EGR 234 – Digital Logic Design

Lab 10:

Digital Clock Using Counters and VHDL

Lab Report by: Christopher Parisi

Lab Partner: No Partner

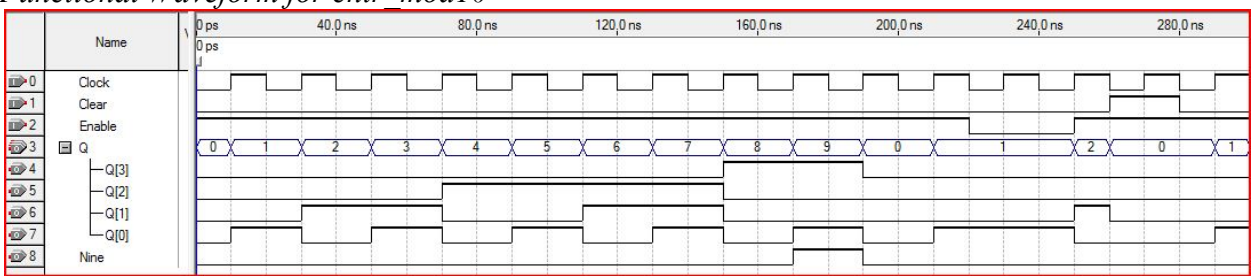
December 10, 2010

Exercise 1

VHDL Code for cntr_mod10

```
library ieee ;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

--------------------------------------------------

entity cntr_mod10 is
port( Clear: in std_logic;
      Clock: in std_logic;
     Enable: in std_logic;
                  Q: out std_logic_vector(3 downto 0);
       Nine: out std_logic );
end entity cntr_mod10;

--------------------------------------------------

architecture cntr_mod10_arch of cntr_mod10 is

          component cntr_reg is
          port(    Clear: in std_logic;
                            Clock: in std_logic;
                   Enable: in std_logic;
                    Load: in std_logic;
                       D: in std_logic_vector(3 downto 0);
                       Q: out std_logic_vector(3 downto 0) );
          end component cntr_reg;

  signal Load: std_logic;
  signal Q_tmp: std_logic_vector(3 downto 0);

begin

   Load <= Q_tmp(3) and Q_tmp(0);
   cntr_reg1: cntr_reg
          port map (  Clear => Clear, Clock => Clock,
              Enable => Enable, Load => Load, D => "0000", Q => Q_tmp );

   Q <= Q_tmp;
         Nine <= Load;

end architecture cntr_mod10_arch;
```

Functional Waveform for cntr_mod10



Exercise 2

VHDL Code for one_sec_pulse.

```
library ieee ;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

-------------------------------------------------

entity one_sec_pulse is
port( Clock: in std_logic;
      Clear: in std_logic;
      Pulse: out std_logic );
end entity one_sec_pulse;

-------------------------------------------------

architecture one_sec_pulse_arch of one_sec_pulse is

          component cntr_mod10 is
          port( Clear: in std_logic;
                    Clock: in std_logic;
        Enable: in std_logic;
            Q:  out std_logic_vector(3 downto 0);
         Nine: out std_logic );
          end component cntr_mod10;

          component cntr_mod5 is
          port( Clear: in std_logic;
         Clock: in std_logic;
        Enable: in std_logic;
            Q: out std_logic_vector(3 downto 0);
          Four: out std_logic );
    end component cntr_mod5;
```

```
  signal Four8, Nine7, Nine6, Nine5, Nine4, Nine3, Nine2, Nine1: std_logic;
  signal Enable8, Enable7, Enable6, Enable5, Enable4, Enable3, Enable2: std_logic;
  signal Q_tmp8, Q_tmp7, Q_tmp6, Q_tmp5, Q_tmp4, Q_tmp3, Q_tmp2, Q_tmp1:
std_logic_vector(3 downto 0);

begin

   cntr_mod10_1: cntr_mod10
        port map ( Clear => Clear, Clock => Clock, Enable => '1',
                Q => Q_tmp1, Nine => Nine1 );

        Enable2 <= Nine1;

        cntr_mod10_2: cntr_mod10
        port map ( Clear => Clear, Clock => Clock, Enable => Enable2,
                Q => Q_tmp2, Nine => Nine2 );

        Enable3 <= Nine2 and Nine1;

        cntr_mod10_3: cntr_mod10
        port map ( Clear => Clear, Clock => Clock, Enable => Enable3,
                Q => Q_tmp3, Nine => Nine3 );

        Enable4 <= Nine3 and Nine2 and Nine1;

        cntr_mod10_4: cntr_mod10
        port map ( Clear => Clear, Clock => Clock, Enable => Enable4,
                Q => Q_tmp4, Nine => Nine4 );

        Enable5 <= Nine4 and Nine3 and Nine2 and Nine1;

        cntr_mod10_5: cntr_mod10
        port map ( Clear => Clear, Clock => Clock, Enable => Enable5,
                Q => Q_tmp5, Nine => Nine5 );

        Enable6 <= Nine5 and Nine4 and Nine3 and Nine2 and Nine1;


        cntr_mod10_6: cntr_mod10
        port map ( Clear => Clear, Clock => Clock, Enable => Enable6,
                Q => Q_tmp6, Nine => Nine6 );

        Enable7 <= Nine6 and Nine5 and Nine4 and Nine3 and Nine2 and Nine1;

   cntr_mod10_7: cntr_mod10
        port map ( Clear => Clear, Clock => Clock, Enable => Enable7,
```

```
                Q => Q_tmp7, Nine => Nine7 );

        Enable8 <= Nine7 and Nine6 and Nine5 and Nine4 and Nine3 and Nine2 and Nine1;

   cntr_mod5_8: cntr_mod5
         port map ( Clear => Clear, Clock => Clock, Enable => Enable8,
                Q => Q_tmp8, Four => Four8 );


   Pulse <= Four8 and Nine7 and Nine6 and Nine5 and Nine4 and Nine3 and Nine2 and Nine1;

end architecture one_sec_pulse_arch;
```

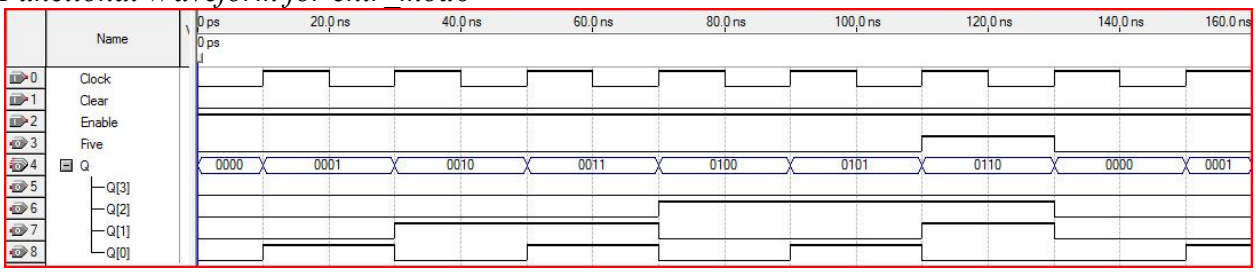
Exercise 3

VHDL Code for cntr_mod6

```
library ieee ;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

--------------------------------------------------

entity cntr_mod6 is
port( Clear: in std_logic;
     Clock: in std_logic;
    Enable: in std_logic;
                 Q: out std_logic_vector(3 downto 0);
      Five: out std_logic );
end entity cntr_mod6;

--------------------------------------------------

architecture cntr_mod6_arch of cntr_mod6 is

          component cntr_reg is
          port(    Clear: in std_logic;
                        Clock: in std_logic;
         Enable: in std_logic;
           Load: in std_logic;
              D: in std_logic_vector(3 downto 0);
              Q: out std_logic_vector(3 downto 0) );
          end component cntr_reg;

  signal Load: std_logic;
```

```
  signal Q_tmp: std_logic_vector(3 downto 0);

begin

   Load <= Q_tmp(2) and Q_tmp(1);
   cntr_reg1: cntr_reg
         port map (  Clear => Clear, Clock => Clock,
             Enable => Enable, Load => Load, D => "0000", Q => Q_tmp );

   Q <= Q_tmp;
        Five <= Load;

end architecture cntr_mod6_arch;
```

Functional Waveform for cntr_mod6



VHDL Code for Digital Clock

```
library ieee ;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

--------------------------------------------------

entity Digital_Clock is
port( Clock: in std_logic;
     Clear: in std_logic;
     SecondOut0: out std_logic_vector(6 downto 0);
     SecondOut1: out std_logic_vector(6 downto 0);
     MinuteOut0: out std_logic_vector(6 downto 0);
     MinuteOut1: out std_logic_vector(6 downto 0) );
end entity Digital_Clock;

--------------------------------------------------

architecture Digital_Clock_arch of Digital_Clock is

          component cntr_mod10 is
          port( Clear: in std_logic;
```

```
                Clock: in std_logic;
        Enable: in std_logic;
            Q: out std_logic_vector(3 downto 0);
          Nine: out std_logic );
          end component cntr_mod10;

          component cntr_mod6 is
          port( Clear: in std_logic;
         Clock: in std_logic;
        Enable: in std_logic;
            Q: out std_logic_vector(3 downto 0);
         Five: out std_logic );
    end component cntr_mod6;

          component one_sec_pulse is
          port( Clear: in std_logic;
         Clock: in std_logic;
        Pulse: out std_logic );
    end component one_sec_pulse;

    component hex7seg is
          port (hex : in std_logic_vector(3 downto 0);
                  seg : out std_logic_vector(6 downto 0));
    end component hex7seg;

   signal Pulse0, Five4, Nine3, Five2, Nine1: std_logic;
   signal Enable4, Enable3, Enable2, Enable1: std_logic;
   signal Q_tmp4, Q_tmp3, Q_tmp2, Q_tmp1: std_logic_vector(3 downto 0);
   signal Second0, Second1, Minute0, Minute1: std_logic_vector(6 downto 0);

begin

          One_sec_pulse_0: one_sec_pulse
                          port map ( Clear => Clear, Clock => Clock, Pulse => Pulse0 );

          Enable1 <= Pulse0;

    cntr_mod10_1: cntr_mod10
          port map ( Clear => Clear, Clock => Clock, Enable => Enable1,
                  Q => Q_tmp1, Nine => Nine1 );

          Enable2 <= Nine1;

    cntr_mod6_2: cntr_mod6
          port map ( Clear => Clear, Clock => Clock, Enable => Enable2,
                  Q => Q_tmp2, Five => Five2);
```

```
    Enable3 <= Nine1 and Five2;

    cntr_mod10_3: cntr_mod10
          port map ( Clear => Clear, Clock => Clock, Enable => Enable3,
                Q => Q_tmp3, Nine => Nine3 );

          Enable4 <= Nine1 and Five2 and Nine3;

    cntr_mod6_4: cntr_mod6
          port map ( Clear => Clear, Clock => Clock, Enable => Enable4,
                Q => Q_tmp4, Five => Five4 );

    hex7seg_5: hex7seg
                      port map (hex => Q_tmp1, seg => Second0);

    hex7seg_6: hex7seg
                      port map (hex => Q_tmp2, seg => Second1);

          hex7seg_7: hex7seg
                      port map (hex => Q_tmp3, seg => Minute0);

          hex7seg_8: hex7seg
                      port map (hex => Q_tmp4, seg => Minute1);

          SecondOut0 <= Second0;
          SecondOut1 <= Second1;
          MinuteOut0 <= Minute0;
          MinuteOut1 <= Minute1;

end architecture Digital_Clock_arch;
```