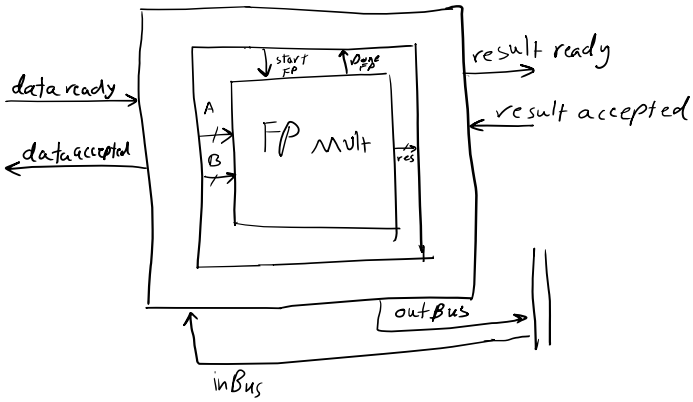
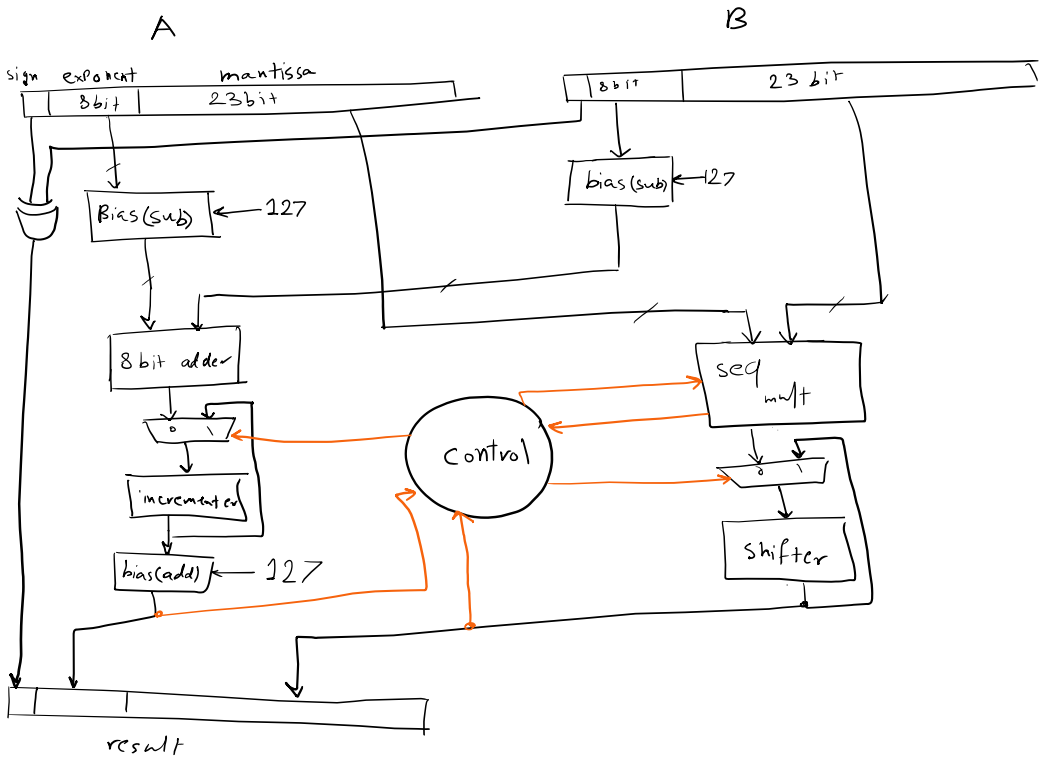
DLD CA 6

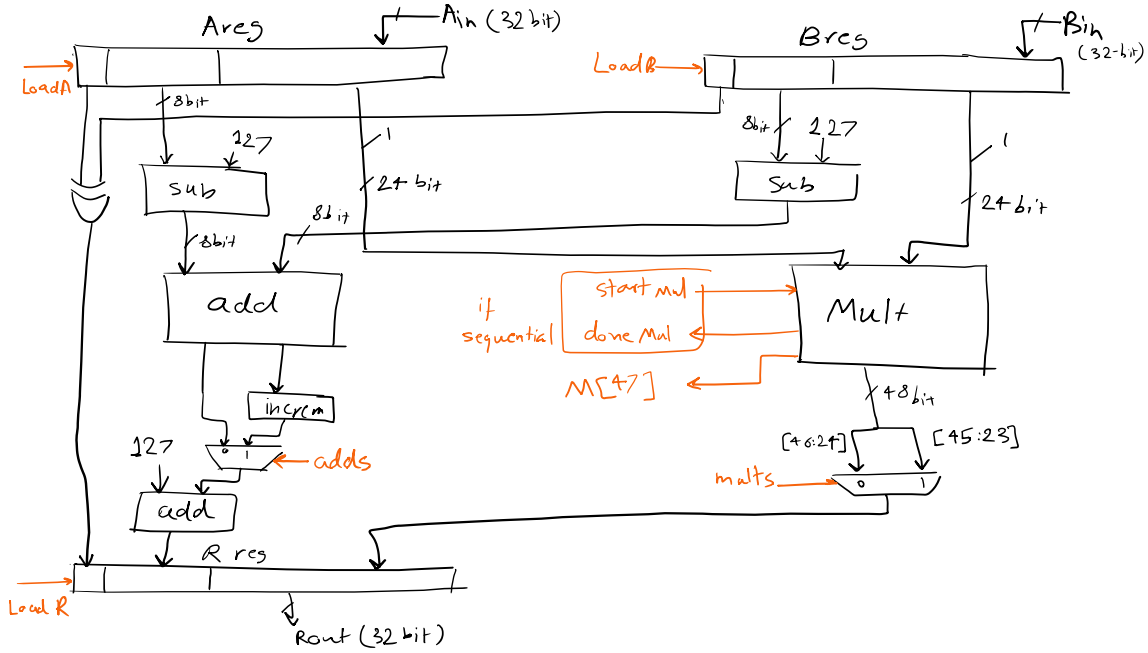
Mohammad Abaeiani

810198432

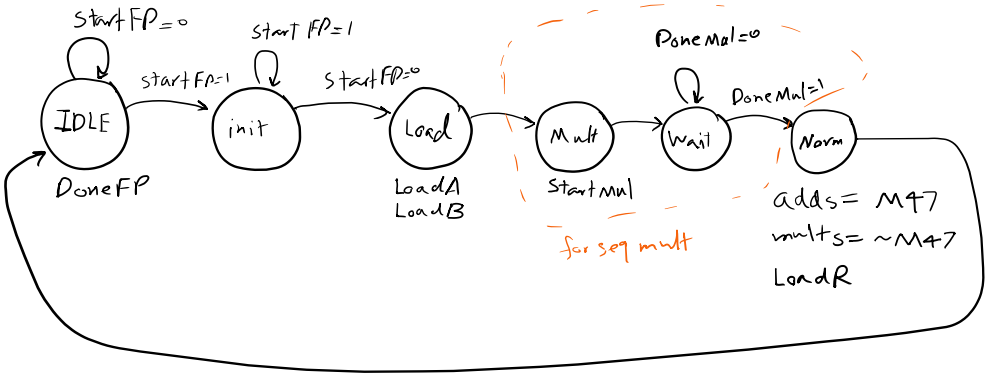

data ready
data accepted
start FP
Done FP
A
B
FP Mult
res
result ready
result accepted
outBus
inBus


A
B
sign
exponent
mantissa
8bit
23bit
8 bit
23 bit
Bias(sub)
127
bias(sub)
127
8 bit adder
seq mult
0
1
incrementer
control
bias(add)
127
0
1
shifter
result

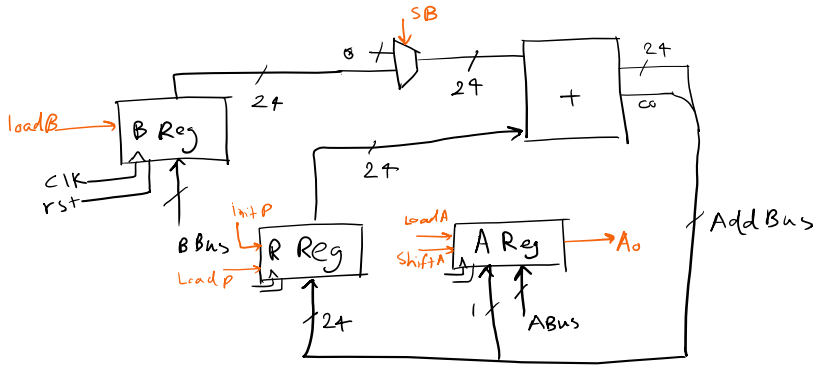
An alternative form of the design is like below (used in this project):



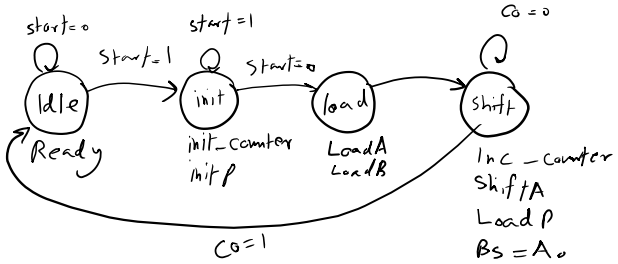
Controller :



Sequential multiplier:

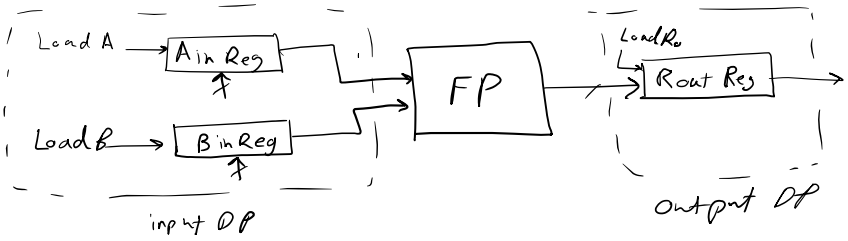


Controller:

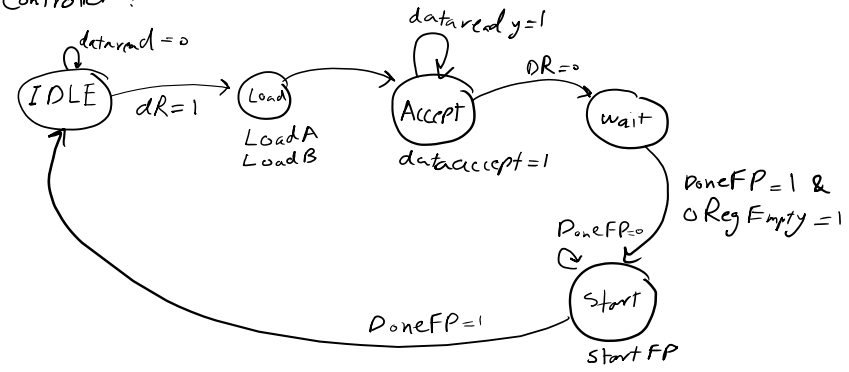


Wrapper:

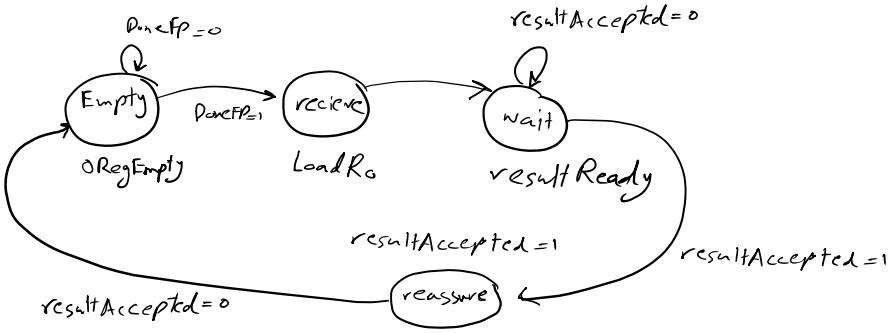
input wrapper:



input Controller:



Output controller



Code:

I used option d for this part

The plain FP multiplier with multiply operation instead of sequential.

Data path:

```
module FP_mult_DP(input clk, rst,adds,mults,LoadA,LoadB,LoadR,input  [31:0]Ain,Bin , output [31:0]Rbus, output M47);

    wire As;
    wire Bs;
    wire Ws;
    wire [7:0] Ae;
    wire [7:0] Be;
    wire [7:0] We;
    wire [22:0] Am;
    wire [22:0] Bm;
    wire [47:0] M;
    wire [22:0] Wm;
    logic [31:0] Areg,Breg,Rreg;

    parameter [6:0] Bias = 127;

    always @ (posedge clk,posedge rst) begin
        if(rst)
            Areg<= 32'b0;
        else
            if(LoadA)
                Areg<=Ain;
    end

    always @ (posedge clk,posedge rst) begin
        if(rst)
            Breg<= 32'b0;
        else
            if(LoadB)
                Breg<=Bin;
    end

    always @ (posedge clk,posedge rst) begin
        if(rst)
            Rreg<= 32'b0;
        else
            if(LoadR)
                Rreg<={Ws,We,Wm};
    end

    //sign of output

    assign As = Areg[31];
    assign Bs = Breg[31];
    assign Ws = As^Bs;

    //exponent of output

    assign Ae = Areg[30:23] - Bias;
    assign Be = Breg[30:23] - Bias;
    assign We = (adds)? Ae + Be + Bias +1 : Ae + Be + Bias;

    //Mantissa operations

    assign Am = Areg[22:0];
    assign Bm = Breg[22:0];
    assign M = {1'b1,Am} * {1'b1,Bm};
    assign M47 = M[47];
    assign Wm = (~mults)? M[46:24]: M[45:23];

    assign Rbus = Rreg;
endmodule
```

Controller:

```
module FP_mult_ctrl(input clk,rst,startFP,M47, output logic LoadA,LoadB,LoadR,adds,mults,doneFP);

    logic [1:0] ps,ns;
    parameter [1:0]
    Idle = 0, Init = 1,load =2,norm = 3;

    always @(ps,startFP,M47) begin
        ns = 0;
        {LoadA,LoadB,LoadR,adds,mults,doneFP} = 6'b0;
        case(ps)
            Idle: begin ns = (startFP) ? Init:Idle; doneFP = 1'b1;end
            Init:  ns = (startFP) ? Init: load;
            load: begin ns = norm; LoadA = 1'b1; LoadB = 1'b1;end
            norm: begin ns = Idle; adds = M47; mults = ~M47; LoadR = 1'b1;end
        endcase
    end

    always @( posedge clk,posedge rst) begin
        if(rst)
            ps<= Idle;
        else
            ps<=ns;
    end
endmodule
```

Top level:

```
module FP_top(input clk,rst,startFP,input [31:0]Abus,Bbus,output [31:0]ResultBus,output doneFP);
    wire M47;
    wire LoadA,LoadB,LoadR,adds,mults;

    FP_mult_DP dp(clk,rst,adds,mults,LoadA,LoadB,LoadR,Abus,Bbus,ResultBus,M47);
    FP_mult_ctrl cu(clk,rst,startFP,M47,LoadA,LoadB,LoadR,adds,mults,doneFP);

endmodule
```

Output:

As an example I did

14.79 * -98.13 = 1451.3427

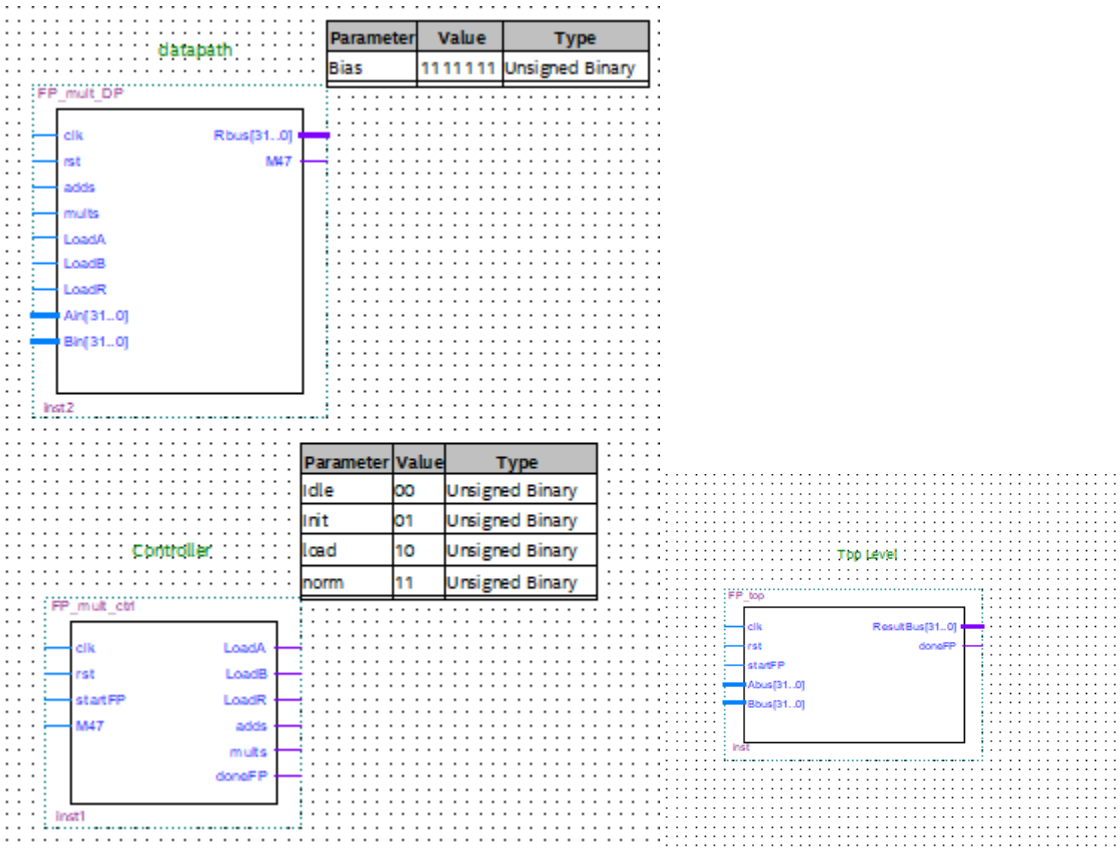
The output is 11000100101101010110101011110111 which is -1451.342651 in decimal and its fairly close to the exact output.

Synthesis:

Report:

Flow Status	Successful - Sun Jun 27 16:26:21 2021
Quartus Prime Version	20.1.0 Build 711 06/05/2020 SJ Lite Edition
Revision Name	FP_top
Top-level Entity Name	FP_top
Family	Cyclone IV E
Device	EP4CE15F23C9L
Timing Models	Final
Total logic elements	127 / 15,408 (< 1 %)
Total registers	64
Total pins	100 / 344 (29 %)
Total virtual pins	0
Total memory bits	0 / 516,096 (0 %)
Embedded Multiplier 9-bit elements	7 / 112 (6 %)
Total PLLs	0 / 4 (0 %)

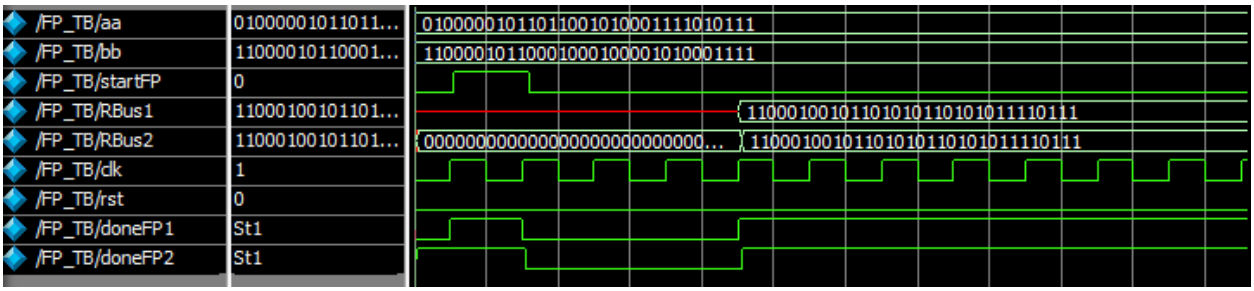
Symbols:



Test bench for pre and post simulation:

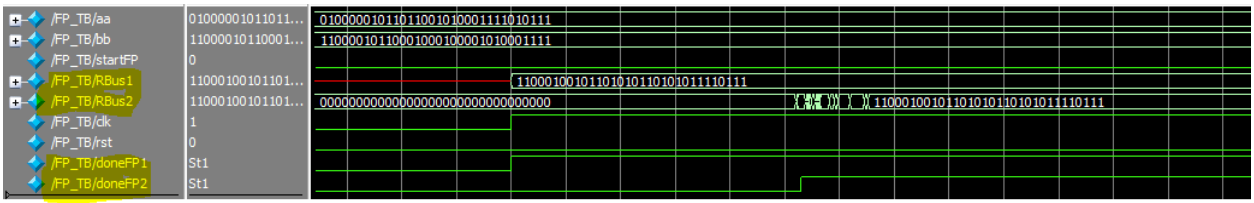
```
`timescale 1ns/1ns

module FP_TB();
//14.79 * -98.13 = 1451.3427
logic [31:0] aa = 32'b01000001011011001010001111010111;
logic [31:0] bb = 32'b11000010110001000100001010001111;
logic startFP = 0;
wire [31:0] RBus1;
wire [31:0] RBus2;
logic clk = 0;
logic rst = 0;
wire doneFP1;
wire doneFP2;
//output is 11000100101101010110101011110111 whic is -1451.342651

FP_top FP1(clk,rst,startFP,aa,bb,RBus1,doneFP1);
FP_topQ FP2(clk,rst,startFP,aa,bb,RBus2,doneFP2);

always #100 clk = ~clk;

initial begin
#110 startFP =1;
#210 startFP = 0;
#2000 $stop;
end
endmodule
```

Wave Form:



RBus1 and FP1 are for pre synthesis and RBus2 and FP2 are for post synthesis

If we zoom in we find that the post synthesis one triggers with a bit of delay and also it prepares the output in different steps.



Sequential mult:

Datapath:

```
module Mult24_DP (input clk,rst,loadA,loadB,loadP,shiftA,initP,Bsel,
         input[23:0] Abus,Bbus, output [47:0] multresult , output A0);
  reg [23:0] Areg,Breg,Preg;
  wire  [23:0] B_and;
  wire  [24:0] AddBus;
  always @(posedge clk , posedge rst) begin
    if (rst)
      Breg <= 24'b0;
    else
      if(loadB)
        Breg <= Bbus;
  end
  always @(posedge clk , posedge rst) begin
    if (rst)
      Preg <= 24'b0;
    else begin
         if(initP)
        Preg <= 24'b0;
        else
          if(loadP)
            Preg <= AddBus [24:1];
          end
  end
  always @(posedge clk , posedge rst)begin
    if (rst)
      Areg <= 24'b0;
    else begin
      if(loadA)
        Areg <= Abus ;
        else
          if(shiftA)
            Areg <= {AddBus[0],Areg[23:1]};
      end
  end
  assign B_and = Bsel ? Breg : 24'b0;
  assign AddBus = B_and + Preg;
  assign multresult = {Preg,Areg};
  assign A0 = Areg[0];
endmodule
```

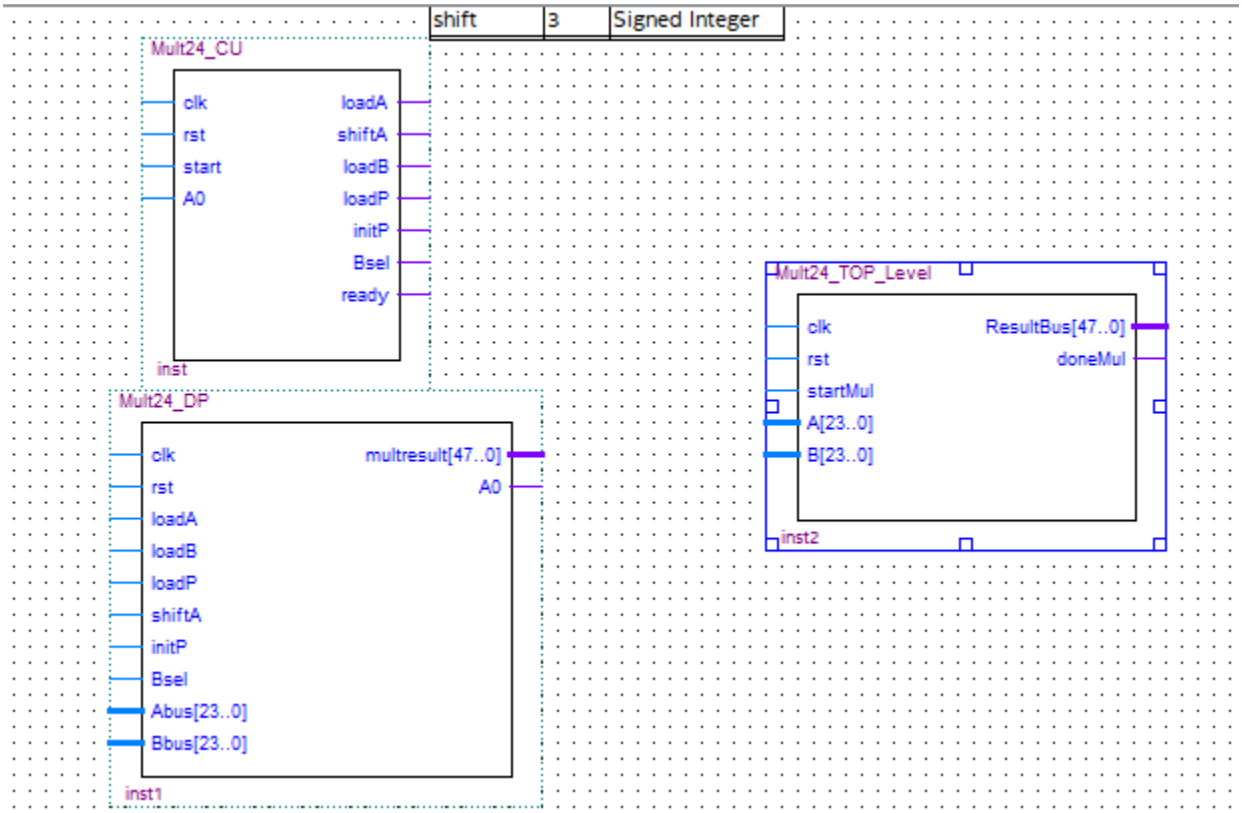
Controller and top level:

```
//Sequential Multiplier controller module:
module Mult24_CU (input clk,rst,start,A0,output reg loadA,shiftA,loadB,loadP,initP,Bsel,ready);
  wire Co;
  reg Init_counter , Inc_counter;
  reg [1:0] pstate, nstate;
  reg [4:0] Count;
  parameter Idle = 0 , Init = 1 , load = 2 , shift = 3;
  always@(pstate,start,A0,Co) begin
    nstate = 0;
    {loadA,shiftA,loadB,loadP,initP,Bsel,ready} = 7'b0;
    {Init_counter,Inc_counter} = 2'b0;
      case(pstate)
        Idle : begin nstate = start ? Init : Idle; ready = 1'b1; end
        Init : begin nstate = start ? Init : load; Init_counter = 1'b1;initP = 1'b1;end
        load : begin nstate = shift ; loadA = 1'b1;loadB = 1'b1; end
        shift : begin nstate = Co ? Idle : shift;Inc_counter = 1'b1;shiftA = 1'b1;loadP = 1'b1; Bsel = A0;end
      endcase
  end
  always @(posedge clk,posedge rst) begin
    if(rst)
    pstate <= Idle;
    else
    pstate <= nstate;
  end
  always @(posedge clk,posedge rst) begin
    if (rst) Count <= 5'd0;
    else begin
      if(Init_counter) Count <= 5'd8;
        else
          if(Inc_counter) Count <= Count +1;
  end
  end
  assign Co = & Count;
endmodule

//Sequential Multiplier top level module:
module Mult24_TOP_Level (input clk , rst , startMul ,input [23:0] A,B , output [47:0] ResultBus , output doneMul);
  wire A0;
  wire loadA,shiftA,loadB,loadP,InitP,Bsel;
  Mult24_DP dp(clk,rst,loadA,loadB,loadP,shiftA,InitP,Bsel,A,B,ResultBus,A0);
  Mult24_CU co(clk,rst,startMul,A0,loadA,shiftA,loadB,loadP,InitP,Bsel,doneMul);
endmodule
```

Synthesis report:

Flow Status	Successful - Sun Jun 27 19:05:37 2021
Quartus Prime Version	20.1.0 Build 711 06/05/2020 SJ Lite Edition
Revision Name	Mult24_TOP_Level
Top-level Entity Name	Mult24_TOP_Level
Family	Cyclone IV E
Device	EP4CE55F23C8
Timing Models	Final
Total logic elements	112 / 55,856 (< 1 %)
Total registers	81
Total pins	100 / 325 (31 %)
Total virtual pins	0
Total memory bits	0 / 2,396,160 (0 %)
Embedded Multiplier 9-bit elements	0 / 308 (0 %)
Total PLLs	0 / 4 (0 %)

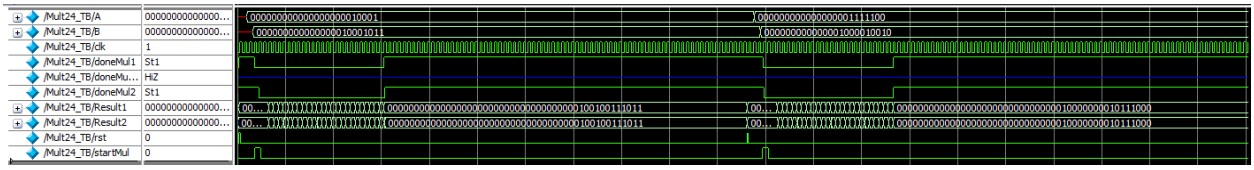
Symbols:



Test Bench:

```
`timescale 1ns/1ns
module Mult24_TB();
  reg [23:0]A,B;
  reg clk = 0, rst = 0 , startMul = 0;
  wire [47:0]Result1,Result2;
  wire doneMulldoneMul2;
  Mult24_TOP_Level M1(clk,rst,startMul,A,B,Result1,doneMul1);
  Mult24_TOP_Level0 M3(clk,rst,startMul,A,B,Result2,doneMul2);
  always #50 clk <= ~clk;
  initial begin
    #30 rst = 1;
    #30 rst = 0;
    #130 A = 23'd17;
    #130 B = 23'd139;
    #30 startMul = 1;
    #130 startMul = 0;
    #10130 rst = 1;
    #30 rst = 0;
    #130 A = 23'd124;
    #130 B = 23'd530;
    #30 startMul = 1;
    #130 startMul = 0;
    #10000 $stop;
  end
endmodule
```

Waveform :



1 is for pre synthesis and 2 is for post synthesis

Overall there is a tiny delay after synthesis