Class Midterm
Spring 2016

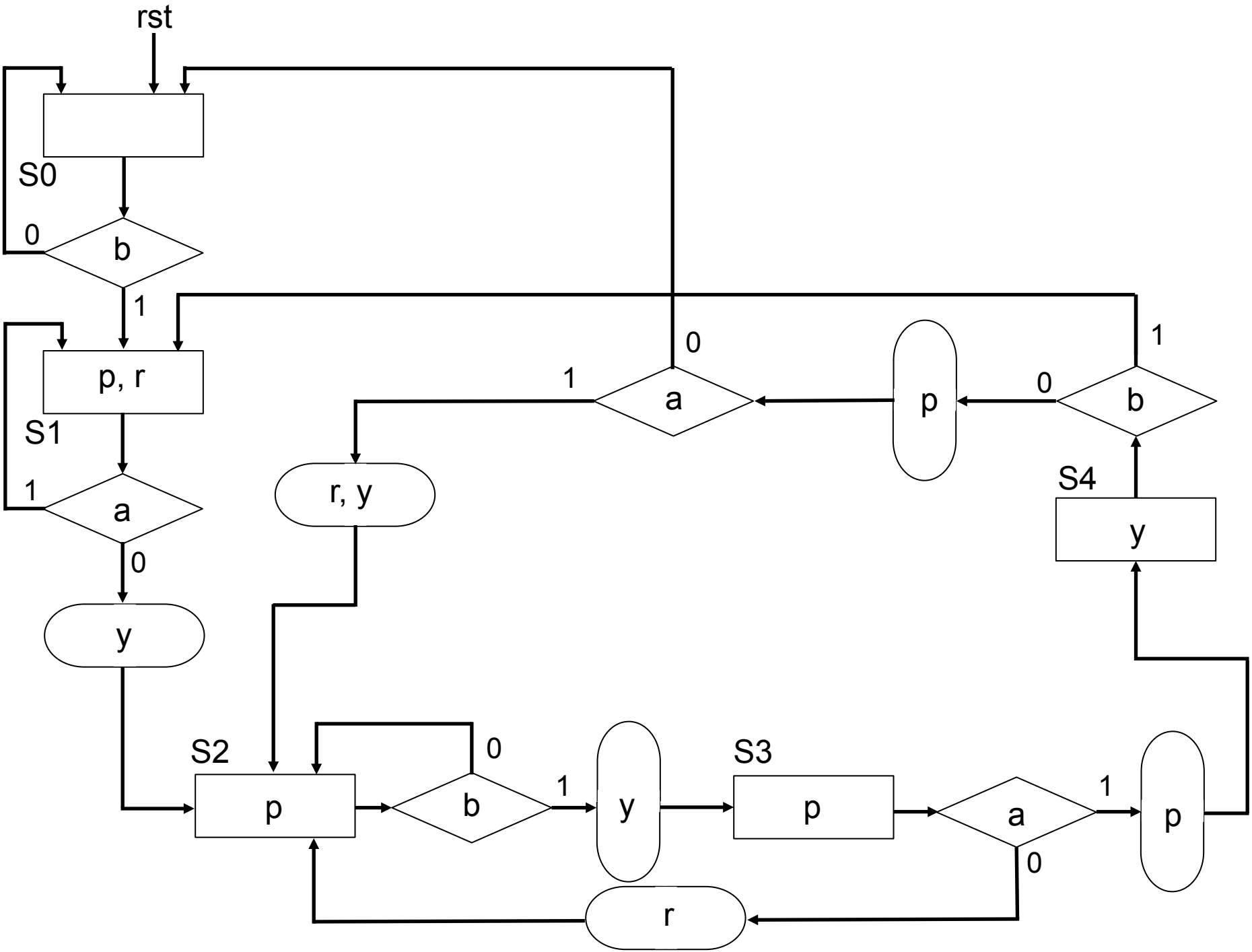
Solutions

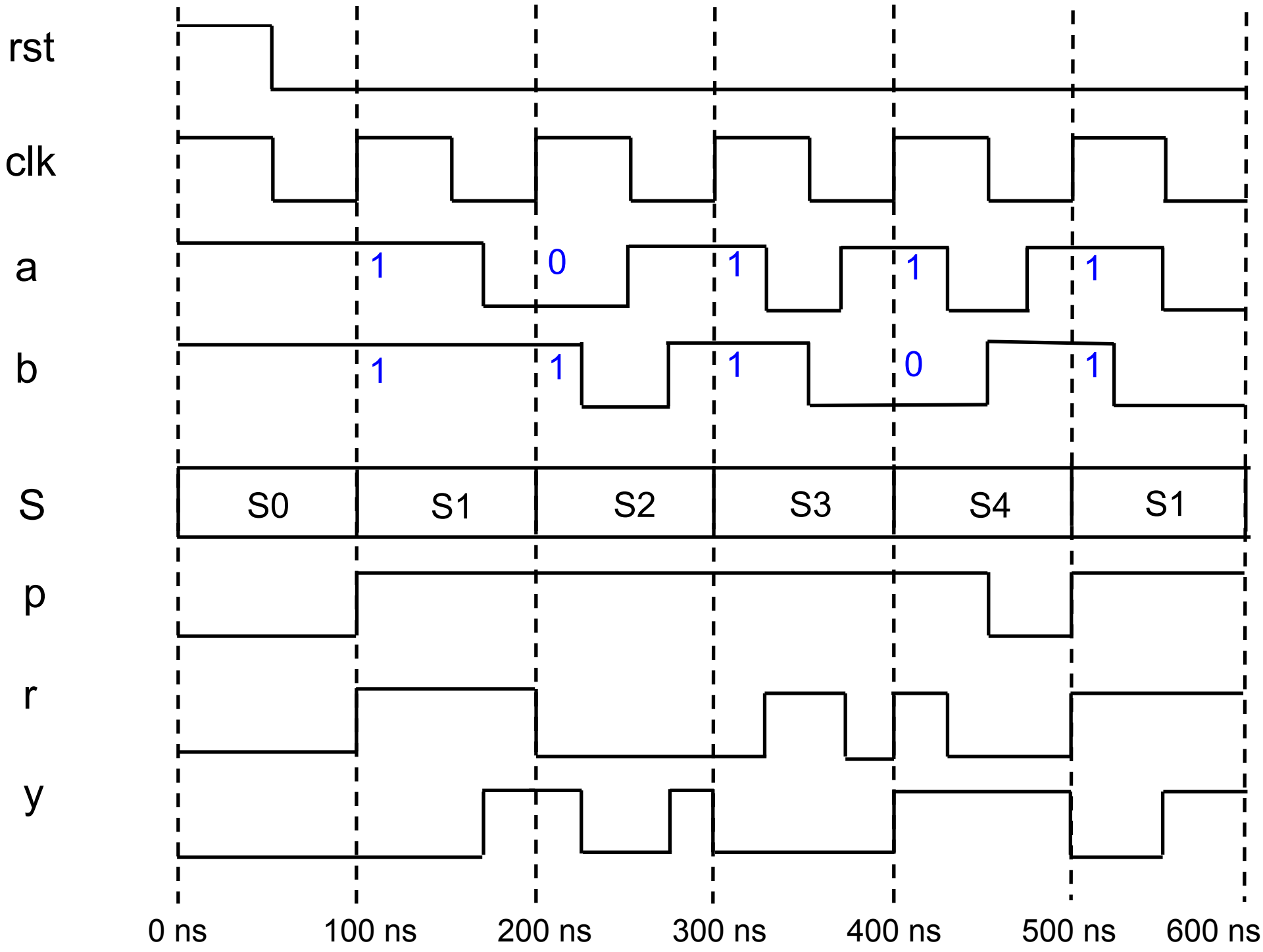
Problem 1

Given the controller, described using the ASM chart shown in Fig. 1:

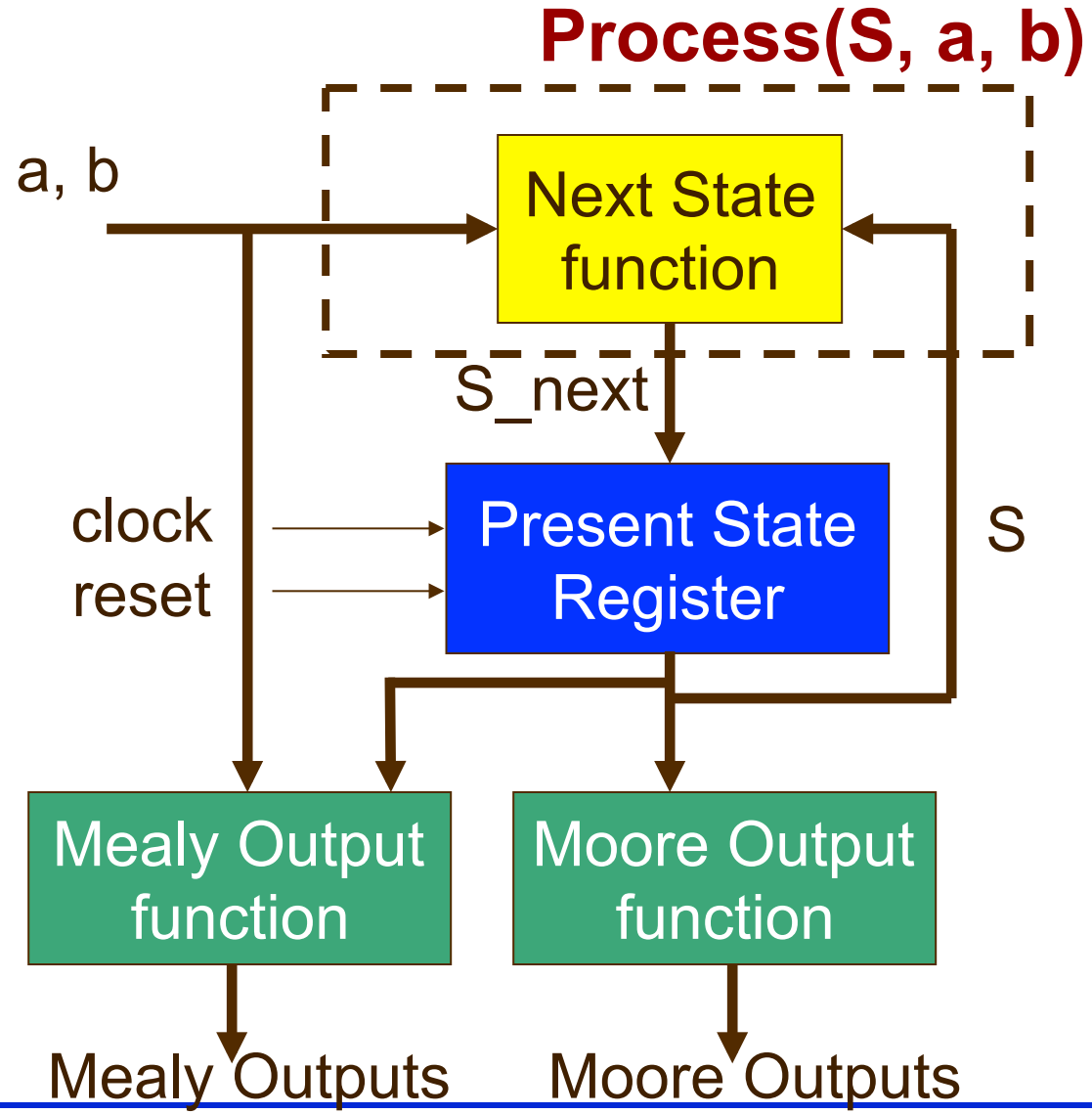
A. supplement timing waveforms provided in the answer sheet with the values of the state S, and the values of the outputs p, r, and y

B. write the VHDL behavioral code for the next state function (only), calculating value of the next state S.

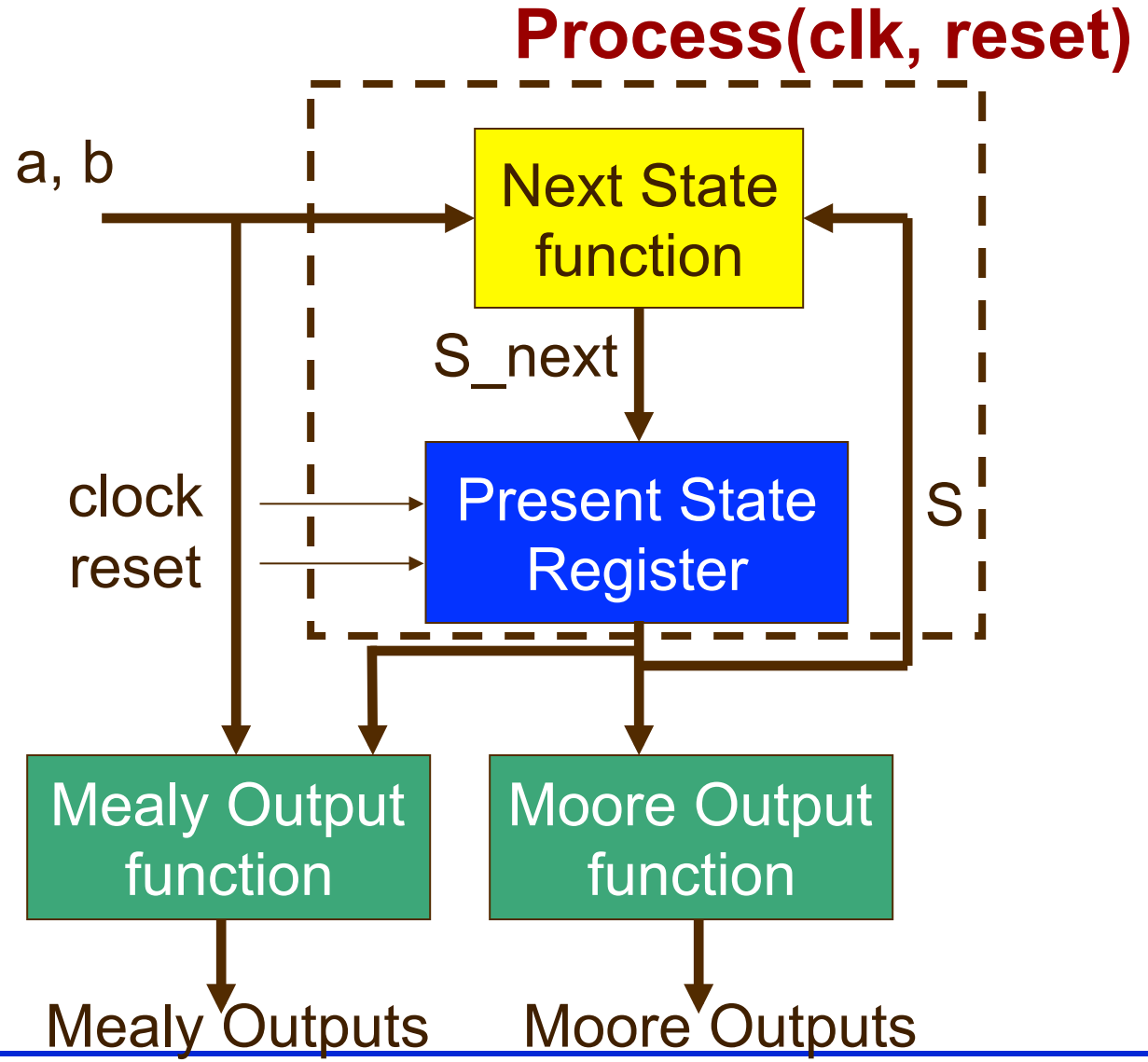

rst
S0
b
0
1
p, r
S1
a
1
0
y
r, y
S2
p
b
0
1
y
S3
p
a
1
0
r
p
0
b
1
a
1
0
p
S4
y
1
p

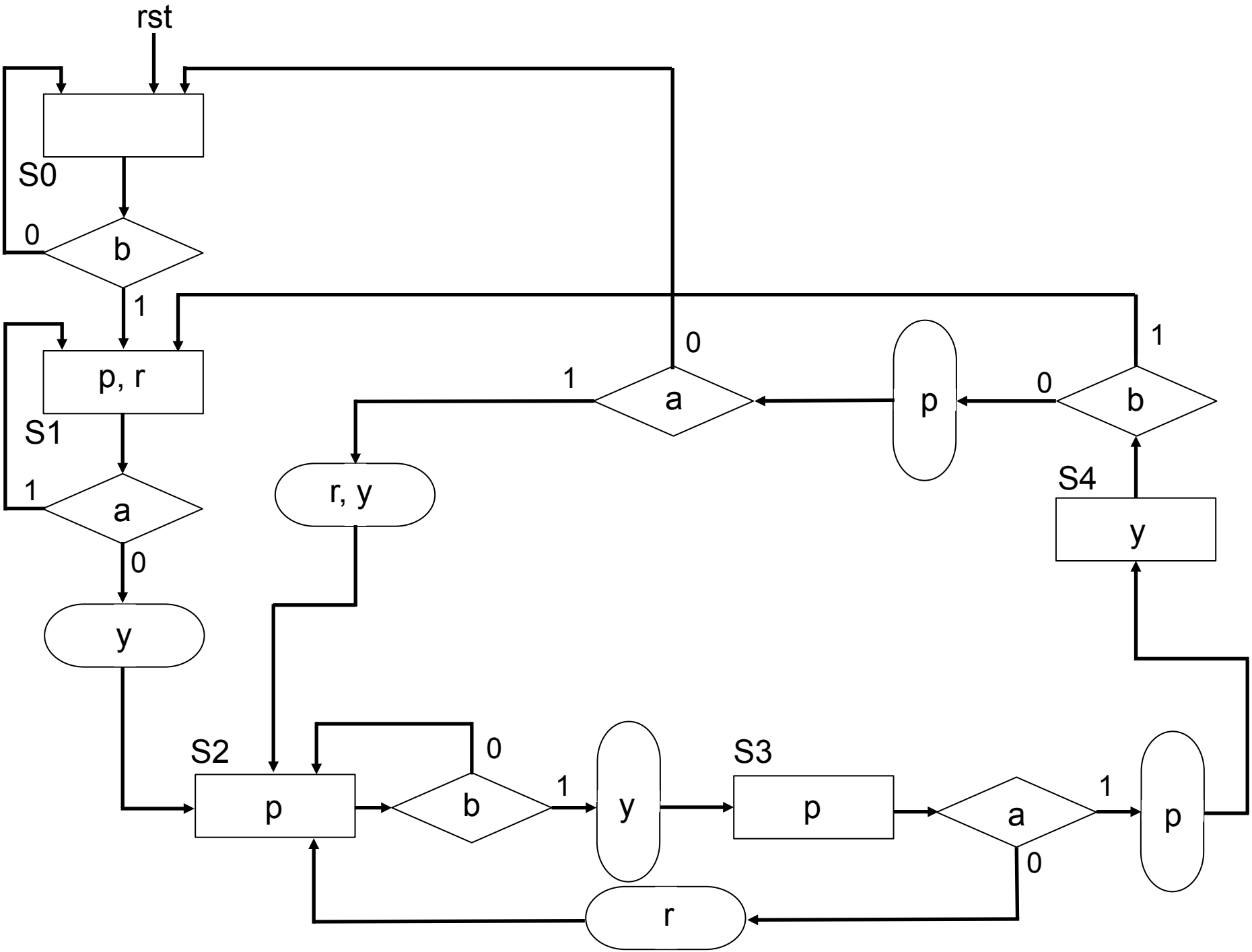

rst
clk
a
b
S
p
r
y
1
0
1
1
1
1
1
1
0
1
S0
S1
S2
S3
S4
S1
0 ns
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns

FSM


Process(S, a, b)
a, b
Next State function
S_next
clock
reset
Present State Register
S
Mealy Output function
Moore Output function
Mealy Outputs
Moore Outputs

FSM


Process(clk, reset)
a, b
Next State function
S_next
clock
reset
Present State Register
S
Mealy Output function
Moore Output function
Mealy Outputs
Moore Outputs


rst
S0
b
0
1
p, r
S1
a
1
0
y
r, y
S2
p
b
0
1
y
S3
p
a
1
0
r
p
0
b
1
a
1
0
p
S4
y
1
p

Problem 1 – VHDL code (1)

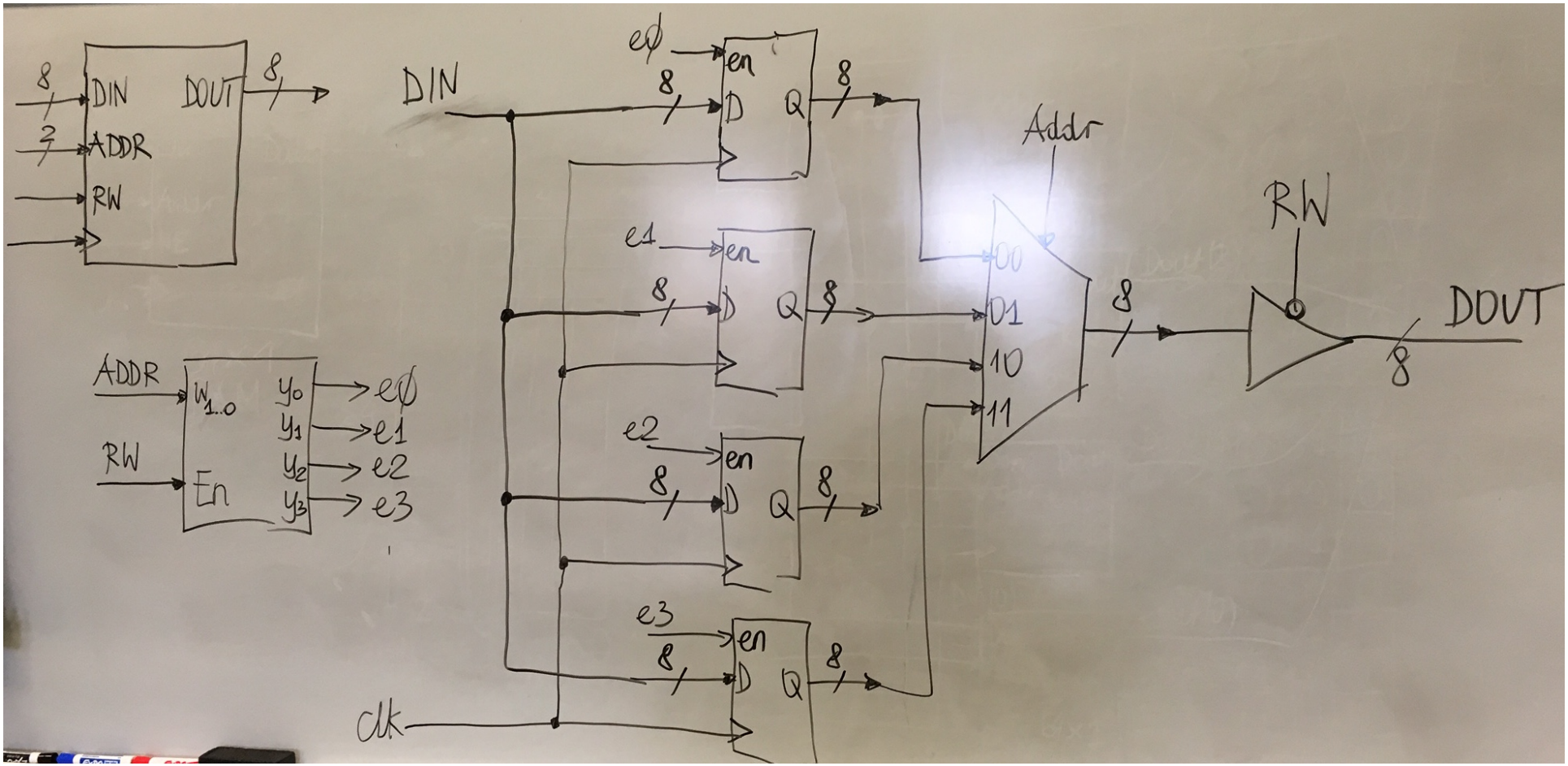
```
PROCESS ( S, a, b )
BEGIN
   S_next <= S;
   CASE S IS
         WHEN S0 =>
               IF b = '1' THEN
                     S_next <= S1;
               END IF ;
         WHEN S1 =>
               IF a = '0' THEN
                     S_next <= S2;
               END IF;
         WHEN S2 =>
               IF  b = '1'  THEN
                     S_next <= S3;
               END IF;
```

Problem 1 – VHDL code (2)

```
        WHEN S3 =>
            IF  a = '1'  THEN
                S_next <= S4;
            ELSE
                S_next <= S2;
            END IF;
        WHEN S4 =>
            IF  b = '1' THEN
                S_next <= S1;
            ELSIF a = '1'  THEN
                S_next <= S2;
            ELSE
                S_next <= S0;
            END IF;

END PROCESS ;
```

Problem 3 – Block diagram



Problem 3 – VHDL code (1)

```
ARCHITECTURE mixed OF problem_3 IS

SIGNAL Q0, Q1, Q2, Q3, mux_out : STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL en : STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL enw : STD_LOGIC_VECTOR(2 DOWNTO 0);

BEGIN

enw <= RW & ADDR ;
WITH enw SELECT
        en <=   "0001" WHEN "100",
                "0010" WHEN "101",
                "0100" WHEN "110",
                "1000" WHEN "111",
                "0000" WHEN OTHERS ;

WITH ADDR SELECT
        mux_out <=  Q0 WHEN "00",
                    Q1 WHEN "01",
                    Q2 WHEN "10",
                    Q3 WHEN OTHERS;
```

Problem 3 – VHDL code (2)

```
DOUT <= mux_out WHEN RW='0' ELSE (OTHERS => ‘Z’);

PROCESS(CLK)
        IF rising_edge(CLK) THEN
                IF en(0) THEN
                        Q0 <= DIN;
                END IF;
                IF en(1) THEN
                        Q1 <= DIN;
                END IF;
                IF en(2) THEN
                        Q2 <= DIN;
                END IF;
                IF en(3) THEN
                        Q3 <= DIN;
                END IF;
        END IF;
END PROCESS;

END mixed;
```

Problem 3 – VHDL code with for-generate (1)

```
ARCHITECTURE mixed OF problem_3 IS

TYPE ARRAY8 IS ARRAY 0 TO 3 OF STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL Q: ARRAY8;
SIGNAL mux_out : STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL en : STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL enw : STD_LOGIC_VECTOR(2 DOWNTO 0);

BEGIN

enw <= RW & ADDR ;
WITH enw SELECT
        en <=   "0001" WHEN "100",
                "0010" WHEN "101",
                "0100" WHEN "110",
                "1000" WHEN "111",
                "0000" WHEN OTHERS ;
```

Problem 3 – VHDL code with for-generate (2)

```
WITH ADDR SELECT
        mux_out <=  Q(0) WHEN "00",
                    Q(1) WHEN "01",
                    Q(2) WHEN "10",
                    Q(3) WHEN OTHERS;

DOUT <= mux_out WHEN RW='0' ELSE (OTHERS => 'Z');

FOR i IN 0 TO 3 GENERATE
      PROCESS(CLK)
            IF rising_edge(CLK) THEN
                  IF en(i) THEN
                        Q(I) <= DIN;
                  END IF;
            END IF;
      END PROCESS;
END GENERATE;

END mixed;
```

Problem 4 (1)

Write a complete simple testbench capable of verifying the operation of MISR (Multiple Input Signature Register) shown in Fig. 2.

The testbench should instantiate MISR, using the VHDL-93 convention, with the value of the generic C equal to X”B8”. Assume that the name of the MISR entity is MISR, and the name of its architecture is behavioral.

Problem 4 (2)

The test should consist of

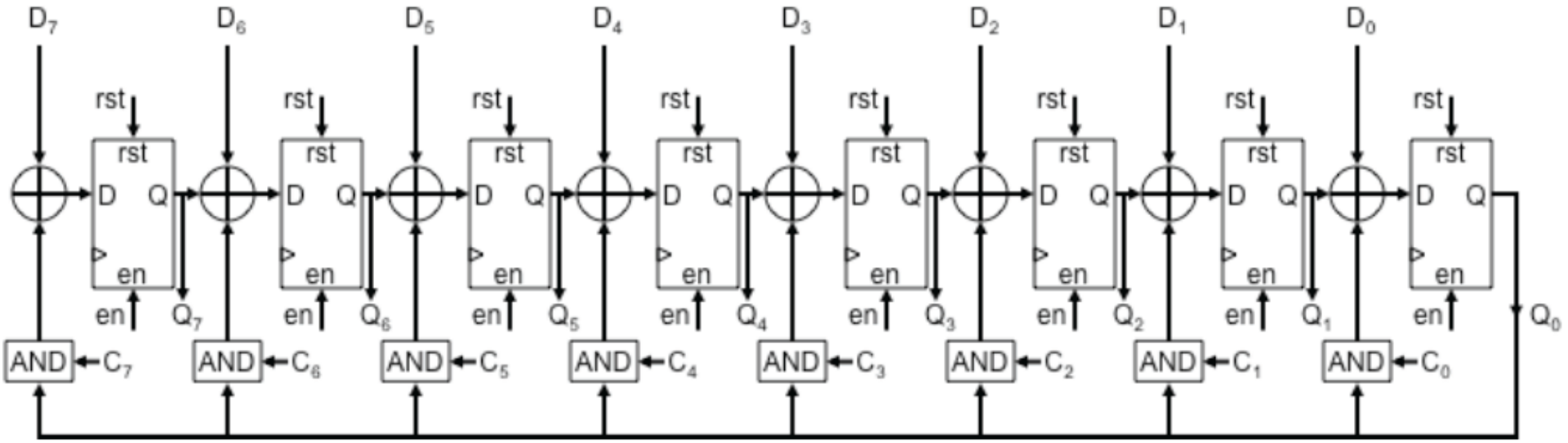
1. resetting MISR
2. applying 128 inputs D, varying between X”00” and X”FE” with the step of 2 (i.e., values X”00”, X”02”, X”04”, …, X”FE”), one per clock cycle, with the input en active
3. deactivating en for 10 clock cycles
4. applying 128 inputs D, varying between X”01” and X”FF” with the step of 2 (i.e., values X”01”, X”03”, X”05”, …, X”FF”), one per clock cycle, with the input en active.
5. deactivating en.

Problem 4 (3)

Assume that all external inputs should be changed on the falling edges of the clock.

After step 5, all inputs (other than clk) should remain stable (i.e., keep their last value) till the end of simulation.

Block Diagram



Problem 4 – Testbench (1)

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

USE ieee.std_logic_unsigned.all;

ENTITY MISR_tb IS
END MISR_tb;

ARCHITECTURE behavioral OF MISR_tb IS

SIGNAL clk : STD_LOGIC := '0';
SIGNAL rst : STD_LOGIC := '1';
SIGNAL en : STD_LOGIC := '0';
SIGNAL D, Q : STD_LOGIC_VECTOR(7 downto 0);

CONSTANT clk_period := 10ns;
CONSTANT rst_width :=  clk_period;
```

Problem 4 – Testbench (2)

```
BEGIN

    DUT: ENTITY work.MISR(behavioral)
        GENERIC MAP (C => XB8")
        PORT MAP (
            clk => clk,
            rst => rst,
            en => en,
            D => D,
            Q => Q);
        );


    clk <= not clk after clk2_period/2;
```

Problem 4 - Testbench (3)

```
main: PROCESS
        D <= X"00";
        WAIT for rst_width;
        rst <= '0';
        en <= '1';
        FOR i IN 0 TO 127 LOOP
                WAIT for clk_period;
                D <= D + 2;
        END LOOP;
        en <= '0';
        WAIT FOR 10*clk_period;
        en <= '1';
        D <= X"01";
        FOR i IN 0 TO 127 LOOP
                WAIT for clk_period;
                D <= D + 2;
        END LOOP;
        en <= '0';
        WAIT;
END PROCESS;
END behavioral;
```