Discussion 7: ALU & FSM

PEIFAN LI

Recall

Synchronous Digital Systems consist of two basic types of circuits:

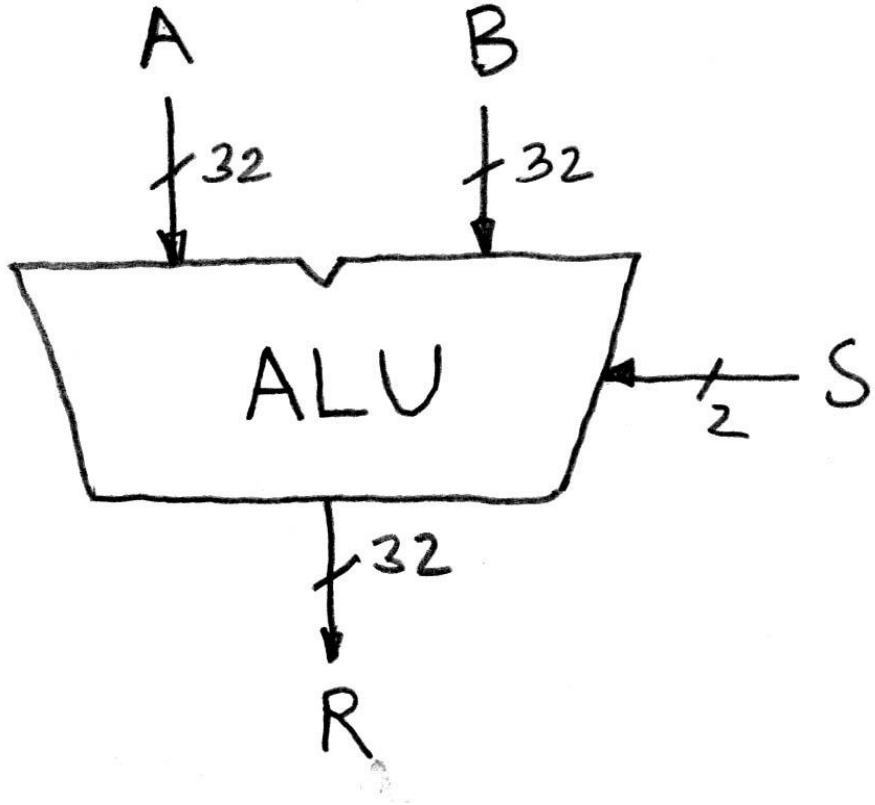
- Combinational Logic (CL) circuits
  -Output is a function of the inputs only, not the history of its execution
  -E.g. ALU(add, mul, sll…)

- Sequential Logic (SL) circuits
  -Circuits that “remember” or store information
  -E.g. memories and registers

ALU (Arithmetic and Logic Unit)

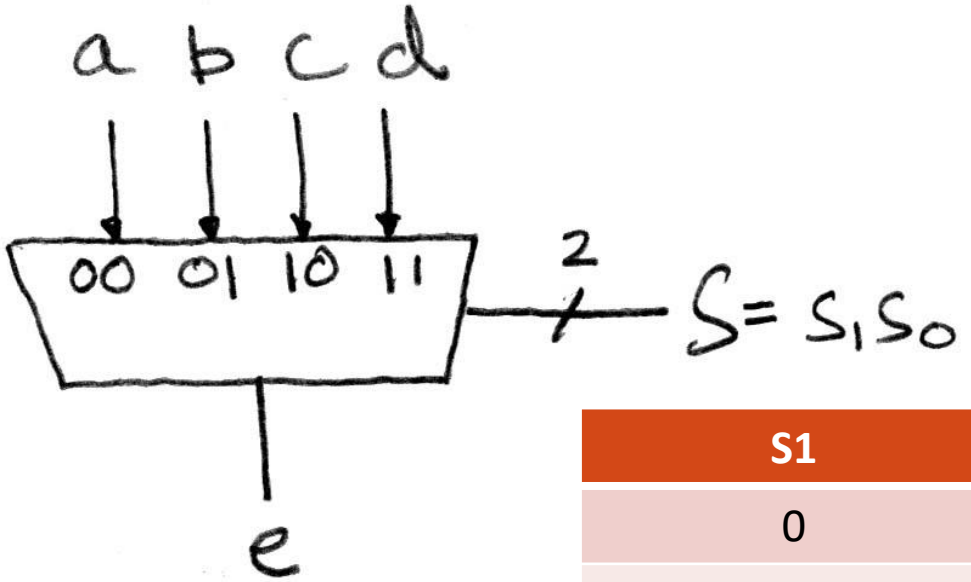
- Arithmetic operation and Logical operation
- Simplest example: 32bit, 2 function
- S, control
- E.g.

when S=00, R=A+B
when S=01, R=A-B
when S=10, R=A AND B
when S=11, R=A OR B



multiplexer

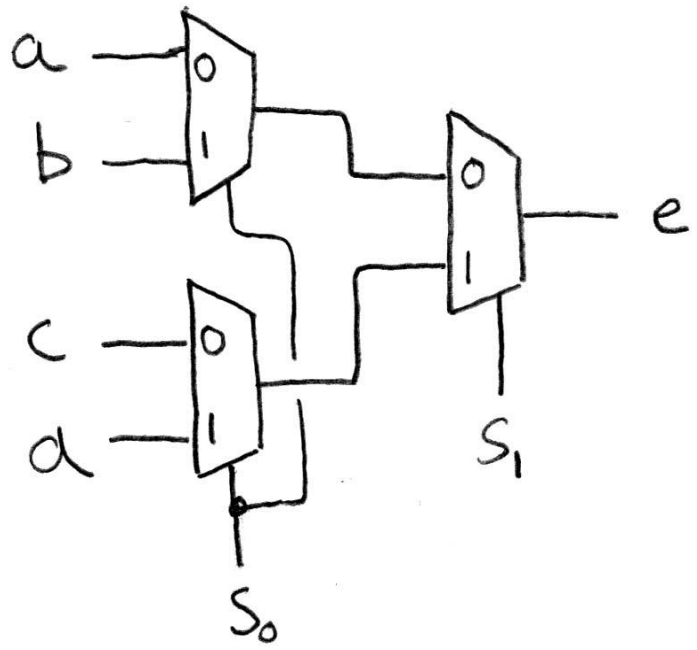
For a 4-to-1 multiplexer, S is a two-bit string



S1	S0	S	e
0	0	00(0)	a
0	1	01(1)	b
1	0	10(2)	c
1	1	11(3)	d

multiplexer

- Another way: Hierarchy



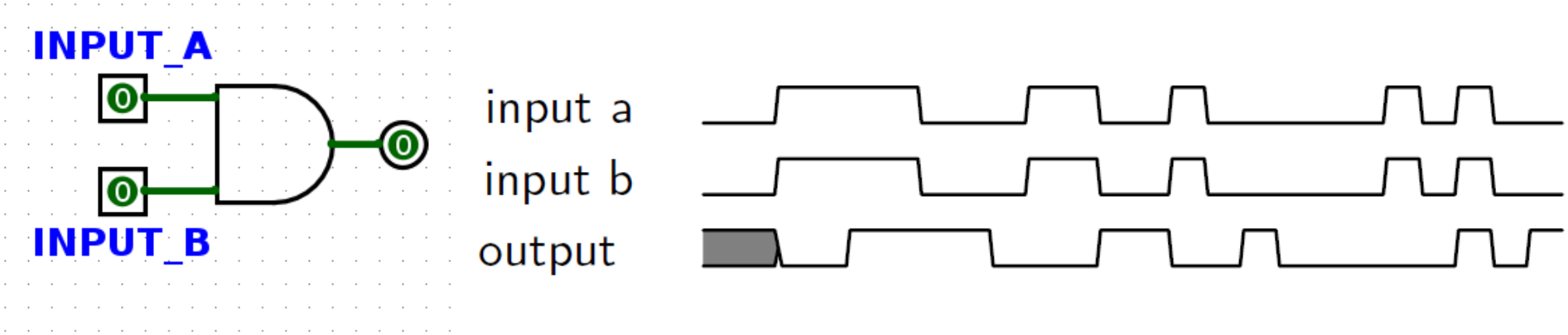
$$e = \overline{s_1 s_0} a + \overline{s_1} s_0 b + s_1 \overline{s_0} c + s_1 s_0 d$$

Combinational Logic

- Combinational logic circuits simply change based on their inputs after whatever propagation delay is associated with them.

Combinational Logic

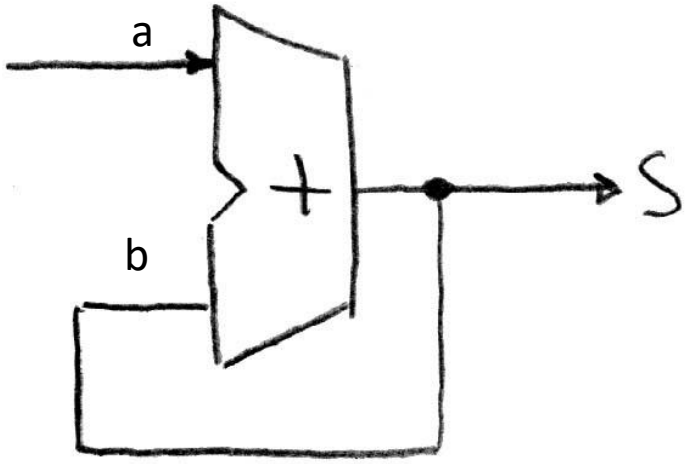
Suppose the AND gate has an associated propagation delay of 2ps



Sequential Logic

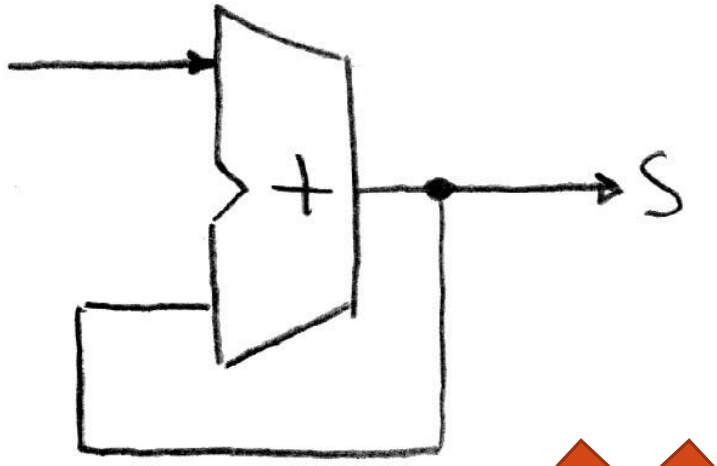
Accumulation operation

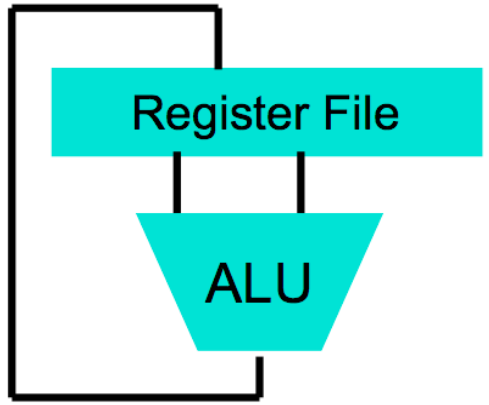
1+2+3+4…



Sequential Logic

Operation like accumulation needs to store information



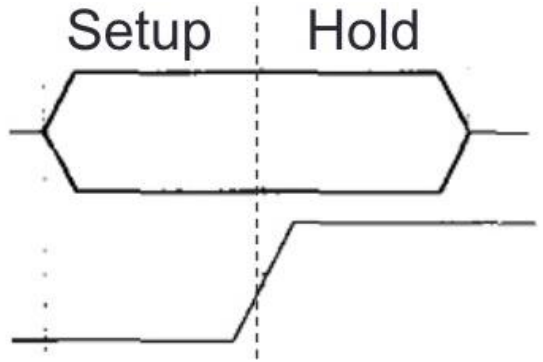


Sequential Logic

- Sequential Logic can remember their inputs even after the inputs change.
- State elements change value based on a clock signal.

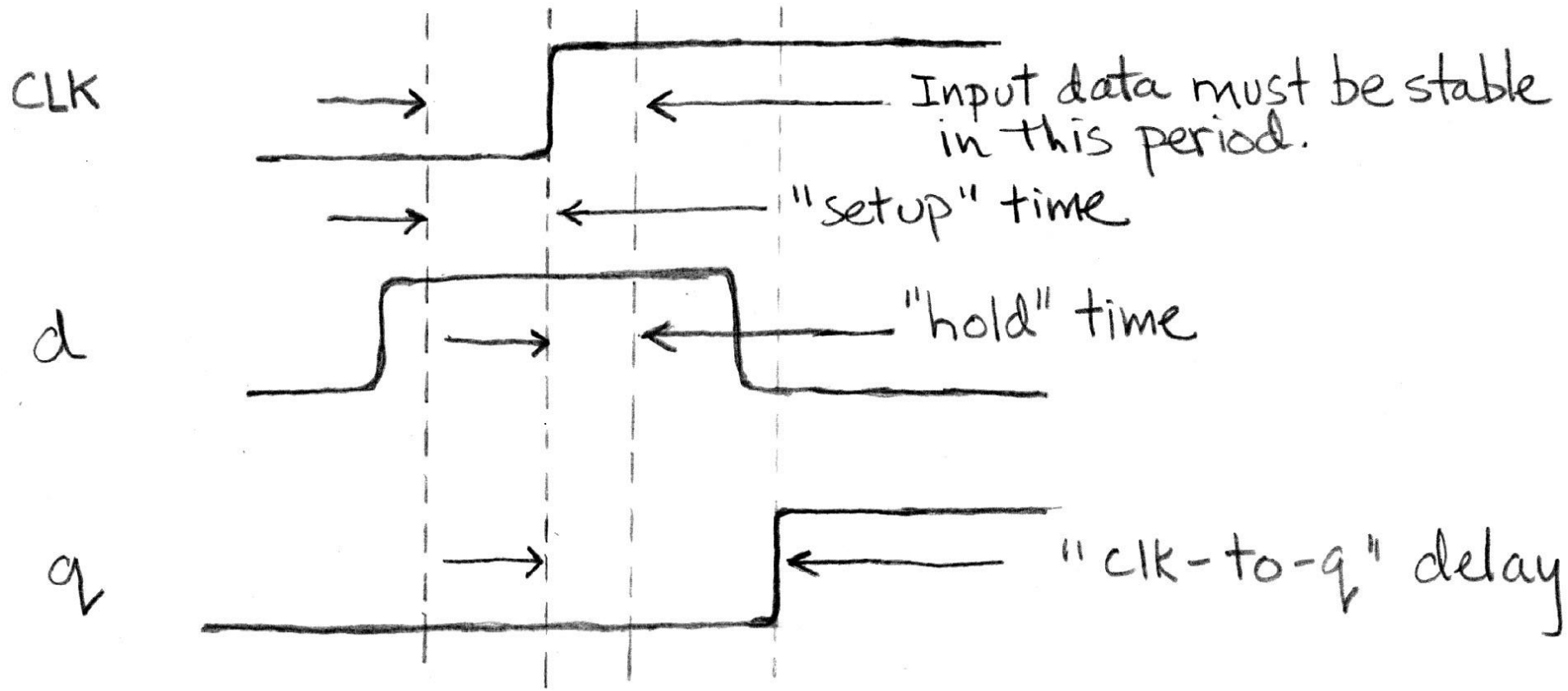
Sequential Logic

- The input to the register samples has to be stable for a certain amount of time around the rising edge of the clock for the input to be sampled accurately.



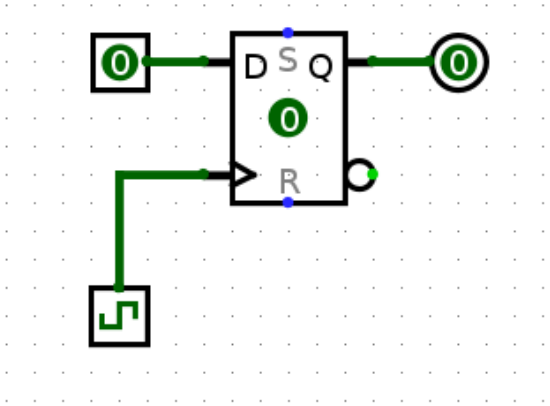
- Setup time: the amount of time before the rising edge the input must be stable
- Hold time: the time after the rising edge the input must be stable

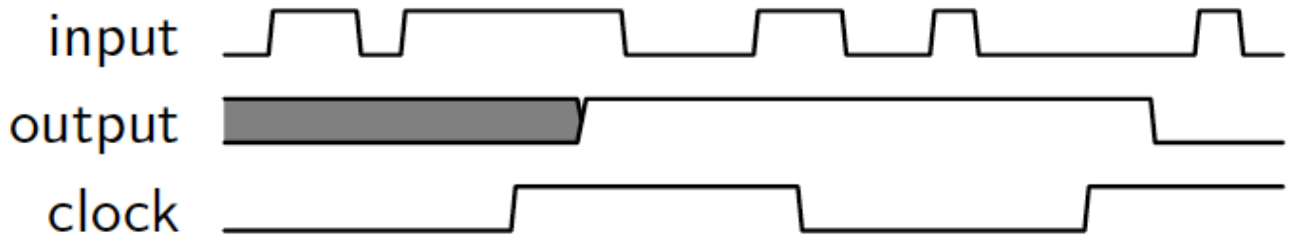
Sequential Logic



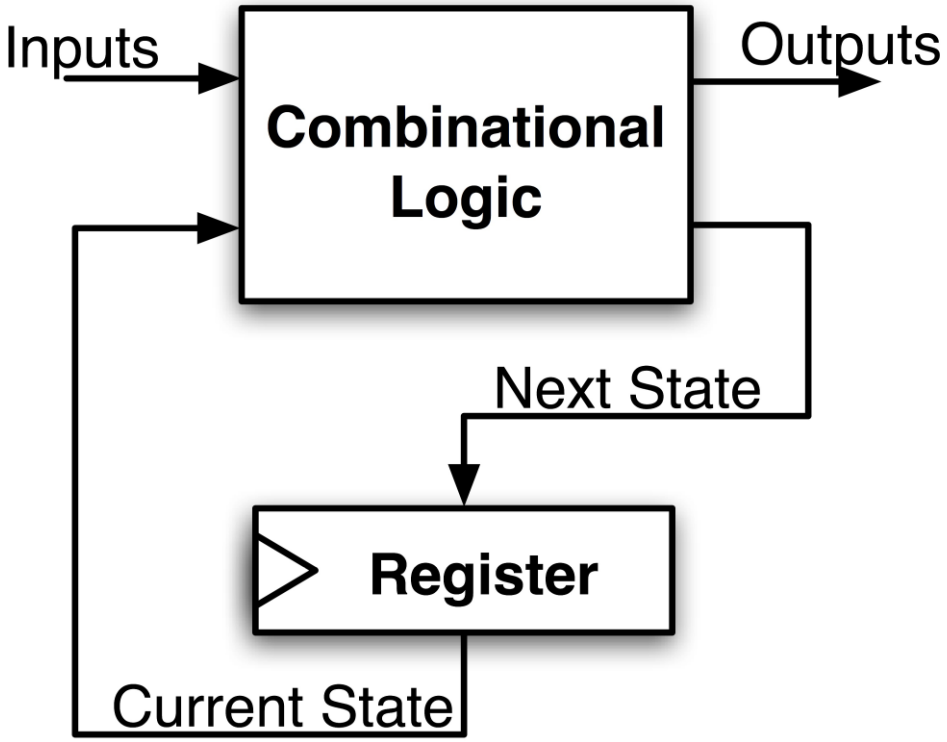
Sequential Logic

Assume **setup** of 2.5ps, **hold** time of 1.5ps, and a **clk-to-q** time of 1.5ps.
The clock signal has a period of 13ps.





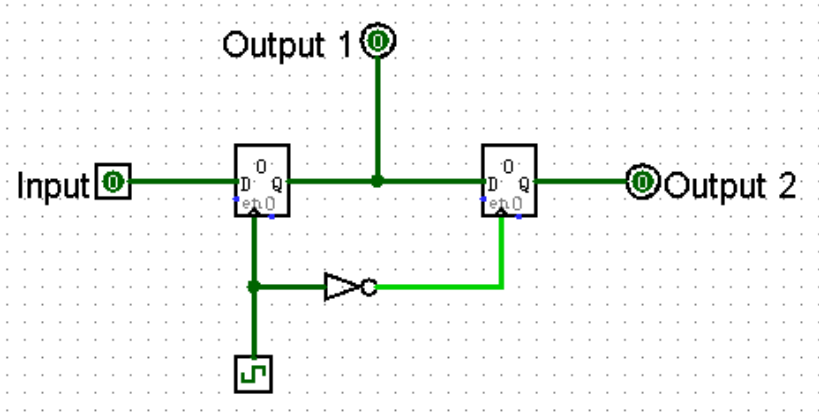
Max Delay

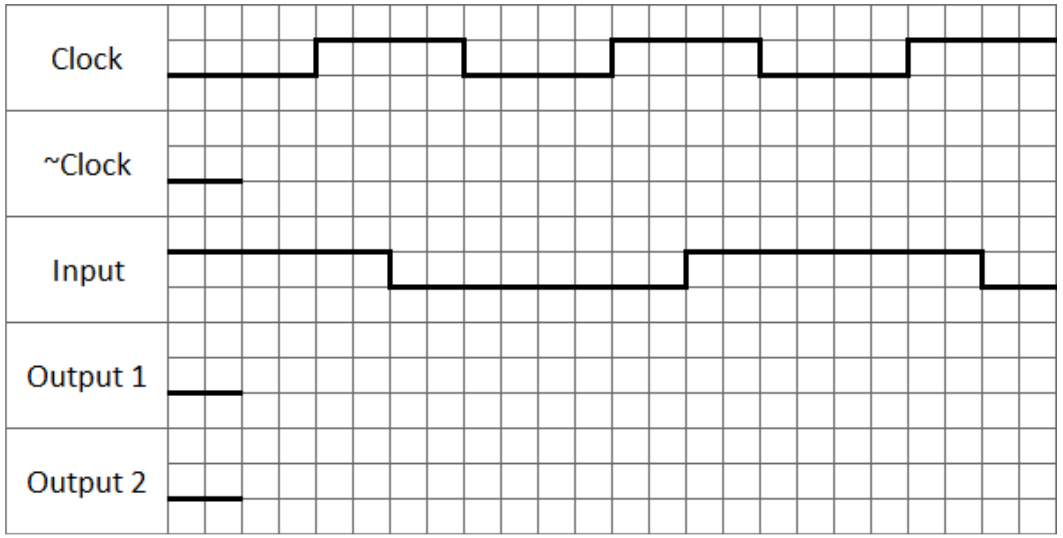


Max Delay = CLK-to-Q Delay + CL Delay + Setup Time

Example

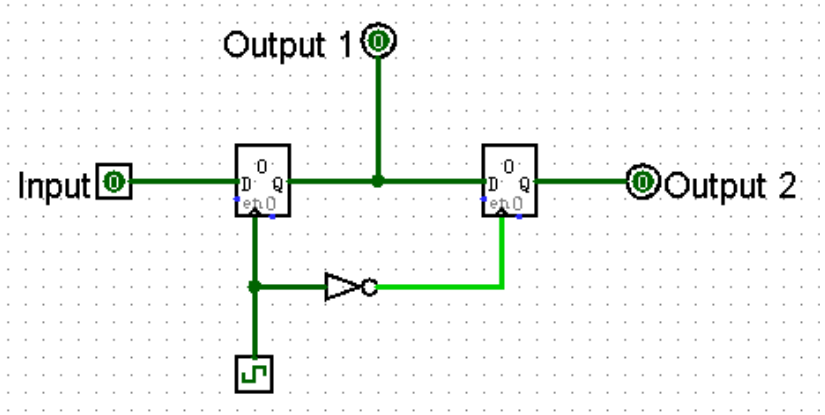
- NOT gates have a 2 ns propagation delay.
- For each register, the clk-to-q delay is 2 ns and setup time is 2 ns.
- The clock period is 8 ns, and each grid in the following diagram is a unit of 1 ns.

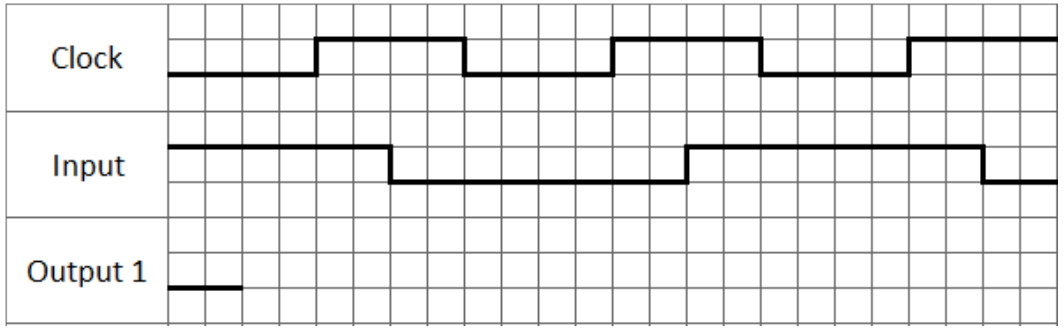




Example

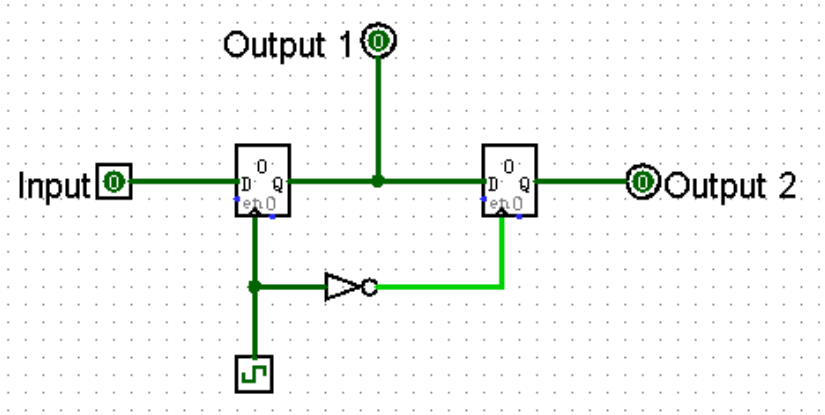
- NOT gates have a 2 ns propagation delay.
- For each register, the clk-to-q delay is 2 ns and setup time is 2 ns.
- The clock period is 8 ns, and each grid in the following diagram is a unit of 1 ns.

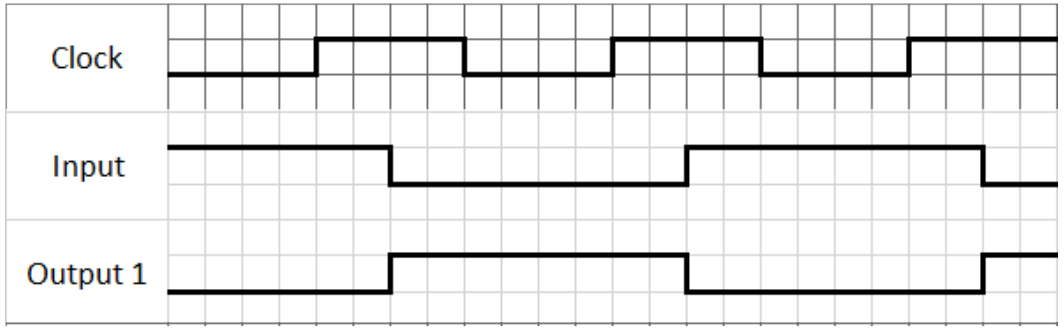




Example

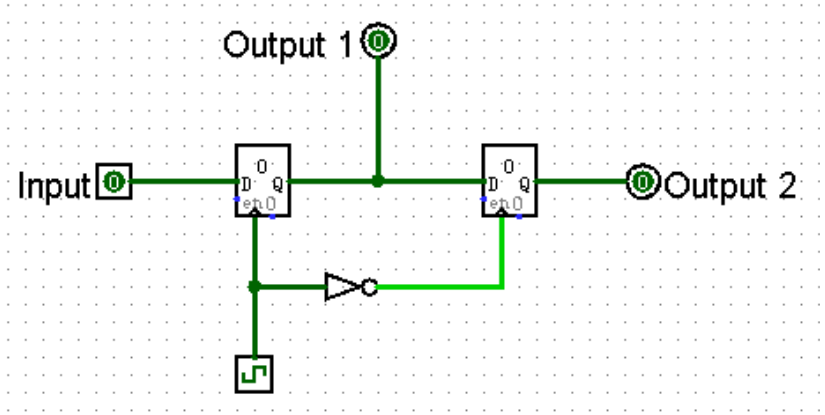
- NOT gates have a 2 ns propagation delay.
- For each register, the clk-to-q delay is 2 ns and setup time is 2 ns.
- The clock period is 8 ns, and each grid in the following diagram is a unit of 1 ns.

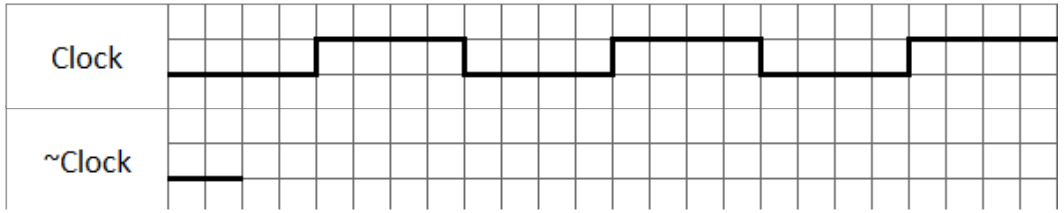




Example

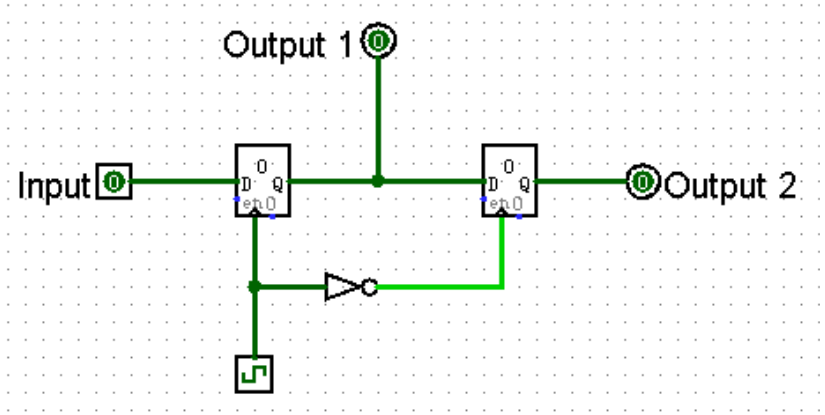
- NOT gates have a 2 ns propagation delay.
- For each register, the clk-to-q delay is 2 ns and setup time is 2 ns.
- The clock period is 8 ns, and each grid in the following diagram is a unit of 1 ns.

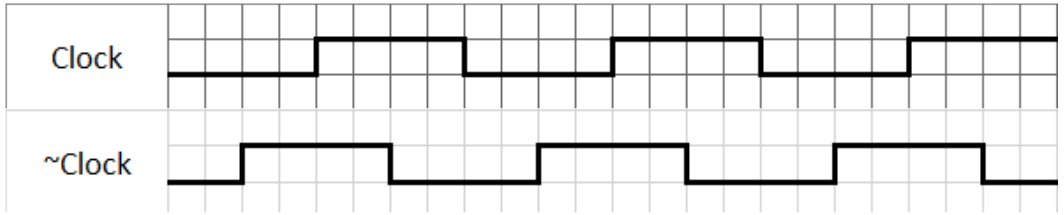




Example

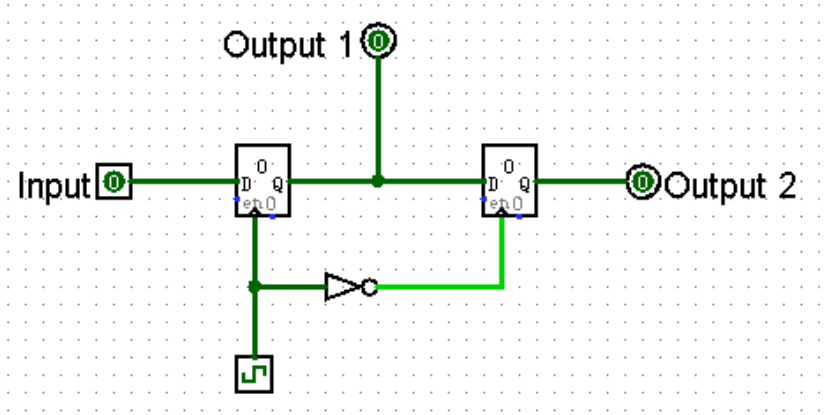
- NOT gates have a 2 ns propagation delay.
- For each register, the clk-to-q delay is 2 ns and setup time is 2 ns.
- The clock period is 8 ns, and each grid in the following diagram is a unit of 1 ns.

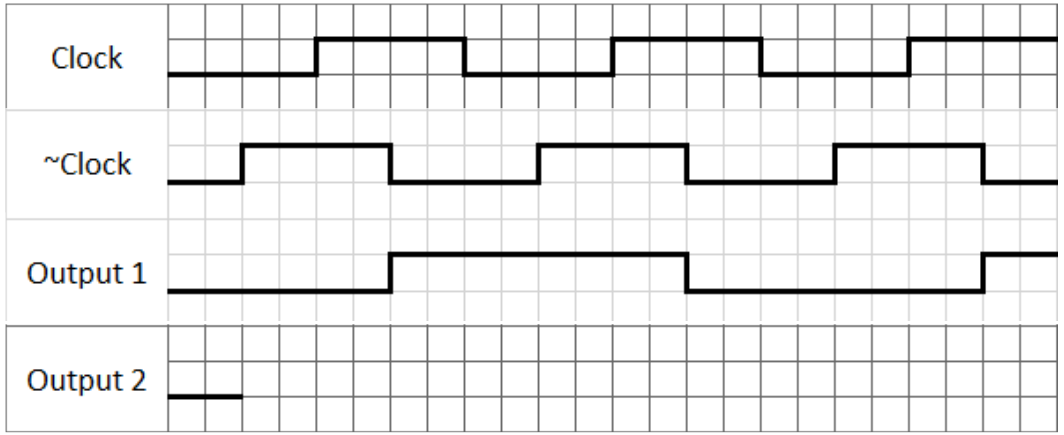




Example

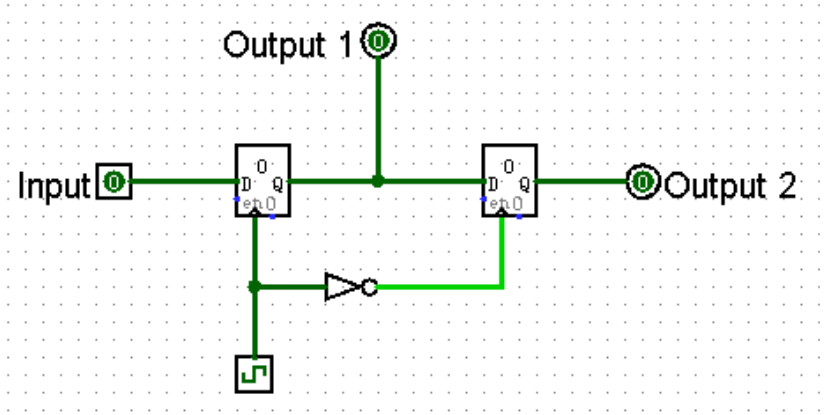
- NOT gates have a 2 ns propagation delay.
- For each register, the clk-to-q delay is 2 ns and setup time is 2 ns.
- The clock period is 8 ns, and each grid in the following diagram is a unit of 1 ns.

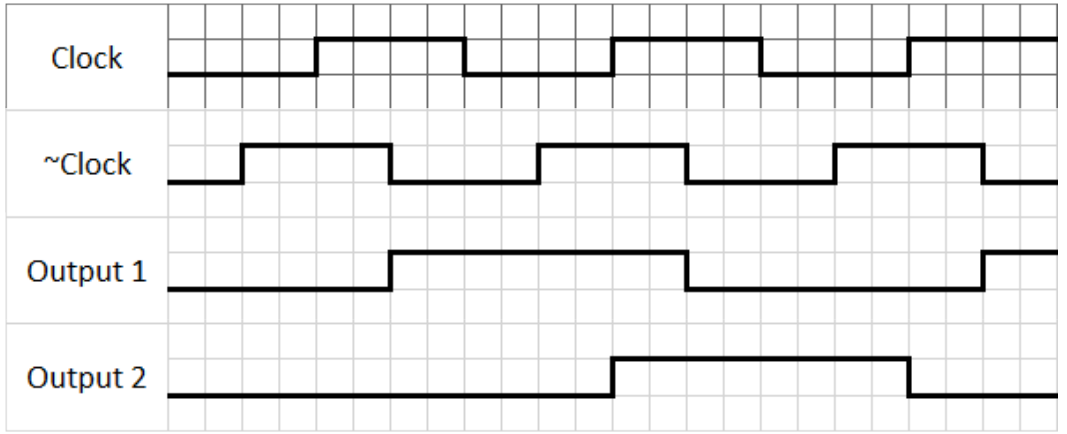




Example

- NOT gates have a 2 ns propagation delay.
- For each register, the clk-to-q delay is 2 ns and setup time is 2 ns.
- The clock period is 8 ns, and each grid in the following diagram is a unit of 1 ns.

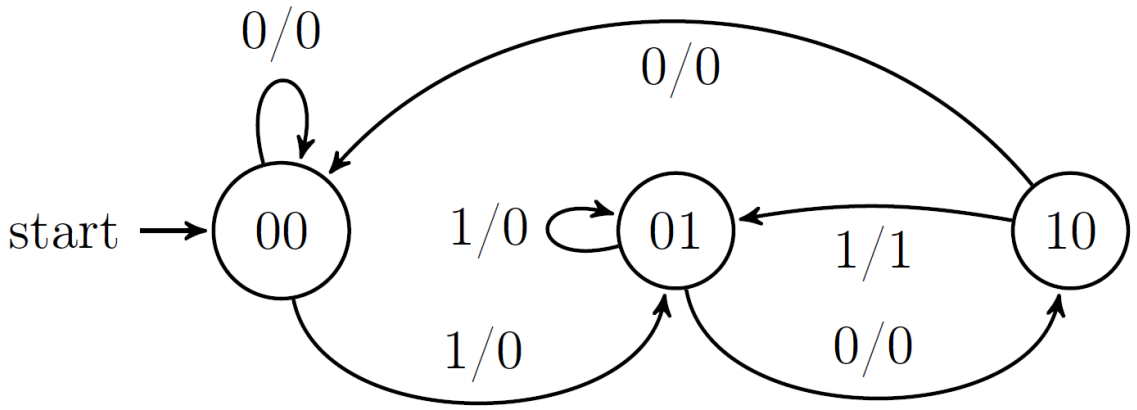




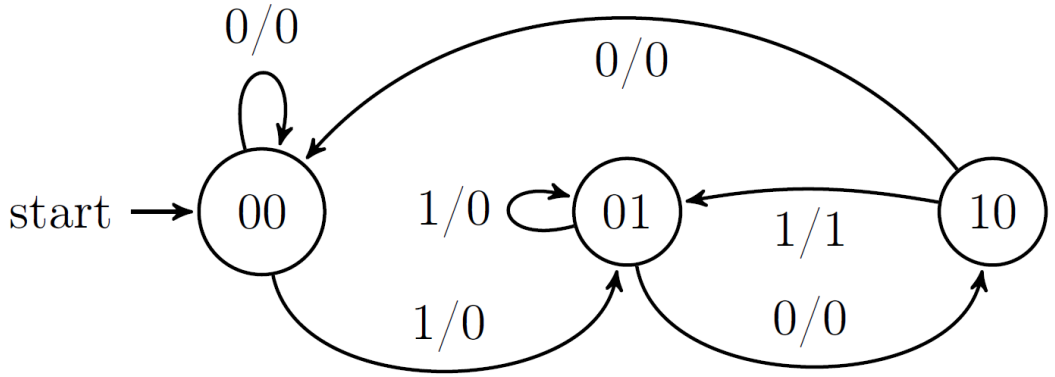
Finite State Machines

- A convenient way to conceptualize computation over time
- Start at a state, given an input, follow some edge to another
- With combinational logic and registers, any FSM can be implemented in hardware.
- edge: input/output

Example

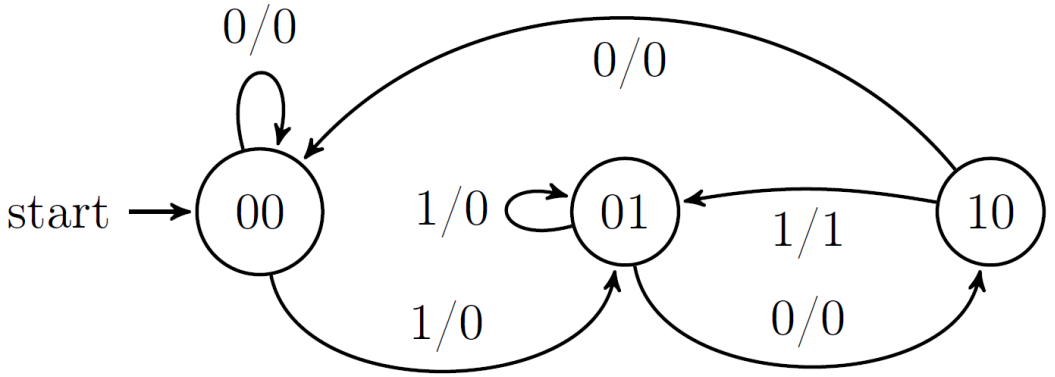


Example



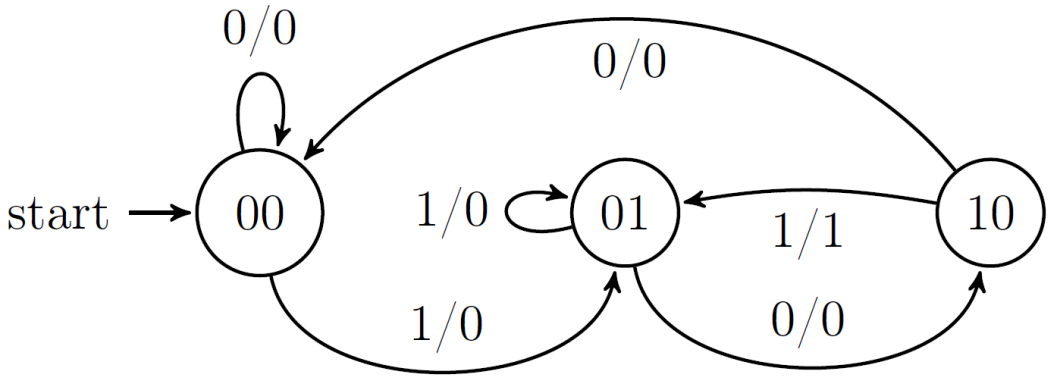
state bit1	state bit0	input	next state bit1	next state bit0	output
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			

Example



state bit1	state bit0	Input	next state bit1	next state bit0	Output
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	0	0	0
1	0	1	0	1	1

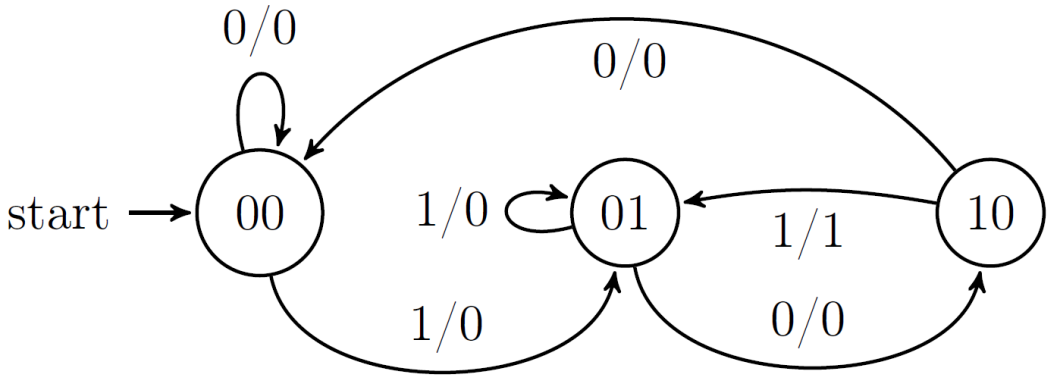
Example



Input ‘0100101010’

Output ?

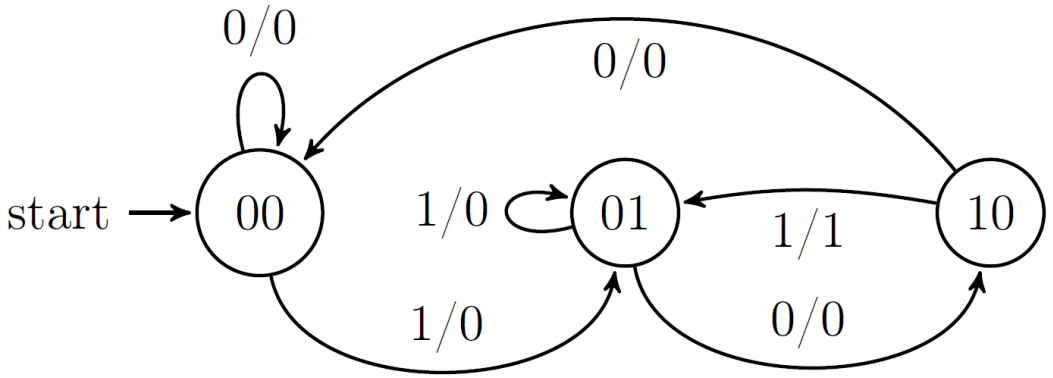
Example



Input ‘0100101010’

Output ‘0000001010’

Example



What does the given FSM implement (Describe when the FSM will output 1)?

When the FSM receives ‘101’, it outputs 1.

Example

Draw a FSM that outputs 1 when it receives two or more successive ‘0’.

Example

Draw a FSM that outputs 1 when it receives two or more successive ‘0’.