NWEN 242

4. Combinational and sequential logic

Agenda

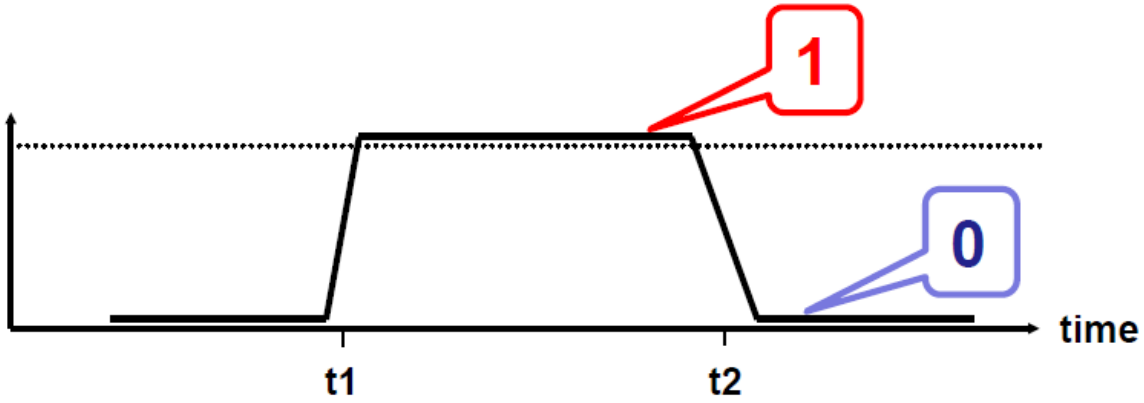
- Combinational logic
  - AND gate, OR gate, NOT (inverter) gate
  - Multiplexors
  - Decoders
- Sequential logic
  - S-R latch
  - D latch (S-R latch with a clock)
  - D flip-flop
- Register file

Types of logic circuit

- In digital circuit theory, combinational logic is a type of digital logic where the output is a pure function of the present input only.
- Sequential logic, in which the output depends not only on the present input but also on the history of the input.
  - In other words, sequential logic has memory while combinational logic does not.

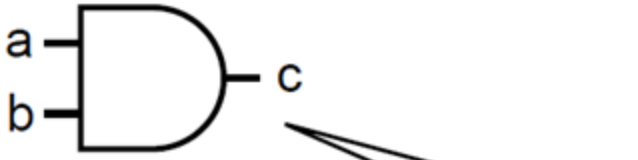
Definition of TRUE of ASSERTED

- Modern digital computers use two-level logic
- Not all digital computers use a high voltage for binary 1 and a low voltage for binary 0
  - People talk about "true" or "asserted" and "false" or "deasserted"
  - To make things simple, we map asserted to **1** and deasserted to 0



AND and OR gates

$c = a \cdot b$ Boolean expression

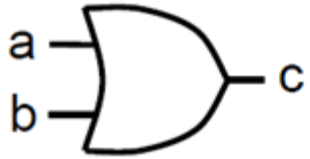


Logic gate

a	b	c
0	0	0
0	1	0
1	0	0
1	1	1

Truth table

$c = a + b$ Boolean expression



Logic gate

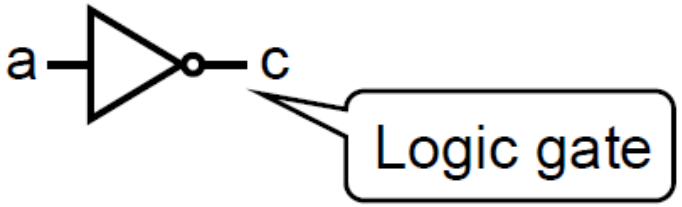
a	b	c
0	0	0
0	1	1
1	0	1
1	1	1

Truth table

Inverter gate

- The inverter gate performs the logic operation NOT

$c = \overline{a}$ Boolean expression

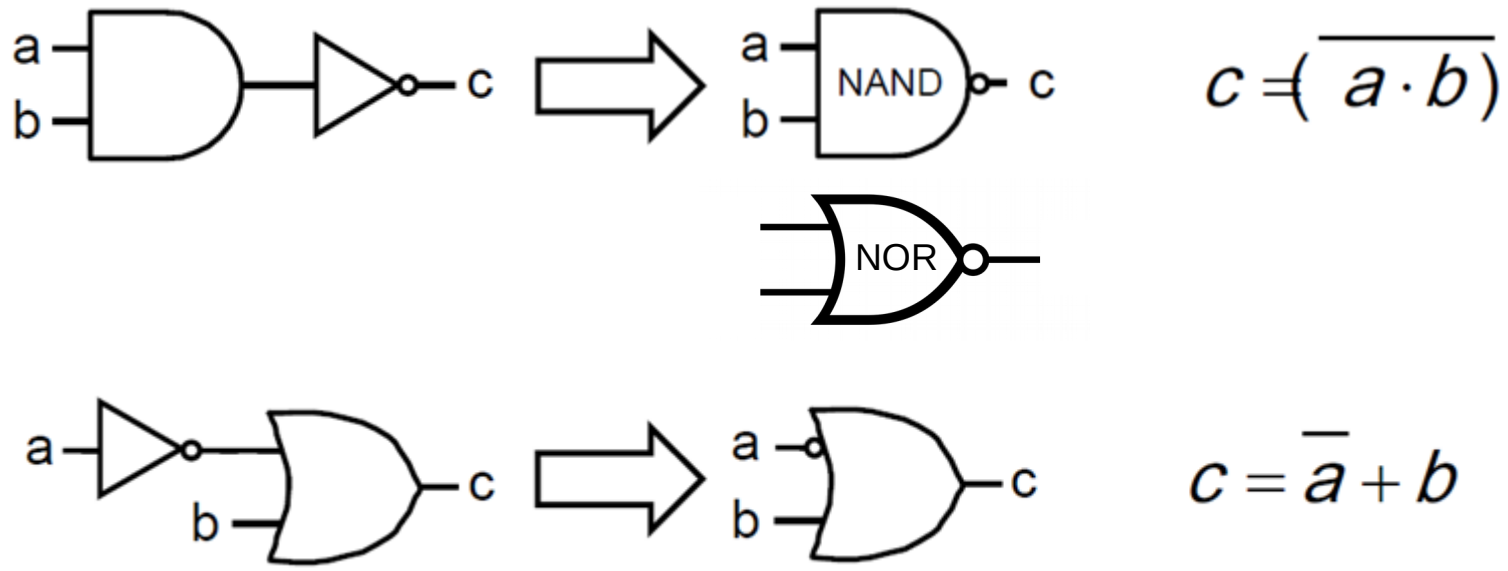


a	c
0	1
1	0

Truth table

"Bubbles"

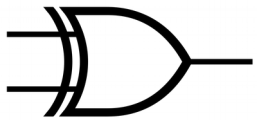
- The NOT gate is sometimes denoted by a circle on an output or input



Quick exercise

- A toggle operation cannot be performed by using a single
  - A. NOT gate
  - B. AND gate
  - C. XOR gate
  - D. None of the above

A	B	out
0	0	0
0	1	1
1	0	1
1	1	0



How to design a logic block using gates?

- Goal: identify a logic function
- Design steps:
  - Represent the logic function using a truth table
    - Each row associated with an asserted output
  - Use an AND gate to represent each conjunction term
    - Connect all inputs with an AND gate
  - Connect all conjunction terms through disjunction
    - Use an OR gate to represent each disjunction

Example

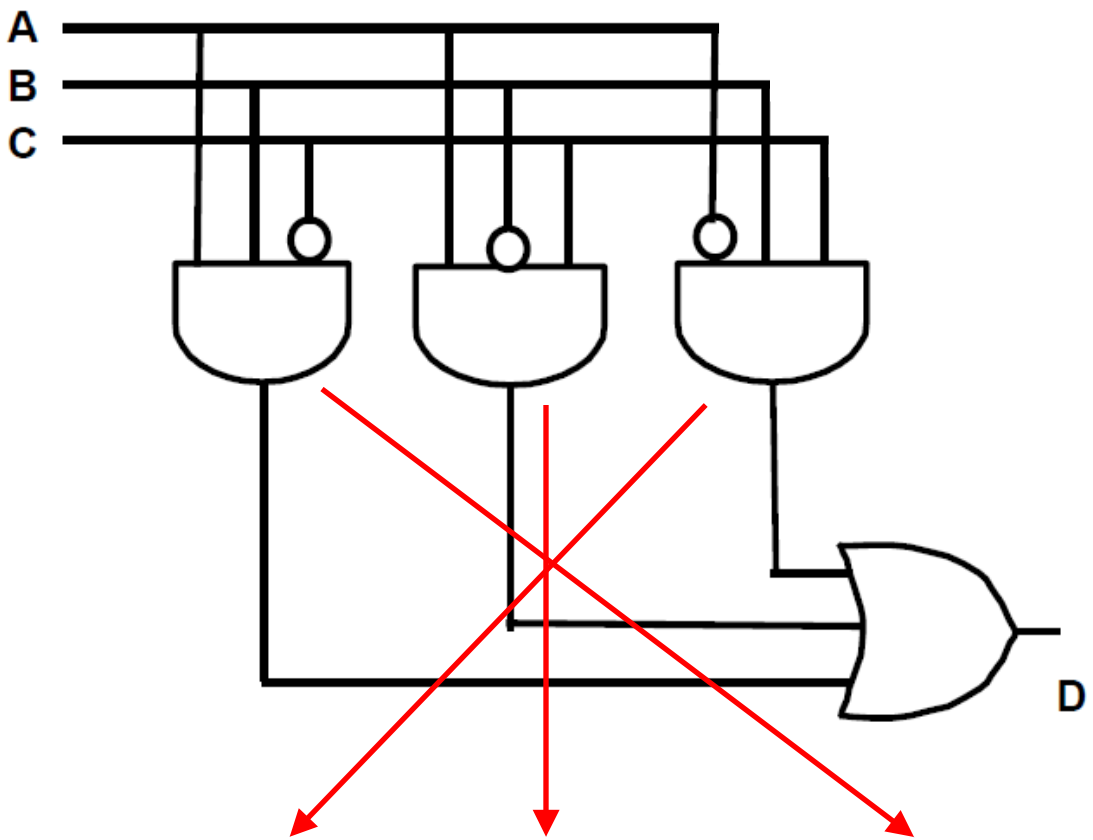
- Design a logic block using gates
- Consider a three input and one output logic function:

The output D is true when two and only two of the inputs A, B, and C are true

Truth Table			
A	B	C	D
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	1
1	1	1	0

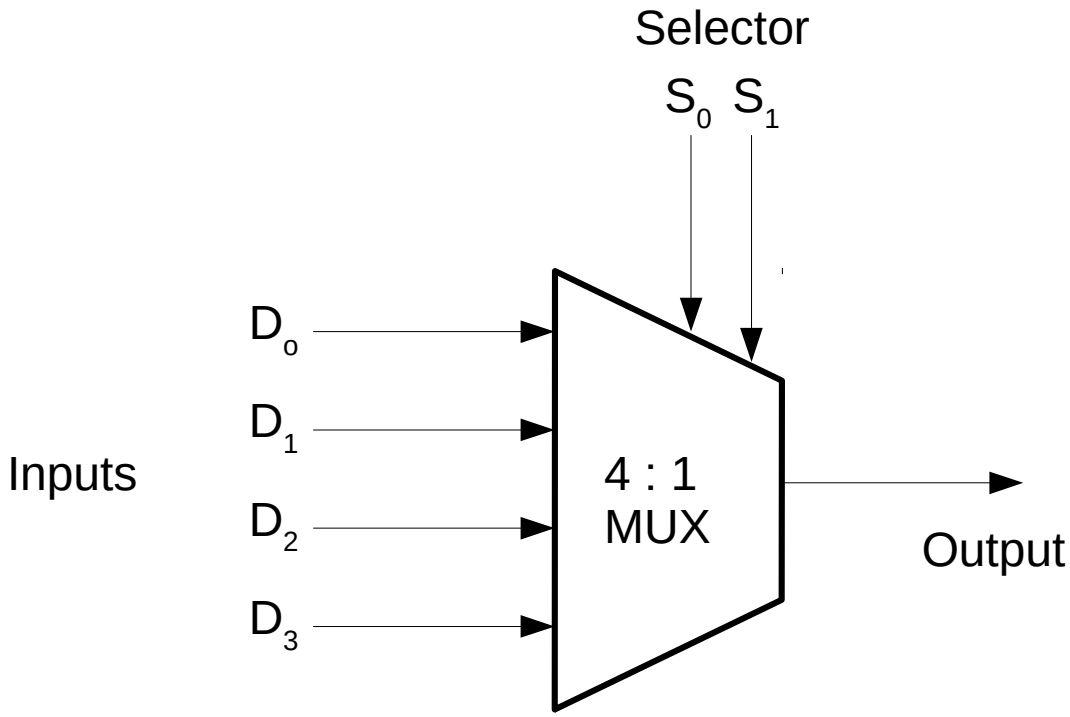
$$D = \overline{A} \bullet B \bullet C + A \bullet \overline{B} \bullet C + A \bullet B \bullet \overline{C}$$

Answer: The logic block using gates



$$D = \overline{A} \bullet B \bullet C + A \bullet \overline{B} \bullet C + A \bullet B \bullet \overline{C}$$

Multiplexors



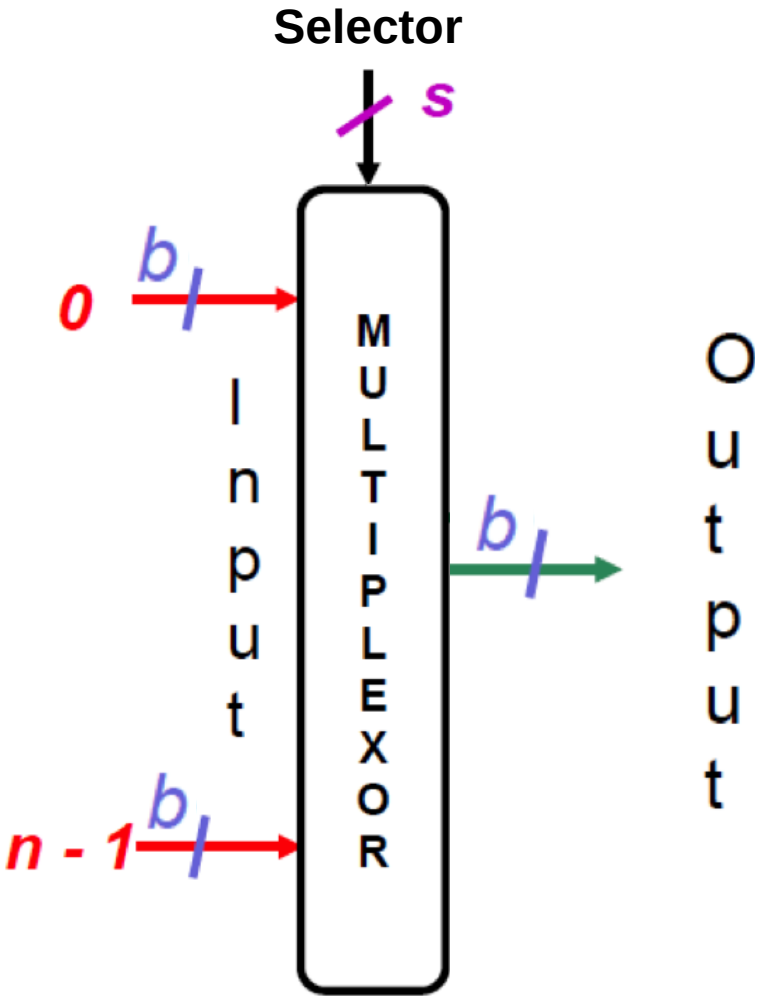
Inputs might be 1 or 2 bits or words of 32 bits
The output will be ONE of the inputs

Multiplexors

- A multiplexor is a combinational logic block containing **n** data inputs and 1 output
- The *s* selector inputs determine which of the n inputs will be the output.

$$s = \lceil \log_2 n \rceil$$

- If each input is a bunch of b bits, then the output is also a bunch of b bits.



Design a multiplexor

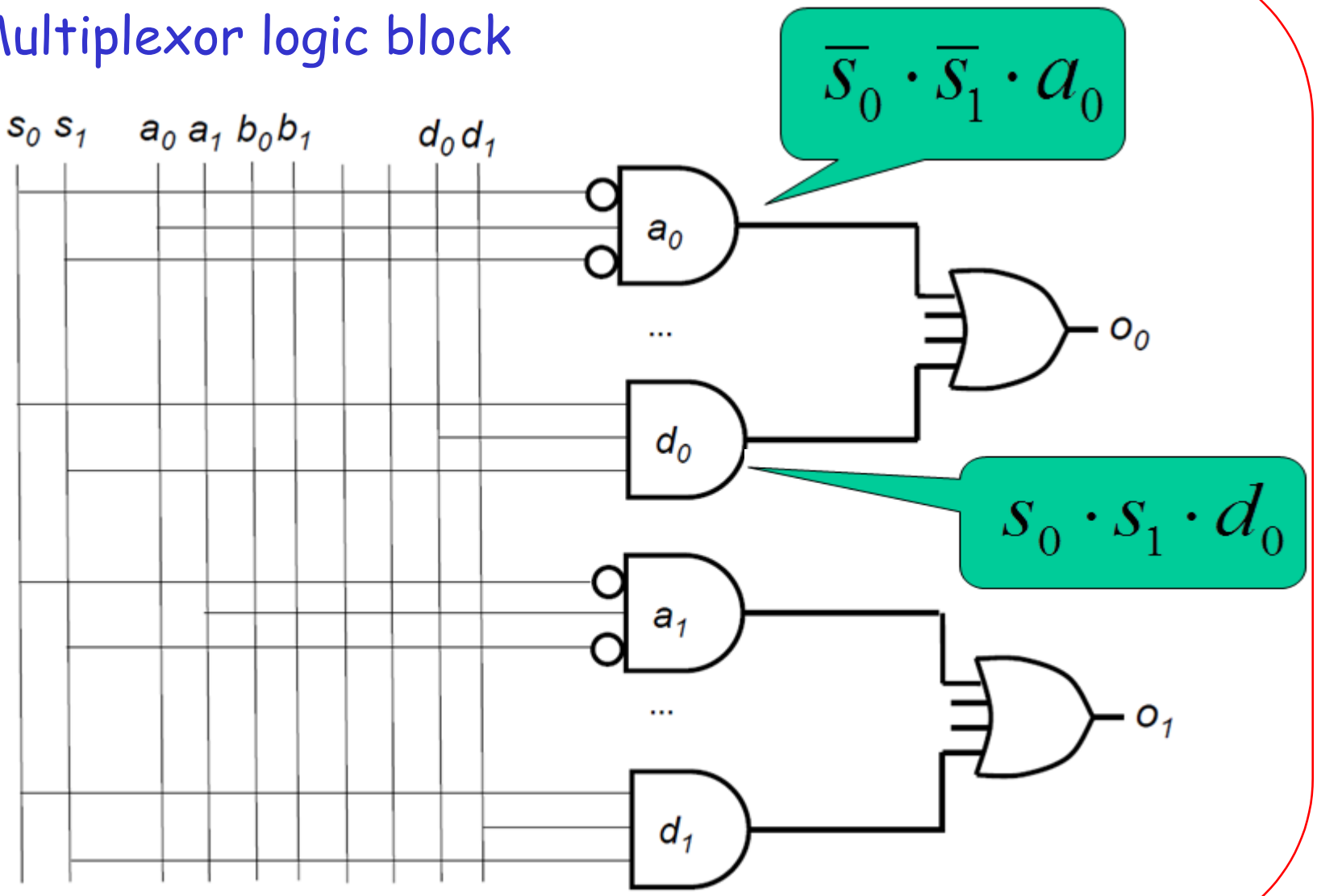
- Let n = 4 inputs
  - then $s = \text{Log}_2 n = \text{Log}_2 4 = 2$ bits (S_0, S_1)
- Let b = 2 bits, input pairs (a_0, a_1), (b_0, b_1), (c_0, c_1), and (d_0, d_1)
- Then the multiplexor's truth table is:

Selector Inputs		Outputs	
s_0	s_1	o_0	o_1
0	0	a_0	a_1
0	1	b_0	b_1
1	0	c_0	c_1
1	1	d_0	d_1

$S_0 = 0, S_1 = 0 \Rightarrow O_0 = A_0, O_1 = A_1,$

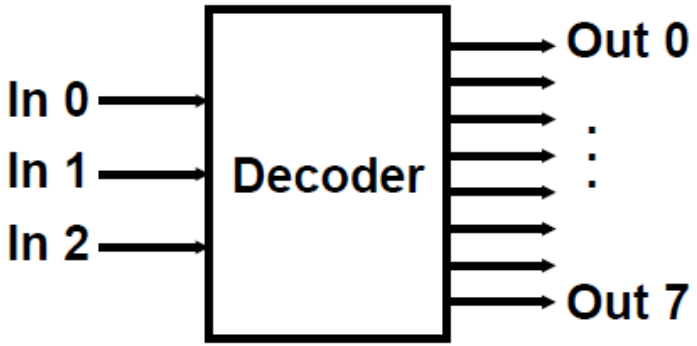
$$O_0 = \overline{S}_0 \bullet \overline{S}_1 \bullet A_0$$
$$O_1 = \overline{S}_0 \bullet \overline{S}_1 \bullet A_1$$

Multiplexor logic block



Decoders

- A 3-to-8 decoder example
- If we interpret the 3 inputs as a 3-bit binary number n
  - Then the active output is output n
- Generally, we talk about n-to-2^n decoders



$In_0 = 0$
$In_1 = 1$
$In_2 = 0$

$Out_0 = 0$
$Out_1 = 0$
$Out_2 = 1$
$Out_3 = 0$
...

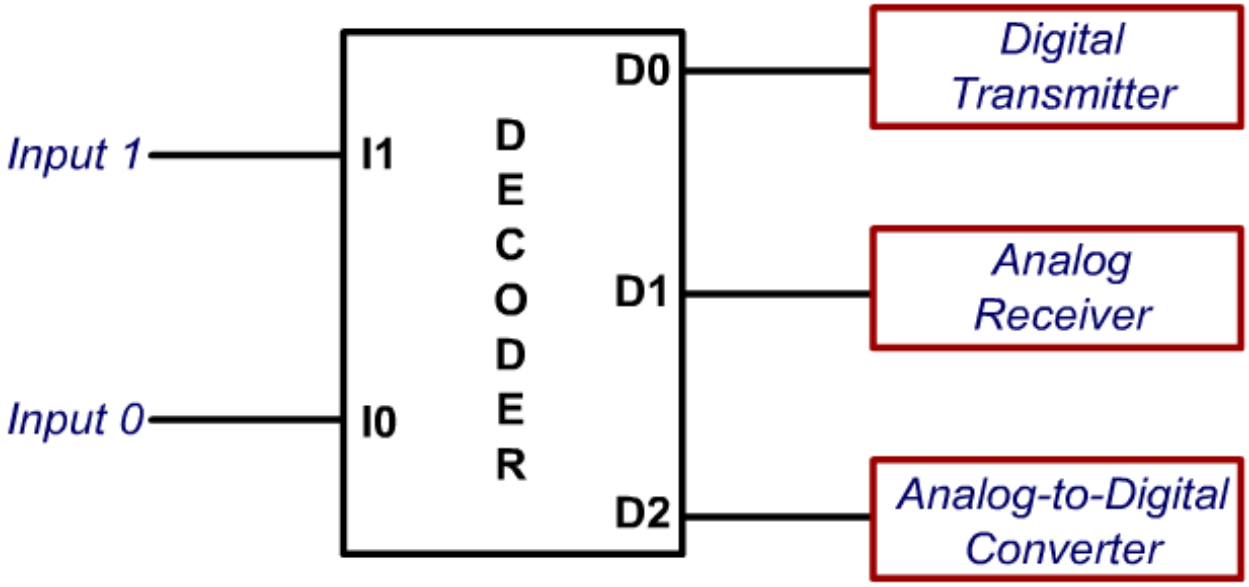
Decoder truth table

$\overline{In_0} \cdot \overline{In_1} \cdot \overline{In_2}$

Inputs			Outputs							
In2	In1	In0	Out7	Out6	Out5	Out4	Out3	Out2	Out1	Out0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

Quick exercise

- A decoder is set up as shown, what should the input sequence be to for the Digital Transmitter?



- A: $In_1 = 0, In_0 = 1$
- B: $In_1 = 1, In_0 = 0$
- C: $In_1 = 0, In_0 = 0$
- D: $In_1 = 1, In_0 = 1$

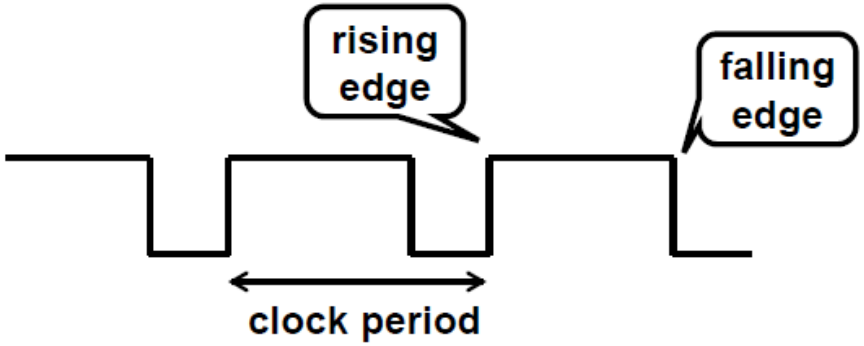
The register

- In MIPS, you have 32 registers
- You can store data in registers and later read from them
- The set of registers is called the register file
- It is built of **multiplexors**, **decoders**, and **flip-flops**
- We have discussed multiplexors and decoders
- Next flip-flops.

Sequential logic

A clock signal

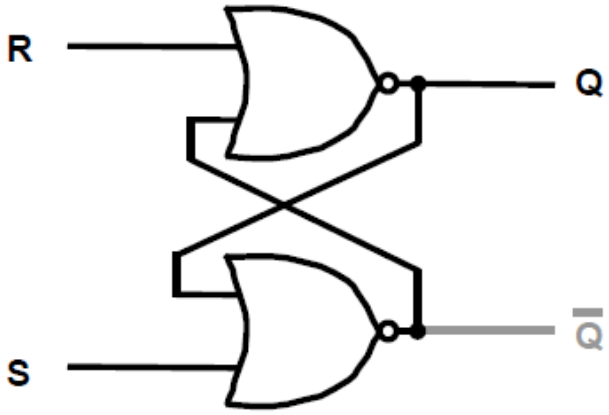
- **Sequential logic** blocks usually update their output on the clock edge



- This is edge-triggered clocking
- Change in inputs at any other time have no effect on the outputs

Set-reset (SR) latch

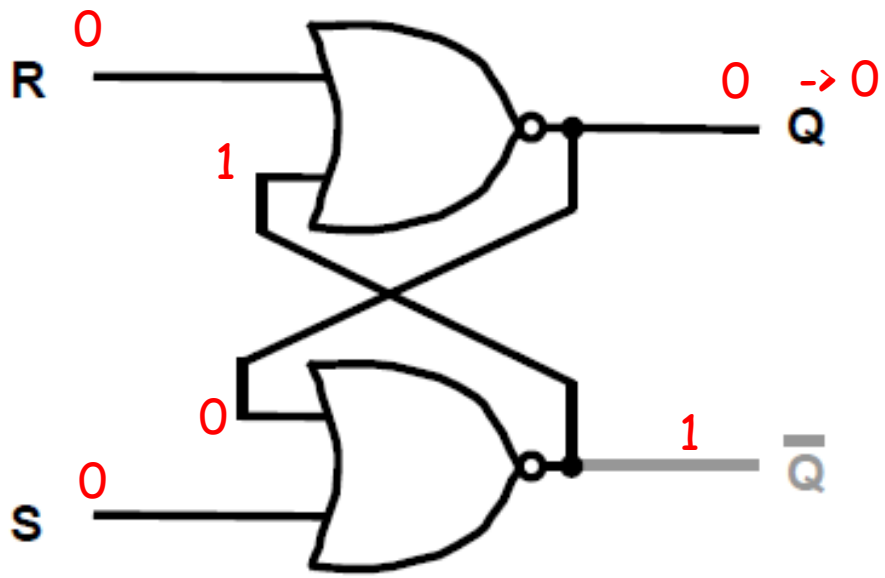
- Inputs
  - Reset
  - Set
- Not clocked
- Output undefined when S and R asserted simultaneously



NOR		
0	0	1
0	1	0
1	0	0
1	1	0

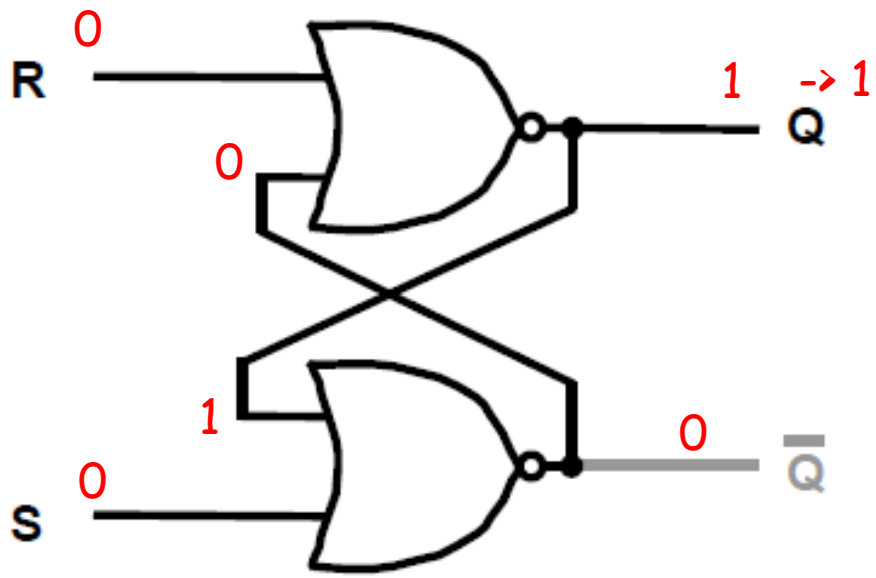
Q	S	R	Q_n
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	?
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	?

State transition table



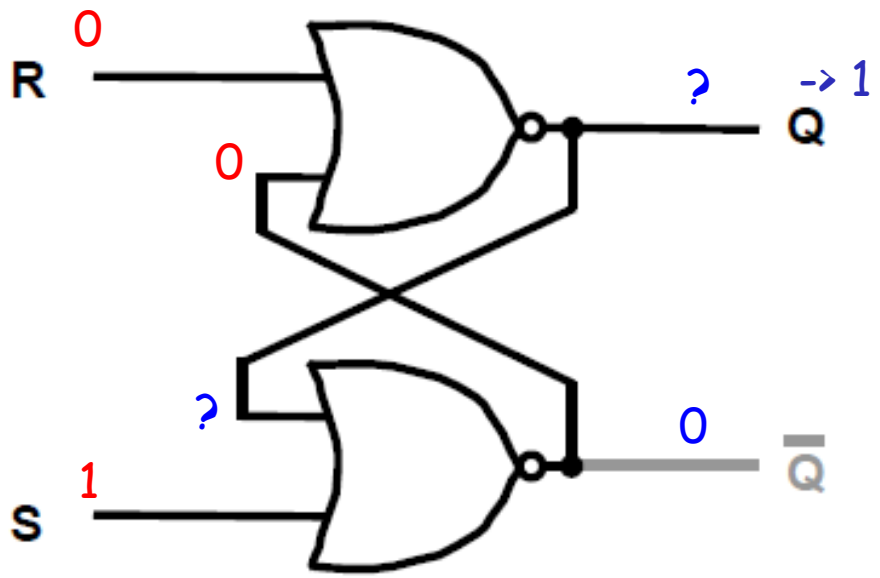
Q	S	R	Q_n
0	0	0	0
0	0	1	
0	1	0	
0	1	1	?
1	0	0	
1	0	1	
1	1	0	
1	1	1	?

State transition table



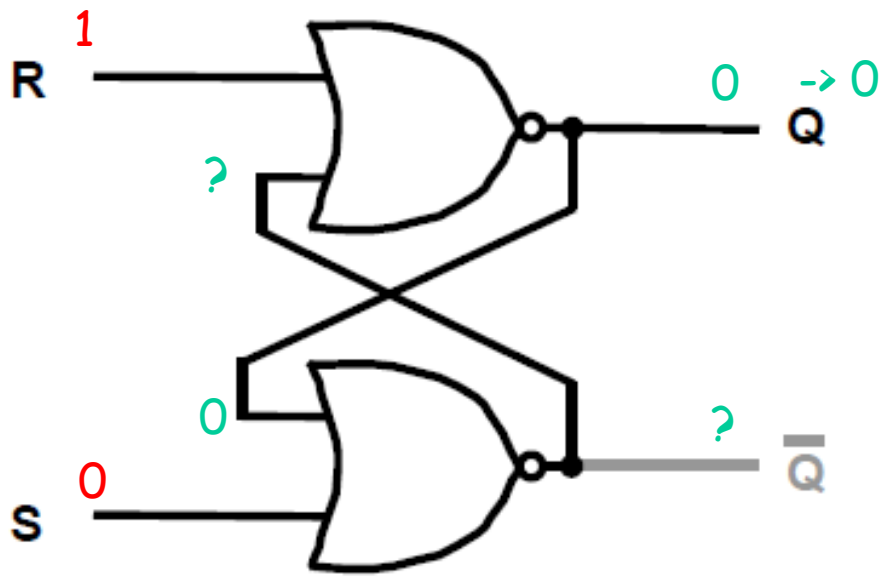
Q	S	R	Q_n
0	0	0	0
0	0	1	
0	1	0	
0	1	1	?
1	0	0	1
1	0	1	
1	1	0	
1	1	1	?

State transition table



Q	S	R	Q_n	
0	0	0	0	
0	0	1		
0	1	0	1	←
0	1	1	?	
1	0	0	1	
1	0	1		
1	1	0	1	←
1	1	1	?	

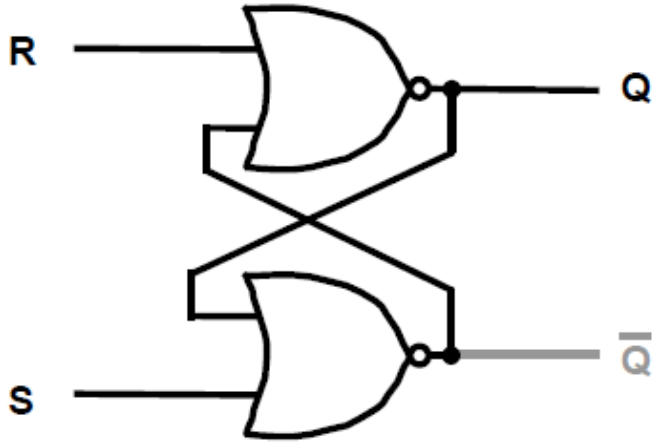
State transition table

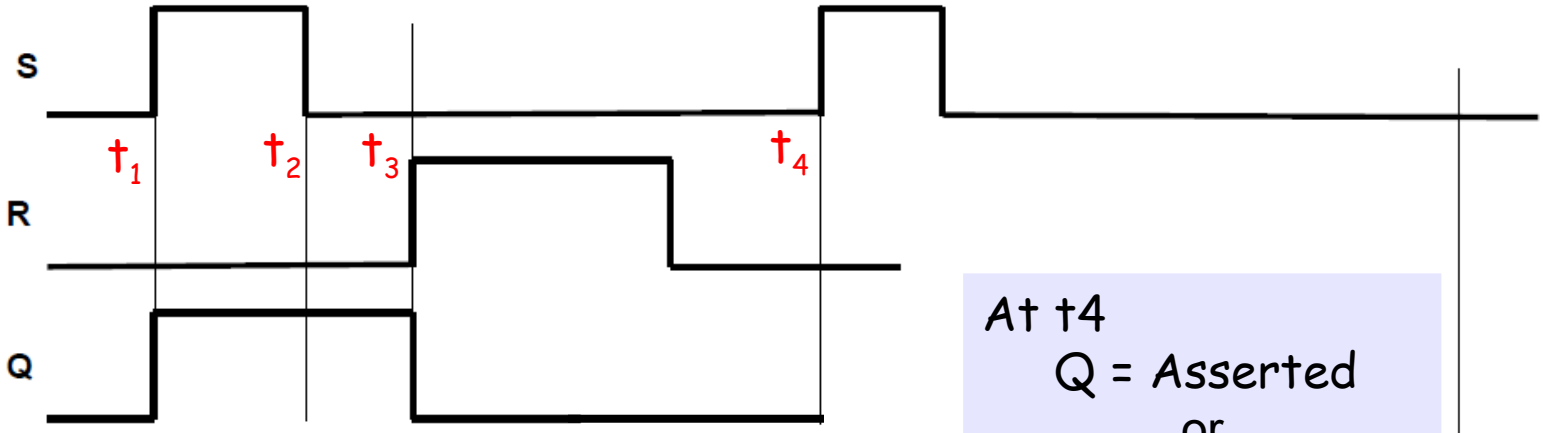


Q	S	R	Q_n
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	?
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	?

State transition table

S-R latch example





At t4
Q = Asserted
or
Q = Deasserted