OF THE
DEPARTMENT OF
INFORMATION ENGINEERING

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Digital Systems

Analysis of Sequential Circuits

Marta Bagatin, marta.bagatin@unipd.it

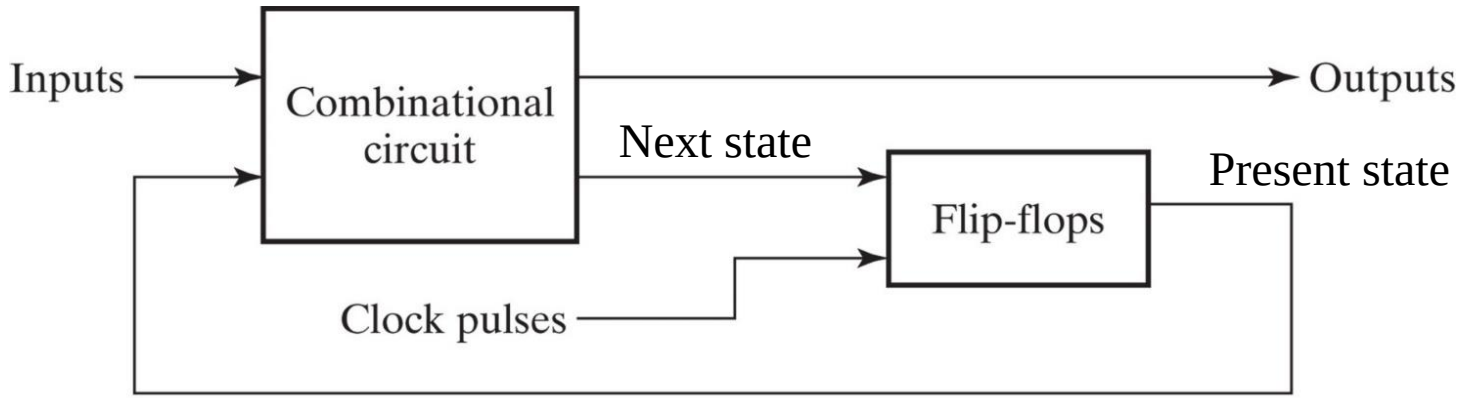
Degree Course in Information Engineering

Academic Year 2023-2024

Purpose of the Lesson

- Learn how to **analyze a sequential circuit** through the study of
  - Equations of flip-flop inputs/ equations of circuit outputs
  - Next state table
  - State diagram
  - Simulation / timing diagram
- Understand the difference between **Mealy model** and **Moore model** for sequential circuits

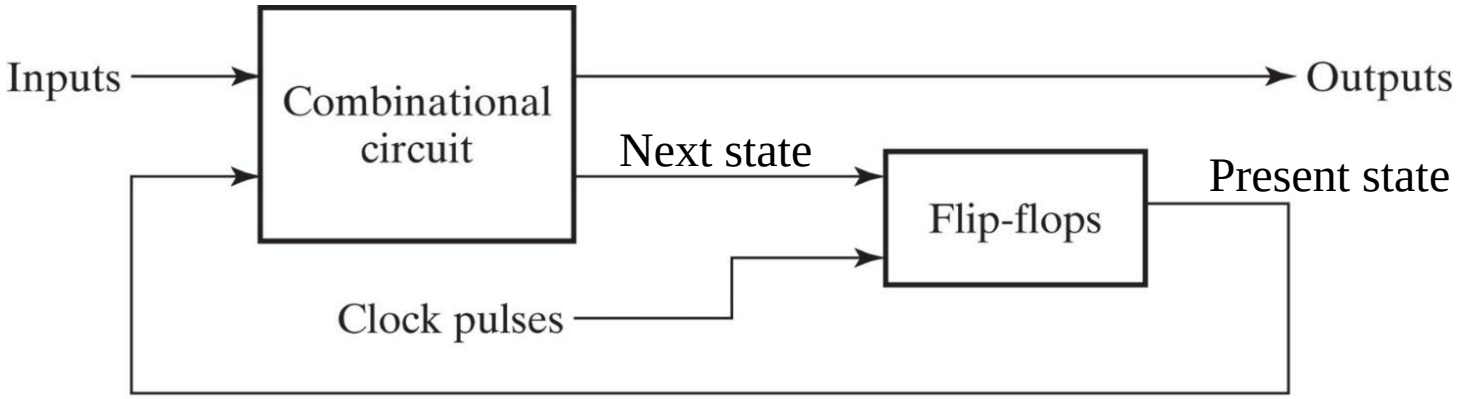
Synchronous Sequential Circuits: Summary



- In a sequential circuit the **outputs depend not only on the inputs, but also on its state**, which reflects its past history
- A sequential system contains a **feedback** path
- In a sequential circuit, the outputs and next state are calculated by a combinational block, starting from the sequential circuit inputs and present state
- A synchronous sequential circuit (or **finite state machine**) is **timed by a clock signal**, the **state variables are stored in flip-flops**, which state is **updated only in presence of specific events in the clock** (e.g. rising edges)

Analysis of a Sequential Circuit

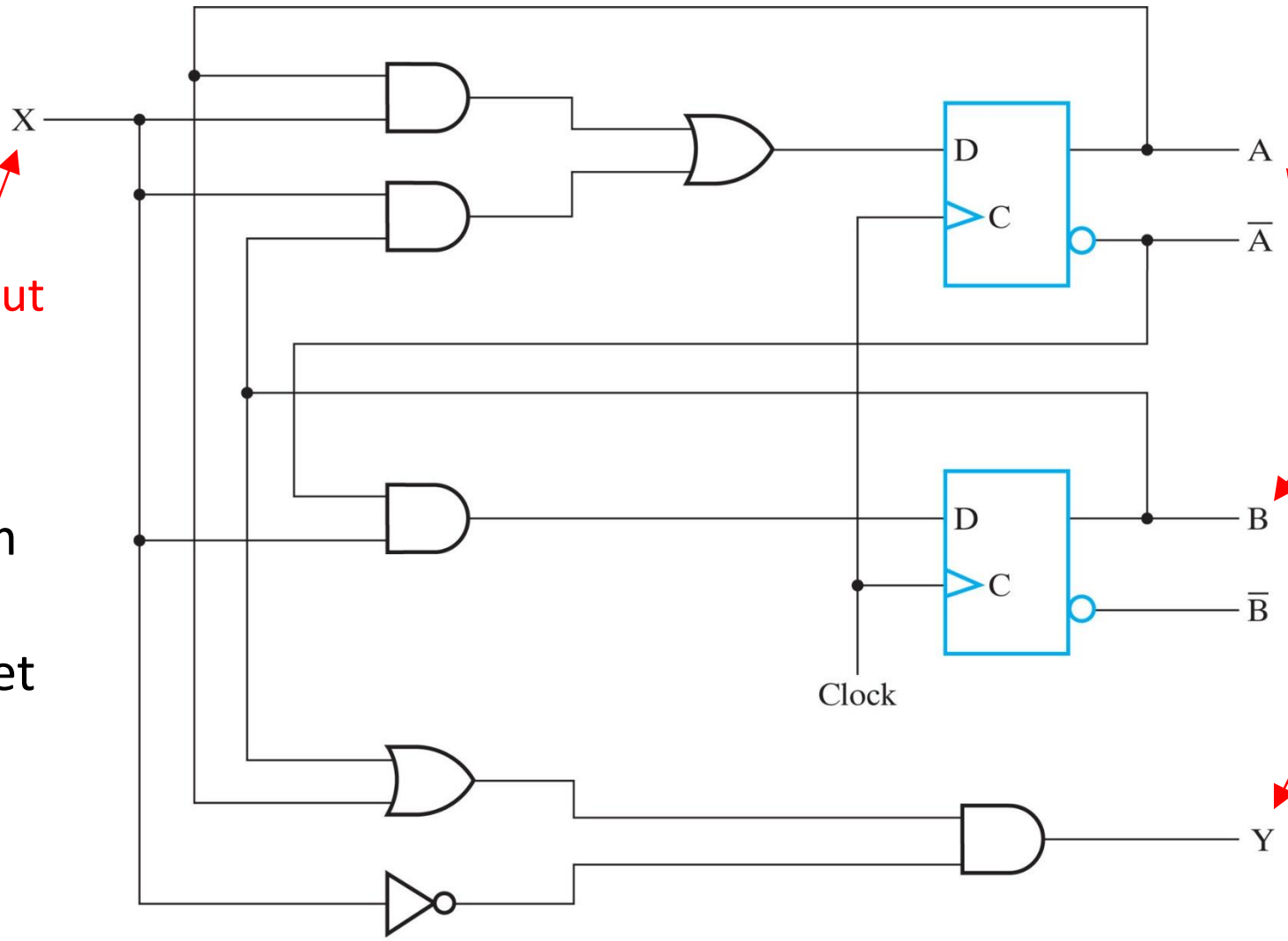
Analysis of Synchronous Sequential Circuits



- To analyze the behavior of a synchronous sequential circuit, we describe the **temporal evolution of outputs and next state, as a function of inputs and current state of the system**. In particular:
  - Flip-flop input equations: provide the next state of the system
  - Output equations: calculate the system output
  - State table: details next state and outputs for all possible combinations of input and present state
  - State diagram: represents in graphical form the information contained in the state table
  - Simulation (time diagram): verifies the correct operation of the system

Example: Analysis of a Synchronous Sequential Circuit

(Fig.4.13)



X: Input

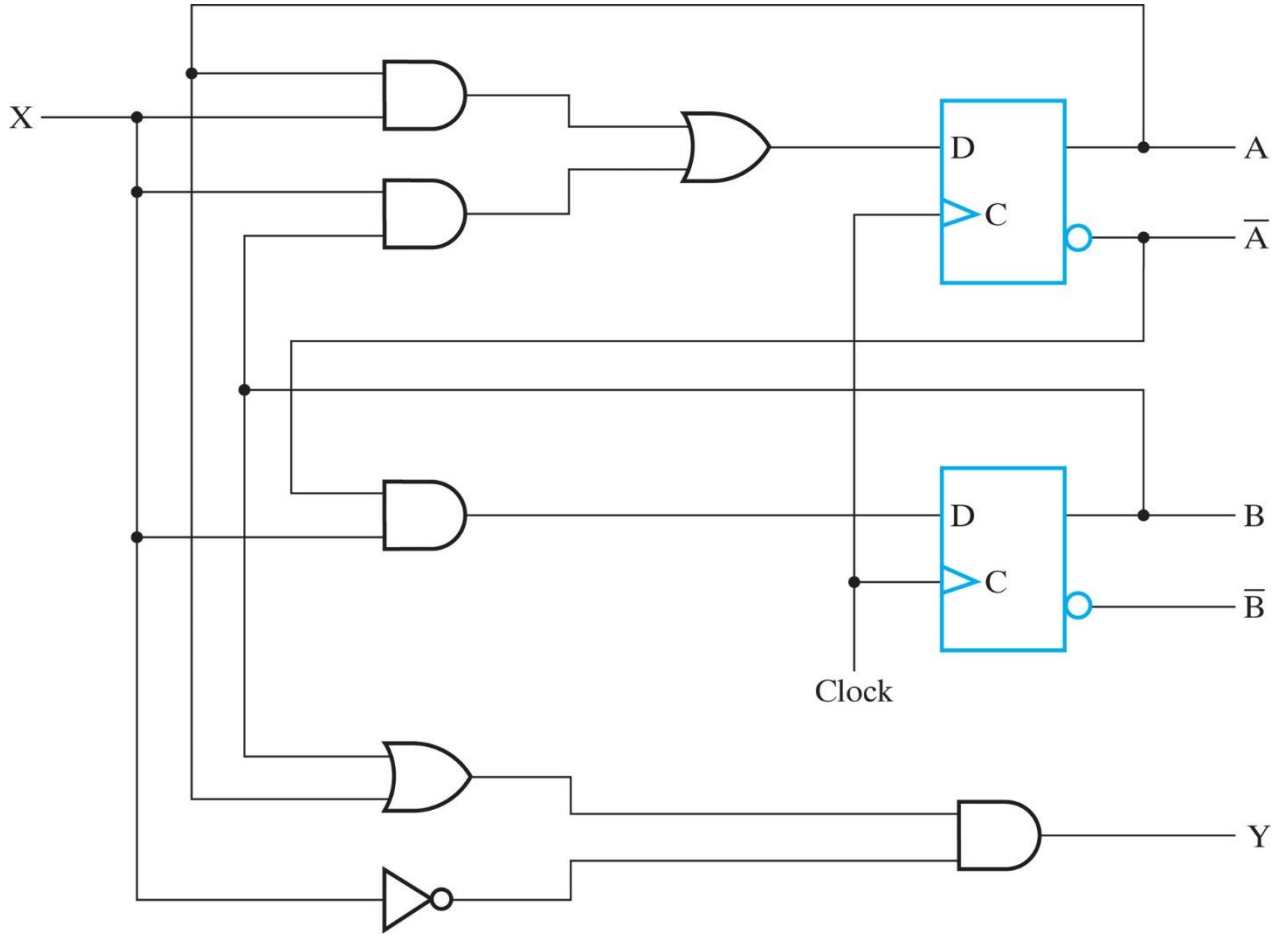
A, B: State variables (one for each FF): bits representing the state

Y: Output

The logic diagram is composed of flip-flops and a set of logic gates

Flip-flop Input Equations

- The flip-flop input equations describe the **combinational logic that generates the input signals for the flip-flops** (next state, which will become the present state at the next clock cycle)



Input equations of flip-flops:

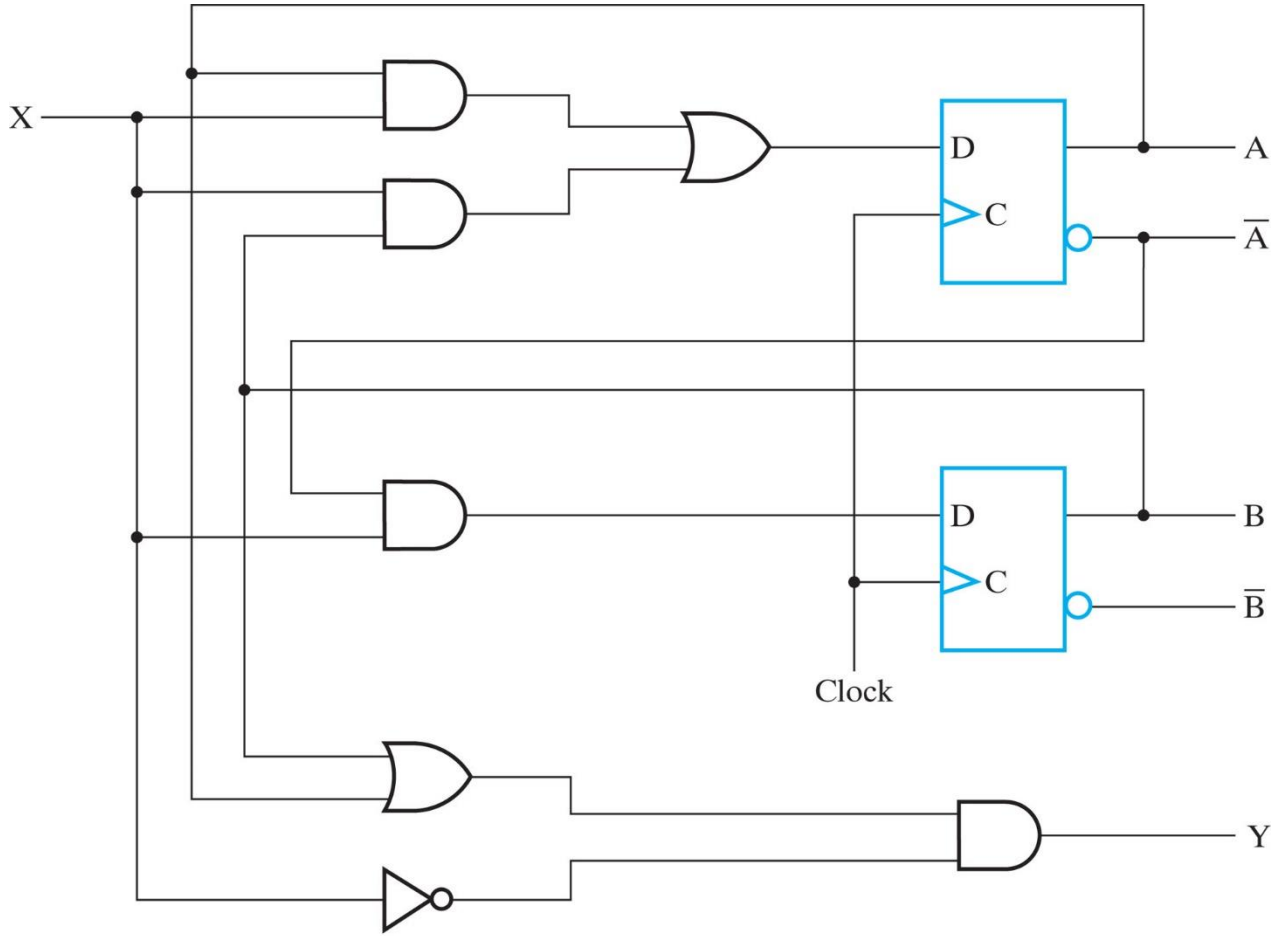
$$D_A = AX + BX$$

$$D_B = \bar{A}X$$

D indicates the type of flip-flop (D-FF), the subscript indicates the name of the state variable

Circuit Output Equation

- The circuit output equation describes **the combinational logic that generates the output of the sequential circuit**



Circuit output equation:

$$Y = (A + B)\,\bar{X}$$

State Table

Time t			Time t+1		Time t
Present State		**Input**	**Next State**		**Output**
A	**B**	**X**	**A**	**B**	**Y**
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

- The state table describes the **state transitions**: for each combination of present state and input (time t), the table lists the next state (state at time t + 1, i.e. the flip-flops input, which will become their output at the next clock cycle)
- The table also lists the **outputs**, for each combination of present state and input (at time t, i.e. at the current clock cycle)

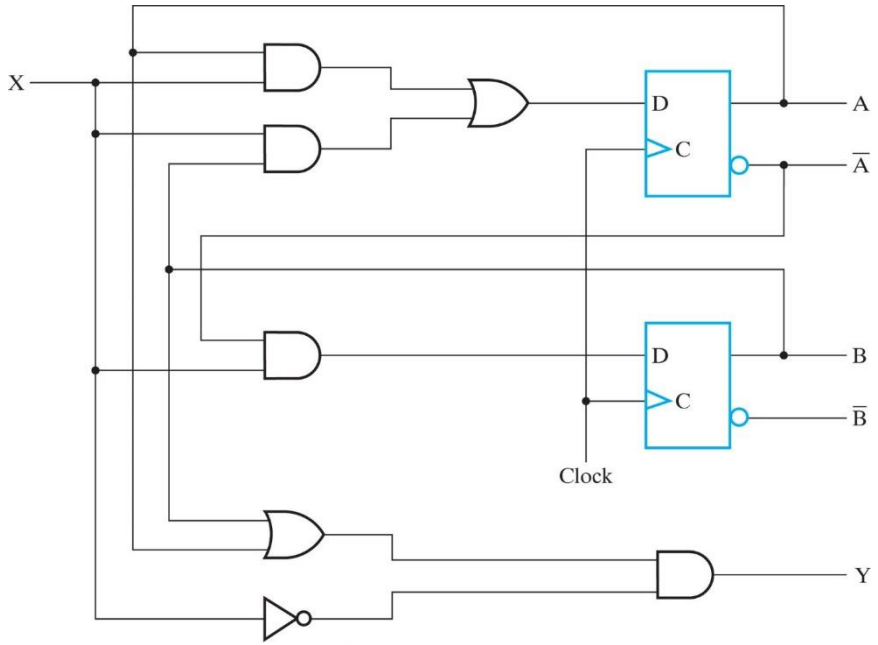
State Table

Time t			Time t+1		Time t
Present State		**Input**	**Next State**		**Output**
A	**B**	**X**	**A**	**B**	**Y**
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

For each value of the inputs, all possible combinations of states are listed: for a circuit with m flip-flops (i.e. with m state variables) and n inputs, the state table has 2^{m+n} rows

State Table: Determination

Present State		Input	Next State		Output
A	B	X	A	B	Y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0



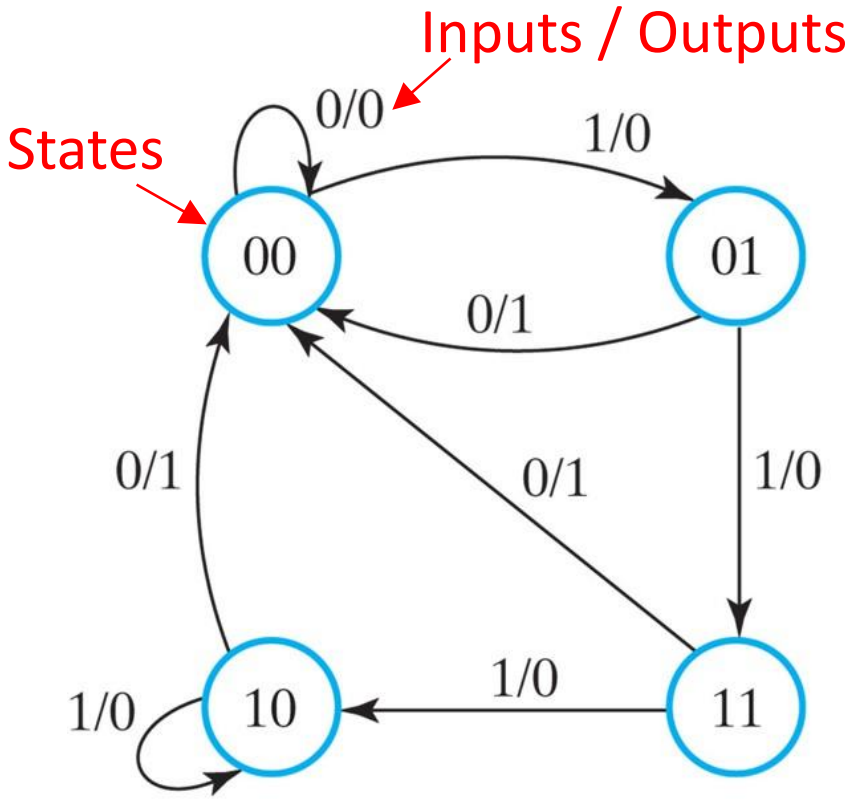
- Build a table with all the combinations of present states and inputs
- For each row, determine the future states based on the logic diagram or the input equations of the flip-flops [e.g. for a D-FF: $A(t+1) = D_A(t)$]
- For each row, determine the outputs, based on the logic diagram or system output equation

Two-Dimensional State Table

Present State		Next State				Output	
		X = 0		X = 1		X = 0	X = 1
A	B	A	B	A	B	Y	Y
0	0	0	0	0	1	0	0
0	1	0	0	1	1	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0

- More compact way to describe the state transitions: each state is associated with a single row (with input values indicated in different columns), listing the next state and the output
- Information is the same as the state table

State Diagram

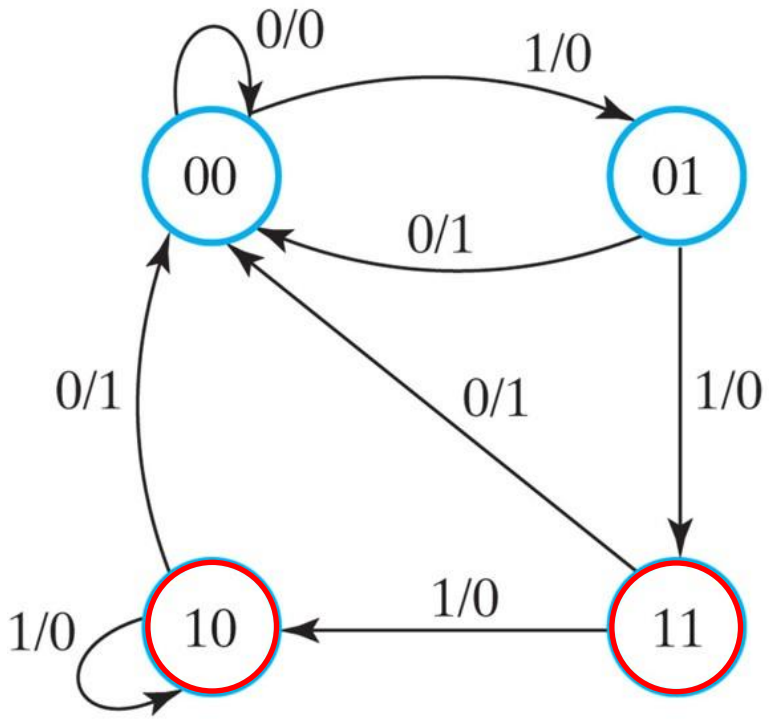


- **States** of the system are represented by circles (the name or the encoding of the states is specified inside the circles)
- **Transitions between states** are indicated by arrows from one state to another
- The inputs and outputs corresponding to each transition are indicated close to the arrow and separated by a slash

- The state diagram contains the same information as the **state table**, but in this case information is represented **graphically**
- Looking at the state diagram, the operation of the circuit can be interpreted in a more intuitive way compared to the state table

Equivalent States

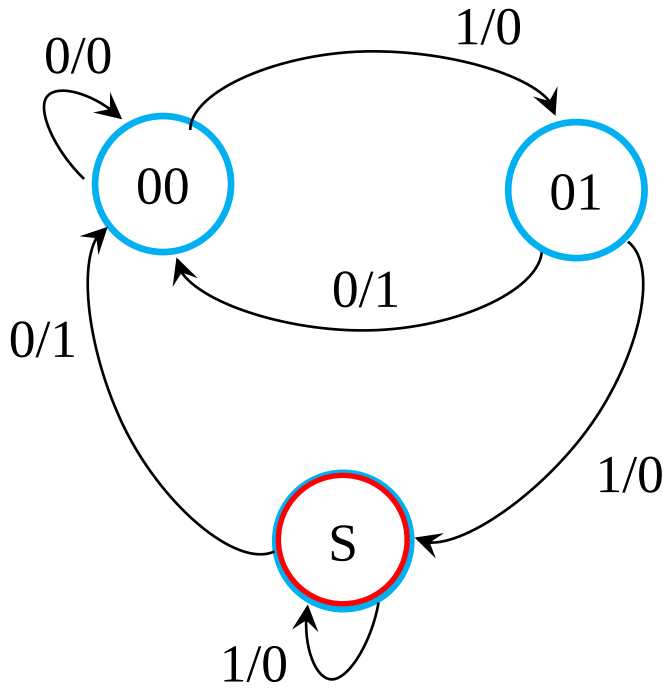
- Two states are equivalent if, starting from both states, the **application of all possible inputs produces the same outputs and leads to the same (or equivalent) next state**



  - The states '11' and '10' are equivalent
    - If input '1' is applied, both states produce the same output ('0') and the system transitions to the same next state ('10')
    - If input '0' is applied, both states produce the same output ('1') and the system transitions to the same next state ('00')

Equivalent States

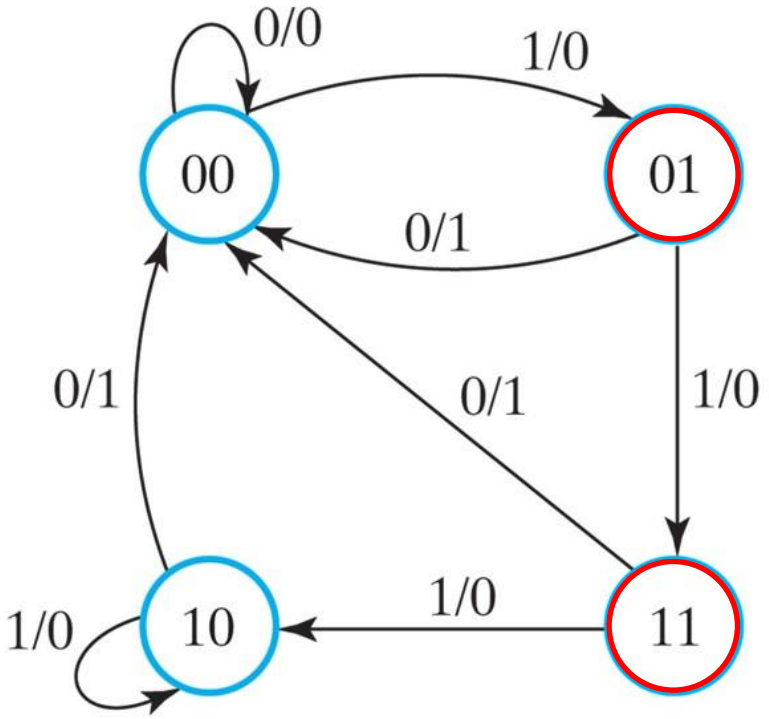
- Two states are equivalent if, starting from both states, the **application of all possible inputs produces the same outputs and leads to the same (or equivalent) next state**



- If we merge the two equivalent states to a single state S, we obtain an **equivalent circuit** (simplified compared to the previous circuit)
- Note: the equivalence of the two states can also be seen from the state table

Equivalent States

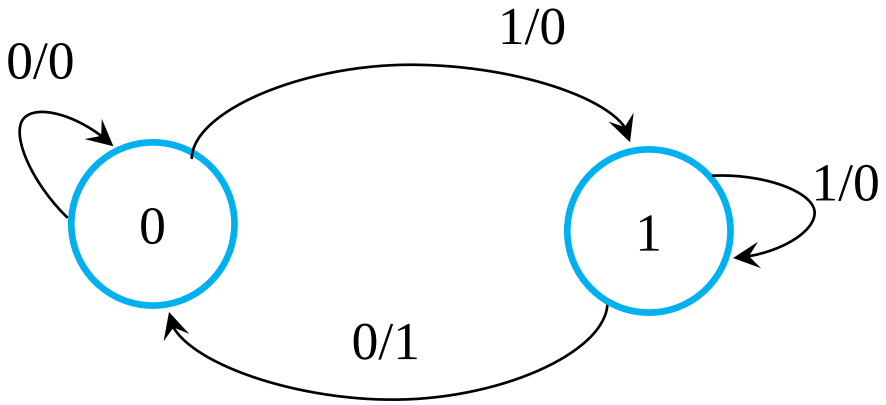
- Two states are equivalent if, starting from both states, the **application of all possible inputs produces the same outputs and leads to the same (or equivalent) next state**



- Let’s go back to the initial state diagram
- The states '01' and '11' are also equivalent
  - If input '1' is applied from states '01' and '11', the next states are respectively '11' and '10', (which, we have seen, are equivalent) and output '0' is produced in both cases
  - If input '0' is applied, the next state is '00' and the output '1' is produced in both cases

Equivalent States

- As a result, the 3 equivalent states can be merged into a single state, and the circuit functionality does not change



  - An equivalent state diagram includes 2 states (= 1 flip-flop), instead of the initial diagram with 4 states (= 2 flip-flops)

- The identification of equivalent states can be **useful to reduce the number of states in a sequential circuit** (hence the number of flip-flops)
- However, this simplification may not necessarily lead to a lower cost (the cost of the sequential circuit also depends on the combinational part, in addition to the flip-flops)

Mealy and Moore Models

Mealy and Moore Model Circuits

- In a **Mealy** model circuit, the **outputs depend both on the current state and the inputs** of the system
  - The example we have just seen is a Mealy type circuit
- In a **Moore** model circuit, the **outputs depend ONLY on the current state** of the system (and not on the inputs)
  - In the state diagram, the outputs are indicated inside each state, rather than next to each transition
- A Mealy model circuit, in general, compared to a Moore type circuit
  - Has fewer states
  - Contains less logic and is faster (reacts faster to inputs, in the same clock cycle)

Example: Moore Model Circuit

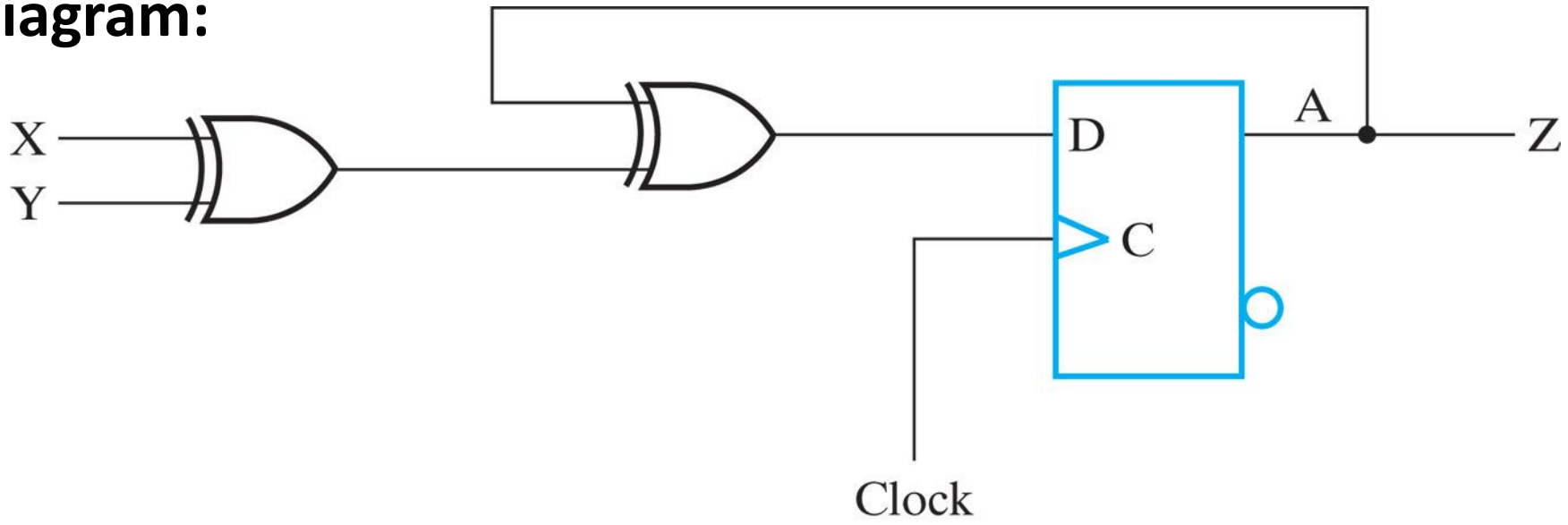
- Find the logic diagram, state table, and state diagram for the circuit specified by the input equation:

$$D_A = A \oplus X \oplus Y$$

and the output equation:

$$Z = A$$

Logic diagram:



Example: Moore Model Circuit

- Find the logic diagram, state table, and state diagram for the circuit specified by the input equation:

$$D_A = A \oplus X \oplus Y$$

and the output equation:

$$Z = A$$

State table

With m=1 state variable (= 1 FF) and n=2 inputs, the table has $2^{m+n} = 8$ rows

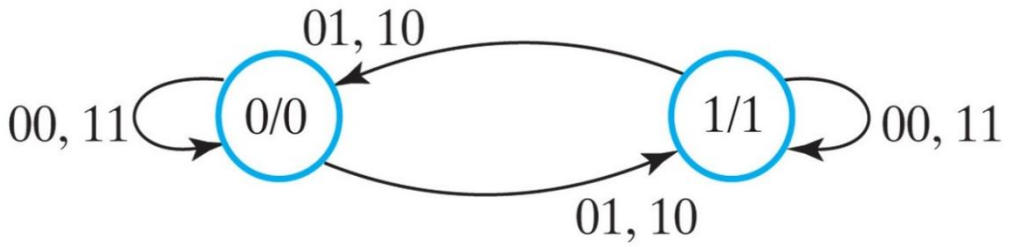
Present State	Inputs		Next State	Output
A	X	Y	A	Z
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Example: Moore Model Circuit

State table:

Present State	Inputs		Next State	Output
A	X	Y	A	Z
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

State diagram:



- The output is indicated inside each state (does not depend on the input: Moore model) rather than next to the transition arrows
- Each transition is represented by an arrow from one state to another one
- There must be one transition for each input. Multiple input values can be indicated on a single arrow, separated by a comma

Simulation of a Sequential Circuit

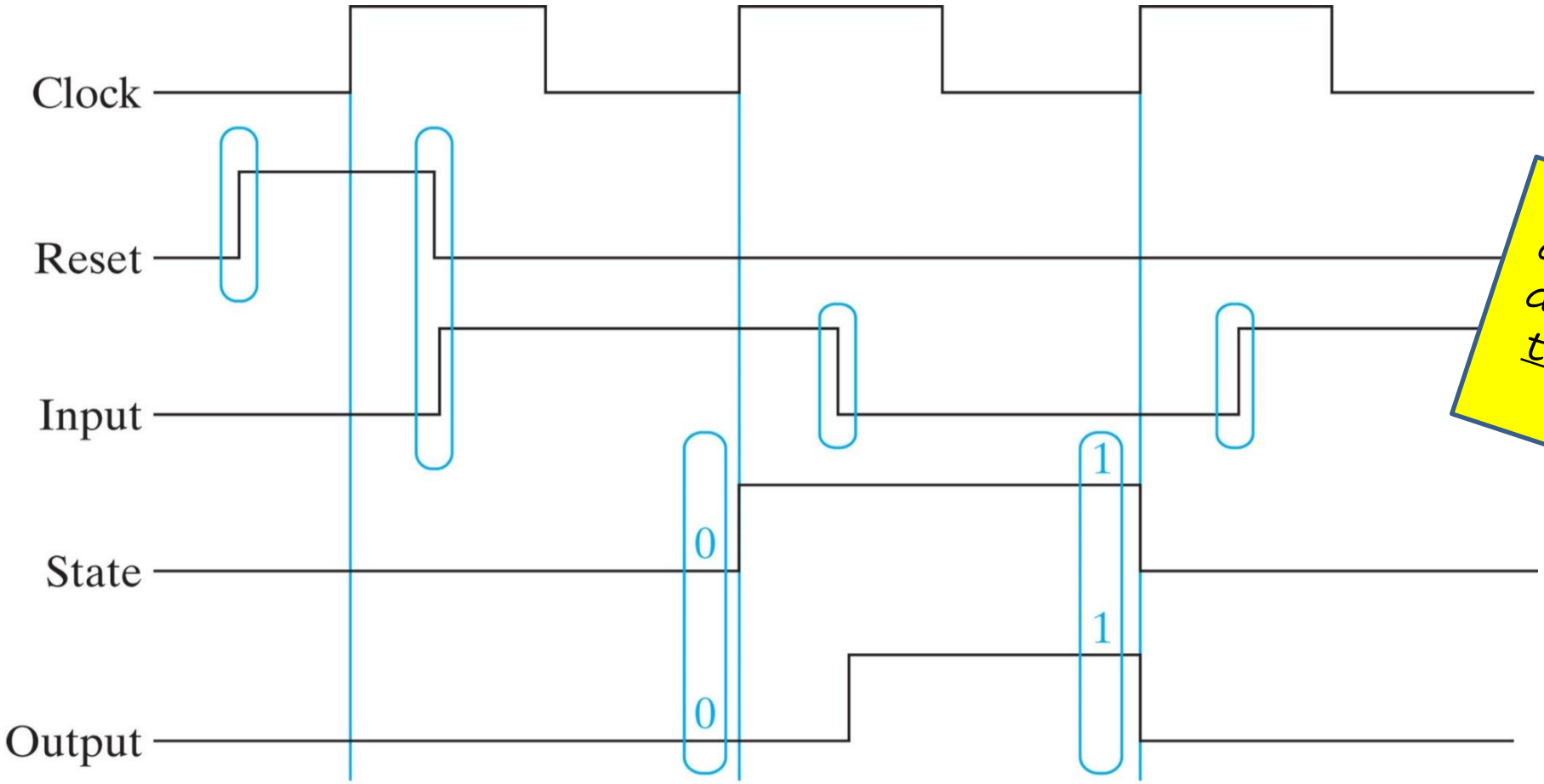
Simulation of a Sequential Circuit

- There are some issues that were not present in combinational circuits
  i. The behavior of the circuit **depends on the temporal order in which the inputs are applied**, also in relation to the variations of the clock
  ii. We need to start from a scenario where the circuit is in a known state, therefore it is necessary to initialize the circuit (provide a **reset signal**) at the beginning of the simulation
  iii. To check the behavior of the circuit we **apply specific input sequences**, correctly timed with the clock and **observe the outputs at proper instants with respect to the clock**
- The simulation of a sequential circuit can be
  - **Functional simulation** (the elements of the circuit are ideal, with no or very little delay that can be neglected): used to check the correct functionality of the circuit
  - **Timing simulation** (circuit elements have realistic delay times): used to check the correct operation of the circuit in terms of timing

Simulation of a Sequential Circuit

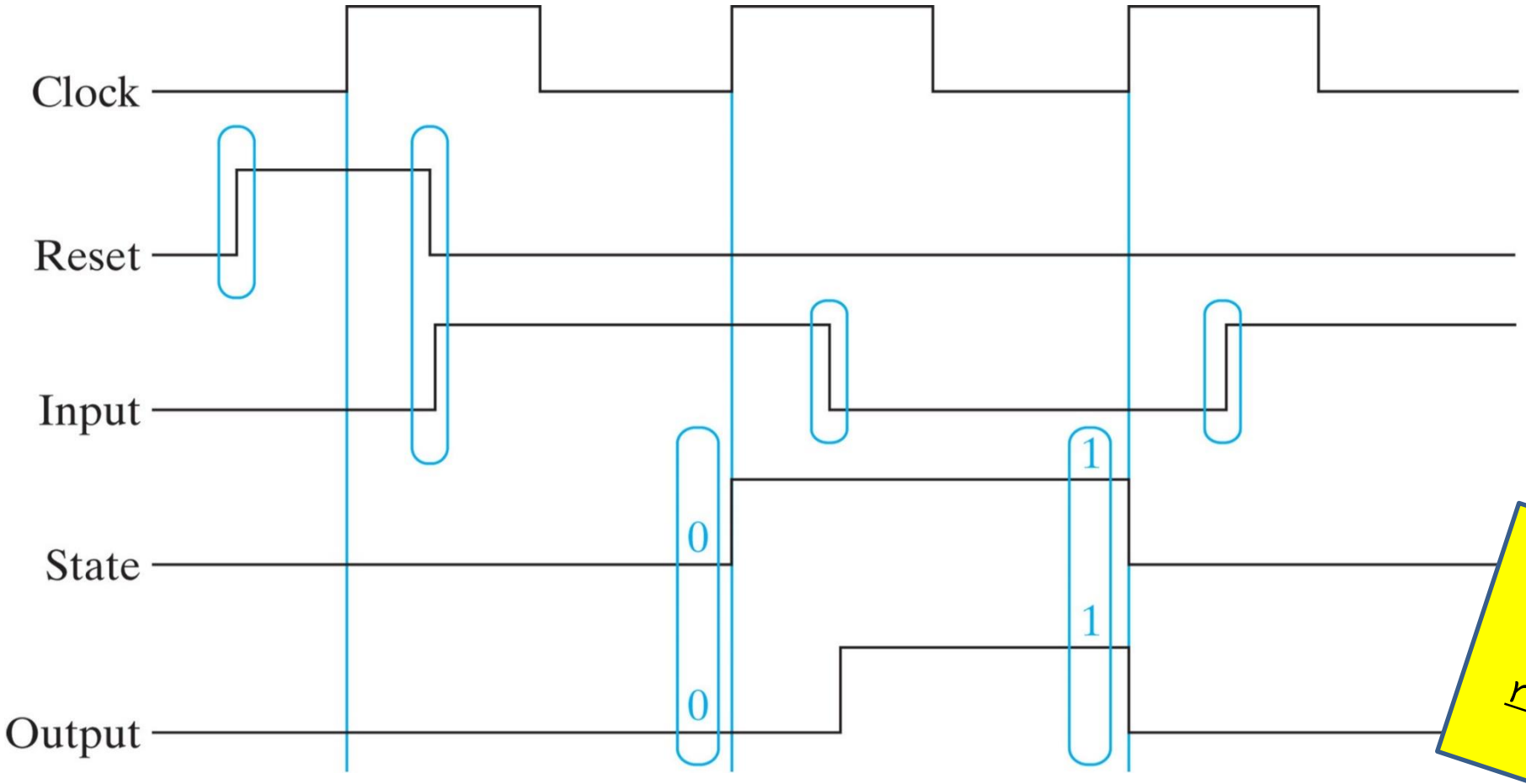
- For a correct simulation, the following points are important:
  - If small delay times are used (also in functional simulations), make sure that the **clock period is long enough** to exceed the worst-case delay of the combinational logic
  - **Apply the inputs earlier enough with respect to the active edge of the clock** (in case of positive-edge triggered flip-flop, e.g. at the falling edge): in this way, the combinational logic has the time to calculate the outputs and next state before the next clock rising edge
  - **Examine the simulation results when the state variables and outputs have reached their final values**, e.g. just before the rising edge of the clock in case of positive-edge triggered flip-flop

Example of Time Diagram (with Positive-edge-triggered Flip-flops)



Changes in RESET and INPUT must be applied well before the rising edge of the Clock

Example of Time Diagram (with Positive-edge-triggered Flip-flops)



States and outputs must be observed just before the rising edge of the Clock

Disclaimer

Figures from*Logic and Computer Design Fundamentals*, Fifth Edition, GE Mano |Kime| Martin