Lecture 16

- Logistics
  - HW5 out, due next wednesday
- Last lecture
  - Finished combinational logic
  - Introduction to sequential logic and systems
- Today
  - Memory storage elements
    - Latches
    - Flip-flops
  - State Diagrams

Example from last time

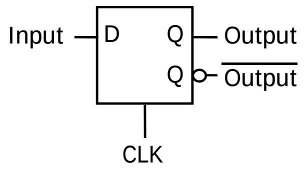
- Door combination lock
  - Enter three numbers in sequence and the door opens
  - When one number is entered, press ‘enter’
  - If there is an error the lock must be reset
  - After the door opens the lock must be reset
  - Inputs: Sequence of numbers, reset, enter
  - Outputs: Door open/close
  - Memory: Must remember the combination
  - Memory: Must remember which state we are in

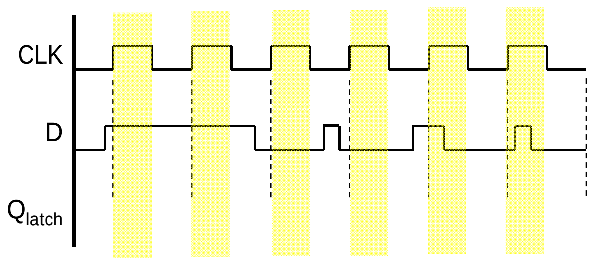
The “WHY” slide

- Memory storage elements
  - In order to do fun problems like door combination lock, we must know the building blocks (like how you had to learn AND and OR before you could do functional things). Be patient --- once you know these elements, you can build a lot of meaningful functions
- State diagrams
  - For combinational logic, truth table was an invaluable visualization tool for a function. For sequential logic, state diagram serves as a way to visualize a function.

The D latch

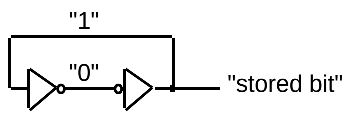
- Output depends on clock
  - Clock high: Input passes to output
  - Clock low: Latch holds its output
- Latch are level sensitive and transparent



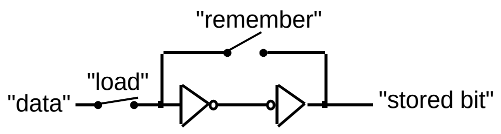


How do we store info like the latch?

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

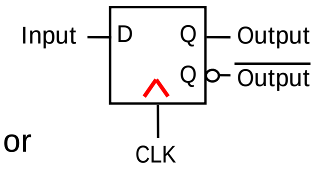


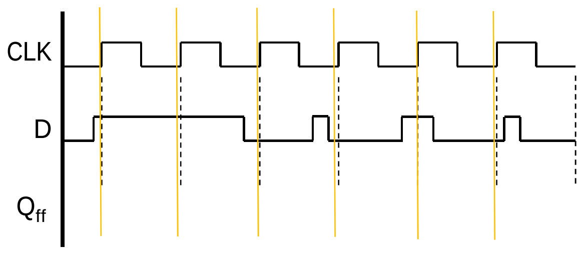
- Storing a new memory
  - Temporarily break the feedback path



The D flip-flop

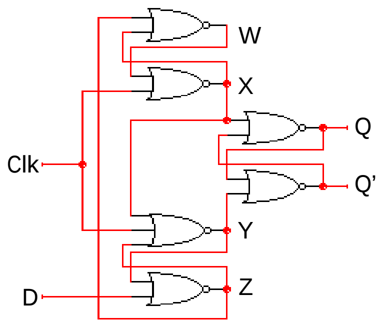
- Input sampled at clock edge
  - Rising edge: Input passes to output
  - Otherwise: Flip-flop holds its output
- Flip-flops are rising-edge triggered or falling-edge triggered





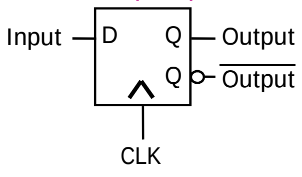
How do we make a D flip flop?

- Edge triggering is difficult
  - You can do this at home:
    - Label the internal nodes
    - Draw a timing diagram
    - Start with Clk=1

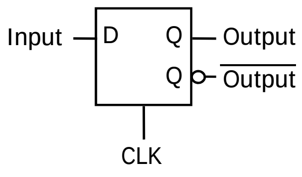


Terminology & notation

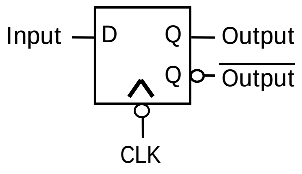
Rising-edge triggered D flip-flop



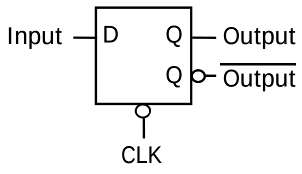
Positive D latch



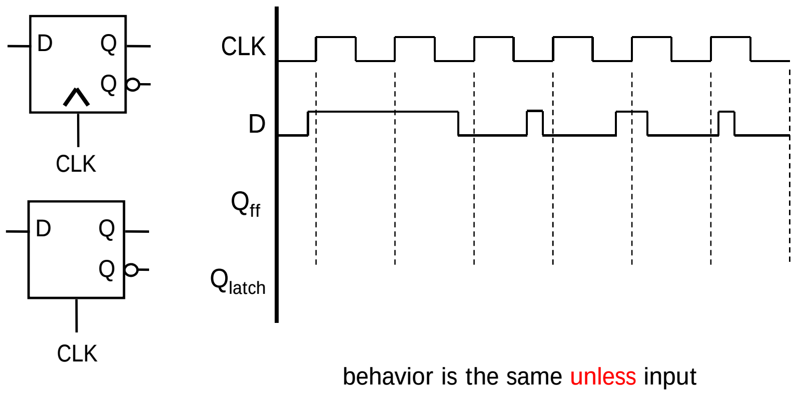
Falling-edge triggered D flip-flop



Negative D latch



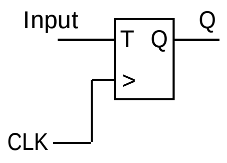
Latches versus flip-flops



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

T flip-flop

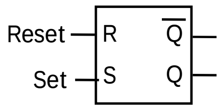
- Full name: Toggle flip-flop
- Output toggles when input is asserted
  - If T=1, then Q $\rightarrow$ Q' when CLK $\uparrow$
  - If T=0, then Q $\rightarrow$ Q when CLK $\uparrow$



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

The SR latch

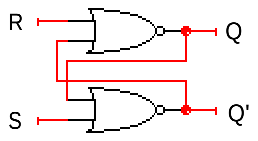
- Cross-coupled NOR gates
  - Can set (S=1, R=0) or reset (R=1, S=0) the output



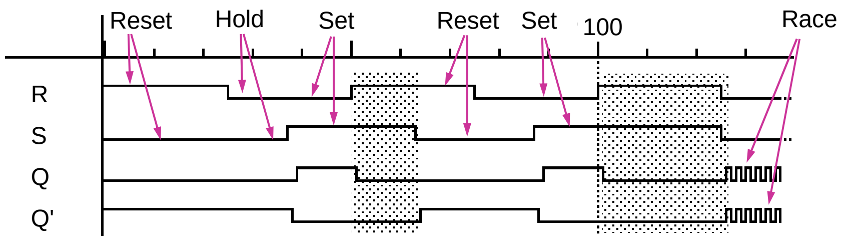
S	R	Q
0	0	hold
0	1	0
1	0	1
1	1	disallow

SR latch behavior

- Truth table and timing

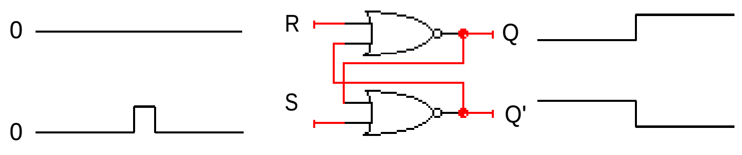


S	R	Q
0	0	hold
0	1	0
1	0	1
1	1	disallow

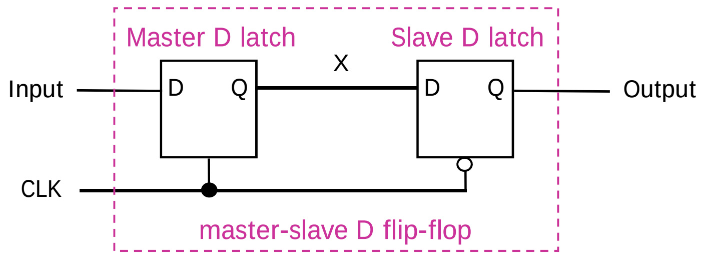


SR latch is glitch sensitive

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



The master-slave D



Clear and preset in flip-flops

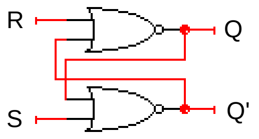
- Clear and Preset set flip-flop to a known state
  - Used at startup, reset
- Clear or Reset to a logic 0
  - Synchronous: Q=0 when next clock edge arrives
  - Asynchronous: Q=0 when reset is asserted
    - Doesn't wait for clock
    - Quick but dangerous
- Preset or Set the state to logic 1
  - Synchronous: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

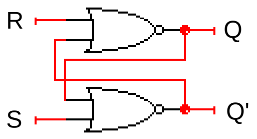
Example: SR latch

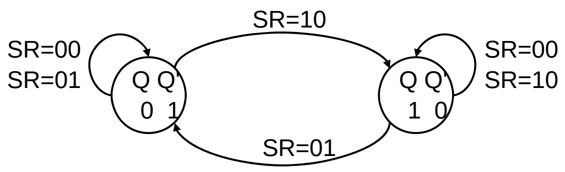
- Begin by drawing the states
  - States ≡ Unique circuit configurations
  - Possible values for feedback (Q, Q')



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)




SR=10
SR=00
SR=01
Q Q'
0 1
Q Q'
1 0
SR=00
SR=10
SR=01