-Muller canonical forms," *IEEE TC*, Vol. C-36, No. 2, Feb. 1987.

[19] C. Fuhs and P. Schneider-Kamp, "Synthesizing shortest linear straight-line programs over GF(2) using SAT," SAT-2010, pp.71-88, 2010.

[20] M. Goto, "Applications of logical equations to the theory of relay contact networks," *Electric Soc. of Japan* (in Japanese), Vol. 69, p. 125, April, 1949.

[21] D. Green, *Modern Logic Design*, Addison-Wesley Publishing Company, 1986.

[22] M. A. Harrison, *Introduction to Switching and Automata Theory*, McGraw-Hill, 1965.

[23] M. A. Harrison, "Counting theorems and their applications to classification of switching functions," in A. Mukhopadhyay (ed.), *Recent Developments in Switching Theory*, Academic Press, New York, 1971.

[24] M. Helliwell and M. Perkowski, "A fast algorithm to minimize multi-output mixed-polarity generalized Reed-Muller forms," *DAC*, pp. 427-432, 1988.

[25] T. Hirayama, Y. Nishitani and T. Sato, " A Faster Algorithm of Minimizing AND-EXOR Expressions," *IEICE* on Fundamentals, Vol. E85-A No. 12 pp. 2708-2714, Dec. 2002.

[26] T. Hirayama and Y. Nishitani, "Exact minimization of and-exor expressions of practical benchmark functions," *Journal of Circuits, Systems and Computers*, Vol. 18, No. 3, pp. 465-486, 2009.

[27] S. L. Hurst, D. M. Miller, and J. C. Muzio, *Spectral Techniques in Digital Logic*, Academic Press, London, 1985.

[28] M. G. Karpovsky (ed.), *Spectral Techniques and Fault Detection*, Academic Press, 1985.

[29] O. Keren and I. Levin, "Linearization of multi-output logic functions by ordering of the autocorrelation values," *FACTA UNIVERSITATIS (NIS)*, Vol. 20, no. 3, December 2007, pp. 479-498.

[30] N. Koda and T. Sasao, "Four-variable AND-EXOR minimum expressions and their properties" *(in Japanese), IEICE Trans.*, Vol. J74-D-1, No. 11, pp.765-773, Nov., 1991. Also in *System Computer of Japan* (U.S.A.), Vol. 23, No. 10, pp. 27-41, 1992 (English translation).

[31] Y. Komamiya, *Theory of Computing Networks*, Researches of ETL, Sept. 1959.

[32] R. J. Lechner, "Harmonic analysis of switching functions," in A. Mukhopadhyay (ed.), *Recent Developments in Switching Theory*, Academic Press, New York, 1971.

[33] F. Luccio and L. Pagli, "On a new Boolean function with applications," *IEEETC* vol. 48, pp. 296-310, Mar. 1999.

[34] A. Mishchenko and M. Perkowski, "Fast heuristic minimization of exclusive-sums-of-products," RM-2001, August 2001

[35] A. Mukhopadhyay and G. Schmitz, "Minimization of exclusive OR and logical equivalence of switching circuits," *IEEE TC*, Vol. C-19, No. 2, pp. 132-140, Feb. 1970.

[36] A. Mukhopadhyay (ed.), *Recent Developments in Switching Theory*, Academic Press, New York, 1971.

[37] D. E. Muller, "Application of Boolean algebra to switching circuit design and to error detection," *IRE TEC*, Vol. EC-3, No. 3, pp. 6-12, Sept. 1954.

[38] S. Nagayama, T. Sasao, and J. T. Butler, "An efficient heuristic algorithm for linear decomposition of index generation functions," *ISMVL*, May 2016, pp. 96-101.

[39] S. Nagayama, T. Sasao and J. T. Butler, "An exact optimization algorithm for linear decomposition of index generation functions," *ISMVL*, Novi Sad, Serbia, May 23, 2017, pp. 161-166.

[40] S. Nagayama, T. Sasao, and J. T. Butler, "A balanced decision tree based heuristic for linear decomposition of index generation functions," *IEICE Trans. Inf. and Syst.* Vol. E100, No. 88, pp. 1583-1591, Aug. 2017.

[41] S. Nagayama, T. Sasao and J. Butler, "An exact optimization method using ZDDs for linear decomposition of index generation function," *ISMVL*, May 16-18, 2018, Linz, Austria.

[42] S. Nagayama, T. Sasao and J. Butler, "A dynamic programming based method for optimum linear decomposition of index generation functions," *ISMVL*, May 21-23, 2019, Fredericton, Canada.

[43] H. Nakahara, T. Sasao, and M. Matsuura, "A virus scanning engine using an MPU and an IGU based on row-shift decomposition," *IEICE Transactions on Information and Systems*, Vol. E96-D, No.8, Aug. 2013, pp. 1667-1675.

[44] H. Nakahara,T. Sasao and M. Matsuura, "An update method for a CAM emulator using an LUT cascade based on an EVMDD(k)," *ISMVL*, Bremen, Germany, May 19-22, 2014, pp. 1-6.

[45] H. Nakahara, T. Sasao, M. Matsuura, H. Iwamoto, and Y. Terao, " A memory-based IPv6 lookup architecture using parallel index generation units," *IEICE Trans. Inf. and Syst.* Vol. E98-D, No. 2, pp. 262-271, Feb. 2015.

[46] E. I. Nechiporuk, "On the synthesis of networks using linear transformations of variables," *Dokl. AN SSSR*, vol. 123, no. 4, pp. 610-612, Dec. 1958 (in Russian).

[47] I. Ninomiya, "A study of the structure of Boolean functions and its application to the synthesis of switching circuits," *Men. Fac. Eng., Nagoya Univ.*, Vol. 13, Ph.D. Thesis, Univ. Tokyo, 1961. (Tables reprinted in [22]).

[48] T. Nozaki, *Switching Theory* (in Japanese), Kyoritsu Shuppan, 1972.

[49] G. Papakonstantinou, "Minimization of modulo-2 sum of products," *IEEE TC*, C-28, pp. 163-167, 1979.

[50] M. Perkowski and M. Chrzanowska-Jeske, "An exact algorithm to minimize mixed-radix exclusive sums of products for incompletely specified Boolean functions," *ISCAS*, pp. 1652-1655, June 1990.

[51] D. K. Pradhan, "Universal test sets for multiple fault detection in AND-EXOR arrays," *IEEE TC*, Vol. C-27, No. 2, pp. 181-187, Feb. 1978.

[52] S. M. Reddy, "Easily testable realization for logic functions," *IEEE TC*, Vol. C-21, No. 11, pp. 1083-1088, Nov. 1972.

[53] I. S. Reed, "A class of multiple-error-correcting codes and the decoding scheme," *IRE TIT*, PGIT-4, pp. 38-49, 1954.

[54] J. P. Robinson and Chia-Lung Yeh, "A method for modulo-2 minimization," *IEEE TC*, Vol. C-31, No. 8, pp. 800-801, Aug. 1982.

[55] K. K. Saluja and S. M. Reddy, "Fault detecting test sets for Reed-Muller canonic networks," *IEEE TC*, Vol. C-24, No. 10, pp. 995-998, Oct. 1975.

[56] K. K. Saluja and E. H. Ong, "Minimization of Reed-Muller canonic expansion," *IEEE TC*, Vol. C-28, No. 7, pp. 535-537, July 1979.

[57] T. Sasao and Ph. Besslich, "On the complexity of MOD-2 Sum PLA's," *IEEE TC*, Vol. 39, No. 2, pp. 262-266, Feb. 1990.

[58] T. Sasao, " EXMIN: A simplification algorithm for exclusive-or sum-of-products expressions for multiple-valued input two-valued output functions," *ISMVL*, Charlotte, North Carolina, pp. 128-135, May 1990.

[59] T. Sasao (ed.), *Logic Synthesis and Optimization*, Kluwer Academic Publishers, Boston, 1993.

[60] T. Sasao, "Logic synthesis with EXOR logic gates," Chapter 12 in [59].

[61] T. Sasao, "AND-EXOR expressions and their optimization," Chapter 13 in [59].

[62] T. Sasao, "EXMIN2: A simplification algorithm for exclusive-OR-sum-of-products expressions for multiple-valued input two-valued output functions," *IEEE TCAD*, Vol. 12, No. 5, pp. 621-632, May 1993.

[63] T. Sasao, "Optimization of pseudo-Kronecker expressions using multiple-place decision diagrams," *IEICE Transactions on Information and Systems*, Vol. E76-D, No. 5, pp. 562-570, May 1993.

[64] T. Sasao, "An exact minimization of AND-EXOR expressions using reduced covering functions," *Proc. of the Synthesis and Simulation Meeting and International Interchange (SASIMI'93)*, pp. 46-53, Oct. 1993.

[65] T. Sasao, "A design method for AND-OR-EXOR three-level networks," *IWLS*, pp. 8:11-8:20, Tahoe City, California, May 1995.

[66] T. Sasao and M. Fujita (ed.), *Representation of Discrete Functions*, Kluwer Academic Publishers, Boston, 1996.

[67] T. Sasao, "Representation of logic functions using EXOR operators," Chapter 2 in [66].

[68] T. Sasao and D. Debnath, "Generalized Reed-Muller expressions: Complexity and an exact minimization algorithm," *IEICE Transactions Fundamentals*, Vol. E79-A, No. 12, pp. 2123-2130, Dec. 1996.

[69] T. Sasao, "Easily testable realizations for generalized Reed-Muller expressions," *IEEE TC*, Vol. 46, No. 6, pp. 709-716, June 1997.

[70] T. Sasao, *Switching Theory for Logic Synthesis*, Kluwer Academic Publishers, 1999.

[71] T. Sasao, "A design method of address generators using hash memories," *IWLS*, Vail, Colorado, U.S.A, June 7-9, 2006, pp. 102-109.

[72] T. Sasao, "On the number of variables to represent sparse logic functions," *ICCAD*, San Jose, California, USA, Nov.10-13, 2008, pp. 45-51.

[73] T. Sasao, T. Nakamura, and M. Matsuura, "Representation of incompletely specified index generation functions using minimal number of compound variables," *12th EUROMICRO Conference on Digital System Design, Architectures, Methods and Tools*, (DSD-2009), Aug. 2009, pp. 765-772.

[74] T. Sasao and J. T. Butler, (eds.) *Progress in Applications of Boolean Functions,* Morgan & Claypool Publishers, Jan 2010. pp.1-153.

[75] T. Sasao, *Memory-Based Logic Synthesis*, Springer, 2011.

[76] T. Sasao,"Index generation functions: Recent developments," (invited paper), *ISMVL*, Tuusula, Finland, May 23-25, 2011, pp. 1-9.

[77] T. Sasao, "Linear decomposition of index generation functions," *ASP-DAC*, Jan. 30–Feb. 2, 2012, Sydney, Australia, pp. 781-788.

[78] T. Sasao, "An application of autocorrelation functions to find linear decompositions for incompletely specified index generation functions," *ISMVL*, Toyama, Japan, May 2013.

[79] T. Sasao, "Forty years of logic synthesis: Memoir," *RM*, May 24, 2013, Toyama, Japan.

[80] T. Sasao,"Multiple-valued index generation functions: Reduction of variables by linear transformation," *Journal of Multiple-Valued Logic and Soft Computing*, Vol. 21, No.5-6, pp. 541-559, 2013.

[81] T. Sasao, "Index generation functions: Tutorial," *Journal of Multiple-Valued Logic and Soft Computing*, Vol. 23, No.3-4, pp. 235-263, 2014.

[82] T. Sasao and J. T. Butler, *Applications of Zero-Suppressed Decision Diagrams*, Synthesis Lectures on Digital Circuits and Systems, 2015, 123 pages, Morgan-Claypool.

[83] T. Sasao, "A reduction method for the number of variables to represent index generation functions: s-Min method," *ISMVL*, May 2015, pp. 164-169.

[84] T. Sasao, I. Fumishi, and Y. Iguchi, "A method to minimize variables for incompletely specified index generation functions using a SAT solver," *IWLS*, Mountain View, June 12-13, 2015, pp. 161-167.

[85] T. Sasao, "A linear decomposition of index generation functions: Optimization using autocorrelation functions," *Journal of Multiple-Valued Logic and Soft Computing*, Vol. 28, No. 1, pp. 105-127, 2017.

[86] T. Sasao, "Index generation functions: Minimization methods," *ISMVL*, Novi Sad, Serbia, May 23, 2017, pp. 197-206. (invited).

[87] T. Sasao and M. Maeta, "On affine equivalence of logic functions," *IWLS*, Austin, Texas, June 2017.

[88] T. Sasao, "A logic synthesis for multiple-output linear circuits," *IWLS*, San Francisco, June 23-24, 2018.

[89] T. Sasao,"An improved upper bound on the number of variables to represent index generation functions," *IWLS*, Lausanne, Switzerland, June 21-23, 2019.

[90] T. Sasao, K. Matsuura, and Y. Iguchi, "On irreducible index generation functions," *IWLS*, Lausanne, Switzerland, June 21-23, 2019.

[91] B. Schaeffer,"Product transformation and heuristic EXOR-AND-OR logic synthesis of incompletely specified functions," *IEEE TCAD*, Vol. 36, No. 11, pp. 1831-1841, Nov. 2017.

[92] D. A. Simovici, D. Pletea, and R. Vetro, "Information-theoretical mining of determining sets for partially defined functions," *ISMVL-2010*, May 2010, pp. 294-299.

[93] S. C. Seth and K. L. Kodandapani, "Diagnosis of faults in linear tree networks," *IEEE TC*, Vol. C-26, No. 1, pp. 29-33, Jan. 1977.

[94] N. Song and M. A. Perkowski, "Minimization of exclusive sum of products expressions for multiple-valued input, incompletely specified functions," *IEEE TCAD*, Vol. CAD-15, No. 4, pp. 385-395, April 1996.

[95] S. Stanković, T. Sasao, and C. Moraga, "Spectral transforms decision diagrams," Chapter 3 in [66].

[96] R. S. Stankovic and T. Sasao, "A discussion on the history of research in arithmetic and Reed-Muller expressions," *IEEE TCAD*, Vol. 20, No. 9, Sept. 2001, pp. 1177-1179.

[97] R. S. Stankovic and J. Astola (eds.) E. I. Nechiporuk, "Network synthesis by using linear transformation of variables," in *Reprints from the Early Days of Information Sciences*, Tampere International Center for Signal Processing, Tampere 2007.

[98] R. S. Stankovic, T. Sasao, and J. T. Astola, "Contributions of Yasuo Komamiya to switching theory,"*RM*, May 21, 2015, Waterloo, Ontario, Canada.

[99] R. S. Stankovic, T. Sasao, J. T. Astola, and A. Yamada, "Remarks on the design of first digital computer in Japan - Contributions of Yasuo Komamiya," 17th International Conference on Computer Aided Systems Theory, Las Palmas de Gran Canaria, Spain, 17-22 Feb. 2019.

[100] S. Stergiou and G. Papakonstantinou, "Exact minimization of ESOP expressions with less than eight product terms," *Journal of Circuits, Systems and Computers*, Vol. 13, No. 1, pp. 1-15, 2004.

[101] M. A. Thornton and V. S. S. Nair, "BDD-based spectral approach for Reed-Muller circuit realization," *IEE Proc. Comput. Digt. Tech.*, Vol. 143, No. 2, pp. 145-150, March 1996.

[102] M. A. Thornton, R. Drechsler, and D. M. Miller, *Spectral techniques in VLSI CAD*, Springer Science & Business Media, 2012.

[103] C-C. Tsai and M. Marek-Sadowska, "Generalized Reed-Muller forms as a tool to detect symmetries," *IEEE TC.*, Vol. 45, No. 1, pp. 33-40, Jan. 1996.

[104] D. Varma and E. Trachtenberg, "Design automation tools for efficient implementation of logic functions by decomposition," *IEEE TCAD*, Vol.8, No.8, pp.901-916, 1989.

[105] D. Varma and E. A. Trachtenberg, "Computation of Reed-Muller expansions of incompletely specified Boolean functions from reduced representation," *IEE*, Part E, Vol. 138, No. 2, pp. 85-92, March 1991.

[106] D. Varma and E. A. Trachtenberg, "Efficient spectral techniques for logic synthesis," Chapter 10 in [59], pp. 215-232.

[107] X. Wu, X. Chen, and S. L. Hurst, "Mapping of Reed-Muller coefficients and the minimisation of Exclusive-OR switching functions," *IEE*, Part E, Vol. 129, pp. 15-20, 1982.

[108] I. I. Zhegalkin, "The technique of calculation of statements in symbolic logic," *(in Russian) Mathe. Sbornik*, Vol. 34, pp. 9-28, 1927.