به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده مهندسی برق و کامپیوتر

سیستم‌های دیجیتال 1

--- ECE 894 ---

نیمسال دوم (00-99)

استاد: پروفسور نوابی

COMPUTER ASSIGNMENT 6

Hierarchical RTL Design

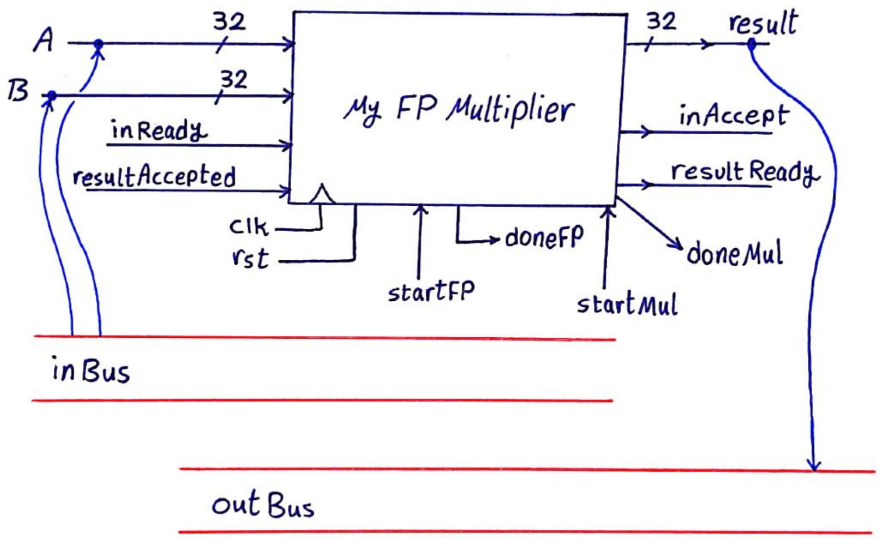
محمدمهدی عبدالحسینی

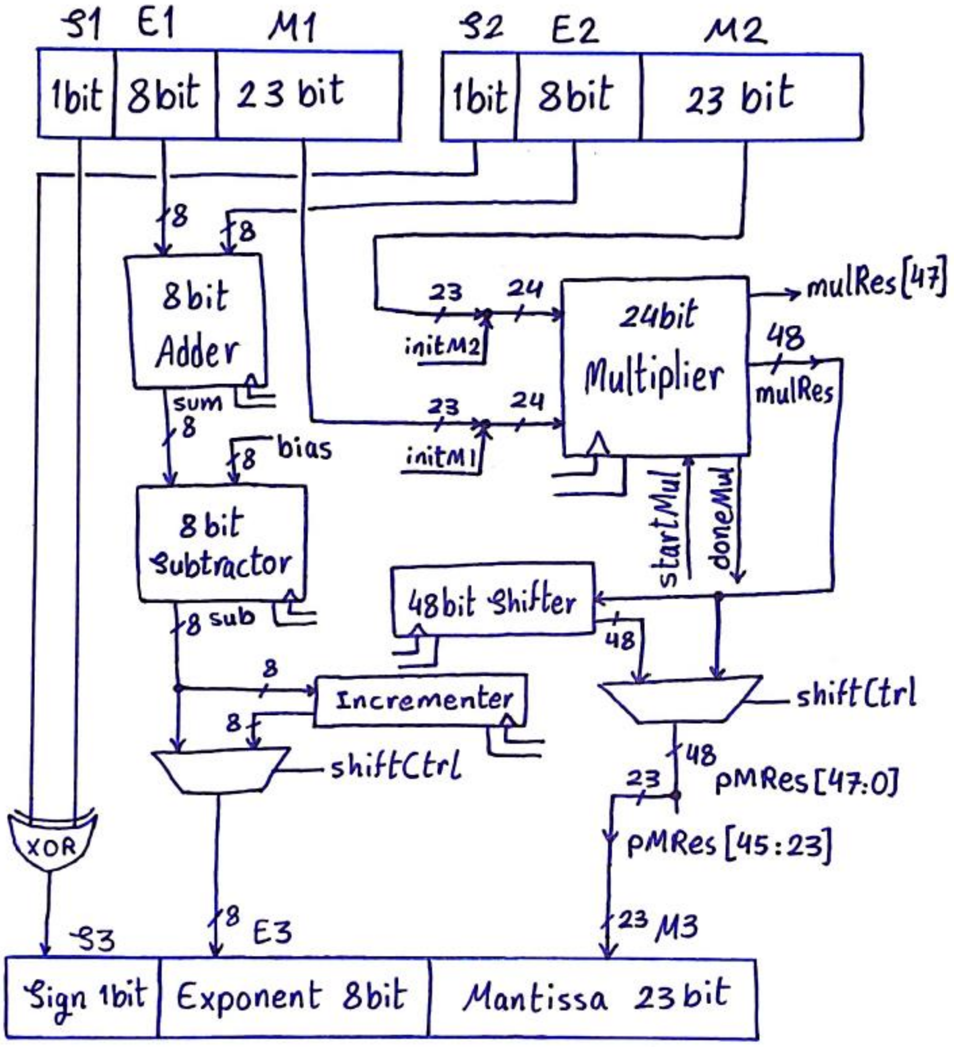
810 198 434

DIGITAL SYSTEMS

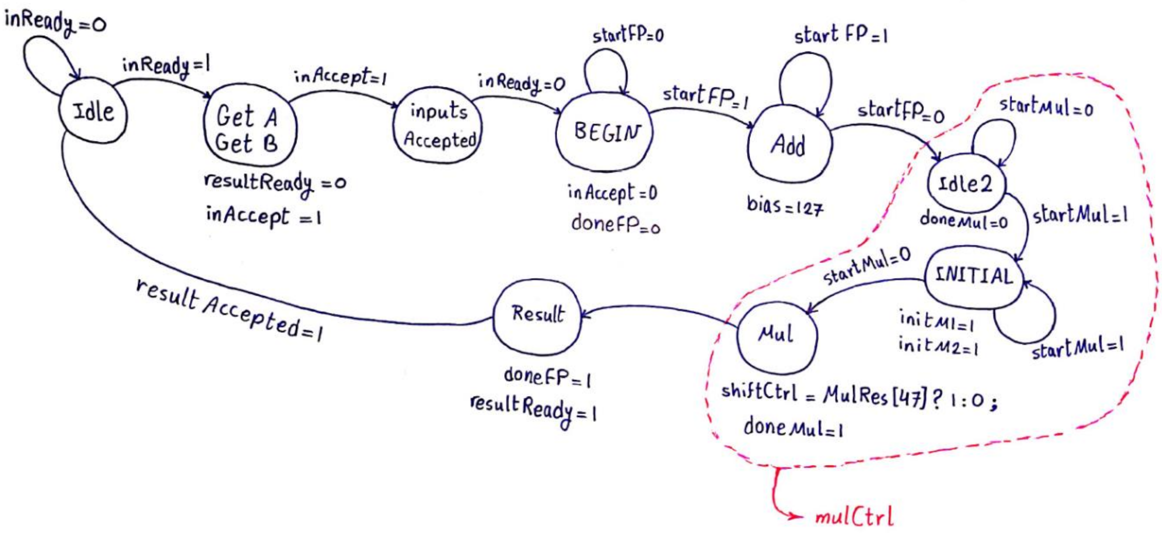
فهرست مطالب

بخش اول : Datapath

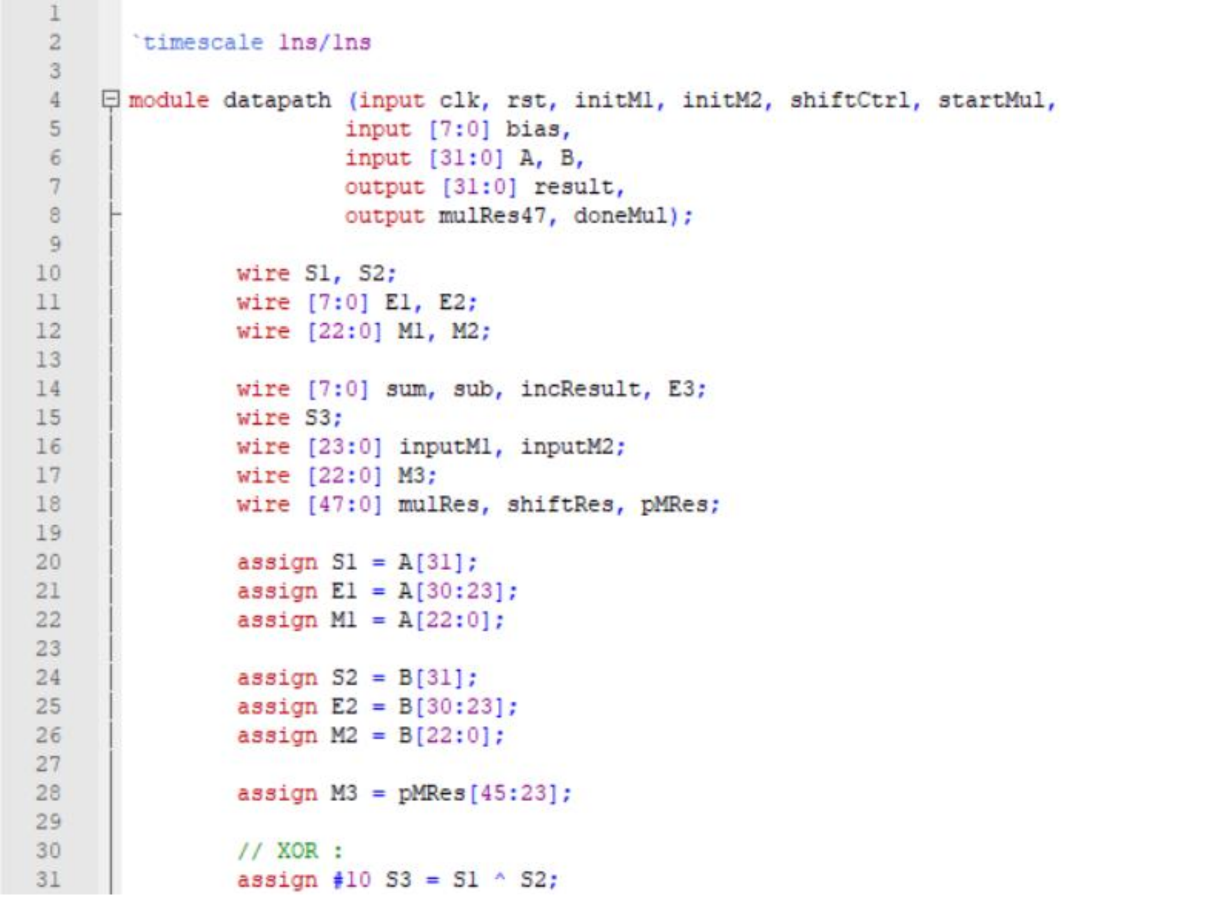




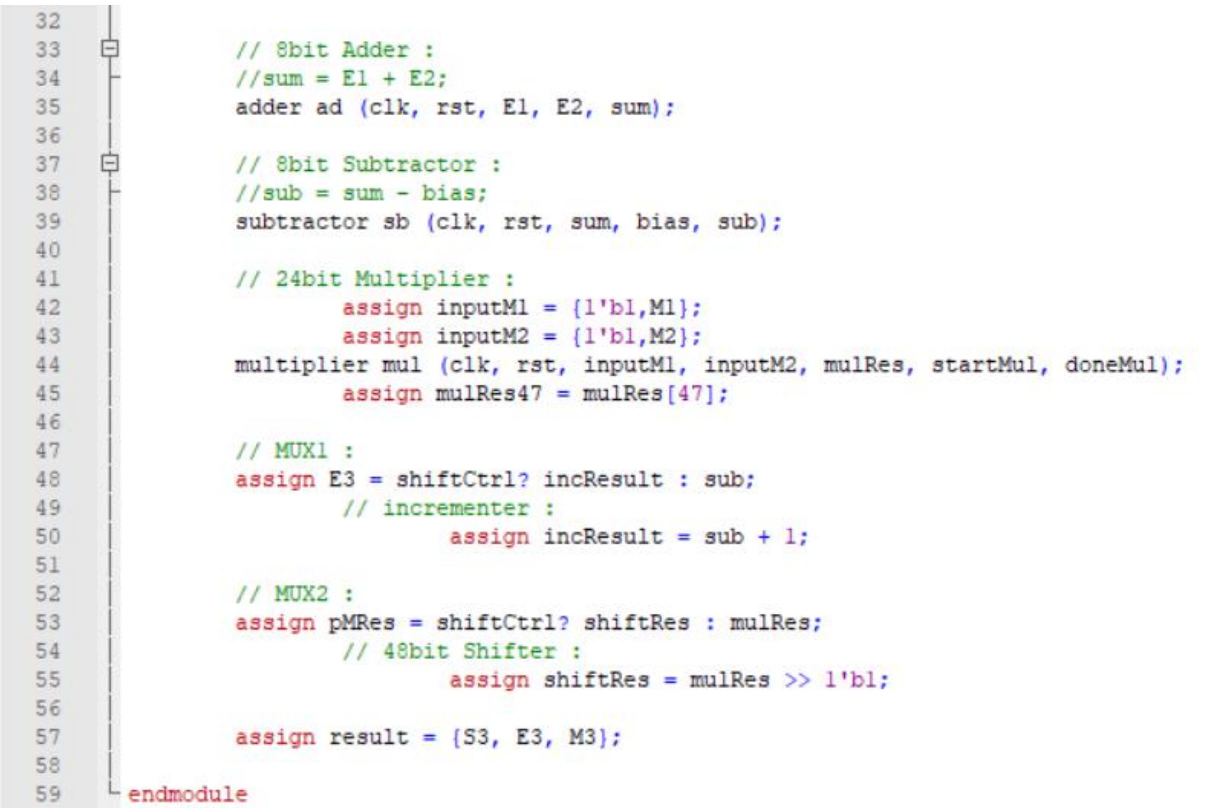
بخش دوم : Controller



بخش سوم : Datapath Verilog

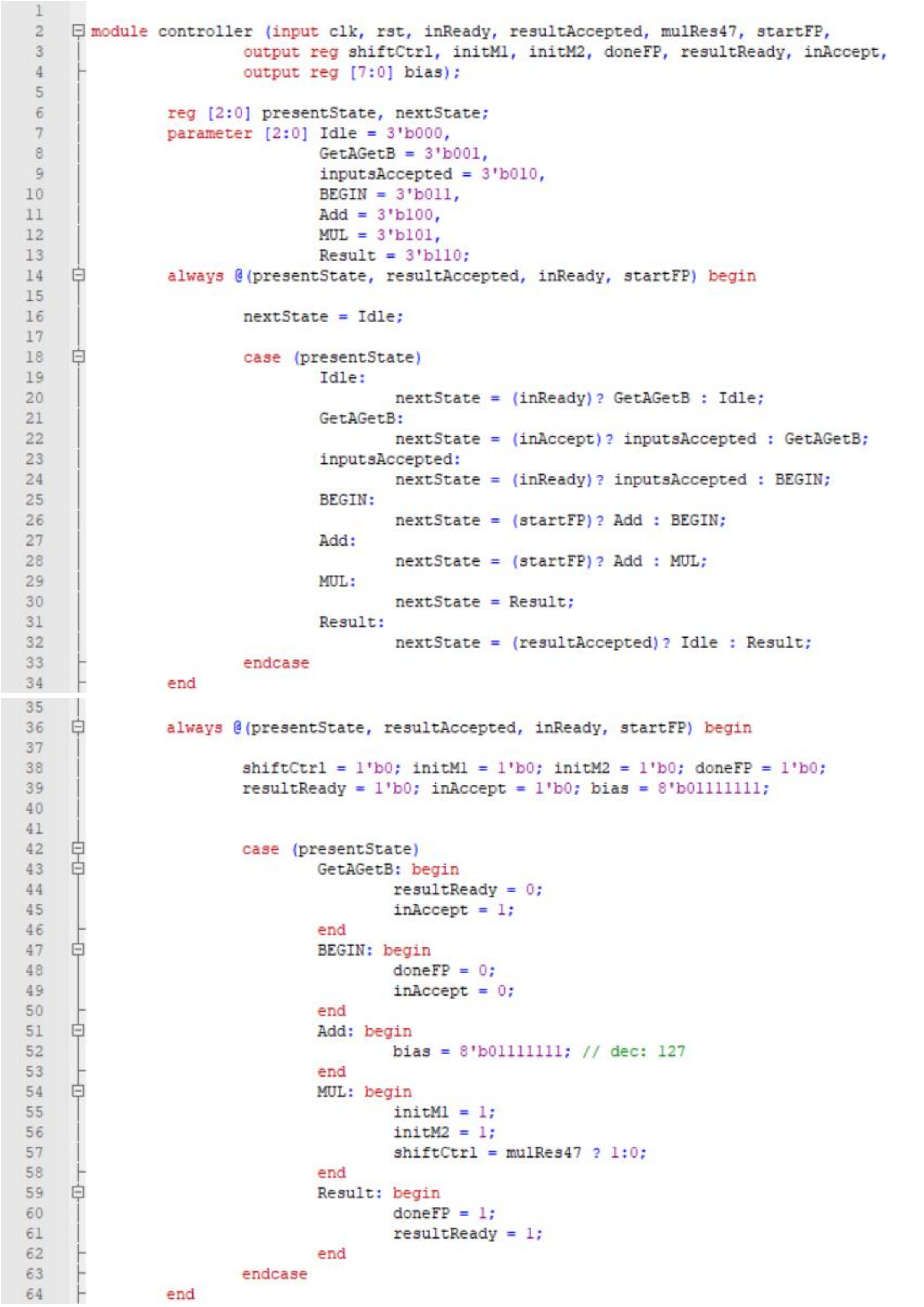


```verilog

`timescale 1ns/1ns

module datapath (input clk, rst, initM1, initM2, shiftCtrl, startMul,
                 input [7:0] bias,
                 input [31:0] A, B,
                 output [31:0] result,
                 output mulRes47, doneMul);

        wire S1, S2;
        wire [7:0] E1, E2;
        wire [22:0] M1, M2;

        wire [7:0] sum, sub, incResult, E3;
        wire S3;
        wire [23:0] inputM1, inputM2;
        wire [22:0] M3;
        wire [47:0] mulRes, shiftRes, pMRes;

        assign S1 = A[31];
        assign E1 = A[30:23];
        assign M1 = A[22:0];

        assign S2 = B[31];
        assign E2 = B[30:23];
        assign M2 = B[22:0];

        assign M3 = pMRes[45:23];

        // XOR :
        assign #10 S3 = S1 ^ S2;
```

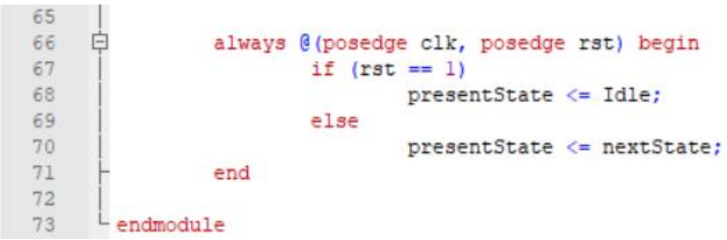


```verilog

        // 8bit Adder :
        //sum = E1 + E2;
        adder ad (clk, rst, E1, E2, sum);

        // 8bit Subtractor :
        //sub = sum - bias;
        subtractor sb (clk, rst, sum, bias, sub);

        // 24bit Multiplier :
                assign inputM1 = {1'b1,M1};
                assign inputM2 = {1'b1,M2};
        multiplier mul (clk, rst, inputM1, inputM2, mulRes, startMul, doneMul);
                assign mulRes47 = mulRes[47];

        // MUX1 :
        assign E3 = shiftCtrl? incResult : sub;
                // incrementer :
                        assign incResult = sub + 1;

        // MUX2 :
        assign pMRes = shiftCtrl? shiftRes : mulRes;
                // 48bit Shifter :
                        assign shiftRes = mulRes >> 1'b1;

        assign result = {S3, E3, M3};

endmodule
```

بخش چهارم : Controller Verilog

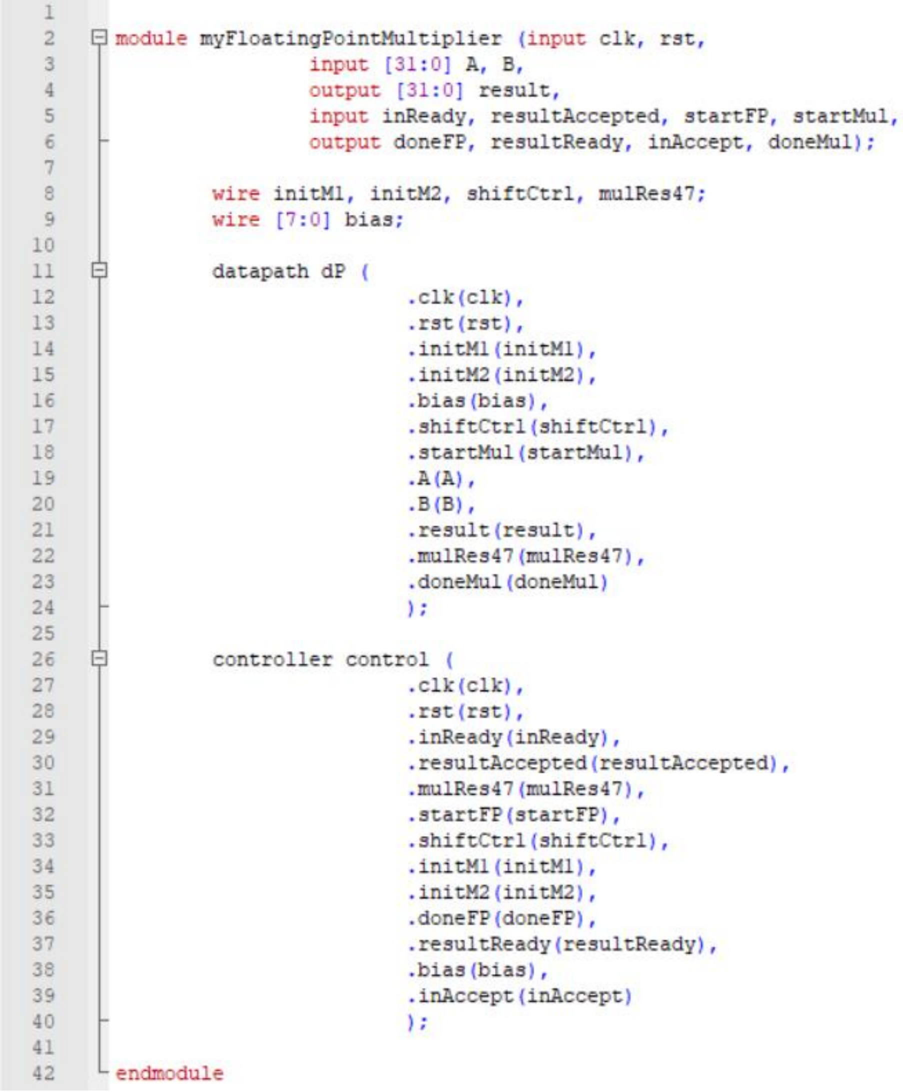


```verilog

module controller (input clk, rst, inReady, resultAccepted, mulRes47, startFP,
                output reg shiftCtrl, initM1, initM2, doneFP, resultReady, inAccept,
                output reg [7:0] bias);

        reg [2:0] presentState, nextState;
        parameter [2:0] Idle = 3'b000,
                        GetAGetB = 3'b001,
                        inputsAccepted = 3'b010,
                        BEGIN = 3'b011,
                        Add = 3'b100,
                        MUL = 3'b101,
                        Result = 3'b110;
        always @(presentState, resultAccepted, inReady, startFP) begin

                nextState = Idle;

                case (presentState)
                        Idle:
                                nextState = (inReady)? GetAGetB : Idle;
                        GetAGetB:
                                nextState = (inAccept)? inputsAccepted : GetAGetB;
                        inputsAccepted:
                                nextState = (inReady)? inputsAccepted : BEGIN;
                        BEGIN:
                                nextState = (startFP)? Add : BEGIN;
                        Add:
                                nextState = (startFP)? Add : MUL;
                        MUL:
                                nextState = Result;
                        Result:
                                nextState = (resultAccepted)? Idle : Result;
                endcase
        end

        always @(presentState, resultAccepted, inReady, startFP) begin

                shiftCtrl = 1'b0; initM1 = 1'b0; initM2 = 1'b0; doneFP = 1'b0;
                resultReady = 1'b0; inAccept = 1'b0; bias = 8'b01111111;


                case (presentState)
                        GetAGetB: begin
                                resultReady = 0;
                                inAccept = 1;
                        end
                        BEGIN: begin
                                doneFP = 0;
                                inAccept = 0;
                        end
                        Add: begin
                                bias = 8'b01111111; // dec: 127
                        end
                        MUL: begin
                                initM1 = 1;
                                initM2 = 1;
                                shiftCtrl = mulRes47 ? 1:0;
                        end
                        Result: begin
                                doneFP = 1;
                                resultReady = 1;
                        end
                endcase
        end
```



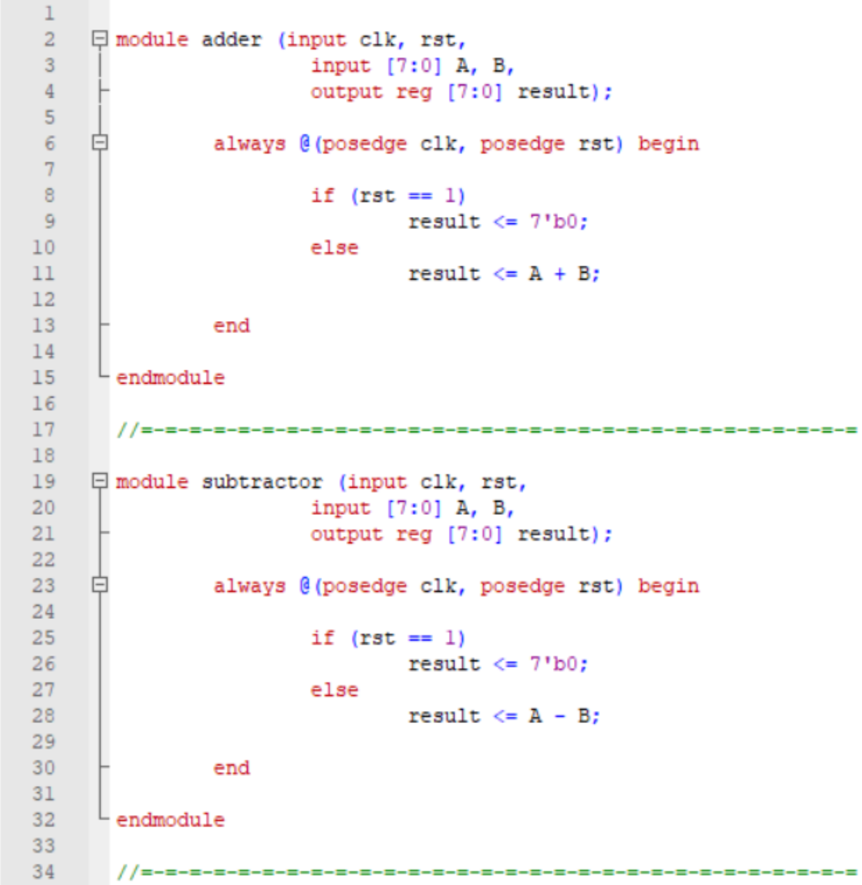
```

        always @(posedge clk, posedge rst) begin
                if (rst == 1)
                        presentState <= Idle;
                else
                        presentState <= nextState;
        end

endmodule
```

بخش پنجم : My Floating-Point Multiplier Verilog

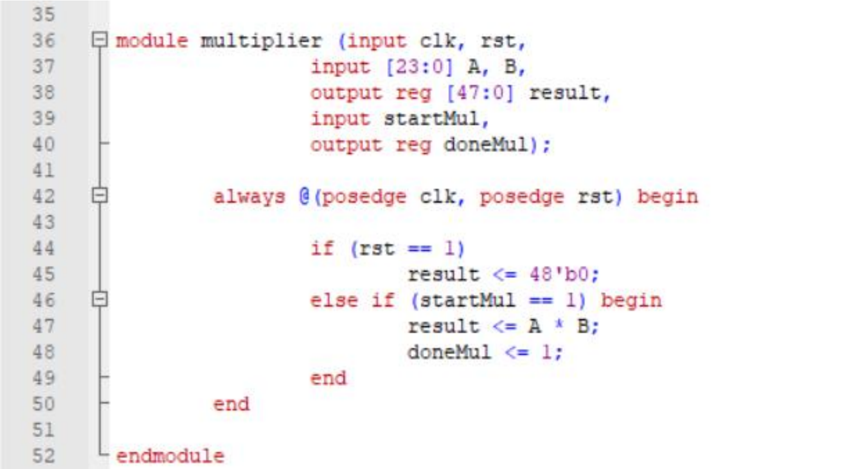


```

module myFloatingPointMultiplier (input clk, rst,
                input [31:0] A, B,
                output [31:0] result,
                input inReady, resultAccepted, startFP, startMul,
                output doneFP, resultReady, inAccept, doneMul);

        wire initM1, initM2, shiftCtrl, mulRes47;
        wire [7:0] bias;

        datapath dP (
                        .clk(clk),
                        .rst(rst),
                        .initM1(initM1),
                        .initM2(initM2),
                        .bias(bias),
                        .shiftCtrl(shiftCtrl),
                        .startMul(startMul),
                        .A(A),
                        .B(B),
                        .result(result),
                        .mulRes47(mulRes47),
                        .doneMul(doneMul)
                        );

        controller control (
                        .clk(clk),
                        .rst(rst),
                        .inReady(inReady),
                        .resultAccepted(resultAccepted),
                        .mulRes47(mulRes47),
                        .startFP(startFP),
                        .shiftCtrl(shiftCtrl),
                        .initM1(initM1),
                        .initM2(initM2),
                        .doneFP(doneFP),
                        .resultReady(resultReady),
                        .bias(bias),
                        .inAccept(inAccept)
                        );

endmodule
```

بخش ششم : Other Components

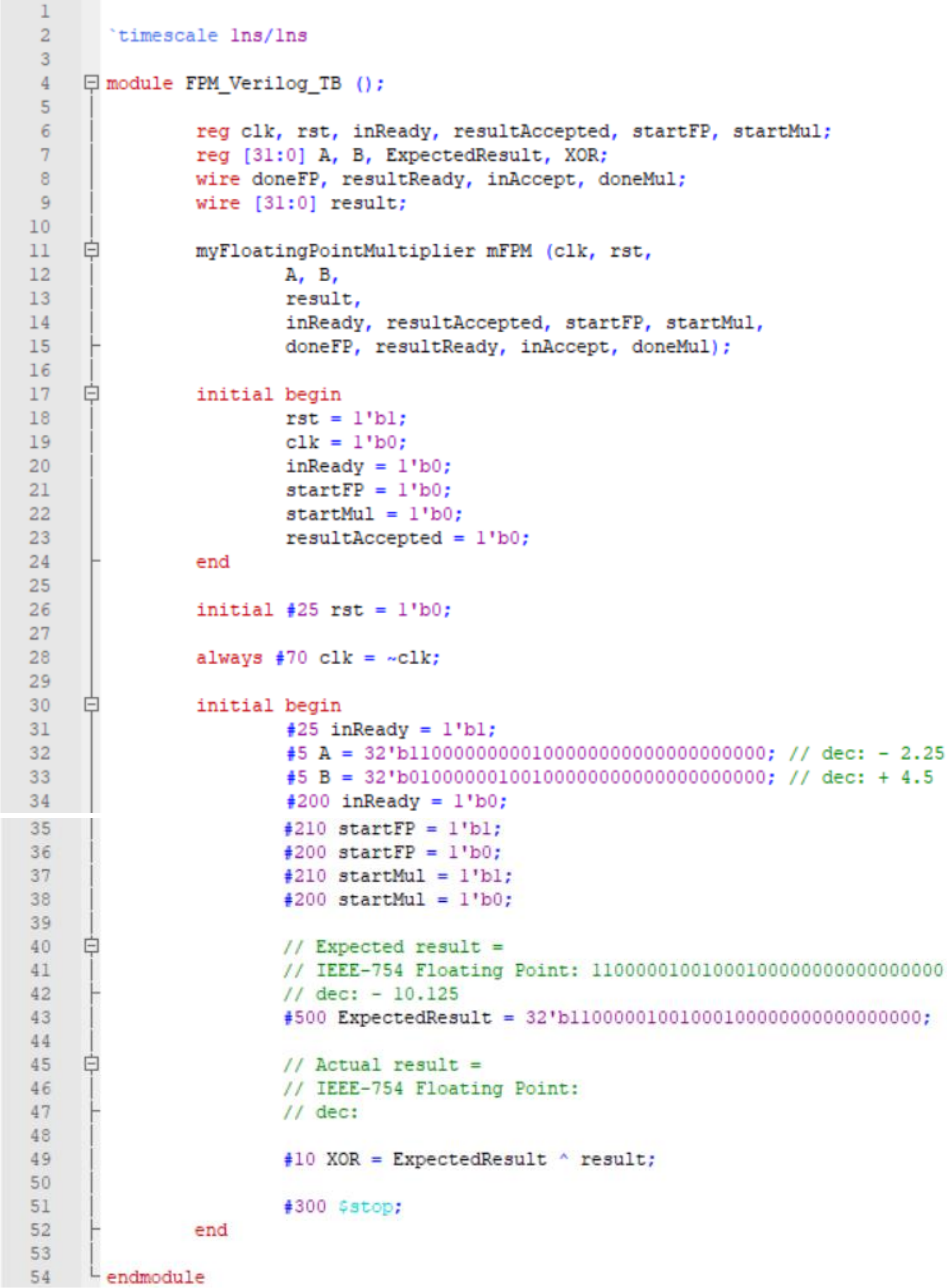


```
module adder (input clk, rst,
                input [7:0] A, B,
                output reg [7:0] result);

        always @(posedge clk, posedge rst) begin

                if (rst == 1)
                        result <= 7'b0;
                else
                        result <= A + B;

        end

endmodule

//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

module subtractor (input clk, rst,
                input [7:0] A, B,
                output reg [7:0] result);

        always @(posedge clk, posedge rst) begin

                if (rst == 1)
                        result <= 7'b0;
                else
                        result <= A - B;

        end

endmodule

//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
```

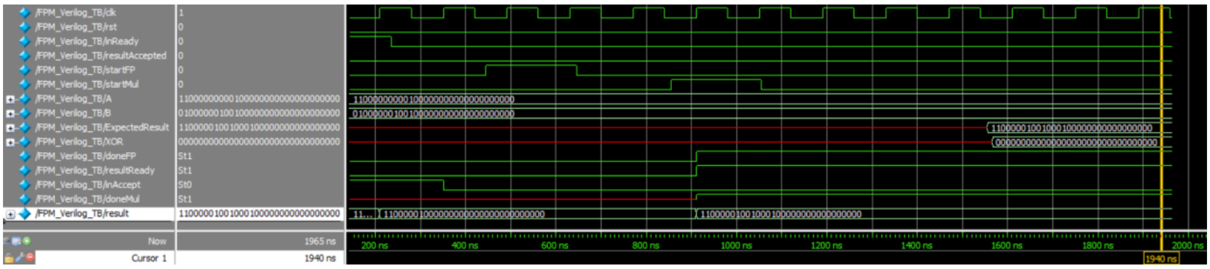


```

module multiplier (input clk, rst,
                input [23:0] A, B,
                output reg [47:0] result,
                input startMul,
                output reg doneMul);

        always @(posedge clk, posedge rst) begin

                if (rst == 1)
                        result <= 48'b0;
                else if (startMul == 1) begin
                        result <= A * B;
                        doneMul <= 1;
                end
        end

endmodule
```

بخش هفتم : FPM Verilog TB

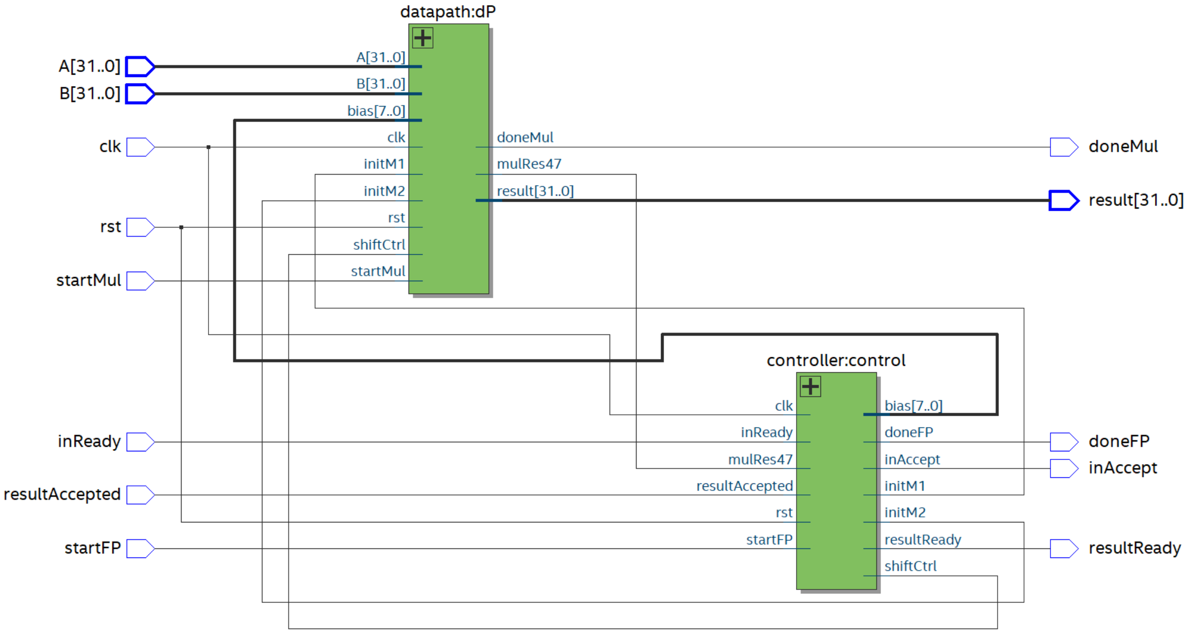


```verilog

`timescale 1ns/1ns

module FPM_Verilog_TB ();

        reg clk, rst, inReady, resultAccepted, startFP, startMul;
        reg [31:0] A, B, ExpectedResult, XOR;
        wire doneFP, resultReady, inAccept, doneMul;
        wire [31:0] result;

        myFloatingPointMultiplier mFPM (clk, rst,
                A, B,
                result,
                inReady, resultAccepted, startFP, startMul,
                doneFP, resultReady, inAccept, doneMul);

        initial begin
                rst = 1'b1;
                clk = 1'b0;
                inReady = 1'b0;
                startFP = 1'b0;
                startMul = 1'b0;
                resultAccepted = 1'b0;
        end

        initial #25 rst = 1'b0;

        always #70 clk = ~clk;

        initial begin
                #25 inReady = 1'b1;
                #5 A = 32'b11000000000100000000000000000000; // dec: - 2.25
                #5 B = 32'b01000000100100000000000000000000; // dec: + 4.5
                #200 inReady = 1'b0;
                #210 startFP = 1'b1;
                #200 startFP = 1'b0;
                #210 startMul = 1'b1;
                #200 startMul = 1'b0;

                // Expected result =
                // IEEE-754 Floating Point: 11000001001000100000000000000000
                // dec: - 10.125
                #500 ExpectedResult = 32'b11000001001000100000000000000000;

                // Actual result =
                // IEEE-754 Floating Point:
                // dec:

                #10 XOR = ExpectedResult ^ result;

                #300 $stop;
        end

endmodule
```

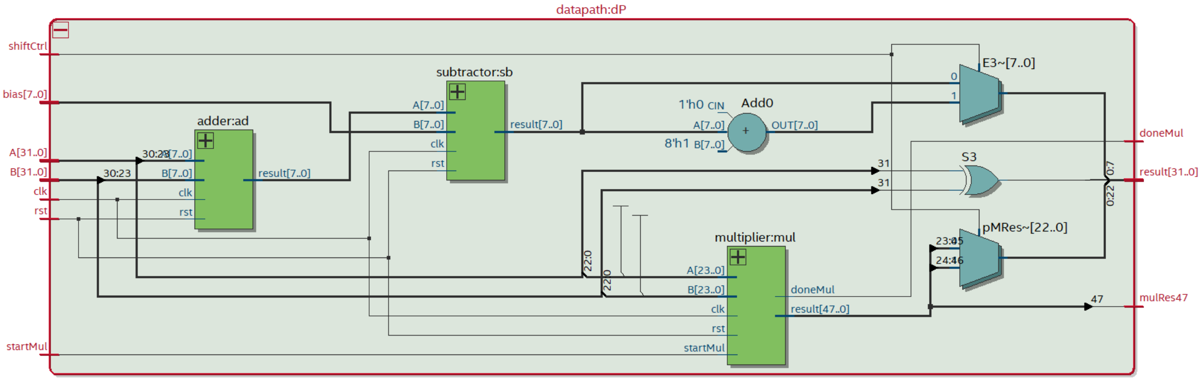


```
/FPM_Verilog_TB/clk             1
/FPM_Verilog_TB/rst             0
/FPM_Verilog_TB/inReady         0
/FPM_Verilog_TB/resultAccepted  0
/FPM_Verilog_TB/startFP         0
/FPM_Verilog_TB/startMul        0
/FPM_Verilog_TB/A               11000000000100000000000000000000
/FPM_Verilog_TB/B               01000000100100000000000000000000
/FPM_Verilog_TB/ExpectedResult  11000001001000100000000000000000
/FPM_Verilog_TB/XOR             00000000000000000000000000000000
/FPM_Verilog_TB/doneFP          St1
/FPM_Verilog_TB/resultReady     St1
/FPM_Verilog_TB/inAccept        St0
/FPM_Verilog_TB/doneMul         St1
/FPM_Verilog_TB/result          11000001001000100000000000000000

Now       1965 ns
Cursor 1  1940 ns
```

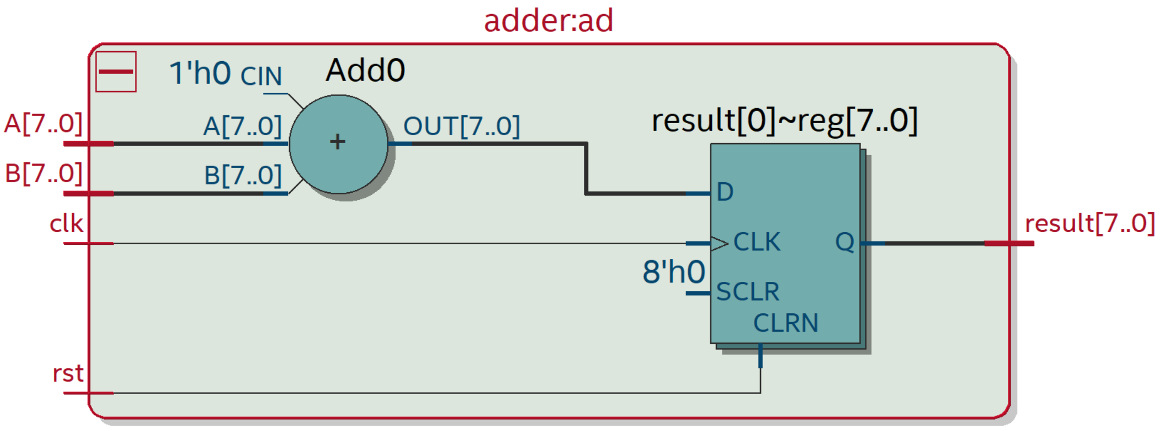
بخش هشتم : My Floating-Point Multiplier Synthesis

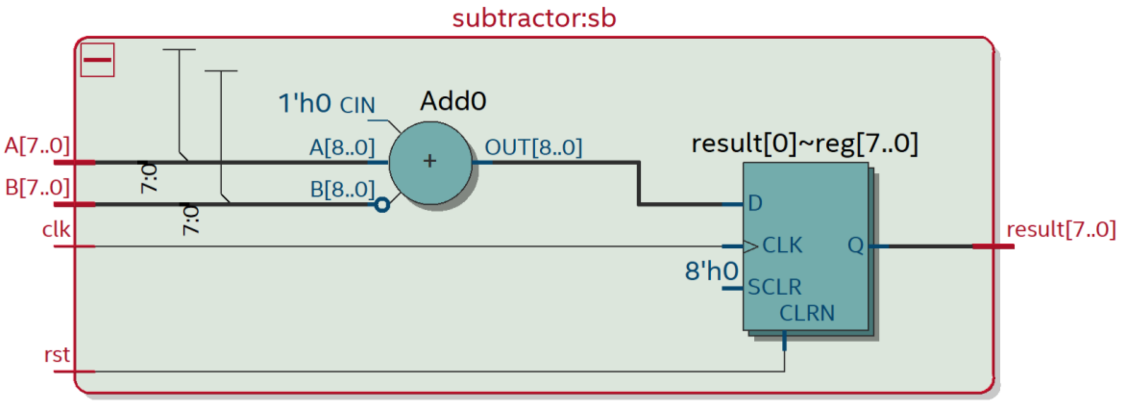


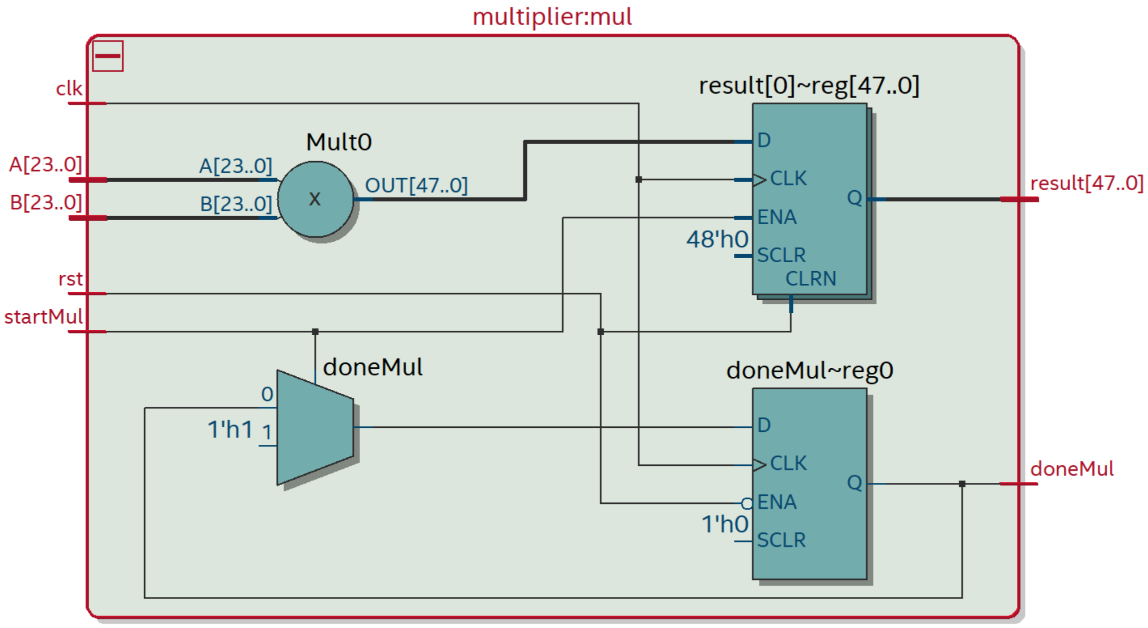
بخش نهم : Datapath Synthesis



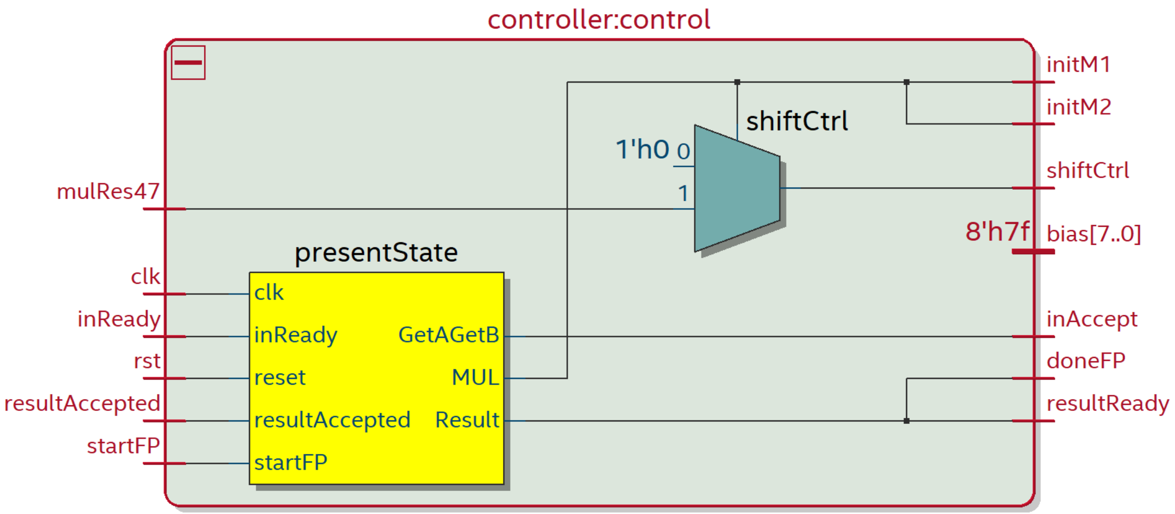
بخش دهم : Other Components Synthesis

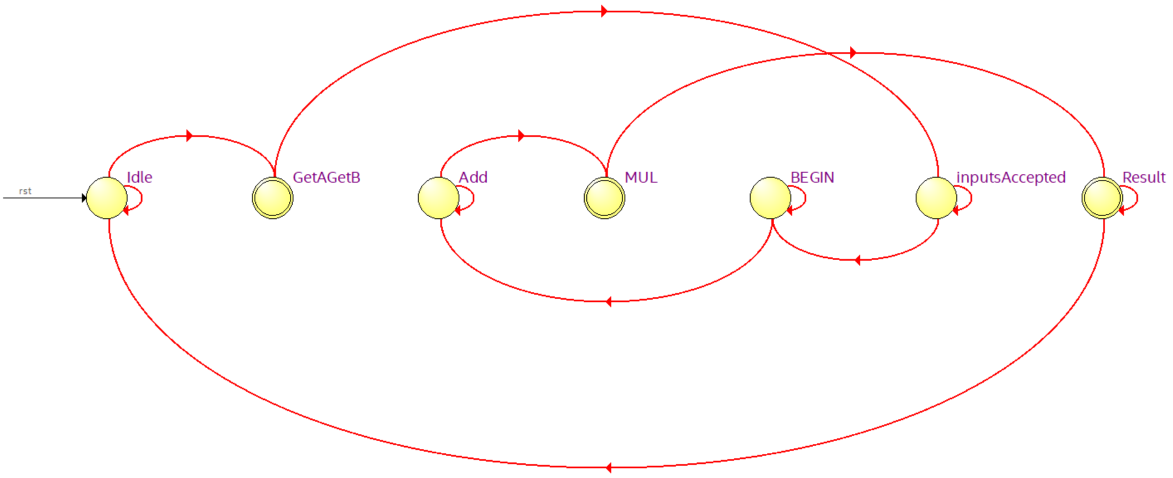




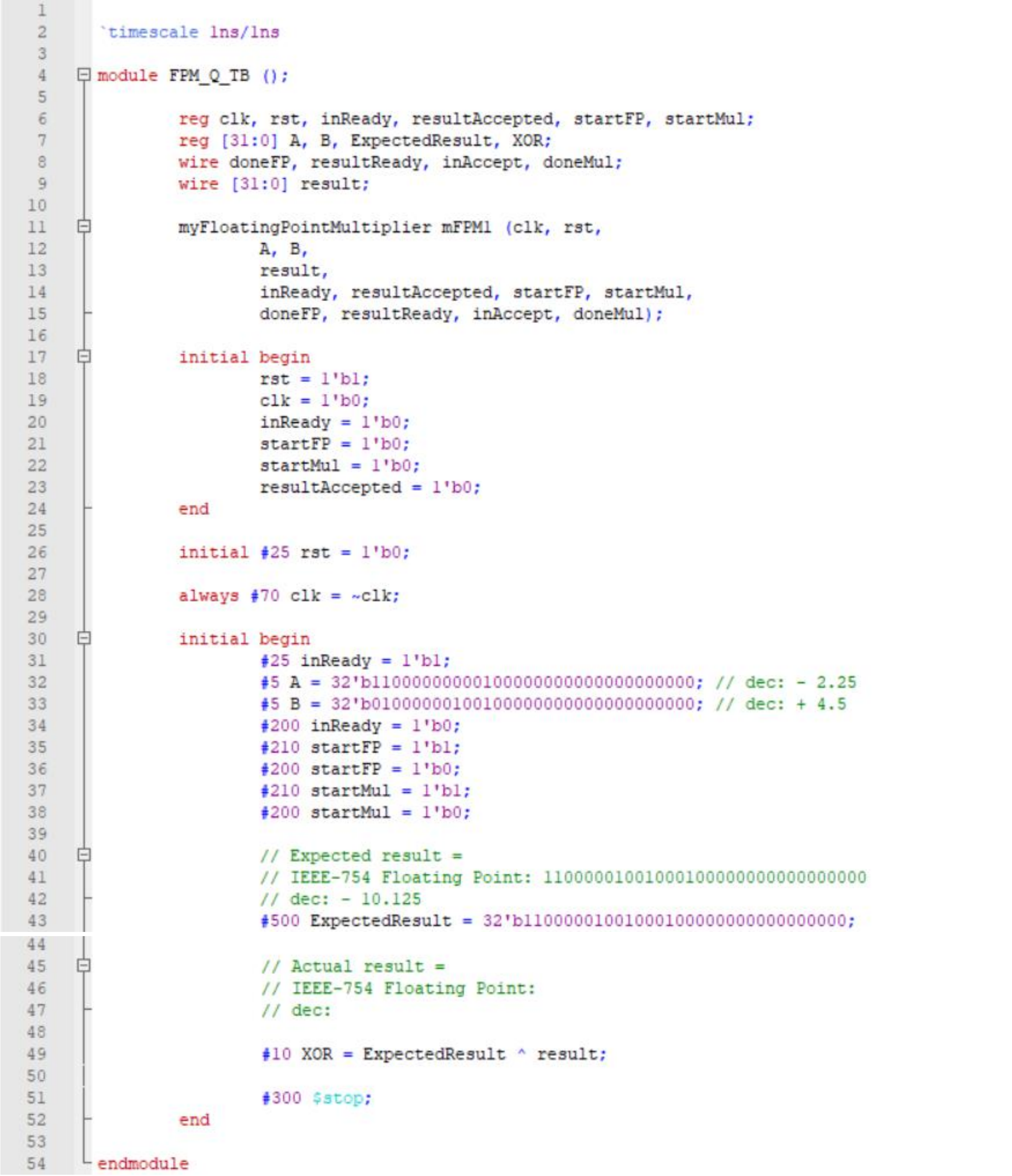


بخش یازدهم : Controller Synthesis

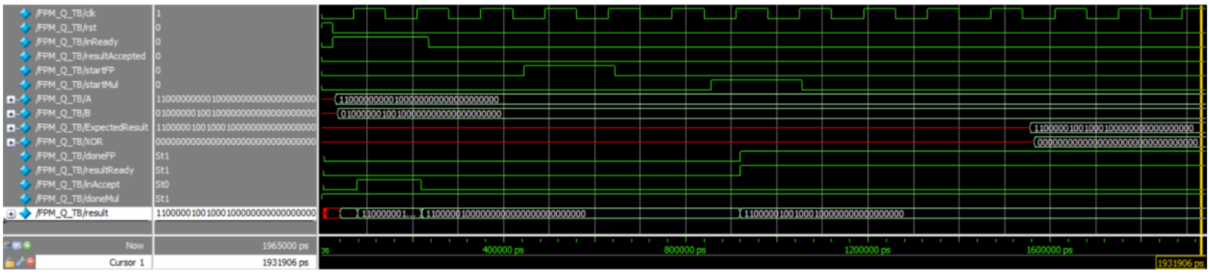




بخش دوازدهم : FPM Q TB

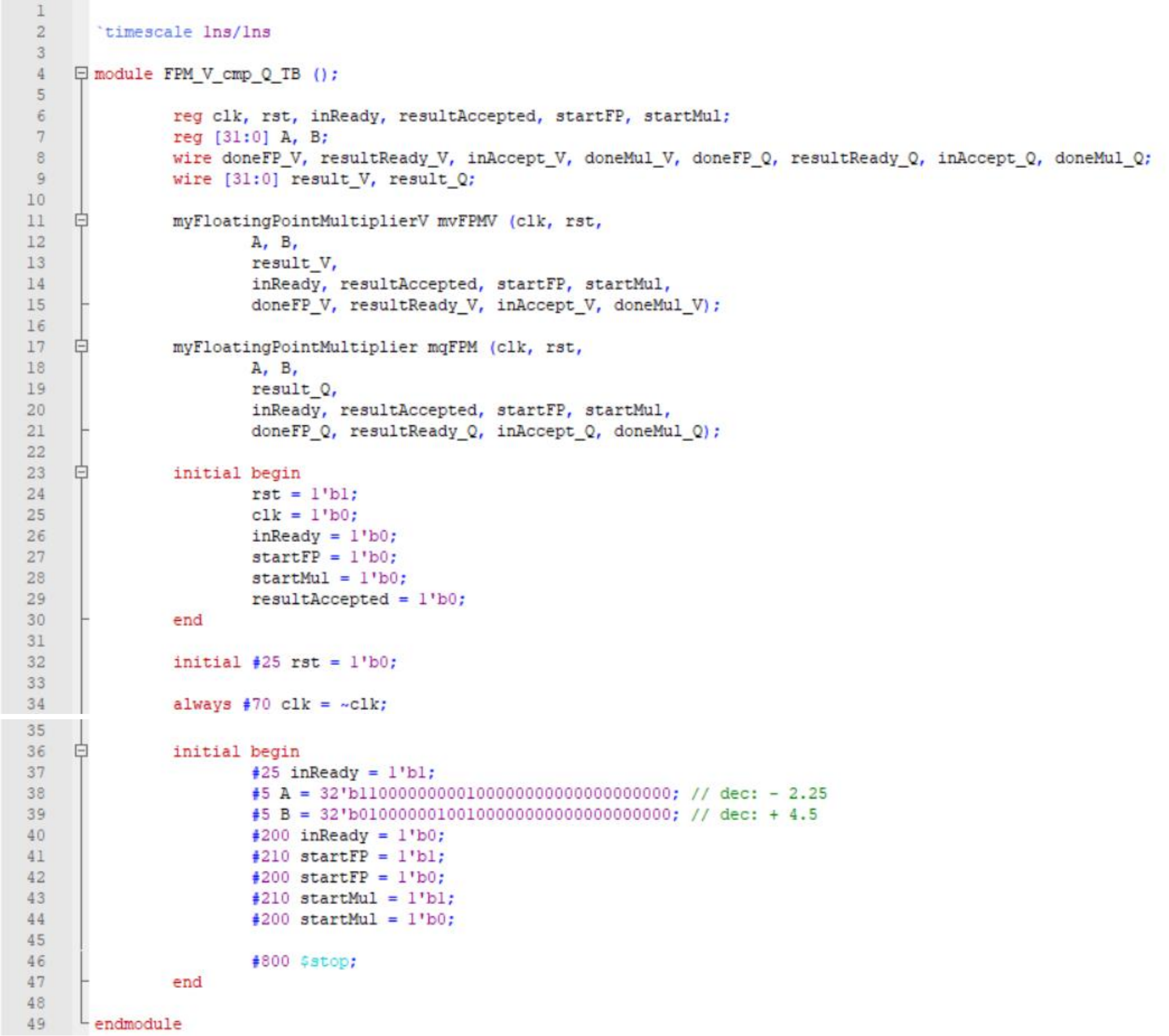


```
`timescale 1ns/1ns

module FPM_Q_TB ();

        reg clk, rst, inReady, resultAccepted, startFP, startMul;
        reg [31:0] A, B, ExpectedResult, XOR;
        wire doneFP, resultReady, inAccept, doneMul;
        wire [31:0] result;

        myFloatingPointMultiplier mFPM1 (clk, rst,
                A, B,
                result,
                inReady, resultAccepted, startFP, startMul,
                doneFP, resultReady, inAccept, doneMul);

        initial begin
                rst = 1'b1;
                clk = 1'b0;
                inReady = 1'b0;
                startFP = 1'b0;
                startMul = 1'b0;
                resultAccepted = 1'b0;
        end

        initial #25 rst = 1'b0;

        always #70 clk = ~clk;

        initial begin
                #25 inReady = 1'b1;
                #5 A = 32'b11000000000100000000000000000000; // dec: - 2.25
                #5 B = 32'b01000000100100000000000000000000; // dec: + 4.5
                #200 inReady = 1'b0;
                #210 startFP = 1'b1;
                #200 startFP = 1'b0;
                #210 startMul = 1'b1;
                #200 startMul = 1'b0;

                // Expected result =
                // IEEE-754 Floating Point: 11000001001000100000000000000000
                // dec: - 10.125
                #500 ExpectedResult = 32'b11000001001000100000000000000000;

                // Actual result =
                // IEEE-754 Floating Point:
                // dec:

                #10 XOR = ExpectedResult ^ result;

                #300 $stop;
        end

endmodule
```

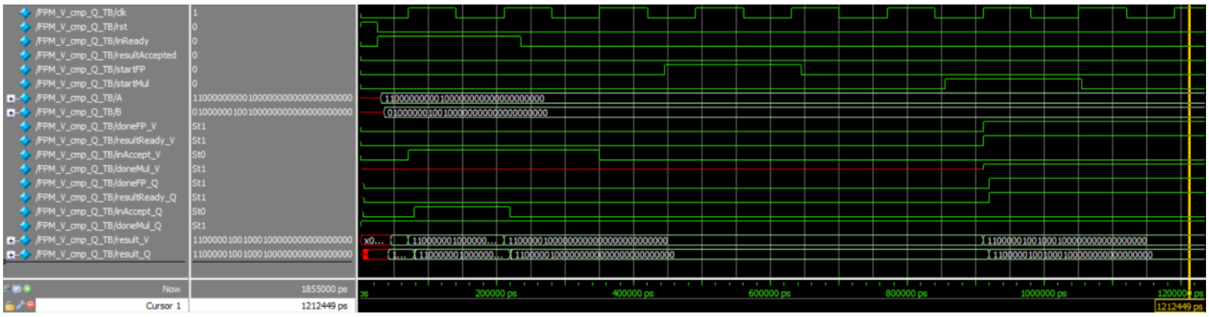


```
/FPM_Q_TB/clk              1
/FPM_Q_TB/rst              0
/FPM_Q_TB/inReady          0
/FPM_Q_TB/resultAccepted   0
/FPM_Q_TB/startFP          0
/FPM_Q_TB/startMul         0
/FPM_Q_TB/A                11000000000100000000000000000000
/FPM_Q_TB/B                01000000100100000000000000000000
/FPM_Q_TB/ExpectedResult   11000001001000100000000000000000
/FPM_Q_TB/XOR              00000000000000000000000000000000
/FPM_Q_TB/doneFP           St1
/FPM_Q_TB/resultReady      St1
/FPM_Q_TB/inAccept         St0
/FPM_Q_TB/doneMul          St1
/FPM_Q_TB/result           11000001001000100000000000000000
Now                        1965000 ps
Cursor 1                   1931906 ps
```

بخش سیزدهم : FPM V cmp Q TB



```
`timescale 1ns/1ns

module FPM_V_cmp_Q_TB ();

        reg clk, rst, inReady, resultAccepted, startFP, startMul;
        reg [31:0] A, B;
        wire doneFP_V, resultReady_V, inAccept_V, doneMul_V, doneFP_Q, resultReady_Q, inAccept_Q, doneMul_Q;
        wire [31:0] result_V, result_Q;

        myFloatingPointMultiplierV mvFPMV (clk, rst,
                A, B,
                result_V,
                inReady, resultAccepted, startFP, startMul,
                doneFP_V, resultReady_V, inAccept_V, doneMul_V);

        myFloatingPointMultiplier mqFPM (clk, rst,
                A, B,
                result_Q,
                inReady, resultAccepted, startFP, startMul,
                doneFP_Q, resultReady_Q, inAccept_Q, doneMul_Q);

        initial begin
                rst = 1'b1;
                clk = 1'b0;
                inReady = 1'b0;
                startFP = 1'b0;
                startMul = 1'b0;
                resultAccepted = 1'b0;
        end

        initial #25 rst = 1'b0;

        always #70 clk = ~clk;

        initial begin
                #25 inReady = 1'b1;
                #5 A = 32'b11000000000100000000000000000000; // dec: - 2.25
                #5 B = 32'b01000000100100000000000000000000; // dec: + 4.5
                #200 inReady = 1'b0;
                #210 startFP = 1'b1;
                #200 startFP = 1'b0;
                #210 startMul = 1'b1;
                #200 startMul = 1'b0;

                #800 $stop;
        end

endmodule
```



/FPM_V_cmp_Q_TB/clk	1
/FPM_V_cmp_Q_TB/rst	0
/FPM_V_cmp_Q_TB/inReady	0
/FPM_V_cmp_Q_TB/resultAccepted	0
/FPM_V_cmp_Q_TB/startFP	0
/FPM_V_cmp_Q_TB/startMul	0
/FPM_V_cmp_Q_TB/A	11000000000100000000000000000000
/FPM_V_cmp_Q_TB/B	01000000100100000000000000000000
/FPM_V_cmp_Q_TB/doneFP_V	St1
/FPM_V_cmp_Q_TB/resultReady_V	St1
/FPM_V_cmp_Q_TB/inAccept_V	St0
/FPM_V_cmp_Q_TB/doneMul_V	St1
/FPM_V_cmp_Q_TB/doneFP_Q	St1
/FPM_V_cmp_Q_TB/resultReady_Q	St1
/FPM_V_cmp_Q_TB/inAccept_Q	St0
/FPM_V_cmp_Q_TB/doneMul_Q	St1
/FPM_V_cmp_Q_TB/result_V	11000001001000100000000000000000
/FPM_V_cmp_Q_TB/result_Q	11000001001000100000000000000000
Now	1855000 ps
Cursor 1	1212449 ps