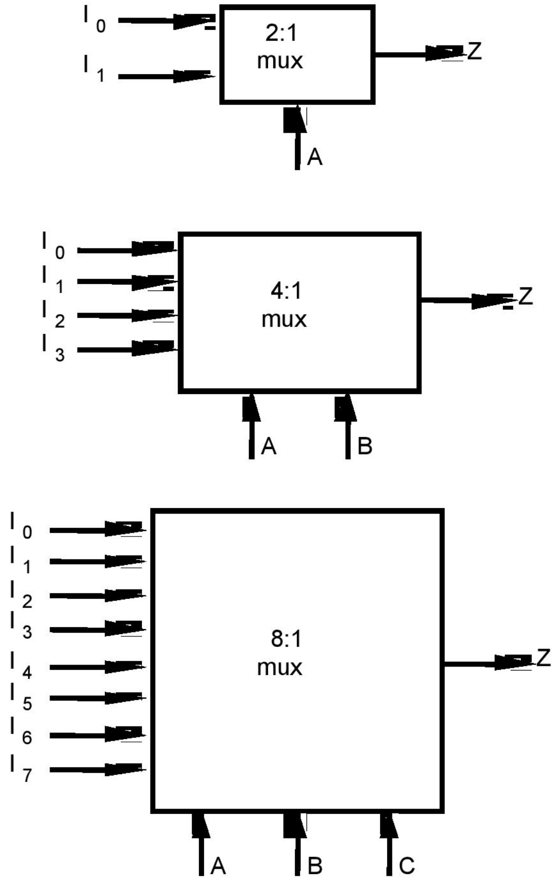
Multiplexer

MUX

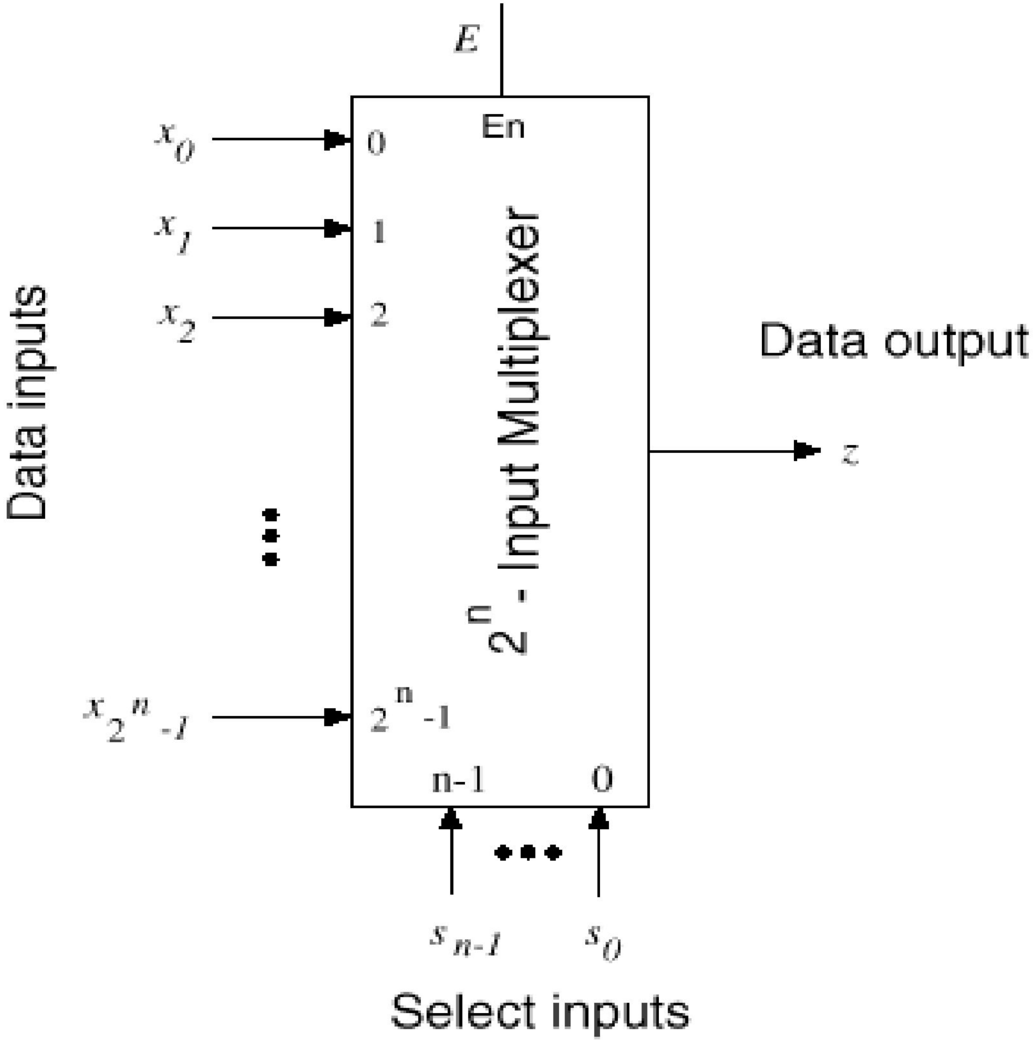
Multiplexer

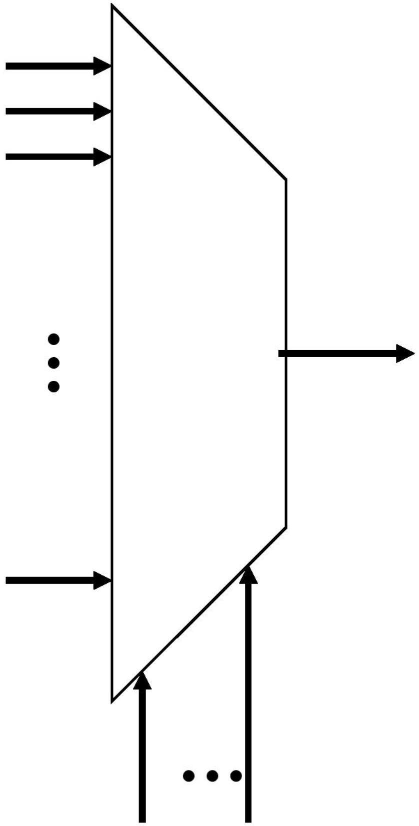
- **Multiplexer (Selector)**
  - 2^n data inputs,
  - n control inputs,
  - 1 output
  - Used to connect 2^n points to a single point
  - Form binary index of input connected to output

A	Z
0	I_0
1	I_1

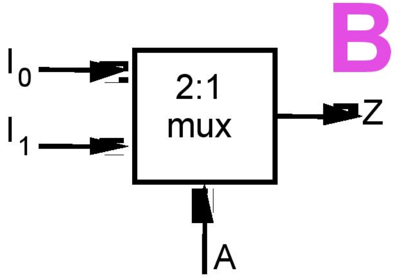


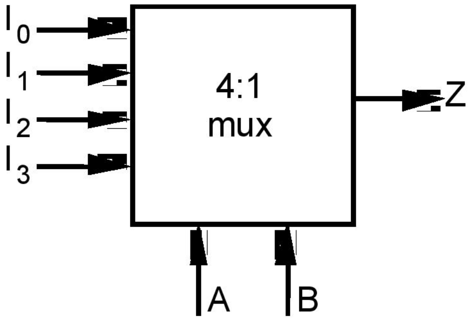
Multiplexer

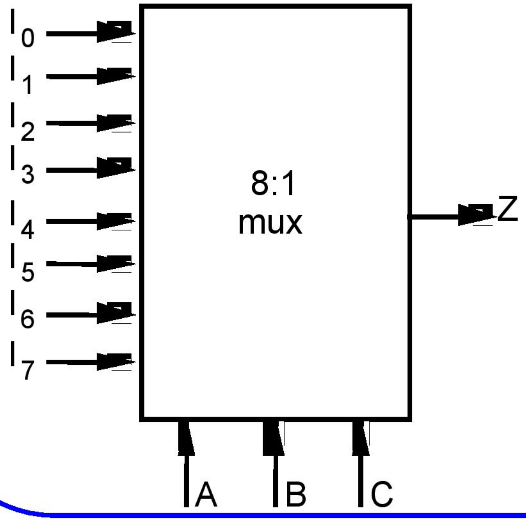


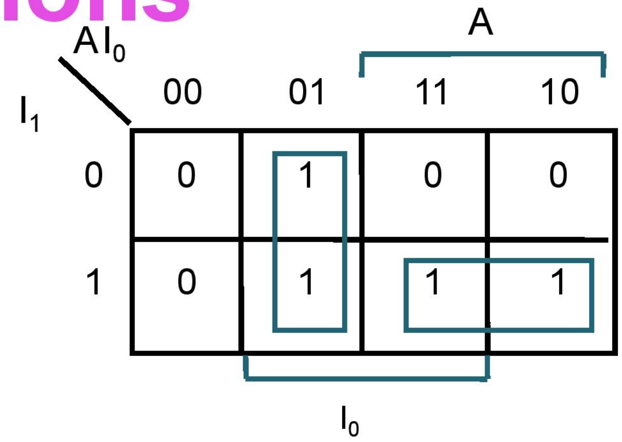


Boolean Functions



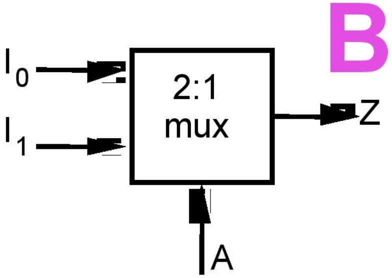




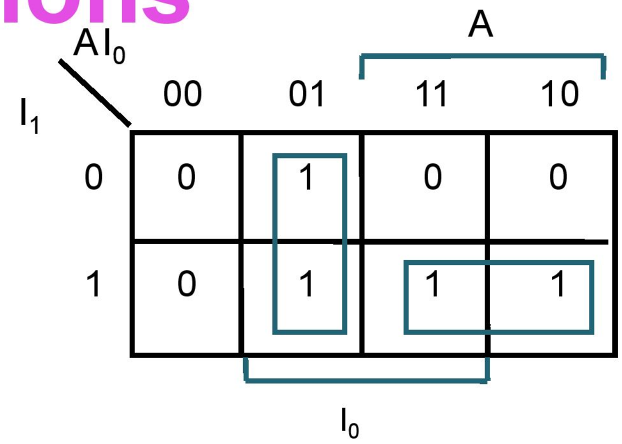


I_1 \ AI_0	00	01	11	10
0	0	1	0	0
1	0	1	1	1

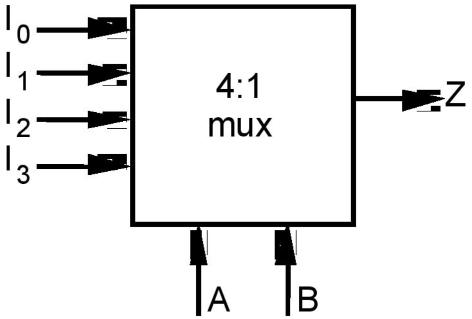
Boolean Functions



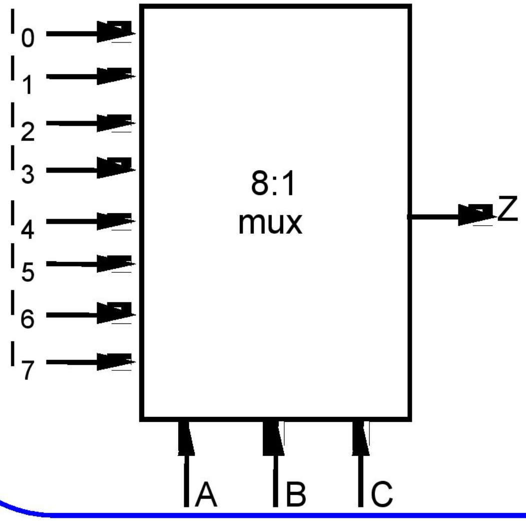
$$Z = A' I_0 + A I_1$$



I_1 \ AI_0	00	01	11	10
0	0	1	0	0
1	0	1	1	1



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



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

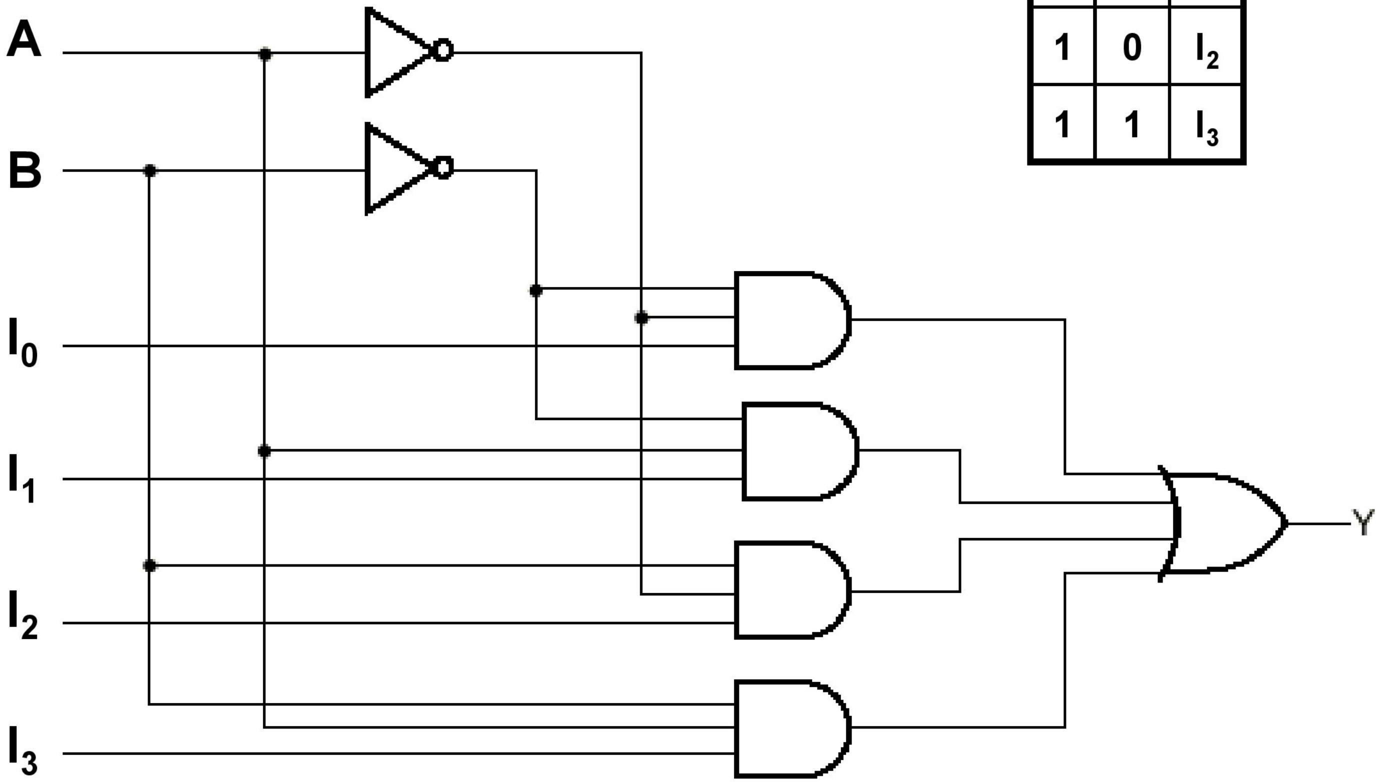
In general, $Z = \sum_{k=0}^{2^n -1} m_k \; I_k$

in minterm shorthand form

Circuit Diagram

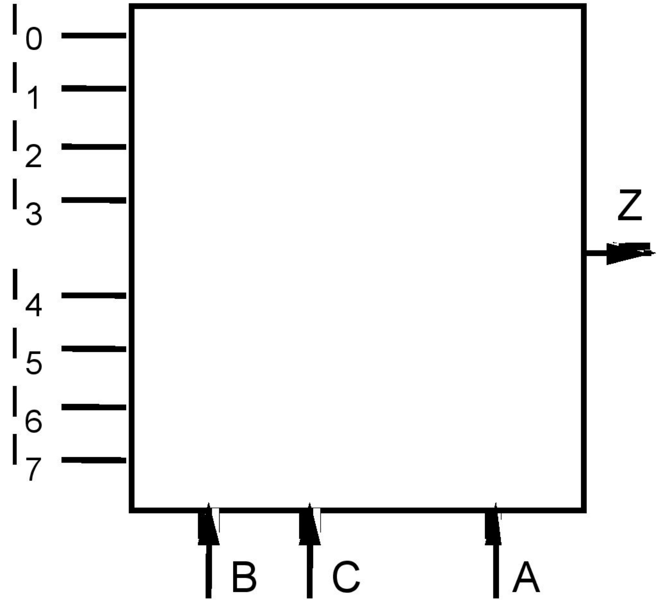
- **4-to-1 MUX**

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



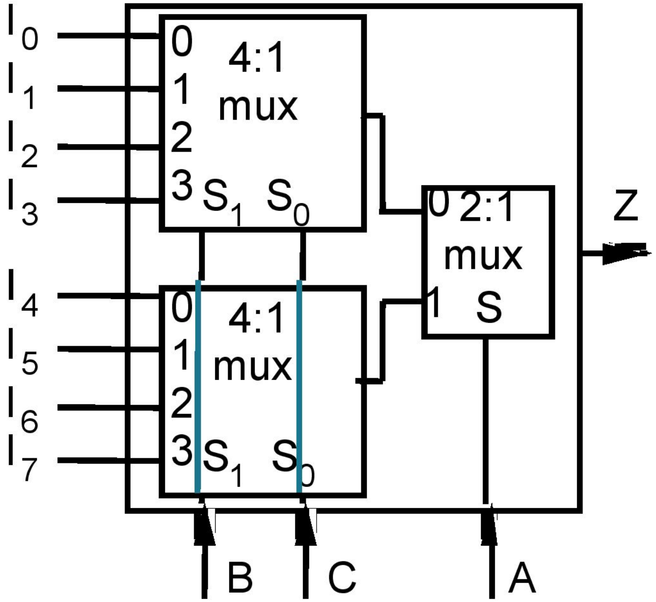
Cascading MUXes

- Design a MUX (8:1) by smaller MUXes

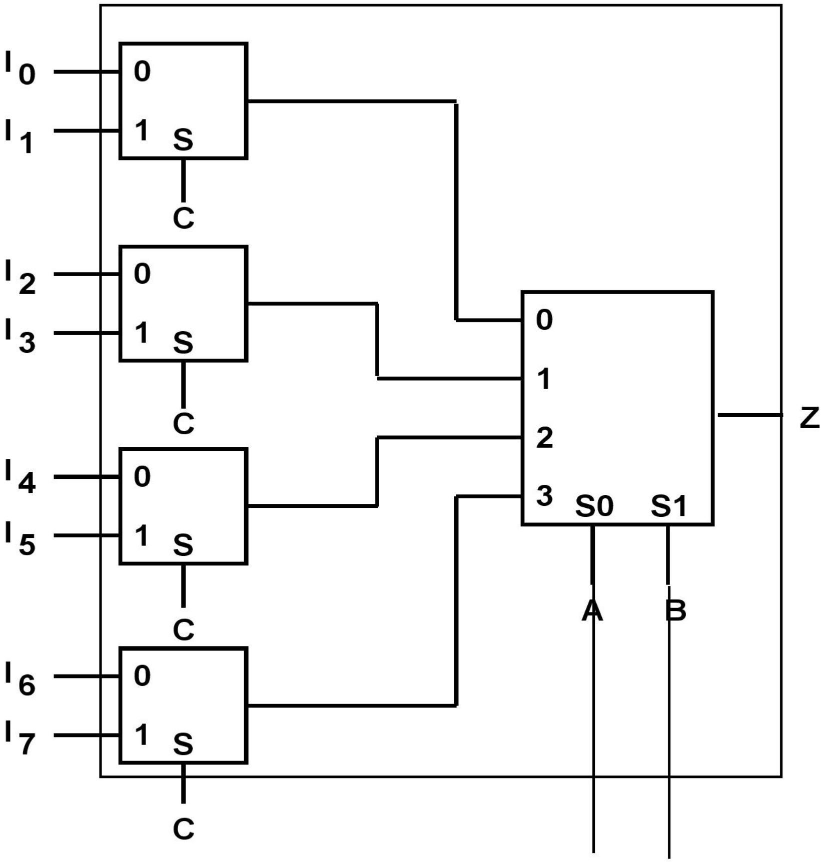


Cascading MUXes

- Design a MUX (8:1) by smaller MUXes



Another Implementation

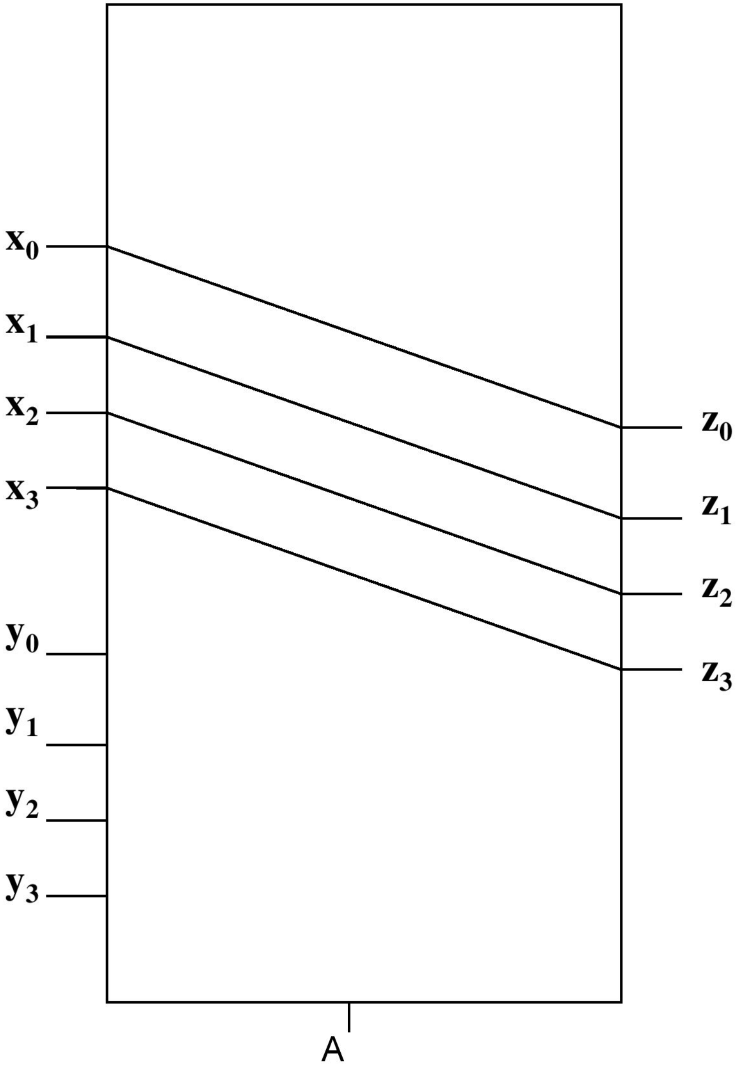


Larger Data Lines

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

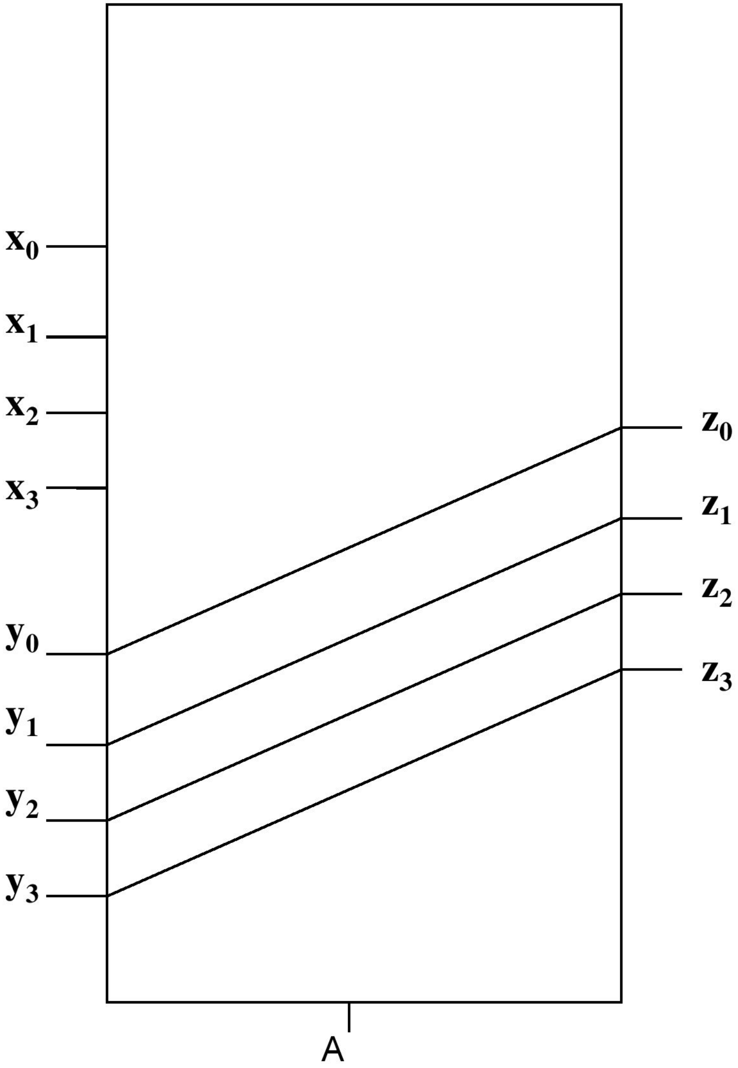
4-bit data

- **Example:**
  - Selection between 2 sets of 4-bit inputs
  - Enable line turns MUX on and off (E=1 is on).



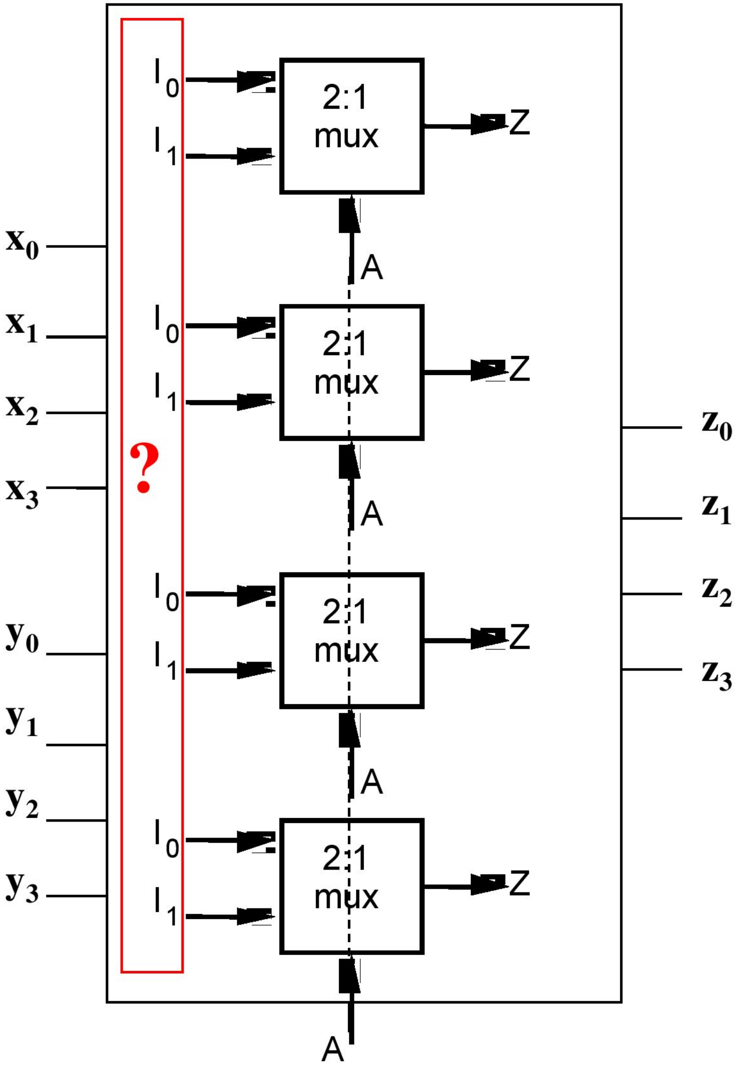
4-bit data

- **Example:**
  - Selection between 2 sets of 4-bit inputs

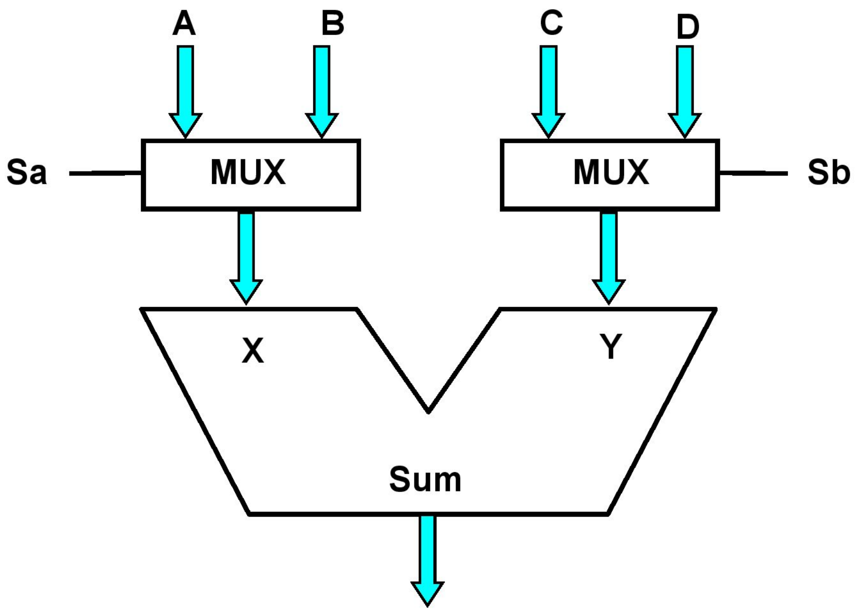


4-bit data

- **Example:**
  - Selection between 2 sets of 4-bit inputs



Application



Multiple input sources:

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

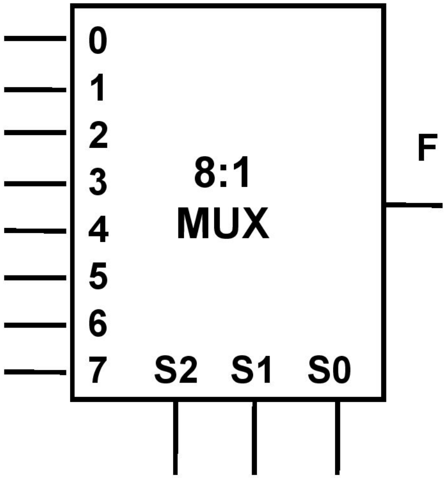
General Logic by MUX

- Any Boolean function of n variables can be implemented using a 2^{n-1}-to-1 multiplexer.

General Logic by MUX

- **Example:**

  $F = A'\,B'\,C' + A'\,B\,C' + A\,B\,C' + A\,B\,C$

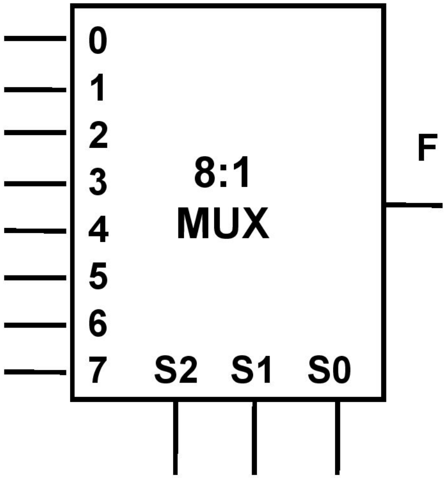


A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

General Logic by MUX

- **Example:**

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



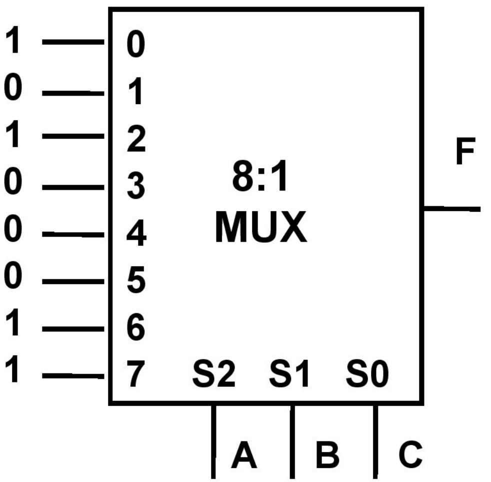
A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

General Logic by MUX

- **Example:**

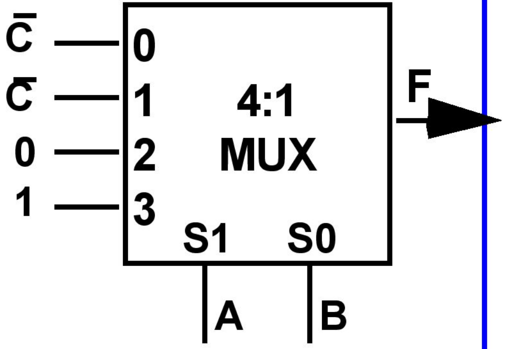
$F = A'\,B'\,C' + A'\,B\,C' + A\,B\,C' + A\,B\,C$

$= A'\,B'\,(C') + A'\,B\,(C') + A\,B'\,(0) + A\,B\,(1)$



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



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

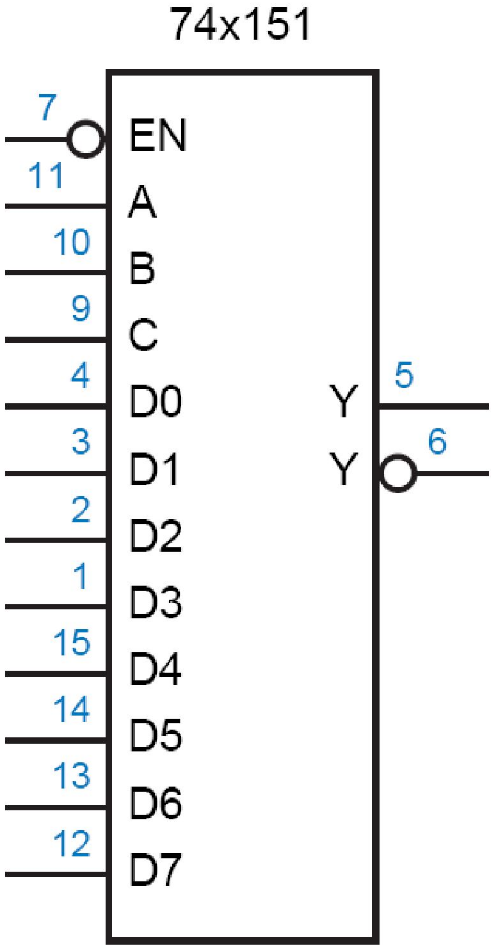
- **By decoder:**
  - Multiple outputs:
    - A single decoder,
    - One OR for each output
- **By MUX:**
  - Multiple outputs:
    - One MUX for each output,
    - No need for OR

- **→ Use MUX for few outputs,**
- **→ Use decoder for many outputs.**

Standard MSI MUXes

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

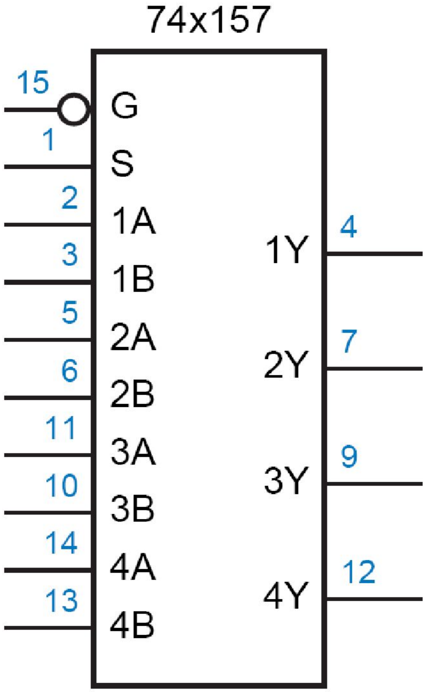
Inputs				*Outputs*	
EN_L	C	B	A	Y	Y_L
1	x	x	x	0	1
0	0	0	0	D0	D0′
0	0	0	1	D1	D1′
0	0	1	0	D2	D2′
0	0	1	1	D3	D3′
0	1	0	0	D4	D4′
0	1	0	1	D5	D5′
0	1	1	0	D6	D6′
0	1	1	1	D7	D7′



Standard MSI MUXes

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

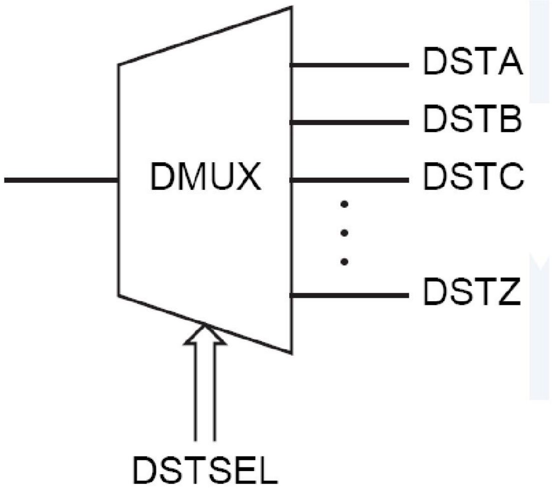
Inputs		*Outputs*			
G_L	S	1Y	2Y	3Y	4Y
1	x	0	0	0	0
0	0	1A	2A	3A	4A
0	1	1B	2B	3B	4B

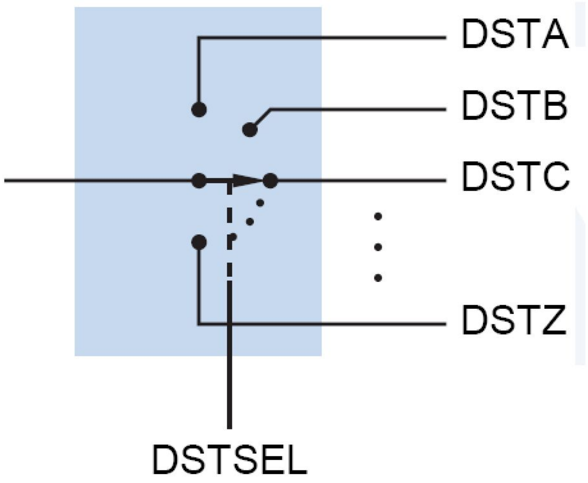


Demultiplexer

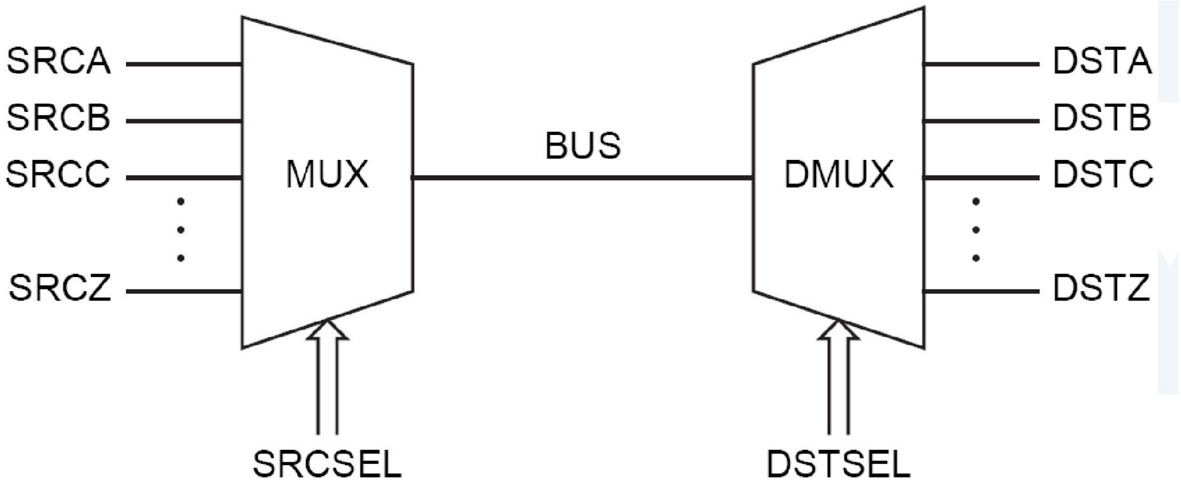
DEMUX

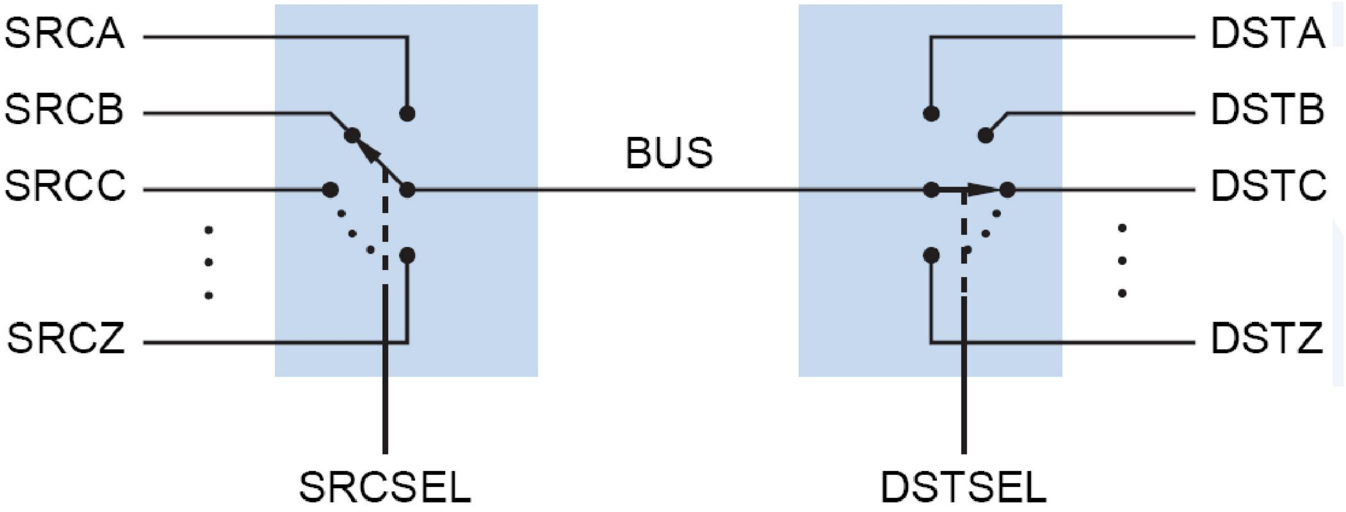
DEMUX





DEMUX


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


SRCA
SRCB
SRCC
SRCZ
BUS
DSTA
DSTB
DSTC
DSTZ
SRCSEL
DSTSEL