Improvements in multioutput threshold-logic gates

J. M. Eves* and S. L. Hurst†

Previous developments by Winn (1975) on the design of single-output threshold-logic gates using Boolean product-of-sum expansions have been extended to the consideration of multioutput threshold-logic gates, by searching for optimum factorising of Winn functions to provide common terms for dissimilar threshold outputs. It has been found that such searches do not yield good results, but the combination of Winn plus parity functions is more viable. A need for more powerful digital synthesis techniques to handle multioutput design, with the ability to choose the number of logic levels per realisation, is found to be desirable for this and all other multioutput design problems.

(Received November 1976)

List of symbols used

x_i, $i = 1, 2, \ldots, n$	independent binary gate inputs each of which may take the value 0 or 1
$f(x)$	function of the binary inputs $x_1, \ldots, x_n$
a_i, $i = 1, 2, \ldots, n$	independent real number coefficients or 'weights' associated with inputs $x_1, \ldots, x_n$, respectively
t	real number gate output threshold value

To differentiate between Boolean equations and threshold expressions, the following notations are employed: Boolean equations employ [ ] for outer brackets, and ( ) and { } for any necessary internal brackets. Within these defining brackets + and . take the normal Boolean meaning of OR and AND, respectively. The latter is omitted where no ambiguity results.

Threshold expressions employ ⟨ ⟩ for outer brackets. Within such brackets normal arithmetic rules of addition and multiplication hold, with the binary input signals x_i taking the numerical values of 0 or 1.

1. Introduction

Threshold-logic gates represent a particular class of digital logic gates with considerable theoretical attractions, but with limited practical exploitation to date except for the specific case of Majority gates (Halligan, 1974). Nevertheless the continuing study of threshold-logic functions and corresponding realising gates is of significance in the broader context of the advancement of logic theory and development along lines not constrained by classic Boolean (AND/OR) techniques.

The function $f(x)$ represented by a single-output threshold-logic gate is dependent upon the number of binary inputs x_i, $1 \leqslant i \leqslant n$, the weighting factor a_i of each input, and the gate output threshold value t, $1 \leqslant t \leqslant \sum_{i=1}^{n} a_i x_i$. The gate output is given by the summation:

$$f(x) = 1 \text{ if } \sum_{i=1}^{n} a_i x_i \geqslant t ,$$
$$= 0 \text{ otherwise,}$$

which may also be expressed in the form

$$f(x) = \langle a_1 x_1 + a_2 x_2 + \ldots + a_n x_n \rangle_t$$

Such a gate may be used to realise any linearly-separable Boolean function by suitable choice of the gate parameters. Further details of existing theory may be found in several sources (Dertouzos, 1965; Hurst, 1969; Muroga, 1971).

If a given Boolean function is not linearly-separable, it cannot by definition be directly realised by one single-output threshold logic gate, and a multilevel realisation of some kind becomes necessary. Dertouzos (1965) and others have shown that the Rademacher–Walsh transform may be applied to Boolean functions, whilst Edwards (1975) has used the spectral domain resulting from the Rademacher–Walsh transform to show that further classifications of non-linearly-separable functions may be transformed into a linearly-separable class by the application of the Exclusive-OR operator. Further developments lead on to the consideration of threshold-logic gates with multiple outputs, each output being associated with a particular output threshold value t_j, giving rise to the powerful concept of multioutput threshold-logic gates (Haring and Diephuis, 1967; Hampel, 1973; Hurst, 1976). At least one practical usage of this concept has been pursued (Wooley and Baugh, 1974). The theoretical potential of the multioutput threshold-logic gate as a generalised circuit element is therefore established.

The circuit implementation of multioutput gates, however, remains an area of invention. Analogue summation techniques within the gate structure to achieve the output discrimination are possible, but with increasing difficulty when multioutput facilities are required. All-digital circuit techniques, giving rise to Digital-Summation Threshold-Logic ('DSTL') gates, show advantages in most respects (Hurst, 1973; Winn, 1975).

The first of the DSTL circuit proposals consists of a regular matrix array of identical two-input cells, the number of cells required being a function of the number of inputs and maximum gate threshold value required (Hurst, 1973). The signal propagation time through the matrix is also a function of these parameters, and hence for a large number of inputs or a high output threshold value this time may be excessive for certain applications. Against this factor, advantages include the features that every gate threshold is directly available from the full matrix, thus providing multioutput capability; also the matrix array is perfectly regular, involving no crossovers of any cell interconnection paths.

The alternative circuit realisation proposed by Winn (1975) overcomes the propagation delay problem, but is not a regular array configuration. The gate is instead implemented in a product-of-sums form, such that the theoretical maximum path length through the gate is that of three simple Boolean operations in cascade, independent of the number of inputs or gate threshold value required. The total number of Boolean operations per gate, however, may be larger than that of the matrix assembly, whilst no direct provision for multioutput capability is present.

Several developments have been proposed to modify the regular structure cellular array DSTL gate to attempt to im-

*Formerly University of Bath, now MOD Directorate of Communications Projects, London.
†School of Electrical Engineering, University of Bath, Claverton Down, Bath, BA2 7AY.

prove its performance (Reddy and Swamy, 1974; Edwards, 1978). All however involve loss of the perfectly regular matrix configuration and introduce crossovers of signal paths, although on balance overall improvements are achieved with certain modifications. Further developments of the sum-of-products DSTL concept of Winn, particularly to provide multioutput capability whilst retaining the low propagation delay, are now disclosed in the following sections.

2. The extension of Winn's technique to multioutput gates

Before considering the synthesis of multioutput threshold-logic gates using the basic concepts of Winn, let us very briefly summarise his approach as applied to single-output gate assemblies.

Consider the threshold-logic gate shown in **Fig. 1**(*a*), which has six unity-weighted inputs $x_1, \ldots, x_6$ and a gate output threshold of $t = 2$. The Boolean relationship which such a gate realises is:

$$f(x) = [x_1x_2 + x_1x_3 + x_1x_4 + x_1x_5 + x_1x_6 + x_2x_3 + \ldots + x_5x_6] ,$$

which involves in total 15 first-level product (AND) terms plus the 15-input second-level sum (OR) term. However if this sum-of-products expression is re-expressed in a product-of-sums form, we may have

$$\begin{aligned} f(x) = [&(x_1 + x_2 + x_3)(x_4 + x_5 + x_6) \\ &+ (x_1 + x_2 + x_4)(x_3 + x_5 + x_6) \\ &+ (x_1 + x_5)(x_2 + x_6)] , \end{aligned}$$

which now involves only 10 Boolean operators in a 3-level realisation, as shown in Fig. 1(*b*).* Winn's work (1975) consisted in determining optimum product-of-sums factorisations for a wide range of functions, with minimisation of the number of sum terms being the principle criteria. This optimisation however was made considering one gate specification, that is one threshold value t, at a time.

Should more than one threshold output be required from a given set of x_i inputs, then clearly some sharing of the Winn sum terms is desirable. However this may now mean that the minimum set of such terms for each specific threshold is no longer the previous optimum, and alternative factorisations may allow a better sharing of terms between the multioutput requirements.

Consider a multioutput DSTL gate with eight unity-weighted inputs x_1 to x_8 inclusive, and three threshold outputs, $t = 2$, $t = 3$ and $t = 4$. The optimised Winn product-of-sums realisation for each threshold considered independently is as follows:

(i) for $t = 2$,

$$\begin{aligned} f(x) &= \langle x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 \rangle_2 : \\ f(x)_{t=2} &= [(x_1 + x_2 + x_3 + x_4)(x_5 + x_6 + x_7 + x_8) \\ &\quad + (x_1 + x_2 + x_5 + x_6)(x_3 + x_4 + x_7 + x_8) \\ &\quad + (x_1 + x_3 + x_5 + x_7)(x_2 + x_4 + x_6 + x_8)], \\ &= 10 \text{ Boolean operations} \end{aligned}$$

(ii) for $t = 3$,

$$\begin{aligned} f(x) &= \langle x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 \rangle_3 : \\ f(x)_{t=3} &= [(x_1 + x_2 + x_3)(x_4 + x_5 + x_6)(x_7 + x_8) \\ &\quad + (x_1 + x_4 + x_7)(x_2 + x_5 + x_8)(x_3 + x_6) \\ &\quad + (x_2 + x_5 + x_7)(x_3 + x_4 + x_8)(x_1 + x_5) \\ &\quad + (x_1 + x_6 + x_8)(x_3 + x_5 + x_7)(x_2 + x_4)] , \\ &= 17 \text{ Boolean operations} \end{aligned}$$

(iii) for $t = 4$,

$$\begin{aligned} f(x) &= \langle x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 \rangle_4 : \\ f(x)_{t=4} &= [(x_1 + x_2)(x_3 + x_4)(x_5 + x_6)(x_7 + x_8) \\ &\quad + (x_1 + x_3)(x_2 + x_5)(x_4 + x_7)(x_6 + x_8) \\ &\quad + (x_1 + x_2)(x_3 + x_8)(x_4 + x_6)(x_5 + x_7) \\ &\quad + (x_1 + x_6)(x_2 + x_8)(x_3 + x_5)(x_4 + x_7) \\ &\quad + (x_1 + x_7)(x_2 + x_4)(x_3 + x_5)(x_6 + x_8) \\ &\quad + (x_1 + x_3)(x_2 + x_5)(x_4 + x_6)(x_7 + x_8) \\ &\quad + (x_1 + x_4)(x_2 + x_6)(x_3 + x_7)(x_5 + x_8)] , \\ &= 28 \text{ dissimilar Boolean operations.} \end{aligned}$$

Thus a simple assembly of the above in one package with common inputs to provide $t = 2$, $t = 3$ and $t = 4$ would use a total of 55 Boolean operations. A quick inspection, however, will reveal two factors in common in the $t = 3$ and $t = 4$ listings, thus giving a slightly reduced total of 53 Boolean operations.

If the $t = 2$ expression is rewitten as

$$\begin{aligned} f(x)_{t=2} = [&\{(x_1 + x_2) + (x_3 + x_4)\} \\ &\{(x_5 + x_6) + (x_7 + x_8)\} \\ &+ \{(x_1 + x_2) + (x_5 + x_6)\} \\ &\{(x_3 + x_4) + (x_7 + x_8)\} \\ &+ \{(x_1 + x_3) + (x_5 + x_7)\} \\ &\{(x_2 + x_4) + (x_6 + x_8)\}] , \end{aligned}$$

the resulting two-input OR functions are now all available from the $t = 4$ realisation. Unfortunately to sum these 2-input OR terms to make the required 4-input OR terms for $t = 2$ would reintroduce as many additional Boolean operations as were saved by using these common factors.

Tests will readily show that no overall savings are achieved

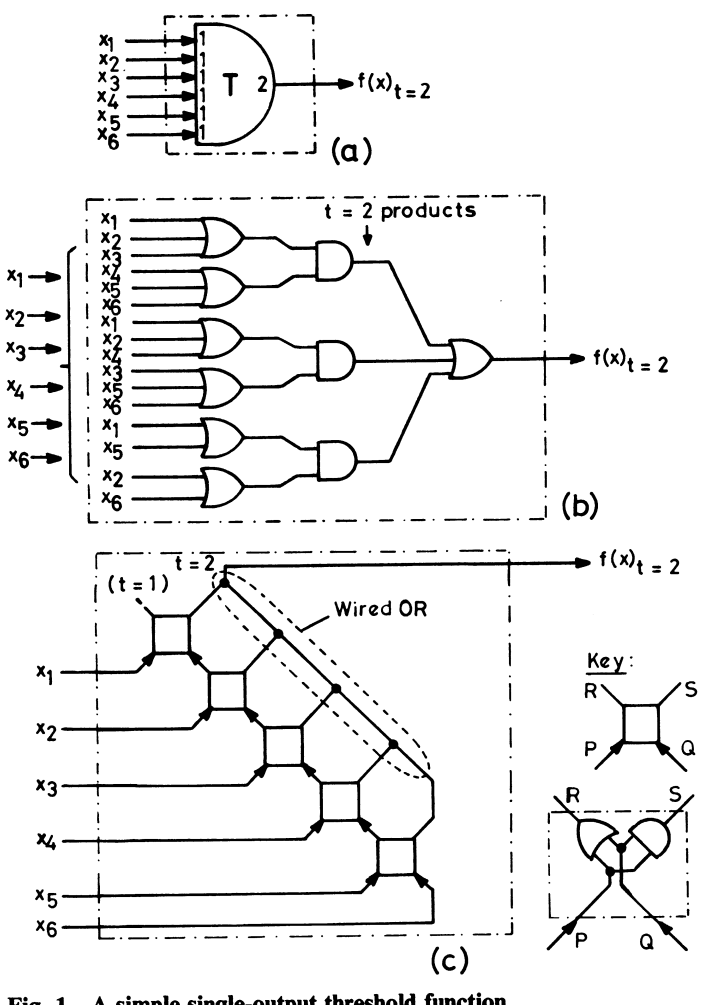


Fig. 1 **A simple single-output threshold function** $f(x) = \langle x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \rangle_2$
(*a*) **the threshold-logic gate**
(*b*) **Winn's product-of-sums DSTL realisation**
(*c*) **the equivalent cellular-array DSTL realisation, truncated to amputate threshold outputs higher than** $t = 2$

*Such product-of-sums expressions may of course be converted into all-NAND or all-NOR form by the application of De Morgan's laws, but this will not reduce the total number of terms from that given in the minimal product-of-sums expression.

by merely factorising all the sum terms of published Winn functions into smaller sum-of-sum expressions, in order to obtain common factors for different threshold outputs. Other means of employing factors of common significance must be sought, as will now be considered.

3. Detailed considerations of extracting factors

From previous publications (Winn, 1975; Eves, 1976) and from the above examples it will be noted that an optimised Winn function for any given output threshold value t consists of several second-level t-input product (AND) terms, the inputs to these product terms collectively involving the input variables x_1 to x_n. For example see Fig. 1(b). If we therefore take a Winn function with a given t value, it may be relevant to select less than t of the second-level input signals to form components for a function of threshold value less than t. This concept is illustrated in **Fig. 2**(a). Notice that this does not increase the total number of input/output gating levels in either threshold output above the optimised 3-level configuration.

Alternatively, or in addition, it may be relevant to sum (OR) together two (or more) of the higher-threshold second-level input signals to provide alternative inputs for the lower threshold product terms, as illustrated in Fig. 2(b). Notice that unlike the previous arrangement we have now introduced an additional level of gating in the lower-t output realisation above that of the original optimised Winn configurations.

Should more than two gate outputs be required then selection of factors from a $< t$-valued Winn function and from a $> t$-valued Winn function may be possible in order to form an intermediate t-valued threshold function. This concept is illustrated in **Fig. 3**, from which it will be noted that if this is possible then no increase in the number of gating levels is present for any threshold output.

Examples of multithreshold realisations using these techniques may readily be constructed to illustrate such possibilities, but to date it has not proved possible to develop any design algorithms which converge towards an optimal or near-optimal solution. Instead methods to date have been semi-exhaustive computer searches and listings (Eves, 1976), similar to those employed by Winn in his work for the single-output cases. Further fundamental work in this rather intractable area of optimal minimisation of multioutput networks would be of great interest and significance, particularly if not confined to linearly-separable situations; possible approaches may be suggested using n-dimensional geometric considerations (Eves, 1976), or certain matrix methods (Robinson and Hoffner, 1975), or possibly methods involving considerations of symmetry (Hurst, 1977), but none of these so far seems to generate a 'best' solution without trial runs.

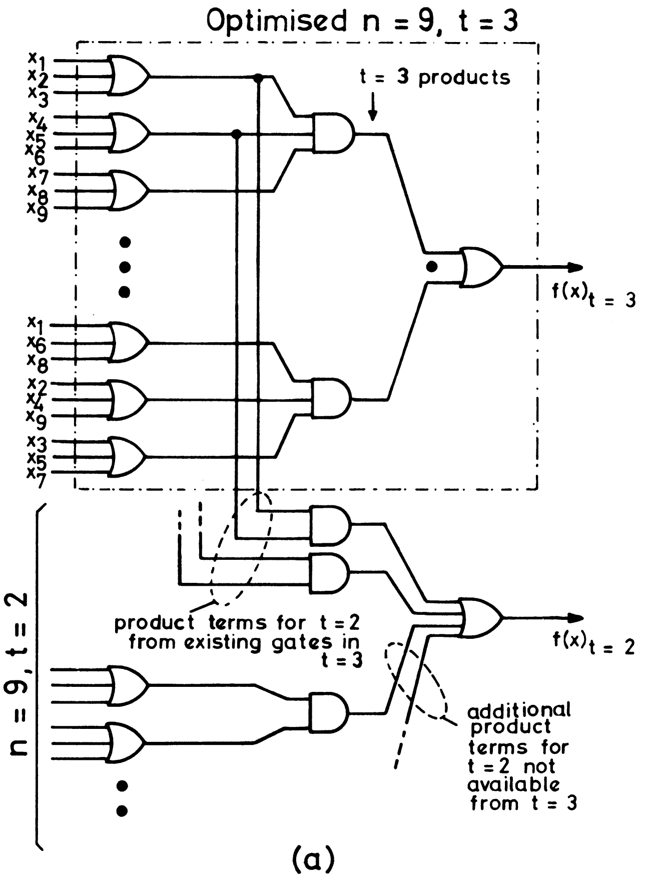
Examples of results which have been obtained by semi-exhaustive computer searches on simple examples are as follows, from which further observations and developments will be made.

Example: $n = 8$, three outputs $t = 4$, 3 and 2

The individual optimised Winn functions for these three outputs have been given in Section 2. Separately they involve the following Boolean operations:

$n = 8, t = 2$:	six sum terms, three product terms, one final summation, total 10
$n = 8, t = 3$:	twelve sum terms, four product terms, one final summation, total 17
$n = 8, t = 4$:	twenty sum terms, seven product terms, one final summation, total 28

With the three-level realisation method of Fig. 2(a) it is possible to synthesise both the $t = 2$ and the $t = 3$ outputs from



Optimised n = 9, t = 3
t = 3 products
x1 x2 x3 x4 x5 x6 x7 x8 x9
x1 x6 x8 x2 x4 x9 x3 x5 x7
f(x)t = 3
n = 9, t = 2
product terms for t = 2 from existing gates in t = 3
f(x)t = 2
additional product terms for t = 2 not available from t = 3
(a)

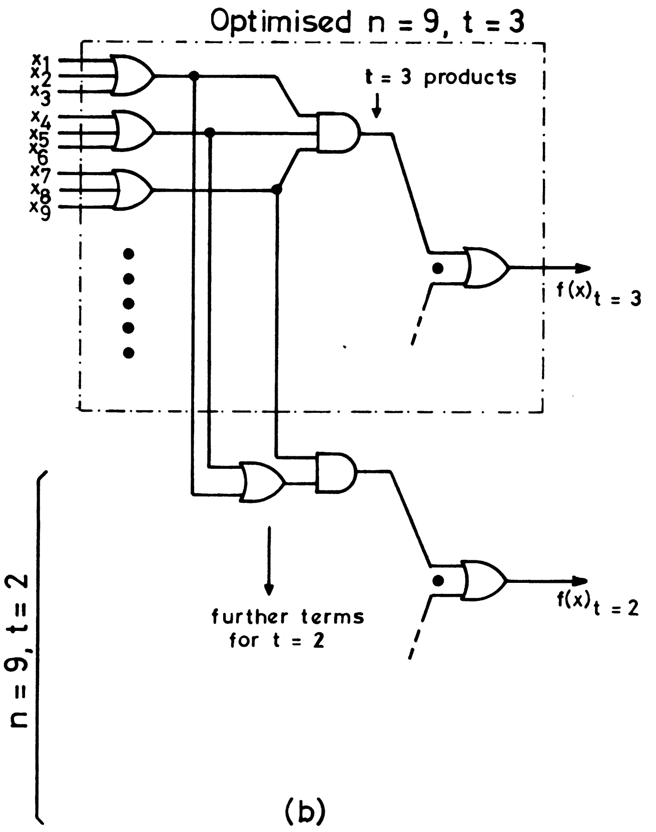


Fig. 2 The possibilities of factors from a gate of threshold t to produce gates of threshold $< t$
(a) extraction of product input signals from t to form direct product input signals for $< t$
(b) extraction and summation of product input signals from t to form product input signals for $< t$

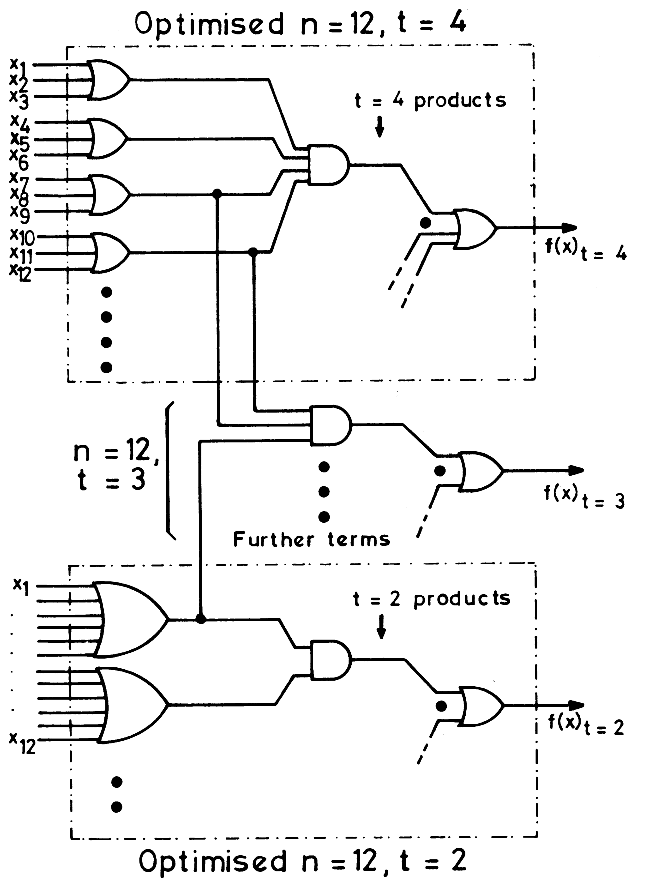


Fig. 3 The possibility of factors for threshold t from Winn functions of $< t$, $> t$

the $t = 4$ Winn function factors, without generating any additional OR (sum) terms. The realisations are as follows:

$$
\begin{aligned}
f(x)_{t=2} = [&(x_1 + x_2)(x_7 + x_8) + (x_1 + x_2)(x_5 + x_6) \\
&+ (x_1 + x_2)(x_3 + x_4) + (x_3 + x_4)(x_7 + x_8) \\
&+ (x_3 + x_4)(x_5 + x_6) + (x_5 + x_6)(x_7 + x_8) \\
&+ (x_1 + x_3)(x_2 + x_4) + (x_5 + x_7)(x_6 + x_8)]
\end{aligned}
$$

$$
\begin{aligned}
f(x)_{t=3} = [&(x_1 + x_2)(x_3 + x_4)(x_5 + x_6) \\
&+ (x_1 + x_2)(x_3 + x_4)(x_7 + x_8) \\
&+ (x_1 + x_2)(x_5 + x_6)(x_7 + x_8) \\
&+ (x_3 + x_4)(x_5 + x_6)(x_7 + x_8) \\
&+ (x_3 + x_5)(x_4 + x_7)(x_6 + x_8) \\
&+ (x_3 + x_8)(x_4 + x_6)(x_5 + x_7) \\
&+ (x_1 + x_2)(x_5 + x_7)(x_6 + x_8) \\
&+ (x_1 + x_3)(x_2 + x_5)(x_4 + x_6) \\
&+ (x_1 + x_6)(x_2 + x_4)(x_3 + x_5) \\
&+ (x_1 + x_3)(x_2 + x_4)(x_6 + x_8)] \,.
\end{aligned}
$$

With the four-level additional OR realisation method of Fig. 2(*b*), several equally optimal realisations for $t = 2$ and $t = 3$ using the $t = 4$ Winn factors are available. For example:

$$
\begin{aligned}
f(x)_{t=2} = [&\{(x_1 + x_2) + (x_3 + x_4) + (x_5 + x_6)\}\{x_7 + x_8\} \\
&+ \{(x_5 + x_6) + (x_7 + x_8)\}\{x_3 + x_4\} \\
&+ (x_5 + x_6)(x_7 + x_8) \\
&+ (x_1 + x_3)(x_2 + x_4) + (x_5 + x_7)(x_6 + x_8)] \,.
\end{aligned}
$$

$$
\begin{aligned}
f(x)_{t=3} = [&\{(x_1 + x_2) + (x_3 + x_4)\}(x_5 + x_6)(x_7 + x_8) \\
&+ (x_1 + x_2)(x_3 + x_4)\{(x_5 + x_6) + (x_7 + x_8)\} \\
&+ \{(x_1 + x_2) + (x_3 + x_4)\}(x_5 + x_7)(x_6 + x_8) \\
&+ (x_1 + x_3)(x_2 + x_4)\{(x_5 + x_6) + (x_7 + x_8)\} \\
&+ \{(x_1 + x_2) + (x_7 + x_8)\}(x_3 + x_5)(x_4 + x_6) \\
&+ (x_1 + x_7)(x_3 + x_8)\{(x_2 + x_4) + (x_5 + x_6)\} \\
&+ (x_1 + x_3)(x_2 + x_5)\{(x_4 + x_6) + (x_7 + x_8)\}] \,.
\end{aligned}
$$

However from such results when we compute what saving in the total number of Boolean functions has been achieved, the results are not encouraging. For the above example we may compile **Table 1.**

Further examples similar to that above show that only very small savings in the total number of Boolean operators can be found by searching for common sum-of-product factors for a three or four-level network realisation. The necessary recombining of common factors to realise the threshold outputs tends to require almost as many operators as present in separate single-output Winn realisations. Particularly unproductive is the search for useful factors from a Winn function of threshold t to realise any further function of higher threshold value.

If now we consider using more than one optimised Winn function for three (or more) threshold outputs, we may consider network realisations as suggested in Fig. 3. Again, however, little saving has been found by merely searching for useful common factors, but an alternative approach can now be suggested, involving modulo-two summation considerations.

4. Modulo-two (parity) detection

Modulo-two summation of two or more binary inputs x_i is logically equivalent to the Exclusive-OR of the inputs. This is also equivalent to odd-parity detection. Similarly modulo-two summation with output inversion is equivalent to Exclusive-NOR, which in turn is the same as even-parity detection. This is indicated in **Fig. 4.** It will be noted that if 2-input Exclusive-OR or NOR functions are considered to be basic logic gates, as will be discussed in a following Section, then n-input odd and even parity realisations are possible from a $\log_2 n$ level assembly, e.g. 3 levels for eight inputs, etc.

A single odd-parity circuit will therefore detect when 1, 3, 5, 7, . . . of n binary inputs are at logic 1, but will not distinguish between these conditions. Similarly an even-parity circuit will detect but not distinguish between 0, 2, 4, 6, . . . inputs at logic 1. From this springs the possibility of using separate optimised Winn functions to generate *every other* threshold output, with one common parity circuit to enable detection of the in-between thresholds to be accomplished. **Fig. 5** illustrates such possibilities.

Since the number of Boolean operators per Winn function increases steeply with increasing value of threshold t, it is generally desirable to employ the common parity circuit to realise the *highest* threshold value required in any specific application. Thus Fig. 5(*a*) is most relevant where $t_{\max}$ is odd-valued, whilst Fig. 5(*b*) is most relevant for $t_{\max}$ even-valued. The maximum number of gating levels in either realisation will be seen to be two levels above the Winn or parity function for the threshold outputs not provided directly

Table 1 A comparison of the different DSTL techniques so far considered for realising a $n = 8$, $t = 2$, 3 and 4 gate (Note, the cellular array realisation also provides a $t = 1$ threshold output)

Realisation	*Max. no. of logic levels*	*Total no. of Boolean operators*
Entirely separate Winn functions for $t = 2$, 3 and 4	3	55
Sharing of any duplicated factors in the above optimised Winn functions	3	53
Realisations as in Fig. 2(*a*), based upon the $t = 4$ Winn function factors	3	48
Realisations as in Fig. 2(*b*), based upon the $t = 4$ Winn function factors	4	51
Cellular array DSTL realisation as in Fig. 1(*c*), truncated at $t = 4$ by one OR function	10	37

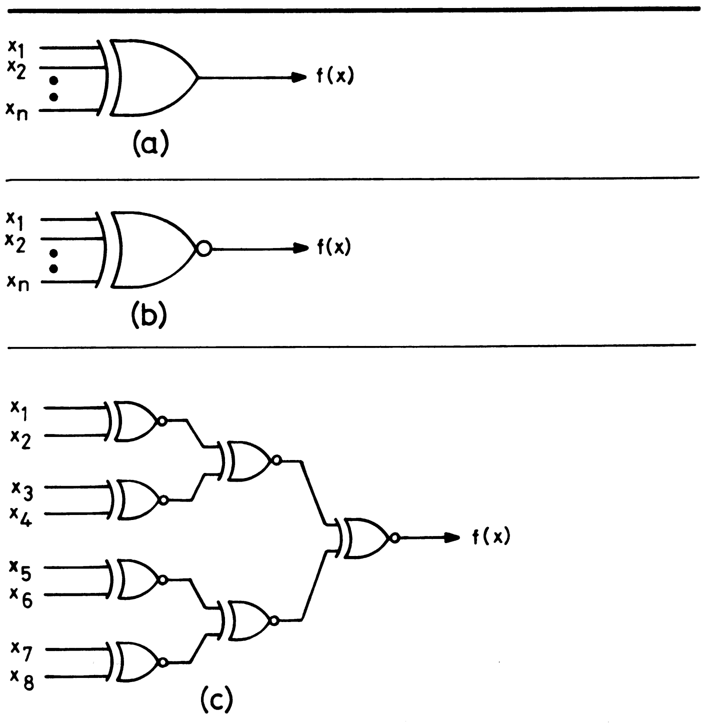


Fig. 4 Modulo-two networks
(*a*) **odd parity (multi-Exclusive-OR) gate**
(*b*) **even parity (multi-Exclusive-NOR) gate**
(*c*) **8-input 3-level even-parity gate compiled from 2-input Exlusive-NOR functions**

by the Winn functions. Hence the representative statistics of **Table 2** may be compiled. Note that the Boolean operator count in the relevant optimised Winn functions is as follows (Winn, 1975):

$n = 8,$	$t = 2$	10 operators,
$n = 8,$	$t = 3$	17 operators,
$n = 8,$	$t = 4$	28 operators,
$n = 8,$	$t = 5$	37 operators,
$n = 12,$	$t = 2$	13 operators,
$n = 12,$	$t = 3$	29 operators,
$n = 12,$	$t = 4$	59 operators,
$n = 12,$	$t = 6$	93 operators.

In contrasting the selected results detailed in Tables 1 and 2 it will be seen that the techniques of Fig. 5 are in general preferable to the previous methods of searching for common factors in the Winn functions. The principal disadvantage is that a small increase in the maximum number of gating levels occurs in the later method. However where a large number of inputs and/or outputs are involved, for example $n = 8, t = 1$ to 5 or the $n = 12$ cases, then the DSTL array structure of Fig. 1(*c*) still proves more compact, though at the expense of longer maximum gate propagation times.

5. A comparison of design techniques

It may be noted with interest that the preferable topology of Fig. 5 in comparison with the concepts of Figs. 2 and 3 was developed without the guidance of any Boolean algebraic relationships. The system structure rather than the detailed binary signals were factors of significance. The odd and the even parity functions may certainly be given Boolean expressions to define their input/output relationships, from which final threshold output equations may be derived, but algebraic synthesis of the particular configurations of Fig. 5 back from the Boolean output equations is not straightforward.

In a similar manner, the regular matrix array type of DSTL gate of Fig. 1(*c*) was not developed by any Boolean algebraic technique, but rather was built up by considering signal path routing (Hurst, 1973). Should we consider the detailed Boolean equations involved in such DSTL arrays, which was not done in the original disclosures, then for, say, a six-input ($n = 6$) array as shown in Fig. 1(*a*) we have the equations following.

(i) output of the first row of the array ($t = 1, = \text{OR}$):

$$f(x)_{t=1} = [((((x_6 + x_5) + x_4) + x_3) + x_2) + x_1]$$

(ii) output of the next row of the array ($t = 2$):

$$\begin{aligned} f(x)_{t=2} = [\{(x_6x_5)\} &+ \{(x_6 + x_5)x_4\} \\ &+ \{((x_6 + x_5) + x_4)x_3\} \\ &+ \{(((x_6 + x_5) + x_4) + x_3)x_2\} \\ &+ \{((((x_6 + x_5) + x_4) + x_3)x_2)x_1\}] \,. \end{aligned}$$

(iii) output from the third row of the array ($t = 3$), if not truncated at $t = 2$:

$$f(x)_{t=3} = [y_1y_2 + y_2y_3 + y_3y_4 + y_4y_5]$$

where y_1, y_2, y_3, y_4 and y_5 are the five terms in order in the { } brackets of the previous $f(x)_{t=2}$ equation.

Similarly for the remaining $t = 4$, 5 and 6 outputs of the complete untruncated array. The algebraic build-up of the individual threshold outputs is clear from this analysis, but the automatic synthesis of this realisation back from the given output equations is not possible with available algebraic techniques, short of exhaustive searching for the common

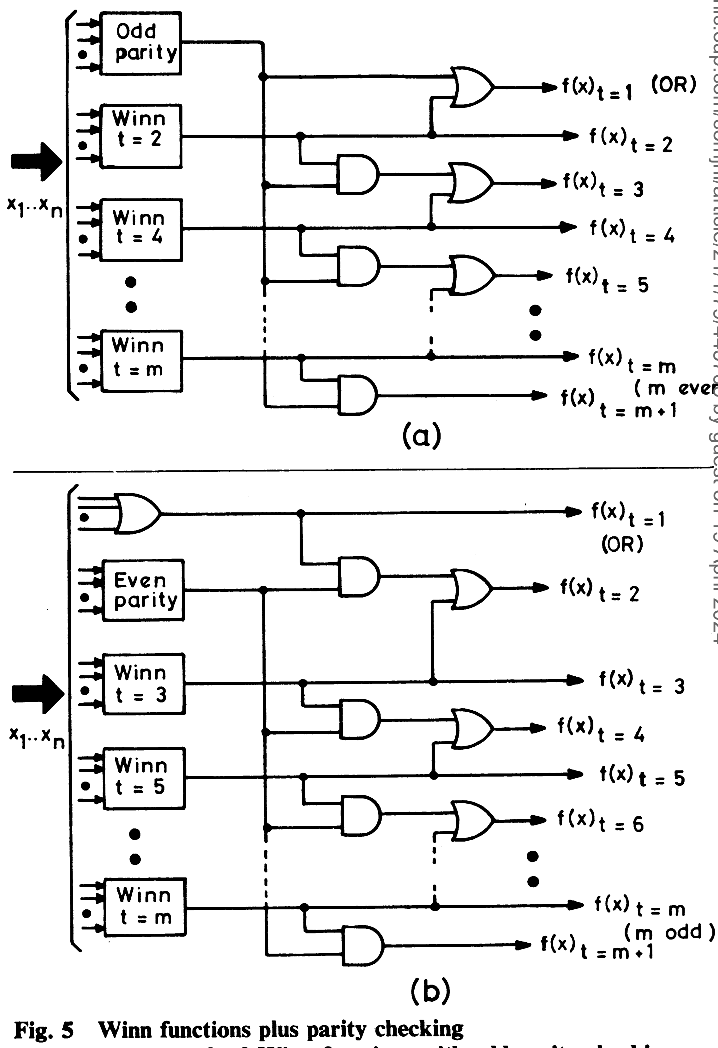


Fig. 5 Winn functions plus parity checking
(*a*) **even-valued Winn functions with odd-parity checking**
(*b*) **odd-valued Winn functions with even-parity checking**

Table 2 A comparison of the techniques of Fig. 5 with the array structure of Fig. 1(*c*) in providing multithreshold outputs from various $n = 8$ and $n = 12$ input sets

Multithresholds required	*Type of realisation*	*Total no. of Boolean operators*	*Maximum no. of logic levels, assuming 3 logic levels per parity unit in Winn assemblies*
$n = 8$	Or + Winn 2	11	4
$t = 1$ and 2	Truncated DSTL array	15	8
$n = 8$	Odd-parity + Winn 2	19	4
$t = 1$, 2 and 3	Truncated DSTL array	27	9
$n = 8$	Or + Even-parity + Winn 3	29	5
$t = 1$ to 4 inclusive	Truncated DSTL array	37	10
$n = 8$	Odd-parity + Winn 2 + Winn 4	49	5
$t = 1$ to 5 inclusive	Truncated DSTL array	45	11
$n = 8$	Or + Even-parity + Winn 3 + Winn 5	68	5
$t = 1$ to 6 inclusive	Truncated DSTL array	51	12
$n = 12$	Or + Even-parity + Winn 3	41	6
$t = 1$ to 4 inclusive	Truncated DSTL array	31	14
$n = 12$	Odd-parity + Winn 2 + Winn 4 + Winn 6	178	6
$t = 1$ to 7 inclusive	Truncated DSTL array	52	17

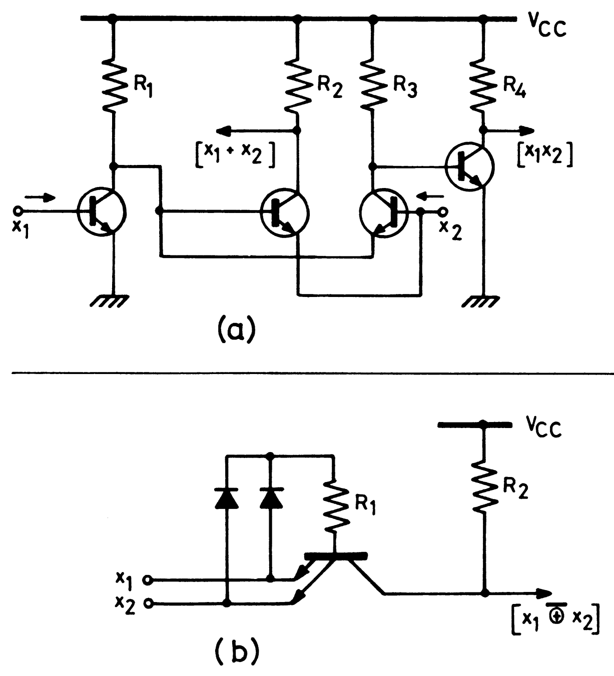


Fig. 6 Possible monolithic function realisations
(*a*) an AND/OR cell
(*b*) an Exclusive-NOR gate
(Note these circuits may be covered by patent rights)

factorisations.

It may therefore be concluded from several aspects of this work that Boolean algebra provides a very limited means of synthesis of multioutput logic networks, particularly where three or more levels of realisation are permitted and when maximum sharing of terms between several outputs is desirable. More powerful synthesis techniques are needed in this area; possibly developments of symmetry relationships may prove to be of significance in this area (Edwards, 1978; Hurst and Edwards, 1976).

6. Comments and conclusions

Digital methods of realising threshold-logic gates with one or more outputs may at first appear to be a fundamentally unprofitable exercise, as a considerable number of Boolean operations may be involved per threshold output (see Tables 1 and 2 for example); however with monolithic circuit realisations the actual number of Boolean operations per circuit is not of supreme significance—flexibility of the input/output logic capability per package is of greater importance. In this respect multioutput threshold gates form just one part of ongoing research into various possible general purpose logic packages (Hurst, 1976; Tabloski and Mowle, 1976; Kautz, 1971).

Further, one should certainly not regard an AND operation or an OR operation in a Boolean equation as necessarily implying that these must be a separate identifiable AND gate or a separate identifiable OR gate corresponding to each Boolean operator. For example, it may be possible to realise the AND/OR cell of Fig. 1(*a*), which we have counted as two separate Boolean terms, by one combined assembly, such as suggested in **Fig. 6**(*a*). Exclusive functions should also not be regarded as assemblies of separate Boolean gates, but may be regarded as circuit configurations in their own right, as for example shown in Fig. 6(*b*).

Thus although one may express the final design of a complex gate in conventional AND/OR or NAND or NOR algebraic form, the translation of such expressions into a monolithic realisation may not involve a one-to-one translation. However without considering in detail the precise final monolithic realisation, a count of the number of Boolean operations and number of logic levels forms a ready initial comparison of the efficiency of one gate design against another. But to revert to the main area of this paper, it has not proved possible to refactorise the single-output Winn threshold functions in order to optimally realise multioutput threshold functions; instead the use of alternate-value Winn functions plus parity functions provides a more viable alternative.

In the wider context of multioutput functions, covering both threshold and non-threshold functions, more powerful methods of synthesis are still required, which (ideally) can optimise function design with any chosen number of levels of realisation.

Acknowledgements

The work on considering Winn functions for multioutput threshold realisations was undertaken as a Final Year B.Sc. Degree project in the School of Electrical Engineering, University of Bath.

References

Dertouzos, M. L. (1965). *Threshold logic: a synthesis approach*, Research Monograph No. 32, MIT Press.

EDWARDS, C. R. (1975). The application of the Rademacher-Walsh transform to Boolean function classification and threshold-logic synthesis, *IEEE Trans*, Vol. C.24, pp. 48-62.
EDWARDS, C. R. (1978). Some improved designs for the digital-summation threshold-logic (DSTL) gate, *The Computer Journal*, Vol. 21, No. 1, pp. 73-78.
EVES, J. M. (1976). Multioutput digital summation threshold logic, B.Sc. Final Year project report, Ref. EE 12/76, School of Electrical Engineering, University of Bath.
HALLIGAN, J. (1974). Using Majority logic blocks, *Electronic Equipment News*, Vol. 16, Nov. 1974, p. 105.
HAMPEL, D. (1973). Multifunction threshold gates, *IEEE Trans*, Vol. C.22, pp. 197-203.
HARING, D. R. and DIEPHUIS, R. J. (1967). A realization procedure for multi-threshold threshold elements, *IEEE Trans*, Vol. EC.16, pp. 828-835.
HURST. S. L. (1969). An introduction to threshold logic: a survey of present theory and practice, *The Radio and Electronic Engineer*, Vol. 37, pp. 339-351.
HURST, S. L. (1973). Digital-summation threshold-logic gates: a new circuit element, *Proc. IEE*, Vol. 120, pp. 1301-1307.
HURST, S. L. (1976). Application of multioutput threshold-logic gates to digital-network design, *Proc. IEE*, Vol. 123, pp. 297-306.
HURST, S. L. (1977). *The logical processing of digital signals*, New York: Crane-Russak.
HURST, S. L. and EDWARDS, C. R. (1976). Preliminary consideration of the design of combinatorial and sequential digital systems under symmetry methods, *International Journal of Electronics*, Vol. 40, No. 5, pp. 499-507.
KAUTZ, W. H. (1971). Programmable cellular logic, in *Recent Developments in Switching Theory*, ed. A. Mukhopadhay, Academic Press.
MUROGA, S. B. (1971). *Threshold logic and its applications*, New York: Wiley Interscience.
REDDY, V. C. V. P. and SWAMY, P. S. N. (1974). Note on digital-summation threshold-logic gates, *Proc. IEE*, Vol. 121, pp. 1085-1086.
ROBINSON, J. P. and HOFFNER, C. W. (1975). Easily-tested three-level gate networks for *T* or more of *N* symmetric functions, *IEEE Trans*, Vol. C.24, pp. 331-335.
TABLOSKI, T. E. and MOWLE, F. J. (1976). A numerical expansion technique and its applications to minimal multiplexer circuits, *IEEE Trans*, Vol. C.25, pp. 684-702.
WINN, G. C. E. (1975). A digital approach to the efficient synthesis of threshold gates, *The Computer Journal*, Vol. 18, pp. 239-242.
WOOLEY, B. A. and BAUGH, C. R. (1974). An integrated *m*-out-of-*n* detection using threshold logic, *IEEE Trans*, Vol. SC. 9, pp. 297-306.

Book reviews

Software Metrics, by Tom Gilb, 1977; 282 pages. (*Prentice-Hall*, £11·50)

This is a curate's egg of a book if ever there was one. Those who read Tom Gilb's articles in the computer press in late 1975 will already be familiar with some of the ideas to be found in this book, such as 'bebugging' (the deliberate seeding of errors in a program) and 'dual coding' (two versions of a program written completely independently).

Gilb's aim is the cost-effective design, construction and maintenance of software. He seeks to apply certain 'metrics' in order to monitor the quality of software. 'Bebugging', for instance, may be used to discover the total number of bugs in a program. You sow the program with a fixed number of bugs and set someone to find bugs. From among the bugs discovered you then find out how many were deliberately implanted. This gives you some idea of the number of bugs actually present before you started. It is, in fact, similar to the method of finding the total number of fish in a lake by introducing a certain number of tagged fish, and then going fishing.

Such a simplistic view of things does not, of course, take into account the fact that bugs (unlike fish) may be astonishingly different from one another in form. Two cards in the wrong order may be responsible for one bug, but another may be due to a mismatch in the interface between modules, and a third may only show up as an inadequate fix-up after a rare combination of invalid data. If you think that the author concerns himself with such awkward complications, you are wrong. He wants a metric, and therefore to him a bug is a bug is a bug.

In fact, sometimes even bigger corners are cut in the search for something which he can call a 'metric'. In Fig. 32 (p. 69) the metric corresponding to 'EFFECTIVENESS' is 'Transactions per cost-unit' (which is very reasonable), but the metric corresponding to 'ROBUSTNESS' is 'garbage in does not lead to G. out' which is hardly something one can measure (not to speak of the inadequacies in punctuation).

There are without doubt nuggets of wisdom hidden in this book, but the form of the book and its style tend to keep them hidden. Perhaps it is hardly surprising that one who disdains structured programming should produce a book with so little structure. The chapters are not numbered, and no page has a running heading, so one has to resort to a minute comparison of type sizes to determine when a new chapter is indeed starting. One half of the book (Part II) reads like a collection of notes and jottings. Many diagrams appear to be untouched reproductions of foils for an overhead projector.

As far as the style is concerned, the text seems to have been 'be-bugged'. On p. 60, for instance, the 'Hawthorne effect' is mentioned twice, but the term is only explained on its second occurrence, not the first. P. 60 also refers to 'lines of source code of PLS (abbreviated as LOC in the figure)'. What is PLS? (The meaning of APL is given on p. 86, but that of PLS is nowhere explained.) And who in his right mind would use LOC as an abbreviation for PLS? After no small amount of research, it dawned that PLS is a language and LOC is short for 'lines of code'.

These are typical of the obstacles with which the reader must contend in order to dig out the nuggets. Perhaps what is needed is for someone else to take the information and to write it up in a well-structured, easy-to-understand manner. Surely it would be fitting to apply to Gilb's own work one of his own concepts—that of dual coding.

COLIN DAY (London)

Digital Signal Processing in FORTRAN, by F. Taylor and S. L. Smith, 1976; 402 pages. (*Lexington Books*, £13·25)

My first reaction to the title of this book was negative. Here was another title in the already overburdened market for books about FORTRAN and its applications. However, a closer study of the text showed that this was a much more interesting book than usual. It describes a comprehensive package, written by the authors (in FORTRAN) providing digital filtering algorithms within an easy to use, user-oriented, software system called SPECTRUM IV. Full details are given of the structure and use of the package, and each section is accompanied by a thorough treatment of the mathematical and information-theoretic background of the algorithms in question.

The software components include features for spectral analysis (fast Fourier transform, autocorrelation and power spectral density), bivariate spectral transforms, and various digital filters. These sections are clearly presented with examples of the use of the programs, and the graphical output produced.

On the whole, however, I feel the book falls between two stools. It contains too much background to be a simple manual of SPECTRUM IV, but the reader seeking to understand digital filtering techniques is confused by the descriptions of card-decks required for its use.

The package was written initially for the CDC 3100, but was subsequently transferred to an IBM 360. The authors generously offer copies for general use at nominal cost, and an application form is included in the back of the book.

S. J. GOLDSACK (London)