Digital Logic Structures

Electrons

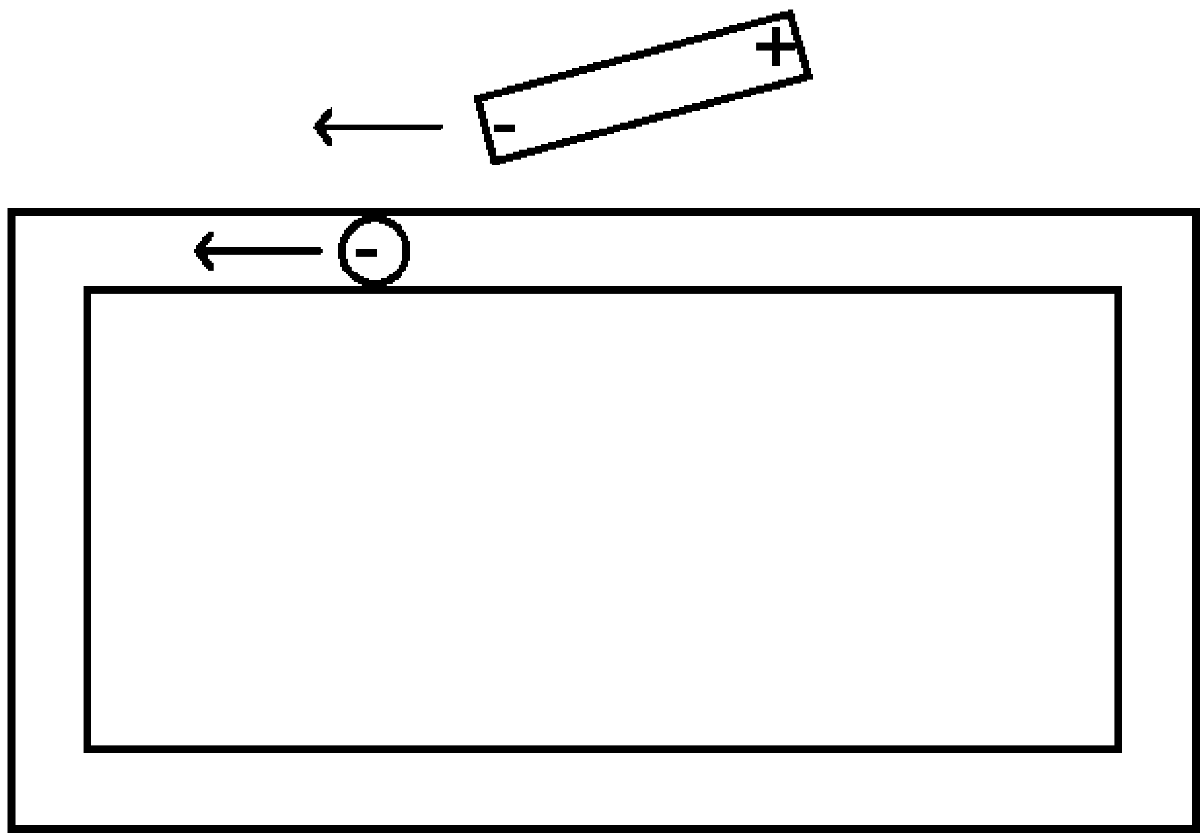
- Are tiny
- Are negatively charged
- Hurt
- Are the basis for our modern society

Circuits

- A circuit is a path which allows electrons to flow.
- A generator or battery provides the push to the electrons known as Voltage.
- Pushing the electrons doesn’t mean they will move.
- Electrons can only move if they have somewhere to move.
- Think of pushing a line of people against a closed door. The people wont move no matter how hard you push.
- Open the door and they all fall down.
- When electrons move that is called Current.

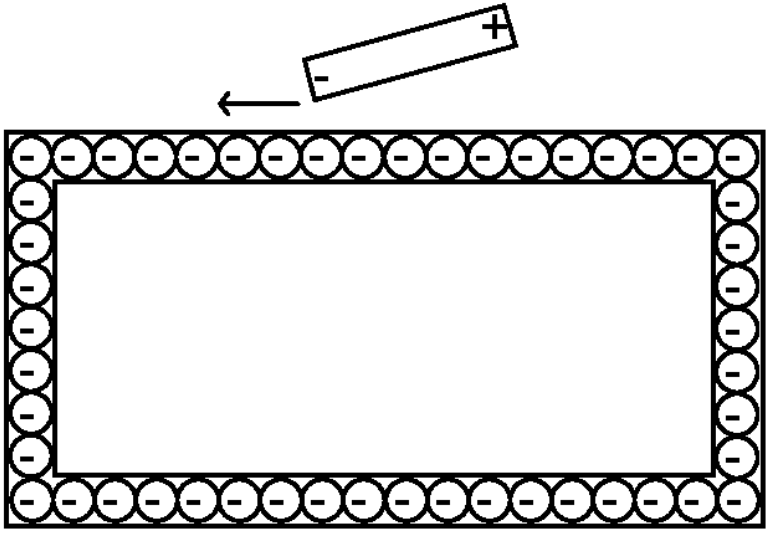
Magnets Move Electrons

- The magnet, the electron, and the wire.



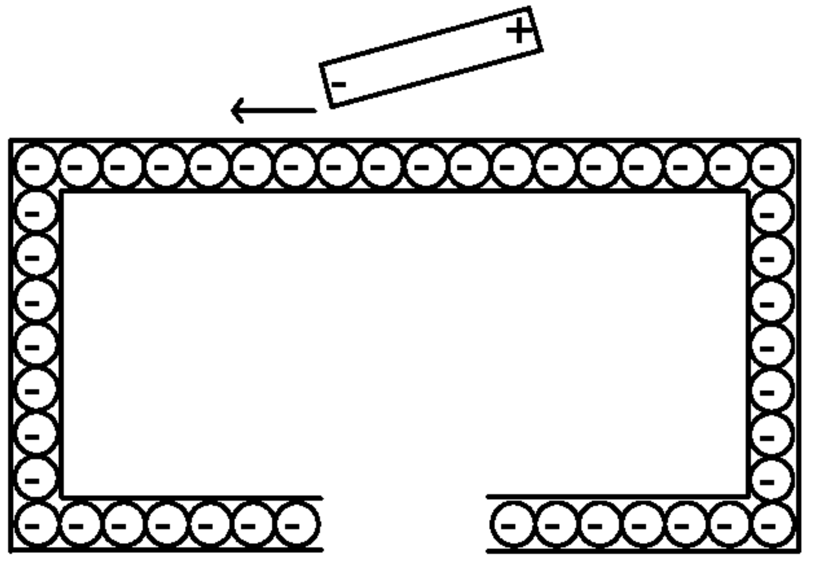
A wire is FULL of electrons.

- Will this work? This is a complete circuit.
- Think of it like a bicycle chain.
- The moving magnet provides pressure or voltage.



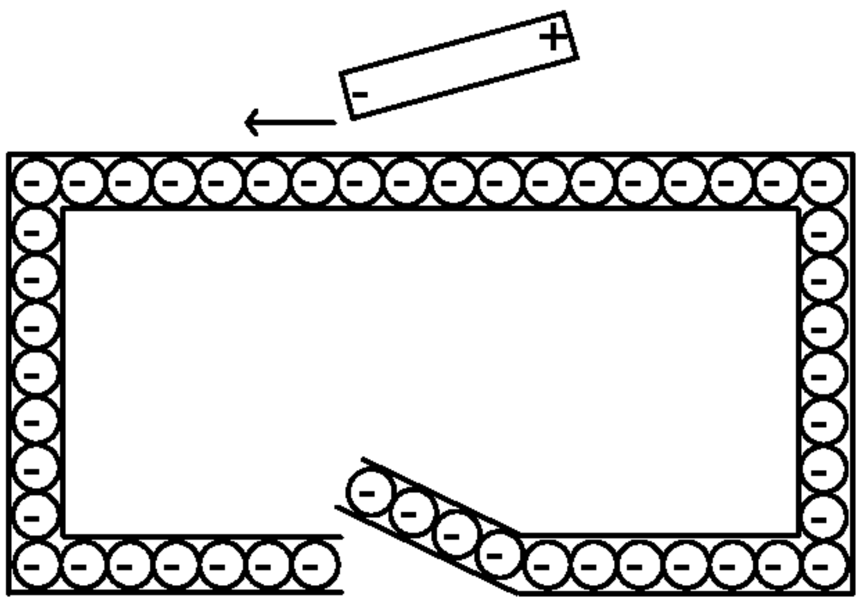
Open Circuit.

- Electrons can't leave the wire (normally).
- The pressure is still being applied (moving magnet).
- Voltage is present even if current is not flowing.



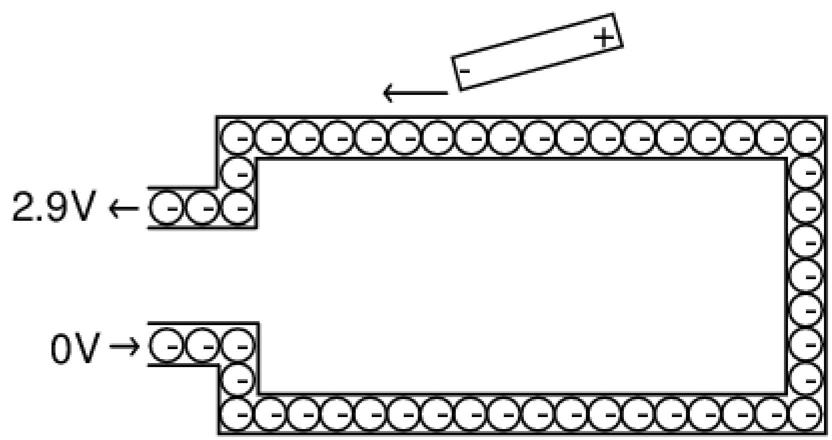
The Switch

- If the pressure (voltage) is there …
- closing the switch lets electrons flow (current).



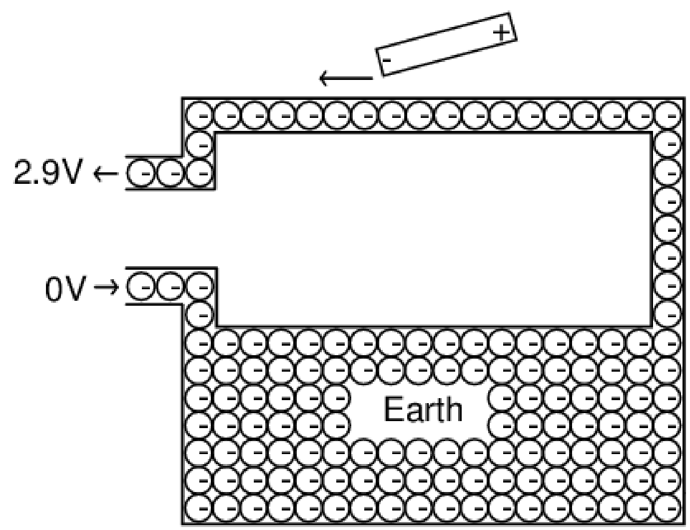
Source and Ground

- One side supplies electrons = 2.9V = 1
- One side provides path back = ground = 0



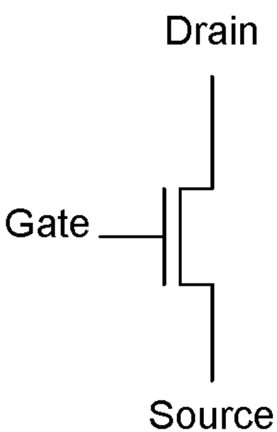
Why Ground?

- One side (the return side) is sometimes connected to the actual ground.
- The electrical system in your home is ALWAYS connected to earth this way.



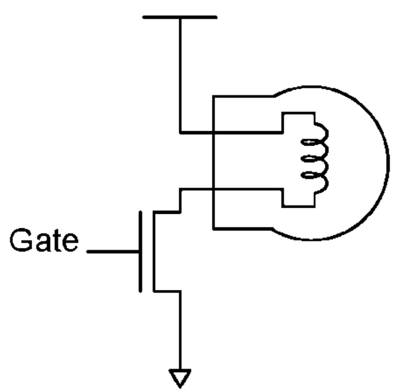
Transistors

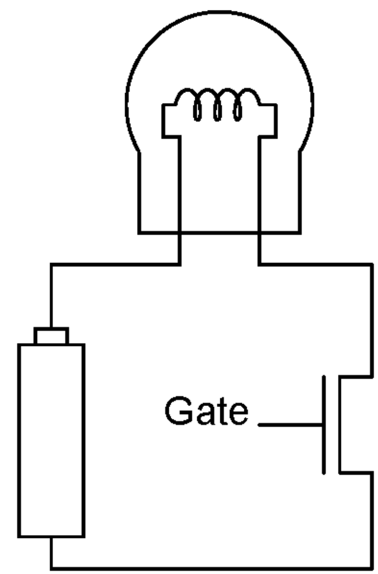
- A transistor is like an electronic switch.
- Metal Oxide Semiconductor (MOS)
- The GATE is the control.
- 1 = on = closed = conducting
- 0 = off = open = not conducting



Light Circuit with Transistor

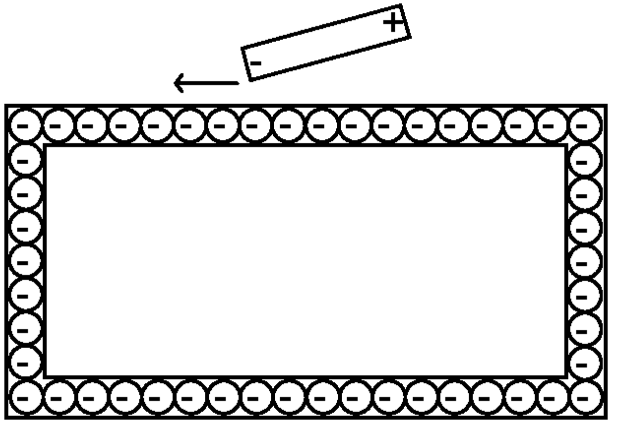
- 1 on Gate turns transistor "on".
- Transistor "on" means a complete path.
- A complete path means current flow.
- Current flow means light.





Electrons Move Easily

- The electrons here move TOO easily.



- If there is nothing to slow the electrons down the material quickly overheats and will probably melt.
- You MUST always have some sort of resistance to current flow to prevent this

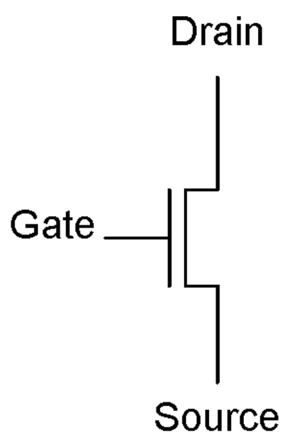
Important Points

- A transistor is a switch that can be considered open or closed.
- Voltage levels, not current, represent 1s and 0s.
- 1 is usually some positive voltage and 0 is a connection to ground or zero voltage.
- Voltage is present without current flow.
- You must always have some resistance in a circuit to prevent excessive heat.

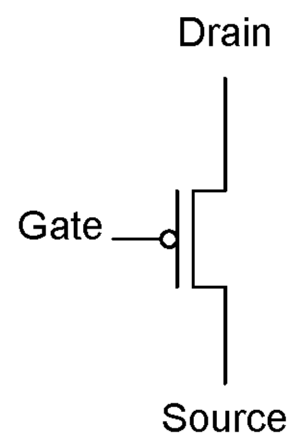
n-type or p-type

- n-type requires 1 to conduct
- p-type requires 0 to conduct

n-type

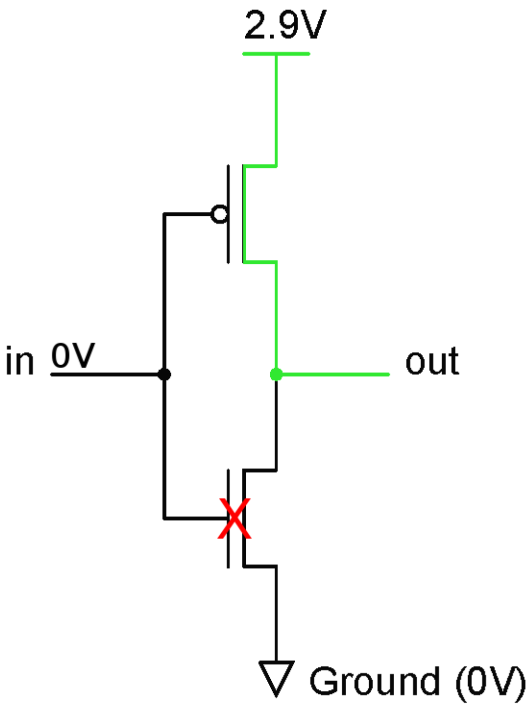


p-type



Building Circuits

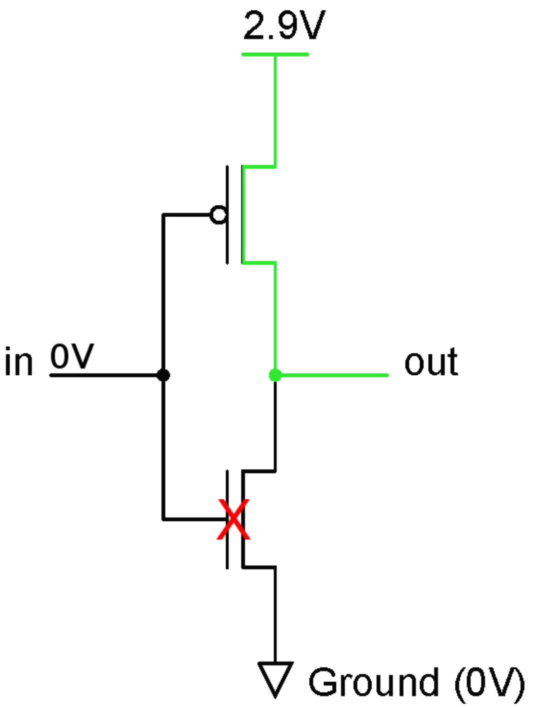
in	out
0	
1	



We are trying to either connect voltage (1) or ground (0) to the output.

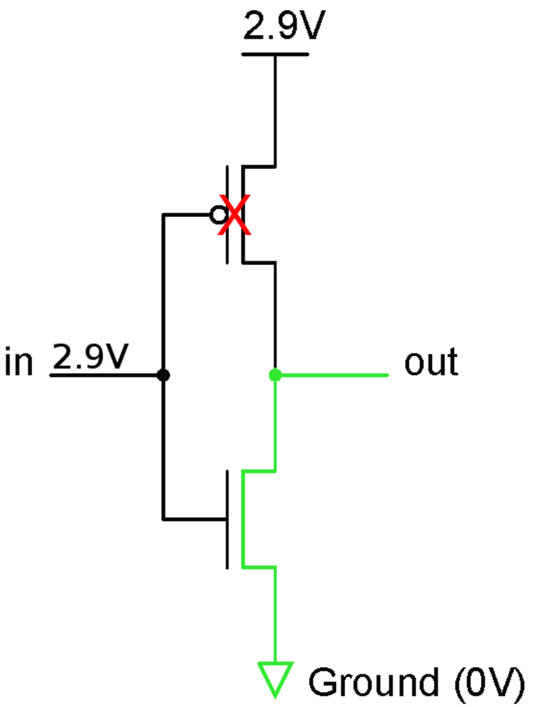
Building Circuits

in	out
0	1
1	



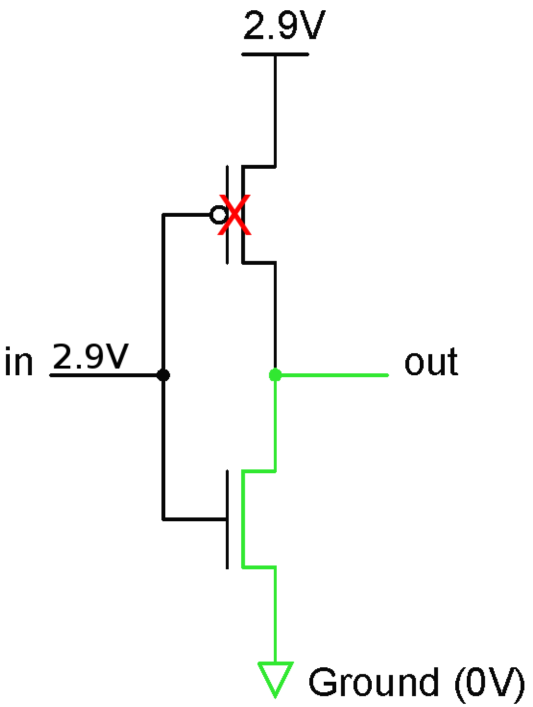
Building Circuits

in	out
0	1
1	



Building Circuits

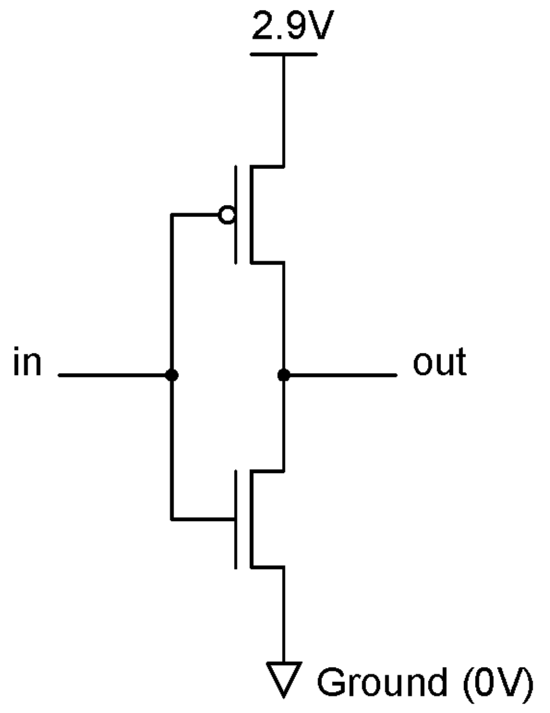
in	out
0	1
1	0



CMOS Inverter

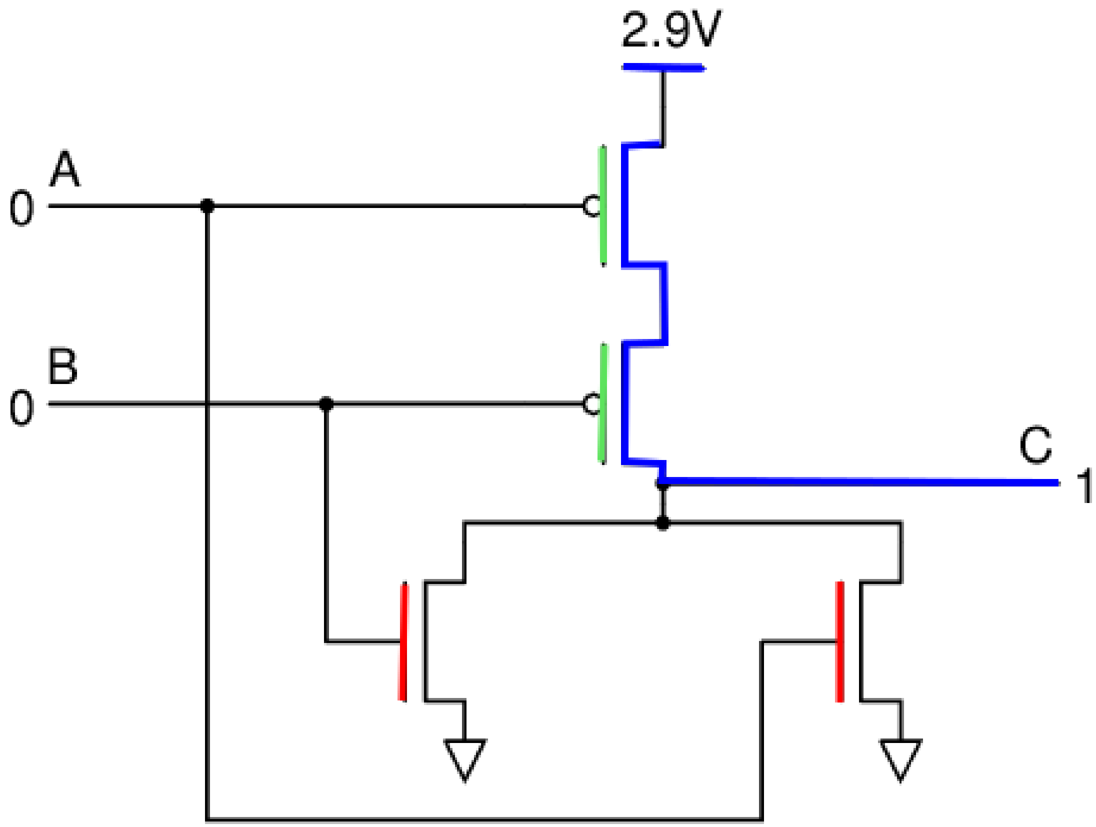
Complementary because it has both n-type and p-type

in	out
0	1
1	0



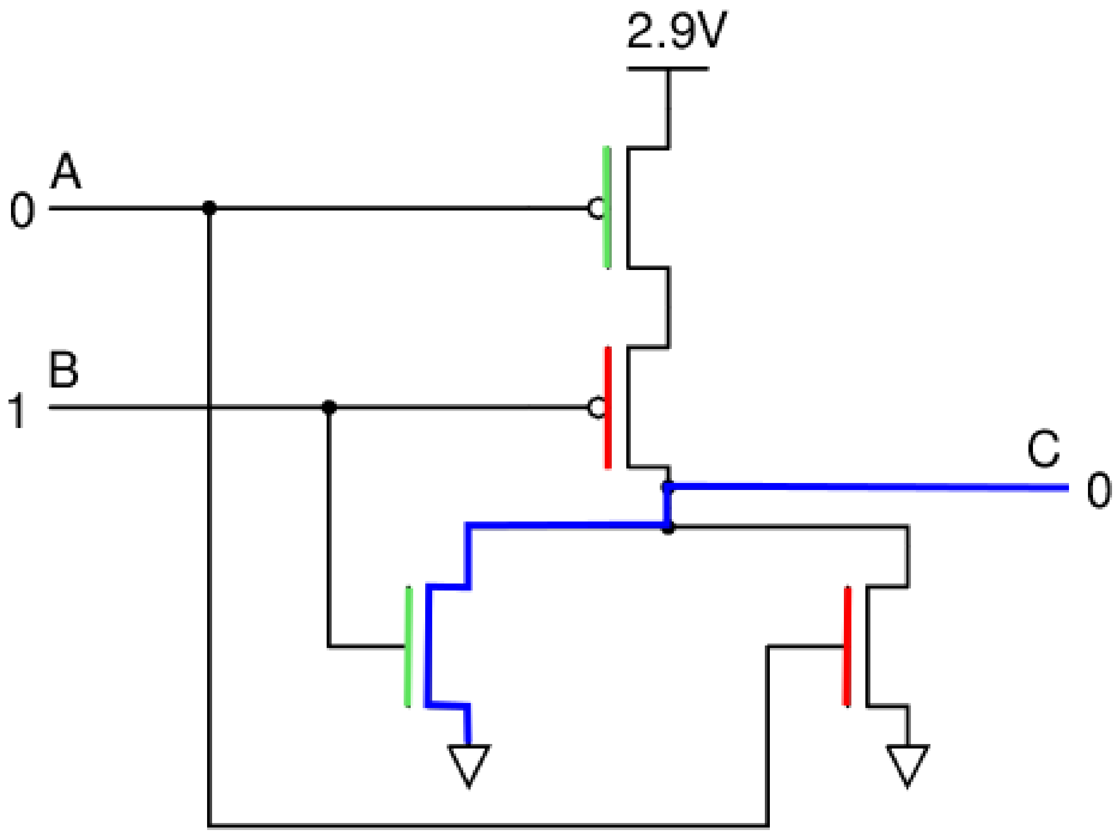
Fill in the truth table

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



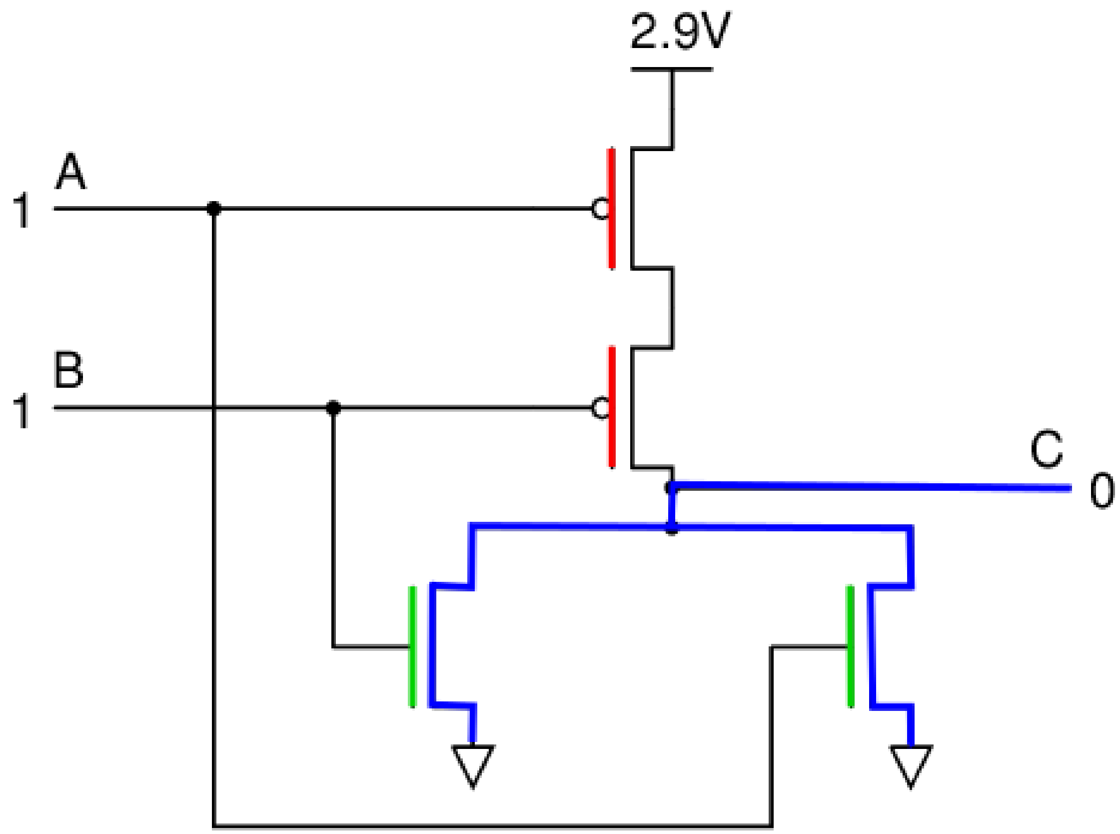
Fill in the truth table

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



Fill in the truth table

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

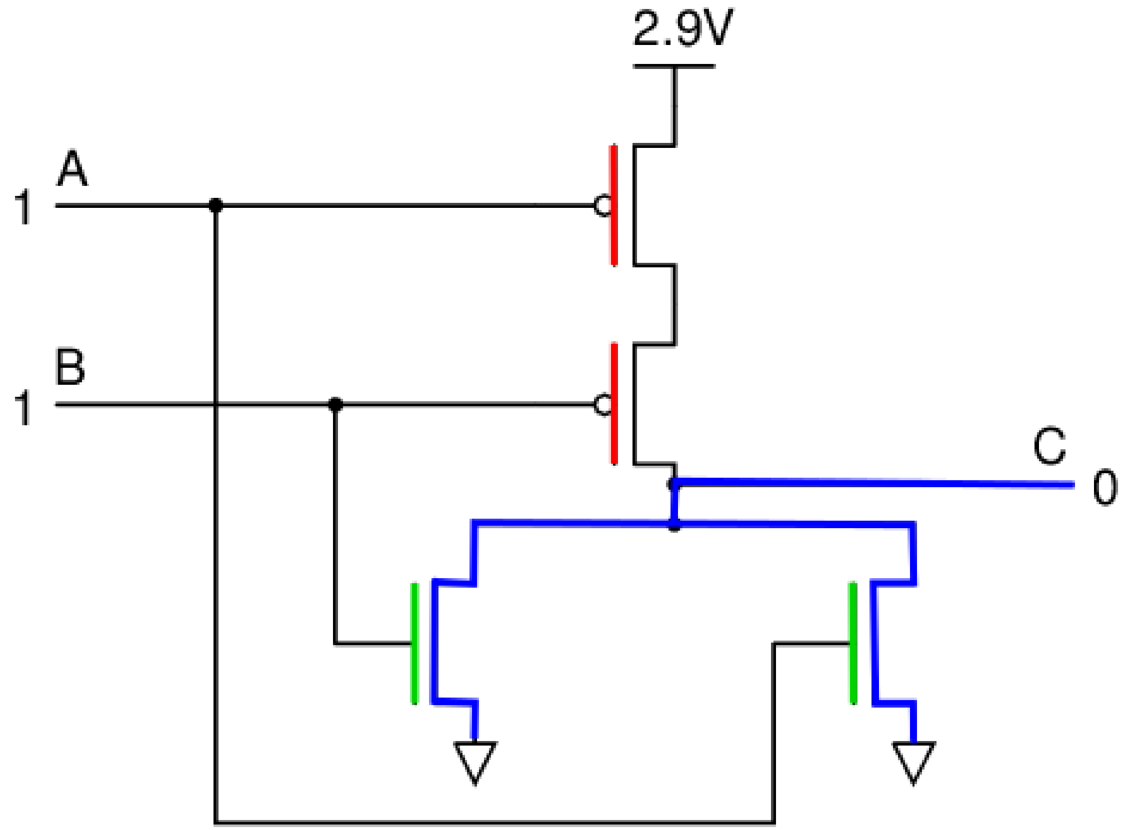


These circuits are designed so that either 1 or 0 is ALWAYS connected to the output and so that 1 and 0 are never connected at the same time.

Fill in the truth table

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

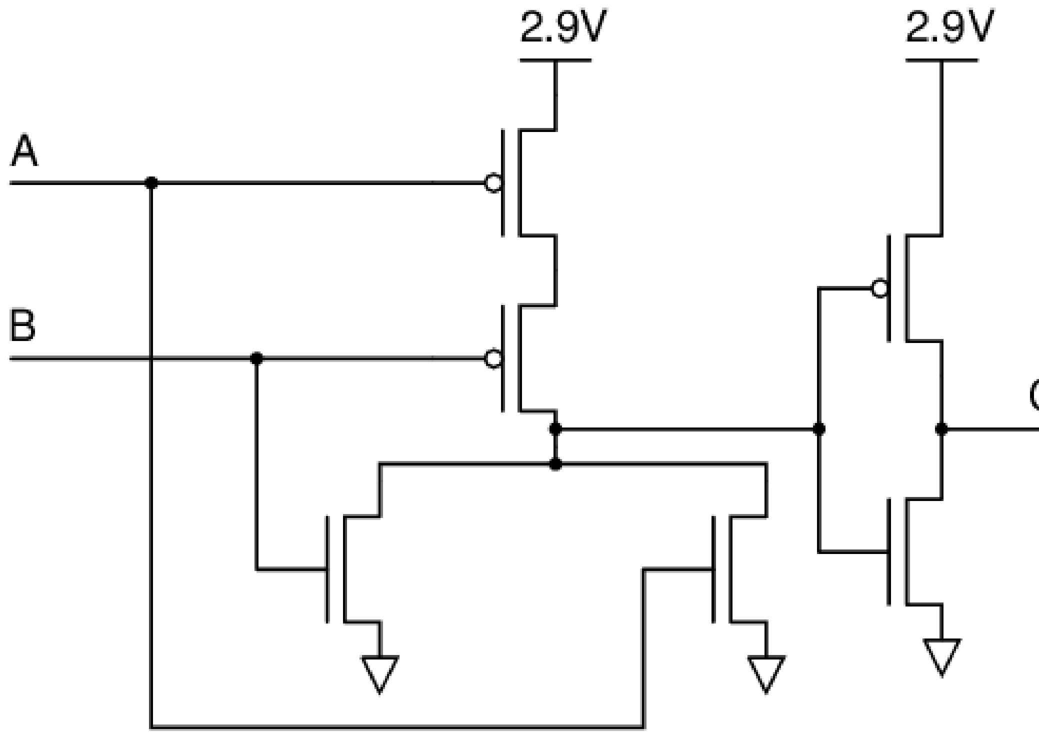
So what is it?



Fill in the truth table

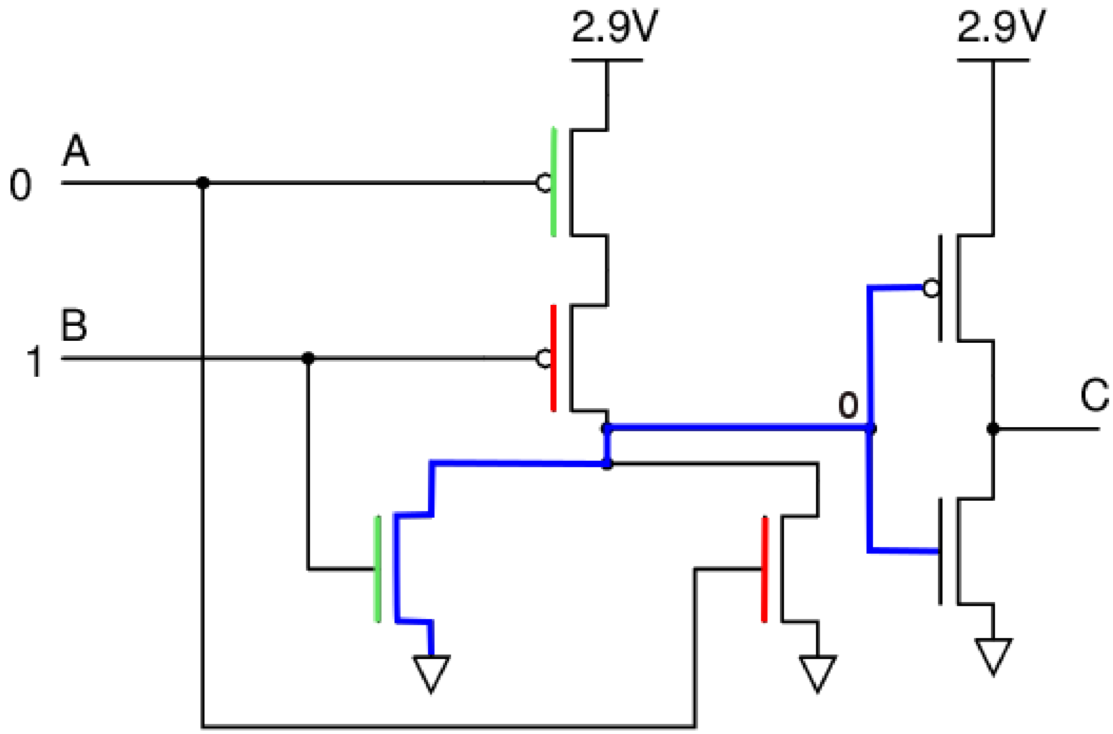
A	B	C
0	0	
0	1	
1	0	
1	1	

What is this?



Fill in the truth table

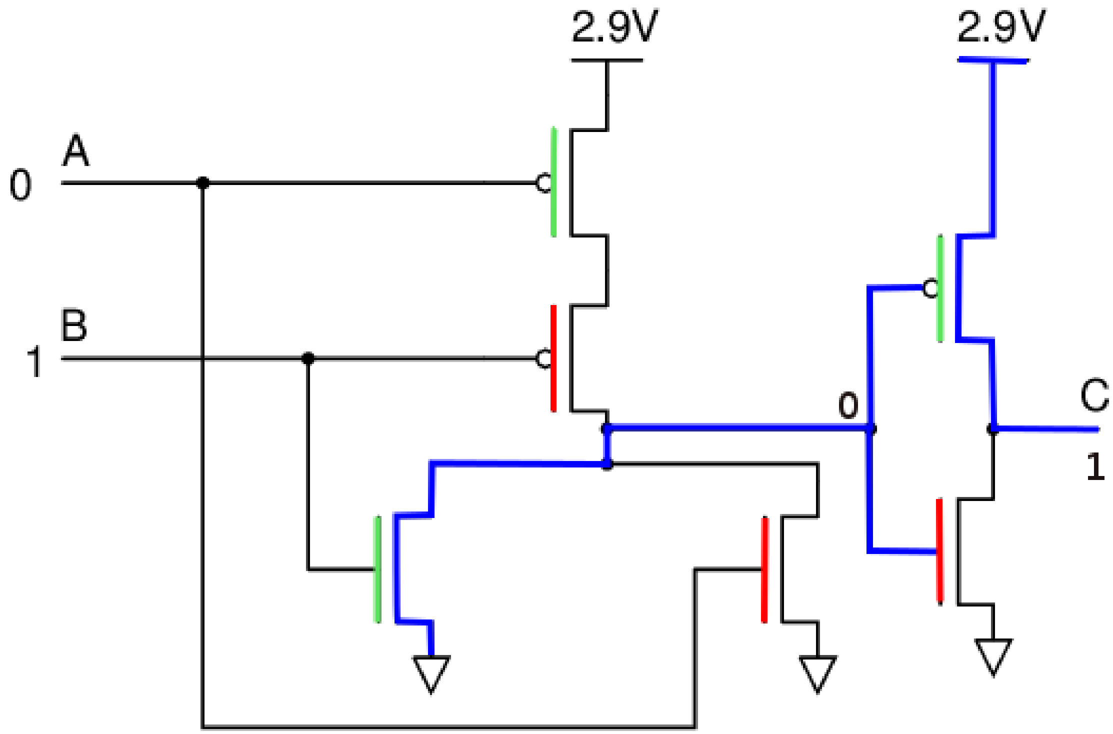
A	B	C
0	0	
0	1	?
1	0	
1	1	



Fill in the truth table

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

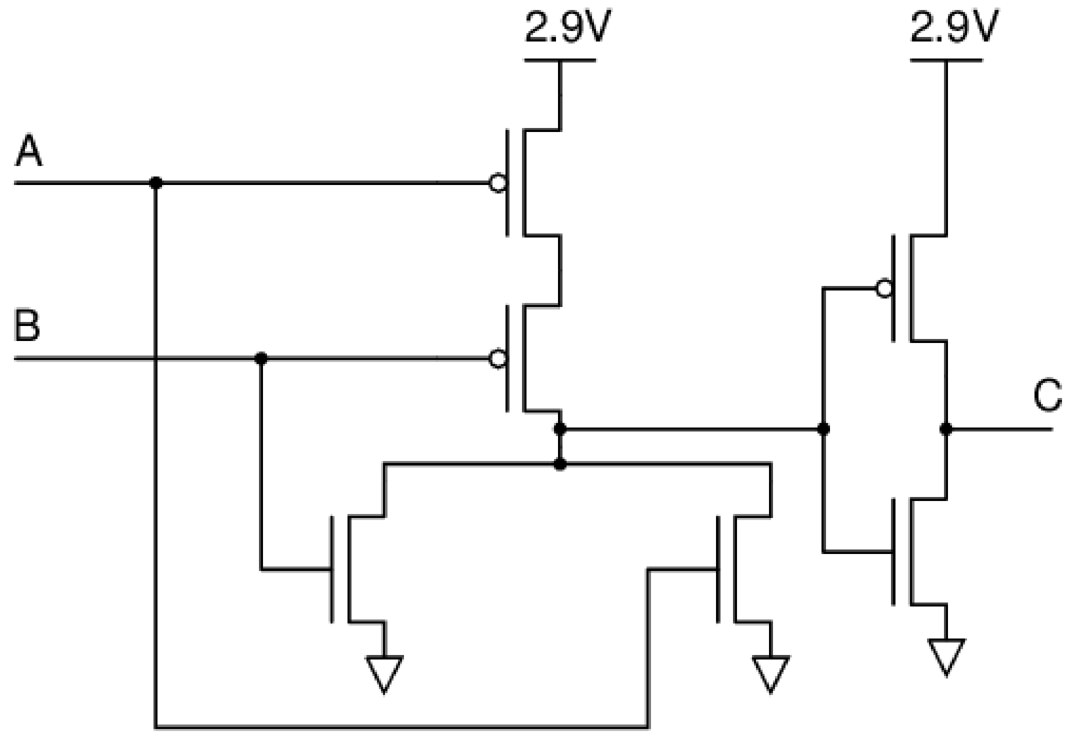
What is this gate?



Fill in the truth table

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

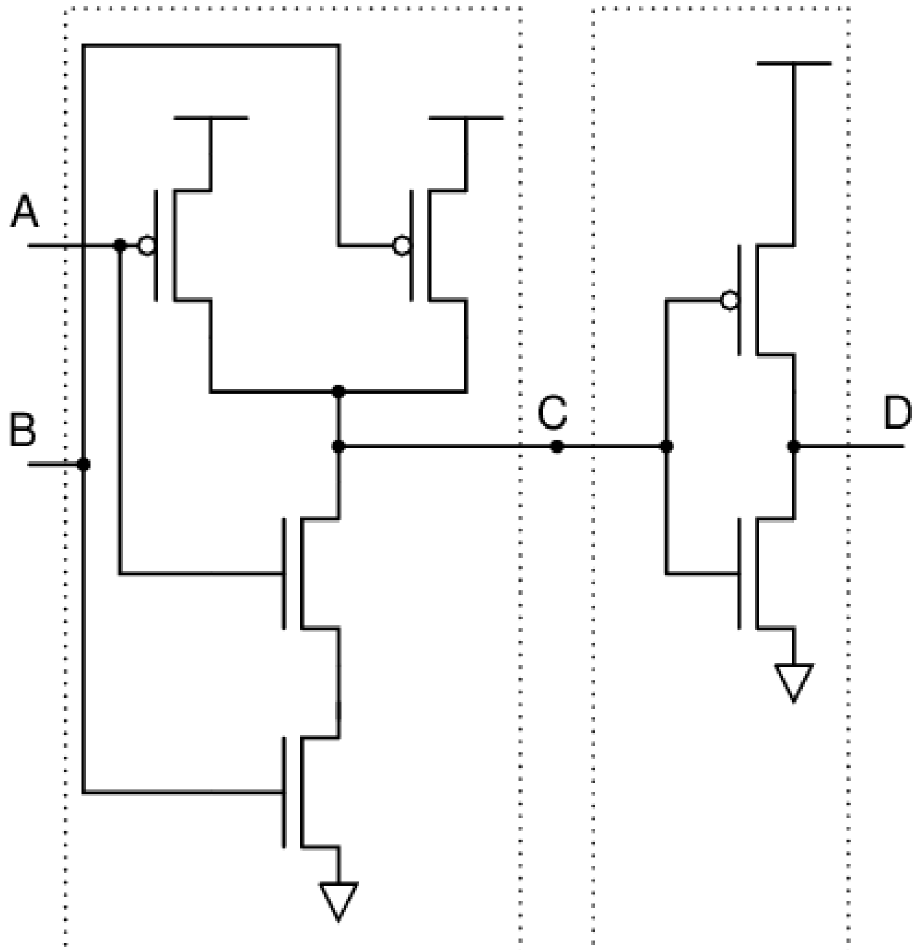
What is this gate?

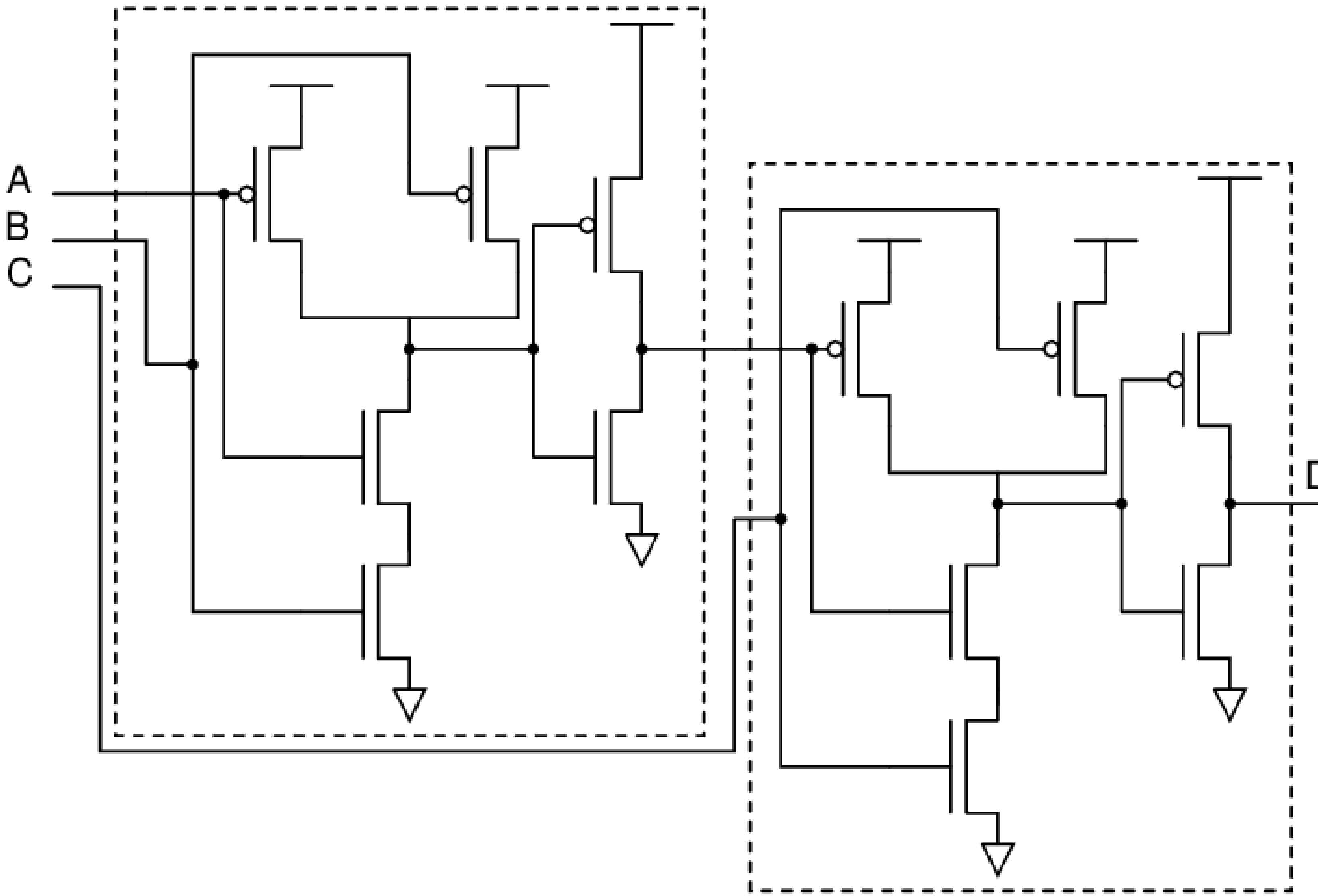


Fill in the truth table

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

What is this?




A
B
C
D

And Gate

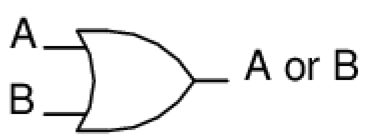
- And
  - The book uses the word ***and*** as the symbol
  - Logic uses ^
  - Other digital sources use * (as in multiplication)
  - Examples:

    A and B   A^B   A*B   AB
  - Electronic Symbol

Truth Table		
A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1

Or Gate

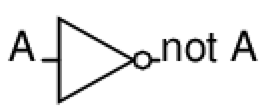
- Or
  - The book uses the word ***or*** as the symbol
  - Logic uses v
  - Other digital sources use + (as in addition)
  - Examples
    - A or B AvB A+B
  - Electronic Symbol



Truth Table		
A	B	A or B
0	0	0
0	1	1
1	0	1
1	1	1

Inverter Gate

- Inverter
  - The book uses the word ***not***
  - Logic uses $\neg$
  - Other digital sources use an over line or ' (apostrophe)
  - Examples
    - not A A' $\bar{A}$
  - Electronic Symbol

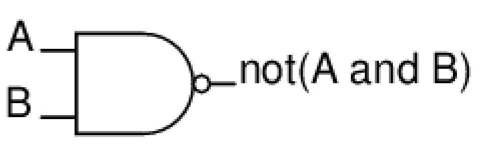


Truth Table	
A	not A
0	1
1	0

Nand Gate

- Nand
  - Combination of "***not gate"*** and "***and gate"***
  - Examples (note that most require parenthesis):

    not(A and B) $\neg$ (A^B)   (A*B)' $\overline{AB}$
  - Note that these **ARE NOT** equivalent to the following
    - not A and B   not A and not B $\bar{A}\bar{B}$
  - Electronic Symbol

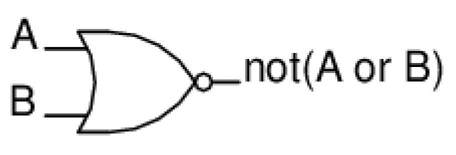


Truth Table			
A	B	A and B	Not(A and B)
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Nor Gate

- Nor
  - Combination of "***not gate"*** and "***or gate"***
  - Examples (note that most require parenthesis):

    not(A or B) $\neg$ (AvB) (A+B)' $\overline{A + B}$
  - Note that these **ARE NOT** equivalent to the following
    - not A or B not A or not B $\bar{A} + \bar{B}$
  - Electronic Symbol



Truth Table			
A	B	A or B	Not(A or B)
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

XOR - XNOR

A	B	A ^ B
0	0	0
0	1	1
1	0	1
1	1	0

A	B	(A ^ B)'
0	0	1
0	1	0
1	0	0
1	1	1

- Exclusive OR (XOR) means A or B but not both.

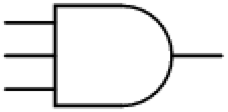
$$\mathrm{A} \oplus B = \bar{A}B + A\bar{B}$$

N-Input Gates

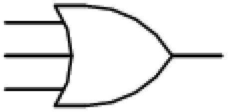
- The gates shown with two inputs can have multiple input versions.
- The internals are more complicated but the result is the same.
- ***And*** gates will have an output of 0 if any of the inputs are 0.
- ***Or*** gates will have an output of 1 if any of the inputs are 1.

Three input examples:

And



Or



DeMorgan's Laws

- You can use DeMorgan's Law to say the same thing in a different way.

- DeMorgan

  not (p or q) is the same as not p and not q

  not (p and q) is the same as not p or not q

- A phrase with a negation can be rewritten using DeMorgan

  I do not speak either Spanish or German

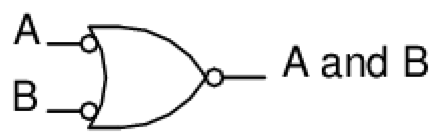
  I do not speak Spanish and I do not speak German

  I do not drink and drive

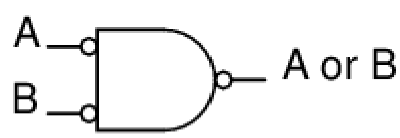
  I do not drink or I do not drive

DeMorgan Version of Gates

- And: $A\ and\ B = \overline{\overline{A\ and\ B}} = \overline{\bar{A}\ or\ \bar{B}}$



- Or: $A\ or\ B = \overline{\overline{A\ or\ B}} = \overline{\bar{A}\ and\ \bar{B}}$

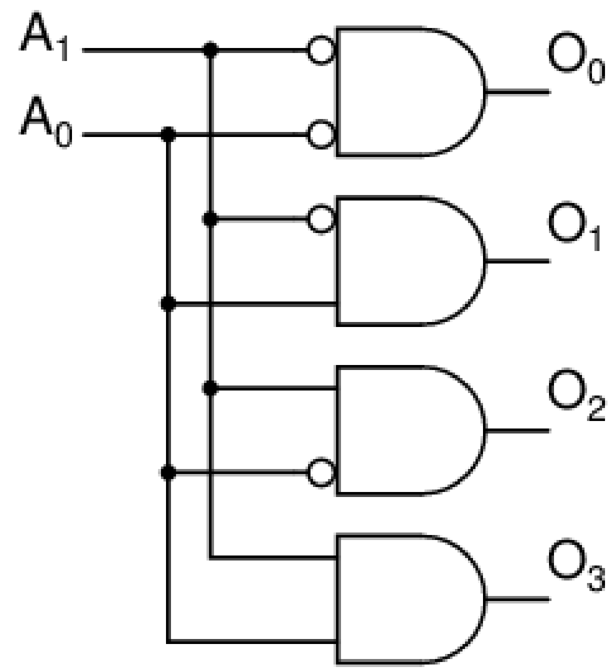


Combination Circuits

- Bigger circuits using gates to accomplish a specific task.
- Often it isn't important to see the internal workings as long as you know what they are and how they work.
  - Decoder
  - Multiplexor
  - Adder
  - Counter

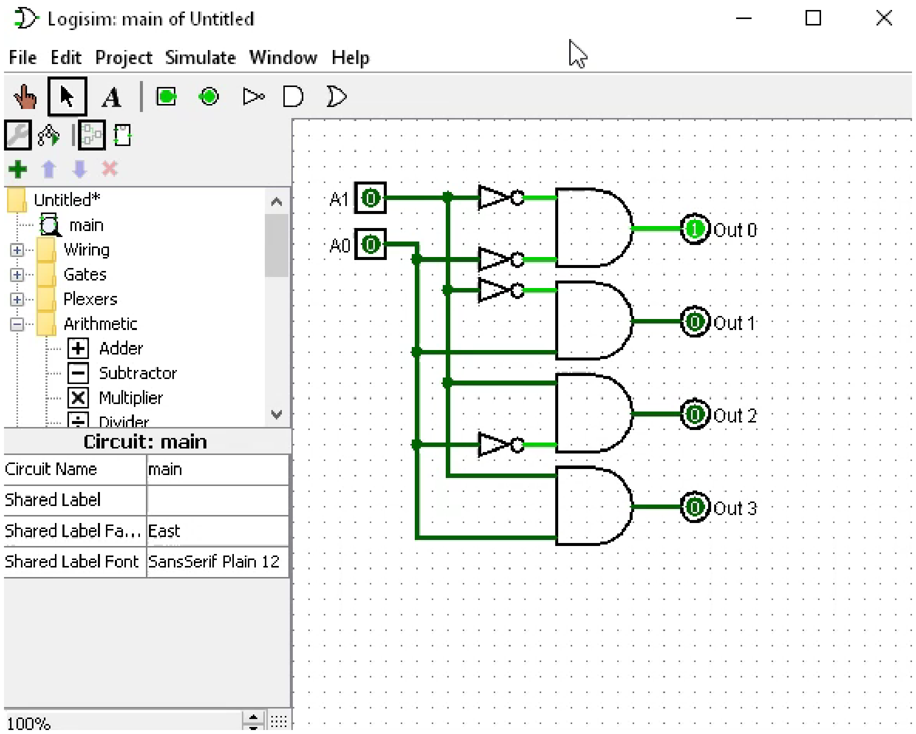
2 to 4 Decoder

- Select a specific output pattern for each input pattern
- Typically a single output is 1 for a given input but others are possible (7 segment decoder).
- Think of it as decoding a binary number.
- This is a 2x4 decoder. Other sizes are common.



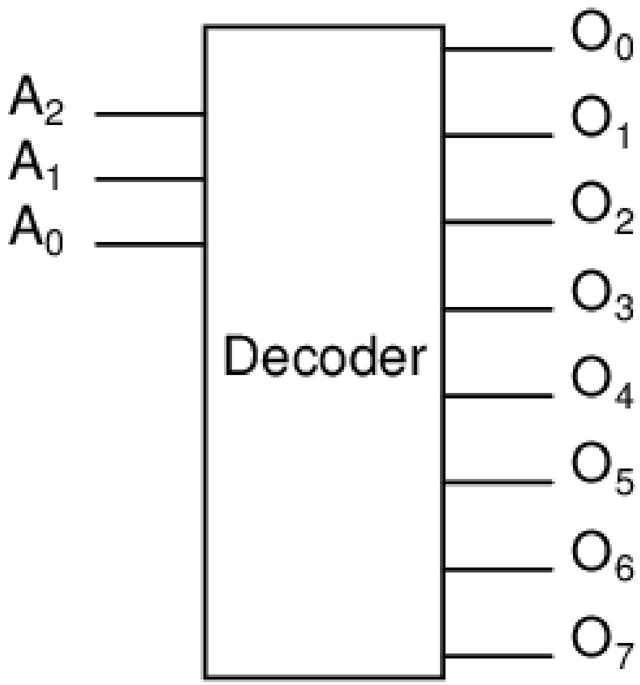
Logisim

- Digital Logic Simulator
- Written in Java
- Works on any OS
- 4-bit Decoder



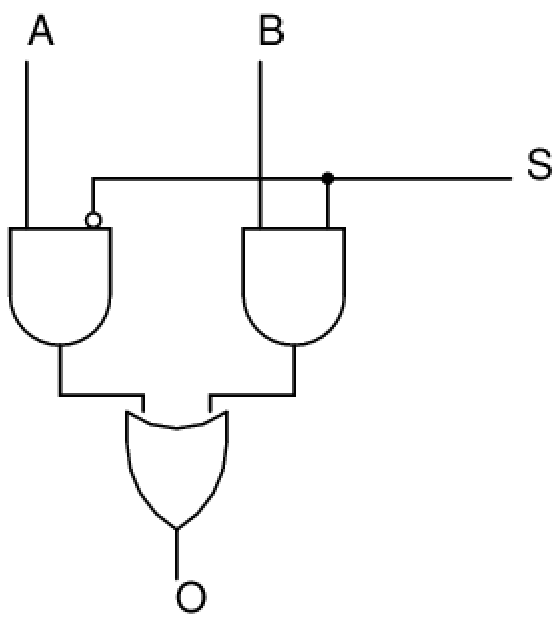
3 to 8 Decoder Block

- Could you draw the internals?
- Why 8 output lines?
- What would require 16 output lines?
- Assume 5 on input, what would the output be? What about 7?
- Build a 3 to 8 decoder in Logisim tonight.



Multiplexer

- A multiplexer (or mux) is a digital selector.
- A multiplexer chooses one of several inputs to appear on an output.
- More complex multiplexers consist of multiple single multiplexers working together.



Multiplexer

- S is the select bit (or bits).
- A and B are inputs.
- The value of A or B will appear on the output according to the value on S.

  A=0, B=0, S=0, O=____

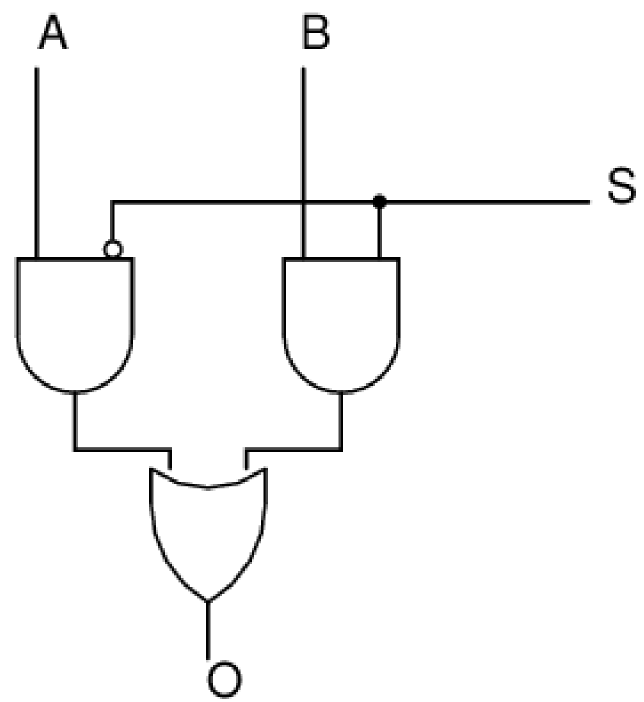
  A=0, B=1, S=0, O=____

  A=0, B=1, S=1, O=____

  A=1, B=0, S=0, O=____

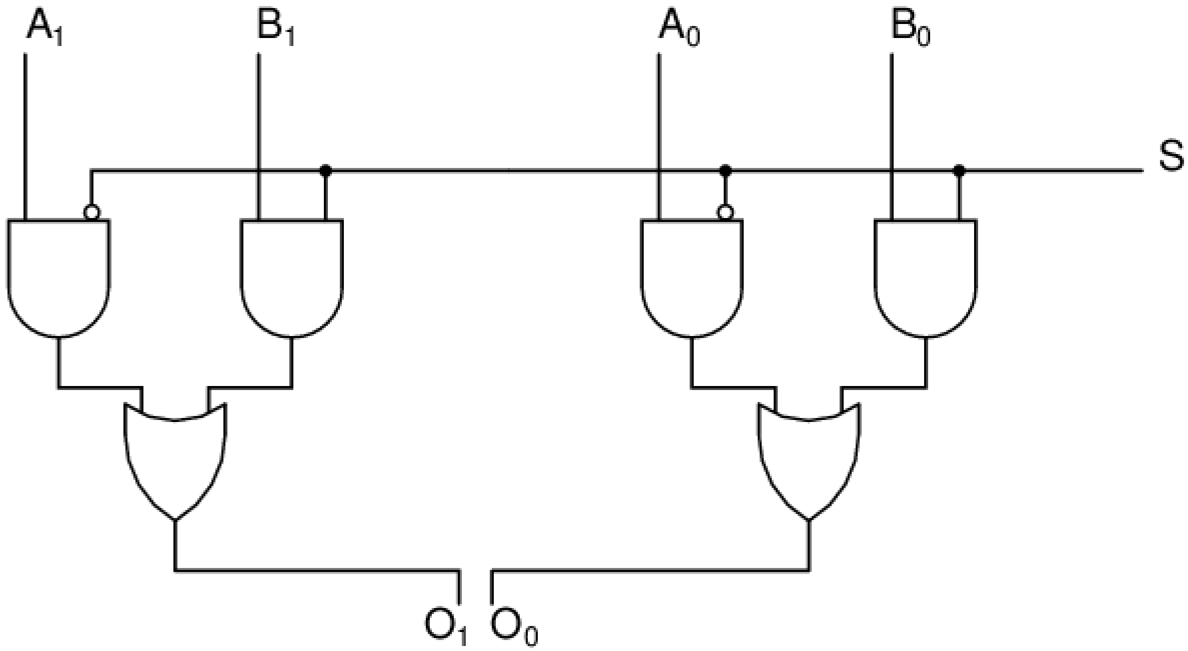
  A=1, B=0, S=1, O=____

  A=1, B=1, S=1, O=____

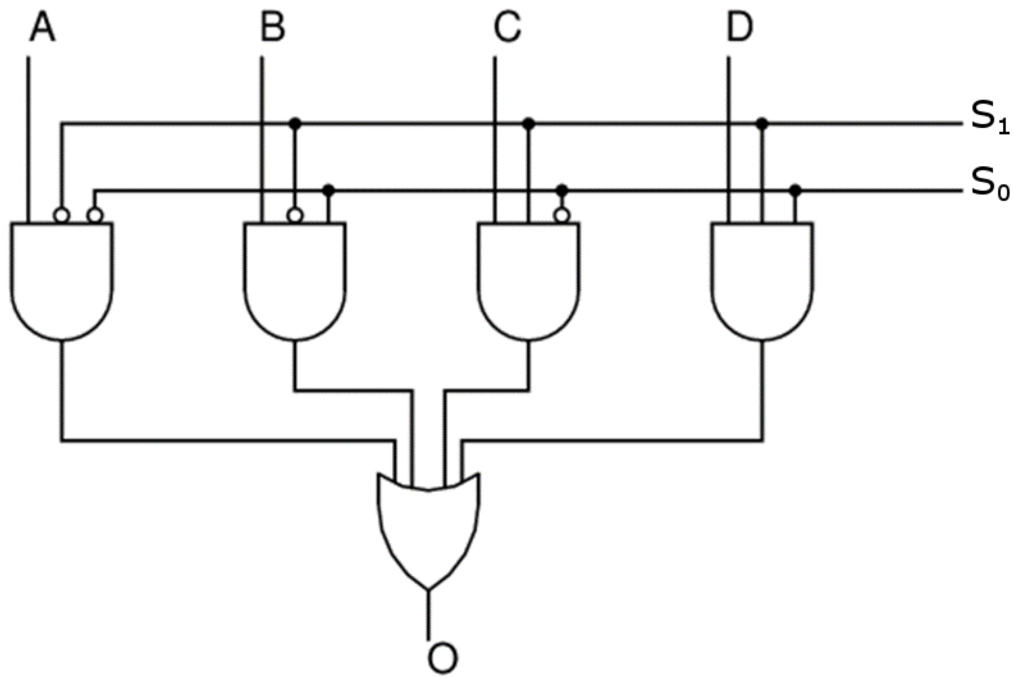


More Complex Mux 4x2

- A = 3, B = 2, S = 0: What is O?
- A = 1, B = 3, S = 1: What is O?

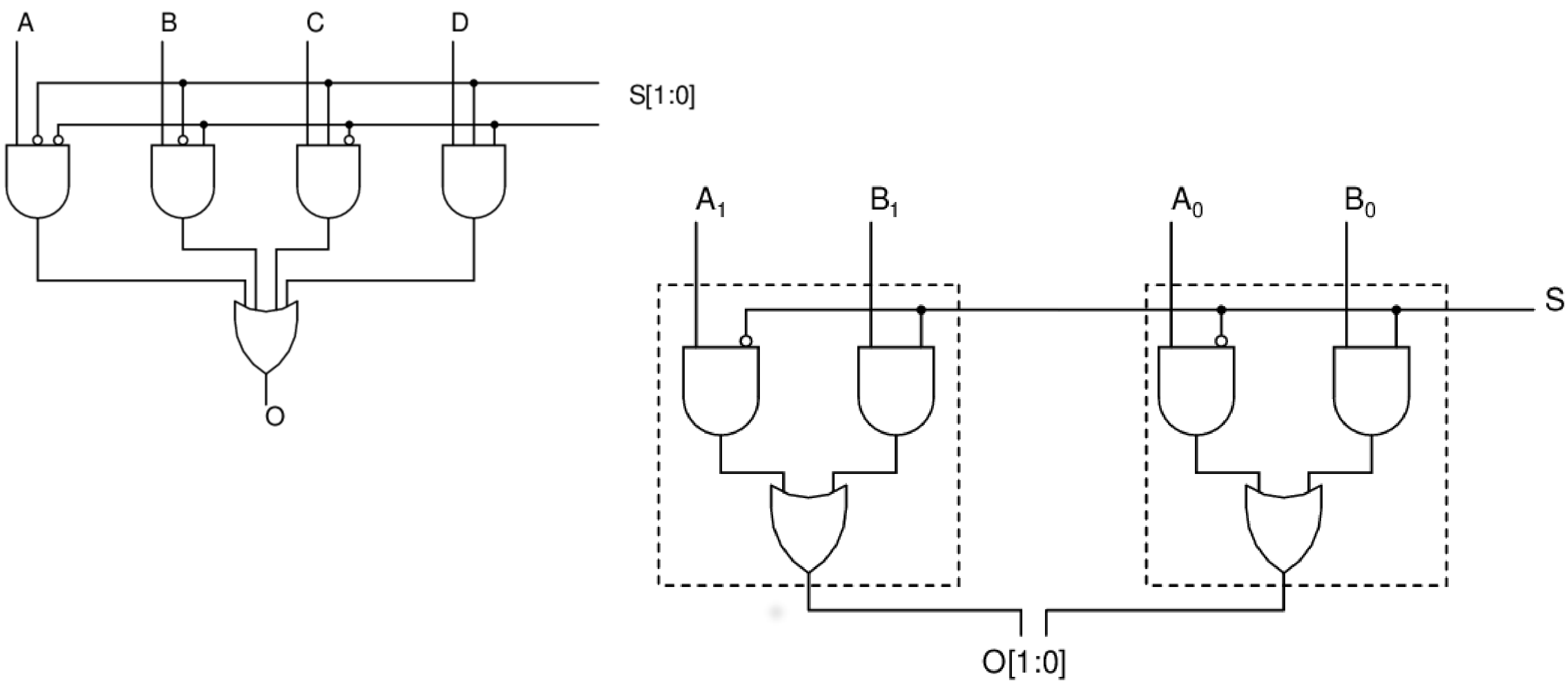


Another Complex Mux 4x1

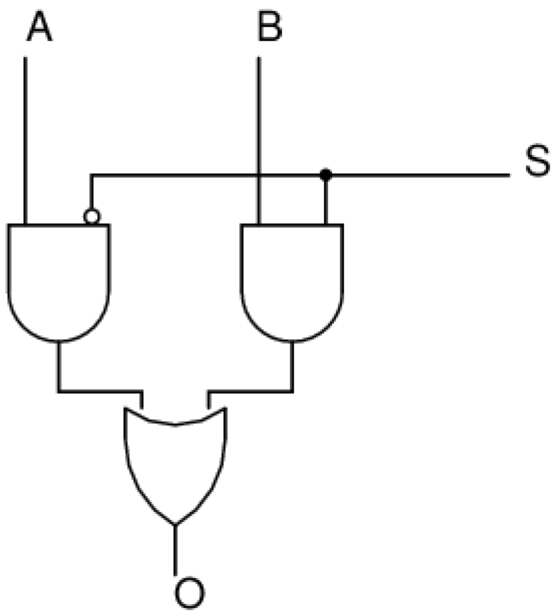


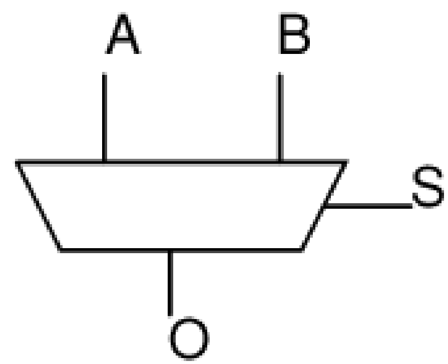
4x1 vs 4x2

- Notice the difference

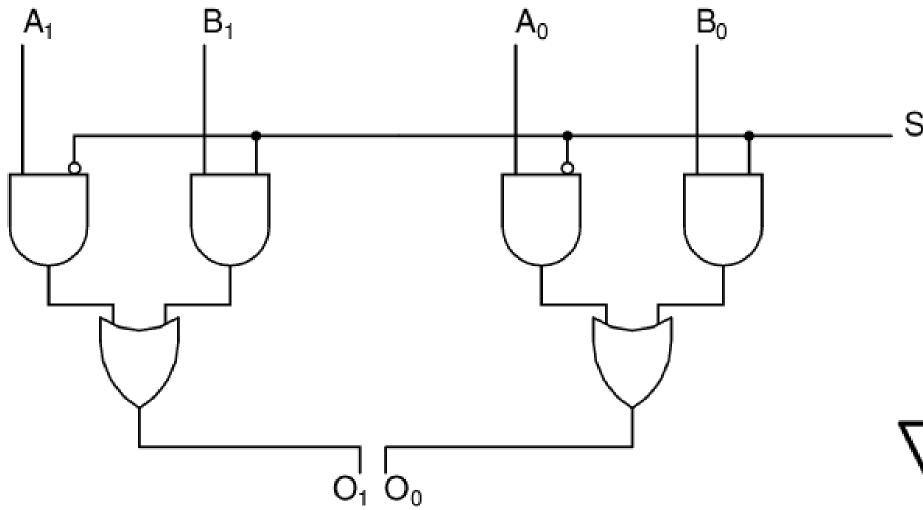


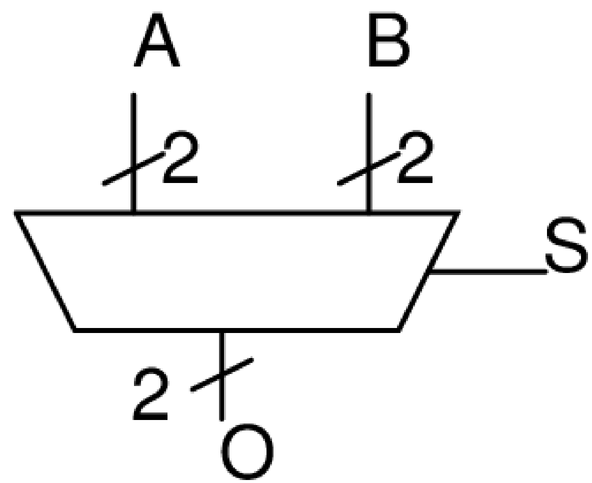
2x1 Block Diagram of a Mux



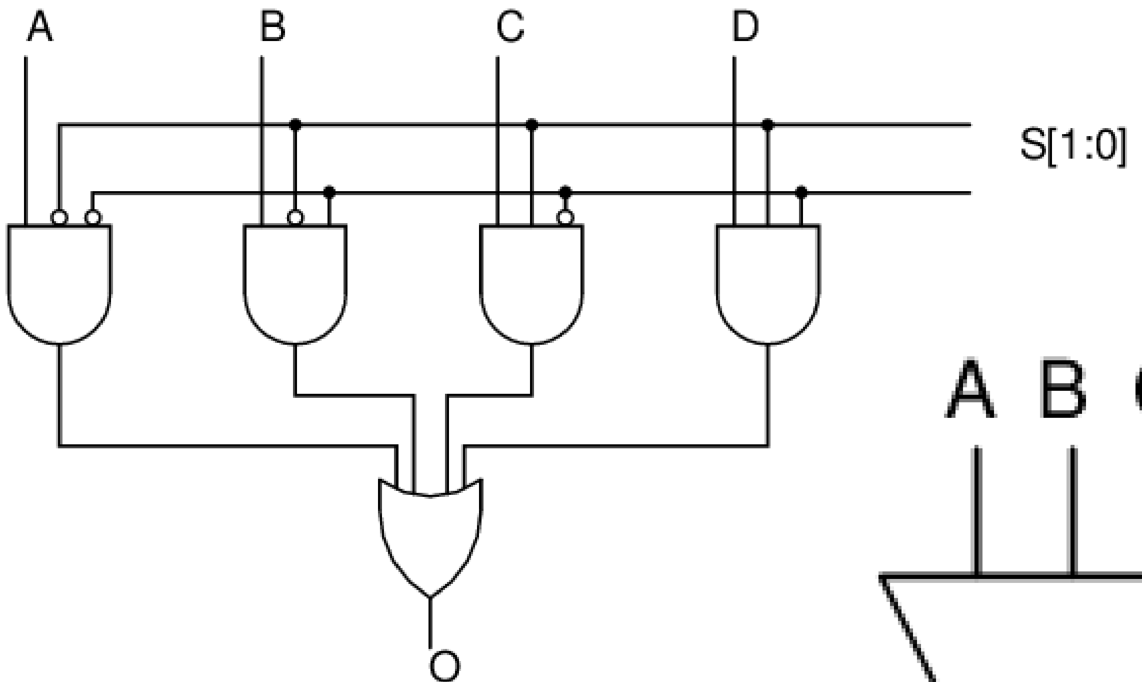


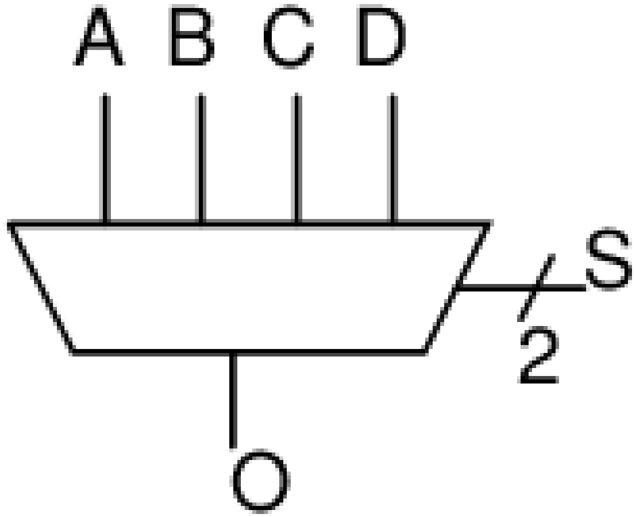
4x2 Block Diagram of a Mux





4x1 Block Diagram



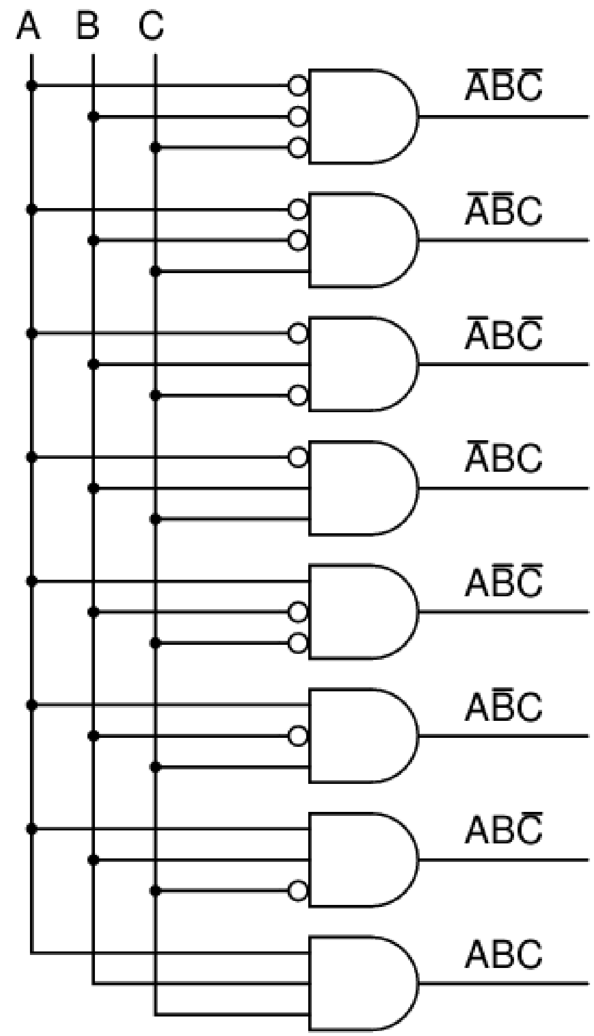


Homework:

Build a 16x4 Multiplexer in Logisim

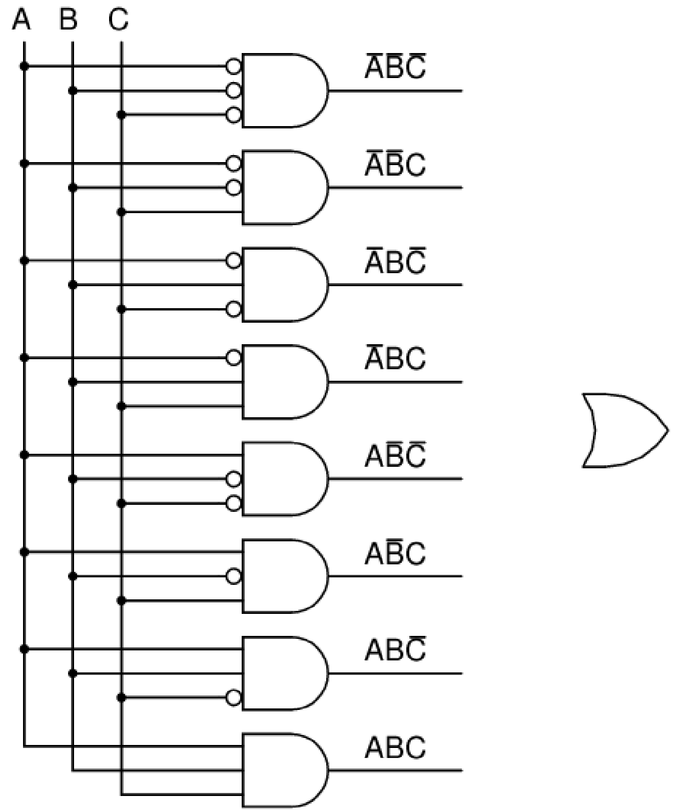
Decoder as a Universal Circuit

- What is the output when A=0, B=0, and C=0?



Decoder as a Universal Circuit

- $f(A,B,C) = \bar{A}BC + \bar{A}\bar{B}C + ABC$



Adding in Binary

Add A + B

	1	1	1	0	<-Carry
	0	1	1	1	A=7
+	0	0	1	1	B=3
	1	0	1	0	(10)

What is required to add one column?
One bit from A, One bit from B, and a carry in.

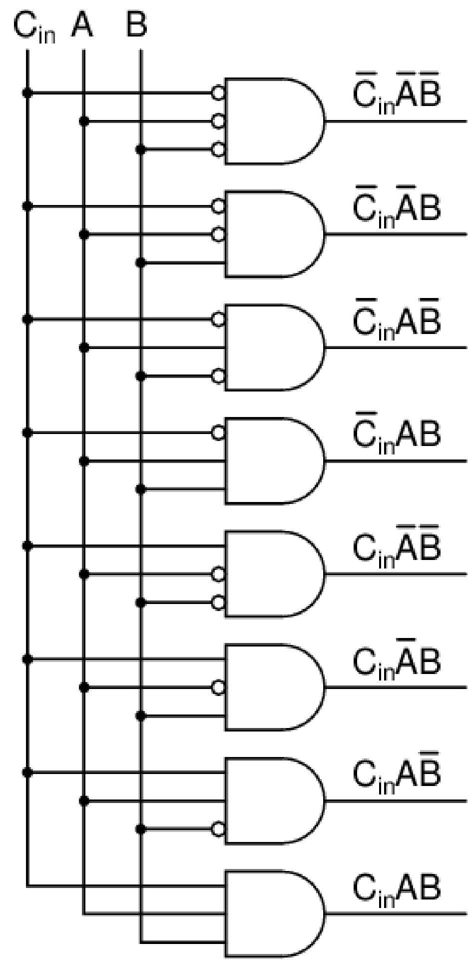
What is the output?
Sum and Carry out.

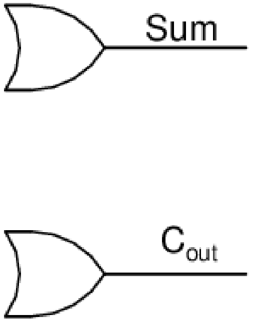
Truth table for adding one bit.

Cin	Bit From A	Bit from B	Sum	Cout
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

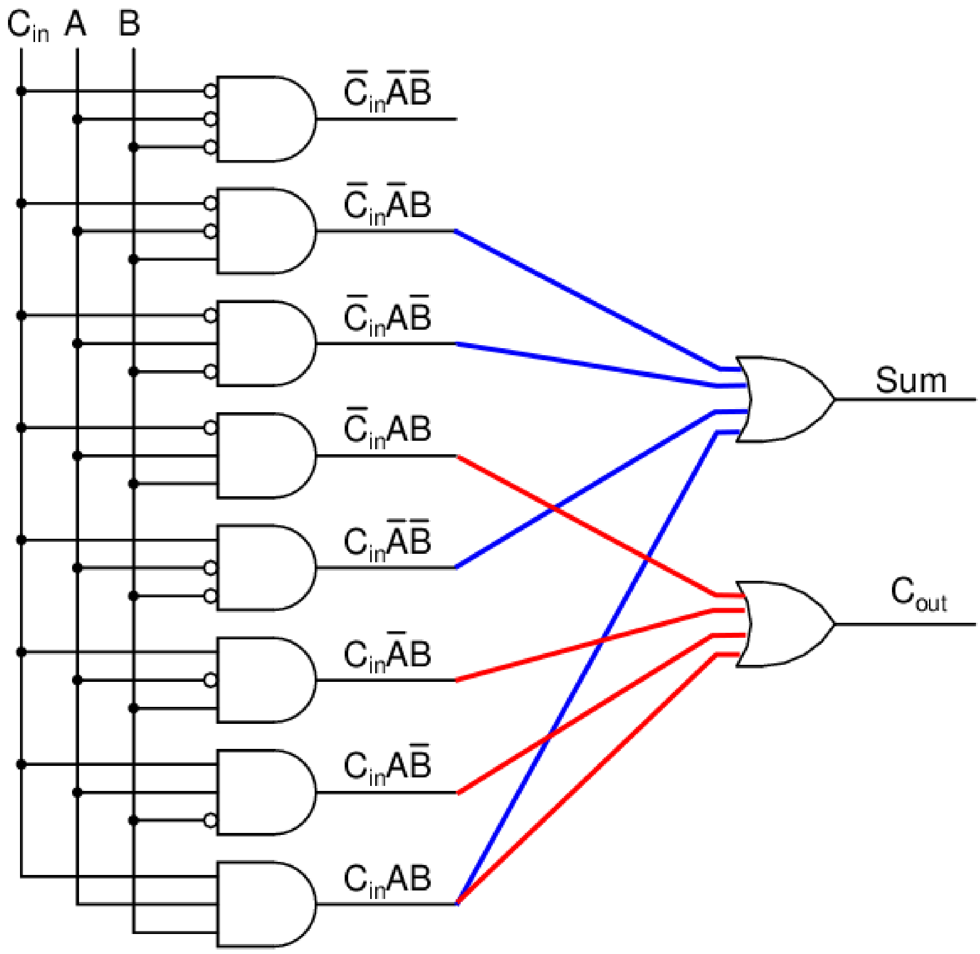
Create the circuit

C_{in}	A	B	Sum	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



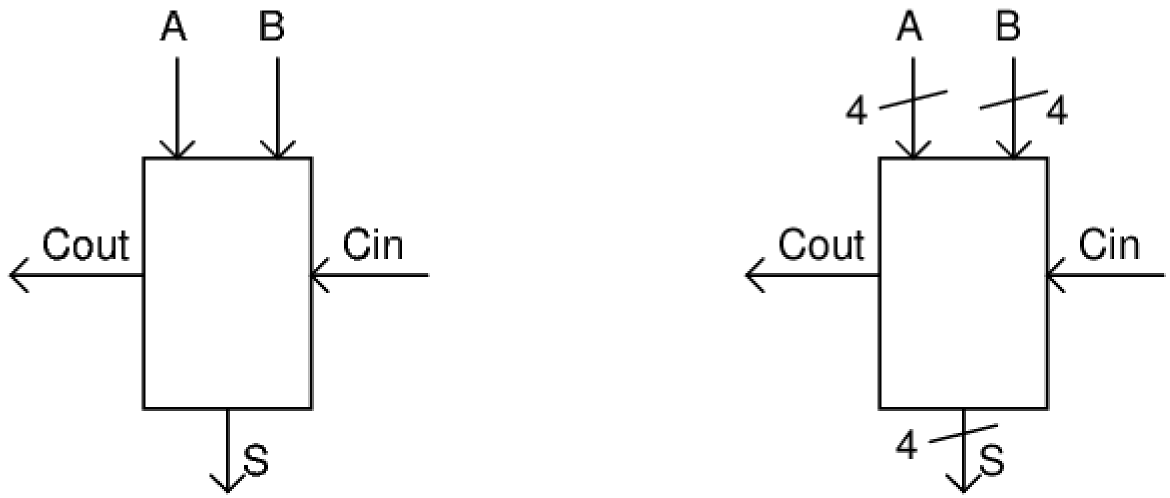


Adder

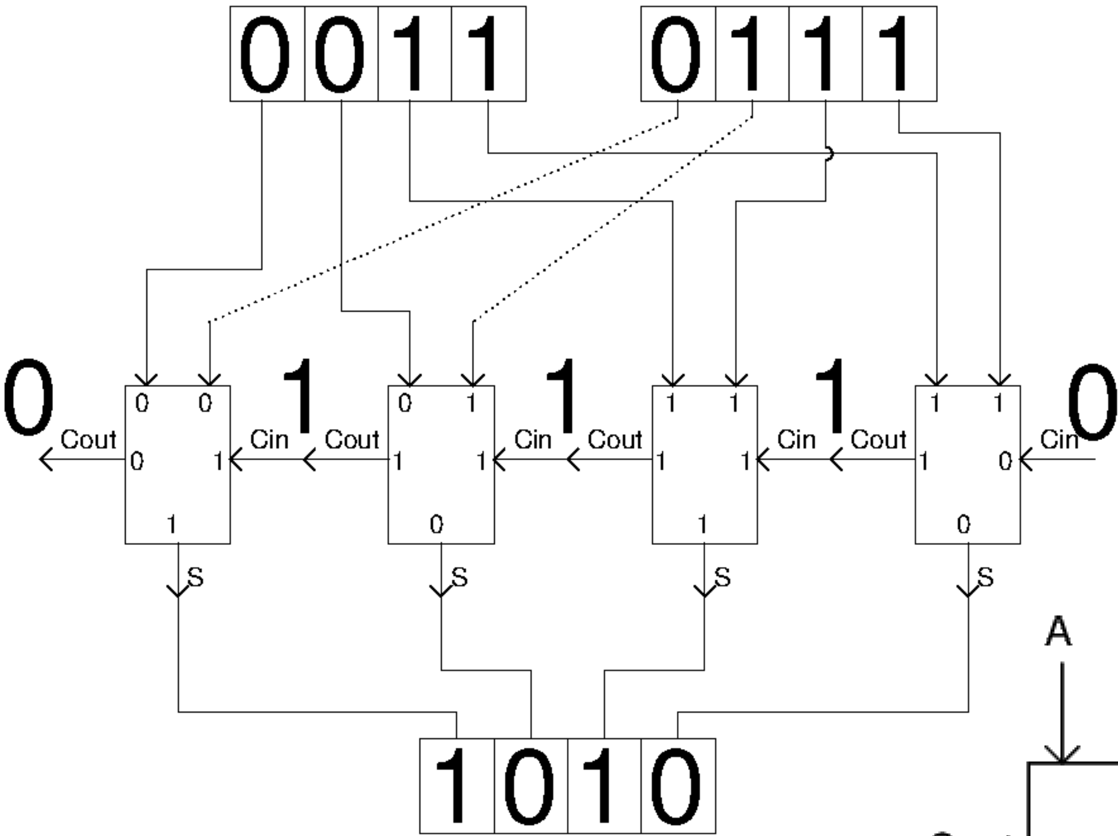

C_{in} A B
$\overline{C}_{in}\overline{A}\overline{B}$
$\overline{C}_{in}\overline{A}B$
$\overline{C}_{in}A\overline{B}$
$\overline{C}_{in}AB$
$C_{in}\overline{A}\overline{B}$
$C_{in}\overline{A}B$
$C_{in}A\overline{B}$
$C_{in}AB$
Sum
C_{out}

Adder Block Diagram

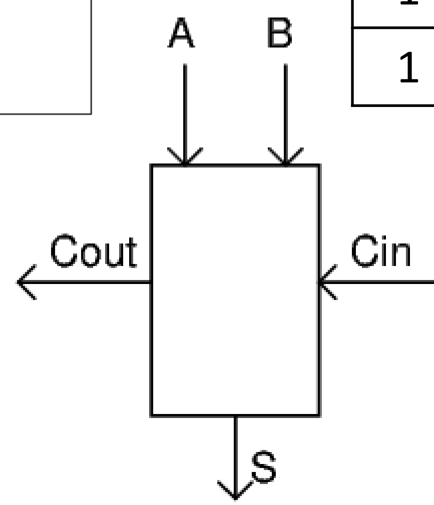
By creating a circuit for a single bit, it makes it easier to add more bits by simply duplicating the adder circuit for each bit.



4 bit adder



Cin	B1	B2	Sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



Homework:

Build a 4-bit Adder in Logisim

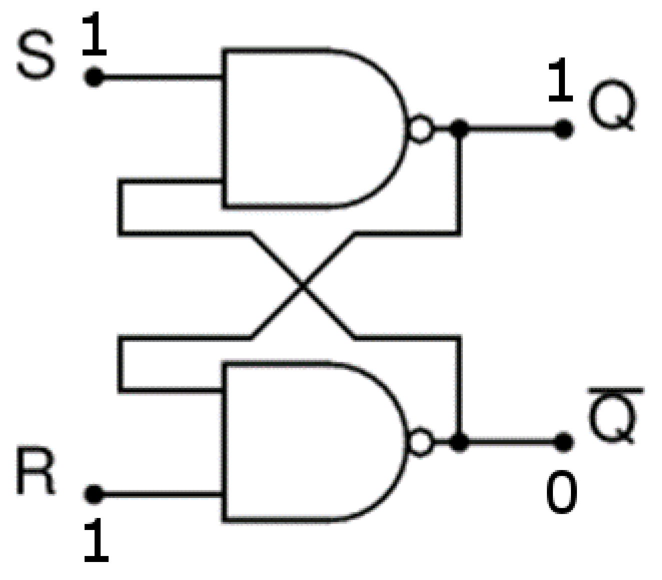
Storing 1s and 0s

- So far 1s and 0s have just appeared on inputs without explanation about where they come from.
- For a computer to add it needs an adder circuit, but the inputs must be stored somewhere on the computer also.
- The result, also, must get stored somewhere for it to be useful.
- We store these bits in a circuit called a latch.

R-S Latch

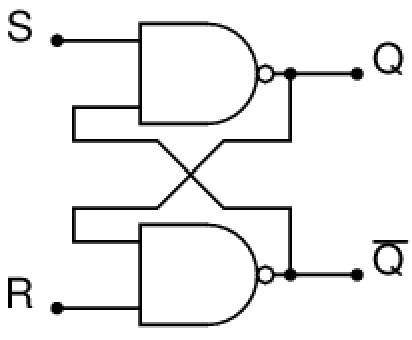
NAND TRUTH TABLE

A	B	$\overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0



R-S Latch

R	S	Q
1	1	No Change
1	0	1
0	1	0
0	0	Unused

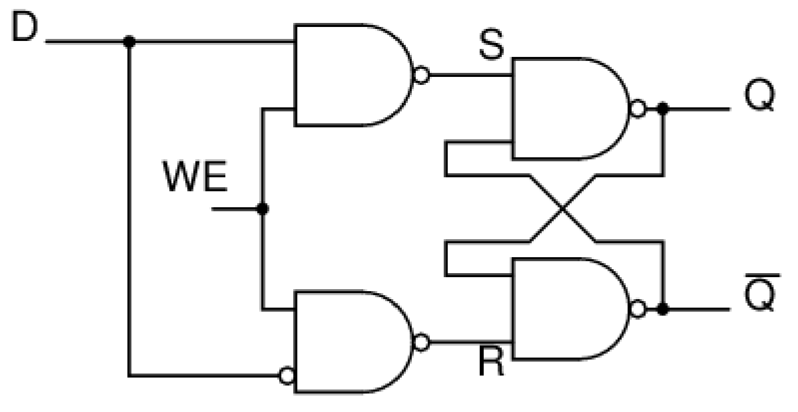


1 on S and R means Q doesn't change.
0 on S means **set** Q to 1
0 on R means **reset** Q to 0
Do not put 0 on S and R at same time.

The Gated D Latch

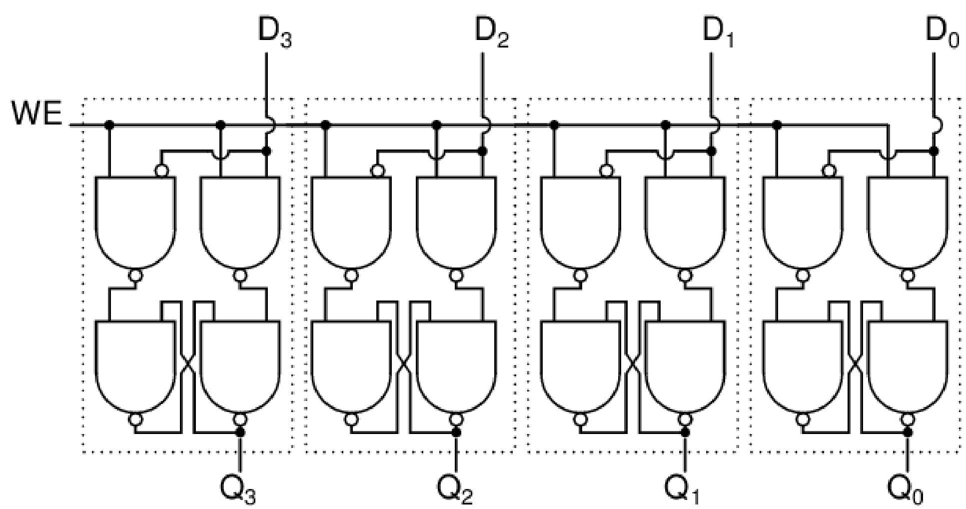
- A modified R-S latch.
  - A single input (D)
  - A write enable circuit (WE)
- A gate circuit directs inputs to R and S.

D	WE	Q
0	0	No Change
0	1	0
1	0	No Change
1	1	1



Storing More than One Bit

- How could four bits be stored? Use four D latches working together.
- The device for storing multiple bits as a unit is called a **REGISTER**.
- Bigger numbers simply use more latches.
- 64 bit microprocessors use 64 latches for registers.



Register Annotation

- The bits are labeled with subscripts from low order to high order which is typically right to left.
- The instruction register (IR) is a 16 bit register

IR_{15}	IR_{14}	IR_{13}	IR_{12}	IR_{11}	IR_{10}	IR_9	IR_8	IR_7	IR_6	IR_5	IR_4	IR_3	IR_2	IR_1	IR_0
1	0	0	1	0	1	0	0	0	1	1	1	0	1	1	0

- We can refer to ranges of bits using brackets

  IR[15:12] = 1001

  IR[7:0] = 01110110

More Register Annotation

R1 <- R2 + R3

R1 <- R2 + 0

R1 <- R1 and 0

R1 <- not R1

R1 <- R1 + 1

R1 <- R1 + R2

What do these do?

MDR <- MEM[MAR]

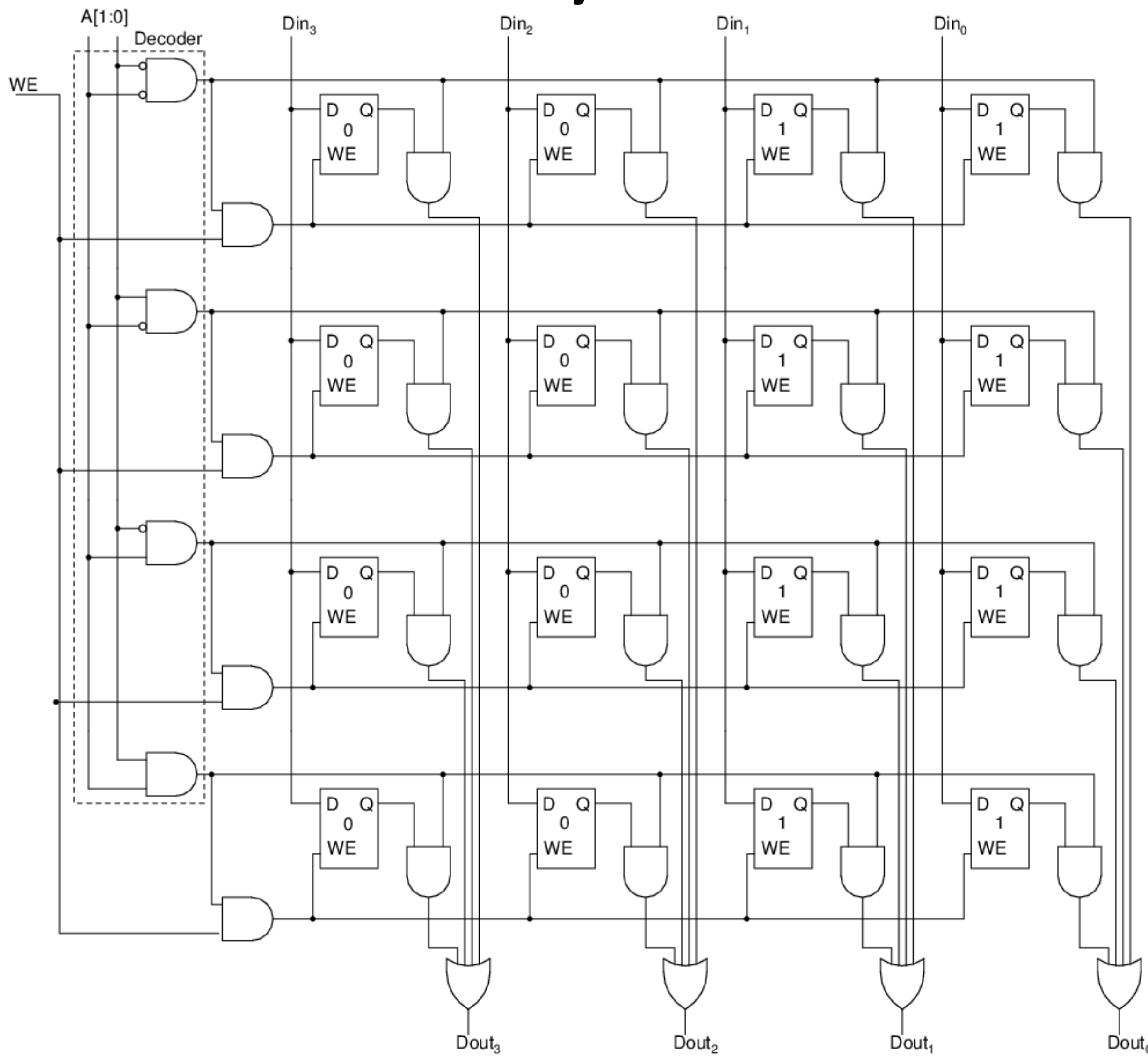
IR <- MDR

What do these do?

Memory

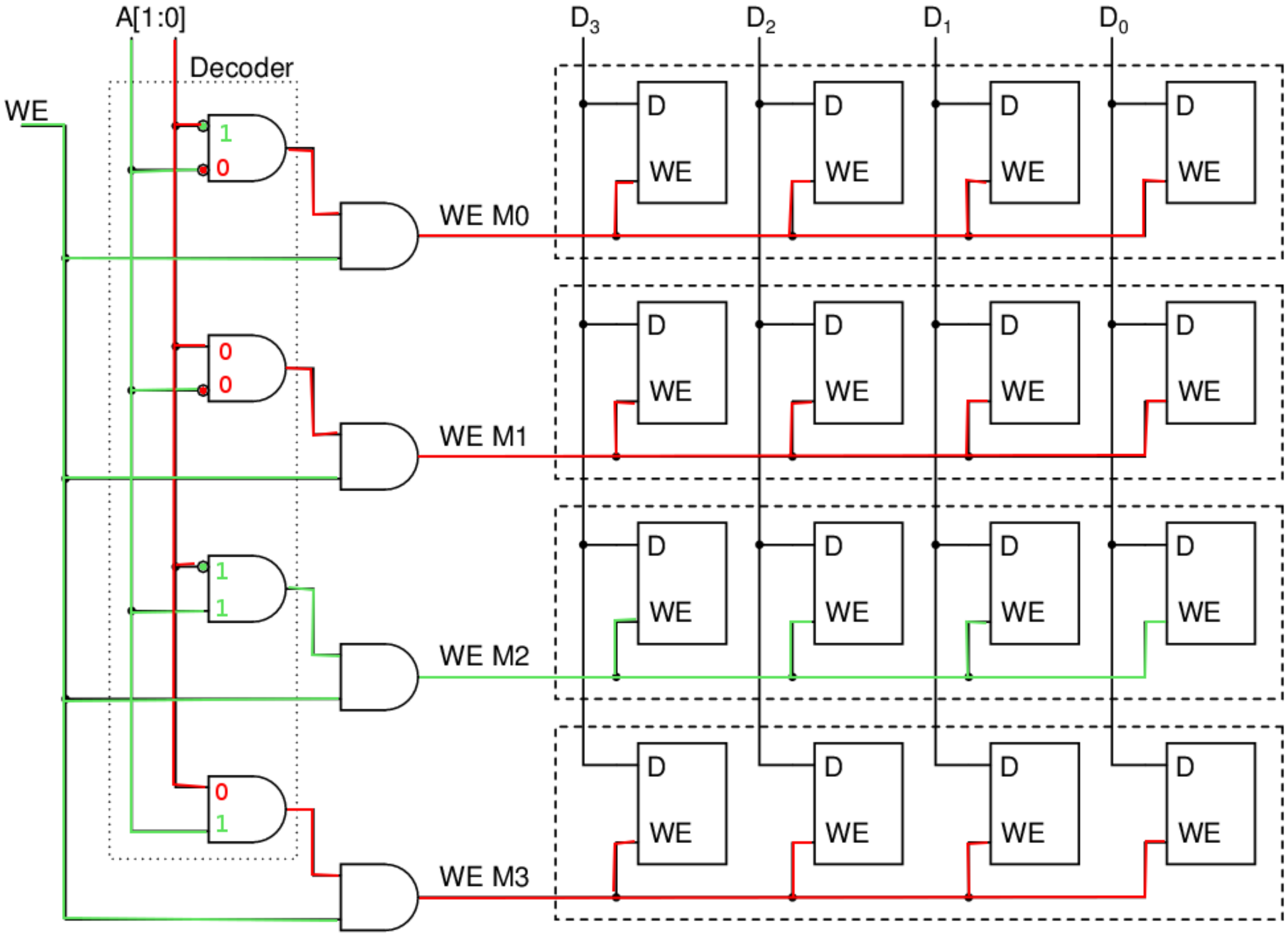
- There is a need to use more than one number in a computer.
- Memory gives the ability to store and retrieve many numbers.
- Write to memory
  - Must provide the data to store.
  - Must provide the address to store it in.
  - Must tell the memory to write and not read.
- Reading from memory
  - Must provide the address to read.
  - Must tell the memory to read and not write.
- Think of it as multiple registers working together. All that is needed is a way to pick out the correct register and determine read or write.

Memory Circuit

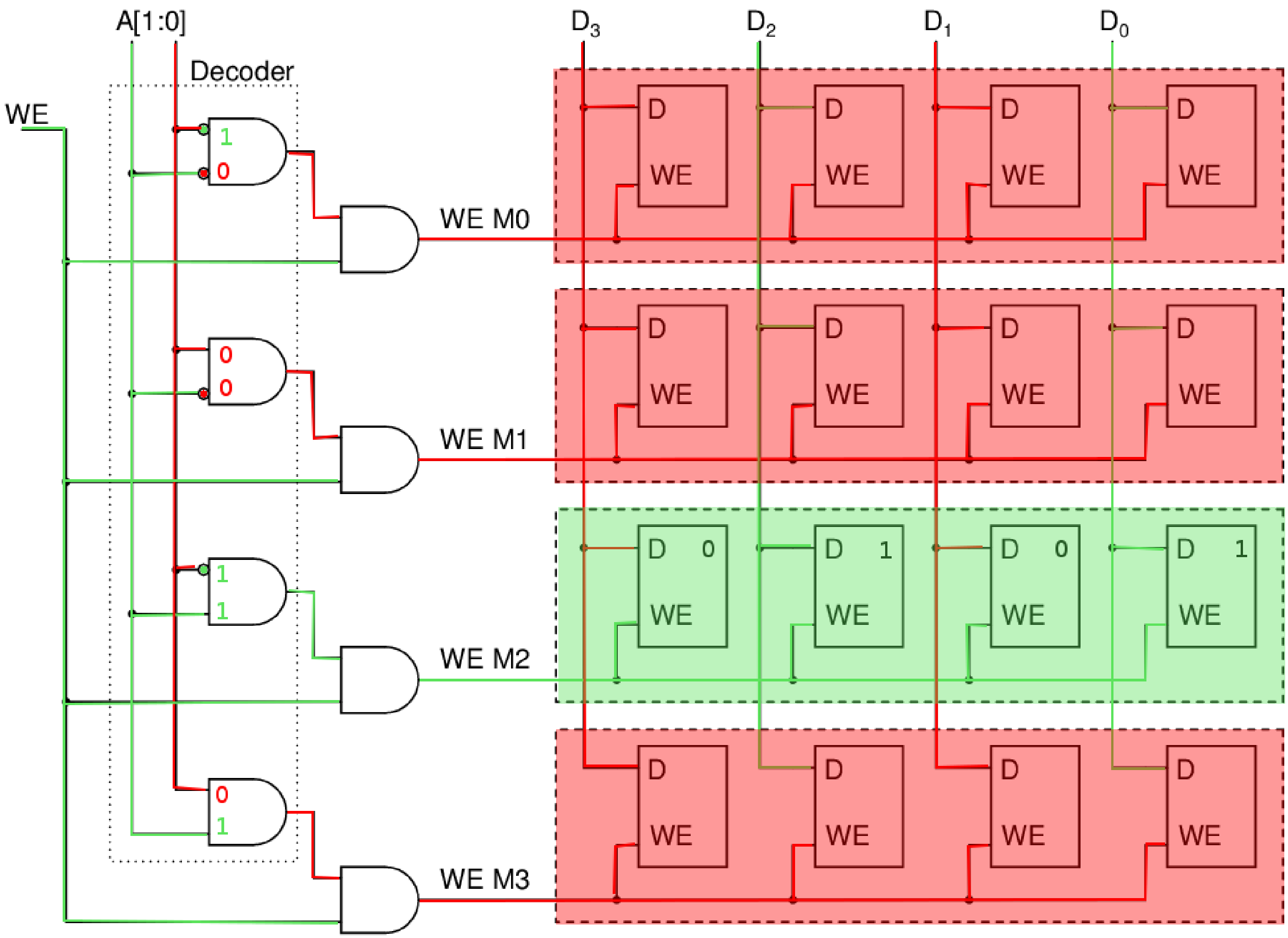


Writing

- Write the number 5 to address 2.

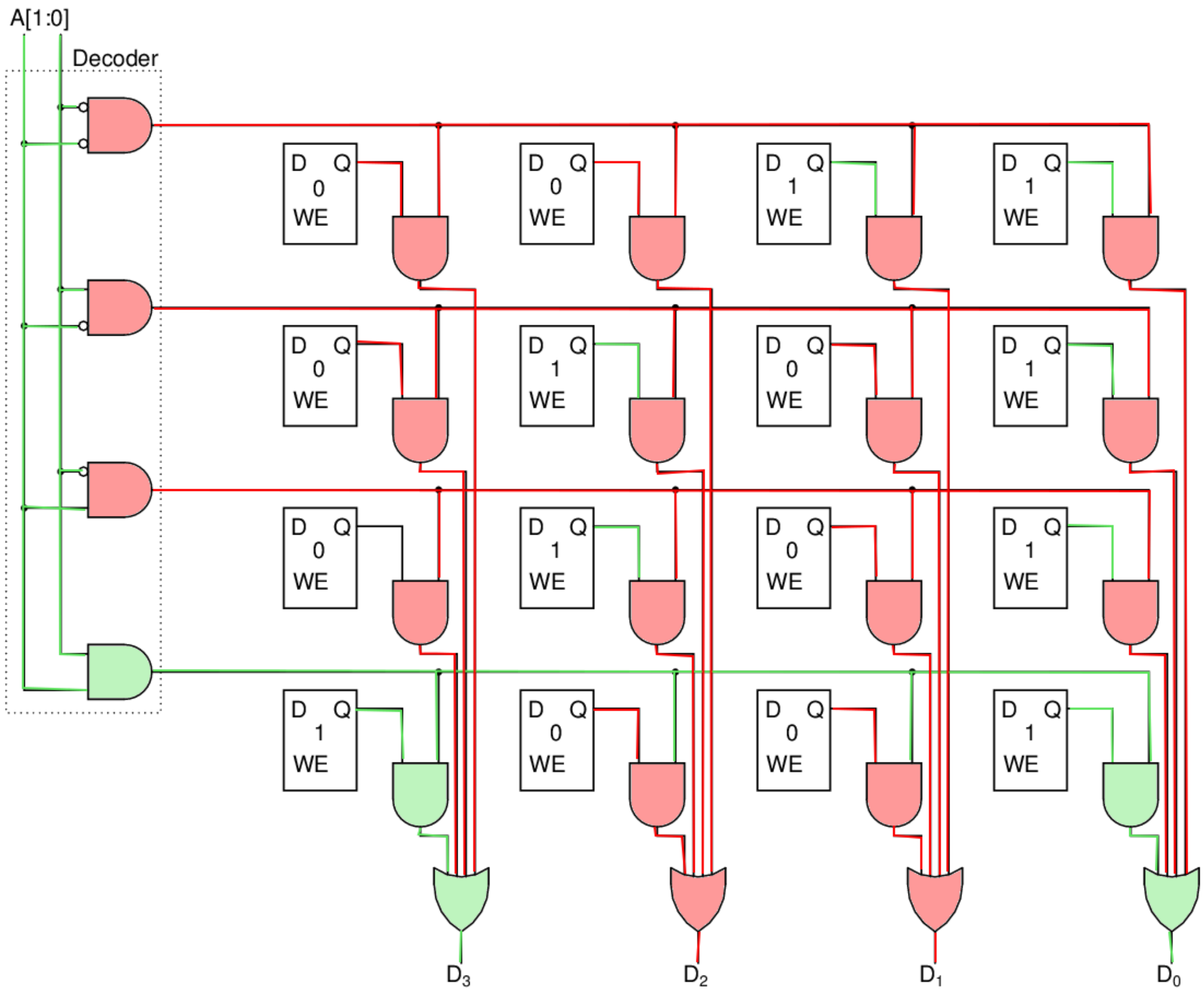


0101 Stored in Address 2

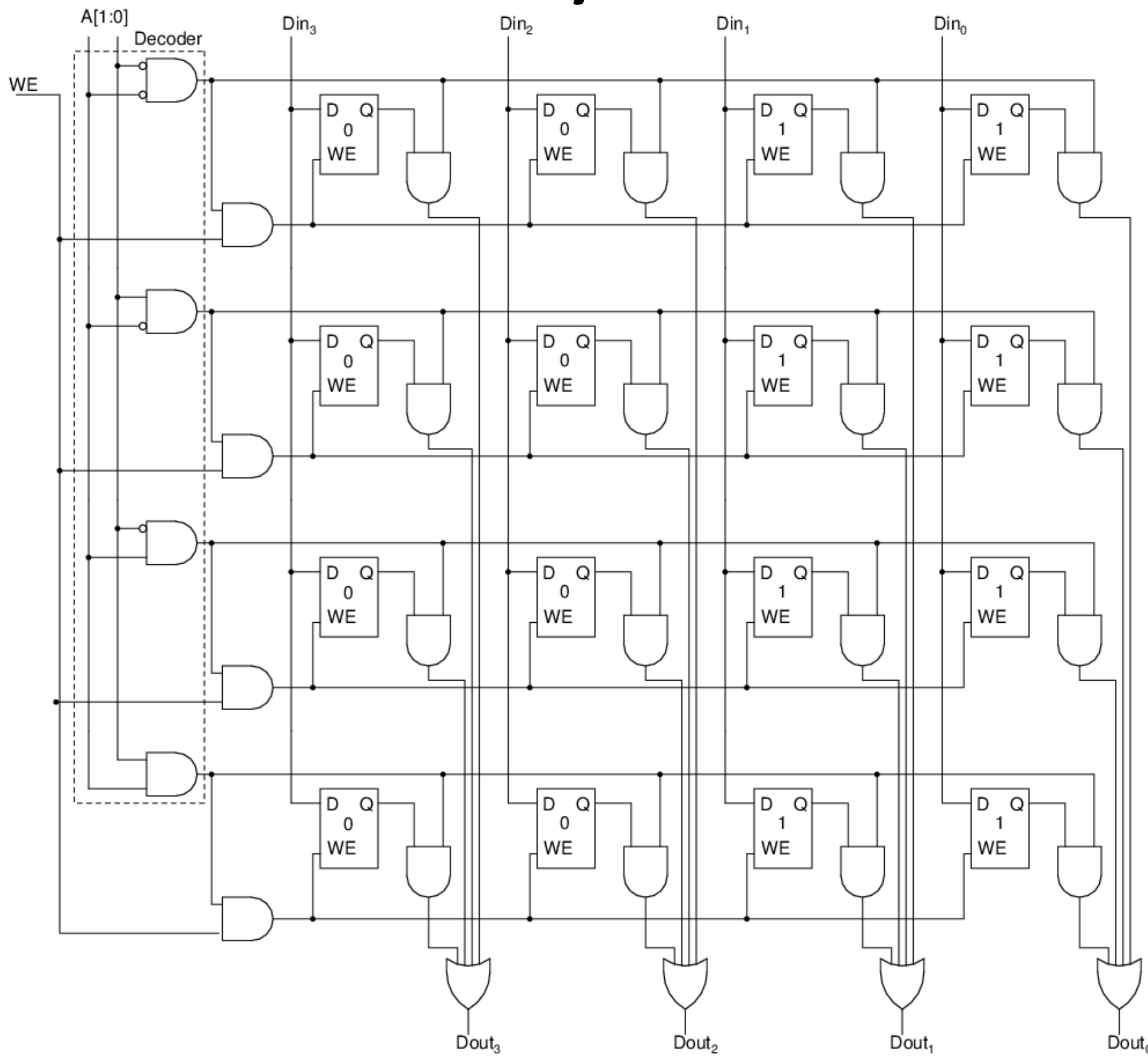


Reading

- Read value stored in address 3

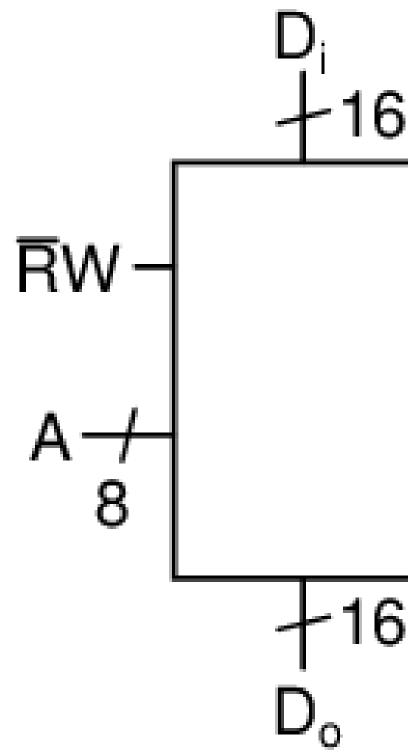


Memory Circuit

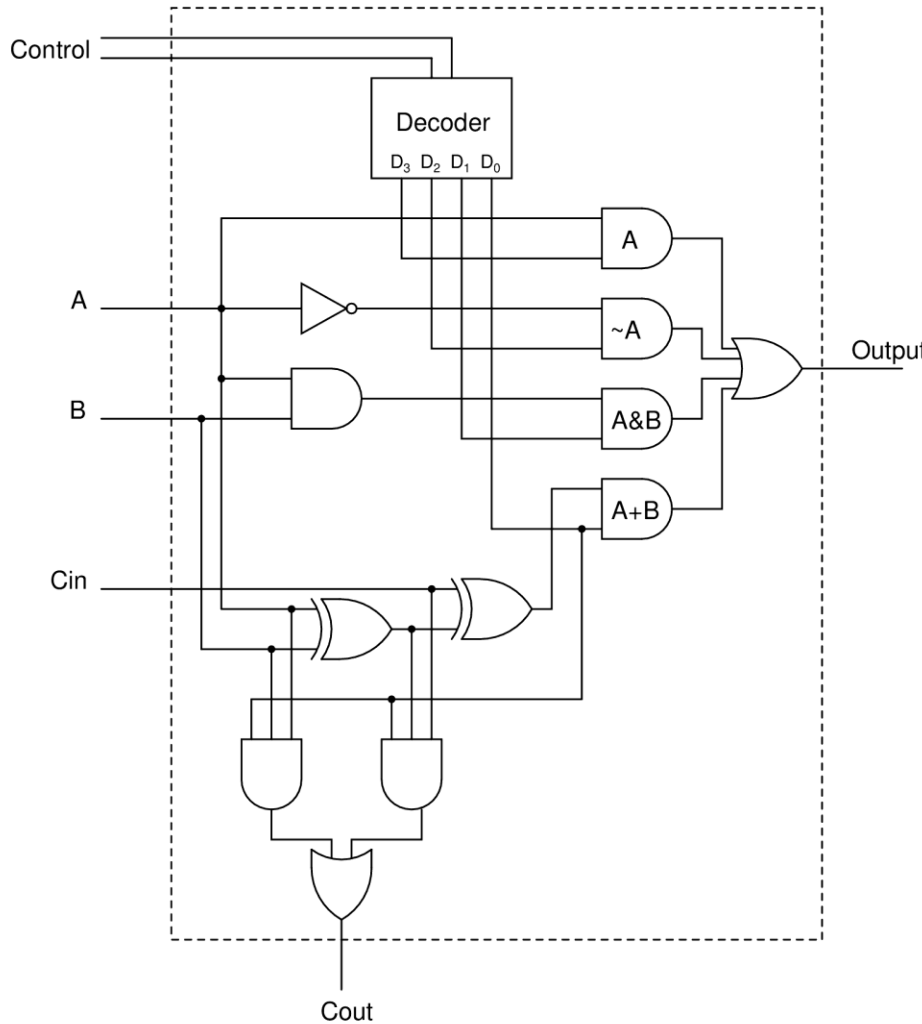


Memory Block

- Addressability - How big are the data units?
- Address Space - How many data units are there?



ALU

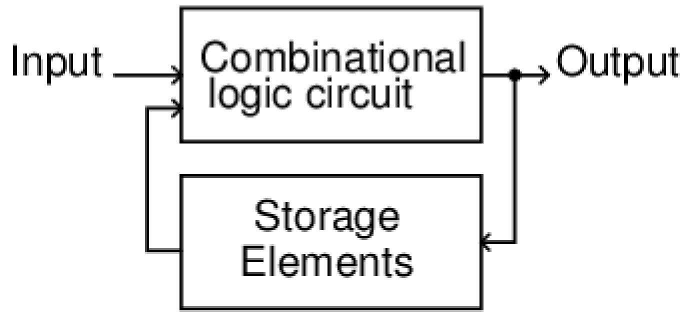


Sequential Logic Circuits

- The circuits we studied before are called ***Combinational Logic***.
- ***Combinational Logic*** takes some inputs and determines the output by tracing the gate outputs from start to finish.
- ***Sequential Logic*** specifies an ordered sequence of outputs. For the sequence to be ordered the circuit must somehow "know" the last set of outputs.
- ***Sequential logic*** has memory units (latches) and those memory units provide input to determine the next sequence or state.

Sequential Logic Block Diagram

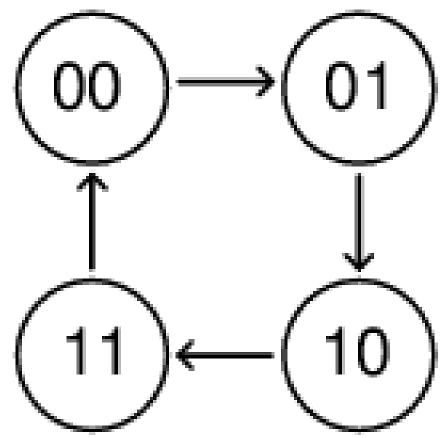
Contains a combinational portion as well as some storage elements for keeping track of the current state.



State Diagram

- Let's count from 0 to 3 in binary and repeat
- 00 -> 01 -> 10 -> 11

- This is a simple state diagram.



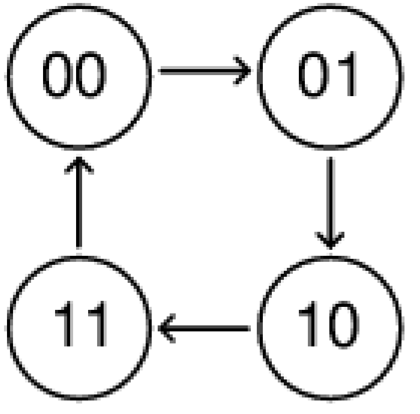
The states are specified by a labeled circle.
The transitions are shown by the arrows.

Finite State Machine

- A machine to generate all possible states and consists of:
  - a finite number of states
  - a finite number in external inputs
  - a finite number of external outputs
  - an explicit specification of all state transitions
  - an explicit specification of what determines an output

State Diagram With Input

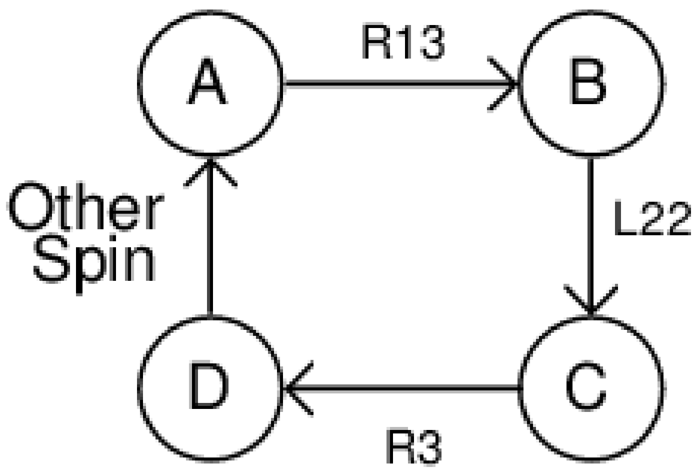
- Let's count from 0 to 3 in binary and repeat
- 00 -> 01 -> 10 -> 11
- This is a simple state diagram.



Combination Lock Example

- Four states
  - A – Incorrect turns – lock not open
  - B – Correct first turn - lock not open
  - C – Correct second turn - lock not open
  - D – Correct third turn - lock open
- The combination is R13, L22, R3
- The internals of the lock represent the memory in the positioning of the opening mechanism.

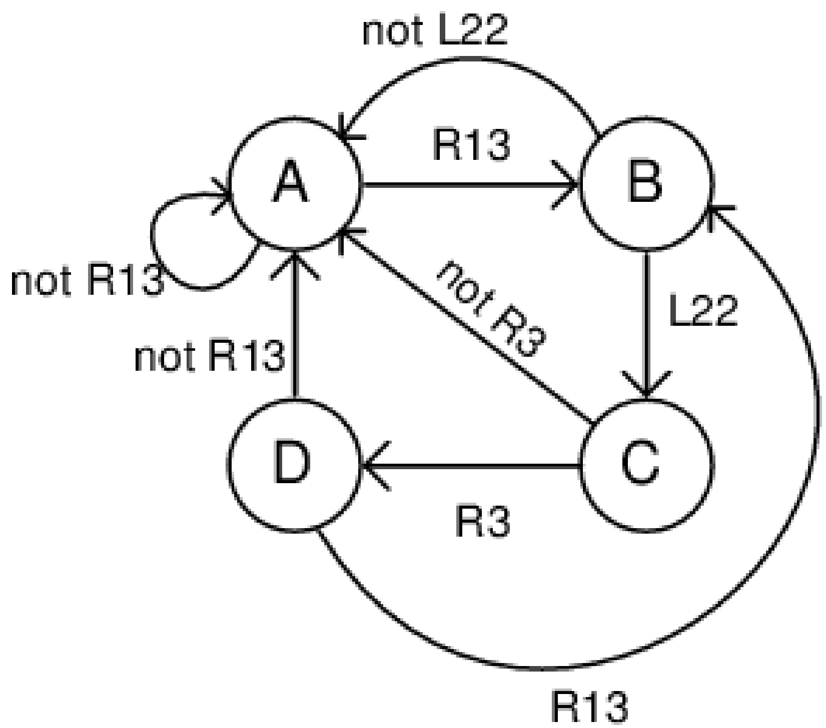
Combination Lock State Diagram



Are we done? Are all transitions labeled?

Combination Lock State Diagram Complete

- We must show ALL transitions.
- This is much more complicated than before.

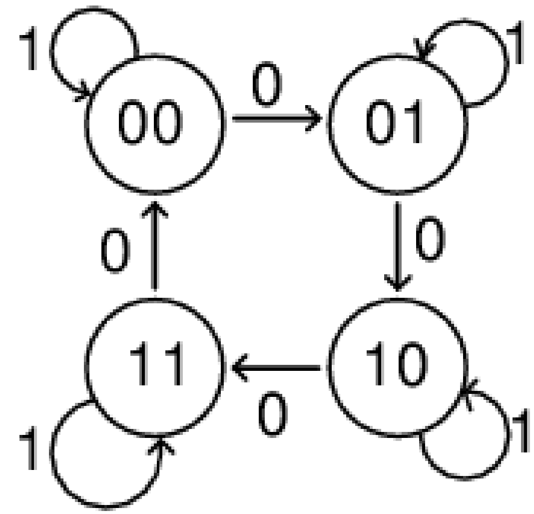


Example 2 bit patterns

- 2 bits can form four states. They won't necessarily be in order.
- 1 switch input (0=count, 1=pause).
- Build a count up circuit with pause.
  - Draw the state diagram.
  - Create a truth table showing next states.
  - Build the circuit for each latch input

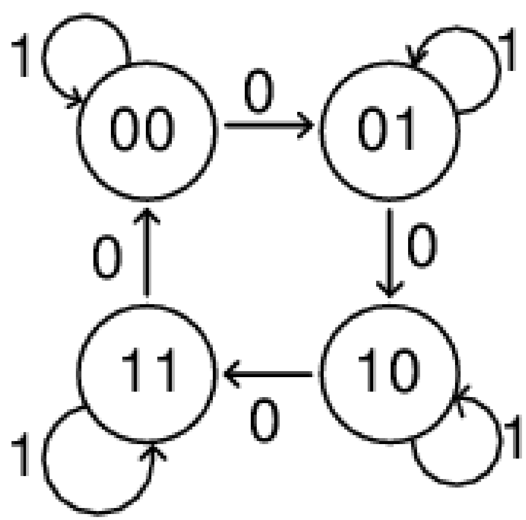
State Diagram

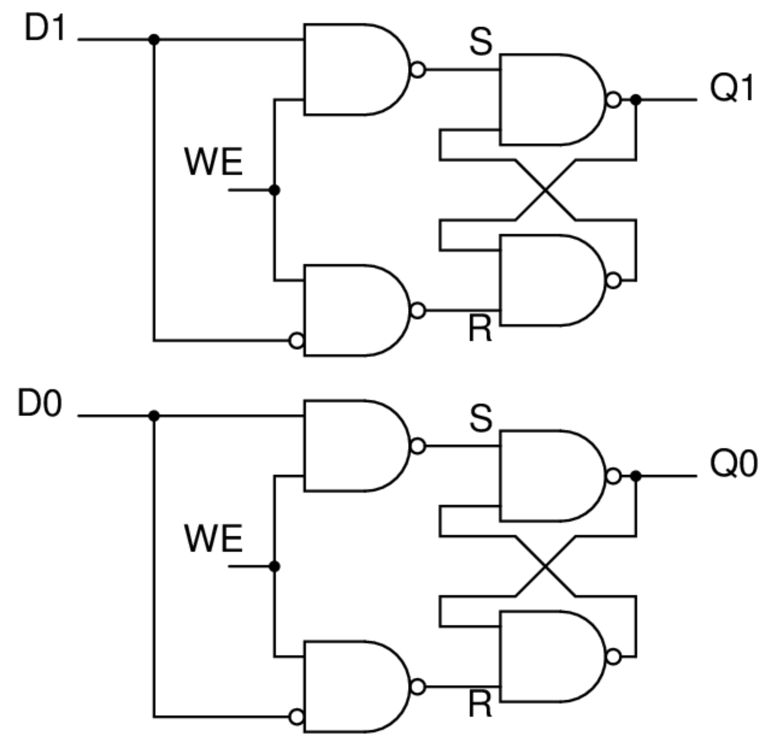
- Each state must have a transition of 1 and a transition of 0 OUT of the state.
- Each state is a pair of D latches with the output connected to LEDs.
- State 00 represents a zero in both latches with both LEDs being off.
- State 11 represents a 1 in both latches with both LEDs being on.
- State 01 and 10 represent a zero in one latch and a zero in the other. One of the LEDs will be on. The other LED will be off.



Latches and States Example 1

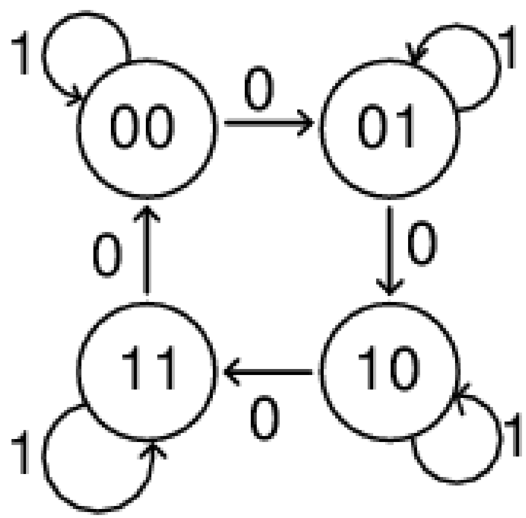
- Q1=0   Q2=0   Sw=0
- What do we need for proper state transition?

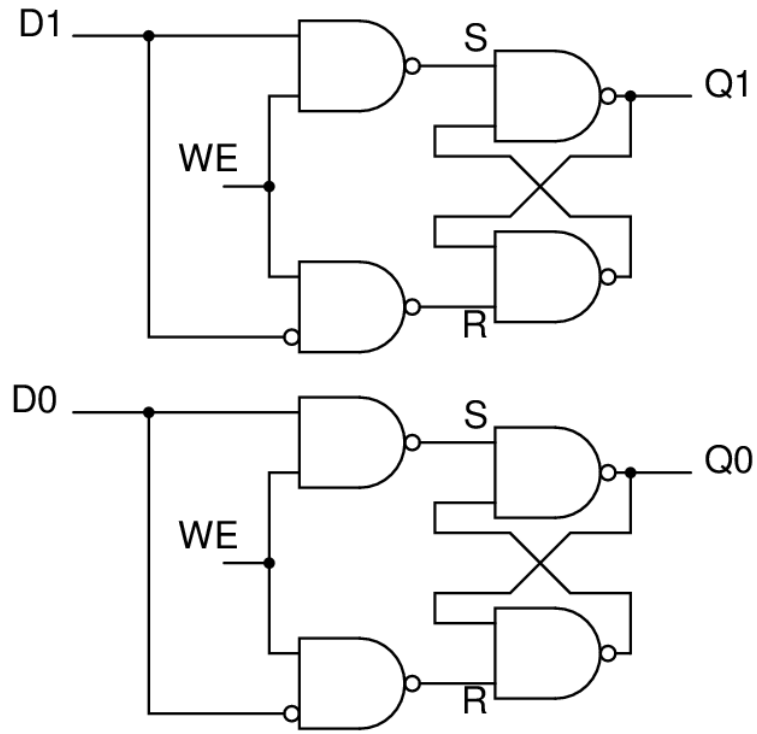




Latches and States Example 2

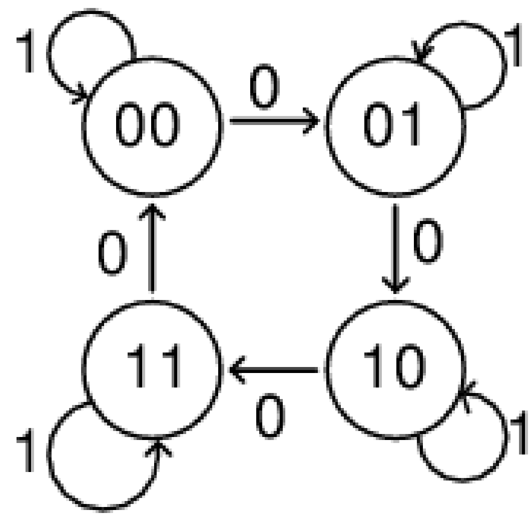
- Q1=1 Q2=0 Sw=1
- What do we need on D1 and D0 for proper state transition?





Count up with pause

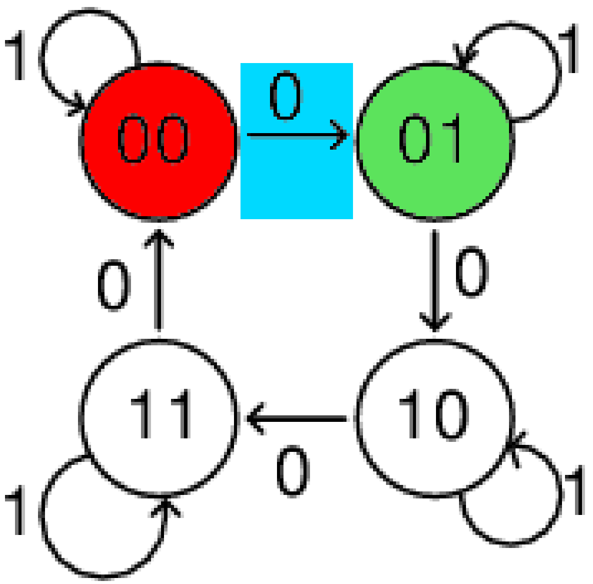
Current State			Next State	
Q1	Q0	Sw	D1	D0
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		



Q1 and Q0 are the current state.
The switch SW represent the transition.

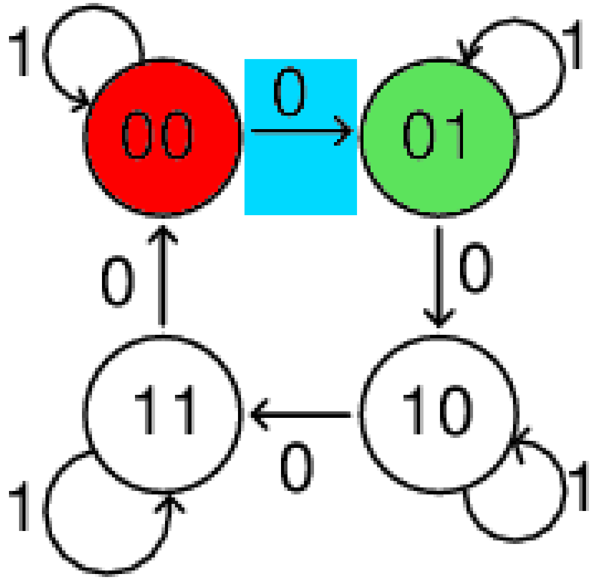
Count up with pause

Current State			Next State	
Q1	Q0	Sw	D1	D0
0	0	0	0	1
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		



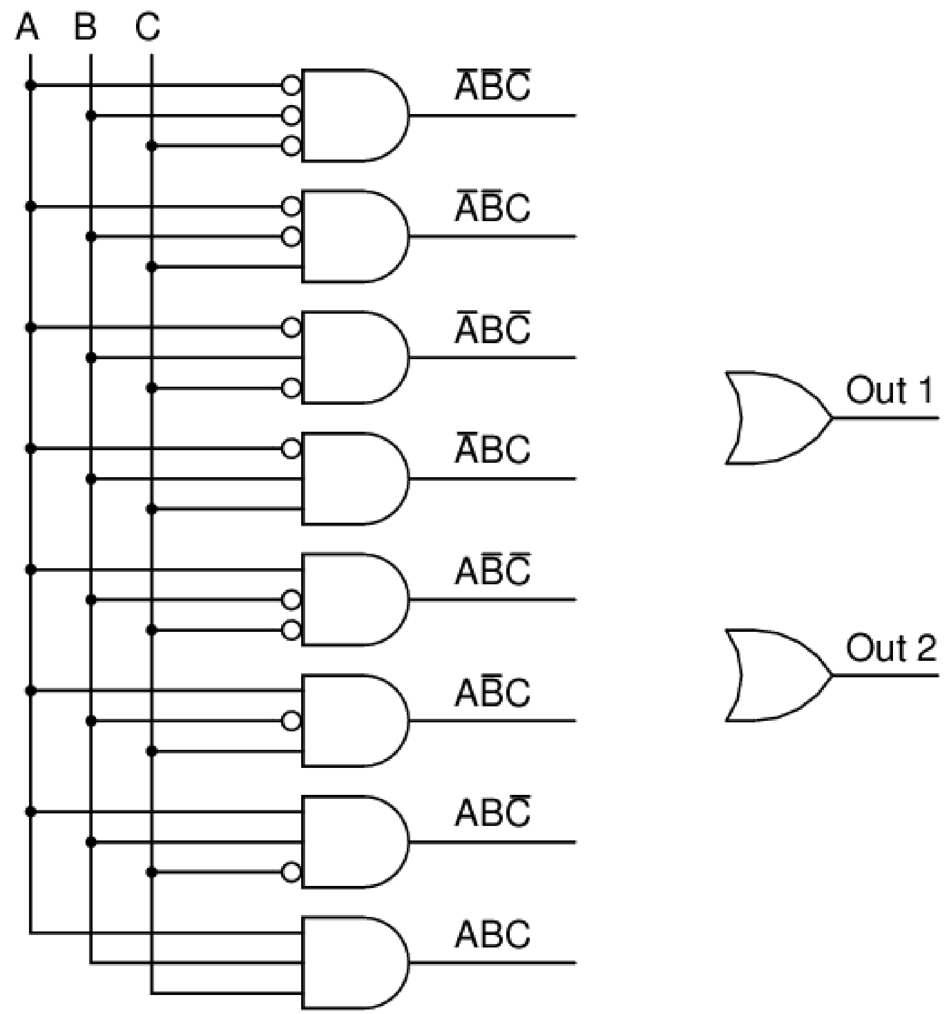
Count up with pause

Current State			Next State	
Q1	Q0	Sw	D1	D0
0	0	0	0	1
0	0	1	0	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	0
1	1	0	0	0
1	1	1	1	1



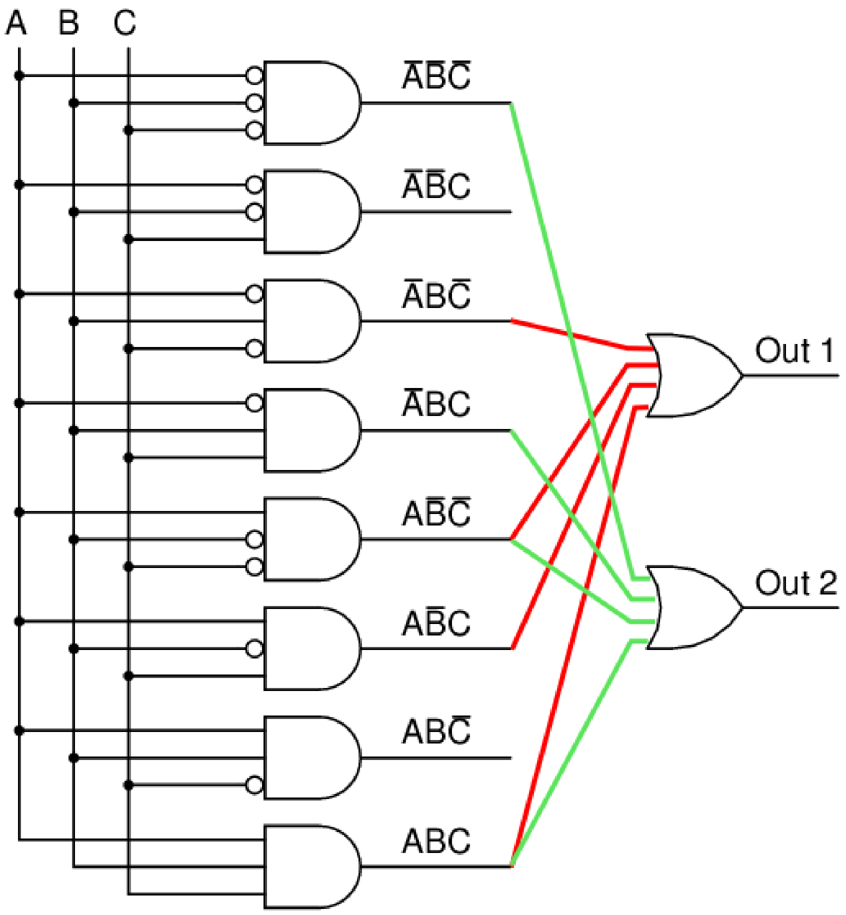
Count up with pause

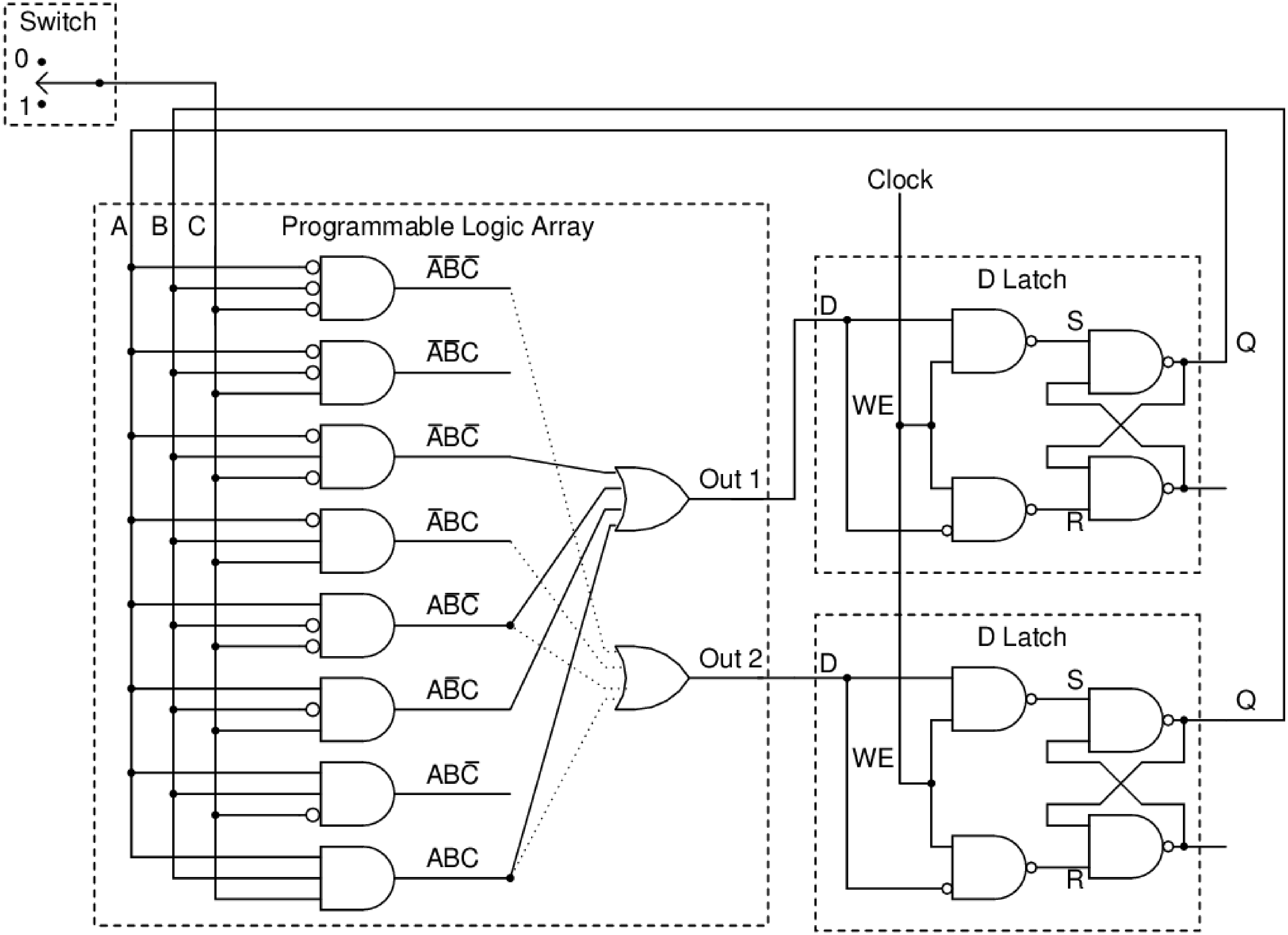
Current State			Next State	
Q1 (A)	Q0 (B)	Sw (C)	D1 (Out1)	D0 (Out 2)
0	0	0	0	1
0	0	1	0	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	0
1	1	0	0	0
1	1	1	1	1



Count up with pause

Current State			Next State	
Q1 (A)	Q0 (B)	Sw (C)	D1 (Out1)	D0 (Out 2)
0	0	0	0	1
0	0	1	0	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	0
1	1	0	0	0
1	1	1	1	1





Switch
0
1
A
B
C
Programmable Logic Array
$\overline{A}\overline{B}\overline{C}$
$\overline{A}\overline{B}C$
$\overline{A}B\overline{C}$
$\overline{A}BC$
$A\overline{B}\overline{C}$
$A\overline{B}C$
$AB\overline{C}$
ABC
Out 1
Out 2
Clock
D Latch
D
WE
S
R
Q
D Latch
D
WE
S
R
Q

Other Circuits

- Good test questions.
  - Count up and repeat
    00 - 01 - 10 - 11 - 00 - 01 - …
  - Count up and stop at 11
  - Count down and repeat
    11 - 10 - 01 - 00 - 11 - 10 - …
  - Count down and stop at 00
  - Both on – both off
    00 -11 - 00 - 11 -…
  - Alternate
    01 - 10 - 01 - 10 - …
  - Pause, remain in current state.
  - others …
- Could do any of the above on sw = 0 and any other on sw = 1

Problem?

- There is a bit of a problem with using the D-Latch as shown with the clock.
- The output changes, which changes the inputs, which will change the output, etc.. as long as the clock is high.
- Use a master slave flip-flop to prevent the problem.

Master – Slave Flip-flop

- First is set when clock goes high.
- Second is set from first when clock goes low.
- When clock is low, first cant be modified.