FPGA – based Embedded System Desgin

CA #1

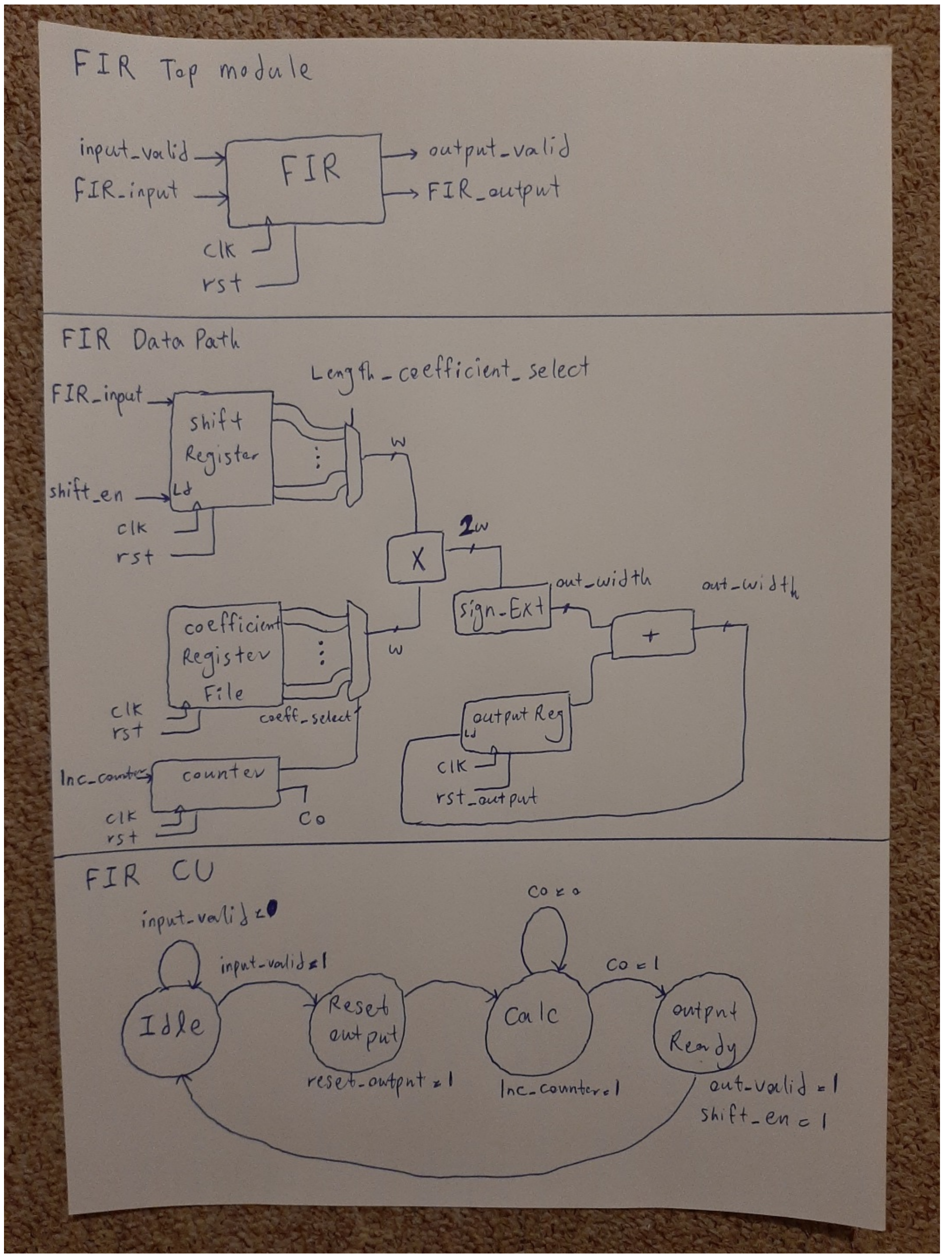
Name: Mohammad

Last name: Taghizadeh Givari

Student No: 810198373

Contents

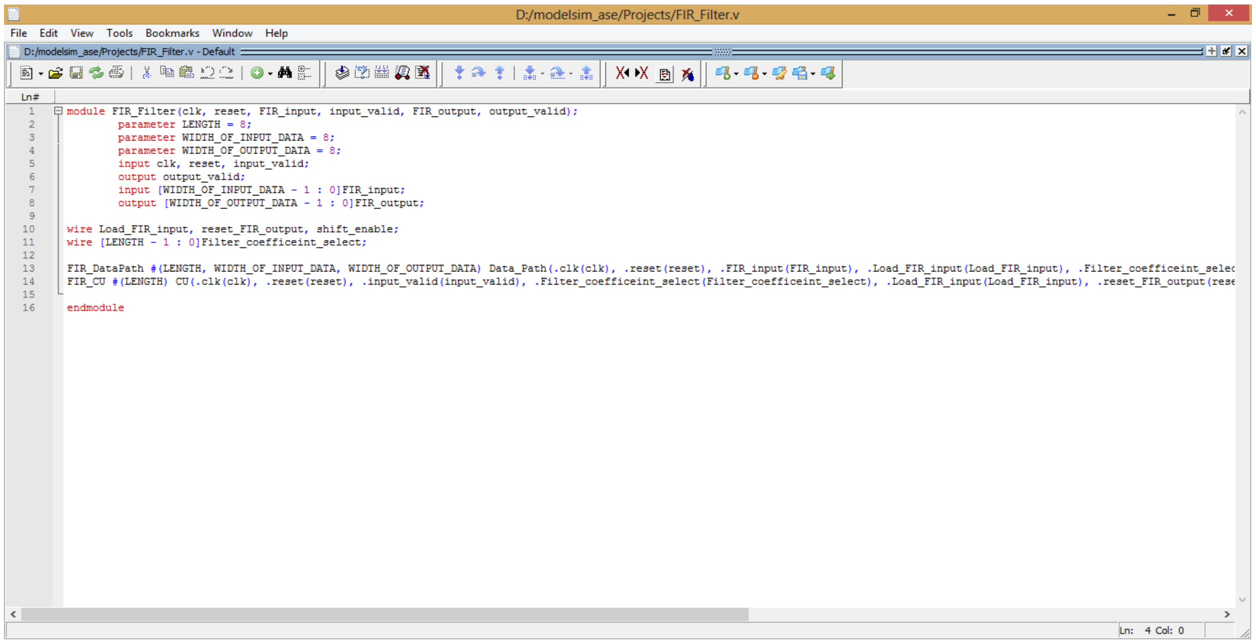
Introduction



With assertion of input_valid, calculation starts so at first we rest the previous output, then in a loop, we multiple b[k] by x[n – k]. k is determined with coefiicient_select which is the output of a counter which counts coefficients till the end. We need to measure summation of multiplications so we use a looped back adder which accumulates the multiplications and make the correct output.

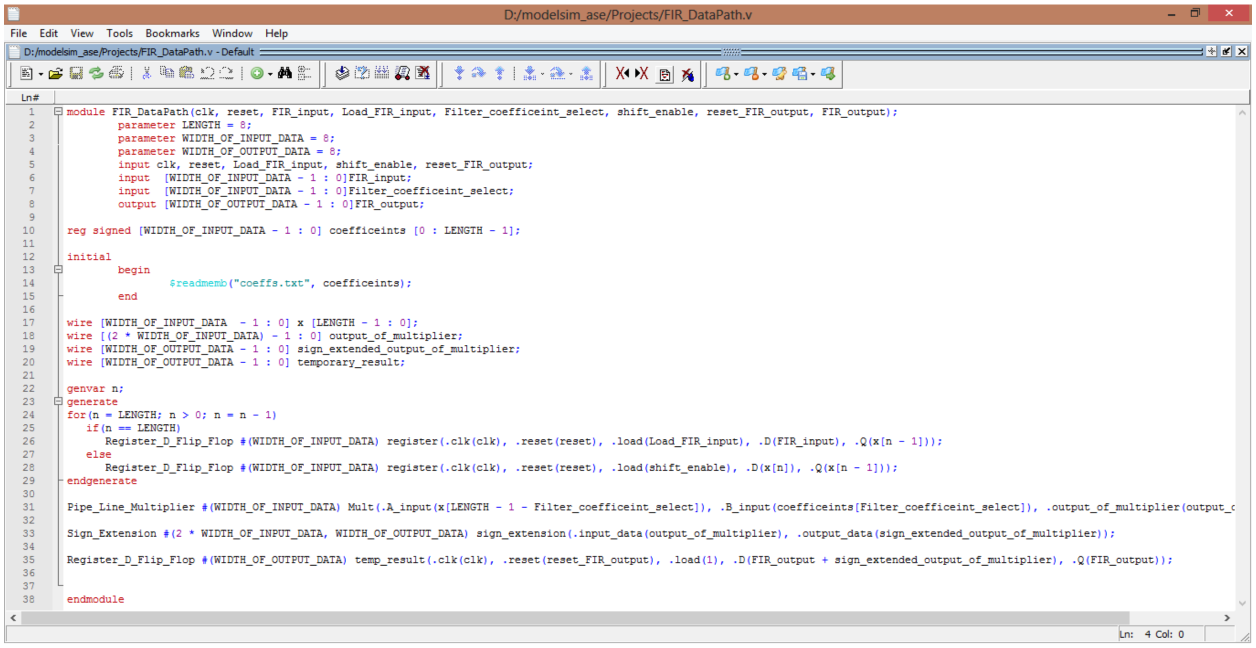
Verilog code of FIR Filter

1. Top Module



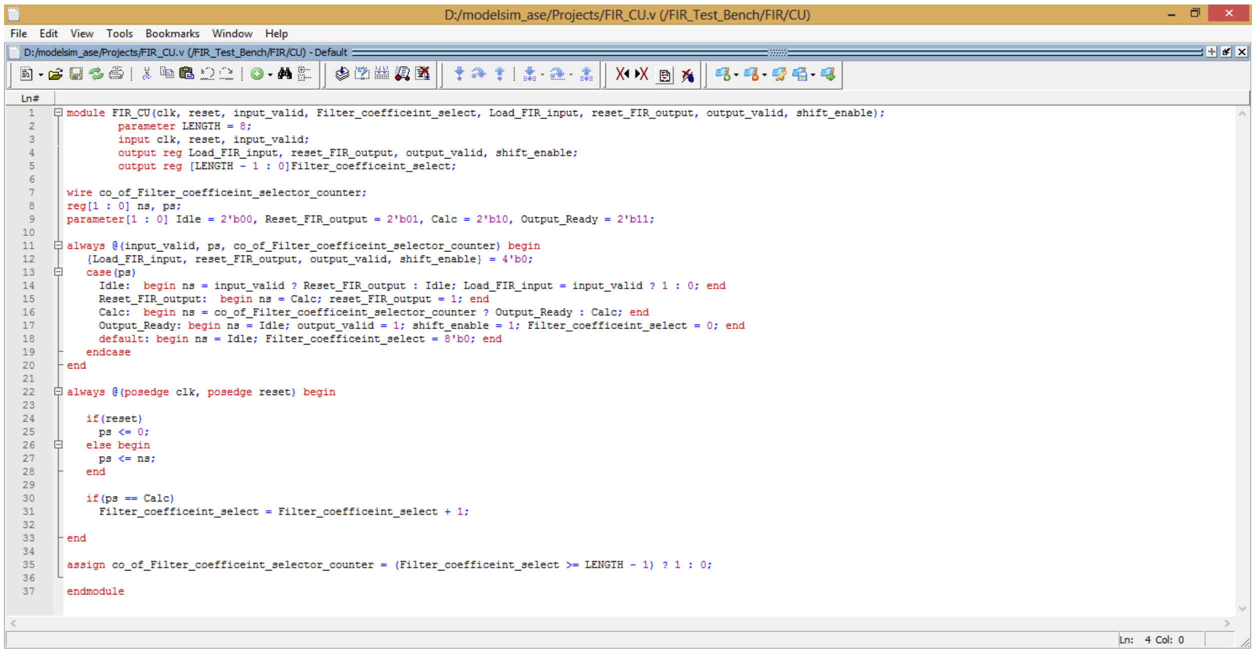
```verilog
module FIR_Filter(clk, reset, FIR_input, input_valid, FIR_output, output_valid);
        parameter LENGTH = 8;
        parameter WIDTH_OF_INPUT_DATA = 8;
        parameter WIDTH_OF_OUTPUT_DATA = 8;
        input clk, reset, input_valid;
        output output_valid;
        input [WIDTH_OF_INPUT_DATA - 1 : 0]FIR_input;
        output [WIDTH_OF_OUTPUT_DATA - 1 : 0]FIR_output;

wire Load_FIR_input, reset_FIR_output, shift_enable;
wire [LENGTH - 1 : 0]Filter_coefficeint_select;

FIR_DataPath #(LENGTH, WIDTH_OF_INPUT_DATA, WIDTH_OF_OUTPUT_DATA) Data_Path(.clk(clk), .reset(reset), .FIR_input(FIR_input), .Load_FIR_input(Load_FIR_input), .Filter_coefficeint_selec
FIR_CU #(LENGTH) CU(.clk(clk), .reset(reset), .input_valid(input_valid), .Filter_coefficeint_select(Filter_coefficeint_select), .Load_FIR_input(Load_FIR_input), .reset_FIR_output(rese

endmodule
```

2. Data Path



```verilog
module FIR_DataPath(clk, reset, FIR_input, Load_FIR_input, Filter_coefficeint_select, shift_enable, reset_FIR_output, FIR_output);
        parameter LENGTH = 8;
        parameter WIDTH_OF_INPUT_DATA = 8;
        parameter WIDTH_OF_OUTPUT_DATA = 8;
        input clk, reset, Load_FIR_input, shift_enable, reset_FIR_output;
        input  [WIDTH_OF_INPUT_DATA - 1 : 0]FIR_input;
        input  [WIDTH_OF_INPUT_DATA - 1 : 0]Filter_coefficeint_select;
        output [WIDTH_OF_OUTPUT_DATA - 1 : 0]FIR_output;

reg signed [WIDTH_OF_INPUT_DATA - 1 : 0] coefficeints [0 : LENGTH - 1];

initial
        begin
                $readmemb("coeffs.txt", coefficeints);
        end

wire [WIDTH_OF_INPUT_DATA  - 1 : 0] x [LENGTH - 1 : 0];
wire [(2 * WIDTH_OF_INPUT_DATA) - 1 : 0] output_of_multiplier;
wire [WIDTH_OF_OUTPUT_DATA - 1 : 0] sign_extended_output_of_multiplier;
wire [WIDTH_OF_OUTPUT_DATA - 1 : 0] temporary_result;

genvar n;
generate
for(n = LENGTH; n > 0; n = n - 1)
   if(n == LENGTH)
      Register_D_Flip_Flop #(WIDTH_OF_INPUT_DATA) register(.clk(clk), .reset(reset), .load(Load_FIR_input), .D(FIR_input), .Q(x[n - 1]));
   else
      Register_D_Flip_Flop #(WIDTH_OF_INPUT_DATA) register(.clk(clk), .reset(reset), .load(shift_enable), .D(x[n]), .Q(x[n - 1]));
endgenerate

Pipe_Line_Multiplier #(WIDTH_OF_INPUT_DATA) Mult(.A_input(x[LENGTH - 1 - Filter_coefficeint_select]), .B_input(coefficeints[Filter_coefficeint_select]), .output_of_multiplier(output_c

Sign_Extension #(2 * WIDTH_OF_INPUT_DATA, WIDTH_OF_OUTPUT_DATA) sign_extension(.input_data(output_of_multiplier), .output_data(sign_extended_output_of_multiplier));

Register_D_Flip_Flop #(WIDTH_OF_OUTPUT_DATA) temp_result(.clk(clk), .reset(reset_FIR_output), .load(1), .D(FIR_output + sign_extended_output_of_multiplier), .Q(FIR_output));


endmodule
```

3. Controller



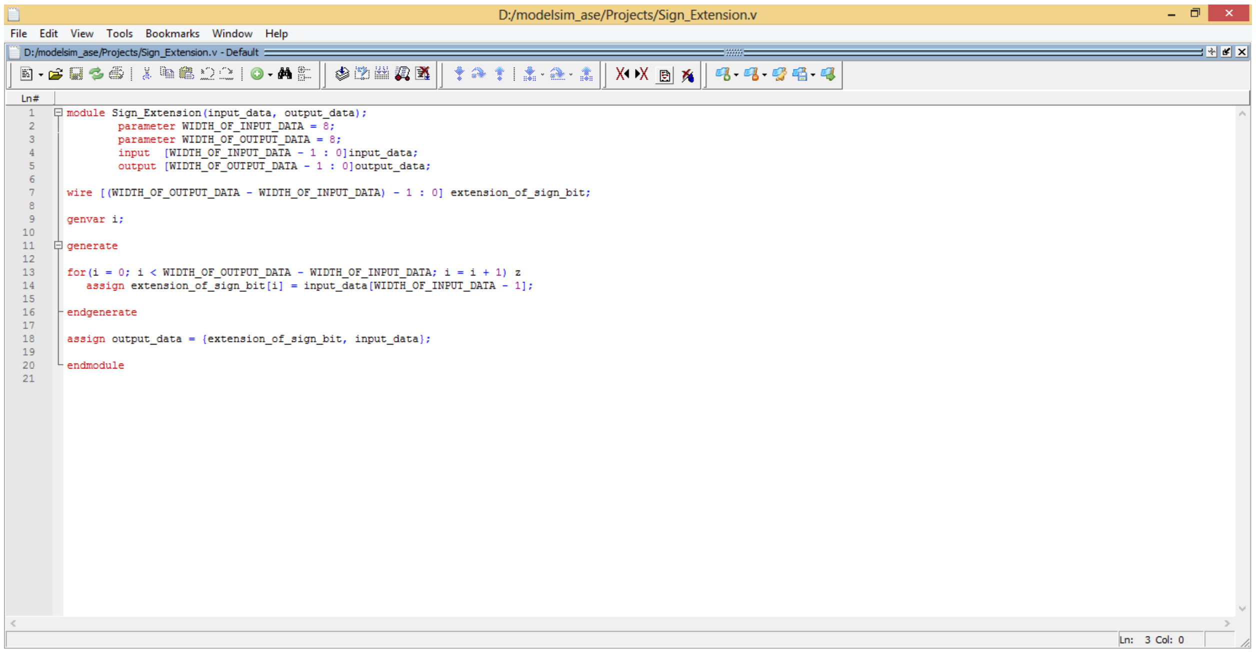
D:/modelsim_ase/Projects/FIR_CU.v (/FIR_Test_Bench/FIR/CU)

```
module FIR_CU(clk, reset, input_valid, Filter_coefficeint_select, Load_FIR_input, reset_FIR_output, output_valid, shift_enable);
        parameter LENGTH = 8;
        input clk, reset, input_valid;
        output reg Load_FIR_input, reset_FIR_output, output_valid, shift_enable;
        output reg [LENGTH - 1 : 0]Filter_coefficeint_select;

wire co_of_Filter_coefficeint_selector_counter;
reg[1 : 0] ns, ps;
parameter[1 : 0] Idle = 2'b00, Reset_FIR_output = 2'b01, Calc = 2'b10, Output_Ready = 2'b11;

always @(input_valid, ps, co_of_Filter_coefficeint_selector_counter) begin
   {Load_FIR_input, reset_FIR_output, output_valid, shift_enable} = 4'b0;
   case(ps)
     Idle:  begin ns = input_valid ? Reset_FIR_output : Idle; Load_FIR_input = input_valid ? 1 : 0; end
     Reset_FIR_output:  begin ns = Calc; reset_FIR_output = 1; end
     Calc:  begin ns = co_of_Filter_coefficeint_selector_counter ? Output_Ready : Calc; end
     Output_Ready: begin ns = Idle; output_valid = 1; shift_enable = 1; Filter_coefficeint_select = 0; end
     default: begin ns = Idle; Filter_coefficeint_select = 8'b0; end
   endcase
end

always @(posedge clk, posedge reset) begin

   if(reset)
     ps <= 0;
   else begin
     ps <= ns;
   end

   if(ps == Calc)
     Filter_coefficeint_select = Filter_coefficeint_select + 1;

end

assign co_of_Filter_coefficeint_selector_counter = (Filter_coefficeint_select >= LENGTH - 1) ? 1 : 0;

endmodule
```

Ln: 4 Col: 0

Side Modules

1. Sign Extension

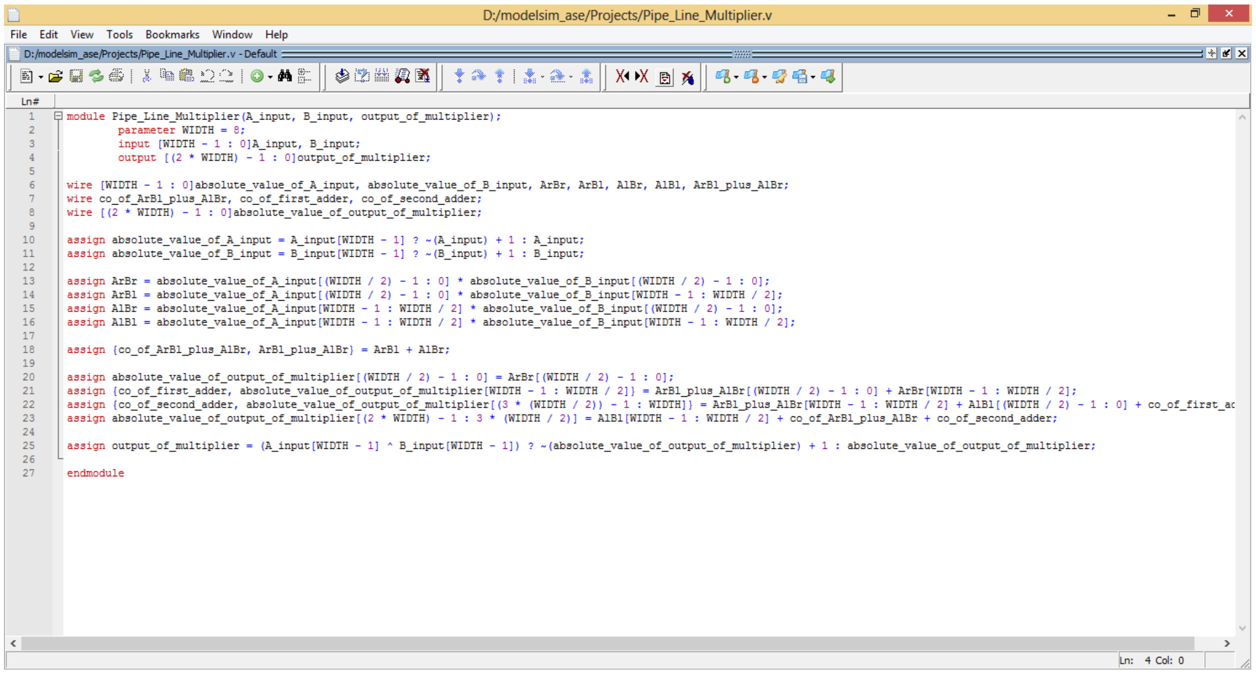


D:/modelsim_ase/Projects/Sign_Extension.v

```
module Sign_Extension(input_data, output_data);
        parameter WIDTH_OF_INPUT_DATA = 8;
        parameter WIDTH_OF_OUTPUT_DATA = 8;
        input  [WIDTH_OF_INPUT_DATA - 1 : 0]input_data;
        output [WIDTH_OF_OUTPUT_DATA - 1 : 0]output_data;

wire [(WIDTH_OF_OUTPUT_DATA - WIDTH_OF_INPUT_DATA) - 1 : 0] extension_of_sign_bit;

genvar i;

generate

for(i = 0; i < WIDTH_OF_OUTPUT_DATA - WIDTH_OF_INPUT_DATA; i = i + 1) z
   assign extension_of_sign_bit[i] = input_data[WIDTH_OF_INPUT_DATA - 1];

endgenerate

assign output_data = {extension_of_sign_bit, input_data};

endmodule
```

Ln: 3 Col: 0

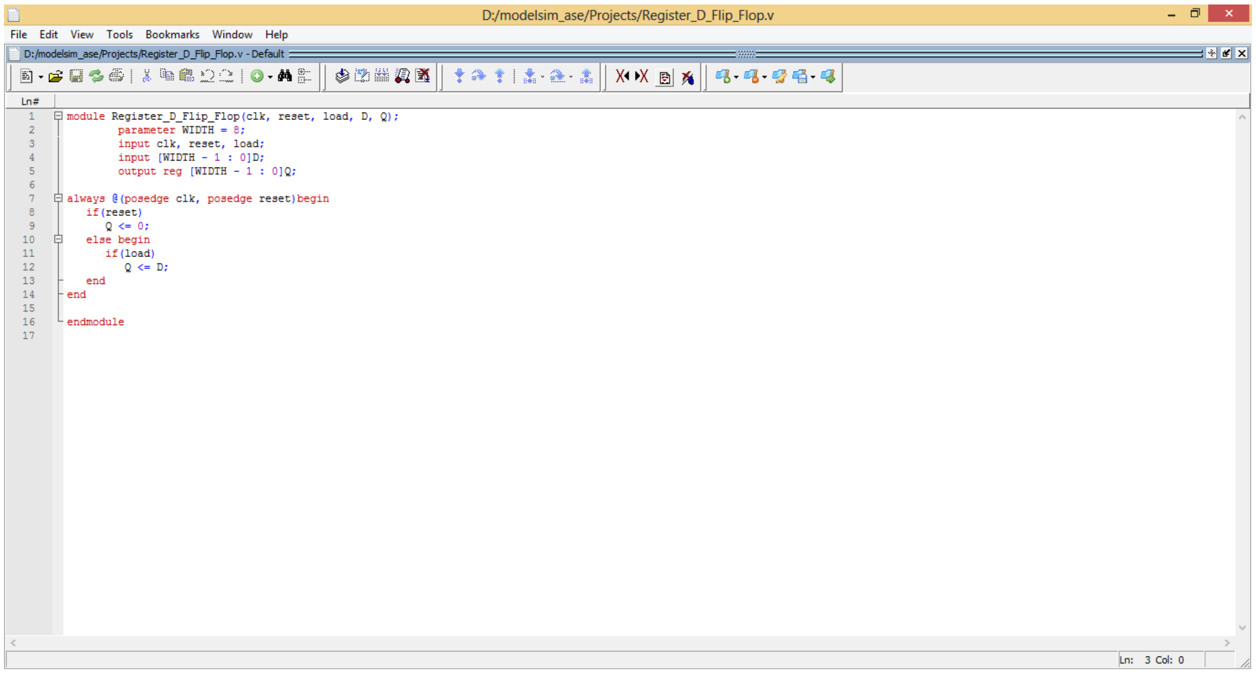
2. Pipe Line Multiplier



D:/modelsim_ase/Projects/Pipe_Line_Multiplier.v

```verilog
module Pipe_Line_Multiplier(A_input, B_input, output_of_multiplier);
        parameter WIDTH = 8;
        input [WIDTH - 1 : 0]A_input, B_input;
        output [(2 * WIDTH) - 1 : 0]output_of_multiplier;

wire [WIDTH - 1 : 0]absolute_value_of_A_input, absolute_value_of_B_input, ArBr, ArBl, AlBr, AlBl, ArBl_plus_AlBr;
wire co_of_ArBl_plus_AlBr, co_of_first_adder, co_of_second_adder;
wire [(2 * WIDTH) - 1 : 0]absolute_value_of_output_of_multiplier;

assign absolute_value_of_A_input = A_input[WIDTH - 1] ? ~(A_input) + 1 : A_input;
assign absolute_value_of_B_input = B_input[WIDTH - 1] ? ~(B_input) + 1 : B_input;

assign ArBr = absolute_value_of_A_input[(WIDTH / 2) - 1 : 0] * absolute_value_of_B_input[(WIDTH / 2) - 1 : 0];
assign ArBl = absolute_value_of_A_input[(WIDTH / 2) - 1 : 0] * absolute_value_of_B_input[WIDTH - 1 : WIDTH / 2];
assign AlBr = absolute_value_of_A_input[WIDTH - 1 : WIDTH / 2] * absolute_value_of_B_input[(WIDTH / 2) - 1 : 0];
assign AlBl = absolute_value_of_A_input[WIDTH - 1 : WIDTH / 2] * absolute_value_of_B_input[WIDTH - 1 : WIDTH / 2];

assign {co_of_ArBl_plus_AlBr, ArBl_plus_AlBr} = ArBl + AlBr;

assign absolute_value_of_output_of_multiplier[(WIDTH / 2) - 1 : 0] = ArBr[(WIDTH / 2) - 1 : 0];
assign {co_of_first_adder, absolute_value_of_output_of_multiplier[WIDTH - 1 : WIDTH / 2]} = ArBl_plus_AlBr[(WIDTH / 2) - 1 : 0] + ArBr[WIDTH - 1 : WIDTH / 2];
assign {co_of_second_adder, absolute_value_of_output_of_multiplier[(3 * (WIDTH / 2)) - 1 : WIDTH]} = ArBl_plus_AlBr[WIDTH - 1 : WIDTH / 2] + AlBl[(WIDTH / 2) - 1 : 0] + co_of_first_ac
assign absolute_value_of_output_of_multiplier[(2 * WIDTH) - 1 : 3 * (WIDTH / 2)] = AlBl[WIDTH - 1 : WIDTH / 2] + co_of_ArBl_plus_AlBr + co_of_second_adder;

assign output_of_multiplier = (A_input[WIDTH - 1] ^ B_input[WIDTH - 1]) ? ~(absolute_value_of_output_of_multiplier) + 1 : absolute_value_of_output_of_multiplier;

endmodule
```

3. Register

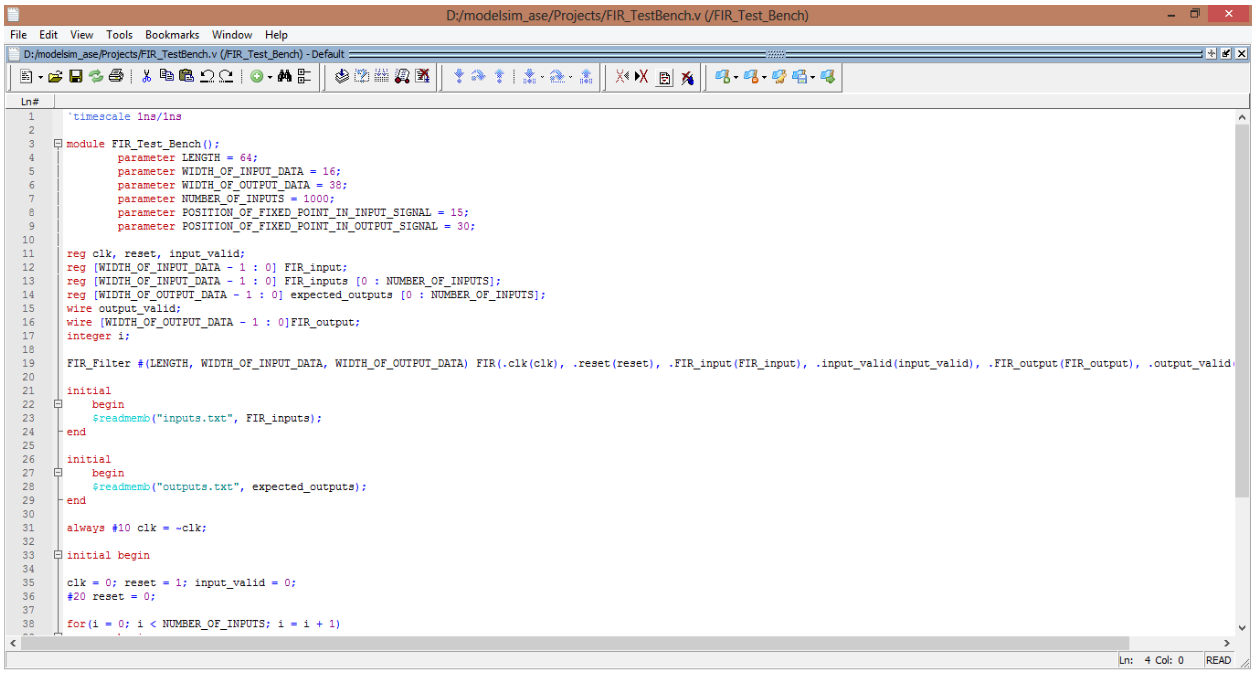


D:/modelsim_ase/Projects/Register_D_Flip_Flop.v

```verilog
module Register_D_Flip_Flop(clk, reset, load, D, Q);
        parameter WIDTH = 8;
        input clk, reset, load;
        input [WIDTH - 1 : 0]D;
        output reg [WIDTH - 1 : 0]Q;

always @(posedge clk, posedge reset)begin
   if(reset)
      Q <= 0;
   else begin
      if(load)
         Q <= D;
   end
end

endmodule
```

Simulation verification of FIR Filter

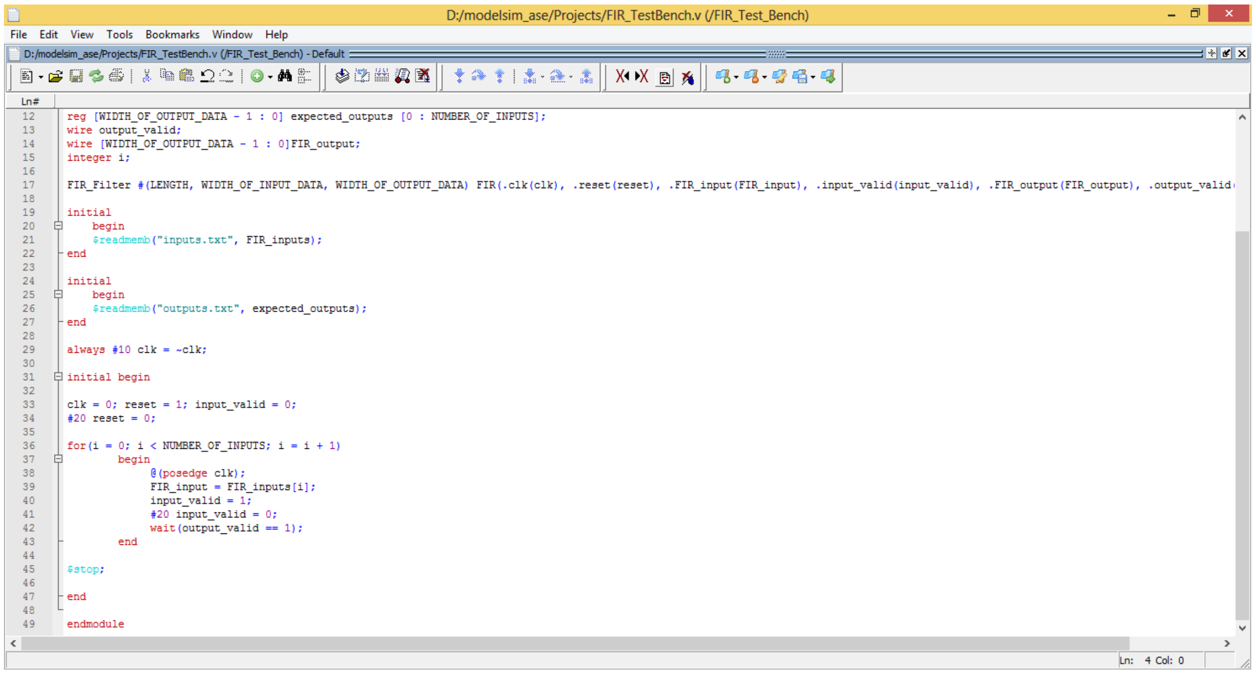
1. Test Bench of FIR Filter



D:/modelsim_ase/Projects/FIR_TestBench.v (/FIR_Test_Bench)

```verilog
`timescale 1ns/1ns

module FIR_Test_Bench();
        parameter LENGTH = 64;
        parameter WIDTH_OF_INPUT_DATA = 16;
        parameter WIDTH_OF_OUTPUT_DATA = 38;
        parameter NUMBER_OF_INPUTS = 1000;
        parameter POSITION_OF_FIXED_POINT_IN_INPUT_SIGNAL = 15;
        parameter POSITION_OF_FIXED_POINT_IN_OUTPUT_SIGNAL = 30;

reg clk, reset, input_valid;
reg [WIDTH_OF_INPUT_DATA - 1 : 0] FIR_input;
reg [WIDTH_OF_INPUT_DATA - 1 : 0] FIR_inputs [0 : NUMBER_OF_INPUTS];
reg [WIDTH_OF_OUTPUT_DATA - 1 : 0] expected_outputs [0 : NUMBER_OF_INPUTS];
wire output_valid;
wire [WIDTH_OF_OUTPUT_DATA - 1 : 0]FIR_output;
integer i;

FIR_Filter #(LENGTH, WIDTH_OF_INPUT_DATA, WIDTH_OF_OUTPUT_DATA) FIR(.clk(clk), .reset(reset), .FIR_input(FIR_input), .input_valid(input_valid), .FIR_output(FIR_output), .output_valid

initial
    begin
    $readmemb("inputs.txt", FIR_inputs);
end

initial
    begin
    $readmemb("outputs.txt", expected_outputs);
end

always #10 clk = ~clk;

initial begin

clk = 0; reset = 1; input_valid = 0;
#20 reset = 0;

for(i = 0; i < NUMBER_OF_INPUTS; i = i + 1)
```

Ln: 4 Col: 0 READ

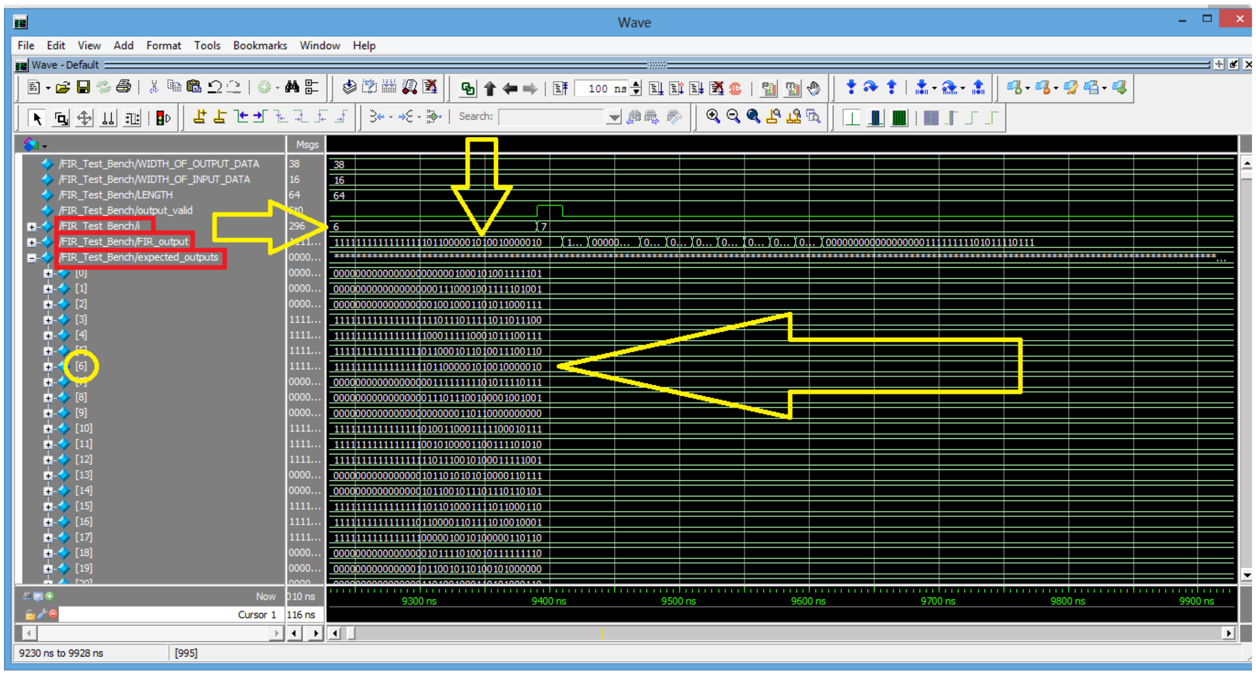


D:/modelsim_ase/Projects/FIR_TestBench.v (/FIR_Test_Bench)

```verilog
reg [WIDTH_OF_OUTPUT_DATA - 1 : 0] expected_outputs [0 : NUMBER_OF_INPUTS];
wire output_valid;
wire [WIDTH_OF_OUTPUT_DATA - 1 : 0]FIR_output;
integer i;

FIR_Filter #(LENGTH, WIDTH_OF_INPUT_DATA, WIDTH_OF_OUTPUT_DATA) FIR(.clk(clk), .reset(reset), .FIR_input(FIR_input), .input_valid(input_valid), .FIR_output(FIR_output), .output_valid

initial
    begin
    $readmemb("inputs.txt", FIR_inputs);
end

initial
    begin
    $readmemb("outputs.txt", expected_outputs);
end

always #10 clk = ~clk;

initial begin

clk = 0; reset = 1; input_valid = 0;
#20 reset = 0;

for(i = 0; i < NUMBER_OF_INPUTS; i = i + 1)
        begin
             @(posedge clk);
             FIR_input = FIR_inputs[i];
             input_valid = 1;
             #20 input_valid = 0;
             wait(output_valid == 1);
        end

$stop;

end

endmodule
```

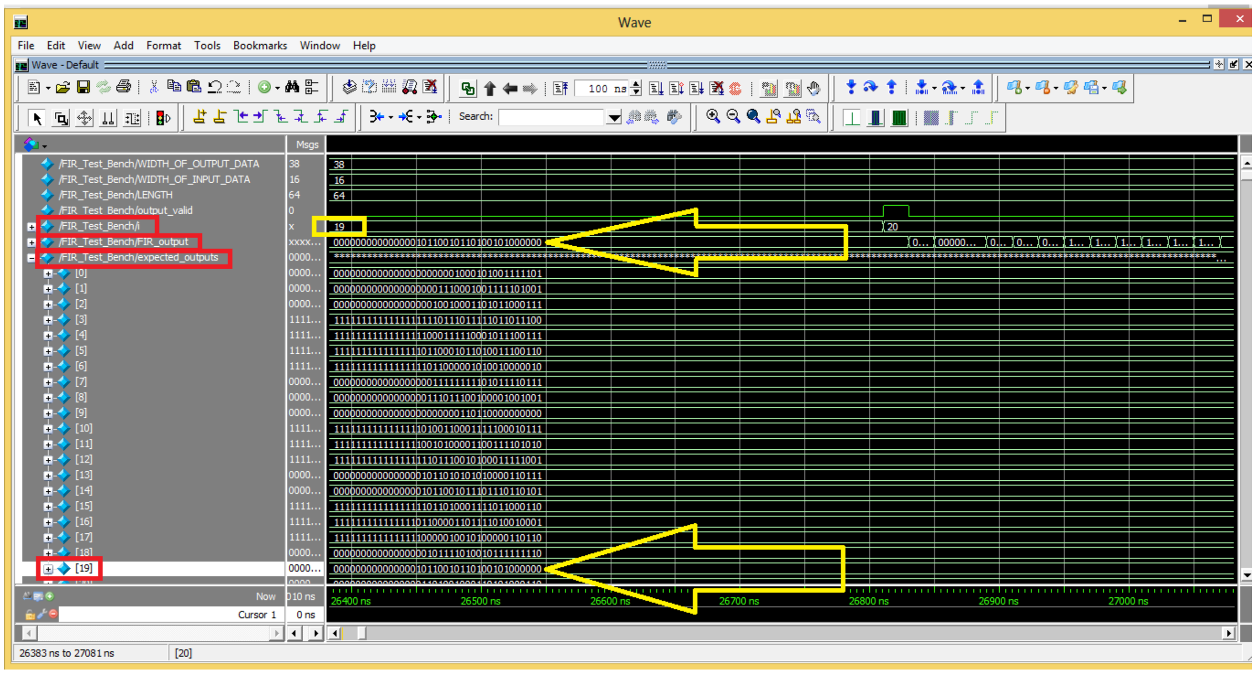
Ln: 4 Col: 0

1. Wave form of Test Bench of FIR Filter



```
Wave - Default                                         Msgs
/FIR_Test_Bench/WIDTH_OF_OUTPUT_DATA                   38       38
/FIR_Test_Bench/WIDTH_OF_INPUT_DATA                    16       16
/FIR_Test_Bench/LENGTH                                 64       64
/FIR_Test_Bench/output_valid                           St0
/FIR_Test_Bench/i                                      296      6                                            7
/FIR_Test_Bench/FIR_output                             1111...  11111111111111111011000001010010000010  1... 00000... 0... 0... 0... 0... 0... 0... 0... 00000000000000000011111111101011110111
/FIR_Test_Bench/expected_outputs                       0000...  ******************************************...
  [0]                                                  0000...  00000000000000000000001000101001111101
  [1]                                                  0000...  00000000000000000001110001001111101001
  [2]                                                  0000...  00000000000000000010010001101011000111
  [3]                                                  1111...  11111111111111111110111011111011011100
  [4]                                                  1111...  11111111111111111000111110001011100111
  [5]                                                  1111...  11111111111111110110001011010011100110
  [6]                                                  1111...  11111111111111111011000001010010000010
  [7]                                                  0000...  00000000000000000011111111101011110111
  [8]                                                  0000...  00000000000000000111011100100001001001
  [9]                                                  0000...  00000000000000000000000110110000000000
  [10]                                                 1111...  11111111111111110100110001111000101111
  [11]                                                 1111...  11111111111111100101000011001111010 10
  [12]                                                 1111...  11111111111111111101110010100011111001
  [13]                                                 0000...  00000000000000001011010101010000110111
  [14]                                                 0000...  00000000000000001011001011101110110101
  [15]                                                 1111...  11111111111111111011010001110110001 10
  [16]                                                 1111...  11111111111111011000011011110100100 01
  [17]                                                 1111...  11111111111111100000100101000001101 10
  [18]                                                 0000...  00000000000000000101111010010111111110
  [19]                                                 0000...  00000000000000010110010110100101000000
Now                                                    010 ns      9300 ns   9400 ns   9500 ns   9600 ns   9700 ns   9800 ns   9900 ns
Cursor 1                                               116 ns
9230 ns to 9928 ns          [995]
```

As you see 7th output of FIR Filter is equal to expected output of 7th FIR input.

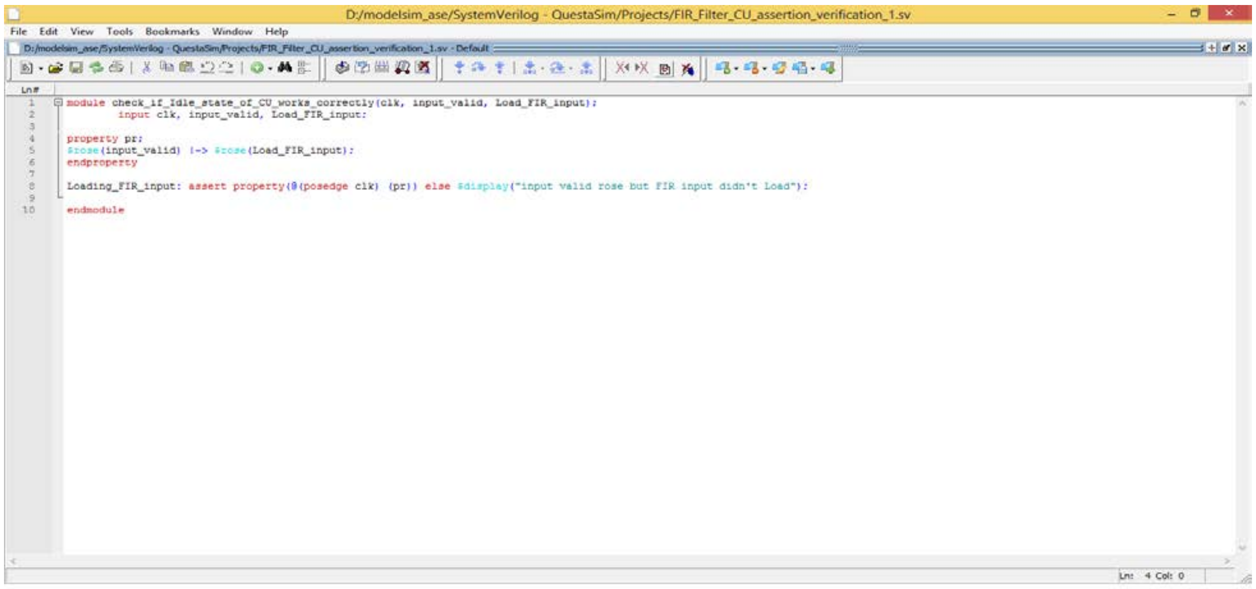


```
Wave - Default                                         Msgs
/FIR_Test_Bench/WIDTH_OF_OUTPUT_DATA                   38       38
/FIR_Test_Bench/WIDTH_OF_INPUT_DATA                    16       16
/FIR_Test_Bench/LENGTH                                 64       64
/FIR_Test_Bench/output_valid                           0
/FIR_Test_Bench/i                                      x        19                                           20
/FIR_Test_Bench/FIR_output                             xxxx...  00000000000000010110010110100101000000  0... 00000... 0... 0... 0... 1... 1... 1... 1... 1... 1...
/FIR_Test_Bench/expected_outputs                       0000...  ******************************************...
  [0]                                                  0000...  00000000000000000000001000101001111101
  [1]                                                  0000...  00000000000000000001110001001111101001
  [2]                                                  0000...  00000000000000000010010001101011000111
  [3]                                                  1111...  11111111111111111110111011111011011100
  [4]                                                  1111...  11111111111111111000111110001011100111
  [5]                                                  1111...  11111111111111110110001011010011100110
  [6]                                                  1111...  11111111111111111011000001010010000010
  [7]                                                  0000...  00000000000000000011111111101011110111
  [8]                                                  0000...  00000000000000000111011100100001001001
  [9]                                                  0000...  00000000000000000000000110110000000000
  [10]                                                 1111...  11111111111111110100110001111000101111
  [11]                                                 1111...  11111111111111100101000011001111010 10
  [12]                                                 1111...  11111111111111111101110010100011111001
  [13]                                                 0000...  00000000000000001011010101010000110111
  [14]                                                 0000...  00000000000000001011001011101110110101
  [15]                                                 1111...  11111111111111111011010001110110001 10
  [16]                                                 1111...  11111111111111011000011011110100100 01
  [17]                                                 1111...  11111111111111100000100101000001101 10
  [18]                                                 0000...  00000000000000000101111010010111111110
  [19]                                                 0000...  00000000000000010110010110100101000000
Now                                                    010 ns      26400 ns  26500 ns  26600 ns  26700 ns  26800 ns  26900 ns  27000 ns
Cursor 1                                               0 ns
26383 ns to 27081 ns        [20]
```

As you see 20th output of FIR Filter is equal to expected output of 20th FIR input.

Assertion verification of FIR Filter

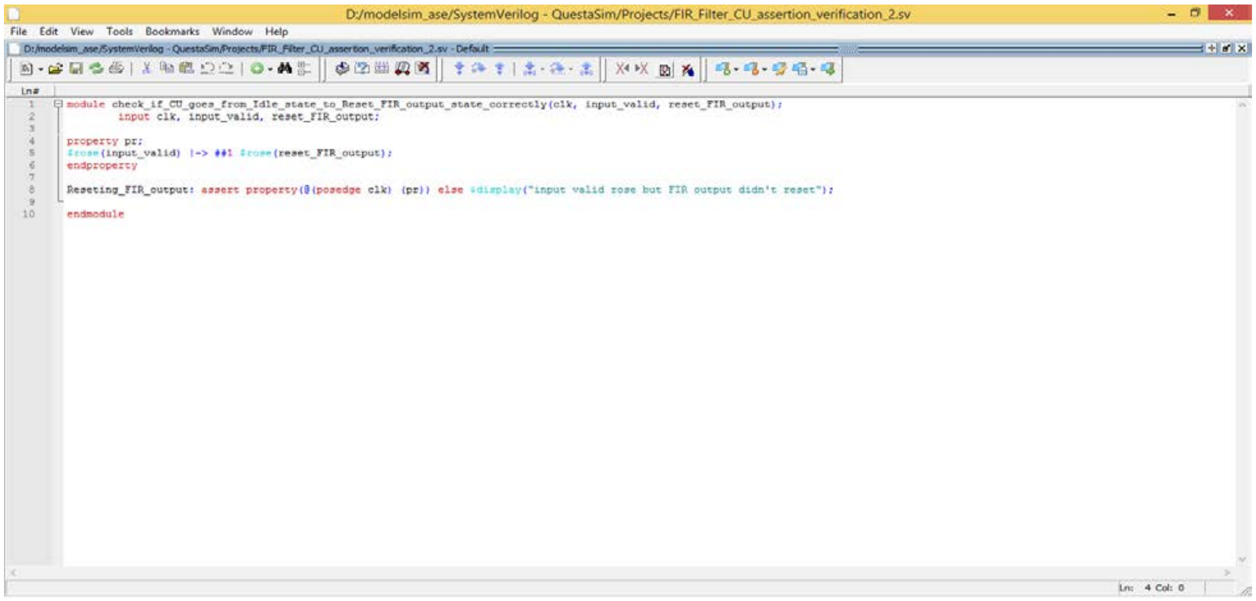
Assertion 1: With assertion of input_valid, input must be loaded by asserting Load_FIR_input.



D:/modelsim_ase/SystemVerilog - QuestaSim/Projects/FIR_Filter_CU_assertion_verification_1.sv
File Edit View Tools Bookmarks Window Help
D:/modelsim_ase/SystemVerilog - QuestaSim/Projects/FIR_Filter_CU_assertion_verification_1.sv - Default
Ln#
Ln: 4 Col: 0

```
module check_if_Idle_state_of_CU_works_correctly(clk, input_valid, Load_FIR_input);
        input clk, input_valid, Load_FIR_input;

property pr;
$rose(input_valid) |-> $rose(Load_FIR_input);
endproperty

Loading_FIR_input: assert property(@(posedge clk) (pr)) else $display("input valid rose but FIR input didn't Load");

endmodule
```

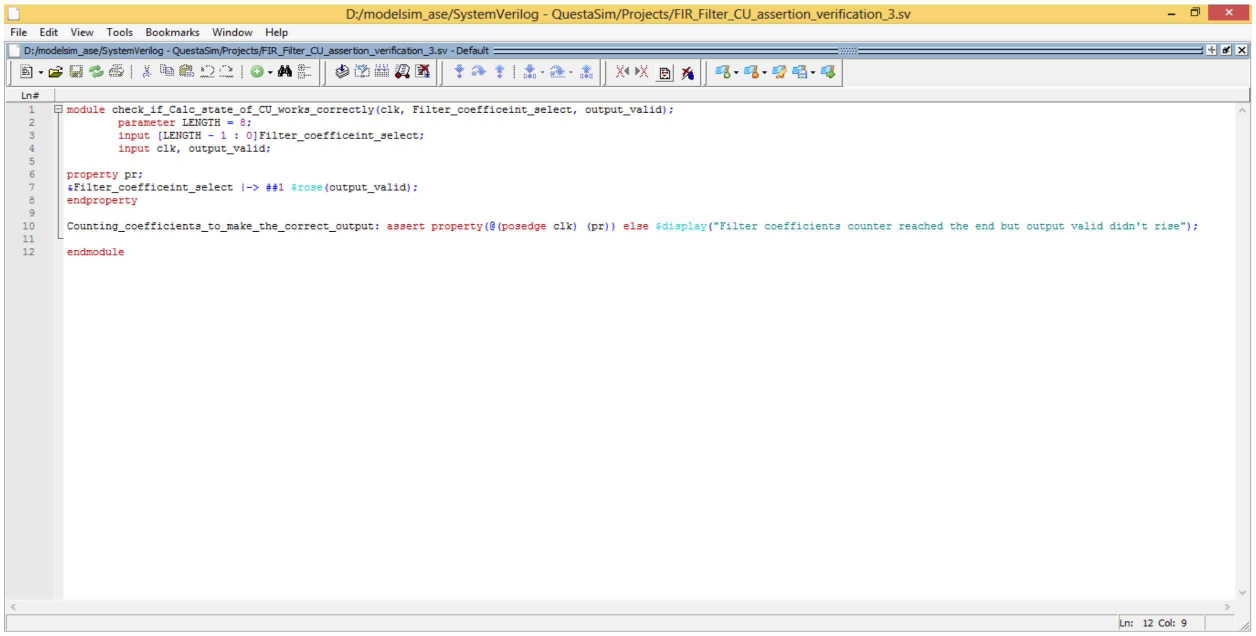
Assertion 2: With assertion of input_valid, calculation of FIR output must be started so previous FIR output must be reset in next clock.



D:/modelsim_ase/SystemVerilog - QuestaSim/Projects/FIR_Filter_CU_assertion_verification_2.sv
File Edit View Tools Bookmarks Window Help
D:/modelsim_ase/SystemVerilog - QuestaSim/Projects/FIR_Filter_CU_assertion_verification_2.sv - Default
Ln#
Ln: 4 Col: 0

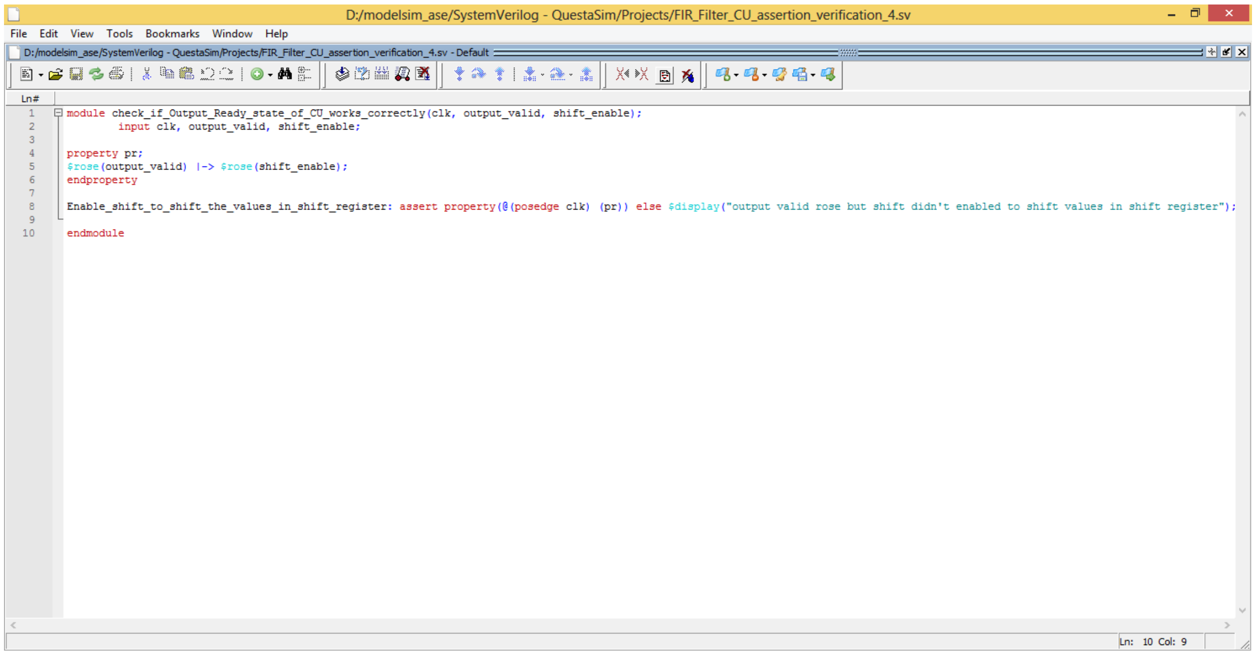
```
module check_if_CU_goes_from_Idle_state_to_Reset_FIR_output_state_correctly(clk, input_valid, reset_FIR_output);
        input clk, input_valid, reset_FIR_output;

property pr;
$rose(input_valid) |-> ##1 $rose(reset_FIR_output);
endproperty

Reseting_FIR_output: assert property(@(posedge clk) (pr)) else $display("input valid rose but FIR output didn't reset");

endmodule
```

Assertion 3: Whenever Filter coefficient reaches the end and multiplications of all coefficients are calculated, output is ready and output_valid must be asserted.


D:/modelsim_ase/SystemVerilog - QuestaSim/Projects/FIR_Filter_CU_assertion_verification_3.sv

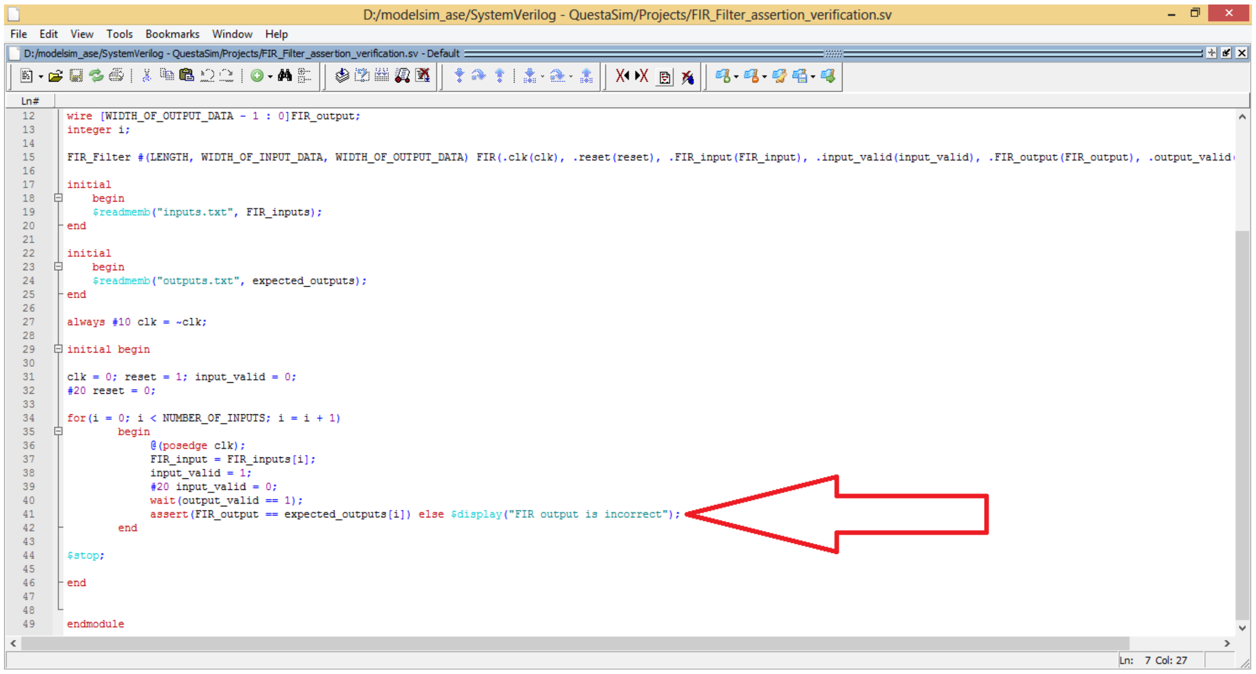
```
module check_if_Calc_state_of_CU_works_correctly(clk, Filter_coefficeint_select, output_valid);
        parameter LENGTH = 8;
        input [LENGTH - 1 : 0]Filter_coefficeint_select;
        input clk, output_valid;

property pr;
&Filter_coefficeint_select |-> ##1 $rose(output_valid);
endproperty

Counting_coefficients_to_make_the_correct_output: assert property(@(posedge clk) (pr)) else $display("Filter coefficients counter reached the end but output valid didn't rise");

endmodule
```

Assertion 4: With assertion of output_valid, values of shift register must be shifted for calculation next outputs so shift_enable must be asserted.


D:/modelsim_ase/SystemVerilog - QuestaSim/Projects/FIR_Filter_CU_assertion_verification_4.sv

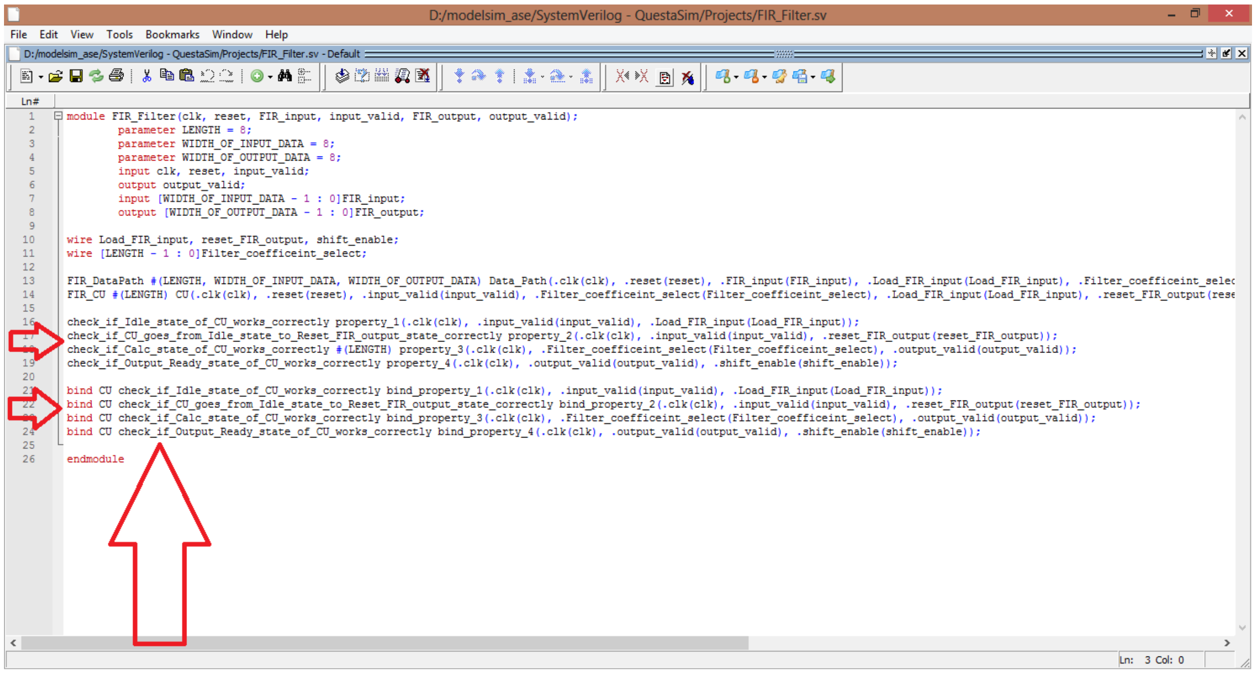
```
module check_if_Output_Ready_state_of_CU_works_correctly(clk, output_valid, shift_enable);
        input clk, output_valid, shift_enable;

property pr;
$rose(output_valid) |-> $rose(shift_enable);
endproperty

Enable_shift_to_shift_the_values_in_shift_register: assert property(@(posedge clk) (pr)) else $display("output valid rose but shift didn't enabled to shift values in shift register");

endmodule
```

Assertion 5: checks if output of FIR filter is correct and equal to expected output of FIR filter.


D:/modelsim_ase/SystemVerilog - QuestaSim/Projects/FIR_Filter_assertion_verification.sv
File Edit View Tools Bookmarks Window Help
D:/modelsim_ase/SystemVerilog - QuestaSim/Projects/FIR_Filter_assertion_verification.sv - Default
Ln#
Ln: 7 Col: 27

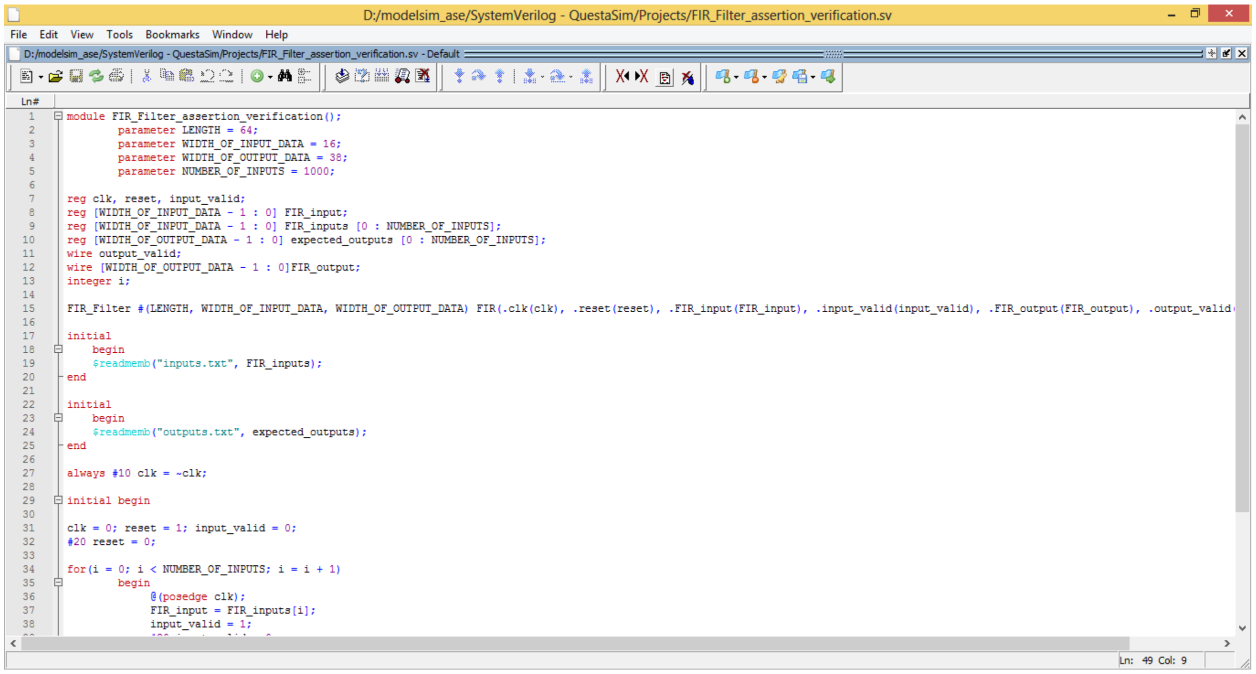
```
wire [WIDTH_OF_OUTPUT_DATA - 1 : 0]FIR_output;
integer i;

FIR_Filter #(LENGTH, WIDTH_OF_INPUT_DATA, WIDTH_OF_OUTPUT_DATA) FIR(.clk(clk), .reset(reset), .FIR_input(FIR_input), .input_valid(input_valid), .FIR_output(FIR_output), .output_valid

initial
    begin
    $readmemb("inputs.txt", FIR_inputs);
end

initial
    begin
    $readmemb("outputs.txt", expected_outputs);
end

always #10 clk = ~clk;

initial begin

clk = 0; reset = 1; input_valid = 0;
#20 reset = 0;

for(i = 0; i < NUMBER_OF_INPUTS; i = i + 1)
        begin
            @(posedge clk);
            FIR_input = FIR_inputs[i];
            input_valid = 1;
            #20 input_valid = 0;
            wait(output_valid == 1);
            assert(FIR_output == expected_outputs[i]) else $display("FIR output is incorrect");
        end

$stop;

end


endmodule
```

Binding Assertions

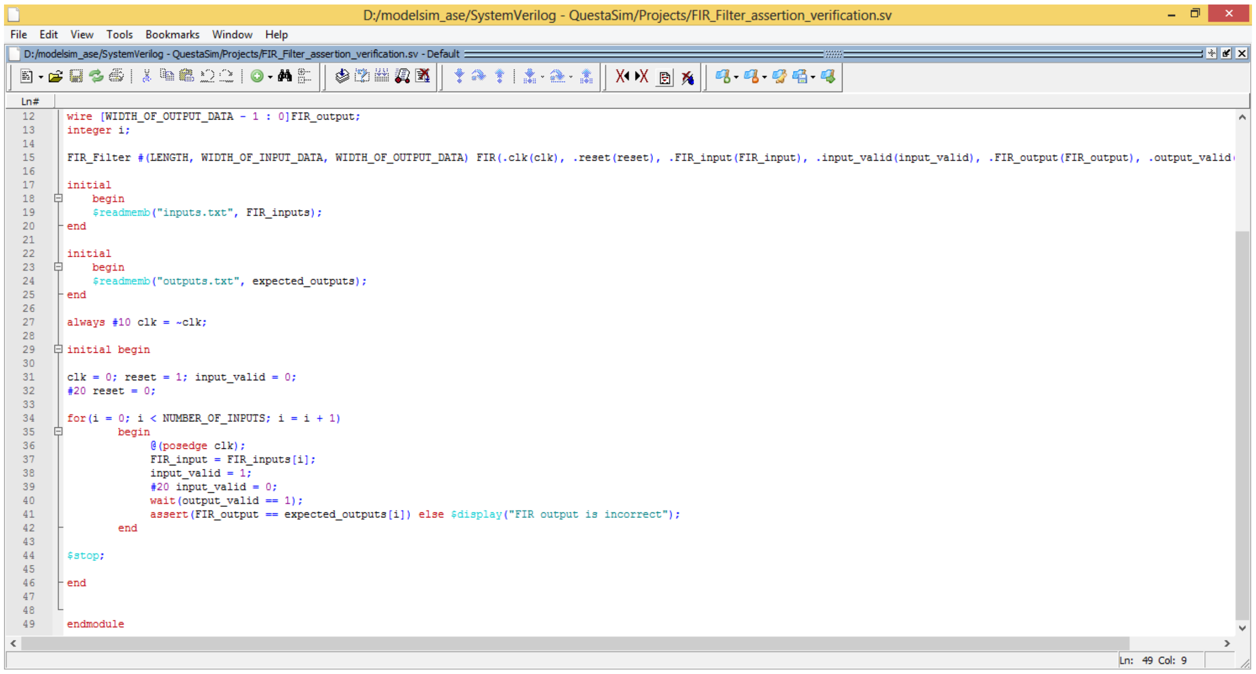

D:/modelsim_ase/SystemVerilog - QuestaSim/Projects/FIR_Filter.sv
File Edit View Tools Bookmarks Window Help
D:/modelsim_ase/SystemVerilog - QuestaSim/Projects/FIR_Filter.sv - Default
Ln#
Ln: 3 Col: 0

```
module FIR_Filter(clk, reset, FIR_input, input_valid, FIR_output, output_valid);
        parameter LENGTH = 8;
        parameter WIDTH_OF_INPUT_DATA = 8;
        parameter WIDTH_OF_OUTPUT_DATA = 8;
        input clk, reset, input_valid;
        output output_valid;
        input [WIDTH_OF_INPUT_DATA - 1 : 0]FIR_input;
        output [WIDTH_OF_OUTPUT_DATA - 1 : 0]FIR_output;

wire Load_FIR_input, reset_FIR_output, shift_enable;
wire [LENGTH - 1 : 0]Filter_coefficeint_select;

FIR_DataPath #(LENGTH, WIDTH_OF_INPUT_DATA, WIDTH_OF_OUTPUT_DATA) Data_Path(.clk(clk), .reset(reset), .FIR_input(FIR_input), .Load_FIR_input(Load_FIR_input), .Filter_coefficeint_selec
FIR_CU #(LENGTH) CU(.clk(clk), .reset(reset), .input_valid(input_valid), .Filter_coefficeint_select(Filter_coefficeint_select), .Load_FIR_input(Load_FIR_input), .reset_FIR_output(rese

check_if_Idle_state_of_CU_works_correctly property_1(.clk(clk), .input_valid(input_valid), .Load_FIR_input(Load_FIR_input));
check_if_CU_goes_from_Idle_state_to_Reset_FIR_output_state_correctly property_2(.clk(clk), .input_valid(input_valid), .reset_FIR_output(reset_FIR_output));
check_if_Calc_state_of_CU_works_correctly #(LENGTH) property_3(.clk(clk), .Filter_coefficeint_select(Filter_coefficeint_select), .output_valid(output_valid));
check_if_Output_Ready_state_of_CU_works_correctly property_4(.clk(clk), .output_valid(output_valid), .shift_enable(shift_enable));

bind CU check_if_Idle_state_of_CU_works_correctly bind_property_1(.clk(clk), .input_valid(input_valid), .Load_FIR_input(Load_FIR_input));
bind CU check_if_CU_goes_from_Idle_state_to_Reset_FIR_output_state_correctly bind_property_2(.clk(clk), .input_valid(input_valid), .reset_FIR_output(reset_FIR_output));
bind CU check_if_Calc_state_of_CU_works_correctly bind_property_3(.clk(clk), .Filter_coefficeint_select(Filter_coefficeint_select), .output_valid(output_valid));
bind CU check_if_Output_Ready_state_of_CU_works_correctly bind_property_4(.clk(clk), .output_valid(output_valid), .shift_enable(shift_enable));

endmodule
```

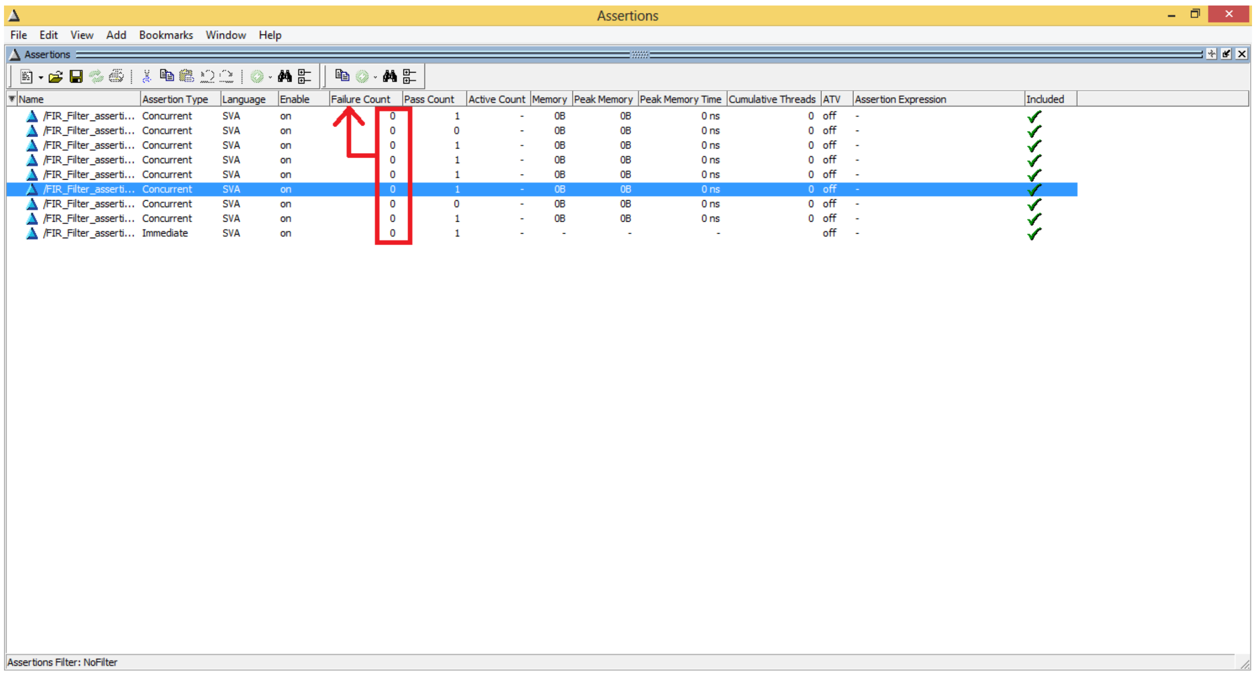
Testing FIR Filter by assertions


D:/modelsim_ase/SystemVerilog - QuestaSim/Projects/FIR_Filter_assertion_verification.sv
File Edit View Tools Bookmarks Window Help
D:/modelsim_ase/SystemVerilog - QuestaSim/Projects/FIR_Filter_assertion_verification.sv - Default
Ln#
Ln: 49 Col: 9

```
module FIR_Filter_assertion_verification();
        parameter LENGTH = 64;
        parameter WIDTH_OF_INPUT_DATA = 16;
        parameter WIDTH_OF_OUTPUT_DATA = 38;
        parameter NUMBER_OF_INPUTS = 1000;

reg clk, reset, input_valid;
reg [WIDTH_OF_INPUT_DATA - 1 : 0] FIR_input;
reg [WIDTH_OF_INPUT_DATA - 1 : 0] FIR_inputs [0 : NUMBER_OF_INPUTS];
reg [WIDTH_OF_OUTPUT_DATA - 1 : 0] expected_outputs [0 : NUMBER_OF_INPUTS];
wire output_valid;
wire [WIDTH_OF_OUTPUT_DATA - 1 : 0]FIR_output;
integer i;

FIR_Filter #(LENGTH, WIDTH_OF_INPUT_DATA, WIDTH_OF_OUTPUT_DATA) FIR(.clk(clk), .reset(reset), .FIR_input(FIR_input), .input_valid(input_valid), .FIR_output(FIR_output), .output_valid

initial
    begin
    $readmemb("inputs.txt", FIR_inputs);
end

initial
    begin
    $readmemb("outputs.txt", expected_outputs);
end

always #10 clk = ~clk;

initial begin

clk = 0; reset = 1; input_valid = 0;
#20 reset = 0;

for(i = 0; i < NUMBER_OF_INPUTS; i = i + 1)
        begin
             @(posedge clk);
             FIR_input = FIR_inputs[i];
             input_valid = 1;
```


D:/modelsim_ase/SystemVerilog - QuestaSim/Projects/FIR_Filter_assertion_verification.sv
File Edit View Tools Bookmarks Window Help
D:/modelsim_ase/SystemVerilog - QuestaSim/Projects/FIR_Filter_assertion_verification.sv - Default
Ln#
Ln: 49 Col: 9

```
wire [WIDTH_OF_OUTPUT_DATA - 1 : 0]FIR_output;
integer i;

FIR_Filter #(LENGTH, WIDTH_OF_INPUT_DATA, WIDTH_OF_OUTPUT_DATA) FIR(.clk(clk), .reset(reset), .FIR_input(FIR_input), .input_valid(input_valid), .FIR_output(FIR_output), .output_valid

initial
    begin
    $readmemb("inputs.txt", FIR_inputs);
end

initial
    begin
    $readmemb("outputs.txt", expected_outputs);
end

always #10 clk = ~clk;

initial begin

clk = 0; reset = 1; input_valid = 0;
#20 reset = 0;

for(i = 0; i < NUMBER_OF_INPUTS; i = i + 1)
        begin
             @(posedge clk);
             FIR_input = FIR_inputs[i];
             input_valid = 1;
             #20 input_valid = 0;
             wait(output_valid == 1);
             assert(FIR_output == expected_outputs[i]) else $display("FIR output is incorrect");
        end

$stop;

end


endmodule
```



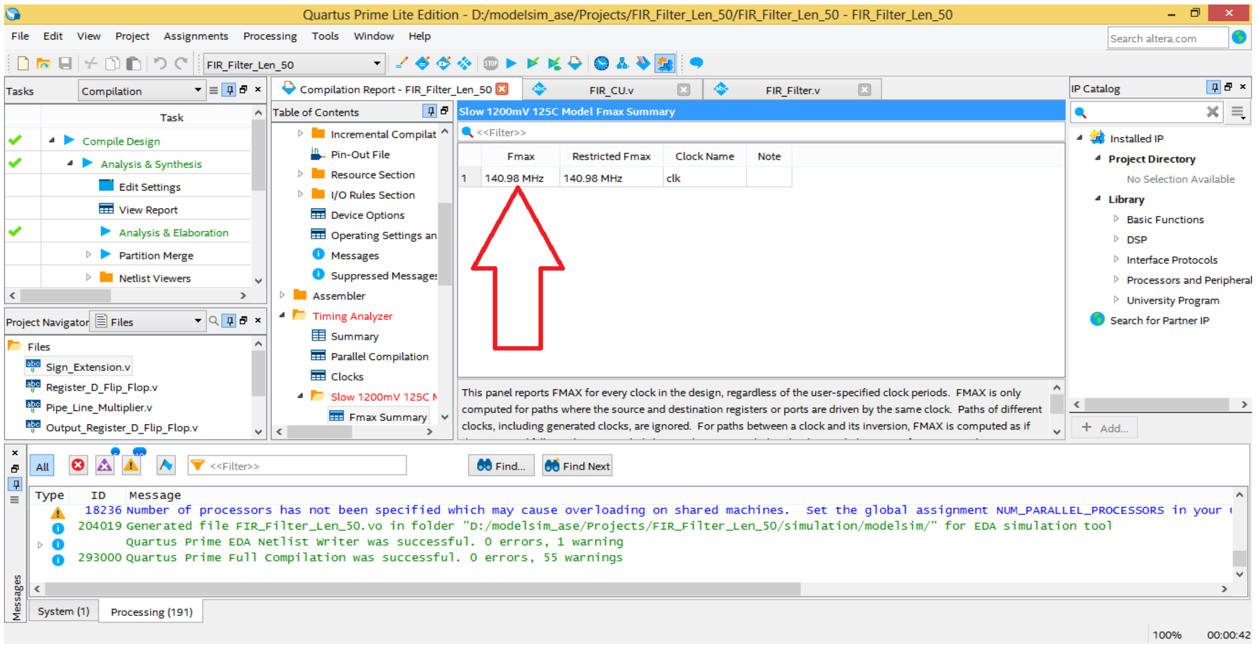
Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Peak Memory Time	Cumulative Threads	ATV	Assertion Expression	Included
/FIR_Filter_asserti…	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	-	✓
/FIR_Filter_asserti…	Concurrent	SVA	on	0	0	-	0B	0B	0 ns	0	off	-	✓
/FIR_Filter_asserti…	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	-	✓
/FIR_Filter_asserti…	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	-	✓
/FIR_Filter_asserti…	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	-	✓
/FIR_Filter_asserti…	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	-	✓
/FIR_Filter_asserti…	Concurrent	SVA	on	0	0	-	0B	0B	0 ns	0	off	-	✓
/FIR_Filter_asserti…	Concurrent	SVA	on	0	1	-	0B	0B	0 ns	0	off	-	✓
/FIR_Filter_asserti…	Immediate	SVA	on	0	1	-	-	-	-		off	-	✓

Assertions Filter: NoFilter

As you see all assertions are passed and there is no failure on assertions.

Synthesize of FIR Filter in Quartus

1. Maximum Frequency of synthesized FIR Filter with length 50



Quartus Prime Lite Edition - D:/modelsim_ase/Projects/FIR_Filter_Len_50/FIR_Filter_Len_50 - FIR_Filter_Len_50

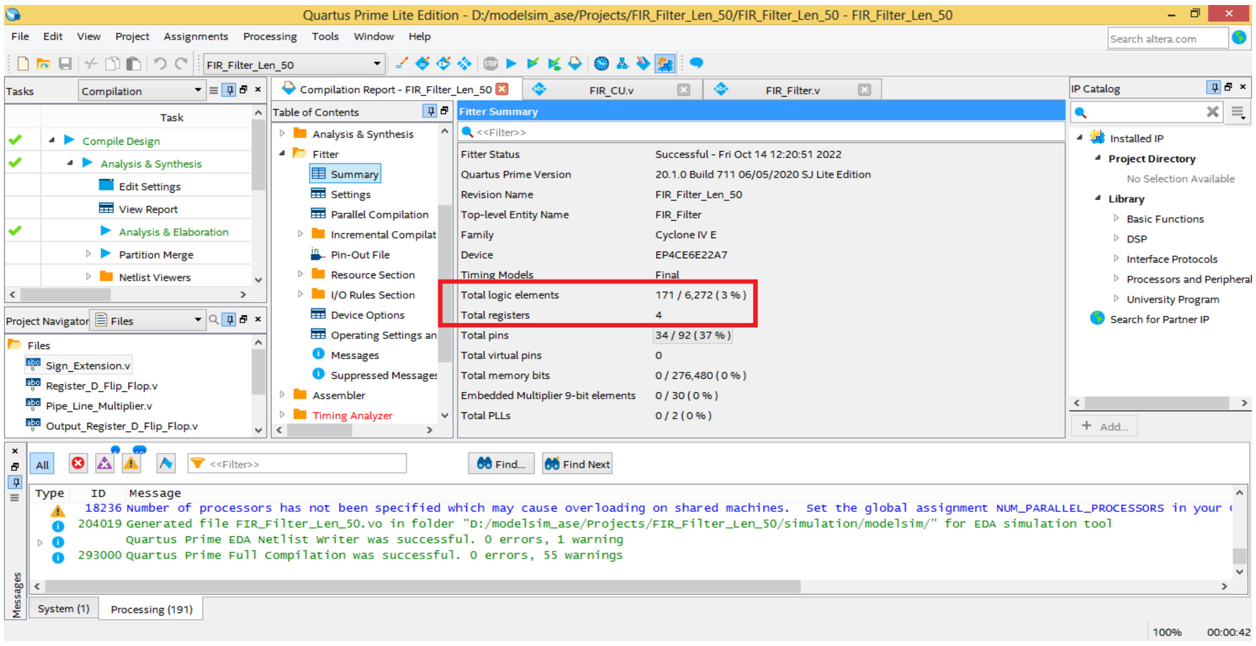
Slow 1200mV 125C Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	140.98 MHz	140.98 MHz	clk	

This panel reports FMAX for every clock in the design, regardless of the user-specified clock periods. FMAX is only computed for paths where the source and destination registers or ports are driven by the same clock. Paths of different clocks, including generated clocks, are ignored. For paths between a clock and its inversion, FMAX is computed as if

```
Type   ID    Message
     18236 Number of processors has not been specified which may cause overloading on shared machines.  Set the global assignment NUM_PARALLEL_PROCESSORS in your
    204019 Generated file FIR_Filter_Len_50.vo in folder "D:/modelsim_ase/Projects/FIR_Filter_Len_50/simulation/modelsim/" for EDA simulation tool
           Quartus Prime EDA Netlist Writer was successful. 0 errors, 1 warning
    293000 Quartus Prime Full Compilation was successful. 0 errors, 55 warnings
```

Number of Flip Flops and registers of synthesized FIR Filter with length 50



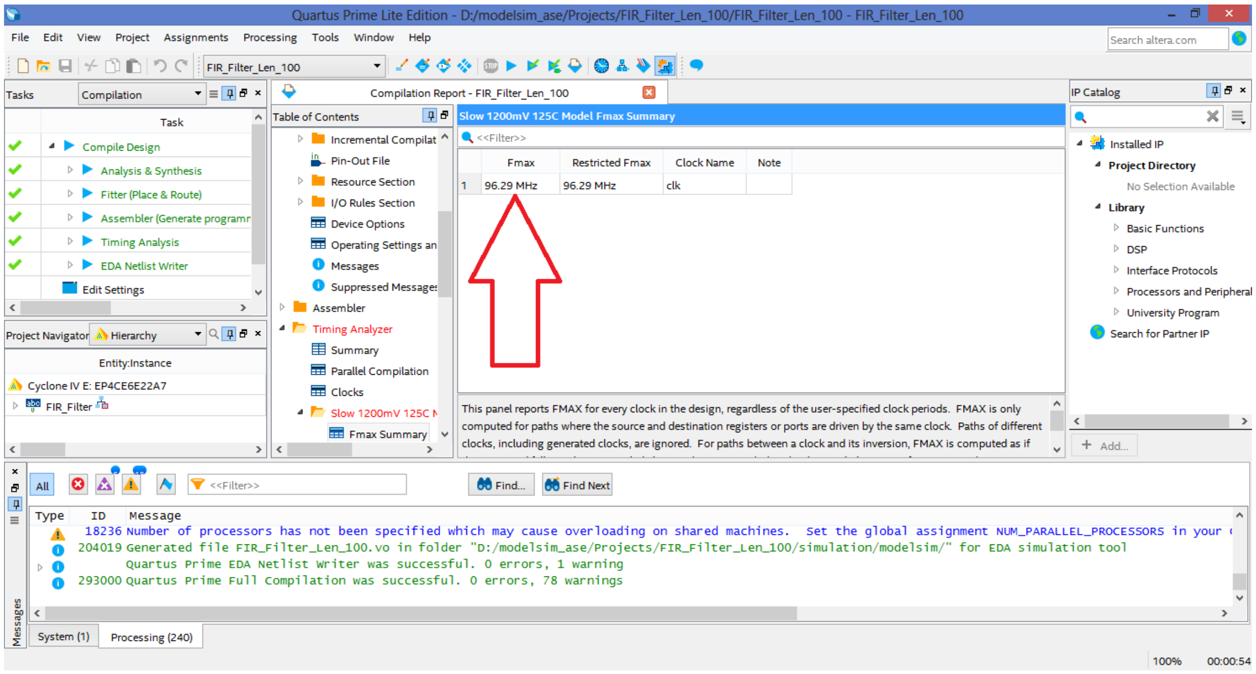
Quartus Prime Lite Edition - D:/modelsim_ase/Projects/FIR_Filter_Len_50/FIR_Filter_Len_50 - FIR_Filter_Len_50

Fitter Summary

Fitter Status	Successful - Fri Oct 14 12:20:51 2022
Quartus Prime Version	20.1.0 Build 711 06/05/2020 SJ Lite Edition
Revision Name	FIR_Filter_Len_50
Top-level Entity Name	FIR_Filter
Family	Cyclone IV E
Device	EP4CE6E22A7
Timing Models	Final
Total logic elements	171 / 6,272 (3 %)
Total registers	4
Total pins	34 / 92 (37 %)
Total virtual pins	0
Total memory bits	0 / 276,480 (0 %)
Embedded Multiplier 9-bit elements	0 / 30 (0 %)
Total PLLs	0 / 2 (0 %)

```
Type   ID    Message
     18236 Number of processors has not been specified which may cause overloading on shared machines.  Set the global assignment NUM_PARALLEL_PROCESSORS in your
    204019 Generated file FIR_Filter_Len_50.vo in folder "D:/modelsim_ase/Projects/FIR_Filter_Len_50/simulation/modelsim/" for EDA simulation tool
           Quartus Prime EDA Netlist Writer was successful. 0 errors, 1 warning
    293000 Quartus Prime Full Compilation was successful. 0 errors, 55 warnings
```

2. Maximum Frequency of synthesized FIR Filter with length 100


Quartus Prime Lite Edition - D:/modelsim_ase/Projects/FIR_Filter_Len_100/FIR_Filter_Len_100 - FIR_Filter_Len_100
File Edit View Project Assignments Processing Tools Window Help
Compilation Report - FIR_Filter_Len_100
Table of Contents
IP Catalog

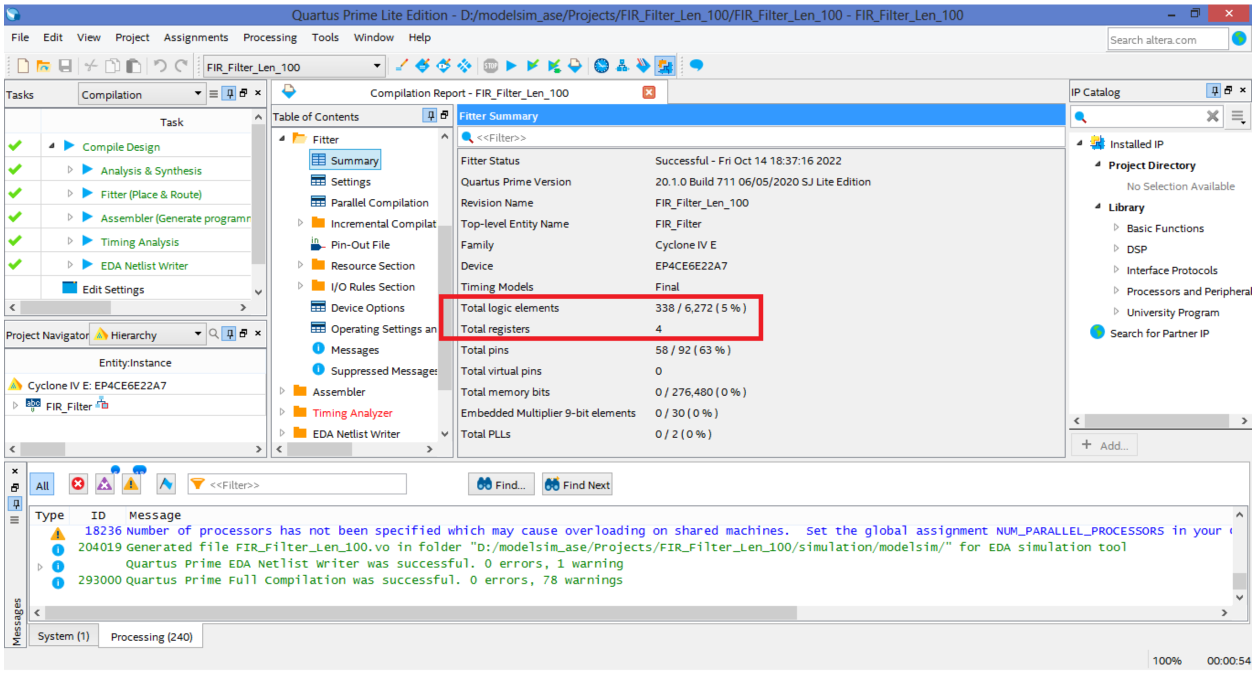
Slow 1200mV 125C Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	96.29 MHz	96.29 MHz	clk	

This panel reports FMAX for every clock in the design, regardless of the user-specified clock periods. FMAX is only computed for paths where the source and destination registers or ports are driven by the same clock. Paths of different clocks, including generated clocks, are ignored. For paths between a clock and its inversion, FMAX is computed as if

```
Type  ID     Message
      18236 Number of processors has not been specified which may cause overloading on shared machines.  Set the global assignment NUM_PARALLEL_PROCESSORS in your
     204019 Generated file FIR_Filter_Len_100.vo in folder "D:/modelsim_ase/Projects/FIR_Filter_Len_100/simulation/modelsim/" for EDA simulation tool
            Quartus Prime EDA Netlist Writer was successful. 0 errors, 1 warning
     293000 Quartus Prime Full Compilation was successful. 0 errors, 78 warnings
```

Number of Flip Flops and registers of synthesized FIR Filter with length 100


Quartus Prime Lite Edition - D:/modelsim_ase/Projects/FIR_Filter_Len_100/FIR_Filter_Len_100 - FIR_Filter_Len_100
File Edit View Project Assignments Processing Tools Window Help
Compilation Report - FIR_Filter_Len_100
Table of Contents
IP Catalog

Fitter Summary

Fitter Status	Successful - Fri Oct 14 18:37:16 2022
Quartus Prime Version	20.1.0 Build 711 06/05/2020 SJ Lite Edition
Revision Name	FIR_Filter_Len_100
Top-level Entity Name	FIR_Filter
Family	Cyclone IV E
Device	EP4CE6E22A7
Timing Models	Final
Total logic elements	338 / 6,272 (5 %)
Total registers	4
Total pins	58 / 92 (63 %)
Total virtual pins	0
Total memory bits	0 / 276,480 (0 %)
Embedded Multiplier 9-bit elements	0 / 30 (0 %)
Total PLLs	0 / 2 (0 %)

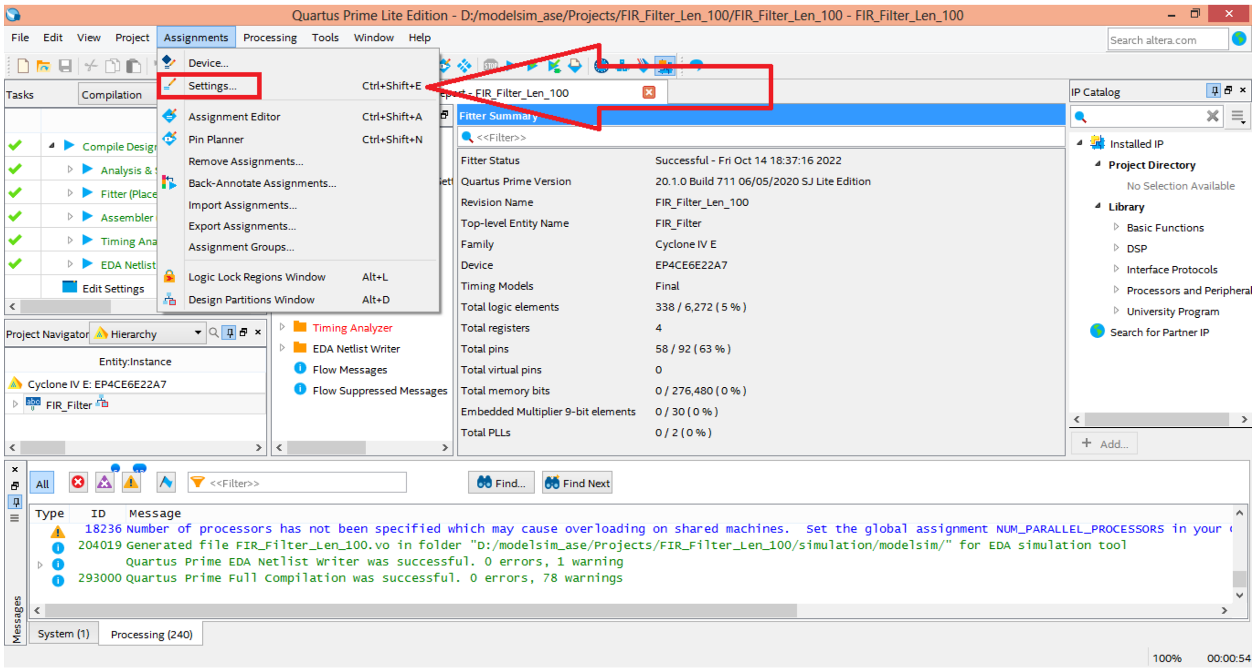
```
Type  ID     Message
      18236 Number of processors has not been specified which may cause overloading on shared machines.  Set the global assignment NUM_PARALLEL_PROCESSORS in your
     204019 Generated file FIR_Filter_Len_100.vo in folder "D:/modelsim_ase/Projects/FIR_Filter_Len_100/simulation/modelsim/" for EDA simulation tool
            Quartus Prime EDA Netlist Writer was successful. 0 errors, 1 warning
     293000 Quartus Prime Full Compilation was successful. 0 errors, 78 warnings
```

As you see as length of FIR Filter increases, number of registers and logical elements increase and Fmax decreases.

With increasing length, number of coefficients increases, so length of the shift register and registers which store values of coefficients increases, so number of registers and logical elements increases.

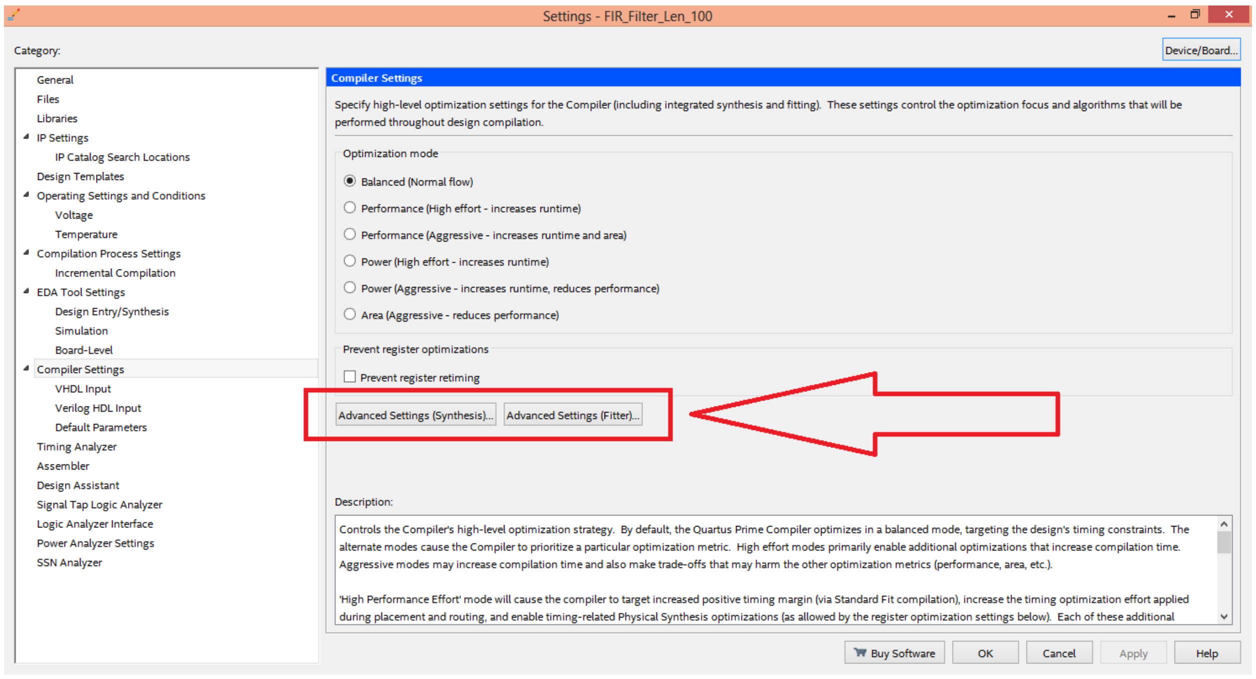
With increasing number of registers and logical elements, delay of FIR Filter increases and it leads to lower Fmax.

To achieve higher maximum frequency, we can change parameters which are related to placing and routing. To do this, follow the path shown in below images:



Fitter Status	Successful - Fri Oct 14 18:37:16 2022
Quartus Prime Version	20.1.0 Build 711 06/05/2020 SJ Lite Edition
Revision Name	FIR_Filter_Len_100
Top-level Entity Name	FIR_Filter
Family	Cyclone IV E
Device	EP4CE6E22A7
Timing Models	Final
Total logic elements	338 / 6,272 (5 %)
Total registers	4
Total pins	58 / 92 (63 %)
Total virtual pins	0
Total memory bits	0 / 276,480 (0 %)
Embedded Multiplier 9-bit elements	0 / 30 (0 %)
Total PLLs	0 / 2 (0 %)

```
Type   ID     Message
      18236 Number of processors has not been specified which may cause overloading on shared machines.  Set the global assignment NUM_PARALLEL_PROCESSORS in your
     204019 Generated file FIR_Filter_Len_100.vo in folder "D:/modelsim_ase/Projects/FIR_Filter_Len_100/simulation/modelsim/" for EDA simulation tool
            Quartus Prime EDA Netlist Writer was successful. 0 errors, 1 warning
     293000 Quartus Prime Full Compilation was successful. 0 errors, 78 warnings
```



Compiler Settings

Specify high-level optimization settings for the Compiler (including integrated synthesis and fitting). These settings control the optimization focus and algorithms that will be performed throughout design compilation.

Optimization mode

- ● Balanced (Normal flow)
- ○ Performance (High effort - increases runtime)
- ○ Performance (Aggressive - increases runtime and area)
- ○ Power (High effort - increases runtime)
- ○ Power (Aggressive - increases runtime, reduces performance)
- ○ Area (Aggressive - reduces performance)

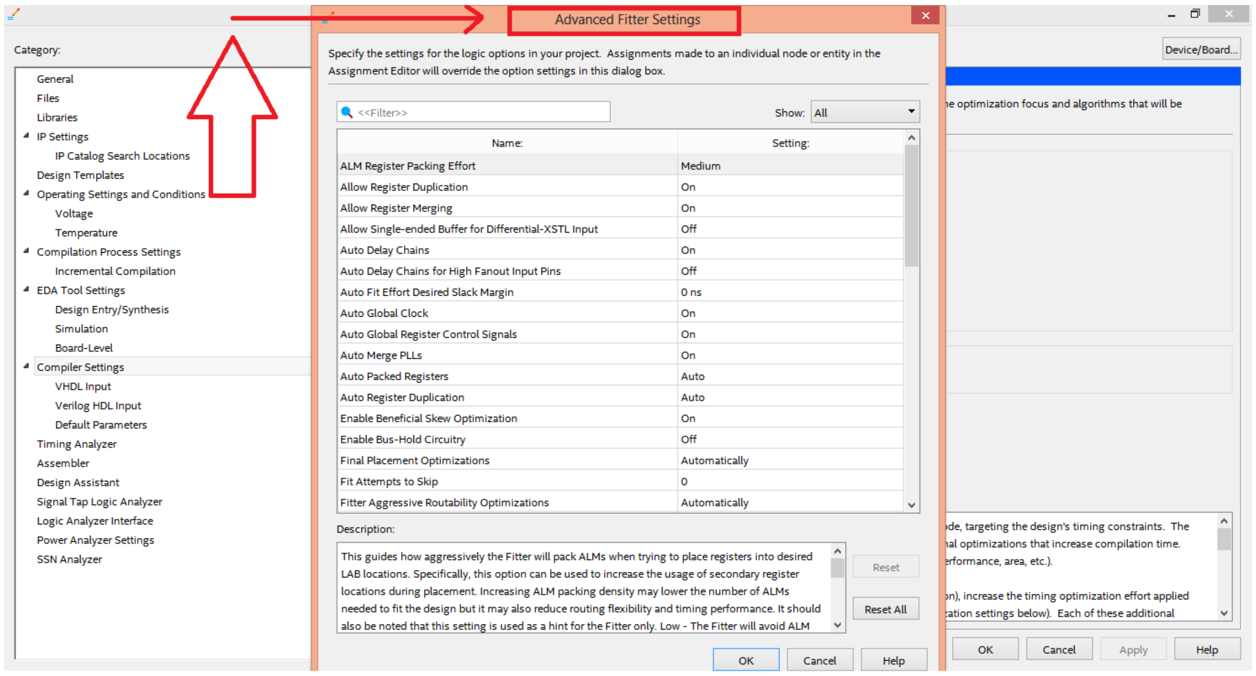
Prevent register optimizations

- ☐ Prevent register retiming

Description:

Controls the Compiler's high-level optimization strategy. By default, the Quartus Prime Compiler optimizes in a balanced mode, targeting the design's timing constraints. The alternate modes cause the Compiler to prioritize a particular optimization metric. High effort modes primarily enable additional optimizations that increase compilation time. Aggressive modes may increase compilation time and also make trade-offs that may harm the other optimization metrics (performance, area, etc.).

'High Performance Effort' mode will cause the compiler to target increased positive timing margin (via Standard Fit compilation), increase the timing optimization effort applied during placement and routing, and enable timing-related Physical Synthesis optimizations (as allowed by the register optimization settings below). Each of these additional



Specify the settings for the logic options in your project. Assignments made to an individual node or entity in the Assignment Editor will override the option settings in this dialog box.

Name:	Setting:
ALM Register Packing Effort	Medium
Allow Register Duplication	On
Allow Register Merging	On
Allow Single-ended Buffer for Differential-XSTL Input	Off
Auto Delay Chains	On
Auto Delay Chains for High Fanout Input Pins	Off
Auto Fit Effort Desired Slack Margin	0 ns
Auto Global Clock	On
Auto Global Register Control Signals	On
Auto Merge PLLs	On
Auto Packed Registers	Auto
Auto Register Duplication	Auto
Enable Beneficial Skew Optimization	On
Enable Bus-Hold Circuitry	Off
Final Placement Optimizations	Automatically
Fit Attempts to Skip	0
Fitter Aggressive Routability Optimizations	Automatically

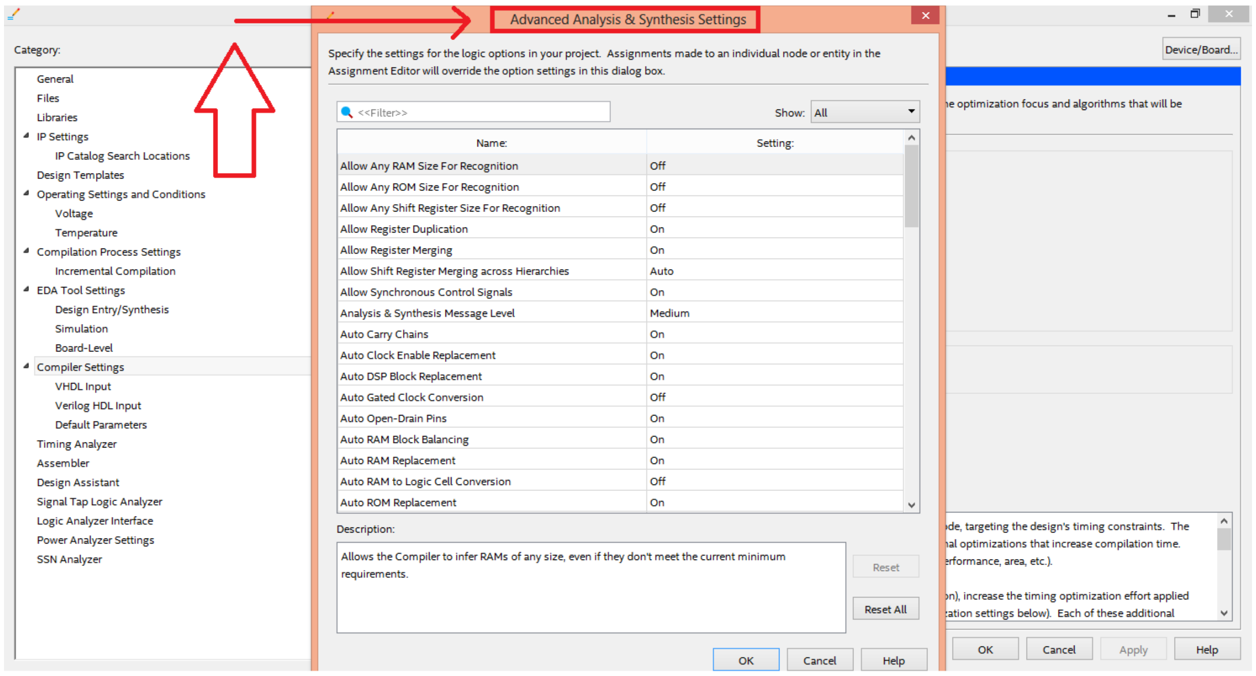
Description:

This guides how aggressively the Fitter will pack ALMs when trying to place registers into desired LAB locations. Specifically, this option can be used to increase the usage of secondary register locations during placement. Increasing ALM packing density may lower the number of ALMs needed to fit the design but it may also reduce routing flexibility and timing performance. It should also be noted that this setting is used as a hint for the Fitter only. Low - The Fitter will avoid ALM

he optimization focus and algorithms that will be

de, targeting the design's timing constraints. The
al optimizations that increase compilation time.
erformance, area, etc.).

n), increase the timing optimization effort applied
ation settings below). Each of these additional



Advanced Analysis & Synthesis Settings

Specify the settings for the logic options in your project. Assignments made to an individual node or entity in the Assignment Editor will override the option settings in this dialog box.

Category:

- General
- Files
- Libraries
- IP Settings
  - IP Catalog Search Locations
- Design Templates
- Operating Settings and Conditions
  - Voltage
  - Temperature
- Compilation Process Settings
  - Incremental Compilation
- EDA Tool Settings
  - Design Entry/Synthesis
  - Simulation
  - Board-Level
- Compiler Settings
  - VHDL Input
  - Verilog HDL Input
  - Default Parameters
- Timing Analyzer
- Assembler
- Design Assistant
- Signal Tap Logic Analyzer
- Logic Analyzer Interface
- Power Analyzer Settings
- SSN Analyzer

Show: All

Name:	Setting:
Allow Any RAM Size For Recognition	Off
Allow Any ROM Size For Recognition	Off
Allow Any Shift Register Size For Recognition	Off
Allow Register Duplication	On
Allow Register Merging	On
Allow Shift Register Merging across Hierarchies	Auto
Allow Synchronous Control Signals	On
Analysis & Synthesis Message Level	Medium
Auto Carry Chains	On
Auto Clock Enable Replacement	On
Auto DSP Block Replacement	On
Auto Gated Clock Conversion	Off
Auto Open-Drain Pins	On
Auto RAM Block Balancing	On
Auto RAM Replacement	On
Auto RAM to Logic Cell Conversion	Off
Auto ROM Replacement	On

Description:

Allows the Compiler to infer RAMs of any size, even if they don't meet the current minimum requirements.

ne optimization focus and algorithms that will be

de, targeting the design's timing constraints. The
al optimizations that increase compilation time.
erformance, area, etc.).

on), increase the timing optimization effort applied
zation settings below). Each of these additional