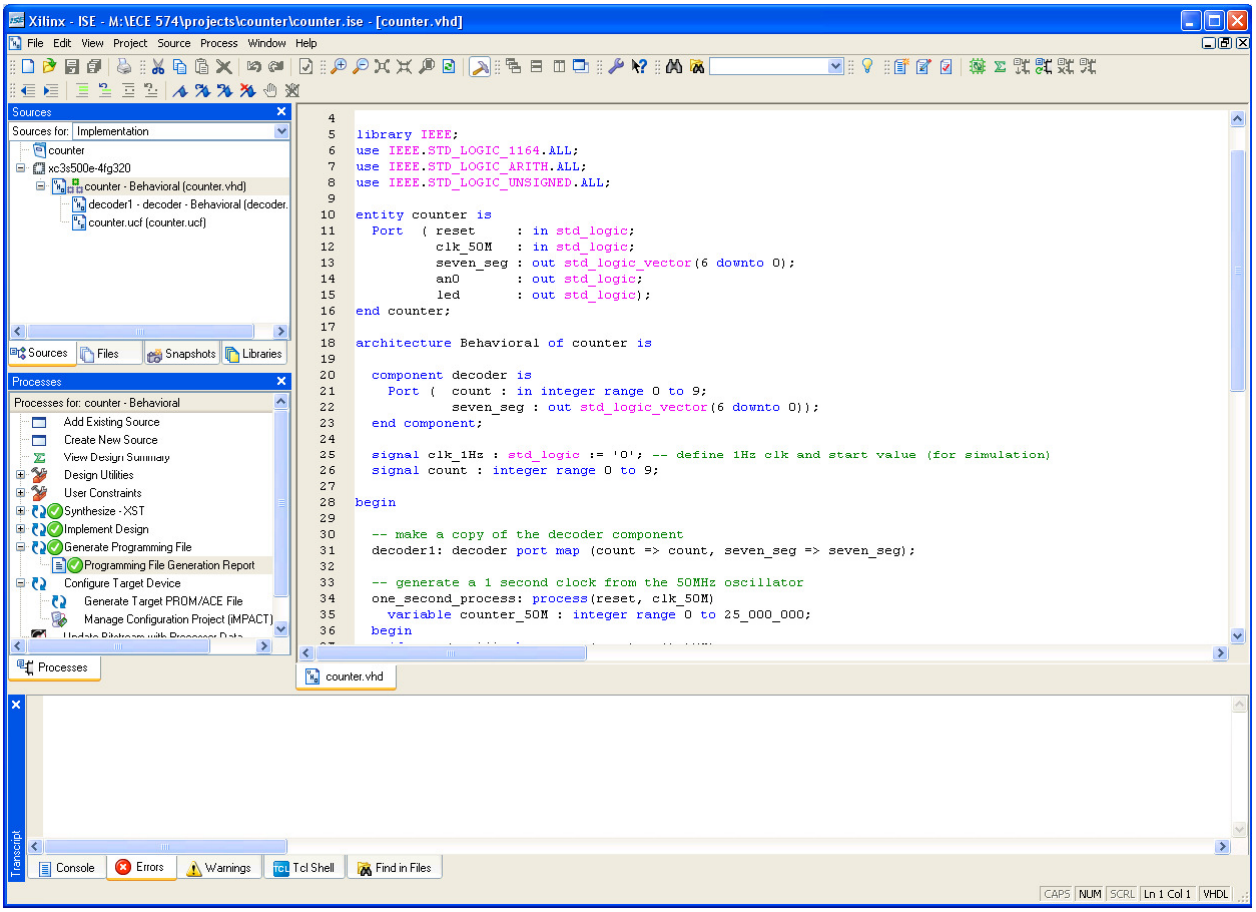
Nexys 2 board tutorial

(Counter with decoder, ISE 10.1)

Jim Duckworth, January 2005, WPI.
(updated by Jorge Alejandro, September 2008)

This design shows how to create a simple sequential circuit (a counter).
It also demonstrates hierarchical design by using a separate decoder component that converts a binary count value to a seven segment display.

There are three files for this design as shown in the *Sources in Project* Window:



Top level entity and architecture description:

```
--  Jim Duckworth
--  ECE Dept, WPI
--  January 6, 2005
--  counter.vhd

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity counter is
  Port  ( reset     : in std_logic;
          clk_50M   : in std_logic;
          seven_seg : out std_logic_vector(6 downto 0);
          an0       : out std_logic;
          led       : out std_logic);
end counter;

architecture Behavioral of counter is

  component decoder is
    Port (  count : in integer range 0 to 9;
            seven_seg : out std_logic_vector(6 downto 0));
  end component;

  signal clk_1Hz : std_logic := '0'; -- define 1Hz clk and start value (for simulation)
  signal count : integer range 0 to 9;

begin

  -- make a copy of the decoder component
  decoder1: decoder port map (count => count, seven_seg => seven_seg);

  -- generate a 1 second clock from the 50MHz oscillator
  one_second_process: process(reset, clk_50M)
    variable counter_50M : integer range 0 to 25_000_000;
  begin
    if reset = '1' then
      counter_50M := 0;
    elsif rising_edge(clk_50M) then
      counter_50M := counter_50M + 1;
      if counter_50M = 25_000_000 then
        clk_1Hz <= NOT clk_1Hz;
        counter_50M := 0;
      end if;
    end if;
  end process one_second_process;

  -- count seconds from 0 to 9
  count_process: process(reset, clk_1Hz)
  begin
    if reset = '1' then
      count <= 0;
    elsif rising_edge(clk_1Hz) then
      if count = 9 then
        count <= 0;
      else
        count <= count + 1;
      end if;
    end if;
  end process count_process;

  an0 <= '0'; -- just keep anode0 on

  led <= '1' when count = 9 else '0';

end Behavioral;
```

Decoder Entity and Architecture Description:

```
--  Jim Duckworth
--  ECE Dept, WPI
--  January 6, 2005
--  decoder.vhd

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity decoder is
  Port (  count : in integer range 0 to 9;
          seven_seg : out std_logic_vector(6 downto 0));
end decoder;

architecture Behavioral of decoder is

  constant zero   : std_logic_vector(6 downto 0) := "1000000";
  constant one    : std_logic_vector(6 downto 0) := "1111001";
  constant two    : std_logic_vector(6 downto 0) := "0100100";
  constant three  : std_logic_vector(6 downto 0) := "0110000";
  constant four   : std_logic_vector(6 downto 0) := "0011001";
  constant five   : std_logic_vector(6 downto 0) := "0010010";
  constant six    : std_logic_vector(6 downto 0) := "0000010";
  constant seven  : std_logic_vector(6 downto 0) := "1111000";
  constant eight  : std_logic_vector(6 downto 0) := "0000000";
  constant nine   : std_logic_vector(6 downto 0) := "0010000";

begin

  --display the count value on the seven segments
  seven_segment_decoder_process: process(count)
  begin
    case count is
      when 0 => seven_seg <= zero;
      when 1 => seven_seg <= one;
      when 2 => seven_seg <= two;
      when 3 => seven_seg <= three;
      when 4 => seven_seg <= four;
      when 5 => seven_seg <= five;
      when 6 => seven_seg <= six;
      when 7 => seven_seg <= seven;
      when 8 => seven_seg <= eight;
      when 9 => seven_seg <= nine;
    end case;
  end process seven_segment_decoder_process;

end Behavioral;
```


User constraints file for counter:

```
NET "an0"   LOC = "F17";
NET "clk_50M" LOC = "B8";
NET "led" LOC = "J14";
NET "reset" LOC = "B18";
NET "seven_seg<0>" LOC = "L18";
NET "seven_seg<1>" LOC = "F18";
NET "seven_seg<2>" LOC = "D17";
NET "seven_seg<3>" LOC = "D16";
NET "seven_seg<4>" LOC = "G14";
NET "seven_seg<5>" LOC = "J17";
NET "seven_seg<6>" LOC = "H14";
```

Note: also can use one of the buttons for reset (e.g BTN 3 = H13)

Synthesized Results: