ACPL-798J Evaluation Board Kit (PMOD Type 1 Interface)

Isolated Sigma-Delta Modulator with LVDS Interface

User Guide

Description

The ACPL-798J isolated sigma-delta (Σ–Δ) modulator converts an analog input signal into a high-speed (up to 25MHz) single-bit data stream by means of a sigma-delta over-sampling modulator. The time average of the modulator data is directly proportional to the input signal voltage. The modulator uses external clock ranges from 5 MHz to 25 MHz that is coupled across the isolation barrier. This arrangement allows synchronous operation of data acquisition to any digital controller, and adjustable clock for speed requirements of the application. The modulator data are encoded and transmitted across the isolation boundary where they are recovered and decoded into high-speed data stream of digital ones and zeros. The original signal information is represented by the density of ones in the data output. [1]

Input signal information is contained in the modulator output data stream, represented by the density of ones and zeros. The density of ones is proportional to the input signal voltage, as shown in Figure 1. A differential input signal of 0 V ideally produces a data stream of ones 50% of the time and zeros 50% of the time. A differential input of –200 mV corresponds to 18.75% density of ones, and a differential input of +200 mV is represented by 81.25% density of ones in the data stream. A differential input of +320 mV or higher results in ideally all ones in the data stream, while input of –320 mV or lower will result in all zeros ideally. Table 1 shows this relationship.

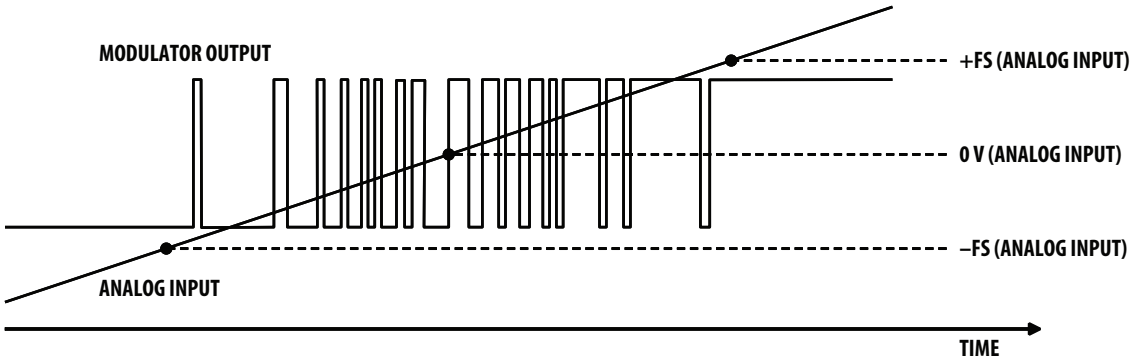


Figure 1. Modulator output vs. analog input

Table 1 Input voltage with ideal corresponding density of 1s at modulator data output, and ADC code.

Table 1. Input voltage with ideal corresponding density of 1s at modulator data output, and ADC code.

Analog Input	Voltage Input	Density of 1s	ADC Code (16-bit unsigned decimation)
Full-Scale Range	640 mV		
+Full-Scale	+320 mV	100%	65,535
+Recommended Input Range	+200 mV	81.25%	53,248
Zero	0 mV	50%	32,768
–Recommended Input Range	–200 mV	18.75%	12,288
–Full-Scale	–320 mV	0%	0

This User Manual is provided to assist you in the evaluation of product(s) currently under development. Until Avago Technologies releases this product for general sales, Avago Technologies reserves the right to alter prices, specifications, features, capabilities, functions, release dates, and remove availability of the product(s) at anytime.

The original analog signal that is converted to a digital bit stream by the over-sampling sigma-delta modulator, can be recovered by means of filtering in the digital domain. A common and simple way is through implementation of a cascaded integrated comb (CIC) filter or Sinc3 filter. The digital filter averages or decimates the over-sampled bit stream and effectively converts it into a multi-bit digital equivalent code of the original analog input signal. With a 20MHz external clock frequency, 256 decimation ratio and 16-bit word settings, the output data rate is 78 kHz (= 20MHz/256). This filter can be implemented in an ASIC, an FPGA or a DSP.

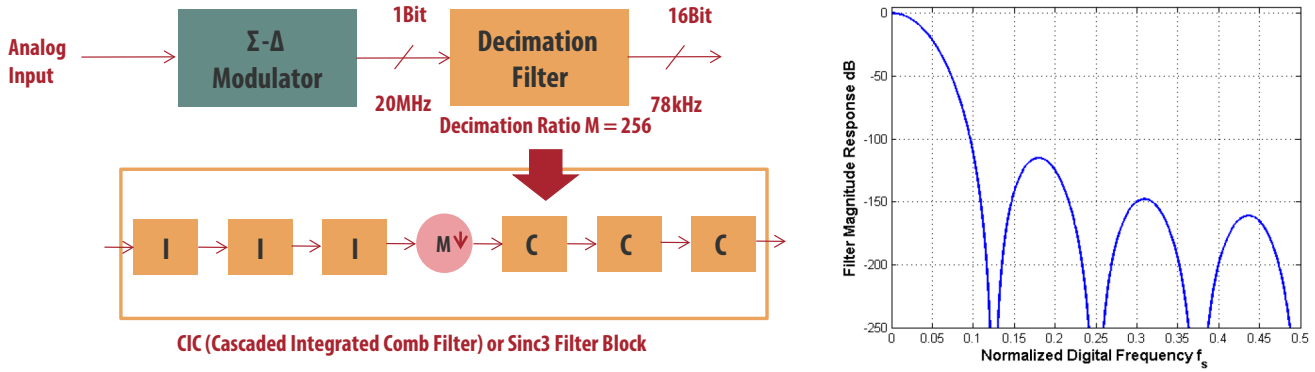


Figure 2. CIC or Sinc3 filter block.

By scaling the filter decimation ratio, it is possible to scale the resolution vs response speed accordingly and vice versa.

Table 2. Flexibility to scale, resolution vs speed.

	Fs=20MHz			Fs=10MHz		
Decimation Ratio (R)	Throughput Rate (Fs/R) KHz	Effective Number of Bits (ENOB)	Filter Delay (us)	Throughput Rate (Fs/R)	Effective Number of Bits (ENOB)	Filter Delay (us)
256	78.1	12	12.8	39.1	12	25.6
128	156.2	11	6.4	78.1	11	12.8
64	312.5	11	3.2	156.2	11	6.4
32	625	9	1.6	312.5	10	3.2

PMOD Interface Evaluation Board

The purpose of the ACPL-798J PMOD interface type 1 evaluation board is to make it easier for system designer to quickly assemble and integrate Avago's ACPL-798J LVDS digital modulator to FPGA, DSP or microcontroller development kits / reference boards which also come with PMOD interface for prototyping or evaluation purpose with not soldering required.

Pmod interface or Peripheral Module interface is a standard defined by Digilent Inc in the Digilent Pmod™ Interface Specification [2] for peripherals used with FPGAs or micro-controllers. Pmods come in a standard 6-Pin interface with 4 signals, one ground and one power pin. Double and quad Pmods also exist. These duplicate the standard interface to allow more signals to pass through to the module.

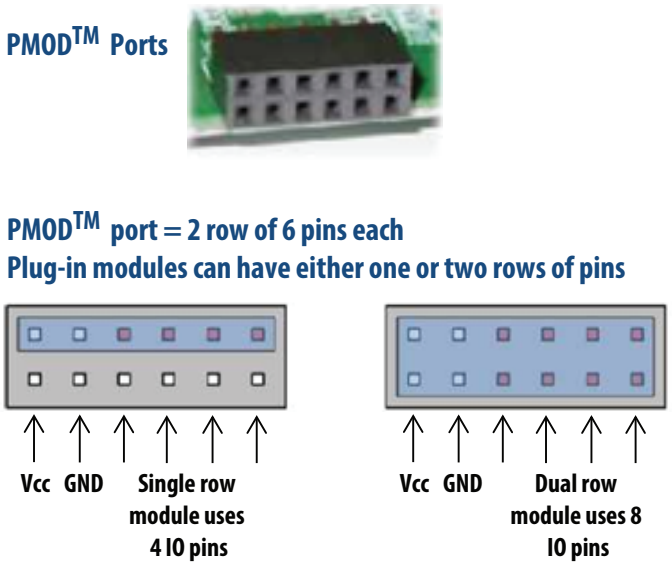


Figure 3. PMOD type 1 port (6-pin configuration)

Figure 4 demonstrates how the 798J PMOD interface type 1 evaluation board is used together with a FPGA development kit.

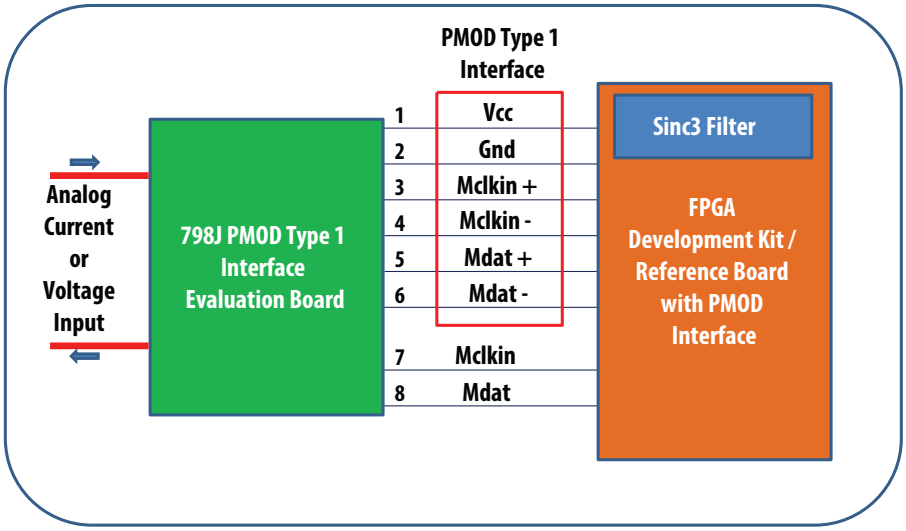


Figure 4. 798J PMOD type 1 evaluation board interface to FPGA development kit

Sinc3 filter can be easily programmed into FPGA in VHDL or Verilog environment. An example of a 16-bit output Sinc3 filter code is provided at the Appendix of this userguide for both VHDL and Verilog. In this example, two pins are assigned to allow selection of three decimation ratio settings, 256, 128 and 64.

If the FPGA is configured for LVDS interface, toggle all pins 1 to 4 of dip switch selector to the left. If FPGA is configured to drive single-ended board clock frequency and to receive single ended data, toggle all pins 1 to 4 of dip switch selector to the right. A pair of LVDS driver and receiver are included in the evaluation board to translates LVDS differential clock and data signals to single ended signal respectively. Please refer to schematic diagram on the last page for illustration.

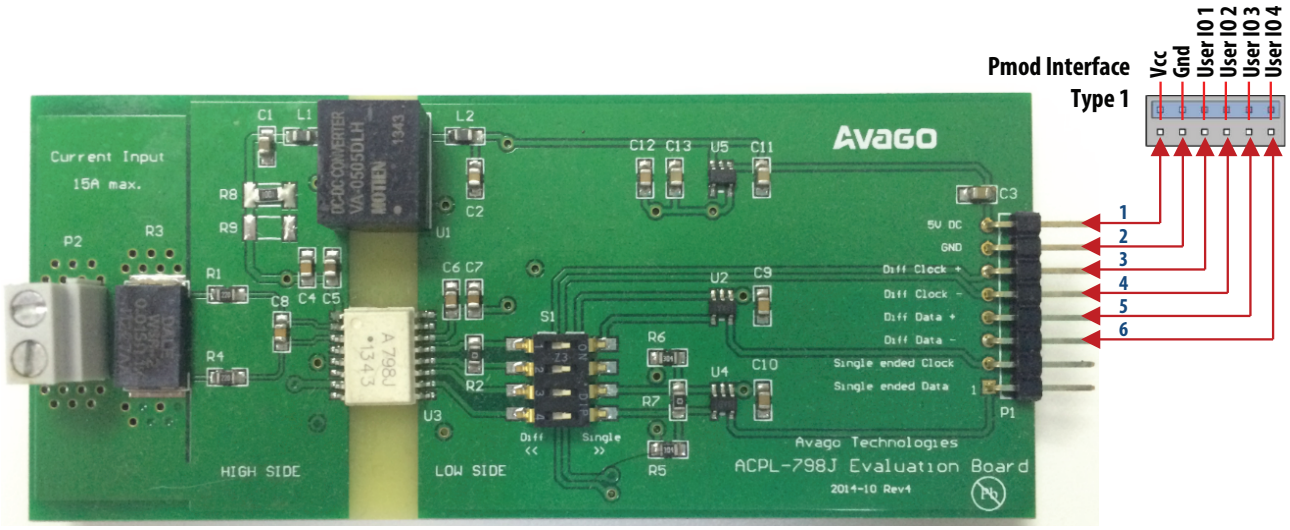


Figure 5. Avago 798J PMOD interface evaluation board output interfacing to PMOD type 1

Output pin configuration of Avago 798J PMOD interface evaluation board

Pin	Evaluation Board	PMOD Type 1 Port
1	5V	Vcc
2	GND	Gnd
3	Differential Clock +	User I/O
4	Differential Clock -	User I/O
5	Differential Data +	User I/O
6	Differential Data -	User I/O

A 10mΩ shunt resistor is included in this evaluation board to demonstrate the current sensor function. It's suitable for current sensing up to 15Arms. For higher current sensing application, choose appropriate shunt resistance value and power rating accordingly.

Shunt Resistor Value Selection

One example to select the shunt resistor value is shown below:

If maximum rms current through motor = 10A, 50% overloads during normal operation, then, peak current is 21.1 A (=10 x 1.414 x 1.5). Recommended max. input voltage for ACPL-798J = ±200mV.

- Shunt resistor value = V/I = 200mV/21.1A ≈ 10mΩ
- Power dissipation = I2*R = (10)2 * 10mΩ = 1W

A list of high precision shunt resistor manufacturers is available at the Appendix.

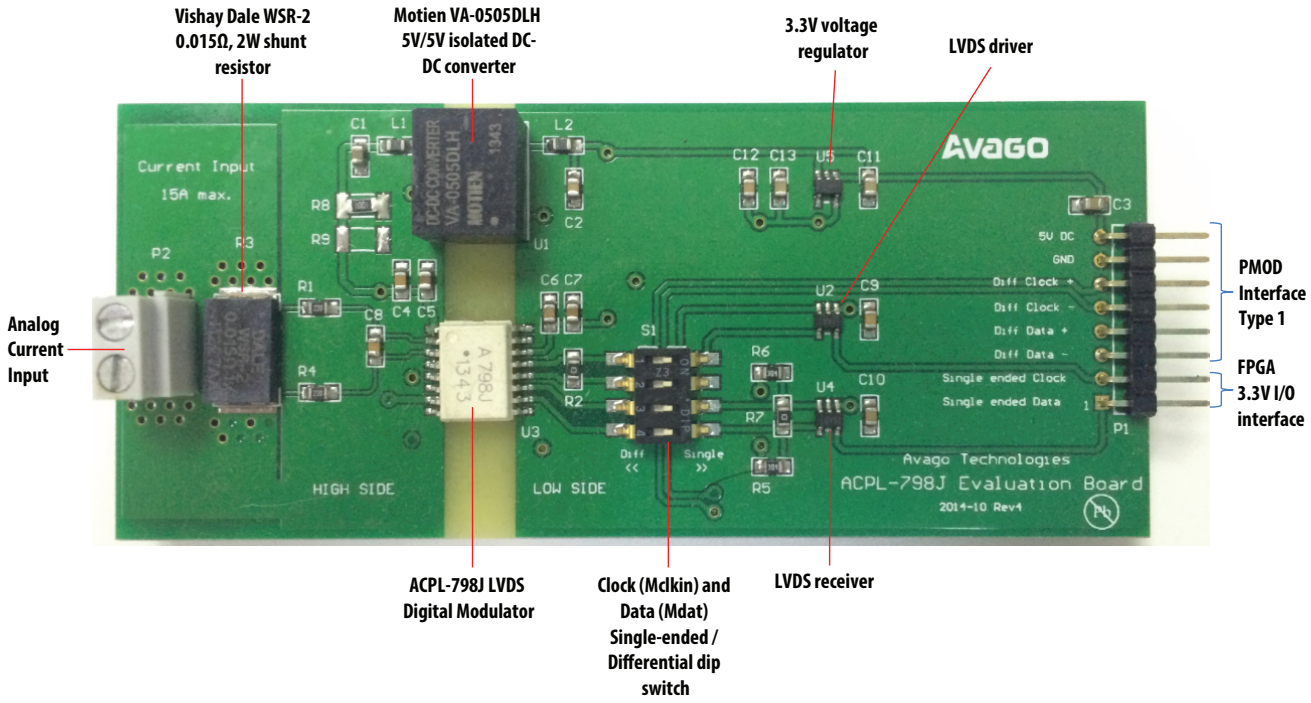
Reference

[1] Datasheet ACPL-798J Optically Isolated Sigma-Delta Modulator with LVDS Interface, publication number AV02-4339EN

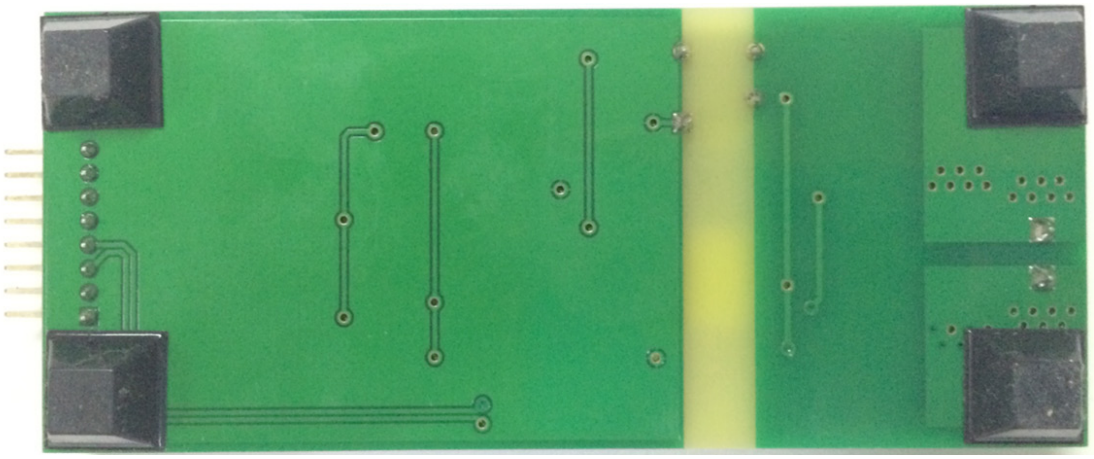
[2] Digilent Pmod™ Interface Specification (PDF), Digilent, Inc., November 20, 2011

APPENDIX A - PCB

Front PCB



Rear PCB



APPENDIX B - 16-bit Output Sinc3 Filter Code

16-bit Output Sinc3 Filter Code with three selectable decimation factors 256, 128 and 64 (hardware pin select)

1. Verilog Code

```
/*-------------------------------------------------------------
          Avago Technologies Confidential
 -------------------------------------------------------------
 -------------------------------------------------------------*/
/*  Create Date:  09/13/2011
    Design Name:  cic_filter
    Module Name:  cic_filter.v

    Description:  CIC filter (SINC3 digital filter)
                  with decimation value 64, 128, 256
    Revision:
    Revision 0.01 - File Created
    Additional Comments:
                                        */

module cic_filter ( clk, reset, sel, filter_in, filter_out, word_clk );

            input clk;      // sigma delta adc clock
            input reset;
            input [1:0] sel; //Control decimation factor 64, 128 and 256
            input filter_in; // sigle bit sigma delta bit stream


            output [15:0] filter_out;  // 16 bit digital filter output
            output word_clk;         // Decimated Clock

//***********************************
////////////////////////////////
//sel = 2'b00 --> Decimation Factor = 256
//sel = 2'b01 --> Decimation Factor = 128
//sel = 2'b10 --> Decimation Factor = 64
//sel = 2'b11 --> Decimation Factor = 256
////////////////////////////////

            wire  [24:0] ip_data1;
            reg   [24:0] acc1;
            reg   [24:0] acc2;
            reg   [24:0] acc3;
            reg   [24:0] acc3_d2;
            reg   [24:0] diff1;
            reg   [24:0] diff2;
            reg   [24:0] diff3;
            reg   [24:0] diff1_d;
            reg   [24:0] diff2_d;
            reg   [15:0] filter_out;
            reg   [7:0] word_count;
            reg   word_clk;

//***********************************

assign ip_data1 = (filter_in == 1'b1)? 25'h1 : 25'h0;
```

Verilog Code (Continued...)

```
//accumulation process
always @(posedge clk or posedge reset)
begin
        if (reset)
        begin
                acc1 <= 0;
                acc2 <= 0;
                acc3 <= 0;
        end
        else begin
                acc1 <= acc1 + ip_data1;
                acc2 <= acc2 + acc1;
                acc3 <= acc3 + acc2;
        end
end

always @(posedge clk or posedge reset)
begin
        if (reset)
                word_count <= 0;
        else begin
                        if(word_count == 8'b11111111)
                                word_count <= 0;
                        else
                                word_count <= word_count + 1;
        end
end

//Decimation Stage
always @(sel or word_count)
begin
                if(sel == 2'd0)
                  if (word_count == 8'b11111111)
                        word_clk <= 1'b1;
                  else
                        word_clk <= 1'b0;
                else if(sel == 2'd1)
                  if (word_count[6:0] == 7'b1111111)
    word_clk <= 1'b1;
  else
    word_clk <= 1'b0;
                else if(sel == 2'd2)
                  if (word_count[5:0] == 6'b111111)
    word_clk <= 1'b1;
  else
    word_clk <= 1'b0;
                else
                        if (word_count == 8'b11111111)
    word_clk <= 1'b1;
  else
    word_clk <= 1'b0;
end
```

Verilog Code (Continued...)

```
//DIFFERENTIATOR
always @(posedge clk or posedge reset)
begin
        if(reset) begin
                acc3_d2 <= 0;
                diff1_d <= 0;
                diff2_d <= 0;
                diff1 <= 0;
                diff2 <= 0;
                diff3 <= 0;
        end
        else begin
         if (word_clk)
           begin
                diff1 <= acc3 - acc3_d2;
                diff2 <= diff1 - diff1_d;
                diff3 <= diff2 - diff2_d;
                acc3_d2 <= acc3;
                diff1_d <= diff1;
                diff2_d <= diff2;
               end
        end
end

// filter_out --> Filtered 16 bit output
always @(posedge clk or posedge reset)
begin
 if (reset)
     filter_out <= 16'h0000;
 else begin
  if (word_clk)
   begin
            if(sel == 2'd0) begin        //Decimation ratio 256
               if (diff3[24] ==1'b1)
                                filter_out <= 16'hffff;
                        else
                                filter_out <= diff3[23:8];
             end
            else if(sel == 2'd1) begin    //Decimation ratio 128
                        if (diff3[21] ==1'b1)
                                filter_out <= 16'hffff;
                        else
                                filter_out <= diff3[20:5];
            end
            else if(sel == 2'd2) begin    //Decimation ratio 32
                        if (diff3[18] == 1'b1)
                                filter_out <= 16'hffff;
                        else
                                filter_out <= diff3[17:2];
            end
            else begin
                        if (diff3[24] ==1'b1)      //Decimation ratio 256
                                filter_out <= 16'hffff;
                        else
                          filter_out <= diff3[23:8];
            end
           end
        end
end

endmodule
```

2. VHDL Code

```
----------------------------------------------------------------------------
-- Avago Technologies Confidential
----------------------------------------------------------------------------
--
-- Author:WongCH
--
-- Create Date:   31-05-2011
-- Design Name:   filter
-- Module Name:   filter.vhd
-- Project Name:
-- Target Device:
-- Tool versions:
-- Description:   SINC3 digital filter
--
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity filter is
   Port ( clk              : in  STD_LOGIC;
          reset            : in  STD_LOGIC;
                   mdata            : in         std_logic;
                   setting          : in         std_logic_vector(7 downto 0);
                   word_clk         : out std_logic;
                  fil_data          : out std_logic_vector(15 downto 0)
);
end filter;

Architecture rtl of filter is

signal ipdata1     : std_logic_vector(24 downto 0);
signal acc1        : std_logic_vector(24 downto 0);
signal acc2        : std_logic_vector(24 downto 0);
signal acc3        : std_logic_vector(24 downto 0);
signal acc3_d2     : std_logic_vector(24 downto 0);
signal diff1       : std_logic_vector(24 downto 0);
signal diff2       : std_logic_vector(24 downto 0);
signal diff3       : std_logic_vector(24 downto 0);
signal diff1_d     : std_logic_vector(24 downto 0);
signal diff2_d     : std_logic_vector(24 downto 0);
signal data        : std_logic_vector(15 downto 0);
signal word_count  : std_logic_vector(7 downto 0);
signal i_word_clk  : std_logic;


begin


p_mdata: process(mdata)
begin
if mdata = '0' then
            ipdata1 <= "0000000000000000000000000";
else
            ipdata1 <= "0000000000000000000000001";
end if;
end process;
```

VHDL Code (Continued...)

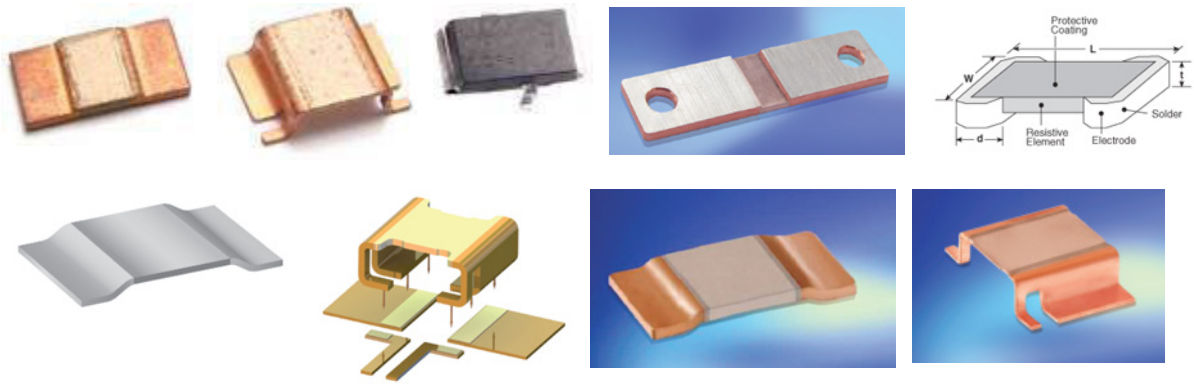
```
p_acc: process(reset, clk)
begin
if (reset = '1') then
        acc1 <= (others => '0');
        acc2 <= (others => '0');
        acc3 <= (others => '0');
elsif (clk ='1' and clk'event) then
        acc1 <= acc1 + ipdata1;
        acc2 <= acc2 + acc1;
        acc3 <= acc3 + acc2;
end if;
end process;

p_dec_clk: process(reset, clk)
begin
        if (reset = '1') then
                word_count <= (others => '0');
        elsif (clk ='1' and clk'event) then
                word_count <= word_count + '1';
        end if;
end process;

process(word_count, setting)
begin
 if setting(7 downto 4) = "1111" then
   case setting(1 downto 0) is
    when "00" =>
      if word_count(7 downto 0) = "11111111" then
          i_word_clk <= '1';
      else
          i_word_clk <= '0';
      end if;
    when "01" =>
                                    if word_count(6 downto 0) = "1111111" then
          i_word_clk <= '1';
      else
          i_word_clk <= '0';
      end if;
                                when "10" =>
                                  if word_count(5 downto 0) = "111111" then
          i_word_clk <= '1';
      else
          i_word_clk <= '0';
      end if;
                                when Others =>
                                  if word_count(4 downto 0) = "11111" then
          i_word_clk <= '1';
      else
          i_word_clk <= '0';
      end if;
                end case;
         else
             i_word_clk <= '0';
         end if;
end process;

word_clk <= i_word_clk;

p_diff: process(reset, clk)
```

VHDL Code (Continued...)

```
begin
        if (reset = '1') then
                        acc3_d2 <= (others =>'0');
                        diff1_d <= (others =>'0');
                        diff2_d <= (others =>'0');
                        diff1   <= (others =>'0');
                        diff2   <= (others =>'0');
                        diff3   <= (others =>'0');
        elsif (clk ='1' and clk'event) then
                        if i_word_clk ='1' then
                                        acc3_d2 <= acc3;
                                        diff1 <= acc3 -acc3_d2;
                                        diff1_d <= diff1;
                                        diff2 <= diff1 -diff1_d;
                                        diff2_d <= diff2;
                                        diff3 <= diff2 -diff2_d;
                        end if;
        end if;
        end process;

p_data: process(reset, clk)
begin
        if (reset = '1') then
                        data <= (others =>'0');
        elsif (clk ='1' and clk'event) then
                        if i_word_clk ='1' then
                             if setting(7 downto 4) = "1111" then
       case setting(1 downto 0) is
          when "00" =>    --decimation ratio= 256
           if diff3(24) ='1' then
              data <= (others =>'1');
           else
              data <= diff3(23 downto 8);
           end if;
          when "01" =>  --decimation ratio = 128
           if diff3(21) ='1' then
              data <= (others =>'1');
           else
              data <= diff3(20 downto 5);
           end if;
          when "10" =>  --decimation ratio =64
           if diff3(18) ='1' then
              data <= (others =>'1');
           else
              data <= diff3(17 downto 2);
           end if;
          when others => --decimation ratio = 32
           if diff3(15) ='1' then
              data <= (others =>'1');
           else
              data <= diff3(14 downto 0) & '0';
           end if;
         end case;
      else
         data <= "0000000000000000";
      end if;
                                 end if;
          end if;
end process;

fil_data <= data;

end rtl;
```

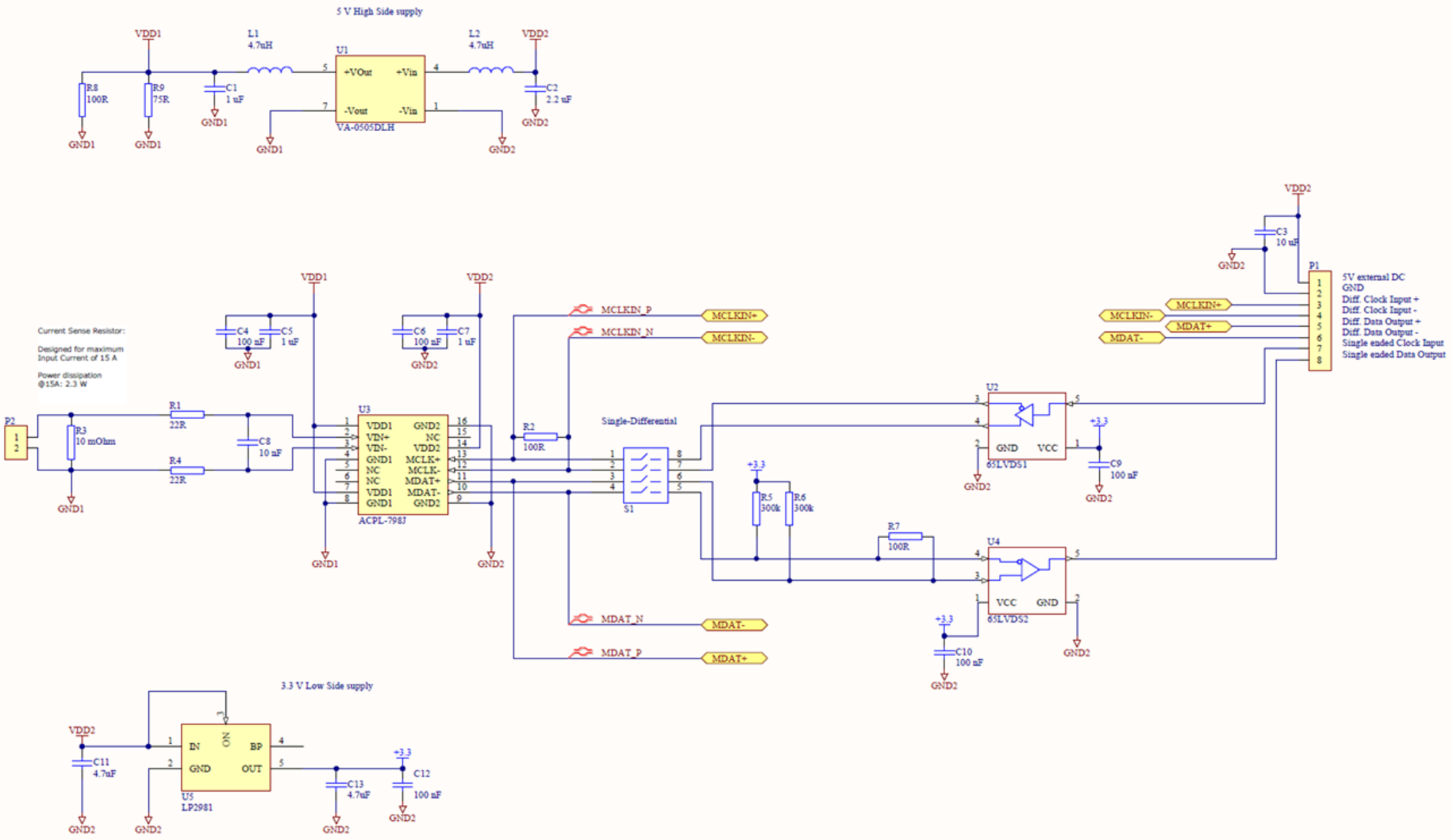
APPENDIX C - Shunt Resistor Manufacturer

1) KOA {http://www.koanet.co.jp}
2) Micron Electric (Japan) {http://www.micron-e.co.jp/}
3) International Resistive Company (IRC) {http://www.irctt.com/}
4) Isabellenhuette Isotek {http://www.isotekcorp.com/about-us/isabellenhutte}
5) Precision Resistor {http://www.precisionresistor.com}
6) Vishay-Dale {http://www.vishay.com/videos/resistors/vishay-dale-shunt-resistors-an-overview}



The above pictures show different types of high precision shunt resistors with different resistance values, tolerance and power dissipations offered by the manufacturers listed above.

APPENDIX D - Schematic Diagram



Author: Lim Shiun Pin, Avago Isolation Product Division Application Engineer

For product information and a complete list of distributors, please go to our web site: **www.avagotech.com**

AV02-4961EN - June 23, 2015