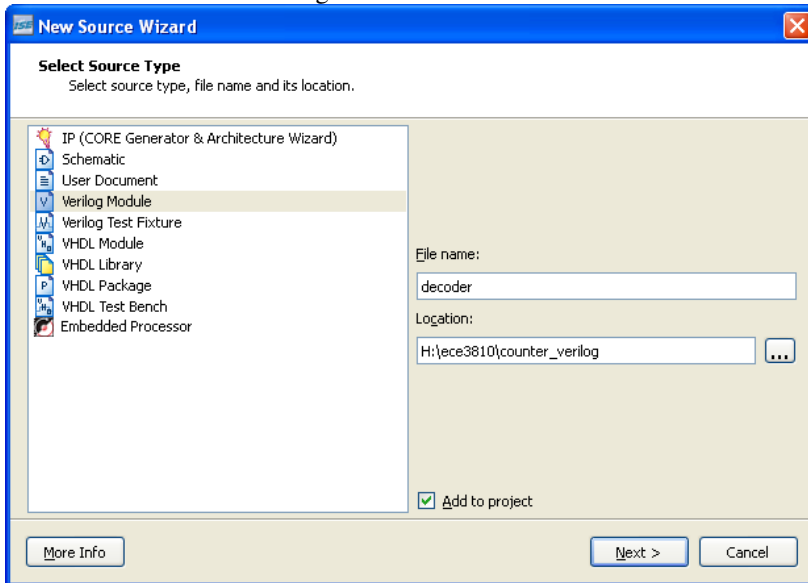


Nexys 2 board tutorial - Counter (Verilog Version)

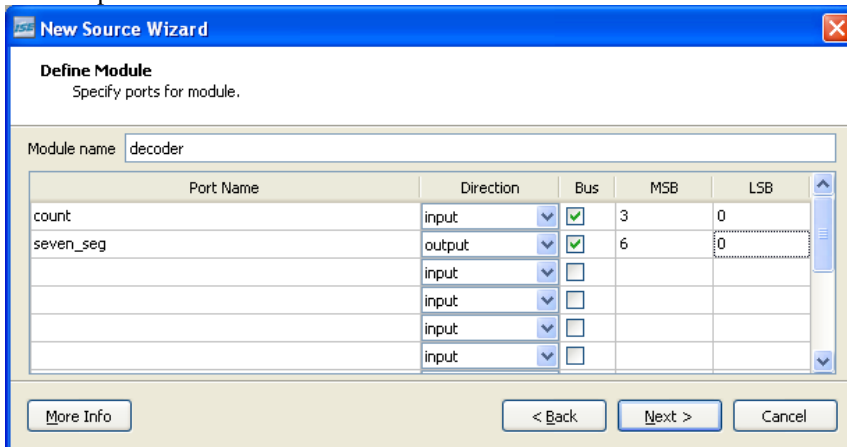
Jim Duckworth, Zoe Fu - September 2010

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

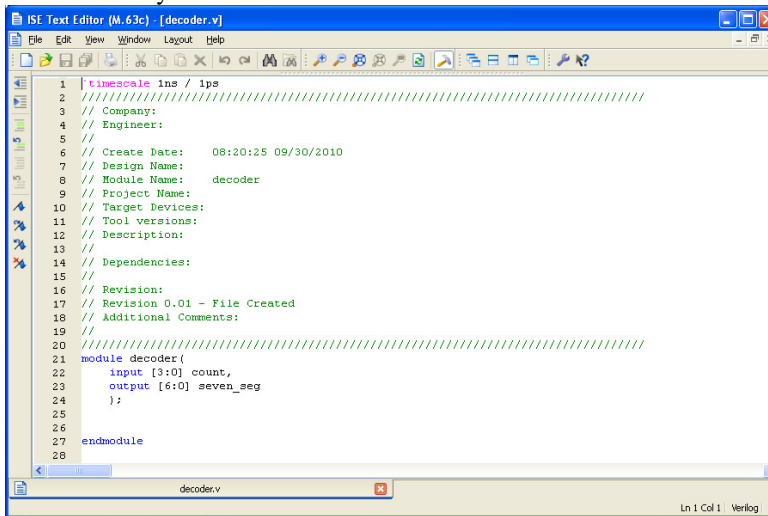
Start with the decoder Verilog module:



Add the port names and sizes:

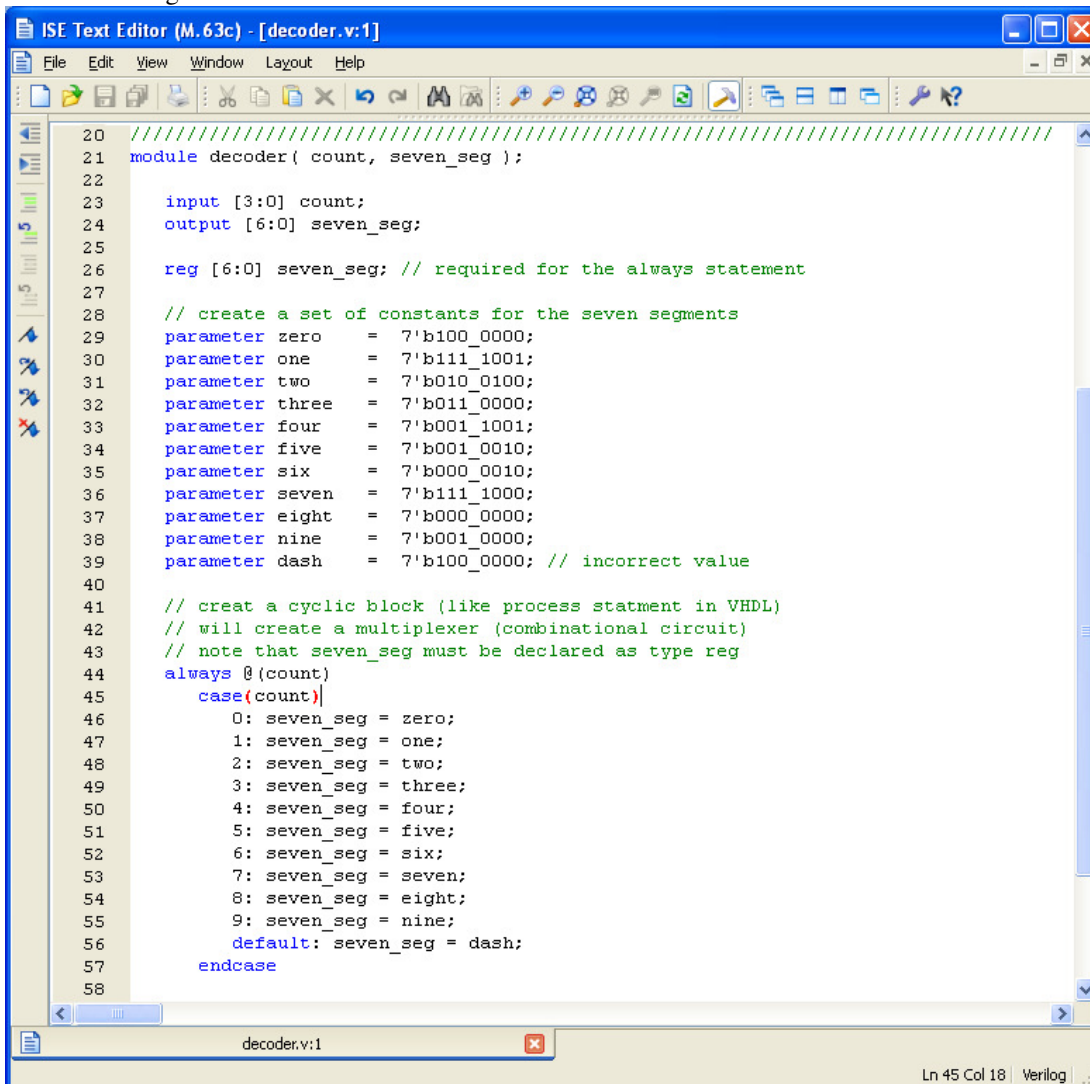


A skeleton of your module is created:



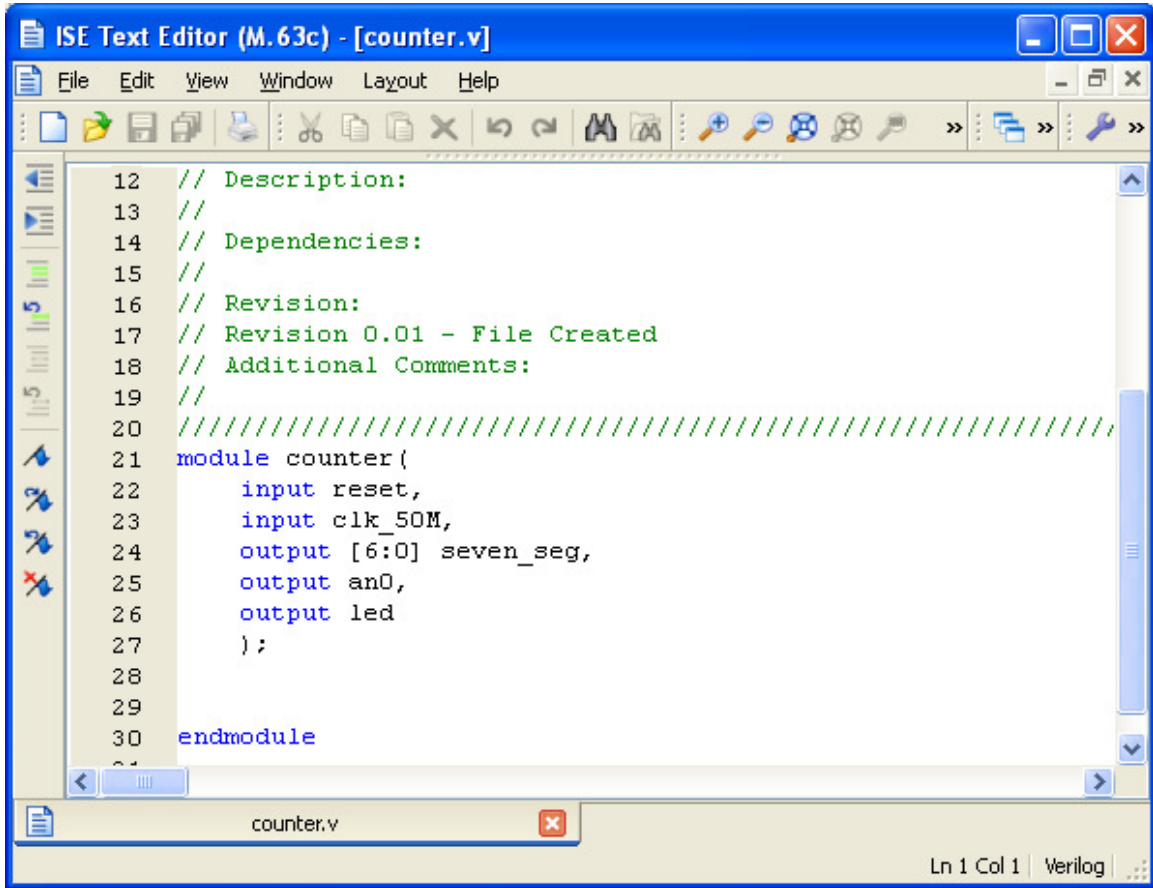
```
ISE Text Editor (M.63c) - [decoder.v]
File Edit View Window Layout Help
1 timescale 1ns / 1ps
2 //////////////////////////////////////////////////
3 // Company:
4 // Engineer:
5 //
6 // Create Date:      08:20:25 09/30/2010
7 // Design Name:
8 // Module Name:     decoder
9 // Project Name:
10 // Target Devices:
11 // Tool versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////
21 module decoder(
22     input [3:0] count,
23     output [6:0] seven_seg
24 );
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
endmodule
decoder.v Ln 1 Col 1 Verilog
```

Add the Verilog statements to make the decoder:



```
ISE Text Editor (M.63c) - [decoder.v:1]
File Edit View Window Layout Help
20 //////////////////////////////////////////////////
21 module decoder( count, seven_seg );
22
23     input [3:0] count;
24     output [6:0] seven_seg;
25
26     reg [6:0] seven_seg; // required for the always statement
27
28     // create a set of constants for the seven segments
29     parameter zero    = 7'b100_0000;
30     parameter one     = 7'b111_1001;
31     parameter two     = 7'b010_0100;
32     parameter three   = 7'b011_0000;
33     parameter four    = 7'b001_1001;
34     parameter five    = 7'b001_0010;
35     parameter six     = 7'b000_0010;
36     parameter seven   = 7'b111_1000;
37     parameter eight   = 7'b000_0000;
38     parameter nine    = 7'b001_0000;
39     parameter dash    = 7'b100_0000; // incorrect value
40
41     // creat a cyclic block (like process statement in VHDL)
42     // will create a multiplexer (combinational circuit)
43     // note that seven_seg must be declared as type reg
44     always @(count)
45     case(count)
46     0: seven_seg = zero;
47     1: seven_seg = one;
48     2: seven_seg = two;
49     3: seven_seg = three;
50     4: seven_seg = four;
51     5: seven_seg = five;
52     6: seven_seg = six;
53     7: seven_seg = seven;
54     8: seven_seg = eight;
55     9: seven_seg = nine;
56     default: seven_seg = dash;
57     endcase
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
decoder.v:1 Ln 45 Col 18 Verilog
```

Now create the top level Counter module:

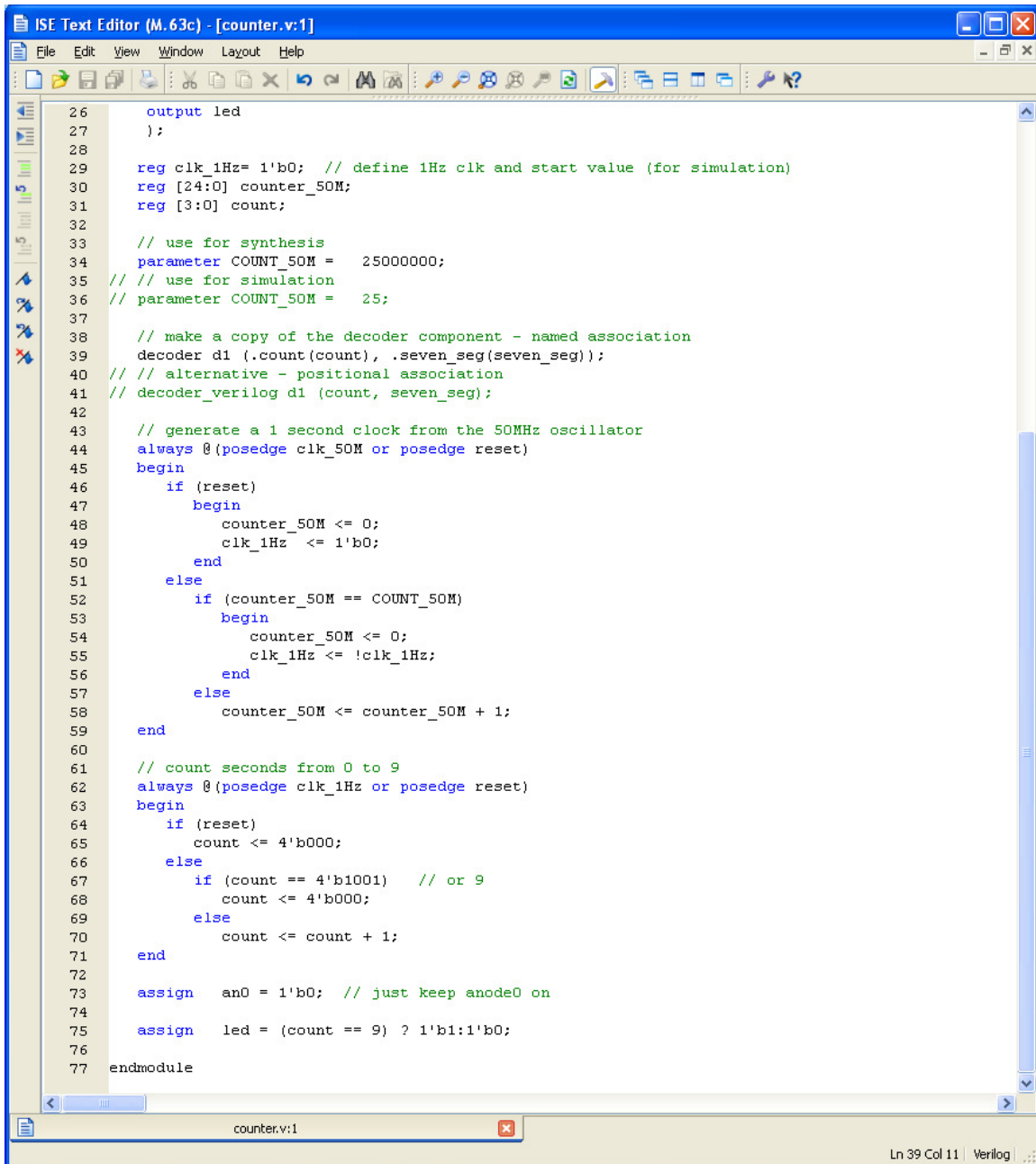


The screenshot shows the ISE Text Editor window titled "ISE Text Editor (M.63c) - [counter.v]". The editor contains the following Verilog code:

```
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 ///////////////////////////////////////////////////////////////////
21 module counter(
22     input reset,
23     input clk_50M,
24     output [6:0] seven_seg,
25     output an0,
26     output led
27 );
28
29
30 endmodule
```

The status bar at the bottom right indicates "Ln 1 Col 1 | Verilog".

And add the behavioral statements:

The image shows a screenshot of the ISE Text Editor window titled "ISE Text Editor (M.63c) - [counter.v:1]". The window contains Verilog code for a counter module. The code includes declarations for outputs, registers, and parameters, followed by logic for clock generation and counting. The status bar at the bottom indicates "Ln 39 Col 11 | Verilog".

```
26     output led
27     );
28
29     reg clk_1Hz= 1'b0; // define 1Hz clk and start value (for simulation)
30     reg [24:0] counter_50M;
31     reg [3:0] count;
32
33     // use for synthesis
34     parameter COUNT_50M = 25000000;
35     // // use for simulation
36     // parameter COUNT_50M = 25;
37
38     // make a copy of the decoder component - named association
39     decoder d1 (.count(count), .seven_seg(seven_seg));
40     // // alternative - positional association
41     // decoder_verilog d1 (count, seven_seg);
42
43     // generate a 1 second clock from the 50MHz oscillator
44     always @(posedge clk_50M or posedge reset)
45     begin
46         if (reset)
47             begin
48                 counter_50M <= 0;
49                 clk_1Hz <= 1'b0;
50             end
51         else
52             if (counter_50M == COUNT_50M)
53                 begin
54                     counter_50M <= 0;
55                     clk_1Hz <= !clk_1Hz;
56                 end
57             else
58                 counter_50M <= counter_50M + 1;
59     end
60
61     // count seconds from 0 to 9
62     always @(posedge clk_1Hz or posedge reset)
63     begin
64         if (reset)
65             count <= 4'b000;
66         else
67             if (count == 4'b1001) // or 9
68                 count <= 4'b000;
69             else
70                 count <= count + 1;
71     end
72
73     assign an0 = 1'b0; // just keep anode0 on
74
75     assign led = (count == 9) ? 1'b1:1'b0;
76
77 endmodule
```

And finally create the UCF file:

```

ISE Text Editor (M.63c) - [counter.ucf]
File Edit View Window Layout Help
1 NET "ANO" LOC = F17;
2 NET "CLK_50M" LOC = B8;
3 # we want the design to run at 50MHz
4 NET "CLK_50M" PERIOD = 20.0ns HIGH 50%;
5 NET "LED" LOC = J14;
6 NET "RESET" LOC = H13;
7 NET "SEVEN_SEG[0]" LOC = L18;
8 NET "SEVEN_SEG[1]" LOC = F18;
9 NET "SEVEN_SEG[2]" LOC = D17;
10 NET "SEVEN_SEG[3]" LOC = D16;
11 NET "SEVEN_SEG[4]" LOC = G14;
12 NET "SEVEN_SEG[5]" LOC = J17;
13 NET "SEVEN_SEG[6]" LOC = H14;
14
counter.ucf
Ln 1 Col 1 UCF

```

Note: we have added a timing constraint to help guide the synthesis tool

We can now synthesize and implement the design:

```

ISE Project Navigator (M.63c) - H:\ece3810\counter_verilog\counter_verilog.xise - [decoder.v:2]
File Edit View Project Source Process Tools Window Layout Help
Design Summary (Programming File Generated) decoder.v:2 counter.v:2
20 //////////////////////////////////////
21 module decoder( count, seven_seg );
22
23 input [3:0] count;
24 output [6:0] seven_seg;
25
26 reg [6:0] seven_seg; // required for the always statement
27
28 // create a set of constants for the seven segments
29 parameter zero = 7'b100_0000;
30 parameter one = 7'b111_1001;
31 parameter two = 7'b010_0100;
32 parameter three = 7'b011_0000;
33 parameter four = 7'b001_1001;
34 parameter five = 7'b001_0010;
35 parameter six = 7'b000_0010;
36 parameter seven = 7'b111_1000;
37 parameter eight = 7'b000_0000;
38 parameter nine = 7'b001_0000;
39 parameter dash = 7'b100_0000; // incorrect value
40
41 // create a cyclic block (like process statement in VHDL)
42 // will create a multiplexer (combinational circuit)
43 // note that seven_seg must be declared as type reg
44 always @ (count)
45 case (count)
46 0: seven_seg = zero;
47 1: seven_seg = one;

```

```

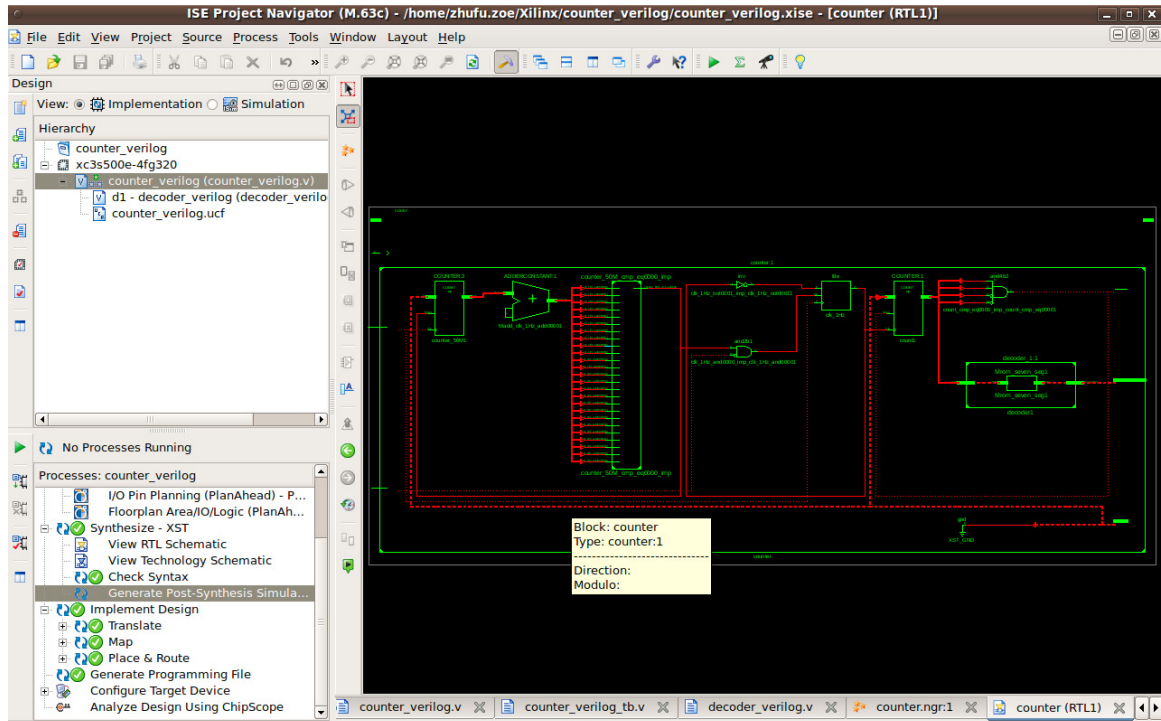
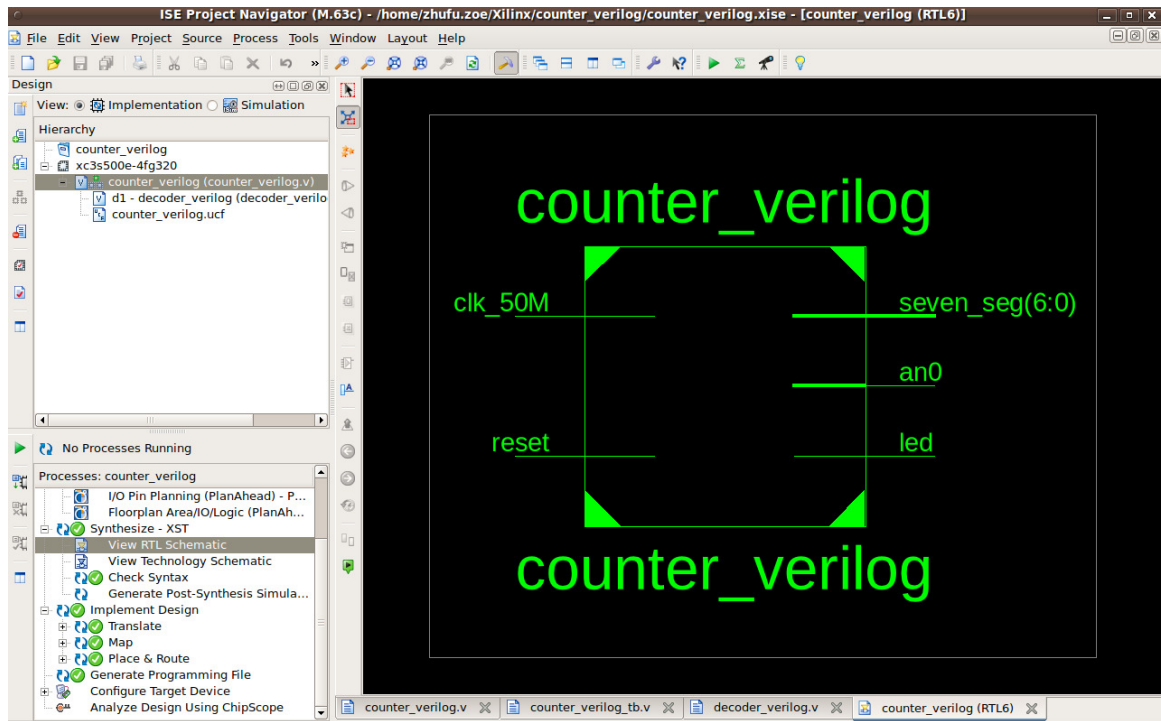
Started: "Generate Programming File".
Running bitgen...
Command Line: bitgen -intstyle ise -f counter.ut counter.ncd
J:\INFO:WebTalk:4 - H:\ece3810\counter_verilog\usage_statistics_webtalk.html
WebTalk report has been successfully sent to Xilinx. For additional details
about this file, please refer to the WebTalk log file at
H:\ece3810\counter_verilog\webtalk.log

WebTalk is complete.

Process "Generate Programming File" completed successfully

```

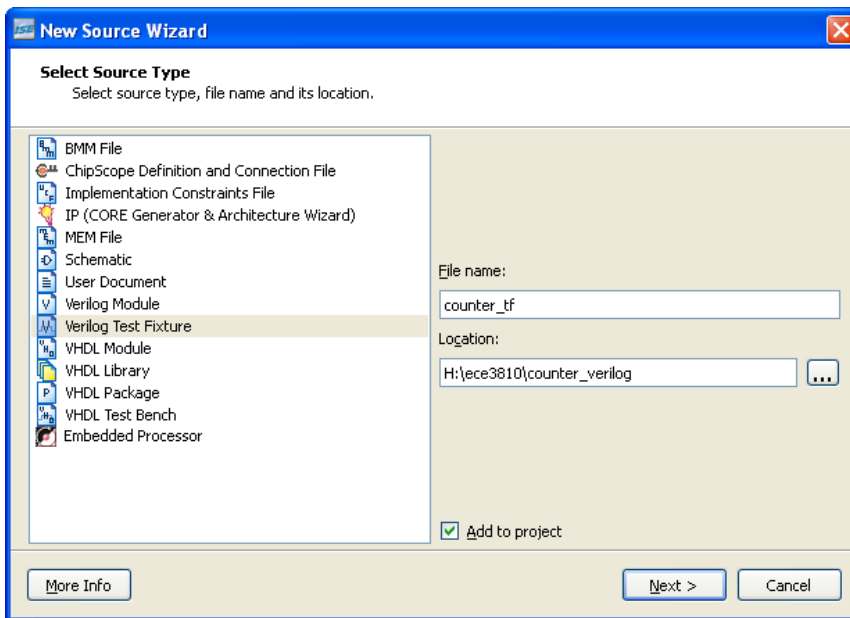
Synthesized Results:



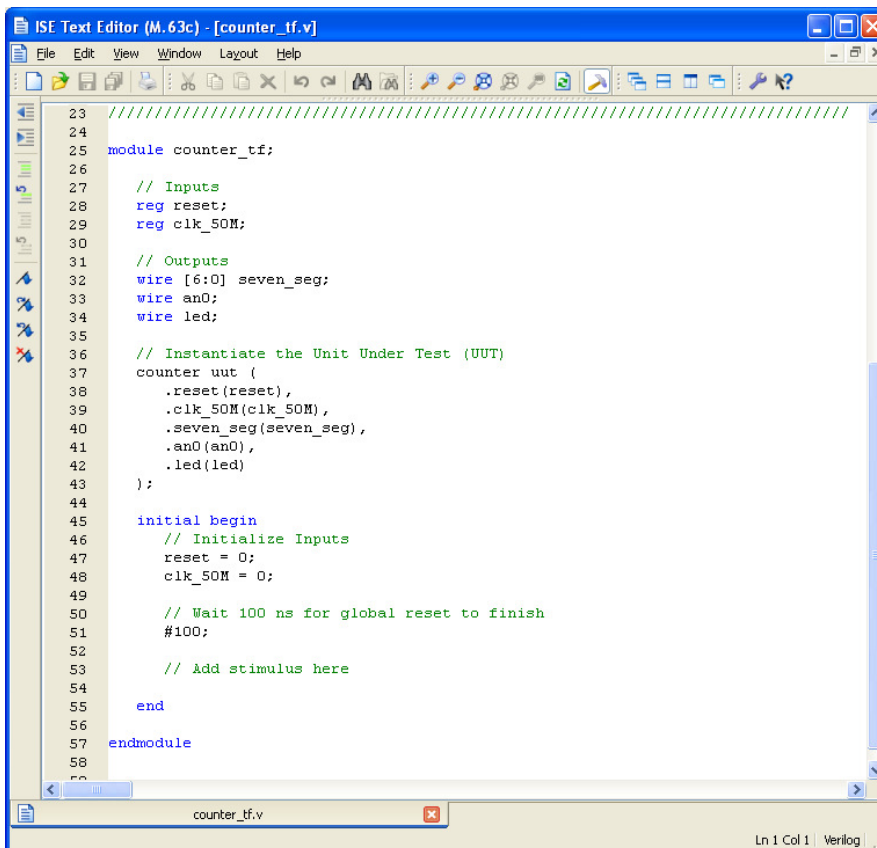
Download the design to the board and test the operation.

We can also create a Test Fixture (test bench to test and simulate the operation).

Create a Test Fixture module:



A skeleton of a test fixture to test your counter is produced:



Modify it so it creates a 50MHz clock, and provides the reset signal for 100ns:

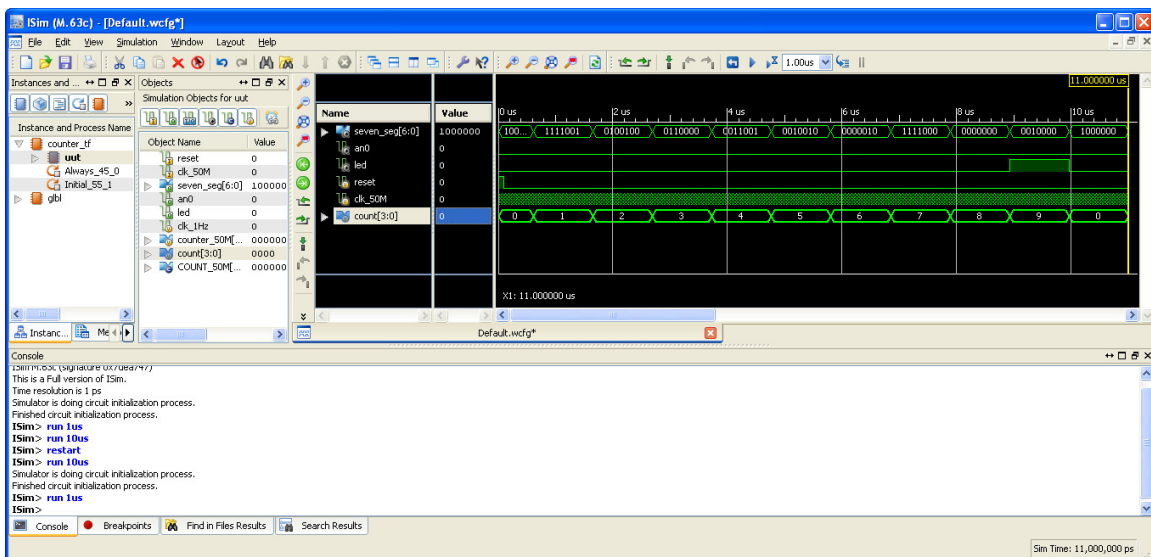
```

ISE Text Editor (M.63c) - [counter_tf.v*]
File Edit View Window Layout Help
32 wire [6:0] seven_seg;
33 wire an0;
34 wire led;
35
36 // Instantiate the Unit Under Test (UUT)
37 counter uut (
38     .reset(reset),
39     .clk_50M(clk_50M),
40     .seven_seg(seven_seg),
41     .an0(an0),
42     .led(led)
43 );
44
45 // create 50MHz clock
46 always
47 begin
48     clk_50M = 0;
49     #10;
50     clk_50M = 1;
51     #10;
52 end
53
54 initial begin
55     // Initialize Inputs
56     reset = 1;
57     // Wait 100 ns for global reset to finish
58     #100;
59     reset = 0;
60
61     // Add other stimulus here
62
63 end
64
65 endmodule
66
67
counter_tf.v*
Ln 43 Col 6 | Verilog

```

Run the simulation (also add the count internal signal) and verify operation.

Note: don't forget to change the PARAMETER COUNT_50M to speed up the simulation



Zoomed in for more detail:

