Homework

- Reading: Chapter 9
- Homework: Chapter 9, exercises 1–4 & 7
- Due Monday, October 1

COS 140: Foundations of Computer Science

Parallel Registers

Fall 2018

Computer Science Foundations

Problem

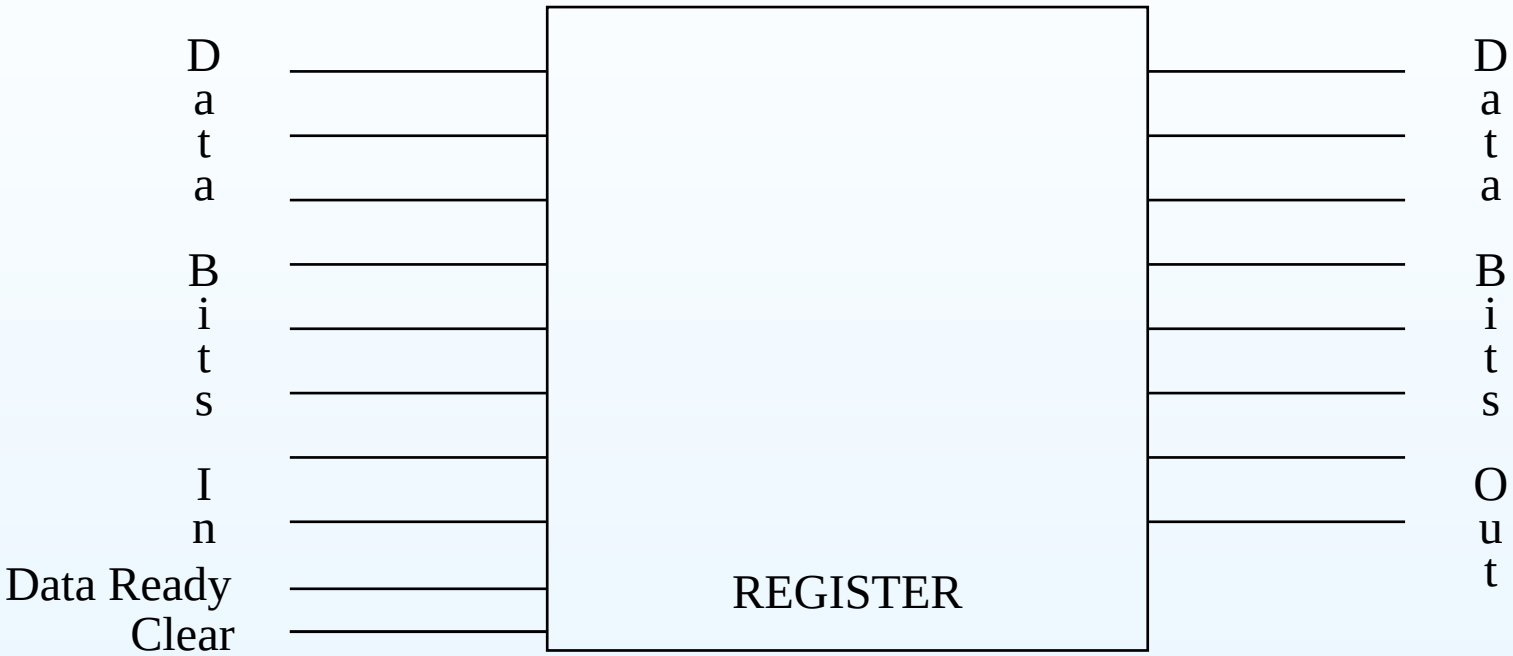
- The *central processing unit* (CPU) needs some *memory*
  - Place to store instructions being executed
  - Place to store instruction's *operands*
  - Place to store intermediate values, output of calculations
- This memory has to be extremely fast: faster than RAM

Computer Science Foundations

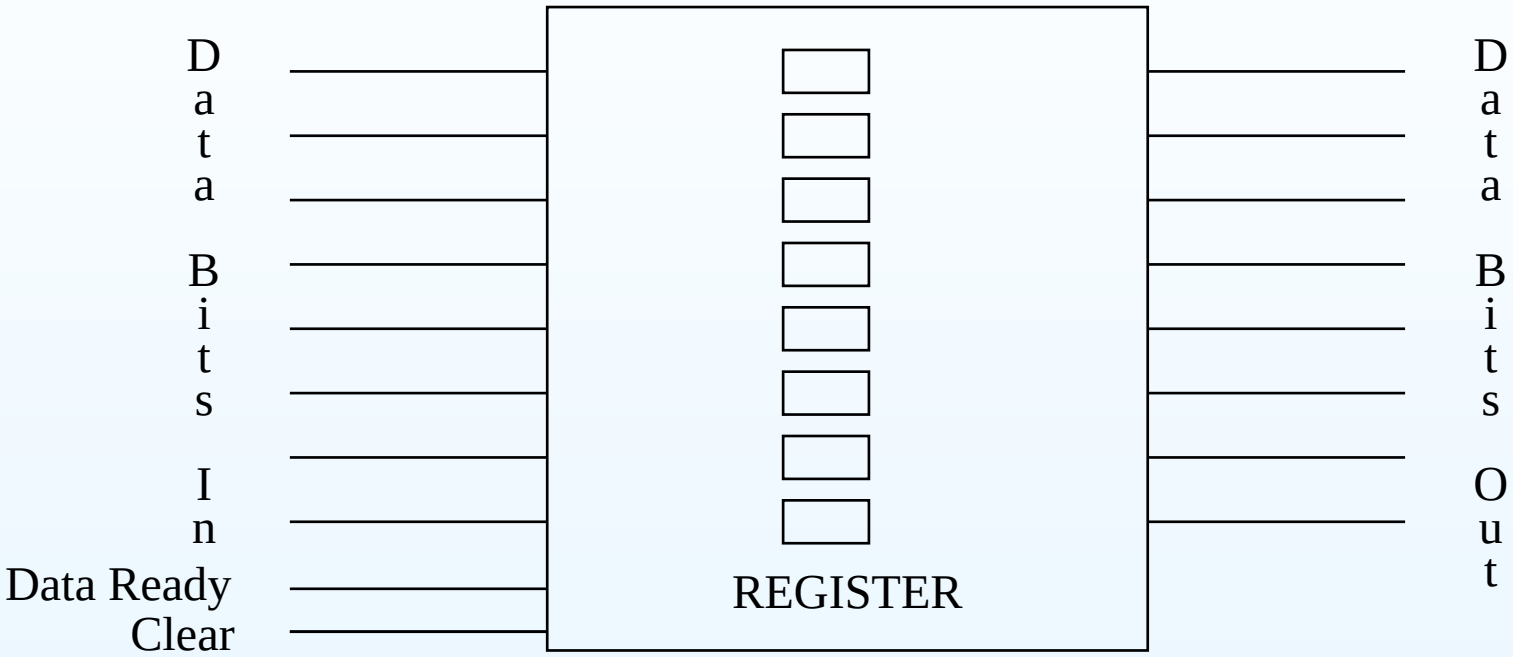
Solution: Parallel Registers

- Use *parallel registers*
- Very fast type of memory
- Stores several bits at a time, function together as a unit.
- Could be one chip or more than one chip used in parallel
- Most often: part of the CPU

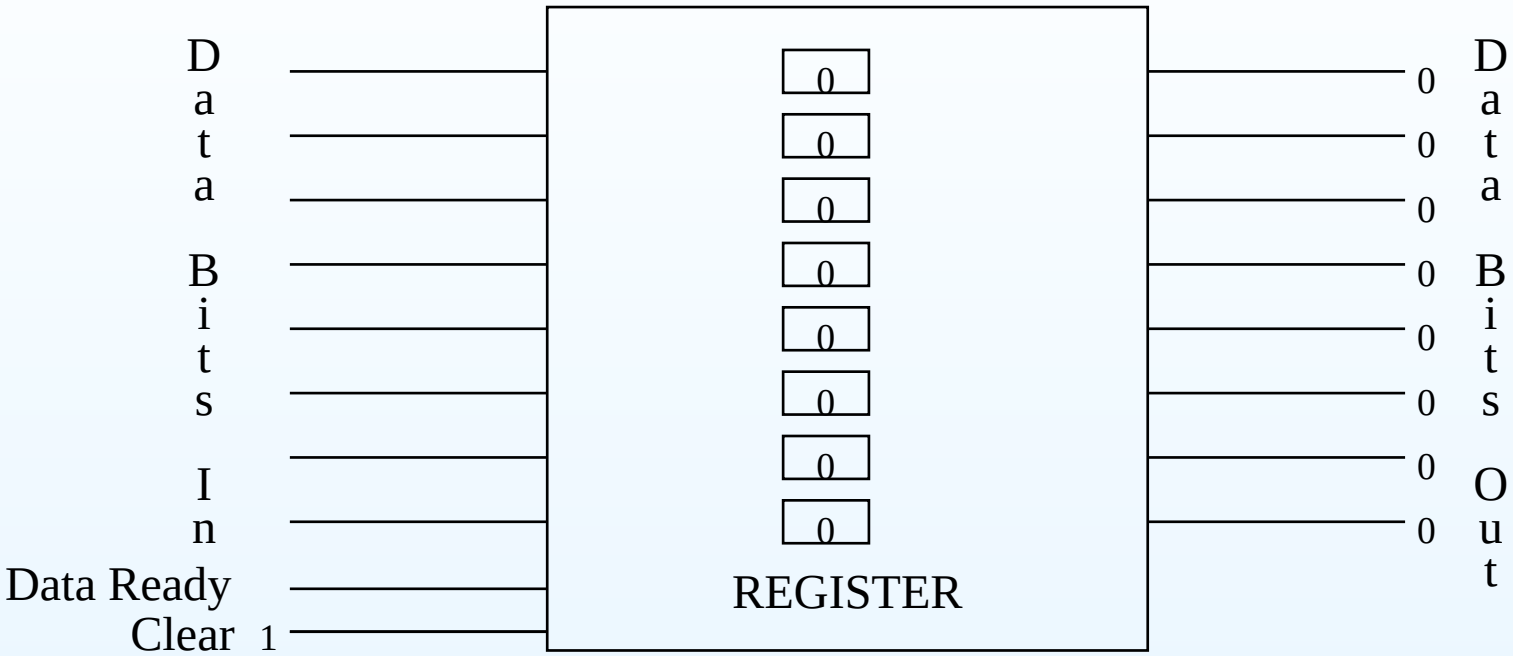
A Parallel Register



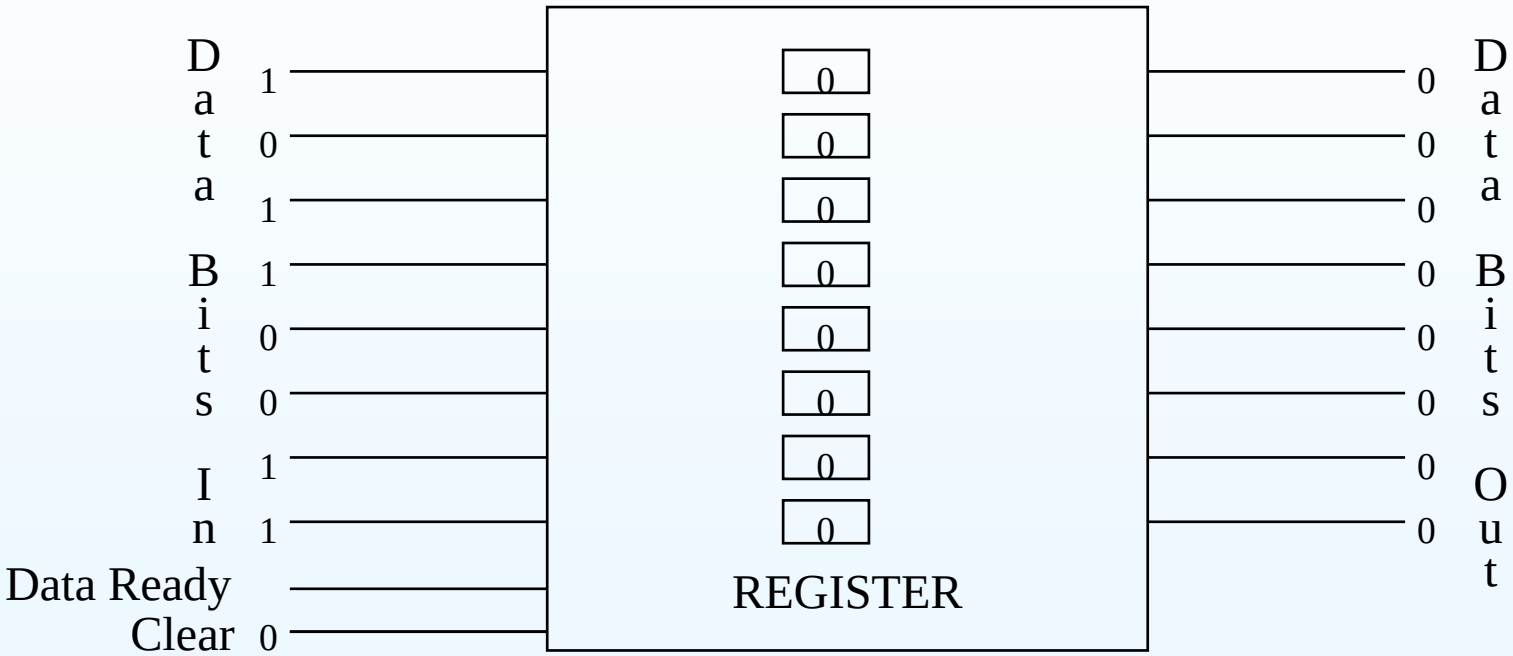
A Parallel Register



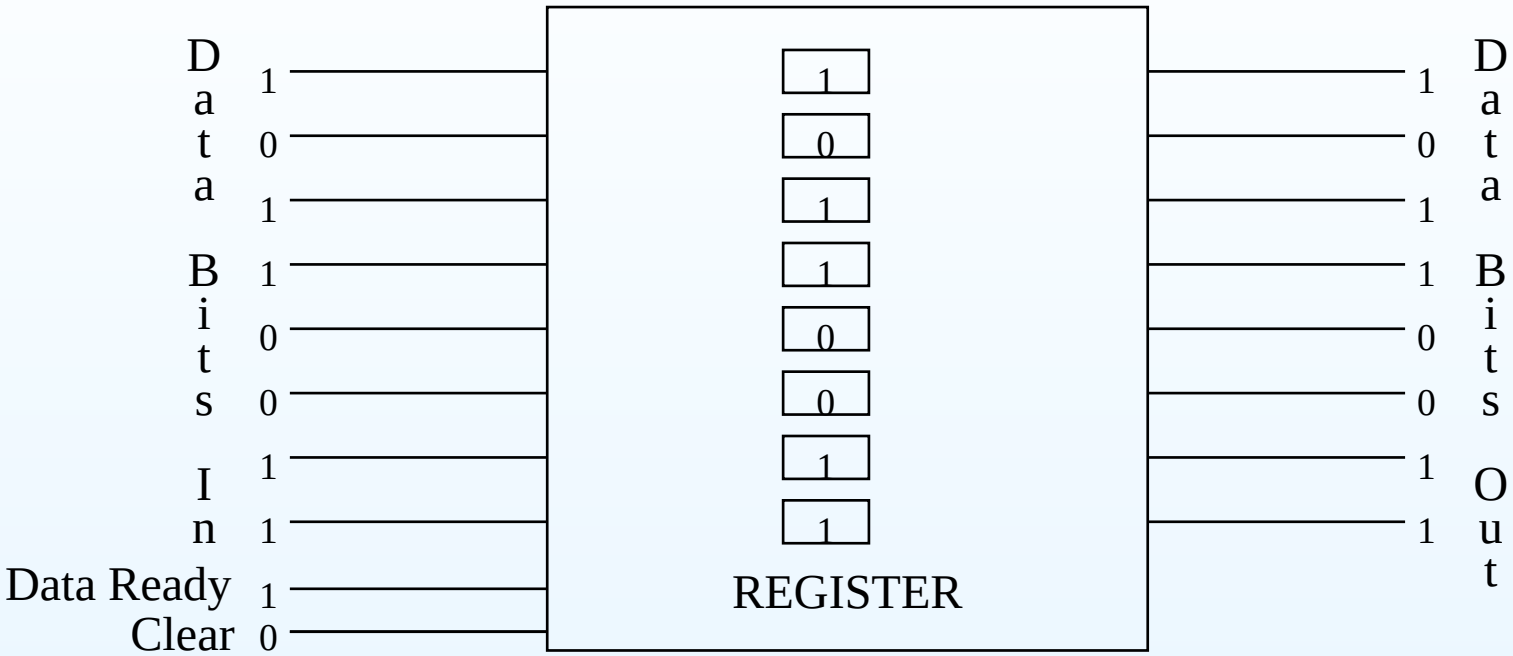
A Parallel Register



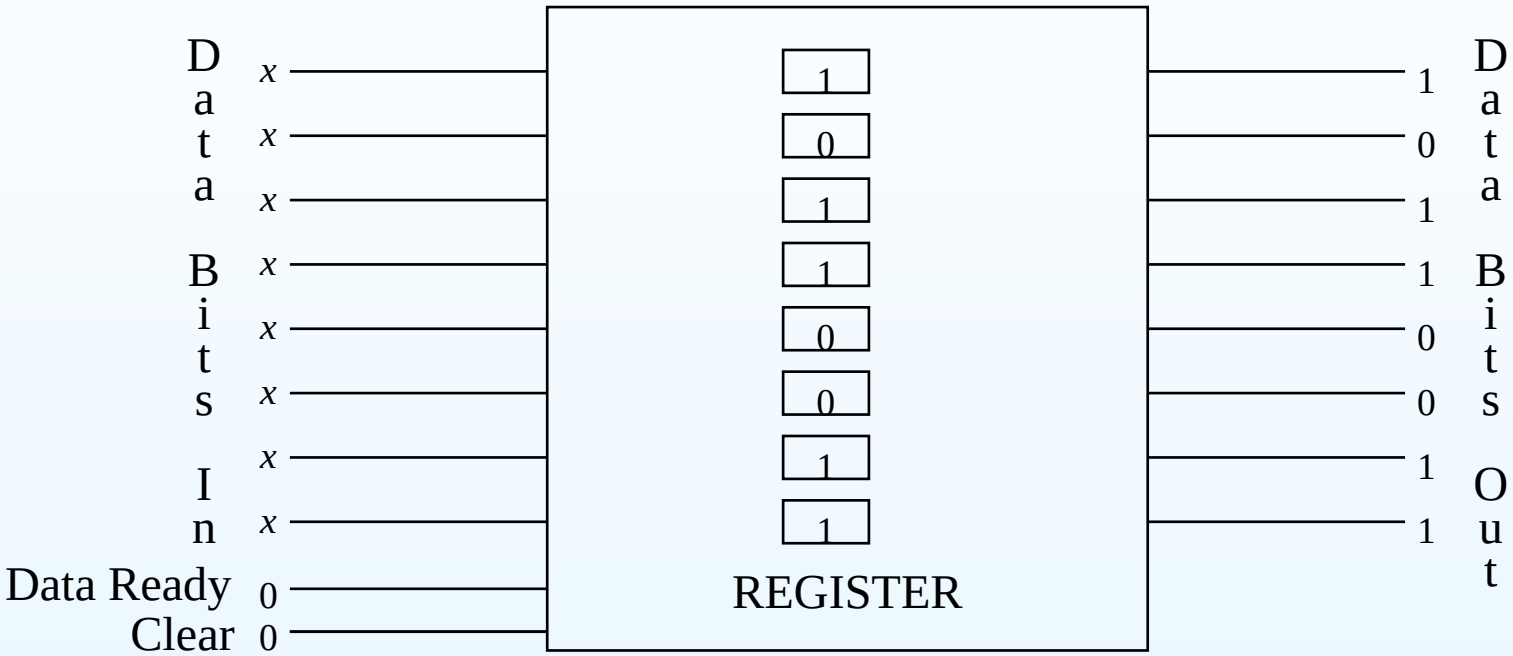
A Parallel Register



A Parallel Register



A Parallel Register



WANTED: A Circuit that Can Remember

- To be useful for memory, the circuit must:

WANTED: A Circuit that Can Remember

- To be useful for memory, the circuit must:
  - Be readable.

Computer Science Foundations

WANTED: A Circuit that Can Remember

- To be useful for memory, the circuit must:
  - Be readable.
  - Maintain the current data, unless its inputs tell it to change.

Computer Science Foundations

WANTED: A Circuit that Can Remember

- To be useful for memory, the circuit must:
  - Be readable.
  - Maintain the current data, unless its inputs tell it to change.
  - Allow the data to be changed.

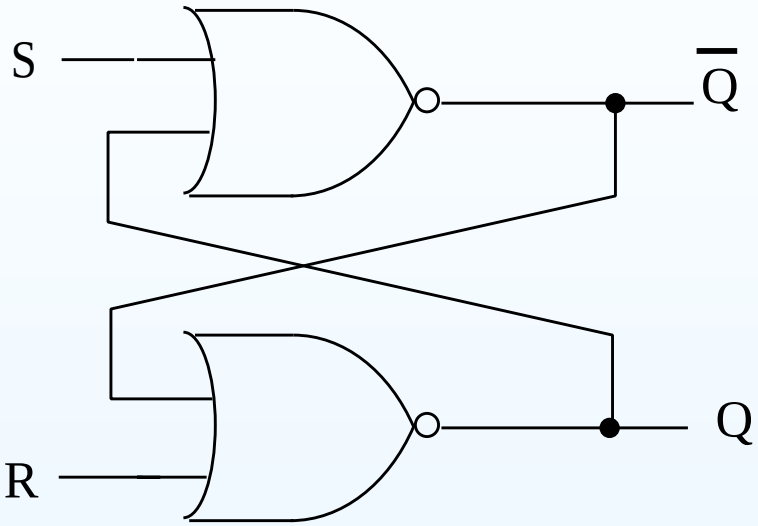
WANTED: A Circuit that Can Remember

- To be useful for memory, the circuit must:
  - Be readable.
  - Maintain the current data, unless its inputs tell it to change.
  - Allow the data to be changed.
- Combination circuits have outputs that are a function only of inputs, so no ability to maintain the current data.

Sequential Circuits

- Have state which can be 0 or 1.
- New state depends on inputs and previous state.
- Often have *clock* (strobe, enable) that allows input to enter the circuit only at particular times.
- Sequential circuits work for memory because
  - The current state can be easily read from its output (packaging often makes the state's complement available as well)
  - Some set of inputs (usually all 0's) cause it to maintain state
  - Other inputs can change the data

A Circuit that Can Maintain State: The S-R Latch



- Can be set (with S) and reset (with R)
- Maintains whatever state it's set to when inputs removed (set to 0)

Characteristic Table

- How to specify the behavior of a sequential circuit?

Characteristic Table

- How to specify the behavior of a sequential circuit?
- Truth table won't work: doesn't reference current state
- Instead, use a *characteristic table*; like a truth table, but:
  - Shows current state as well as inputs
  - Gives new state (which is output as well)
  - Note: Can have inputs for which there is no stable state, an undefined state, or a state that does not make sense

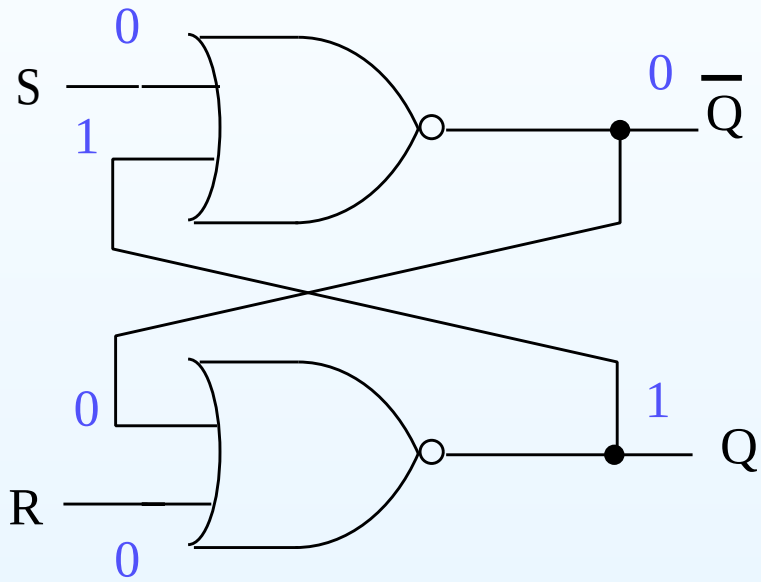
What Would We Expect?

What Would We Expect?

S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	-

How the S-R Latch Maintains State

Have feedback from the current state and its complement that act as input to the circuit.

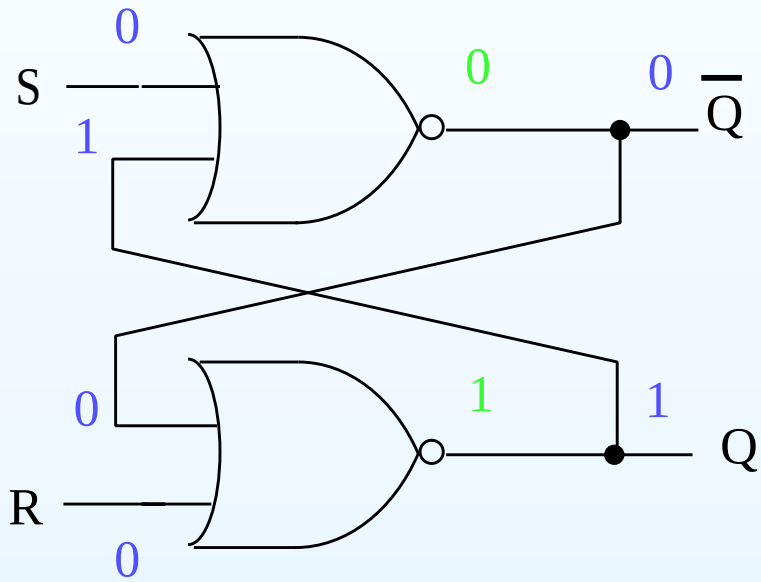


Suppose initial state is 1 and complement is 0. S and R are both 0 (neither set or reset).

Computer Science Foundations

How the S-R Latch Maintains State

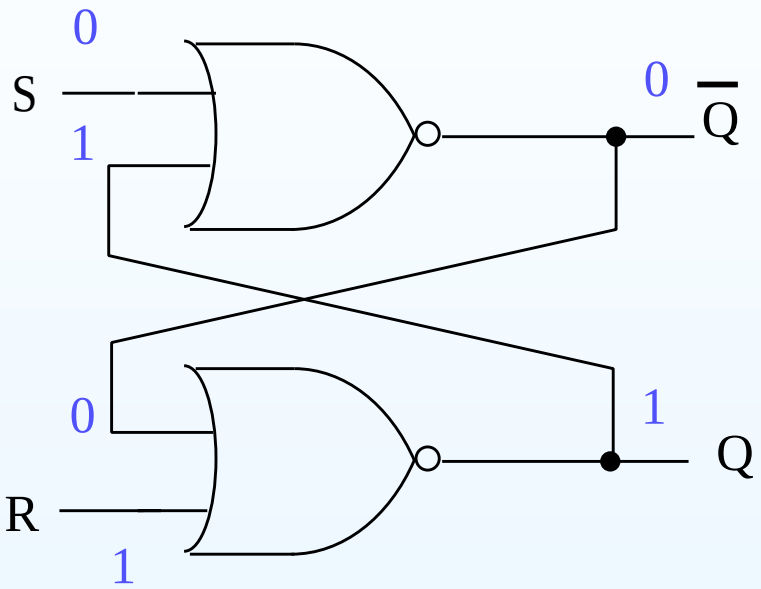
Have feedback from the current state and its complement that act as input to the circuit.



Suppose initial state is 1 and complement is 0. S and R are both 0 (neither set or reset).

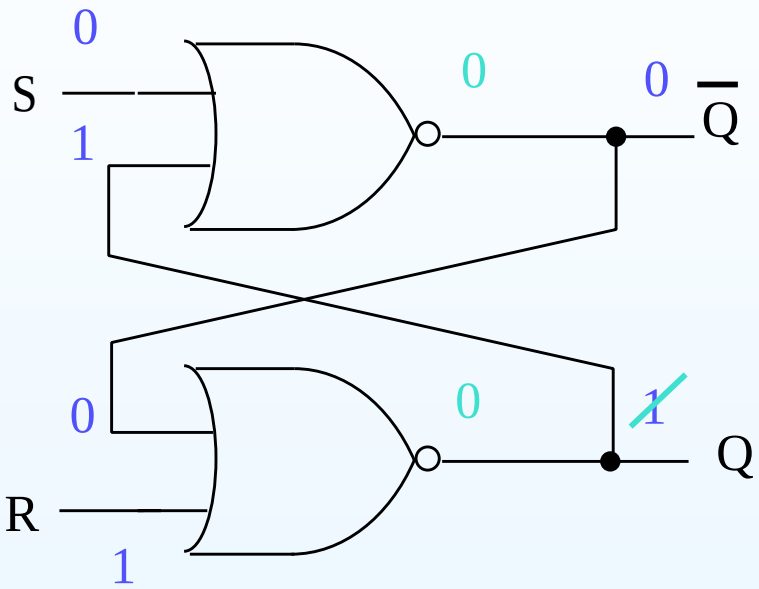
Computer Science Foundations

How the S-R Latch's State Is Reset to 0

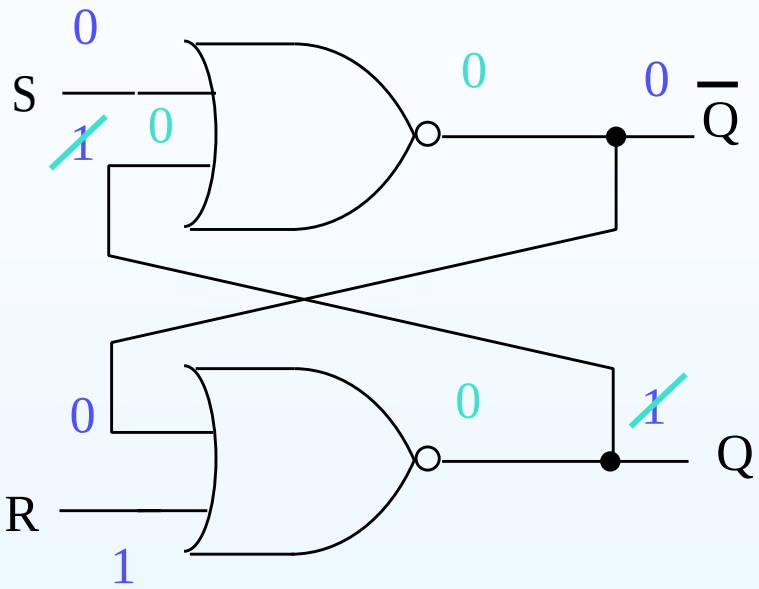


Computer Science Foundations

How the S-R Latch's State Is Reset to 0

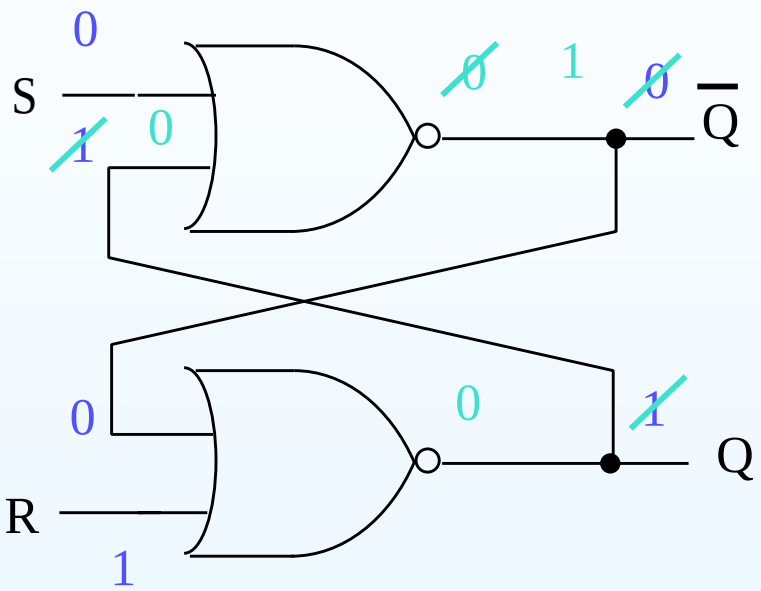


How the S-R Latch's State Is Reset to 0



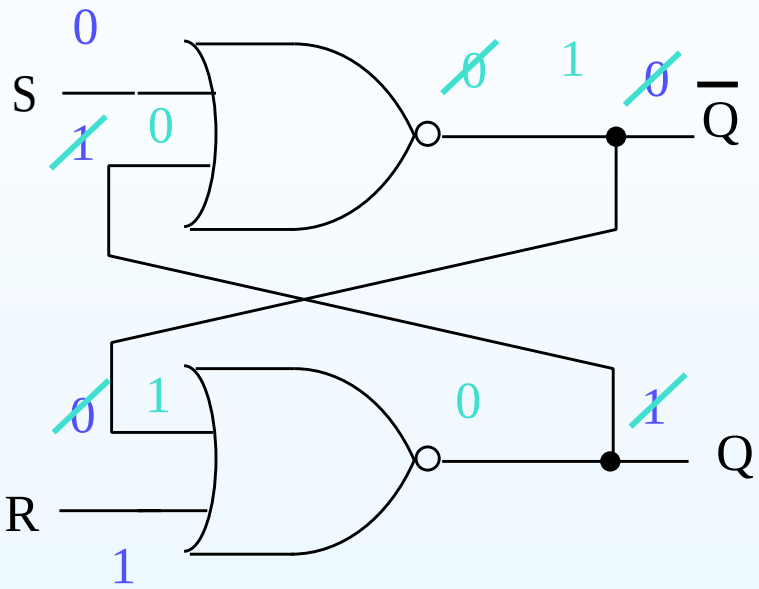
Computer Science Foundations

How the S-R Latch's State Is Reset to 0

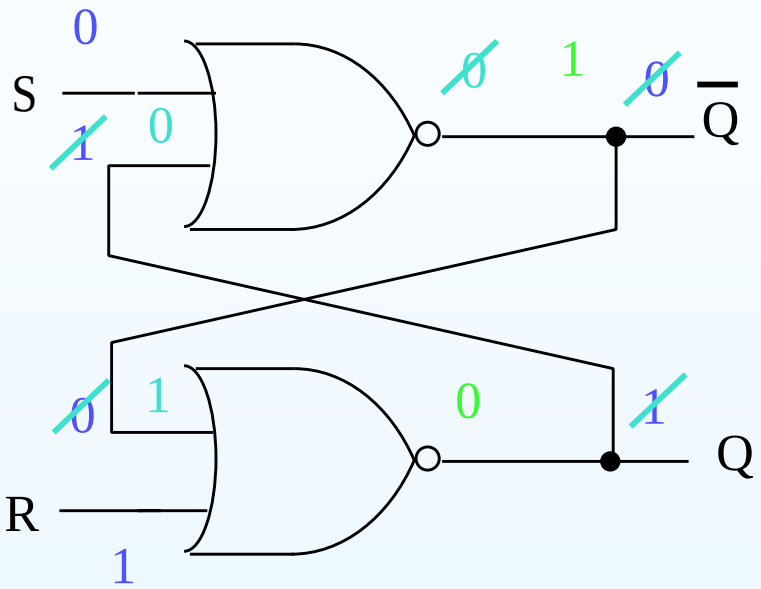


Computer Science Foundations

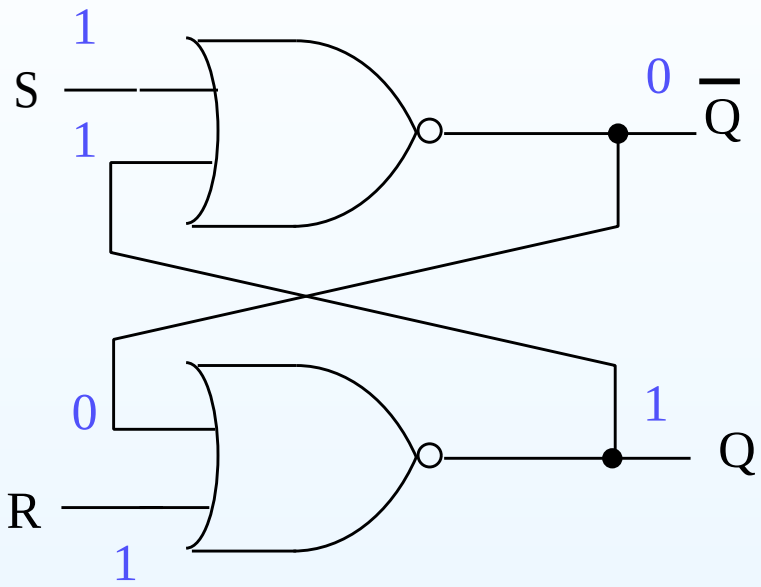
How the S-R Latch's State Is Reset to 0



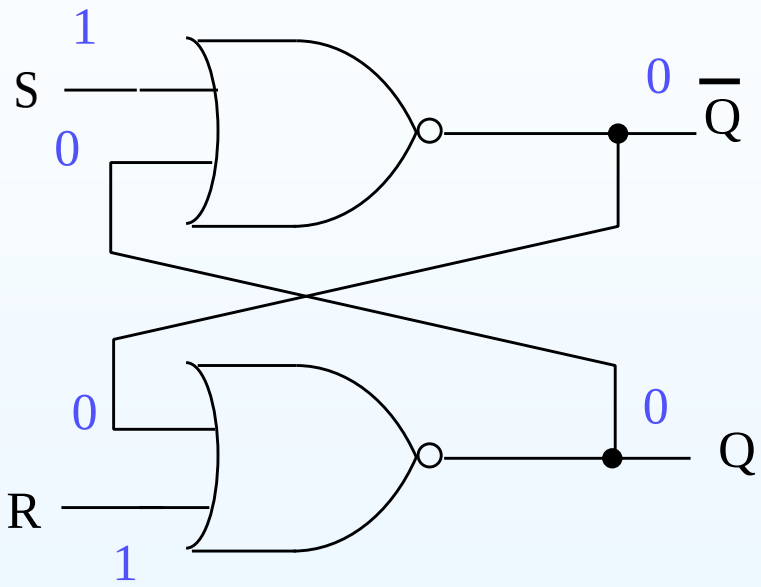
How the S-R Latch's State Is Reset to 0



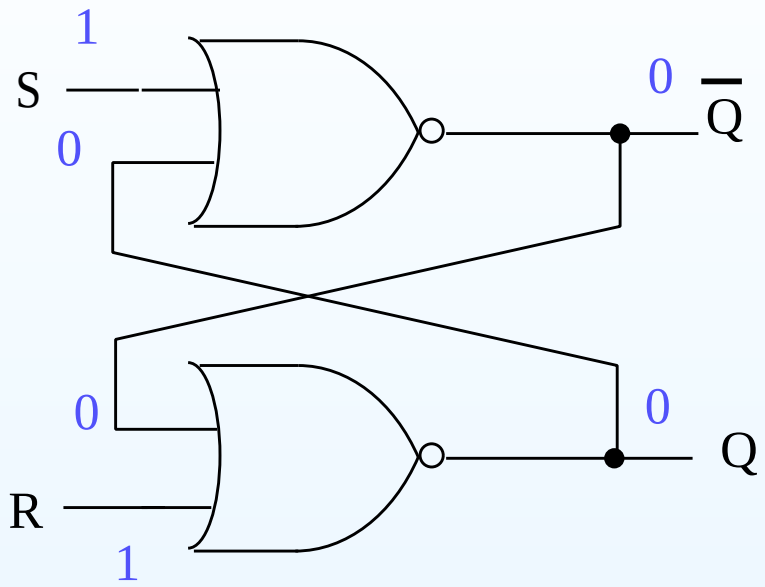
What Happens with 1 1 Input?



What Happens with 1 1 Input?



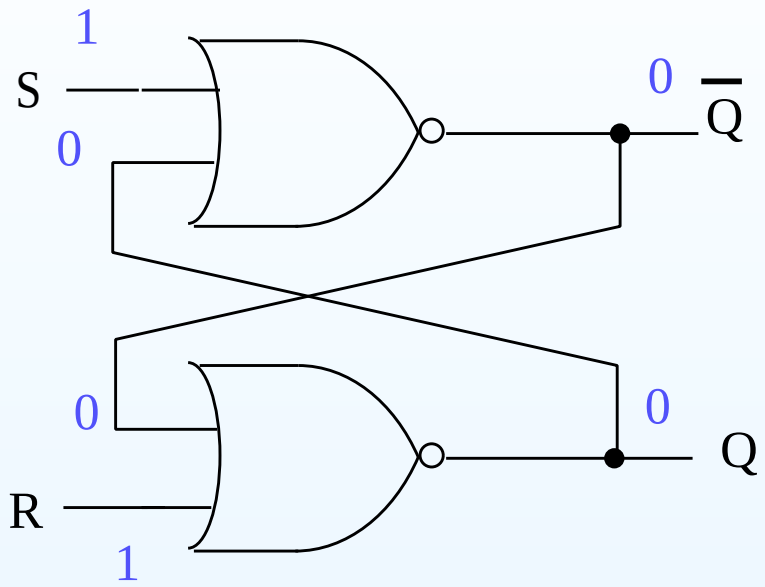
What Happens with 1 1 Input?



- State not reasonable: $Q = \overline{Q}$.

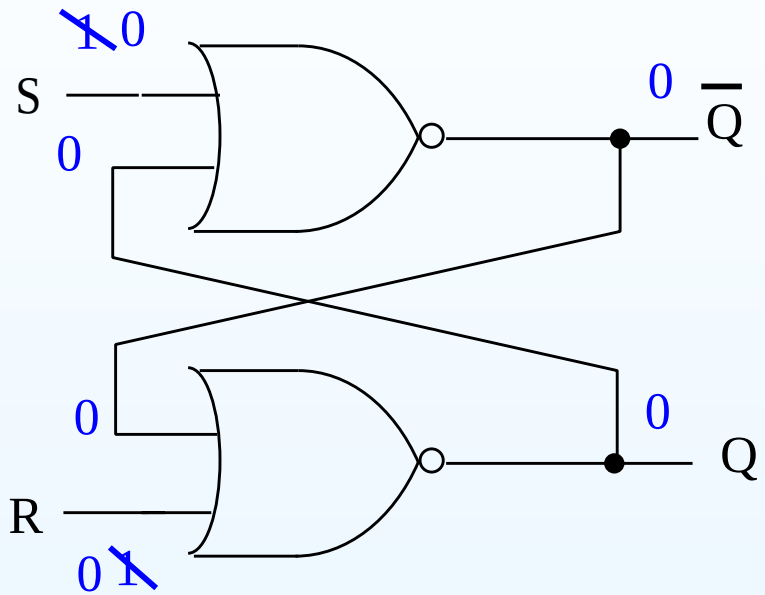
Computer Science Foundations

What Happens with 1 1 Input?



- State not reasonable: $Q = \overline{Q}$.
- What happens when $S = R = 0$?

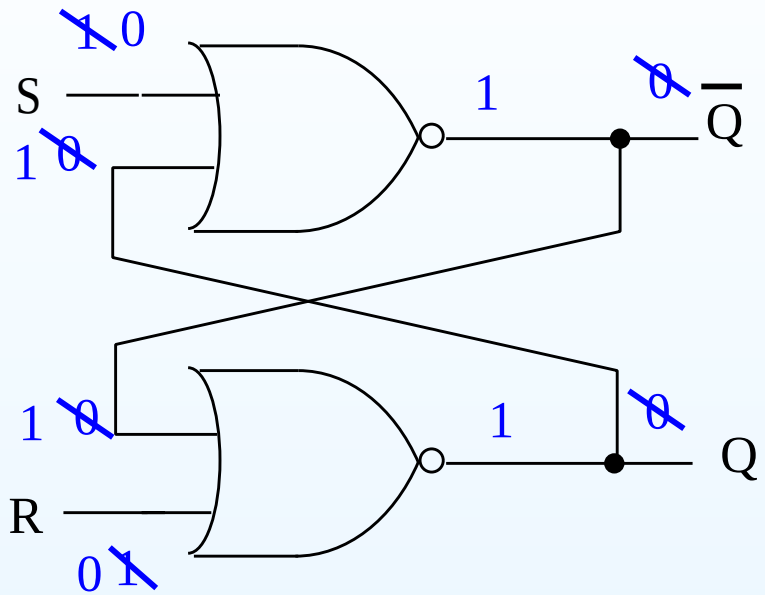
What Happens with 1 1 Input?



- State not reasonable: $Q = \overline{Q}$.
- What happens when $S = R = 0$?

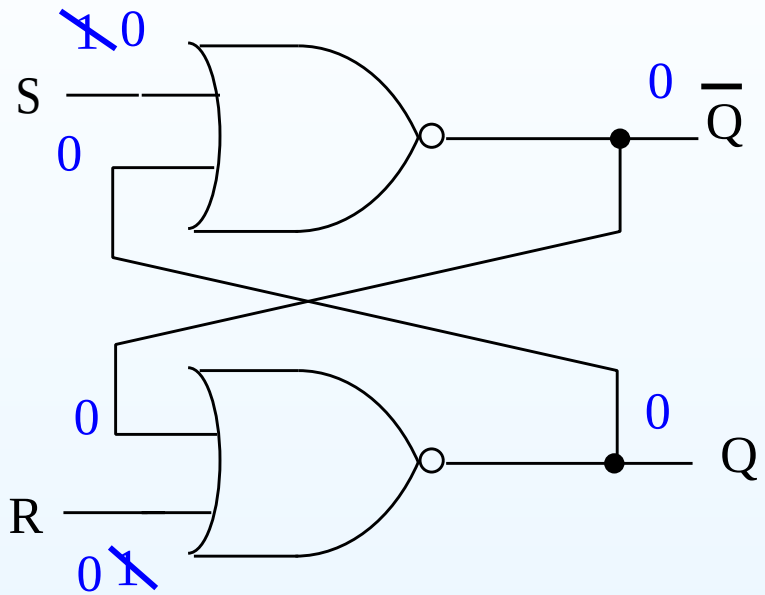
Computer Science Foundations

What Happens with 1 1 Input?



- State not reasonable: $Q = \overline{Q}$.
- What happens when $S = R = 0$?

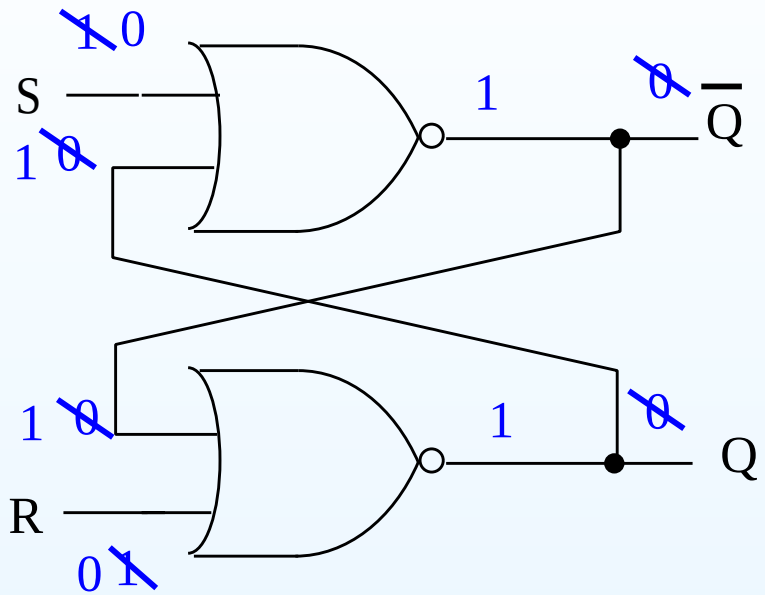
What Happens with 1 1 Input?



- State not reasonable: $Q = \overline{Q}$.
- What happens when $S = R = 0$?

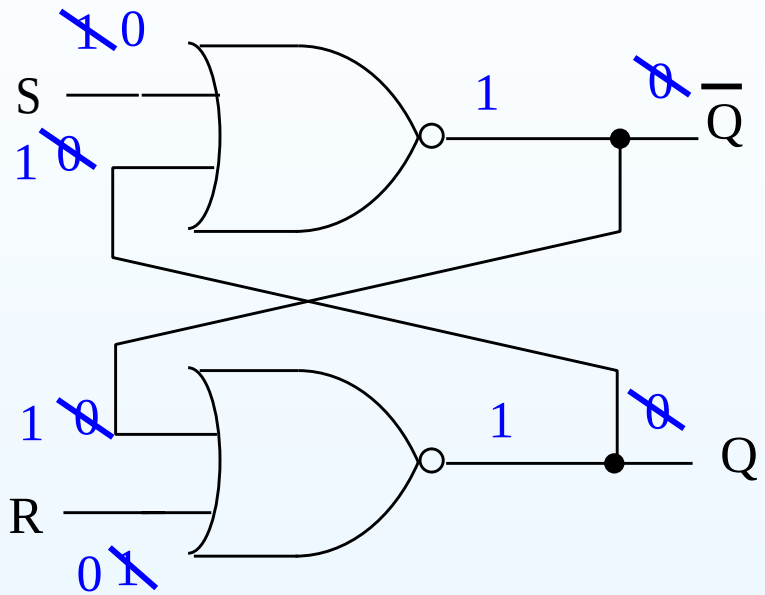
Computer Science Foundations

What Happens with 1 1 Input?



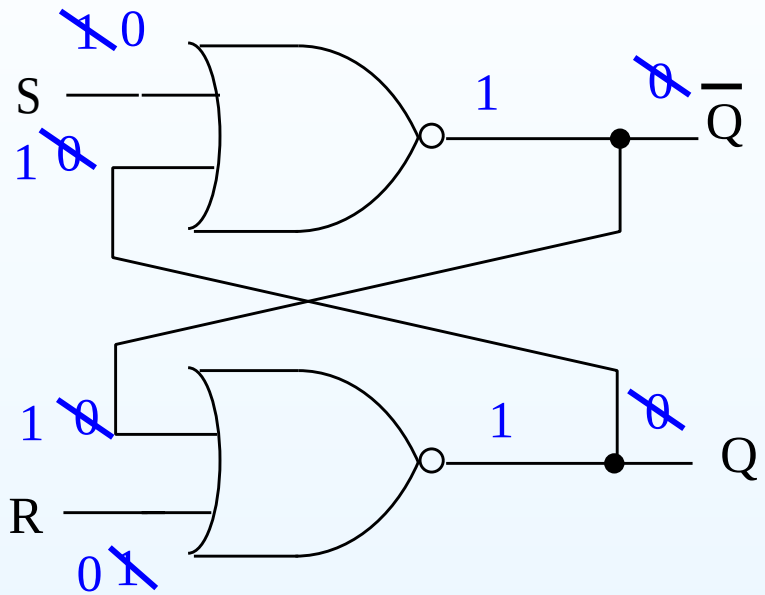
- State not reasonable: $Q = \overline{Q}$.
- What happens when $S = R = 0$?

What Happens with 1 1 Input?



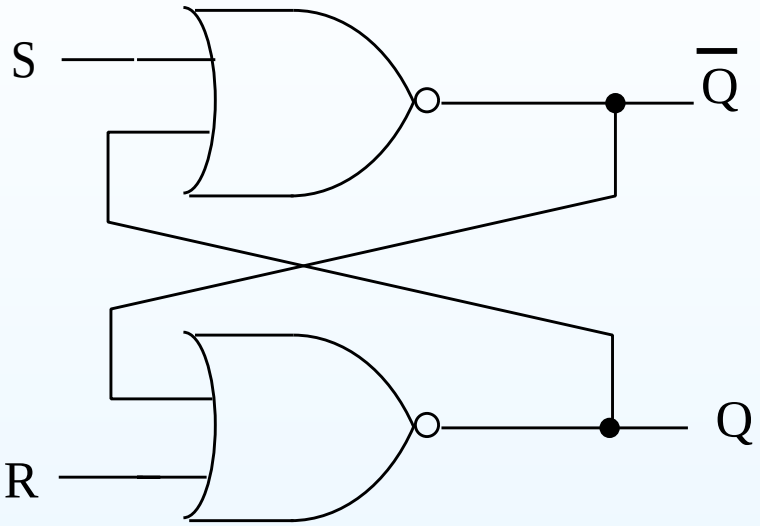
- State not reasonable: $Q = \overline{Q}$.
- What happens when $S = R = 0$?
- If absolutely no difference in timing, gate delays: oscillation (very unlikely)

What Happens with 1 1 Input?

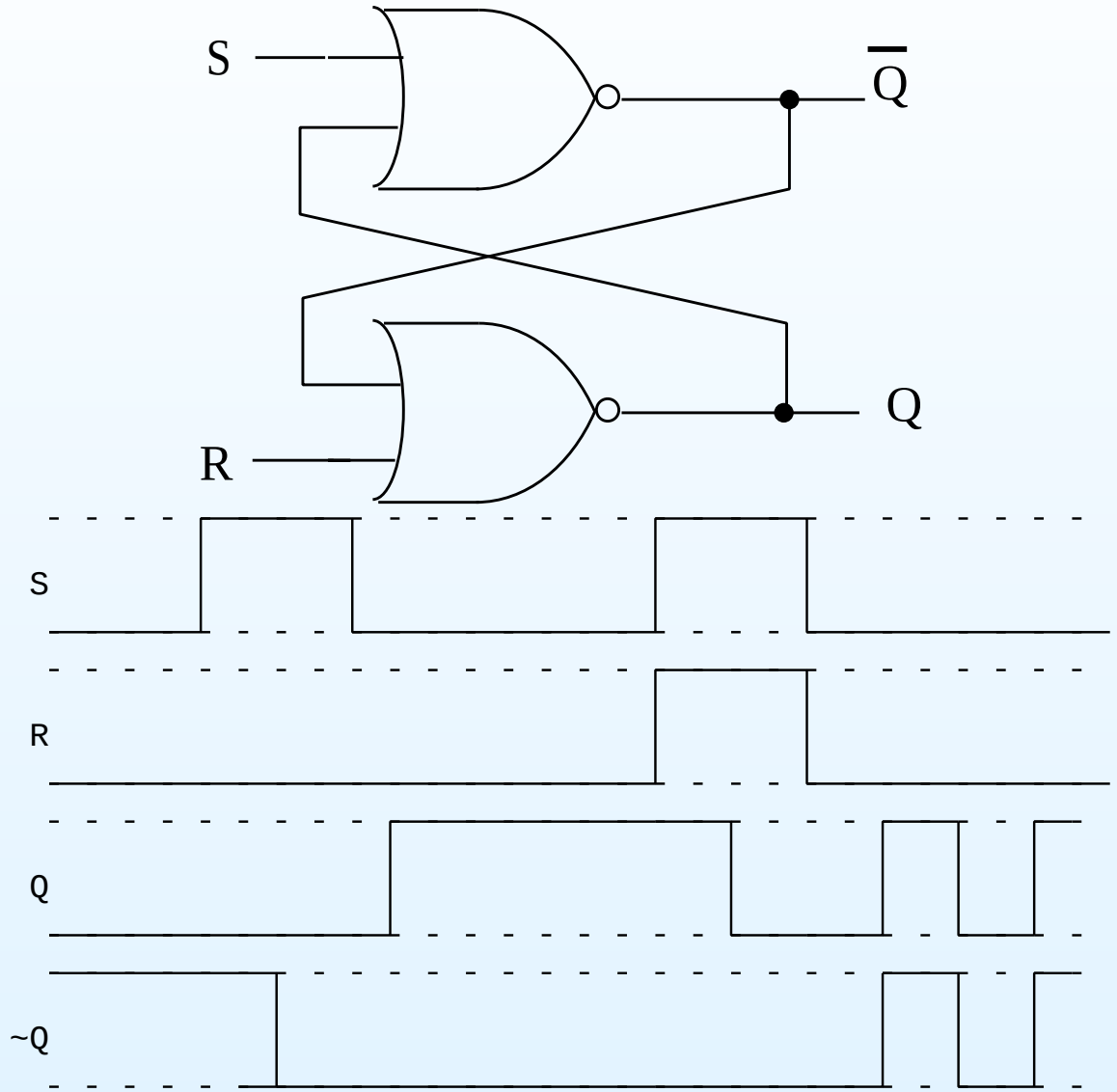


- State not reasonable: $Q = \overline{Q}$.
- What happens when $S = R = 0$?
- If absolutely no difference in timing, gate delays: oscillation (very unlikely)
- With different gate delay or, timing of S & R $\Rightarrow$ random stable state

Timing Diagrams

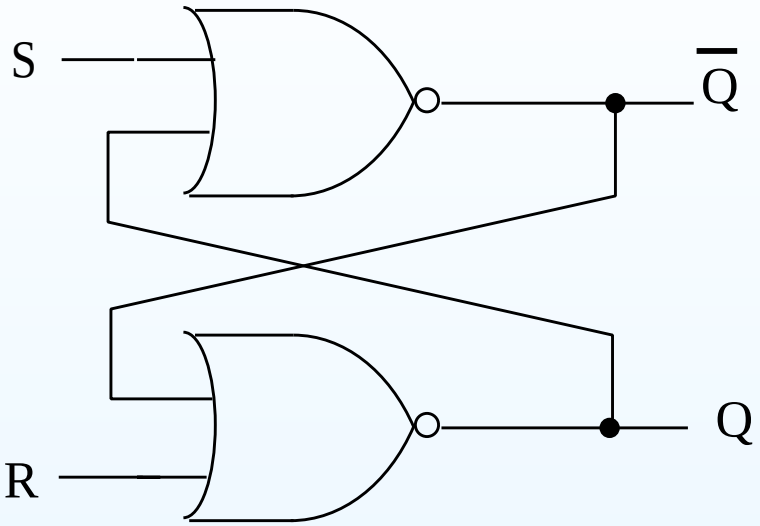


Timing Diagrams

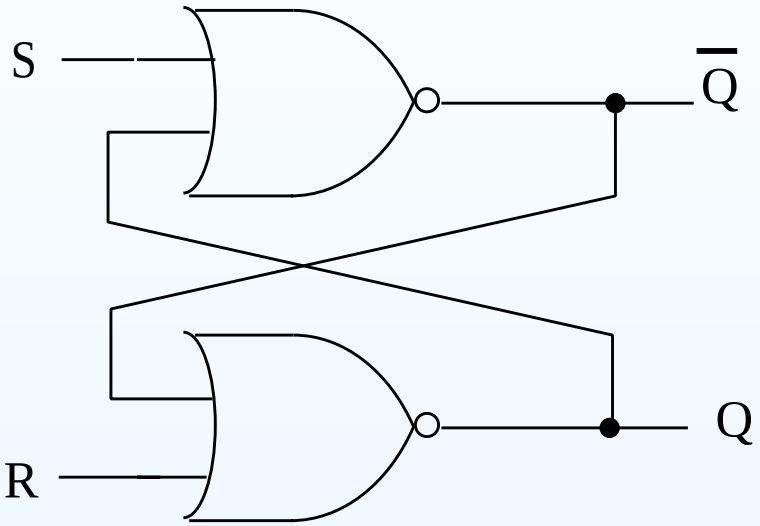


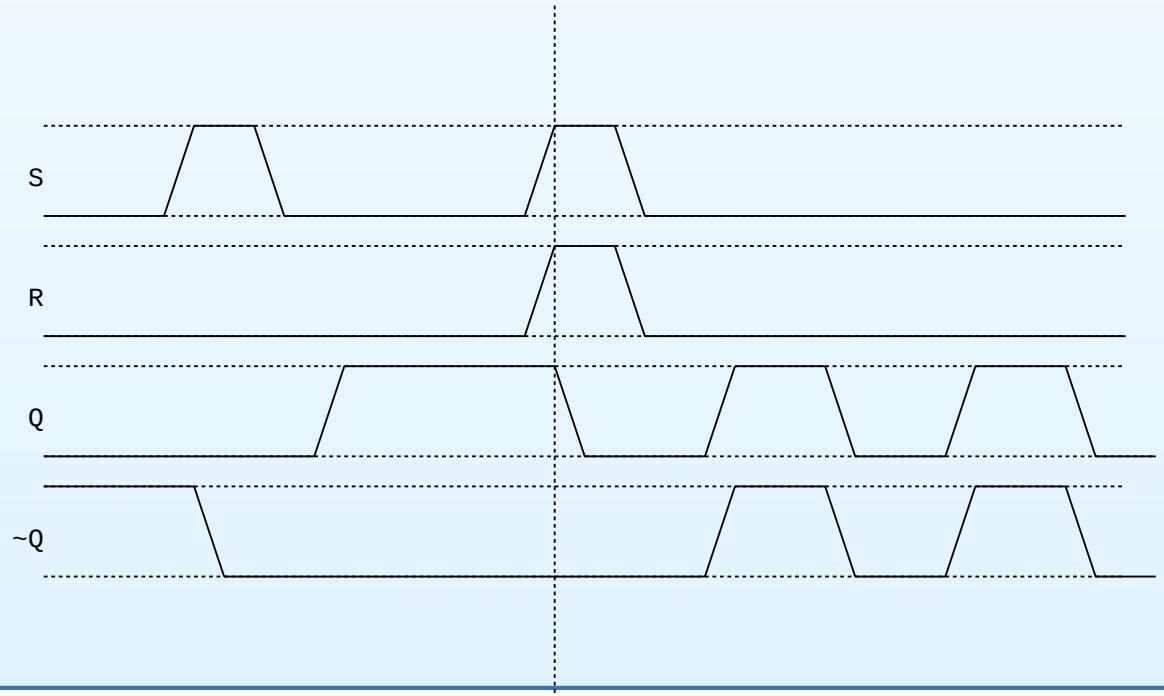
Computer Science Foundations

Timing Diagrams



Timing Diagrams





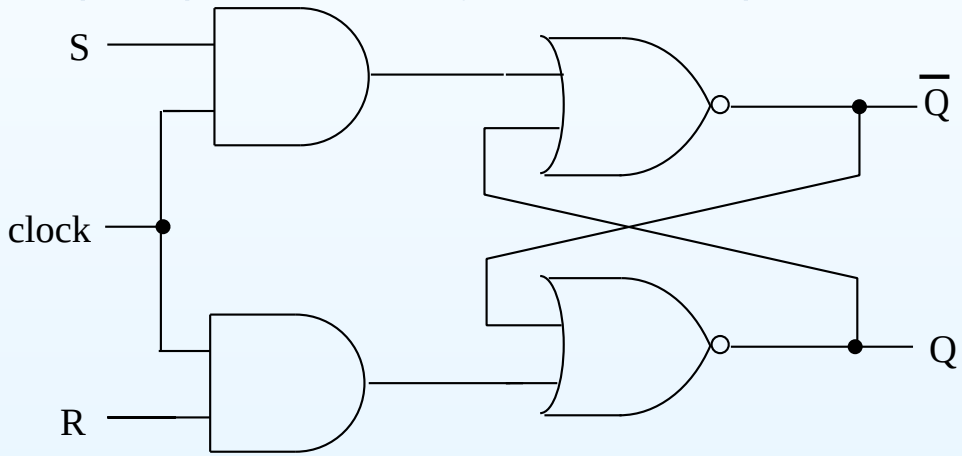
Computer Science Foundations

Clocked Latches

- Sometimes you want the state to change only at certain times.
- To do this, latches can be hooked to a clock.
- The clock is a signal that is 1 at certain intervals.
- Creating clocks in hardware and dealing with timing issues in circuits is a complex problem...beyond the scope of this course.

Clocked Latches

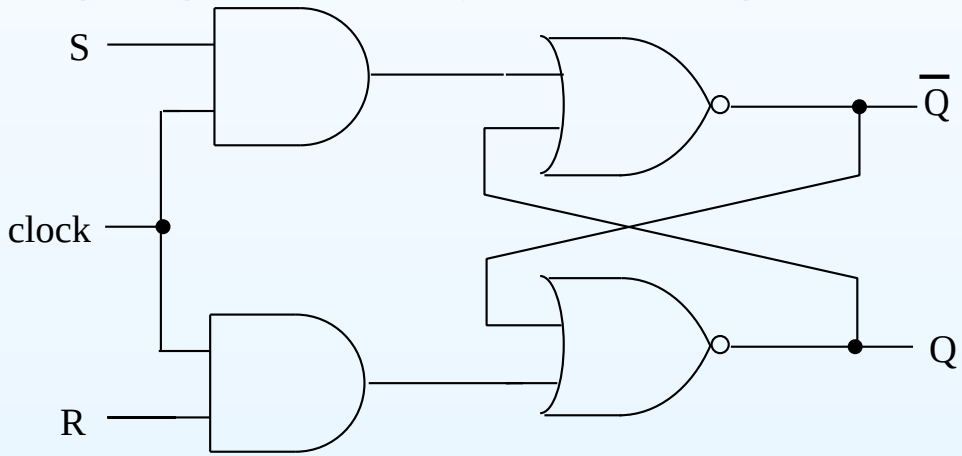
- Sometimes you want the state to change only at certain times.
- To do this, latches can be hooked to a clock.
- The clock is a signal that is 1 at certain intervals.
- Creating clocks in hardware and dealing with timing issues in circuits is a complex problem...beyond the scope of this course.



Computer Science Foundations

Clocked Latches

- Sometimes you want the state to change only at certain times.
- To do this, latches can be hooked to a clock.
- The clock is a signal that is 1 at certain intervals.
- Creating clocks in hardware and dealing with timing issues in circuits is a complex problem...beyond the scope of this course.



- If clock is not 1, equivalent to S and R of unclocked latch being 0, and maintain state. If clock is 1, value of S and R get through AND gate.

Computer Science Foundations

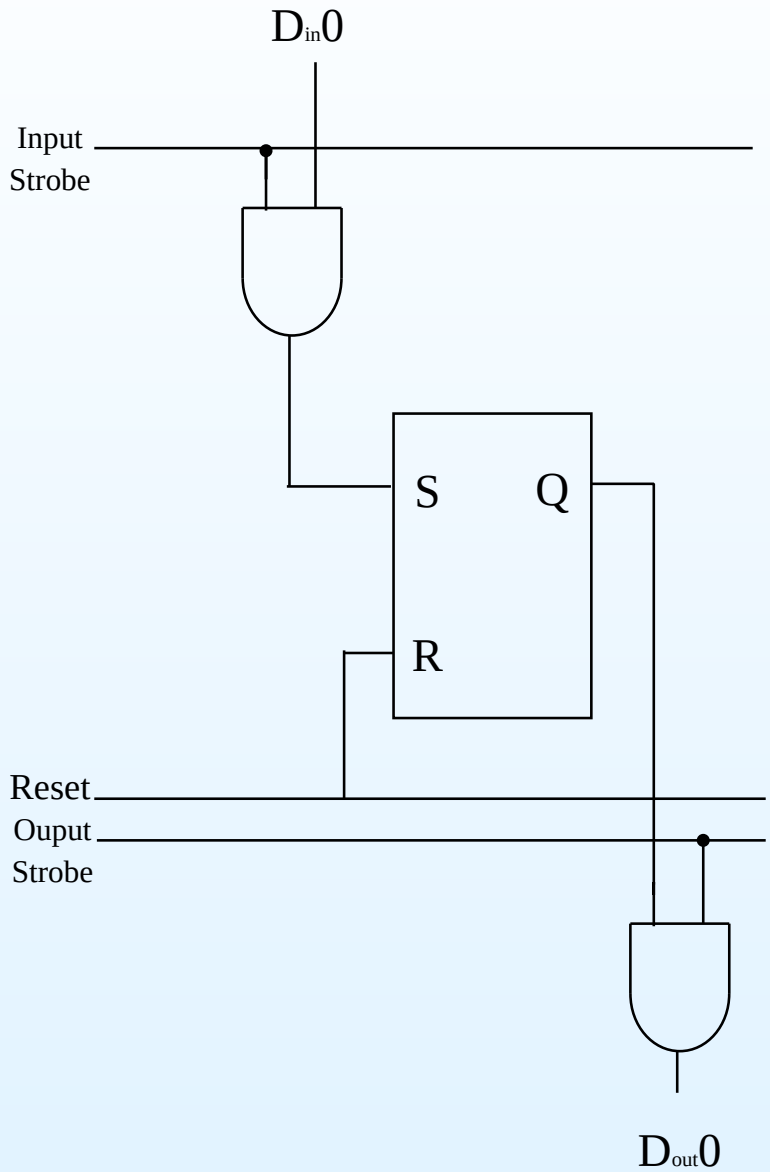
What Do We Need in a Register?

- Each bit can be stored in a latch (usually 8 or more)
- Nice to be able to reset register to all 0's
- Output available from each bit
- Input can be directed to each bit

How Can We Construct a Register?

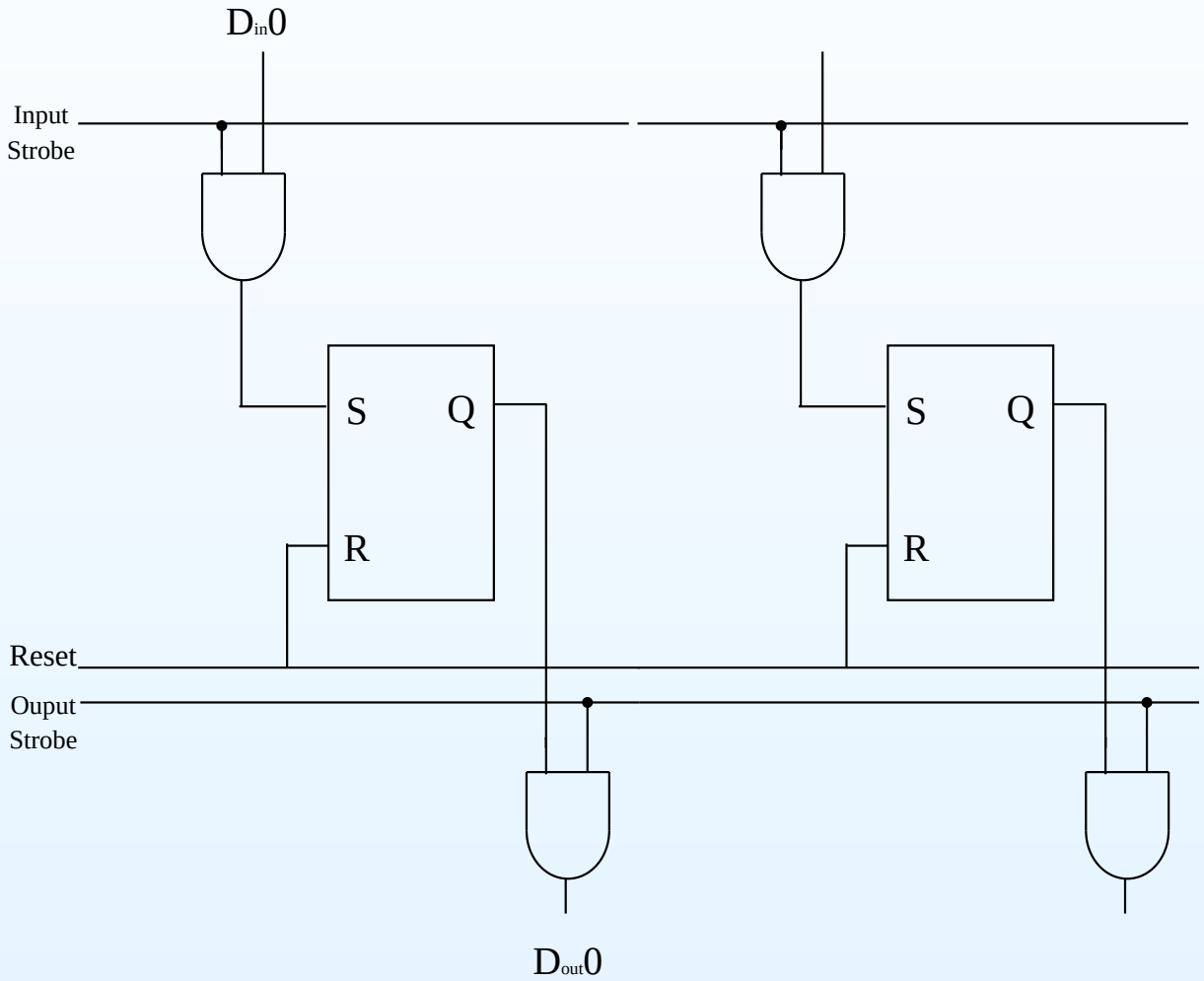
- We can use S-R Latches and simply make sure S and R are not both 1 at the same time.
- Have a reset line that goes to R in each latch.
- “Strobe” input and output so that write or read data only at specific times. (Works similarly to a clock.)

One Bit of a Register



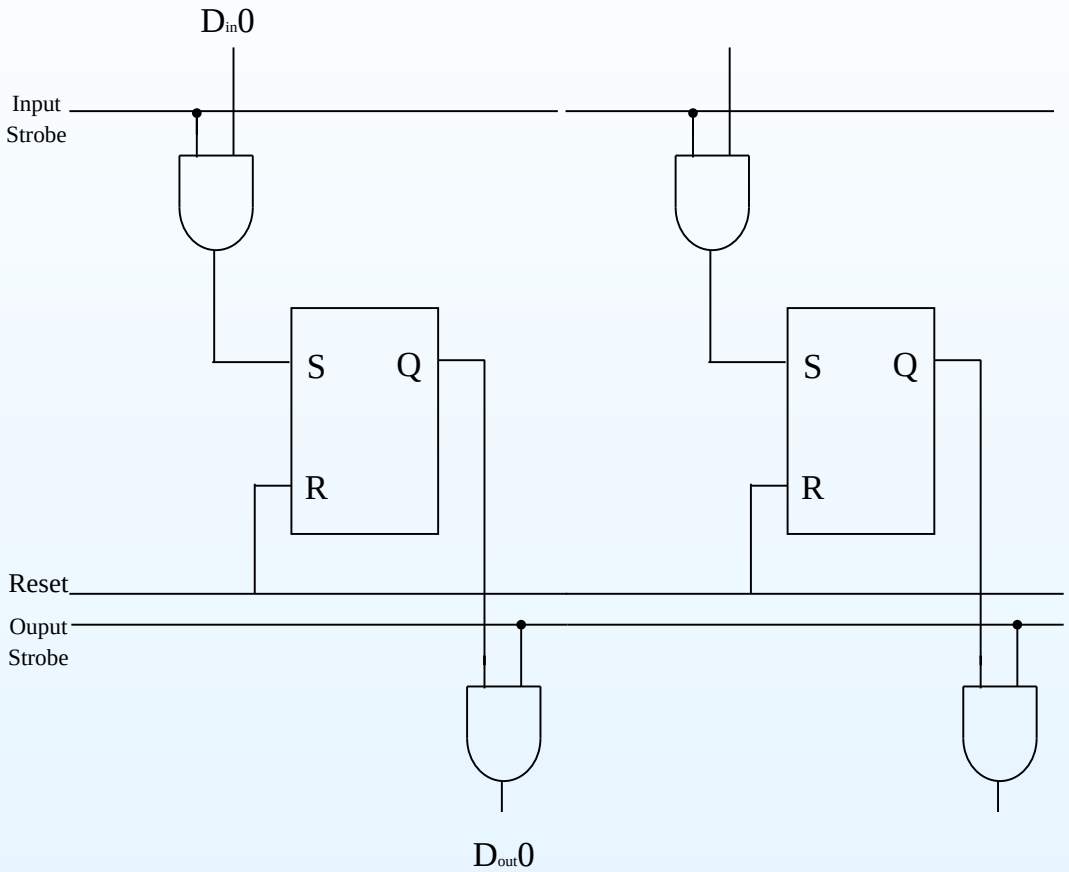
Computer Science Foundations

A Two-Bit Piece of a Register



Computer Science Foundations

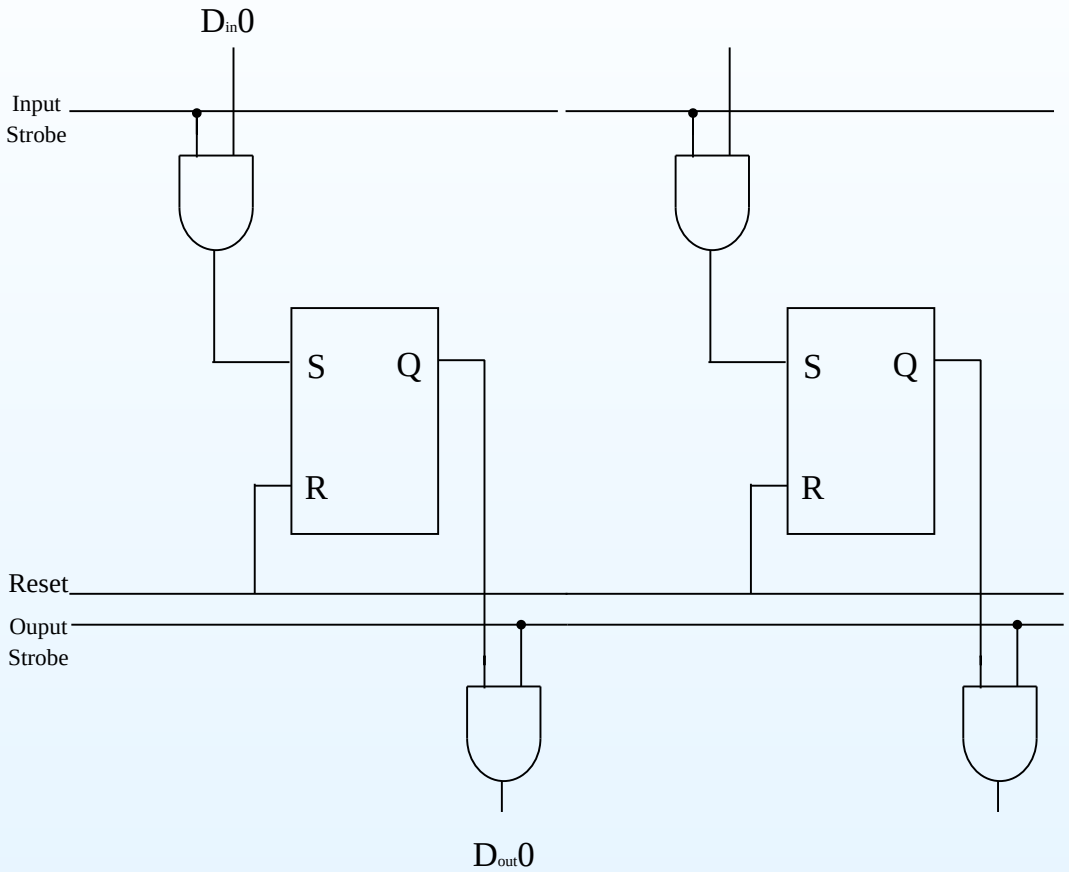
Using the Register



Can we just put in the data? What happens if the new data is 0, but the previous data was 1?

Computer Science Foundations

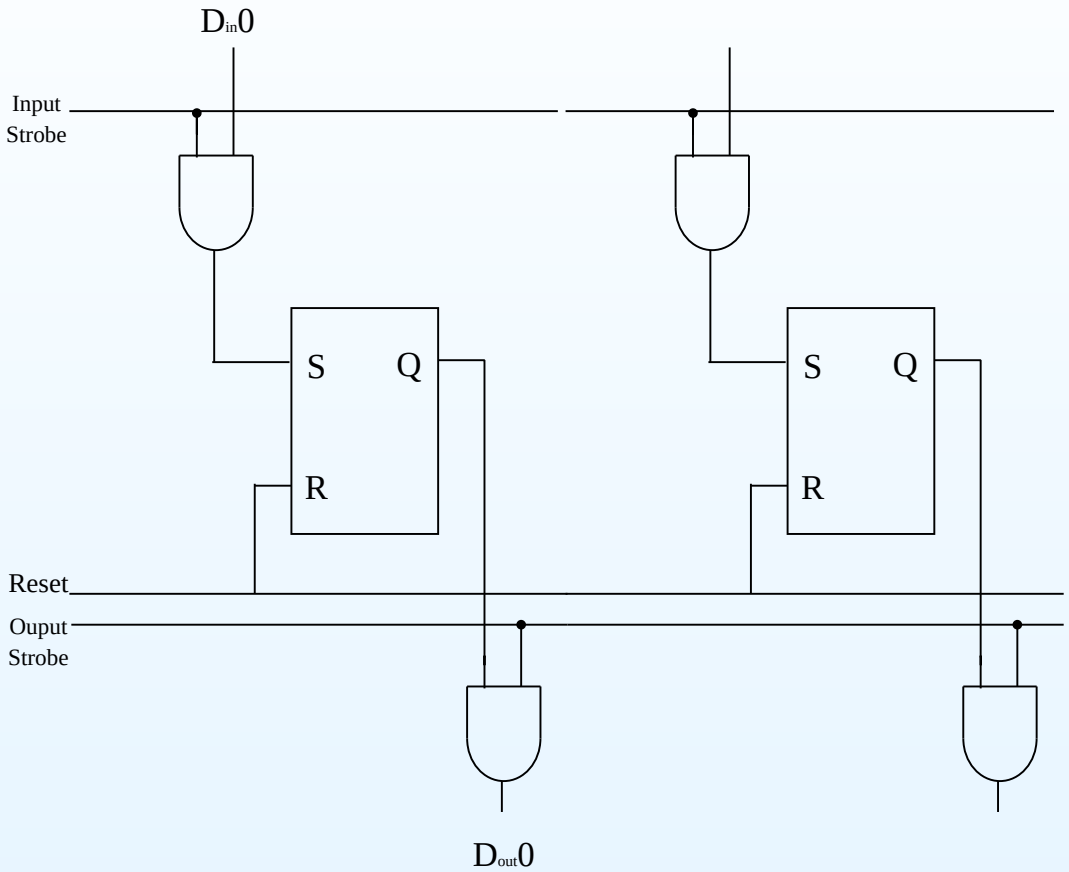
Using the Register



Can we just put in the data? What happens if the new data is 0, but the previous data was 1? What can we do to fix this?

Computer Science Foundations

Using the Register



Can we just put in the data? What happens if the new data is 0, but the previous data was 1? What can we do to fix this?

Need to reset each time before loading data.

Computer Science Foundations

More About Latches and Registers

- There are other kinds of latches which:
  - Do not require resetting register before loading data – so can toggle individual bits
  - Handle the problem of 1–1 inputs
- Registers can be constructed out of these latches, as well.

Computer Science Foundations

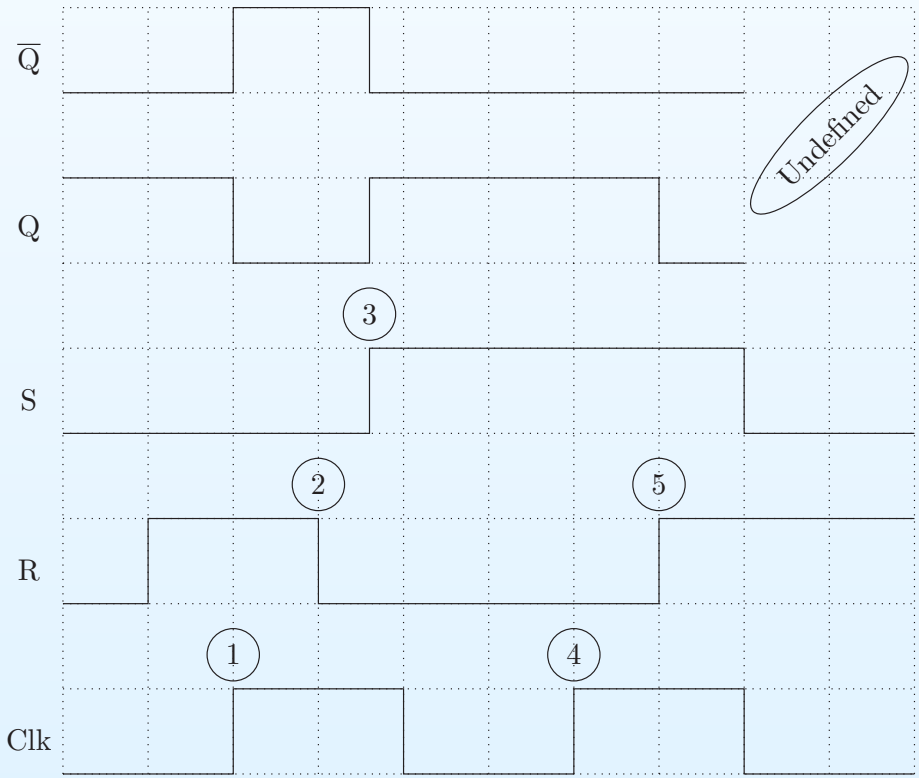
Flip-flops

- So far: *latches* – *level-triggered*
- Problem:
  - Only want changes once per clock pulse
  - But: clock pulse can be long $\Rightarrow$ time for inputs to change
  - Could lead to unwanted states: e.g., 1–1 to SR latch

Computer Science Foundations

Flip-flops

- So far: *latches* – *level-triggered*
- Problem:
  - Only want changes once per clock pulse
  - But: clock pulse can be long $\Rightarrow$ time for inputs to change
  - Could lead to unwanted states: e.g., 1–1 to SR latch



Computer Science Foundations

Flip-flops

- So far: *latches* – *level-triggered*
- Problem:
  - Only want changes once per clock pulse
  - But: clock pulse can be long $\Rightarrow$ time for inputs to change
  - Could lead to unwanted states: e.g., 1–1 to SR latch
- Solution: *edge-triggering*

Flip-flops

- So far: *latches* – *level-triggered*
- Problem:
  - Only want changes once per clock pulse
  - But: clock pulse can be long $\Rightarrow$ time for inputs to change
  - Could lead to unwanted states: e.g., 1–1 to SR latch
- Solution: *edge-triggering*
- Edge-triggered latch $\equiv$ *flip-flop*

Computer Science Foundations

Edge-triggering

- Problem: How to trigger only on *edge* of clock pulse?

Edge-triggering

- Problem: How to trigger only on *edge* of clock pulse?
- One way: use *pulse detector* circuit

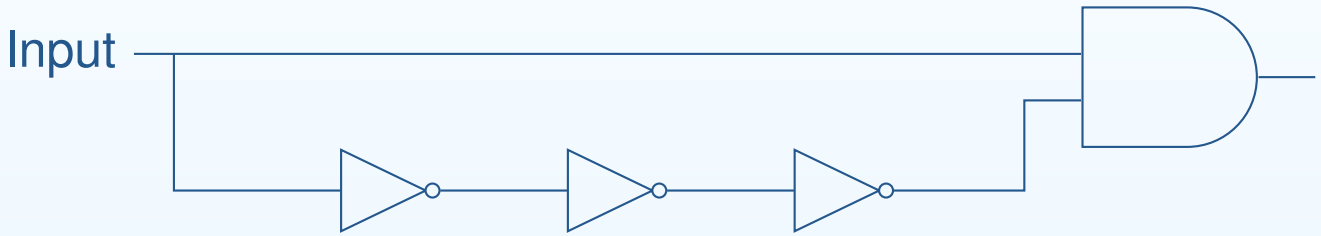
Edge-triggering

- Problem: How to trigger only on *edge* of clock pulse?
- One way: use *pulse detector* circuit
- Can use gate delays to our advantage:

Computer Science Foundations

Edge-triggering

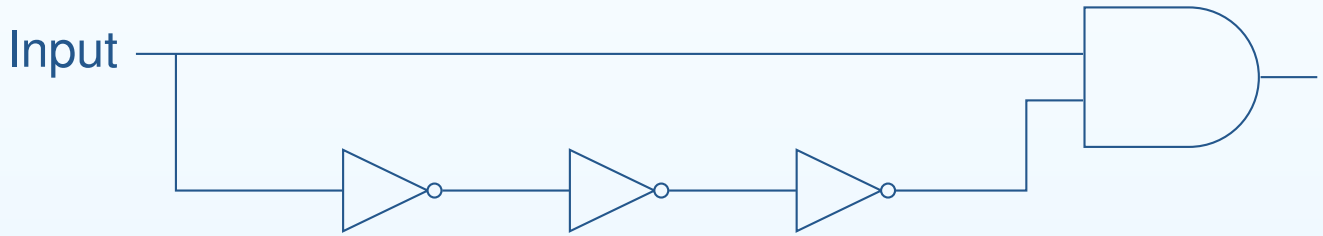
- Problem: How to trigger only on *edge* of clock pulse?
- One way: use *pulse detector* circuit
- Can use gate delays to our advantage:



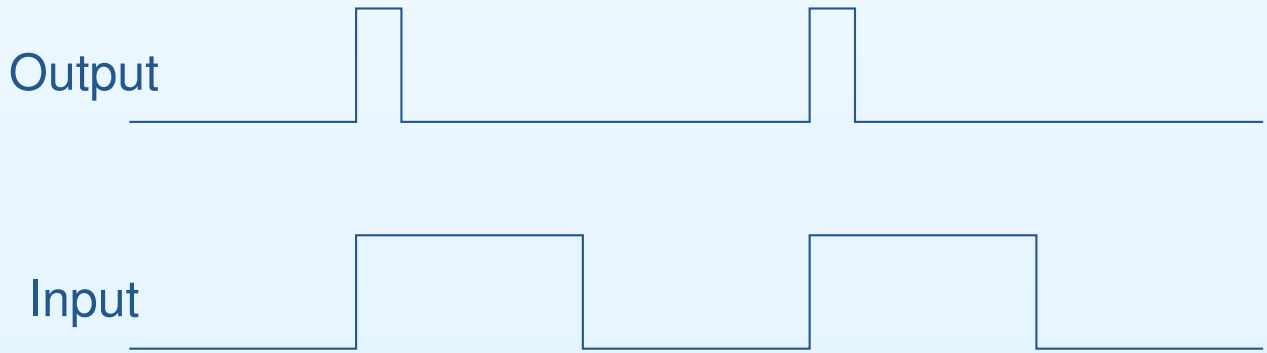
Computer Science Foundations

Edge-triggering

- Problem: How to trigger only on *edge* of clock pulse?
- One way: use *pulse detector* circuit
- Can use gate delays to our advantage:



- Gives this behavior:



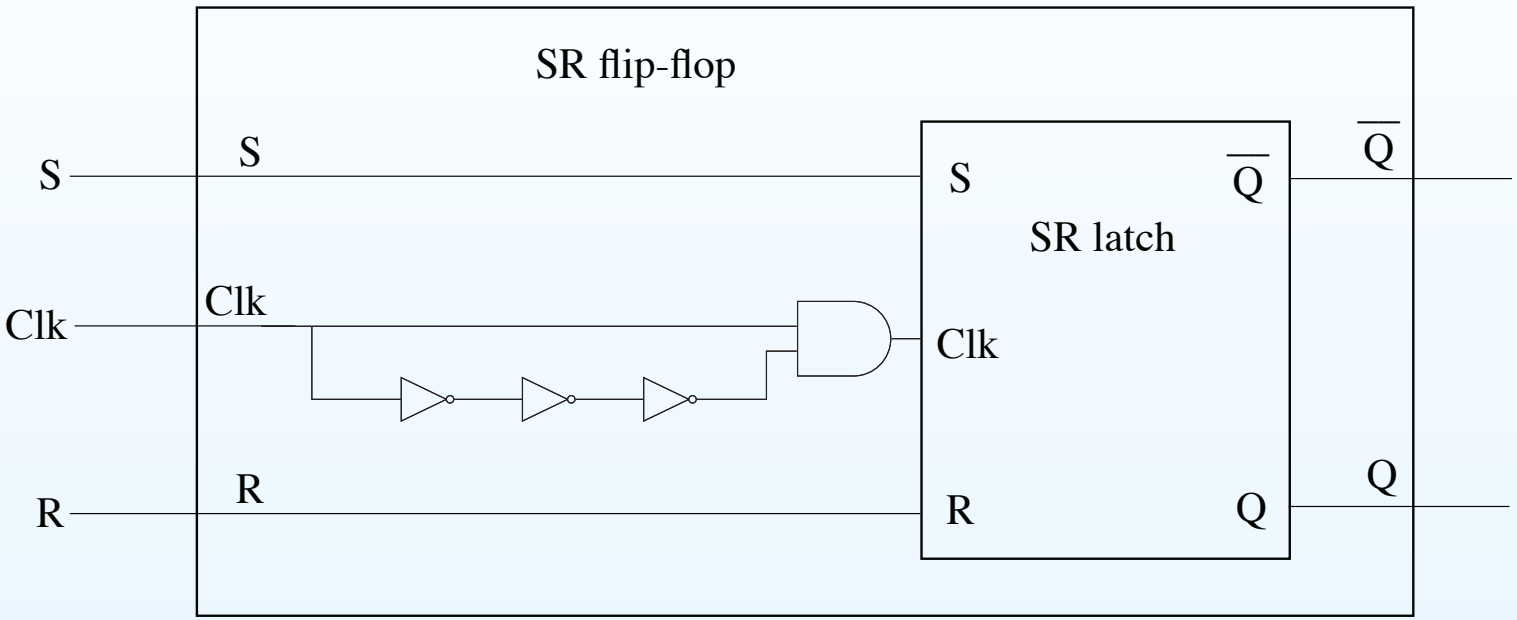
Computer Science Foundations

SR flip-flop

- Put pulse detector on clock of clocked latch $\Rightarrow$ flip-flop

SR flip-flop

- Put pulse detector on clock of clocked latch $\Rightarrow$ flip-flop



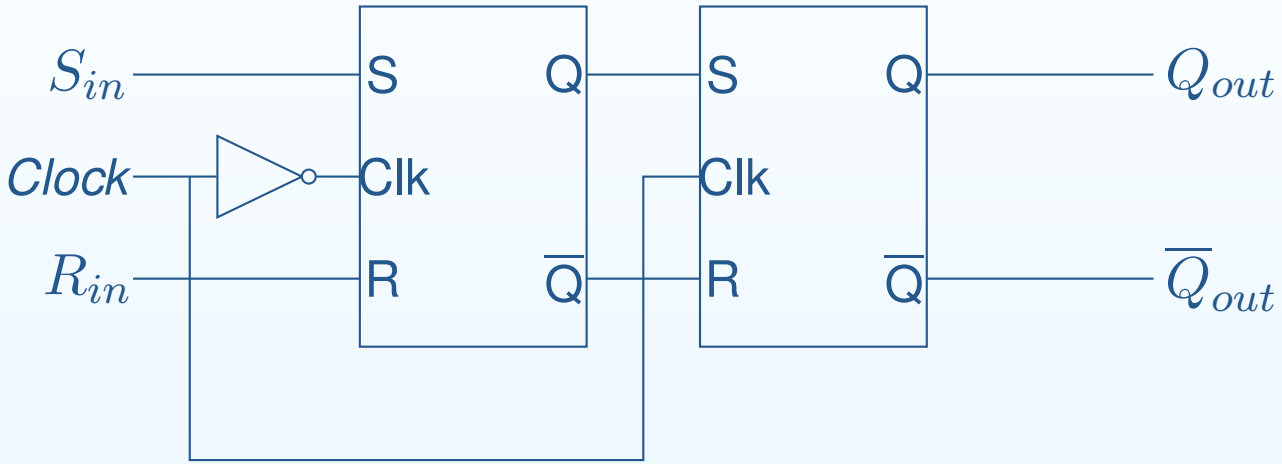
Computer Science Foundations

Edge triggering: master–slave configuration

- Can also get edge triggering by using two latches:

Edge triggering: master–slave configuration

- Can also get edge triggering by using two latches:



Computer Science Foundations