

ASSIGNMENT-2
ANANNYA MATHUR
2019TT10953

Since there are only 4 seven segment displays, I have chosen two display modes-

- 1)Hours and minutes
- 2)Minutes and seconds

In these display modes, we will have to generate pulses of various time periods.

If we want a pulse to be generated every 0.01s, the clock has to be modified to 100Hz.

Clock divider for 100 Hz:

- Clock available is 10MHz.
- Count value= $10000000/100=100000$.

```
clk_100hz : process (clk) --clock divider for 100Hz  
variable count1 : bit_vector(15 downto 0):= "0000000000000000";  
begin  
    if (rising_edge(clk)) then  
        if count1 = "1100001101010000" then --upto 50000  
            clk1 <= not clk1;  
            count1 := "0000000000000000";  
        end if;  
        count1 := count1 + 1;  
    end if;  
end process
```

We need a clock divider of 250Hz for cathodes and anodes(to drive the displays of BASYS3 Board).

- Count value= $10000000/250=40000$.

```
clk_250hz : process (clk) --clock divider for 250Hz  
variable count2 : bit_vector(14 downto 0):= "0000000000000000";  
begin
```

```

    if (rising_edge(clk)) then
        if count2 = "100111000100000" then --upto 20000
            clk2 <= not clk2;
            count2 := "0000000000000000";
        end if;
        count2 := count2 + 1;
    end if;
end process

```

Clock Divider for 1Hz(To generate 1s):

-Count value=10000000.

```

clk_1hz : process (clk) --clock divider for 1Hz
variable count3 : bit_vector(22 downto 0):=
"000000000000000000000000";
begin
    if (rising_edge(clk)) then
        if count3 = "10011000100101101000000" then --upto 5000000
            clk3 <= not clk3;
            count3 := "000000000000000000000000";
        end if;
        count3 := count3 + 1;
    end if;
end process

```

Clock divider for 1/60Hz(To generate 1 pulse per 60s=1 pulse per minute):

-Count value=60MHz.

```

clk_1/60hz : process (clk) --clock divider for 1/60Hz
variable count4 : bit_vector(24 downto 0):=
"00000000000000000000000000";
begin
    if (rising_edge(clk)) then
        if count4 = "1110010011100001110000000" then --upto
30000000
            clk4 <= not clk4;
            count4 := "00000000000000000000000000" ;
        end if;
    end if;
end process

```

```

        count4 := count4 + 1;
    end if;
end process

```

CLOCK DIVIDER:

```

entity clock_dividers is

```

```

    Port ( clk : in bit;
           clk_250 : out bit;
           clk_1hz : out bit;
           clk_1/60hz: out bit);

```

```

end clock_dividers;

```

```

architecture Behavioral of clock_dividers is

```

```

    signal clk2, clk3, clk4 : BIT;

```

```

begin

```

```

    clk_250hz : process (clk) --clock divider for 250Hz

```

```

        variable count2 : bit_vector(14 downto 0) := "0000000000000000";

```

```

    begin

```

```

        if (rising_edge(clk)) then

```

```

            if count2 = "100111000100000" then --upto 20000

```

```

                clk2 <= not clk2;

```

```

                count2 := "0000000000000000";

```

```

            end if;

```

```

            count2 := count2 + 1;

```

```

        end if;

```

```

    end process

```

```

    clk_1hz : process (clk) --clock divider for 1Hz

```

```

        variable count3 : bit_vector(22 downto 0) :=

```

```

        "000000000000000000000000";

```

```

    begin

```

```

        if (rising_edge(clk)) then

```

```

            if count3 = "10011000100101101000000" then --upto 5000000

```

```

                clk3 <= not clk3;

```

```

                count3 := "000000000000000000000000";

```

```

            end if;

```

```

            count3 := count3 + 1;

```

```

    end if;
end process

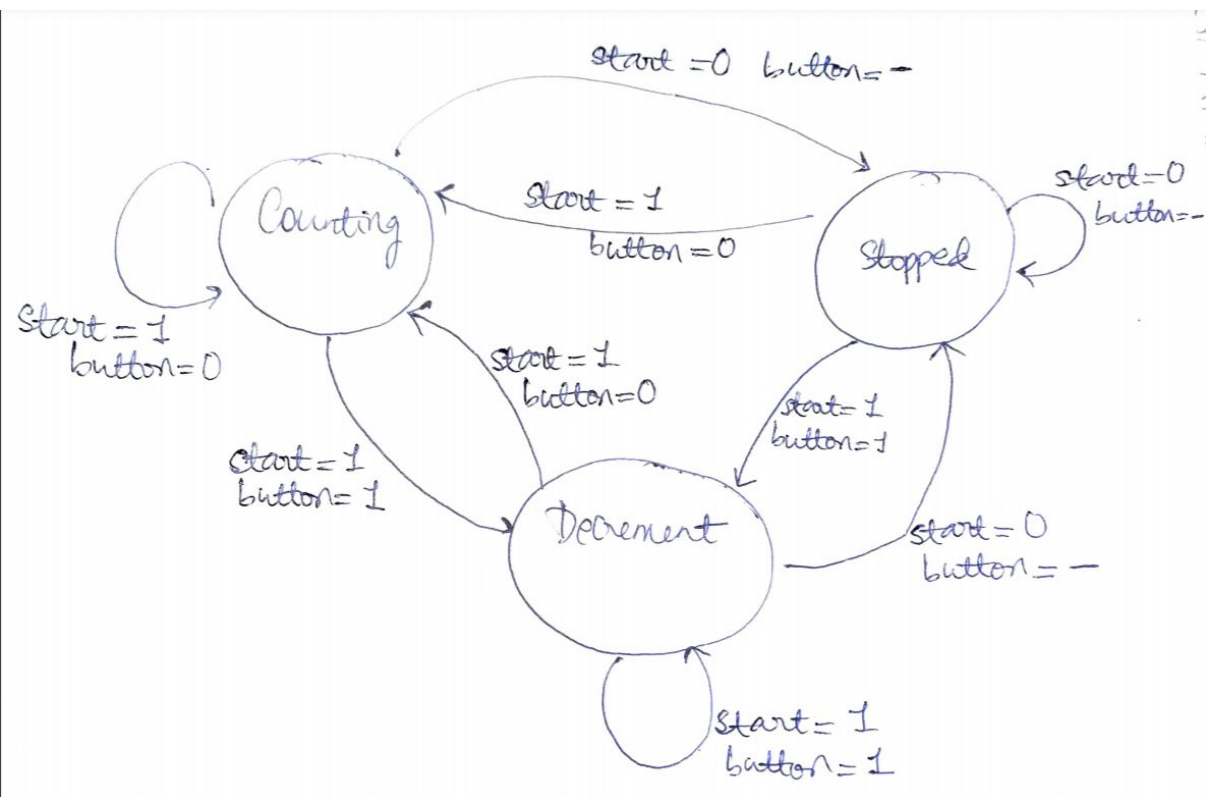
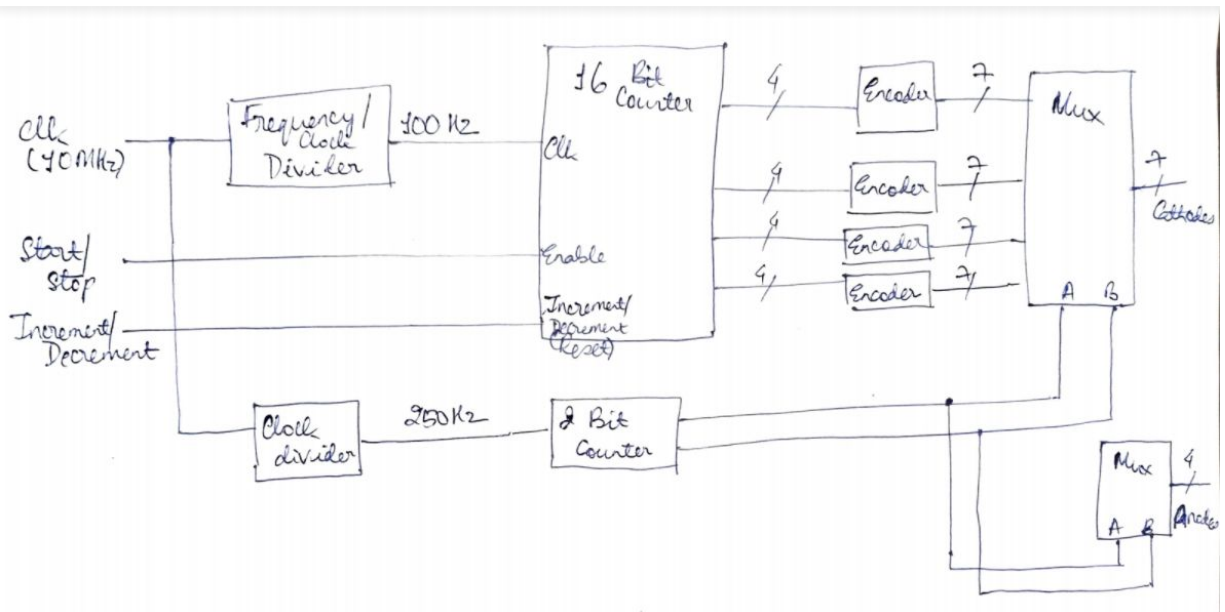
clk_1/60hz : process (clk) --clock divider for 1/60Hz
variable count4 : bit_vector(24 downto 0):=
"000000000000000000000000";
begin
    if (rising_edge(clk)) then
        if count4 = "1110010011100001110000000" then --upto
30000000
            clk4 <= not clk4;
            count4 := "000000000000000000000000" ;
        end if;
        count4 := count4 + 1;
    end if;
end process

clk_1/60hz <= clk4
clk_1hz <=clk3
clk_250hz<=clk2

end Behavioral;

```

CIRCUIT DESIGN (This is designed for calculating seconds and hundredths of seconds but this design would be repeated with a few changes for other display modes as well):



Designing 16 bit counter:

entity 16BitCounter is

Port (button: in bit;

start: in bit;

```

    select: in bit;
    rst : in bit;
    clk : in bit;
    clk_1hz : in bit;
    clk_250hz : in bit;
    clk_1/60hz: in bit;
    anode : out BIT_VECTOR (3 downto 0);
    cathode : out BIT_VECTOR (7 downto 0));
end 16BitCounter;

```

ARCHITECTURE Behavioral of 16BitCounter IS

signal a0, a1, b0, b1 : bit_vector(3 downto 0):= "0000"; *--for the 4 leds on the seven-segment display*

signal PS, NS : BIT_VECTOR (1 downto 0):="00"; *--present state and next state*

signal ss1, ss2, ens, enb, bt1, bt2 : bit:='0';

begin

 process(clk)

 begin

 if (rising_edge(clk)) then

 PS <= NS;

 end if;

 end process;

 process(clk, select) *--For selecting the display mode*

 begin

 if (rising_edge(clk)) then

 if select=1 *--Hours and Minutes will be displayed*

 process (button, rst, clk_1/60Hz, PS, NS, ss1, ss2,start,bt1,bt2)

 begin

 if rst = '1' then *--if reset is "high" then the clock will display*

all zeros

 a0 <= "0000";

 a1 <= "0000";

 b0 <= "0000";

 b1 <= "0000";

```

else
  if (rising_edge(clk_1/60Hz)) then
    if start = '1' then --to detect "risingedge" for the start
      ss1 <= '1';
    else if button = '0' then
      ss1 <= '0';
    end if;
    ss2 <= ss1;
    if ss2 = '0' and ss1 = '1' then
      ens <= not ens;
    end if;

    if button = '1' then --to detect "risingedge" for the button
      bt1 <= '1';
    else if button = '0' then
      bt1 <= '0';
    end if;
    bt2 <= bt1;
    if bt2 = '0' and bt1 = '1' then
      enb <= not enb;
    end if;
  end if;

```

```

case (PS) is
  when "11" => -- the clock is running
    if ens = '1' then
      if enb = '0' then
        NS <= "11";
        b1 <= b1 + 1; --code to have the stopwatch count
        if b1 = "1001" then
          b0 <= b0 + 1;
          b1 <= "0000";
          if b0 = "0110" then
            a1 <= a1 + 1;
            b0 <= "0000";
            if a1 = "1001" then
              a0 <= a0 + 1;
              a1 <= "0000";
            end if;
          end if;
        end if;
      end if;
    end if;
  end case;

```

if a0 = "0010" and a1= "0100" then --rolls over when it
counts upto 24:00

```
        b1 <= 0;
        b0 <= 0;
        a1 <= 0;
        a0 <= 0;
    end if;
end if;
end if;
else    --enb= '1'
    NS <= "01";    --counting down
else if ens = '0' then
    NS <= "00";
end if;
when "00" => --when the stopwatch is stopped
if ens = '0' then
    NS <= "00";
    b1 <= b1;
    b0 <= b0;
    a1 <= a1;
    a0 <= a0;
else if ens = '1' then
    if enb= '1' then
        NS <= "11";
    else
        NS <= "01";
    end if;
when "01"=> --when it should decrement one minute with every click
    if ens= '0' then
        NS<= "00";
    end if;
else if ens= '1' then
    if enb= '1' then
        NS<= "01"
        b1 <= b1 - 1; --code to have the stopwatch decrement
        if b1 = "0000" then
            b0 <= b0 - 1;
```



```

b1 <= "1001";
if b0 = "0000" then
    a1 <= a1 - 1;
    b0 <= "0101";
    if a1 = "0000" then
        a0 <= a0 - 1;
        a1 <= "1001";
        if a0 = "0000" then --rolls over when it counts upto

```

00:00

```

        b1 <= 0;
        b0 <= 0;
        a1 <= 0;
        a0 <= 0;
    end if;
end if;
end if;

end if;
else if enb= '0' then
    NS<= "11";
end if;

```

```

when others => --should never take place
    NS <= "00";
    a <= 0;
    b <= 0;
    x <= 0;
    y <= 0;
end case;
end if;
end if;
end process;

```

else **--when select is '0'; Minutes and seconds will be displayed**
 process (button, rst, clk_1Hz, PS, NS, ss1, ss2,start,bt1,bt2)

```

begin
    if rst = '1' then --if reset is "high" then the clock will display
all zeros
        a0 <= "0000";
        a1 <= "0000";
        b0 <= "0000";
        b1 <= "0000";
    else
        if (rising_edge(clk_1Hz)) then
            if start = '1' then --to detect "risingedge" for the start
                ss1 <= '1';
            else if button = '0' then
                ss1 <= '0';
            end if;
            ss2 <= ss1;
            if ss2 = '0' and ss1 = '1' then
                ens <= not ens;
            end if;

            if button = '1' then --to detect "risingedge" for the button
                bt1 <= '1';
            else if button = '0' then
                bt1 <= '0';
            end if;
            bt2 <= bt1;
            if bt2 = '0' and bt1 = '1' then
                enb <= not enb;
            end if;

        case (PS) is
            when "11" => -- the clock is running
                if ens = '1' then
                    if enb = '0' then
                        NS <= "11";
                        b1 <= b1 + 1; --code to have the stopwatch count
                        if b1 = "1001" then
                            b0 <= b0 + 1;

```

```

b1 <= "0000";
if b0 = "0110" then
    a1 <= a1 + 1;
    b0 <= "0000";
    if a1 = "1001" then
        a0 <= a0 + 1;
        a1 <= "0000";
        if a0 = "0110" then --rolls over when it counts upto

```

60:00

```

        b1 <= 0;
        b0 <= 0;
        a1 <= 0;
        a0 <= 0;
    end if;
end if;
end if;
else    --enb= '1'
    NS <= "01";    --counting down
else if ens = '0' then
    NS <= "00";
end if;

```

when "00" => --when the stopwatch is stopped

```

if ens = '0' then
    NS <= "00";
    b1 <= b1;
    b0 <= b0;
    a1 <= a1;
    a0 <= a0;
else if ens = '1' then
    if enb= '1' then
        NS <= "11";
    else
        NS <= "01";
    end if;
end if;

```

when "01"=> --when it should decrement one second with every click

```

if ens= '0' then

```

```

    NS<= "00";
end if;
else if ens= '1' then
    if enb= '1' then
        NS<= "01"
        b1 <= b1 - 1; --code to have the stopwatch decrement
        if b1 = "0000" then
            b0 <= b0 - 1;
            b1 <= "1001";
            if b0 = "0000" then
                a1 <= a1 - 1;
                b0 <= "0101";
                if a1 = "0000" then
                    a0 <= a0 - 1;
                    a1 <= "1001";
                    if a0 = "0000" then --rolls over when it counts upto
00:00
                        b1 <= 0;
                        b0 <= 0;
                        a1 <= 0;
                        a0 <= 0;
                    end if;
                end if;
            end if;
        end if;
    end if;
else if enb= '0' then
    NS<= "11";
end if;

when others => --should never take place
    NS <= "00";
    a <= 0;
    b <= 0;
    x <= 0;

```

```
    y <= 0;  
end case;
```

```
end if;
```

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

```
LED : process (clk_250hz) --determines when to turn on leds
```

```
variable dig1, dig2 : bit_vector (1 downto 0):="00";
```

```
begin
```

```
if (rising_edge(clk_250hz)) then
```

```
    case (dig1) is
```

```
        when "00" =>
```

```
            anode <= "0111";
```

```
        when "01" =>
```

```
            anode <= "1011";
```

```
        when "10" =>
```

```
            anode <= "1101";
```

```
        when "11" =>
```

```
            anode <= "1110";
```

```
    end case;
```

```
    case (dig2) is
```

```
        when "00" =>
```

```
            case (a0) is
```

```
                when "0000" =>
```

```
                    cathode <= "00000011";
```

```
                when "0001" =>
```

```
                    cathode <= "10011111";
```

```
                when "0010" =>
```

```
                    cathode <= "00100101";
```

```
                when "0011" =>
```

```
                    cathode <= "00001101";
```

```
                when "0100" =>
```

```

        cathode <= "10011001";
when "0101" =>
        cathode <= "01001001";
when "0110" =>
        cathode <= "01000001";
when "0111" =>
        cathode <= "00011111";
when "1000" =>
        cathode <= "00000001";
when "1001" =>
        cathode <= "00011001";
when others =>
        cathode <= "11111111";
end case;
when "01" =>
case (a1) is
when "0000" =>
        cathode <= "00000011";
when "0001" =>
        cathode <= "10011111";
when "0010" =>
        cathode <= "00100101";
when "0011" =>
        cathode <= "00001101";
when "0100" =>
        cathode <= "10011001";
when "0101" =>
        cathode <= "01001001";
when "0110" =>
        cathode <= "01000001";
when "0111" =>
        cathode <= "00011111";
when "1000" =>
        cathode <= "00000001";
when "1001" =>
        cathode <= "00011001";
when others =>

```

```

                                cathode <= "11111111";
        end case;
when "10" =>
    case (b0) is
        when "0000" =>
            cathode <= "00000011";
        when "0001" =>
            cathode <= "10011111";
        when "0010" =>
            cathode <= "00100101";
        when "0011" =>
            cathode <= "00001101";
        when "0100" =>
            cathode <= "10011001";
        when "0101" =>
            cathode <= "01001001";
        when "0110" =>
            cathode <= "01000001";
        when "0111" =>
            cathode <= "00011111";
        when "1000" =>
            cathode <= "00000001";
        when "1001" =>
            cathode <= "00011001";
        when others =>
            cathode <= "11111111";
    end case;
when "11"=>
    case (b1) is
        when "0000" =>
            cathode <= "00000011";
        when "0001" =>
            cathode <= "10011111";
        when "0010" =>
            cathode <= "00100101";
        when "0011" =>
            cathode <= "00001101";
    end case;

```

```

        when "0100" =>
            cathode <= "10011001";
        when "0101" =>
            cathode <= "01001001";
        when "0110" =>
            cathode <= "01000001";
        when "0111" =>
            cathode <= "00011111";
        when "1000" =>
            cathode <= "00000001";
        when "1001" =>
            cathode <= "00011001";
        when others =>
            cathode <= "11111111";
    end case;
end case;
dig1 := dig1 + 1;
dig2 := dig2 + 1;
end if;
end process;
end Behavioral;

```

STRINGING TOGETHER ALL THE COMPONENTS:

entity Main_Code is

Port (button : in bit; *-- For decrementing; the standard mode is counting: button=0 => Counting; button=1 => Decrement by 1*

 select:in bit; *--for choosing display mode*

 start:in bit; *--start/stop*

 clk : in bit;

 rst : in bit; *--reset*

 anode : out BIT_VECTOR (3 downto 0);

 cathode : out BIT_VECTOR (7 downto 0));

end Main_Code;

architecture Behavioral of Main_Code is

component clock_dividers


```
Port ( clk: in bit;  
      clk_250hz : out bit;  
      clk_1/60hz: out bit;  
      clk_1hz : out bit);  
end component;
```

```
component 16BitCounter  
  Port (button: in bit;  
        start: in bit;  
        select: in bit;  
        rst : in bit;  
        clk : in bit;  
        clk_1hz : in bit;  
        clk_250hz : in bit;  
        clk_1/60hz: in bit;  
        anode : out BIT_VECTOR (3 downto 0);  
        cathode : out BIT_VECTOR (7 downto 0) );  
end component;
```

```
signal clk2, clk3, clk4 : bit;  
begin  
  comp1 : clock_dividers port map (clk =>clk, clk_250hz => clk2, clk_1hz  
=> clk3, clk_1/60hz=>clk4);  
  comp2 : 16BitCounter port map (clk => clk, clk_250hz => clk2, clk_1hz  
=> clk3, clk_1/60hz=>clk4, button =>  
button, rst => rst, select=>select, start=>start, anode => anode, cathode  
=> cathode);  
end Behavioral;
```

