EN3021 Individual Project - Progress as of OCT17

210609M   Silva M.K.Y.U.N.

October 17, 2024

Current Status: 70% complete

Progress this week

Following parts of the project were done for the week OCT10 - OCT17:

- Implemented the Wallace tree design for multiplier
- Integrated multiplier into the floating point ALU

Implementation of the multiplier

As shown in the following design, the layers were implemented. However the design was not tested out yet. With the new addition, the entire circuit (including the adder) takes up 2192 logic elements, and 408 registers.

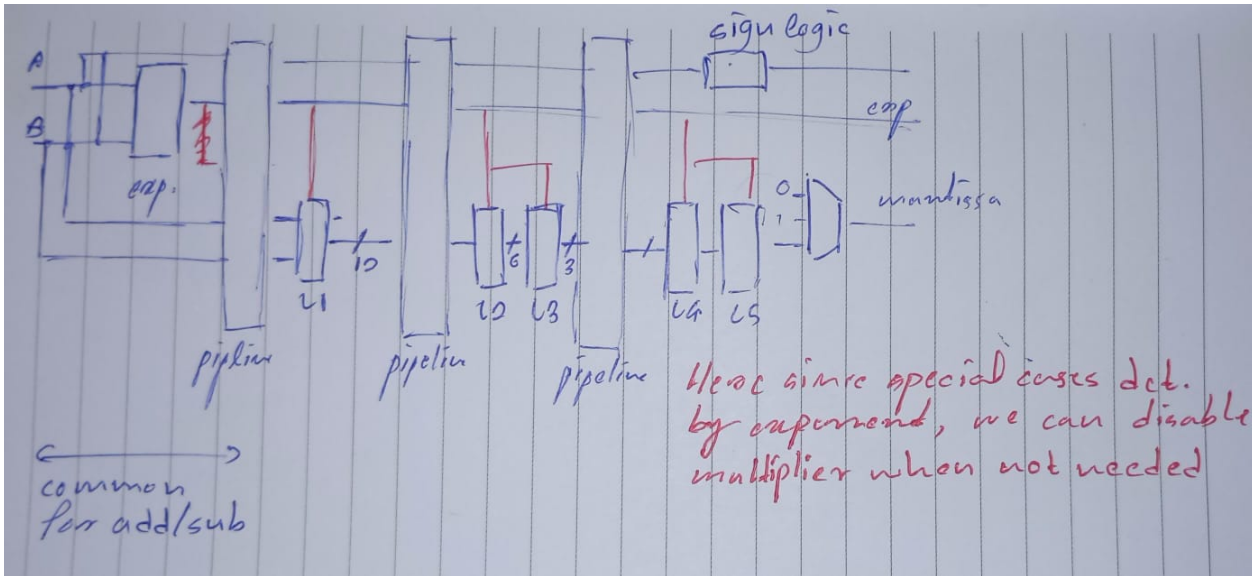

Figure 1: Overall design

Layer 1

The layer 1 contains 36 CLAs and 6 modified half CLAs. These adders are arranged similar to the Wallace tree method, and such that it ensures least number of adders required to do parallel addition.

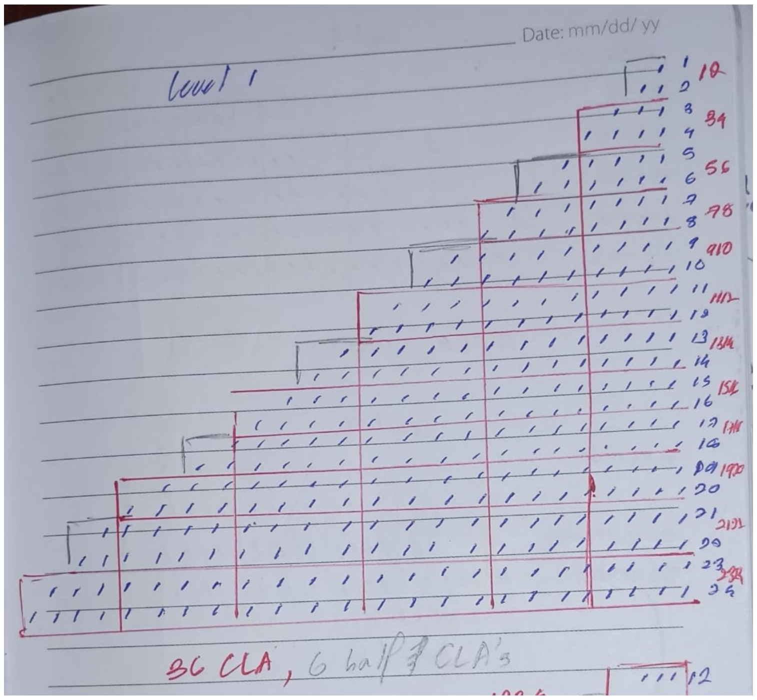

Figure 2: Level 1 - Adding consequently shifted versions where A is shifted left each time, depending on the corresponding bit of B

```
module mul_l1(
    input  [23:0] A, B,
    output [2:0]  OUT12,
    output [4:0]  OUT34,
    output [6:0]  OUT56,
    output [8:0]  OUT78,
    output [10:0] OUT910,
    output [12:0] OUT1112,
    output [14:0] OUT1314,
    output [16:0] OUT1516,
    output [18:0] OUT1718,
    output [20:0] OUT1920,
    output [22:0] OUT2122,
    output [24:0] OUT2324
  );

  wire W1;
  wire [1:0] W2;
  wire [2:0] W3;
  wire [3:0] W4;
  wire [4:0] W5;
  wire [5:0] W6;
  wire [6:0] W7;
  wire [7:0] W8;
  wire [8:0] W9;
  wire [9:0] W10;
  wire [10:0] W11;
  wire [11:0] W12;
  wire [12:0] W13;
  wire [13:0] W14;
  wire [14:0] W15;
  wire [15:0] W16;
  wire [16:0] W17;
  wire [17:0] W18;
  wire [18:0] W19;
  wire [19:0] W20;
  wire [20:0] W21;
  wire [21:0] W22;
  wire [22:0] W23;
  wire [23:0] W24;

  assign W1  = (B[0]  == 1'b1) ? A[23]    : 1'b0;
  assign W2  = (B[1]  == 1'b1) ? A[23:22] : 2'b0;
  assign W3  = (B[2]  == 1'b1) ? A[23:21] : 3'b0;
  assign W4  = (B[3]  == 1'b1) ? A[23:20] : 4'b0;
  assign W5  = (B[4]  == 1'b1) ? A[23:19] : 5'b0;
  assign W6  = (B[5]  == 1'b1) ? A[23:18] : 6'b0;
  assign W7  = (B[6]  == 1'b1) ? A[23:17] : 7'b0;
  assign W8  = (B[7]  == 1'b1) ? A[23:16] : 8'b0;
  assign W9  = (B[8]  == 1'b1) ? A[23:15] : 9'b0;
  assign W10 = (B[9]  == 1'b1) ? A[23:14] : 10'b0;
  assign W11 = (B[10] == 1'b1) ? A[23:13] : 11'b0;
  assign W12 = (B[11] == 1'b1) ? A[23:12] : 12'b0;
  assign W13 = (B[12] == 1'b1) ? A[23:11] : 13'b0;
  assign W14 = (B[13] == 1'b1) ? A[23:10] : 14'b0;
  assign W15 = (B[14] == 1'b1) ? A[23:9]  : 15'b0;
  assign W16 = (B[15] == 1'b1) ? A[23:8]  : 16'b0;
  assign W17 = (B[16] == 1'b1) ? A[23:7]  : 17'b0;
  assign W18 = (B[17] == 1'b1) ? A[23:6]  : 18'b0;
  assign W19 = (B[18] == 1'b1) ? A[23:5]  : 19'b0;
  assign W20 = (B[19] == 1'b1) ? A[23:4]  : 20'b0;
  assign W21 = (B[20] == 1'b1) ? A[23:3]  : 21'b0;
  assign W22 = (B[21] == 1'b1) ? A[23:2]  : 22'b0;
  assign W23 = (B[22] == 1'b1) ? A[23:1]  : 23'b0;
  assign W24 = (B[23] == 1'b1) ? A[23:0]  : 24'b0;

  // line 1, 2
  hcla l1_00(.A({1'b0,W1}), .B(W2), .CIN(1'b0), .COUT(OUT12[2]), .S(OUT12[1:0]));

  // line 3, 4
  cla  l1_10(.A({1'b0,W3}), .B(W4), .CIN(1'b0), .COUT(OUT34[4]), .S(OUT34[3:0]));

  // line 5, 6
  wire CARRY56;
  cla  l1_20(.A(W5[3:0]), .B(W6[3:0]), .CIN(1'b0), .COUT(CARRY56), .S(OUT56[3:0]));
  hcla l1_21(.A({1'b0,W5[4]}), .B(W6[5:4]), .CIN(CARRY56), .COUT(OUT56[6]), .S(OUT56[5:4]));

  // line 7, 8
  wire CARRY78;
  cla  l1_30(.A(W7[3:0]), .B(W8[3:0]), .CIN(1'b0), .COUT(CARRY78), .S(OUT78[3:0]));
  cla  l1_31(.A({1'b0,W7[6:4]}), .B(W8[7:4]), .CIN(CARRY78), .COUT(OUT78[8]), .S(OUT78[7:4])
    );

  // line 9, 10
  wire CARRY910_0, CARRY910_1;
  cla  l1_40(.A(W9[3:0]), .B(W10[3:0]), .CIN(1'b0), .COUT(CARRY910_0), .S(OUT910[3:0]));
  cla  l1_41(.A(W9[7:4]), .B(W10[7:4]), .CIN(CARRY910_0), .COUT(CARRY910_1), .S(OUT910[7:4])
    );
  hcla l1_42(.A({1'b0,W9[8]}), .B(W10[9:8]), .CIN(CARRY910_1), .COUT(OUT910[10]), .S(OUT910
    [9:8]));

  // line 11, 12
  wire CARRY1112_0, CARRY1112_1;
  cla  l1_50(.A(W11[3:0]), .B(W12[3:0]), .CIN(1'b0), .COUT(CARRY1112_0), .S(OUT1112[3:0]));
  cla  l1_51(.A(W11[7:4]), .B(W12[7:4]), .CIN(CARRY1112_0), .COUT(CARRY1112_1), .S(OUT1112[7:4]));
  cla  l1_52(.A({1'b0,W11[10:0]}), .B(W12[11:8]), .CIN(CARRY1112_1), .COUT(OUT1112[12]), .S(OUT1112[11:8]));

  // line 13,14
  wire CARRY1314_0, CARRY1314_1, CARRY1314_2;
  cla  l1_60(.A(W13[3:0]), .B(W14[3:0]), .CIN(1'b0), .COUT(CARRY1314_0), .S(OUT1314[3:0]));
  cla  l1_61(.A(W13[7:4]), .B(W14[7:4]), .CIN(CARRY1314_0), .COUT(CARRY1314_1), .S(OUT1314[7:4]));
  cla  l1_62(.A(W13[11:8]), .B(W14[11:8]), .CIN(CARRY1314_1), .COUT(CARRY1314_2), .S(OUT1314[11:8]));
  hcla l1_63(.A({1'b0,W13[12]}), .B(W14[13:12]), .CIN(CARRY1314_2), .COUT(OUT1314[14]), .S(OUT1314[13:12]));

  // line 15, 16
  wire CARRY1516_0, CARRY1516_1, CARRY1516_2;
  cla  l1_70(.A(W15[3:0]), .B(W16[3:0]), .CIN(1'b0), .COUT(CARRY1516_0), .S(OUT1516[3:0]));
  cla  l1_71(.A(W15[7:4]), .B(W16[7:4]), .CIN(CARRY1516_0), .COUT(CARRY1516_1), .S(OUT1516[7:4]));
  cla  l1_72(.A(W15[11:8]), .B(W16[11:8]), .CIN(CARRY1516_1), .COUT(CARRY1516_2), .S(OUT1516[11:8]));
  cla  l1_73(.A({1'b0,W15[14:12]}), .B(W16[15:12]), .CIN(CARRY1516_2), .COUT(OUT1516[16]), .S(OUT1516[15:12]));

  // line 17, 18
  wire CARRY1718_0, CARRY1718_1, CARRY1718_2, CARRY1718_3;
  cla  l1_80(.A(W17[3:0]), .B(W18[3:0]), .CIN(1'b0), .COUT(CARRY1718_0), .S(OUT1718[3:0]));
  cla  l1_81(.A(W17[7:4]), .B(W18[7:4]), .CIN(CARRY1718_0), .COUT(CARRY1718_1), .S(OUT1718[7:4]));
  cla  l1_82(.A(W17[11:8]), .B(W18[11:8]), .CIN(CARRY1718_1), .COUT(CARRY1718_2), .S(OUT1718[11:8]));
  cla  l1_83(.A(W17[15:12]), .B(W18[15:12]), .CIN(CARRY1718_2), .COUT(CARRY1718_3), .S(OUT1718[15:12]));
  hcla l1_84(.A({1'b0,W17[16]}), .B(W18[17:16]), .CIN(CARRY1718_3), .COUT(OUT1718[18]), .S(OUT1718[17:16]));

  // line 19, 20
  wire CARRY1920_0, CARRY1920_1, CARRY1920_2, CARRY1920_3;
  cla  l1_90(.A(W19[3:0]), .B(W20[3:0]), .CIN(1'b0), .COUT(CARRY1920_0), .S(OUT1920[3:0]));
  cla  l1_91(.A(W19[7:4]), .B(W20[7:4]), .CIN(CARRY1920_0), .COUT(CARRY1920_1), .S(OUT1920[7:4]));
  cla  l1_92(.A(W19[11:8]), .B(W20[11:8]), .CIN(CARRY1920_1), .COUT(CARRY1920_2), .S(OUT1920[11:8]));
  cla  l1_93(.A(W19[15:12]), .B(W20[15:12]), .CIN(CARRY1920_2), .COUT(CARRY1920_3), .S(OUT1920[15:12]));
  cla  l1_94(.A({1'b0,W19[18:16]}), .B(W20[19:16]), .CIN(CARRY1920_3), .COUT(OUT1920[20]), .S(OUT1920[19:16]));

  // line 21, 22
  wire CARRY2122_0, CARRY2122_1, CARRY2122_2, CARRY2122_3, CARRY2122_4;
  cla  l1_100(.A(W21[3:0]), .B(W22[3:0]), .CIN(1'b0), .COUT(CARRY2122_0), .S(OUT2122[3:0]));
  cla  l1_101(.A(W21[7:4]), .B(W22[7:4]), .CIN(CARRY2122_0), .COUT(CARRY2122_1), .S(OUT2122[7:4]));
  cla  l1_102(.A(W21[11:8]), .B(W22[11:8]), .CIN(CARRY2122_1), .COUT(CARRY2122_2), .S(OUT2122[11:8]));
  cla  l1_103(.A(W21[15:12]), .B(W22[15:12]), .CIN(CARRY2122_2), .COUT(CARRY2122_3), .S(OUT2122[15:12]));
  cla  l1_104(.A(W21[19:16]), .B(W22[19:16]), .CIN(CARRY2122_3), .COUT(CARRY2122_4), .S(OUT2122[19:16]));
  hcla l1_105(.A({1'b0,W21[20]}), .B(W22[21:20]), .CIN(CARRY2122_4), .COUT(OUT2122[22]), .S(OUT2122[21:20]));

  wire CARRY2324_0, CARRY2324_1, CARRY2324_2, CARRY2324_3, CARRY2324_4;
  cla  l1_110(.A(W23[3:0]), .B(W24[3:0]), .CIN(1'b0), .COUT(CARRY2324_0), .S(OUT2324[3:0]));
  cla  l1_111(.A(W23[7:4]), .B(W24[7:4]), .CIN(CARRY2324_0), .COUT(CARRY2324_1), .S(OUT2324[7:4]));
```

```
  cla  l1_112(.A(W23[11:8]), .B(W24[11:8]), .CIN(CARRY2324_1), .COUT(CARRY2324_2), .S(OUT2324[11:8]));
  cla  l1_113(.A(W23[15:12]), .B(W24[15:12]), .CIN(CARRY2324_2), .COUT(CARRY2324_3), .S(OUT2324[15:12]));
  cla  l1_114(.A(W23[19:16]), .B(W24[19:16]), .CIN(CARRY2324_3), .COUT(CARRY2324_4), .S(OUT2324[19:16]));
  cla  l1_115(.A({1'b0,W23[22:20]}), .B(W24[23:20]), .CIN(CARRY2324_4), .COUT(OUT2324[24]), .S(OUT2324[23:20]));
endmodule
```

Layer 2

In layer 2 there are 21 CLAs and 5 half adders. This is implemented to add the results of layer 1.

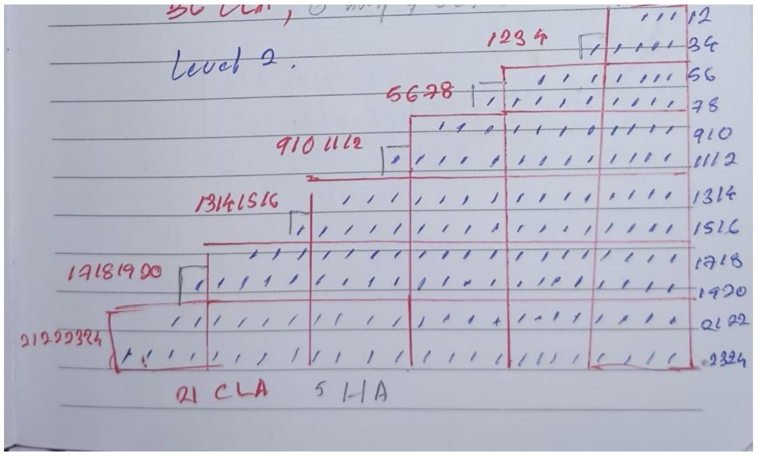

Figure 3: Level 2 - Adding consequently shifted versions of level 1 results

```
module mul_l2(
    input  [2:0]  W12,
    input  [4:0]  W34,
    input  [6:0]  W56,
    input  [8:0]  W78,
    input  [10:0] W910,
    input  [12:0] W1112,
    input  [14:0] W1314,
    input  [16:0] W1516,
    input  [18:0] W1718,
     input  [20:0] W1920,
    input  [22:0] W2122,
    input  [24:0] W2324,
    output [5:0]  OUT1234,
    output [9:0]  OUT5678,
    output [13:0] OUT9101112,
    output [17:0] OUT13141516,
    output [21:0] OUT17181920,
    output [24:0] OUT21222324
  );

  // line 12, 34
  wire CARRY1234_0;
  cla l2_00(.A({1'b0,W12}), .B(W34[3:0]), .CIN(1'b0), .COUT(CARRY1234_0), .S(OUT1234[3:0]));
  ha  l2_01(.A(W34[4]), .B(CARRY1234_0), .COUT(OUT1234[5]), .SUM(OUT1234[4]));

  // line 56, 78
  wire CARRY5678_0, CARRY5678_1;
  cla l2_10(.A(W56[3:0]), .B(W78[3:0]), .CIN(1'b0), .COUT(CARRY5678_0), .S(OUT5678[3:0]));
  cla l2_11(.A({1'b0,W56[6:4]}), .B(W78[7:4]), .CIN(CARRY5678_0), .COUT(CARRY5678_1), .S(OUT5678[7:4]));
  ha  l2_12(.A(W78[8]), .B(CARRY5678_1), .COUT(OUT5678[9]), .SUM(OUT5678[8]));

  // line 910, 1112
  wire CARRY9101112_0, CARRY9101112_1, CARRY9101112_2;
  cla l2_20(.A(W910[3:0]), .B(W1112[3:0]), .CIN(1'b0), .COUT(CARRY9101112_0), .S(OUT9101112[3:0]));
  cla l2_21(.A(W910[7:4]), .B(W1112[7:4]), .CIN(CARRY9101112_0), .COUT(CARRY9101112_1), .S(OUT9101112[7:4]));
  cla l2_22(.A({1'b0,W910[10:8]}), .B(W1112[11:8]), .CIN(CARRY9101112_1), .COUT(CARRY9101112_2), .S(OUT9101112[11:8]));
  ha  l2_23(.A(W1112[12]), .B(CARRY9101112_2), .COUT(OUT9101112[13]), .SUM(OUT9101112[12]));

  // LINE 1314, 1516
  wire CARRY13141516_0, CARRY13141516_1, CARRY13141516_2, CARRY13141516_3;
  cla l2_30(.A(W1314[3:0]), .B(W1516[3:0]), .CIN(1'b0), .COUT(CARRY13141516_0), .S(OUT13141516[3:0]));
  cla l2_31(.A(W1314[7:4]), .B(W1516[7:4]), .CIN(CARRY13141516_0), .COUT(CARRY13141516_1), .S(OUT13141516[7:4]));
  cla l2_32(.A(W1314[11:8]), .B(W1516[11:8]), .CIN(CARRY13141516_1), .COUT(CARRY13141516_2), .S(OUT13141516[11:8]));
  cla l2_33(.A({1'b0,W1314[14:12]}), .B(W1516[15:12]), .CIN(CARRY13141516_2), .COUT(CARRY13141516_3), .S(OUT13141516[15:12]));
  ha  l2_34(.A(W1516[16]), .B(CARRY13141516_3), .COUT(OUT13141516[17]), .SUM(OUT13141516[16]));

  // line 1718, 1920
  wire CARRY17181920_0, CARRY17181920_1, CARRY17181920_2, CARRY17181920_3, CARRY17181920_4;
  cla l2_40(.A(W1718[3:0]), .B(W1920[3:0]), .CIN(1'b0), .COUT(CARRY17181920_0), .S(OUT17181920[3:0]));
  cla l2_41(.A(W1718[7:4]), .B(W1920[7:4]), .CIN(CARRY17181920_0), .COUT(CARRY17181920_1), .S(OUT17181920[7:4]));
  cla l2_42(.A(W1718[11:8]), .B(W1920[11:8]), .CIN(CARRY17181920_1), .COUT(CARRY17181920_2), .S(OUT17181920[11:8]));
  cla l2_43(.A(W1718[15:12]), .B(W1920[15:12]), .CIN(CARRY17181920_2), .COUT(CARRY17181920_3), .S(OUT17181920[15:12]));
  cla l2_44(.A({1'b0,W1718[18:16]}), .B(W1920[19:16]), .CIN(CARRY17181920_3), .COUT(CARRY17181920_4), .S(OUT17181920[19:16]));
  ha  l2_45(.A(W1920[20]), .B(CARRY17181920_4), .COUT(OUT17181920[21]), .SUM(OUT17181920[20]));

  // line 2122, 2324
  wire CARRY21222324_0, CARRY21222324_1, CARRY21222324_2, CARRY21222324_3, CARRY21222324_4, CARRY21222324_5;
  cla l2_50(.A(W2122[3:0]), .B(W2324[3:0]), .CIN(1'b0), .COUT(CARRY21222324_0), .S(OUT21222324[3:0]));
  cla l2_51(.A(W2122[7:4]), .B(W2324[7:4]), .CIN(CARRY21222324_0), .COUT(CARRY21222324_1), .S(OUT21222324[7:4]));
  cla l2_52(.A(W2122[11:8]), .B(W2324[11:8]), .CIN(CARRY21222324_1), .COUT(CARRY21222324_2), .S(OUT21222324[11:8]));
  cla l2_53(.A(W2122[15:12]), .B(W2324[15:12]), .CIN(CARRY21222324_2), .COUT(CARRY21222324_3), .S(OUT21222324[15:12]));
  cla l2_54(.A(W2122[19:16]), .B(W2324[19:16]), .CIN(CARRY21222324_3), .COUT(CARRY21222324_4), .S(OUT21222324[19:16]));
  cla l2_55(.A({1'b0,W2122[22:20]}), .B(W2324[23:20]), .CIN(CARRY21222324_4), .COUT(CARRY21222324_5), .S(OUT21222324[23:20]));
  ha  l2_56(.A(W2324[24]), .B(CARRY21222324_5), .COUT(), .SUM(OUT21222324[24]));
endmodule
```

Layer 3

In layer 3 there are 12 CLAs and 2 modified half CLAs. This is implemented to add the results of layer 2.

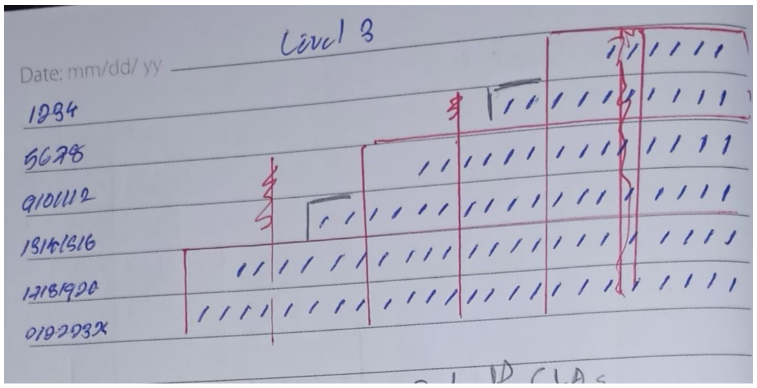

Figure 4: Level 3 - Adding consequently shifted versions of level 2 results

```verilog
module mul_l3(
    input  [5:0]  W1234,
    input  [9:0]  W5678,
    input  [13:0] W9101112,
    input  [17:0] W13141516,
    input  [21:0] W17181920,
    input  [24:0] W21222324,
    output [10:0] OUT12345678,
    output [18:0] OUT910111213141516,
    output [24:0] OUT1718192021222324
  );

  // LINE 1234, 5678
  wire CARRY12345678_0, CARRY12345678_1;
  cla  l3_00(.A(W1234[3:0]), .B(W5678[3:0]), .CIN(1'b0), .COUT(CARRY12345678_0), .S(OUT12345678[3:0]));
  cla  l3_01(.A({2'b0,W1234[5:4]}), .B(W5678[7:4]), .CIN(CARRY12345678_0), .COUT(CARRY1234578_1), .S(OUT12345678[7:4]));
  tcla l3_02(.A(W5678[9:8]), .CIN(CARRY12345678_1), .COUT(OUT12345678[10]), .S(OUT12345678[9:8]));

  // line 9101112, 13141516
  wire CARRY910111213141516_0, CARRY910111213141516_1, CARRY910111213141516_2, CARRY910111213141516_3;
  cla  l3_10(.A(W9101112[3:0]), .B(W13141516[3:0]), .CIN(1'b0), .COUT(CARRY910111213141516_0), .S(OUT910111213141516[3:0]));
  cla  l3_11(.A(W9101112[7:4]), .B(W13141516[7:4]), .CIN(CARRY910111213141516_0), .COUT(CARRY910111213141516_1), .S(OUT910111213141516[7:4]));
  cla  l3_12(.A(W9101112[11:8]), .B(W13141516[11:8]), .CIN(CARRY910111213141516_1), .COUT(CARRY910111213141516_2), .S(OUT910111213141516[11:8]));
  cla  l3_13(.A({2'b0,W9101112[13:12]}), .B(W13141516[15:12]), .CIN(CARRY910111213141516_2), .COUT(CARRY910111213141516_3), .S(OUT910111213141516[15:12]));
  tcla l3_14(.A(W13141516[17:16]), .CIN(CARRY910111213141516_3), .COUT(OUT910111213141516[18]), .S(OUT910111213141516[17:16]));

  // line 17181920, 21222324
  wire CARRY1718192021222324_0, CARRY1718192021222324_1, CARRY1718192021222324_2, CARRY1718192021222324_3, CARRY1718192021222324_4, CARRY1718192021222324_5;
  cla l3_20(.A(W17181920[3:0]), .B(W21222324[3:0]), .CIN(1'b0), .COUT(CARRY1718192021222324_0), .S(OUT1718192021222324[3:0]));
  cla l3_21(.A(W17181920[7:4]), .B(W21222324[7:4]), .CIN(CARRY1718192021222324_0), .COUT(CARRY1718192021222324_1), .S(OUT1718192021222324[7:4]));
  cla l3_22(.A(W17181920[11:8]), .B(W21222324[11:8]), .CIN(CARRY1718192021222324_1), .COUT(CARRY1718192021222324_2), .S(OUT1718192021222324[11:8]));
  cla l3_23(.A(W17181920[15:12]), .B(W21222324[15:12]), .CIN(CARRY1718192021222324_2), .COUT(CARRY1718192021222324_3), .S(OUT1718192021222324[15:12]));
  cla l3_24(.A(W17181920[19:16]), .B(W21222324[19:16]), .CIN(CARRY1718192021222324_3), .COUT(CARRY1718192021222324_4), .S(OUT1718192021222324[19:16]));
  cla l3_25(.A({2'b0,W17181920[21:20]}), .B(W21222324[23:20]), .CIN(CARRY1718192021222324_4)
    , .COUT(CARRY1718192021222324_5), .S(OUT1718192021222324[23:20]));
  ha  l3_26(.A(W21222324[24]), .B(CARRY1718192021222324_5), .COUT(), .SUM(
    OUT1718192021222324[24]));

endmodule
```

Layer4

In layer 4 there are 3 CLAs and 2 modified half CLAs. This is implemented to add the results of layer 3. Here however there's 3 results from layer 3, one of these results is passed straight through to layer 5 to be added with the result of the addition of the other two results.

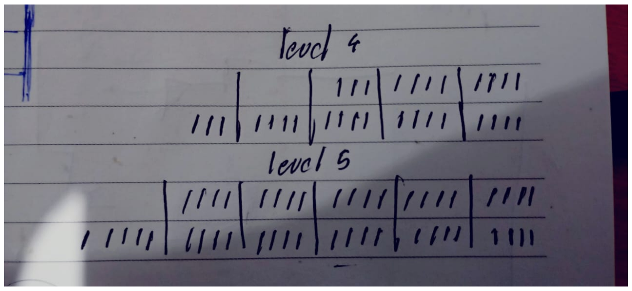

Figure 5: Level 4 and level 5

```
module mul_l4(
    input  [10:0] W12345678,
    input  [18:0] W910111213141516,
    input  [24:0] W1718192021222324,
    output [19:0] OUT12345678910111213141516,
    output [24:0] OUT1718192021222324
  );

  // line 17181920, 21222324
  wire CARRY12345678910111213141516_0, CARRY12345678910111213141516_1,
    CARRY12345678910111213141516_2, CARRY12345678910111213141516_3;
  cla  l4_00(.A(W12345678[3:0]), .B(W910111213141516[3:0]), .CIN(1'b0), .COUT(
    CARRY12345678910111213141516_0), .S(OUT12345678910111213141516[3:0]));
  cla  l4_01(.A(W12345678[7:4]), .B(W910111213141516[7:4]), .CIN(
    CARRY12345678910111213141516_0), .COUT(CARRY12345678910111213141516_1), .S(
    OUT12345678910111213141516[7:4]));
  cla  l4_02(.A({1'b0,W12345678[10:8]}), .B(W910111213141516[11:8]), .CIN(
    CARRY12345678910111213141516_1), .COUT(CARRY12345678910111213141516_2), .S(
    OUT12345678910111213141516[11:8]));
  ocla l4_03(.A(W910111213141516[15:12]), .CIN(CARRY12345678910111213141516_2), .COUT(
    CARRY12345678910111213141516_3), .S(OUT12345678910111213141516[15:12]));
  ocla l4_04(.A({1'b0,W910111213141516[18:16]}), .CIN(CARRY12345678910111213141516_3), .COUT
    (), .S(OUT12345678910111213141516[19:16]));

  assign OUT1718192021222324 = W1718192021222324;

endmodule
```

Layer 5

In layer 5 there are 5 CLAs, 1 modified half CLA, and an half adder. This is implemented to add the results of layer 4.

```verilog
module mul_l5(
    input  [19:0] W12345678910111213141516,
    input  [24:0] W1718192021222324,
    output [24:0] OUT
  );

  // line 17181920, 21222324
  wire CARRY_0, CARRY_1, CARRY_2, CARRY_3, CARRY_4, CARRY_5;
  cla  l5_00(.A(W12345678910111213141516[3:0]), .B(W1718192021222324[3:0]), .CIN(1'b0), .COUT(CARRY_0), .S(OUT[3:0]));
  cla  l5_01(.A(W12345678910111213141516[7:4]), .B(W1718192021222324[7:4]), .CIN(CARRY_0), .COUT(CARRY_1), .S(OUT[7:4]));
  cla  l5_02(.A(W12345678910111213141516[11:8]), .B(W1718192021222324[11:8]), .CIN(CARRY_1), .COUT(CARRY_2), .S(OUT[11:8]));
  cla  l5_03(.A(W12345678910111213141516[15:12]), .B(W1718192021222324[15:12]), .CIN(CARRY_2), .COUT(CARRY_3), .S(OUT[15:12]));
  cla  l5_04(.A(W12345678910111213141516[19:16]), .B(W1718192021222324[19:16]), .CIN(CARRY_3), .COUT(CARRY_4), .S(OUT[19:16]));
  ocla l5_05(.A(W1718192021222324[23:20]), .CIN(CARRY_4), .COUT(CARRY_5), .S(OUT[23:20]));
  ha   l5_06(.A(W1718192021222324[24]), .B(CARRY_5), .COUT(), .SUM(OUT[24]));
endmodule
```

Integration into Floating point ALU

For tasks such as exponent calculation and normalisation, the existing modules of the adder pipeline have been used.

```verilog
// IEEE754 Format
//
// +------+-------------+-------------+
// |  31  |   [30:23]   |    [22:0]   |
// | Sign |   Exponent  |   Mantissa  |
// +------+-------------+-------------+


module alu(
    input  FPUCLK,
    input  [31:0] A, B,
    input  [2:0]  CTRL,
    output reg [31:0] OUT
  );


  // Pipeline registers
  reg [23:0] APP0, APP1, BPP0, BPP1, RESM;  // Pipeline registers for number and cmd
  reg [31:0] OUT0;
  reg ASIGN0, ASIGN1, BSIGN0, BSIGN1, SIGN, OF0;
  reg [2:0]  CTRL1, CTRL2, CTRL3;
  reg [7:0] EXPONENT;


  // For exponents
  reg  EXPCIN;        // Add or subtract the exponents
  wire [7:0] EXPCAL_w;    // Result of exponents (for pipeline stage1)
  wire COUT_EX;
  reg [7:0]  EXPCARRY0, EXPCARRY1;            // Exp of greater to be carried
  reg  EXPZFLAG0, EXPZFLAG1, EXPZFLAG2;

  cla claexp_0(.A(A[26:23]), .B(B[26:23]^{4{EXPCIN}}), .CIN(EXPCIN),  .COUT(COUT_EX), .S(EXPCAL_w[3:0]));
  cla claexp_1(.A(A[30:27]),  .B(B[30:27]^{4{EXPCIN}}), .CIN(COUT_EX), .COUT(),        .S(EXPCAL_w[7:4]));



  reg [7:0] EXPCAL0;
```

```

// For shifter
reg [23:0] TBSHIFTED;
wire [23:0] SHIFTOUT_w;

shifter shift_0(.IN(TBSHIFTED), .BY(EXPCAL0), .OUT(SHIFTOUT_w));

// For operations
reg  [23:0] A_ALUIN, B_ALUIN;
wire [23:0] ALUOUT;
reg ALUCIN;
wire OVERFLOW;

cla_add cla_add0(
  .A(A_ALUIN),
  .B(B_ALUIN),
  .CIN(ALUCIN),
  .OUT(ALUOUT),
  .OF(OVERFLOW)
);

// For sign resolution
wire [23:0] TWOSOUT;

twoscomp twosconv0(
  .B(ALUOUT),
  .OUT(TWOSOUT)
);

// Normalising
wire [22:0] NORMOUT;
wire NORMFLAG, NORMZERO;
wire [7:0] NORM_SHIFT;

normal normalize0(
  .IN(RESM),
  .INOF(OF0),
  .OUT(NORMOUT),
  .COUNT(NORM_SHIFT),
  .ZEROFLAG(NORMZERO)
);

// Signres
wire COUT_SRES;
wire [7:0] EXPSRES;

cla clasresexp_0(.A(EXPCARRY1[3:0]),  .B(NORM_SHIFT[3:0]), .CIN(1'b0),      .COUT(
  COUT_SRES), .S(EXPSRES[3:0]));
cla clasresexp_1(.A(EXPCARRY1[7:4]),  .B(NORM_SHIFT[7:4]), .CIN(COUT_SRES), .COUT(),
      .S(EXPSRES[7:4]));


// multipliers
// LAYER 1
wire [2:0]  WML1_12;
wire [4:0]  WML1_34;
wire [6:0]  WML1_56;
wire [8:0]  WML1_78;
wire [10:0] WML1_910;
wire [12:0] WML1_1112;
wire [14:0] WML1_1314;
wire [16:0] WML1_1516;
wire [18:0] WML1_1718;
wire [20:0] WML1_1920;
wire [22:0] WML1_2122;
wire [24:0] WML1_2324;

reg [2:0]  RML1_12;
```

```verilog
  reg [4:0]  RML1_34;
  reg [6:0]  RML1_56;
  reg [8:0]  RML1_78;
  reg [10:0] RML1_910;
  reg [12:0] RML1_1112;
  reg [14:0] RML1_1314;
  reg [16:0] RML1_1516;
  reg [18:0] RML1_1718;
  reg [20:0] RML1_1920;
  reg [22:0] RML1_2122;
  reg [24:0] RML1_2324;

  mul_l1 l1(.A(APP0), .B(BPP0), .OUT12(WML1_12), .OUT34(WML1_34), .OUT56(WML1_56), .OUT78(WML1_78), .OUT910(WML1_910), .OUT1112(WML1_1112),
         .OUT1314(WML1_1314), .OUT1516(WML1_1516), .OUT1718(WML1_1718), .OUT1920(WML1_1920), .OUT2122(WML1_2122), .OUT2324(WML1_2324));

  wire [5:0]  WML2_1234;
  wire [9:0]  WML2_5678;
  wire [13:0] WML2_9101112;
  wire [17:0] WML2_13141516;
  wire [21:0] WML2_17181920;
  wire [24:0] WML2_21222324;

  mul_l2 l2(.W12(RML1_12), .W34(RML1_34), .W56(RML1_56), .W78(RML1_78), .W910(RML1_910), .W1112(RML1_1112), .W1314(RML1_1314), .W1516(RML1_1516),
         .W1718(RML1_1718), .W1920(RML1_1920), .W2122(RML1_2122), .W2324(RML1_2324), .OUT1234(WML2_1234), .OUT5678(WML2_5678),
         .OUT9101112(WML2_9101112), .OUT13141516(WML2_13141516), .OUT17181920(WML2_17181920), .OUT21222324(WML2_21222324));

  wire [10:0] WML3_12345678;
  wire [18:0] WML3_910111213141516;
  wire [24:0] WML3_1718192021222324;

  reg [10:0] RML3_12345678;
  reg [18:0] RML3_910111213141516;
  reg [24:0] RML3_1718192021222324;

  mul_l3 l3(.W1234(WML2_1234), .W5678(WML2_5678), .W9101112(WML2_9101112), .W13141516(WML2_13141516), .W17181920(WML2_17181920),
         .W21222324(WML2_21222324), .OUT12345678(WML3_12345678), .OUT910111213141516(WML3_910111213141516),
         .OUT1718192021222324(WML3_1718192021222324));

  wire [19:0] WML4_12345678910111213141516;
  wire [24:0] WML4_1718192021222324;

  mul_l4 l4(.W12345678(RML3_12345678), .W910111213141516(RML3_910111213141516), .W1718192021222324(RML3_1718192021222324),
         .OUT12345678910111213141516(WML4_12345678910111213141516), .OUT1718192021222324(WML4_1718192021222324));

  wire [24:0] WML5_OUT;

  mul_l5 l5(.W12345678910111213141516(WML4_12345678910111213141516), .W1718192021222324(WML4_1718192021222324), .OUT(WML5_OUT));

  // codes
  parameter ADD = 3'b000, SUB = 3'b001, MUL = 3'b010;



  // Combinational parts of each stage

  always@(*) begin
    // ------------------------------------
    // Stage 1 : Exponent calculation
    // -----------------------------------
    if (CTRL == ADD | CTRL == SUB) begin
      EXPCIN <= 1'b1;
    end else begin      // for MUL
      EXPCIN <= 1'b0;
    end

    // -----------------------------------
    // Stage 2 : Shifting
    // -----------------------------------
    if (CTRL1 == ADD | CTRL1 == SUB) begin
      // Deciding which to be shifted
      if (EXPCAL0[7] == 1'b0) begin     // exp(A) > exp(B), shift B to right
        TBSHIFTED <= BPP0;
      end else begin                 // exp(B) > exp(A), shift A to right
        TBSHIFTED <= APP0;
      end
    end else begin
      TBSHIFTED <= 24'bX;
    end


    // -----------------------------------
    // Stage 3 : Operation
    // -----------------------------------
    if (CTRL2 == ADD) begin
      if (((ASIGN1 == 1'b0) & (BSIGN1 == 1'b0)) | ((ASIGN1 == 1'b1) & (BSIGN1 == 1'b1)))
    begin  // if signs are the same
        A_ALUIN <= APP1;
        B_ALUIN <= BPP1;
        ALUCIN  <= 1'b0;
      end else if ((ASIGN1 == 1'b0) & (BSIGN1 == 1'b1)) begin
        A_ALUIN <= APP1;  // (+A) + (-B) = (A-B)
        B_ALUIN <= BPP1;
        ALUCIN  <= 1'b1;
      end else if ((ASIGN1 == 1'b1) & (BSIGN1 == 1'b0)) begin
        A_ALUIN <= BPP1;  // Flipping (-A)+(+B) = (B-A)
        B_ALUIN <= APP1;
        ALUCIN  <= 1'b1;
      end else begin
        A_ALUIN <= 24'bX;
        B_ALUIN <= 24'bX;
        ALUCIN  <= 1'bX;
      end
    end else if (CTRL2 == SUB) begin
      if (((ASIGN1 == 1'b0) & (BSIGN1 == 1'b1)) | ((ASIGN1 == 1'b1) & (BSIGN1 == 1'b0)))
    begin  // if signs are different
        A_ALUIN <= APP1;
        B_ALUIN <= BPP1;
        ALUCIN  <= 1'b0;
      end else if ((ASIGN1 == 1'b0) & (BSIGN1 == 1'b0)) begin
        A_ALUIN <= APP1;  // (+A) - (+B) = (A-B)
        B_ALUIN <= BPP1;
        ALUCIN  <= 1'b1;
      end else if ((ASIGN1 == 1'b1) & (BSIGN1 == 1'b0)) begin
        A_ALUIN <= BPP1;  // Flipping (-A)-(-B) = (B-A)
        B_ALUIN <= APP1;
        ALUCIN  <= 1'b1;
      end else begin
        A_ALUIN <= 24'bX;
        B_ALUIN <= 24'bX;
        ALUCIN  <= 1'bX;
      end
    end else begin
      A_ALUIN <= 24'bX;
      B_ALUIN <= 24'bX;
      ALUCIN  <= 1'bX;
    end

    // ----------------------------------
    // Stage 4 : Sign resolution
    // ----------------------------------
    if ((CTRL3 == ADD)|(CTRL3 == SUB)) begin
      if ((EXPZFLAG2 == 1'b1) & (NORMZERO == 1'b1)) begin
        EXPONENT <= 8'b0;
      end else begin
        EXPONENT <= EXPSRES;
      end
    end else begin
      EXPONENT <= EXPSRES;      // TO BE CHANGED!!!
    end
  end


  // Let ADD be 3'b000
  always@(posedge FPUCLK) begin
    // ----------------------------------
    // Stage 1 : Exponent calculation
    // ----------------------------------

    if (CTRL == ADD | CTRL == SUB | CTRL == MUL) begin
      APP0    <= {1'b1,A[22:0]};
      BPP0    <= {1'b1,B[22:0]};
      ASIGN0  <= A[31];
      BSIGN0  <= B[31];

      EXPCAL0 <= EXPCAL_w;
      CTRL1   <= CTRL;
    end

    if (EXPCAL_w == 8'b0) begin
      EXPZFLAG0 <= 1'b1;
    end else begin
      EXPZFLAG0 <= 1'b0;
    end


    // ----------------------------------
    // Stage 2 : Shifting
    // ----------------------------------
    if (CTRL1 == ADD | CTRL1 == SUB) begin
      EXPZFLAG1 <= EXPZFLAG0;
      CTRL2  <= CTRL1;
      ASIGN1 <= ASIGN0;
      BSIGN1 <= BSIGN0;

      // Deciding which to be shifted
      if (EXPCAL0[7] == 1'b0) begin     // exp(A) > exp(B), shift B to right
        APP1      <= APP0;
        BPP1      <= SHIFTOUT_w;
        EXPCARRY0  <= A[30:23];
      end else begin                   // exp(B) > exp(A), shift A to right
        APP1     <= SHIFTOUT_w;
        BPP1      <= BPP0;
        EXPCARRY0  <= B[30:23];
      end
    end else if (CTRL1 == MUL) begin
      CTRL2  <= CTRL1;
      ASIGN1 <= ASIGN0;
      BSIGN1 <= BSIGN0;
      EXPCARRY0 <= EXPCAL0;

      RML1_12   <= WML1_12;
      RML1_34   <= WML1_34;
      RML1_56   <= WML1_56;
      RML1_78   <= WML1_78;
```

```verilog
      RML1_910  <= WML1_910;
      RML1_1112 <= WML1_1112;
      RML1_1314 <= WML1_1314;
      RML1_1516 <= WML1_1516;
      RML1_1718 <= WML1_1718;
      RML1_1920 <= WML1_1920;
      RML1_2122 <= WML1_2122;
      RML1_2324 <= WML1_2324;
    end


    // ----------------------------------
    // Stage 3 : Operation
    // ----------------------------------
    if (CTRL2 == ADD) begin
      EXPZFLAG2 <= EXPZFLAG1;
      CTRL3 <= CTRL2;

      if (((ASIGN1 == 1'b0) & (BSIGN1 == 1'b0)) | ((ASIGN1 == 1'b1) & (BSIGN1 == 1'b1)))
    begin  // if signs are the same
        RESM <= ALUOUT;
        OF0  <= OVERFLOW;
        EXPCARRY1 <= EXPCARRY0;
        SIGN <= ASIGN1;

      end else if ((ASIGN1 == 1'b0) & (BSIGN1 == 1'b1)) begin

        if (OVERFLOW == 1'b0) begin
          RESM <= TWOSOUT;
        end else begin
          RESM <= ALUOUT;
        end

        OF0 <= 1'b0;
        EXPCARRY1 <= EXPCARRY0;
        SIGN <= ~OVERFLOW;

      end else if ((ASIGN1 == 1'b1) & (BSIGN1 == 1'b0)) begin

        if (OVERFLOW == 1'b0) begin
          RESM <= TWOSOUT;
        end else begin
          RESM <= ALUOUT;
        end

        OF0 <= 1'b0;
        EXPCARRY1 <= EXPCARRY0;
        SIGN <= ~OVERFLOW;
      end

    end else if (CTRL2 == SUB) begin
      CTRL3 <= CTRL2;
      EXPZFLAG2 <= EXPZFLAG1;

      if (((ASIGN1 == 1'b0) & (BSIGN1 == 1'b1)) | ((ASIGN1 == 1'b1) & (BSIGN1 == 1'b0)))
    begin  // if signs are different
        RESM <= ALUOUT;
        OF0 <= OVERFLOW;
        EXPCARRY1 <= EXPCARRY0;
        SIGN <= ASIGN1;

      end else if ((ASIGN1 == 1'b0) & (BSIGN1 == 1'b0)) begin
        if (OVERFLOW == 1'b0) begin
          RESM <= TWOSOUT;
        end else begin
          RESM <= ALUOUT;
        end

        OFO <= 1'b0;
        EXPCARRY1 <= EXPCARRY0;
        SIGN <= ~OVERFLOW;

      end else if ((ASIGN1 == 1'b1) & (BSIGN1 == 1'b0)) begin
        if (OVERFLOW == 1'b0) begin
          RESM <= TWOSOUT;
        end else begin
          RESM <= ALUOUT;
        end

        OFO <= 1'b0;
        EXPCARRY1 <= EXPCARRY0;
        SIGN <= ~OVERFLOW;
      end
    end else if (CTRL == MUL) begin
      CTRL3 <= CTRL2;
      EXPCARRY1 <= EXPCARRY0;
      SIGN <= ASIGN1 ^ BSIGN1;

      RML3_12345678          <= WML3_12345678;
      RML3_910111213141516   <= WML3_910111213141516;
      RML3_1718192021222324 <= WML3_1718192021222324;
    end


    // -----------------------------------
    // Stage 4 : Sign resolution
    // -----------------------------------
    if ((CTRL3 == ADD)|(CTRL3 == SUB)) begin
      OUT <= {SIGN, EXPONENT, NORMOUT};
    end else if (CTRL == MUL) begin
      if (WML5_OUT[24] == 1'b0) begin
        OUT <= {SIGN, EXPCARRY1, WML5_OUT[22:0]};
      end else begin
        OUT <= {SIGN, EXPCARRY1, WML5_OUT[23:1]};
      end
    end
  end
endmodule
```