



A8.8 D-type flip-flops and clocks

Boolean algebra and logic circuits · A level · OCR H446 1.4.3, AQA 7517

4.6.4.1 · about 20 min

Name _____

Class _____

Date _____

What this lesson is about

Clock signals, edge triggering, the D-type flip-flop as one bit of memory, registers and a divide-by-two counter.

- Memory needs feedback
- The clock
- The D-type flip-flop
- Reading a timing diagram
- What flip-flops build

Questions

5 marks in all

1. When does a positive edge-triggered D-type flip-flop change its output Q? [1 mark]

- ☐ A At the rising edge of the clock, when Q takes the value of D
- ☐ B Whenever D changes
- ☐ C Whenever the clock is 1
- ☐ D At the falling edge of the clock, when Q becomes NOT D

2. What is a D-type flip-flop used for? [1 mark]

- ☐ A Storing one bit
- ☐ B Adding two bits
- ☐ C Generating the clock signal
- ☐ D Inverting a signal

3. What does this program print? [1 mark]

```
clock = [0, 1, 0, 1, 1, 0, 1]
d = [1, 1, 0, 0, 1, 0, 0]
q, last = 0, 0
for i in range(len(clock)):
    if last == 0 and clock[i] == 1:
        q = d[i]
        last = clock[i]
print(q)
```

4. A flip-flop has $\bar{Q}$ wired back to D. The clock runs at 1000 Hz. What is the frequency of Q, in Hz? [1 mark]

5. What is the difference between combinational and sequential logic?

[1 mark]

- ☐ **A** Sequential logic's output also depends on what it has stored; combinational logic's depends only on its current inputs
- ☐ **B** Combinational logic uses a clock and sequential logic does not
- ☐ **C** Sequential logic uses only NAND gates
- ☐ **D** There is no difference

The task: a D-type flip-flop

Write `rising_edge(previous, now)`, which takes two clock values (each 0 or 1) and returns `True` if the clock has just gone from 0 to 1, otherwise `False`. **Part 1.** Simulate a D-type flip-flop over the 14 steps in the lists `CLOCK` and `D`. `Q` starts at 0 and the clock value before step 0 counts as 0. At each step, if `rising_edge` is true for the previous and current clock values, `Q` becomes `D` at that step; otherwise `Q` keeps its value. Record `Q` after every step, and print all 14 values on one line, separated by spaces, in the form `Q: 0 0 0 ...`. **Part 2.** Build a divide-by-two counter: a flip-flop whose `D` is always `NOT Q`. `Q` starts at 0. Simulate 8 rising edges. After each edge, show `Q` on the LED (`led(0, 255, 0)` when `Q` is 1, `led(0, 0, 0)` when it is 0) and `wait(0.25)`. Then print the 8 values of `Q` on one line, in the form `divider: 1 0 ...`.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

CLOCK = [0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1]
D = [0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1]

def rising_edge(previous, now):
    return False
```

Plan your program here, then type it in and press Run.



Do it on the robot

www.bugbotlab.com/learn/a8-8-d-type-flip-flops-and-clocks/

The simulator checks it and tells you when it passes. Nothing to install, no account.

Challenges

1. Change Part 1 so the flip-flop triggers on the falling edge instead. Which values of `Q` change?
2. Chain two dividers: the second flip-flop changes whenever the first one's `Q` falls from 1 to 0. Print both `Qs` for 8 edges and read them as a 2-bit binary number.
3. Build a 4-bit register from four `DFlipFlop` objects sharing one clock, and store 1011 on one rising edge.

