Lattice Semiconductor Corporation

Analog-to-Digital Converter Interface

April 2011 Reference Design RD1089

Introduction

Analog-to-Digital Converters (ADCs) are used to convert analog signals into digital representations that can be communicated and processed using digital logic. This ADC reference design provides an example of how the LatticeECP3™ FPGA can be used to interface to a high-speed ADC device. Specifically, this ADC interface reference design supports the ability to interface with the Texas Instruments (TI) ADS64XX family of ADCs via LatticeECP3 FPGA high-speed LVDS I/O. The ADS64XX ADCs are high-performance ADC converters that use serial LVDS data outputs to reduce the number of interface signals required. This reference design receives data samples input via either one or two high-speed LVDS signals and converts the serial data to parallel word format.

Features

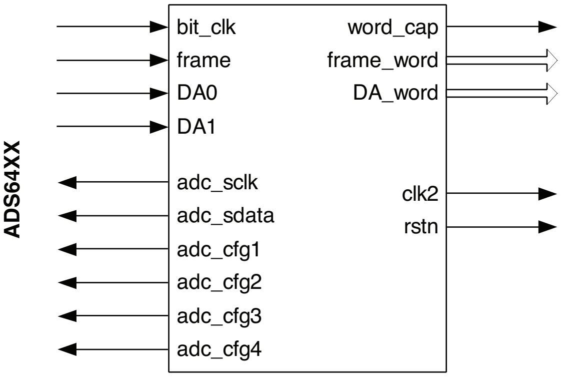
- Supports interface with TI ADS64XX ADC converters
- Supports 12-bit, 14-bit and 16-bit ADC sample data widths
- Supports 1-wire (one LVDS pair) and 2-wire (two LVDS pairs) interfaces

Functional Description

A high-level block diagram of the ADC interface reference design is shown in Figure 1. The reference design top-level I/O are listed in Table 1.

This reference design can be configured to support 12-bit, 14-bit or 16-bit ADC sample data. The ADC interface may be either 1-wire, where each ADC data is serialized and provided over one LVDS pair, or 2-wire, where each ADC data is serialized and provided over two LVDS pairs.

Figure 1. ADC Interface Reference Design Block Diagram



Serial data from the ADC converter are input via either one or two data signals, DA0 or DA1, together with a bit clock and a frame clock. The ADC converter configuration is specified by the adc_sclk, adc_sdata and adc_cfgx signals. A complete description of these signals and interfacing with TI ADCs can be found in the associated dat sheets (see, for example, Texas Instruments Data Sheet SLAS532A – ADS6424, ADS6423, ADS6422 - Quad Channel, 12-bit 105/80/65 MSPS ADC with Serial LVDS Interface).

The received serial data from the ADC is deserialized and provided as parallel data via the output bus DA_word. Output signal word_cap when set to ‘1’ indicates that DA_word contains valid output data. The clk2 output is the output data clock and runs at half the frequency of bit_clk. For a 12-bit 2-wire ADC interface, a frame (12 bits) of data is received every three bit_clk cycles. Hence, two valid DA_words are available every three clk2 clock data and word_cap is asserted twice every three clk2 clock cycles.

This reference design supports the ability to automatically adjust the LatticeECP3 LVDS input delay skew to ensure that the input data is sampled at the center of the data valid region. The ADC bit clock output is positioned at the center of the output data transitions. The ADC supports a SYNC test pattern capability that can be used to adjust the receiver sampling to compensate for relative skew between the bit clock and data signals due to routing delay differences. When configured for SYNC mode (via the adc_sclk and adc_sdata signals) the ADC continuously transmits a fixed test pattern. The reference design trains on the test pattern and automatically adjusts the input delay skew to ensure that the data is sampled near the center of the signal transition region.

Table 1. Top Level Interface and I/O

Signal	Input/Output	I/O Type	Description
ADC Interface			
bit_clk	Input	LVDS	ADC output data bit clock.
frame	Input	LVDS	ADC output data frame clock.
DA0	Input	LVDS	ADC Channel A differential LVDS data input, wire 0.
DA1	Input	LVDS	ADC Channel A differential LVDS data input, wire 1.
adc_sen	Output	LVCMOS	Coarse gain and internal/external reference.
adc_sclk	Output	LVCMOS	Sync, deskew patterns and global power down control.
adc_sdata	Output	LVCMOS	Sync, deskew patterns and global power down control.
adc_cfg1	Output	LVCMOS	1-wire/2-wire and DDR/SDR bit clock select.
adc_cfg2	Output	LVCMOS	12x/14x serialization and SDR bit clock capture edge select.
adc_cfg3	Output	LVCMOS	Reserved function. Tied to ground.
adc_cfg4	Output	LVCMOS	MSB/LSB first and data format select.
FPGA Output Parallel Interface			
word_cap	Output	LVCMOS	Word captured signal, indicates output data is valid when set to 1.
frame_word	Output	LVCMOS	Captured frame word used to verify proper alignment. Updated every clk2 clock cycle, contains 8 new bits of sampled frame data each clock cycle.
DA_word	Output	LVCMOS	Channel A output data word, valid when word_cap asserted.
System Signal			
clk2	Output	LVCMOS	Data processing clock. Divided by 2 from bit_clk.
rstn	Input	LVCMOS	System reset signal.

Design Module Description

A functional block diagram of the ADC interface reference design is shown in Figure 2. As indicated previously, this reference design can be configured to support 12-bit, 14-bit or 16-bit ADC sample data. The ADC interface may be either 1-wire, where each ADC data is serialized and provided over one LVDS pair (Figure 3), or 2-wire, where each ADC data is serialized and provided over two LVDS pairs (Figure 4). The reference design configuration is specified via settings in the adc_def.v file.

The reference design top module instantiates two submodules: adc_delay and adc_bit. The adc_delay module calibrates the LVDS input delay required to center the frame and AD0/AD1 data signals relative to bit_clk and provides output data nibbles to the adc_bit module. The adc_bit module performs bit slip calibration and alignment. It first aligns the data nibbles from the adc_delay module with the SYNC pattern nibbles, then ensures the data and frame nibbles are aligned (via the data_slip sub module).

The top level module combines the aligned nibble data to the output data bus width. As described previously, the DA_word output is only valid every two out of three clk2 clock cycles. Valid data is available when word_cap is set to ‘1’. The relative timing of DA_word and word_cap is shown in Figure 5.

Figure 2. ADC Interface Reference Design Functional Block Diagram

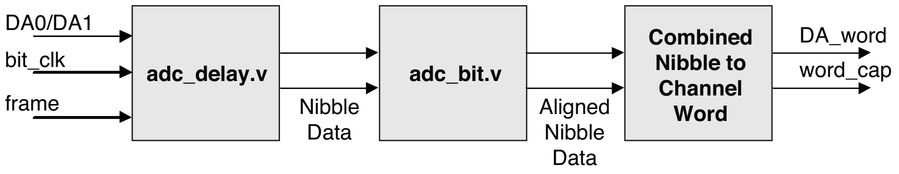


Figure 3. 1-Wire Interface – 12-bit Serialization with DDR Bit Clock

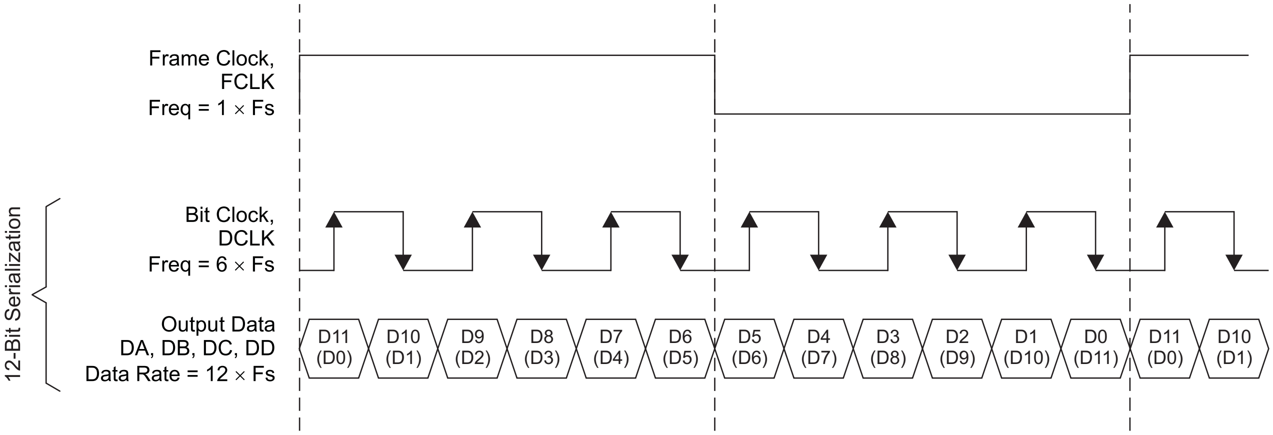


Figure 4. 2-Wire Interface – 12-bit Serialization with DDR Bit Clock

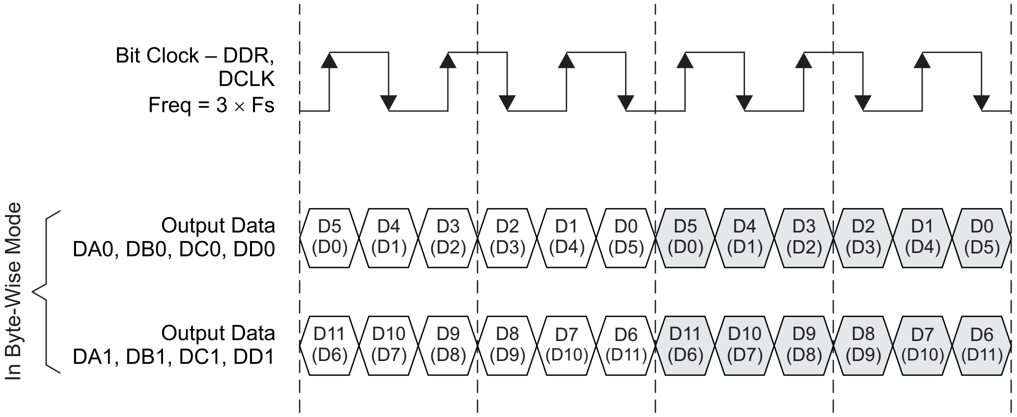
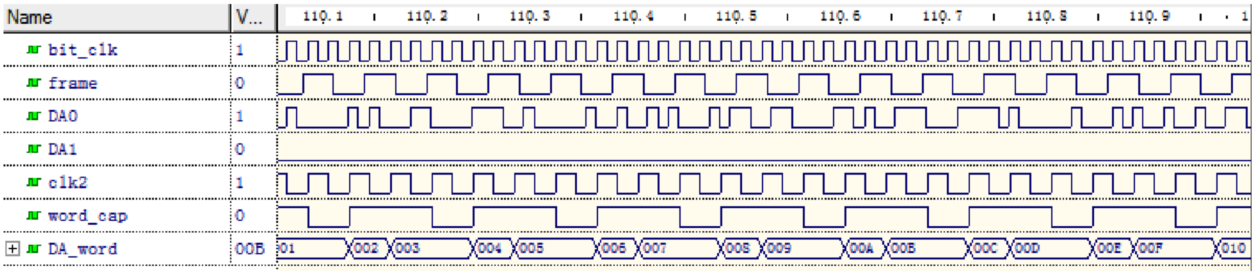


Figure 5. 2-Wire 12-Bit ADC Output Waveform

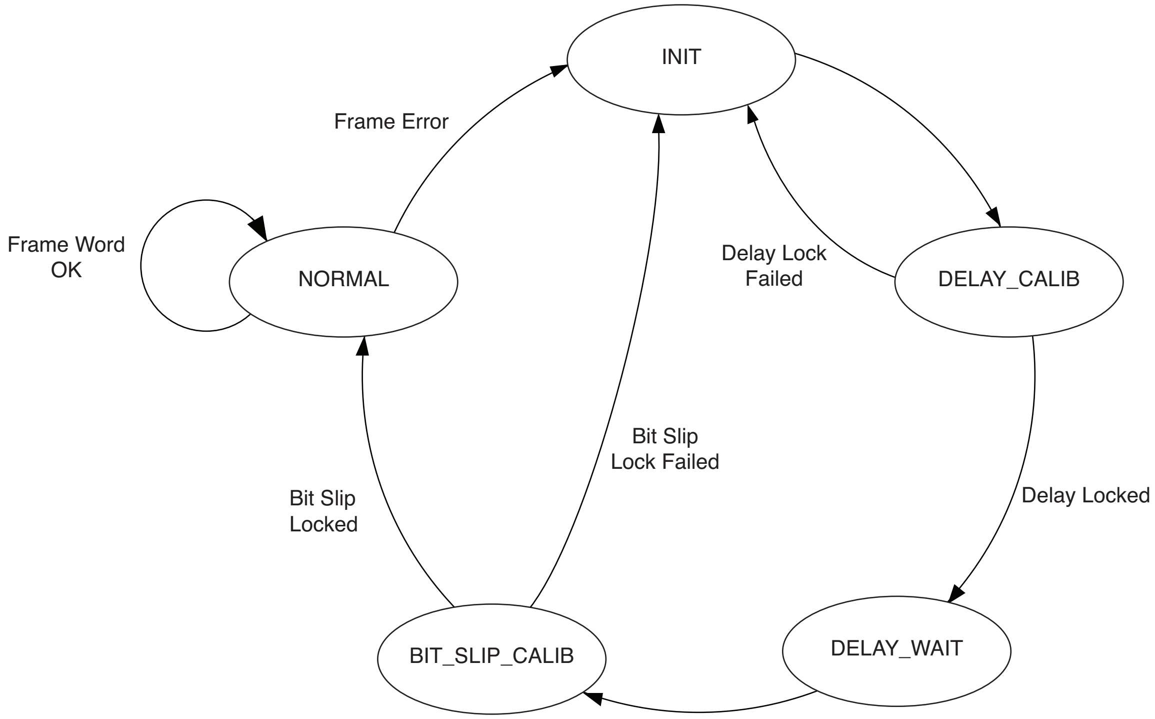


Data Sync and Alignment State Machine

A diagram of the data sync and alignment state machine is shown in Figure 6. Upon power-up, the state machine enters the INIT state and sets the ADC interface bits to configure the ADC mode and initiate transmission of the

SYNC pattern by the ADC. The state machine enters the DELAY_CALIB state and automatically adjusts the LatticeECP3 LVDS input delay skew to sample the input data at the center of the data valid region. After completing the input delay calibration process, there is brief delay to ensure all inputs are stable. The state machine next performs bit slip calibration and nibble alignment. After all input and data alignment processing is complete, the ADC is configured for normal operating mode and parallel data is available at the FGPA output. The state machine continuously monitors the received frame signal for proper alignment and re-initializes the link if any problems are detected.

Figure 6. Data Sync and Alignment State Machine



Delay Calibration

The adc_delay module uses the LatticeECP3 input delay dynamic setting capability to adjust the LVDS input delay skew to ensure that the input data is sampled at the center of the data valid region. This feature supports the ability to adjust the data input delay in 35 ps increments over 16 steps (total delay range approximately 560 ps). The adc_module automatically adjusts the delay setting, checking for correct data recovery at each delay setting. Once the range of valid settings is determined, the middle of the range is selected as the final delay value setting. Delay settings are determined independently for the frame, AD0 and AD1 inputs. Delay calibration is only active during the DELAY_CALIB state. For all subsequent states the input delay is fixed to the final delay value. The logic returns to the INIT state if a delay value supporting valid data recovery is not found.

Bit Slip Calibration

The adc_bit module implements the bit slip calibration function that aligns the deserialized data to the proper nibble boundaries. For example, the frame word for a 2-wire 12-bit ADC interface is 12’b111000111000. The bit slip calibration capability aligns the parallel nibbles to the corresponding valid aligned nibble values of 4’b1110, 4’b0011,4’b1000. Any other nibble values indicate invalid alignment.

The adc_bit module first determines the proper frame signal nibble alignment. It then determines the appropriate ADC data bit alignment (data_slip submodule) by comparing the frame signal nibble alignment with the nibble alignment of the SYNC pattern data transmitted by the ADC in the AD0 and AD1 data signals. The alignment capability ensures that the frame and data signals area aligned within +/- one bit position. The logic returns to the INIT state if proper frame and data nibble alignment can not be determined.

Normal Data Processing

Normal data processing begins once proper delay calibration and alignment have been determined. Logic in the top level combines the data nibbles to create the proper data words and generates the appropriate data valid (word_cap) signal. The received frame signal is continuously monitored for proper alignment. The logic returns to the INIT state if any problem with the frame signal is detected.

HDL Simulation and Verification

The reference design package includes a test bench that can be used to verify functionality and observe the associated signals. The test bench provides ADC frame and DA0/DA1 data signals with associated delay and skew. Figure 7 shows the initial values of the ADC signal delay settings. If the design cannot recover data correctly, del_cal_req is asserted by the adc_bit module and the test bench reduces the DA0/DA1 delay and the design is reinitialized. This flow is shown in Figure 8.

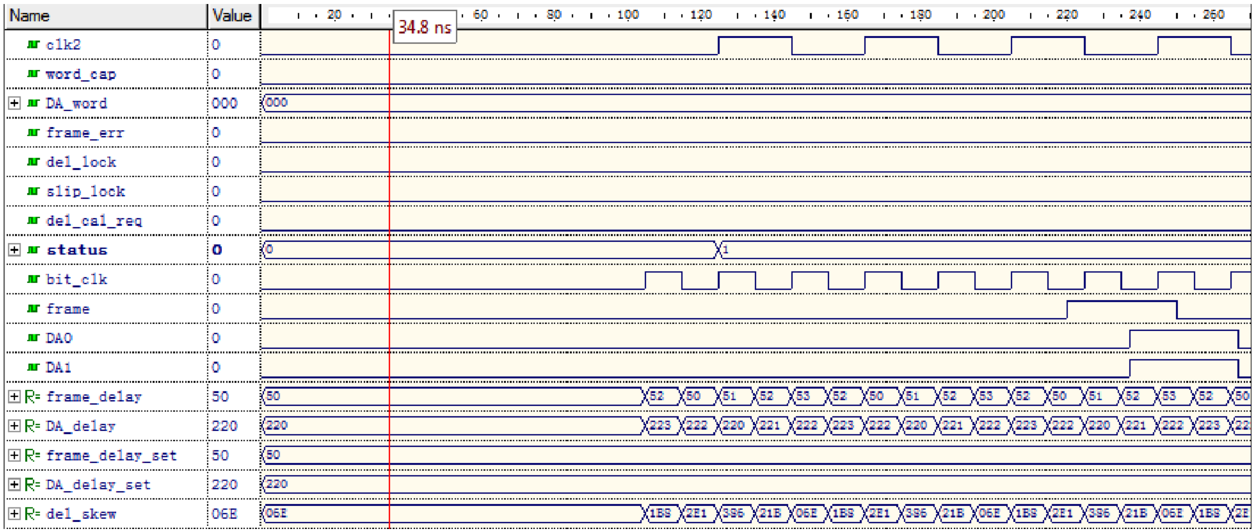
When the design correctly finishes delay and bit slip calibration, it enters the normal data processing state. The test bench sources a simple counting data pattern on AD0/AD1 (0, 1, 2, …) which appears in parallel format at the DA_word output as shown in Figure 9.

During normal signal processing, the test bench introduces changes the frame and DA0/DA1 signal delays, causing the design to go out of lock and assert the frame_err signal. At this point, data processing is stopped, word_cap is deasserted. The design returns to the INIT state and recalibrates to the new delays. This flow is shown in Figure 10.

Table 2. Test Bench Delay Setting Signals

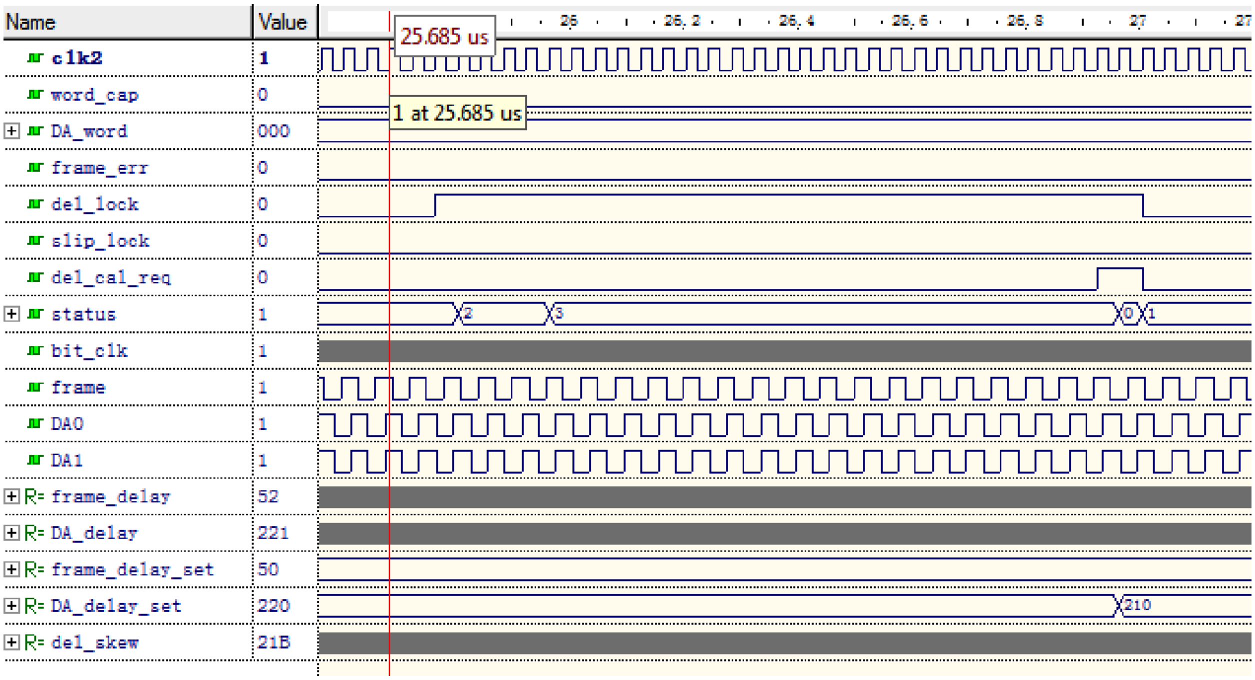
Delay Signal	Initial Value	Bus Width	Description
frame_delay_set	50	16	Initial frame signal delay value
DA_delay_set	220	16	Initial DA delay value. When bit slip calibration fails, the test bench decreases this value by 10.
frame_delay	50	16	Actual delay setting on frame signal. Equal to frame_delay_setting plus delay skew value
DA_delay	220	16	Actual delay setting on DA0/DA1 It is equal to DA_delay_set plus delay skew value.
del_skew	10’b0001101110	10	Simulate a delay skew between the frame and DA signals. Bit0 and bit1 are used for frame delay skew, bit2 and bit3 are used for DA delay skew.

Figure 7. Test Bench Initializes ADC Delay Value



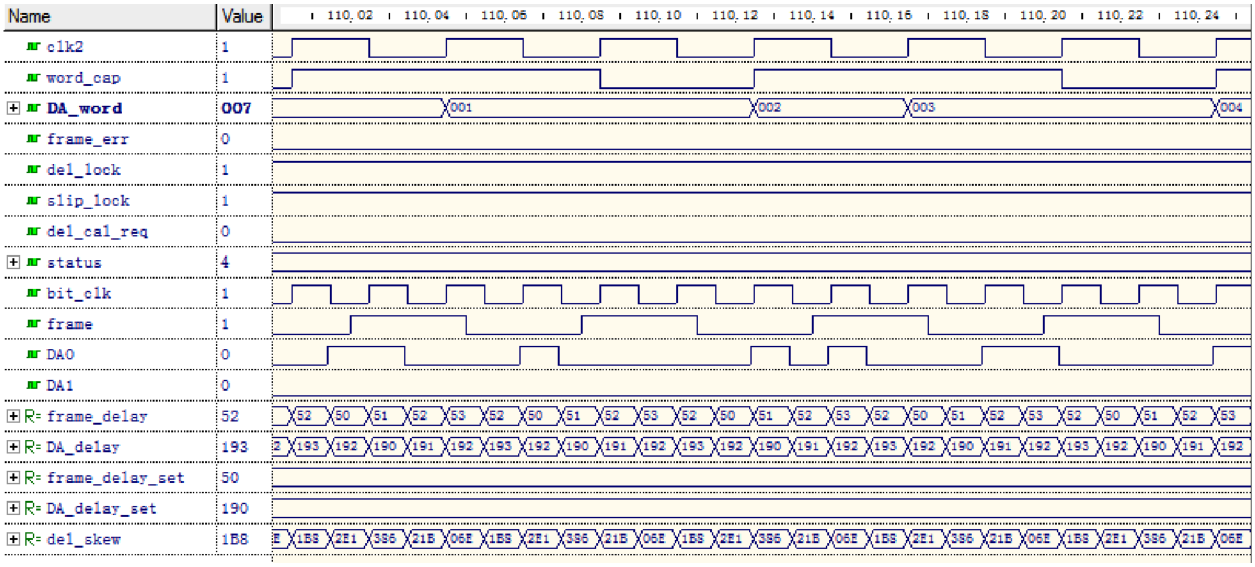
Name	Value
clk2	0
word_cap	0
DA_word	000
frame_err	0
del_lock	0
slip_lock	0
del_cal_req	0
status	0
bit_clk	0
frame	0
DA0	0
DA1	0
frame_delay	50
DA_delay	220
frame_delay_set	50
DA_delay_set	220
del_skew	06E

Figure 8. Design Return to INIT and Test Bench Reduces DA_delay_set



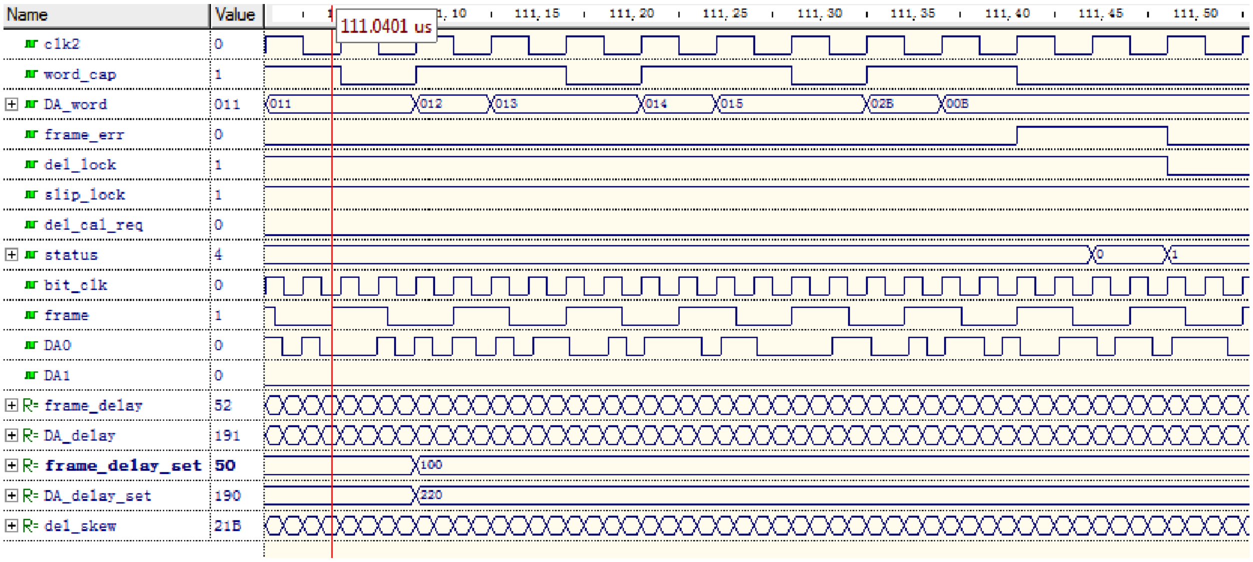
Name	Value
clk2	1
word_cap	0
DA_word	000
frame_err	0
del_lock	0
slip_lock	0
del_cal_req	0
status	1
bit_clk	1
frame	1
DA0	1
DA1	1
frame_delay	52
DA_delay	221
frame_delay_set	50
DA_delay_set	220
del_skew	21B

Figure 9. Design in Normal State



Name	Value
clk2	1
word_cap	1
DA_word	007
frame_err	0
del_lock	1
slip_lock	1
del_cal_req	0
status	4
bit_clk	1
frame	1
DA0	0
DA1	0
frame_delay	52
DA_delay	193
frame_delay_set	50
DA_delay_set	190
del_skew	1B8

Figure 10. Test Frame Error During NORMAL ADC



Implementation

This design is implemented in Verilog. When using this design in a different device, density, speed, or grade, performance and utilization may vary. Default settings are used during the fitting of the design.

Table 3. Performance and Resource Utilization

Device Family	Language	Speed Grade	Utilization (LUTs)	f_{MAX} (MHz)	I/Os	Architecture Resources
LatticeECP3™ [1]	Verilog	-8	337	500 ECLK, 250 SCLK	42	3 IDDRs
LatticeECP2™ [2]	Verilog	-7	350	500 ECLK, 250 SCLK	42	3 IDDRs

1. Performance and utilization characteristics are generated using a LFE3-70EA-8FN484C, with Lattice Diamond™ 1.2 design software.
2. Performance and utilization characteristics are generated using LFE2-6E-7T144C, with Lattice Diamond 1.2 design software.

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
March 2010	01.0	Initial release.
July 2010	01.1	Updated for ispLEVER 8.1 support.
April 2011	01.2	Updated for Lattice Diamond 1.2 software support.