18.03.21 12:00

CSCI 150
Introduction to Digital and Computer System Design
Lecture 4: Sequential Circuit II

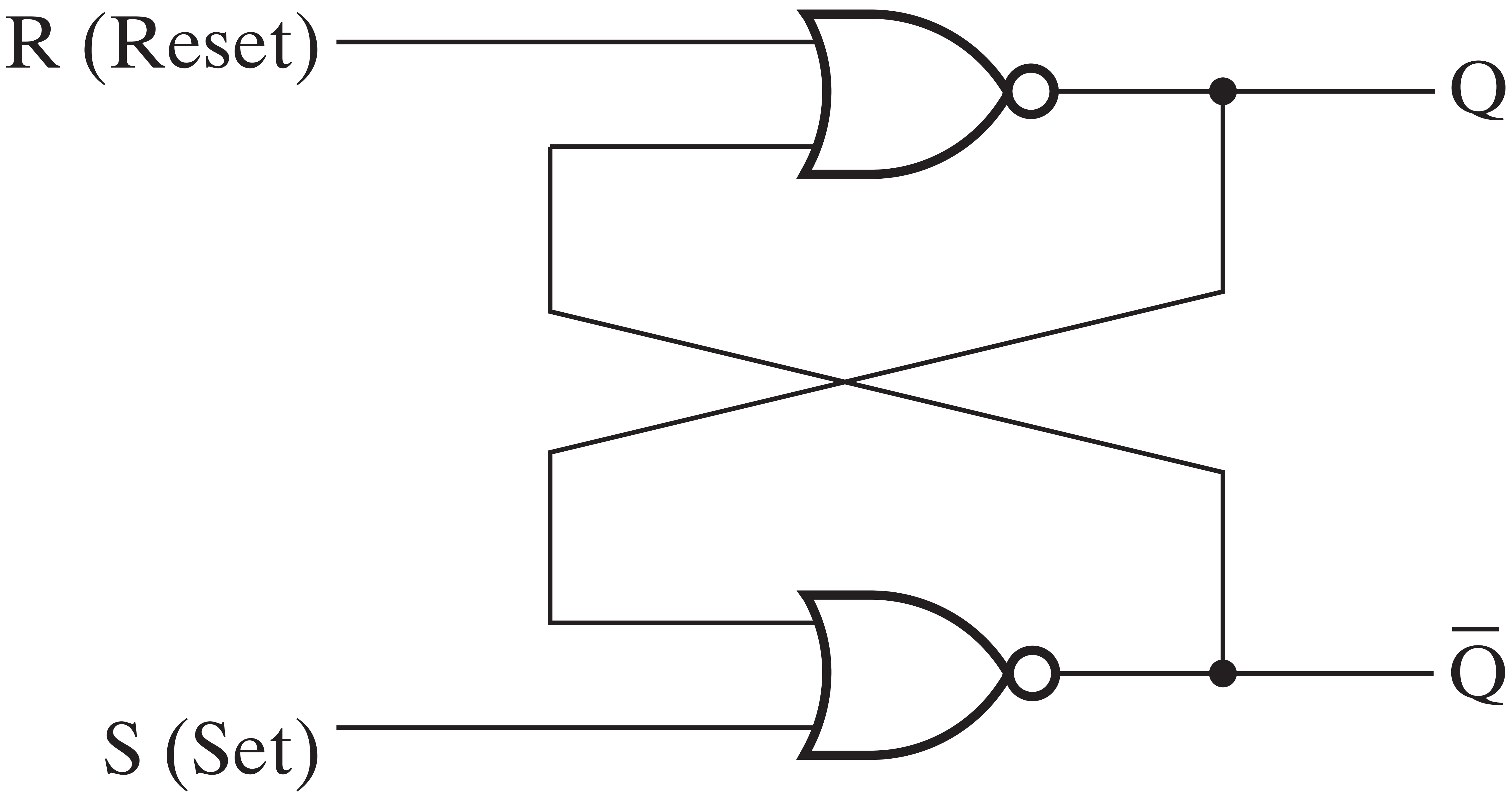
Jetic Gū

Overview

- Focus: Basic Information Retaining Blocks
- Architecture: Sequential Circuit
- Textbook v4: Ch5 5.2, 5.3; v5: Ch4 4.2, 5.3
- Core Ideas:
    1. Flip-Flops

P0
Review

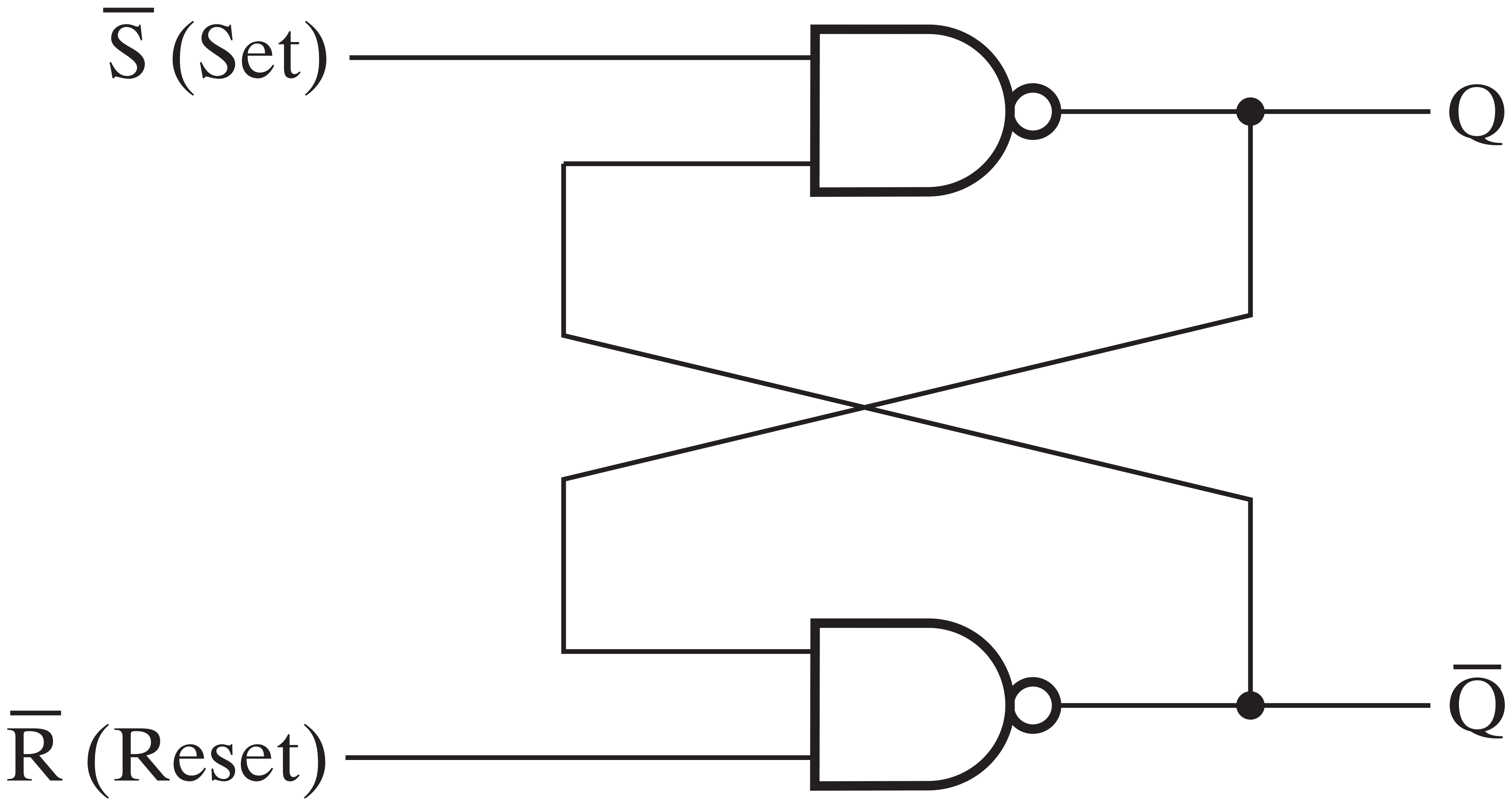
SR Latch



Demo

P0
Review

$\overline{SR}$ Latch



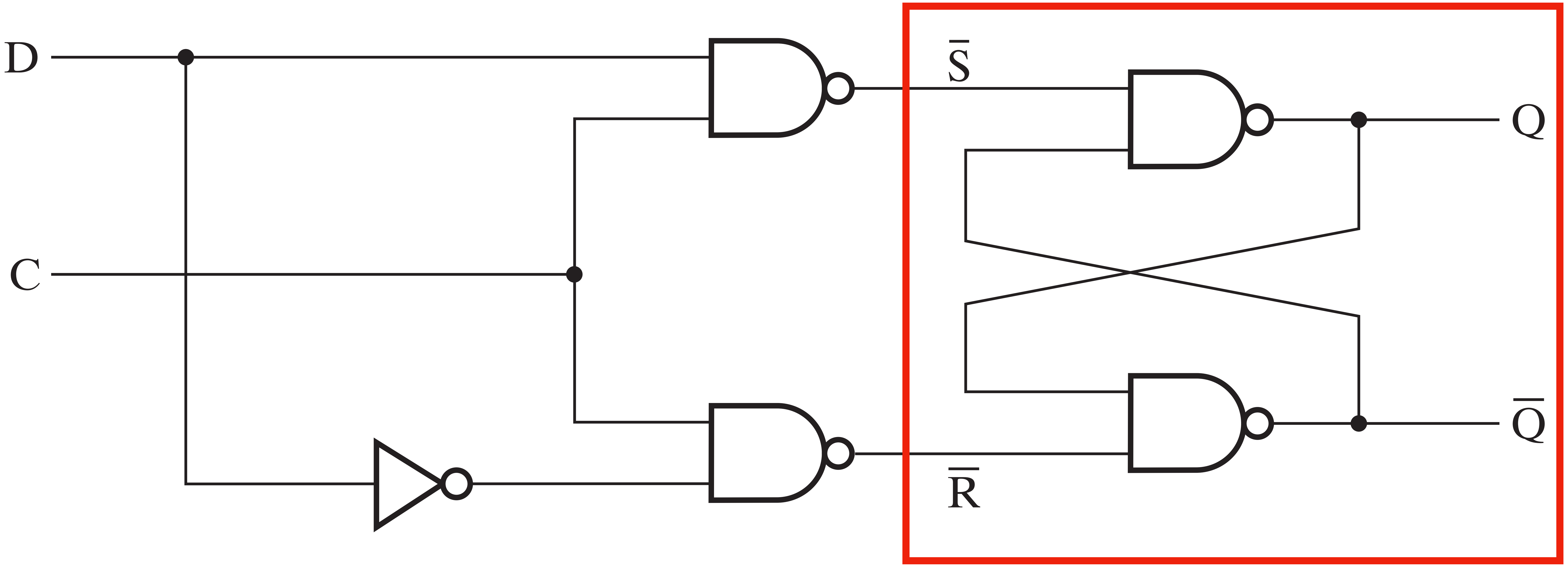
$\overline{\text{S}}$	$\overline{\text{R}}$	Q	$\overline{\text{Q}}$	
0	1	1	0	Set state
1	1	1	0	
1	0	0	1	Reset state
1	1	0	1	
0	0	1	1	Undefined

- Design similar to SR latches, but with NANDS
- Functions equivalent to SR latches with S and R inverted

Review

P0
Review

D Latch



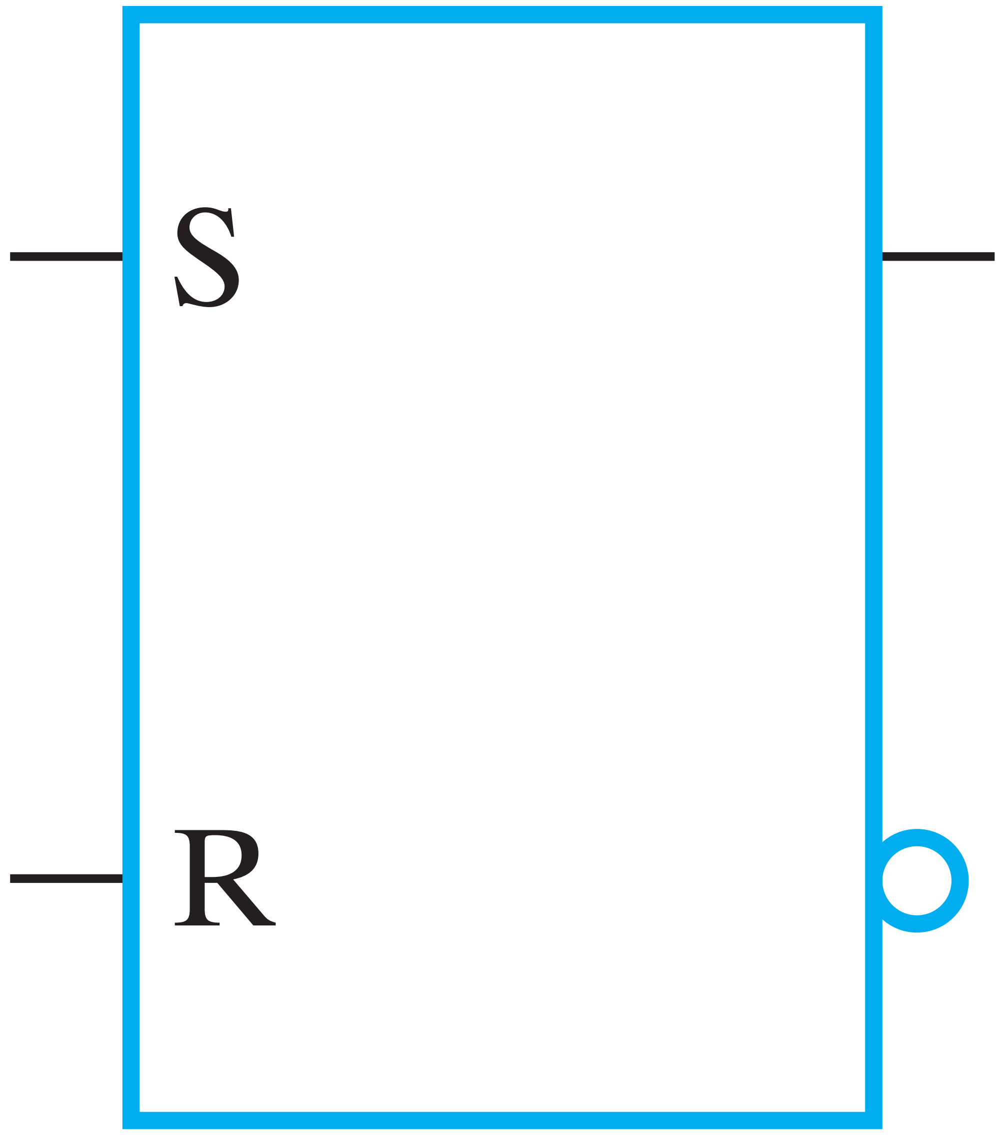
C	D	Next state of Q
0	X	No change
1	0	Q = 0; Reset state
1	1	Q = 1; Set state

- Implemented using $\overline{SR}$ latches
- C: Signals changes to the stored states; D the value to change to

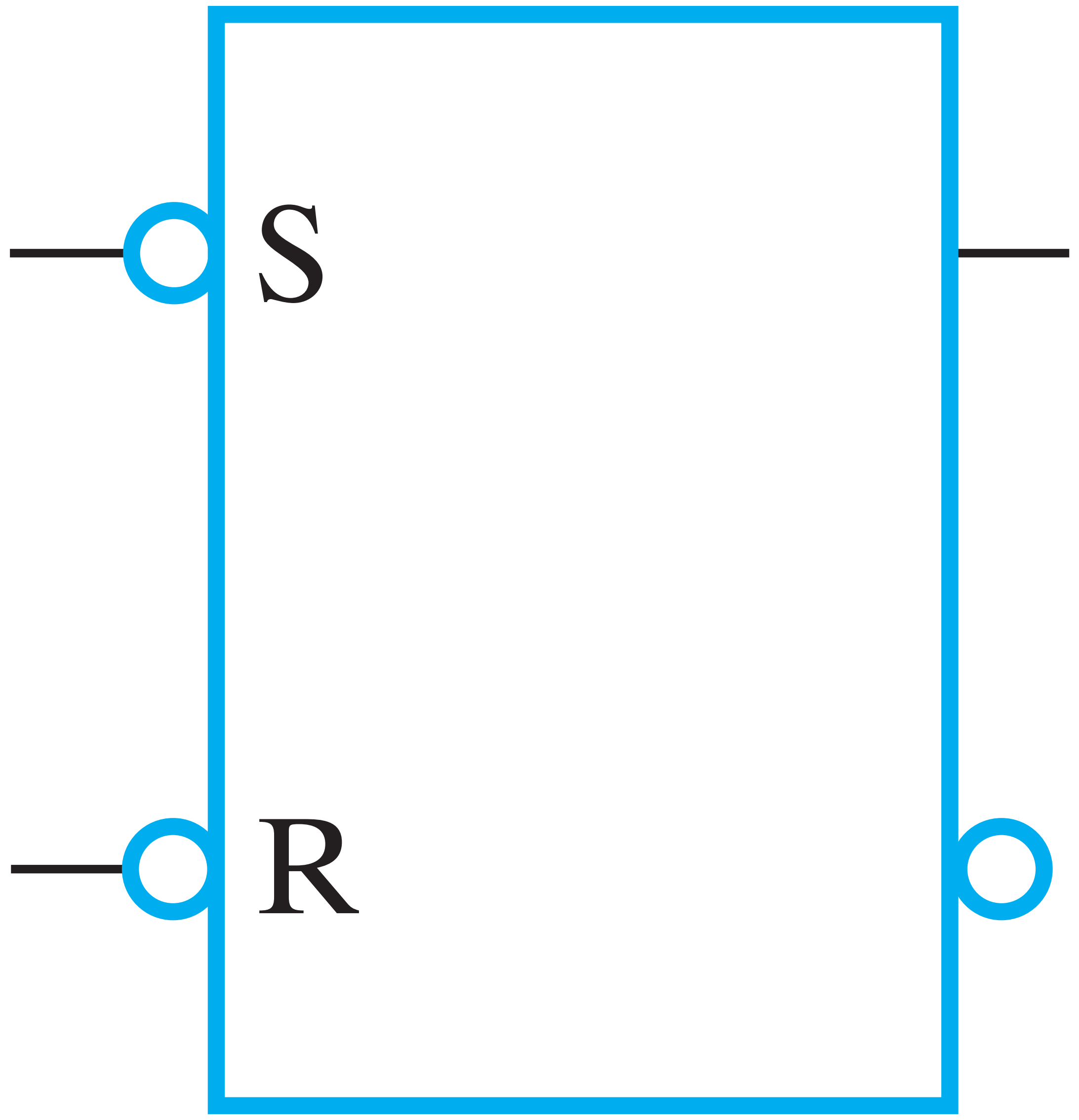
Review

P0
Review

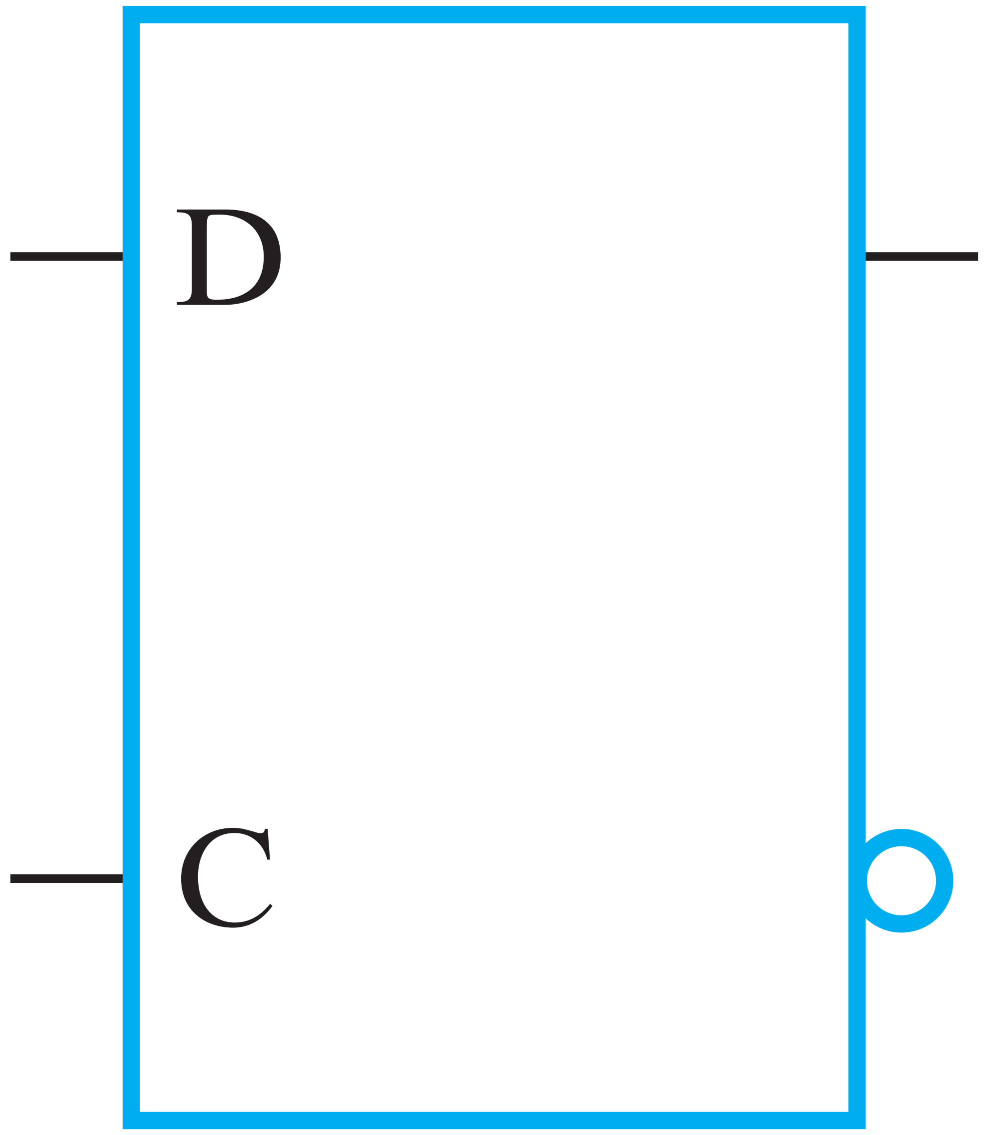
Latches



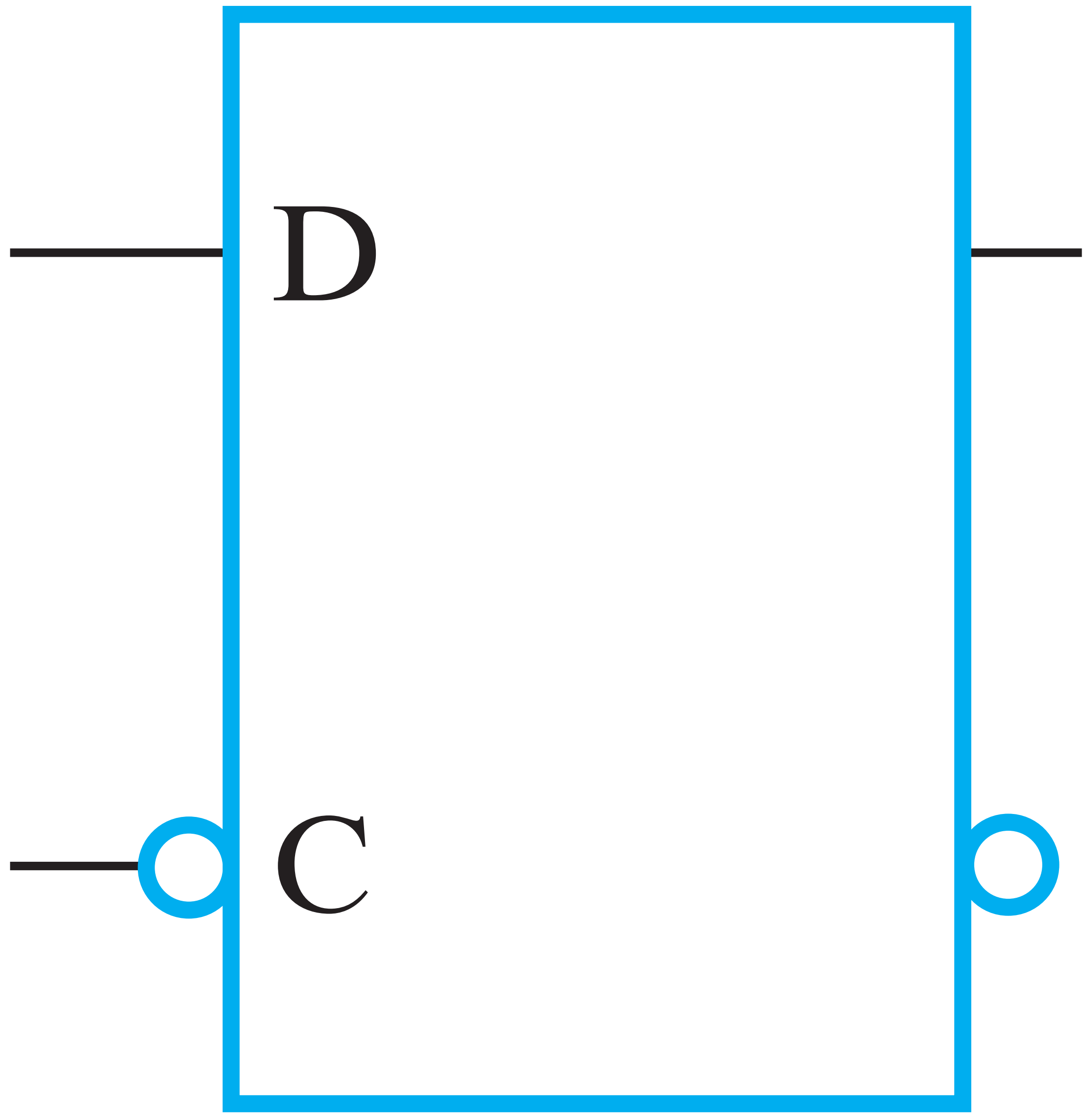
SR



$\overline{S}\overline{R}$



D with 1 Control



D with 0 Control

Concept

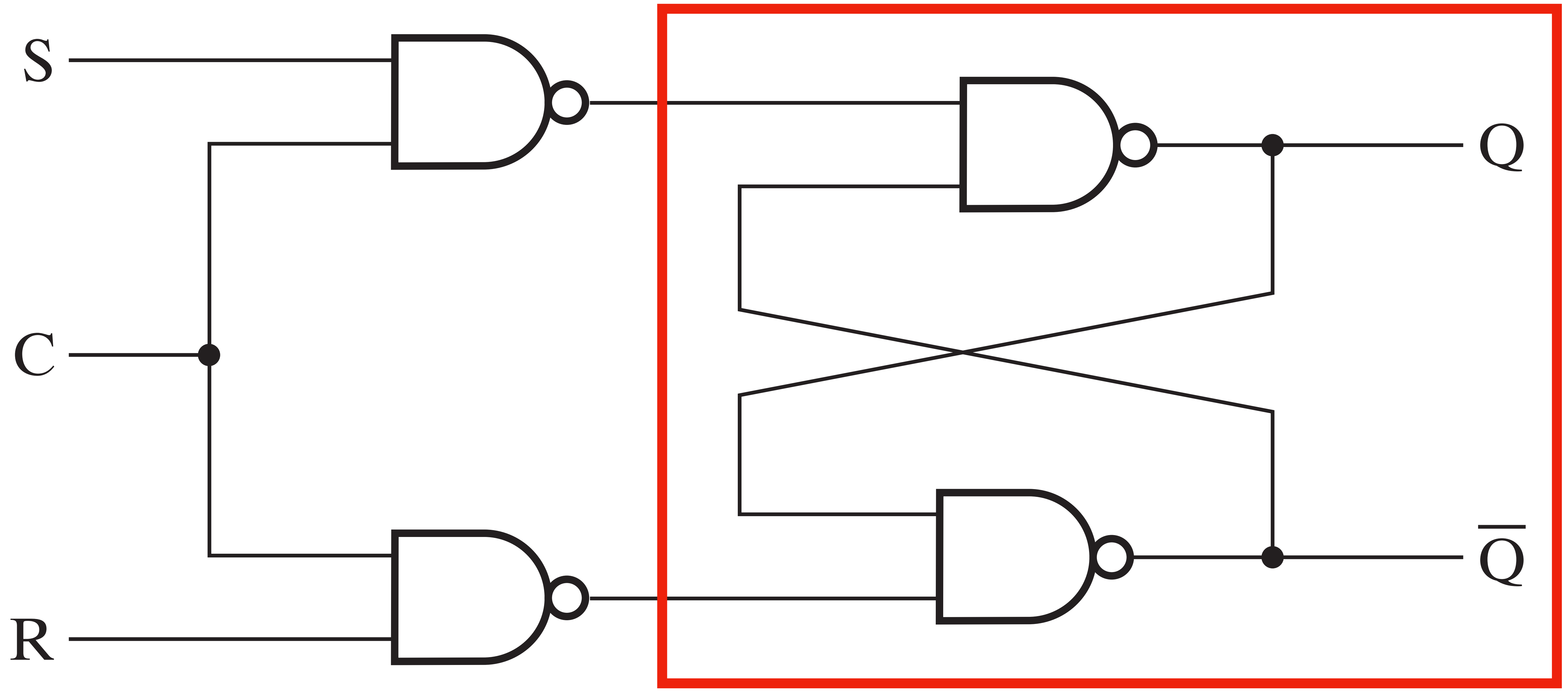
Flip-Flops

No, flip-flops are not proper shoes, nor shoes

Summary

P1
Flip-Flops

SR Latch with Control Input

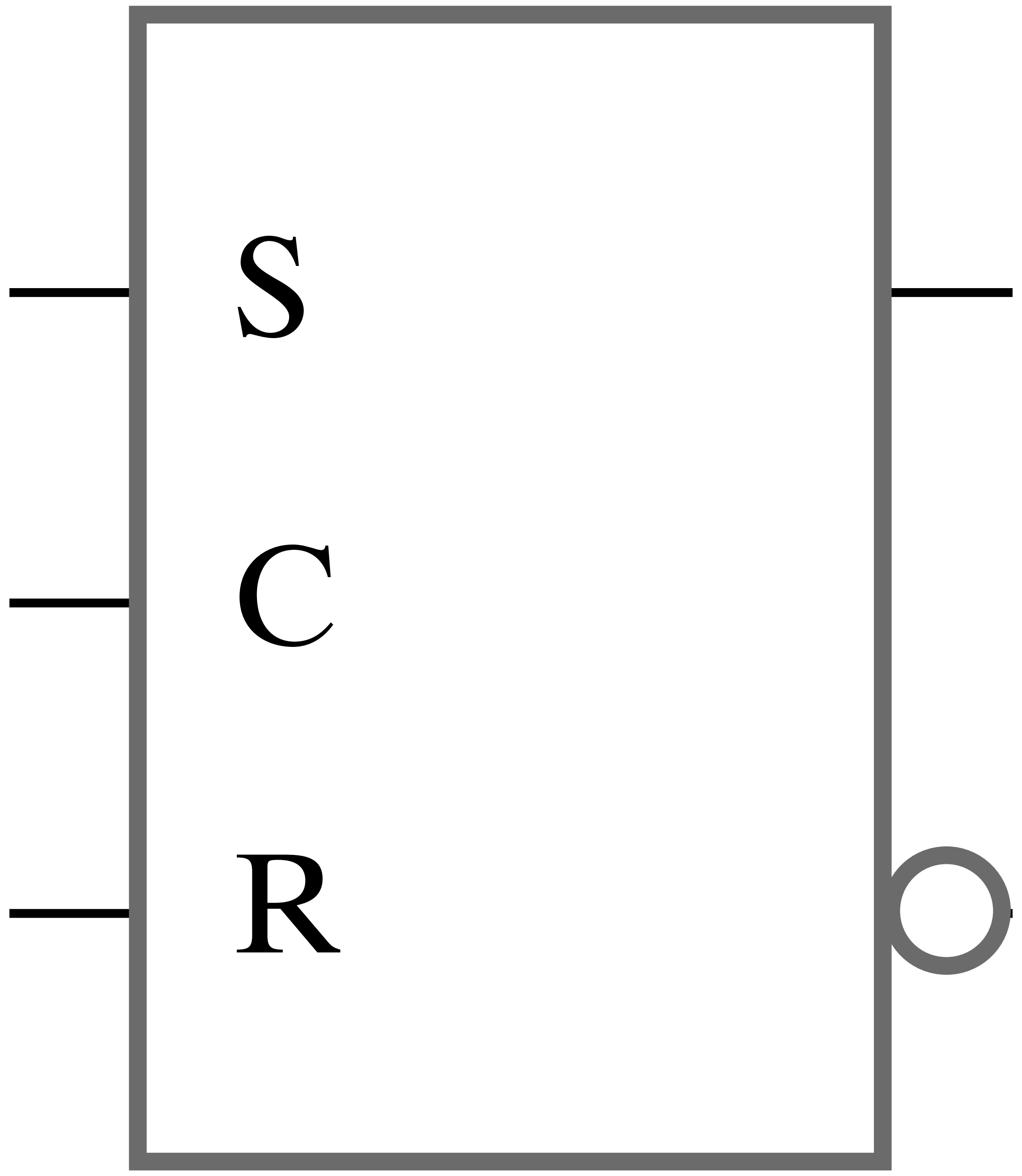


C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	Q = 0; Reset state
1	1	0	Q = 1; Set state
1	1	1	Undefined

- Implemented using $\overline{SR}$ latches
- C acts as an enabler; otherwise the entire circuit functions as an SR latch

Concept

SR Latch with Control Input



C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	Q = 0; Reset state
1	1	0	Q = 1; Set state
1	1	1	Undefined

- Implemented using $\overline{SR}$ latches
- C acts as an enabler; otherwise the entire circuit functions as an SR latch

Concept

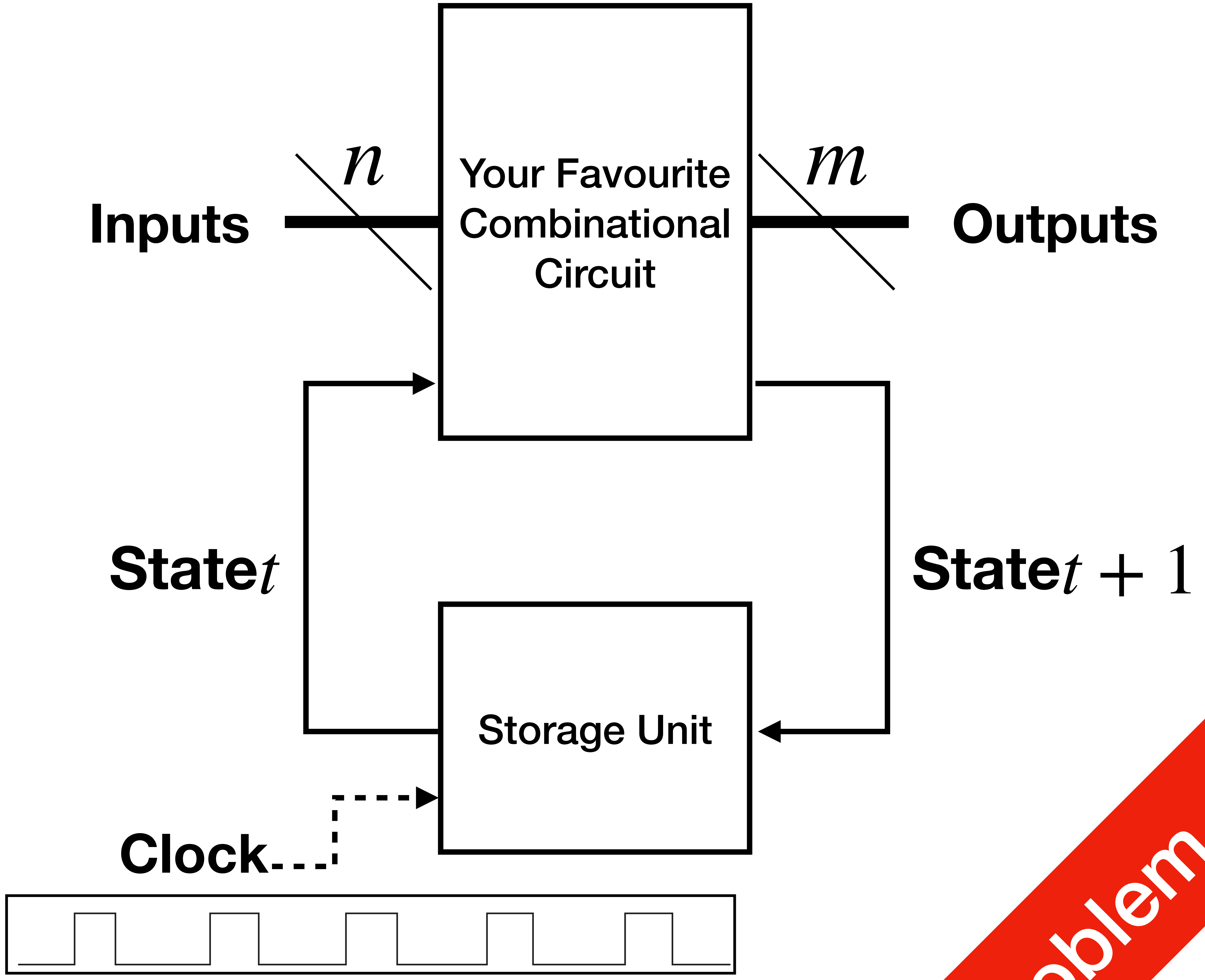
Latches

- What happens if the control pulse remains active?
  - any changes in the data input will change the state of the latch immediately!
- latches are **transparent**
  input can be seen from outputs while control pulse is 1

Concept

P1
Flip-Flops

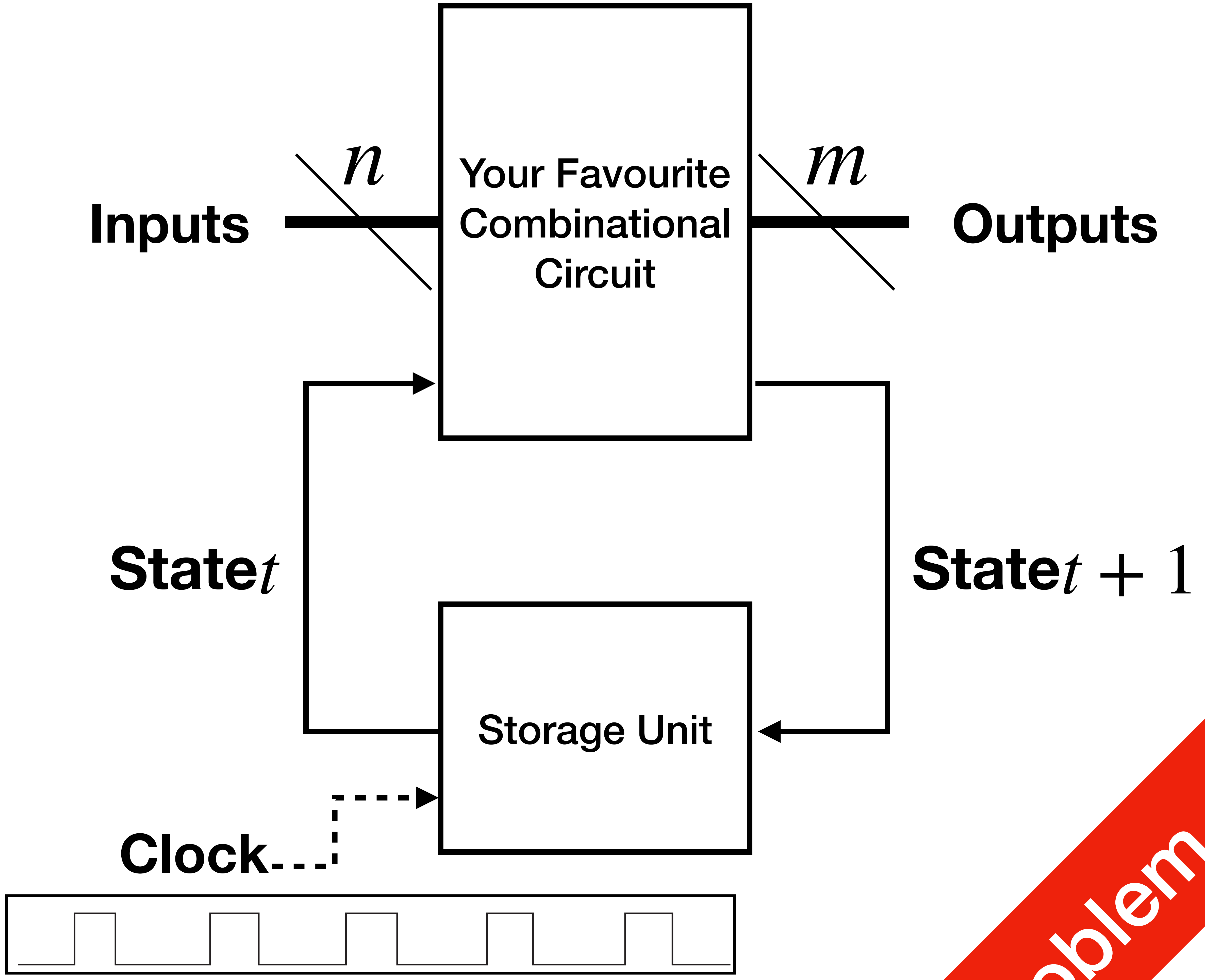
Problems with Latches



Problem

Problems with Latches

- Transparent: changes happen instantly

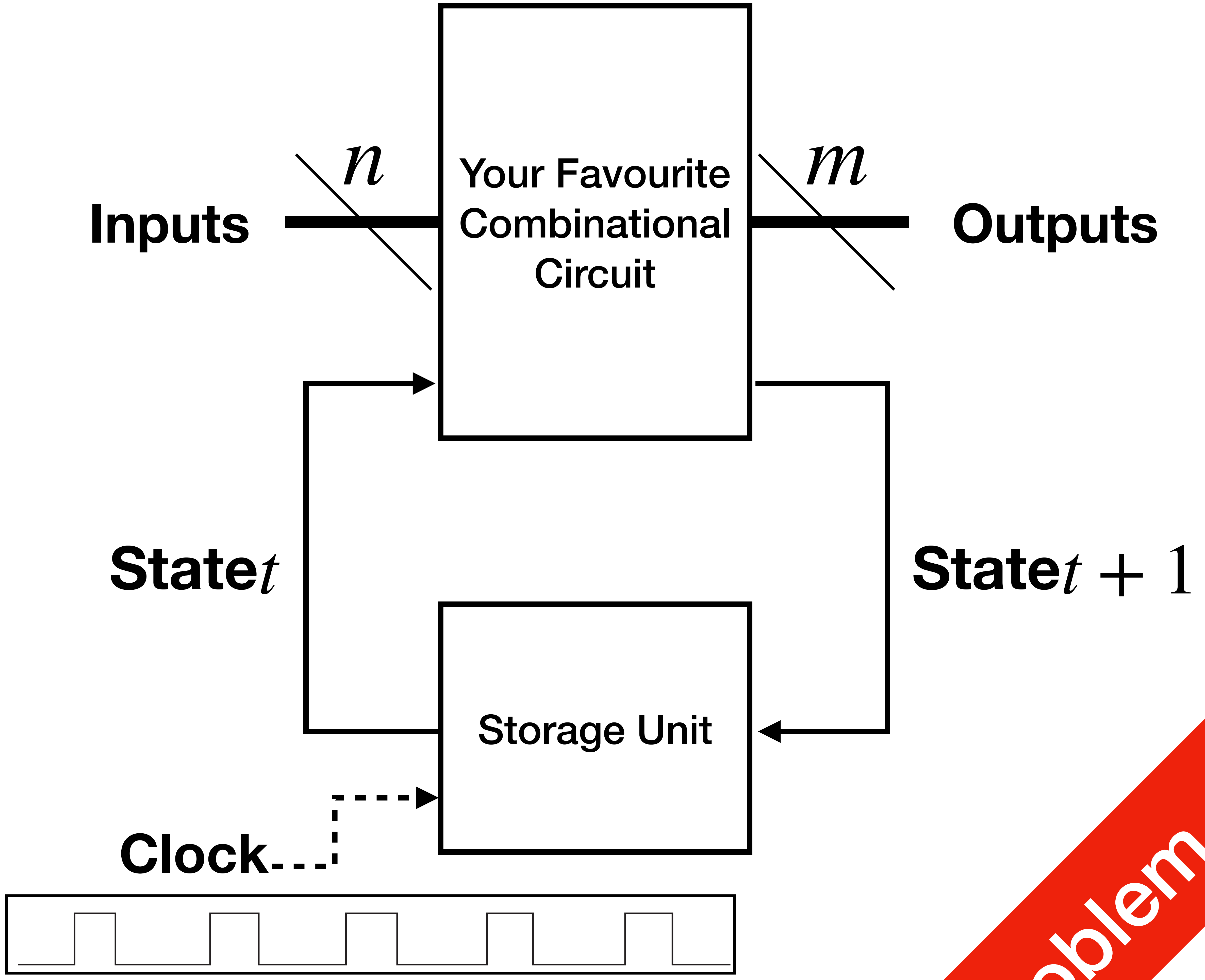


Problem

P1
Flip-Flops

Problems with Latches

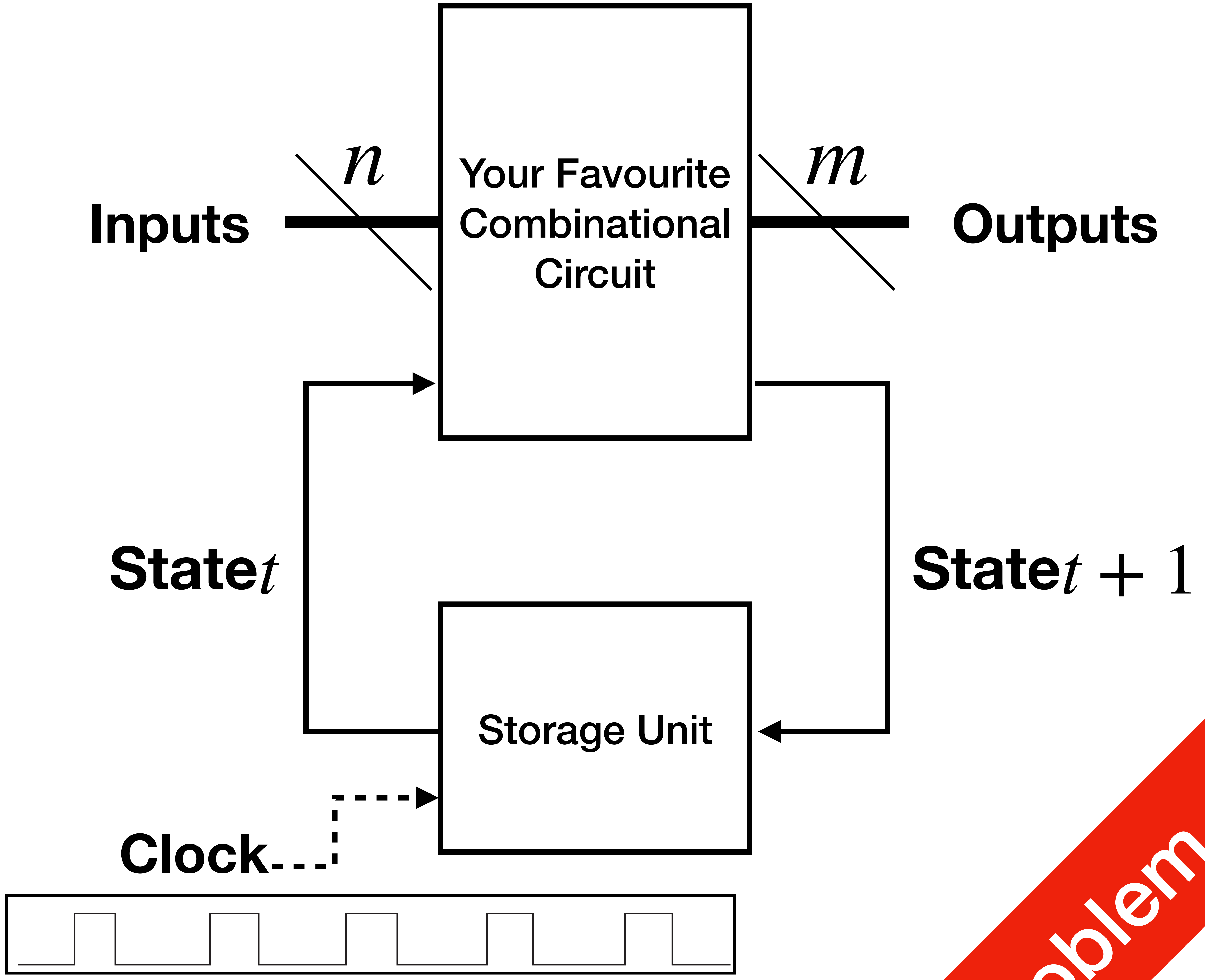
- Transparent: changes happen instantly
    - Time t, input changes



Problem

Problems with Latches

- Transparent: changes happen instantly
    - Time t, input changes
    - Time $(t, t+1)$, output stabilises

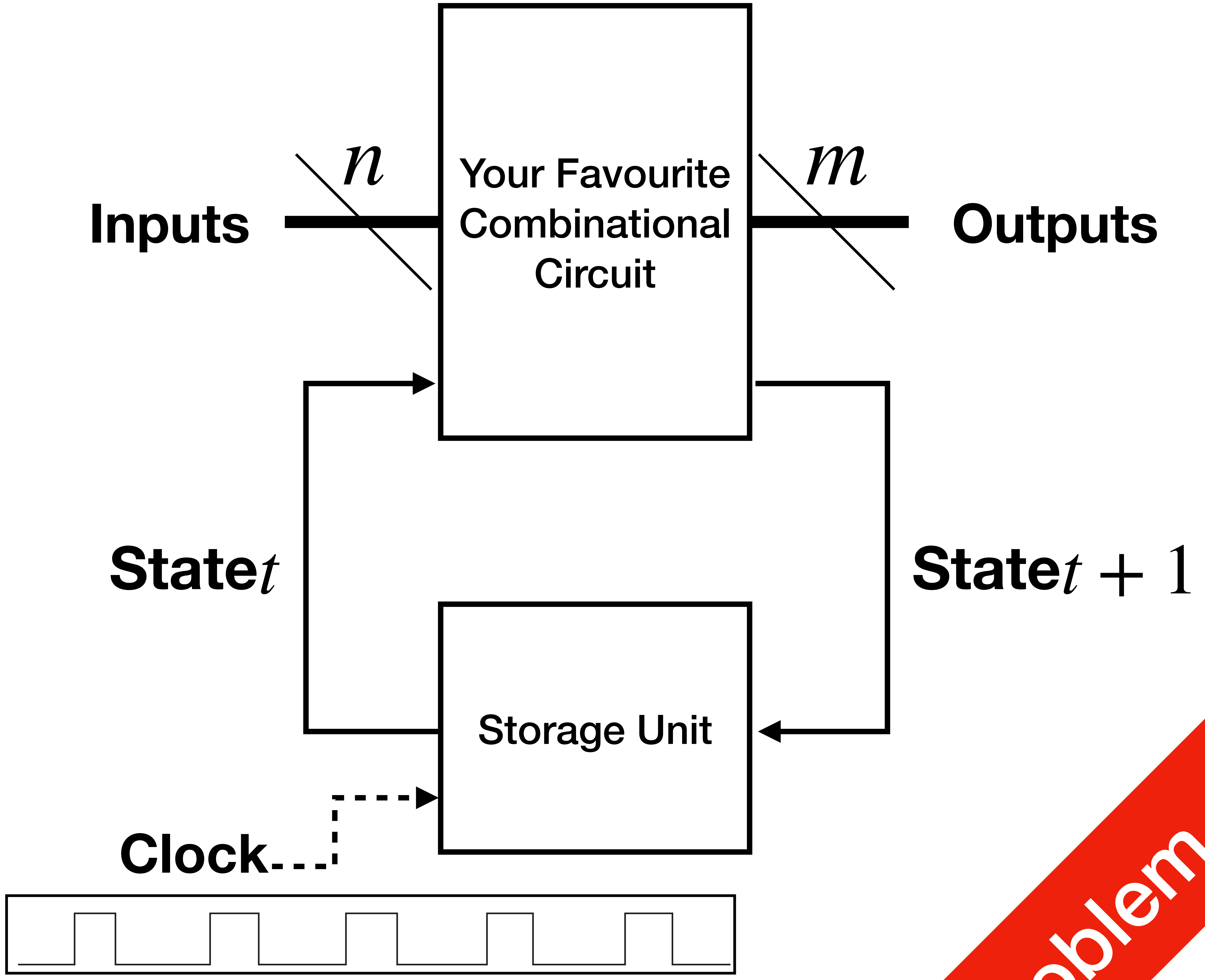


Problem

P1
Flip-Flops

Problems with Latches

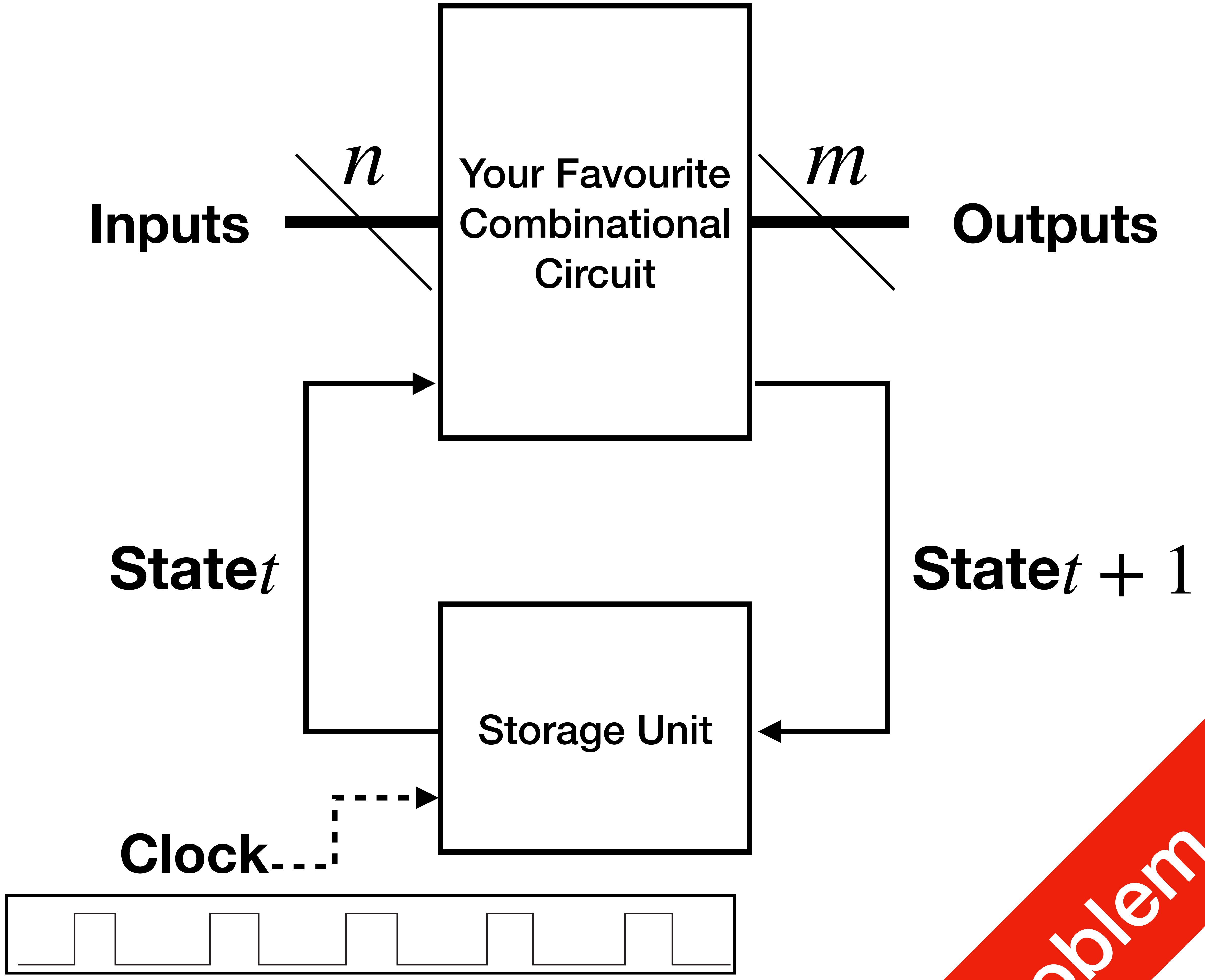
- Transparent: changes happen instantly
  - Time t, input changes
  - Time $(t, t+1)$, output stabilises
  - Time $t+1$, output stored in Storage Unit



Problem

Problems with Latches

- Transparent: changes happen instantly
  - Time t, input changes
  - Time $(t, t + 1)$, output stabilises
  - Time $t + 1$, output stored in Storage Unit

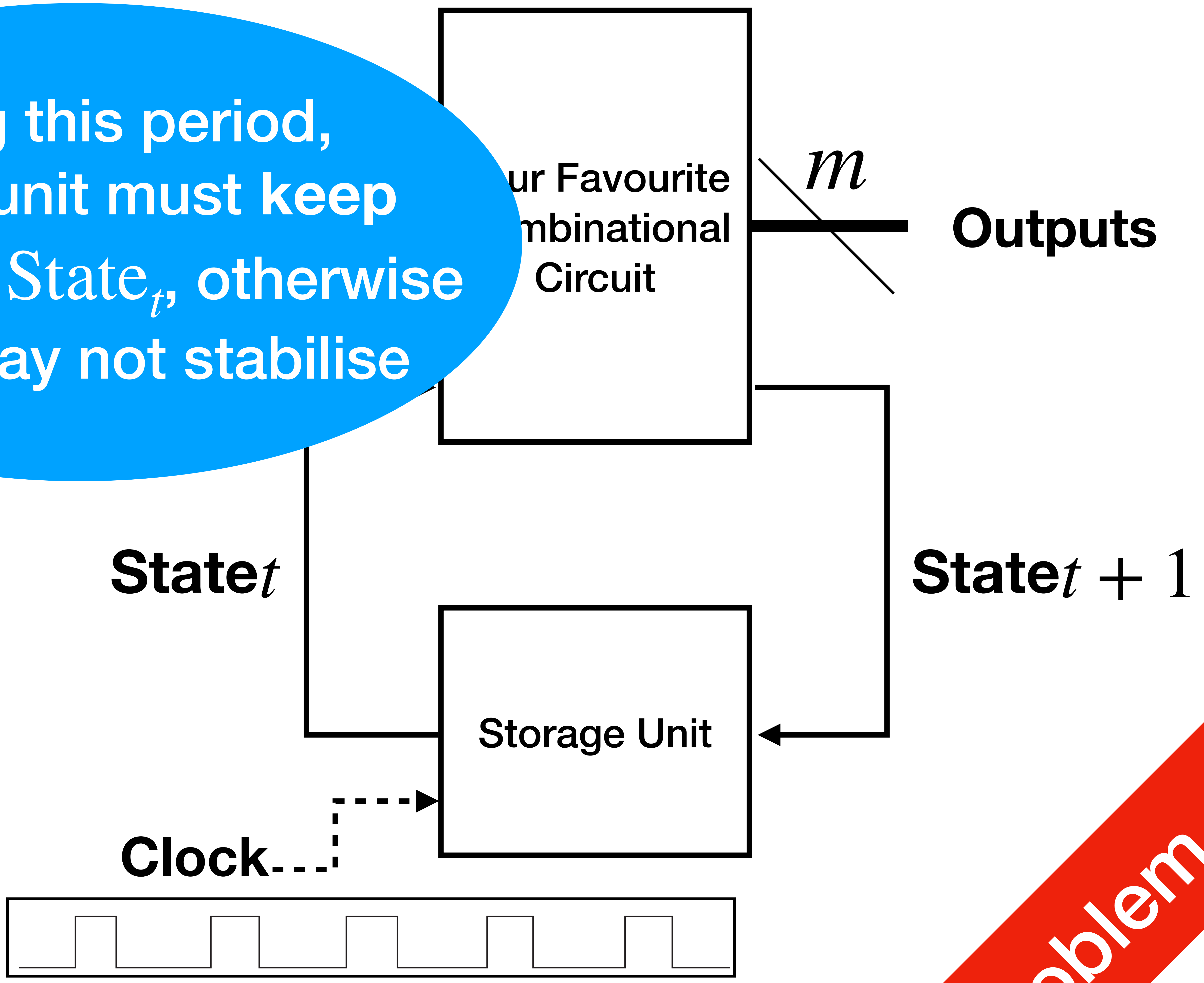


Problem

Problems with Latches

- Transparent: changes happ
  - Time t, input changes
  - Time $(t, t+1)$, output stabilises
  - Time $t+1$, output stored in Storage Unit

During this period, storage unit must keep outputting State_t, otherwise output may not stabilise

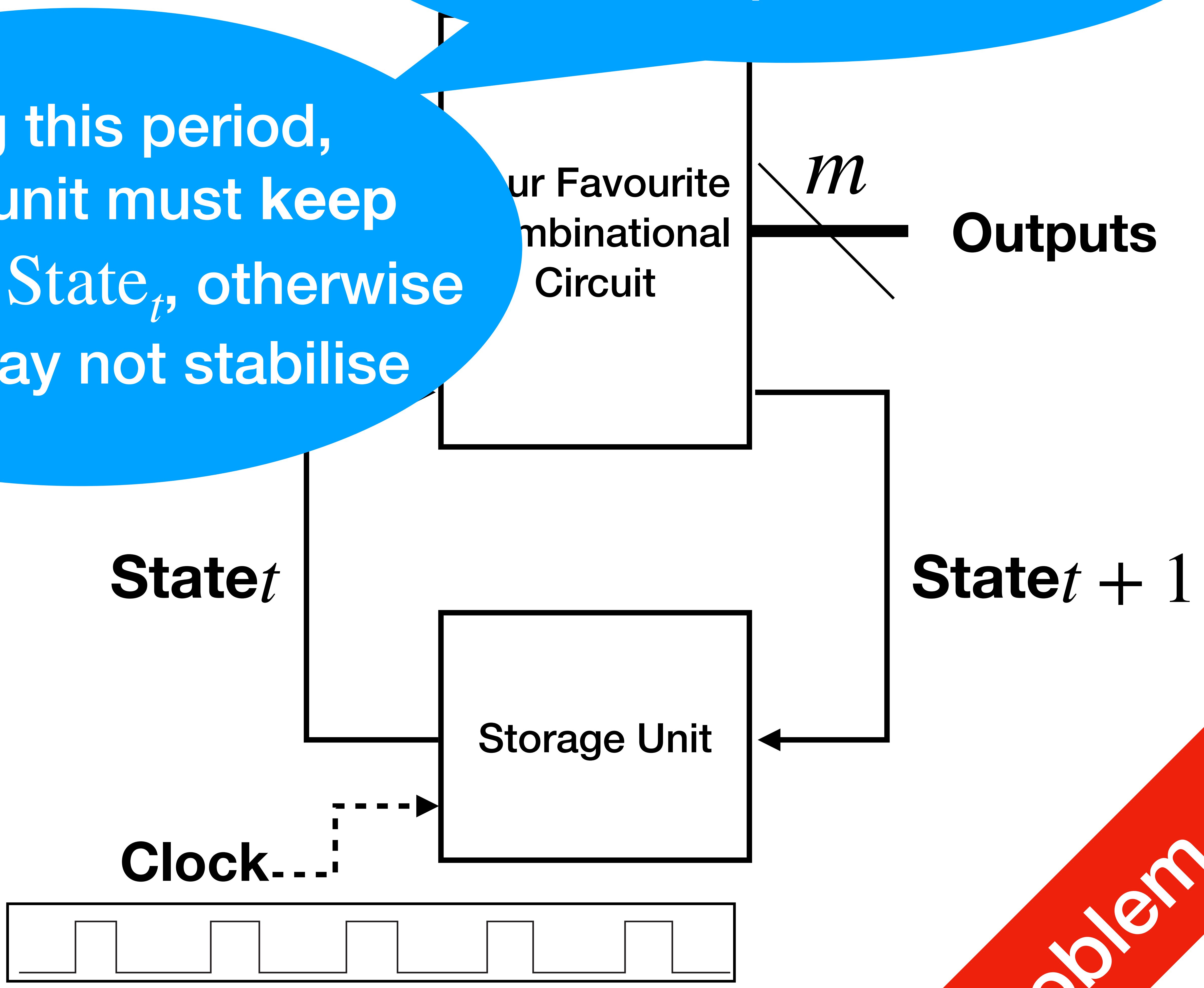


Problem

P1
Flip-Flops

Problems with Latch

- Transparent: changes happ
  - Time t, input changes
  - Time $(t, t+1)$, output stabilises
  - Time $t+1$, output stored in Storage Unit



During this period, storage unit must keep outputting State_t, otherwise output may not stabilise

Latches cannot accomplish this!

Problem

Flip-Flops

Concept

Flip-Flops

- Time t, clock flips, new input arrives

Flip-Flops

- Time t, clock flips, new input arrives
- Time $(t, t+1)$: output State_t

Flip-Flops

- Time t, clock flips, new input arrives
- Time $(t, t+1)$: output State_t
  - meanwhile, new inputs arrives; State_{t+1} stabilises

Flip-Flops

- Time t, clock flips, new input arrives
- Time $(t, t+1)$: output State_t
  - meanwhile, new inputs arrives;
    State_{t+1} stabilises
- Time $t+1$, clock flips, storage rewritten as State_{t+1}, new input arrives

Concept

Flip-Flops

- Time t, clock flips, new input arrives
- Time $(t, t+1)$: output State_t
    - meanwhile, new inputs arrives;
      State_{t+1} stabilises
- Time $t+1$, clock flips, storage rewritten as State_{t+1}, new input arrives
- Time $(t+1, t+2)$: output State_{t+1}

P1
Flip-Flops

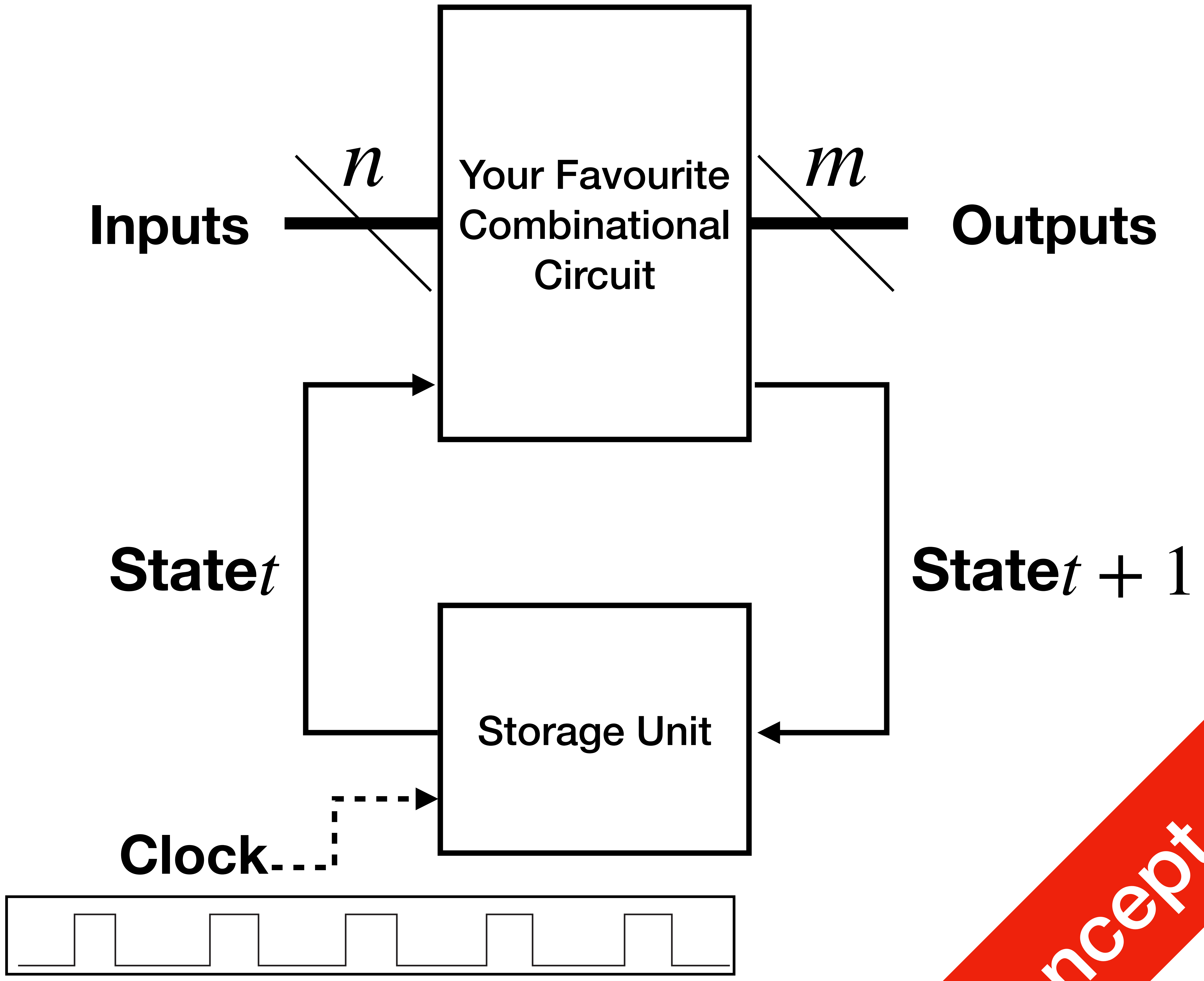
Flip-Flops

- Time t, clock flips, new input arrives
- Time $(t, t+1)$: output State_t
  - meanwhile, new inputs arrives;
    State_{t+1} stabilises
- Time $t+1$, clock flips, storage rewritten as State_{t+1}, new input arrives
- Time $(t+1, t+2)$: output State_{t+1}
  - ...

Concept

P1
Flip-Flops

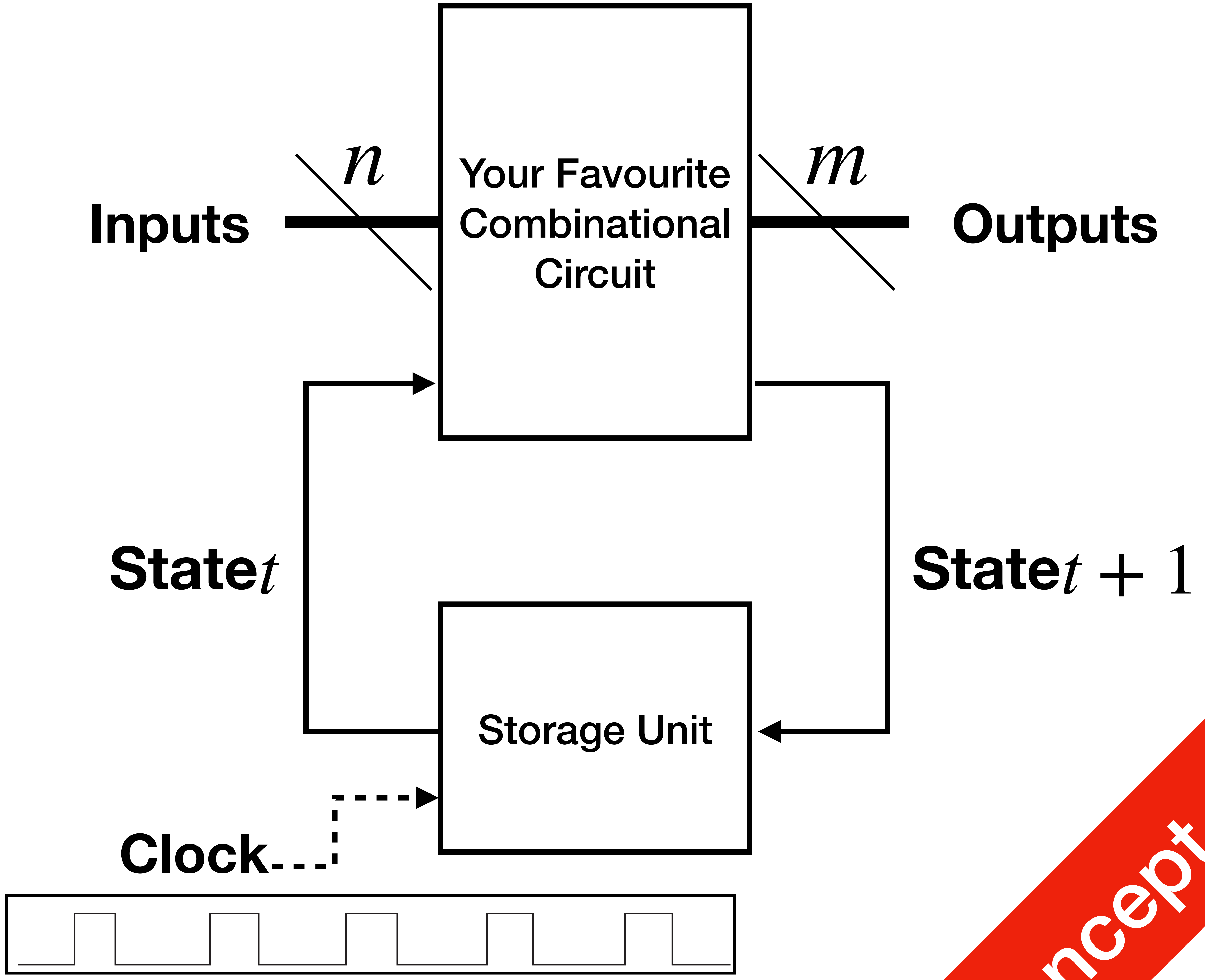
Flip-Flops



Concept

Flip-Flops

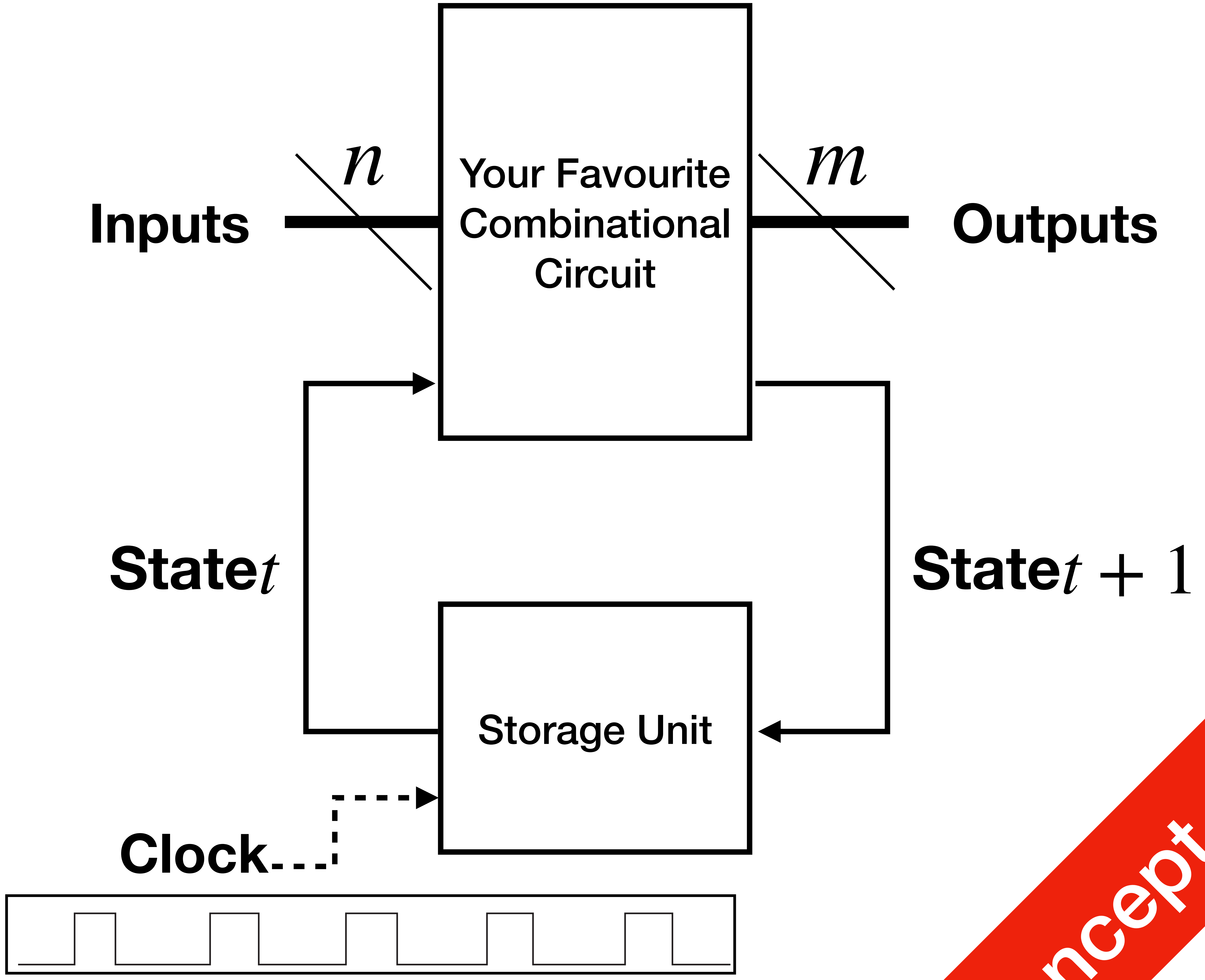
- Time t, clock flips, new input arrives



Concept

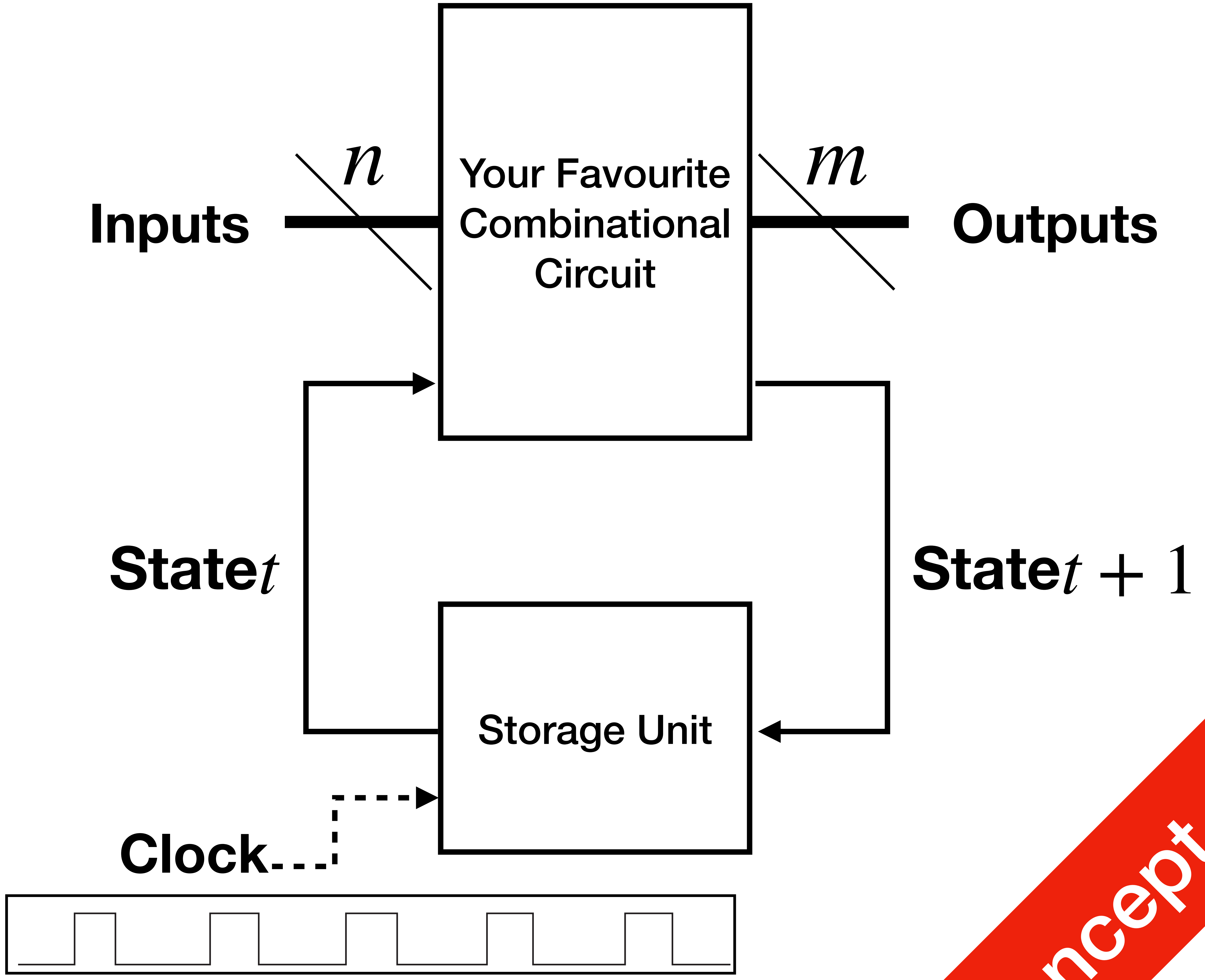
Flip-Flops

- Time t, clock flips, new input arrives
- Time $(t, t+1)$: output State_t



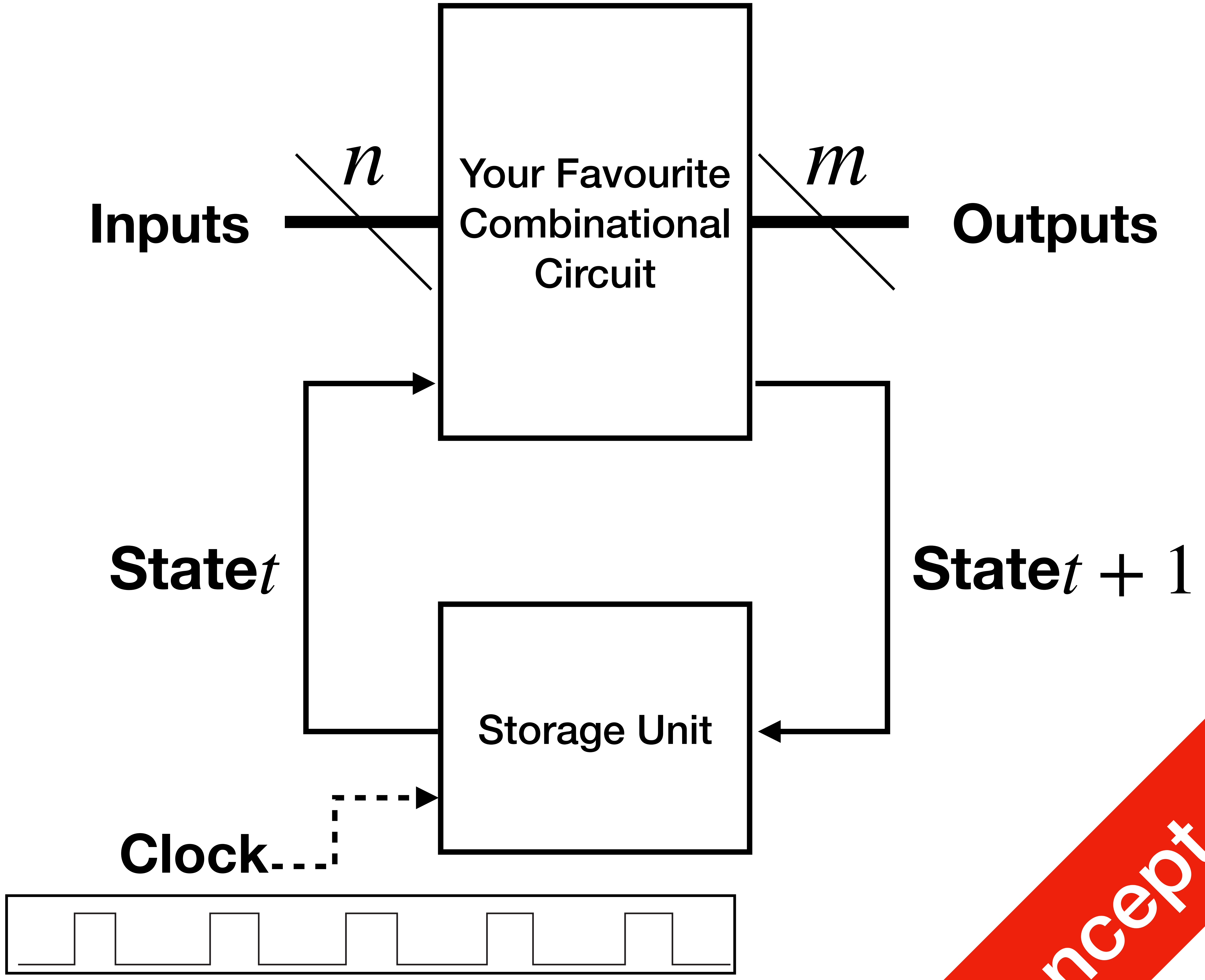
Flip-Flops

- Time t, clock flips, new input arrives
- Time $(t, t+1)$: output State_t
  - meanwhile, new inputs arrives; State_{t+1} stabilises



Flip-Flops

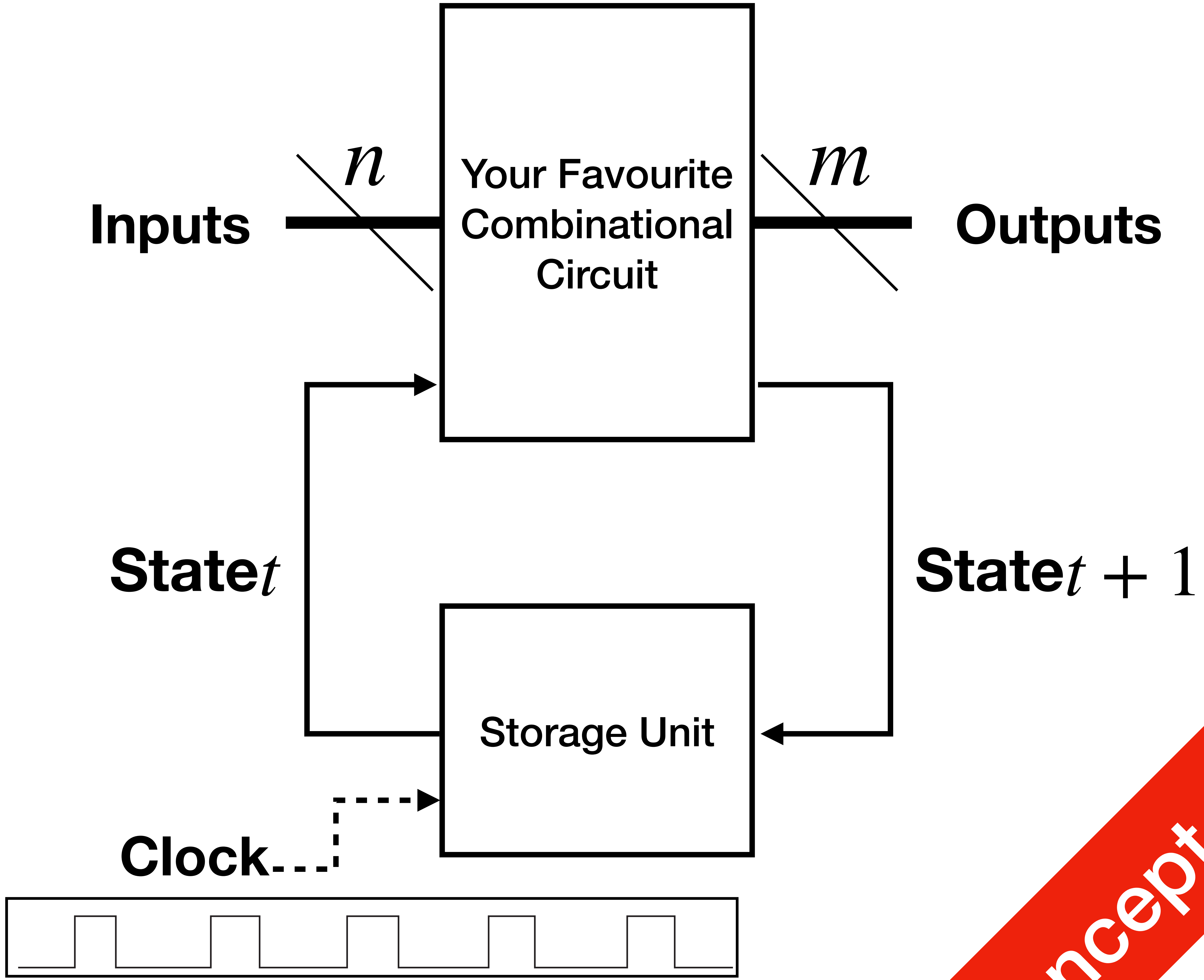
- Time t, clock flips, new input arrives
- Time $(t, t+1)$: output State_t
  - meanwhile, new inputs arrives; State_{t+1} stabilises
- Time $t+1$, clock flips, storage rewritten as State_{t+1}, new input arrives



Concept

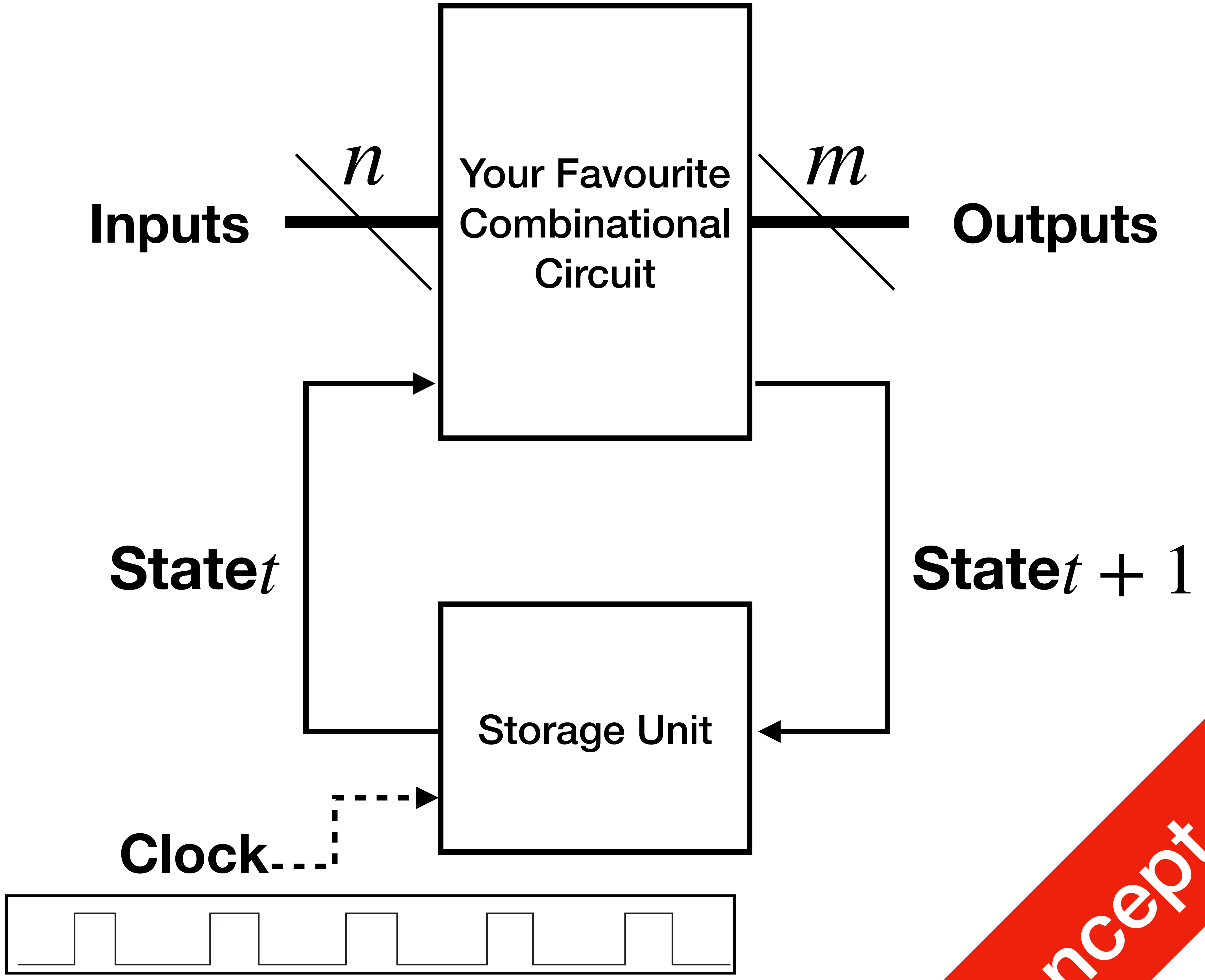
Flip-Flops

- Time t, clock flips, new input arrives
- Time $(t, t+1)$: output State_t
  - meanwhile, new inputs arrives; State_{t+1} stabilises
- Time $t+1$, clock flips, storage rewritten as State_{t+1}, new input arrives
- Time $(t+1, t+2)$: output State_{t+1}



Flip-Flops

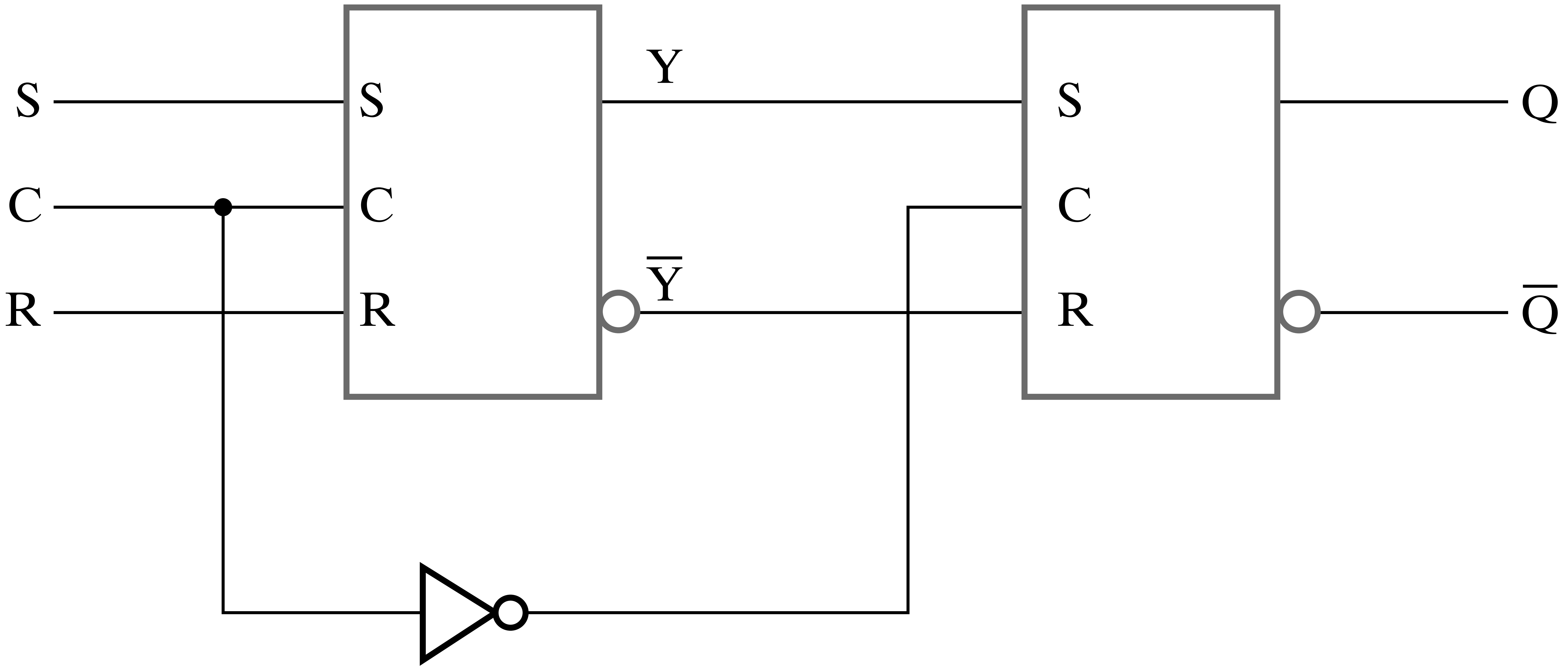
- Time t, clock flips, new input arrives
- Time $(t, t+1)$: output State_t
  - meanwhile, new inputs arrives; State_{t+1} stabilises
- Time $t+1$, clock flips, storage rewritten as State_{t+1}, new input arrives
- Time $(t+1, t+2)$: output State_{t+1}
  - …



Concept

P1
Flip-Flops

SR Master-Slave Flip-Flop

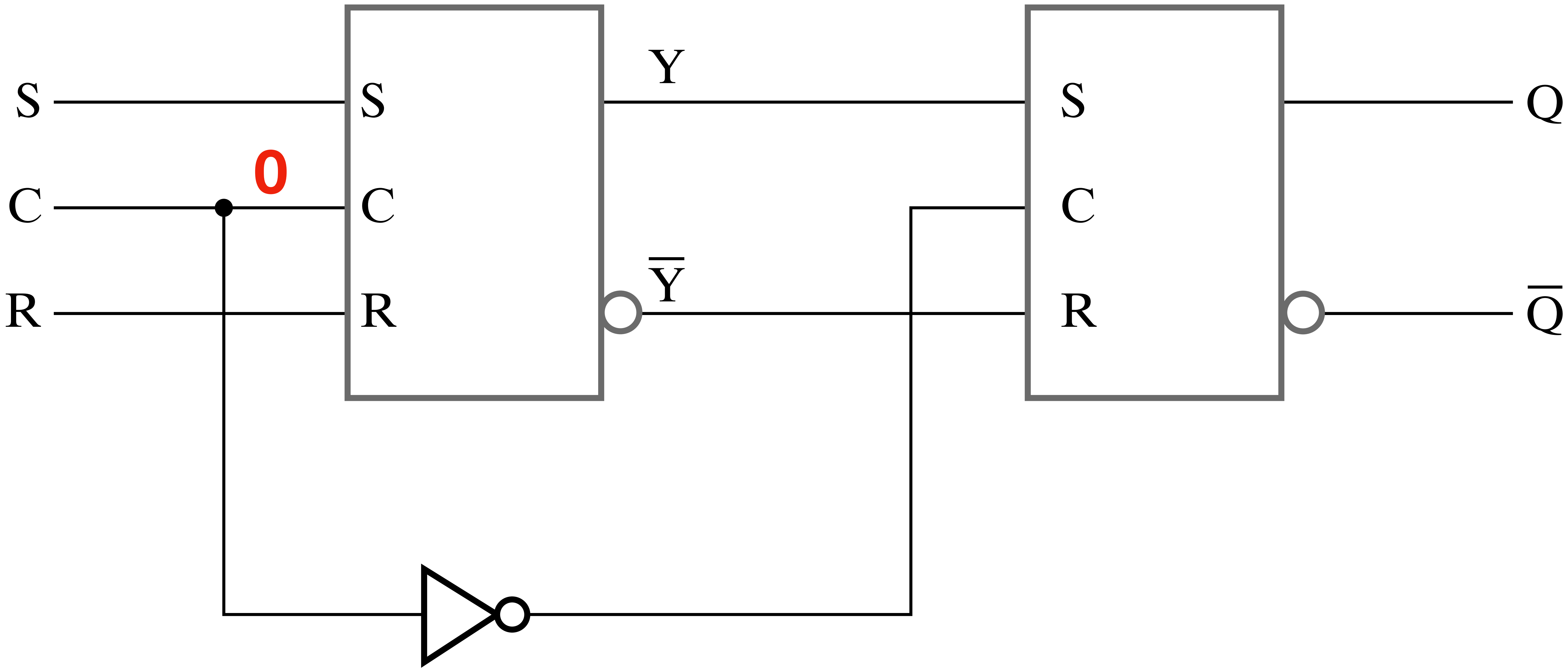


- Constructed using SR latches, left Master, right Slave
- Output state changes require $C = 0$ -> $C = 1$ -> $C = 0$ (Positive Pulse)

Concept

P1
Flip-Flops

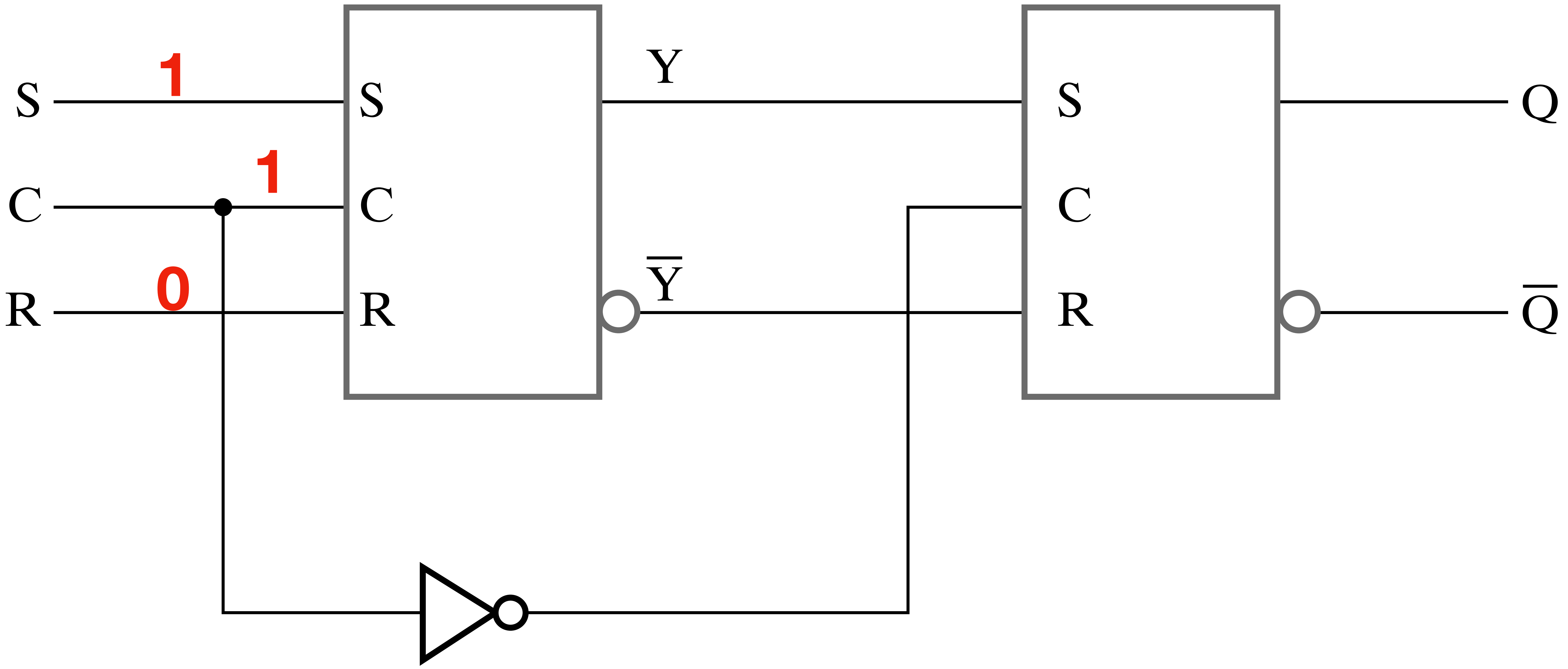
SR Master-Slave Flip-Flop



- Constructed using SR latches, left Master, right Slave
- Output state changes require $C = 0$ -> $C = 1$ -> $C = 0$ (Positive Pulse)

Technical

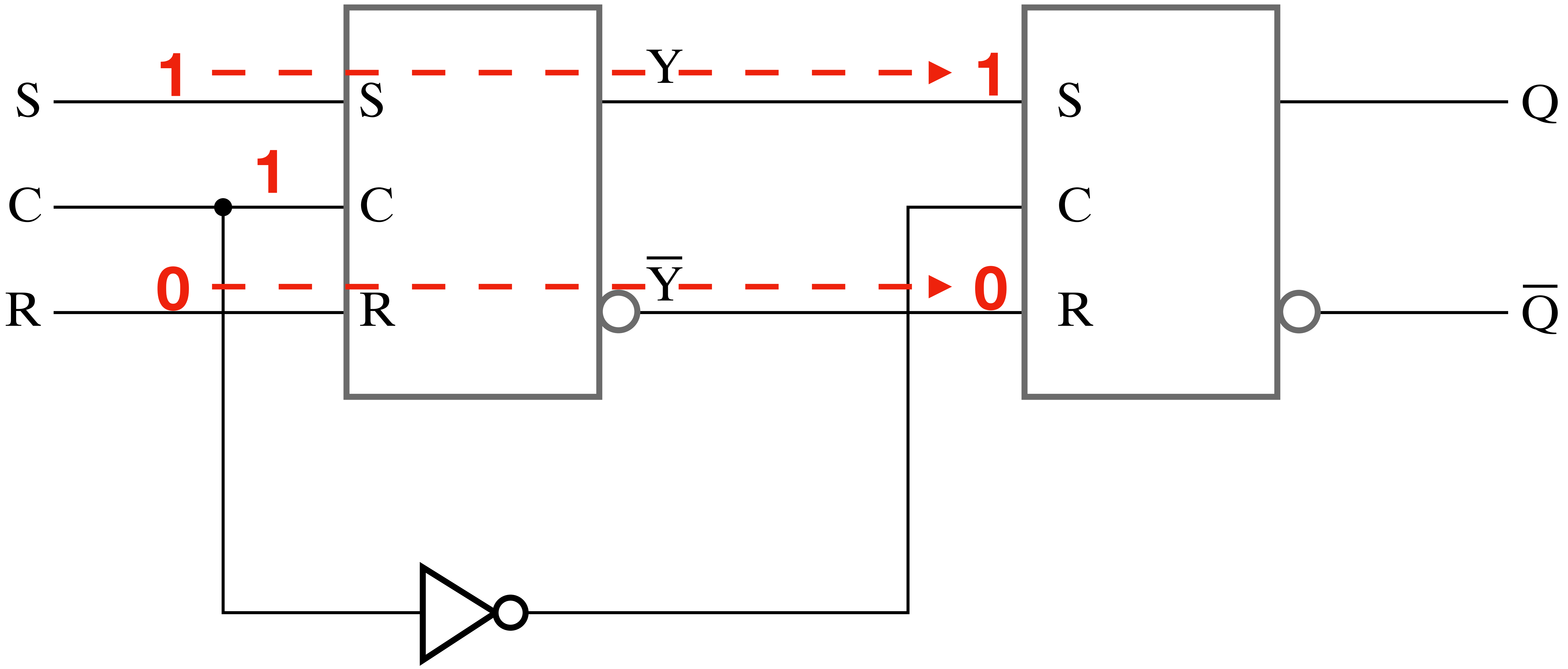
SR Master-Slave Flip-Flop



- Constructed using SR latches, left Master, right Slave
- Output state changes require $C = 0$ -> $C = 1$ -> $C = 0$ (Positive Pulse)

P1
Flip-Flops

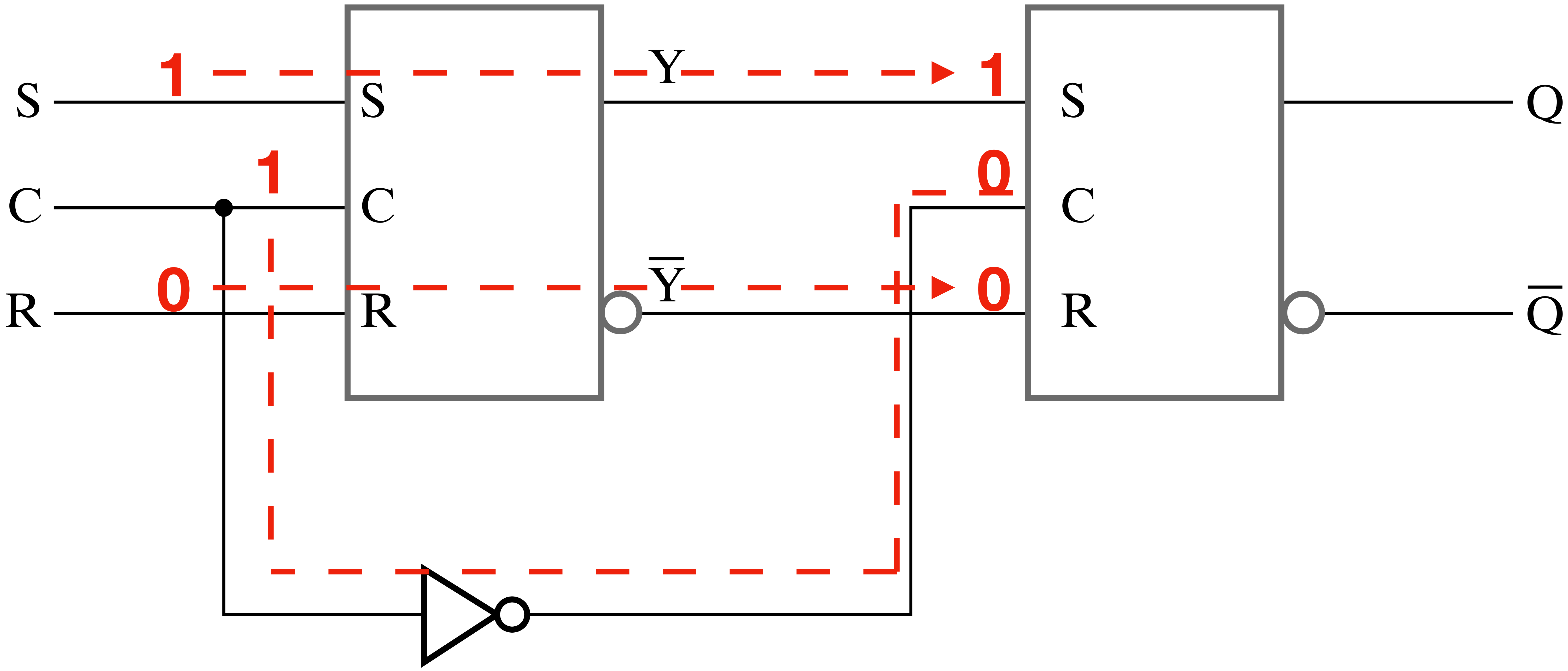
SR Master-Slave Flip-Flop



- Constructed using SR latches, left Master, right Slave
- Output state changes require $C = 0$ -> $C = 1$ -> $C = 0$ (Positive Pulse)

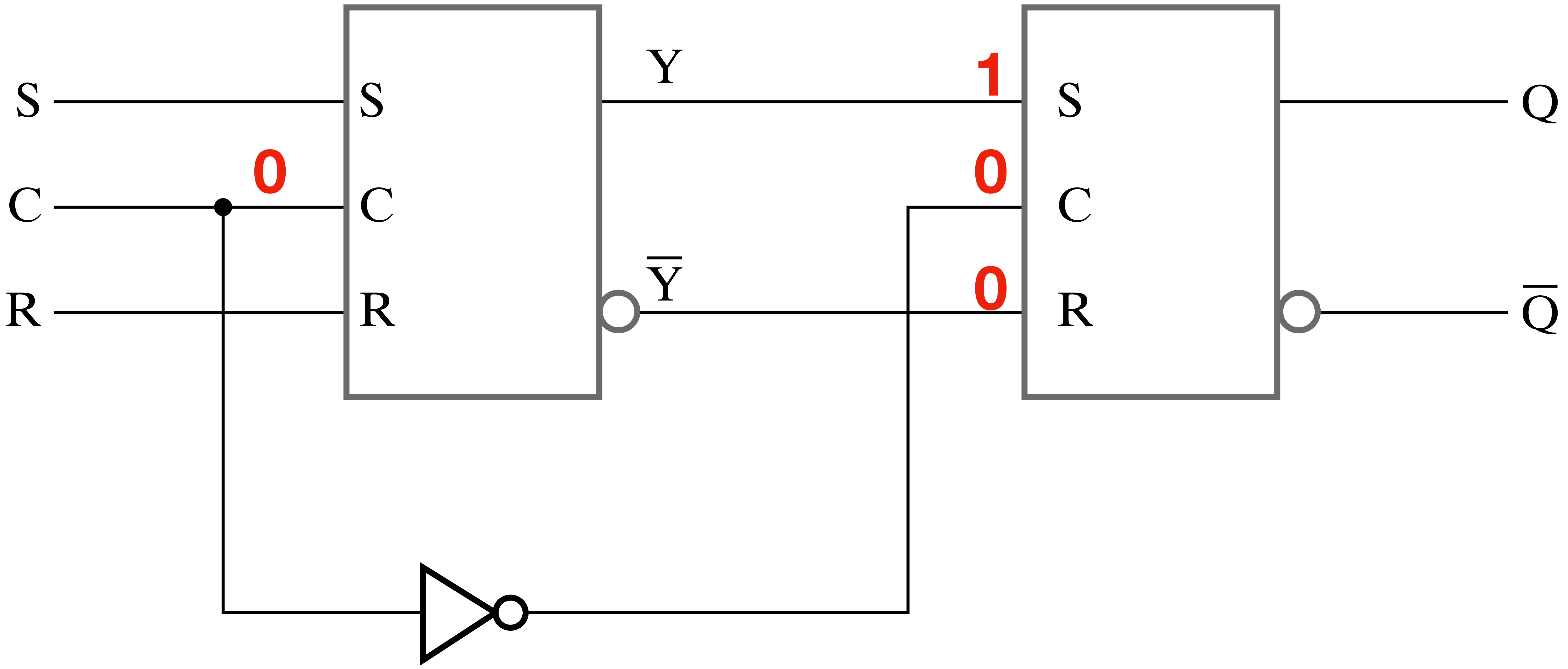
Technical

SR Master-Slave Flip-Flop



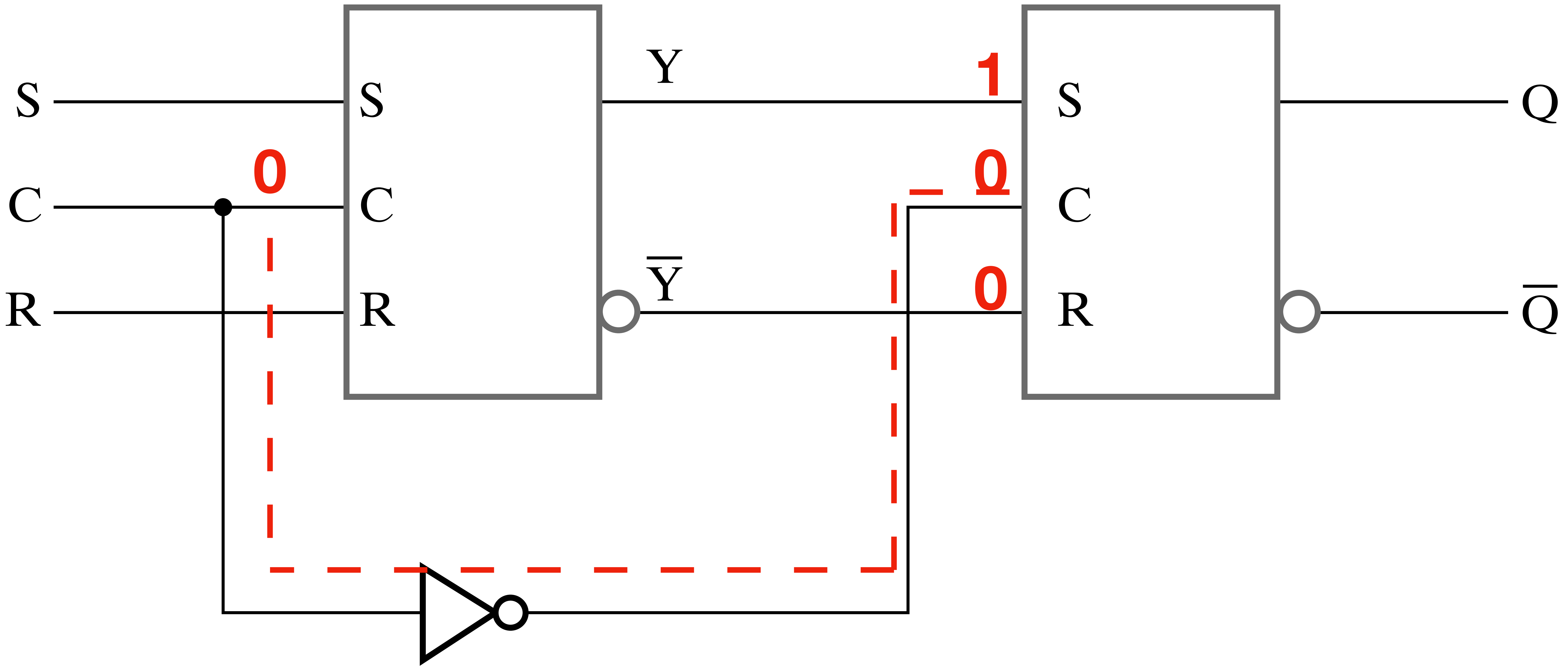
- Constructed using SR latches, left Master, right Slave
- Output state changes require $C = 0$ -> $C = 1$ -> $C = 0$ (Positive Pulse)

SR Master-Slave Flip-Flop



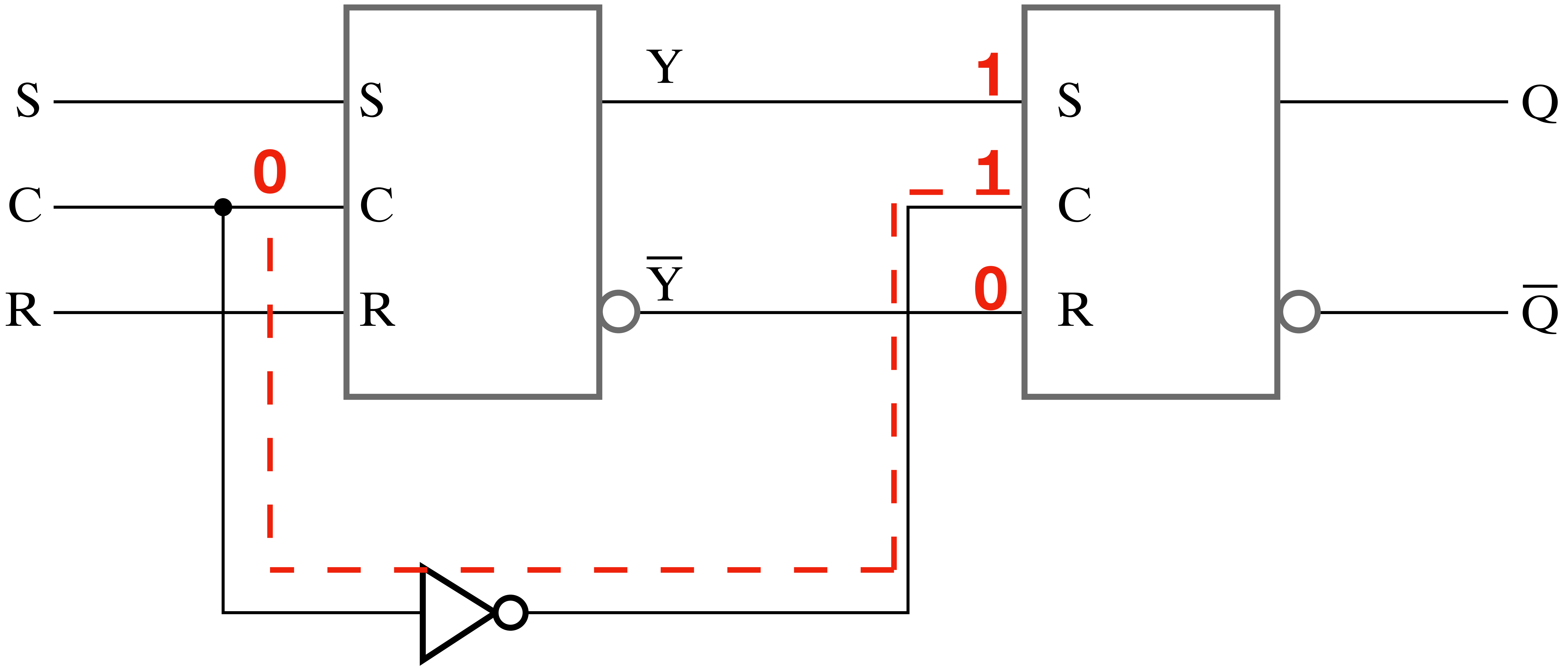
- Constructed using SR latches, left Master, right Slave
- Output state changes require $C = 0$ -> $C = 1$ -> $C = 0$ (Positive Pulse)

SR Master-Slave Flip-Flop



- Constructed using SR latches, left Master, right Slave
- Output state changes require $C = 0$ -> $C = 1$ -> $C = 0$ (Positive Pulse)

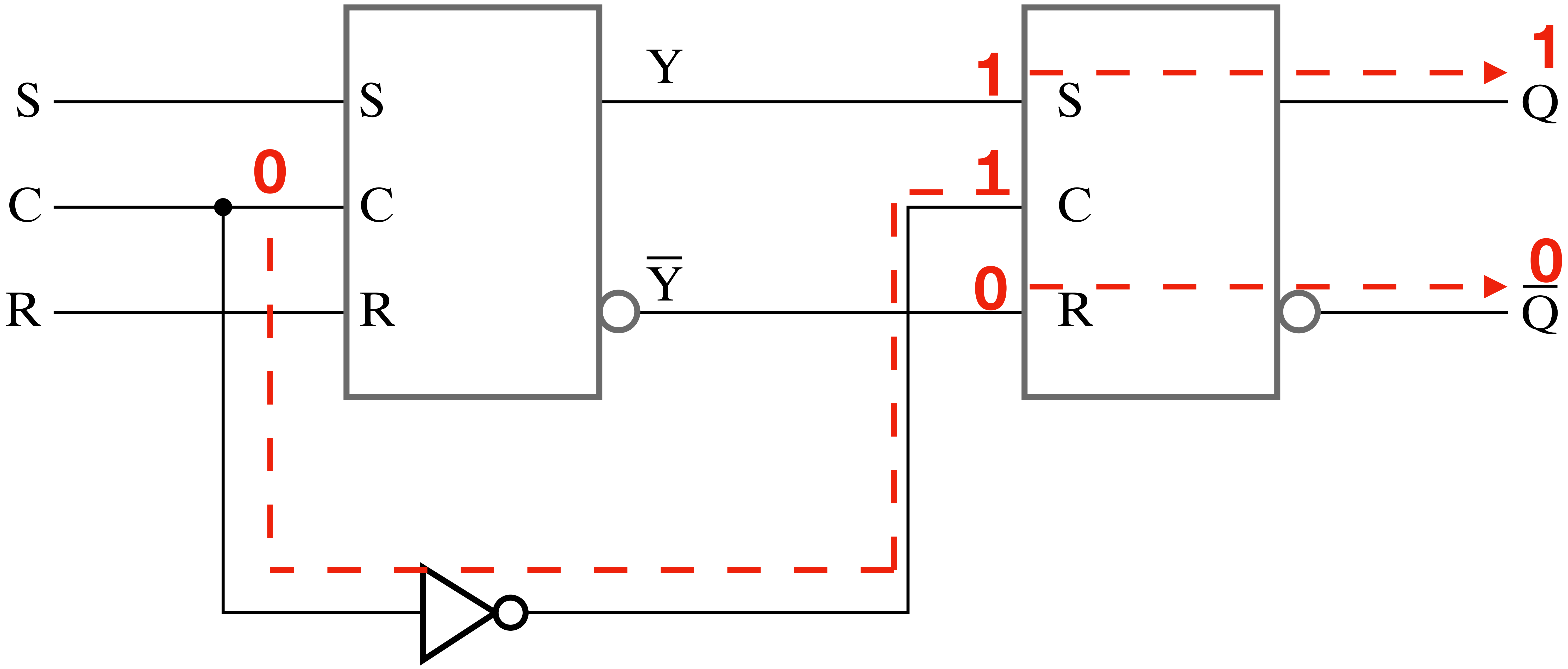
SR Master-Slave Flip-Flop



- Constructed using SR latches, left Master, right Slave
- Output state changes require $C = 0$ -> $C = 1$ -> $C = 0$ (Positive Pulse)

P1
Flip-Flops

SR Master-Slave Flip-Flop

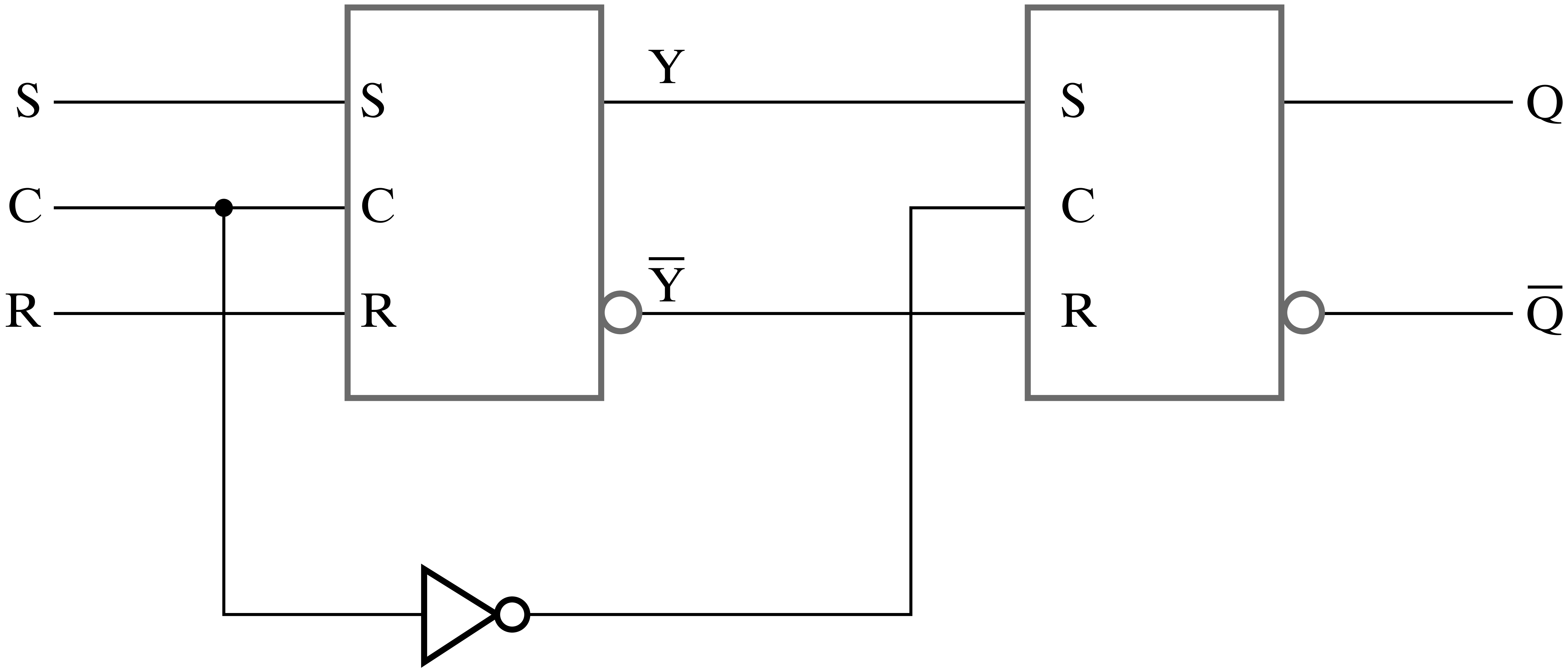


- Constructed using SR latches, left Master, right Slave
- Output state changes require $C = 0$ -> $C = 1$ -> $C = 0$ (Positive Pulse)

Technical

P1
Flip-Flops

SR Master-Slave Flip-Flop

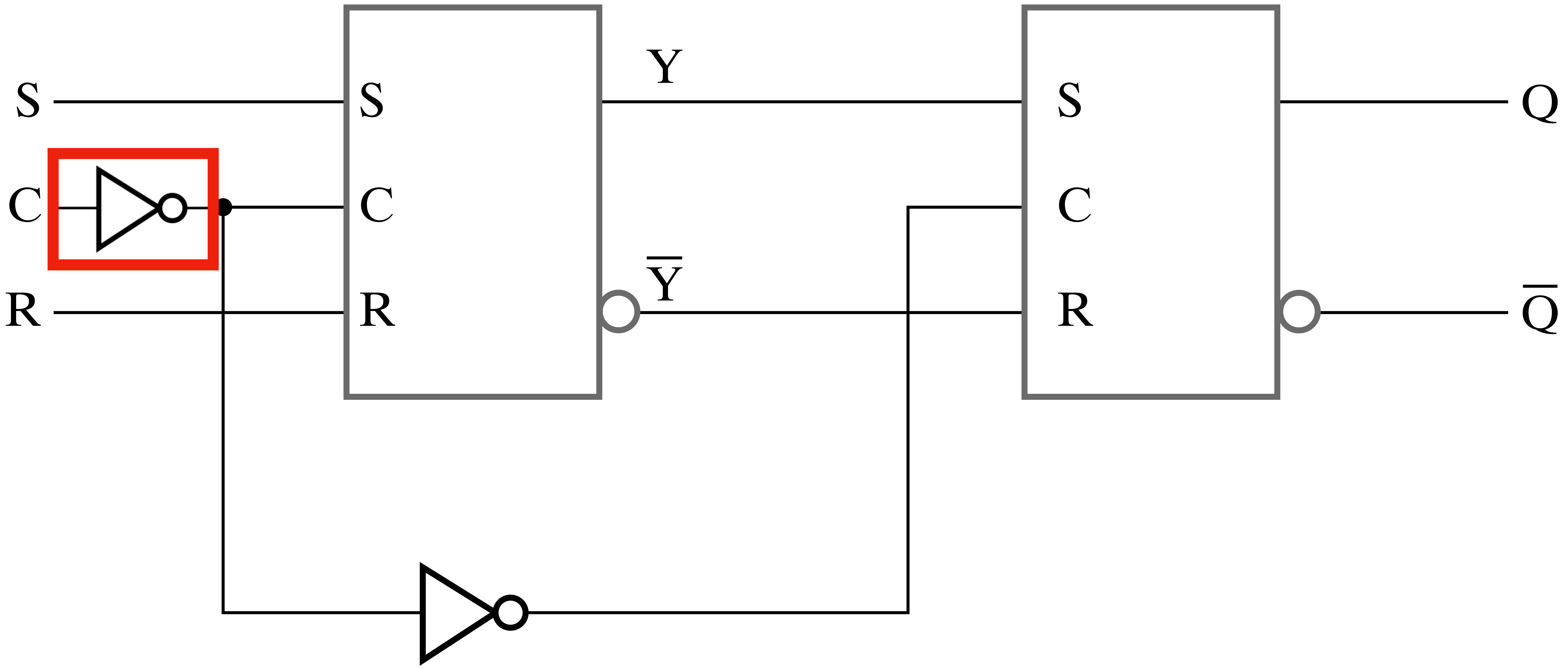


- Constructed using SR latches, left Master, right Slave
- Output state changes require $C = 0$ -> $C = 1$ -> $C = 0$ (Positive Pulse)
- Also called: **Positive Pulse Triggered** SR (Flip-Flop)

Concept

P1
Flip-Flops

SR Master-Slave Flip-Flop



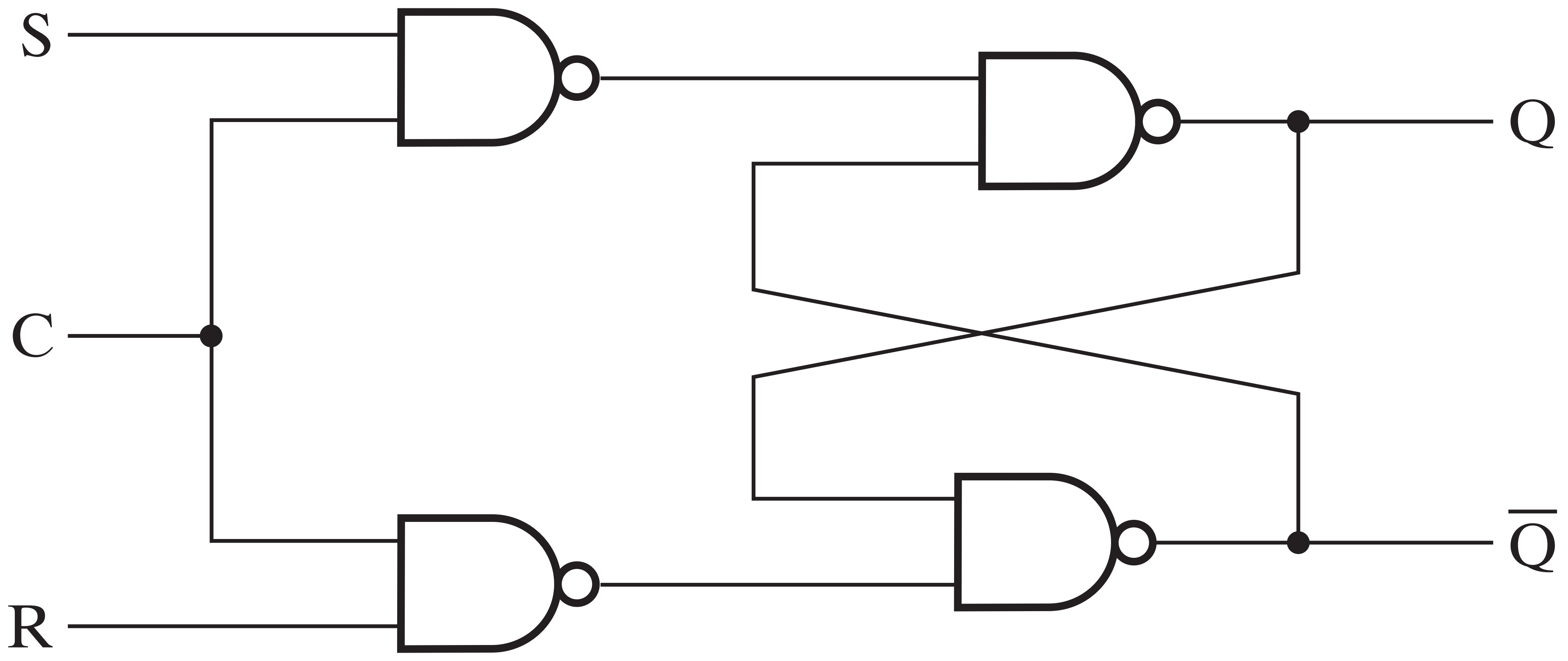
- Output state changes require $C = 1$ -> $C = 0$ -> $C = 1$ (Negative Pulse)

- **Negative Pulse Triggered** SR (Flip-Flop)

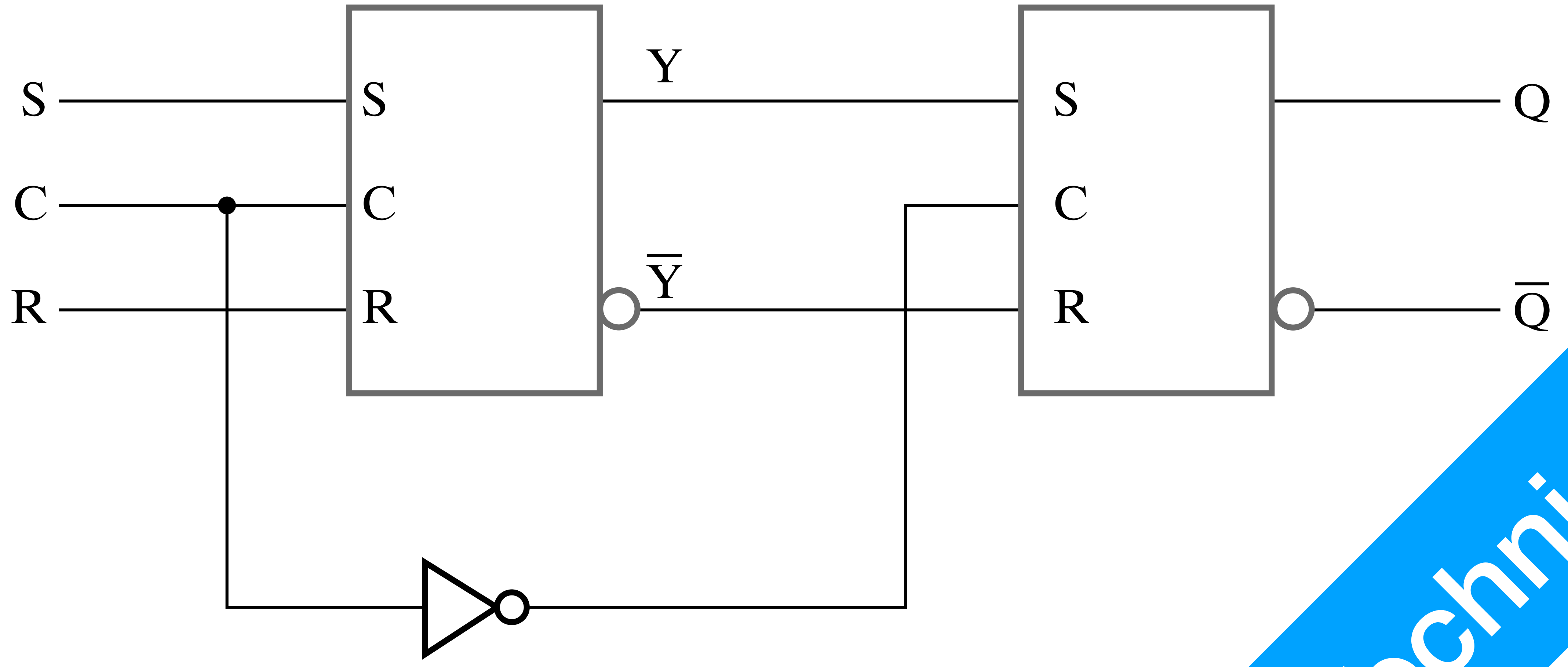
Concept

Implement
Positive Pulse Triggered SR

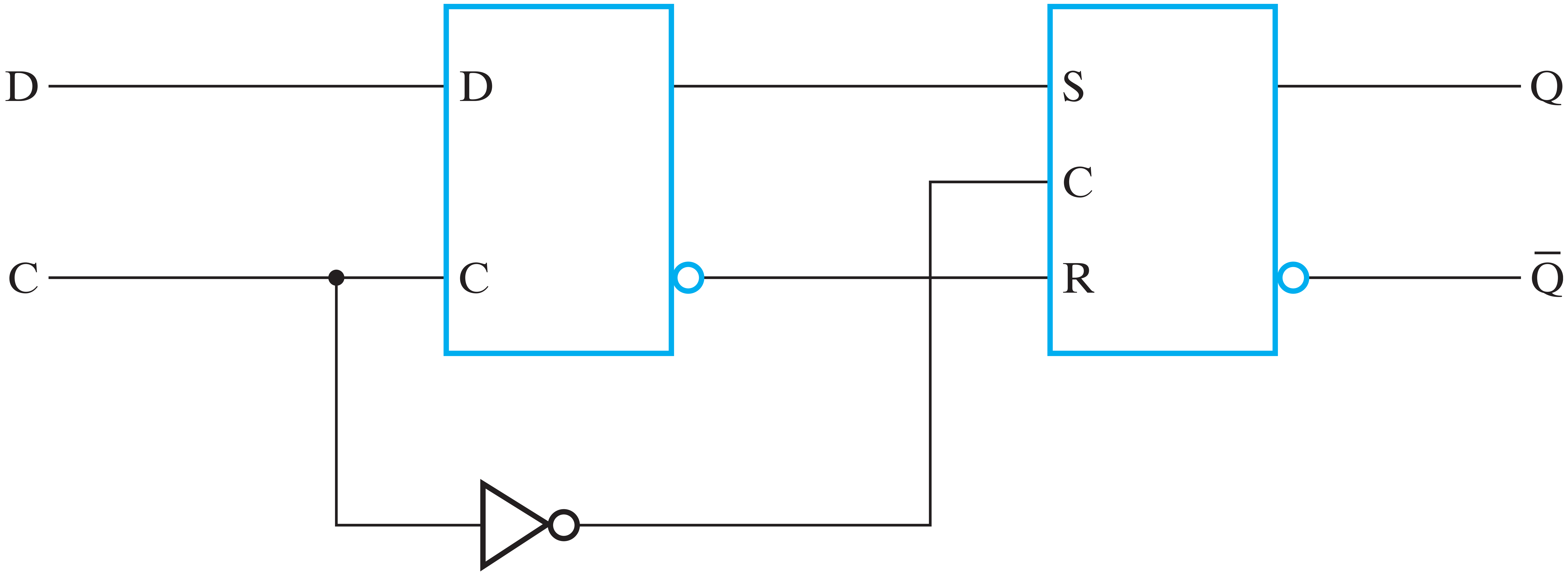
- Implement SR **Latch with Control Input** using $\overline{S}\overline{R}$ Latch



- Implement **Positive Pulse Triggered** SR using SR **latch with Control Input**



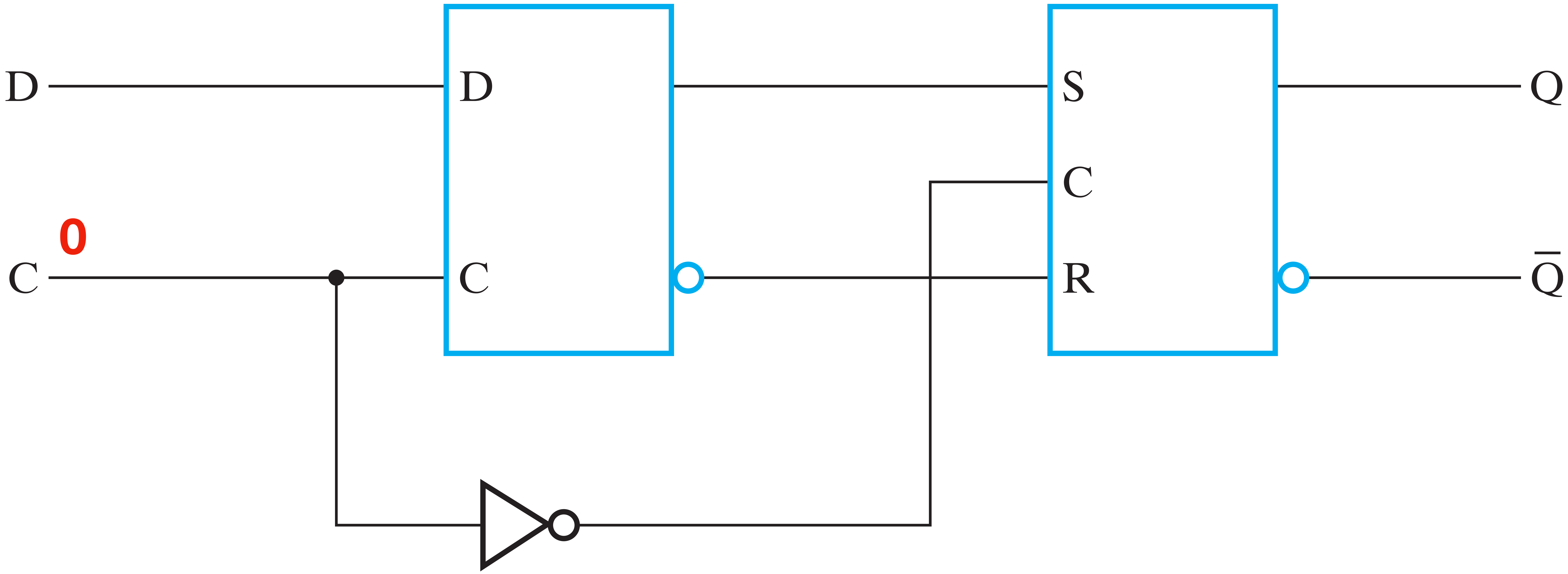
D Flip-Flop



- Replaces SR master in SR Master-Slave with D master Latch
- **Negative Edge Triggered** D (Flip-Flop): $C = 1$ -> $C = 0$

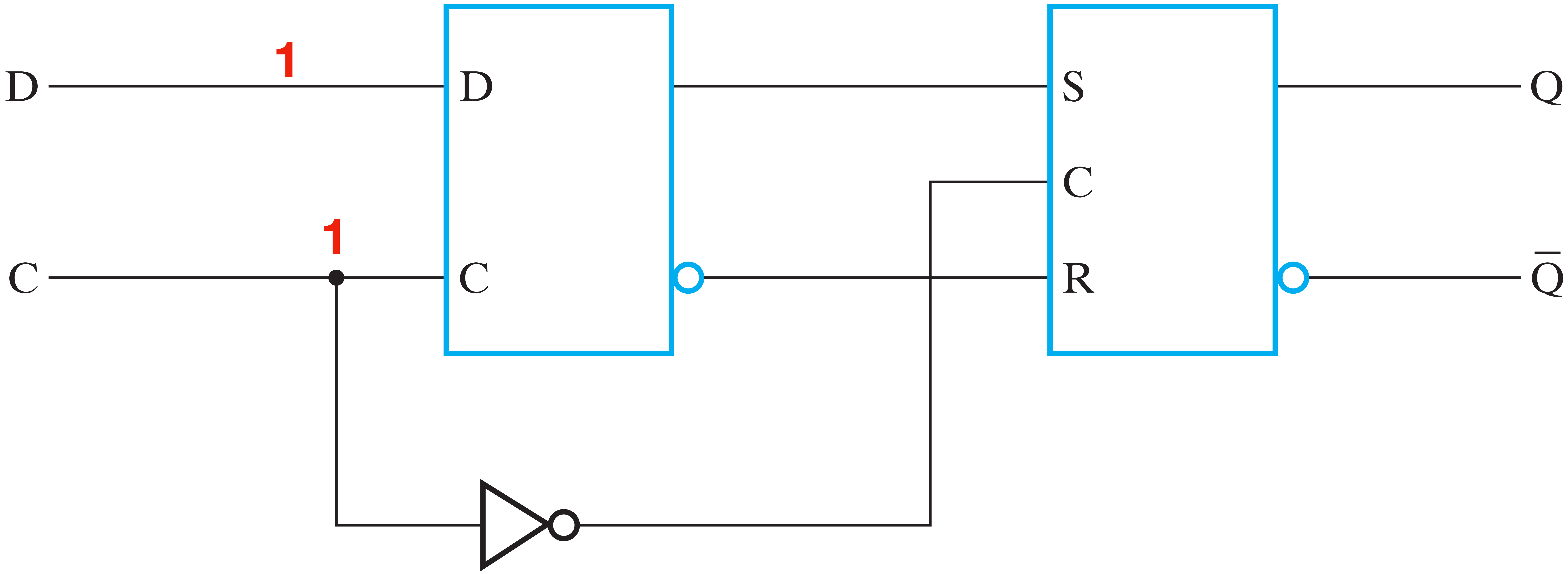
Concept

D Flip-Flop



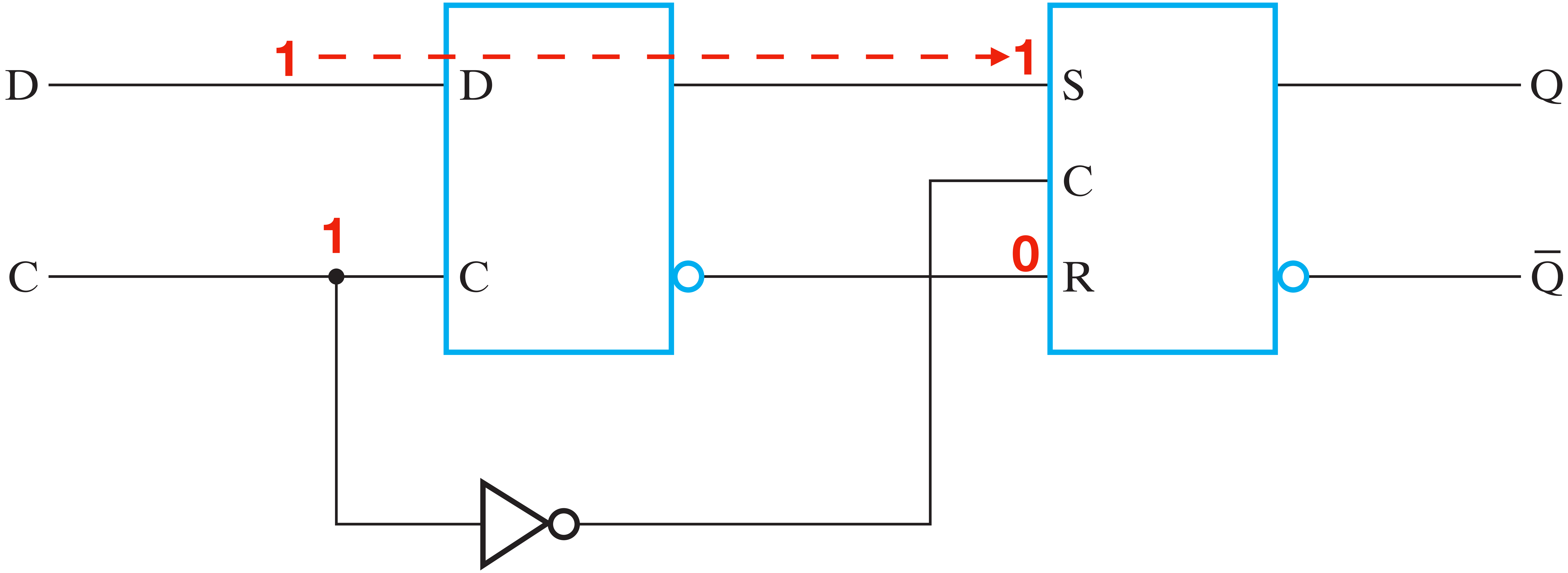
- Replaces SR master in SR Master-Slave with D master Latch
- **Negative Edge Triggered** D (Flip-Flop): $C = 1$ -> $C = 0$

D Flip-Flop



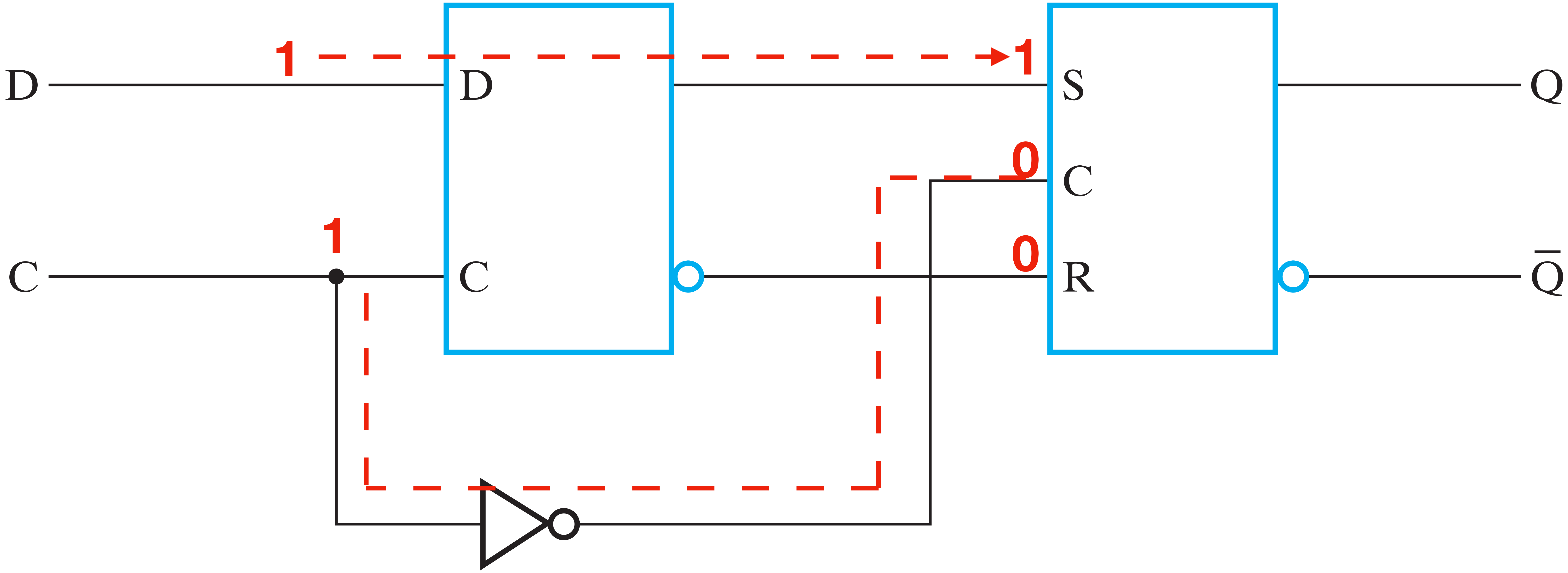
- Replaces SR master in SR Master-Slave with D master Latch
- **Negative Edge Triggered** D (Flip-Flop): $C = 1$ -> $C = 0$

D Flip-Flop



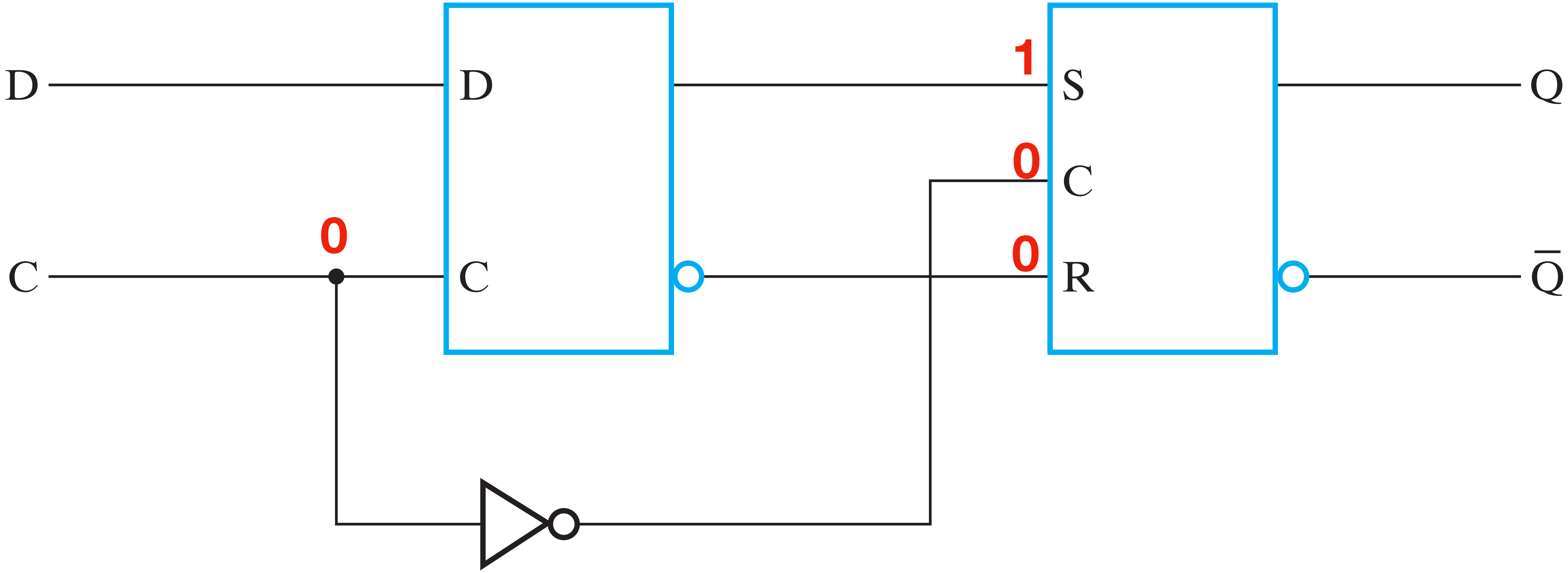
- Replaces SR master in SR Master-Slave with D master Latch
- **Negative Edge Triggered** D (Flip-Flop): $C = 1$ -> $C = 0$

D Flip-Flop



- Replaces SR master in SR Master-Slave with D master Latch
- **Negative Edge Triggered** D (Flip-Flop): $C = 1$ -> $C = 0$

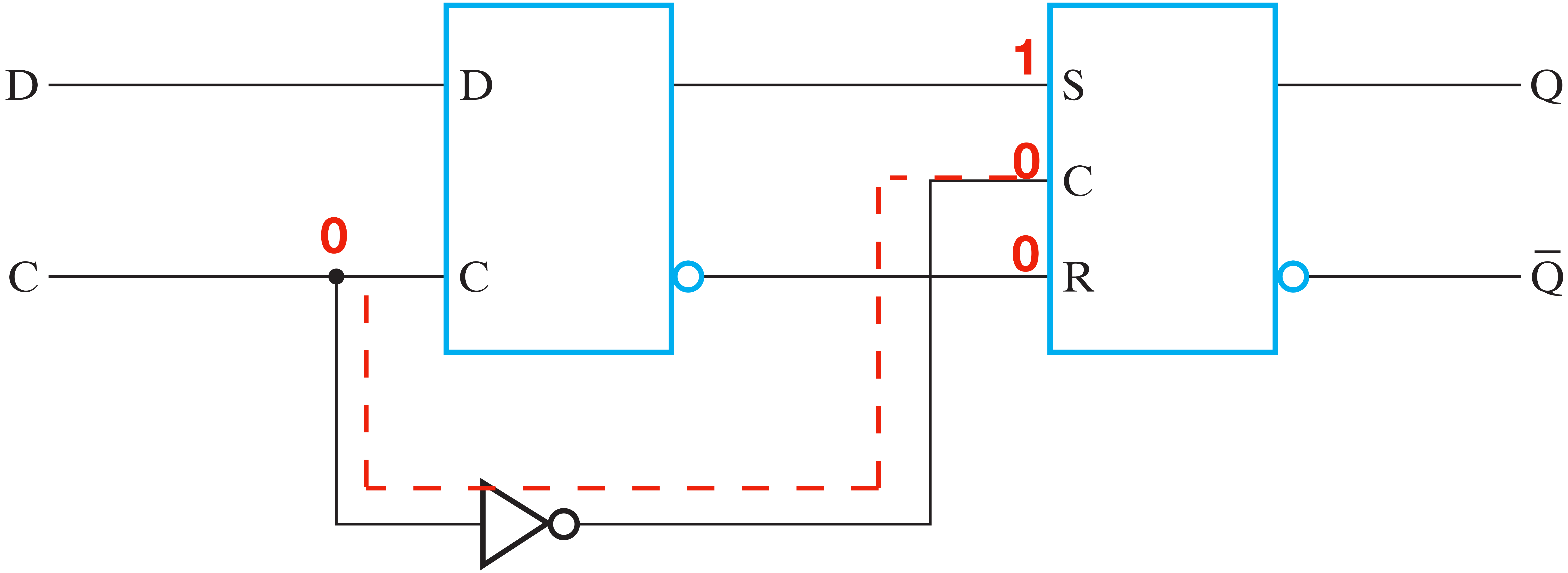
D Flip-Flop



- Replaces SR master in SR Master-Slave with D master Latch
- **Negative Edge Triggered** D (Flip-Flop): $C = 1$ -> $C = 0$

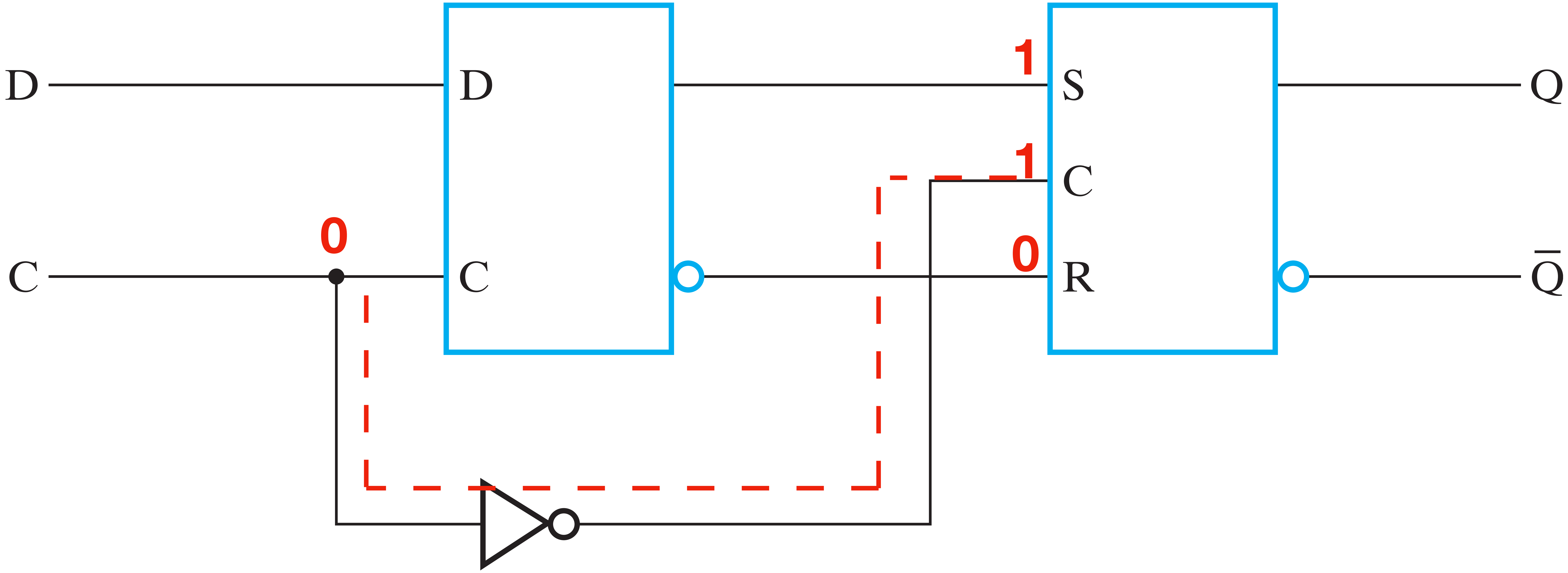
Demo

D Flip-Flop



- Replaces SR master in SR Master-Slave with D master Latch
- **Negative Edge Triggered** D (Flip-Flop): $C = 1$ -> $C = 0$

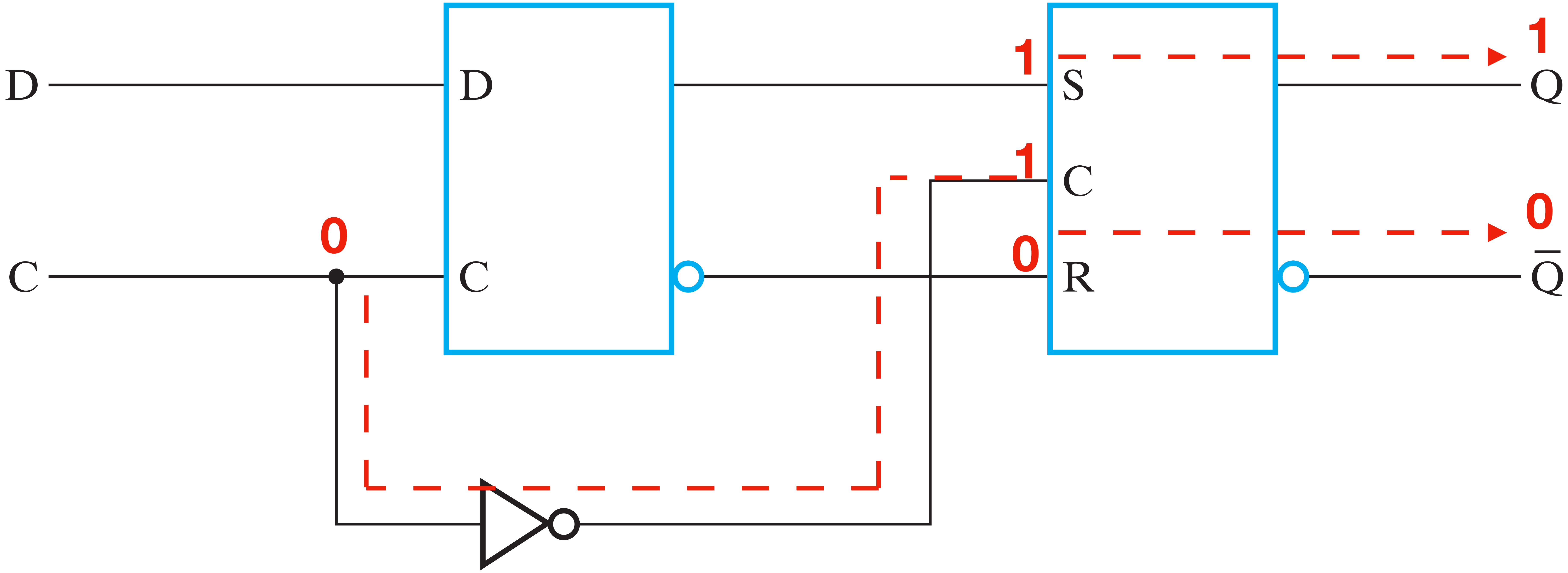
D Flip-Flop



- Replaces SR master in SR Master-Slave with D master Latch
- **Negative Edge Triggered** D (Flip-Flop): $C = 1$ -> $C = 0$

Demo

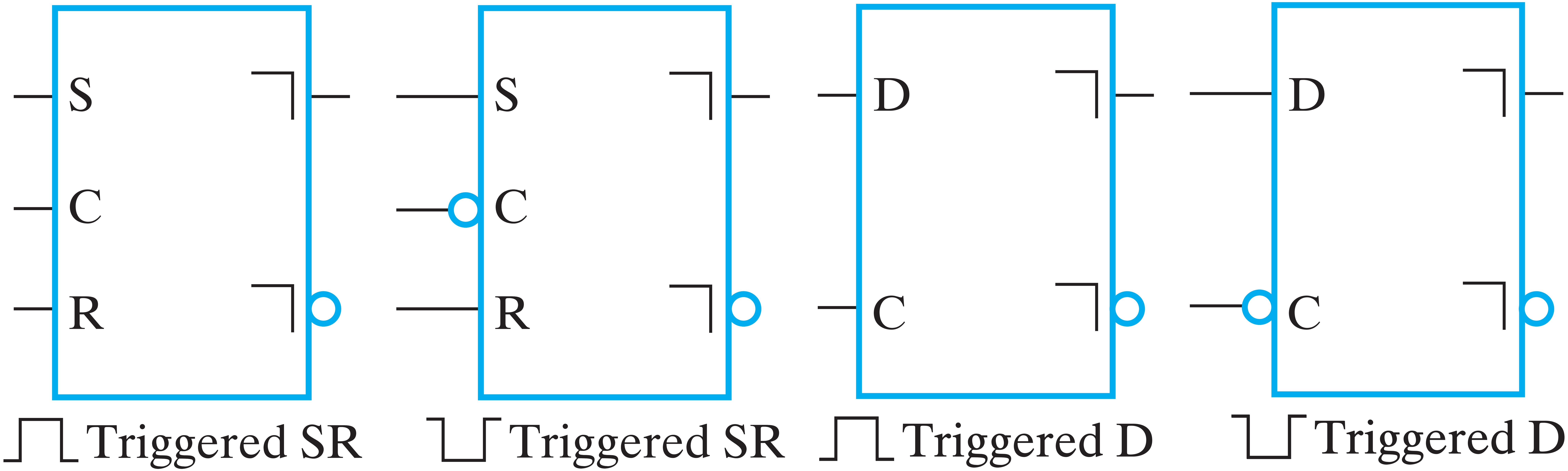
D Flip-Flop



- Replaces SR master in SR Master-Slave with D master Latch
- **Negative Edge Triggered** D (Flip-Flop): $C = 1$ -> $C = 0$

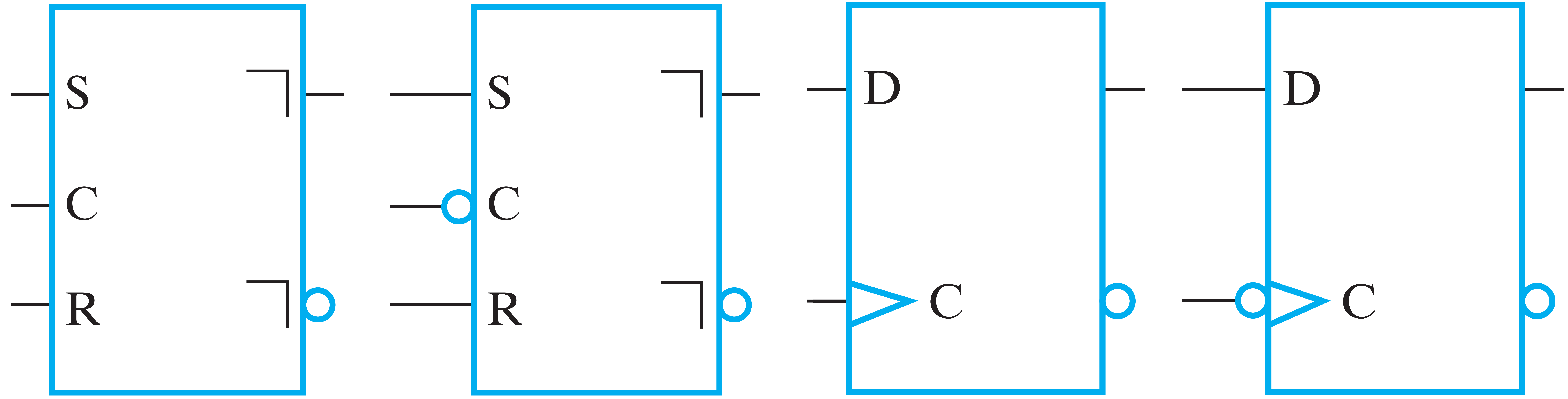
Demo

Flip Flops



Concept

Flip Flops



⎍ Triggered SR

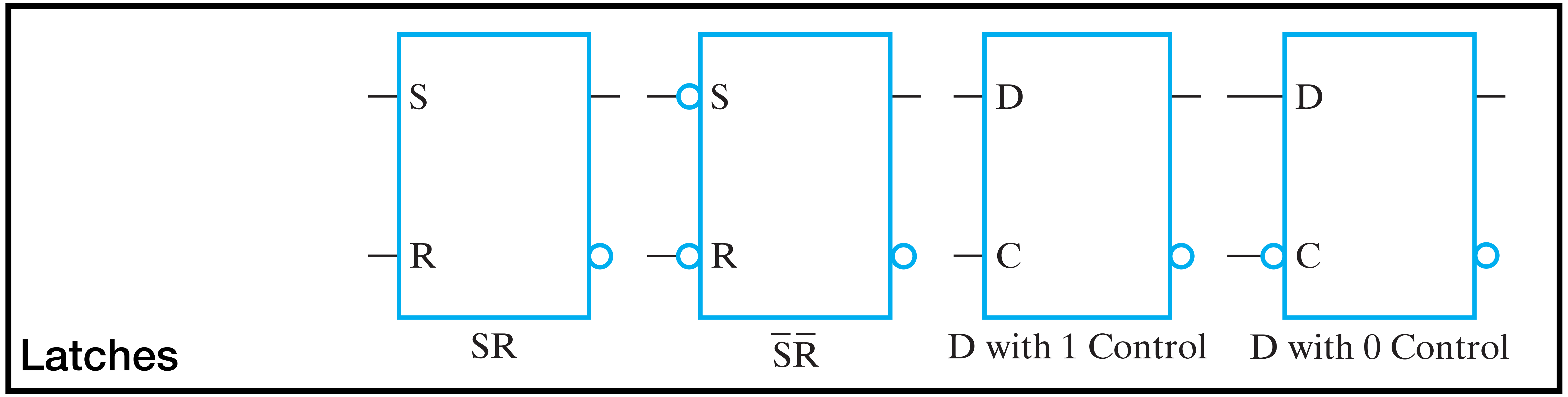
⊔ Triggered SR

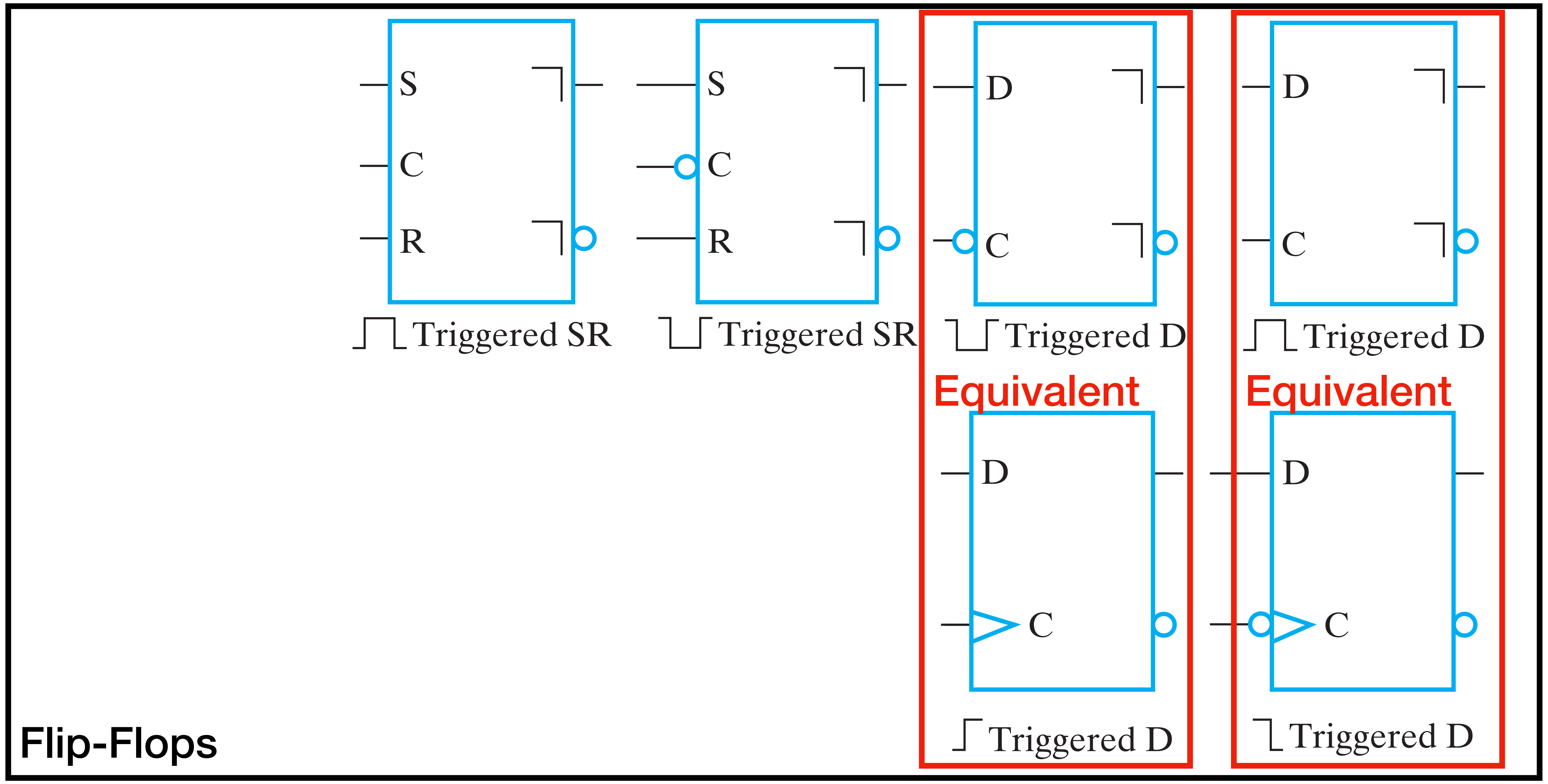
↑ Triggered D
⊔ Triggered D

↓ Triggered D
⎍ Triggered D

Concept

Summary





Demo