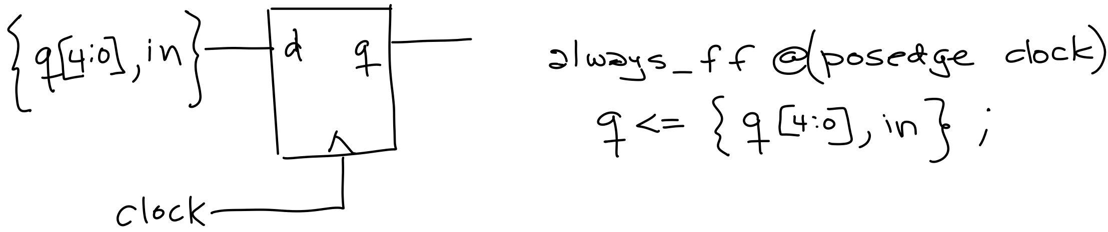
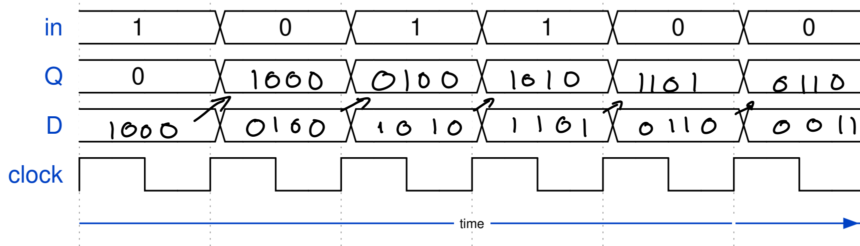
Applications of State Machines

Exercise 1: The example above is an N-bit shift register that shifts the bits right. Draw a block diagram and write the Verilog for a 6-bit shift register that shifts left.



Exercise 2:

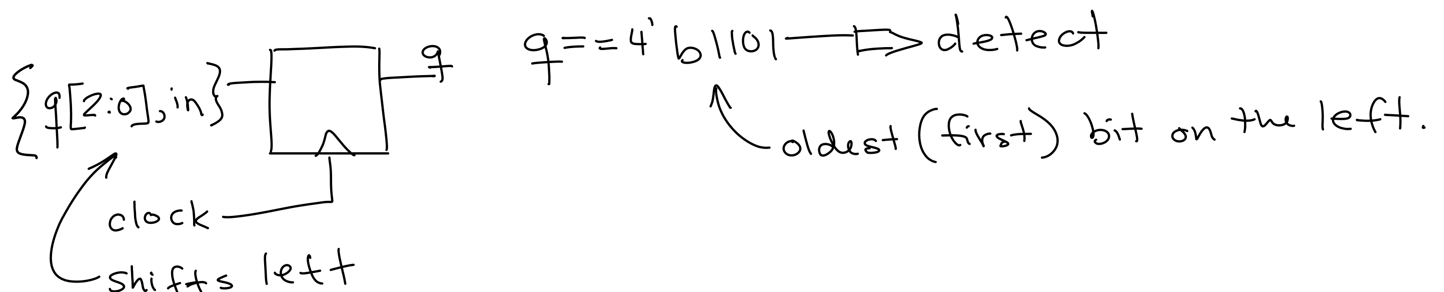


Fill in the diagram above for a 4-bit ($N = 4$) right-shift shift register. Assume the initial value is zero. Which bit is the oldest (first) value in the D waveform? Which bit of the shift register holds the oldest value?

oldest value in waveform is leftmost one.

Oldest bit in shift register is rightmost one.

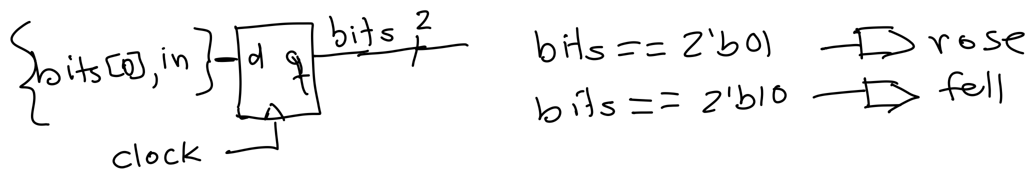
Exercise 3: Draw a block diagram and write the Verilog for a circuit that sets an output named **`detect`** high when the sequence of values 1, 1, 0, 1 has appeared on an input named **`in`** on successive rising edges of the clock.



Exercise 4: For which states would a **`fell`** output be asserted? A **`rose`** output? Draw the schematic and write the Verilog for this state machine. Assume an input **`in`** and a 2-bit register **`bits`** that holds the two most recent input values.

fell asserted when shift register is 2'b10

rose when 2'b01

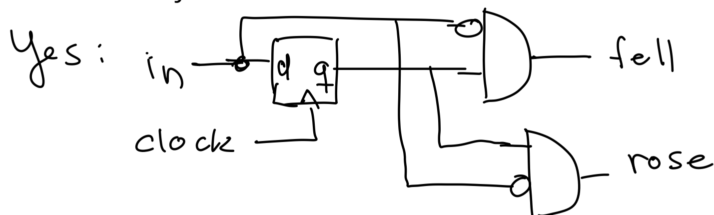



```
always_ff @(posedge clock)
    bits <= {bits[0], in};

assign rose = bits == 2'b01;
assign fell = bits == 2'b10;
```

Exercise 5: Can you design an edge detector that uses only one bit? Is this a Mealy or a Moore state machine?

Yes:




```
alway-ff @(posedge clock)
    q <= in;
assign {fell, rose} = {~in & q, in & ~q};
```

This is a Mealy S.M. → the output is a function of the state (q) and the input (in).

Exercise 6: Write the state transition table for this state machine.

State	reset	next state
XX	1	00
11	0	00
n	0	n+1

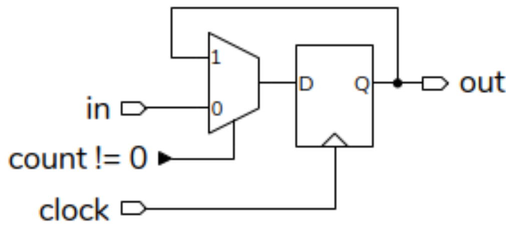
Exercise 7: Write `always_ff` statements that implement these state machines.

e.g. N = 20001

count	in == out	next count
x	1	$N-1$
0	x	$N-1$
n	0	$n-1$

```
logic [15:0] count;

always_ff @(posedge clock)
  count <= in==out ? 16'd20_000 :
           !count ? 16'd20_000 :
           count-1;
```




```
always_ff @(posedge clock)
  out <= count!=0 ? out : in;
```

Exercise 8: How could you modify the code so that **digits** is only updated when an **enable** input is asserted?

```
always_ff @(posedge clock)
  digits <= enable ? {digits[1:3], digit} :
            digits;
```

Exercise 9: How many states can this state machine have?

4 digits x 4 bits = 16.

Exercise 10: Draw the state transition diagram for this simpler implementation. How many states are there? ~~Write the Verilog using a 3-bit **count** state variable.~~