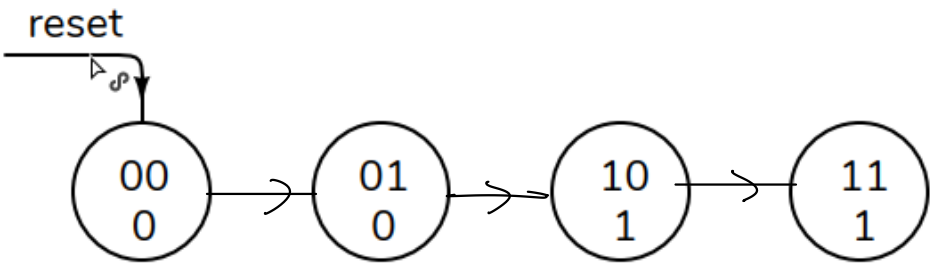
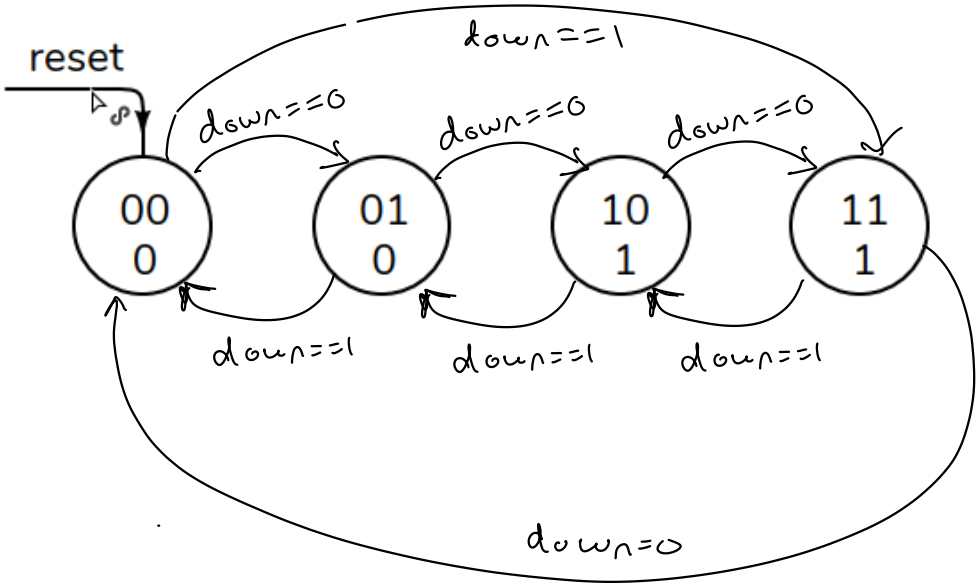
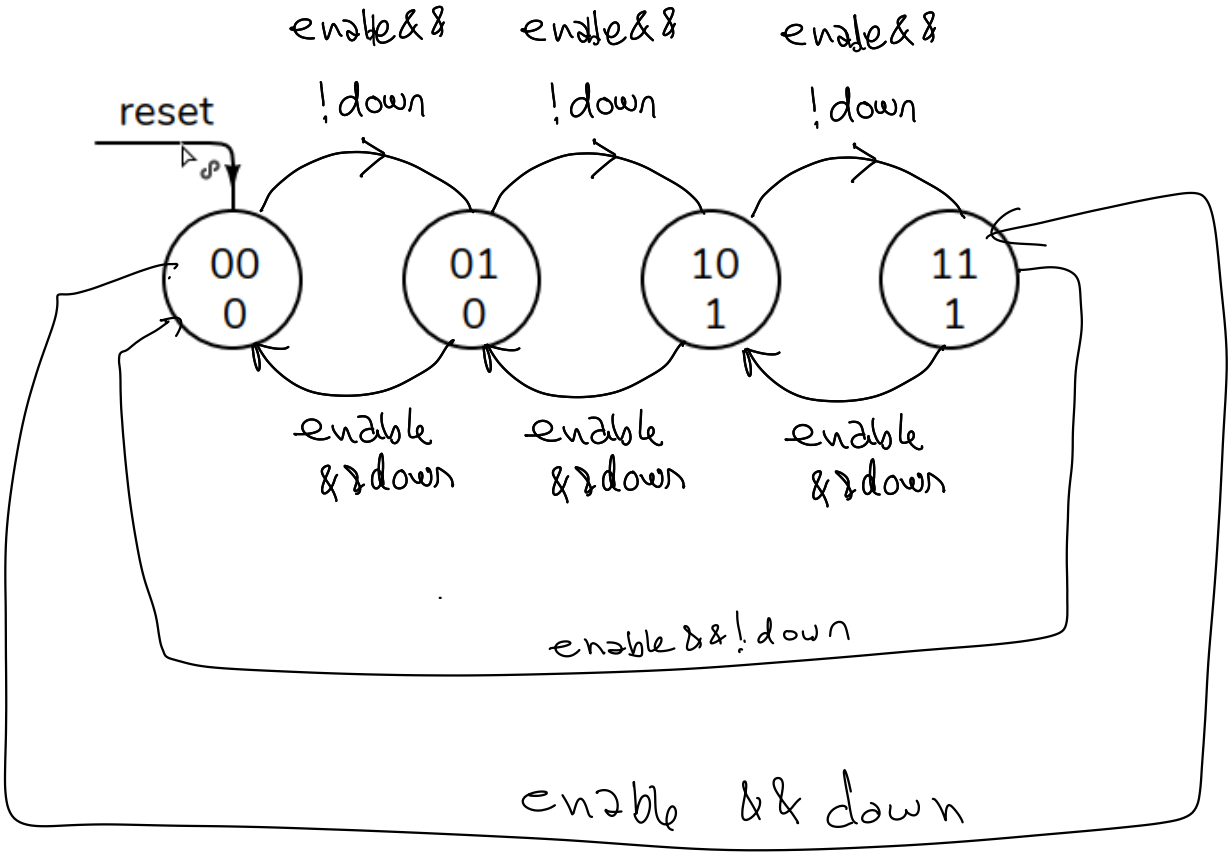
State Machines

Exercise 1: Modify the diagram so the state machine counts to 11 and stops. Add a down input that cause the values to count down.





Exercise 2: Show the state transition diagram and table for a 2-bit counter with reset, enable, and down inputs. Reset should have priority. Write the Verilog.



count	reset	enable	down	count
x x	1	x	x	0 0
n	0	0	x	n
1 1	0	1	0	0 0
0 0	0	1	1	1 1
n	0	1	0	n+1
n	0	1	1	n-1

Exercise 2: Show the state transition diagram and table for a 2-bit counter with reset, enable, and down inputs. Reset should have priority. Write the Verilog.

count	reset	enable	down	count ←
x x	1	x	x	0 0
n	0	0	x	n
1 1	0	1	0	0 0
0 0	0	1	1	1 1
n	0	1	0	n+1
n	0	1	1	n-1

```
always_ff @(posedge clk) count <=
    reset ? 2'b00 :
    !enable ? count :
    count == 2'b11 && !down : 2'b00;
```

Exercise 3: What value of N would result in a 20 ms delay if the clock frequency is 50 MHz? How many bits are needed for this timer's register?

clock frequency = $f = 50 \times 10^6$ Hz

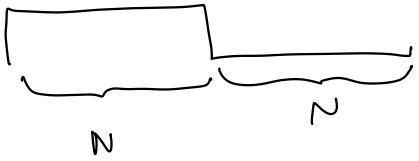
clock period = $T = 20 \times 10^{-9}$ s

delay = $t = NT = 20 \times 10^{-3}$

clock cycles = $N = \frac{t}{T} = \frac{20 \times 10^{-3}}{20 \times 10^{-9}} = 10^6$

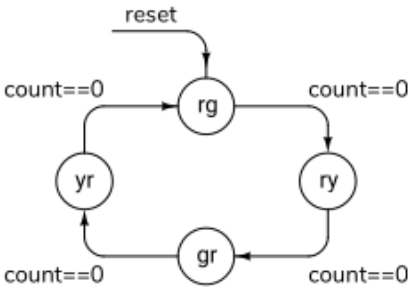
Exercise 4: Assume the timer above is reset to $N - 1$ each time it reaches 0. For how long is the register value 0? What are the period and frequency of a signal that is inverted each time the count reaches 0?

the register is 0 for one clock period.

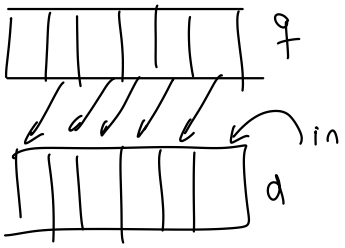


→ period is $2N$ clock periods.

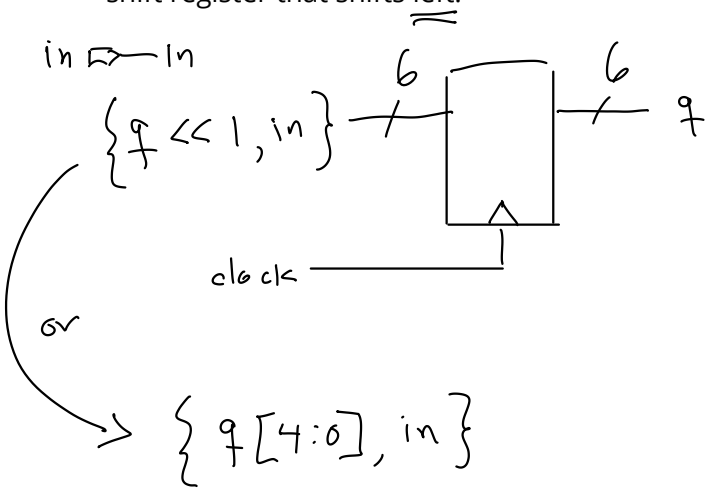
Exercise 5: Write the state transition table for the state machine for the **`lights`** output.



lights	reset	count	next lights
X	1	X	rg
rg	0	0	ry
ry	0	0	gr
gr	0	0	yr
yr	0	0	rg
n	0	≠0	n

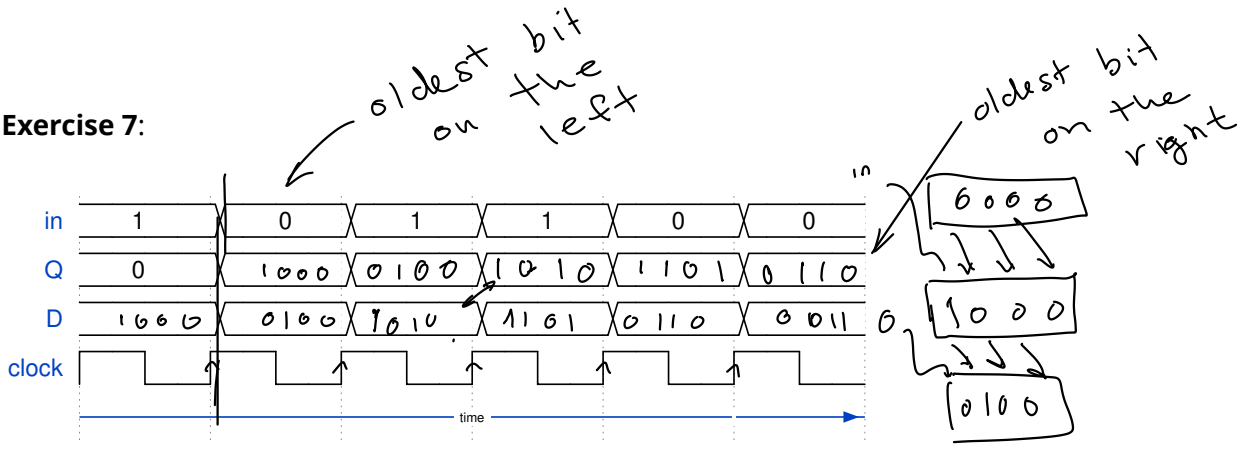


Exercise 6: The example above is an N-bit shift register that shifts the bits right. Draw a block diagram and write the Verilog for a 6-bit shift register that shifts left.

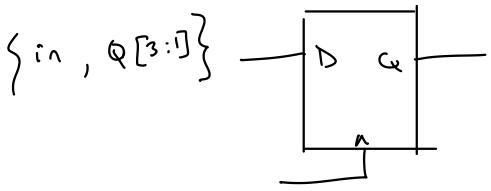



```
always_ff @(posedge clock)
  q <= { q[4:0], in };
```

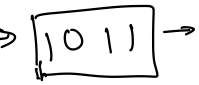
Exercise 7:

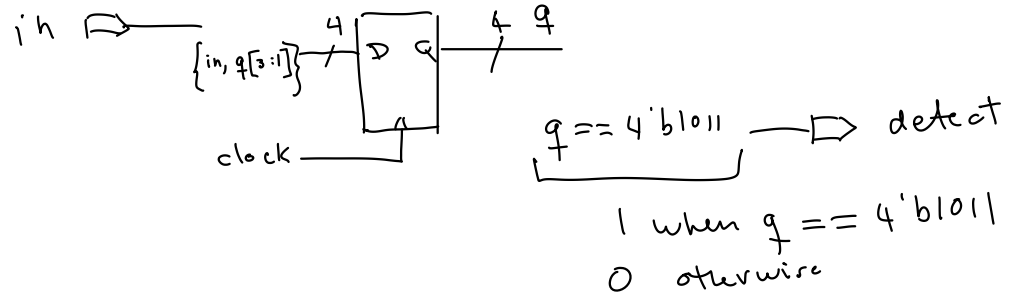


Fill in the diagram above for a 4-bit ($N = 4$) right-shift shift register. Assume the initial value is zero. Which bit is the oldest (first) value in the ~~D~~ waveform? Which bit of the shift register holds the oldest value? in



Exercise 8: 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.



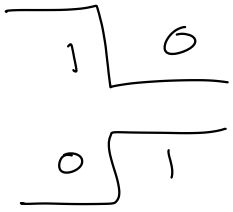


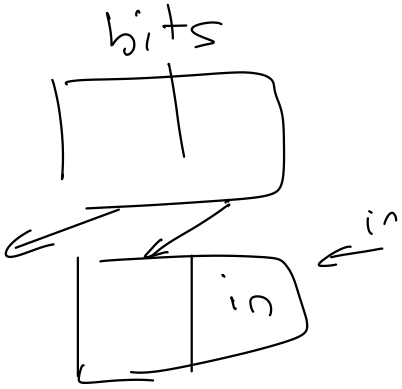

```
always_ff @(posedge clock)
    q <= {in, q[3:1]};

assign detect = q == 4'b1011;
```

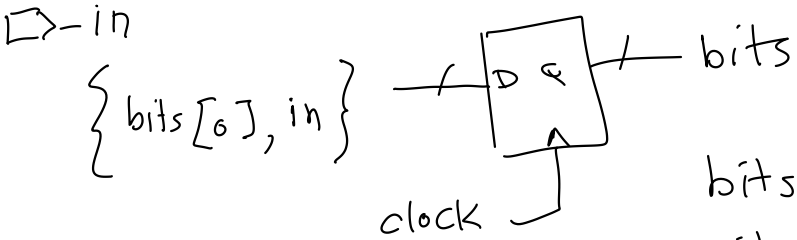
Exercise 9: For which states would **`falling`** be asserted? **`rising`**? Draw the schematic and write the Verilog for this state machine. Assume an input **`in`** and a 2-bit register **`bits`** that holds the two most recent input values.

10 → falling

01 → rising







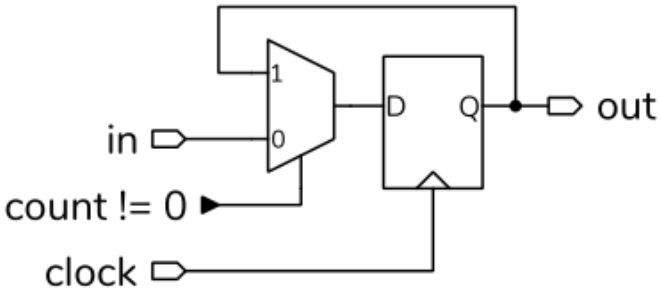
bits == 2'b01 → rising

bits == 2'b10 → falling.

```
module ex9 (
    input logic in, clock,
    output logic rising, falling);

logic [1:0] bits;

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

    assign rising = bits == 2'b01;
    assign falling = bits == 2'b10;

endmodule
```

Exercise 10: Write **`always_ff`** statements that implement these state machines.

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



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

```
always_ff @(posedge clk)
  count <= in == out || count == 0 ? N-1 : count-1;
```