Digital Video Broadcasting - Asynchronous Serial Interface (DVB-ASI) IP Core User's Guide

December 2010
IPUG90_01.1

Table of Contents

Digital Video Broadcasting (DVB) is a suite of open standards for digital television. The interface schemes for DVB are defined in the European standard EN 50083-9, “Interfaces for CATV/SMATV Headends and Similar Professional Equipment”. DVB via Asynchronous Serial Interface (DVB-ASI) is one of the primary mechanisms to transport MPEG-2 based streams over cable media. DVB-ASI is a three-layered architecture with the top layer (Layer 2) specified by IEC 13818-1 and the bottom layers, Layer 1 and Layer 0, specified by the Fibre Channel standard IEC 141165-1, part 1. This IP core along with LatticeECP3™ SERDES/PCS implements Layer 1 of the DVB-ASI interface. A part of Layer 2 functionality is also supported by the IP core.

The transmit (Tx) path in the Layer 1 protocol includes rate matching FIFO, comma insertion, 8b10b encoding and serialization. The receive (Rx) path includes clock data recovery, deserialization, 8b10b decoding, comma removal and rate matching FIFO. The LatticeECP3 SERDES/PCS block is used for the 8b10b encoding/decoding, serialization, clock data recovery and deserialization. The rest of the Layer 1 functions are implemented by the IP core. The Layer 2 functions in the IP include packet size/sync byte determination, runt/giant packet detection and receiver lock status indication. The DVB-ASI IP core is designed to be easily and seamlessly integrated with the LatticeECP3 SERDES.

Refer to the Lattice DVB-ASI web page at www.latticesemi.com/products/intellectualproperty/ipcores/dvbasi.cfm for links to the documents related to the DVB-ASI as well as related video interfaces.

Note: Throughout this document, “comma” is used to denote the Fibre Channel comma character (K28.5) or physical “sync bytes” that are used as padding bytes for rate matching and as preceding bytes for start of MPEG-2 packets. The term sync byte is used to denote the start of packet byte (47h or B8h) of the higher level MPEG-2 transport stream packets.

Quick Facts

Table 1-1 gives quick facts about the DVB-ASI IP core.

Table 1-1. DVB-ASI IP Core Quick Facts

		DVB-ASI IP Core		
		Tx and Rx	**Tx Only**	**Rx Only**
Core Requirements	FPGA Family Supported	LatticeECP3		
	Minimum Device Required	LFE3-17E-6FN484CES		
Typical Resource Utilization	Targeted Device	LFE3-95E-7FN1156CES		
	Data Path Width	8	8	8
	LUTs	480	170	310
	sysMEM™ EBRs	2	1	1
	Registers	350	140	210
Design Tool Support[1]	Lattice Implementation	Lattice Diamond™ 1.1 or ispLEVER® 8.1SP1		
	Synthesis	Synopsys® Synplify™ Pro for Lattice D-2010.03L-SP1		
		Mentor Graphics® Precision™ RTL 2010a.218		
	Simulation	Aldec® Active-HDL™ 8.2 Lattice Edition		
		Mentor Graphics ModelSim™ SE 6.3F		

1. Design tool support for IP core version 1.1.

Features

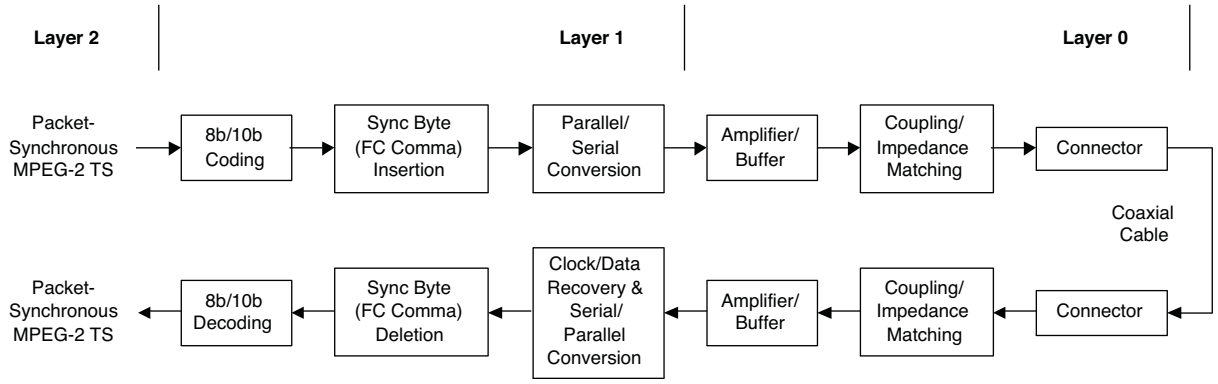
The following are the major features of the Lattice DVB-ASI IP core:

- Configurable Tx FIFO supporting independent Layer 2 transmit clock
- Configurable Rx FIFO supporting independent Layer 2 receive clock
- Automatic start of packet comma insertion in the transmit side
- Rate-matching comma insertion in the transmit side
- Handshake signals for sync and data valid for transmitter and receiver
- Tx and Rx FIFO full indicators
- Rx sync byte indicator for identifying start-of-packet
- Optional configurable almost full and almost empty indicators for the FIFOs
- Matching port names for seamless connection with LatticeECP3 SERDES/PCS
- Layer 2 functions including lock, sync byte, packet size, runt and giant indicators

Overview

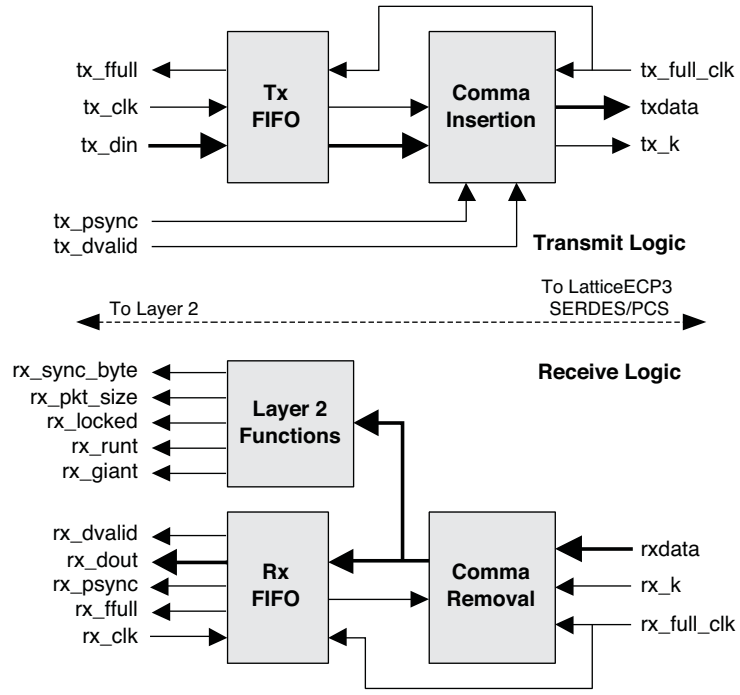
DVB-ASI is based on a layered architecture, with each layer doing a specific electrical or logic function. The layered architecture for DVB-ASI over coaxial cable is shown in Figure 2-1. Layer 0 is the physical layer that includes the amplifier, impedance matching and the connectors that ensure electrical compliance to the interface. Layer 1 is the coding layer that includes the 8b10b encoding/decoding, comma character insertion/deletion, clock/data recovery and parallel-to-serial/serial-to-parallel conversions. Layer 2 deals with higher level functionalities such as creating transport stream packets and Reed-Solomon encoding/decoding.

Figure 2-1. DVB-ASI Interface Layers



The IP core along with LatticeECP3 SERDES implements Layer 1 interface functions. The top-level functional block diagram of the DVB-ASI IP core is shown in Figure 2-2. The upper half of the figure shows the transmitter and the lower part, the receiver. A detailed description of the transmitter and the receiver are given below.

Figure 2-2. Top Level View of the DVB-ASI IP Core



Transmitter

As shown in Figure 2-2, the transmitter consists of a transmit (Tx) FIFO and comma insertion logic. The input data to the transmitter, tx_din is the MPEG-2 payload data from Layer 2. This data contains the packet synchronization (sync) byte (47h or B8h) at the start of each packet. As the transmitter does not process the data, the start of packet must be indicated by the one clock cycle wide tx_psync input. The transmitter inserts two comma words just before the start of packet. The IP transmitter does not check for giant or runt packets. It is important that the tx_psync is properly flagged to the IP to avoid packet loss.

The Tx FIFO is an asynchronous FIFO that is written using the user's tx_clk and read using the SERDES' tx_full_clk. The SERDES transmit clock, tx_full_clk is required to be maintained at 27 MHz plus or minus 100 ppm (parts per million) to comply with the DVB-ASI standard. The user clock for transmit, tx_clk must be maintained at or less than (188/190)*27 MHz or (204/206)*27 MHz (depending on whether the packet size is 188 or 204), for continuous operation with a shallow FIFO depth. This reduced rate is required to allow for the insertion of the mandatory twin sync words at the beginning of each packet by the IP. By providing a configurable depth FIFO and an input valid control signal (tx_dvalid), the IP allows the use of a wide range of clock rates and non-contiguous data inputs. For example, the input clock can be equal to 27 MHz, but the tx_dvalid input can be made zero for two clock cycles per packet. If the input clock is very low, the IP inserts comma characters to keep the FIFO from becoming empty. On the other hand, if the input clock rate is higher than 27 MHz, tx_dvalid can be de-asserted for sufficient number of cycles to ensure the FIFO is not getting filled. If the data rate is fluctuating, but averages out to 27 Mbps in the long run, a deeper FIFO will be required, the depth depending on the kind of fluctuation. If the rate of valid input data is continuously high for an extended period of time, the FIFO becomes full and asserts Tx FIFO full (tx_ffull). If this happens, the input data is lost and may lead to loss of one or more packets.

Receiver

As shown in Figure 2-2, the receiver has a comma removal block, a FIFO and a Layer 2 functions block. The comma removal block deletes comma characters from the received parallel data stream.

Rx FIFO is a configurable asynchronous FIFO which is written with rx_full_clk and read using rx_clk. The rate of rx_clk must be lower than the received data rate, which is equal to the rate of received data excluding all comma characters. If rx_clk is lower than the data rate, the Rx FIFO gets filled and rx_ffull is asserted, leading to erroneous output data. If rx_clk is higher than the data rate, the IP receiver asserts rx_dvalid low for appropriate duration to offset the high rate clock.

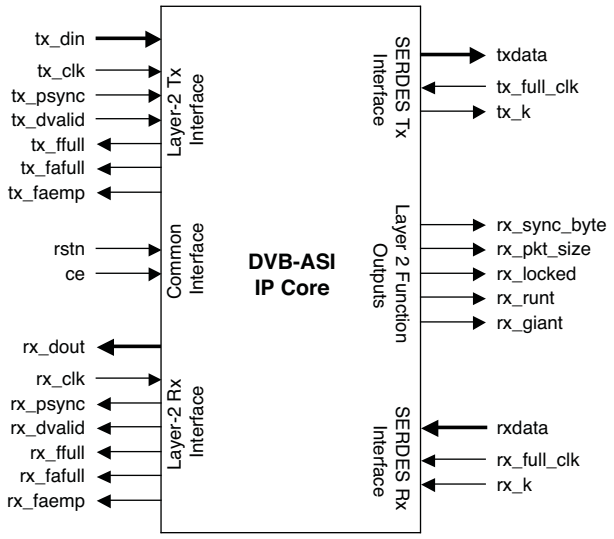
The receiver provides limited Layer 2 functionality which includes receiver lock detection, synchronization (sync) byte determination, packet size determination, runt packet detection and giant packet detection. The receiver determines the sync byte and packet size by looking for continuous periodic occurrences of a valid sync byte over a valid packet size. To do this, the receiver sequentially sets the sync byte and packet size values to one of their valid values and checks to see if it can lock. The receiver locks when the number of matches of continuous valid sync bytes is more than the value set by lock match threshold. It unlocks when the number of errors (sync not received at the expected instances) exceeds unlock error threshold. If the receiver is not able to lock within a certain number of packets, defined by sync hunt threshold, it goes on to look for sync matches for a different combination of packet size and sync byte values.

Once the receiver is locked and the values of sync byte and packet size have been determined, the receiver checks for runt and giant packets. If the number of valid words between two consecutive start-of-packet sync bytes is less than the determined packet size (either 188 or 204), it is identified as a runt packet. If the number of valid words between two consecutive start-of-packet sync bytes is more than the determined packet size (188 or 204), it is identified as a giant packet.

Interface Description

The top-level interface diagram for the DVB-ASI IP is shown in Figure 2-3.

Figure 2-3. DVB-ASI IP Interfaces



Top-Level I/O Interface

Table 2-1 provides the list of ports and descriptions of each port for the DVB-ASI IP core.

Table 2-1. DVB-ASI IP Core Port Descriptions

Port	Bits	I/O	Description
Layer 2 Tx Interface			
tx_din	8	I	Parallel input data from Layer 2. This data is read using tx_clk.
tx_clk	1	I	Clock input for the parallel input data.
tx_psync	1	I	Transmit packet sync word identifier. This one clock cycle pulse indicates the start of a MPEG-2 packet. The IP will insert two consecutive comma characters before transmitting the packet.
tx_dvalid	1	I	Transmit data valid. The data at tx_din is considered valid only if this input is high. The data at tx_din is ignored if tx_dvalid is low.
tx_ffull	1	O	Transmit FIFO full output. If this output is high, the IP can not read any further data from the input.
tx_fafull	1	O	Transmit FIFO almost full output.
tx_faemp	1	O	Transmit FIFO almost empty output.
Common Interface			
rstn	1	I	System-wide asynchronous active-low reset signal. This signal resets all the registers in the IP.
ce	1	I	Optional clock enable signal. An active high input at ce freezes all the switching operations in the IP. It is useful for putting the IP in a power save mode.
Layer 2 Rx Interface			
rx_dout	8	O	Parallel output data to Layer 2. This output is synchronous with rx_clk.
rx_clk	1	I	This is the receive clock with which Layer 2 reads out the received parallel data.
rx_psync	1	O	Received packet sync identifier. This one clock cycle pulse indicates the start of a MPEG-2 packet, as determined by the occurrence of a relevant 47h or B8h character in the received data.
rx_dvalid	1	O	Rx data valid output. If rx_clk is too fast, the Rx FIFO becomes empty and there is no data available for reading. Under these conditions, rx_dvalid goes low.
rx_ffull	1	O	Rx FIFO full output. If rx_clk is too slow, Layer 2 will not able to read all the received data from the Rx FIFO. Under these conditions, rx_ffull is asserted to indicate that Layer 2 is losing some of the streamed-in data.
rx_fafull	1	O	Rx FIFO almost full output.

Table 2-1. DVB-ASI IP Core Port Descriptions (Continued)

Port	Bits	I/O	Description
rx_faemp	1	O	Rx FIFO almost empty output.
SERDES Tx Interface			
txdata	8	O	Parallel data that is to be serialized by the SERDES. This is connected to txdata_chx (x is the channel number in the quad) input of the SERDES block.
tx_full_clk	1	I	This is the clock that is used to read the data out of the IP, txdata. Since the data rate is the same as the transmit data line rate, the clock should be the same as the line rate transmit clock. This clock can be driven by the Tx PLL output clock, tx_full_clk, from the SERDES.
tx_k	1	O	Transmit comma character identifier. This output identifies that a comma character is placed at the parallel output, txdata. This output must be connected to tx_k_chx input of the SERDES/PCS block to enable appropriate encoding of the comma data.
Layer 2 Function Outputs			
rx_sync_byte	1	O	Rx sync byte identifier. The sync byte is 47h, if rx_sync_byte is 0 and B8h, if 1.
rx_pkt_size	1	O	Indicates the received packet size. The received packet size is 188 if rx_pkt_size is 0 and 204 if 1.
rx_locked	1	O	Indicates the receiver is locked to a DVB-ASI input. The lock implies that valid sync bytes are being received at appropriate packet intervals.
rx_runt	1	O	Runt packet indicator. This indicates that a runt packet has been received. This condition is ascertained if the packet size as counted between two consecutive start-of-packet identifiers, is less than the standard packet size (188 or 204).
rx_giant	1	O	Giant packet indicator. This indicates that a giant packet has been received. This condition is ascertained if the packet size as counted between two consecutive start-of-packet identifiers, is more than the standard packet size (188 or 204).
SERDES Rx Interface			
rxdata	8	I	Received parallel data from the SERDES/PCS. This data is read using rx_full_clk.
rx_full_clk	1	I	This is the clock that is used to read-in rxdata. The nominal frequency of this clock for DVB-ASI is 27 MHz. This input can be driven by the Rx recovered clock from the SERDES.
rx_k	1	I	Received comma character identifier. This input identifies if the parallel input at rxdata is a comma character. This input is connected to the SERDES output rx_k_chx, where x is the SERDES channel that is used.

Interface Description

This section describes the Layer 2 and SERDES interfaces of the IP core.

Layer 2 Tx Interface

The DVB-ASI IP core is a Layer 1 implementation and hence does not process the data values. The IP core transmits the data that comes in through tx_din as is except for inserting comma characters (K28.5) to match the input data rate to the line rate of 270 Mbps. The IP also inserts two consecutive comma characters (K28.5) right before the sync byte. The occurrence of the sync byte at the input needs to be informed to the IP by asserting the tx_psync input. The IP does not verify the sync byte by reading the data at tx_din nor does it monitor the frequency of tx_psync. The input signal, tx_dvalid informs the IP if the data on tx_din is valid and transmitted or to be ignored. It must be noted that tx_dvalid is used strictly as a data valid identifier as opposed to the signal dvalid defined in DVB-SPI which is used to identify normal bytes from null bytes.

When the Tx FIFO full status output, tx_ffull is high, the IP will ignore the input data at tx_din, irrespective of the value of tx_dvalid. It is important that this output is monitored and the data input is suspended when it is high to avoid packet loss. The flags tx_fafull and tx_faemp are early warning flags for Tx FIFO full and empty respectively. The user may be able to set the thresholds for almost empty and almost full flags from the IP GUI before generating the IP.

Layer 2 Rx Interface

The received parallel data appears at rx_dout port. The received data is valid only when the qualifier output, rx_dvalid is high. Similar to the transmit data valid signal, the rx_dvalid is just a data valid qualifier and bears a different meaning compared to the dvalid signal defined in the DVB-SPI standard. The duration and frequency of rx_dvalid going low depends on the received data rate and the user receive clock, rx_clk. The output at rx_dout during the times rx_dvalid is high contains only the data and no comma characters (K28.5).

The Rx FIFO full output indicates that the there is more data received from the DVB-ASI link than that is read out at rx_dout. The receive clock (rx_clk) rate has to be increased appropriately to avoid losing data. The optional outputs, rx_fafull and rx_faemp give early indication of Rx FIFO almost full and almost empty conditions.

SERDES Tx Interface

The SERDES Tx interface signals have been designed to make the connection to LatticeECP3 SERDES simple and straightforward. The outputs txdata and tx_k can be connected to the SERDES inputs txdata_chX and tx_k_chX respectively, where X is the channel number. The transmit full clock output from the SERDES, tx_full_clk_chX can be directly connected to the tx_full_clk input of the DVB-ASI IP.

Layer 2 Function Outputs

The Layer 2 function outputs, rx_sync_byte, rx_pkt_size, rx_locked, rx_runt and rx_giant are available in this category. If there is a runt or a giant packet, the output rx_runt or rx_giant goes high for the duration of the packet.

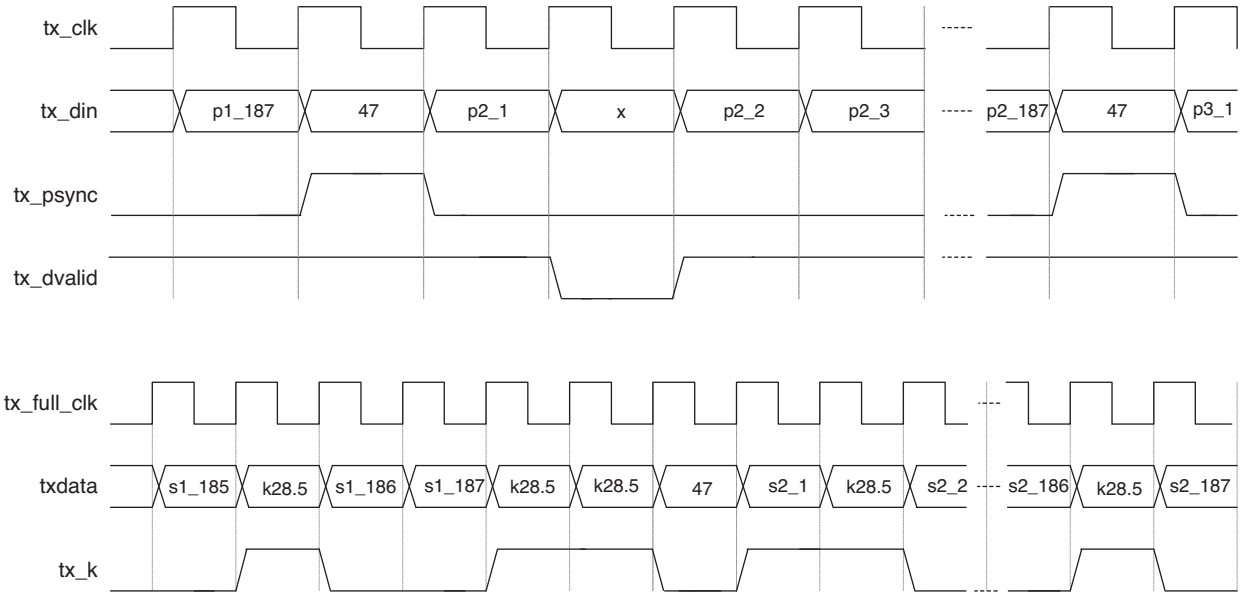
SERDES Rx Interface

The inputs rxdata and rx_k can be driven directly by the SERDES outputs rxdata_chX and rx_k_chX respectively, where X is the channel number. The recovered clock output from the SERDES, rx_full_clk_chX can be directly connected to the rx_full_clk input of the DVB-ASI IP.

Interface Waveforms

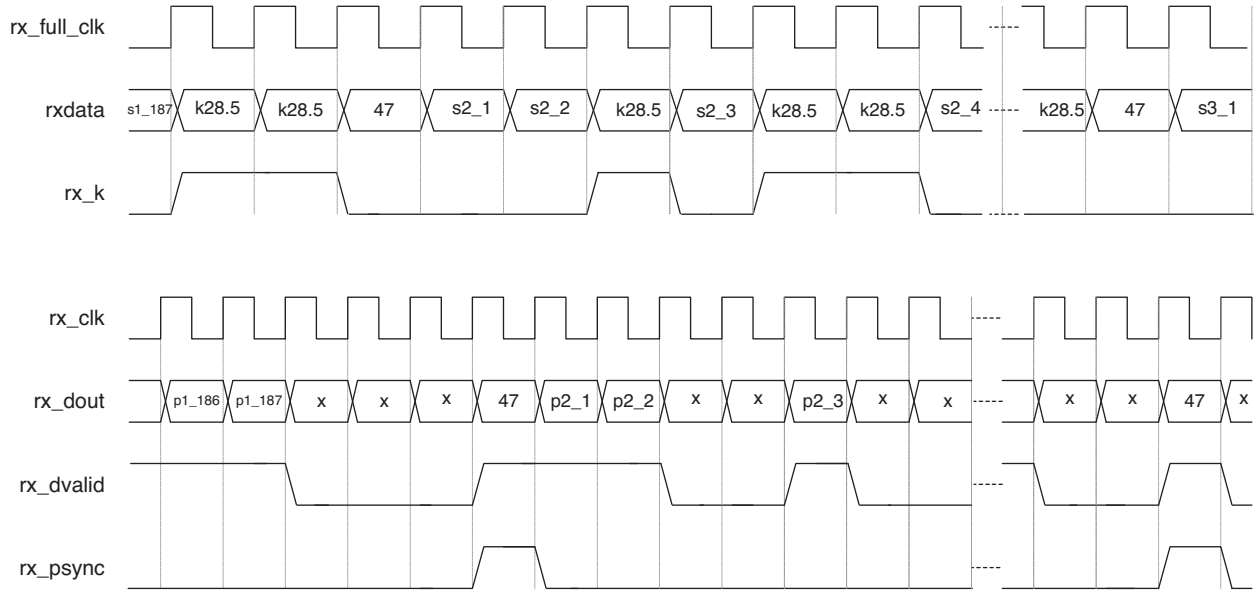
The timing diagrams for the transmit and receive interfaces are provided in Figure 2-4 and Figure 2-5.

Figure 2-4. Timing Diagram for the Transmit Interface



In Figure 2-4, p<x>_<y> denotes the input data to the IP core corresponding to packet x and data number y. Similarly, s<x>_<y> denotes the parallel data out of the transmitter to the SERDES corresponding to packet x and data number y. As shown in Figure 2-4, a tx_psync pulse causes two consecutive comma characters (K28.5) to be added to the output data stream just before the sync byte. The comma characters preceding the sync byte as well as those inserted to match the rate are flagged to the SERDES by asserting tx_k high during those periods.

Figure 2-5. Timing Diagram for the Receive Interface



As shown in Figure 2-5, the data coming out of the SERDES contain several comma characters, each identified by the signal rx_k going high. The IP core strips the data of all the comma characters and provides only the packet data at the output rx_dout. If the rx_clk rate is higher than the packet data rate, periods of data not being valid occur that are identified by rx_dvalid going low.

Parameter Settings

The IPexpress™ tool is used to create IP and architectural modules in the Diamond or ispLEVER software. Refer to "IP Core Generation" on page 16 for a description on how to generate the IP.

Table 3-1 describes the user-configurable parameters for the DVB-ASI IP core. The parameter settings are specified using the DVB-ASI IP core Configuration GUI in IPexpress. The DVB-ASI parameter options are partitioned across two GUI tabs as shown in this chapter.

Table 3-1. IP Core Parameters

Name	Range	Default
PHY Function		
PHY function	{Tx, Rx, Both}	Both
Tx FIFO Options		
Tx FIFO depth	{32,64,128,256,512,1024,2048,4096}	256
Enable almost full/empty flags for Tx FIFO	{Yes, No}	Yes
Almost full threshold for Tx FIFO	1 to Tx FIFO depth	242
Almost empty threshold for Tx FIFO	1 to Tx FIFO depth	16
Rx FIFO Options		
Rx FIFO depth	{32,64,128,256,512,1024,2048,4096}	256
Enable almost full/empty flags for Rx FIFO	{Yes, No}	Yes
Almost full threshold for Rx FIFO	1 to Rx FIFO depth	242
Almost empty threshold for Rx FIFO	1 to Rx FIFO depth	16
Optional Ports		
Clock enable port	{Yes, No}	No
Layer 2 Rx Settings		
Rx Sync Identification Scheme	{sync word, 2 successive, 2-in-5}	2 successive
Sync hunt threshold	{2 to 254 }	12
Lock match threshold	{2 to 254 }	3
Unlock error threshold	{2 to 254 }	3

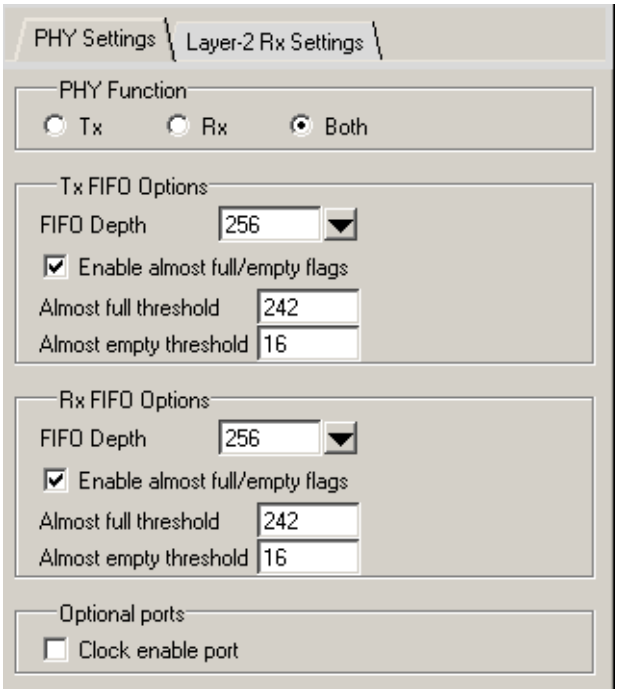
The various IP core options for each tab are discussed in further detail in the following sections.

PHY Settings Tab

The PHY function option selects whether transmit, receive or both transmit and receive functions are required for the IP core. Since transmit and receive functionalities are independent of each other, it is possible to select only the part that is required. The Tx and Rx FIFO options available are FIFO depth, whether almost full/empty flags are brought out and if yes, the threshold values for the almost full and almost empty flags.

Figure 3-1 shows the controls in the PHY settings tab.

Figure 3-1. DVB-ASI IP Core PHY Settings Tab



PHY Function

This parameter selects whether transmit, receive or both transmit and receive functions are required for the IP core. Since transmit and receive functionalities are independent of each other, it is possible to select only the part that is required.

Tx FIFO Options

FIFO Depth
This parameter sets Tx FIFO depth.

Enable Almost Full/Empty Flags
This parameter determines if almost full/empty flags are enabled for Tx FIFO. If yes, almost full and almost empty output flags are provided for the Tx FIFO. The following two additional parameters for the thresholds are also made available.

Almost Full Threshold
This parameter sets Almost full threshold for Tx FIFO.

Almost Empty Threshold
This parameter sets Almost empty threshold for Tx FIFO.

Rx FIFO Options

FIFO Depth
This parameter sets Rx FIFO depth.

Enable Almost Full/Empty Flags
This parameter determines if almost full/empty flags are enabled for Rx FIFO. If yes, almost Full and almost empty output flags are provided for the Rx FIFO. The following two additional parameters for the threshold are also made available.

Almost Full Threshold
This parameter sets Almost full threshold for Rx FIFO.

Almost Empty Threshold
This parameter sets Almost empty threshold for Rx FIFO.

The Tx and Rx FIFO options available are FIFO depth, whether almost full/empty flags are brought out and if yes, the threshold values for the almost full and almost empty flags.

Optional Ports
Determines if a clock enable port is required in the IP.

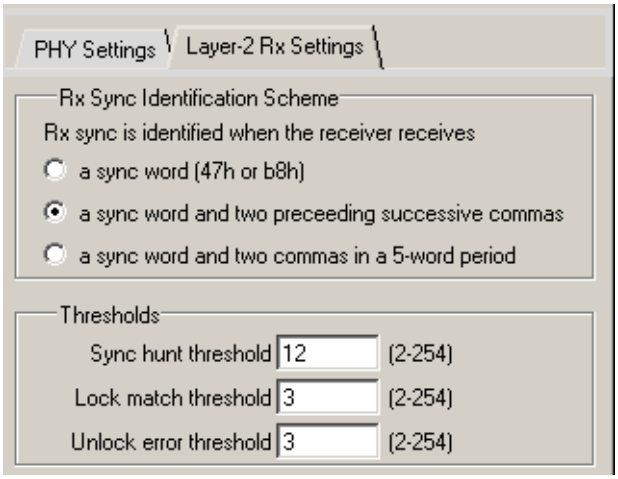
Layer 2 Rx Settings

The Layer 2 Rx Settings tab offers some advanced settings for the receive Layer 2 functions. The parameter Rx Sync Identification Scheme determines how a receive start of packet is identified. In most applications, the Rx start of packet is identified by the occurrence of the sync byte (47_h or $B8_h$) preceded by two consecutive comma characters (K28.5). There are some systems that mark the start of packet with a sync byte (47_h or $B8_h$) which is preceded by two comma characters (K28.5) in the 5-word period preceding the sync byte.

The receiver goes to lock state (asserts rx_locked) when the packet sync bytes are correctly received for a certain period of time. The Lock match threshold parameter sets the time in number of packets after which the receiver goes to lock state. Similarly, when the packets are not received regularly or received with errors, the receiver transitions into unlock state. The number of sync byte errors that is required to take the receiver from locked to unlocked state is set by the Unlock error threshold parameter.

Figure 3-2 shows the Layer 2 Rx settings tab.

Figure 3-2. DVB-ASI IP Core Layer 2 Rx Settings Tab



Rx Sync Identification Scheme

This parameter determines how Rx sync is identified. The sync is identified as follows for different options:

- **Sync word**: Whenever a sync word (47_h or $B8_h$) is received
- **2 successive**: When a sync word and two preceding successive commas are received.
- **2-in-5**: When a sync word and two preceding commas are received in a 5-word period.

Thresholds

Sync Hunt Threshod
This parameter determines how many packets to wait while trying to lock before proceeding to the next sync byte and packet size combination.

Look Match Threshold
This parameter determines the number of sync matches required to transition to the receive lock state from the unlock state.

Unlock Error Threshold
This parameter determines the number of sync errors tolerated before going to the receive unlock state from the lock state.

IP Core Generation

This chapter provides information on licensing the DVB-ASI IP core, generating the core using the Diamond or ispLEVER software IPexpress tool, running functional simulation, and including the core in a top-level design.

The Lattice DVB-ASI IP core can be used only in LatticeECP3 device families.

Licensing the IP Core

An IP license is required to enable full, unrestricted use of the DVB-ASI IP core in a complete, top-level design. The license specifies the IP core (DVB-ASI) and device family (LatticeECP3).

Instructions on how to obtain licenses for Lattice IP cores are availabe on the Lattice web site:

www.latticesemi.com/products/intellectualproperty/aboutip/isplevercoreonlinepurchas.cfm

Users may download and generate the DVB-ASI IP core for LatticeECP3 and fully evaluate the core through functional simulation and implementation (synthesis, map, place and route) without an IP license. The DVB-ASI IP core for LatticeECP3 also supports Lattice's IP hardware evaluation capability, which makes it possible to create versions of the IP core that operate in hardware for a limited time (approximately four hours) without requiring an IP license (see "Hardware Evaluation" on page 22 for further details). However, a license is required to enable timing simulation, to open the design in the Diamond or ispLEVER EPIC tool, and to generate bitstreams that do not include the hardware evaluation timeout limitation.

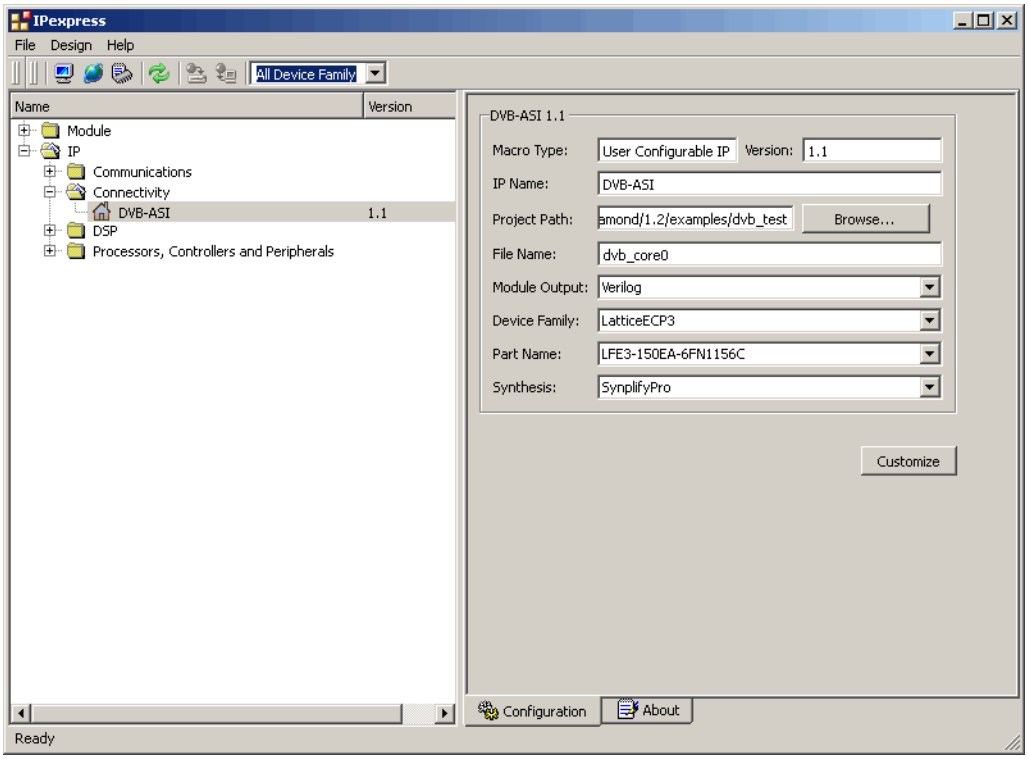
Getting Started

The DVB-ASI IP core is available for download from the Lattice IP Server using the IPexpress tool. The IP files are automatically installed using ispUPDATE technology in any customer-specified directory. After the IP core has been installed, the IP core will be available in the IPexpress GUI dialog box shown in Figure 4-1

The ispLEVER IPexpress tool GUI dialog box for the DVB-ASI IP core is shown in Figure 4-1. To generate a specific IP core configuration the user specifies:

- **Project Path** – Path to the directory where the generated IP files will be located.
- **File Name** – "username" designation given to the generated IP core and corresponding folders and files.
- **(Diamond) Module Output** – Verilog or VHDL.
- **(ispLEVER) Design Entry Type** – Verilog HDL or VHDL.
- **Device Family** – Device family to which IP is to be targeted (e.g. LatticeSCM, Lattice ECP2M, LatticeECP3, etc.). Only families that support the particular IP core are listed.
- **Part Name** – Specific targeted part within the selected device family.

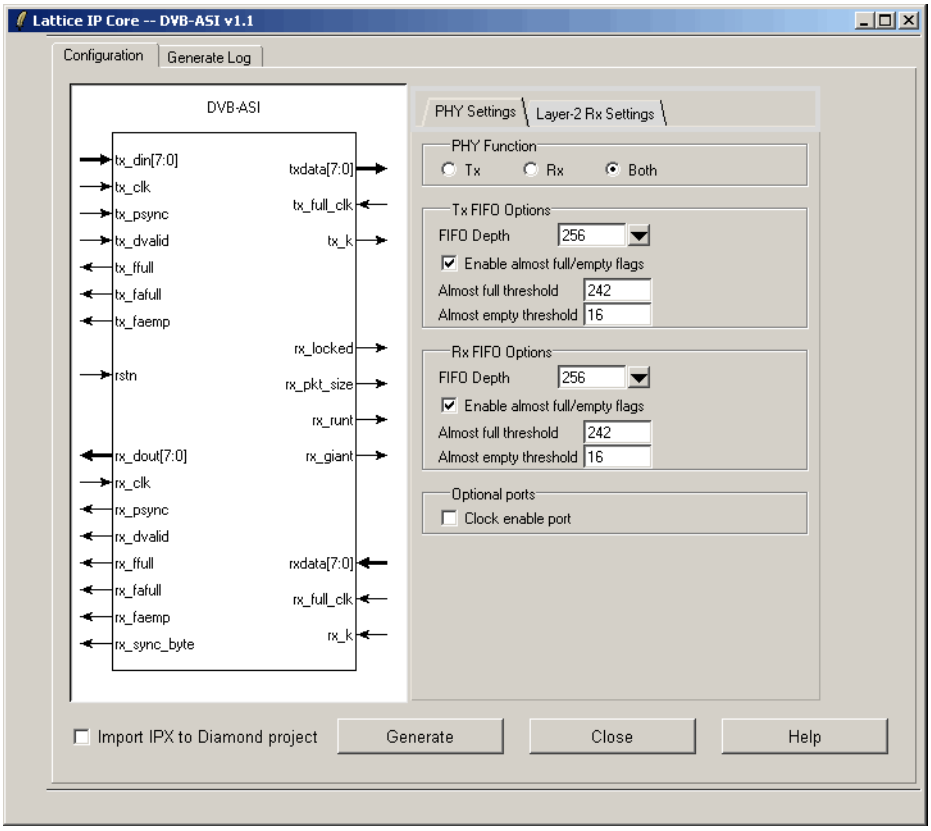
Figure 4-1. IPexpress Dialog Box (Diamond Version)



Note that if the IPexpress tool is called from within an existing project, Project Path, Module Output (Design Entry in ispLEVER), Device Family and Part Name default to the specified project parameters. Refer to the IPexpress tool online help for further information.

To create an IP configuration, the user clicks the **Customize** button in the IPexpress tool dialog box to display the DVB-ASI IP core Configuration GUI, as shown in Figure 4-2. From this dialog box, the user can select the IP parameter options specific to their application. Refer to "Parameter Settings" on page 12 for more information on the DVB-ASI parameter settings. Additional information and known issues about the DVB-ASI IP core are provided in a ReadMe document that may be opened by clicking on the **Help** button in the Configuration GUI.

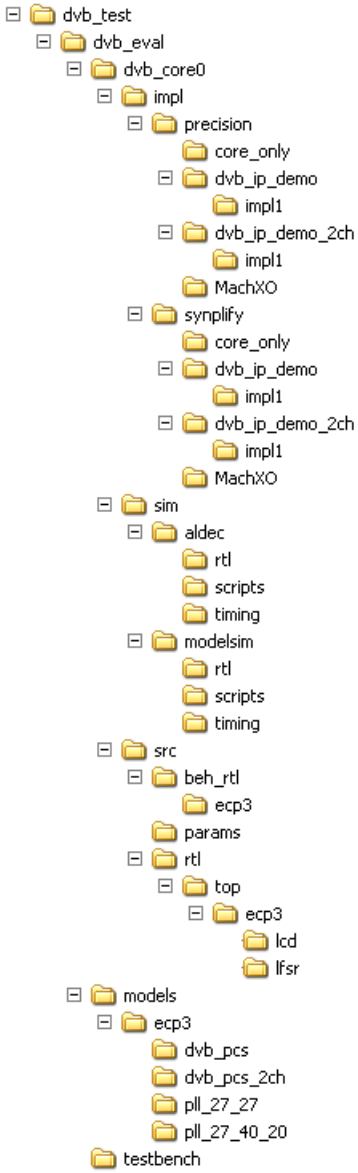
Figure 4-2. Configuration GUI (Diamond Version)



IPexpress-Created Files and Top-Level Directory Structure

When the user clicks the **Generate** button in the IP Configuration dialog box, the IP core and supporting files are generated in the specified Project Path directory. The directory structure of the generated files is shown in Figure 4-3.

Figure 4-3. Lattice DVB-ASI IP Core Directory Structure



The design flow for IP created with the IPexpress tool uses a post-synthesized module (NGO) for synthesis and a protected model for simulation. The post-synthesized module is customized and created by the IPexpress tool during IP generation. The protected simulation model is not customized by the IPexpress tool during the generation process, and relies on parameters provided to customize behavior during simulation.

Table 4-1 provides a list of key files and directories created by the IPexpress tool and how they are used. The IPexpress tool creates several files that are used throughout the design cycle. The names of most of the created files are customized to the user's module name (<*username*>) specified in the IPexpress tool.

Table 4-1. File List

File	Description
<username>_inst.v	Provides an instance template for the IP.
<username>_beh.v	Provides the front-end simulation library for the DVB-ASI IP core.
dvb_params.v	Provides the user options of the IP for the simulation model.
<username>_bb.v	Provides the synthesis black box for the user's synthesis.
<username>.ngo	Provides the synthesized IP core.
<username>.lpc	This file contains the IPexpress tool and DVB-ASI IP GUI options used to recreate or modify the core in the IPexpress tool.
<*username*>.ipx	The IPX file holds references to all of the elements of an IP or Module after it is generated from the IPexpress tool (Diamond version only). The file is used to bring in the appropriate files during the design implementation and analysis. It is also used to re-load parameter settings into the IP/Module generation GUI when an IP/Module is being re-generated.
pmi_*.ngo	These files contain the FIFOs used by the IP core. These files need to be pointed to by the Build step by using the search path property.
dvb_eval	This directory contains a sample design. This design is capable of performing a simulation and running through the Diamond or ispLEVER software.

Most of the files required to use the DVB-ASI IP core reside in the root directory created by the IPexpress tool. This includes the synthesis black box, simulation model, and post synthesis NGO files for the PMI modules.

The `\dvb_eval` and subtending directories provide files supporting DVB-ASI IP core evaluation. The `\dvb_eval` directory contains files/folders with content that is constant for all configurations of the DVB-ASI IP core. The `\<username>` subfolder (`\dvb_core0` in this example) contains files/folders with content specific to the <*username*> configuration.

The DVB-ASI ReadMe document is also provided in the `\dvb_eval` directory.

For example information and known issues on this core, see the Lattice DVB-ASI ReadMe document. This file is available when the core is installed in the Diamond or ispLEVER software. The document provides information on creating an evaluation version of the core for use in Diamond or ispLEVER and simulation.

The \dvb_eval directory is created by the IPexpress tool the first time the core is generated and updated each time the core is regenerated. A `\<username>` directory is created by the IPexpress tool each time the core is generated and regenerated each time the core with the same file name is regenerated. A separate \<username> directory is generated for cores with different names, e.g. `\<my_core_0>`, `\<my_core_1>`, etc.

The \dvb_eval directory provides an evaluation design which can be used to determine the size of the IP core and a design which can be pushed through the Diamond or ispLEVER software including front-end and timing simulations. The models directory provides the library element for the PCS and PLLs.

The \<username> directory contains the sample design for the configuration specified by the customer. The \<username>\impl directory provides project files supporting Precision RTL and Synplify synthesis flows. The sample design pulls the user ports out to external pins. This design and associated project files can be used to determine the size of the core and to push it through the mechanics of the Diamond or ispLEVER software design flow.

The `\<username>\sim` directory provides project files supporting RTL and timing simulation for both the Active-HDL and ModelSim simulators. The `\<username>\src` directory provides the top-level source code for the eval design. The `\testbench` directory provides a top-level testbench file.

Running Functional Simulation

Simulation support for the DVB-ASI IP core is provided for Aldec and ModelSim simulators. The DVB-ASI IP core simulation model is generated from the IPexpress tool with the name <username>_beh.v. This file is an obfuscated simulation model. An obfuscated simulation model is Lattice's unique IP protection technique which scrambles the Verilog HDL while maintaining logical equivalence. VHDL users will use the same Verilog model for simulation.

When compiling the DVB-ASI IP core, the file, dvb_params.v must be compiled with the model. This files provides "define constants" that are necessary for the simulation model.

The ModelSim environment is located in \<*project_dir*>\dvb_eval\<*username*>\sim\modelsim. Users can run the ModelSim simulation by performing the following steps:

1. Open ModelSim.
2. Under the **File** tab, select **Change Directory** and choose folder:

   \<*project_dir*>\dvb_eval\<*username*>\sim\modelsim\scripts.

3. Under the **Tools** tab, select **Tcl > Execute Macro** and execute one of the ModelSim "do" scripts shown, depending on whether RTL or netlist simulation is required.

The Aldec Active-HDL environment is located in \<*project_dir*>\dvb_eval\<*username*>\sim\aldec. Users can run the Aldec evaluation simulation by performing the following steps:

1. Open Active-HDL.
2. Under the **Tools** tab, select **Execute Macro**.
3. Browse to the directory \<*project_dir*>\dvb_eval\<*username*>\sim\aldec\scripts and execute one of the Active-HDL "do" scripts shown.

Synthesizing and Implementing the Core in a Top-Level Design

The DVB-ASI IP core itself is synthesized and provided in NGO format when the core is generated through the IPexpress tool. You can combine the core in your own top-level design by instantiating the core in your top-level file as described above in the "Instantiating the Core" section and then synthesizing the entire design with either Synplify or Precision RTL Synthesis.

The top-level file <username>_top.v provided in:

\<*project_dir*>\dvb_eval\<*username*>\src\top\<device_family>

supports the ability to implement the DVB-ASI IP core in isolation. Push-button implementation of this top-level design with either Synplify or Precision RTL Synthesis is supported via the Diamond or ispLEVER software project files <username>_top.syn located in the
\<*project_dir*>\dvb_eval\<*username*>\impl\synplify\core_only
and
\<*project_dir*>\dvb_eval\<*username*>\impl\precision\core_only directories, respectively.

To use this project file in Diamond:

1. Choose **File > Open > Project**.
2. Browse to
   \<*project_dir*>\dvb_eval\<*username*>\impl\synplify (or precision) in the Open Project dialog box.

3. Select and open <*username*>.ldf. At this point, all of the files needed to support top-level synthesis and implementation will be imported to the project.

4. Select the **Process** tab in the left-hand GUI window.

5. Implement the complete design via the standard Diamond GUI flow.

To use this project file in ispLEVER:

1. Choose **File > Open Project.**

2. Browse to \<*project_dir*>\dvb_eval\<*username*>\impl\synplify (or precision) in the Open Project dialog box.

3. Select and open <*username*>.syn. At this point, all of the files needed to support top-level synthesis and implementation will be imported to the project.

4. Select the device top-level entry in the left-hand GUI window.

5. Implement the complete design via the standard ispLEVER GUI flow.

Hardware Evaluation

The DVB-ASI IP core supports supports Lattice's IP hardware evaluation capability, which makes it possible to create versions of the IP core that operate in hardware for a limited period of time (approximately four hours) without requiring the purchase of an IP license. It may also be used to evaluate the core in hardware in user-defined designs.

Enabling Hardware Evaluation in Diamond

Choose **Project > Active Strategy > Translate Design Settings**. The hardware evaluation capability may be enabled/disabled in the Strategy dialog box. It is enabled by default.

Enabling Hardware Evaluation in ispLEVER

In the Processes for Current Source pane, right-click the **Build Database** process and choose **Properties** from the dropdown menu. The hardware evaluation capability may be enabled/disabled in the Properties dialog box. It is enabled by default.

Updating/Regenerating the IP Core

By regenerating an IP core with the IPexpress tool, you can modify any of its settings including device type, design entry method, and any of the options specific to the IP core. Regenerating can be done to modify an existing IP core or to create a new but similar one.

Regenerating an IP Core in Diamond

To regenerate an IP core in Diamond:

1. In IPexpress, click the **Regenerate** button.

2. In the Regenerate view of IPexpress, choose the IPX source file of the module or IP you wish to regenerate.

3. IPexpress shows the current settings for the module or IP in the Source box. Make your new settings in the T**arget** box.

4. If you want to generate a new set of files in a new location, set the new location in the **IPX Target File** box. The base of the file name will be the base of all the new file names. The IPX Target File must end with an .ipx extension.

5. Click **Regenerate.** The module's dialog box opens showing the current option settings.
6. In the dialog box, choose the desired options. To get information about the options, click **Help**. Also, check the About tab in IPexpress for links to technical notes and user guides. IP may come with additional information. As the options change, the schematic diagram of the module changes to show the I/O and the device resources the module will need.
7. To import the module into your project, if it's not already there, select **Import IPX to Diamond Project** (not available in stand-alone mode).
8. Click **Generate**.
9. Check the Generate Log tab to check for warnings and error messages.
10. Click **Close**.

The IPexpress package file (.ipx) supported by Diamond holds references to all of the elements of the generated IP core required to support simulation, synthesis and implementation. The IP core may be included in a user's design by importing the .ipx file to the associated Diamond project. To change the option settings of a module or IP that is already in a design project, double-click the module's .ipx file in the File List view. This opens IPexpress and the module's dialog box showing the current option settings. Then go to step 6 above.

Regenerating an IP Core in ispLEVER

To regenerate an IP core in ispLEVER:

1. In the IPexpress tool, choose **Tools > Regenerate IP/Module**.
2. In the Select a Parameter File dialog box, choose the Lattice Parameter Configuration (.lpc) file of the IP core you wish to regenerate, and click **Open**.
3. The Select Target Core Version, Design Entry, and Device dialog box shows the current settings for the IP core in the Source Value box. Make your new settings in the Target Value box.
4. If you want to generate a new set of files in a new location, set the location in the LPC Target File box. The base of the .lpc file name will be the base of all the new file names. The LPC Target File must end with an .lpc extension.
5. Click **Next**. The IP core's dialog box opens showing the current option settings.
6. In the dialog box, choose desired options. To get information about the options, click **Help**. Also, check the About tab in the IPexpress tool for links to technical notes and user guides. The IP core might come with additional information. As the options change, the schematic diagram of the IP core changes to show the I/O and the device resources the IP core will need.
7. Click **Generate**.
8. Click the **Generate Log** tab to check for warnings and error messages.

This chapter provides supporting information on how to use the DVB-ASI IP core in complete designs. Topics discussed include IP simulation and verification, DVB-ASI sample designs, and board-level implementation and testing.

Simulation and Verification

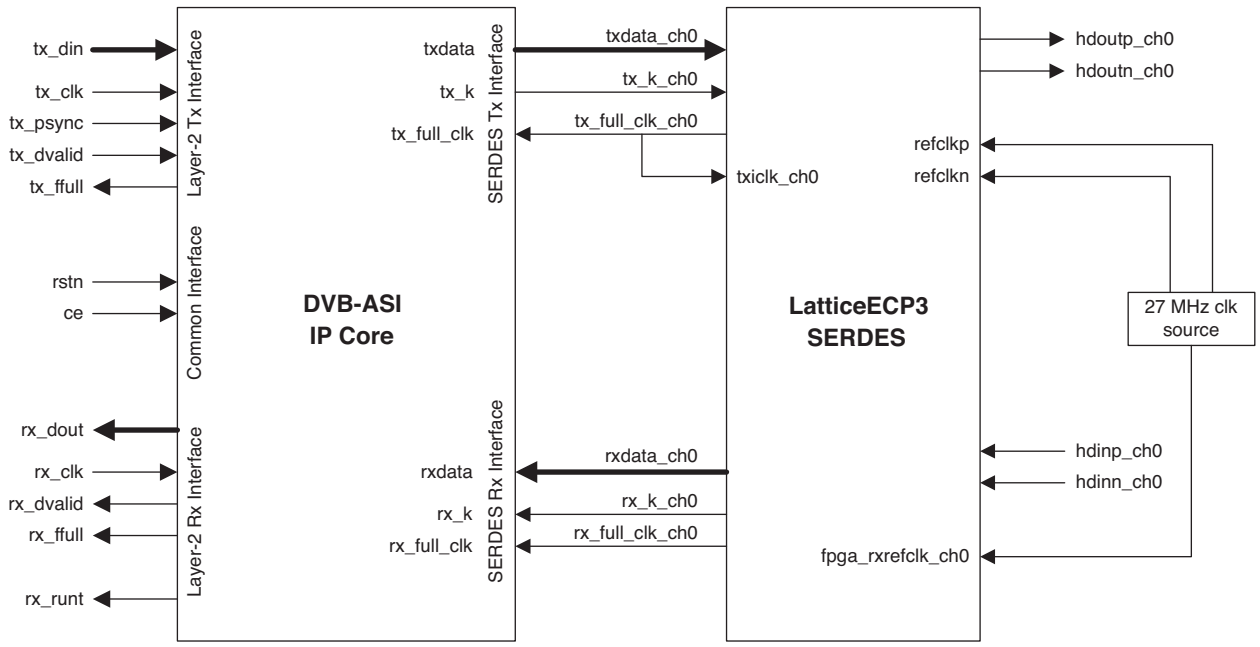
This section discusses strategies and approaches for verifying the proper functionality of the DVB-ASI IP core through simulation.

Simulation Strategy

Included with the core from the IPexpress tool is the evaluation test bench located in the <username> directory. The intent of the evaluation test bench is to show the core performing in simulation, as well as to provide post place and route netlist simulations. Many IP cores work in a loopback format to simplify the data generation process and to meet the simple objectives of this evaluation test bench. A loopback format has been used in this case as well.

The DVB-ASI IP is a single-channel transmit and receive PHY that does not include the SERDES in it. The SERDES can be generated by the user using the .lpc template provided for the SERDES configuration in the DVB-ASI demo designs. The names of the IP ports that connect to the SERDES are made the same as those of the SERDES, except for the channel number suffix. Figure 5-1 shows a typical connection between the IP and the SERDES. In the configuration shown, the SERDES channel 0 is used for Rx and Tx. The SERDES transmitter is configured to use REFCLK and receiver, the FPGA reference clock.

Figure 5-1. Interfacing the DVB-ASI IP Core with LatticeECP3 SERDES

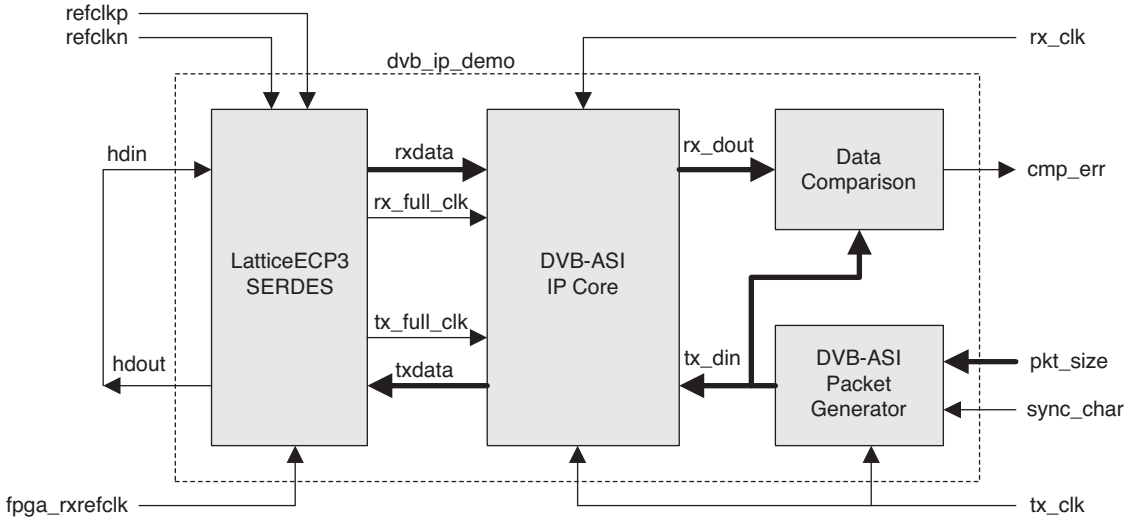


DVB-ASI IP Demo Design

A self-checking top-level design is provided with the IP core that includes the IP, LatticeECP3 SERDES, DVB-ASI traffic generator and a data comparison block. The test bench instantiates the sample design and applies clock and other control signals to it. The demo designs generated along with the IP are in plain Verilog and provide examples of complete DVB-ASI Tx/Rx systems using the IP, SERDES and other test logic.

A block diagram of the single-channel demo design is shown in Figure 5-2.

Figure 5-2. Sample Top-Level (Demo) Design



A brief description of each of the major blocks of the demo design is given below.

DVB-ASI Packet Generator

The DVB-ASI packet generator is a simple LFSR to generate random data for the DVB-ASI packet. Each packet is either 188 or 204 bytes long and a sync byte (47_h or $B8_h$) is inserted at the beginning of each packet. Since the IP is a DVB-ASI Layer 1 implementation, the simple packet generator does not include MPEG-2 packet headers. The payload data does include occurrences of 47_h and $B8_h$ sync-bytes. A random data valid signal is created so the functionality of tx_dvalid can be tested as well.

SERDES/PCS

The LatticeECP3 SERDES/PCS is not a part of the DVB-ASI IP. The demo design includes the SERDES/PCS module that has been pre-generated by IPexpress. The SERDES module is available under `<project_dir>/dvb_eval/models/<device>/dvb_pcs`. The SERDES channel is set for generic 8b10b protocol. The Max. Data Rate is set to 270 Mbps for both Tx and Rx paths. For this sample design, the transmit reference clock is set to external and the receive reference clock to internal. However, users may set the clock sources according to their requirements. The dvb_pcs.lpc file provided under the model directory can be used as a starting point to create custom SERDES modules.

Data Comparison

A data comparison module is provided in the demo design to verify the correctness of the received data. The received data (rx_dout) is compared with the delayed version of the transmitted data (tx_din), using tx_psync and rx_psync signals to synchronize the delay. The result of the comparison is provided at the cmp_err output signal. A synthesizable data comparison block in the top-level sample design absolves the self-checking complexity from the test bench. This also makes the sample design usable for hardware validation of the DVB-ASI functionality.

Test Bench

The test bench instantiates the DVB IP demo design. Since the self-checking capability is provided in the top-level design, the test bench is not required to do data comparisons. The test bench supplies all the required clocks to the demo design. The test bench also counts the number of compare, FIFO full, run packet and giant packet errors.

DVB-ASI Sample Designs

To help users create a complete DVB-ASI transmit/receive system including the SERDES, two sample designs are provided with the generated DVB-ASI IP. The design, dvb_ip_demo, is a single-channel DVB-ASI transmit/receive

system and the design, dvb_ip_demo_2ch is a two-channel design. The single-channel and two-channel designs are available in the following locations:

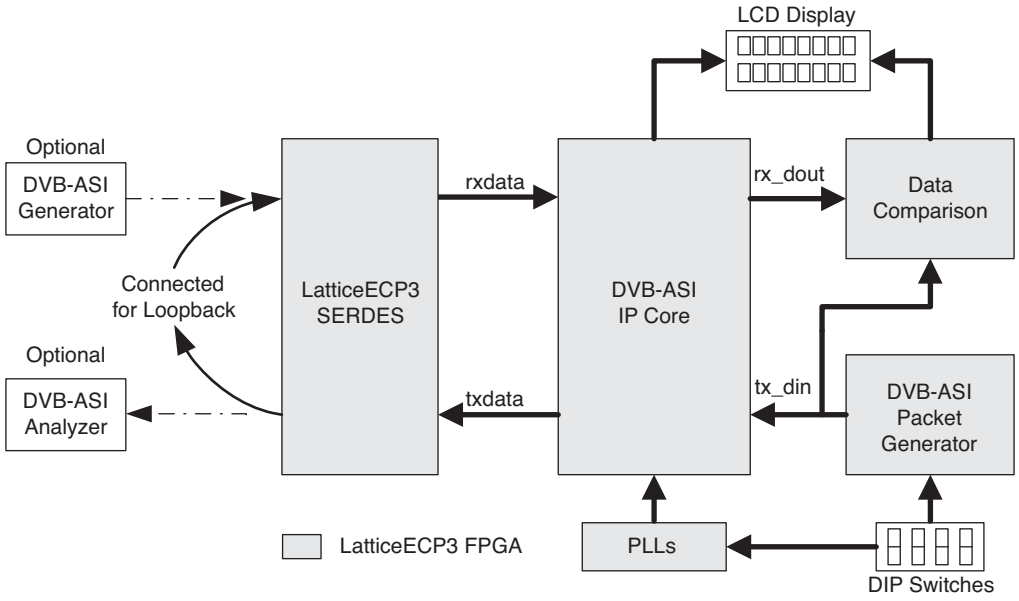
- **Single-channel**: <*project_dir*>/dvb_eval/impl/<*synplify/precision*>/dvb_ip_demo
- **Two-channel**: <*project_dir*>/di_eval/impl/<*synplify/precision*>/dvb_ip_demo_2ch

The sample designs are designed to work on the LatticeECP3 Video Protocol Board (Rev B) using a pushbutton implementation flow. The LatticeECP3 Video Protocol Board has a LFE3-95E-7FN1156CES ECP3 device, clock generator chipsets, crystal oscillators, LEDs and an LCD display unit. The sample designs are designed to work with the IP created using the default parameters in the IP GUI. If one or more parameters are changed from their default values in the IP GUI, the demo designs can still be used, but may require some changes depending on the changed configuration of the IP. This section provides more details of the demo designs and their usage.

Single-Channel Demo Design

The single-channel DVB-ASI demo design can be used to test the transmit (Tx) and receive (Rx) logic of the IP without the help of an external DVB-ASI source or sink. It includes a random data generator module, LatticeECP3 SERDES/PCS, a character LCD display interface and other test logic, in addition to the IP. The demo scheme is shown in Figure 5-3.

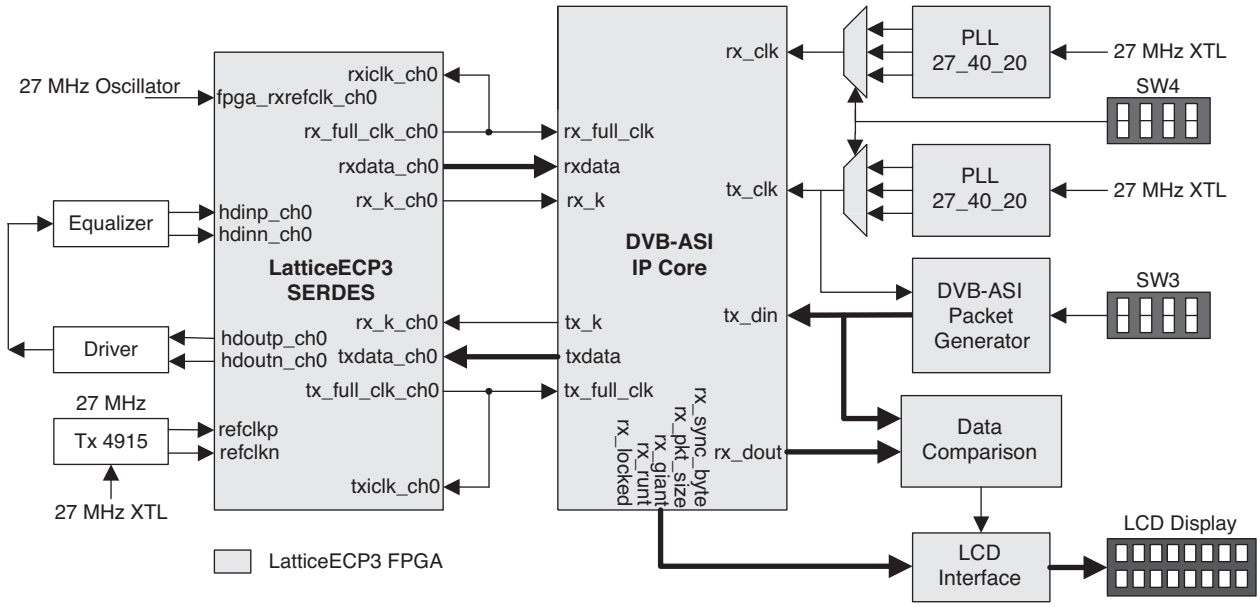
Figure 5-3. DVB-ASI Demo Design Scheme



A DVB-ASI generator and/or analyzer can be optionally used instead of the loopback cable to source and/or monitor the DVB-ASI traffic. All the transmit and receive functions of the IP can be tested with this loopback setup.

A detailed block diagram of the single-channel demo design is shown in Figure 5-4. Some of the functional blocks shown in the figure have been explained in the previous section. A brief description of the other major blocks is given below.

Figure 5-4. Detailed Block Diagram of the Single-Channel DVB-ASI Demo Design



Clocks

The LatticeECP3 Video Protocol Board (Rev B) has two 27 MHz clock sources, one 100 MHz clock and two Gennum GS4911 video clock generator chips. The Tx reference clock for the SERDES comes from the Gennum GS4911 video clock generator and GS4915 clock cleaner chip sets. GS4911 is programmed to generate a 27 MHz clock. The clock output of GS4911 is cleaned by GS4915 to provide a low-jitter Tx reference clock to the SERDES. The SERDES clock outputs, tx_full_clk_ch0 and rx_full_clk_ch0 are directly used to drive the tx_full_clk and rx_full_clk ports of the IP core. In order to test and demonstrate the effect of different rates for the user clocks (tx_clk and rx_clk), three different frequencies (20 MHz, 27 MHz and 40 MHz) are generated from the 27 MHz onboard crystal oscillator clock using a PLL for each transmit and receive path. The PLL outputs are multiplexed and connected to the tx_clk and rx_clk ports of the IP core. The select inputs to the multiplexers are connected to the DIP switches in SW4.

LCD Interface

The LCD interface module provides the interface for the 2-line LCD character display on the LatticeECP3 Video Protocol Board. The module also counts the number of compare, runt and giant errors. As the primary means of showing the test results, the module displays Rx lock condition, sync byte value, packet size, compare, runt and giant errors, and time since reset. The module also has counters that continuously monitor the frequencies of tx_clk, tx_full_clk, rx_full_clk and rx_clk and display them on the LCD display. The details of the LCD display are provided later in this document.

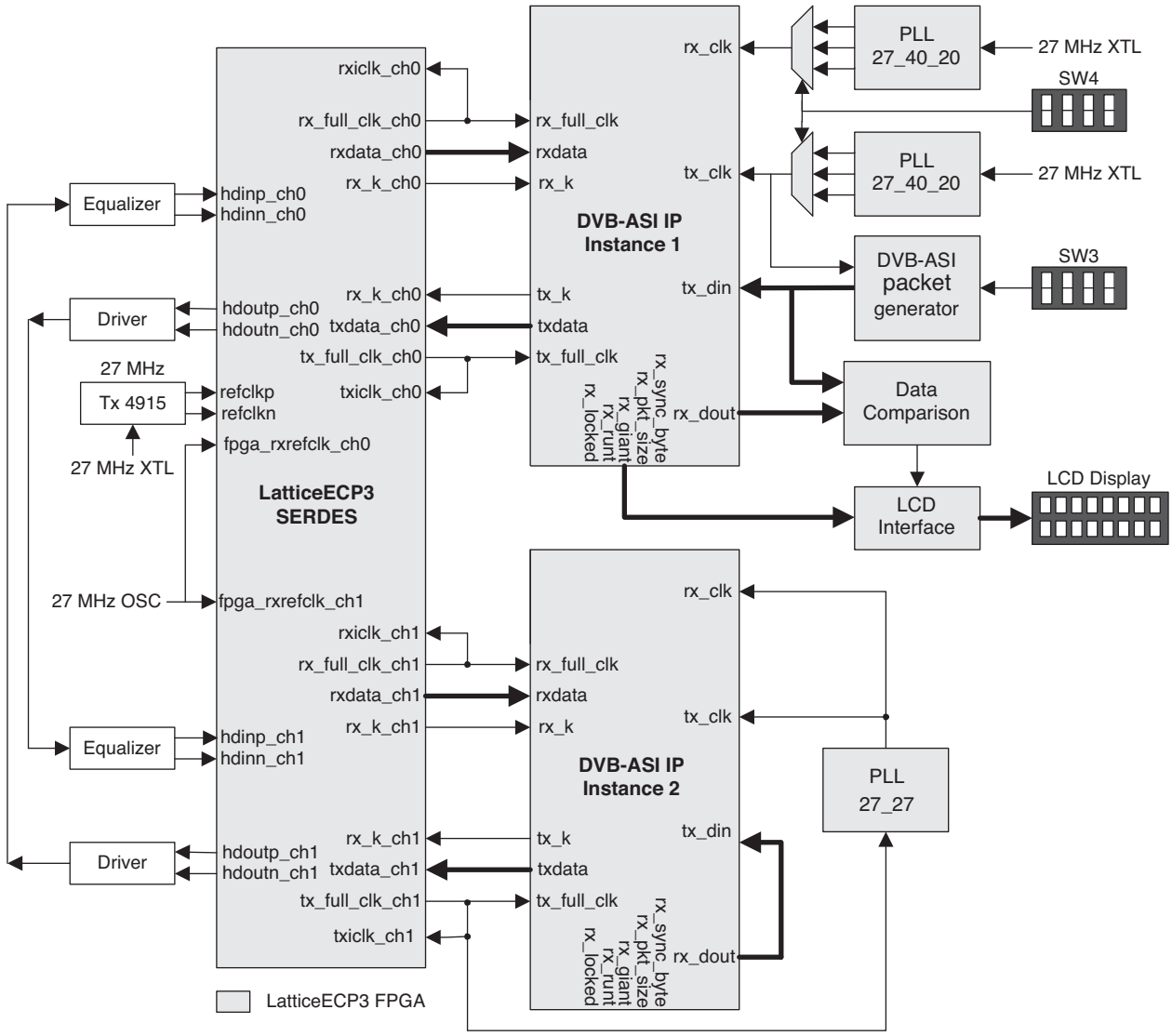
Serial I/Os

The SERDES channel 0 transmit output is connected to a cable driver on the board. The output of the cable driver goes to a 75-ohm BNC connector on the board edge. The external DVB-ASI input comes through a 75-ohm BNC input connector on the board edge which is connected to the cable equalizer on the board. The equalizer output is connected to the SERDES channel 0 Rx. For the loopback tests, the Tx 0 BNC output (labeled SDI Tx #0) is connected to the Rx 0 BNC input (labeled SDI Rx #0) using a coaxial cable. Since the transmit and receive logic are independent of each other, any DVB traffic generator can be connected to the BNC input to observe the reception. However, since the data comparison is between the data created inside the FPGA and that received by it, the comparison will fail if external source and/or sink are used instead of a loopback cable. The BNC output can be connected to a DVB-ASI analyzer to verify the DVB-ASI stream transmitted out of the FPGA. Since the simple DVB-ASI traffic generator in the demo design is not a MPEG-2 transport stream generator, the analyzer will be able to sync to the output, but will not detect a valid DVB-ASI transport stream.

Two-Channel Demo Design

The two-channel demo design is designed to test and demonstrate the functionality of two DVB-ASI channels operating within the same SERDES quad. A detailed block diagram of the two-channel demo design is shown in Figure 5-5. As shown in the diagram, the two-channel design is an extension of the single-channel design with many common modules and a few differences.

Figure 5-5. Detailed Block Diagram of the Two-Channel DVB-ASI Demo Design



The design contains two instances of the DVB-ASI IP cores, one LatticeECP3 SERDES/PCS quad configured for two generic 8b10b channels and other test logic that are part of the single channel design. The description below is focused on the aspects of the two-channel sample design that are different from the single-channel design.

The DVB traffic originates from the DVB-ASI packet generator, goes to the first instance of the IP and is transmitted through the channel 0 Tx of the SERDES quad. The SERDES output goes out to the BNC connector through a cable driver. The BNC output is connected by a cable to the BNC input for channel 1, which goes into SERDES channel 1 Rx through an equalizer. The channel 1 Rx parallel data is connected to the second instance of the DVB-ASI IP. The IP's parallel output, rx_dout, is directly connected to the parallel input, tx_din. This is possible by using the same clock for the second IP instance's tx_clk and rx_clk. The parallel output of IP instance 2 is connected to

SERDES channel 1 transmit. The channel 1 Tx BNC connector is then connected to channel 0 Rx BNC input through a cable. The channel 0 receive logic is similar to that for the single channel design. The final received data and status are displayed on the character LCD display.

Board-Level Implementation and Testing

A simple pushbutton option is offered for implementing the demo designs on the LatticeECP3 Video Protocol Board. The demo designs can be implemented as follows:

- In Diamond, open the project files provided, check **Bitstream File** under Export Files in the Process window, right-click and select **Run**.
- In ispLEVER, open the project files provided and run the **Geneate Bitstream Data** process in the Project Navigator.

Refer to the Diamond or ispLEVER online help for more information.

Some components, like the Gennum chip sets, are controlled by a MachXO™ device on the board. Since the demo designs use Gennum GS 4911 and GS4915 chip sets, it is required that both the LatticeECP3 and the MachXO devices are programmed for proper operation. The source, constraint and project files for the MachXO design are provided under `<project_dir>/dvb_eval/<module_name>/impl/<synplify/precision>/MachXO`. Open the MachXO project by double-clicking on the project file and create the bitstream by following the usual process.

Board Switch Assignments for the Demo Designs

Pushbutton switches

The pushbutton switches on the board are configured as follows:

- **SW10** – Master reset. Depress the switch momentarily for resetting the logic and the SERDES/PCS.
- **SW8** – Error reset. Depress this switch momentarily to reset the compare, Tx FIFO full, Rx FIFO full, runt and giant error counters. This switch also resets the time that is displayed on the LCD display.
- **SW7** – LCD page advance. Depress the switch momentarily for advancing the displayed page of the LCD display.

DIP Switches

The demo designs use the DIP switch banks SW4 and SW3. Each bank has four numbered switches. The switch configurations are shown in Table 5-1. The switches have a “0” value when pushed toward the bottom of the board (toward the PCI Express fingers) and a “1” value when pushed toward the top of the board (toward the channel link connector).

Table 5-1. Switch Connections on the LatticeECP3 Video Prototocol Board

Switch	Name	Description
SW4-4	Transmit Clock Frequency Selection	tx_clk_sel = {SW4-4,SW4-3}. 00 - 20 MHz, 01- 27 MHz (PLL out), 10- 40 MHz, 11- 27 MHz (PLL in)
SW4-3		
SW4-2	Receive Clock Frequency Selection	rx_clk_sel = {SW4-2,SW4-1}. 00 – 20 MHz, 01- 27 MHz (PLL out), 10- 40 MHz, 11- 27 MHz (PLL in)
SW4-1		
SW3-4	Tx Sync Character	tx_sync_char 0- 47_h, 1- $B8_h$
SW3-3	Tx Packet Size	tx_pkt_size = {SW3-3,SW3-2,SW3-1} 000- 188, 001- 204, 010- 187, 011- 203 100- 189, 101- 205, 110- 175, 111- 220
SW3-2		
SW3-1		

LEDs

The LED status (0 is off, 1 is lit) is shown in Table 5-2.

Table 5-2. LED Status Display

LED	LED Color	Name	Description
D26	Red	Tx FIFO Full	tx_ffull. Lights when Tx FIFO Full goes high
D25	Orange	Rx FIFO Full	rx_ffull. Lights when Rx FIFO Full goes high
D23	Green	RUNT	rx_runt. Lights when runt packets are received
D22	Blue	GIANT	rx_giant. Lights when giant packets are received
D17	Red	Rx Loss of Lock	rx_lol_ch0. Rx loss of lock: 0 - Rx CDR is locked, 1 - Rx CDR loss of lock error
D16	Orange	Compare Error	cmp_err output. Lights when there is data comparison error
D15	Green	Packet Size	pkt_size output. This identifies the size of the packet. 0 - 188, 1 - 204.
D14	Blue	Sync Byte	sync_byte output. This identifies the received sync byte. $0\text{-}47_h$, $1\text{-}B8_h$.
D13	Red	Tx Loss of Lock	pll_lol. SERDES transmit PLL loss of lock: 0- PLL locked, 1- PLL unlocked
D12	Orange	Rx Loss of Signal	rx_los_low_ch0_s. 0 - Signal detected at SERDES input. 1 - No signal detected at SERDES input
D11	Green	Rx No Lock	Rx not locked. This is the complement of rx_locked output.
D10	Blue	Rx Locked	rx_locked. 0 - Receiver not locked to DVB-ASI, 1 - Receiver locked to DVB-ASI

LCD Display

The demo design includes an interface for 2-line, 16-character LCD display. There are four display pages in the LCD display as shown in Table 5-3.

Table 5-3. Display Pages

Page	Line	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Page 0	Line 1	T	x	C		tx_clk						Tx pkt_size				Tx sync	
	Line 2	Locked/Unknown/No Signal										Rx pkt_size				Rx sync	
Page 1	Line 1	T	x	C		tx_clk						f		tx_full_clk			
	Line 2	R	x	C		rx_clk						f		rx_full_clk			
Page 2	Line 1	T	F	F		tx_ffull cnt					R	F	F		rx_ffull cnt		
	Line 2	E	r		cmp_err cnt				T	i	m	e		Hr : Sec Sec			
Page 3	Line 1	R	N	T		rx_runt cnt					G	N	T		rx_giant cnt		
	Line 2	E	r		cmp_err cnt				T	i	m	e		Hr : Sec Sec			

Page 0 displays the frequency of tx_clk, Tx and Rx packet sizes (188 0r 204), Tx and Rx sync bytes (47_h or $B8_h$) and receiver lock status. The receiver is “Locked” when rx_locked is high, “Unknown” when rx_locked, Rx loss of lock and Rx loss of signal are all low and “No Signal” when there is a Rx loss of signal or loss of lock error.

Page 1 displays the measured frequencies of tx_clk, tx_full_clk, rx_full_clk and rx_clk.

Page 2 displays the number of Tx FIFO Full, Rx FIFO full and Data compare errors. It also displays the time since reset. The error values saturate at 999. The timer saturates at 9 hours and 59 minutes.

Page 3 displays the number of Rx runt, Rx giant and Data compare errors. It also displays the time since reset. The errors values saturate at 999. The timer saturates at 9 hours and 59 minutes.

On reset, the LCD displays page 0. The pushbutton switch SW7 is used to cycle the display through the pages. Depressing pushbutton switch SW8 resets the error counts for compare error, Tx FIFO full, Rx FIFO full, runt and giant. The time display, which is the time elapsed since reset, is also reset to 0:00 when SW8 is depressed.

Core Verification

The functionality of the Lattice DVB-ASI IP core has been verified via simulation and hardware testing in a variety of environments, including:

- Simulation environment verifying DVB-ASI functionality using pseudo random number generator. The simulation was carried out using RTL as well as using post-place and route netlist.
- Hardware testing of the simulation environment in loopback mode, with a cable providing the loopback of serial transmit out to serial receive in.
- Hardware testing of 2-channel sample design, with cable connecting channel 0 transmit to channel 1 receive and channel 1 transmit to channel 0 receive.
- Hardware testing of the sample designs by using a standard DVB-ASI source and sink. A setup similar to the loopback scheme was used, but either the source or sync or both were replaced by a DVB-ASI standard source and/or sink.

Lattice Technical Support

There are a number of ways to receive technical support as listed below.

Online Forums

The first place to look is Lattice Forums (www.latticesemi.com/support/forums.cfm). Lattice Forums contain a wealth of knowledge and are actively monitored by Lattice Applications Engineers.

Telephone Support Hotline

Receive direct technical support for all Lattice products by calling Lattice Applications from 5:30 a.m. to 6 p.m. Pacific Time.

- For USA and Canada: 1-800-LATTICE (528-8423)
- For other locations: +1 503 268 8001

In Asia, call Lattice Applications from 8:30 a.m. to 5:30 p.m. Beijing Time (CST), +0800 UTC. Chinese and English language only.

- For Asia: +86 21 52989090

E-mail Support

- techsupport@latticesemi.com
- techsupport-asia@latticesemi.com

Local Support

Contact your nearest Lattice sales office.

Internet

www.latticesemi.com

Lattice DVB-ASI Web Page

For more information on Lattice's DVB-ASI and other video solutions, visit the DVB-ASI IP core web page at www.latticesemi.com/products/intellectualproperty/ipcores/dvbasi.cfm.

References

DVB-ASI

- EN 50083-9, Part 9 – Interfaces for CATV/SMATV headend and similar professional equipment for DVB/MPEG-2 transport streams.
- ISO/IEC 13818-1 Information Technology – Generic coding of moving pictures and associated audio information Systems.

LatticeECP3

- HB1009, LatticeECP3 Family Handbook
- TN1176, LatticeECP3 SERDES/PCS Usage Guide

Evaluation Boards

- LatticeECP3 Video Protocol Evaluation Board Web Page
  www.latticesemi.com/products/fpga/ecp3/ecp3evalboards/ecp3videoprotocolboard.cfm
- EB39, LatticeECP3 Video Protocol Board- Revision B User's Guide

Revision History

Date	Document Version	IP Core Version	Change Summary
April 2010	01.0	1.0	Initial release.
December 2010	01.1	1.1	Added support for Diamond software throughout.