Lecture 8

- Logistics
  - HW2 due Friday
  - Shorter HW3 posted--- due Monday (materials covered in midterm1) sol'n out on Monday, late assignment not accepted
  - Verilog tutorial (overview + handout): provided with Lab 4
  - Schedule shift: adjustment on the web
  - Midterm 1: Wednesday in class --- materials up to Lecture 9
  - Review session on Tuesday? Time?
- Last lecture
  - K-maps, don't cares, POS K-maps
- Today
  - K-maps 5, 6 dimensions
  - "Switching-network" logic blocks (multiplexers/demultiplexers)
  - Programmable logic devices (PLDs) --- sneak peak

Karnaugh Maps, 6 dimensions

K-maps become 3D for 5 & 6 variables

AB = 00

EF \ CD	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	0	0	1	1
10	0	0	0	0

AB = 01

EF \ CD	00	01	11	10
00	1	0	0	0
01	0	0	1	1
11	1	0	1	1
10	1	0	0	0

AB = 11

EF \ CD	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	1	0	1	1
10	1	0	0	0

AB = 10

EF \ CD	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	0	0	1	1
10	0	0	0	0

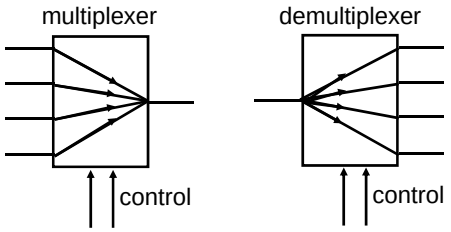
OUTPUT =

A'BC'D'F' +

CF + BC'D'E

Switching-network logic blocks

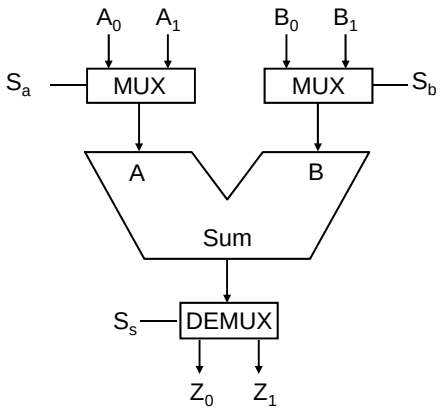
- Multiplexer (MUX)
  - Routes one of many inputs to a single output
  - Also called a *selector*
- Demultiplexer (DEMUX)
  - Routes a single input to one of many outputs
  - Also called a *decoder*



We construct these devices from:
(1) logic gates
(2) networks of transistor switches

Rationale: Sharing complex logic functions

- Share an adder: Select inputs; route sum



multiple inputs

single adder

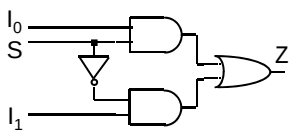
multiple output destinations

Multiplexers

- Basic concept
  - 2^n data inputs; n control inputs ("selects"); 1 output
  - Connects one of 2^n inputs to the output
  - "Selects" decide which input connects to output
  - Two alternative truth-tables: Functional and Logical

Example: A 2:1 Mux

$Z = SIn_1 + S'In_0$



Functional truth table

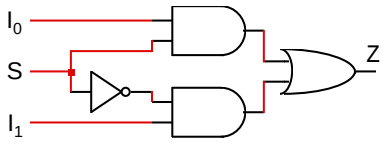
S	Z
0	In_0
1	In_1

Logical truth table

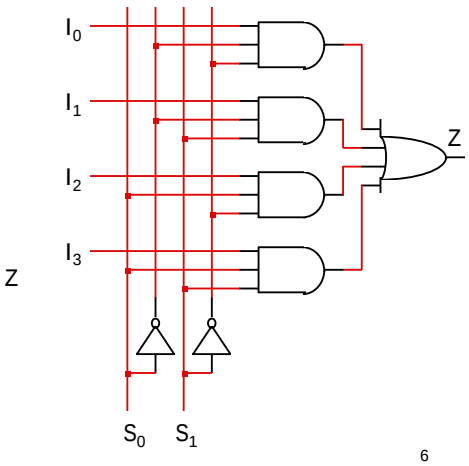
In_1	In_0	S	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Logic-gate implementation of multiplexers

2:1 mux

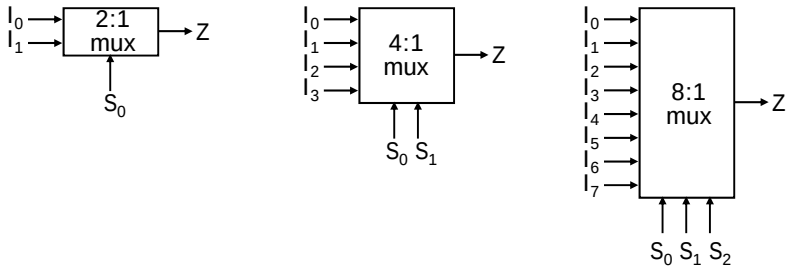


4:1 mux



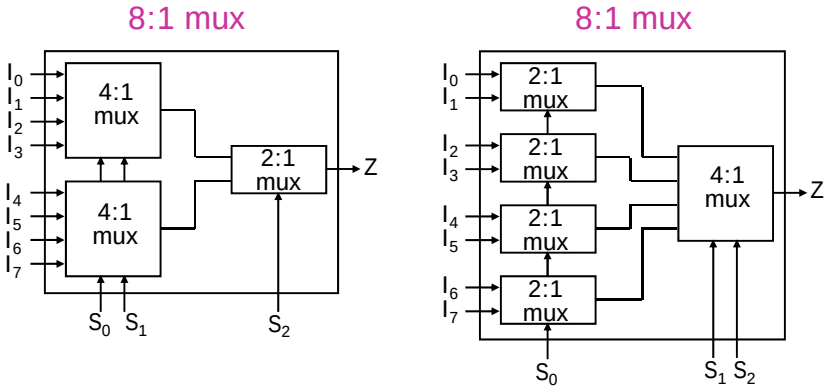
Multiplexers (con't)

- 2:1 mux: $Z = S'In_0 + SIn_1$
- 4:1 mux: $Z = S_0'S_1'In_0 + S_0'S_1In_1 + S_0S_1'In_2 + S_0S_1In_3$
- 8:1 mux: $Z = S_0'S_1'S_2'In_0 + S_0'S_1S_2In_1 + \ldots$



Cascading multiplexers

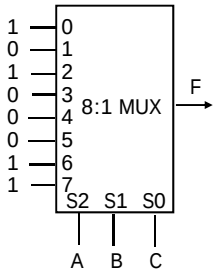
- Can form large multiplexers from smaller ones
  - Many implementation options

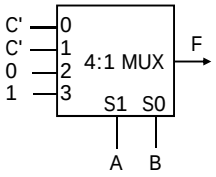


Multiplexers as general-purpose logic

- A 2^n:1 mux can implement any function of n variables
  - A lookup table
  - A 2^{n-1}:1 mux also can implement any function of n variables
- Example: F(A,B,C) = m0 + m2 + m6 + m7
  = A'B'C' + A'BC' + ABC' + ABC
  = A'B'(C') + A'B(C') + AB(0) + AB(1)

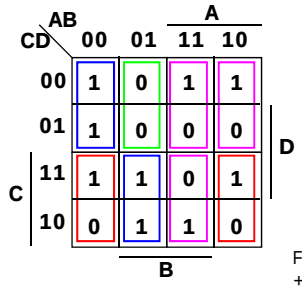
A	B	C	F	
0	0	0	1	C'
0	0	1	0	
0	1	0	1	C'
0	1	1	0	
1	0	0	0	0
1	0	1	0	
1	1	0	1	1
1	1	1	1	





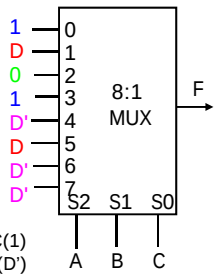
Multiplexers as general-purpose logic

- Implementing a 2^{n-1}:1 mux as a function of n variables
  - (n-1) mux control variables S_0 – S_{n-1}
  - One data variable S_n
  - Four possible values for each data input: 0, 1, S_n, S_n'
  - 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



F=A'B'C'(1)+A'B'C(D)+A'BC'(0)+A'BC(1)
+AB'C'(D')+AB'C(D)+ABC'(D')+ABC(D')

F = A'B'C'+A'CD+B'CD+AC'D'+BCD'

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)

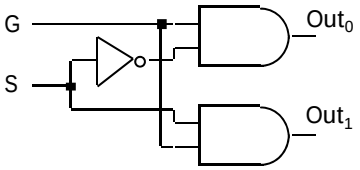
1:2 Decoder:
Out0 = G • S'
Out1 = G • S

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

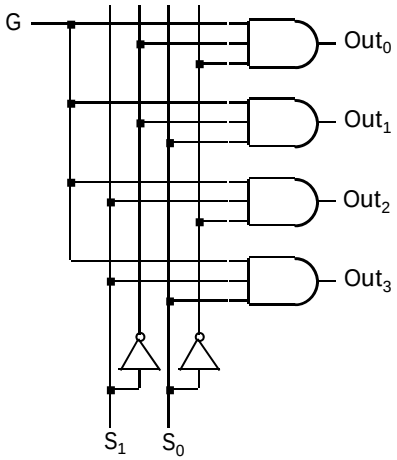
3:8 Decoder:
Out0 = G • S2' • S1' • S0'
Out1 = G • S2' • S1' • S0
Out2 = G • S2' • S1 • S0'
Out3 = G • S2' • S1 • S0
Out4 = G • S2 • S1' • S0'
Out5 = G • S2 • S1' • S0
Out6 = G • S2 • S1 • S0'
Out7 = G • S2 • S1 • S0

Logic-gate implementation of demultiplexers

1:2 demux

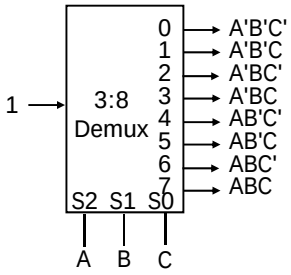


2:4 demux



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

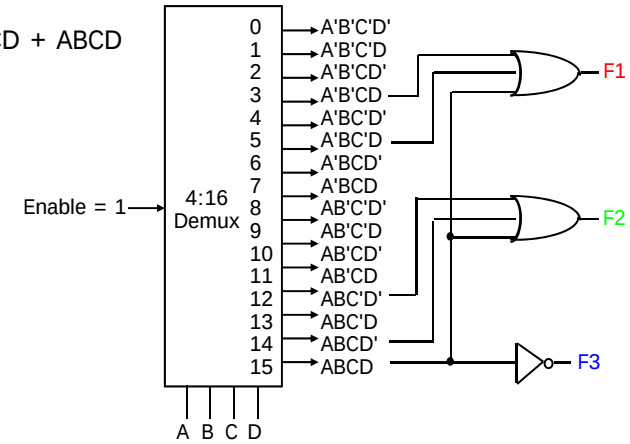
Demultiplexers as general-purpose logic

Example

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

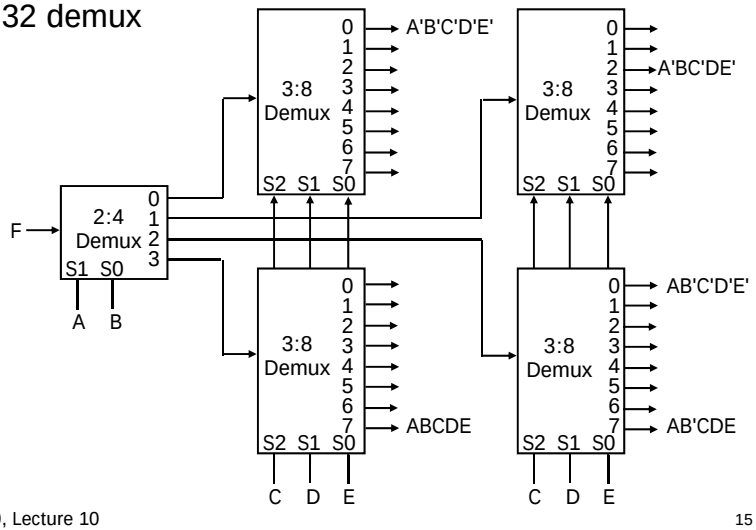
F2 = ABC'D' + ABC

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



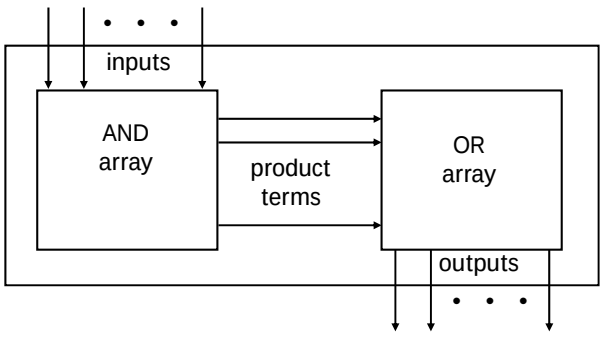
Cascading demultiplexers

- 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