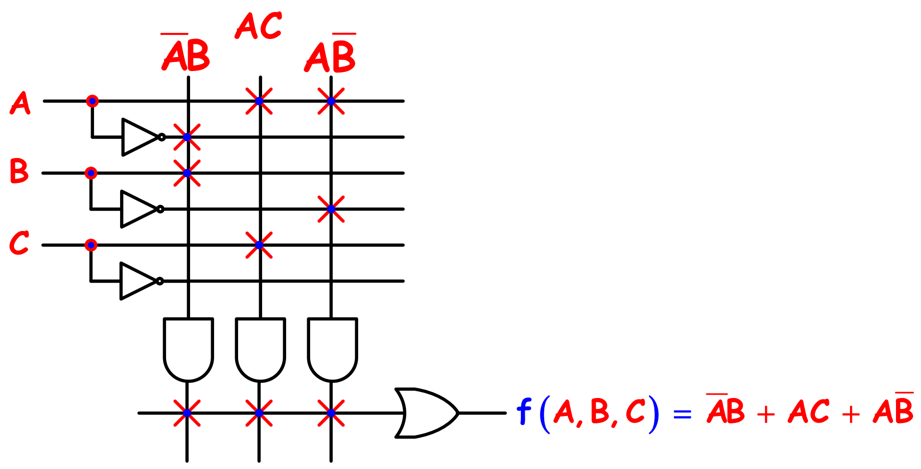
5.4 The Programmable Logic Array (PLA) and the Field Programmable Logic Array (FPLA)

We can now redraw the previous AND/OR array circuit in a more compact way. With reference to the figure below, the AND function is represented by the vertical lines and the single input AND gates represent the **pull-up resistors.**

The OR function is represented by a horizontal line with a single input OR gate symbol representing the **pull down resistor**. The X symbols **represent the actual diodes connections which have been left in place. Any connection that doesn't have an X is no longer present (burnt out).**

When we put this array into a chip, we call it a Programmable Logic Array.



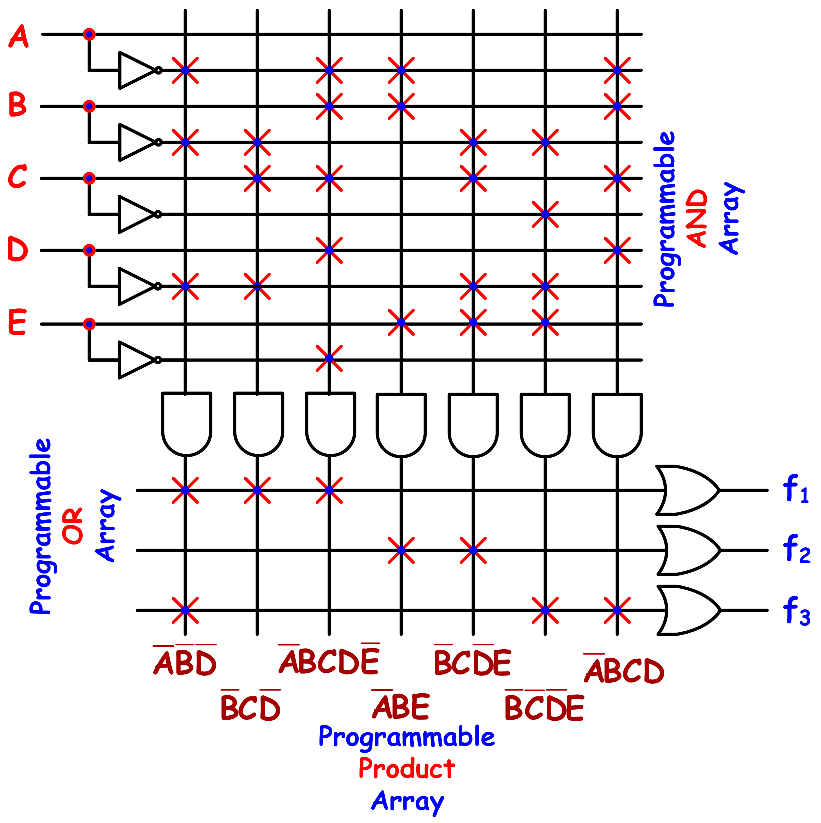
5.4.1 PLA Example

Example: **Implement the following three functions below using a PLA.**

$$f_1(A,B,C,D,E) = \overline{A}\ \overline{B}\ \overline{D} + \overline{B}C\overline{D} + \overline{A}BCD\overline{E}$$

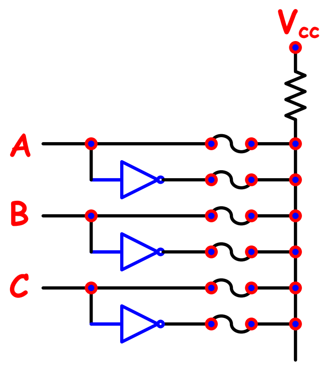
$$f_2(A,B,C,D,E) = \overline{A}BE + \overline{B}C\overline{D}E$$

$$f_3(A,B,C,D,E) = \overline{A}\ \overline{B}\ \overline{D} + \overline{B}\ \overline{C}\ \overline{D}E + \overline{A}BCD$$


A
B
C
D
E
Programmable AND Array
Programmable OR Array
f_1
f_2
f_3
$\overline{A}\,\overline{B}\,\overline{D}$
$\overline{B}C\overline{D}$
$\overline{A}BCD\overline{E}$
$\overline{A}BE$
$\overline{B}C\overline{D}E$
$\overline{B}\,\overline{C}\,\overline{D}E$
$\overline{A}BCD$
Programmable Product Array

Up until now we have talked about PLA's in general. **PLA's are programmed at the factory with input from the user.** Also available is one which is **Programmable by the user** called a **FIELD Programmable Logic Array (FPLA)**. Everything available in the PLA is also available in the **FPLA**.

In the case of the **FPLA**, each diode has a **fuse-able link in series with it.** This fuse can be "**blown**" away by **passing a high current through it**, making the output independent of the corresponding input.



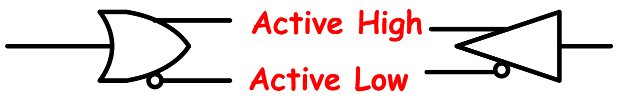
Remember that the upper **horizontal lines are the diodes**. They are not explicitly shown in this figure.

5.4.2 PLA and FPLA add-on features

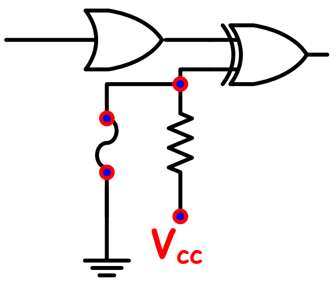
5.4.2.1 Active High or Active Low output

Besides **Product terms** and **Sum terms**, there are other features which are quite often included in a PLA. Instead of having **ACTIVE HIGH outputs**, the 'OR gates' in the 'OR array' are sometimes changed to '**NOR gates**' to provide an **ACTIVE LOW output**.

Sometimes it is nice to have **both ACTIVE HIGH** and **ACTIVE LOW outputs** available. So in this case, the 'OR gates' in the 'OR array' are changed to **Complementary Output Gates** (also known as a Buffer/Inverter Combo).



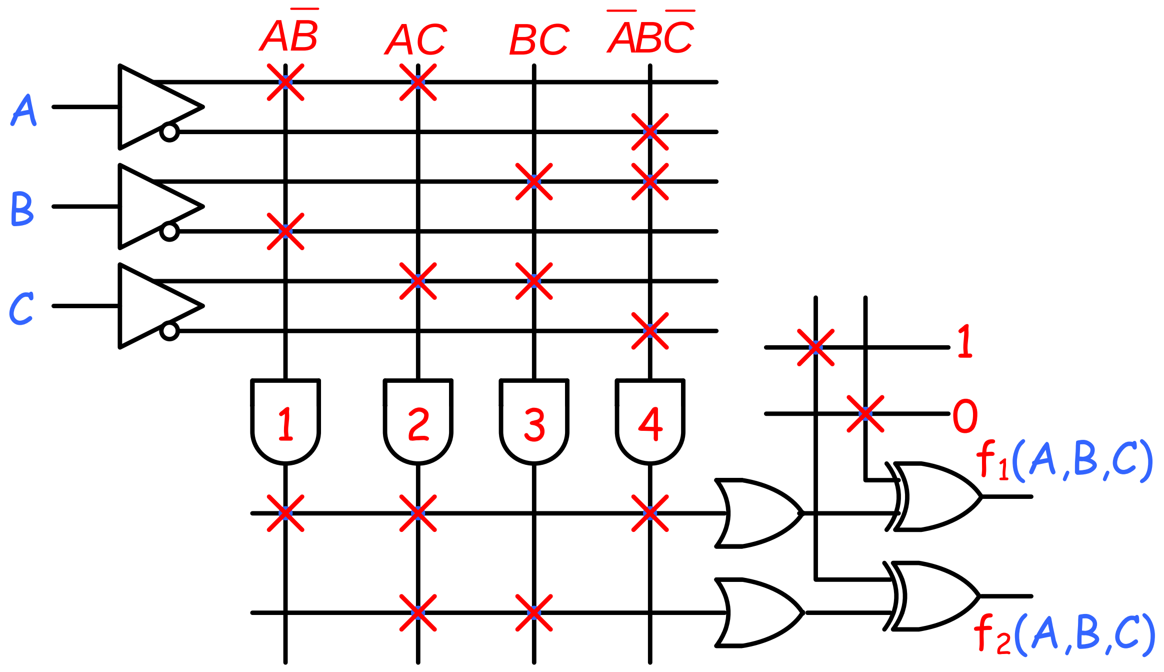
But at times, what you really want to do is to **Program the polarity of the output**. Here again, the **fuse-able link** is used in parallel with a pull-up resistor tied to Vcc to allow the programmer the choice of how he wants the output to be represented.



5.4.2.1.1 PLA Example

$$f_1(A,B,C) = A\overline{B} + AC + \overline{A}B\overline{C}$$

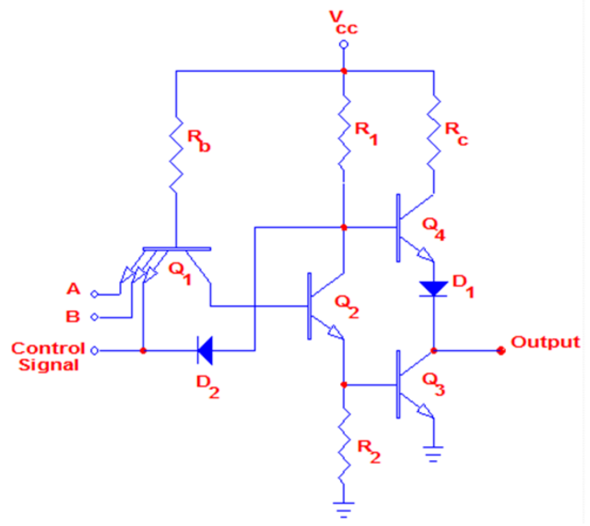
$$f_2(A,B,C) = \overline{AC + BC}$$



5.4.2.2 Tri-state outputs

In a computer, there is **only one data bus and all the devices which use or supply data must share it.** If a device isn't using the bus, its output **MUST be in a "High Z"** or **"Open Circuit"** state so that the bus isn't **"Loaded down"**. So, these types of devices need to have a "**TRI-STATE Output stage**" which will allow a **"0", "1", and a "High Impedance (Z)"** as its possible output states.

The circuit to the right is a standard **NAND gate** with a **tri-state output**. When the control signal is "**HIGH**", **Diode D2 and the EB junction of Q1 associated with the control signal are off.** Therefore, the circuit will **act just like a standard NAND gate**. When the **Control Signal goes "LOW"**, **Q1 will be turned on to saturation independent of A or B inputs and Q2 will go into cutoff.** Normally, in a standard circuit, this will cause the collector of **Q2** to see **Vcc**, but in this circuit the **"LOW" control signal will forward bias D2, clamping the collector voltage V_{c2} at approximately 0.8v.** At this voltage, **Q3, Q4, and Q1 are all off.** Thus the output is an **open circuit** with only the **PN junction leakage current** flowing through the output.[1] **When the**



[1] Gopalan, K. Gopal, **Introduction to digital electronic circuits,** Irwin Publishing, 1995, pp198-99.

output has this high impedance between ground and Vcc, the gate can neither supply or sink more than a few micro-amperes of current and can neither drive nor load any device connected to it. The gate is effectively disconnected from the circuit.

Common tri-state gates include the NAND, NOR, INVERTER, and DRIVER (BUFFER)[2].

5.4.2 Programming the FPLA

The FPLA would be extremely difficult to program by hand, so there are software and hardware devices which do it for you. The programmer needs to remember that the standard FPLA is a **PROGRAM ONCE** device. You can't go back. In addition, the more changes that need to be made in the device, the more chances there are of having an error, i.e., the more connections which need to be burned out, the more chance that an improper connection will result due to a poor burn. In addition, the PLD devices only come with a certain number of product terms possibilities and output possibilities. For both these reasons, the designer needs to simplify the logic expressions as much as possible first. Then the user can let the computer program the device for him.

[2] Stonham, T. J., **Digital Logic Techniques**, 3rd edition, p 138, Chapman and Hall, 1996

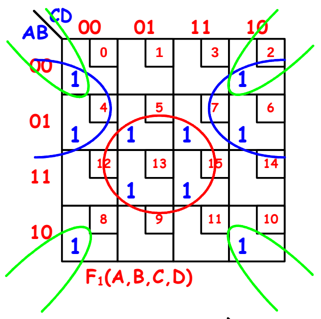
Example: Implementing multiple functions with limited product lines available.

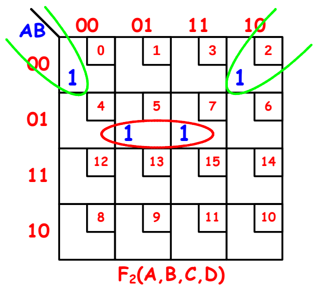
Implement the following three functions on a single 4 input, 3 output, 5 product line FPLA.

$$f_1(A,B,C,D) = \bar{B}\bar{D} + BD + \bar{A}B$$
$$f_2(A,B,C,D) = \bar{A}\bar{B}\bar{D} + \bar{A}BD$$
$$f_3(A,B,C,D) = \bar{A}B\bar{D} + ABD$$

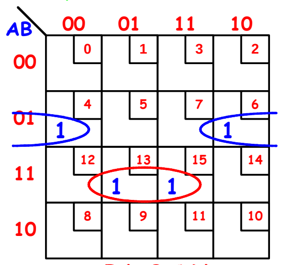
These three functions require a total of <u>7 product lines</u> to implement and there are only 5 product lines available.

The best way to solve this dilemma is to expand each term into its **Numerical Canonical form.** Then, K-map each term in its own K-map. They should be on the same page so they can be compared.


F₁(A,B,C,D)


F₂(A,B,C,D)

Examine each k-map to see if there can be terms which when grouped will share a group in one or more of the other K-maps.

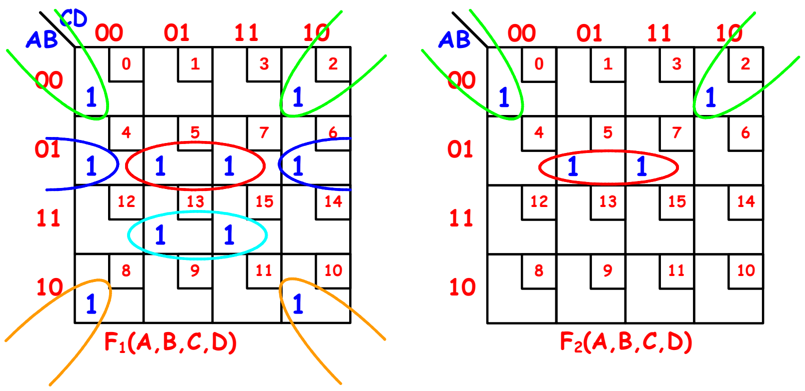

F₃(A,B,C,D)

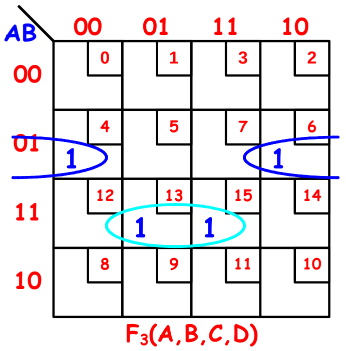
The idea is to **reduce the number of groups to 5 or fewer**.

Essentially you are **<u>de-simplifying</u>** the expressions so that they will share terms and fit on the smaller number of product lines.

Continued on the next page

The K-maps to the right represent the same three functions but have been **grouped differently**. Note that each function is not necessarily grouped into its simplest form but the **3 functions together will only take 5 product lines vice the original 7 product lines.**

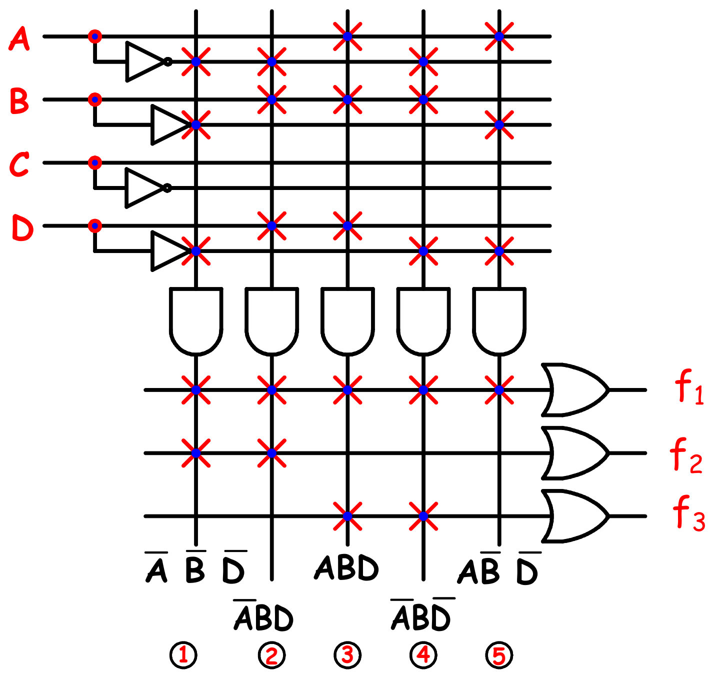




$$f_3(A,B,C,D) = \underbrace{ABD}_{\text{TERM 3}} + \underbrace{\overline{A}B\overline{D}}_{\text{TERM 4}}$$

$$f_2(A,B,C,D) = \underbrace{\overline{A}\,\overline{B}\,\overline{D}}_{\text{TERM 1}} + \underbrace{\overline{A}BD}_{\text{TERM 2}}$$

$$f_1(A,B,C,D) = \underbrace{\overline{A}\,\overline{B}\,\overline{D}}_{\text{TERM 1}} + \underbrace{\overline{A}BD}_{\text{TERM 2}} + \underbrace{ABD}_{\text{TERM 3}} + \underbrace{\overline{A}B\overline{D}}_{\text{TERM 4}} + \underbrace{A\overline{B}\,\overline{D}}_{\text{TERM 5}}$$


A
B
C
D
f₁
f₂
f₃
$\overline{A}\,\overline{B}\,\overline{D}$
$\overline{A}BD$
ABD
$\overline{A}B\overline{D}$
$A\overline{B}\,\overline{D}$
1
2
3
4
5