Lattice TI ADC Demo

User's Guide

January 2008
UG04_01.0

Introduction

This design demonstrates the ability of the LatticeECP2™ FPGA to interface to the Texas Instruments (TI) ADS644X and ADS642X family of ADC ICs using the TI ADS6XXX-EVM (e.g. ADS6245EVM), LatticeECP2 Advanced evaluation board, and an interface card that connects them.

Demo Overview

The demo design uses the LatticeECP2 Advanced Evaluation Board, the LatticeECP2 Advanced - TI ADS6000 EVM Interface Card (Lattice-TI Interface Card), and one of the TI ADS6XXX-EVM evaluation boards. An input clock signal and the analog input signal are both connected to the TI ADS EVM board. The TI ADS EVM board will convert the analog signal to a digital value and send this value to the LatticeECP2 FPGA using a single LVDS data signal and two LVDS clock signals. The data is sent serially over the single data line. One of the clock signals is a bit clock to read the individual data bits in the serial data line. The other clock is a Frame clock to mark the start of a data word or frame. The TI ADS EVM board will generate the bit clock and Frame clock from the input clock signal.

In this design, the LatticeECP2 FPGA reads the value of the digital signal from the TI ADS EVM board using the LVDS data and clock signals. The input data is captured by the LatticeECP2 FPGA using the built-in DDR registers and the 2X gearing function. The Frame clock is adjusted as required using the onboard PLL of the LatticeECP2 FPGA. The logic stores the first 1020 data samples into the onboard EBR memory block so they can be read out later. The design also incorporates the Lattice Reveal Logic Analyzer tool to allow the user to easily read out data values from the memory.

The TI ADS6xxx EVM boards are available in two versions. One is capable of a 12-bit analog-to-digital conversion and the other version is capable of 14-bit analog-to-digital conversion. This Demo is built for versions which use a 12-bit conversion. The TI ADS board is also available in either 2 or 4 channel versions. This Demo was developed using the 4-channel version of the TI ADS board, though using the 2-channel board is also possible.

The TI ADS6245EVM board can be set up to use either a single wire serial interface or a 2-wire serial interface. The 2-wire interface is recommended when using sample rates above 65 MSPS. When using the 2-wire interface, the data samples are split across both serial lines so that the bit clock frequency can be lower. The demo design uses the 1-wire interface and a sample rate of 60 MSPS. In this configuration, the data rate for the single serial wire is equal to or greater than the data rate for each of the 2 wires at the higher data sample rates.

Please see the TI ADS6XXX-EVM Board User Guide and the ADS6425 ADC Data Sheet for further details about the capabilities of the TI ADC devices.

Please see EB23, *LatticeECP2 Advanced Evaluation Board User's Guide,* for additional information about the evaluation board operation.

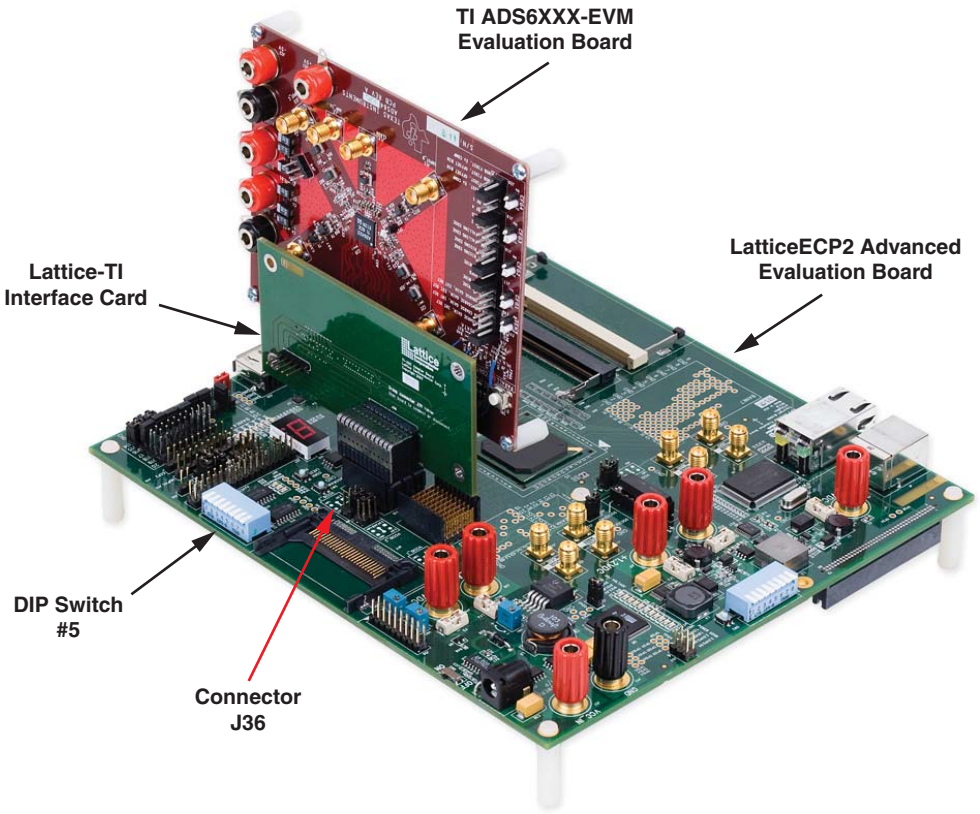
Test Setup

The demo design uses the LatticeECP2 Advanced Evaluation board, the Lattice-TI Interface Card, and the TI ADS6245-EVM evaluation boards. An analog signal is connected to the TI ADS board channel B, using SMA connector J11. An input clock signal is connected to the TI ADS board using SMA connector J12.

The TI ADS6245EVM board is plugged into the Lattice-TI Interface Card at connector J15 and this assembly is plugged into connector J36 on the LatticeECP2 Advanced Evaluation board. The connector number of the Lattice-TI Interface Card matches the mating connector on the board it connects to. Therefore, connector J15 on the Lattice-TI Interface Card connects to J15 on the TI ADS board, and connector J36 connects to J36 on the LatticeECP2 Advanced evaluation board.

The LatticeECP2 Advanced evaluation board and the Lattice-TI Interface Card both have a North indicator in the silkscreen just below the Lattice logo. When this North indicator is pointed up as you view the LatticeECP2 Advanced Evaluation board, the Lattice-TI Interface Card should be aligned so its J36 connector is facing toward the user's right. In this configuration, the TI ADS board component side is also facing to the user's right when it is plugged into the Lattice-TI Interface Card. A picture of the correct installation orientation is shown in Figure 1.

Figure 1. Physical Arrangement of the TI ADS6245EVM Board and LatticeECP2 Advanced Evaluation Board



The TI ADS6245EVM board uses jumpers to set various features on the board. The jumpers are located along the left edge of the board when viewed with the silkscreen text readable. The jumper settings should be adjusted to match the following:

J16	0dB Coarse gain, INT ref.	(default setting)
J17	DDR, 1 Wire	(default is DDR, 2 wire)
J18	14X, Rising edge	(default setting)
J19	DIV by 1	(default setting)
J20	MSB first, Offset Bin	(default is MSB first, 2's Comp)

The TI ADS6245EVM board will require a 3.3V power supply for this demo to function. The board also has +5V and -5V connectors which are not used for this demo. The +3.3V will be plugged into both the P1 and P3 connectors. P1 is the digital power source and P3 is the analog power source for the board, these can be connected together. The ground connections, P2 and P4 can also be connected together.

It is recommended to plug in the power cables to the TI ADS board before it is connected to the Lattice-TI Interface Card and the LatticeECP2 Evaluation board.

The pin assignments and a schematic of the Lattice-TI Interface Card are shown in Appendix A of this user's guide.

This design uses DIP switch #5 on the LatticeECP2 Evaluation board for different settings. The DIP switch #5 signal assignments are listed below.

Function Name	Switch Assignment	Demo Setting
Read enable	pos. 8	varies
PLL dphase[0]	pos. 7	down
PLL dphase[1]	pos. 6	down
PLL dphase[2]	pos. 5	down
PLL dphase[3]	pos. 4	Up
N/a	pos. 3	not used
N/a	pos. 2	not used
Reset	pos. 1	down

The Reset switch will reset the FPGA logic when it is in the up position. After the reset has been accomplished, the switch should be set in the down position.

The Read enable switch is used to restart the read address counter. When the switch is set in the down position, the counter is reset. The switch should be moved to the up position to read the data from the FPGA.

The TI ADS6245EVM board will need an input clock signal of the same frequency as the sample rate. If the user wants the sample rate to be 60 MSPS, then the input clock must be 60 MHz. The clock signal must meet the requirements specified in the TI data sheet. The recommended clock source is a 1.5 VRMS sine-wave with a 0V offset. Other options are supported as shown in the TI data sheet.

The recommended analog input signal is 1V amplitude sine wave at 1/10th of the sample frequency (i.e., 6 MHz using 60 MSPS) and with a 0V offset. This signal should give an acceptable amplitude for the initial readings. The magnitude of the signal will vary with amplitude of the sine wave. Using this frequency ratio will give 10 samples per period of the sine wave so it should be easy to observe the basic shape of the input signal. For additional resolution of the input sine wave, the analog input signal frequency can be reduced to 1/20th of the sample frequency (i.e., 3 MHz). This is the ratio used for the sample data graphs shown in Figures 3 and 5 of this user's guide.

Operation

Connect the power cables to the TI ADS6245EVM board and verify that the jumpers on the card are set as shown above. Do NOT power on the TI ADS board at this time. Connect the Lattice-TI Interface Card to the TI ADS board. Then plug this assembly into the LatticeECP2 Advanced Evaluation board using connector J36, being sure to use the alignment described above. The arrangement should look like Figure 1.

Connect the clock input signal to the TI ADS board using SMA connector J12. Connect the analog input signal to the TI ADS board using SMA connector J11, channel B. (If you are using a 2-channel TI ADS6245EVM board, use SMA connector J10, channel A, when using the demo bitstream file).

Plug in a USB version of the Lattice ispDOWNLOAD® cable to the LatticeECP2 Advanced Evaluation board and your PC. Plug in the power adapter to the LatticeECP2 Evaluation board and turn on the power switch. Download the demo bitstream file to the LatticeECP2 device. Now power up the TI ADS board and turn on the analog input and clock sources. Put DIP switch #5, position 1, in the up position to initiate a reset of the design and then push it back down to run the design. Leave SW #5, position 8, in the down position for now.

The design will store the first 1020 data samples in an EBR memory block. The Reveal logic analyzer will be used to read some of the sample values back out of the memory. The memory read address is generated from a free running counter which will roll over automatically so that it is always reading from addresses 0-1023. Which values are read are a function of the address counter at the time that Reveal runs a trigger to sample the data in memory.

Start ispLEVER® on your PC and then open the demo design. Start Reveal and open the demo project (adc_serial12_ddr.rva). The Reveal demo project is set up for 20 triggers with 8 samples per trigger. If all 20 trigger values are not visible when Reveal is started you can change the zoom setting of the display. Run the Reveal Logic Analyzer by clicking the **RUN** icon on the toolbar or by using the menu. The data capture will start when read enable is set high. This is done by moving SW #5, position 8, to the UP position. The data value in read address 0

and address 1 will usually be empty. The remaining values should be correct. Additional runs of the logic analyzer will read data values from different memory addresses.

The memory read address is generated by a free running counter that will roll over automatically. There is not a way to read specific addresses from memory. Each run of the Reveal logic analyzer will read the values out of the memory location that is currently in the read address counter. If SW #5, position 8, is moved to the down position, the read memory address counter will be reset, but the data values in memory will not be changed. Moving SW #5, position 8, back to the up position will cause the design to begin reading data values out of memory again beginning with address zero. In order to change the data samples that are in memory, use the Reset switch, SW #5, position 1, to initiate a system reset.

The data is read out of the memory using a clock signal generated by a second PLL on the LatticeECP2 FPGA. This PLL uses the oscillator chip on the LatticeECP2 Advanced Evaluation board as its input signal. Therefore, the oscillator must be in place and functioning for the demo design to read the data out of the memory properly. The data is stored into the memory using the Frame clock signal from the TI ADS board.

Additional Design Notes

The demo design reads the data inputs on the transitions of the bit-clock, called bitclk in the design, using the built-in DDR registers and the 2X gearing. This is done using a generic DDR input block which is generated using IP Express and then instantiated in the design. Four data bits are read out of the DDR input registers and into a shift register to store them until all the data values are present and then the Frame clock signal will cause the values to be read out of the shift register and stored into the memory.

The Frame clock signal from the TI ADS6245EVM board is used as the input to a PLL and the CLKOS output of the PLL is used to create a shifted Frame clock, called Frame_S. This Frame_S signal is used to read the data into the shift register and into the memory as well. The PLL gives the user the ability to shift the Frame_S clock signal using the dynamic phase settings, which are set using the DIP switch on the LatticeECP2 Advanced Evaluation board. The shifted Frame_S clock signal must be properly aligned with the data in the shift register to read the correct data out.

If it becomes necessary to change the dynamic phase setting, it is recommended to use the SYNC test pattern output of the TI ADS6245EVM board. The SYNC test pattern for the TI ADS6245EVM is 111111000000 in the 12 bit versions of the TI ADS6245EVM board. For the Demo bitstream, the setting shown above (dphase[3:0] = 1000) has been tested and shown to produce good results. Adjusting this setting should only be necessary if the design has been changed significantly.

Once the dynamic phase setting has been adjusted, the TI ADS6245EVM board can be returned to the normal operating mode and the input data sampled.

Results

The results of several different demo runs are shown below. The values are displayed using the Lattice Reveal Logic Analyzer tool. The datout signal is shown highlighted in Figures 2 and 4. These figures show typical results that may be obtained when using this demo.

The input signal used was a sine wave of fixed magnitude for each run. The input sine wave is not synchronized to the sample clock, so the sample window can be at any location along the sine wave. Hence, the sample sine wave will appear to be phase shifted depending upon where the sample began. Included in Figures 3 and 5 are some Excel graphs of data that shows the input sine wave pattern for various test runs. Also included in these graphs is a calculated sine wave for comparison purposes.

There were a total of 16 different data sets collected using sine wave inputs at varying frequencies and amplitudes. In Table 1, the conditions for these 16 test runs are shown.

Table 1. Test Run Conditions

Test Run #	Sample Rate	Input Frequency	Input Amplitude
1	60 MSPS	6 MHz	725 mV
2	60 MSPS	6 MHz	725 mV
3	60 MSPS	6 MHz	860 mV
4	60 MSPS	6 MHz	860 mV
5	60 MSPS	3 MHz	810 mV
6	60 MSPS	3 MHz	810 mV
7	60 MSPS	3 MHz	910 mV
8	57.5 MSPS	2.875 MHz	900 mV
9	57.5 MSPS	2.875 MHz	900 mV
10	57.5 MSPS	2.875 MHz	1.11 V
11	57.5 MSPS	2.875 MHz	1.11 V
12	57.5 MSPS	2.875 MHz	1.39 V
13	56 MSPS	2.8 MHz	1.51 V
14	56 MSPS	2.8 MHz	1.51 V
15	56 MSPS	2.8 MHz	1.68 V
16	56 MSPS	2.8 MHz	1.68 V

Figure 2. Figure 2. Test #7: Using a sine wave input at 3 MHz with a 60 MSPS sample rate. Sine wave amplitude is 910 mVolts as measured with oscilloscope.

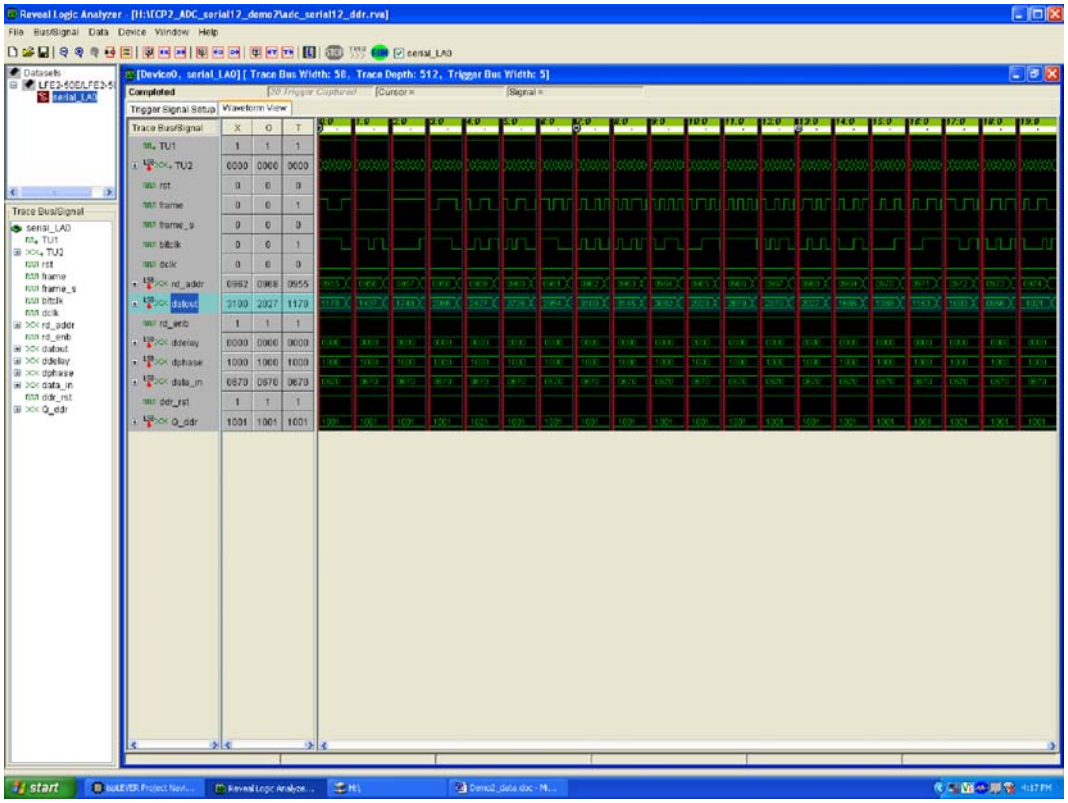


Figure 3. Graph of results for tests 5, 6, and 7. All tests at 60 MSPS - Amplitudes varied (calculated cosine-wave generated in Excel for comparison).

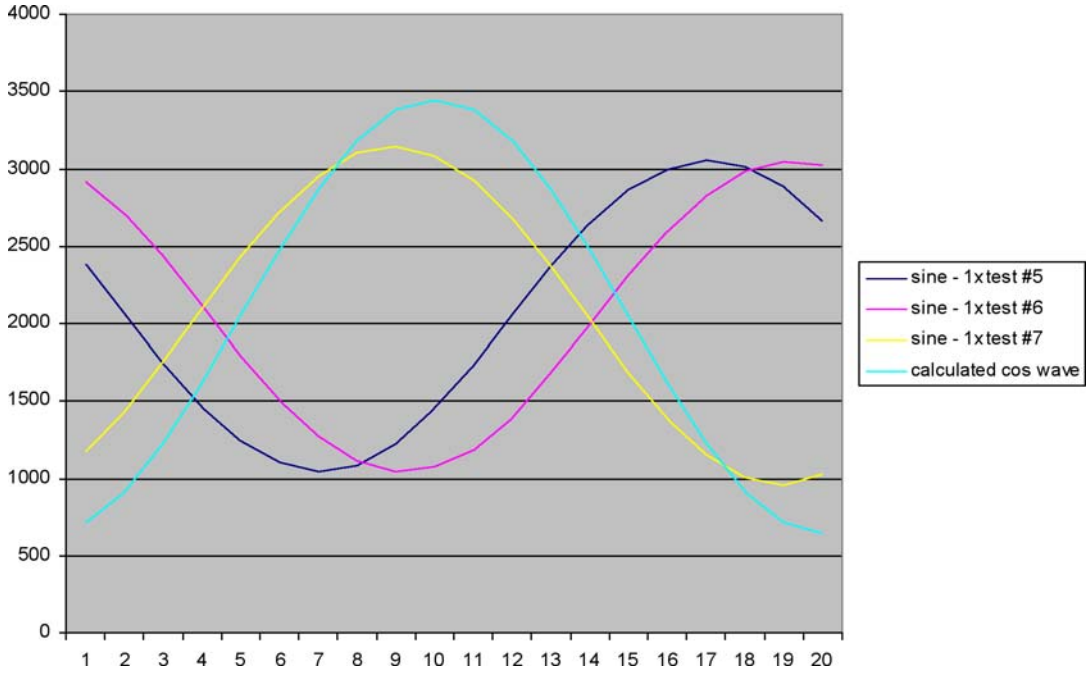


Figure 4. Test #14 Using a sine wave input at 2.8 MHz with a 56 MSPS sample rate. Sine wave amplitude is 1.51V as measured with oscilloscope.

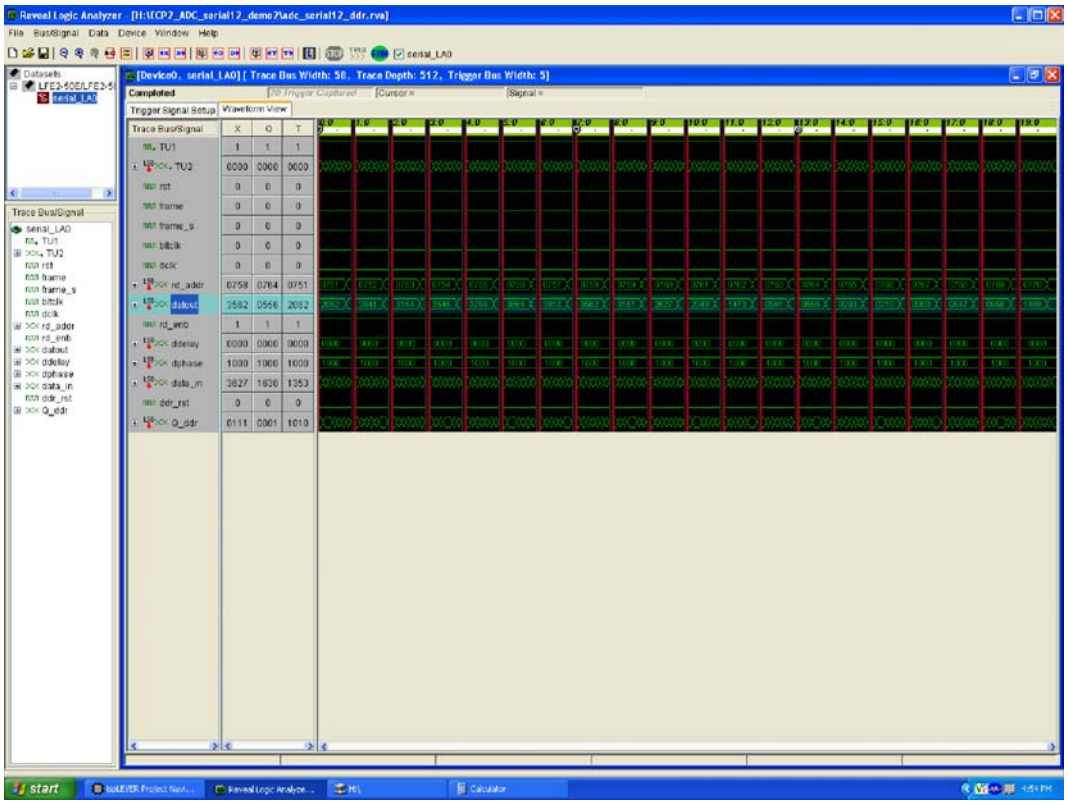
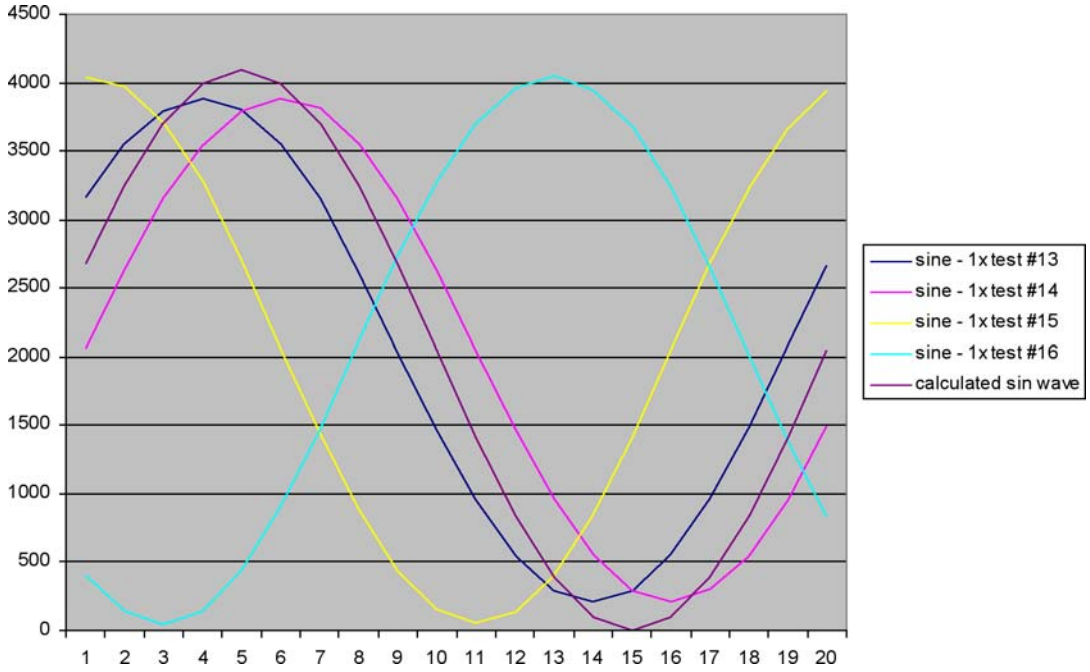


Figure 5. Graph of results for tests 13 through 16. All tests at 56 MSPS - Amplitudes varied. (Calculated sine wave generated in Excel for comparison).



Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Revision History

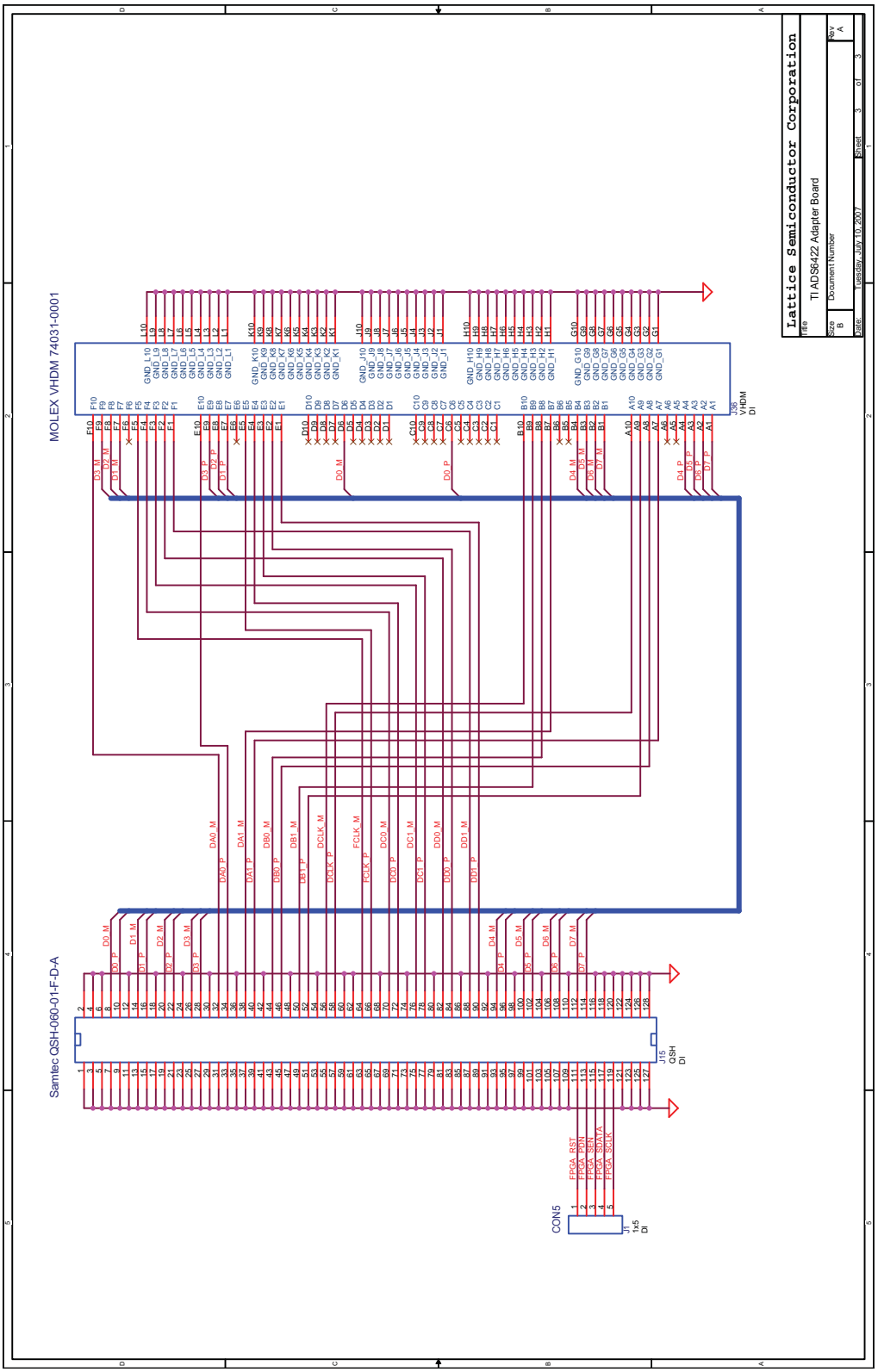
Date	Version	Change Summary
January 2008	01.0	Initial release.

Appendix A. Lattice-TI Interface Card Pin Assignments and Schematic

Table 2. Pin Assignments for the Lattice-TI Interface Card

J36 Connector	LatticeECP2 Pin	Signal Name	T/C	Dual Function	TI Signal Name	J15 Connector
A1	R23	PR55A	T		D7_P	114
B1	R24	PR55B	C		D7_M	112
A2	T21	PR57A	T		D6_P	108
B2	R22	PR57B	C		D6_M	106
A3	R21	PR51A	T		D5_P	102
B3	N22	PR51B	C		D5_M	100
A4	T22	PR59A	T		D4_P	96
B4	T20	PR59B	C		D4_M	94
E1	P19	PR53A	T		DD1_P	90
F1	P21	PR53B	C		DD1_M	88
E2	P25	PR56A	T		DD0_P	84
F2	P26	PR56B	C		DD0_M	82
E3	R25	PR58A	T	RDQS58	DC1_P	78
F3	R26	PR58B	C		DC1_M	76
E4	U20	PR61A	T	GDLLT_FB_A	DC0_P	72
F4	T19	PR61B	C	GDLLC_FB_A	DC0_M	70
E5	T26	PR60A	T	GDLLT_IN_A	FCLK_P	66
F5	T25	PR60B	C	GDLLC_IN_A	FCLK_M	64
A10	U25	PR63A	T	GPLLT_IN_A	DCLK_P	58
B10	U24	PR63B	C	GPLLC_IN_A	DCLK_M	56
A9	W26	PR67A	T	RDQS67	DB1_P	52
B9	W25	PR67B	C		DB1_M	50
A8	Y26	PR69A	T		DB0_P	46
B8	AA26	PR69B	C		DB0_M	44
A7	U26	PR65A	T		DA1_P	40
B7	V26	PR65B	C		DA1_M	38
E10	V23	PR70A	T		DA0_P	34
F10	W24	PR70B	C		DA0_M	32
E9	U19	PR68A	T		D3_P	28
F9	U21	PR68B	C		D3_M	26
E8	V25	PR66A	T		D2_P	22
F8	V24	PR66B	C		D2_M	20
E7	U23	PR64A	T	GPLLT_FB_A	D1_P	16
F7	U22	PR64B	C	GPLLC_FB_A	D1_M	14
C6	N21	PR47A	T	VREF1_3	D0_P	10
D6	N18	PR47B	C	VREF2_3	D0_M	8
D10	N19	PR49B	C			
C10	N20	PR49A	T			
C5	M24	PR42B		BANK 2		

Figure 6. Schematic



Appendix B. Demo Reference Design Verilog Sample Code

```
//
// DEMO - Serial ADC interface to Lattice ECP2/M FPGA
// ----------------------------------------------------------------------
// >>>>>>>>>>>>>>>>>>>>>>>> COPYRIGHT NOTICE <<<<<<<<<<<<<<<<<<<<<<<<<
// ----------------------------------------------------------------------
// Copyright (c) 2007 by Lattice Semiconductor Corporation
// ----------------------------------------------------------------------
//
// Permission:
//
//   Lattice Semiconductor grants permission to use this code for use
//   in synthesis for any Lattice programmable logic product.  Other
//   use of this code, including the selling or duplication of any
//   portion is strictly prohibited.
//
// Disclaimer:
//
//   This VHDL or Verilog source code is intended as a design reference
//   which illustrates how these types of functions can be implemented.
//   It is the user's responsibility to verify their design for
//   consistency and functionality through the use of formal
//   verification methods.  Lattice Semiconductor provides no warranty
//   regarding the use or functionality of this code.
//
// ----------------------------------------------------------------------
//
//                     Lattice Semiconductor Corporation
//                     5555 NE Moore Court
//                     Hillsboro, OR 97214
//                     U.S.A
//
//                     TEL: 1-800-Lattice (USA and Canada)
//                          (503)268-8001 (other locations)
//
//                     web:   http://www.latticesemi.com
//                     email: techsupport@latticesemi.com
//
// ----------------------------------------------------------------------
//
//  This is the top level module of the:
//  Serial ADC interface to Lattice ECP2/M FPGA Demo design using:
//
//     12 bit Data input from Analog to digital converter using a
//     serial single channel. Uses generic DDR input module with 2X Gearing
//
//  Supported devices for implementing this design:
//    * LatticeECP2/M
//    * LatticeXP2    (not tested yet)
//
//
// ----------------------------------------------------------------------
// $Header:
//
// Revision History :
// ----------------------------------------------------------------------
//   Ver  :| Author            :| Mod. Date :| Changes Made:
//   V1.0 :| J.D.              :| 11/27/07  :| Initial Release
```

```
// --------------------------------------------------------------------


module serial(rst, clki, bitclk, frame, data_sd0, rd_enb, dphase, clock1, clock2
       , lock_pll, clkop, frame_s, datout, rd_addr, wr_enable, dclk, Q_ddr  );

input  rst, clki, bitclk, frame ;
input  data_sd0, rd_enb ;
input  [3:0] dphase ;

output clock1, clock2 ;
wire   clock1, clock2 ;

output lock_pll, clkop, frame_s ;
output [11:0] datout ;
output [9:0] rd_addr ;

output wr_enable ;
reg    wr_enable ;

output dclk ;
output [3:0] Q_ddr ;

reg    rst_pll ;
reg    reset_pulse ;
reg    ddr_rst ;
reg    [11:0] data_in ;

reg    [11:0] datareg0 ;

//reg    [11:0] datareg1, datareg2, datareg3 ;
//reg    [11:0] datareg4, datareg5, datareg6, datareg7 ;

reg    [9:0] wr_addr ;
reg    [9:0] rd_addr ;
reg    [3:0] rd_addr_delay ;



/* Verilog module instantiation template generated by SCUBA ispLever_v70_SP1_Build (25) */
/* Module Version: 3.6 */  /* Fri Nov 16 19:44:08 2007 */
/* parameterized module instance */
PLL_frame PLL_frame (.CLK(frame), .RESET(rst_pll), .DPAMODE(1'b1), .DPHASE0(dphase[0]),
.DPHASE1(dphase[1]),
    .DPHASE2(dphase[2]), .DPHASE3(dphase[3]), .CLKOP(clkop), .CLKOS(frame_s), .LOCK(lock_pll));


/* parameterized module instance */
ecp2_pll ecp2_pll1 (.CLK(clki), .RESET(rst_pll), .CLKOP(clock1), .CLKOK(clock2), .LOCK( ));


/* Verilog module instantiation template generated by SCUBA ispLever_v70_Prod_Build (55) */
/* Module Version: 4.1 */   /* Wed Aug 15 11:34:57 2007 */
/* parameterized module instance */
RAM_dp RAM_u1 (.WrAddress(wr_addr), .RdAddress(rd_addr), .Data(datareg0), .RdClock(clock1),
    .RdClockEn(rd_enb), .Reset(rst), .WrClock(frame_s), .WrClockEn(1'b1), .WE(wr_enable),
.Q(datout));
```

```
//  PLL reset pulse at startup or if PLL loses lock
always @(posedge clki)
     begin
    if (lock_pll)
       reset_pulse = 1'b0 ;
    else
       if (reset_pulse )
        rst_pll = 1'b0 ;
       else
        begin
        rst_pll = 1'b1 ;
        reset_pulse = 1'b1 ;
        end
     end


/* Verilog module instantiation template generated by SCUBA ispLever_v70_SP1_Build (25) */
/* Module Version: 3.3 */  /* Fri Nov 16 10:44:12 2007 */
/* parameterized module instance */
DDR_gen DDRin1 (.EClk(bitclk), .SClk(dclk), .Rst(ddr_rst), .Data(data_sd0), .Q(Q_ddr));


//   Shift register to capture the serial data from generic DDR module on positive edge dataclk
always @(posedge dclk or posedge rst)
    begin
       if (rst)
        begin
        data_in = 12'b000000000000 ;
        end
       else
        begin
        data_in[11] = data_in[7] ;//  shift register to accumulate data bits
        data_in[10] = data_in[6] ;
        data_in[9]  = data_in[5] ;
        data_in[8]  = data_in[4] ;
        data_in[7]  = data_in[3] ;
        data_in[6]  = data_in[2] ;
        data_in[5]  = data_in[1] ;
        data_in[4]  = data_in[0] ;
        data_in[3]  = Q_ddr[0] ;//  add new data bit into shift register
        data_in[2]  = Q_ddr[2] ;//  first data bit read is Q_drr[0]
        data_in[1]  = Q_ddr[1] ;//  2nd bit read is Q_drr[2], 3rd bit read is Q_drr[1]
        data_in[0]  = Q_ddr[3] ;//  last data bit read is Q_drr[3]
               //  this bit order is due to mapping used by
               //  IP Express when module is generated - mapping shown below:
               //  .QA0(Q[0]), .QA1(Q[1]), .QB0(Q[2]), .QB1(Q[3])
        end
    end


//   data capture at frame signal
always @(posedge frame_s or posedge rst)
    begin
       if (rst)
        begin
        datareg0 = 12'b000000000000 ;
        end
       else
        begin
```

```
        datareg0 = data_in ;//  add new data into shift register
        end
    end

//   memory write address counter
always @(posedge frame_s or posedge rst)
    if (rst)
        wr_addr  = 9'b000000000 ;//  initialize the memory write address counter
    else
       begin
        if (wr_addr == 10'b1111111100)//  disable write enable when address = 1111111100
            wr_enable = 1'b0 ;//  to prevent overwriting data in memory
        else
            if (ddr_rst)
               wr_enable = 1'b0 ;
            else
               begin
                wr_enable = 1'b1 ;
                wr_addr = wr_addr + 1 ;//  increment the address counter
               end
       end


//   memory read address counter
always @(posedge frame_s or posedge rst)
    begin
       if (rst)
        begin
        rd_addr_delay = 4'b0000 ;//  initialize the delay counter
        rd_addr  = 10'b0000000000 ;//  initialize the memory read address counter
        end
       else
        if (rd_enb)
           begin
           rd_addr_delay = rd_addr_delay + 1 ;//  increment delay counter
            if (rd_addr_delay == 4'b0110 )//  when delay reaches preset value
                   rd_addr = rd_addr + 1 ;//  increment the read address counter
            else
               rd_addr = rd_addr ;//  else hold present value
           end
        else
           begin
           rd_addr_delay = 4'b0000 ;//  reset the read address delay counter when rd_enb is '0'
           rd_addr  = 10'b0000000000 ;//  reset the memory read address counter when rd_enb is '0'
           end
    end


//  DDR module reset -  hold the DDR module in reset until the PLL_frame is locked
always @(posedge frame or posedge rst)
    if (rst)
       ddr_rst = 1'b1 ;
    else
       if (lock_pll)
        ddr_rst = 1'b0 ;
       else
        ddr_rst = 1'b1 ;

endmodule
```