Techfest™
INDIAN INSTITUTE OF TECHNOLOGY, BOMBAY

VYORIUS

VLSI DESIGN PROJECTS/INTERNSHIP

College Name

→SRM Institute of Science and Technology.

Company Name

→Vyorius.

Domain

→VLSI design.

Project topics

1. 16-Bit RISC Processor
2. Digital Design

Software used

Hardware,

Mimas A7 Mini FPGA Development Board
Mimas A7 Mini is an easy-to-use FPGA Development board featuring Artix 7 FPGA (XC7A35T – FTG256C package) with FTDI's FT2232H Dual-Channel USB device. It is an Artix-7 based replacement and upgrades of Mimas Spartan 6 FPGA Board. It is specially designed for the development and integration of FPGA based accelerated features to other designs. The USB 2.0 host

interface based on popular FT2232H offers high bandwidth data transfer and board programming without the need for any external programming adapters.

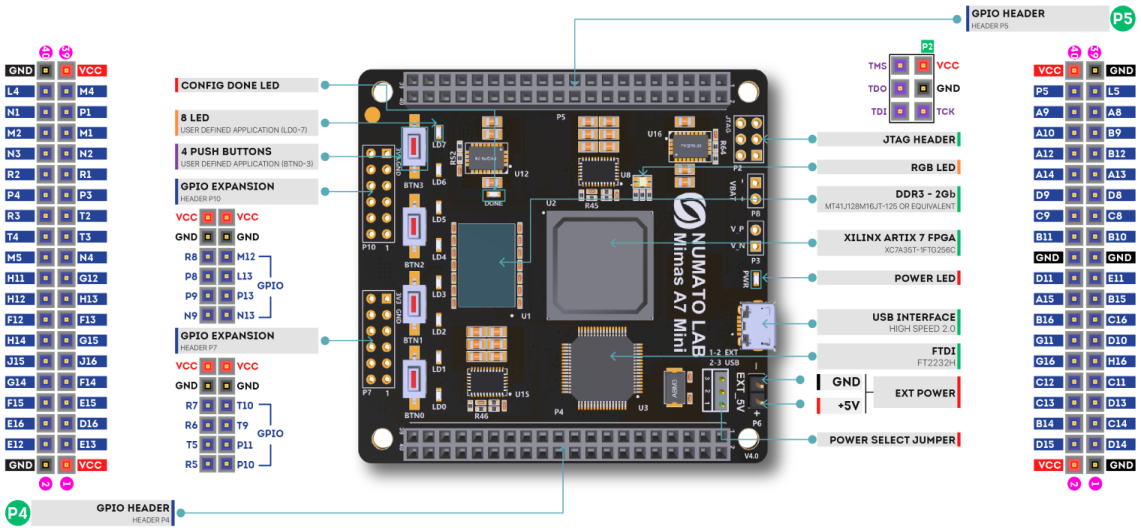
Features,

→ Device: Xilinx Artix 7 FPGA (XC7A35T-1FTG256C)
→ DDR3: 2Gb DDR3 (MT41J128M16JT-125 or equivalent)
→ Built-in programming interface. No expensive JTAG adapters needed for programming the board
→ Onboard 128Mb flash memory for FPGA configuration storage and custom user data storage
→ High-Speed USB 2.0 interface for On-board flash programming. FT2232H Channel B is dedicated to JTAG Programming. Channel A can be used for custom applications
→ 100MHz CMOS oscillator
→ 8 LEDs, 1 RGB LED and 4 Push Buttons for user-defined purposes
→ FPGA configuration via JTAG and USB
→ Maximum IOs for user-defined purposes

FPGA – 70 IOs (35 professionally length matched Differential Pairs) and two 2×6 Expansion Headers

Applications,

→ Product Prototype Development
→ Accelerated computing integration
→ Development and testing of custom embedded processors
→ Communication devices development
→ Educational tool for Schools and Universities



Specifications,

Attribute	Value
Dimensions	6 × 4 × 1 in
FPGA	**XC7A35T – 1FTG256C**
Memory	**DDR3 – 2Gb**
Configuration Options	**JTAG**, **USB**
Host Interface	**USB 2.0**
Primary Clock Frequency	**100MHz**
Number Of GPIOs (Max)	**70**
Number Of Diff Pairs	**35**

Software

Icarus Verilog is a Verilog simulation and synthesis tool. It operates as a compiler, compiling source code written in Verilog (IEEE-1364) into some target format. For batch simulation, the compiler can generate an intermediate form called *vvp assembly*. This intermediate form is executed by the ``vvp'' command. For synthesis, the compiler generates netlists in the desired format.

The compiler proper is intended to parse and elaborate design descriptions written to the IEEE standard *IEEE Std 1364-2005*. This is a fairly large and complex standard, so it will take some time to fill all the dark alleys of the standard, but that's the goal.

Icarus Verilog is a work in progress, and since the language standard is not standing still either, it probably always will be. That is as it should be. However, I will make stable releases from time to time, and will endeavour to not retract any features that appear in these stable releases. The quick links above will show the current stable release.

The main porting target is Linux, although it works well on many similar operating systems. Various people have contributed precompiled binaries of stable releases for a variety of targets. These releases are ported by volunteers, so what binaries are available depends on who takes the time to do the packaging. *Icarus Verilog* has been ported to That Other Operating System, as a command line tool, and there are installers for users without compilers. You can compile it entirely with free tools, too, although there are precompiled binaries of stable releases.

Vivado Design Suite is a software suite produced by Xilinx for synthesis and analysis of HDL designs, superseding Xilinx ISE with additional features for system on a chip development and high-level synthesis.[1][5][6][7] Vivado represents a ground-up rewrite and re-thinking of the entire design flow (compared to ISE).[8][9][10]

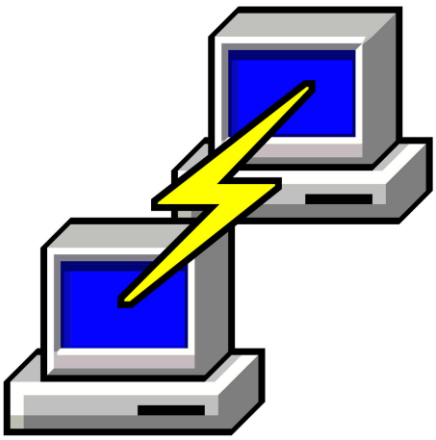
Like the later versions of ISE, Vivado includes the in-built logic simulator ISIM.[11] Vivado also introduces high-level synthesis, with a toolchain that converts C code into programmable logic.[6]

Replacing the 15-year-old ISE with Vivado Design Suite took 1000 person-years and cost US$200 million.

NUMATO LAB®

Tenagra is an FPGA System management tool for configuring and communicating with Numato Lab's supported FPGA modules and development platforms. This software is designed to be a single interface for managing the devices and exercising some of the available features. Currently, Tenagra supports configuring the FPGA module/board (programming) and Memory Exerciser that can transfer data between various memories on the device. This includes both external DDR Memory and Block RAM available within the FPGA device. With Tenagra, you can create multiple configuration setups with different bitstreams and settings for each device model so that switching between multiple bitstreams is a breeze. This is especially helpful during development where the device may need to be reprogrammed with various bitstreams repeatedly.

Driver for Mimas A7 Mini Module



PuTTY is a free implementation of SSH **(and telnet) for PCs running Microsoft Windows** (it also includes an xterm terminal emulator). You will find PuTTY useful if you want to access an account on a Unix or other multi-user system from a PC (for example your own or one in an internet cafe).

Things to know

What Is RISC?

A Reduced Instruction Set Computer is a type of microprocessor architecture that utilizes a small, highly-optimized set of instructions rather than the highly-specialized set of instructions typically found in other architectures. RISC is an alternative to the Complex Instruction Set Computing (CISC) architecture and is often considered the most efficient CPU architecture technology available today.

With RISC, a central processing unit (CPU) implements the processor design principle of simplified instructions that can do less but can execute more rapidly. The result is improved performance. A key RISC feature is that it allows developers to increase the register set and increase internal parallelism by increasing the number of parallel threads executed by the CPU and increasing the speed of the CPU's executing instructions. ARM, or “Advanced RISC Machine” is a specific family of instruction set architecture that’s based on reduced instruction set architecture developed by Arm Ltd. Processors based on this architecture are common in smartphones, tablets, laptops, gaming consoles and desktops, as well as a growing number of other intelligent devices.

Why Is RISC Important?

RISC provides high performance per watt for battery operated devices where energy efficiency is key. A RISC processor executes one action per instruction. By taking just one cycle to complete, operation execution time is optimized.

Because the architecture uses a fixed length of instruction, it’s easier to pipeline. And because it lacks complex instruction decoding logic, it supports more registers and spends less time on loading and storing values to memory.

For chip designers, RISC processors simplify the design and deployment process and provide a lower per-chip cost due to the smaller components required. Because of the reduced instruction set and simple decoding logic, less chip space is used, fewer transistors are required, and more general-purpose registers can fit into the central processing unit.

1	1	0	0	1	Jump (Register)

Design Modules

1. **Instruction Decoder**
2. **Control Unit**
3. **ALU**
4. **Registers handler**
5. **Program counter**
6. **RAM**

Testing Modules

1. **Top test bench**
2. **Instruction Decoder**
3. **Control Unit**
4. **ALU**
5. **Registers handler**
6. **Program counter**
7. **RAM**

1. 16-Bit RISC Processor

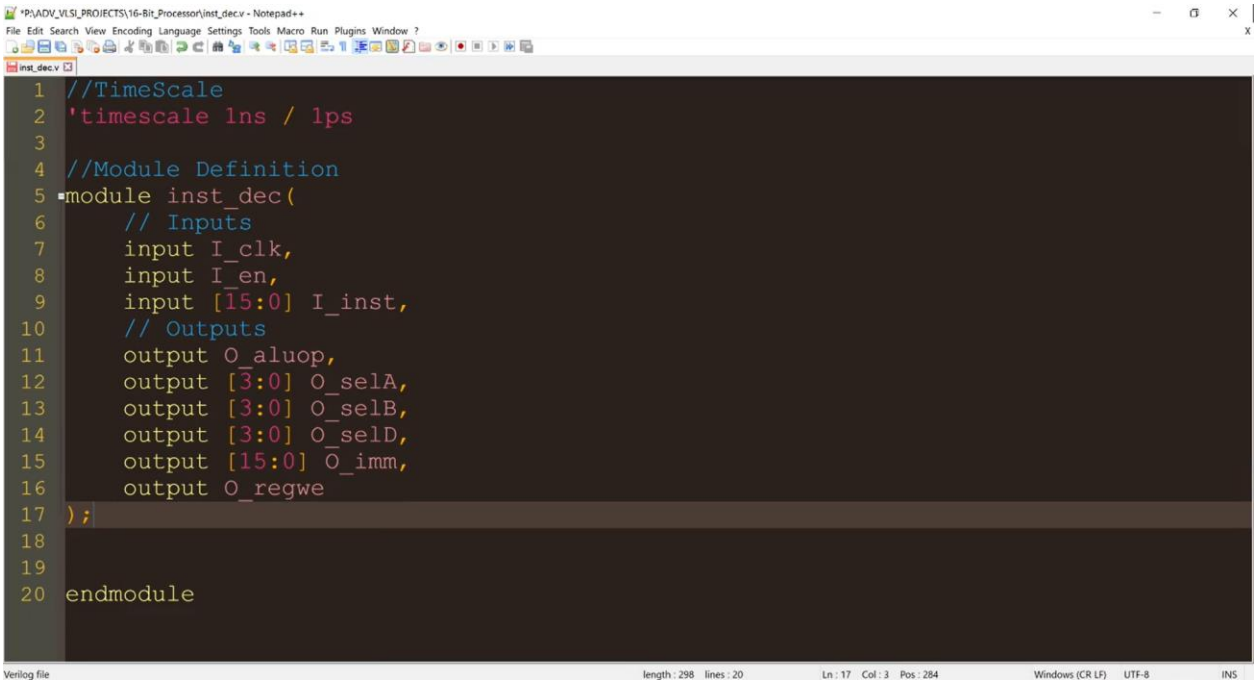
Software to download

- Notepad++

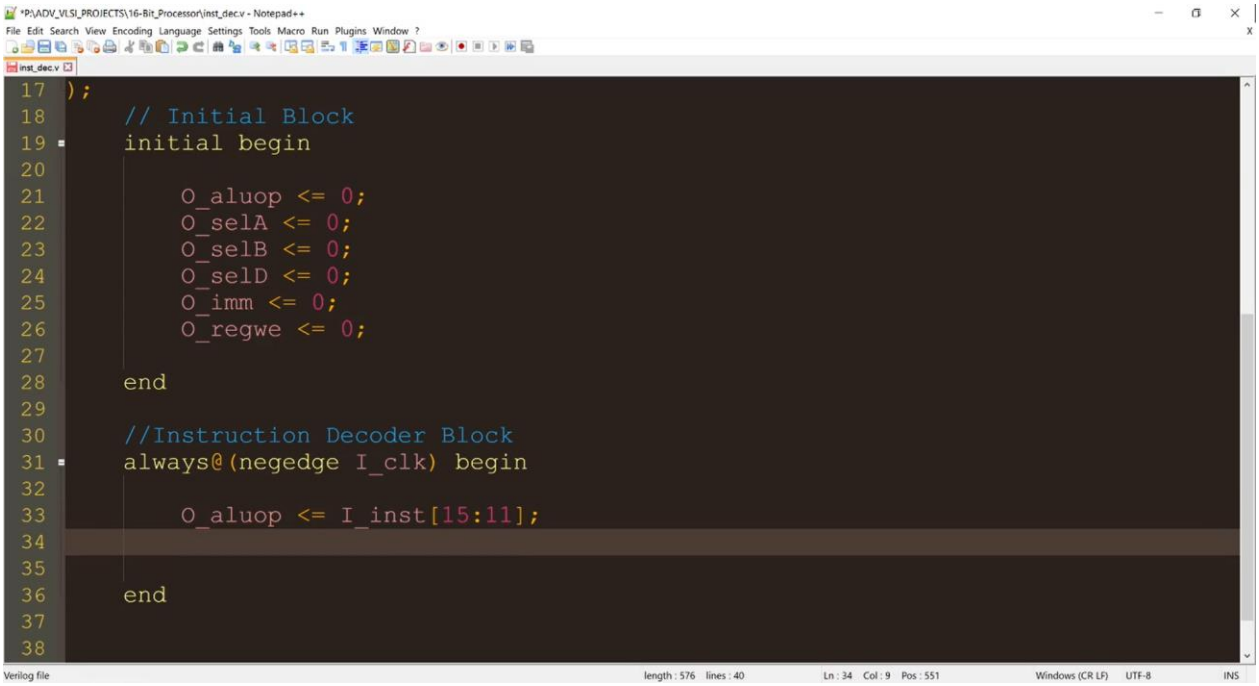
Step 1- Write the verilog program in notepad ++ .

DESIGN CODES

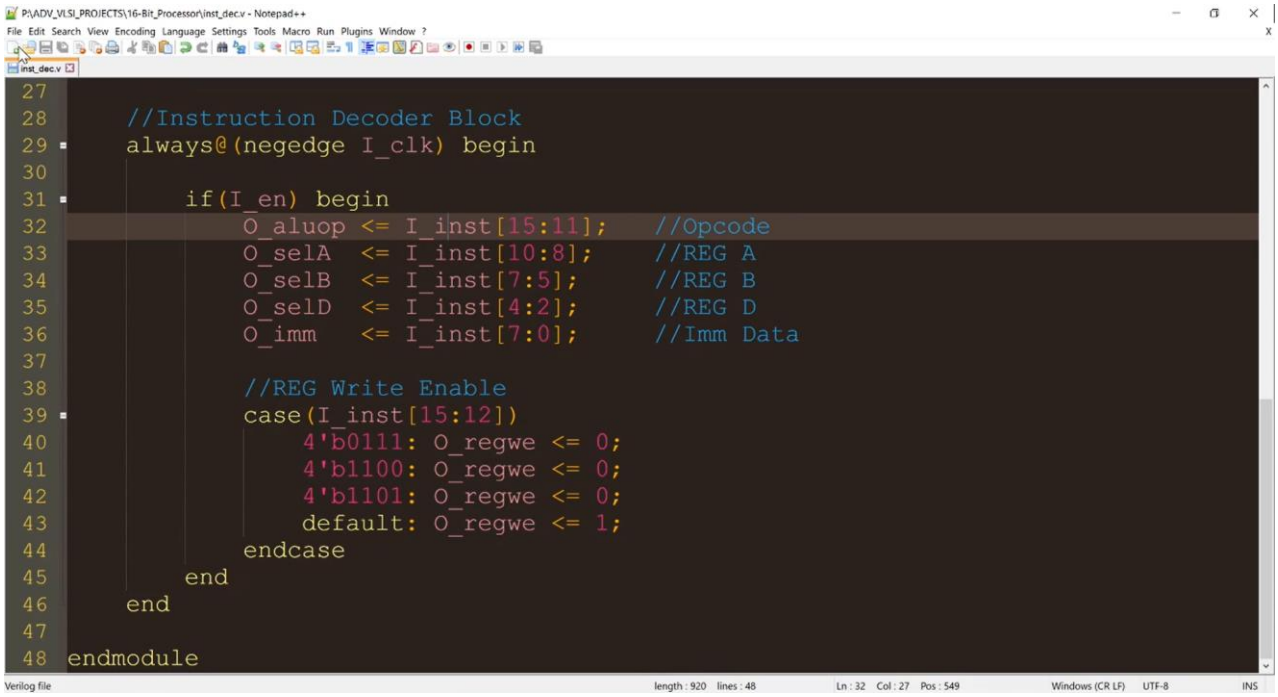
Module inst_dec



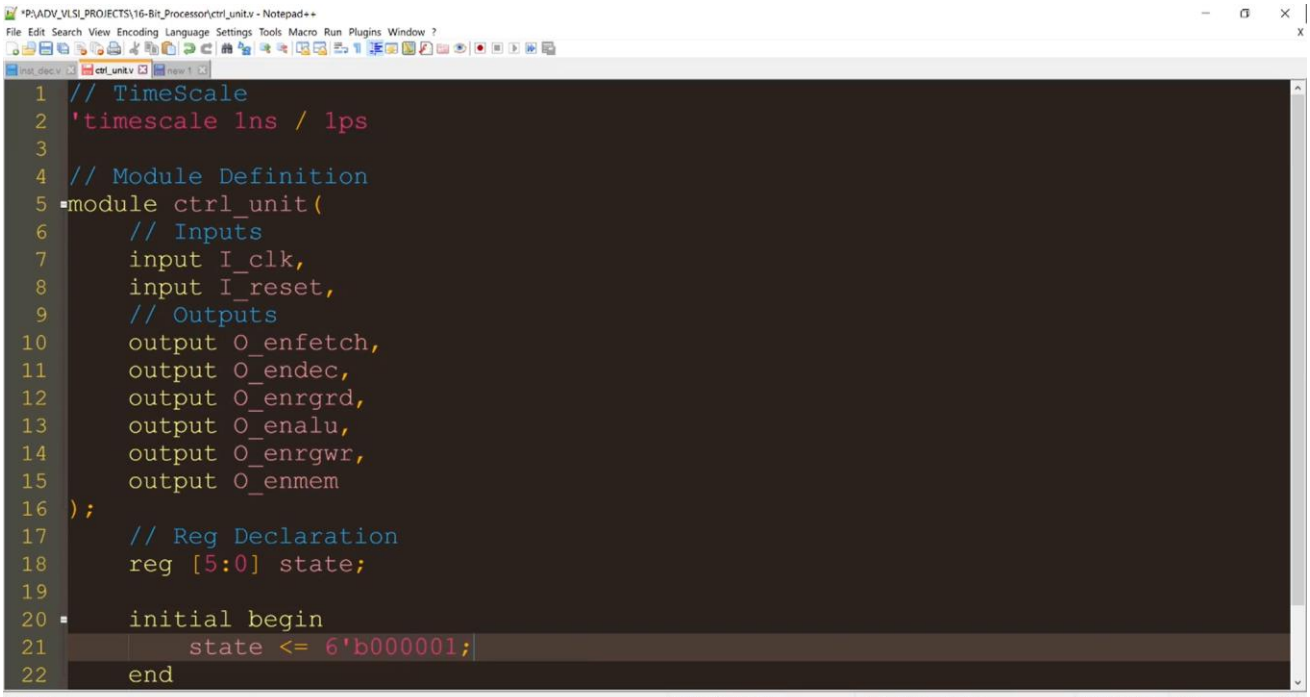
```
1  //TimeScale
2  'timescale 1ns / 1ps
3
4  //Module Definition
5  module inst_dec(
6      // Inputs
7      input I_clk,
8      input I_en,
9      input [15:0] I_inst,
10     // Outputs
11     output O_aluop,
12     output [3:0] O_selA,
13     output [3:0] O_selB,
14     output [3:0] O_selD,
15     output [15:0] O_imm,
16     output O_regwe
17 );
18
19
20 endmodule
```

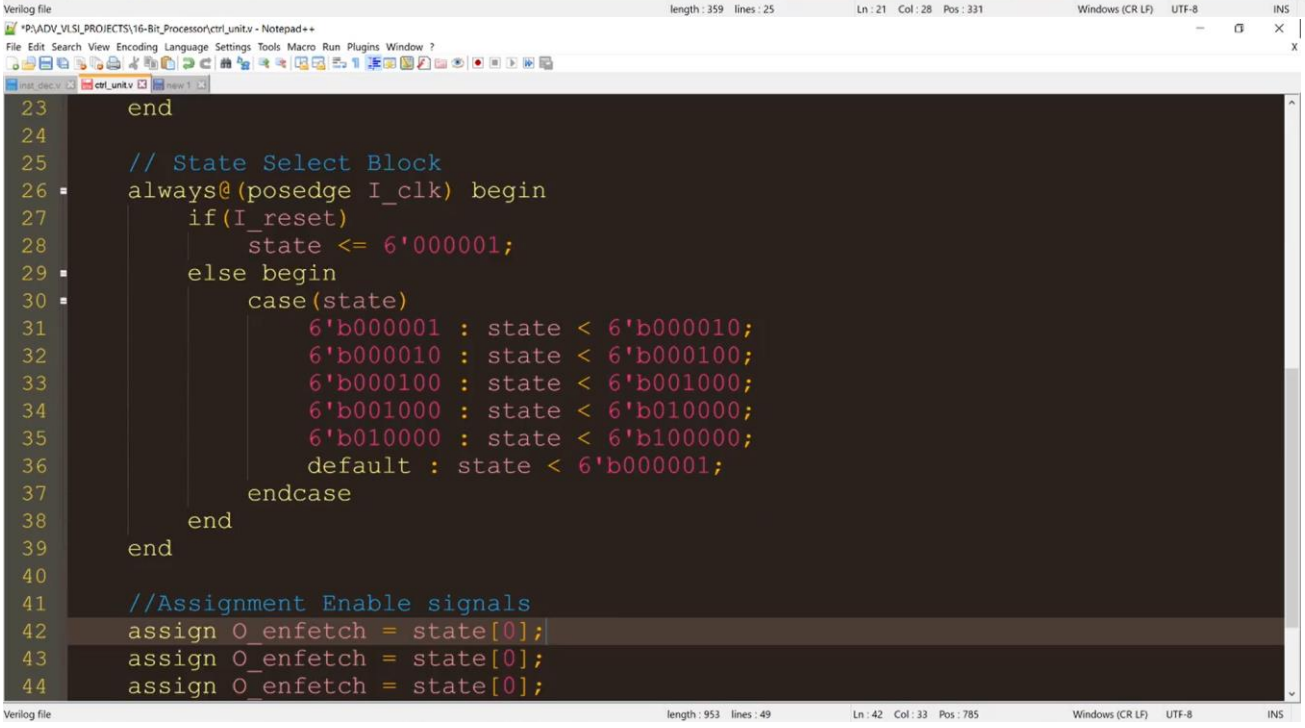



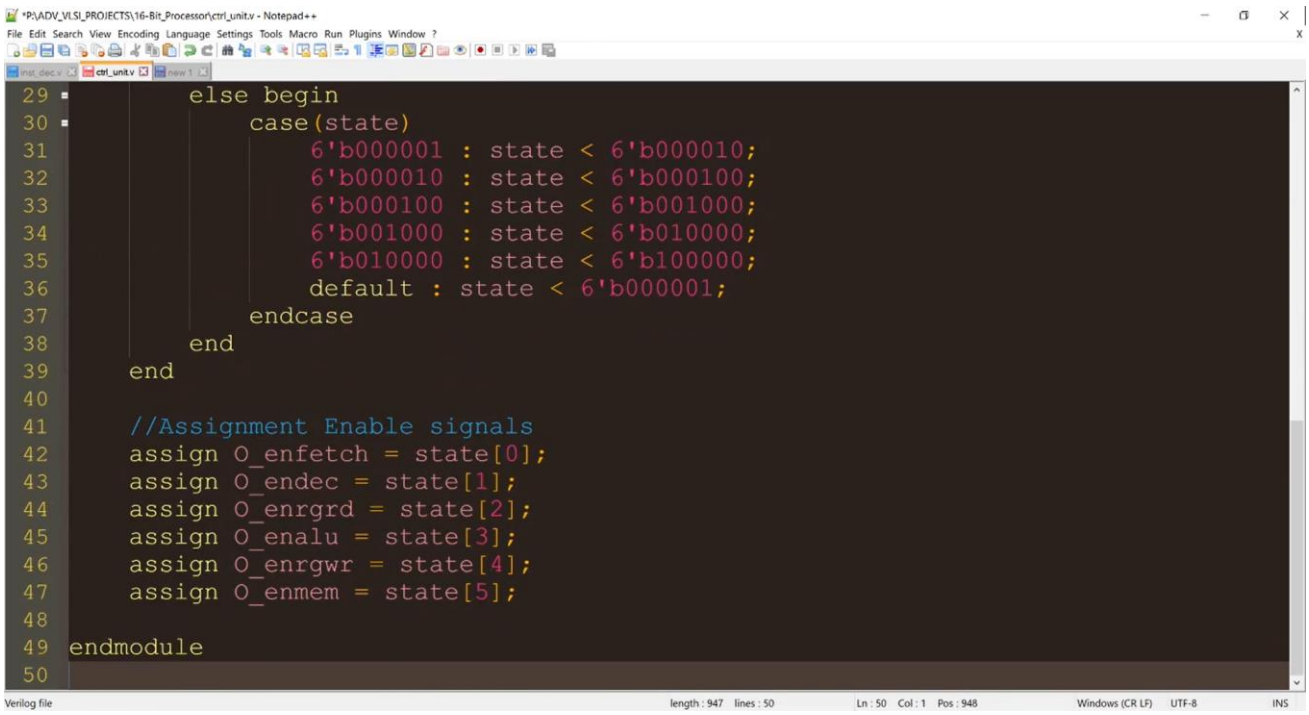
```
17 );
18     // Initial Block
19     initial begin
20
21         O_aluop <= 0;
22         O_selA <= 0;
23         O_selB <= 0;
24         O_selD <= 0;
25         O_imm <= 0;
26         O_regwe <= 0;
27
28     end
29
30     //Instruction Decoder Block
31     always@(negedge I_clk) begin
32
33         O_aluop <= I_inst[15:11];
34
35
36     end
37
38
```



Module ctrl_unit

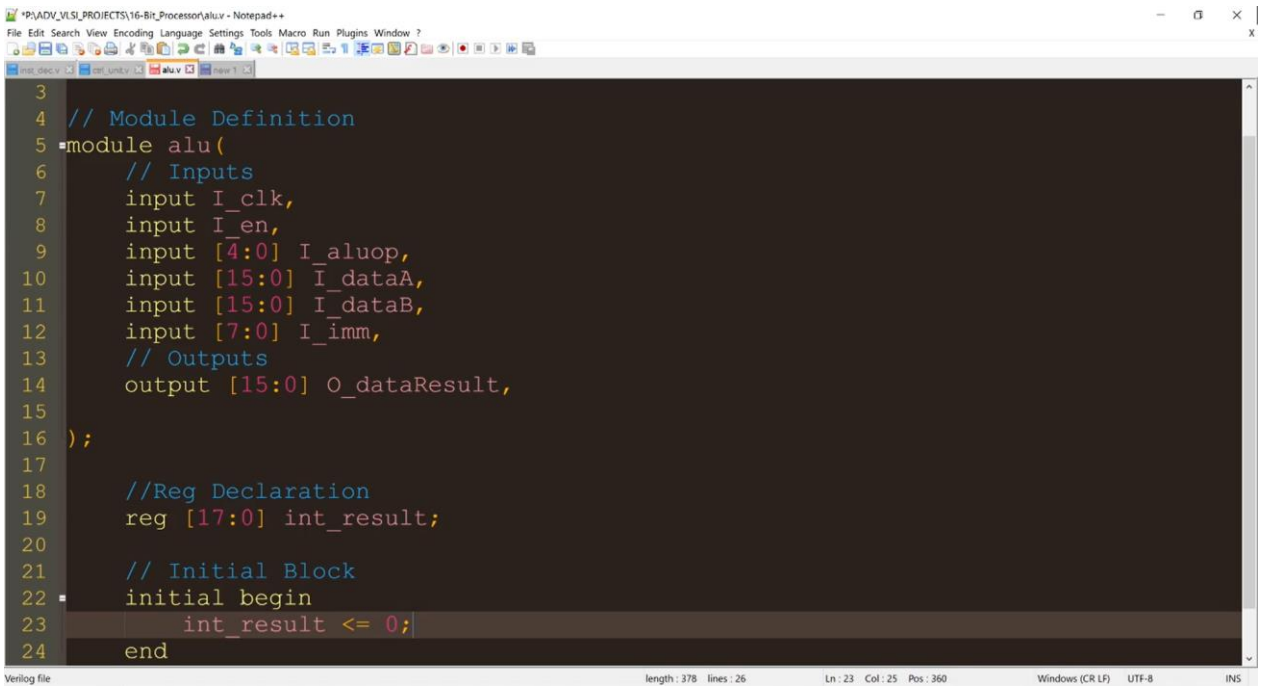

*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\ctrl_unit.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
inst_dec.v
ctrl_unit.v
new 1
// TimeScale
'timescale 1ns / 1ps
// Module Definition
module ctrl_unit(
// Inputs
input I_clk,
input I_reset,
// Outputs
output O_enfetch,
output O_endec,
output O_enrgrd,
output O_enalu,
output O_enrgwr,
output O_enmem
);
// Reg Declaration
reg [5:0] state;
initial begin
state <= 6'b000001;
end
Verilog file
length : 359 lines : 25
Ln : 21 Col : 28 Pos : 331
Windows (CR LF)
UTF-8
INS


*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\ctrl_unit.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
inst_dec.v
ctrl_unit.v
new 1
end
// State Select Block
always@(posedge I_clk) begin
if(I_reset)
state <= 6'000001;
else begin
case(state)
6'b000001 : state < 6'b000010;
6'b000010 : state < 6'b000100;
6'b000100 : state < 6'b001000;
6'b001000 : state < 6'b010000;
6'b010000 : state < 6'b100000;
default : state < 6'b000001;
endcase
end
end
//Assignment Enable signals
assign O_enfetch = state[0];
assign O_enfetch = state[0];
assign O_enfetch = state[0];
Verilog file
length : 953 lines : 49
Ln : 42 Col : 33 Pos : 785
Windows (CR LF)
UTF-8
INS

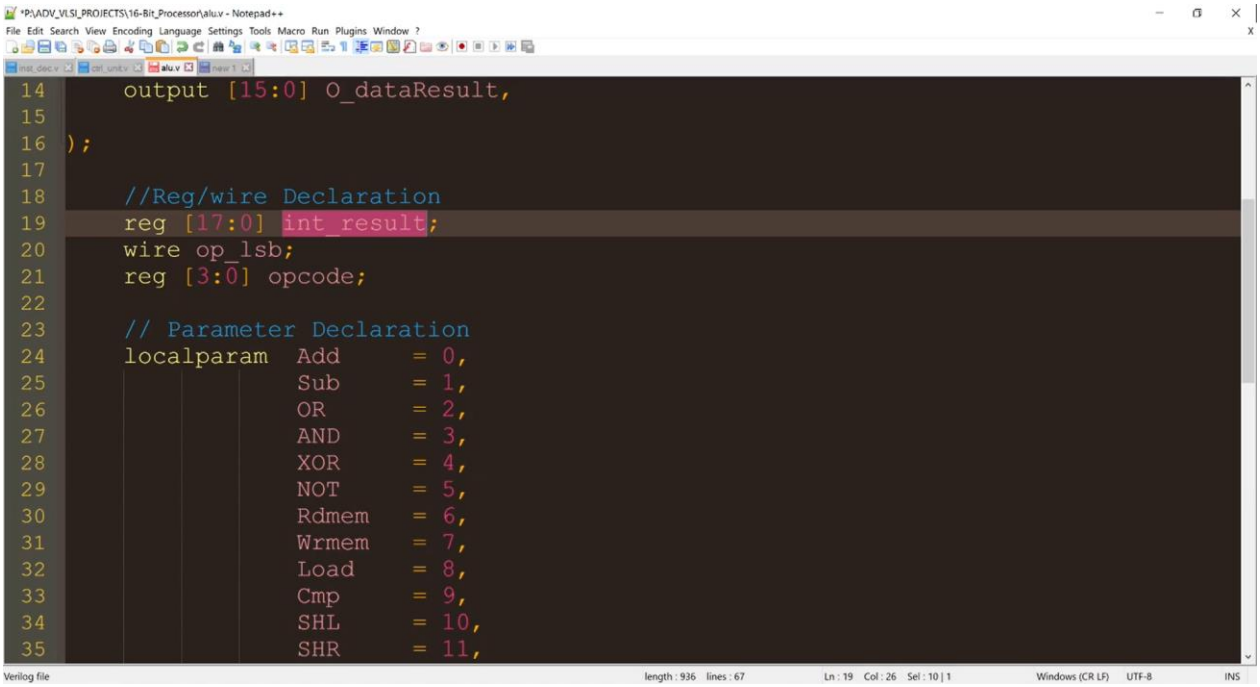


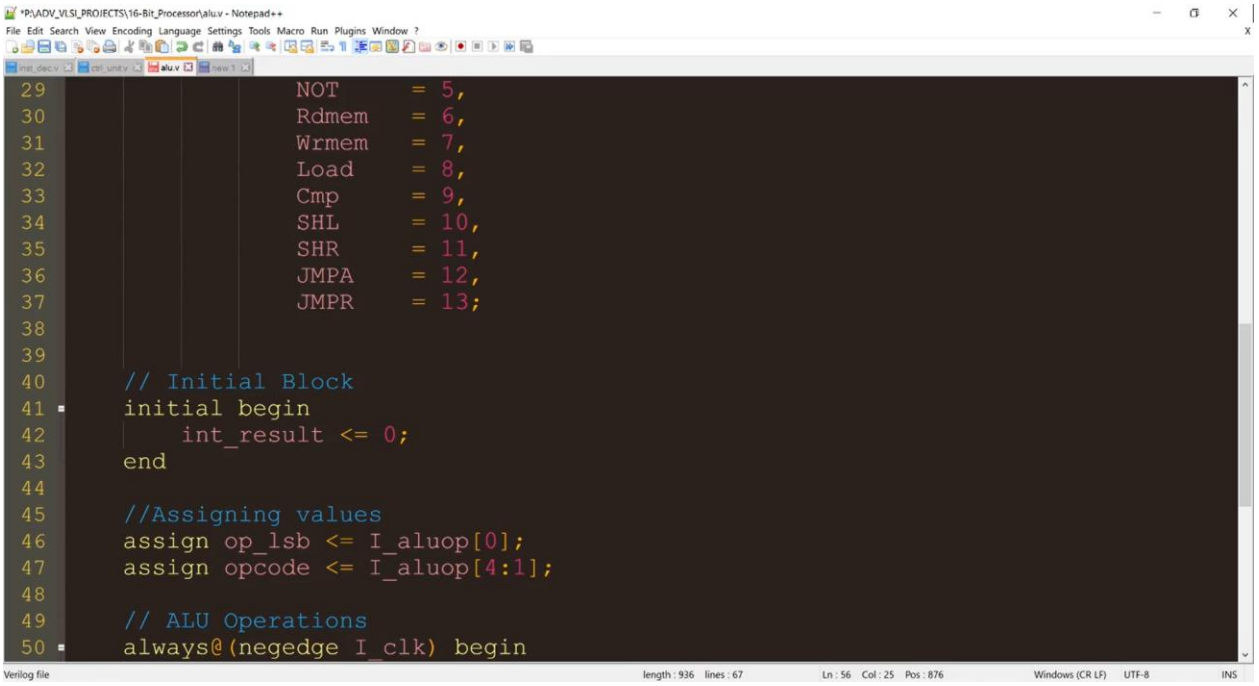
```
        else begin
            case(state)
                6'b000001 : state < 6'b000010;
                6'b000010 : state < 6'b000100;
                6'b000100 : state < 6'b001000;
                6'b001000 : state < 6'b010000;
                6'b010000 : state < 6'b100000;
                default : state < 6'b000001;
            endcase
        end
    end

    //Assignment Enable signals
    assign O_enfetch = state[0];
    assign O_endec = state[1];
    assign O_enrgrd = state[2];
    assign O_enalu = state[3];
    assign O_enrgwr = state[4];
    assign O_enmem = state[5];

endmodule
```

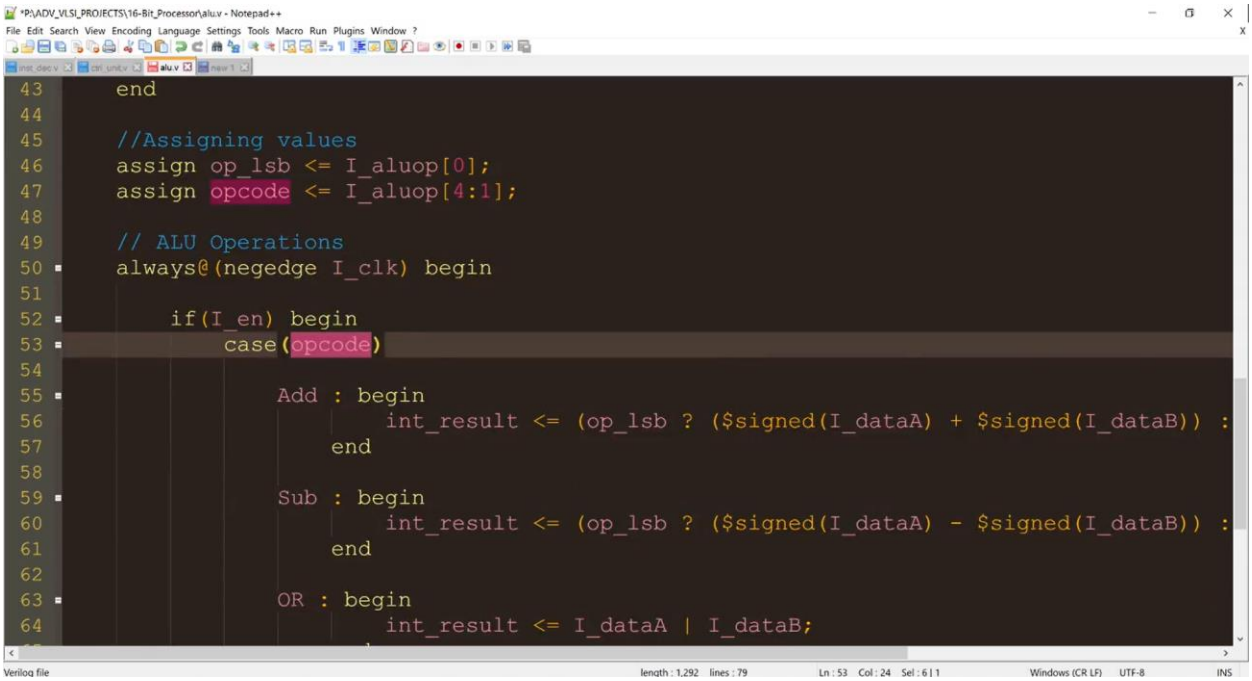
Module alu

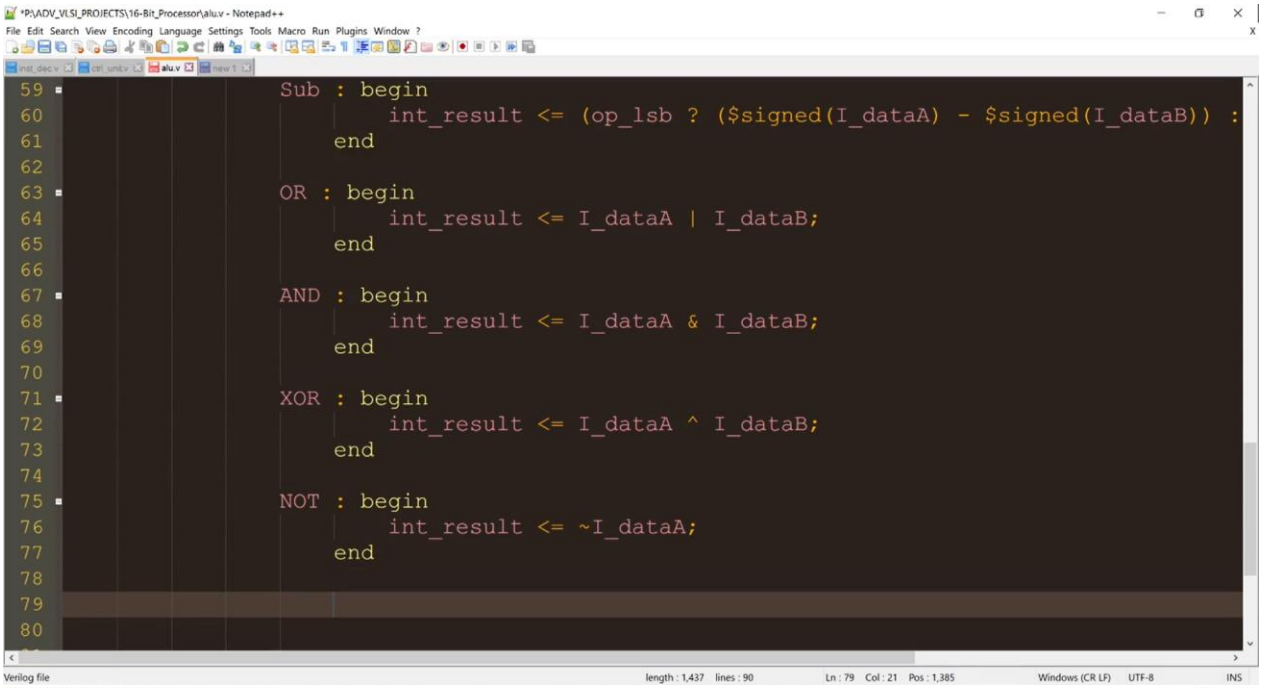


```
// Module Definition
module alu(
    // Inputs
    input I_clk,
    input I_en,
    input [4:0] I_aluop,
    input [15:0] I_dataA,
    input [15:0] I_dataB,
    input [7:0] I_imm,
    // Outputs
    output [15:0] O_dataResult,

);

    //Reg Declaration
    reg [17:0] int_result;

    // Initial Block
    initial begin
        int_result <= 0;
    end
```


*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\alu.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
inst_dec.v
ctrl_unit.v
alu.v
new 1
output [15:0] O_dataResult,
);
//Reg/wire Declaration
reg [17:0] int_result;
wire op_lsb;
reg [3:0] opcode;
// Parameter Declaration
localparam Add = 0,
Sub = 1,
OR = 2,
AND = 3,
XOR = 4,
NOT = 5,
Rdmem = 6,
Wrmem = 7,
Load = 8,
Cmp = 9,
SHL = 10,
SHR = 11,
Verilog file
length : 936 lines : 67
Ln : 19 Col : 26 Sel : 10 | 1
Windows (CR LF)
UTF-8
INS

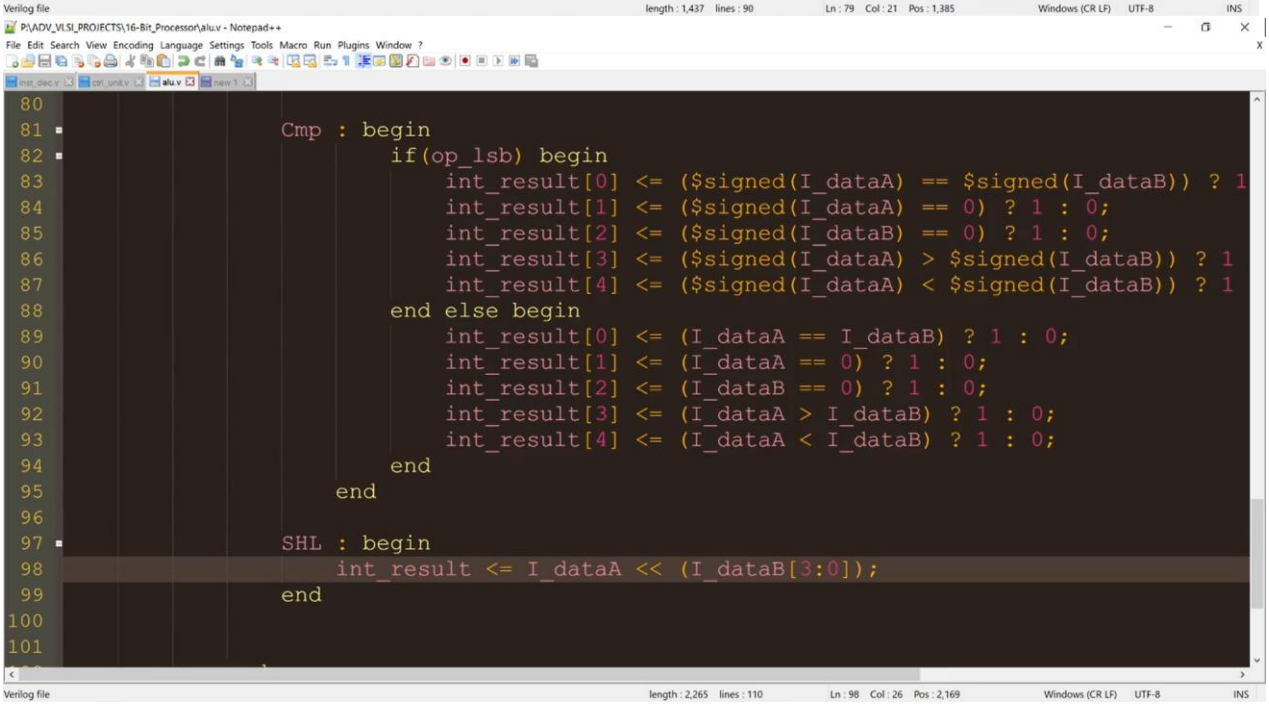

*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\alu.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
inst_dec.v
ctrl_unit.v
alu.v
new 1
NOT = 5,
Rdmem = 6,
Wrmem = 7,
Load = 8,
Cmp = 9,
SHL = 10,
SHR = 11,
JMPA = 12,
JMPR = 13;
// Initial Block
initial begin
int_result <= 0;
end
//Assigning values
assign op_lsb <= I_aluop[0];
assign opcode <= I_aluop[4:1];
// ALU Operations
always@(negedge I_clk) begin
Verilog file
length : 936 lines : 67
Ln : 56 Col : 25 Pos : 876
Windows (CR LF)
UTF-8
INS


*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\alu.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
inst_dec.v
ctrl_unit.v
alu.v
new 1
43 end
44
45 //Assigning values
46 assign op_lsb <= I_aluop[0];
47 assign opcode <= I_aluop[4:1];
48
49 // ALU Operations
50 always@(negedge I_clk) begin
51
52 if(I_en) begin
53 case(opcode)
54
55 Add : begin
56 int_result <= (op_lsb ? ($signed(I_dataA) + $signed(I_dataB)) :
57 end
58
59 Sub : begin
60 int_result <= (op_lsb ? ($signed(I_dataA) - $signed(I_dataB)) :
61 end
62
63 OR : begin
64 int_result <= I_dataA | I_dataB;
Verilog file
length : 1,292 lines : 79
Ln : 53 Col : 24 Sel : 6 | 1
Windows (CR LF)
UTF-8
INS



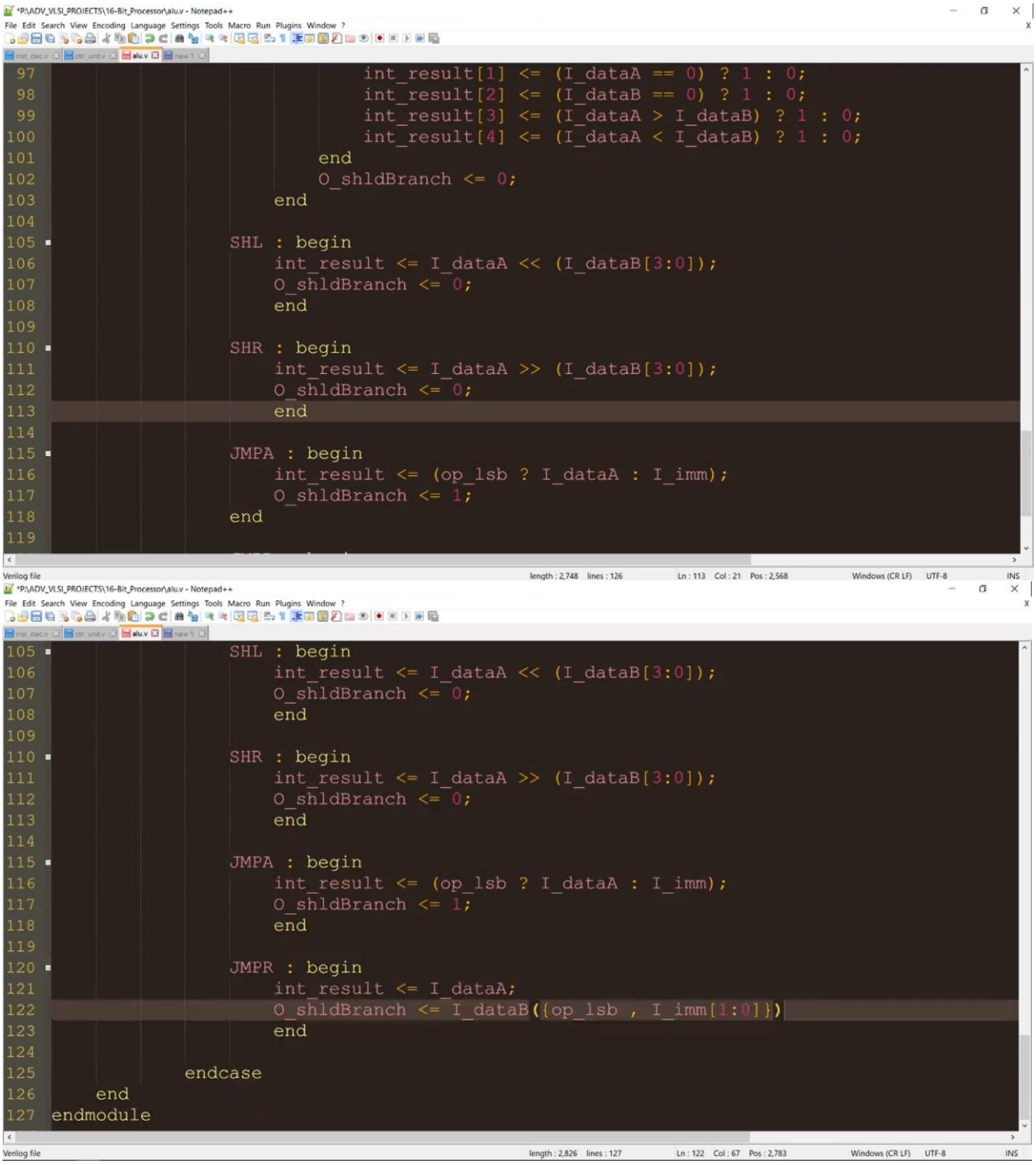
```
59          Sub : begin
60                  int_result <= (op_lsb ? ($signed(I_dataA) - $signed(I_dataB)) :
61              end
62
63          OR : begin
64                  int_result <= I_dataA | I_dataB;
65              end
66
67          AND : begin
68                  int_result <= I_dataA & I_dataB;
69              end
70
71          XOR : begin
72                  int_result <= I_dataA ^ I_dataB;
73              end
74
75          NOT : begin
76                  int_result <= ~I_dataA;
77              end
78
79
80
```

Verilog file | length : 1,437 lines : 90 | Ln : 79 Col : 21 Pos : 1,385 | Windows (CR LF) UTF-8 INS



```
80
81          Cmp : begin
82              if(op_lsb) begin
83                  int_result[0] <= ($signed(I_dataA) == $signed(I_dataB)) ? 1
84                  int_result[1] <= ($signed(I_dataA) == 0) ? 1 : 0;
85                  int_result[2] <= ($signed(I_dataB) == 0) ? 1 : 0;
86                  int_result[3] <= ($signed(I_dataA) > $signed(I_dataB)) ? 1
87                  int_result[4] <= ($signed(I_dataA) < $signed(I_dataB)) ? 1
88              end else begin
89                  int_result[0] <= (I_dataA == I_dataB) ? 1 : 0;
90                  int_result[1] <= (I_dataA == 0) ? 1 : 0;
91                  int_result[2] <= (I_dataB == 0) ? 1 : 0;
92                  int_result[3] <= (I_dataA > I_dataB) ? 1 : 0;
93                  int_result[4] <= (I_dataA < I_dataB) ? 1 : 0;
94              end
95          end
96
97          SHL : begin
98              int_result <= I_dataA << (I_dataB[3:0]);
99          end
100
101
```

Verilog file | length : 2,265 lines : 110 | Ln : 98 Col : 26 Pos : 2,169 | Windows (CR LF) UTF-8 INS



```
                                int_result[1] <= (I_dataA == 0) ? 1 : 0;
                                int_result[2] <= (I_dataB == 0) ? 1 : 0;
                                int_result[3] <= (I_dataA > I_dataB) ? 1 : 0;
                                int_result[4] <= (I_dataA < I_dataB) ? 1 : 0;
                            end
                            O_shldBranch <= 0;
                        end

                    SHL : begin
                        int_result <= I_dataA << (I_dataB[3:0]);
                        O_shldBranch <= 0;
                        end

                    SHR : begin
                        int_result <= I_dataA >> (I_dataB[3:0]);
                        O_shldBranch <= 0;
                        end

                    JMPA : begin
                        int_result <= (op_lsb ? I_dataA : I_imm);
                        O_shldBranch <= 1;
                    end
```

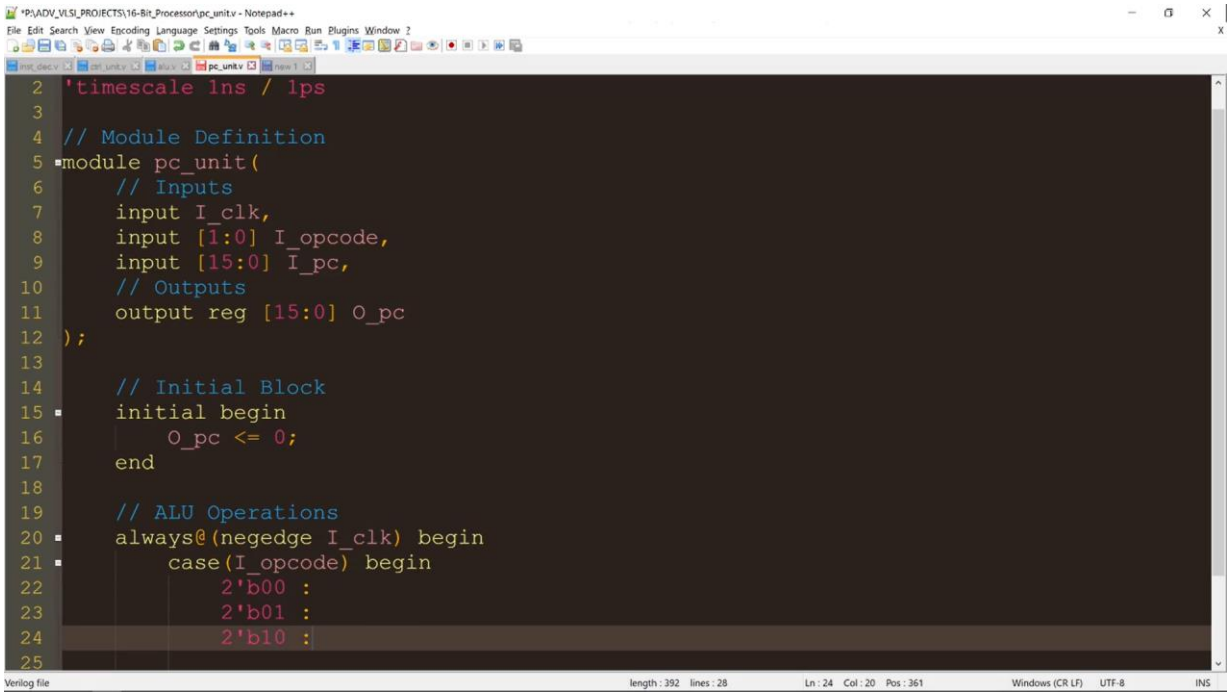


```
                    SHL : begin
                        int_result <= I_dataA << (I_dataB[3:0]);
                        O_shldBranch <= 0;
                        end

                    SHR : begin
                        int_result <= I_dataA >> (I_dataB[3:0]);
                        O_shldBranch <= 0;
                        end

                    JMPA : begin
                        int_result <= (op_lsb ? I_dataA : I_imm);
                        O_shldBranch <= 1;
                        end

                    JMPR : begin
                        int_result <= I_dataA;
                        O_shldBranch <= I_dataB({op_lsb , I_imm[1:0]})
                        end

            endcase
    end
endmodule
```

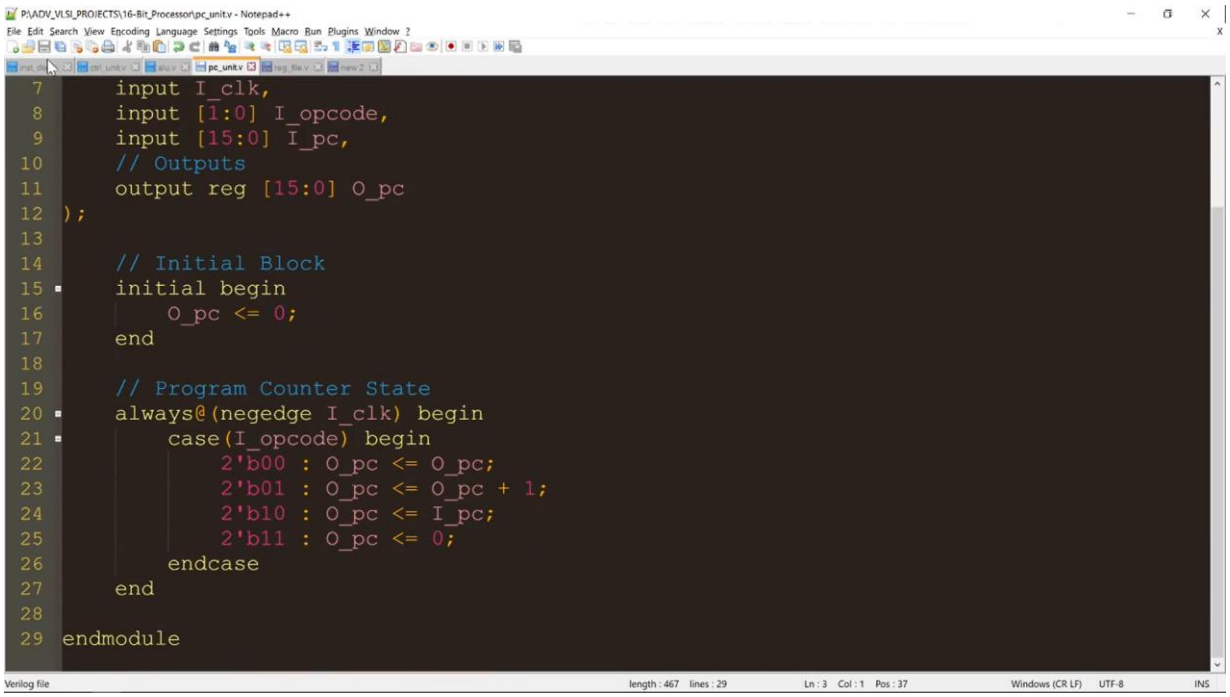
PS- O_shldBranch defined at the input and also in the ALU operations for the input one as

O_shldBranch <= 0;

Module pc_unit

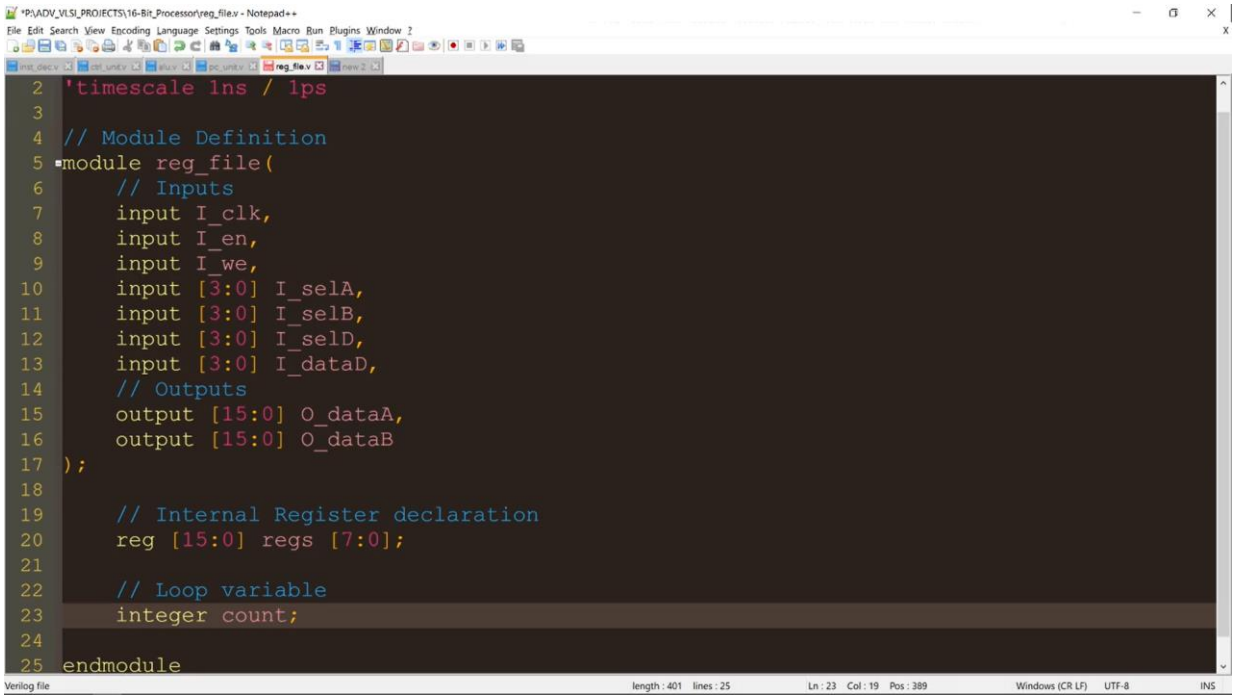


```
'timescale 1ns / 1ps

// Module Definition
module pc_unit(
    // Inputs
    input I_clk,
    input [1:0] I_opcode,
    input [15:0] I_pc,
    // Outputs
    output reg [15:0] O_pc
);

    // Initial Block
    initial begin
        O_pc <= 0;
    end

    // ALU Operations
    always@(negedge I_clk) begin
        case(I_opcode) begin
            2'b00 :
            2'b01 :
            2'b10 :
```

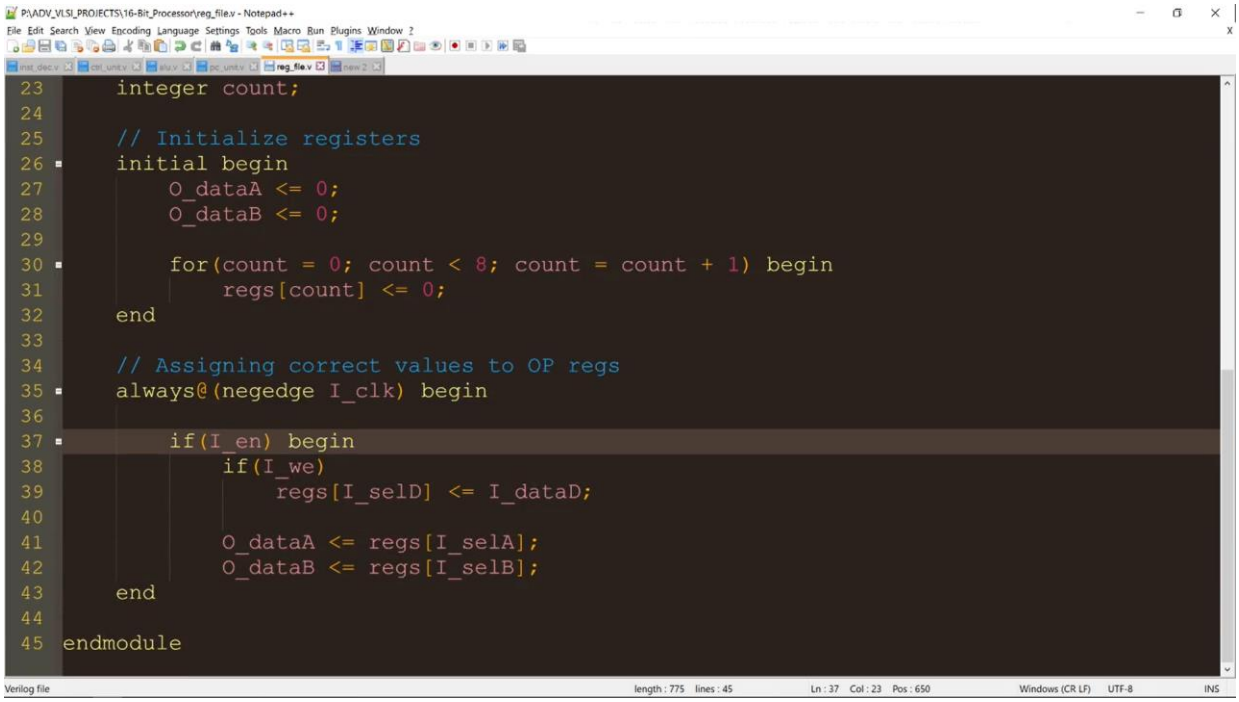


```
    input I_clk,
    input [1:0] I_opcode,
    input [15:0] I_pc,
    // Outputs
    output reg [15:0] O_pc
);

    // Initial Block
    initial begin
        O_pc <= 0;
    end

    // Program Counter State
    always@(negedge I_clk) begin
        case(I_opcode) begin
            2'b00 : O_pc <= O_pc;
            2'b01 : O_pc <= O_pc + 1;
            2'b10 : O_pc <= I_pc;
            2'b11 : O_pc <= 0;
        endcase
    end

endmodule
```

Module reg_file

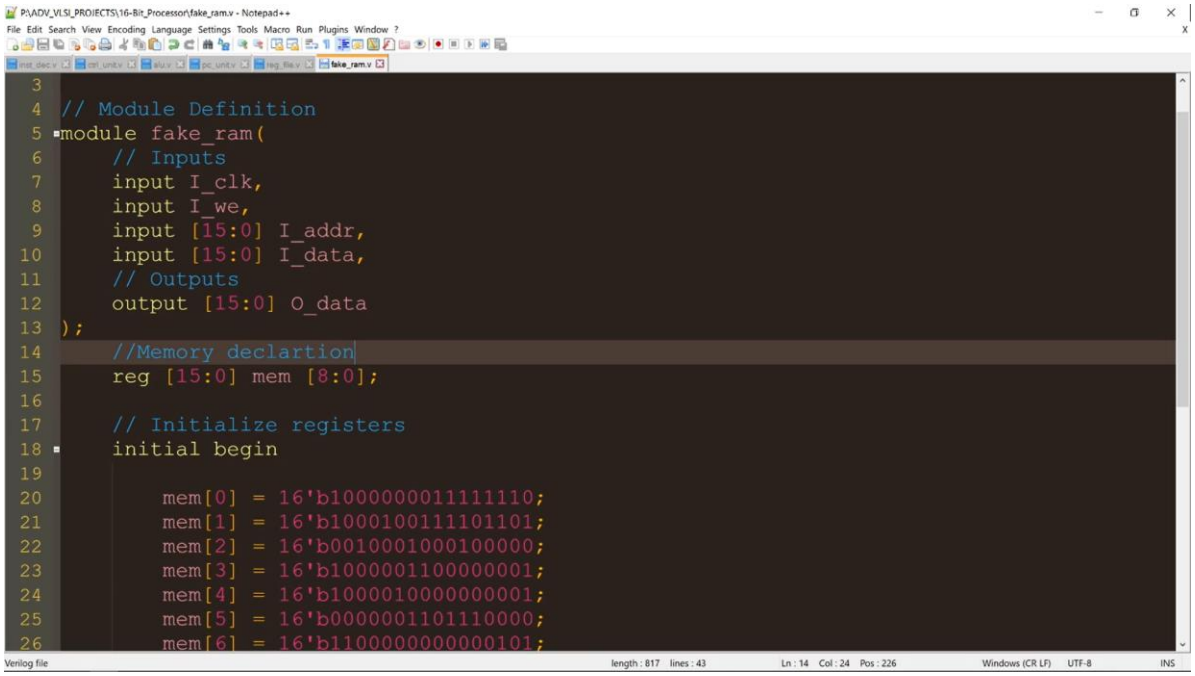


```
'timescale 1ns / 1ps

// Module Definition
module reg_file(
    // Inputs
    input I_clk,
    input I_en,
    input I_we,
    input [3:0] I_selA,
    input [3:0] I_selB,
    input [3:0] I_selD,
    input [3:0] I_dataD,
    // Outputs
    output [15:0] O_dataA,
    output [15:0] O_dataB
);

    // Internal Register declaration
    reg [15:0] regs [7:0];

    // Loop variable
    integer count;

endmodule
```



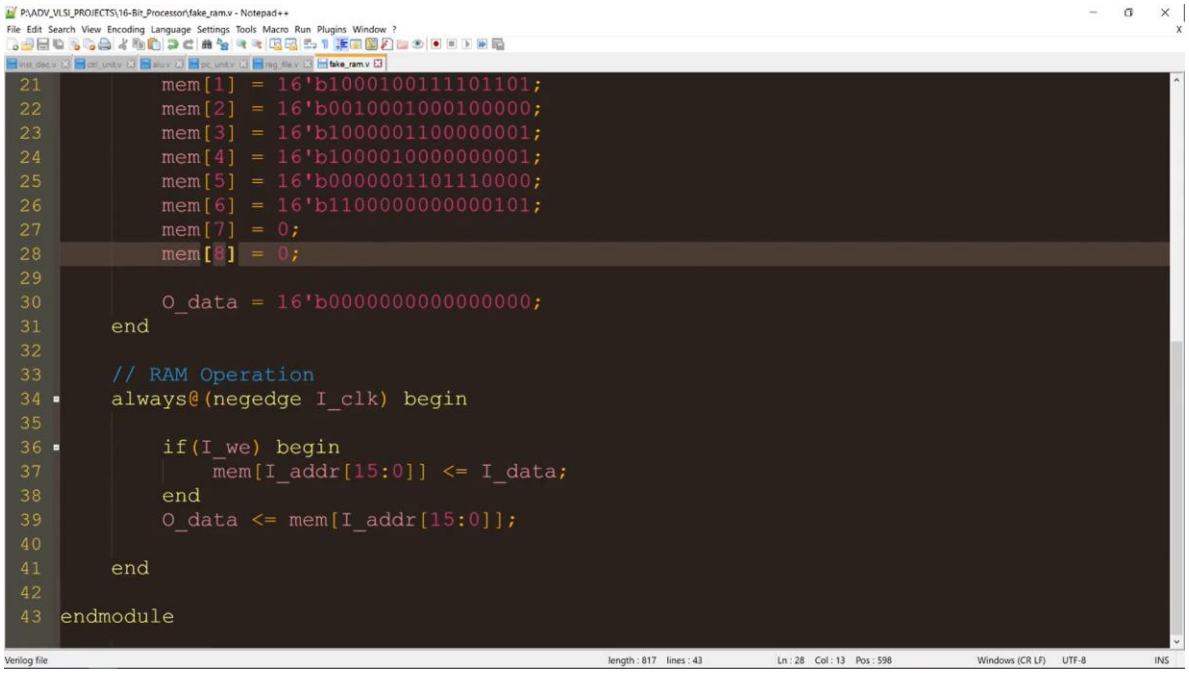
```
    integer count;

    // Initialize registers
    initial begin
        O_dataA <= 0;
        O_dataB <= 0;

        for(count = 0; count < 8; count = count + 1) begin
            regs[count] <= 0;
    end

    // Assigning correct values to OP regs
    always@(negedge I_clk) begin

        if(I_en) begin
            if(I_we)
                regs[I_selD] <= I_dataD;

            O_dataA <= regs[I_selA];
            O_dataB <= regs[I_selB];
    end

endmodule
```

Module fake_ram



```
3
4  // Module Definition
5  module fake_ram(
6      // Inputs
7      input I_clk,
8      input I_we,
9      input [15:0] I_addr,
10     input [15:0] I_data,
11     // Outputs
12     output [15:0] O_data
13 );
14     //Memory declartion
15     reg [15:0] mem [8:0];
16
17     // Initialize registers
18     initial begin
19
20         mem[0] = 16'b1000000011111110;
21         mem[1] = 16'b1000100111101101;
22         mem[2] = 16'b0010001000100000;
23         mem[3] = 16'b1000001100000001;
24         mem[4] = 16'b1000010000000001;
25         mem[5] = 16'b0000001101110000;
26         mem[6] = 16'b1100000000000101;
```

Verilog file length : 817 lines : 43 Ln : 14 Col : 24 Pos : 226 Windows (CR LF) UTF-8 INS

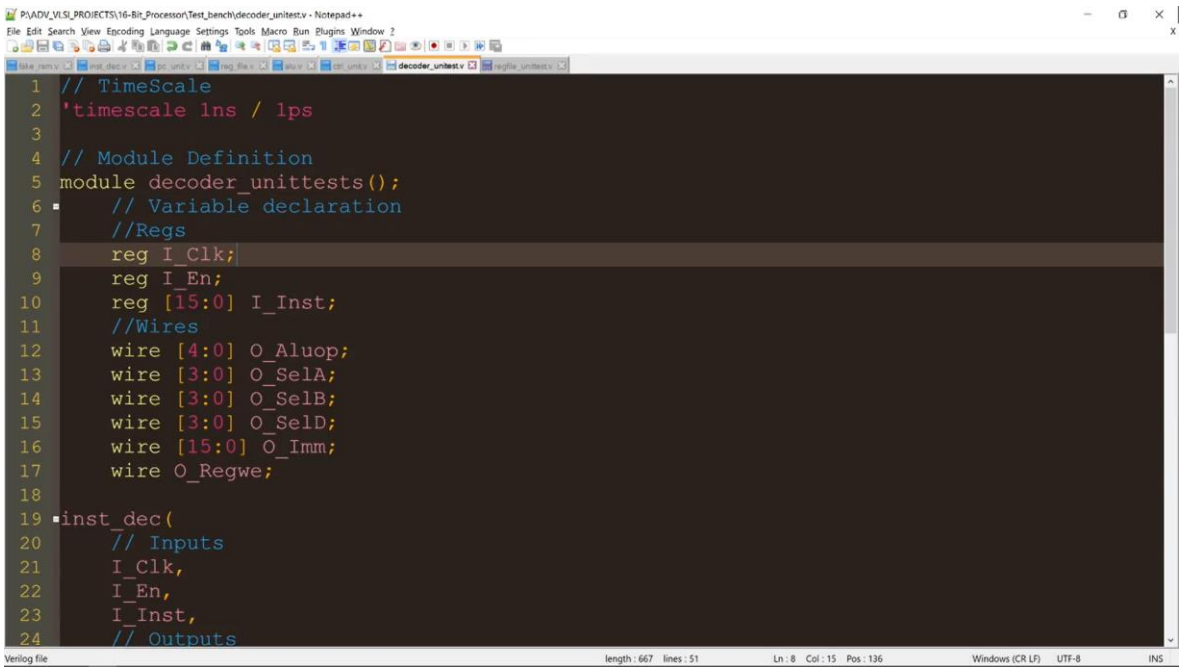


```
21         mem[1] = 16'b1000100111101101;
22         mem[2] = 16'b0010001000100000;
23         mem[3] = 16'b1000001100000001;
24         mem[4] = 16'b1000010000000001;
25         mem[5] = 16'b0000001101110000;
26         mem[6] = 16'b1100000000000101;
27         mem[7] = 0;
28         mem[8] = 0;
29
30         O_data = 16'b0000000000000000;
31     end
32
33     // RAM Operation
34     always@(negedge I_clk) begin
35
36         if(I_we) begin
37             mem[I_addr[15:0]] <= I_data;
38         end
39         O_data <= mem[I_addr[15:0]];
40
41     end
42
43 endmodule
```

Verilog file length : 817 lines : 43 Ln : 28 Col : 13 Pos : 598 Windows (CR LF) UTF-8 INS

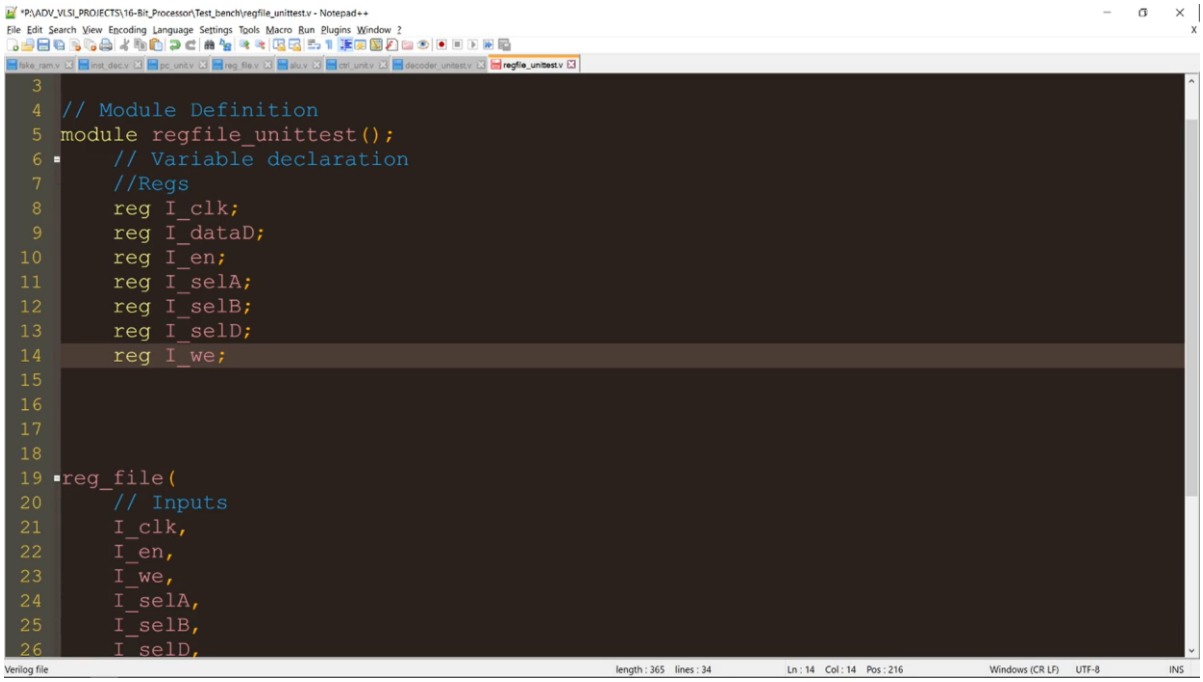
TESTING CODES

Module decoder unit_test

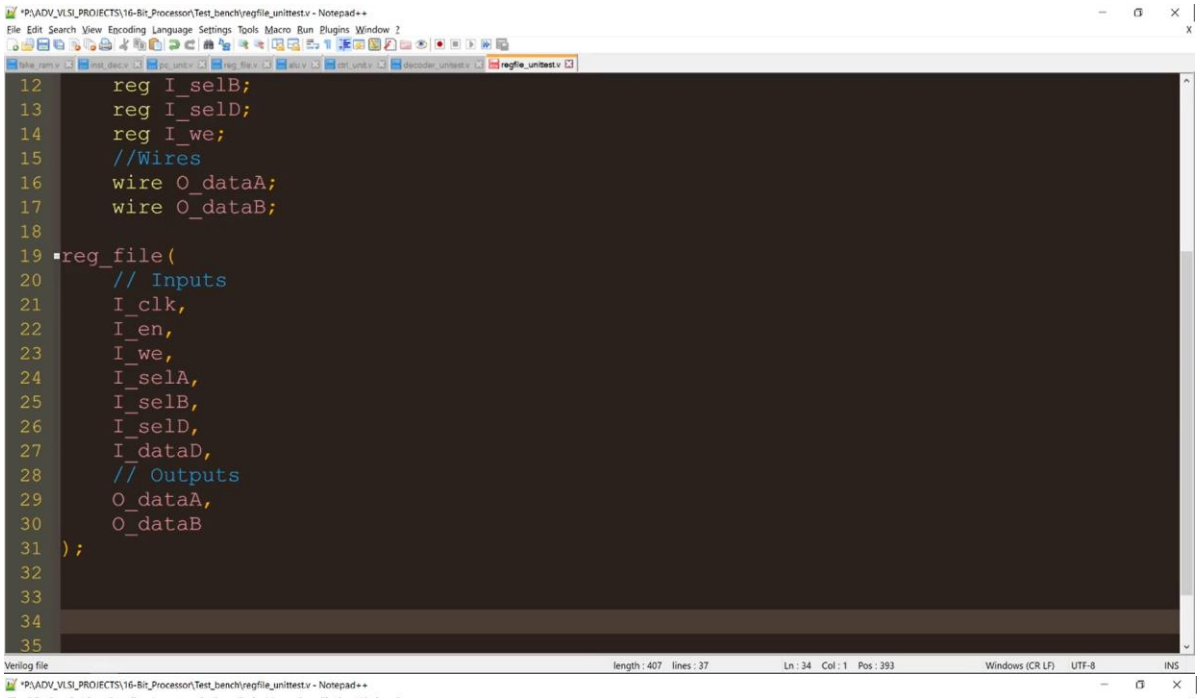


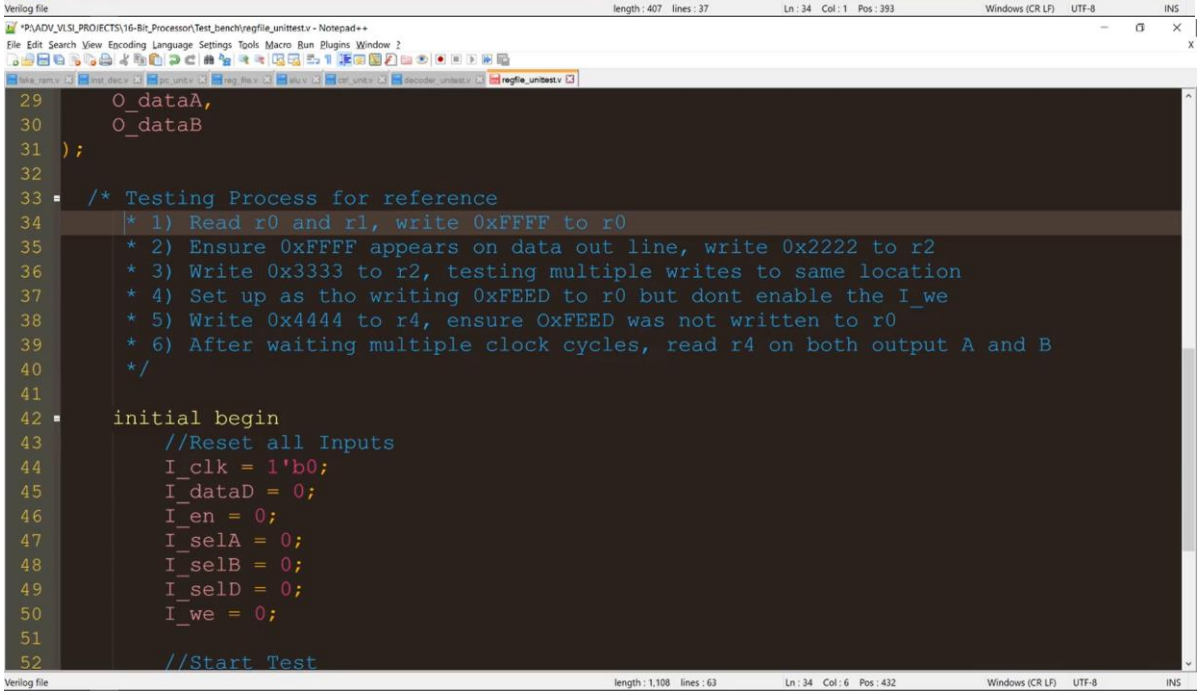
```
1  // TimeScale
2  'timescale 1ns / 1ps
3
4  // Module Definition
5  module decoder_unittests();
6      // Variable declaration
7      //Regs
8      reg I_Clk;
9      reg I_En;
10     reg [15:0] I_Inst;
11     //Wires
12     wire [4:0] O_Aluop;
13     wire [3:0] O_SelA;
14     wire [3:0] O_SelB;
15     wire [3:0] O_SelD;
16     wire [15:0] O_Imm;
17     wire O_Regwe;
18
19 inst_dec(
20     // Inputs
21     I_Clk,
22     I_En,
23     I_Inst,
24     // Outputs
```

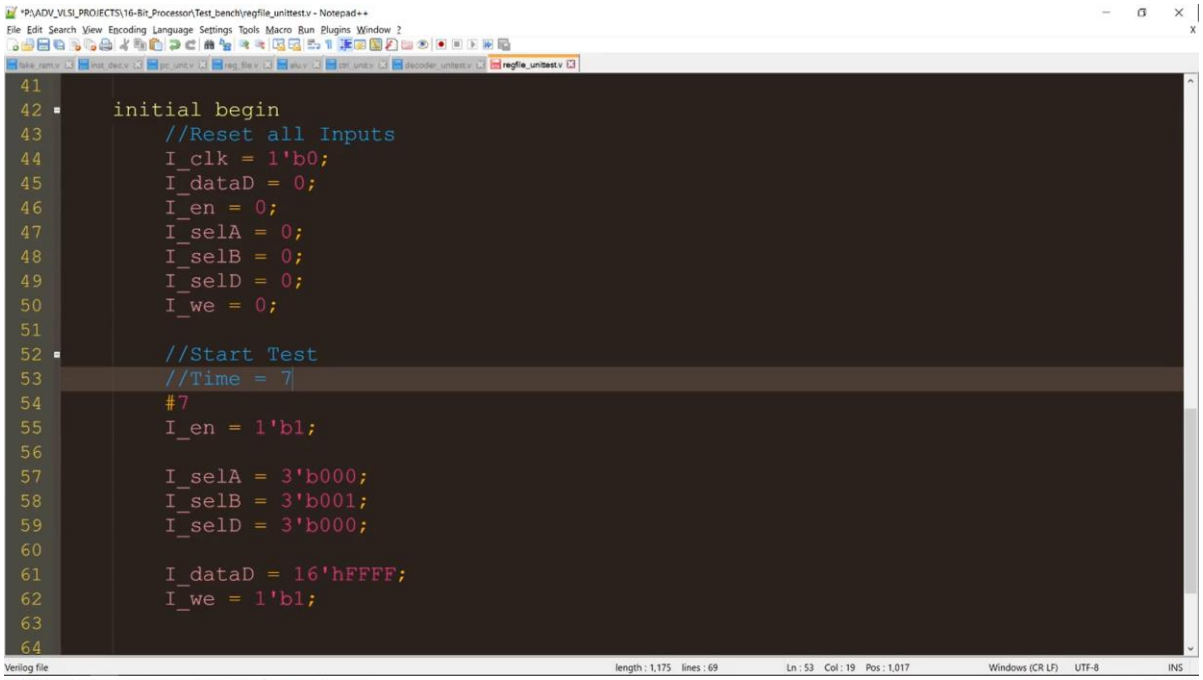
Module regfile unit_test

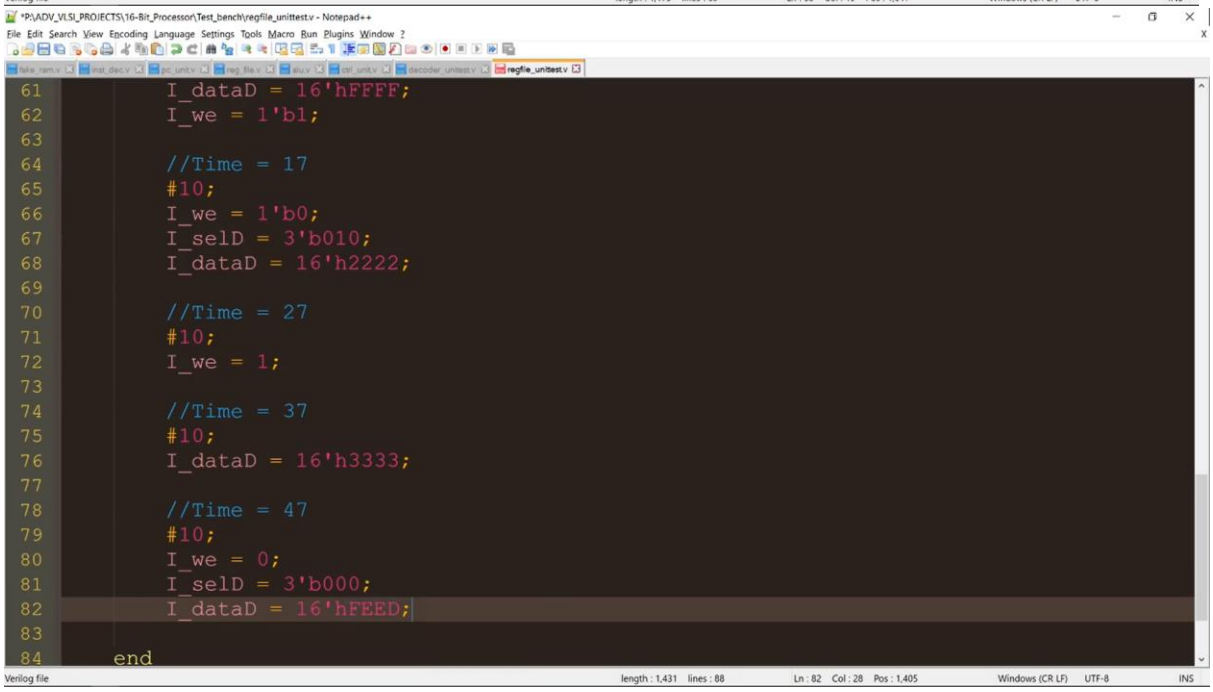


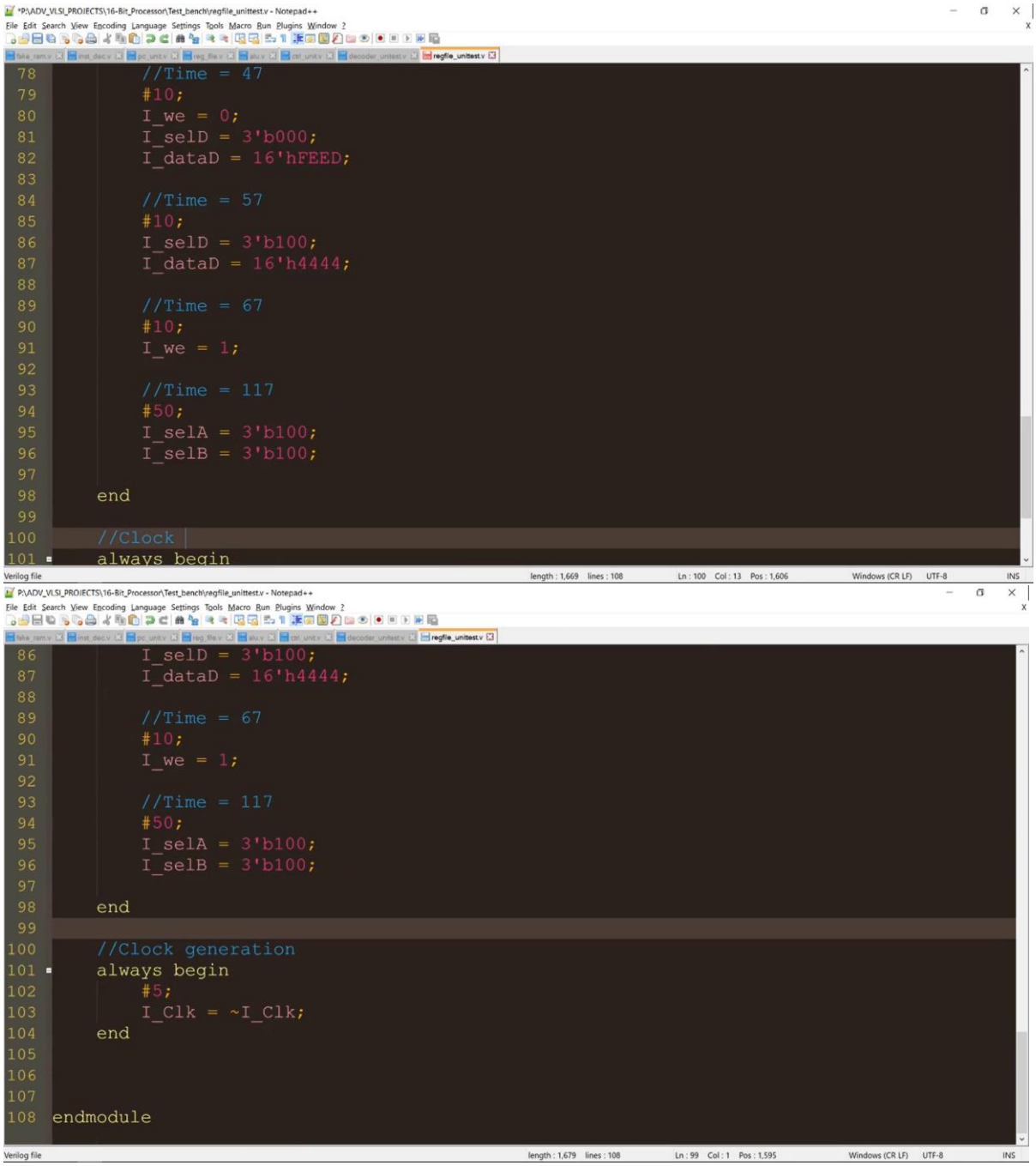
```
3
4  // Module Definition
5  module regfile_unittest();
6      // Variable declaration
7      //Regs
8      reg I_clk;
9      reg I_dataD;
10     reg I_en;
11     reg I_selA;
12     reg I_selB;
13     reg I_selD;
14     reg I_we;
15
16
17
18
19 reg_file(
20     // Inputs
21     I_clk,
22     I_en,
23     I_we,
24     I_selA,
25     I_selB,
26     I_selD,
```


*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\Test_bench\regfile_unittest.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
regfile_unittest.v
reg I_selB;
reg I_selD;
reg I_we;
//Wires
wire O_dataA;
wire O_dataB;
reg_file(
// Inputs
I_clk,
I_en,
I_we,
I_selA,
I_selB,
I_selD,
I_dataD,
// Outputs
O_dataA,
O_dataB
);
Verilog file
length : 407 lines : 37
Ln : 34 Col : 1 Pos : 393
Windows (CR LF)
UTF-8
INS


*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\Test_bench\regfile_unittest.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
regfile_unittest.v
O_dataA,
O_dataB
);
/* Testing Process for reference
* 1) Read r0 and r1, write 0xFFFF to r0
* 2) Ensure 0xFFFF appears on data out line, write 0x2222 to r2
* 3) Write 0x3333 to r2, testing multiple writes to same location
* 4) Set up as tho writing 0xFEED to r0 but dont enable the I_we
* 5) Write 0x4444 to r4, ensure 0xFEED was not written to r0
* 6) After waiting multiple clock cycles, read r4 on both output A and B
*/
initial begin
//Reset all Inputs
I_clk = 1'b0;
I_dataD = 0;
I_en = 0;
I_selA = 0;
I_selB = 0;
I_selD = 0;
I_we = 0;
//Start Test
Verilog file
length : 1,108 lines : 63
Ln : 34 Col : 6 Pos : 432
Windows (CR LF)
UTF-8
INS


*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\Test_bench\regfile_unittest.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
regfile_unittest.v
41
42 initial begin
43 //Reset all Inputs
44 I_clk = 1'b0;
45 I_dataD = 0;
46 I_en = 0;
47 I_selA = 0;
48 I_selB = 0;
49 I_selD = 0;
50 I_we = 0;
51
52 //Start Test
53 //Time = 7
54 #7
55 I_en = 1'b1;
56
57 I_selA = 3'b000;
58 I_selB = 3'b001;
59 I_selD = 3'b000;
60
61 I_dataD = 16'hFFFF;
62 I_we = 1'b1;
63
64
Verilog file
length : 1,175 lines : 69
Ln : 53 Col : 19 Pos : 1,017
Windows (CR LF) UTF-8 INS


*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\Test_bench\regfile_unittest.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
regfile_unittest.v
61 I_dataD = 16'hFFFF;
62 I_we = 1'b1;
63
64 //Time = 17
65 #10;
66 I_we = 1'b0;
67 I_selD = 3'b010;
68 I_dataD = 16'h2222;
69
70 //Time = 27
71 #10;
72 I_we = 1;
73
74 //Time = 37
75 #10;
76 I_dataD = 16'h3333;
77
78 //Time = 47
79 #10;
80 I_we = 0;
81 I_selD = 3'b000;
82 I_dataD = 16'hFEED;
83
84 end
Verilog file
length : 1,431 lines : 88
Ln : 82 Col : 28 Pos : 1,405
Windows (CR LF) UTF-8 INS

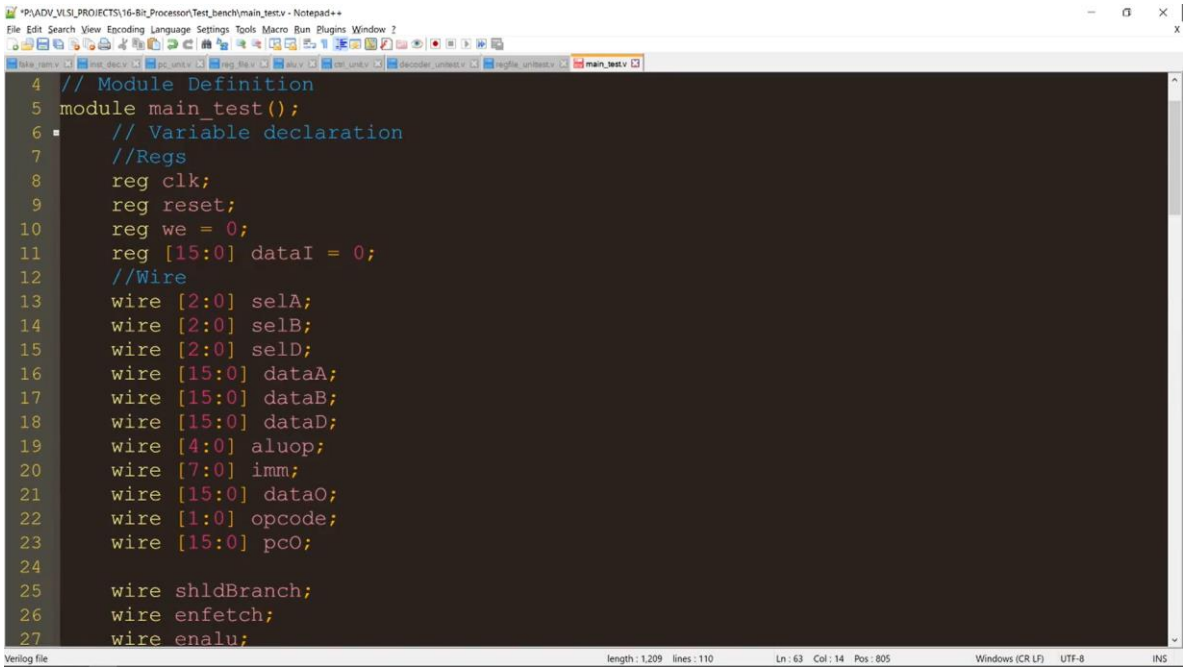


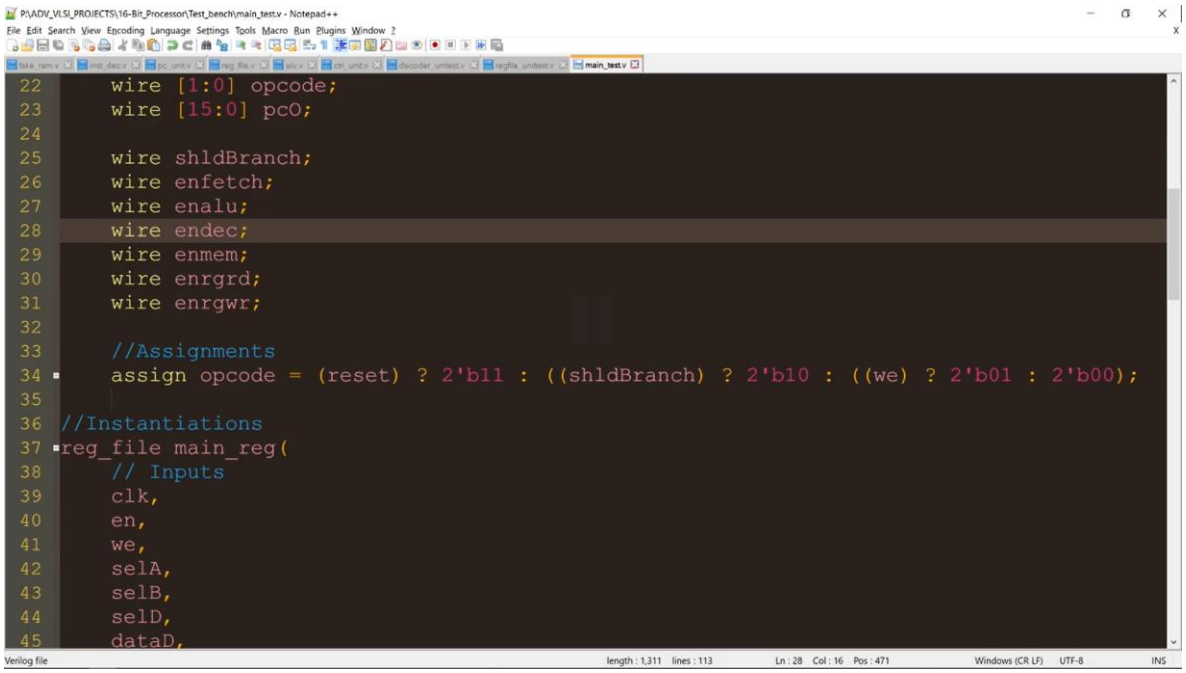
```
        //Time = 47
        #10;
        I_we = 0;
        I_selD = 3'b000;
        I_dataD = 16'hFEED;

        //Time = 57
        #10;
        I_selD = 3'b100;
        I_dataD = 16'h4444;

        //Time = 67
        #10;
        I_we = 1;

        //Time = 117
        #50;
        I_selA = 3'b100;
        I_selB = 3'b100;

    end

    //Clock
    always begin
```



```
        I_selD = 3'b100;
        I_dataD = 16'h4444;

        //Time = 67
        #10;
        I_we = 1;

        //Time = 117
        #50;
        I_selA = 3'b100;
        I_selB = 3'b100;

    end

    //Clock generation
    always begin
        #5;
        I_Clk = ~I_Clk;
    end


endmodule
```


Module regfile main_test


*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\Test_bench\main_test.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
main_test.v
// Module Definition
module main_test();
// Variable declaration
//Regs
reg clk;
reg reset;
reg we = 0;
reg [15:0] dataI = 0;
//Wire
wire [2:0] selA;
wire [2:0] selB;
wire [2:0] selD;
wire [15:0] dataA;
wire [15:0] dataB;
wire [15:0] dataD;
wire [4:0] aluop;
wire [7:0] imm;
wire [15:0] dataO;
wire [1:0] opcode;
wire [15:0] pcO;
wire shldBranch;
wire enfetch;
wire enalu;
Verilog file
length : 1,209 lines : 110
Ln : 63 Col : 14 Pos : 805
Windows (CR LF)
UTF-8
INS


P:\ADV_VLSI_PROJECTS\16-Bit_Processor\Test_bench\main_test.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
main_test.v
wire [1:0] opcode;
wire [15:0] pcO;
wire shldBranch;
wire enfetch;
wire enalu;
wire endec;
wire enmem;
wire enrgrd;
wire enrgwr;
//Assignments
assign opcode = (reset) ? 2'b11 : ((shldBranch) ? 2'b10 : ((we) ? 2'b01 : 2'b00));
//Instantiations
reg_file main_reg(
// Inputs
clk,
en,
we,
selA,
selB,
selD,
dataD,
Verilog file
length : 1,311 lines : 113
Ln : 28 Col : 16 Pos : 471
Windows (CR LF)
UTF-8
INS

```
32
33  //Instantiations
34  reg_file main_reg(
35      // Inputs
36      clk,
37      en,
38      we,
39      selA,
40      selB,
41      selD,
42      dataD,
```

Verilog file length : 1,209 lines : 110 Ln : 25 Col : 20 Sel : 10 | 1 Windows (CR LF) UTF-8 INS

*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\Test_bench\main_test.v - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

fake_ram.v | inst_dec.v | pc_unit.v | reg_file.v | alu.v | ctrl_unit.v | decoder_unitest.v | regfile_unitest.v | main_test.v

```
34  reg_file main_reg(
35      // Inputs
36      clk,
37      en,
38      we,
39      selA,
40      selB,
41      selD,
42      dataD,
43      // Outputs
44      dataA,
45      dataB
46  );
47
48  inst_dec main_inst(
49      // Inputs
50      clk,
51      en,
52      inst,
53      // Outputs
54      aluop,
55      selA,
56      selB,
57      selD,
```

Verilog file length : 1,209 lines : 110 Ln : 25 Col : 20 Sel : 10 | 1 Windows (CR LF) UTF-8 INS

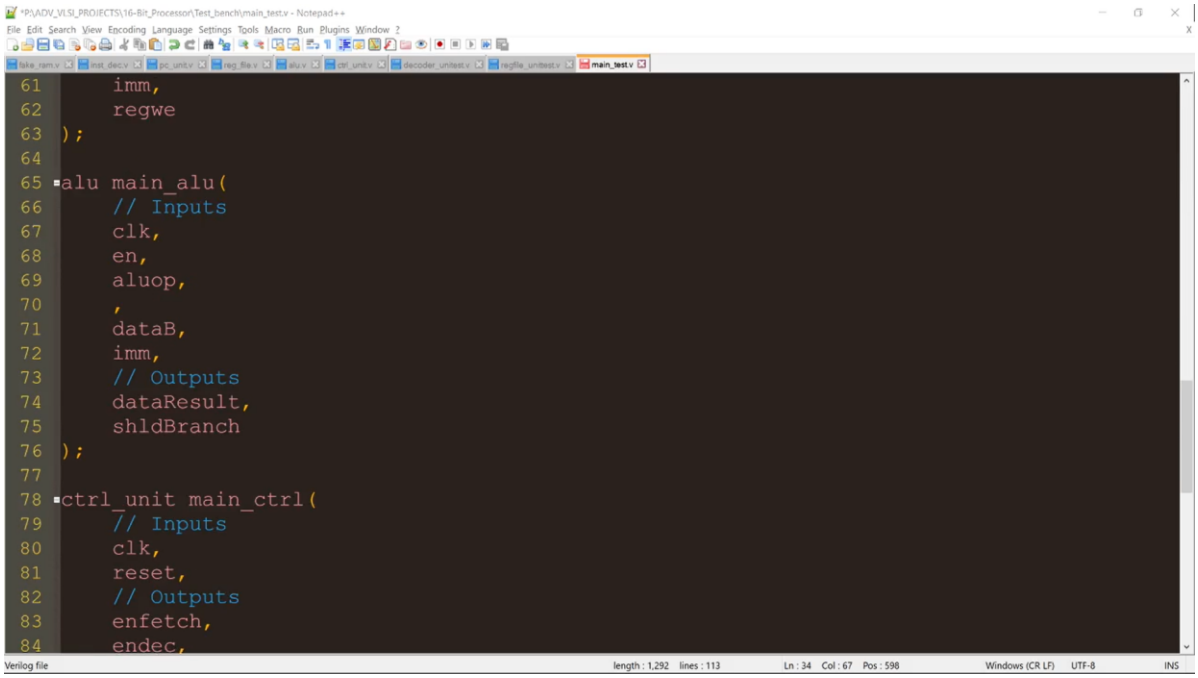
*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\Test_bench\main_test.v - Notepad++

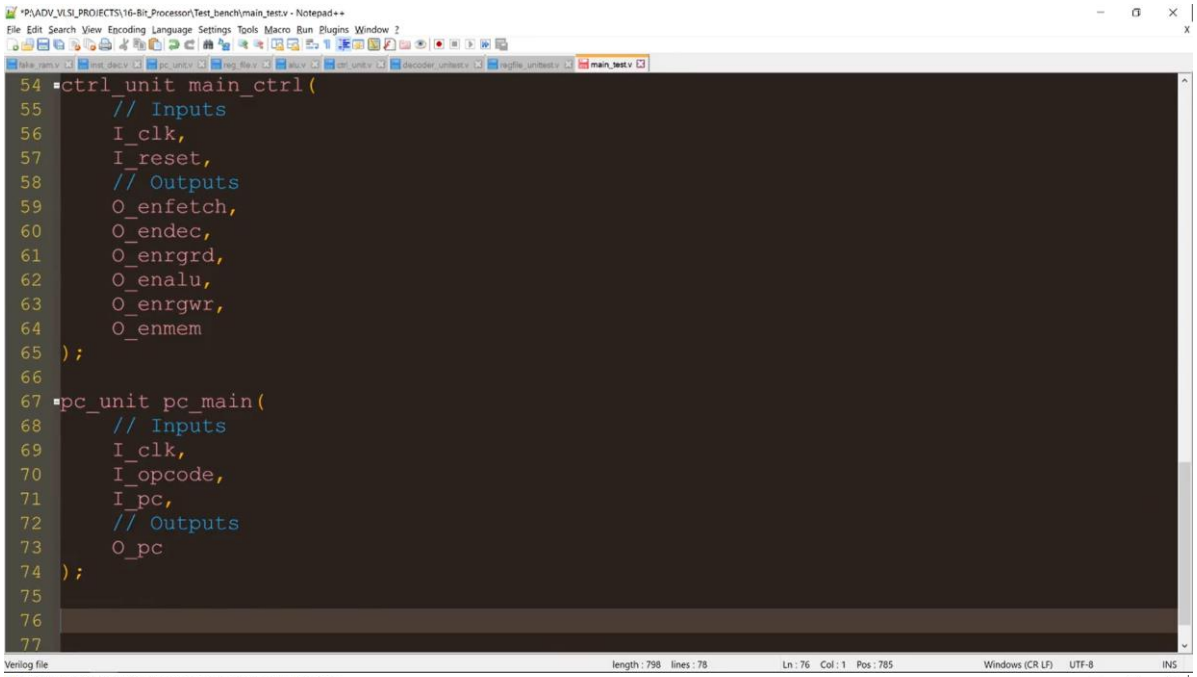
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

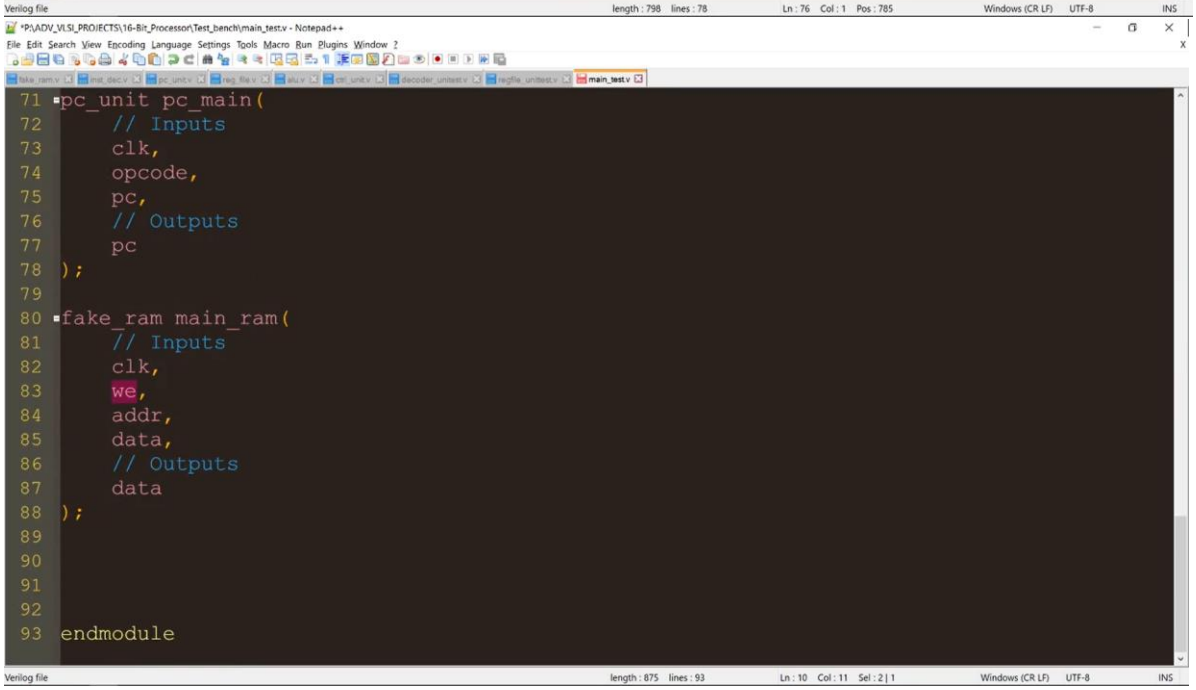
fake_ram.v | inst_dec.v | pc_unit.v | reg_file.v | alu.v | ctrl_unit.v | decoder_unitest.v | regfile_unitest.v | main_test.v

```
43      selB,
44      selD,
45      dataD,
46      // Outputs
47      dataA,
48      dataB
49  );
50
51  inst_dec main_inst(
52      // Inputs
53      clk,
54      en,
55      inst,
56      // Outputs
57      aluop,
58      selA,
59      selB,
60      selD,
61      imm,
62      regwe
63  );
64
65  alu main_alu(
66      // Inputs
```

Verilog file length : 1,261 lines : 113 Ln : 34 Col : 34 Pos : 565 Windows (CR LF) UTF-8 INS


*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\Test_bench\main_test.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
fake_ram.v
inst_dec.v
pc_unit.v
reg_file.v
alu.v
ctrl_unit.v
decoder_unitest.v
regfile_unitest.v
main_test.v
61 imm,
62 regwe
63);
64
65 alu main_alu(
66 // Inputs
67 clk,
68 en,
69 aluop,
70 ,
71 dataB,
72 imm,
73 // Outputs
74 dataResult,
75 shldBranch
76);
77
78 ctrl_unit main_ctrl(
79 // Inputs
80 clk,
81 reset,
82 // Outputs
83 enfetch,
84 endec,
Verilog file
length : 1,292 lines : 113
Ln : 34 Col : 67 Pos : 598
Windows (CR LF) UTF-8
INS


*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\Test_bench\main_test.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
main_test.v
ctrl_unit main_ctrl(
// Inputs
I_clk,
I_reset,
// Outputs
O_enfetch,
O_endec,
O_enrgrd,
O_enalu,
O_enrgwr,
O_enmem
);
pc_unit pc_main(
// Inputs
I_clk,
I_opcode,
I_pc,
// Outputs
O_pc
);
Verilog file
length : 798 lines : 78
Ln : 76 Col : 1 Pos : 785
Windows (CR LF) UTF-8 INS

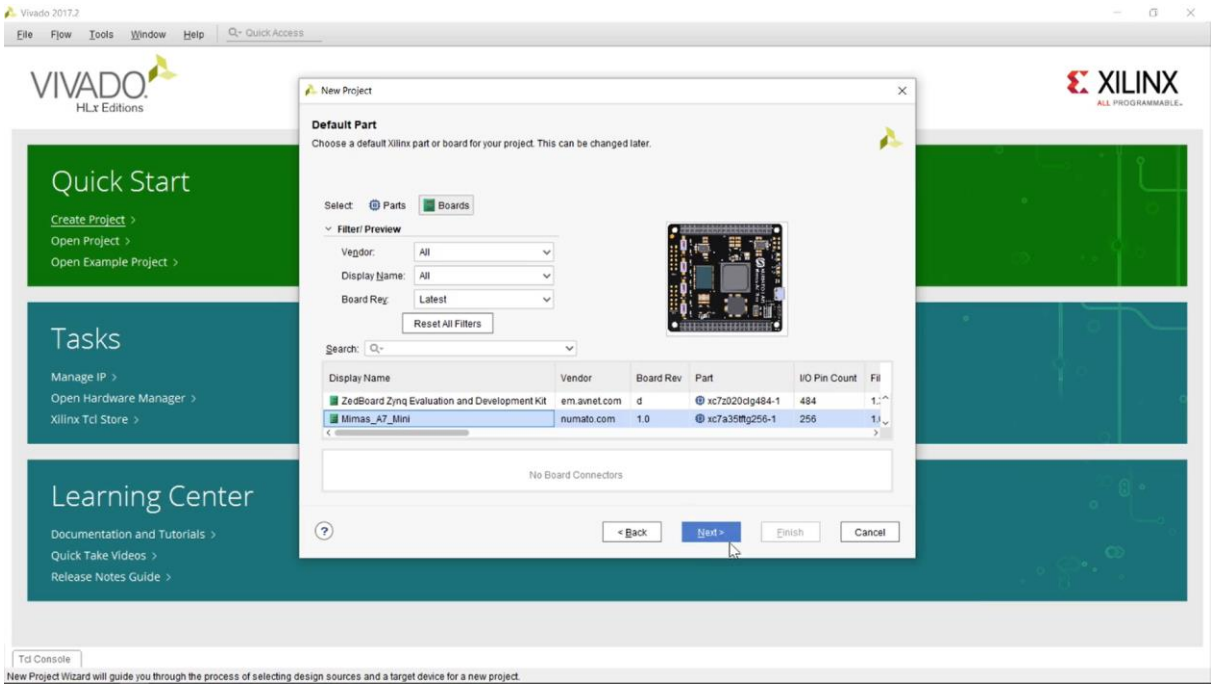

*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\Test_bench\main_test.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
main_test.v
pc_unit pc_main(
// Inputs
clk,
opcode,
pc,
// Outputs
pc
);
fake_ram main_ram(
// Inputs
clk,
we,
addr,
data,
// Outputs
data
);
endmodule
Verilog file
length : 875 lines : 93
Ln : 10 Col : 11 Sel : 2 | 1
Windows (CR LF) UTF-8 INS

```
    addr,
    dataI,
    // Outputs
    dataO
);

    initial begin

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

    end

    //Clock generation
    always begin
        #5;
        I_Clk = ~I_Clk;
    end


endmodule
```

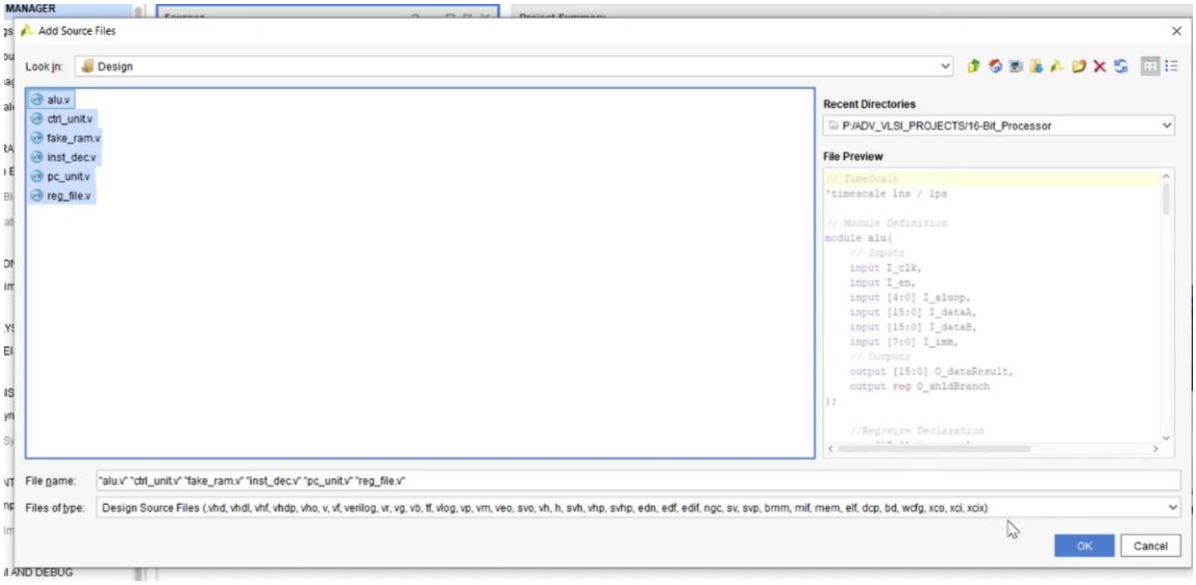
Time to simulate!

Create a project with a name.

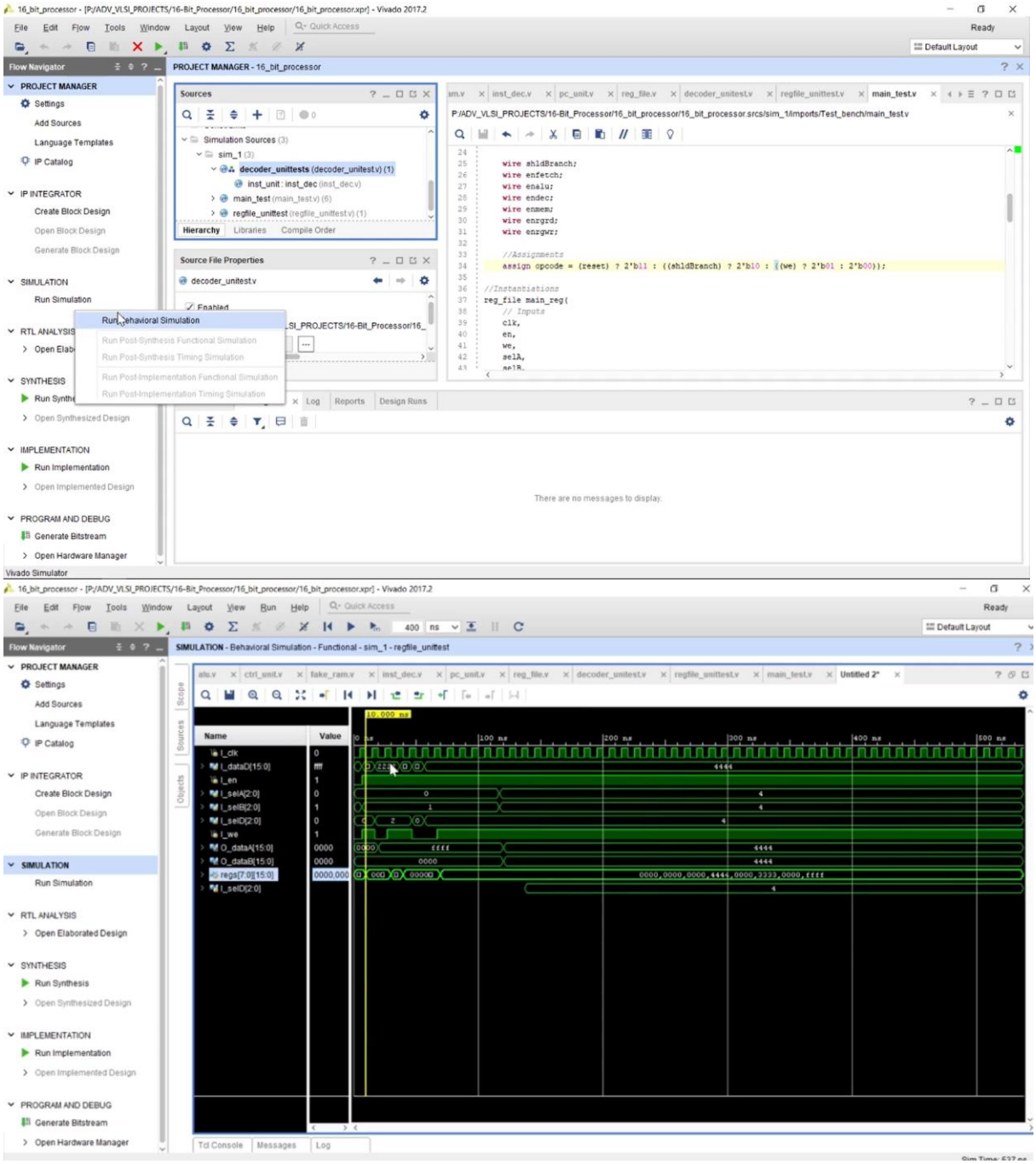
Chose the board, Mimas_A7_Mini



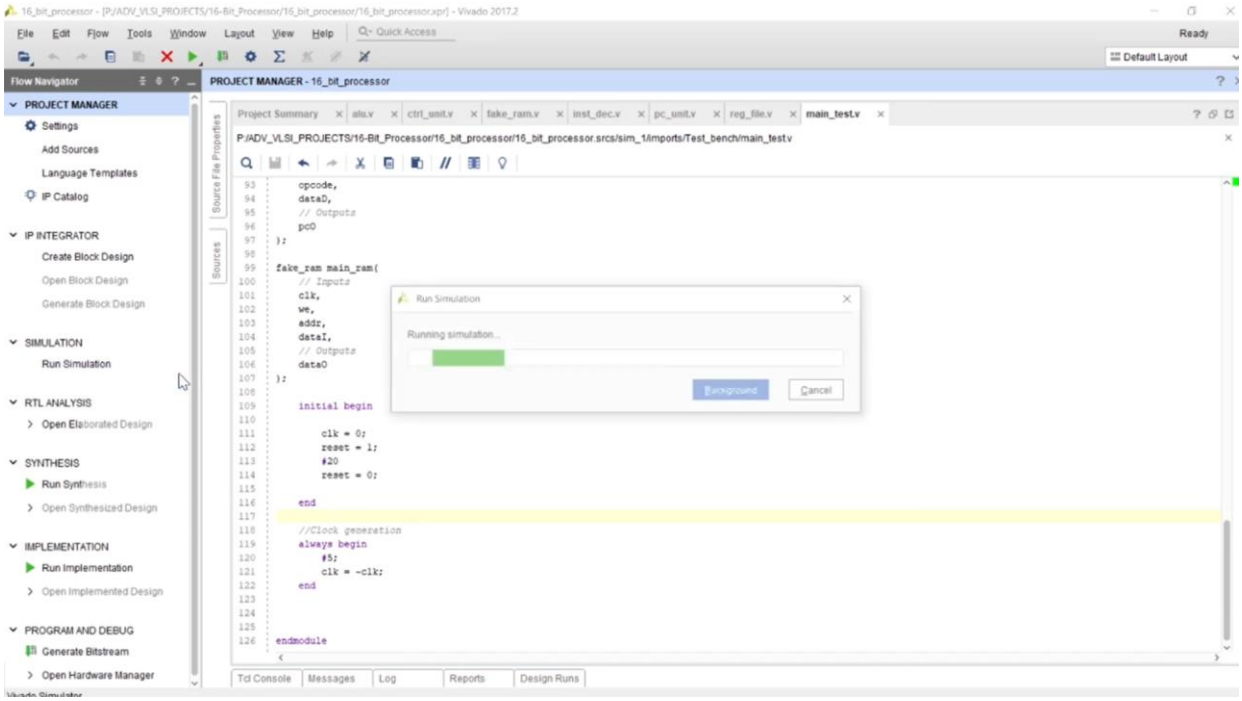
Import all the verilog files

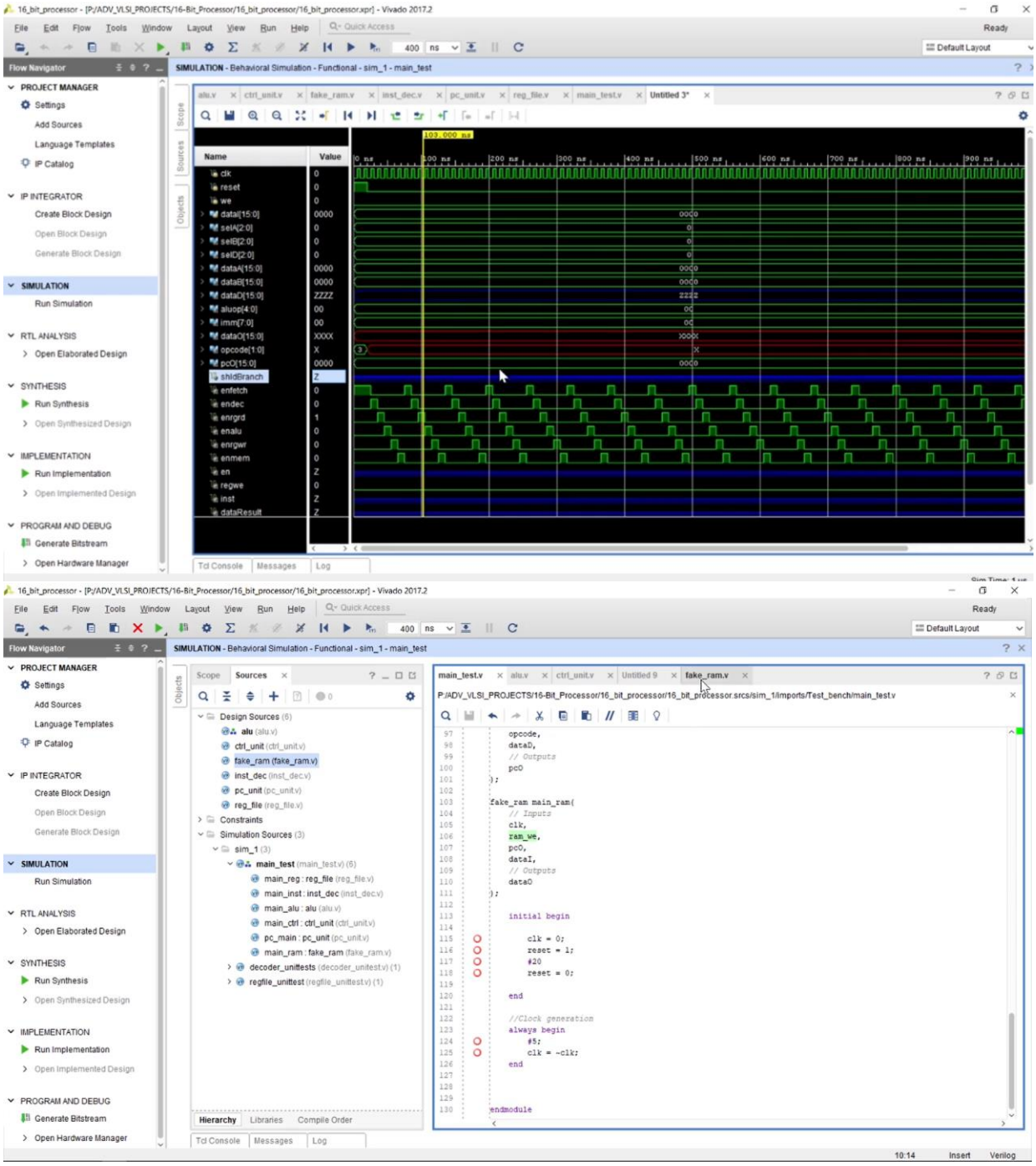


After successfully compiling the errors, run Behavorial simulation model of our instruction decoder.

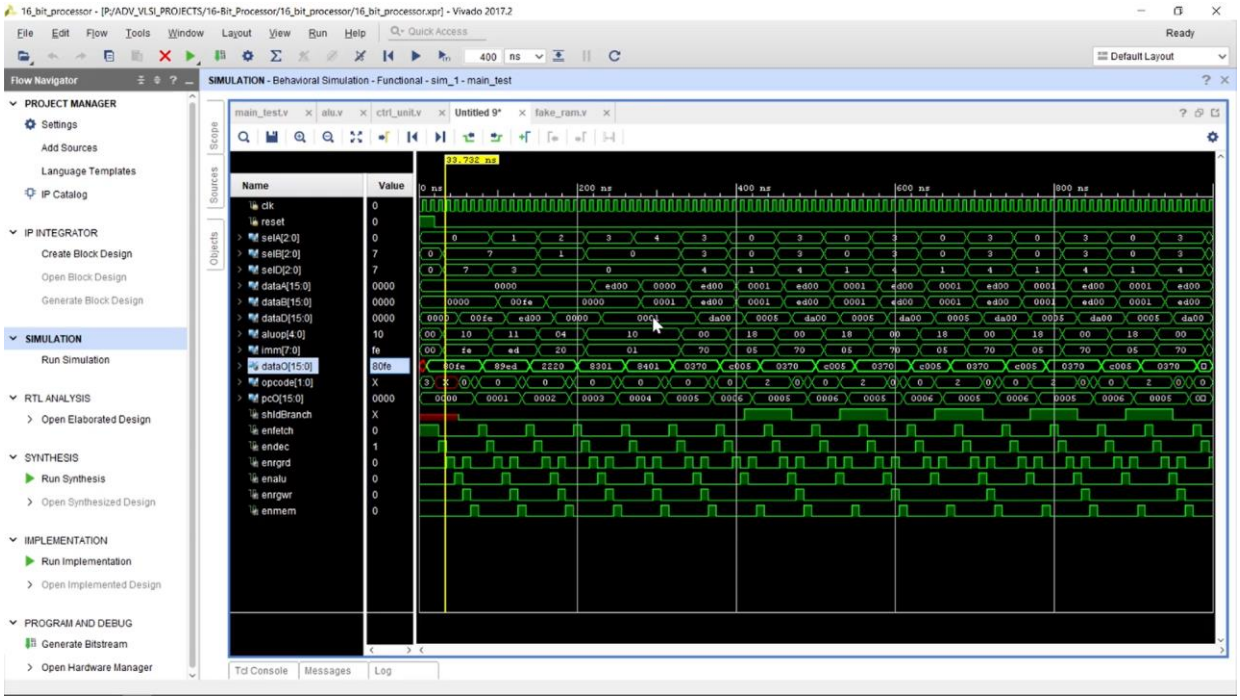


Run the main test bench simulation


16_bit_processor - [P:/ADV_VLSI_PROJECTS/16-Bit_Processor/16_bit_processor/16_bit_processor.xpr] - Vivado 2017.2
File Edit Flow Tools Window Layout View Help
Quick Access
Ready
Default Layout
Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
PROGRAM AND DEBUG
Generate Bitstream
Open Hardware Manager
PROJECT MANAGER - 16_bit_processor
Project Summary
alu.v
ctrl_unit.v
fake_ram.v
inst_dec.v
pc_unit.v
reg_file.v
main_test.v
P:/ADV_VLSI_PROJECTS/16-Bit_Processor/16_bit_processor.srcs/sim_1/imports/Test_bench/main_test.v
Source File Properties
Sources
93 opcode,
94 dataD,
95 // Outputs
96 pc0
97);
98
99 fake_ram main_ram(
100 // Inputs
101 clk,
102 we,
103 addr,
104 dataI,
105 // Outputs
106 dataO
107);
108
109 initial begin
110
111 clk = 0;
112 reset = 1;
113 #20
114 reset = 0;
115
116 end
117
118 //Clock generation
119 always begin
120 #5;
121 clk = ~clk;
122 end
123
124
125
126 endmodule
Run Simulation
Running simulation...
Background
Cancel
Tcl Console
Messages
Log
Reports
Design Runs


16_bit_processor - [P:/ADV_VLSI_PROJECTS/16-Bit_Processor/16_bit_processor/16_bit_processor.xpr] - Vivado 2017.2
File Edit Flow Tools Window Layout View Run Help
Quick Access
Ready
Default Layout
Flow Navigator
SIMULATION - Behavioral Simulation - Functional - sim_1 - main_test
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
PROGRAM AND DEBUG
Generate Bitstream
Open Hardware Manager
Name
Value
clk
reset
we
data[15:0]
selA[2:0]
selB[2:0]
selD[2:0]
dataA[15:0]
dataB[15:0]
dataD[15:0]
aluop[4:0]
imm[7:0]
dataO[15:0]
opcode[1:0]
pcO[15:0]
shldBranch
enfetch
endec
enrgrd
enalu
enrgwr
enmem
en
regwe
inst
dataResult
103.000 ns
Tcl Console
Messages
Log

16_bit_processor - [P:/ADV_VLSI_PROJECTS/16-Bit_Processor/16_bit_processor/16_bit_processor.xpr] - Vivado 2017.2
SIMULATION - Behavioral Simulation - Functional - sim_1 - main_test
Scope
Sources
Design Sources (6)
alu (alu.v)
ctrl_unit (ctrl_unit.v)
fake_ram (fake_ram.v)
inst_dec (inst_dec.v)
pc_unit (pc_unit.v)
reg_file (reg_file.v)
Constraints
Simulation Sources (3)
sim_1 (3)
main_test (main_test.v) (6)
main_reg : reg_file (reg_file.v)
main_inst : inst_dec (inst_dec.v)
main_alu : alu (alu.v)
main_ctrl : ctrl_unit (ctrl_unit.v)
pc_main : pc_unit (pc_unit.v)
main_ram : fake_ram (fake_ram.v)
decoder_unittests (decoder_unittest.v) (1)
regfile_unittest (regfile_unittest.v) (1)
Hierarchy
Libraries
Compile Order
main_test.v
alu.v
ctrl_unit.v
Untitled 9
fake_ram.v
P:/ADV_VLSI_PROJECTS/16-Bit_Processor/16_bit_processor/16_bit_processor.srcs/sim_1/imports/Test_bench/main_test.v
opcode,
dataD,
// Outputs
pcO
);
fake_ram main_ram(
// Inputs
clk,
ram_we,
pcO,
dataI,
// Outputs
dataO
);
initial begin
clk = 0;
reset = 1;
#20
reset = 0;
end
//Clock generation
always begin
#5;
clk = ~clk;
end
endmodule
Tcl Console
Messages
Log
10:14
Insert
Verilog


16_bit_processor - [P:/ADV_VLSI_PROJECTS/16-Bit_Processor/16_bit_processor/16_bit_processor.xpr] - Vivado 2017.2
File Edit Flow Tools Window Layout View Run Help
Quick Access
Ready
Default Layout
Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
PROGRAM AND DEBUG
Generate Bitstream
Open Hardware Manager
SIMULATION - Behavioral Simulation - Functional - sim_1 - main_test
main_test.v
alu.v
ctrl_unit.v
Untitled 9*
fake_ram.v
Scope
Sources
Objects
Name
Value
clk
reset
selA[2:0]
selB[2:0]
selD[2:0]
dataA[15:0]
dataB[15:0]
dataD[15:0]
aluop[4:0]
imm[7:0]
dataO[15:0]
opcode[1:0]
pcO[15:0]
shldBranch
enfetch
endec
enrgrd
enalu
enrgwr
enmem
33.792 ns
0 ns
200 ns
400 ns
600 ns
800 ns
Tcl Console
Messages
Log