CS61C: Great Ideas in Computer Architecture (aka Machine Structures)

Lecture 11: FSMs, Synchronous Digital Systems

Instructors: Rosalie Fang, **Charles Hong**, Jero Wang

Announcements

- Labs 3 and 4 due today 7/11
- Homework 3 due tomorrow (Wednesday 7/12)
- Midterm review sessions:
    - Today, 5-7pm in the Woz (Soda 430)
    - Thursday, 3-5pm in Cory 540AB

Last Time: Combinational Logic

- Logic gates can be built out of transistors
- More complicated circuits can be built out of logic gates
- Logic gates and circuits can be represented with:
  - Circuit diagrams
  - Truth tables
  - Boolean algebra expressions
- Manipulating Boolean algebra expressions
  - Laws of Boolean algebra
  - Sum-of-products representation
- Applications of circuits
  - Arithmetic logic unit (ALU)
- Next: How do we store values in circuits?

CS 61C Hardware Roadmap

Processor (CPU)

- Datapath
- Control Logic
- Pipelining

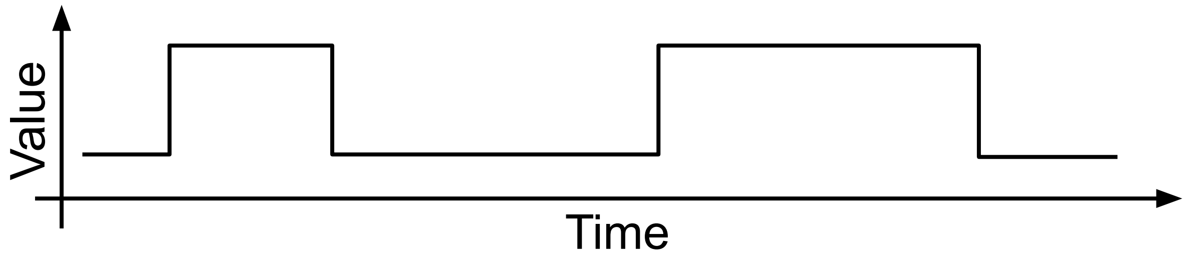
Synchronous Digital Systems

- Combinational Logic
- **Sequential Logic**

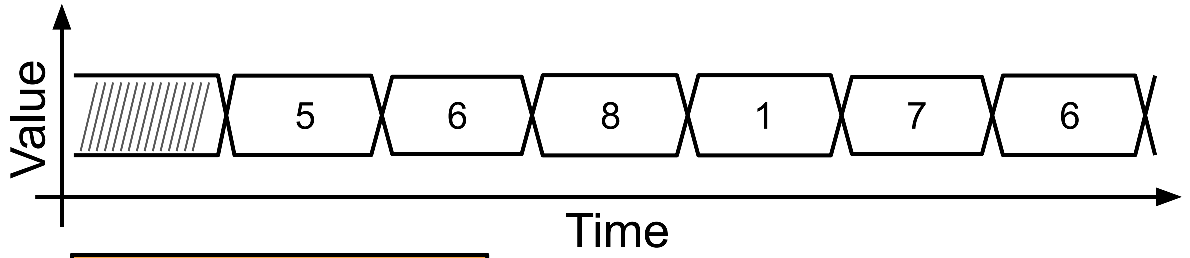
Transistors

Waveforms

- Graph of a signal's value over time



For 1-bit signals, high=1, low=0.

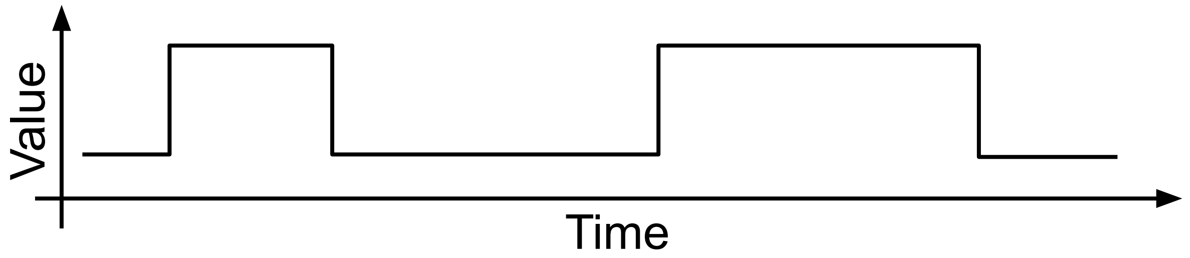


For *n*-bit signals, we can write the value on the waveform.

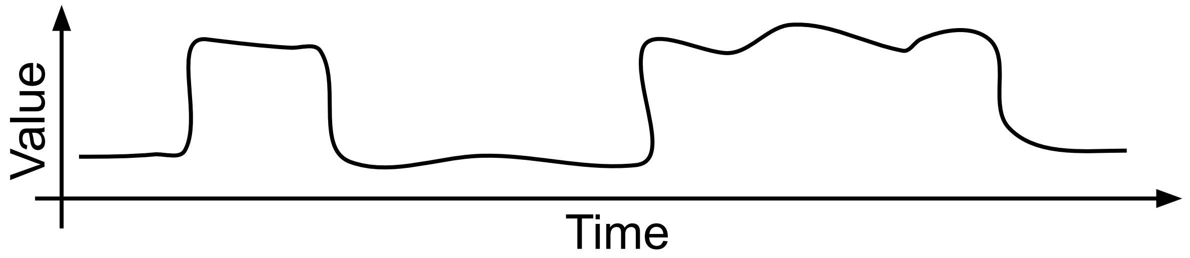
Grayed out = unknown or garbage value.

Waveforms

- Graph of a signal’s value over time



An ideal signal looks like this, but…

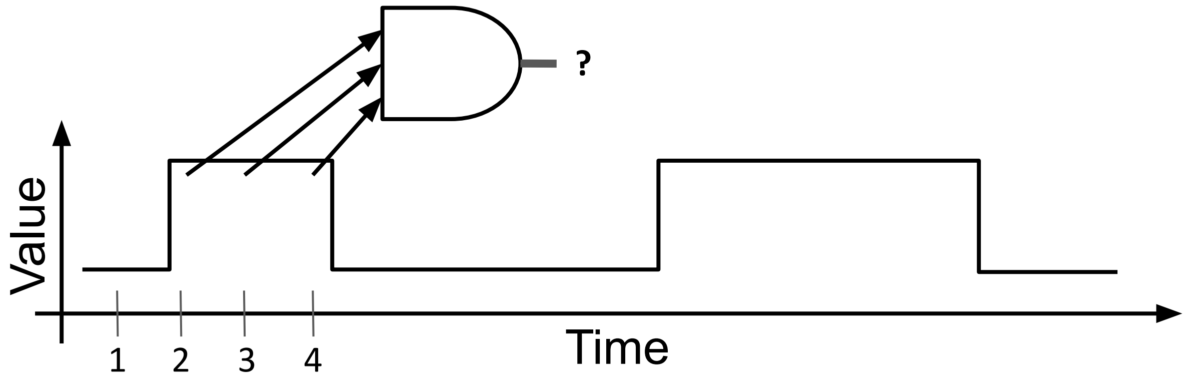


…in reality, signals are noisy!

Finite State Machines (FSMs)

Motivating FSMs

- We can AND signals together
- What if we want to AND three time steps, of one signal?
    - Set an output to 1, if input was 1 for the last three time steps

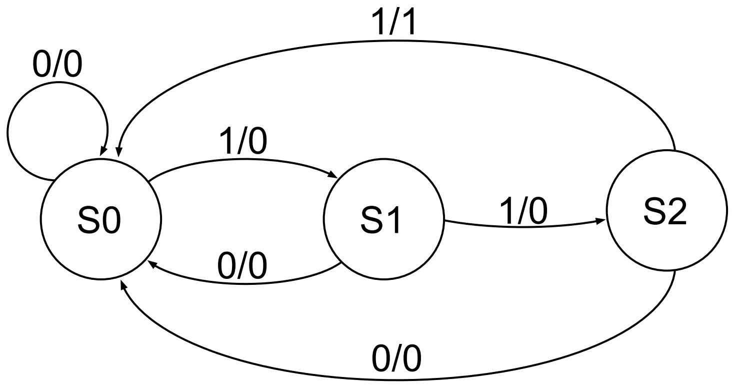


Finite State Machines (FSMs)

- An FSM consists of:
  - A set of states
  - A transition function: f(current state, input) → next state, output
- To run an FSM, repeat these steps:
  - Receive an input
  - Based on current state and input, move to the next state and generate an output

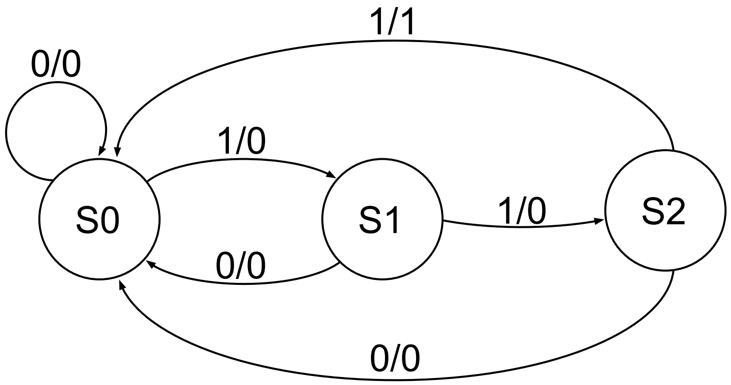
FSM: State Transition Diagram

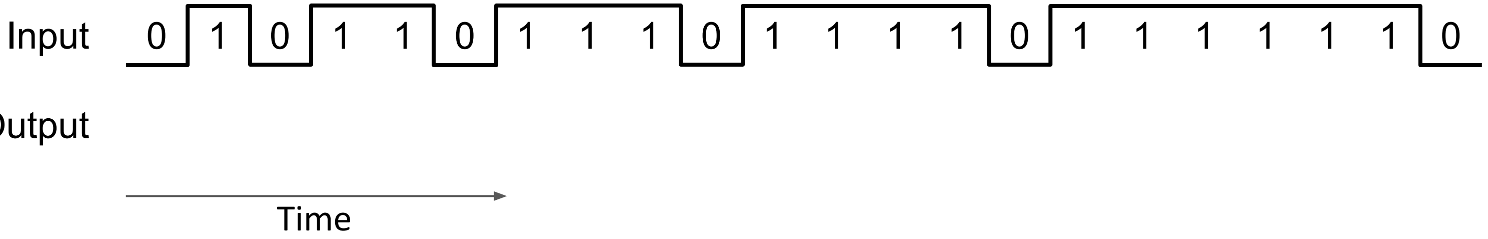
- S0, S1, S2 are the states
- Each arrow represents a transition
  - Arrow from current state to next state
  - Left label on arrow is the input
  - Right label on arrow is the output
- Example of arrow transition:
  - Arrow from S0 to S1, labeled 1/0
  - If you're at state S0 and receive input 1, then move to state S1, and output 0
- Example of arrow transition:
  - Arrow from S2 to S0, labeled 1/1
  - If you're at state S2 and receive input 1, then move to state S0, and output 1



FSM: Waveform

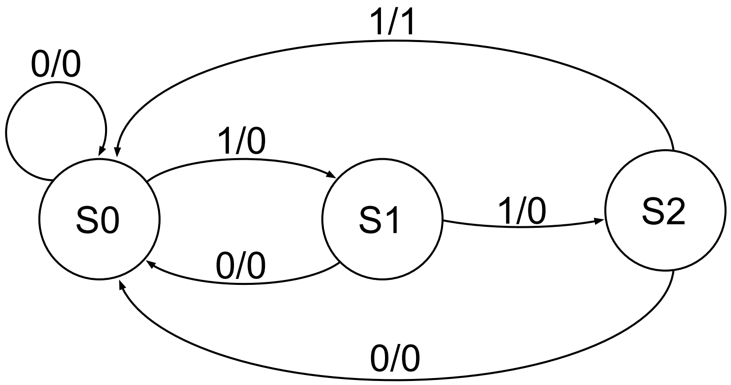
Given this state transition diagram and input signal, what is the output signal?

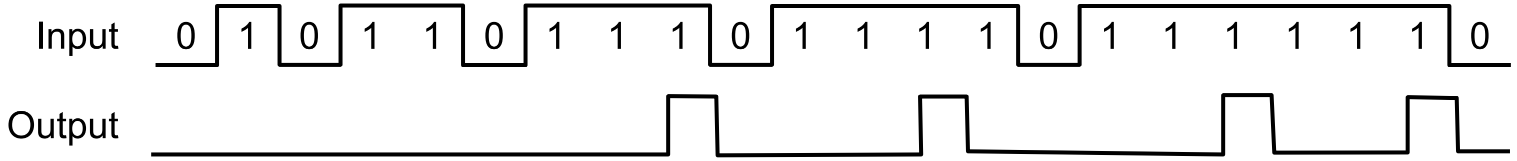




FSM: Waveform

What pattern in the input is this FSM detecting?

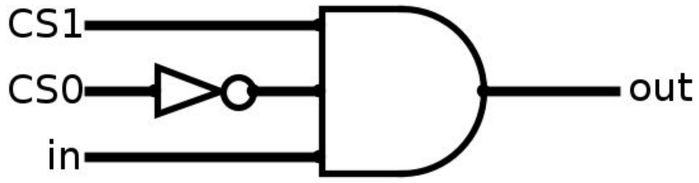




FSM: Transition Function Circuit

- Transition function is combinational logic with:
  - Two inputs: current state (CS) and input (in)
  - Two outputs: next state (NS) and output (out)
  - Each state is labeled with a unique binary number

CS	in	NS	out
00	0	00	0
00	1	01	0
01	0	00	0
01	1	10	0
10	0	00	0
10	1	00	1

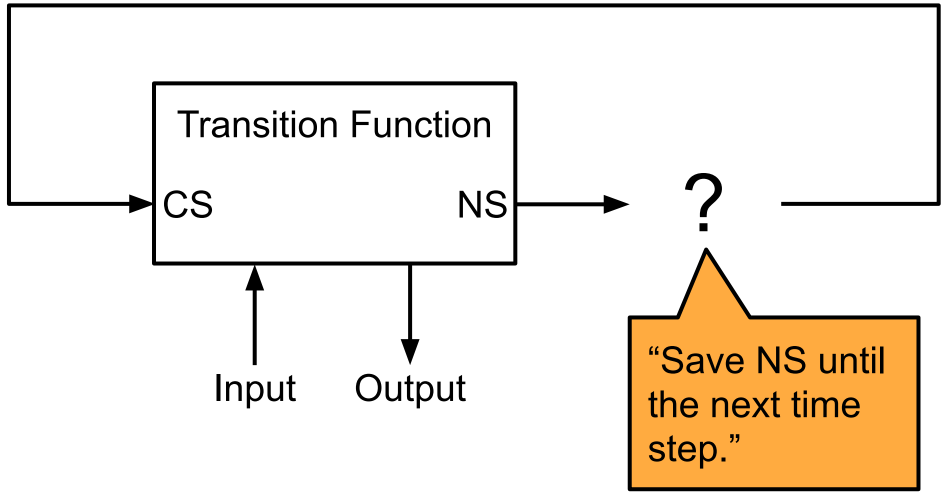


You can also derive a circuit and Boolean algebra expression for the other two outputs, NS0 and NS1.

$$out = (CS1)(\neg CS0)(in)$$

FSM: State

How do we differentiate between timesteps?
How do we "save" which state we're in?



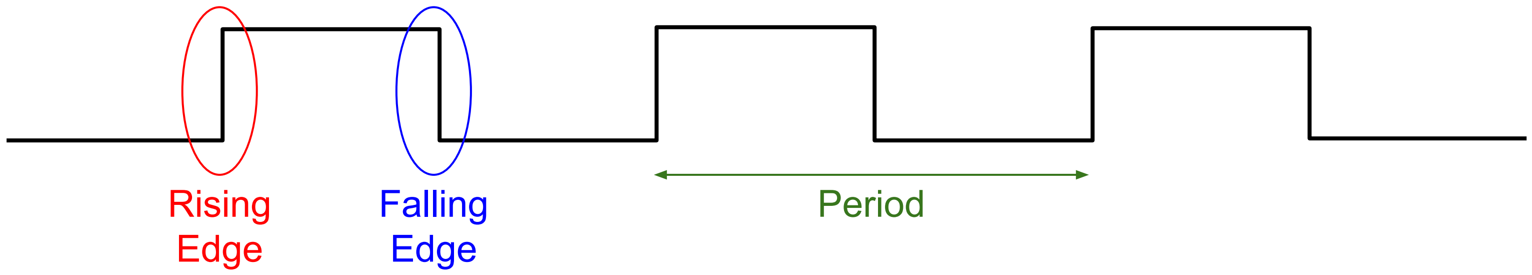
Sequential Logic: Clock and Registers

Synchronous Digital Systems

- What does it mean?
- Synchronous: behaves in time with each other, at the same time, simultaneously
- Digital: with regards to signals, takes on only discrete values
  - Different from analogue which takes on intermediate values
- Systems: a group of interdependent elements that form a whole
- Synchronous Digital System: elements that behave in time with one another as part of a greater whole, producing discrete signal outputs

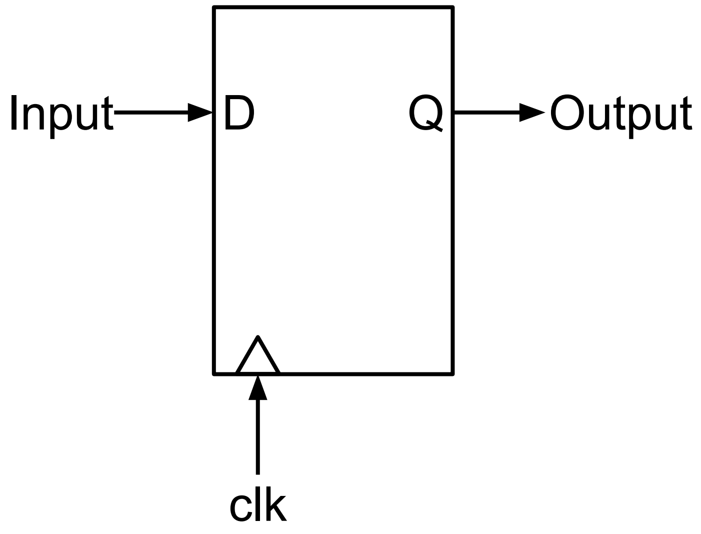
Clock

- Clock: Signal that alternates between 0 and 1
- Rising edge: Time when the clock switches from 0 to 1
- Falling edge: Time when the clock switches from 1 to 0
- Clock period: Time between rising edges
- Clock frequency: Number of rising edges per second
    - Frequency = 1 / period

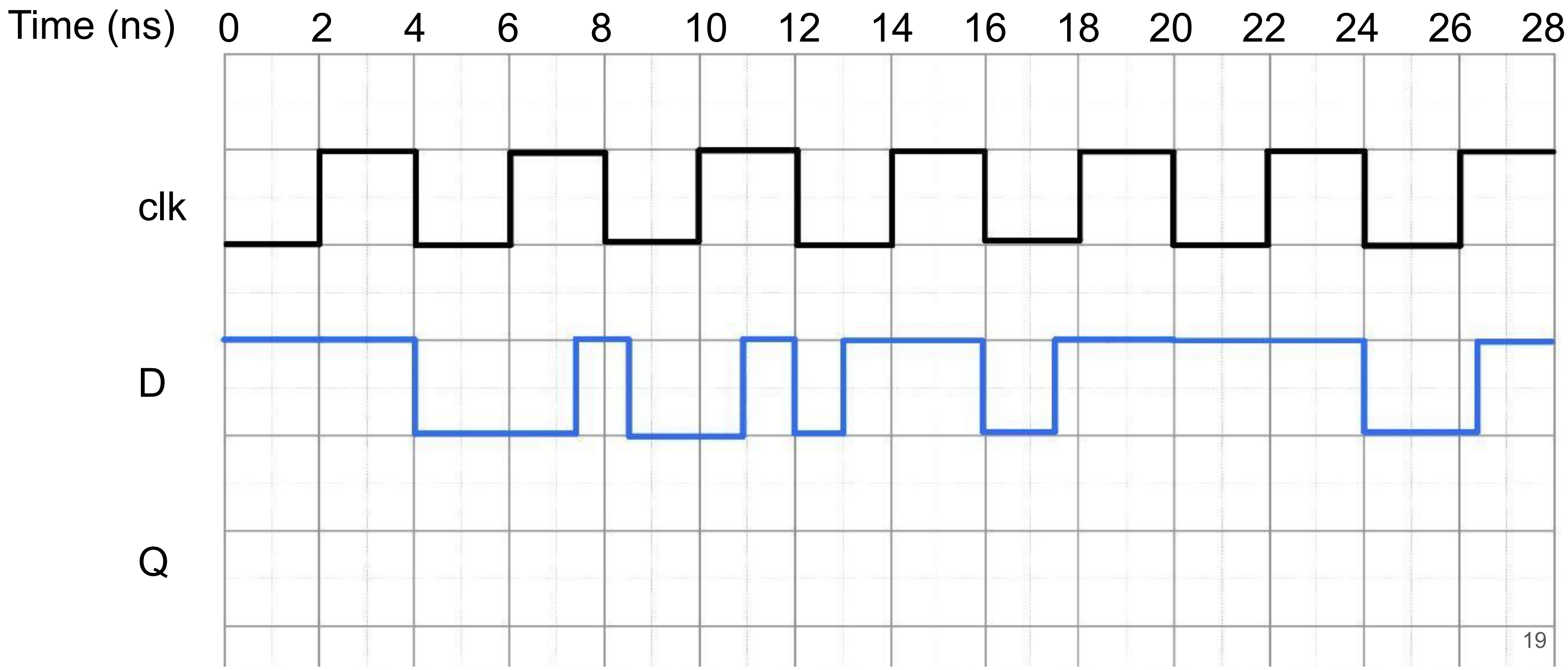


Flip-Flop

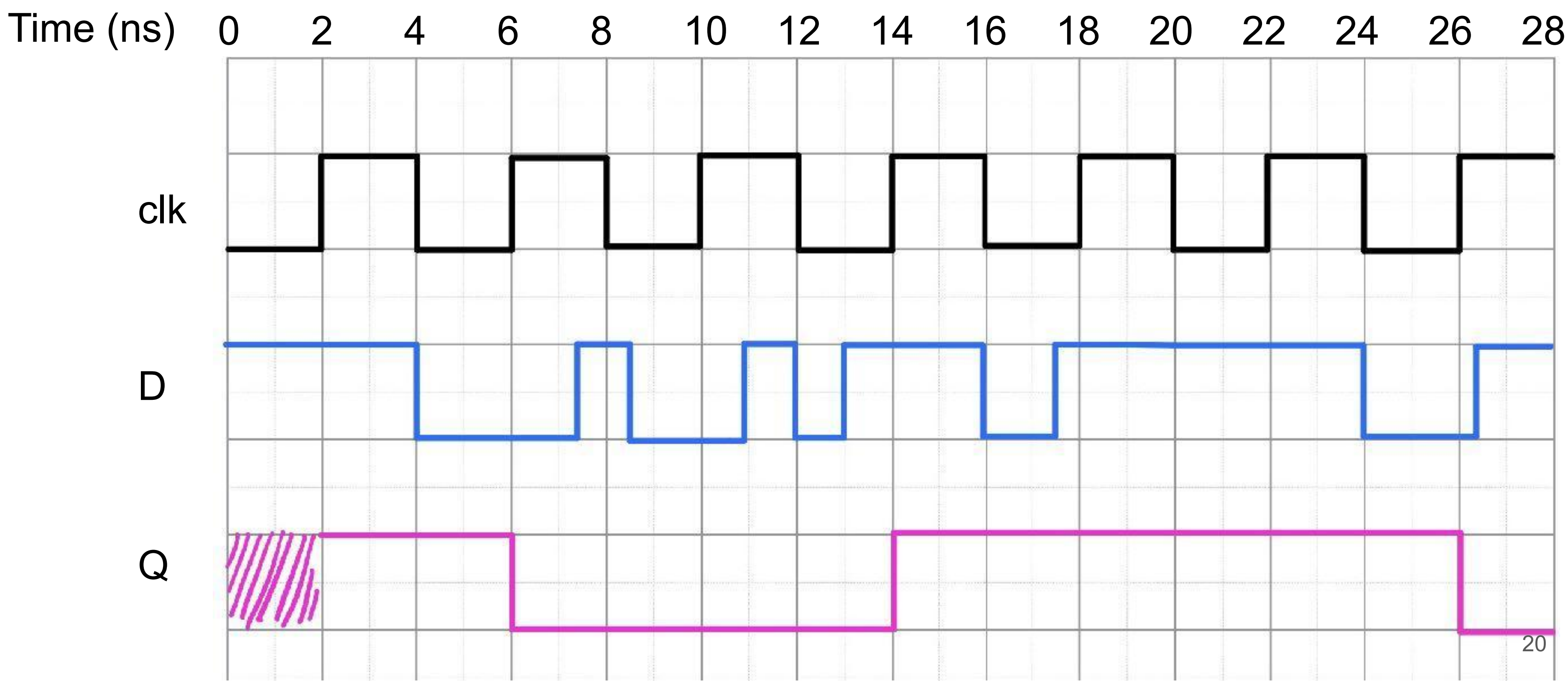
- Inputs:
  - One-bit value D
  - One-bit clock value (often drawn as triangle)
- Outputs:
  - One-bit value Q
- Behavior:
  - On the rising edge of the clock, set Q=D
  - At all other times, do nothing
- Usage:
  - Store data: Between rising edges, the output value at Q will stay steady
  - Control the flow of data: Hold data at the D input until the next rising edge



Flip-Flop Timing Example

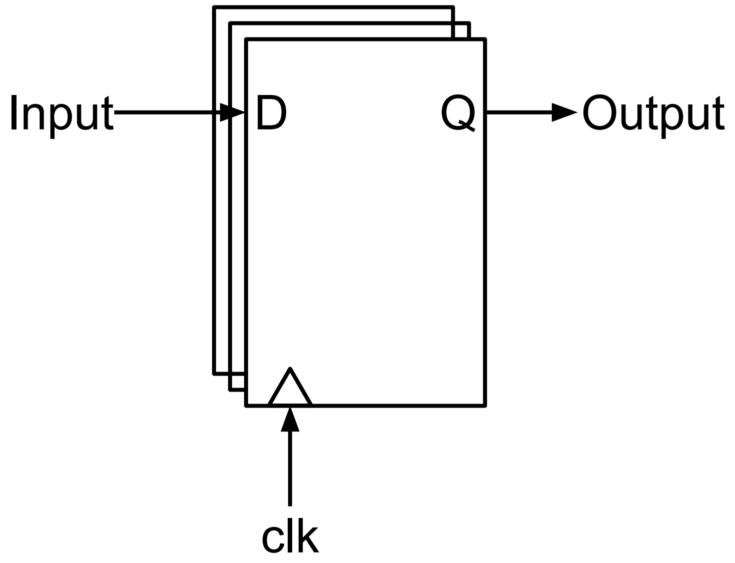


Flip-Flop Timing Example



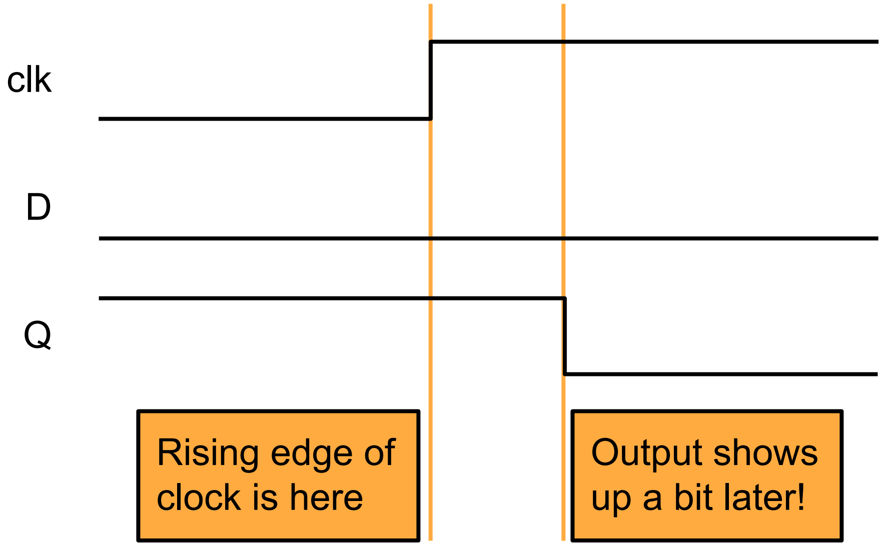
Registers

- *n*-bit register: *n* flip-flops bundled together
  - Each flip-flop stores 1 bit of data
  - The register stores *n* bits of data in total
- Inputs:
  - *n*-bit value D
  - One-bit clock value
- Outputs:
  - *n*-bit value Q
- Behavior:
  - On the rising edge of the clock, set Q=D
  - At all other times, do nothing

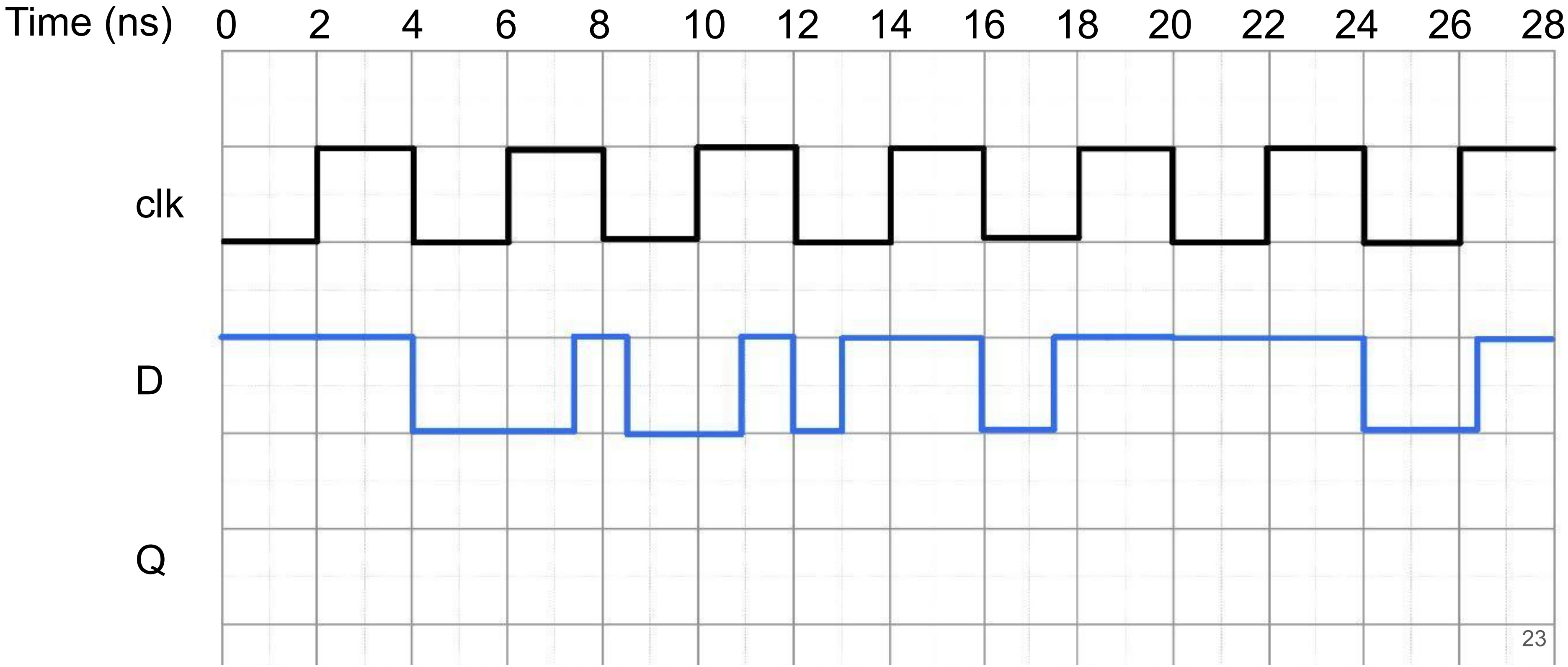


Register Delay: clk-to-q delay

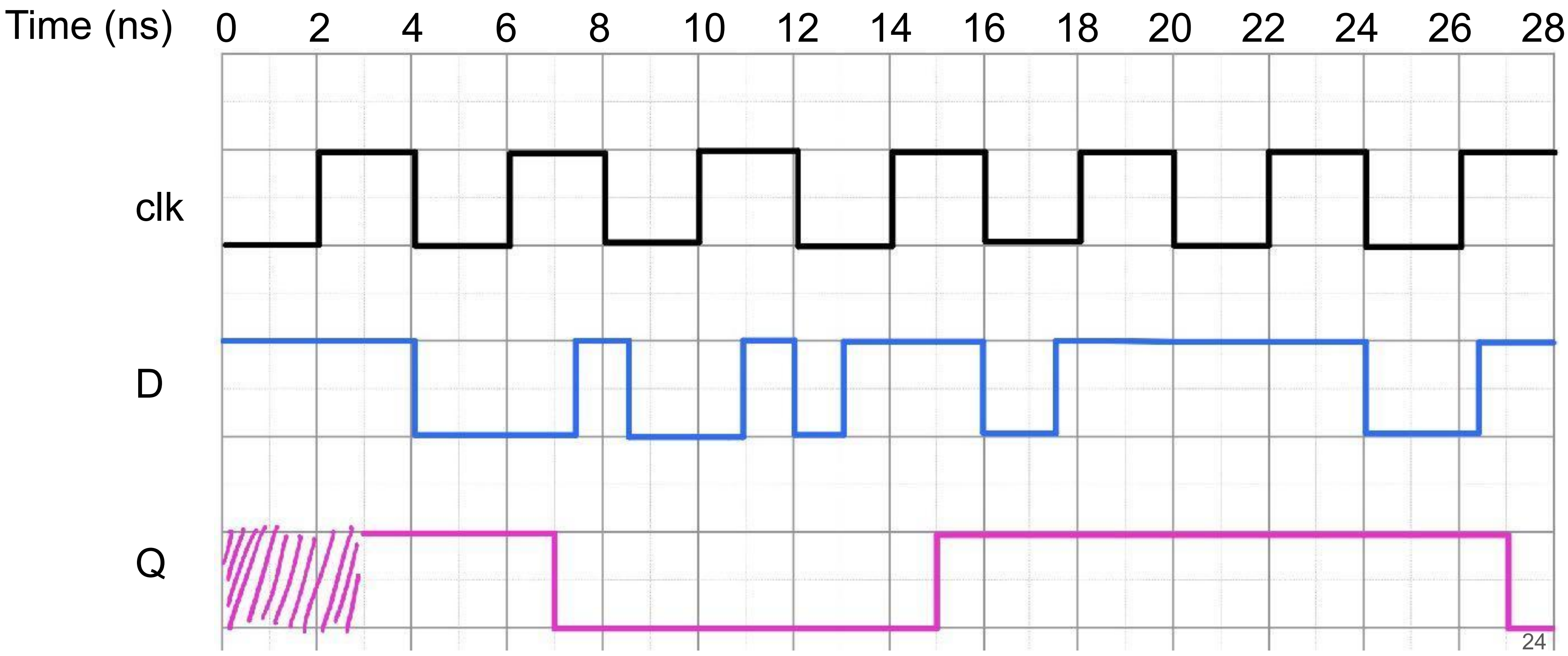
- Registers can’t transfer the D input to Q output instantly
- clk-to-q delay: Time it takes after the rising edge for the Q output to change



Flip-Flop Timing with 1 ns clk-to-q Delay



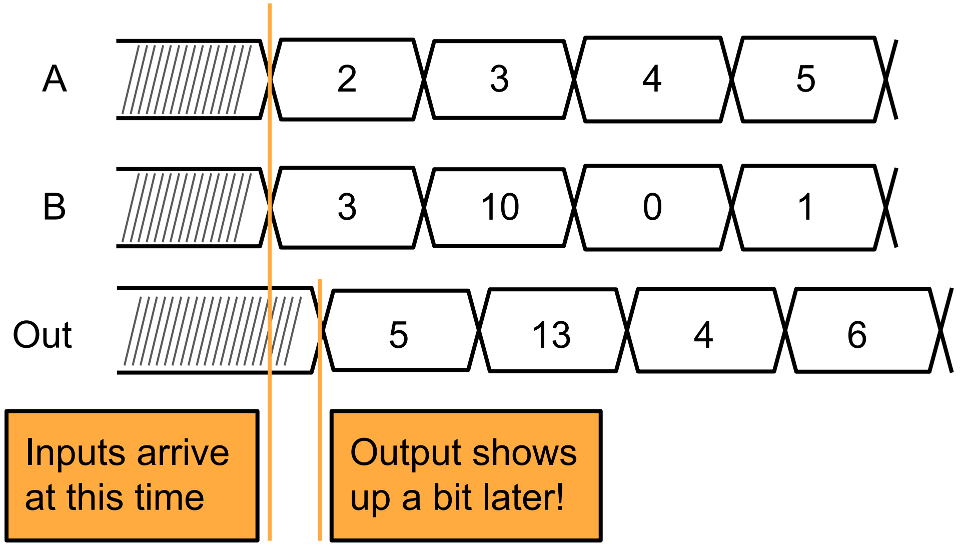
Flip-Flop Timing with 1 ns clk-to-q Delay



Combinational Delay

Combinational Delay

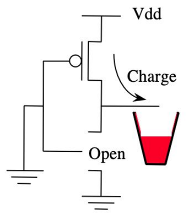
- Combinational logic circuits have propagation delay: time it takes for the output to change after the input changes
- Example: Adder circuit that sets Out = A + B:

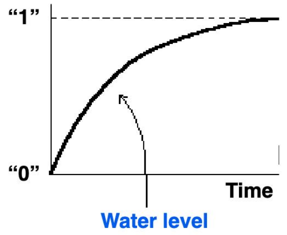


Why does combinational delay exist?

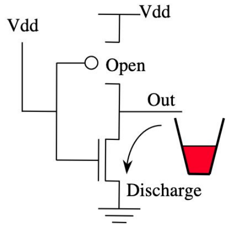
- Transistors are not perfect switches.
- If electrons are water molecules, transistor resistance like pipe diameters, and capacitors are buckets.

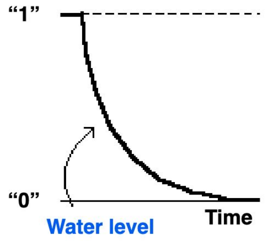
An "on" p-type transistor **fills up** the output capacitor with charge.





An "on" n-type transistor **drains** the bucket.



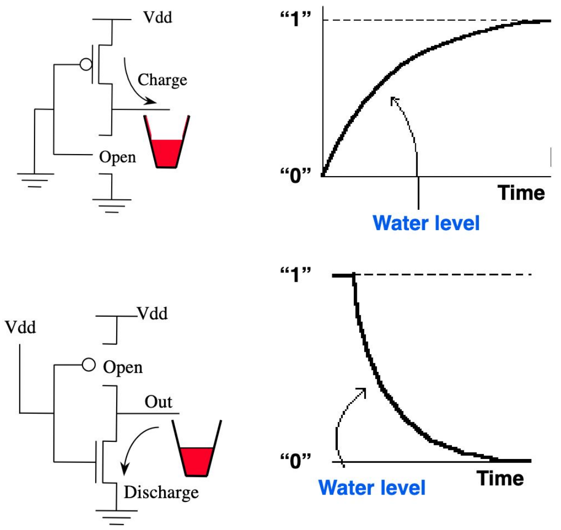


Why does combinational delay exist?

- Transistors are not perfect switches.
- If electrons are water molecules, transistor resistance like pipe diameters, and capacitors are buckets.

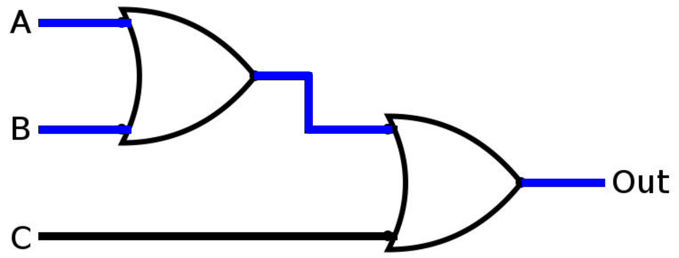
Side note: Filling and draining the "bucket" uses energy!

The main reason why your devices heat up when they're working hard.



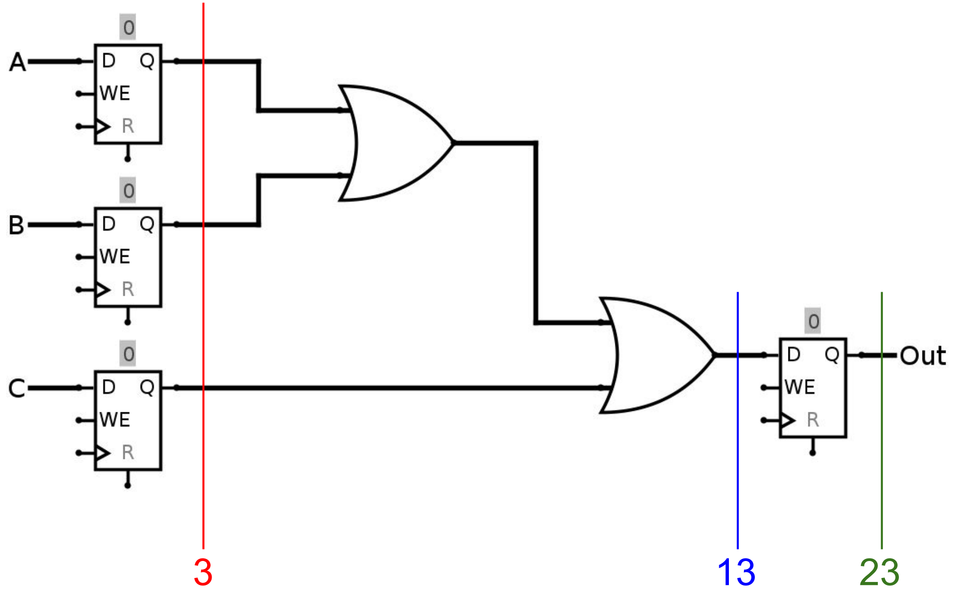
Combinational Delay

- Example
  - OR gates have a delay of 5 ps
  - Suppose A, B, C change at t=0 ps. When does the output change?
  - Wait 5 ps for A + B to be calculated
  - Then wait 5 ps for (A + B) + C to be calculated
- We have to consider the **critical path**: the path between input and output with the longest delay



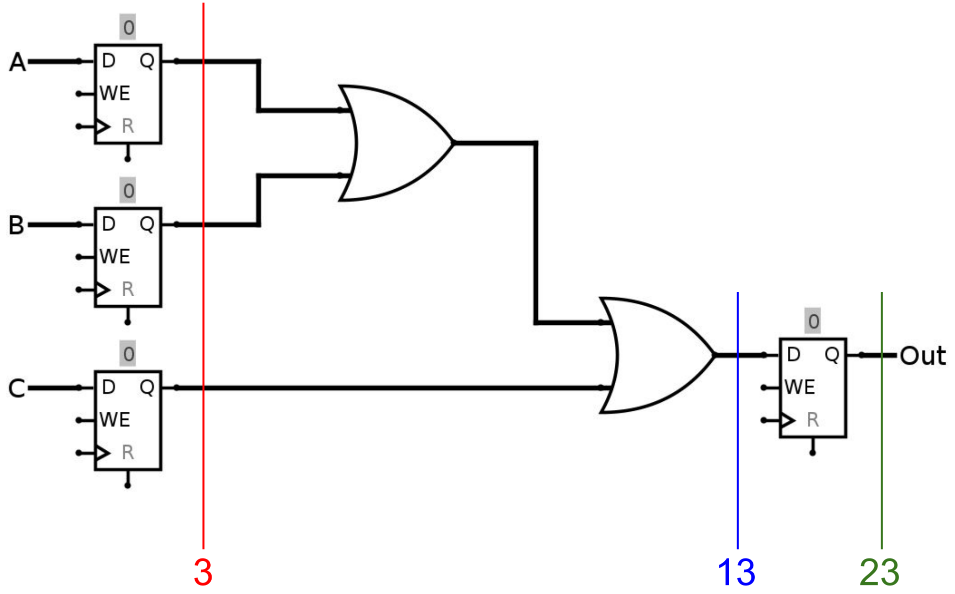
Three-way OR with registers

- Delay assumptions:
  - OR gate delay = 5 ps
  - clk-to-q delay = 3 ps
  - clock period = 20 ps
- Timing:
  - `t=0ps`: Rising edge of clock
  - `t=3ps`: Inputs appear at Q output of the left-side registers
  - `t=13ps`: Output appears at D input of the right-side register
  - `t=20ps`: Rising edge of clock
  - `t=23ps`: Output appears at Q output of the right-side register



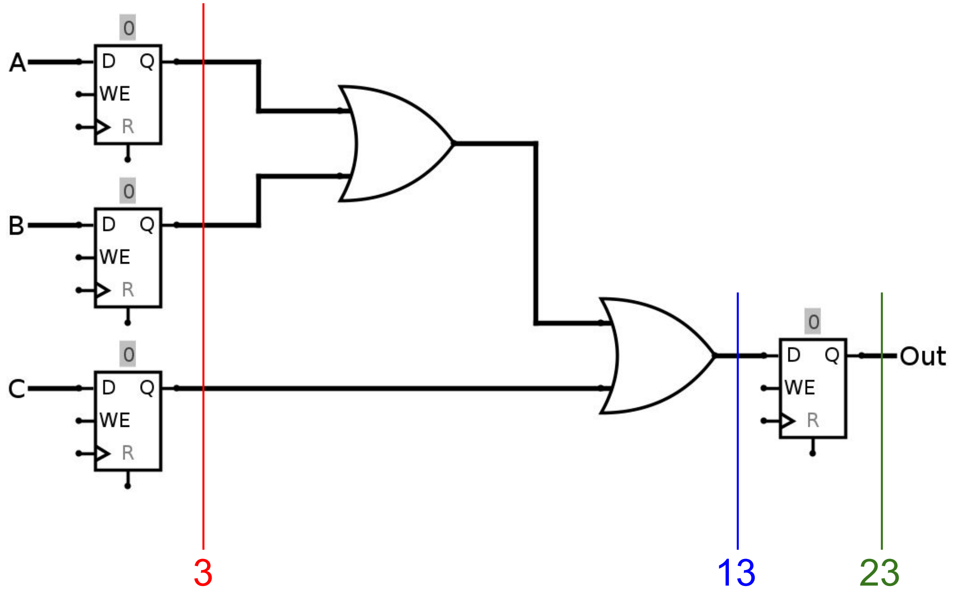
Maximizing Clock Frequency

- Problem
  - Computation was done at t=13ps
  - But we had to wait until the next rising edge (t=20ps), plus the clk-to-q delay (t=23ps), to access the result
- Solution: Reduce the time between rising edges
  - Shorter clock period, higher clock frequency
  - More computations per second



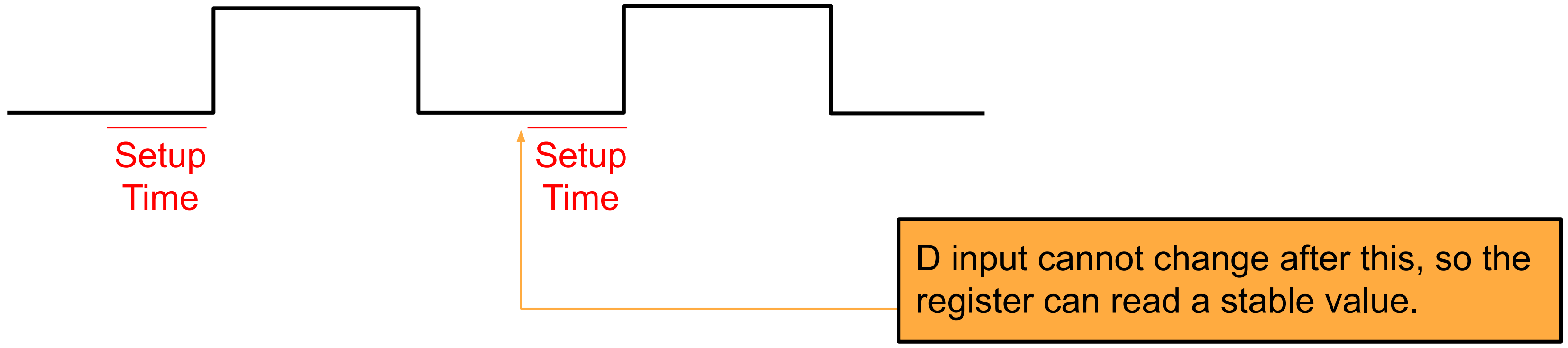
Maximizing Clock Frequency

- Can we increase clock frequency as much as we want?
  - Suppose we set the clock period to be 10 ps
  - Problem: Computation isn't finished on the next rising edge of the clock!
  - What is the maximum clock frequency that produces correct behavior?
  - Is it safe to have the rising edge at 13 ps, exactly when the computation is done?



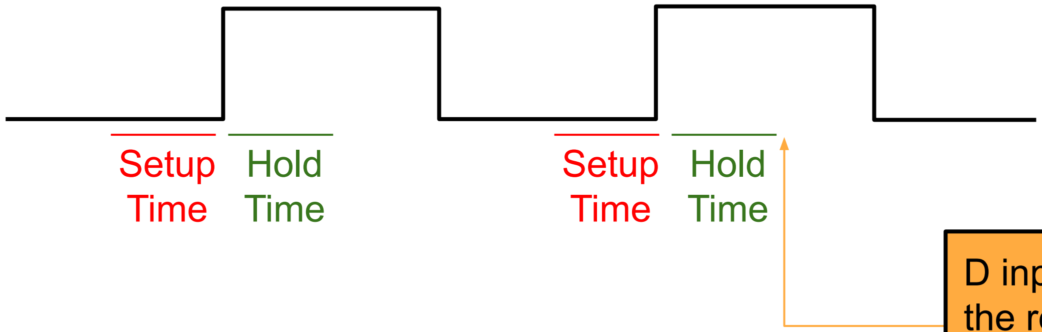
Register Constraints: Setup Time

- Intuition:
  - Assume clock period = 20ps
  - What if signal arrives at the D input of the right-side register at 19.9ps?
  - At the rising edge (t=20ps), the D input hasn't stabilized yet. Register doesn't know what to send to the Q output!
- Setup time: Time before rising edge when the D input must be stable


Setup Time
Setup Time
D input cannot change after this, so the register can read a stable value.

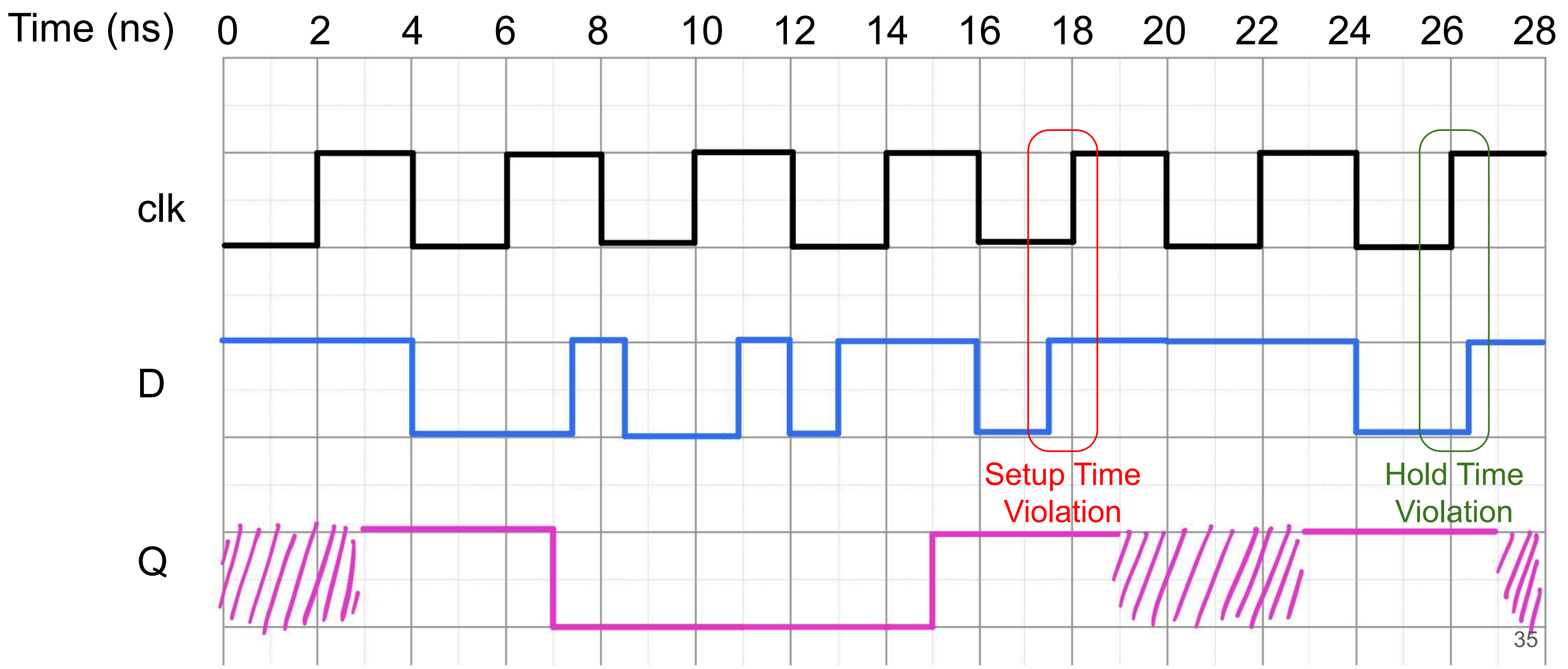
Register Constraints: Hold Time

- Intuition:
  - Assume clock period = 20ps
  - What if a new signal arrives at the D input of the right-side register at 20.1ps?
  - After the rising edge (t=20ps), the D input isn't stable. Register doesn't know what to send to the Q output!
- Setup time: Time after rising edge when the D input must be stable



D input cannot change before this, so the register can read a stable value.

Setup and Hold Time Violations



Maximizing Clock Frequency

- Setup time and combinational delay constraint:

$$\boxed{\text{Clock period} \geq \text{clk-to-q delay} + \text{longest combinational delay} + \text{setup time}}$$

  - After the rising edge, we have to wait clk-to-q delay for the Q output to change
  - Then, we have to wait the longest combinational delay for the result to appear at the D input
  - Then, we have to hold that D input stable for the setup time

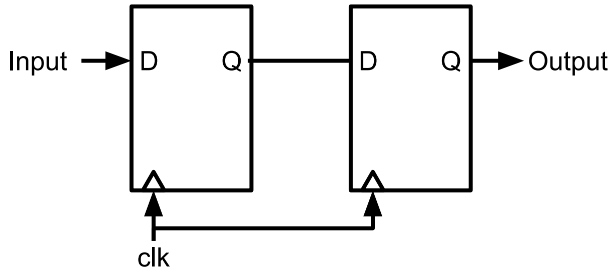
Satisfying Hold Time Constraints

- Hold time constraint:

$$\boxed{\text{Hold time} \leq \text{clk-to-q delay} + \text{shortest combinational delay}}$$

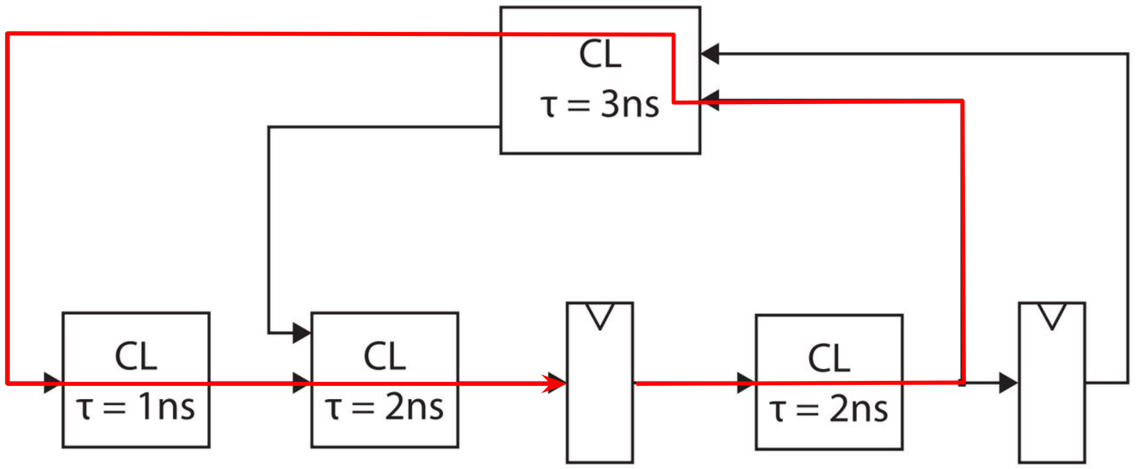
  - After the rising edge, we have to wait clk-to-q delay for the Q output to change
  - Then, after the shortest combinational delay, one of the D inputs will change
  - We need the hold time to be over by the time the D input changes

- What happens if you connect two registers in a row?



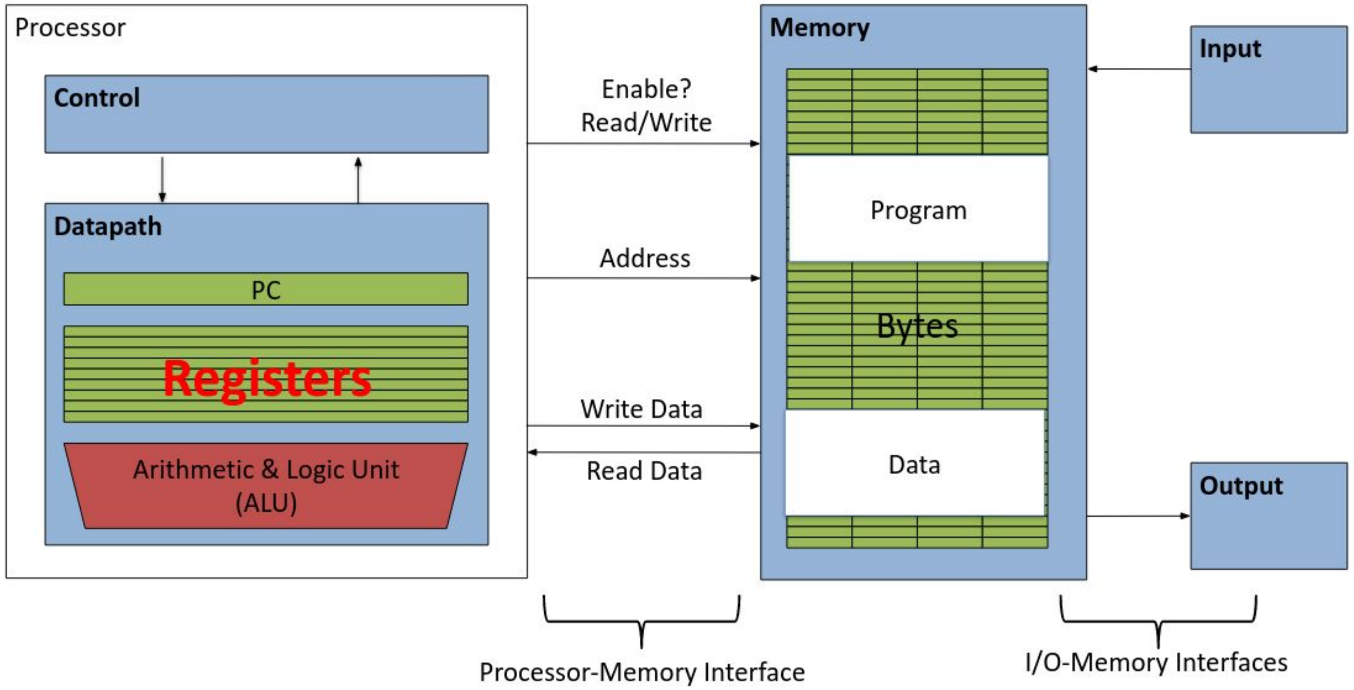
Finding Critical Paths

- **Critical path** can also be thought of as the “longest path between registers”
- To find: check all paths between two registers, or a register and itself



Memory

Recall: Registers are Inside the Processor (Memory is Not!)



Memory vs. Registers

- Memory is different from registers
  - Memory is in a different location on hardware
  - We don’t have space for 2^{32} registers on the processor!
- The memory unit is abstracted away for this class
  - Memory accesses are usually much slower than register accesses, but for now we’ll assume they’re about as fast as register accesses
  - In a couple weeks, we’ll see how we can achieve this with caches

Summary, Next Time (1/2)

- Finite State Machines (FSMs)
  - Set of states, with a transition function: f(current state, input) → next state, output
  - In a circuit: Transition function is combinational logic, current state stored in a register
- Sequential logic blocks
  - Waveforms show how signals change over time
  - Clock: Signal that alternates between 1 and 0. Rising edge used for timing a circuit.
  - Flip-flop: On the rising edge of the clock, set Q output = D input
  - Register: *n* flip-flops bundled together. Used to store values in the circuit and control signal flow.
- Register delays
  - clk-to-q delay: Time after the rising edge when Q output is updated with D input
  - Setup time: Time before the rising edge when the D input must be stable
  - Hold time: Time after the rising edge when the D input must be stable

Summary, Next Time (2/2)

- Combinational logic circuits have propagation delay
  - Calculate delay by looking for the critical path (longest path through the circuit)
- Memory
  - Interface between processor and memory for writing to and reading from memory addresses
- Next time: Use combinational and sequential logic blocks to build a circuit that runs RISC-V instructions (also known as: CPU)!