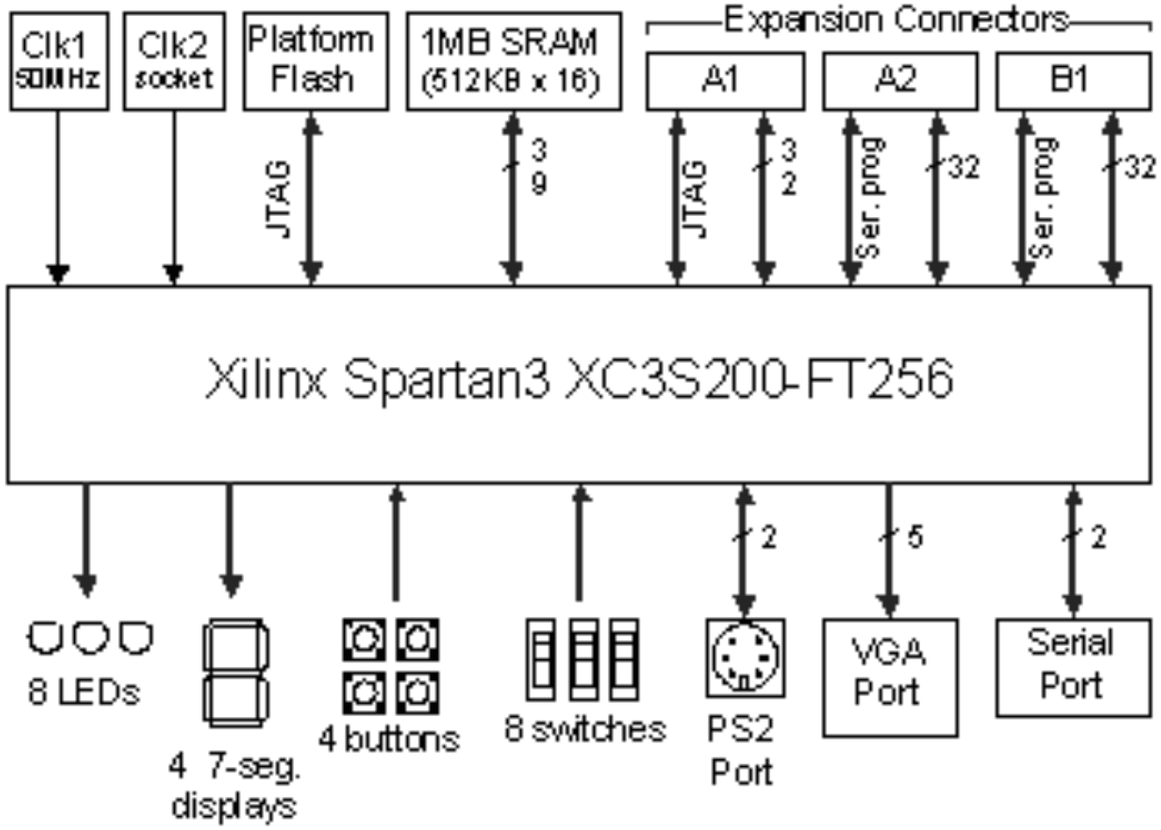
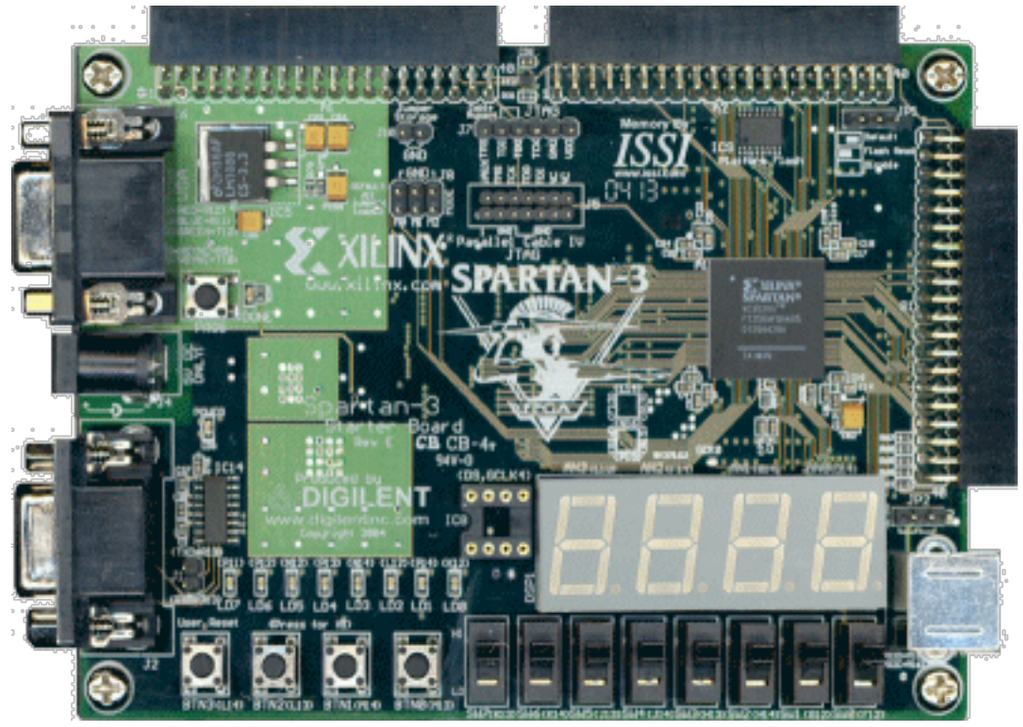
CPE 462
VHDL: Simulation and Synthesis

Topic #06 - b) Running sequential code in a FPGA board

Spartan-3 FPGA board

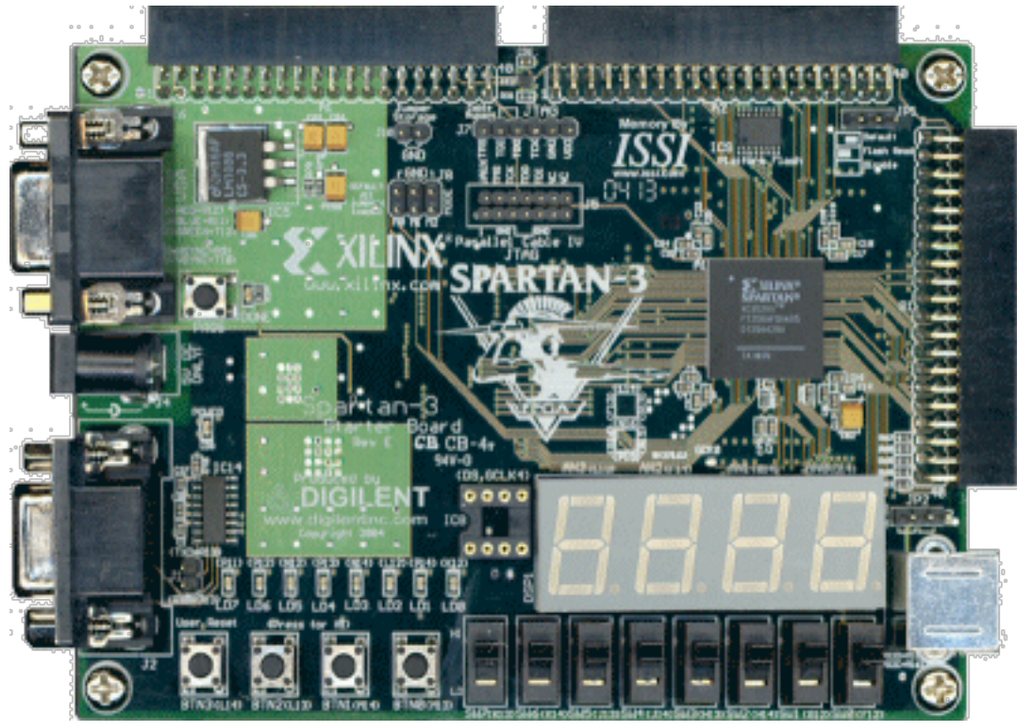




http://www.xilinx.com/support/documentation/boards_and_kits/ug130.pdf

Spartan-3 FPGA board

- In a "standard" circuits, everything is clocked.
- Our FPGA board has a CLOCK of 50MHz.
- The clock pin is "**T9**"
- 50MHz means, that the period is 1/(50*10^6)
- Frequency = 1 / Period



Make sure the board settings are correct

If your clock is "acting funny", you probably have the incorrect FPGA board settings!

So make sure you add the correct board in your project:

- **Family:** Spartan-3
- **Device:** xc3s200
- **Package:** ft256
- **Speed Grade:** -4

Using the Spartan3 clock

Nuno Alves (nalves@wne.edu), College of Engineering

How do we actually slow down the clock?

- For example, we have a 50MHz clock and we want a slow clock of 15MHz
- Here are the steps:

(1) Find the period of the 50MHz clock => 20ns

(2) Find the period of the 12.5MHz clock => 80ns

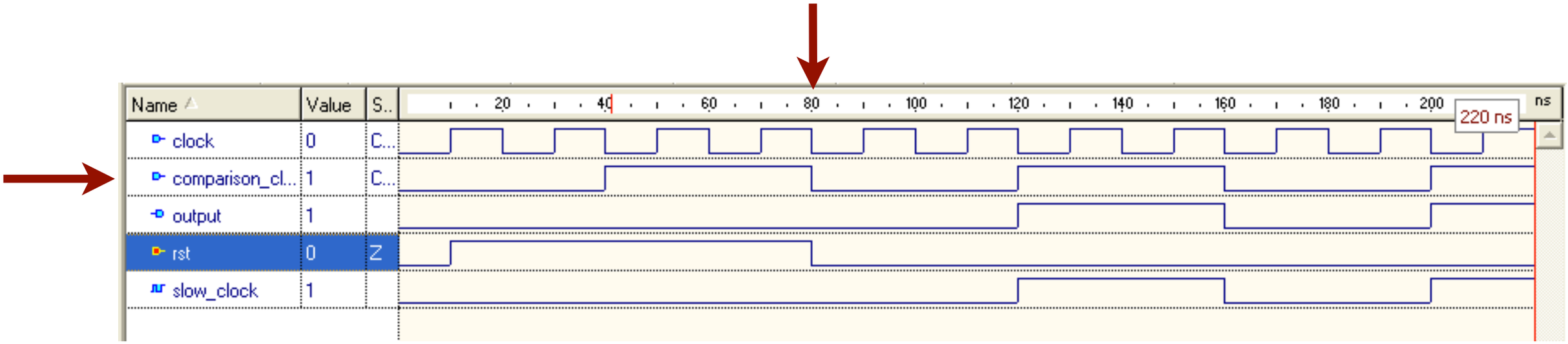
(3) Divide period of slowest clock with fastest clock => 80/20 = 4

(4) Since a clock period contains clock high and clock low, and we are only counting rising edges, we divide step (3) by 2. => 4/2 = 2

- This code template will create a slower clock.
- It will take a 50MHz clock and it will create a slow clock of 12.5MHz
- All you have to do is replace the **divider** constant with the appropriate value (step 4 of previous slide)

```
library IEEE; use IEEE.STD_LOGIC_1164.ALL;

entity main is
    Port ( clock, rst : in std_logic;
           output : out std_logic);
end main;

architecture Behavioral of main is
   signal slow_clock : std_logic := '0';
   constant divider : integer := 2;
begin
     process (clock)
     variable counter : integer;
     begin
        if (clock'event and clock='0') then
             counter := counter + 1;
                  if (counter=divider) then
                       counter := 0;
                       slow_clock <= not (slow_clock);
                  end if;

        end if;

        if (rst='1') then counter:=0; end if;
      end process;

   output<=slow_clock;

end Behavioral;
```

Verification of the template code



- I created a **comparison_clock** signal with 12.5MHz in Active HDL
- The **clock** signal has a frequency of 50 MHz
- The signal **rst** is high. At 80ns it goes low so we can start counting.
- The **slow_clock** is identical to our **comparison_clock.**

Explaining the clock divider in detail

- This process will run every-time the clock changes

- We don’t really know what value the variable counter has.

```
library IEEE; use IEEE.STD_LOGIC_1164.ALL;

entity main is
    Port ( clock, rst : in std_logic;
           output : out std_logic);
end main;

architecture Behavioral of main is
   signal slow_clock : std_logic := '0';
   constant divider : integer := 2;
begin
   process (clock)
   variable counter : integer;
   begin
      if (clock'event and clock='0') then
           counter := counter + 1;
                if (counter=divider) then
                     counter := 0;
                     slow_clock <= not (slow_clock);
                end if;

      end if;

      if (rst='1') then counter:=0; end if;
   end process;

   output<=slow_clock;

end Behavioral;
```

- In the SPARTAN3 board we can't have more than one 'clock event!
- Every time the clock changes and clock is 0, it will increment the counter
- If the counter equals the divider, then we have reached the desired frequency

```
library IEEE; use IEEE.STD_LOGIC_1164.ALL;

entity main is
    Port ( clock, rst : in std_logic;
           output : out std_logic);
end main;

architecture Behavioral of main is
   signal slow_clock : std_logic := '0';
   constant divider : integer := 2;
begin
     process (clock)
     variable counter : integer;
     begin
     if (clock'event and clock='0') then
          counter := counter + 1;
               if (counter=divider) then
                    counter := 0;
                    slow_clock <= not (slow_clock);
               end if;

     end if;

     if (rst='1') then counter:=0; end if;
   end process;

  output<=slow_clock;

end Behavioral;
```

- This line pretty much ensures the counter variable is initialized whenever the reset switch is flipped.

```
library IEEE; use IEEE.STD_LOGIC_1164.ALL;

entity main is
    Port ( clock, rst : in std_logic;
           output : out std_logic);
end main;

architecture Behavioral of main is
   signal slow_clock : std_logic := '0';
   constant divider : integer := 2;
begin
     process (clock)
     variable counter : integer;
     begin
        if (clock'event and clock='0') then
             counter := counter + 1;
             if (counter=divider) then
                  counter := 0;
                  slow_clock <= not (slow_clock);
             end if;

        end if;

     if (rst='1') then counter:=0; end if;
     end process;

   output<=slow_clock;

end Behavioral;
```

Running the clock divider in hardware

- By setting up 25000000 as the divider constant we are going to create a slow_clock with a frequency of 1Hz
- Frequency of 1Hz corresponds to a period of 1 second
- This means, the clock will alternate between high and low every 0.5 seconds

```
library IEEE; use IEEE.STD_LOGIC_1164.ALL;

entity main is
    Port ( clock, rst : in std_logic;
           output : out std_logic);
end main;

architecture Behavioral of main is
   signal slow_clock : std_logic := '0';
   constant divider : integer := 25000000;
begin
     process (clock)
     variable counter : integer;
     begin
        if (clock'event and clock='0') then
             counter := counter + 1;
                  if (counter=divider) then
                       counter := 0;
                       slow_clock <= not (slow_clock);
                  end if;

        end if;

        if (rst='1') then counter:=0; end if;
      end process;

    output<=slow_clock;

end Behavioral;
```

- Map the pins to the following signals:

```
NET "clock" LOC = "t9";

NET "output" LOC = "k12";

NET "rst" LOC = "k13";
```

- The **output** is now LED0 and the **rst** is switch 7

- The code will turn **ON** the LED for 0.5 seconds and turn it **OFF** for 0.5 seconds.

```
library IEEE; use IEEE.STD_LOGIC_1164.ALL;

entity main is
    Port ( clock, rst : in std_logic;
           output : out std_logic);
end main;

architecture Behavioral of main is
   signal slow_clock : std_logic := '0';
   constant divider : integer := 25000000;
begin
     process (clock)
     variable counter : integer;
     begin
        if (clock'event and clock='0') then
           counter := counter + 1;
                if (counter=divider) then
                   counter := 0;
                   slow_clock <= not (slow_clock);
                end if;

        end if;

        if (rst='1') then counter:=0; end if;
      end process;

    output<=slow_clock;

end Behavioral;
```

Debugging your code in active HDL

- If you want to do something at the beat of 1Hz, you need to create a new clock with the appropriate divider.
- However you don't want to wait 25000000 clock cycles. Just modify this to something you can visualize on a wave

```
library IEEE; use IEEE.STD_LOGIC_1164.ALL;

entity main is
    Port ( clock, rst : in std_logic;
           output : out std_logic);
end main;

architecture Behavioral of main is
   signal slow_clock : std_logic := '0';
   constant divider : integer := 25000000;
begin
     process (clock)
     variable counter : integer;
     begin
        if (clock'event and clock='0') then
             counter := counter + 1;
             if (counter=divider) then
                  counter := 0;
                  slow_clock <= not (slow_clock);
             end if;

        end if;

        if (rst='1') then counter:=0; end if;
      end process;

    output<=slow_clock;

end Behavioral;
```

- Create a new process with this new slow_clock and add some code in there

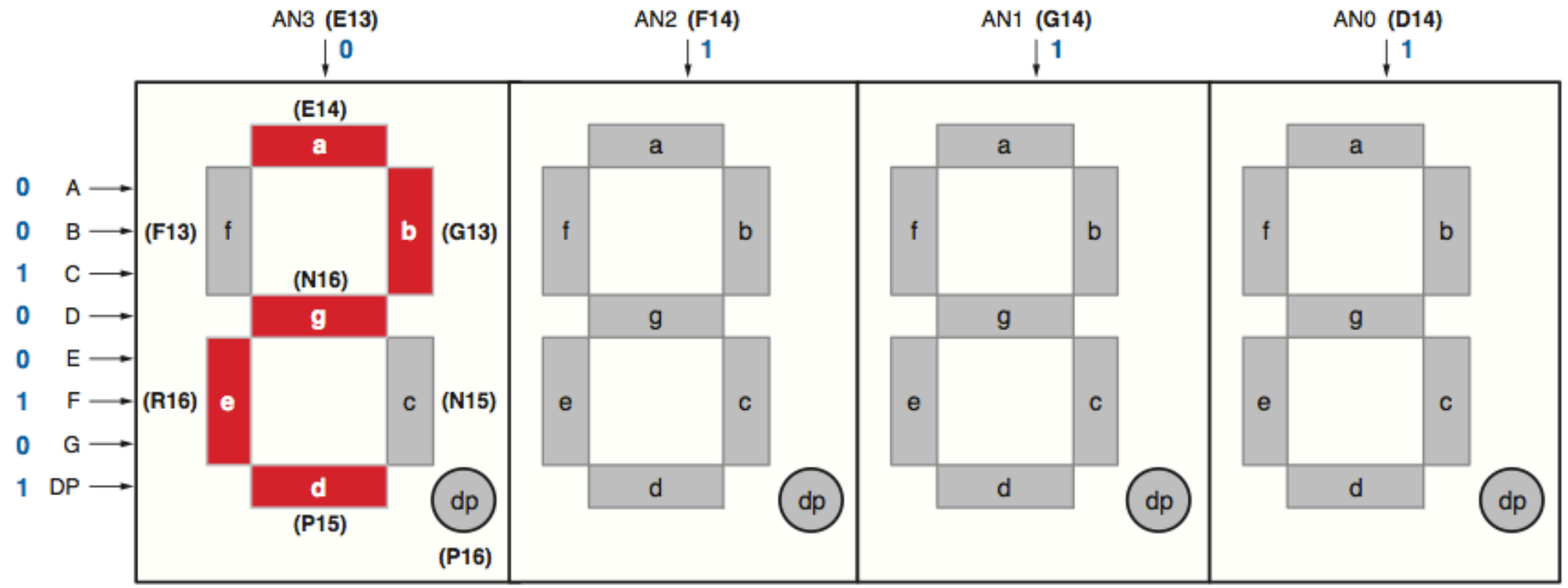
```
library IEEE; use IEEE.STD_LOGIC_1164.ALL;
entity main is
    Port ( clock, rst : in std_logic;
           output : out std_logic);
end main;

architecture Behavioral of main is
   signal slow_clock : std_logic := '0';
   constant divider : integer := 25000000;
begin
     process (clock)
     variable counter : integer;
     begin
        if (clock'event and clock='0') then
             counter := counter + 1;
             if (counter=divider) then
                  counter := 0;
                  slow_clock <= not (slow_clock);
             end if;

        end if;
        if (rst='1') then counter:=0; end if;
     end process;

  process(slow_clock)
       begin
         (...) some code here (...)
       end
  output<=slow_clock;
end Behavioral;
```

7 Segment displays

Using the Spartan3 seven segment display

The seven segment display are time-multiplexed.

This means, with the same pins, we can address each of the four seven-segment displays.



Writting FFFF on all four displays

- Make sure you use the following pin mapping:

```
NET "LED<0>"  LOC = "E14"  ;

NET "LED<1>"  LOC = "G13"  ;

NET "LED<2>"  LOC = "N15"  ;

NET "LED<3>"  LOC = "P15"  ;

NET "LED<4>"  LOC = "R16"  ;

NET "LED<5>"  LOC = "F13"  ;

NET "LED<6>"  LOC = "N16"  ;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity ffff is
    Port ( LED : out std_logic_vector(6 downto 0));
end ffff;

architecture Behavioral of ffff is

begin
  -- writes the 'F' pattern to the led.
  LED <= "0001110";
end Behavioral;
```

- This code will write “Anne” on all four seven-segment displays.

```
library IEEE; use IEEE.STD_LOGIC_1164.ALL;

entity vhdlmodule is
    Port ( CLKIN : in std_logic;
           AN3 : inout std_logic;
           AN2 : inout std_logic;
           AN1 : inout std_logic;
           AN0 : inout std_logic;
           LED : out std_logic_vector(6 downto 0));
end vhdlmodule;

architecture Behavioral of vhdlmodule is
signal CTR : STD_LOGIC_VECTOR(12 downto 0);
begin
  Process (CLKIN)
  begin
    if CLKIN'event and CLKIN = '1' then
      if (CTR="0000000000000") then
           if (AN0='0') then AN0 <= '1'; LED <= "0101011"; AN1 <= '0';
           elsif (AN1='0') then AN1 <= '1'; LED <= "0101011"; AN2 <= '0';
           elsif (AN2='0') then AN2 <= '1'; LED <= "0001000"; AN3 <= '0';
           elsif (AN3='0') then AN3 <= '1'; LED <= "0000110"; AN0 <= '0';
           end if;
      end if;
      CTR<=CTR+"0000000000001";
      if (CTR > "1000000000000") then   -- counter reaches 2^13
        CTR<="0000000000000";
      end if;
    end if; -- CLK'event and CLK = '1'
  End Process;
End Behavioral;
```

- ANx will select which segment we are writing to
- The LED signal contains the character we are trying to write

```
library IEEE; use IEEE.STD_LOGIC_1164.ALL;

entity vhdlmodule is
  Port ( CLKIN : in std_logic;
         AN3 : inout std_logic;
         AN2 : inout std_logic;
         AN1 : inout std_logic;
         AN0 : inout std_logic;
         LED : out std_logic_vector(6 downto 0));
end vhdlmodule;

architecture Behavioral of vhdlmodule is
signal CTR : STD_LOGIC_VECTOR(12 downto 0);
begin
  Process (CLKIN)
  begin
    if CLKIN'event and CLKIN = '1' then
      if (CTR="0000000000000") then
           if (AN0='0') then AN0 <= '1'; LED <= "0101011"; AN1 <= '0';
           elsif (AN1='0') then AN1 <= '1'; LED <= "0101011"; AN2 <= '0';
           elsif (AN2='0') then AN2 <= '1'; LED <= "0001000"; AN3 <= '0';
           elsif (AN3='0') then AN3 <= '1'; LED <= "0000110"; AN0 <= '0';
           end if;
      end if;
      CTR<=CTR+"0000000000001";
      if (CTR > "1000000000000") then   -- counter reaches 2^13
        CTR<="0000000000000";
      end if;
    end if; -- CLK'event and CLK = '1'
  End Process;
End Behavioral;
```

- Unfortunately the clock signal is too quick to switch between digits on the led panel.
- So we will be using a counter named CTR, each time the counter reaches 2^13 we change the digit to be displayed, an we reset the counter.

```
library IEEE; use IEEE.STD_LOGIC_1164.ALL;

entity vhdlmodule is
    Port ( CLKIN : in std_logic;
           AN3 : inout std_logic;
           AN2 : inout std_logic;
           AN1 : inout std_logic;
           AN0 : inout std_logic;
           LED : out std_logic_vector(6 downto 0));
end vhdlmodule;

architecture Behavioral of vhdlmodule is
signal CTR : STD_LOGIC_VECTOR(12 downto 0);
begin
  Process (CLKIN)
  begin
    if CLKIN'event and CLKIN = '1' then
      if (CTR="0000000000000") then
           if (AN0='0') then AN0 <= '1'; LED <= "0101011"; AN1 <= '0';
           elsif (AN1='0') then AN1 <= '1'; LED <= "0101011"; AN2 <= '0';
           elsif (AN2='0') then AN2 <= '1'; LED <= "0001000"; AN3 <= '0';
           elsif (AN3='0') then AN3 <= '1'; LED <= "0000110"; AN0 <= '0';
           end if;
      end if;
      CTR<=CTR+"0000000000001";
      if (CTR > "1000000000000") then   -- counter reaches 2^13
        CTR<="0000000000000";
      end if;
    end if; -- CLK'event and CLK = '1'
  End Process;
End Behavioral;
```

- We alternatively write the characters on each display inside this section

```
library IEEE; use IEEE.STD_LOGIC_1164.ALL;

entity vhdlmodule is
    Port ( CLKIN : in std_logic;
           AN3 : inout std_logic;
           AN2 : inout std_logic;
           AN1 : inout std_logic;
           AN0 : inout std_logic;
           LED : out std_logic_vector(6 downto 0));
end vhdlmodule;

architecture Behavioral of vhdlmodule is
signal CTR : STD_LOGIC_VECTOR(12 downto 0);
begin
  Process (CLKIN)
  begin
    if CLKIN'event and CLKIN = '1' then
      if (CTR="0000000000000") then
        if (AN0='0') then AN0 <= '1'; LED <= "0101011"; AN1 <= '0';
        elsif (AN1='0') then AN1 <= '1'; LED <= "0101011"; AN2 <= '0';
        elsif (AN2='0') then AN2 <= '1'; LED <= "0001000"; AN3 <= '0';
        elsif (AN3='0') then AN3 <= '1'; LED <= "0000110"; AN0 <= '0';
        end if;
      end if;
      CTR<=CTR+"0000000000001";
      if (CTR > "1000000000000") then   -- counter reaches 2^13
        CTR<="0000000000000";
      end if;
    end if; -- CLK'event and CLK = '1'
  End Process;
End Behavioral;
```

Don't forget to use the correct pin mapping:

```
NET "AN0"  LOC = "D14"  ;

NET "AN1"  LOC = "G14"  ;

NET "AN2"  LOC = "F14"  ;

NET "AN3"  LOC = "E13"  ;

NET "CLKIN"  LOC = "T9"  ;

NET "LED<0>"  LOC = "E14"  ;

NET "LED<1>"  LOC = "G13"  ;

NET "LED<2>"  LOC = "N15"  ;

NET "LED<3>"  LOC = "P15"  ;

NET "LED<4>"  LOC = "R16"  ;

NET "LED<5>"  LOC = "F13"  ;

NET "LED<6>"  LOC = "N16"  ;
```

```
library IEEE; use IEEE.STD_LOGIC_1164.ALL;

entity vhdlmodule is
    Port ( CLKIN : in std_logic;
           AN3 : inout std_logic;
           AN2 : inout std_logic;
           AN1 : inout std_logic;
           AN0 : inout std_logic;
           LED : out std_logic_vector(6 downto 0));
end vhdlmodule;

architecture Behavioral of vhdlmodule is
signal CTR : STD_LOGIC_VECTOR(12 downto 0);
begin
  Process (CLKIN)
  begin
    if CLKIN'event and CLKIN = '1' then
      if (CTR="0000000000000") then
        if (AN0='0') then AN0 <= '1'; LED <= "0101011"; AN1 <= '0';
        elsif (AN1='0') then AN1 <= '1'; LED <= "0101011"; AN2 <= '0';
        elsif (AN2='0') then AN2 <= '1'; LED <= "0001000"; AN3 <= '0';
        elsif (AN3='0') then AN3 <= '1'; LED <= "0000110"; AN0 <= '0';
        end if;
      end if;
      CTR<=CTR+"0000000000001";
      if (CTR > "1000000000000") then   -- counter reaches 2^13
        CTR<="0000000000000";
      end if;
    end if; -- CLK'event and CLK = '1'
  End Process;
End Behavioral;
```