4CS015
Lecture 5: Sequential Logic and Memory

Prepared by: Uttam Acharya

1. The story so far

- We’ve looked at combinational logic circuits
- We’ve designed an ALU from logic gates

2. The next step

- An ALU without any way of running a programme or storing information is just a calculator
- What we are going to looked at now, is how logic circuits work with **time** and what the implications are.

3. Combinational vs. Sequential

- **Combinational Logic**
  - The output of the logic circuit is a function of the inputs
  - Think ‘OR’ gate
- **Sequential Logic**
  - The output of the circuit is a function of the current inputs <u>and</u> the previous output state
  - To accomplish this, the circuit needs to remember what the previous state was
  - i.e. It needs memory!

3.1 Sequential logic

- Creation of memory from Logic.
- By looking at the output of the last logic state.
- Modifying new logic state based on what went on before.

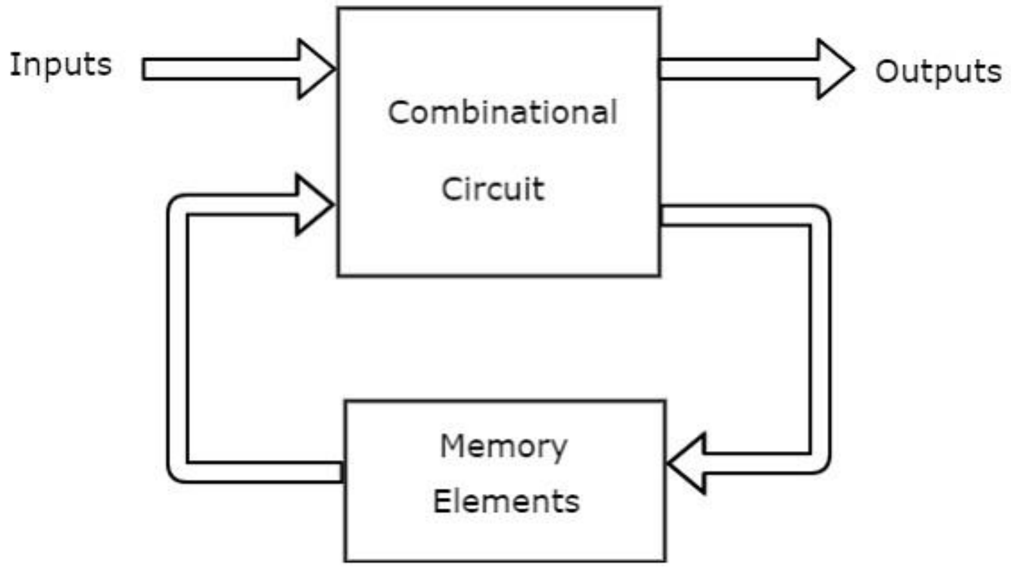
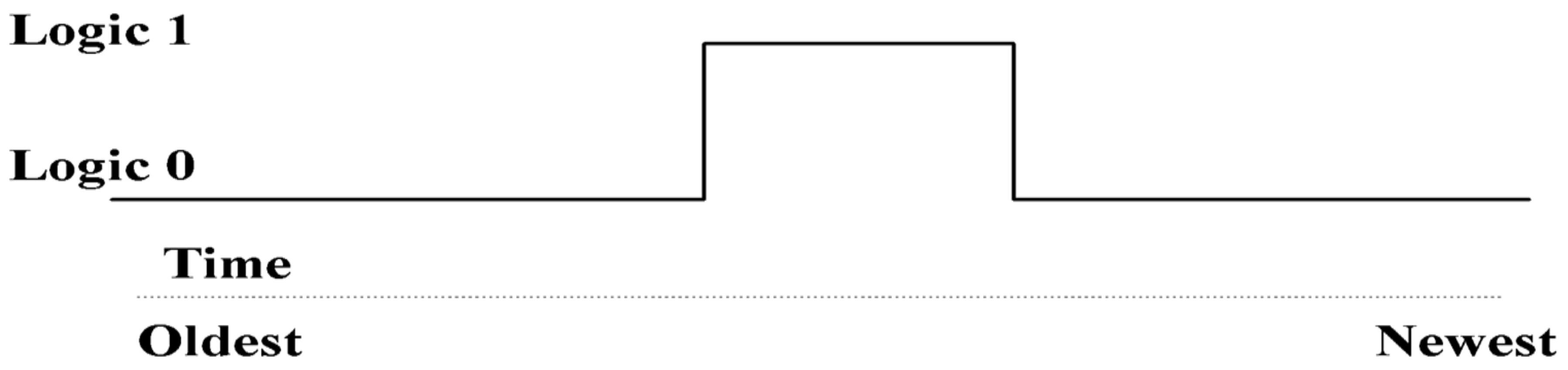


Fig: block diagram of sequential circuit

4. Timing Diagram

- To show what goes on in a logic circuit over time, we use **Timing Diagrams**
- A Timing diagram is a **Picture** of the effect of a **Wave Form**, passing through a Logic Gate.
- The changes of a logic input signal with respect to time.

4.1 Timing Example – A Pulse

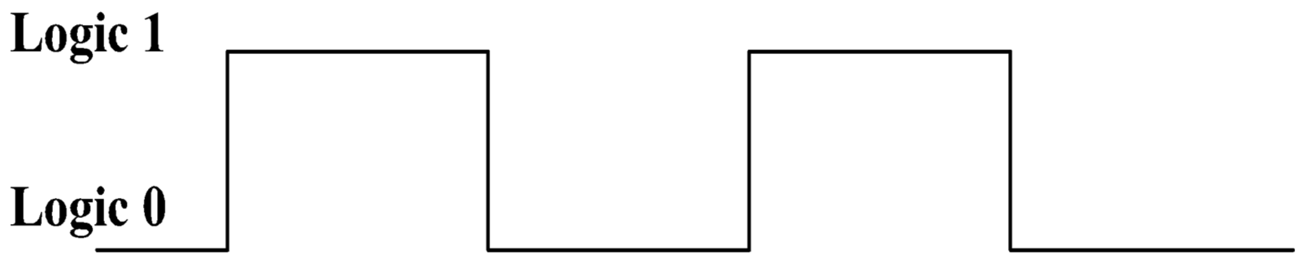


- The signal level is at a logical 0 before the pulse.
- It then rises to a logical 1 at the rising edge.
- It stays at a logical 1 for a period of time (the pulse duration).
- It then falls back to a logical 0 the falling edge.

4.2 A Clock

- A clock has a repeating wave form (or pulse).
- The period of a clock is measured from one rising edge to the next rising edge.
- The frequency of the clock is the number of complete periods in a Second. (Hz).
- The pulses of a CPU clock control its operation.

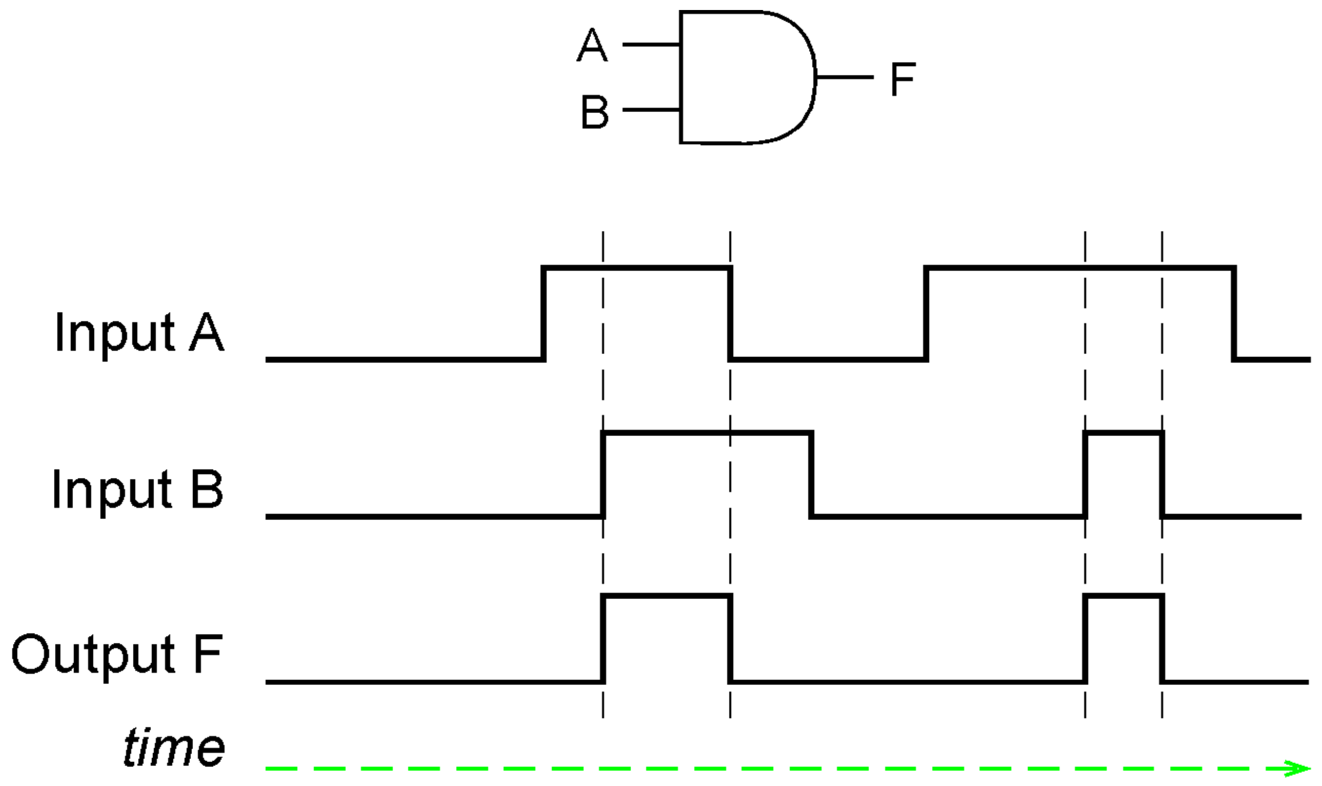
  i.e. A Clock Synchronises Operations in CPU.



4.3 Nano seconds

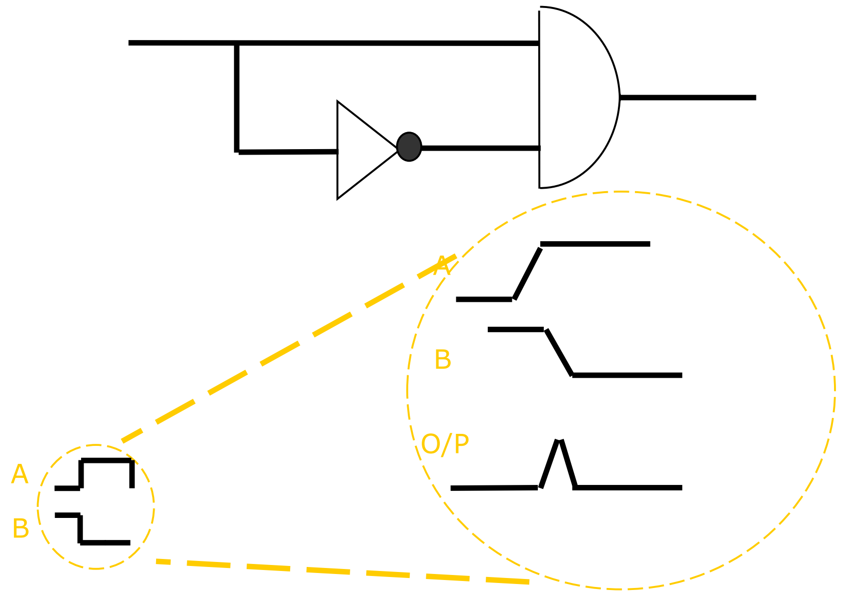
- The timing diagrams show time to any level.
- If the clock waveform on the previous slide was for an Intel Pentium 3.2, the diagram would show 1.56 X 10^{-9} seconds (1.56 **nano** seconds)
- A typical TTL logic gate takes up to 4 Nano seconds to go from 0 to 1.

4.4 Timing Diagram for AND Gate



5. Race Hazard

- A race hazard occurs when an unintended spike (very short pulse ~4 to 10 nano seconds) causes an unexpected and undesired change of output to occur
- Output should always theoretically be 0 (Boole's Law), but transition time can cause a spike

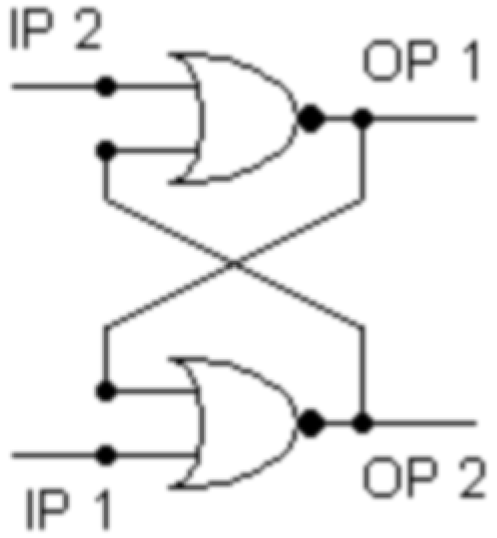


6. An Interesting Circuit

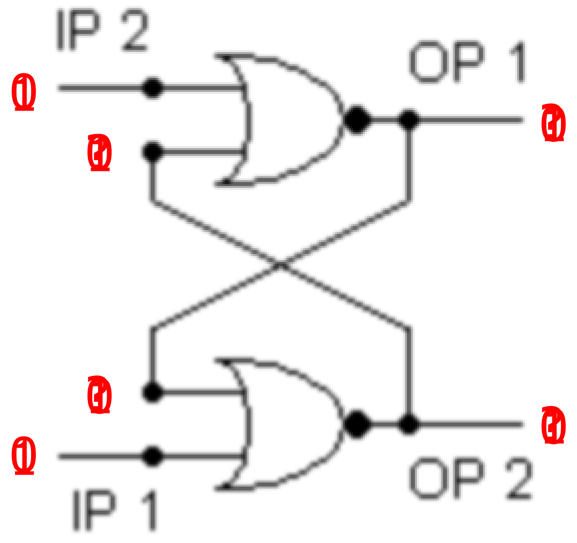
- What do you think this circuit does?

NOR Truth Table

A	B	O/P
0	0	1
0	1	0
1	0	0
1	1	0



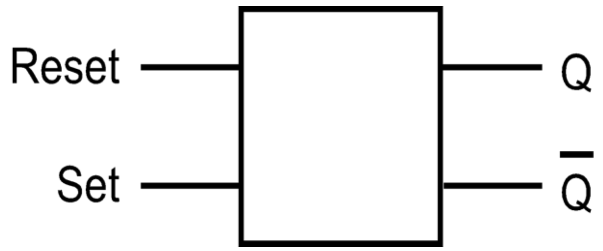
6. An Interesting Circuit



NOR Truth Table

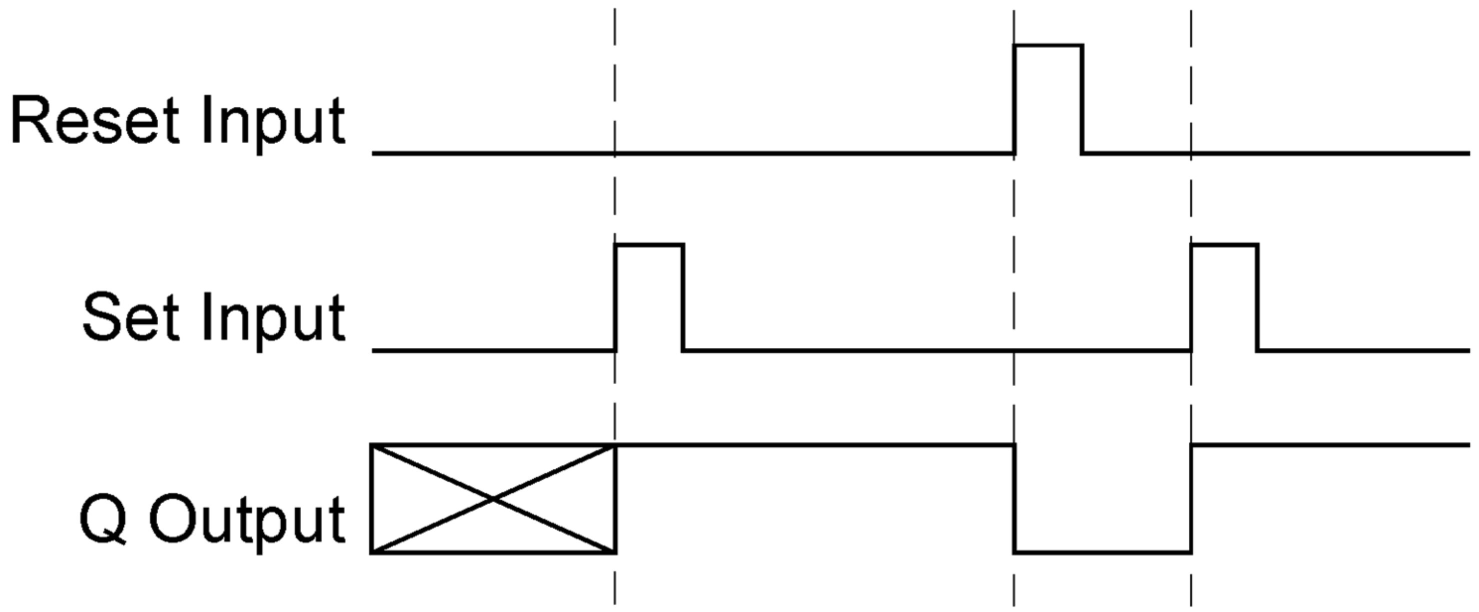
A	B	O/P
0	0	1
0	1	0
1	0	0
1	1	0

7. Simple Memory Block



- Initially Set and Reset are at Logic 0
  - Output Q $\overline{Q}$ at this point are not known.
  - Putting a pulse on the **Set** input causes the **Q** output to become 1.
  - Putting a pulse on the **Reset** causes the **Q** output to become 0.
- At rest, both inputs 0, the circuit remains in the same state. It **remembers** what happened last.

8. Timing For R-S Flip-Flop



Any pulse on the SET input will ensure that the Q output is at a logic '1’.

Any Pulse on the RESET input will ensure that the Q output is at a logic '0‘.

8.1 R-S Flip-Flop Truth Table

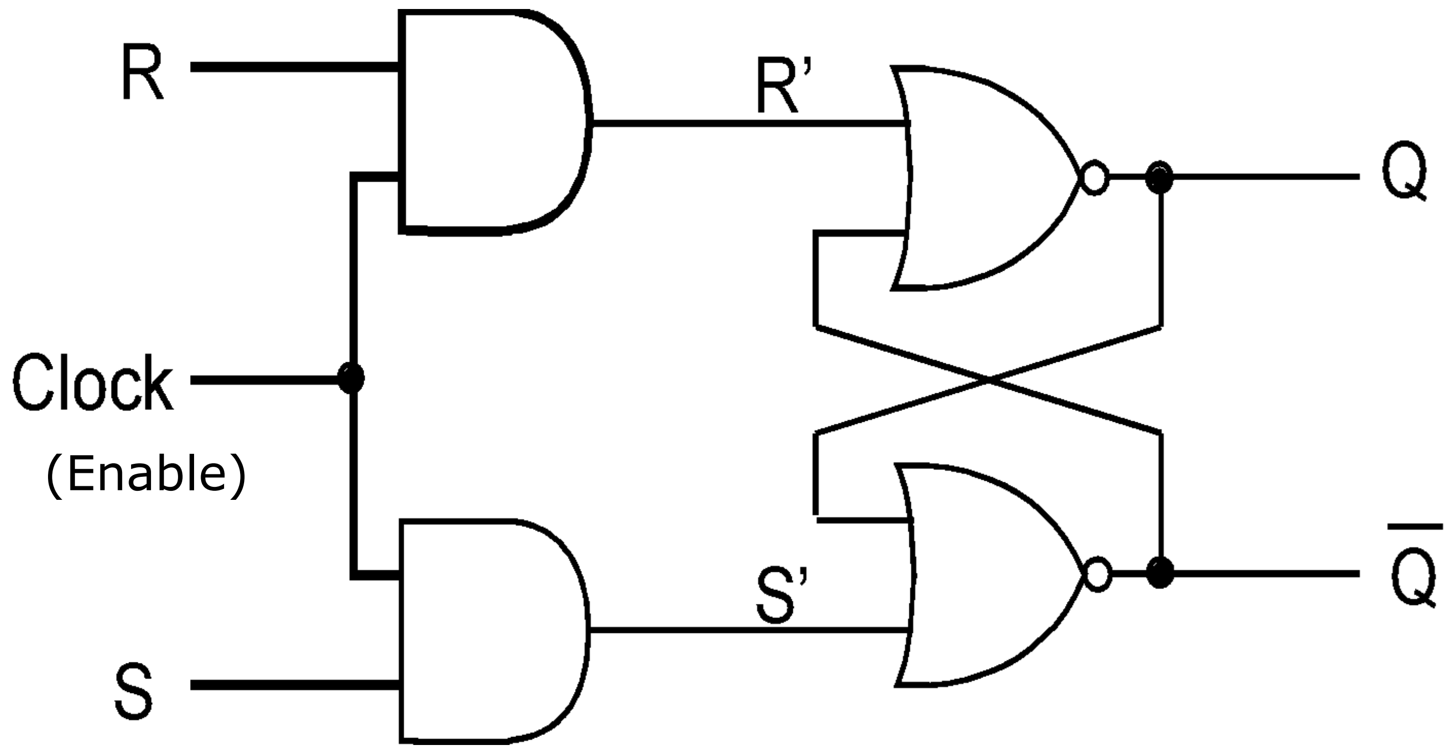
- Applying a pulse to both 'Set' and 'Reset' should never occur.
- Therefore, this is not defined in the truth table
- Note: n = time, n + 1 = time plus 1 (i.e. 'Next')

S	R	Q_{n+1}	$\bar{Q}_{n+1}$
0	0	Q_n	$\bar{Q}_n$
1	0	1	0
0	1	0	1
1	1	X	X

8.2 More Practical Memory

- The R-S Flip-Flop does not offer a very practical memory.
- It changes whenever the R or the S input is pulsed.
- In a larger circuit, we need to synchronise the changes on the output.

8.3 Clocked R-S Flip-Flop

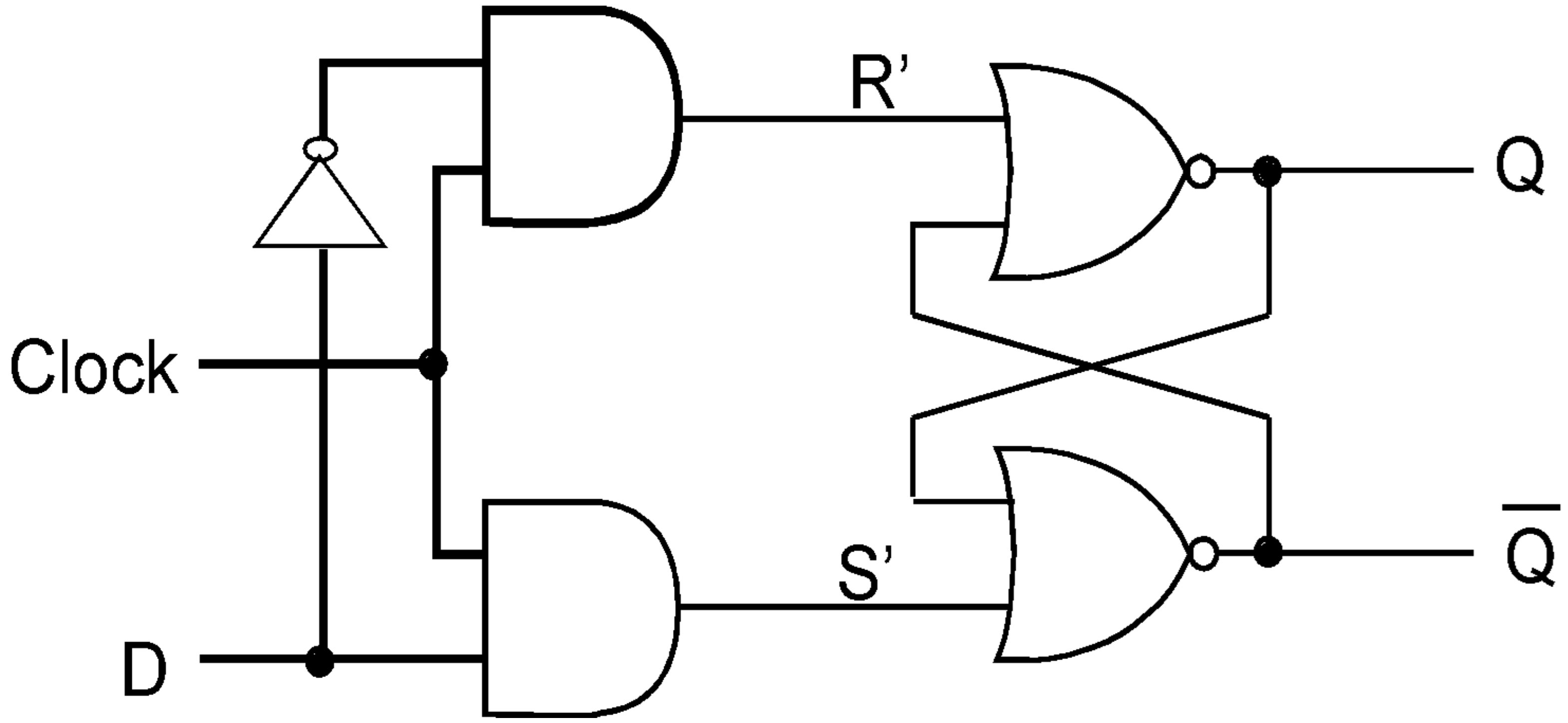


Changes are synchronized by the clock signal. **R or S** signal will only move to the flip flop when the clock pulse is high.

8.4 Memory Input

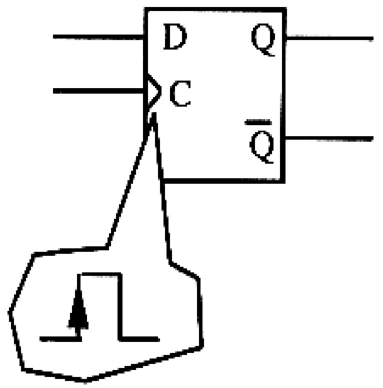
- Having 2 separate lines, Set and Reset, to store 1 bit of memory is inconvenient.
- To use the memory on a CPU data bus, we much rather use one input to store the 1 bit.

9. D-type Flip-Flop

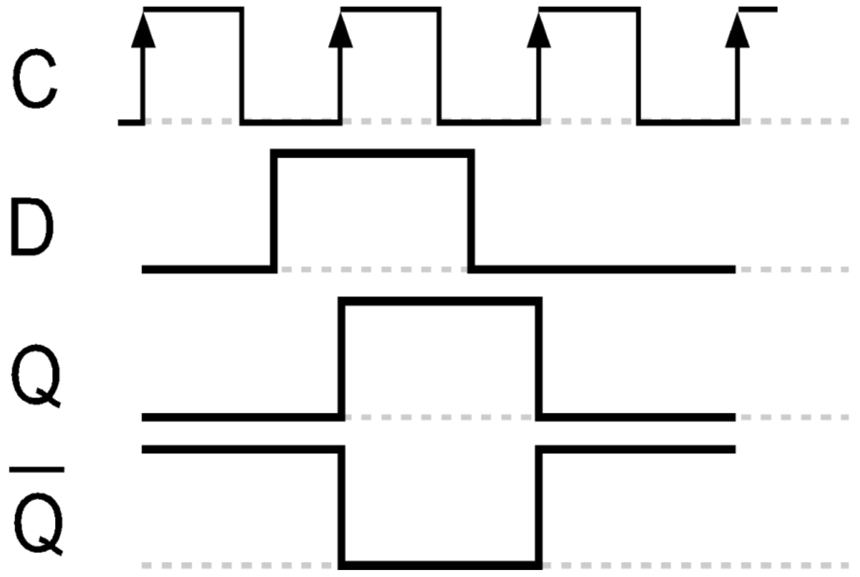


Input D is transferred to output Q. If D (for data) is 1, what gets stored at Q? And if D is 0?

9.1 D-type Flip-Flop (rising edge clock)

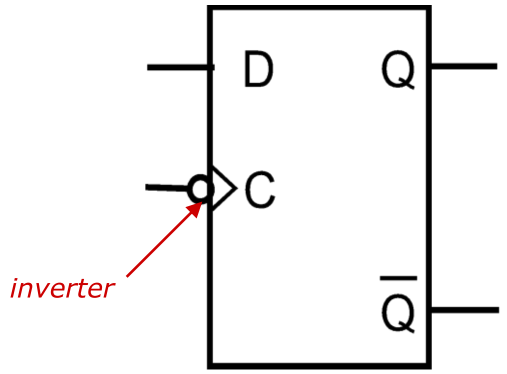


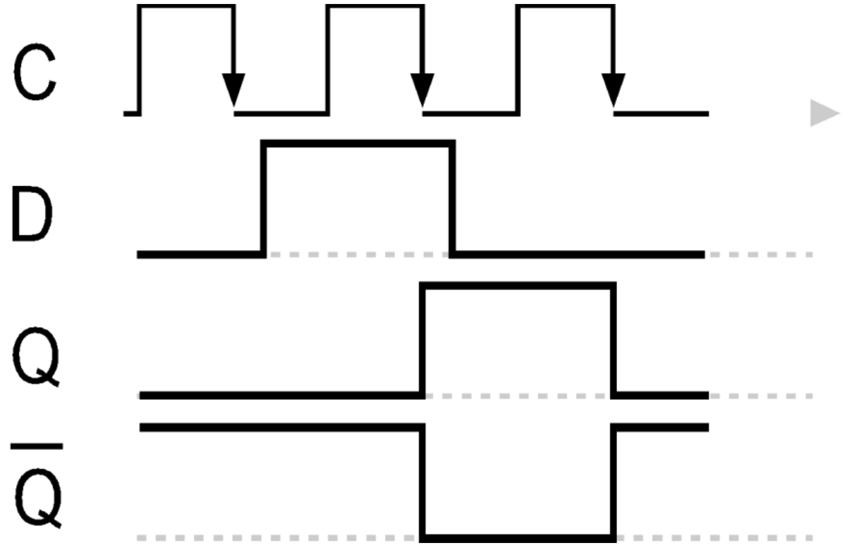
Clock C transfers data D to Q on the rising edge of pulse.



Timing Diagram

9.2 D-type Flip-Flop (falling edge clock)

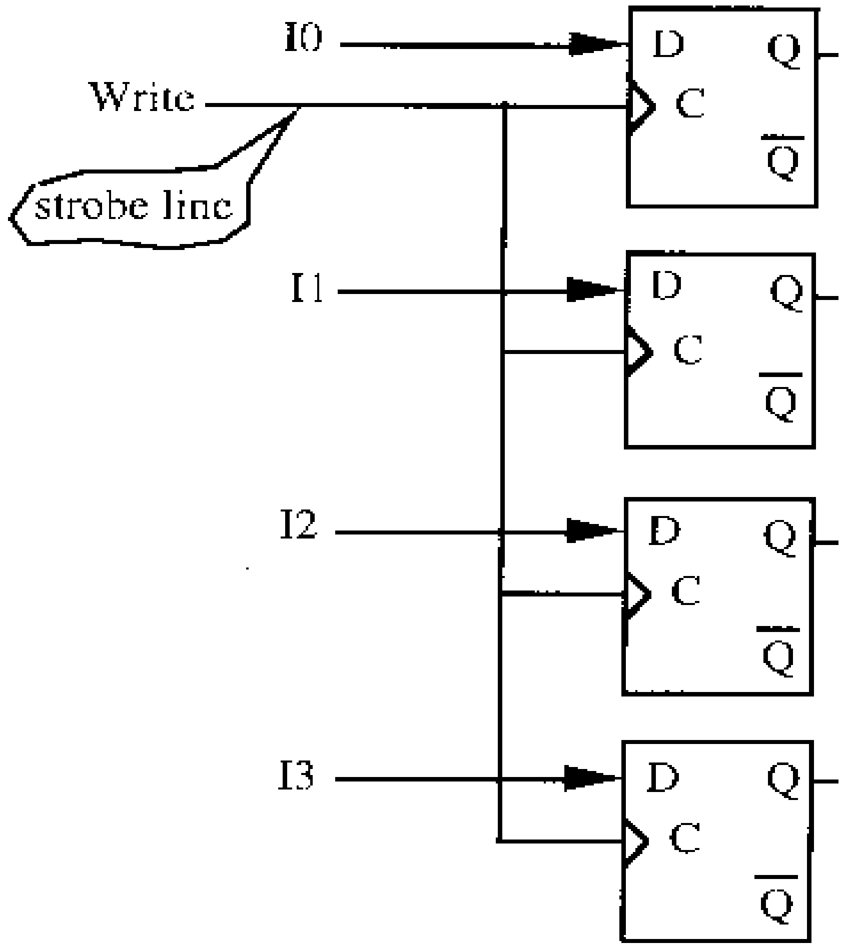




Clock C transfers data D to Q on the falling edge of pulse.

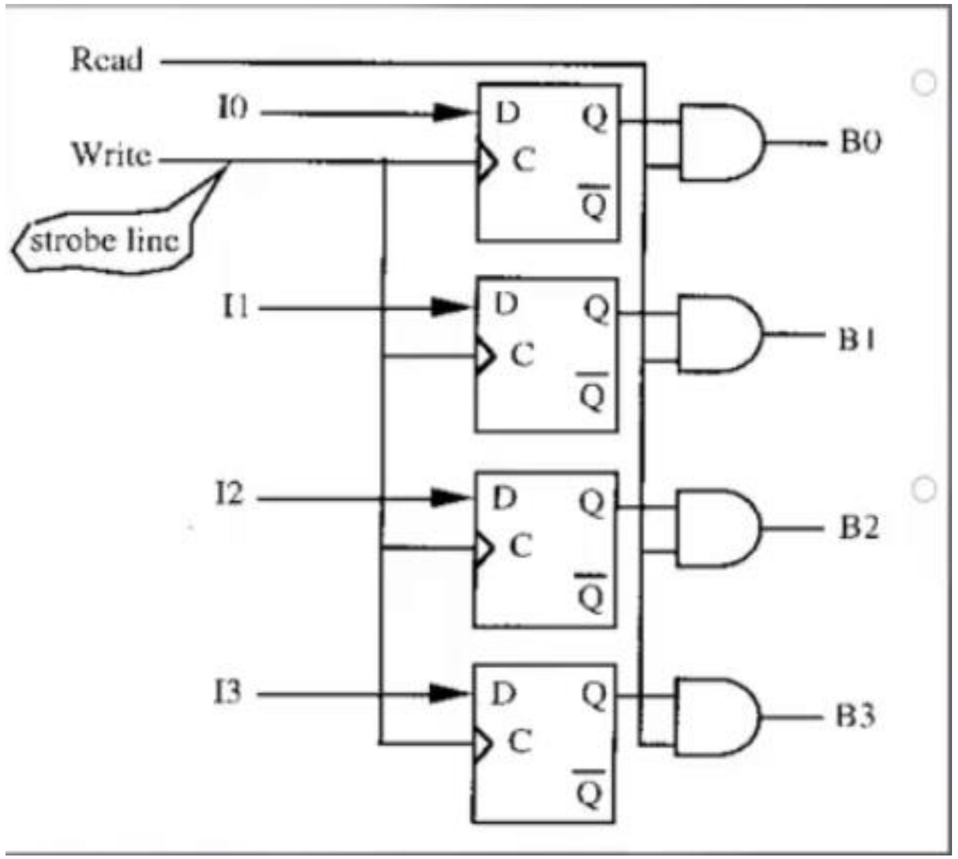
Timing Diagram

10. 4-bit Register



When the Write is pulsed, the Input I0 to I3 are transferred to the register.

10. 4-bit Register (Contd.)

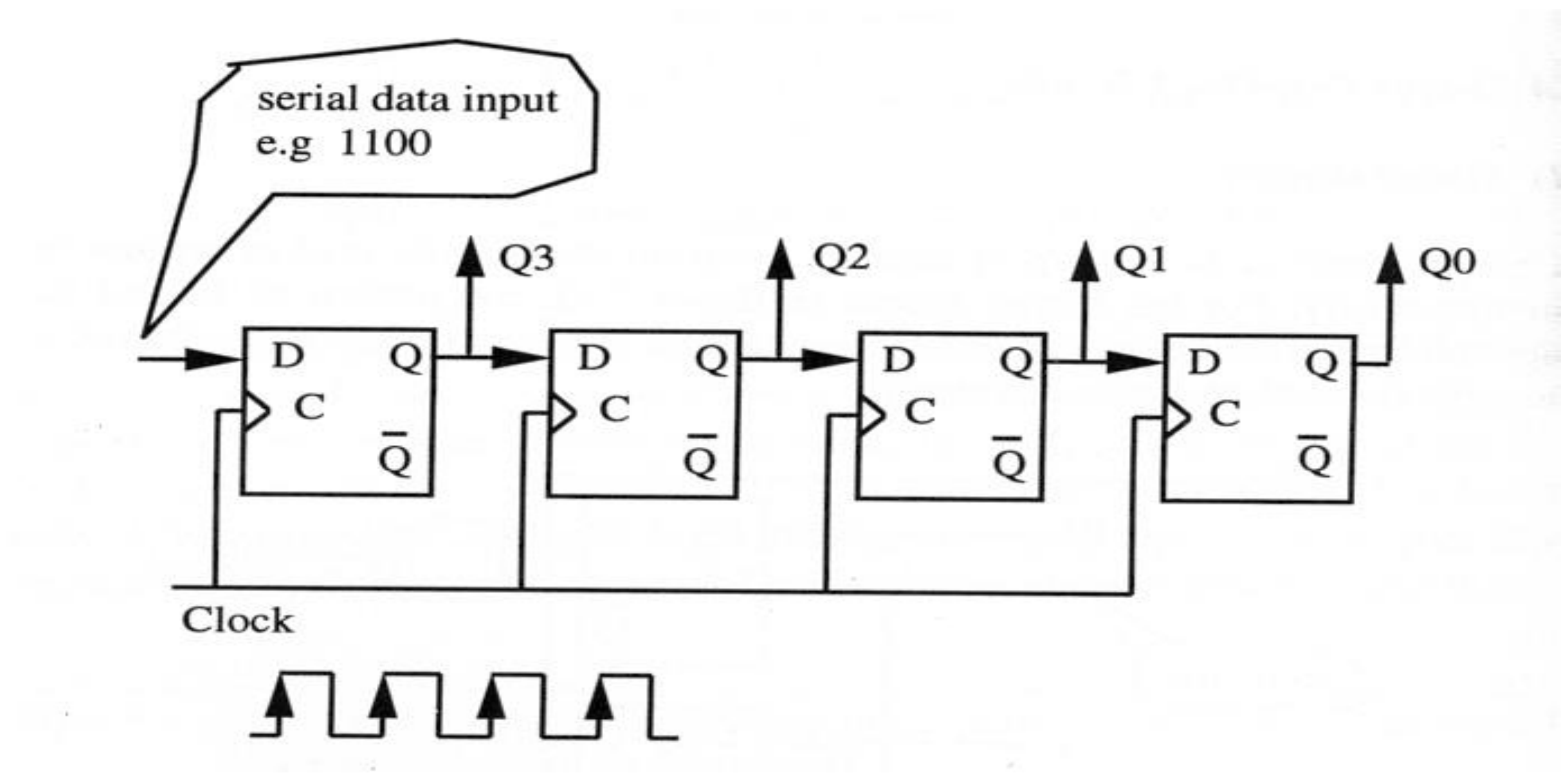


When the Write is pulsed, the Input I0 to I3 are transferred to the register.

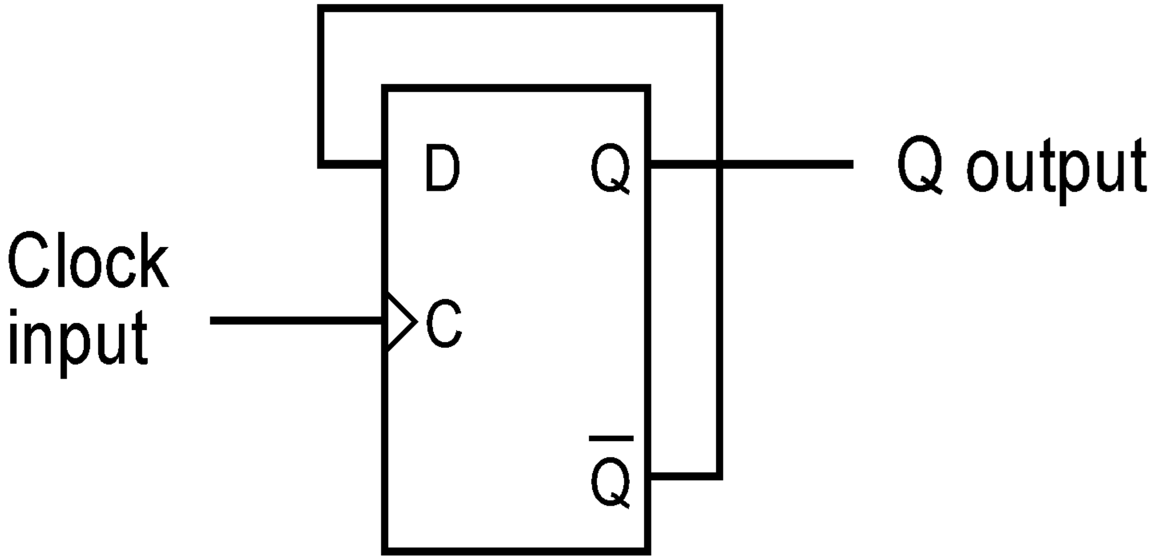
When the Read is pulsed, the register contents appear on the bus lines B0 to B3.

11. A Shift Register

- Converts Serial Data to Parallel Data

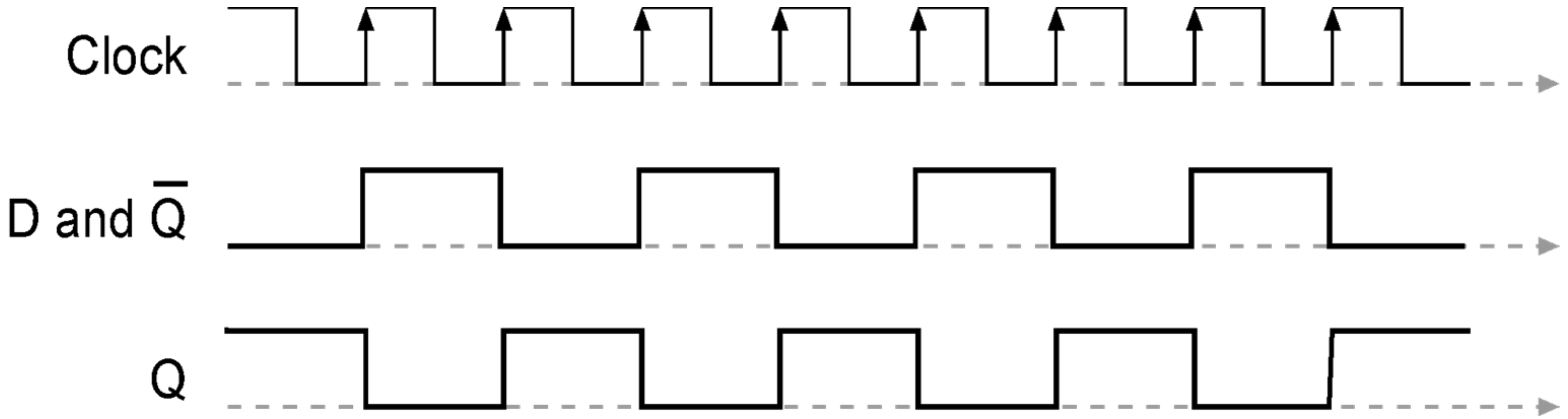


12. The Toggle D-type



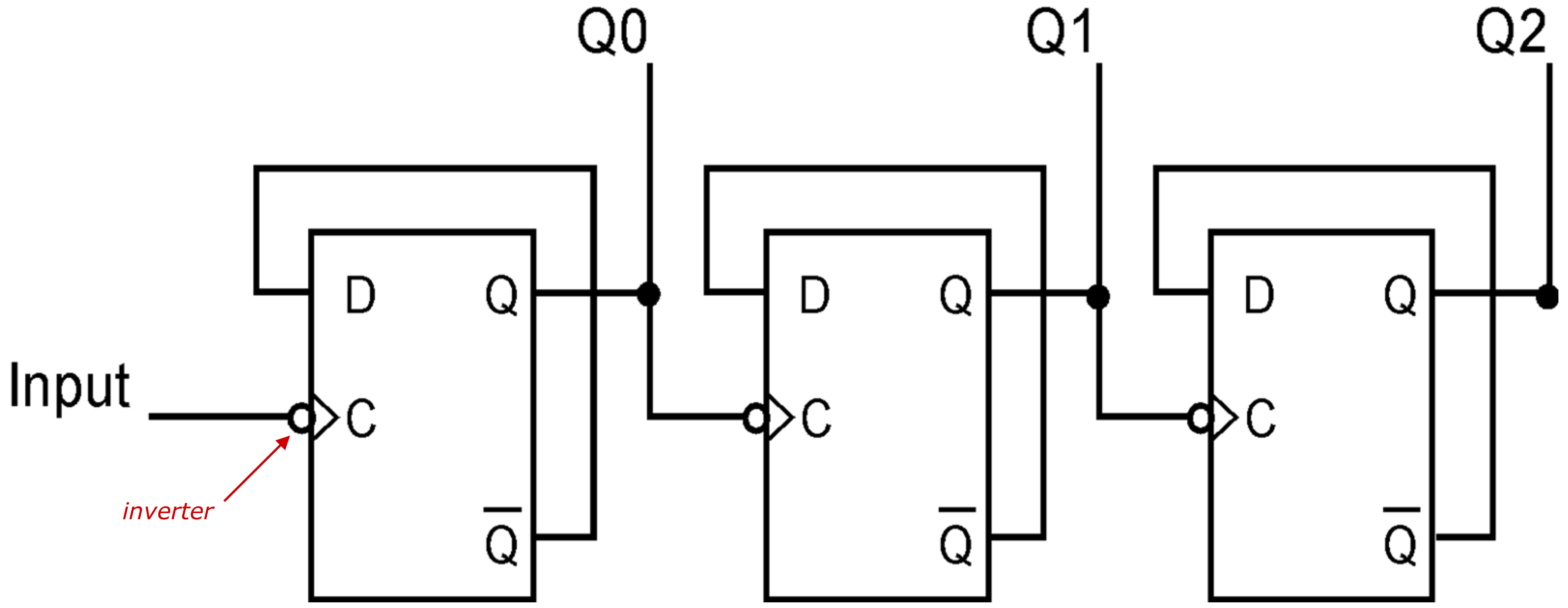
- The $\overline{\text{Q}}$ output is fed back to the ‘D’ input.
- The Q output toggles between ‘0’ and ‘1’ with each clock cycle.

12.1 Timing on a Toggle



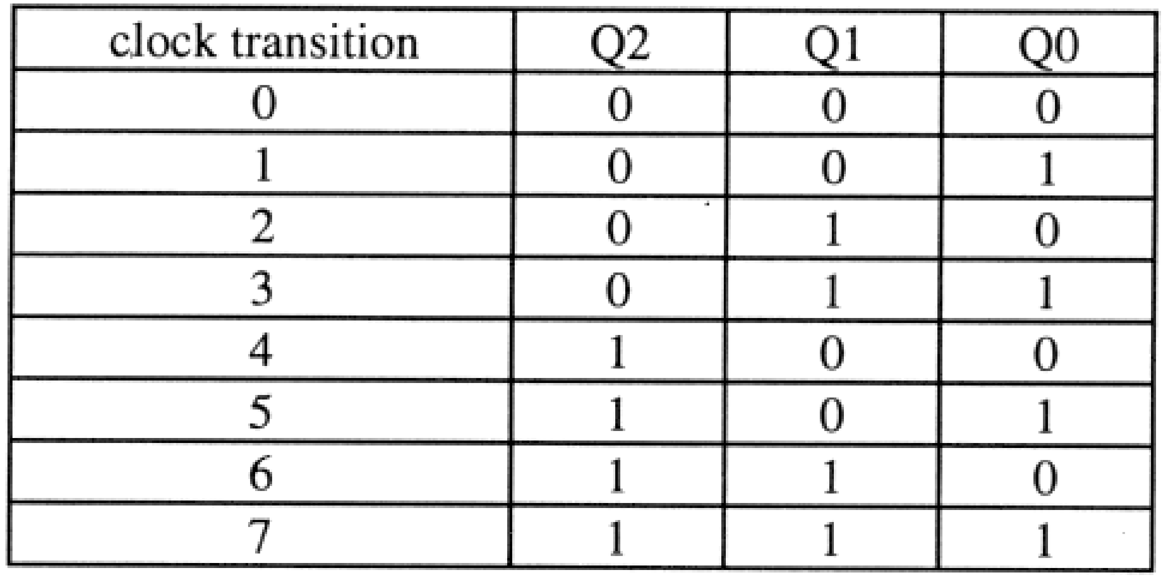
12.3 A Counter using Toggles

- 3 Bit Counter:



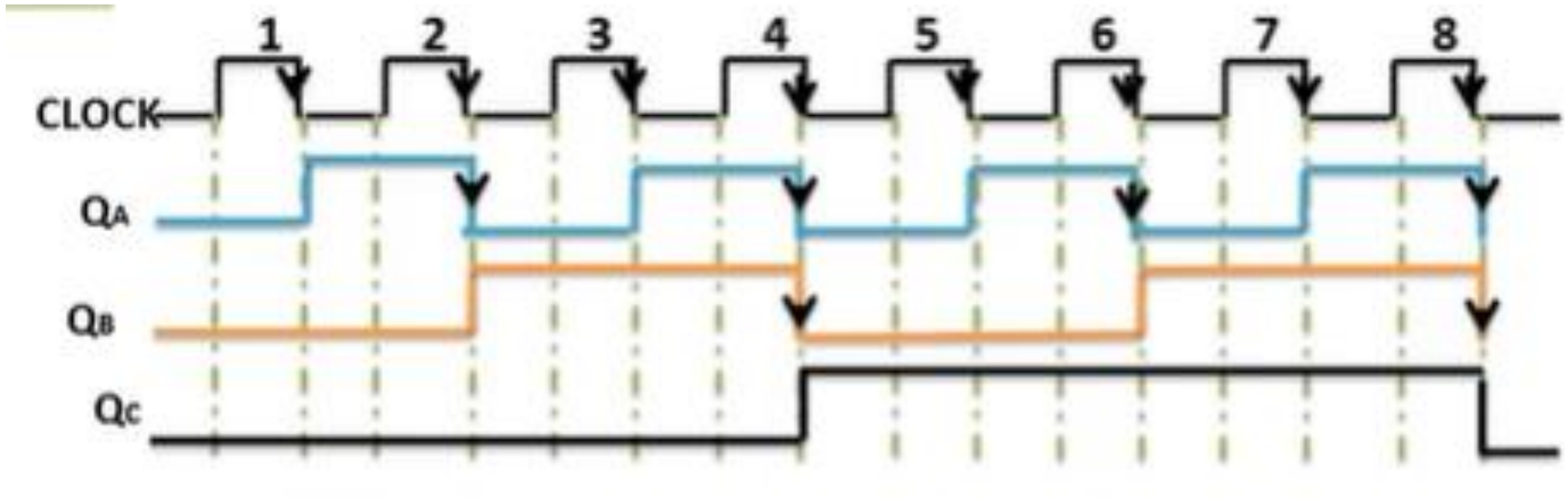
12.2 A Binary 3-bit Up Counter

- The required output is:



clock transition	Q2	Q1	Q0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

12.4 Timing of a Counter



13. Summary

- Timing diagrams, pulses and clocks
- Logic feedback, simple memory, RS Flip-flops
- Clocked memory, D-type Flip-flops
- Memory registers, Shift registers
- Counters