Lab 11: Programmable Logic Arrays

Arturo Salinas-Aguayo
CSE 2301: Principles and Practice of Digital Logic Design
Dr. Mohammad Khan, Section 003L-1248
Electrical and Computer Engineering Department

College of Engineering, University of Connecticut
Coded in LaTeX

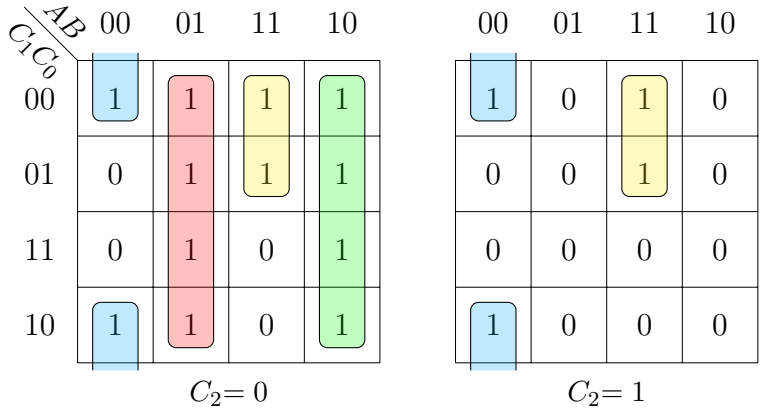
Discussion

This lab dives into describing and designing what Programmable Logic Arrays consist of and how they are packaged. Firstly, the entire course has utilized conventional simplification of boolean algebra utilizing the Quine-McClusky and the Karnaugh Map methods as well as the standard Boolean Algebra simplification methods.

However, this lab marks a transition from theoretical simplification to practical implementation, giving students a glimpse of how real-world engineers approach circuit design. In practice, engineers move beyond discrete logic chips like the 7400 series and simple hand-drawn logic gate configurations. Instead, they rely on sophisticated Computer-Aided Design (CAD) tools that integrate Hardware Description Languages (HDLs) such as VHDL or Verilog.

These tools streamline the design process by offering automated *place and route* capabilities. This allows designers to focus on high-level functionality, with software handling optimization for the targeted hardware.

Example 1 Karnaugh-Map Review
This exercise required a 5 Variable Karnaugh-Map, merely to show how much effort it takes to traditionally solve these simplification questions compared to the upcoming simplicity of the PLA.



$$F = \overline{C_0}\,\overline{A}\,\overline{B} + \overline{C_1}AB + \overline{C_2}\,\overline{A}B + \overline{C_2}A\overline{B}$$

With the PLA implementation, each minterm for each output bit is placed and routed by the software and abstracted away from the wires. For the $C_2 = 0$ block, notice how there are potential hazards at bay. These are kept seperate in order to best maximize the effect of the 3-Dimensional 5 Variable Karnaugh Map. In this case, the blue and yellow squares "wrap around" the corresponding C_2 block.

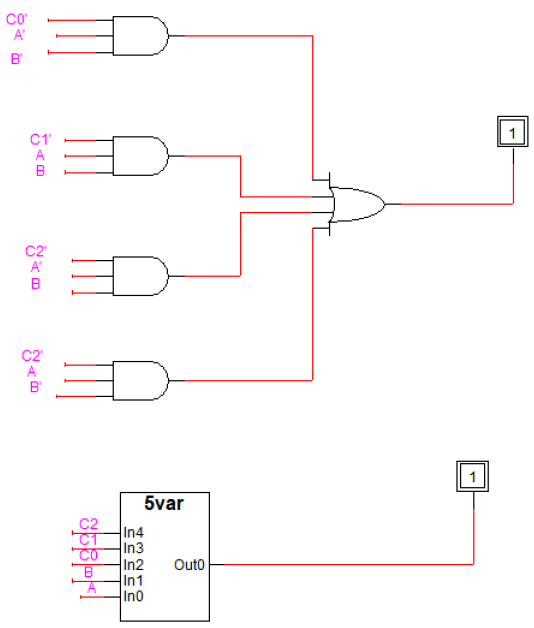


Figure 1: Traditional Gate Implementation and Black Box Implementation

The PLA form greatly reduces the complexity of visual schematic design by abstracting away the gate logic by programming it directly.

Example 2 A Continuation of the Triangle Parity

In this lab, we examined a 15-bit data stream configured with EVEN parity using a triangular code. The output identifies error positions by mapping the detected location to a four-bit decimal representation. An error is flagged whenever a mismatch in parity is detected across specific bit groupings, similar to Hamming code principles. Recall that in a Hamming code, the minimum number of parity bits p needed for m data bits satisfies $2^p \geq m + p + 1$. This ensures unique identification of single-bit errors. The triangular parity code, however, uses overlapping parity checks to pinpoint errors more precisely by leveraging redundancy within intersecting bit groups.

A	B	C	D	E	F	G	H	I	Error Position
0	0	0	0	0	0	0	0	0	None (No Error)
1	0	0	0	0	0	1	0	1	Parity Bit 5
1	1	0	0	0	0	1	0	0	Data Bit 4
1	0	1	0	0	0	0	1	1	Data Bit 3
1	0	0	1	0	0	0	1	0	Data Bit 2
1	0	0	0	1	0	0	0	1	Data Bit 1
0	1	0	0	0	1	0	0	1	Parity Bit 9
0	1	1	0	0	1	0	0	0	Data Bit 8
0	1	0	1	0	0	1	1	1	Data Bit 7
0	1	0	0	1	0	1	1	0	Data Bit 6
0	0	1	0	0	1	1	0	0	Parity Bit 12
0	0	1	1	0	1	0	1	1	Data Bit 11
0	0	1	0	1	1	0	1	0	Data Bit 10
0	0	0	1	0	1	1	1	0	Parity Bit 14
0	0	0	1	1	1	1	0	1	Data Bit 13
0	0	0	0	1	1	1	1	1	Parity Bit 15

Table 1: Truth Table for Error Detection (5 Inputs, 4 Outputs)

Example 3 The Overflow of a 3 bit by 3 bit full Adder

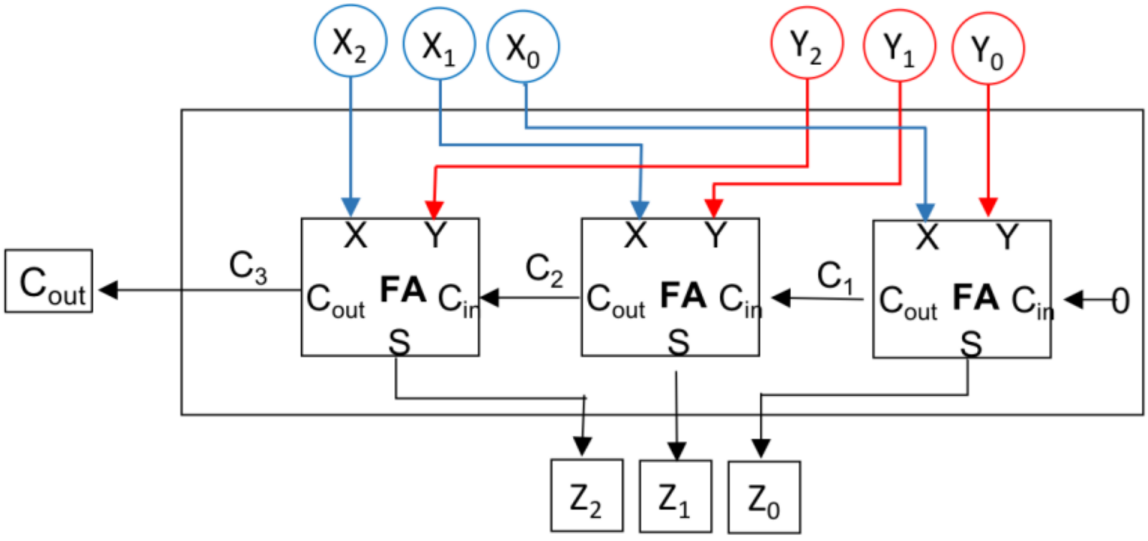


Figure 2: The 3x3 bit Full Adder Circuit

The output **OVERFLOW** bit, which can be used as a signal for the logic to indicate a potentially erroneous operation, can be determined quickly by comparing the MSB of Y and X. The equation can simply be put as:

$$\textbf{OVERFLOW} = \overline{X_2}\,\overline{Y_2}Z_2 + X_2Y_2\overline{Z_2}$$

Example 4 Programmable Array Logic
For this, the task is to draw a PLA for the equation:

$$F = \overline{C}A + BC$$

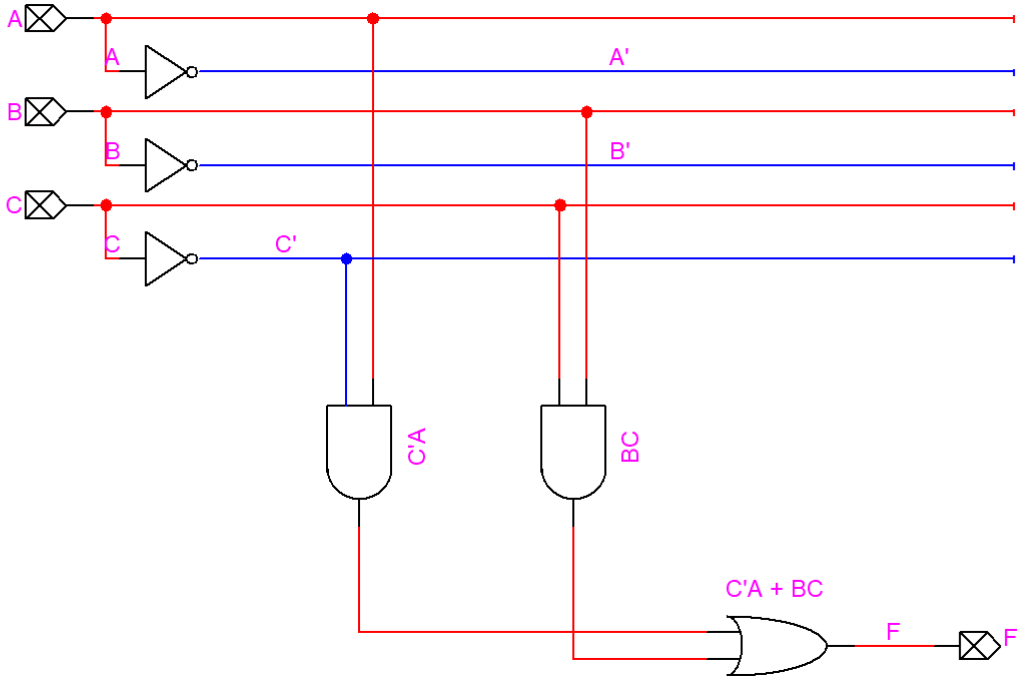


Figure 3: $F = \overline{C}A + BC$

This is done similarly to creating combinational logic at the begining of the course. Every wire being drawn out and extended to the combination of AND and OR gates is the characteristic of the PLA.