```
`timescale 1 ns / 1 ns

module pol75
          (
           clk,
           reset,
           clk_enable,
           ce_out,
           Out1
          );


  input   clk;
  input   reset;
  input   clk_enable;
  output  ce_out;
  output  signed [31:0] Out1;  // sfix32_En29


  wire enb_1_7_0;
  wire enb_1_5_0;
  wire enb_1_35_1;
  wire enb_1_35_0;
  wire enb_7_35_1;
  wire enb_1_5_1;
  reg signed [15:0] Sine_Wave1_out1;  // sfix16_En14
  wire signed [15:0] Downsample8_out1;  // sfix16_En14
  wire signed [15:0] Constant5_out1;  // sfix16_En18
  wire signed [31:0] Product1_out1;  // sfix32_En32
  reg signed [15:0] Delay_out1;  // sfix16_En14
  wire signed [15:0] Downsample1_out1;  // sfix16_En14
  wire signed [15:0] Constant1_out1;  // sfix16_En15
  wire signed [31:0] Product25_out1;  // sfix32_En29
  wire signed [31:0] Add1_out1;  // sfix32_En29
  reg signed [15:0] Delay1_out1;  // sfix16_En14
  wire signed [15:0] Downsample2_out1;  // sfix16_En14
  wire signed [15:0] Constant3_out1;  // sfix16_En15
  wire signed [31:0] Product8_out1;  // sfix32_En29
  wire signed [31:0] Add3_out1;  // sfix32_En29
  reg signed [15:0] Delay2_out1;  // sfix16_En14
  wire signed [15:0] Downsample3_out1;  // sfix16_En14
  wire signed [15:0] Constant16_out1;  // sfix16_En15
  wire signed [31:0] Product16_out1;  // sfix32_En29
  wire signed [31:0] Add11_out1;  // sfix32_En29
  reg signed [15:0] Delay3_out1;  // sfix16_En14
  wire signed [15:0] Downsample4_out1;  // sfix16_En14
  wire signed [15:0] Constant7_out1;  // sfix16_En17
  wire signed [31:0] Product5_out1;  // sfix32_En31
  wire signed [31:0] Add5_out1;  // sfix32_En29
  wire signed [31:0] Upsample6_out1;  // sfix32_En29
  reg signed [31:0] Delay9_out1;  // sfix32_En29
  wire signed [15:0] Constant2_out1;  // sfix16_En17
  wire signed [31:0] Product2_out1;  // sfix32_En31
  wire signed [15:0] Constant14_out1;  // sfix16_En15
```

```
wire signed [31:0] Product7_out1;  // sfix32_En29
wire signed [31:0] Add4_out1;  // sfix32_En29
wire signed [15:0] Constant18_out1;  // sfix16_En15
wire signed [31:0] Product15_out1;  // sfix32_En29
wire signed [31:0] Add12_out1;  // sfix32_En29
wire signed [15:0] Constant9_out1;  // sfix16_En15
wire signed [31:0] Product4_out1;  // sfix32_En29
wire signed [31:0] Add6_out1;  // sfix32_En29
reg signed [15:0] Delay4_out1;  // sfix16_En14
wire signed [15:0] Downsample5_out1;  // sfix16_En14
wire signed [15:0] Constant25_out1;  // sfix16_En18
wire signed [31:0] Product24_out1;  // sfix32_En32
wire signed [31:0] Add14_out1;  // sfix32_En29
wire signed [31:0] Upsample1_out1;  // sfix32_En29
wire signed [31:0] Add28_out1;  // sfix32_En29
reg signed [31:0] Delay8_out1;  // sfix32_En29
wire signed [15:0] Constant15_out1;  // sfix16_En16
wire signed [31:0] Product6_out1;  // sfix32_En30
wire signed [15:0] Constant13_out1;  // sfix16_En15
wire signed [31:0] Product14_out1;  // sfix32_En29
wire signed [31:0] Add13_out1;  // sfix32_En29
wire signed [15:0] Constant6_out1;  // sfix16_En15
wire signed [31:0] Product3_out1;  // sfix32_En29
wire signed [31:0] Add7_out1;  // sfix32_En29
wire signed [15:0] Constant23_out1;  // sfix16_En16
wire signed [31:0] Product23_out1;  // sfix32_En30
wire signed [31:0] Add15_out1;  // sfix32_En29
wire signed [31:0] Upsample2_out1;  // sfix32_En29
wire signed [31:0] Add29_out1;  // sfix32_En29
reg signed [31:0] Delay11_out1;  // sfix32_En29
wire signed [15:0] Constant4_out1;  // sfix16_En18
wire signed [31:0] Product9_out1;  // sfix32_En32
wire signed [15:0] Constant17_out1;  // sfix16_En15
wire signed [31:0] Product17_out1;  // sfix32_En29
wire signed [31:0] Add10_out1;  // sfix32_En29
wire signed [15:0] Constant8_out1;  // sfix16_En15
wire signed [31:0] Product10_out1;  // sfix32_En29
wire signed [31:0] Add2_out1;  // sfix32_En29
wire signed [15:0] Constant24_out1;  // sfix16_En15
wire signed [31:0] Product19_out1;  // sfix32_En29
wire signed [31:0] Add9_out1;  // sfix32_En29
reg signed [15:0] Delay5_out1;  // sfix16_En14
wire signed [15:0] Downsample6_out1;  // sfix16_En14
wire signed [15:0] Constant28_out1;  // sfix16_En17
wire signed [31:0] Product28_out1;  // sfix32_En31
wire signed [31:0] Add19_out1;  // sfix32_En29
wire signed [31:0] Upsample3_out1;  // sfix32_En29
wire signed [31:0] Add30_out1;  // sfix32_En29
reg signed [31:0] Delay12_out1;  // sfix32_En29
wire signed [15:0] Constant19_out1;  // sfix16_En17
wire signed [31:0] Product18_out1;  // sfix32_En31
wire signed [15:0] Constant10_out1;  // sfix16_En15
wire signed [31:0] Product11_out1;  // sfix32_En29
wire signed [31:0] Add8_out1;  // sfix32_En29
```

```
wire signed [15:0] Constant20_out1;  // sfix16_En15
wire signed [31:0] Product20_out1;  // sfix32_En29
wire signed [31:0] Add16_out1;  // sfix32_En29
wire signed [15:0] Constant26_out1;  // sfix16_En15
wire signed [31:0] Product27_out1;  // sfix32_En29
wire signed [31:0] Add20_out1;  // sfix32_En29
reg signed [15:0] Delay6_out1;  // sfix16_En14
wire signed [15:0] Downsample7_out1;  // sfix16_En14
wire signed [15:0] Constant30_out1;  // sfix16_En18
wire signed [31:0] Product31_out1;  // sfix32_En32
wire signed [31:0] Add23_out1;  // sfix32_En29
wire signed [31:0] Upsample4_out1;  // sfix32_En29
wire signed [31:0] Add31_out1;  // sfix32_En29
reg signed [31:0] Delay10_out1;  // sfix32_En29
wire signed [15:0] Constant12_out1;  // sfix16_En18
wire signed [31:0] Product13_out1;  // sfix32_En32
wire signed [15:0] Constant22_out1;  // sfix16_En16
wire signed [31:0] Product22_out1;  // sfix32_En30
wire signed [31:0] Add18_out1;  // sfix32_En30
wire signed [15:0] Constant29_out1;  // sfix16_En15
wire signed [31:0] Product29_out1;  // sfix32_En29
wire signed [31:0] Add22_out1;  // sfix32_En29
wire signed [15:0] Constant32_out1;  // sfix16_En15
wire signed [31:0] Product32_out1;  // sfix32_En29
wire signed [31:0] Add25_out1;  // sfix32_En29
reg signed [15:0] Delay7_out1;  // sfix16_En14
wire signed [15:0] Downsample9_out1;  // sfix16_En14
wire signed [15:0] Constant34_out1;  // sfix16_En16
wire signed [31:0] Product34_out1;  // sfix32_En30
wire signed [31:0] Add27_out1;  // sfix32_En29
wire signed [31:0] Upsample5_out1;  // sfix32_En29
wire signed [31:0] Add32_out1;  // sfix32_En29
reg signed [31:0] Delay13_out1;  // sfix32_En29
wire signed [15:0] Constant11_out1;  // sfix16_En16
wire signed [31:0] Product12_out1;  // sfix32_En30
wire signed [15:0] Constant21_out1;  // sfix16_En15
wire signed [31:0] Product21_out1;  // sfix32_En29
wire signed [31:0] Add17_out1;  // sfix32_En29
wire signed [15:0] Constant27_out1;  // sfix16_En15
wire signed [31:0] Product26_out1;  // sfix32_En29
wire signed [31:0] Add21_out1;  // sfix32_En29
wire signed [15:0] Constant31_out1;  // sfix16_En16
wire signed [31:0] Product30_out1;  // sfix32_En30
wire signed [31:0] Add24_out1;  // sfix32_En29
wire signed [15:0] Constant33_out1;  // sfix16_En18
wire signed [31:0] Product33_out1;  // sfix32_En32
wire signed [31:0] Add26_out1;  // sfix32_En29
wire signed [31:0] Upsample7_out1;  // sfix32_En29
wire signed [31:0] Add33_out1;  // sfix32_En29
reg [6:0] address_cnt;  // ufix7
wire signed [15:0] tmpout;  // sfix16_En14
reg signed [15:0] regout;  // sfix16_En14
wire signed [15:0] tmpout_1;  // sfix16_En14
reg signed [15:0] regout_1;  // sfix16_En14
```

```
wire signed [31:0] add_cast;  // sfix32_En29
wire signed [31:0] add_cast_1;  // sfix32_En29
wire signed [32:0] add_temp;  // sfix33_En29
wire signed [15:0] tmpout_2;  // sfix16_En14
reg signed [15:0] regout_2;  // sfix16_En14
wire signed [31:0] add_cast_2;  // sfix32_En29
wire signed [31:0] add_cast_3;  // sfix32_En29
wire signed [32:0] add_temp_1;  // sfix33_En29
wire signed [15:0] tmpout_3;  // sfix16_En14
reg signed [15:0] regout_3;  // sfix16_En14
wire signed [31:0] add_cast_4;  // sfix32_En29
wire signed [31:0] add_cast_5;  // sfix32_En29
wire signed [32:0] add_temp_2;  // sfix33_En29
wire signed [15:0] tmpout_4;  // sfix16_En14
reg signed [15:0] regout_4;  // sfix16_En14
wire signed [31:0] add_cast_6;  // sfix32_En29
wire signed [31:0] add_cast_7;  // sfix32_En29
wire signed [32:0] add_temp_3;  // sfix33_En29
// Up sample by 7 Constant Declaration
wire signed [31:0] zero;  // sfix32_En29
// Up sample by 7 Signal Declaration
wire signed [31:0] muxout;  // sfix32_En29
reg signed [31:0] regout_5;  // sfix32_En29
wire signed [31:0] add_cast_8;  // sfix32_En29
wire signed [31:0] add_cast_9;  // sfix32_En29
wire signed [32:0] add_temp_4;  // sfix33_En29
wire signed [31:0] add_cast_10;  // sfix32_En29
wire signed [31:0] add_cast_11;  // sfix32_En29
wire signed [32:0] add_temp_5;  // sfix33_En29
wire signed [31:0] add_cast_12;  // sfix32_En29
wire signed [31:0] add_cast_13;  // sfix32_En29
wire signed [32:0] add_temp_6;  // sfix33_En29
wire signed [15:0] tmpout_5;  // sfix16_En14
reg signed [15:0] regout_6;  // sfix16_En14
wire signed [31:0] add_cast_14;  // sfix32_En29
wire signed [31:0] add_cast_15;  // sfix32_En29
wire signed [32:0] add_temp_7;  // sfix33_En29
// Up sample by 7 Constant Declaration
wire signed [31:0] zero_1;  // sfix32_En29
// Up sample by 7 Signal Declaration
wire signed [31:0] muxout_1;  // sfix32_En29
reg signed [31:0] regout_7;  // sfix32_En29
wire signed [31:0] add_cast_16;  // sfix32_En29
wire signed [31:0] add_cast_17;  // sfix32_En29
wire signed [32:0] add_temp_8;  // sfix33_En29
wire signed [31:0] add_cast_18;  // sfix32_En29
wire signed [31:0] add_cast_19;  // sfix32_En29
wire signed [32:0] add_temp_9;  // sfix33_En29
wire signed [31:0] add_cast_20;  // sfix32_En29
wire signed [31:0] add_cast_21;  // sfix32_En29
wire signed [32:0] add_temp_10;  // sfix33_En29
wire signed [31:0] add_cast_22;  // sfix32_En29
wire signed [31:0] add_cast_23;  // sfix32_En29
wire signed [32:0] add_temp_11;  // sfix33_En29
```

```
// Up sample by 7 Constant Declaration
wire signed [31:0] zero_2;  // sfix32_En29
// Up sample by 7 Signal Declaration
wire signed [31:0] muxout_2;  // sfix32_En29
reg signed [31:0] regout_8;  // sfix32_En29
wire signed [31:0] add_cast_24;  // sfix32_En29
wire signed [31:0] add_cast_25;  // sfix32_En29
wire signed [32:0] add_temp_12;  // sfix33_En29
wire signed [31:0] add_cast_26;  // sfix32_En29
wire signed [31:0] add_cast_27;  // sfix32_En29
wire signed [32:0] add_temp_13;  // sfix33_En29
wire signed [31:0] add_cast_28;  // sfix32_En29
wire signed [31:0] add_cast_29;  // sfix32_En29
wire signed [32:0] add_temp_14;  // sfix33_En29
wire signed [31:0] add_cast_30;  // sfix32_En29
wire signed [31:0] add_cast_31;  // sfix32_En29
wire signed [32:0] add_temp_15;  // sfix33_En29
wire signed [15:0] tmpout_6;  // sfix16_En14
reg signed [15:0] regout_9;  // sfix16_En14
wire signed [31:0] add_cast_32;  // sfix32_En29
wire signed [31:0] add_cast_33;  // sfix32_En29
wire signed [32:0] add_temp_16;  // sfix33_En29
// Up sample by 7 Constant Declaration
wire signed [31:0] zero_3;  // sfix32_En29
// Up sample by 7 Signal Declaration
wire signed [31:0] muxout_3;  // sfix32_En29
reg signed [31:0] regout_10;  // sfix32_En29
wire signed [31:0] add_cast_34;  // sfix32_En29
wire signed [31:0] add_cast_35;  // sfix32_En29
wire signed [32:0] add_temp_17;  // sfix33_En29
wire signed [31:0] add_cast_36;  // sfix32_En29
wire signed [31:0] add_cast_37;  // sfix32_En29
wire signed [32:0] add_temp_18;  // sfix33_En29
wire signed [31:0] add_cast_38;  // sfix32_En29
wire signed [31:0] add_cast_39;  // sfix32_En29
wire signed [32:0] add_temp_19;  // sfix33_En29
wire signed [31:0] add_cast_40;  // sfix32_En29
wire signed [31:0] add_cast_41;  // sfix32_En29
wire signed [32:0] add_temp_20;  // sfix33_En29
wire signed [15:0] tmpout_7;  // sfix16_En14
reg signed [15:0] regout_11;  // sfix16_En14
wire signed [31:0] add_cast_42;  // sfix32_En29
wire signed [31:0] add_cast_43;  // sfix32_En29
wire signed [32:0] add_temp_21;  // sfix33_En29
// Up sample by 7 Constant Declaration
wire signed [31:0] zero_4;  // sfix32_En29
// Up sample by 7 Signal Declaration
wire signed [31:0] muxout_4;  // sfix32_En29
reg signed [31:0] regout_12;  // sfix32_En29
wire signed [31:0] add_cast_44;  // sfix32_En29
wire signed [31:0] add_cast_45;  // sfix32_En29
wire signed [32:0] add_temp_22;  // sfix33_En29
wire signed [31:0] add_cast_46;  // sfix32_En30
wire signed [31:0] add_cast_47;  // sfix32_En30
```

```
  wire signed [32:0] add_temp_23;  // sfix33_En30
  wire signed [31:0] add_cast_48;  // sfix32_En29
  wire signed [31:0] add_cast_49;  // sfix32_En29
  wire signed [32:0] add_temp_24;  // sfix33_En29
  wire signed [31:0] add_cast_50;  // sfix32_En29
  wire signed [31:0] add_cast_51;  // sfix32_En29
  wire signed [32:0] add_temp_25;  // sfix33_En29
  wire signed [15:0] tmpout_8;  // sfix16_En14
  reg signed [15:0] regout_13;  // sfix16_En14
  wire signed [31:0] add_cast_52;  // sfix32_En29
  wire signed [31:0] add_cast_53;  // sfix32_En29
  wire signed [32:0] add_temp_26;  // sfix33_En29
  // Up sample by 7 Constant Declaration
  wire signed [31:0] zero_5;  // sfix32_En29
  // Up sample by 7 Signal Declaration
  wire signed [31:0] muxout_5;  // sfix32_En29
  reg signed [31:0] regout_14;  // sfix32_En29
  wire signed [31:0] add_cast_54;  // sfix32_En29
  wire signed [31:0] add_cast_55;  // sfix32_En29
  wire signed [32:0] add_temp_27;  // sfix33_En29
  wire signed [31:0] add_cast_56;  // sfix32_En29
  wire signed [31:0] add_cast_57;  // sfix32_En29
  wire signed [32:0] add_temp_28;  // sfix33_En29
  wire signed [31:0] add_cast_58;  // sfix32_En29
  wire signed [31:0] add_cast_59;  // sfix32_En29
  wire signed [32:0] add_temp_29;  // sfix33_En29
  wire signed [31:0] add_cast_60;  // sfix32_En29
  wire signed [31:0] add_cast_61;  // sfix32_En29
  wire signed [32:0] add_temp_30;  // sfix33_En29
  wire signed [31:0] add_cast_62;  // sfix32_En29
  wire signed [31:0] add_cast_63;  // sfix32_En29
  wire signed [32:0] add_temp_31;  // sfix33_En29
  // Up sample by 7 Constant Declaration
  wire signed [31:0] zero_6;  // sfix32_En29
  // Up sample by 7 Signal Declaration
  wire signed [31:0] muxout_6;  // sfix32_En29
  reg signed [31:0] regout_15;  // sfix32_En29
  wire signed [31:0] add_cast_64;  // sfix32_En29
  wire signed [31:0] add_cast_65;  // sfix32_En29
  wire signed [32:0] add_temp_32;  // sfix33_En29

  Timing_Controller   u_Timing_Controller   (.clk(clk),
                                             .reset(reset),
                                             .clk_enable(clk_enable),
                                             .enb_1_5_0_1(enb_1_5_0),
                                             .enb_1_5_1_1(enb_1_5_1),
                                             .enb_1_7_0_1(enb_1_7_0),
                                             .enb_1_35_0_1(enb_1_35_0),
                                             .enb_1_35_1_1(enb_1_35_1),
                                             .enb_7_35_1_1(enb_7_35_1)
                                             );

  assign ce_out = enb_1_5_1;
// ADDRESS COUNTER
```

```
  always @ (posedge clk or posedge reset)
    begin: Sine_Wave1_addrcnt_temp_process6
      if (reset == 1'b1) begin
        address_cnt <= 7'b0000000;
      end
      else begin
        if (enb_1_7_0 == 1'b1) begin
          if (address_cnt == 7'b1100011) begin
            address_cnt <= 7'b0000000;
          end
          else begin
            address_cnt <= address_cnt + 1;
          end
        end
      end
    end // Sine_Wave1_addrcnt_temp_process6

// FULL WAVE LOOKUP TABLE
  always @(address_cnt)
  begin
    case(address_cnt)
      7'b0000000 : Sine_Wave1_out1 = 16'b0000000000000000;
      7'b0000001 : Sine_Wave1_out1 = 16'b0000010000000101;
      7'b0000010 : Sine_Wave1_out1 = 16'b0000100000000101;
      7'b0000011 : Sine_Wave1_out1 = 16'b0000101111111110;
      7'b0000100 : Sine_Wave1_out1 = 16'b0000111111101011;
      7'b0000101 : Sine_Wave1_out1 = 16'b0001001111000111;
      7'b0000110 : Sine_Wave1_out1 = 16'b0001011110001111;
      7'b0000111 : Sine_Wave1_out1 = 16'b0001101101000000;
      7'b0001000 : Sine_Wave1_out1 = 16'b0001111011010101;
      7'b0001001 : Sine_Wave1_out1 = 16'b0010001001001011;
      7'b0001010 : Sine_Wave1_out1 = 16'b0010010110011110;
      7'b0001011 : Sine_Wave1_out1 = 16'b0010100011001100;
      7'b0001100 : Sine_Wave1_out1 = 16'b0010101111010000;
      7'b0001101 : Sine_Wave1_out1 = 16'b0010111010100111;
      7'b0001110 : Sine_Wave1_out1 = 16'b0011000101010000;
      7'b0001111 : Sine_Wave1_out1 = 16'b0011001111000111;
      7'b0010000 : Sine_Wave1_out1 = 16'b0011011000001001;
      7'b0010001 : Sine_Wave1_out1 = 16'b0011100000010101;
      7'b0010010 : Sine_Wave1_out1 = 16'b0011100111101001;
      7'b0010011 : Sine_Wave1_out1 = 16'b0011101110000001;
      7'b0010100 : Sine_Wave1_out1 = 16'b0011110011011110;
      7'b0010101 : Sine_Wave1_out1 = 16'b0011110111111101;
      7'b0010110 : Sine_Wave1_out1 = 16'b0011111011011110;
      7'b0010111 : Sine_Wave1_out1 = 16'b0011111101111111;
      7'b0011000 : Sine_Wave1_out1 = 16'b0011111111100000;
      7'b0011001 : Sine_Wave1_out1 = 16'b0100000000000000;
      7'b0011010 : Sine_Wave1_out1 = 16'b0011111111100000;
      7'b0011011 : Sine_Wave1_out1 = 16'b0011111101111111;
      7'b0011100 : Sine_Wave1_out1 = 16'b0011111011011110;
      7'b0011101 : Sine_Wave1_out1 = 16'b0011110111111101;
      7'b0011110 : Sine_Wave1_out1 = 16'b0011110011011110;
      7'b0011111 : Sine_Wave1_out1 = 16'b0011101110000001;
      7'b0100000 : Sine_Wave1_out1 = 16'b0011100111101001;
```

```
7'b0100001 : Sine_Wave1_out1 = 16'b0011100000010101;
7'b0100010 : Sine_Wave1_out1 = 16'b0011011000001001;
7'b0100011 : Sine_Wave1_out1 = 16'b0011001111000111;
7'b0100100 : Sine_Wave1_out1 = 16'b0011000101010000;
7'b0100101 : Sine_Wave1_out1 = 16'b0010111010100111;
7'b0100110 : Sine_Wave1_out1 = 16'b0010101111010000;
7'b0100111 : Sine_Wave1_out1 = 16'b0010100011001100;
7'b0101000 : Sine_Wave1_out1 = 16'b0010010110011110;
7'b0101001 : Sine_Wave1_out1 = 16'b0010001001001011;
7'b0101010 : Sine_Wave1_out1 = 16'b0001111011010101;
7'b0101011 : Sine_Wave1_out1 = 16'b0001101101000000;
7'b0101100 : Sine_Wave1_out1 = 16'b0001011110001111;
7'b0101101 : Sine_Wave1_out1 = 16'b0001001111000111;
7'b0101110 : Sine_Wave1_out1 = 16'b0000111111101011;
7'b0101111 : Sine_Wave1_out1 = 16'b0000101111111110;
7'b0110000 : Sine_Wave1_out1 = 16'b0000100000000101;
7'b0110001 : Sine_Wave1_out1 = 16'b0000010000000101;
7'b0110010 : Sine_Wave1_out1 = 16'b0000000000000000;
7'b0110011 : Sine_Wave1_out1 = 16'b1111101111111011;
7'b0110100 : Sine_Wave1_out1 = 16'b1111011111111011;
7'b0110101 : Sine_Wave1_out1 = 16'b1111010000000010;
7'b0110110 : Sine_Wave1_out1 = 16'b1111000000010101;
7'b0110111 : Sine_Wave1_out1 = 16'b1110110000111001;
7'b0111000 : Sine_Wave1_out1 = 16'b1110100001110001;
7'b0111001 : Sine_Wave1_out1 = 16'b1110010011000000;
7'b0111010 : Sine_Wave1_out1 = 16'b1110000100101011;
7'b0111011 : Sine_Wave1_out1 = 16'b1101110110110101;
7'b0111100 : Sine_Wave1_out1 = 16'b1101101001100010;
7'b0111101 : Sine_Wave1_out1 = 16'b1101011100110100;
7'b0111110 : Sine_Wave1_out1 = 16'b1101010000110000;
7'b0111111 : Sine_Wave1_out1 = 16'b1101000101011001;
7'b1000000 : Sine_Wave1_out1 = 16'b1100111010110000;
7'b1000001 : Sine_Wave1_out1 = 16'b1100110000111001;
7'b1000010 : Sine_Wave1_out1 = 16'b1100100111110111;
7'b1000011 : Sine_Wave1_out1 = 16'b1100011111101011;
7'b1000100 : Sine_Wave1_out1 = 16'b1100011000010111;
7'b1000101 : Sine_Wave1_out1 = 16'b1100010001111111;
7'b1000110 : Sine_Wave1_out1 = 16'b1100001100100010;
7'b1000111 : Sine_Wave1_out1 = 16'b1100001000000011;
7'b1001000 : Sine_Wave1_out1 = 16'b1100000100100010;
7'b1001001 : Sine_Wave1_out1 = 16'b1100000010000001;
7'b1001010 : Sine_Wave1_out1 = 16'b1100000000100000;
7'b1001011 : Sine_Wave1_out1 = 16'b1100000000000000;
7'b1001100 : Sine_Wave1_out1 = 16'b1100000000100000;
7'b1001101 : Sine_Wave1_out1 = 16'b1100000010000001;
7'b1001110 : Sine_Wave1_out1 = 16'b1100000100100010;
7'b1001111 : Sine_Wave1_out1 = 16'b1100001000000011;
7'b1010000 : Sine_Wave1_out1 = 16'b1100001100100010;
7'b1010001 : Sine_Wave1_out1 = 16'b1100010001111111;
7'b1010010 : Sine_Wave1_out1 = 16'b1100011000010111;
7'b1010011 : Sine_Wave1_out1 = 16'b1100011111101011;
7'b1010100 : Sine_Wave1_out1 = 16'b1100100111110111;
7'b1010101 : Sine_Wave1_out1 = 16'b1100110000111001;
7'b1010110 : Sine_Wave1_out1 = 16'b1100111010110000;
```

```
      7'b1010111 : Sine_Wave1_out1 = 16'b1101000101011001;
      7'b1011000 : Sine_Wave1_out1 = 16'b1101010000110000;
      7'b1011001 : Sine_Wave1_out1 = 16'b1101011100110100;
      7'b1011010 : Sine_Wave1_out1 = 16'b1101101001100010;
      7'b1011011 : Sine_Wave1_out1 = 16'b1101110110110101;
      7'b1011100 : Sine_Wave1_out1 = 16'b1110000100101011;
      7'b1011101 : Sine_Wave1_out1 = 16'b1110010011000000;
      7'b1011110 : Sine_Wave1_out1 = 16'b1110100001110001;
      7'b1011111 : Sine_Wave1_out1 = 16'b1110110000111001;
      7'b1100000 : Sine_Wave1_out1 = 16'b1111000000010101;
      7'b1100001 : Sine_Wave1_out1 = 16'b1111010000000010;
      7'b1100010 : Sine_Wave1_out1 = 16'b1111011111111011;
      7'b1100011 : Sine_Wave1_out1 = 16'b1111101111111011;
      default : Sine_Wave1_out1 = 16'b1111101111111011;
    endcase
  end

// %%%% Bypass Register %%%%
  always @ (posedge clk or posedge reset)
    begin: DataHoldRegister_temp_process7
      if (reset == 1'b1) begin
        regout <= 0;
      end
      else begin
        if (enb_1_35_1 == 1'b1) begin
          regout <= Sine_Wave1_out1;
        end
      end
    end // DataHoldRegister_temp_process7

  assign tmpout = (enb_1_35_1 == 1'b1) ? Sine_Wave1_out1 :
            regout;
  assign Downsample8_out1 = tmpout;

  assign Constant5_out1 = 16'b0111001110110110;

  assign Product1_out1 = Downsample8_out1 * Constant5_out1;


  always @ (posedge clk or posedge reset)
    begin: Delay_process
      if (reset == 1'b1) begin
        Delay_out1 <= 0;
      end
      else begin
        if (enb_1_7_0 == 1'b1) begin
          Delay_out1 <= Sine_Wave1_out1;
        end
      end
    end // Delay_process

// %%%% Bypass Register %%%%
  always @ (posedge clk or posedge reset)
    begin: DataHoldRegister_temp_process8
```

```
      if (reset == 1'b1) begin
        regout_1 <= 0;
      end
      else begin
        if (enb_1_35_1 == 1'b1) begin
          regout_1 <= Delay_out1;
        end
      end
    end // DataHoldRegister_temp_process8

  assign tmpout_1 = (enb_1_35_1 == 1'b1) ? Delay_out1 :
              regout_1;
  assign Downsample1_out1 = tmpout_1;

  assign Constant1_out1 = 16'b0100110110000001;

  assign Product25_out1 = Downsample1_out1 * Constant1_out1;

  assign add_cast = $signed({{3{Product1_out1[31]}},
Product1_out1[31:3]});
  assign add_cast_1 = Product25_out1;
  assign add_temp = add_cast + add_cast_1;
  assign Add1_out1 = add_temp[31:0];


  always @ (posedge clk or posedge reset)
    begin: Delay1_process
      if (reset == 1'b1) begin
        Delay1_out1 <= 0;
      end
      else begin
        if (enb_1_7_0 == 1'b1) begin
          Delay1_out1 <= Delay_out1;
        end
      end
    end // Delay1_process

// %%%% Bypass Register %%%%
  always @ (posedge clk or posedge reset)
    begin: DataHoldRegister_temp_process9
      if (reset == 1'b1) begin
        regout_2 <= 0;
      end
      else begin
        if (enb_1_35_1 == 1'b1) begin
          regout_2 <= Delay1_out1;
        end
      end
    end // DataHoldRegister_temp_process9

  assign tmpout_2 = (enb_1_35_1 == 1'b1) ? Delay1_out1 :
              regout_2;
  assign Downsample2_out1 = tmpout_2;
```

```
  assign Constant3_out1 = 16'b0111111110111011;

  assign Product8_out1 = Downsample2_out1 * Constant3_out1;

  assign add_cast_2 = Add1_out1;
  assign add_cast_3 = Product8_out1;
  assign add_temp_1 = add_cast_2 + add_cast_3;
  assign Add3_out1 = add_temp_1[31:0];


  always @ (posedge clk or posedge reset)
    begin: Delay2_process
      if (reset == 1'b1) begin
        Delay2_out1 <= 0;
      end
      else begin
        if (enb_1_7_0 == 1'b1) begin
          Delay2_out1 <= Delay1_out1;
        end
      end
    end // Delay2_process

// %%%% Bypass Register %%%%
  always @ (posedge clk or posedge reset)
    begin: DataHoldRegister_temp_process10
      if (reset == 1'b1) begin
        regout_3 <= 0;
      end
      else begin
        if (enb_1_35_1 == 1'b1) begin
          regout_3 <= Delay2_out1;
        end
      end
    end // DataHoldRegister_temp_process10

  assign tmpout_3 = (enb_1_35_1 == 1'b1) ? Delay2_out1 :
              regout_3;
  assign Downsample3_out1 = tmpout_3;

  assign Constant16_out1 = 16'b0101100001100010;

  assign Product16_out1 = Downsample3_out1 * Constant16_out1;

  assign add_cast_4 = Add3_out1;
  assign add_cast_5 = Product16_out1;
  assign add_temp_2 = add_cast_4 + add_cast_5;
  assign Add11_out1 = add_temp_2[31:0];


  always @ (posedge clk or posedge reset)
    begin: Delay3_process
      if (reset == 1'b1) begin
        Delay3_out1 <= 0;
      end
```

```
      else begin
        if (enb_1_7_0 == 1'b1) begin
          Delay3_out1 <= Delay2_out1;
        end
      end
    end // Delay3_process

// %%%% Bypass Register %%%%
  always @ (posedge clk or posedge reset)
    begin: DataHoldRegister_temp_process11
      if (reset == 1'b1) begin
        regout_4 <= 0;
      end
      else begin
        if (enb_1_35_1 == 1'b1) begin
          regout_4 <= Delay3_out1;
        end
      end
    end // DataHoldRegister_temp_process11

  assign tmpout_4 = (enb_1_35_1 == 1'b1) ? Delay3_out1 :
              regout_4;
  assign Downsample4_out1 = tmpout_4;

  assign Constant7_out1 = 16'b0100111001010110;

  assign Product5_out1 = Downsample4_out1 * Constant7_out1;

  assign add_cast_6 = Add11_out1;
  assign add_cast_7 = $signed({{2{Product5_out1[31]}},
Product5_out1[31:2]});
  assign add_temp_3 = add_cast_6 + add_cast_7;
  assign Add5_out1 = add_temp_3[31:0];

// %%%% Up sample by 7, Sample offset 0 %%%%
  assign zero = 32'h00000000;
  assign muxout = (enb_1_35_1 == 1'b1) ? Add5_out1 :
            zero;

// %%%% Bypass Register %%%%
  always @ (posedge clk or posedge reset)
    begin: DataHoldRegister_temp_process12
      if (reset == 1'b1) begin
        regout_5 <= 0;
      end
      else begin
        if (enb_7_35_1 == 1'b1) begin
          regout_5 <= muxout;
        end
      end
    end // DataHoldRegister_temp_process12

  assign Upsample6_out1 = (enb_7_35_1 == 1'b1) ? muxout :
                    regout_5;
```

```
  always @ (posedge clk or posedge reset)
    begin: Delay9_process
      if (reset == 1'b1) begin
        Delay9_out1 <= 0;
      end
      else begin
        if (enb_1_5_0 == 1'b1) begin
          Delay9_out1 <= Upsample6_out1;
        end
      end
    end // Delay9_process

  assign Constant2_out1 = 16'b0110101000001001;

  assign Product2_out1 = Downsample1_out1 * Constant2_out1;

  assign Constant14_out1 = 16'b0110001010001111;

  assign Product7_out1 = Downsample2_out1 * Constant14_out1;

  assign add_cast_8 = $signed({{2{Product2_out1[31]}},
Product2_out1[31:2]});
  assign add_cast_9 = Product7_out1;
  assign add_temp_4 = add_cast_8 + add_cast_9;
  assign Add4_out1 = add_temp_4[31:0];

  assign Constant18_out1 = 16'b0111110110011111;

  assign Product15_out1 = Downsample3_out1 * Constant18_out1;

  assign add_cast_10 = Add4_out1;
  assign add_cast_11 = Product15_out1;
  assign add_temp_5 = add_cast_10 + add_cast_11;
  assign Add12_out1 = add_temp_5[31:0];

  assign Constant9_out1 = 16'b0100001001010001;

  assign Product4_out1 = Downsample4_out1 * Constant9_out1;

  assign add_cast_12 = Add12_out1;
  assign add_cast_13 = Product4_out1;
  assign add_temp_6 = add_cast_12 + add_cast_13;
  assign Add6_out1 = add_temp_6[31:0];


  always @ (posedge clk or posedge reset)
    begin: Delay4_process
      if (reset == 1'b1) begin
        Delay4_out1 <= 0;
      end
      else begin
        if (enb_1_7_0 == 1'b1) begin
          Delay4_out1 <= Delay3_out1;
```

```
        end
      end
    end // Delay4_process

// %%%% Bypass Register %%%%
  always @ (posedge clk or posedge reset)
    begin: DataHoldRegister_temp_process13
      if (reset == 1'b1) begin
        regout_6 <= 0;
      end
      else begin
        if (enb_1_35_1 == 1'b1) begin
          regout_6 <= Delay4_out1;
        end
      end
    end // DataHoldRegister_temp_process13

  assign tmpout_5 = (enb_1_35_1 == 1'b1) ? Delay4_out1 :
              regout_6;
  assign Downsample5_out1 = tmpout_5;

  assign Constant25_out1 = 16'b0101101001101011;

  assign Product24_out1 = Downsample5_out1 * Constant25_out1;

  assign add_cast_14 = Add6_out1;
  assign add_cast_15 = $signed({{3{Product24_out1[31]}},
Product24_out1[31:3]});
  assign add_temp_7 = add_cast_14 + add_cast_15;
  assign Add14_out1 = add_temp_7[31:0];

// %%%% Up sample by 7, Sample offset 0 %%%%
  assign zero_1 = 32'h00000000;
  assign muxout_1 = (enb_1_35_1 == 1'b1) ? Add14_out1 :
              zero_1;

// %%%% Bypass Register %%%%
  always @ (posedge clk or posedge reset)
    begin: DataHoldRegister_temp_process14
      if (reset == 1'b1) begin
        regout_7 <= 0;
      end
      else begin
        if (enb_7_35_1 == 1'b1) begin
          regout_7 <= muxout_1;
        end
      end
    end // DataHoldRegister_temp_process14

  assign Upsample1_out1 = (enb_7_35_1 == 1'b1) ? muxout_1 :
                    regout_7;
  assign add_cast_16 = Delay9_out1;
  assign add_cast_17 = Upsample1_out1;
  assign add_temp_8 = add_cast_16 + add_cast_17;
```

```
  assign Add28_out1 = add_temp_8[31:0];


  always @ (posedge clk or posedge reset)
    begin: Delay8_process
      if (reset == 1'b1) begin
        Delay8_out1 <= 0;
      end
      else begin
        if (enb_1_5_0 == 1'b1) begin
          Delay8_out1 <= Add28_out1;
        end
      end
    end // Delay8_process

  assign Constant15_out1 = 16'b0101100101010010;

  assign Product6_out1 = Downsample2_out1 * Constant15_out1;

  assign Constant13_out1 = 16'b0111001101101000;

  assign Product14_out1 = Downsample3_out1 * Constant13_out1;

  assign add_cast_18 = $signed({{1{Product6_out1[31]}},
Product6_out1[31:1]});
  assign add_cast_19 = Product14_out1;
  assign add_temp_9 = add_cast_18 + add_cast_19;
  assign Add13_out1 = add_temp_9[31:0];

  assign Constant6_out1 = 16'b0111001101101000;

  assign Product3_out1 = Downsample4_out1 * Constant6_out1;

  assign add_cast_20 = Add13_out1;
  assign add_cast_21 = Product3_out1;
  assign add_temp_10 = add_cast_20 + add_cast_21;
  assign Add7_out1 = add_temp_10[31:0];

  assign Constant23_out1 = 16'b0101100101010010;

  assign Product23_out1 = Downsample5_out1 * Constant23_out1;

  assign add_cast_22 = Add7_out1;
  assign add_cast_23 = $signed({{1{Product23_out1[31]}},
Product23_out1[31:1]});
  assign add_temp_11 = add_cast_22 + add_cast_23;
  assign Add15_out1 = add_temp_11[31:0];

// %%%% Up sample by 7, Sample offset 0 %%%%
  assign zero_2 = 32'h00000000;
  assign muxout_2 = (enb_1_35_1 == 1'b1) ? Add15_out1 :
              zero_2;

// %%%% Bypass Register %%%%
```

```
  always @ (posedge clk or posedge reset)
    begin: DataHoldRegister_temp_process15
      if (reset == 1'b1) begin
        regout_8 <= 0;
      end
      else begin
        if (enb_7_35_1 == 1'b1) begin
          regout_8 <= muxout_2;
        end
      end
    end // DataHoldRegister_temp_process15

  assign Upsample2_out1 = (enb_7_35_1 == 1'b1) ? muxout_2 :
                   regout_8;
  assign add_cast_24 = Delay8_out1;
  assign add_cast_25 = Upsample2_out1;
  assign add_temp_12 = add_cast_24 + add_cast_25;
  assign Add29_out1 = add_temp_12[31:0];


  always @ (posedge clk or posedge reset)
    begin: Delay11_process
      if (reset == 1'b1) begin
        Delay11_out1 <= 0;
      end
      else begin
        if (enb_1_5_0 == 1'b1) begin
          Delay11_out1 <= Add29_out1;
        end
      end
    end // Delay11_process

  assign Constant4_out1 = 16'b0101101001101011;

  assign Product9_out1 = Downsample2_out1 * Constant4_out1;

  assign Constant17_out1 = 16'b0100001001010001;

  assign Product17_out1 = Downsample3_out1 * Constant17_out1;

  assign add_cast_26 = $signed({{3{Product9_out1[31]}},
Product9_out1[31:3]});
  assign add_cast_27 = Product17_out1;
  assign add_temp_13 = add_cast_26 + add_cast_27;
  assign Add10_out1 = add_temp_13[31:0];

  assign Constant8_out1 = 16'b0111110110011111;

  assign Product10_out1 = Downsample4_out1 * Constant8_out1;

  assign add_cast_28 = Add10_out1;
  assign add_cast_29 = Product10_out1;
  assign add_temp_14 = add_cast_28 + add_cast_29;
  assign Add2_out1 = add_temp_14[31:0];
```

```
  assign Constant24_out1 = 16'b0110001010001111;

  assign Product19_out1 = Downsample5_out1 * Constant24_out1;

  assign add_cast_30 = Add2_out1;
  assign add_cast_31 = Product19_out1;
  assign add_temp_15 = add_cast_30 + add_cast_31;
  assign Add9_out1 = add_temp_15[31:0];


  always @ (posedge clk or posedge reset)
    begin: Delay5_process
      if (reset == 1'b1) begin
        Delay5_out1 <= 0;
      end
      else begin
        if (enb_1_7_0 == 1'b1) begin
          Delay5_out1 <= Delay4_out1;
        end
      end
    end // Delay5_process

// %%%% Bypass Register %%%%
  always @ (posedge clk or posedge reset)
    begin: DataHoldRegister_temp_process16
      if (reset == 1'b1) begin
        regout_9 <= 0;
      end
      else begin
        if (enb_1_35_1 == 1'b1) begin
          regout_9 <= Delay5_out1;
        end
      end
    end // DataHoldRegister_temp_process16

  assign tmpout_6 = (enb_1_35_1 == 1'b1) ? Delay5_out1 :
              regout_9;
  assign Downsample6_out1 = tmpout_6;

  assign Constant28_out1 = 16'b0110101000001001;

  assign Product28_out1 = Downsample6_out1 * Constant28_out1;

  assign add_cast_32 = Add9_out1;
  assign add_cast_33 = $signed({{2{Product28_out1[31]}},
Product28_out1[31:2]});
  assign add_temp_16 = add_cast_32 + add_cast_33;
  assign Add19_out1 = add_temp_16[31:0];

// %%%% Up sample by 7, Sample offset 0 %%%%
  assign zero_3 = 32'h00000000;
  assign muxout_3 = (enb_1_35_1 == 1'b1) ? Add19_out1 :
              zero_3;
```

```
// %%%% Bypass Register %%%%
  always @ (posedge clk or posedge reset)
    begin: DataHoldRegister_temp_process17
      if (reset == 1'b1) begin
        regout_10 <= 0;
      end
      else begin
        if (enb_7_35_1 == 1'b1) begin
          regout_10 <= muxout_3;
        end
      end
    end // DataHoldRegister_temp_process17

  assign Upsample3_out1 = (enb_7_35_1 == 1'b1) ? muxout_3 :
                      regout_10;
  assign add_cast_34 = Delay11_out1;
  assign add_cast_35 = Upsample3_out1;
  assign add_temp_17 = add_cast_34 + add_cast_35;
  assign Add30_out1 = add_temp_17[31:0];


  always @ (posedge clk or posedge reset)
    begin: Delay12_process
      if (reset == 1'b1) begin
        Delay12_out1 <= 0;
      end
      else begin
        if (enb_1_5_0 == 1'b1) begin
          Delay12_out1 <= Add30_out1;
        end
      end
    end // Delay12_process

  assign Constant19_out1 = 16'b0100111001010110;

  assign Product18_out1 = Downsample3_out1 * Constant19_out1;

  assign Constant10_out1 = 16'b0101100001100010;

  assign Product11_out1 = Downsample4_out1 * Constant10_out1;

  assign add_cast_36 = $signed({{2{Product18_out1[31]}},
Product18_out1[31:2]});
  assign add_cast_37 = Product11_out1;
  assign add_temp_18 = add_cast_36 + add_cast_37;
  assign Add8_out1 = add_temp_18[31:0];

  assign Constant20_out1 = 16'b0111111110111011;

  assign Product20_out1 = Downsample5_out1 * Constant20_out1;

  assign add_cast_38 = Add8_out1;
  assign add_cast_39 = Product20_out1;
```

```
  assign add_temp_19 = add_cast_38 + add_cast_39;
  assign Add16_out1 = add_temp_19[31:0];

  assign Constant26_out1 = 16'b0100110110000001;

  assign Product27_out1 = Downsample6_out1 * Constant26_out1;

  assign add_cast_40 = Add16_out1;
  assign add_cast_41 = Product27_out1;
  assign add_temp_20 = add_cast_40 + add_cast_41;
  assign Add20_out1 = add_temp_20[31:0];


  always @ (posedge clk or posedge reset)
    begin: Delay6_process
      if (reset == 1'b1) begin
        Delay6_out1 <= 0;
      end
      else begin
        if (enb_1_7_0 == 1'b1) begin
          Delay6_out1 <= Delay5_out1;
        end
      end
    end // Delay6_process

// %%%% Bypass Register %%%%
  always @ (posedge clk or posedge reset)
    begin: DataHoldRegister_temp_process18
      if (reset == 1'b1) begin
        regout_11 <= 0;
      end
      else begin
        if (enb_1_35_1 == 1'b1) begin
          regout_11 <= Delay6_out1;
        end
      end
    end // DataHoldRegister_temp_process18

  assign tmpout_7 = (enb_1_35_1 == 1'b1) ? Delay6_out1 :
              regout_11;
  assign Downsample7_out1 = tmpout_7;

  assign Constant30_out1 = 16'b0111001110110110;

  assign Product31_out1 = Downsample7_out1 * Constant30_out1;

  assign add_cast_42 = Add20_out1;
  assign add_cast_43 = $signed({{3{Product31_out1[31]}},
Product31_out1[31:3]});
  assign add_temp_21 = add_cast_42 + add_cast_43;
  assign Add23_out1 = add_temp_21[31:0];

// %%%% Up sample by 7, Sample offset 0 %%%%
  assign zero_4 = 32'h00000000;
```

```
  assign muxout_4 = (enb_1_35_1 == 1'b1) ? Add23_out1 :
              zero_4;

// %%%% Bypass Register %%%%
  always @ (posedge clk or posedge reset)
    begin: DataHoldRegister_temp_process19
      if (reset == 1'b1) begin
        regout_12 <= 0;
      end
      else begin
        if (enb_7_35_1 == 1'b1) begin
          regout_12 <= muxout_4;
        end
      end
    end // DataHoldRegister_temp_process19

  assign Upsample4_out1 = (enb_7_35_1 == 1'b1) ? muxout_4 :
                    regout_12;
  assign add_cast_44 = Delay12_out1;
  assign add_cast_45 = Upsample4_out1;
  assign add_temp_22 = add_cast_44 + add_cast_45;
  assign Add31_out1 = add_temp_22[31:0];


  always @ (posedge clk or posedge reset)
    begin: Delay10_process
      if (reset == 1'b1) begin
        Delay10_out1 <= 0;
      end
      else begin
        if (enb_1_5_0 == 1'b1) begin
          Delay10_out1 <= Add31_out1;
        end
      end
    end // Delay10_process

  assign Constant12_out1 = 16'b0101000111101100;

  assign Product13_out1 = Downsample4_out1 * Constant12_out1;

  assign Constant22_out1 = 16'b0110111001111101;

  assign Product22_out1 = Downsample5_out1 * Constant22_out1;

  assign add_cast_46 = $signed({{2{Product13_out1[31]}},
Product13_out1[31:2]});
  assign add_cast_47 = Product22_out1;
  assign add_temp_23 = add_cast_46 + add_cast_47;
  assign Add18_out1 = add_temp_23[31:0];

  assign Constant29_out1 = 16'b0111100101110110;

  assign Product29_out1 = Downsample6_out1 * Constant29_out1;
```

```
  assign add_cast_48 = $signed({{1{Add18_out1[31]}}, Add18_out1[31:1]});
  assign add_cast_49 = Product29_out1;
  assign add_temp_24 = add_cast_48 + add_cast_49;
  assign Add22_out1 = add_temp_24[31:0];

  assign Constant32_out1 = 16'b0110101110101100;

  assign Product32_out1 = Downsample7_out1 * Constant32_out1;

  assign add_cast_50 = Add22_out1;
  assign add_cast_51 = Product32_out1;
  assign add_temp_25 = add_cast_50 + add_cast_51;
  assign Add25_out1 = add_temp_25[31:0];


  always @ (posedge clk or posedge reset)
    begin: Delay7_process
      if (reset == 1'b1) begin
        Delay7_out1 <= 0;
      end
      else begin
        if (enb_1_7_0 == 1'b1) begin
          Delay7_out1 <= Delay6_out1;
        end
      end
    end // Delay7_process

// %%%% Bypass Register %%%%
  always @ (posedge clk or posedge reset)
    begin: DataHoldRegister_temp_process20
      if (reset == 1'b1) begin
        regout_13 <= 0;
      end
      else begin
        if (enb_1_35_1 == 1'b1) begin
          regout_13 <= Delay7_out1;
        end
      end
    end // DataHoldRegister_temp_process20

  assign tmpout_8 = (enb_1_35_1 == 1'b1) ? Delay7_out1 :
              regout_13;
  assign Downsample9_out1 = tmpout_8;

  assign Constant34_out1 = 16'b0100010111110000;

  assign Product34_out1 = Downsample9_out1 * Constant34_out1;

  assign add_cast_52 = Add25_out1;
  assign add_cast_53 = $signed({{1{Product34_out1[31]}},
Product34_out1[31:1]});
  assign add_temp_26 = add_cast_52 + add_cast_53;
  assign Add27_out1 = add_temp_26[31:0];
```

```
// %%%% Up sample by 7, Sample offset 0 %%%%
  assign zero_5 = 32'h00000000;
  assign muxout_5 = (enb_1_35_1 == 1'b1) ? Add27_out1 :
              zero_5;

// %%%% Bypass Register %%%%
  always @ (posedge clk or posedge reset)
    begin: DataHoldRegister_temp_process21
      if (reset == 1'b1) begin
        regout_14 <= 0;
      end
      else begin
        if (enb_7_35_1 == 1'b1) begin
          regout_14 <= muxout_5;
        end
      end
    end // DataHoldRegister_temp_process21

  assign Upsample5_out1 = (enb_7_35_1 == 1'b1) ? muxout_5 :
                    regout_14;
  assign add_cast_54 = Delay10_out1;
  assign add_cast_55 = Upsample5_out1;
  assign add_temp_27 = add_cast_54 + add_cast_55;
  assign Add32_out1 = add_temp_27[31:0];


  always @ (posedge clk or posedge reset)
    begin: Delay13_process
      if (reset == 1'b1) begin
        Delay13_out1 <= 0;
      end
      else begin
        if (enb_1_5_0 == 1'b1) begin
          Delay13_out1 <= Add32_out1;
        end
      end
    end // Delay13_process

  assign Constant11_out1 = 16'b0100010111110000;

  assign Product12_out1 = Downsample4_out1 * Constant11_out1;

  assign Constant21_out1 = 16'b0110101110101100;

  assign Product21_out1 = Downsample5_out1 * Constant21_out1;

  assign add_cast_56 = $signed({{1{Product12_out1[31]}},
Product12_out1[31:1]});
  assign add_cast_57 = Product21_out1;
  assign add_temp_28 = add_cast_56 + add_cast_57;
  assign Add17_out1 = add_temp_28[31:0];

  assign Constant27_out1 = 16'b0111100101110110;
```

```
  assign Product26_out1 = Downsample6_out1 * Constant27_out1;

  assign add_cast_58 = Add17_out1;
  assign add_cast_59 = Product26_out1;
  assign add_temp_29 = add_cast_58 + add_cast_59;
  assign Add21_out1 = add_temp_29[31:0];

  assign Constant31_out1 = 16'b0110111001111101;

  assign Product30_out1 = Downsample7_out1 * Constant31_out1;

  assign add_cast_60 = Add21_out1;
  assign add_cast_61 = $signed({{1{Product30_out1[31]}},
Product30_out1[31:1]});
  assign add_temp_30 = add_cast_60 + add_cast_61;
  assign Add24_out1 = add_temp_30[31:0];

  assign Constant33_out1 = 16'b0101000111101100;

  assign Product33_out1 = Downsample9_out1 * Constant33_out1;

  assign add_cast_62 = Add24_out1;
  assign add_cast_63 = $signed({{3{Product33_out1[31]}},
Product33_out1[31:3]});
  assign add_temp_31 = add_cast_62 + add_cast_63;
  assign Add26_out1 = add_temp_31[31:0];

// %%%% Up sample by 7, Sample offset 0 %%%%
  assign zero_6 = 32'h00000000;
  assign muxout_6 = (enb_1_35_1 == 1'b1) ? Add26_out1 :
              zero_6;

// %%%% Bypass Register %%%%
  always @ (posedge clk or posedge reset)
    begin: DataHoldRegister_temp_process22
      if (reset == 1'b1) begin
        regout_15 <= 0;
      end
      else begin
        if (enb_7_35_1 == 1'b1) begin
          regout_15 <= muxout_6;
        end
      end
    end // DataHoldRegister_temp_process22

  assign Upsample7_out1 = (enb_7_35_1 == 1'b1) ? muxout_6 :
                    regout_15;
  assign add_cast_64 = Delay13_out1;
  assign add_cast_65 = Upsample7_out1;
  assign add_temp_32 = add_cast_64 + add_cast_65;
  assign Add33_out1 = add_temp_32[31:0];

  assign Out1 = Add33_out1;
```

```
endmodule  // pol75
```