Digital Logic Design : Sequential Circuit

Z. Jerry Shi

Department of Computer Science and Engineering

University of Connecticut

Based on materials from Computer Organization: the Hardware/Software Interface by D. Patterson and J. Hennessy

Appendix A

- Clock
- Memory elements
  - D Flip-flops, registers, register file
- State machines and timing

G	M	K	unit	m	u	n
10^9	10^6	10^3	10^0	10^{-3}	10^{-6}	10^{-9}

Reading: Sections A.7 - A.11

Combinational versus sequential

Two types of circuit:

- Combinational circuit: the outputs depend on the current input values

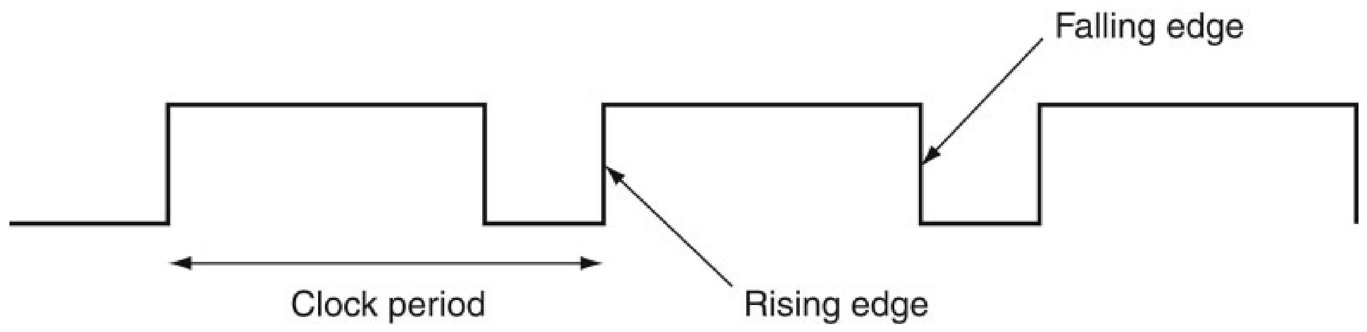
- Sequential circuit: the outputs also depend on the history of inputs
  - Two identical sequential circuits may produce different outputs even if their current inputs are the same

Clock

- A clock signal oscillates between high and low values
- The clock period is the time for one full cycle
  - Also called clock cycle time
  - The clock rate is the reciprocal of the cycle time

If the clock cycle time is 1 ns, the clock rate is 1 GHz.

If the clock rate is 2 GHz, the clock cycle time is 0.5ns.



Question

If a processor runs at 200 MHz, what is the clock cycle time in ns?

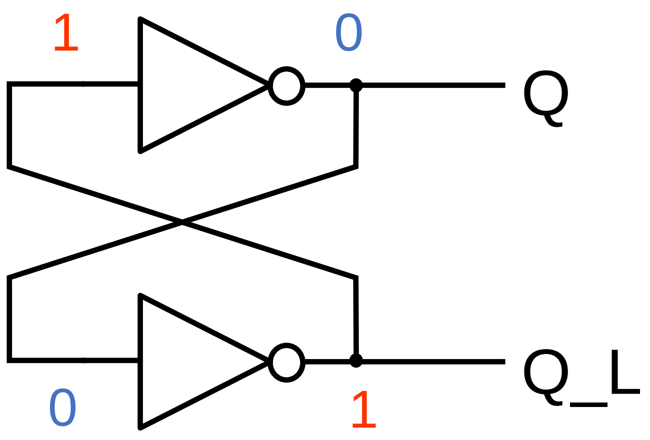
Round to the nearest integer.

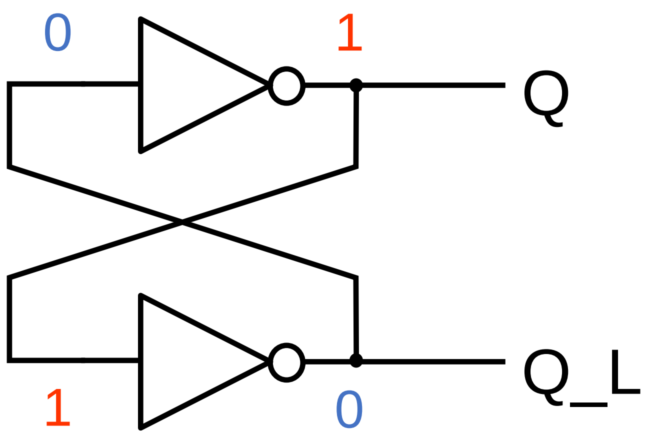
Why cannot the processor run at higher clock rates?

We are going to see why.

Bistable element

- A bistable element has two states: Q is 0 or 1
- The simplest sequential circuit to remember something
  - Need memory to remember history in sequential circuit

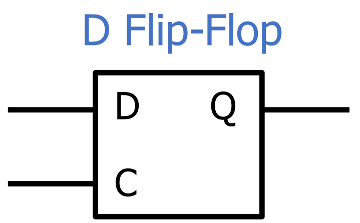


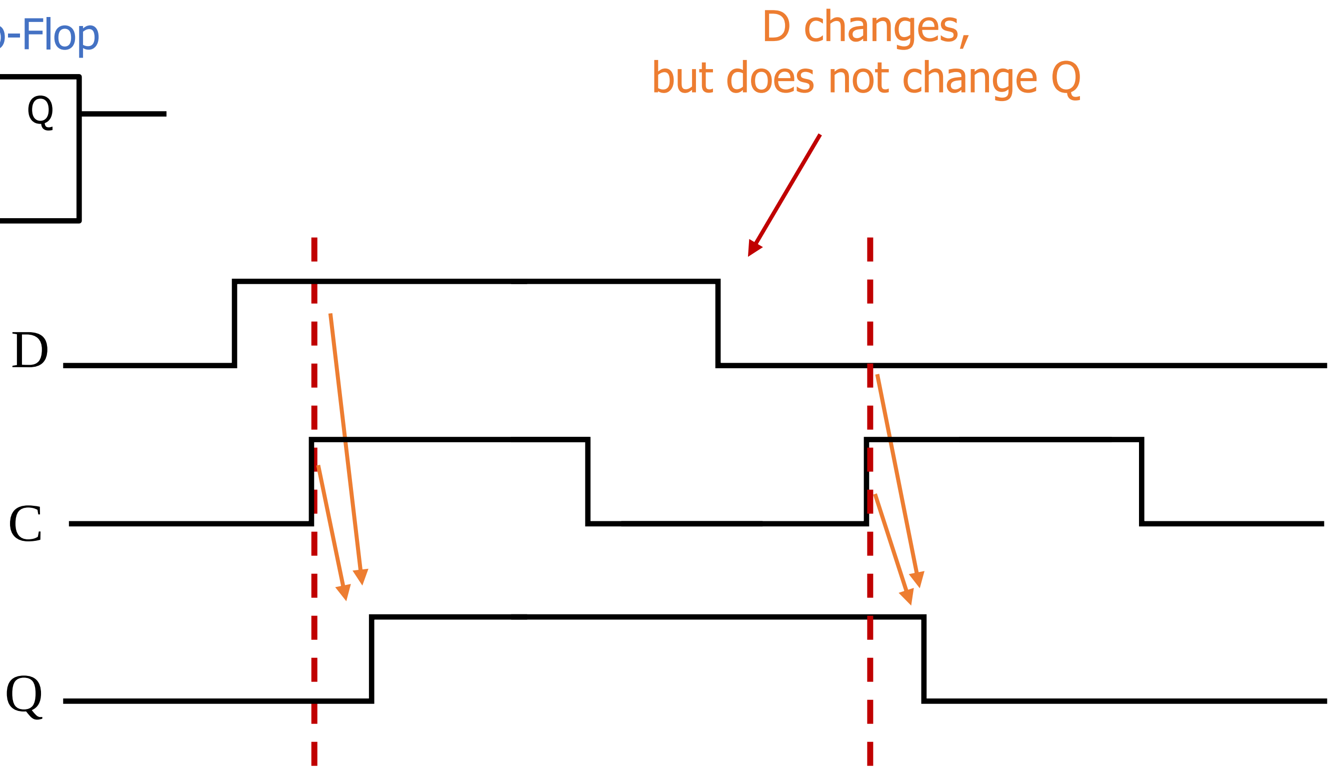


Based on bistable element, we build flip-flops, which are easier to use

D flip-flop, positive edge triggered

- D flip-flop stores input D at the trigger, the rising edge of C
  - We can control what and when
  - To store a bit in D flip-flop: set up D, and then make C transit from 0 to 1
  - The saved bit does not change until another value is stored



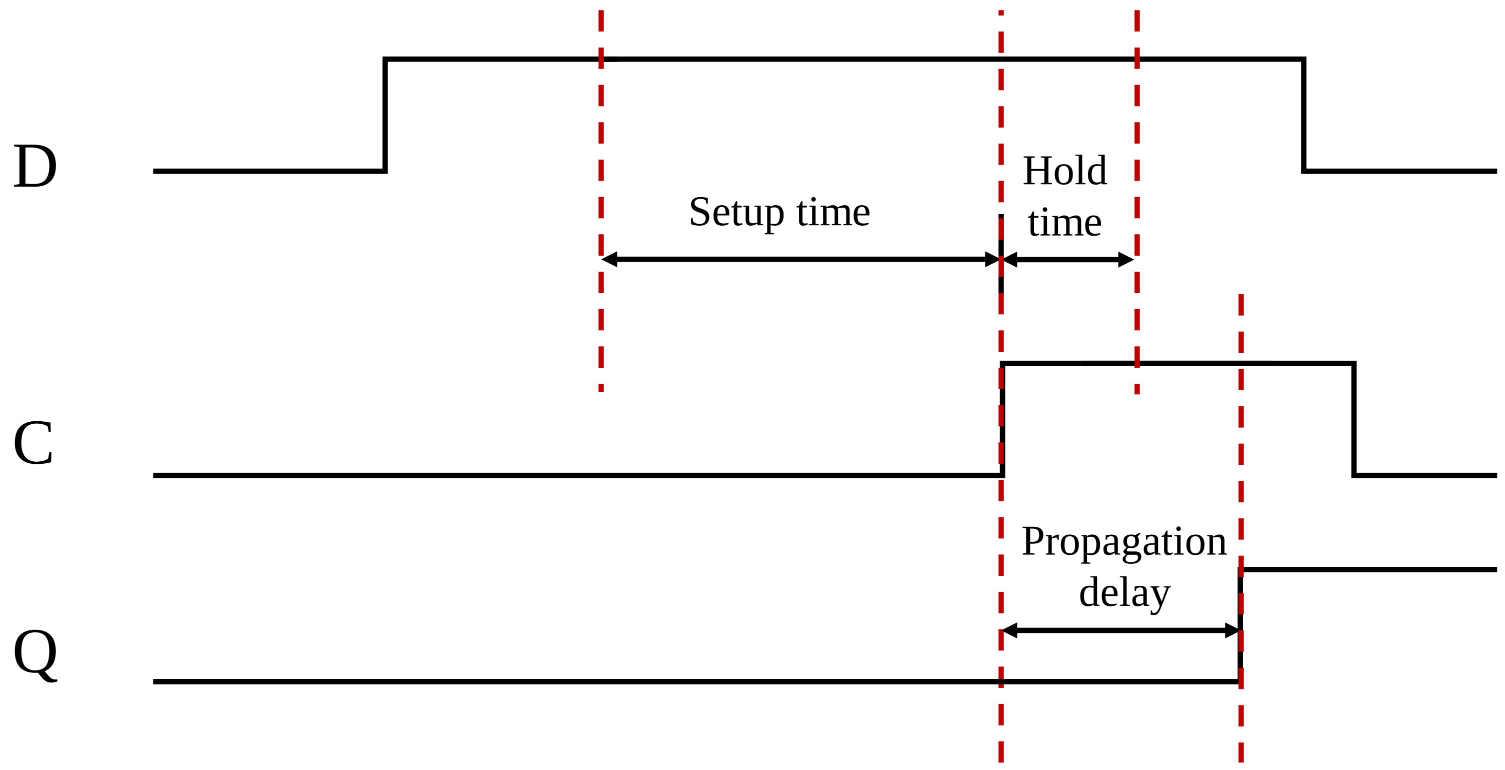


Timing requirements of D Flip-Flop

Setup time: D has to be steady for some time before the edge

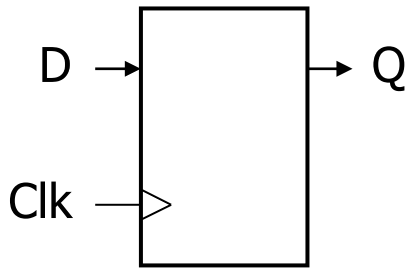
Hold time: D has to be steady for some time after the edge

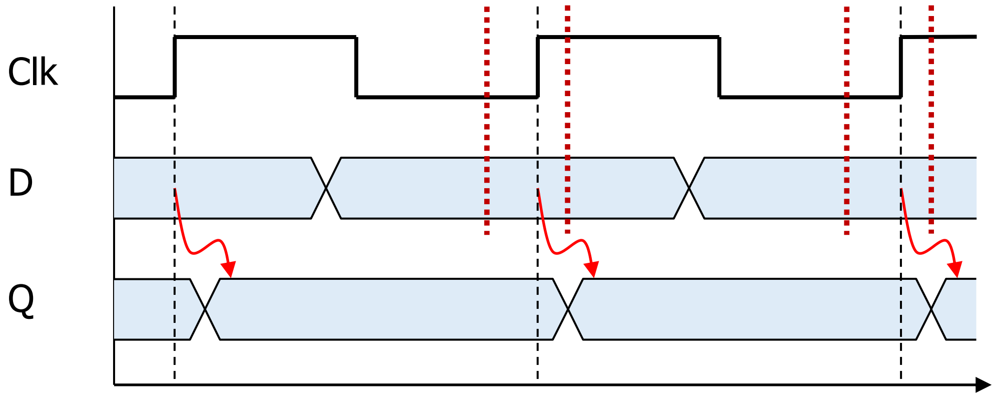
Propagation delay: Time for input to propagate to output



Register

- Register: a memory element that stores data
  - Can be a D flip-flop, or other kind of flip-flops
- A clock signal determines when to update the data
  - The timing diagram below is for a positive edge-triggered register
  - Update happens when clock changes from 0 to 1
- Data stored in registers are steady until next trigger

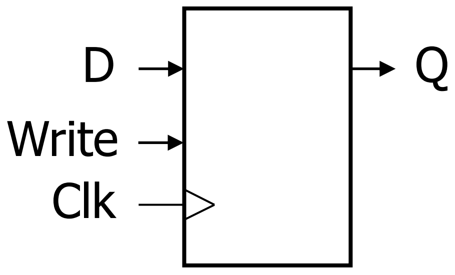


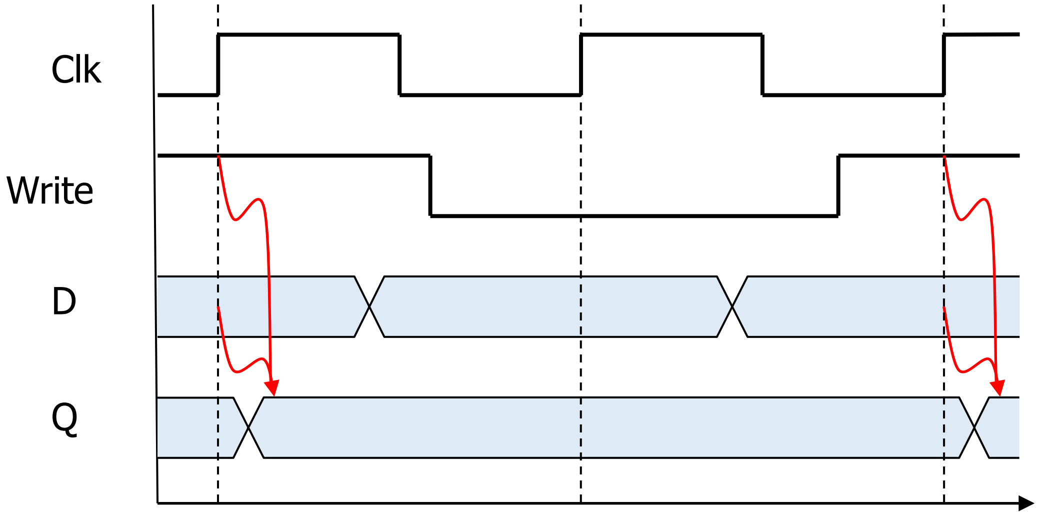


Register with write control

- The register is updated with D on clock edge only when write control input is 1
  - Otherwise, keep the original value in the register
  - Often, the Write signal is also called Enable

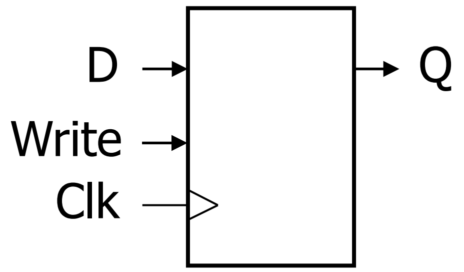
How would you add the Write control?





Multibit Register

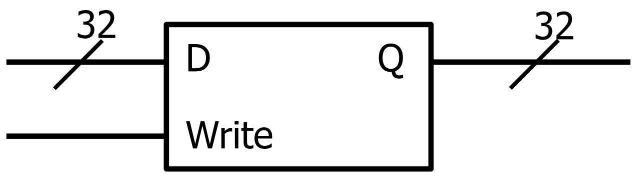
- But that's only 1 bit
- How would you build a 32-bit register?

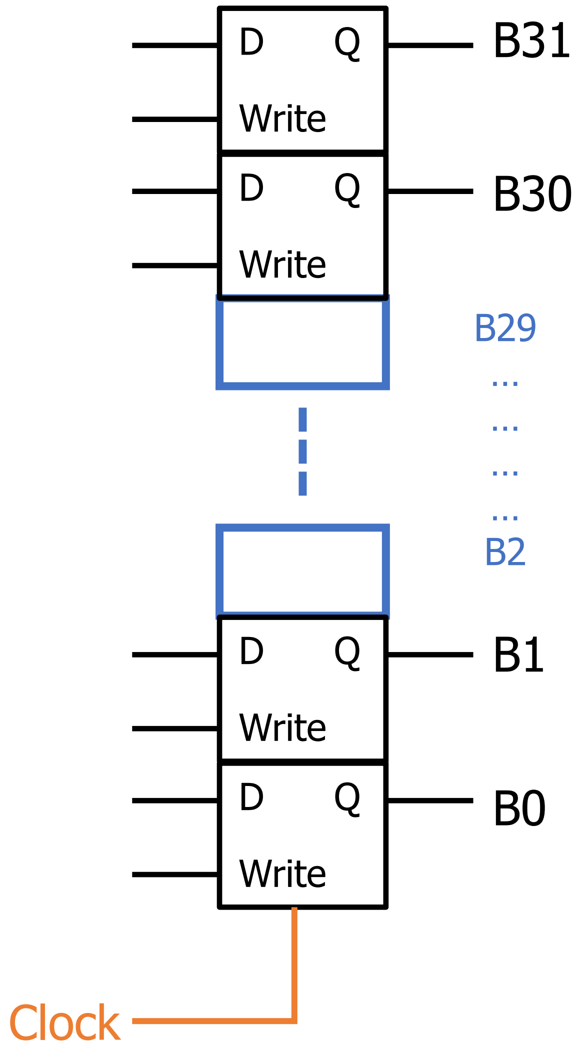


We could use Write to block clock.
It is better to use Write to select one of D and Q

32-bit Register

- An array of 1-bit registers
  - Controlled by the same clock

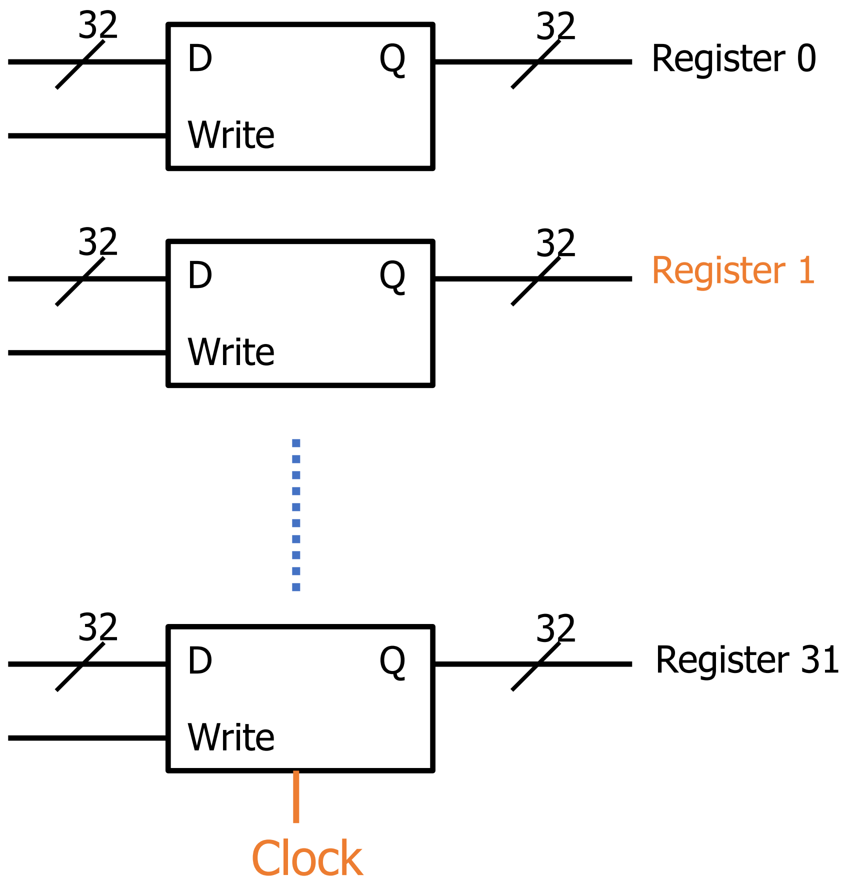




Remember RISC-V has 32 registers?

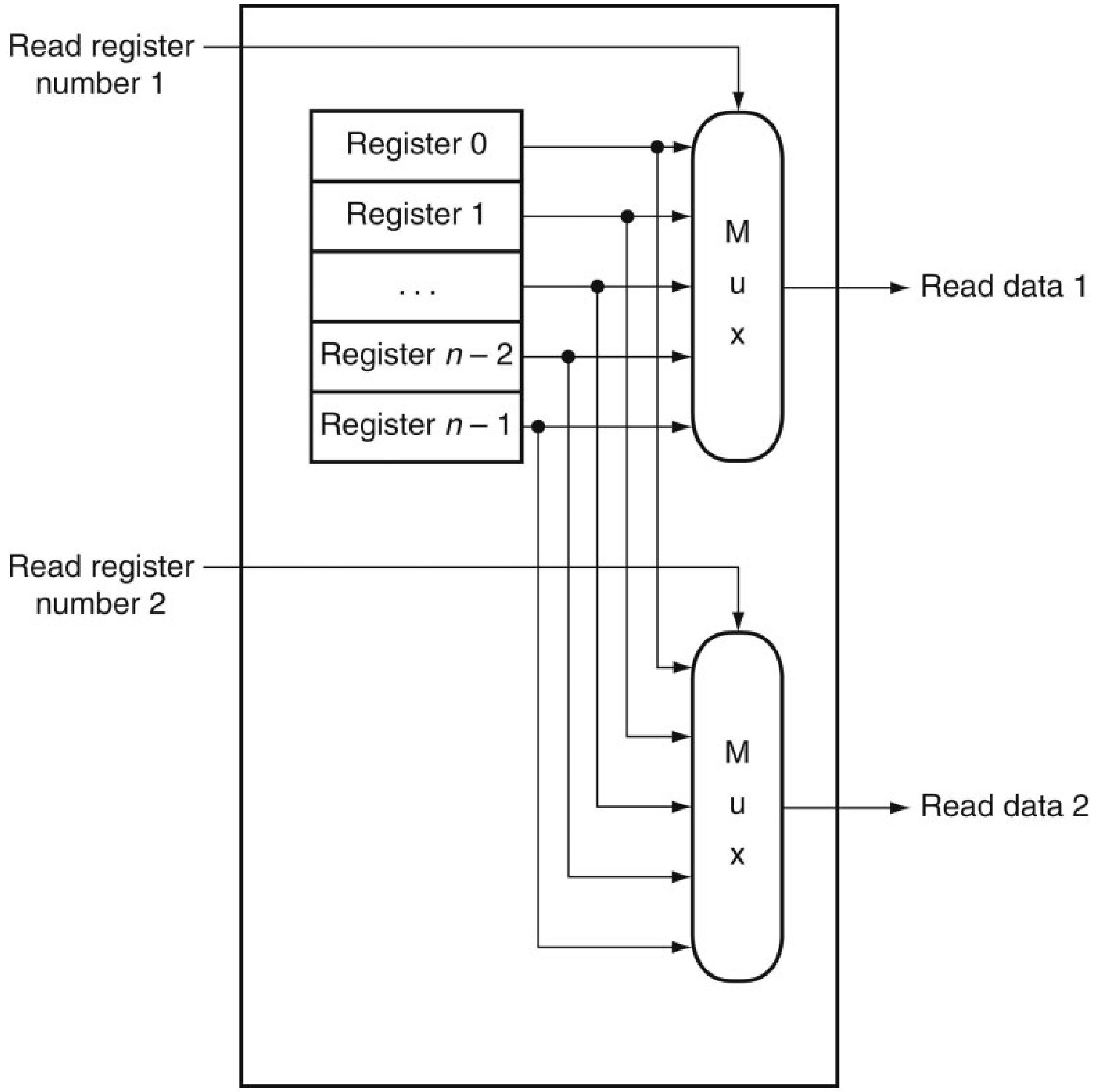
Register File

- The register file has 32 32-bit Registers
- How do we select the register we need?



Inside the RF: Read ports

- Two MUXes for Two read ports



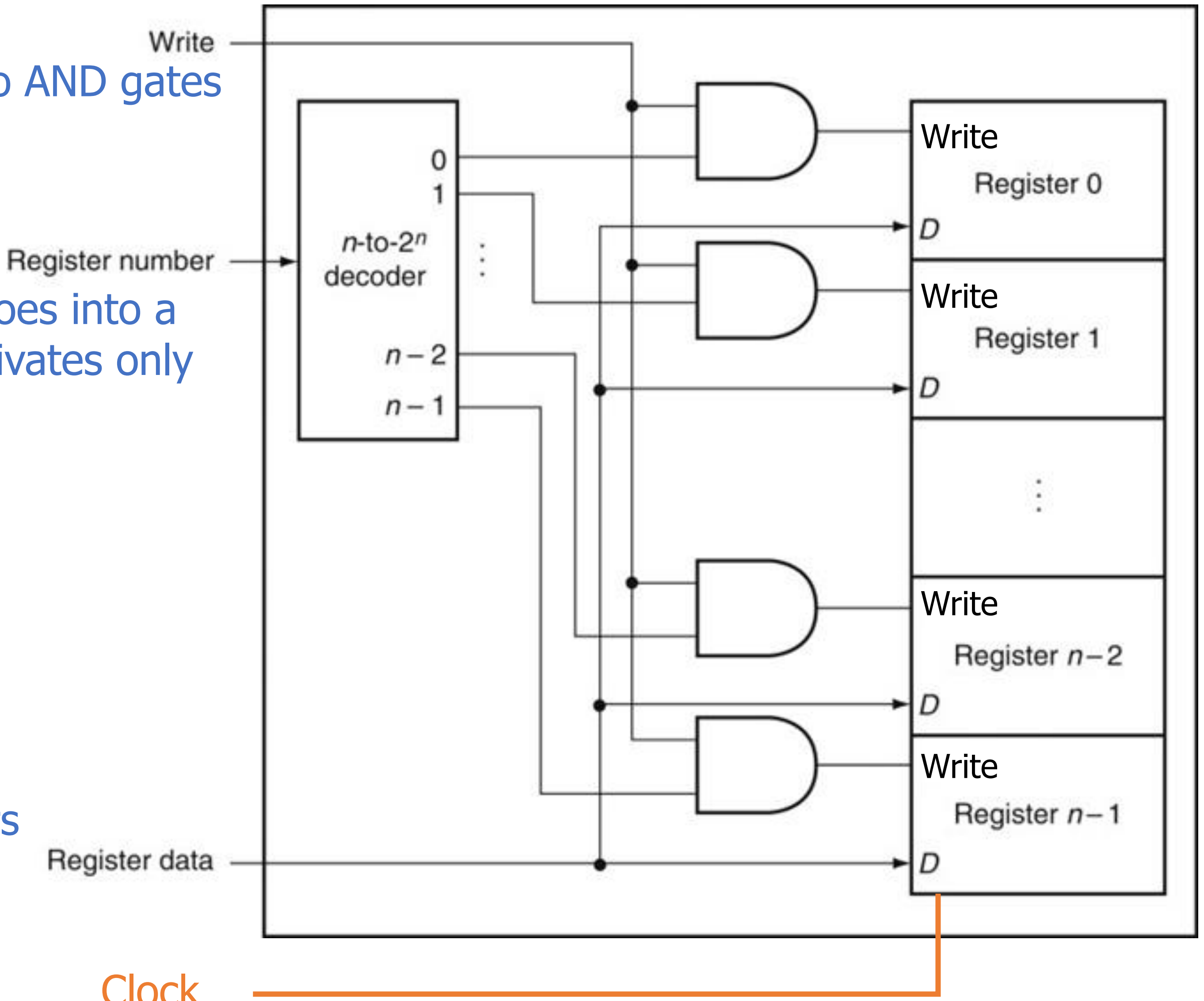
How do we select the register to write?

Inside the RF: Write port

Write goes to AND gates

Register number goes into a decoder, which activates only one register

Register Data are sent to all registers



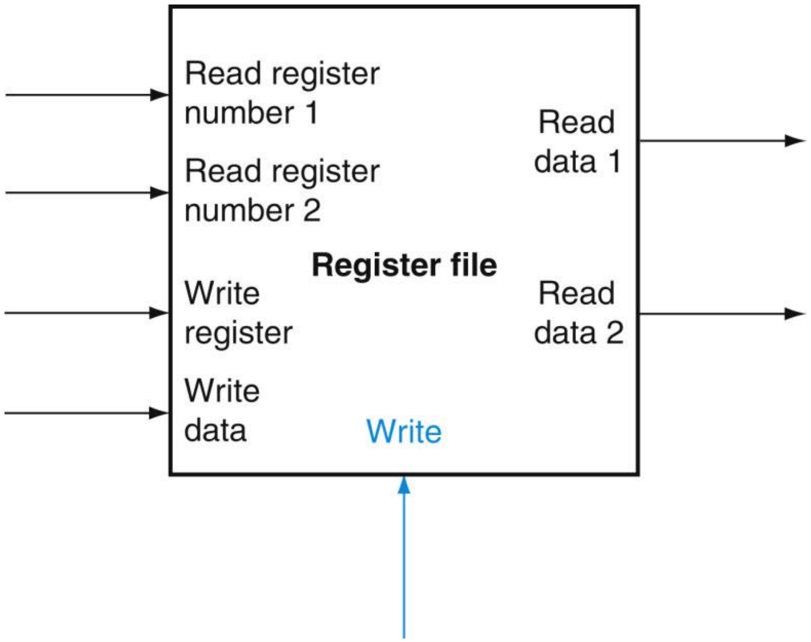
Register File

- Register File (RF) has a collection of registers
- RISC-V RF has 32 32-bit registers
  - Two read ports: can read two registers at the same time
  - One write port
    - Set Write to 0 if the instruction does not write to a register

Read
Set read register numbers
Wait for data to be ready

Write
Set write register and write data
Set Write to 1
Wait for clock to change

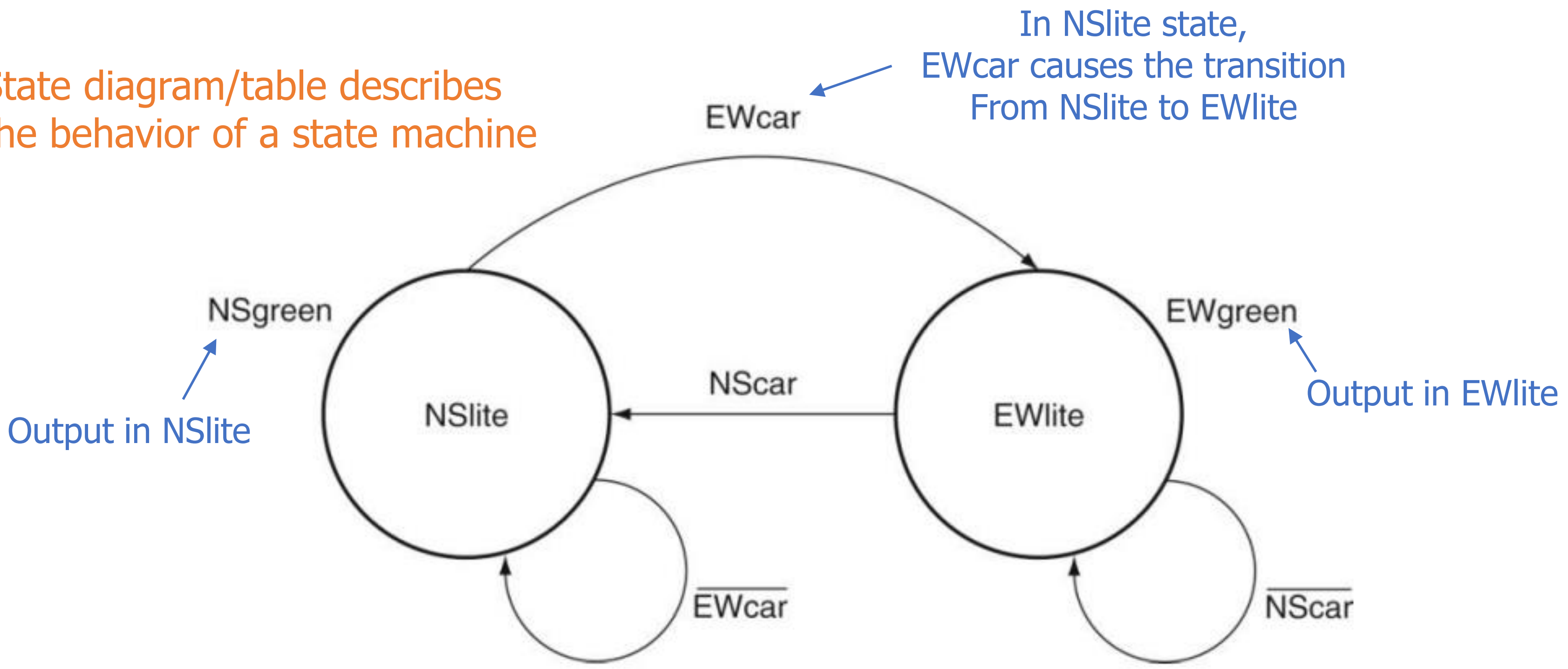
Clock? Not shown.



State Machine

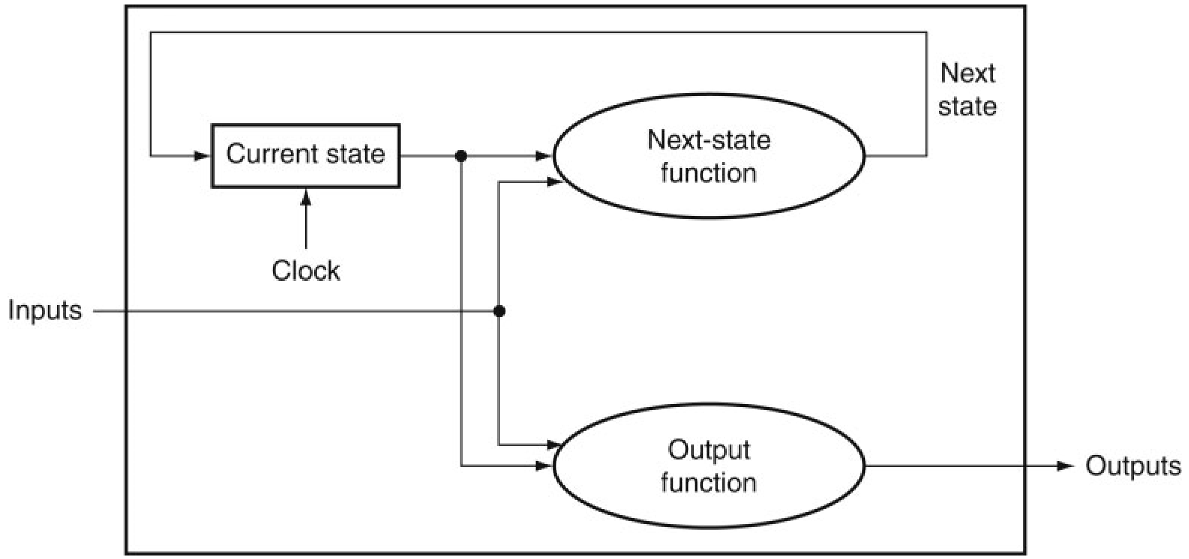
- The circuit has multiple states
  - A state is a summary of previous inputs
- The circuit does two things, depending on current state and input
  - Generate output,
  - Decide the new state to transit to

State diagram/table describes the behavior of a state machine



Components in a State Machine

- A state machine consists of
  - Memory elements that keep the state
    - Commonly use edge-triggered memory elements
    - All bits are updated at the clock edges and kept steady during the cycle
  - Two combinational functions
    - One generates output
    - One generates the next-state, which will be saved in memory elements when triggered

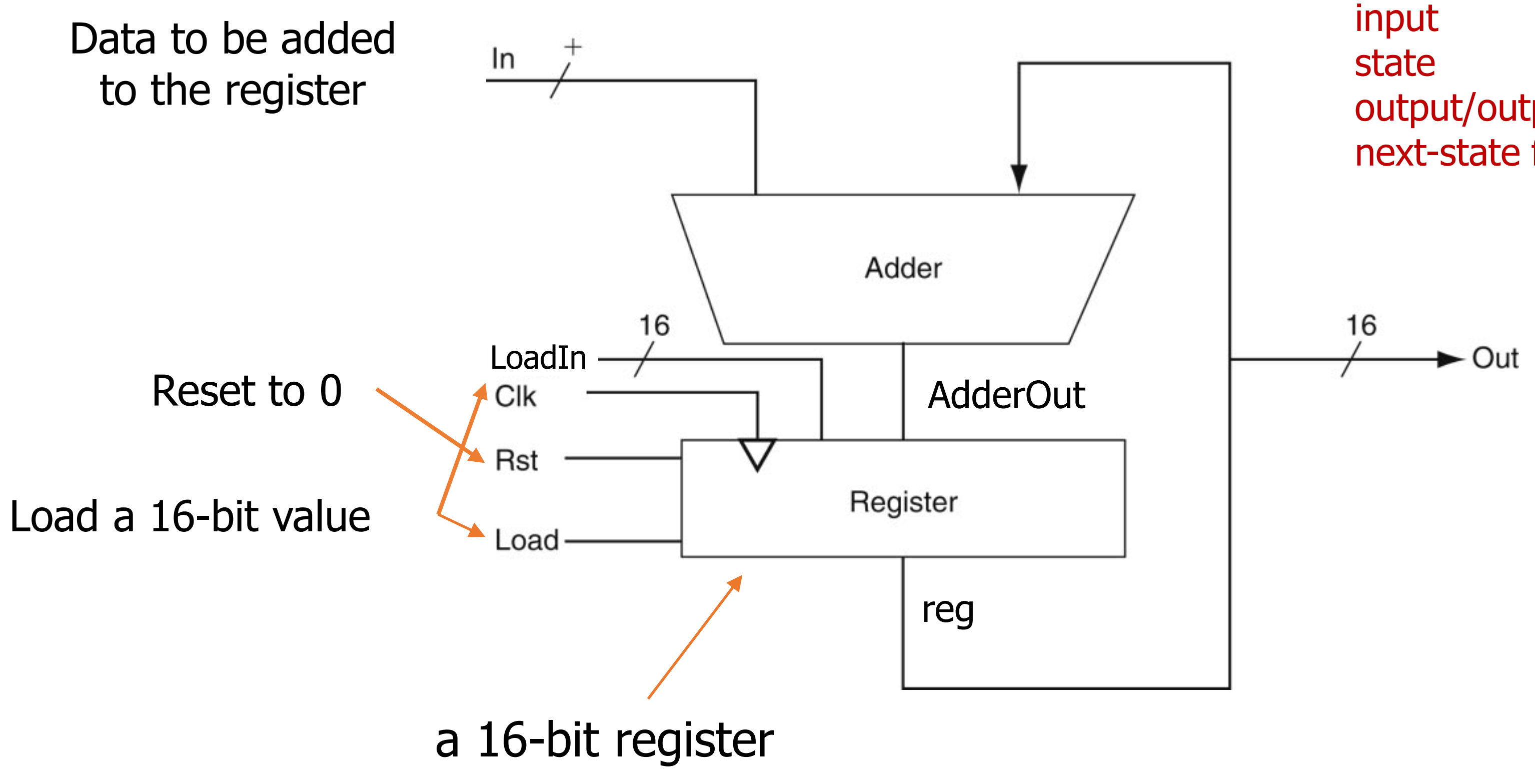


Example: a 16-bit Accumulator

- Add an input number to the existing value in the register
  - If In is 1, the accumulator works as a counter

Identify the following:

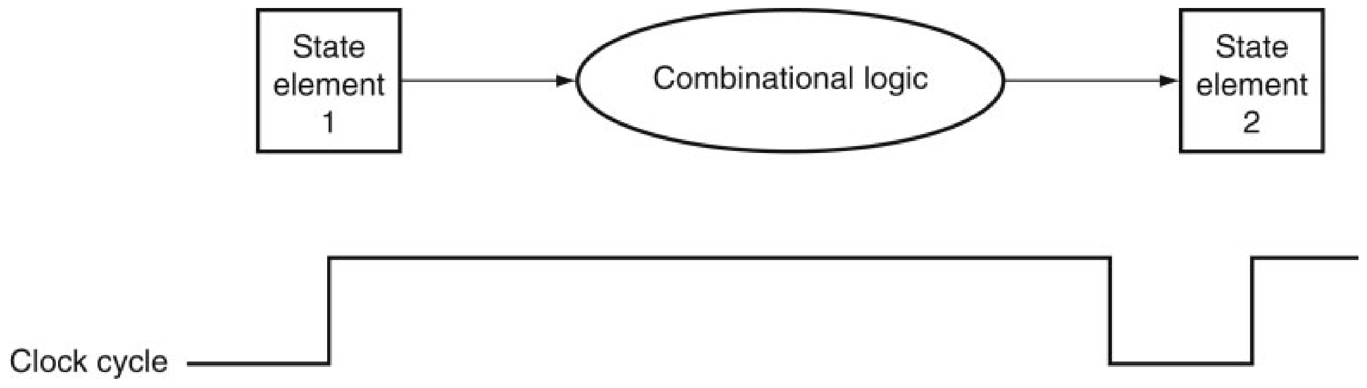
input
state
output/output function
next-state function



cse3666/counter.py at master · zhijieshi/cse3666 (github.com)

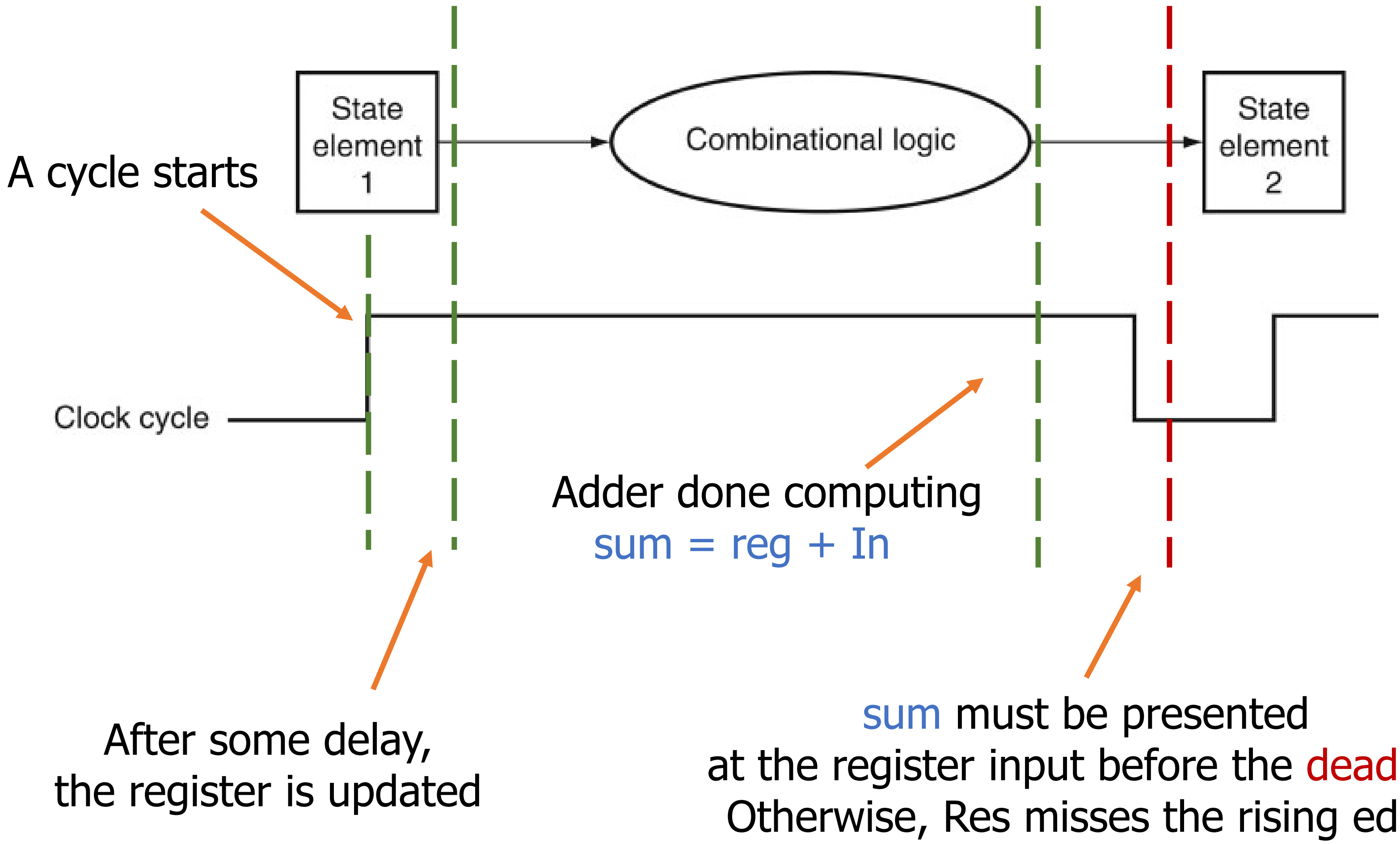
What is happening in a cycle

- Between clock edges:
  - New state is stored in the state elements
  - Combinational logic computes
  - State for next cycle is presented at the input of the state elements
- The clock cycle must be long enough to complete all work



Example of a 16-bit Accumulator

- What happens in a cycle (when accumulating)?



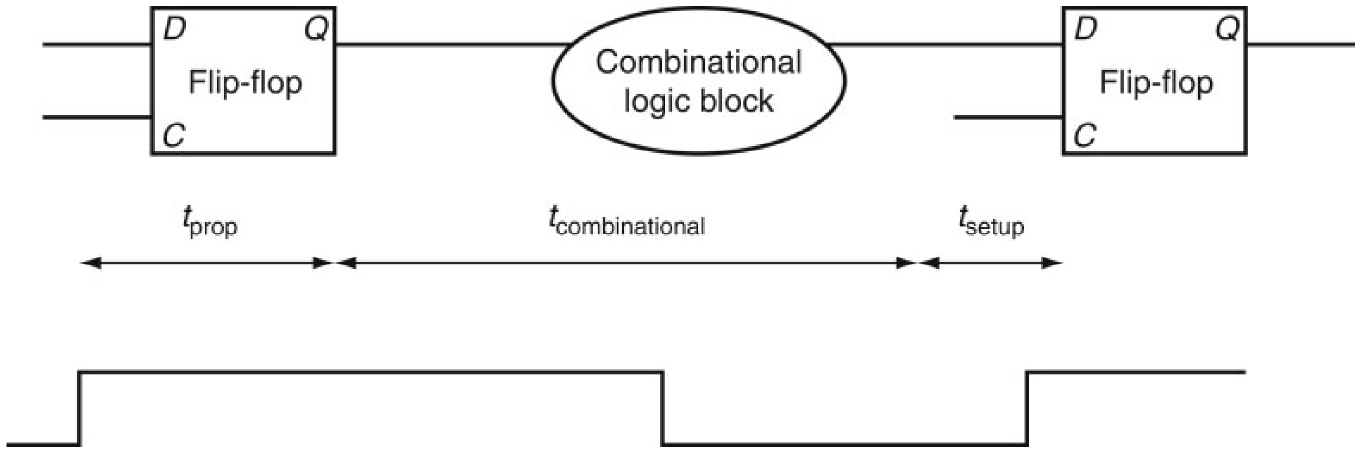
Clock Rate

The clock cycle must be longer than the sum of the following delays

t_{prop} : The time for a flip-flop to propagate input to the output;

$t_{combinational}$: The time for the combinational logic to work;

t_{setup} : New state must arrive early enough to meet the setup requirement



Identify the delays on the previous slides.

Where is the hold time?

Question

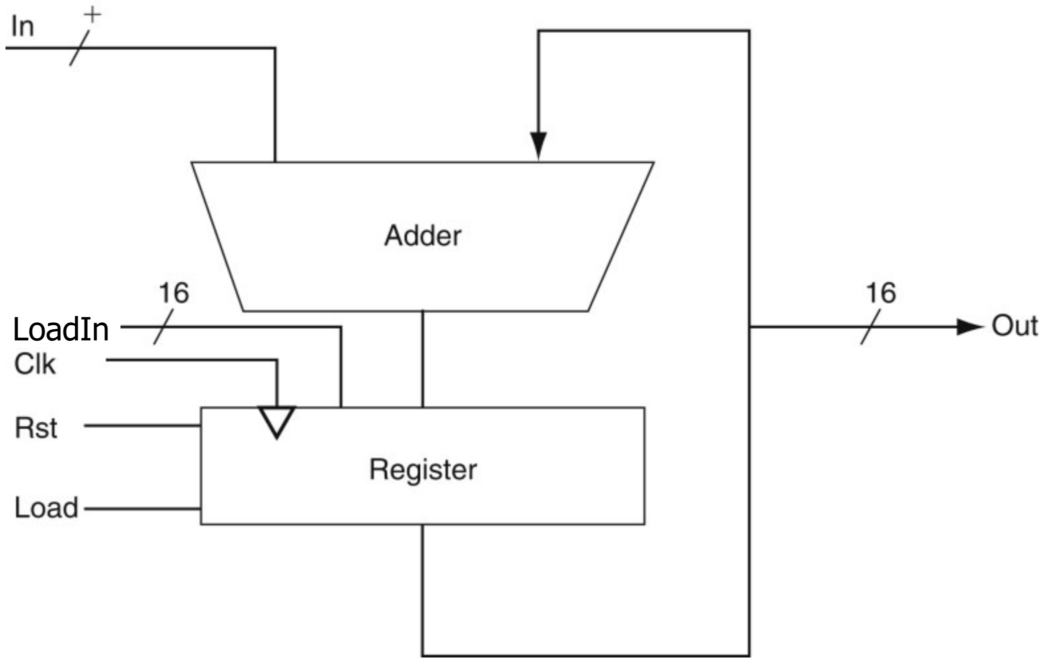
The setup time and hold time of the register is 2ns and 1ns, respectively

The propagation delay of the register is 3 ns.

The propagation delay of the adder is 10 ns.

What is the fastest clock rate in MHz the accumulator can work at?

Truncate to the nearest integer.



Solutions

The clock cycle must be longer than the sum of the following delays

t_{prop} : The time for a flip-flop to propagate input to the output;

$t_{combinational}$: The time for the combinational logic to work;

t_{setup} : New state must arrive early enough to meet the setup requirement

Clock cycle time >= $t_{prop} + t_{combinational} + t_{setup}$ = 3+10+2 = 15ns

Clock rate <= 1 / 15ns = 0.0667 GHz = 66 MHz

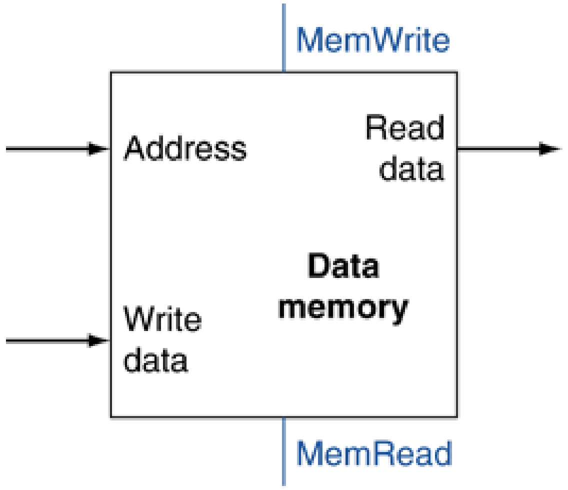
Design of memory

- Memory provides a large storage for processors

- Conceptually, memory is just a large register file, but it is very large, which changes many design decisions
  - Each cell (for a bit) must be small and cheap
    - Not using flip-flops
  - Memory is slow (very slow)
    - Pick one word out of 32 vs one out of 1 billion
    - Memory has its own clock
  - Memory consumes a lot of energy

Memory

- Read
  - Set Address and MemRead (to 1), and wait
  - Get the data from Read data
- Write
  - Set Address, Write data, and MemWrite (to 1), and wait

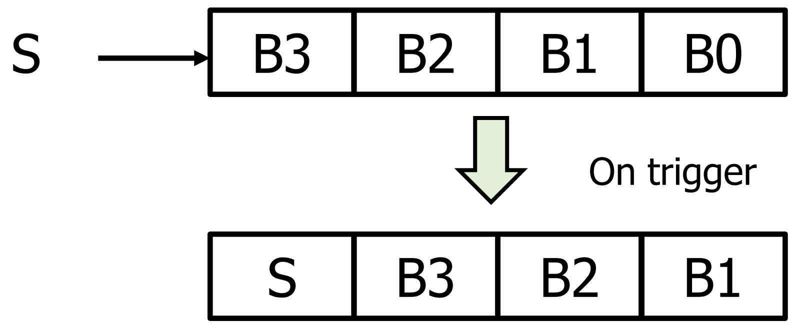


MemWrite and MemRead should not be 1 at the same time

Example: Shift Register

- Registers that can shift bits to right (or left)
  - A simple state machine

For example: 4-bit shift register (shift right)



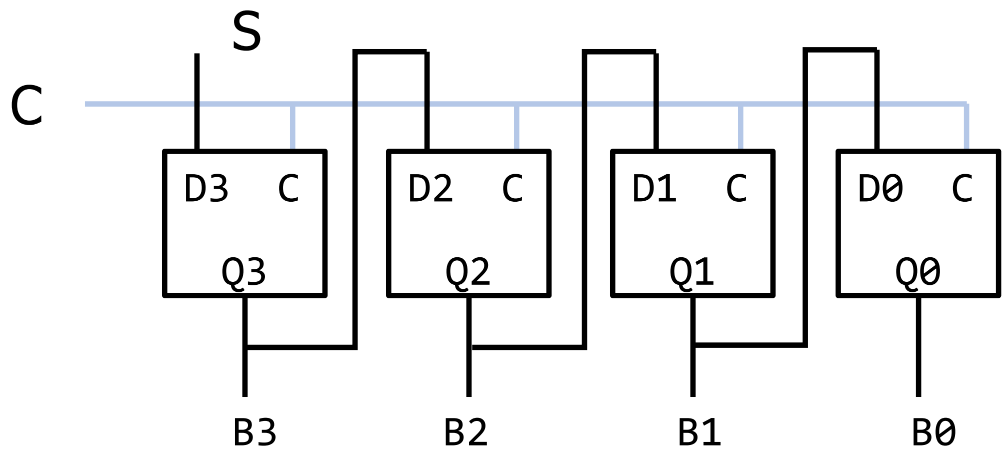
Cycle	S	B3	B2	B1	B0
0	0	1	0	1	1
1	1	0	1	0	1
2	1	1	0	1	0
…					

Example: 4-bit shift register

Shift right

Bits to be stored in the register in the next cycle: (S, B3, B2, B1)

- S is connected to D3 because we want S to be stored in bit 3 of the register
- Similarly, B3 is connected to D2, B2 to D1, and B1 to D0

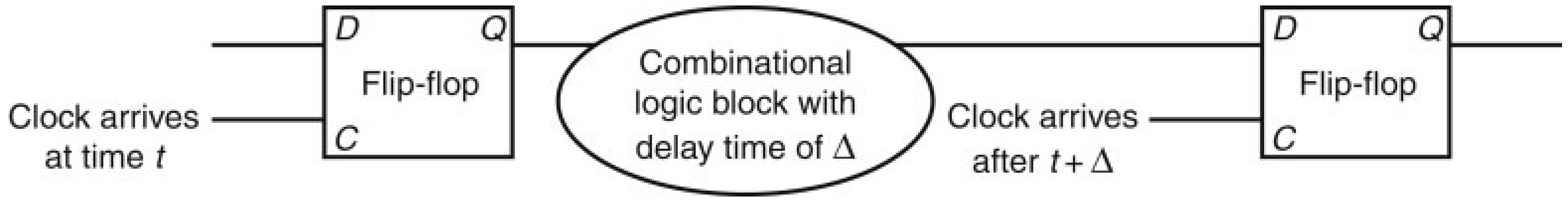


How about shift left?

cse3666/shift-regsiter.py at master · zhijieshi/cse3666 (github.com)

Clock skew

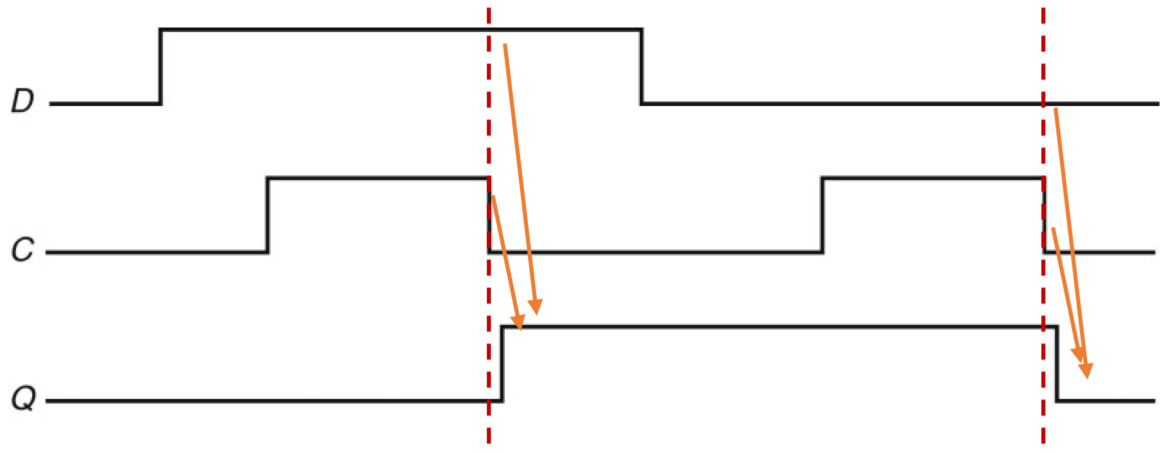
- Clock arrives at memory elements at different times
  - Also called timing skew
- Clock cycle time needs to include the clock skew
  - However, we don't consider it in this course



D
Q
Flip-flop
C
Clock arrives at time t
Combinational logic block with delay time of Δ
Clock arrives after $t+\Delta$
D
Q
Flip-flop
C

D flip-flop, negative edge triggered

- Negative edge triggered D flip-flop stores D at the falling edge



Register with reset and enable in HDL

In Verilog:

```
always @ (posedge clk or posedge reset)
begin
    if (reset == 1) begin
        q <= 0;
    end else if (enable == 1) begin
        q <= d;
    end
end
```

In MyHDL:

```
@always_seq(clk.posedge, reset=reset)
def seq_reg():
    if enable:
        q.next = d
```

Truth table in state machine

Assume 3-bit state, 2-bit input, 2-bit output
For each state and input combination, specify what the next state and output are

						Next state			Output	
	S2	S1	S0	I1	I0	S2	S1	S0	Out1	Out0
State 0	0	0	0	0	0					
	0	0	0	0	1					
	0	0	0	1	0					
	0	0	0	0	1					
State 1	0	0	1	0	0					
	0	0	1	0	1					
	0	0	1	1	0					
	0	0	1	0	1					
	...									