CEG3155: DGD 4

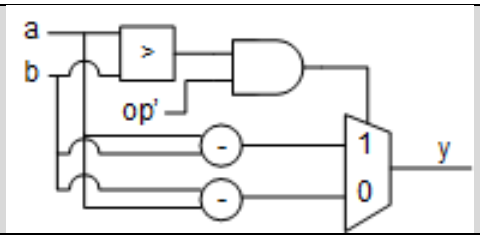
Questions

Question 1 (Chu 5.5)

Derive the conceptual diagram for the following code segment:

```
3   if (a > b and op="00") then
4       y <= a - b;
5       z <= a - 1;
6       status <= '0';
7   else
8       y <= b - a;
9       z <= b - 1;
10      status <= '1';
11  end if;
12
```

Solution



Question 2 (Chu 5.8)

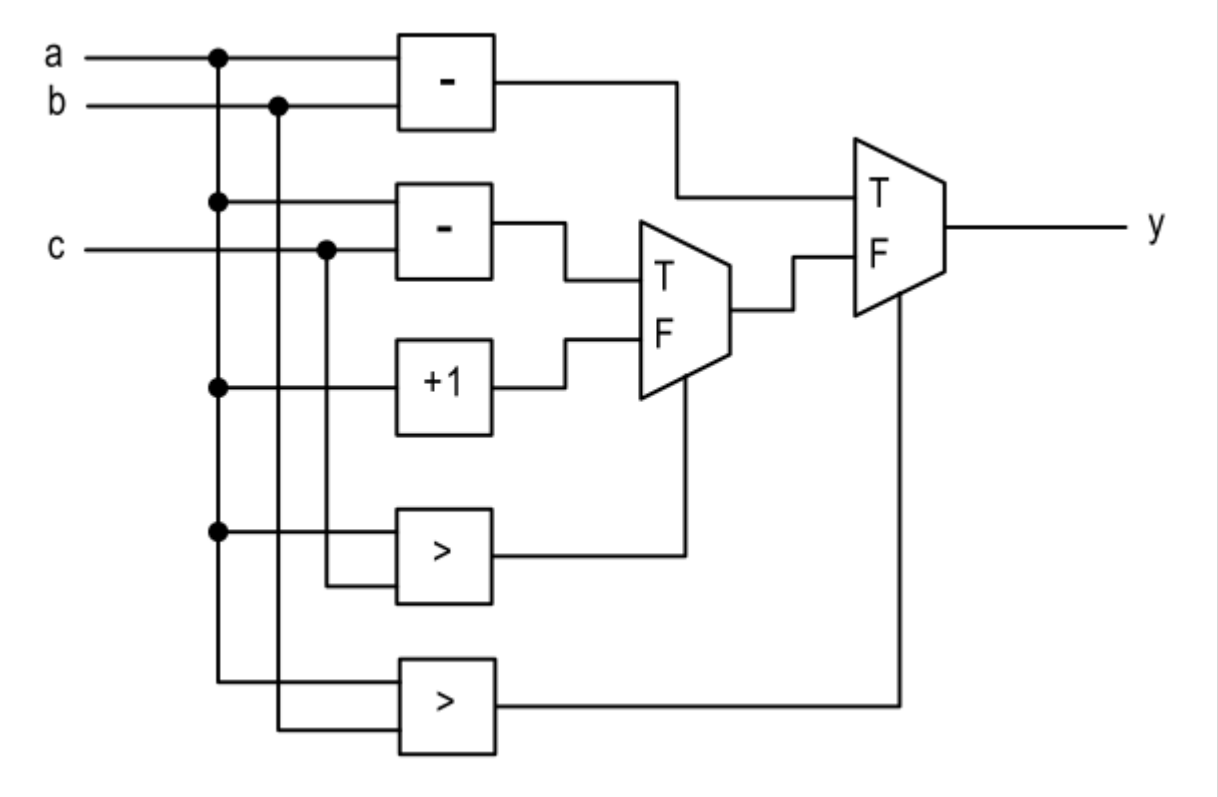
Consider the following code segment:

```
3   if (a > b) then
4       y <= a - b;
5   else
6       if(a > c)
7           y <= a - c;
8       else
9           y <= a + 1;
10      end if;
11  end if;
12
```

a) Draw the conceptual diagram.
b) Rewrite the code using two concurrent conditional signal assignment statements.
c) Rewrite the code using one concurrent conditional signal assignment statement.
d) Rewrite the code using one case statement.

Solution

a)



b)

```
y_tmp <= a - c when a > c else
         a + 1;
y <= a - b when a > b else
     y_tmp;
```

c)

```
y <= a - b when a > b else
     a - c when a > c else
     a + 1;
```

d)

```
    tmp(1) <= '1' when a > b else '0';
    tmp(0) <= '1' when a > c else '0';
    . . .
    case tmp is
        when "11"|"10" => y <= a - b;
        when "01"      => y <= a - c;
        when others    => y <= a - 1;
    end case;
```

Question 3 (Chu 5.12)

Consider the shift-left circuit discussed in Problem 4.6. The inputs include a, which is an 8-bit signal to be shifted, and ctrl, which is a 3-bit signal specifying the amount to be shifted. Both are with the std_logic_vector data type. The output y is an 8-bit signal with the std_logic_vector data type. Use a case statement to derive the circuit and draw the conceptual diagram.

Solution

```
----------------------------------------------
-- 8-bit shift left circuit
----------------------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity shift_left is
   port(
      a: in std_logic_vector(7 downto 0);
      ctrl: in std_logic_vector(2 downto 0);
      y: out std_logic_vector(7 downto 0)
   );
end shift_left ;

architecture case_arch of shift_left is
begin
   process(ctrl,a)
   begin
      case ctrl is
        when "000" => y <= a;
        when "001" =>
            y <= a(6 downto 0) & "0";
        when "010" =>
            y <= a(5 downto 0) & "00";
        when "011" => "=>
            y <= a(4 downto 0) & "000";
        when "100"=> "=>
            y <= a(3 downto 0) & "0000";
        when "101" => "=>
            y <= a(2 downto 0) & "00000";
        when "110" => "=>
            y <= a(1 downto 0) & "000000";
        when others => "=>
            y <= a(0) & "0000000";
      end case;
   end process;
end case_arch ;
```

Question 4

Using a conditional signal assignment, write VHDL code for an 8-to-3 priority encoder.

Solution

Priority encoder works as follows: If two or more single-bit inputs are at logic 1, the input with the high priority will have effect.

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity priEnc is
6      port (
7          A: In unsigned (7 downto 0);
8          Y: out unsigned (2 downto 0)
9          Valid: out std_logic;
10     );
11 end priEnc;
12
13 architecture priEnc_arch of priEnc is
14 begin
15     Y <= "000" when A(0) = '1' else
16          "001" when A(1) = '1' else
17          "010" when A(2) = '1' else
18          "011" when A(3) = '1' else
19          "100" when A(4) = '1' else
20          "101" when A(5) = '1' else
21          "110" when A(6) = '1' else
22          "111" when A(7) = '1' else
23          "XXX";
24 end priEnc_arch;
25
```

Question 5

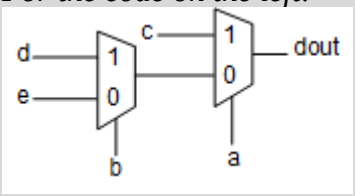
Explain the difference between these two VHDL statements and show their conceptual implementations (i.e. draw the circuits):

```
3  if (a='1') then
4      dout <= c;
5  elsif (b='1')
6      dout <= d;
7  else
8      dout <= e;
9  end if;
```

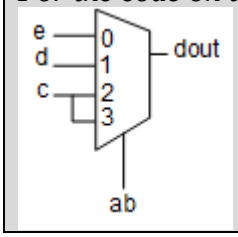
```
3  ab <= a & b;
4  case ab is
5      when "10"|"11" => dout <= c;
6      when "01" => dout <= d;
7      when others => dout <= e;
8  end case;
9
```

Solution

For the code on the left:



For the code on the right:



Question 6

Rewrite the combinational VHDL code given below so that it uses one adder. Show the conceptual diagram (i.e. draw the circuit) of both designs.

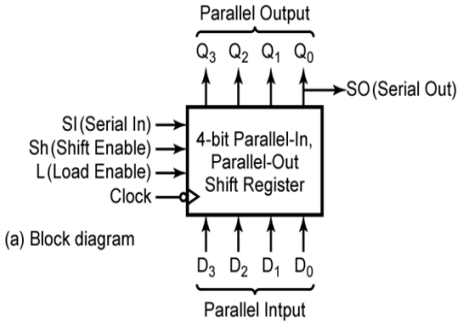
```
with  sel_exp  select
    r <= a+b  when "00",
         a+c when "01",
         d+1   when others;
```

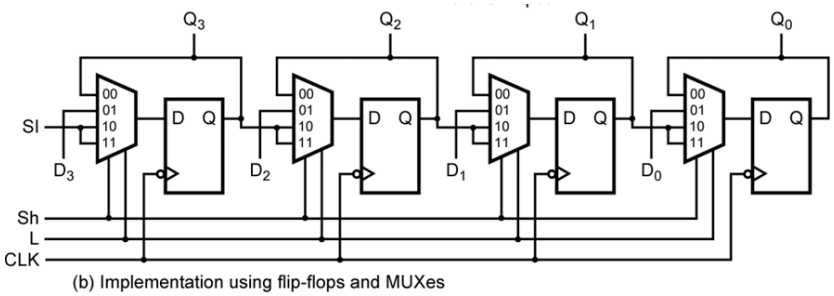
Solution

```
with sel_exp select
    temp1 <= a when "00"|"01",
             d when others;
    temp2 <= b when "00",
             c when "01",
             1 when others;
```

Question 7

The following figure shows a parallel-in parallel out right shift register, Part a is the block diagram and part b is the implementation using flip flops and multiplexers





Inputs		Next State				Action
Sh (Shift)	Ld (Load)	Q_3^+	Q_2^+	Q_1^+	Q_0^+	
0	0	Q_3	Q_2	Q_1	Q_0	no change
0	1	D_3	D_2	D_1	D_0	load
1	X	SI	Q_3	Q_2	Q_1	right shift

Show how to make the shift register reverse the order of its bits: i.e. $Q_3^+ = Q_0$, $Q_2^+ = Q_1$, $Q_1^+ = Q_2$, $Q_0^+ = Q_3$

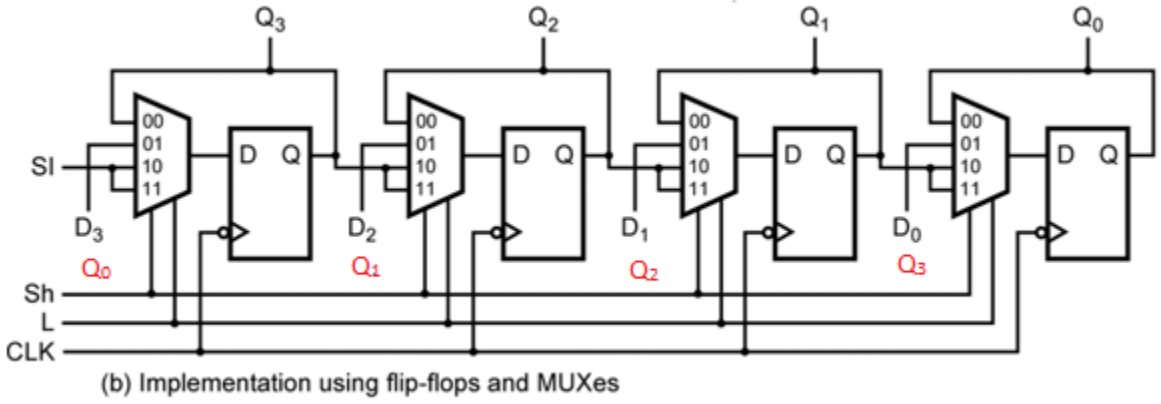
1. Use external connections between the Q output and D inputs, what should the values of sh and L be for a reversal?

2. Change the internal circuitry to allow bit reversal, so that the D inputs may be used for other purposes. Replace Sh and L with A and B and the register operate according to the following table:

Inputs A B	Next states $Q_3^+ Q_2^+ Q_1^+ Q_0^+$	Action
0 0	$Q_3 Q_2 Q_1 Q_0$	No
0 1	$D_3 D_2 D_1 D_0$	Load
1 0	SI $Q_3 Q_2 Q_1$	Right shift
1 1	$Q_0 Q_1 Q_2 Q_3$	Reverse bits

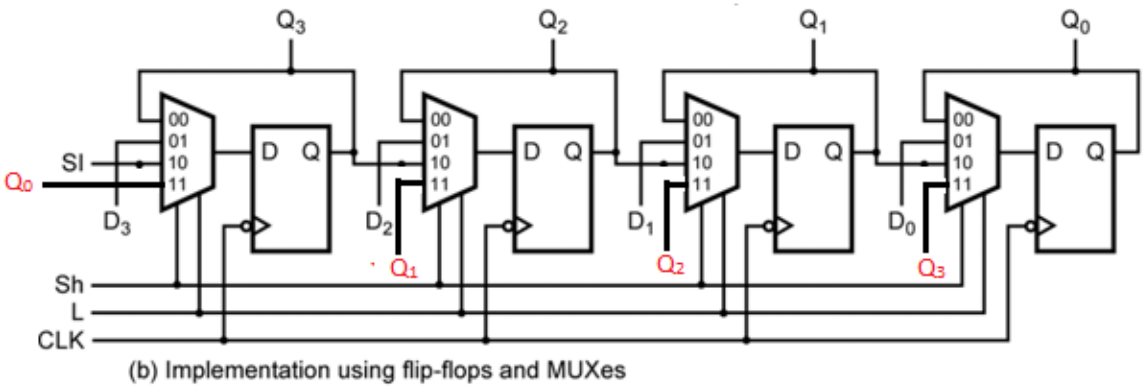
Solution

1.



(b) Implementation using flip-flops and MUXes

2.



(b) Implementation using flip-flops and MUXes