Last Time

Analog Circuits Review

- Voltage
- Amperage
- Resistance
- Capacitance

Today

- A bit more on analog circuits
  - RC circuits
  - Transistors
- Analog to digital circuits
  - Transistors to basic logic gates
  - Introduction to Boolean Algebra
  - Connecting digital circuits to Boolean Algebra

Administrivia

Homework 1 will be delayed by a lecture

Encoding Information

In your ‘circuits and sensors’ class: how did you encode information?

- e.g., the acceleration measured by your accelerometer?
- or the rate of bend of a piezoelectric device?

Encoding Information

- Acceleration (or bend rate) is encoded in the voltage that is output from the circuit
- As acceleration increases, the voltage also increases

Encoding Information

- Acceleration (or bend rate) is encoded in the voltage that is output from the circuit
- As acceleration increases, the voltage also increases

- We say that this is an **analog** or **continuous** encoding of the information

Analog Encoding

What is the problem with analog encoding?

Analog Encoding

What is the problem with analog encoding?

- Small injections of noise – either in the sensor itself or from external sources – will affect this analog signal
- This leads to errors in how we interpret the sensory data

How do we fix this?

Digital Encoding

How do we fix this?

- At any instant, a single signal encodes one of two values:
  - A voltage around 0 (zero) Volts is interpreted as one value
  - A voltage around +5 V is interpreted as another value

Binary Encoding

- Binary digits can have one of two values: 0 or 1
- We call 0V a binary “0” (or FALSE)
- And +5V a binary “1” (or TRUE)

Binary Encoding

- Exactly what these levels are depends on the technology that is used (it is common now to see +1.8V as a binary 1 in low-power processors)

- This encoding is much less sensitive to noise: small changes in voltage do not affect how we interpret the signal

Transistors

What do transistors do for us?

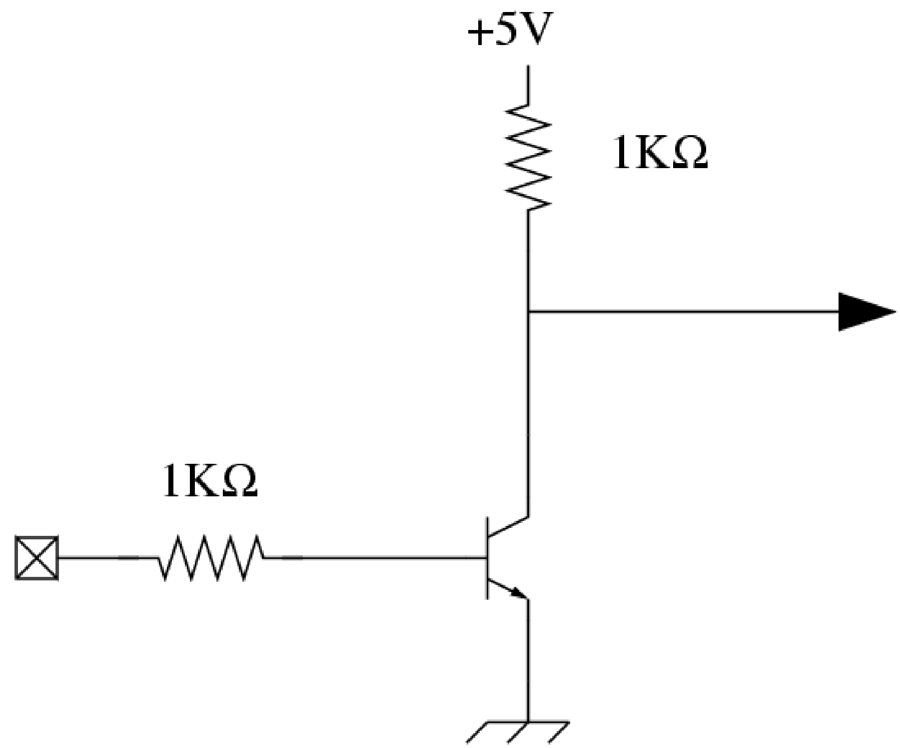
- In general: they act as current amplifiers

- But: we can use them as electronic switches to process digital signals

Transistors to Digital Processing

Consider the following circuit:

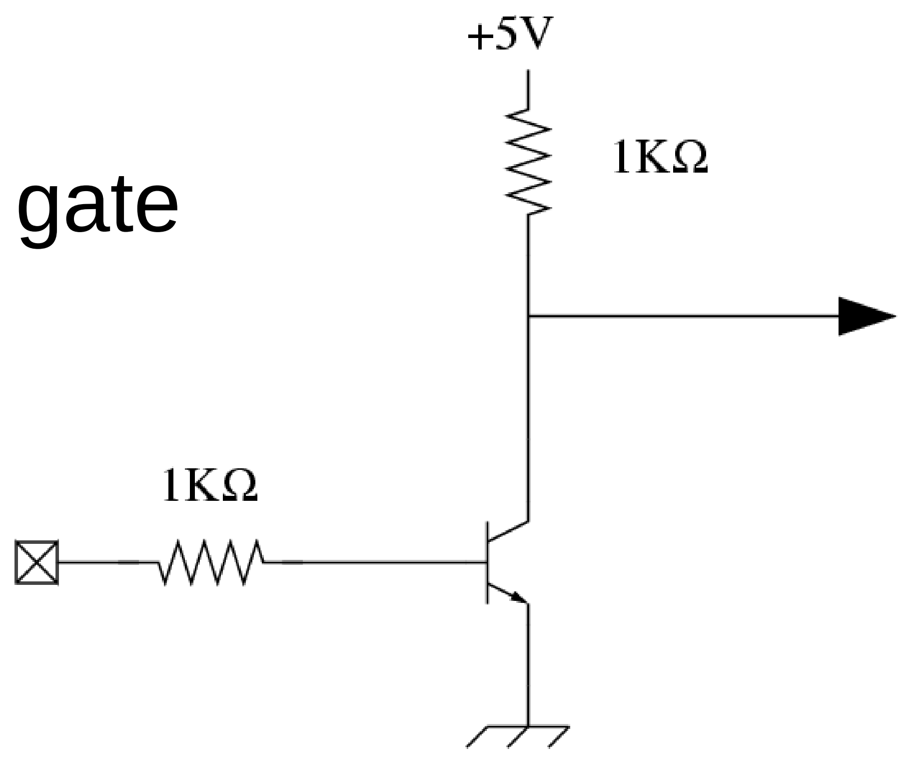
- What is the output given an input of 0V?
- An input of +5V?



Transistors to Digital Processing

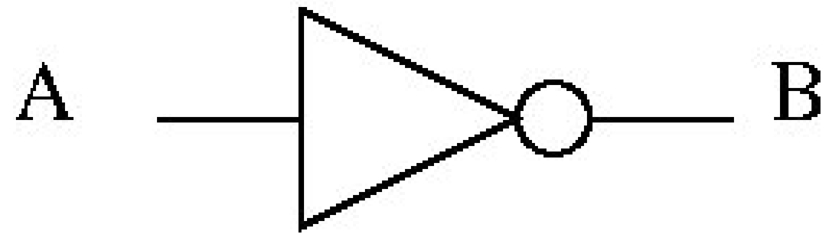
- Input: 0V -> Output +5V
- Input: +5V -> Output 0V

- We call this a "NOT" gate



The NOT Gate

- Logical Symbol:



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

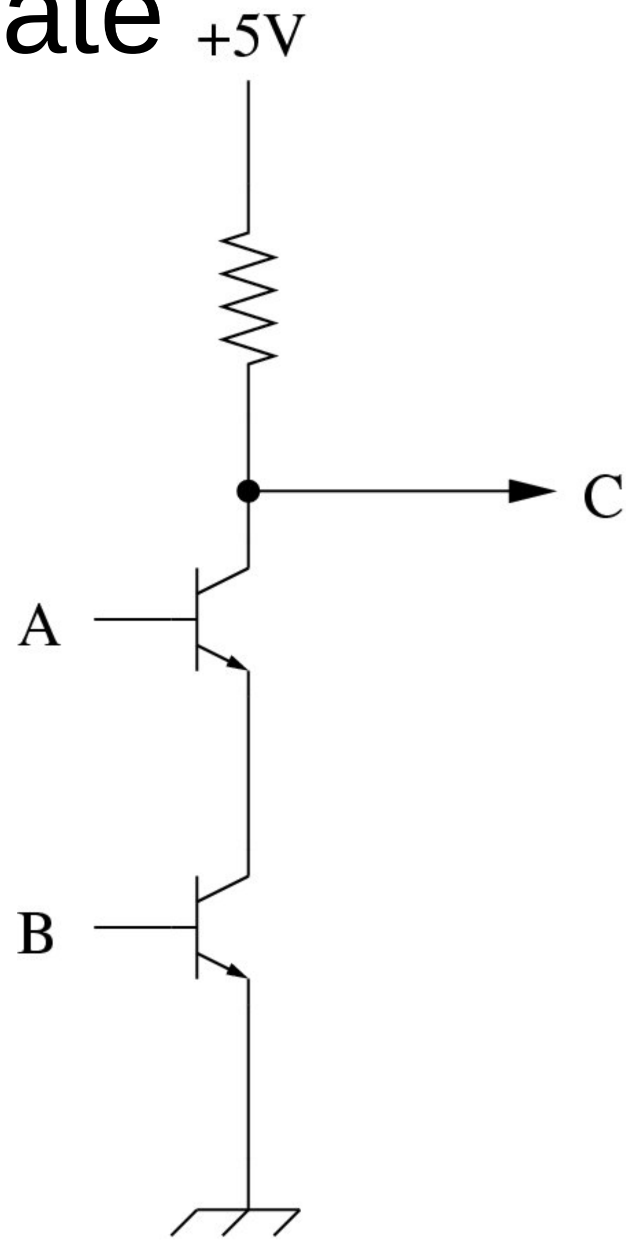
- Truth Table:

A	B
0	1
1	0

A Two-Input Gate

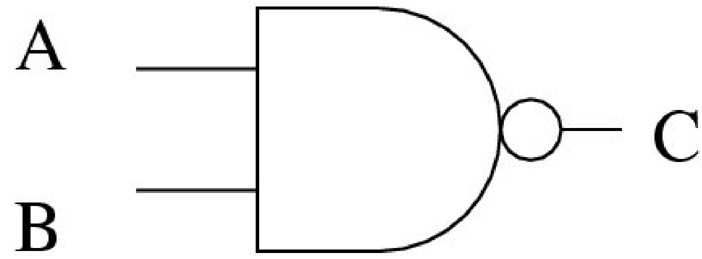
What does this circuit compute?

- A and B are inputs
- C is the output



The “NAND” Gate

- Logical Symbol:



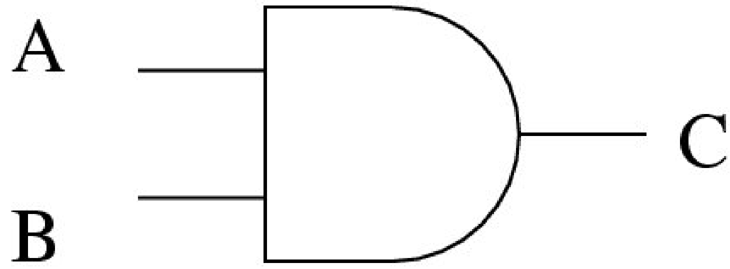
- Algebraic Notation: $C = \overline{A*B} = \overline{AB}$

- Truth Table:

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

The "AND" Gate

- Logical Symbol:



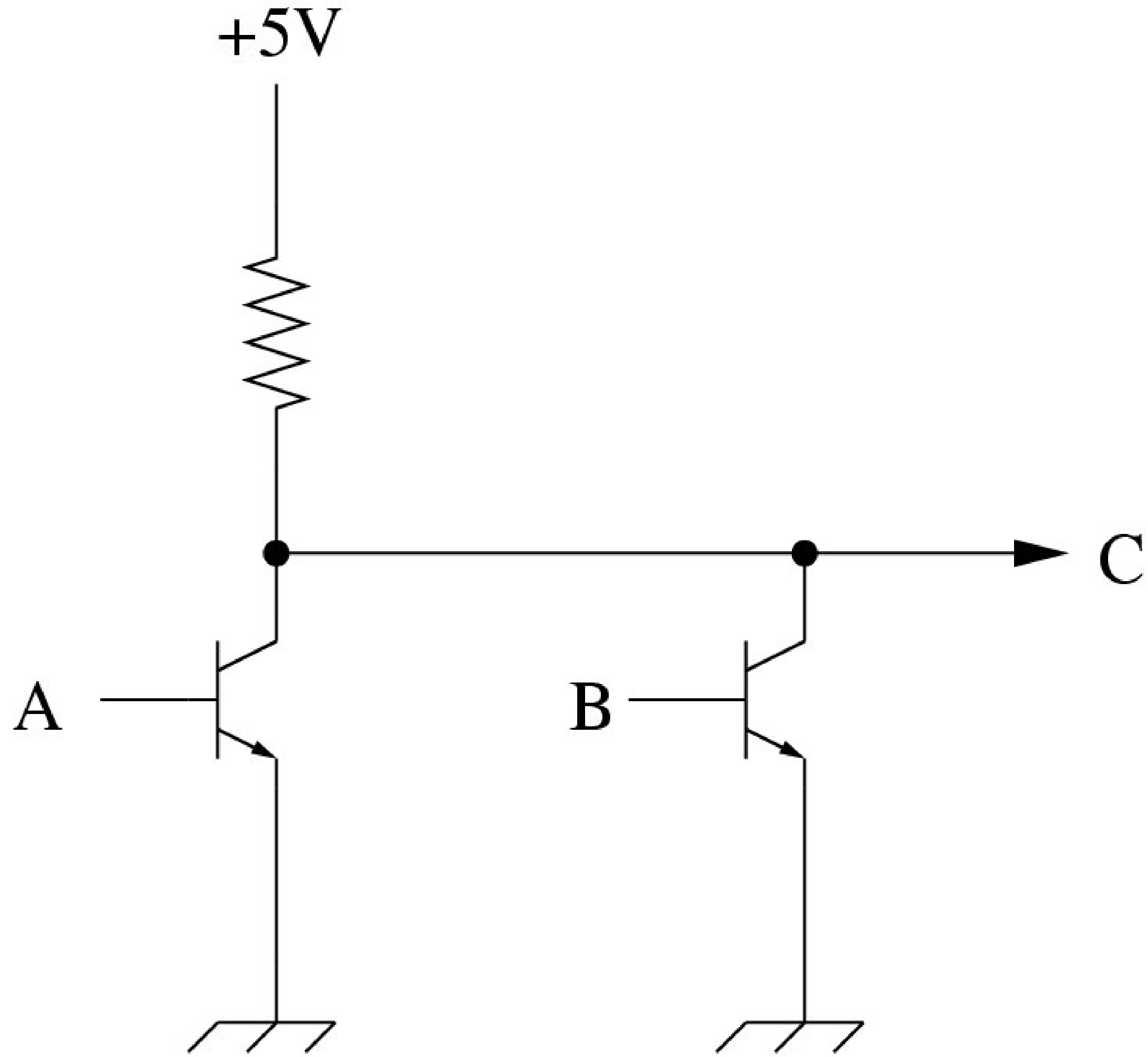
- Algebraic Notation: C = A*B = AB

- Truth Table:

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

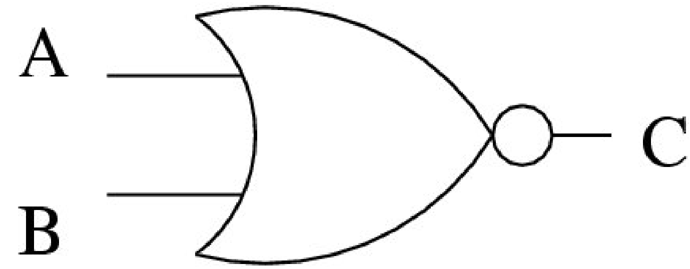
Yet Another Gate

What does this circuit compute?



The “NOR” Gate

- Logical Symbol:



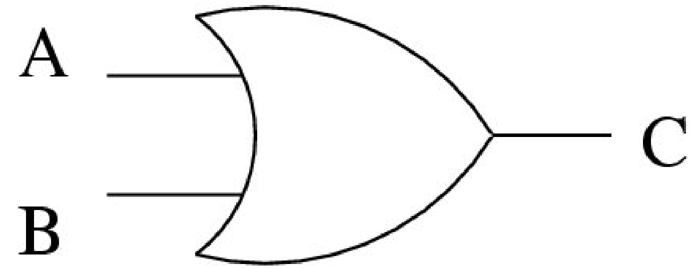
- Algebraic Notation: $C = \overline{A+B}$

- Truth Table:

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

The “OR” Gate

- Logical Symbol:



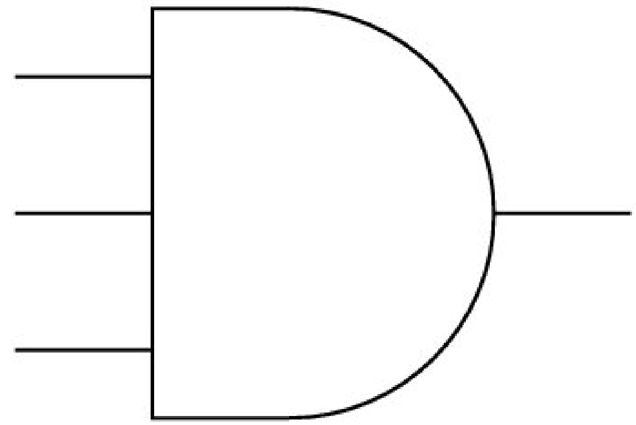
- Algebraic Notation: C = A+B

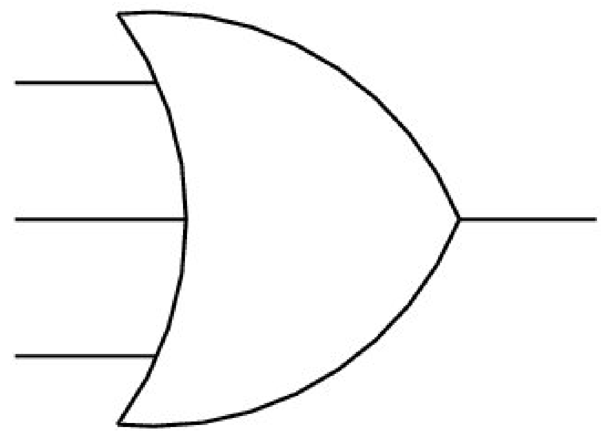
- Truth Table:

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)





Last Time

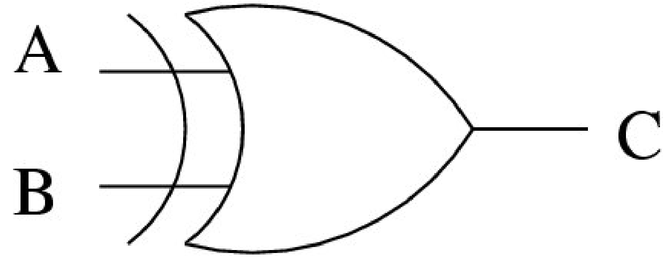
- Transistors to Logic Gates
- 3 ways of specifying logical computation:
  - Circuit (gates): AND, OR, NOT
  - Algebra
  - Truth Tables

Today

- Building more complicated logic circuits
- Algebraic manipulation and circuit reduction

Exclusive OR ("XOR") Gates

- Logical Symbol:



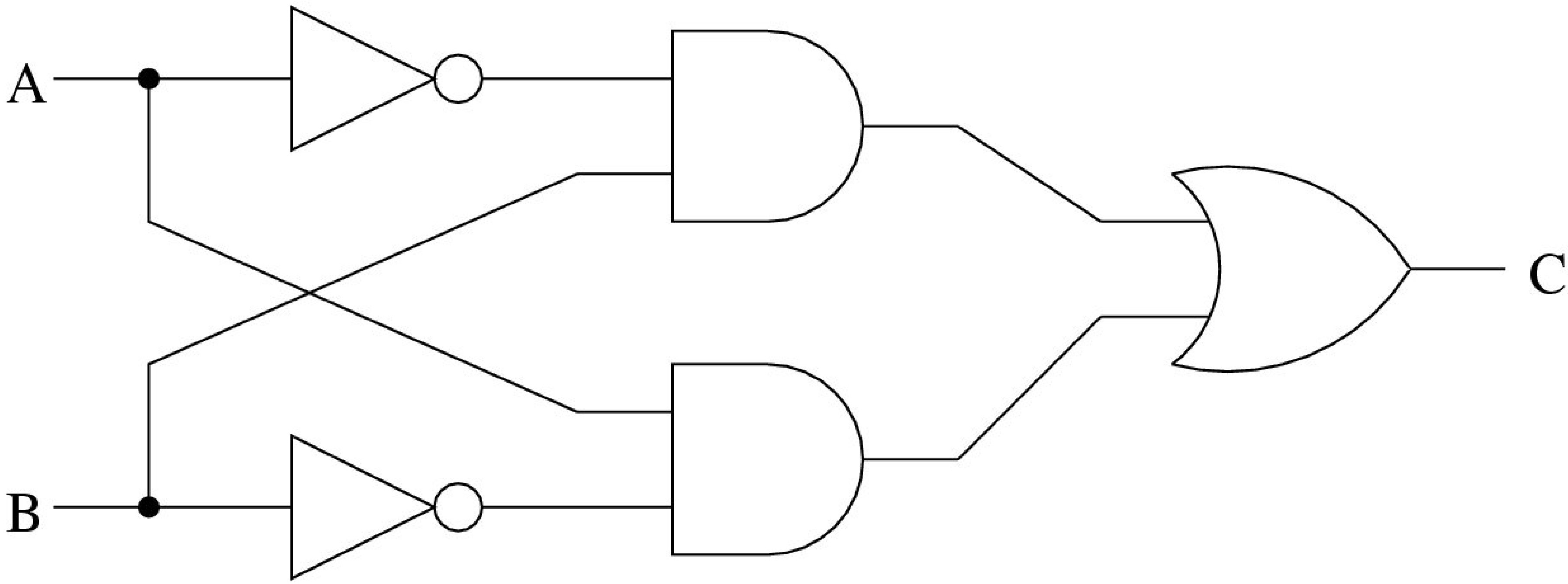
- Algebraic Notation: C = A⊕B

- Truth Table:

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

How would we implement this with AND/OR/NOT gates?

An XOR Implementation



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?

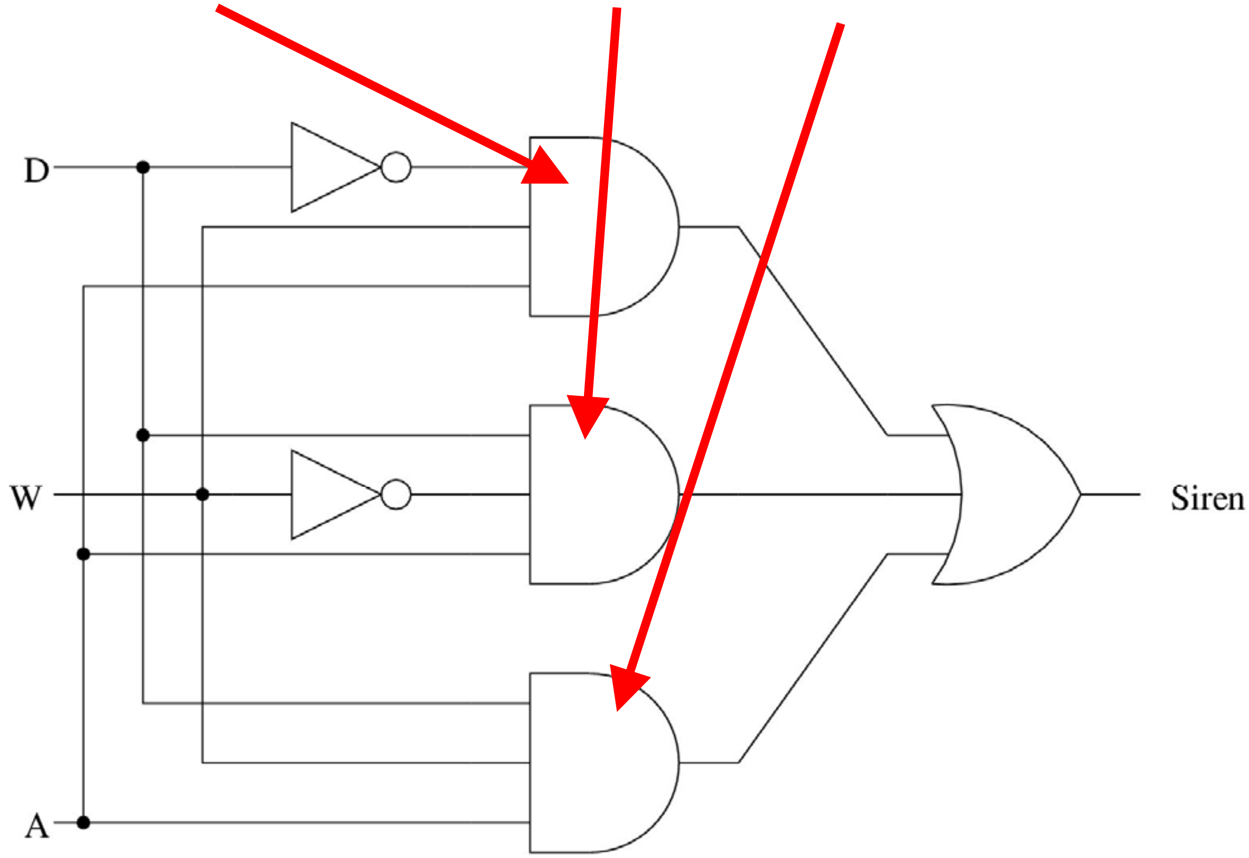
Alarm Example: Truth Table

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: Circuit

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



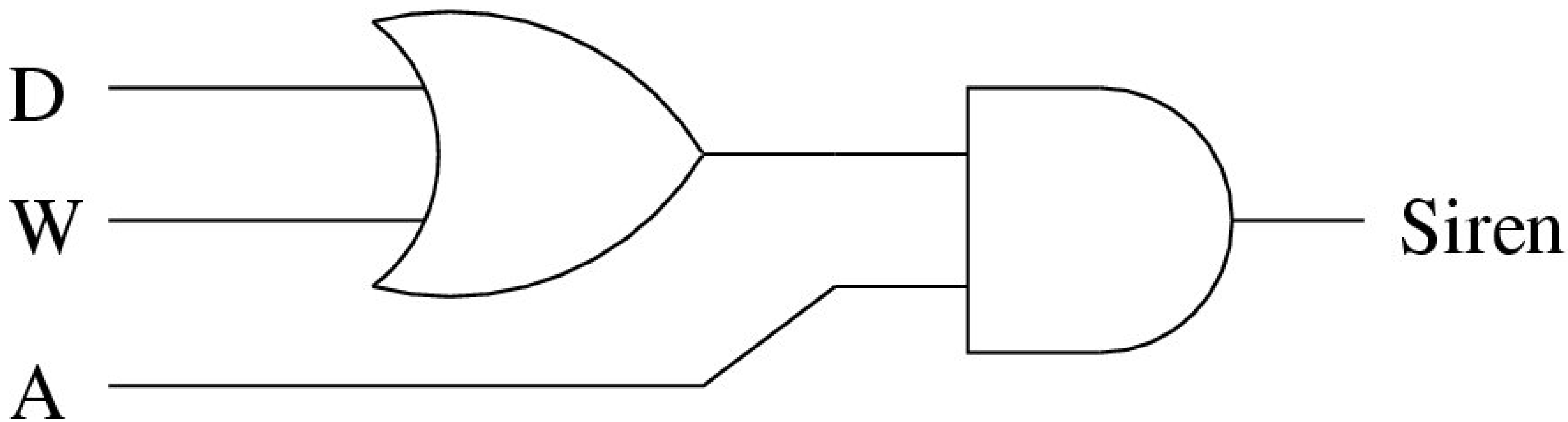
Alarm Example: Circuit

Is a simpler circuit possible?

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

Alarm: An Alternative Circuit



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”

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

Minterms

- AND containing all function inputs
  – in our example these are D, W, A
- Some of the inputs may be “NOT’ed”

- Called a “minterm” because the AND is
  – 1 for exactly one case, and
  – 0 otherwise

Minterm Approach to Representing a Truth Table/Function

- OR together a set of minterms:
  - One minterm for each row for which the output is 1

- Example:

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

- Circuit is correct, but may not be smallest

Boolean Algebra

- There are exactly two numbers in Boolean System: “0” and “1”
- You are already familiar with the “integers”: {… -2, -1, 0, 1, 2, …}

  (and Integer Algebra)

Boolean Algebra

- Like the integers, Boolean Algebra has the following operators:

		Integers	Boolean
+		addition	OR
*		product	AND
inverse		negation	NOT

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):

$A + B = B + A$

$A * B = B * A$

DeMorgan’s Laws

(A * B)’ = A’ + B’

How do we convince ourselves that this is true?

Proof by Truth Table

A	B		(A * B)’	A’ + B’
0	0		1	1
0	1		1	1
1	0		1	1
1	1		0	0

NOTE: change in the NOT notation

DeMorgan's Laws (cont)

$(A + B)' = A' * B'$

Proof by Truth Table

A	B		(A + B)'	A' * B'
0	0		1	1
0	1		0	0
1	0		0	0
1	1		0	0

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

D’ W A
+
D W’ A
+
D W A

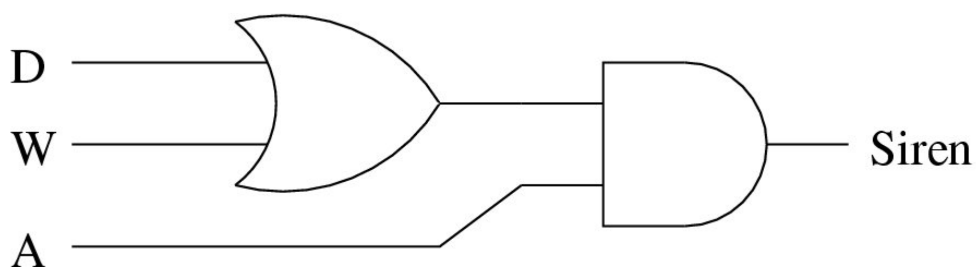
Reduction with Algebra

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 the same circuit as before!



Last Time

- Truth Tables
- Basic logic gates: AND, OR, NOT
- Introduction to Boolean Algebra
  - Minterms
  - Precedence: AND before OR
  - Laws: Commutative, Associative, and Distributive
  - Identities
  - DeMorgan's Laws
- Boolean Algebra to Circuits (and back)

Circuit Design Process

- Start with a truth table
- Convert to “minterms” algebraic representation
- Simplify using Boolean Algebra
- Translate into circuit diagram

Administrivia

- Homework 1 due in one week

Today

- Multiplexers
- Demultiplexers
- Tri-state Buffers

- Circuit design practice

Multiple Output Variables

Suppose we have a function with multiple output variables?

- How do we handle this?

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

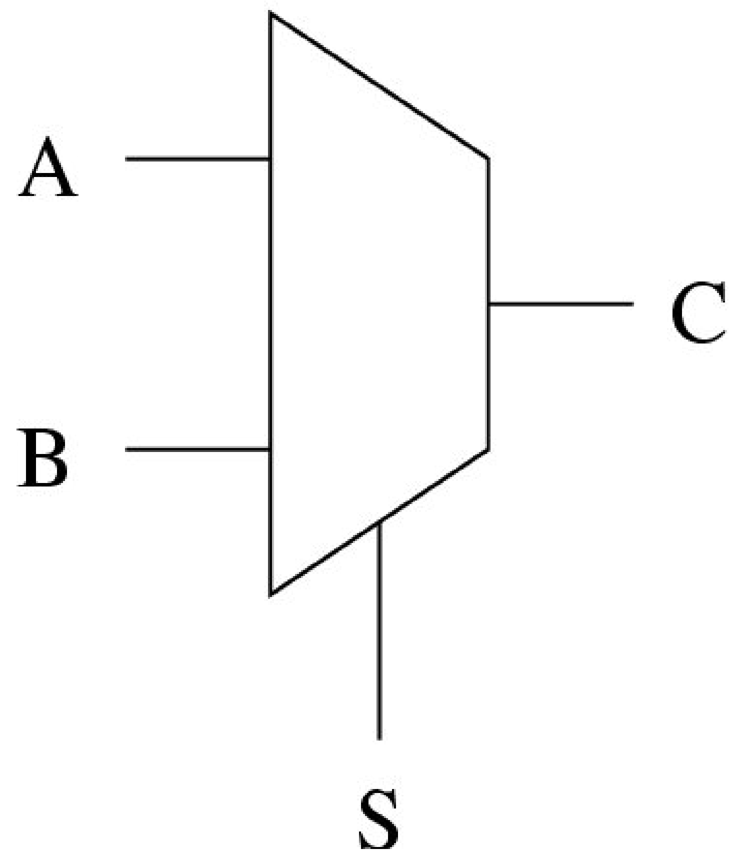
More Logical Components

- Multiplexer
- Demultiplexer
- Tristate buffer

2-Input Multiplexer

A multiplexer is a device that selects between two input lines

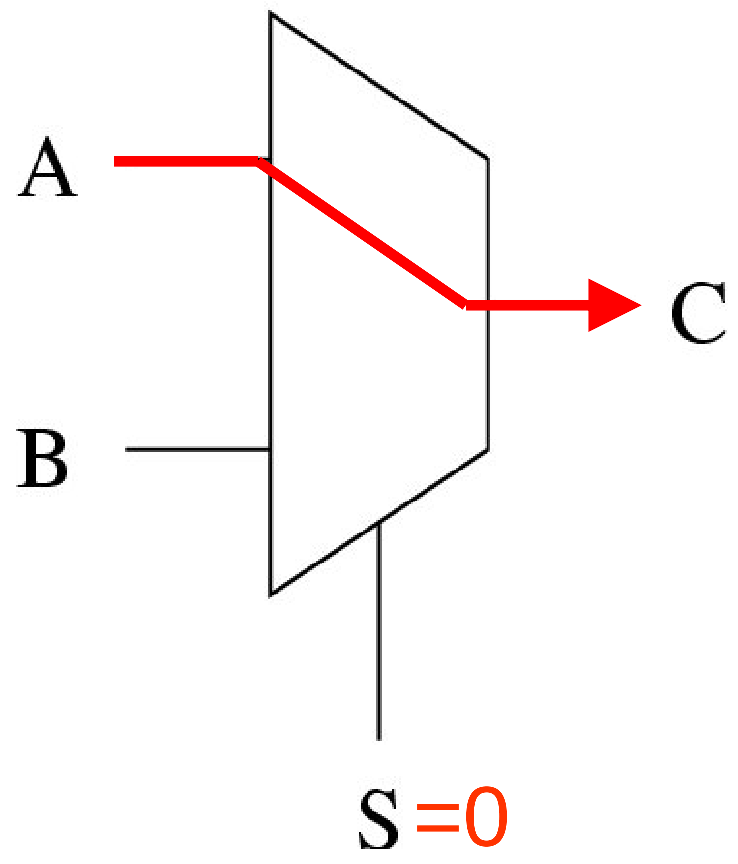
- A & B are the inputs
- S is the selection signal (also an input)
- C is a copy of A if S=0
- C is a copy of B if S=1



2-Input Multiplexer

A multiplexer is a device that selects between two input lines

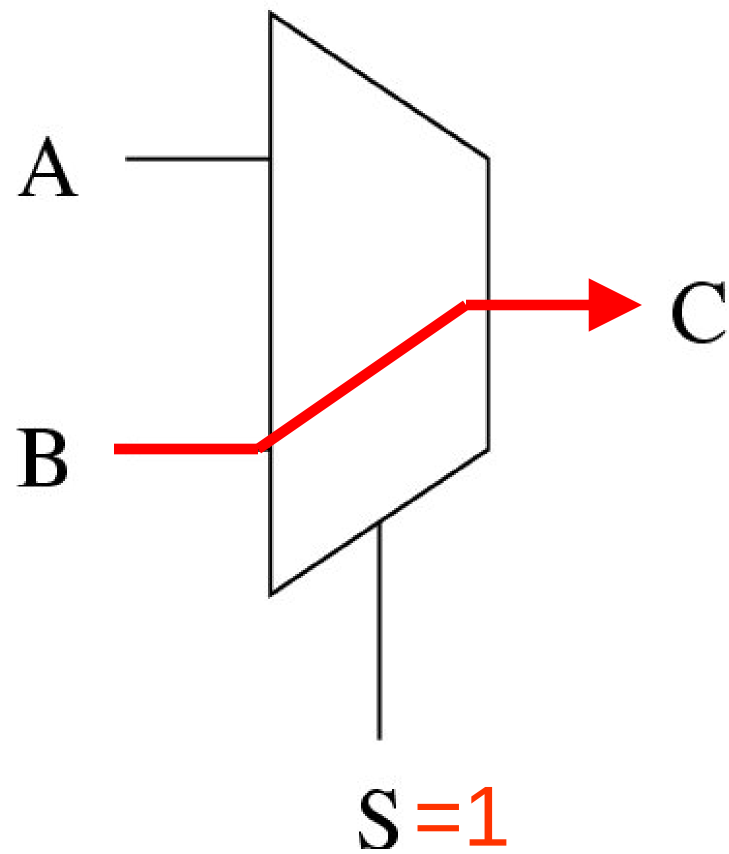
- A & B are the inputs
- S is the selection signal (also an input)
- C is a copy of A if S=0
- C is a copy of B if S=1



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

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

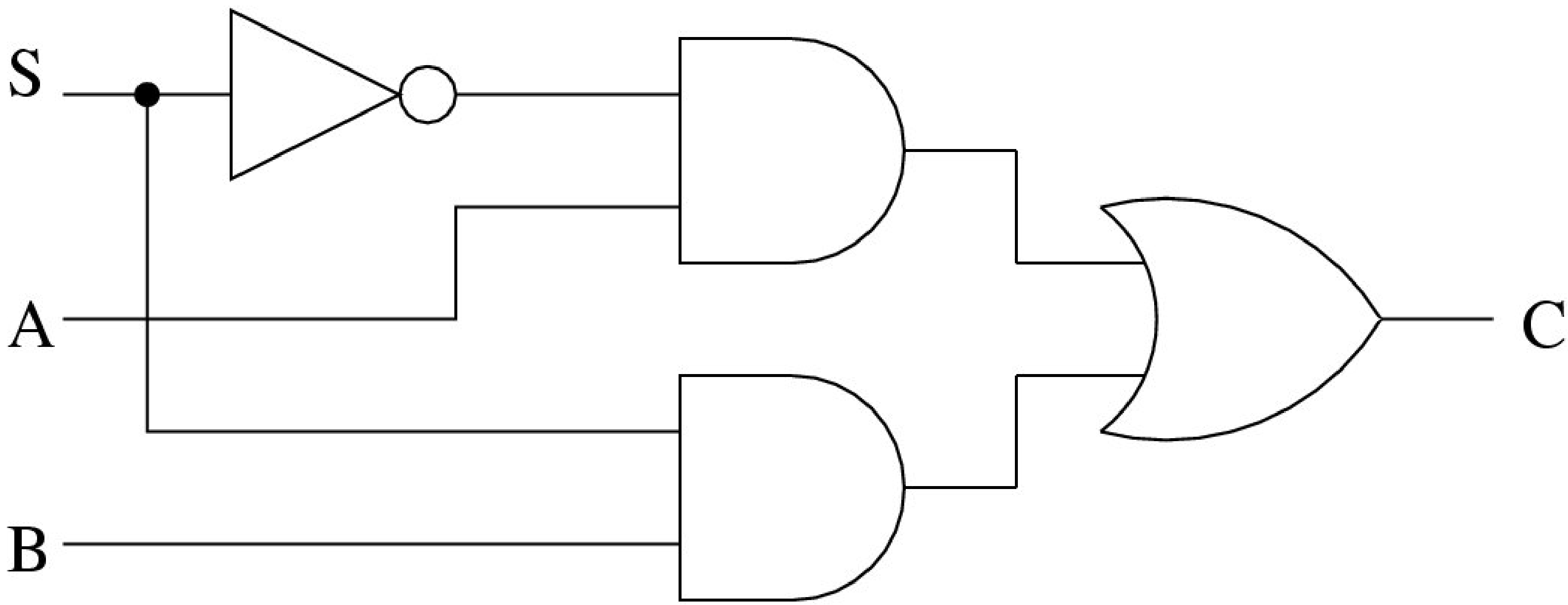
Is there a simpler expression?

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

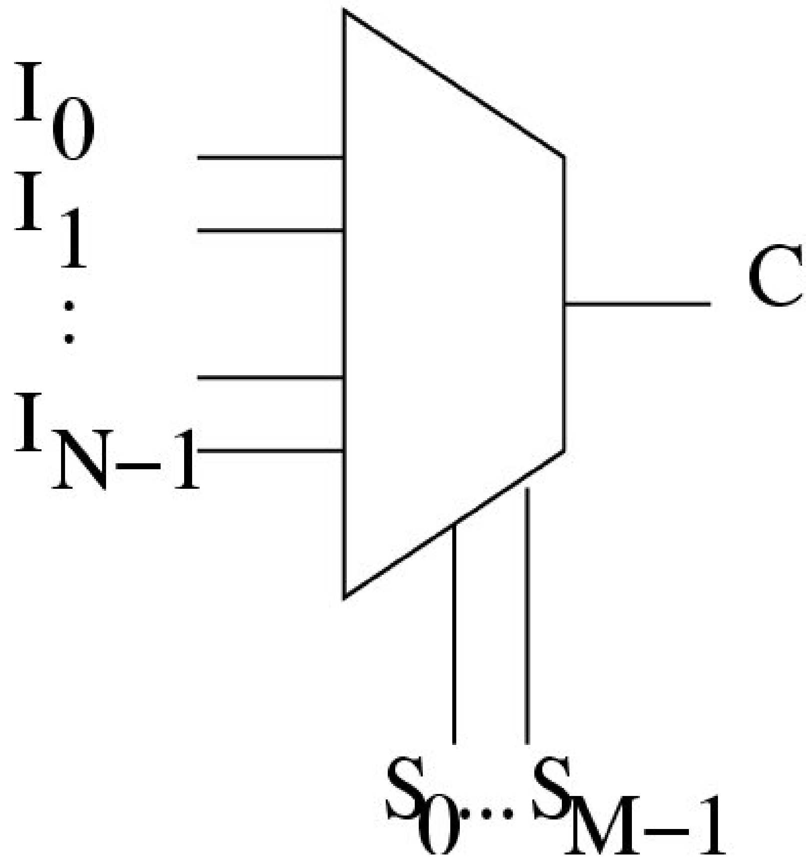
C = S’A + SB



N-Input Multiplexer

Suppose we want to select from between N different inputs.

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



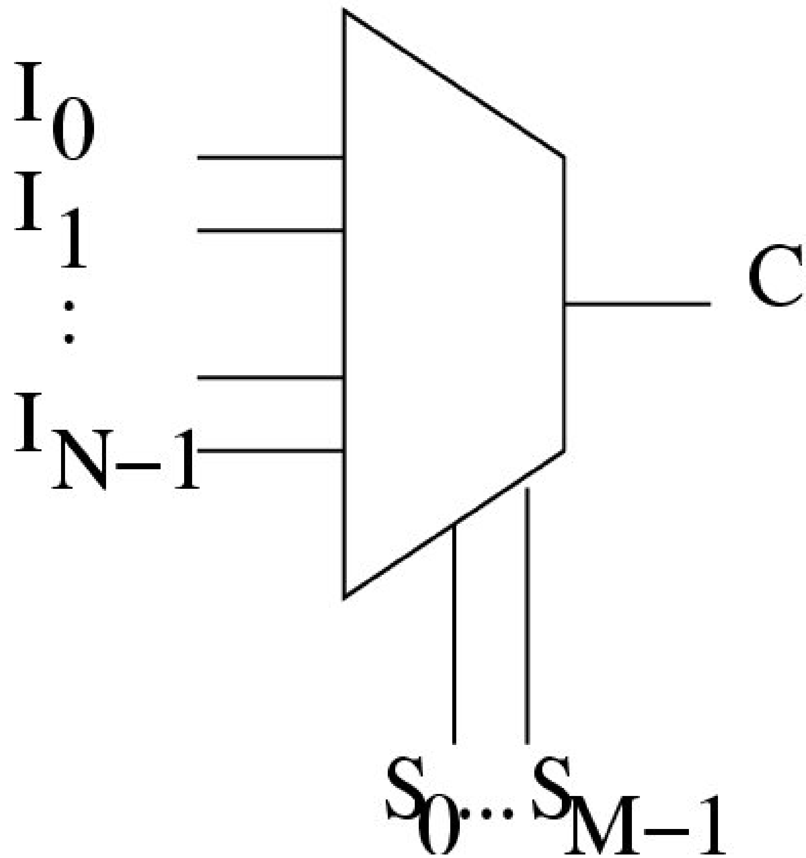
N-Input Multiplexer

How many select lines?

- $M = \log_2 N$

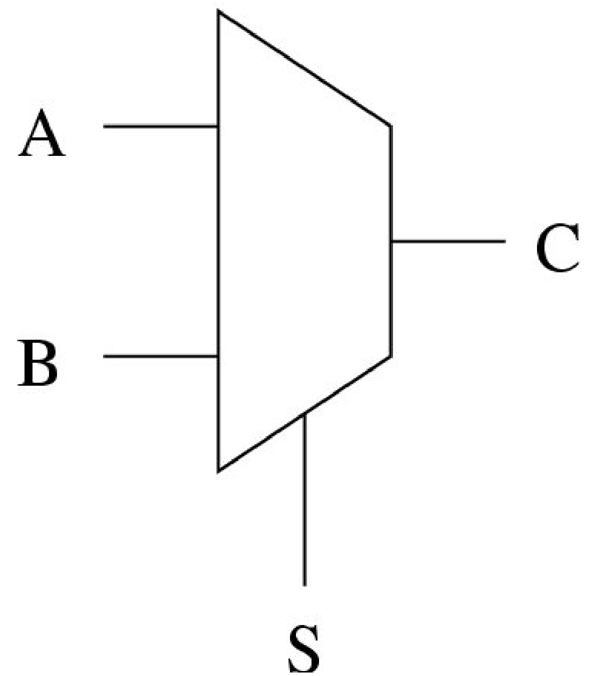
  or
- $N = 2^M$

What would the N=8 implementation look like?



More Complicated Multiplexers

- Assume that you have a simple multiplexer (1 select line / 2 inputs)
- How do you create a multiplexer with 2 select lines and 4 inputs?



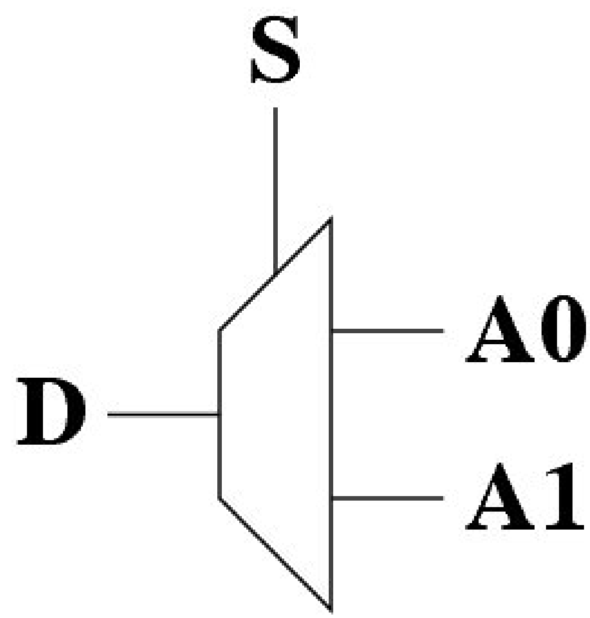
8-Input Multiplexer Implementation

$C = I_0S'_2S'_1S'_0 + I_1S'_2S'_1S_0 + I_2S'_2S_1S'_0 + I_3S'_2S_1S_0 + I_4S_2S'_1S'_0 + I_5S_2S'_1S_0 + I_6S_2S_1S'_0 + I_7S_2S_1S_0$

Note that we have one of each possible select line combination (or addressing terms)

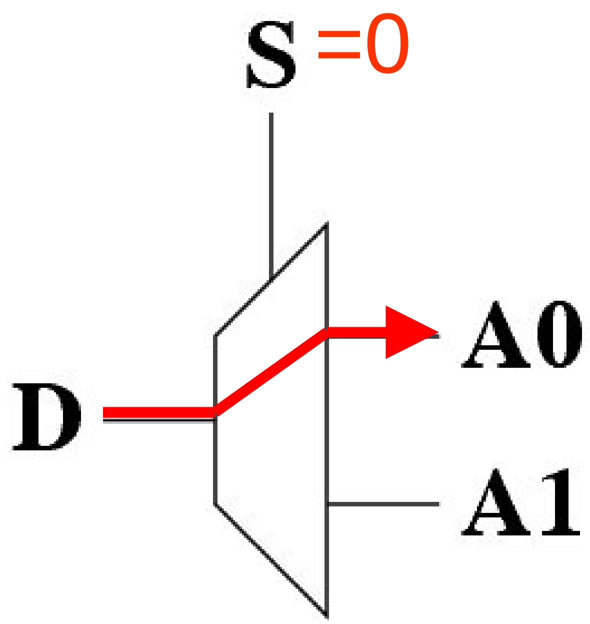
Demultiplexer

- 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



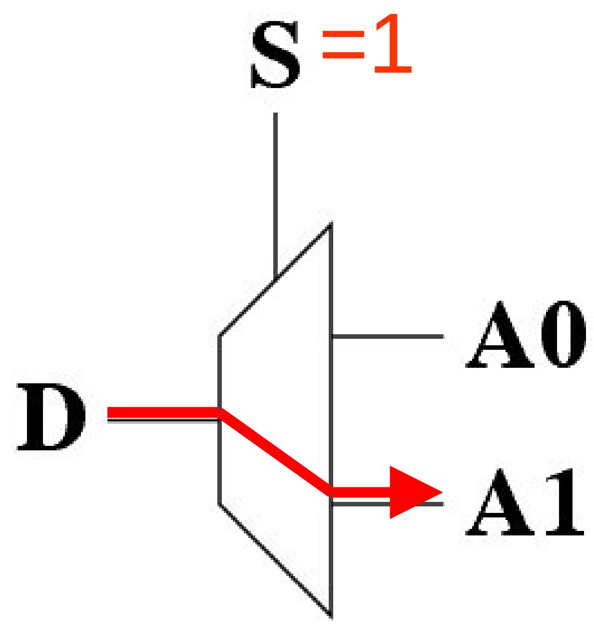
Demultiplexer

- 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



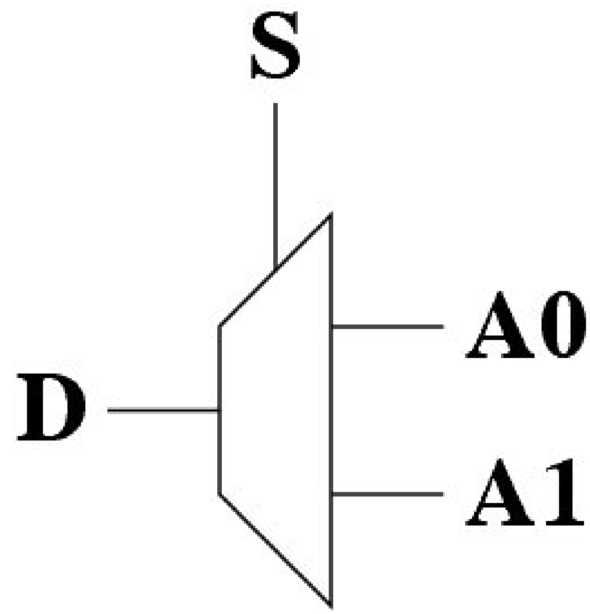
Demultiplexer

- 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



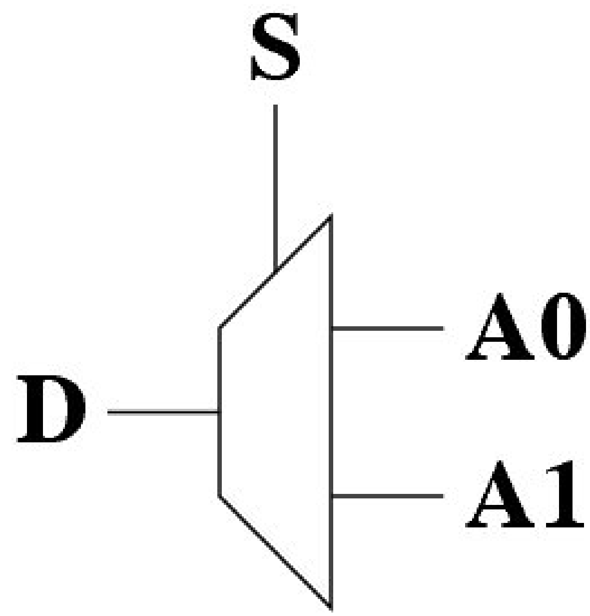
Demultiplexer Notes

- 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



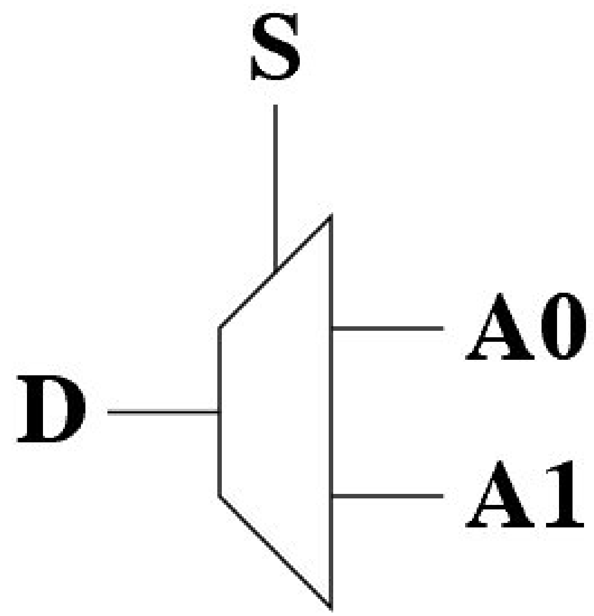
Demultiplexer Truth Table

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



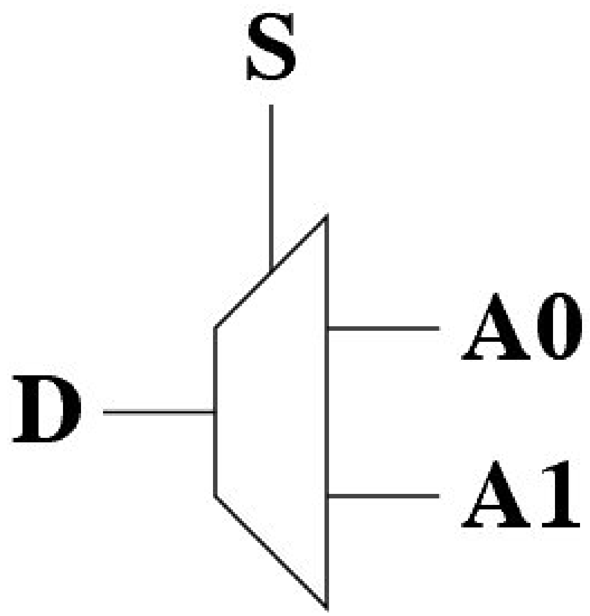
Demultiplexer Truth Table

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



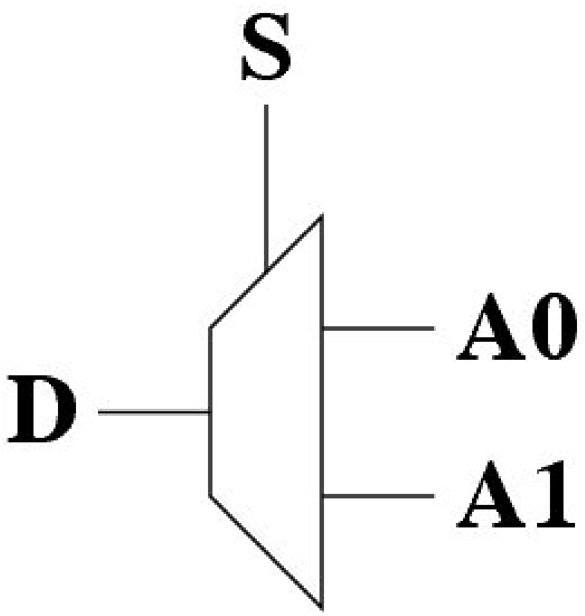
Demultiplexer Algebraic Specification

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

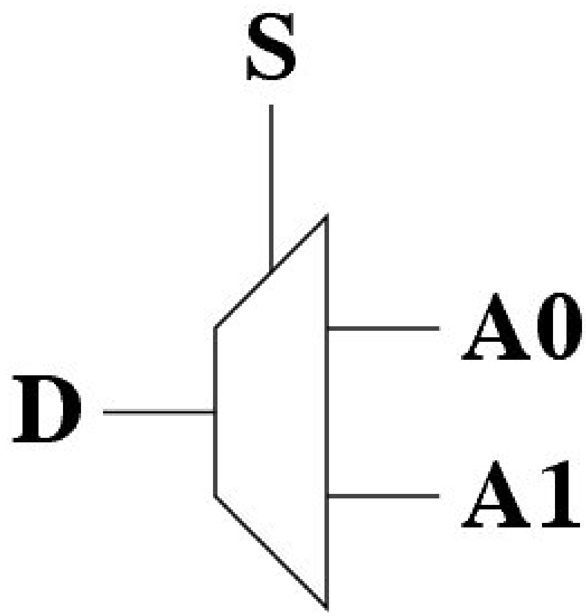
S	D		A_1	A_0
0	0		0	0
0	1		0	1
1	0		0	0
1	1		1	0



$A_0 = S'D$

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



$A_1 = SD$

$A_0 = S'D$

Demultiplexer Circuit

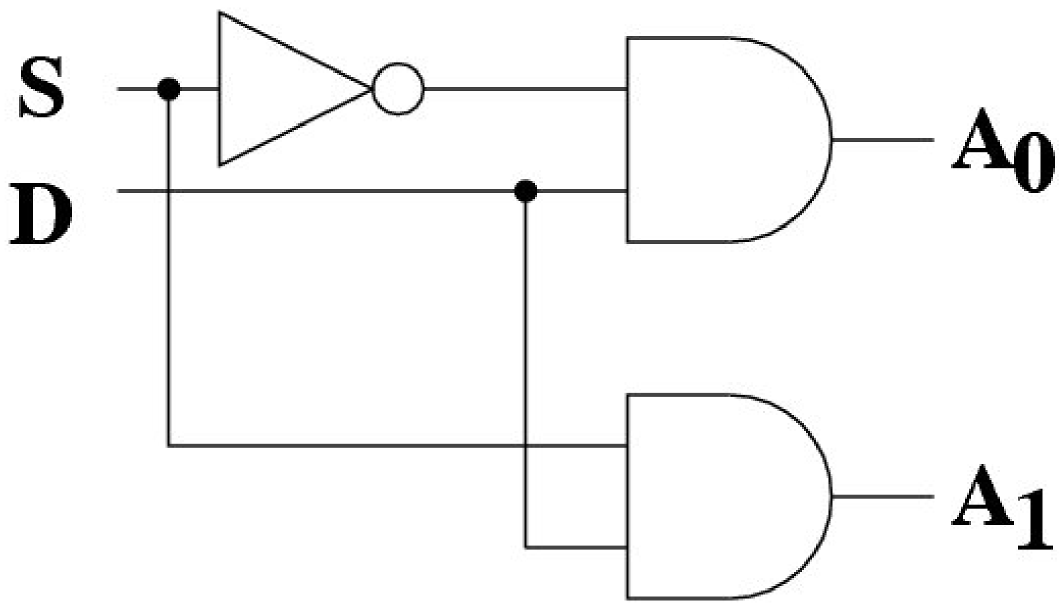
$A_0 = S'D$

$A_1 = SD$

Demultiplexer Circuit

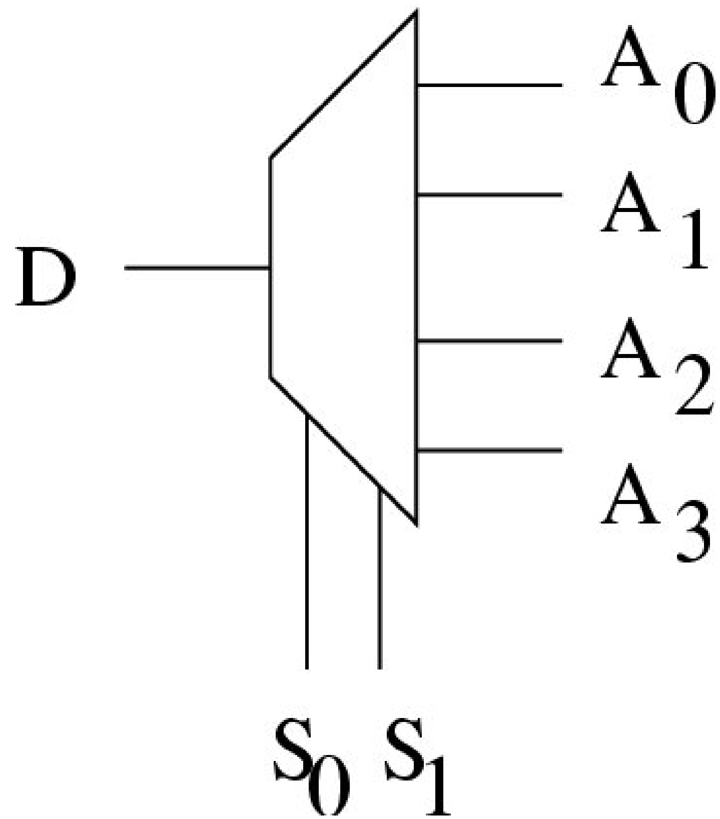
$A_0 = S'D$

$A_1 = SD$



2-Input Demultiplexer

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



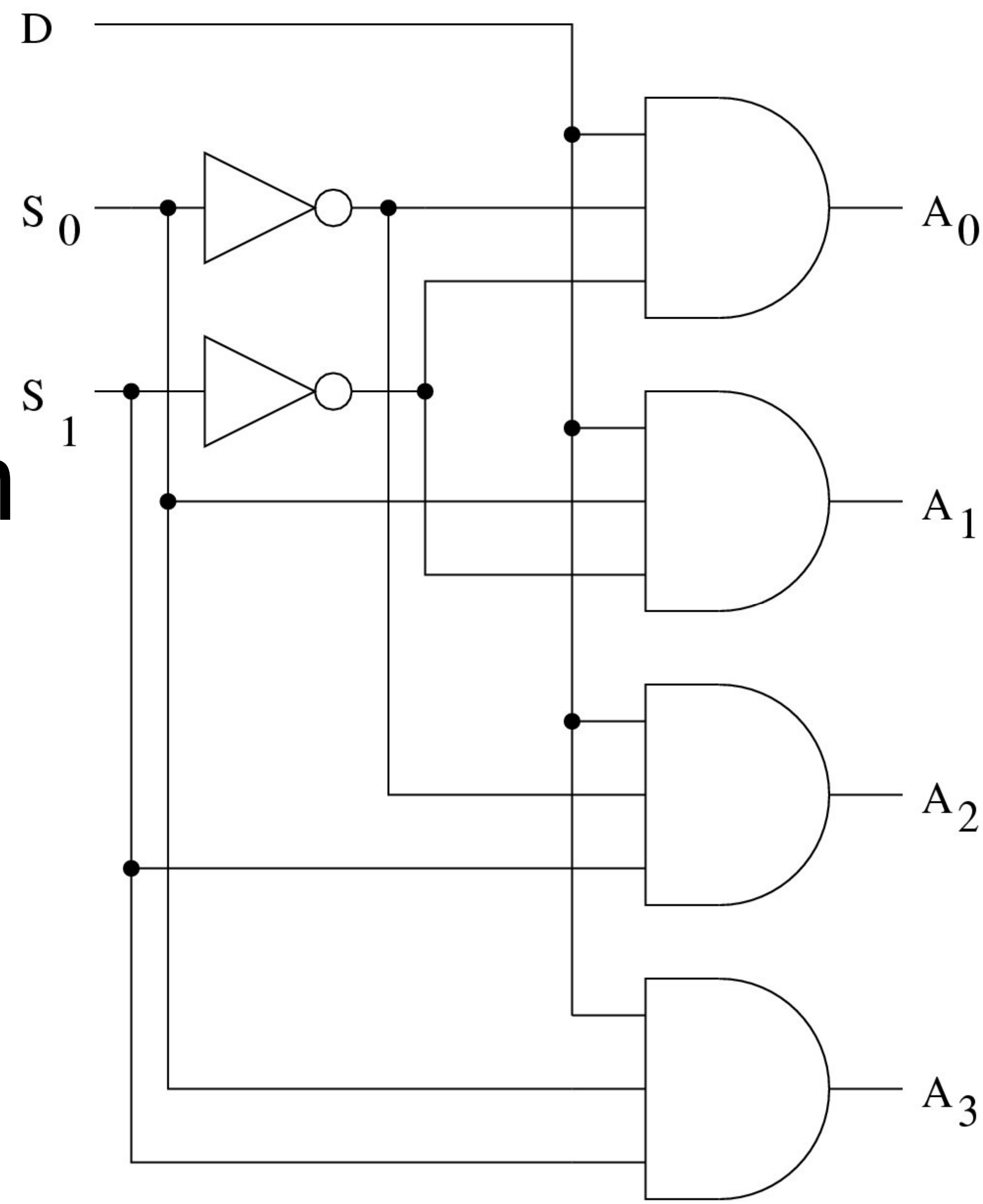
2-Input Demultiplexer Truth Table

S_1	S_0	D		A_3	A_2	A_1	A_0
0	0	0		0	0	0	0
0	0	1		0	0	0	1
0	1	0		0	0	0	0
0	1	1		0	0	1	0
1	0	0		0	0	0	0
1	0	1		0	1	0	0
1	1	0		0	0	0	0
1	1	1		1	0	0	0

Demultiplexer Implementation

- What does it look like?

Demultiplexer Implementation



Another Implementation

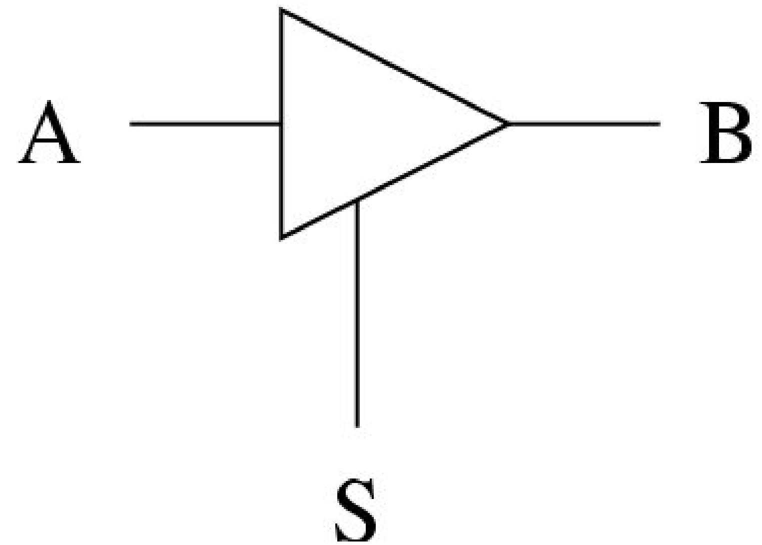
- Assume that you have a simple demultiplexer (1 select line / 2 outputs)
- How do you create a demultiplexer with 2 select lines and 4 outputs?

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)

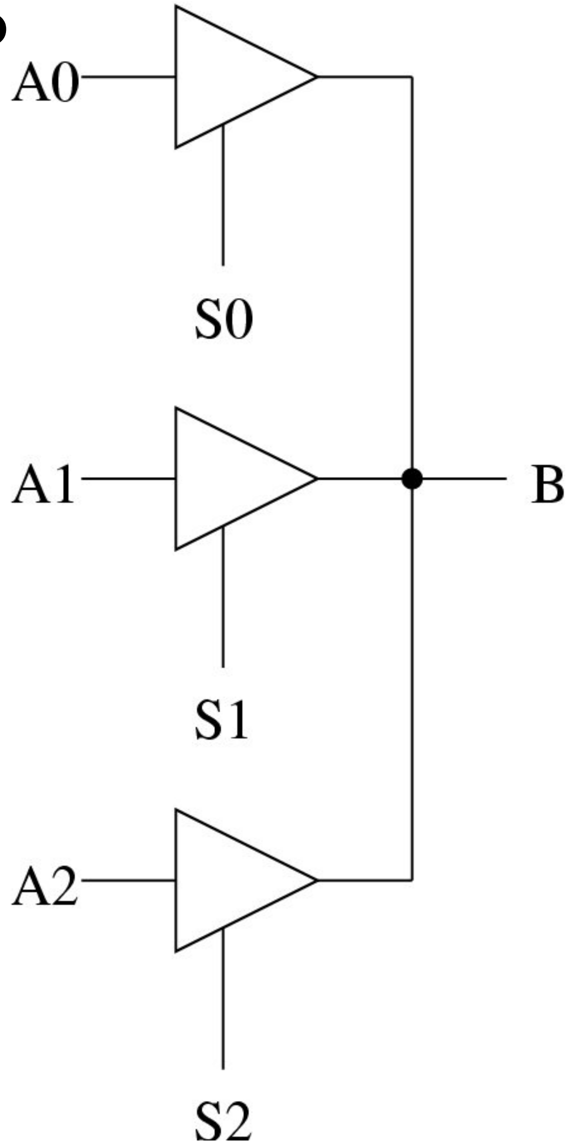
Tristate Buffers

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



How are tristate buffers useful?

Tristate Buffers



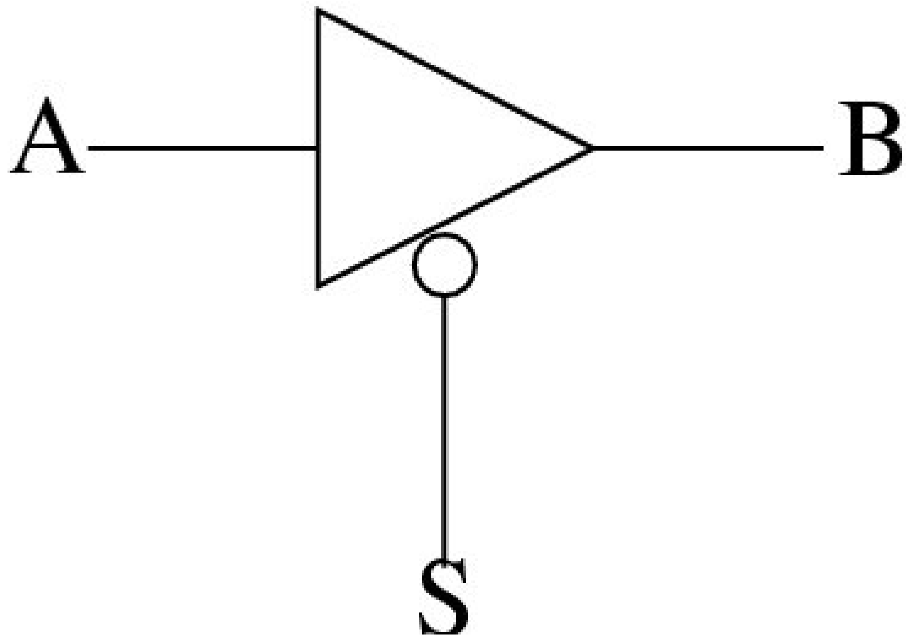
We can wire the outputs of multiple tristate buffers together without any other logic

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