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.

Sources	?_□₽×
으 🄀 🖨 🖬 🔂	
Design Sources (1)	
🖶 🐠 🔒 counter - Behavioral (counter.vhd) (1)	
decoder 1 - decoder - Behavioral (decoder.v	'hd)
🖨 🕞 Constraints (1)	
😑 🗁 constrs_1 (1)	
counter_tutorial.xdc	
È imulation Sources (1)	
Hierarchy Libraries Compile Order	

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

	cou	nter.v	hd _ 🗆 🗸	١×						
	8	C:/ECE574/counter_tutorial/counter_tutorial.srcs/sources_1/new/counter.vhd								
	-	20	· · · · · · · · · · · · · · · · · · ·							
		21	library IEEE;							
	Ch.	22	use IEEE.STD_LOGIC_1164.ALL;							
	to	23	use IEEE.STD_LOGIC_ARITH.ALL; add these two libraries for arithmetic operations							
		24 use IEEE.STD_LOGIC_UNSIGNED.ALL; on std_logic_vectors								
\triangleleft	13	25								
	3	26	Uncomment the following library declaration if using							
	~	20	arithmetic functions with Signed or Unsigned values							
	//	20	USE IEEE.NUMERIC_STD.ALL;							
		30	Uncomment the following library declaration if instantiating							
		31	any Xilinx leaf cells in this code.							
	den .	32	library UNISIM;							
	8	33	use UNISIM.VComponents.all;							
	1	34								
		35 Ę	entity counter is							
		36	<pre>Port (reset : in STD_LOGIC;</pre>							
		37	clk_fpga : in STD_LOGIC;							
		38	<pre>seg : out STD_LOGIC_VECTOR (6 downto 0);</pre>							
		39	AN : out STD_LOGIC_VECTOR (7 downto 0);							
		40	<pre>led : out STD_LOGIC);</pre>							
		41 6	end counter;	-						
		12	•	F						

```
_ 🗆 🖉 ×
counter.vhd
C:/ECE574/counter_tutorial/counter_tutorial.srcs/sources_1/new/counter.vhd
  34
                                                                                             .
35 entity counter is
CT 36
                              : in STD LOGIC;
          Port ( reset
37
                              : in STD LOGIC;
                 clk_fpga
   38
                               : out STD LOGIC VECTOR (6 downto 0);
                  seg
39
                              : out STD LOGIC VECTOR (7 downto 0);
                 AN
40
                  led
                              : out STD LOGIC);
X 41 end counter;
   42
1 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;
          signal counter_enable : std logic;
   51
   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 Ġ
           one_second_process: process (reset, clk_fpga)
   61
           begin
   62 🤅
               if reset = '1' then
   63
                  counter 100M <= 0;
               elsif clk fpga'event and clk fpga = '1' then -- rising edge of clock
   64
   65 🤤
                  if counter 100M = 99 999 999 then
                      counter_100M <= 0;</pre>
   66
   67
                   else
                       counter_100M <= counter_100M + 1;</pre>
   68
   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';</pre>
   75
   76
           -- count seconds from 0 to 9
   77 0
           process ( reset, clk_fpga )
   78
           begin
   79 🖯
               if reset = '1' then
                   counter_10 <= "0000";</pre>
   80
   81
               elsif rising edge(clk_fpga) then
   82 🖯
                  if counter_enable = '1' then
                                                 -- increment every second
                       if counter_10 = 9 then
   83 🖯
                           counter 10 <= "0000";
   84
   85
                       else
   86
                          counter_10 <= counter_10 + '1';</pre>
   87 👌
                       end if;
   88 🖨
                   end if;
   89 🏟
               end if;
   90 🖨
          end process;
   91
           led <= '1' when counter_10 = 9 else '0'; -- turn on led every 10 seconds</pre>
   92
   93
   94
           AN <= X"FE"; -- just turn first seven segment display on, others off
   95
   96 end Behavioral;
        4
```

```
decoder.vhd
                                                                     _ 🗆 🖉 ×
C:/ECE574/counter_tutorial/counter_tutorial.srcs/sources_1/new/decoder.vhd
   33
                                                                              .
10
   34 entity decoder is
35
                              : in STD LOGIC VECTOR (3 downto 0);
         Port ( count
                 seven_seg : out STD LOGIC VECTOR (6 downto 0));
J. 36
   37 ⊖end decoder;
38
39 earchitecture Behavioral of decoder is
× 40
          constant zero : std logic vector(6 downto 0) := "1000000";
   41
//
         constant one : std logic vector(6 downto 0) := "1111001";
   42
43
         constant two : std logic vector(6 downto 0) := "0100100";
         constant three : std logic vector (6 downto 0) := "0110000";
   44
A
         constant four : std logic vector(6 downto 0) := "0011001";
   45
8
         constant five : std logic vector(6 downto 0) := "0010010";
   46
  47
          constant six : std logic vector(6 downto 0) := "0000010";
R
   48
          constant seven : std logic vector(6 downto 0) := "1111000";
         constant eight : std_logic_vector(6 downto 0) := "00000000";
   49
   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 0
         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;
   ---
       .€
                                    111
```

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

```
counter tutorial.xdc
                                                                                                                                     _ 🗆 🖉 🗙
C:/ECE574/counter_tutorial/counter_tutorial.srcs/sources_1/new/counter_tutorial.xdc
10
    6 # Clock signal
7 set_property PACKAGE_PIN E3 [get_ports clk_fpga]
          set property IOSTANDARD LVCMOS33 [get ports clk fpga]
8
          create clock -add -name sys clk pin -period 10.00 -waveform {0 5} [get ports clk fpga]
10
11 # LEDs
X 12 set_property PACKAGE_PIN H17 [get_ports led]
          set_property IOSTANDARD LVCMOS33 [get_ports led]
   13
// 14
15 #7 segment display
16 set_property PACKAGE_PIN T10 [get_ports {seg[0]}]
          set_property IOSTANDARD LVCMOS33 [get_ports {seg[0]}]
    17
18 set property PACKAGE_PIN R10 [get_ports {seg[1]}]
19
         set property IOSTANDARD LVCMOS33 [get ports {seg[1]}]
   20 set_property PACKAGE_PIN K16 [get_ports {seg[2]}]
          set property IOSTANDARD LVCMOS33 [get ports {seg[2]}]
   21
   22 set property PACKAGE_PIN K13 [get ports {seg[3]}]
          set property IOSTANDARD LVCMOS33 [get ports {seg[3]}]
   23
   24 set_property PACKAGE_PIN P15 [get_ports {seg[4]}]
   25
          set property IOSTANDARD LVCMOS33 [get ports {seg[4]}]
   26 set property PACKAGE_PIN T11 [get ports {seg[5]}]
   27
          set property IOSTANDARD LVCMOS33 [get ports {seg[5]}]
   28 set property PACKAGE_PIN L18 [get ports {seg[6]}]
          set property IOSTANDARD LVCMOS33 [get ports {seg[6]}]
   29
   30
   31 set_property -dict { PACKAGE_PIN J17 IOSTANDARD LVCMOS33 } [get_ports { AN[0] }]; #IO_L23P_T3_FOE_B_15 Sch=an[0]
   32 set_property -dict { PACKAGE_PIN J18 IOSTANDARD LVCMOS33 } [get_ports { AN[1] }]; #IO_L23N_T3_FWE_B_15 Sch=an[1]

        33 set_property -dict { PACKAGE_PIN T9
        IOSTANDARD LVCMOS33 } [get_ports { AN[2] }]; #IO_L24P_T3_A01_D17_14 Sch=an[2]

        34 set_property -dict { PACKAGE_PIN J14
        IOSTANDARD LVCMOS33 } [get_ports { AN[3] }]; #IO_L19P_T3_A22_15 Sch=an[3]

      35 set_property -dict { PACKAGE_PIN P14
      IOSTANDARD LVCM0S33 } [get_ports { AN[4] }]; #IO_L8N_T1_D12_14 Sch=an[4]

      36 set_property -dict { PACKAGE_PIN T14
      IOSTANDARD LVCM0S33 } [get_ports { AN[5] }]; #IO_L14P_T2_SRCC_14 Sch=an[5]

   37 set property -dict { PACKAGE PIN K2 IOSTANDARD LVCMOS33 } [get ports { AN[6] }]; #IO L23P T3 35 Sch=an[6]
   38 set_property -dict { PACKAGE_PIN U13 IOSTANDARD LVCMOS33 } [get_ports { AN[7] }]; #IO_L23N_T3_A02_D18_14 Sch=an[7]
   39
   40 #Buttons
   41 set property PACKAGE PIN N17 [get ports reset]
   42
          set property IOSTANDARD LVCMOS33 [get ports reset]
   43
    44 #CFGBVS-1#1 Warning
   45 # Missing CFGBVS and CONFIG VOLTAGE Design Properties
   46
          set_property CFGBVS VCC0 [current_design]
   47
          #where value1 is either VCCO or GND
   48
   49
          set property CONFIG VOLTAGE 3.3 [current design]
   50
          #where value2 is the voltage provided to configuration bank 0
    51
```

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.

Synthesis			Implementation		
Status: 🗸	Complete		\searrow	Status: 🗸 Comple	te
Messages: 1 16 warnings			Messages: No errors or warnings		
Part: xc7a100tcsg324-1			Part: xc7a100tcsg324-1		
Strategy: Vivado Synthesis Defaults			Strategy: Vivado Implementation Defaults		
			Incremental compile: None Summary Route Status		
					DRC Violations
No DRC violation	s were found.		/	Worst Negative Slack (WNS): 5.	064 ns
			Total Negative Slack (TNS): 0 ns		
			Number of Failing Endpoints: 0		
			(Total Number of Endpoints: 35	
				Implemented Timing Report	
				Setup Hold Pulse Width	
Resource	Utilization Ava	ilable Utiliz	ation %	Junction Temperature:	25.4 °C
LUT	27	63400	0.04	Thermal Margin:	59.6 °C (12.9 W)
IO	18	210	8.57	Effective 0JA:	4.6 °C/W
BUFG	1	32	3.13	Power supplied to off-chip device	s: 0 W
				Confidence level:	Medium
				Implemented Power Report	
				2323	
Graph Table					

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).