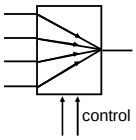
Lecture 9

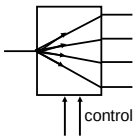
- Demultiplexers
- Programmable Logic Devices
  - Programmable logic array (PLA)
  - Programmable array logic (PAL)

Switching networks logic blocks

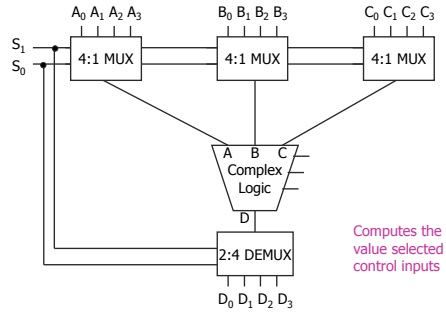
- Multiplexer (MUX)
  - Routes one of many inputs to a single output
  - Also called a *selector*



- Demultiplexer (DEMUX)
  - Routes a single input to one of many outputs
  - Also called a *decoder*



Logic sharing example



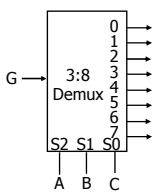
Computes the output value selected by the control inputs

Demultiplexers

- Basic concept
  - Single data input; n control inputs ("selects"); 2^n outputs
  - Single input connects to one of 2^n outputs
  - "Selects" decide which output is connected to the input

Demultiplexers

- The input is called an "enable" (G)



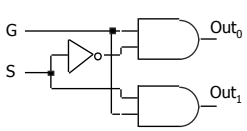
1:2 Decoder:
Out0 = G • S'
Out1 = G • S

2:4 Decoder:
Out0 = G • S1' • S0'
Out1 = G • S1' • S0
Out2 = G • S1 • S0'
Out3 = G • S1 • S0

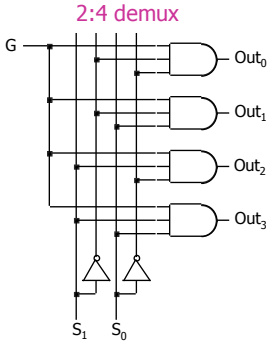
3:8 Decoder:
Out0 = G • S2' • S1' • S0'
Out1 = G • S2' • S1' • S0
Out2 = G • S2' • S1 • S0'
Out3 = G • S2' • S1 • S0
Out4 = G • S2 • S1' • S0'
Out5 = G • S2 • S1' • S0
Out6 = G • S2 • S1 • S0'
Out7 = G • S2 • S1 • S0

Demultiplexers: Implemtation

1:2 demux

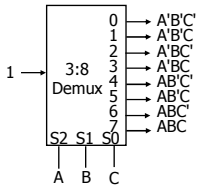


2:4 demux



Demultiplexer as logic block

- A n:2^n demux can implement any function of n variables
  - Use variables as select inputs
  - Tie enable input to logic 1
  - Sum the appropriate minterms (extra OR gate)



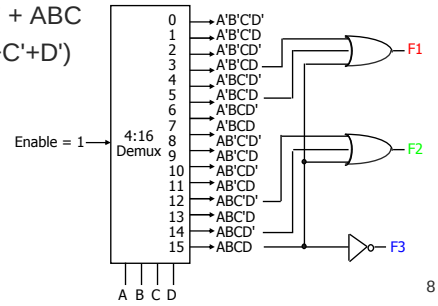
demultiplexer "decodes" appropriate minterms from the control signals

Demultiplexer as logic block

F1 = A'BC'D + A'B'CD + ABCD

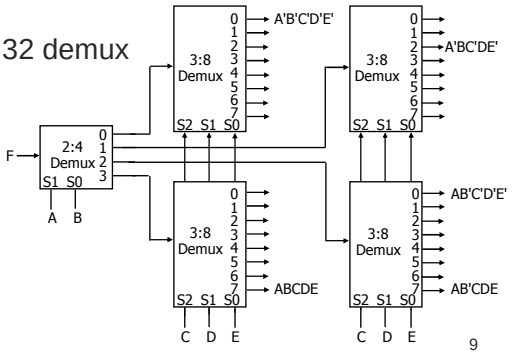
F2 = ABC'D' + ABC

F3 = (A'+B'+C'+D')



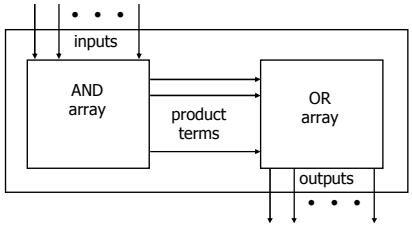
Cascading demultiplexers

- 5:32 demux



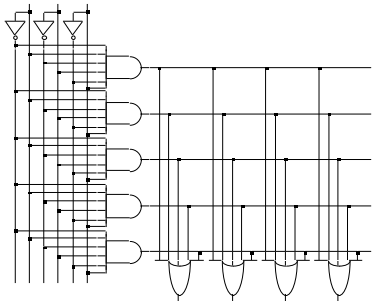
Programmable logic

- Concept: Large array of uncommitted AND/OR gates (actually NAND/NOR gates)
  - You program the array by making or breaking connections
    - Programmable block for sum-of-products logic



All two-level functions available

- You "program" the wire connections



This is a 3-input, 5-term, 4-function programmable logic array (PLA).

Connections ("fuse") are designed to break down under high current.

After analyzing Boolean equations, software determines which fuses should be blown.

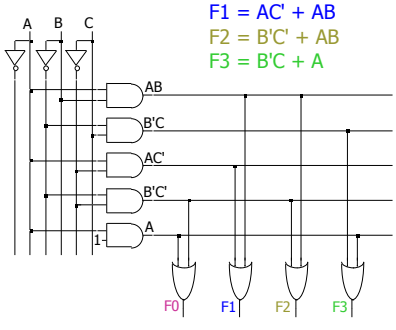
Example

F0 = A + B'C'

F1 = AC' + AB

F2 = B'C' + AB

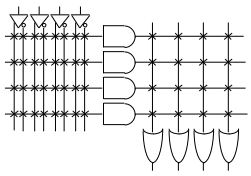
F3 = B'C + A



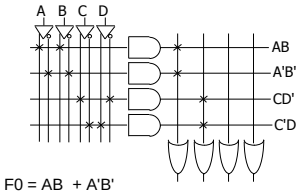
Short-hand notation

- Draw multiple wires as a single wire or bus
- × signifies a connection

Before Programming



After Programming



F0 = AB + A'B'
F1 = CD' + C'D

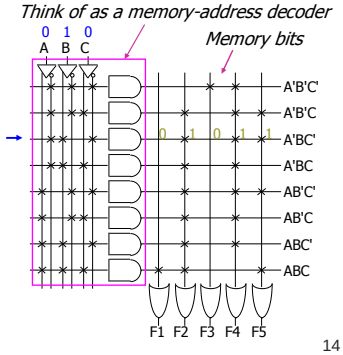
PLA example

F1 = ABC
F2 = A + B + C
F3 = A' B' C'
F4 = A' + B' + C'
F5 = A xor B xor C

A	B	C	F1	F2	F3	F4	F5
0	0	0	0	0	1	1	0
0	0	1	0	1	0	1	1
0	1	0	0	1	0	1	1
0	1	1	0	1	0	1	0
1	0	0	0	1	0	1	1
1	0	1	0	1	0	1	0
1	1	0	0	1	0	1	0
1	1	1	1	1	0	0	1

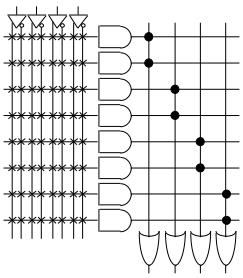
Think of as a memory-address decoder

Memory bits



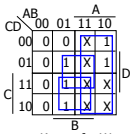
PLAs versus PALs

- We've been looking at PLAs
  - Fully programmable AND/OR arrays
- Programmable array logic (PAL)
  - Programmable AND array
  - OR array is prewired
    - Cheaper and faster than PLAs

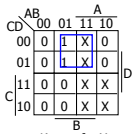


Example: BCD to Gray code

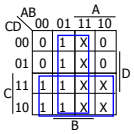
A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X



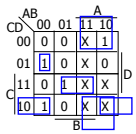
K-map for W



K-map for X



K-map for Y

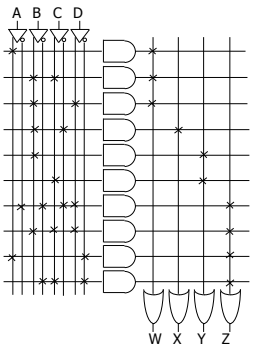


K-map for Z

Wiring a PLA

Minimized functions:

W = A + BC + BD
X = BC'
Y = B + C
Z = A'B'C'D + BCD + AD' + B'CD'



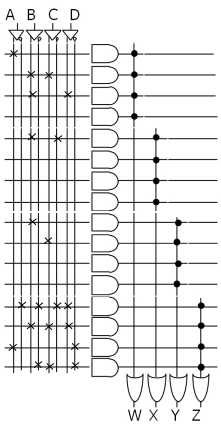
Wiring a PAL

Minimized functions:

W = A + BC + BD
X = BC'
Y = B + C
Z = A'B'C'D + BCD + AD' + B'CD'

Fine example for the use of PAL (because no shared AND terms)

Many AND gates wasted, but still faster and cheaper than PLA



Implementation comparison

PLA:

- No shared logic terms in this example
- 10 decoded functions (10 AND gates)

- Example is a good candidate for PALs:
  - 10 of 16 possible inputs are decoded
  - No sharing among AND terms

PAL:

- Z requires 4 product terms
- Need a PAL that handles 4 product terms for each output
- 16 decoded functions (16 AND gates)
- 6 unused AND gates