(1)

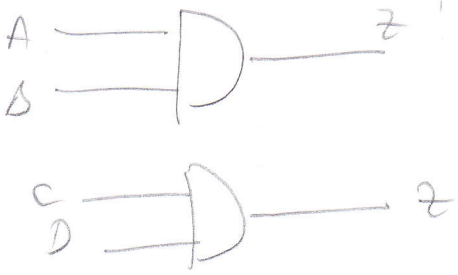
a) Signal Z, A, B, C, D : std_logic
begin
Z <= A AND B ;
Z <= C AND D ;

b) signal Z, A, B, C, D : std_logic
begin
proces (A, B, C, D)
begin
Z <= A AND B ;
Z <= C AND D ;
end proces ;

Drow the corresponding logic circuits to These VHDL codes

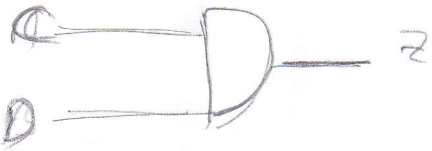
SOL 1 :

a)



(5p)

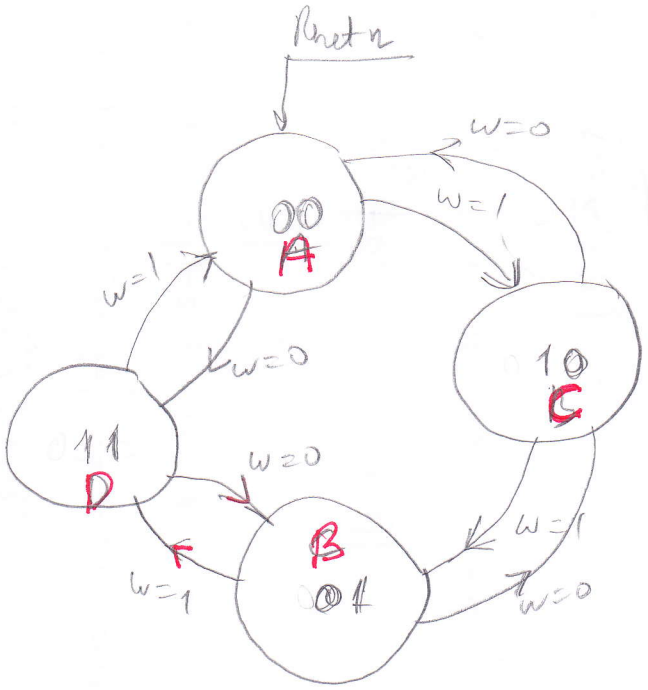
b)



(5p)

(2) a) Design a counter, with T Flip Flops and combinational circuit such that when Resetn=0 it should asynchronously returns to "00" state; when Resetn=1 it should count the sequence 00, 10, 01, 11, 00, 10, ... when w=1, and count the same sequence backward (i.e. 00, 11, 01, 10, 00, 11 --) when w=0.

b) Write the VHDL code for this Finite State Machine (FSM).

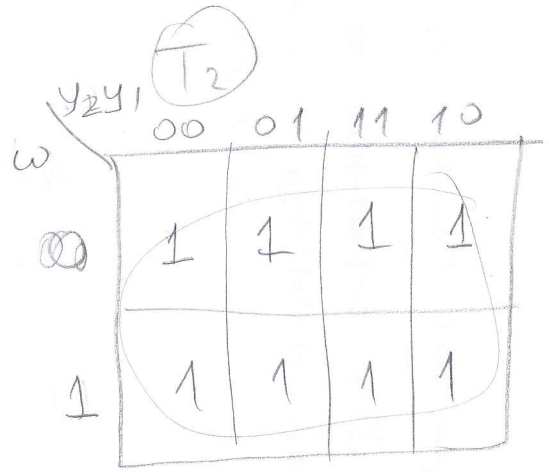


5

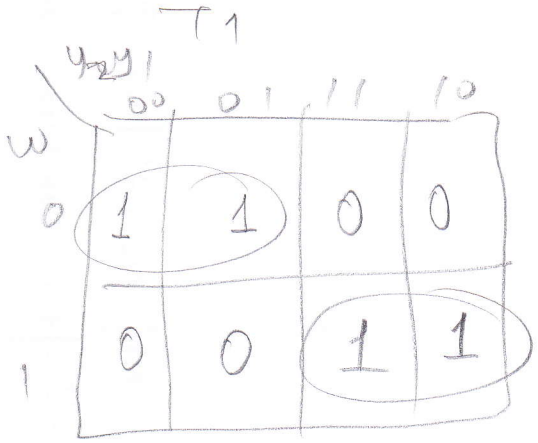
	NEXT w=0	NEXT w=1	Q_t w=0	Q_t w=1
A 00	11 D	10 C	A	A
B 01	10 C	11 D	B	B
C 10	00 A	01 B	C	C
D 11	01 B	00 A	D	D

cont2 (2)

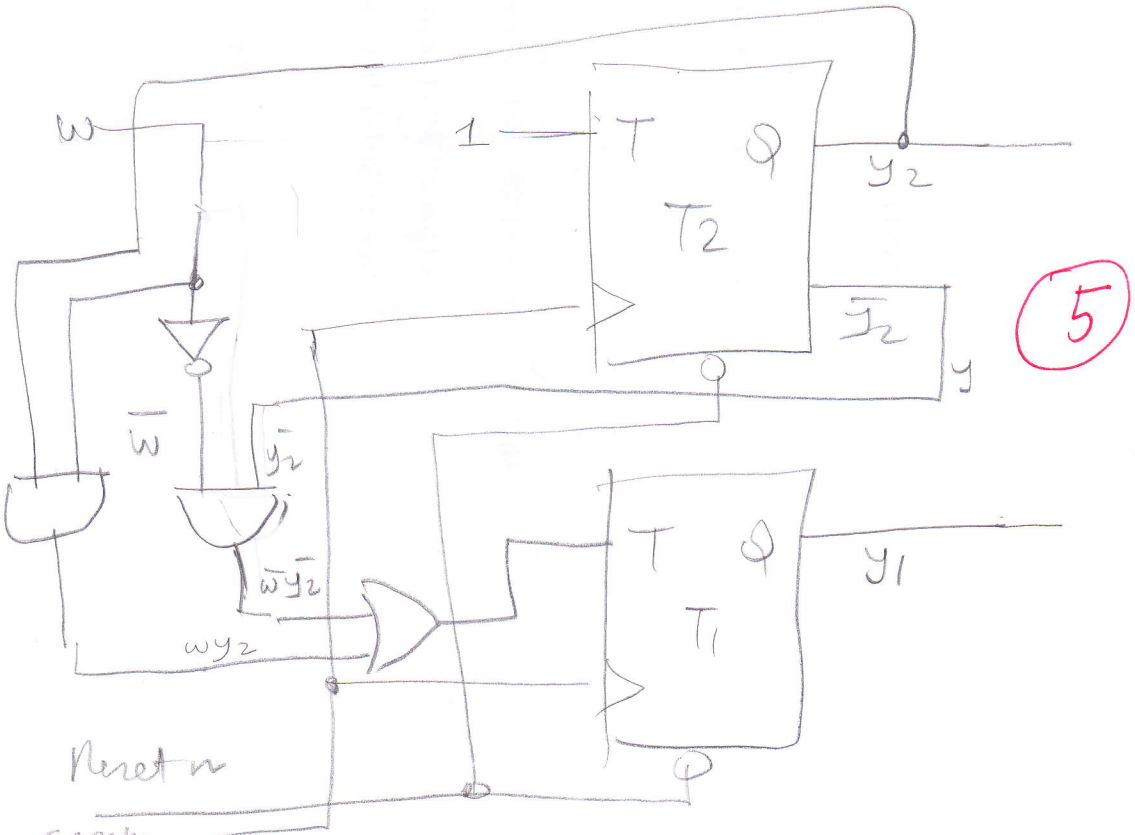
	Next				
	$w=0$		$w=1$		
$y_2\ y_1$	T_2	T_1	T_2	T_1	z
0 0	1	1	1	0	
0 1	1	1	1	0	
1 0	1	0	1	1	
1 1	1	0	1	1	



$T_2 = 1$



$T_1 = \bar{w}\bar{y_2} + w y_2$



5

cont'd (3)

continued

```
ENTITY counter IS
PORT ( clock, Resetn : IN STD_LOGIC;
       w             : IN STD_LOGIC;
       z             : OUT STD_LOGIC_VECTOR (0 TO 1));
END counter;

ARCHITECTURE Behavior OF counter IS
TYPE State_type IS (A, B, C, D);
SIGNAL y : State_type;
BEGIN
  PROCESS (Resetn, Clock)
  BEGIN
    IF Resetn = '0' THEN
        y <= A;
    ELSIF (Clock'EVENT AND Clock = '1') THEN
      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;
          END IF;
        WHEN C =>
```

10

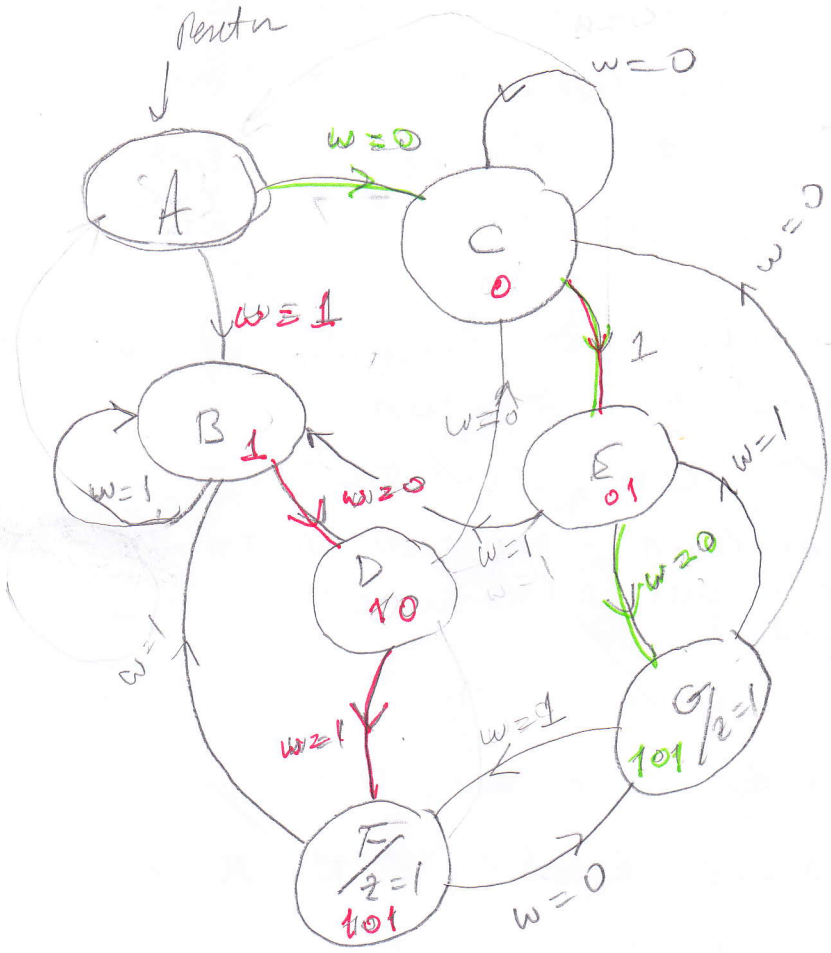
cont(4)

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

z <= y;
END Behavior;
```

SOLUTION

(3) 101
010

STATE DIAGRAM



FDC

w = 0 0 0 0 1 0 1 1 0 1 0 1 1 0 0 0 1 0 1 0 0 0
0 0 0 0 0 1 1 0 0 1 1 1 0 0 0 0 1 1 1 0

$Y_2 Y_1 Y_0$	w=0 $Y_2 Y_1 Y_0$	w=1 $Y_2 Y_1 Y_0$	z
1 A	C	B	0
2 B	D	B	0
3 C	C	E	0
4 D	C	F	0
5 E	G	B	0
6 F	G	B	1
7 G	C	F	1

	Present		Next		output
		w=0	w=1		
	$y_2 y_1 y_0$	$Y_2 Y_1 Y_0$	$Y_2 Y_1 Y_0$		z
1 9 A	0 0 0	0 1 0	0 0 1		0
2 10 B	0 0 1	0 1 1	0 0 1		0
3 11 C	0 1 0	0 1 0	1 0 0		0
4 12 D	0 1 1	0 1 0	1 0 1		0
5 13 E	1 0 0	1 1 0	0 0 1		0
6 14 F	1 0 1	1 1 0	0 0 1		1
7 15 G	1 1 0	0 1 0	1 0 1		1
8 16	1 1 1	- - -	- - -		-

Y_2

y_2y_1 \ y_0w	00	01	11	10
00	0_1	0_9	0_{10}	0_2
01	0_3	1_{11}	1_{12}	0_4
11	0_7	1_{15}	$-_{16}$	$-_8$
10	1_5	0_{13}	0_{14}	1_6

$Y_2 = y_1 w + y_2 \bar{y}_1 \bar{y}_0 w$

Y_1

y_2y_1 \ y_0w	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	1	0	-	-
10	1	0	0	1

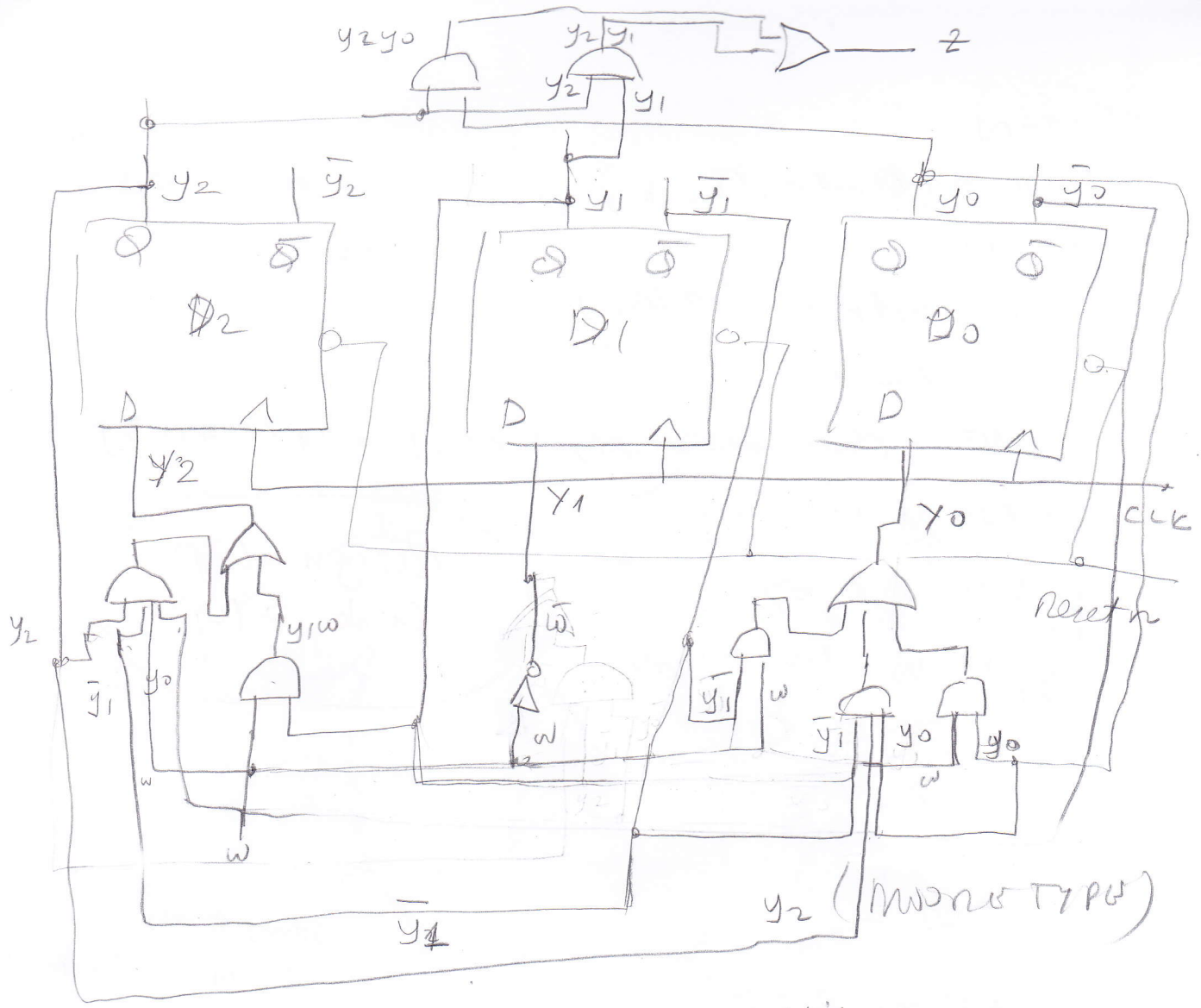
$Y_1 = \bar{w}\bar{w}\, y_2 \bar{y}_1 \bar{y}_0$

y_2y_1 \ y_0w	00	01	11	10
00	0	1	1	1
01	0	0	1	0
11	0	0	-	-
10	0	1	1	0

$Y_0 = \bar{y}_1 w + y_0 w + \bar{y}_2 \bar{y}_1 y_0$

y_2 \ y_1y_0	00	01	11	10
0	0	0	0	0
1	0	1	-	1

$z = y_2 y_0 + y_2 y_1$



seqdet

VHDL CODE

(FROM STATE DIAGRAM)

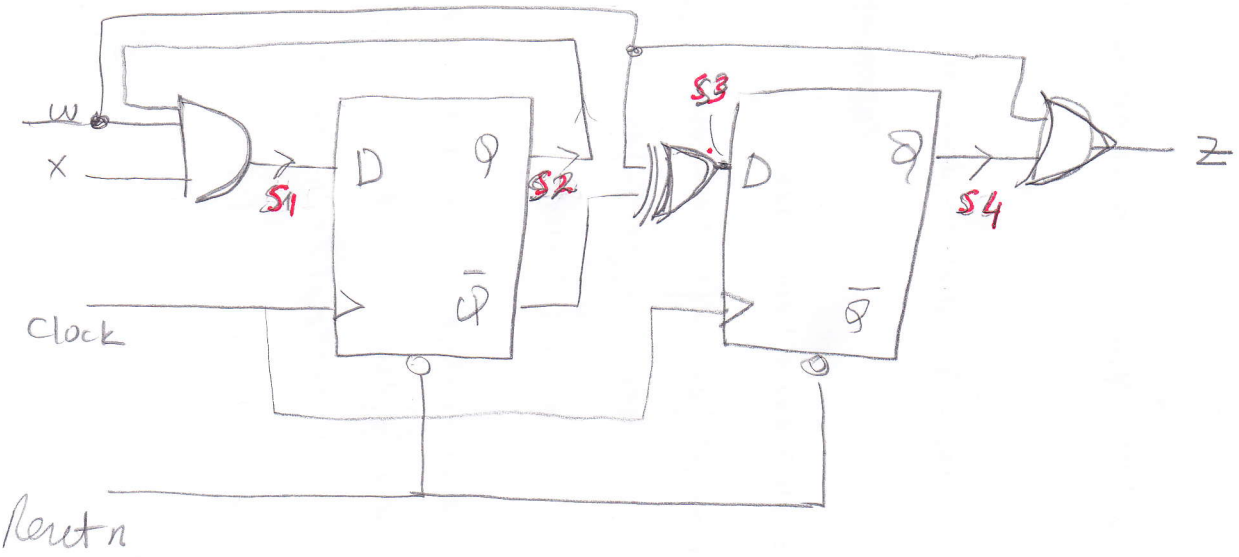
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY seqdet IS
PORT ( clock, Resetn, w : IN  STD_LOGIC;
       z                : OUT STD_LOGIC);

END seqdet
ARCHITECTURE Behavior OF seqdet IS
TYPE State_type IS (A, B, C, D, E, F, G);
SIGNAL y : State_type;
BEGIN
```

Contd (back)

```
BEGIN
 PROCESS (Resetn, Clock)
 BEGIN
   IF Resetn = '0' THEN
       y <= A;
   ELSIF (Clock'EVENT AND Clock = '1') THEN
   CASE y IS
    WHEN A =>
     If w = '0' THEN
        y <= C;
     ELSE
        y <= B;
     END IF;
    WHEN B =>
        IF w = '0' THEN
           y <= D;
        ELSE y <= B;
        END IF;
    WHEN C =>
        IF w = '0' THEN
           y <= C;
        ELSE y <= E;
        END IF;
    When D =>
        IF w = '0' THEN
           y <= C;
        ELSE y <= F;
        END IF
    WHEN E =>
        IF w = '0' THEN
           y <= G;
        ELSE y <= B;
        END IF;
    WHEN F =>
        IF w = '0' THEN
           y <= G;
        ELSE y <= B;
        END IF;
    WHEN G =>
        IF w = '0' THEN
           y <= C;
        ELSE y <= F;
        END IF;
   END CASE;
   END IF;
 END PROCESS;
 z <= '1' WHEN y = F OR G
      ELSE '0';
END BEHAVIOR;
```

Design = Top
Code = Top

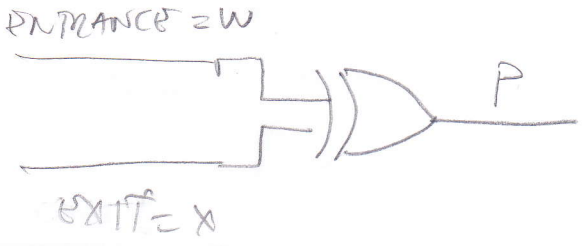
(4) WRITE THE VHDL CODE TO IMPLEMENT THE CIRCUIT



SOL:

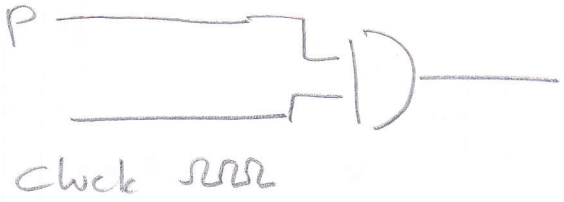
```
ENTITY FFCIRCUIT IS
PORT ( W, X : in  STD_LOGIC;
       Clock, Resetn : in  STD_LOGIC;
       Z : OUT  STD_LOGIC; )
END FFCIRCUIT;
ARCHITECTURE Behavior OF FFCIRCUIT IS
signal S1, S2, S3, S4: std_logic;
BEGIN
 PROCESS ( Resetn, Clock)
BEGIN
IF Resetn = '0' THEN S2 <= '0' AND S4 <= '0';
ELSIF ( clock'EVENT AND clock = '1') THEN
S1 <= W AND X AND S2;
S3 <= NOT S2 XOR W;
S4 <= S1;
END PROCESS
Z <= S4 OR W;
END IF;
```

SOLUTION (5)



W	X	P
0	0	0
0	1	1
1	0	1
1	1	0

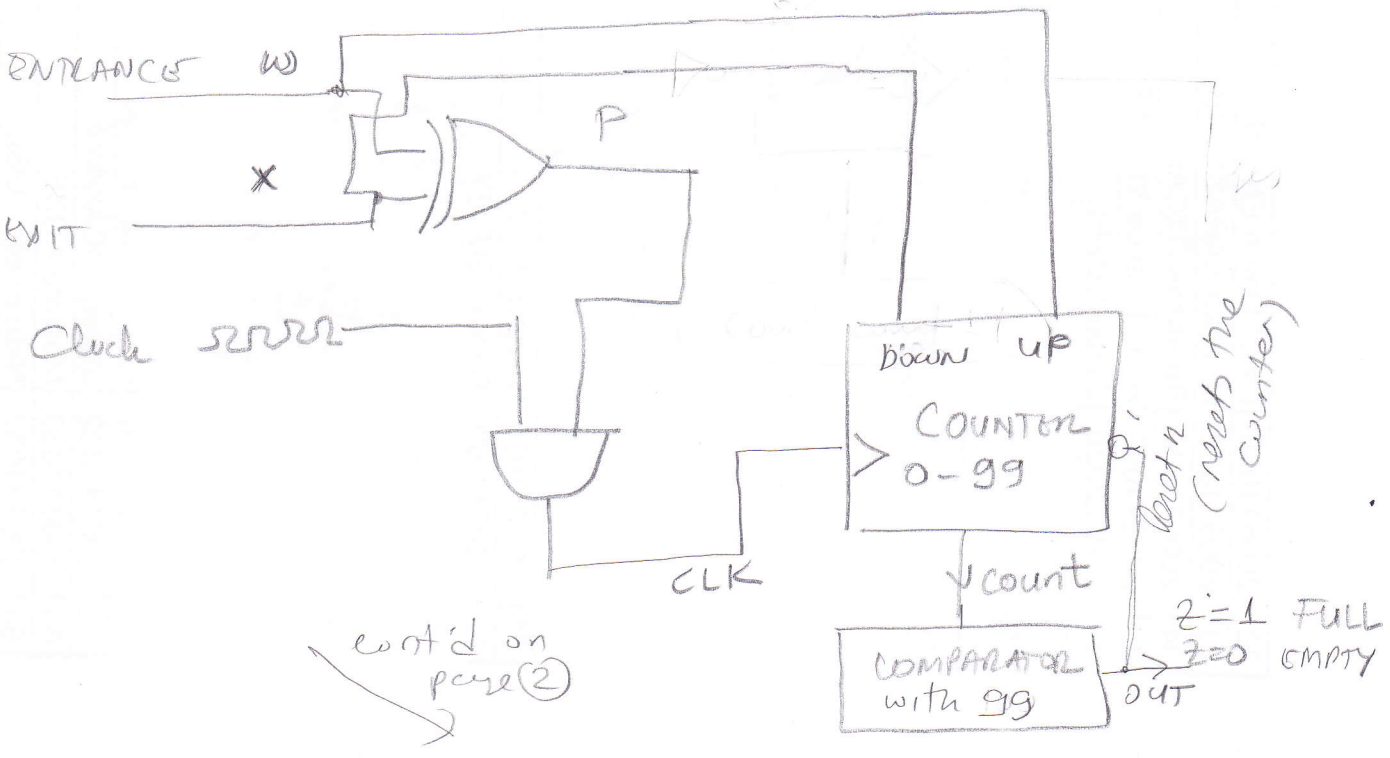
P becomes "1" when entrance or exit signals come from the sensors.
This P signal will be used for generating the clock signal.
A clock is selected (let's select f_{clock} = 1 kHz for example).



This signal will be used to clock the counter.

If X = 1 the counter counts down
If W = 1 the counter counts up.

Design = Up



b) VHDL code

```
library ieee;
use IEEE.STD_LOGIC_1164.ALL
use IEEE.STD_LOGIC_ARITH.ALL
use IEEE.STD_LOGIC_UNSIGNED.ALL

ENTITY CONTROL IS
GENERIC (MODULUS : INTEGER := 99);
PORT (W, X : IN STD_LOGIC;
      Clock, Resetn : IN STD_LOGIC;
      OUT1 : OUT STD_LOGIC);
END CONTROL

ARCHITECTURE Behavior OF CONTROL IS
SIGNAL P : STD_LOGIC;
SIGNAL CLK : STD_LOGIC;
SIGNAL COUNT : INTEGER RANGE 0 TO MODULUS-1;
BEGIN

CLK <= Clock AND P;
PROCESS (Resetn, CLK, X, W, COUNT)
BEGIN
IF RESETn='0' THEN COUNT <= 0;
ELSIF (CLK'EVENT AND CLK='1') THEN
IF X='1' THEN
COUNT <= COUNT-1;
ELSIF W='1' THEN COUNT <= COUNT+1;
IF COUNT >= MODULUS THEN OUT='0';
ELSE OUT1 <= '1'
END IF;
END IF;

END PROCESS;
END Behavior;
```

code = 10p

b) VHDL code

```
* Library ieee;
  - use IEEE.STD_LOGIC_1164.ALL
  - use IEEE.STD_LOGIC_ARITH.ALL
  - use IEEE.STD_LOGIC_UNSIGNED.ALL

ENTITY CONTROL IS
GENERIC (MODULUS : INTEGER := 99);
PORT (W, X   : IN STD_LOGIC;
      Clock, Resetn : IN STD_LOGIC;
      OUT1 : OUT STD_LOGIC);
END CONTROL

ARCHITECTURE Behavior OF CONTROL IS
SIGNAL P : STD_LOGIC;
SIGNAL CLK : STD_LOGIC;
SIGNAL COUNT : INTEGER RANGE 0 TO MODULUS-1;
BEGIN
P
CLK <= CLOCK AND P;
PROCESS (Resetn, CLK, X, W, COUNT)
BEGIN
IF RESETn='0' THEN COUNT <= 0;
ELSIF (CLK'EVENT AND CLK='1') THEN
IF X='1' THEN
COUNT <= COUNT-1;
ELSIF W='1' THEN COUNT <= COUNT+1;
IF COUNT >= MODULUS THEN OUT='0';
ELSE OUT1 <= '1'
END IF;
END IF;

END PROCESS;
END Behavior;
```

code = 10p

(6) SOL6 :