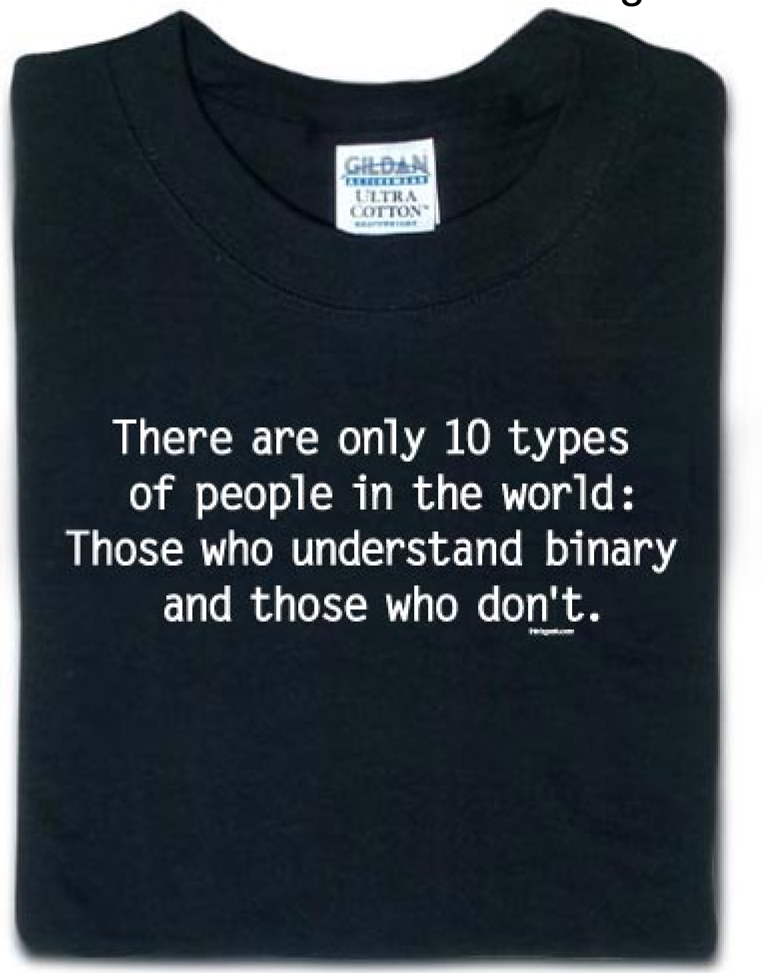
Last Time

- Resistors
- Ohm's Law
- Digital to analog conversion
- Binary numbers

www.thinkgeek.com

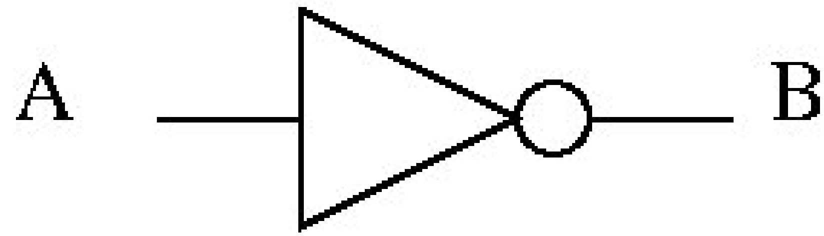


Today

- R-C Circuits
- Digital Circuits

What is the Gate?

- Logical Symbol:



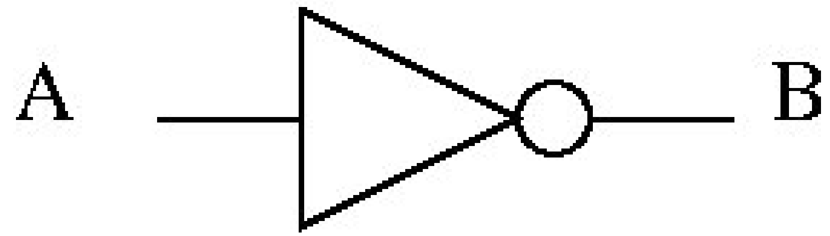
- Algebraic Notation:

- Truth Table:

A	B
0	
1	

The NOT Gate

- Logical Symbol:

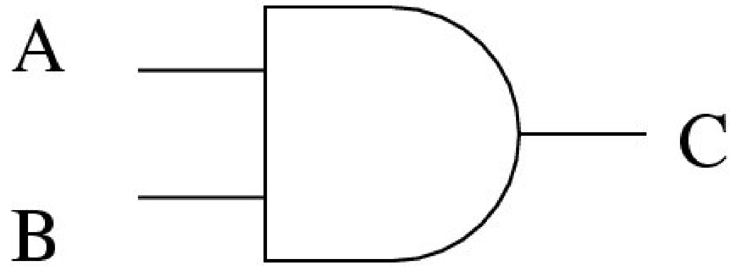


- Algebraic Notation: $B = \overline{A}$
- Truth Table:

A	B
0	1
1	0

And This Gate?

- Logical Symbol:



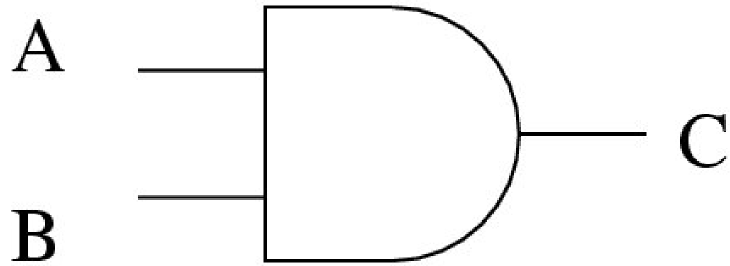
- Algebraic Notation: C = ?

- Truth Table:

A	B		C
0	0		
0	1		
1	0		
1	1		

The "AND" Gate

- Logical Symbol:



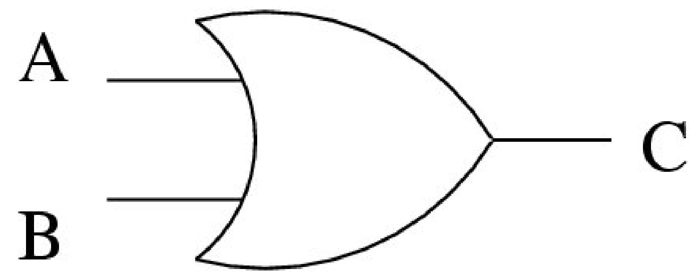
- Algebraic Notation: C = A*B = AB

- Truth Table:

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

And This Gate?

- Logical Symbol:



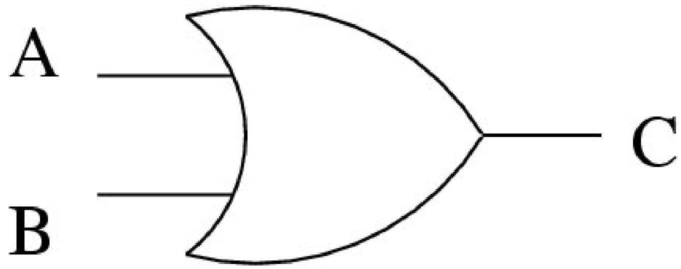
- Algebraic Notation: C = ?

- Truth Table:

A	B		C
0	0		
0	1		
1	0		
1	1		

The "OR" Gate

- Logical Symbol:



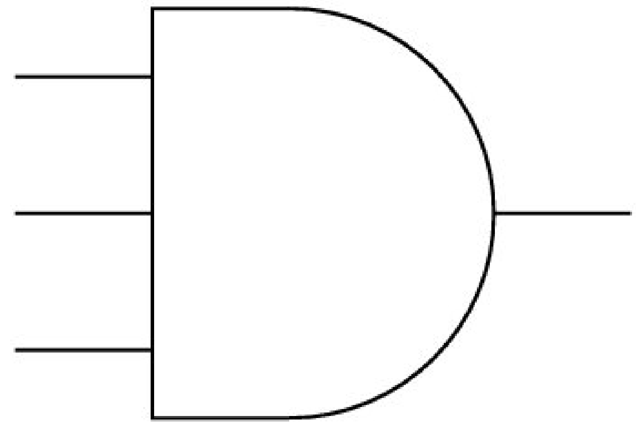
- Algebraic Notation: C = A+B

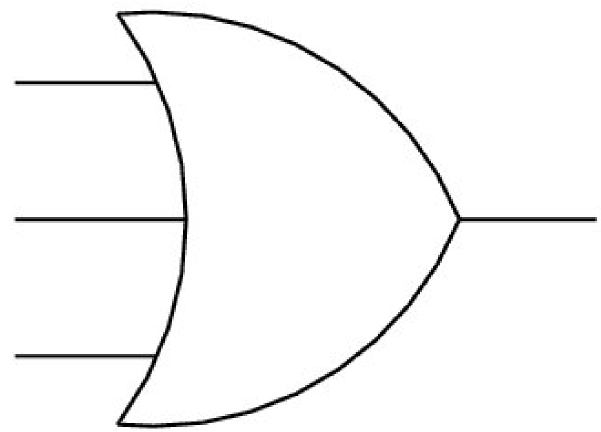
- Truth Table:

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

N-Input Gates

Gates can have an arbitrary number of inputs (2,3,4,8,16 are common)





An Example

Problem: implement an alarm system

- There are 3 inputs:
  - Door open ("1" represents open)
  - Window open
  - Alarm active ("1" represents active)
- And one output:
  - Siren is on ("1" represents on) when either the door or window are open – but only if the alarm is active

What is the truth table?

Alarm Example: Truth Table

D	W	A		Siren
0	0	0		0
0	0	1		0
0	1	0		0
0	1	1		1
1	0	0		0
1	0	1		1
1	1	0		0
1	1	1		1

$$\overline{D}\,W\,A$$
$$+$$
$$D\,\overline{W}\,A$$
$$+$$
$$D\,W\,A$$

Alarm Example: Truth Table

D	W	A		Siren
0	0	0		0
0	0	1		0
0	1	0		0
0	1	1		1
1	0	0		0
1	0	1		1
1	1	0		0
1	1	1		1

“minterm”

$\overline{D}\,W\,A$

$+$

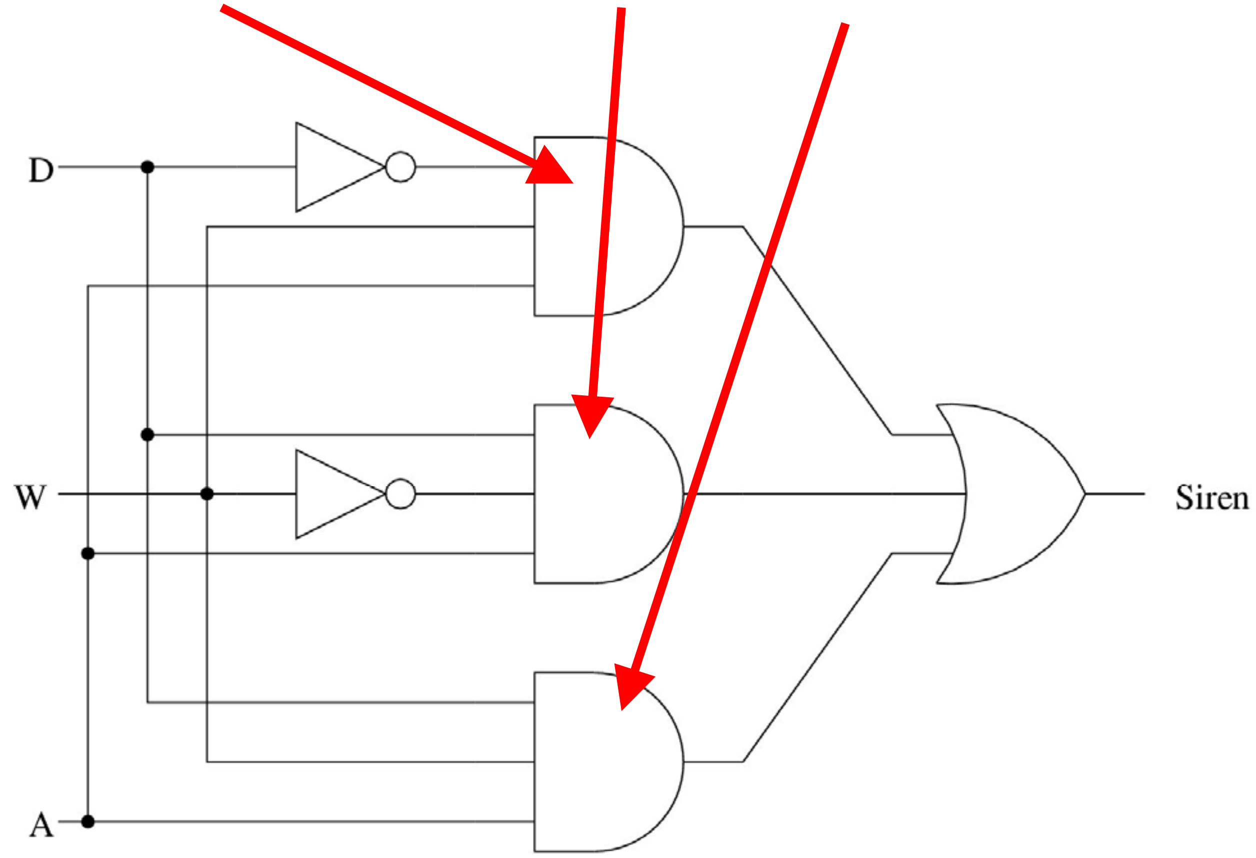
$D\,\overline{W}\,A$

$+$

$D\,W\,A$

Alarm Example: Circuit

Siren = $\overline{D}$ W A + D $\overline{W}$ A + D W A



Alarm Example: Circuit

Is a simpler circuit possible?

NOT Operator

Definition:

- $\overline{0} = 0' = 1$
- $\overline{1} = 1' = 0$

NOTE: this is identical to our truth table (just a slightly different notation)

Suppose that "X" is a Boolean variable, then:

- $\overline{\overline{X}} = X'' = X$

OR (+) Operator

Definition:

- $0+0 = 0$
- $0+1 = 1$
- $1+0 = 1$
- $1+1 = 1$

OR (+) Operator

Suppose "X" is a Boolean variable, then:

- $0 + X = X$
- $1 + X = 1$
- $X + X = X$
- $X + X' = 1$

AND (*) Operator

Definition:

- 0*0 = 0
- 0*1 = 0
- 1*0 = 0
- 1*1 = 1

AND (*) Operator

Suppose “X” is a Boolean variable, then:

- 0 * X = 0
- 1 * X = X
- X * X = X
- X * X’ = 0

Boolean Algebra Rules: Precedence

The AND operator applies before the OR operator:

A * B + C = (A * B) + C

A + B * C = A + (B * C)

Boolean Algebra Rules: Association Law

If there are several AND operations, it does not matter which order they are applied in:

$$A * B * C = (A * B) * C = A * (B * C)$$

Boolean Algebra Rules: Association Law

Likewise for the OR operator:

$$A + B + C = (A + B) + C = A + (B + C)$$

Boolean Algebra Rules: Distributive Law

AND distributes across OR:

A * (B + C) = (A * B) + (A * C)

A + (B * C) = (A + B) * (A + C)

Boolean Algebra Rules: Commutative Law

Both AND and OR are symmetric operators (the order of the variables does not matter):

A + B = B + A

A * B = B * A

Back to our Alarm Example

D	W	A		Siren
0	0	0		0
0	0	1		0
0	1	0		0
0	1	1		1
1	0	0		0
1	0	1		1
1	1	0		0
1	1	1		1

D' W A

+

D W' A

+

D W A

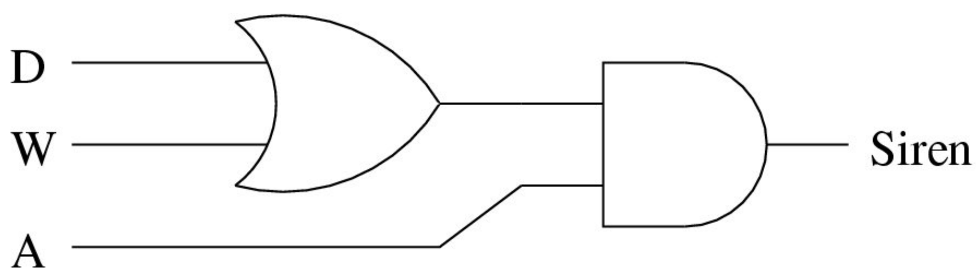
Reduction with Algebra

D’ W A + D W’ A + DWA	
= D’ W A + D W’ A + DWA + DWA	X + X = X
= D’ WA + DWA + DW’A + DWA	Commutative Law
= D’ WA + DWA + W’DA + WDA	“
= (D’ + D) WA + (W’ + W) DA	Assoc +Dist Laws
= 1* W A + 1* DA	X + X’ = 1

Reduction with Algebra (cont)

1* W A + 1* DA	
= WA + DA	X * 1 = X
= (W + D) * A	Distributive Law

We have a much smaller circuit!



Multiple Output Variables

How do we handle this?

- One algebraic expression for each output

- But: in the final implementation, some sub-circuits may be shared

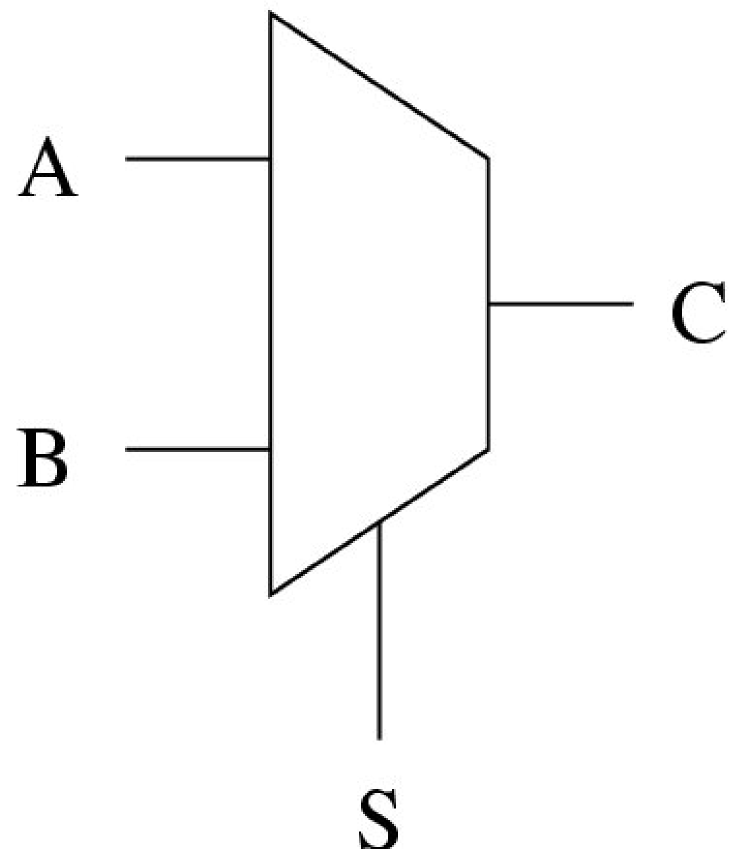
More Logical Components

- Multiplexer
- Demultiplexer
- Tristate buffer

2-Input Multiplexer

A multiplexer is a device that selects between two input lines

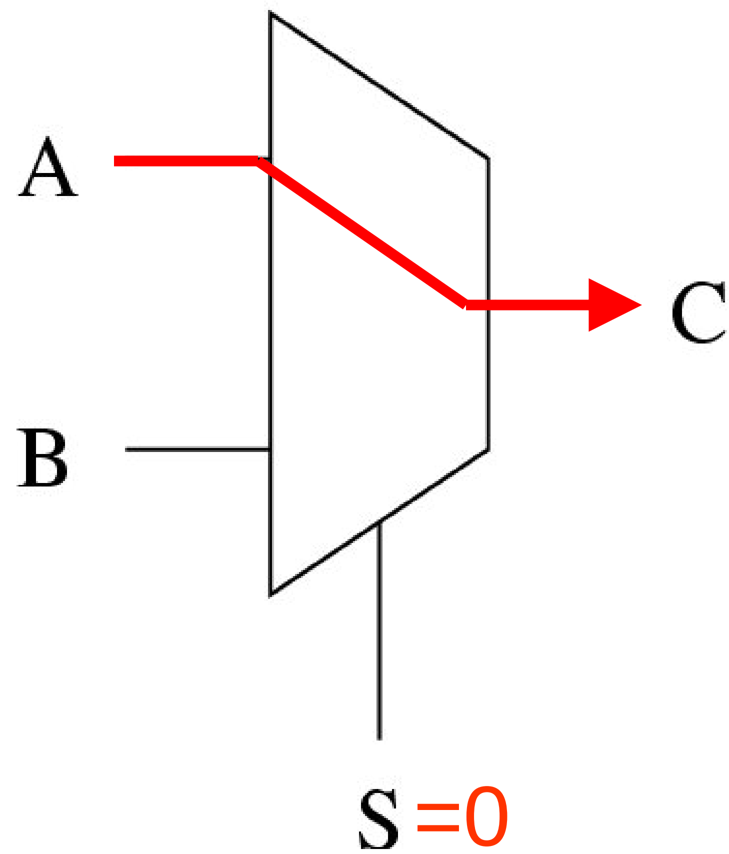
- A & B are the inputs
- S is the selection signal (also an input)
- C is a copy of A if S=0
- C is a copy of B if S=1



2-Input Multiplexer

A multiplexer is a device that selects between two input lines

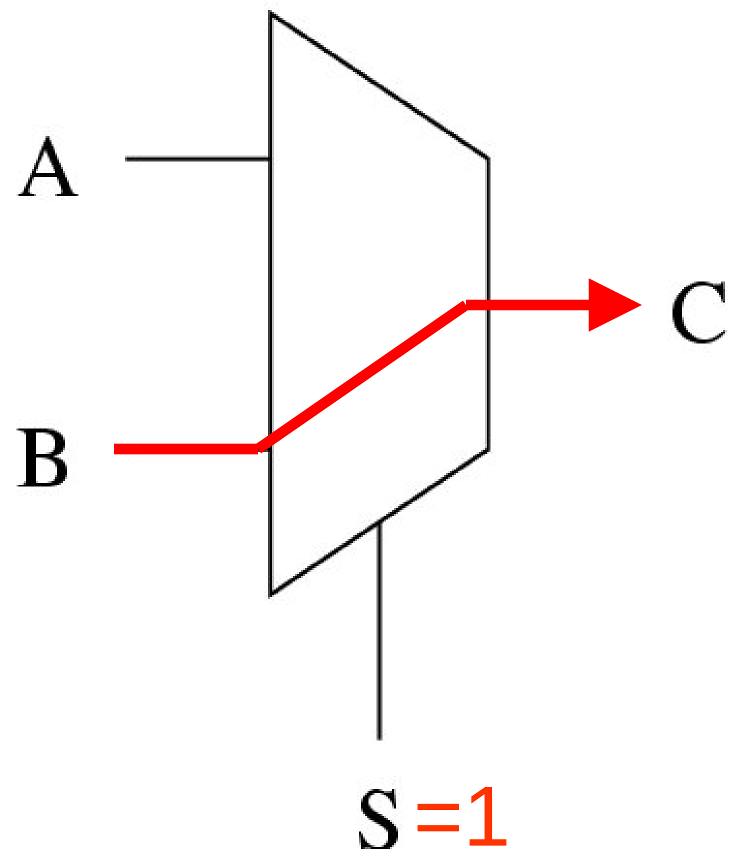
- A & B are the inputs
- S is the selection signal (also an input)
- C is a copy of A if S=0
- C is a copy of B if S=1



2-Input Multiplexer

A multiplexer is a device that selects between two input lines

- A & B are the inputs
- S is the selection signal (also an input)
- C is a copy of A if S=0
- C is a copy of B if S=1



Multiplexer Truth Table

What does the algebraic expression look like?

S	A	B		C
0	0	0		0
0	0	1		0
0	1	0		1
0	1	1		1
1	0	0		0
1	0	1		1
1	1	0		0
1	1	1		1

Multiplexer Truth Table

	S	A	B	C
	0	0	0	0
	0	0	1	0
S'AB'	0	1	0	1
S'AB	0	1	1	1
	1	0	0	0
SA'B	1	0	1	1
	1	1	0	0
SAB	1	1	1	1

Multiplexer

$C = S'AB' + S'AB + SA'B + SAB$

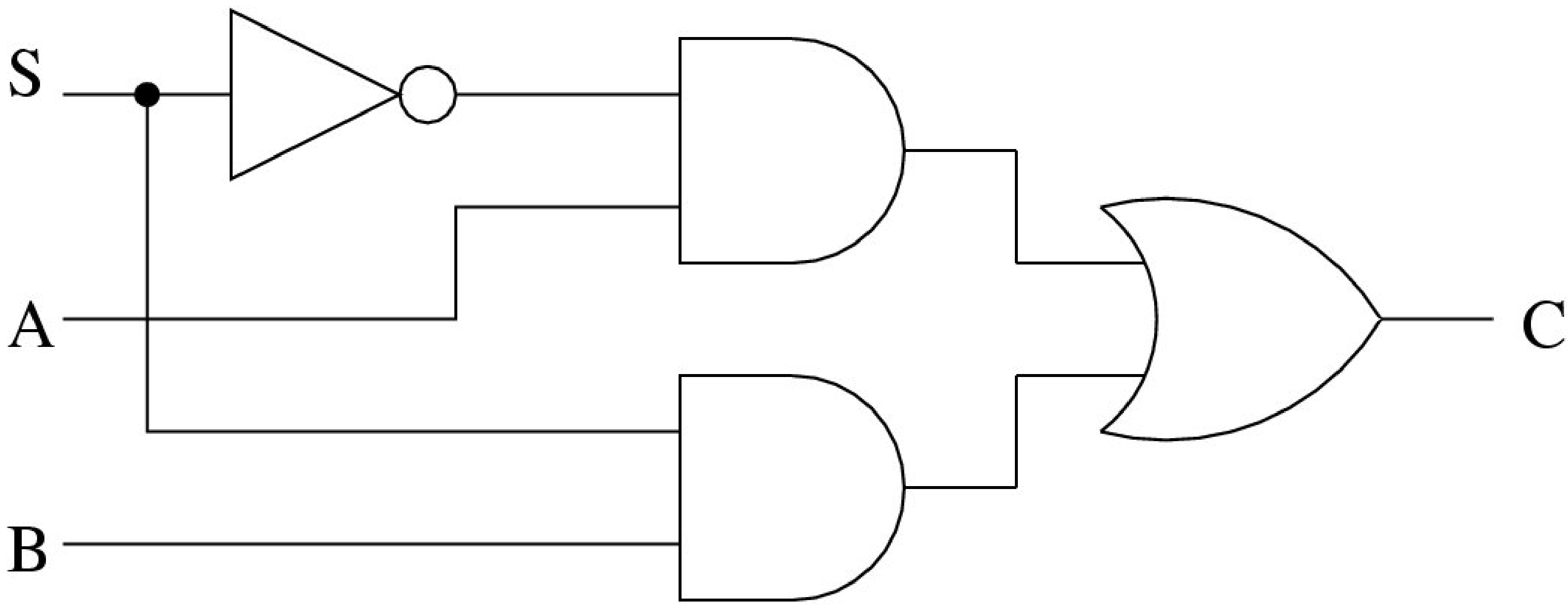
Is there a simpler expression?

Reduction with Algebra

S'AB' + S'AB + SA'B + SAB	
= S'A(B' + B) + SB(A' + A)	Associative + Distributive
= S'A 1 + SB 1	X + X' = 1
= S'A + SB	X + 1 = X

Multiplexer Implementation

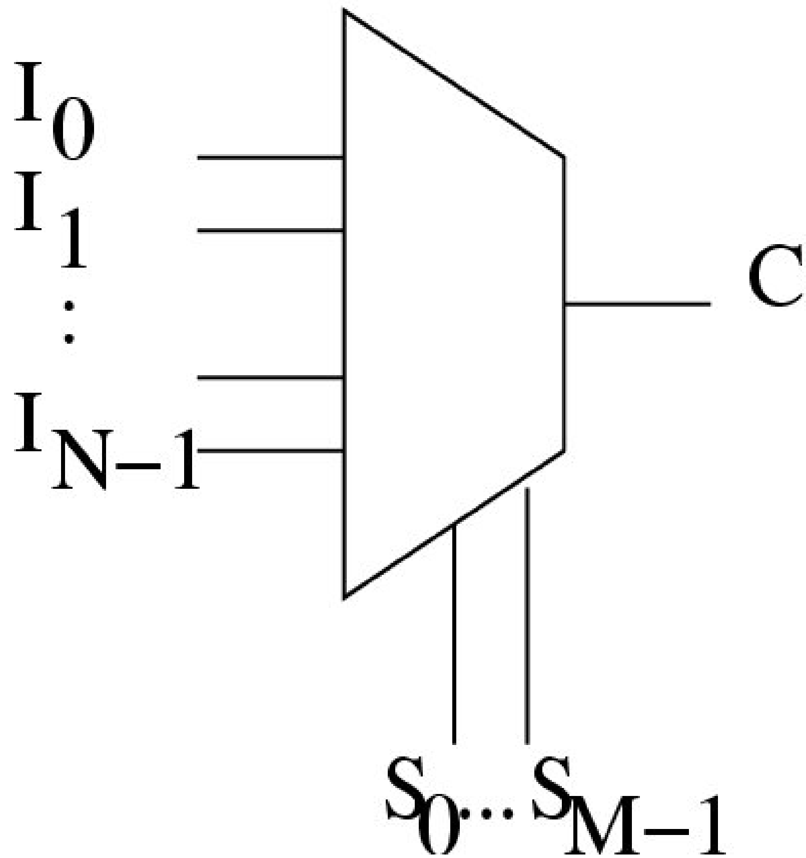
C = S'A + SB



N-Input Multiplexer

Suppose we want to select from between N different inputs.

- This requires more than one select line. How many?



N-Input Multiplexer

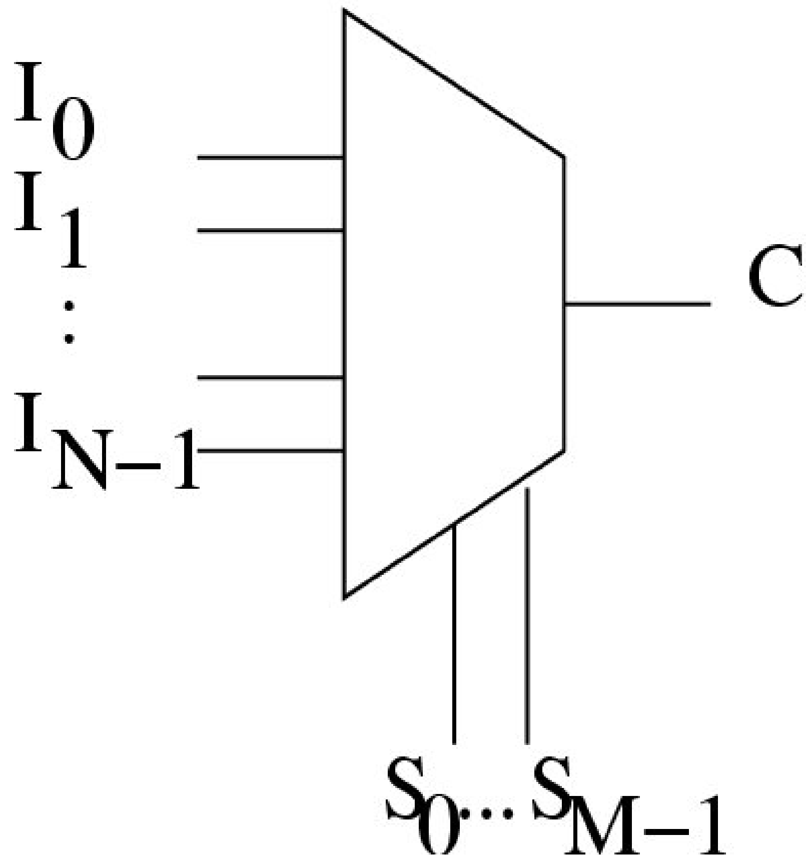
How many select lines?

- $M = \log_2 N$

  or

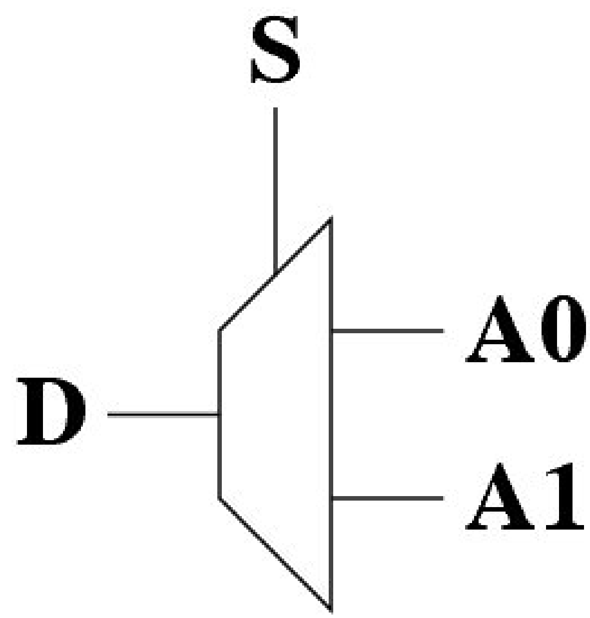
- $N = 2^M$

What would the N=8 implementation look like?



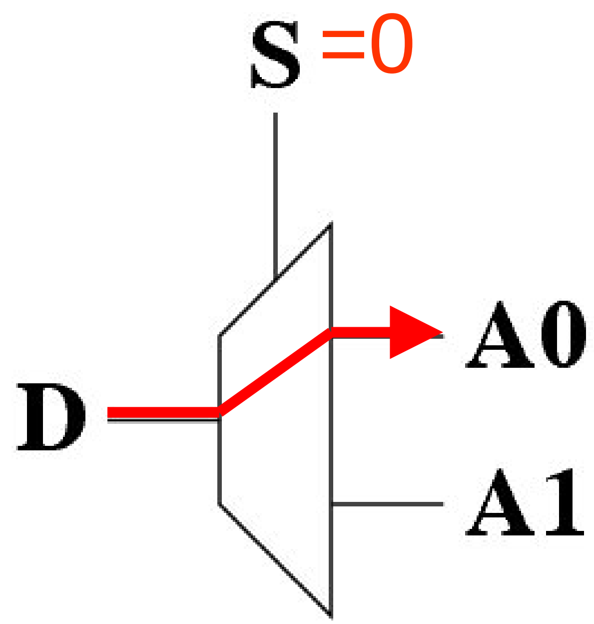
Demultiplexer

- The multiplexer reduces N signals down to 1 (with M select lines)
- A demultiplexer routes a data input (D) to one of N output lines (As)
  - Which A depends on the select lines



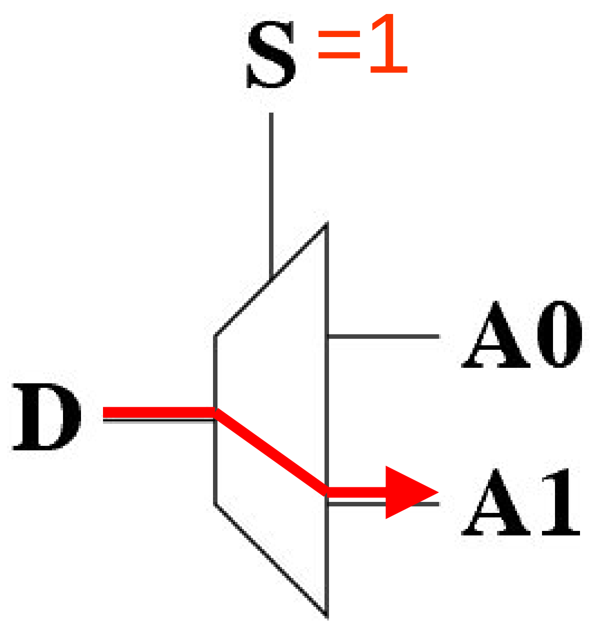
Demultiplexer

- The multiplexer reduces N signals down to 1 (with M select lines)
- A demultiplexer routes a data input (D) to one of N output lines (As)
  – Which A depends on the select lines



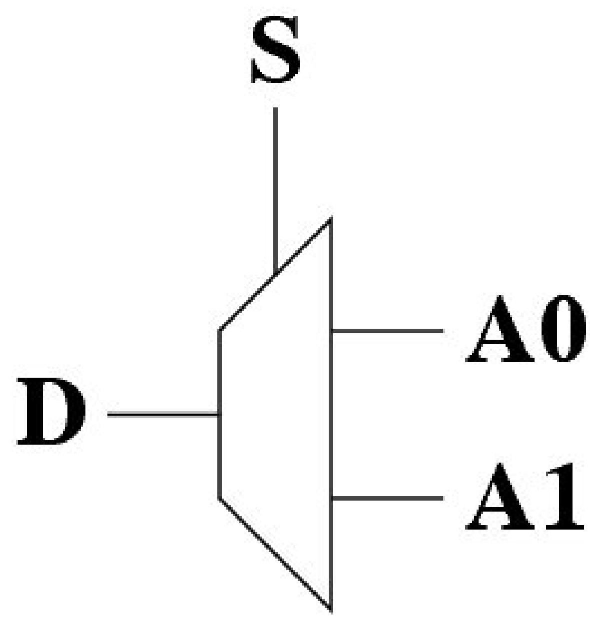
Demultiplexer

- The multiplexer reduces N signals down to 1 (with M select lines)
- A demultiplexer routes a data input (D) to one of N output lines (As)
  - Which A depends on the select lines



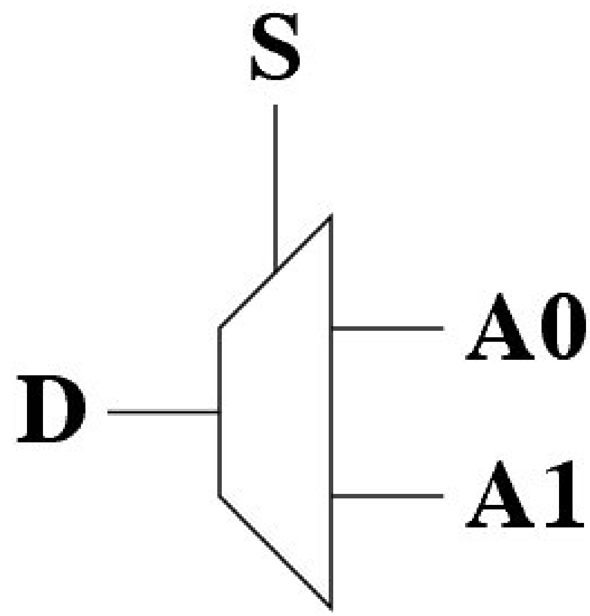
Demultiplexer Notes

- The symbol that we use is the same as for the multiplexer
  – Select lines are still inputs
  – But the other inputs and outputs are reversed
- Multiplexer or demultiplexer in a circuit: you must infer this from the labels or from the rest of the circuit
  – When in doubt, ask



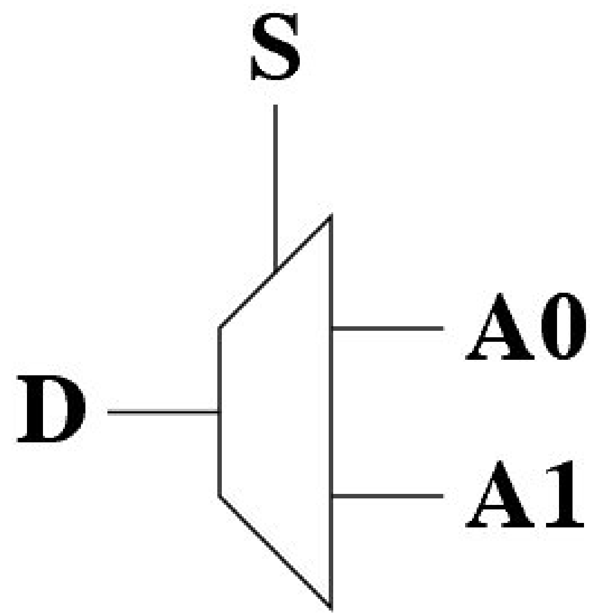
Demultiplexer Truth Table

S	D		A_1	A_0
0	0			
0	1			
1	0			
1	1			



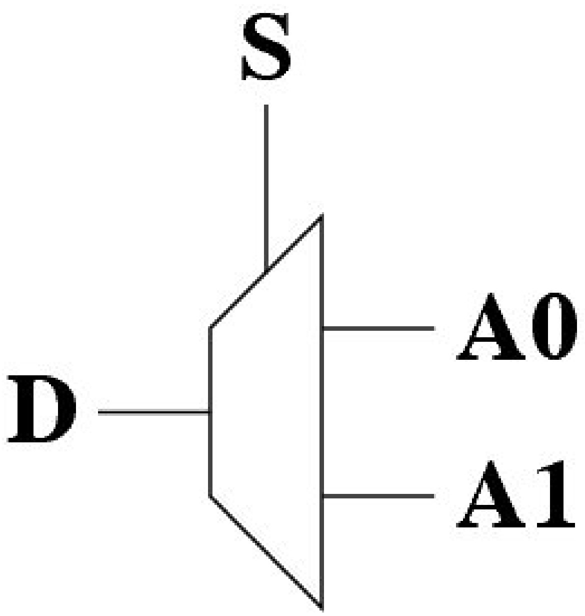
Demultiplexer Truth Table

S	D		A_1	A_0
0	0		0	0
0	1		0	1
1	0		0	0
1	1		1	0



Demultiplexer Algebraic Specification

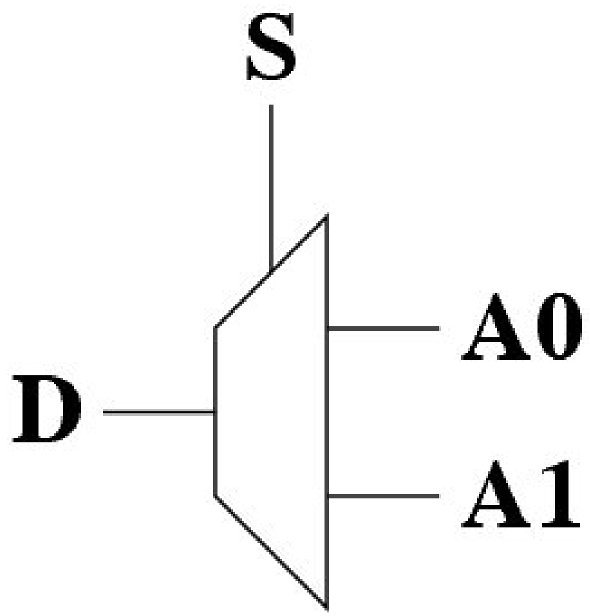
S	D		A_1	A_0
0	0		0	0
0	1		0	1
1	0		0	0
1	1		1	0



Demultiplexer Algebraic Specification

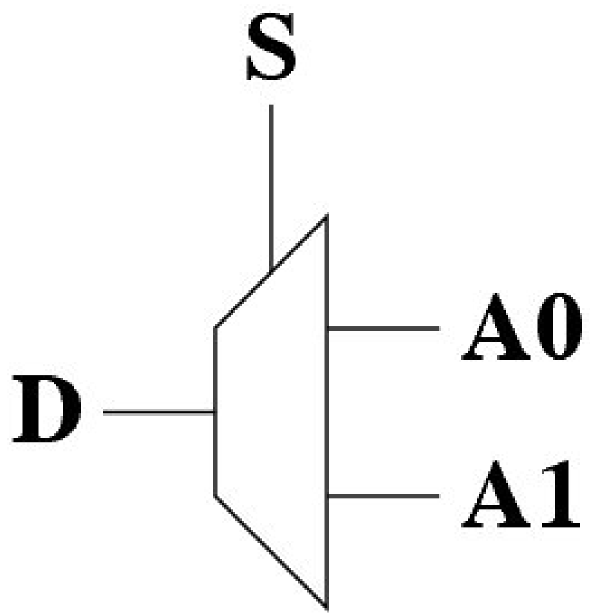
S	D		A_1	A_0
0	0		0	0
0	1		0	1
1	0		0	0
1	1		1	0

$A_0 = S'D$



Demultiplexer Algebraic Specification

S	D		A_1	A_0
0	0		0	0
0	1		0	1
1	0		0	0
1	1		1	0



$A_0 = S'D$

$A_1 = SD$

Demultiplexer Circuit

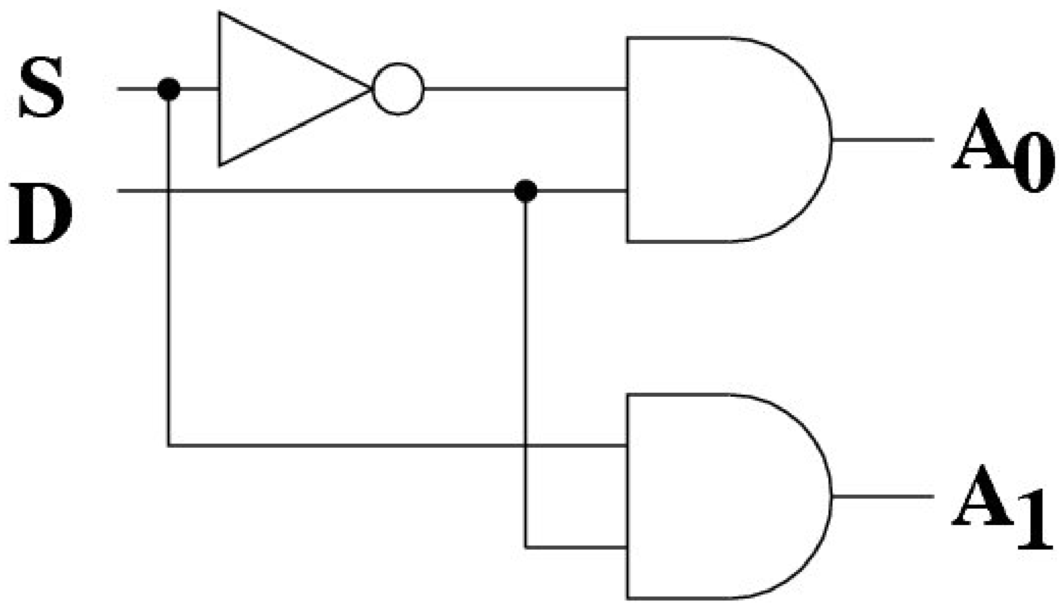
$A_0 = S'D$

$A_1 = SD$

Demultiplexer Circuit

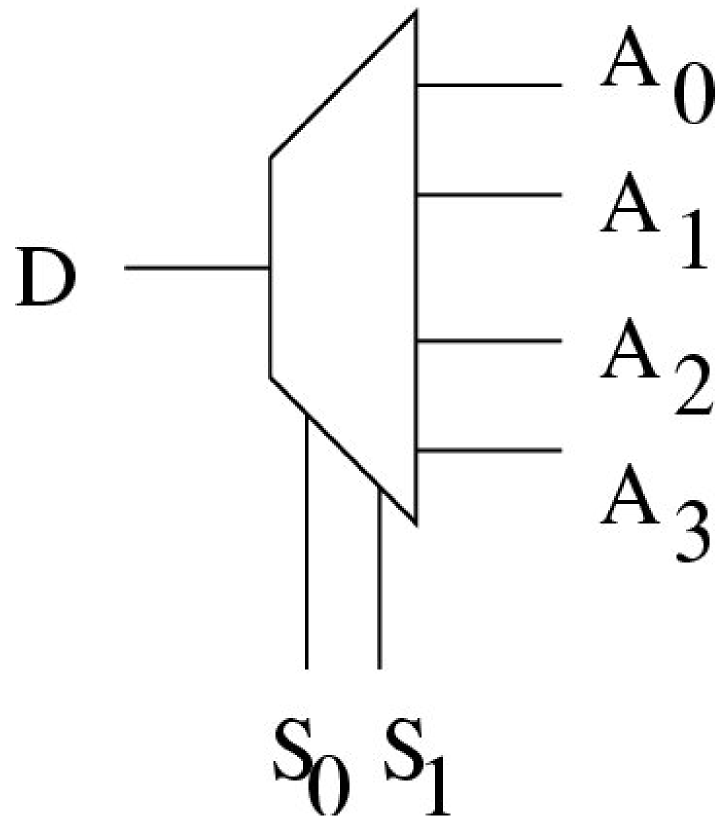
$A_0 = S'D$

$A_1 = SD$



2-Input Demultiplexer

Here, "input" refers to the number of select lines

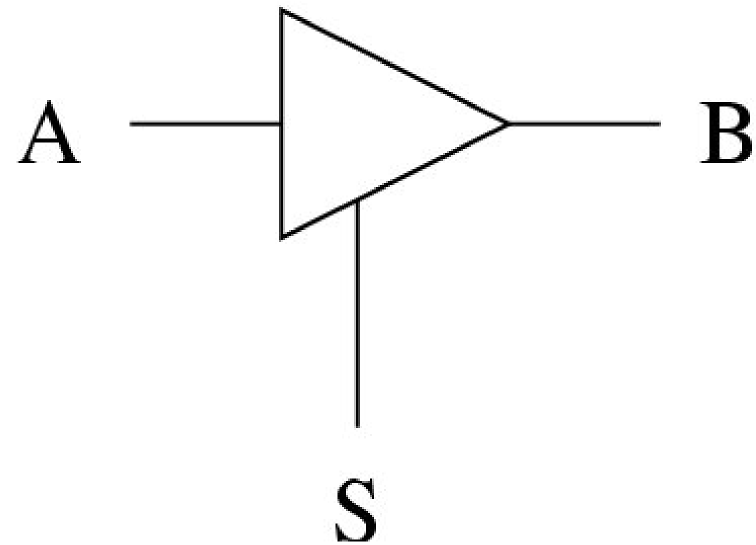


Tristate Buffers

- Until now: the output line(s) of each device are driven either high or low
  - So the line is either a source or a sink of current
- Tristate buffers can do this or leave the line floating (as if it were not connected to anything)

Tristate Buffers

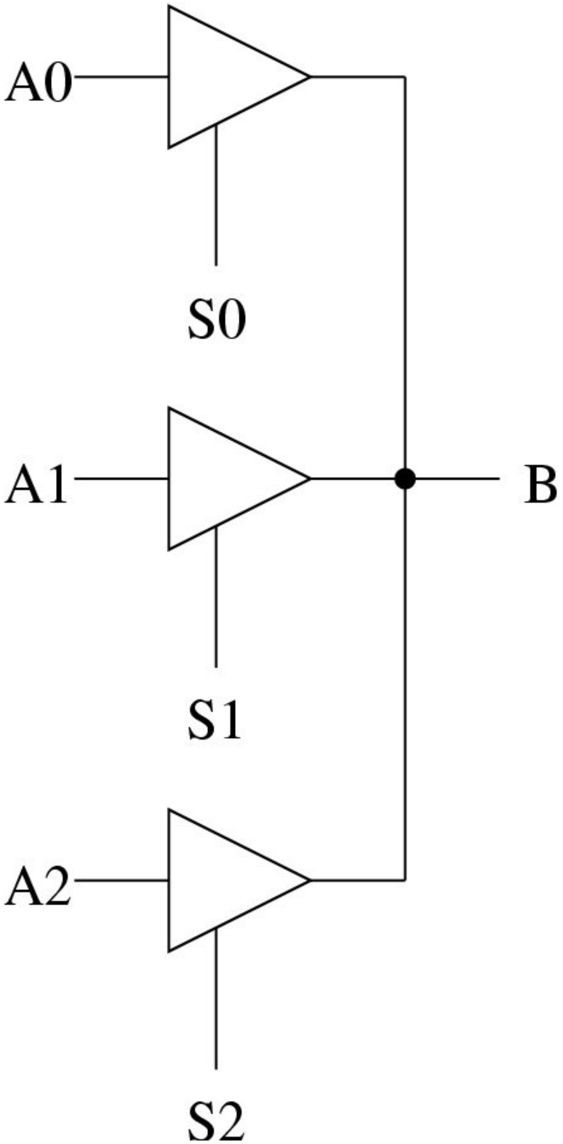
S	A		B
0	0		floating
0	1		floating
1	0		0
1	1		1



How are tristate buffers useful?

Tristate Buffers

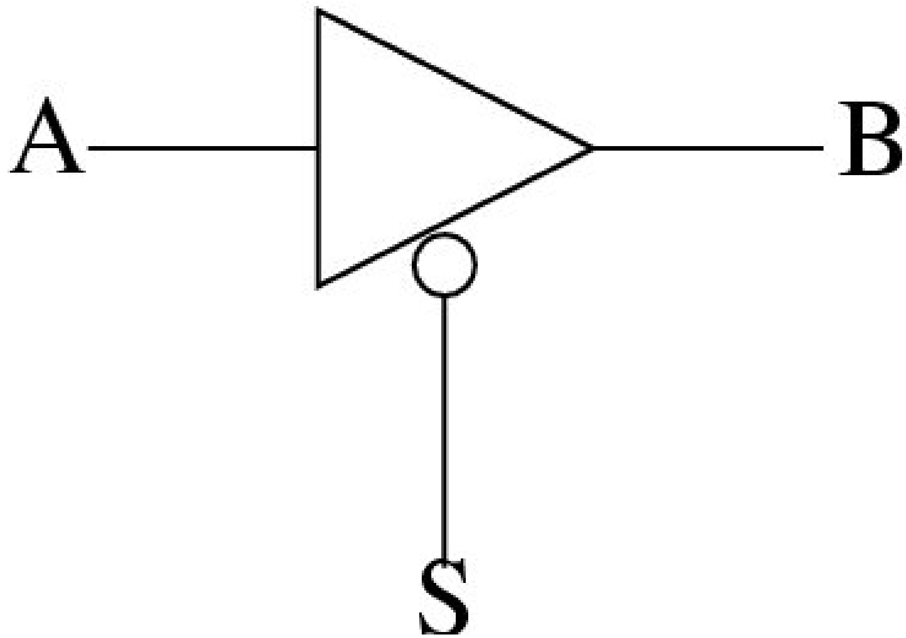
We can wire the outputs of multiple tristate buffers together without any other logic



Tristate Buffers

- We must guarantee that only one select line is active at any one time
- Tristate buffers will turn out to be useful when we start building data and address buses

Another Tristate Buffer



What does the truth table look like?

Another Tristate Buffer

S	A		B
0	0		0
0	1		1
1	0		floating
1	1		floating