Logic Design IV

CSCI 2021: Machine Architecture and Organization

Antonia Zhai
Department Computer Science and Engineering
University of Minnesota
http://www.cs.umn.edu/~zhai

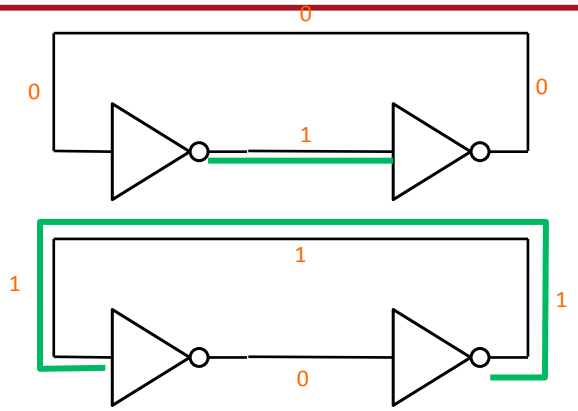
With Slides from Stephen McCamant and Wei-Chung Hsu

UNIVERSITY OF MINNESOTA

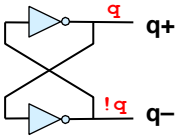
Sequential Logic

- The combinational logic circuits we have been studying so far have no memory. The outputs always follow the inputs.
- There is a need for circuits with memory, which behave differently depending upon their previous state.
- An example is a vending machine, which must remember how many and what kinds of coins have been inserted. The machine should behave according to not only the current coin inserted, but also upon how many and what kinds of coins have been inserted previously.
- These are referred to as *finite state machines*, because they can have at most a finite number of states.

Paired Inverters



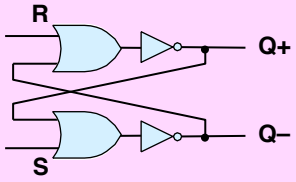
- Good: maintains a particular state
- Bad: no way to set



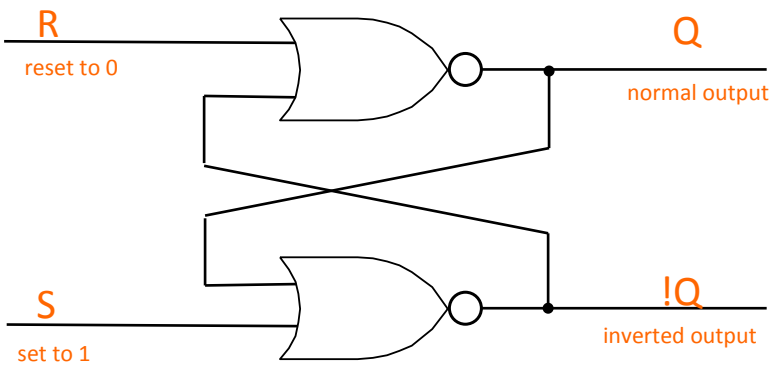
q = 0 or 1

1-Bit Latch

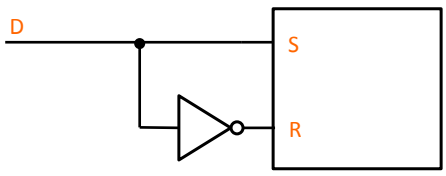
R	S
0	0
0	1
1	0
1	1



S-R Latch



S-R to D



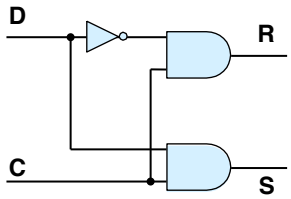
- D = 1: S = 1, R = 0
- D = 0: S = 0, R = 1
- Avoids ever having S and R together

Coordination and Clock Signals

- Transparency:
  - A device is transparent if input changes immediately propagate to the output
  - S-R latch is an example
  - Undesirable in many situations
    - E.g., remember Y86 pipeline stages
- Standard approach: clock signal
  - Alternates between 0 and 1
  - Same signal used throughout circuit
    - Challenge in high-speed designs: propagation speed
  - Rate controls speed of entire circuit
  - Design circuit to allow highest possible clock speed
  - Example: 3.0 GHz CPU
- Use clock to control when sequential devices "read"

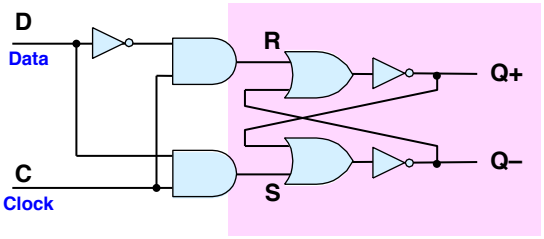
1 Bit Latch (cont.)

C	D	R	S
0	0		
0	1		
1	0		
1	1		

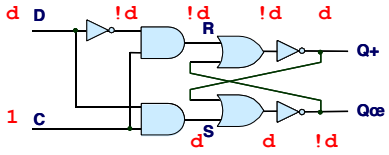


1 Bit Latch (cont.)

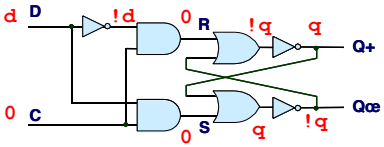
D Latch



Latching

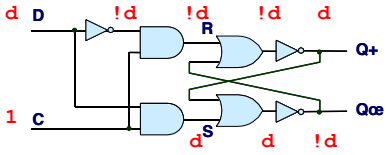


Storing

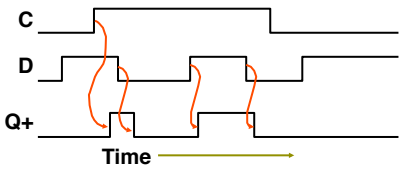


Transparent 1-Bit Latch

Latching



Changing D

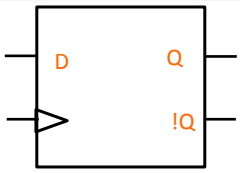


- When in latching mode, combinational propogation from D to Q+ and Q–
- Value latched depends on value of D as C falls

Edge-triggered Devices

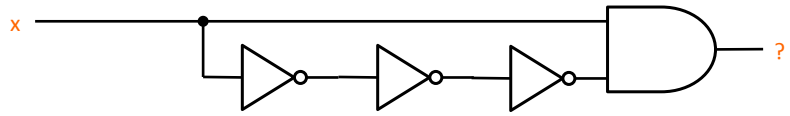
- Idea: update only on a clock "edge":
  - Positive/rising edge: 0 to 1
  - Negative/falling edge: 1 to 0
  - One update per clock cycle
- An edge-triggered bit-storage device is a "flip-flop"
- Flip-flops in series:
  - Previous output changes only after next input is "read"
  - Leads to lock-step propagation, one flip-flop per cycle

D Flip-Flop



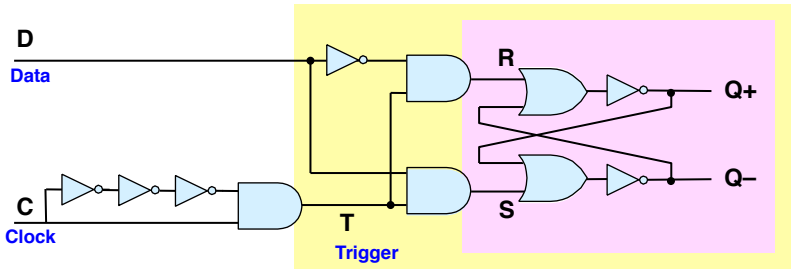
- Triangle indicates clock input
  - No bubble → rising edge triggered
- On edge, store the value of D ("data")
- This was our main building block for Y86 registers
- !Q is often unused, but available for free

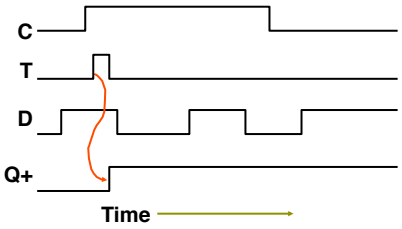
Transient timing



- What does this circuit do?
- Functional perspective:
  - (x & !!!x) = (x & !x) = 0, useless?
- Actually, rising edge of x causes a brief output pulse
  - Fast path goes to 1 before delayed path goes to 0

Edge-Triggered Latch





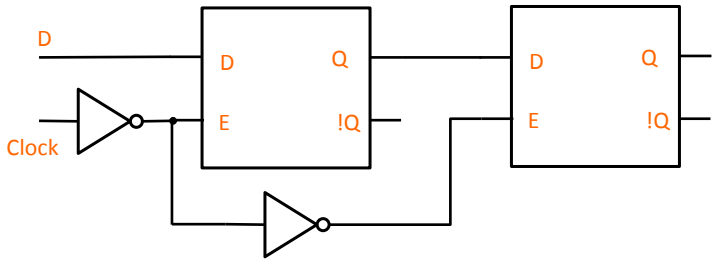
Only in latching mode for brief period

- Rising clock edge

Value latched depends on data as clock rises

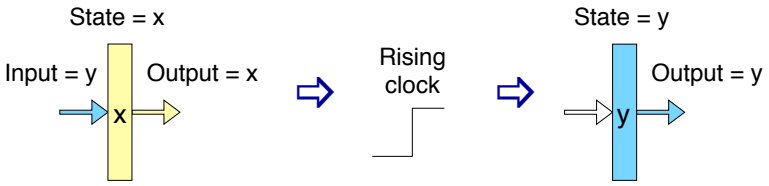
Output remains stable at all other times

Master-slave D flip-flop



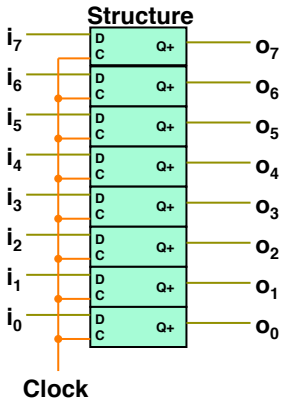
- Make flip-flop out of two gated latches
- Updates only on rising clock edge
  - Master freezes first
  - Then slave is enabled

Register Operation



- Stores data bits
- For most of time acts as barrier between input and output
- As clock rises, loads input

Register



- Stores word of data
- Collection of edge-triggered latches
- Loads input on rising edge of clock

Summary

- Storage
  - Latches
  - Flip-flops
  - Edge-triggered flip-flops
  - Registers