Implementation Technologies

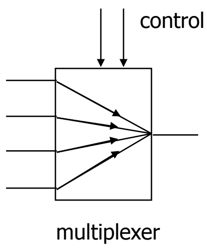
- Standard gates (pretty much done)
  - gate packages
  - cell libraries
- Regular logic (we are here)
  - multiplexers
  - decoders
- Two-level programmable logic (a little later)
  - PALs, PLAs, PLDs
  - ROMs
  - FPGAs

Regular logic

- Need to make design faster
- Need to make engineering changes easier to make
- Simpler for designers to understand and map to functionality
  - harder to think in terms of specific gates
  - easier to think in terms of larger multi-purpose blocks

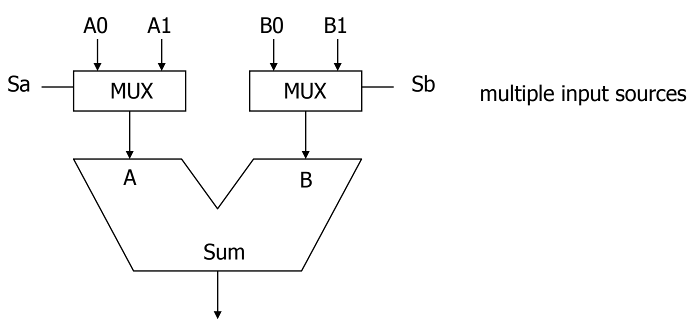
Making connections

- Direct point-to-point connections using wires
- Route one of many inputs to a single output --- multiplexer



Muxes (cont'd)

- Uses of multiplexers in multi-point connections



Multiplexers/selectors

- Multiplexers/selectors: general concept
  - 2^n data inputs, n control inputs (called "selects"), 1 output
  - used to connect 2^n points to a single point
  - control signal pattern forms binary index of input connected to output

$Z = A' I_0 + A I_1$

A	Z
0	I_0
1	I_1

I_1	I_0	A	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

functional form

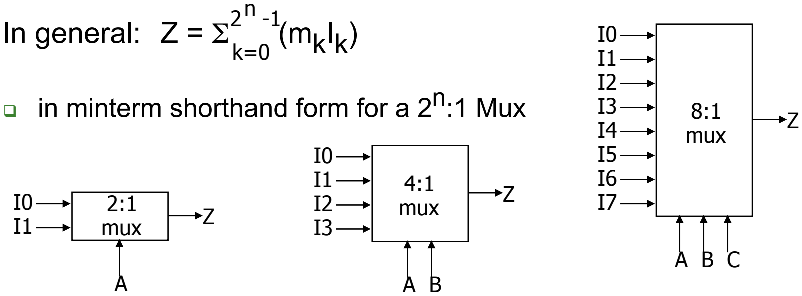
logical form

two alternative forms
for a 2:1 Mux truth table

Multiplexers/selectors (cont'd)

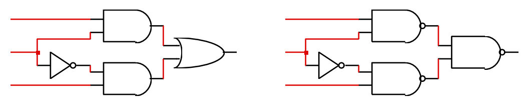
- 2:1 mux: $Z = A'I_0 + AI_1$
- 4:1 mux: $Z = A'B'I_0 + A'BI_1 + AB'I_2 + ABI_3$
- 8:1 mux: $Z = A'B'C'I_0 + A'B'CI_1 + A'BC'I_2 + A'BCI_3 + AB'C'I_4 + AB'CI_5 + ABC'I_6 + ABCI_7$

- In general: $Z = \Sigma_{k=0}^{2^n-1}(m_k I_k)$
  - in minterm shorthand form for a 2^n:1 Mux

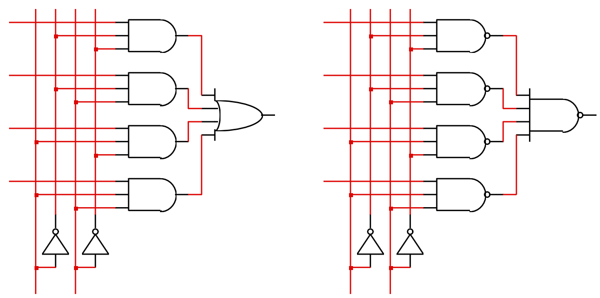


Gate level implementation of muxes

- 2:1 mux

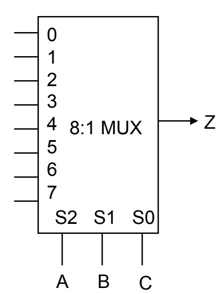


- 4:1 mux



Multiplexers as general-purpose logic

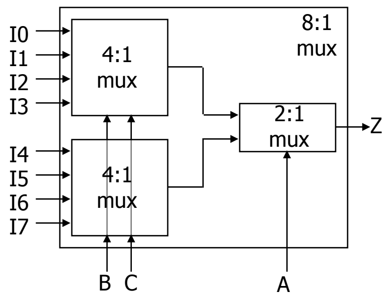
- A 2^n:1 multiplexer can implement any function of n variables
  - with the variables used as control inputs and
  - the data inputs tied to 0 or 1
  - in essence, a lookup table (LUT)
- Example:
  - F(A,B,C) = m0 + m2 + m6 + m7
    = A'B'C' + A'BC' + ABC' + ABC



$$Z = A'B'C'I_0 + A'B'CI_1 + A'BC'I_2 + A'BCI_3 + AB'C'I_4 + AB'CI_5 + ABC'I_6 + ABCI_7$$

Cascading multiplexers

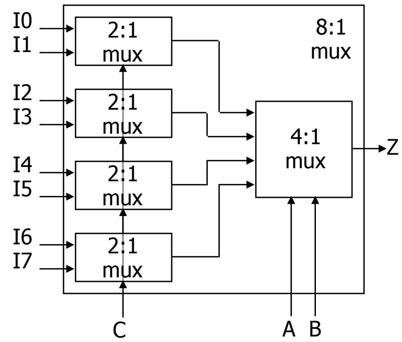
- Large multiplexers can be made by cascading smaller ones



control signals B and C simultaneously choose one of I0, I1, I2, I3 and one of I4, I5, I6, I7

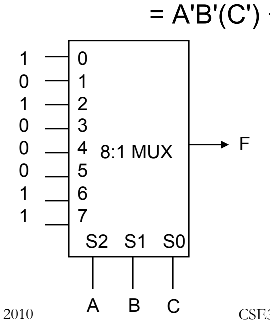
control signal A chooses which of the upper or lower mux's output to gate to Z

alternative implementation

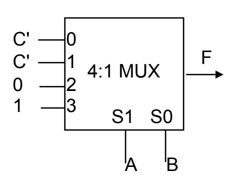


Multiplexers as general-purpose logic (cont'd)

- A 2^{n-1}:1 multiplexer can implement any function of n variables
  - with n-1 variables used as control inputs and
  - the data inputs tied to the last variable or its complement
- 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)

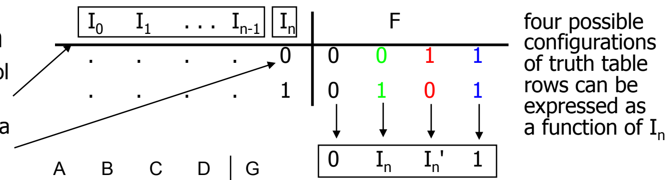


A	B	C	F	
0	0	0	1	C'
0	0	1	0	
0	1	0	1	C'
0	1	1	0	
1	0	0	0	0
1	0	1	0	
1	1	0	1	1
1	1	1	1	



Multiplexers as general-purpose logic (cont'd)

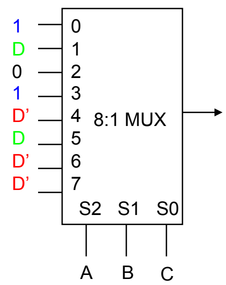
- Generalization



- Example: G(A,B,C,D) can be realized by an 8:1 MUX

A	B	C	D	G	
0	0	0	0	1	1
0	0	0	1	1	
0	0	1	0	0	D
0	0	1	1	1	
0	1	0	0	0	0
0	1	0	1	0	
0	1	1	0	1	1
0	1	1	1	1	
1	0	0	0	1	D'
1	0	0	1	0	
1	0	1	0	0	D
1	0	1	1	1	
1	1	0	0	1	D'
1	1	0	1	0	
1	1	1	0	1	D'
1	1	1	1	0	

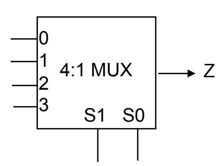
choose A,B,C as control variables



Activity

- Realize F = B'CD' + ABC' with a 4:1 multiplexer and a minimum of other gates:

A	B	C	D	Z
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0



Decoders/demultiplexers

- Decoders/demultiplexers: general concept
  - single data input, n control inputs, 2^n outputs
  - control inputs (called "selects" (S)) represent binary index of output to which the input is connected
  - data input usually called "enable" (G)

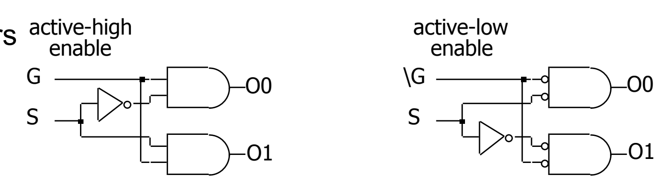
1:2 Decoder:
O0 = G • S′
O1 = G • S

2:4 Decoder:
O0 = G • S1′ • S0′
O1 = G • S1′ • S0
O2 = G • S1 • S0′
O3 = G • S1 • S0

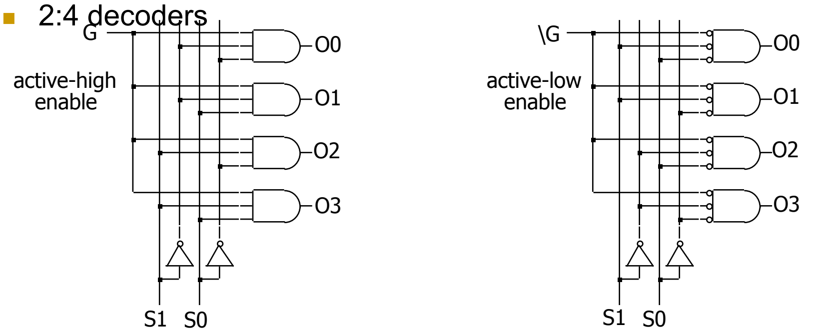
3:8 Decoder:
O0 = G • S2′ • S1′ • S0′
O1 = G • S2′ • S1′ • S0
O2 = G • S2′ • S1 • S0′
O3 = G • S2′ • S1 • S0
O4 = G • S2 • S1′ • S0′
O5 = G • S2 • S1′ • S0
O6 = G • S2 • S1 • S0′
O7 = G • S2 • S1 • S0

Gate level implementation of demultiplexers

- 1:2 decoders

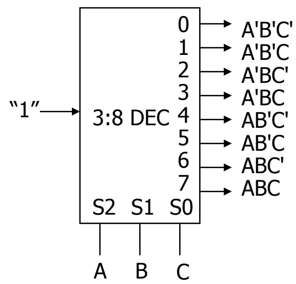


- 2:4 decoders



Decoders as general-purpose logic

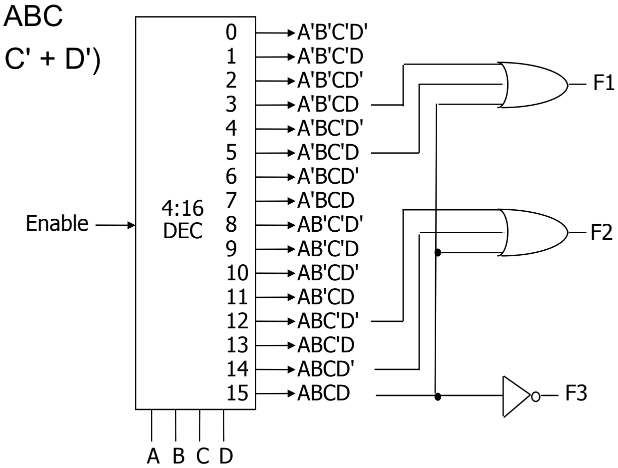
- A n:2^n decoder can implement any function of n variables
  - with the variables used as control inputs
  - the enable inputs tied to 1 and
  - the appropriate minterms summed to form the function



decoder generates appropriate minterm based on control signals (it "decodes" control signals)

Decoders as general-purpose logic (cont'd)

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



Cascading decoders

- 5:32 decoder
  - 1x2:4 decoder
  - 4x3:8 decoders