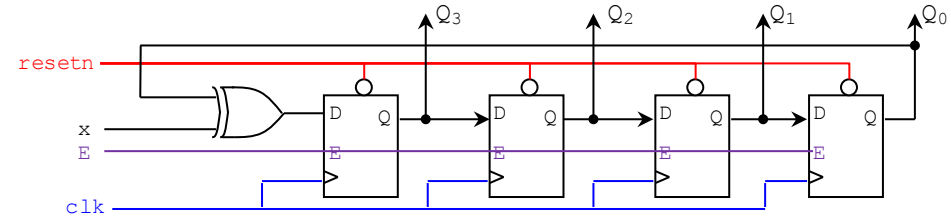
Solutions - Final Exam

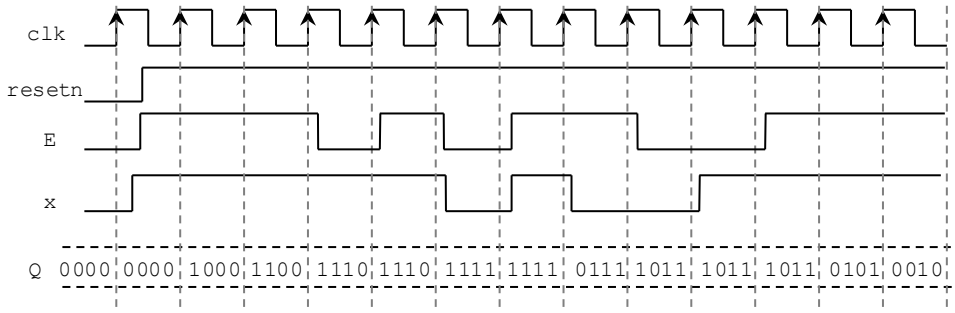
(December 10th @ 7:30 am)

Clarity is very important! Show your procedure!

PROBLEM 1 (12 PTS)

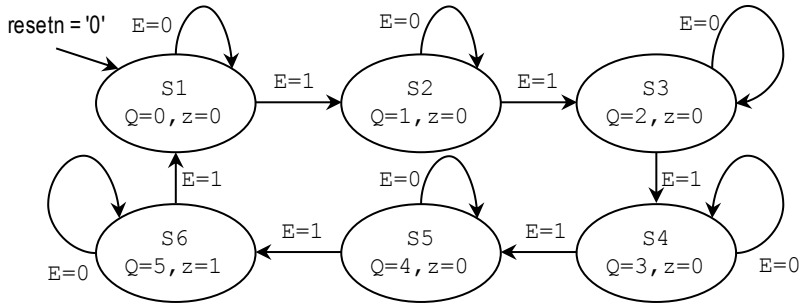
- Complete the timing diagram of the following circuit. $Q = Q_3Q_2Q_1Q_0$





PROBLEM 2 (10 PTS)

- We want to design a counter modulo-6 (count from 0 to 5) with enable using a State Machine. The counter must assert an output `z='1'` when the count reaches 5.
- Provide the State Diagram (any representation) and the Excitation table $\|Inputs|Present\ State\|Next\ State|Outputs\|$. Is this a Moore or a Mealy machine?



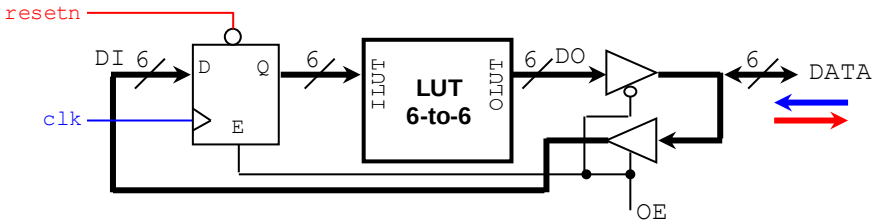
E	PRESENT STATE	NEXT STATE	z
0	S1	S1	0
0	S2	S2	0
0	S3	S3	0
0	S4	S4	0
0	S5	S5	0
0	S6	S6	1
1	S1	S2	0
1	S2	S3	0
1	S3	S4	0
1	S4	S5	0
1	S5	S6	0
1	S6	S1	1

MOORE MACHINE

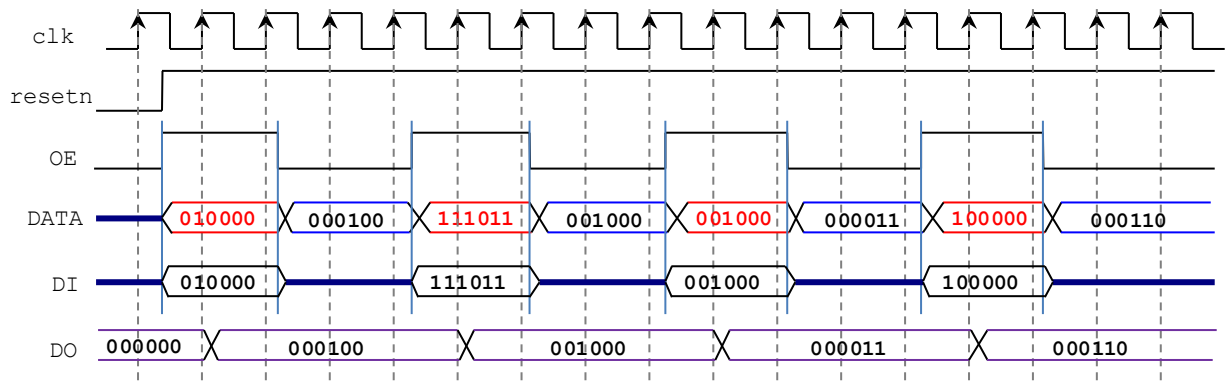
PRESENT STATE E $Q_2Q_1Q_0$ (t)	NEXTSTATE $Q_2Q_1Q_0$ (t+1)	z
0 0 0 0	0 0 0	0
0 0 0 1	0 0 1	0
0 0 1 0	0 1 0	0
0 0 1 1	0 1 1	0
0 1 0 0	1 0 0	0
0 1 0 1	1 0 1	1
0 1 1 0	X X X	X
0 1 1 1	X X X	X
1 0 0 0	0 0 1	0
1 0 0 1	0 1 0	0
1 0 1 0	0 1 1	0
1 0 1 1	1 0 0	0
1 1 0 0	1 0 1	0
1 1 0 1	0 0 0	1
1 1 1 0	X X X	X
1 1 1 1	X X X	X

PROBLEM 3 (15 PTS)

- Given the following system, complete the Timing Diagram.
- The LUT 6-to-6 implements the following function: $OLUT = \lceil sqrt(ILUT) \rceil$



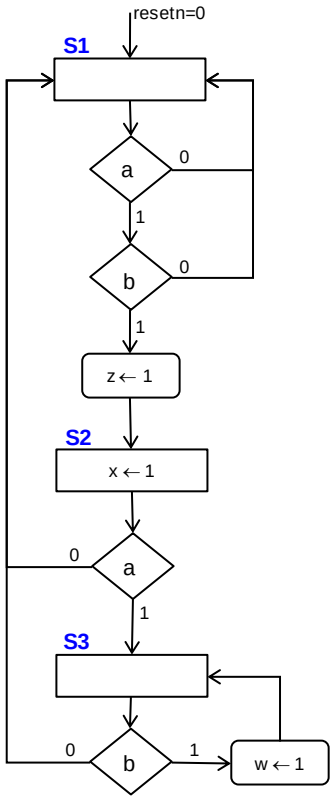

```
ceil(sqrt(16))= 4
ceil(sqrt(59))= 8
ceil(sqrt(8)) = 3
ceil(sqrt(32))= 6
```



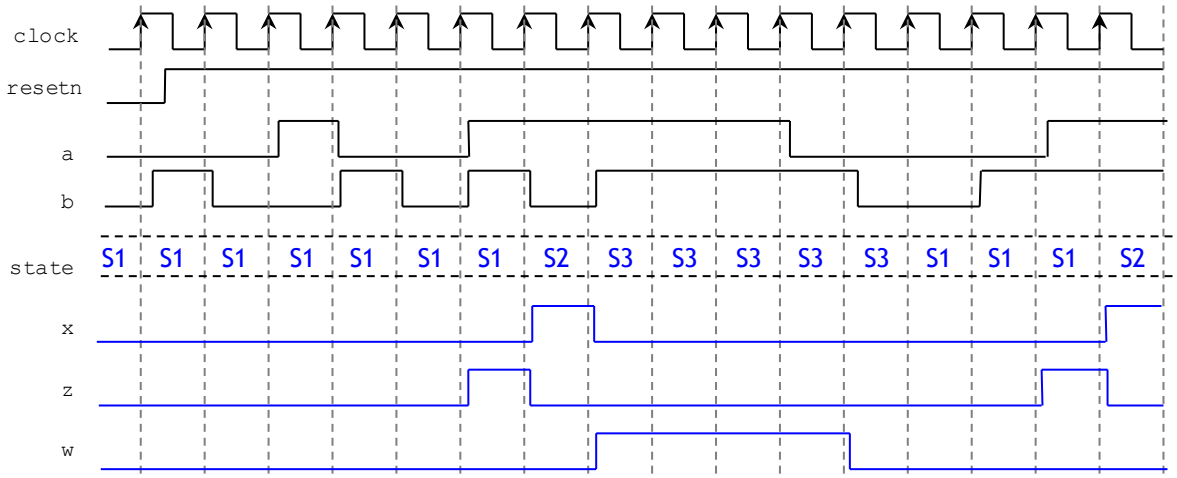
PROBLEM 4 (18 PTS)

- Provide the State Diagram (any representation) of the FSM whose VHDL description is shown below.
- Complete the Timing Diagram.

```
library ieee;
use ieee.std_logic_1164.all;

entity circ is
   port ( clk, resetn: in std_logic;
          a, b: in std_logic;
          x,w,z: out std_logic);
end circ;
```

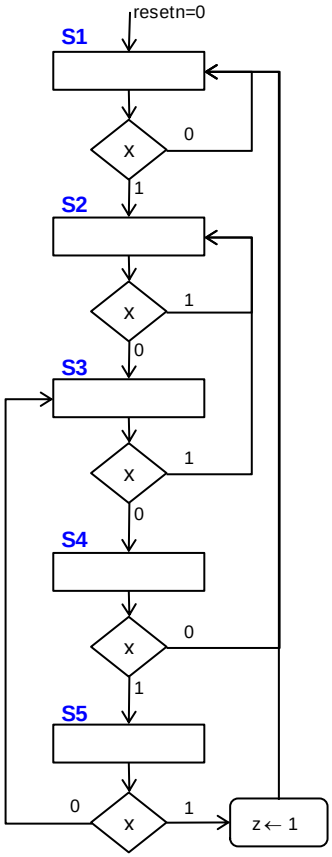



```
architecture behavioral of circ is
   type state is (S1, S2, S3);
   signal y: state;
begin
  Transitions: process (resetn, clk, a, b)
  begin
     if resetn = '0' then y <= S1;
     elsif (clk'event and clk = '1') then
        case y is
           when S1 =>
              if a = '1' then
                if b = '1' then y <= S2;
                else y <= S1; end if;
              else
                 y <= S1;
              end if;

            when S2 =>
               if a = '1' then y <= S3;
               else y <= S1; end if;

            when S3 =>
               if b = '1' then y <= S3;
               else y <= S1; end if;
        end case;
     end if;
  end process;

  Outputs: process (y,a,b)
  begin
      x <= '0'; w <= '0'; z <= '0';
      case y is
         when S1 =>
            if a = '1' and b = '1' then
            z <= '1'; end if;
         when S2 => x <= '1';
         when S3 =>
            if b = '1' then w <= '1'; end if;
      end case;
  end process;
end behavioral;
```



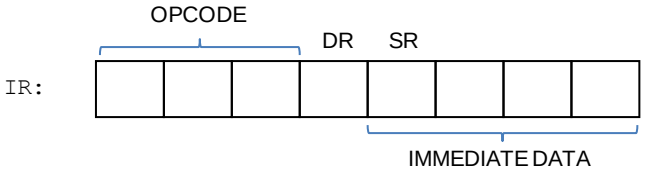
PROBLEM 5 (10 PTS)

- Sequence detector (with overlap): Draw the state diagram (in ASM form) of a circuit (with an input 'x') that detects the following sequence: `10011`. The detector must assert and output `z='1'` when the sequence is detected.



PROBLEM 6 (15 PTS)

- **Basic Processor**:
  Available Registers: `R0` (register 0, 4 bits), `R1`(register 1, 4 bits), `PC` (program counter, 4 bits),
  `OUT` (output register, 4 bits)
  `IR` (instruction register, 8 bits)
  Instruction Memory: Stores up to 16 8-bit instructions.
  Instruction Set: Instructions are specified on the Instruction Register (`IR`):



`DR=0` ⇒ `R0` is the destination register, `DR=1` ⇒ `R1` is the destination register.
`SR=0` ⇒ `R0` is the source register, `SR=1` ⇒ `R1` is the source register.

OPCODE (`IR[7..5]`)	Instruction	Operation
000	MOV DR, SR	DR ← SR
001	LOADI DR, DATA	DR ← DATA, DATA = IR[3..0]
010	ADD DR, SR	DR ← DR + SR
011	ADDI DR, DATA	DR ← DR + DATA, DATA = IR[3..0]
100	SR0 DR, SR	DR ← 0&SR[3..1]
101	IN DR	DR ← IN
110	OUT DR	OUT ← DR
111	JNZ DR, ADDRESS	PC ← PC + 1 if DR=0 PC ← IR[3..0] if DR≠0 * ADDRESS = IR[3..0]

- Write an assembly program for a counter from 2 to 13: `2,3,…, 13,2,3,…` The count must be shown on the output register (OUT). Use labels to specify any address where your program jumps. Note that you can have only up to 16 instructions.
- Provide the contents of the Instruction Memory.

address	**INSTRUCTION MEMORY**
0000	00100010
0001	00110100
0010	11000000
0011	01100001
0100	01110001
0101	11110000
0110	00100001
0111	11100010
1000	
1001	
1010	
1011	
1100	
1101	
1110	
1111	

```
start: loadi R0,2
       loadi R1,4
loop:  out R0 → OUT: shows the count
       addi R0,1
       addi R1,1
       jnz R1, loop
       loadi R0,1
       jnz R0, start
```

PROBLEM 7 (20 PT)

- Complete the timing diagram of the following digital circuit that includes an FSM (in ASM form) and a datapath circuit.