Outline

1. Digital Systems
   - flip-flops
   - registers
2. Intrinsic Operations
   - values of numbers given in words
   - queues and stacks

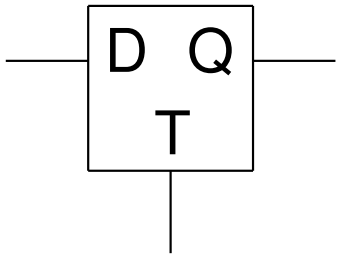
MCS 260 Lecture 10
Introduction to Computer Science
Jan Verschelde, 23 June 2023

flip-flops and registers
queues and stacks

Flip-Flops

one bit memory

Flip-Flops (and latches) are the simplest circuits to store one bit.



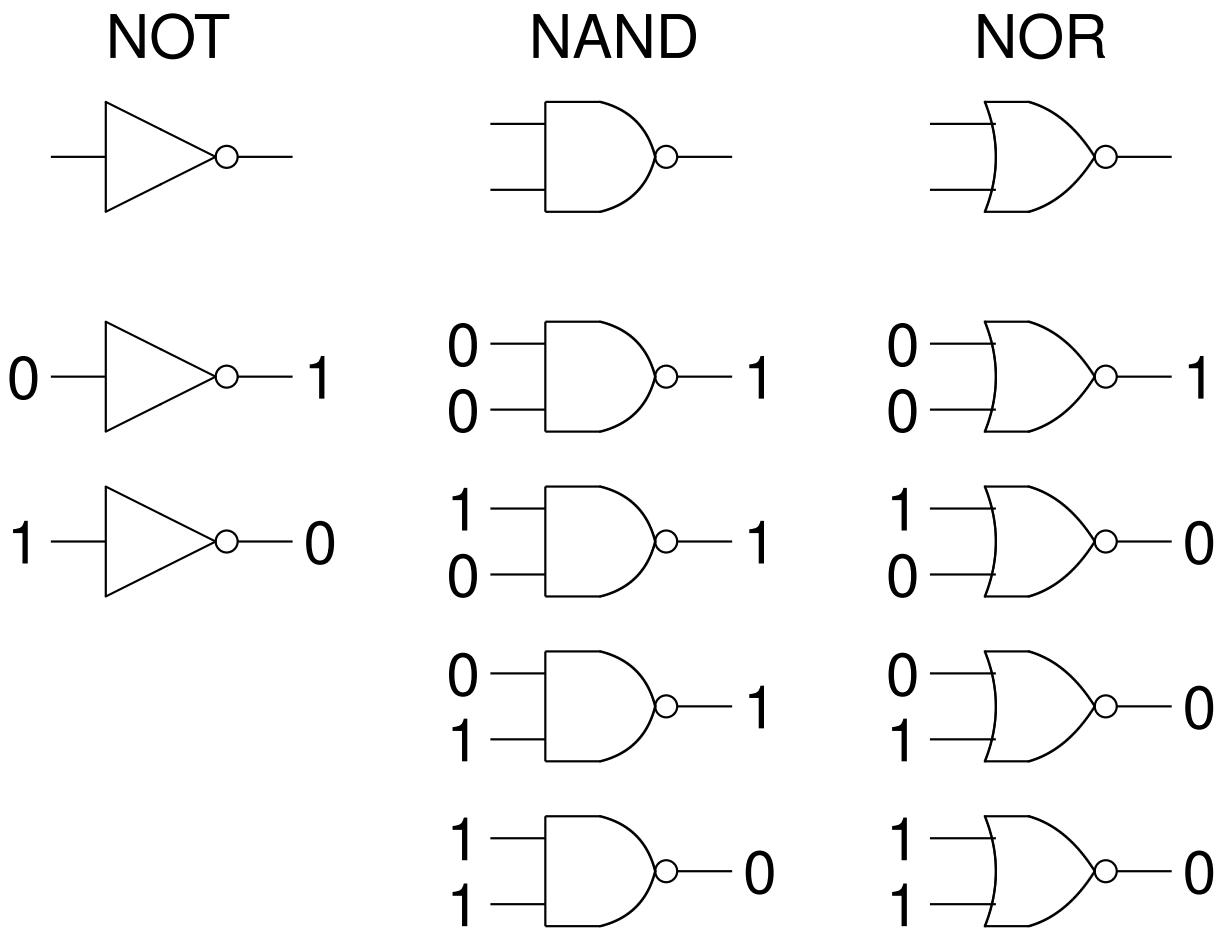
D: input line
T: clock line
Q: output line

Its behavior is as follows:

1. When one arrives on the clock line, the output line is set to the value present on the input line.
2. The value at the output line is stored at the flip-flop, until a new one arrives on the clock line.

Logic Gates

A flip-flop is realized with NOT, NAND, and NOR gates:

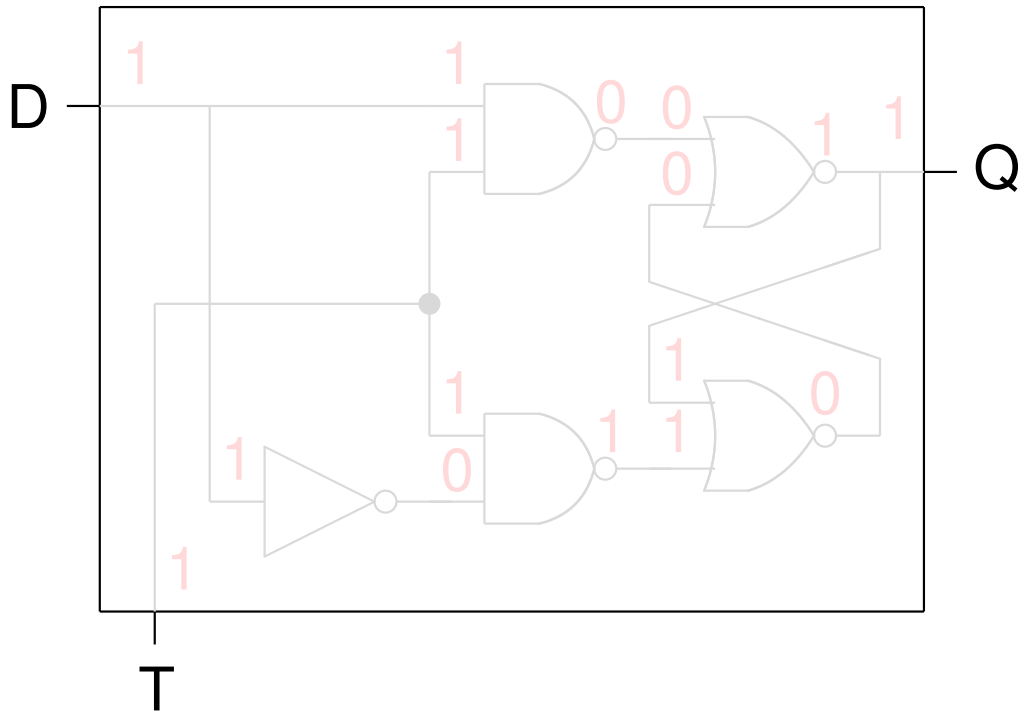


Exercise: represent `NOT ( x NOR ( NOT y ) ).`

Realization of a Flip-Flop

one NOT, two NANDs, and two NORs

We simulate the *latching* of a 1 at D to Q, with 1 at Q.

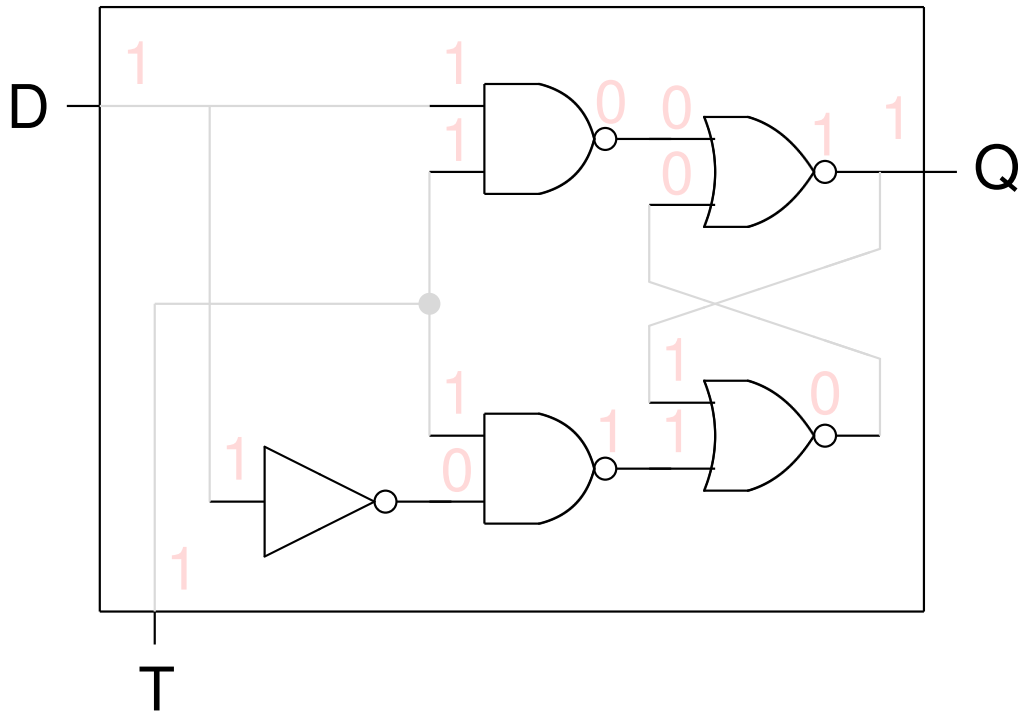


Exercise: verify the effect is the same for 0 at Q.

Realization of a Flip-Flop

one NOT, two NANDs, and two NORs

We simulate the *latching* of a 1 at D to Q, with 1 at Q.

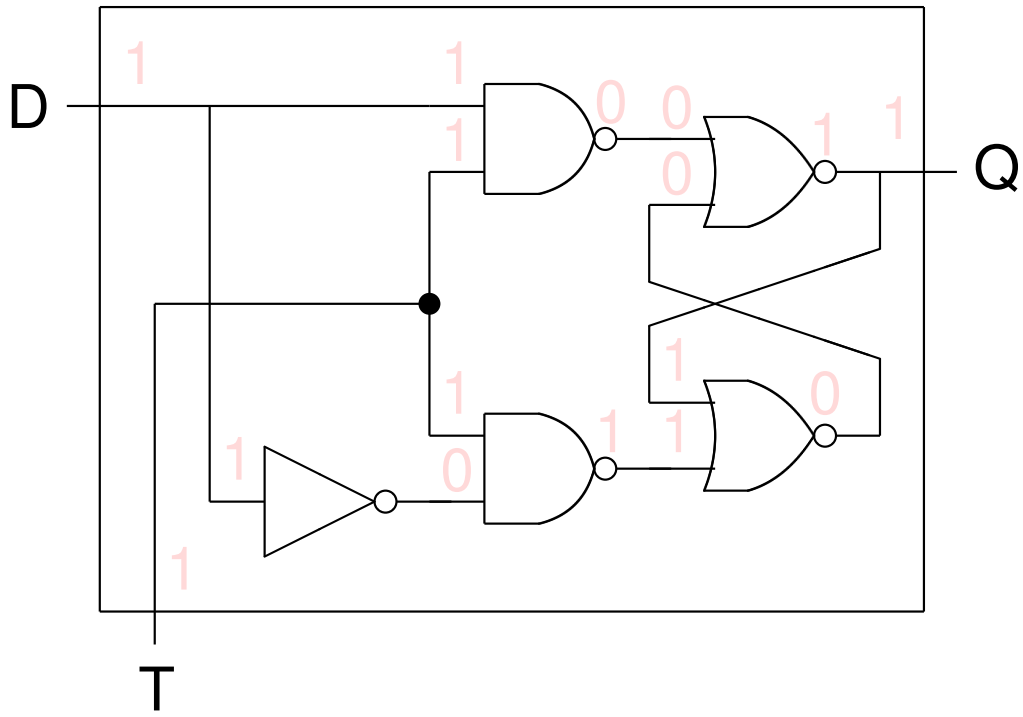


Exercise: verify the effect is the same for 0 at Q.

Realization of a Flip-Flop

one NOT, two NANDs, and two NORs

We simulate the *latching* of a 1 at D to Q, with 1 at Q.

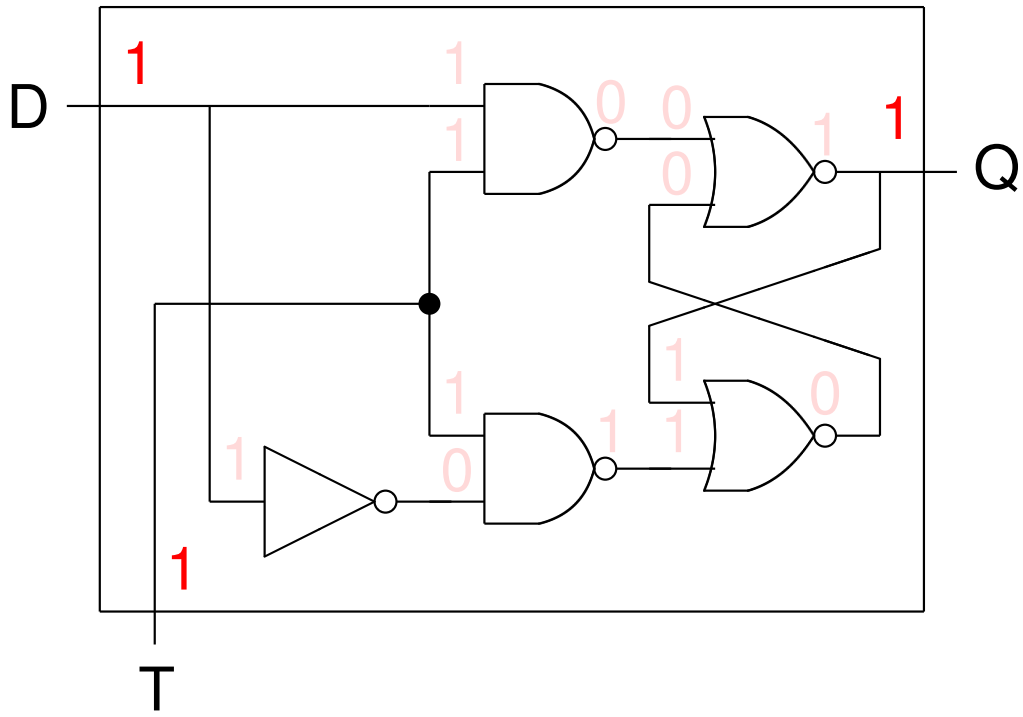


Exercise: verify the effect is the same for 0 at Q.

Realization of a Flip-Flop

one NOT, two NANDs, and two NORs

We simulate the *latching* of a 1 at D to Q, with 1 at Q.

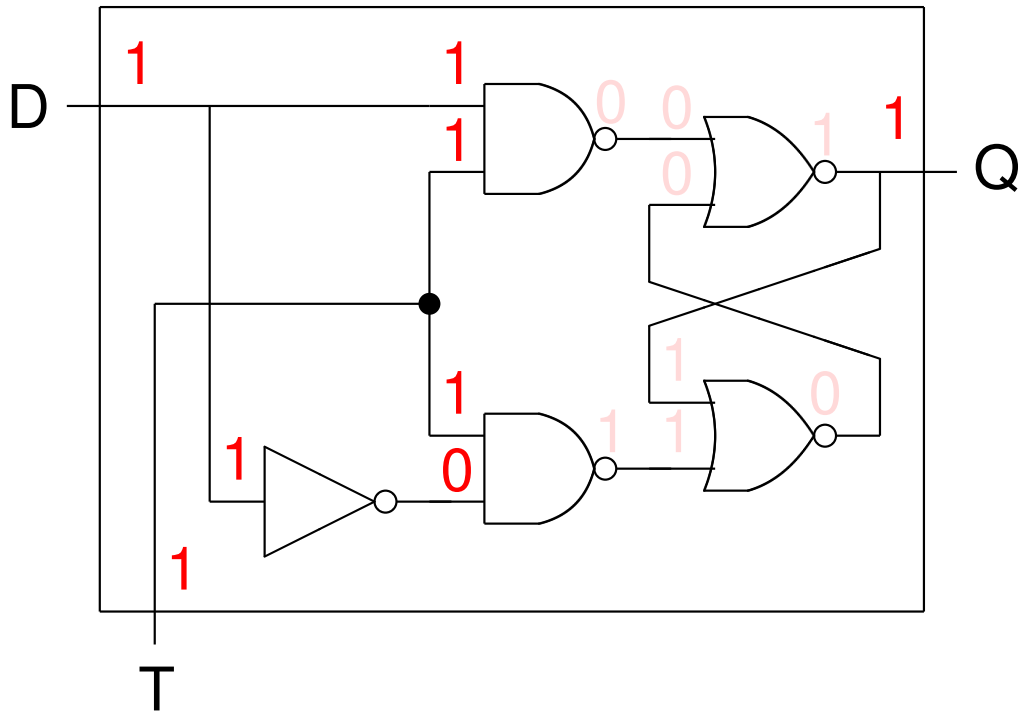


Exercise: verify the effect is the same for 0 at Q.

Realization of a Flip-Flop

one NOT, two NANDs, and two NORs

We simulate the *latching* of a 1 at D to Q, with 1 at Q.

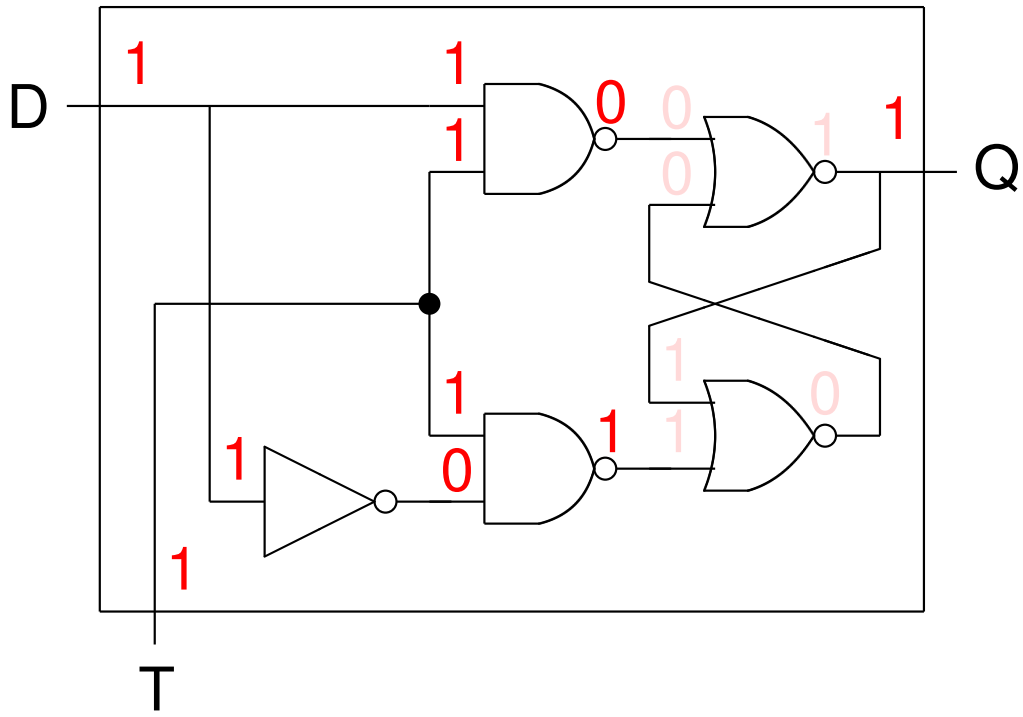


Exercise: verify the effect is the same for 0 at Q.

Realization of a Flip-Flop

one NOT, two NANDs, and two NORs

We simulate the *latching* of a 1 at D to Q, with 1 at Q.

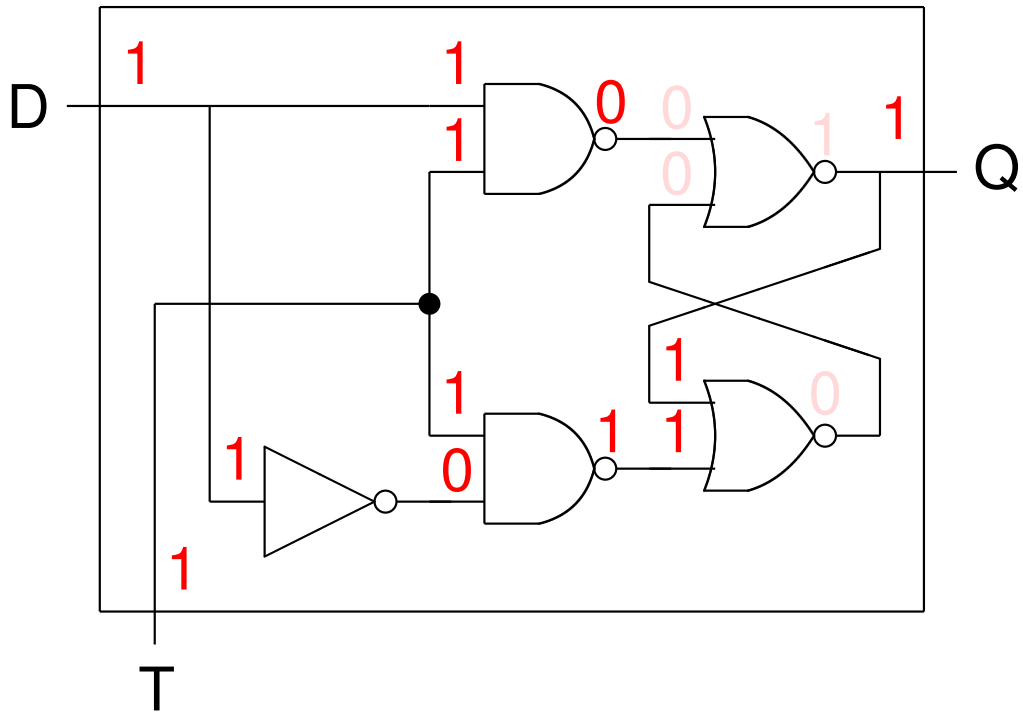


Exercise: verify the effect is the same for 0 at Q.

Realization of a Flip-Flop

one NOT, two NANDs, and two NORs

We simulate the *latching* of a 1 at D to Q, with 1 at Q.

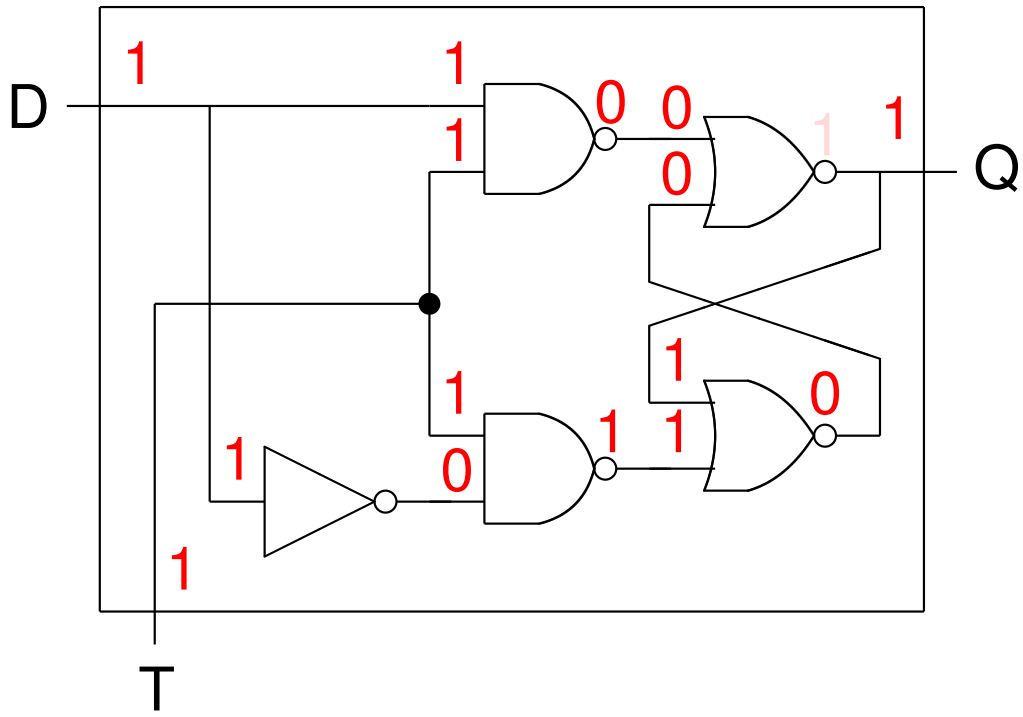


Exercise: verify the effect is the same for 0 at Q.

Realization of a Flip-Flop

one NOT, two NANDs, and two NORs

We simulate the *latching* of a 1 at D to Q, with 1 at Q.

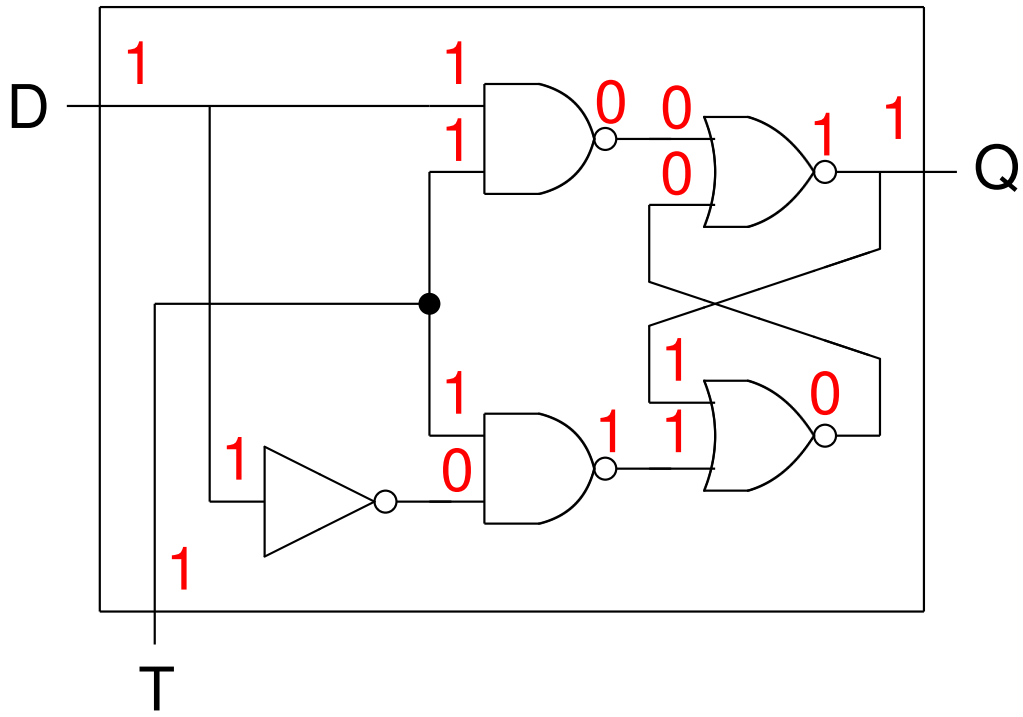


Exercise: verify the effect is the same for 0 at Q.

Realization of a Flip-Flop

one NOT, two NANDs, and two NORs

We simulate the *latching* of a 1 at D to Q, with 1 at Q.

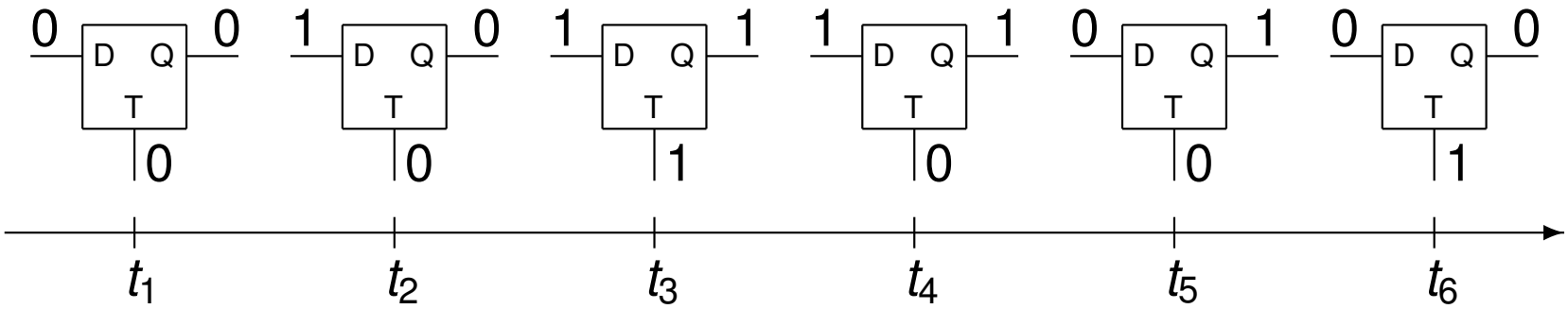


Exercise: verify the effect is the same for 0 at Q.

Behavior in Time

how latching of bits works

The evolution in time is



At t_1: 0 at D, 0 at T, and 0 at Q

At t_2: 1 at D, but 0 at T and nothing happens

At t_3: 1 at T $\Rightarrow$ 1 at D copied to 1 at Q

At t_4: 0 at T and nothing happens

At t_5: 0 at D, but 0 at T and nothing happens

At t_6: 1 at T $\Rightarrow$ 0 at D copied to 0 at Q

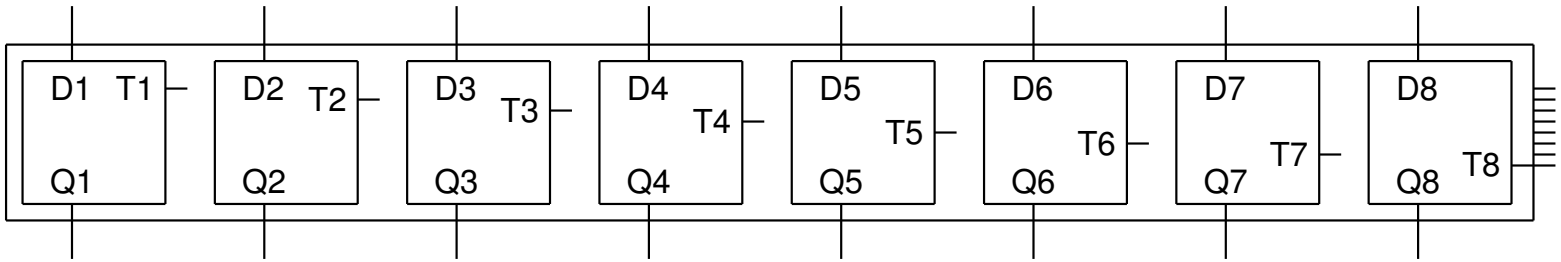
flip-flops and registers
queues and stacks

1. Digital Systems
   - flip-flops
   - registers
2. Intrinsic Operations
   - values of numbers given in words
   - queues and stacks

Registers

An 8-bit register is realized with 8 flip-flops.

eight input lines and eight clock lines

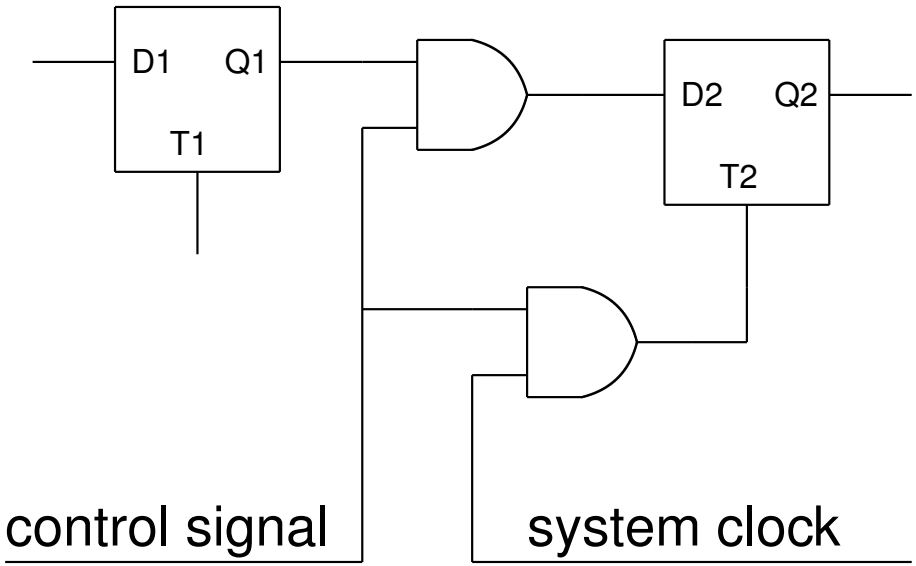


eight output lines

Copy Bits

using 2 AND gates

We want to copy a bit from one latch to the other.
The bit is at the output line of the first latch, at Q1 and has to get to the output line of the second latch, at Q2.



The copy is activated by the control signal.
For synchronization, another signal from the system clock copies the bit from the input line at D2 to Q2.

flip-flops and registers
queues and stacks

1. Digital Systems
   - flip-flops
   - registers

2. Intrinsic Operations
   - values of numbers given in words
   - queues and stacks

value of a number given in words

Problem Statement:

Input: string with *at most* two words,
separated by *exactly one* space.

Output: value of the number represented by the string.

Running the Python code `write_values.py`:

```
$ python write_values.py
give a number in words : forty seven
the value of forty seven is 47
```

Note: reverse of `write_numbers.py` of the previous lecture.

flip-flops and registers
queues and stacks

1. Digital Systems
   - flip-flops
   - registers
2. Intrinsic Operations
   - values of numbers given in words
   - queues and stacks

Queues and Stacks — defined by lists

Two protocols to retrieve elements sequentially:

FIFO: First In First Out, a queue
think of a normal waiting list

FILO: First in Last Out, a stack
think of a pile of papers on a desk

Intrinsic operations on a list `L`:

`L.append(<item>)`	appends `<item>` to `L`
`<item> = L.pop()`	removes last item added to `L`
`<item> = L.pop(0)`	removes first item added to `L`
`L.insert(0,<item>)`	inserts `<item>` to front of `L`

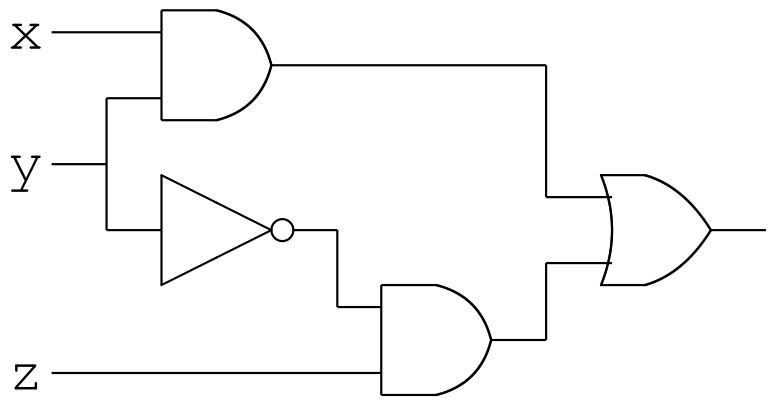
All these operations modify `L`!

How to select from `L`, *without modifications*?

```
>>> L[0]
>>> L[-1]
```

Exercises

1. Consider the circuit:



   Write the logical expression represented by the circuit.

2. Translate the realization diagram for a flip-flop into a logical expression involving the variables D, T, and Q, using NOT, NAND, and NOR.

3. Extend `write_values.py` so it works for all strings which spell out numbers less than one thousand.

4. To print the bits in the correct order in the conversion of a number from decimal to binary notation, do we use a queue or a stack? Illustrate with an example.