CS2002 Logic Lecture 7

Sequential Circuits

Ian Gent

Last time

- Combinational Circuits
- Circuits for Arithmetic
  - half adders and full adders
  - n-bit ripple adders
- Circuits for other activities in computers

This Time

- Sequential Logic Circuits
- Interlude: LogicSim for simulating circuits

Combinational vs Sequential Circuits

Combinational vs Sequential Circuits

Combinational Circuits

- Everything we've seen so far
- Data flows from input to output
- Outputs are fully determined by **current** inputs
- After stabilisation
  - All you need to know is the corresponding **truth table**

Combinational vs Sequential Circuits

Combinational Circuits

- Everything we've seen so far
- Data flows from input to output
- Outputs are fully determined by **current** inputs
- After stabilisation
  - All you need to know is the corresponding **truth table**

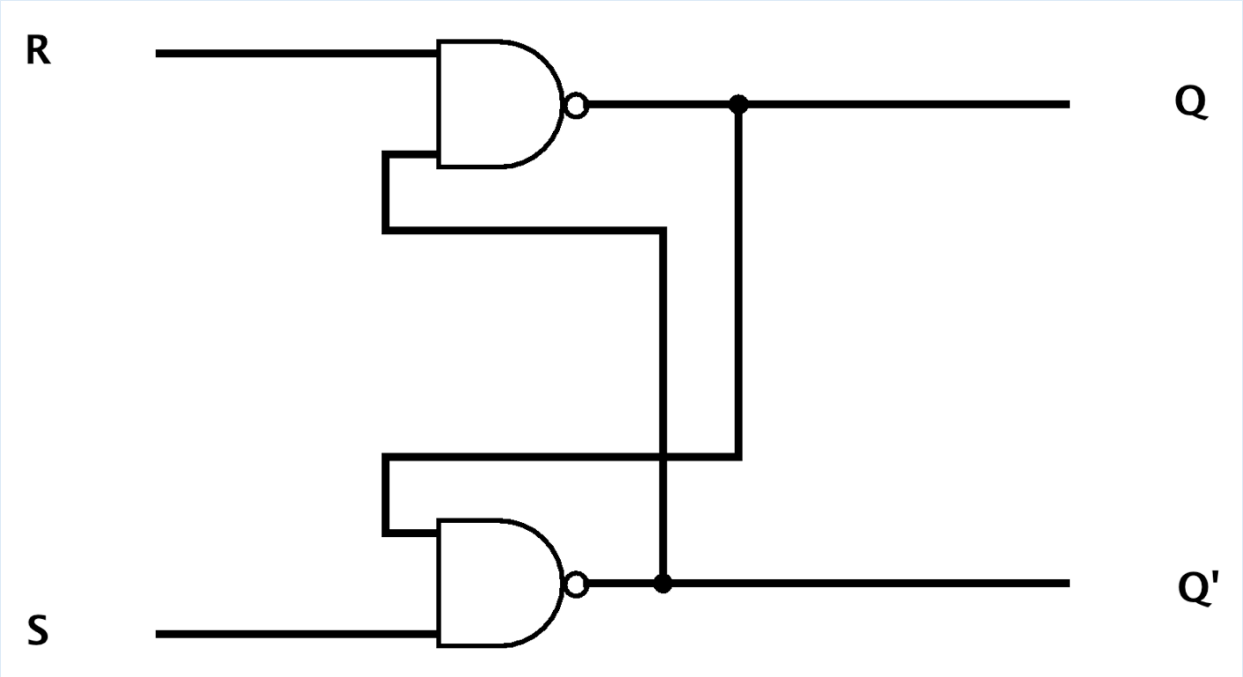
Sequential Circuits

- May have feedback loops
- Circuit has **state**
  - future behaviour determined by **past** state and current inputs
- more complex to model
- needed to make interesting computers

Feedback in Circuits

- Let’s see a simple example of feedback
- Feeding back the output of gates to their input
- What happens?

Feedback Example



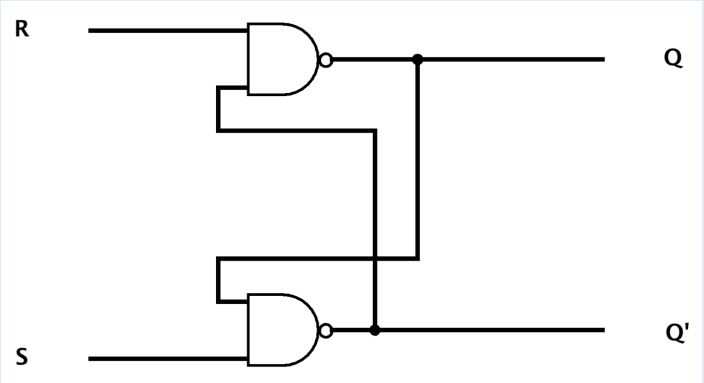
What Happens? (1)

Set R=0, S=0
Q=Q'=1
Q and Q' feedback to NANDs

Now set R=1
Q = 1 NAND 1 = 0
Q'= 0 NAND 0 = 1

Now set S = 1
Nothing changes

Result: R=S=1, Q=0, Q'=1



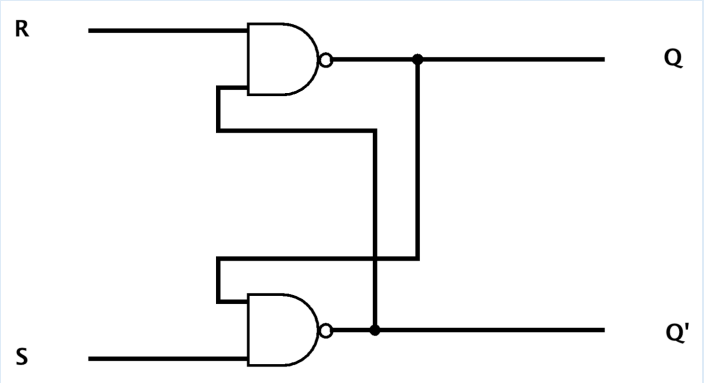
What Happens? (2)

Set R=0, S=0
Q=Q'=1
Q and Q' feedback to NANDs

Now set S=1
Q' = 1 NAND 1 = 0
Q = 0 NAND 0 = 1

Now set R=1
Nothing changes

Result: R=S=1, Q=1, Q'=0



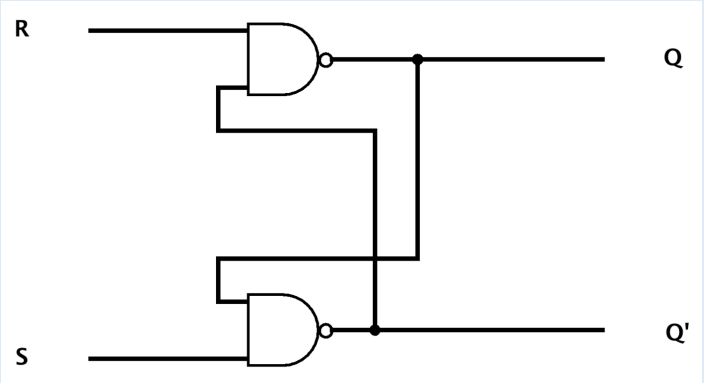
What Happened?

If we set R=1 first then S=1 we get
Result: R=S=1, Q=0, Q'=1

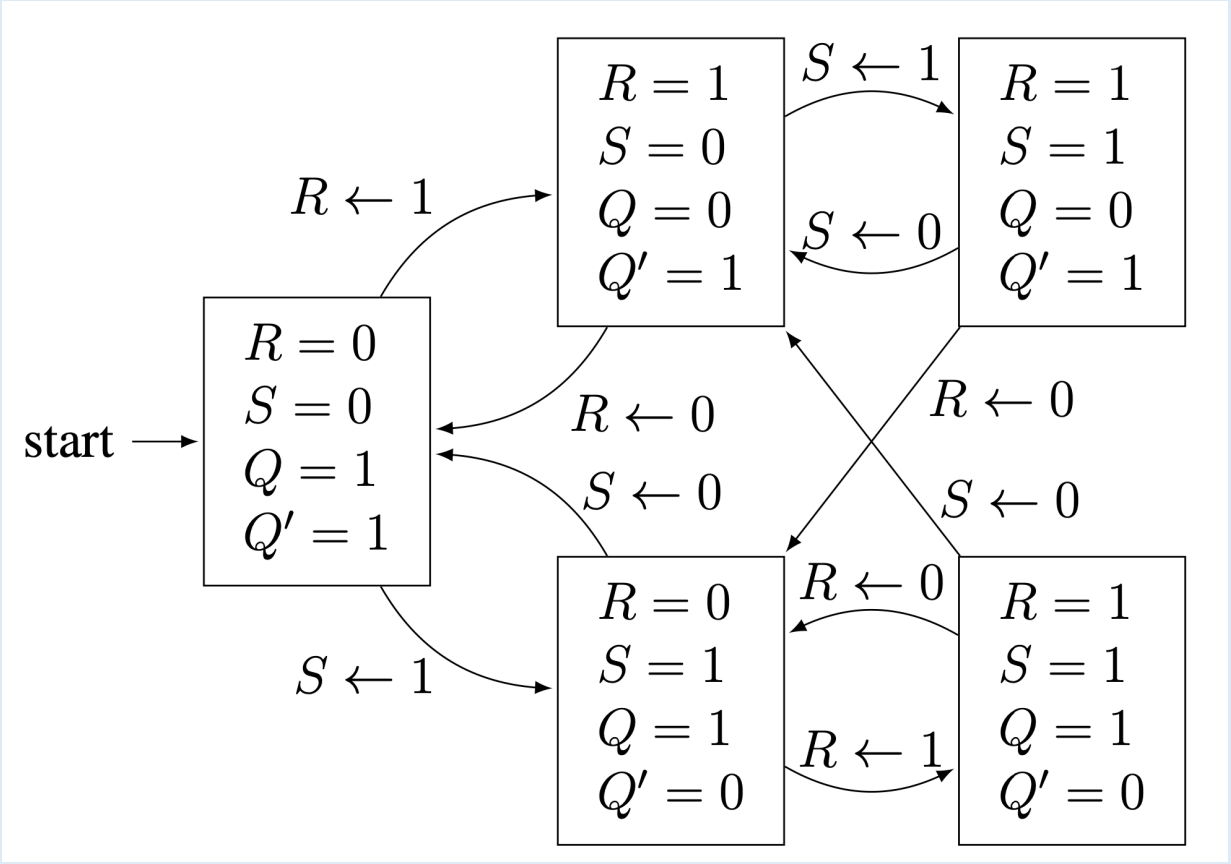
If we set S=1 first then R=1 we get
Result: R=S=1, Q=1, Q'=0

For the same inputs the result depends on the sequence of inputs over time

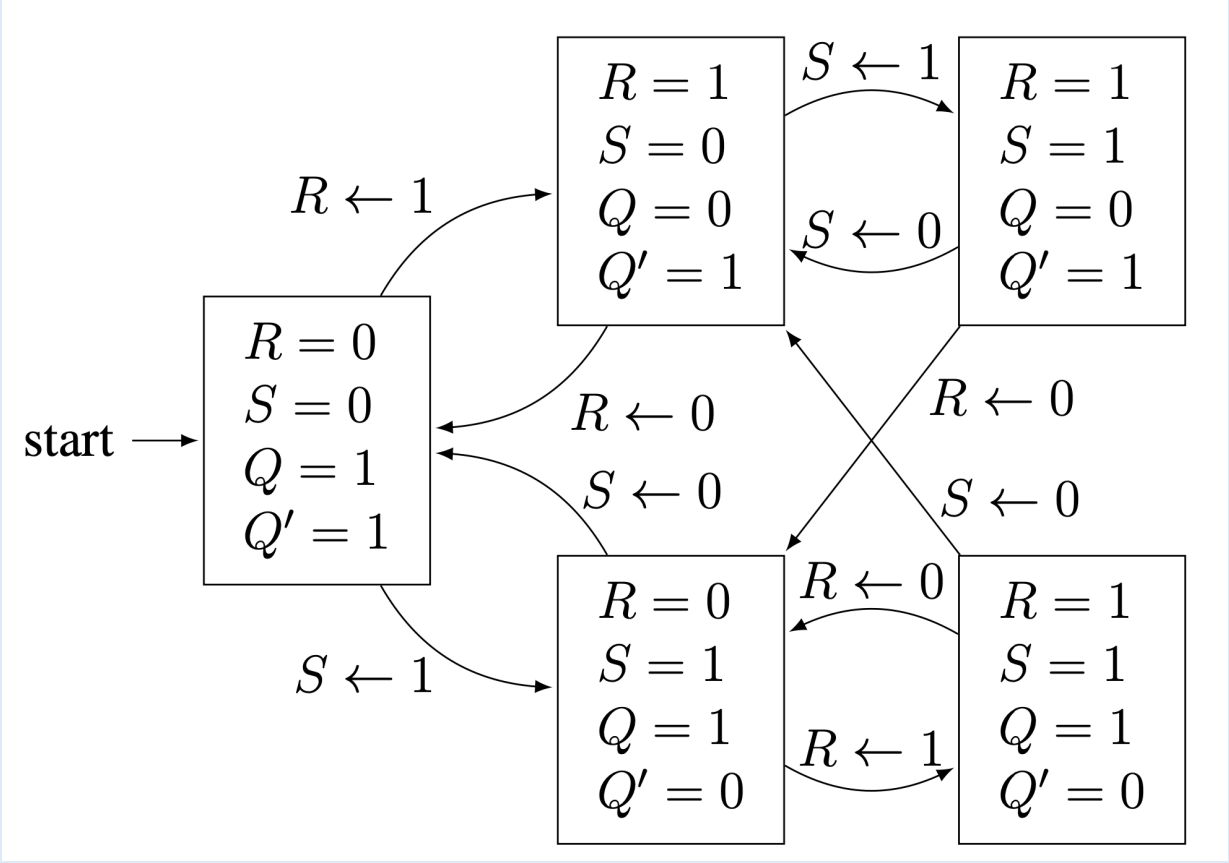
This is a sequential circuit



What Can Happen?



What Can Happen?

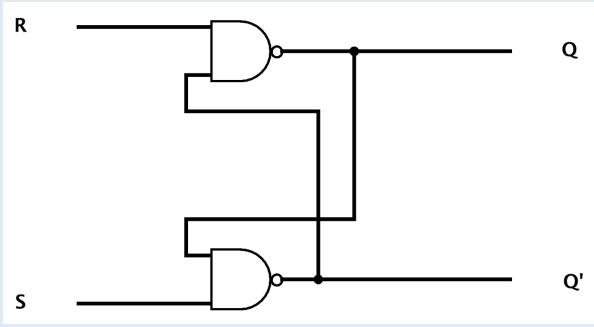


Note:
This assumes something

R, S not allowed to
Change simultaneously

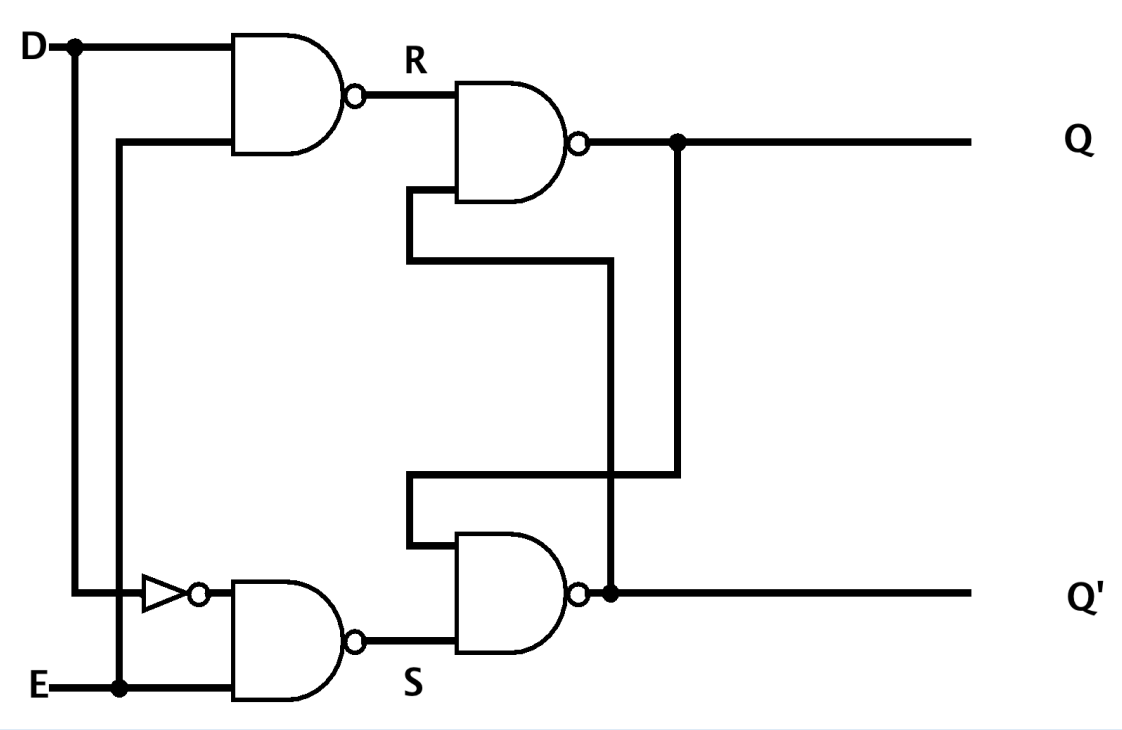
RS NAND Latch

R	S	Q	Q’
0	0	1	1
0	1	1	0
1	0	0	1
1	1	?	?



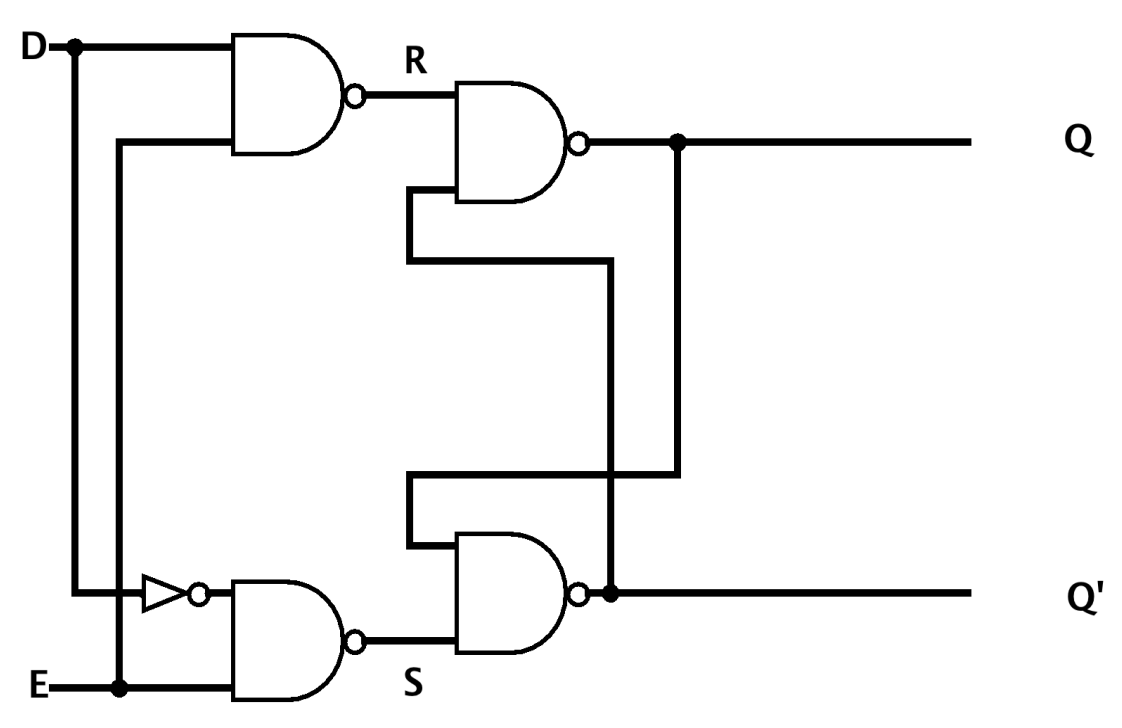
- ?s Depend on whether R or S got to 1 first
- Unstable if we switch R and S from 0 to 1 together
- Stable with Q = 0, Q’ = 1 or vice versa

Memory Example: Unclocked D Latch



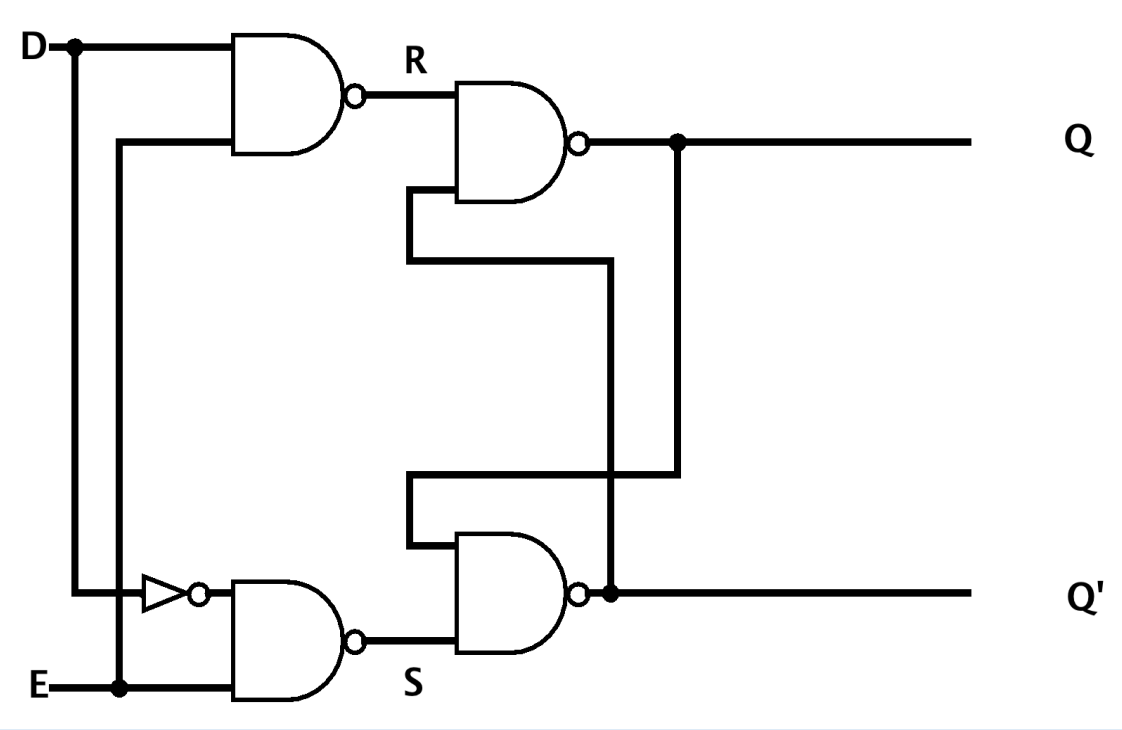
- Called an (unclocked) D latch
- We have conditioning logic on the left
- Feeds into RS NAND Latch on Right
- E stands for “Enable”
- But what does E enable?

Memory Example: Unclocked D Latch



- But what does E enable?
- E = 0
  - R = S = 1
  - Q fixed
  - D ignored
- E = 1
  - R = ¬D, S = D
  - Q = S = D
- R and S never both 0
- Note that unlike raw RS NAND Latch, no state is undefined

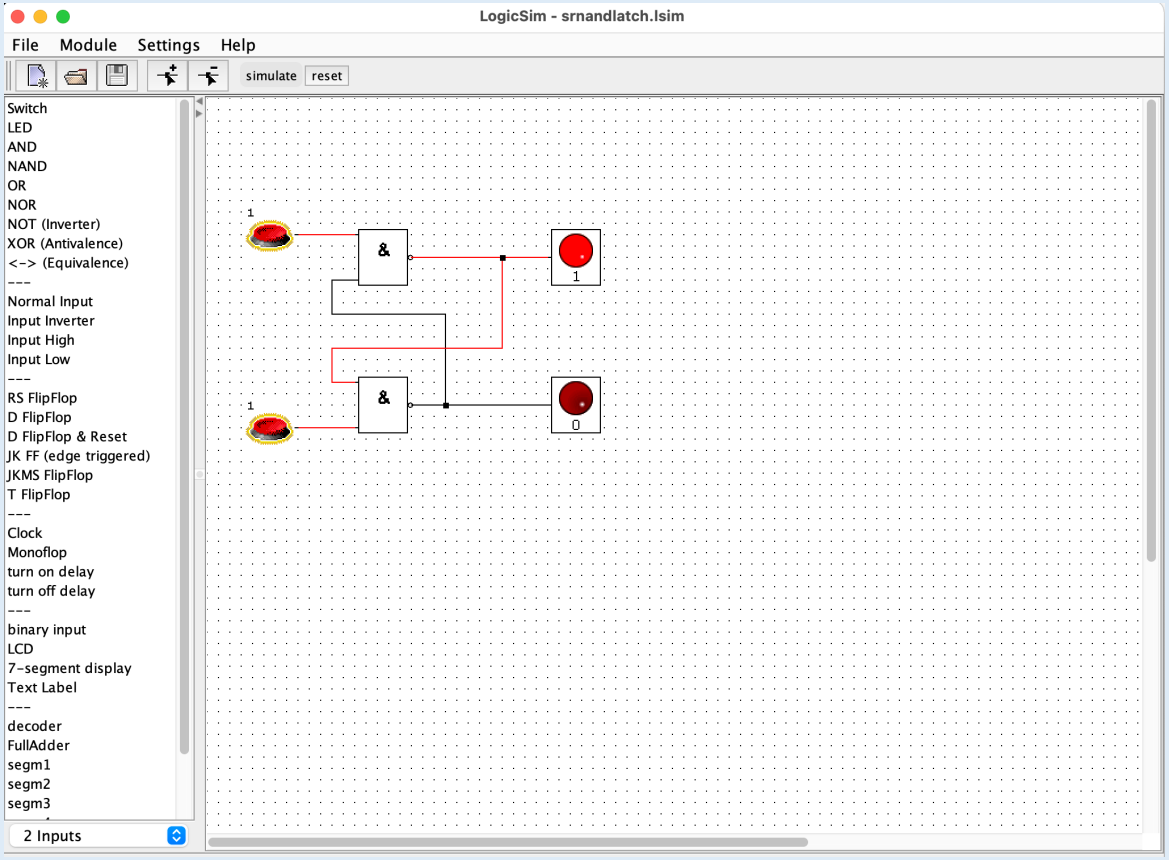
Memory Example: Unclocked D Latch



- **E** is write-**E**nable
- **D** is **D**ata
- Output is value of D last time E was 1
- Basic bit of memory

Interlude: LogicSim

Interlude: LogicSim



Useful tool for exploring circuit Behaviour

Runs from java

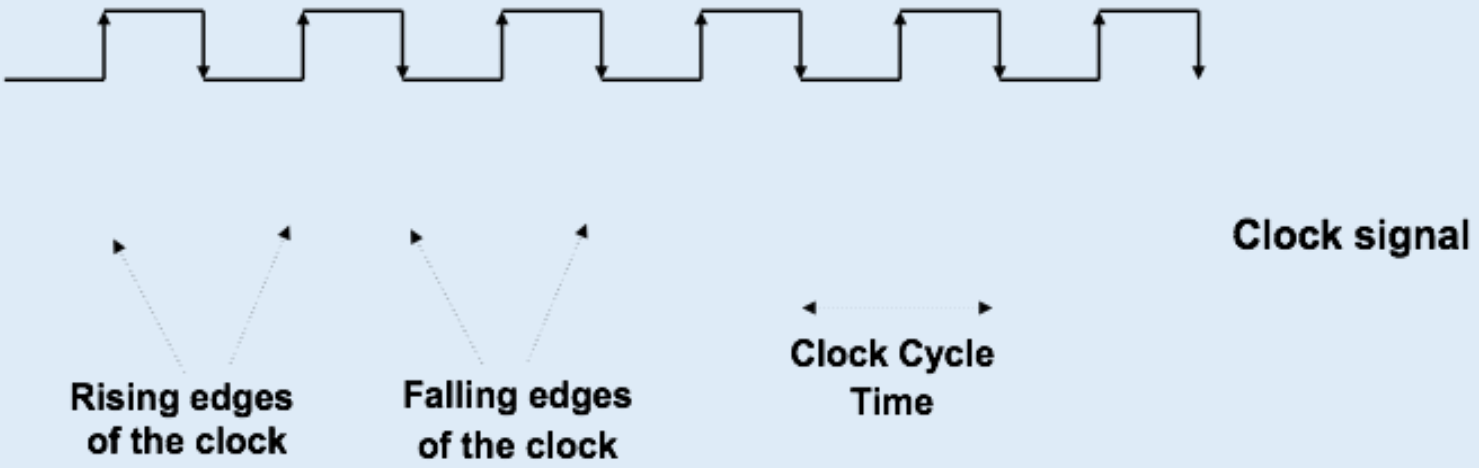
Available on Studres
under “Exercises”

Video by Mark-Jan Nederhof
Gives you a quick guide

Feel free to use other logic simulators

A Clock Signal

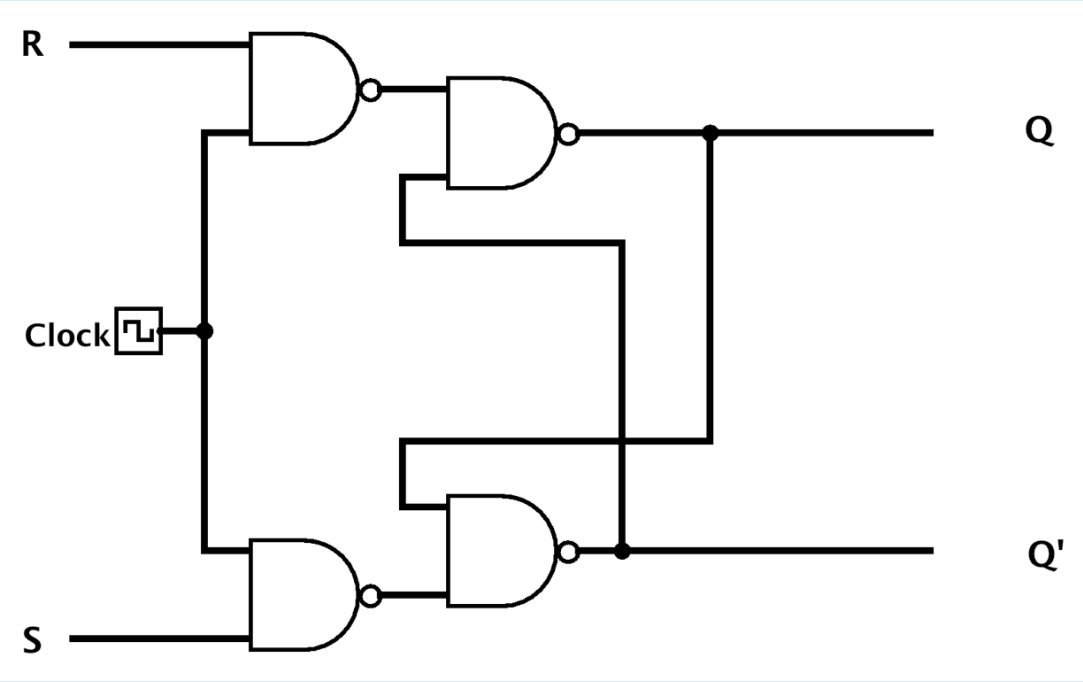
- The clock signal changes regularly from 0 to 1 and back again
- Clock cycle is time to go from 0 to 1 and back again



CPU Clocks

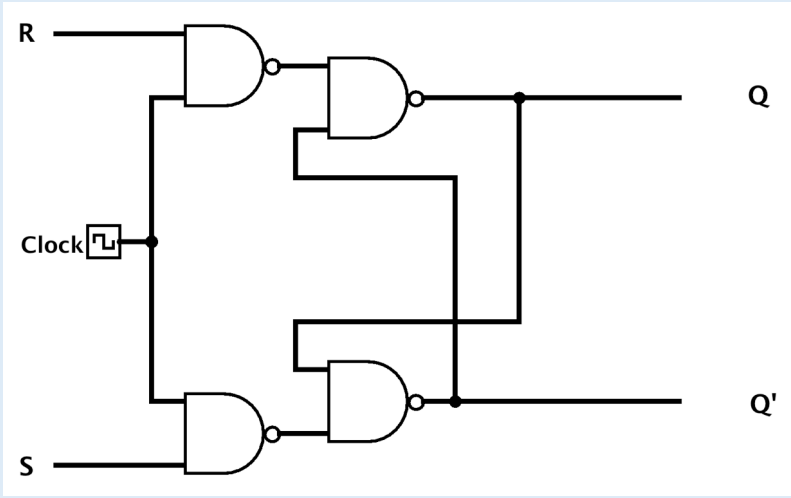
- CPU has a clock signal distributed to all parts of it
  - care needed to keep it in sync
  - also consumes a lot of power
- State of CPU is defined by design on edges of the clock only
  - each state follows logically from the one before
  - latches etc. are used to make this work
  - clock cycle has to be long enough to allow for all gate delays
- 1951 Ferranti Mark 1 - 1.2ms (milliseconds)
- 1976 Cray 1 - clock cycle 12.5ns, Z80 400ns (nanoseconds)
- 2005 Pentium IV, clock cycle 260ps (picoseconds)
- 2021 AMD's, Ryzen TR 3990X clock cycle (in boost mode) **232ps**
  - Note lack of speedup in 15 years though the TR3990X has 64 cores!

Clocked Example



Clocked RS NAND Latch

- RS NAND Latch guarded by NANDS and clock
- When Clock is 0 output of first NANDS must be 1
  - Q and Q’ will not change
  - A simple form of memory
- When clock goes to 1
  - R and S can change Q/Q’



Multistage computation example

Cycle	Clock value	Stage	Type	Notes
17	1	1	Combinational	
17	1	2	Latch	
17	0	3	Combinational	stage 3 runs using stabilized results from stage 1
17	0	4	Latch	
18	1	1	Combinational	stage 1 runs may use results from stage 4 of previous cycle
18	1	2	Latch	

Basic Concepts of Sequential Circuits

- A sequential circuit defines its output in terms of both its current inputs and its previous inputs
- To remember previous inputs, we need a storage element – e.g. flip-flop, latch
- The state of the latch is a function of the previous inputs to the circuit
- We need event ordering
- **Synchronous** sequential circuits use clocks to order events - activities synchronise with clock signals
- We pretend time is discrete: hence the notion of a clock “tick”

Next Time

- Flip Flops