Digital Logic Design + Computer Architecture

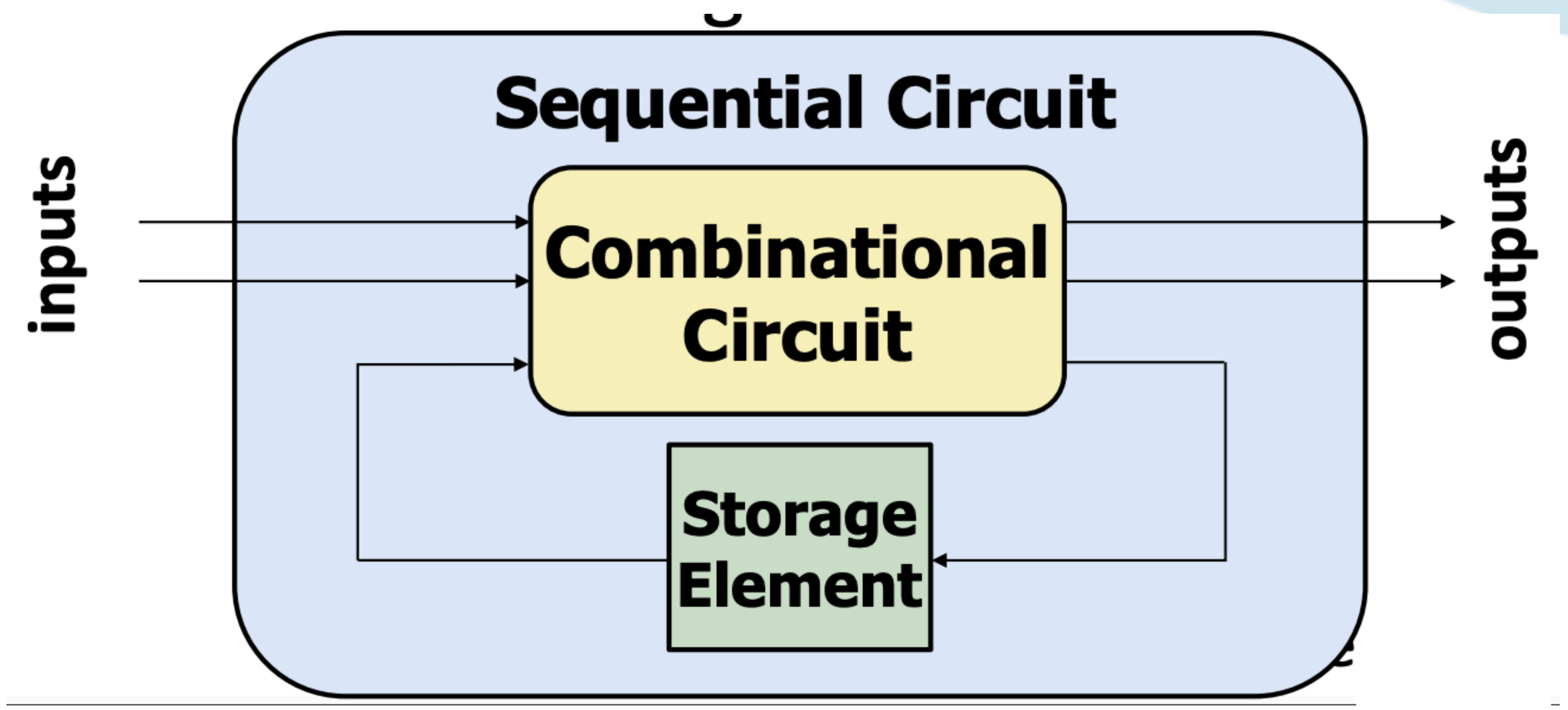
Sayandeep Saha

Assistant Professor
Department of Computer Science and Engineering
Indian Institute of Technology Bombay

Sequential Circuits

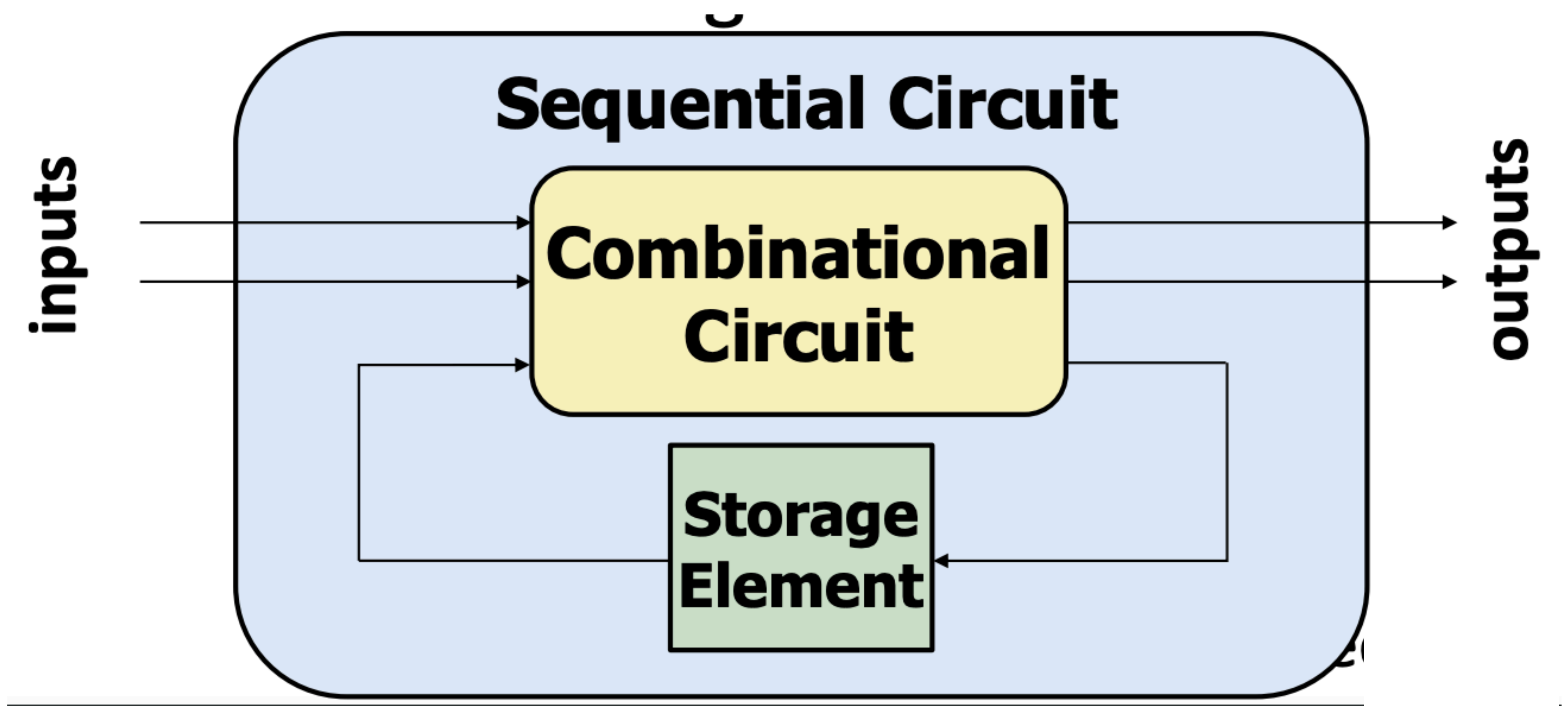
A Circuit that Remembers

- **How do you remember things?**
  - Memory
- **Can we design a circuit which remembers?**
  - A formal way to model this capability is called a **state**
  - **So we will be modelling circuits to create a state.**



A Circuit that Remembers

- **Every digital logic you see in real life is sequential**
  - Your processors — that you going to see in the rest of the course
  - Your washing machine — it remembers your setting and washes accordingly
  - Your elevator — it remembers which floors to stop
  - Your ATM machine — it remembers your choice and updates your account after despatching money

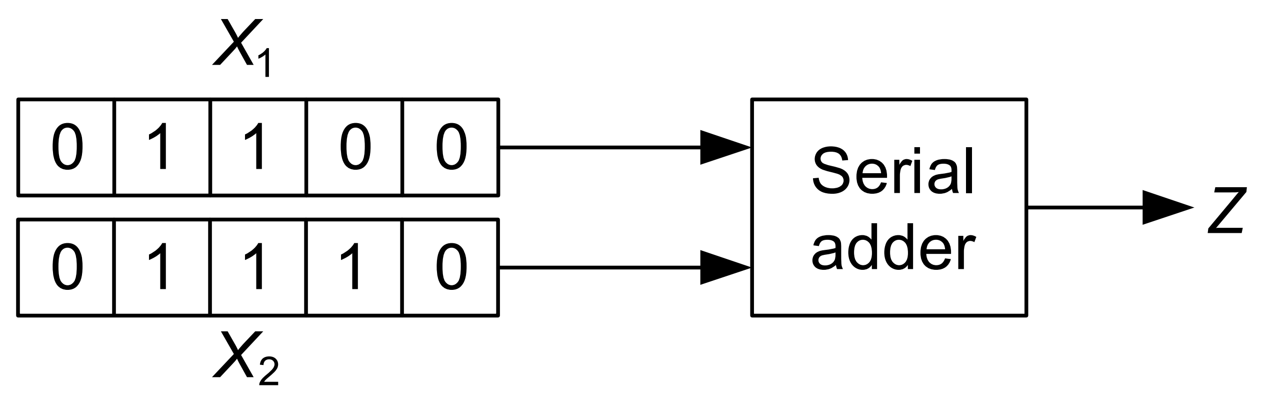


Sequential Circuits and Finite State Machines

Sequential circuit: its outputs a function of external inputs as well as stored information (aka. State)

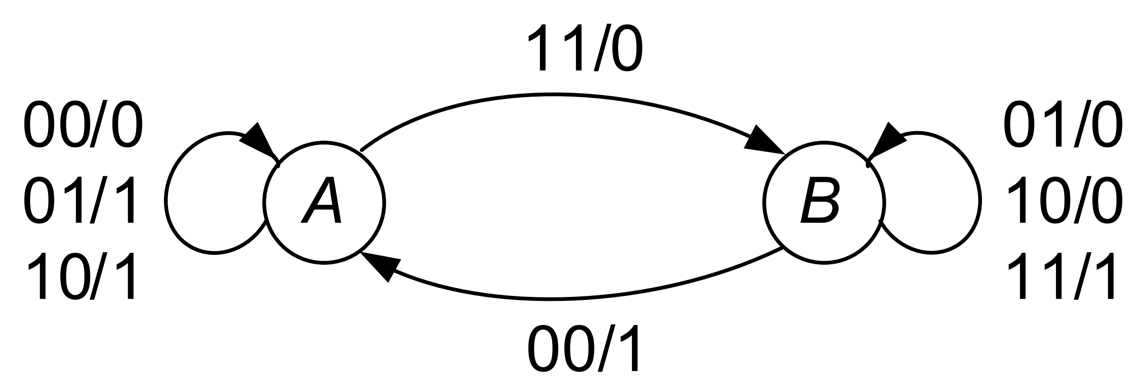
Finite-state machine (FSM): abstract model to describe the synchronous sequential machines. It has finite memory.

Serial binary adder example: block diagram, addition process, <u>state table</u> and <u>state diagram</u>
Let A denote the state of the adder at t_i if the carry 0 is generated at t_{i-1}
Let B denote the state of the adder at t_i if the carry 1 is generated at t_{i-1}



$$\begin{array}{rccccccl} & t_5 & t_4 & t_3 & t_2 & t_1 & & \\ & 0 & 1 & 1 & 0 & 0 & = & X_1 \\ + & 0 & 1 & 1 & 1 & 0 & = & X_2 \\ \hline & 1 & 1 & 0 & 1 & 0 & = & Z \end{array}$$

	NS, z			
PS	$x_1x_2 = 00$	01	11	10
A	$A, 0$	$A, 1$	$B, 0$	$A, 1$
B	$A, 1$	$B, 0$	$B, 1$	$B, 0$



Sequential Circuits and Finite State Machines

Two states capable of storing information regarding the **presence or absence of carry**:

delay element with input Y and output y

- Two states: $y = 0$ and $y = 1$
- The capability of the device to store information is the result of the fact that it takes some time for the input signal Y to pass to the output y
  - Compare with a combinational gate where the output changes almost immediately. In this device there is some well-defined time required before the input Y passes to the output y. **During that time-window, the old value stays!**
- Since the **present input value Y** of the delay element is equal to its **next output value**: the input value is referred to as the next state of the delay
  - $y(t+1)=Y(t)$

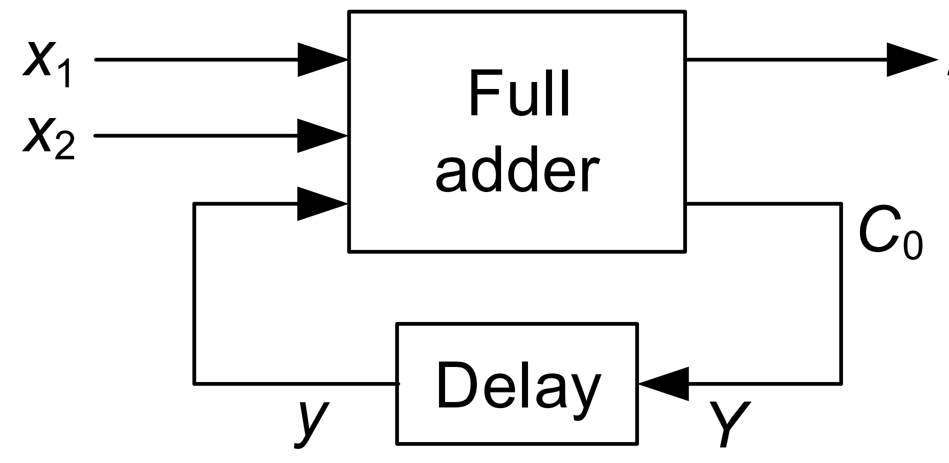
Example: assign state $y = 0$ to state A of the adder and $y = 1$ to B

- The value of y at t_i corresponds to the value of the carry generated at t_{i-1}
- Process of assigning the states of a physical device to the states of the serial adder: called **state assignment**
- Output value y: referred to as the **state variable**
- **Transition/output table** for the serial adder:

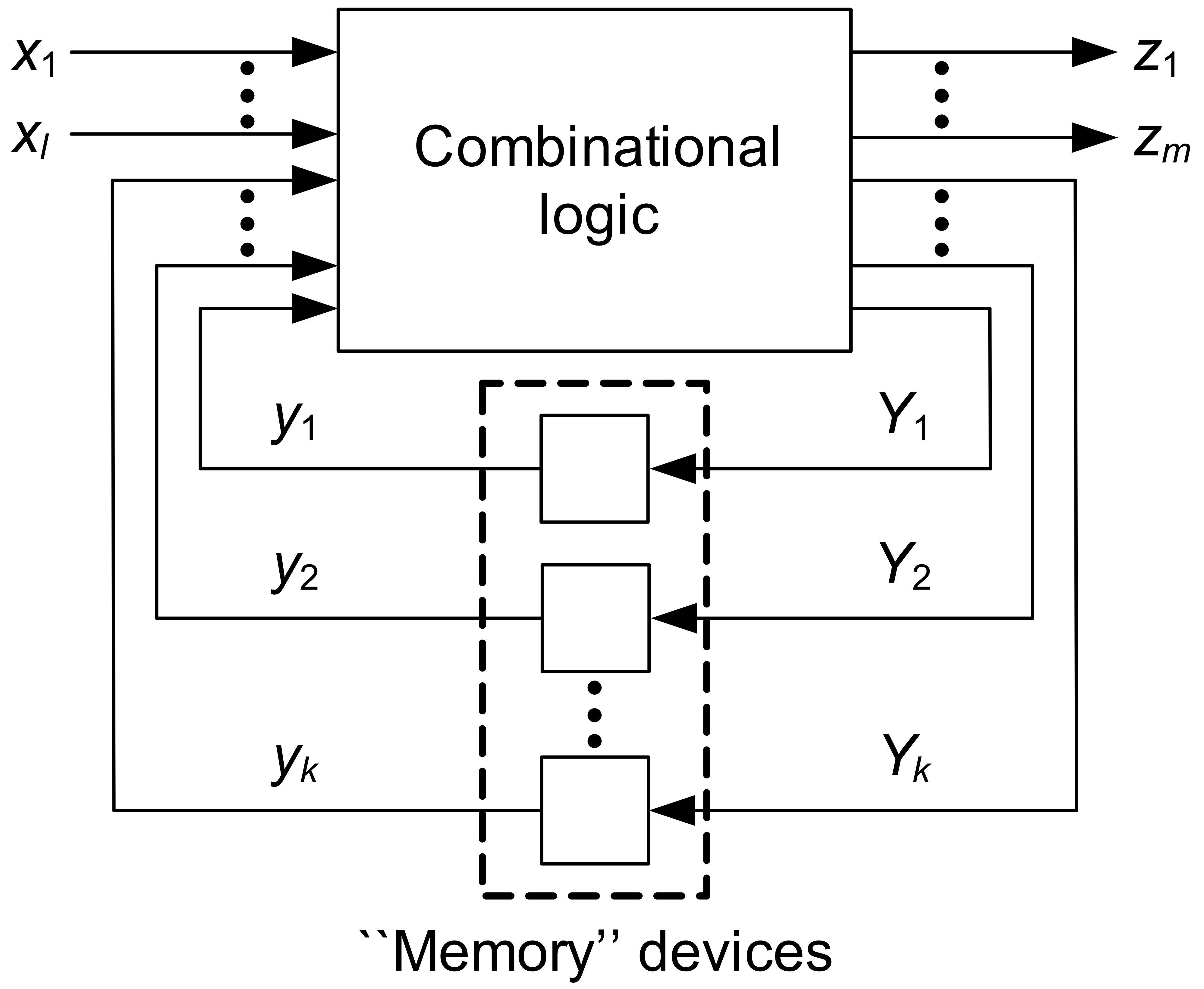
y	*Next state* Y				*Output* z			
	x_1x_2				x_1x_2			
	00	01	11	10	00	01	11	10
0	0	0	1	0	0	1	0	1
1	0	1	1	1	1	0	1	0

$$Y = x_1x_2 + x_1y + x_2y$$

$$z = x_1 \oplus x_2 \oplus y$$



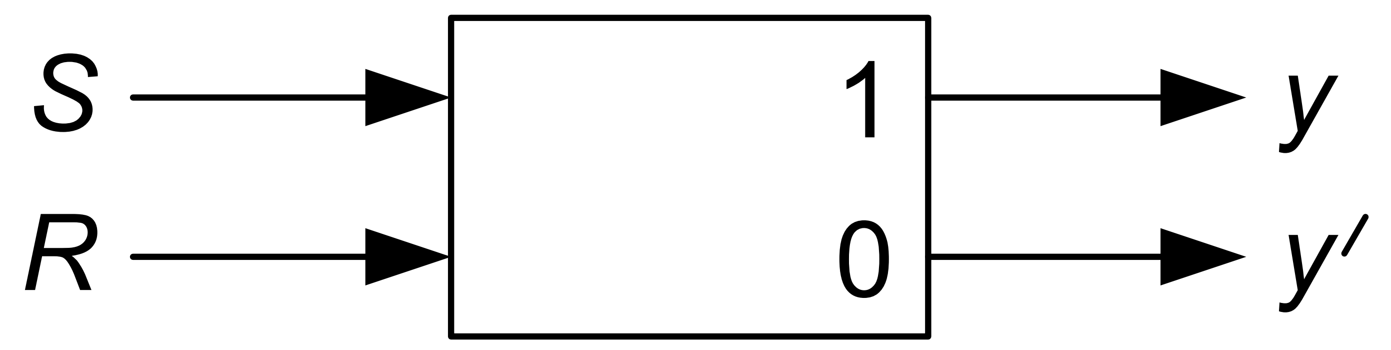
Sequential Circuits and Finite State Machines



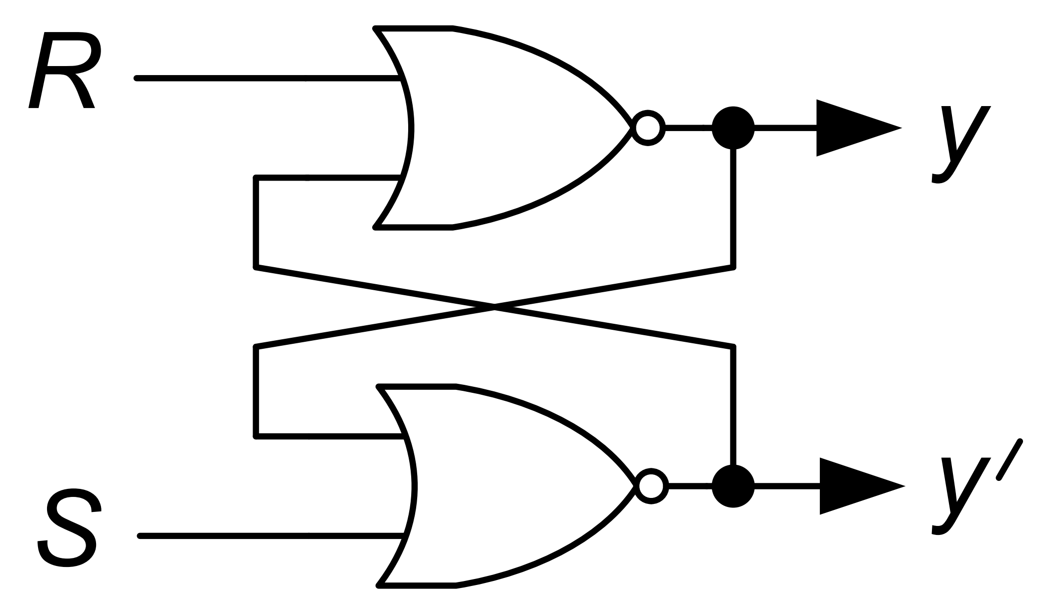
Memory Element: Latches

Latch: remains in one state indefinitely until an input signals directs it to do otherwise

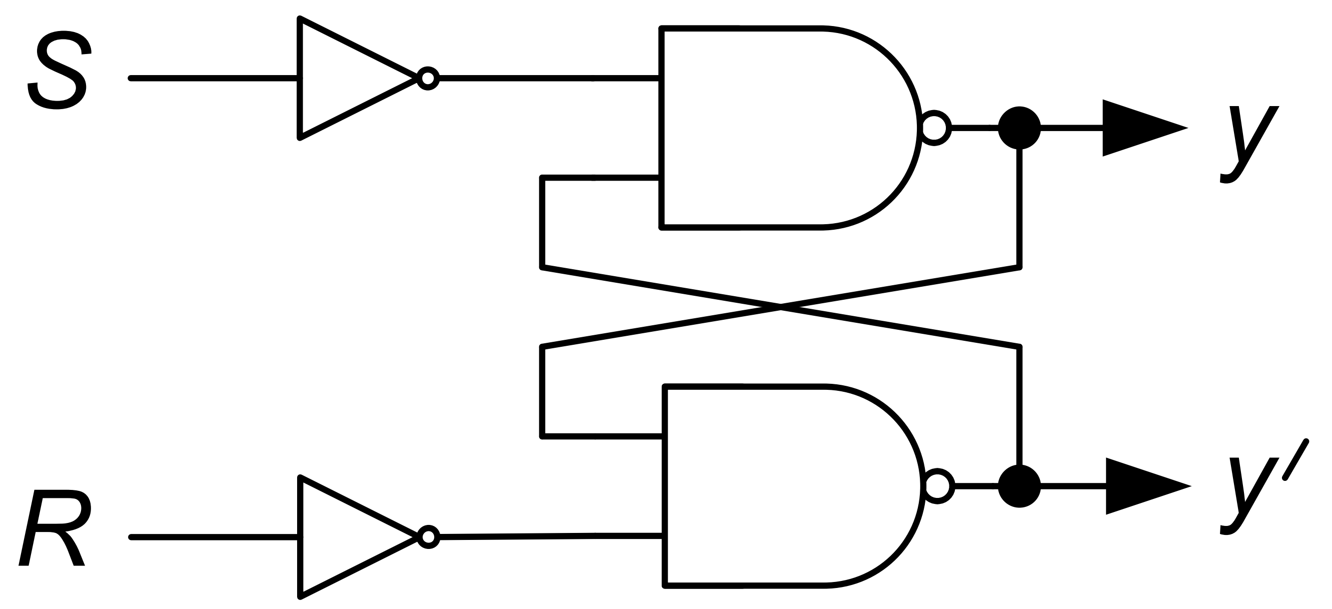
Set-reset of *SR* latch:



(*a*) Block diagram.



(*b*) NOR latch.



(*c*) NAND latch.

Memory Element: Latches

Characteristic table and excitation requirements:

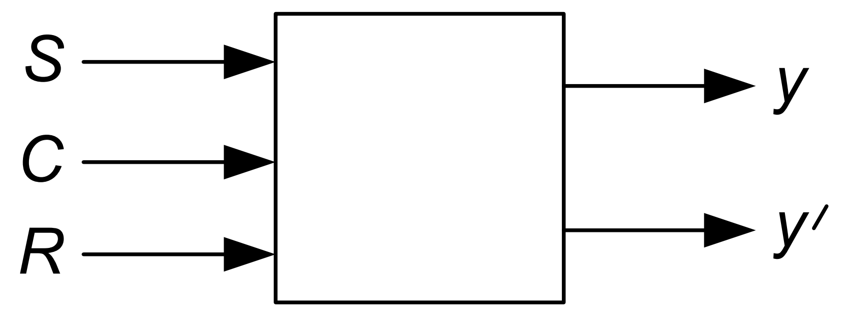
$y(t)$	$S(t)$	$R(t)$	$y(t+1)$
0	0	0	0
0	0	1	0
0	1	1	?
0	1	0	1
1	1	0	1
1	1	1	?
1	0	1	0
1	0	0	1

Circuit change		*Required value*	
From:	*To:*	S	R
0	0	0	–
0	1	1	0
1	0	0	1
1	1	–	0

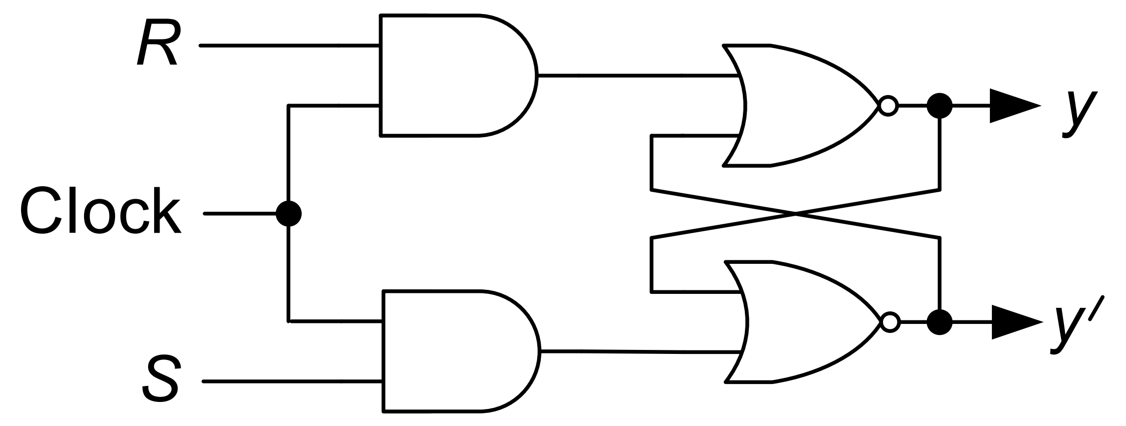
$$\begin{aligned} RS &= 0 \\ y(t+1) &= R'y(t) + S \end{aligned}$$

Clocked *SR* latch: all state changes synchronized to clock pulses

- Restrictions placed on the length and frequency of clock pulses: so that the circuit changes state no more than once for each clock pulse



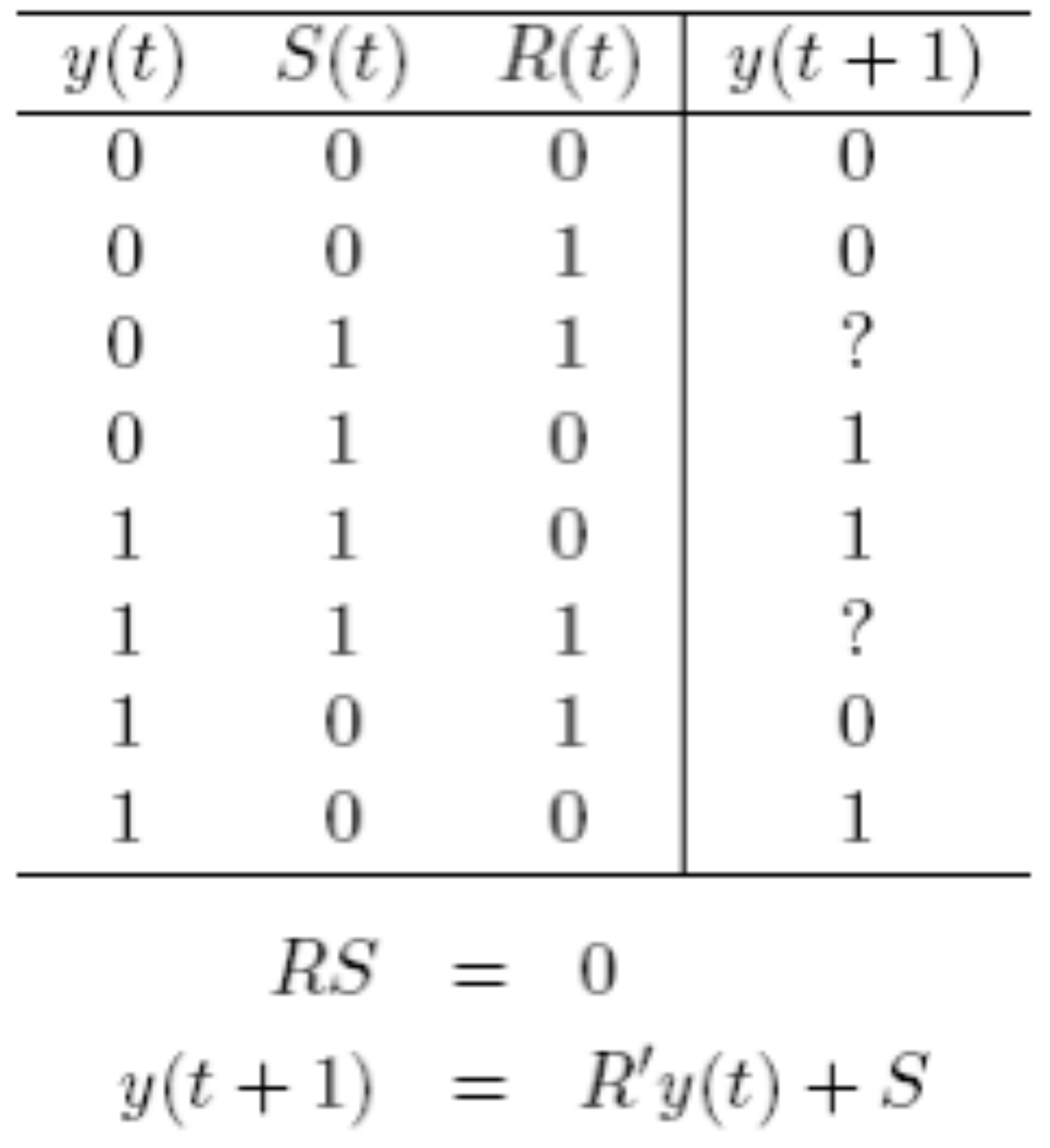
(*a*) Block diagram.



(*b*) Logic diagram.

Memory Element: Latches

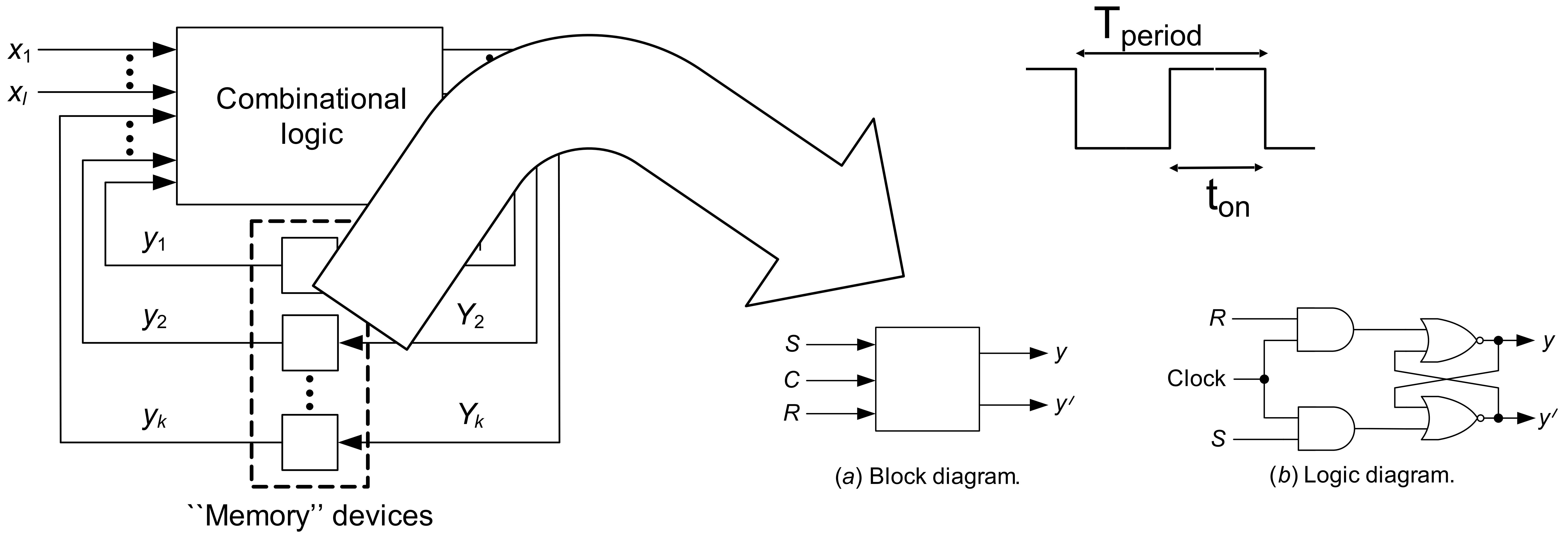
Why is the (1,1) input forbidden?



$y(t)$	$S(t)$	$R(t)$	$y(t+1)$
0	0	0	0
0	0	1	0
0	1	1	?
0	1	0	1
1	1	0	1
1	1	1	?
1	0	1	0
1	0	0	1

$$\begin{aligned} RS &= 0 \\ y(t+1) &= R'y(t) + S \end{aligned}$$

Memory Element: Latches



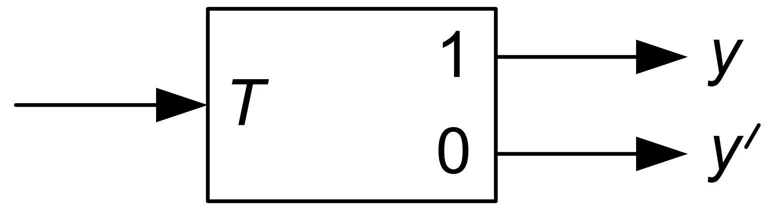
(*a*) Block diagram.

(*b*) Logic diagram.

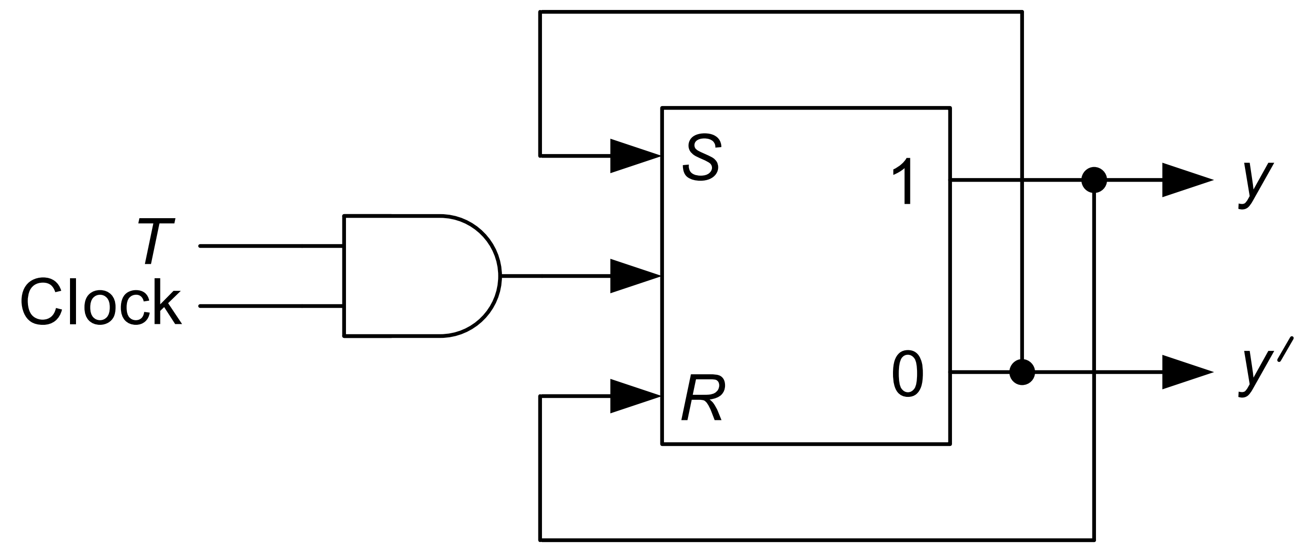
- A **clock** is a periodic signal that is used to keep time in sequential circuits.
- **Duty Cycle** is the ration of t_{on}/T_{period}
- We want to keep t_{on} small so that in the same clock pulse only a single computation is performed.
- We want to keep T_{period} sufficient so that there is enough time for the next input to be computed.

Memory Element: T Latch

Value 1 applied to its input triggers the latch to change state



(*a*) Block diagram.



(*b*) Deriving the T latch from the clocked SR latch.

Excitations requirements:

Circuit change		*Required*
From:	*To:*	*value T*
0	0	0
0	1	1
1	0	1
1	1	0

"Q" is basically "y"

Characteristic Table

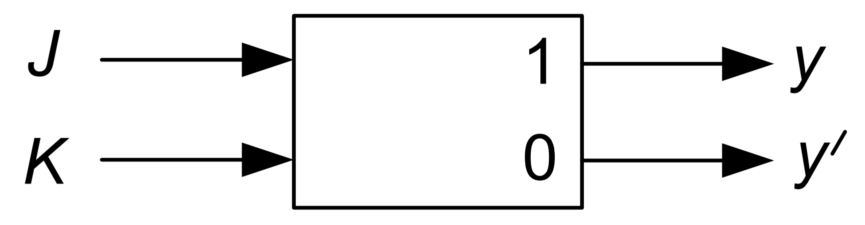
***T* Flip-Flop**

T	**Q(*t* + 1)**	
0	$Q(t)$	No change
1	$Q'(t)$	Complement

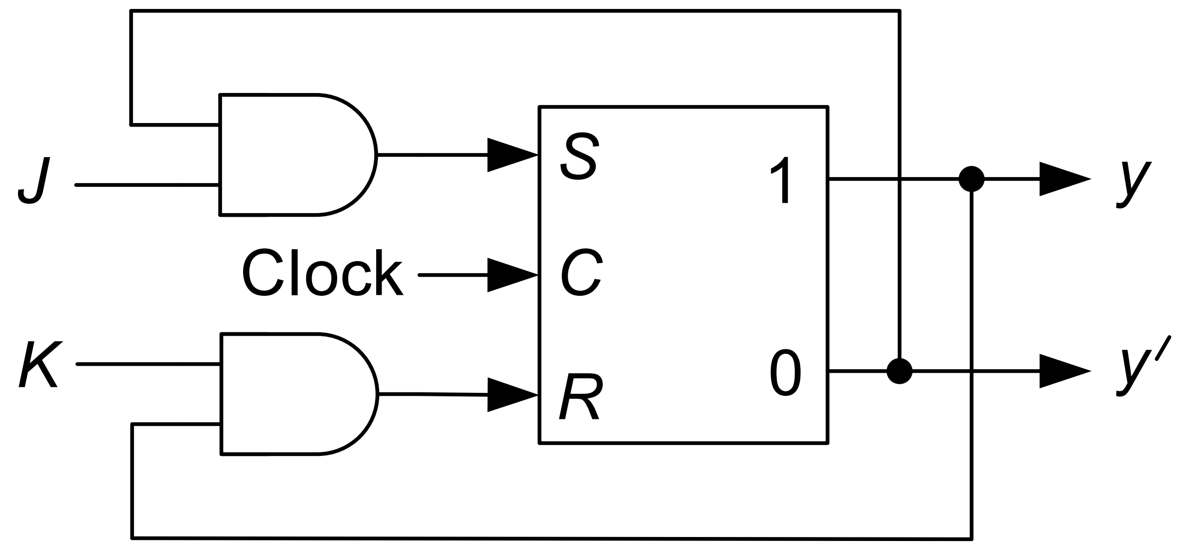
$$y(t+1) = Ty'(t) + T'y(t)$$
$$= T \oplus y(t)$$

Memory Element: JK Latch

Unlike the *SR* latch, *J* = *K* = 1 is permitted: when it occurs, the latch acts like a trigger and switches to the complement state



(*a*) Block diagram.



(*b*) Constructing the JK latch from the clocked SR latch.

Excitation requirements:

Circuit change		*Required value*	
From:	*To:*	*J*	*K*
0	0	0	–
0	1	1	–
1	0	–	1
1	1	–	0

"Q" is basically "y"

Characteristic Table

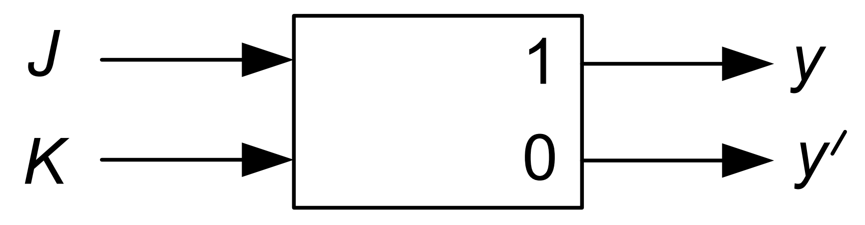
***JK* Flip-Flop**

J	***K***	**Q(*t* + 1)**	
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$Q'(t)$	Complement

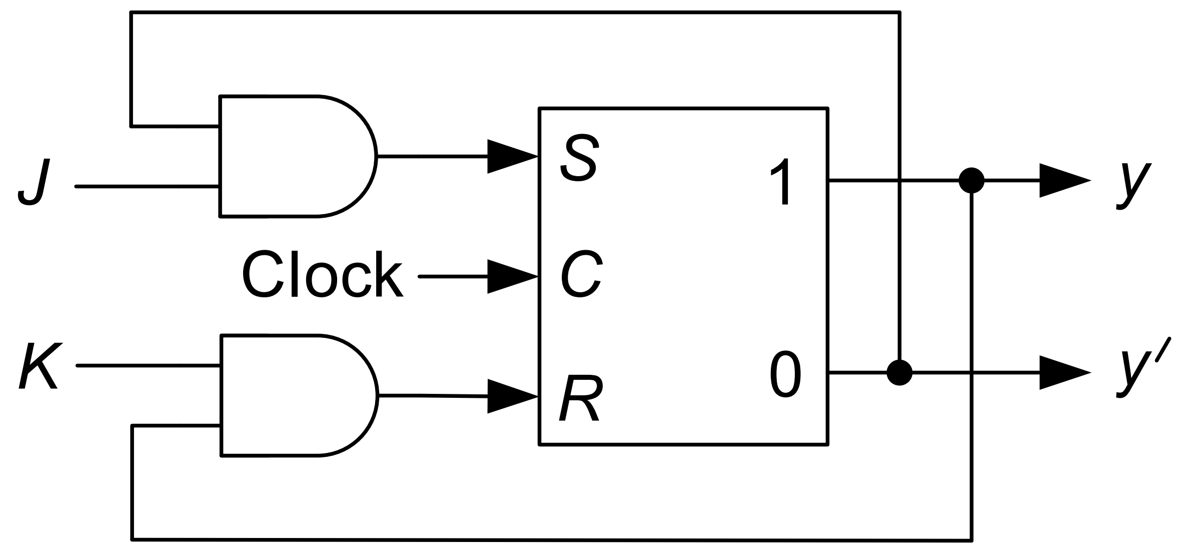
Can you write the characteristic equation?

Memory Element: JK Latch

Unlike the *SR* latch, $J = K = 1$ is permitted: when it occurs, the latch acts like a trigger and switches to the complement state



(*a*) Block diagram.



(*b*) Constructing the JK latch from the clocked SR latch.

Excitation requirements:

Circuit change		*Required value*	
From:	*To:*	*J*	*K*
0	0	0	–
0	1	1	–
1	0	–	1
1	1	–	0

"Q" is basically "y"

Characteristic Table

***JK* Flip-Flop**

J	***K***	**Q(*t* + 1)**	
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$Q'(t)$	Complement

Can you write the characteristic equation?

$$y(t+1) = Jy(t)' + K'y(t)$$

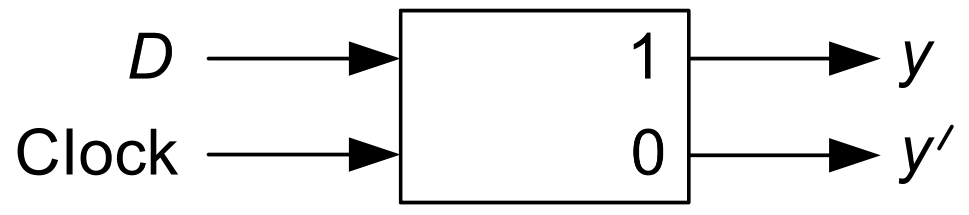
D Latch — The Latch of Your Life

The next state of the *D* latch is equal to its present excitation:

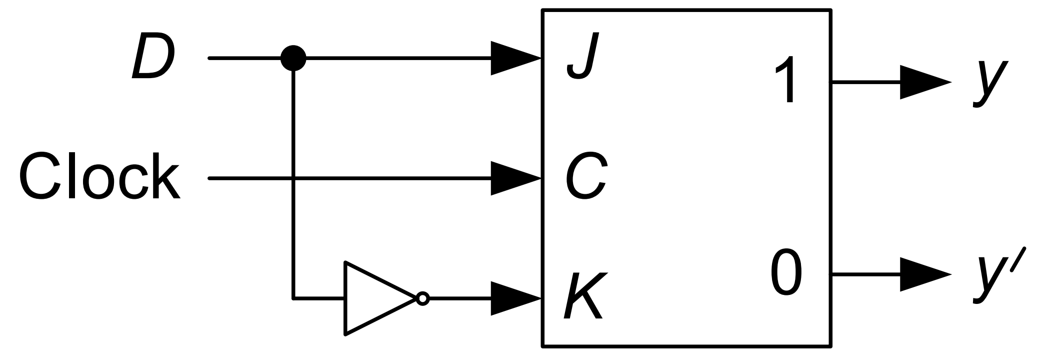
$$y(t+1) = D(t)$$

D Flip-Flop

D	$Q(t+1)$	
0	0	Reset
1	1	Set



(*a*) Block diagram.



(*b*) Transforming the JK latch to the D latch.

Excitation Table

Q(t)	Q(t+1)	D
0	0	0
0	1	1
1	0	0
1	1	1

How is Your Clock?

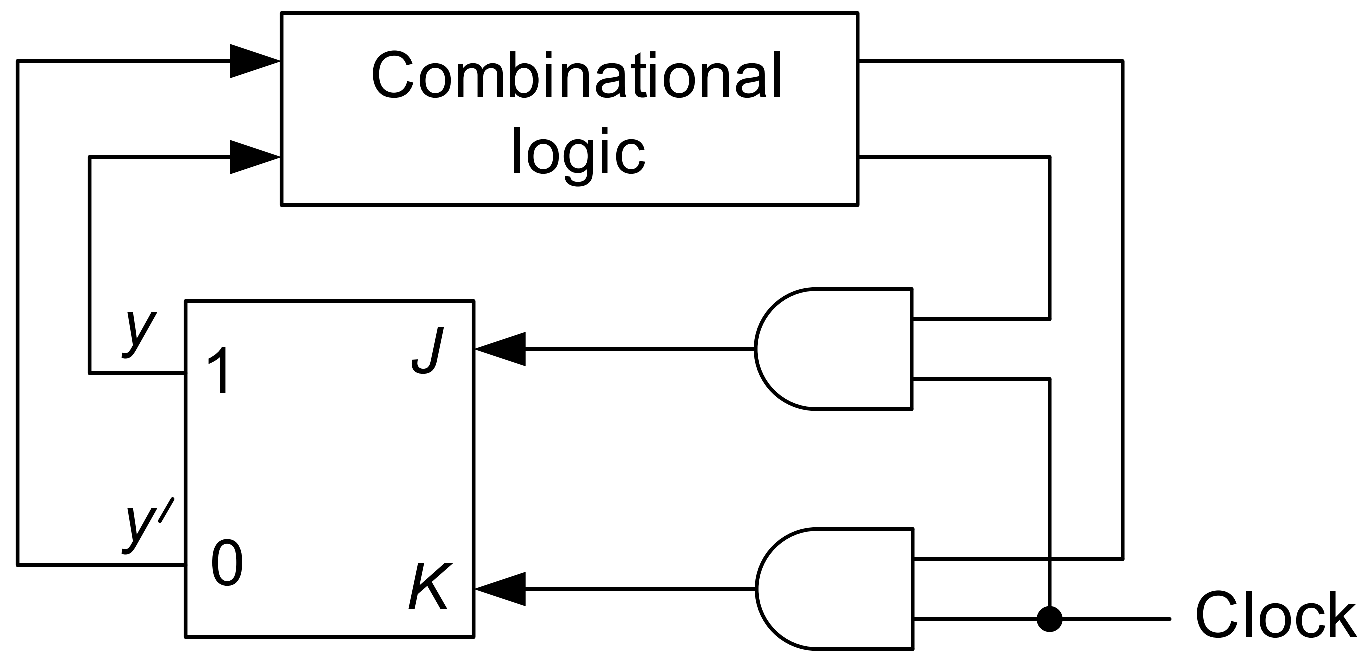
Clocked latch: changes state only in synchronization with the clock pulse and no more than once during each occurrence of the clock pulse

Duration of clock pulse: determined by circuit delays and signal propagation time through the latches

- Must be long enough to allow latch to change state, and
- Short enough so that the latch will not change state twice due to the same excitation

Excitation of a *JK* latch within a sequential circuit:

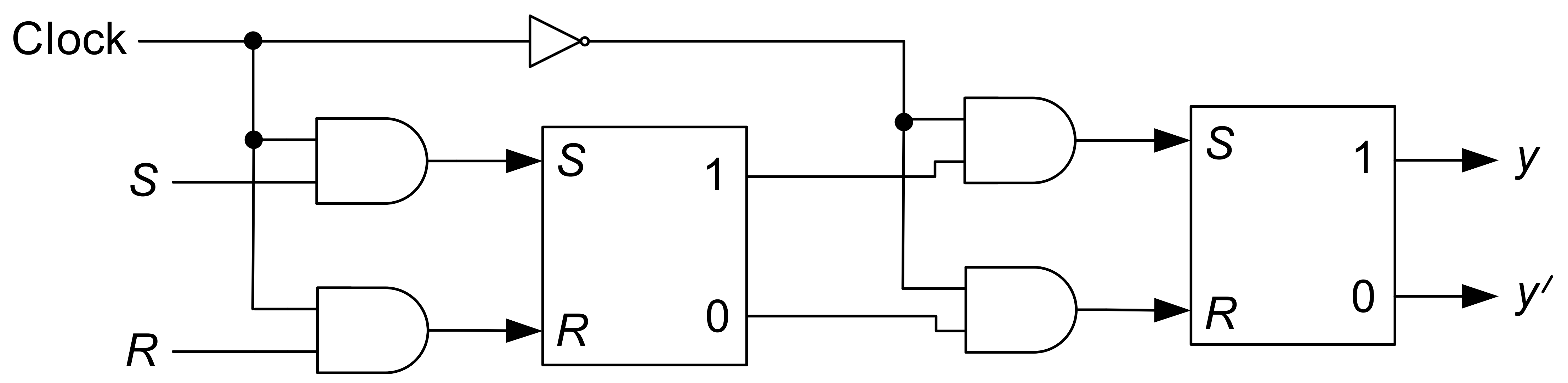
- Length of the clock pulse must allow the latch to generate the *y*'s
- But should not be present when the values of the *y*'s have propagated through the combinational circuit



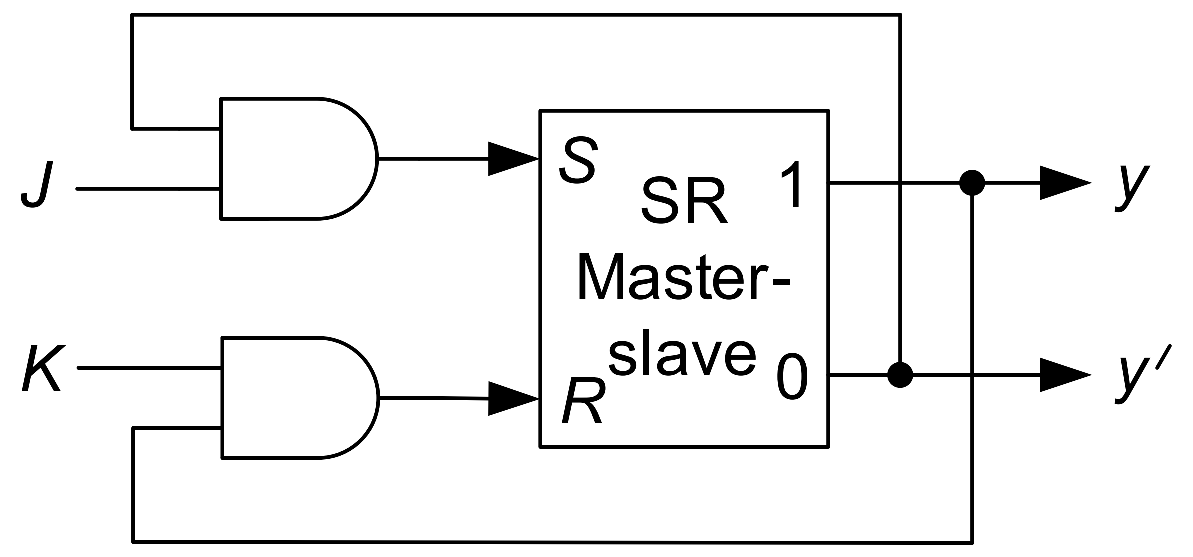
Master Slave Flip-Flop

Master-slave flip-flop: a type of synchronous memory element that eliminates the timing problems by isolating its inputs from its outputs

Master-slave *SR* flip-flop:



Master-slave *JK* flip-flop: since master-slave *SR* flip-flop suffers from the problem that both its inputs cannot be 1, it can be converted to a *JK* flip-flip



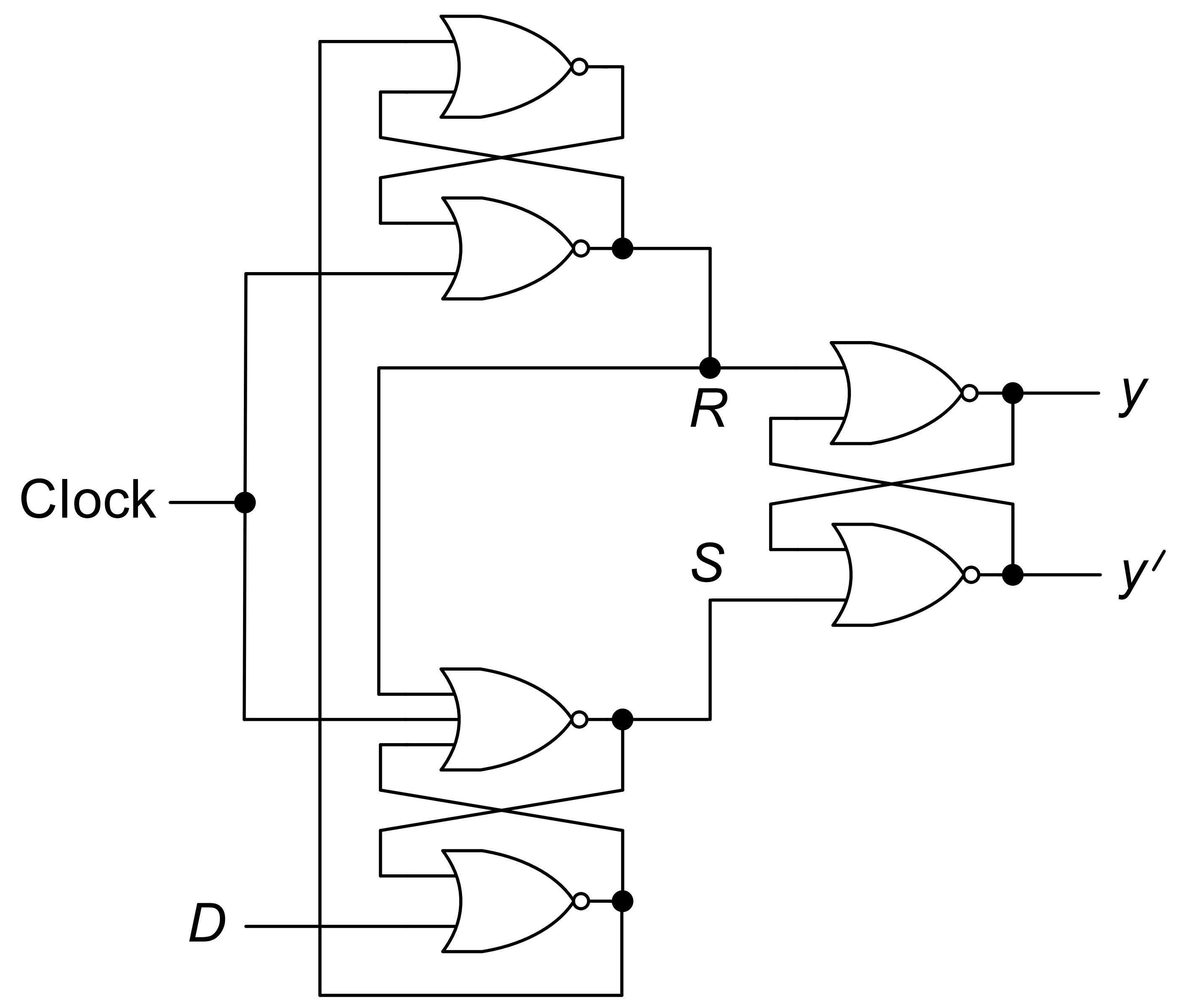
Edge Triggered Flip-Flop

Positive (negative) edge-triggered *D* flip-flip: stores the value at the *D* input when the clock makes a 0 -> 1 (1 -> 0) transition

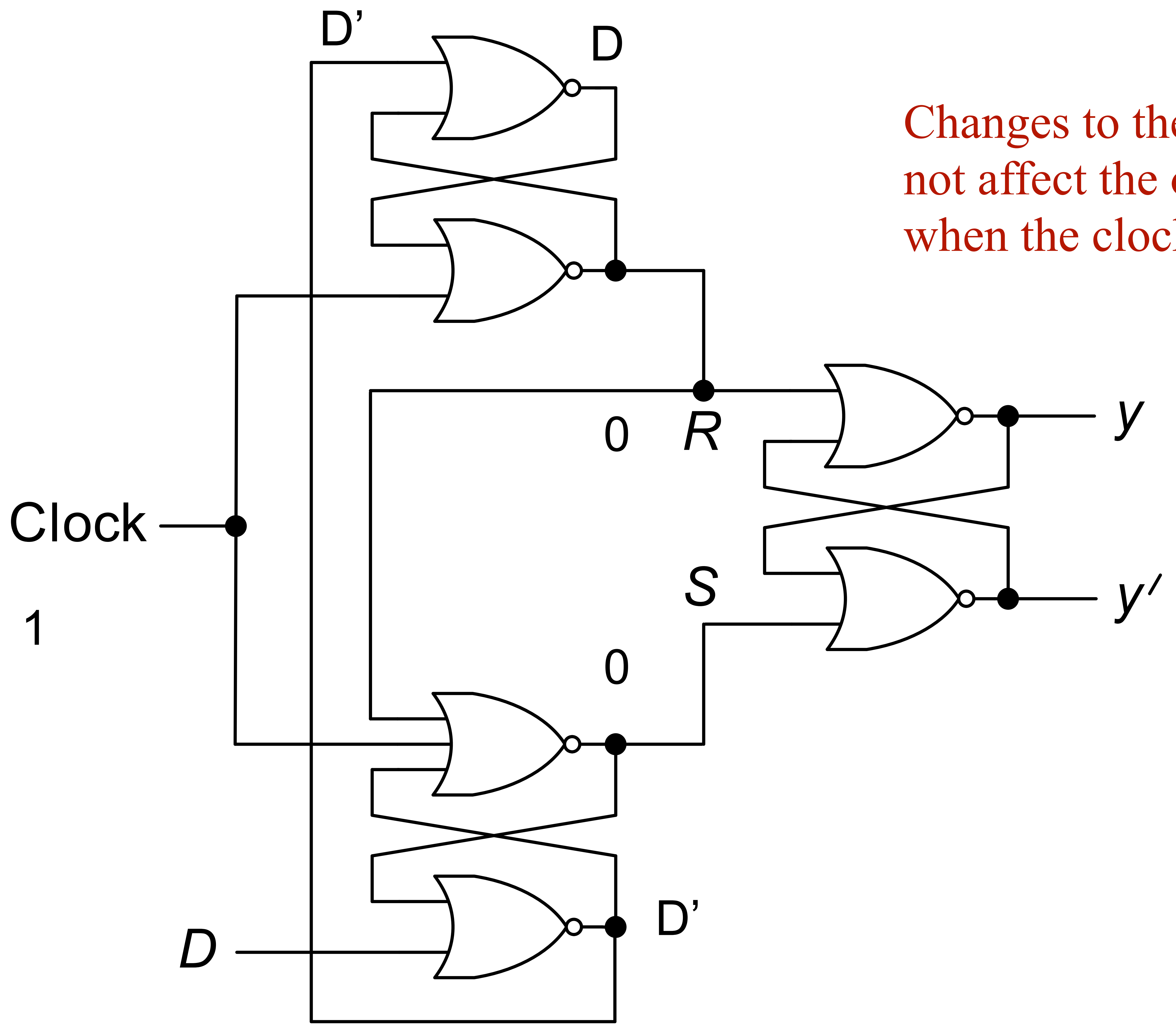
- Any change at the *D* input after the clock has made a transition does not have any effect on the value stored in the flip-flop

A negative edge-triggered *D* flip-flop:

- When the clock is high, the output of the bottommost (topmost) NOR gate is at *D'* (*D*), whereas the *S*-*R* inputs of the output latch are at 0, causing it to hold previous value
- When the clock goes low, the value from the bottommost (topmost) NOR gate gets transferred as *D* (*D'*) to the *S* (*R*) input of the output latch
  - Thus, output latch stores the value of *D*
- If there is a change in the value of the *D* input after the clock has made its transition, the bottommost NOR gate attains value 0
  - However, this cannot change the *SR* inputs of the output latch

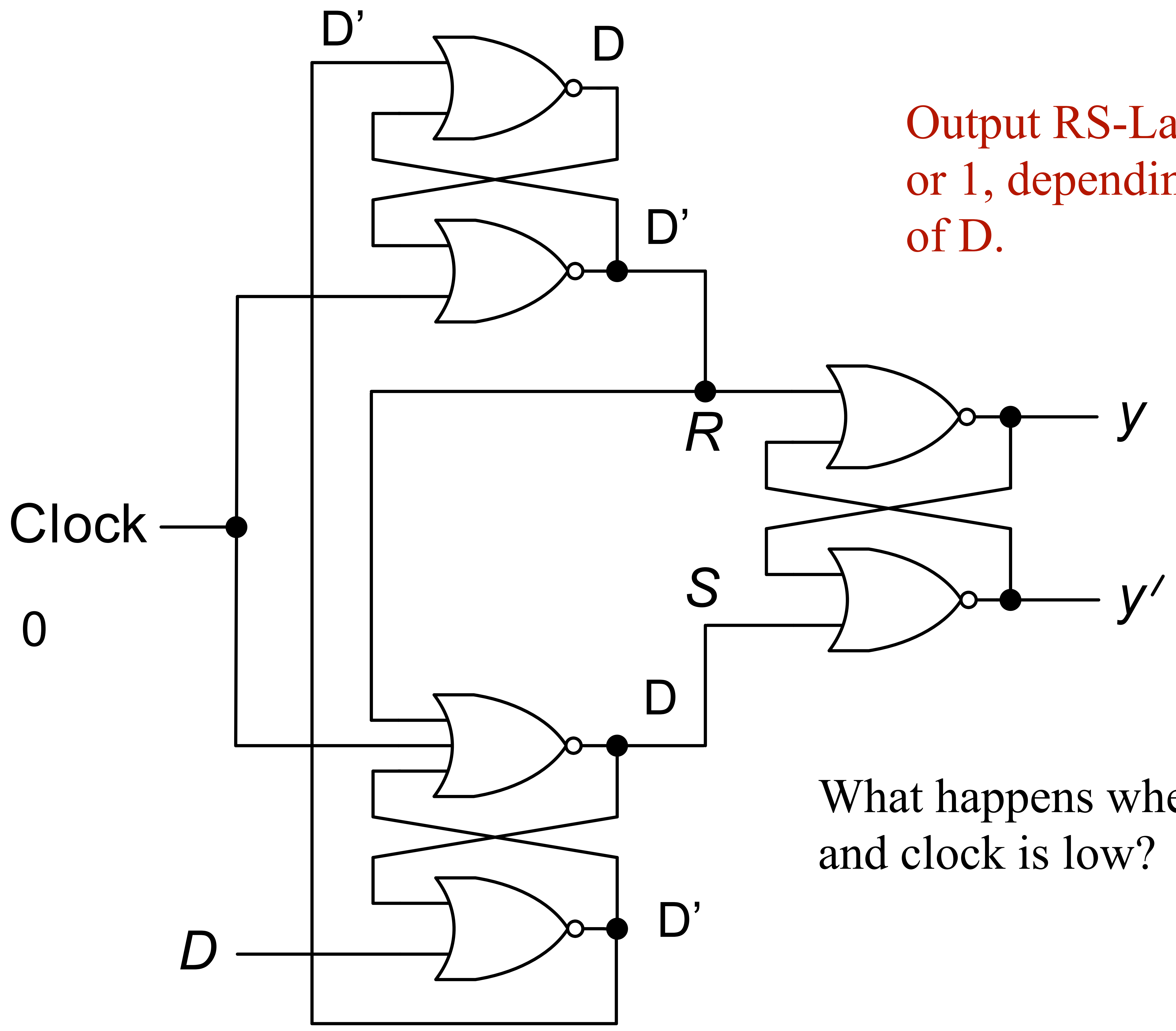


Edge Triggered Flip-Flop



Changes to the input D does not affect the output y and y' when the clock is high.

Edge Triggered Flip-Flop



Output RS-Latch stores a 0 or 1, depending on the value of D.

What happens when D changes and clock is low?