COMPUTER SYSTEMS ARCHITECTURE LECTURE 0x03

DIGITAL ABSTRACTION

COMBINATIONAL AND SEQUENTIAL CIRCUITS

Cristian Rusu, Bogdan Macovei

LAST TIME

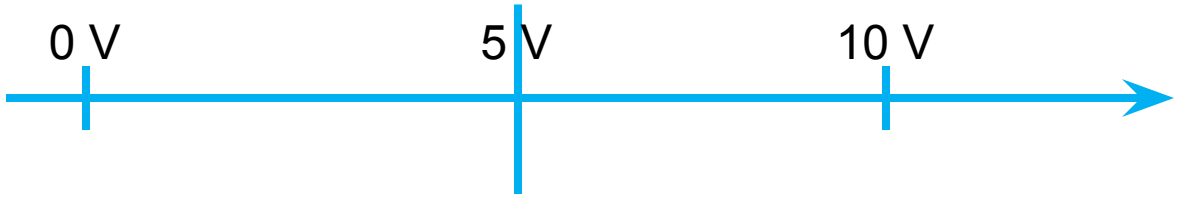
- we discussed the concept of information
- we saw how to measure information
- we computed Shannon entropy
- we saw the Huffman algorithm for variable-length data encoding
- we used the Hamming distance between two bit strings
- today, we will see how to implement these bits in circuits

OUTLINE

- digital abstraction
- digital circuits
- the transistor
- combinational circuits
- logical simplifications
- addition circuit
- examples of combinational and sequential circuits
- bibliographic references

DIGITAL ABSTRACTION

- **from continuous to digital**

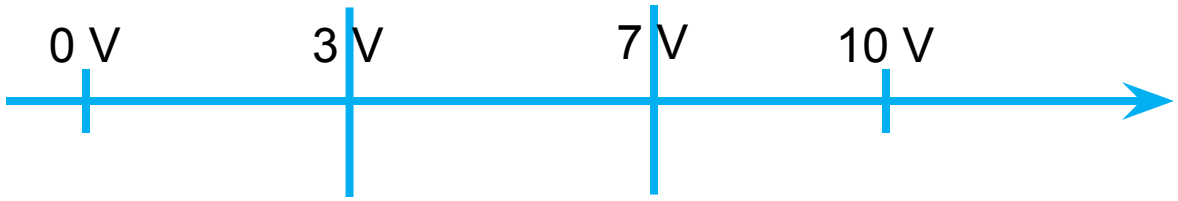


  - the simplest idea: we have a maximum voltage that can be reached; therefore, from 0V to 5V we encode “0”, and from 5V to 10V we encode “1”

  - what difficulties do we have in this situation?
    - it is difficult to understand what happens around 5V

DIGITAL ABSTRACTION

- **from continuous to digital**

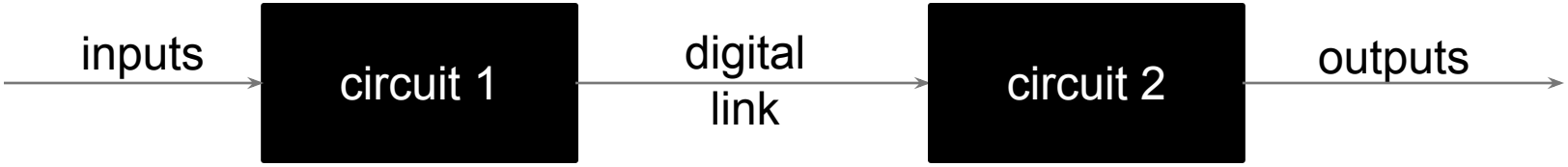


  - a slightly more sophisticated solution: we have two thresholds
  - this time: “0” is between 0V and 3V, while “1” is between 7V and 10V
  - the interval between 3V and 7V is a “no man’s land”
    - we cannot decide the voltage
    - we wait for stabilization at a value < 3V or > 7V

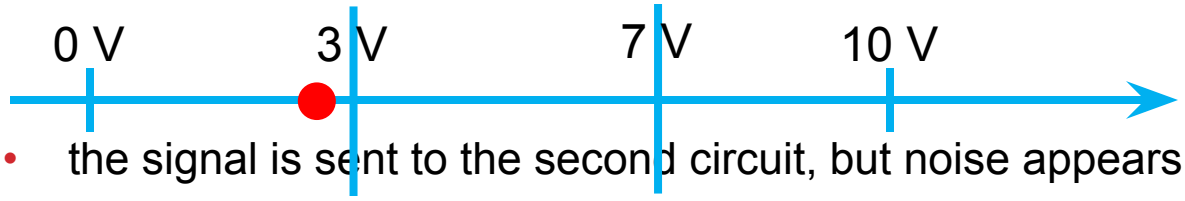
Warning: the system is not perfect; at 3V we are at “0”, but noise of only 4V from this point can take us to 7V, hence to “1”

DIGITAL ABSTRACTION

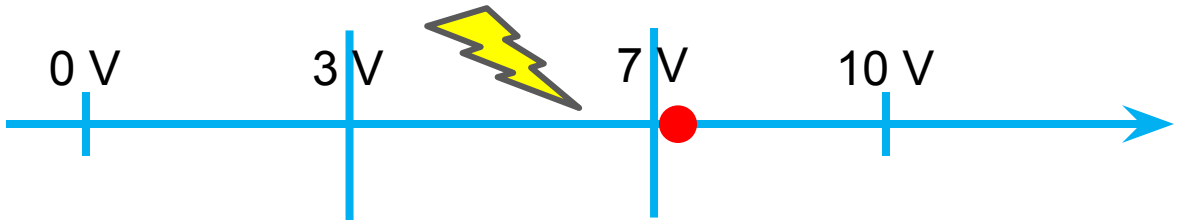
- **from continuous to digital**
  - in most cases, we will connect digital devices to each other



  - for example:
    - the first circuit outputs a “0”, but at the upper limit



    - the signal is sent to the second circuit, but noise appears on the wire



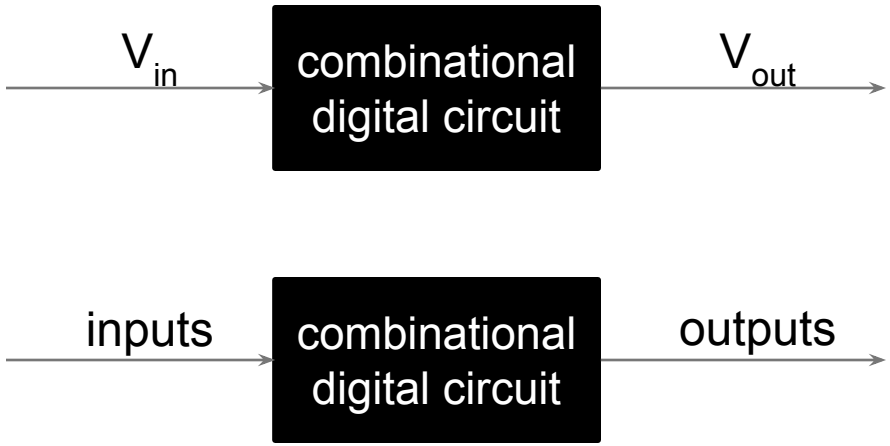
Warning: the system is not perfect; at 3V we are at “0”, but noise of only 4V from this point can take us to 7V, hence to “1”

The solution? for outputs we will impose stricter thresholds (below 2V, above 8V)

ideally, the thresholds should be as narrow as possible: 0V is “0” and 10V is “1”

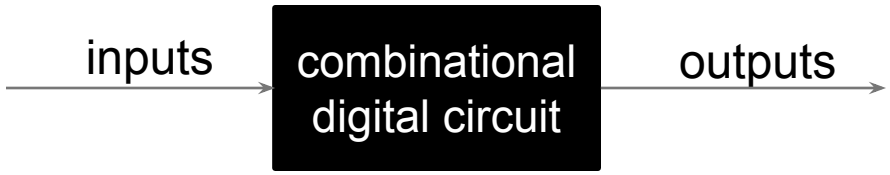
DIGITAL CIRCUITS

- **combinational digital circuit**



DIGITAL CIRCUITS

- **combinational digital circuit**

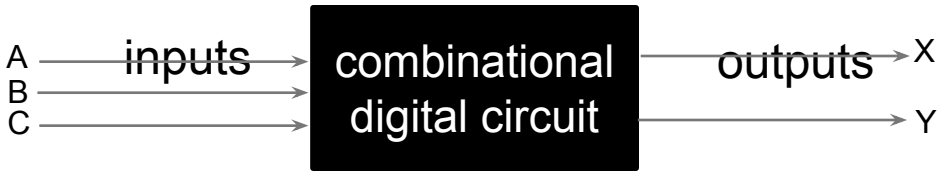


  - digital inputs, there may be many
  - digital outputs, there may be many
  - it has no internal states
    - you put a constant digital signal at the input and get another constant digital signal at the output
    - but it cannot “remember” anything
    - it has no “internal state” (memory)
  - we have a propagation time (tp): the maximum time required to produce correct and valid digital signals at the output from the moment correct and valid digital signals are specified at the input

  - why are they called combinational circuits?
    - because the output is a combination (a logical function that combines) all (or part) of the inputs

DIGITAL CIRCUITS

- **combinational digital circuit**



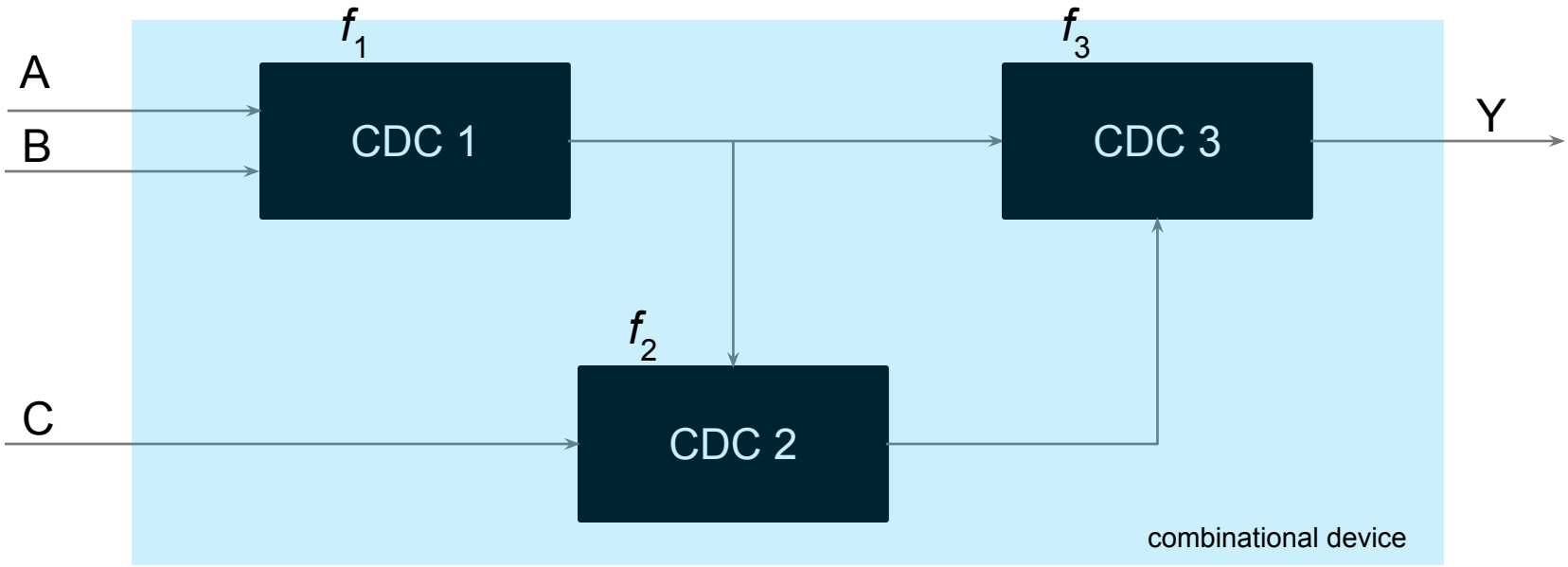
- why are they called combinational circuits?
  - because the output is a combination (a logical function that combines) all (or part) of the inputs
  - therefore, for each input, we must know what the output is

A	B	C	X	Y
0	0	0	0	1
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	0

truth table

DIGITAL CIRCUITS

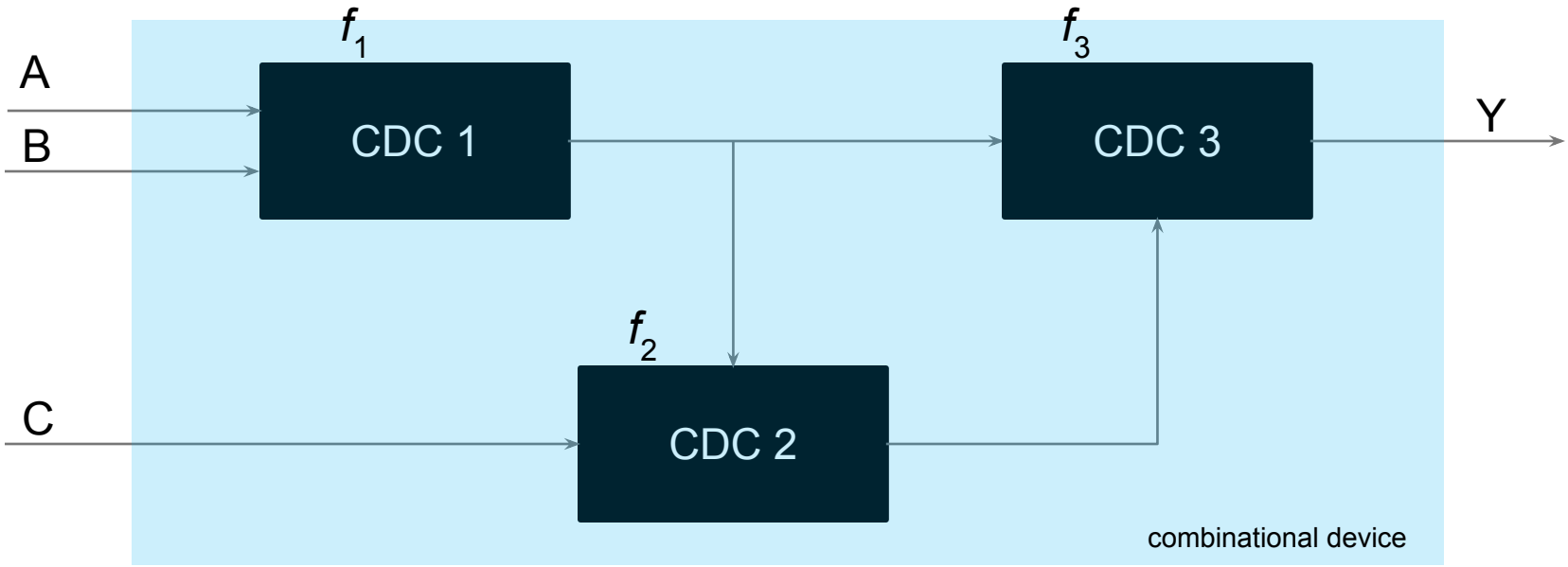
- **combinational device**
  - each element is a combinational circuit
  - each input is connected to exactly one output or to a constant
  - there is no cycle in the directed graph of the device



- what is the function of the device? Y = f3(f1(A, B), f2(f1(A, B), C))

DIGITAL CIRCUITS

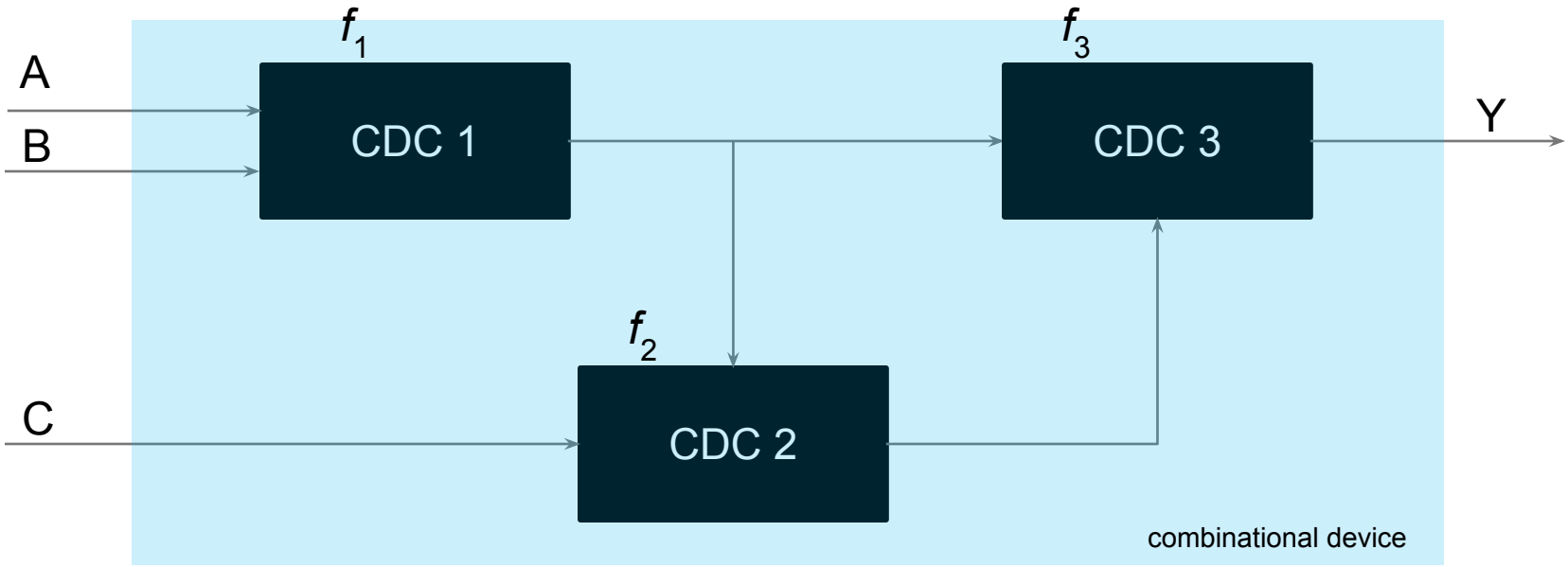
- **combinational device**
  - each element is a combinational circuit
  - each input is connected to exactly one output or to a constant
  - there is no cycle in the directed graph of the device



  - total propagation time? tp,total = tp,1 + tp,2 + tp,3 (longest path)

DIGITAL CIRCUITS

- **combinational device**
  - each element is a combinational circuit
  - each input is connected to exactly one output or to a constant
  - there is no cycle in the directed graph of the device

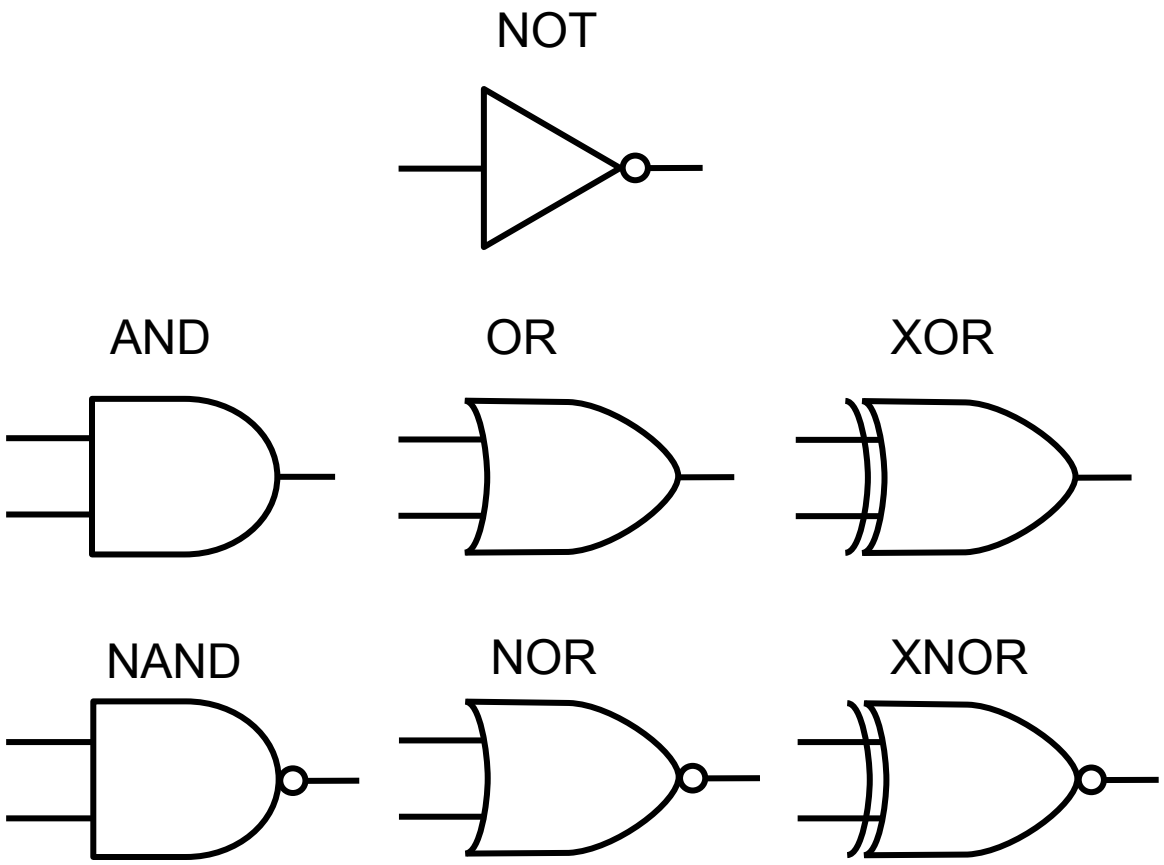


- total propagation time? tp,total = tp,1 + tp,2 + tp,3 (longest path)

the maximum time to obtain a valid output from valid inputs

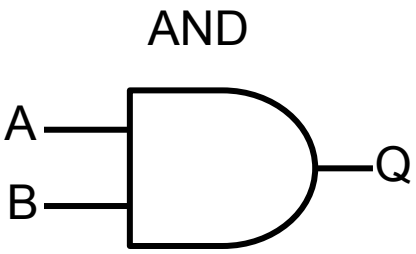
DIGITAL CIRCUITS

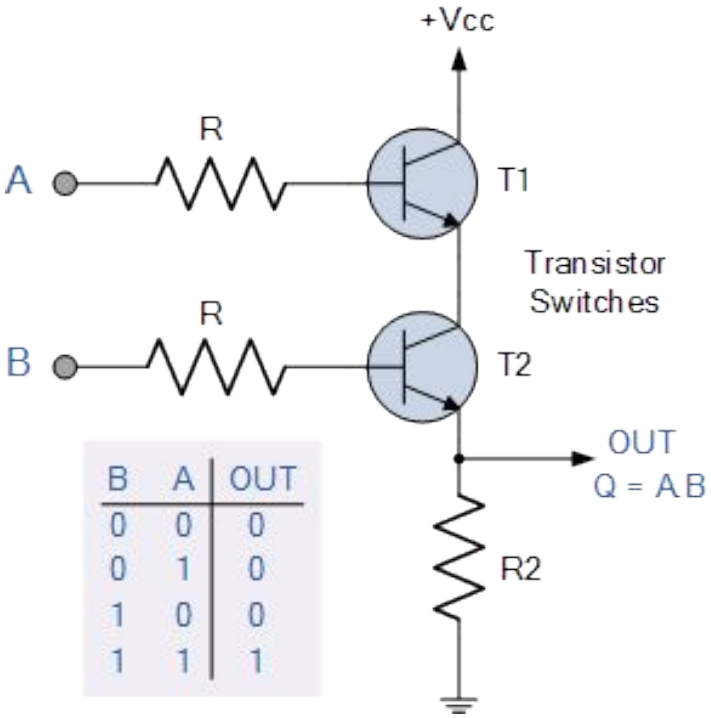
- **combinational digital circuit**
  - fundamental examples



DIGITAL CIRCUITS

- **combinational digital circuit**
  - fundamental examples
  - at the base of all of them are transistor-based electronic circuits

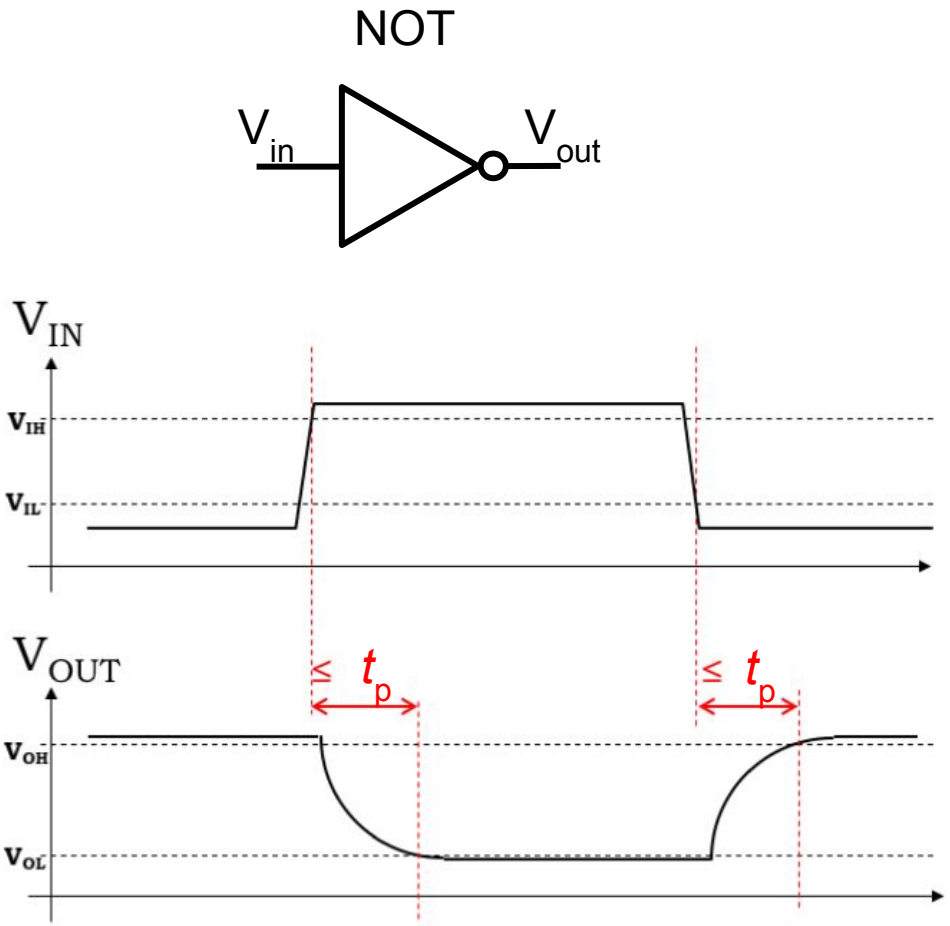




https://www.electronics-tutorials.ws/logic/logic_2.html

DIGITAL CIRCUITS

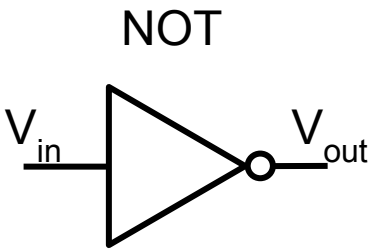
- **combinational digital circuit**
  - fundamental examples: let us not forget that everything is analog

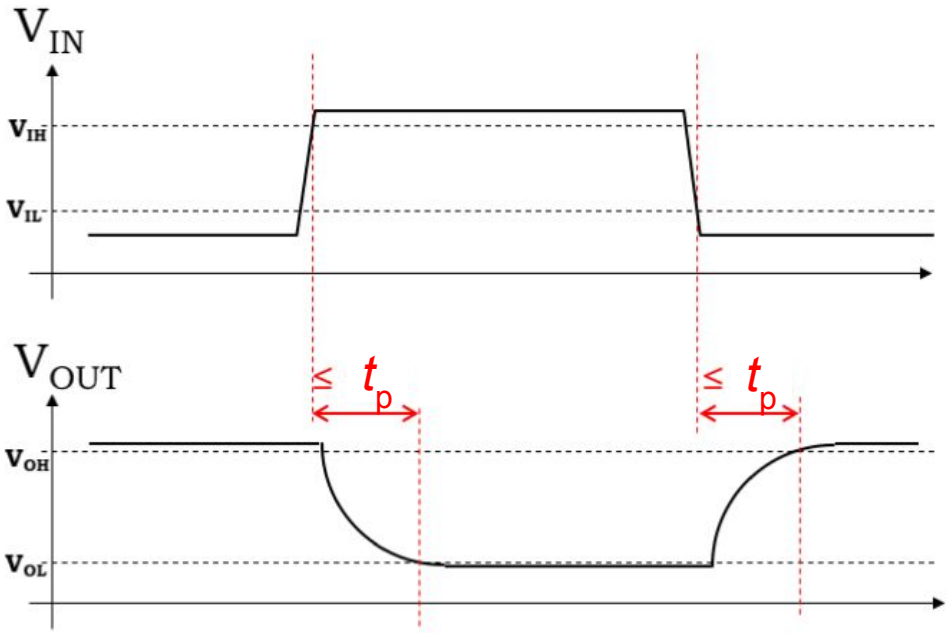


https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-004-computation-structures-spring-2017/c3/c3s1/

DIGITAL CIRCUITS

- **combinational digital circuit**
  - fundamental examples: let us not forget that everything is analog



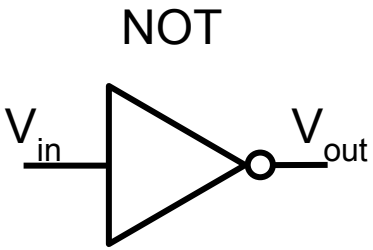


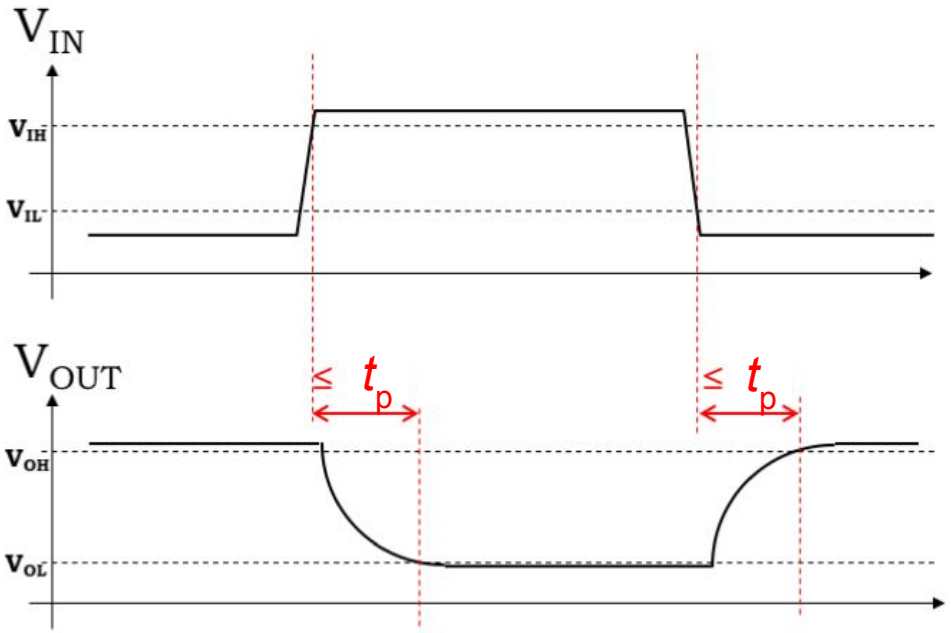
why is this tp important for computer architecture?

https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-004-computation-structures-spring-2017/c3/c3s1/

DIGITAL CIRCUITS

- **combinational digital circuit**
  - fundamental examples: let us not forget that everything is analog





a computer running at 1GHz sends commands once every 1ns

https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-004-computation-structures-spring-2017/c3/c3s1/

DIGITAL ABSTRACTION

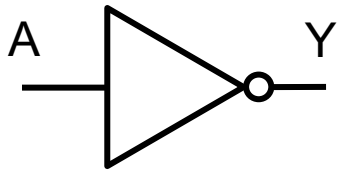
- **two important questions:**
  - why do we use digital signals instead of analog ones?
  - why do we use the binary system? would it be more advantageous to use hex?

DIGITAL ABSTRACTION

- **two important questions:**
  - why do we use digital signals instead of analog ones?
    - because of noise
    - in an analog system, noise accumulates
    - in a digital system, we have noise corrections (we have margins)
  - why do we use the binary system? would it be more advantageous to use hex?
    - yes, it would be more advantageous to use hex (it is 4 times more advantageous)
    - the problem is that, instead of two states, we would now need 16
    - this means that we have to distinguish 16 voltage levels in the presence of noise (that is, including noise margins)
    - 16 levels is probably too much ... but 4 levels would probably be feasible
    - if we had 4 levels (that is, base B = 4), we would be twice as efficient

BASIC LOGIC GATES

- **NOT**

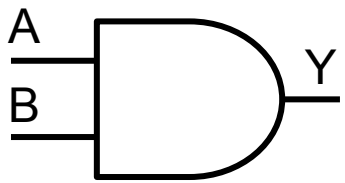


A	Y
0	1
1	0

- **notation:**
  - NOT A (explicit operation)
  - $\bar{A}$ (A bar, A complement)
  - ¬A (notation from logic)
  - ~A (not A)
  - - A (minus A)
  - A' (A prime, A complement)
  - !A (bang A)

BASIC LOGIC GATES

- **AND**

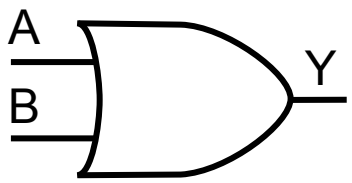


A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

- **notation:**
  - A AND B (explicit operation)
  - A x B (multiplication)
  - A • B (multiplication)
  - A * B (multiplication)
  - A . B (multiplication)
  - **AB (multiplication)**
  - A ∧ B (notation from logic)
  - A & B (bitwise operation, C)
  - A && B (logical operation, C)

BASIC LOGIC GATES

- **OR**

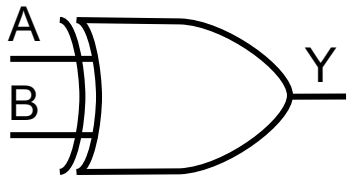


A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

- **notation:**
  - A OR B (explicit operation)
  - **A + B (addition)**
  - A ∨ B (notation from logic)
  - A | B (bitwise operation, C)
  - A || B (logical operation, C)

BASIC LOGIC GATES

- **XOR**

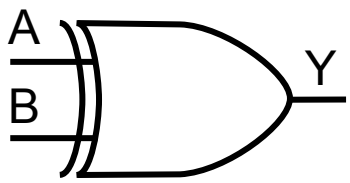


A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

- **notation:**
  - A XOR B (explicit operation)
  - **A ⊕ B (XOR addition)**
  - A ^ B (bitwise operation, C)
  - A ^^ B (does this exist ???)

BASIC LOGIC GATES

- **XOR**

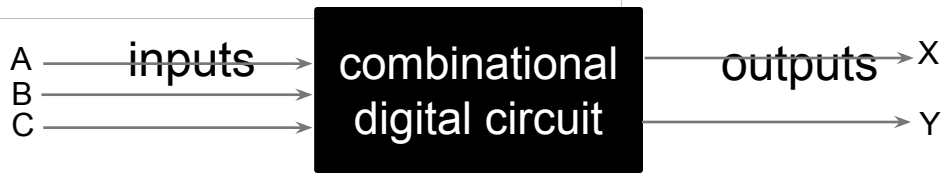


A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

- **notation:**
  - A XOR B (explicit operation)
  - **A ⊕ B (XOR addition)**
  - A ^ B (bitwise operation, C)
  - A != B (logical operation, C)

COMBINATIONAL DIGITAL CIRCUIT

- **combinational digital circuit**



A	B	C	X	Y
0	0	0	0	1
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	0

- **a Boolean expression containing the rules from the table?**
  - $X = \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$
  - Y = ... (exercise for you)

normal form, sum of products

COMBINATIONAL DIGITAL CIRCUIT

- **truth table**

A	B	C	X	Y
0	0	0	0	1
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	0

- **equivalent Boolean expression**
  - $X = \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$

- **conclusion:** **NOT, AND, and OR are universal (they can implement any combinational circuit)**
- **why is XOR missing?**

COMBINATIONAL DIGITAL CIRCUIT

- **truth table**

A	B	C	X	Y
0	0	0	0	1
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	0

- **equivalent Boolean expression**
  - $X = \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$

- **conclusion: NOT, AND, and OR are universal (they can implement any combinational circuit)**
- **why is XOR missing? $A \oplus B = \bar{A}B + A\bar{B}$**

A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

COMBINATIONAL DIGITAL CIRCUIT

- **truth table**

A	B	C	X	Y
0	0	0	0	1
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	0

- **equivalent Boolean expression**
  - $X = \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$

- **a potential problem:**
  if we have N inputs, the expression for X depends on a (potentially) large number of sums of products
- **what is the maximum number of terms in the sums for X?**

COMBINATIONAL DIGITAL CIRCUIT

- **truth table**

A	B	C	X	Y
0	0	0	0	1
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	0

- **equivalent Boolean expression**
  - $X = \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$

- **a potential problem:**
  if we have N inputs, the expression for X depends on a (potentially) large number of sums of products
- **maximum number of terms? 2^(N-1)**

COMBINATIONAL DIGITAL CIRCUIT

- **truth table**

A	B	C	X	Y
0	0	0	0	1
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	0

- **equivalent Boolean expression**
  - $X = \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$

- **a potential problem:**
  if we have N inputs, the expression for X depends on a (potentially) large number of sums of products
- **general solution: we want minimal representations**

COMBINATIONAL DIGITAL CIRCUIT

- **truth table**

A	B	C	X	Y
0	0	0	0	1
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	0

- **equivalent Boolean expression**
  - $X = \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$
- **minimal representation of X?**

COMBINATIONAL DIGITAL CIRCUIT

- **truth table**

A	B	C	X	Y
0	0	0	0	1
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	0

- **equivalent Boolean expression**

  - $X = \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$

    $=$

COMBINATIONAL DIGITAL CIRCUIT

- **truth table**

A	B	C	X	Y
0	0	0	0	1
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	0

- **equivalent Boolean expression**

  - $X = \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$

    $= \overline{A}\,(\overline{B}C + B\overline{C}) + A\,(\overline{B}\overline{C} + BC)$

    $=$

COMBINATIONAL DIGITAL CIRCUIT

- **truth table**

A	B	C	X	Y
0	0	0	0	1
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	0

- **equivalent Boolean expression**

  - $X = \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$

    $= \overline{A}\,(\overline{B}C + B\overline{C}) + A\,(\overline{B}\overline{C} + BC)$

    $= \overline{A}\,(B \oplus C) + A\,\overline{(B \oplus C)}$

COMBINATIONAL DIGITAL CIRCUIT

- **truth table**

A	B	C	X	Y
0	0	0	0	1
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	0

- **equivalent Boolean expression**

  - $X = \overline{A}\overline{B}C + \overline{A}B\overline{C} + A\overline{B}\overline{C} + ABC$

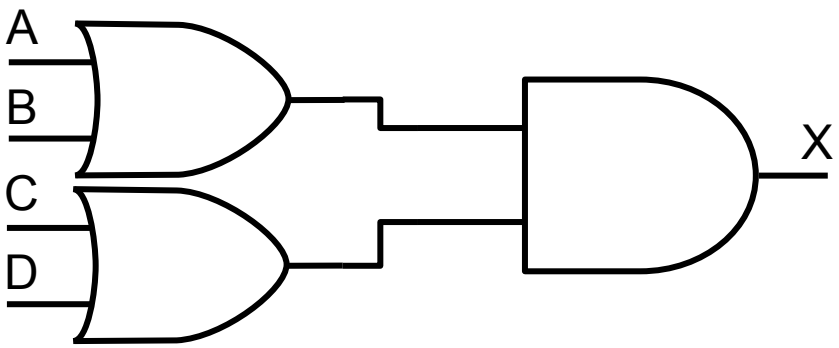
    $= \overline{A}\,(\overline{B}C + B\overline{C}) + A\,(\overline{B}\overline{C} + BC)$

    $= \overline{A}\,(B \oplus C) + A\,\overline{(B \oplus C)}$

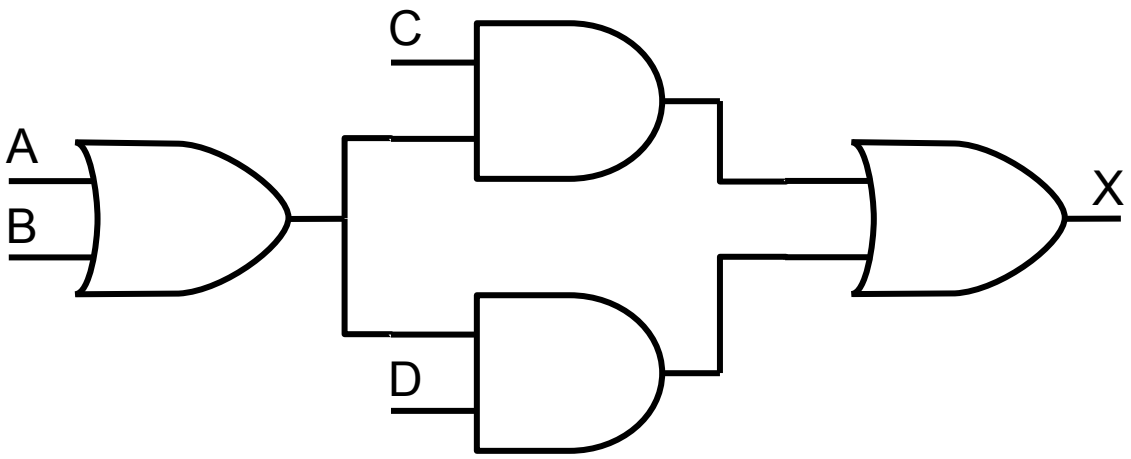
    $= A \oplus B \oplus C$

COMBINATIONAL DIGITAL CIRCUIT

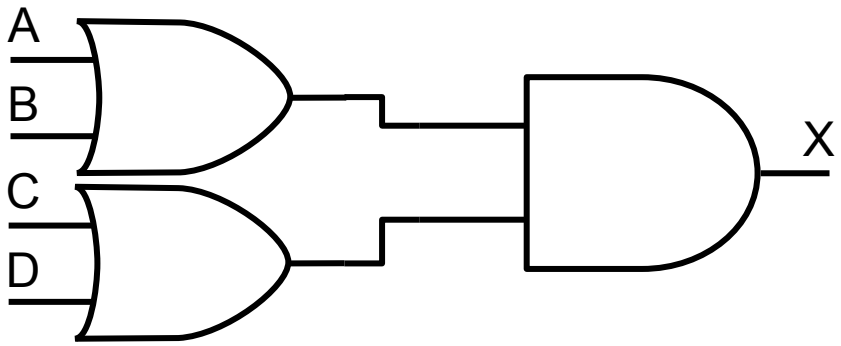
- **now consider**
  - X = AC + BC + AD + BD
    = (A + B)C + (A + B)D
    = (A + B)(C + D)



COMBINATIONAL DIGITAL CIRCUIT



- **versus**



in general, it is good to carry the grouping calculations all the way through
but remember that we also care about propagation time

number of gates vs. computation stages vs. gate type ...

MORE COMPLEX CIRCUITS

- the important conclusion:

- everything we do on a computing system must reduce to circuits that are logic gates

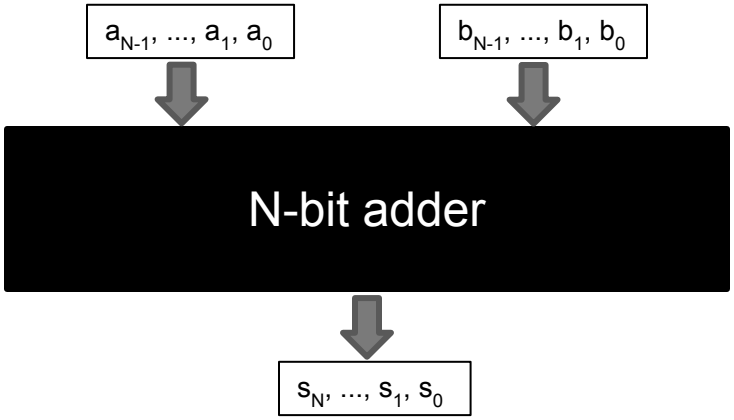
- nothing else exists at this level

MORE COMPLEX CIRCUITS

- **addition circuit**
  - given a and b on N bits
  - we want to compute s = a + b
    - how many bits does s have?
    -

MORE COMPLEX CIRCUITS

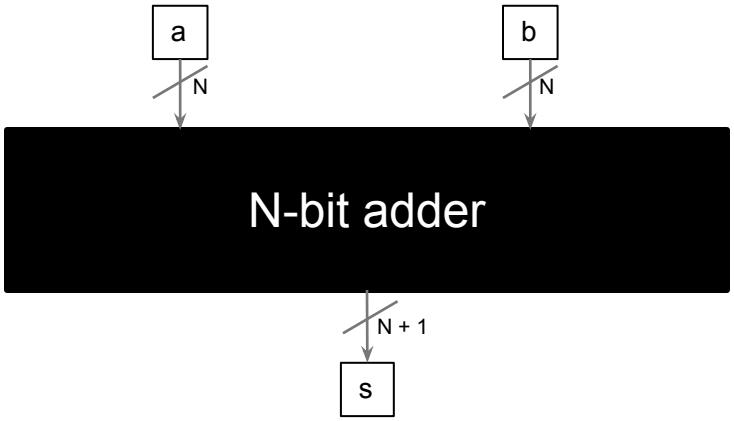
- **addition circuit**
  - given a and b on N bits
  - we want to compute s = a + b
    - how many bits does s have?
    - *N+1 bits*



  - how many inputs do we have?
  - how many outputs?
  - how large will the combinational circuit be?

MORE COMPLEX CIRCUITS

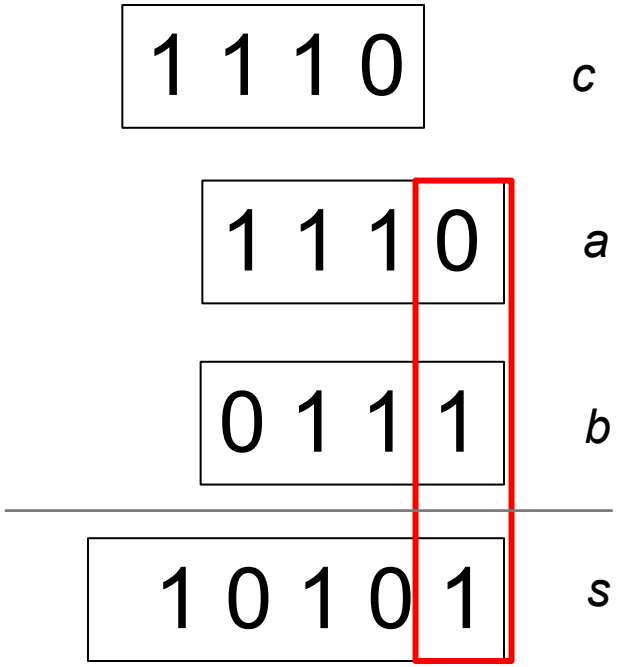
- **addition circuit**
  - given a and b on N bits
  - we want to compute s = a + b
    - how many bits does s have?
    - *N+1 bits*



    - how many inputs do we have? 2N
    - how many outputs? N + 1
    - how large will the combinational circuit be? $(N+1)2^{2N-1}$

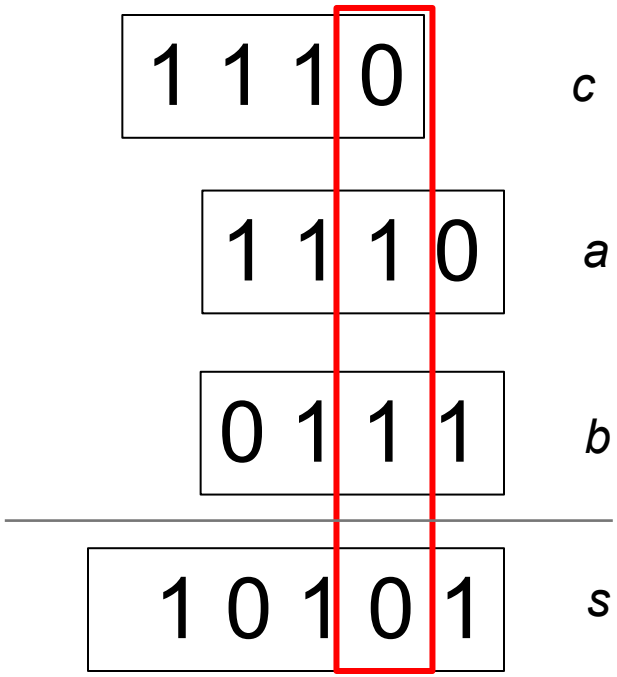
MORE COMPLEX CIRCUITS

- **addition circuit**
  - how have we performed binary addition so far?
    - $s = a + b$
    - example:



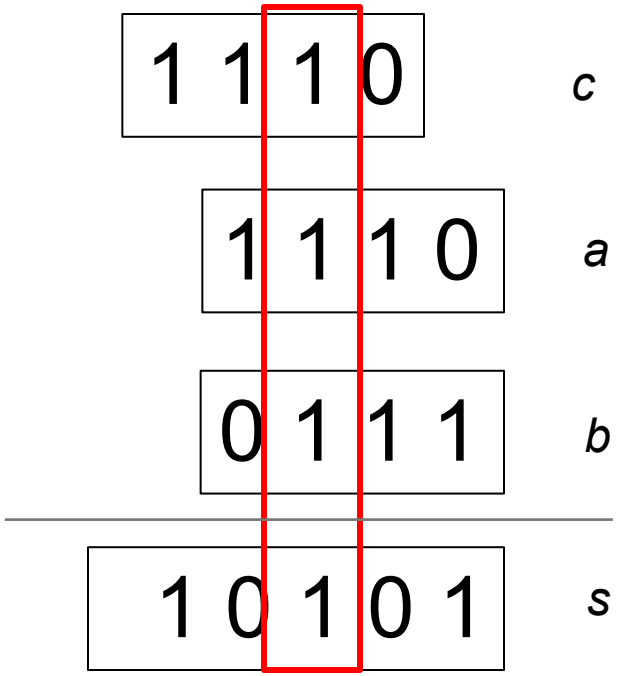
MORE COMPLEX CIRCUITS

- **addition circuit**
  - how have we performed binary addition so far?
    - $s = a + b$
    - example:



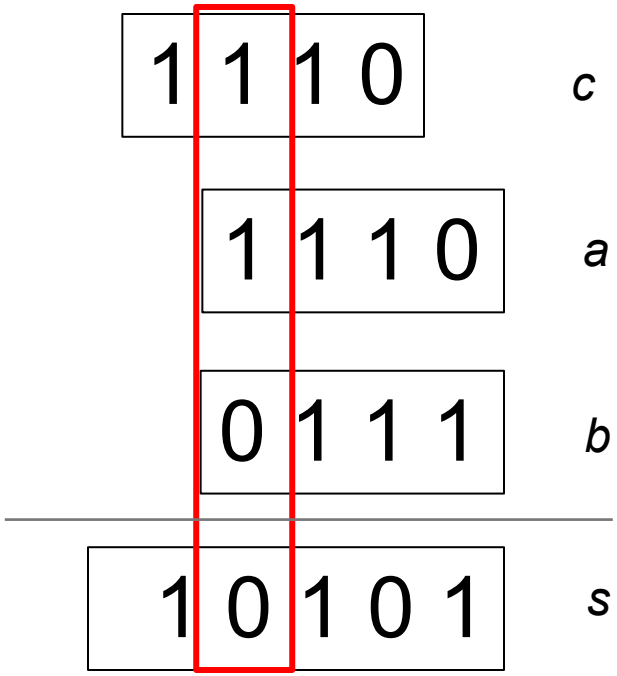
MORE COMPLEX CIRCUITS

- **addition circuit**
  - how have we performed binary addition so far?
    - $s = a + b$
    - example:



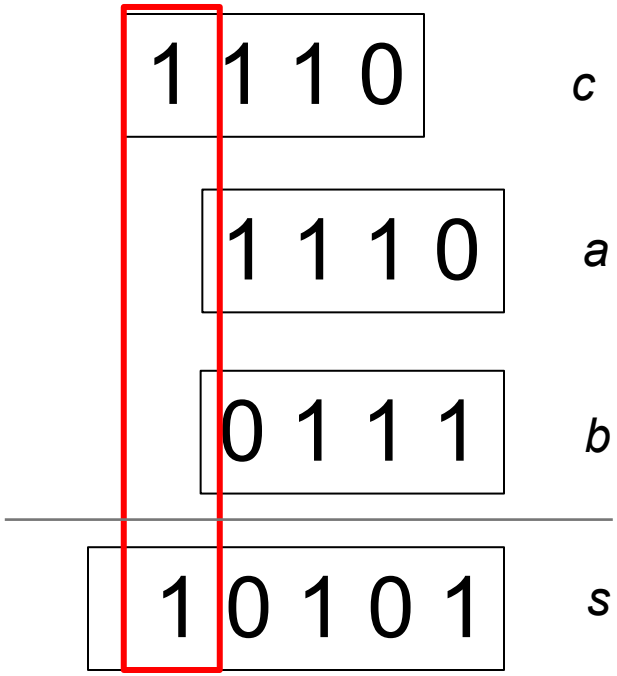
MORE COMPLEX CIRCUITS

- **addition circuit**
  - how have we performed binary addition so far?
    - $s = a + b$
    - example:



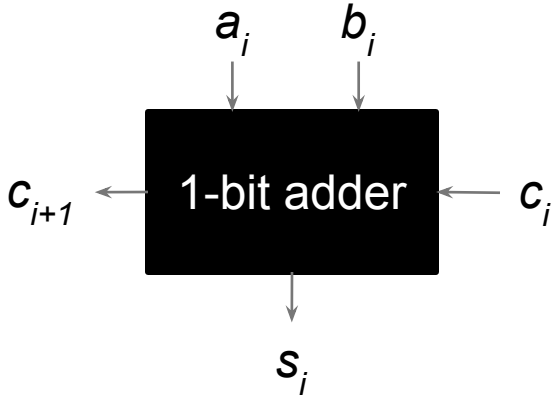
MORE COMPLEX CIRCUITS

- **addition circuit**
  - how have we performed binary addition so far?
    - $s = a + b$
    - example:



MORE COMPLEX CIRCUITS

- **addition circuit**
  - the idea: define a fundamental circuit block on which we base everything



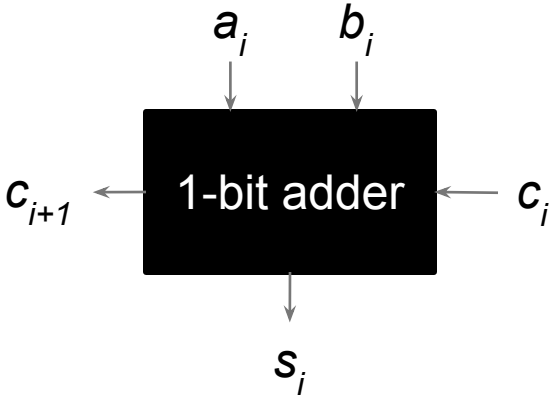
s_i = ?

c_{out} = ?

a_i	b_i	c_{in}/c_i	s_i	c_{out}/c_{i+1}
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

MORE COMPLEX CIRCUITS

- **addition circuit**
  - the idea: define a fundamental circuit block on which we base everything



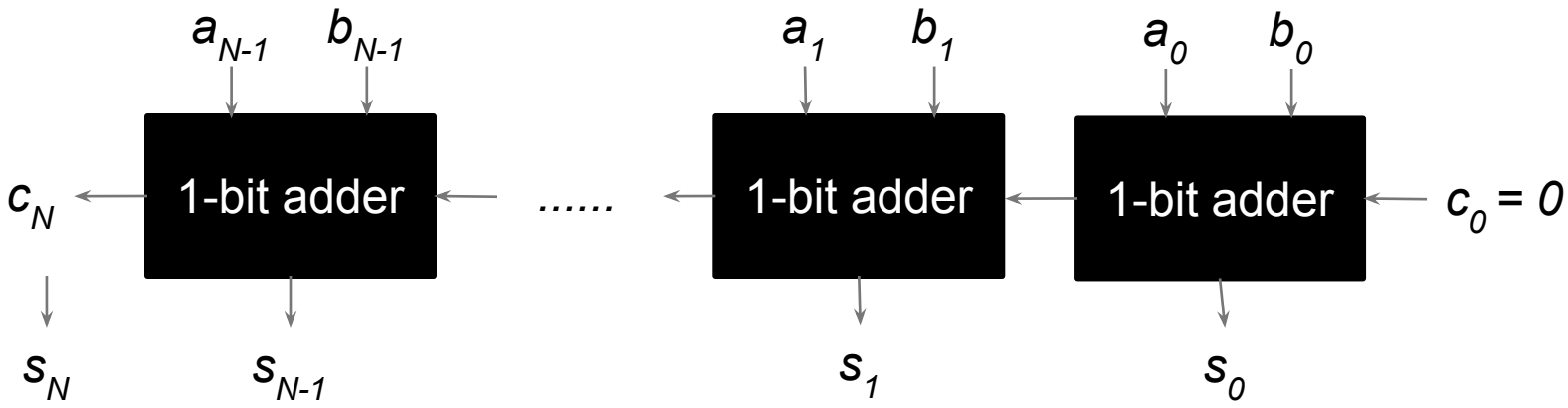
$$s_i = a_i \oplus b_i \oplus c_{in}$$

$$c_{out} = a_i b_i + (a_i + b_i) c_{in}$$

a_i	b_i	c_{in}/c_i	s_i	c_{out}/c_{i+1}
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

MORE COMPLEX CIRCUITS

- **addition circuit**
  - the idea: define a fundamental circuit block on which we base everything
  - the idea: use the fundamental circuit in cascade

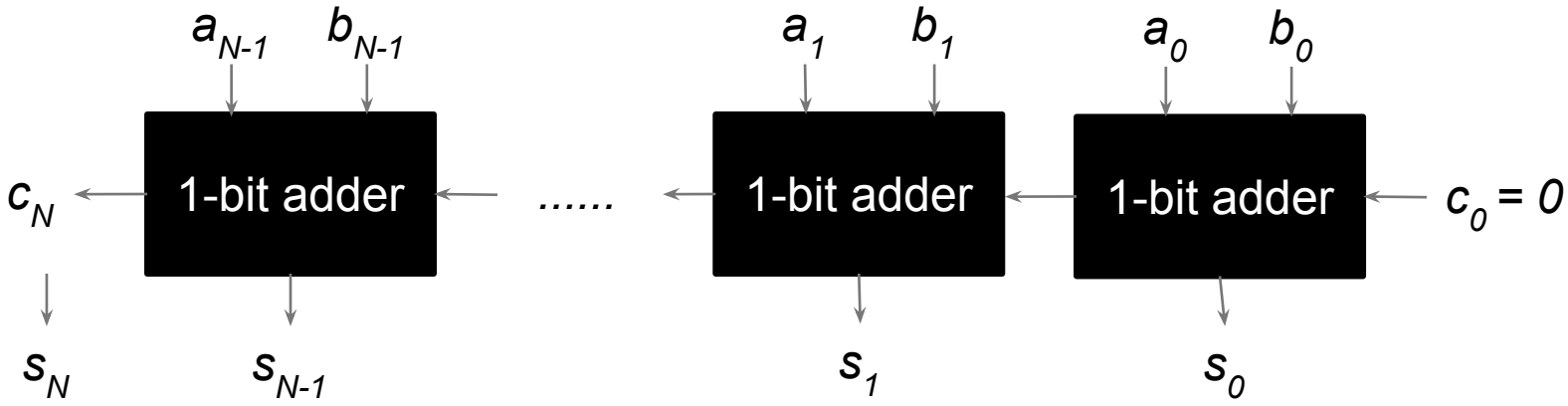


$$s_i = a_i \oplus b_i \oplus c_{in}$$

$$c_{out} = a_i b_i + (a_i + b_i) c_{in}$$

MORE COMPLEX CIRCUITS

- **addition circuit**
  - the idea: define a fundamental circuit block on which we base everything
  - the idea: use the fundamental circuit in cascade



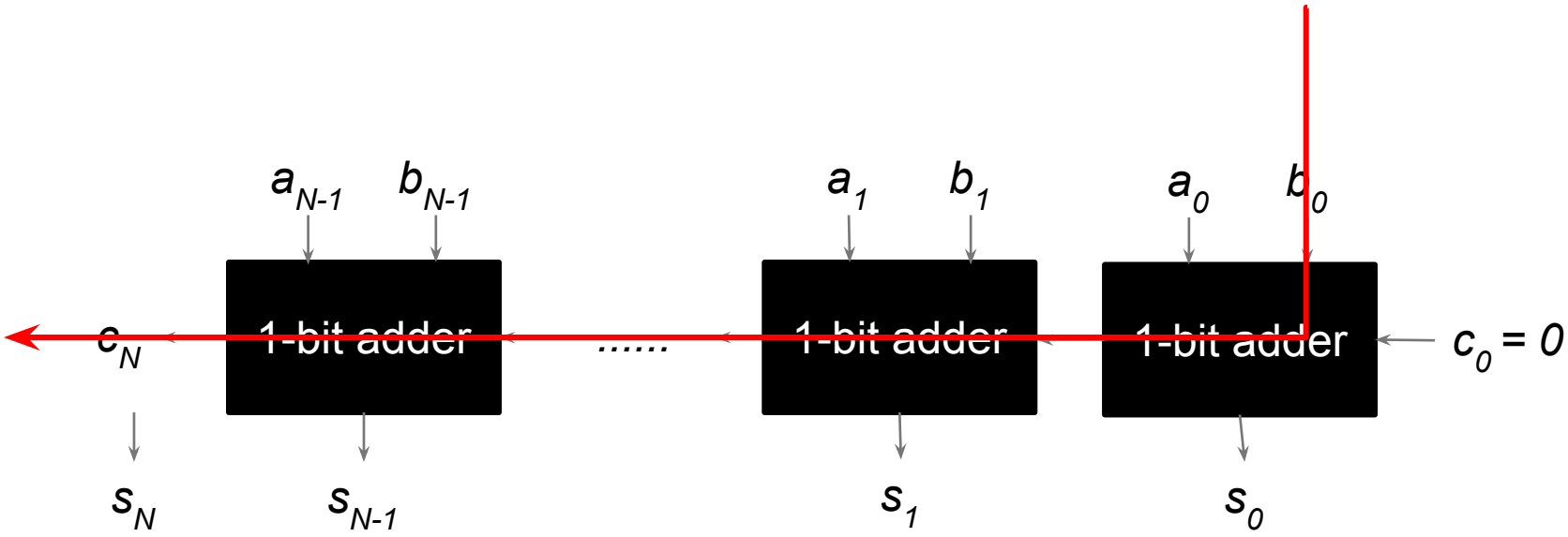
- last time we talked about t_P
- what is the propagation time in this case?

$$s_i = a_i \oplus b_i \oplus c_{in}$$

$$c_{out} = a_i b_i + (a_i + b_i) c_{in}$$

MORE COMPLEX CIRCUITS

- **addition circuit**
  - the idea: define a fundamental circuit block on which we base everything
  - the idea: use the fundamental circuit in cascade



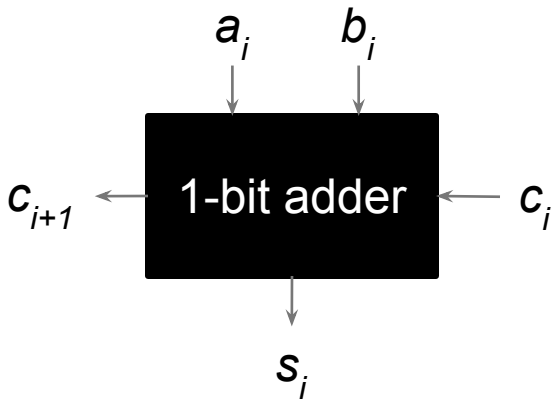
$$s_i = a_i \oplus b_i \oplus c_{in}$$

$$c_{out} = a_i b_i + (a_i + b_i) c_{in}$$

- last time we talked about t_P
- what is the propagation time in this case?
- $N\, t_{P,\ 1\text{-bit adder}}$

MORE COMPLEX CIRCUITS

- **addition circuit**
  - what is the fundamental problem in the circuit above?
    - we have to wait until we can compute those carry bits



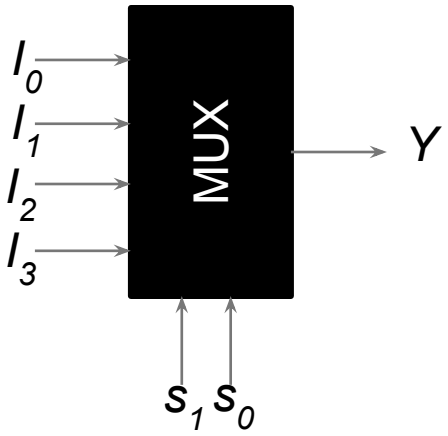
$$s_i = a_i \oplus b_i \oplus c_{in}$$

$$c_{out} = a_i b_i + (a_i + b_i) c_{in}$$

- we denote $g_i = a_i b_i$ and $p_i = a_i + b_i$
  - *if $g_i = 1$ then $c_{out} = 1$*
  - *if $g_i = 0$ and $p_i = 0$ then $c_{out} = 0$*
  - *if $g_i = 0$ and $p_i = 1$ then $c_{out} = c_{in}$*

MORE COMPLEX CIRCUITS

- **multiplexing**
  - a circuit that selects a digital input signal based on an activation signal s

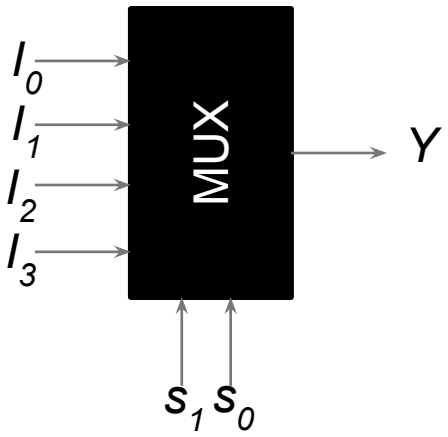


s_1	s_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

  - why is this circuit useful?

MORE COMPLEX CIRCUITS

- **multiplexing**
  - a circuit that selects a digital input signal based on an activation signal s

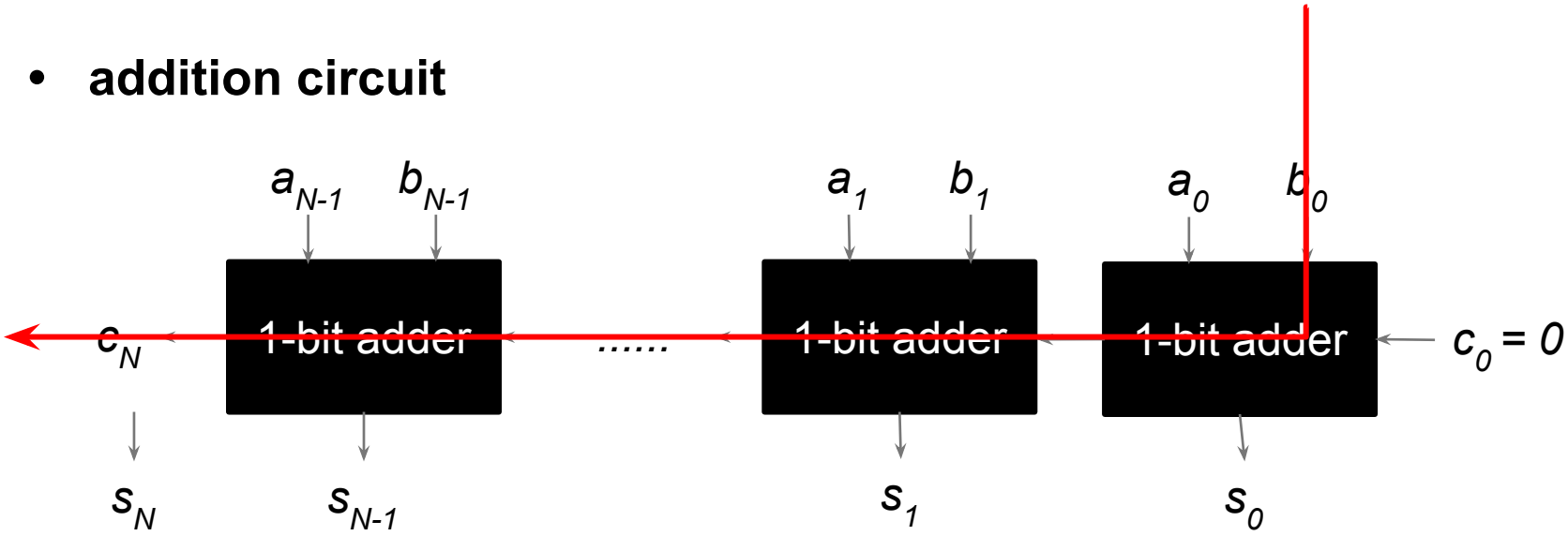


s_1	s_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

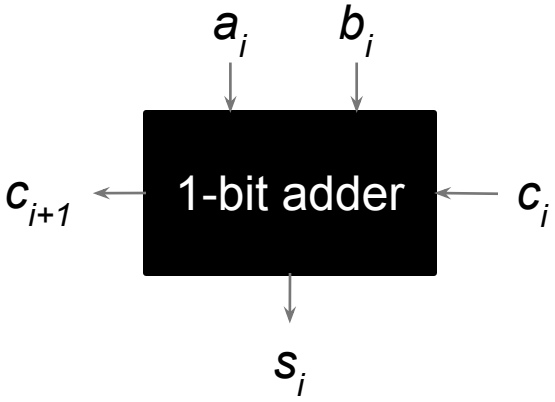
  - why is this circuit useful?
    - hardware implementation for “if” and “case”
    - hardware implementation for shift operations
    - we will see this in the seminar

MORE COMPLEX CIRCUITS

- **addition circuit**



- what is the fundamental problem in the circuit above?
  - we have to wait until we can compute those carry bits

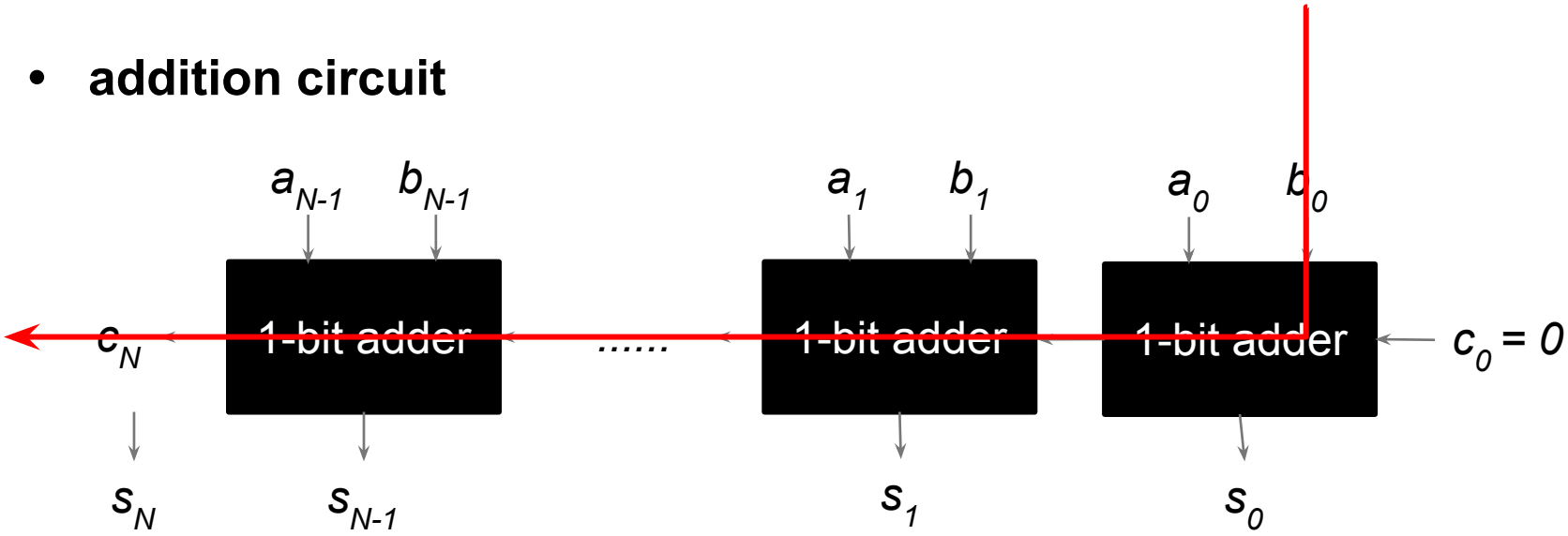


$$s_i = a_i \oplus b_i \oplus c_{in}$$

$$c_{out} = a_i b_i + (a_i + b_i) c_{in}$$

MORE COMPLEX CIRCUITS

- **addition circuit**

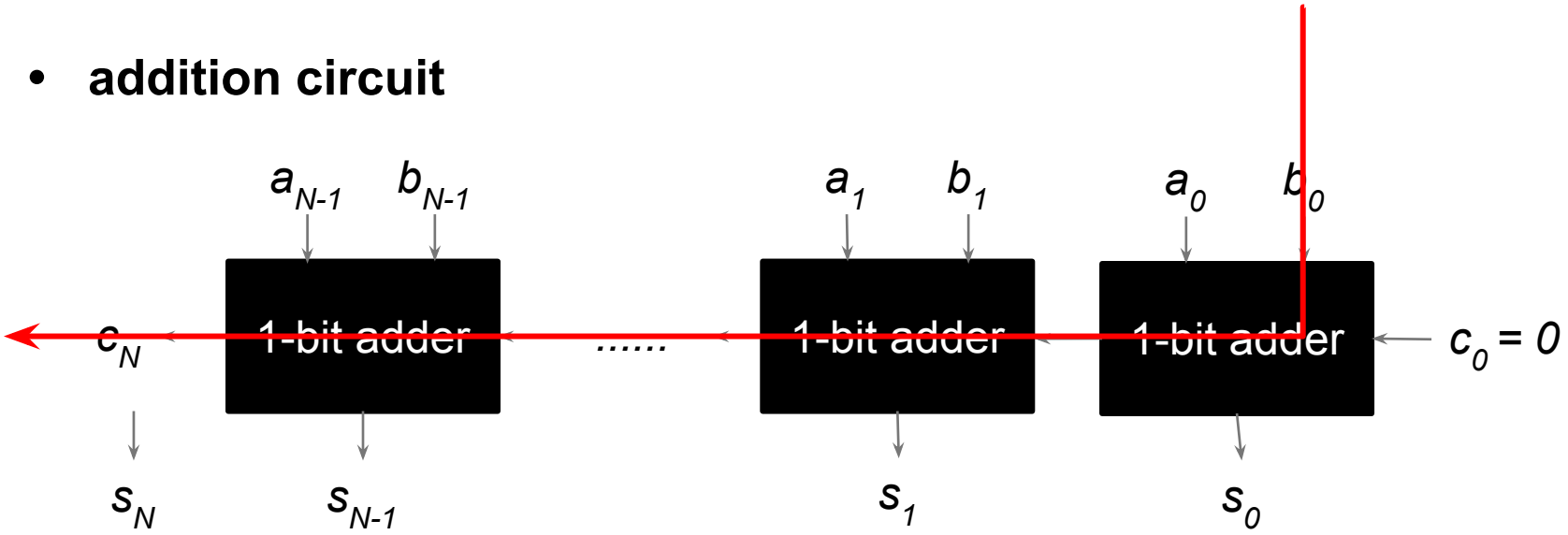


- what is the propagation time in this case?
- $N\,t_P$

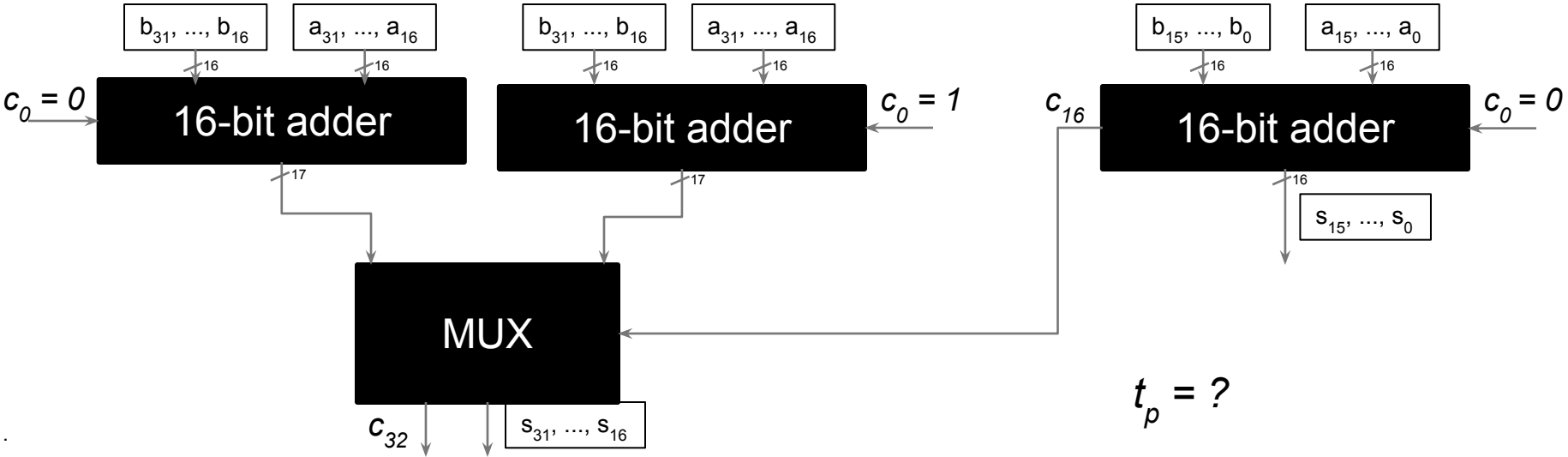
- what can we do to improve the propagation time?

MORE COMPLEX CIRCUITS

- **addition circuit**

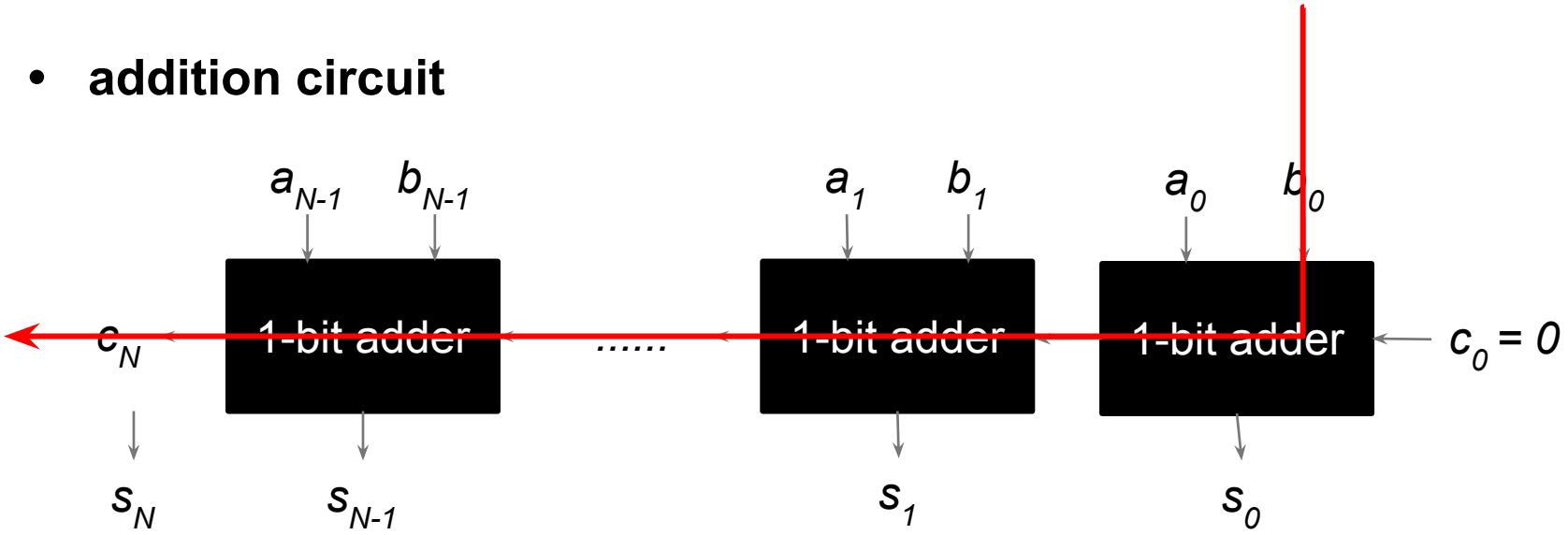


- what is the propagation time in this case?
- $N\,t_P$

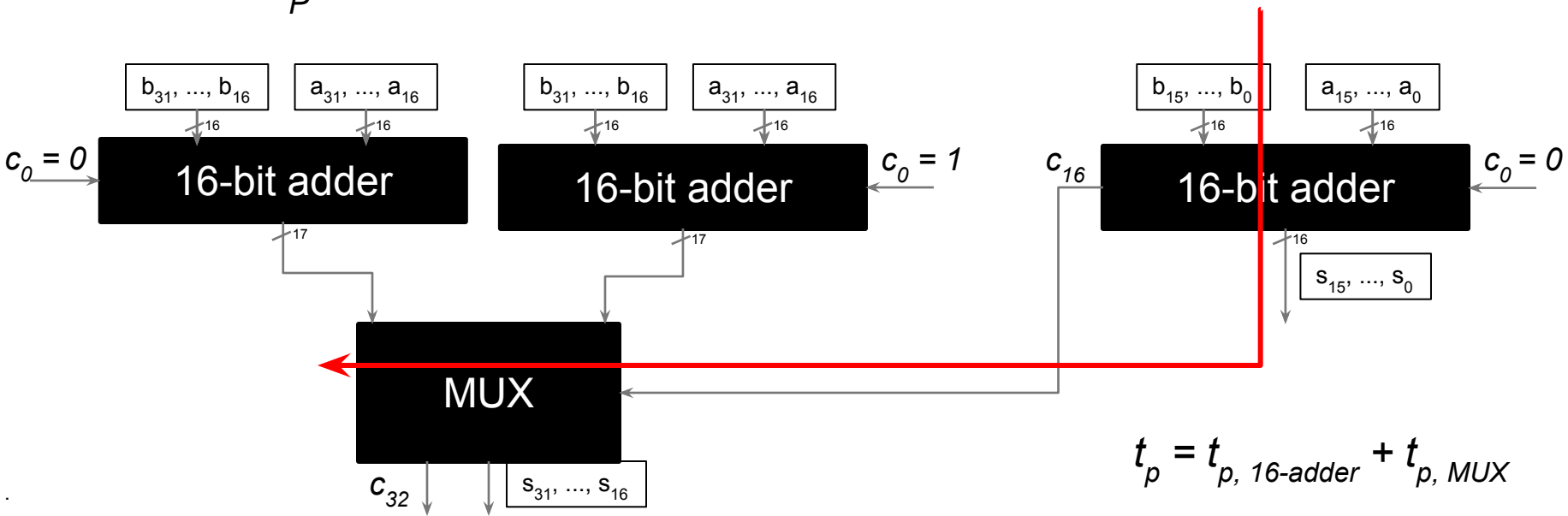


MORE COMPLEX CIRCUITS

- **addition circuit**



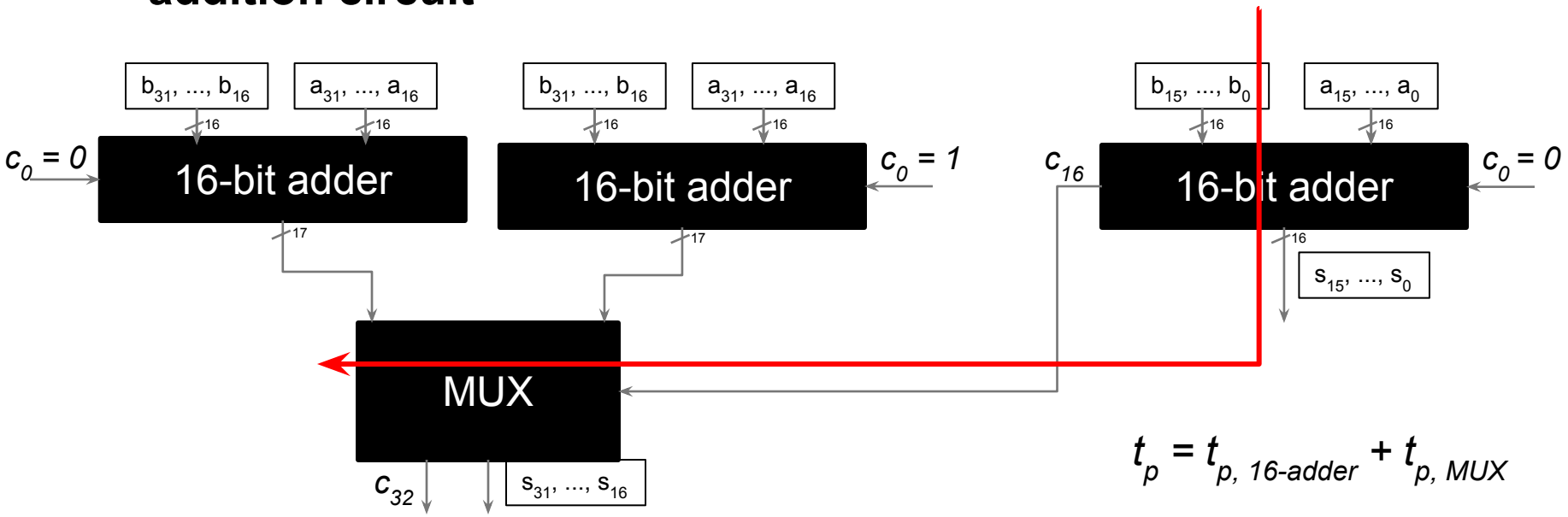
- what is the propagation time in this case?
- $N\,t_P$



$$t_p = t_{p,\ 16\text{-adder}} + t_{p,\ MUX}$$

MORE COMPLEX CIRCUITS

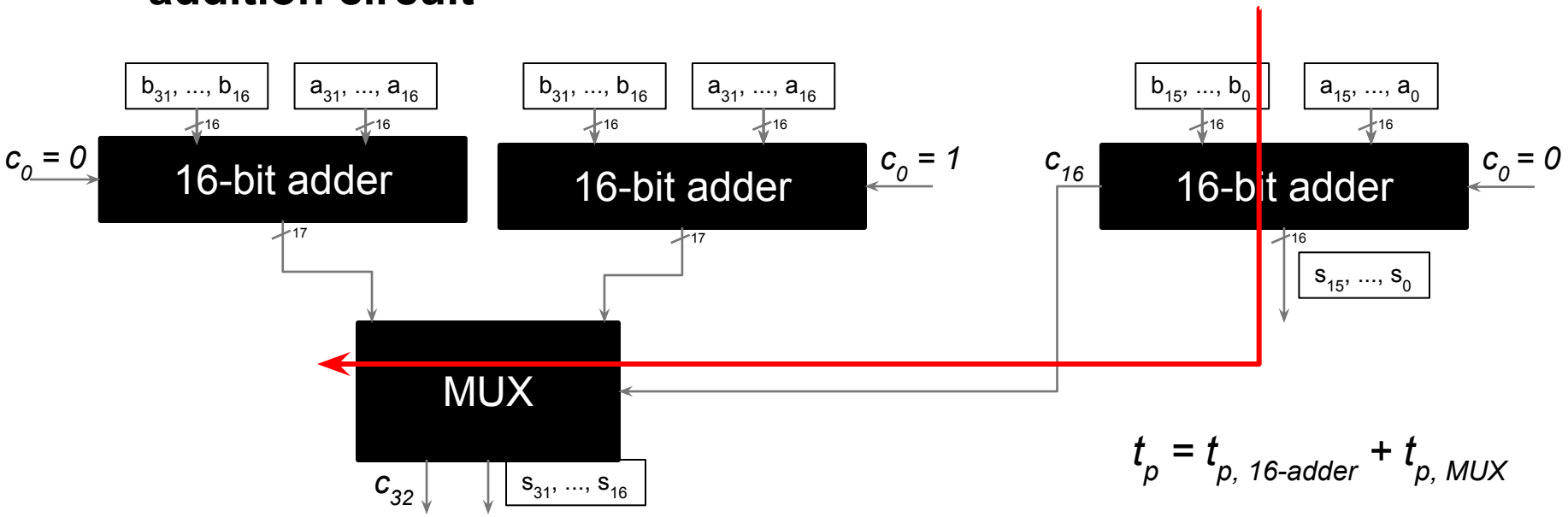
- **addition circuit**



- here we considered a 32-bit addition example
- we can apply the same idea to the 16-bit circuits above
- t_p = ?

MORE COMPLEX CIRCUITS

- **addition circuit**



- here we considered a 32-bit addition example
- we can apply the same idea to the 16-bit circuits above
- $t_p = O(\log_2 N)$

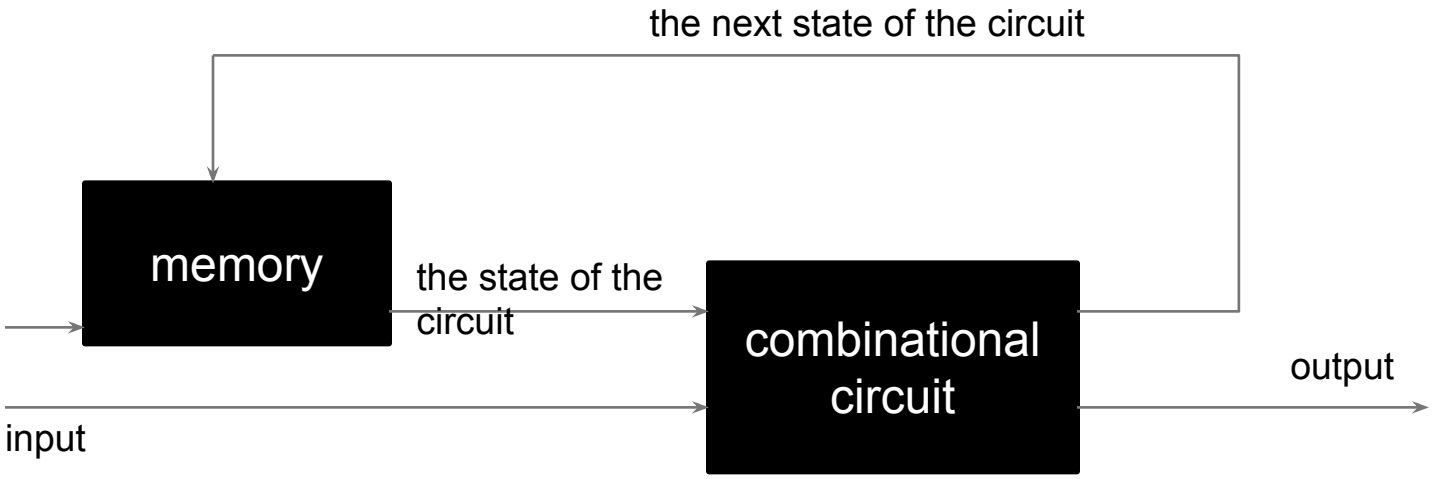
SEQUENTIAL CIRCUITS

- **combinational circuits are efficient (in general), but they have a major problem: they are one-shot**
  - we cannot iterate
  - they do not allow any kind of “internal logic” or “internal memory”
  - there is no internal state of the circuit
  - they are too simple: they associate a logical function of the inputs with an output (after a certain time)
  - some things cannot be implemented using only combinational logic

- **sequential circuits address some of the limitations of combinational circuits**

SEQUENTIAL CIRCUITS

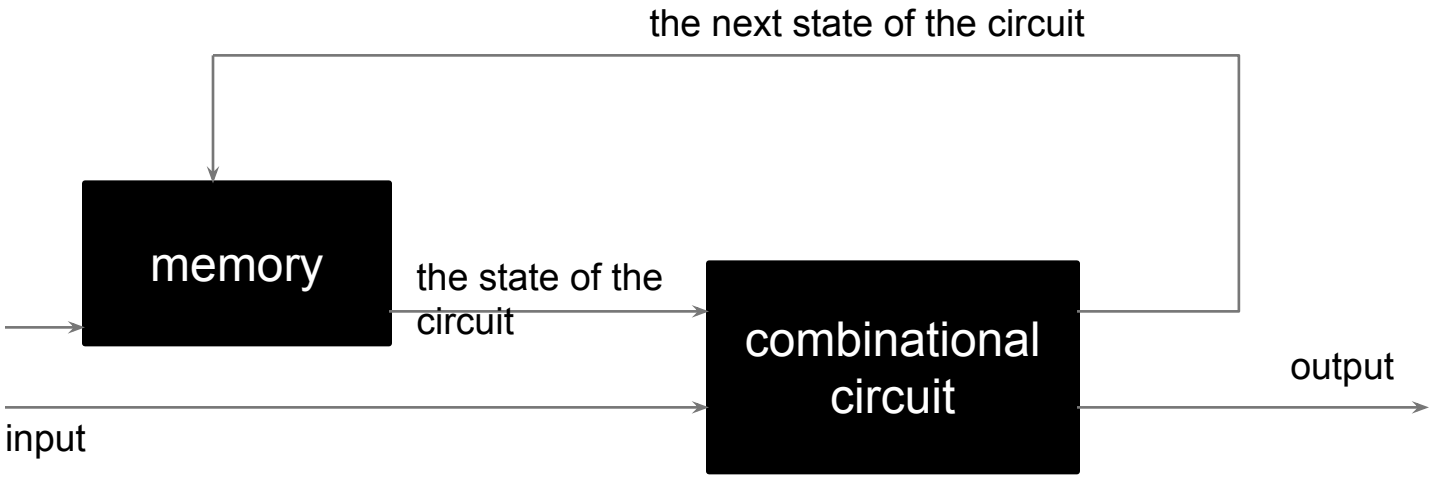
- **we need to introduce “memory”-type elements**



- **in this way we add a state to the circuit**

SEQUENTIAL CIRCUITS

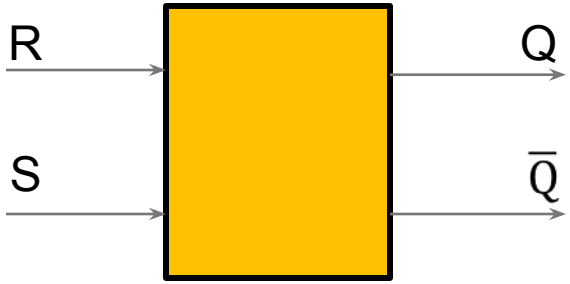
- **we need to introduce “memory”-type elements**

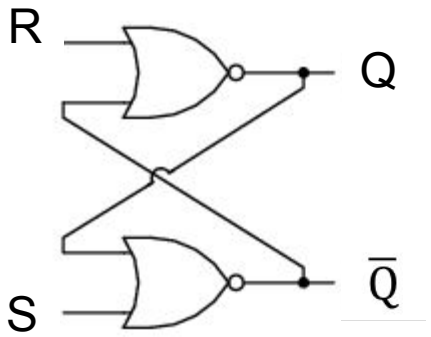


- **so far we have said that the bits we represent are voltage**
- **but electrical energy is difficult to store (over time)**
  - leakage phenomenon
- **if we want to store something, we must refresh it from time to time to update the electrical energy level**

SEQUENTIAL CIRCUITS

- **SR Latch (Set-Reset Latch)**
  - stores one bit of information

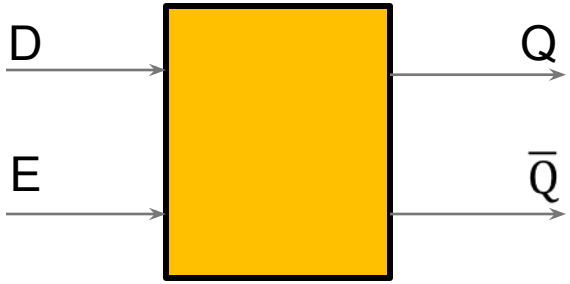


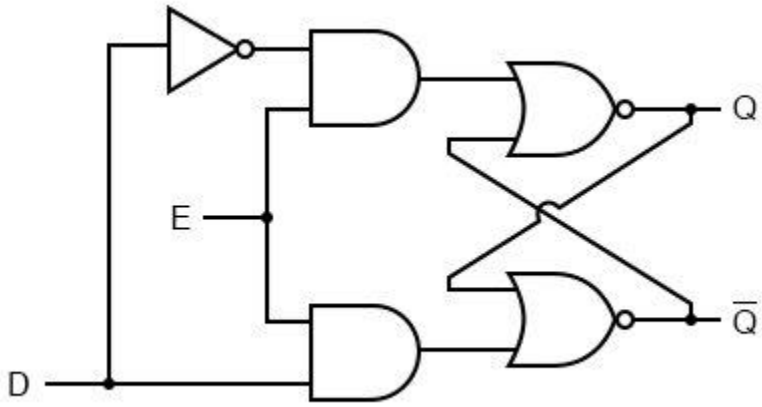


S	R	Q		
0	0	latch	latch	nothing changes
0	1	0	1	here we put “0” in memory
1	0	1	0	here we put “1” in memory
1	1	0	0	invalid state

SEQUENTIAL CIRCUITS

- **SR Latch (Set-Reset Latch)**
  - it is good, but it has two inputs; can we do something with a single input?
  - **D Latch**



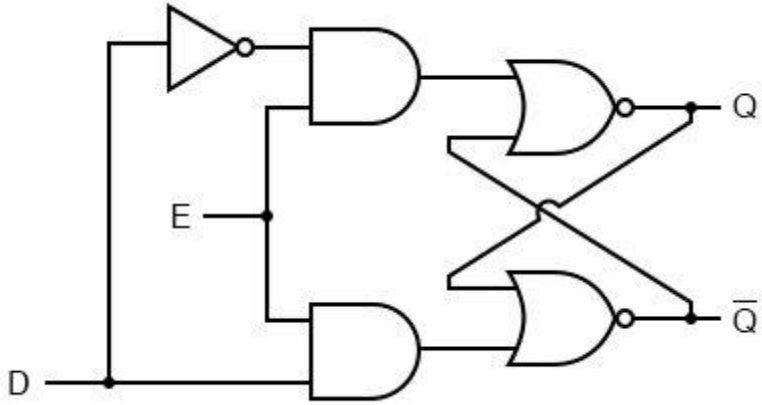


E	D	Q		
0	0	latch	latch	nothing changes
0	1	latch	latch	nothing changes
1	0	0	1	here we put “0” in memory
1	1	1	0	here we put “1” in memory

E = Enable (activation)
if E = 0, nothing happens

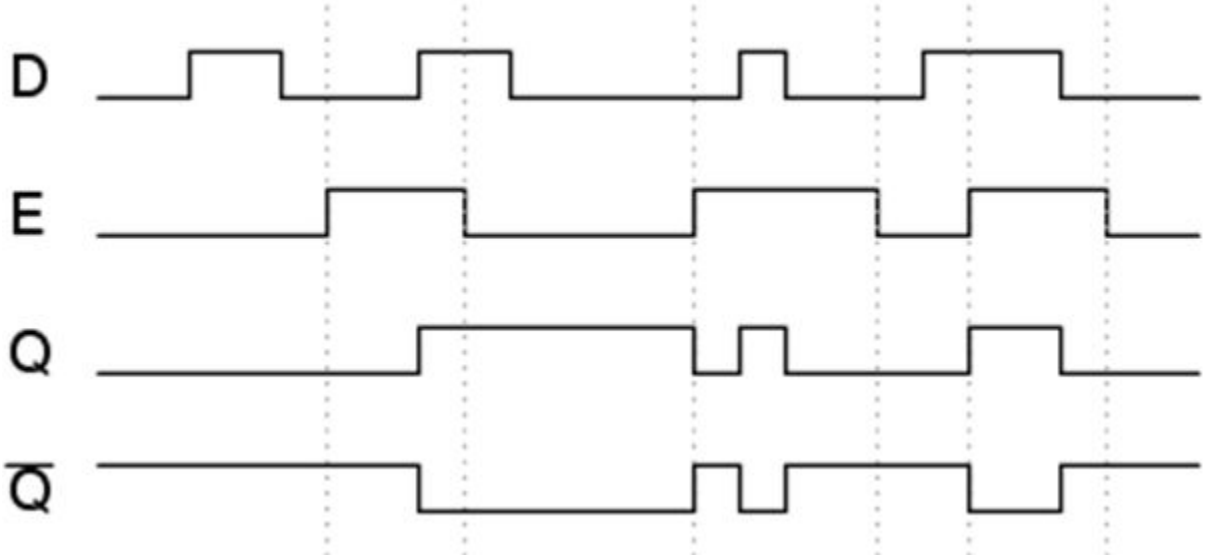
SEQUENTIAL CIRCUITS

- **D Latch over time**



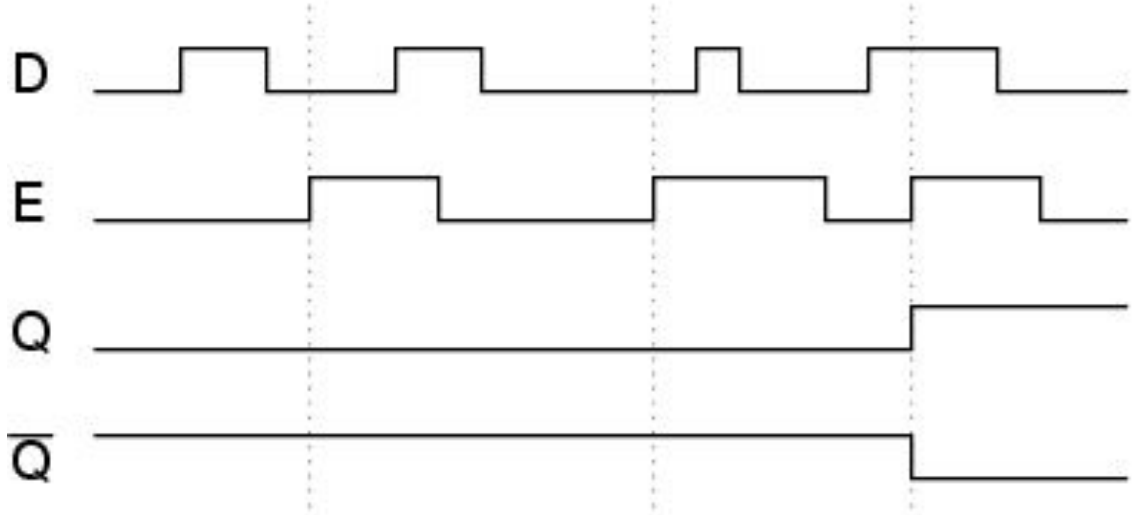
E	D	Q		
0	0	latch	latch	nothing changes
0	1	latch	latch	nothing changes
1	0	0	1	here we put “0” in memory
1	1	1	0	here we put “1” in memory

E = Enable (activation)
if E = 0, nothing happens



SEQUENTIAL CIRCUITS

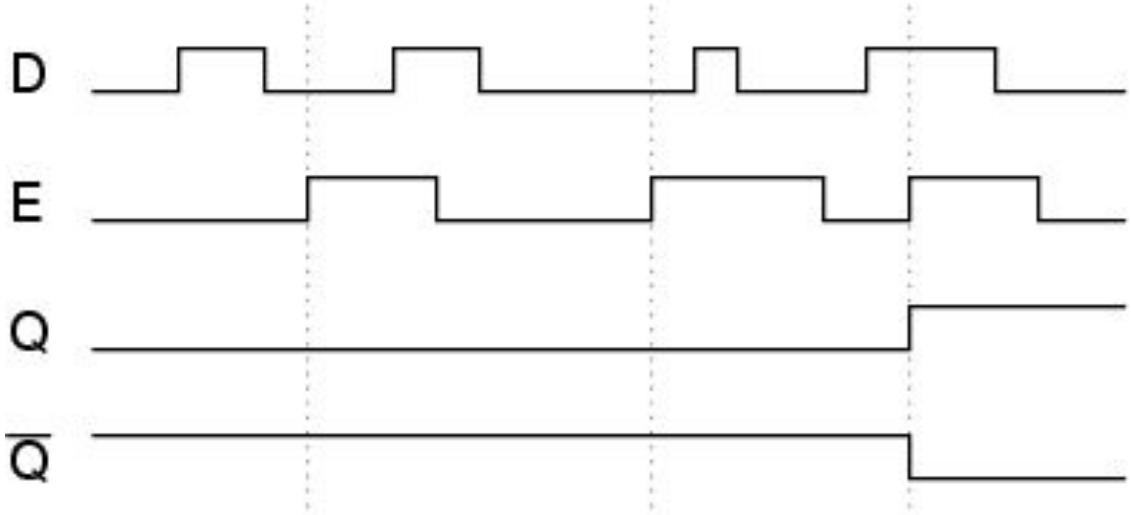
- **D Flip-Flop**
  - D Latch is good, but we want to synchronize several devices
  - we want activation to happen when E rises, not while E is active



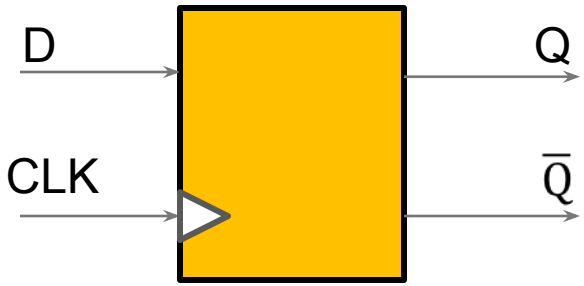
  - E becomes clock — that is, the system clock
    - at a fixed time interval, the system does something

SEQUENTIAL CIRCUITS

- **D Flip-Flop**



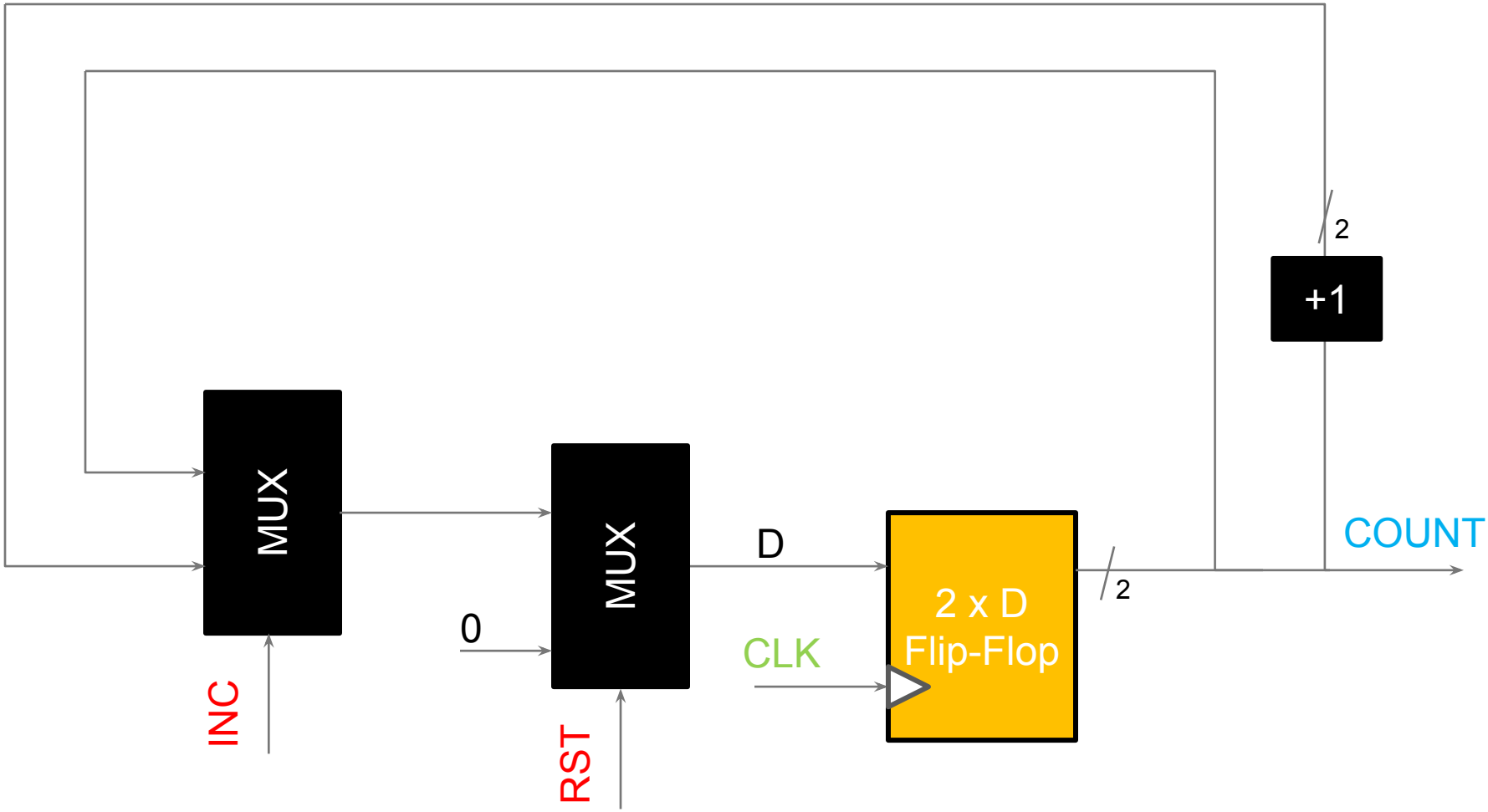
- E becomes clock — that is, the system clock
  - at a fixed time interval (one cycle), the system does something



D Flip-Flops sharing the same CLK form a register

SEQUENTIAL CIRCUITS

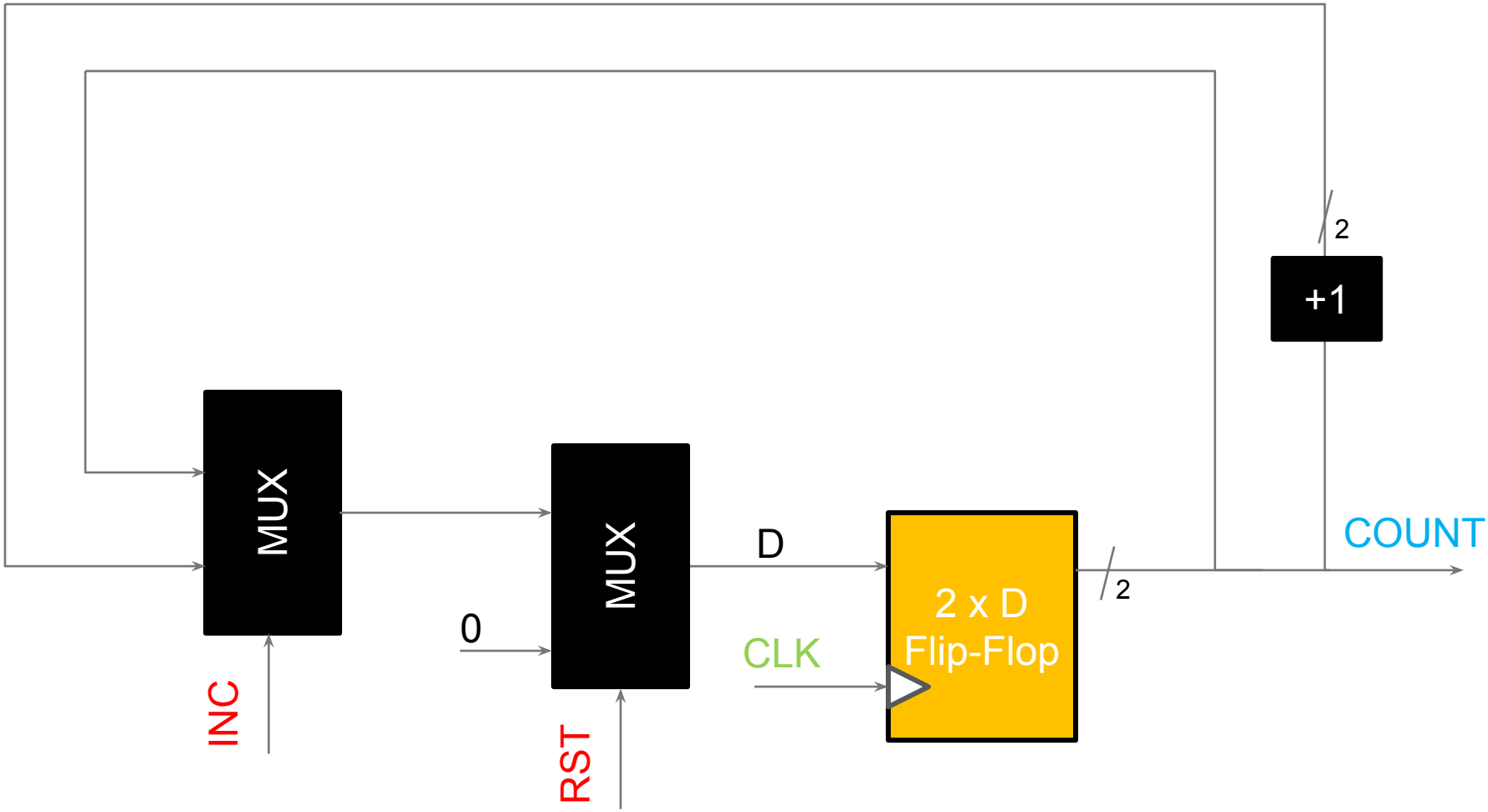
- **an example**



what does this circuit do?

SEQUENTIAL CIRCUITS

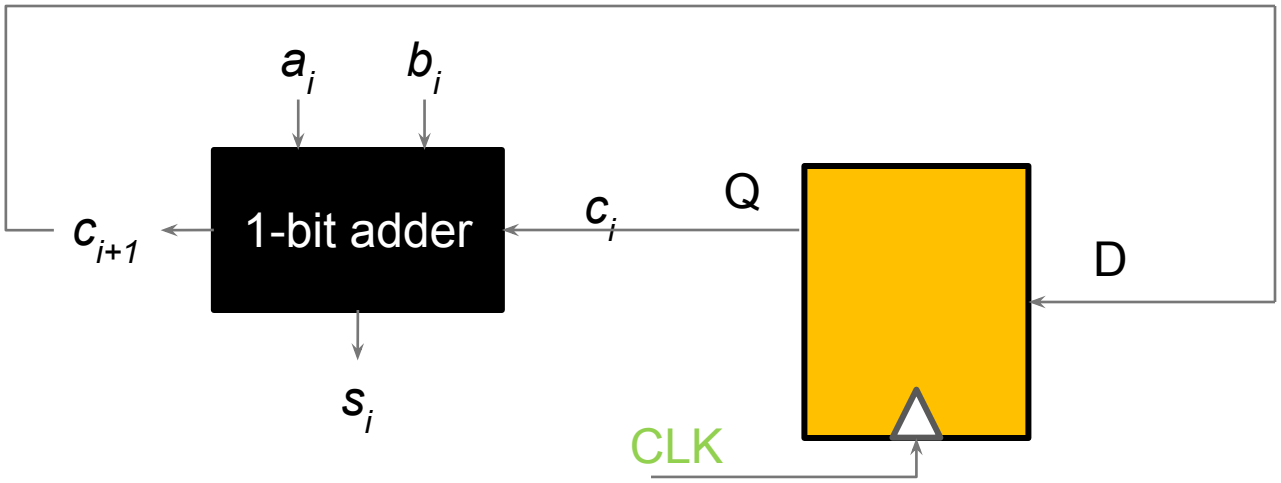
- **an example**



it is a 2-bit counter

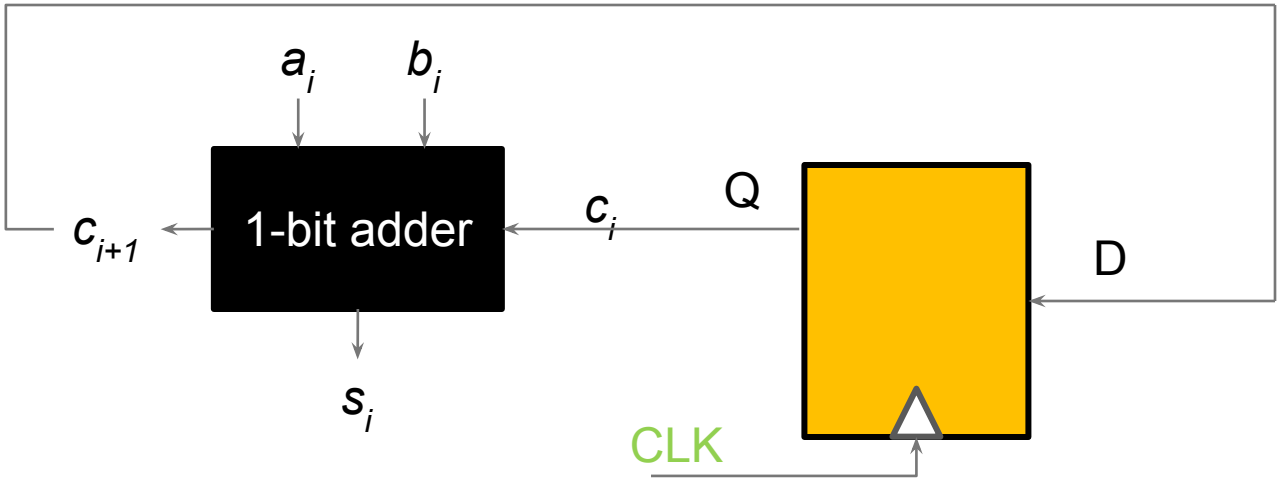
SEQUENTIAL CIRCUITS

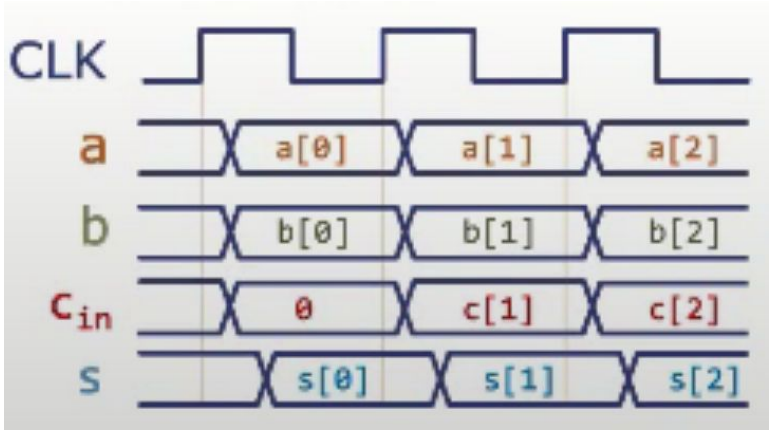
- **advantages of sequential circuits**
  - we have internal state
    - example: we have a register in which we store the current count
  - we have the time variable
    - the inputs/outputs are not fixed
    - a variable number of steps in the solution
    - example: build a circuit that can add two numbers of variable size (number of bits) — that is, we do not predefine how many bits each number has
      - this is how we perform binary addition. what does the circuit look like?



SEQUENTIAL CIRCUITS

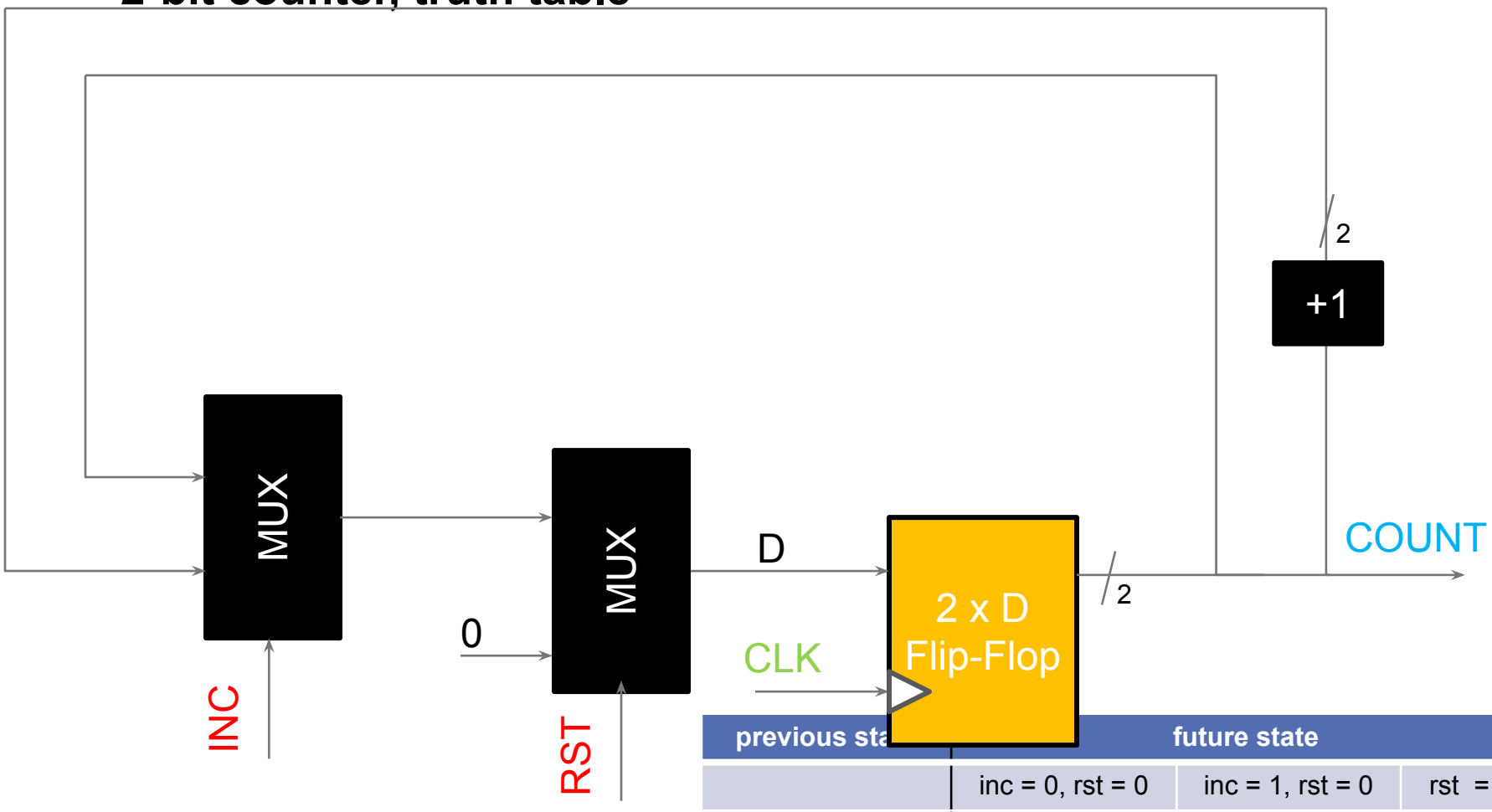
- **addition circuit, over time**





SEQUENTIAL CIRCUITS

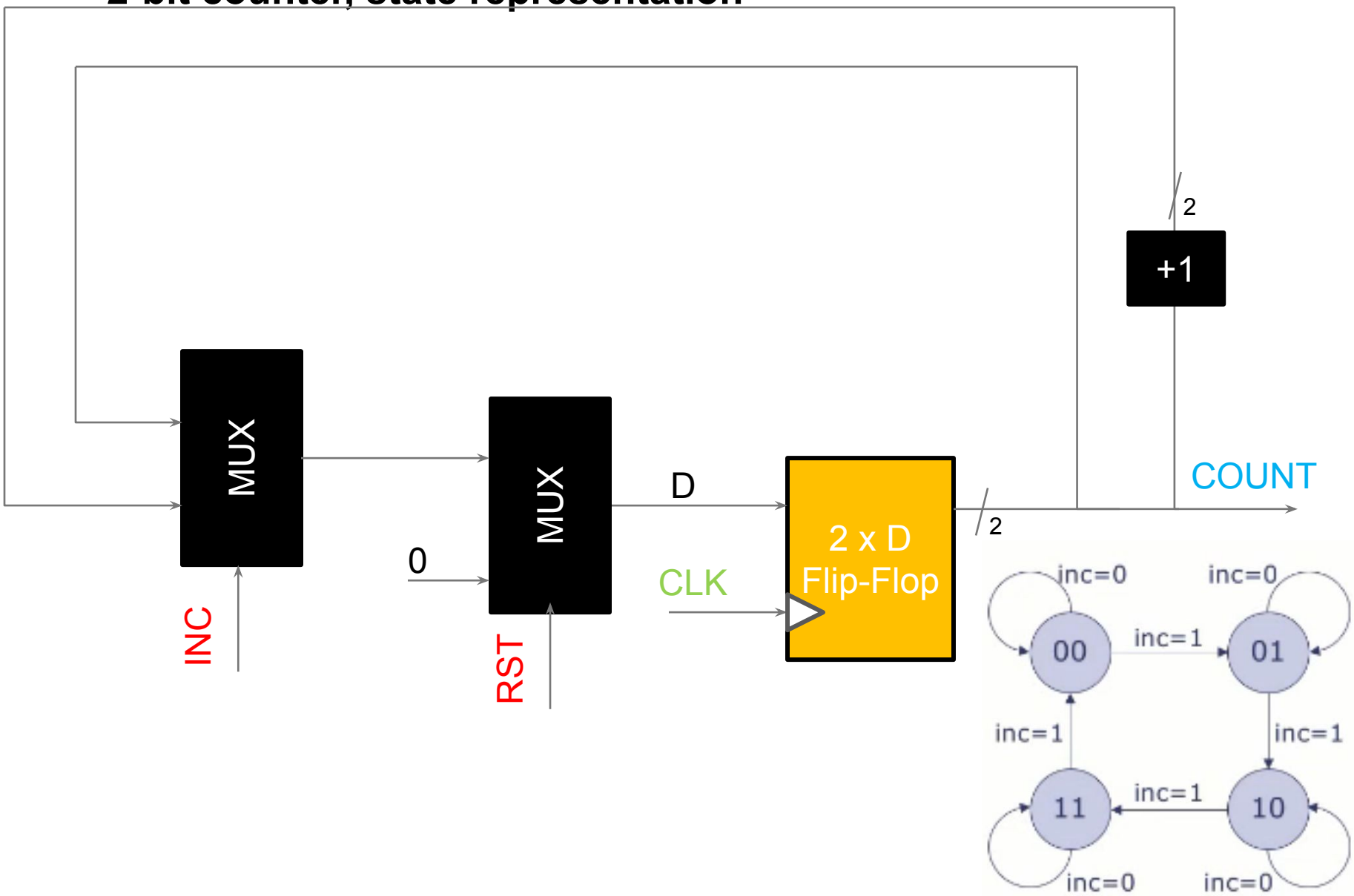
- **2-bit counter, truth table**



previous sta	future state		
	inc = 0, rst = 0	inc = 1, rst = 0	rst = 1
00	00	01	00
01	01	10	00
10	10	11	00
11	11	00	00

SEQUENTIAL CIRCUITS

- **2-bit counter, state representation**



WHAT WE DID TODAY

- digital abstraction
- binary vs hex discussion
- digital circuits
- combinational circuits
- sequential circuits
- we saw a single example of a transistor-based circuit

NEXT TIME ...

- more about basic circuits
- Boolean logic
- logic synthesis
- more complex circuits
- sequential circuits

FURTHER READING

- **PH book**
  - Appendix B

- **Computerphile, Why Use Binary?, https://www.youtube.com/watch?v=thrx3SBEpL8**

- **Real Engineering, Transistors - The Invention That Changed The World, https://www.youtube.com/watch?v=OwS9aTE2Go4**

- **Ben Eater, Making logic gates from transistors, https://www.youtube.com/watch?v=sTu3LwpF6XI**

- **DrPhysicsA, An Introduction to Logic Gates, https://www.youtube.com/watch?v=95kv5BF2Z9E**

FURTHER READING

- Computerphile, Logic gates playlist, https://www.youtube.com/playlist?list=PLzH6n4zXucko1YVRjhnquPaNOfZrymVQH
- Computerphile, Binary Addition & Overflow, https://www.youtube.com/watch?v=WN8i5cwjkSE
- Ben Eater, SR latch, https://www.youtube.com/watch?v=KM0DdEaY5sY
- Ben Eater, D latch, https://www.youtube.com/watch?v=peCh_859q7Q
- Ben Eater, D flip-flop, https://www.youtube.com/watch?v=YW-_GkUguMM
- How do computers remember?, https://www.youtube.com/watch?v=l0-izyq6q5s
- Visualizing binary data, https://www.youtube.com/watch?v=hEDQpqhY2MA