UMBC

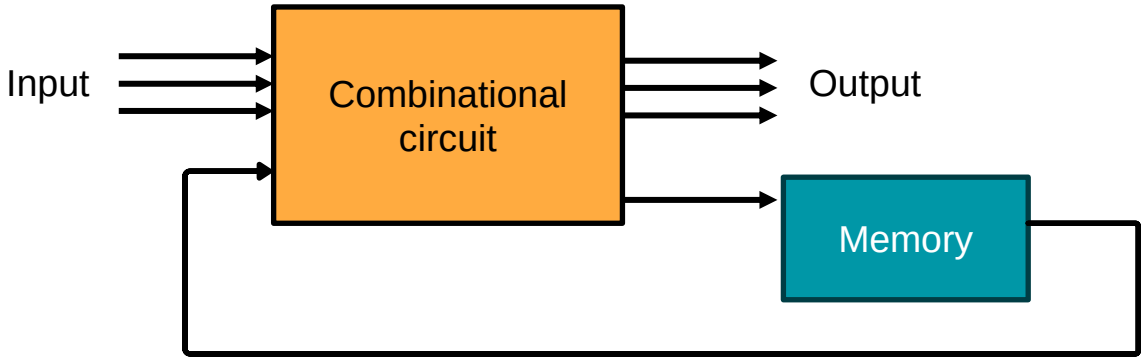
# Sequential Circuits

CMSC 313

Raphael Elspas

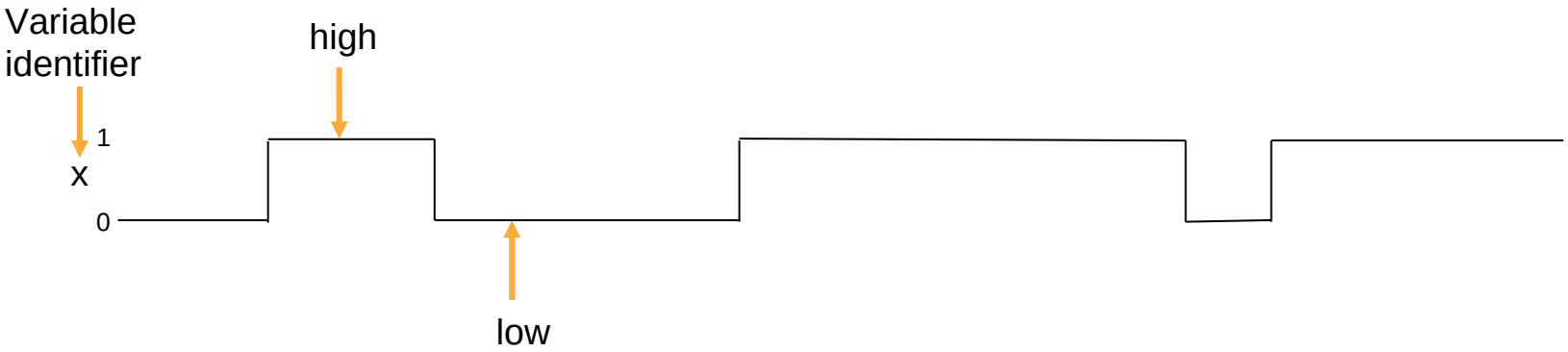
# Introduction

- In **combinational circuits**, the present output depends on the present input.
- In **sequential circuits** the present output depends on the present input as well as past outputs

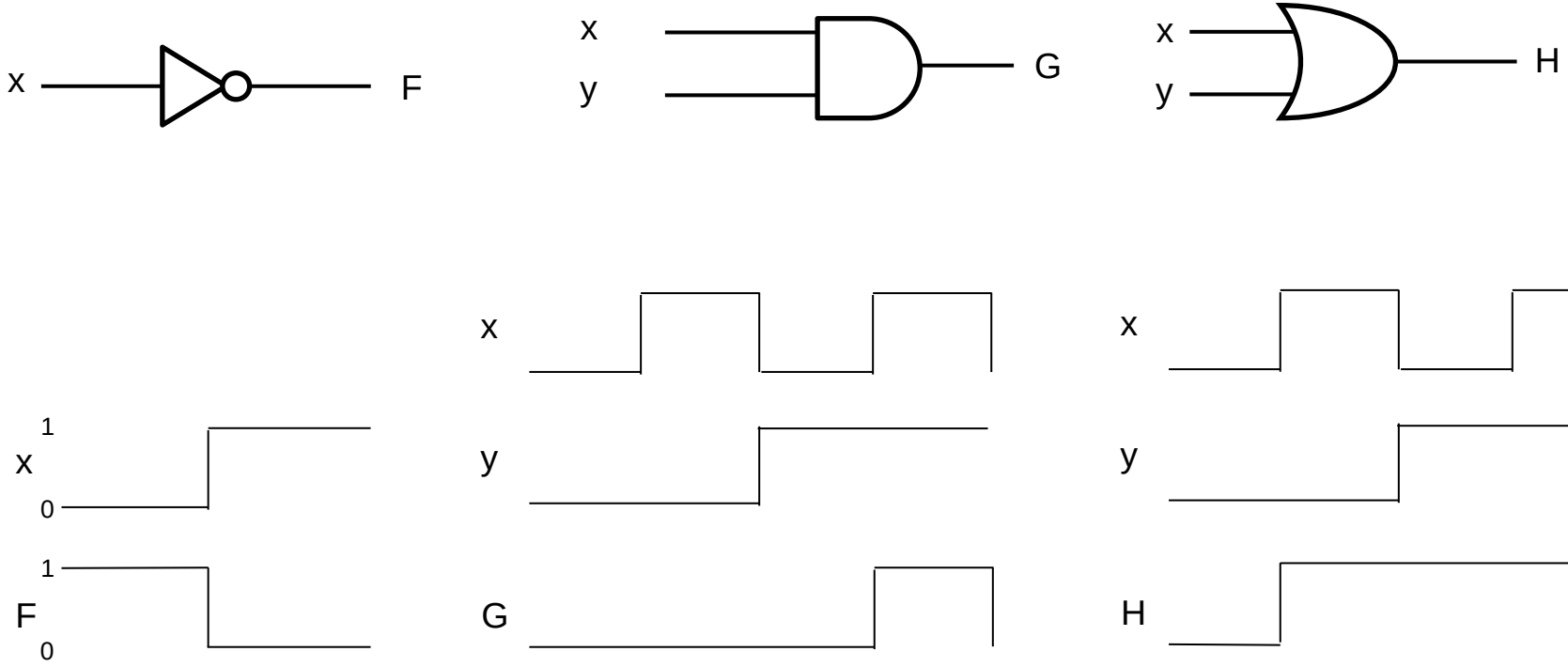


# Waveforms

- A way to represent digital values at any point in a circuit with respect to time.
- Historical representation of circuit values
- Can show simultaneous values across entire circuit
- Also called traces



# Waveforms, cycle through inputs

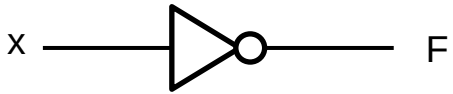


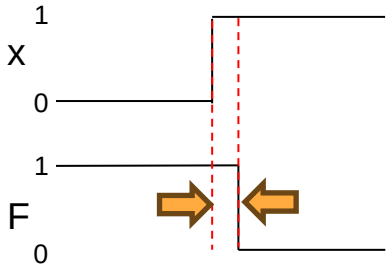
# Gate Delay

- Gate delay is time it takes for a digital logic gate to produce an output in response to a change in its input
- Gate delay is measured from the time when inputs to a gate change until the time when the output stabilizes.
- Two types:
  - **Propagation Delay**: the time it takes for the output to change after a change in input. It includes both the delay caused by the signal propagation through the gate and the time required for the internal circuitry to settle to a stable state.
  - **Rise/Fall Time**: the times required for the output signal to transition from one logic level to another. Rise time is the time taken for the output to change from low to high, while fall time is the time taken for the output to change from high to low.

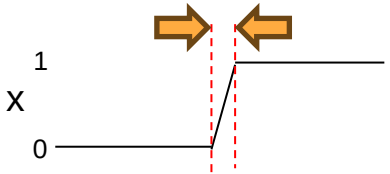
# Delay Visualization

- **Propagation Delay**: the time it takes for the output to change after a change in input.



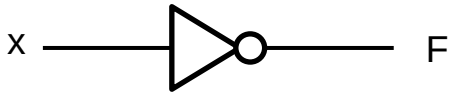



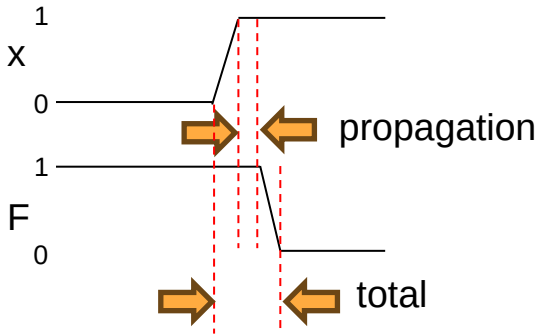

- **Rise/Fall Time**: the times required for the output signal to transition from one logic level to another.



# Propagation and Rise/Fall together

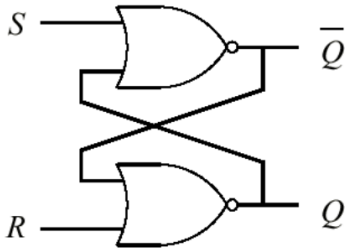
- Delay compounds: Total = $\tau_{x,rise} + \tau_{gate} + \tau_{F,fall}$



# SR Latch (NOR)

- The basic storage element is called a latch
- S means *set*, R means *reset*




Case 1: $S = 0, R = 1, Q = 0, \overline{Q} = 1$

$S = 0, R = 0, Q = 0, \overline{Q} = 1$ ← memory store 0

Case 2: $S = 1, R = 0, Q = 1, \overline{Q} = 0$

$S = 0, R = 0, Q = 1, \overline{Q} = 0$ ← memory store 1

Case 3: $S = 1, R = 1, Q = 0, \overline{Q} = 0$

$S = 0, R = 0, Q = 0, \overline{Q} = 1$ analysis from Q

$Q = 1, \overline{Q} = 0$ analysis from $\overline{Q}$

} Two different outputs for same analysis

NOR

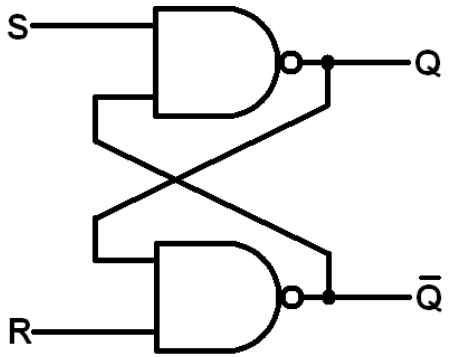
| A | B | $\overline{A + B}$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

SR Latch (NOR)

| S | R | $Q$ | $\overline{Q}$ |
|---|---|---|---|
| 0 | 0 | Hold output | |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | Not used | |

# SR Latch (NAND)

- Use a NAND instead of NOR for similar properties




NAND

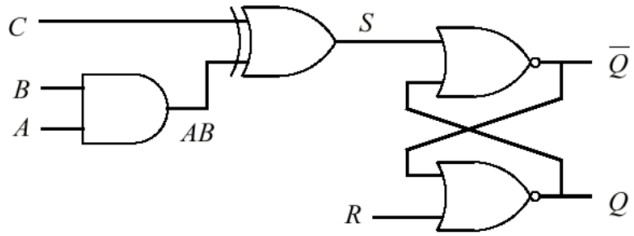
| A | B | $\overline{AB}$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

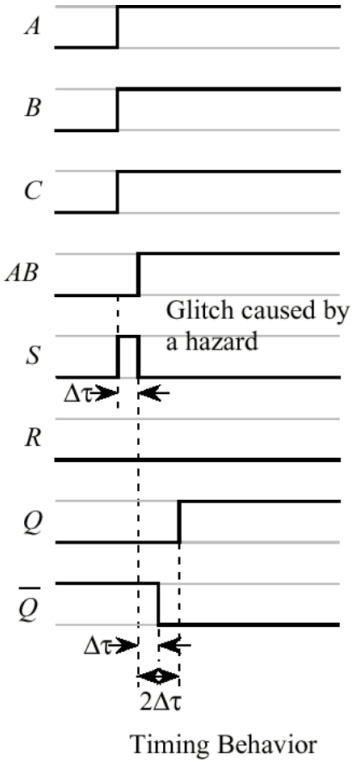
SR Latch (NAND)

| S | R | Q | $\overline{Q}$ |
|---|---|---|---|
| 0 | 0 | Not used | |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | Hold output | |

# Hazard

- It is desirable to turn off the latch so that it doesn't respond to such hazards

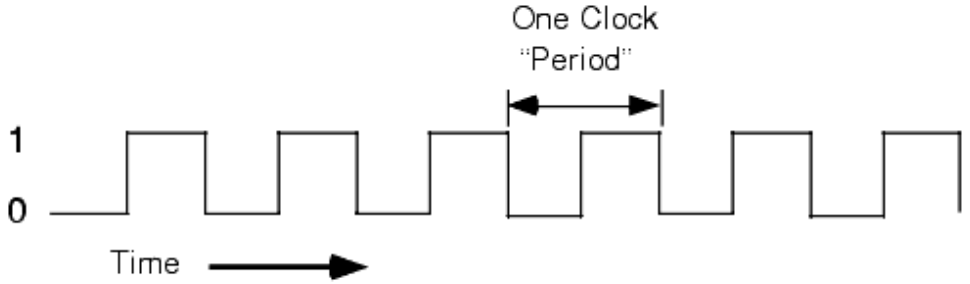



Timing Behavior

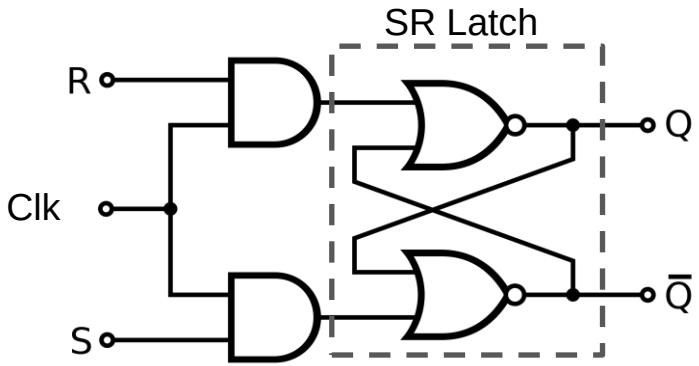
# Clock

- In a sequential circuit, we have inputs, feedback and changing states and we need to keep everything in time with a **clock**.
- A clock is a signal that goes from low to high with a duty cycle of 50%
- A clock decides the time of an input to a sequential circuit.
- When the signal is evaluated to 1 we call it “high”, when 0 we call it “low” and we call the in between states “rising edge” and “falling edge”



# SR Flip Flop

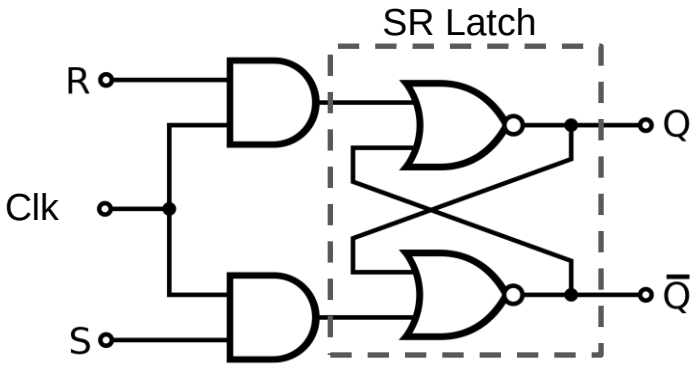
- The SR latch provides allows hazards to trigger the value storage.
- The SR Flip Flop is a **gated** SR Latch which prevents hazards from interrupting the circuit.
- Feeding a clock (clk) into the input of the gate creates a **level trigger**.

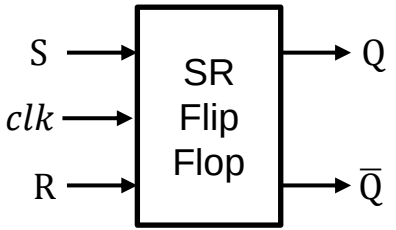



| clk | S | R | Q | $\overline{Q}$ |
|---|---|---|---|---|
| 0 | X | X | Hold value | |
| 1 | 0 | 0 | Hold value | |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | Not used | |

# SR Flip Flop

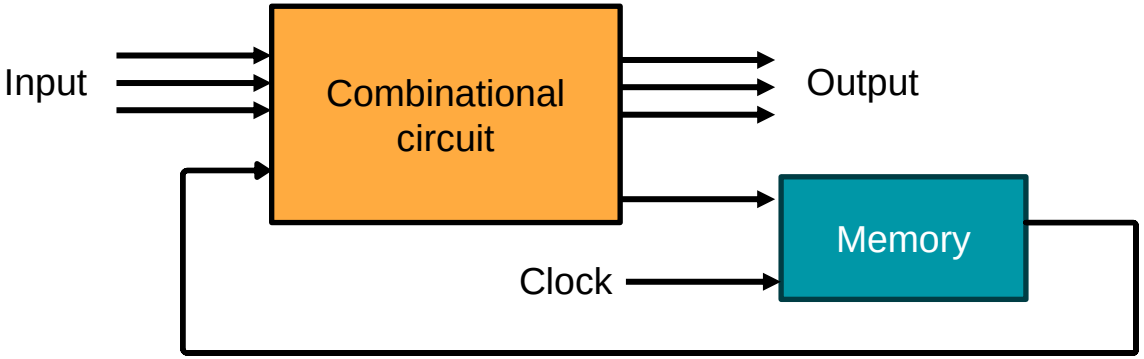
- Put it in a box

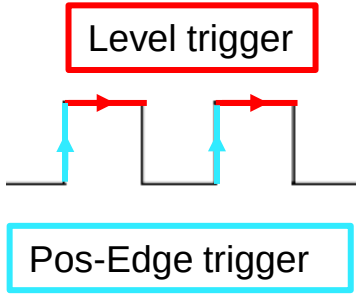


# Triggering methods

- There are two methods of triggering a signal with a clock
  - Level triggering: whenever clock is high, change can happen within the circuit
  - Edge triggering, positive edge/negative edge: memory element will trigger when clock transitions from low to high

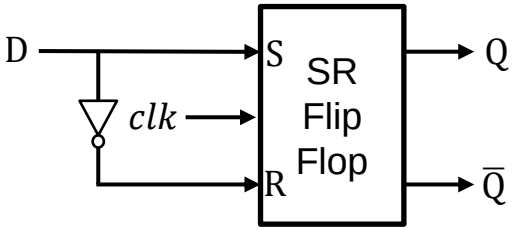


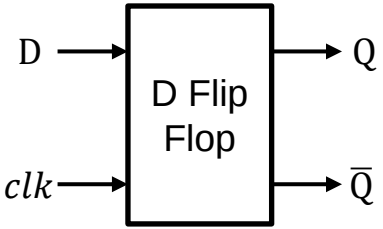



# D Flip Flop

- We want to prevent invalid state as an output
- Idea: force S and R to be opposites

| | D | |
|---|---|---|
| clk | D | $Q_{n+1}$ |
| 0 | X | $Q_n$ |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

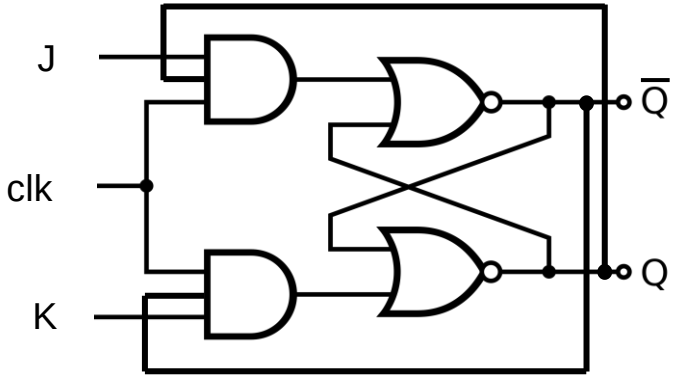


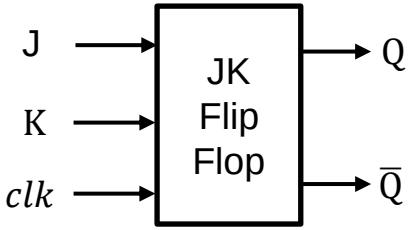



# JK Flip Flop

- SR flip flops have an invalid state, lets repurpose that state for something useful
- JK flip flops will behave same as SR flip flops, but with S=1, R=1 being a Toggle switch

| clk | J | K | $Q_{n+1}$ |
|---|---|---|---|
| 0 | X | X | $Q_n$ |
| 1 | 0 | 0 | $Q_n$ |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | $\overline{Q_n}$ |

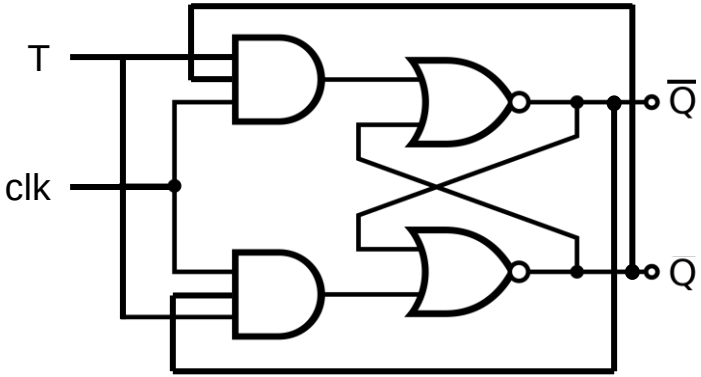


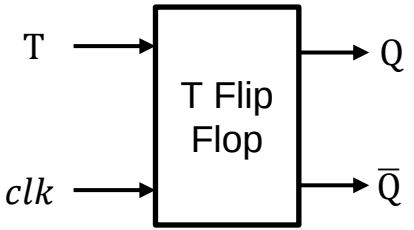



# T Flip Flop

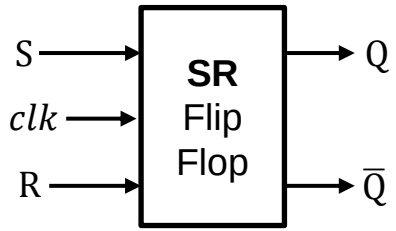
- A T flip flop implements toggling the past input.
- If T = 0, do not toggle previous input
- If T = 1, toggle previous input
- Take a JK and only allow states: J=0, K=0 and J=1, K=1. Tie J and K together

| clk | T | $Q_{n+1}$ |
|---|---|---|
| 0 | x | $Q_n$ |
| 1 | 0 | $Q_n$ |
| 1 | 1 | $\overline{Q_n}$ |

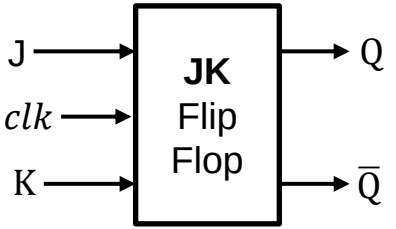


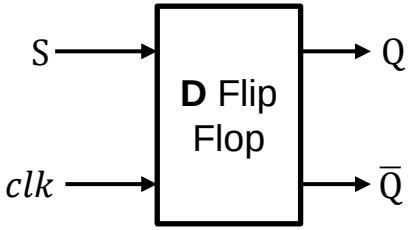
# FF summary



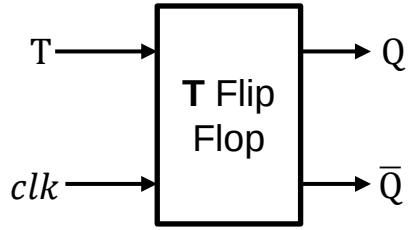

| S | R | $Q_{n+1}$ |
|---|---|---|
| 0 | 0 | $Q_n$ |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | Invalid |




| J | K | $Q_{n+1}$ |
|---|---|---|
| 0 | 0 | $Q_n$ |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | $\overline{Q_n}$ |




| D | $Q_{n+1}$ |
|---|---|
| 0 | 0 |
| 1 | 1 |

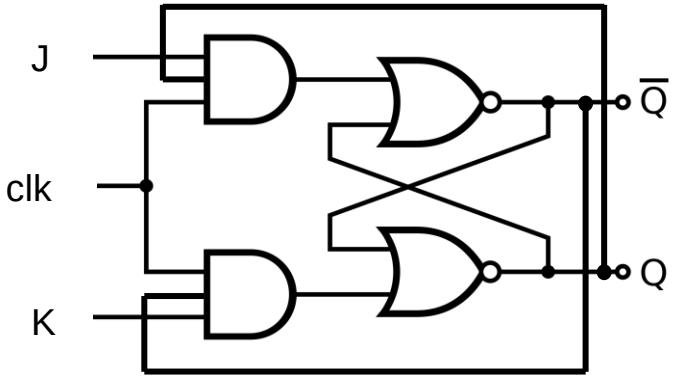


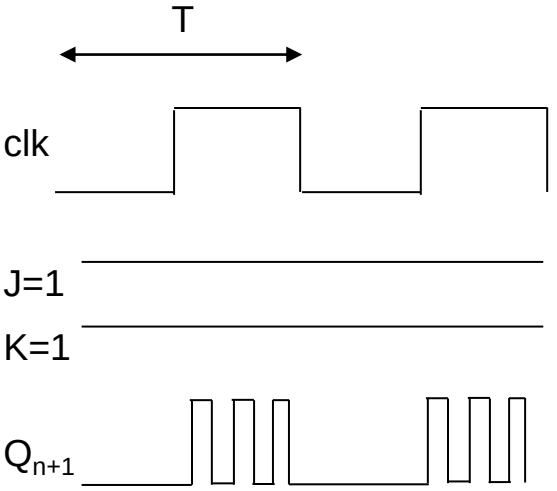

| T | $Q_{n+1}$ |
|---|---|
| 0 | $Q_n$ |
| 1 | $\overline{Q_n}$ |

# JK Flip Flop race condition

- We created flip flops from latches to avoid hazard issues
- We created JK flip flops from SR flip flops to avoid invalid states
- When we set J=1, K=1 for toggle mode, it’s possible for a race condition to start
- The race condition happens since the output of the circuit gets fed back into the input while the clock is still high, and this feedback can occur multiple times.

# JK Flip Flop race condition



# Overcoming JK Flip Flop race condition

Overcoming race condition can use any of these techniques:

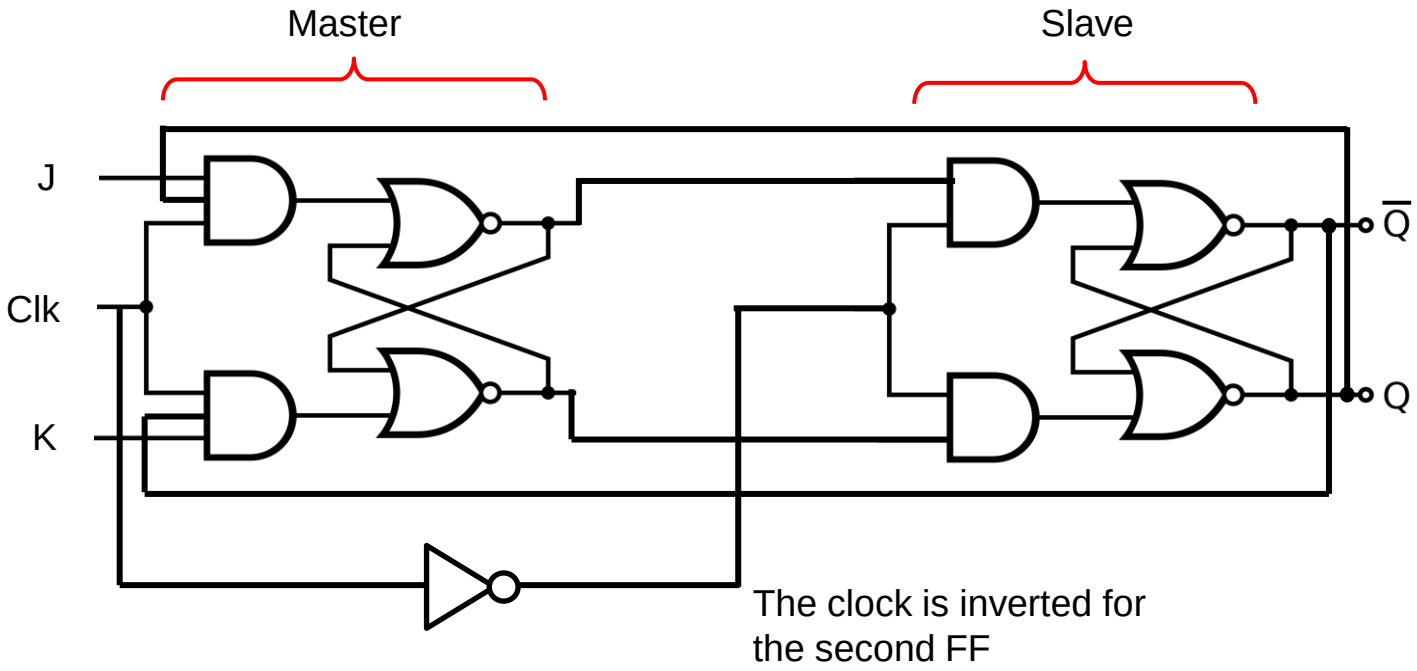
- T/2 < gate delay from FF
- Edge triggering
- Master-Slave flip flop

} These are actually the same

# Master Slave JK Flip Flop

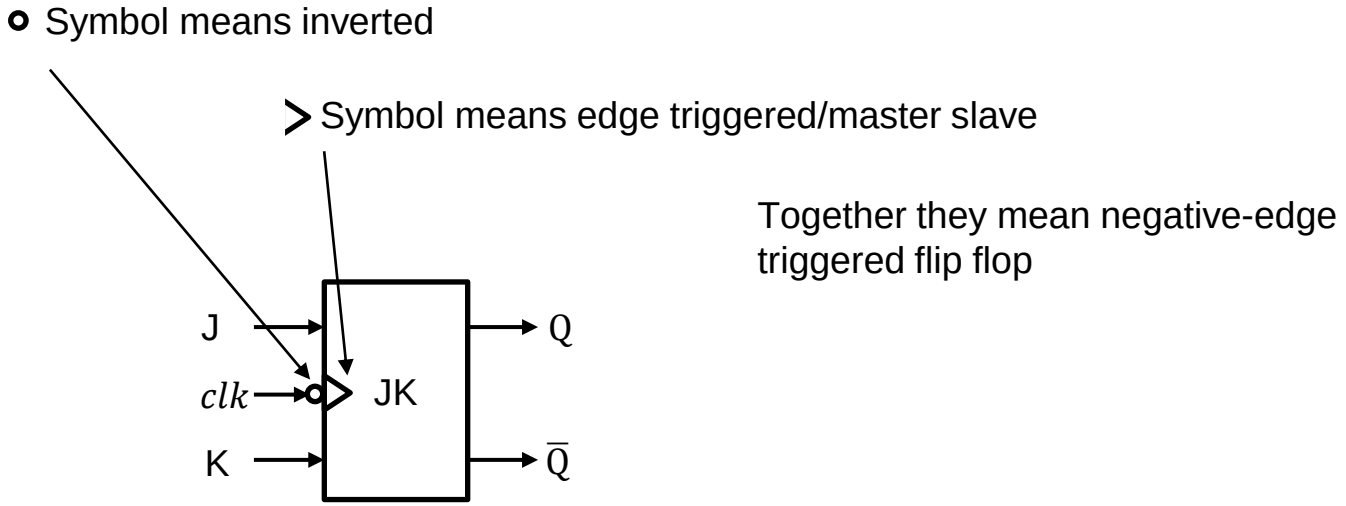
- We will have two flip flops, a master and a slave, where the clock is complemented for the second flip flop
- When the clock is high, the first FF gets written to, but the second FF has its clock inverted so the second FF will maintain its state.
- Instead of the output changing continuously from 1 to 0, the output will change once in a clock cycle

# Master Slave JK Flip Flop



# JK edge triggered Flip Flop

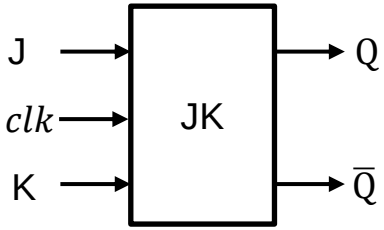
- Master slave is equivalent to negative edge triggered flip flop



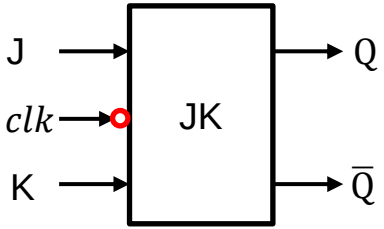
# All clock triggers

JK FFs are used as an example, but these triggers work for all FFs.

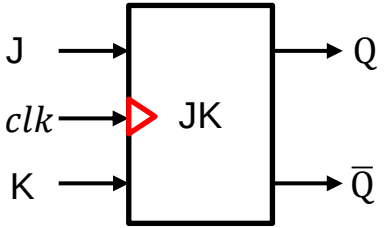
Level trigger



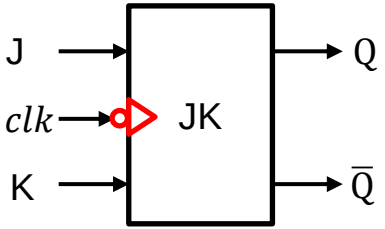

Negated (or inverted) Level trigger




Positive edge trigger

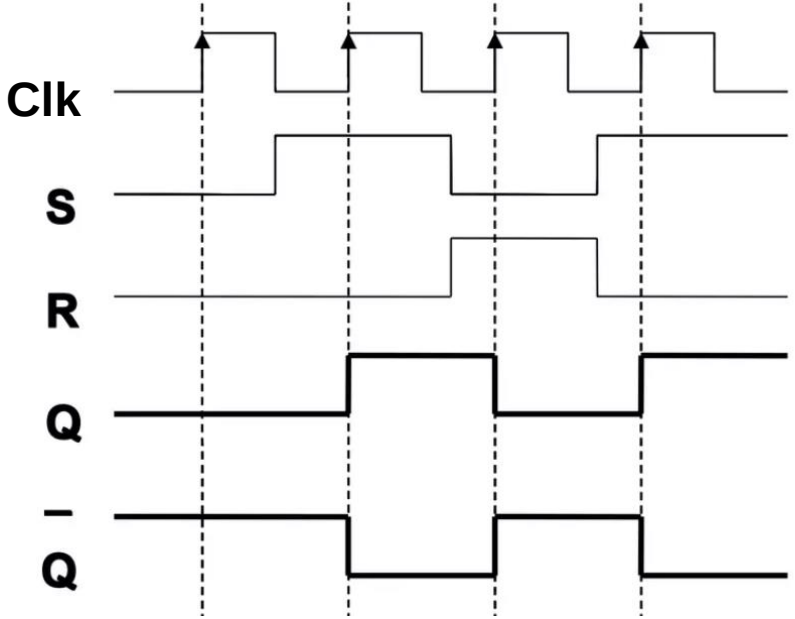



Negative edge trigger

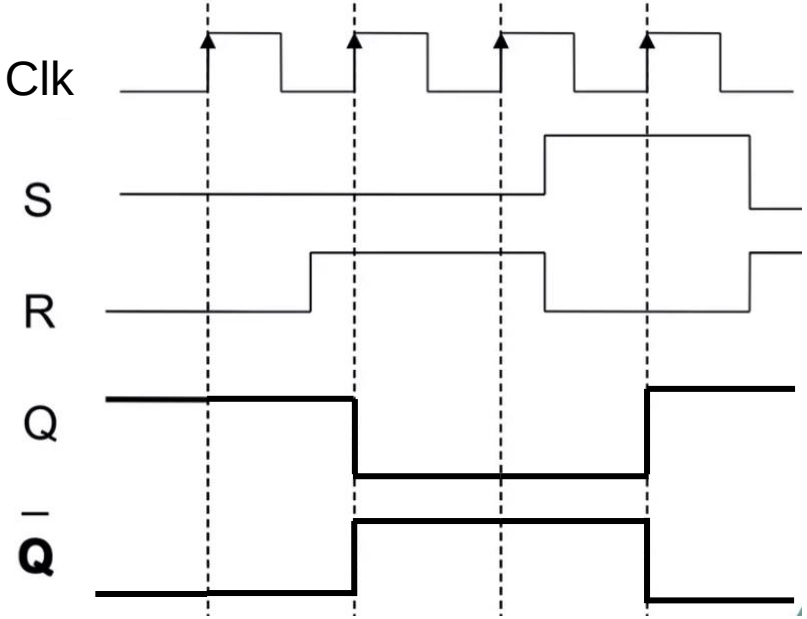


# Example waveforms

Pos edge trigger, Q initially 0

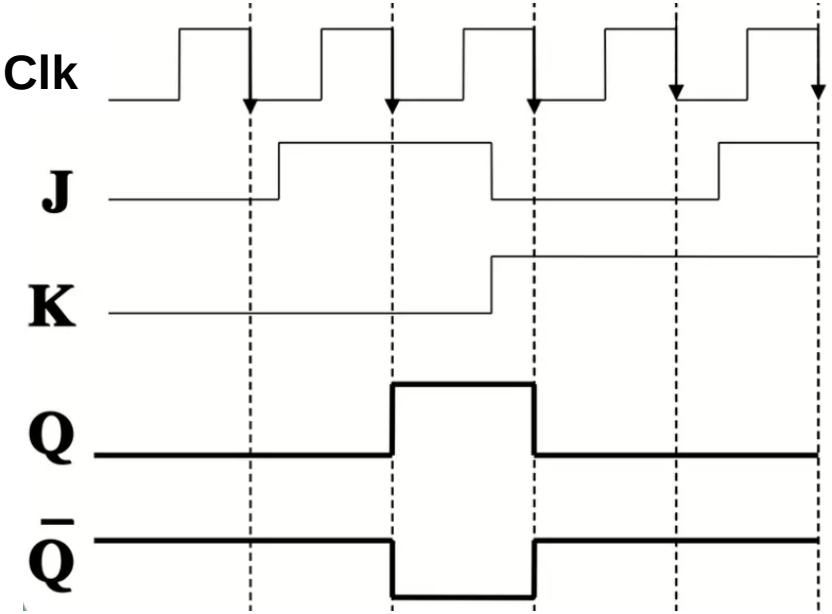



Pos edge trigger, Q initially 1

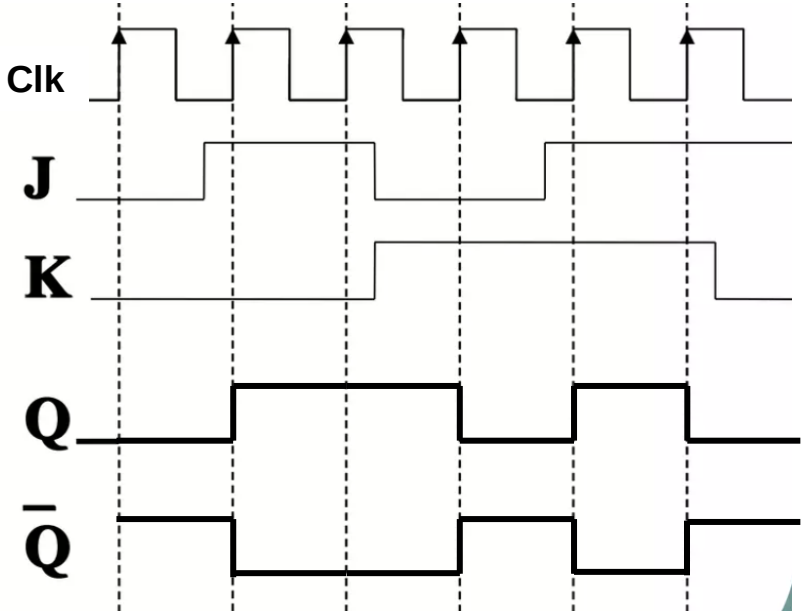


# Another example

Neg Edge Trigger, Q starts low




Pos Edge Trigger, Q starts low



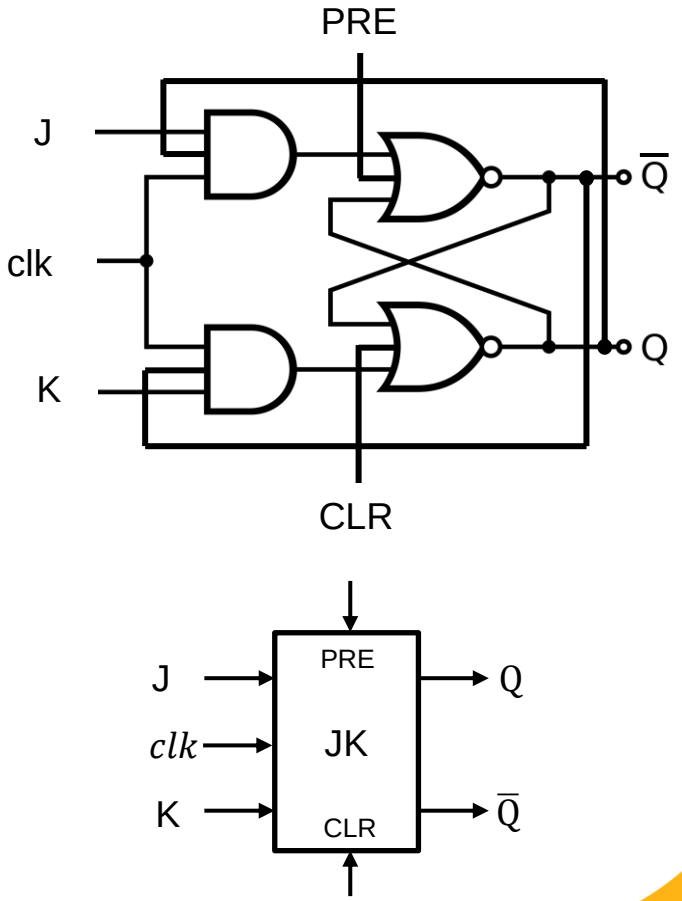
# Asynchronous trigger

- The normal data inputs to a flip flop (D, S and R, T, or J and K) are referred to as synchronous inputs because they effect the output only with respect to the clock signal.
- We can asynchronously trigger a flip flop by bypassing the clock:
- We'll use Preset (PRE) to mean set to 1 asynchronously (when PRE = 1)
- We'll use Clear (CLR) to mean set to 0 asynchronously (when CLR = 1)
- Note: sometimes these inputs are inverted: $\overline{\text{PRE}}$ or $\overline{\text{CLR}}$. These will have to be set to the reverse. Their signal is active when they are low (0).

# Asynchronous trigger

- For a JK flip flop, for example, there will be 6 possible inputs:

| J | K | PRE | CLR | $Q_{n+1}$ | timing |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | $Q_n$ | synchronous |
| 0 | 1 | 0 | 0 | 0 | synchronous |
| 1 | 0 | 0 | 0 | 1 | synchronous |
| 1 | 1 | 0 | 0 | $\overline{Q_n}$ | synchronous |
| x | x | 1 | 0 | 1 | asynchronous |
| x | x | 0 | 1 | 0 | asynchronous |



# Asynchronous trigger

PRE and CLR are active high

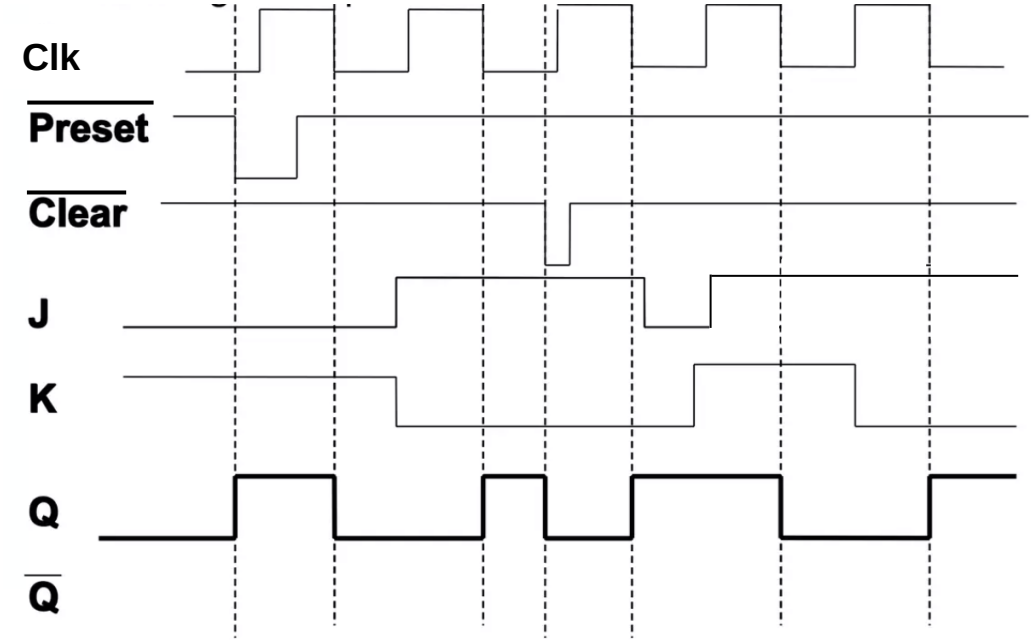
| J | K | PRE | CLR | $Q_{n+1}$ | timing |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | $Q_n$ | synchronous |
| 0 | 1 | 0 | 0 | 0 | synchronous |
| 1 | 0 | 0 | 0 | 1 | synchronous |
| 1 | 1 | 0 | 0 | $\overline{Q_n}$ | synchronous |
| x | x | 1 | 0 | **1** | asynchronous |
| x | x | 0 | 1 | **0** | asynchronous |

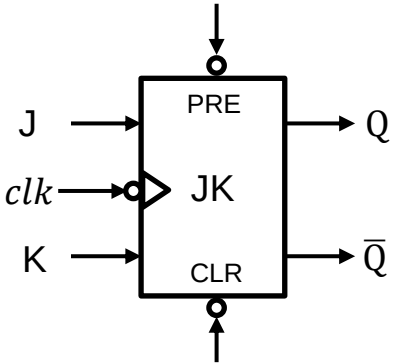
$\overline{\text{PRE}}$ and $\overline{\text{CLR}}$ are active low

| J | K | $\overline{\text{PRE}}$ | $\overline{\text{CLR}}$ | $Q_{n+1}$ | timing |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | $Q_n$ | synchronous |
| 0 | 1 | 1 | 1 | 0 | synchronous |
| 1 | 0 | 1 | 1 | 1 | synchronous |
| 1 | 1 | 1 | 1 | $\overline{Q_n}$ | synchronous |
| x | x | 0 | 1 | **1** | asynchronous |
| x | x | 1 | 0 | **0** | asynchronous |

# Example Asynchronous/synchronous mixed

Negative edge trigger, Q starts low



# Applications

- Counters
- Frequency Dividers
- Shift Registers
- Storage Registers

# Summary

- A flip flop is a latch with gated inputs
- Four main FFs: SR, D, JK, & T
- Master Slave used to create edge trigger and avoid race condition
- Asynchronous triggers bypass clocked gates

# References


- https://redirect.cs.umbc.edu/courses/undergraduate/CMSC313/Fall03/cpatel2/slides/slides19.pdf
- https://www.youtube.com/playlist?list=PLmjEXDyU3L-mSz3eG4_JwVZt2fSon3tQX