Automation and Remote Control, Vol. 57, No. 1, 1996

AUTOMATA

REALIZATION OF BOOLEAN FUNCTIONS BY ONE LINEAR ARITHMETIC POLYNOMIAL WITH MASKING

V. N. Kondrat'ev and A. A. Shalyto

UDC 519.714

Efficient — in terms of labor consumption and compactness of representation — methods realizing the threshold, threshold-linear, and linear Boolean functions by one linear (noniterated) arithmetic polynomial with masking are proposed. Realization of symmetric Boolean functions is discussed.

1. INTRODUCTION

The use of microprocessors and microcomputers in systems of logic control has enabled one to realize Boolean functions (BF) by nontraditional methods relying on the tremendous possibilities offered by these facilities. Realization of BFs by arithmetic polynomials (AP) [1–14] is among these methods.

Since, in the general case, the AP for a BF of n variables is nonlinear (iterated) and has 2^n terms, its computational complexity grows dramatically with n. Realization of BFs by linear (noniterated) APs (LAP) is especially important, because the computation of LAPs reduces to summing the coefficients of the variables that are equal to unity. Yet, the matters involved in realization of BFs by LAPs have not yet been adequately explored.

Artukhov, Kondrat'ev, and Shalyto [7] determined the linearity condition under which a BF of n variables is realizable by one irredundant LAP of these variables. We recall that a LAP is referred to as irredundant if the number of bits in the binary representation of the maximal value as computed from the polynomial equals the number of realizable BFs, or redundant if this number exceeds the number of realizable functions. As shown below, the irredundant LAP realizes only BFs that depend essentially on one variable at most.

The class of LAP-realizable functions can be expanded by redundantization, that is, by realization of other functions besides the given one [1]. Here, the LAP computes from an input pattern the decimal value of the cortege of functions whose binary representation has in the corresponding digit the value which the given function has on this pattern.

If the least-significant digit is assumed to be first, then the operation of selecting the ℓth digit in the m-bit binary representation of the decimal number will be called masking and denoted by r_m^ℓ, and the transformation of a decimal number into binary form will be denoted by "bin."

A LAP whose decimal value is masked in order to select a bit in its binary representation will be referred to as a masked LAP (MLAP). Efremov, Kuz'min, and Stepanov proposed [4] a method for MLAP construction, which is very difficult because of the need to transform involved logic formulas and solve complex logic equations by the use of integer programming based on the Balas algorithm.

Malyugin [3] described a much simpler method for realization of arbitrary BFs defined in the normal disjunctive form by superposing two MLAPs of which the first MLAP calculates all conjunctions and the second one establishes the disjunction resulting from substituting a new letter for each conjunction in the BF. Thus, we can state with reasonable confidence that no constructive method exists for introducing additional functions into a cortege with the given BF with the aim of realizing them by a single MLAP.

The present publication proposes efficient — in terms of labor consumption and compactness of representation — methods for realizing threshold, threshold-linear, and linear BFs by one MLAP. This is of both practical (if the MLAP is more compact than other forms of function representation) and theoretical importance, because

Research Institute of the Research and Production Association "AVRORA," St. Petersburg. Translated from Avtomatika i Telemekhanika, No. 1, pp. 158–170, January, 1996. Original article submitted November 15, 1994.

0005-1179/96/5701-0127$15.00

the proposed methods are meant for determining the interrelations between such important mathematical objects as Boolean functions (formulas) and linear arithmetic polynomials. The theorems of [4, 6] associated with the BF classes were also used in developing these methods.

2. REALIZATION OF BOOLEAN FUNCTIONS BY AN IRREDUNDANT LINEAR ARITHMETIC POLYNOMIAL

THEOREM 1. *The linear arithmetic polynomial can realize noniteratively only a Boolean function which is dependent essentially on one variable at most* — 0, 1, $!x$, x, *where* ! *stands for "inversion."*

Proof. Each value of the BF must be either 0 or 1, and the LAP coefficients can assume values -1, 0, 1, except for $a0 = 0, 1$. If $a0 = 0$ ($a0 = 1$), then at most one of the remaining coefficients can be 1 (-1), whereas the remaining coefficients must be 0. Otherwise, the polynomial will be neither 0 nor 1 on at least one input pattern, which proves the theorem.

As follows from the theorem, a single BF, which is essentially dependent on two or more variables, is not realizable by the noniterated LAP; therefore, we will consider realization of such functions by the MLAP.

3. REALIZATION OF THE THRESHOLD FUNCTIONS

This section considers two methods for realization of the threshold functions (TF). The first method constructs a MLAP of n variables using the threshold representation of the function of n variables, and the second one uses the MLAP of the threshold function of the $(n-1)$st variable.

(a) *First method.* A Boolean function is called a threshold function if

$$f = \begin{cases} 1 \quad \text{for} \quad \sum_{i=1}^{n} wi * xi \geqslant T, \\ 0 \quad \text{for} \quad \sum_{i=1}^{n} wi * xi < T, \end{cases} \tag{1}$$

where wi is the weight of variable xi, T is the threshold of a function, and $*$ stands for "multiplication."

Thus, we can assert that the threshold function f of n variables has the LAP

$$P = -T + w1 * x1 + w2 * x2 + \ldots + wn * xn$$

and the conditional operator

$$\text{if } (P \geqslant 0) \text{ then } f = 1 \text{ else } f = 0.$$

In this section, our task is to find for f another LAP

$$P = w0 + w1 * x1 + w2 * x2 + \ldots + wn * xn,$$

such that

$$f = r_m^m(\text{bin } P).$$

Therefore, the problem reduces, in a sense, to checking whether it is possible to replace the conditional operator by masking, which is simpler for software realization.

For a function f of n variables, which belongs to the classes of nonthreshold functions considered in this work, a LAP is sought

$$P = a0 + a1 * x1 + a2 * x2 + \ldots + an * xn$$

such that

$$f = r_m^t(\text{bin } P).$$

Returning now to the threshold functions, we note that if the threshold is 2^α ($\alpha = 0, 1, 2, \ldots$), it follows that 1 in at least one bit of the binary representation of the values $\sum_{i=1}^{n} wi * xi$ beginning from the bit $\alpha + 1$ (for bit enumeration beginning from 1) testifies that the computed value has reached or exceeded T.

If $T \neq 2^\alpha$, then by increasing the threshold to the nearest higher degree of two the problem can be reduced to the above one. To this end, we add $2^{]\log_2 T[} - T$ to both sides of (1):

$$2^{]\log_2 T[} - T + \sum_{i=1}^{n} wi * xi \geqslant 2^{]\log_2 T[}. \tag{2}$$

Here, the threshold will be $T_1 = 2^\alpha = 2^{]\log_2 T[}$. The left-hand side of (2) is a LAP of the form

$$P = 2^{]\log_2 T[} - T + \sum_{i=1}^{n} wi * xi. \tag{3}$$

It follows that an arbitrary TF is realizable as

$$f = r_\beta^\beta(\text{bin } P) \,|\, r_\beta^{\beta-1}(\text{bin } P) \,|\, \ldots \,|\, r_\beta^{\alpha+1}(\text{bin } P), \tag{4}$$

where β is the number of bits in the binary representation of the maximal value of P and | denotes "disjunction."

Example 1. It is required to realize the TF $f = x1 \,|\, x2 \,|\, x3$.

Here, (1) takes the form $x1 + x2 + x3 \geqslant 2^0$. Since here the maximal value of P is equal to three, it follows that $\beta = 2$. Since $\alpha = 0$, it follows from (4) that

$$f = r_2^2\big(\text{bin}\,(x1 + x2 + x3)\big) \,|\, r_2^1\big(\text{bin}\,(x1 + x2 + x3)\big).$$

We consider the case of $\beta = \alpha + 1$ — the threshold is exceeded only in one bit of the binary representation of P. To this end, β must not exceed the number of bits in the binary representation of the threshold T_1. This condition is representable as

$$P_{\max} < 2 * T. \tag{5}$$

where $P_{\max}$ is the maximal value of the polynomial P.

We note that if the condition (5) for TF is not satisfied, then the value of T_1 must be doubled by adding T_1 to both sides of (2). If again (5) is not satisfied, this procedure is iterated until this condition is satisfied. Therefore, (5) can be satisfied for any TF, because the right-hand side of (5) grows with double speed upon increasing both sides of (2).

Thus, we have proved that an arbitrary TF is computable as

$$f = r_\beta^\beta(\text{bin } P). \tag{6}$$

The software realization of this relationship in a high-level language such as C is extremely simple: compute the value of the polynomial P on the given input pattern, shift the result by $\beta - 1$ bits to the right, and read the final result which equals the TF on this pattern.

To apply this approach to the given TF, it is first necessary to find its threshold realization (TR) (1).

The procedure for determination of the weights of variables and the TF threshold is very laborious [15]. Therefore, we consider three simpler approaches to establishing the threshold realization. The first approach constructs the TR directly from its description. The second approach is based on the catalogs of optimal TRs (for example, [16]) for arbitrary TFs of six variables at most. The third one, which is described in the Appendix, is applicable to the noniterated threshold formulas (NTF), which are of both practical and theoretical significance [17].

Example 2. It is required to realize the TF "two or more of three" described by $f = (x1 \,|\, x2)\&x3 \,|\, x1\&x2$.

It follows from the verbal description of the function that $x1 + x2 + x3 \geqslant 2$. Since here $T = 2^\alpha$ and (5) is satisfied, it follows that

$$f = r_2^2(\text{bin}\,(x1 + x2 + x3)).$$

We illustrate the obtained relationship by Table 1.

It follows from Table 1 that by means of redundancy (realization of the polynomials of two functions instead of one) and masking one can construct a MLAP

$$f = x1 * x2 + x1 * x3 + x2 * x3 - 2 * x1 * x2 * x3,$$

instead of an irredundant (in terms of united functions) but more complicated nonlinear AP (NAP) [7].

TABLE 1

$x1$	$x2$	$x3$	$x1+x2+x3$	$\text{bin}\,(x1+x2+x3)$	
				r_2^2	r_2^1
0	0	0	0	0	0
0	0	1	1	0	1
0	1	0	1	0	1
0	1	1	2	1	0
1	0	0	1	0	1
1	0	1	2	1	0
1	1	0	2	1	0
1	1	1	3	1	1

Thus, the fact that a noniterated AP with masking corresponds to an iterated BF underlies its efficient software realization. For a noniterated Boolean function, its direct software realization is always more efficient than that of a MLAP-based realization. Yet, for systems of threshold (even noniterated) functions, MLAPs can enable a memory-minimal software realization [13].

The determined MLAP is realized in C by the following program:

```
mlap (int x1, int x2, int x3)
    { int f;
      f = (x1 + x2 + x3) >> 1; },
```

where $\gg L$ stands for "L-bit right shift."

Importantly, for the chosen types of variables, this fourteen-byte realization of the function "two or more of three" is the simplest one, compared with more than sixty other variants of its software realization known to the present authors. For example, the direct realization of

```
for (int x1, int x2, int x3)
   { int f;
     f = (x1 | x2)&x3 | x1&x2; }
```

requires 20 bytes of memory space, and the TR

```
prg (int x1, int x2, int x3)
   { int f = 0;
     if ((x1 + x2 + x3) >= 2)  f = 1; }
```

requires even 24 bytes.

Example 3. It is required to realize the TF "seven or more of ten."

It follows from the description of the function that $\sum_{i=1}^{10} xi \geqslant 7$. If we increase the function threshold to 2^a, then $1+\sum_{i=1}^{10} xi \geqslant 8$. Since (5) is satisfied, it follows that

$$f = r_4^4\left(\text{bin}\left(1+\sum_{i=1}^{10} xi\right)\right).$$

This example illustrates the high efficiency of MLAPs, because here the truth table, Boolean function, and NAP are very inconvenient and require much memory.

Example 4. It is required to realize the TF $f = (x1 \mid x2 \mid x3)\&x4 \mid x1\&x2\&x3$.

It follows from [15] that here $x1+x2+x3+2*x4 \geqslant 3$. Using the fact that $T \neq 2^a$, we increase the threshold of $1 + x1 + x2 + x3 + 2 * x4 \geqslant 4$. Since (2) and (5) are satisfied, it follows that

$$f = r_3^3\big(\text{bin}\,(1 + x1 + x2 + x3 + 2 * x4)\big).$$

Example 5. It is required to realize the NTF $f = x1\&x2$.

Using the method described in the Appendix, we get $x1 + x2 \geqslant 2$. Since here $T = 2^{\alpha}$ and (5) is satisfied, it follows that

$$f = r_2^2(\mathrm{bin}\,(x1 + x2)).$$

Example 6. It is required to realize the NTF $f = x1 \mid x2$.

Using the method described in the Appendix, we get $x1 + x2 \geqslant 1$. Since $T = 2^{\alpha}$, but (5) is not satisfied, we increase the function threshold — $1 + x1 + x2 \geqslant 2$. Since now both conditions are satisfied, it follows that

$$f = r_2^2(\mathrm{bin}\,(1 + x1 + x2)).$$

Example 7. It is required to realize the NTF $f = x1\&x2\&x3$.

Using the method described in the Appendix, we get $x1 + x2 + x3 \geqslant 3$. Since $T \neq 2^{\alpha}$, we increase the function threshold up to the nearest degree of two — $1 + x1 + x2 + x3 \geqslant 4$. In view of the fact that here (5) is satisfied, it follows that

$$f = r_3^3(\mathrm{bin}\,(1 + x1 + x2 + x3)).$$

Example 8. It is required to realize the NTF $f = x1\&x2 \mid x3$.

Using the method described in the Appendix, we get $x1 + x2 + 2 * x3 \geqslant 2$. Here, $T = 2^{\alpha}$, but since (5) is not satisfied ($P_{\max} = 2 * T$), we increase the threshold up to the next degree of two — $2 + x1 + x2 + 2 * x3 \geqslant 4$. Since now (5) is satisfied, it follows that

$$f = r_3^3(\mathrm{bin}\,(2 + x1 + x2 + 2 * x3)).$$

(b) *Second method.* The method is based on the theorem presented in [6].

THEOREM 2. *If the cortege $f_N \# f_{N-1} \#, \ldots, \#, f_1$ of N Boolean functions of n variables is realized by the linear arithmetic polynomial P_N, it follows that the cortege of $(N + 1)$ function*

$$(f_N \& x_{n+1}) \# (f_N \oplus x_{n+1}) \# f_{N-1} \# \ldots \# f_1 \tag{7}$$

is realized by LAP of the form $P_{N+1} = P_N + 2^{N-1} * x_{n+1}$, and a cortege of the same number of functions

$$(f_N \mid x_{n+1}) \# (f_N \oplus x_{n+1} \oplus 1) \# f_{N-1} \# \ldots \# f_1 \tag{8}$$

is realized by a LAP of the form $P_{N+1} = P_N + 2^{N-1} * x_{n+1} + 2^{N-1}$, where $\#$ is the operation of linking into a cortege and $\oplus$ is "modulo 2 sum."

By virtue of the fact that the LAP realizes only function corteges where the extreme left function f_N is a TF [13] and that $f_N \& x_{n+1}$ and $f_N \mid x_{n+1}$ are also threshold functions if f_N is a threshold function of n variables [15], Theorem 2 defines:

- a recursive method of constructing a MLAP for a NTF and
- a simple method of realizing a TF of the form $f_N \& x_{n+1}$ and $f_N \mid x_{n+1}$ for the case where the MLAP for f_N has been established using, for example, the above approaches.

Example 9. It is required to realize the BF $f = x1\&x2\&x3$.

It follows from Example 5 that $N = 2$ for $f1 = x1\&x2$ $n = 2$, and, therefore, we have from (7) that

$$f = r_3^3(\mathrm{bin}\,(x1 + x2 + 2 * x3)).$$

It follows from the above relationship and Example 7 that the representation of BFs by MLAPs is not unique — in contrast to their representation by APs [1, 7].

Example 10. It is required to realize the BF $f = (x1 \mid x2)\&x3 \mid x1\&x2 \mid x4$.

It follows from Example 2 that $N = 2$ for $f1 = (x1 \mid x2)\&x3 \mid x1\&x2$ $n = 2$, and, therefore, we get from (8) that

$$f = r_3^3(\mathrm{bin}\,(2 + x1 + x2 + x3 + 2 * x4)).$$

The above methods are applicable to the TFs representable by inversionless formulas — the positive monotone TFs.

If the realized formula involves variables suchas $!xi$, then a MLAP is constructed for a positive monotone TF of the same type and then $1 - xi$ is substituted into it for the variable $!xi$.

Example 11. It is required to realize the TF $f = x1 \,|\, !x2$.

It follows from the aforesaid and Example 6 that

$$f = r_2^2(\text{bin}\,(2 + x1 - x2)).$$

Example 12. It is required to realize the TF $f = (x1 \,|\, x2)\&x3 \,|\, x1\&x2 \,|\, !x4$.

It follows from the aforesaid and Example 10 that

$$f = r_3^3(\text{bin}\,(4 + x1 + x2 + x3 - 2 * x4)).$$

4. REALIZATION OF THE LINEAR FUNCTIONS

By a linear BF is meant a BF representable by

$$f = c0 \oplus c1\&x1 \oplus c2\&x2 \oplus \ldots \oplus cn\&xn,$$

where $ci = 0, 1$.

The linear function is realized by a MLAP of the form

$$f = r_{1+[\log_2 n]}^1\left(\text{bin}\,(c0 * (1 - xn) + \sum_{k=1}^{n-c0} ck * xk)\right).$$

Therefore,

$$f1 = x1 \oplus x2 = r_2^1(\text{bin}\,(x1 + x2)),$$
$$f2 = x1 \oplus x2 \oplus x3 = r_2^1(\text{bin}\,(x1 + x2 + x3)).$$

The second of the above functions is realized in C by the following program:

```
mod (int x1, int x2, int x3)
{ int f;
  f = (x1 + x2 + x3) & 1; }.
```

The above functions are realized by the same MLAPs as the functions $f1 = x1\&x2$ and $f2 = (x1 \,|\, x2)\&x3 \,|\, x1\&x2$, respectively, as they must be for the binary one-bit half-adder and adder.

5. REALIZATION OF THE THRESHOLD-LINEAR FUNCTIONS

By a threshold-linear function (TLF) is meant a Boolean function representable as

$$f = f1(x1, \ldots, xN) \oplus \widetilde{xi} \oplus \ldots \oplus \widetilde{xj},$$

where $f1$ is the threshold function, i and j are integers, and $\widetilde{xi} = \{!xi, xi\}$.

Let us consider the methods for constructing the MLAPs for TLFs, provided that the realized function is defined in threshold-linear form.

Theorem 2 allows one to realize the TLFs for which $i, j > n$ and the MLAP for $f1$ is determined using the methods of Sec. 3.

Example 13. It is required to realize the TLF $f = x1\&x2\&x3 \oplus !x4$.

It follows from Example 7 that $f1 = x1\&x2\&x3 = r_3^3(\text{bin}\,(1 + x1 + x2 + x3))$. Therefore, we determine from (8) that $f = x1\&x2\&x3 \oplus !x4 = x1\&x2\&x3 \oplus x4 + 1 = r_4^3(\text{bin}\,(5 + x1 + x2 + x3 + 4 * x4))$.

We realize the given function in a different way. It follows from (7) and Sec. 3 that $f = x1\&x2\&x3 \oplus !x4 = r_4^3(\text{bin}\,(5 + x1 + x2 + x3 - 4 * x4))$.

This relationship is realized in C by the following program:

```
plf ( int x1, int x2, int x3, int x4 )
{ int f;
  f = ((5 + x1 + x2 + x3 - 4 * x4) >> 2) & 1;}.
```

The disadvantage of Theorem 2 — inapplicability to the case $i, j \leqslant n$ — can be eliminated using the following theorem [4].

THEOREM 3. *If* $f1 = r_m^k(\mathrm{bin}\, P_N)$, *then* $f = f1 \oplus xi$ *is realized in the* k*th digit of the polynomial* $P_N + 2^{k-1} * xi$, *where* $i = \{1, 2, \ldots, n+1\}$.

Example 14. It is required to realize the TLF $f = x1 \oplus x2\&(!x3 \,|\, x4)$.
It follows from Example 11 that $f1 = !x3 \,|\, x4 = r_2^2(\mathrm{bin}\,(2 - x3 + x4))$.
From (7) we have that $f2 = x2\&(!x3 \,|\, x4) = r_3^3(\mathrm{bin}\,(2 + 2 * x2 - x3 + x4))$. By Theorem 3, $f = r_4^3(\mathrm{bin}\,(2 + 4 * x1 + 2 * x2 - x3 + x4))$. We note that in [1] this formula is realized by a substantially more complex NAP:

$$f = 1 - x1 - x2 - x3 + 2 * x1 * x2 + 2 * x1 * x3 + 2 * x2 * x3 + x3 * x4$$
$$-4 * x1 * x2 * x3 - 2 * x1 * x3 * x4 - 2 * x2 * x3 * x4 + 4 * x1 * x2 * x3 * x4.$$

Example 15. It is required to realize the TLF $f = (x1 \,|\, x2 \,|\, x3) \oplus x3$. ·
By virtue of $f1 = x1 \,|\, x2 \,|\, x3 = r_3^3(\mathrm{bin}\,(3 + x1 + x2 + x3))$, it follows from Theorem 3 that

$$f = r_4^3(\mathrm{bin}\,(3 + x1 + x2 + 5 * x3)).$$

On the other hand, since $f = (x1 \,|\, x2 \,|\, x3) \oplus x3 = (x1 \,|\, x2)\&x3$ is a NTF, it follows from the results of Sec. 3 that

$$f = r_3^3(\mathrm{bin}\,(3 + x1 + x2 - 2 * x3)).$$

Example 16. It is required to realize the TLF $f = x1\&x2 \oplus x3 \oplus x4 \oplus x5$.
Iterative application of Theorem 3 provides $f1 = x1\&x2 = r_2^2(\mathrm{bin}\,(x1 + x2))$, $f2 = x1\&x2 \oplus x3 = r_3^2(\mathrm{bin}\,(x1 + x2 + 2 * x3))$, and $f3 = x1\&x2 \oplus x3 \oplus x4 = r_3^2(\mathrm{bin}\,(x1 + x2 + 2 * x3 + 2 * x4))$. Thus,

$$f = r_4^2(\mathrm{bin}\,(x1 + x2 + 2 * x3 + 2 * x4 + 2 * x5)).$$

Interestingly, in contrast to the above examples, the realized function has "shifted" here to the middle of the binary representation.

Now we consider the case where a TLF is defined in a form different from the threshold-linear one. It is suggested to seek such a form using the relationships [7]

$$f = (!xi\&f(0) \,|\, xi\&!f(1)) \oplus xi; \tag{9}$$
$$f = (!xi\&!f(0) \,|\, xi\&f(1)) \oplus !xi. \tag{10}$$

Example 17. It is required to realize the TLF $f = !x1\&x2 \,|\, x1\&!x2\&x3$.
We get from (9) that $f = f1 \oplus x1 = (x1\&!x3 \,|\, x2) \oplus x1$. For the NTF, $f1 = r_3^3(\mathrm{bin}\,(3 + x1 + 2 * x2 - x3))$. Therefore, we get from Theorem 3 that

$$f = r_4^3(\mathrm{bin}\,(3 + x1 + 2 * x2 - x3)).$$

This function was realized by Efremov, Kuz'min, and Stepanov [4] using the Balas algorithm and a more complex polynomial:

$$f = r_6^4(\mathrm{bin}\,(5 + 12 * x1 + 6 * x2 + 14 * x3)).$$

Example 18. It is required to realize the TLF, $f = (!x1 \,|\, !x2 \,|\, !x3)\&(x1 \,|\, x2 \,|\, x3))$.
We get from (9) that $f = f1 \oplus x1 = (!x1\&(!x2 \,|\, x3) \,|\, x2\&x3) \oplus x1$. For the TLF, $f1 = r_2^2(\mathrm{bin}\,(1 - x1 + x2 + x3))$. Therefore, we get from Theorem 3 that

$$f = r_3^2(\mathrm{bin}\,(1 + x1 + x2 + x3)).$$

Example 19. It is required to realize the TLF $f = x1\&!x2\&!x3\&x4 \,|\, !x1\&x2\&x3\&x4 \,|\, !x1\&!x2\&!x4 \,|\, x1\&x3\&!x4 \,|\, x2\&!x3\&!x4$.

It follows from (9) for $i = 4$ that $f = f1 \oplus x4 = (!x1\&!x2 \,|\, x1\&x3 \,|\, x2\&!x3) \oplus x4$. By applying (10) to $f1$, we get $f = f2 \oplus !x1 \oplus x4 = ((x1 \,|\, x2)\&x3 \,|\, x1\&x2) \oplus !x1 \oplus x4$.

It follows from Example 2 that $f2 = r_2^2(\mathrm{bin}\,(x1 + x2 + x3))$. Therefore, by applying Theorem 3 we get that $f1 = r_3^2(\mathrm{bin}\,(2 - x1 + x2 + x3))$. By using Theorem 3 again we get

$$f = r_3^2\big(\mathrm{bin}\,(2 - x1 + x2 + x3 + 2 * x4)\big).$$

In contrast to other forms of representing this function, the obtained relationship is noniterated, thus enabling a simpler software realization.

The form and complexity of the MLAP found from the TLF depend also on the particular relationship used and on the variable used as xi.

Example 20. It is required to realize the TLF $f = !x1\&!x2\&x3 \,|\, x1\&x2\&!x3$.

Assuming in (9) that $xi = x1$, we get $f = f1 \oplus x1 = (!x1\&(!x2\&x3) \,|\, x1\&!(x2\&!x3)) \oplus x1 = ((x1 \,|\, x3)\&!x2 \,|\, x1\&x3) \oplus x1$.

Here, $f1 = r_2^2(\mathrm{bin}\,(1 + x1 - x2 + x3))$, because $f2 = (x1 \,|\, x3)\&x2 \,|\, x1\&x3 = r_2^2(\mathrm{bin}\,(x1 + x2 + x3))$. By Theorem 3, we have

$$f = r_3^2\big(\mathrm{bin}\,(1 + 3 * x1 - x2 + x3)\big).$$

Using the variables $!x1$, $x2$, $!x2$, $x3$, and $!x3$ as xi, we get the following relationships: $f = r_3^2(\mathrm{bin}\,(4 - 3 * x1 + x2 - x3)) = r_3^2(\mathrm{bin}\,(1 - x1 + 3 * x2 + x3)) = r_3^2(\mathrm{bin}\,(4 + x1 - 3 * x2 - x3)) = r_3^2(\mathrm{bin}\,(x1 + x2 + 3 * x3)) = r_3^2(\mathrm{bin}\,(5 - x1 - x2 - 3 * x3))$.

The simplest realization, thus, is obtained with $xi = x3$. The questions of membership of the function in the TLF class and of the choice of the variable xi in (9) and (10) still remain open.

6. REALIZATION OF SYMMETRIC FUNCTIONS

We consider the class of LAPs of the form $P = \sum_{i=1}^{n} xi$ describing combinatorial counters. Their design requires the determination of a system of $1 + [\log_2 n]$ symmetric Boolean functions that are realized as follows.

We write an $(n+1) * 1$ column matrix whose elements assume values from $0, 1, \ldots, n$. It is transformed into an $(n+1) * (1 + [\log_2 n])$ binary matrix whose ith $(i = 0, 1, \ldots, n)$ row is the binary representation of the ith element of the column matrix. Here, the jth column of the binary matrix is the characteristic number of the jth function, and 1 in its ith bit indicates that this function has a working number.

Example 21. It is required to determine the symmetric functions (SF) realized by the LAP $P = x1 + x2 + x3 + x4$.

By executing the aforementioned constructions and transformations, we get

$$P = \left|\begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array}\right| = \left|\begin{array}{ccc} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{array}\right| = \left|\; S_4^4 \,\#\, S_{2,3}^4 \,\#\, S_{1,3}^4 \;\right|.$$

Let us consider a wider class of LAPs:

$$P1 = c1 + c2 * \sum_{i=1}^{n} xi, \tag{11}$$

where $c1$, $c2$ are positive integers. This class of LAPs realizes SFs as above.

Example 22. It is required to determine the SFs realizable by the LAP $P1 = 3 + 5 * (x1 + x2 + x3 + x4)$.

The required functions are determined as follows:

$$P1 = 3 + 5 * \left|\begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array}\right| = \left|\begin{array}{c} \mathbf{3} \\ \mathbf{8} \\ \mathbf{13} \\ \mathbf{18} \\ \mathbf{23} \end{array}\right| = \left|\begin{array}{ccccc} 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \end{array}\right| = \left|\; S_{3,4}^4 \,\#\, S_{1,2}^4 \,\#\, S_{2,4}^4 \,\#\, S_{0,3,4}^4 \,\#\, S_{0,2,4}^4 \;\right|.$$

It follows from above, for instance, that

$$f = S^4_{2,4} = r^3_5\Big(\mathrm{bin}\big(3 + 5 * (x1 + x2 + x3 + x4)\big)\Big).$$

We prove that this realization is more efficient than the direct construction of the AP for a SF.

An arbitrary SF of n variables was shown [12] to be realizable by an AP with at most $n + 1$ coefficients ai. For $n = 4$, the following relationship holds:

$$f = a4 * \psi^4_4 + a3 * \psi^3_4 + a2 * \psi^2_4 + a1 * \psi^1_4 + a0,$$

where

$$\psi^4_4 = x1 * x2 * x3 * x4;$$
$$\psi^3_4 = x1 * x2 * x3 + x1 * x2 * x4 + x1 * x3 * x4 + x2 * x3 * x4;$$
$$\psi^2_4 = x1 * x2 + x1 * x3 + x1 * x4 + x2 * x3 + x2 * x4 + x3 * x4;$$
$$\psi^1_4 = x1 + x2 + x3 + x4.$$

For the function $S^4_{2,4}$, it was proved in [12] that $(a4, a3, a2, a1, a0) = (7, -3, 1, 0, 0)$.

It follows from the aforesaid that (11) can be regarded as an efficient SF generator. Studies have shown that all SFs of two variables are generated by five LAPs for $c1 = 0, \ldots, 4$, $c2 = 1$, and all SFs of three variables are generated by eleven LAPs, where $c1 = 0, \ldots, 5$, $c2 = 1$; $c1 = 7$, $c2 = 1$; $c1 = 0$, $c2 = 3$; $c1 = 2$, $c2 = 3$; $c1 = 4$, $c2 = 3$; $c1 = 5$, $c2 = 5$.

This method is based on enumerative search and does not ensure realization of an arbitrary SF of n variables. Constructive methods for realization of threshold, threshold-linear, and linear SFs were considered above. For example, using the method of Sec. 5 we can show that

$$S^3_{0,2,3} = (x1\&x3 \,|\, !x2) \oplus x1 \oplus x3 = r^3_4\big(\mathrm{bin}\,(4 + 5 * x1 - 2 * x2 + 5 * x3)\big);$$
$$S^3_1 = (!x1 \,|\, !x3)\&x2 \oplus x1 \oplus x3 = r^3_4\big(\mathrm{bin}\,(3 + 3 * x1 + 2 * x2 + 3 * x3)\big).$$

The question of membership of an arbitrary SF of n variables in one of the classes listed above remains open.

APPENDIX

CONSTRUCTION OF THE THRESHOLD REALIZATION FOR A POSITIVE MONOTONE NONITERATED THRESHOLD FORMULA

1. In the basis $\{\&, |\}$ of h characters, a tree-like circuit of two-input ANDs and ORs of maximal depth $h - 1$ is constructed. If this circuit is "linear," that is, if all its elements are connected serially, then the given formula is a positive monotone noniterated threshold formula (PMNTF).

2. A linear binary graph (LBG) [17] with maximal number of paths from the input to the unity and zero outputs is constructed for the PMNTF [18].

3. By the modified [19] Akers method [20], we count the number of paths in the LBG, the zero and unity marks of the conditional vertices (CV) being eliminated and new marking being performed:

• the beginning of the LBG is marked 1, and it is assumed that the arcs going out of each CV are marked as its input, and

• in each point of junction of arcs, the "Kirchhoff law" is obeyed, that is, the mark of the outgoing arc equals the sum of marks of the incoming arcs.

4. The input mark of the ith CV equals the weight of the variable xi, that of the "zero" operator vertex being equal to the function threshold.

Example A1. Construct a TR for $f = (x1\&x2 \,|\, x3)\&x4$.

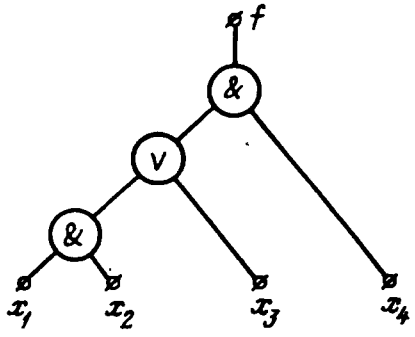

Fig. 1

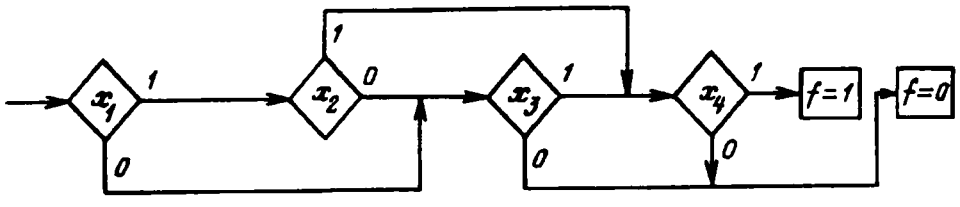

Fig. 2

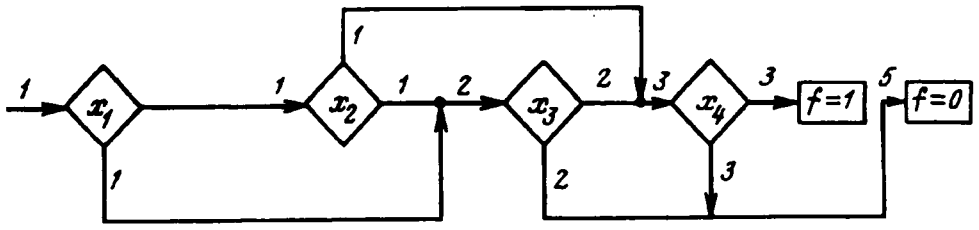

Fig. 3

This formula is a PMNTF, because it is noniterated, involves no inversions, and is realized by a "linear' circuit (Fig. 1). Figure 2 depicts a LBG with the maximal number of paths. It follows from the LBG marked witl the aim of counting the number of paths (Fig. 3) that $w1 = 1$, $w2 = 1$, $w3 = 2$, $w4 = 3$, $T = 5$. Therefore $x1 + x2 + 2 * x3 + 3 * x4 \geqslant 5$, and it follows from Sec. 3 that $f = r_3^3(\mathrm{bin}\,(3 + x1 + x2 + 2 * x3 + 3 * x4))$.

REFERENCES

1. V. D. Malyugin, "Representation of Boolean functions by arithmetic polynomials," *Avtomat. Telemekh.* No. 4, 84–93 (1982).
2. V. D. Malyugin, "On polynomial realization of a cortege of Boolean functions," *Dokl. Akad. Nauk SSSR* 265, No. 6, 1338–1341 (1982).
3. V. D. Malyugin, "Realization of corteges of Boolean functions by linear arithmetic polynomials," *Avtomat Telemekh.*, No. 2, 114–122 (1984).
4. V. D. Efremov, A. A. Kuz'min, and V. A. Stepanov, "Computing logical functions with the aid of the Rademacher transform," *Avtomat. Telemekh.*, No. 2, 105–113 (1984).
5. V. D. Malyugin, G. A. Kukharev, and V. P. Shmerko, *Transformation of Polynomial Forms of Boolean Functions*, Preprint, Institute of Control Sciences, Moscow (1986).

6. V. Malyugin, "Realization of logical function sets by arithmetic polynomials," *Comput. Artificial Intelligence*, No. 6, 541–552 (1987).
7. V. L. Artukhov, V. N. Kondrat'ev, and A. A. Shalyto, "Generating Boolean functions via arithmetic polynomials," *Avtomat. Telemekh.*, No. 4, 138–147 (1988).
8. V. A. Osipov, A. A. Shalyto, and V. N. Kondrat'ev, *Software Realization of Logic Control Algorithms in Ship Systems* [in Russian], Institute of Advanced Studies for Managers and Specialists of the Ship Building Industry, Leningrad (1988).
9. A. A. Shalyto, *Microprocessor-Based Realization of Algorithms of Logic Control Systems* [in Russian], Institute of Advanced Studies for Managers and Specialists of the Ship Building Industry, Leningrad (1988).
10. V. P. Shmerko, "Synthesis of arithmetic forms of Boolean functions using the Fourier transform," *Avtomat. Telemekh.*, No. 5, 134–142 (1989).
11. V. L. Artyukhov, V. N. Kondrat'ev, and A. A. Shalyto, "Using linear arithmetic polynomials for realization of logic control systems," *Proc. IX All-Union Control Conf., Abstracts*, Akad. Nauk SSSR (1989), pp. 495–496.
12. V. L. Artyukhov, N. S. Belyaev, and A. A. Shalyto, *Methods of Software Realization of Symmetric Boolean Functions* [in Russian], A. N. Krylov VNTO, Issue 19, Sudostroenie, Leningrad (1989), pp. 37–51.
13. A. A. Shalyto and V. N. Kondrat'ev, *Methods for Software Realization of Logic Control Algorithms for Ship Microprocessor Systems* [in Russian], Institute of Advanced Studies for Managers and Specialists of the Ship Building Industry, Leningrad (1990).
14. V. N. Kondrat'ev and A. A. Shalyto, "Realization of systems of Boolean functions by linear arithmetic polynomials," *Avtomat. Telemekh.*, No. 3, 135–151 (1993).
15. E. A. Butakov, *Methods of Synthesizing Relay Devices of Threshold Components* [in Russian], Energiya, Moscow (1970).
16. S. Muroga, I. Toda, and M. Kondo, "Majority decision function of up to six variables," *Math. Comput.*, **16**, No. 80, 132–150 (Oct.,1962).
17. V. L. Artyukhov, L. Ya. Rozenblyum, and A. A. Shalyto, "Logic possibilities of some types of cascaded structures," in: *Communication Networks and Discrete Control Systems.* [in Russian], Nauka, Moscow (1976), pp. 138–144.
18. B. P. Kuznetsov and A. A. Shalyto, "System of transformations of certain representations of Boolean functions," *Avtomat. Telemekh.*, No. 11, 120–127 (1985).
19. V. L. Artyukhov, B. P. Kuznetzov, and A. A. Shalyto, *Adjustable Logic Devices for Ship Control Systems* [in Russian], Institute of Advanced Studies for Managers and Specialists of the Ship Building Industry, Leningrad (1986).
20. S. B. Akers, "Binary decision diagrams," *IEEE Trans. Comput.*, No. 6, 509–516 (1978).