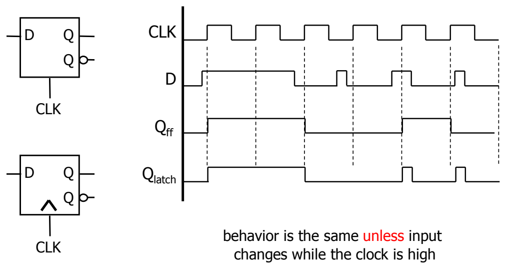
## Overview

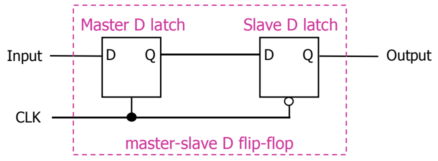
- Last lecture
  - Sequential Verilog
- Today
  - Review of D latches and flip-flops
  - T flip-flops and SR latches
  - State diagrams
  - Asynchronous inputs

## Latches versus flip-flops



behavior is the same unless input changes while the clock is high

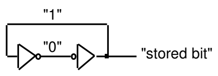
## The master-slave D



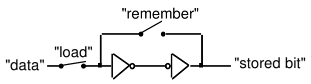
Class example: Draw the timing diagram

## How do we make a latch?

- Two inverters hold a bit
  - As long as power is applied

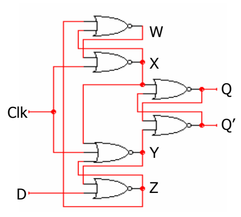


- Storing a new memory
  - Temporarily break the feedback path



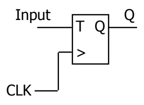
## How do we make a D F/F?

- Edge triggering is difficult
  - Label the internal nodes
  - Draw a timing diagram
  - Start with Clk=1



## T flip-flop

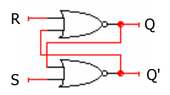
- Full name: Toggle flip-flop
- Output toggles when input is asserted
  - If T=1, then Q → Q' when CLK ↑
  - If T=0, then Q → Q when CLK ↑



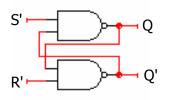
| Input($t$) | Q($t$) | Q($t+\Delta t$) |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

## The SR latch

- Cross-coupled NOR gates
  - Can set (S=1, R=0) or reset (R=1, S=0) the output

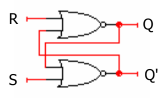


- Cross-coupled NAND gates
  - Can set (S=1, R=0) or reset (R=1, S=0) the output

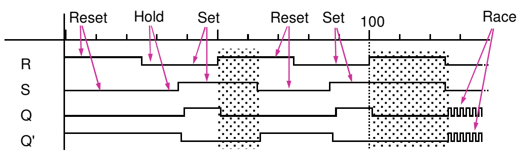


## SR latch behavior

- Truth table and timing

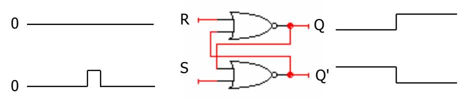


| S | R | Q |
|---|---|---|
| 0 | 0 | hold |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | disallow |



## SR latch is glitch sensitive

- Static 0 hazards can set/reset latch
  - Glitch on S input sets latch
  - Glitch on R input resets latch



## Clear and preset in flip-flops

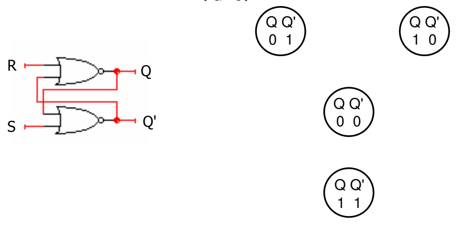
- Clear and Preset set flip-flop to a known state
  - Used at startup, reset
- Clear or Reset to a logic 0
  - Synchronous: Q=0 when next clock edge arrives
  - Asynchronous: Q=0 when reset is asserted
    - Doesn't wait for clock
    - Quick but dangerous
- Preset or Set the state to logic 1
  - Synchronous: Q=1 when next clock edge arrives
  - Asynchronous: Q=1 when reset is asserted
    - Doesn't wait for clock
    - Quick but dangerous

## State diagrams

- How do we characterize logic circuits?
  - Combinational circuits: Truth tables
  - Sequential circuits: State diagrams
- First draw the states
  - States ≡ Unique circuit configurations
- Second draw the transitions between states
  - Transitions ≡ Changes in state caused by inputs

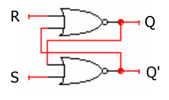
## Example: SR latch

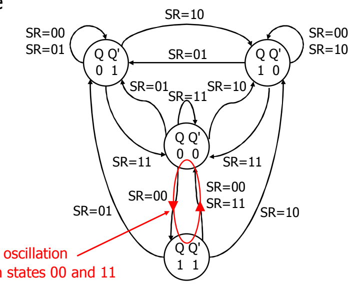
- Begin by drawing the states
  - States ≡ Unique circuit configurations
  - Possible values for feedback (Q, Q')



## Example: SR latch

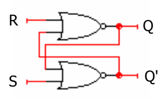
- Now draw the state transitions
  - Labeling inputs

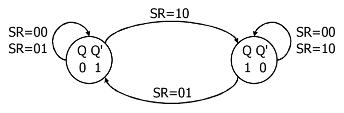




## Observed SR latch behavior

- The 1–1 state is transitory
  - Either R or S "gets ahead"
  - Latch settles to 0–1 or 1–0 state ambiguously
  - Race condition → non-deterministic transition
    - Disallow (R,S) = (1,1)





## System considerations

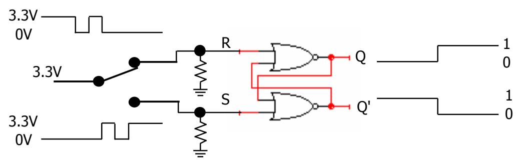
- Use edge-triggered flip-flops wherever possible
  - Avoid latches
  - Most common: Master-slave D
- Basic rules for correct timing
  - Clock flip-flops synchronously (all at the same time)
    - No flip-flop changes state more than once per clock cycle
    - FF propagation delay > hold time
  - Avoid mixing positive-edge triggered and negative-edge triggered flip-flops in the same circuit

## Asynchronous inputs

- Clocked circuits are synchronous
  - Circuit changes state only at clock edges
  - Signals (voltages) settle in-between clock edges
- Unclocked circuits or signals are asynchronous
  - No master clock
  - Real-world inputs (e.g. a keypress) are asynchronous
- Synchronous circuits have asynchronous inputs
  - Reset signal, memory wait, user input, etc.
  - Inputs "bounce"
  - Inputs can change at any time
    - We must synchronize the input to our clock
    - Inputs will violate flip-flop setup/hold times

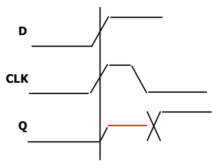
## Debouncing

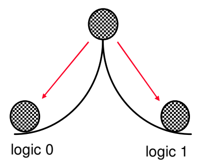
- Switch inputs bounce
  - i. e. don't make clean transitions
- Can use SR latch for debouncing
  - Eliminates dynamic hazards
  - "Cleans-up" inputs



## Synchronizer failure

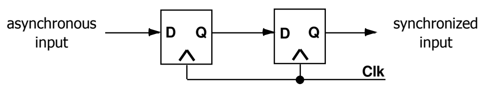
- Occurs when FF input changes near clock edge
  - Input is neither 1 or 0 when clock goes high
  - Output may be neither 0 or 1
    - May stay undefined for a long time
  - Undefined state is called metastability





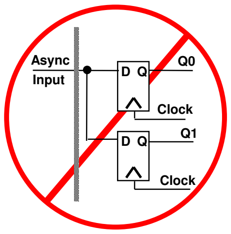
## Minimizing synchronizer failures

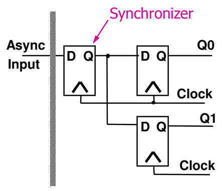
- Failure probability can never be 0
  - Can make it small
  - Cascade two (or more) flip-flops
    - Effectively synchronizes twice
    - Both would have to fail for system to fail



## Handling asynchronous inputs

- Never fan-out asynchronous inputs
  - Synchronize at circuit boundary
  - Fan-out synchronized signal


Async Input
D Q
Q0
Clock
D Q
Q1
Clock


Synchronizer
Async Input
D Q
D Q
Q0
Clock
D Q
Q1
Clock