User's Guide

DLP® DLPC910 Apps FPGA Guide

TEXAS INSTRUMENTS

ABSTRACT

The DLPC910 Apps FPGA Guide describes the functions and registers of the DLPC910 Applications FPGA (Apps FPGA) designed to work with a DLP LightCrafter DLPC910 EVM (DLPLCRC910EVM) and a supported DMD EVM. In addition, the guide provides an overview of the VHDL code and implementation.

Table of Contents

List of Figures

List of Tables

Trademarks

Virtex™ and Vivado™ are trademarks of AMD.
All trademarks are the property of their respective owners.

1 Introduction

The DLPC910 Apps FPGA code provides functionality and an example of interfacing to a DLPC910 controller and supported DMD. The DLPC910 Apps FPGA Guide details the functions and registers of the DLPC910 Applications FPGA (Apps FPGA) as well as the organization of the VHDL code used to create.

1.1 Welcome

The DLP LightCrafter DLPC910 EVM (DLPLCRC910EVM) is a great way to evaluate the highest bandwidth data rates the DLP chip portfolio offers. Designers get pixel-accurate control of all DMD micromirrors with Global, Quad, Dual and Single Block Modes available for tailoring DMD micromirror pattern timing for continuous (lamp) or solid state (switched) illuminated applications.

The DLPC910 Apps FPGA implementation enables exploration of the capabilities of the DLPLCRC910EVM and supported DMDs such as the DLP6500, DLP9000X, and DLP9000XUV

The DLPLCRC910EVM is also a path for customers to design with the high-performance DLP9000XUV chip for systems such as:

- Lithography
- 3D printing and additive manufacturing
- Dynamic grayscale marking and coding
- Industrial printing
- Structured light such as:
  - Factory automation and 3D machine vision
  - 3D In-line automated optical inspection (AOI)
  - Robotic vision
  - Offline 3D metrology
  - 3D scanners
  - 3D identification and biometrics
- Medical and life sciences
- High-speed imaging and display

2 Overview

2.1 Purpose

This document describes the functions and registers of the DLPC910 Applications FPGA (Apps FPGA). An overview of the VHDL code is also provided.

2.2 Apps FPGA Hardware Target

The apps FPGA reference code is targeted to the AMD Xilinx Virtex™-7 FPGA housed on an AMD Xilinx VC-707 Evaluation Board. As shown in Apps FPGA Hardware Target, the VC-707 Evaluation Board connects to the Texas Instruments (TI) DLPC910 Evaluation Module (DLPLCRC910EVM), which connects to one of three available TI DMD boards.

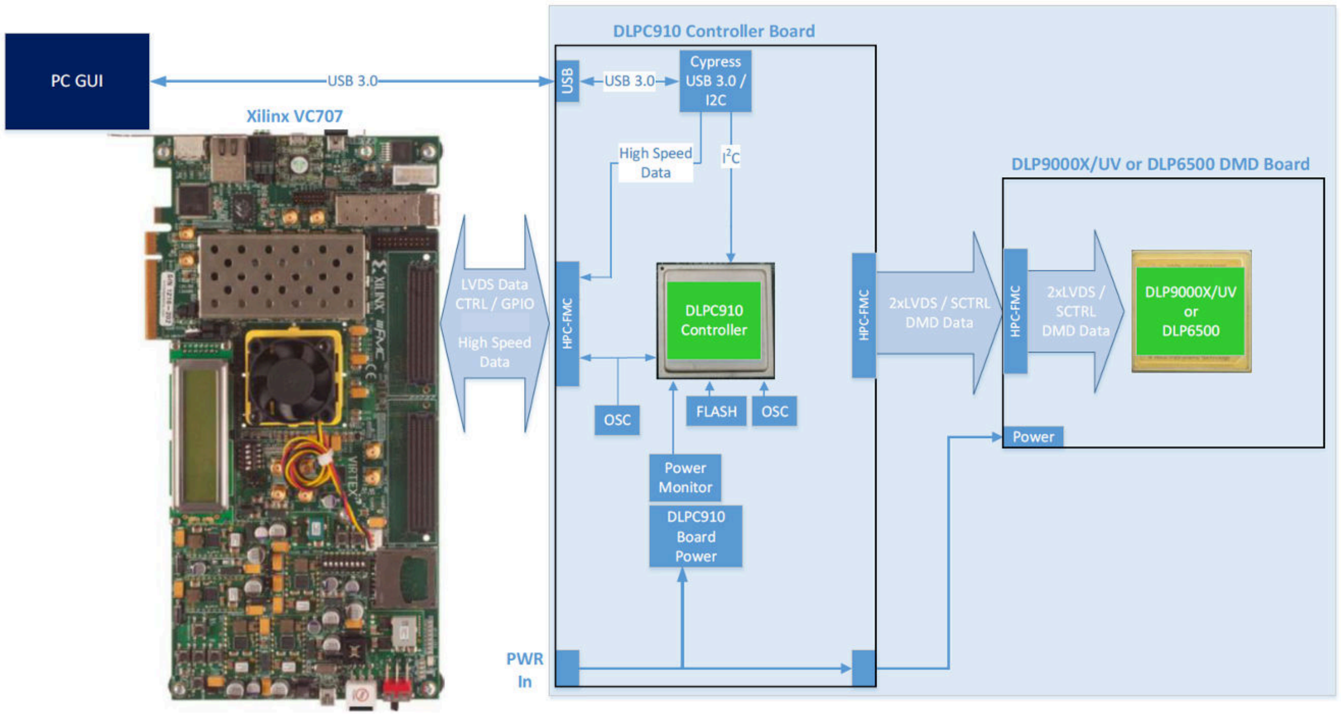


Figure 2-1. Apps FPGA Hardware Target

3 Interfaces

This section describes the apps FPGA interface signals. Apps FPGA interfaces to the DLPC910 EVM board through two high pin count FMC connectors (J500 and J501 on the DLPLCRC910EVM board). In addition to the DLPC910 interfaces, the apps FPGA uses push button switches, dip switches, and LEDs on the VC-707 evaluation board.

Apps FPGA interface signal pin numbers and signal properties are described in the input-output (IO) physical constraints (.xdc) file provided with the Apps FPGA source code.

Signal names used in this section match the signal names used in the Apps FPGA top level VHDL module.

3.1 LVDS high speed data interface to DLPC910

The Apps FPGA drives image data to the DLPC910 over VC-707 FMC HPC1 connector, consisting of four, 16 data pair, LVDS buses. Each of the four buses also has their own clock and data valid signal pairs. Signal level, timing, and data mapping details can be found in the DLPC910 data sheet.

Apps FPGA output signal names for the LVDS data interface to the DLPC910 are summarized in Apps FPGA - LVDS data interface output signal names.

Table 3-1. Apps FPGA - LVDS data interface output signal names

	LVDS Bus A	LVDS Bus B	LVDS Bus C	LVDS Bus D

Data	`dout_ap(15:0)`
	`dout_an(15:0)`
		`dout_bp(15:0)`
		`dout_bn(15:0)`
			`dout_cp(15:0)`
			`dout_cn(15:0)`
				`dout_dp(15:0)`
				`dout_dn(15:0)`

Clock	`dclk_ap` `dclk_an`	`dclk_bp` `dclk_bn`	`dclk_cp` `dclk_cn`	`dclk_dp` `dclk_dn`
Data Valid	`dvalid_ap` `dvalid_an`	`dvalid_bp` `dvalid_bn`	`dvalid_cp` `dvalid_cn`	`dvalid_dp` `dvalid_dn`

3.1.1 DLP9000X and DLP9000XUV

The Apps FPGA sends data to DLPC910 over all four (A,B,C,D) LVDS data buses when a DLP9000X or a DLP9000XUV DMD is in use. At startup, the Apps FPGA determines the connected DMD Type using `dmd_type(3:0)` input from the DLPLCRC910EVM, and drives the appropriate number of LVDS buses. Additionally, at startup, the Apps FPGA sets the interface clock speed to either 400 MHz or 480 MHz according to `dmd_speed_sel(1:0)` input from the DLPLCRC910EVM.

Note

Although the DLP9000X and DLP9000XUV run at 400 MHz, only 480 MHz operation has been fully validated.

3.1.2 DLP6500

The Apps FPGA sends data to the DLPC910 over LVDS data buses A and B when a DLP6500 DMD is in use. The Apps FPGA determines the connected DMD using `dmd_type(3:0)` input. For a DLP6500 DMD, clock frequency is fixed to 400 MHz.

3.2 Data Load Control Signals to DLPC910

Data load control signals listed in Table 3-2 are output to the DLPC910 over VC-707 FMC connectors. Signal function is described in the DLPC910 data sheet.

Table 3-2. Data Load Control Signals

Name	Apps FPGA I/O	Function
`rowmd(1:0)`	out	DMD row mode
`rowad(10:0)`	out	DMD row address
`comp_data`	out	Complement data
`load4z`	out	DMD load 4 function enable (active low)

Table 3-2. Data Load Control Signals (continued)

Name	Apps FPGA I/O	Function
`ns_flip`	out	Top/bottom image flip on DMD

3.3 DMD Reset and Block Clear Signals to the DLPC910

Table 3-3 lists the DMD Reset and Block Clear signals to and from the DLPC910. These signals connect through the VC-707 FMC connectors. Refer to the DLPC910 data sheet for additional information.

Table 3-3. Reset and Block Clear Signals

Name	Apps FPGA I/O	Function
`blkad(3:0)`	out	block address
`blkmd(1:0)`	out	block mode
`rst2blkz`	out	dual and quad block control
`wdt_enablez`	out	DMD reset pulse watch dog timer enable
`rst_active`	in	DMD mirror clocking pulse (MCP) in progress

3.4 DLPC910 Initialization and Controller Reset Signals

Table 3-4 describes the initialization signals for the DLPC910 controller. See the DLPC910 data sheet for additional signal information.

Table 3-4. DLPC910 Initialization and Controller Reset Signals

Name	Apps FPGA I/O	Function
`ctrl_rstz`	out	DLPC910 controller reset
`pwr_floatz`	out	DLPC910 PWR_FLOAT
`ecp2_finished`	in	DLPC910 configuration from SPI Flash status
`init_active`	in	DLPC910 initialization active status

3.5 Apps FPGA Reset Signal - `apps_resetz`

Signal `apps_resetz` originates from a push button switch (SW1 APP RST) on the DLPLCRC910EVM board. When the switch is pressed the `apps_resetz` signal goes low, causing the init-run-park state machine in the apps FPGA to drive `pwr_floatz` low to park the DMD mirrors. Upon release of the push button switch, the init-run-park state machine re-initializes the DLPC910 controller and resets the Apps FPGA logic.

3.6 DLPC910 Status-Info Signals

DLPC910 status-info signals are listed in Table 3-5. The `ddc_version, DMD_type`, and `DMD_irq` signals are from the DLPC910 controller. Only `DMD_type` is used by the Apps FPGA logic. The version and irq signals are made available in status registers via the USB GPIF.

The `DMD_speed_sel` signal is from a set of jumpers on the DLPLCRC910EVM board. The jumper settings are used by DLPC910 controller and by Apps FPGA to determine clock frequency of the LVDS high speed interface DLP9000X and DLP9000XUV.

Reference the DLPC910 data sheet for additional information.

Table 3-5. DLPC910 Status-Info Signals

Name	Apps FPGA I/O	Function
`ddc_version(2:0)`	in	DLPC910 firmware version
`dmd_type(3:0)`	in	DLPC910 DMD type
`dmd_speed_sel(1:0)`	in	Apps/DLPC910 LVDS speed select jumpers
`dmd_irq`	in	DMD irq status signal

3.7 USB GPIF (Interface)

The USB GPIF port provides access to Apps FPGA internal control-status registers. The USB GPIF port also provides a data input FIFO used for loading the user image buffer with user images for display. Table 3-6 describes the signals in the interface.

Table 3-6. USB GPIF Interface Signals

Name	Apps FPGA I/O	Function
`gpif_addr(8:0)`	in	Not used
`usb_fd(15:0)`	in/out	USB GPIF data, bi-directional
`usb_ctrl(5:3)`	in	USB GPIF control – not used(1)
`usb_ctrl(2:0)`	in	USB GPIF control, identifies transaction type
`usb_rdy(2:1)`	out	USB GPIF ready outputs – not used(2)
`usb_rdy(0)`	out	USB GPIF ready output 0(3)
`usb_reset`	out	Not used – apps FPGA drives low always
`usb_if_clock`	in	USB GPIF clock, 48 MHz

(1) `usb_ctrl(5:3)` inputs are not used.
(2) `usb_rdy(2:1)` outputs are always driven low.
(3) `usb_rdy(0)` is the empty flag of the USB GPIF input FIFO.

The type of GPIF transaction is defined by `usb_ctrl(2:0)` as shown in Table 3-7.

Table 3-7. USB GPIF Transaction Type Definition

Signal	Idle	Address	Data write	Data read	FIFO burst
`usb_ctrl(2)`	1	0	0	0	1
`usb_ctrl(1)`	1	1	1	0	1
`usb_ctrl(0)`	1	1	0	1	0

3.7.1 Apps FPGA Register Address Read-Write Transactions

Reading or writing Apps FPGA registers requires first sending a register address followed by register data. Register addresses and data are 32 bits in width. Two 16-bit bus transactions are required to transfer 32 bits.

Note

Data is sampled on the falling edges of the `usb_if_clk`.

3.7.1.1 Apps FPGA Register Address Transaction

Register address transaction timing is shown in Register Address Transaction Timing Diagram.

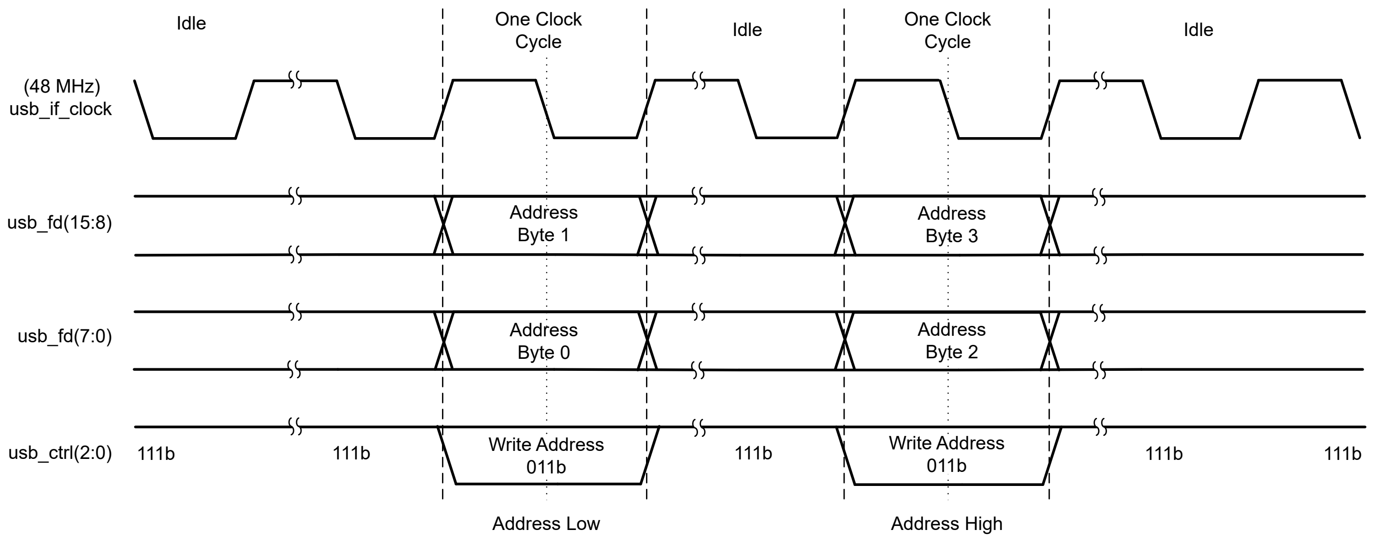


Figure 3-1. Register Address Transaction Timing Diagram

A register address transaction example is shown in Register Address Transaction Example. There can be more than one idle clock cycle between address transfers. The address transaction table shows an address transfer of `0x00010228`.

Table 3-8. Register Address Transaction Example

Signal	Idle	Write low address	Idle	Write high address	Idle
`usb_if_clock`		1 clock cycle		1 clock cycle	
`usb_fd(15:8)`	0x00	0x02	0x00	0x00	0x00
`usb_fd(7:0)`	0x00	0x28	0x00	0x01	0x00
`usb_ctrl(2:0)`	Idle	Address – “011”	Idle	Address – “011”	Idle

3.7.1.2 Apps FPGA Register Data Write Transaction

Register data write transaction timing is shown in Register Data Write Transaction Timing Diagram.

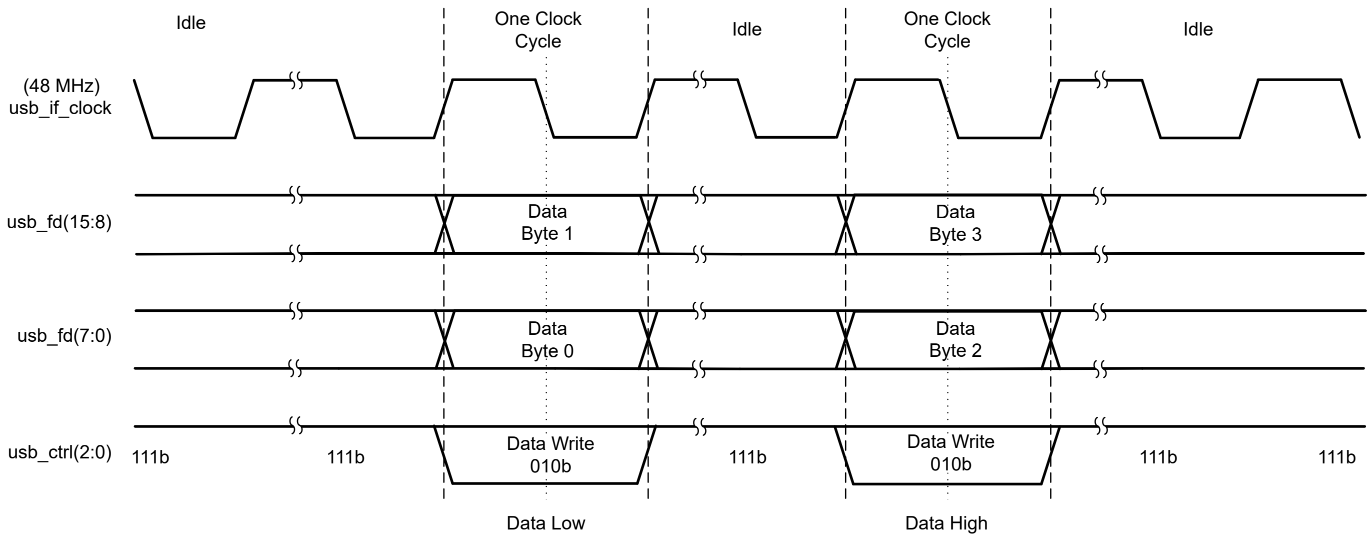


Figure 3-2. Register Data Write Transaction Timing Diagram

A register data write transaction example is shown in Register Data Write Transaction Example. A register write data value of 0x04030201 is being transferred.

Table 3-9. Register Data Write Transaction Example

Signal	Idle	Write low word	Idle	Write high word	Idle
usb_if_clock		1 clock cycle		1 clock cycle	
usb_fd(15:8)	0x00	0x02	0x00	0x04	0x00
usb_fd(7:0)	0x00	0x01	0x00	0x03	0x00
usb_ctrl(2:0)	Idle	Data wrt – “010”	Idle	Data wrt – “010”	Idle

3.7.1.3 Apps FPGA Register Data Read Transaction

Register read word transactions are 3 clocks in length. Register data read transaction timing is shown in Register Data Read Transaction Timing Diagram.

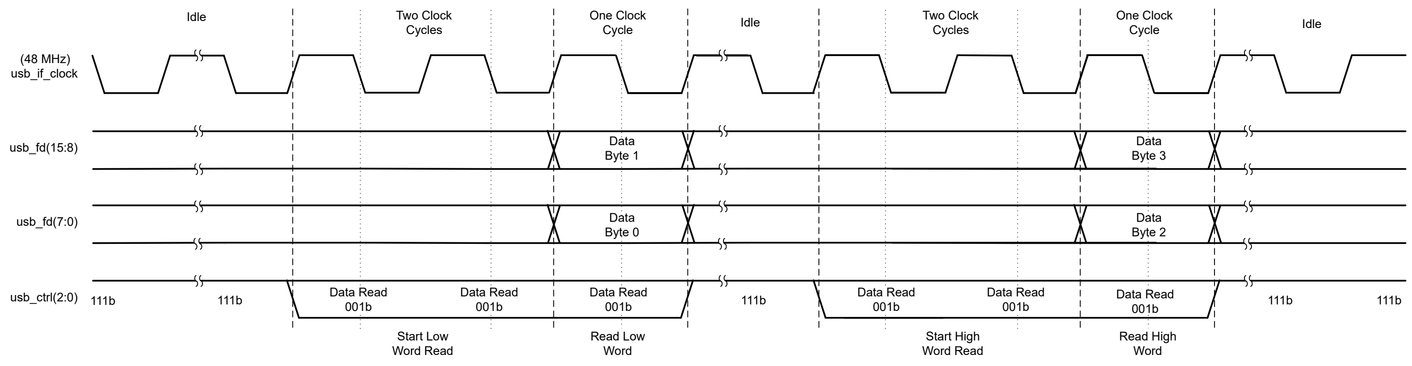


Figure 3-3. Register Data Read Transaction Timing Diagram

The Apps FPGA GPIF supports multiple register read transactions from the previously written register address, without needing a new address transaction. This lowers the time overhead when polling a status register.

A Register read word transaction example is shown in Register Data Read Transaction Example and illustrates reading of register value 0x04030201.

Table 3-10. Register Data Read Transaction Example

Signal	Idle	Start low word read	Read low word	Idle	Start high word read	Read high word	Idle
usb_if_clock		2 clock cycles	1 clock cycle		2 clock cycles	1 clock cycle	
usb_fd(15:8)	0x00	--	0x02	0x00	--	0x04	0x00
usb_fd(7:0)	0x00	--	0x01	0x00	--	0x03	0x00
usb_ctrl(2:0)	Idle	Data read– “001”	Data read– “001”	Idle	Data read– “001”	Data read – “001”	Idle

3.7.2 FIFO Write Transaction

The USB GPIF FIFO is used to transfer image data from the USB GPIF to the user image buffer in the Apps FPGA. The FIFO size is 256 words of 16 bits each. FIFO write transactions are always 256 words (512 bytes) in length.

FIFO write transaction timing is shown in FIFO Write Transaction Timing Diagram.

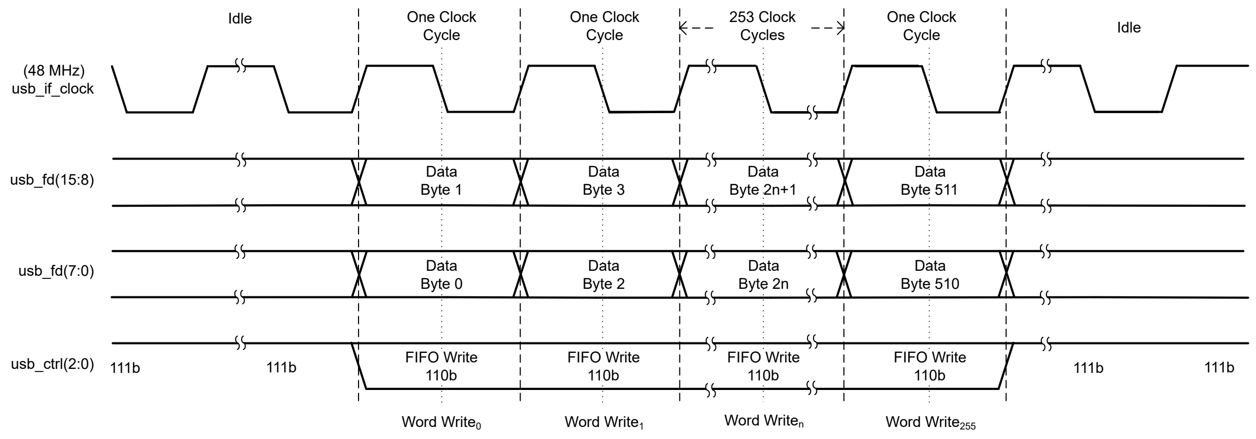


Figure 3-4. FIFO Write Transaction Timing Diagram

3.8 DLPLCRC910EVM Dip Switch (SW2)

The DLPLCRC910EVM houses a dip switch (SW2) whose switch settings are used by the Apps FPGA as described in Table 3-11.

Note

The inputs connected to DLPLCRC910EVM SW2 are pulled high through a pullup resistor when in the "OFF" position [logic 1] and pulled low when in the "ON" position [logic 0]. Positions 0 and 1 are not enabled when the switch is "OFF" [logic 1] and positions 2, 3, and 7 are not enabled when "ON" [logic 0].

Table 3-11. DLPLCRC910EVM Dip Switch (SW2)

Signal	Switch Label	Function	Default Position(1)
`evm_in_dip_sw(7)`	8	wdt_enblz(2)	ON [logic 0]
`evm_in_dip_sw(6)`	7	not used	ON [logic 0]
`evm_in_dip_sw(5)`	6	not used	ON [logic 0]
`evm_in_dip_sw(4)`	5	not used	ON [logic 0]
`evm_in_dip_sw(3)`	4	ns_flip(2)	ON [logic 0]
`evm_in_dip_sw(2)`	3	comp_data(2)	ON [logic 0]
`evm_in_dip_sw(1)`	2	load4_enz(2)	OFF [logic 1]
`evm_in_dip_sw(0)`	1	pwr_float(3)	OFF [logic 1]

(1) Default Positions are as seen on the physical switch.
(2) Instructs the Apps FPGA to perform the function as described in DLPC910 data sheet. These functions are replicated in register 0x0010 (section Data Loading Control (0x0010)).
(3) Instructs the Apps FPGA to issue `pwr_floatz` to DLPC910. This function is replicated in register 0x0044 (section Park [PWR_FLOAT] (0x0044)).

3.9 VC-707 Dip Switch (SW2)

The VC-707 EVM board houses a dip switch (SW2 GPIO DIP SW) whose switch settings are used by the Apps FPGA as described in Table 3-12. For normal operation, set position 0 to ON, all other switches to OFF.

Table 3-12. VC-707 Dip Switch (SW2)

Signal	Switch Label	Function	Default Position
`vc707_in_dip_sw(7)`	8	not used	OFF
`vc707_in_dip_sw(6)`	7	not used	OFF
`vc707_in_dip_sw(5)`	6	alt_rows_sel(1)	OFF
`vc707_in_dip_sw(4)`	5	not used	OFF
`vc707_in_dip_sw(3)`	4	not used	OFF
`vc707_in_dip_sw(2)`	3	not used	OFF
`vc707_in_dip_sw(1)`	2	not used	OFF
`vc707_in_dip_sw(0)`	1	USB GPIF FIFO input word swap(2)	ON

(1) When position 6 switch is ON, a horizontal line pattern (all 0's even rows, all 1's odd rows) is inserted after the test pattern buffer. Used for debug only.
(2) When position 1 switch is ON, even input 16-bit words are swapped with odd input 16-bit words, before writing to the USB GPIF FIFO. The DLPLCRC910EVM implementation requires this setting to be enabled (ON).

3.10 VC-707 Push Button Switches

The VC-707 EVM houses push button switches whose switch inputs are used by the Apps FPGA as described in Table 3-13.

Table 3-13. VC-707 Push Button Switches

Signal	Switch Label	Function
`vc707_in_pb_sw(7)`	SW7	Not used
`vc707_in_pb_sw(6)`	SW6	Not used
`vc707_in_pb_sw(5)`	SW5	Debounced external trigger test(1)
`vc707_in_pb_sw(4)`	SW4	Reset for init-run-park – debug only
`vc707_in_pb_sw(3)`	SW3	Test pattern cycling(2)

(1) A debounced SW5 on the VC-707 board is made available to use as a test external trigger on the Apps FPGA test header (J3-pin 3 on DLPLCRC910EVM board). To test the external trigger using SW5 short J3-pin 2 to J3-pin3.
(2) When the Apps FPGA starts up, test patterns are sent to the DLPC910 for display on DMD. Test patterns change periodically (current default is two seconds). Pressing SW3 halts test pattern cycling. Subsequent presses of SW3 changes to the next test pattern. SW3 controlled pattern sequencing is ended by pressing the Apps FPGA reset switch on the DLPLCRC910EVM board.

3.11 VC-707 Status LEDs

The apps FPGA uses the VC-707 GPIO LEDs to indicate initialization & operating modes as described in Table 3-14.

Table 3-14. VC-707 Status LEDs

Signal	LED Label	Function
`vc707_led_ds(9)`	7	ON = `dmd_type` of 0xF, DLP9000X or DLP9000XUV
`vc707_led_ds(8)`	6	ON = `dmd_type` of 0xE, DLP6500
`vc707_led_ds(7)`	5	Spare
`vc707_led_ds(6)`	4	Spare
`vc707_led_ds(5)`	3	ON = USB GPIF clock PLL locked(1)
`vc707_led_ds(4)`	2	ON = `pwr_floatz` asserted, system parked
`vc707_led_ds(3)`	1	ON = Initialization complete, system running
`vc707_led_ds(2)`	0	ON = Initialization error(2)

(1) GPIO LED 3 is illuminated when the USB GPIF PLL is locked to the input clock, `usb_if_clk`, from the Infineon - Cypress USB interface chip.

(2) GPIO LED 0 is illuminated by the init-run-park state machine when anomalies occur during DLPC910 initialization.

3.12 DLPLCRC910EVM Apps FPGA Test Points

The DLPLCRC910EVM houses a dip switch (SW2) whose switch settings are used by the Apps FPGA as described in Table 3-15.

Table 3-15. DLPLCRC910EVM Apps FPGA Test Points

Signal	J3 pin	Apps FPGA signal
APP_TSTPT7	2	External global reset trigger input for user control
APP_TSTPT6	3	Debounced VC707 push button switch SW5 output
APP_TSTPT5	4	Data enable from the apps loader, output
APP_TSTPT4	5	Load busy from the apps loader, output
APP_TSTPT3	6	Mirror settling busy from the apps loader, output
APP_TSTPT2	7	Expose trigger from the apps loader, output
APP_TSTPT1	8	Mirror reset busy from the apps loader finite state machine (FSM), output
APP_TSTPT0	9	Reset active signal, output(1)

(1) Reset active signal does not go high during a *Mirror Float* operation

4 Operation

A functional block diagram of the Apps FPGA is shown in Figure 4-1. The overlay colors highlight the VHDL modules that perform each set of functions, with VHDL module names in *italics*.

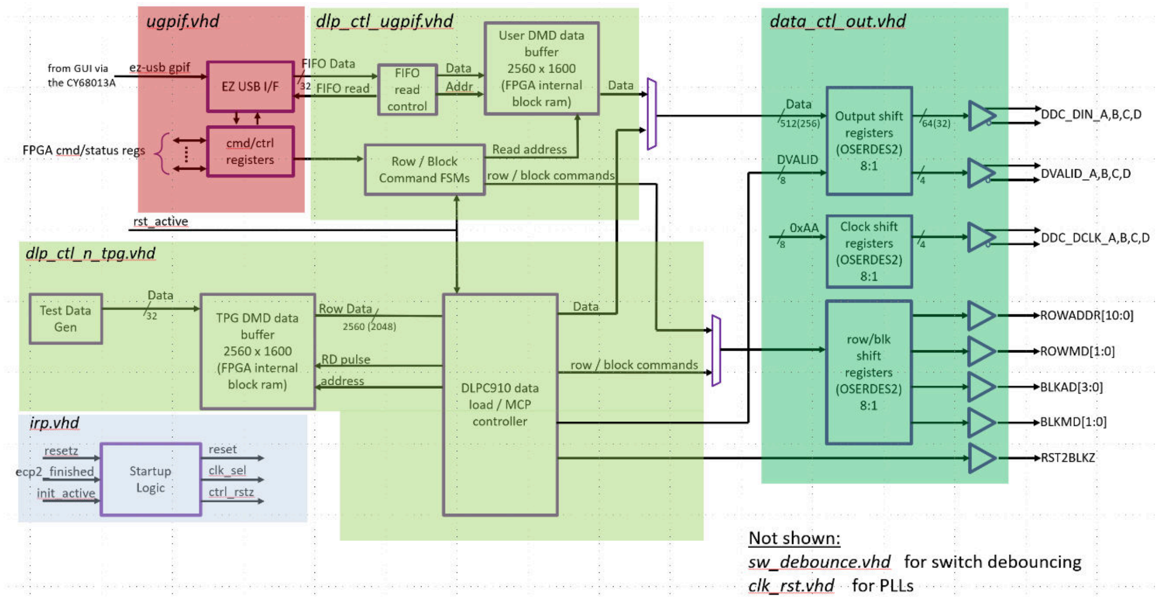


Figure 4-1. Apps FPGA Functional Block Diagram

The Apps FPGA has three operating states:

1. Initializing
2. Initialization error
3. Run
    a. Test pattern / Apps DLP load control
    b. User DLP load control

The Apps FPGA performs initialization and responds to initialization errors, as described in the Loader Control (0x0040) section. Once initialization is complete, the run state begins. In the run state, there are two independent DLP control methods selected by bit zero of the loader control register (Loader Control (0x0040)). The first is the test pattern Apps FPGA loader control of the DLP component set. The second is when the Apps FPGA loader is not in control, the DLP component set can be controlled through the USB GPIF interface (user control mode).

4.1 Initialization

The Apps FPGA provides the DLPC910 and attached DMD chip set initialization per the DLPC910 data sheet requirements. VHDL module *irp.vhd* (init-run-park control) and the sub-module *irp_fsm.vhd* (init-run-park state machine) provide this startup logic.

4.1.1 Initialization Prompts

Any of the following events cause the init-run-park state machine to execute the initialization routine:

1. Completion of the Apps FPGA configuration.
2. Pressing SW1 APP RST switch on DLPLCRC910EVM (`apps_resetz` to the Apps FPGA).
3. ON to OFF transition by position 1 of dip switch SW2 on DLPLCRC910EVM (*un-parking* the system using dip switch).
4. ‘1’ to ‘0’ transition of park bit in USB GPIF register 0x0044 (for example, *un-parking* the system using register command bit).

4.1.2 Init Routine

The init routine performed by the VC-707 consists of the following basic steps:

1. Issue reset to the DLPC910 controller (`ctrl_rstz`).
2. Verify that DLPC910 is configured (`ecp2_finished`).
3. Ensure Apps FPGA DLP PLLs are locked (`dclk_ap/n`).
4. Capture the selected DLPC910 interface clock speed (`dmd_speed_sel`).
5. Select either 400 or 480 MHz clock based on the value read from `dmd_speed_sel`.
6. Reset the AMD / Xilinx oserdes output primitives.
7. Ensure DLPC910 begins the initialization (`init_active`).
8. Turn on DLPC910 training pattern.
9. Release reset to DLPC910 (`ctrl_rstz`).
10. Wait for training to complete (`init_active` low).
11. Capture the type of DMD (`dmd_type`).
12. Reset Apps FPGA control logic (automatic).
13. Run.

Upon successful init completion, the init-run-park state machine enters the RUN state. From there, the USB GPIF park register, switch SW1 APP RST, and position 1 of SW2 are monitored. If any one of these inputs indicates mirrors require parking, then `pwr_floatz` is issued, and the state machine waits for these inputs to return to an operating state, before running the init routine again.

Note

A change to the `dmd_speed_sel` jumpers on the DLPLCRC910EVM requires re-running initialization (press SW1 APP RST switch).

4.1.3 GPIO Status LEDs

Three VC-707 board GPIO LEDs are used to give init-run-park status:

- GPIO LED 1 is illuminated when in the RUN state
- GPIO LED 2 is illuminated when DMD mirrors are parked
- GPIO LED 0 is illuminated when there is an initialization error

4.1.4 Errors

The initialization error flag is raised if any of the following occur:

1. DLPC910 does not issue ecp2_finished.
2. Apps FPGA DLP PLLs not locked.
3. DLPC910 init_active behavior is not as expected.
4. Mis-match between selected clock speed and DMD type.

Once an init error flag is raised, the init-run-park state machine stays in error state until reset (SW2 APP RST), upon which another initialization is attempted.

4.2 Test Pattern Generator (TPG) and Apps Loader - DLP Control

When the apps loader is controlling the DLP component set, basic data flow is as shown in Apps Loader Data Flow. The test pattern generator (TPG), data buffer, and apps loader control functions are all contained in VHDL module *dlp_ctl_n_tpg.vhd*.

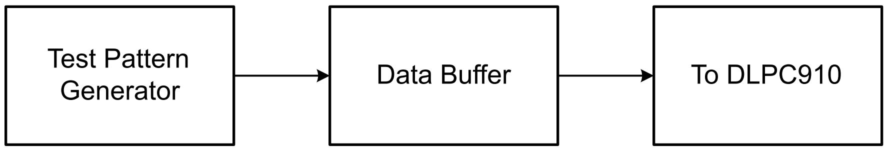


Figure 4-2. Apps Loader Data Flow

4.2.1 Test Pattern Generator (TPG)

Test pattern generation is provided by VHDL module *tpg.vhd* along with the sub-modules *tpg_trig.vhd* and *tpg_timing.vhd*. A VHDL package *tpg_pkg.vhd* provides test data constants for various test patterns.

After apps FPGA initialization is complete, 16 test patterns (section Test Pattern Control (0x0014)) are continuously cycled through periodically. Modify the cycle interval field of the test pattern control register (Test Pattern Control (0x0014)) to change the cycling rate. The test pattern control register also provides control bits to disable pattern cycling and continuously display a selected test pattern.

When test patterns are cycling after initialization, pressing momentary push button switch SW3 on the VC-707 board stops the pattern cycling. Successive activation of SW3 selects the next test pattern in the sequence. Pressing the Apps FPGA reset switch on the DLPLCRC910EVM board re-initializes the Apps FPGA and DLP component set and return to auto pattern cycling.

The test pattern generator provides control and addressing to write test patterns into the DMD data buffer. Patterns are read from the data buffer and sent to the DLPC910 under control of the DMD load state machine.

4.2.2 DMD Data Buffer

The DMD data buffer is comprised of five instantiations of a 512x1600x1 dual port RAM. The dual port RAM is created using the AMD Xilinx Block Memory Generator tool, accessible from the IP catalog in the AMD Vivado™ GUI. The VHDL module *dmd_data_buffer.vhd* provides the dual port data buffer function.

The DLP6500 uses only four of the five instantiations.

4.2.3 DMD Load State Machine

VHDL module, *dmd_load_fsm.vhd*, performs the DLPC910 test pattern data load function (“loader” function). The loader reads test pattern data from the DMD data buffer, formats the data, and sends the data to the *data_ctl_out.vhd* module for serialization and buffering to the DLPC910. In addition, the appropriate row address, row command, and DVALID control signals are created to send along with the data.

By default, the loader loads test pattern images in response to a load trigger pulse from the *dload_trigger.vhd* module. The default period for the trigger pulse is 400 us. The loader load interval register provides the capability to change the trigger period. Section 5.1.12.

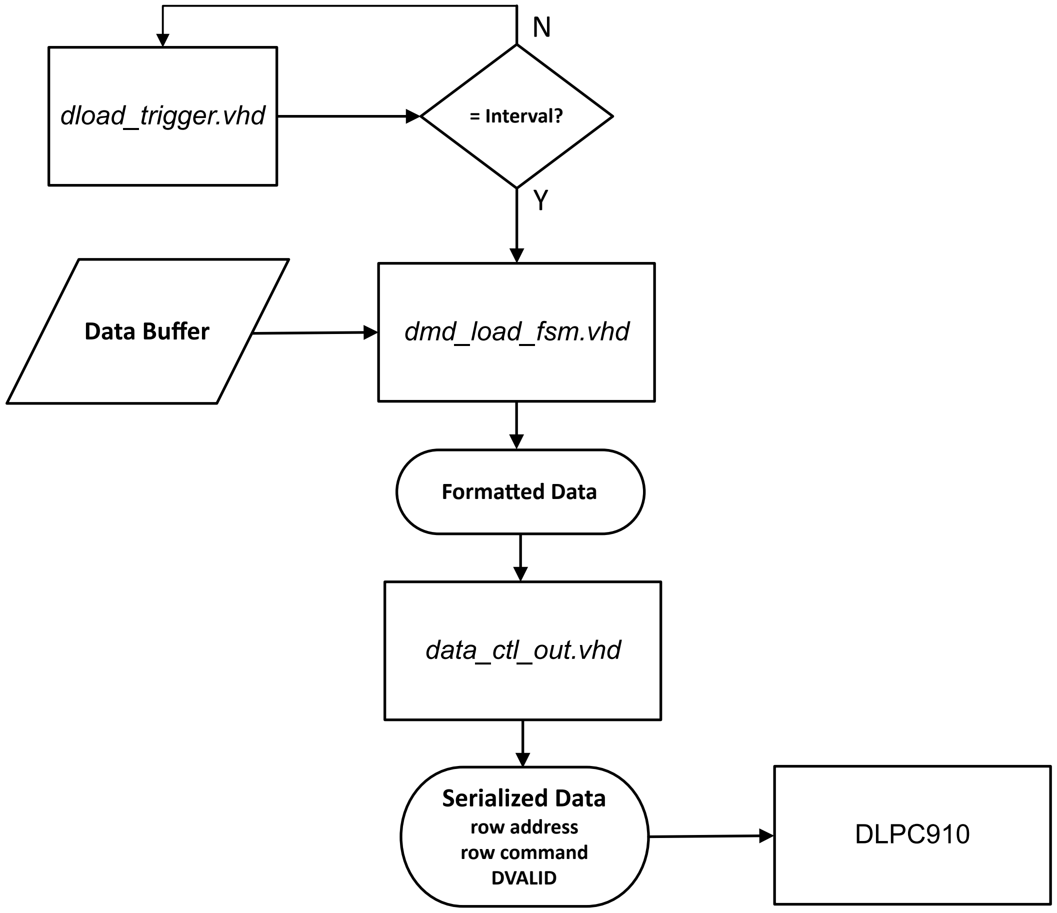


Figure 4-3. DMD Load State Machine

When the reset type is global or quad-block phased, the loader can be put into free-run mode, where the load trigger is ignored and the loader loads images successively, and as quickly as possible. The free-run mode demonstrates the minimum DLP component set image load times. Loader Control (0x0040)

The following loader control settings are configurable through USB GPIF registers:

1. Reset type. Default is global. See Loader Reset Type (0x001C).
2. Starting DMD row and number of DMD rows to load. Default is row 0 and full device load. See Loader Row Control (0x0034).
3. Expose time. Default is 0. See Loader Expose Time (0x003C).
4. Data loading control. See Data Loading Control (0x0010) for default values.

Note

To avoid unpredictable results when the loader is in free-run mode, do not change the above loader control settings. To see the effect of changing the above settings in free-run mode, exit free-run mode, modify settings, then re-enter free-run mode.

Once a full image is loaded, or a single mirror block is loaded, the loader sends mirror reset request to the phased DMD reset state machine. The loader uses DLP6500 and DLP9000X look up tables, addressable by the current device row number, to determine on which rows to send phased reset requests.

4.2.4 DMD Reset State Machine

The phased reset state machine, *phase_dmd_reset_fsm.vhd*, processes mirror reset requests from the loader state machine. The state machine processes requests for **global** and **phased** resets, issuing the required block mode and block address commands to the DLPC910 controller.

After issuing a block command, the state machine monitors `rst_active` from the DLPC910 controller. Once `rst_active` goes high then returns to a low state, a mirror settling time interval is executed before indicating the reset request is complete.

A small FIFO is used to queue reset commands for cases where back-to-back phased reset requests occur. These are likely to occur when loading an image that has a resolution of 1 mirror block plus 1 DMD row. The reset for the block is requested, and then 1 row later another reset is requested for the next block.

4.2.5 DMD Load Parameters

The DMD load parameters module, *dmd_load_params.vhd*, selects load parameters from the package file, *dmd_params_pkg.vhd*, based on the DMD type. The selected data is used by the loader state machines for resolution and timing settings.

Refer to the DLPC910 data sheet for resolution and timing details.

4.2.6 Synchronization Pulse

Both the loader and DMD reset state machines synchronize their operation to an internal sync pulse in the Apps FPGA. The pulse is one `clkd` cycle in width. The pulse period varies by `dmd_type`. For the DLP9000X devices, there is one sync pulse every five `clkd` cycles. For the DLP6500, there is one sync pulse every eight `clkd` cycles. For each DMD type, the period indicates the row cycle time – the time required to load data and issue row-block command for a given DMD row.

The row command and block command state machines in the user DLP control logic also time their operation to the row cycle sync pulse.

4.3 User DLP Control

The Apps FPGA provides users the ability to load images to the DLPC910 - DMD chip set over the USB GPIF. Two VHDL modules provide the user control, *ugpif.vhd* and *dlp_ctl_ugpif.vhd*. A full resolution, 1-bit deep, dual port RAM buffer is provided to store an image.

The basic steps to display a user image on DMD are:

1. Write image data to user image buffer (USB GPIF FIFO write transfers, FIFO Write Transaction).
2. Send USB GPIF row commands to load buffer data to DLPC910 - DMD (User Row Command Register (0x002C)).
3. Send USB GPIF block commands to create mirror control pulses to display image data (User Block Command Register (0x0030)).

4.3.1 DLP6500 (1920 x 1080) User Image Display Example (Global)

The following steps describe how to display an image on a DLP6500 DMD:

Send the Buffer Data from the PC to the Apps FPGA

Step	Description	Register Address	Data
1	Set Apps FPGA user load mode (all zeros)	0x0040	enable bit = '0'
2	Set buffer starting write to zero and number of rows to 1080 (0x438)	0x0024	0x438
3	**Send the 1080p image to the user image buffer.** Requires 507 write bursts (FIFO Write Transaction)		
	Burst 1		256, 16-bit words
	Burst 2		256, 16-bit words
	⋮		
	Burst *n*		256, 16-bit words
	⋮		
	Burst 507		256, 16-bit words

Send Data from the Apps FPGA Data Buffer to the DLPC910 - DMD

Set image data modifiers if needed (ns_flip, comp_data). Use dip switch (SW2) on DLPLCRC910EVM board or use register 0x0010.

Step	Description	Register Address	Data
1	Send first row of data to DLPC910 - DMD	0x002C	0x00000003
2	Send the remaining 1079 rows	0x002C	0x04370001
3	Issue a global mirror reset to DLPC910 - DMD	0x0030	0x00000183

1. Send first row of image data to DLPC910 - DMD by writing 0x00000003 to row command register 0x002C.
2. Send all remaining rows by writing 0x04370001 to register 0x002C.
3. Issue a global mirror reset to DLPC910 - DMD by writing to register 0x0030.

4.3.2 DLP9000X (2560 x 1600) User Image Display Example (Global)

The following steps describe how to display an image on a DLP9000X or DLP9000XUV DMD:

Send the Buffer Data from the PC to the Apps FPGA

Step	Description	Register Address	Data
1	Set Apps FPGA user load mode (all zeros)	0x0040	enable bit = '0'
2	Set buffer starting write to zeros and number of rows to 1600 (0x640)	0x0024	0x640
3	**Send the WQXGA image to the user image buffer.** This requires 1000 write bursts (FIFO Write Transaction)		
	Burst 1		256, 16-bit words
	Burst 2		256, 16-bit words
	⋮		
	Burst *n*		256, 16-bit words
	⋮		
	Burst 1000		256, 16-bit words

Send Data from the Apps FPGA Data Buffer to the DLPC910 - DMD

Set image data modifiers if needed (ns_flip, comp_data). Use dip switch (SW2) on DLPLCRC910EVM board or use register 0x0010.

Step	Description	Register Address	Data
1	Send first row of data to DLPC910 - DMD	0x002C	0x00000003
2	Send the remaining 1599 rows	0x002C	0x063F0001
3	Issue a global mirror reset to DLPC910 - DMD	0x0030	0x00000183

4.3.3 Load4 - Using with DLP6500 DMD

Note

When using the DLP6500 DMD in load4 mode, the DMD row pointer does not accept pre-loads (rowmd = “10”). The row pointer can only be cleared (rowmd = “11”) and incremented (rowmd = “01”). Furthermore, there are 15 row pointer counts that do not result in data being written to DMD memory. So, to write to row 0 of the DMD, the row pointer must be cleared first (rowmd = “11”), followed by 14 dummy writes (rowmd = “01”).

4.3.4 USB GPIF FIFO Data Writes

An image is transferred to the user image buffer, by writing to the USB GPIF FIFO.

Images must be transferred from top left to bottom right. The first 16-bit word received, after the USB IF FIFO is reset (reg 0x0024), are placed in columns 0 through 15 of the DMD. The LSB of the first 16-bit word received goes to DMD column 0, MSB to column 15.

For a DLP6500 DMD, the transfer process continues until 120, 16-bit FIFO words are sent, which comprises a 1920 column row on the DMD. Then the next 120 words sent are placed on the next DMD row.

For DLP9000X or DLP9000XUV DMDs, the transfer process continues until 160, 16-bit words per DMD row which comprises a 2560 column row on the DMD. Then the next 160 words sent are placed on the next DMD row.

4.3.5 External Trigger

In lieu of issuing a global reset using a register write, an external trigger port is available on the DLPLCRC910EVM (Enable External Global Reset Trigger). When the trigger input is pulsed with an active high signal, a global reset block command is issued to the DLP component set.

4.4 USB GPIF (Operation)

The Apps FPGA provides access to the internal control-status registers through the USB General Purpose Interface (GPIF). The USB GPIF signals and bus transactions are described in USB GPIF (Interface). Registers are described in USB GPIF Registers. The top level VHDL module for the USB GPIF function is *ugpif.vhd*.

4.5 Clocks and Resets

The *clk_rst.vhd* module contains the clock PLLs for the Apps FPGA and also synchronizes reset signals for each clock domain.

4.5.1 Reference Clocks

The Apps FPGA receives two externally sourced clocks as reference clocks for internal PLLs:

- `sysclk_p/n` from the VC-707 EVM, 200 MHz oscillator
- `usb_if_clk` from the Infineon - Cypress USB interface chip

A third external clock source, spare_clk, driven from the DLPLCRC910EVM, is not used by the Apps FPGA. This clock is provided as both differential and single ended signal types.

4.5.2 Clk50 and Clk100

The general-purpose clock, `clk50`, is used for switch debounce and by the init-run-park state machine and is a 50 MHz free-running clock.

A free-running 100 MHz clock, `clk100`, is used by the test pattern generator.

Both clocks are generated by a single PLL (AMD - Xilinx IP), and are phase-aligned. The reference clock for the PLL is `sysclk_p/n`.

4.5.3 DLP Clocks

For internal DLP control logic, the Apps FPGA uses a clock that is ¼ the frequency of the DLPC910 high speed interface clock. This clock is named `clkd` in the VHDL code.

The DLPC910 high speed interface is a double data rate (DDR) interface, resulting in an 8:1, high-speed to internal, clock ratio. This allows the Apps FPGA to use AMD - Xilinx OSERDESE2 DDR primitives for data - control output. For every Apps FPGA internal clock rising edge, 8 data bits are loaded into the OSERDESE2 primitives, for shifting out at the high-speed DDR clock rate.

The Apps FPGA is required to support two different high-speed interface clock frequencies, 400 MHz and 480 MHz. Separate PLLs (AMD / Xilinx IP) are used so that exact clock frequencies can be created. One PLL generates 100/400 MHz clocks, the other PLL generates 120/480 MHz clocks. Both PLLs use the 200 MHz `sysclk_p/n` for their reference clock, and both PLLs run continuously.

Clock mux primitives are used to create both `clkd` and `clkd4x` (`clkd4x` is phase-aligned to `clkd` and is used by OSERDESE2 primitives). The 100 or 120 MHz clock is selected by a single clock mux to create `clkd`. The 400 or 480 MHz clock is selected by another clock mux to create `clkd4x`. Clock selection is performed once during initialization by the init-run-park state machine. After clock selection, the init-run-park state machine issues reset to the OSERDESE2 primitives and to the Apps FPGA logic.

4.5.4 USB GPIF Clock

The clock for the USB GPIF logic (`clku`) is generated from a PLL whose reference clock is `usb_if_clock`. The PLL output clock (`clku`) is phase-frequency aligned to the PLL reference clock. The primary purpose of the PLL is compensation for clock tree and buffer delay.

4.5.5 Logic Resets

Per AMD - Xilinx recommendation for 7 series FPGAs, all Apps FPGA clocked logic is coded with synchronous resets. There are a few exceptions to this rule, such as a loss of lock reset for a PLL, which is asserted asynchronously, de-asserted synchronously.

The synchronous reset signal names generally use names similar to the clock for a given domain. For example, `rstu` is the synchronous reset for the `clku` domain.

4.5.6 Clock Domain Crossings (CDC)

For data crossing clock domains, the Apps FPGA uses FIFOs and dual port RAM. Both components are created using AMD - Xilinx IP generation tools in Vivado.

For control signal & register value domain crossings, the Apps FPGA uses AMD - Xilinx parameterized macros.

Asynchronous signal inputs to the Apps FPGA are synchronized through three flip-flops. The three flip-flops have the ASYNC_REG attribute applied so that Vivado tools places them as close as possible to one another to minimize metastability.

4.6 Switch Debounce

The sw_debounce.vhd module debounces the switch inputs to the Apps FPGA. The following switch inputs are debounced:

- `evm_in_dip_sw(7:0)` from dip switch on DLPLCRC910EVM
- `vc707_in_dip_sw(7:0)` from dip switch on the VC-707 evaluation board
- `vc707_in_pb_sw(7:3)` from push button switches on the VC-707 evaluation board
- `resetz` from the push button switch on DLPLCRC910EVM (SW1)

5 USB GPIF Registers

Apps FPGA provides a read-write, 32-bit register interface that communicates over USB through an Infineon, CY7C68013A-128 USB Peripheral Controller. The single-chip peripheral controller resides on the DLPC910 EVM board.

Register read-write transaction protocol over the USB GPIF is described in USB GPIF (Interface).

5.1 Register Definitions

The following designations are used throughout this section of the document:

- R – designates a register that is read only.
- R/W – designates a register that is both readable and writable.

5.1.1 Status (0x000C)

Address	BITS	Description	Default(1)	R/W
0x000C	(31:16)	not used	0x0000	R
	15	not used	‘0’	R
	14	not used	‘0’	R
	13	not used	‘0’	R
	12	not used	‘0’	R
	11	not used	‘0’	R
	10	not used	‘0’	R
	9	not used	‘0’	R
	8	not used	‘0’	R
	7	not used	‘0’	R
	6	not used	‘0’	R
	5	DLOK: 1 = DLP (DMD) PLL locked. 0 = not locked	‘1’	R
	4	SLOK: 1 = system PLL locked. 0 = not locked	‘1’	R
	3	RACT: rst_active signal from DLPC910(2)	‘0’ or ‘1’	R
	2	IRQ – DMD_IRQ signal from DLPC910(2)	‘0’	R
	1	ECP: ecp2_finished signal from DLPC910(2)	‘1’	R
	0	CAL: 1 = apps FPGA is sending training patterns to DLPC910	‘0’	R

(1) The default column shows the status bit value for normal system operation after initialization is complete. Status values other than those in the default column indicate an issue with the system.
(2) This status bit is an apps FPGA pass through of a signal from the DLPC910.

5.1.2 Data Loading Control (0x0010)

Address	BITS	Description	Default	R/W
0x0010	(31:16)	not used	0x0000	R
	15	not used	N/A	
	14	not used	N/A	
	13	not used	N/A	
	12	not used	N/A	
	11	not used	N/A	
	10	not used	N/A	
	9	not used	N/A	
	8	ETRG: 1 = enable external global reset trigger; 0 = not enabled (1)	‘0’	R/W
	7	FLOT: 1 = float DMD mirrors with blkmd command; 0 = float not enabled(2)	‘0’	R/W
	6	not used	N/A	
	5	not used	N/A	
	4	CD: 1 = complement data before display; 0 = no complement(3)	‘0’	R/W
	3	WD : 1 = DLPC910 watch dog timer enabled; 0 = disabled(3)	‘0’	R/W
	2	NS: 1 = vertical flip of DMD image; 0 = no flip(3)	‘0’	R/W
	1	L4: 0 = enable DMD load4 mode; 1 = normal load(3)	‘1’	R/W
	0	PC: 1 = PC GUI control; 0 = dip switch control(3)	‘0’	R/W

(1) ETRG bit, when set to ‘1’, enables externally triggered global mirror reset for GUI-controlled DMD loads (when register 0x0040, bit 0 = ‘0’). This bit has no effect on apps FPGA image (test pattern) loader.
(2) When set to ‘1’, FLOT bit commands the apps FPGA image loader to float the DMD mirrors using blkmd = “11” and blkad = “11XX”. The system remains in this state until FLOT is set to zero. This bit has no effect on GUI controlled image loading.
(3) When PC bit is 0, bits (4:1) values are taken from dip switch SW2 on the DLPC910 EVM board. When PC bit is 1, bits (4:1) values are taken from this register.

Note

With the exception of the ETRG bit, the other control bits are used by both apps test pattern loading and the USB GPIF user image loading.

5.1.3 Test Pattern Control (0x0014)

Address	BITS	Description	Default	R/W
0x0014	(31:12)	TPG cycle interval(19:0) – defines test pattern display interval for the test pattern generator. LSB = 1 ms(1)	0x7D0	R/W
	11	not used		
	10	not used		
	9	not used		
	8	CEN – 1 = test pattern cycling. 0 = cycling disabled(2)	‘1’	R/W
	(7:0)	Test pattern select(7:0)(3)	0x00	R/W

(1) 1 ms is the step size for this value. A value of 0x000 results in unpredictable behavior.
(2) When pattern cycling is enabled, the test pattern generator cycles through writing patterns 0x00 through 0x0E into the test pattern buffer at the rate given by the TPG cycle interval. When pattern cycling is disabled, the test pattern generator writes the selected pattern into the test pattern buffer.
(3) Test pattern select chooses test pattern to display when test pattern cycling is disabled:

- 0x00 – full ON pattern

- 0x01 – full OFF pattern
- 0x02 – ANSI checkerboard
- 0x03 – single pixel grid line pattern with single pixel outside border
- 0x04 – west to east single pixel diagonal lines
- 0x05 – east to west single pixel diagonal lines
- 0x06 – horizontal lines
- 0x07 – vertical lines
- 0x08 – load4 checkerboard
- 0x09 – checkerboard
- 0x0A – inverted checkerboard
- 0x0B – 1x1 horizontal lines
- 0x0C – 1x1 vertical lines
- 0x0D – random noise pattern
- 0x0E – block boundary checkerboard
- 0x0F – user defined pattern (loaded through USB/GPIF)
- *0xFF-0x10 – not used*

5.1.4 Test Row Address (0x0018) - [Unused]

Address	BITS	Description	Default	R/W
0x0018	(31:11)	Not used	zeros	R/W
	(10:0)	Test row address(1)	0x000	R/W

(1) The test row address field is unused by apps FPGA. The test, force to 1's function is supported by the F1S bit in register 0x002C.

5.1.5 Loader Reset Type (0x001C)

Address	BITS	Description	Default	R/W
0x001C	(31:4)	Not used	zeros	R/W
	(3:0)	Reset type(3:0)(1)	0x02	R/W

(1) Reset type instructs the apps FPGA loader to use the specified mirror reset type:
- 0x0 = phased x1
- 0x1 = phased x2
- 0x2 = global
- 0x3 = phased x4

5.1.6 Type and Version (0x0020)

Address	BITS	Description	Default	R/W
0x0020	(31:24)	EVM_DIPSW(7:0)(1)	Read from DLPLCRC910EVM	R
	(23:7)	Not used	zeros	R
	(6:4)	DDC_VER(2:0)(2)	Read from DLPC910	R
	(3:0)	DMD_TYPE(3:0)(3)	Read from DLPC910	R

(1) EVM_DIPSW field is the 8-bit value signaled to the Apps FPGA from the 8-position dip switch on the DLPLCRC910EVM board. Out of box logic default is 0x03. See DLPLCRC910EVM Dip Switch (SW2).
(2) DDC_VER field is the 3-bit value signaled to the Apps FPGA at the DDC version input pins. The Apps FPGA does not use this value.
(3) DMD_TYPE field is the 4-bit value signaled to the Apps FPGA at the DMD type input pins. The Apps FPGA uses DMD type to determine DMD resolution and clocks per row cycle. Additional information is found in the DLPC910 data sheet.

5.1.7 User Image Buffer Write Settings (0x0024)

Address	BITS	Description	Default	R/W
0x0024	(31:16)	buf_wstart_row(15:0)(1) (2)	0x0000	R/W
	(15:11)	Not used	0x00	R/W
	(10:0)	buf_wstart_numrows(10:0)(1) (3)	1	R/W

(1) This register contains the user image buffer write settings. Writing this register primes the user image buffer write state machine to begin monitoring the USB / GPIF FIFO for an image transfer to the user image buffer. The first row received over USB / GPIF is written

to buf_wstart_row of the user image buffer. The buffer write state machine accepts buf_wstart_numrows of data, after which the buffer stops writing data until a new write to this register is performed. In addition, writing to this register resets the USB / GPIF FIFO.

(2) Valid values for buf_wstart_row are 0 through (TOTAL_ROWS_ON_DMD – 1). Buf_wstart_row must be less than or equal to TOTAL_ROWS_ON_DMD – buf_wstart_numrows.

(3) Buf_wstart_numrows valid values are 1 thru TOTAL_ROWS_ON_DMD.

5.1.8 USB GPIF FIFO Read Burst Size (0x0028) - [Obsolete]

Address	BITS	Description	Default	R/W
0x0028	(31:10)	Not used	zeros	R/W
	(9:0)	FIFO read burst size - obsolete(1)	N/A	R/W

(1) This register is no longer used by the apps FPGA logic. The USB GPIF logic still supports reads/writes for this register, but the contents are not currently used.

5.1.9 User Row Command Register (0x002C)

Address	BITS	Description	Default	R/W
0x002C	(31:29)	Not used	zeros	R/W
	28	F1S(1)	0	R/W
	27	Not used	0	R/W
	(26:16)	Numrows(2) (3)	1	R/W
	15	Not used	0	R/W
	(14:4)	ROWAD(2) (3)	0x000	R/W
	(3:2)	Not used	“00”	R/W
	(1:0)	ROWMD(2) (3)	“00”	R/W

(1) When F1S is ‘1’ (F1S = force ones), all 1’s data is sent to the DLPC910 for the given row cycles.

(2) ROWMD has the following effect on user image buffer reads:

- “00” – no action.
- “01” – increments buffer read address counter then sends addressed buffer row data to DMD. Continues this until numrows (bits 26:16) have been sent.
- “10” – loads apps DLP ROWAD counter with ROWAD. Sends ROWAD to DLPC910. Loads buffer read address counter with ROWAD. Sends addressed buffer row to DLPC910.
- “11” – Sends ROWAD = zeros to DLPC910. Clears buffer read address counter to zeros. Sends buffer row 0 data to DLPC910.

(3) The *User Row Command* register is used to move data from the user image buffer to the DMD. When the register is written, ROWMD and ROWAD are forwarded to the DLPC910 controller along with data read from the user image buffer. Additional details on the function of these row control signals can be found in the DLPC910 data sheet.

Apps FPGA continuously sends no-op row commands to the DLPC910 controller when the controller is not responding to writes to this register.

5.1.10 User Block Command Register (0x0030)

Address	BITS	Description	Default	R/W
0x0030	(31:9)	Not used	zeros	R/W
	8	RST2BLKZ(1)	1	R/W
	(7:4)	BLKAD(1)	0x0	R/W
	(3:2)	Not used	zeros	R/W
	(1:0)	BLKMD(1)	“00”	R/W

(1) Writing to the user block command register instructs apps FPGA to forward the block command in this register to the DLPC910 controller. Block command details are described in the DLPC910 data sheet. Forwarded block commands include the appropriate synchronization with DCLK and DVALID.

5.1.11 Loader Row Control (0x0034)

Address	BITS	Description	Default	R/W
0x0034	(31:27)	Not used	zeros	R/W
	(26:16)	Start row[1]	0x000	R/W
	(15:11)	Not used	zeros	R/W
	(10:0)	Load rows[2]	1	R/W

(1) The start row field tells the test pattern loader on which DMD row to start loading the image.
(2) The load rows field tells the test pattern loader how many rows of test pattern image to send to the DLP chip set.

5.1.12 Loader Load Interval (0x0038)

Address	BITS	Description	Default	R/W
0x0038	(31:10)	Not used	zeros	R/W
	(9:0)	Load interval. LSB = 1 us[1]	0x190	R/W

(1) The load interval field sets the interval for triggering test pattern buffer reads. The default interval is 400 us. The LSB is the step size.

Note

If the load interval is set to a period of time shorter than the configured DLP chip set load time, then the loader ignores any triggers that occur when busy. Values less than 50 us are not recommended.

5.1.13 Loader Expose Time (0x003C)

Address	BITS	Description	Default	R/W
0x003C	(31:16)	Not used	zeros	R/W
	(15:0)	Expose time counts[1]	0x0000	R/W

(1) When a trigger occurs, the Loader sends the pattern buffer to the DMD and performs a reset to display the pattern. Expose time is wait time added after the reset is complete before the Loader waits for the next trigger. Expose time is measured in row cycle counts:

- For DLP9000X and DLP9000XUV DMDs, row cycles are 20 dclk cycles long at 400 MHz or 480 MHz (50 ns or 41.67 ns).
- For DLP65000 DMDs, row cycles are 32 dclk cycles long at 400 MHz (80 ns).

For example with the default *Load Interval* of 400 us no change will be seen in the pattern load frequency until the total of load time plus reset time plus expose time exceeds a multiple of 400 us. Therefore, the effective step size is the Loader Load Interval (0x0038).

5.1.14 Address Write (0x003F) - [Unused]

Address	BITS	Description	Default	R/W
0x003F	(31:0)	USB GPIF unused address write data[1]	zeros	R/W

(1) The Apps FPGA USB GPIF logic places 32-bit data written to unused register addresses into this register.

5.1.15 Loader Control (0x0040)

Address	BITS	Description	Default	R/W
0x0040	(31:2)	Not used	zeros	R/W
	1	Free-run[1]	0	R/W
	0	Loader enable[2]	1	R/W

(1) When set to ‘1’, free-run bit tells the test pattern loader to run at maximum rate, ignoring the load trigger.
(2) When loader enable is set to ‘1’, apps FPGA test pattern loader sends test pattern images to the DLP chip set. When ‘0’, test pattern loading stops, and DLP chip set is controlled through the USB GPIF. When loader enable bit transitions from ‘1’ to ‘0’, the DMD image is cleared using block clear commands, after which the user is given control of the DLP chip set through the USB GPIF.

5.1.16 Park [PWR_FLOAT] (0x0044)

Address	BITS	Description	Default	R/W
0x0044	(31:1)	Not used	zeros	R/W
	0	Park (PWR_FLOAT)(1)	0	R/W

(1) Writing a value of ‘1’ to the park bit commands apps FPGA to drive `PWR_FLOATZ` signal low to the DLPC910. When park bit state is changed from ‘1’ to ‘0’, apps FPGA drives `PWR_FLOATZ` high, resets (`CTL_RSTZ` = ‘0’) and re-initializes the DLPC910. The Apps FPGA logic is also reset.

5.1.17 External Trigger Status (0x0048)

Address	BITS	Description	Default	R/W
0x0048	(31:1)	Not used	zeros	R/W
	0	External trigger received(1)	0	R/W

(1) In GUI script mode, bit 0, when ‘1’, indicates an external trigger was received. This bit can be cleared by writing any value to this register.

5.1.18 FPGA Build Date (0x0080)

Address	BITS	Description	Default	R/W
0x0080	(31:16)	Build year in binary-coded decimal (BCD) format. 1 nibble for each digit in the year (for example, 0x2022 == year 2022)		R
	(15:8)	Build month in BCD format. 1 nibble for each digit in the month. (for example, 0x11 == month 11 == November)		R
	(7:0)	Build day of month in BCD format. (for example, 0x12 == 12th day of the month)		R

5.1.19 Major-Minor Revision (0x0084)

Address	BITS	Description	Default	R/W
0x0084	(31:16)	Not used	zeros	R
	(15:8)	1-byte major revision, hexadecimal		R
	(7:0)	1-byte minor revision, hexadecimal		R

5.1.20 Fixed Value FPGA Identifier (0x0088)

Address	BITS	Description	Default	R/W
0x0088	(31:0)	32-bit fixed FPGA identifier = 0x000AC910	0x000AC910	R

5.1.21 Test Register (0x008C)

Address	BITS	Description	Default	R/W
0x008C	(31:0)	USB GPIF 32-bit test read/write value(1)	zeros	R/W

(1) This register is used for USB GPIF testing only. The contents of this register do not affect the Apps FPGA operation.

6 FPGA Configuration

The AMD - Xilinx VC707 board uses a 16-bit flash interface for FPGA configuration. The flash memory is configured over JTAG using AMD - Xilinx Vivado lab tools.

CAUTION

TI recommends that the flash memory be written with the Apps FPGA configuration memory file before connecting the VC-707 board to the DLPLCRC910EVM avoiding potential FMC connector I/O conflicts between the factory loaded VC-707 FPGA configuration and DLPC910 controller.

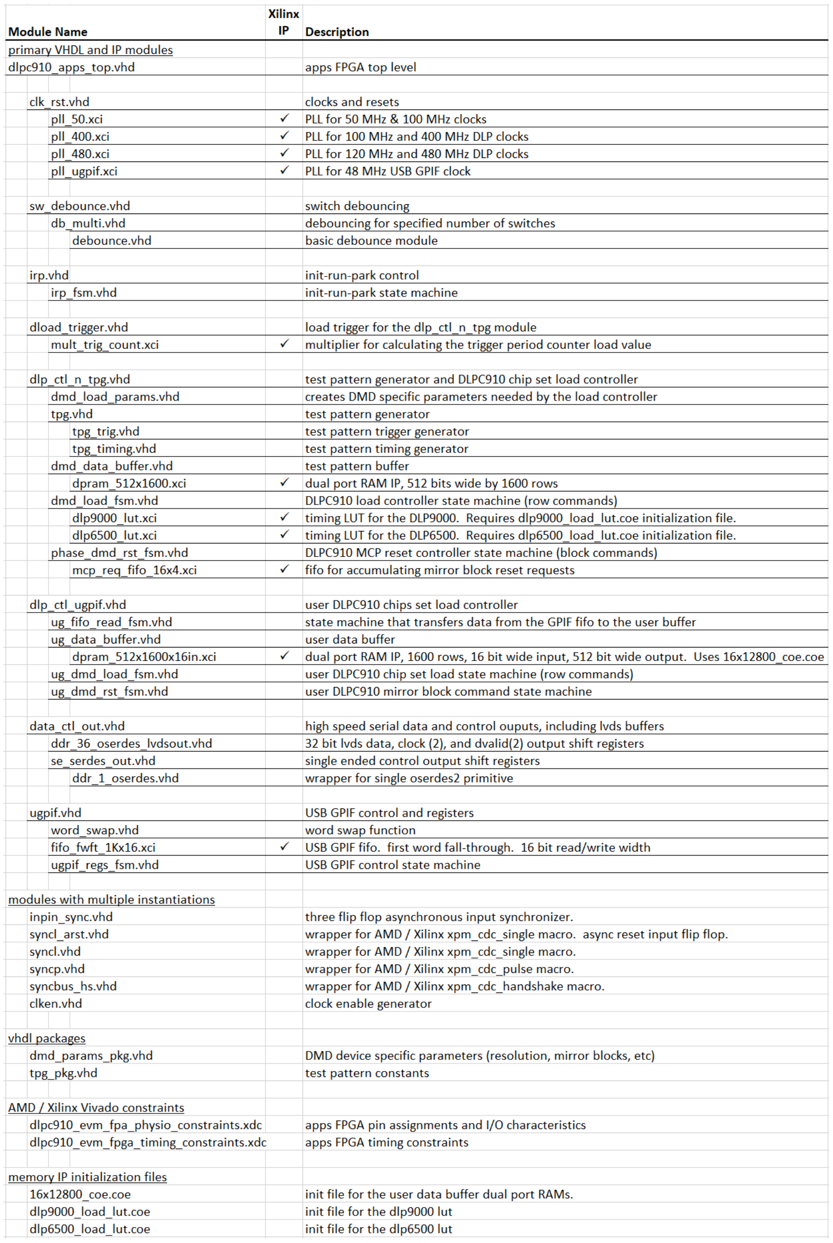
7 Apps FPGA Source Files and Compilation

7.1 Design Tools

AMD Xilinx Vivado, version 2019.2 was used for the Apps FPGA synthesis, implementation, and simulation. The SW tool license provided with the VC-707 Evaluation Kit enables all necessary Vivado functionality and IP to build the apps FPGA and to simulate with the provided VHDL test bench files.

7.2 Source Files

The source files required to build the Apps FPGA are listed, with short descriptions (shown in Figure 7-1):



Module Name	Xilinx IP	Description
primary VHDL and IP modules		
dlpc910_apps_top.vhd		apps FPGA top level
clk_rst.vhd		clocks and resets
pll_50.xci	✓	PLL for 50 MHz & 100 MHz clocks
pll_400.xci	✓	PLL for 100 MHz and 400 MHz DLP clocks
pll_480.xci	✓	PLL for 120 MHz and 480 MHz DLP clocks
pll_ugpif.xci	✓	PLL for 48 MHz USB GPIF clock
sw_debounce.vhd		switch debouncing
db_multi.vhd		debouncing for specified number of switches
debounce.vhd		basic debounce module
irp.vhd		init-run-park control
irp_fsm.vhd		init-run-park state machine
dload_trigger.vhd		load trigger for the dlp_ctl_n_tpg module
mult_trig_count.xci	✓	multiplier for calculating the trigger period counter load value
dlp_ctl_n_tpg.vhd		test pattern generator and DLPC910 chip set load controller
dmd_load_params.vhd		creates DMD specific parameters needed by the load controller
tpg.vhd		test pattern generator
tpg_trig.vhd		test pattern trigger generator
tpg_timing.vhd		test pattern timing generator
dmd_data_buffer.vhd		test pattern buffer
dpram_512x1600.xci	✓	dual port RAM IP, 512 bits wide by 1600 rows
dmd_load_fsm.vhd		DLPC910 load controller state machine (row commands)
dlp9000_lut.xci	✓	timing LUT for the DLP9000. Requires dlp9000_load_lut.coe initialization file.
dlp6500_lut.xci	✓	timing LUT for the DLP6500. Requires dlp6500_load_lut.coe initialization file.
phase_dmd_rst_fsm.vhd		DLPC910 MCP reset controller state machine (block commands)
mcp_req_fifo_16x4.xci	✓	fifo for accumulating mirror block reset requests
dlp_ctl_ugpif.vhd		user DLPC910 chips set load controller
ug_fifo_read_fsm.vhd		state machine that transfers data from the GPIF fifo to the user buffer
ug_data_buffer.vhd		user data buffer
dpram_512x1600x16in.xci	✓	dual port RAM IP, 1600 rows, 16 bit wide input, 512 bit wide output. Uses 16x12800_coe.coe
ug_dmd_load_fsm.vhd		user DLPC910 chip set load state machine (row commands)
ug_dmd_rst_fsm.vhd		user DLPC910 mirror block command state machine
data_ctl_out.vhd		high speed serial data and control ouputs, including lvds buffers
ddr_36_oserdes_lvdsout.vhd		32 bit lvds data, clock (2), and dvalid(2) output shift registers
se_serdes_out.vhd		single ended control output shift registers
ddr_1_oserdes.vhd		wrapper for single oserdes2 primitive
ugpif.vhd		USB GPIF control and registers
word_swap.vhd		word swap function
fifo_fwft_1Kx16.xci	✓	USB GPIF fifo. first word fall-through. 16 bit read/write width
ugpif_regs_fsm.vhd		USB GPIF control state machine
modules with multiple instantiations		
inpin_sync.vhd		three flip flop asynchronous input synchronizer.
syncl_arst.vhd		wrapper for AMD / Xilinx xpm_cdc_single macro. async reset input flip flop.
syncl.vhd		wrapper for AMD / Xilinx xpm_cdc_single macro.
syncp.vhd		wrapper for AMD / Xilinx xpm_cdc_pulse macro.
syncbus_hs.vhd		wrapper for AMD / Xilinx xpm_cdc_handshake macro.
clken.vhd		clock enable generator
vhdl packages		
dmd_params_pkg.vhd		DMD device specific parameters (resolution, mirror blocks, etc)
tpg_pkg.vhd		test pattern constants
AMD / Xilinx Vivado constraints		
dlpc910_evm_fpa_physio_constraints.xdc		apps FPGA pin assignments and I/O characteristics
dlpc910_evm_fpga_timing_constraints.xdc		apps FPGA timing constraints
memory IP initialization files		
16x12800_coe.coe		init file for the user data buffer dual port RAMs.
dlp9000_load_lut.coe		init file for the dlp9000 lut
dlp6500_load_lut.coe		init file for the dlp6500 lut

Figure 7-1. Source Files

7.2.1 Primary VHDL and IP Modules

The primary VHDL modules (*.vhd*) provide the main DLP control functionality of the Apps FPGA. AMD - Xilinx IP modules (*.xci*) are used for PLLs, FIFOs, memories, and a multiplier.

7.2.2 Modules with Multiple Instantiations

Grouped under this heading, are six VHDL modules (.vhd) that are used by several of the primary VHDL modules, with multiple instantiations. There are five clock-domain-crossings (CDC) and one clock enable generator.

The CDC modules are wrappers for Xilinx parameterized CDC macros – using these macros simplifies CDC analysis, as Vivado automatically recognizes them as CDC functions.

7.2.3 VHDL Packages

The *dmd_params_pkg.vhd* file contains DMD specific resolution and timing parameters for the three DMDs supported by the Apps FPGA. The parameters are used by the test pattern loader module (*dlp_ctl_n_tpg.vhd*).

The *tpg_pkg.vhd* file contains several test pattern data constants used by the test pattern generator module (*tpg.vhd*). Data constants are supplied to support DLP9000X, DLP9000XUV, and DLP6500 DMDs.

7.2.4 Vivado Constraints

In addition to constraints created by the Vivado IP generator, two constraint files are required by the Apps FPGA project. The physical constraints file contains I/O pins and characteristics. The timing constraints file provides clock and timing constraints and directives.

7.2.5 Memory IP Initialization Files

The two Look Up Tables (LUT) initialization (*.coe*) files are used to populate look-up tables for the test pattern loader state machine. The 16x12800_coe file is used to initialize the first 40 DMD Rows of the user image buffer with a wide vertical stripe pattern. The remaining user image buffer rows are initialized with vertical stripes (0x00FF) by settings within the Vivado block memory generator IP tool for the dpram_512x1600x16 in IP component.

7.2.5.1 Look Up Tables

The Excel spreadsheet *dlpc910_apps_fpga_luts_coe.xlsx* was used to create the look up tables for DLP9000X, DLP9000XUV, and DLP6500 DMDs. The look-up tables are addressed by the DMD row that is being loaded. For each DMD row, the LUTs indicate if the row address is for last row in a mirror block. The last row indicator is used by the loader state machine to request an MCP reset for the block that just completed loading. NS FLIP, phased reset mode (x1, x2, x4), and load4z are all accommodated by the table. The block address is also included in the table for forwarding to the DLPC910.

The excel file contains four worksheets, two for DLP9000X - DLP9000XUV DMDs and two for DLP6500 DMDs. For each DMD, the first worksheet creates the LUT data per row address. The second worksheet simply copies the LUT data for export to a CSV file. Then add the .coe header and footer information using a text editor.

7.3 Building the Apps FPGA Code

7.3.1 Source Code

The <source code directory> contains the source files and .tcl script for creating the dlpc910_apps FPGA project. The contains:

- source folders and files
- .tcl file for creating the project

7.3.1.1 Source Folder

The source folder contains four sub-folders:

- constr – folder for the Vivado physical and timing constraint files
- ip – folder for the Virtex 7 ip (PLLs, memories, FIFOs, etc)
- rtl – folder for the VHDL modules and packages
- tbench – folder for VHDL test bench files

In addition, the following four files are included in the source folder:

- 16x12800_coe.coe
- dlp6500_load_lut.coe

- dlp9000_load_lut.coe

Note
File for DLP9000X and DLP9000XUV DMDs only.

- dlpc910_apps_fpga_luts_coe.xlsx

Descriptions of all the source files are given in section Source Files.

7.3.2 Creating the Vivado Project

Use the following steps to create the dlpc910_apps FPGA Vivado project:

1. Create a folder for the project
2. Copy the *source* directory from the <source dir> in installation> into the created folder. The source folder and *.tcl* file (described above) now appear the folder of step 1.
3. Open Vivado 2019.2 Tcl shell and navigate to the folder of step 1.
4. Run the *.tcl* file from the Tcl shell: source *<tcl_filename>.tcl*
5. The *.tcl* script creates the Vivado project in folder dlpc910_apps.
6. Close the Tcl shell and start Vivado 2019.2 program
7. Under Quick Start, select Open Project > and navigate into the dlpc910_apps folder that was created in step 5.
8. Select the *dlpc910_apps.xpr* file and click OK. The dlpc910_apps FPGA project opens in the Vivado project manager GUI.

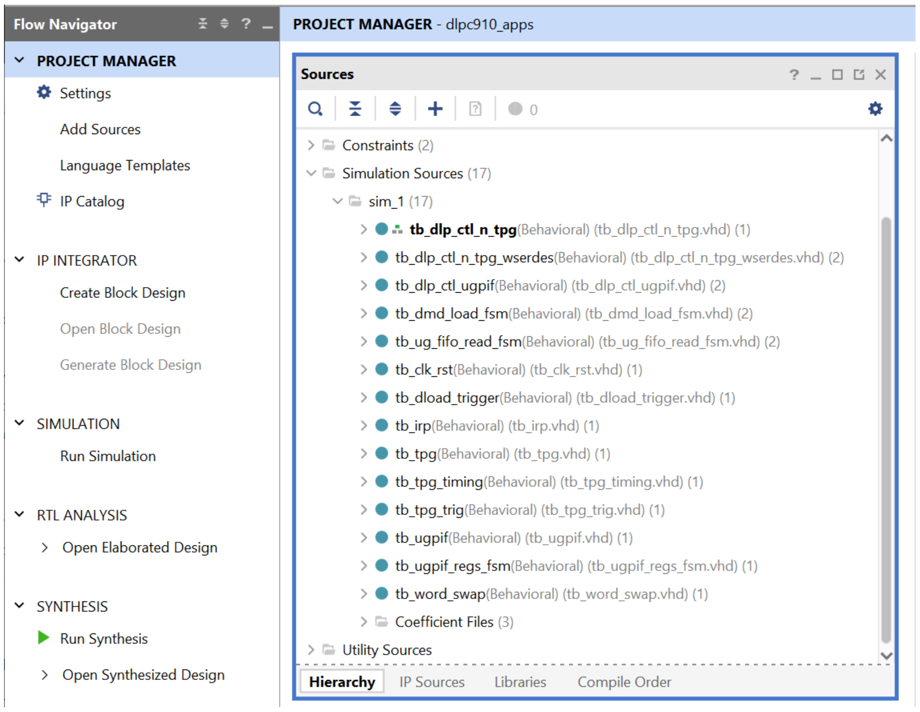
7.3.3 Compiling the Design

Once the Vivado project has been created, and opened in the project manager GUI, the DLPC910_apps FPGA can be built by clicking on <Generate Bitstream> in the flow navigator pane. The project manager detects that there are no synthesis and implementation results, and runs the processes needed to implement the design.

7.3.4 Simulation

7.3.4.1 Test Benches

The source code archive, described in section Source Code, also includes a folder that contains test benches for the primary VHDL modules in the apps FPGA. When the project .tcl script is executed to recreate the apps FPGA project, the project .tcl script adds the test bench files to the project. The test bench sources are viewable in the Vivado project manager GUI, sources hierarchy window. See Figure 7-2.

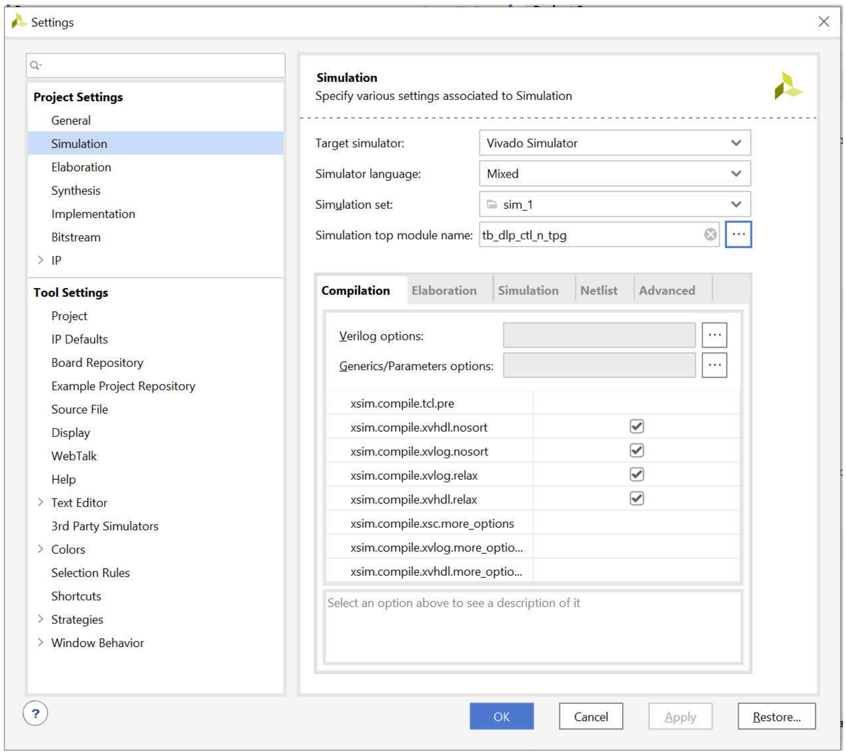


Constraints (2)
Simulation Sources (17)
sim_1 (17)
tb_dlp_ctl_n_tpg(Behavioral) (tb_dlp_ctl_n_tpg.vhd) (1)
tb_dlp_ctl_n_tpg_wserdes(Behavioral) (tb_dlp_ctl_n_tpg_wserdes.vhd) (2)
tb_dlp_ctl_ugpif(Behavioral) (tb_dlp_ctl_ugpif.vhd) (2)
tb_dmd_load_fsm(Behavioral) (tb_dmd_load_fsm.vhd) (2)
tb_ug_fifo_read_fsm(Behavioral) (tb_ug_fifo_read_fsm.vhd) (2)
tb_clk_rst(Behavioral) (tb_clk_rst.vhd) (1)
tb_dload_trigger(Behavioral) (tb_dload_trigger.vhd) (1)
tb_irp(Behavioral) (tb_irp.vhd) (1)
tb_tpg(Behavioral) (tb_tpg.vhd) (1)
tb_tpg_timing(Behavioral) (tb_tpg_timing.vhd) (1)
tb_tpg_trig(Behavioral) (tb_tpg_trig.vhd) (1)
tb_ugpif(Behavioral) (tb_ugpif.vhd) (1)
tb_ugpif_regs_fsm(Behavioral) (tb_ugpif_regs_fsm.vhd) (1)
tb_word_swap(Behavioral) (tb_word_swap.vhd) (1)
Coefficient Files (3)
Utility Sources

Figure 7-2. Test Benches

7.3.4.2 Steps to Simulate a Module

In the flow navigator pane (Figure 7-2), right-click on SIMULATION, and select Simulation Settings. The settings dialog as shown in Figure 7-3 appears. Browse to select one of the test bench files of interest so that the file appears in the Simulation top module name: box. Then click on OK.



Simulation
Specify various settings associated to Simulation

Target simulator: Vivado Simulator
Simulator language: Mixed
Simulation set: sim_1
Simulation top module name: tb_dlp_ctl_n_tpg

Verilog options:
Generics/Parameters options:

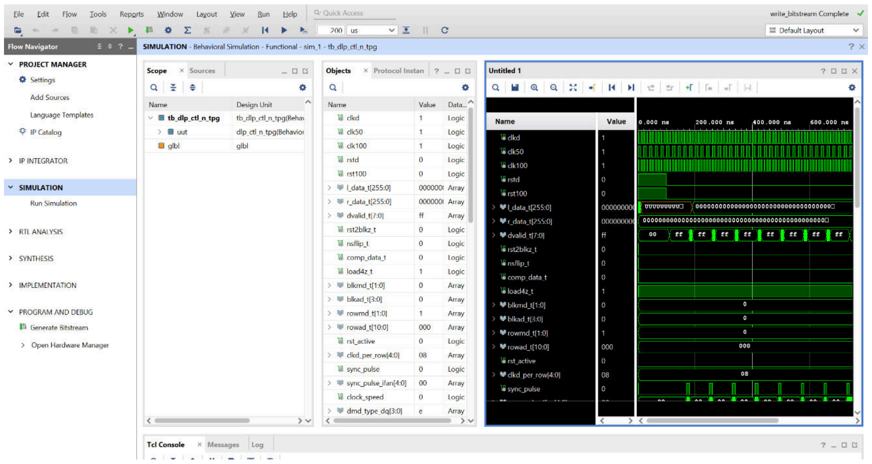
xsim.compile.tcl.pre	
xsim.compile.xvhdl.nosort	✓
xsim.compile.xvlog.nosort	✓
xsim.compile.xvlog.relax	✓
xsim.compile.xvhdl.relax	✓
xsim.compile.xsc.more_options	
xsim.compile.xvlog.more_optio...	
xsim.compile.xvhdl.more_optio...	

Select an option above to see a description of it

Figure 7-3. Settings Dialog

Next, select Run Simulation on the flow navigator pane, then select Run Behavioral Simulation . The test bench and unit under test (UUT) compile and Vivado changes to simulation context, with a waveform window as shown in Figure 7-4.

The user can then work with the default waveform window, or open an associated waveform configuration file (under the File pull-down menu, select Simulation Waveform). A waveform configuration file is supplied for each test bench to give a starting point for working with module simulations.



Name	Design Unit
tb_dlp_ctl_n_tpg	tb_dlp_ctl_n_tpg(Behav
uut	dlp_ctl_n_tpg(Behavior
glbl	glbl

Name	Value	Data...
clkd	1	Logic
clk50	1	Logic
clk100	1	Logic
rstd	0	Logic
rst100	0	Logic
l_data_t[255:0]	000000(	Array
r_data_t[255:0]	000000(	Array
dvalid_t[7:0]	ff	Array
rst2blkz_t	0	Logic
nsflip_t	0	Logic
comp_data_t	0	Logic
load4z_t	1	Logic
blkmd_t[1:0]	0	Array
blkad_t[3:0]	0	Array
rowmd_t[1:0]	1	Array
rowad_t[10:0]	000	Array
rst_active	0	Logic
clkd_per_row[4:0]	08	Array
sync_pulse	0	Logic
sync_pulse_ifan[4:0]	00	Array
clock_speed	0	Logic
dmd_type_dq[3:0]	e	Array

Figure 7-4. Vivado Waveform Window

8 Related Documentation from Texas Instruments

Component data sheets, technical documents, design documents, and ordering information related to the DLPC910 are found at the following links:

DLPC910 Digital Controller Product Folder

LightCrafter DLPC910 EVM Product Folder

DLP6500FLQ DMD Product Folder

DLP6500FYE DMD Product Folder

DLP LightCrafter DLP6500FLQ DMD EVM Product Folder

DLP9000X DMD Product Folder

DLP LightCrafter DLP9000X DMD EVM Product Folder

DLP9000XUV DMD Product Folder

DLP LightCrafter DLP9000XUV DMD EVM Product Folder

AMD/Xilinx VC707 Evaluation Board for the Virtex-7 FPGA – User's Guide (UG855)

AMD/Xilinx VC707 Evaluation Platform - Schematic

9 Appendix

9.1 Abbreviations and Acronyms

The following lists abbreviations and acronyms used in this manual.

Apps FPGA	AMD Xilinx Virtex 7 FPGA on the VC-707 EVM or similar board for customer applications
BCD	Binary-Coded Decimal
CDC	Clock Domain Crossing
DDR	Double Data Rate
DLL	Dynamic Link Library
DMD	Digital Micromirror Device
FCC	Federal Communications Commission
EVM	Evaluation Module (Board)
FIFO	First In First Out
FMC	FPGA Mezzanine Connector
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
FW	Firmware
GPIF	General Purpose Interface
GPIO	General Purpose Input Output
GUI	Graphical user interface
HPC	High Pin Count
HW	Hardware
IO	Input Output
JTAG	Joint Test Action Group
LED	Light Emitting Diode
LUT	Look Up Table
MCP	Mirror Clocking Pulse
PBC	Processor Bus Control
SPI	Serial-Peripheral Interface
TPG	Test Pattern Generator

SW	Switch
USB	Universal Serial Bus
UUT	Unit Under Test
VHDL	Verification and Hardware Description Language

9.2 Information About Cautions and Warnings

This book contains cautions and warnings.

CAUTION

This is a description of a caution statement: A caution statement describes a situation that can potentially damage your software or equipment.

WARNING

This is a description of a warning statement: A warning statement describes a situation that can potentially cause harm to you.

The information in a caution or a warning is provided for your protection. Please read each caution and warning carefully.

FCC Warning: This equipment is intended for use in a laboratory test environment only. The equipment generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to subpart J of part 15 of FCC rules, which are designed to provide reasonable protection against radio frequency interference. Operation of this equipment in other environments can cause interference with radio communications, in which case the user at his/her own expense is required to take whatever measures can be required to correct this interference.

STANDARD TERMS FOR EVALUATION MODULES

1. *Delivery:* TI delivers TI evaluation boards, kits, or modules, including any accompanying demonstration software, components, and/or documentation which may be provided together or separately (collectively, an “EVM” or “EVMs”) to the User (“User”) in accordance with the terms set forth herein. User's acceptance of the EVM is expressly subject to the following terms.

   1.1 EVMs are intended solely for product or software developers for use in a research and development setting to facilitate feasibility evaluation, experimentation, or scientific analysis of TI semiconductors products. EVMs have no direct function and are not finished products. EVMs shall not be directly or indirectly assembled as a part or subassembly in any finished product. For clarification, any software or software tools provided with the EVM (“Software”) shall not be subject to the terms and conditions set forth herein but rather shall be subject to the applicable terms that accompany such Software

   1.2 EVMs are not intended for consumer or household use. EVMs may not be sold, sublicensed, leased, rented, loaned, assigned, or otherwise distributed for commercial purposes by Users, in whole or in part, or used in any finished product or production system.

2 *Limited Warranty and Related Remedies/Disclaimers*:

   2.1 These terms do not apply to Software. The warranty, if any, for Software is covered in the applicable Software License Agreement.

   2.2 TI warrants that the TI EVM will conform to TI's published specifications for ninety (90) days after the date TI delivers such EVM to User. Notwithstanding the foregoing, TI shall not be liable for a nonconforming EVM if (a) the nonconformity was caused by neglect, misuse or mistreatment by an entity other than TI, including improper installation or testing, or for any EVMs that have been altered or modified in any way by an entity other than TI, (b) the nonconformity resulted from User's design, specifications or instructions for such EVMs or improper system design, or (c) User has not paid on time. Testing and other quality control techniques are used to the extent TI deems necessary. TI does not test all parameters of each EVM. User's claims against TI under this Section 2 are void if User fails to notify TI of any apparent defects in the EVMs within ten (10) business days after delivery, or of any hidden defects with ten (10) business days after the defect has been detected.

   2.3 TI's sole liability shall be at its option to repair or replace EVMs that fail to conform to the warranty set forth above, or credit User's account for such EVM. TI's liability under this warranty shall be limited to EVMs that are returned during the warranty period to the address designated by TI and that are determined by TI not to conform to such warranty. If TI elects to repair or replace such EVM, TI shall have a reasonable time to repair such EVM or provide replacements. Repaired EVMs shall be warranted for the remainder of the original warranty period. Replaced EVMs shall be warranted for a new full ninety (90) day warranty period.

WARNING

Evaluation Kits are intended solely for use by technically qualified, professional electronics experts who are familiar with the dangers and application risks associated with handling electrical mechanical components, systems, and subsystems.

User shall operate the Evaluation Kit within TI’s recommended guidelines and any applicable legal or environmental requirements as well as reasonable and customary safeguards. Failure to set up and/or operate the Evaluation Kit within TI’s recommended guidelines may result in personal injury or death or property damage. Proper set up entails following TI’s instructions for electrical ratings of interface circuits such as input, output and electrical loads.

NOTE:

EXPOSURE TO ELECTROSTATIC DISCHARGE (ESD) MAY CAUSE DEGREDATION OR FAILURE OF THE EVALUATION KIT; TI RECOMMENDS STORAGE OF THE EVALUATION KIT IN A PROTECTIVE ESD BAG.

3 *Regulatory Notices:*

3.1 *United States*

3.1.1 *Notice applicable to EVMs not FCC-Approved:*

FCC NOTICE: This kit is designed to allow product developers to evaluate electronic components, circuitry, or software associated with the kit to determine whether to incorporate such items in a finished product and software developers to write software applications for use with the end product. This kit is not a finished product and when assembled may not be resold or otherwise marketed unless all required FCC equipment authorizations are first obtained. Operation is subject to the condition that this product not cause harmful interference to licensed radio stations and that this product accept harmful interference. Unless the assembled kit is designed to operate under part 15, part 18 or part 95 of this chapter, the operator of the kit must operate under the authority of an FCC license holder or must secure an experimental authorization under part 5 of this chapter.

3.1.2 *For EVMs annotated as FCC – FEDERAL COMMUNICATIONS COMMISSION Part 15 Compliant:*

CAUTION

This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

FCC Interference Statement for Class A EVM devices

NOTE: This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

FCC Interference Statement for Class B EVM devices

NOTE: This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- *Reorient or relocate the receiving antenna.*
- *Increase the separation between the equipment and receiver.*
- *Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.*
- *Consult the dealer or an experienced radio/TV technician for help.*

3.2 *Canada*

3.2.1 *For EVMs issued with an Industry Canada Certificate of Conformance to RSS-210 or RSS-247*

Concerning EVMs Including Radio Transmitters:

This device complies with Industry Canada license-exempt RSSs. Operation is subject to the following two conditions:

(1) this device may not cause interference, and (2) this device must accept any interference, including interference that may cause undesired operation of the device.

Concernant les EVMs avec appareils radio:

Le présent appareil est conforme aux CNR d'Industrie Canada applicables aux appareils radio exempts de licence. L'exploitation est autorisée aux deux conditions suivantes: (1) l'appareil ne doit pas produire de brouillage, et (2) l'utilisateur de l'appareil doit accepter tout brouillage radioélectrique subi, même si le brouillage est susceptible d'en compromettre le fonctionnement.

Concerning EVMs Including Detachable Antennas:

Under Industry Canada regulations, this radio transmitter may only operate using an antenna of a type and maximum (or lesser) gain approved for the transmitter by Industry Canada. To reduce potential radio interference to other users, the antenna type and its gain should be so chosen that the equivalent isotropically radiated power (e.i.r.p.) is not more than that necessary for successful communication. This radio transmitter has been approved by Industry Canada to operate with the antenna types listed in the user guide with the maximum permissible gain and required antenna impedance for each antenna type indicated. Antenna types not included in this list, having a gain greater than the maximum gain indicated for that type, are strictly prohibited for use with this device.

Concernant les EVMs avec antennes détachables

Conformément à la réglementation d'Industrie Canada, le présent émetteur radio peut fonctionner avec une antenne d'un type et d'un gain maximal (ou inférieur) approuvé pour l'émetteur par Industrie Canada. Dans le but de réduire les risques de brouillage radioélectrique à l'intention des autres utilisateurs, il faut choisir le type d'antenne et son gain de sorte que la puissance isotrope rayonnée équivalente (p.i.r.e.) ne dépasse pas l'intensité nécessaire à l'établissement d'une communication satisfaisante. Le présent émetteur radio a été approuvé par Industrie Canada pour fonctionner avec les types d'antenne énumérés dans le manuel d'usage et ayant un gain admissible maximal et l'impédance requise pour chaque type d'antenne. Les types d'antenne non inclus dans cette liste, ou dont le gain est supérieur au gain maximal indiqué, sont strictement interdits pour l'exploitation de l'émetteur

3.3 *Japan*

3.3.1 *Notice for EVMs delivered in Japan:* Please see http://www.tij.co.jp/lsds/ti_ja/general/eStore/notice_01.page 日本国内に輸入される評価用キット、ボードについては、次のところをご覧ください。

https://www.ti.com/ja-jp/legal/notice-for-evaluation-kits-delivered-in-japan.html

3.3.2 *Notice for Users of EVMs Considered “Radio Frequency Products” in Japan:* EVMs entering Japan may not be certified by TI as conforming to Technical Regulations of Radio Law of Japan.

If User uses EVMs in Japan, not certified to Technical Regulations of Radio Law of Japan, User is required to follow the instructions set forth by Radio Law of Japan, which includes, but is not limited to, the instructions below with respect to EVMs (which for the avoidance of doubt are stated strictly for convenience and should be verified by User):

1. Use EVMs in a shielded room or any other test facility as defined in the notification #173 issued by Ministry of Internal Affairs and Communications on March 28, 2006, based on Sub-section 1.1 of Article 6 of the Ministry's Rule for Enforcement of Radio Law of Japan,
2. Use EVMs only after User obtains the license of Test Radio Station as provided in Radio Law of Japan with respect to EVMs, or
3. Use of EVMs only after User obtains the Technical Regulations Conformity Certification as provided in Radio Law of Japan with respect to EVMs. Also, do not transfer EVMs, unless User gives the same notice above to the transferee. Please note that if User does not follow the instructions above, User will be subject to penalties of Radio Law of Japan.

【無線電波を送信する製品の開発キットをお使いになる際の注意事項】 開発キットの中には技術基準適合証明を受けていないものがあります。 技術適合証明を受けていないもののご使用に際しては、電波法遵守のため、以下のいずれかの措置を取っていただく必要がありますのでご注意ください。

1. 電波法施行規則第6条第1項第1号に基づく平成18年3月28日総務省告示第173号で定められた電波暗室等の試験設備でご使用いただく。
2. 実験局の免許を取得後ご使用いただく。
3. 技術基準適合証明を取得後ご使用いただく。

なお、本製品は、上記の「ご使用にあたっての注意」を譲渡先、移転先に通知しない限り、譲渡、移転できないものとします。

上記を遵守頂けない場合は、電波法の罰則が適用される可能性があることをご留意ください。 日本テキサス・イ
ンスツルメンツ株式会社

東京都新宿区西新宿６丁目２４番１号

西新宿三井ビル

3.3.3 *Notice for EVMs for Power Line Communication:* Please see http://www.tij.co.jp/lsds/ti_ja/general/eStore/notice_02.page 電力線搬送波通信についての開発キットをお使いになる際の注意事項については、次のところをご覧ください。https://www.ti.com/ja-jp/legal/notice-for-evaluation-kits-for-power-line-communication.html

3.4 *European Union*

3.4.1 *For EVMs subject to EU Directive 2014/30/EU (Electromagnetic Compatibility Directive)*:

This is a class A product intended for use in environments other than domestic environments that are connected to a low-voltage power-supply network that supplies buildings used for domestic purposes. In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures.

4 *EVM Use Restrictions and Warnings:*

4.1 EVMS ARE NOT FOR USE IN FUNCTIONAL SAFETY AND/OR SAFETY CRITICAL EVALUATIONS, INCLUDING BUT NOT LIMITED TO EVALUATIONS OF LIFE SUPPORT APPLICATIONS.

4.2 User must read and apply the user guide and other available documentation provided by TI regarding the EVM prior to handling or using the EVM, including without limitation any warning or restriction notices. The notices contain important safety information related to, for example, temperatures and voltages.

4.3 *Safety-Related Warnings and Restrictions:*

4.3.1 User shall operate the EVM within TI's recommended specifications and environmental considerations stated in the user guide, other available documentation provided by TI, and any other applicable requirements and employ reasonable and customary safeguards. Exceeding the specified performance ratings and specifications (including but not limited to input and output voltage, current, power, and environmental ranges) for the EVM may cause personal injury or death, or property damage. If there are questions concerning performance ratings and specifications, User should contact a TI field representative prior to connecting interface electronics including input power and intended loads. Any loads applied outside of the specified output range may also result in unintended and/or inaccurate operation and/or possible permanent damage to the EVM and/or interface electronics. Please consult the EVM user guide prior to connecting any load to the EVM output. If there is uncertainty as to the load specification, please contact a TI field representative. During normal operation, even with the inputs and outputs kept within the specified allowable ranges, some circuit components may have elevated case temperatures. These components include but are not limited to linear regulators, switching transistors, pass transistors, current sense resistors, and heat sinks, which can be identified using the information in the associated documentation. When working with the EVM, please be aware that the EVM may become very warm.

4.3.2 EVMs are intended solely for use by technically qualified, professional electronics experts who are familiar with the dangers and application risks associated with handling electrical mechanical components, systems, and subsystems. User assumes all responsibility and liability for proper and safe handling and use of the EVM by User or its employees, affiliates, contractors or designees. User assumes all responsibility and liability to ensure that any interfaces (electronic and/or mechanical) between the EVM and any human body are designed with suitable isolation and means to safely limit accessible leakage currents to minimize the risk of electrical shock hazard. User assumes all responsibility and liability for any improper or unsafe handling or use of the EVM by User or its employees, affiliates, contractors or designees.

4.4 User assumes all responsibility and liability to determine whether the EVM is subject to any applicable international, federal, state, or local laws and regulations related to User's handling and use of the EVM and, if applicable, User assumes all responsibility and liability for compliance in all respects with such laws and regulations. User assumes all responsibility and liability for proper disposal and recycling of the EVM consistent with all applicable international, federal, state, and local requirements.

5. *Accuracy of Information:* To the extent TI provides information on the availability and function of EVMs, TI attempts to be as accurate as possible. However, TI does not warrant the accuracy of EVM descriptions, EVM availability or other information on its websites as accurate, complete, reliable, current, or error-free.

6. *Disclaimers:*

6.1 EXCEPT AS SET FORTH ABOVE, EVMS AND ANY MATERIALS PROVIDED WITH THE EVM (INCLUDING, BUT NOT LIMITED TO, REFERENCE DESIGNS AND THE DESIGN OF THE EVM ITSELF) ARE PROVIDED "AS IS" AND "WITH ALL FAULTS." TI DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING SUCH ITEMS, INCLUDING BUT NOT LIMITED TO ANY EPIDEMIC FAILURE WARRANTY OR IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADE SECRETS OR OTHER INTELLECTUAL PROPERTY RIGHTS.

6.2 EXCEPT FOR THE LIMITED RIGHT TO USE THE EVM SET FORTH HEREIN, NOTHING IN THESE TERMS SHALL BE CONSTRUED AS GRANTING OR CONFERRING ANY RIGHTS BY LICENSE, PATENT, OR ANY OTHER INDUSTRIAL OR INTELLECTUAL PROPERTY RIGHT OF TI, ITS SUPPLIERS/LICENSORS OR ANY OTHER THIRD PARTY, TO USE THE EVM IN ANY FINISHED END-USER OR READY-TO-USE FINAL PRODUCT, OR FOR ANY INVENTION, DISCOVERY OR IMPROVEMENT, REGARDLESS OF WHEN MADE, CONCEIVED OR ACQUIRED.

7. *USER'S INDEMNITY OBLIGATIONS AND REPRESENTATIONS.* USER WILL DEFEND, INDEMNIFY AND HOLD TI, ITS LICENSORS AND THEIR REPRESENTATIVES HARMLESS FROM AND AGAINST ANY AND ALL CLAIMS, DAMAGES, LOSSES, EXPENSES, COSTS AND LIABILITIES (COLLECTIVELY, "CLAIMS") ARISING OUT OF OR IN CONNECTION WITH ANY HANDLING OR USE OF THE EVM THAT IS NOT IN ACCORDANCE WITH THESE TERMS. THIS OBLIGATION SHALL APPLY WHETHER CLAIMS ARISE UNDER STATUTE, REGULATION, OR THE LAW OF TORT, CONTRACT OR ANY OTHER LEGAL THEORY, AND EVEN IF THE EVM FAILS TO PERFORM AS DESCRIBED OR EXPECTED.

8. *Limitations on Damages and Liability:*

   8.1 *General Limitations*. IN NO EVENT SHALL TI BE LIABLE FOR ANY SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL, OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF THESE TERMS OR THE USE OF THE EVMS , REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. EXCLUDED DAMAGES INCLUDE, BUT ARE NOT LIMITED TO, COST OF REMOVAL OR REINSTALLATION, ANCILLARY COSTS TO THE PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, RETESTING, OUTSIDE COMPUTER TIME, LABOR COSTS, LOSS OF GOODWILL, LOSS OF PROFITS, LOSS OF SAVINGS, LOSS OF USE, LOSS OF DATA, OR BUSINESS INTERRUPTION. NO CLAIM, SUIT OR ACTION SHALL BE BROUGHT AGAINST TI MORE THAN TWELVE (12) MONTHS AFTER THE EVENT THAT GAVE RISE TO THE CAUSE OF ACTION HAS OCCURRED.

   8.2 *Specific Limitations.* IN NO EVENT SHALL TI'S AGGREGATE LIABILITY FROM ANY USE OF AN EVM PROVIDED HEREUNDER, INCLUDING FROM ANY WARRANTY, INDEMITY OR OTHER OBLIGATION ARISING OUT OF OR IN CONNECTION WITH THESE TERMS, , EXCEED THE TOTAL AMOUNT PAID TO TI BY USER FOR THE PARTICULAR EVM(S) AT ISSUE DURING THE PRIOR TWELVE (12) MONTHS WITH RESPECT TO WHICH LOSSES OR DAMAGES ARE CLAIMED. THE EXISTENCE OF MORE THAN ONE CLAIM SHALL NOT ENLARGE OR EXTEND THIS LIMIT.

9. *Return Policy.* Except as otherwise provided, TI does not offer any refunds, returns, or exchanges. Furthermore, no return of EVM(s) will be accepted if the package has been opened and no return of the EVM(s) will be accepted if they are damaged or otherwise not in a resalable condition. If User feels it has been incorrectly charged for the EVM(s) it ordered or that delivery violates the applicable order, User should contact TI. All refunds will be made in full within thirty (30) working days from the return of the components(s), excluding any postage or packaging costs.

10. *Governing Law:* These terms and conditions shall be governed by and interpreted in accordance with the laws of the State of Texas, without reference to conflict-of-laws principles. User agrees that non-exclusive jurisdiction for any dispute arising out of or relating to these terms and conditions lies within courts located in the State of Texas and consents to venue in Dallas County, Texas. Notwithstanding the foregoing, any judgment may be enforced in any United States or foreign court, and TI may seek injunctive relief in any United States or foreign court.