

Sequential Circuit

Professor: Yang Peng

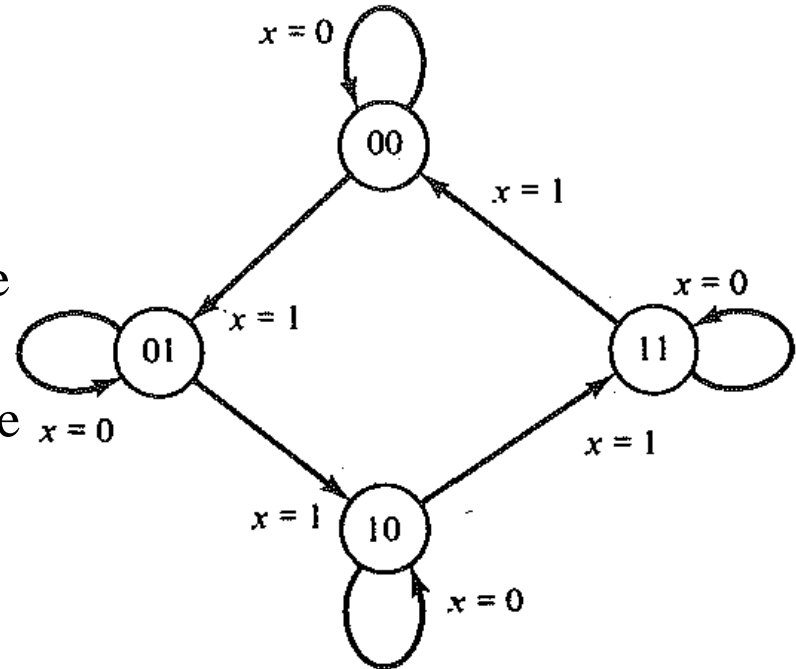
The slides are re-produced by the courtesy of
Dr. Arnie Berger and Dr. Wooyoung Kim

Topics

- Sequential Circuit
 - Chapter 4, 5 by Berger
 - Chapter 3 by Null
 - Flip-Flop
 - Excitation table and K-maps
 - Design registers

Example: 2-Binary Counter

- To design a 2-binary counter system with logic gates
- 2-binary counter system
 - Input: one control bit
 - Output: 2 bits that are stored in the system
 - If input is 0, then the 2 bits stay the same
 - If input is 1, then the 2 bits are counted up and if they reach the maximum, then they are reset to 0



Can we make this with combinational logic gates?

What is missing in the combinational circuit to design this system?

Combinational vs. Sequential

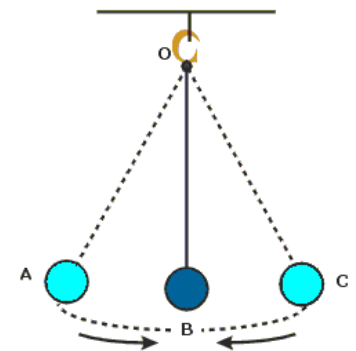
- **Combinational** circuit
 - The current output state **depends only on the input states**
 - It does not provide memory or state information, except ROM
- **Sequential** circuits
 - The current output **depends on the current input** and the **past history of inputs/outputs**
 - The most common type of sequential circuit is the **synchronous** type
 - **Edge-triggered Flip-Flop: synchronous (clocked)** sequential circuit
 - Interconnection of flip-flops and gates

Asynchronous vs. Synchronous

- **Asynchronous logic**
 - The change in the state depends on the input variables
- **Synchronous logic**
 - With millions of logic gates, we need to synchronize the change of logic state with some **master signal – clock**
 - Suppose we need to move data from memory to register to do arithmetic addition. How the computer will know the sequence of the work?
 - We need a system that works according to the order of tasks as time goes → synchronous sequential circuit design

Clocks and Time

- The **clock** in a digital system is an electronic analog of the pendulum which synchronizes the circuits
 - Clock signal is the master control signal
 - Circuit output changes on the rising or falling edges of a clock pulse
- Frequencies are the inverse of time (speed)
 - Hz = the number of cycles per second: how to compute the time per each cycle?
 - 1 kilohertz (KHz) = 10^3 Hz (cycle per second)
 - 1 megahertz (MHz) = 10^6 Hz
 - 1 gigahertz (GHz) = 10^9 Hz
 - 1 terahertz (THz) = 10^{12} Hz

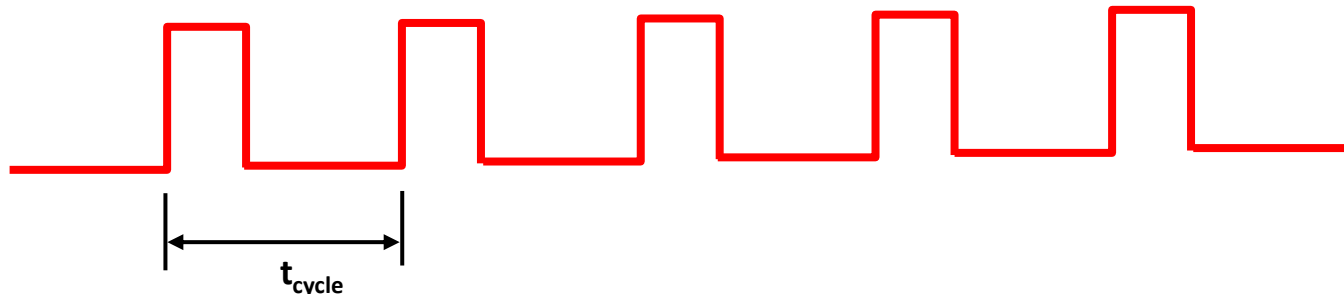


Clocks and Pulses

- Clocks are continuous streams of pulses

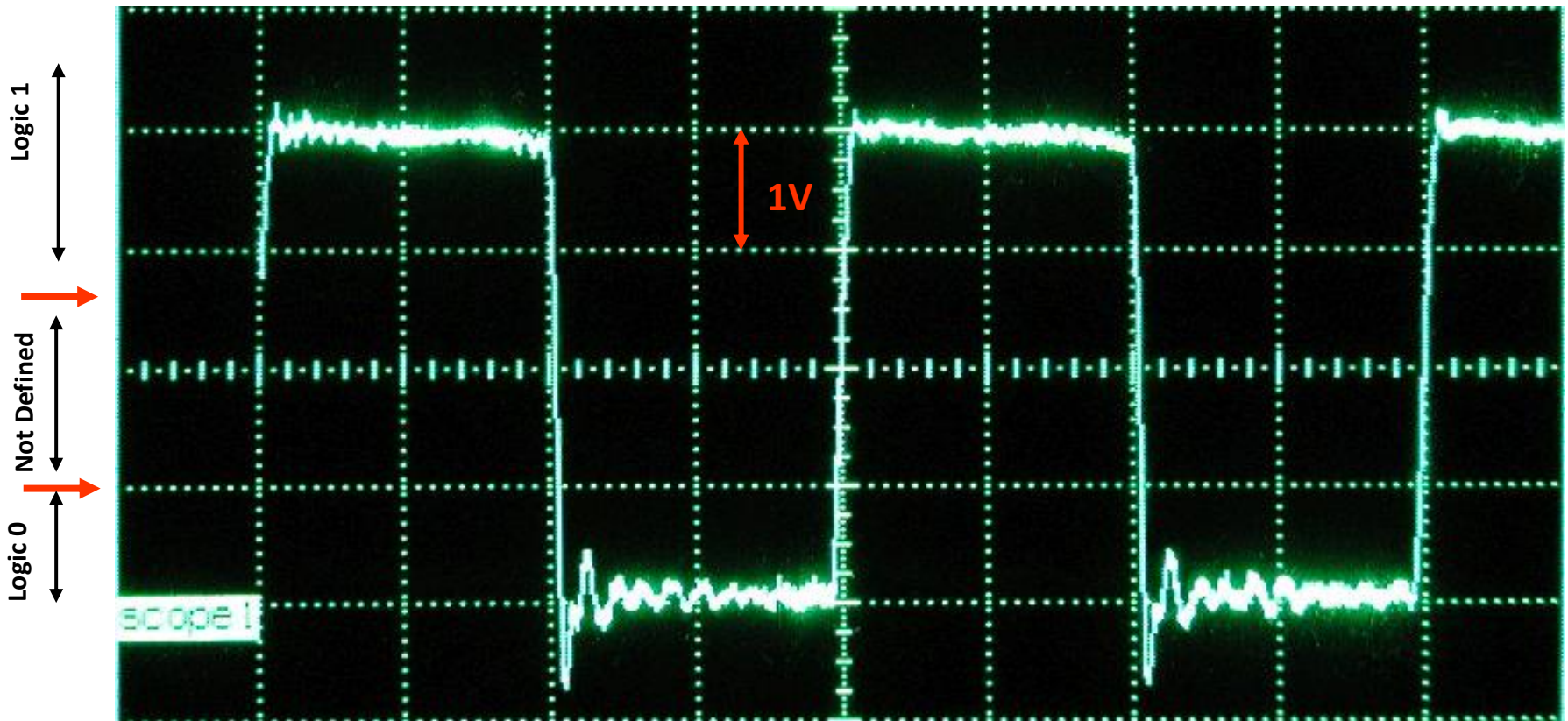


- Duty cycle = $t_{PH} / (t_{PH} + t_{PL}) \times 100\%$
 - The clock signal shown above has a 50% duty cycle
 - The clock signal shown below has a 25% duty cycle
 - **Period:** The time to complete one clock cycle
 - Period = t_{cycle}
 - **Frequency:** The inverse of the period, $f = 1/\text{period}$

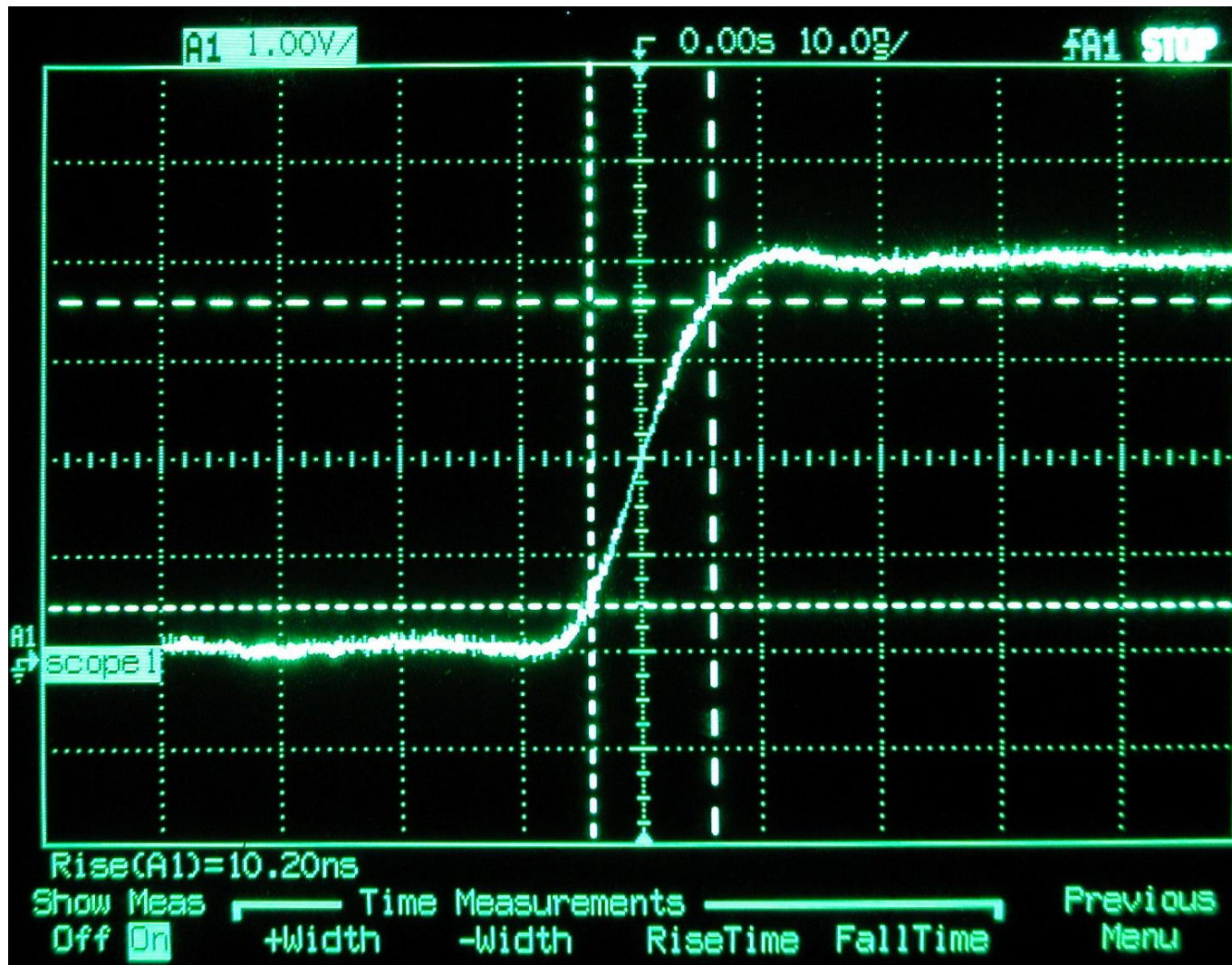


A Real Clock

- Note that signal fidelity is not an issue. As long as the logic 1 is greater than 2.5 volts and logic 0 is less than 1 volt, the pulse will be properly interpreted



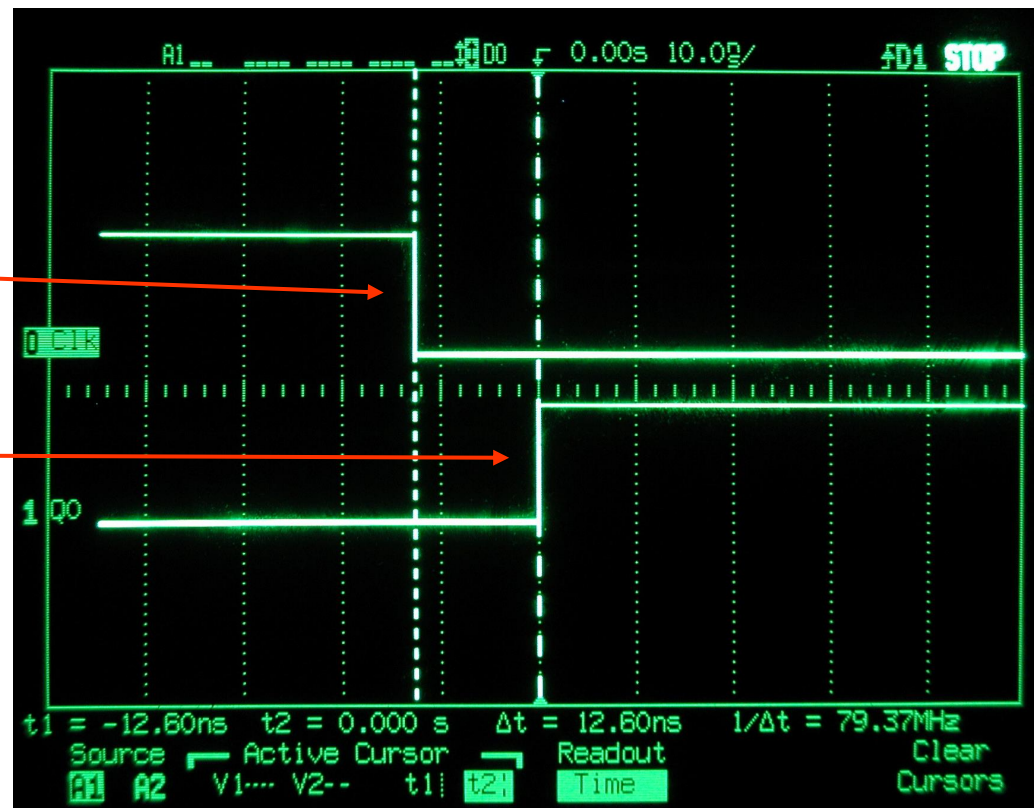
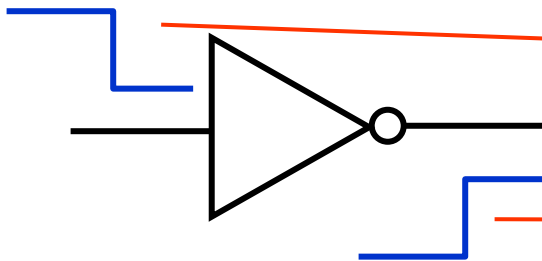
Pulse Rise Time Measurement



Propagation Delay

- Why can't we simply rev-up the clock on your PC to as high as you want?
 - Answer: Gates require **a finite amount of time to switch state** when their inputs change
- Consider the inverter gate below
 - The **propagation delay** (time delay from input change to output change)

$\Delta t = 12.60\text{ns}$

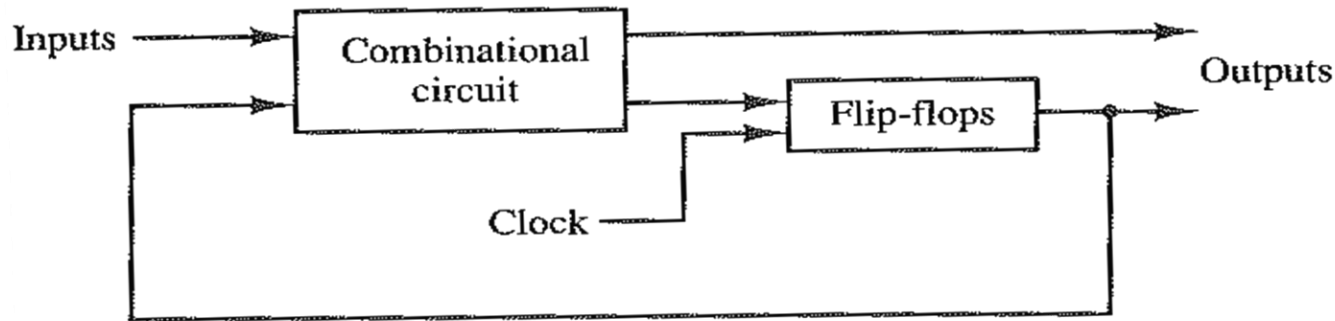


Basic Units for Sequential Circuit

- **Logic Gates:** the building blocks of **combinational circuit**
- **Flip-Flops:** the building blocks of **sequential circuit**
 - Flip-flops are actually built with logical gates
- Before we study flip-flops, let's start with a **latch**
- **Latch:** Use the feedback functionality
 - ***Feedback***: Bringing the output **back to an input**
 - ***Lock*** the gate pair into a new state
 - Transition is triggered by the application of the ***input pulse***

Sequential Circuit Design

- A sequential circuit can be a combination of combinational circuits and flip-flops

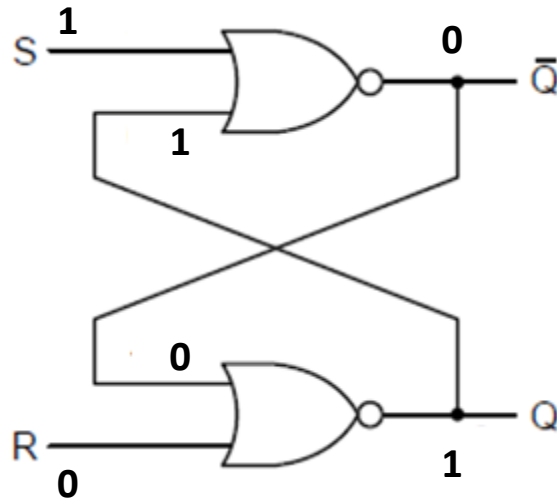


Block diagram of a sequential circuit

- Because the sequential circuit involves **state transition**, we have different types of tables for this circuit
 - **State Table**: Based on the **input and current state**, give the **next state** information (similar to truth table)
 - **Characteristic Table**: Give the state transition information based on inputs
 - **Excitation Table**: Inputs are the current and next states, and outputs are input signals

Latches

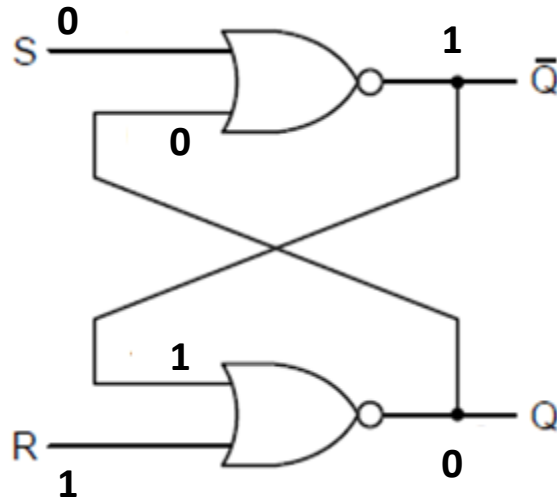
- A circuit that remembers previous input values – states
- **S-R Latch**: two **stable states** – **Set** and **Reset**
- Regardless of previous states,
 - When **S = 1** and **R = 0**, **Q becomes 1**



A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

Latches

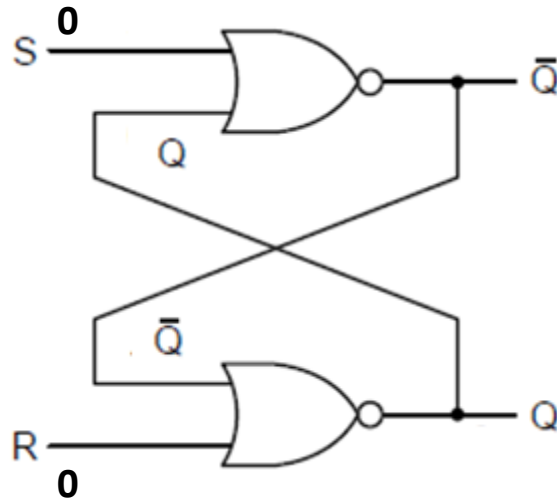
- A circuit that remembers previous input values – states
- **S-R Latch**: two **stable states** – **Set** and **Reset**
- Regardless of previous states,
 - When **S = 0** and **R = 1**, **Q becomes 0**



A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

Latches

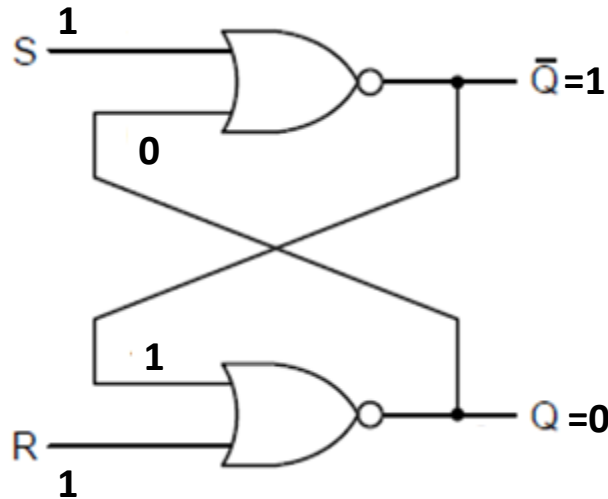
- A circuit that remembers previous input values – states
- **S-R Latch**: two **stable states** – **Set** and **Reset**
- Regardless of previous states,
 - When **S = 0** and **R = 0**, the output will be “locked”



A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

Latches

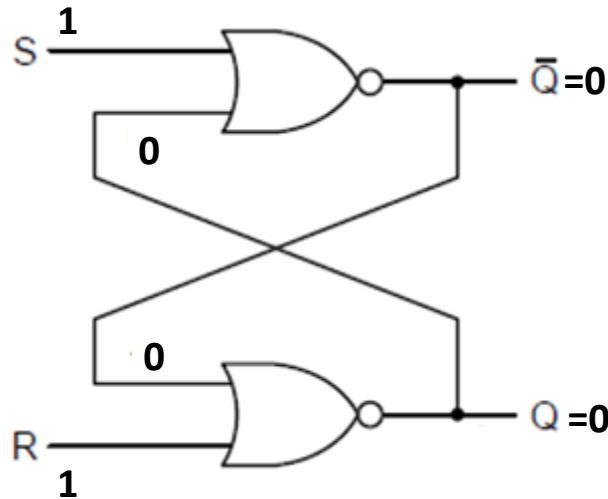
- A circuit that remembers previous input values – states
- **S-R Latch**: two **stable states** – **Set** and **Reset**
- Regardless of previous states,
 - What if $S=1$ and $R=1$, simultaneously?



A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

Latches

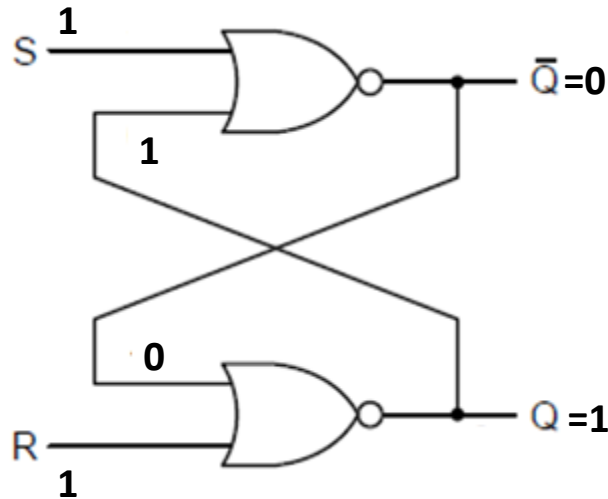
- A circuit that remembers previous input values – states
- **S-R Latch**: two **stable states** – **Set** and **Reset**
- Regardless of previous states,
 - What if $S=1$ and $R=1$, simultaneously?



A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

Latches

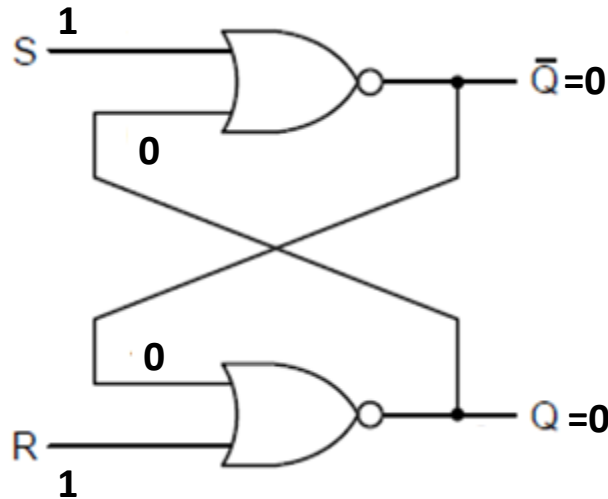
- A circuit that remembers previous input values – states
- **S-R Latch**: two **stable states** – **Set** and **Reset**
- Regardless of previous states,
 - What if $S=1$ and $R=1$, simultaneously?



A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

Latches

- A circuit that remembers previous input values – states
- **S-R Latch**: two **stable states** – **Set** and **Reset**
- Regardless of previous states,
 - What if $S=1$ and $R=1$, simultaneously?



A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

Latches

- A circuit that remembers previous input values – states
- **S-R Latch**: two **stable states** – **Set** and **Reset**
- Regardless of previous states,
 - When S is 1 and R is 0, Q becomes 1
 - When S is 0 and R is 1, Q becomes 0
 - When S=R=0, the output will be “**locked**”
 - When S=R=1, the output will be in “race condition” (forbidden state)

SR Latch

(a) State Table

Current Inputs SR	Current State Q_n	Next State Q_{n+1}
00	0	0
00	1	1
01	0	0
01	1	0
10	0	1
10	1	1
11	0	—
11	1	—

(b) Characteristic Table

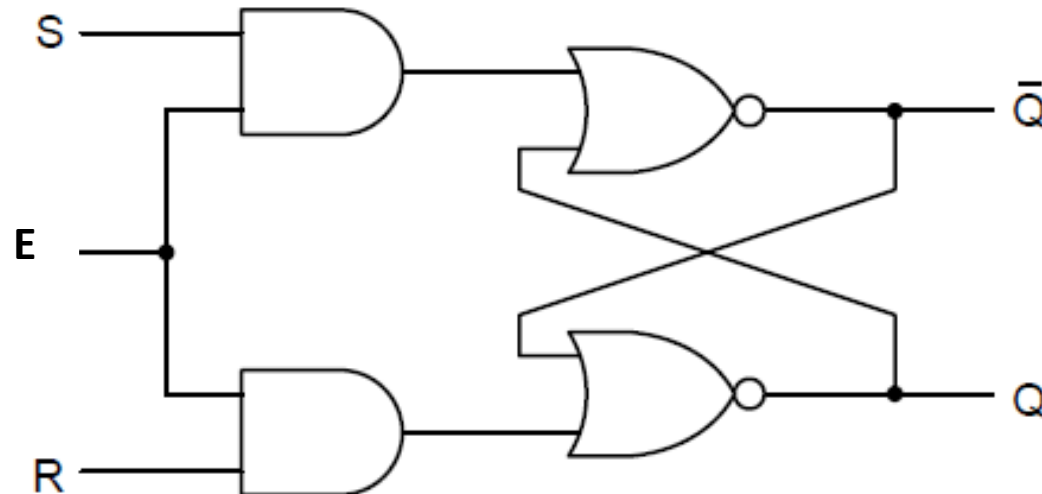
S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	—

(c) Response to Series of Inputs

t	0	1	2	3	4	5	6	7	8	9
S	1	0	0	0	0	0	0	0	1	0
R	0	0	0	1	0	0	1	0	0	0
Q_{n+1}	1	1	1	0	0	0	0	0	1	1

Gated SR Latches

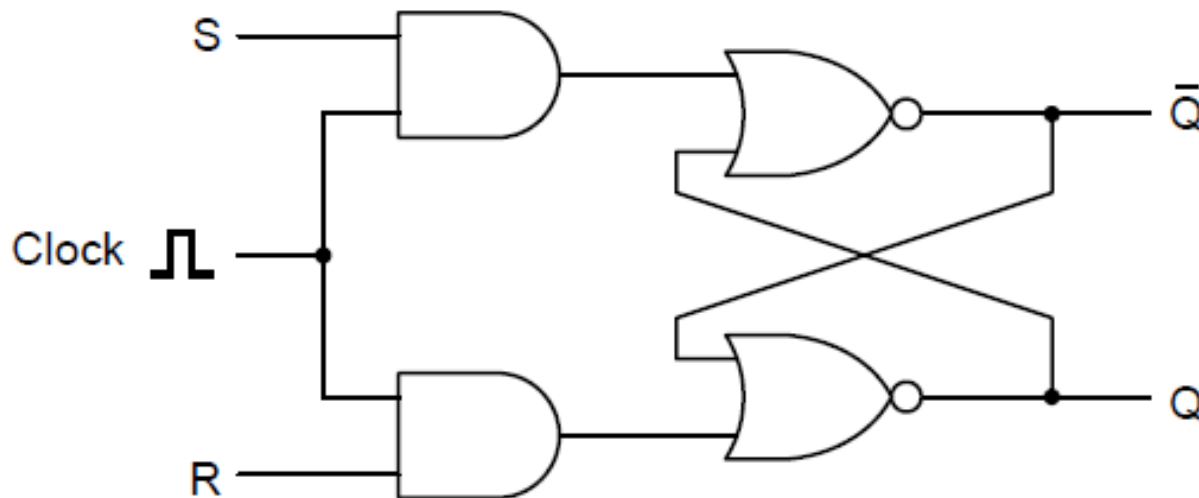
- How to resolve the forbidden condition when $S=R=1$?
- Does this solve the forbidden condition problem?
 - When E is 0 (at logic low), the problem is solved



**E is the enable
line**

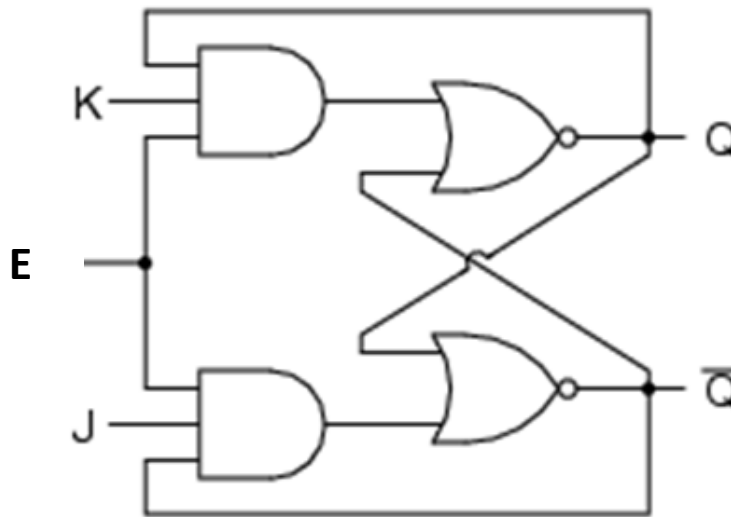
Clocked SR Latches

- SR Latch is an asynchronous operation, yet
- Preventing the latch from changing state, except at
“certain specific time”
 - Only when clock is up, the latch is sensitive to S or R
 - How to resolve the forbidden condition when $S=R=1$?
 - When the clock is down, the problem is solved



Gated JK Latch

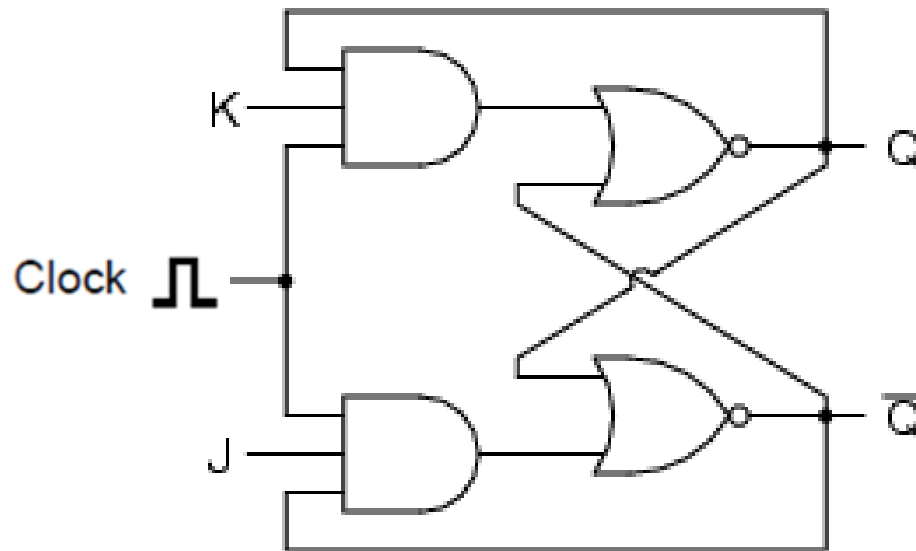
- Avoid the SR latch's instability by preventing the inputs being 1 at the same time
 - “Toggle” does not cause the race condition, which is just OK but not perfect.
- In this case, all possible combinations of input values are valid



E	J	K	Q	\overline{Q}
1	0	0	latch	latch
1	0	1	0	1
1	1	0	1	0
1	1	1	toggle	toggle
0	0	0	latch	latch
0	0	1	latch	latch
0	1	0	latch	latch
0	1	1	latch	latch

Clocked JK Latch

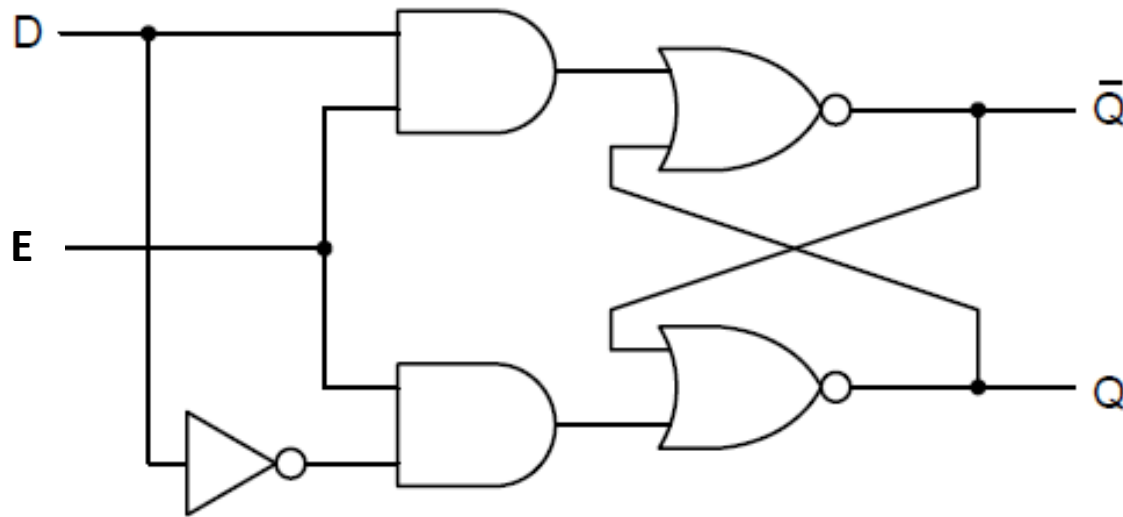
- Avoid the SR latch's instability by preventing the inputs being 1 at the same time
- In this case, all possible combinations of input values are valid



C	J	K	Q	\bar{Q}
0	0	0	latch	latch
0	0	1	0	1
0	1	0	1	0
0	1	1	toggle	toggle
x	0	0	latch	latch
x	0	1	latch	latch
x	1	0	latch	latch
x	1	1	latch	latch

Gated D Latches

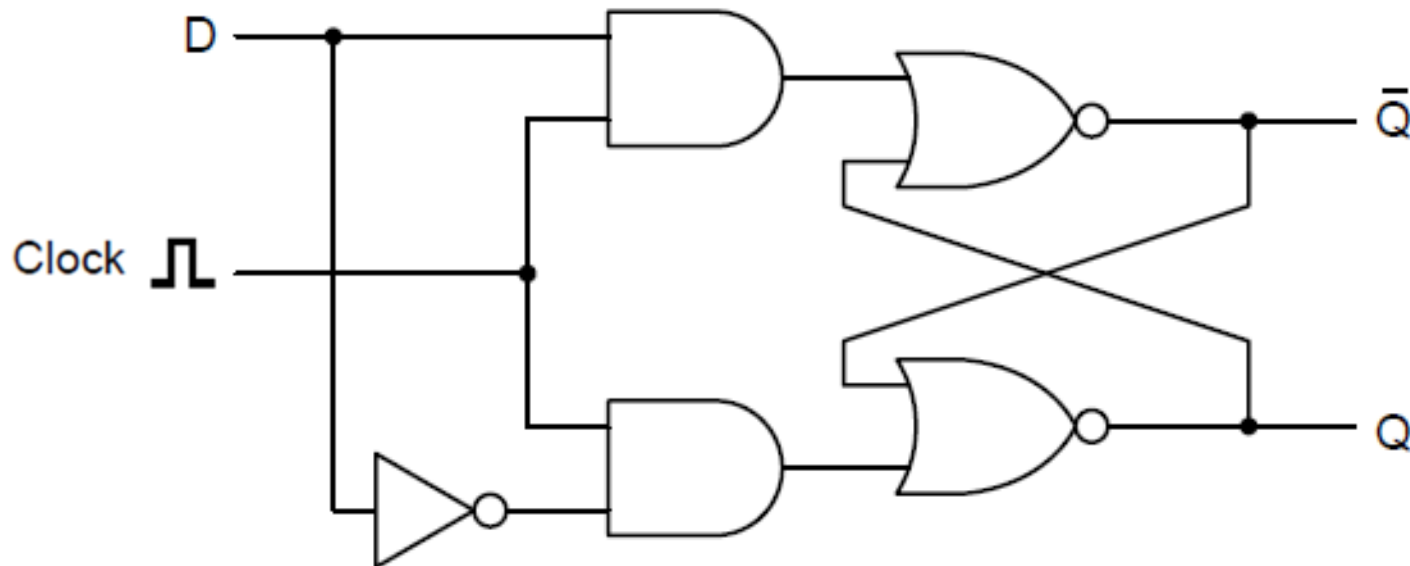
- Avoid the SR latch's instability by giving R as the inverse of S
- D: input – the current data
- Q: output – the stored value



E	D	Q	\bar{Q}
1	0	0	1
1	1	1	0
0	0	latch	latch
0	1	latch	latch

Clocked D Latches

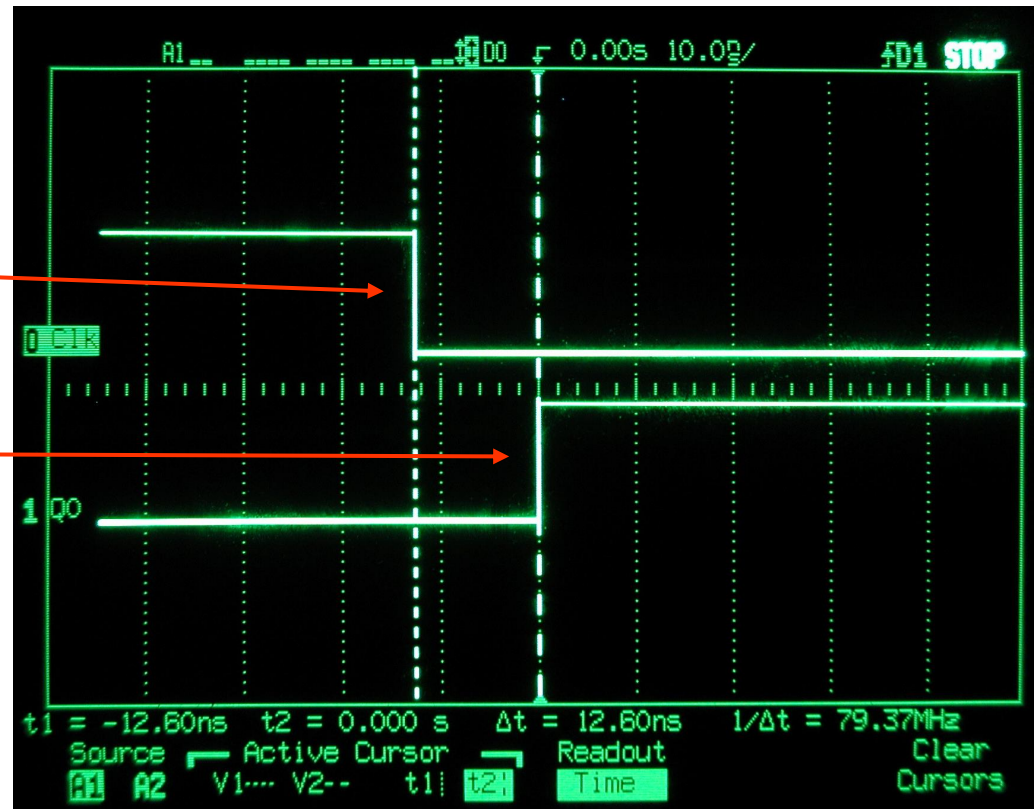
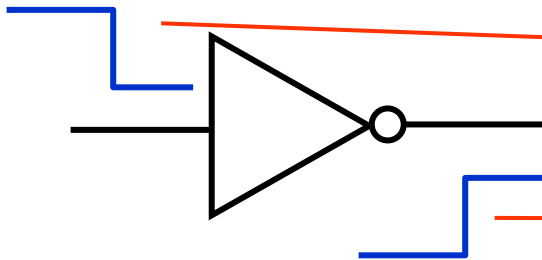
- Avoid the SR latch's instability by giving R as the inverse of S
- D: input – the current data
- Q: output – the stored value
- To **load** the current value of D, just give a **positive pulse** on the clock line
- *What if D changes while the clock stays in HIGH (active)?*



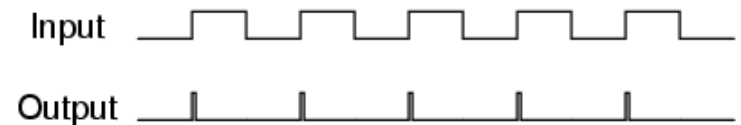
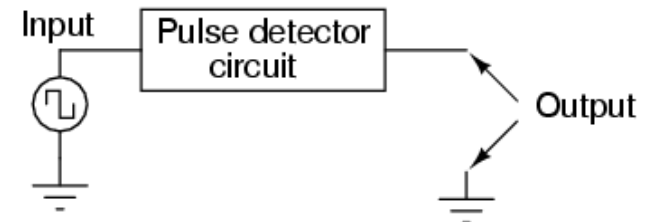
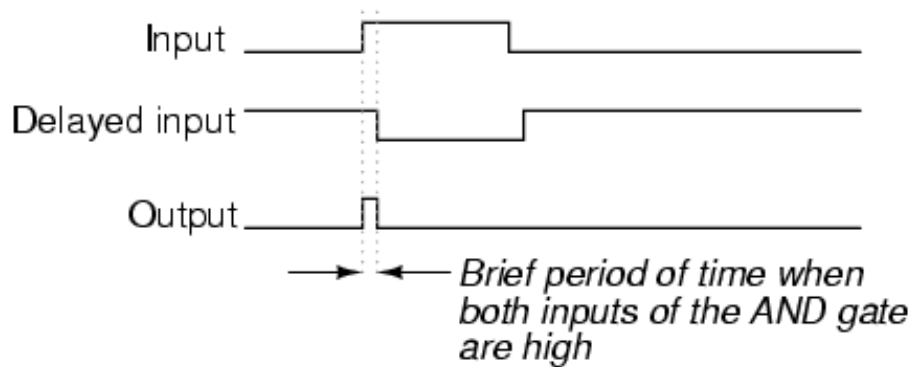
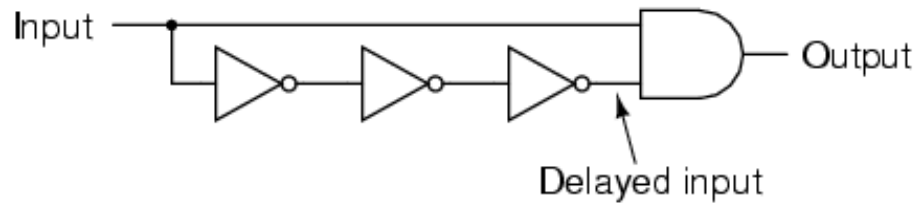
Propagation Delay

- The **propagation delay** (time delay from input change to output change)

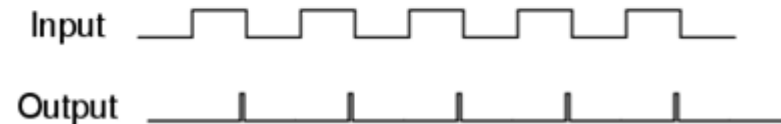
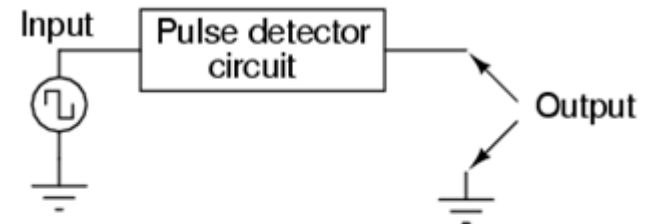
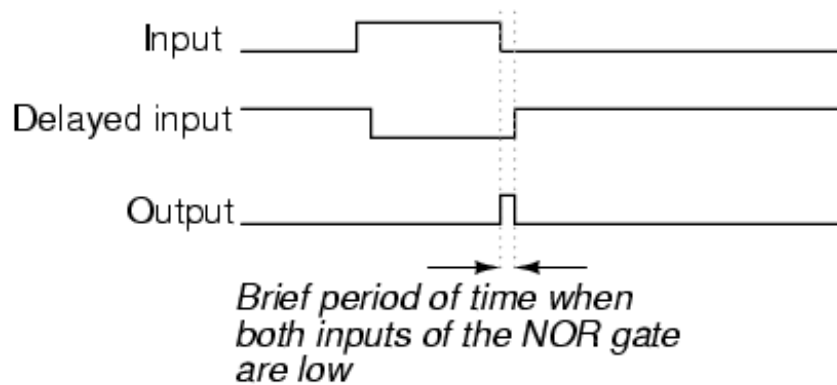
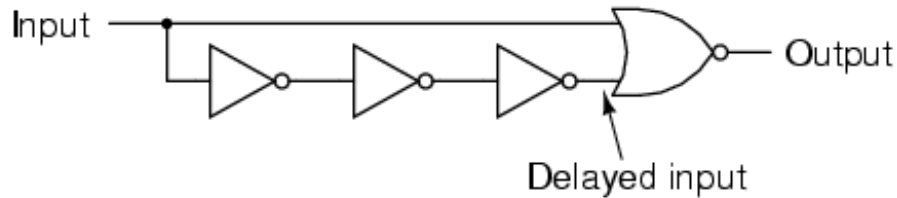
$$\Delta t = 12.60\text{ns}$$



Rising-Edge Pulse Detector

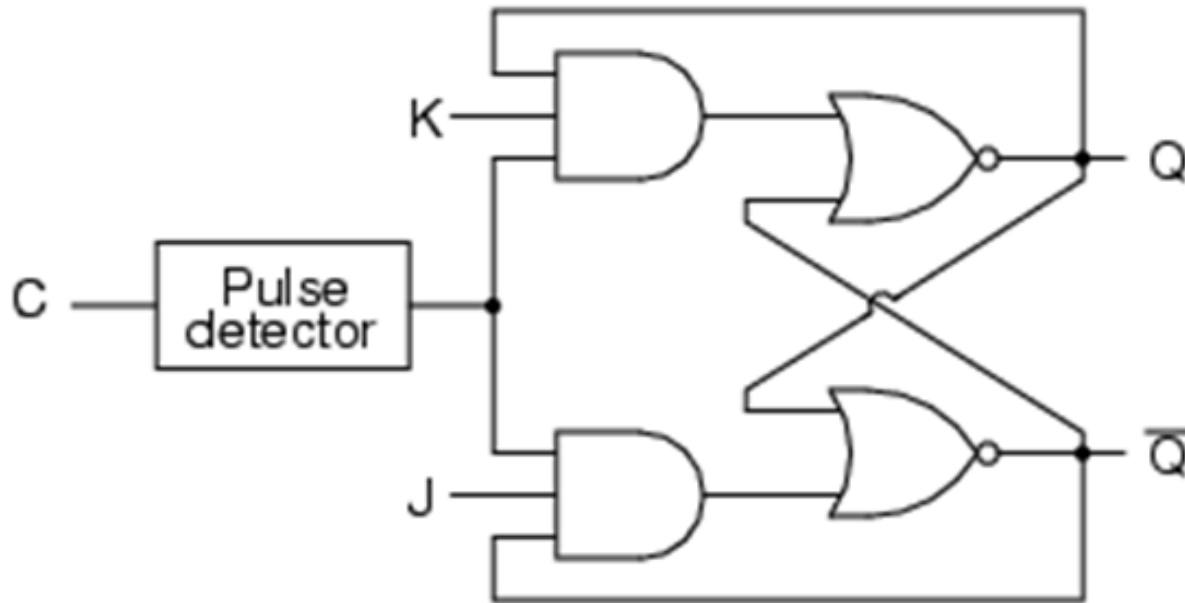


Falling-Edge Pulse Detector



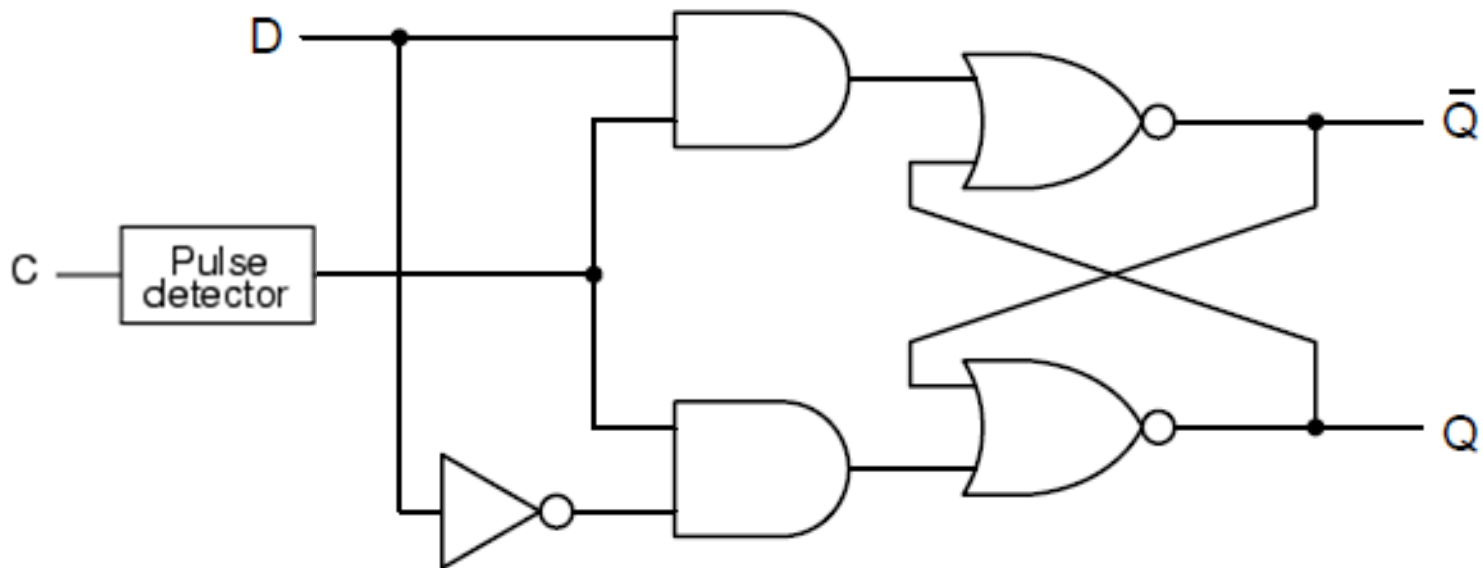
J-K Flip-Flop

- Only allow the output to change at a clock pulse



D Flip-Flop

- Only allow the output to change at a clock pulse



Latches vs. Flip-Flops

- Latch lacks a mechanism to shift control to the **clock edge**
- The state changes when the clock is active

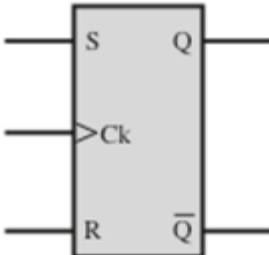
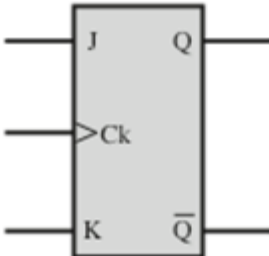
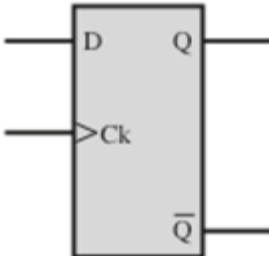
Level-Triggered

- **Flip-Flop**: State transition occurs **when the clock transitions from 0 to 1 (rising) or from 1 to 0 (falling)**

Edge Triggered

- It's called a Flip-flop because **output Q is flipped back and forth**
- Sometimes Flip-Flops and latches are used as the same
- **But in our class, we make the difference clear**
- **Without further specification, we use Rising Edge as a trigger in this class**

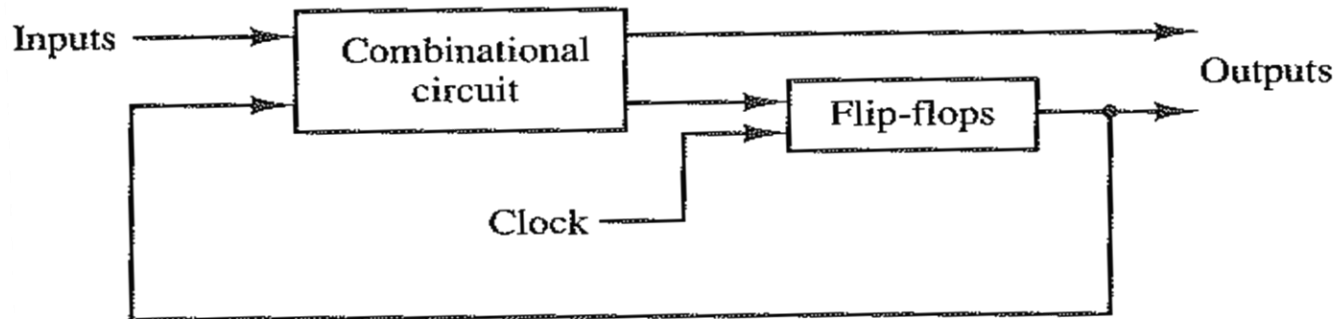
Characteristic Table for FF

Name	Graphical Symbol	Characteristic Table															
S-R		<table><tr><th>S</th><th>R</th><th>Q_{n+1}</th></tr><tr><td>0</td><td>0</td><td>Q_n</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>–</td></tr></table>	S	R	Q_{n+1}	0	0	Q_n	0	1	0	1	0	1	1	1	–
S	R	Q_{n+1}															
0	0	Q_n															
0	1	0															
1	0	1															
1	1	–															
J-K		<table><tr><th>J</th><th>K</th><th>Q_{n+1}</th></tr><tr><td>0</td><td>0</td><td>Q_n</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>$\overline{Q_n}$</td></tr></table>	J	K	Q_{n+1}	0	0	Q_n	0	1	0	1	0	1	1	1	$\overline{Q_n}$
J	K	Q_{n+1}															
0	0	Q_n															
0	1	0															
1	0	1															
1	1	$\overline{Q_n}$															
D		<table><tr><th>D</th><th>Q_{n+1}</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	D	Q_{n+1}	0	0	1	1									
D	Q_{n+1}																
0	0																
1	1																

Basic Flip-Flops and the characteristic table

Sequential Circuit Design

- A sequential circuit can be a combination of combinational circuits and flip-flops



Block diagram of sequential circuit

- Because the sequential circuit involves state transition, we have different types of tables for this circuit
 - **State Table:** Based on the input and current state, give the next state information (like truth table)
 - **Characteristic Table:** Give the state transition information based on inputs
 - **Excitation Table:** Inputs are the current and next states, and outputs are input signals (S, R, J, K) → why do we need this table?

Excitation Table for Flip-Flops

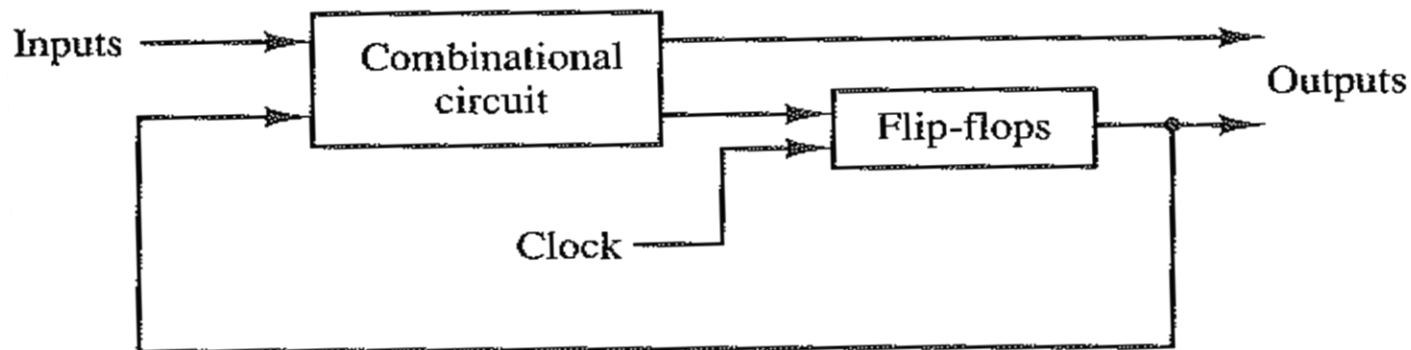
SR flip-flop				D flip-flop		
$Q(t)$	$Q(t + 1)$	S	R	$Q(t)$	$Q(t + 1)$	D
0	0	0	\times	0	0	0
0	1	1	0	0	1	1
1	0	0	1	1	0	0
1	1	\times	0	1	1	1

JK flip-flop				T flip-flop		
$Q(t)$	$Q(t + 1)$	J	K	$Q(t)$	$Q(t + 1)$	T
0	0	0	\times	0	0	0
0	1	1	\times	0	1	1
1	0	\times	1	1	0	1
1	1	\times	0	1	1	0

“X” is the “don’t care” state

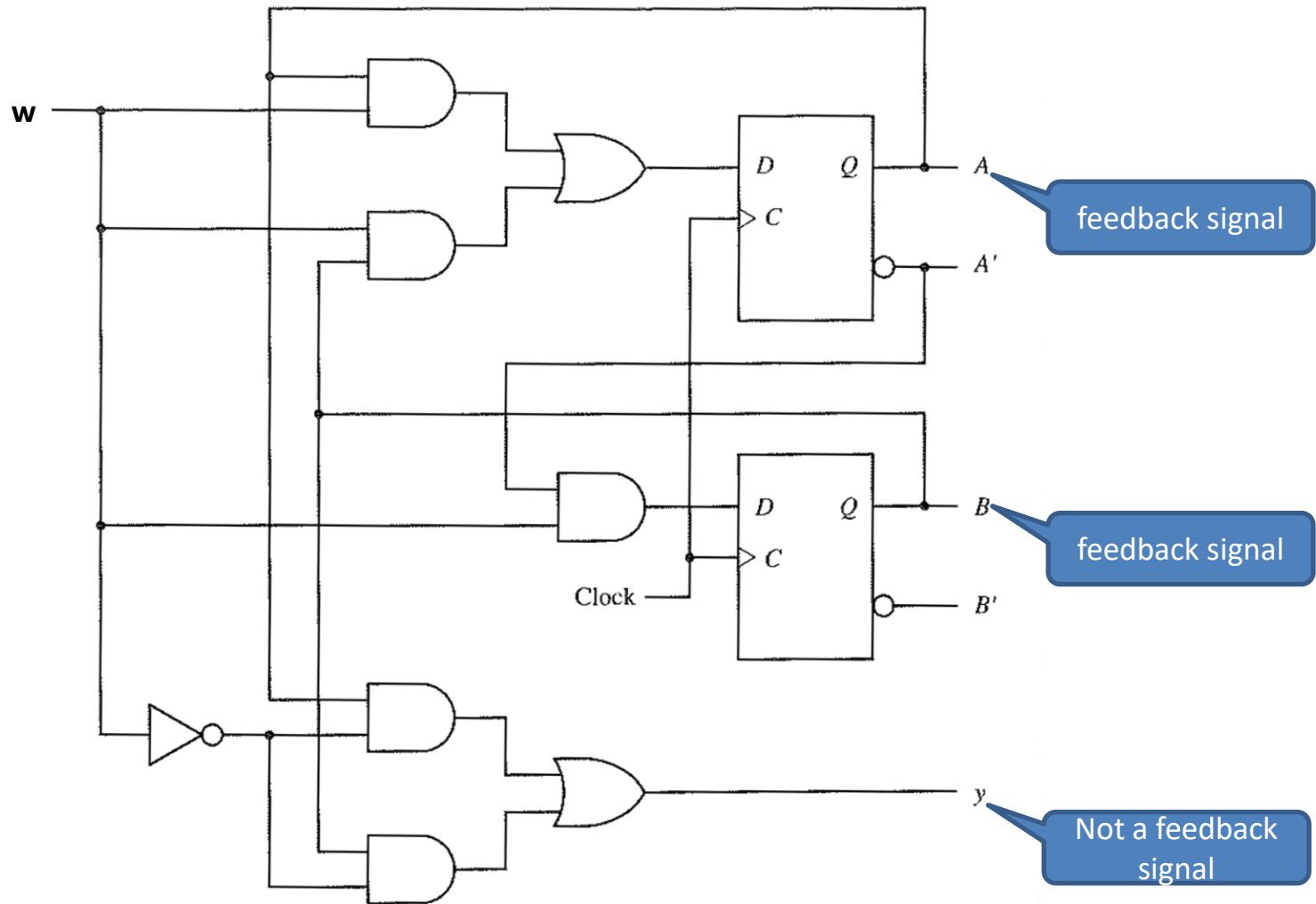
Excitation Table

- Excitation table (*inverted* characteristic table)
 - Input: current state and next state
 - Output: **S R** or **J K** or **D** – **the control bit(s) of FF**
- Necessity
 - FF is a feedback system
 - The transition information goes back to the states for inputs (S, R, etc.)
 - **If you know the excitation table, you know how the sequential circuit can be built**



Block diagram of a generic sequential circuit

Example of a Sequential Circuit



What does this circuit do?

Draw a state table to analyze a sequential circuit

How to draw a state table?

State Table

From the sequential circuit, can we build a state table?

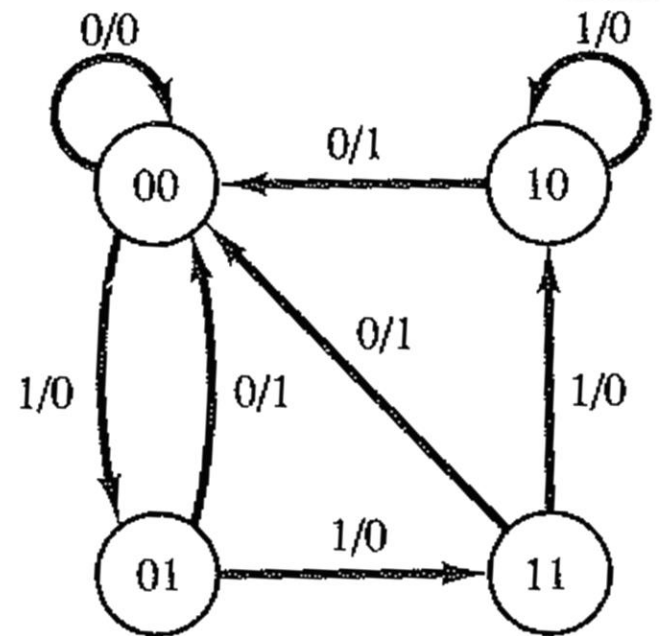
A(t)	B(t)	(input) w	(output) Y	A(t+1)	B(t+1)
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	1	0	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	0	1	0

How to draw a state machine
from a state table?

State Machine/Diagram

From the state table, can we build a state machine?

A(t)	B(t)	(input) w	(output) Y	A(t+1)	B(t+1)
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	1	0	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	0	1	0



How to design a sequential circuit?

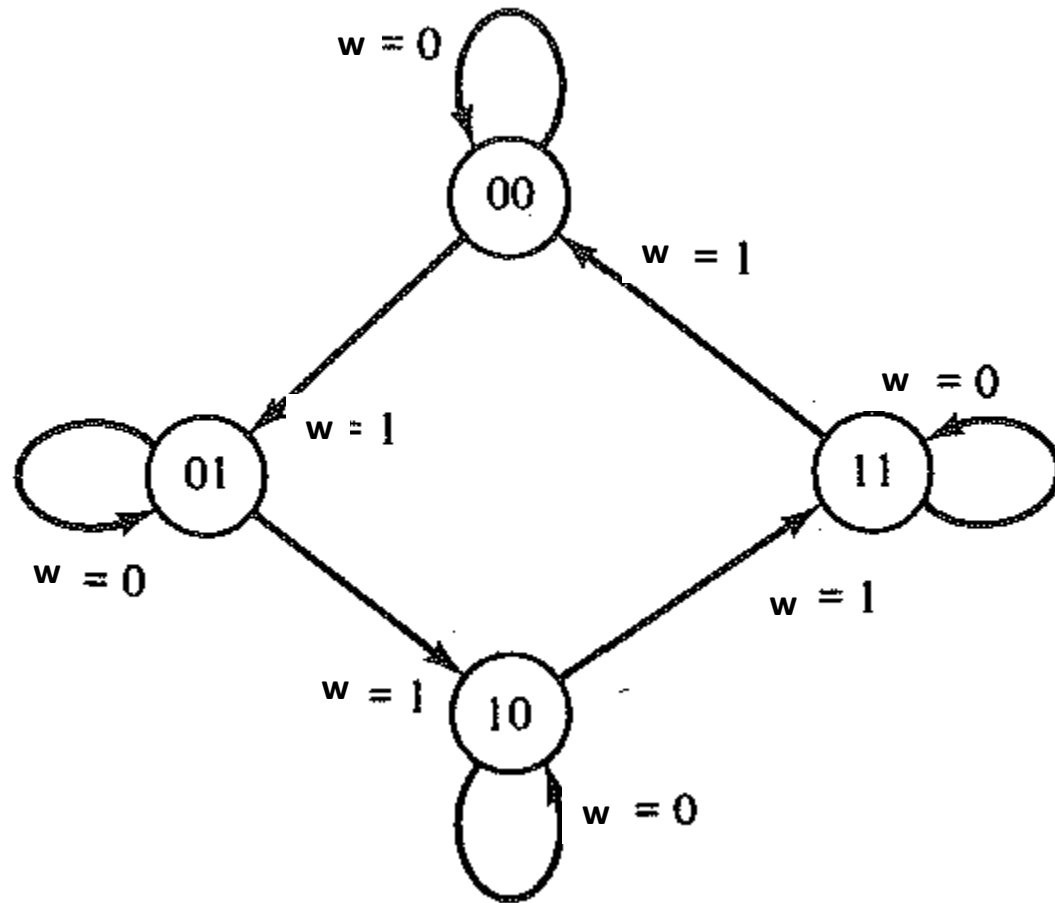
Design a Sequential Circuit

- Design a sequential circuit
 1. Draw a *State Machine* (state diagram)
 2. Figure out the *Inputs* and *Outputs*
 3. Build a *State Table* and derive an *Excitation Table*
 4. Derive a *Boolean Equation* using *K-map*
 5. Build the **sequential circuit**

Let's design a 2-bit binary counter

- A sequence of repeated binary states 00, 01, 10, 11 whenever the input is 1.

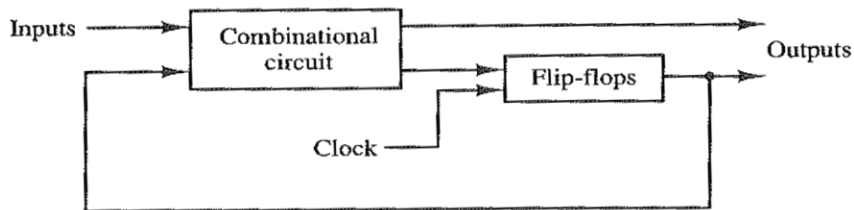
2-bit Binary Counter – State Machine



2-bit Binary Counter – Input/Output

- Input (control signal): w
- Output (# of FF): 2 (2 bits)
- Suppose we are using JK FFs
- Then we need two J's and two K's
- In the state machine, you can build a state table
- The next states are connected to JK's
- We need an extended table \rightarrow excitation table

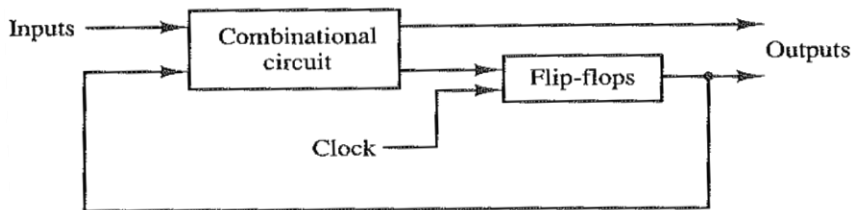
Present state (t)		Input	Next state (t+1)	
A	B		A	B
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		



2-bit Binary Counter – State Table

- Input (control signal): w
- Output (# of FF): 2 (2 bits)
- Suppose we are using JK FFs
- Then we need two J's and two K's
- In the state machine, you can build a state table
- The next states are connected to JK's
- We need an extended table \rightarrow excitation table

Present state (t)		Input	Next state (t+1)	
A	B		A	B
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	1	0
1	0	1	1	1
1	1	0	1	1
1	1	1	0	0



2-bit Binary Counter – Excitation Table 1

Q(t)	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Reference:

JK FF's excitation Table

Present state (t)		Input	Next state (t+1)		Flip-flop inputs			
A	B	w	A	B	J _A	K _A	J _B	K _B
0	0	0	0	0				
0	0	1	0	1				
0	1	0	0	1				
0	1	1	1	0				
1	0	0	1	0				
1	0	1	1	1				
1	1	0	1	1				
1	1	1	0	0				

Extended table based on the state table

2-bit Binary Counter – Excitation Table 2

Q(t)	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Reference:

JK FF's excitation Table

Present state (t)		Input	Next state (t+1)		Flip-flop inputs			
A	B	w	A	B	J _A	K _A	J _B	K _B
0	0	0	0	0	0	x	0	x
0	0	1	0	1	0	x	1	x
0	1	0	0	1	0	x	x	0
0	1	1	1	0	1	x	x	1
1	0	0	1	0	x	0	0	x
1	0	1	1	1	x	0	1	x
1	1	0	1	1	x	0	x	0
1	1	1	0	0	x	1	x	1

Excitation table with J and K inputs

2-bit Binary Counter – Excitation Table 3

Q(t)	Q(t+1)	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Reference:

JK FF's excitation Table

You cannot control the next state: So, not an input nor an output

Present state (t)		Input	Next state (t+1)		Flip-flop inputs			
A	B	w	A	B	J _A	K _A	J _B	K _B
0	0	0	0	0	0	x	0	x
0	0	1	0	1	0	x	1	x
0	1	0	0	1	0	x	x	0
0	1	1	1	0	1	x	x	1
1	0	0	1	0	x	0	0	x
1	0	1	1	1	x	0	1	x
1	1	0	1	1	x	0	x	0
1	1	1	0	0	x	1	x	1

Excitation table with the "Next state" been removed

2-bit Binary Counter – Boolean Equation

	$\sim B \sim w$	$\sim B w$	$B w$	$B \sim w$
$\sim A$			1	
A	X	X	X	X

$$J_A = Bw$$

	$\sim B \sim w$	$\sim B w$	$B w$	$B \sim w$
$\sim A$	X	X	X	X
A			1	

$$K_A = Bw$$

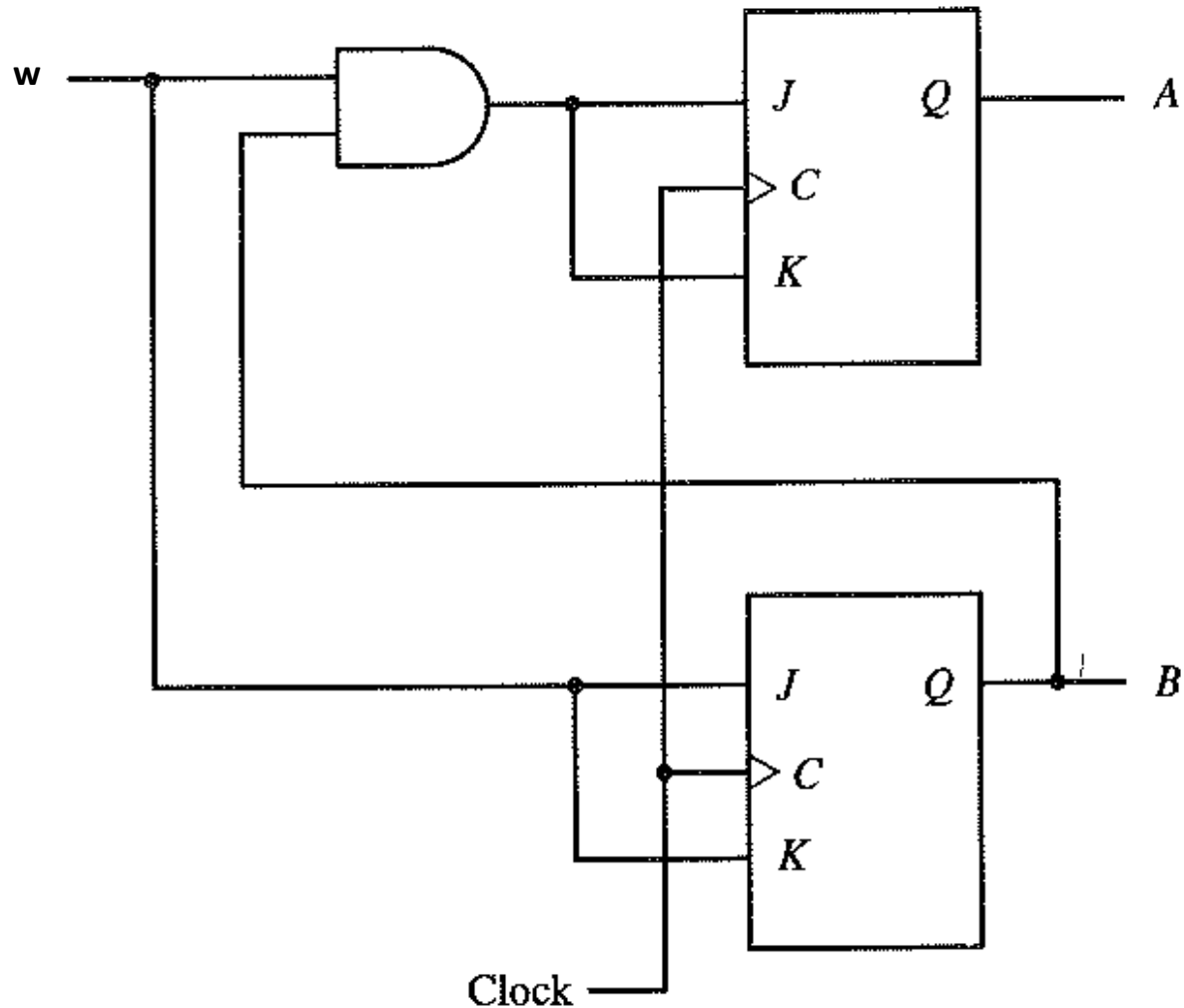
	$\sim B \sim w$	$\sim B w$	$B w$	$B \sim w$
$\sim A$		1	X	X
A		1	X	X

$$J_B = w$$

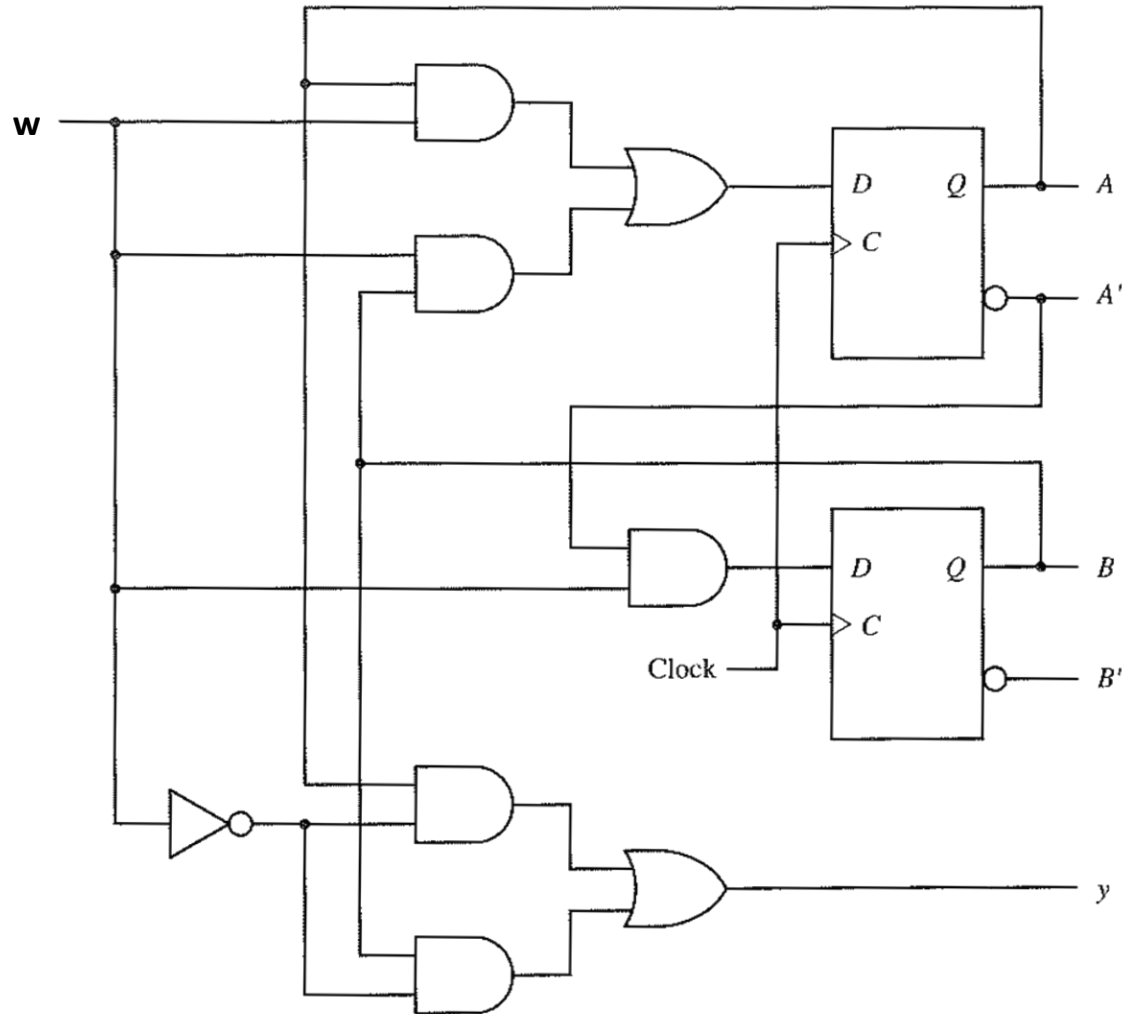
	$\sim B \sim w$	$\sim B w$	$B w$	$B \sim w$
$\sim A$	X	X	1	
A	X	X	1	

$$K_B = w$$

2-bit Binary Counter – Sequential Circuit



Example of a Sequential Circuit



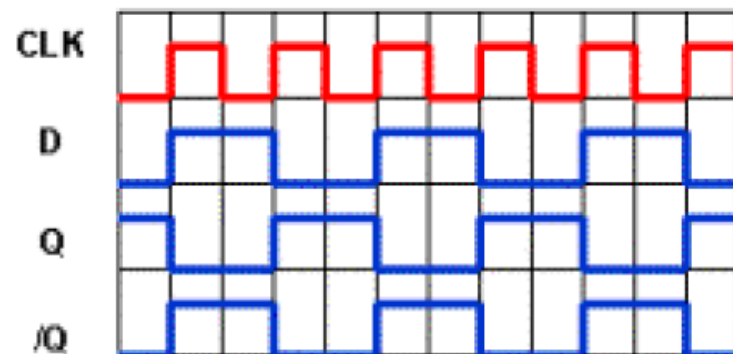
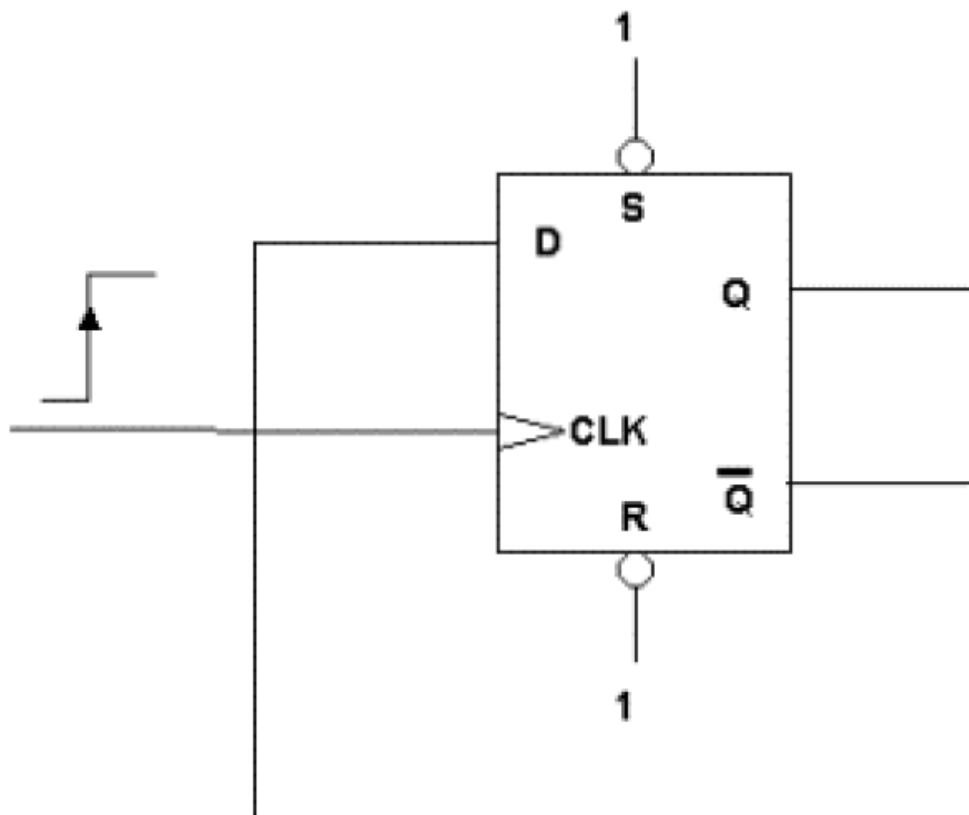
What does this circuit do?

Draw a state table to analyze a sequential circuit

A(t)	B(t)	(input) w	(output) Y	A(t+1)	B(t+1)
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	1	0	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	0	1	0
1	1	0	1	0	0
1	1	1	0	1	0

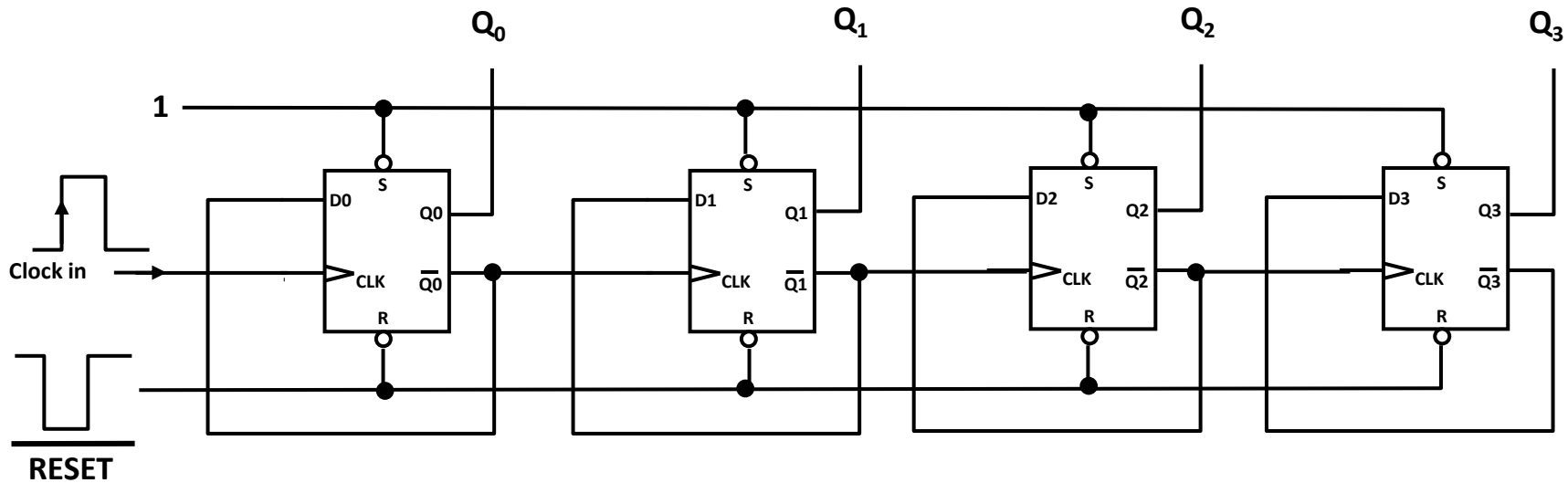
Examples of complicated sequential circuits

D flip-flop connected in a divide-by-two configuration



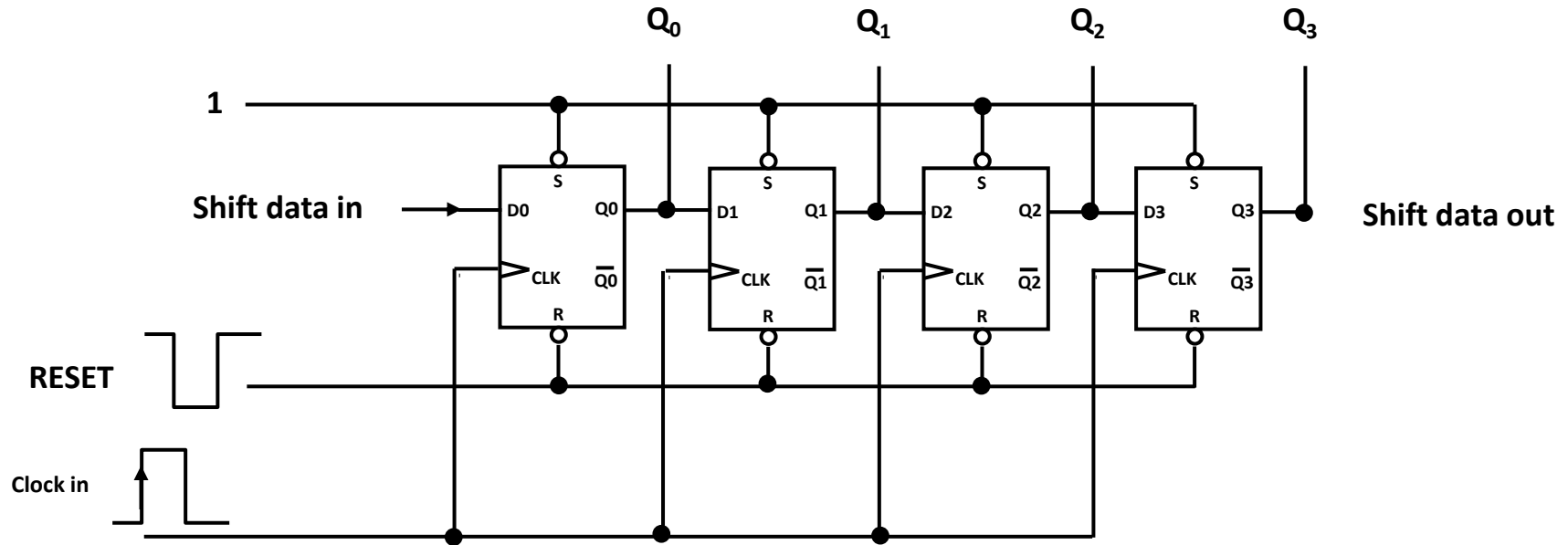
The “D” FF as a Counting Element

- A 4-bit binary *ripple counter*
 - the pulses “ripple” through the circuit



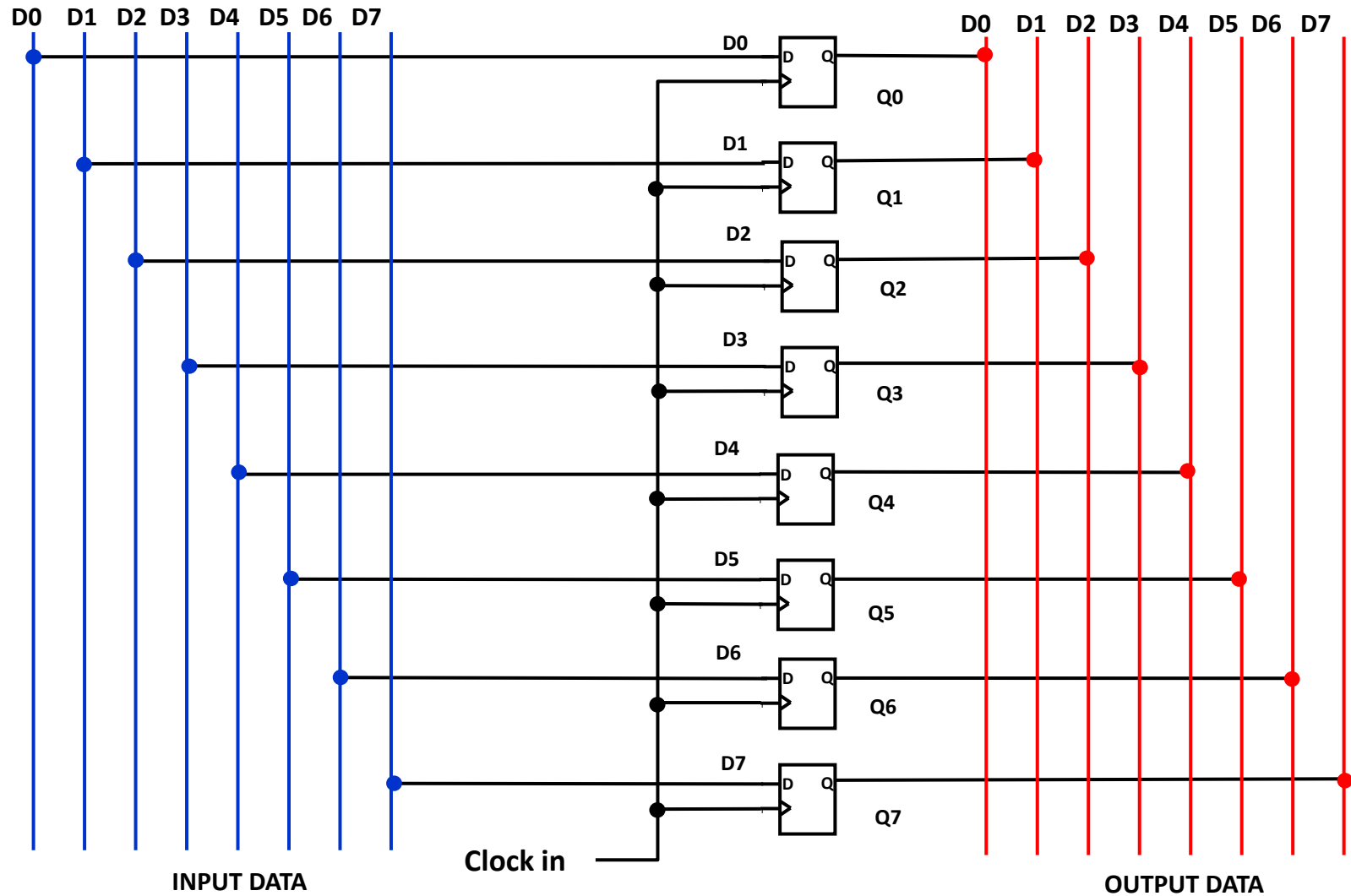
- Each “D” FF **divides the incoming clock frequency by 2**
- **RESET** sets **all Q output to 0** without a clock signal (asynchronous)
- Counts as fast as the first stage can toggle, but cannot be read until the count has rippled through to the last stage
- Can build counter/dividers of any length, any binary divisor
 - **Clock frequency at output Q3 equals $f_{\text{Clock in}} \div 16$**

“D” FF as a Shift Register

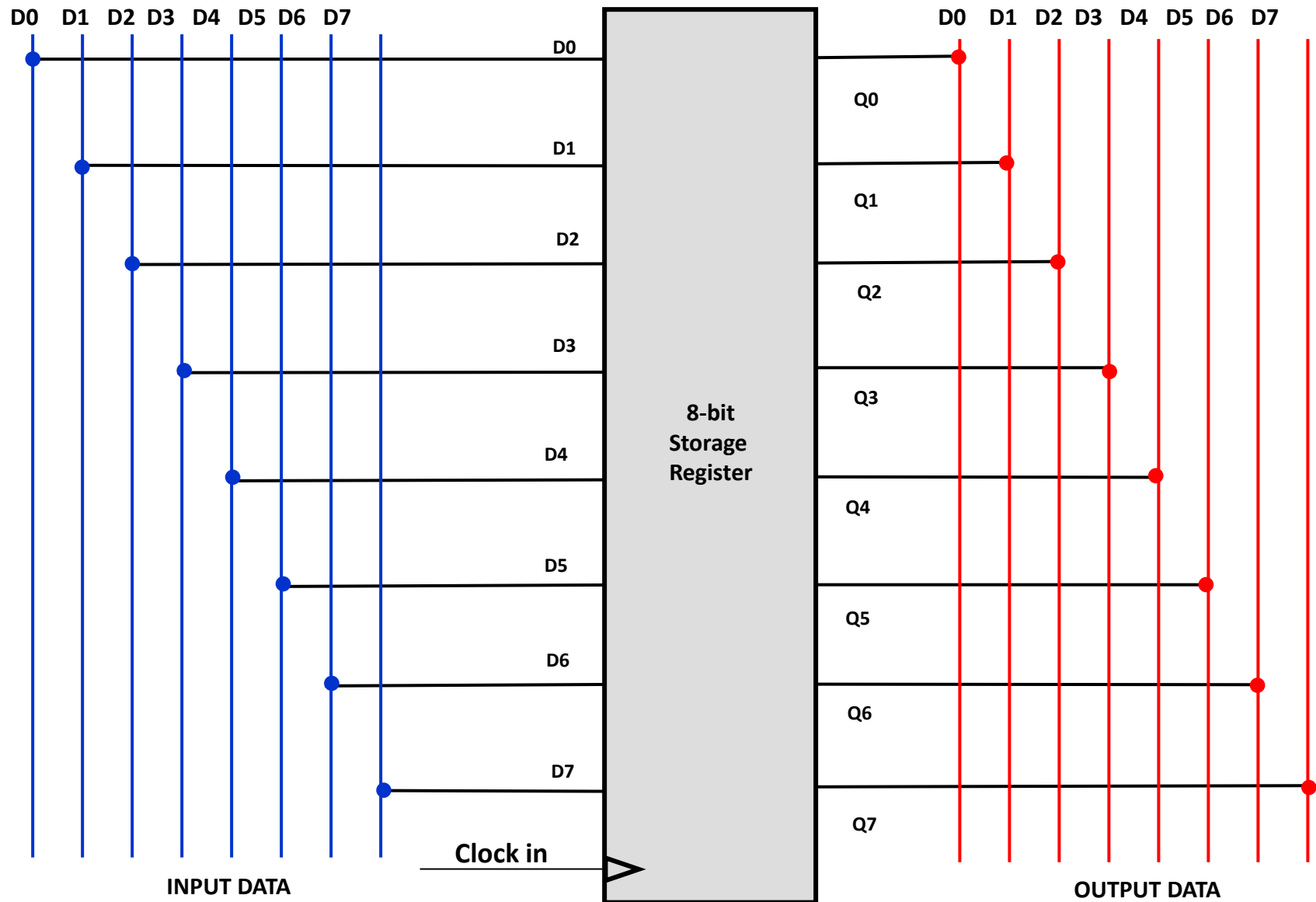


- Shift register moves data through successive stages on each clock pulse
- Used for serial data communications, multiplication, image processing
- **Basis for UART (*Universal Asynchronous Receiver/Transmitter*)**
- Data can be **read in serial** and then **read out in parallel**
- Serial data communications limit the number of signal wires needed to transmit byte-wide

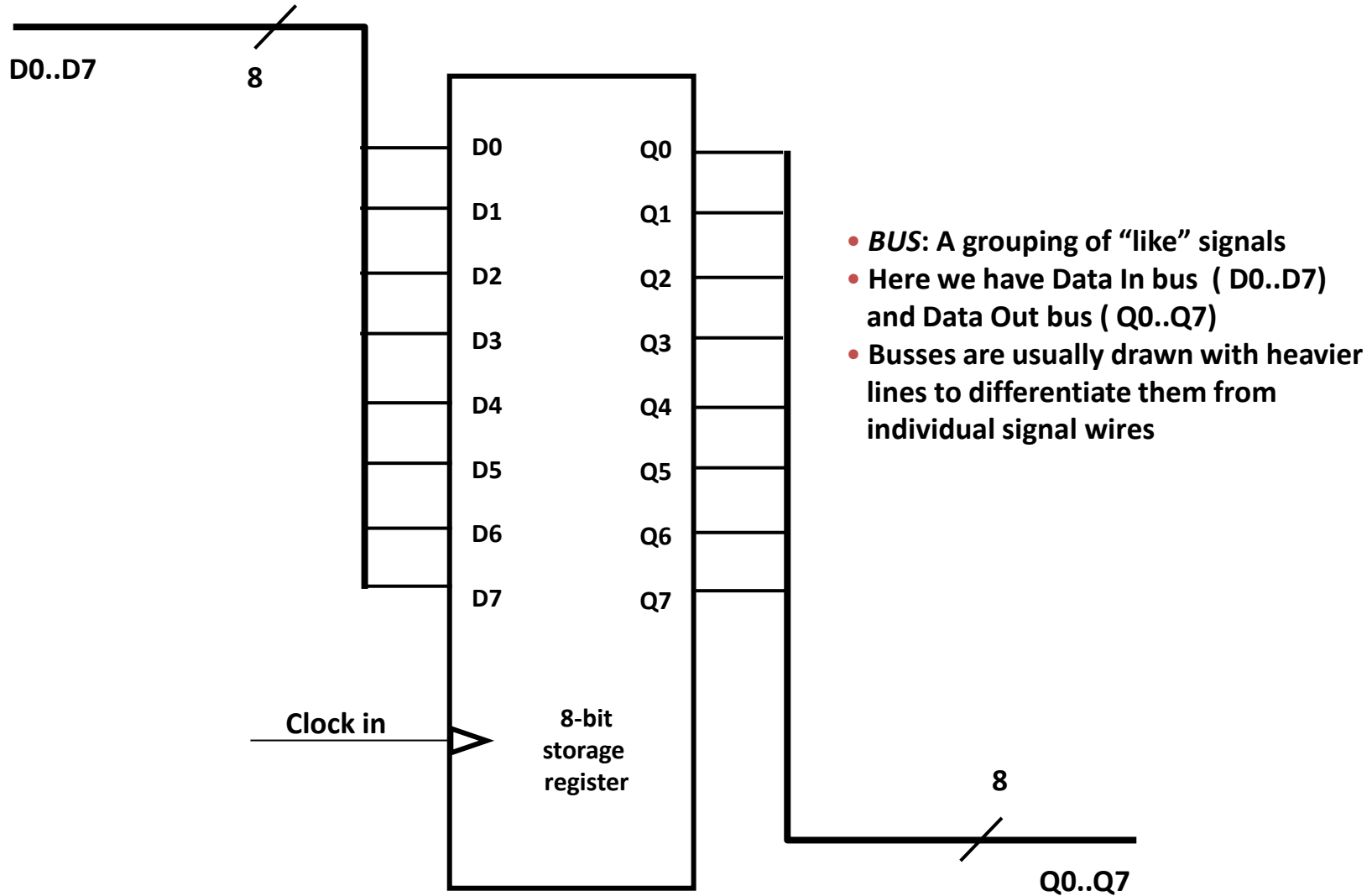
"D" Flip-Flop as a Storage Register



An 8-bit Storage Register



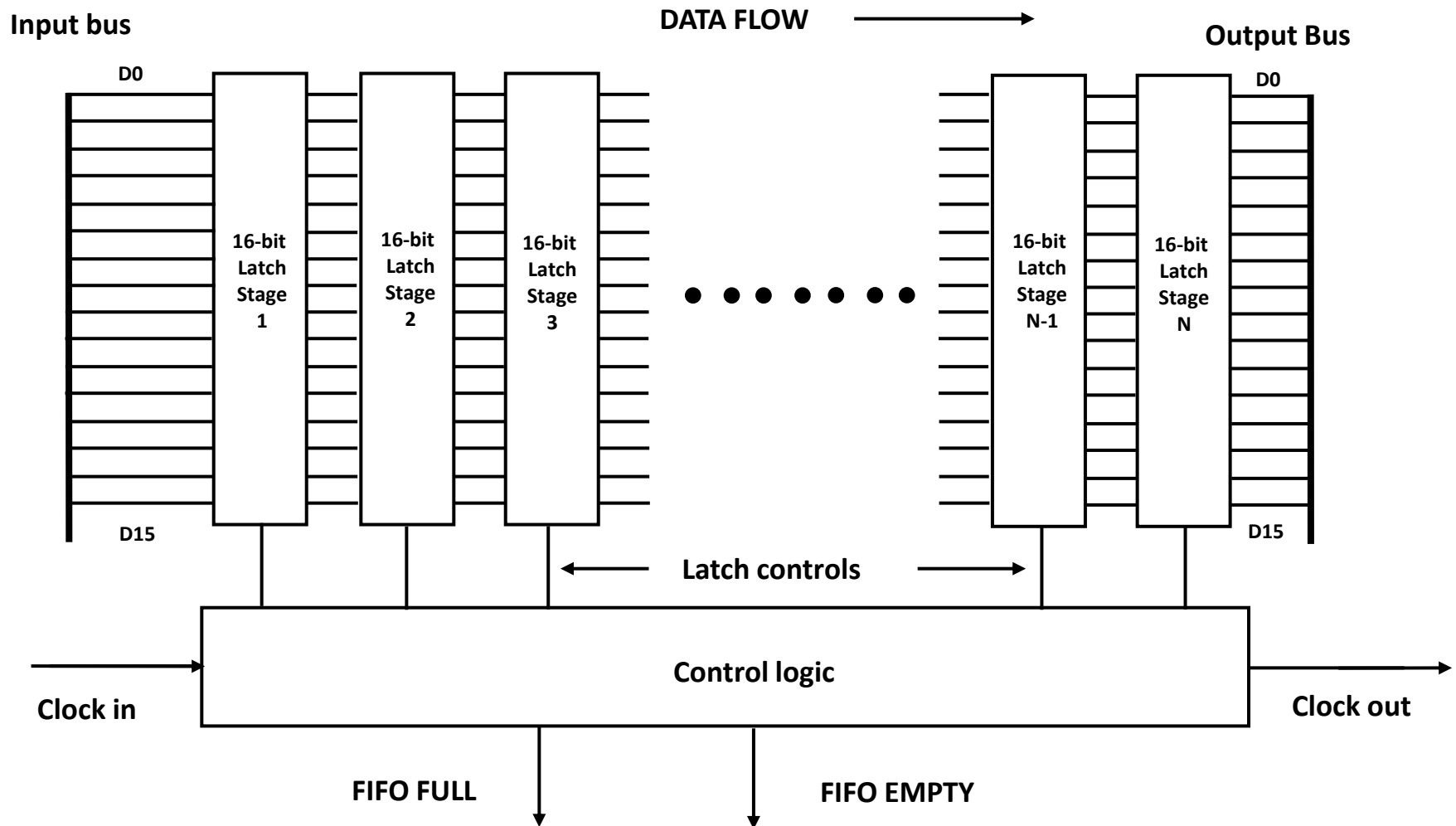
The Storage Register with Busses



More on Storage Registers

- The storage register provides a stable location to store data moving along buses
 - Data on buses are transitory
 - The width of a storage register typically matches the width of the data path
 - May be 4, 8, 16, 32, 64 or 128 bits wide
 - Registers interface data between a computer and the outside world
 - Registers are the key data holders in computers and microprocessors
 - A computer's architecture is often defined by organization of the storage registers
- Two variety of storage registers: "D" type and Latch

First In First Out (FIFO) Data Storage



Summary

- Gate is a fundamental building block of all digital systems
- Flip-flop is a basic unit for sequential circuit, which are built using **gates** and a **clock**
- Design a sequential circuit system
 - Draw a state diagram
 - Draw a state table
 - Build K-maps to derive Boolean equations
 - Draw a circuit diagram based on the equations