International Journal of Research (IJR)
e-ISSN: 2348-6848, p- ISSN: 2348-795X Volume 2, Issue 4, April 2015
Available at http://internationaljournalofresearch.org

Programmable Logic Arrays

Akshay dutt (16195),Anosh Justin (16199),Gautamsaini (16211)
Department of ECE Dronacharya College of Engineering Khentawas, Farrukh Nagar-123506
Gurgaon, Haryana
Email-gautamsaini1996@gmail.com

ABSTRACT

A high performance CMOS Programmable Logic Array (PLA) circuit implemented by a new circuit technique is presented. The gate outputs are preconditioned to minimize delay using a new clocking scheme and circuit design. A multi-level logic and layout synthesis tool which utilizes the CVTL circuit technique is also presented. We describe the overall design methodology for generating the high performance PLA. The simulated benchmark circuits show that the average power-delay product is 2.1 times smaller than the pseudo-n MOS implementations for 0.25 μm process

KEYWORDS

CMOS logic circuits, delays, integrated circuits design, logical partitioning, multivalued logics, programmable logic arrays, performance evaluation.

INTRODUCTION

A **programmable logic array** (**PLA**) is a kind of programmable logic device used to implement combinational logic circuits. The PLA has a set of programmable AND gate planes, which link to a set of programmable OR gate planes, which can then be conditionally complementuce an output. This layout allows for a large number of logic functions to be synthesized in the sum of products (and sometimes product of sums) canonical forms.

PLA's differ from Programmable Array Logic devices (PALs and GALs) in that both the AND and OR gate planes are programmable.

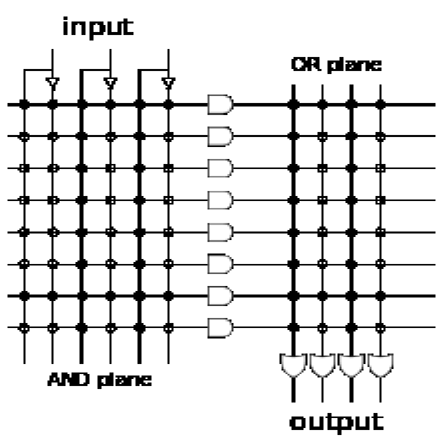


Figure: A schematic example of PLA

HISTORY

In 1970, Texas Instruments developed a mask-programmable IC based on the IBM read-only associative memory or ROAM. This device, the TMS2000, was programmed by altering the metal layer during the production of the IC. The TMS2000 had up to 17 inputs and 18 outputs with 8 JK flip flop for memory. TI coined the term *Programmable Logic Array* for this device

IMPLEMENTATION PROCEDURE

1. Preparation in SOP (sum of product form).

2. Obtain the minimum SOP form to reduce a product of terms to a minimum.
3. Decide the input connections of the AND matrix for generating the required product terms.
4. Then decide the input connections of OR matrix for generating the sum of terms.
5. Decide the connections for inverse matrix.
6. Program the PLA.

PLA BLOCK DIAGRAM:

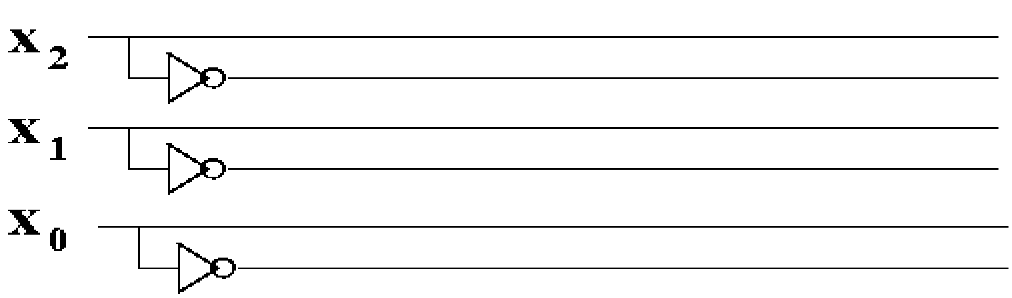
1ST BLOCK	2ND BLOCK	3RD BLOCK	4TH BLOCK	5TH BLOCK
INPUT BUFFER	AND MATRIX	OR MATRIX	INVERT/ NON INVERT MATRIX	FLIP FLOP OUTPUT BUFFER

Why Pla Over Rom

the desired outputs for each combination of inputs *could* be programmed into a read-only memory, with the inputs being loaded onto the address bus and the outputs being read out as data. However, that would require a separate memory location for *every* possible combination of inputs, including combinations that are never supposed to occur, and also duplicating data for "don't care" conditions (for example, logic like "if input A is 1, then, as far as output X is concerned, we don't care what input B is": in a ROM this would have to be written out twice, once for each possible value of B, and as more "don't care" inputs are added, the duplication grows exponentially); therefore, a programmable logic array can often implement a piece of logic using fewer transistors than the equivalent in read-only memory. This is particularly valuable when it is part of a processing chip where transistors are scarce (for example, the original 6502 chip contained a PLA to direct various operations of the processor

STARTING OUT

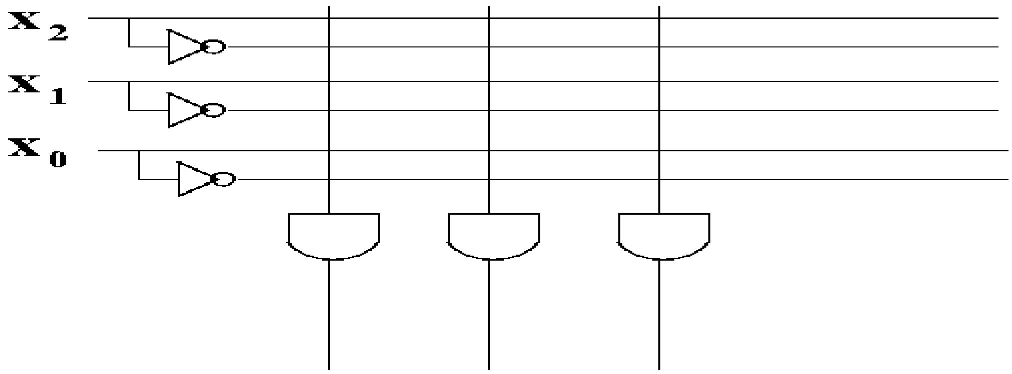
The first part of a PLA looks like:



Each variable is hooked to a wire, and to a wire with a NOT gate. So the top wire is $\mathbf{x_2}$ and the one just below is its negation, \$\mathbf{x_2}$.

Then there's x_1 and just below it, its negation, \x_1.

The next part is to draw a vertical wire with an AND gate. I've drawn 3 of them.



Let's try to implement a truth table with a PLA.

x_2	x_1	x_0	z_1	z_0
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	1	0
1	0	0	1	1
1	0	1	0	0
1	1	0	0	0
1	1	1	0	1

Each of the vertical lines with an AND gate corresponds to a minterm. For example, the first AND gate (on the left) is the minterm: **\$x_2$\$x_1$$x_0$**.

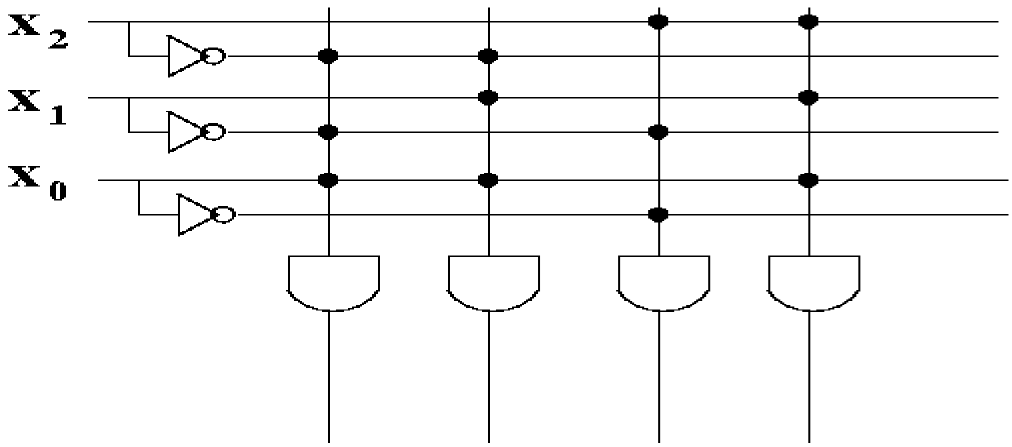
The second AND gate (from the left) is the minterm: **\$x_2$$x_1$$x_0$**.

The third AND gate (from the left) is the minterm: **x_2\$x_1$\$x_0$**.

I've added a fourth AND gate which is the minterm: **$x_2$$x_1$$x_0$**.

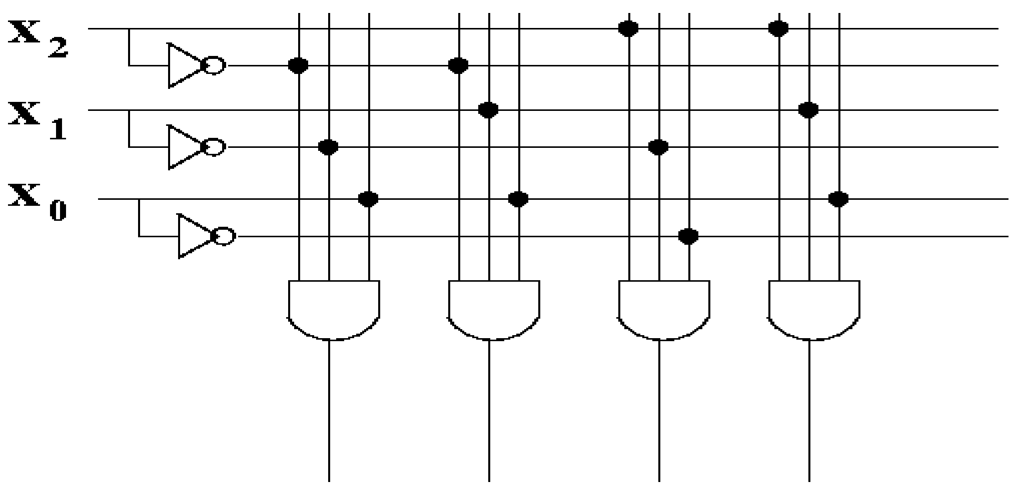
The first three minterms are used to implement **z_1**. The third and fourth minterm are used to implement **z_0**.

This is how the PLA looks after we have all four minterms.



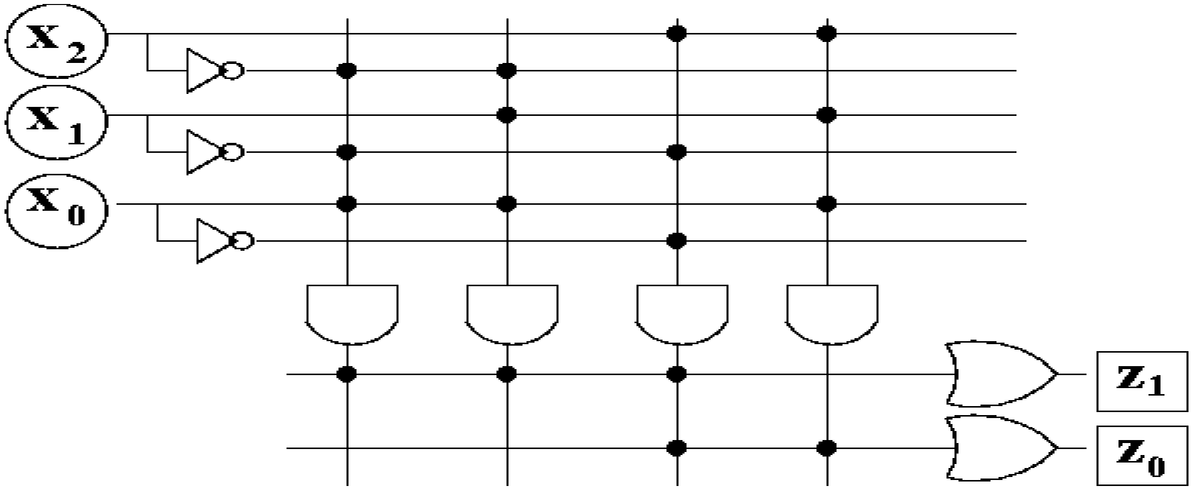
Now you might complain. How is it possible to have a one input AND gate? How can three inputs be hooked to the same wire to an AND gate? Isn't that invalid for combinational logic circuits?

That's true, it is invalid. However, the diagram is merely a simplification. I've drawn the each of AND gate with three input wires, which is what it is in reality (there is as many input wires as variables). For each connection (shown with a black dot), there's really a separate wire. We draw one wire just to make it look neat.



The vertical wires are called the AND plane. We often leave out the AND gates to make it even easier to draw.

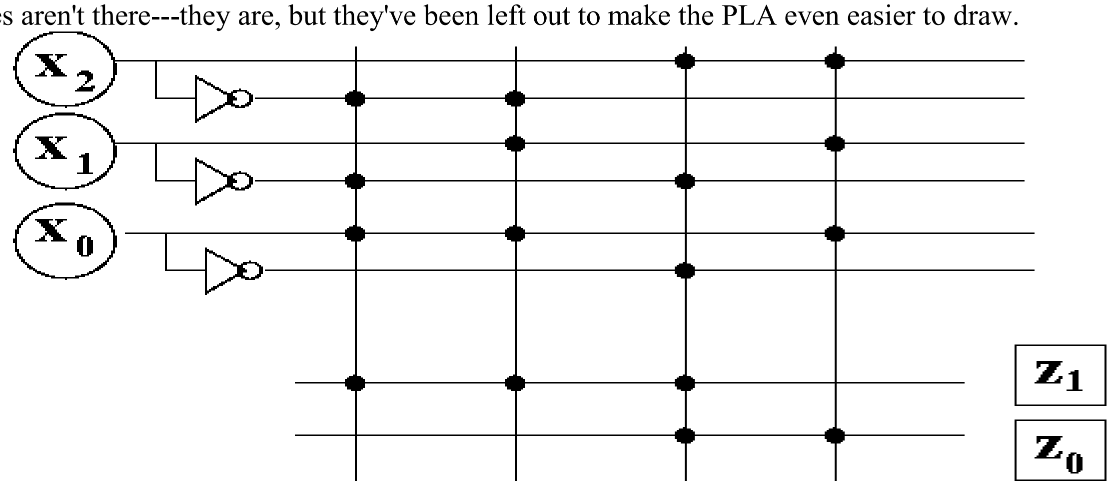
We then add OR gates using horizontal wires, to connect the minterms together.



Again, a single wire into the OR gate is really 4 wires. We use the same simplification to make it easier to read.

The horizontal wires make up the OR plane.

This is how the PLA looks when we leave out the AND gates and the OR gates. It's not that the AND gates and OR gates aren't there---they are, but they've been left out to make the PLA even easier to draw.



APPLICATIONS

One application of a PLA is to implement the control over a data path. It defines various states in an instruction set, and produces the next state (by conditional branching). [e.g. if the machine is in state 2, and will go to state 4 if the instruction contains an immediate field; then the PLA should define the actions of the control in state 2, will set the next state to be 4 if the instruction contains an immediate field, and will define the actions of the control in state 4]. Programmable logic arrays should correspond to a state diagram for the system.

Other commonly used programmable logic devices are PAL, CPLD and FPGA.

Note that the use of the word "programmable" does not indicate that all PLAs are field-programmable; in fact many are mask-programmed during manufacture in the same manner as a mask ROM. This is particularly true of PLAs that are embedded in more complex and numerous integrated circuits such as microprocessors. PLAs that can be programmed after manufacture are called FPGA (Field-programmable gate array), or less frequently FPLA (Field-programmable logic array).

The Commodore 64 home computer released in 1982 used a "906114-01 PLA" to handle system signals.

REFRENCES

[1.] *Motorola Semiconductor Data Book, Fourth Edition*. Motorola Inc. 1969. p. IC-73.

[2.] Andres, Kent (October 1970). *A Texas Instruments Application Report: MOS programmable logic arrays*. Texas Instruments. Bulletin CA-158. Report introduces the TMS2000 and TMS2200 series of mask programmable PLAs.

[3.] Greer, David L. *Electrically Programmable Logic Circuits* US Patent 3,818,452. Assignee: General Electric, Filed: April 28, 1972, Granted: June 18, 1974

[4.] Greer, David L. *Multiple Level Associative Logic Circuits* US Patent 3,816,725. Assignee: General Electric, Filed: April 28, 1972, Granted: June 11, 1974

[5.] Greer, David L. *Segmented Associative Logic Circuits* US Patent 3,849,638. Assignee: General Electric, Filed: July 18, 1973, Granted: November 19, 1974

[6.] "Semiconductors and IC's: FPLA". *EDN* (Boston, MA: Cahners Publishing) **20** (13): 66. July 20, 1975. Press release on Intersil IM5200 field programmable logic array. Fourteen inputs pins and 48 product terms. Avalanched-induced-migration programming. Unit price was $37.50

[7.] FPLA's give quick custom logic". *EDN* (Boston, MA: Cahners Publishing) **20** (13): 61. July 20, 1975. Press release on Signe tics 82S100 and 82S101 field programmable logic arrays. Fourteen inputs pins, 8 output pins and 48 product terms. NiCr fuse link programming.

[8.] Pellerin, David; Michael Holley (1991). *Practical Design Using Programmable Logic*. Prentice-Hall. p. 15. ISBN 0-13-723834-7.