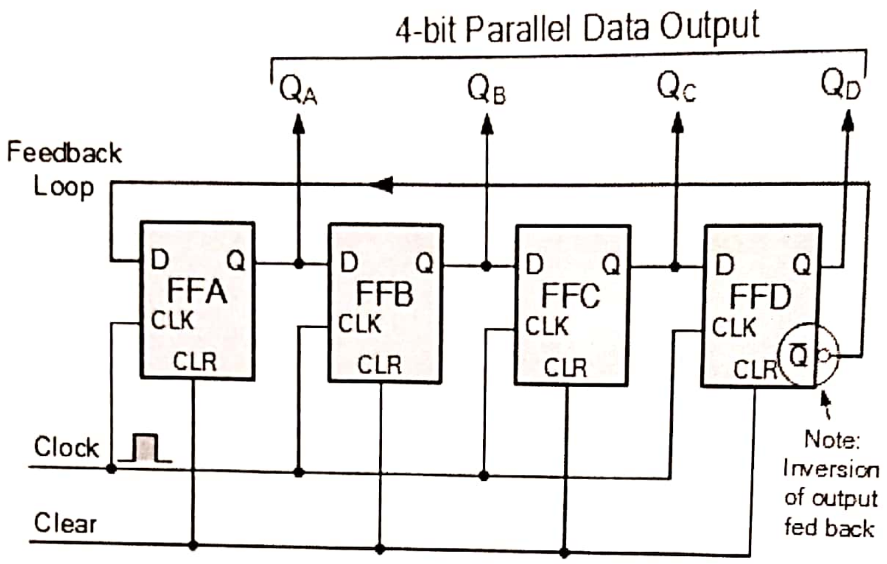
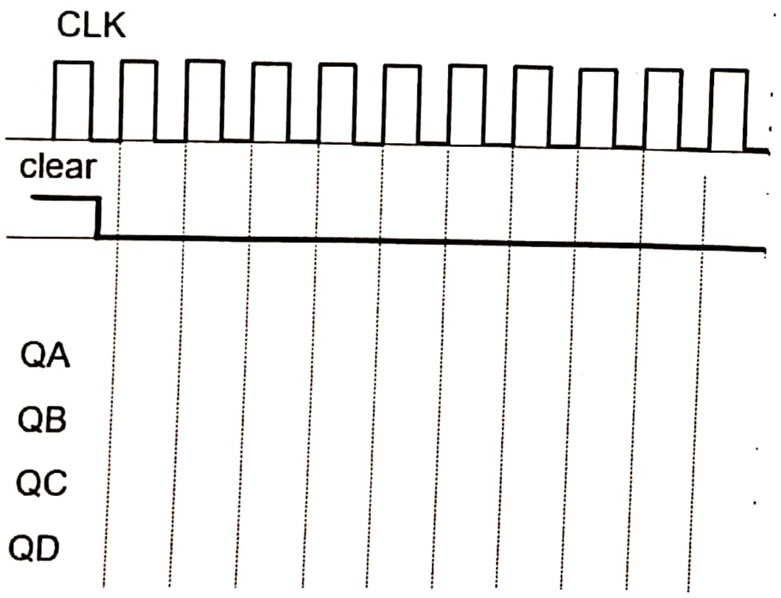
Question 1 (5 points each, total 25 points)

Short questions

a) 4-bit Johnson Ring Counter is shown below. Finish the timing diagram below.

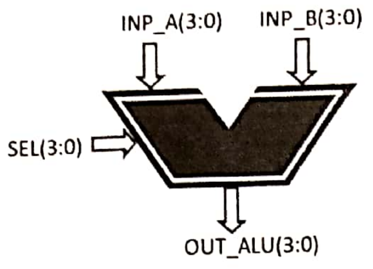




b) Add the following 4-bit unsigned numbers: 8+9. The result is stored in 4-bit word. What is the result? Did the overflow happen?

c) Add the following 4-bit signed (2's complement) numbers: 3+6. The result is stored in 4-bit word. What is the result? Did the overflow happen?

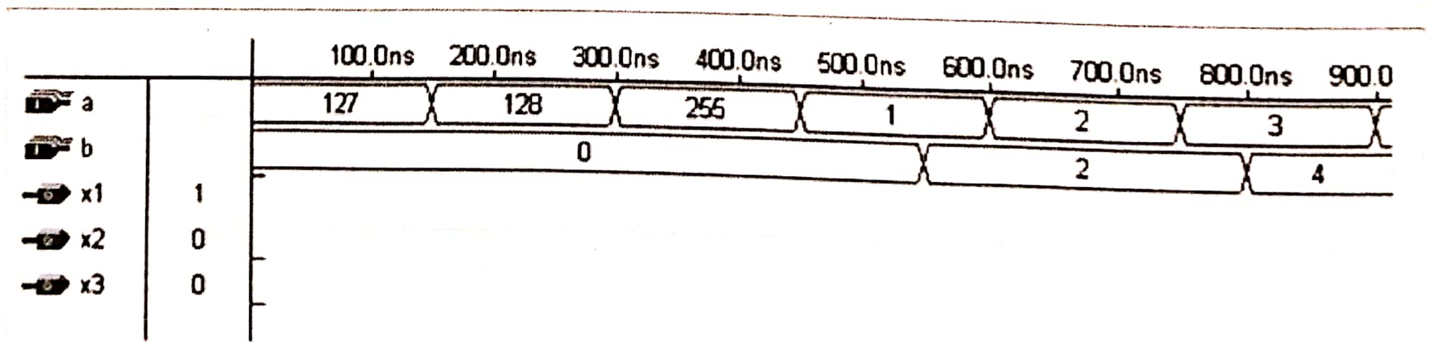
d) Write VHDL code for the following ALU.



Selection Input			Operation Performed
0	0	0	A + B
0	0	1	A - B
0	1	0	A - 1
0	1	1	A + 1
1	0	0	A and B
1	0	1	A or B
1	1	0	not A
1	1	1	A xor B

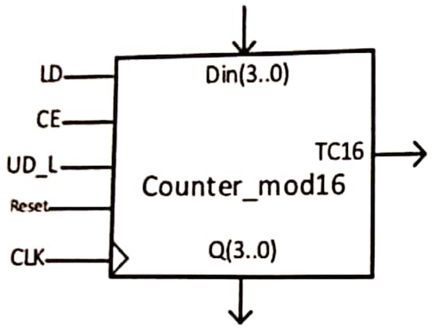
e) The following code is given for the signed comparator. Finish the timing diagram:

```
---- Signed Comparator: ----------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;   -- necessar
----------------------------------------
ENTITY comparator IS
   GENERIC (n: INTEGER := 7);
   PORT (a, b: IN SIGNED (n DOWNTO 0);
         x1, x2, x3: OUT STD_LOGIC);
END comparator;
----------------------------------------
ARCHITECTURE signed OF comparator IS
BEGIN
   x1 <= '1' WHEN a > b ELSE '0';
   x2 <= '1' WHEN a = b ELSE '0';
   x3 <= '1' WHEN a < b ELSE '0';
END signed;
----------------------------------------
```



Question 2 (5 points each, total 20 points)

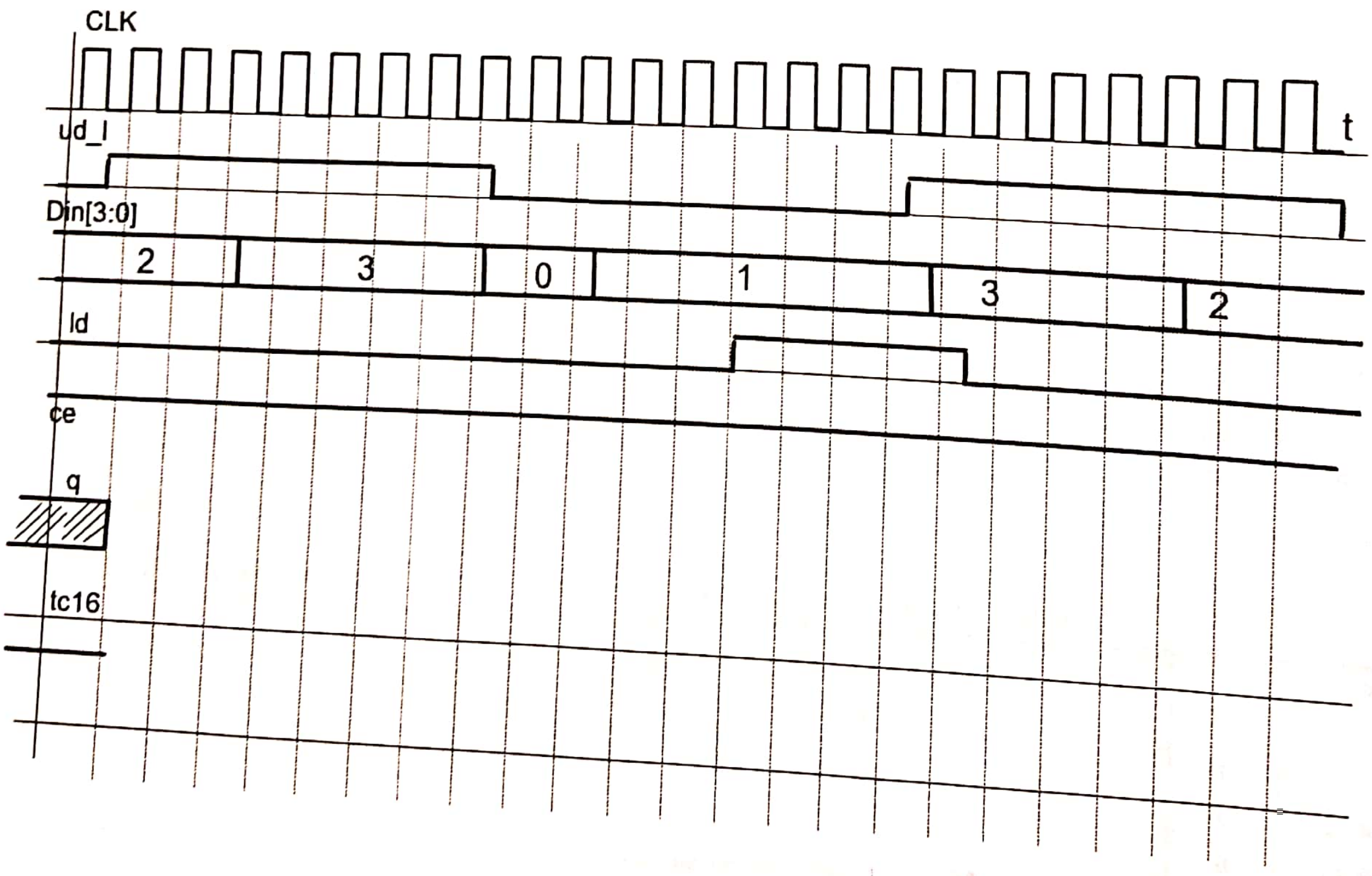
The goal is to design a synchronous, 4-bit (modulo 16), reversible (up and down), binary counter with count enable (CE) and parallel load (LD). The output is the count Q(3:0) as well as terminal count (TC16) as represented in Figure below. TC16=1 when the output Q="1111" when counting Up, as well as when the output Q="0000" when counting Down.



LD	CE	UD_L	Q^+	Synchronous operation after the CLK's rising edge
1	x	x	Din	Parallel load (register data)
0	0	x	Q	Do nothing (inhibit)
0	1	1	$(Q+1)_{mod16}$	Up counting in binary
0	1	0	$(Q-1)_{mod16}$	Down counting in binary

TC16 = '1' when CE = '1' *and* [(Q = 15 *and* UD_L = '1') *or* (Q = 0 *and* UD_L = '0')]; '0' otherwise

1. Design this counter using FSM
2. Design the counter using ASMD algorithm
3. Write VHDL for either 1. or 2.
4. Complete the timing diagram (ce is one all the time, start with q=0 initially)



Question 3A (total 20 points)

In a gym and fitness centre, there are some shower stalls like the one represented in Figure 1 that have to be automated to generate cycles of contrasting hot (48 °C), warm (26°C) and cold (4°C) water sprays simply clicking a single start push button (SB). After clicking the SB, initially warm water flows for 50 s (H = C = '1'), then hot water (H = '1', C = '0') for 10 s, and thirdly cold water (H = '0', C = '1') for 20 s, and this cycle is repeated one more time; finally the system goes idle (H = C = '0') to wait for another user service. During the operation the R_LED turns on.

a) Design the digital control system connected using timed FSM or ASMD. Assume that 1kHz clock is used.
b) Show block diagram of your design
c) Write VHDL code for your design

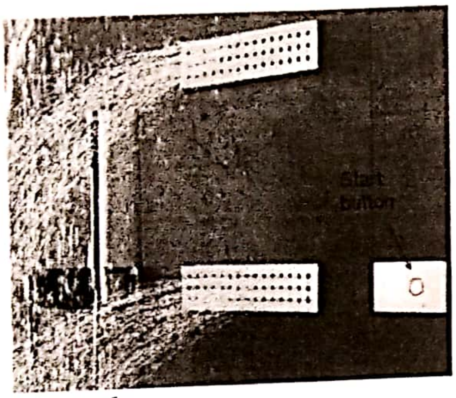


Figure 1

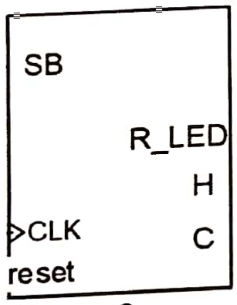


Figure 2

Question 3A (total 20 points)

Derive the minimal flow table that specifies the same functional behavior as the flow table in Figure Question 3A.

Present state	Next state				Output
	$w_2w_1 = 00$	01	10	11	z
A	A	F	C	–	0
B	A	B	–	H	1
C	G	–	C	D	0
D	–	F	–	D	1
E	G	–	E	D	1
F	–	F	–	K	0
G	G	B	J	–	0
H	–	L	E	H	1
J	G	–	J	–	0
K	–	B	E	K	1
L	A	L	–	K	1

Figure: Flow table for Question 3A.

Question 4A (total 20 points)

Consider the following algorithm $z=\max(a^2+b^2, 4*a*b)$.

a) Derive dataflow diagram.

b) Derive the ASMD chart for the schedule. Assume that one arithmetic unit is used and that unit can perform any operation.

c) You need to process input stream of 1000 data points a and b that arrive at the maximum speed that your design supports. How many clock cycles does it take to execute 1000 data points based on ASMD in b)?

d) Implement this circuit as a pipeline circuit so that the clock period that you can achieve is smaller than 25 ns. Show where you would add pipelined registers so that the number of pipeline stages is minimal. Assume the following delays of the components: mul: 19ns, max: 8ns, add/sub 11ns, Tcq=0.5ns and Tsetup=0.5ns. What is the minimum clock period?

e) You need to process input stream of 1000 data points a and b that arrive at the maximum speed that your design supports. How many clock cycles does it take to execute 1000 data points based on pipelined design in d)?

Question 4B (total 20 points)

Find a hazard-free minimum-cost POS implementation of the function:

$f(x_1, \ldots, x_4) = \Pi M(0, 2, 3, 7, 10) + D(5, 13, 15)$

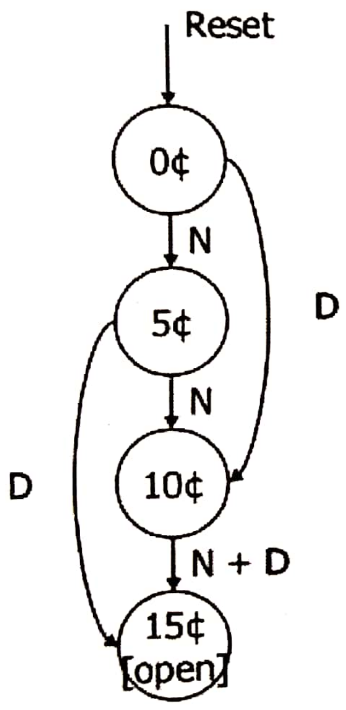
Question 5A (20 points)

FSM below represents a simple vending machine. N is asserted for one clock period when a nickel is inserted into the coin slot. D is asserted when a dime has been deposited. The machine asserts Open for one clock period when 15 cents (or more) has been deposited since the last reset.

Let us enumerate all the possible input sequences that lead to releasing the gum:

- Three nickels in sequence: N, N, N
- Two nickels followed by a dime: N, N, D
- A nickel followed by a dime: N, D
- A dime followed by a nickel: D, N Two dimes in sequence: D, D

The FSM is shown below. Using the state transition table Karnough maps, derive logic for this FSM and implement the circuit.



Question 5B (20 points)

Write a description of a system that determines whether two 8-bit std_logic_vector inputs are both even or both odd parity. The entity's name is parity_comp. The input vectors are x and y. The output is the std_logic signal same_par. Signal same_par is a '1' if both input vectors are even parity or both are odd parity; otherwise, it is a '0'. Use this function parity_even from this listing.

```
library ieee;
use ieee.std_logic_1164.all;
entity parity_even_entity is
port (
        in_vector : in std_logic_vector(7 downto 0);
        even : out std_logic
);
end parity_even_entity;

architecture behavioral of parity_even_entity is

        function parity_even (s: std_logic_vector(7 downto 0)) return std_logic is
                variable result : std_logic := '1';
                begin
                for i in 7 downto 0 loop
                        if s(i) = '1' then
                                result := not result;
                        end if;
                end loop;
                return result;
        end parity_even;
        begin
        even <= parity_even(in_vector);
end behavioral;
```