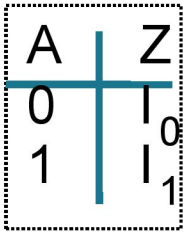
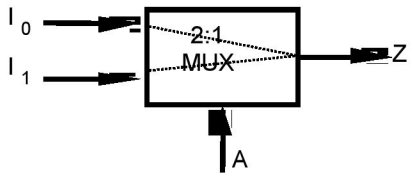
Multiplexer

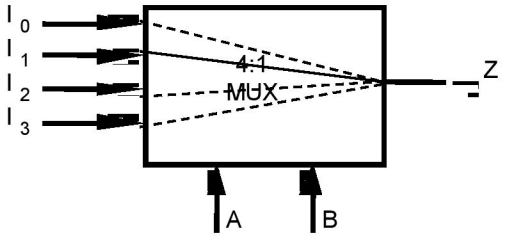
MUX

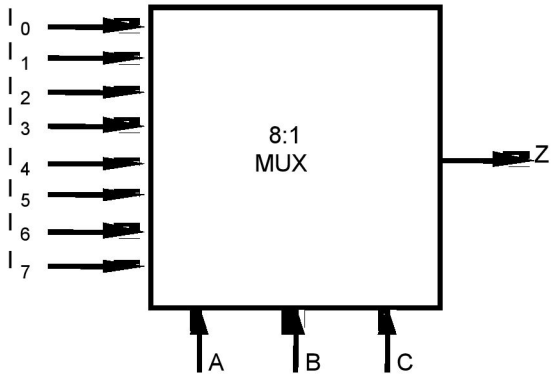
Multiplexer

- **Multiplexer (Selector)**
  - 2^n data inputs
  - n control inputs
  - 1 output
  - Described as 2^n:1
  - Is used to connect only one of 2^n inputs to a single output at a given time
  - The control signal pattern forms the binary index of the input connected to the output
  - Called “MUX” for brevity

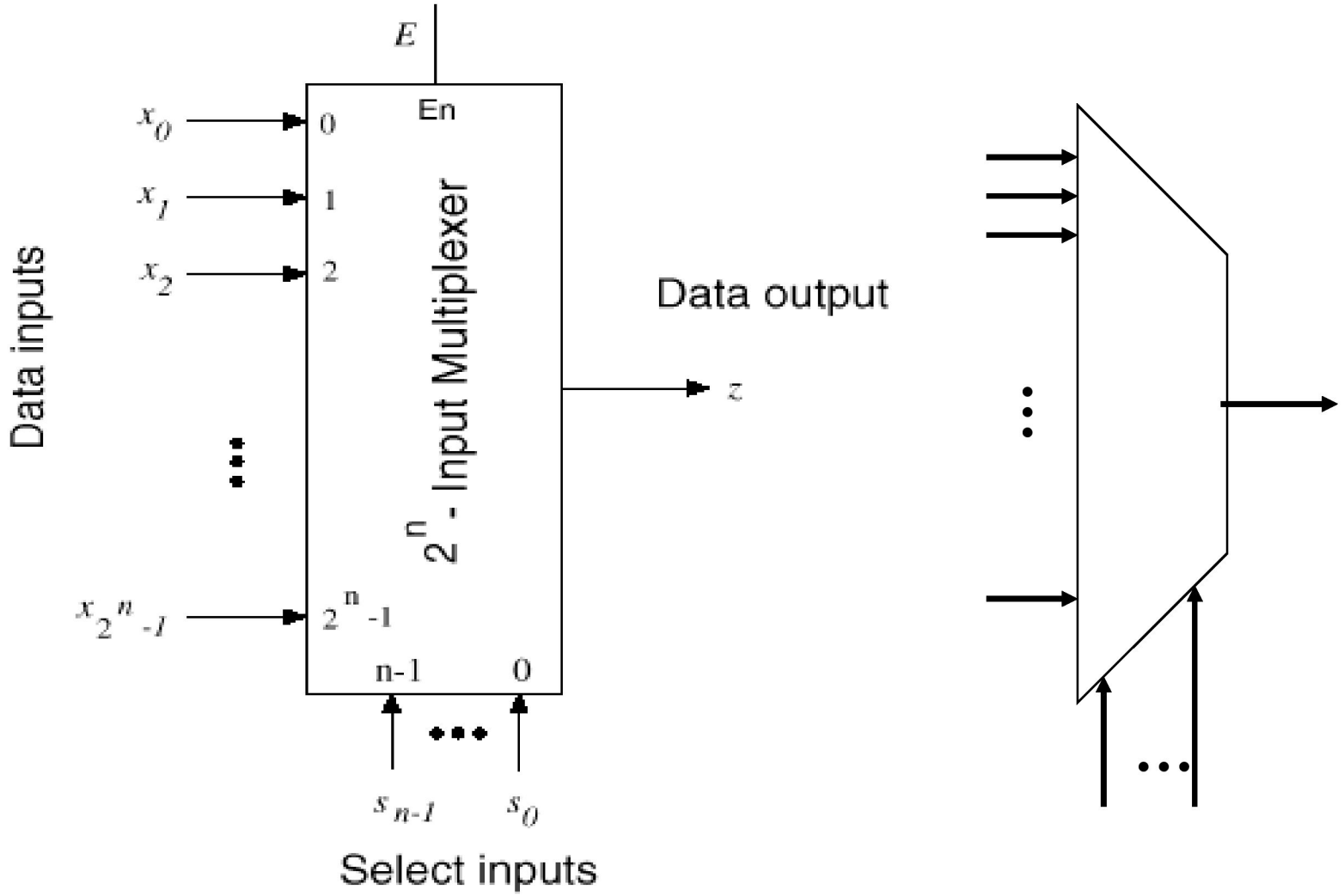




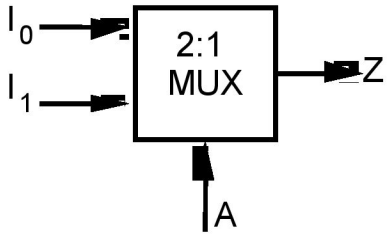




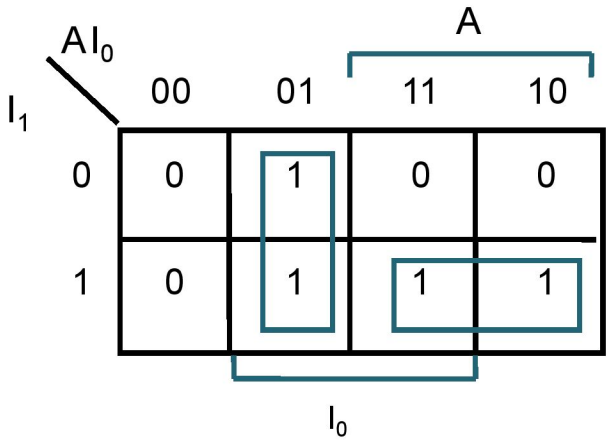
Multiplexer

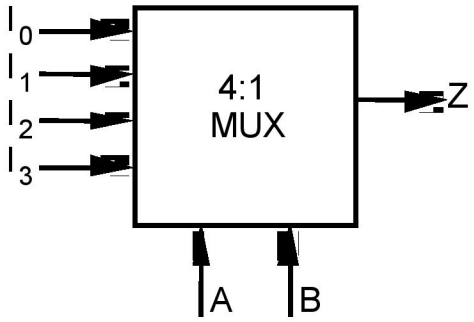


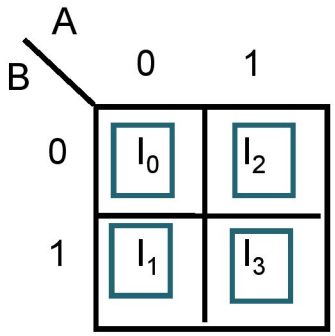
MUX Boolean Functions



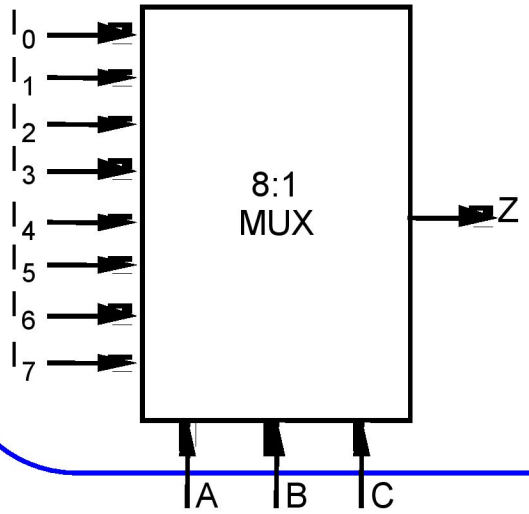
$$Z = A' I_0 + A I_1$$







$$Z = A' B' I_0 + A' B I_1 + A B' I_2 + A B I_3$$



$$Z = A' B' C' I_0 + A' B' C I_1 + A' B C' I_2 + A' B C I_3 + A B' C' I_4 + A B' C I_5 + A B C' I_6 + A B C I_7$$

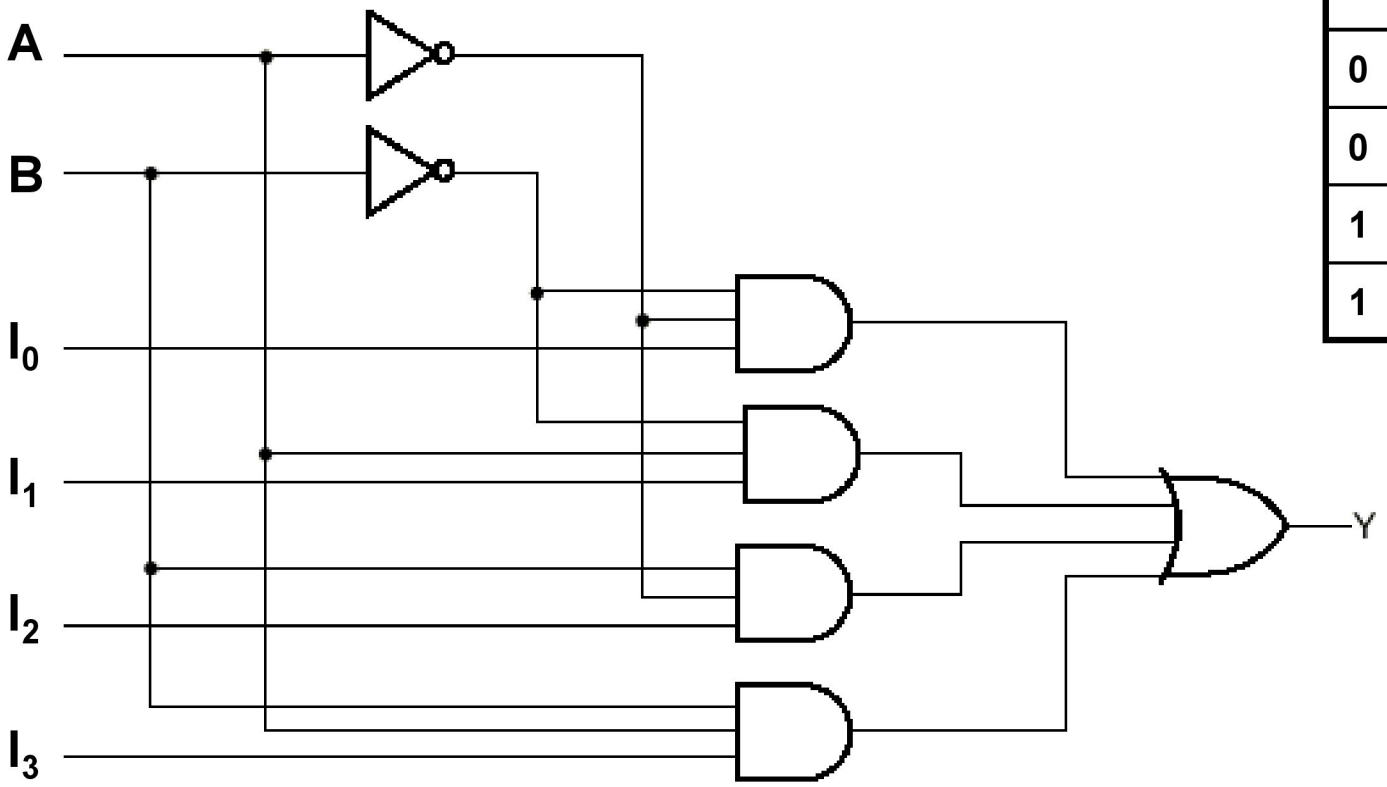
In general: $Z = \sum_{k=0}^{2^n - 1} m_k I_k$, in minterm form

Circuit Diagram

- **4:1 MUX**

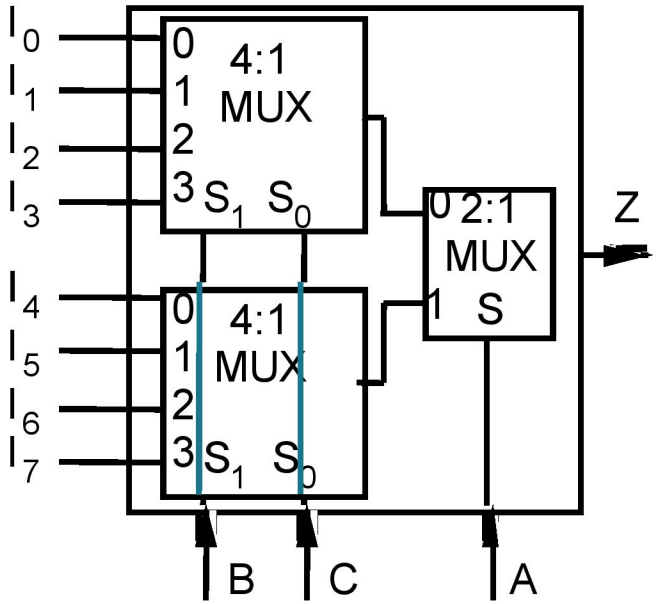
$$Y = A' B' I_0 + A' B I_1 + A B' I_2 + A B I_3$$

A	B	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

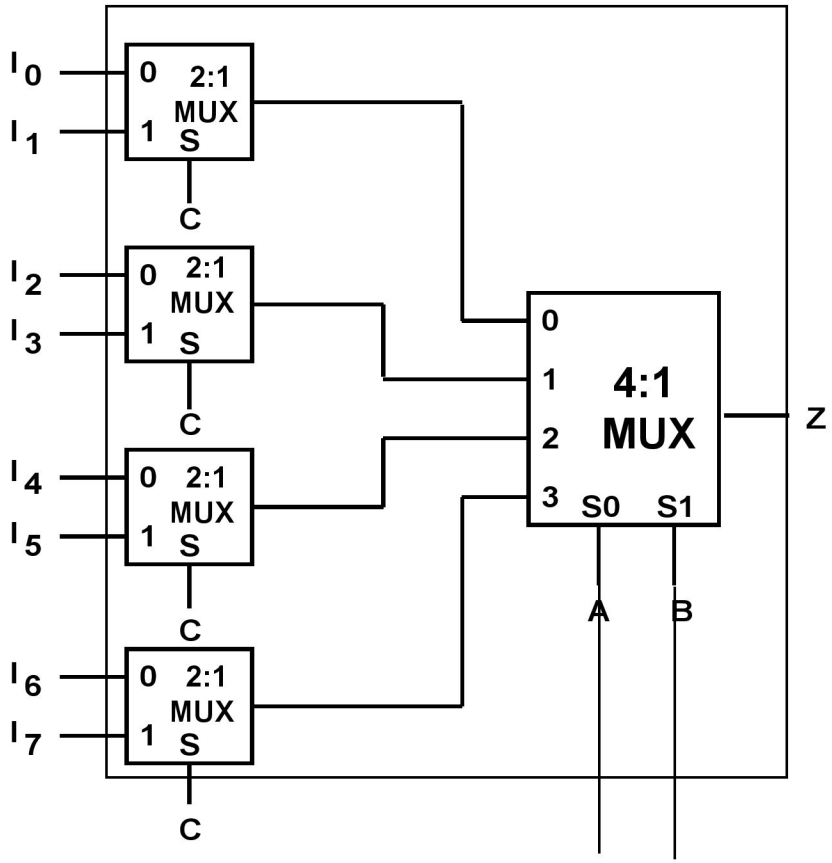


Cascading MUXes

- Design an 8:1 MUX by using smaller MUXes



Another Implementation

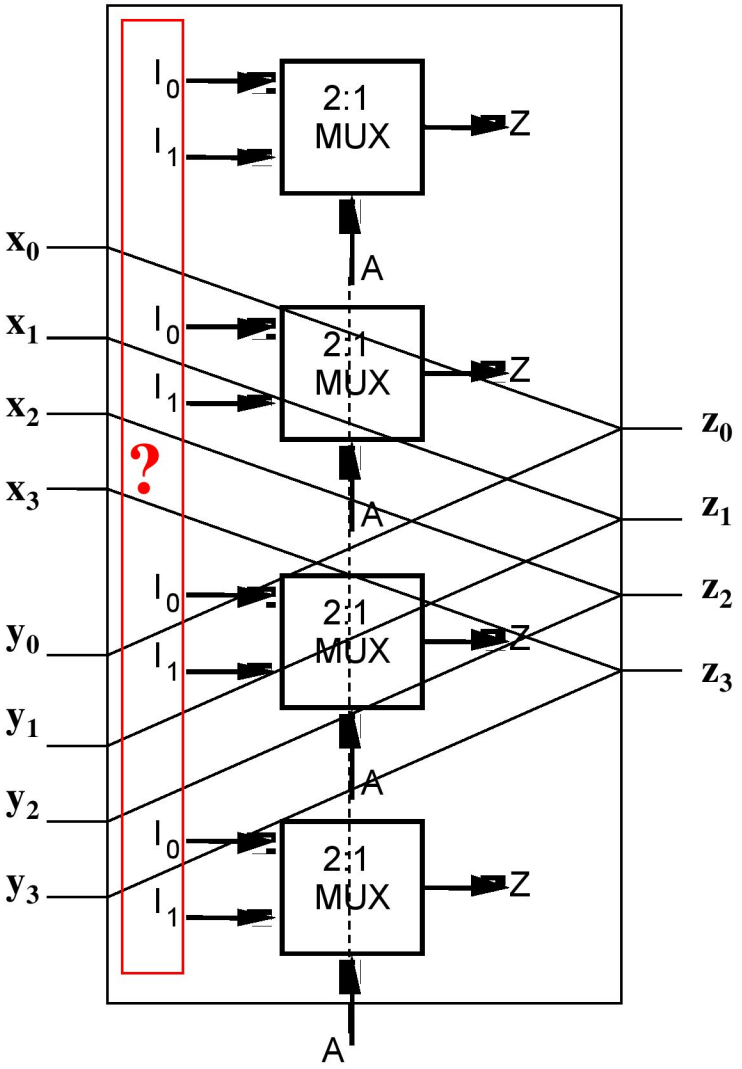


Larger Data Lines

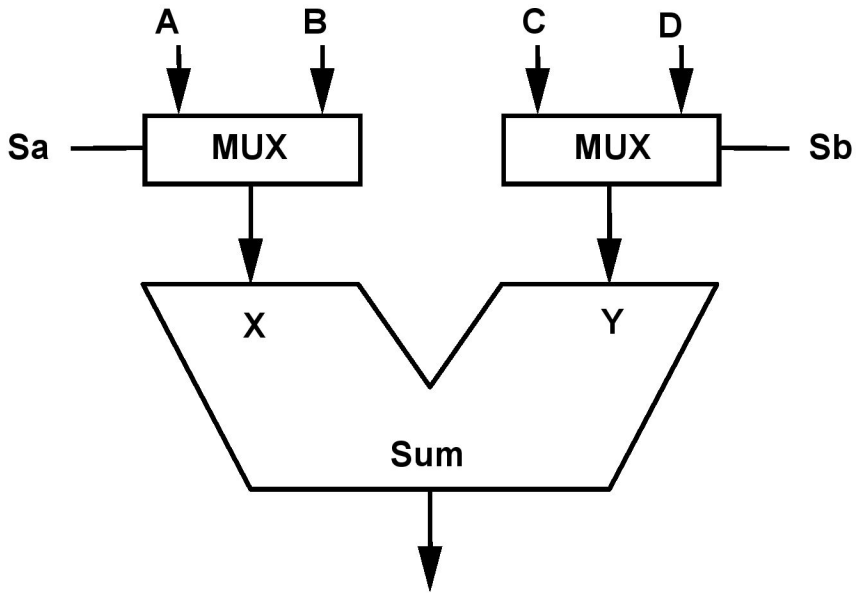
- What if we want to select m-bit datawords?
- Combine MUX blocks in parallel with common select and enable signals

4-bit data

- **Example:**
  - Selection between 2 sets of 4-bit inputs (x[3:0] or y[3:0])
  - Use four 2:1 MUXes in parallel
  - Tie all four control lines together
  - Connect x[3:0] to I_0s and y[3:0] to I_1s



Application

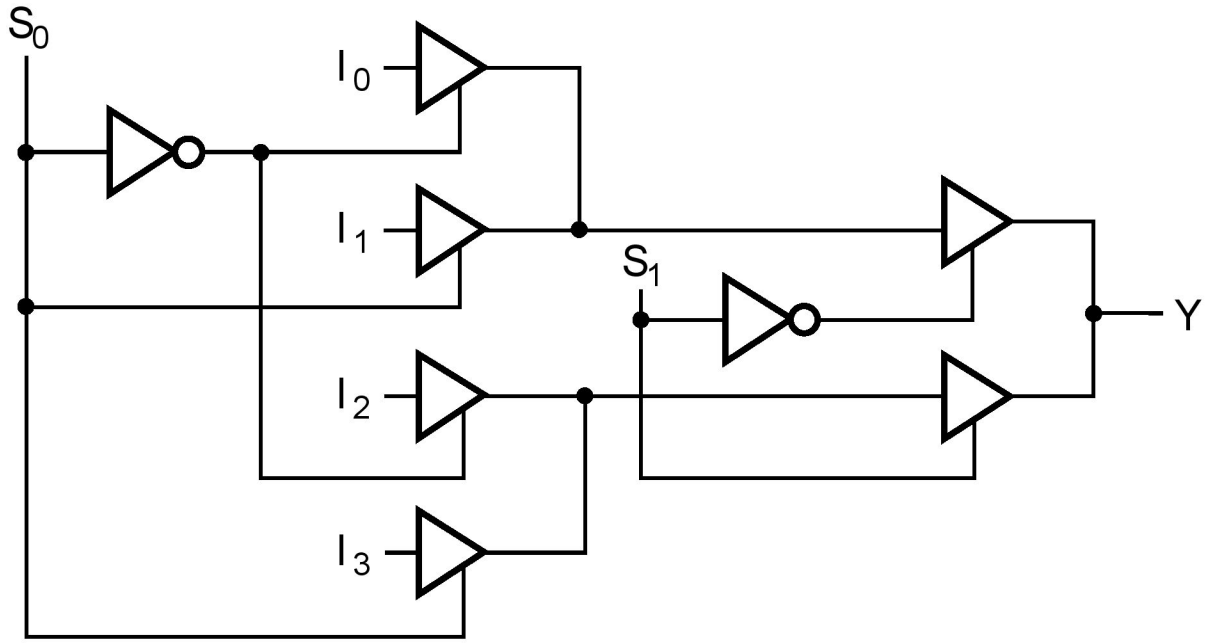


Adder with multiple input sources:

A+C or
A+D or
B+C or
B+D

MUX by 3-State Buffers

- Three-state logic in place of AND-OR



(b)

Output of gates can be connected

General Logic by MUX

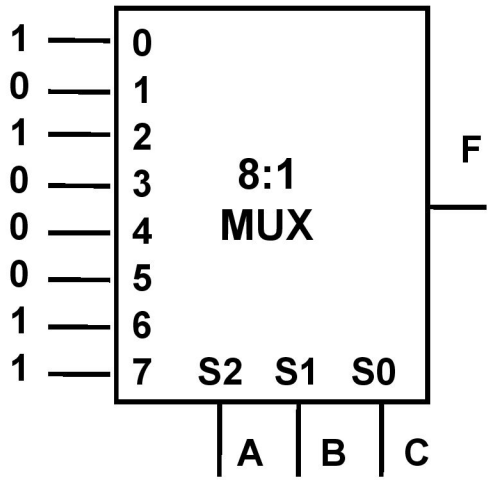
- Any Boolean function of n variables can be implemented using a 2^{n-1}:1 multiplexer
- n-1 inputs go to the control lines directly
- One input is used to appropriately determine the output
- The choice of the singled-out input doesn't change the functionality but it affects the complexity
- MUXes can only implement single-output functions

General Logic by MUX

- **Example:**

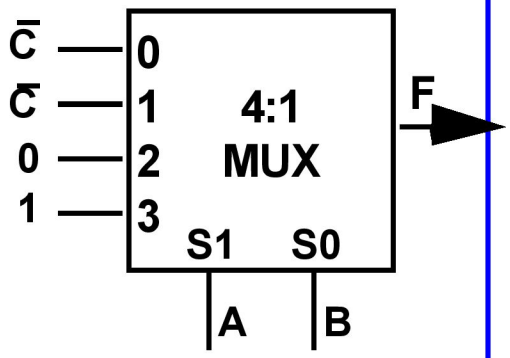
F = A' B' C' + A' B C' + A B C' + A B C

= A' B' (C') + A' B (C') + A B' (0) + A B (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	

A	B	F
0	0	C’
0	1	C’
1	0	0
1	1	1



What if A or B had been chosen as the singled-out input?

Using Smaller MUX

- How about implementing a 4-variable function by a 4:1 MUX
  - Can still be done
  - 2 input variables go to the 2 control lines directly
  - The other 2 input variables will go to the MUX inputs using some necessary gates
  - Choose the 2 that go to control inputs and the 2 that go to the MUX inputs carefully!
  - The choice affects the number of necessary gates

General Logic

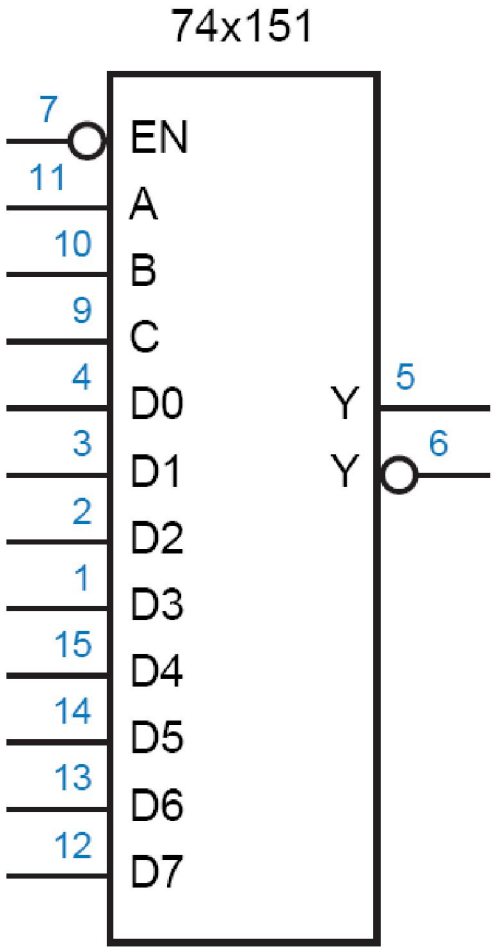
- **By decoder:**
  - Multiple outputs:
    - A single decoder
    - One OR gate for each output
- **By MUX:**
  - Multiple outputs:
    - One MUX needed for each output
    - No need for OR gates

- **Use MUX for single-output functions**
- **Use decoder for multiple-output functions**

Standard MSI MUXes

- **74x151**
  - 8:1 MUX

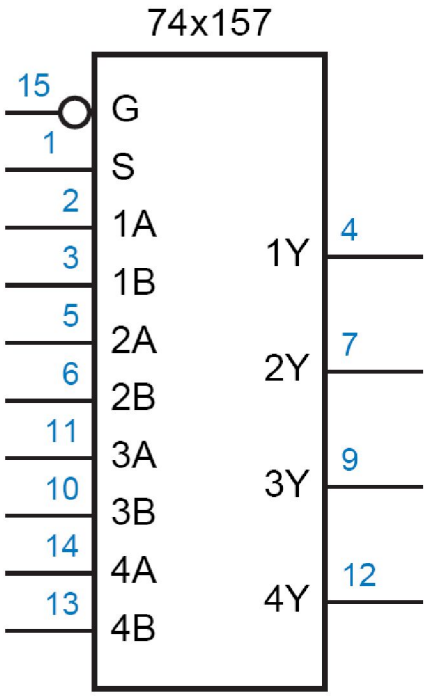
Inputs				*Outputs*	
EN_L	C	B	A	Y	Y_L
1	x	x	x	0	1
0	0	0	0	D0	D0′
0	0	0	1	D1	D1′
0	0	1	0	D2	D2′
0	0	1	1	D3	D3′
0	1	0	0	D4	D4′
0	1	0	1	D5	D5′
0	1	1	0	D6	D6′
0	1	1	1	D7	D7′



Standard MSI MUXes

- **74x157**
  - 4-bit wide, 2:1 MUX

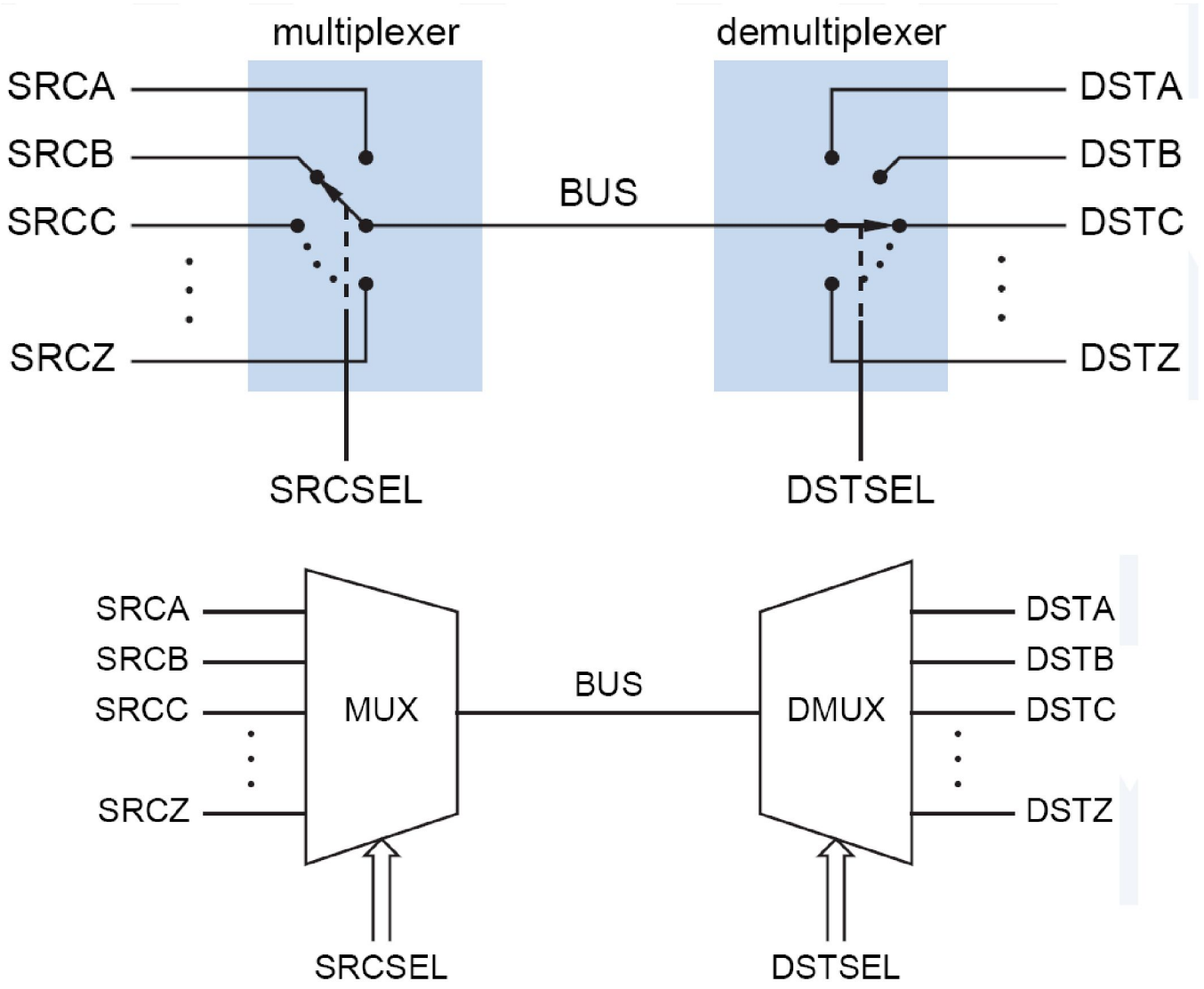
Inputs		*Outputs*			
G_L	S	1Y	2Y	3Y	4Y
1	x	0	0	0	0
0	0	1A	2A	3A	4A
0	1	1B	2B	3B	4B



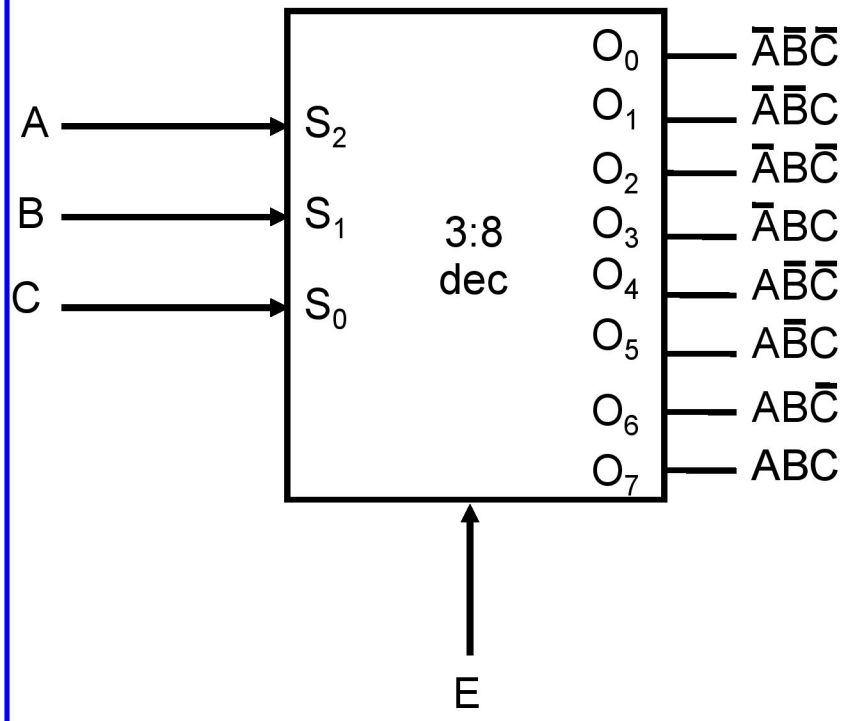
Demultiplexer

DEMUX

DEMUX


multiplexer
demultiplexer
SRCA
SRCB
SRCC
SRCZ
BUS
DSTA
DSTB
DSTC
DSTZ
SRCSEL
DSTSEL
SRCA
SRCB
SRCC
SRCZ
MUX
BUS
DMUX
DSTA
DSTB
DSTC
DSTZ
SRCSEL
DSTSEL

Decoder vs. DEMUX



E	A	B	C	O_0	O_1	O_2	O_3	O_4	O_5	O_6	O_7
0	X	X	X	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
1	0	1	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	1

By switching the view between (A, B, C) and E, can a decoder be considered a DEMUX?

Decoder vs. DEMUX