ECE 341

Lecture # 4

Instructor: Zeshan Chishti
zeshan@ece.pdx.edu

October 8, 2014

Portland State University

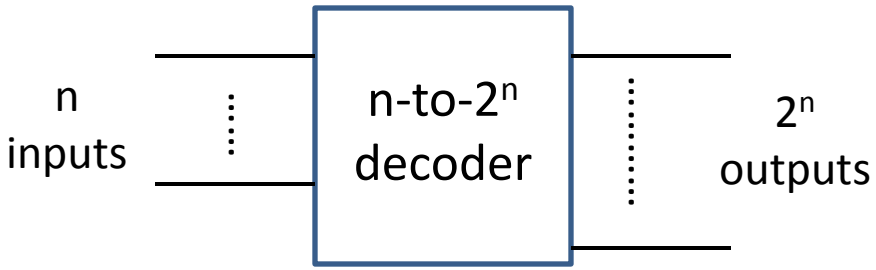
Lecture Topics

- Decoders
- Multiplexers
- Programmable Logic Devices (PLDs)
  - General Structure of PLDs
  - PLA
  - PAL
  - CPLD

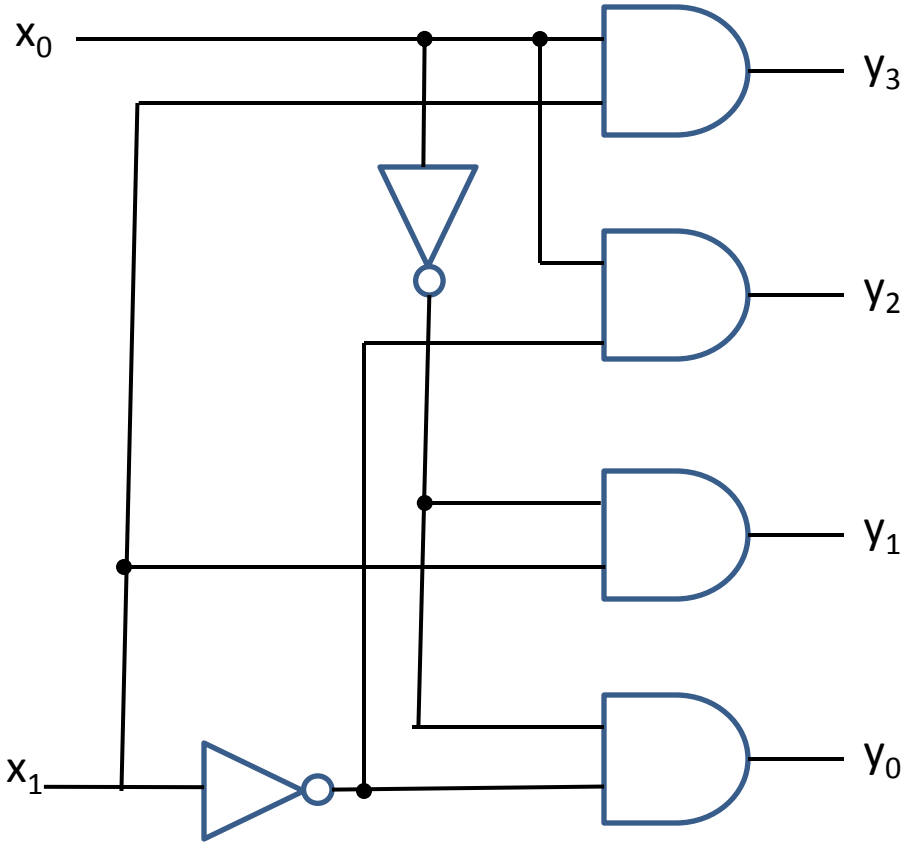
- Reference:
  - Appendix A: Sections A.9 to A.12

Decoder

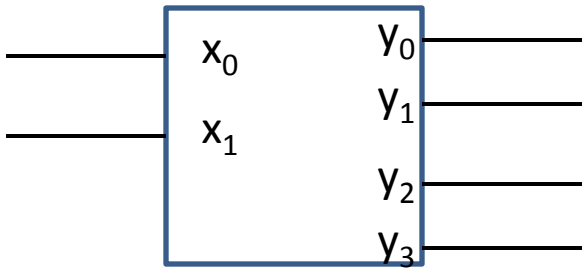
- Decoder is used to decode encoded information
- A decoder has ***n*** data inputs and 2^n outputs
- For any input data combination, a unique output line has logic value 1 and all the other outputs have the value 0 (one-hot encoding)
- Example: Consider an instruction which performs 8 different functions. A 3-bit field may be used to denote 1 out of the 8 possible functions. A 3-to-8 decoder would decode any instance of the instruction to determine the desired function



2-to-4 Decoder Circuit

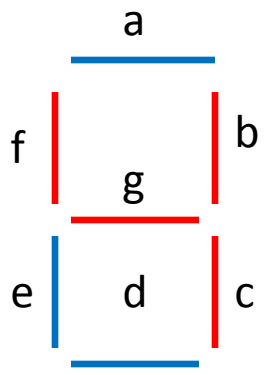


x_0	x_1	Active Output
0	0	y_0
0	1	y_1
1	0	y_2
1	1	y_3



BCD to Seven-Segment Display Decoder

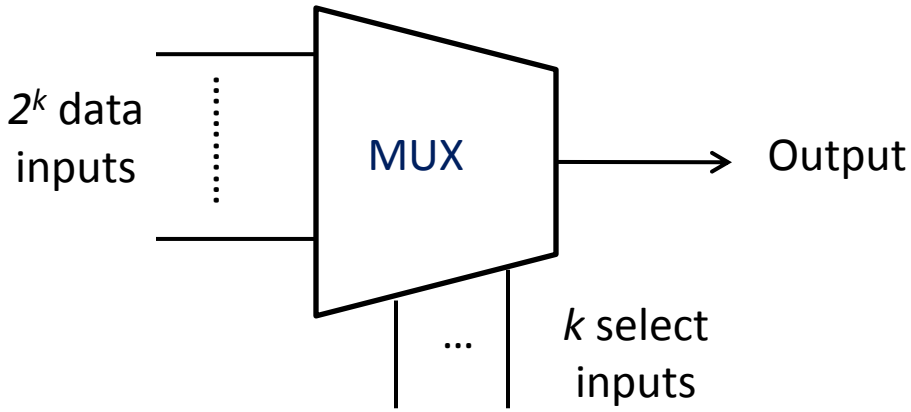
- In typical decoders, only one output line asserted for an input combination
- There are other special decoders, where multiple lines may be asserted
- Example: BCD (binary-coded decimal) to seven-segment display decoder
  - Input: a 4-bit BCD digit
  - Output: 7 bits (*a* through *g*) corresponding to 7 display segments
  - Any number from 0 to 9 can be displayed by turning some lights on and others off
  - Multiple outputs may be asserted at once
    - E.g., if input is 0100 (digit 4): *b*, *c*, *f* and *g* are on
  - See Figure A.36 in book for truth table and circuit



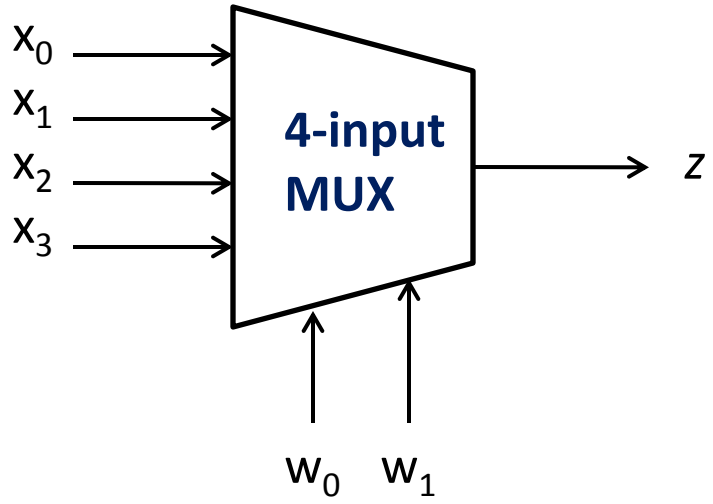
Multiplexer

- A multiplexer (MUX) circuit has:
  - 2^k data inputs
  - k select inputs
  - One output
- A MUX passes the signal value on one of its data inputs to the output based on the value of the *select* signals
  - Can be used for gating of data that may come from many different sources

Multiplexer Symbol



A 4-Input Multiplexer



w_0	w_1	z
0	0	x_0
0	1	x_1
1	0	x_2
1	1	x_3

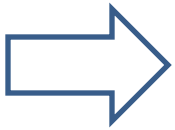
$$z = x0\overline{w0}\,\overline{w1} + x1\overline{w0}w1 + x2w0\overline{w1} + x3w0w1$$

- Logic circuit implementation shown in Figure A.37
- Example usage: A register can be loaded from one of four distinct sources by using a 4-input MUX

Logic Functions using MUXes

- MUXes can be used to synthesize logic functions
- Example: Consider a function *f* of 3 input variables x_0, x_1 and x_2 defined by following truth table. This function can be synthesized with a 4-input mux

x_0	x_1	x_2	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

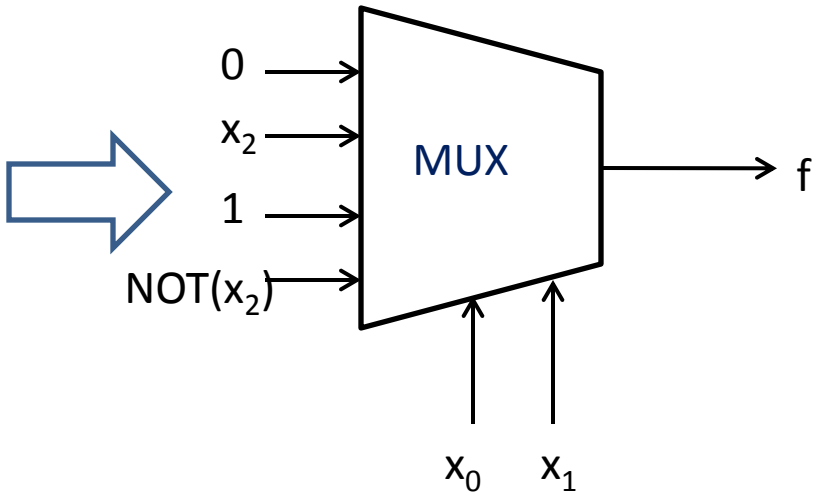


x_0	x_1	f
0	0	0
0	1	X_2
1	0	1
1	1	NOT(x_2)

Logic Functions using MUXes

- MUXes can be used to synthesize logic functions
- Example: Consider a function *f* of 3 input variables x_0, x_1 and x_2 defined by following truth table. This function can be synthesized with a 4-input mux

x_0	x_1	x_2	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



Practice Problem

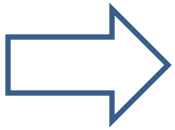
- Synthesize the function f_1 in the following truth table by using a 4-input mux with y_1 and y_2 as selector inputs.

y_0	y_1	y_2	f_1
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Solution

- Transform the truth table to use y1 and y2 as inputs

y_0	y_1	y_2	f_1
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

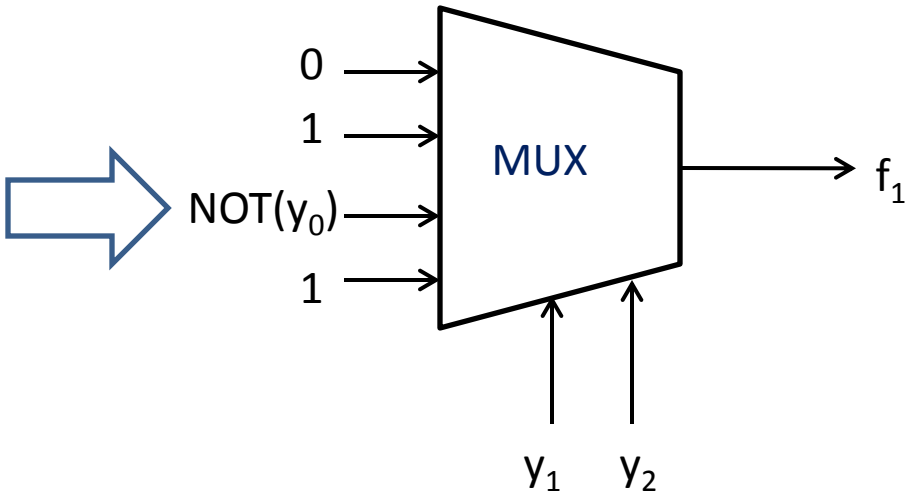


y_1	y_2	f_1
0	0	0
0	1	1
1	0	NOT(y_0)
1	1	1

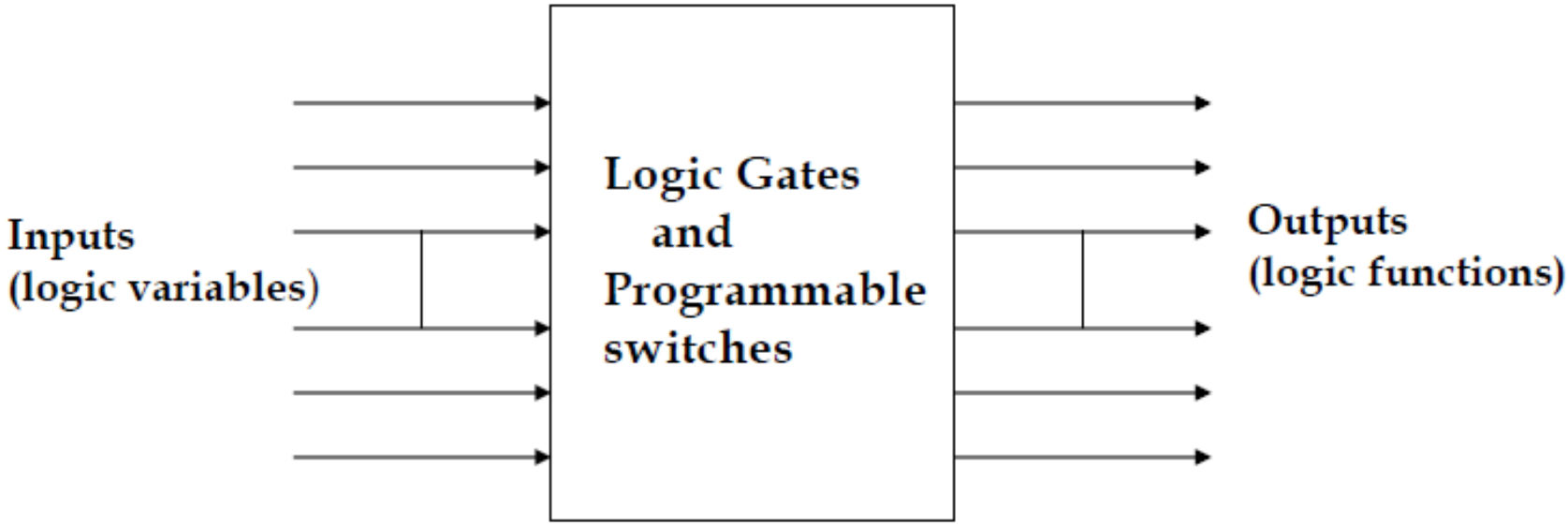
Solution

- 4-input MUX with y_1 and y_2 as selector inputs

y_0	y_1	y_2	f_1
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

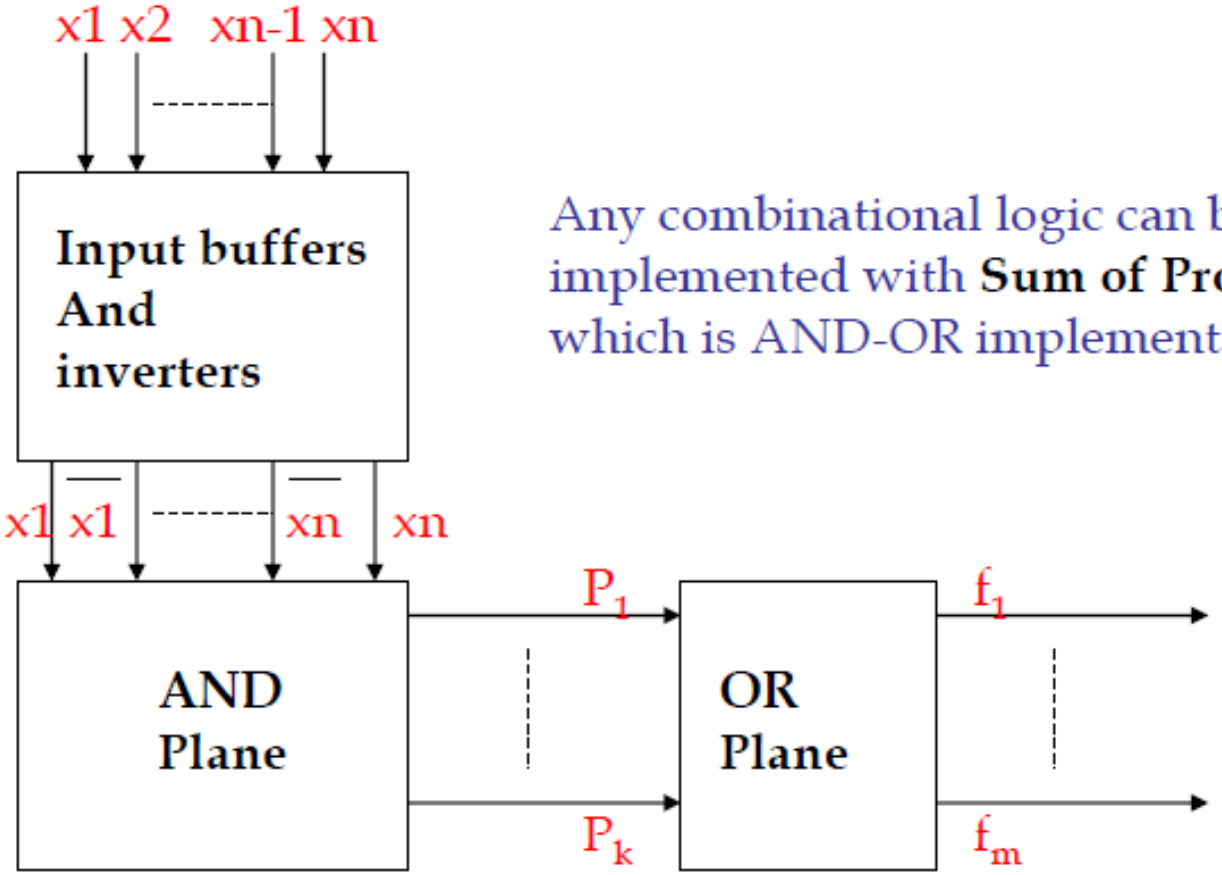


Programmable Logic Devices (PLDs)



A PLD is a customizable device that can be programmed with switches to implement a variety of combinational functions

General Structure of PLD



Any combinational logic can be implemented with **Sum of Product** which is AND-OR implementation.

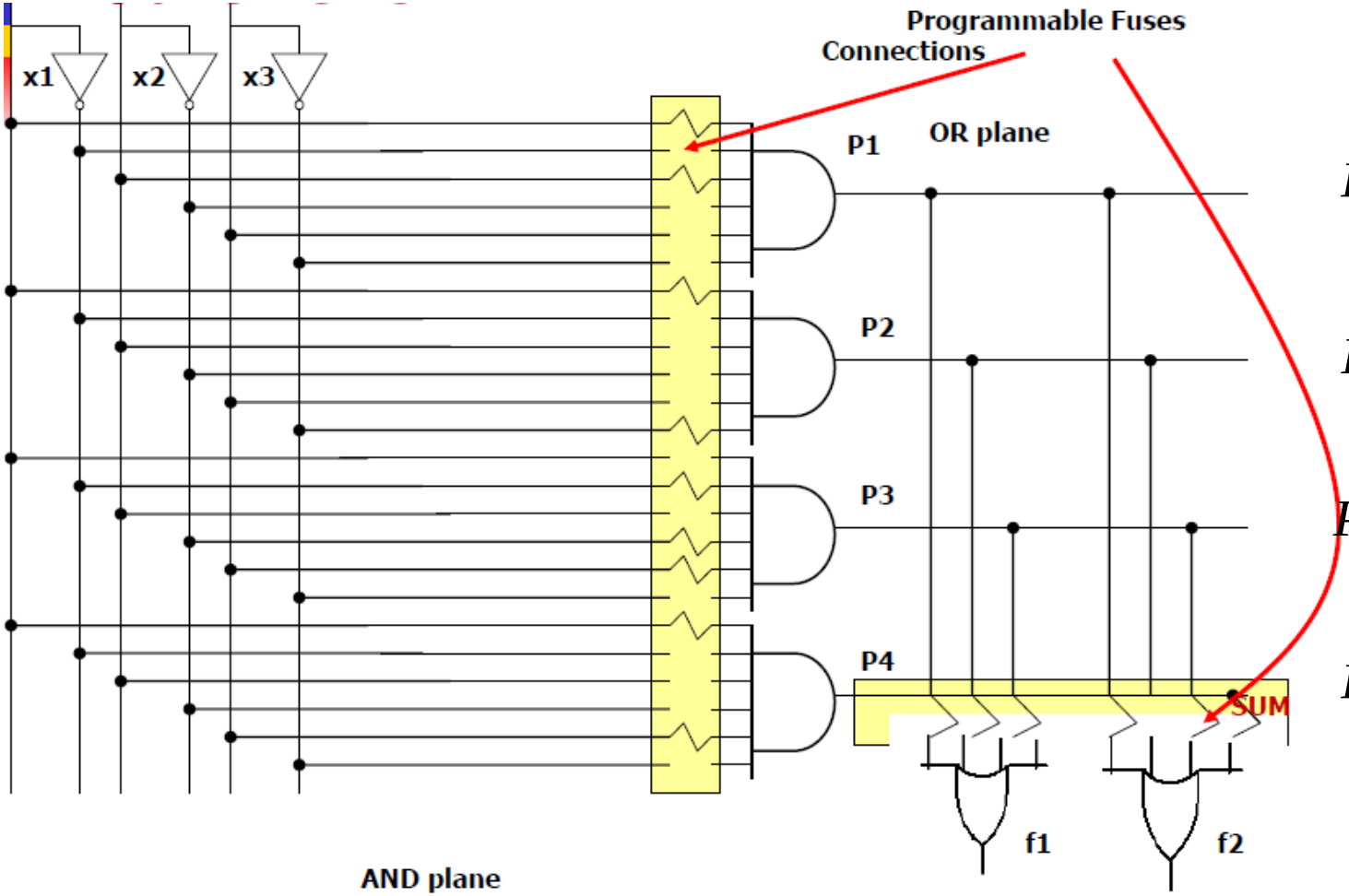
To make a PLD customizable, AND/OR arrays may use programmable switches

PLD Functionality Table

AND	OR	Device
Fixed	Fixed	Not Programmable
Fixed	Programmable	PROM
Programmable	Fixed	PAL
Programmable	Programmable	PLA

We will focus only on PLA and PAL

Programmable Logic Array (PLA)



$P1 = x1x2$

$P2 = x1\overline{x3}$

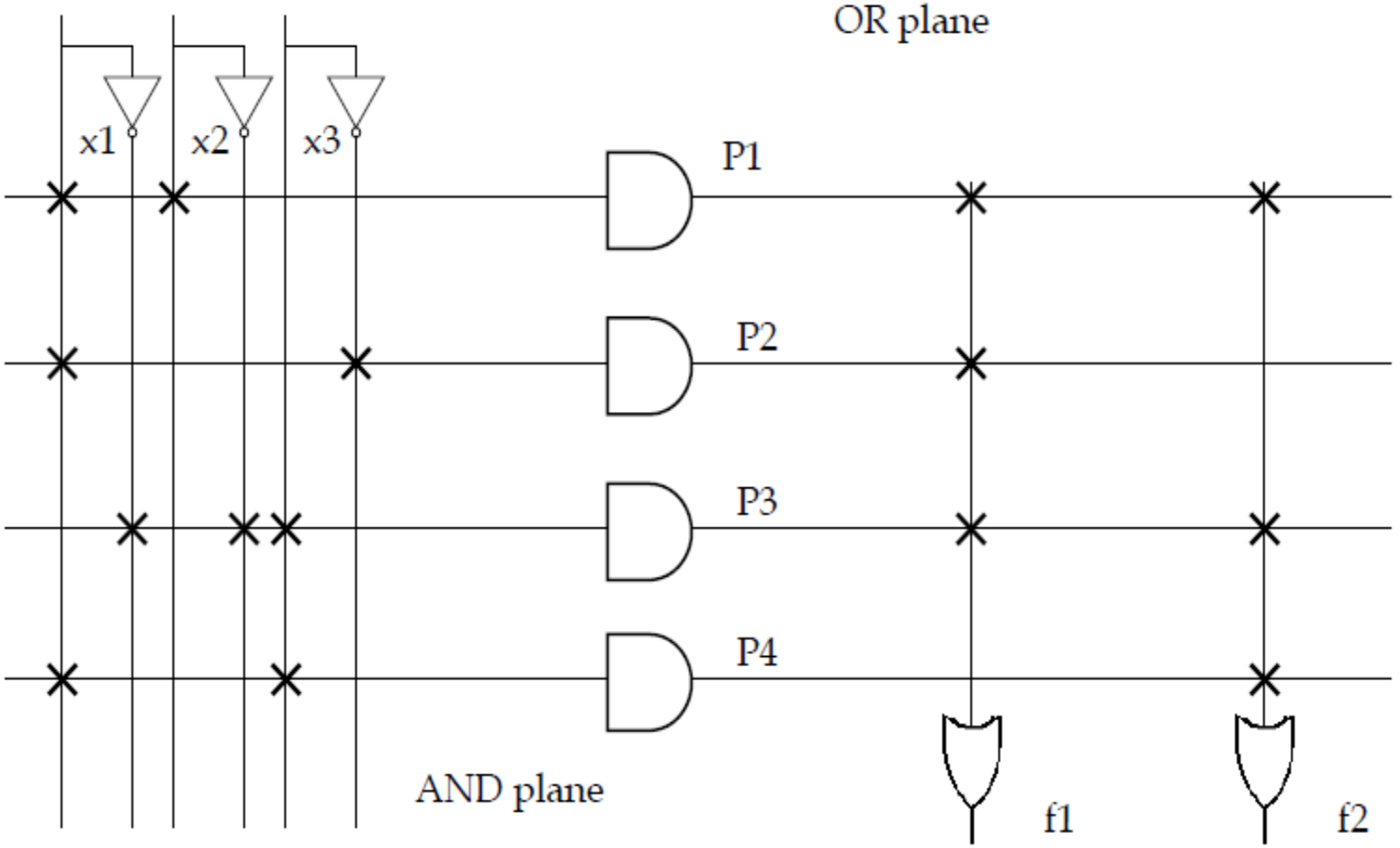
$P3 = \overline{x1}\,\overline{x2}x3$

$P4 = x1x3$

$f1 = x1.x2 + x1.\overline{x3} + \overline{x1}.\overline{x2}.x3$

$f2 = x1.x2 + \overline{x1}.\overline{x2}.x3 + x1.x3$

PLA in Simplified Form



$$P1 = x1x2$$

$$P2 = x1\overline{x3}$$

$$P3 = \overline{x1}\,\overline{x2}x3$$

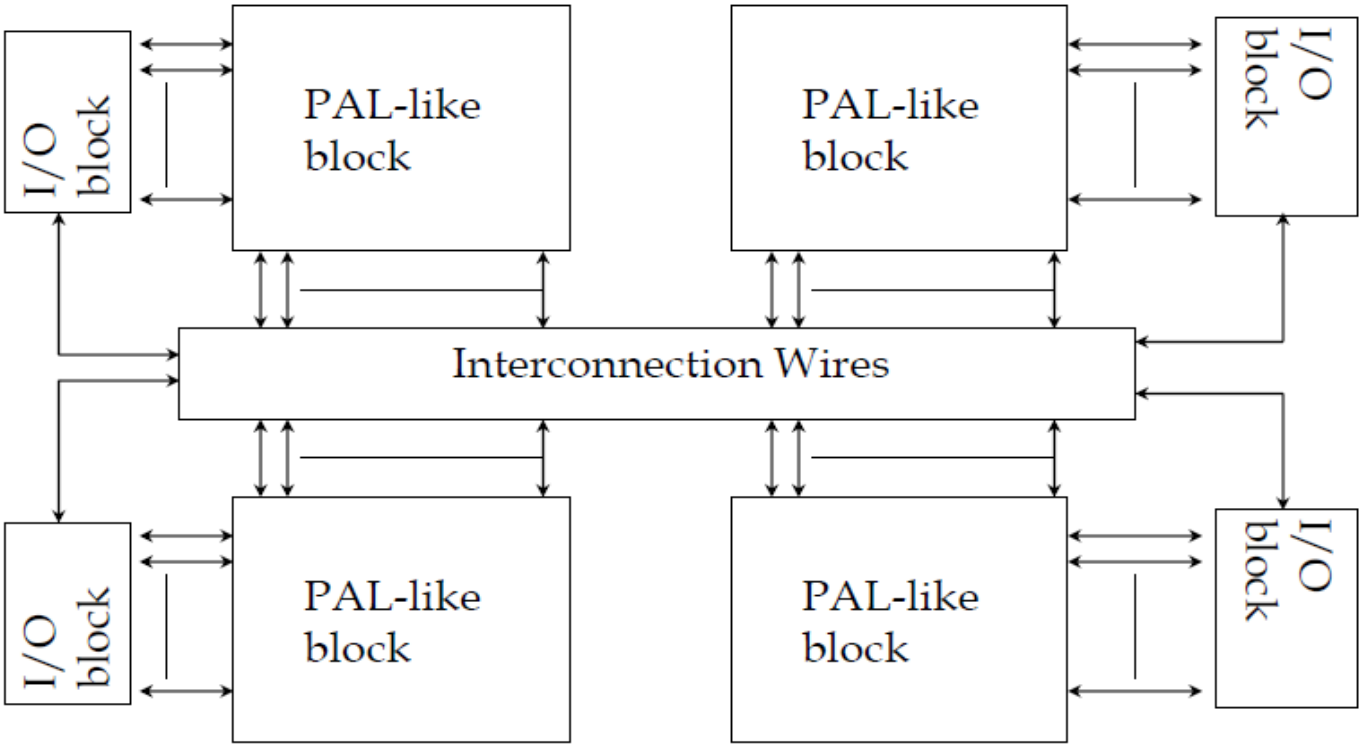
$$P4 = x1x3$$

$$f1 = x1.x2 + x1.\overline{x3} + \overline{x1}.\overline{x2}.x3 \qquad f2 = x1.x2 + \overline{x1}.\overline{x2}.x3 + x1.x3$$

Programmable Array Logic (PAL)

- PLA: *Both* the AND and OR arrays are *programmable*
- PAL: *Programmable* AND array, *Fixed* OR array
  - AND gates permanently connected to specific OR gates
    - # of product terms in a function limited by # of AND gates connected to a OR gate
    - Product terms cannot be shared amongst multiple output functions
  - See example of PAL in Figure A.42
- PLA vs. PAL
  - PLA has more programming flexibility
  - PAL is cheaper to implement (less switches)
  - PAL has higher speed (more fixed connections)
- PAL circuits often include flip-flops and MUXes after OR gate outputs to provide additional flexibility (Figure A.43)

Complex Programmable Logic Devices (CPLDs)



- Connections between PAL blocks established by programming interconnect switches
- Programming information loaded via JTAG port
- CAD tools often used to program large CPLDs

Programmable vs. Custom Devices

- Programmable devices, such as CPLDs and field-programmable gate arrays (FPGAs) can be configured to implement a *variety* of complex logic circuits
- Specialized devices, such as Application-Specific Integrated Circuits (ASICs) are designed *specifically* for a particular operation, e.g., MPEG decode
- There is a tradeoff between programmability and cost/performance/energy efficiency
- CPLDs/FPGAs are more programmable
  - Faster design times
  - Less development cost
- ASICs are tuned for a specific task
  - Higher performance
  - Better energy efficiency