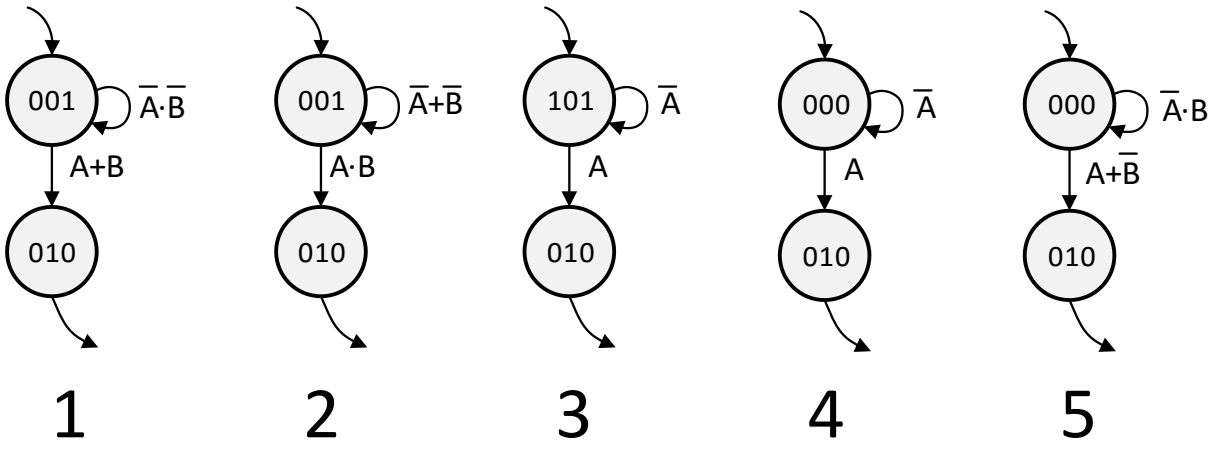
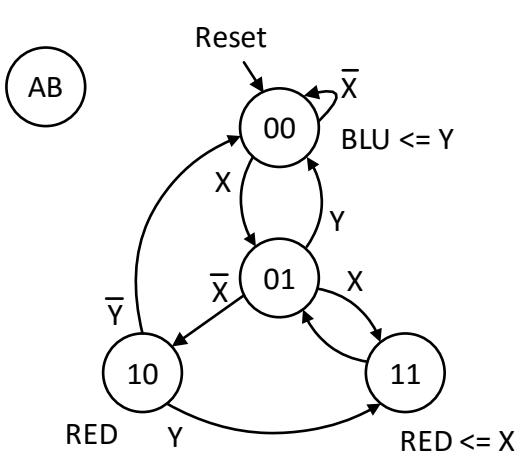
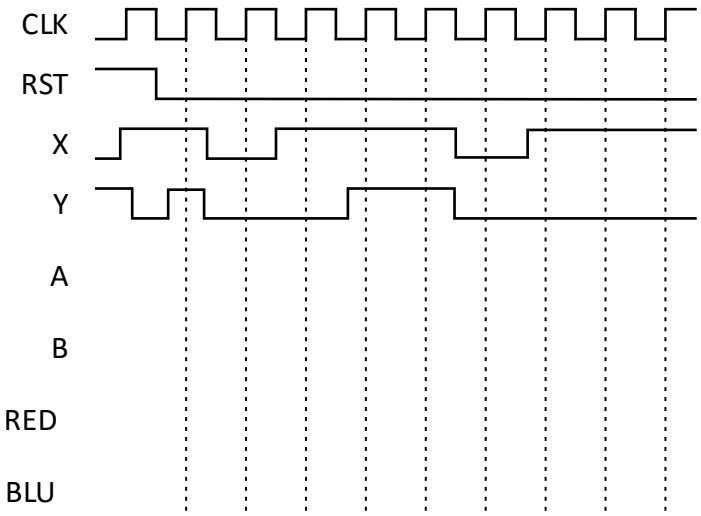
(15 points) Draw a block diagram and state diagram for an input processing circuit that can receive an input X that may be asserted anywhere from 1 to 32 clock cycles, and then asserts an output Y for exactly two clock cycles, regardless of how long X was asserted. Make sure that the output Y cannot glitch (hint: choose state codes well).

(5 points) Circle the number of any state diagram below that could branch to the wrong state if the inputs A and B are not pre-synchronized.



(18 points) In the timing diagram below, show the time courses of the flip-flops (labeled A and B) and output signals defined by the state diagram. Then in the timing diagram, circle the name of any output that can suffer from an ORG, and draw an arrow pointing to any clock edge where the ORG could occur.





(10 points) A particular problem can be solved can using a Moore model binary-coded state machine, or a Johnson-counter that uses one two-input logic gate to drive each output. Circle BC (for binary coded), or JC (for Johnson counter), or both BC and JC to indicate which architecture exhibits the described behavior.

BC JC Can use the fastest system clock available

BC JC Outputs may glitch

BC JC Can generate outputs that turn on and off multiple times and overlap in any sequence

BC JC Can use a 2-input gate to generate glitch-free outputs that turn on and off only once

BC JC Has arbitrarily complex next state logic

BC JC Has no next state logic

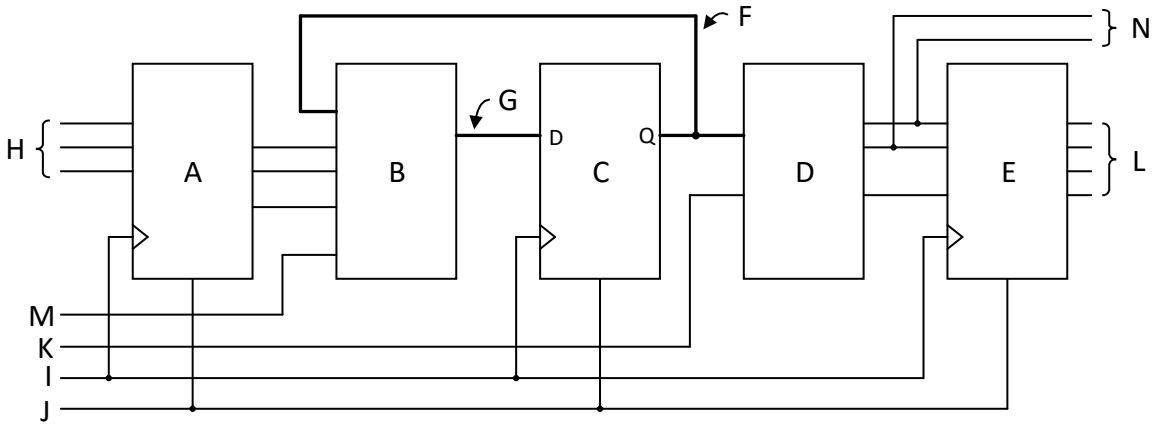
BC JC Uses arbitrarily complex logic to generate output signals

BC JC Can use conditional (“Mealy”) outputs

BC JC Can encode N states with Log_2N flip-flops

BC JC Can encode N states with N/2 flip-flops

(23 points) Place letters in the blanks below to match the circuit blocks with the best description. Some blanks may be left blank (because no letter identifies the block or signal), and some letters may be used more than once.



___ Next state bus	___ State register	___ Output logic
___ Pipelining FIFO	___ Output filtering	___ Smith-trigger filter block
___ Input processing/synchronizing	___ Next-state logic	___ Low pass filter
___ Conditional (“Mealy”) input	___ Current state bus	___ Clock
___ Asynch. input – use “go/no-go”	___ Outputs that may glitch	___ Combinational logic block (2)
___ Glitch-free outputs	___ Reset	___ Could be a counter

___ The number of flip-flops here determine how many states are possible

___ Glitches and hazards in this circuit can generate “logic noise” on outputs

___ This signal needs to be the lowest-skew signal, or else this circuit might not function

___ Delay here is primary determinant of overall clock speed

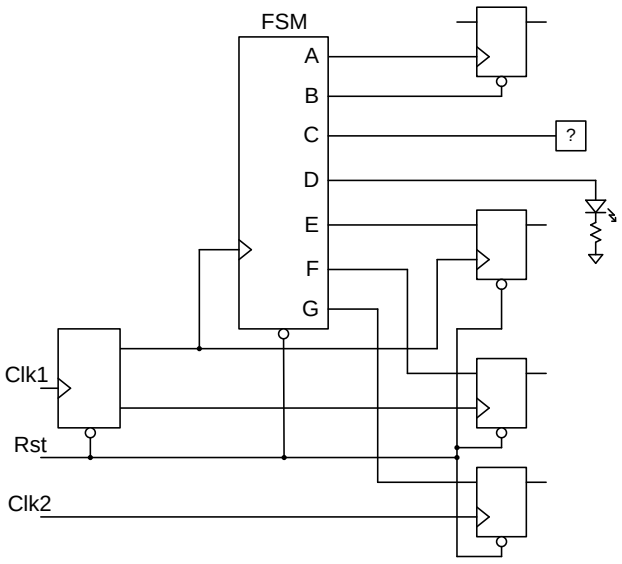
___ Best place for a de-metastabilizing and/or debouncing circuit

___ This circuit can eliminate/filter potential output race glitches and other logic noise

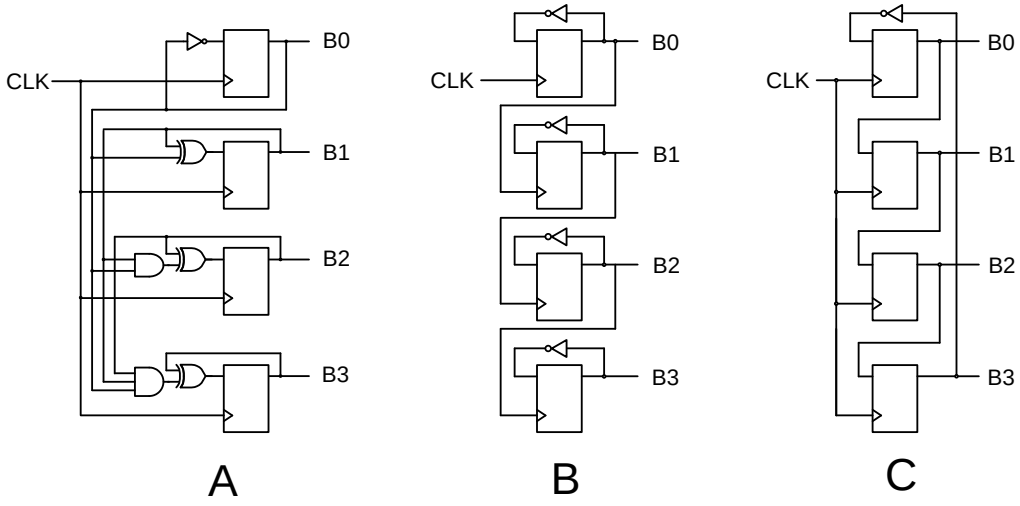
___ The binary number stored here defines the state of the state machine

___ Delay here may skew the outputs, but won’t limit how fast the state machine can run

(7 points) The figure shows a state machine with outputs going to various destinations. Circle the letter of any output that should have logic noise (glitches) removed.



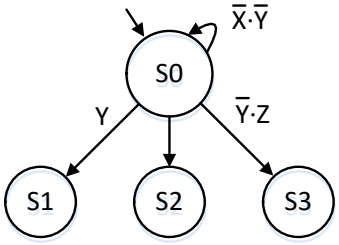
(10 points) The figures below show three different types of 4-bit counters. Circle the letters to indicate which counters (if any) the statements describe - you may circle 0, 1, 2, or all 3 letters for any row.



A	B	C	Creates 2^N binary numbers from N flip-flops
A	B	C	Suffers from output-bit skew problems in higher-order bits
A	B	C	Can be limited in operating frequency by next-state logic delays
A	B	C	Is commonly called a Binary Counter
A	B	C	Is commonly called a Ring or Johnson Counter
A	B	C	Is commonly called an Asynchronous Counter
A	B	C	Creates 2*N binary numbers from N flip-flops
A	B	C	Can suffer from output glitches on the bit signals (B0 – B3)
A	B	C	Bit outputs can be used as clocks for other circuit blocks
A	B	C	Generates numbers in a natural counting sequence (0,1, 2, 3, etc)

(4 points) No branch condition is shown for the S2 branch. If some combination of branch conditions exists for which no next state is specified, then make that combination the S2 branch condition. Then, modify the hold condition so that only one next state is specified for all combinations of branch conditions. Enter the branch conditions below.

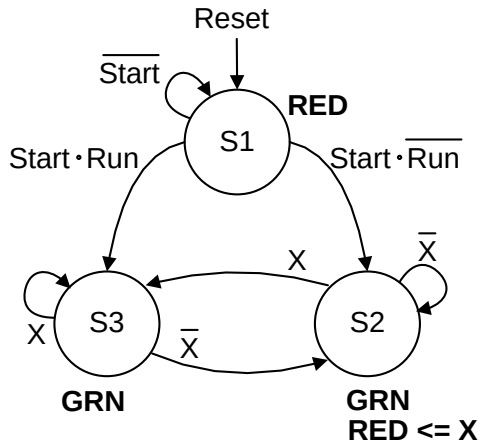
To S2:_____ Holding condition:_____



(10 points) Sketch a block diagram, state diagram, and circuit for the Verilog code shown below. Your circuit should show a D flip-flop together with any other needed logic circuits.

```
module FSM (clk, rst, t, Q);

input clk, rst, t;
output Q;

always @ (posedge(clk))
begin
    if (rst) Q <= 1'b0;
    else if (t) Q <= !Q;
end
endmodule
```

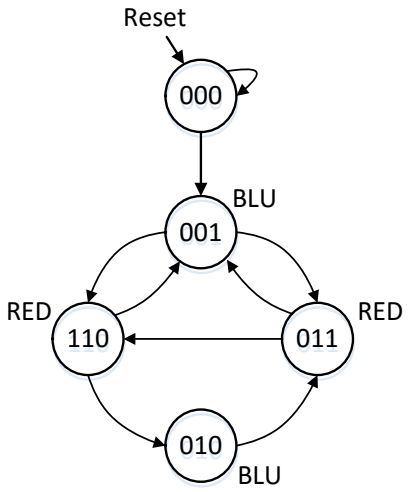
(17 points) Complete the VHDL code listing to define a state machine with the behavior shown in the state diagram (fill in the blanks, and complete all of "state 3").



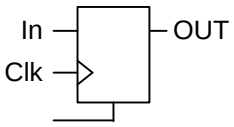

```
library IEEE; use IEEE.std_logic_1164.all;

entity fsm is
  port (_____________________________: in STD_LOGIC;

        _____________________________: out STD_LOGIC);
end fsm;

architecture behavioral of fsm is

  type sreg_type is (_____________________);

  signal ps, ns : sreg_type;
begin

  comb_logic: process (_________________)
  begin
    case ps is

      when S1 =>

        _____________________________ ;
        if Start = '0' then ________ <= S1;
          elsif Start ='1' and Run ='1' then ns <=___________;

          else ____________________________;
        end if;

      when S2 =>

        _____________________________ ;
        if X='0' then ns <= __________;

          else _________________;
        end if;

      when S3 =>









    end case;
  end process;

  state_reg: process (_________________)
  begin
    if reset ='1' then _____________;
      elsif (CLK'event and CLK='1') then _________________;
    end if;
  end process;
end behavioral;
```

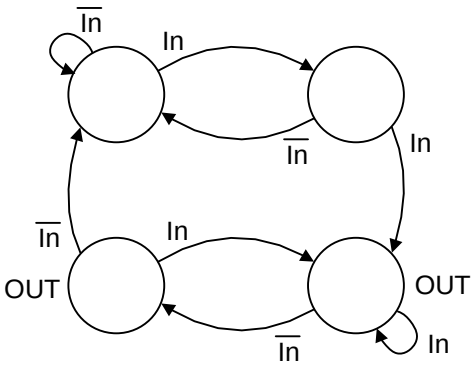
(10 points) Use the table to document the ORGs that can occur in a state machine defined by the state diagram. Note that some outputs may glitch on more than one transition - in this case, use one line in the table for each glitch-prone transition.

Output Name	Type +	Type -	State Transition From	State Transition To



(8 points) The block and state diagram below are proposed for a debouncing circuit. Will it work? If not, what's wrong? If it is to debounce a mechanical switch that can bounce for up to 100uS after a 0-1 or 1-0 transition, what clock frequency range should be used (circle one).





1MHz – 10MHz

100KHz – 1MHz

1KHz – 10KHz

100Hz – 500Hz