

# Design of DSP48A1 Slice

## Spartan-6-FPGAs



SPARTAN6-FPGA

## Table of Contents

<b>1. RTL code:</b>	<b>1</b>
A. Parameterized Reg_Mux:	1
B. Parameterized 4x1 MUX:	1
C. Pre_Add_Sub:	2
D. Multiplier	2
E. Post_Add_Sub:	3
F. Top Module:	4
<b>2. Test Bench Code:</b>	<b>7</b>
<b>3. Do File:</b>	<b>10</b>
<b>4. QuestaSim Snippets</b>	<b>10</b>
<b>5. Constraint File</b>	<b>10</b>
<b>6. Elaboration:</b>	<b>11</b>
<b>7. Synthesis</b>	<b>11</b>
<b>8. Implementation</b>	<b>13</b>
<b>9. Linting:</b>	<b>15</b>

## 1. RTL code:

### A. Parameterized Reg\_Mux:

```
1  module Param_Reg_2x1MUX #(
2      parameter WIDTH = 2,
3      parameter RSTTYPE = "SYNC", // "SYNC" or "ASYNC"
4      parameter REG = 1
5  ) (
6      input  [WIDTH-1:0] in,
7      input          clk,
8      input          clk_enable, // Clock enable signal
9      input          rst,
10     output reg [WIDTH-1:0] out
11 );
12 //Registering the first input based on the reset type
13 generate
14     if(REG == 0) begin
15         always @(in) begin
16             out = in;
17         end
18     end else begin
19         if (RSTTYPE == "ASYNC") begin
20             always @(posedge clk or posedge rst) begin
21                 if (rst) begin
22                     out <= {WIDTH{1'b0}};
23                 end else if(clk_enable)begin
24                     out <= in;
25                 end
26             end
27         end else if (RSTTYPE == "SYNC") begin
28             always @(posedge clk) begin
29                 if (rst) begin
30                     out <= {WIDTH{1'b0}};
31                 end else if(clk_enable)begin
32                     out <= in;
33                 end
34             end
35         end
36     end
37 end
38 endmodule
```

### B. Parameterized 4x1 MUX:

```
1  module Param_4x1MUX(in1, in2, in3, in4, sel, out);
2
3      parameter WIDTH = 2;
4      input  [WIDTH-1:0] in1, in2, in3, in4;
5      input  [1:0] sel;
6      output [WIDTH-1:0] out;
7
8      assign out = (sel == 2'b00) ? in1 :
9                  (sel == 2'b01) ? in2 :
10                 (sel == 2'b10) ? in3 :
11                 (sel == 2'b11) ? in4 : {WIDTH{1'b0}}; // Default case to avoid latches
12
13 endmodule
```



### C. Pre\_Add\_Sub:

```
1  module Pre_Add_Sub #(
2      parameter WIDTH = 18 // Width of the input and output ports
3  ) (
4      input [WIDTH-1:0]  in1,
5      input [WIDTH-1:0]  in2,
6      input              sel, // Select signal for addition or subtraction
7      output [WIDTH-1:0] out
8  );
9
10 assign out = (sel) ? (in1 - in2) : (in1 + in2);
11 endmodule
```

### D. Multiplier

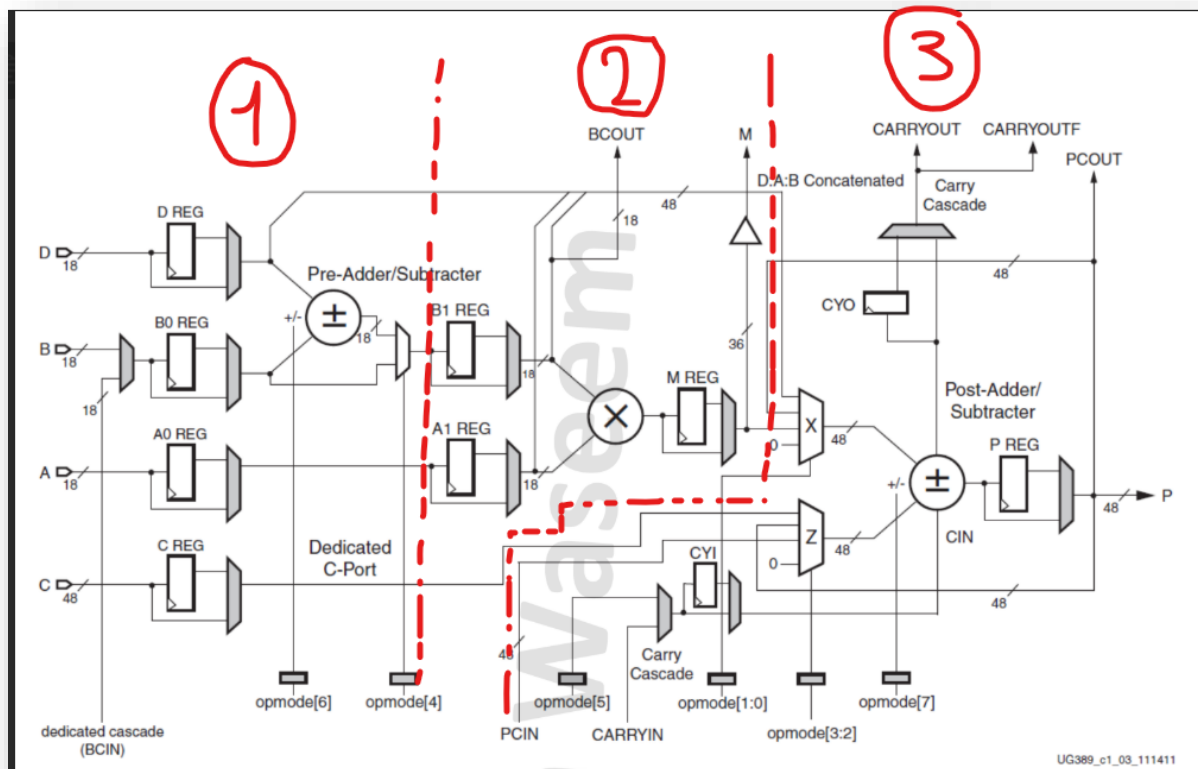
```
1  module MUL #(
2      parameter  WIDTH = 18,
3      localparam OUT_WIDTH = 2* WIDTH,
4      parameter  RSTTYPE = "SYNC", // "SYNC" or "ASYNC"
5      parameter  REG_OUT = 1 // Registering the output
6  ) (
7      input [WIDTH-1:0]  in1,
8      input [WIDTH-1:0]  in2,
9      input              clk,
10     input              clk_enable_out, // Clock enable signals
11     input              rst_out,
12     output [OUT_WIDTH-1:0] out
13 );
14
15 // Internal signal for the registered input
16 wire [OUT_WIDTH-1:0] out_reg;
17
18 //Instantiate the parameterized register module for in1 and in2
19 Param_Reg_2x1MUX #(.WIDTH(OUT_WIDTH), .RSTTYPE(RSTTYPE), .REG(REG_OUT)) MREG (out_reg, clk, clk_enable_out, rst_out, out);
20
21 assign out_reg = in1 * in2;
22
23
24 endmodule
```

## E. Post\_Add\_Sub

```
1  module Post_Add_Sub #(
2      parameter WIDTH = 48, // Width of the input and output ports
3      parameter RSTTYPE = "SYNC", // "SYNC" or "ASYNC"
4      parameter REG_CIN = 1, // Registering the CIN
5      parameter REG_COUT = 1, // Registering the Cout
6      parameter REG_OUT = 1 // Registering the output
7  ) (
8      input [WIDTH-1:0] mux_X_in1,
9      input [WIDTH-1:0] mux_X_in2,
10     input [WIDTH-1:0] mux_X_in3,
11     // input [WIDTH-1:0] mux_X_in4 = 0,
12     input [1:0] mux_X_sel,
13
14     input [WIDTH-1:0] mux_Z_in1,
15     input [WIDTH-1:0] mux_Z_in2,
16     input [WIDTH-1:0] mux_Z_in3,
17     // input [WIDTH-1:0] mux_Z_in4 = 0,
18     input [1:0] mux_Z_sel,
19     input cin,
20     input sel, // Select signal for addition or subtraction
21     input clk,
22     input clk_enable_carry, // Clock enable signals
23     input clk_enable_out, // Clock enable signals
24     input rst_carry,
25     input rst_out,
26     output [WIDTH-1:0] out,
27     output cout
28 );
29
30 // Internal signal for the registered input
31 wire [WIDTH-1:0] out_reg;
32 reg [WIDTH-1:0] mux_X_out;
33 reg [WIDTH-1:0] mux_Z_out;
34 wire cin_reg;
35 wire cout_reg;
36
37 //Instantiate the parameterized register module for in1 and in2
38 Param_Reg_2x1MUX #(.WIDTH(1), .RSTTYPE(RSTTYPE),.REG(REG_CIN)) CY1 (cin, clk, clk_enable_carry, rst_carry, cin_reg);
39 Param_Reg_2x1MUX #(.WIDTH(1), .RSTTYPE(RSTTYPE),.REG(REG_COUT)) CY0 (cout_reg, clk, clk_enable_carry, rst_carry, cout);
40 Param_Reg_2x1MUX #(.WIDTH(WIDTH), .RSTTYPE(RSTTYPE),.REG(REG_OUT)) PREG (out_reg, clk, clk_enable_out, rst_out, out);
41
42 //MUX for X inputs
43 always @(mux_X_sel or mux_X_in1 or mux_X_in2 or mux_X_in3) begin
44
45     case (mux_X_sel)
46         2'b01: mux_X_out = mux_X_in1;
47         2'b10: mux_X_out = mux_X_in2;
48         2'b11: mux_X_out = mux_X_in3;
49         default: mux_X_out = 0; // Default case to handle unexpected values
50     endcase
51
52 end
53
54 //MUX for Z inputs
55 always @(mux_Z_sel or mux_Z_in1 or mux_Z_in2 or mux_Z_in3) begin
56
57     case (mux_Z_sel)
58         2'b01: mux_Z_out = mux_Z_in1;
59         2'b10: mux_Z_out = mux_Z_in2;
60         2'b11: mux_Z_out = mux_Z_in3;
61         default: mux_Z_out = 0; // Default case to handle unexpected values
62     endcase
63
64 end
65
66 //Adder LOGIC
67 assign {cout_reg, out_reg} = (sel) ? (mux_Z_out - (mux_X_out + cin_reg)) : (mux_Z_out + mux_X_out + cin_reg);
68 endmodule
```

## F. Top Module:

**Note: The Design implementation is divide into the following Stages**



**Parameters and I/O ports:**

```

1 module Spartan6_DSP48A1 #(
2     parameter A0REG = 0,          //define the number of pipeline registers in the A and B input paths.
3     parameter A1REG = 1,
4     parameter B0REG = 0,
5     parameter B1REG = 1,
6     parameter CREG = 1,          //value of 0 or 1. The number defines the snumber of pipeline stages. Default: 1 (registered)
7     parameter DREG = 1,
8     parameter MREG = 1,
9     parameter PREG = 1,
10    parameter CARRYINREG = 1,
11    parameter CARRYOUTREG = 1,
12    parameter OPMODEREG = 1,
13    parameter CARRYINSEL = "OPMODE5", /*used in the carry cascade input, either...
14    parameter B_INPUT = "DIRECT", /*The B_INPUT attribute defines whether the input to the B port is...
15    parameter RSTTYPE = "SYNC" /*The RSTTYPE attribute defines the type of reset used in the DSP48A1 slice...
16 ) (
  
```

```

6 //Input Ports
7 input [17:0] A,
8 input [17:0] B,
9 input [47:0] C,
10 input [17:0] D,
11 input [17:0] BCIN, //Cascade input for Port B
12 input CARRYIN,
13 //Control Ports
14 input CLK, //DSP CLK
15 input [7:0] OPMODE, //Control input to select the arithmetic operations of the DSP48A1 slice.
16 //Clock Enable Input Ports
17 input CEA, // Clock enable for the A port registers: (AOREG & A1REG)
18 input CEB, // Clock enable for the B port registers: (BOREG & B1REG)
19 input CEC, // Clock enable for the C port registers (CREG)
20 input CECARRYIN, // Clock enable for the carry-in register (CYI) and the carry-out register (CYO)
21 input CED, // Clock enable for the D port register (DREG)
22 input CEM, // Clock enable for the multiplier register (MREG)
23 input CEOPMODE, // Clock enable for the opmode register (OPMODEREG)
24 input CEP, // Clock enable for the P output port registers (PREG = 1)
25 //Output Ports
26 > output [35:0] M, /*multiplier data output, It is either the output of the M register (MREG = 1) ...
27 > output [47:0] P, /*Primary data output from the post-adder/subtractor. It is either the ...
28 > output CARRYOUT, /*It can be registered in (CARRYOUTREG = 1) or unregistered (CARRYOUTREG = 0)...
29 output CARRYOUTF, //copy of the CARRYOUT signal that can be routed to the user logic.
30
31 //Reset Input Ports: All the resets are active high reset.
32 input RSTA, // Reset for the A registers: (AOREG & A1REG)
33 input RSTB, // Reset for the B registers: (BOREG & B1REG)
34 input RSTC, // Reset for the C registers (CREG)
35 input RSTCARRYIN, // Reset for the carry-in register (CYI) and the carry-out register (CYO)
36 input RSTD, // Reset for the D register (DREG)
37 input RSTM, // Reset for the multiplier register (MREG)
38 input RSTOPMODE, // Reset for the opmode register (OPMODEREG)
39 input RSTP, // Reset for the P output registers (PREG = 1)
40 // Cascade Ports:
41 output [17:0] BCOUT, //Cascade output for Port B
42 input [47:0] PCIN, //Cascade input for Port P
43 output [47:0] PCOUT //Cascade output for Port P
44 );

```

## Stage 1:

```

69 //=====
70 //-----Stage 1: A, B, C, D, and pre_Add_Sub-----
71 //=====
72 /*Inputs: A, B, C, D, BCIN, OPMODE[6], OPMODE[4]
73 Outputs: A_reg, C_reg, D_reg, B_bypass_mux_out
74 */
75
76 //B, D, pre_add_sub
77 wire [17:0] pre_adder_out;
78 wire [17:0] D_reg;
79 wire [17:0] B0_reg;
80 wire [17:0] B_bypass_mux_out;
81 //input A, C
82 wire [17:0] A0_reg;
83 wire [47:0] C_reg;
84 //OPMODE
85 wire [7:0] OPMODE_reg;
86
87 Param_Reg_2x1MUX #(.WIDTH(8), .RSTTYPE(RSTTYPE), .REG(OPMODEREG)) OPMODE_reg_mux (.in(OPMODE), .clk(CLK), .clk_enable(CEOPMODE), .rst(RSTOPMODE), .out(OPMODE_reg));
88
89 wire [17:0] B_temp;
90
91 generate
92     if(B_INPUT == "DIRECT")
93         assign B_temp = B;
94     else if(B_INPUT == "CASCADE")
95         assign B_temp = BCIN;
96     else
97         assign B_temp = 0; // Default value if neither DIRECT nor CASCADE is selected
98 endgenerate
99
100 Param_Reg_2x1MUX #(.WIDTH(18), .RSTTYPE(RSTTYPE), .REG(DREG)) D_reg_mux (.in(D), .clk(CLK), .clk_enable(CED), .rst(RSTD), .out(D_reg));
101 Param_Reg_2x1MUX #(.WIDTH(18), .RSTTYPE(RSTTYPE), .REG(BOREG)) B0REG_reg_mux (.in(B_temp), .clk(CLK), .clk_enable(CEB), .rst(RSTB), .out(B0_reg));
102 Param_Reg_2x1MUX #(.WIDTH(18), .RSTTYPE(RSTTYPE), .REG(AOREG)) A0REG_reg_mux (.in(A), .clk(CLK), .clk_enable(CEA), .rst(RSTA), .out(A0_reg));
103 Param_Reg_2x1MUX #(.WIDTH(48), .RSTTYPE(RSTTYPE), .REG(CREG)) CREG_reg_mux (.in(C), .clk(CLK), .clk_enable(CEC), .rst(RSTC), .out(C_reg));
104
105 Pre_Add_Sub #(.WIDTH(18)) pre_add_sub (.in1(D_reg), .in2(B0_reg), .sel(OPMODE_reg[6]), .out(pre_adder_out));
106
107
108 assign B_bypass_mux_out = (OPMODE_reg[4]) ? pre_adder_out : B0_reg; // MUX for pre_add_sub output
109
110

```

## Stage 2:

```
112 //=====
113 //-----Stage 2: Multiplier-----
114 //=====
115 /*
116 Inputs: A_reg, D_reg, B_bypass_mux_out
117 Outputs: BCOUNT, M, D:A:B Cont
118 */
119
120 wire [47:0] D_A_B_cont; // D:A:B Cont
121 wire [17:0] B1_reg;
122 wire [17:0] A1_reg;
123
124
125 //Multiplier
126 Param_Reg_2x1MUX #(.WIDTH(18), .RSTTYPE(RSTTYPE), .REG(B1REG)) B1REG_reg_mux (.in(B_bypass_mux_out), .clk(CLK), .clk_enable(CEB), .rst(RSTB), .out(B1_reg));
127 Param_Reg_2x1MUX #(.WIDTH(18), .RSTTYPE(RSTTYPE), .REG(A1REG)) A1REG_reg_mux (.in(A0_reg), .clk(CLK), .clk_enable(CEA), .rst(RSTA), .out(A1_reg));
128 MUL #(.WIDTH(18), .RSTTYPE(RSTTYPE), .REG_OUT(MREG)) mul (
129     .in1(B1_reg),
130     .in2(A1_reg),
131     .clk(CLK),
132     .clk_enable_out(CEM), // Clock enable signals for output
133     .rst_out(RSTM),
134     .out(M)
135 );
136
137 assign BCOUNT = B1_reg; // B input registered output
138 assign D_A_B_cont = {D_reg[11:0], A1_reg, B1_reg}; // D:A:B Cont output
139
```

## Stage 3:

```
140 //=====
141 //-----Stage 3: Post Add/Sub-----
142 //=====
143
144 /*Inputs: C_reg, M, D:A:B Cont, PCIN, CARRYIN, opmode[3:0], opmode[5], opmode[7]
145 Outputs: P, CARRYOUT, PCOUT, CARRYOUTF
146 */
147 wire [47:0] M_extended; // Extended M output
148 //Post Add/Sub
149
150 wire cin_temp;
151
152 generate
153     if(CARRYINSEL == "CARRYIN")
154         assign cin_temp = CARRYIN;
155     else if(CARRYINSEL == "OPMODE5")
156         assign cin_temp = OPMODE_reg[5];
157     else
158         assign cin_temp = 0; // Default value if neither CARRYIN nor OPMODE5 is selected
159 endgenerate
160
161 assign M_extended = { 12'b0, M }; // Extend M to match the width of P
162
163
164 Post_Add_Sub #(
165     .WIDTH(48),
166     .RSTTYPE(RSTTYPE),
167     .REG_CIN(CARRYINREG),
168     .REG_COUT(CARRYOUTREG),
169     .REG_OUT(PREG)
170 ) post_add_sub (
171     .mux_X_in1(M_extended), // D:A:B Cont
172     .mux_X_in2(P),
173     .mux_X_in3(D_A_B_cont),
174     .mux_X_sel(OPMODE_reg[1:0]), // Select signal for MUX X inputs
175     .mux_Z_in1(PCIN),
176     .mux_Z_in2(P),
177     .mux_Z_in3(C_reg),
178     .mux_Z_sel(OPMODE_reg[3:2]), // Select signal for MUX Z inputs
179     .cin(cin_temp),
180     .sel(OPMODE_reg[7]), // Select signal for addition or subtraction
181     .clk(CLK),
182     .clk_enable_carry(CECARRYIN), // Clock enable signals for carry-in and carry-out
183     .clk_enable_out(CEP), // Clock enable signals for output
184     .rst_carry(RSTCARRYIN),
185     .rst_out(RSTP),
186     .out(P),
187     .cout(CARRYOUT)
188 );
189
190 assign CARRYOUTF = CARRYOUT; // Copy of the CARRYOUT signal that can be routed to the user logic
191 assign PCOUT = P; // Cascade output for Port P
192
193 endmodule
```



## 2. Test Bench Code:

### Parameters and Stimulus declaration:

```
1 module tb_dsp48a1_00000000;
2   parameter A0REG = 0;
3   parameter A1REG = 1;
4   parameter B0REG = 0;
5   parameter B1REG = 1;
6   parameter CREG = 1;
7   parameter DREG = 1;
8   parameter MREG = 1;
9   parameter PREG = 1;
10  parameter CARRYINREG = 1;
11  parameter CARRYOUTREG = 1;
12  parameter OPMODEREG = 1;
13  parameter CARRYINSEL = "OPMODE5";
14  parameter B_INPUT = "DIRECT";
15  parameter RSTTYPE = "SYNC";
16
17  //stimulus and clock generation
18  //Input Ports
19  reg [17:0] A;
20  reg [17:0] B;
21  reg [47:0] C;
22  reg [17:0] D;
23  reg [17:0] BCIN; //Cascade input for Port B
24  reg CARRYIN;
25  //Control Ports
26  reg CLK; //DSP CLK
27  reg [7:0] OPMODE; //Control input to select the arithmetic operations of the DSP48A1 slice.
28  //Clock Enable Input Ports
29  reg CEA; // Clock enable for the A port registers: (A0REG & A1REG)
30  reg CEB; // Clock enable for the B port registers: (B0REG & B1REG)
31  reg CEC; // Clock enable for the C port registers (CREG)
32  reg CECARRYIN; // Clock enable for the carry-in register (CYI) and the carry-out register (CYO)
33  reg CED; // Clock enable for the D port register (DREG)
34  reg CEM; // Clock enable for the multiplier register (MREG)
35  reg CEOPMODE; // Clock enable for the opmode register (OPMODEREG)
36  reg CEP; // Clock enable for the P output port registers (PREG = 1)
37  //Reset Input Ports: All the resets are active high reset.
38  reg RSTA; // Reset for the A registers: (A0REG & A1REG)
39  reg RSTB; // Reset for the B registers: (B0REG & B1REG)
40  reg RSTC; // Reset for the C registers (CREG)
41  reg RSTCARRYIN; // Reset for the carry-in register (CYI) and the carry-out register (CYO)
42  reg RSTD; // Reset for the D register (DREG)
43  reg RSTM; // Reset for the multiplier register (MREG)
44  reg RSTOPMODE; // Reset for the opmode register (OPMODEREG)
45  reg RSTP; // Reset for the P output registers (PREG = 1)
46
47  // Cascade Ports:
48  reg [47:0] PCIN;
49  wire [17:0] BCOUT;
50  wire [47:0] PCOUT;
51
52  //Output Ports
53  wire [35:0] M;
54  wire [47:0] P;
55  wire CARRYOUT;
56  wire CARRYOUTF;
```

## Instantiation:

```
59 // Instantiate the DSP48A1 module
60 Spartan6_DSP48A1 #(
61     .A0REG(A0REG),
62     .A1REG(A1REG),
63     .B0REG(B0REG),
64     .B1REG(B1REG),
65     .CREG(CREG),
66     .DREG(DREG),
67     .MREG(MREG),
68     .PREG(PREG),
69     .CARRYINREG(CARRYINREG),
70     .CARRYOUTREG(CARRYOUTREG),
71     .OPMODEREG(OPMODEREG),
72     .CARRYINSEL(CARRYINSEL),
73     .B_INPUT(B_INPUT),
74     .RSTTYPE(RSTTYPE)
75 ) dut
76 //Input Ports
77 .A(A),
78 .B(B),
79 .C(C),
80 .D(D),
81 .BCIN(BCIN),
82 .CARRYIN(CARRYIN),
83 //Control Ports
84 .CLK(CLK),
85 .OPMODE(OPMODE),
86 //Clock Enable Input Ports
87 .CEA(CEA),
88 .CEB(CEB),
89 .CEC(CEC),
90 .CECARRYIN(CECARRYIN),
91 .CED(CED),
92 .CEM(CEM),
93 .CEOPMODE(CEOPMODE),
94 .CEP(CEP),
95 //Output Ports
96 .M(M),
97 .P(P),
98 .CARRYOUT(CARRYOUT),
99 .CARRYOUTF(CARRYOUTF),
100 //Reset Input Ports
101 .RSTA(RSTA),
102 .RSTB(RSTB),
103 .RSTC(RSTC),
104 .RSTCARRYIN(RSTCARRYIN),
105 .RSTD(RSTD),
106 .RSTM(RSTM),
107 .RSTOPMODE(RSTOPMODE),
108 .RSTP(RSTP),
109
110 // Cascade Ports:
111 .BCOUT(BCOUT),
112 .PCIN(PCIN),
113 .PCOUT(PCOUT)
114 );
```

```

116 initial begin
117     CLK = 0;
118     forever
119         #1 CLK = ~CLK;
120 end
121 // Testbench
122 initial begin
123     // 2.1. Verify Reset Operation
124     RSTA = 1; RSTB = 1; RSTC = 1; RSTD = 1; RSTM = 1; RSTP = 1; RSTCARRYIN = 1; RSTOPMODE = 1;
125     A = $random; B = $random; C = $random; D = $random; CARRYIN = $random; OPMODE = $random;
126     CEA = $random; CEB = $random; CEC = $random; CED = $random; CEM = $random; CEP = $random;
127     CECARRYIN = $random; CEOPMODE = $random; PCIN = $random; BCIN = $random;
128
129     repeat(10) begin
130         @(negedge CLK);
131         if(P != 48'h000000000000) begin
132             $display("Test failed: in Reset");
133             $stop;
134         end
135     end
136     RSTA = 0; RSTB = 0; RSTC = 0; RSTD = 0; RSTM = 0; RSTP = 0; RSTCARRYIN = 0; RSTOPMODE = 0;
137     CEA = 1; CEB = 1; CEC = 1; CED = 1; CEM = 1; CEP = 1; CECARRYIN = 1; CEOPMODE = 1;
138
139
140     // 2.2. Verify DSP Path 1
141     A = 20; B = 10; C = 350; D = 25;
142     OPMODE = 8'b11011101;
143     BCIN = $random; CARRYIN = 0; PCIN = $random;
144
145     repeat(4)@(negedge CLK);
146     if(BCOUT != 18'hF || M != 36'h12c || P != 48'h32 || PCOUT != 48'h32 || CARRYOUT != 0 || CARRYOUTF != 0) begin
147
148         $display("Test failed: DSP Path 1");
149         $stop;
150     end
151
152     // 2.3. Verify DSP Path 2
153     A = 20; B = 10; C = 350; D = 25;
154     OPMODE = 8'b00010000;
155     BCIN = $random; CARRYIN = 0; PCIN = $random;
156
157     repeat(3)@(negedge CLK);
158     if(BCOUT != 18'h23 || M != 36'h2BC || P != 48'h0 || PCOUT != 0 || CARRYOUT != 0 || CARRYOUTF != 0) begin
159         $display("Test failed: DSP Path 2");
160         $stop;
161     end
162
163     // 2.4. Verify DSP Path 3
164     A = 20; B = 10; C = 350; D = 25;
165     OPMODE = 8'b00001010;
166     BCIN = $random; CARRYIN = 0; PCIN = $random;
167
168     repeat(3)@(negedge CLK);
169     if(BCOUT != 18'ha || M != 36'hc8 || P != 48'h0 || PCOUT != 0 || CARRYOUT != 0 || CARRYOUTF != 0) begin
170         $display("Test failed: DSP Path 3");
171         $stop;
172     end
173
174     // 2.5. Verify DSP Path 4
175     A = 5; B = 6; C = 350; D = 25; PCIN = 3000;
176     OPMODE = 8'b10100111;
177     BCIN = $random; CARRYIN = 0;
178
179     repeat(3)@(negedge CLK);
180     if(BCOUT != 18'h6 || M != 36'h1e || P != 48'hfe6fffec0bb1 || PCOUT != 48'hfe6fffec0bb1 || CARRYOUT != 1 || CARRYOUTF != 1) begin
181         $display("Test failed: DSP Path 4");
182         $stop;
183     end
184
185     $display("ALL TESTS PASSED");
186     $stop;
187
188 end
189 endmodule

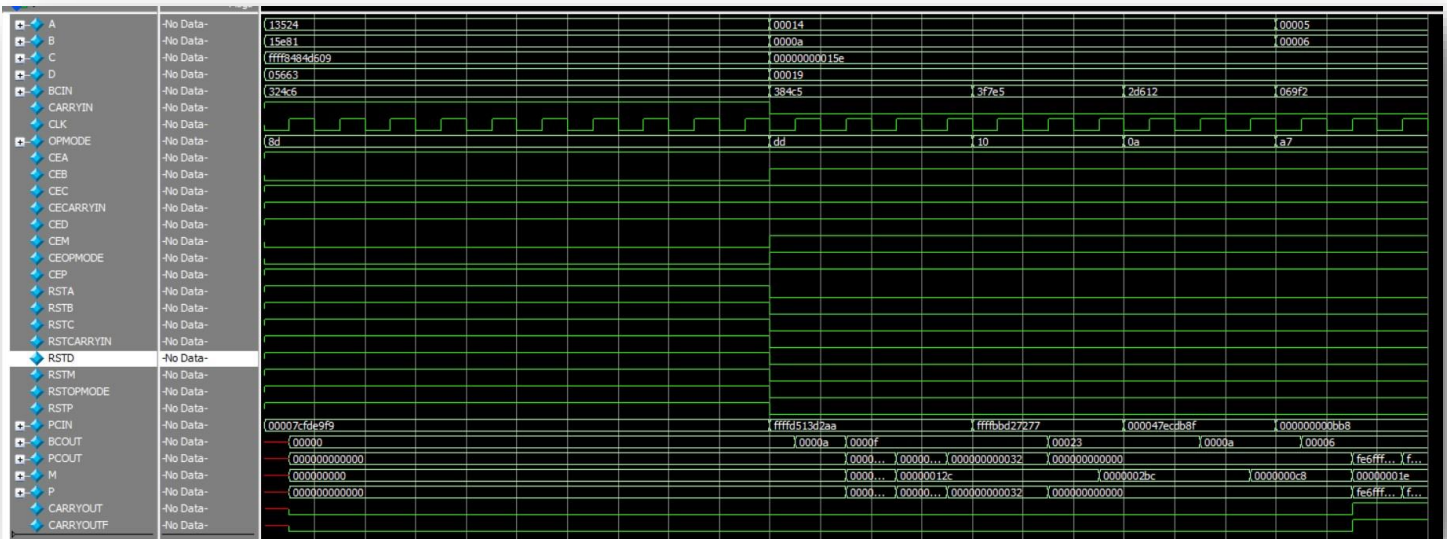
```

### 3. Do File:

Digital\_Design\_Diploma / Project\_1 / = run\_DSP48A1.do

```
1 vlib work
2 vlog Spartan6_DSP48A1.v Spartan6_DSP48A1_tb.v Mul.v Param_4x1MUX.v Param_Reg_2x1MUX.v Post_Add_Sub.v Pre_Add_Sub.v
3 vsim -voptargs=+acc work.Spartan6_DSP48A1_tb
4 add wave *
5 run -all
6 #quit -sim
```

#### 4. QuestaSim Snippets



## 5. Constraint File

```
1 ## This file is a general .xdc for the Basys3 rev B board
2 ## To use it in a project:
3 ## - uncomment the lines corresponding to used pins
4 ## - rename the used ports (in each line, after get_ports) according to the top level signal names in the project
5
6 # Clock signal
7 set_property -dict { PACKAGE_PIN W5    IOSTANDARD LVCMOS33 } [get_ports CLK]
8 create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports CLK]
9
10 # Reset signal
11 ## Configuration options, can be used for all designs
12 set_property CONFIG_VOLTAGE 3.3 [current_design]
13 set_property CFGBVS VCC0 [current_design]
14
15 ## SPI configuration mode options for QSPI boot, can be used for all designs
16 set_property BITSTREAM.GENERAL.COMPRESS TRUE [current_design]
17 set_property BITSTREAM.CONFIG.CONFIGRATE 33 [current_design]
18 set_property CONFIG_MODE SPIx4 [current_design]
```

## 6. Elaboration:

### Message Tab:

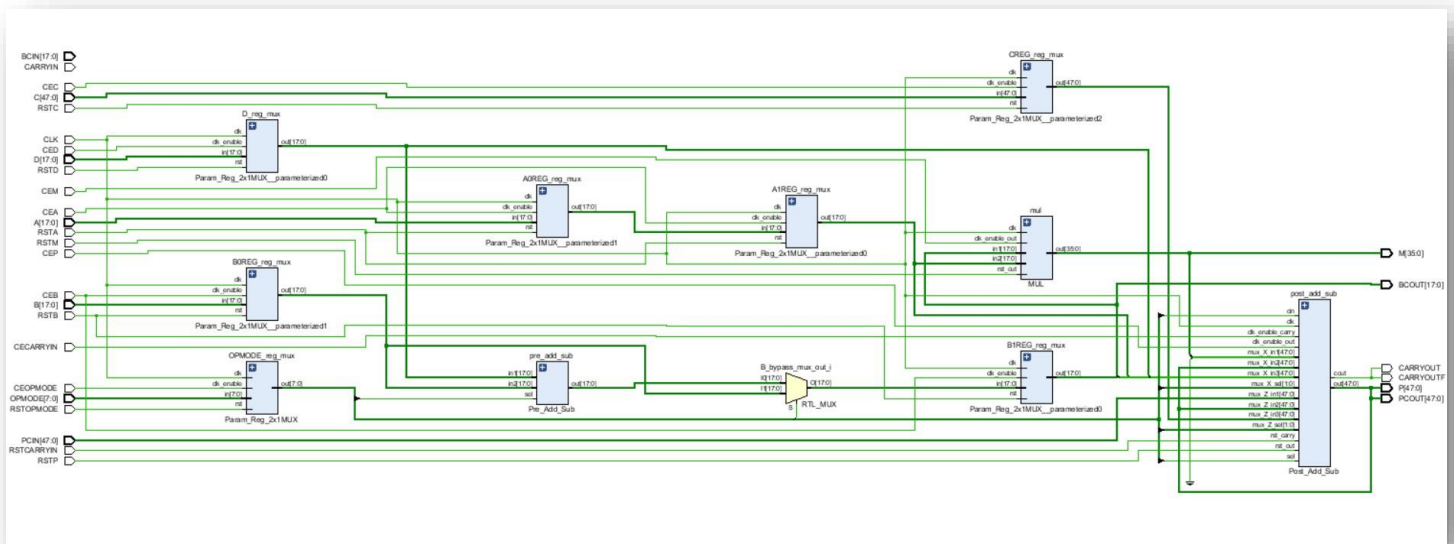
Elaborated Design (2 infos)

General Messages (2 infos)

[Project 1-570] Preparing netlist for logic optimization

[Opt 31-138] Pushed 0 inverter(s) to 0 load pin(s).

### Schematic:



## 7. Synthesis

### Message Tab:

Synthesis (4 warnings)

[Synth 8-3331] design Param\_Reg\_2x1MUX\_\_parameterized1 has unconnected port clk (2 more like this)

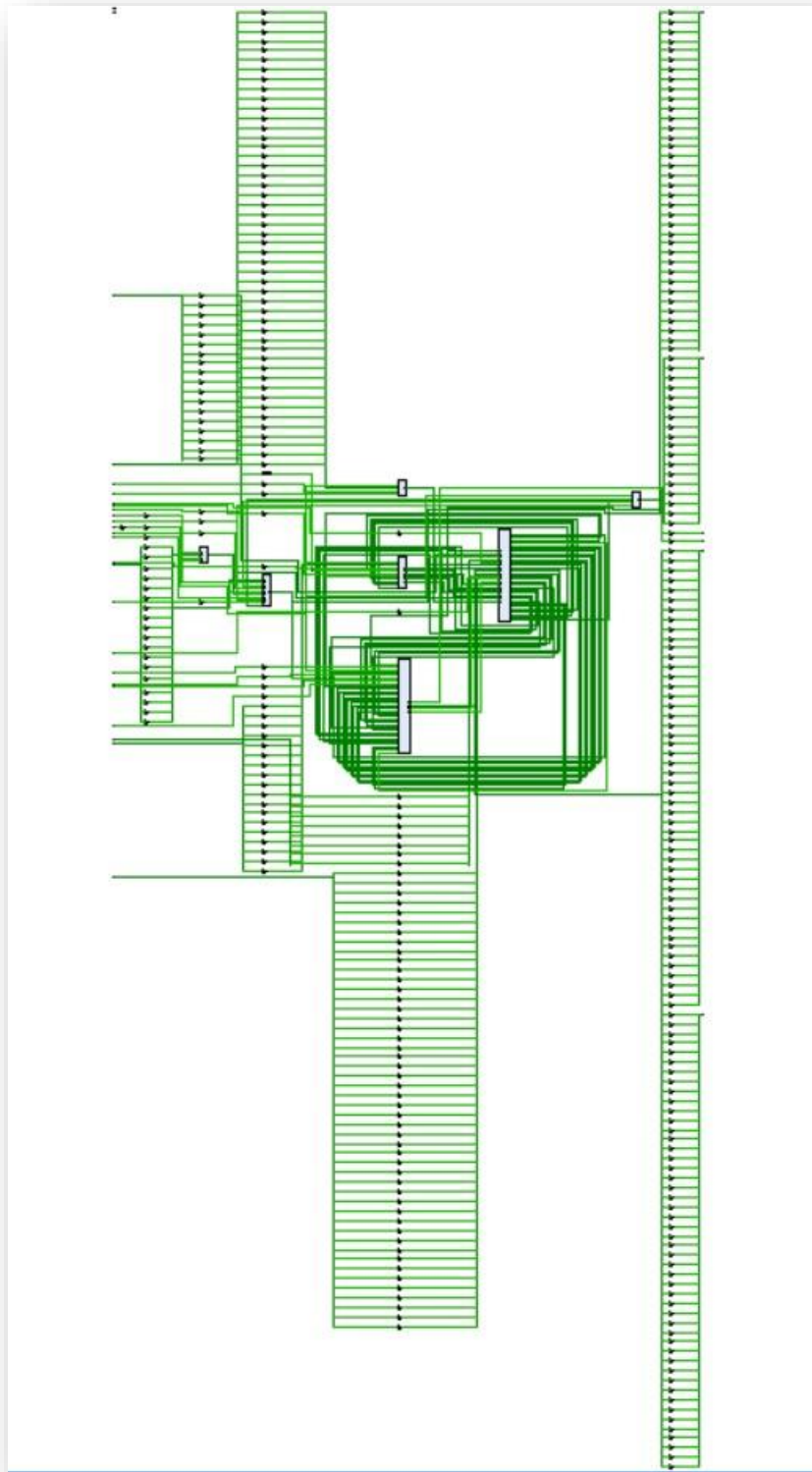
[Synth 8-3331] design Param\_Reg\_2x1MUX\_\_parameterized1 has unconnected port clk\_enable

[Synth 8-3331] design Param\_Reg\_2x1MUX\_\_parameterized1 has unconnected port rst

[Constraints 18-5210] No constraint will be written out.



## Schematic:



# Timing Report:

## Design Timing Summary

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 5.168 ns	Worst Hold Slack (WHS): 0.182 ns	Worst Pulse Width Slack (WPWS): 4.500 ns
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0
Total Number of Endpoints: 106	Total Number of Endpoints: 106	Total Number of Endpoints: 162

All user specified timing constraints are met.

# Utilization Report:

Name	Slice LUTs (134600)	Slice Registers (269200)	DSP s (740 )	Bonded IOB (500)	BUFGCTRL (32)
▼ N Spartan6_DSP48A1	230	160	1	327	1
A1REG_reg_mux (Par...	0	18	0	0	0
B1REG_reg_mux (Par...	0	18	0	0	0
CREG_reg_mux (Para...	0	48	0	0	0
D_reg_mux (Param_R...	0	18	0	0	0
> mul (MUL)	0	0	1	0	0
OPMODE_reg_mux (P...	227	8	0	0	0
> post_add_sub (Post_A...	2	50	0	0	0

## 8. Implementation

### Message Tab:

Implementation (1 warning)

Route Design (1 warning)

DRC (1 warning)

Pin Planning (1 warning)

[DRC CFGBVS-7] CONFIG\_VOLTAGE with Config Bank VCC0: The CONFIG\_MODE property of current\_design specifies a configuration mode (SPIx4) that uses pins in bank 14. I/O standards used in this bank have a voltage requirement of 1.80. However, the CONFIG\_VOLTAGE for current\_design is set to 3.3. Ensure that your configuration voltage is compatible with the I/O standards in banks used by your configuration mode. Refer to device configuration user guide for more information. Pins used by config mode: V28 (IO\_L1P\_T0\_D00\_MOSI\_14), V29 (IO\_L1N\_T0\_D01\_DIN\_14), V26 (IO\_L2P\_T0\_D02\_14), V27 (IO\_L2N\_T0\_D03\_14), W26 (IO\_L3P\_T0\_DQS\_PUDC\_B\_14), and Y27 (IO\_L6P\_T0\_FCS\_B\_14)

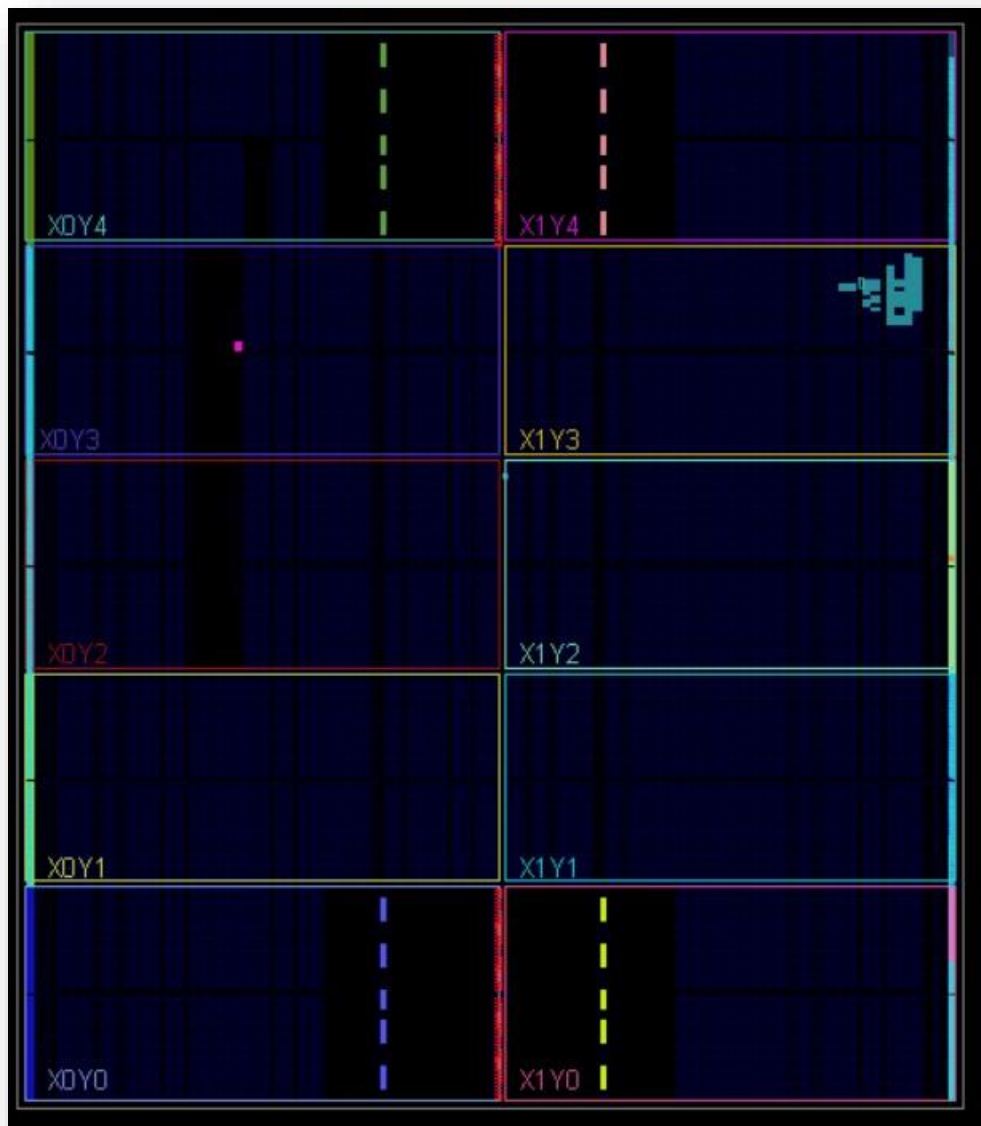
Timing Report:

Design Timing Summary					
Setup		Hold		Pulse Width	
Worst Negative Slack (WNS): 3.678 ns		Worst Hold Slack (WHS): 0.261 ns		Worst Pulse Width Slack (WPWS): 4.500 ns	
Total Negative Slack (TNS): 0.000 ns		Total Hold Slack (THS): 0.000 ns		Total Pulse Width Negative Slack (TPWS): 0.000 ns	
Number of Failing Endpoints: 0		Number of Failing Endpoints: 0		Number of Failing Endpoints: 0	
Total Number of Endpoints: 125		Total Number of Endpoints: 125		Total Number of Endpoints: 181	
All user specified timing constraints are met.					

Utilization Report:

Name	Slice LUTs (133800)	Slice Registers (267600)	Slice (33450)	LUT as Logic (133800)	LUT Flip Flop Pairs (133800)	DSP s (740)	Bonded IOB (500)	BUFGCTRL (32)
▼ N Spartan6_DSP48A1	229	179	105	229	50	1	327	1
A1REG_reg_mux (Par...	0	18	8	0	0	0	0	0
B1REG_reg_mux (Par...	0	36	11	0	0	0	0	0
CREG_reg_mux (Para...	0	48	14	0	0	0	0	0
D_reg_mux (Param_R...	0	18	10	0	0	0	0	0
> mul (MUL)	0	0	0	0	0	1	0	0
OPMODE_reg_mux (P...	227	8	71	227	0	0	0	0
> post_add_sub (Post_A...	2	51	16	2	1	0	0	0

## Device Snippet:



## 9. Linting:

Lint Summary	
(Type Search Text (Press Enter))	
Name	Count
Resolved(verified, fixed, waived)	13
Info	13
parameter_name_duplicate	5
re_entrant_output	1
line_char_large	7