university of groningen

Computer Architecture 2023-24 (WBCS010-05)

Lecture 3: Digital Logic Structures

Reza Hassanpour
r.zare.hassanpour@rug.nl

Topics

- Digital Logic
- Logic Gates
- Combinational Logic
  - Multiplexer
  - Decoder
  - Adder
- Sequential Logic
  - S-R Latch
  - D Latch

Digital Computers

- We use the term digital computer to refer to a device that performs a sequence of computational steps on data items that have discrete values.
- Digital logic is the manipulation of binary values through technology that uses circuits and logic gates to construct the implementation of computer operations.

Boolean Logic

- Mathematical basis for digital circuits
- Three basic functions: *and*, *or*, and *not*

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

A	NOT A
0	1
1	0

Digital Logic

- Is the implementation of Boolean functions with transistors where:
  - Five volts represents Boolean 1 (true)
  - Zero volts represents Boolean 0 (false)
- Voltage:
  - Quantifiable property of electricity in the form of the pressure that pushes charged electrons (current) through a conducting loop.
  - Unit: volt
- Current
  - Measure of electron flow along a path
  - Unit: ampere (amp)

Voltage/Current Analogy

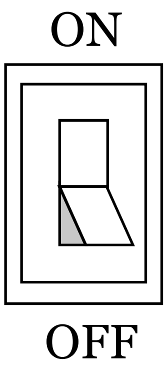
- Voltage is analogous to water pressure
- Current is analogous to flowing water
- Water can have
  - High pressure with little flow
  - Large flow with little pressure

Transistor: Building Block of Computers

› Microprocessors contain billions of transistors
  - Intel Broadwell-E5 (2016): 7 billion
  - IBM Power 9 (2017): 8 billion
  - Intel Ponte Vecchio (2021): 100 billion (is it a CPU?)

› Logically, each transistor acts as a **switch**

› Combined to implement **logic functions**
  - AND, OR, NOT, …



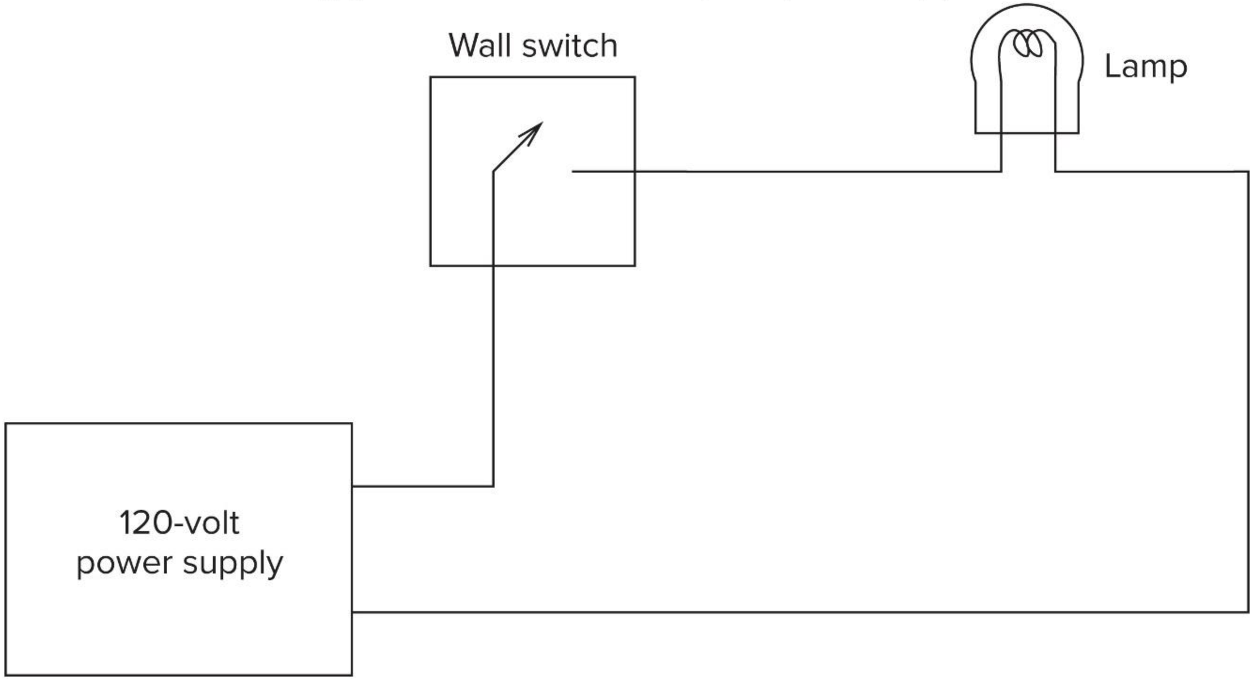
Combined to build **higher-level structures**

- Adder, multiplexer, decoder, register, …

Combined to build **processors**

A Simple Switch Circuit

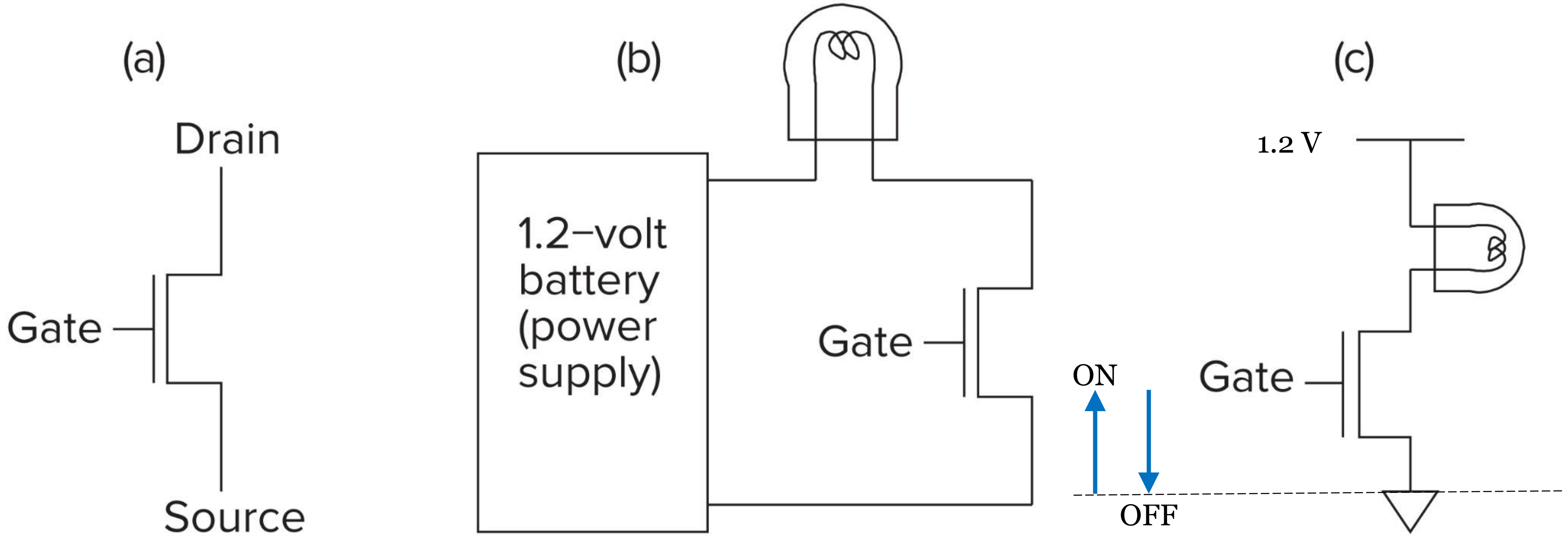
- A wall switch determines whether current flows through the light bulb



- If switch is closed, current flows, lamp is **ON**, voltage across lamp is **non-zero**
- If switch is open, no current flows, lamp is **OFF**, voltage across lamp is **zero**

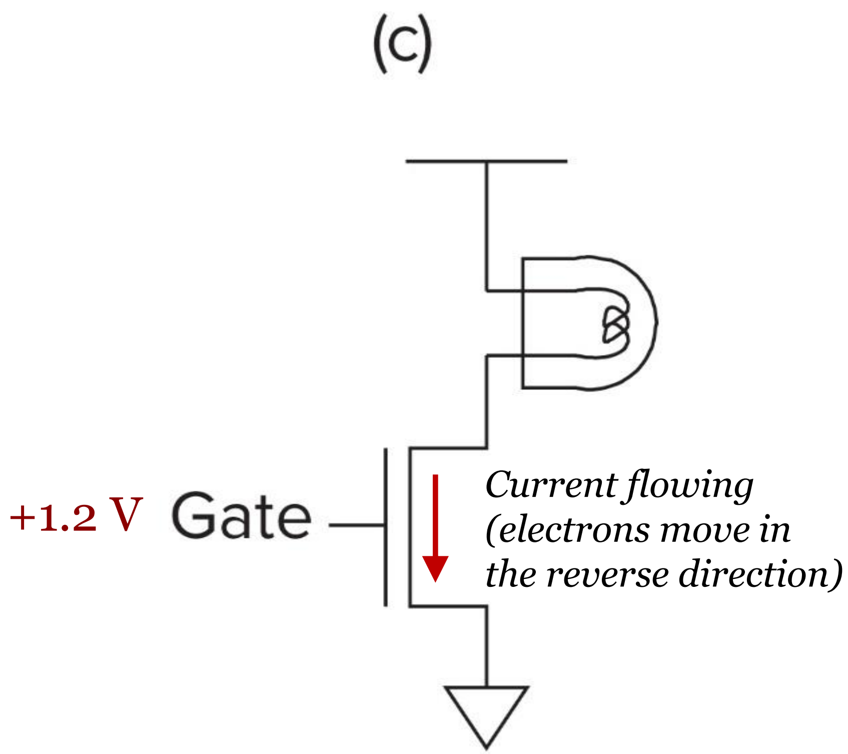
Transistor = Voltage-Controlled Switch 1

- Figure shows an **N-type** transistor
  When Gate voltage is **positive**, relative to Source, transistor acts as a short circuit: a **closed** switch
- When Gate voltage is zero (or negative), relative to Source, transistor acts as an open circuit: an open switch



Transistor = Voltage-Controlled Switch 2

› Consider the circuit below. The bar at the top represents the high voltage rail and the triangle at the bottom represents ground (0V)



› **When Gate = +1.2V, what happens?**

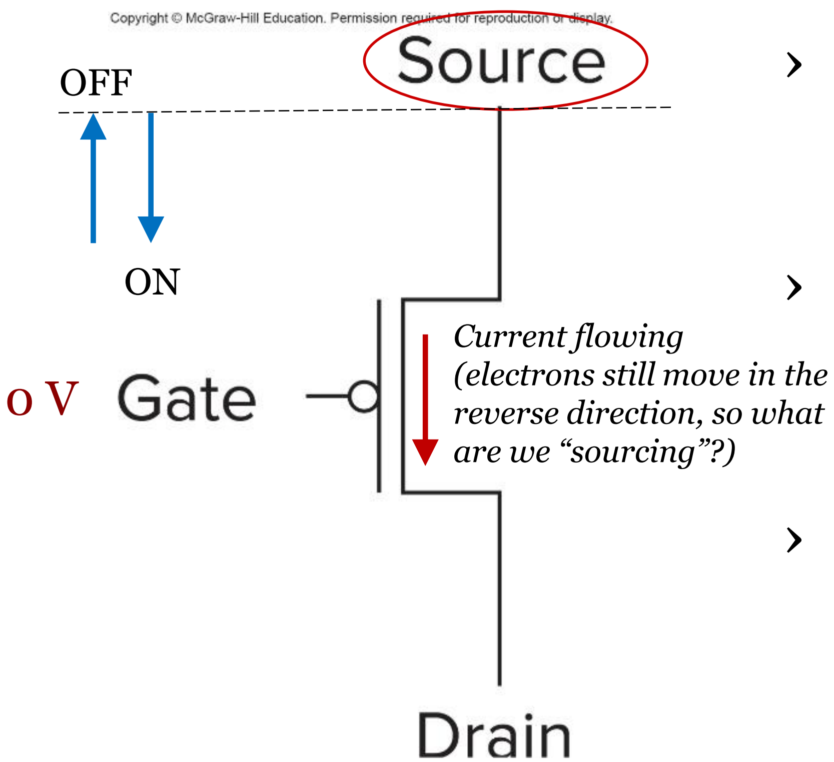
- Gate-to-source voltage > 0
- Transistor = closed switch
- Current flows, lamp is **ON**

› **When Gate = 0V, what happens?**

- Gate-to-source voltage = 0
- Transistor = open switch
- No current flows, lamp is **OFF**

Transistor = Voltage-Controlled Switch 3

- A different type of transistor is shown below, the **P-type** transistor. Notice the little "bubble" on the gate



- When Gate voltage is **zero** (or negative), <u>relative to Source</u>, transistor acts as a short circuit: a **closed** switch
- When Gate voltage is positive, relative to Source, transistor acts as an open circuit: an open switch
- NOTE: This behavior is the opposite of the N-type. Behavior is <u>complementary</u>

- We use both N-type and P-type transistors together to implement logic gates. This is known as **CMOS** or Complementary MOS logic

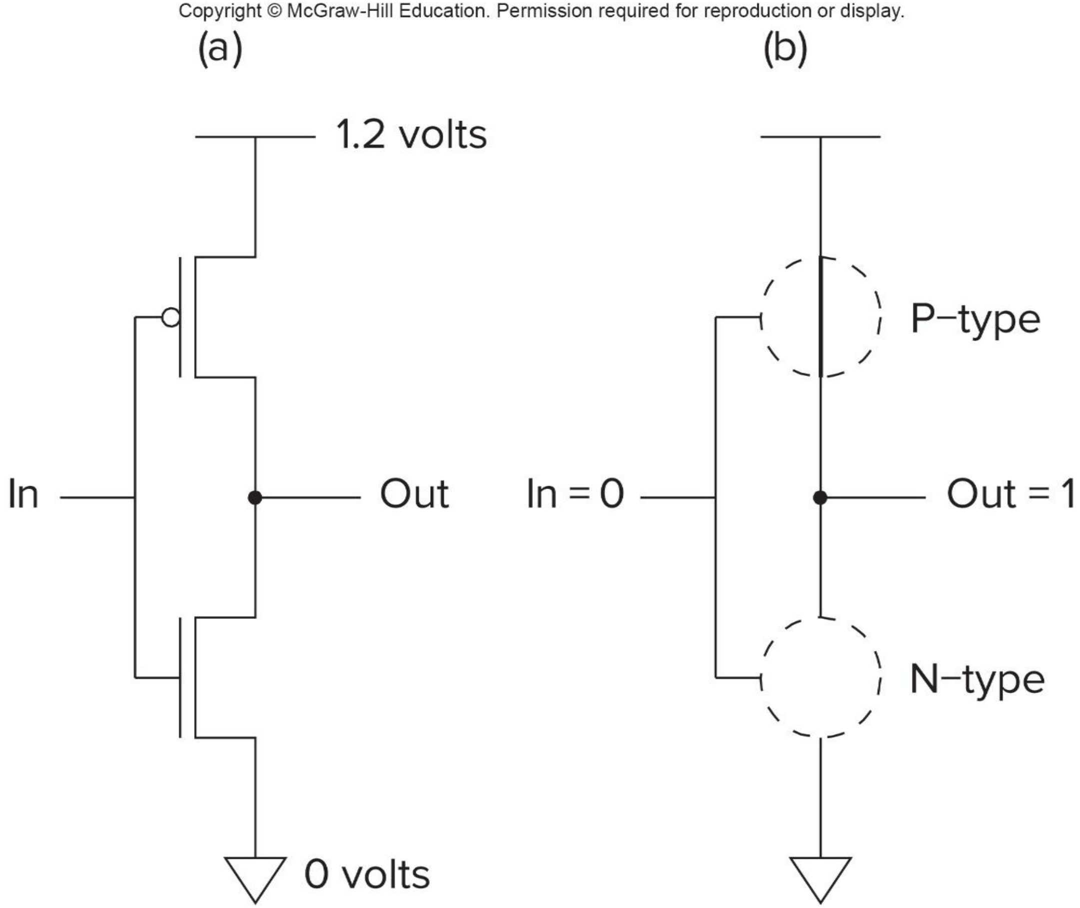
Logic Gate

- Transistors have two possible states:
  - current is flowing or
  - no current is flowing.
- Therefore, circuits are designed using a two-valued mathematical system known as Boolean algebra.

- A logic gate is a circuit implementation of Boolean function.

- Signals are the voltages.

NOT Gate (Inverter)



- Example:
- When input = 0, P-type transistor turns on and N-type transistor turns off. Output is connected to +1.2V, so output = 1

- Logic gate is described using a **truth table**

Input	Output
0	1
1	0

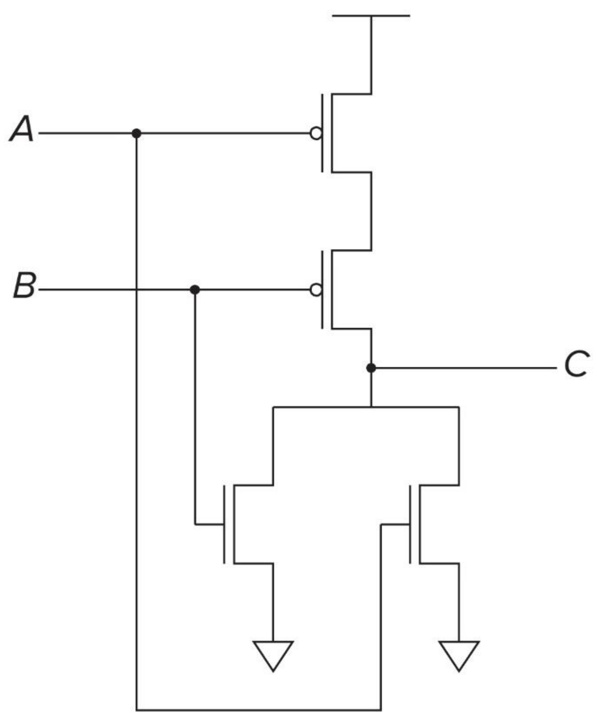
NOR Gate

› When either input is 1, output is 0

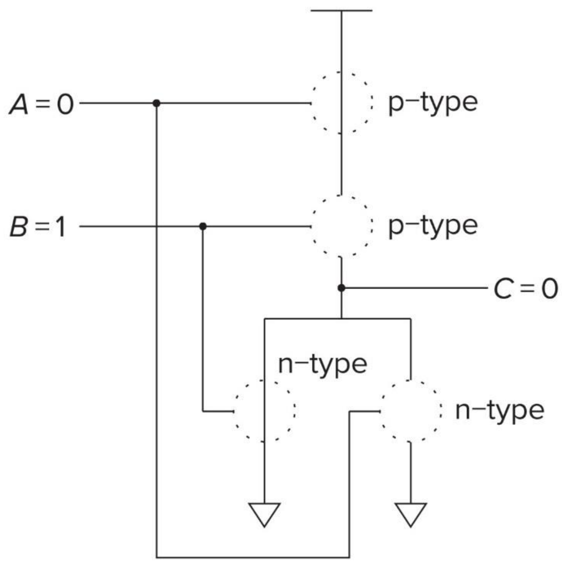
› Example:

› When B = 1, N-type transistor turns on and output (C) is connected to GND. Both inputs must be 0 to connect C to +1.2V.

(a)



(b)



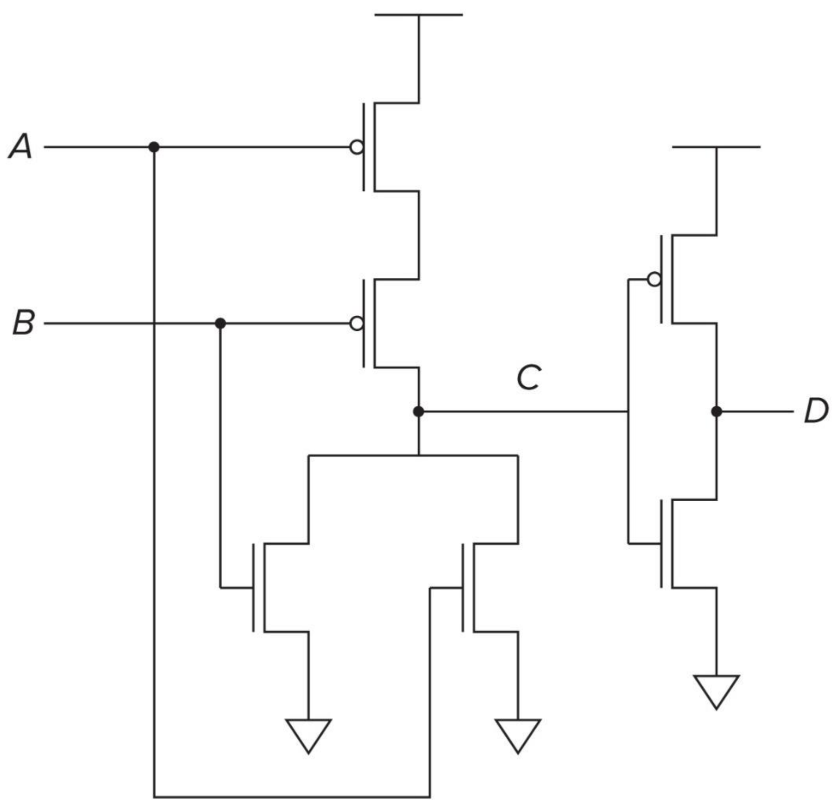
A	B	C
0	0	1
0	1	0
1	0	0
1	1	0

C = **NOT**(A **OR** B)

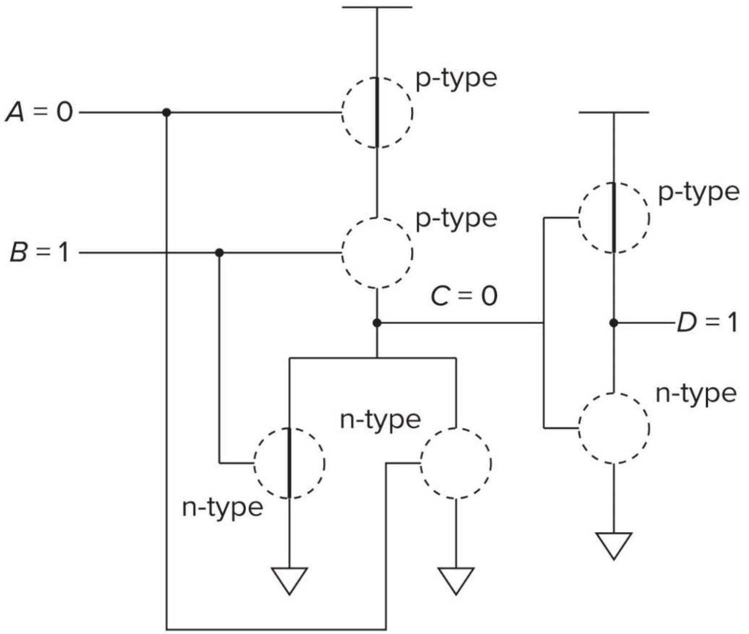
OR Gate

› When either input is 1, output is 1
Add NOT after NOR

(a)



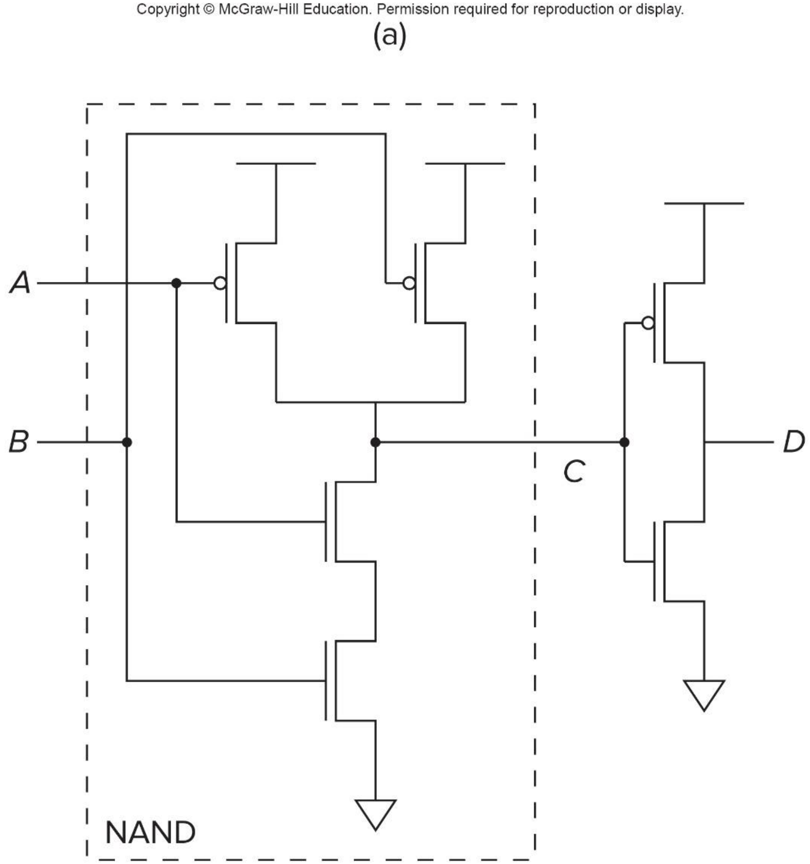
(b)



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

NAND and AND Gates

- NAND: When any input is 0, output is 1
  AND: When all inputs are 1, output is 1



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

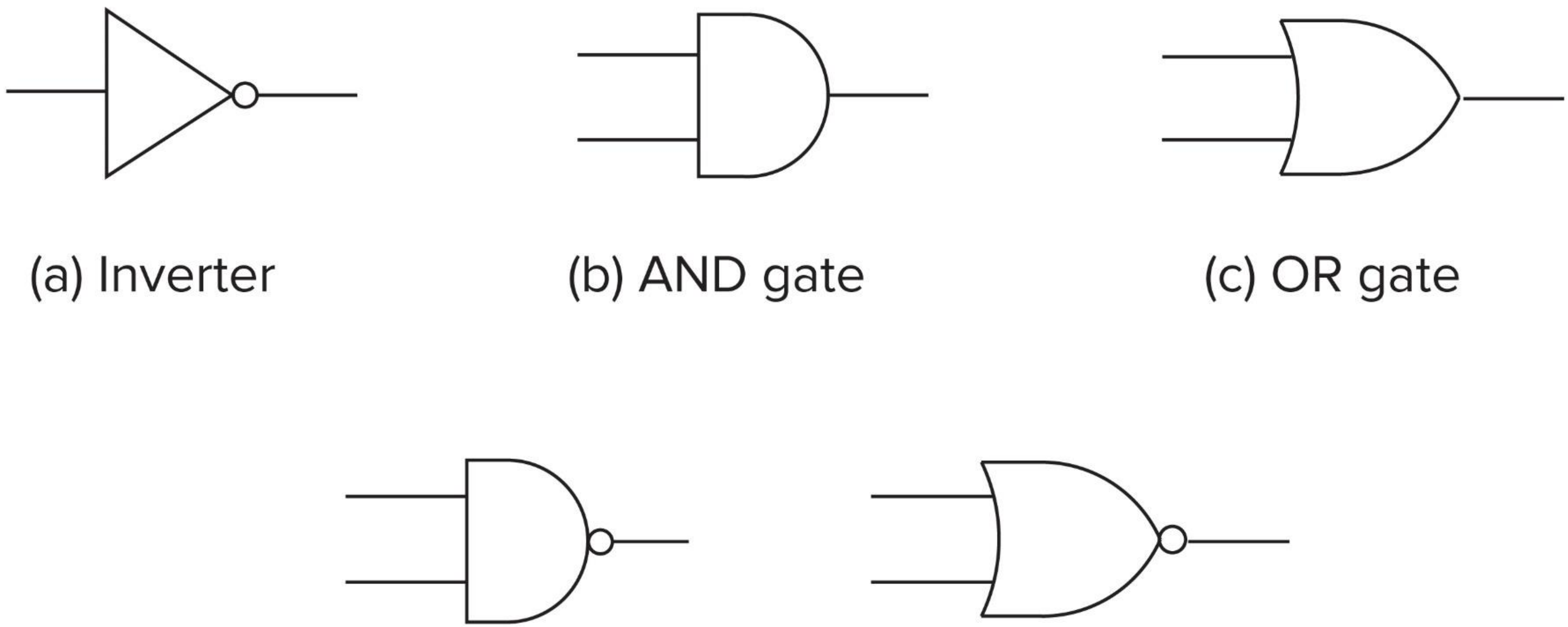
- NAND

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

- AND

Standard Symbols for Logic Gates

- Instead of drawing the circuit diagram, we can **abstract** these logic gates and give each a symbol. The bubble indicates inversion (NOT).



(a) Inverter (b) AND gate (c) OR gate

(d) NAND gate (e) NOR gate

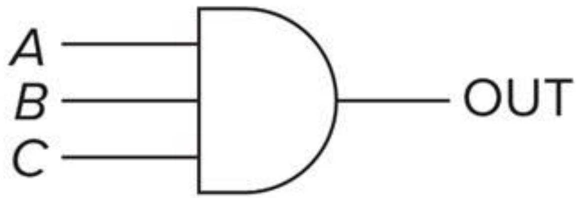
Gates with more than one input

- Notion of AND and OR generalizes to more than two inputs.
- AND: output = 1 if ALL inputs are 1
- OR: output = 1 if ANY input is 1

(a)

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

(b)



Challenge: *Draw the CMOS circuit for a 3-input AND gate*
What about 4-input NOR gate?

Building Logic Circuits from Gates

› To build more complex logic functions, we create circuits using the basic logic gates

› **Combinational** Logic Circuits

- Output depends only on the current input values
- Stateless -- no "memory" of the past

› **Sequential** Logic Circuits

- Output depends on both past and current input values
- "remembers" inputs from the past
  - **Example:** output = 1 if we see four zero inputs in a row

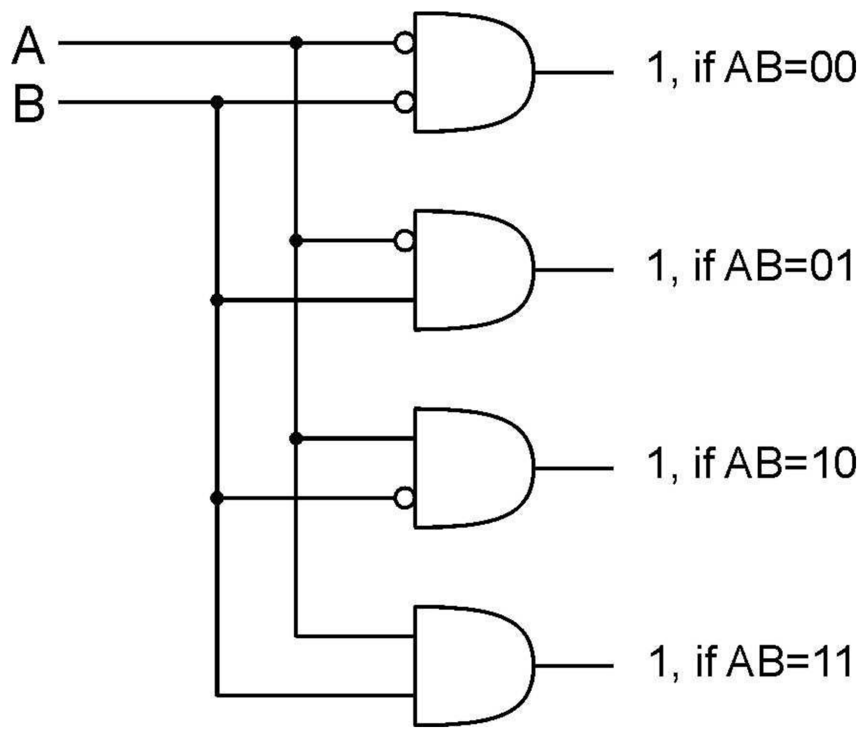
university of groningen

Combinational Logic Circuits

- We need **both** combinational and sequential circuits to build a computer
- First, we will introduce three useful combinational circuits:
- **Decoder**: recognizes specific bit patterns
- **Multiplexer**: chooses among various inputs
- **Adder**: performs addition on unsigned or 2's complement integers

Decoder

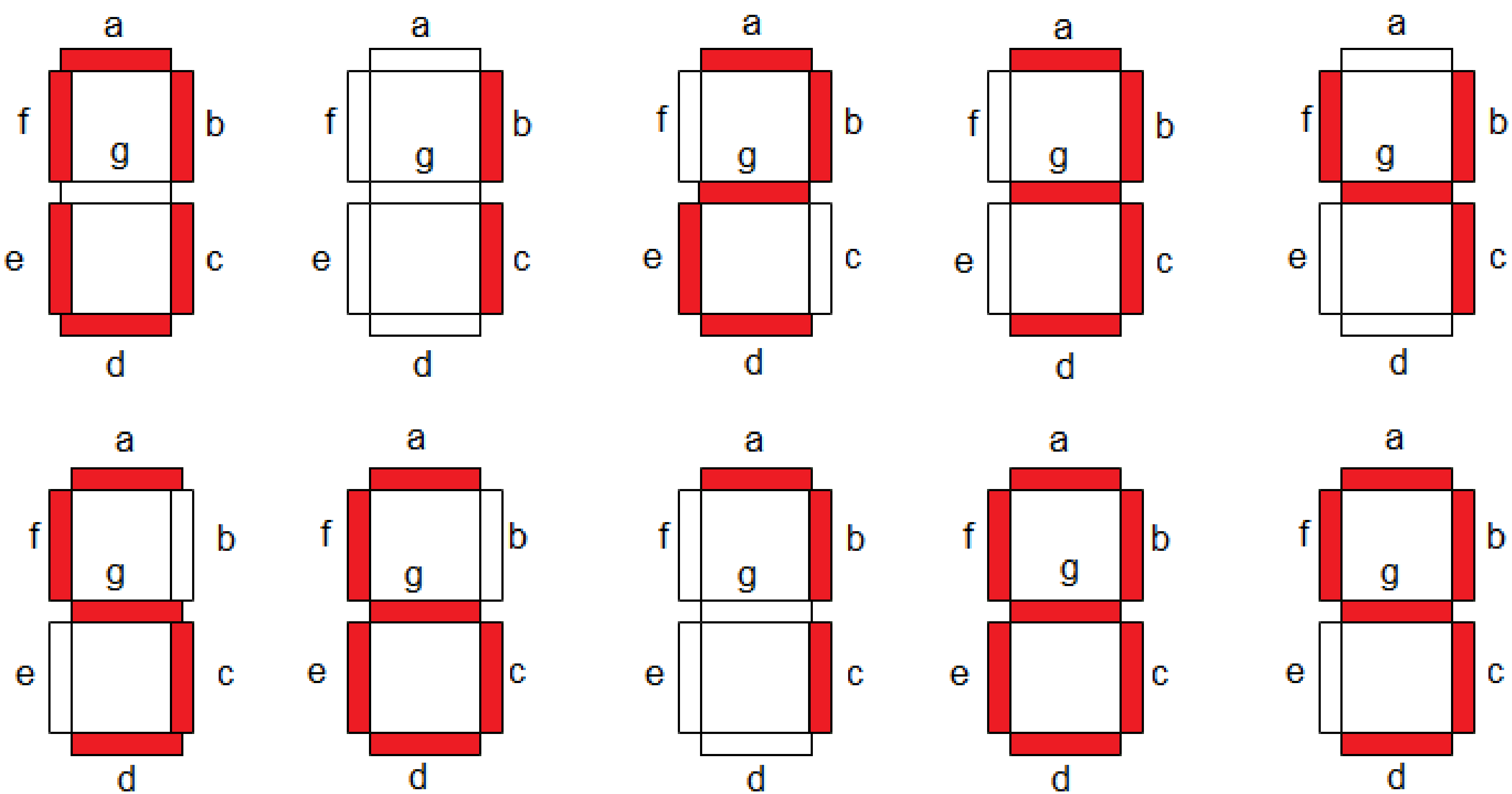
- A decoder has n inputs and 2^n outputs
- Each output corresponds to one possible input combination
- Exactly one output will be 1, and all others will be 0



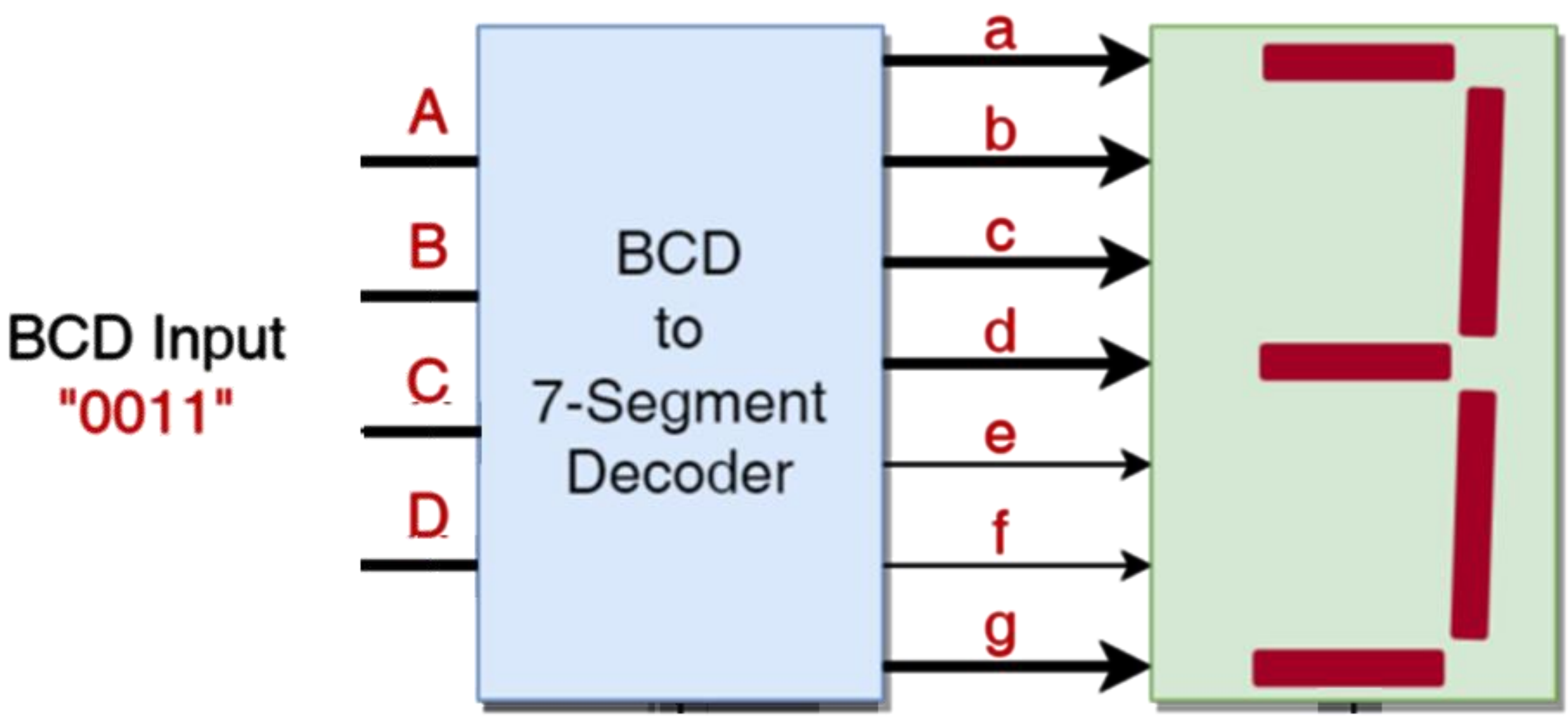
2-bit decoder

Useful for converting encoded values (integers) into a set of control signals

Example: Using Decoders to Display Numbers on a Seven-Segment Display

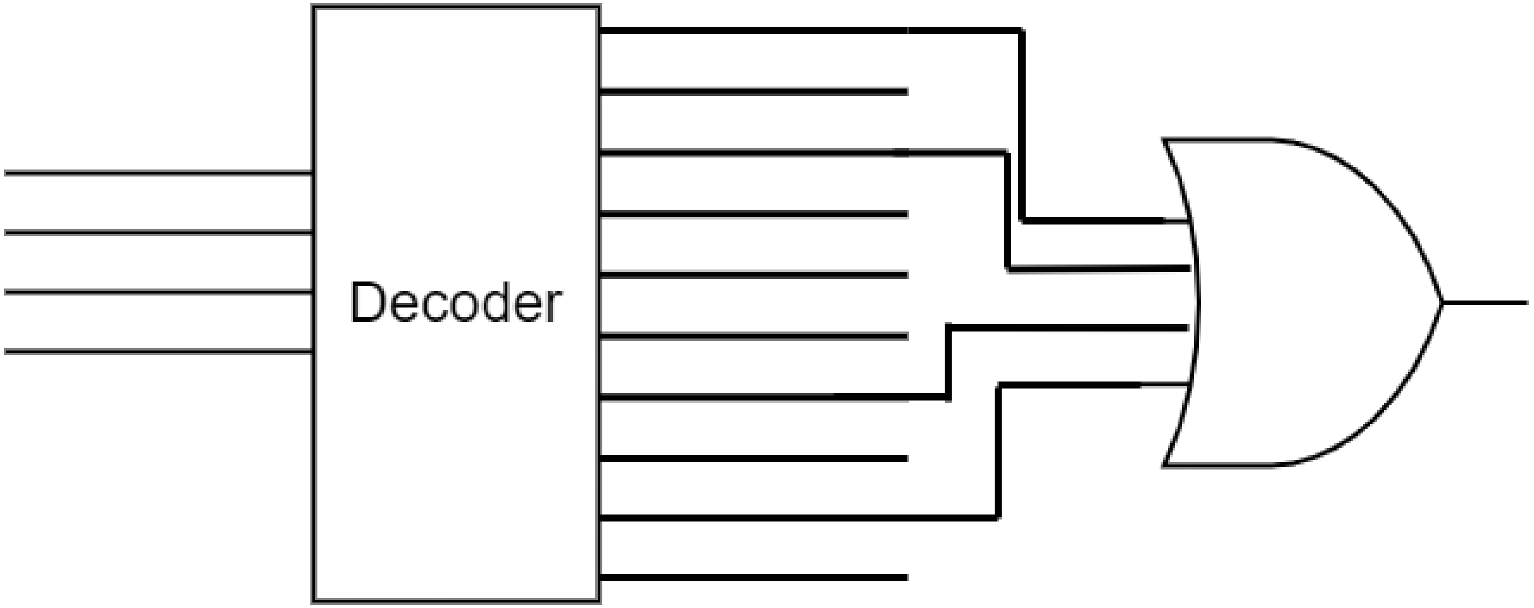


Decoding BCD = 3



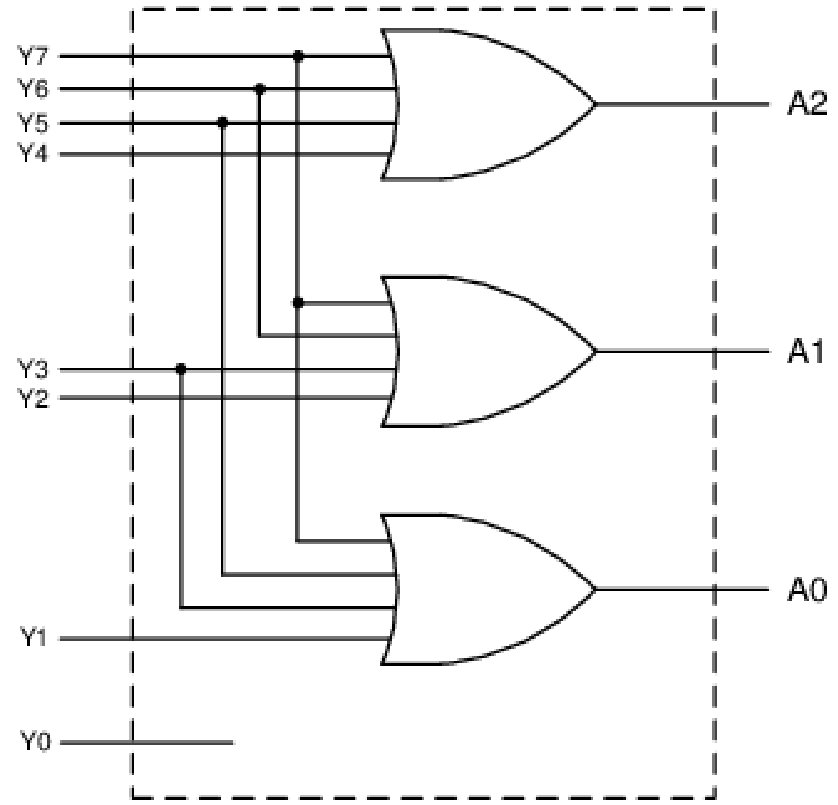
Turn On Segment e

e should be turned on when the input is 0, 2, 6, or 8



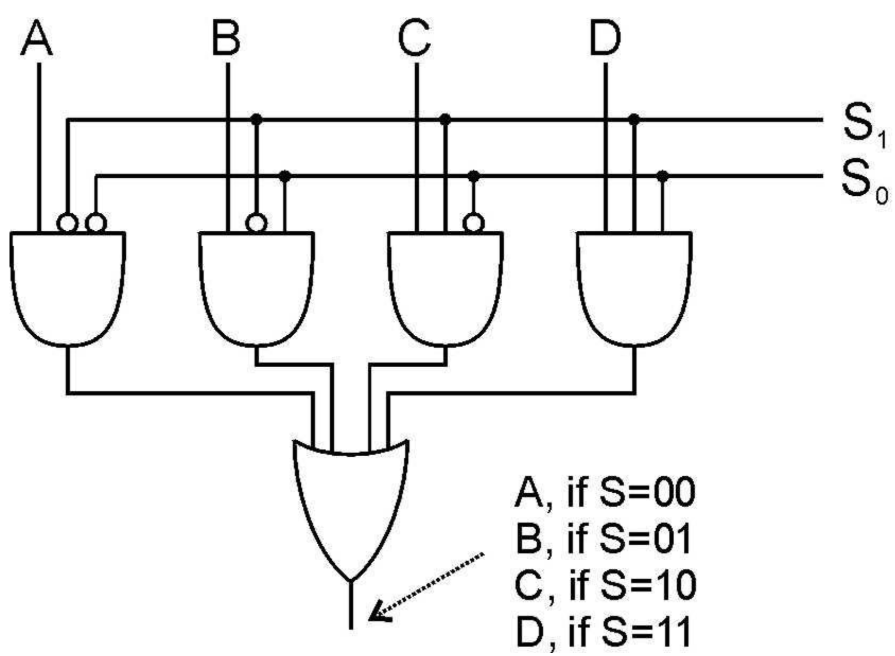
Encoder

- An encoder performs the opposite of a decoder.
- Only one input of its n inputs is 1 each time.
- The output ($\log_2 n$) is the binary representation of the input.

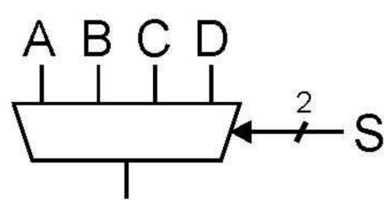


Multiplexer (Mux)

- A mux has 2^n data inputs, *n* select inputs, and one output
- The select bits are used to "choose" one of the data inputs to flow through to the output



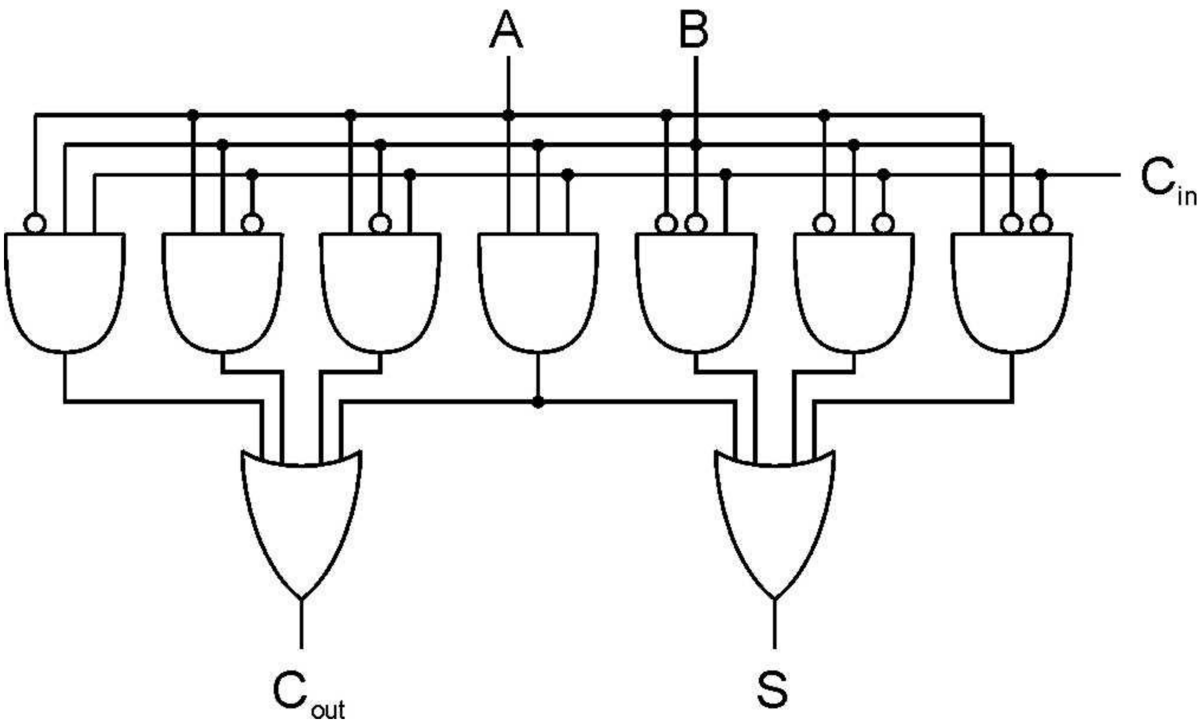
4-to-1 mux
4 input bits,
2 select bits



- *This is the standard symbol for a mux. S is a two-bit signal*
- *S_1 is the most significant bit, and S_0 is the least significant bit*

Full Adder (1-bit binary addition)

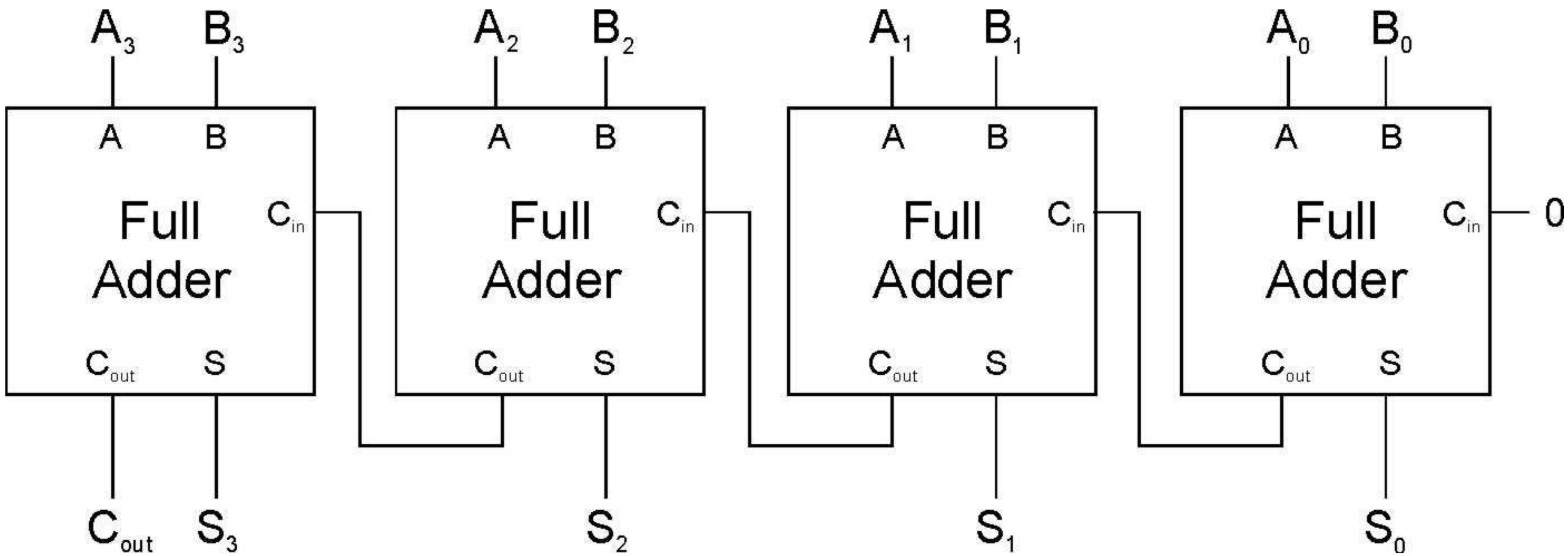
- A full adder represents one column in a binary addition operation. A and B are the bits to be added. C_{in} is the carry-in from the previous column (from the right). There are two outputs: S = sum, C_{out} = carry-out (to the next column).



C_{in}	A	B	S	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

N-bit Adder

- Feed C_{out} from one bit into C_{in} of the next bit...



***Q:** would all additions take the same time?*

university of groningen

Full Subtractor Using an 8 by 1 Multiplexer

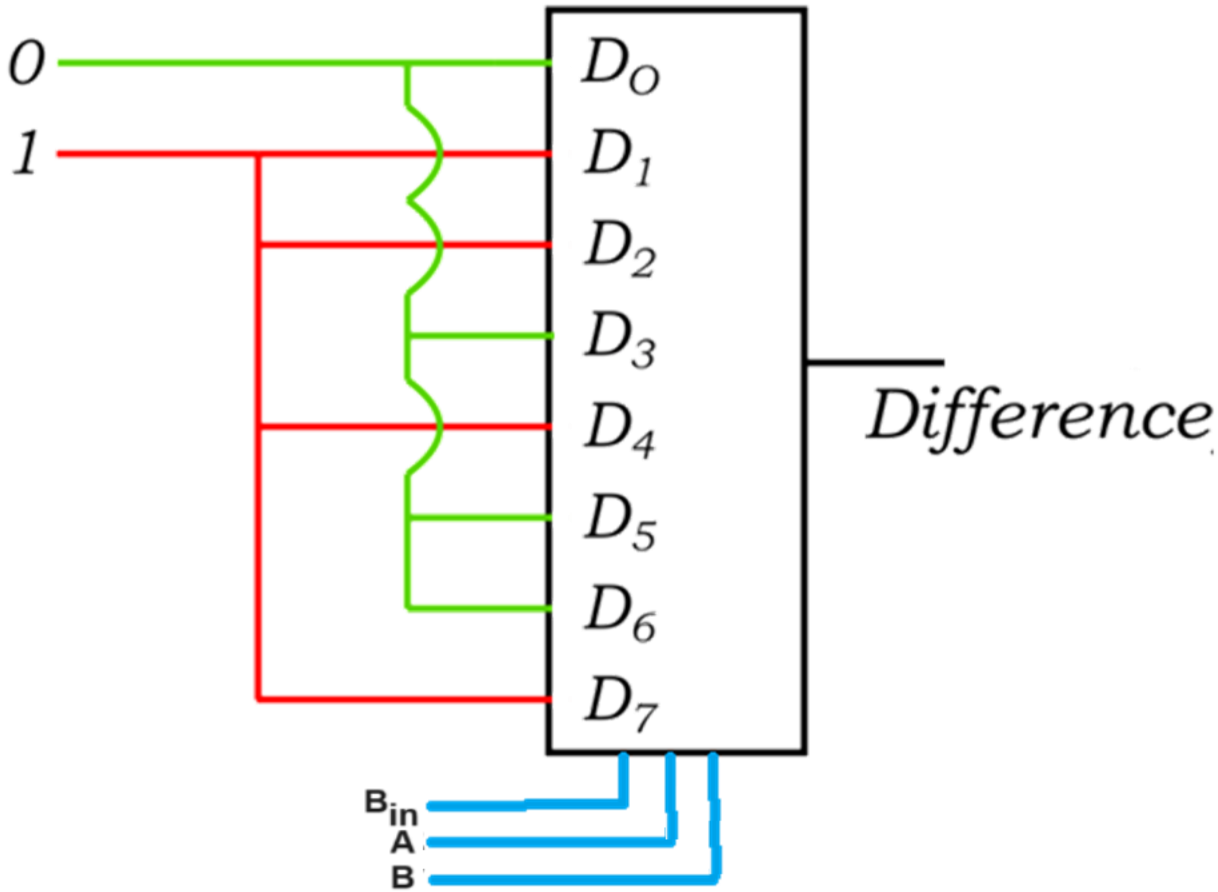
- Subtracting binary digits yields:
  - 0 − 0 = 0
  - 1 − 0 = 1
  - 0 − 1 = 1, borrow = 1
  - 1 − 1 = 0
- Therefore, we will have:

A	B	C_{in}	Difference	Borrow
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

Full Subtractor Using an 8 by 1 Multiplexer

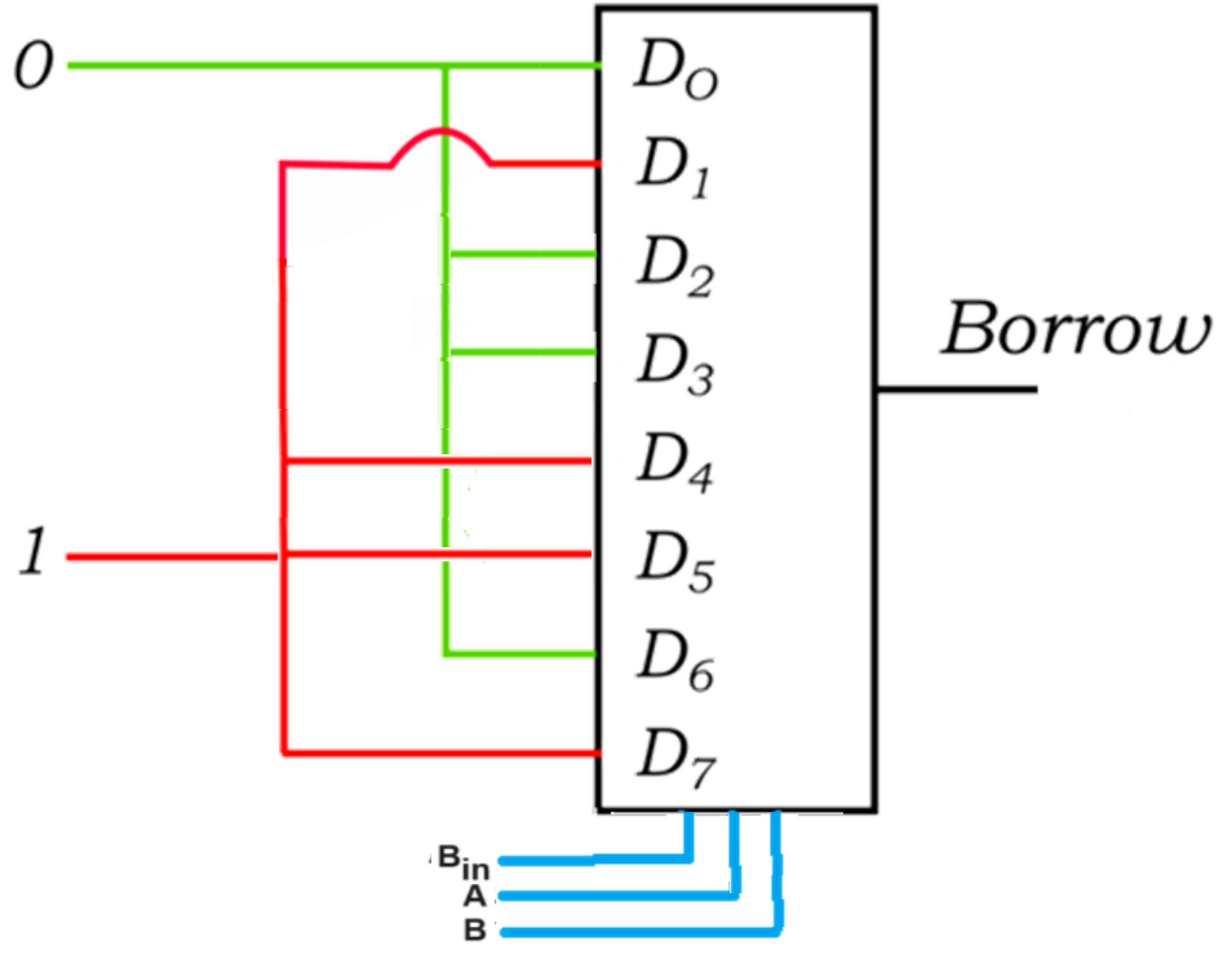
- The input to the multiplexer is either logical 1, or logical 0.
- Selecting one of these inputs in the output will depend on the bits to be subtracted, and carry in.
- One multiplexer will provide the difference, and a second multiplexer will give the borrow bit.

Full Subtractor Using an 8 by 1 Multiplexer



Bin	A	B	D	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

Full Subtractor Using an 8 by 1 Multiplexer



Bin	A	B	D	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

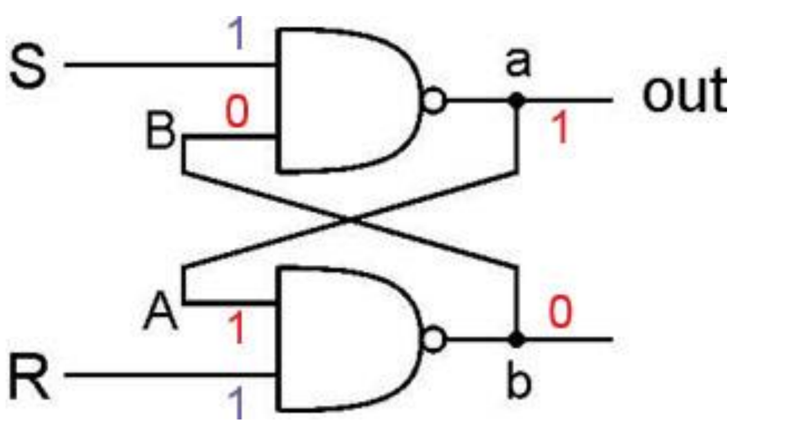
Sequential Logic Circuits

- Output depends on previous state(s)
- Must store previous information in order to act on it
- Example: ticket counter
  - When input is 1, output the "next" count: 0, 1, 2, 3, 4, 5, ...
  - Output depends on stored data (the current count) plus the input

- Requires storage element (memory)
- Will be used to implement "state machines" that perform a prescribed sequence of actions, depending on inputs

Storing One Bit: R-S Latch

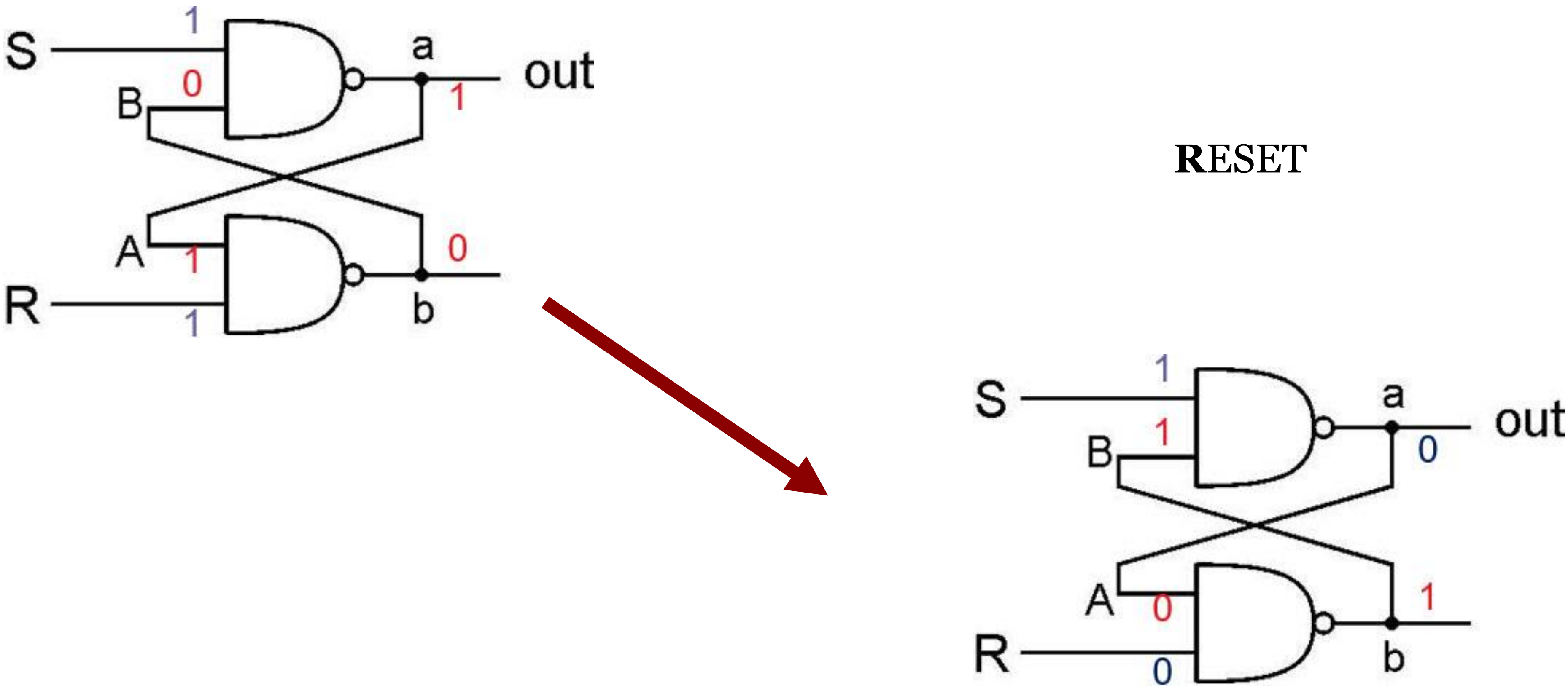
- A latch uses feedback to store a bit of information. In the circuit below, inputs R and S are both 1. The output may be either 0 or 1.
- **NOTE:** This is different than combinational logic -- different possible outputs for a particular input.



As long as R = S = 1, the output is stable. It either stays 0 or stays 1. This is known as the **quiescent** state. The circuit "remembers" this data.

Clearing the latch: R = 0

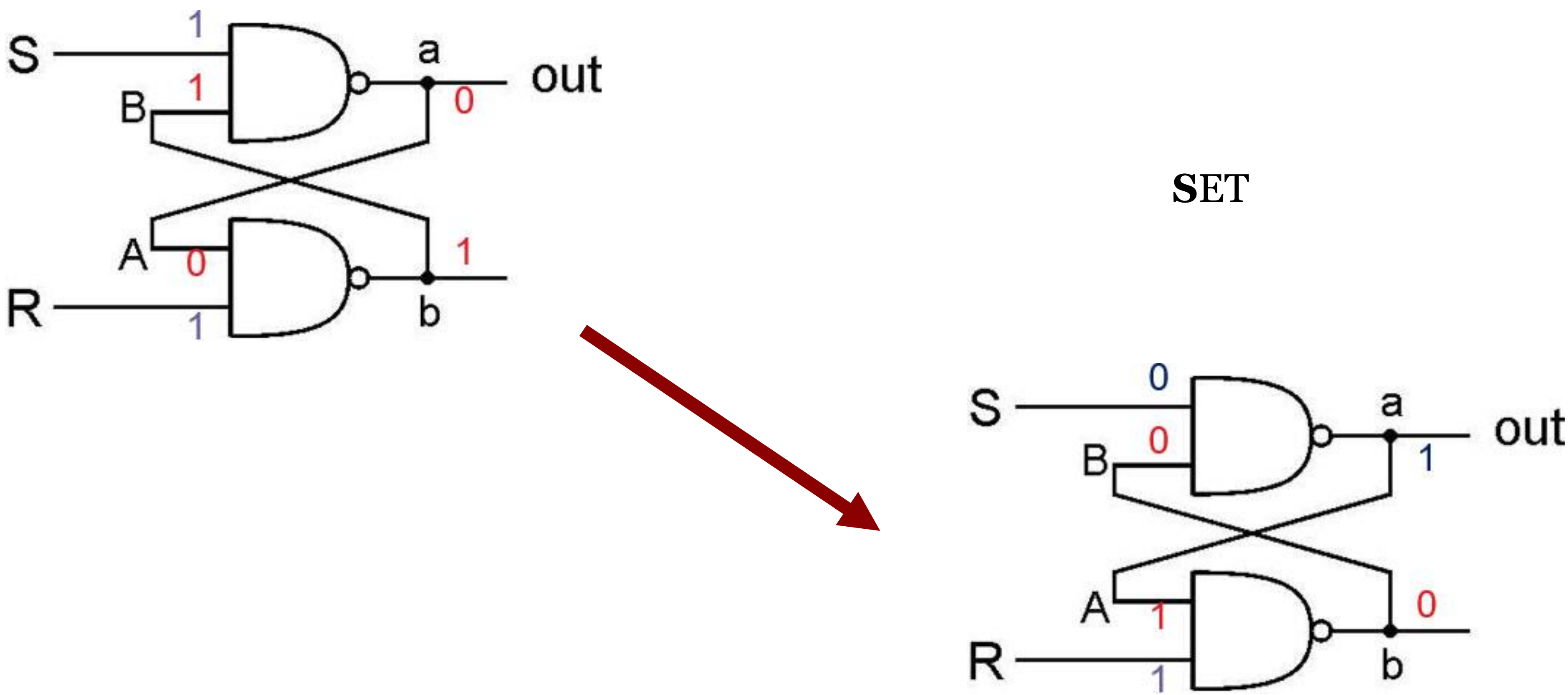
- To force the output (the stored data) to zero, set R = 0 and S = 1



- *Once the latch is cleared, make R = S = 1 to remember the data*

Setting the latch: S = 0

› To force the output (the stored data) to one, set R = 1 and S = 0



› *Once the latch is set, make R = S = 1 to remember the data*

R-S Latch Summary

R = S = 1

Hold current value in latch

R = 0, S = 1

Clear/reset latch (value = 0)

R = 1, S = 0

Set latch (value = 1)

R	S	Q'
0	0	*invalid*
0	1	0
1	0	1
1	1	Q

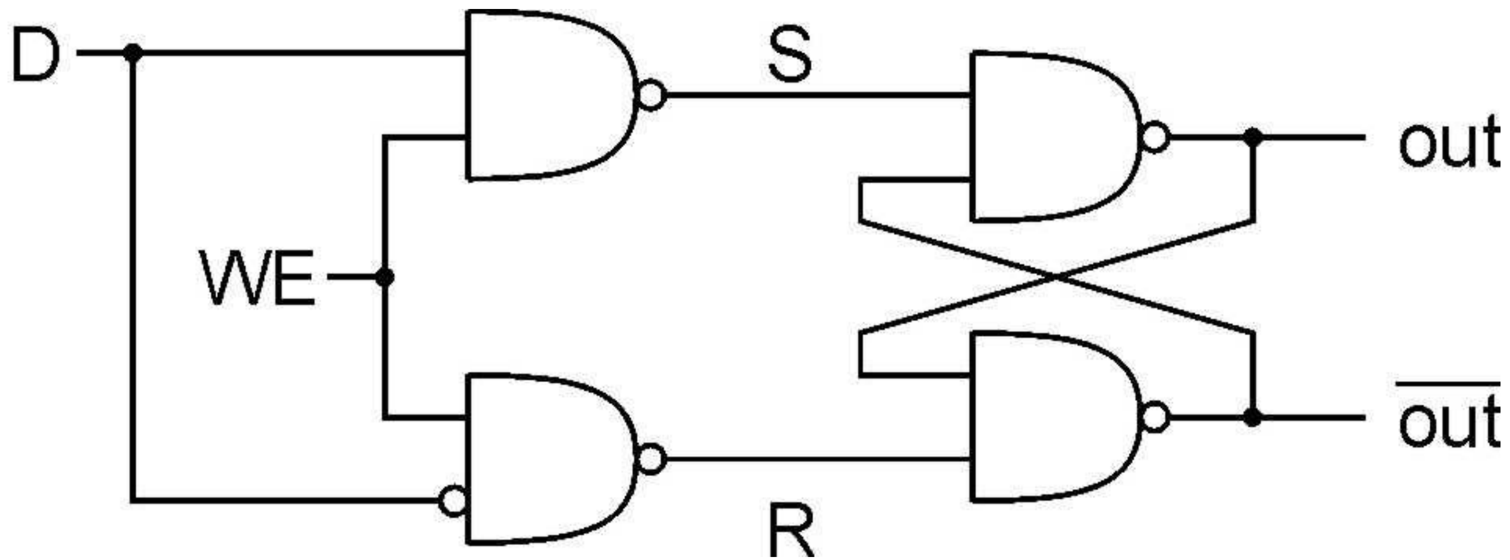
Q' is output. Q is previous output

What about R = 0, S = 0?

Both outputs = 1, final state determined by electrical properties of the gates. Don't do it!!! This set of inputs is prohibited and should never occur during the operation of the latch.

D-Latch: Simpler Control for One-Bit Latch

- D = data input, WE = write enable
- When WE = 0, latch output does not change (D is irrelevant)
- When WE = 1, latch output = D



- **NOTE:** This uses an R-S latch. R and S will never both be zero

Summary

- Digital logic is the implementation of Boolean functions with transistors.
- Combination logic is a circuit where the output depends on the input(s) only.
  - Multiplexer
  - Decoder
  - Full adder/subtractor
- Sequential logic is a circuit where the output depends on the input and the current state.
- Both combinational and sequential logic circuits are required to design a digital computer.

Questions?