CSC 411

Computer Organization (Fall 2024)
Lecture 21: Sequential logic

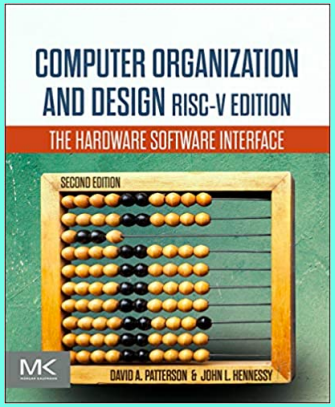
Prof. Marco Alvarez, University of Rhode Island

Disclaimer

Some figures and slides are adapted from:

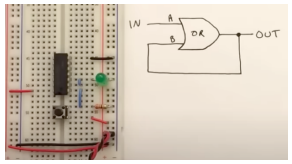
Computer Organization and Design (Patterson and Hennessy)

The Hardware/Software Interface

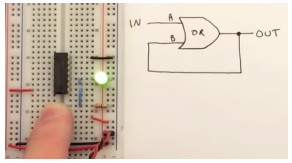


Storing 1 bit

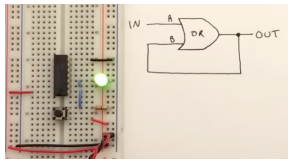
Storing 1 bit



0 => 0



1 => 1



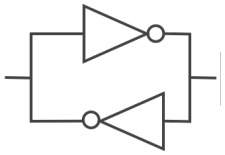
0 => 1

remembers past value,
but can't reset it

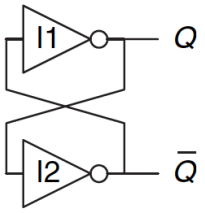
https://www.youtube.com/watch?v=KM0DdEaY5sY

Storing 1 bit

- Bistable circuits
  - can have two distinct states, once they are in one state, they remain there
  - third possible metastable with inputs oscillating between 0 and 1



loop is used to keep the state stable



Not useful without a mechanism for changing the output (Q)

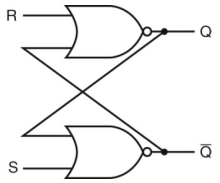
Latches and flip-flops

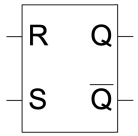
SR-latch

- Set-reset latch
  - when S is high output 1, when R is high output 0, when both are low preserve the current state (memory!)
    - use a pair of cross-coupled NOR gates (could also use NAND gates)

R	S	Q	
0	0	Q	HOLD
0	1	1	SET
1	0	0	RESET
1	1	INVALID	

breaks the invariant





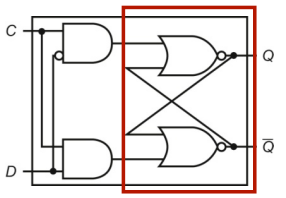
Invariant: always two outputs Q and $\overline{Q}$

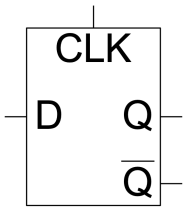
D-latch (level triggered)

- Guarantees correct operation
  - adds an inverter to avoid the undef state (S=1 R=1)
- Clock signal C (clock) controls when to store D
  - C is high, Q continuously changes as D changes (**transparent mode**) — latching
  - C is low, Q doesn't change with D (**opaque/latch mode**)

C	D	Q	
0	0	Q	HOLD
0	1	Q	HOLD
1	0	0	RESET
1	1	1	SET

SR latch

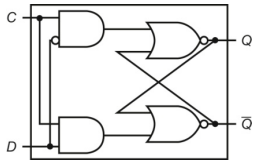


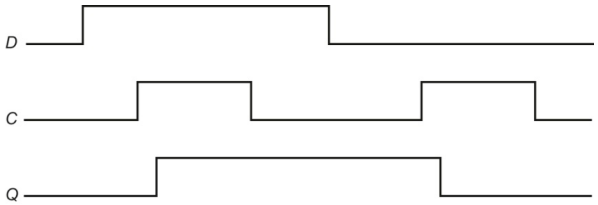


Practice

- Consider the following operation of a D-latch

C	D	Q
0	0	Q
0	1	Q
1	0	0
1	1	1

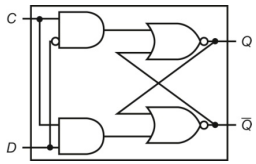


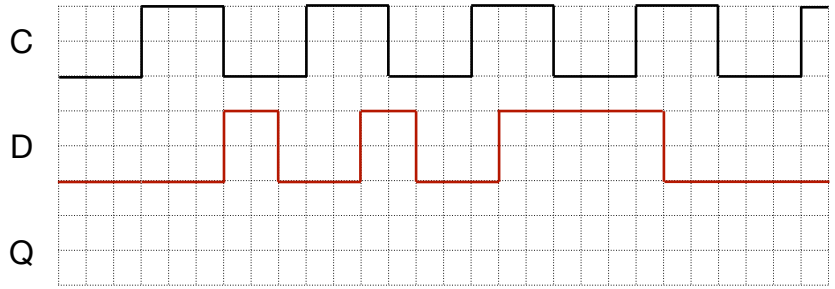


Note there is a small propagation delay from D to Q

Practice

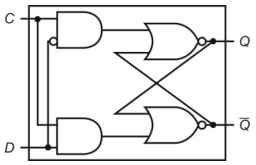
- Draw the values of Q over time

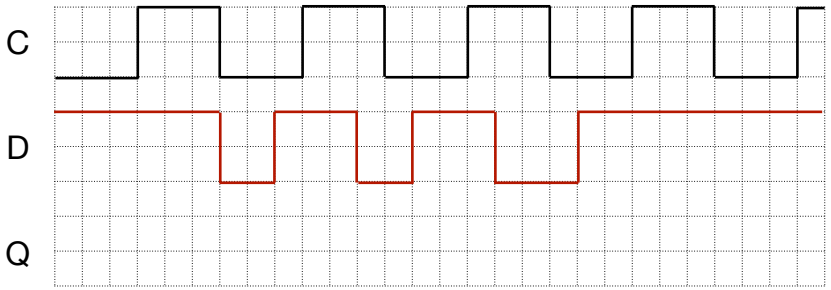




Practice

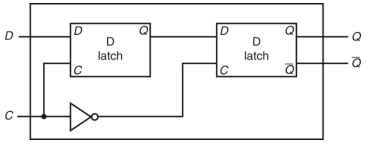
- Draw the values of Q over time
  - is there a problem with the output?



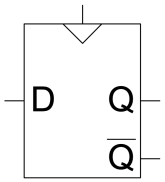


D-flip-flop (edge triggered)

- D-latches are transparent
  - doesn't help if we want to ensure data is available during the **entire clock cycle**
- D-flip-flop with a **"falling"** edge trigger
  - **master** and **slave** latches connected in series
  - the flip-flop updates its stored value <u>only on a falling clock edge</u>
  - state updated in a synchronized fashion
    - when C rises $0 \rightarrow 1$, Q does not change; when C falls $1 \rightarrow 0$, Q changes

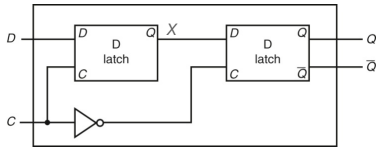


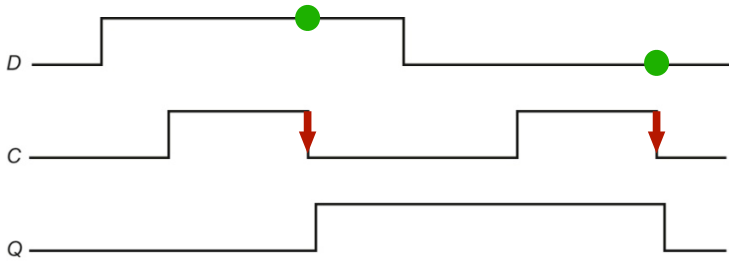
Master latches the input D when the clock C is high. When the clock C falls, the master is opaque and the slave is transparent.



Practice

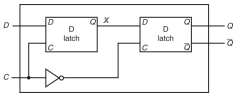
- Consider the following operation of a D-flip-flop with a **falling-edge trigger**

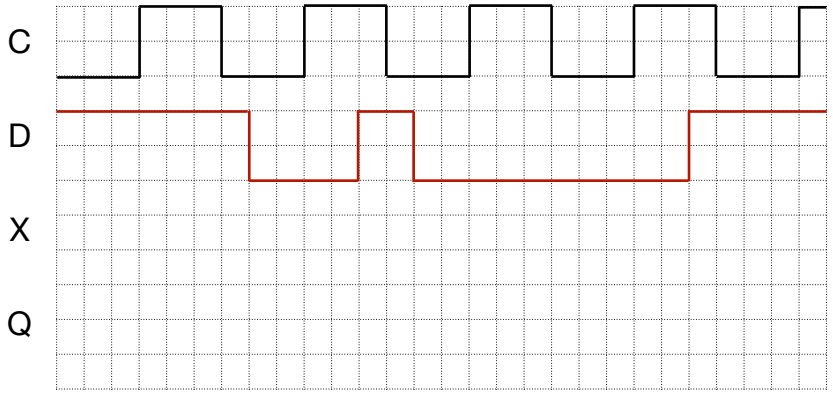




Practice

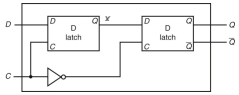
- Draw the values of Q and X over time

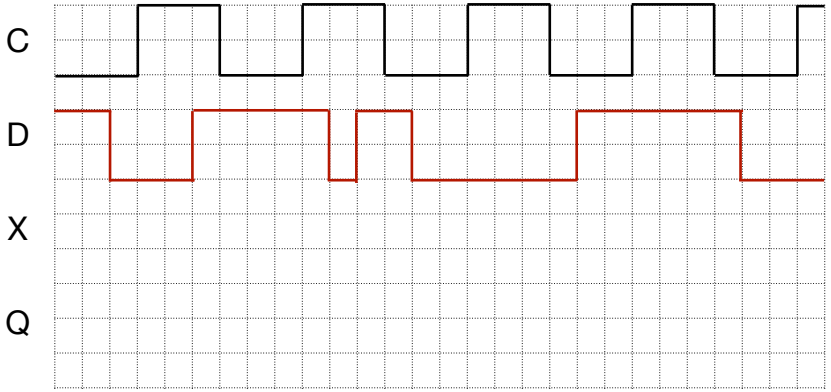




Practice

- Draw the values of Q and X over time





D-latch vs D-flip-flop

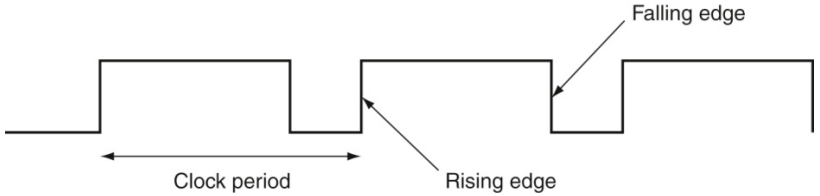
- D-latch is a **transparent** storage element
  - the output follows the input as long as the clock signal is active
  - captures and propagates any **temporary glitches** or **fluctuations** on the input
- D-flip-flop is an **edge-triggered** storage element
  - on the clock edge (rising/falling depending on implementation), the D-flip-flop will latch the input value
  - **output will remain constant until the next clock edge**, even if the input changes in the meantime
  - d-flip-flops filter out temporary glitches or fluctuations on the input and therefore are useful when we want to capture and hold a stable value, rather than continuously track an input
- D-flip-flop needs two D-latches
  - requires more transistors, power and an increased propagation delay
  - single D-latches can be used as long as the need for stable inputs is not higher than half a cycle

Sequential logic

CPU clocks

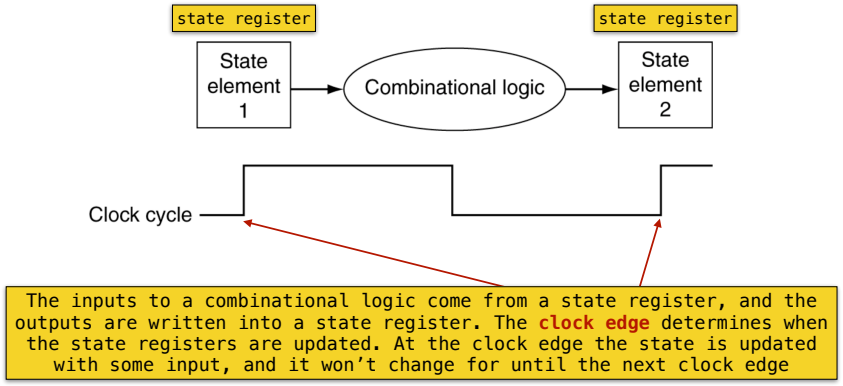
$$F = \frac{1}{T}$$

- All circuits on a chip share a **clock signal**
  - tells circuits when to accept inputs and how much time they have
  - **Clock frequency (F)** (clock rate) is number of clock cycles that occur per second
  - **Clock period (T)** is the amount of time it takes for one complete clock cycle to occur
- Clock signal is used to **trigger state changes**, ensuring the correct sequence of operations



Sequential logic

- Sequential circuits consist of combinational logic and a storage element (used to store state)
  - update memory according to input and previous state

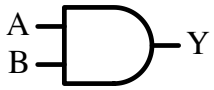


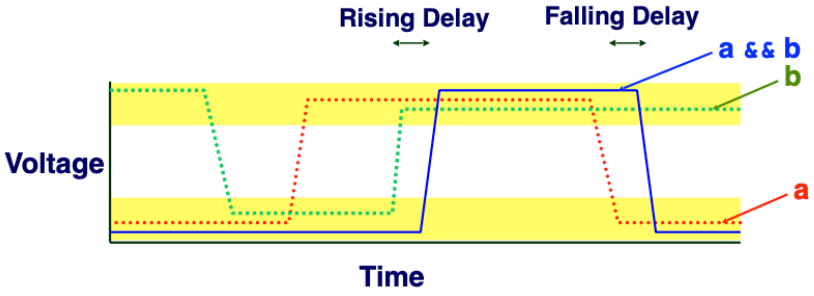
Clocking methodology

- Combinational logic transforms data during clock cycles
  - to be precise, between clock edges (can be rising or falling)
  - takes input from state elements and stores output into state elements
- Longest delay (critical path) determines minimum clock period

Timing

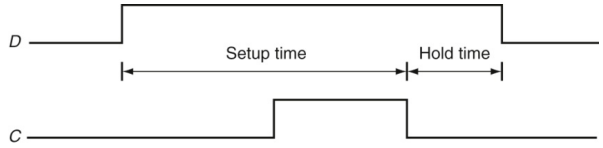
- Delays are a fact of life





Timing

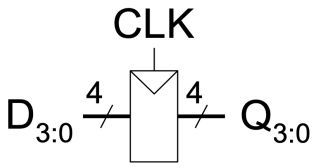
- The input must be **stable** for a period of time before the clock edge, as well as after the clock edge
  - minimum time the signal must be stable before the clock edge is called the **setup time**
  - minimum time the signal must be stable after the clock edge is called the **hold time** (usually 0 or very small)
- Failure to meet these minimum requirements can result in a situation where the output of the state element may not be predictable



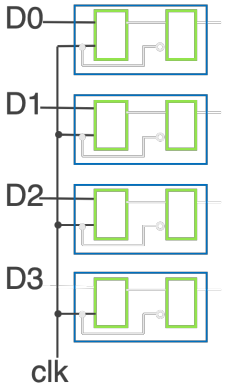
Register file

Register

- Can use D-flip-flops in parallel
  - shared clock, and may have extra inputs, e.g., write enable
- 4-bit register

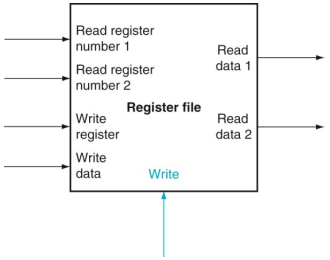


Register, stores 4 bits and can be read from and written to



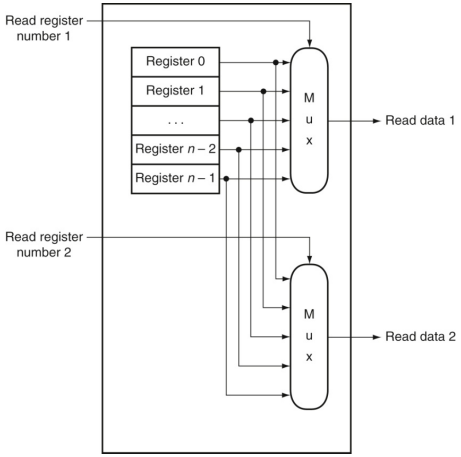
Register file

- For RISC-V, the register file has two read ports and one write port
  - five inputs (2 read addresses, 1 write address, write data, write control) and two outputs (2 read data)
- Very fast storage with only a few gate delays



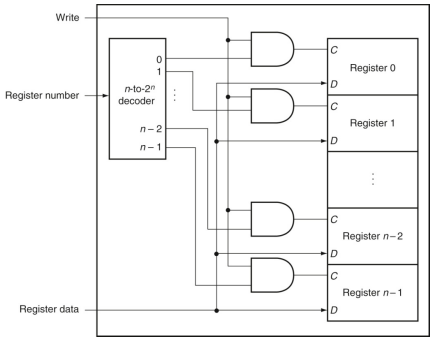
Read ports

- Two read ports for a register file with 32 registers
  - read ports use a pair of multiplexors, each 32 bits wide
  - register read number is used as the multiplexor selector signal
- In RISC-V, most instructions have two source operands



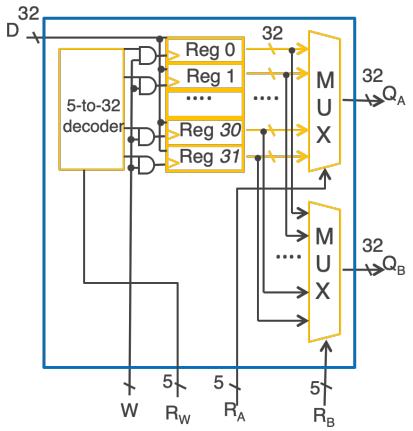
Write port

- Implementation of the write port for a register file
  - requires a decoder and the write signal (clock) to generate the C input to the registers
- All three inputs have setup and hold-time constraints that ensure that the correct data are written into the register file



32-bit register file

- A register file with 32 registers, each storing 32 bits



By using a rising edge triggered register, we can write and read in the same clock cycle. The write occurs at the rising clock edge The read can access the previous value during the clock cycle before the next rising edge.