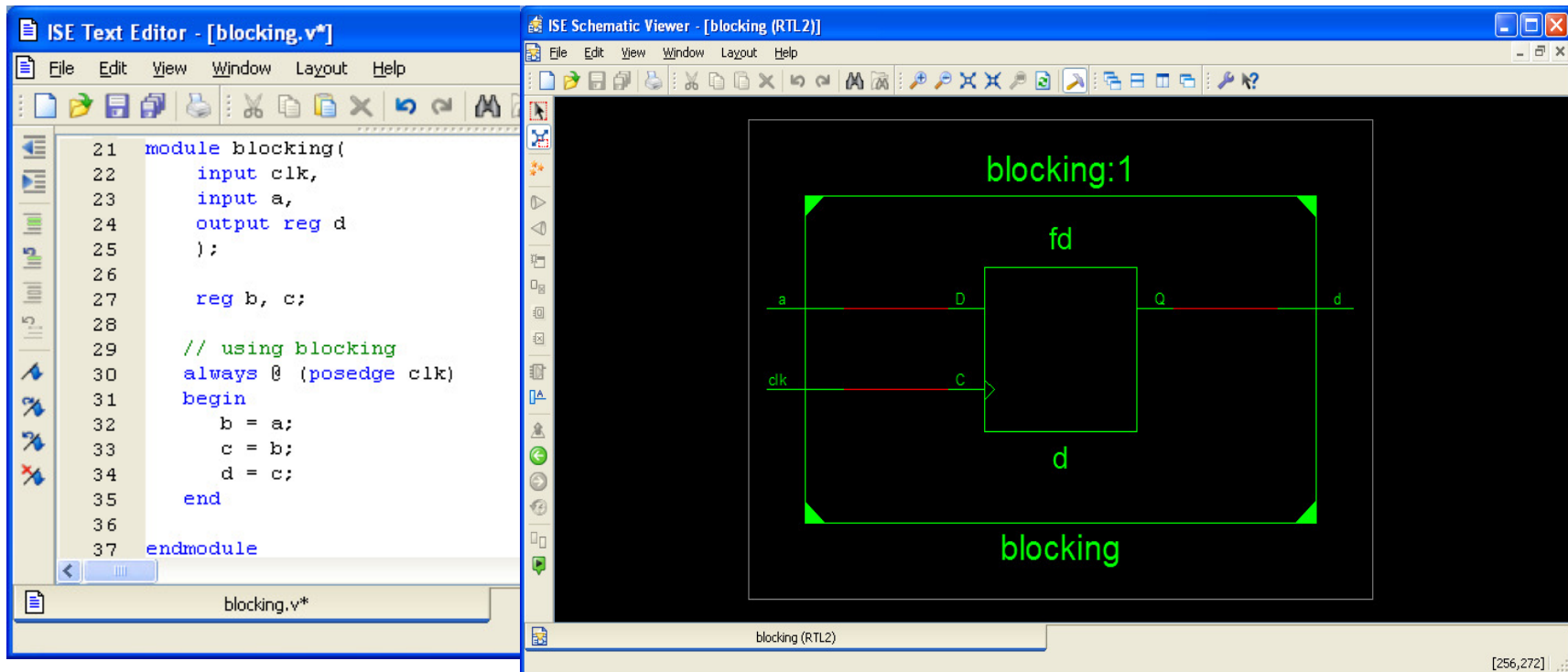

Verilog

Misc

(blocking, time constraints, clocks, customizing
modules)

Blocking Example (incorrect result)



Synthesizing Unit <blocking>.

WARNING:Xst:646 - Signal <c> is assigned but never used. This unconnected signal will be trimmed during the optimization process.

WARNING:Xst:646 - Signal is assigned but never used. This unconnected signal will be trimmed during the optimization process.

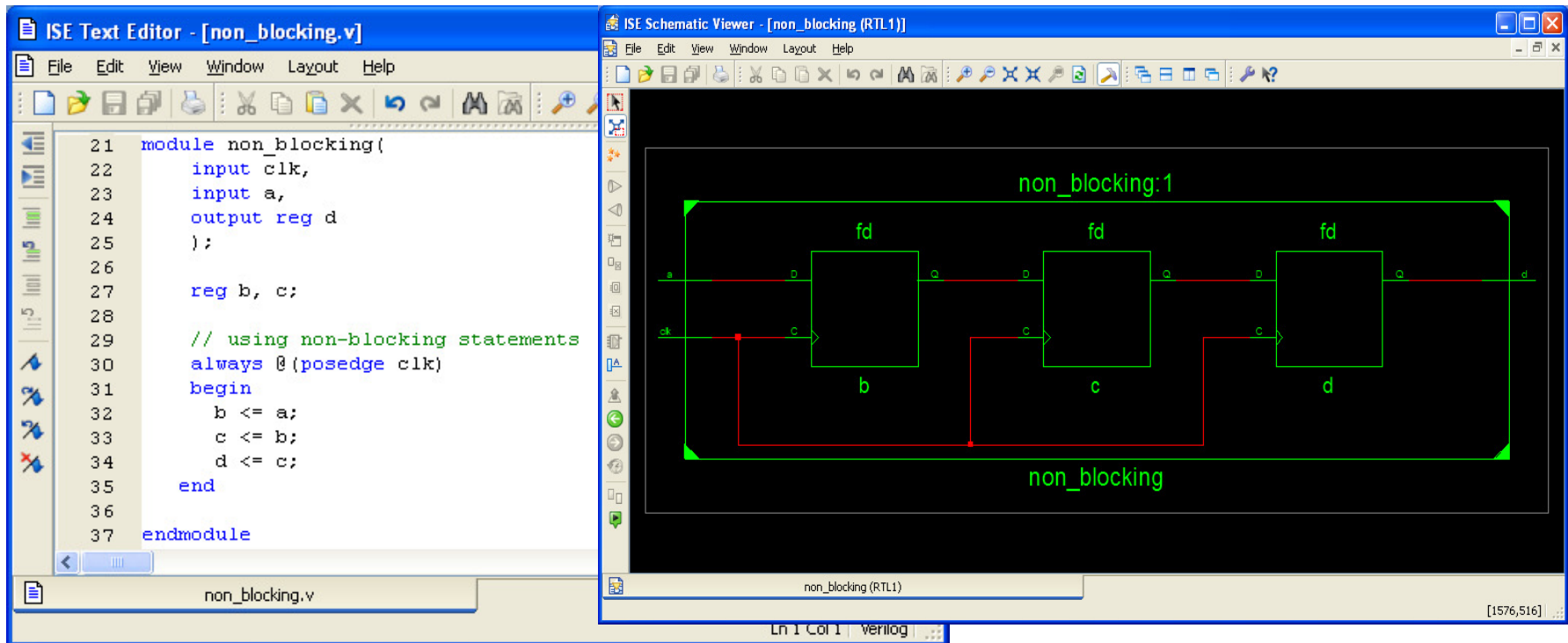
Found 1-bit register for signal <d>.

Summary:

inferred 1 D-type flip-flop(s).

Unit <blocking> synthesized.

Non-Blocking (expected result)



Synthesizing Unit <non_blocking>.
Related source file is "non_blocking.v".
Found 1-bit register for signal <d>.
Found 1-bit register for signal .
Found 1-bit register for signal <c>.
Summary:
 inferred 3 D-type flip-flop(s).
Unit <non_blocking> synthesized.

Timing Constraints

- Used to guide the synthesis tools
- Example 32-bit counter - no constraints (speed grades -4 and -5)

=====

Advanced HDL Synthesis Report

Macro Statistics

| | |
|-------------------|-----|
| # Counters | : 1 |
| 32-bit up counter | : 1 |

=====

Timing Summary:

Speed Grade: -4

Minimum period: **6.680ns** (Maximum Frequency: 149.703MHz)

Minimum input arrival time before clock: No path found

Maximum output required time after clock: 8.094ns

Maximum combinational path delay: No path found

=====

Speed Grade: -5

Minimum period: **5.767ns** (Maximum Frequency: 173.400MHz)

Adding timing constraint

- Add to UCF file:
 - NET "clk" PERIOD = 6ns HIGH 50%;

WARNING:Par:62 - Your design did not meet timing.

```
-----  
Constraint | Check | Worst Case | Best Case | Timing |Timing | Slack | Achievable |  
Errors | Score  
-----  
* NET "clk_BUFGP/IBUFG" PERIOD = 6 ns HIGH | SETUP| -0.456ns | 6.456ns | 9| 1430  
50% | HOLD | 2.432ns | | 0| 0  
-----
```

1 constraint not met.

Try relaxing constraint

- `NET "clk" PERIOD = 6.5ns HIGH 50%;`

| Constraint | Check Errors | Worst Case Score | Best Case | Timing | Timing | Slack | Achievable |
|--|-----------------|---------------------|-----------|--------|---------|---------|------------|
| NET "clk_BUFPG/IBUFG" PERIOD = 6.5 ns HIGH 50% | | | | SETUP | 0.203ns | 6.297ns | 0 |
| | | | | HOLD | 2.280ns | | 0 |

All constraints were met.

- Also see Timing Constraint User Guide
- Examples:

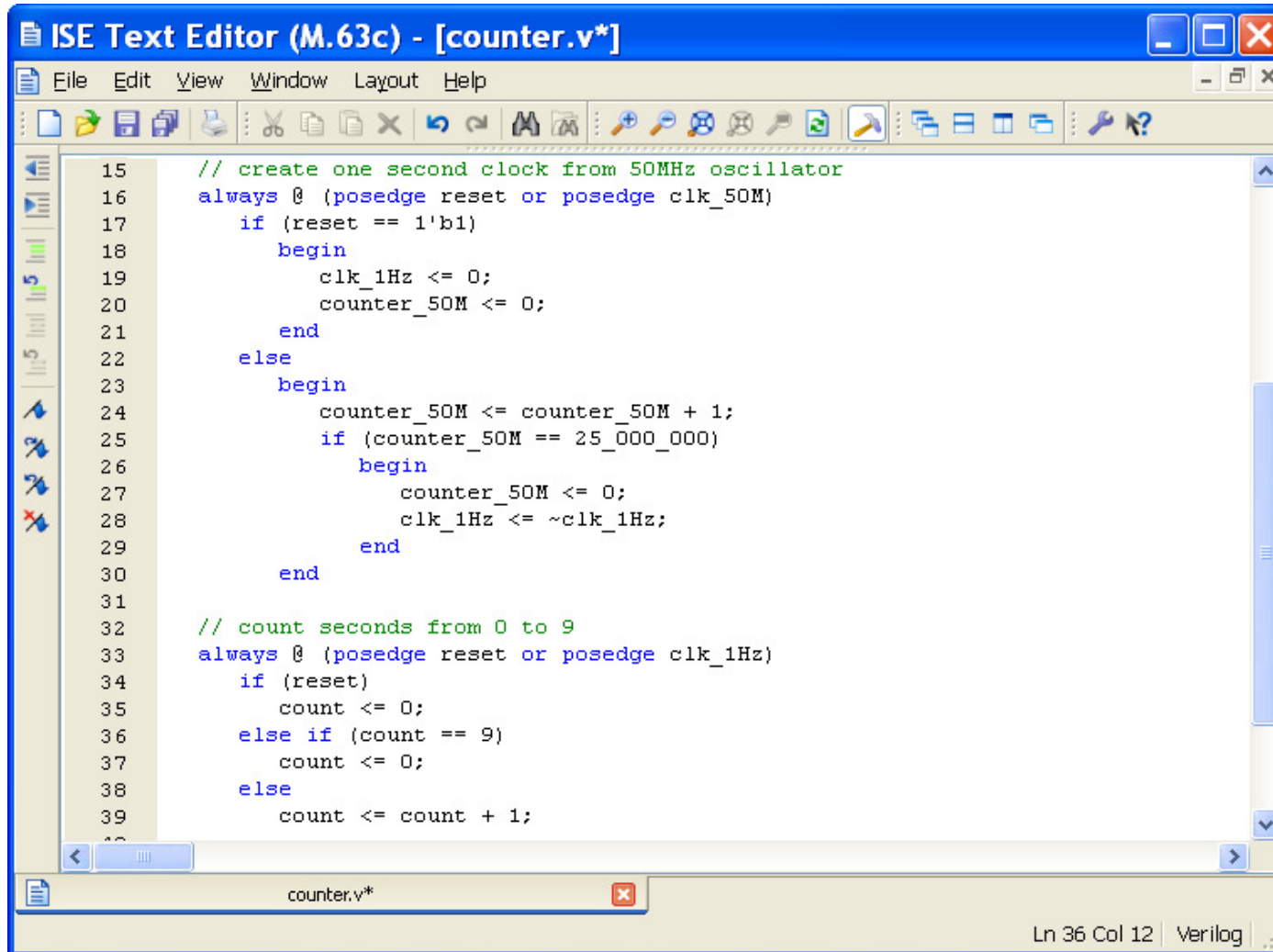
`NET 'abc' OFFSET = OUT xx ns AFTER 'clk';`

`NET 'def' OFFSET = IN xx ns BEFORE 'clk';`

Clock Skew

- The difference between the time a **clock** signal arrives at the source flip-flop in a path and the time it arrives at the destination flip-flop.
 - Misalignment of clock edges
 - Degrades (reduces) time for flip-flop to flip-flop timing
- Can be caused by different things but wire interconnect delays are the main cause inside FPGAs
 - There are fast dedicated clock circuits and slower data paths

Old method of deriving slower clocks

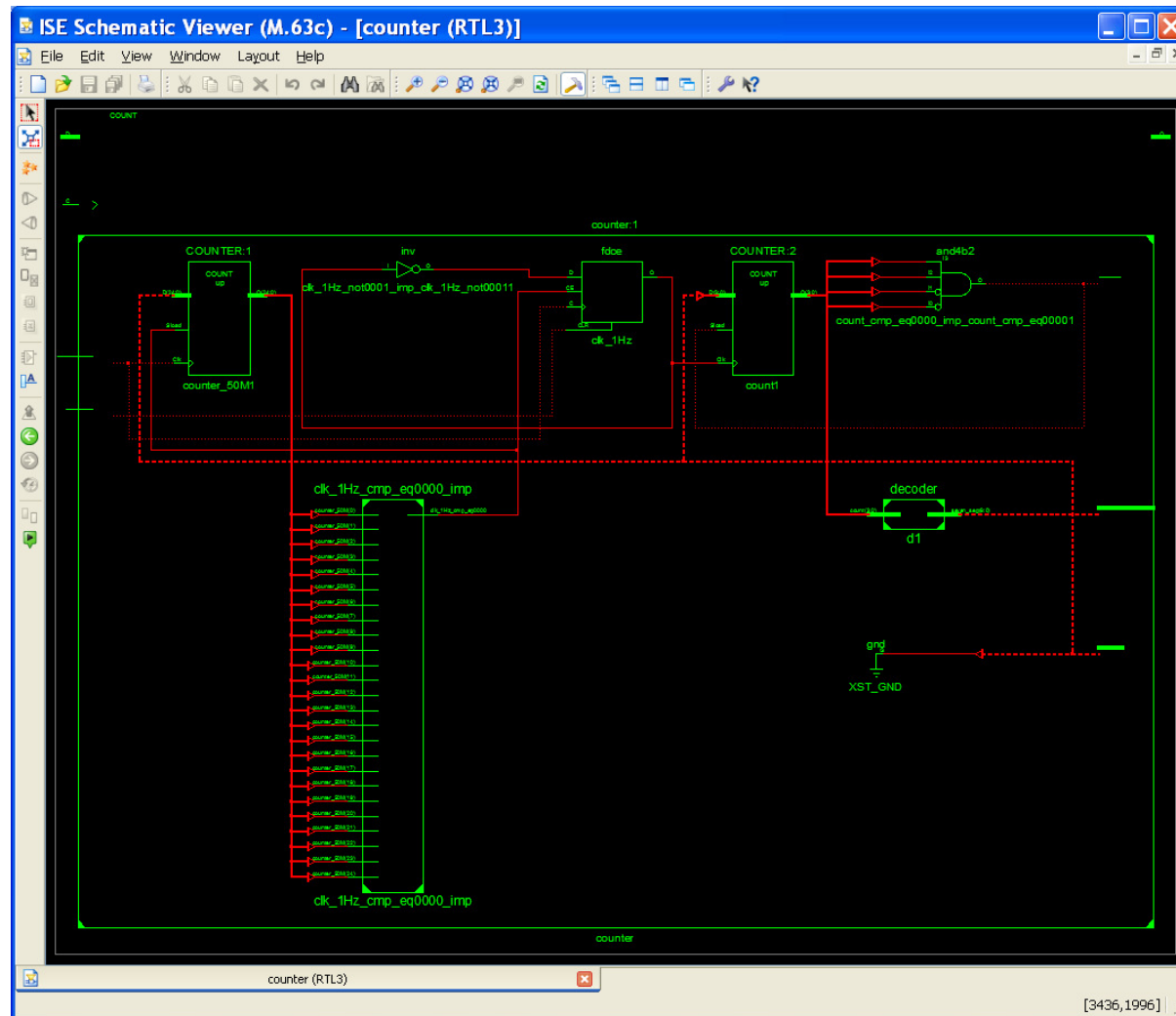
A screenshot of the ISE Text Editor window titled "ISE Text Editor (M.63c) - [counter.v*]". The window has a menu bar (File, Edit, View, Window, Layout, Help) and a toolbar with various icons. The main text area displays Verilog code for a counter module. The code is as follows:

```
15 // create one second clock from 50MHz oscillator
16 always @ (posedge reset or posedge clk_50M)
17     if (reset == 1'b1)
18         begin
19             clk_1Hz <= 0;
20             counter_50M <= 0;
21         end
22     else
23         begin
24             counter_50M <= counter_50M + 1;
25             if (counter_50M == 25_000_000)
26                 begin
27                     counter_50M <= 0;
28                     clk_1Hz <= ~clk_1Hz;
29                 end
30         end
31
32 // count seconds from 0 to 9
33 always @ (posedge reset or posedge clk_1Hz)
34     if (reset)
35         count <= 0;
36     else if (count == 9)
37         count <= 0;
38     else
39         count <= count + 1;
```

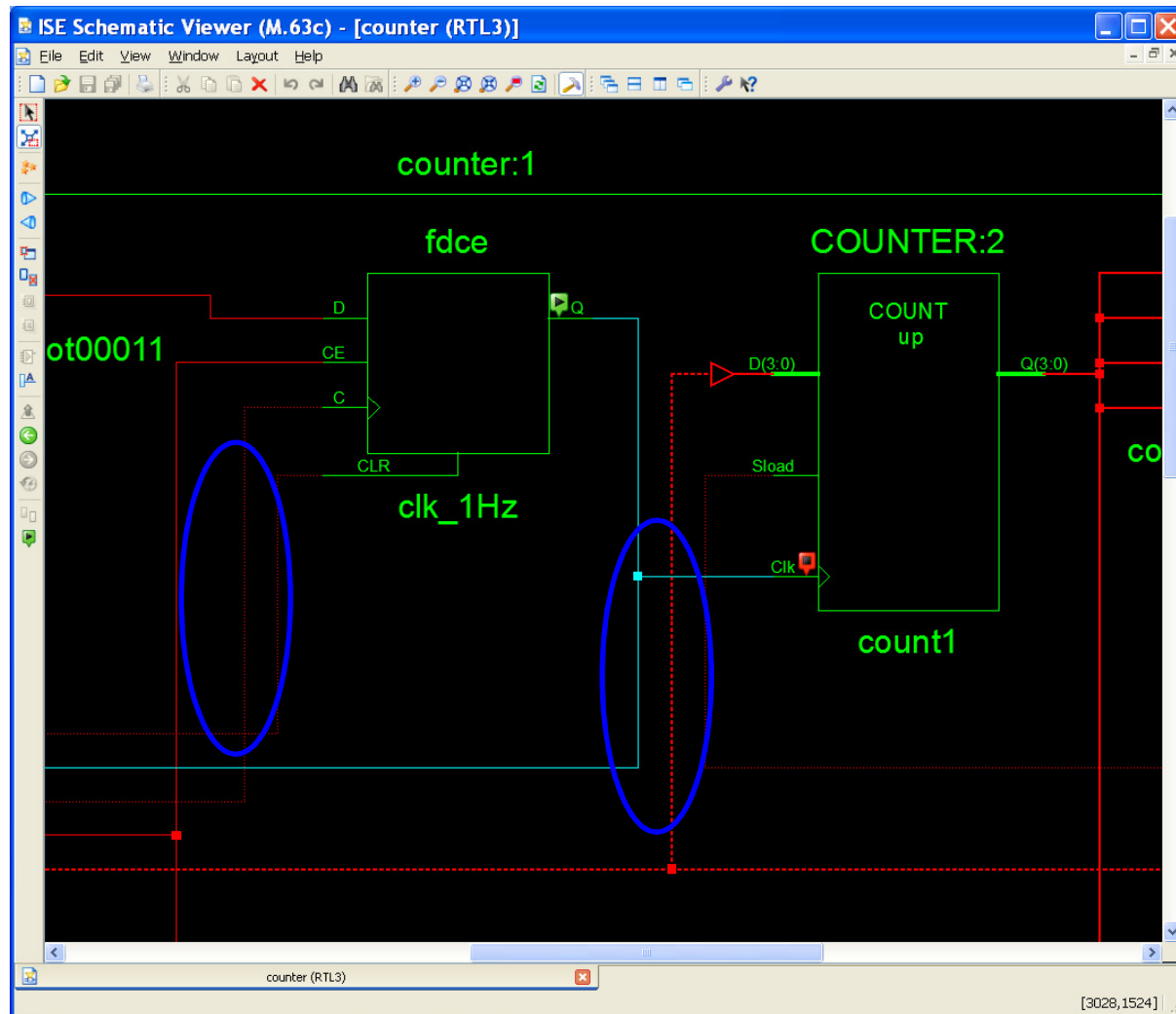
The status bar at the bottom right shows "Ln 36 Col 12" and "Verilog".

WARNING:Route:455 - CLK Net:clk_1Hz may have excessive skew because 0 CLK pins and 1 NON_CLK pins failed to route using a CLK template.

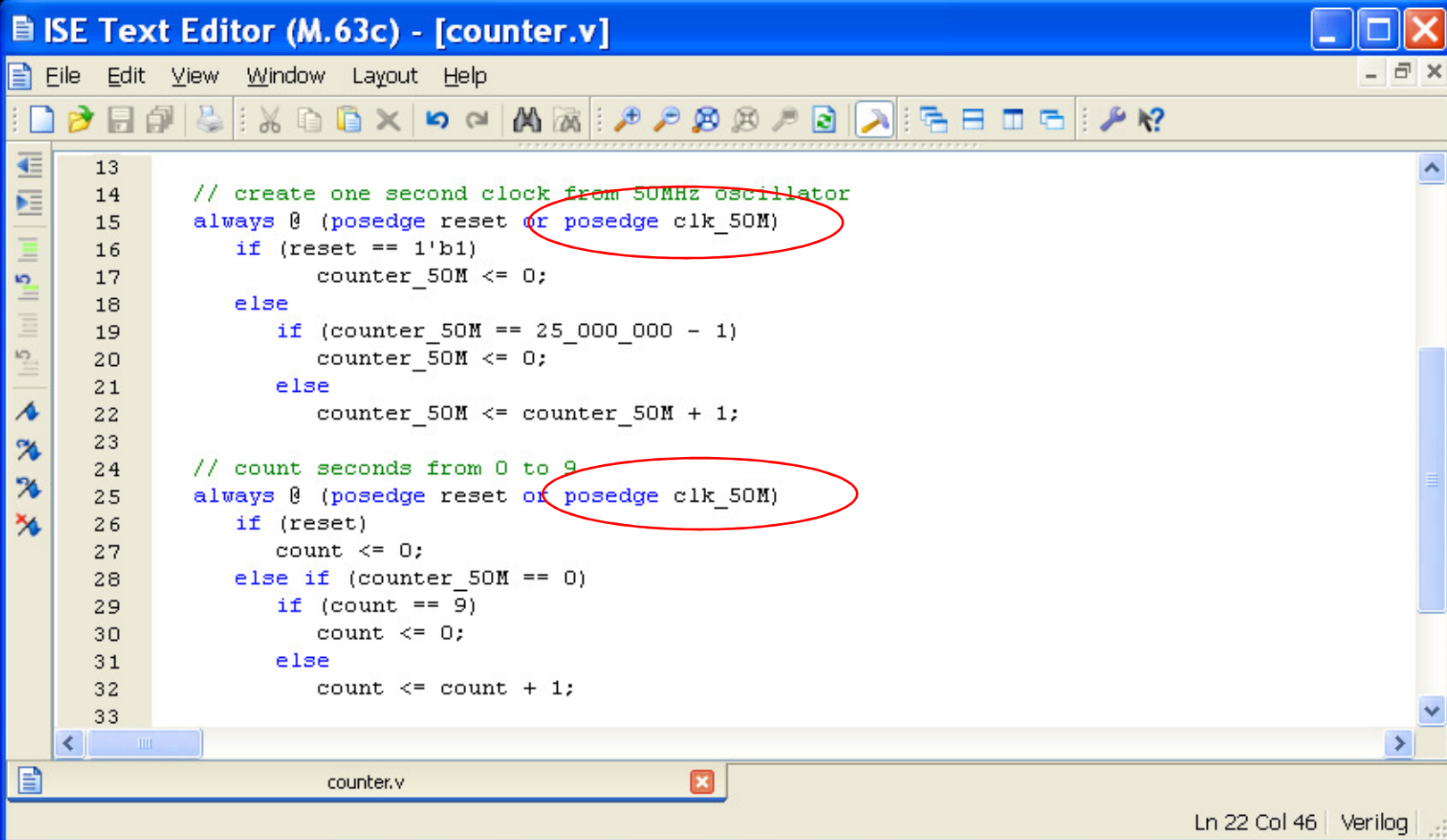
Synthesis uses two clock signals



Two clock signals – not good!



Modify to only use one clock signal



ISE Text Editor (M.63c) - [counter.v]

```
13
14 // create one second clock from 50MHz oscillator
15 always @ (posedge reset or posedge clk_50M)
16     if (reset == 1'b1)
17         counter_50M <= 0;
18     else
19         if (counter_50M == 25_000_000 - 1)
20             counter_50M <= 0;
21         else
22             counter_50M <= counter_50M + 1;
23
24 // count seconds from 0 to 9
25 always @ (posedge reset or posedge clk_50M)
26     if (reset)
27         count <= 0;
28     else if (counter_50M == 0)
29         if (count == 9)
30             count <= 0;
31         else
32             count <= count + 1;
33
```

counter.v

Ln 22 Col 46 Verilog

Clock signals use dedicated lines

The screenshot displays the Xilinx FPGA Editor interface for a project named 'counter_verilog\counter.ncd'. The main workspace, labeled 'Array1', shows a grid of logic blocks with a clock signal (red line) and data signals (cyan lines) routed through the array. A 'List1' window on the right shows a table of nets, and a 'World1' window at the bottom right shows a 3D view of the routing.

List1

| | Name | Fanout | Max Pin Del | Hilited |
|----|--------------|--------|-------------|----------|
| 1 | clk_50M_BUF | 15 | ? | red |
| 2 | clk_50M_BUF | 1 | ? | no color |
| 3 | count<0> | 12 | ? | no color |
| 4 | count<1> | 11 | ? | no color |
| 5 | count<2> | 11 | ? | no color |
| 6 | count<3> | 10 | ? | no color |
| 7 | counter_50M< | 2 | ? | no color |
| 8 | counter_50M< | 2 | ? | no color |
| 9 | counter_50M< | 2 | ? | no color |
| 10 | counter_50M< | 2 | ? | no color |
| 11 | counter_50M< | 2 | ? | no color |
| 12 | counter_50M< | 2 | ? | no color |
| 13 | counter_50M< | 2 | ? | no color |
| 14 | counter_50M< | 2 | ? | no color |
| 15 | counter_50M< | 2 | ? | no color |
| 16 | counter_50M< | 2 | ? | no color |

World1

net "clk_50M_BUFGP/IBUFG"

For Help, press F1

xc3s200-4R256 No Logic Changes Wild Cards

Clock drives all flip-flops

Xilinx FPGA Editor - H:\ee574\counter_verilog\counter.ncd

File Edit View Tools Window Help

Array1

List1

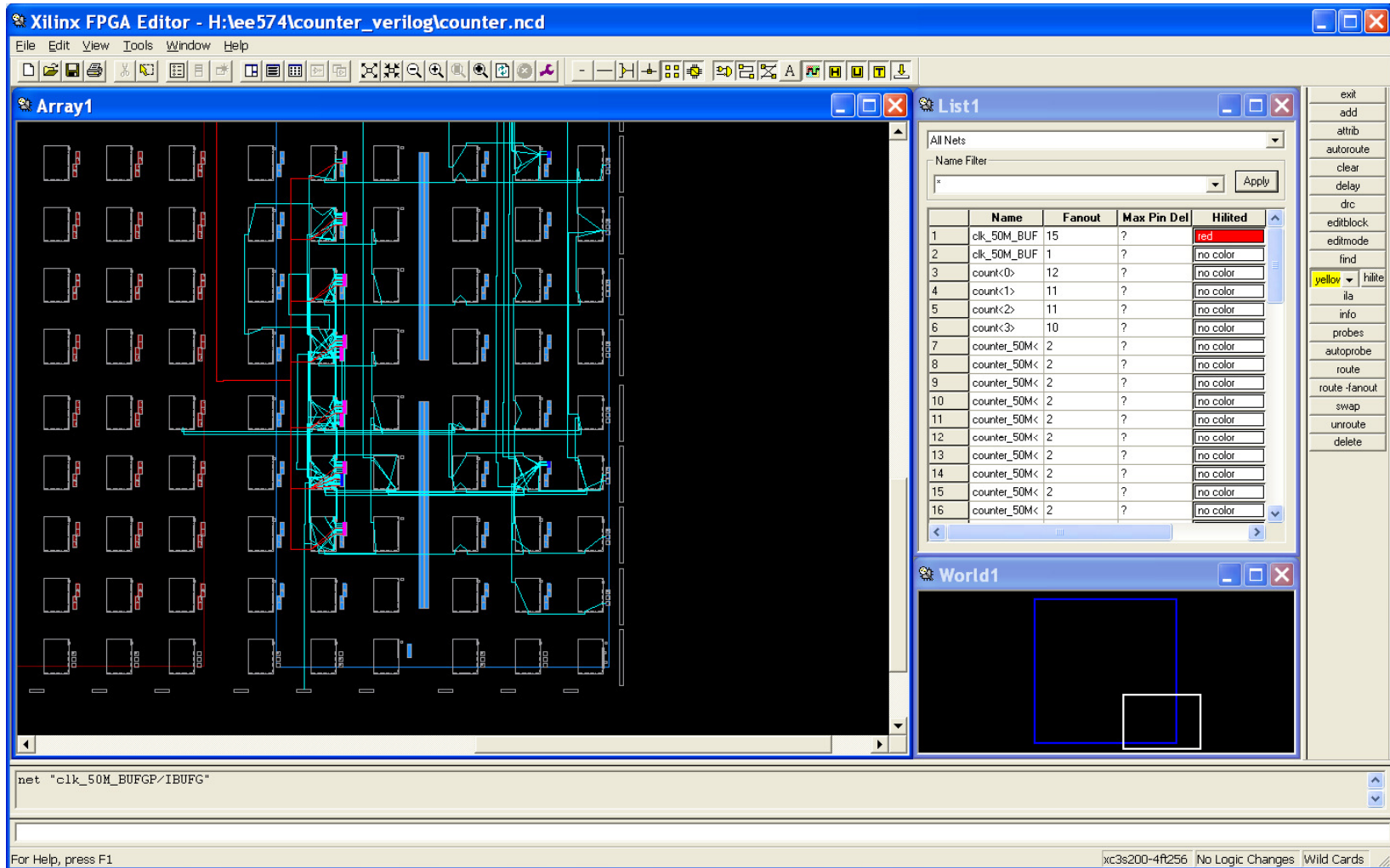
| | Name | Fanout | Max Pin Del | Hilited |
|----|--------------|--------|-------------|----------|
| 1 | clk_50M_BUF | 15 | ? | red |
| 2 | clk_50M_BUF | 1 | ? | no color |
| 3 | count<0> | 12 | ? | no color |
| 4 | count<1> | 11 | ? | no color |
| 5 | count<2> | 11 | ? | no color |
| 6 | count<3> | 10 | ? | no color |
| 7 | counter_50M< | 2 | ? | no color |
| 8 | counter_50M< | 2 | ? | no color |
| 9 | counter_50M< | 2 | ? | no color |
| 10 | counter_50M< | 2 | ? | no color |
| 11 | counter_50M< | 2 | ? | no color |
| 12 | counter_50M< | 2 | ? | no color |
| 13 | counter_50M< | 2 | ? | no color |
| 14 | counter_50M< | 2 | ? | no color |
| 15 | counter_50M< | 2 | ? | no color |
| 16 | counter_50M< | 2 | ? | no color |

World1

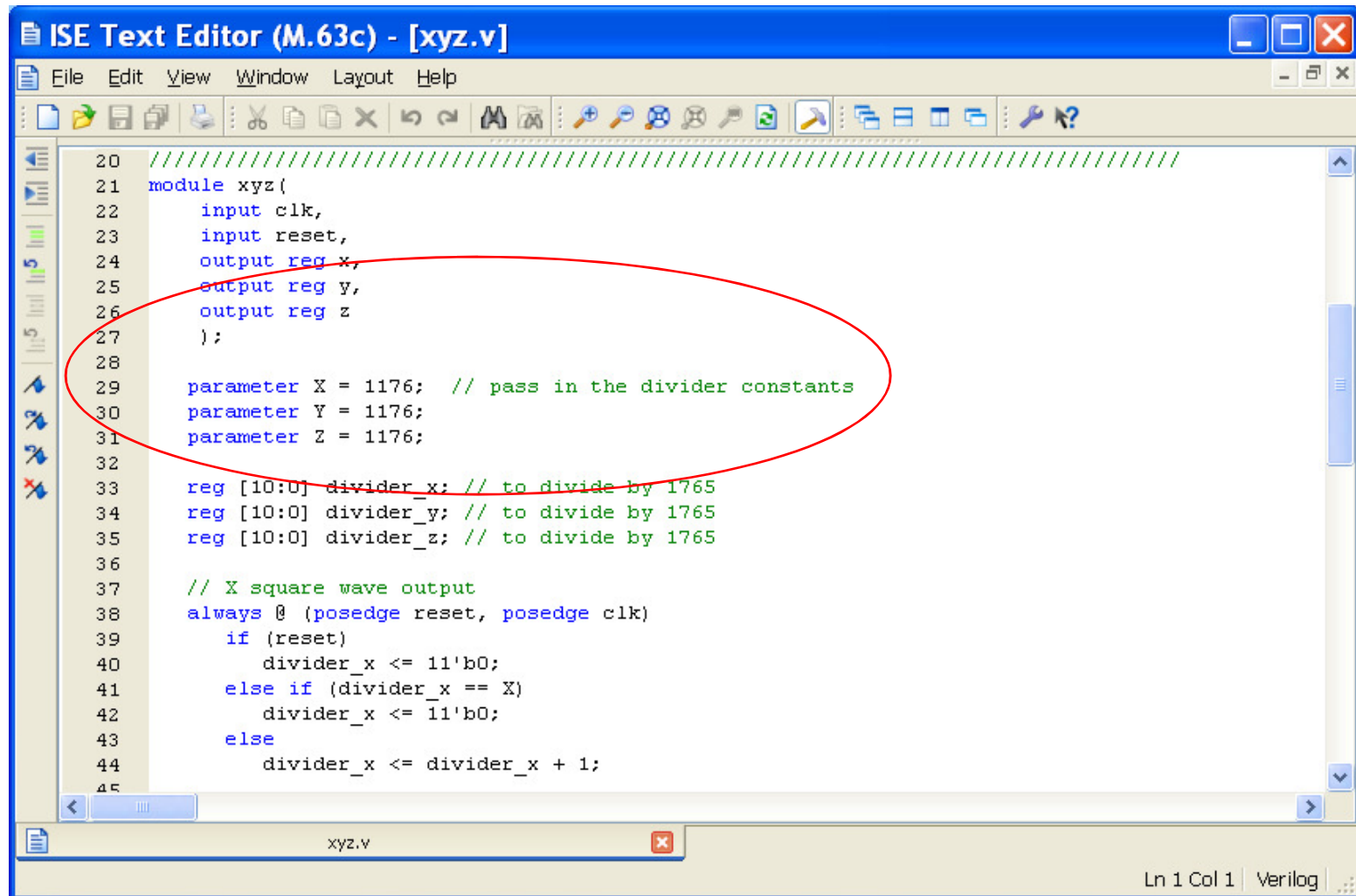
net "clk_50M_BUFGP/IBUFG"

For Help, press F1

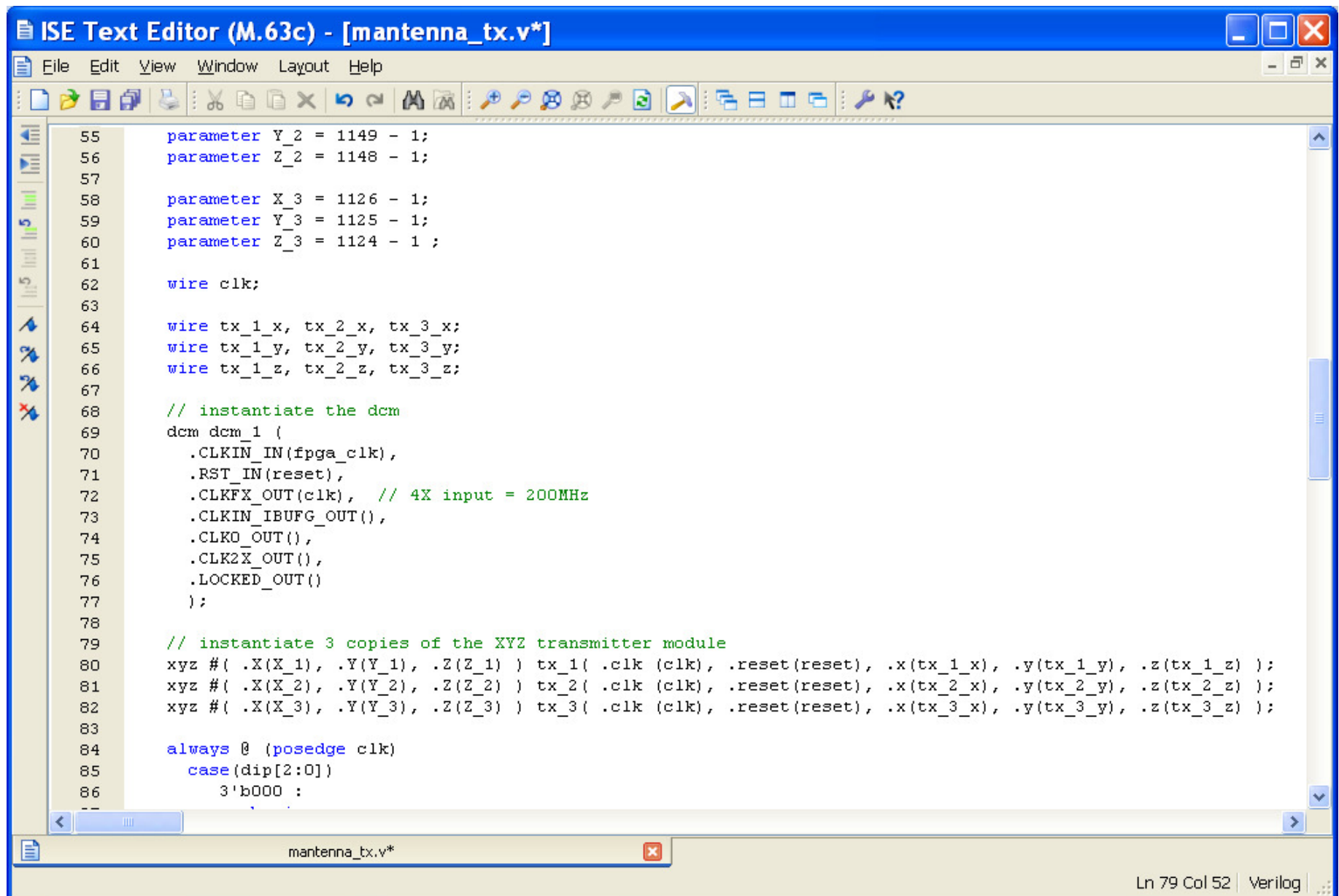
xc3s200-4ft256 No Logic Changes Wild Cards



Verilog Lower Level Module with Parameters

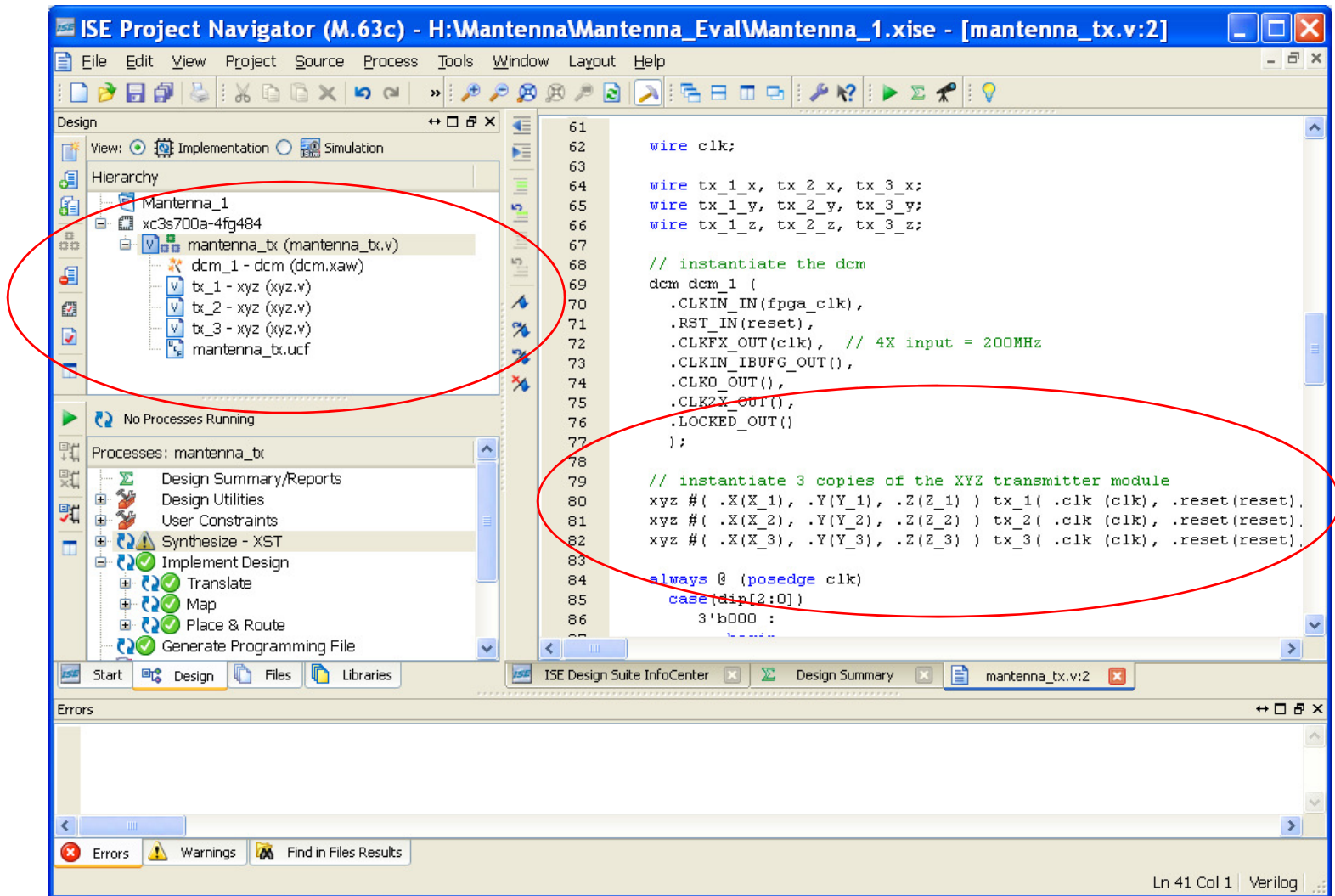


```
20 ///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
21 module xyz(
22     input clk,
23     input reset,
24     output reg x,
25     output reg y,
26     output reg z
27 );
28
29     parameter X = 1176; // pass in the divider constants
30     parameter Y = 1176;
31     parameter Z = 1176;
32
33     reg [10:0] divider_x; // to divide by 1765
34     reg [10:0] divider_y; // to divide by 1765
35     reg [10:0] divider_z; // to divide by 1765
36
37     // X square wave output
38     always @ (posedge reset, posedge clk)
39         if (reset)
40             divider_x <= 11'b0;
41         else if (divider_x == X)
42             divider_x <= 11'b0;
43         else
44             divider_x <= divider_x + 1;
45
```

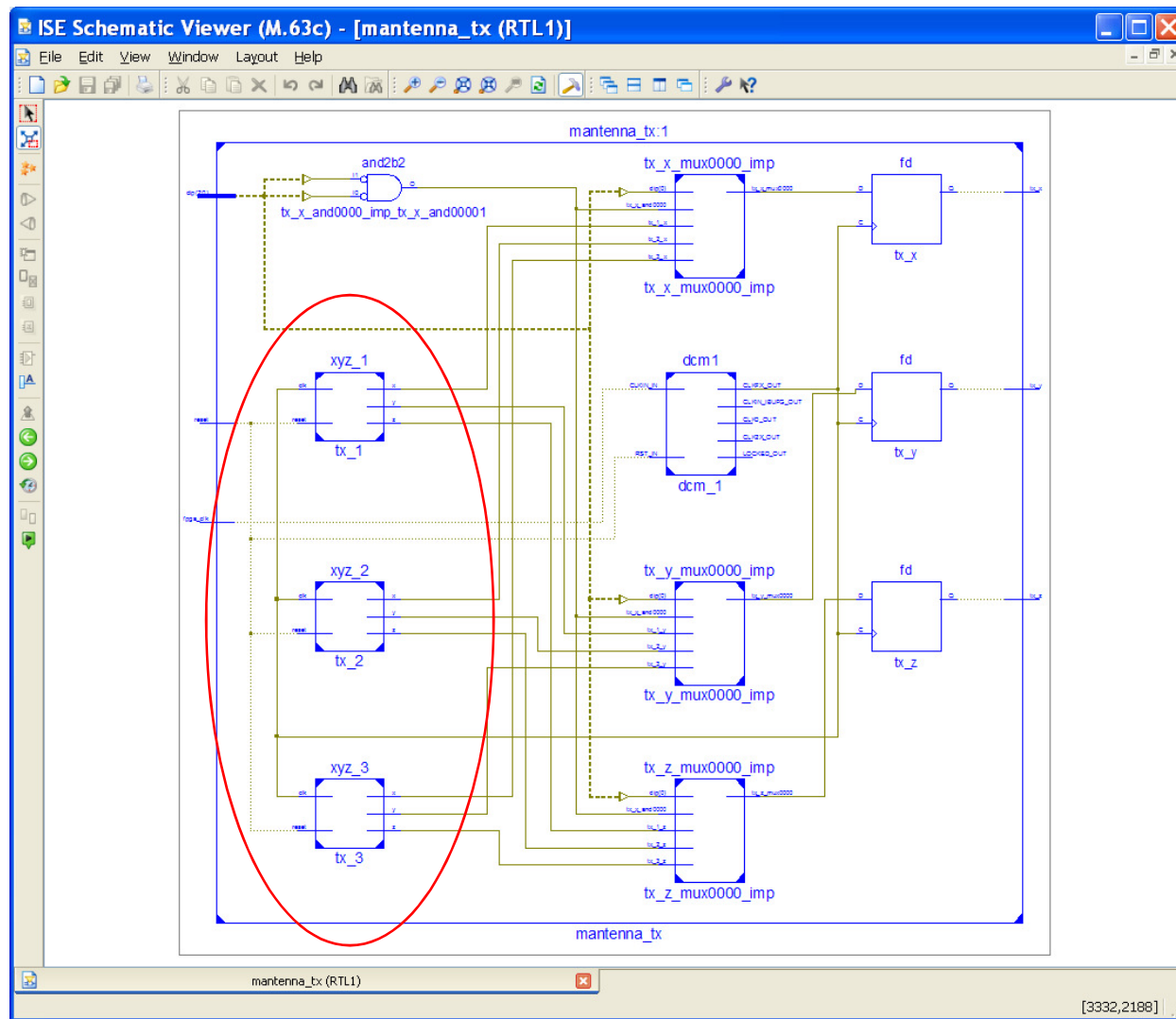


```
55     parameter Y_2 = 1149 - 1;
56     parameter Z_2 = 1148 - 1;
57
58     parameter X_3 = 1126 - 1;
59     parameter Y_3 = 1125 - 1;
60     parameter Z_3 = 1124 - 1 ;
61
62     wire clk;
63
64     wire tx_1_x, tx_2_x, tx_3_x;
65     wire tx_1_y, tx_2_y, tx_3_y;
66     wire tx_1_z, tx_2_z, tx_3_z;
67
68     // instantiate the dcm
69     dcm dcm_1 (
70         .CLKIN_IN(fpga_clk),
71         .RST_IN(reset),
72         .CLKFX_OUT(clk), // 4X input = 200MHz
73         .CLKIN_IBUFG_OUT(),
74         .CLKO_OUT(),
75         .CLK2X_OUT(),
76         .LOCKED_OUT()
77     );
78
79     // instantiate 3 copies of the XYZ transmitter module
80     xyz #( .X(X_1), .Y(Y_1), .Z(Z_1) ) tx_1( .clk(clk), .reset(reset), .x(tx_1_x), .y(tx_1_y), .z(tx_1_z) );
81     xyz #( .X(X_2), .Y(Y_2), .Z(Z_2) ) tx_2( .clk(clk), .reset(reset), .x(tx_2_x), .y(tx_2_y), .z(tx_2_z) );
82     xyz #( .X(X_3), .Y(Y_3), .Z(Z_3) ) tx_3( .clk(clk), .reset(reset), .x(tx_3_x), .y(tx_3_y), .z(tx_3_z) );
83
84     always @ (posedge clk)
85         case(dip[2:0])
86             3'b000 :
87
```

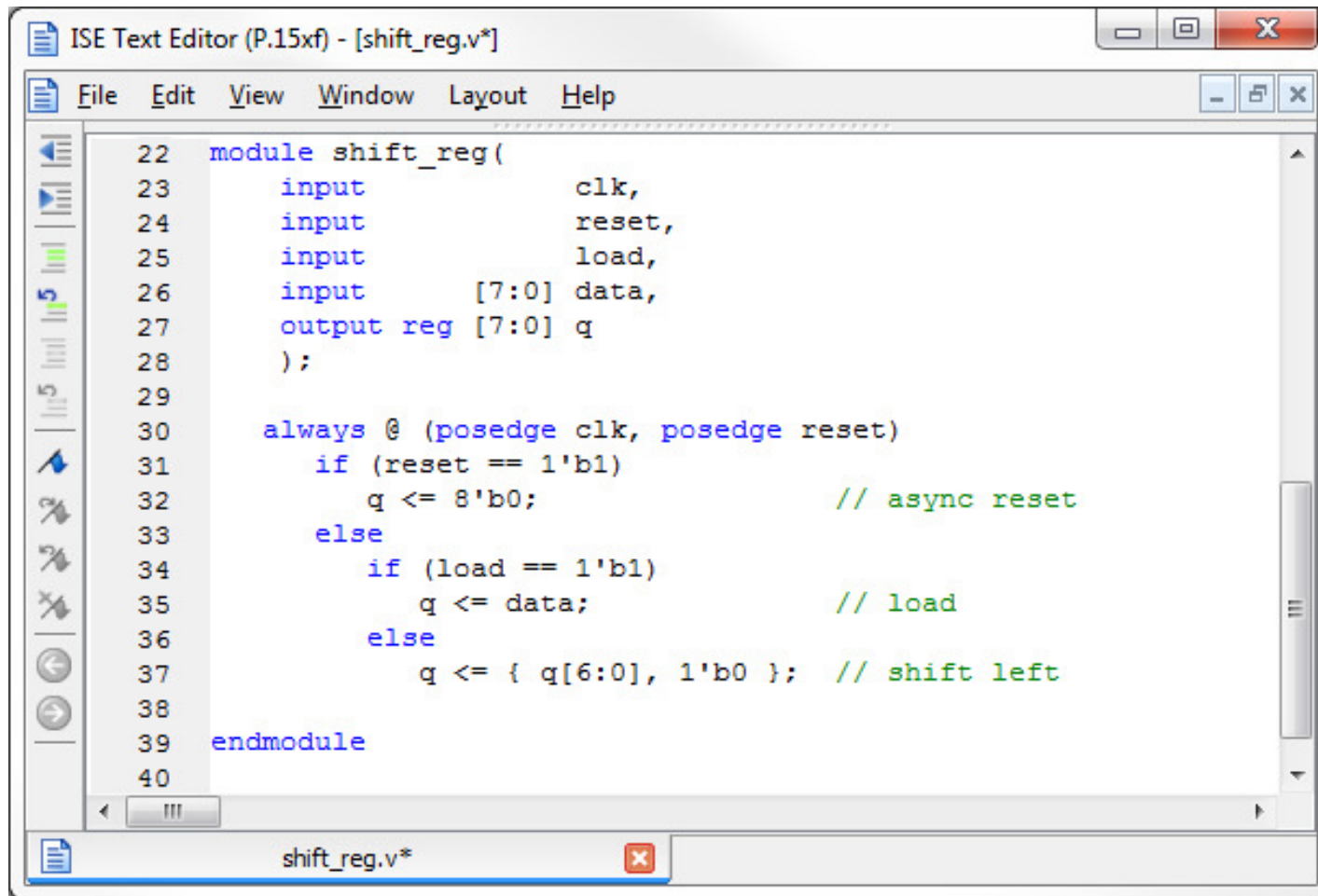
Three copies of XYZ module



RTL Schematic showing 3 modules



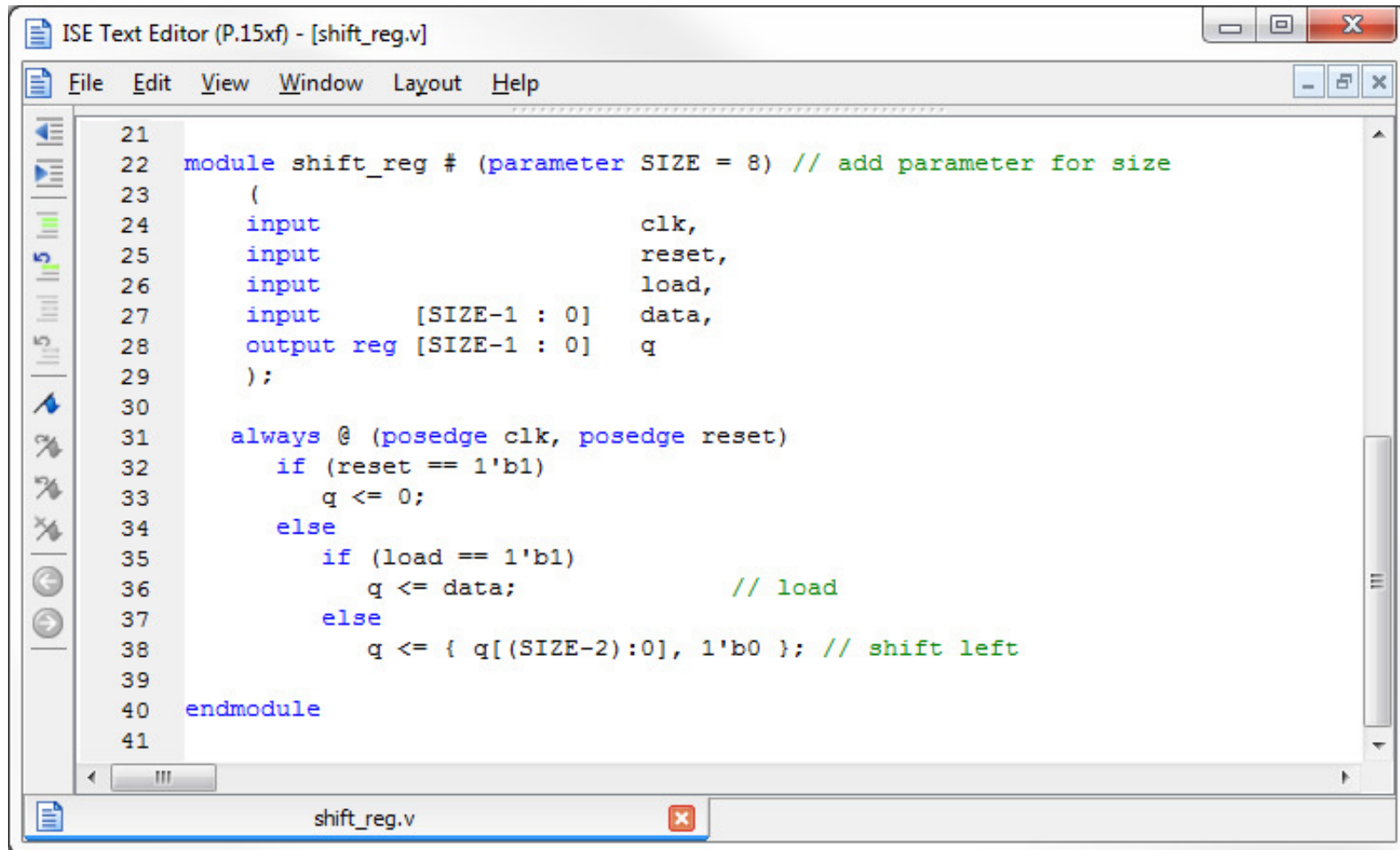
Shift Register – Fixed Size



The screenshot shows the ISE Text Editor window with the file name "shift_reg.v*" in the title bar. The menu bar includes File, Edit, View, Window, Layout, and Help. The code is written in Verilog and defines a module named "shift_reg". The module has four inputs: "clk", "reset", "load", and "data" (a 7-bit vector [7:0]). It has one output: "q", which is a 7-bit register [7:0]. The code includes an "always" block triggered by the positive edges of "clk" and "reset". Inside this block, there is an "if" statement for "reset == 1'b1" which sets "q" to 8'b0 (commented as "async reset"). An "else" branch contains another "if" statement for "load == 1'b1" which sets "q" to "data" (commented as "load"). A final "else" branch sets "q" to { q[6:0], 1'b0 } (commented as "shift left"). The module ends with "endmodule".

```
22 module shift_reg(  
23     input          clk,  
24     input          reset,  
25     input          load,  
26     input [7:0] data,  
27     output reg [7:0] q  
28 );  
29  
30 always @ (posedge clk, posedge reset)  
31     if (reset == 1'b1)  
32         q <= 8'b0;                // async reset  
33     else  
34         if (load == 1'b1)  
35             q <= data;            // load  
36         else  
37             q <= { q[6:0], 1'b0 }; // shift left  
38  
39 endmodule  
40
```

Modifying Component Size

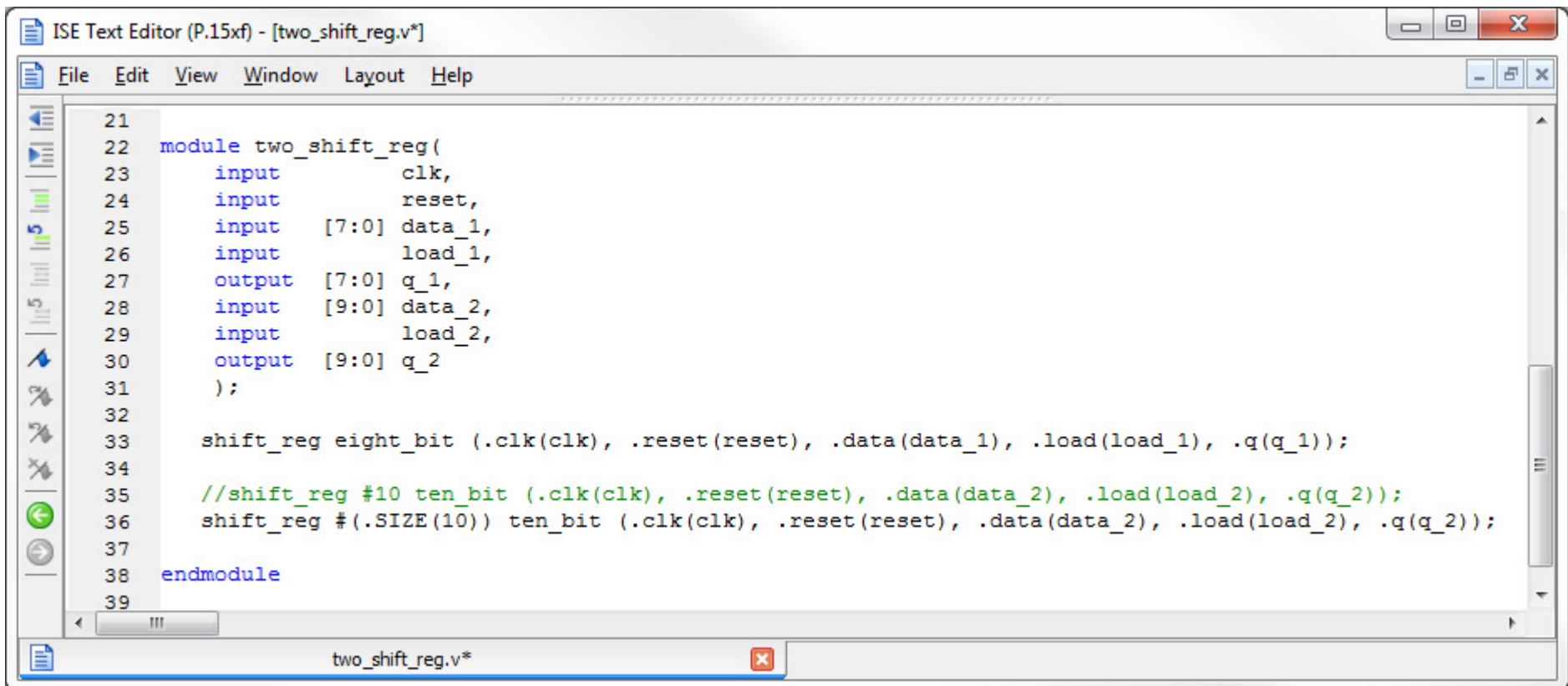


The screenshot shows the ISE Text Editor window titled "ISE Text Editor (P.15xf) - [shift_reg.v]". The menu bar includes File, Edit, View, Window, Layout, and Help. The code is as follows:

```
21
22 module shift_reg # (parameter SIZE = 8) // add parameter for size
23     (
24         input                clk,
25         input                reset,
26         input                load,
27         input [SIZE-1 : 0]   data,
28         output reg [SIZE-1 : 0] q
29     );
30
31     always @ (posedge clk, posedge reset)
32         if (reset == 1'b1)
33             q <= 0;
34         else
35             if (load == 1'b1)
36                 q <= data; // load
37             else
38                 q <= { q[(SIZE-2):0], 1'b0 }; // shift left
39
40 endmodule
41
```

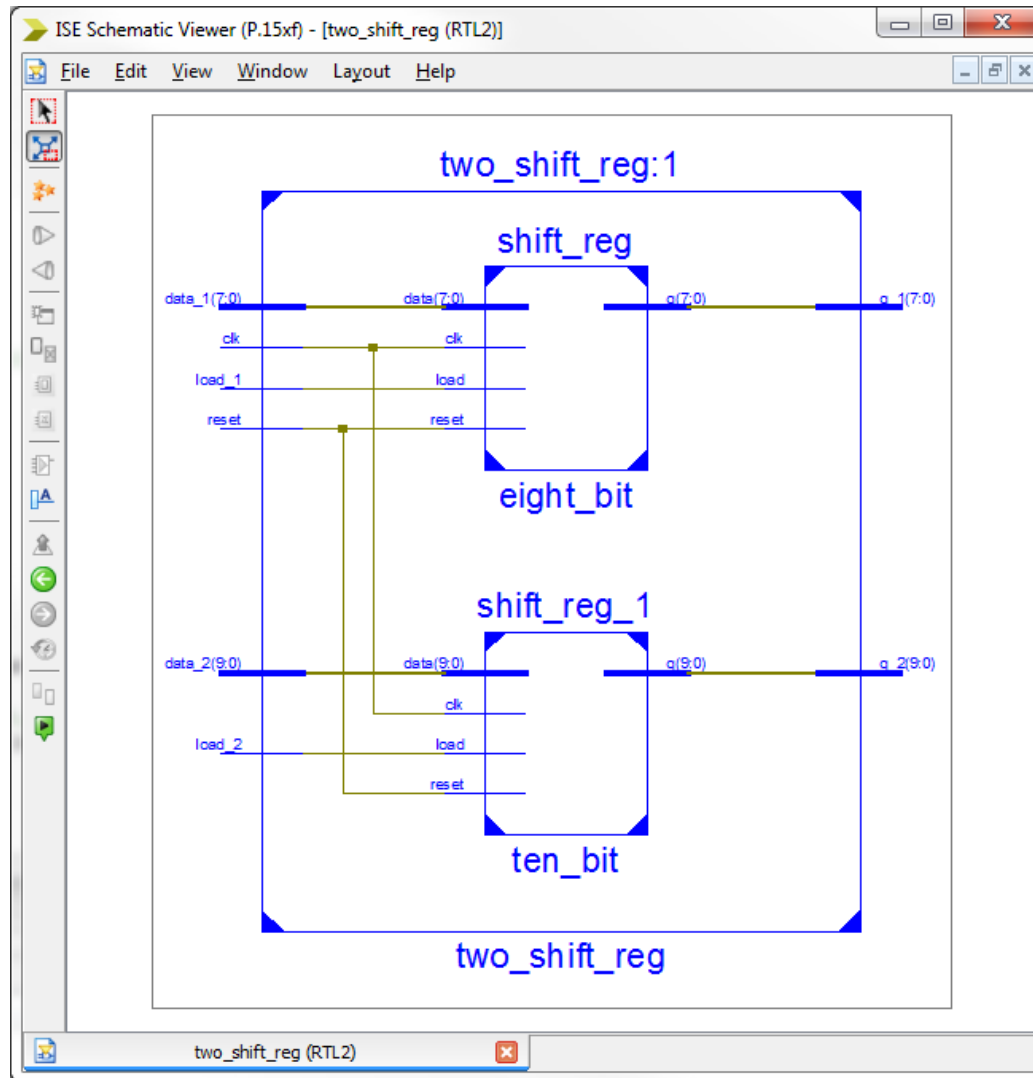
The status bar at the bottom shows the file name "shift_reg.v".

Making two copies – different size

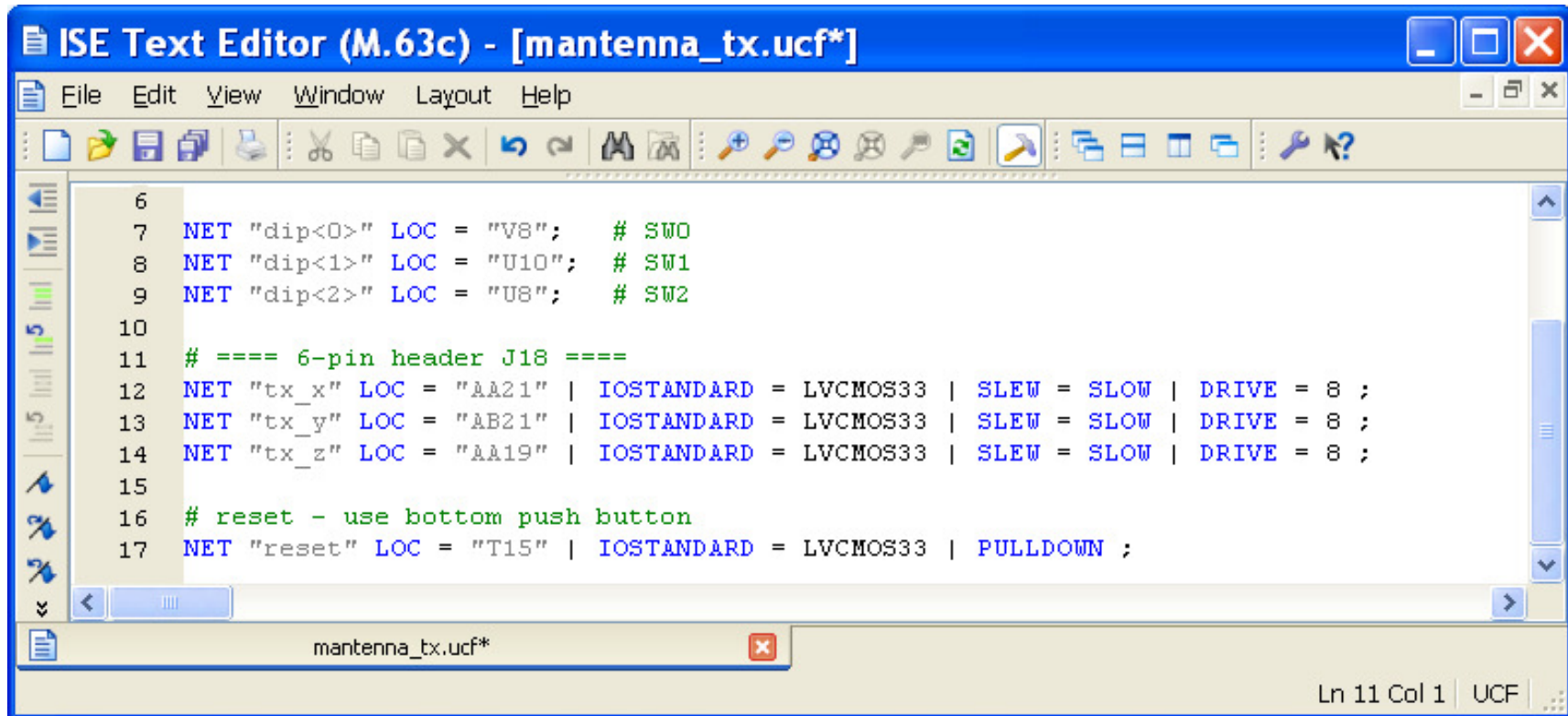


```
21
22 module two_shift_reg(
23     input      clk,
24     input      reset,
25     input [7:0] data_1,
26     input      load_1,
27     output [7:0] q_1,
28     input [9:0] data_2,
29     input      load_2,
30     output [9:0] q_2
31 );
32
33     shift_reg eight_bit (.clk(clk), .reset(reset), .data(data_1), .load(load_1), .q(q_1));
34
35     //shift_reg #10 ten_bit (.clk(clk), .reset(reset), .data(data_2), .load(load_2), .q(q_2));
36     shift_reg #(.SIZE(10)) ten_bit (.clk(clk), .reset(reset), .data(data_2), .load(load_2), .q(q_2));
37
38 endmodule
39
```

Schematic of two shift registers



(Misc) Using UCF to specify pin options



```
6
7 NET "dip<0>" LOC = "V8";    # SW0
8 NET "dip<1>" LOC = "U10";    # SW1
9 NET "dip<2>" LOC = "U8";    # SW2
10
11 # ==== 6-pin header J18 ====
12 NET "tx_x" LOC = "AA21" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
13 NET "tx_y" LOC = "AB21" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
14 NET "tx_z" LOC = "AA19" | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 8 ;
15
16 # reset - use bottom push button
17 NET "reset" LOC = "T15" | IOSTANDARD = LVCMOS33 | PULLDOWN ;
```

Ln 11 Col 1 | UCF

