Digital Logic Design + Computer Architecture

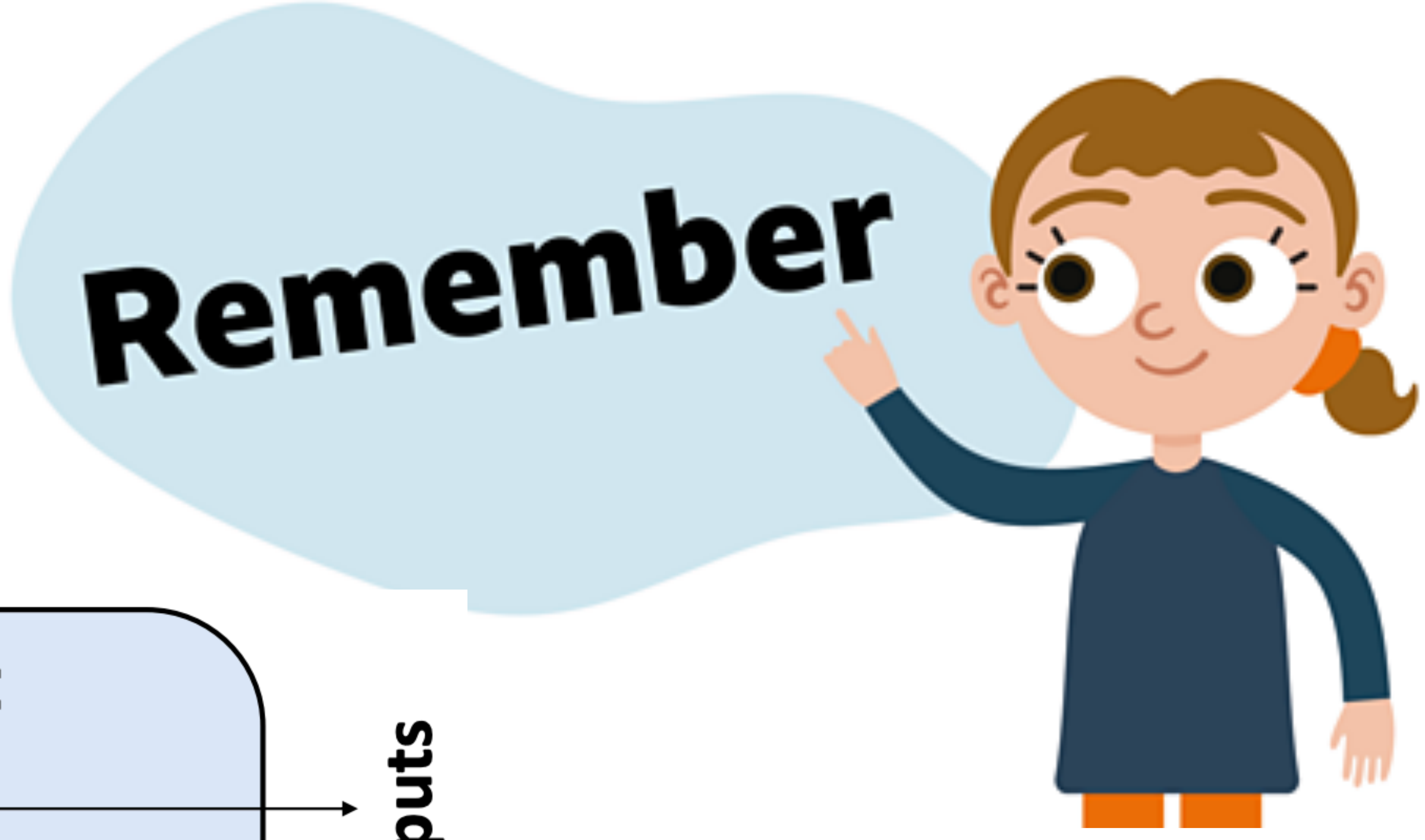
Sayandeep Saha

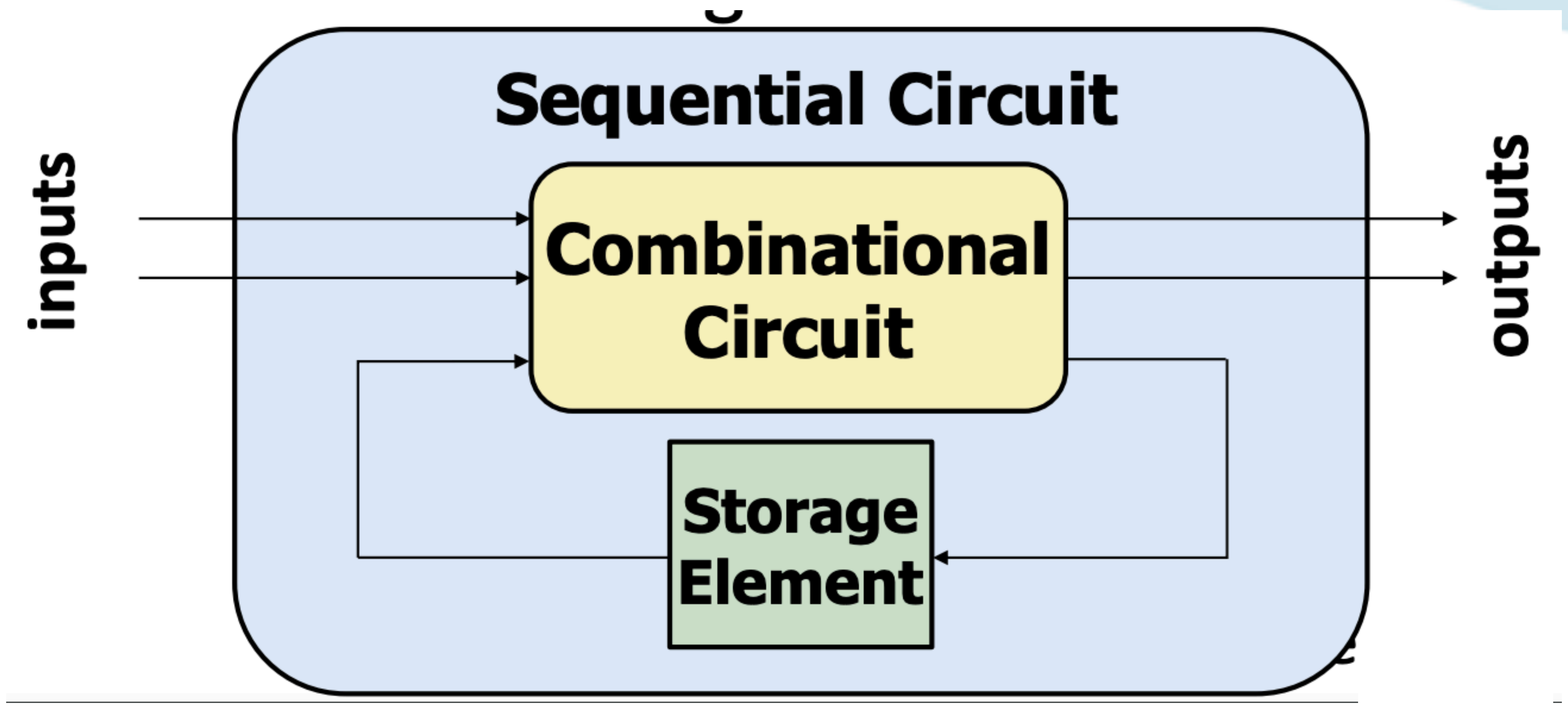
Assistant Professor
Department of Computer Science and Engineering
Indian Institute of Technology Bombay

Sequential Circuits

A Circuit that Remembers

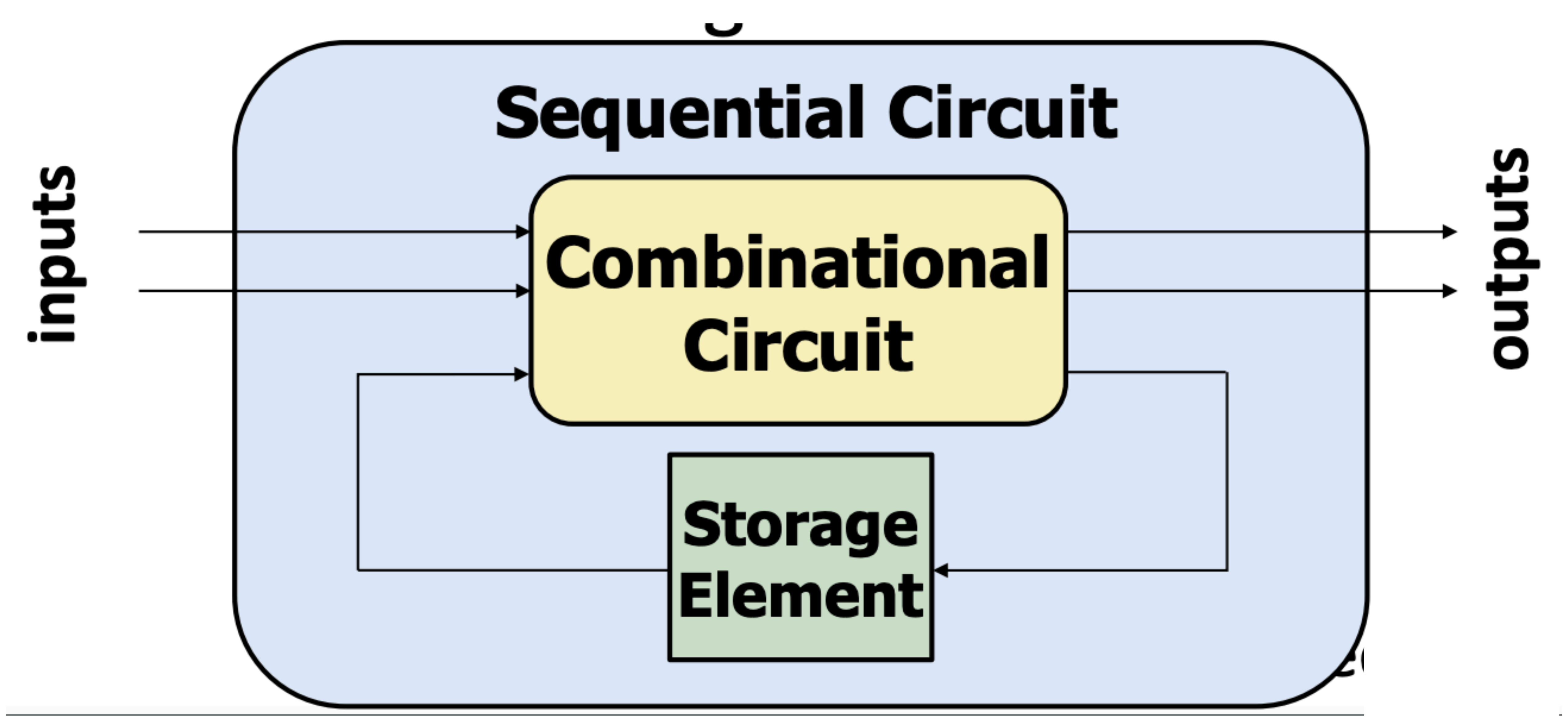
- **How do you remember things?**
  - Memory
- **Can we design a circuit which remembers?**
  - A formal way to model this capability is called a **state**
  - **So we will be modelling circuits to create a state.**



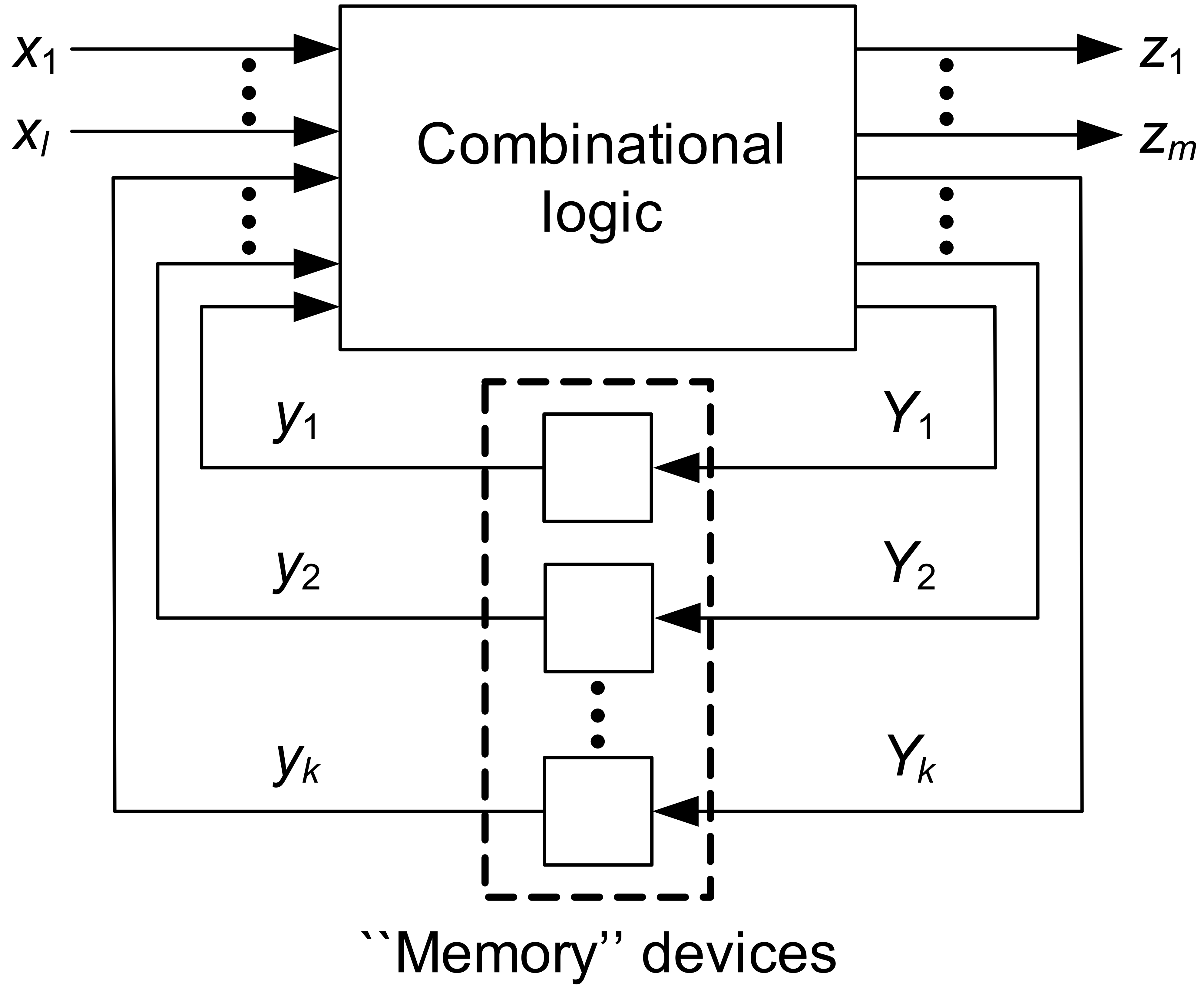


A Circuit that Remembers

- **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



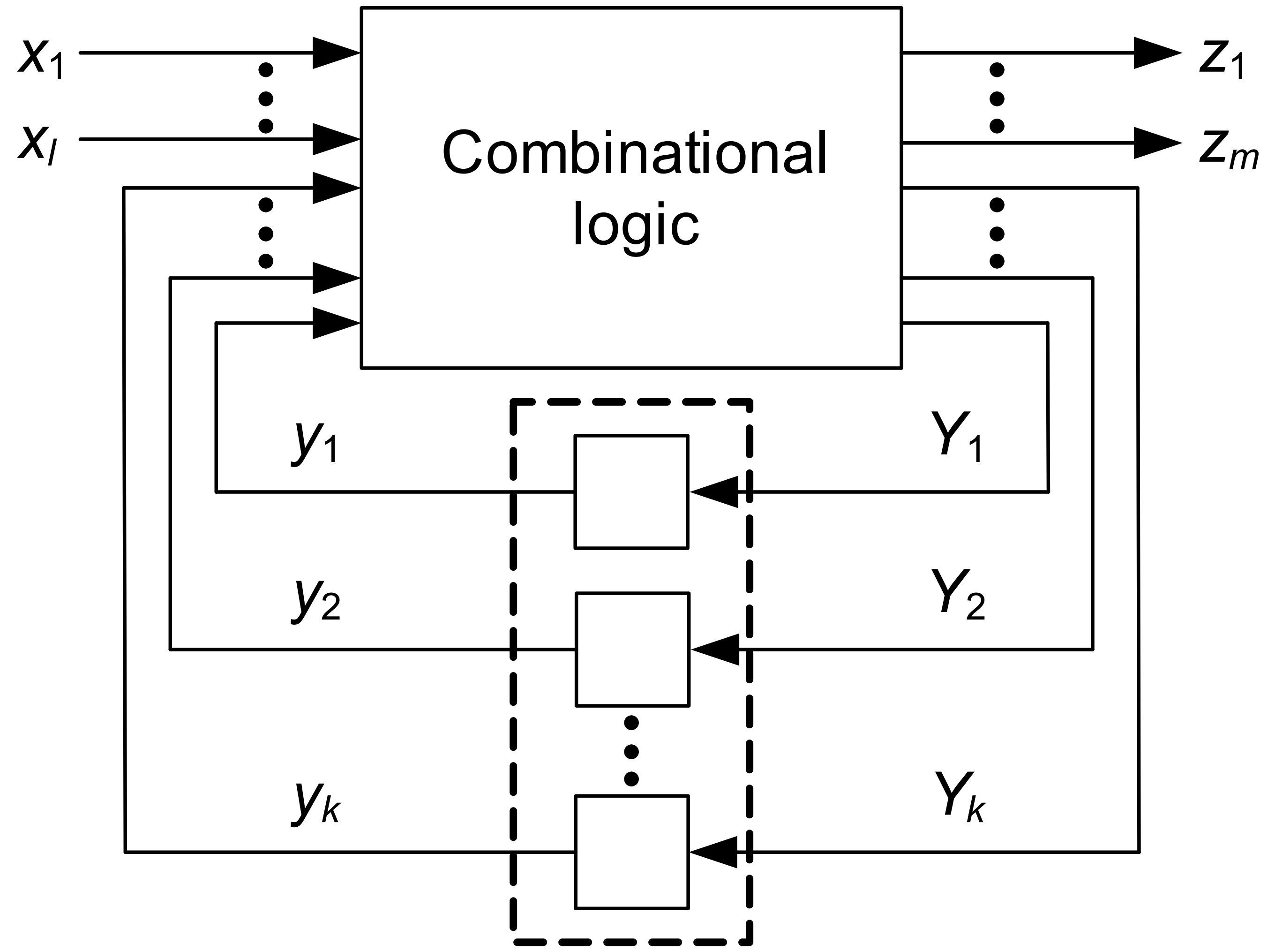
Sequential Circuits

To generate the Y's: memory devices must be supplied with appropriate input values

- **Characteristic table/functions**: switching functions that describe the impact of x_i's and y_j's on the memory-element input
- **Excitation table**: its entries are the values of the memory-element inputs

Most widely used memory elements: **flip-flops**, which are made of **latches**

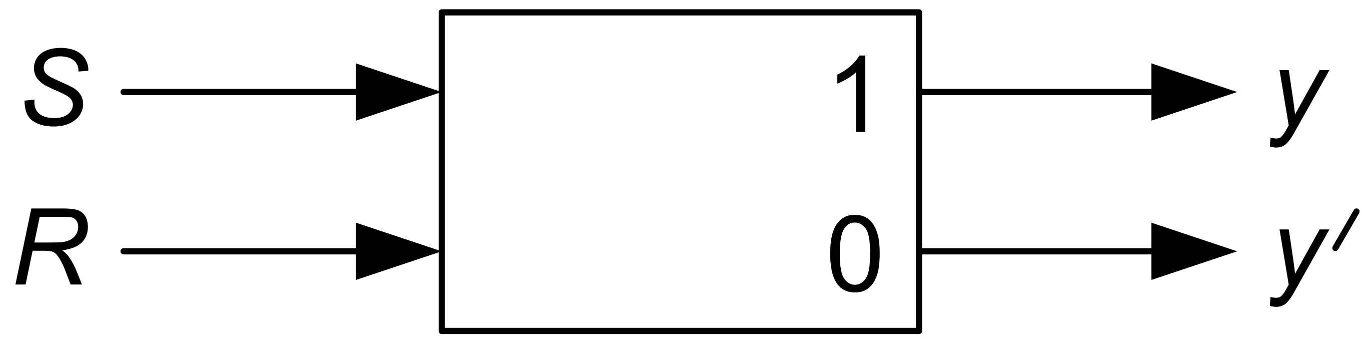
- **Latch**: remains in one state indefinitely until an input signals directs it to do otherwise



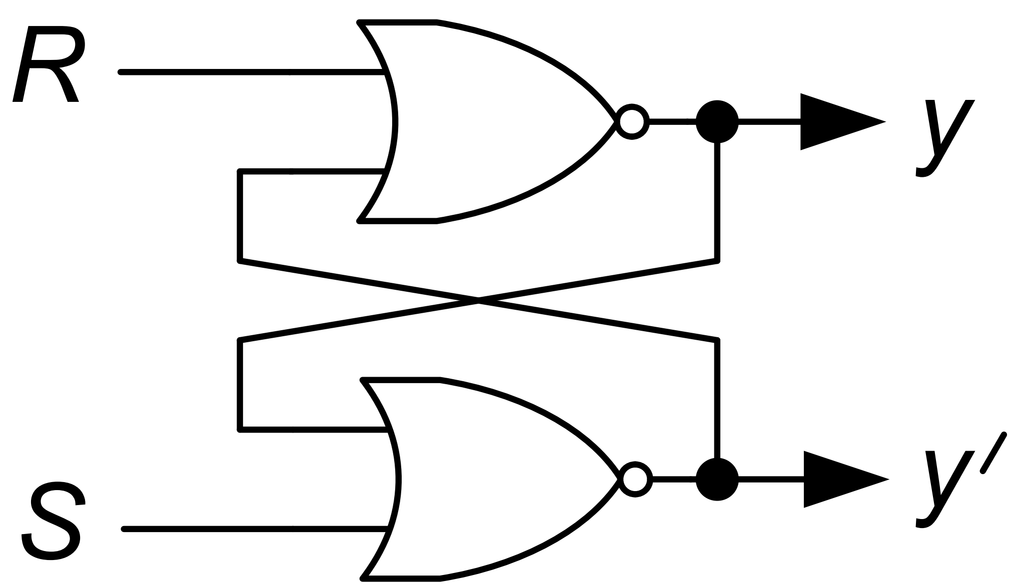
Memory Element: Latches

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

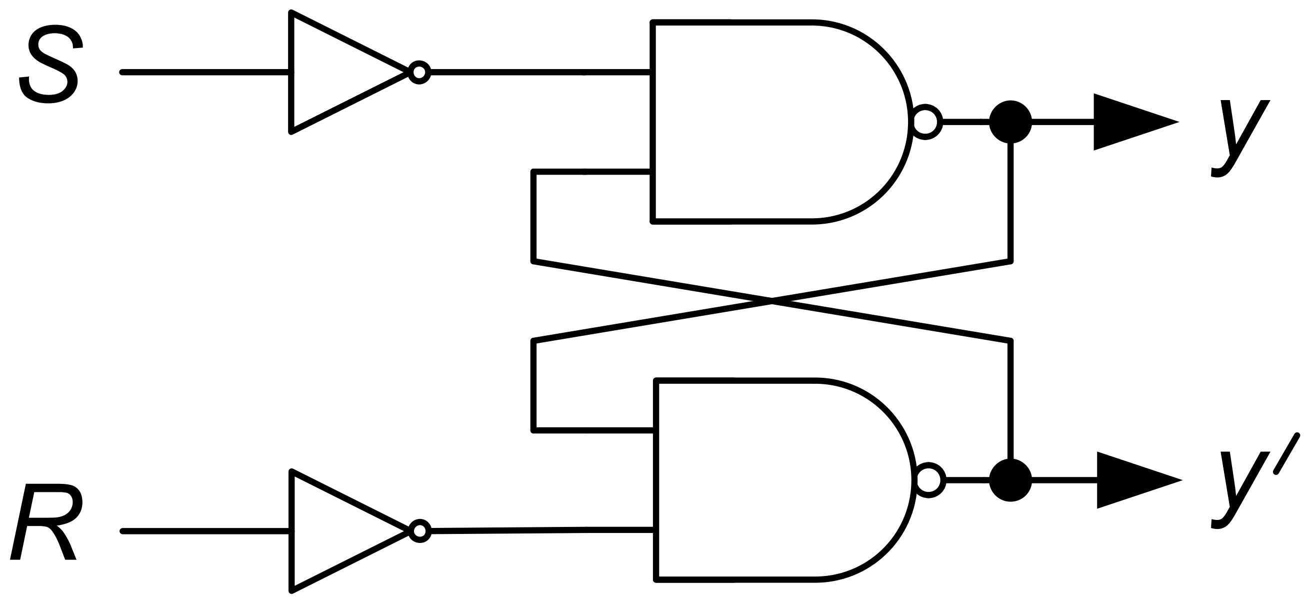
Set-reset of *SR* latch:



(*a*) Block diagram.



(*b*) NOR latch.



(*c*) NAND latch.

Memory Element: Latches

Characteristic table and excitation requirements:

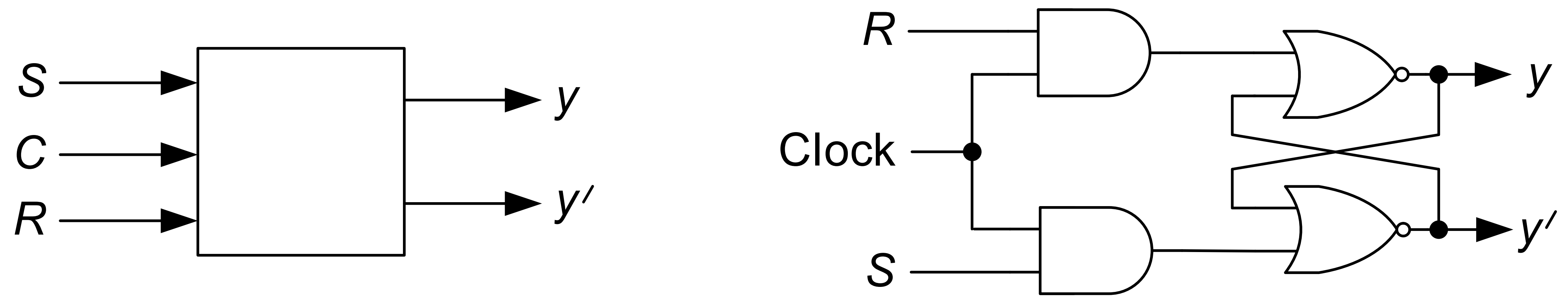
$y(t)$	$S(t)$	$R(t)$	$y(t+1)$
0	0	0	0
0	0	1	0
0	1	1	?
0	1	0	1
1	1	0	1
1	1	1	?
1	0	1	0
1	0	0	1

$$RS = 0$$
$$y(t+1) = R'y(t) + S$$

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

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

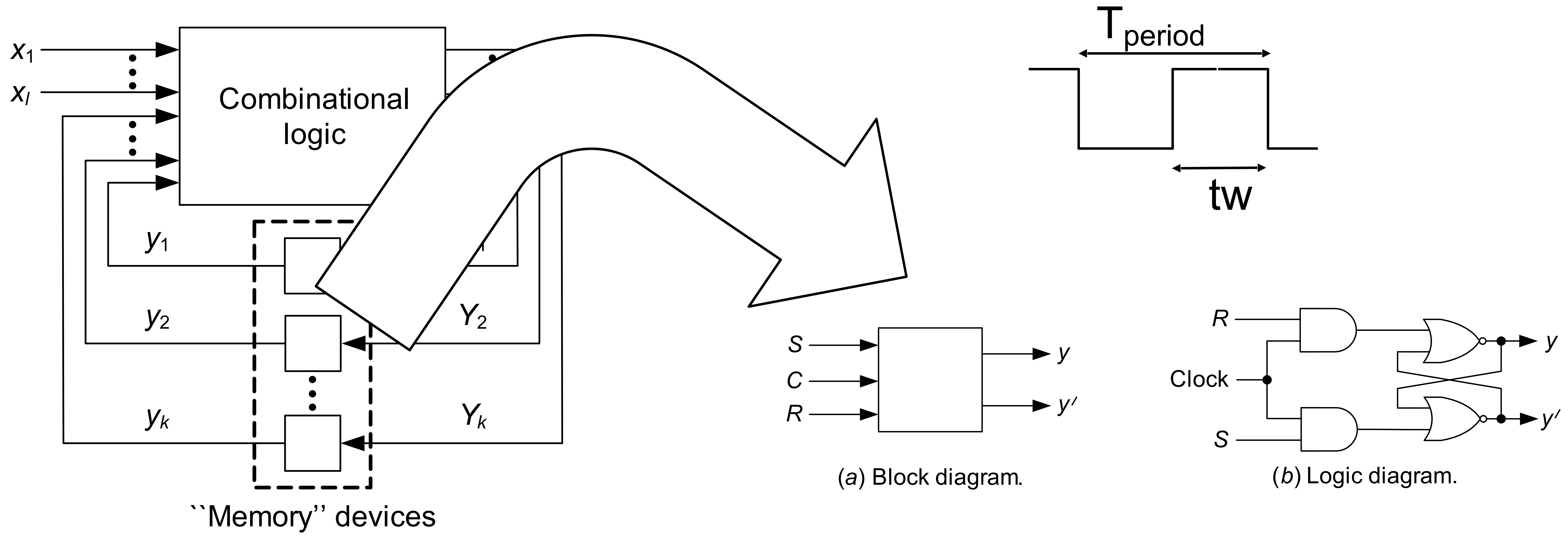
Why is the (1,1) input forbidden?

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

$$RS = 0$$
$$y(t+1) = R'y(t) + S$$

1. If **R=S=1, Q** and **Q'** will both settle to 1, which **breaks** our invariant that **Q** = !**Q'**
2. If **S** and **R** transition back to 0 at the same time, **Q** and **Q'** begin to oscillate between 1 and 0 because their final values depend on each other (**metastability**)
   - This eventually settles depending on **variation in the circuits**

Memory Element: Latches



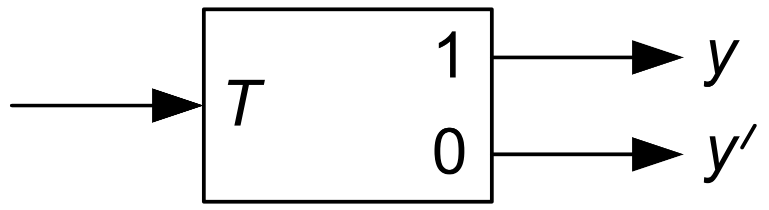
(a) Block diagram.

(b) Logic diagram.

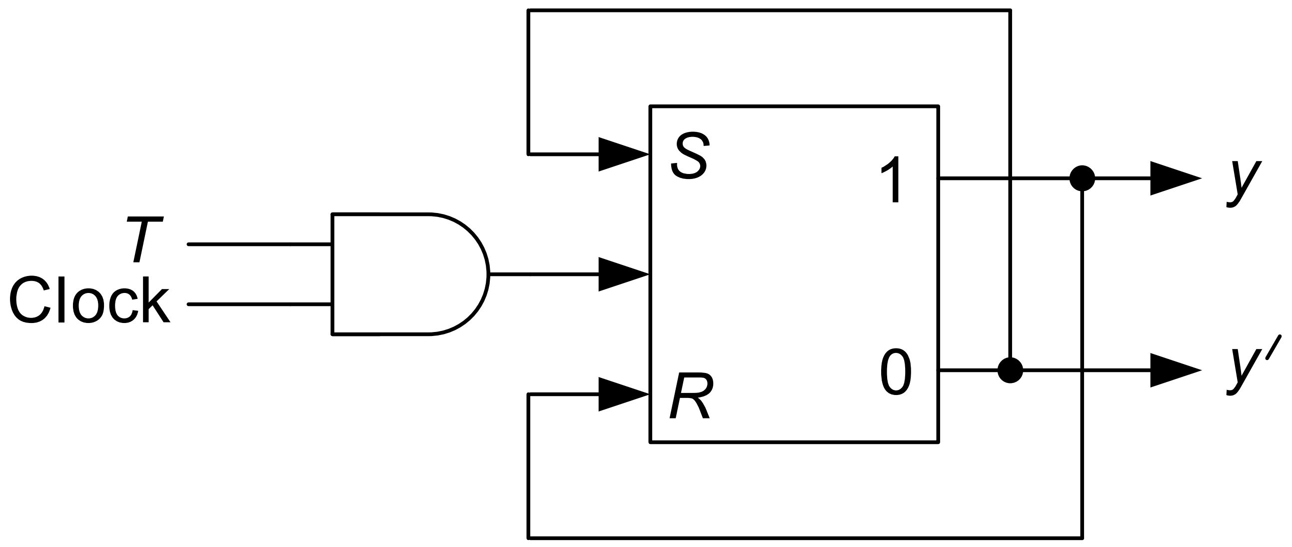
- A **clock** is a periodic signal that is used to keep time in sequential circuits.
- **Duty Cycle** is the ration of t_w/T_{period}
- We want to keep t_w small so that in the same clock pulse only a single computation is performed.
- We want to keep T_{period} sufficient so that there is enough time 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 *From:*	*To:*	*Required value T*
0	0	0
0	1	1
1	0	1
1	1	0

"Q" is basically "y"

Characteristic Table

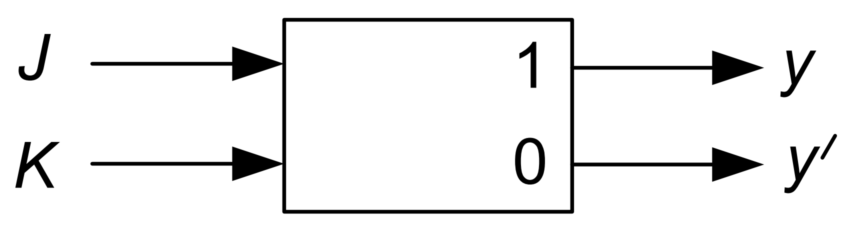
***T* Flip-Flop**

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

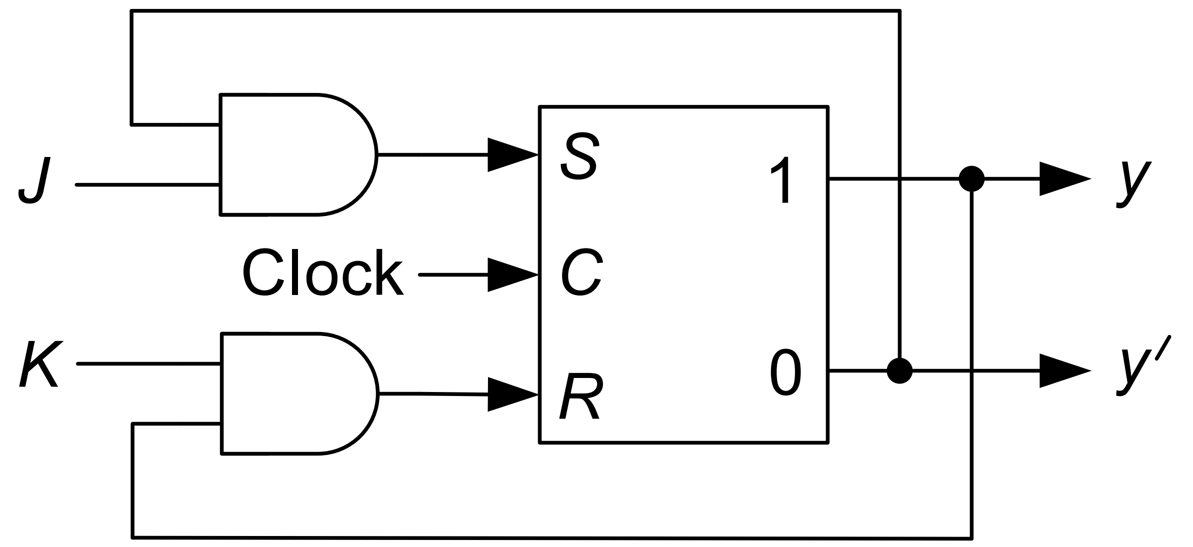
$$y(t+1) = Ty'(t) + T'y(t)$$
$$= T \oplus y(t)$$

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?

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

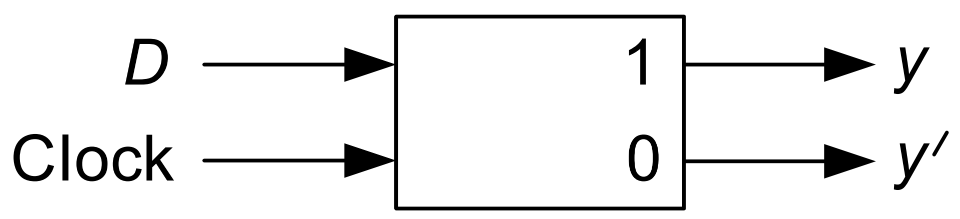
D Latch — The Latch of Your Life

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

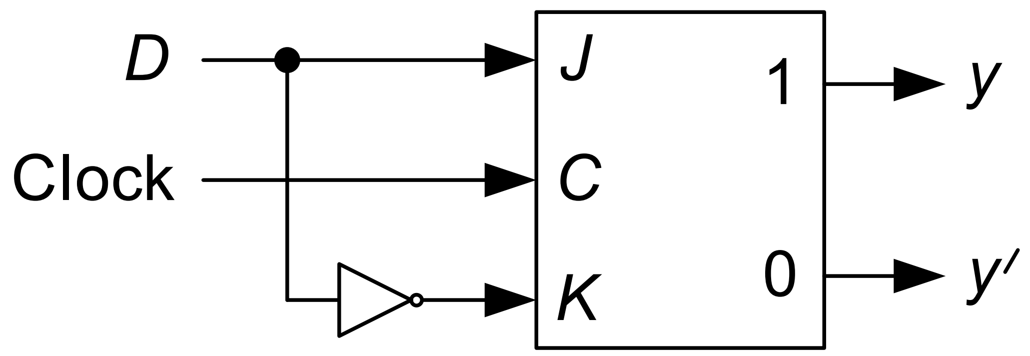
$$y(t+1) = D(t)$$

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

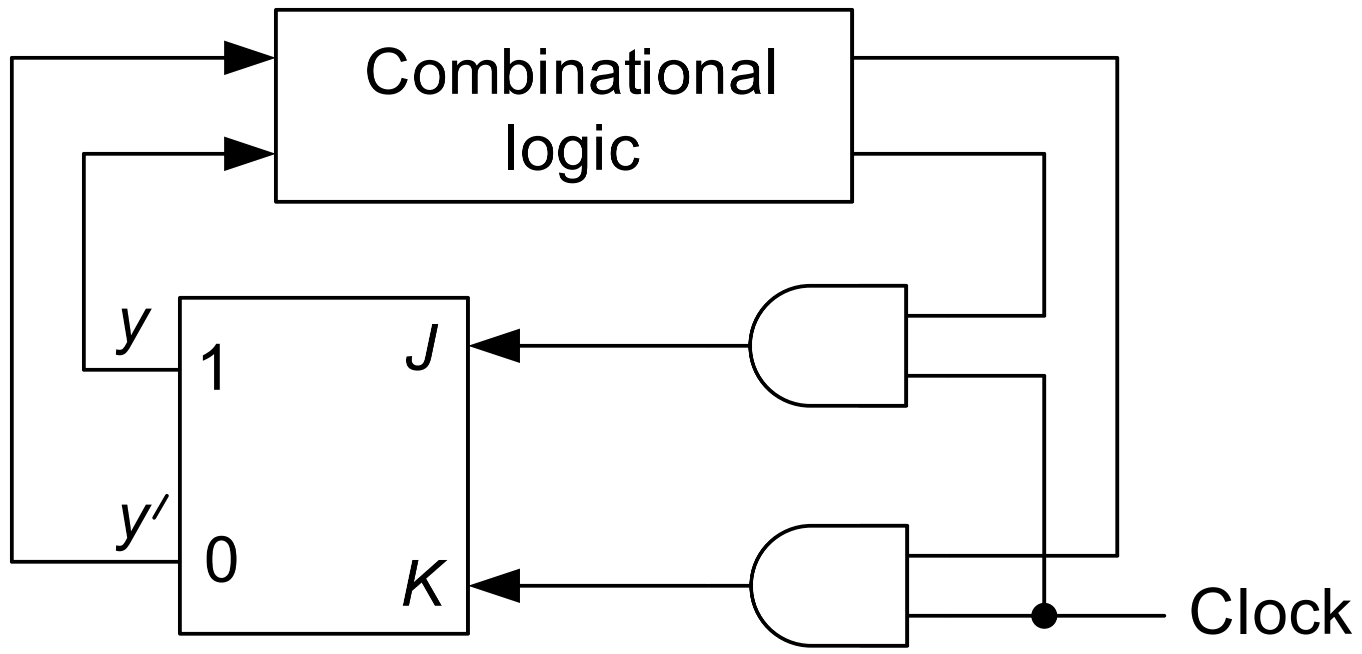
- Must be long enough to allow latch to change state, and
- Short enough so that the latch will not change state twice due to the same excitation

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

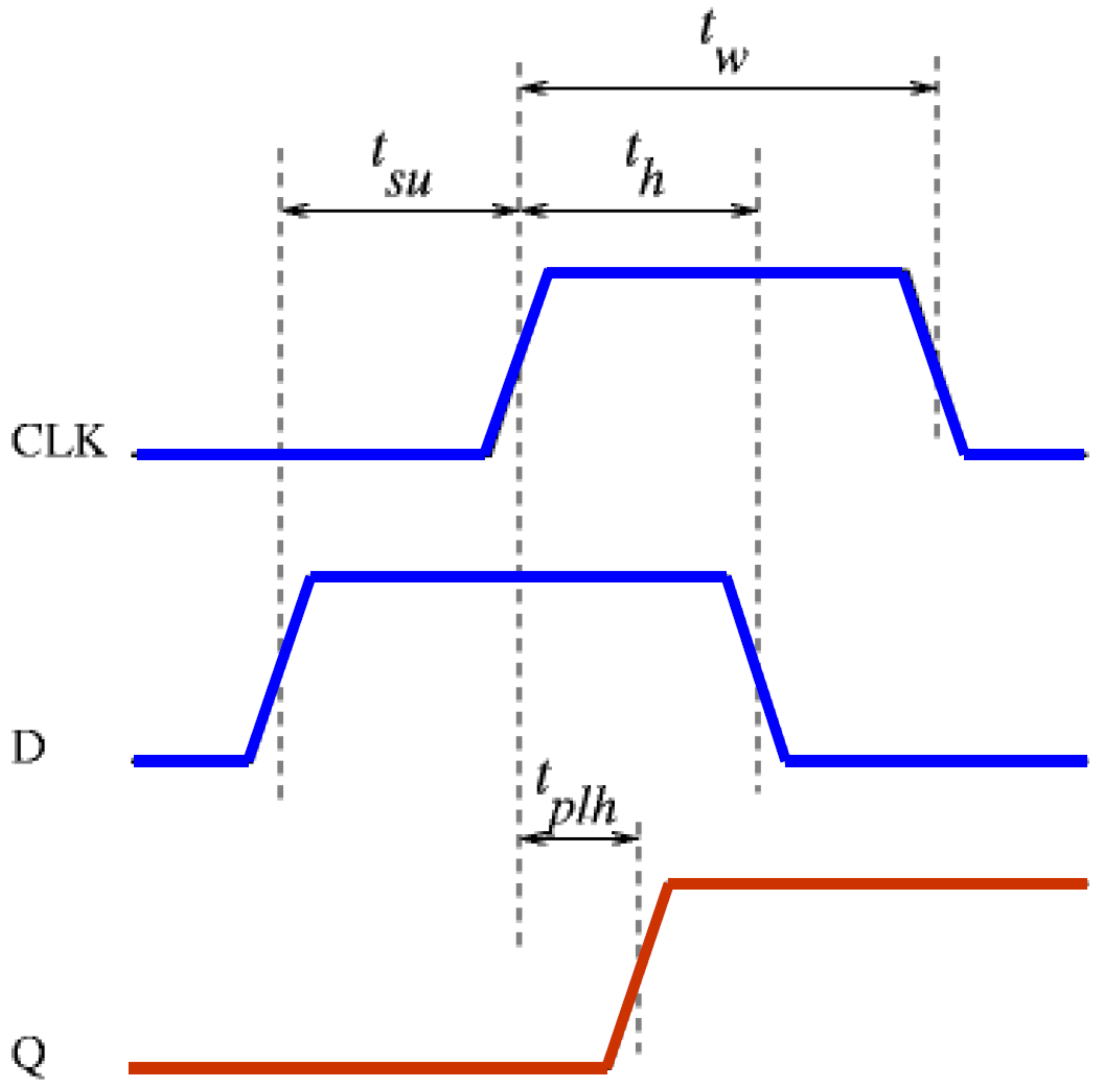
- Length of the clock pulse must allow the latch to generate the *y*'s
- But should not be present when the values of the *y*'s have propagated through the combinational circuit

How fast/slow should be the clock really?

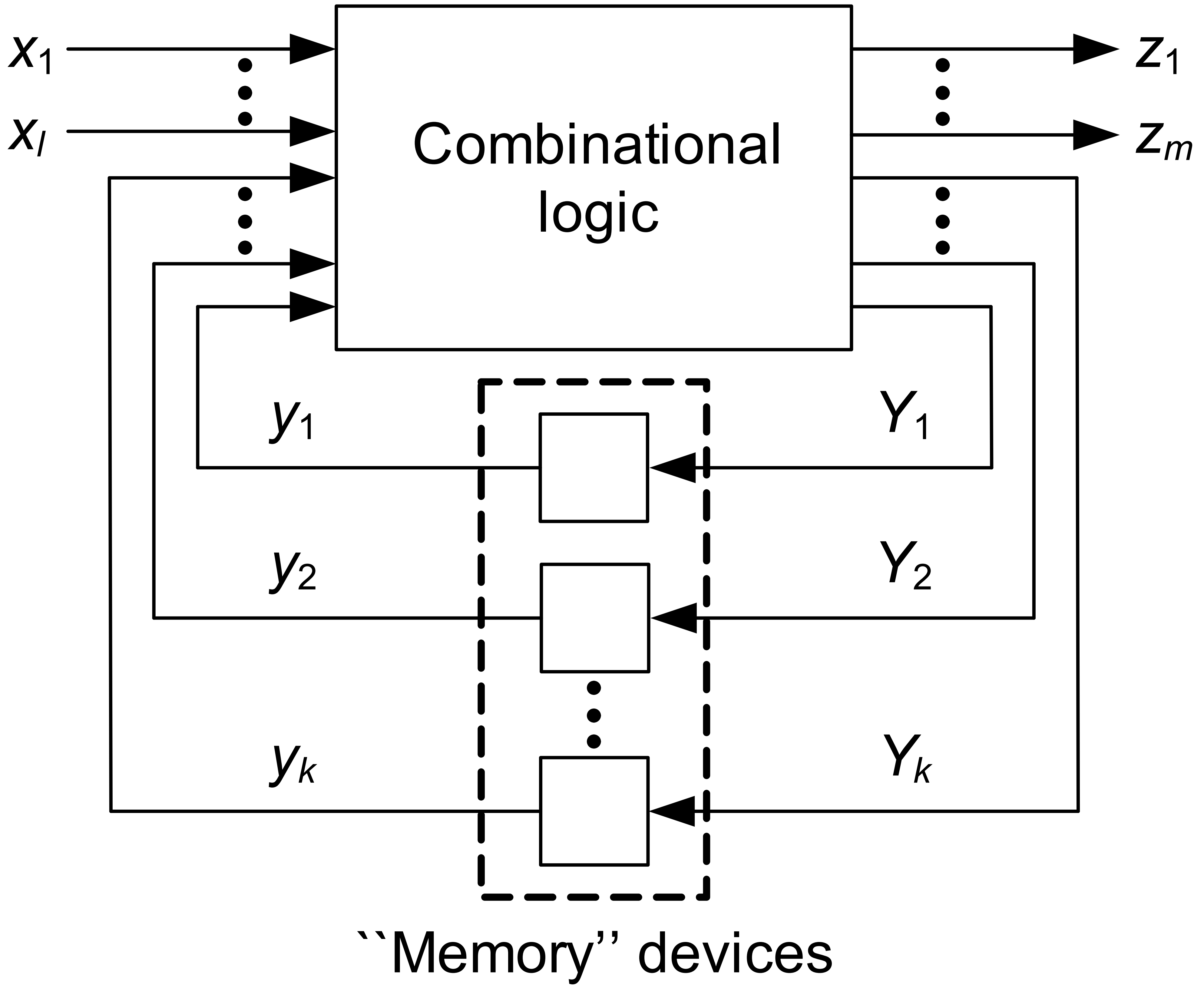
But when does the flip-flop changes its state???



All in One



D Flip–flop (edge–triggered)
(positive edge triggering)



Delay to make sure all is well

- **Setup time**, t_{su}, is the time period prior to the clock becoming active (edge or level) during which the flip-flop inputs must remain stable.

- **Hold time**, t_h, is the time after the clock becomes inactive during which the flip-flop inputs must remain stable.

- Setup time and hold time define a *window of time during which the flip-flop inputs cannot change* – quiescent interval.

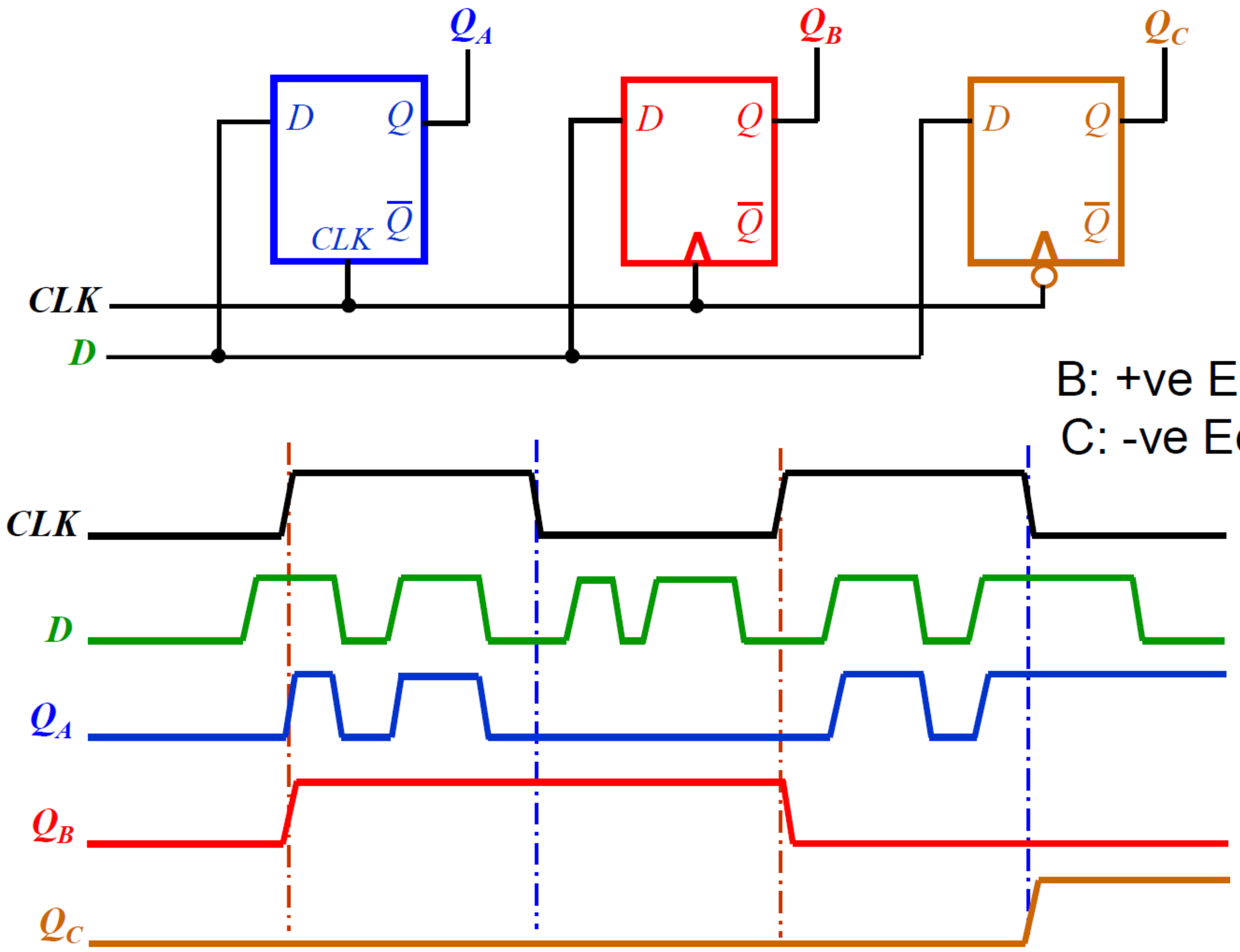
More Delay

- **Propagation delay,** t_{pHL} and t_{pLH} , has the same meaning as in combinational circuit – beware propagation delays usually will not be equal for all input to output pairs. There can be two propagation delays: $t_{C\text{-}Q}$ (*clock*→Q delay) and $t_{D\text{-}Q}$ (*data*→Q delay).

- For a level or pulse triggered latch:
  - Data input should remain stable till the clock becomes inactive.
  - Clock should remain active till the input change is propagated to Q output. That is, active period of the clock,

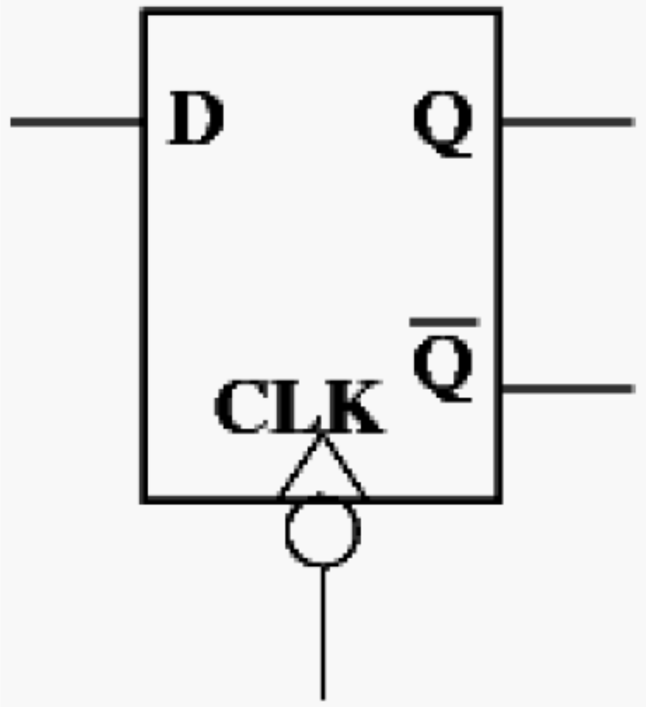
$$t_W > \max\{t_{pLH}, t_{pHL}\}$$

The Triggering Dilemma

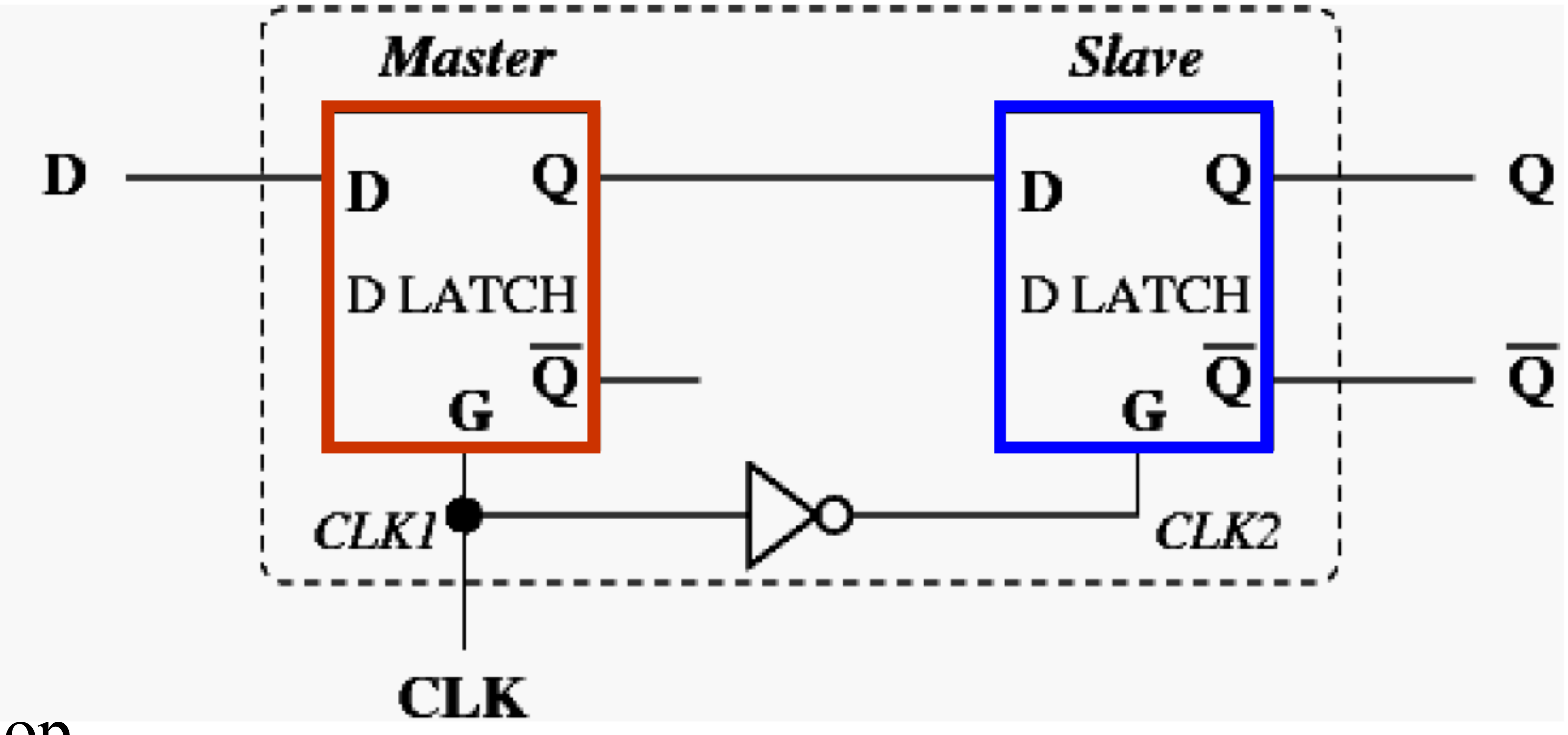


A: Latch
B: +ve Edge trigerred DFF
C: -ve Edge trigerred DFF

Master Slave Flip-Flop



Negative edge triggered flipflop

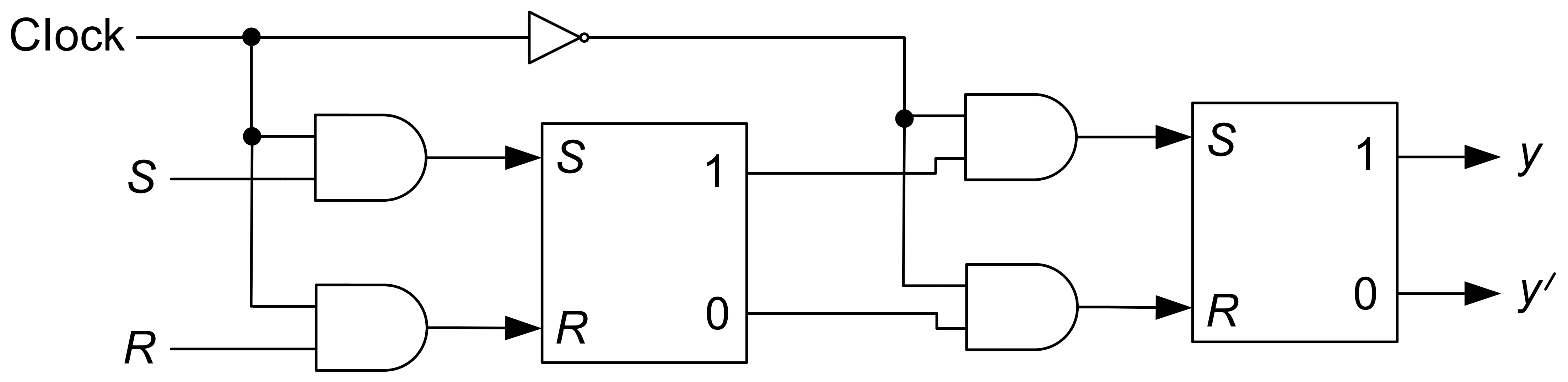


At a given time, only one latch is alive (either master or slave)

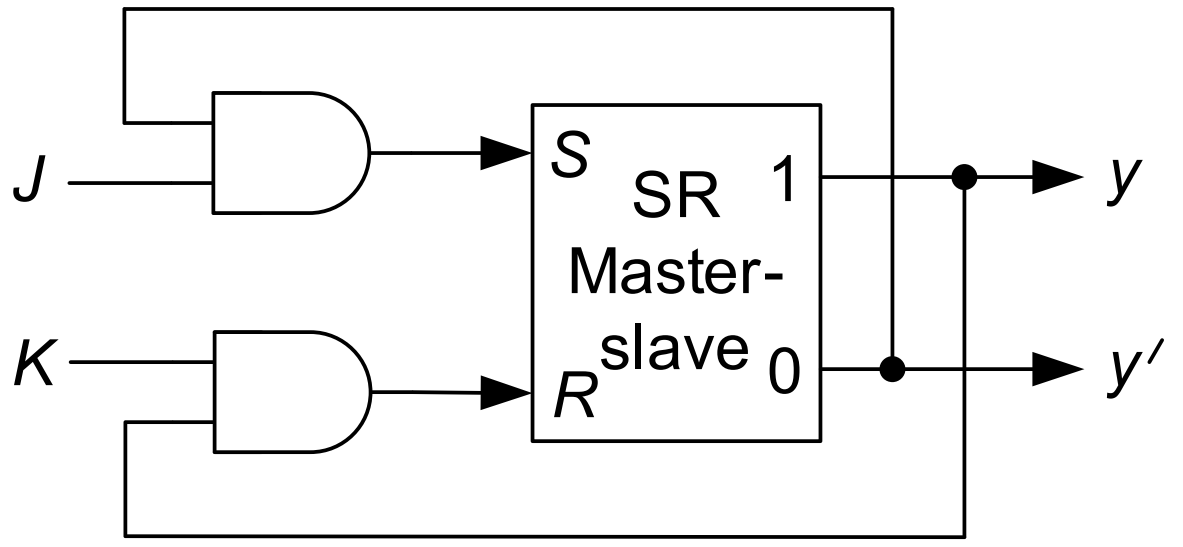
Master Slave Flip-Flop

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



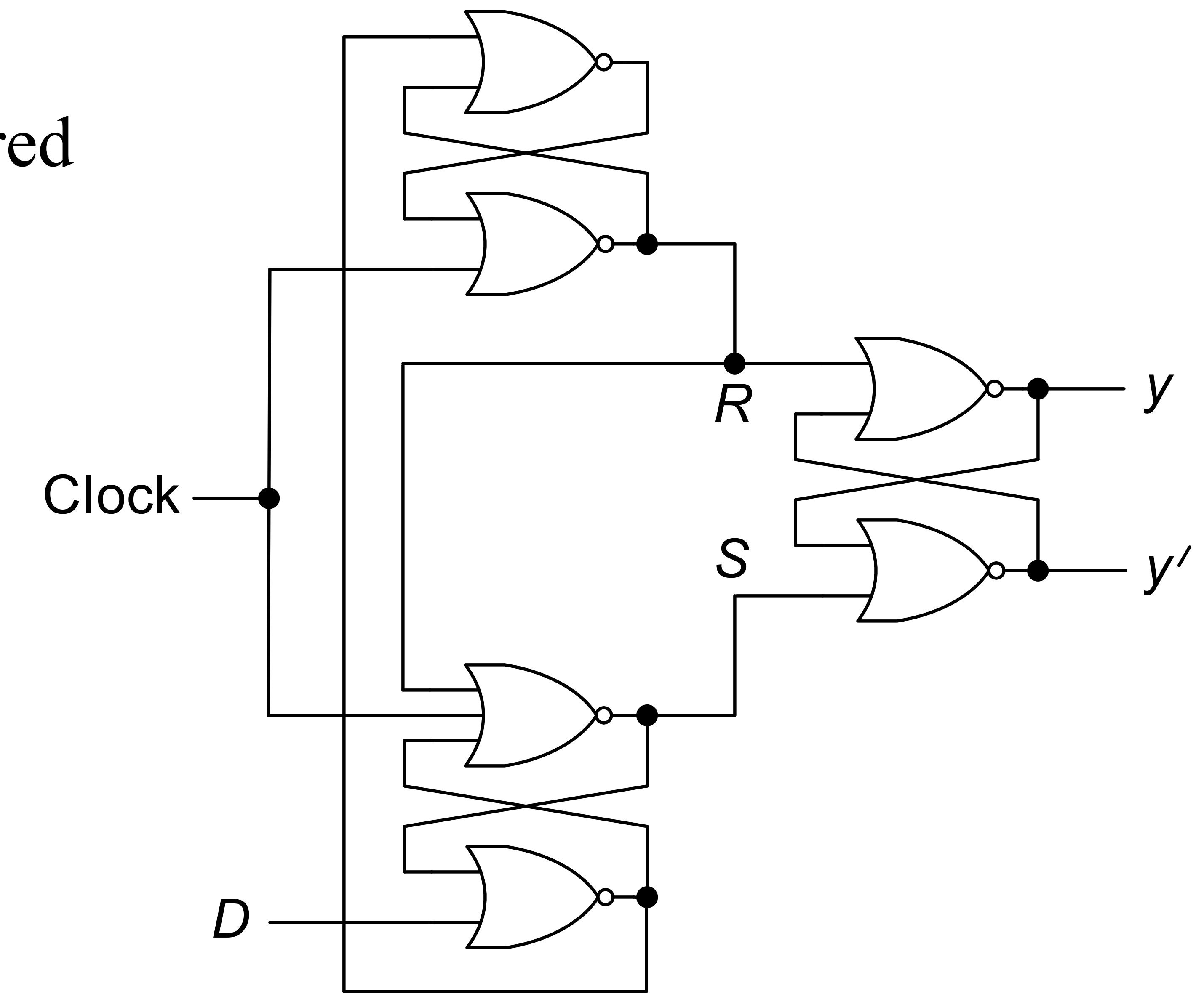
Edge Triggered Flip-Flop

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

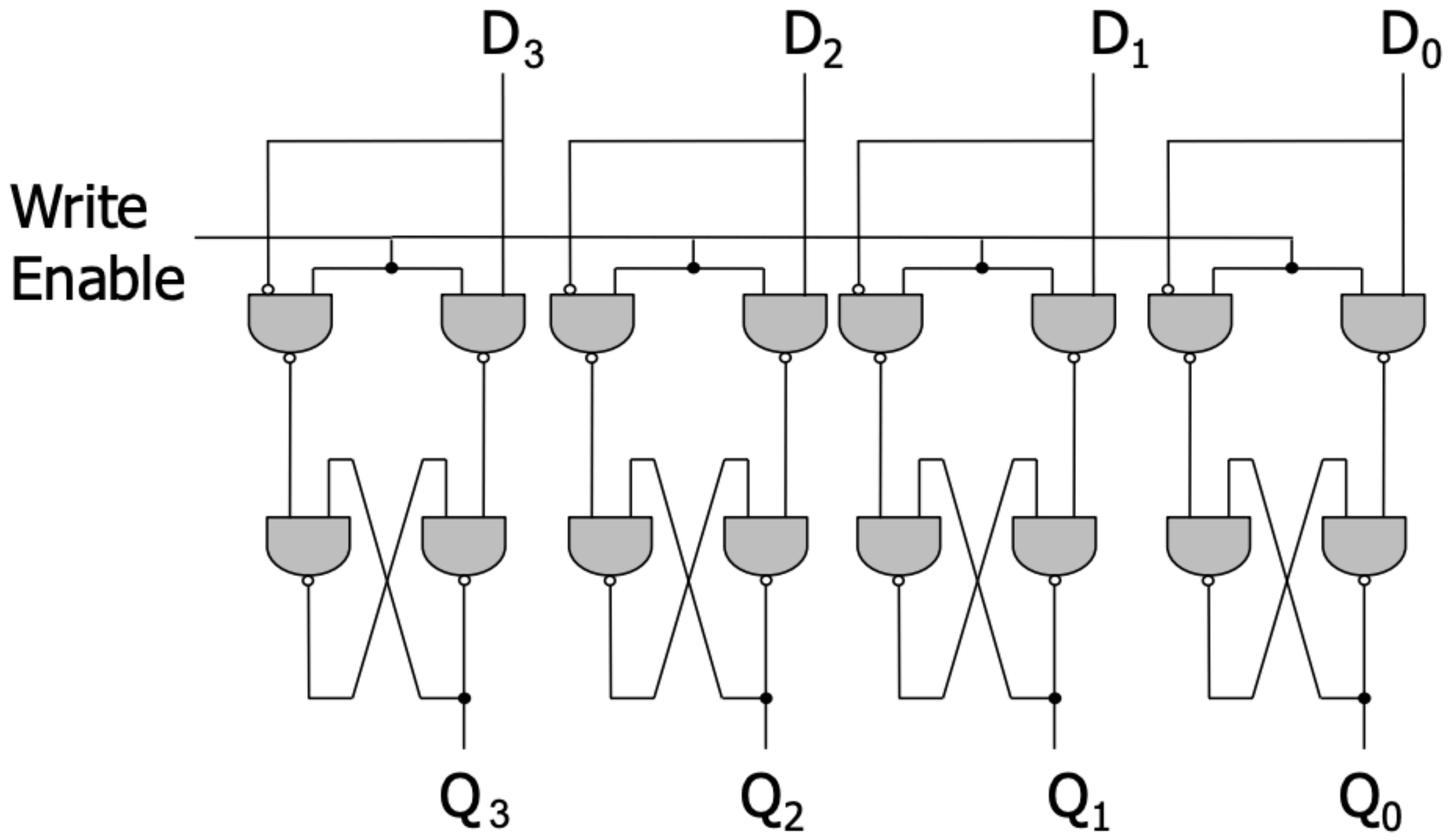
- Any change at the *D* input after the clock has made a transition does not have any effect on the value stored in the flip-flop

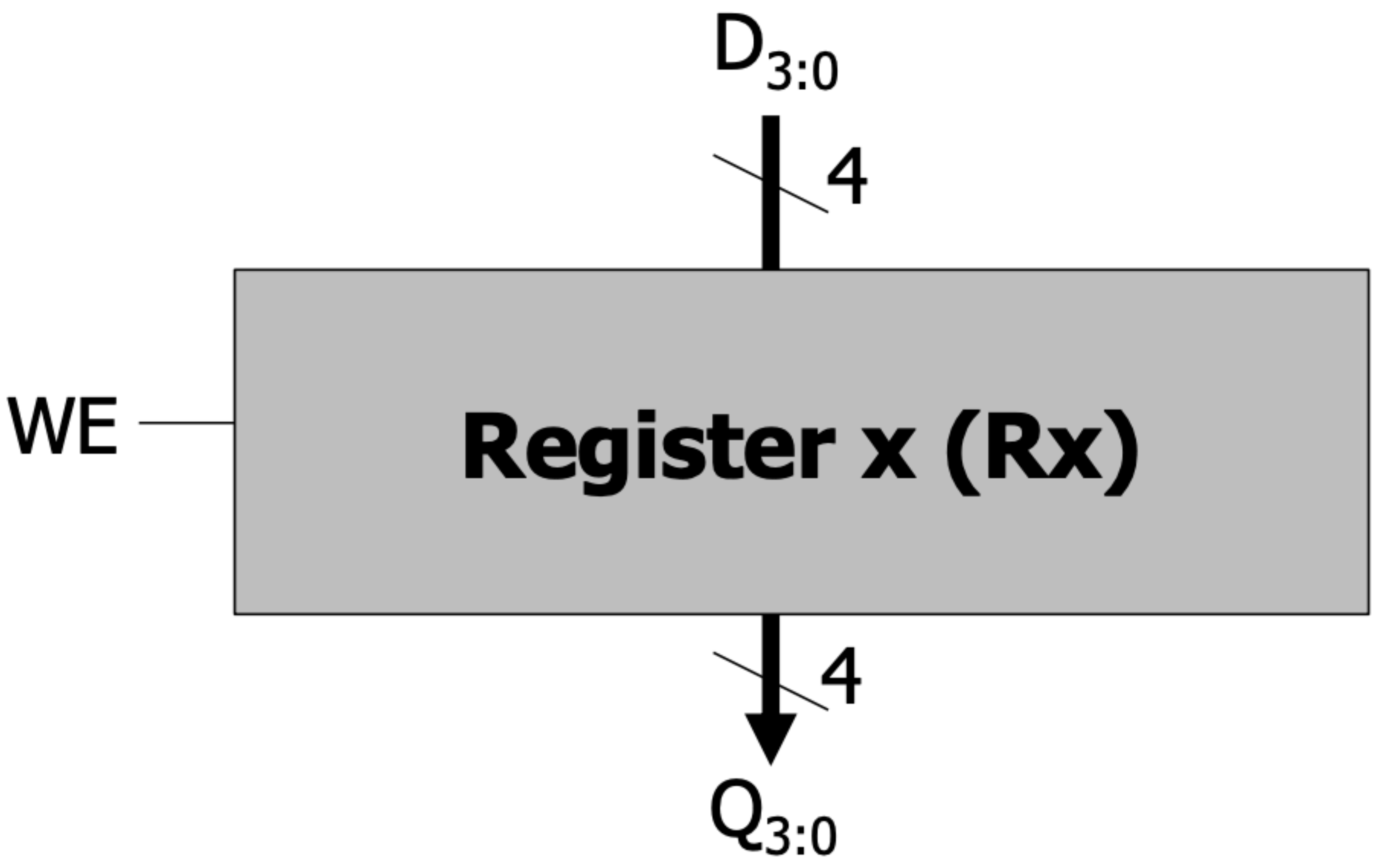
A negative edge-triggered *D* flip-flop:

- 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



Registers: Your Main Sequential Element





- Used to store data
- Basically an **array of D-flip-flops**
- You can **load data**, **reset it to zero**, and **shift it to left and right**

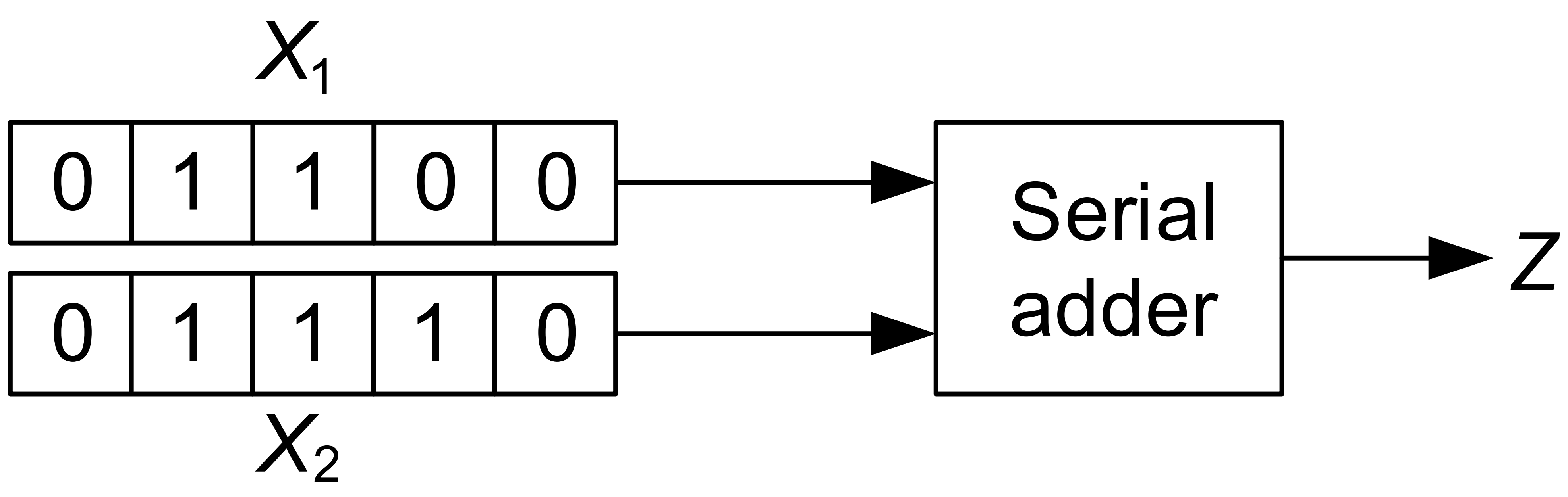
Sequential Circuits and Finite State Machines

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.

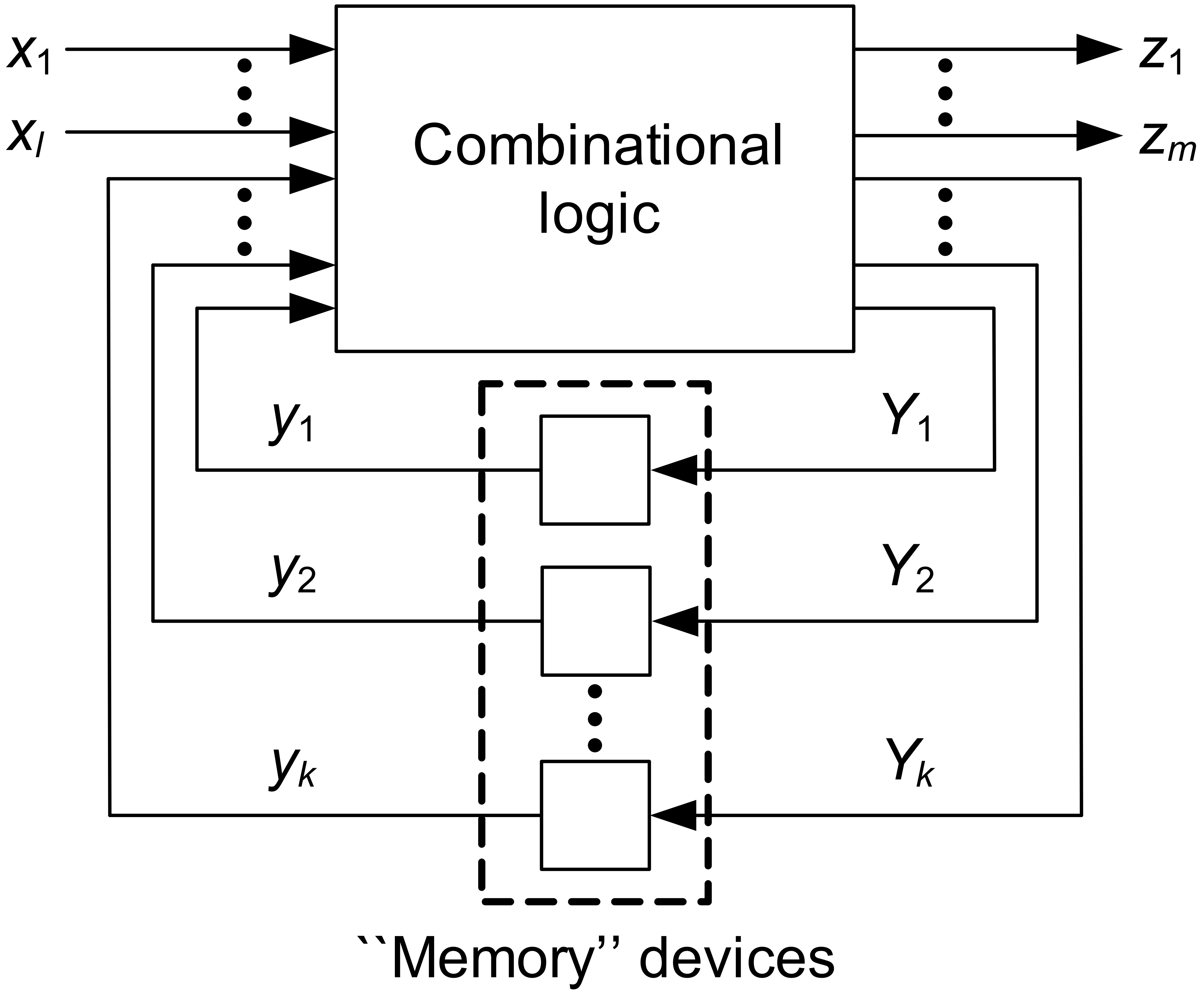
Serial binary adder example: You are given a 1-bit adder. But you have to add n-bit numbers

- First, decide what to remember??
- Then decide how many bits to remember??



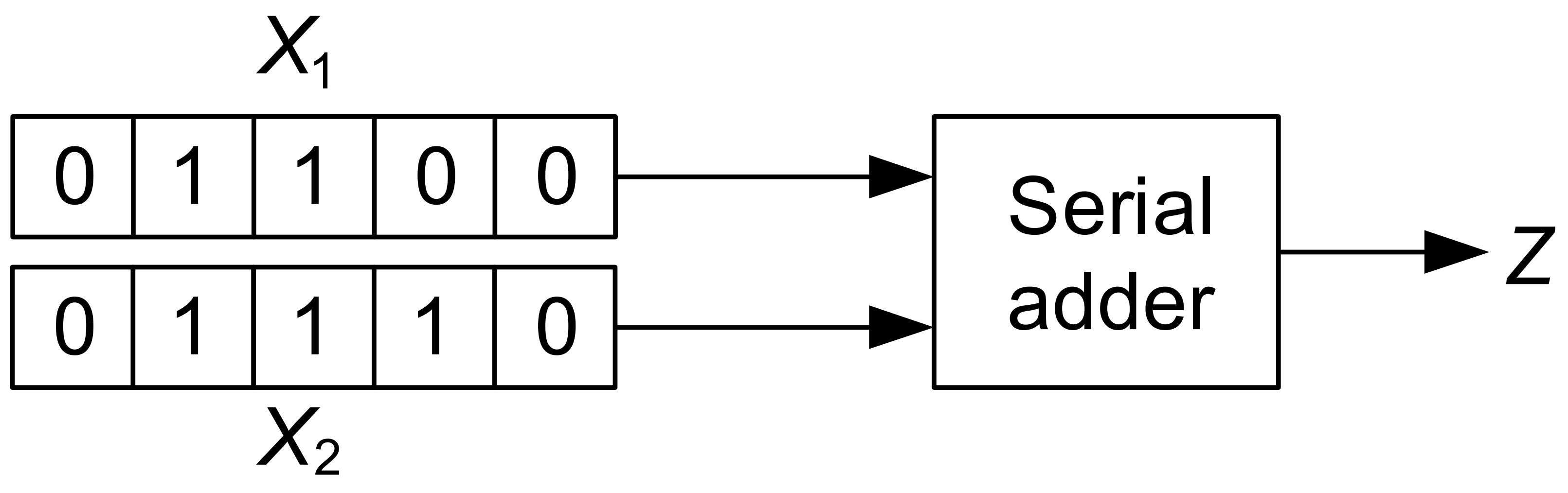
$$
\begin{array}{rccccccc}
 & t_5 & t_4 & t_3 & t_2 & t_1 & & \\
 & 0 & 1 & 1 & 0 & 0 & = & X_1 \\
+ & 0 & 1 & 1 & 1 & 0 & = & X_2 \\
\hline
 & 1 & 1 & 0 & 1 & 0 & = & Z
\end{array}
$$

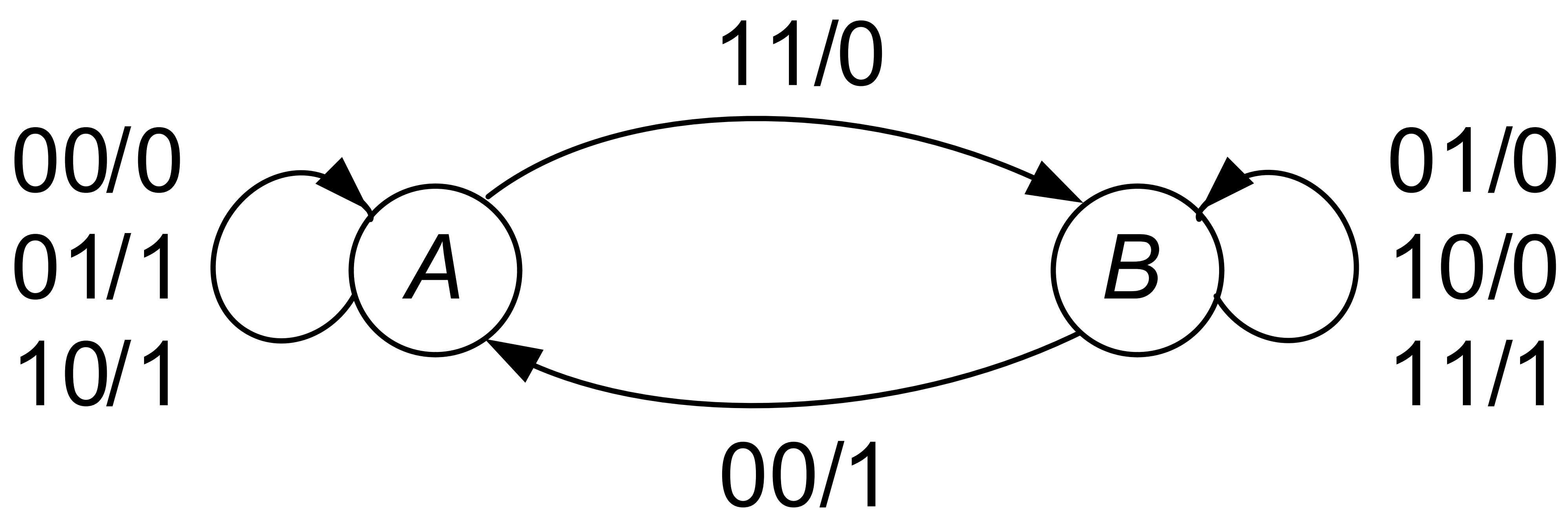
Sequential Circuits and Finite State Machines



Sequential Circuits and Finite State Machines

- 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}

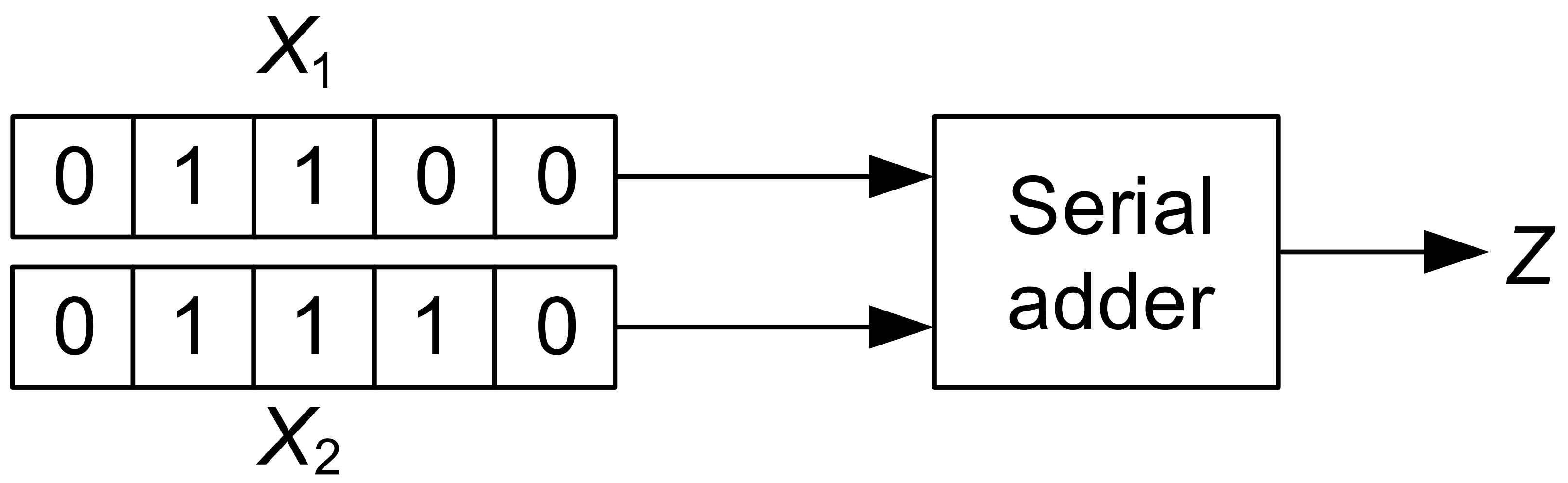




$$
\begin{array}{rccccccc}
 & t_5 & t_4 & t_3 & t_2 & t_1 & & \\
 & 0 & 1 & 1 & 0 & 0 & = & X_1 \\
+ & 0 & 1 & 1 & 1 & 0 & = & X_2 \\
\hline
 & 1 & 1 & 0 & 1 & 0 & = & Z
\end{array}
$$

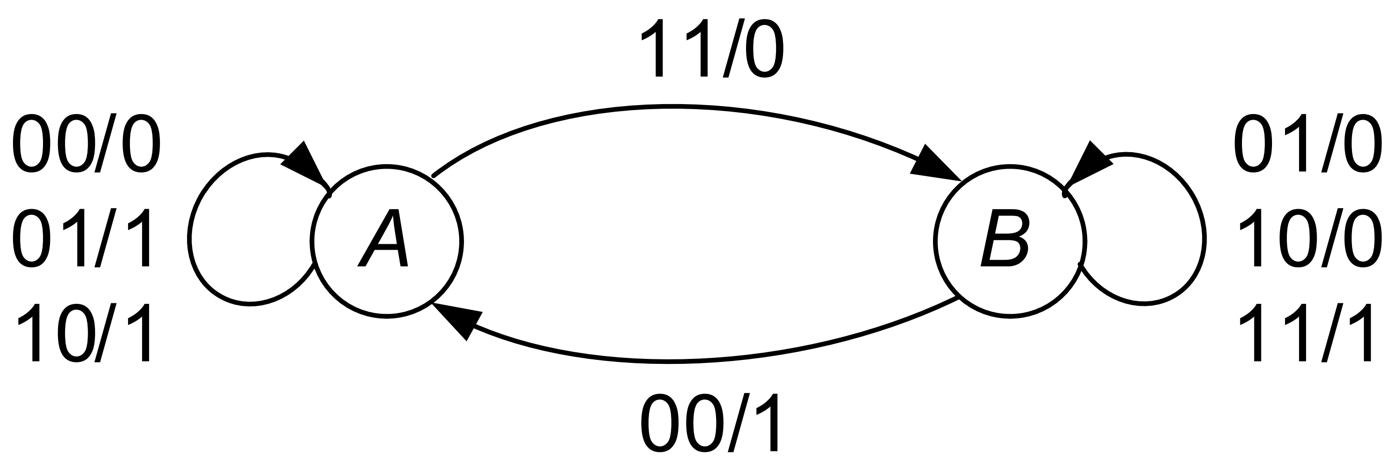
Sequential Circuits and Finite State Machines

- 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}



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

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

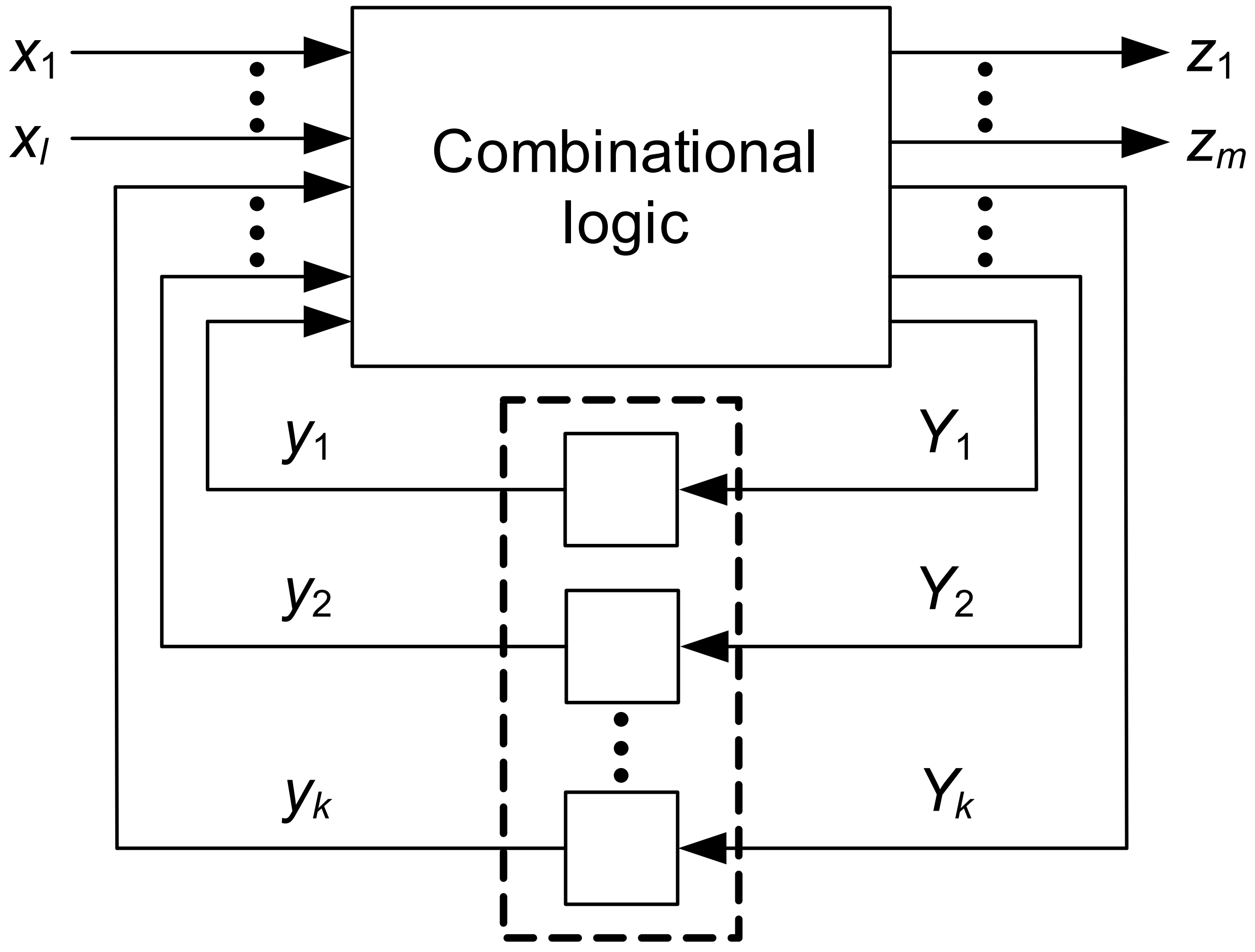


Sequential Circuits and Finite State Machines

FSMs: whose past histories can affect their future behavior in only a finite number of ways

- **Serial adder**: its response to the signals at time *t* is only a function of these signals and the value of the carry at *t*-1
  - Thus, its input histories can be grouped into just two classes: those resulting in a 1 carry and those resulting in a 0 carry at *t*
- **Thus, every finite-state machine contains a finite number of memory devices**: which store the information regarding the past input history

Sequential Circuits and Finite State Machines



Input variables: $\{x_1, x_2, .., x_l\}$

Input configuration, symbol, pattern or vector: ordered l-tuple of 0's and 1's

Input alphabet: set of $p = 2^l$ distinct input patterns

- Thus, input alphabet $I = \{I_1, I_2, .., I_p\}$
- Example: for two variables x_1 and x_2
  - $I = \{00, 01, 10, 11\}$

Output variables: $\{z_1, z_2, .., z_m\}$

Output configuration, symbol, pattern or vector: ordered m-tuple of 0's and 1's

Output alphabet: set of $q = 2^m$ distinct output patterns

- Thus, output alphabet $O = \{O_1, O_2, .., O_q\}$

Synthesis of Synchronous Sequential Circuits

Main steps:

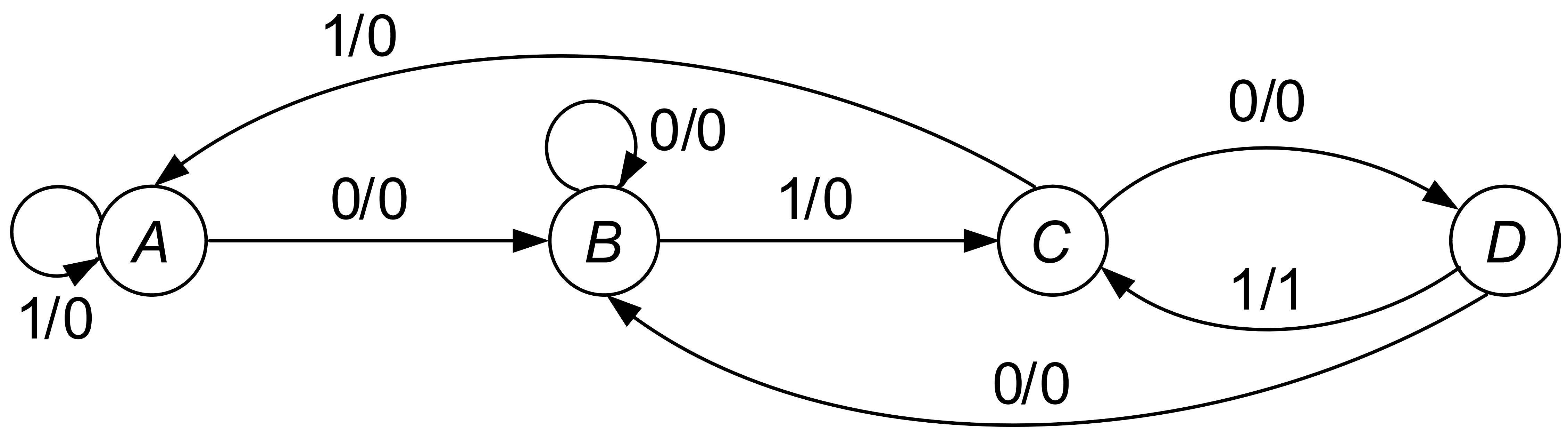
1. From a word description of the problem, form a **state diagram** or **table**
2. Select a **state assignment** and determine the type of memory elements
3. Derive **transition** and **output** tables
4. Derive an **excitation table** and obtain **excitation** and **output functions** from their respective tables
5. Draw a **circuit diagram**

Synthesis of Synchronous Sequential Circuits

One-input/one-output sequence detector: produces output value 1 every time sequence 0101 is detected, else 0

- Example: 010101 -> 000101

State diagram and state table:



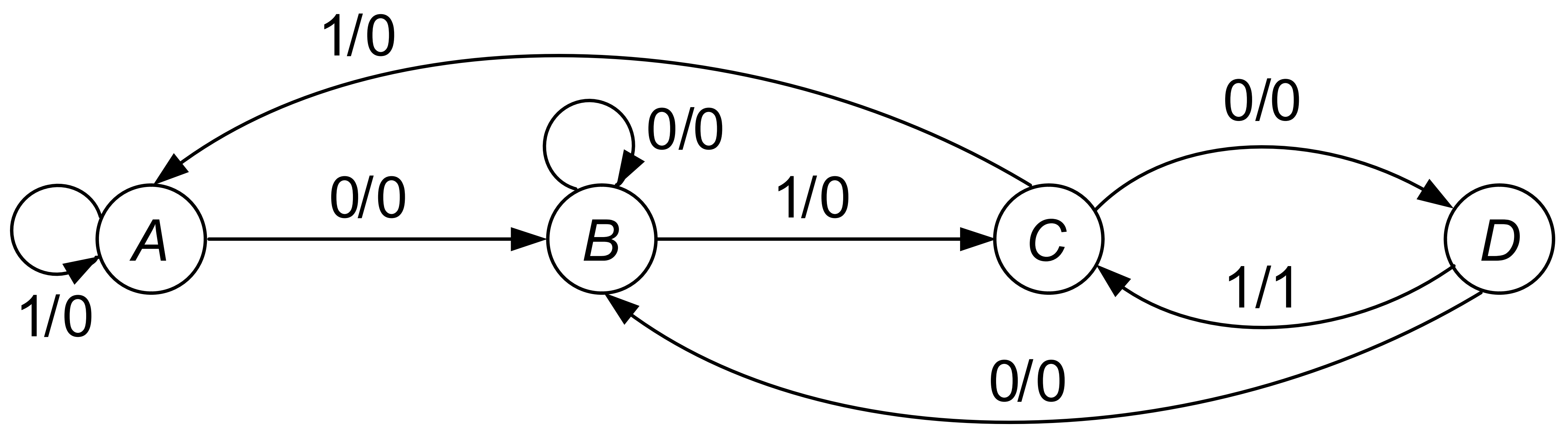
PS	NS, z $x=0$	$x=1$
A	$B, 0$	$A, 0$
B	$B, 0$	$C, 0$
C	$D, 0$	$A, 0$
D	$B, 0$	$C, 1$

Synthesis of Synchronous Sequential Circuits

One-input/one-output sequence detector: produces output value 1 every time sequence 0101 is detected, else 0

- Example: 010101 -> 000101

State diagram and state table:



PS	NS, z	
	$x = 0$	$x = 1$
A	$B, 0$	$A, 0$
B	$B, 0$	$C, 0$
C	$D, 0$	$A, 0$
D	$B, 0$	$C, 1$

Transition and output tables:

y_1y_2	Y_1Y_2		z	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
$A \rightarrow 00$	01	00	0	0
$B \rightarrow 01$	01	11	0	0
$C \rightarrow 11$	10	00	0	0
$D \rightarrow 10$	01	11	0	1

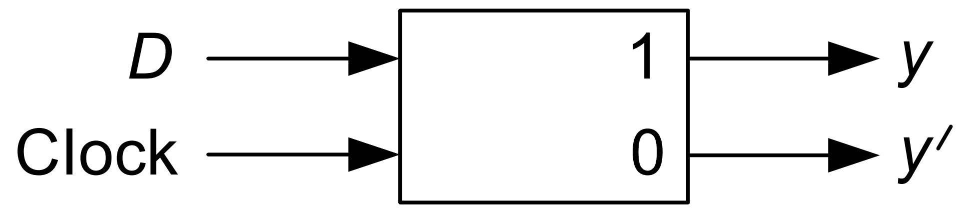
D Latch — The Latch of Your Life

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

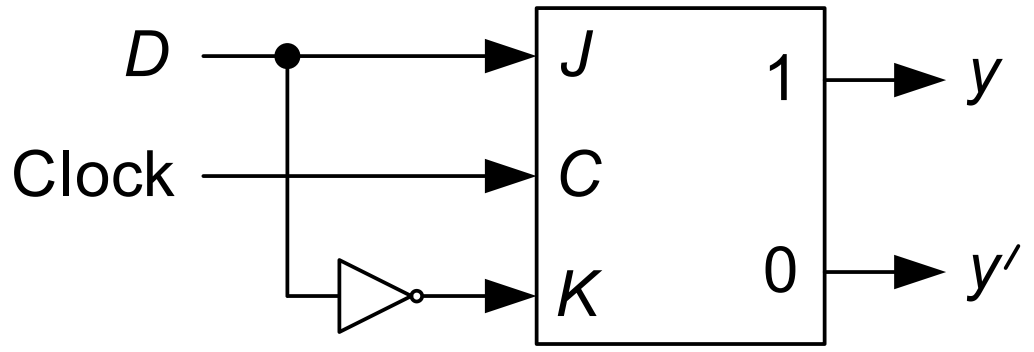
$$y(t+1) = D(t)$$

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

Synthesis of Synchronous Sequential Circuits

Excitation Table

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

- Let us use DFF as our state elements
- We need 2 DFFs as our state is 2 bit
- Now how to set the inputs of the DFFs??

y_1y_2	Y_1Y_2		z	
	$x=0$	$x=1$	$x=0$	$x=1$
$A \rightarrow 00$	01	00	0	0
$B \rightarrow 01$	01	11	0	0
$C \rightarrow 11$	10	00	0	0
$D \rightarrow 10$	01	11	0	1

Synthesis of Synchronous Sequential Circuits

- Let us use DFF as our state elements
- We need 2 DFFs as our state is 2 bit
- Now how to set the inputs of the DFFs??

Excitation Table

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

y_1y_2	Y_1Y_2 $x=0$	Y_1Y_2 $x=1$	z $x=0$	z $x=1$
$A \rightarrow 00$	01	00	0	0
$B \rightarrow 01$	01	11	0	0
$C \rightarrow 11$	10	00	0	0
$D \rightarrow 10$	01	11	0	1

D(Y1)		D(Y2)	
x = 0	x = 1	x=0	x=1
0	0	1	0
0	1	1	1
1	0	0	0
0	1	1	1

Synthesis of Synchronous Sequential Circuits

Excitation Table

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

y_1y_2 \ x	0	1
00		
01		1
11	1	
10		1

(b) Y_1 map.

y_1y_2	Y_1Y_2		z	
	$x=0$	$x=1$	$x=0$	$x=1$
$A \rightarrow 00$	01	00	0	0
$B \rightarrow 01$	01	11	0	0
$C \rightarrow 11$	10	00	0	0
$D \rightarrow 10$	01	11	0	1

D(Y1)		D(Y2)	
x = 0	x = 1	x=0	x=1
0	0	1	0
0	1	1	1
1	0	0	0
0	1	1	1

Synthesis of Synchronous Sequential Circuits

Excitation Table

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

y_1y_2 \ x	0	1
00		
01		1
11	1	
10		1

(*b*) Y_1 map.

y_1y_2 \ x	0	1
00	1	
01	1	1
11		
10	1	1

(*c*) Y_2 map.

y_1y_2	Y_1Y_2 $x=0$	Y_1Y_2 $x=1$	z $x=0$	z $x=1$
$A \rightarrow 00$	01	00	0	0
$B \rightarrow 01$	01	11	0	0
$C \rightarrow 11$	10	00	0	0
$D \rightarrow 10$	01	11	0	1

D(Y1)		D(Y2)	
x = 0	x = 1	x=0	x=1
0	0	1	0
0	1	1	1
1	0	0	0
0	1	1	1

Synthesis of Synchronous Sequential Circuits

Excitation Table

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

y_1y_2 \ x	0	1
00		
01		
11		
10		1

(*a*) *z* map.

y_1y_2 \ x	0	1
00		
01		1
11	1	
10		1

(*b*) Y_1 map.

y_1y_2 \ x	0	1
00	1	
01	1	1
11		
10	1	1

(*c*) Y_2 map.

y_1y_2	Y_1Y_2		z	
	$x=0$	$x=1$	$x=0$	$x=1$
$A \rightarrow 00$	01	00	0	0
$B \rightarrow 01$	01	11	0	0
$C \rightarrow 11$	10	00	0	0
$D \rightarrow 10$	01	11	0	1

D(Y1)		D(Y2)	
x = 0	x = 1	x=0	x=1
0	0	1	0
0	1	1	1
1	0	0	0
0	1	1	1

Synthesis of Synchronous Sequential Circuits

Excitation Table

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

y_1y_2 \ x	0	1
00		
01		
11		
10		1

(*a*) *z* map.

y_1y_2 \ x	0	1
00		
01		1
11	1	
10		1

(*b*) Y_1 map.

y_1y_2 \ x	0	1
00	1	
01	1	1
11		
10	1	1

(*c*) Y_2 map.

$z = xy_1y_2'$

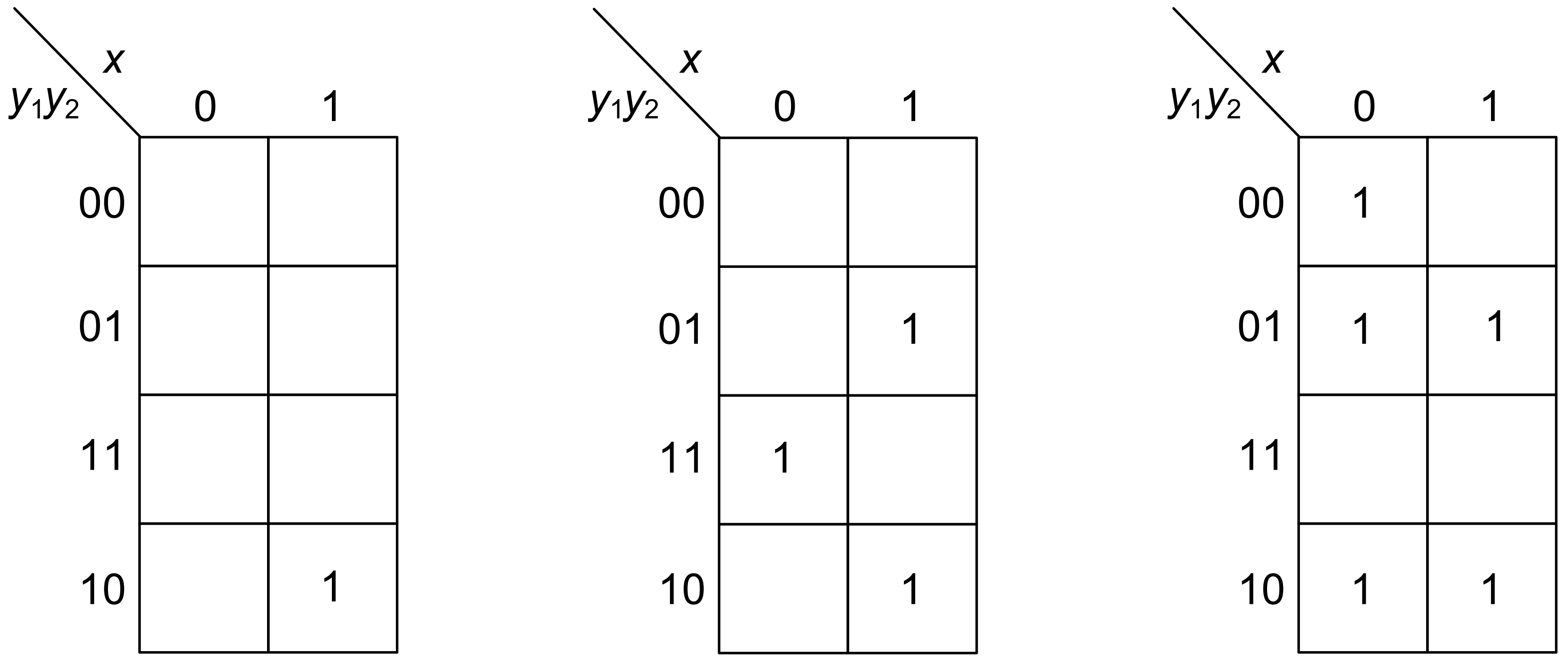
$Y_1 = x'y_1y_2 + xy_1'y_2 + xy_1y_2'$

$Y_2 = y_1y_2' + x'y_1' + y_1'y_2$

y_1y_2	Y_1Y_2, $x=0$	Y_1Y_2, $x=1$	z, $x=0$	z, $x=1$
$A \rightarrow 00$	01	00	0	0
$B \rightarrow 01$	01	11	0	0
$C \rightarrow 11$	10	00	0	0
$D \rightarrow 10$	01	11	0	1

D(Y1)		D(Y2)	
x = 0	x = 1	x=0	x=1
0	0	1	0
0	1	1	1
1	0	0	0
0	1	1	1

Synthesis of Synchronous Sequential Circuits



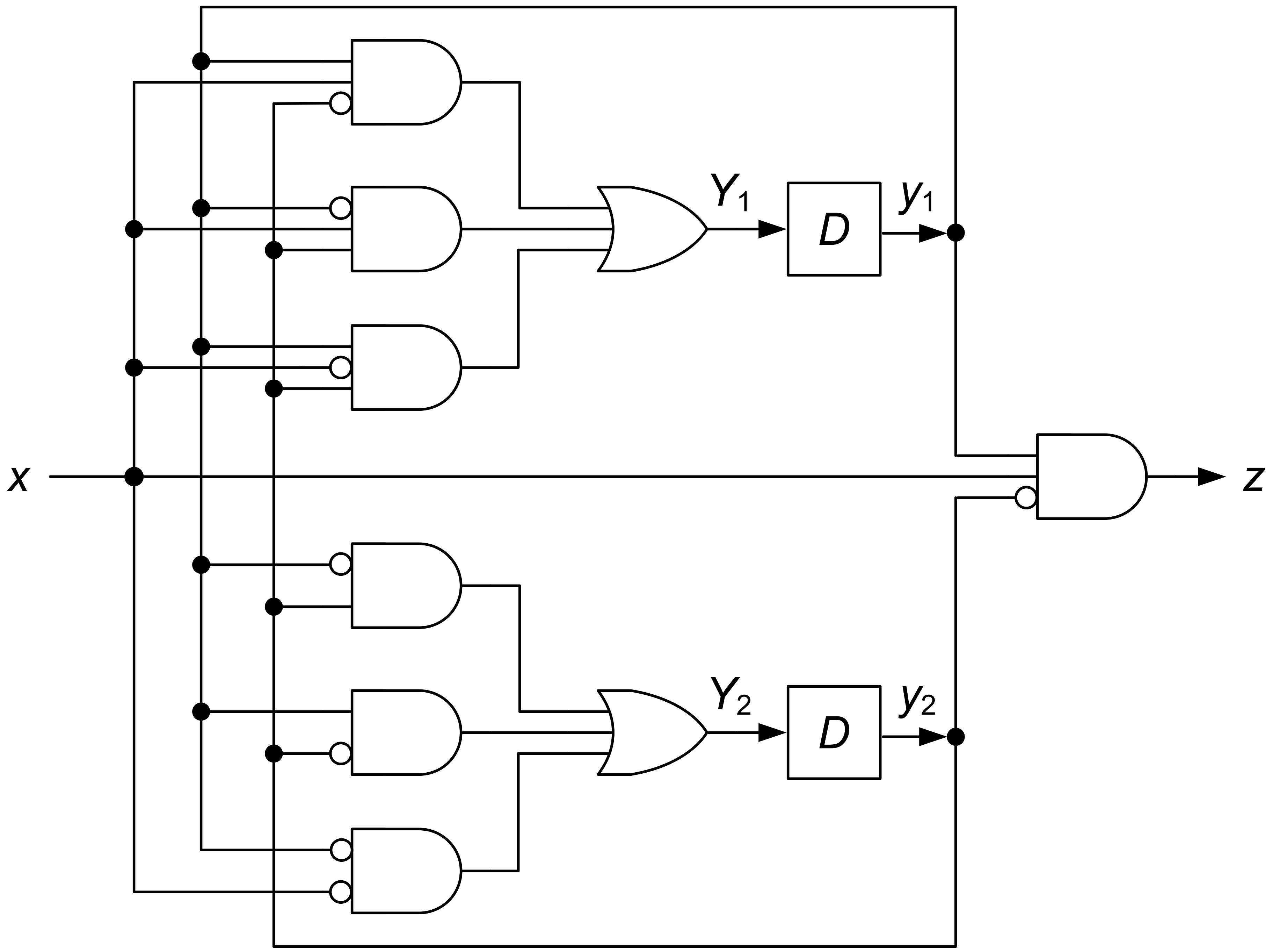
(a) z map.   (b) Y_1 map.   (c) Y_2 map.

$$z = xy_1y_2'$$
$$Y_1 = x'y_1y_2 + xy_1'y_2 + xy_1y_2'$$
$$Y_2 = y_1y_2' + x'y_1' + y_1'y_2$$

Excitation Table

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

Logic Diagram



y_1y_2	Y_1Y_2 $x=0$	Y_1Y_2 $x=1$	z $x=0$	z $x=1$
$A \rightarrow 00$	01	00	0	0
$B \rightarrow 01$	01	11	0	0
$C \rightarrow 11$	10	00	0	0
$D \rightarrow 10$	01	11	0	1

D(Y1)		D(Y2)	
x = 0	x = 1	x=0	x=1
0	0	1	0
0	1	1	1
1	0	0	0
0	1	1	1

Synthesis of Synchronous Sequential Circuits

Another state assignment:

y_1y_2	Y_1Y_2 $x=0$	Y_1Y_2 $x=1$	z $x=0$	z $x=1$
$A \to 00$	01	00	0	0
$B \to 01$	01	10	0	0
$C \to 10$	11	00	0	0
$D \to 11$	01	10	0	1

$z = xy_1y_2$

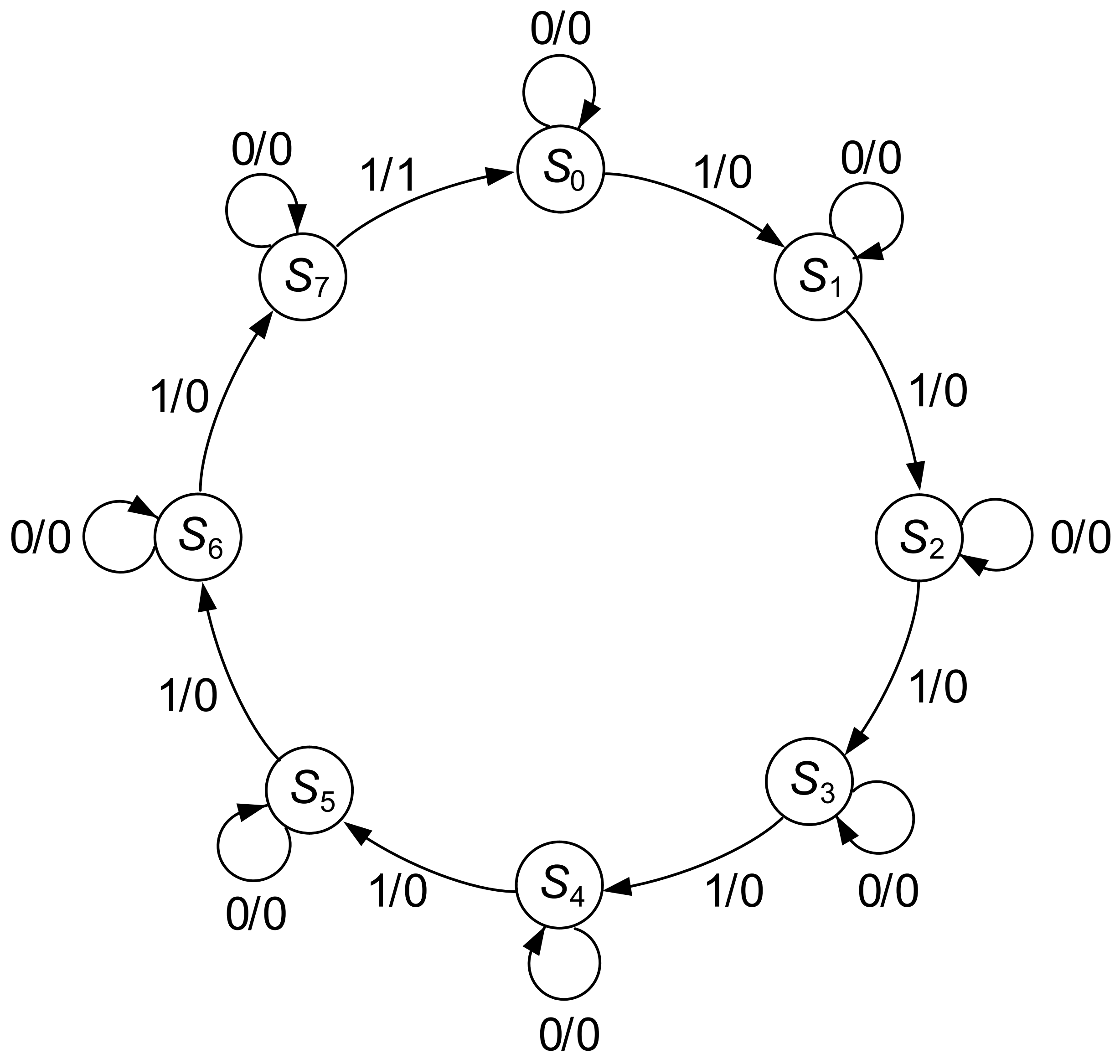
$Y_1 = x'y_1y_2' + xy_2$

$Y_2 = x'$

Binary Counter

One-input/one-output modulo-8 binary counter: produces output value 1 for every eighth input 1 value

State diagram and state table:



PS	NS $x=0$	NS $x=1$	$Output$ $x=0$	$Output$ $x=1$
S_0	S_0	S_1	0	0
S_1	S_1	S_2	0	0
S_2	S_2	S_3	0	0
S_3	S_3	S_4	0	0
S_4	S_4	S_5	0	0
S_5	S_5	S_6	0	0
S_6	S_6	S_7	0	0
S_7	S_7	S_0	0	1

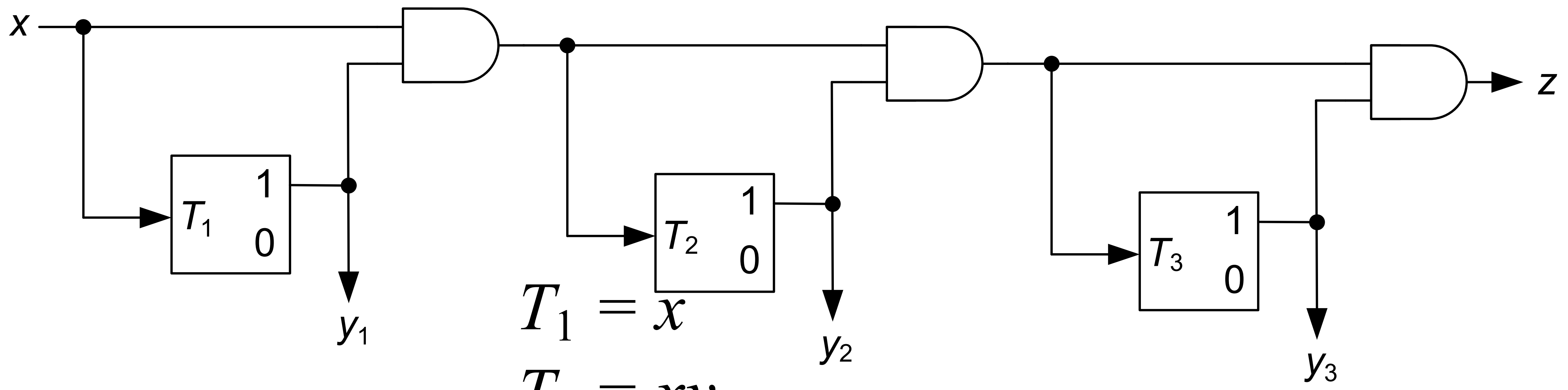
Binary Counter

Transition and output tables:

PS	NS		z		$T_3T_2T_1$	
$y_3y_2y_1$	$x=0$	$x=1$	$x=0$	$x=1$	$x=0$	$x=1$
000	000	001	0	0	000	001
001	001	010	0	0	000	011
010	010	011	0	0	000	001
011	011	100	0	0	000	111
100	100	101	0	0	000	001
101	101	110	0	0	000	011
110	110	111	0	0	000	001
111	111	000	0	1	000	111

Excitation table for *T*

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



$$T_1 = x$$
$$T_2 = xy_1$$
$$T_3 = xy_1y_2$$
$$z = xy_1y_2y_3$$

Binary Counter with SR Flip Flops

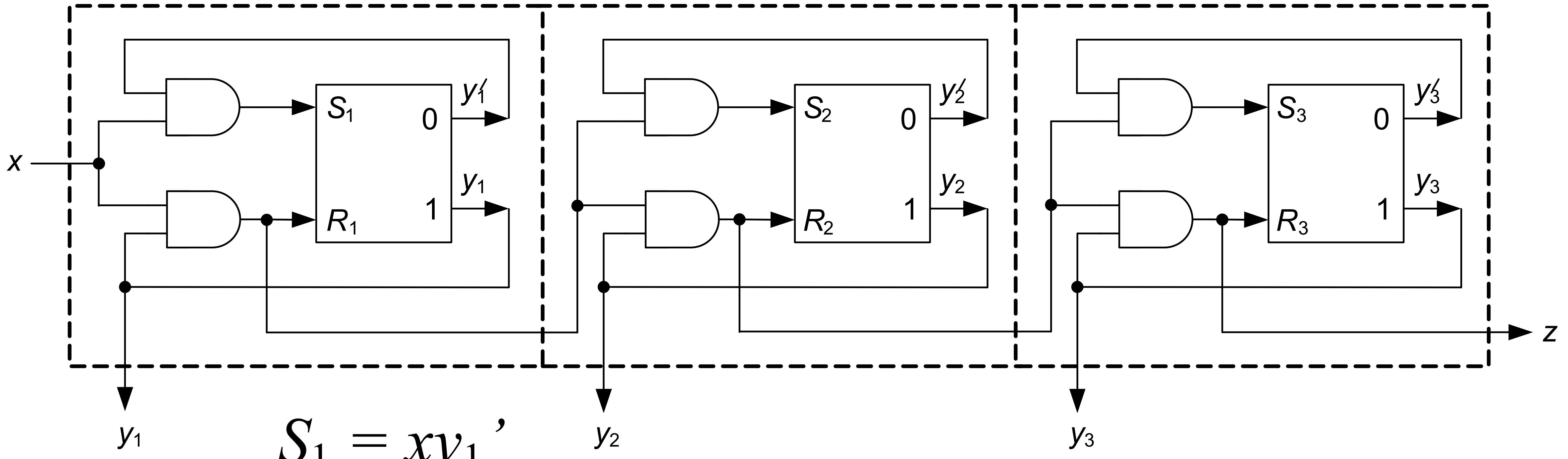
Transition and output tables:

PS	NS		z	
$y_3y_2y_1$	$x=0$	$x=1$	$x=0$	$x=1$
000	000	001	0	0
001	001	010	0	0
010	010	011	0	0
011	011	100	0	0
100	100	101	0	0
101	101	110	0	0
110	110	111	0	0
111	111	000	0	1

$x=0$			$x=1$		
S_3R_3	S_2R_2	S_1R_1	S_3R_3	S_2R_2	S_1R_1
0–	0–	0–	0–	0–	10
0–	0–	–0	0–	10	01
0–	–0	0–	0–	–0	10
0–	–0	–0	10	01	01
–0	0–	0–	–0	0–	10
–0	0–	–0	–0	10	01
–0	–0	0–	–0	–0	10
–0	–0	–0	01	01	01

Excitation table for *SR* flip-flops and logic diagram:

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



$$S_1 = xy_1'$$
$$R_1 = xy_1$$
$$S_2 = xy_1y_2'$$
$$R_2 = xy_1y_2$$
$$S_3 = xy_1y_2y_3'$$
$$R_3 = z = xy_1y_2y_3$$

But…Life is Beautiful End of the day…

```
always @(posedge clk) begin
    if (rst)
        count <= 3'b000;        // Reset to 0
    else if (count == 3'b111)   // If 7, wrap back to 0
        count <= 3'b000;
    else
        count <= count + 1'b1;  // Increment
end
```