Sequential Circuits

Prof. Usagi

Summary of what we have seen so far

- Transistors
- Boolean algebra
- Basic gates
- Logic functions and truth tables
- Canonical forms (SOP and POS)
- Two-level logic minimization
- Kmaps
- Decoders
- MUXes
- Multiplexers (behavior and how to implement logic functions with them)
- Adders, subtractors, and other ALU components
- All above are “combinational circuits”!

Recap: Combinational v.s. sequential logic

- Combinational logic
  - The output is a pure function of its current inputs
  - The output doesn’t change regardless how many times the logic is triggered — Idempotent
- Sequential logic
  - The output depends on current inputs, previous inputs, their history

Sequential circuit has memory!

Recap: Theory behind each

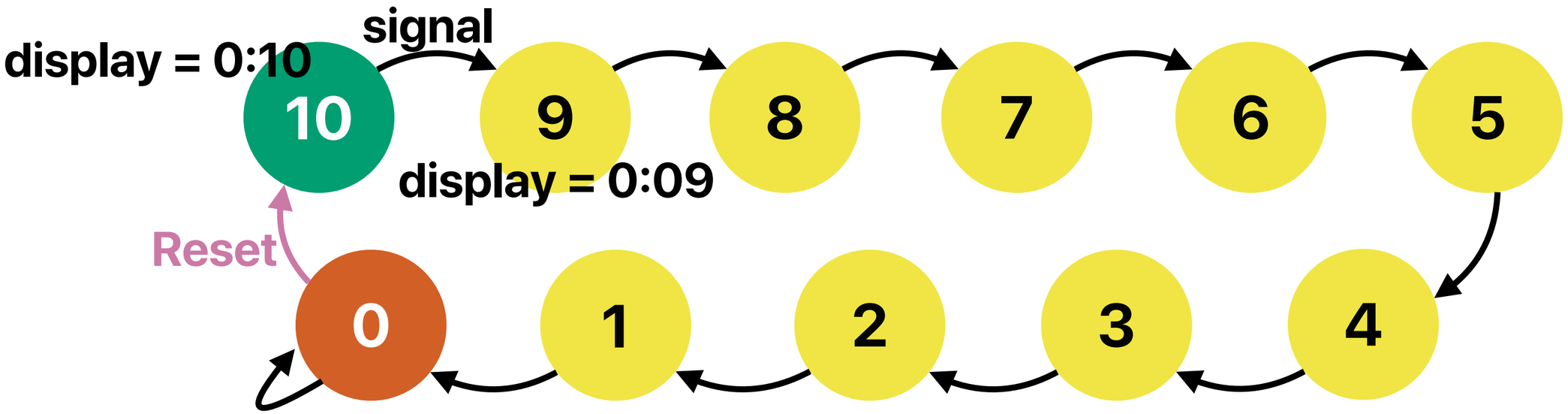
- A **Combinational logic** is the implementation of a **Boolean Algebra** function with only Boolean Variables as their inputs
- A **Sequential logic** is the implementation of a **Finite-State Machine**

Outline

- Finite-State Machines
- Introduction to sequential circuits

Count-down Timer

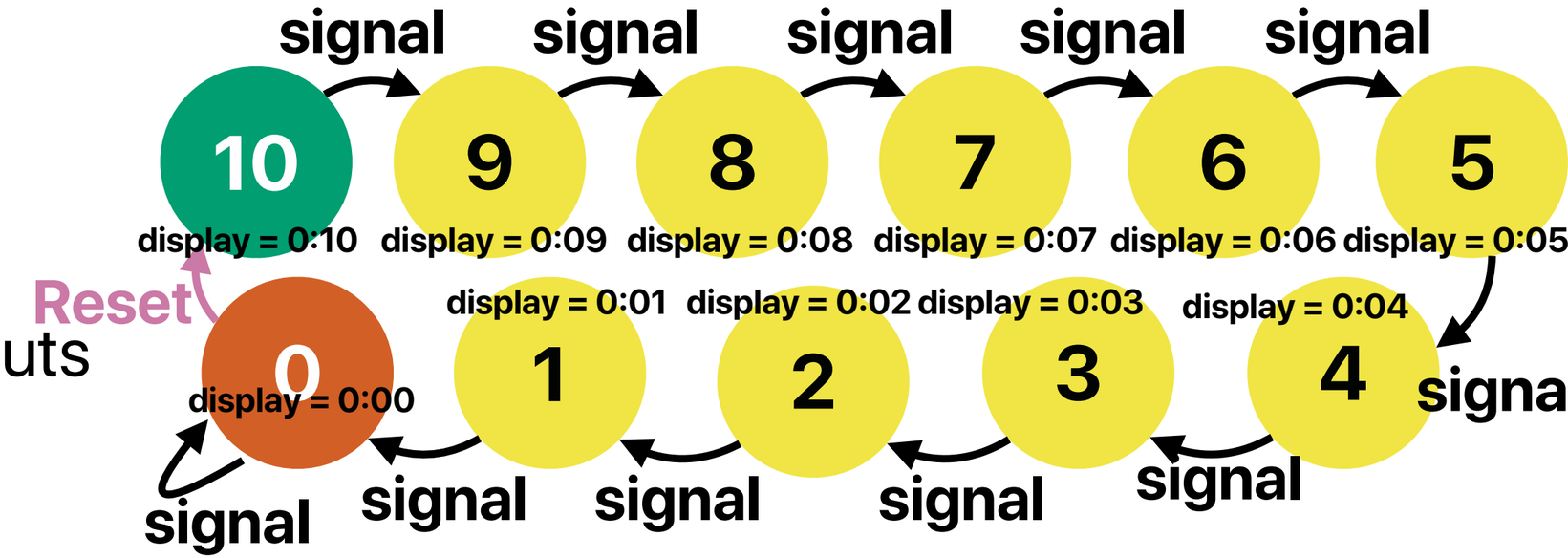
- What do we need to implement this timer?
  - Set an initial value/"state" of the timer
  - "Signal" the design every second
  - The design changes its "state" every time we received the signal until we reaches "0" — the final state



Finite-State Machines

Finite State Machines

- FSM consists of
  - Set of states
  - Set of inputs, set of outputs
  - Initial state
  - Set of transitions
    - Only one can be true at a time
- FSM representations:
  - State diagram
  - State table



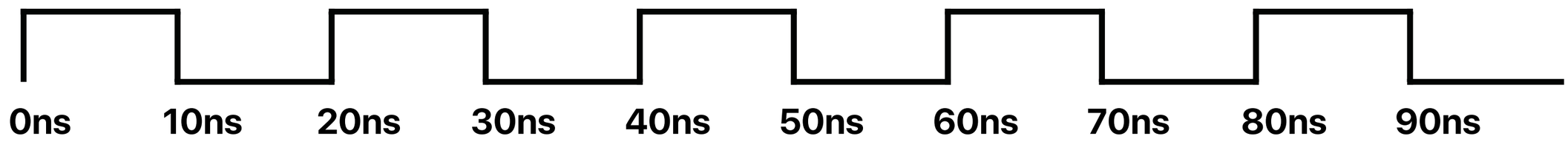
Current State	Next State Signal	
	0	1
10	10	9
9	9	8
8	8	7
7	7	6
6	6	5
5	5	4
4	4	3
3	3	2
2	2	1
1	1	0
0	0	0

How make FSM true?

What do we need to physically implement the timer?

- A set of logic to display the remaining time **— we know how to do this already**
- A logic to keep track of the "current state" **— memory**
- A set of logic that uses the "current state" and "a new input" to transit to a new state and generate the output **— we also know how to build this**
- A control signal that helps us to transit to the right state at the right time **— clock**

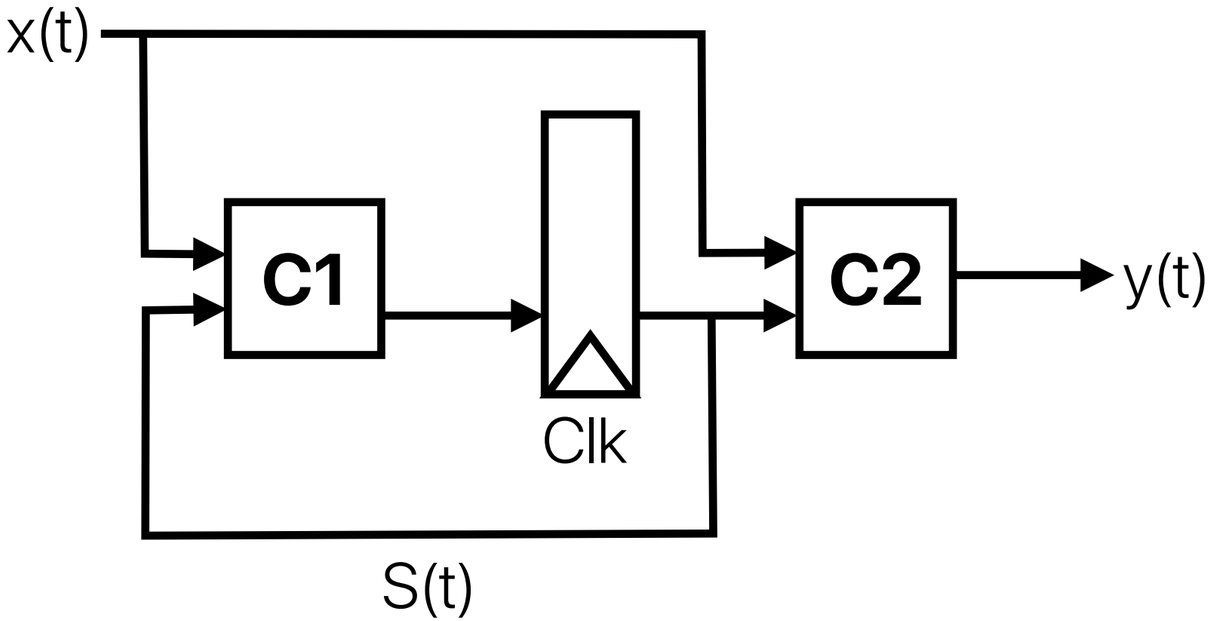
Clock signal



- Clock -- Pulsing signal for enabling latches; ticks like a clock
- Synchronous circuit: sequential circuit with a clock
- Clock period: time between pulse starts
  - Above signal: period = 20 ns
- Clock cycle: one such time interval
  - Above signal shows 3.5 clock cycles
- Clock duty cycle: time clock is high
  - 50% in this case
- Clock frequency: 1/period
  - Above : freq = 1 / 20ns = 50MHz;

Concept of Sequential Network

Mealy Machine

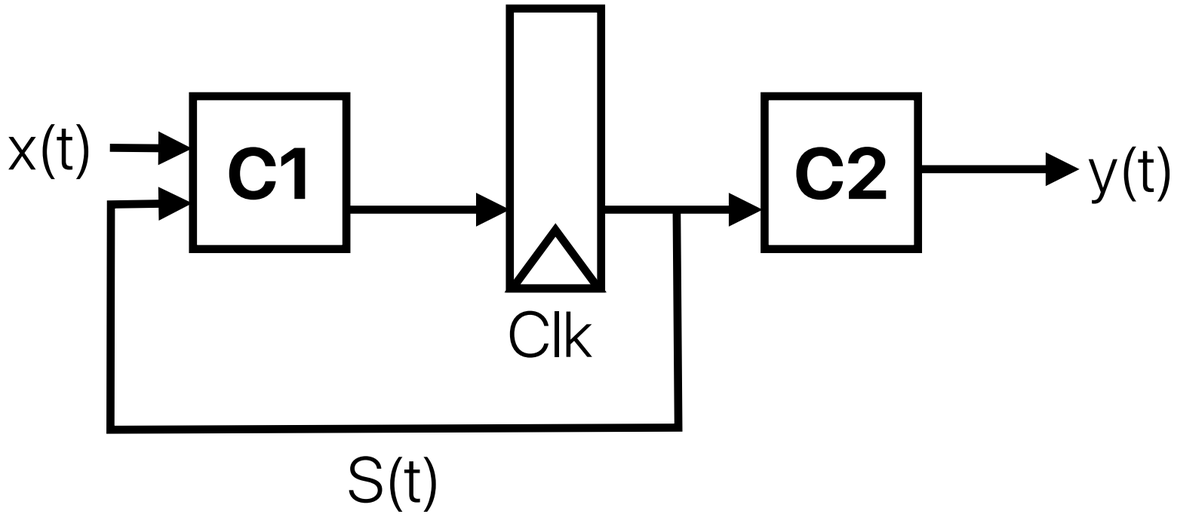


$y_i(t) = f_i(x(t), S(t))$

Result depends on both input and the current state

Life on Mars!

Moore Machine



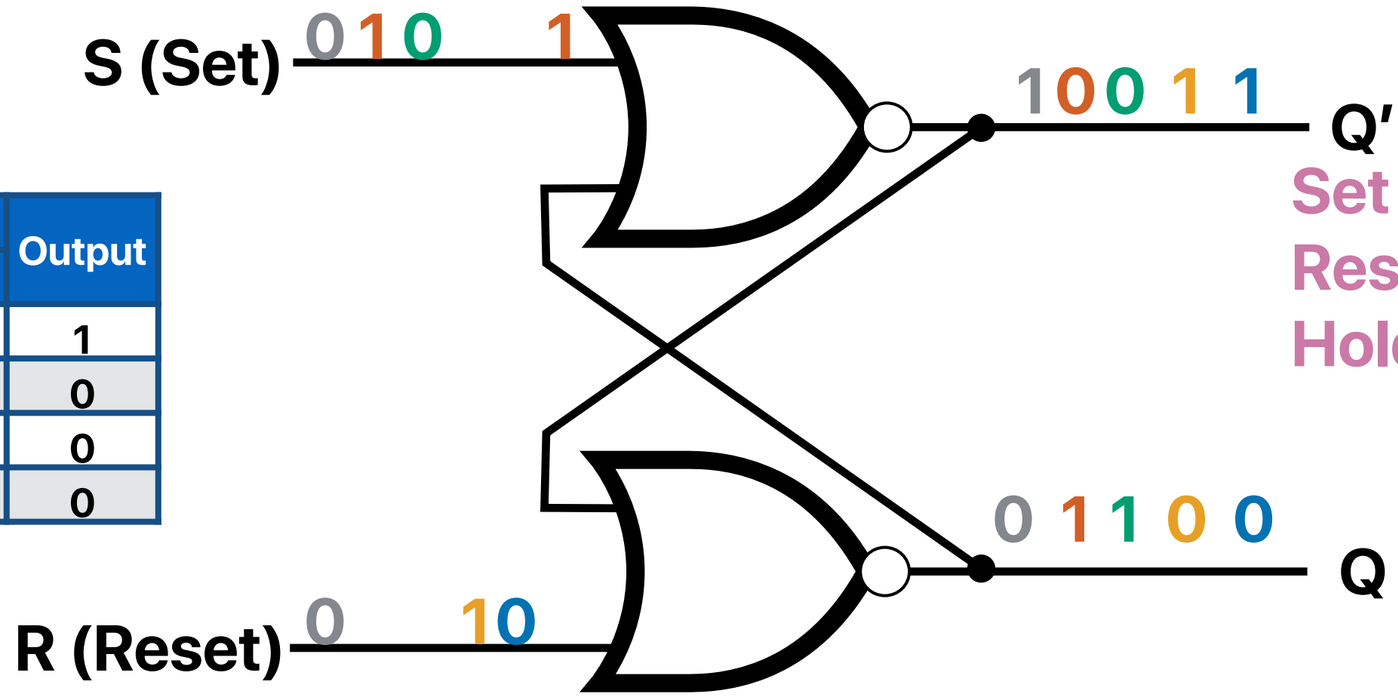
$y_i(t) = f_i(S(t))$

Result only depends on the current state

Like the timer

The basic form of memory

SR-Latch: the very basic “memory”



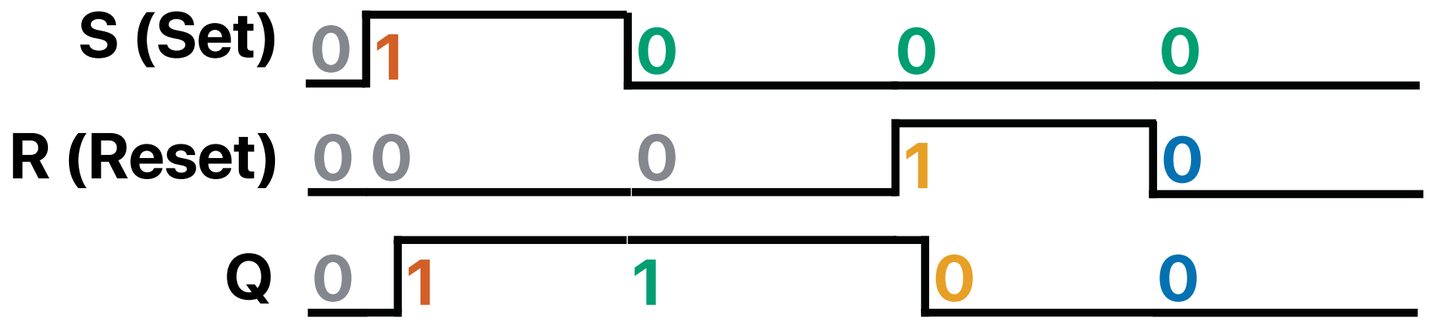
Input		Output
A	B	
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**0**

Set — Make the “stored bit 1”

Reset — Make the “stored bit 0”

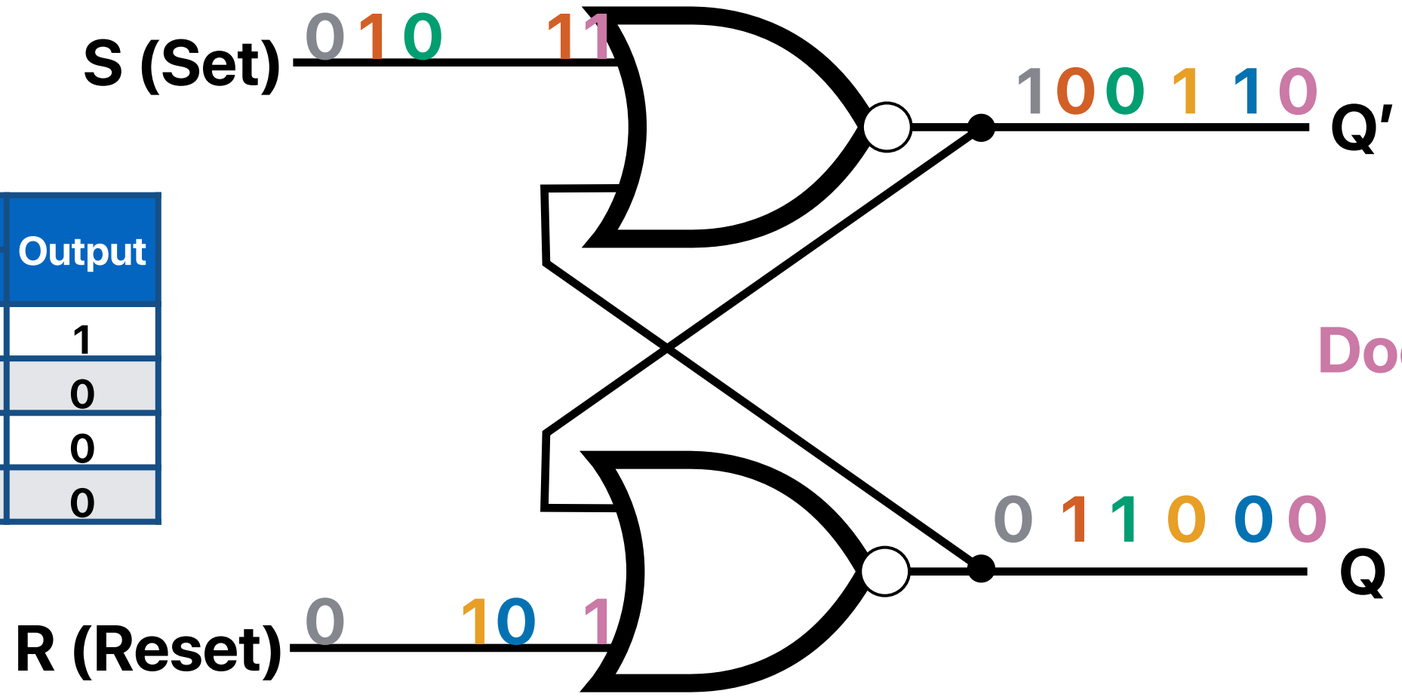
Hold — both set/reset are 0

The circuit has memory!



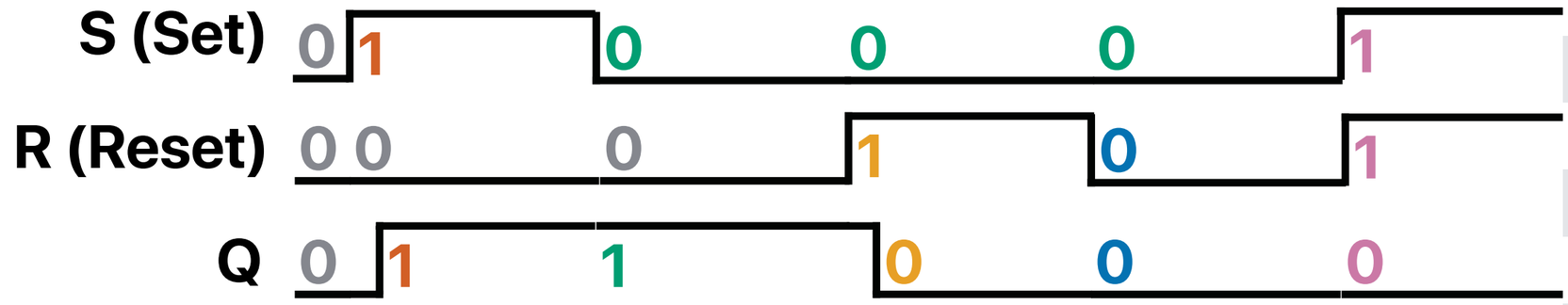
S	R	Q(t)	Q(t+1)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	
1	1	1	

What if S/R are both 1s?



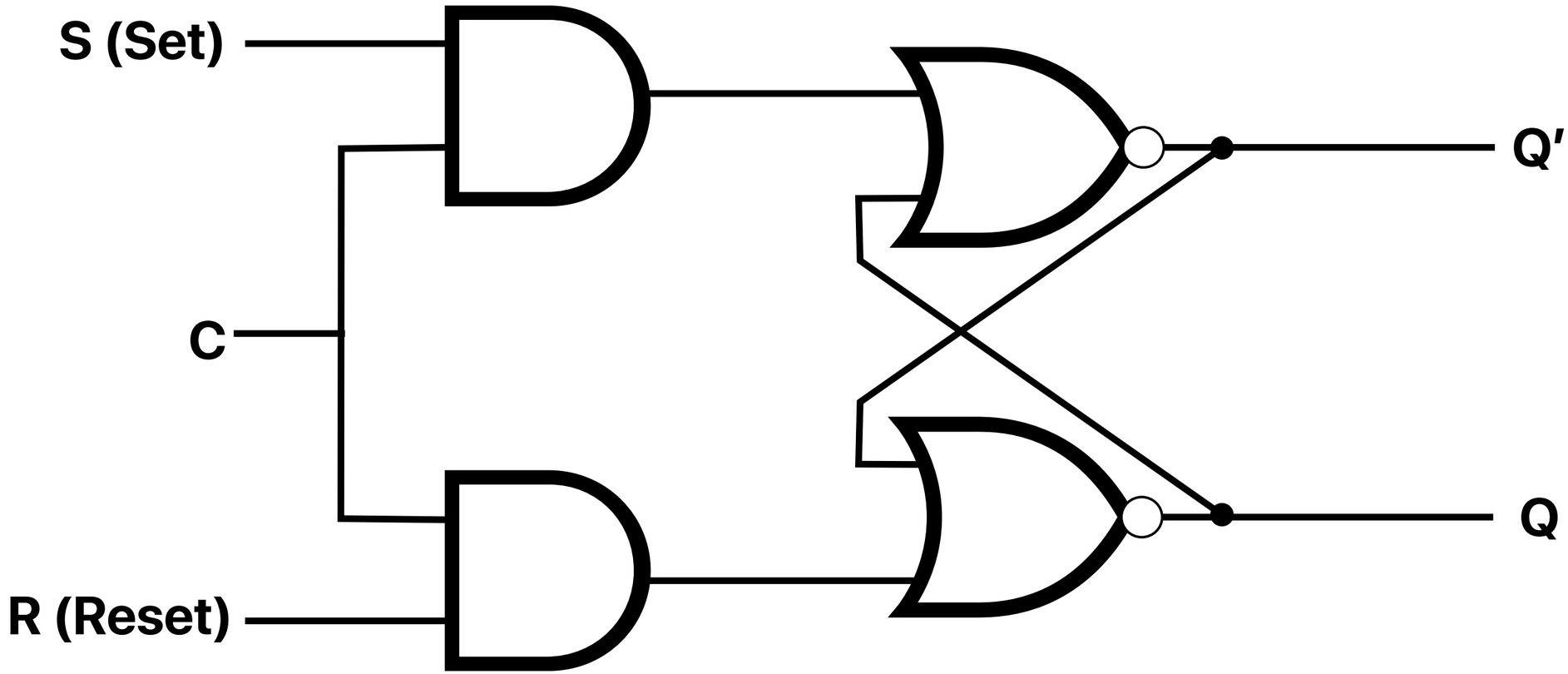
Input		Output
A	B	
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**0**

Doesn't function if both are 1s!



S	R	Q(t)	Q(t+1)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

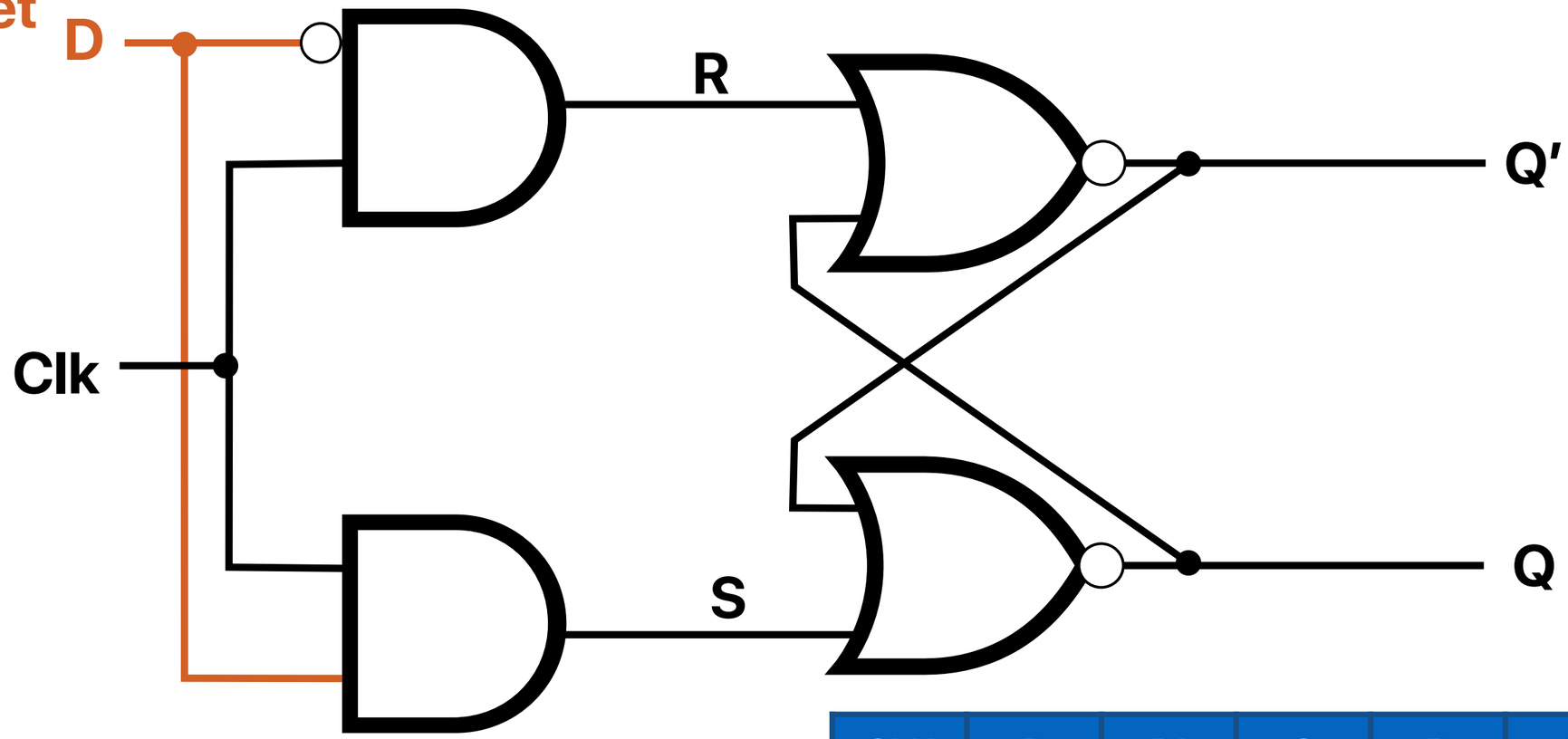
Add an input! — Level-Sensitive SR Latch



- Change C to 1 only after S and R are stable
- C is usually a clock (CLK)

D-Latch

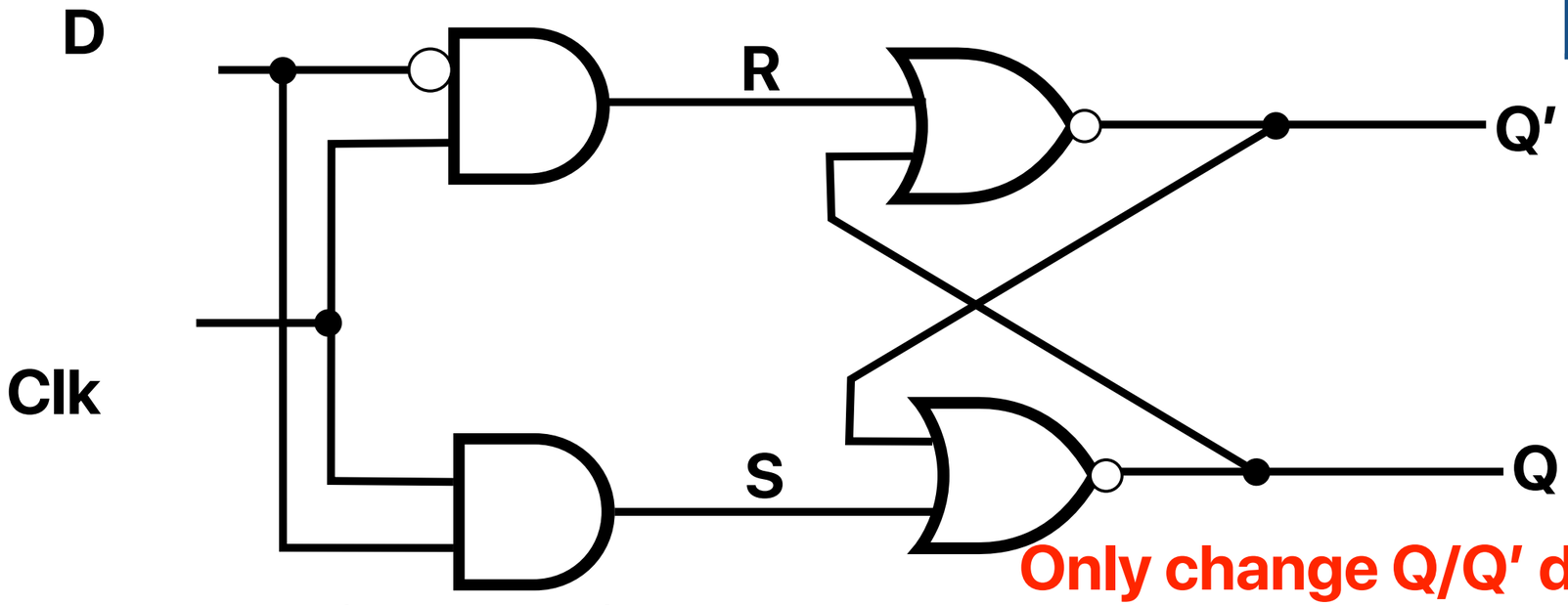
We will never get 1, 1 in this way



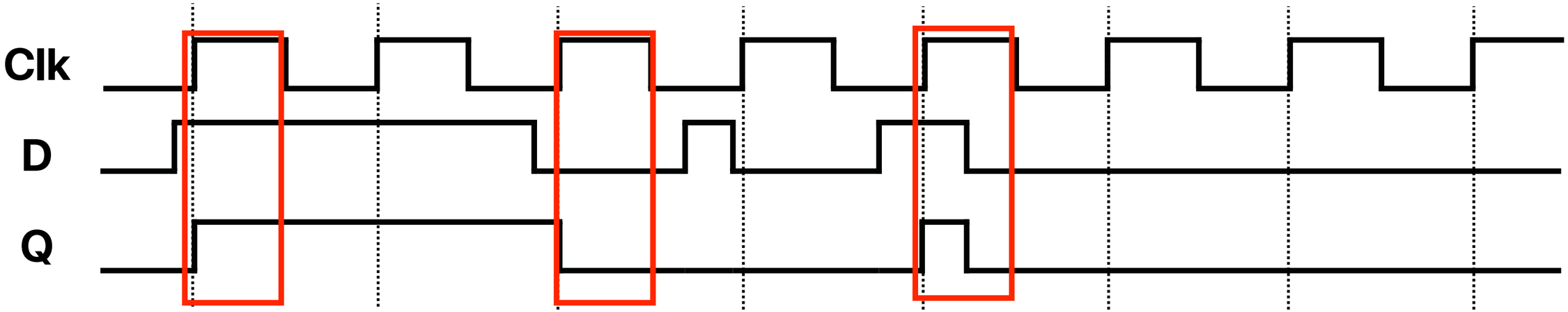
CLK	D	D′	S	R	Q	Q′
0	X	X′	0	0	Qprev	Qprev′
0	0	1	0	1	0	1
1	1	0	1	0	1	0

D-Latch

CLK	D	D′	S	R	Q	Q′
0	X	**X′**	**0**	**0**	**Qprev**	**Qprev′**
0	0	**1**	**0**	**1**	**0**	**1**
1	1	**0**	**1**	**0**	**1**	**0**

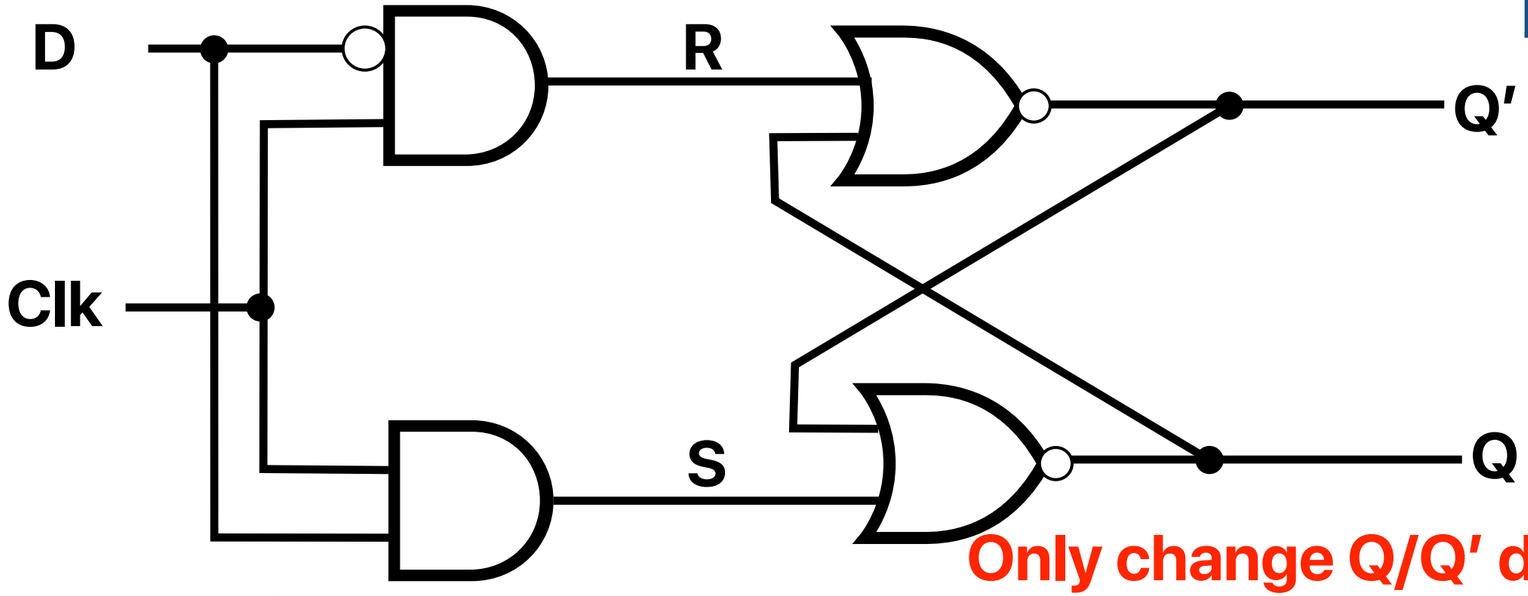


Only change Q/Q′ during positive clock edges

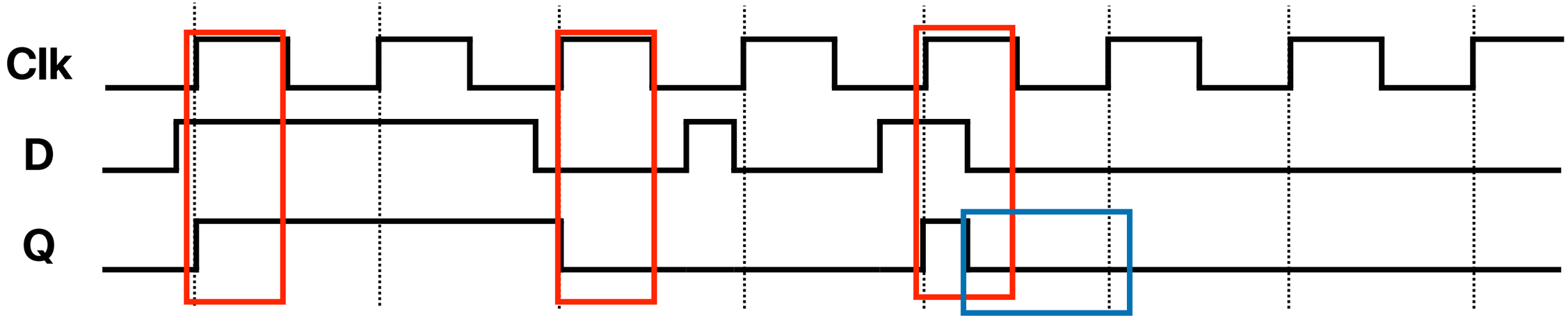


D-Latch

CLK	D	D′	S	R	Q	Q′
0	X	X′	0	0	Qprev	Qprev′
0	0	1	0	1	0	1
1	1	0	1	0	1	0

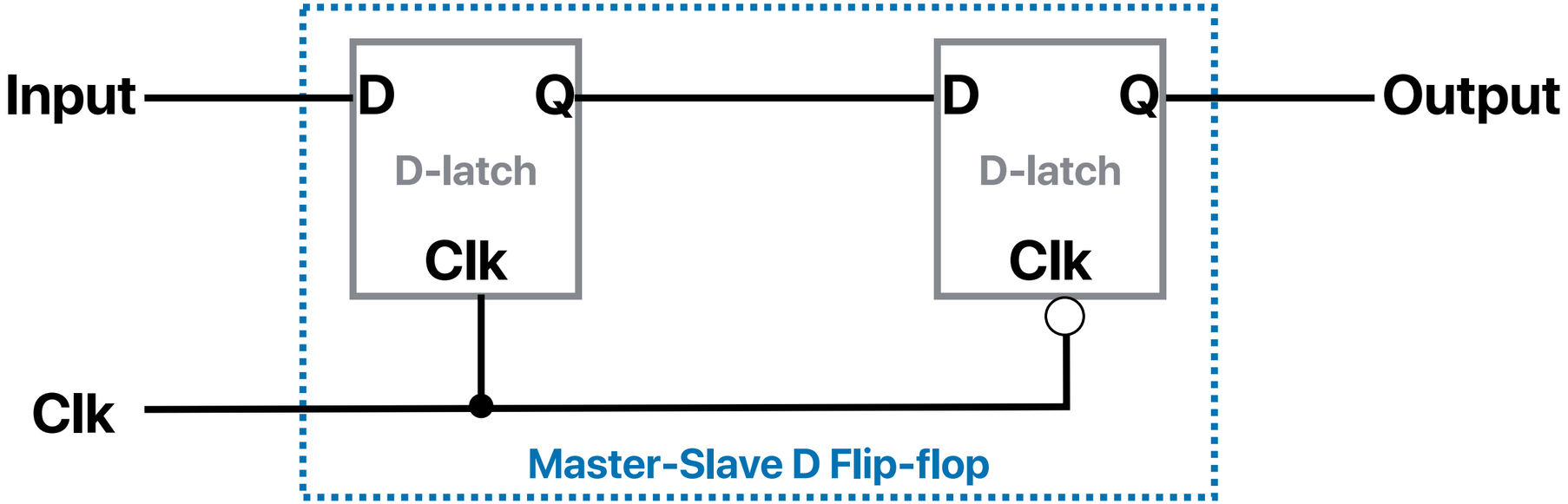


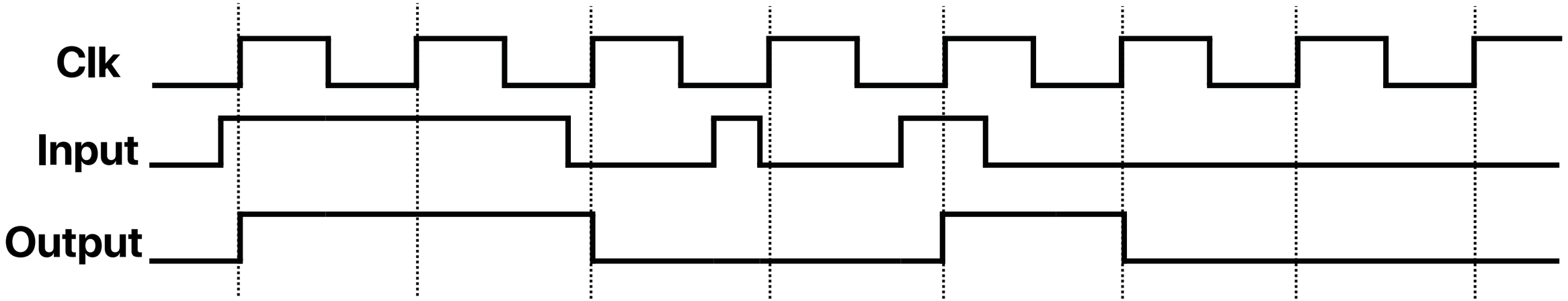
Only change Q/Q′ during positive clock edges



Output doesn’t hold for the whole cycle

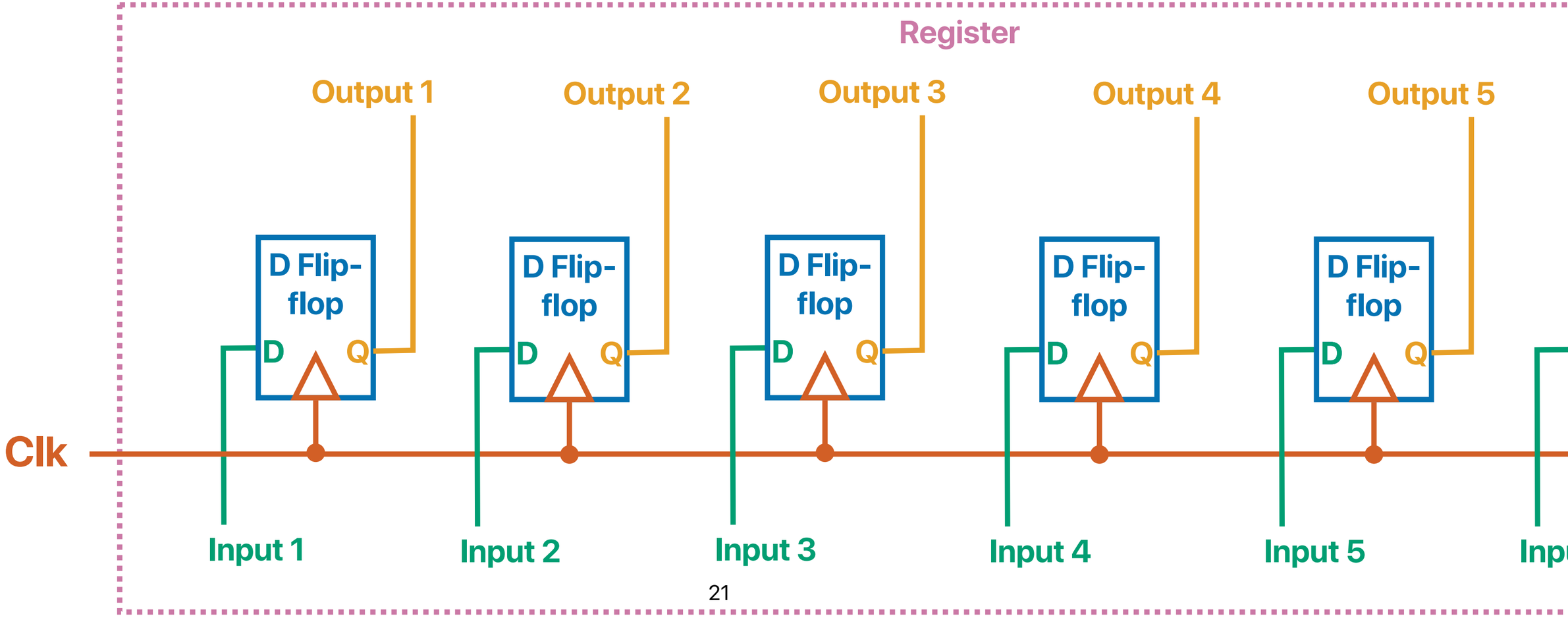
D flip-flop


Input
D
Q
D-latch
Clk
D
Q
D-latch
Clk
Output
Clk
Master-Slave D Flip-flop


Clk
Input
Output

Registers

- Register: a sequential component that can store multiple bits
- A basic register can be built simply by using multiple D-FFs



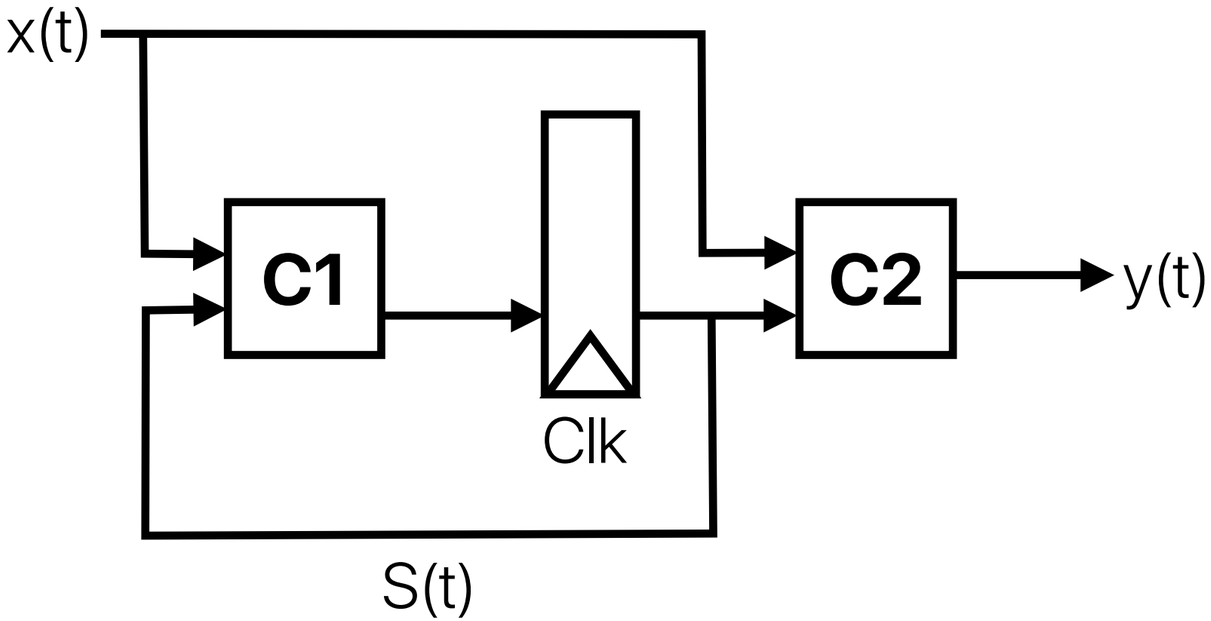
Let’s learn how to design sequential circuits!

Sequential Circuit Design Flow

- Input Output Relation
- State Diagram (Transition of states)
  - State minimization (Reduction)
  - Finite state machine partitioning
- State Assignment (Map states into binary code)
  - Binary code, Gray encoding, One hot encoding, Coding optimization
- State Table (Truth table of states)
- Excitation Table (Truth table of FF inputs)
  - K Map, Minimal Expression
  - Logic Diagram

Recap: Concept of Sequential Network

Mealy Machine

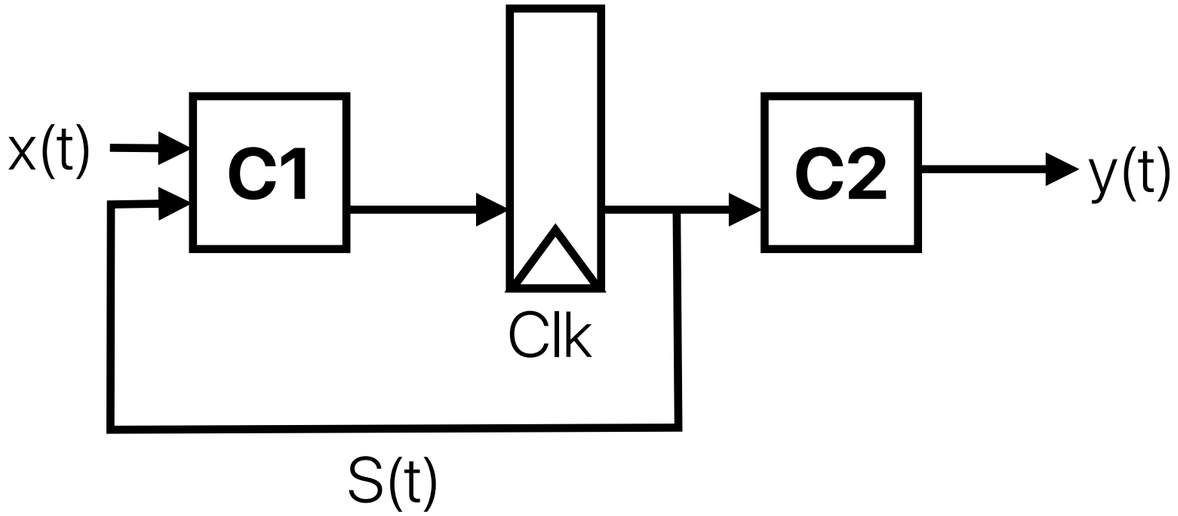


$y_i(t) = f_i(x(t), S(t))$

Result depends on both input and the current state

Life on Mars!

Moore Machine



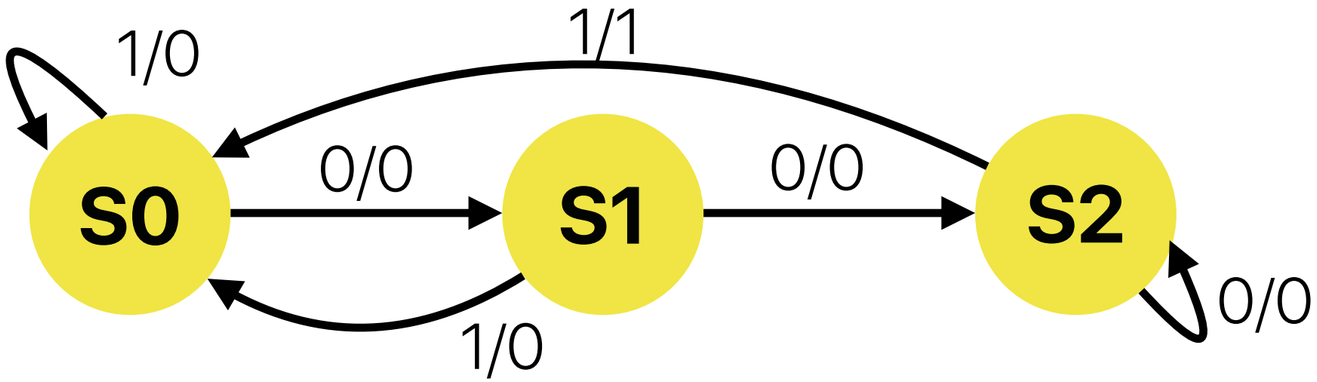
$y_i(t) = f_i(S(t))$

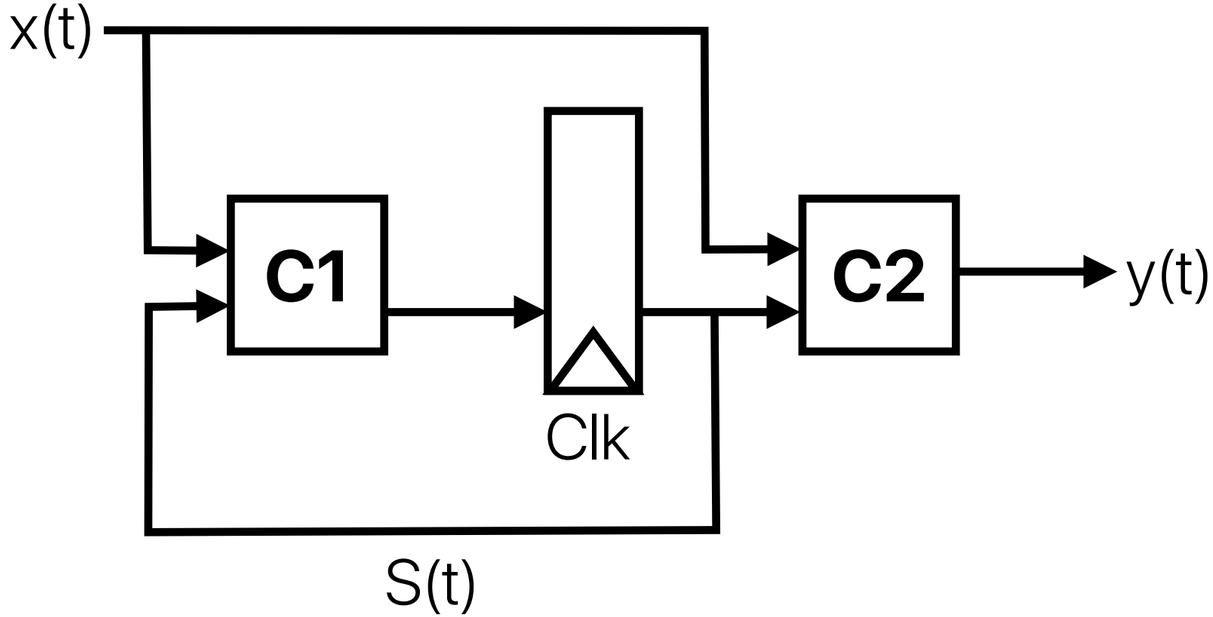
Result only depends on the current state

Like the timer

Life on Mars

State Diagram



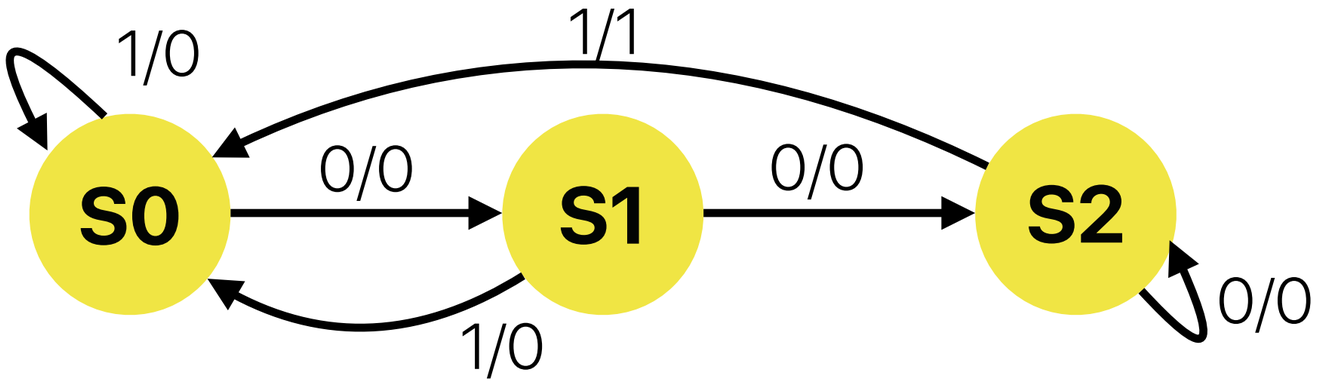


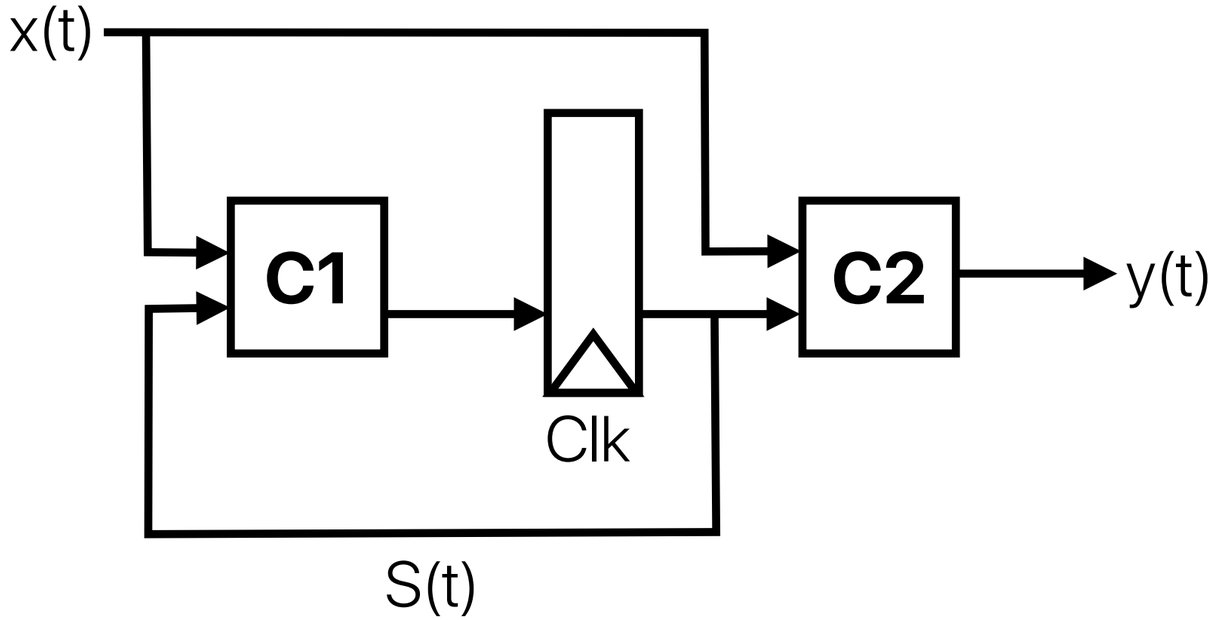
State Diagram

Current State	Next State Input	
	0	1
S0	S1, 0	S0, 0
S1	S2, 0	S0, 0
S2	S2, 0	S0, 1

Life on Mars

State Diagram





State Diagram

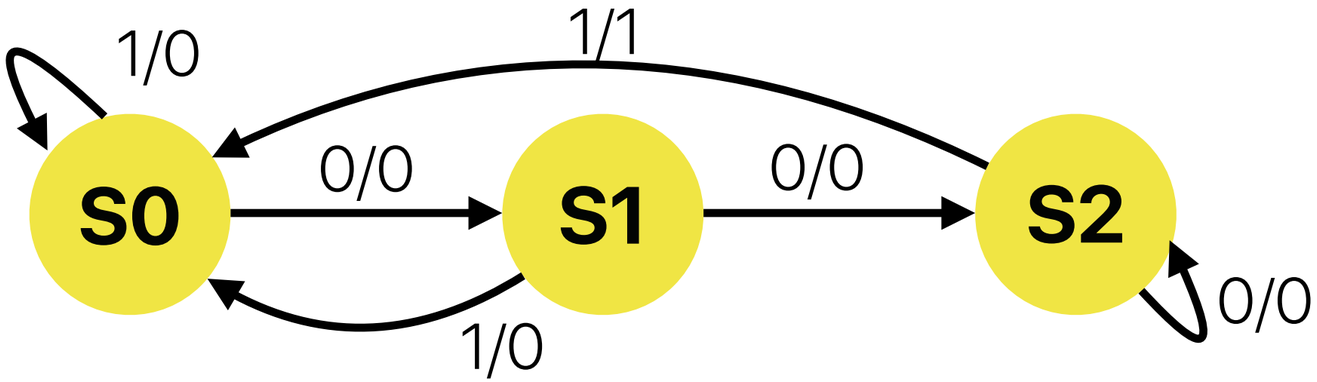
Current State	Next State Input	
	0	1
S0	S1, 0	S0, 0
S1	S2, 0	S0, 0
S2	S2, 0	S0, 1

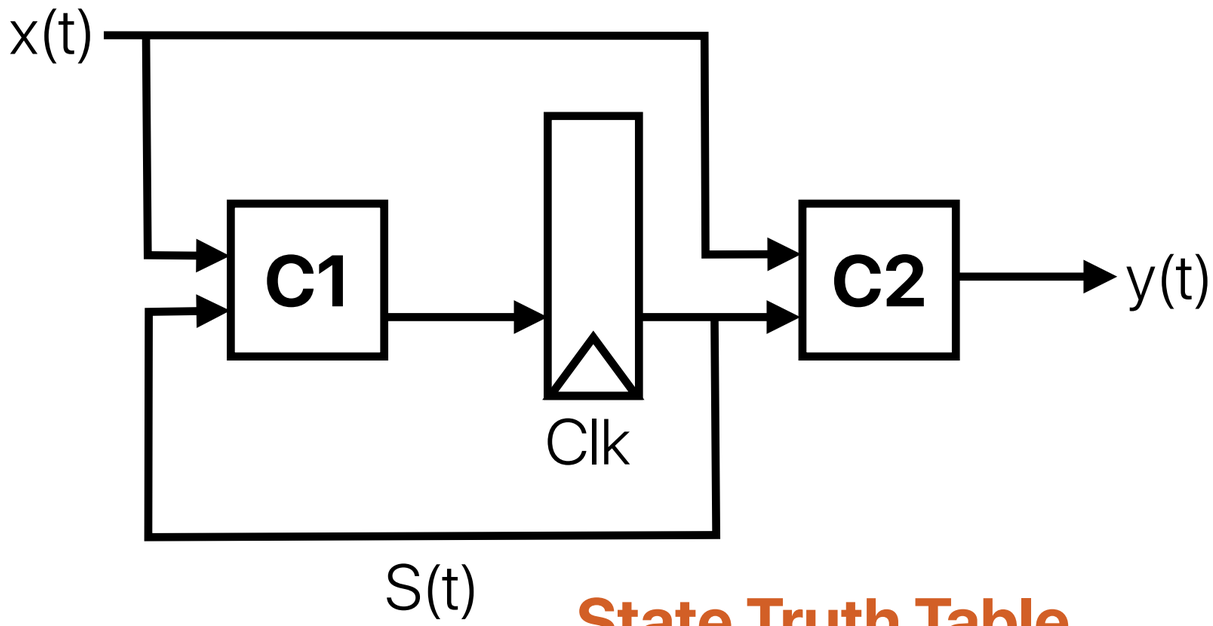
State Assignment

S0	00
S1	01
S2	10

Life on Mars

State Diagram





State Diagram

Current State	Next State Input	
	0	1
S0	S1, 0	S0, 0
S1	S2, 0	S0, 0
S2	S2, 0	S0, 1

State Assignment

S0	00
S1	01
S2	10

State Truth Table

State\Input	0	1
00	01, 0	00, 0
01	10, 0	00, 0
10	10, 0	00, 1

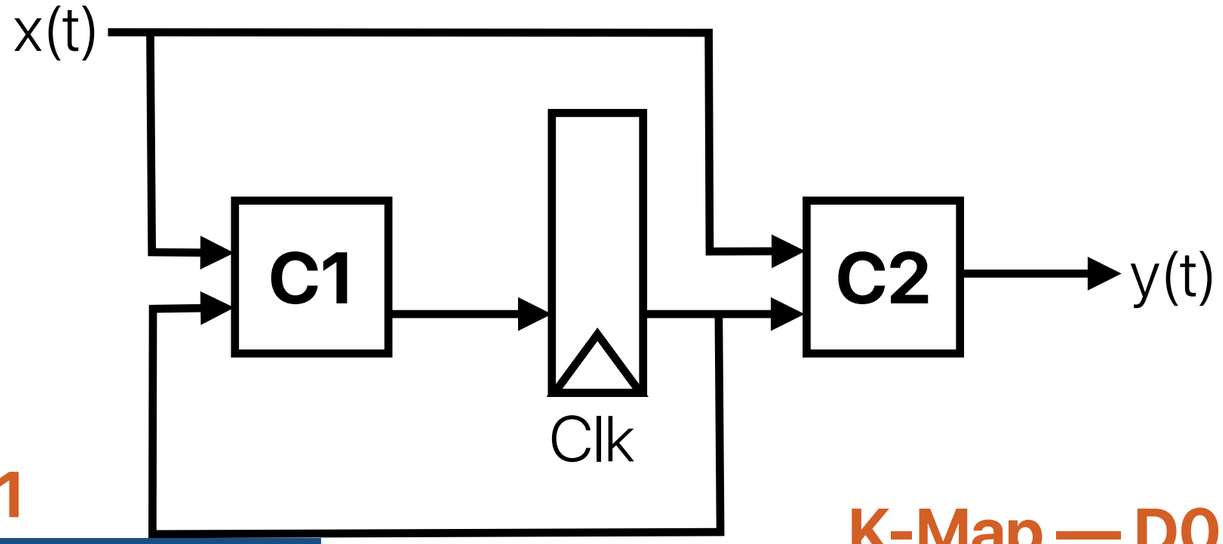
Life on Mars

State Truth Table

State\Input	0	1
00	01, 0	00, 0
01	10, 0	00, 0
10	10, 0	00, 1

Excitation Table

NextStateOput StateInput	D1	D0	y
000	0	1	0
001	0	0	0
010	1	0	0
011	0	0	0
100	1	0	0
101	0	0	1
110	X	X	X
111	X	X	X



K-Map — D1

	0,0	0,1	1,1	1,0
0	0	1	X	1
1	0	0	X	0

$D1 = x'Q_0 + x'Q_1$

K-Map — D0

	0,0	0,1	1,1	1,0
0	1	0	X	0
1	0	0	X	0

$D0 = Q_0'Q_1'x'$

K-Map — y

	0,0	0,1	1,1	1,0
0	0	0	X	0
1	0	0	X	1

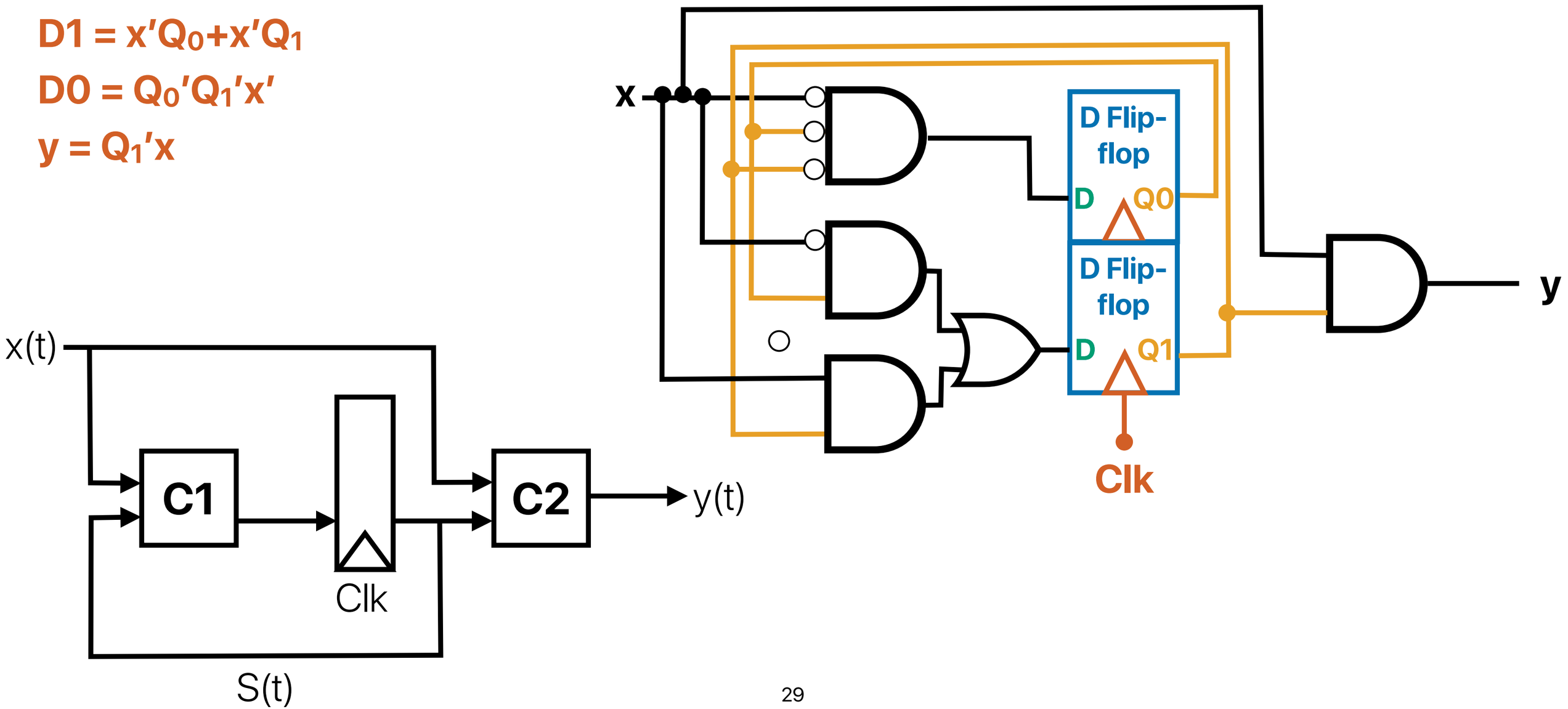
$y = Q_1'x$

Circuit — Life on Mars

$D1 = x'Q_0 + x'Q_1$

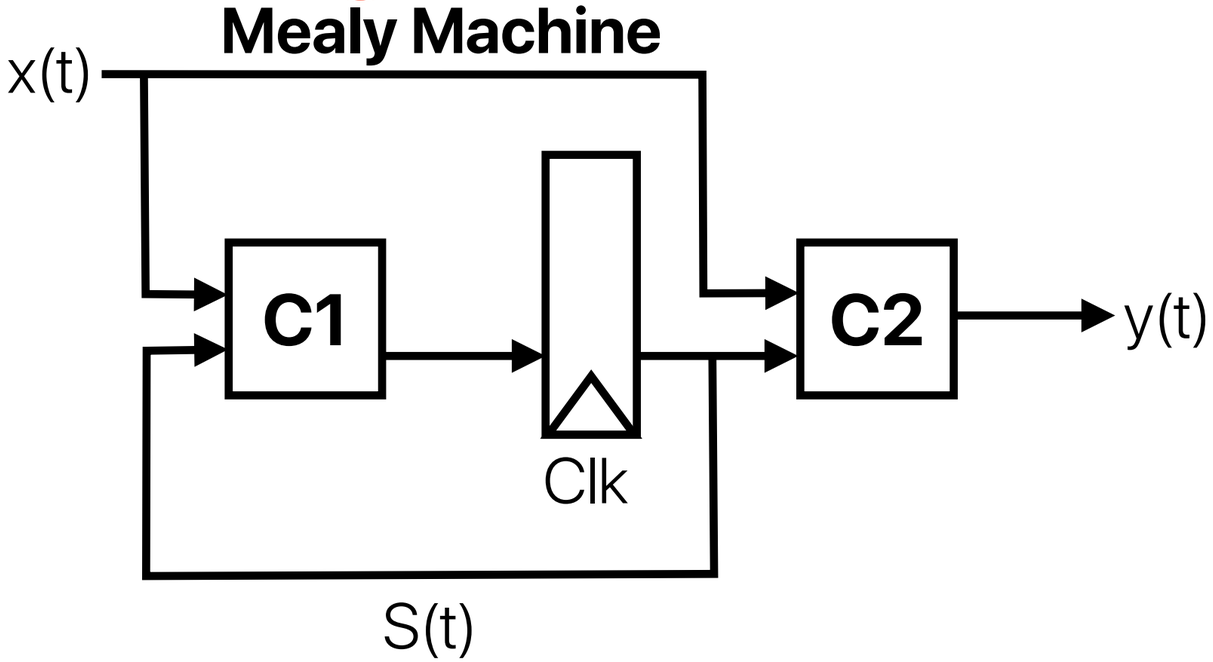
$D0 = Q_0'Q_1'x'$

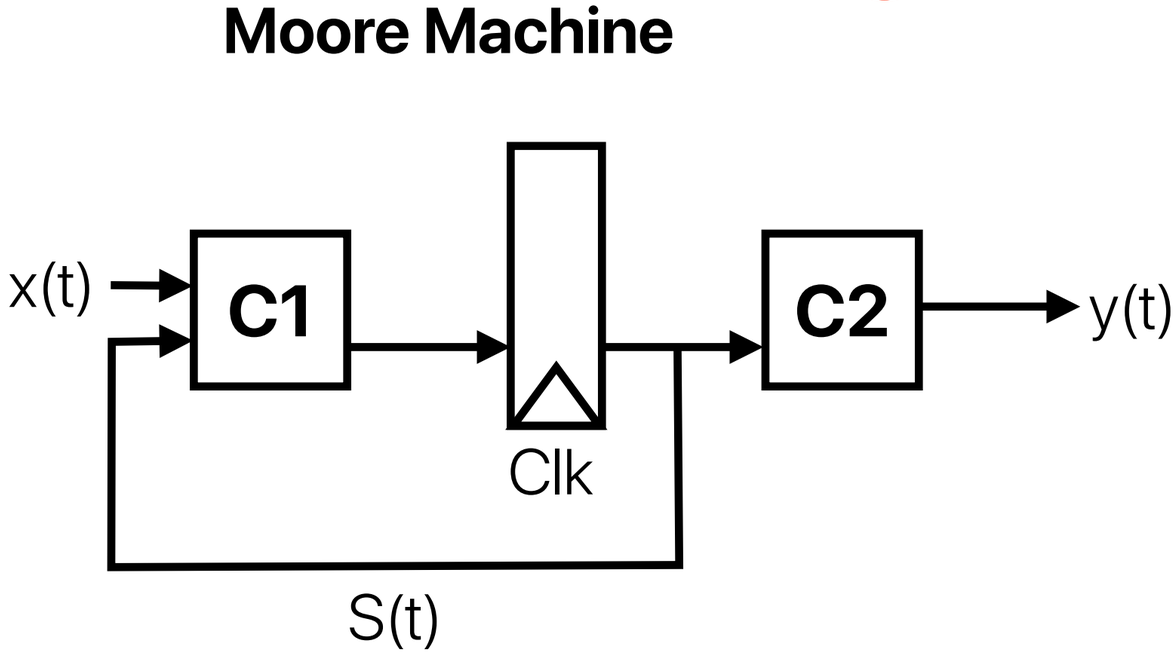
$y = Q_1'x$



Timing constraints on sequential circuits

Timing Constraints in Sequential Circuit Designs



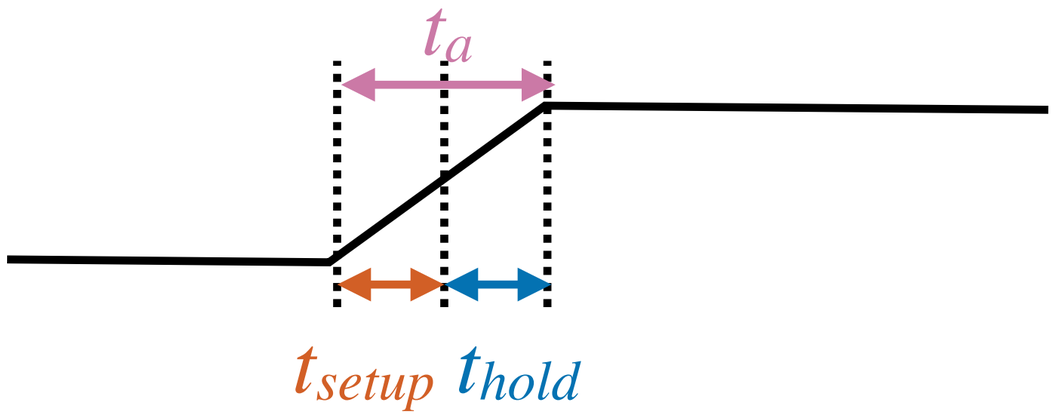


- A seemingly logically correct design can go wrong – signals don't travel in zero time
- We next look at timing constraints for combinational and sequential logic.

Combinational Logic Timing

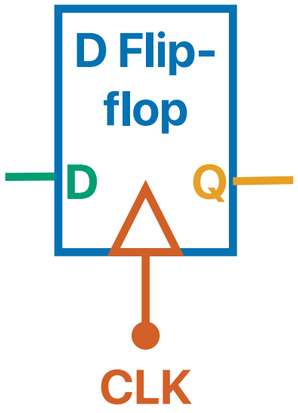
- Min delay of a gate, also called contamination delay: t_{cd}
  - Minimum time from when an input changes until the output starts to change
- Max delay of a gate, also called propagation delay: t_{pd}
  - Maximum time from when an input changes until the output is guaranteed to reach its final value (i.e., stop changing)

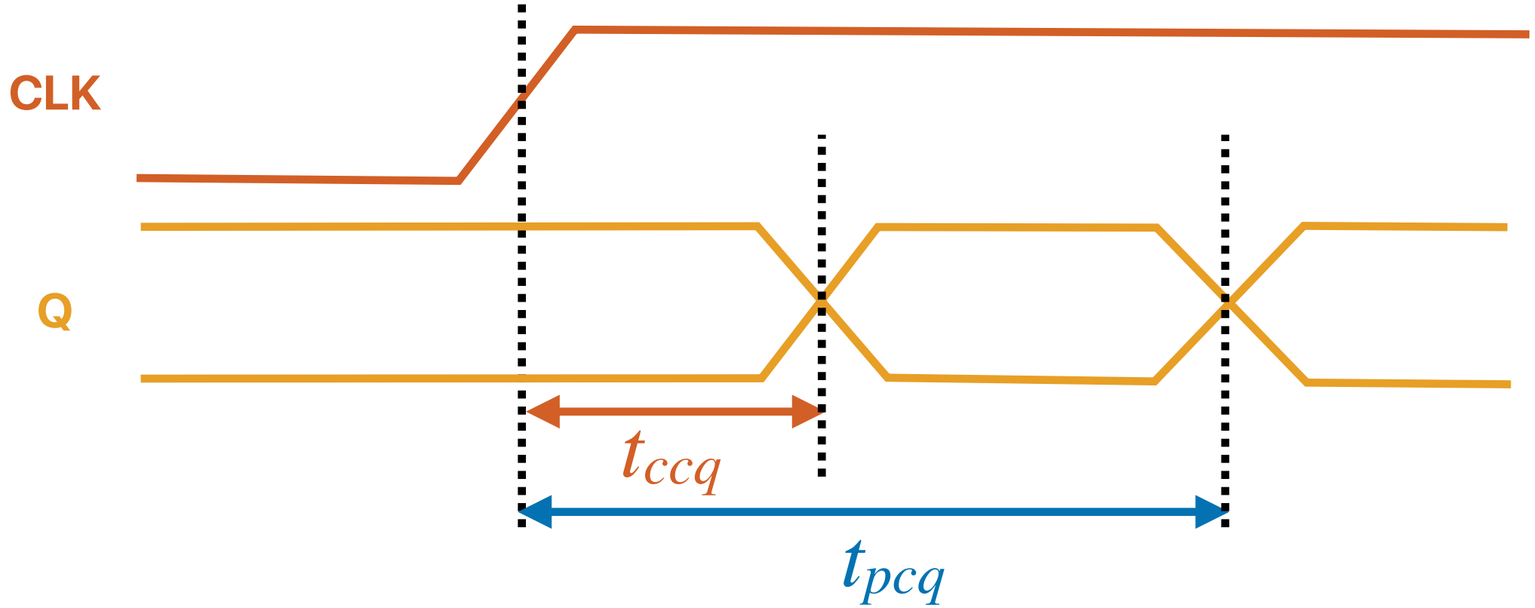
Setup and hold times



- Setup time: t_{setup}
  - Time before the clock edge that data must be stable (i.e. not change)
- Hold time: t_{hold}
  - Time after the clock edge that data must be stable
- Aperture time: t_a
  - Time around clock edge that data must be stable ($t_a = t_{setup} + t_{hold}$)

Output Timing Constraints

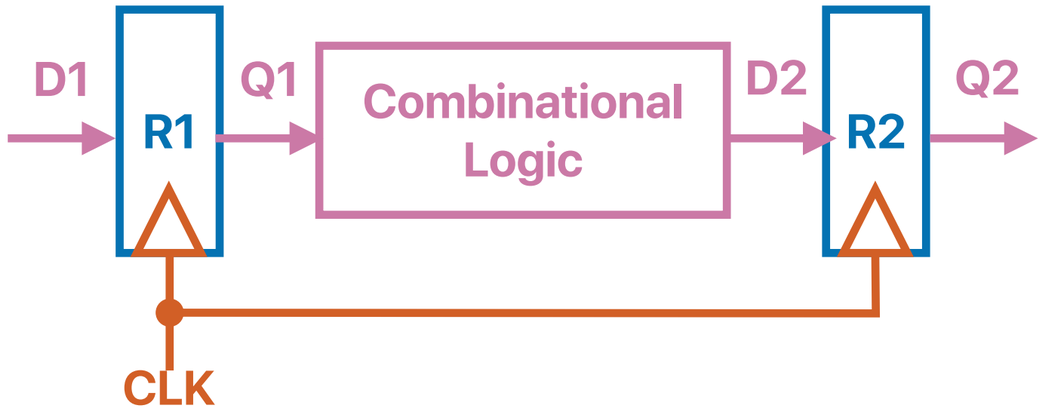




- Min delay of FF, also called contamination delay or min CLK to Q delay: t_{ccq}
  - Time after clock edge that Q might be unstable (i.e., starts changing)
- Max delay of FF, also called propagation delay or maximum CLK to Q delay: t_{pcq}
  - Time after clock edge that the output Q is guaranteed to be stable (i.e. stops changing)

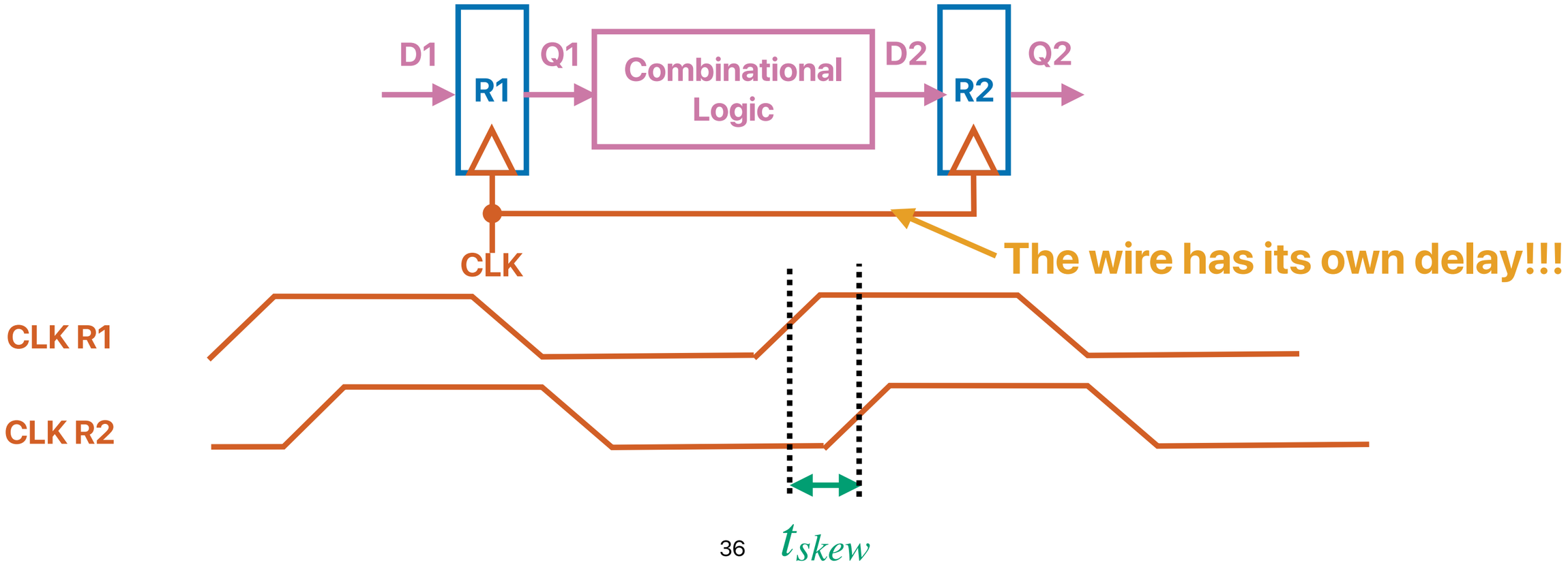
FF Timing Parameters

- Once a flip flop has been built, its timing characteristics stay fixed: t_{setup} , t_{hold} , t_{ccq} , t_{pcq}
- What about the clock? Does the clock edge arrive at the same time to all the D-FFs on the chip?



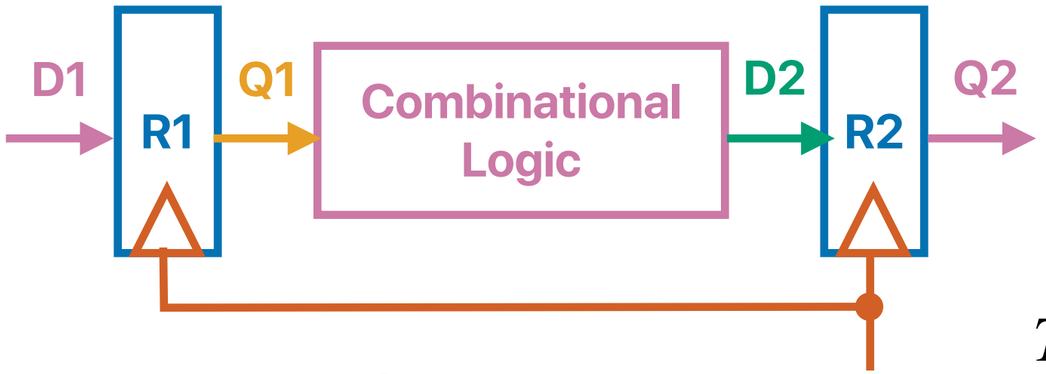
Clock Skew

- The clock doesn't arrive at all registers at the same time
- **Skew**: difference between the two clock edges
- Perform the **worst case analysis**



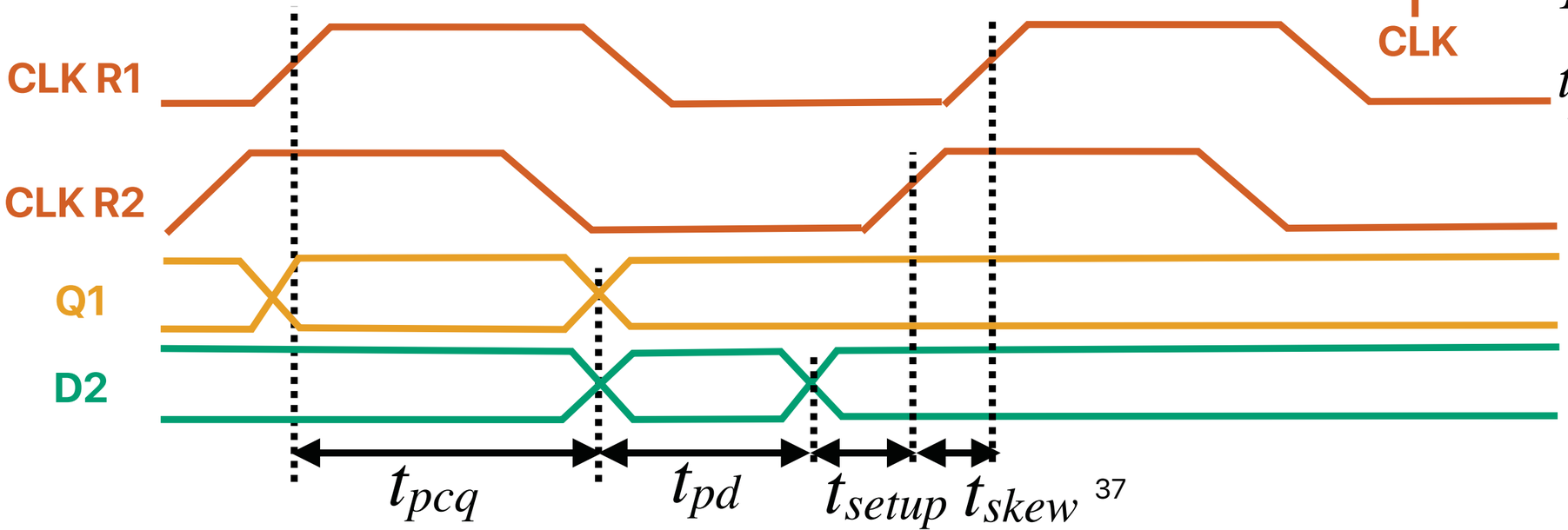
Setup Time Constraint with Skew

- In the worst case, CLK2 is earlier than CLK1
- t_{pcq} is max delay through FF, t_{pd} is max delay through logic



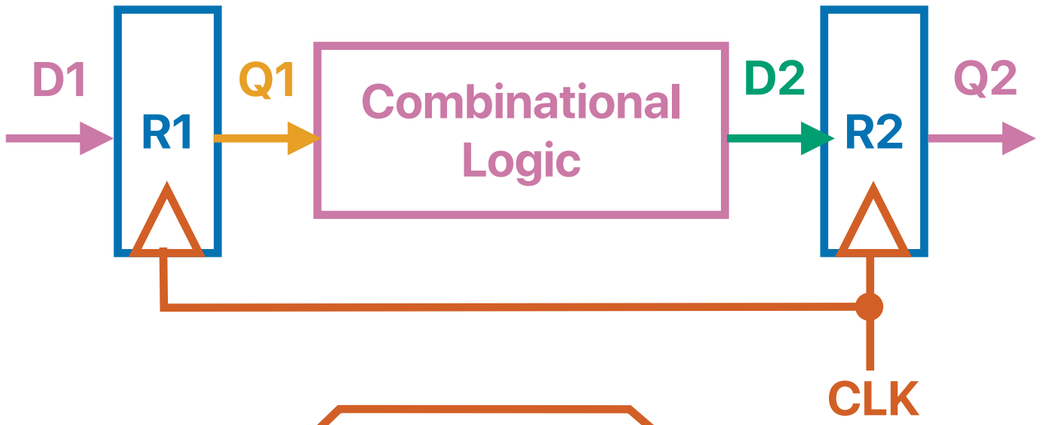
$$T_c \geq t_{pcq} + t_{pd} + t_{setup} + t_{skew}$$

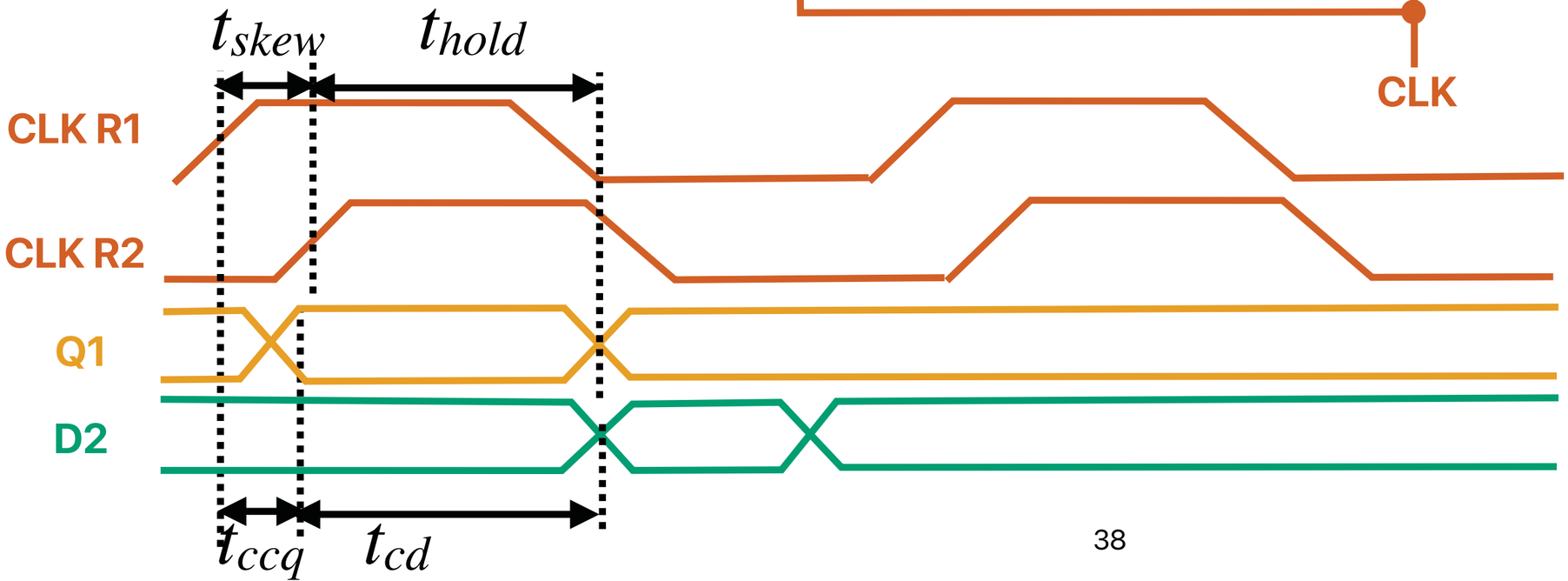
$$t_{pd} \leq T_c - (t_{setup} + t_{pcq} + t_{skew})$$



Hold Time Constraint with Skew

- In the worst case, CLK2 is later than CLK1
- t_{ccq} is min delay through FF, t_{cd} is min delay through logic



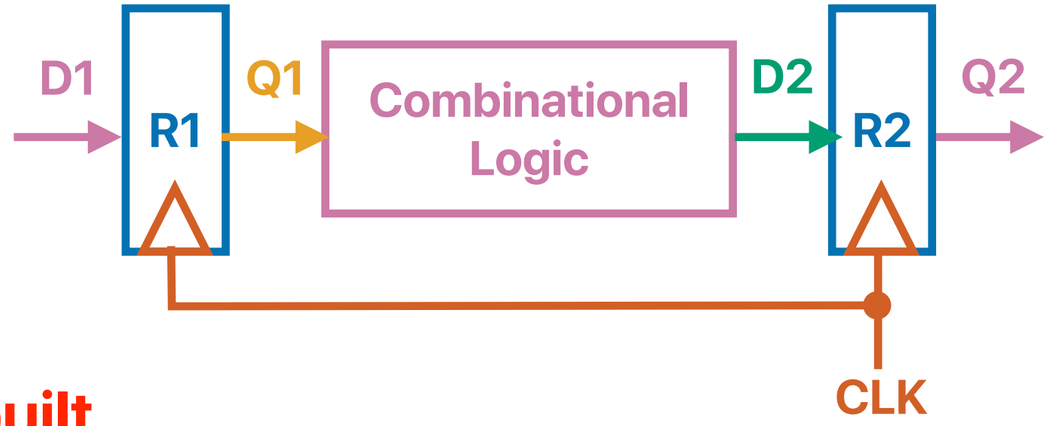


$$t_{ccq} + t_{cd} > t_{hold} + t_{skew}$$

$$t_{cd} > t_{hold} + t_{skew} - t_{ccq}$$

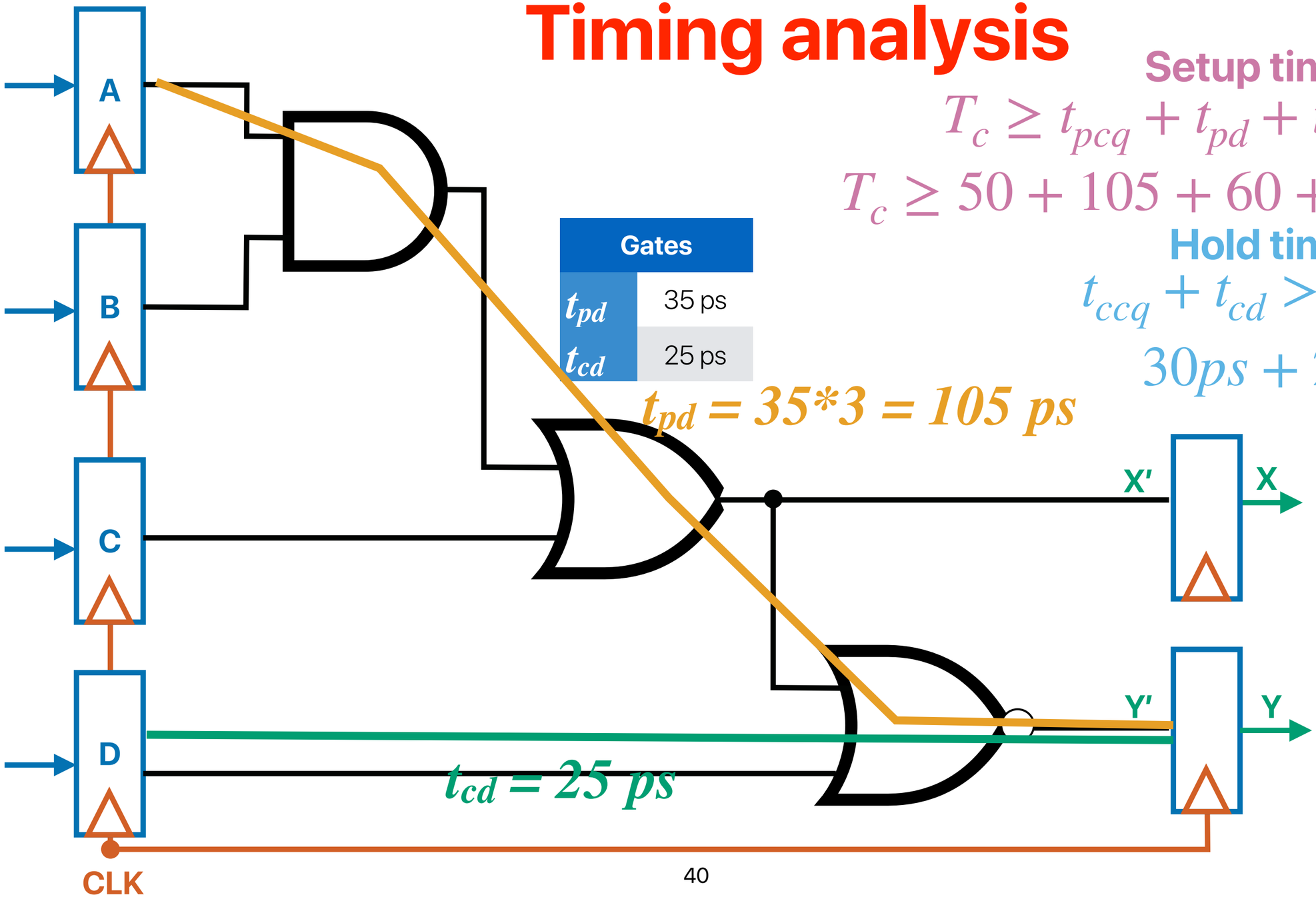
Summary on timing constraints

- Combinational:
  - Maximum delay = Propagation delay
  - Minimum delay = Contamination delay
- Flip Flops:
  - Input
    - Setup time
    - Hold time
  - Output
    - Propagation clock-to-Q time
    - Contamination clock-to-Q time



Once the logic/FFs are built, these timing characteristics are fixed properties

Timing analysis



Gates	
t_{pd}	35 ps
t_{cd}	25 ps

$t_{pd} = 35*3 = 105\ ps$

$t_{cd} = 25\ ps$

Setup time constraints

$$T_c \geq t_{pcq} + t_{pd} + t_{setup} + t_{skew}$$

$$T_c \geq 50 + 105 + 60 + 0 = 215ps$$

Hold time constraints

$$t_{ccq} + t_{cd} > t_{hold} + t_{skew}$$

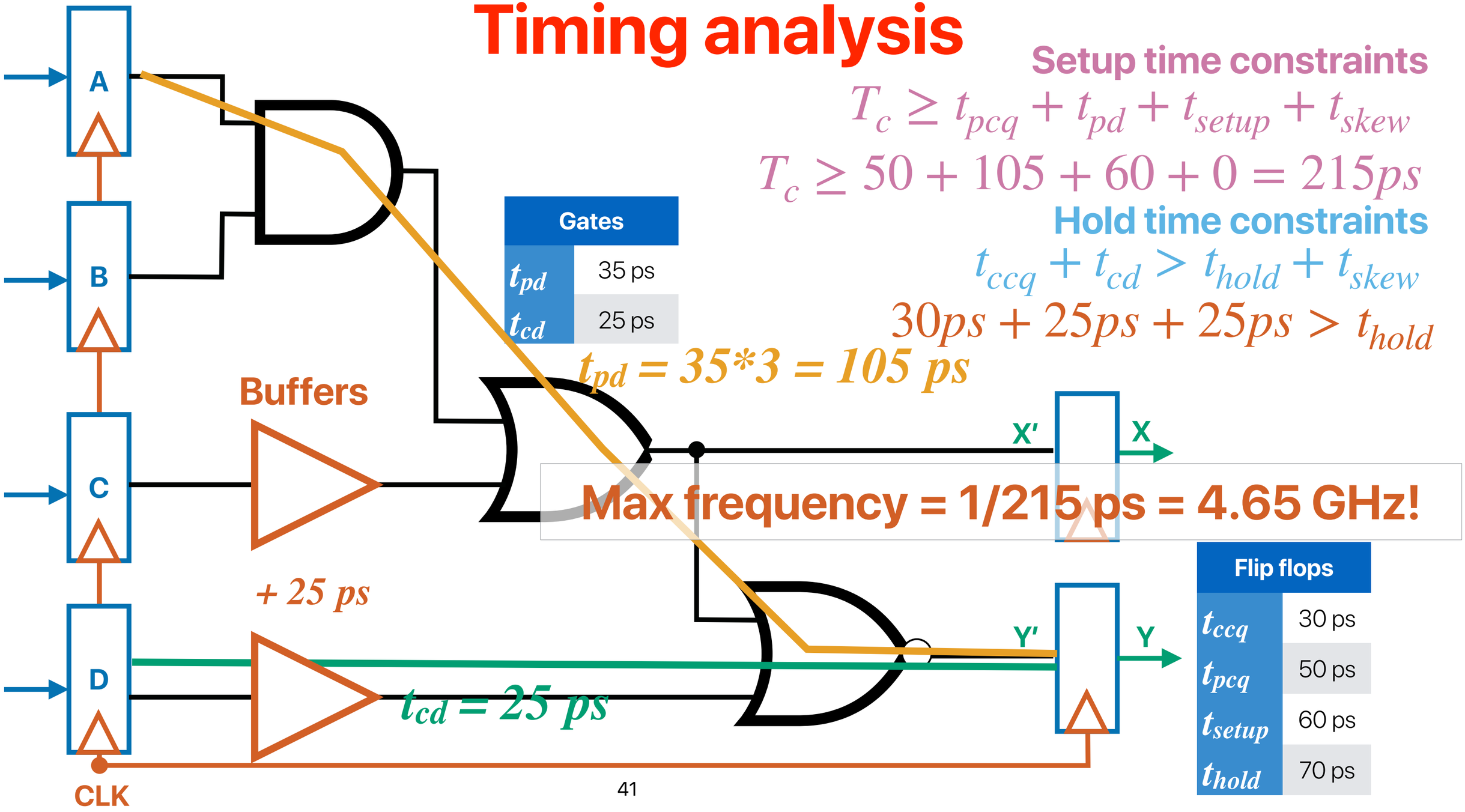
$$30ps + 25ps > t_{hold}$$

t_{hold} **= 70 ps!**

No!!!

Flip flops	
t_{ccq}	30 ps
t_{pcq}	50 ps
t_{setup}	60 ps
t_{hold}	70 ps

Timing analysis



Setup time constraints

$$T_c \geq t_{pcq} + t_{pd} + t_{setup} + t_{skew}$$

$$T_c \geq 50 + 105 + 60 + 0 = 215ps$$

Hold time constraints

$$t_{ccq} + t_{cd} > t_{hold} + t_{skew}$$

$$30ps + 25ps + 25ps > t_{hold}$$

Gates	
t_{pd}	35 ps
t_{cd}	25 ps

$t_{pd} = 35*3 = 105$ ps

$t_{cd} = 25$ ps

Max frequency = 1/215 ps = 4.65 GHz!

Flip flops	
t_{ccq}	30 ps
t_{pcq}	50 ps
t_{setup}	60 ps
t_{hold}	70 ps

Electrical
Computer Science
Engineering

120A

つづく