Sequential Logic and State

Output depends on inputs *and stored values*.
(vs. combinational logic: output depends only on inputs)

Elements to store values: latches, flip-flops, registers, memory

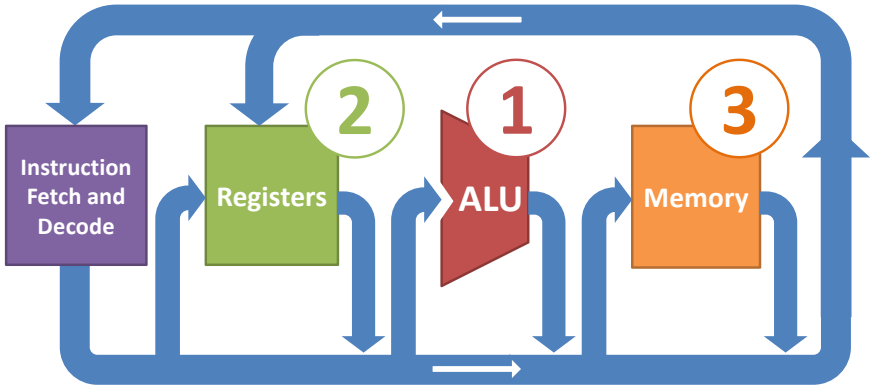
CS 240
Foundations of Computer Systems

Sequential Logic and State

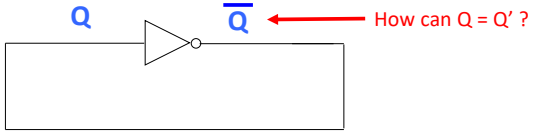
Output depends on inputs *and stored values*.
(vs. combinational logic: output depends only on inputs)

Elements to store values: latches, flip-flops, registers, memory

Processor: Data Path Components



Unstable circuit

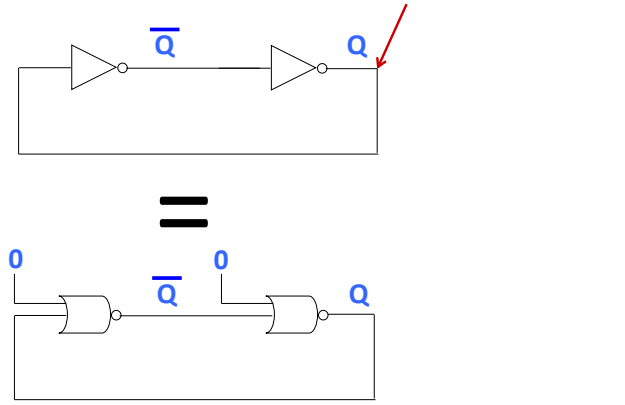


Have this issue with any odd number of inverters in a loop.

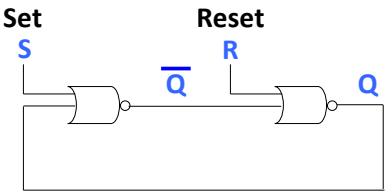
Bistable latches

Things are more sensible with an even number of inverters in a loop.

Suppose we somehow get a 1 (or a 0?) on here.



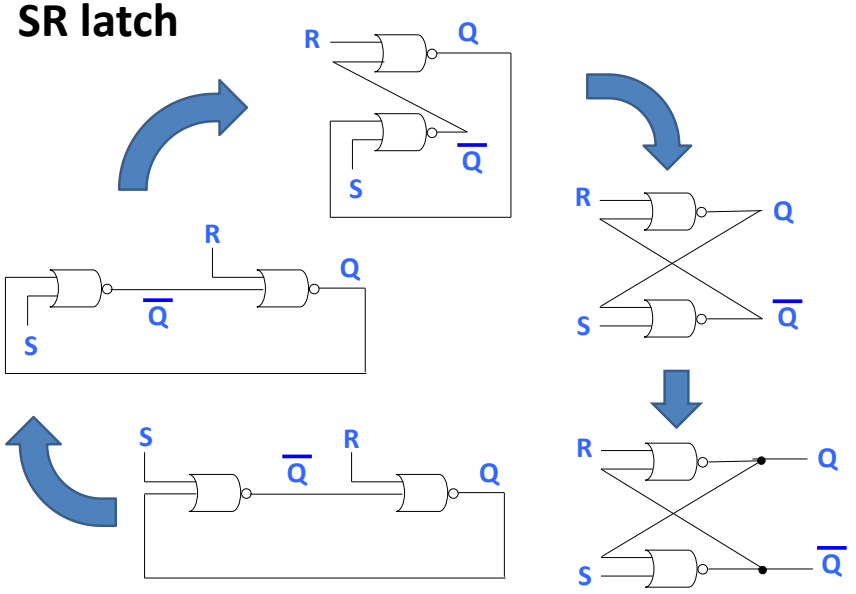
SR latch



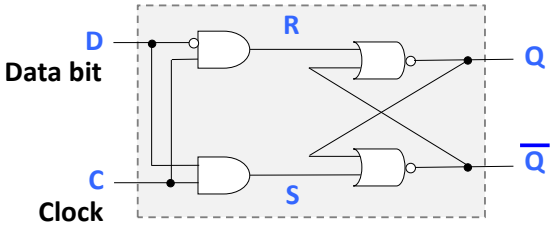
S	R	Q_{prev}	Q'_{prev}	Q_{next} (stable)	Q'_{next} (stable)
0	0	0	1	0	1
0	0	1	0	1	0
1	0	*any*	*any*	1	0
0	1	*any*	*any*	0	1
1	1	*any*	*any*	0	0

violates invariant that Q and Q' are inverses!

SR latch



D latch



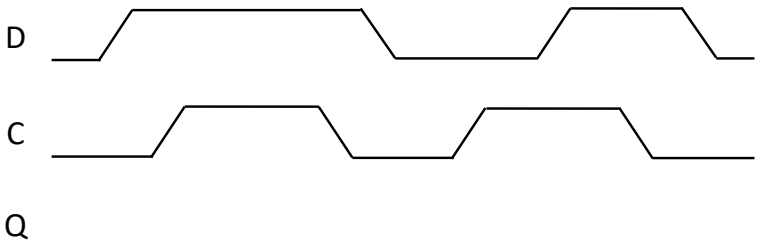
if C = 0, then SR latch stores current value of Q.
if C = 1, then D flows to Q:
    if D = 0, then R = 1 and S = 0, Q = 0
    if D = 1, then R = 0 and S = 1, Q = 1

Notes:

- Data bit D replaces S & R: it's the bit value we want to store when Clock = 1
  - Internally, Data bit D prevents bad case of S = R = 1
- This logic is **level-triggered**; as long as Clock = 1, changes to D have impact

Time matters!

ex

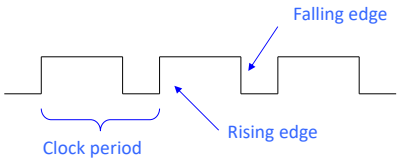


Assume Q has an initial state of 0

Clocks

Clock: free-running signal
with fixed cycle time = clock period = T.

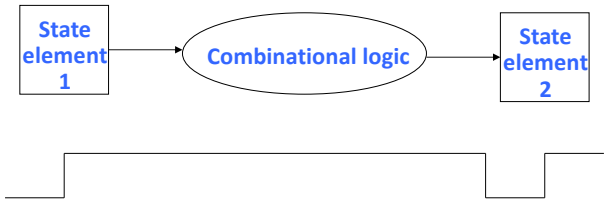
Clock frequency = 1 / clock period



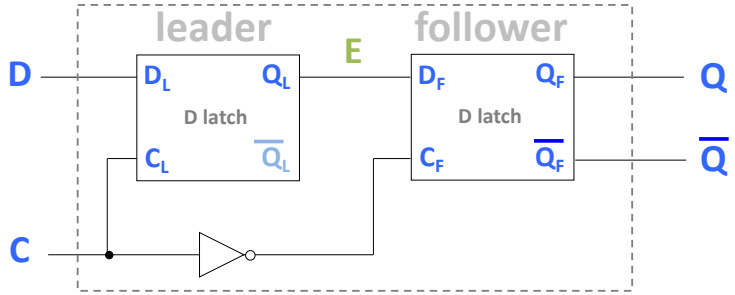
A clock controls when to update
a sequential logic element's state.

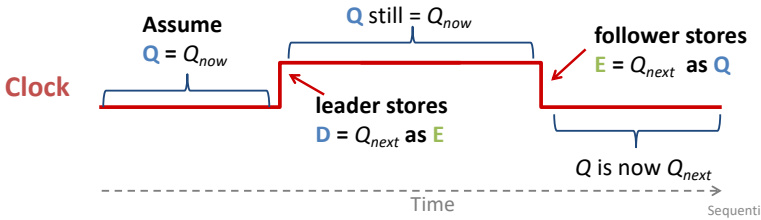
Synchronous systems

Inputs to state elements must be valid on active clock edge.



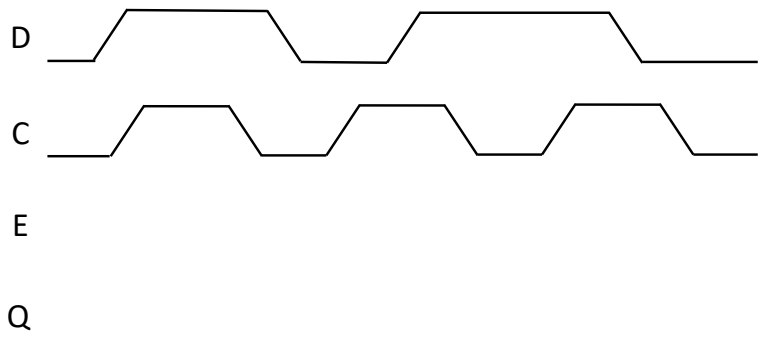
D flip-flop with falling-edge trigger





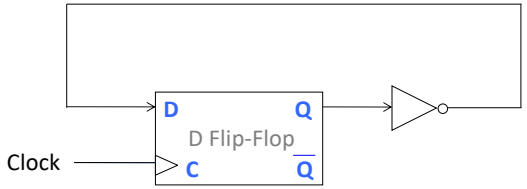
Time matters!

ex



Assume Q and E have an initial state of 0

Reading and writing in the same cycle



Assume Q is initially 0.

Moral: It's OK to use the current output Q of a flip-flop as part of the the next data input D to the same flip-flop.

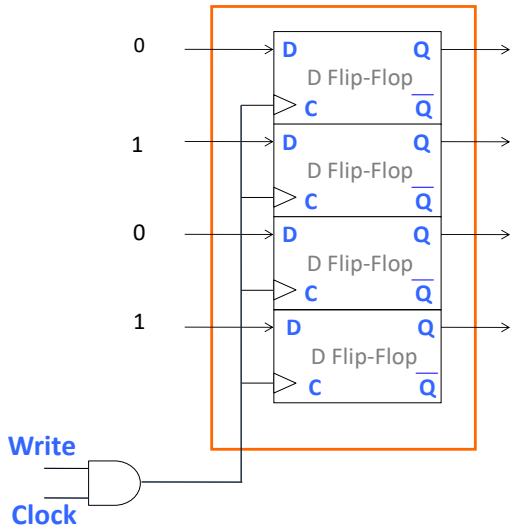
D flip-flop = one bit of storage

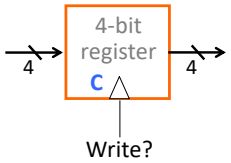
The bit value of D when C has a falling edge is remembered at Q until the next falling edge of C.

A 1-nybble* register

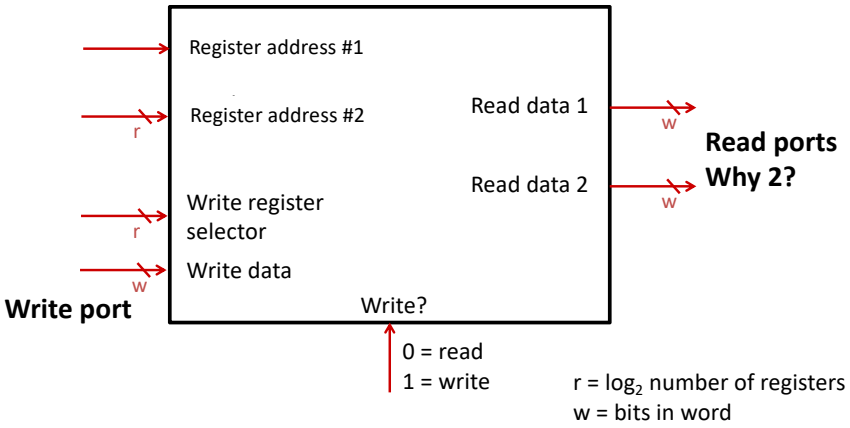
(a 4-bit hardware storage cell)

*Half a byte!



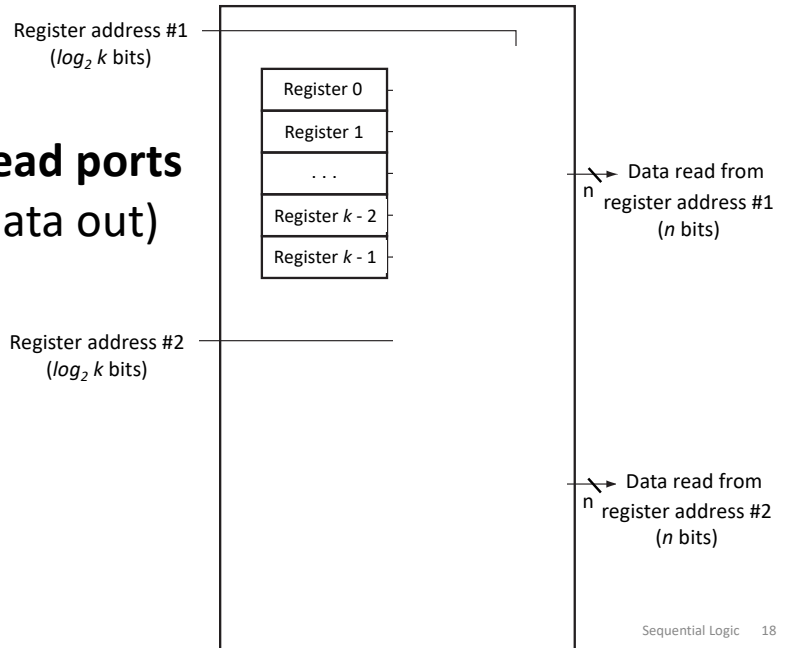


Register file

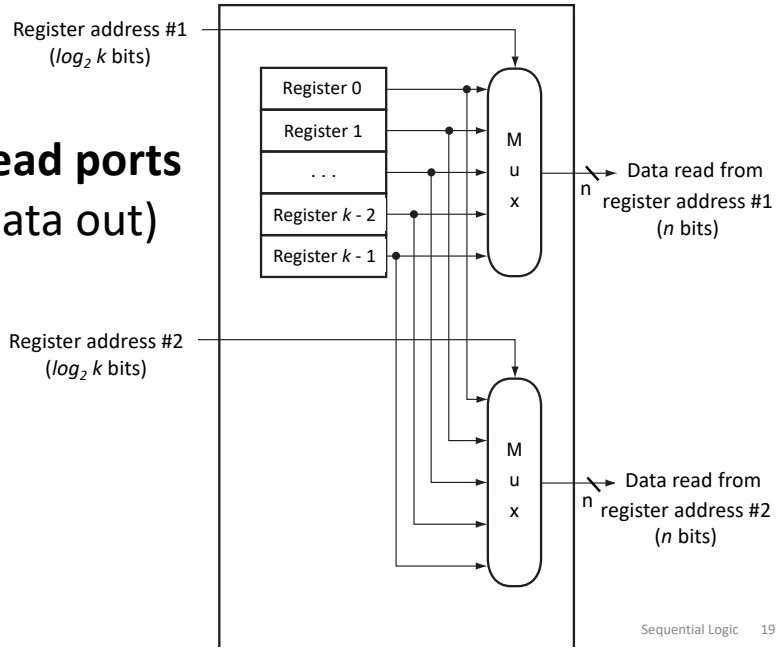


Array of registers, with register selectors, write/read control, input port for writing data, output ports for reading data.

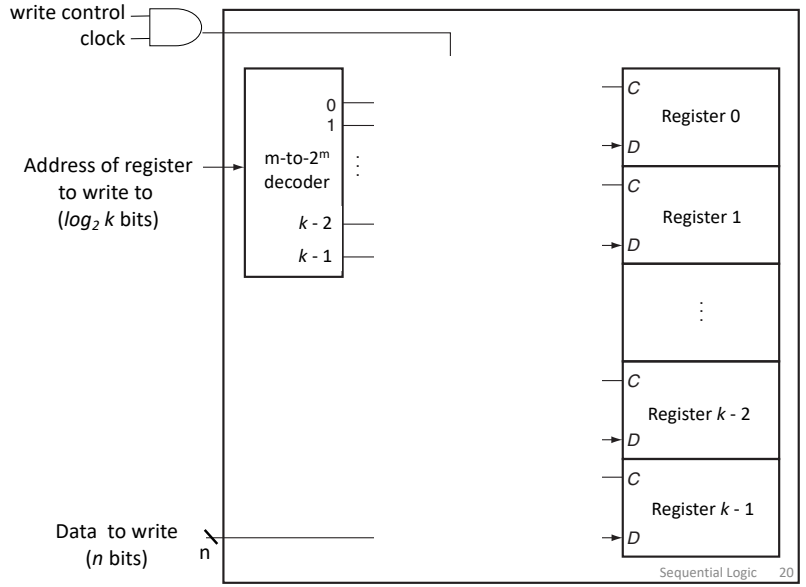
Read ports (data out)



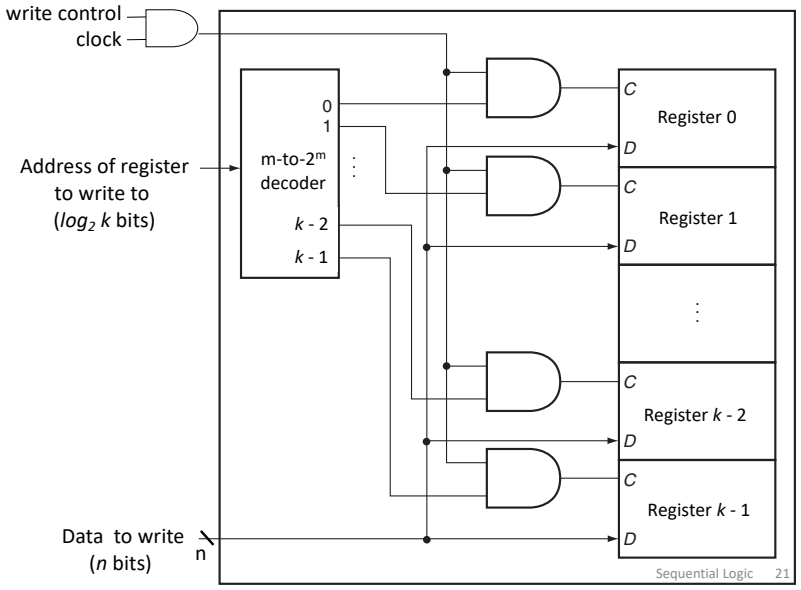
Read ports (data out)



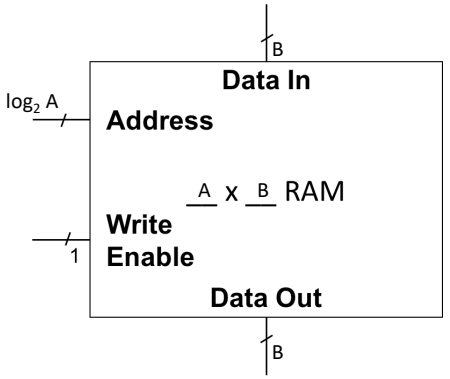
Write port (data in)



Write port (data in)



RAM (Random Access Memory)



- A is number of words in RAM
- Specify the desired word by an address of size $\log_2$ A
- B is the width of each word (in bits)

Similar to register file, except…

16 x 4 RAM