J. C. Logue
N. F. Brickman
F. Howley
J. W. Jones
W. W. Wu

Hardware Implementation of a Small System in Programmable Logic Arrays

Abstract: Large Scale Integration, LSI, is the means by which digital circuits have achieved remarkable manufacturing cost reductions but, unfortunately, at the expense of higher engineering design costs. Programmable Logic Arrays, PLAs, exploit many of the benefits of LSI but without the high engineering design costs. This paper describes an experiment in the design and implementation of a small complex system in array logic. The IBM 7441 Buffered Terminal Control Unit was selected for this comparison because it is a small but complex terminal controller implemented in dual in-line packaged transistor logic, DIP-TL, with small to medium scale integration.

Introduction

The mixed blessings of Large Scale Integration (LSI) at the system level have been discussed by many authors on many occasions [1]. On the one hand, LSI has been successfully exploited in memory applications; on the other hand, it is recognized that in the logic area LSI poses a serious dilemma—it provides the potential for a significant reduction in product cost but with the exposure of high development costs. One might conclude, therefore, that only products with high volume can benefit from LSI.

At the current level of LSI there are at least three approached aimed at the solution to the LSI dilemma:

1. Microprogramming
2. Microprocessor on a chip
3. Programmable logic array (PLA).

Microprogramming, because it utilizes a control memory which is an array structure, fulfills the requirement for successfully exploiting LSI, but it is limited in its system applicability to the control area.

The microprocessor on a chip is essentially a system module that achieves its high volume by exploiting a high volume market area. However, it transfers engineering complexity to the user in the form of software that must be written as well as adapters that must be designed and built to interface the microprocessor with its peripheral hardware. Such adapters are, in many cases, low-volume products. In addition, the microprocessor must, in general, be fabricated out of a higher performance technology than if the function it performed were provided in a hard-wired version.

A PLA can be thought of as combining microprogram control with random logic in the same array. Thus, the LSI benefits stemming from array structures can be realized in the PLA for both logic and control. However, rather than using an address decoder, as in a control store, the PLA does an associative match on the input command word. This operation will be explained in more detail in the "Review of PLA basics" section.

The above discussion should not be interpreted by the reader as an attack on microprocessors. Rather, it is intended to show that a systems designer must consider many possibilities before he can achieve an optimum solution. The purpose of this paper is to acquaint the systems designer with the most complete solution available that solves the LSI dilemma, namely, the Programmable Logic Array. The design and implementation experience substantiating this viewpoint will be presented by means of a design example that culminated in a working system. The experimental system that has been implemented in array logic is functionally identical to the IBM 7441 Buffered Terminal Control Unit (TCU) [2].

Review of PLA basics

The form of array logic with which most readers may be familiar is the microprogrammed array commonly used for control. PLA shares many of the characteristics of the microprogrammed array: simple array structure, standard "master chip" fabrication, late personalization, etc. PLA, however, is different from the microprogrammed array in two critical ways: 1) it uses an associative technique for addressing (content addressable), and 2) the array itself is not limited to control logic but integrates both data path and control logic in the same array (see section on control).

The review that follows consists of three subtopics: the overall structure of the PLA, the macro concept and macro usage.

• *Overall structure*

The PLA is a read-only structure that is programmed to perform both sequential and combinational logic [3]. The combinational logic is implemented by means of sum-of-product functions as performed by AND arrays and OR arrays in cascade. The sequential logic is achieved by means of storage elements in the form of flipflops (registers) that may be driven in either a set/reset or toggle mode as a function of the pattern stored in a portion of the OR array (Fig. 1). The feedback register is connected to internal inputs to the AND array through a feedback path that is internal to the PLA chip.

The PLA has basically a table look-up structure, where the AND array forms the look-up library and the OR array forms the resultant output for the operation. The AND array consists of many product terms (words) and is partitioned into two sections: the external field and the feedback field from the feedback register. Both of these fields are processed in parallel in the AND array to select words in the OR array. The OR array performs the logical OR operation on the values written in the selected words. The OR array is also partitioned into two fields: one field is gated to the external outputs, the other operates on the feedback register.

• *The macro concept*

It should be noted that array logic is well suited to the macro, or modular, building-block concept. The term "macro" as used here is a contraction of "macro-function." A count-down counter is an example of a macro; this macro will be explained in more detail later. There is no concern about, or restriction on, the physical location of the macro on the chip. Our TCU design shows that macro identity is maintained and visible within the overall system design, thus enhancing the readability of the pages of PLA personalization. One can establish a library of common functions to be picked up by anyone without any redesign. Finally, the reliability of array logic macros should be enhanced, since once debugged—always debugged.

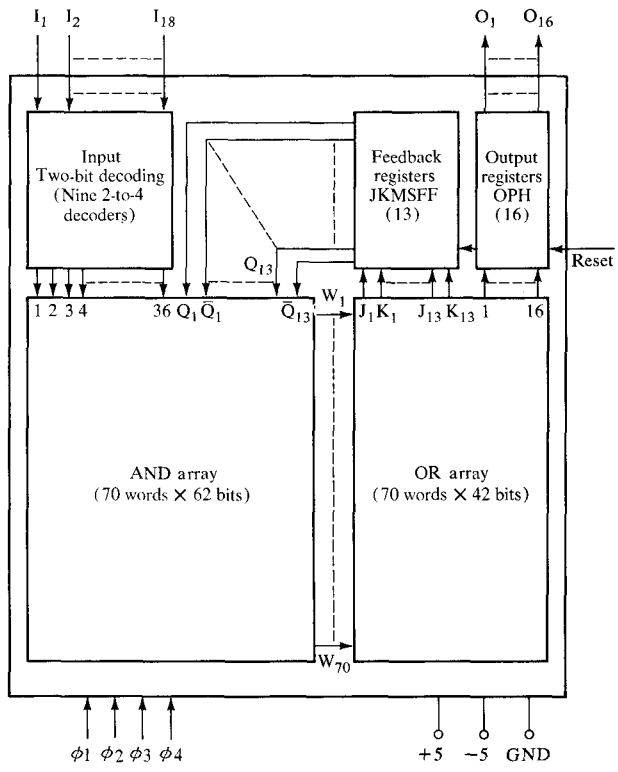


Figure 1 Functional block diagram of PLA including specific data on the chip used in the Terminal Control Unit (TCU).

• *Macro usage*

The internal feedback register provides the facility for sequencing the PLA. The function of this register is ideal for implementing up/down counters or state switching. The PLA can be used as a sequential logic block because it has memory in the feedback path [Fig. 2(a)]. The content of part of the feedback register (state) is used to select macros in the PLA. These macros operate on any combination of input data and feedback information. The state is switched so as to select a series of macros in the PLA; these macros are selected to perform a particular routine. An example of a routine is the selection of the parity-checking macro in the keyboard buffer. If the parity is good, the routine selects the next macro, which in turn transfers the good parity character to the storage buffer system.

A single PLA can select a series of macros by changing the state or feedback control; however, several such routines would be needed in a complex system. The selection of the routines must be based on the initial command to the PLA. An example is shown in Fig. 2(b),

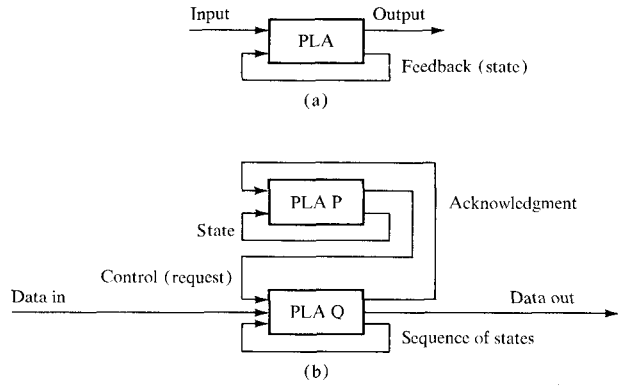


Figure 2 (a) PLA as a sequential logic block. (b) Hierarchial control scheme for PLAs.

where PLA P generates a request for a unique routine in PLA Q. The change of state in PLA Q is based on two fields: the external command and the internal feedback state. Words that are coded in PLA Q select the correct sequence of macros in this PLA. The end of the routine is defined by a state that generates an acknowledgement to PLA P, which in turn causes that PLA to enter its next state.

• *State assignment*

The selection of the external commands between the PLAs and the internal coding of the feedback states is part of the state assignment procedure. This state assignment (the most difficult part in the coding of the arrays) determines the efficiency of the arrays. A macro should only be written once in the PLA. If this macro is selected by many routines, then the designer has to consider the value of the state which selects it. A macro that is shared by many routines has to have DON'T CARE (independent of input) values in its control field [4]. The number of DON'T CARES is at least $\log_2 N$, where N is the number of control states that share this macro.

Another consideration during the state assignment is the state-switching algorithm. The states should be selected so as to minimize the number of terms necessary to change states.

Functional description of the TCU

The TCU [2] is a buffered terminal device that can store either 160 or 480 characters, depending upon customer requirements. The TCU controls the buffer operation such that its contents are either printed or transmitted over a communication line to a central processing unit (CPU). The loading of the buffer is also controlled by the TCU; this is done from either the typewriter keyboard or via the communication line from the CPU.

The following brief description of some of the features available in the TCU is presented to help the reader appreciate the functional complexity that the PLA design had to accommodate.

The CPU addresses the TCU to send it an output message; this action requires the transmission of a sequence of four control characters. The address sequence is: C, followed by D, followed by the address, and then a space character or one of the eleven invalid characters. The TCU has to respond with its status when it recognizes its address; i.e., a Y if its status is such that it can receive and print the message, and an N if not. The CPU polls the TCU when it desires the TCU to transmit its input message. The poll sequence is a C control character followed by the terminal address. The TCU responds with an N control character if it has no message, and with a D control character followed by the message if it is ready to transmit.

Messages are checked by parity on each character, and the TCU has to retransmit the message if the CPU detected an error during transmission. The TCU counts the number of characters entered into its buffer and provides an audible alarm if the number of characters is within ten of the limit. The TCU will lock the keyboard if the number of characters entered is within two of the limit.

The editing of the contents of the buffer is achieved via the backspace function, which erases the last character from the buffer and decrements the character count by one. This operation is initiated by pressing the backspace key on the typewriter keyboard.

The TCU hardware consists of a total of 2,058 logical circuits, four dual 480 shift register modules, and many discrete peripheral circuits. The complexity of its operations made the TCU a suitably challenging test for an experimental implementation of PLA technology.

Implementation of the PLA version of the TCU

Implementation of the TCU in array logic was accomplished in three distinct phases: 1) the design phase, which consisted of mapping the TCU functional specification into the PLA, 2) the simulation phase, which featured the APL programming language, and 3) the build and test phase, which produced the actual working version of the TCU in array logic, and verified that it was functionally identical to its DIP-TL (dual in-line packaged transistor logic) counterpart.

The decoding of pairs of inputs (two-bit decode) provides all of the combinations of ANDS as well as EQUIVALENCES, EXCLUSIVE ORS, and the ORS of the input pairs (see Table 1). The output polarity holds (OPH) are equivalent to D-type flipflops. Outputs Q and $\overline{Q}$ of the JK masterslave flipflops (JKMSFF) are fed back to the AND array inputs. The JKMSFF outputs (controlled

by a clock) are dependent on the conditions of the J and K inputs from the OR array. Table 2 lists the input/output conditions; the hold means that the output does not change, and the toggle indicates that the output is complemented each clock cycle. The symbols listed in Tables 1 and 2 are used in Figs. 4–6.

• *Design process*

The functions performed by the TCU were coded into seven PLA chips. The interface logic between the PLA and the communication line, the keyboard, and printer was implemented in DIP-TL and some special circuits, because the interface circuits required electrical characteristics that were different from those provided by the FET arrays.

The design objective was to include the data flow, logical operations, and the control in a minimum number of PLAs. The system requirements were divided into four subsystems: the communication subsystem, the storage subsystem, the keyboard, and the printer. Each subsystem was mapped into the PLAs. Several design iterations were made in order to obtain the best partitioning scheme. The partitioning of the system into seven PLAs was aided by the macro capability in the PLAs. These macros were coded into the PLAs, and were moved from one PLA to another so as to obtain optimum utilization of feedback registers, input and output pins, and number of words used.

The final partitioning scheme (which satisfied the mapping of the TCU functions into seven PLAs) resulted in all the words being used in all the arrays.

Three levels of control were devised for the TCU. These levels evolved from both the use of macros within the PLA and the sequential logic properties of the PLA. The sequential logic properties formed the first level of control. The second level was necessary to integrate the PLAs into a subsystem; e.g., one PLA being used to receive information from the line at a rate determined by another PLA. The third level was used to link the subsystems together to provide the overall TCU function. This level is demonstrated in the control of the communication and printer priorities in accessing the storage subsystem.

Partitioning

The first problem confronting a designer in LSI is the partitioning of the system into several subsystems that can be packaged efficiently into LSI modules. Array logic structures discipline the designer in deciding on the optimum partitioning because macros defining array personalities are available, and these macros are defined for specific functions. The designer can select the group of macros necessary in each subsystem, and then determine how these macros fit into the arrays. Of the four subsystems partitioned in the TCU, the communication line used three arrays, and each of the other subsystems used one array. A seventh array was used to control the operations in these subsystems.

Table 1 The 16 unique functions of the two-bit decoded inputs.

A *B* Two-bit decoding				*Function*	*Symbol*
AB	$A\bar{B}$	$\bar{A}B$	$\bar{A}\bar{B}$		
0	0	0	0	DON'T CARE	· ·
0	0	0	1	$A + B$	P P
0	0	1	0	$A + \bar{B}$	P N
0	0	1	1	A	1 ·
0	1	0	0	$\bar{A} + B$	N P
0	1	0	1	B	· 1
0	1	1	0	$A = B$	E E
0	1	1	1	$A \cdot B$	1 1
1	0	0	0	$\bar{A} + \bar{B}$	N N
1	0	0	1	$A \oplus B$	U U
1	0	1	0	$\bar{B}$	· 0
1	0	1	1	$A \cdot \bar{B}$	1 0
1	1	0	0	$\bar{A}$	0 ·
1	1	0	1	$\bar{A} \cdot B$	0 1
1	1	1	0	$\bar{A} \cdot \bar{B}$	0 0
1	1	1	1	False	D D

Notes: DON'T CARE – Word line active independent of A and B.
FALSE – Word line inactive independent of A and B (not used)

Control

The PLA has the characteristics of an ideal sequential logic element – the internal feedback maintains the state, while the external input can be used to switch these states. The PLA has solved the traditional problems of undetermined states ("race conditions") associated with sequential logic, the solution being provided by clocked register-register transfer in PLA. The data flow and definitions of some of the macros contained in each subsystem of the TCU are shown in Fig. 3. As stated above, in PLA, data flow and control are not separate as in the more conventional machine design. Figure 3 shows this combined data path and control and the following section describes the respective functions.

Table 2 Input/output conditions of the JKMS flipflops.

Input J	K	*Output*	*Symbol*
0	0	Hold	·
0	1	Reset	R
1	0	Set	S
1	1	Toggle	T

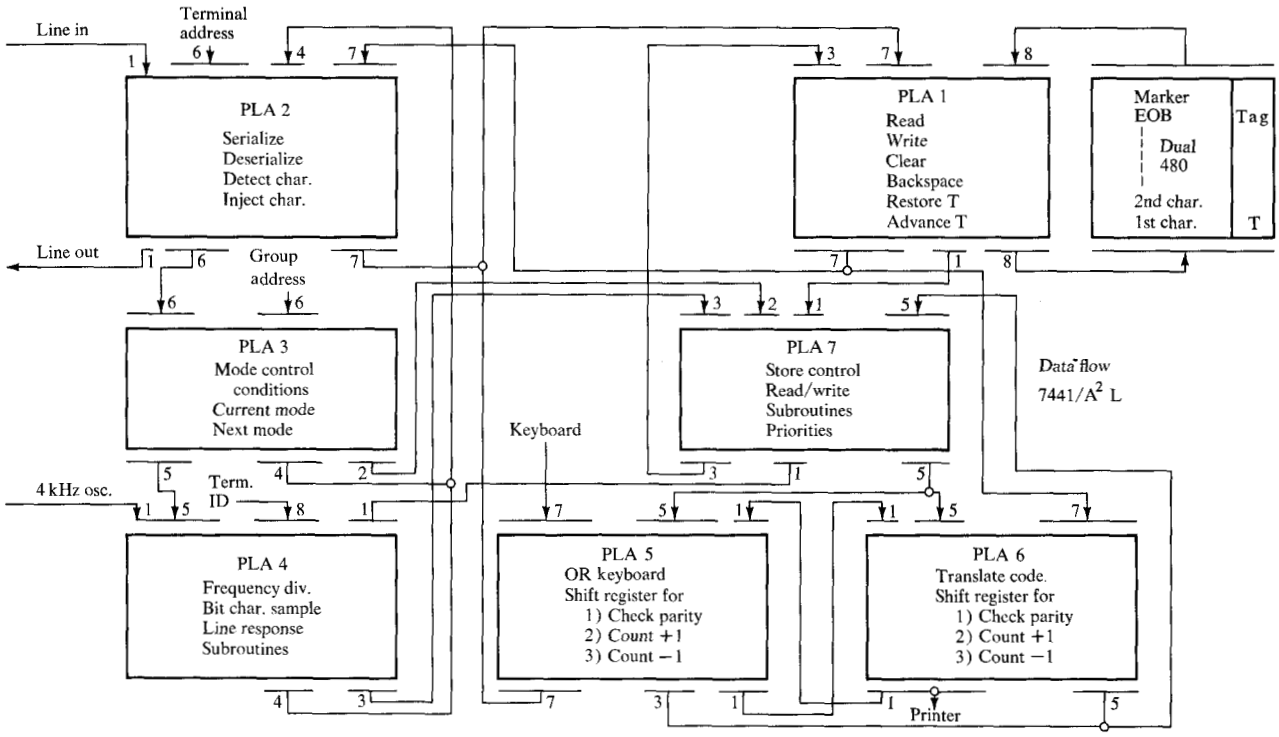


Figure 3 Data flow of PLAs in TCU.

Storage operation and macro examples

The storage device used as a message buffer in the TCU is a dynamic shift register eight bits wide and 480 characters long. The reading, writing, and formatting of messages is described below. The routine used for restoring the shift register following a transmission or printing operation is also described.

The message is formatted into a variable-length contiguous segment in the shift register. The first character of the message is identified by a unique tag bit (T) which is written into a unique position in the eight-bit-wide shift register positions. The last character, End of Block (EOB), is next to a unique marker which is written into the shift register prior to entering the message.

PLA1 interfaces with the shift register and forms the storage subsystem; this array is connected in series with the circulating data in the shift register and runs synchronously with it. PLA1 is personalized with macros (see Fig 4) that operate on the shifted information; these macros include the read and write operations.

Data gating macro

Words one through eight in PLA1 comprise a macro that gates input field A directly to output field A′. Assuming that the value of feedback register position 13 is 1, then each 1 input in the A field selects a word in the group by the associative process. Each selected word in turn generates a corresponding output bit in the A′ field (see Fig. 4).

The setting and resetting of the feedback register position 13 (determined during the design process) is part of the state-assignment process.

Read macro

The read macro (control field = 010) is a sequential process that is initiated when the read command is received. This macro copies the character which is identified with the tag into the feedback register, and subsequently gates this character to the printer and the communication subsystem. Another macro operation (control field = 110) is used to move the tag bit to the next character so that the complete message can be read out serially for printing or transmitting. The entered message can be retransmitted or reprinted several times; therefore, a routine which restores the tag bit to the first character position is necessary. This routine involves invoking a decrement macro on the character count when the end of the message is detected in PLA1 and restoring the tag when the count has been decremented to 1.

The count of the number of characters entered is held in the keyboard subsystem. (This subsystem also has the decrement macro which will be described later.)

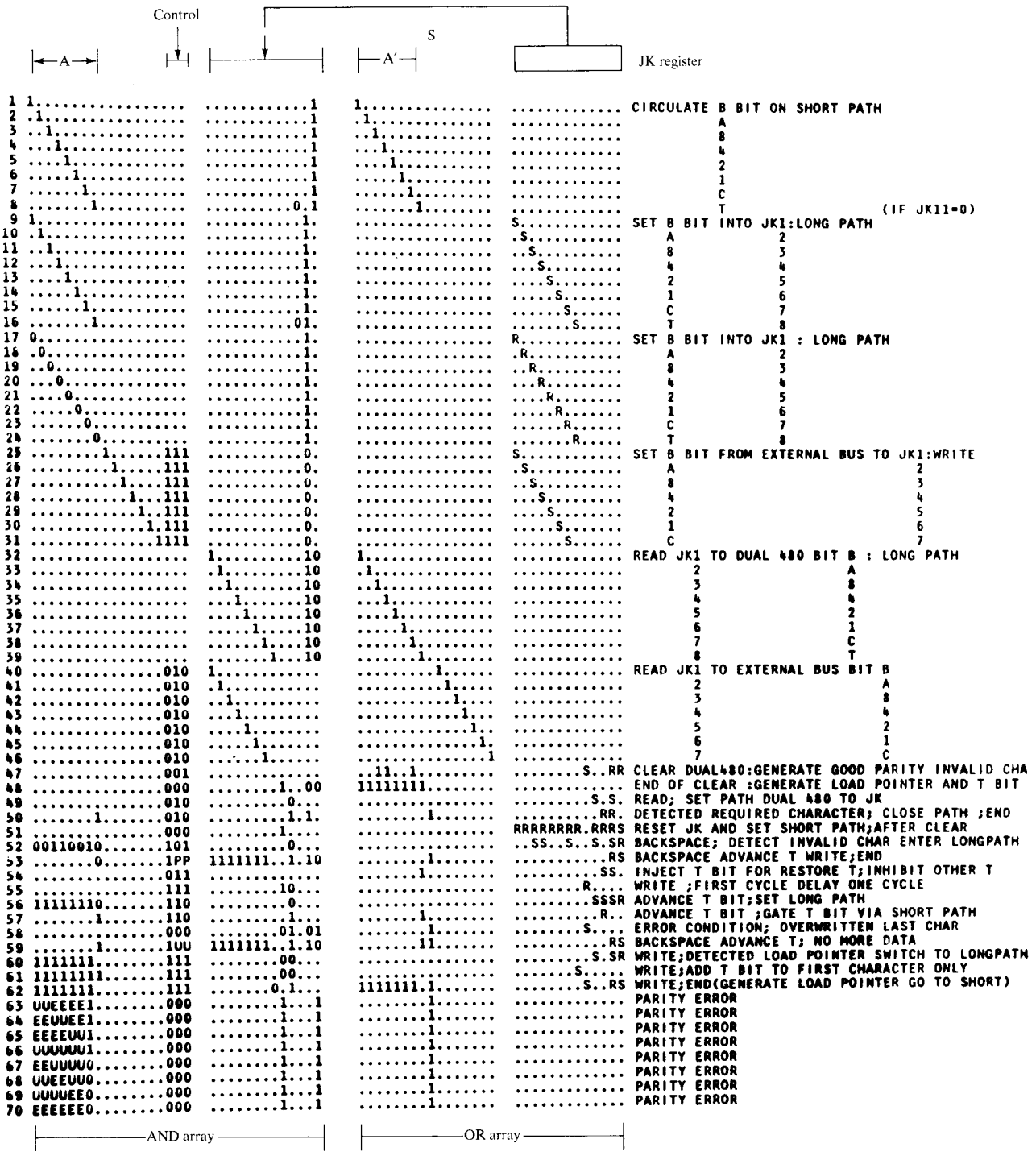


Figure 4 Personality of PLA1.

Write macro

The write macro (control field = 111) is also a sequential process, and is initiated when the write command is received. The received character is transferred from the input bus to the feedback register. This character is subsequently written into the position next to the unique marker; the rest of the message and tag are shifted back one position. The storage subsystem signals the end of each macro operation.

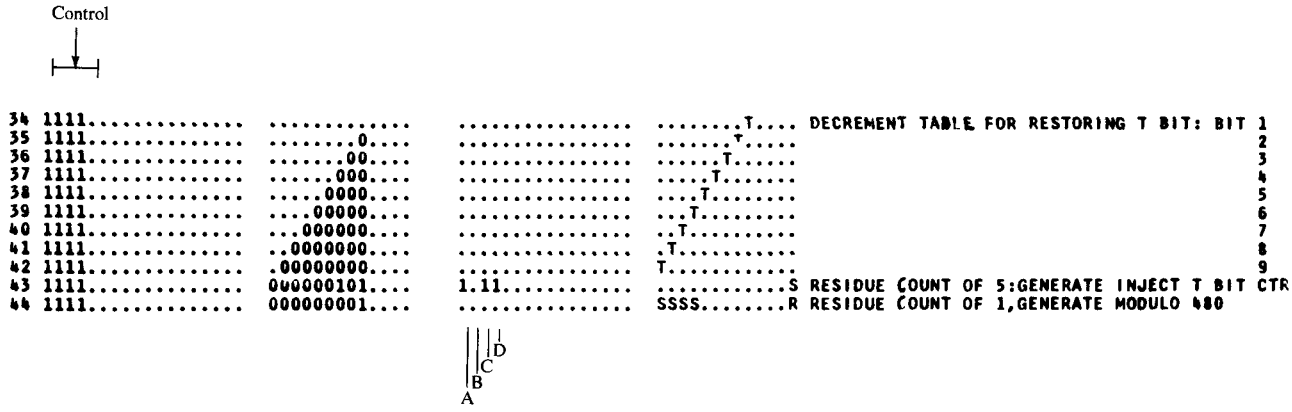


Figure 5 Decrement macro in keyboard system.

Decrement macro

The decrement macro inverts the leading significant bit and all lower order bits (Fig. 5).

The number of words selected is one more than the number of leading zero bits in the count that is held in the feedback register. Each selected word reads out a toggle (T) operator that complements the content of the associated feedback register position (words 34–42). This macro is invoked when the control field is 1111, and continues until this control is removed.

The decrement macro is used to restore the tag bit to the first character position in the dynamic shift-register store. The new tag has to be inserted when the count has been decremented to 1; however, due to the propagation delays through the arrays, this signal has to be generated when the count is being decremented from five to four (word 43).

Routine for restoring the count

The print operation is repeatable; therefore, because the count is used to restore the tag bit to the first character position, the count must be restored. The decrement macro described above is used for restoring the count. This macro is initiated when the tag is detected in the position of the last character (EOB) in the message.

The decrement process continues synchronously with the shift operation in the store until the value of the count reaches 1, at which point 480 is added to it (word 44 in Fig. 5). The decrement operation is stopped when the tag and EOB have been shifted through the 480 positions in the shift register, and are detected and erased. The count is restored to the original value since 480 cycles of decrementing have been done and 480 has been added to it.

Implementation

Macros form the primitive functions in a subsystem, and an operation such as printing the contents of the store is performed by a series of these primitive functions.

The routine for restoring the tag from the last to the first character in the store is initiated when the EOB is read and transferred to the printer subsystem. This character is decoded in the printer and a signal is sent to the controlling array (Fig. 6) which selects word 39.

The selection of word 39 sets the internal states of this PLA from the initial all-zero state to the specific state (JK8, 10 = 1), thereby selecting and reselecting word 41. This state (JK8, 10 = 1) can be defined as the "backspace state" (word 41) because it involves the backspace macro in the storage subsystem. The "backspace state" is changed to the "restore tag state" (word 43) when the storage subsystem responds that it has completed the backspace operation (word 42).

The "restore tag state" generates the decrement control to the keyboard subsystem and the read operation to the storage subsystem. Word 44 (in addition to word 43) is selected when the keyboard subsystem detects that the count has been decremented from five to four. Words 44 and 43 are ORed together to produce the command that writes the new tag bit into the storage subsystem. The exit from the "restore tag state" is achieved by matching word 45 with the operation-complete signal (S) from the storage subsystem.

Specific operations can be performed by chaining macros in series. This technique combined with the encoding and decoding properties of the PLA form the basis of the PLA's utility. The above example also shows that common macro linkage can be effected by the use of the DON'T CARE states in the state assignments of the feedback (see "Macro usage"). These techniques and the ease with which arrays perform the encoding and decoding of control information are major factors that favor array structures.

• *Simulation methods*

Because of the functional complexity of the TCU, computer simulation was required in the designing of the

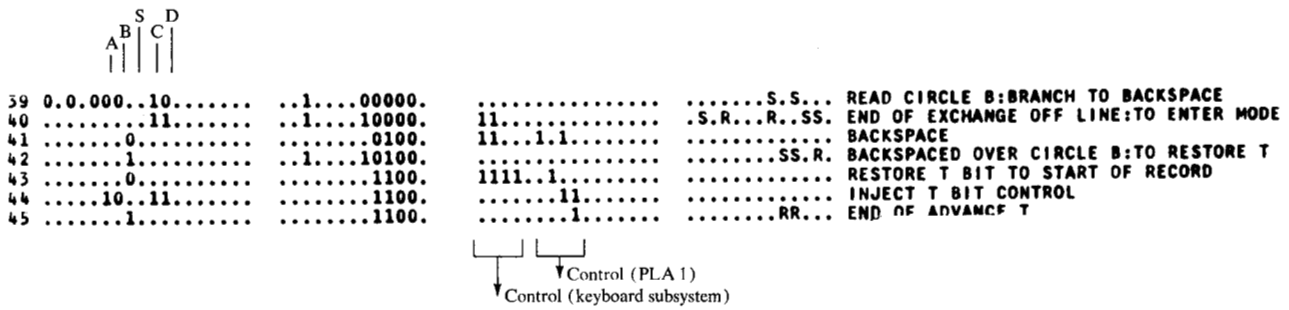


Figure 6 Part of the controlling array showing the routine for restoring the tag to the first character.

PLA version of this system. The programming language selected for this task was APL, whose array structure, power of commands, and interactive terminal approach to the problem made it the logical choice. The following paragraphs provide a brief description of the two basic parts of the PLA simulation task: chip simulation and system simulation.

Chip simulation

The general procedures for chip simulation are as follows:

1. Each PLA is entered into APL as a character array with the same dimensions and symbols as the logic representation. For example, the PLA personality shown in Fig. 4 looks the same when entered into the APL system. Two-bit decoding [5] is preserved, representing a code for the 16 logical functions (see Table 1).
2. The actual simulation was performed with a hardware representation of the array personality. This array conversion was accomplished by a program that converted the character array to a logical (0 or 1) numerical form so as to minimize APL workspace and facilitate subsequent APL processing. In the case of the PLA chip used in this experiment, a 1 represented the presence of an FET at the junction of the row and column, and 0 represented the absence of an FET.
3. The final step of chip simulation pertained to multiple-chip operations, including two-bit decoding of external inputs, the AND array and OR array operations, and operation of the JK feedbacks. The following example demonstrates the suitability of APL for such simulation tasks. Let W be a set of inputs to a PLA AND array (A1) and let A2 be the OR array. The following line in APL simulates the AND array (producing matched product rows):

$\sim A1 \vee . \wedge W$

while the operation of both the AND and OR arrays and production of outputs is simulated by

$(\sim A1 \vee . \wedge W) \vee . \wedge A2$

System simulation

Simulation at the system level involves the interconnection among chips and external inputs and outputs. The usual presence of a clock on the inputs or outputs of PLA chips makes this problem easy to handle. Our simulation of the TCU was split into the following two steps:

1. the seven PLA chips were simulated to obtain outputs as a function of inputs and array personality, and
2. the PLAs were interconnected and were also connected with external inputs/outputs of the system.

• *Build and test*

It was relatively easy to build the hardware using PLAs because the PLAs had been tested and the entire electronic package had been greatly reduced. A 40 cm × 17.75 cm (16″ × 7″) Augat board was found to be adequate to house the entire electronics of the TCU. The electronics consisted of seven PLA modules, 327 logical circuits, four dual 480-bit dynamic MOS shift register modules, oscillators, clock drivers, and discrete peripheral circuits. Figure 7 shows the module location on the Augat board. The PLA module was attached to a specially made 48-pin socket for ease of plugging the module into the board. The existing cabinet, the power supplies, and the cable of the DIP-TL version of the TCU were used (Fig. 8). The terminal utilized +48V, ±12V, +8.5V, and ±5V power supplies. The +48V drives the magnetic relays of the electric typewriter. The ±12V are for powering the TTY interface. The +8.5V is used for the interface between the PLAs that have open-drain outputs. The PLA itself uses ±5V, and the dual 480-bit shift register requires +5V and −12V.

An additional 97 circuits were used for the engineering changes related to the mechanical bounce problem of the keyboard switch of the typewriter and the product term alteration of the PLA chips.

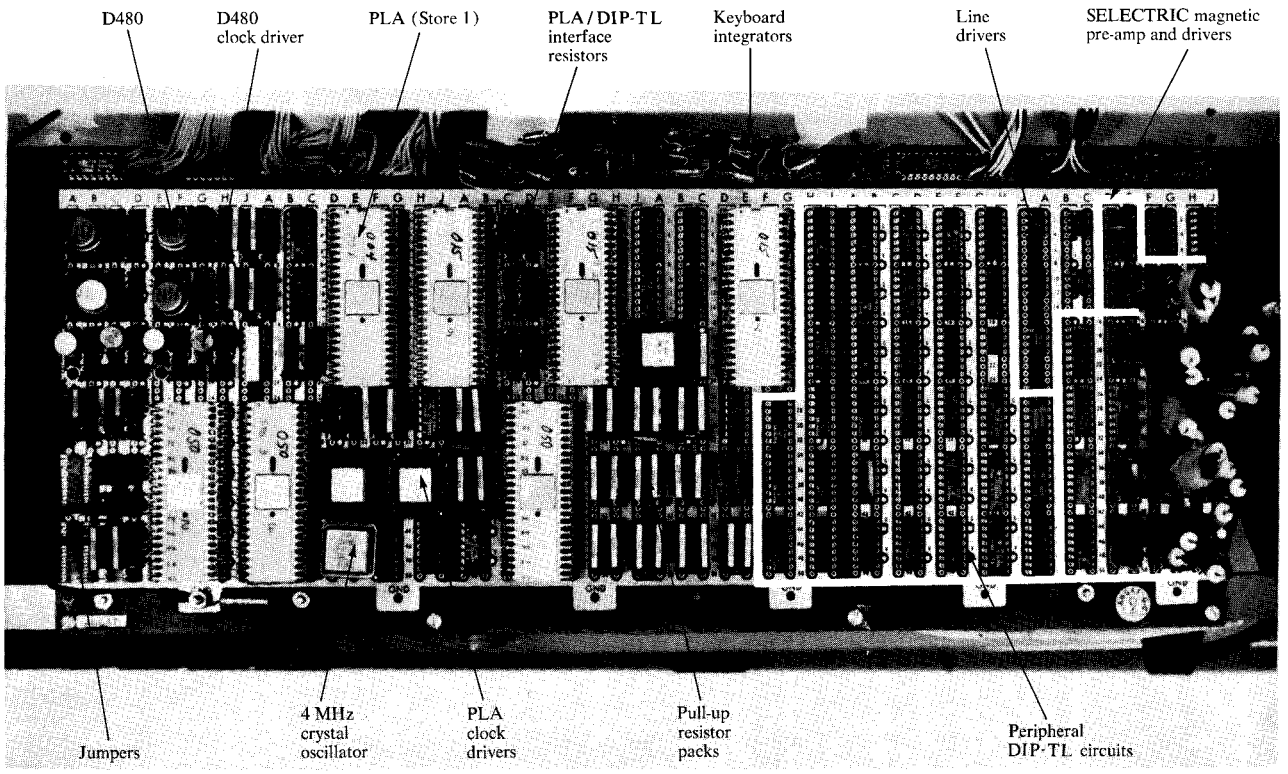


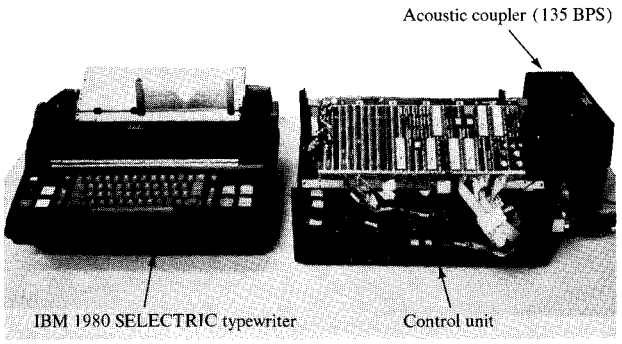
Figure 7 Electronics of the PLA version of the TCU.

The system clock, which drives the PLAs and the dual 480-bit shift registers, is operated at 1MHz (1 μs cycle time). The terminal has been successfully tested over a telephone line to a host machine simulator. The data transmission rate was 135 bits (or 15 characters) per second.

Evaluation of PLA vs DIP-TL

The PLA version of the TCU entirely duplicated the system functions of the DIP-TL version. The DIP-TL version consisted of 2,058 logical circuits, four dual 480-bit dynamic MOS shift register modules, oscillators, clock drivers, and discrete peripheral circuits. All of these components were packed on six and one-half 7.5 cm high, 10 cm wide ($3'' \times 4''$) printed circuit cards. By way of comparison, had the PLA version been packaged on 7.5 cm $\times$ 10 cm printed circuit cards instead of the Augat board, it would have required only $3\frac{1}{2}$ of these cards. The comparison between these two technologies, SSI/MSI DIP-TL random logic vs LSI array logic, revealed that the 7 PLAs replaced 1,731 logical circuits or approximately 250 logical circuits per PLA. Of these 1,731 circuits, 48% were combinational logic and 52% were sequential logic.

Figure 8 PLA version of the 7441 buffered terminal.



Summary

This paper describes a successful PLA design and implementation experience using LSI. The ease with which a small but relatively complex control unit was implemented is ample evidence that PLA is a practical means of utilizing LSI. The claims for PLA are consistent with those noted by other users in the industry [6] and are due mainly to the following PLA attributes:

- PLA exploits the "master chip" approach. PLA realizes greater benefits than those obtained by micropro-

grammed arrays due to integration of both the control and data path within the PLA.

- Documentation is much more compact and easily understood than in previous designs. The extensive macro capability allows the establishment of libraries of tested logic.
- PLA design automation procedures match an array-oriented language, APL, to an array-oriented structure, PLA, at the same time exploiting the advantages of the interactive APL language.

From a system point of view, it is notable that the applicability of the experimental findings is based on the PLA characteristics themselves and not on either the technology used (FET), or the vehicle (7441 TCU). On the one hand, bipolar PLA can be applied where high performance is required; on the other hand, PLA macro usage provides the key to application-oriented design in LSI. Since PLA does not separate data path and control functions, this programming capability can exploit system design tradeoffs and extensions not available in traditional designs. The advantages of PLA increase with LSI complexity; thus, PLA establishes a growth path to ultra LSI.

Acknowledgments

Special thanks and appreciation go to C. E. Ruoff and people in departments reporting to him for their efforts in developing the PLA chips and for their support in personalizing these chips. The authors also thank M. F. Heilweil for his assistance during the early stages of the design process, and acknowledge the help of W. D. Benedict and R. B. Battistoni in understanding the TCU and in other valuable ways.

References and notes

1. J. C. Logue, "Large Scale Integration – Status and Utilization," presented at the Second International Symposium on Microelectronics, Munich, Germany, October 1966.
2. "IBM 7441-1/1980-9 Buffered Terminal Maintenance Manual," *Report SY22-6913-1*, IBM Corporation, White Plains, New York, 1972.
3. J. W. Jones, J. F. Sears, and K. G. Taylor, "Associative Store," French Patent #2168409.
4. J. W. Jones, "Array Logic Macros," *IBM J. Res. Develop.* **19,** 120 (1975), this issue.
5. A. Weinberger, "Functional Memory Using Multistate Associative Cells," U. S. Patent #3761902.
6. W. N. Carr and J. P. Mize, *MOS/LSI Design and Application*, McGraw-Hill Book Co., Inc., New York 1972.

Received April 3, 1974

The authors are located at the IBM System Products Division Laboratory in Poughkeepsie, New York 12602.