COE 405
Homework#2 Solution

Solution to Q.1:
(i)

```
entity count3 is
port(reset,clk,dir:in bit;                          --inputs
        c:buffer bit_vector(2 downto 0)); --outputs
end count3;
```

(ii)

```
architecture behave of count3 is
begin
        process(clk)
        subtype cnt8 is integer range 0 to 7;--subtype to round the counter
        variable cnt:cnt8 :=0;--variable cnt to follow the count
        begin
                if (clk = '1' and clk'event) then --check for rising edge of the clk
                        if reset = '1' then cnt:=0;
                        else
                                if dir='0' then  --count up and check if you reach the end(7)
                                if cnt=7 then cnt:=0; else cnt:=cnt+1; end if;
                                else      --count down and check if you reach the end(0)
                                if cnt=0 then cnt:=7; else cnt:=cnt-1; end if;
                                end if;
                        end if;

                        case cnt is   --output the count
                        when 0 =>               c<="000";
                        when 1 =>               c<="001";
                        when 2 =>               c<="010";
                        when 3 =>               c<="011";
                        when 4 =>               c<="100";
                        when 5 =>               c<="101";
                        when 6 =>               c<="110";
                        when 7 =>               c<="111";
                        end case;
                end if;
        end process;
end behave;
```

(iii)

```
entity inv is  --inverter to be used in the circuit of the counter
        port(i:in bit; o:out bit);
end inv;
```

```
architecture behavior of inv is
begin
        o<= not(i);
end behavior;


entity and2 is  --2-input and gate to be used in the circuit of the counter
        port(i1,i2:in bit;o:out bit);
end and2;

architecture behavior of and2 is
begin
        o<= i1 and i2;
end behavior;


entity and3 is   --3-input and gate to be used in the circuit of the counter
        port(i1,i2,i3:in bit;o:out bit);
end and3;

architecture behavior of and3 is
begin
        o<= i1 and i2 and i3;
end behavior;

entity or2 is  --2-input or gate to be used in the circuit of the counter
        port(i1,i2:in bit;o:out bit);
end or2;

architecture behavior of or2 is
begin
        o<= i1 or i2;
end behavior;

entity t_flop is  --t_flip-flop to be used in the circuit of the counter
        port(rst,t,clk:in bit;
                q:buffer bit);
end t_flop;

architecture arch of t_flop is
begin
        process(clk)
        begin
                if(clk='1' and clk'event)then
                        if rst='1' then q<='0';
                        else
                                if t='0' then q<=q;
                                else q<= not q;
```

```
                                    end if;
                                end if;
                            end if;
                    end process;
end arch;

architecture struct of count3 is
component inv is
        port(i:in bit; o:out bit);
end component;

component and2 is
        port(i1,i2:in bit;o:out bit);
end component and2;

component and3 is
        port(i1,i2,i3:in bit;o:out bit);
end component and3;

component or2 is
        port(i1,i2:in bit;o:out bit);
end component or2;

component t_flop is
        port(rst,t,clk:in bit;
                q:buffer bit);
end component t_flop;

--internal signals(refer to the circuit diagram)
signal s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11:bit;

begin
g1:inv port map(c(0),s2);
g2:inv port map(dir,s1);
g3:inv port map(c(1),s6);
g4:and2 port map(s2,dir,s3);
g5:and2 port map(s1,c(0),s4);
g6:and3 port map(s2,s6,dir,s7);
g7:and3 port map(c(0),c(1),s1,s8);
g8:or2 port map(s3,s4,s5);
g9:or2 port map(s7,s8,s9);

f0:t_flop port map(reset,'1',clk,c(0));
f1:t_flop port map(reset,s5,clk,c(1));
f2:t_flop port map(reset,s9,clk,c(2));
end struct;
```

Solution to Q.2:

(i)

```
entity shifter4 is
port(reset,clk,dir,si:in bit;                    --inputs
        s:buffer bit_vector(3 downto 0)); --outputs
end shifter4;
```

(ii)

```
architecture behave of shifter4 is
begin
        process(clk)
        begin
                if (clk = '1' and clk'event) then   --check for rising edge of the clock
                        if reset = '1' then s<="0000";
                        else
                                if dir='0' then
                                s<= s(2 downto 0) & si; --shift left
                                else
                                s<=si & s(3 downto 1); --shift right
                                end if;
                        end if;
                end if;
        end process;
end behave;
```

(iii)

```
entity inv is  --inverter to be used in the circuit of the mux
        port(i:in bit; o:out bit);
end inv;

architecture behavior of inv is
begin
        o<= not(i);
end behavior;


entity and2 is --and gate to be used in the circuit of the mux
        port(i1,i2:in bit;o:out bit);
end and2;

architecture behavior of and2 is
begin
```

```
        o<= i1 and i2;
end behavior;


entity or2 is  --or gate to be used in the circuit of the mux
        port(i1,i2:in bit;o:out bit);
end or2;

architecture behavior of or2 is
begin
        o<= i1 or i2;
end behavior;

entity d_flop is  -- defining the d flip-flop
        port(rst,d,clk:in bit;
                q:buffer bit);
end d_flop;

architecture arch of d_flop is --architecture of the flop
begin
        process(clk)
        begin
                if(clk='1' and clk'event)then
                        if rst='1' then q<='0';
                        else q<= d;
                        end if;
                end if;
        end process;
end arch;

entity mux is  --defining the mux
        port(i0,i1,d:in bit;
                o:out bit);
end mux;

architecture struct of mux is --structral architecture of the mux
component inv is                                --using many gates
        port(i:in bit; o:out bit);
end component inv;

component and2 is
        port(i1,i2:in bit;o:out bit);
end component and2;

component or2 is
        port(i1,i2:in bit;o:out bit);
end component or2;
```

```
signal s1,s2,s3:bit;  --internal signals of the mux
begin

g1:inv  port map(d,s1);
g2:and2 port map(d,i1,s2);
g3:and2 port map(s1,i0,s3);
g4:or2 port map(s2,s3,o);
end struct;

architecture struct of shifter4 is
component mux is
        port(i0,i1,d:in bit;
                o:out bit);
end component mux;

component d_flop is
        port(rst,d,clk:in bit;
                q:buffer bit);
end component d_flop;

signal m0,m1,m2,m3:bit; --internal signals: mi represents the output of the
begin                   --mux and the input of flip-flop
x3:mux port map(s(2),si,dir,m3);
x2:mux port map(s(1),s(3),dir,m2);
x1:mux port map(s(0),s(2),dir,m1);
x0:mux port map(si,s(1),dir,m0);

f3:d_flop port map(reset,m3,clk,s(3));
f2:d_flop port map(reset,m2,clk,s(2));
f1:d_flop port map(reset,m1,clk,s(1));
f0:d_flop port map(reset,m0,clk,s(0));

end struct;
```