CSC 411

Computer Organization (Spring 2024)
Lecture 21: Sequential logic

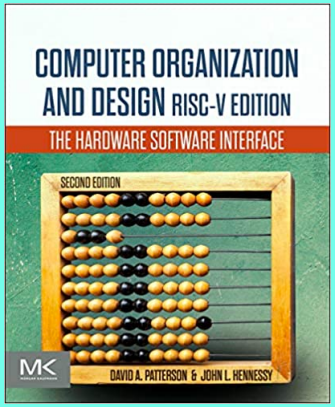
Prof. Marco Alvarez, University of Rhode Island

Disclaimer

Some figures and slides are adapted from:

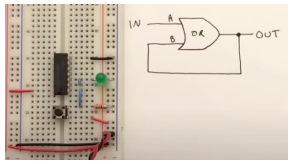
Computer Organization and Design (Patterson and Hennessy)

The Hardware/Software Interface

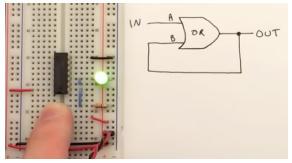


Storing 1 bit

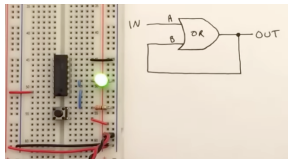
Storing 1 bit



0 => 0



1 => 1



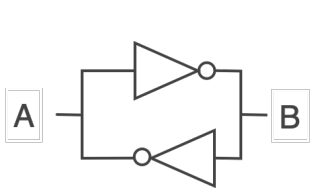
0 => 1

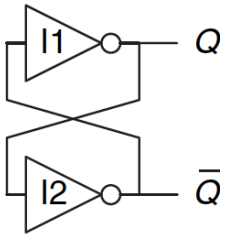
remembers past value,
but can't reset it

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

Storing 1 bit

- Two stable states
  - third possible metastable with inputs oscillating between 0 and 1





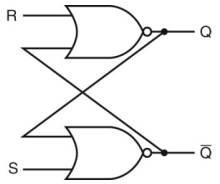
Not useful without a mechanism for changing the output (Q)

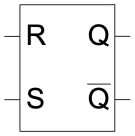
SR-latch

- 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



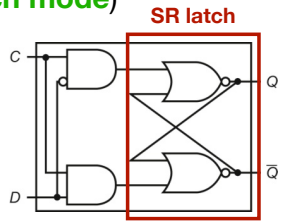


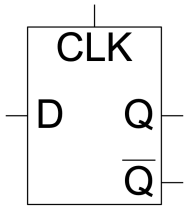
Invariant: always two outputs Q and $\overline{Q}$

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
  - when C is high, Q continuously changes as D changes (**transparent mode**) — latching
  - when C is low, maintain current state, Q doesn't change with D (**opaque/latch mode**)

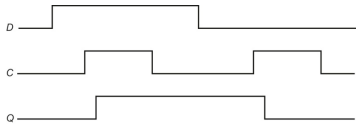
C	D	Q	
0	0	Q	HOLD
0	1	Q	HOLD
1	0	0	RESET
1	1	1	SET



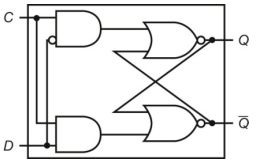


Practice

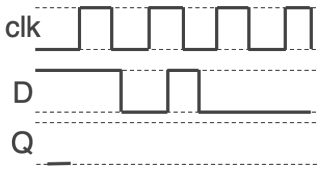
- Consider the following operation of a D-latch



Note there is a propagation delay from D to Q



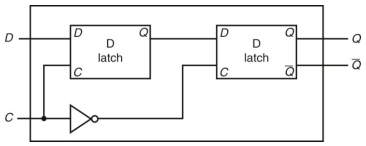
- Draw the values of Q over time
  - is there a problem with the output?



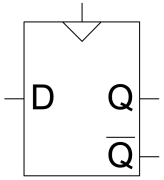
C	D	Q
0	0	Q
0	1	Q
1	0	0
1	1	1

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 only on a falling clock edge
  - state updated in a synchronized fashion: when C rises $0 \rightarrow 1$, Q does not change; when C falls $1 \rightarrow 0$, Q changes

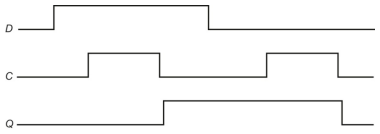


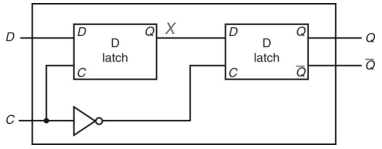
Master latches the input D when the clock C is high. When the clock C falls, the master is opaque and the slave is transparent.



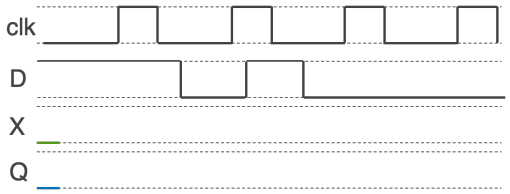
Practice

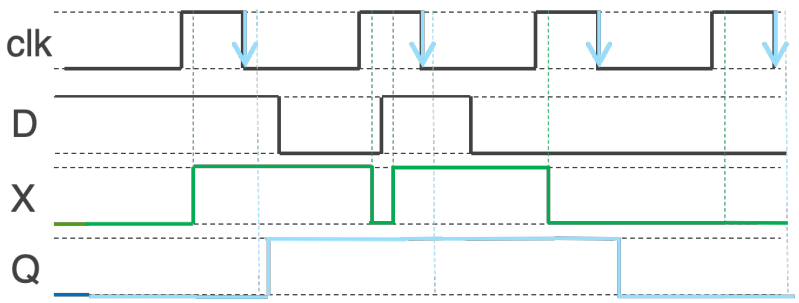
- Consider the following operation of a D-flip-flop with a falling-edge trigger





- Draw the values of Q and X over time





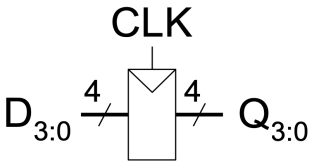
D-latch vs D-flip-flop

- 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 value on the D input
  - **output will remain constant until the next clock edge**, even if the D 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

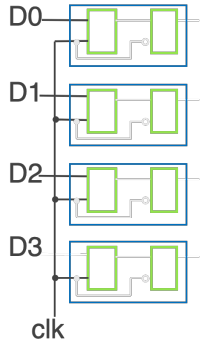
Register file

Register

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

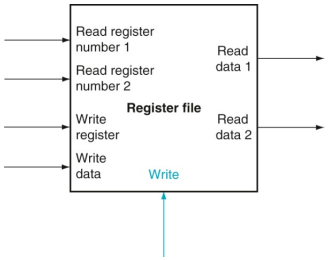


Register, stores 4 bits and can be read from and written to



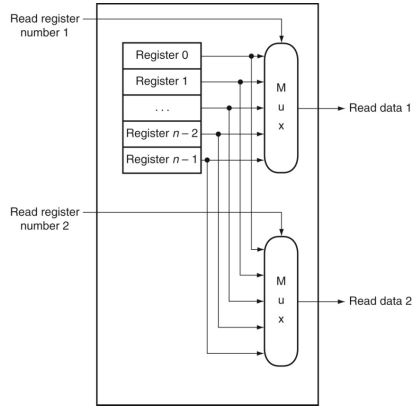
Register file

- For RISC-V, the register file has two read ports and one write port
  - five inputs and two outputs
  - control input **Write** is shown in color
- Very fast storage with only a few gate delays



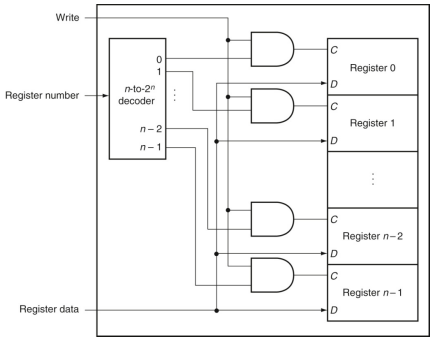
Read ports

- Two read ports for a register file with n 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



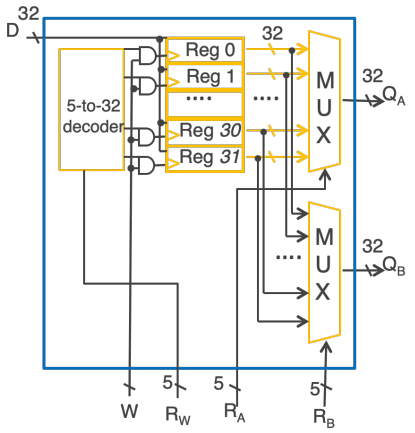
Write port

- 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