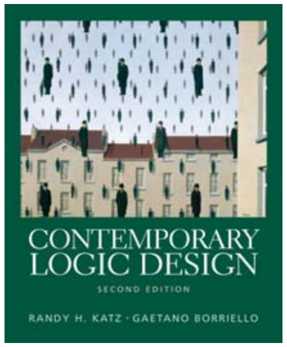
CSE 370 Spring 2006 Introduction to Digital Design

Lecture 10: Multiplexers and Demultiplexers



Last Lecture
- Multilevel Logic
- Hazards

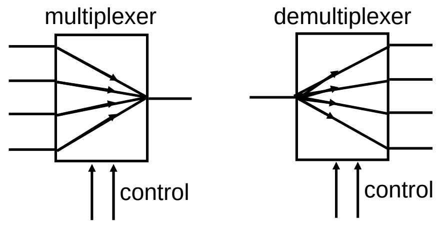
Today
- Multiplexers
- Demultiplexers

Administrivia

- This week: HW #4, Lab 4

Switching-network logic blocks

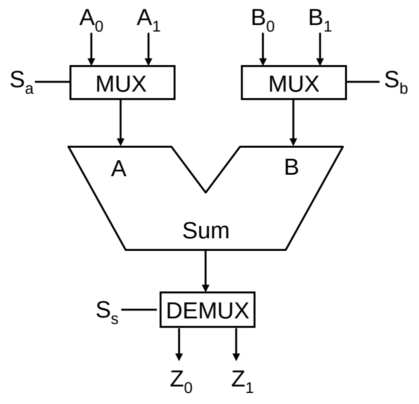
- Multiplexer
  - Routes one of many inputs to a single output
  - Also called a *selector*
- Demultiplexer
  - Routes a single input to one of many outputs
  - Also called a *decoder*



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

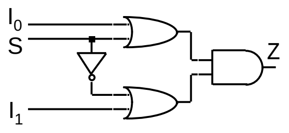
multiple output destinations

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

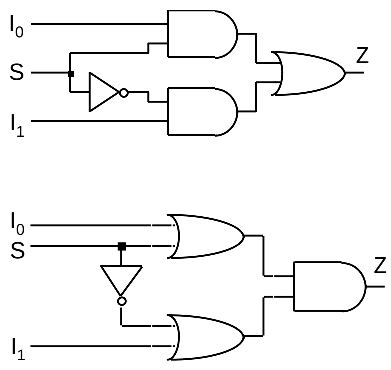
S	Z
0	In_0
1	In_1

Logical truth table

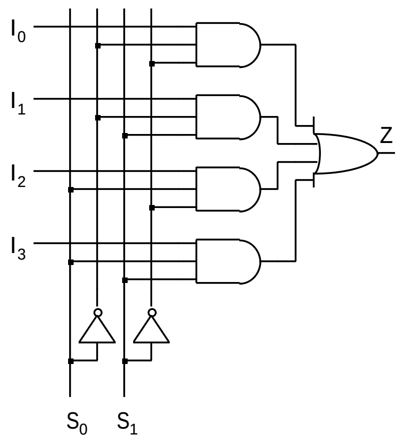
In_1	In_0	S	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Logic-gate implementation of multiplexers

2:1 mux

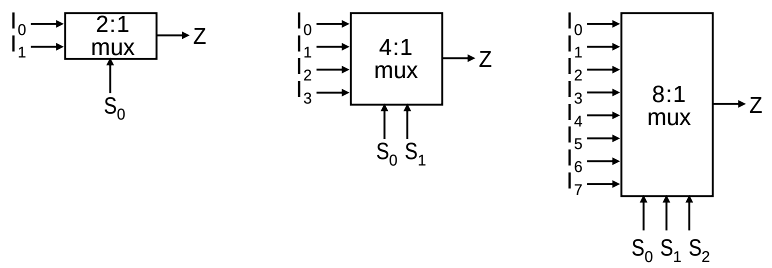


4:1 mux



Multiplexers (con't)

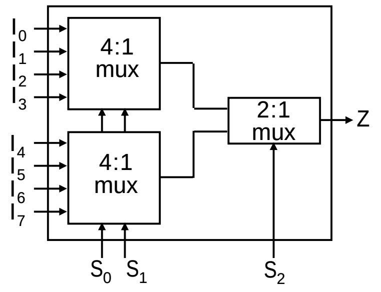
- 2:1 mux: $Z = S'In_0 + SIn_1$
- 4:1 mux: $Z = S_0'S_1'In_0 + S_0'S_1In_1 + S_0S_1'In_2 + S_0S_1In_3$
- 8:1 mux: $Z = S_0'S_1'S_2'In_0 + S_0'S_1S_2In_1 + ...$



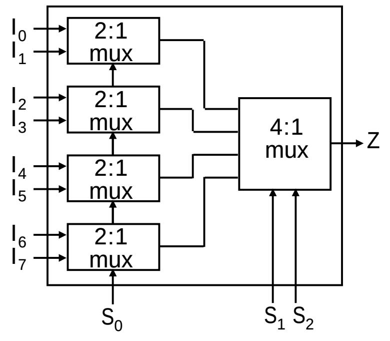
Cascading multiplexers

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

8:1 mux



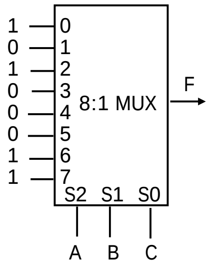
8:1 mux

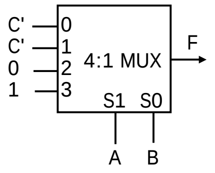


Multiplexers as general-purpose logic

- 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)

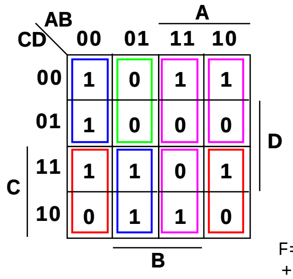
A	B	C	F	
0	0	0	1	C'
0	0	1	0	
0	1	0	1	C'
0	1	1	0	
1	0	0	0	0
1	0	1	0	
1	1	0	1	1
1	1	1	1	





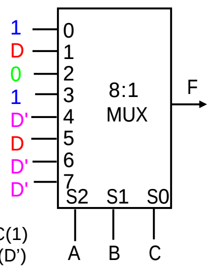
Multiplexers as general-purpose logic

- Implementing a 2^n:1 mux as a function of n–1 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



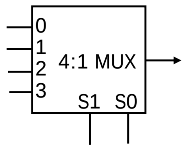
F=A'B'C'(1)+A'B'C(D)+A'BC'(0)+A'BC(1)
+AB'C'(D')+AB'C(D)+ABC'(D')+ABC(D')

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

Exercise

- Implementing the following function as a 4:1 mux

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



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
  - When used as a decoder, the input is called an “enable” (G)

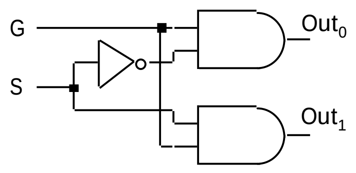
1:2 Decoder:
Out0 = G • S'
Out1 = G • S

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

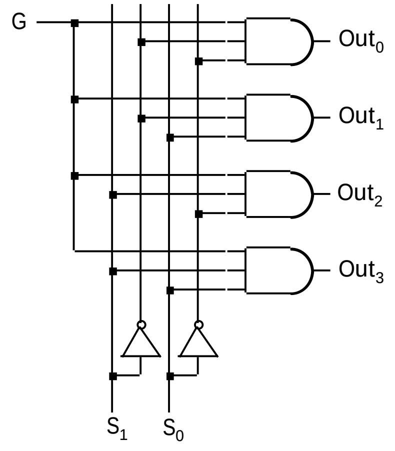
3:8 Decoder:
Out0 = G • S2' • S1' • S0'
Out1 = G • S2' • S1' • S0
Out2 = G • S2' • S1 • S0'
Out3 = G • S2' • S1 • S0
Out4 = G • S2 • S1' • S0'
Out5 = G • S2 • S1' • S0
Out6 = G • S2 • S1 • S0'
Out7 = G • S2 • S1 • S0

Logic-gate implementation of demultiplexers

1:2 demux

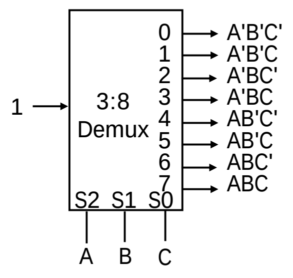


2:4 demux



Demultiplexers as general-purpose logic

- A n:2^n demux can implement any function of n variables
  - 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

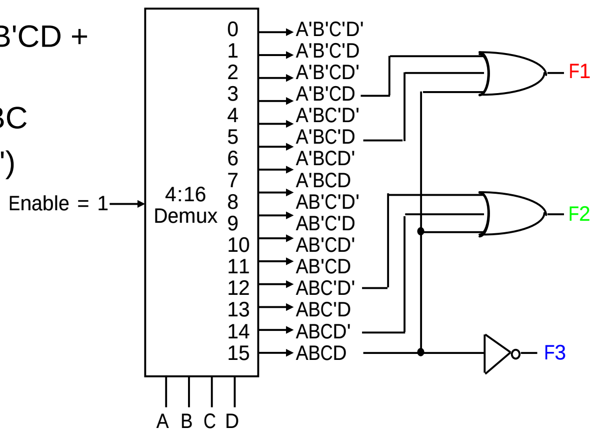
Demultiplexers as general-purpose logic

Example

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

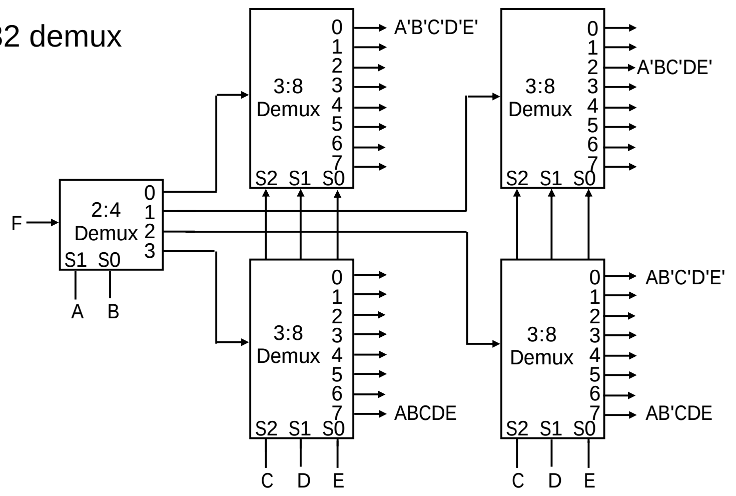
F2 = ABC'D' + ABC

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



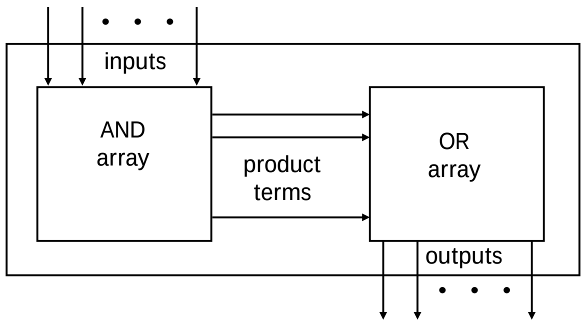
Cascading demultiplexers

- 5:32 demux



Programmable logic (PLAs & PALs)

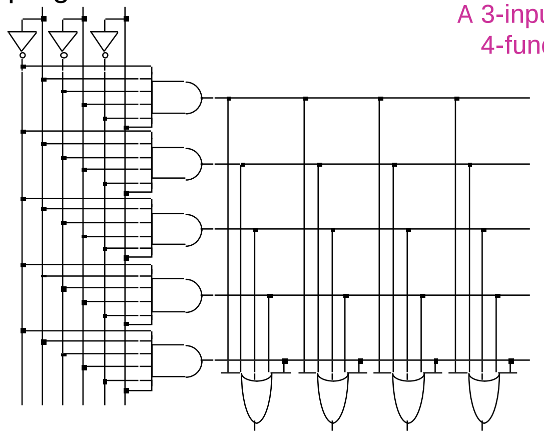
- Concept: Large array of uncommitted AND/OR gates
  - Actually NAND/NOR gates
  - You program the array by making or breaking connections
    - Programmable block for sum-of-products logic



All two-level logic functions are available

- You "program" the wire connections

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



Sharing product terms

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

inputs
1 = asserted in term
0 = negated in term
– = does not participate

outputs
1 = term connected to output
0 = no connection to output

- Personality matrix:

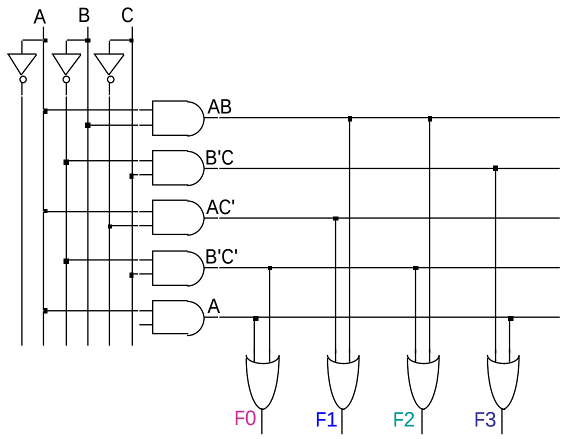
product term	inputs A	B	C	outputs F0	F1	F2	F3
AB	1	1	–	0	1	1	0
B'C	–	0	1	0	0	0	1
AC'	1	–	0	0	1	0	0
B'C'	–	0	0	1	0	1	0
A	1	–	–	1	0	0	1

Reuse terms

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



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