

## **Basys3 Board Tutorial - Counter (Verilog Version)**

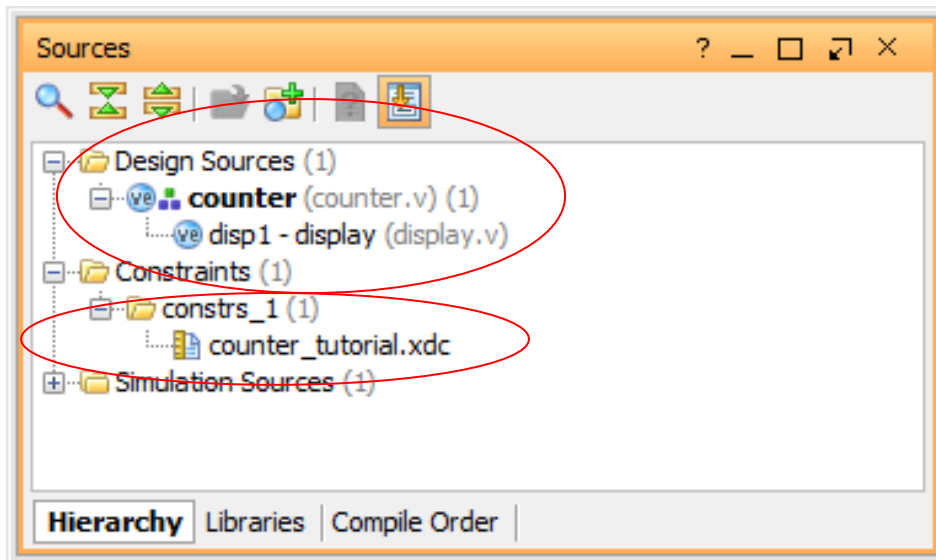
Jim Duckworth, WPI. August 2016

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

This project used Xilinx Vivado 2016.2 targeted to the Basys3 board but it should be easily adapted to other boards such as the Nexy4DDY board.

It is recommended that you complete the simpler Verilog Decoder Tutorial before attempting this tutorial. This will show how to create a new project and add design sources.

There are two design sources and one constraint file used in this project:



The top level module is called *counter* and contains the statements to generate a 1Hz clock from the 100MHz FPGA clock, and a counter to count from 0 to 9 at the 1Hz rate.

The top level module also instantiates a copy of the lower level display module.

The *display* module converts the 0 to 9 value to a seven segment character.

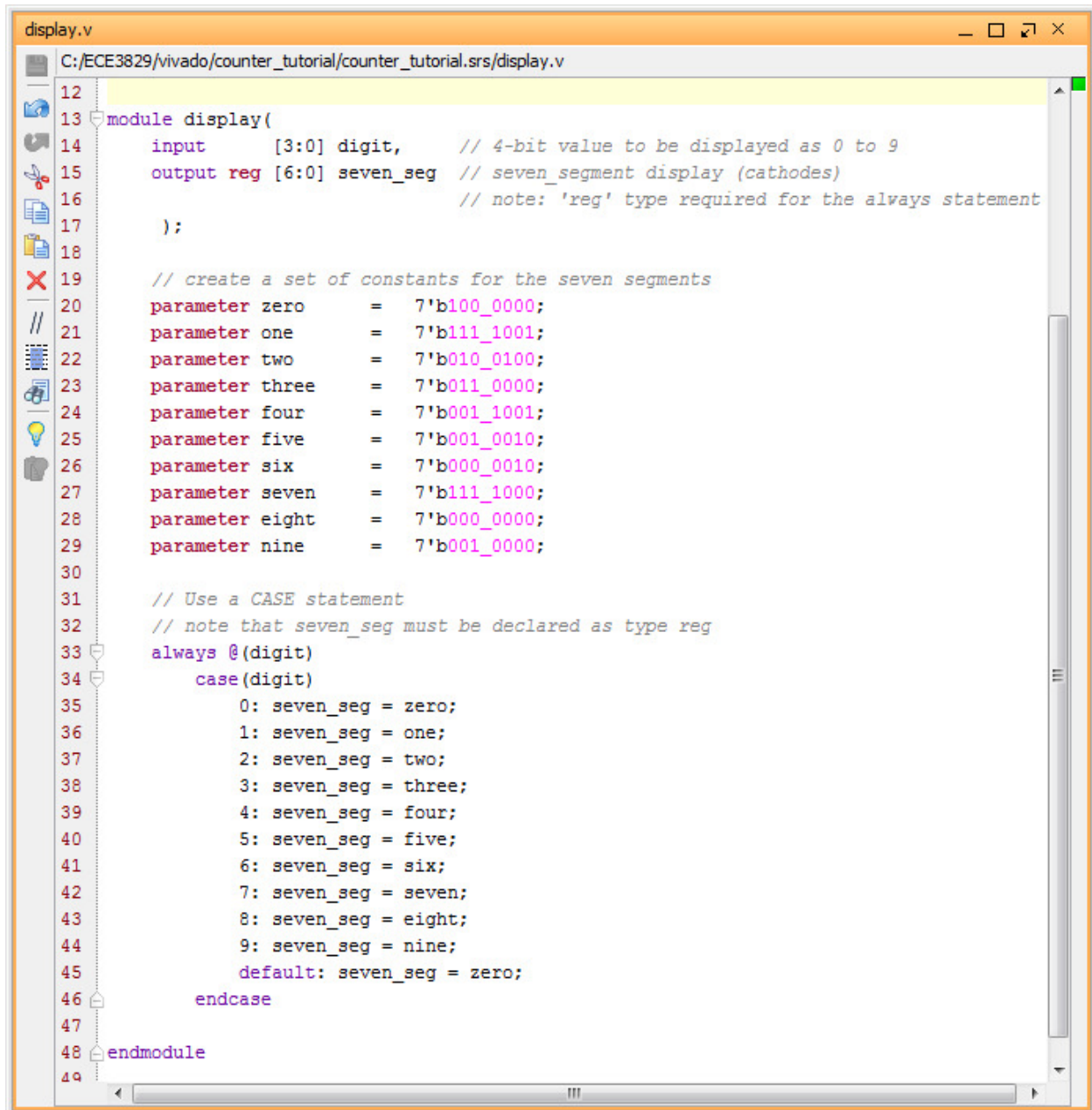
The third source is the constraints file to specify the FPGA pins used for each of the ports in the project. This constraint file is specific to the Basys3 board.

The three design sources are shown below:

```

counter.v
C:/ECE3829/vivado/counter_tutorial/counter_tutorial.srs/counter.v
13
14 module counter(
15     input        clk_fpga,        // 100MHz FPGA clock
16     input        reset,           // external reset (connect to push button)
17     output [6:0]  seg,             // seven_segment display (cathodes)
18     output [3:0]  an,             // seven_segment display (anodes)
19     output        led             // LED
20 );
21
22 parameter MAX_COUNT = 100_000_000 - 1; // max count value
23 wire      counter_en;
24 reg [26:0] counter_100M;           // to count from 0 to 99,999,999
25 reg [3:0]  counter_10;            // count from 0 to 9
26
27 // only turn on the first seven segment display
28 assign an = 4'b1110;
29
30 // instantiate copy of display module using named association
31 // counter_10 signal is connected to the digit port
32 // seg signal is connected to the seven_seg port
33 display disp1 (.digit(counter_10), .seven_seg(seg));
34
35 // create a counter to divide the 100MHz FPGA clock to 1Hz
36 always @ (posedge clk_fpga, posedge reset)
37     if (reset)
38         counter_100M <= 0;
39     else if (counter_100M == MAX_COUNT) // restart every 50,000,000 clocks
40         counter_100M <= 0;
41     else
42         counter_100M <= counter_100M + 1'b1;
43
44 // create a signal that is active every 100,000,000 clocks
45 assign counter_en = counter_100M == 0; // enable counter every 100M clock cycles
46
47 // create a counter to count from 0 to 9 at 1Hz rate
48 always @ (posedge clk_fpga, posedge reset) // must use same 100MHz clock
49     if (reset)
50         counter_10 <= 0;
51     else if (counter_en) // counter enable (at 1 Hz)
52         if (counter_10 == 9) // count from 0 to 9
53             counter_10 <= 0;
54         else
55             counter_10 <= counter_10 + 1'b1;
56
57 // flash an LED every 10 seconds
58 assign led = counter_10 == 9;
59
60 endmodule
61

```



```
12
13 module display(
14     input    [3:0] digit,      // 4-bit value to be displayed as 0 to 9
15     output reg [6:0] seven_seg // seven_segment display (cathodes)
16                                     // note: 'reg' type required for the always statement
17 );
18
19 // create a set of constants for the seven segments
20 parameter zero      = 7'b100_0000;
21 parameter one       = 7'b111_1001;
22 parameter two       = 7'b010_0100;
23 parameter three     = 7'b011_0000;
24 parameter four      = 7'b001_1001;
25 parameter five      = 7'b001_0010;
26 parameter six       = 7'b000_0010;
27 parameter seven     = 7'b111_1000;
28 parameter eight     = 7'b000_0000;
29 parameter nine      = 7'b001_0000;
30
31 // Use a CASE statement
32 // note that seven_seg must be declared as type reg
33 always @(digit)
34     case(digit)
35         0: seven_seg = zero;
36         1: seven_seg = one;
37         2: seven_seg = two;
38         3: seven_seg = three;
39         4: seven_seg = four;
40         5: seven_seg = five;
41         6: seven_seg = six;
42         7: seven_seg = seven;
43         8: seven_seg = eight;
44         9: seven_seg = nine;
45         default: seven_seg = zero;
46     endcase
47
48 endmodule
49
```

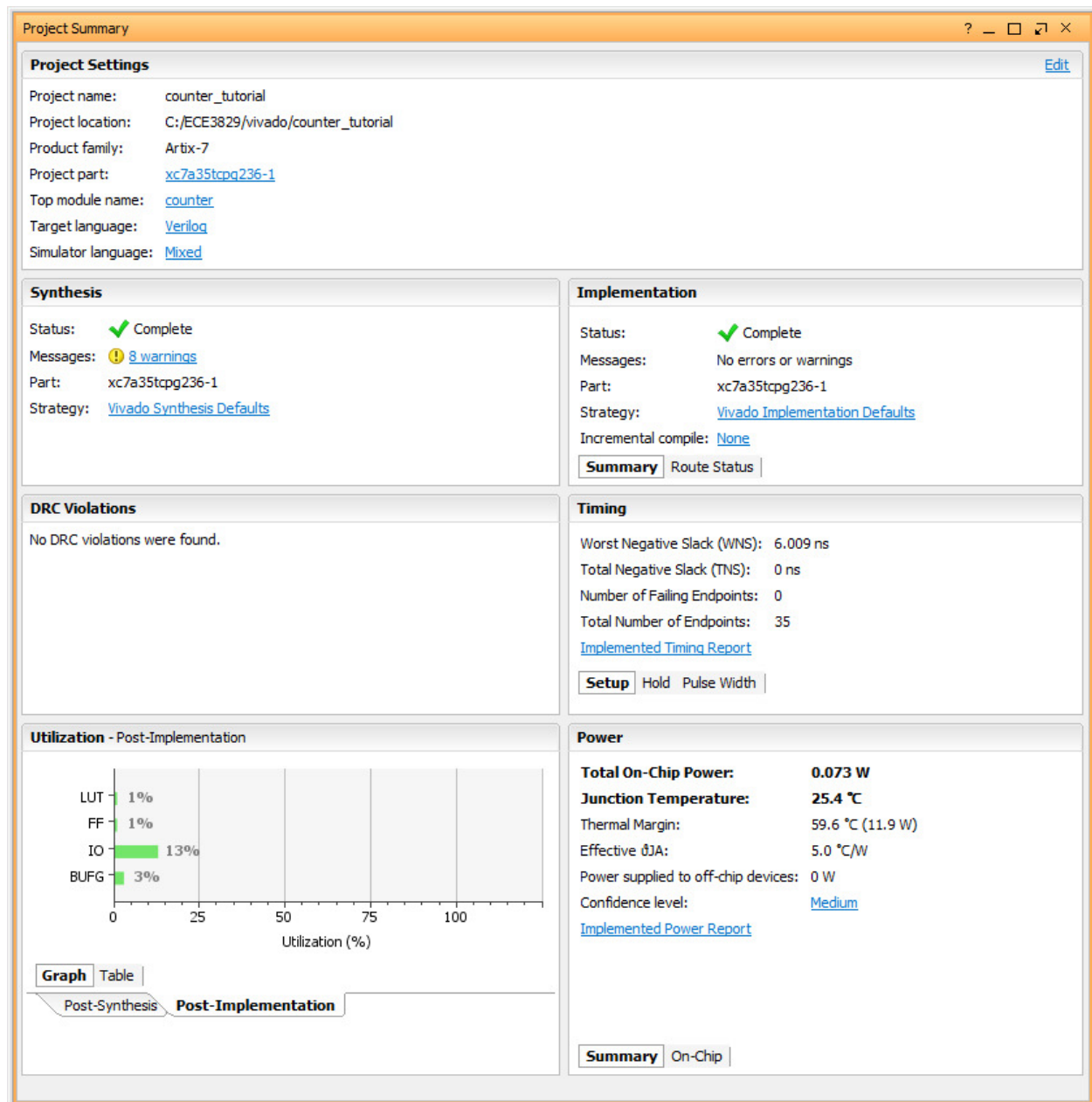
```

counter_tutorial.xdc
C:/ECE3829/vivado/counter_tutorial/counter_tutorial.xdc
5
6 # Clock signal
7 set_property PACKAGE_PIN W5 [get_ports clk_fpga]
8   set_property IOSTANDARD LVCMOS33 [get_ports clk_fpga]
9   create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk_fpga]
10
11 # LEDs
12 set_property PACKAGE_PIN U16 [get_ports led]
13   set_property IOSTANDARD LVCMOS33 [get_ports led]
14
15 #7 segment display
16 set_property PACKAGE_PIN W7 [get_ports {seg[0]}]
17   set_property IOSTANDARD LVCMOS33 [get_ports {seg[0]}]
18 set_property PACKAGE_PIN W6 [get_ports {seg[1]}]
19   set_property IOSTANDARD LVCMOS33 [get_ports {seg[1]}]
20 set_property PACKAGE_PIN U8 [get_ports {seg[2]}]
21   set_property IOSTANDARD LVCMOS33 [get_ports {seg[2]}]
22 set_property PACKAGE_PIN V8 [get_ports {seg[3]}]
23   set_property IOSTANDARD LVCMOS33 [get_ports {seg[3]}]
24 set_property PACKAGE_PIN U5 [get_ports {seg[4]}]
25   set_property IOSTANDARD LVCMOS33 [get_ports {seg[4]}]
26 set_property PACKAGE_PIN V5 [get_ports {seg[5]}]
27   set_property IOSTANDARD LVCMOS33 [get_ports {seg[5]}]
28 set_property PACKAGE_PIN U7 [get_ports {seg[6]}]
29   set_property IOSTANDARD LVCMOS33 [get_ports {seg[6]}]
30
31 set_property PACKAGE_PIN U2 [get_ports {an[0]}]
32   set_property IOSTANDARD LVCMOS33 [get_ports {an[0]}]
33 set_property PACKAGE_PIN U4 [get_ports {an[1]}]
34   set_property IOSTANDARD LVCMOS33 [get_ports {an[1]}]
35 set_property PACKAGE_PIN V4 [get_ports {an[2]}]
36   set_property IOSTANDARD LVCMOS33 [get_ports {an[2]}]
37 set_property PACKAGE_PIN W4 [get_ports {an[3]}]
38   set_property IOSTANDARD LVCMOS33 [get_ports {an[3]}]
39
40 #Buttons
41 set_property PACKAGE_PIN U18 [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 VCCO [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
52

```



Once you have added the three design sources to the project you can ‘Run Synthesis’, ‘Implementation’, and ‘Generate Bitstream’:



The 8 synthesis warnings are due to the constant value being used to drive the anodes for the seven segment display. This is what we intended so we can ignore these warnings.

Once the bitstream has been generated you can program the FPGA using the Hardware Manager in the Project Manager (see the Decoder Tutorial for details).

Once the FPGA is loaded you should see the seven segment display cycle through the digits 0 to 9 every second. When the count gets to 9 the led should also turn on.

**Other useful notes:**

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

```
# Clock signal
set_property PACKAGE_PIN W5 [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.

The Project Summary below shows the requested timing was met.

We can also see the FPGA resources required to implement the design. 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).

The screenshot shows the 'Project Summary' window with two main sections highlighted by red circles. The 'Timing' section on the right shows that the worst negative slack (WNS) is 6.009 ns, total negative slack (TNS) is 0 ns, and there are 0 failing endpoints. The 'Utilization - Post-Implementation' section on the left shows a table of resource utilization.

Resource	Utilization	Available	Utilization %
LUT	42	20800	0.20
FF	31	41600	0.07
IO	14	106	13.21
BUFG	1	32	3.13

The 'Power' section on the right shows the total on-chip power is 0.073 W, junction temperature is 25.4 °C, and thermal margin is 59.6 °C (11.9 W). The confidence level is Medium.