Sequential Logic

Dov Kruger

Department of Electrical and Computer Engineering
Rutgers University

October 29, 2024

Kinds of Logic

- Combinational Logic: outputs depend only on the current inputs
- Sequential Logic: outputs depend on the current inputs and previous state
- State: the memory of the circuit
- Latches and flip-flops are the basic memory elements
- We have learned how to build memory using D-type Flip-Flops
- Now, it's time to learn to build them

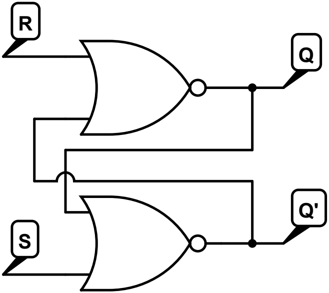
- Sequential logic circuits have memory
- Output depends on both current inputs and previous state
- Key components: latches and flip-flops
- Used for storing information and creating state machines
- Essential for creating registers, counters, and memory elements

Types of Sequential Logic Elements

- Latches (Level-triggered)
  - SR Latch (Set-Reset)
  - D Latch (Data)
- Flip-Flops (Edge-triggered)
  - JK Flip-Flop
  - D Flip-Flop
  - T Flip-Flop (Toggle)
- Special configurations
  - Primary/Secondary Flip-Flop (formerly Master-Slave)
  - Edge-Triggered Flip-Flop implementations

SR Latch Active High (NOR Implementation)

- Simplest form of latch
- Two inputs: Set (S) and Reset (R)
- Two outputs: Q and Q' (complement of Q)
- Disadvantage: Undefined state when $S = R = 1$

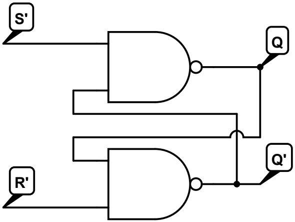


S	R	Q	Q’
0	0	Q	Q’
0	1	0	1
1	0	1	0
1	1	X	X

SR Latch Active Low (NAND Implementation)

- Same two inputs: Set ($\overline{S}$) and Reset ($\overline{R}$)
- Two outputs: Q and Q' (complement of Q)
- Disadvantage: Undefined state when $\overline{S} = \overline{R} = 1$

S	R	Q	Q'
0	0	Q	Q'
0	1	0	1
1	0	1	0
1	1	X	X



Gated SR Latch

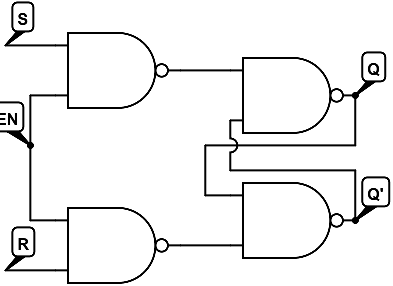
- Similar to SR latch but with an additional Enable input (E)
- Two inputs: Set (S) and Reset (R)
- Two outputs: Q and Q' (complement of Q)
- When Enable (E) is high, the latch behaves like a regular SR latch
- When Enable (E) is low, the latch holds its previous state regardless of S and R inputs
- Eliminates the problem of asynchronous changes in S and R inputs

E	S	R	Q	Q'
0	X	X	Q	Q'
1	0	0	Q	Q'
1	0	1	0	1
1	1	0	1	0
1	1	1	X	X

Youtube video animation:

https://www.youtube.com/watch?v=peCh_859q7Q

Gated SR Latch

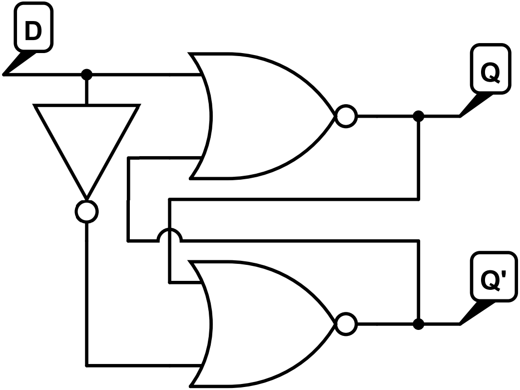


D Latch

- Eliminates the undefined state of SR latch
- Single data input (D) and enable input
- Output follows the input when enabled
- Holds the previous state when disabled
- Building block for more complex flip-flops

E	D	Q	Q'
0	X	Q	Q'
1	0	0	1
1	1	1	0

D Latch

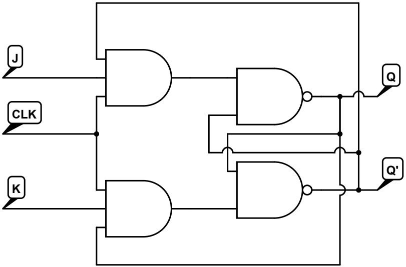


JK Flip-Flop

- Improvement over SR flip-flop
- Two inputs: J (Set) and K (Reset)
- Clock input for synchronization
- Toggles output when J=K=1
- No undefined states

J	K	Q	Q'
0	0	Q	Q'
0	1	0	1
1	0	1	0
1	1	Q'	Q

JK Flip-Flop

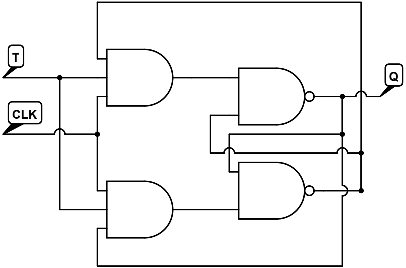


T Flip-Flop

- Toggle flip-flop
- Single input T (Toggle) and clock input
- Implemented using JK flip-flop with J=K=T
- Output toggles when T=1 and clock edge occurs
- Maintains state when T=0
- Useful for creating counters and frequency dividers

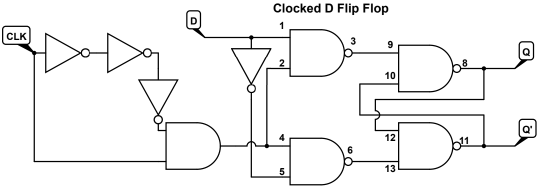
T	Q
0	Q
1	Q'

T Flip-Flop



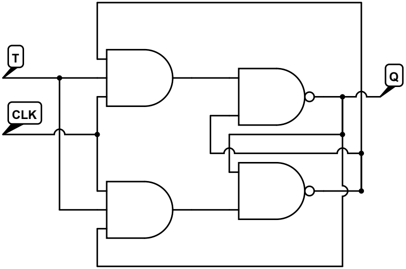
D Flip-Flop (Rising Edge)

- Most commonly used flip-flop
- Single data input (D) and clock input
- Output changes only on clock edge (rising or falling)
- Stores the input state at the time of clock edge
- Used in registers and data storage elements
- This implementation may be a hack?



T Flip-Flop

- Toggle flip-flop
- Single input T (Toggle) and clock input
- Output toggles when T=1 and clock edge occurs
- Maintains state when T=0
- Useful for creating counters and frequency dividers



Edge-Triggered Flip-Flop

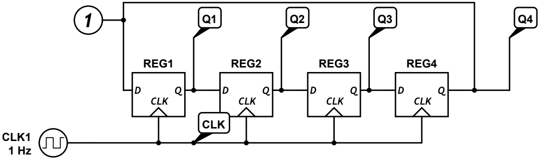
- Changes state only at a specific clock edge (rising or falling)
- Provides precise timing control
- Commonly used in synchronous digital systems
- Can be implemented as positive-edge or negative-edge triggered
- Most modern flip-flops are edge-triggered

Counters: Introduction

- Counters are sequential circuits that count pulses
- They can count up, down, or both
- Implemented using flip-flops (usually T or JK flip-flops)
- Used in various applications: timers, frequency dividers, etc.

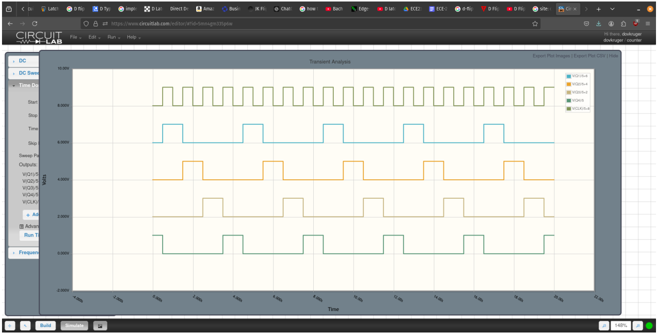
Power of Two Counter

- Simplest form of counter
- Counts from $0 to 2^n - 1$, where n is the number of bits
- Implemented using T flip-flops
- Each flip-flop toggles at half the frequency of the previous one

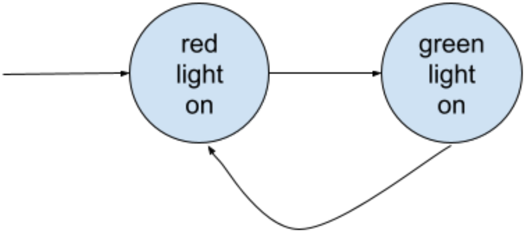


4-Bit Power of Two Counter: Operation

- Counts from 0000 to 1111 (0 to 15 in decimal)
- LSB toggles every clock cycle
- Each subsequent bit toggles when all previous bits are 1
- Automatically resets to 0000 after 1111

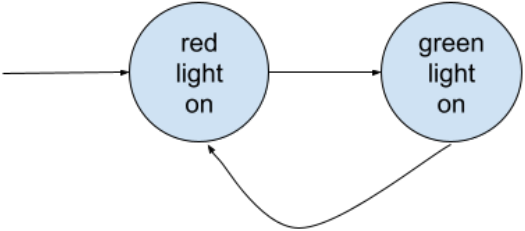


- State machines are sequential circuits that transition between states based on inputs
- Start in a known start state
- Transition to other states based on input signals
- Used in various applications: control systems, communication protocols, etc.



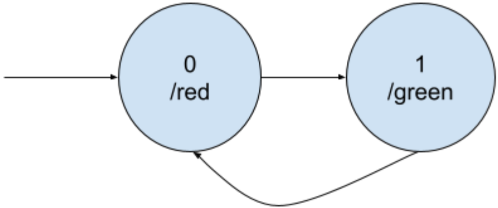
Implementing State Machines with D Flip-Flops

- State machines can be implemented using D flip-flops
- n bits can represent a maximum of 2^n different states
- Each flip-flop stores one bit of the state
- State transitions are determined by the input and current state



State Machine Functions

- Two main functions in a state machine:
  - Output function: Defines the output for each state
  - Transition function: Defines how each state transitions to the next state based on inputs
  - Assign a unique number to each state
  - $bits = \lceil log_2(number\ of\ states) \rceil$



State Machine Example: Red Light Green Light

D_0	R	G
0	1	0
1	0	1

In this case, you can just toggle the state with a T flip-flop.

In general you need to determine for each state how to transition to the next state.

Example: 2-bit State Machine

- State 00: Blue LED turns on
- State 01: Red LED turns on
- State 11: Green LED turns on
- State 10: All LEDs are off

D_1	D_0	R	G	B
0	0	0	0	1
0	1	1	0	0
1	1	0	1	0
1	0	0	0	0

Q_1	Q_0	D_1	D_0
0	0	0	1
0	1	1	1
1	1	1	0
1	0	0	0

Now that you know what state you want to be in, we can implement it

State Machine Schematic