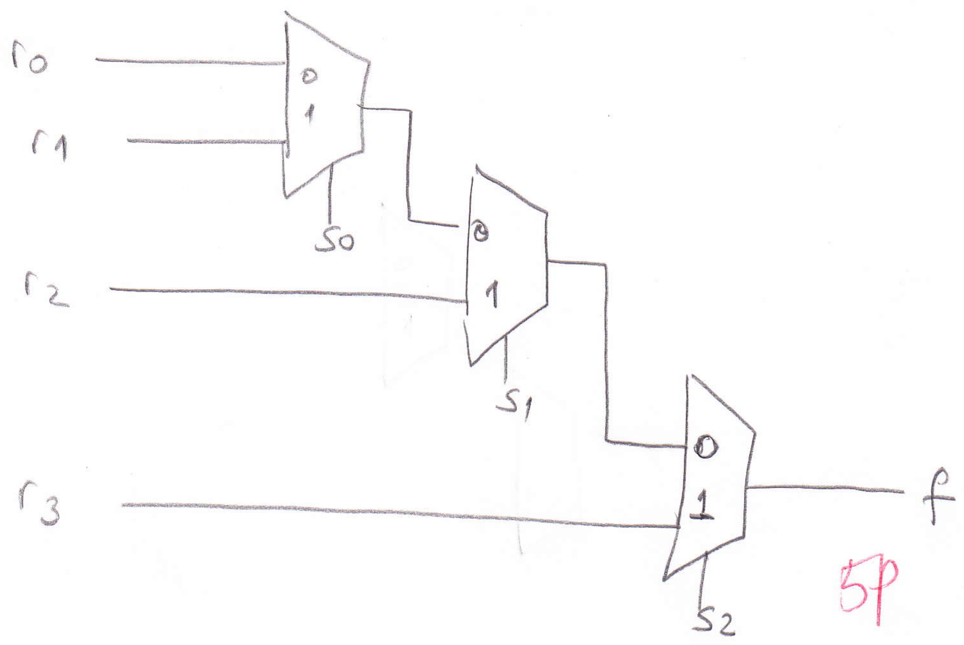
① Design a priority encoder of four inputs (r_3, r_2, r_1, r_0) by using 2-to-1 multiplexers. r_3 having the highest priority and r_0 the lowest. Write the behavioural VHDL code of the circuit. (10p)

SOLUTION



5p

r_3	r_2	r_1	r_0	f
1	X	X	X	r_3
0	1	X	X	r_2
0	0	1	X	r_1
0	0	0	1	r_0

5p

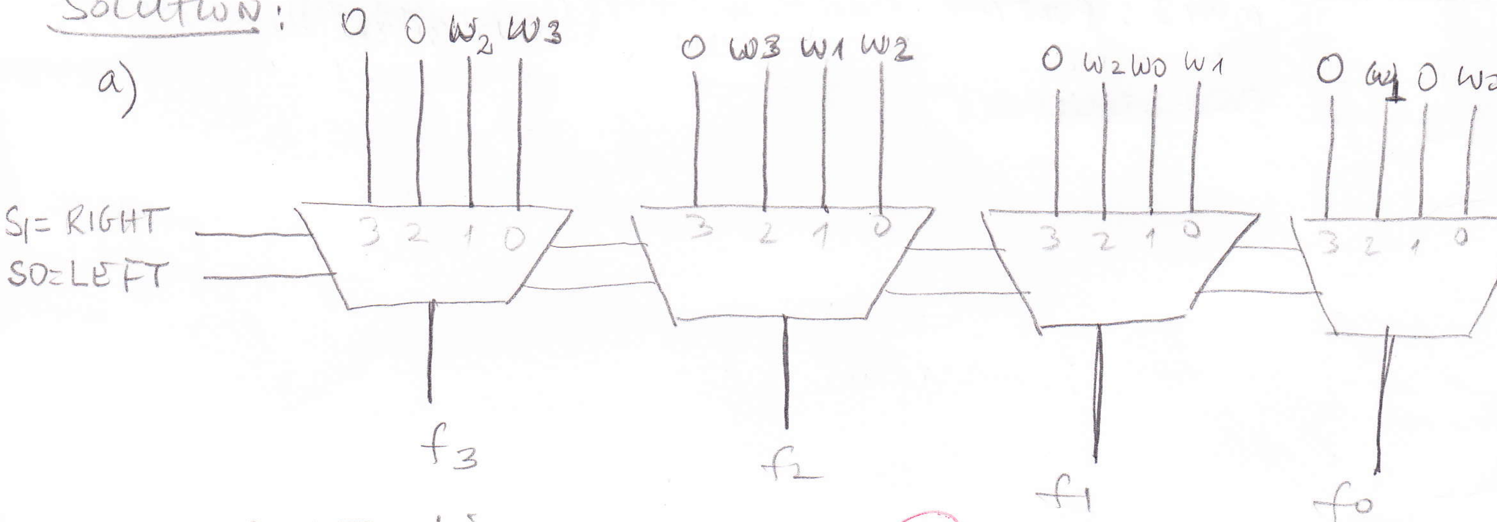
```
ARCHITECTURE Behavior OF Priority IS
SIGNAL: r : STD-LOGIC-VECTOR (0 TO 3);
SIGNAL: S : STD-LOGIC-VECTOR (0 TO 2);
PROCESS (r, S)
IF S(2)='1' THEN f <= R(3);
ELSIF S(1)='1' THEN f <= R(2);
ELSIF S(0)='1' THEN f <= R(1);
ELSE f <= R(0);
ENDIF;
END PROCESS;
```

2 Design a shifter circuit which can shift a four-bit input vector, W = $w_3 w_2 w_1 w_0$, one bit position to the right when the control signal Right = 1, and one bit position to the left when Left=1. When Right = Left = 0, the output of the circuit should be the same as input vector W. When Right = Left = 1 the output shall be resetted (i.e., all zeroes). A zero should come for the rightmost and leftmost bit for shift left and shift right respectively.

a) Design the circuit
b) Write the VHDL code. (10p)

SOLUTION:

a)



S1 S0	Function
0 0	f=W
0 1	Shift LEFT
1 0	Shift RIGHT
1 1	Reset (all zero)

5

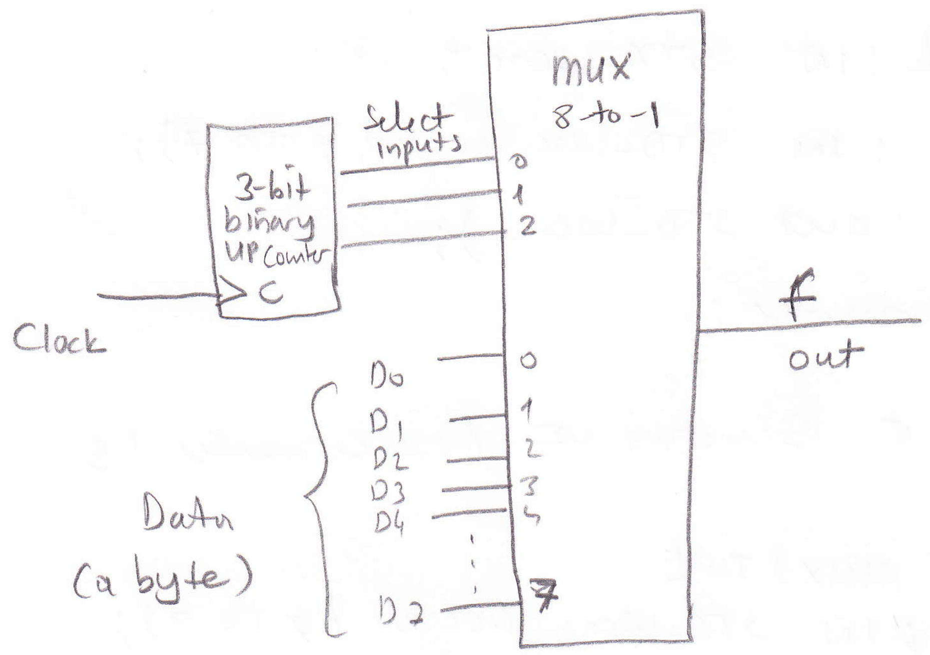
b)

```
entity SHIFTER IS
PORT( w : IN STD-LOGIC-VECTOR (0 TO 3) ;
      sel: IN STD-LOGIC-VECTOR (1 DOWNTO 0);
      f : OUT STD-LOGIC);
END SHIFTER ;
```

5

```
ARCHITECTURE Structure OF SHIFTER IS
 BEGIN
 COMPONENT FOURTOONE MUX
  PORT ( W : IN  STD_LOGIC_VECTOR (0 TO 3);
         SEL : IN STD_LOGIC_VECTOR (1 DOWN TO 0);
         f : OUT STD_LOGIC);
  END FOURTOONE MUX;
  SIGNAL: LOW = '0';

  mux0 : PORTMAP FOURTOONEMUX ((LOW,W(1),LOW,W(0)), SEL, f(0));
  mux1 : PORTMAP FOUR TO ONE MUX ((LOW,W(2),W(0),W(1)), SEL, f(1));
  mux2 : PORTMAP FOUR TO ONE MUX ((LOW,W(3),W(1),W(2)), SEL, f(2));
  mux3 : PORTMAP FOUR TO ONE MUX ((LOW,LOW,W(2),W(3)), SEL, f(3));
  END STRUCTURE;
```

③



a) What do you Think That the circuit is performing? explain

b) Write the VHDL code for the circuit using the counter and the multiplexer as components (assume these components were given, use them in the main code only).

SOLUTION:

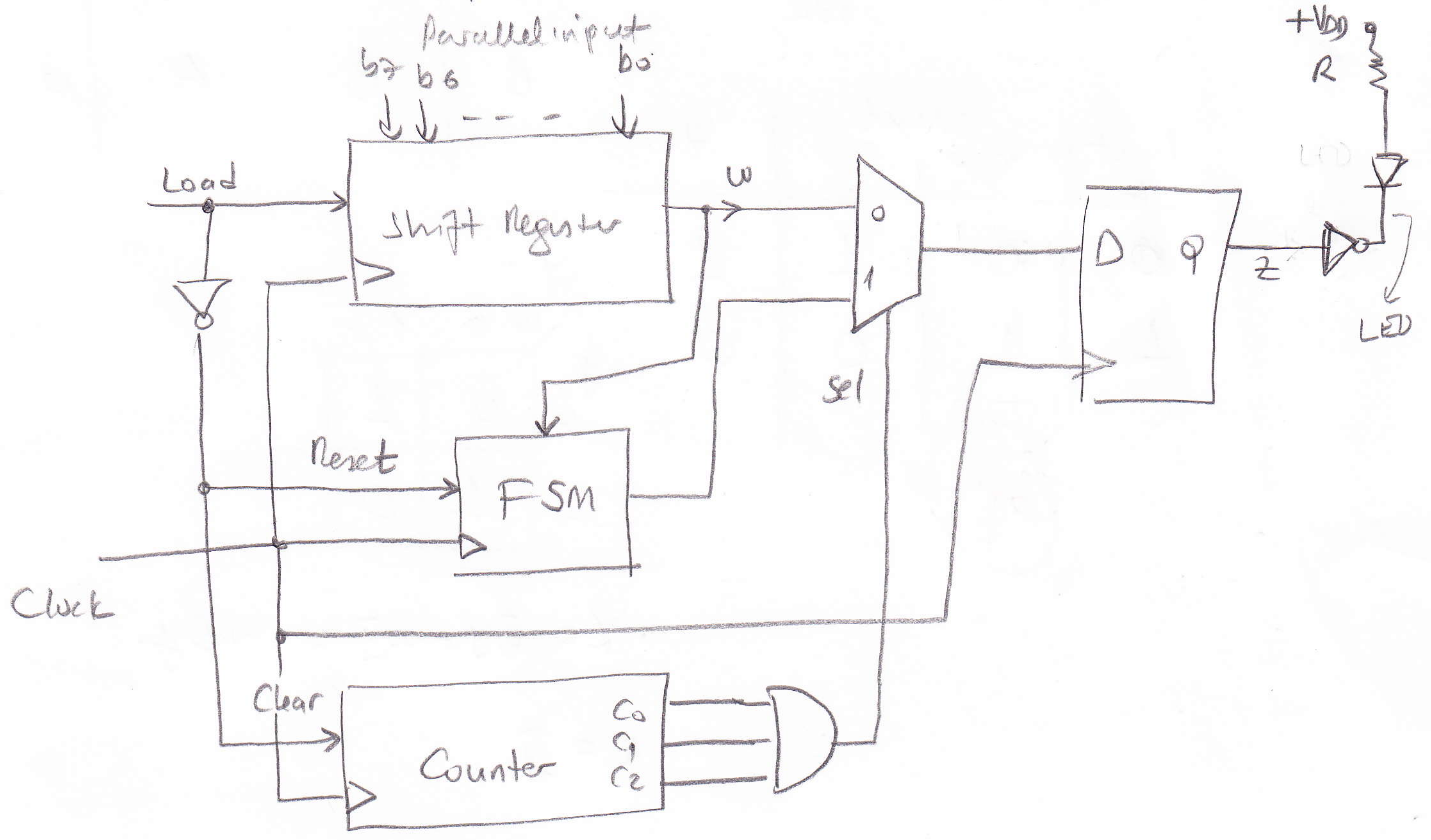
a) This circuit converts 8 bit parallel data into 8-bit serial data. At each clock cycle the counter advances 1 step. The outputs of the counter are used as select inputs of the multiplexer. When counter advances, each input of Data is connected to the output in a sequence one after another at each clock cycle. (10p)

b)

```
ENTITY parserconverter IS
 PORT ( Clock : IN  STD_LOGIC ;
        Data  : IN  STD_LOGIC_VECTOR ( 0 TO 7 ) ;
        f     : OUT STD_LOGIC ) ;
END Parserconverter ;

ARCHITECTURE Structure OF Parserconverter IS
BEGIN
 COMPONENT  mux8To1
 PORT ( Data : IN  STD_LOGIC_VECTOR ( 0 TO 7 ) ;
        Sel  : IN  STD_LOGIC_VECTOR ( 2 DOWN TO 0 ) ;
        f    : OUT STD_LOGIC ) ;
 END COMPONENT ;
 COMPONENT UPCOUNTER
  PORT ( Clock : IN  STD_LOGIC ;
         Count : OUT STD_LOGIC_VECTOR ( 2 DOWNTO 0 ) ) ;
 END COMPONENT ;

SIGNAL  SEL : STD_LOGIC_VECTOR ( 2 DOWN TO 0 ) ;

 Select : UPCOUNTER PORTMAP ( Clock , SEL ) ;
 SERIAL : mux8To1  PORTMAP ( Data , SEL , f ) ;
 END STRUCTURE ;
```

(10p)

④ Design a circuit that determines the even parity condition (even number of 1's) in a byte (8-bit input). The output z becomes 1 and a LED lamp is turned on after 8-bit is count. and the even parity condition is detected. (30p)



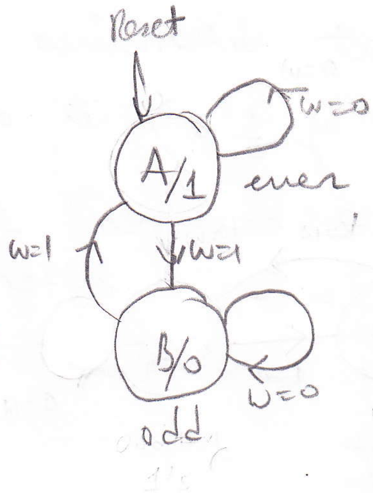
a) Design the FSM to detect the even number of the bits (b0 - - - b7) in the loaded byte into the parallel load shift-right register

b) Write the VHDL codes for the shift register ~~the~~ 3-bit binary up-counter, a 2-to-1 mux and a D FF as components.

c) Write the main VHDL code for the circuit to implement the functionality required, using the components in the main code.

SOLUTION

(a)



4p

Prsnt	Next w=0	Next w=1	z
A	A	B	1
B	B	A	0

Prsnt y	Next w=0 Y	Next w=1 Y	z
0	0	1	1
1	1	0	0

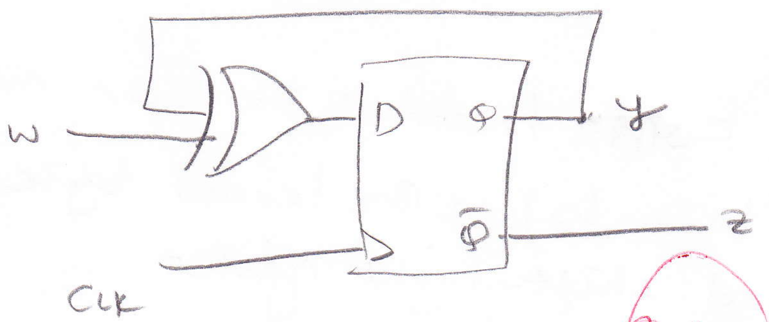
3p

Y

y \ w	0	1
0	0	1
1	1	0

$Y = \bar{w}y + w\bar{y} = w \oplus y$

$z = \bar{y}$



3p

4/1

4b)

```
ENTITY shiftR IS
PORT( Clock, Load : STD_LOGIC ;
        Data        : IN  STD-LOGIC-VECTOR( 7 DOWN TO 0);
        Q           : BUFFER STD-LOGIC_VECTOR (7 DOWN TO 0));
END shiftR;

ARCHITECTURE Behavior OF ShiftR IS
SIGNAL : LOW : IN STD-LOGIC ;
BEGIN
 PROCESS ( Clock, Load, Data)
 BEGIN
  LOW = '0';
  IF Load = '1' THEN
   Q <= Data ;
  ELSIF (Clock'EVENT AND Clock = '1') THEN
  SHIFT: FOR i IN 6 DOWN TO 0 LOOP
          Q(i) <= Q(i+1)                Q(6) <= Q(7) <- 0
           END LOOP ;                   Q(5) <= Q(6)
          Q(7) <= LOW;                  Q(4) <= Q(5)
                                        Q(3) <= Q(4)
   END IF;                              Q(2) <= Q(3)
   END PROCESS ;                        Q(1) <= Q(2)
   END Behavior ;                       Q(0) <= Q(1)
```

4P

```
ENTITY counter IS
PORT( Clear, Clock : IN  STD-LOGIC ;
      C            : OUT STD_LOGIC-VECTOR( 2 DOWN TO 0));
END Counter;
ARCHITECTURE Behavior OF counter IS
SIGNAL Count : STD-LOGIC_VECTOR( 2 DOWN TO 0);
BEGIN
PROCESS ( Clock, Clear)
BEGIN
 IF Clear = '0' THEN
 Count <= "000";
ELSIF ( Clock'EVENT AND Clock = '1') THEN   ->
```

2P

4/2

```
Count <= Count+1 ;
END IF ;
END PROCESS
   C <= COUNT;
END Behaviór ;
```

2p

```
ENTITY mux 2 to 1 IS
PORT ( S : IN STD-LOGIC
       W : IN STD_LOGIC_VECTOR (0 TO 1);
       f : OUT STD-LOGIC) ;
END mux 2 to 1;
ARCHITECTURE Behavior OF mux 2 to 1 IS
BEGIN
PROCESS ( S, W)
BEGIN
IF S = '0' THEN
   f <= W(0) ;
ELSE f <= W(1);
ENDIF ;
END Process ;
END Behaviór ;
```

2p

```
ENTITY DFF IS
PORT ( D, Clock : IN STD-LOGIC;
       Q        : OUT STD-LOGIC);
END DFF ;
ARCHITECTURE Behaviour OF DFF IS
BEGIN
PROCESS (clock)
BEGIN
IF Clock'EVENT AND Clock = '1' THEN
Q <= D ;
ENDIF;
END PROCESS ; END Behaviór;
```

4/3

4c) MAIN CODE

2

```
ENTITY MAIN IS
PORT (Load, Clock : IN STD-LOGIC;
         Data        : IN STD-LOGIC-VECTOR (7 DOWN TO 0);
         Ledn        : OUT STD-LOGIC);
END MAIN ;
```

2

```
ARCHITECTURE Structure OF MAIN IS
                1        2          3     4        5
SIGNAL : Sel ; FSMout, W, muxOUT, Z, Clear: STD-LOGIC;
                6
SIGNAL : COUT : STD-LOGIC-VECTOR (2 DOWN TO 0);
```

2

```
COMPONENT ShiftR
 PORT (Clock, Load ; LOW: IN STD-LOGIC;
         Data          : IN STD-LOGIC-VECTOR (7 DOWN TO 0);
          Q            : BUFFER STD-LOGIC-VECTOR (7 DOWN TO 0));
 END COMPONENT;

 COMPONENT counter
 PORT ( Clear, Clock : IN STD-LOGIC;
          C             : OUT STD-LOGIC-VECTOR (2 DOWN TO 0));
  END COMPONENT

 COMPONENT mux2to1
 PORT ( S : IN STD-LOGIC;
        W : IN STD-LOGIC-VECTOR (0 TO 1);
        f : OUT STD-LOGIC);
 END COMPONENT

 COMPONENT DFF
 POR( D, Clock : IN STD-LOGIC;
      Z        : OUT STD-LOGIC;
 END COMPONENT

 TYPE STATE-TYPE (A, B)
  SIGNAL y : STATE-TYPE ;
  BEGIN
   Sel <= COUT(2) AND COUT(1) AND COUT(0);
   Clear <= NOT Load ; (Clear is used for Reset
                         in the FSM)
```

4/4

```
w <= Q(7);
Count : counter PORT MAP ( Clear, Clock , Cout );
Shift : shiftR PORT MAP ( Clock, Load, Data, Q );
mux : mux2to1 PORTMAP ( Sel, w, FSmout, muxout );
DFlip : DFF  PORTMAP ( muxout , Clock, Z ) ;
```
2

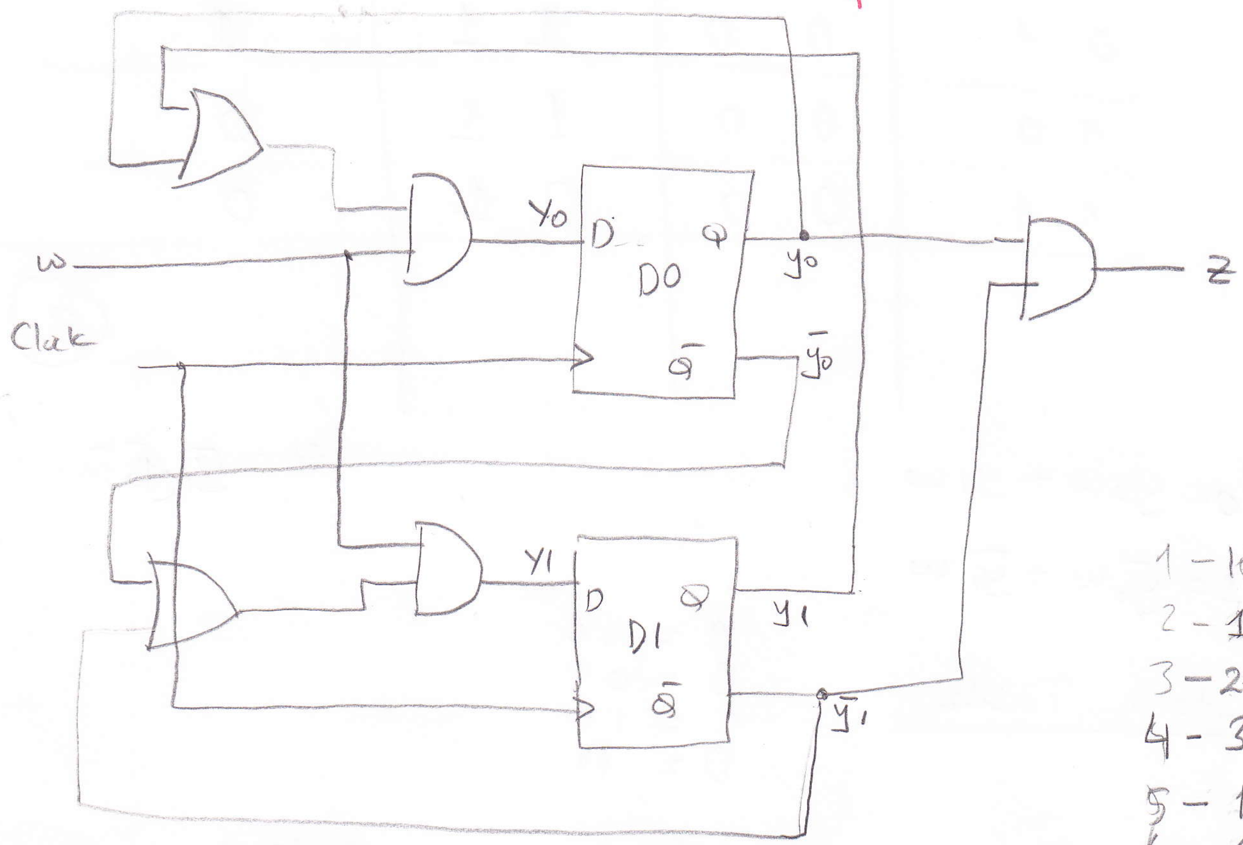
```
FSM : PROCESS (Clock, w, Clear)     as Reset input
BEGIN
 IF Clear = '0' THEN
     y <= A ;
 ELSE (Clock'EVENT AND Clock = '1') THEN
 Case y is
    When A =>
     IF w = '1' THEN    y <= B ;
     ELSE  y <= A ;
     END IF ;
    When B =>
     IF w='1' THEN y <= A ;
     ELSE y <= B ;
     END IF ;
     END CASE ;
     END PROCESS ;
```
$$\boxed{\texttt{z <= '1' when y = A ELSE '0';}}$$ ← * Moore machine

2

```
   Ledn <= z ;
   END Behavior ;
```

4/5

(5) Derive the state table for the circuit given below. What sequence of input values on wire w is detected by this circuit? Explain. (15p)



1	10
2	10
3	20
4	30
5	15
6	15

SOLUTION

$Y_0 = (y_0 + y_1) \cdot w$

$Y_1 = (\bar{y_0} + \bar{y_1}) \cdot w$

$z = y_0 \bar{y_1}$

3

There are 2 FFs.
So the system can have at most 4 states.

It is a MOORE type machine.
The output is not a function of the input (w).

$\rightarrow$

State Assigned Table

Present States	Next States				output z
	$w=0$		$w=1$		
$y_1\ y_0$	Y_1	Y_0	Y_1	Y_0	z
0 0	0	0	1	0	0
0 1	0	0	1	1	1
1 0	0	0	1	1	0
1 1	0	0	0	1	0

3

$Y_0 = y_0 w + y_1 w$

$Y_1 = \bar{y}_0 w + \bar{y}_1 w$

$z = \bar{y}_1 \cdot y_0$

A = 00
B = 01
C = 10
D = 11

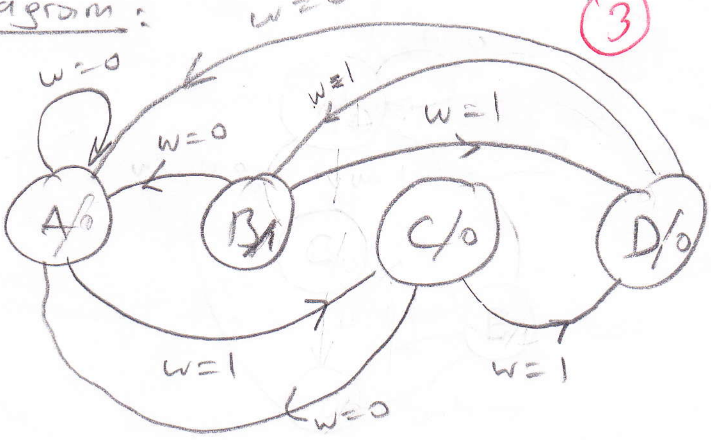
State Table

Present State	Next State		Output
	$w=0$	$w=1$	z
A	A	C	0
B	A	D	1
C	A	D	0
D	A	B.	0

3

State Diagram :

3

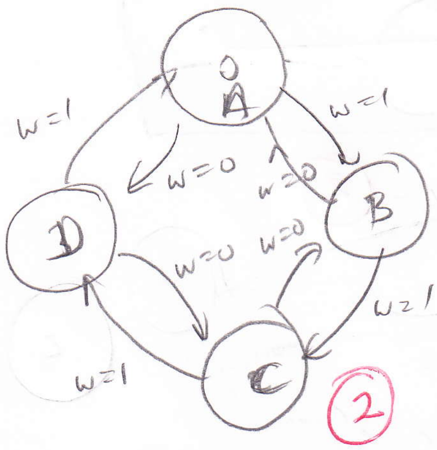


3

The output becomes 1 when 3 1's come one after another Then it becomes 1 at the odd number of inputs, i.e. 5, 7, 9, 11 etc.

(6) Design a counter that counts 0, 1, 2, 3 (up) when w = 1, and 3, 2, 1, 0 (down) when w = 0. Write down the VHDL codes for the circuit. (15p)

State Diagram



present $y_1 y_0$	next w=0 $Y_1 Y_0$	next w=1 $Y_1 Y_0$	Count z
A	D	B	0
B	A	C	1
C	B	D	2
D	C	A	3

State Table

2

Present $y_1 y_0$	Next w=0 $Y_1 Y_0$	Next w=1 $Y_1 Y_0$	z
0 0	1 1	0 1	0
0 1	0 0	1 0	1
1 0	0 1	1 1	2
1 1	1 0	0 0	3

State assigned Table

2

Y_1

w \ $y_1 y_0$	00	01	11	10
0	1	0	1	0
1	0	1	0	1

Y_0

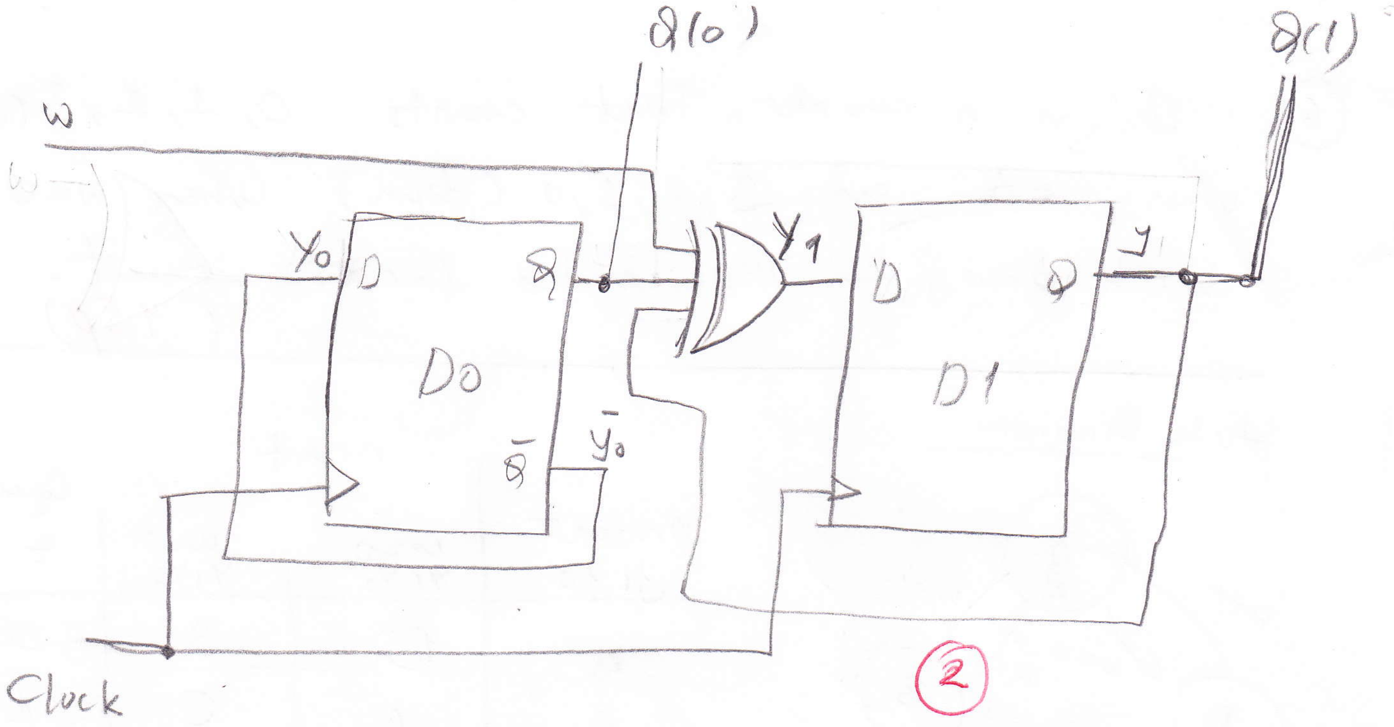
w \ $y_1 y_0$	00	01	11	10
0	1	0	0	1
1	1	0	0	1

$Y_1 = w \oplus y_1 \oplus y_0$

$Y_0 = \overline{y_0}$

6/1

$\searrow$



```
ENTITY counter IS
PORT ( Clock, w : IN STD-LOGIC ;
          Count : OUT STD-LOGIC-VECTOR ( 1 DOWN TO 0 ));
END counter ;
ARCHITECTURE Behavior OF counter IS
TYPE State-type IS ( A, B, C, D );
SIGNAL  y : State_type;
SIGNAL  Q : Standard-logic_vector ( 1 DOWN TO 0 );
BEGIN
PROCESS ( Clock )
BEGIN
Case y IS
  WHEN A =>
    IF w='0' THEN  y <= D ;
    ELSE y <= B ;
    END IF ;
  WHEN B =>
    IF w='0' THEN y <= A ;
    ELSE y <= C ;
    ENDIF ;
```

6/2

```
WHEN C =>
 IF W='0' THEN  y <= B;
 ELSE  y <= D;
 ENDIF;
 WHEN D =>
 IF W='0' THEN  y <= C;
 ELSE  y <= A;
 END IF;
 END CASE;
 END PROCESS;

 PROCESS (clock)
 BEGIN
   CASE y is
   When => A
    Q = "00",
   WHEN => B
    Q = "01";
    When => C
    Q = "10"
    When => (<= OTHERS).
    Q = "11"
  END CASE;
  END PROCESS;
  Count <= Q;
 END Behavior;
```

State machine

Counting

7