

Analysis of input and output configurations for use in four-valued CCD programmable logic arrays

J.T. Butler
H.G. Kerkhoff

Indexing terms: Logic, Circuit theory and design, charge-coupled devices

Abstract: As in binary, a multiple-valued programmable logic array (PLA) realises a sum-of-products expression specified by the user. However, in multiple-valued logic, there are many more operations than in binary, and an important question is the choice of operations which provides the greatest number of functions for a given chip area. In this paper, we analyse various PLA configurations using operations realised in the peristaltic multiple-valued CCD technology. We compare a multiple-valued CCD PLA implementation with four other proposed designs and show that there is a significant difference in chip area required to realise the same set of functions. The basis of comparison is the set of 4-valued unary functions.

1 Introduction

In binary logic, an important circuit is the programmable logic array or PLA. Its widespread use is due to a flexibility which allows the user to specify complex combinatorial functions. Recently, PLAs have been suggested for multiple-valued systems [2–7]. It is likely that the advantages of the PLA will make it an important element in this domain as well. However, the wider choice of operations associated with a radix larger than 2, makes the determination of which PLA configuration to offer the user more difficult than in binary. Two PLA configurations were studied in Bender *et al.* [6], where it was shown that the use of the Sum operation at the second PLA level produces more economical realisations than with the Max operation over two classes of functions. However, the question of which operations are best suited for general multiple-valued functions remains open.

An implementation of a PLA in a radix higher than 2 is described in Kerkhoff and Butler [7]. The PLA is fabricated in CCD or charge-coupled device technology, and is the first implementation of a multiple-valued PLA, not only in CCD, but in any technology. The design is radix independent. That is, no primary circuit changes are required to accommodate changes in radix. Only voltage

values applied to the circuits need to be modified. This PLA is also the first multiple-valued peristaltic CCD. All previous multiple-valued CCD circuits have been surface-channel CCD. Because peristaltic CCD has a lower charge-transfer time, there is a correspondingly higher speed.

The focus in Reference 7 is the implementation of a multiple-valued PLA with an emphasis on device and circuit considerations. In this paper, we consider the logic design of the multiple-valued PLAs. We examine the question of what PLA configuration provides the greatest logic capability. The implementation of the PLA in Reference 7 is based on the Allen-Givone algebra [8] with the Max operation replaced by the Sum (Tirumalai and Butler [2]). It requires a circuit which implements the literal operator. In the implementation of Reference 7, the literal is realised by a pair of subcircuits, one realising the staircase-up and the other the staircase-down function. However, certain literal functions can be realised using only one staircase function, suggesting that more area-efficient PLAs can be built using staircase-function generators instead of literal-function generators. This intuitive notion is quantified in this paper. Specifically, we show that there is a significant reduction in chip area when such a substitution is made. Our analysis includes a total of five PLA designs, and we show that the design described in Reference 7 ranks midway (3rd out of 5) using the chip-area criteria.

All previously implemented CCD circuits have been in 4 values. This includes the peristaltic CCD implementation of Reference 7, although it is expected to work with more values, perhaps 6. Because our analysis requires the use of a specific radix, we choose radix 4.

In the next Section, we introduce the CCD logic operations, and in Section 3, we analyse the various PLA configurations with respect to the functions realised. A summary of results is given in Section 4.

2 CCD logic operations

2.1 Constant

Fig. 1 shows the four basic CCD operations used in the peristaltic CCD PLAs described here. The first three operations have been fabricated in surface-channel CCD technology (Kerkhoff [1]) in the same way as in peristaltic technology, while the last operation, Sum, has been realised in surface-channel CCD in a different form than shown here.

Fig. 1a shows a constant logic-value generator. The square on the left containing a black bar represents a source of charge which flows at the clock pulse into a charge well represented by the square labelled p . p is the

Paper 5443E(C2), first received 10th December 1985 and in revised form 23rd October 1986

J.T. Butler is with the Department of Electrical and Computer Engineering, Naval Postgraduate School, Monterey, CA 93943–5100, USA

H.G. Kerkhoff is with the IC-Technology and Electronics Group, Department of Electrical Engineering, Twente University of Technology, 7500AE Enschede, The Netherlands

Report Documentation Page				Form Approved OMB No. 0704-0188	
Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.					
1. REPORT DATE OCT 1986		2. REPORT TYPE		3. DATES COVERED	
4. TITLE AND SUBTITLE Analysis of input and output configurations for use in four-valued CCD programmable logic arrays				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School, Department of Electrical and Computer Engineering, Monterey, CA, 93943				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT As in binary, a multiple-valued programmable logic array (PLA) realises a sum-of-products expression specified by the user. However, in multiple-valued logic, there are many more operations than in binary, and an important question is the choice of operations which provides the greatest number of functions for a given chip area. In this paper, we analyse various PLA configurations using operations realised in the peristaltic multiple-valued CCD technology. We compare a multiple-valued CCD PLA implementation with four other proposed designs and show that there is a significant difference in chip area required to realise the same set of functions. The basis of comparison is the set of 4-valued unary functions.					
15. SUBJECT TERMS					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES 9	19a. NAME OF RESPONSIBLE PERSON
a. REPORT unclassified	b. ABSTRACT unclassified	c. THIS PAGE unclassified			

capacity of the well. The charge source fills the well completely, producing a quantity of charge equal to a logic p . The black bar to the right of this well is a clocked transfer gate, which transfers the well contents to the right and out of the constant generator. For example, if the well capacity is a logic 1, $p = 1$, and the constant generator produces a logic 1 at its output.

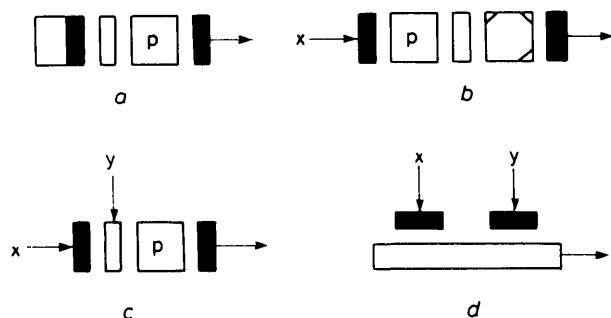


Fig. 1 Basic CCD operations

- a Constant (cost = 1)
- b Multithreshold (cost = 4)
- c Inhibit (cost = 18)
- d Metal summer (cost = 2)

2.2 Multithreshold

Fig. 1b shows a multithreshold or fixed-overflow circuit. At the clock pulse, the input charge at x is transferred into the well labelled p . If this charge exceeds the well capacity p , the excess flows right and into the next well. This has a capacity of logic 3, as indicated by the three corner braces. At the next clock pulse, the contents of this well are transferred out across the clocked transfer gate, represented by the vertical black bar, becoming the multithreshold output. The multithreshold circuit produces $x - p$ at the output if $x > p$ and 0 otherwise.

2.3 Inhibit

The inhibit circuit is shown in Fig. 1c. It has two inputs, a primary input x and a control input y . When a logic 0 appears at y , the logic value at the primary input x flows into the well labelled p . When the control input has a nonzero logic value, this flow is blocked, and a logic 0 appears at the inhibit output at the right. For $y = 0$, x passes to the output if $x \leq p$ and p passes to the output when $x > p$. Thus, the inhibit acts as a switch. The inhibit shown in Fig. 1c is a simplified version of that introduced in Reference 1 for surface-channel CCDs, where there is another output which is x when $y > 0$.

2.4 Sum

Fig. 1d shows the Sum operation. The metal line stores the charge which is dumped from inputs x and y at the clock pulse. The charge adds, creating a voltage which is proportional to the total charge. In the PLA configuration described below, the metal summers are the PLA columns. In the surface-channel CCD implementations [1], the Sum operation is performed in a well rather than a metal line.

2.5 Cost factors

The above four CCD operations are used in the PLA configurations to be described later. To compare configurations, we use relative cost factors of CCD operations, specifically those proposed in Reference 1. This cost is primarily a measure of the chip area occupied by the operation. For example, the constant generator requires much less chip area than the inhibit, and thus has a cor-

respondingly smaller relative cost. Other factors which determine the cost are the number of different power-supply and data lines, as well as the sensitivity of the realised function to process and voltage variations. The costs are integers and are shown in Fig. 1.

3 Analysis of realised functions

3.1 PLA components

In this section, we analyse the functions realised by the proposed PLA designs. The PLA structure has two basic components:

- (a) input configurations — logic required between primary PLA inputs and the PLA columns
- (b) output configurations — logic required between the PLA columns and the PLA output.

Two input configurations, staircase and step, and two output configurations, complement and inhibit, will be considered. Taken together, there are 4 combinations. These are compared to the configuration combination used in the multiple-valued PLA implementation of Reference 7, which consists of a literal-function input configuration and an inhibit output configuration. Thus, the five combinations are

Input configuration	Output configuration
(i) staircase functions	complement or direct
(ii) staircase functions	inhibit or direct
(iii) step functions	complement or direct
(iv) step functions	inhibit or direct
(v) literal functions	inhibit always [7]

3.2 Input configurations

3.2.1 Staircase-function generator: The first type of input configuration is shown in Fig. 2a. It consists of a

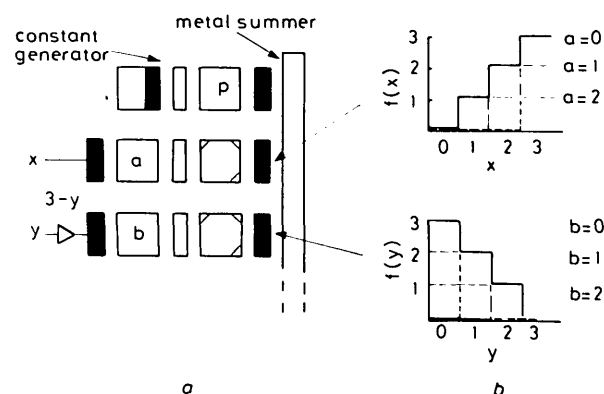


Fig. 2 PLA input configuration: staircase-function generator

fixed overflow driven by a primary input x . The fixed overflow has one well with a programmable capacity given by the value of the lower-case letter in the well and another well which collects the overflow. At a clock pulse, the contents of this latter well are applied to a metal summer, which corresponds to one column of the PLA. Fig. 2a shows two input configurations. In the top configuration, x is applied as the input to the fixed overflow. In the one below, the complement of y , $3 - y$, is applied. Implementation of the complement is straightforward. It is obtained by interchanging the connections to electrodes on the CCD substrate of the device which realises the circuit shown above it (p. 30, Reference 1). Fig. 2b shows that the function realised by each input

configuration is a staircase function in which the maximum logic value depends on the capacity of the programmable well. In specifying the PLA function, the user chooses whether the input is complemented or not. This determines whether a staircase-down or staircase-up function, respectively, is produced.

Also shown in Fig. 2a is a constant generator in which the output is applied to the metal summer. Thus, the output of the metal summer is the sum of staircase functions and possibly a constant. The value of the constant is determined by the user, and there is exactly one constant generator for each PLA column.

3.2.2 Step-function generator: Fig. 3a shows a more complex PLA input configuration. As in the previous

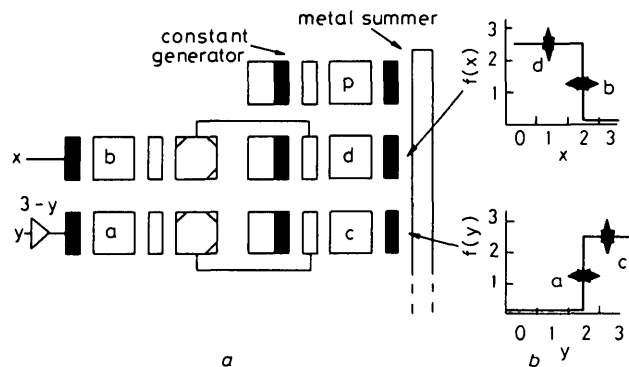


Fig. 3 PLA input configuration: step-function generator

configuration, the input or its complement drives a fixed-overflow circuit. However, in this case, the output well drives the sense input of an inhibit circuit which, in turn, produces the input to the metal summer. For example, in the input configuration driven by x , if $x \leq b$, all of the input charge resides in the well labelled b and none flows to the overflow well. Since there is no charge there, the flow of charge from the source well of the inhibit is not blocked, and the well labelled d fills to its capacity d . The metal summer, in this case, receives a logic d . Fig. 3b shows that the functions realised are step functions in which the height and transition point are determined by well capacities d and b , respectively. If the input is applied uncomplemented, a step-down function is generated, while a complemented input produces a step-up function.

Also shown in Fig. 3a is a constant generator, which is applied to the metal summer. Thus, the metal-summer output is the sum of step functions and a constant. As in the case of the staircase-function generator, the value of the constant is programmable and there is one constant generator per column.

3.2.3 Literal generator: The literal function in 4-valued logic [8] is the unary function

$$a_k x_k^{b_k} = 3 \quad \text{if } a_k \leq x_k \leq b_k \\ = 0 \quad \text{otherwise}$$

Any multiple-valued logic function $f(x_1, x_2, \dots, x_n)$ can be realised as a sum-of-products:

$$f(x_1, x_2, \dots, x_n) = \sum_{u=1}^q p_u \cdot a_1 x_1^{b_1} \cdot a_2 x_2^{b_2} \cdot \dots \cdot a_n x_n^{b_n} \quad (1)$$

where $a \cdot b = \min(a, b)$, $\sum$ is a truncated (to 3) arithmetic sum, $p_u \in \{1, 2, 3\}$, and q is the number of product terms [2].

The realisation of the literal function can be achieved using two copies of the staircase-function generator, one with a direct input and the other with a complemented input. The pair is called a literal generator, and is shown in Fig. 4. The output function is the sum of an increasing and a decreasing staircase function. If the two nonzero parts of the function do not overlap, the resulting sum is 0 for a range of values of x . As it happens, this range corresponds to the range of values of x_k , $a_k \leq x_k \leq b_k$, where the literal function is 3. A literal generator is used with an inhibit circuit at the column output to produce the product term in the summation of eqn. 1. The process by which this is accomplished will be explained later.

3.2.4 Costs of input configurations: The basis on which we compare the various PLAs is the relative cost

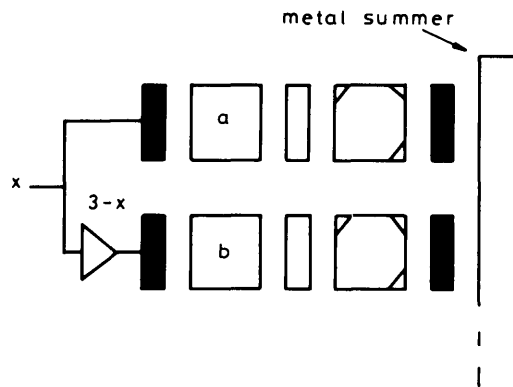


Fig. 4 PLA input configuration: literal generator

factor described in Section 2. The cost for each input configuration is the sum of the costs of the component operators. As mentioned previously, the cost is primarily a measure of the chip area occupied.

The staircase-function generator consists of a multi-threshold operator with a cost of 4. Since complementation is obtained so inexpensively, the staircase-function generator is viewed as having the same cost, regardless of whether or not the complement is present. The step-function generator consists of one fixed overflow and one inhibit and has a total cost of 22. The literal generator consists of two copies of the staircase-function generator and has a cost of 8.

3.3 Output configurations

3.3.1 Complement: Fig. 5a shows the complement circuit at the output of the PLA column. This circuit realises $3 - y$, where y is the input and $-$ is ordinary subtraction. The complement can be realised with a fixed overflow, inhibits, constant generators, and an adder [1]. The dotted line in Fig. 5a shows that the metal summer from the input configurations can be used to directly drive the PLA metal summer. Thus, the user can choose the sum of input functions either directly or after it has been applied to a complement. Also, Fig. 5a shows a constant generator, which can supply a constant to be summed at the PLA output. This is p , the capacity of the constant-generator output well, a parameter which can be specified by the user.

3.3.2 Inhibit: In Fig. 5b, the inhibit circuit is used at the column output. The output of the inhibit is either 0 or f , where f is the capacity of the inhibit output well. 0 is produced when the input from the summer is any nonzero value, and f is produced when the input is 0. f is

determined by the PLA user. As with the complement, the user has a choice, as indicated by the dotted arrow, of connecting the PLA column directly to the PLA-output metal summer or to the sense input of the inhibit.

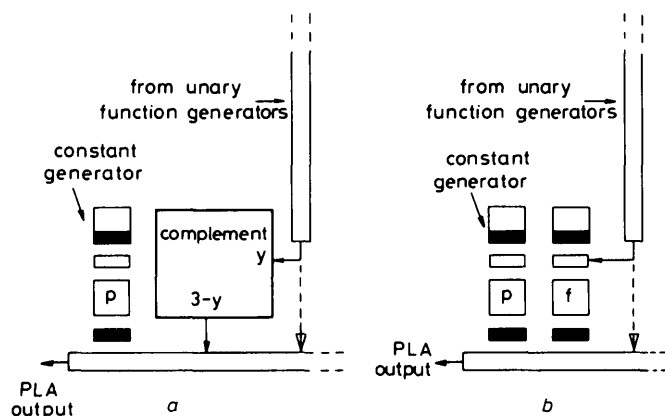


Fig. 5 Two PLA output configurations

a Complement
b Inhibit

3.4 Unary 4-valued functions realised at the output of one PLA column

We now consider the merits of the above configurations by analysing the functions produced by each. As a basis of comparison, we consider the set of all unary 4-valued functions and determine the minimum number of input configurations needed to realise each. Our criteria for comparison is the total number of configurations needed for all unary functions. In all but one of the PLA configurations presented, we assume that the user can apply any input to as many input configurations as needed. Thus, unlike conventional PLAs, an input variable may occur in more than one row. The one exception is the literal generator, where we allow only one occurrence of any variable, a restriction which has applied in past studies.

In the analysis which follows, it will be convenient to represent a unary 4-valued function $f(x)$ as a four-tuple $\langle a_0 a_1 a_2 a_3 \rangle$, where $a_i = f(i)$, for $0 \leq i \leq 3$.

3.4.1 Staircase-function generator with a complement or direct connection to the column output (i): A unary function $\langle a_0 a_1 a_2 a_3 \rangle$ is constant if $a_0 = a_1 = a_2 = a_3$. A unary function $\langle a_0 a_1 a_2 a_3 \rangle$ is increasing (decreasing) if $a_{i-1} \leq a_i$ ($a_{i-1} \geq a_i$) for $1 \leq i \leq 3$. A unary function $\langle a_0 a_1 a_2 a_3 \rangle$ is a V-function if there exists an integer m such that $a_{i-1} \geq a_i$ for all $1 \leq i \leq m$ and $a_j \leq a_{j+1}$ for all $m \leq j \leq 2$, where $1 \leq m \leq 2$. For example, $\langle 2201 \rangle$ is a V-function, and $\langle 1111 \rangle$ is a constant, increasing, decreasing, and V-function.

There are six staircase functions, $\langle 0123 \rangle$, $\langle 0012 \rangle$, $\langle 0001 \rangle$, $\langle 3210 \rangle$, $\langle 2100 \rangle$, and $\langle 1000 \rangle$. The sum of two or more staircase functions or constant functions, is a V-function. The sum of two staircase functions can be a V-function which is neither increasing nor decreasing, e.g. $\langle 2101 \rangle = \langle 2100 \rangle + \langle 0001 \rangle$. A complete characterisation of the functions realised is given in Reference 7. For completeness, we describe this briefly below. Fig. 6 below shows the whole or partial V-functions obtained by summing whole or partial staircase or constant functions. The three triangles along the top represent complete increasing functions. For example, $\langle 0233 \rangle$ is the sum of $\langle 0123 \rangle$ and $\langle 0123 \rangle$. The second and third row represent partial sums of increasing functions where the values associated with $x = 0$ and $0 \leq x \leq 1$, respectively, are missing. The integer along the side represents the number

of staircase functions used to obtain the representations listed just below the triangle. Note that this value is the maximum of the difference between any two adjacent

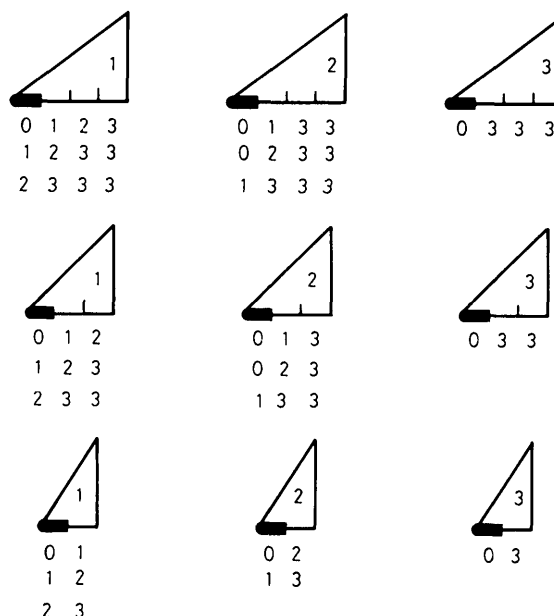


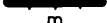
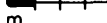
Fig. 6 Whole or partial sums of staircase and constant functions

logic values. This observation will be useful later in counting the number of staircase functions used in specific realisations. By symmetry, there is a set of partial and whole decreasing functions which have the same representations as in Fig. 6 except for the rotation about the vertical axis. Any function which is the sum of staircase or constant functions is a combination of the functions in Fig. 6 or their reverses.

Table 1 shows all possible combinations of staircase and constant functions plus complements of such combinations. There are five columns corresponding to the values of x for which $f(x)$ is minimum. Shown in the heading of the column is the form of the realised functions using representations appearing in Fig. 6. The second column from the left shows the constant functions, the next three show increasing and decreasing functions which are not constant functions, and the next two columns show V-functions which are neither increasing nor decreasing. The entries show how many functions are realised as the sum of s staircase functions and possibly one constant function, with the value of s shown in the extreme left-hand column. For example, $\langle 1012 \rangle$, which is the sum of two staircase functions is counted in the entry '3' of the column corresponding to a function-value minimum located at $m = 1$ (4th column from the right) and in the row corresponding to $s = 2$. The other two functions counted are $\langle 2123 \rangle = \langle 1111 \rangle + \langle 1000 \rangle + \langle 0012 \rangle$ and $\langle 3233 \rangle = \langle 2222 \rangle + \langle 1000 \rangle + \langle 0012 \rangle$. The '(3)' beside the '3' corresponds to the 3 functions which are the reverse of the functions counted in the '3', i.e. $\langle 2101 \rangle$, $\langle 3212 \rangle$, and $\langle 3323 \rangle$. For each column, the function-value minima are listed at the heading, with parenthesised function values corresponding to parenthesised entries in the table.

The second column from the right in Table 1 shows the total number of functions realised as the sum of s staircase functions and possibly one constant function. There are 4 functions for which $s = 0$, the constant functions. There are 18 functions which are the sum of staircase functions exclusively. These are either increasing or decreasing functions. At the other extreme, there are

Table 1: Functions realised as the sum of staircase functions and functions which are the complement of such sums (0 and 1 Column PLAs)

s	$v=u=0$  $0 \leq m \leq 3$	$v=0$  $m=0$ ($m=3$)	$v=0$  $0 \leq m \leq 1$ ($2 \leq m \leq 3$)	$v=0$  $0 \leq m \leq 2$ ($1 \leq m \leq 3$)	v  $m=1$ ($m=2$)	v  $1 \leq m \leq 2$	$P(s)$	$Q(s)$
0	4	0	0	0	0	0	4	[0]
1	0	3(3)	3(3) [3(3)]	3(3) [2(2)]	0	0	18	[10]
2	0	3(3)	3(3) [1(1)]	2(2) [1(1)]	3(3) [5(5)]	3 [3]	25	[17]
3	0	1(1)	1(1)	1(1)	5(5) [6(6)]	4 [4]	20	[16]
4	0	0	0	0	5(5) [6(6)]	4 [4]	14	[16]
5	0	0	0	0	3(3) [3(3)]	2 [2]	8	[8]
6	0	0	0	0	1(1) [1(1)]	1 [1]	3	[3]
Number of functions realised without complement							92	
Number of extra functions realised with complement								[70]

m = value of x for which $f(x)$ is minimum

s = number of staircase functions used in the realisations

$P(s)$ = number of functions requiring s staircase functions (without complement)

$Q(s)$ = number of functions requiring s staircase functions (with complement)

Underlined entries contain functions requiring a nonzero constant applied to the output metal summer in its minimal realisation

three functions which are the sum of 6 staircase functions. These are $\langle 3033 \rangle$, $\langle 3003 \rangle$, and $\langle 3303 \rangle$. Summing more than 6 staircase functions yields no additional functions.

It should be noted that only minimal realisations are counted in Table 1. For example, while $\langle 3213 \rangle$ can be realised as the sum of 4 staircase functions, $\langle 3210 \rangle$, $\langle 0001 \rangle$, $\langle 0001 \rangle$, and $\langle 0001 \rangle$, a minimal realisation requires only 3 staircase functions and a constant, $\langle 2100 \rangle$, $\langle 0001 \rangle$, $\langle 0001 \rangle$, and $\langle 1111 \rangle$. Thus, $\langle 3213 \rangle$ is counted among the 5 functions in the entry corresponding to the column $m = 2$ and the row $s = 3$.

An examination of Fig. 6 and Table 1 shows that not all V -functions are realised as the sum of staircase functions. For example, $\langle 0223 \rangle$ is not included. The first two logic values, 02, can only be realised as the sum of two increasing functions $\langle 0123 \rangle$ and $\langle 0123 \rangle$. However, the corresponding function would be $\langle 0233 \rangle$ not $\langle 0223 \rangle$. In fact, any V -function with 022 as a subpattern is not realisable. There are 5 other nonrealisable subpatterns.

Observation: Let $f(x) = \langle a_0 a_1 a_2 a_3 \rangle$ be a unary 4-valued function realised as the sum of staircase functions and possibly a constant function, where $f(x)$ is in the set of V -functions not containing forbidden subpatterns 011, 110, 022, 220, 122, and 221. The minimum number of staircase functions used to realise $f(x)$ is

$$\max(a_{i-1} - a_i) + \max(a_{j+1} - a_j) \quad 1 \leq i \leq m \quad m \leq j \leq 2$$

where m is a value of x for which $f(x)$ is minimum.

The minimum number of staircase functions required to form a given realisable V -function is the sum of the maximum difference between adjacent logic values along

the decreasing part of the function and along the increasing part. This follows from an earlier observation on partial staircase functions.

If we allow complementation at the output of a metal summer, then additional functions can be realised. For example, any V -function that is neither increasing nor decreasing, produces an inverted V -function which is neither increasing nor decreasing when complemented. Such a function is not the sum of staircase functions. In Table 1 the number of functions realised using the complement is indicated by brackets $[\]$. Thus, in the two columns corresponding to V -functions which are neither increasing nor decreasing, some entries of the form $i(i)$ or i are repeated in brackets, $[i(i)]$ or $[i]$, respectively.

Complements of increasing (decreasing) functions are decreasing (increasing) functions. The only additional functions generated by complementing such functions are those which contain a forbidden subpattern, 011, 110, 022, 220, 122, and 221. Table 1 shows there are, in all, 10 such functions. For example, $\langle 0111 \rangle$ contains a forbidden subpattern, and thus is not the sum of staircase functions and possibly a constant. However, $\langle 0111 \rangle$ is the complement of $\langle 3222 \rangle$, which is a decreasing function realisable as the sum of a single staircase function $\langle 1000 \rangle$ and the constant $\langle 2222 \rangle$. Thus, $\langle 0111 \rangle$ is an additional function only realisable because of the complement.

Table 1 shows that the total number of additional functions obtained by using the complement is 70. This is in addition to the 92 functions which are realised as the sum of staircase functions and perhaps a constant function. The complement is necessary if all functions are to be realised in a PLA using a staircase-function generator. For example, $\langle 0100 \rangle$ is not realised as the sum of any

staircase functions, but it is realised as the complement of such a sum ($\langle 1000 \rangle + \langle 0012 \rangle + \langle 2222 \rangle$). Thus, the function $\langle 0100 \rangle$ itself is not realisable unless the complement is available. Any function on n variables is realisable provided that there are enough columns. This follows from the fact that any product of a constant and literal functions (a term in the summation of eqn. 1) can be realised by an appropriate choice of staircase functions, at most 6 for each variable. For example, $^1x_k^2$ is realised by summing the six staircase functions $\langle 1000 \rangle$, $\langle 1000 \rangle$, $\langle 0001 \rangle$, $\langle 0001 \rangle$, and $\langle 0001 \rangle$ each with input x_k . p_u of eqn. 1 is realised by choosing a constant to the metal summer equal to $3 - p_u$.

The underlined entries in Table 1 correspond to functions which require a nonzero value for the output of the constant generator at the PLA output metal summer. For example, one function counted in the entry $[3, (3)]$ of the fourth column ($0 \leq m \leq 1$, ($2 \leq m \leq 3$)) and the third row ($s = 1$) is $\langle 1122 \rangle$, with the realisation, $\langle 1122 \rangle = \langle 2100 \rangle + \langle 2222 \rangle + \langle 1111 \rangle$. In all, there are 10 such functions. The other 9 are

$$\begin{aligned}\langle 2211 \rangle &= \langle 0012 \rangle + \langle 2222 \rangle + \langle 1111 \rangle \\ \langle 1121 \rangle &= \langle 0001 \rangle + \langle 2100 \rangle + \langle 2222 \rangle + \langle 1111 \rangle \\ \langle 1211 \rangle &= \langle 1000 \rangle + \langle 0012 \rangle + \langle 2222 \rangle + \langle 1111 \rangle \\ \langle 2232 \rangle &= \langle 0001 \rangle + \langle 2100 \rangle + \langle 2222 \rangle + \langle 2222 \rangle \\ \langle 2322 \rangle &= \langle 1000 \rangle + \langle 0012 \rangle + \langle 2222 \rangle + \langle 2222 \rangle \\ \langle 1132 \rangle &= \langle 2100 \rangle + \langle 2100 \rangle + \langle 0001 \rangle + \langle 1111 \rangle + \langle 1111 \rangle \\ \langle 2311 \rangle &= \langle 0012 \rangle + \langle 0012 \rangle + \langle 1000 \rangle + \langle 1111 \rangle + \langle 1111 \rangle \\ \langle 1131 \rangle &= \langle 2100 \rangle + \langle 2100 \rangle + \langle 0001 \rangle + \langle 0001 \rangle + \langle 1111 \rangle + \langle 1111 \rangle \\ \langle 1311 \rangle &= \langle 0012 \rangle + \langle 0012 \rangle + \langle 1000 \rangle + \langle 1000 \rangle + \langle 1111 \rangle + \langle 1111 \rangle\end{aligned}$$

The constant is necessary in the realisation of these functions. Without it, they would be unrealisable in a one-column PLA.

3.4.2 Staircase-function generator with an inhibit or direct connection to the column output (ii): If an inhibit is used instead of the complement at the PLA column output, a different class of functions is realised. The additional functions, shown in Table 2, produce two

Table 2: Functions realised as the sum of staircase functions applied to an inhibit

s	Function	Condition	U(s)	V(s)	
1	c A A A	A = 3 0 ≤ c ≤ 1	12		
	c c A A				
	c c c A				
	A A A c				
	A A c c				
	A c c c				

	0 0 0 A	A = 2	2		
	A 0 0 0				
	c A A A	1 ≤ A ≤ 2 0 ≤ c < A	12		
	c c A A				
	A A A c				
A A c c					
2	c A c c	1 ≤ A ≤ 3 0 ≤ c < A	18		
	c A A c				
	c c A c				
Number of functions improved with inhibit			14		
Number of extra functions realised with inhibit				30	

s = number of staircase functions used in minimal realisation
 $U(s)$ = number of functions which are realised as a sum of staircase functions without inhibit, but which are also realised with fewer staircase functions using an inhibit
 $V(s)$ = number of functions requiring s staircase functions (with inhibit)

output logic levels only. The entries correspond to the $\langle a_0 a_1 a_2 a_3 \rangle$ notation for unary functions. For example, the first row of the column headed by 'Function', $c A A A$, corresponds to the unary function $\langle a_0 a_1 a_2 a_3 \rangle = \langle c A A A \rangle$. The third column specifies the range of values for c and A . For this function $A = 3$ and $0 \leq c \leq 1$. Thus, $c A A A$ represents the two functions $\langle 0333 \rangle$ and $\langle 1333 \rangle$. In addition to the 92 functions which are sums of staircase functions, the inhibit produces 30 additional functions, less than one half the number of additional functions produced by the complement. However, unlike the complement, the inhibit improves on the functions which are the sum of staircase functions. For example, $\langle 0333 \rangle$ can be realised as the sum of three staircase functions ($\langle 0333 \rangle = \langle 0123 \rangle + \langle 0123 \rangle + \langle 0123 \rangle$), but requires only one when the inhibit is used ($\langle 0333 \rangle = \text{inhibit}(\langle 1000 \rangle)$). There are altogether 14 functions which have a better realisation when the inhibit is used.

3.4.3 Step-function generator with a complement or direct connection to the column output (iii): Table 3

shows the functions realised by summing step functions and constants. The function-entry notation is identical to that of Table 2. Compared to the 18 realised by the sum of one staircase function and possibly a constant function, there are 36 functions realised as the sum of one step function and possibly a constant function.

It is interesting to note that the sum of more than 3 step functions and possibly a constant produces no more functions than are realised by summing 3 or fewer functions. Recall that for staircase functions, up to 6 may be summed to produce a distinct function. In all, 156 functions are realised as the sum of step functions, approximately 50% more than the 92 realised as the sum of staircase functions. However, the input configurations required to realise the step function is more complex than for the staircase function.

Also shown in Table 3 are functions which are complements of the sums of step functions and possibly a constant function. As with the noncomplemented functions, the form of each realisation is given. Brackets [] are used to enclose the number of functions realised when a complement is used. In all, 62 additional functions are produced. Thus, the total number of functions realised by the complement operation at the column output is 218 using step functions or 162 using staircase functions at the input.

As with the staircase functions, the complement is necessary in order to realise all functions. For example, neither $\langle 0010 \rangle$ nor $\langle 0100 \rangle$ is realised as a sum of step functions, but both are the complement of the sum of 2 step functions and a constant (e.g. $\langle 0010 \rangle = \langle 3300 \rangle + \langle 0003 \rangle + \langle 2222 \rangle$). Given enough PLA columns, any 4-valued function can be realised.

3.4.4 Step-function generator with an inhibit or direct connection to the column output (iv): Table 3 also shows

Table 3: Functions realised as the sum of step functions and functions which are the complement or inhibit of such sums (0 and 1 Column PLAs)

s	Function				Condition	R(s)	S(s)	T(s)	R(s)	S(s)	T(s)
0	A	A	A	A	$0 \leq A \leq 3$	4			4	0	0
1	c	c	c	A	$0 \leq c < A \leq 3$	36			36	{0}	{0}
	c	c	A	A							
	c	A	A	A							
	A	c	c	c							
	A	A	c	c							
	A	A	A	c							
2	c	A	A + B	A + B	$0 \leq c < A \leq 2$ $1 \leq B \leq 3 - A$	24					
	c	A	A	A + B							
	c	c	A	A + B							
	A + B	A + B	A	c							
	A + B	A + B	A	c							
	A + B	A	c	c							
	A	c	B	B	$0 \leq c < A, B \leq 3$	42			78	{30}	{12}
	A	c	c	B							
	A	A	c	B							
	$\bar{A}$	$\bar{c}$	$\bar{B}$	$\bar{B}$							
$\bar{A}$	$\bar{c}$	$\bar{c}$	$\bar{B}$								
$\bar{A}$	$\bar{A}$	$\bar{c}$	$\bar{B}$	same, except $c = 0: A + B \leq 3$ $c = 1: A - B = c$							
c	A	c	c	$c = 0: 1 \leq A \leq 3$ $c = 1: A = 3$					{12}		
c	A	A	c								
c	c	A	c								
A	A + B	B	B	$1 \leq A, B \leq 2$	12						
A	A + B	A + B	B								
A	A	A + B	B								
0	A	A + B	A + B	A + B + C	$A = B = C = 1$	2					
A + B + C	A + B	A	0								
C	c	A	A + B								
A + B	A	c	C	$0 \leq c < \begin{cases} A \leq 2 \\ C \leq 3 \end{cases}$	22						
$\bar{C}$	$\bar{c}$	$\bar{A}$	$\bar{A} + \bar{B}$								
$\bar{A} + \bar{B}$	A	$\bar{c}$	$\bar{C}$								
$\bar{A} + \bar{B}$	$\bar{B}$	$\bar{B} + \bar{C}$	$\bar{C}$	same, except $c = 0, A = B = C = 1$					[20]		
$\bar{C}$	$\bar{B} + \bar{C}$	$\bar{B}$	$\bar{A} + \bar{B}$								
$\bar{A} + \bar{B} + c$	$\bar{B} + c$	$\bar{B} + \bar{C} + c$	$\bar{C} + c$								
$\bar{C} + c$	$\bar{B} + \bar{C} + c$	$\bar{B} + c$	$\bar{A} + \bar{B} + c$	$A = C = 2, B = 1$ $c = 1$							
A + B	A + B + C	A + C	C	$A = B = C = 1$	2						
C	A + C	A + B + C	A + B								
Number of functions realised without complement or inhibit						156			156		
Number of extra functions realised with complement							[62]			[62]	
Number of extra functions realised with inhibit								{12}			{12}

s = number of step functions used in the realisations
R(s) = number of functions requiring s step functions
S(s) = number of functions requiring s step functions (with complement)
T(s) = number of functions requiring s step functions (with inhibit)

the additional functions realised when an inhibit is used at the output of the PLA column. The number of additional functions is 12, as shown in braces { }. Thus, the inhibit produces about one fifth of the additional functions produced by the complement. Again, any 4-valued function can be realised using this combination of input and output configurations.

3.4.5 Literal generator with an inhibit at the column output and no constant generator (v): A literal generator consists of two staircase-function generators, one with x_k on the input well and the other with $3 - x_k$. The function produced at the PLA column is the arithmetic sum of a staircase-up and a staircase-down function. If the two functions are 0 for a range of x_k , $a_k \leq x_k \leq b_k$, the sum is

0 for that range. Thus, if n copies of the literal generator are applied to the summer column, each with x_k applied at the input for $1 \leq k \leq n$, the column has 0 charge if and only if each x_k is bounded as $a_k \leq x_k \leq b_k$.

With an inhibit at the column output, the inhibit output will be f , the logic capacity of the inhibit output well, if and only if $a_k \leq x_k \leq b_k$ for all $1 \leq k \leq n$. The output function is

$$f \cdot a_1 x_1^{b_1} \cdot a_2 x_2^{b_2} \dots a_n x_n^{b_n}$$

which is one product term in the sum-of-products expression of eqn. 1. The metal summer to which the inhibit output is applied, realises the sum of eqn. 1. We have chosen not to use the constant generator at the PLA output, since the product of literals formulation for a

Table 4: Number of functions realised using the fewest input configurations against the number of columns and number of input configurations.

Number of columns	Number of input configurations	Output configuration				
		complement or direct		inhibit or direct		inhibit only
		Input configuration				
		staircase function	step function	staircase function	step function	literal without constants
0	0	4	4	4	4	0
1	1	28	36	44	36	31
	2	42	108	35	90	0
	3	36	70	4	38	0
	4	30	0	0	0	0
	5	16	0	0	0	0
	6	6	0	0	0	0
2	1	0	0	0	0	0
	2	8	0	61	0	130
	3	20	38	72	50	0
	4	16	0	30	36	0
	5	18	0	0	0	0
	6	10	0	0	0	0
	7	8	0	0	0	0
	8	4	0	0	0	0
	9	2	0	0	0	0
3	1	0	0	0	0	0
	2	0	0	0	0	0
	3	0	0	4	0	91
	4	2	0	0	0	0
	5	2	0	2	2	0
	6	2	0	0	0	0
	7	2	0	0	0	0
4	1	0	0	0	0	0
	2	0	0	0	0	0
	3	0	0	0	0	0
	4	0	0	0	0	4
Total number of functions		256	256	256	256	256
Total number of input configurations		896	576	606	634	580
Total cost of input configurations		3584	12672	2424	13948	4640

multiple-valued function without a constant given in eqn. 1 is common, and thus serves as a basis for comparison with previous formulations. For the same reason, no constant generator is part of the column driven by input configurations.

Since each column output is summed on the metal summer which serves as the PLA output, the PLA realises a sum-of-products involving literals with Sum being normal addition truncated to 3 when the sum exceeds 3. As long as there are sufficiently many columns (product terms), any n -variable 4-valued functions can be realised [2]. The number of functions realised at the output of one column is 31, since there are 10 literal functions for each of 3 nonzero logic levels and the constant function <0000>.

The advantage of the literal generator over the staircase-function generator with the inhibit at the column output is in the additional logic available to the user of the PLA. With the literal generator, both halves of the pair of staircase-function generators are included whether both are needed or not. With the staircase-function generator, however, only one half of the pair will be used when only one half is needed. Thus, the latter PLA makes more efficient use of chip area. This will be discussed in Section 4.

3.5 Functions realised by PLAs with more than one column

In this section, we analyse the 4-valued unary functions realised by PLAs with one, two, or more columns, which

are in one of the five configurations discussed above. A program was written which enumerates the unary functions according to the number of input configurations and columns required. Starting with functions realised by a single column, all traditional unary functions realised with 2 columns were generated, then 3 columns, etc. At each step, the realisation requiring the least number of input configurations is retained. The results are shown in Table 4. The right five columns represent the number of unary functions realized by the following PLA configurations:

- staircase-function generator/complement or direct
- step-function generator/complement or direct
- staircase-function generator/inhibit or direct
- step-function generator/inhibit or direct
- literal generator/inhibit always

The top two sections of rows correspond to PLAs with 0 or 1 column and the data here agrees, as it should, with Tables 1, 2 and 3. It is interesting that the 4 constant functions <0000>, <1111>, <2222>, and <3333> are counted in the functions which require one column, since with the literal generator, there is no constant generator at the PLA output.

It should be noted that the data represent a minimisation of the total number of input units, not columns. That is, the 8 functions using 3 columns in the minimal realisations associated with staircase-function generator/complement or direct can all be realised in such PLAs using only 2 columns. However, with 3 columns, there is

a realisation which requires fewer input units overall than with any 2-column realisation. With this one exception, there are no functions for which there is an advantage to using more columns than the minimum, in any of the three configurations.

Observation: The number of columns necessary to realise any unary 4-valued function in a PLA using either the staircase- or step-function generator and possibly a complement at the column output is at most 2, while in a PLA using the literal generator and an inhibit at the column output is at most 4.

4 Concluding remarks

The next to last row in Table 4 shows the total number of input configurations needed to realise all 256 4-valued unary functions. It shows that staircase-function generator/complement or direct requires significantly more input configurations, 896, than any of the other configurations, each requiring about 600 configurations each. However, considering a cost which reflects chip area occupied, the staircase-function generator/complement or direct is very good, ranking 2nd out of 5.

The main results of this paper are summarised in the last row of Table 4. It shows the total cost (chip area) of the input configurations required to realise all unary functions. Each entry is derived by multiplying the number of required input configurations in the row just above by the cost of each configuration. In this comparison, the staircase-function generator/inhibit or direct requires the least chip area, with staircase-function generator/complement or direct second best, consuming about 50% more chip area. The two combinations using the step-function generator are extremely costly by comparison. As discussed earlier, the step-function generator is the same as the staircase-function generator except that an inhibit has been added. From this analysis, it is clear that the extra circuit complexity of the inhibit does not increase the number of realised functions sufficiently to offset the extra chip area.

As discussed earlier, the literal-function generator consists of two copies of the staircase-function generator. We observed that for some literal functions, one of the two

copies is not needed. From Table 4, it can be seen that it uses almost twice the chip area of the staircase-function generator/inhibit or direct, suggesting that there is significant waste in the use of the added staircase-function generator.

The main conclusion we reach is that the staircase-function generator together with the inhibit at the column output is significantly better than the other combinations. Thus, an important problem is the development of an efficient synthesis technique for this PLA design.

5 Acknowledgment

The authors acknowledge valuable comments by three referees of this paper. This work was funded by NATO Grant 423/84.

6 References

- 1 KERKHOFF, H.G.: 'Theory, design, and applications of digital charge-coupled devices'. Ph.D. Thesis, Twente University of Technology, Enschede, The Netherlands, April 1984
- 2 TIRUMALAI, P., and BUTLER, J.T.: 'On the realization of multiple-valued logic functions using CCD PLA's'. Proceedings of the 14th International Symposium on Multiple-Valued Logic, Winnipeg, Manitoba, Canada, May 1984, pp. 33-42
- 3 KUO, H.-L., and FANG, K.-Y.: 'The multiple-valued programmable logic array and its application in modular design'. Proceedings of the 15th International Symposium on Multiple-Valued Logic, Kingston, Ontario, Canada, May 1985, pp. 10-18
- 4 IMME, M., and PAPACHRISTOU, C.A.: 'Simplification of MVL functions and implementation via a VLSI array structure'. *ibid.*, pp. 242-248
- 5 SASAO, T.: 'An algorithm to derive the complement of a binary function with multiple-valued inputs'. *IEEE Trans.*, 1985, C-34, pp. 131-140
- 6 BENDER, E.A., BUTLER, J.T., and KERKHOFF, H.G.: 'Comparing the SUM with the MAX for use in four-valued PLA's'. Proceedings of the 15th International Symposium on Multiple-Valued Logic, Kingston, Ontario, Canada, May 1985, pp. 30-35
- 7 KERKHOFF, H.G., and BUTLER, J.T.: 'Design of a high-radix programmable logic array using profiled peristaltic charge-coupled devices'. Proceedings of the 16th International Symposium on Multiple-Valued Logic, Blacksburg, Virginia, USA, May 1986, pp. 128-136
- 8 ALLEN, C.M., and GIVONE, D.D.: 'A minimization technique for multiple-valued logic systems', *IEEE Trans.*, 1968, C-17, pp. 182-184