Nexys4DDR Board Tutorial

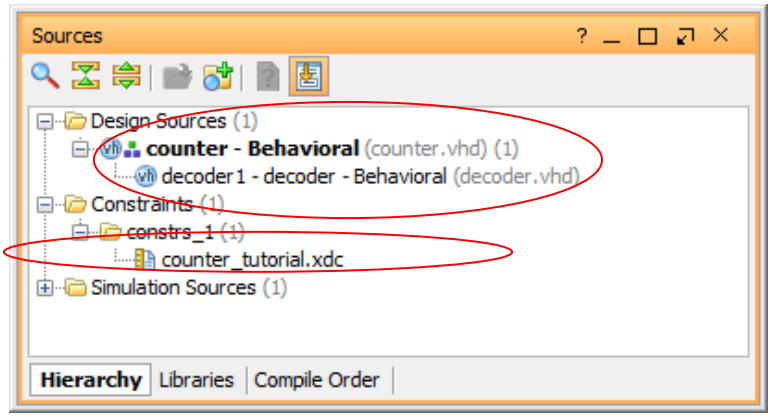
(Counter with decoder, Vivado 2016.2)

Jim Duckworth, August 2016, WPI.

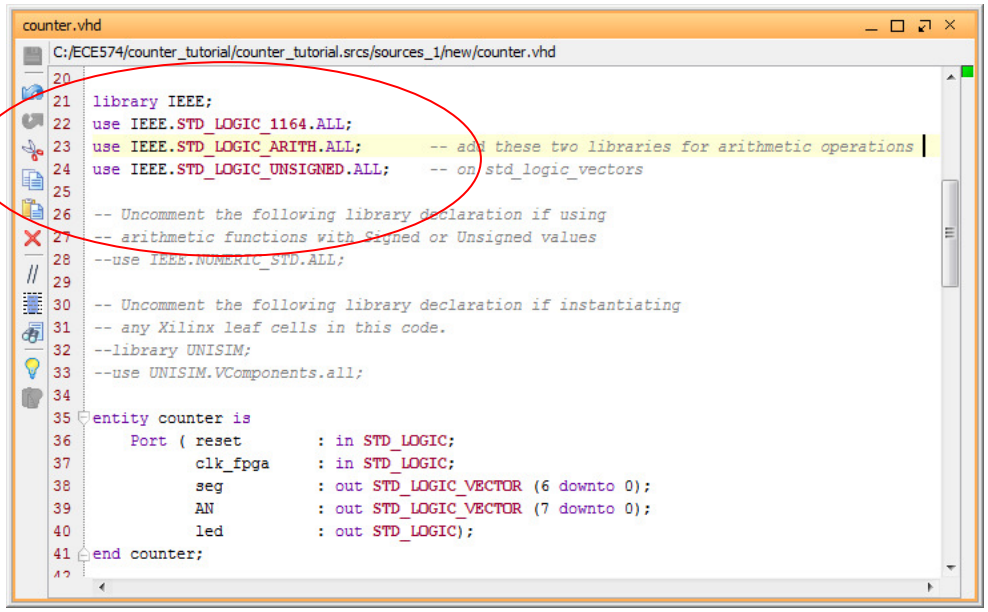
This design shows how to create a simple sequential circuit (a counter).
It also demonstrates hierarchical design by using a separate decoder component that converts a binary count value to a seven segment display.

There are two Design Source files written in VHDL and one Constraint File as shown in the Project Manager Sources window:

The top level design source file is called counter.vhd and the decoder component is in the decoder.vhd file.



These files are shown on the next pages (make sure to add the additional packages to the counter file).



counter.vhd

C:/ECE574/counter_tutorial/counter_tutorial.srcs/sources_1/new/counter.vhd

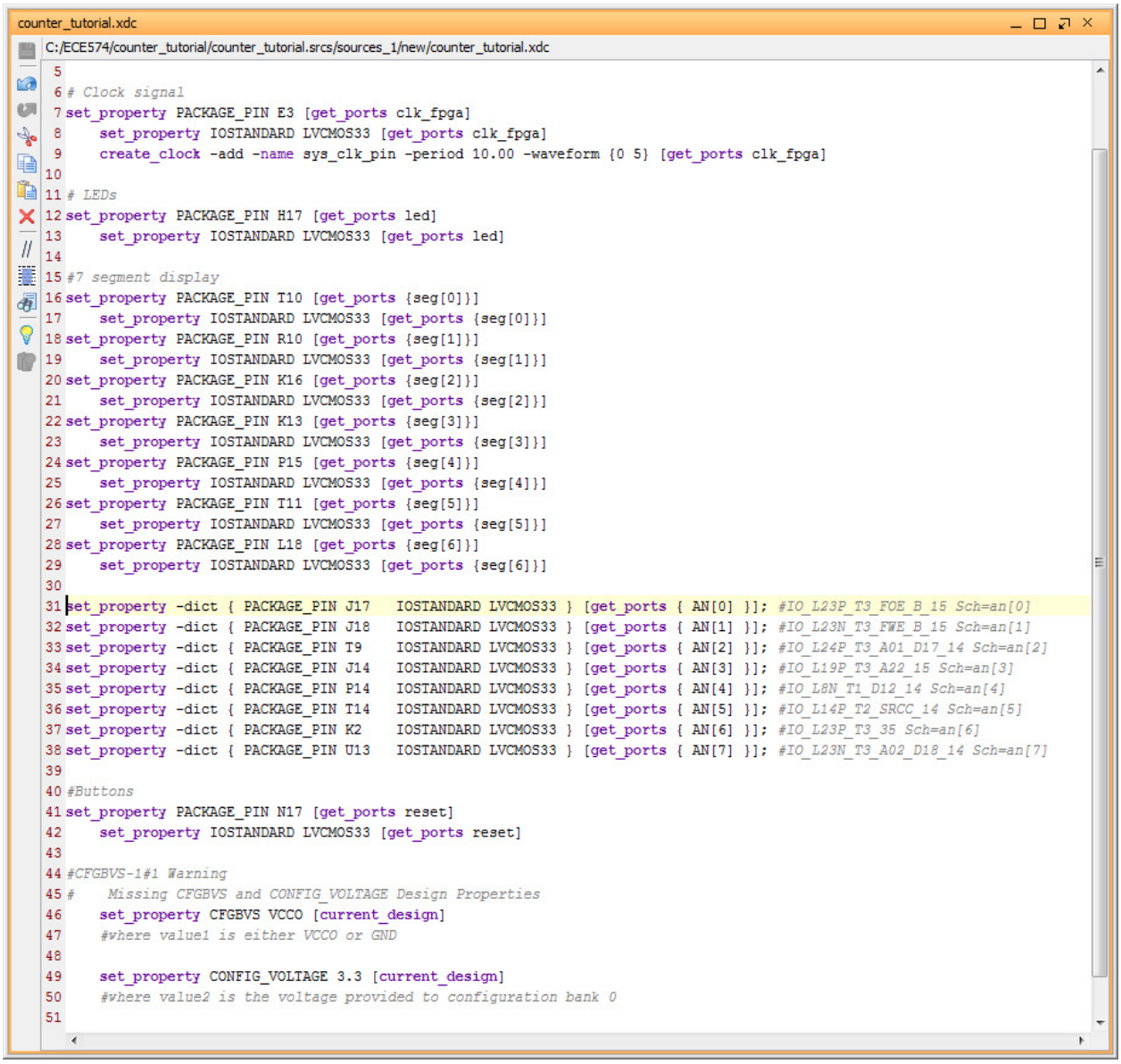
```
34
35  entity counter is
36      Port ( reset        : in STD_LOGIC;
37             clk_fpga     : in STD_LOGIC;
38             seg          : out STD_LOGIC_VECTOR (6 downto 0);
39             AN           : out STD_LOGIC_VECTOR (7 downto 0);
40             led          : out STD_LOGIC);
41  end counter;
42
43  architecture Behavioral of counter is
44
45      component decoder is
46          port ( count        : in std_logic_vector (3 downto 0);
47                 seven_seg    : out std_logic_vector (6 downto 0) );
48      end component;
49
50      signal counter_100M     : integer range 0 to 99_999_999;
51      signal counter_enable   : std_logic;
52      signal counter_10       : std_logic_vector (3 downto 0);
53
54  begin
55
56      -- make a copy of the decoder component using named association
57      decoder1: decoder port map ( count => counter_10, seven_seg => seg );
58
59      -- generate a 1 second clock from the 100MHz FPGA clock
60      one_second_process: process (reset, clk_fpga)
61      begin
62          if reset = '1' then
63              counter_100M <= 0;
64          elsif clk_fpga'event and clk_fpga = '1' then    -- rising edge of clock
65              if counter_100M = 99_999_999 then
66                  counter_100M <= 0;
67              else
68                  counter_100M <= counter_100M + 1;
69              end if;
70          end if;
71      end process one_second_process;
72
73      -- create counter enable signal every 100,000,000 clock cycles
74      counter_enable <= '1' when counter_100M = 99_999_999 else '0';
75
76      -- count seconds from 0 to 9
77      process ( reset, clk_fpga )
78      begin
79          if reset = '1' then
80              counter_10 <= "0000";
81          elsif rising_edge(clk_fpga) then
82              if counter_enable = '1' then    -- increment every second
83                  if counter_10 = 9 then
84                      counter_10 <= "0000";
85                  else
86                      counter_10 <= counter_10 + '1';
87                  end if;
88              end if;
89          end if;
90      end process;
91
92      led <= '1' when counter_10 = 9 else '0';  -- turn on led every 10 seconds
93
94      AN <= X"FE"; -- just turn first seven segment display on, others off
95
96  end Behavioral;
```

decoder.vhd

C:/ECE574/counter_tutorial/counter_tutorial.srcs/sources_1/new/decoder.vhd

```
33
34  entity decoder is
35      Port ( count        : in STD_LOGIC_VECTOR (3 downto 0);
36             seven_seg    : out STD_LOGIC_VECTOR (6 downto 0));
37  end decoder;
38
39  architecture Behavioral of decoder is
40
41      constant zero   : std_logic_vector(6 downto 0) := "1000000";
42      constant one    : std_logic_vector(6 downto 0) := "1111001";
43      constant two    : std_logic_vector(6 downto 0) := "0100100";
44      constant three  : std_logic_vector(6 downto 0) := "0110000";
45      constant four   : std_logic_vector(6 downto 0) := "0011001";
46      constant five   : std_logic_vector(6 downto 0) := "0010010";
47      constant six    : std_logic_vector(6 downto 0) := "0000010";
48      constant seven  : std_logic_vector(6 downto 0) := "1111000";
49      constant eight  : std_logic_vector(6 downto 0) := "0000000";
50      constant nine   : std_logic_vector(6 downto 0) := "0010000";
51
52  begin
53
54    --display the count value on the seven segment display
55    seven_segment_decoder_process: process(count)
56    begin
57      case count is
58        when "0000" => seven_seg <= zero;
59        when "0001" => seven_seg <= one;
60        when "0010" => seven_seg <= two;
61        when "0011" => seven_seg <= three;
62        when "0100" => seven_seg <= four;
63        when "0101" => seven_seg <= five;
64        when "0110" => seven_seg <= six;
65        when "0111" => seven_seg <= seven;
66        when "1000" => seven_seg <= eight;
67        when others => seven_seg <= nine;
68      end case;
69    end process seven_segment_decoder_process;
70
71  end Behavioral;
--
```

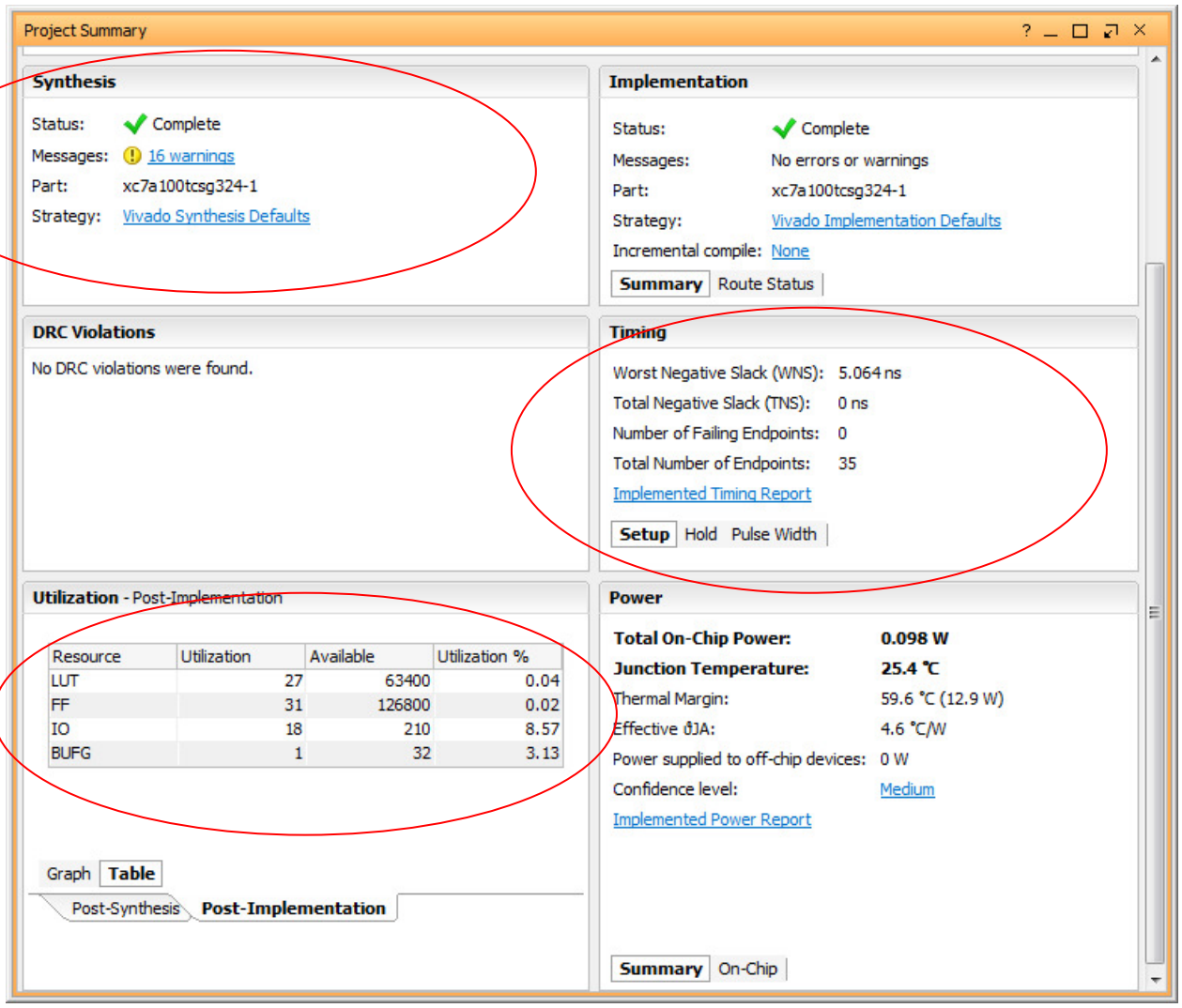
Use the Master XDC file from Digilent to create the constraints file:



After creating the source files, select **Run Implementation**, and **Generate Bitstream**.

You will notice there are 16 Synthesis warnings (shown on the Project Summary window on the next page) due to the constant value we are using to drive the Anodes of the seven segment displays. These warnings are OK.

After generating the bitstream you can Open the **Hardware Manager** and program the Nexys4DDR board and confirm the seven segment display counts from 0 to 9 at 1Hz rate. The LED also turns on when the count reaches 9.



Other useful notes:

In the constraints file we specified the FPGA clock as 100MHz (10 ns period).

```
#  Clock signal
set_property PACKAGE_PIN E3 [get_ports clk_fpga]

        set_property IOSTANDARD LVCMOS33 [get_ports clk_fpga]
        create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports
clk_fpga]
```

This will require the tools to try to implement the design on the FPGA so it can run at this speed.

In the Timing window of the Project Summary it shows the requested timing was met.

We can also see the FPGA resources required to implement the design in the Utilization window. There are a total of 42 LUTs and 31 FFs required. The LUTs will be used to implement the combinational logic used in the design, and the flip-flops will be used to implement the sequential logic. The 31 flip-flops are required to implement the 1Hz clock (27 flip flops) and the digit counter (4 flip flops).