Lecture 14

- Logistics
  - Midterm 1: Average 90/100. Well done!
  - Midterm solutions online
  - HW5 due date delayed until this Friday
- Last lecture
  - Finished combinational logic
  - Introduction to sequential logic and systems
- Today
  - Memory storage elements
    - Latches
    - Flip-flops
  - State Diagrams

Example from last time

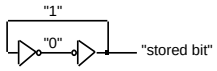
- Door combination lock
  - Enter three numbers in sequence and the door opens
  - As each number is entered, press 'new'
  - If there is an error the lock must be reset
  - After the door opens the lock must be reset
  - Inputs: Sequence of numbers, reset, new
  - Outputs: Door open/close
  - Memory: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

The "WHY" slide

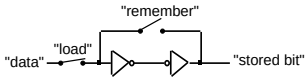
- Memory storage elements
  - In order to do fun problems like the door combination lock, we must know the building blocks (like how you had to learn AND and OR before you could do functional things). Be patient --- once you know these elements, you can build a lot of meaningful functions
- State diagrams
  - 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.

How do we store info? Feedback

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

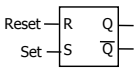


- Storing a new memory
  - Temporarily break the feedback path



The SR latch

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

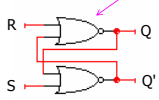


S	R	Q
0	0	hold
0	1	0
1	0	1
1	1	disallow

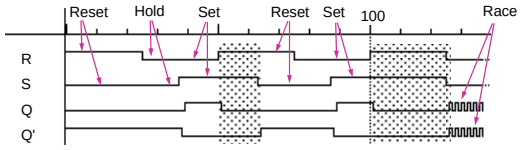
SR latch behavior

NOR output is 1
Only when both inputs are 0

- Truth table and timing

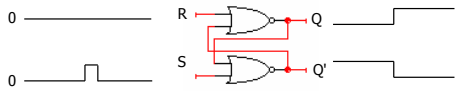


S	R	Q
0	0	hold
0	1	0
1	0	1
1	1	disallow



SR latch is glitch sensitive

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



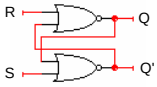
State diagrams

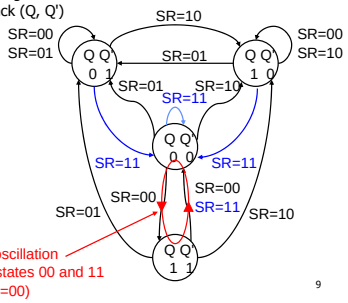
- How do we characterize logic circuits?
  - Combinational circuits: Truth tables
  - Sequential circuits: State diagrams
- First draw the states
  - States ≡ Unique circuit configurations
- Second draw the transitions between states
  - Transitions ≡ Changes in state caused by inputs

Example: SR latch

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

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

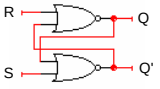


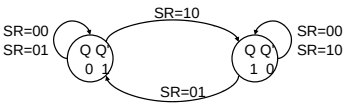


possible oscillation between states 00 and 11 (when SR=00)

Observed SR latch behavior

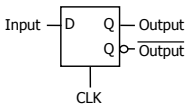
- The 1–1 state is transitory
  - Either R or S "gets ahead"
  - Latch settles to 0–1 or 1–0 state ambiguously
  - Race condition → non-deterministic transition
    - Disallow (R,S) = (1,1)

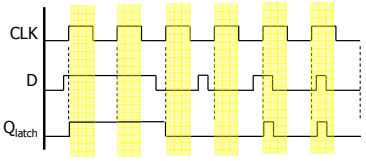




The D latch: store it and look it up

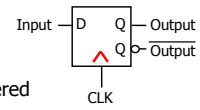
- Output depends on clock
  - Clock high: Input passes to output
  - Clock low: Latch holds its output
- Latches are level sensitive and "transparent"

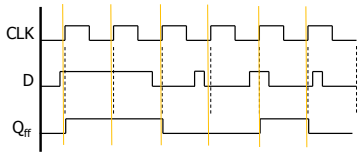




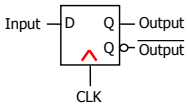
The D flip-flop

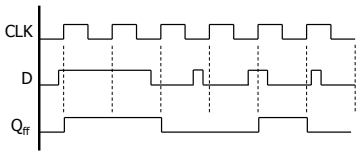
- Input sampled at clock edge
  - Rising edge: Input passes to output
  - Otherwise: Flip-flop holds its output
- Flip-flops can be rising-edge triggered or falling-edge triggered



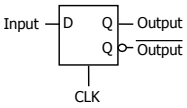


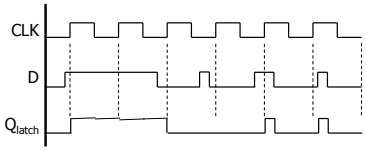
The D flip-flop





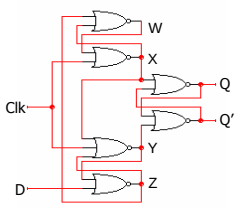
The D latch





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

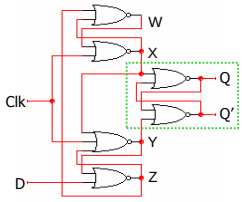


How do we make a D flip flop?

Falling edge-triggered flip-flop

If **Clk=1** then **X=Y=0** and SR-latch block holds previous values of **Q,Q'** also **Z=D'** and **W=Z'=D**

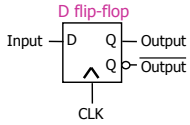
When **Clk→0** then **Y** (set for SR-latch block) becomes **Z'=D** and **X** (reset for SR-latch block) becomes **W'=D'** so **Q** becomes **D**
This is stable until **D** or the **Clk** switches



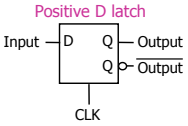
While **Clk=0**, if **D** switches then **Z** becomes **0** and **X** and **W** hold their previous values and **Y=X'=D** as before.

Terminology & notation

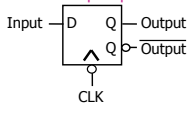
Rising-edge triggered D flip-flop



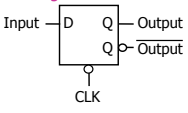
Positive D latch



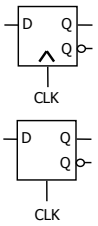
Falling-edge triggered D flip-flop

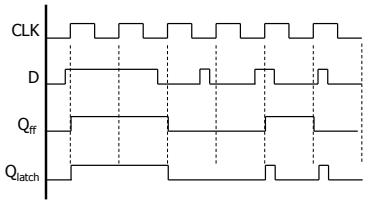


Negative D latch



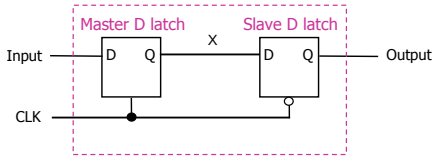
Latches versus flip-flops





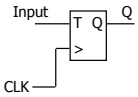
behavior is the same unless input changes while the clock is high

The master-slave D



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

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