Lecture 9

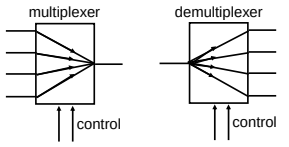
- Logistics
  - HW3 due Wednesday
  - Lab4 going on this week
  - Midterm 1 Friday, Review Session Thursday 4:30, place TBA
    - material up to end of last lecture
- Last lecture
  - K-map design examples
  - Multiplexers
- Today
  - Demultiplexers
  - Programmable Logic Devices
    - PLAs
    - PALs

The "WHY" slide

- Demultiplexers
  - Natural complement to multiplexers
- Programmable Logic Devices (PLDs)
  - Often you want to have a look up table of functions stored away somewhere in your device. Rather than having specific circuits build every time, it would be nice to have a "general-purpose" structure that could be "programmed" for a specific usage. PLDs have a generic structure that allows any function to be expressed and stored.
  - And it is nice if it is reprogrammable. Some PLDs are reprogrammable (like your memory sticks).

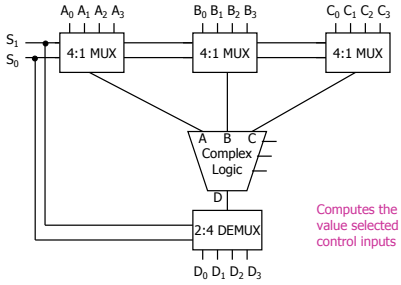
Switching-network logic blocks

- 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*



We construct these devices from:
- logic gates
- networks of transistor switches

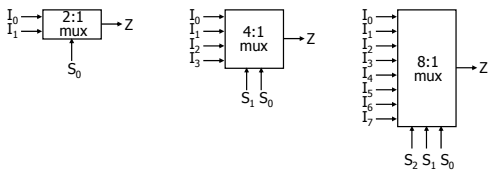
Logic sharing with MUX/DEMUX



Computes the output value selected by the control inputs

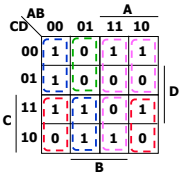
Multiplexers

- 2:1 mux: $Z = S_0'In_0 + S_0In_1$
- 4:1 mux: $Z = S_1'S_0'In_0 + S_1'S_0In_1 + S_1S_0'In_2 + S_1S_0In_3$
- 8:1 mux: $Z = S_2'S_1'S_0'In_0 + S_2'S_1'S_0In_1 + \ldots$



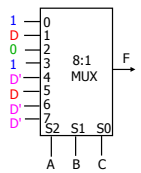
Multiplexers as general-purpose logic

- Implementing a 2^{n-1}:1 mux as a function of n variables
  - (n-1) mux control variables $S_0 - S_{n-2}$
  - One data variable S_{n-1}
  - Four possible values for each data input: 0, 1, S_{n-1}, S_{n-1}'
  - Example: F(A,B,C,D) implemented using an 8:1 mux



Choose A,B,C as control variables

Choose D as a data variable



Demultiplexers (DEMUX)

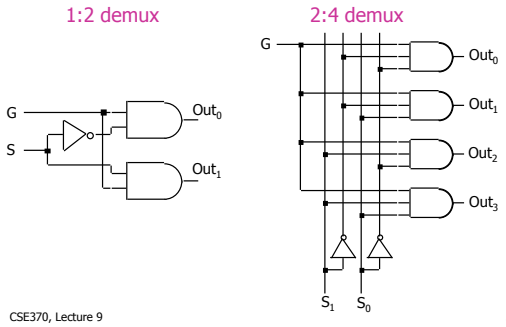
- Basic concept
  - Single data input; n control inputs ("selects"); 2^n outputs
  - Single input connects to one of 2^n outputs
  - "Selects" decide which output is connected to the input
  - When used as a decoder, the input is called an "enable" (G)

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

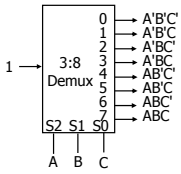
3:8 Decoder:
Out0 = G • S2' • S1' • S0'
Out1 = G • S2' • S1' • S0
Out2 = G • S2' • S1 • S0'
Out3 = G • S2' • S1 • S0
Out4 = G • S2 • S1' • S0'
Out5 = G • S2 • S1' • S0
Out6 = G • S2 • S1 • S0'
Out7 = G • S2 • S1 • S0

Logic-gate implementation of demultiplexers



Demultiplexers as general-purpose logic

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

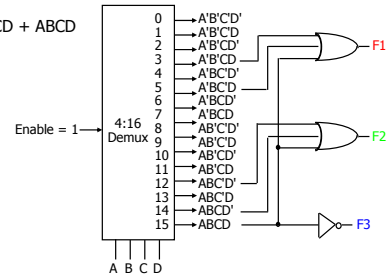


demultiplexer "decodes" appropriate minterms from the control signals

Demultiplexers as general-purpose logic

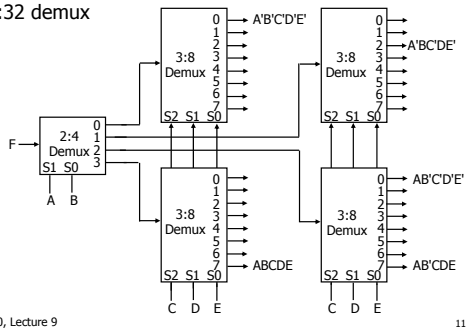
Example

F1 = A'BC'D + A'B'CD + ABCD
F2 = ABC'D' + ABC
F3 = (A'+B'+C'+D')



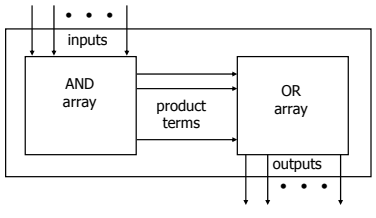
Cascading demultiplexers

- 5:32 demux



Programmable logic (PLAs & PALs)

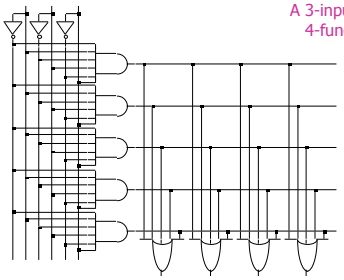
- Concept: Large array of uncommitted AND/OR gates
  - Actually NAND/NOR gates
  - You program the array by making or breaking connections
    - Programmable block for sum-of-products logic



All two-level logic functions are available

- You "program" the wire connections

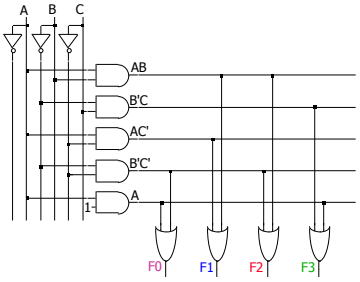
A 3-input, 5-term, 4-function PLA



Programming the wire connections

- Fuse: Comes connected; break unwanted connections
- Anti-fuse: Comes disconnected; make wanted connections

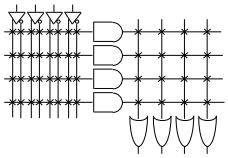
F0 = A + B'C'
F1 = AC' + AB
F2 = B'C' + AB
F3 = B'C + A



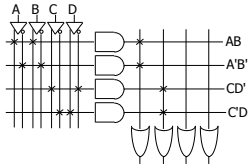
Short-hand notation

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

Before Programming



After Programming



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

F0 F1

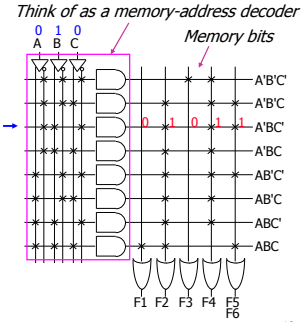
PLA example

F1 = ABC
F2 = A + B + C
F3 = A' B' C'
F4 = A' + B' + C'
F5 = A xor B xor C
F6 = A xor B xor C

A	B	C	F1	F2	F3	F4	F5	F6
0	0	0	0	0	1	1	0	0
0	0	1	0	1	0	1	1	1
0	1	0	0	1	0	1	1	1
0	1	1	0	1	0	1	0	0
1	0	0	0	1	0	1	1	1
1	0	1	0	1	0	1	0	0
1	1	0	0	1	0	1	0	0
1	1	1	1	1	0	0	1	1

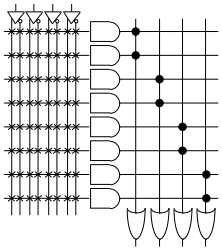
Think of as a memory-address decoder

Memory bits



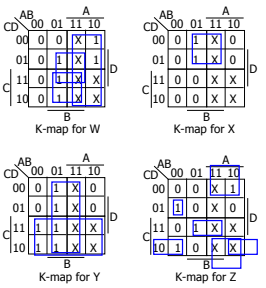
PLAs versus PALs

- 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 converter

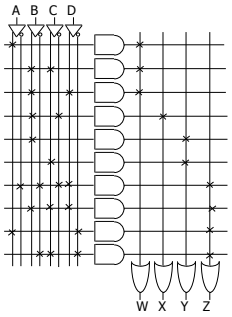
A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	1	1	1	0
0	1	1	0	1	0	1	0
0	1	1	1	1	0	1	1
1	0	0	0	1	0	0	1
1	0	0	1	1	0	0	0
1	0	1	0	X	X	X	X
1	0	1	1	X	X	X	X
1	1	0	0	X	X	X	X
1	1	0	1	X	X	X	X
1	1	1	0	X	X	X	X
1	1	1	1	X	X	X	X



Example: BCD to Gray --- Wire a PLA

Minimized functions:

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



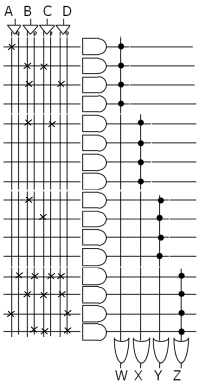
Example: Wire a PAL

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



Compare implementations for this example

- PLA:
  - No shared logic terms in this example
  - 10 decoded functions (10 AND gates)
- 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
- This decoder is a good candidate for PALs
  - 10 of 16 possible inputs are decoded
  - No sharing among AND terms
- Next time an alternative
  - Read-only memory (ROM)

Midterm 1 Topics Covered

- Combinational logic basics
  - Binary/hex/decimal numbers
  - Ones and twos complement arithmetic
  - Truth tables
  - Boolean algebra
  - Basic logic gates
  - Schematic diagrams
  - de Morgan's theorem
  - AND/OR to NAND/NOR logic conversion
  - K-maps (up to 4 variables), logic minimization, don't cares
  - SOP, POS
  - Minterm and maxterm expansions (canonical, minimized)

Midterm 1 Topics Covered (continued)

- Combinational logic applications
  - Combinational design
    - Input/output encoding
    - Truth table
    - K-map
    - Boolean equations
    - Schematics
  - Multiplexers