XAUI IP Core User's Guide

Table of Contents

The 10Gb Ethernet Attachment Unit Interface (XAUI) IP Core User's Guide for the LatticeECP2M™ and LatticeECP3™ FPGAs provides a solution for bridging between XAUI and 10-Gigabit Media Independent Interface (XGMII) devices. This IP core implements 10Gb Ethernet Extended Sublayer (XGXS) capabilities in soft logic that together with PCS and SERDES functions implemented in the FGPA provides a complete XAUI-to-XGMII solution.

The XAUI IP core package comes with the following documentation and files:

- Protected netlist/database

- Behavioral RTL simulation model

- Source files for instantiating and evaluating the core

The XAUI IP core 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 on an IP license. It may also be used to evaluate the core in hardware in user-defined designs. Details for using the hardware evaluation capability are described in the Hardware Evaluation section of this document.

Quick Facts

Table 1-1 gives quick facts about the XAUI IP core.

Table 1-1. XAUI IP Core Quick Facts

		XAUI IP Configuration	
		Across All IP configurations	
Core Requirements	FPGA Families Supported	Lattice ECP3, Lattice ECP2M	
	Minimal Device Supported	LFE3-17E-7FN256C	LFE2M20E-6F256C
Resource Utilization	Data Path Width	72 bits	72 bits
	LUTs	1700-2700	1800-2800
	sysMEM EBRs	0-4	0-4
	Registers	1500-2700	1500-2700
Design Tool Support	Lattice Implementation	Diamond® 1.0 or ispLEVER® 8.1	
	Synthesis	Synopsys® Synplify™ Pro for Lattice D-2009.12L-1	
		Mentor Graphics® Precision™ RTL	
	Simulation	Aldec® Active-HDL™ 8.2 Lattice Edition	
		Mentor Graphics ModelSim™ SE 6.3F (Verilog only)	

Features

- XAUI compliant functionality supported by embedded SERDES PCS functionality implemented in the LatticeECP2M and LatticeECP3, including four channels of 3.125 Gbps serializer/deserializer with 8b10b encoding/decoding.

- Complete 10Gb Ethernet Extended Sublayer (XGXS) solution based on LatticeECP2M and LatticeECP3 FPGA.

- Soft IP targeted to the FPGA implements XGXS functionality conforming to IEEE 802.3-2005, including:
  - 10 GbE Media Independent Interface (XGMII).
  - Optional slip buffers for clock domain transfer to/from the XGMII interface.

- Complete translation between XGMII and XAUI PCS layers, including 8b10b encoding and decoding of Idle, Start, Terminate, Error and Sequence code groups and sequences, and randomized Idle generation in the XAUI transmit direction.
- XAUI compliant lane-by-lane synchronization.
- Lane deskew functionality.
- Interface with the high-speed SERDES block embedded in the LatticeECP2M and LatticeECP3 that implements a standard XAUI.
- Optional standard compliant MDIO/MDC interface.

• Aldec and ModelSim simulation models and test benches provided for free evaluation.

Functional Description

XAUI is a high-speed interconnect that offers reduced pin count and has the ability to drive up to 20 inches of PCB trace on standard FR-4 material. Each XAUI comprises four self-timed 8b10b encoded serial lanes each operating at 3.125 Gbps and thus is capable of transferring data at an aggregate rate of 10 Gbps.

XGMII is a 156 MHz Double Data Rate (DDR), parallel, short-reach interconnect interface (typically less than 2 inches). It supports interfacing to 10 Gbps Ethernet Media Access Control (MAC) and PHY devices.

The locations of XAUI and XGXS in the 10GbE protocol stack are shown in Figure 2-1. A simplified block diagram of the XAUI solution is shown in Figure 2-2. The XGMII interface, XGXS coding and state machines and XAUI multichannel alignment capabilities are implemented in the FPGA array. The XAUI 8b10b coding and SERDES functionality are supported by the embedded SERDES_PCS block. An optional MDIO interface module is also implemented in the FPGA array.

Figure 2-3 shows the I/O interface view of the XAUI IP core.

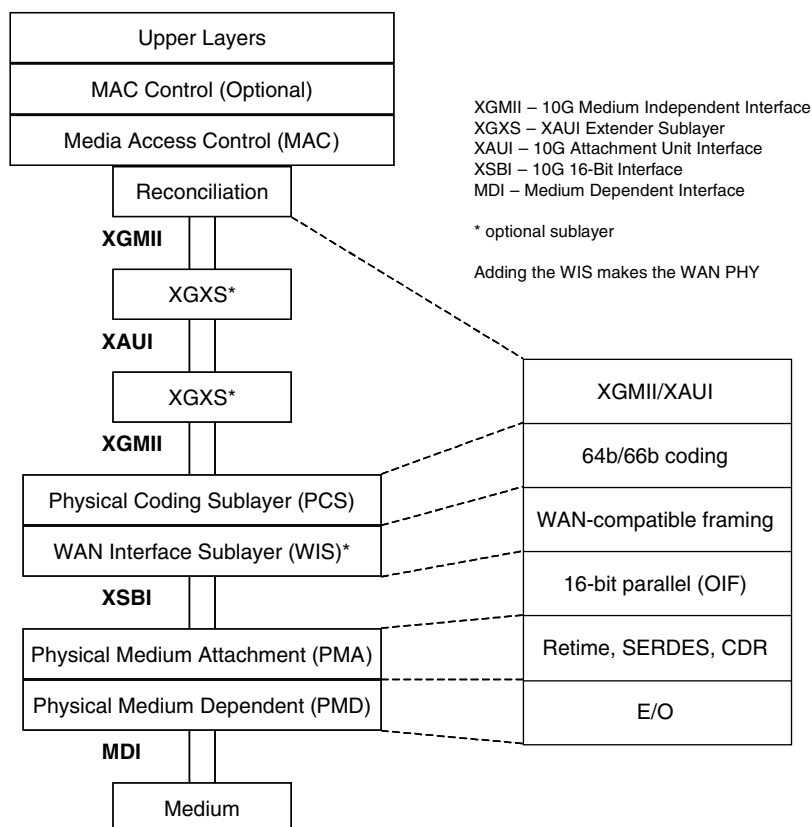*Figure 2-1. XAUI and XGXS Locations in 10 GbE Protocol Stack*

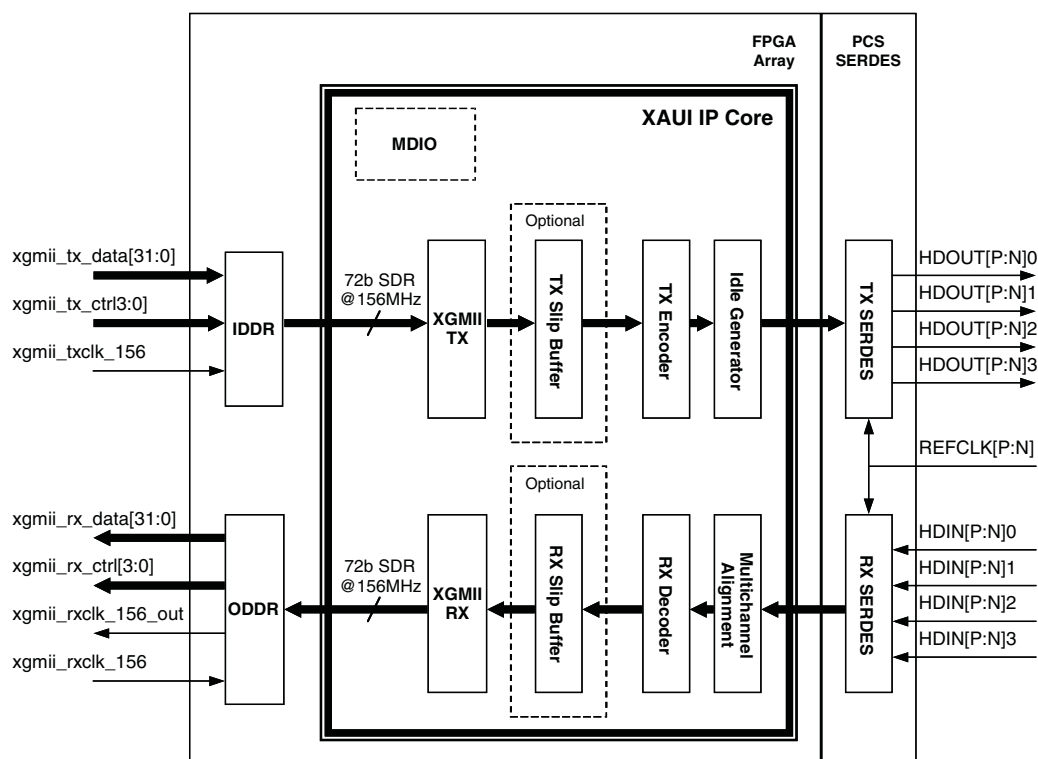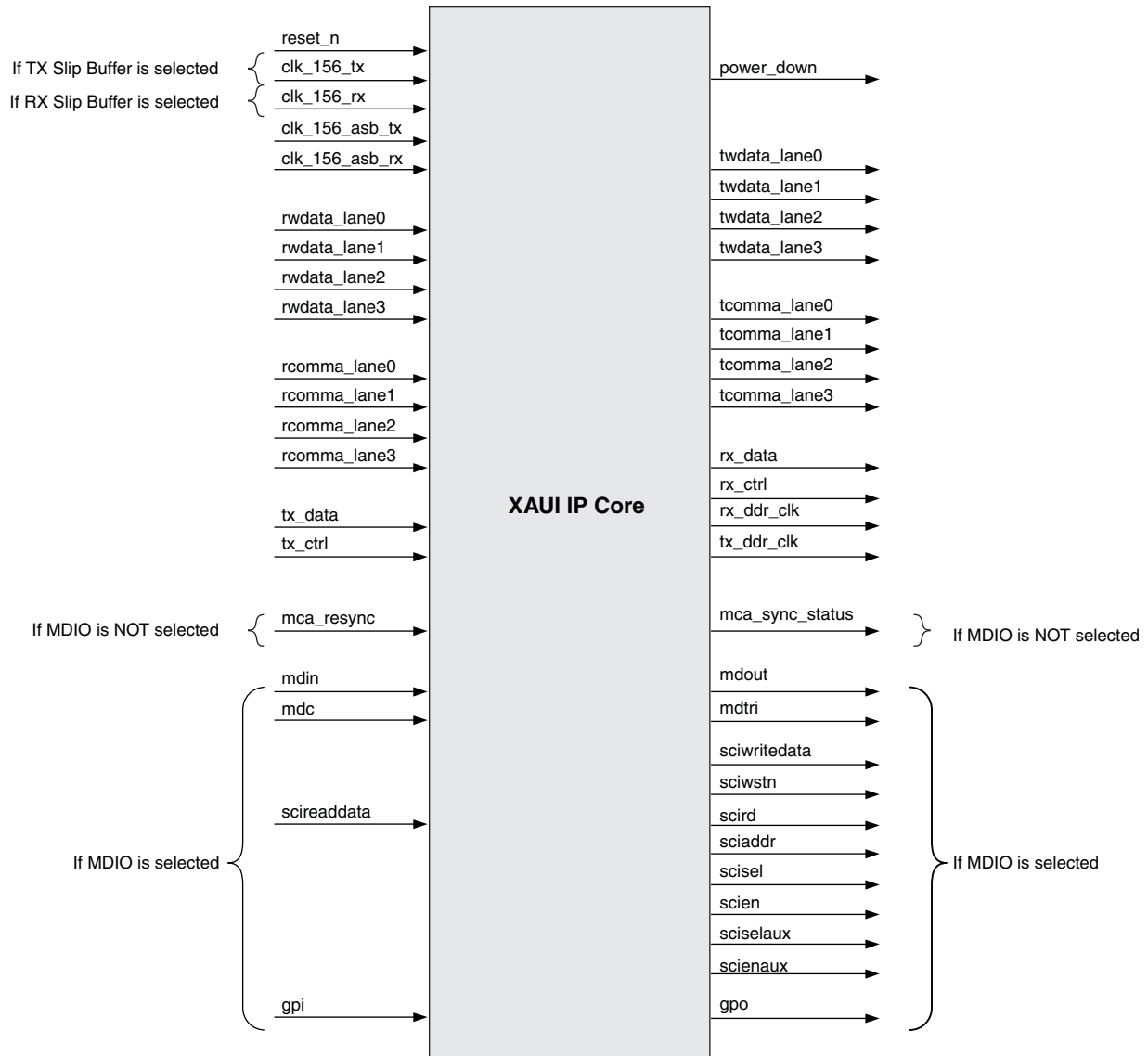*Figure 2-2. XAUI Solution Simplified Block Diagram*

Figure 2-3. XAUI IP Core I/O

XAUI IP Core I/O

Table 2-1 defines all I/O interface ports available in this core.

Table 2-1. IP Core I/O and Signal Definitions

Name	Width (Bits)	Direction	Description
All Configurations			
reset_n	1	I	Active-low asynchronous reset
clk_156_asb_tx	1	I	156MHz XAUI transmit clock (from PCS)
clk_156_asb_rx	1	I	156MHz XAUI receive clock (from PCS)
rwdata_lane0	16	I	XAUI receive data channel 0 (from PCS)
rwdata_lane1	16	I	XAUI receive data channel 1 (from PCS)
rwdata_lane2	16	I	XAUI receive data channel 2 (from PCS)
rwdata_lane3	16	I	XAUI receive data channel 3 (from PCS)
rcomma_lane0	2	I	XAUI receive control channel 0 (from PCS)
rcomma_lane1	2	I	XAUI receive control channel 1 (from PCS)
rcomma_lane2	2	I	XAUI receive control channel 2 (from PCS)
rcomma_lane3	2	I	XAUI receive control channel 3 (from PCS)
twdata_lane0	16	O	XAUI transmit data channel 0 (to PCS)
twdata_lane1	16	O	XAUI transmit data channel 1 (to PCS)
twdata_lane2	16	O	XAUI transmit data channel 2 (to PCS)
twdata_lane3	16	O	XAUI transmit data channel 3 (to PCS)
tcomma_lane0	2	O	XAUI transmit control channel 0 (to PCS)
tcomma_lane1	2	O	XAUI transmit control channel 1 (to PCS)
tcomma_lane2	2	O	XAUI transmit control channel 2 (to PCS)
tcomma_lane3	2	O	XAUI transmit control channel 3 (to PCS)
rx_ddr_clk	1	O	DDR clock from IP Core to the XGMII TX direction
tx_ddr_clk	1	O	DDR clock from IP Core to the XGMII receive direction
tx_data	64	I	IP Core transmit data from XGMII RX
tx_ctrl	8	I	IP Core transmit control from XGMII RX
rx_data	64	O	Receive data from core and sent to XGMII TX
rx_ctrl	8	O	Receive control from core and sent to XGMII TX
With Optional Rx Slip Buffer			
clk_156_rx	1	I	156MHz XGMII TX clock
rx_fifo_empty	1	O	Rx slip buffer FIFO empty flag
rx_fifo_full	1	O	Rx slip buffer FIFO full flag
With Optional Tx Slip Buffer			
clk_156_tx	1	I	156MHz XGMII RX clock
tx_fifo_empty	1	O	Tx slip buffer FIFO empty flag
tx_fifo_full	1	O	Tx slip buffer FIFO full flag
No MDIO Option			
mca_resync	1	I	XAUI multi-channel alignment resynchronization request
mca_sync_status	1	O	XAUI multi-channel alignment status. 1 = All XAUI channels are aligned 0 = XAUI channels are not aligned
MDIO Option			
mdin	1	I	MDIO serial input data
mdc	1	I	MDIO input clock

Table 2-1. IP Core I/O and Signal Definitions (Continued)

Name	Width (Bits)	Direction	Description
mdout	1	O	MDIO serial output data
mdtri	1	O	Tristate control for MDIO port
scireaddata	8	I	SCI read data (from PCS)
sciwritedata	8	O	SCI write data (to PCS)
sciwstn	1	O	SCI write strobe
scird	1	O	SCI read probe
sciaddr	6	O	SCI address bus
scisel	4	O	SCI quad select
scien	4	O	SCI quad enable
sciselaux	1	O	SCI auxiliary select
scienaux	1	O	SCI auxiliary enable
gpi	4	I	General purpose input
gpo	4	O	General purpose output

Functional Description

The XAUI receive path, shown in Figure 2-4, is the data path from the XAUI to the XGMII interface. In the receive direction, 8b10b encoded data received at the XAUI SERDES interface is demultiplexed and passed to the multi-channel alignment block, that compensates for lane-to-lane skew between the four SERDES channels as specified in 802.3-2005. The aligned data streams are passed to decoder logic, where it is translated and mapped to the XGMII data format. The output of the encoder is then passed through an optional slip buffer that compensates for XAUI and XGMII timing differences and then to the XGMII 36-bit (32-bit data and four control bits) 156 MHz DDR external interface.

The receive direction data translations are shown in Figure 2-5. The 8b10b symbols on each XAUI lane are converted to XGMII format and passed to the corresponding XGMII lane. The XGMII comprises four lanes, labeled [0:3], and one clock in both transmit and receive directions. Each lane includes eight data signals and one control signal. Double Data Rate (DDR) transmission is utilized, with the data and control signals sampled on both the rising and falling edges of a 156.25 MHz (nom) clock for an effective data transfer rate of 2.5Gbps.

Figure 2-4. XAUI IP Core Receive Path

With Optional Receive Direction Slip Buffer
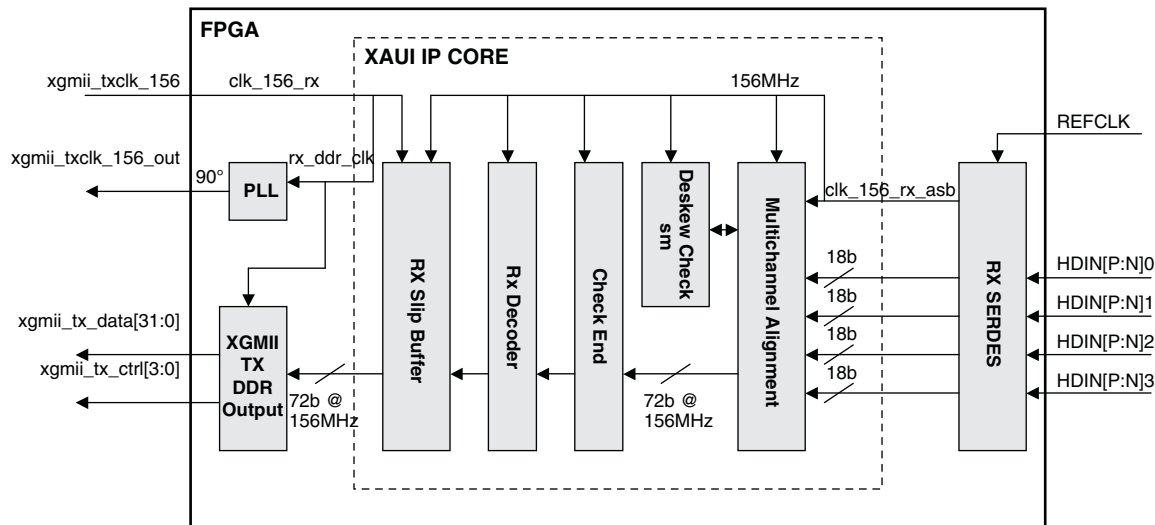
Without Optional Receive Direction Slip Buffer
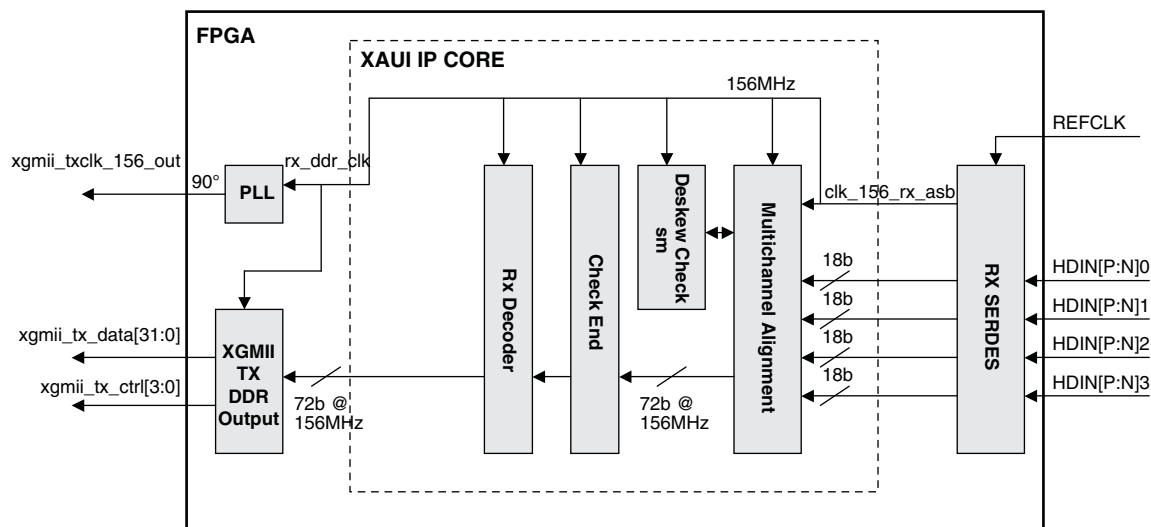
Figure 2-5. XAUI IP Core Receive Direction Data Translations
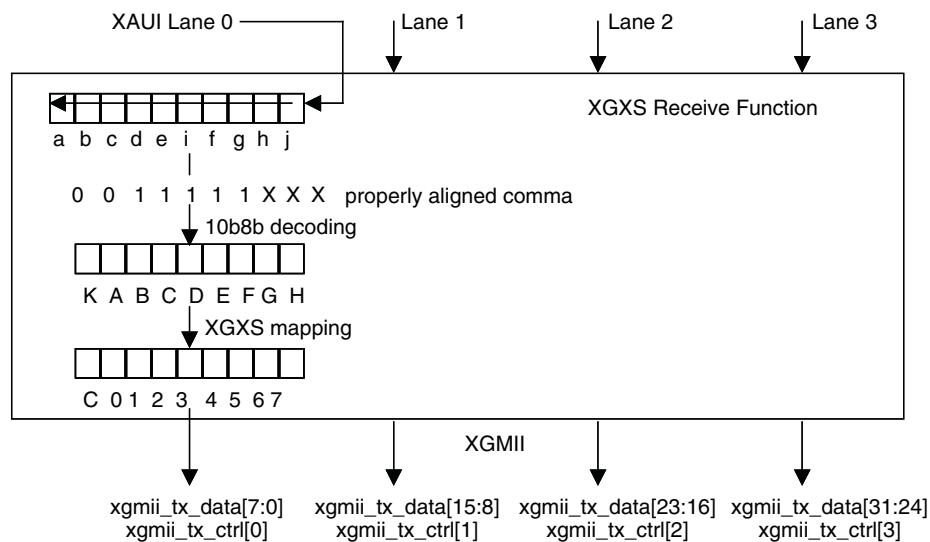
The transmit path, shown in Figure 2-6, is the data path from XGMII to XAUI. In the transmit direction, the 36-bit DDR data and control received at the XGMII are converted to single-edge timing and passed through an optional slip buffer that compensates for XAUI and XGMII timing differences. The XGMII data and control are then passed to the TX encoder, where they are translated and mapped to the 8b10b XAUI transmission code and then passed to the SERDES interface.

The transmit direction data translations are shown in Figure 2-7. Data and control from each of the four XGMII lanes are translated and mapped to the corresponding XAUI lanes. The transmit encoder includes the transmit idle generation state machine that generates a random sequence of /A/, /K/ and /R/ code groups as specified in IEEE 802.3-2005.
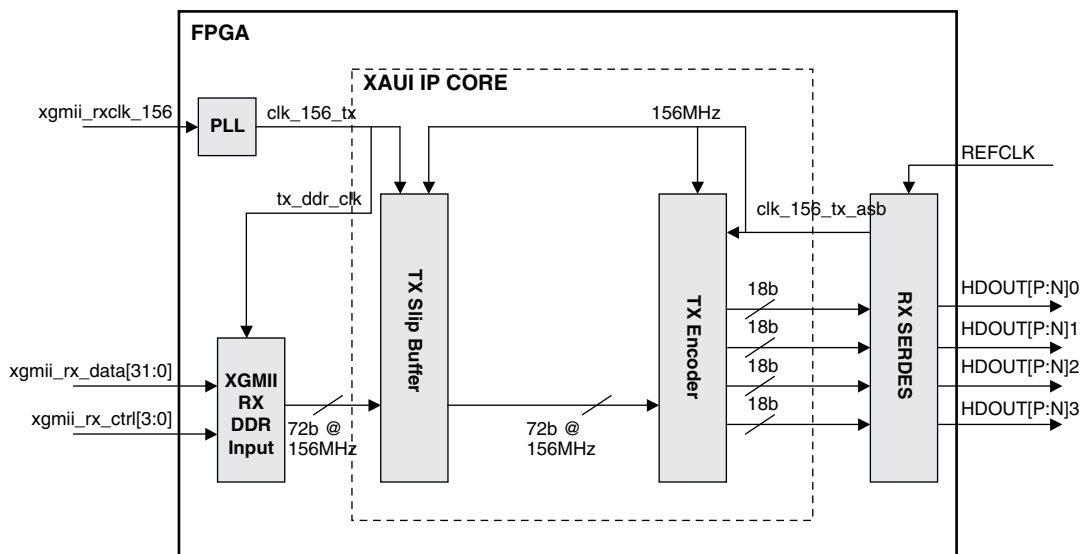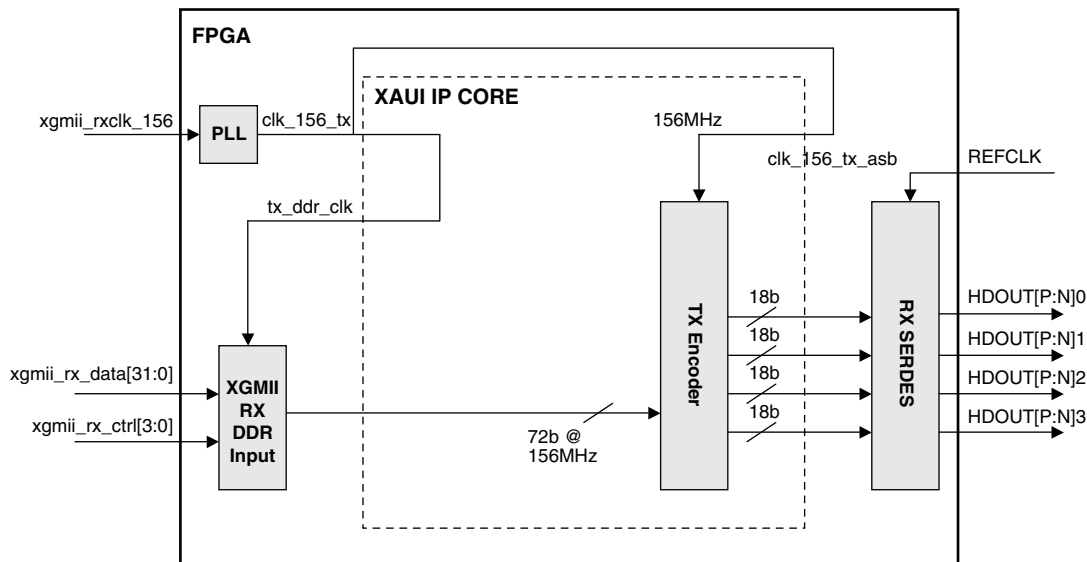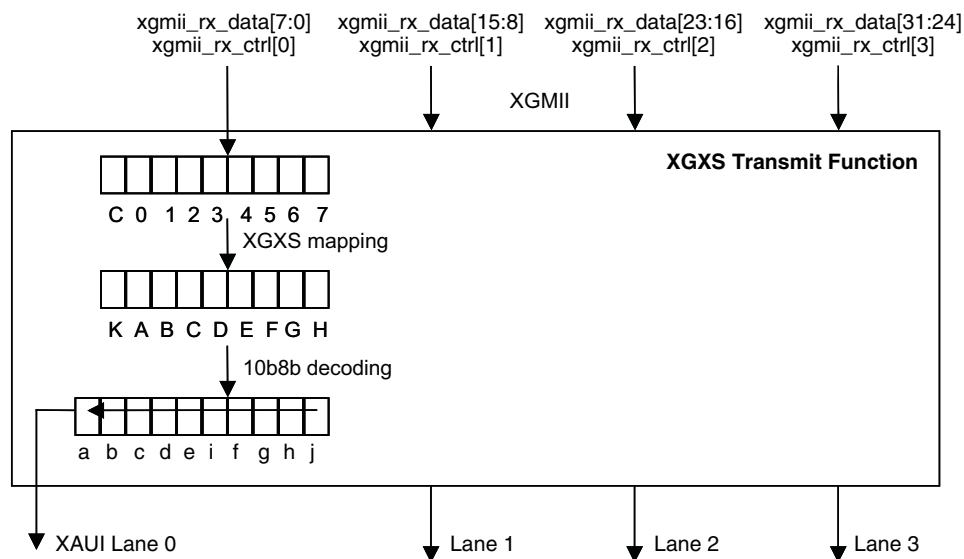
Figure 2-6. XAUI IP Core Transmit Path

With Optional Transmit Direction Slip Buffer

Without Optional Transmit Direction Slip Buffer

Figure 2-7. XAUI IP Core Transmit Direction Data Translations

XGMII and Slip Buffers

The 10Gigabit Media Independent Interface (XGMII) supported by the XAUI IP core solution conforms to Clause 46 of IEEE 802.3-2005. The XGMII is composed of independent transmit and receive paths. Each direction uses 32 data signals, four control signals and a clock. The 32 data signals in each direction are organized into four lanes of eight signals each. Each lane is associated with a control signal as shown in Table 2-2.

Table 2-2. XGMII Transmit and Receive Lane Associations[1]

Tx data (xgmii_rx_data) Rx data (xgmii_tx_data)	Tx control (xgmii_rx_ctrl) Rx control (xgmii_tx_ctrl)	XAUI Lane
[7:0]	[0]	0
[15:8]	[1]	1
[23:16]	[2]	2
[31:24]	[3]	3

1. The XGMII TX signals are XGXS inputs to the transmit path (XGMII to XAUI). The XGMII RX signals are XGXS outputs of the receive path (XAUI to XGMII).

The XGMII supports Double Data Rate (DDR) transmission (i.e., the data and control input signals are sampled on both the rising and falling edges of the corresponding clock). The XGXS XGMII input (RX) data is sampled based on an input clock typically sourced from the XGMII device running at 156.25 MHz, 1/64th of the 10Gb data rate and is transmitted by the IP transmit part. The XGXS XGMII output (TX) data is referenced to a forwarded clock that is phase locked to a 156.25 MHz (typical) input reference, and the data is received from the IP receiver part.

The control signal for each lane is de-asserted when a data octet is being sent on the corresponding lane and asserted when a control character is being sent. Supported control octet encodings are shown in Table 2-3. All data and control signals are passed directly to/from the 8b10b encoding/decoding blocks with no further processing by the XGMII block. Note that the packet Start control word is only valid on lane 0.

Table 2-3. XGMII Control Encoding

Control	Data	Description
0	0x00 - 0xFF	Normal data transmission
1	0x00 - 0x06	Reserved
1	0x07	Idle
1	0x08 - 0x9B	Reserved
1	0x9C	Sequence (only valid on lane 0)
1	0x9D - 0xFA	Reserved
1	0xFB	Start (only valid on lane 0)
1	0xFC	Reserved
1	0xFD	Terminate
1	0xFE	Error
1	0xFF	Reserved

The XGMII blocks incorporate optional slip buffers that accommodate small differences between XGMII and XAUI timing by inserting or deleting idle characters. The slip buffer is implemented as a 256 x 72 FIFO. There are four flags out of the FIFO: full, empty, partially full, and partially empty. The partially empty flag is used as the watermark to start reading from the FIFO. If the difference between write and read pointers falls below the partially empty watermark and the entire packet has been transmitted, idle characters are inserted until the partially full watermark is reached. No idle is inserted during data transmission.

XAUI-to-XGMII Translation (Receive Interface)

A block diagram of the XAUI IP core receive data path was shown previously in Figure 2-4. The IP receive interface converts the incoming XAUI stream into XGMII-compatible signals. At the embedded core interface, the IP core receive block receives 72 bits of data at 156 MHz (64 bits of data, 8 bits of control) from four XAUI lanes. Data from the embedded core are first passed to the RX multi-channel aligner block to de-skew the four XAUI lanes.

The multi-channel alignment block consists of four 16-byte deep FIFOs to individually buffer each XAUI lane. The multi-channel alignment block searches for the presence of the alignment character /A/ in each XAUI lane, and begins storing the data in the FIFO when the /A/ character is detected in a lane. When the alignment character has been detected in all four XAUI lanes, the data is retrieved from each FIFO so that all of the alignment characters /A/ are aligned across all four XAUI lanes. Once synchronization is achieved, the block does not resynchronize until a resynchronization request is issued.

The data from the RX multi-channel alignment is passed to the RX decoder. The RX decoder block converts the XAUI code to the corresponding XGMII code. Table 2-4 shows the 8b10b code points. Table 2-5 shows the code mapping between the two domains in the receive direction. XAUI /A/, /R/, /K/ characters are translated into XGMII Idle (/I/) characters.

Data from the RX decoder block is written to the RX slip buffer. As mentioned previously, the slip buffers are required to compensate for differences in the write and read clocks derived from the XAUI and XGMII reference clocks, respectively. Clock compensation is achieved by deleting (not writing) idle cells into the buffer when the "almost full" threshold is reached and by inserting (writing) additional idle cells into the buffer when the "almost empty" threshold is reached. All idle insertion/deletion occurs during the Inter-Packet Gap (IPG) between data frames.

Table 2-4. XAUI 8b10b Code Points

Symbol	Name	Function	Code Group
/A/	Align	Lane Alignment (XGMII Idle)	K28.3
/K/	Sync	Code-Group Alignment (XGMII Idle)	K28.5
/R/	Skip	Clock Tolerance Compensation (XGMII Idle)	K28.0
/S/	Start	Start of Packet Delimiter (in Lane 0 only)	K27.7
/T/	Terminate	End of Packet Delimiter	K29.7
/E/	Error	Error Propagation	K30.7
/Q/	Sequence	Link Status Message Indicator	K28.4
/d/	Data	Information Bytes	Dxx.x

Table 2-5. XAUI 8b10b to XGMII Code Mapping

8b10b Data from Embedded Core [7:0]	XGMII Data [7:0]	XGMII Control [3:0]
Dxx.x	0x00-0xFF (Data)	0
K28.5 (0xBC)	0x07 (Idle)	1
K28.3 (0x7C)	0x07 (Idle)	1
K28.0 (0x1C)	0x07 (Idle)	1
K27.7 (0xFB)	0xFB (Start)	1
K29.7 (0xFD)	0xFD (Terminate)	1
K30.7 (0xFE)	0xFE (Error)	1
K28.4 (0x9C)	0x9C (Ordered Set)	1

XGMII-to-XAUI Translation (Transmit Interface)

A block diagram of the XAUI IP core transmit data path was shown previously in Figure 2-6. The TX interface converts the incoming XGMII data into XAUI-compatible characters. 36-bit XGMII DDR input data and control signals are initially converted to a 72-bit bus based on a single edge 156MHz clock. The data and control read are then passed into an optional TX slip buffer identical to the one used for the RX interface.

After the slip buffer, the XGMII formatted transmit data and control are input to the TX encoder that converts the XGMII characters into 8b10b format as shown in Table 7. The idle generation state machine in the TX encoder converts XGMII /I/ characters to a random sequence of XAUI /A/, /K/ and /R/ characters as specified in IEEE 802.3-2005. XGMII idles are mapped to a random sequence of code groups to reduce radiated emissions. The /A/ code groups support XAUI lane alignment and have a guaranteed minimum spacing of 16 code-groups. The /R/ code groups are used for clock compensation. The /K/ code groups contain the 8b10b comma sequence.

Table 2-6. XGMII to XAUI Code Mapping

	XGMII	XAUI
Idle	/I/ = 0x07	Randomized /A/, /R/, /K/ Sequence /A/ = K28.3 = 0x7C /R/ = K28.0 = 0x1C /K/ = K28.5 = 0xBC (Comma)
Start	/S/ = 0xFB	/S/ = 0xFB
Error	/E/ = 0xFE	/E/ = 0xFE
Terminate	/T/ = 0xFD	/T/ = 0xFD
Ordered Set	/Q/ = 0x9C	/Q/ = 0x9C
Data	Control = 0	Control = 0

The random /A/, /R/, /K/ sequence is generated as specified in section 48.2.5.2.1 of IEEE 802.3-2005 and shown in the state machine diagram given in Figure 2-8. In addition to idle generation, the state machine also forwards sequences of ||Q|| ordered sets used for link status reporting. These sets have the XGMII sequence control character on lane 0 followed by three data characters in XGMII lanes 1 through 3. Sequence ordered-sets are only sent following an ||A|| ordered set.

The random selection of /A/, /K/, and /R/ characters is based on the generation of uniformly distributed random integers derived from a PRBS. Minimum ||A|| code group spacing is determined by the integer value generated by the PRBS (A_cnt in Figure 2-8). ||K|| and ||R|| selection is driven by the value of the least significant bit of the generated integer value (code_sel in Figure 2-8). The idle generation state machine specified in IEEE 802.3-2005 and shown in Figure 2-7 transitions between states based on a 312 MHz system clock. The TX encoder implemented in the XAUI IP runs at a system clock rate of 156 MHz. Thus the XGXS state machine implementation performs the equivalent of two state transitions each clock cycle.

Figure 2-8. XGXS Idle Transmit State Diagram
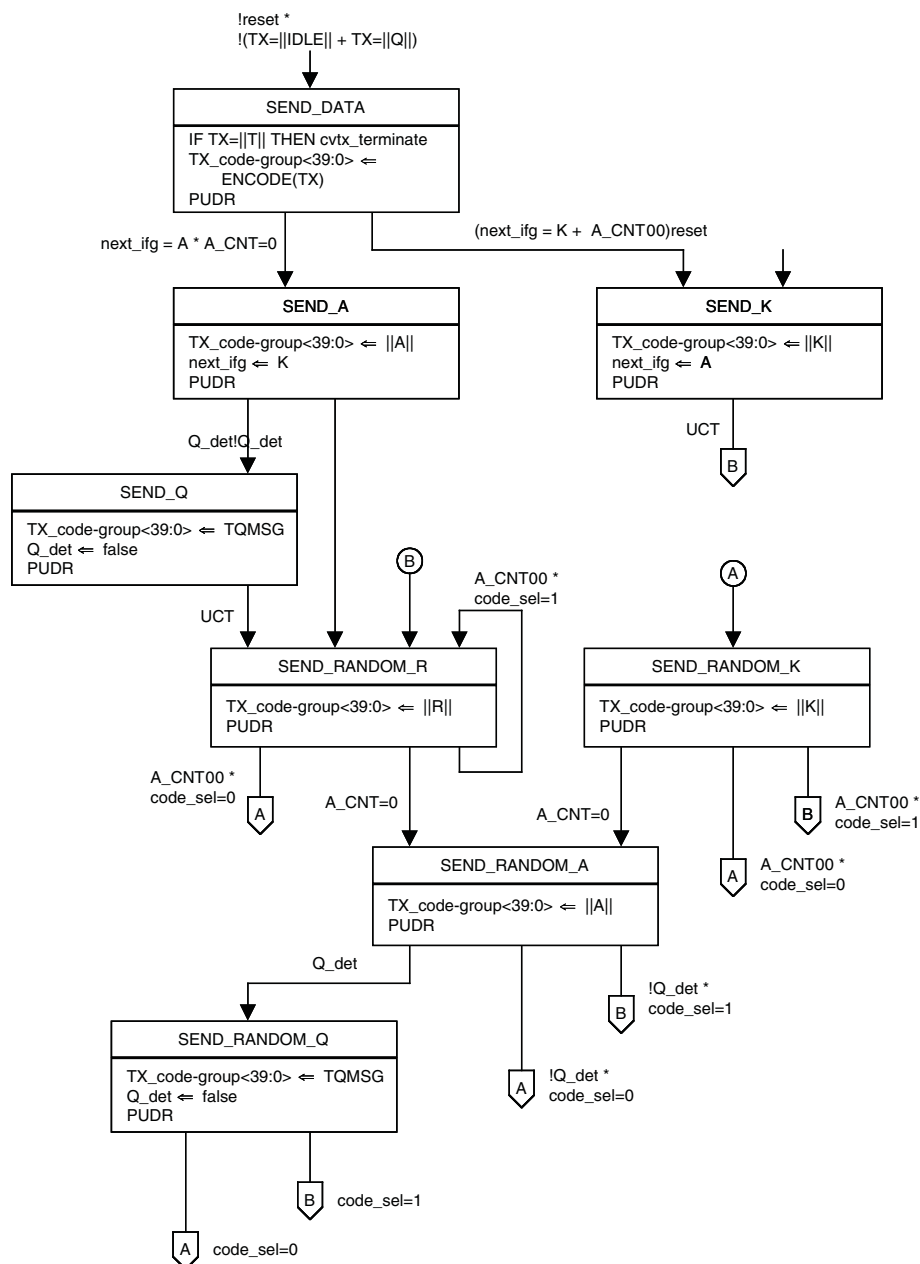
Management Data Input/Output (MDIO) Interface (Optional)

The MDIO interface provides access to the internal XAUI core registers. The register access mechanism corresponds to Clause 45 of IEEE 802.3-2005. The XAUI core provides access to XGXS registers 0x0000-0x0024 as specified in IEEE 802.3-2005. Additional registers in the vendor-specific address space have been allocated for implementation-specific control/status functions.

The physical interface consists of two signals: MDIO to transfer data/address/control to and from the device, and MDC, a clock up to 2.5 MHz sourced externally to provide the synchronization for MDIO. The fields of the MDIO transfer are shown in Figure 2-9.

Figure 2-9. Fields of MDIO Protocol

ST 2b	OP 2b	PRTAD 5b	DTYPE* 5b	TA 2b	ADDRESS/DATA 16b

ST=00

OP	ACCESS TYPE
00	ADDRESS
01	WRITE
10	READ INCREMENT
11	READ

VALUE	DEVICE TYPE
0	RESERVED
1	10G PMA/PMD
2	10G WIS
3	10G PCS
4	10G PHY XGXS
5	10G DTE XGXS
6-15	RESERVED
16-31	VENDOR SPECIFIC

ACCESS TYPE	CONTENTS
ADDRESS	RESERVED
WRITE	10G PMA/PMD
READ INCREMENT	10G WIS
READ	10G PCS
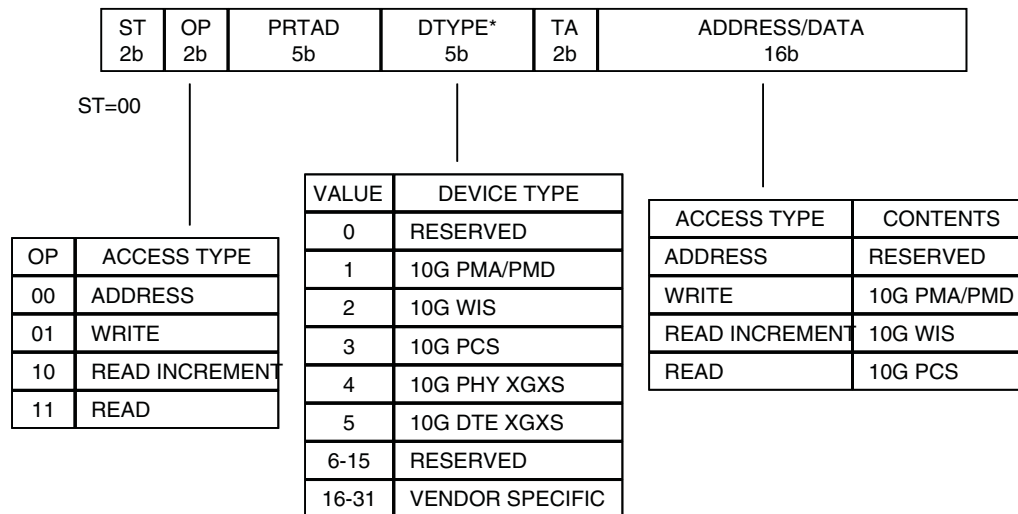
* If ST=01, this field is REGAD (register address).

Management Frame Structure

Each management data frame consists of 64 bits. The first 32 bits are preamble consisting of 32 contiguous 1s on the MDIO. Following the preamble is the start-of-frame field (ST) which is a 00 pattern. The next field is the operation code (OP) that is shown in Figure 2-9.

The next two fields are the port address (PRTAD) and device type (DTYPE). Since the physical layer function in 10 GbE is partitioned into various logical (and possibly separate physical) blocks, two fields are used to access these blocks. The PRTAD provides the overall address to the PHY function. The first port address bit transmitted and received is the MSB of the address. The DTYPE field addresses the specific block within the physical layer function.

Device address zero is reserved to ensure that there is not a long sequence of zeros. If the ST field is 01 then the DTYPE field is replaced with REGAD (register address field of the original clause 22 specification). The XAUI core does not respond to any accesses with ST = 01.

The TA field (Turn Around) is a 2-bit turnaround time spacing between the device address field and the data field to avoid contention during a read transaction. The TA bits are treated as don't cares by the XAUI core.

During a write or address operation, the address/data field transports 16 bits of write data or register address depending on the access type. The register is automatically incremented after a read increment. The address/data field is 16 bits.

For an address cycle, this field contains the address of the register to be accessed on the next cycle. For read/write/increment cycles, the field contains the data for the register. The first bit of data transmitted and received in the address/data field is the MSB (bit 15). An example access is shown in Figure 2-10.
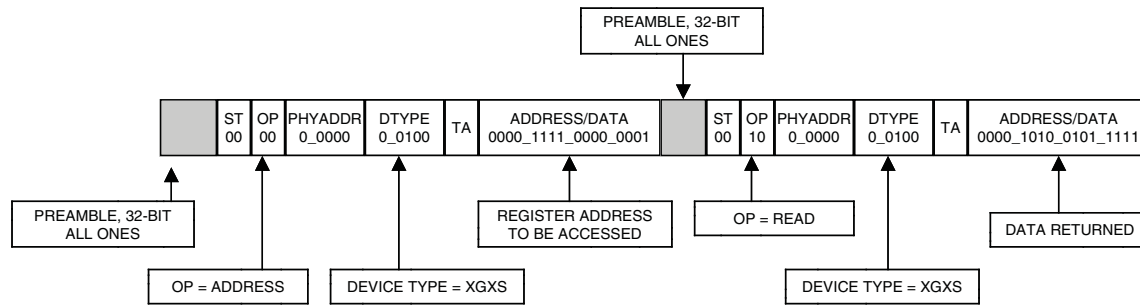
Figure 2-10. Indirect Address Example

Table 2-7 shows PHY XGXS registers as described in IEEE 802.3-2005. The shaded areas are used to indicate register addresses that are specified in the draft but are not used in this implementation.

There are two vendor supported register ranges. The 4.8000h register range is used for accessing and programming the XGXS registers implemented in the programmable array. All corresponding registers are listed in Table 2-8. All PCS embedded core registers can be accessed thru the 4.9xxxh registers shown in Table 2-9, where the address is directly mapped to the PCS embedded registers.

Register Descriptions

Table 2-7. Register Map for XAUI IP Core (Device Address = 4)

Register Address	Register Name
0	PHY XGXS Control 1
1	PHY XGXS Status 1
2, 3	PHY XGXS Identifier
4	Reserved
5	PHY XGXS Status 2
6 - 23	Reserved
24	10G PHY XGXS Lane status
25 - 32767	Reserved
32768 - 65535	Vendor specific

Table 2-8. XAUI IP Core Registers

Bit(s)	Name	Description	R/W	Reset Value
Control 1 Register				
4.0.15	Reset	1 = PHY XA reset, 0 = Normal operation	R/W S/C	0
4.0.14	Loopback (not supported)	Loop back functionality is supported in the PCS core. The XAUI core does not provide loopback capability.	R	0
4.0.13	Speed Selection	Value always 0	R	0
4.0.12	Reserved	Value always 0	R	0
4.0.11	Low Power	0= Low Power Mode 1= Normal operation This bit controls the power_down signal of the XAUI core.	R/W	1
4.0.[10:7]	Reserved	Value always 0	R	0
4.0.6	Speed Selection	Value always 0	R	0
4.0.[5:2]	Speed Selection	Value always 0	R	0
4.0.[1:0]	Reserved	Value always 0	R	0
Status 1 Register				
4.1.[15:8]	Reserved	Value always 0	R	0

Table 2-8. XAUI IP Core Registers (Continued)

Bit(s)	Name	Description	R/W	Reset Value
4.1.7	Fault (not supported)	0 = No Fault condition	R	0
4.1.[6:3]	Reserved	Value always 0	R	0
4.1.2	PHY XS TX link status (not supported)	The Link status is available in the PCS core register.	R	0
4.1.1	Low Power Ability	1 = Low Power Mode support	R	1
4.1.0	Reserved	Value always 0	R	0
XGXS Identifier Registers				
4.2:[15:0]	PHY XS Identifier	MSB = 0x0000	R	0
4.3:[15:0]	PHY XS Identifier	LSB = 0x0004	R	0004
XGXS Reserved Register				
4.4.[15:1]	Reserved	Value always 0	R	0
4.4.0	10 G Capable	Value always 1	R	1
Status 1 Register				
4.5.[15:6]	Reserved	Value always 0	R	0
4.5.5	DTE XS Present	Value always 0	R	0
4.5.4	PHY XS Present	Value always 1	R	1
4.5.3	PCS Present	Value always 0	R	0
4.5.2	WIS Present	Value always 0	R	0
4.5.1	PMD/PMA Present	Value always 0	R	0
4.5.0	Clause 22 regs present	Value always 0	R	0
XGXS Reserved Register				
4.6.15	Vendor specific device present	Value always 0	R	0
4.6.[14:0]	Reserved	Value always 0	R	0
4.8.[15:14]	Device present	10 = Device responding to this address	R	10
4.8.[13:12]	Reserved	Value always 0	R	0
4.8.11	Transmit Fault (not supported)	0 = No fault of tx path	R	0
4.8.10	Receive Fault (not supported)	0 = No fault of tx path	R	0
4.8.9:0	Reserved	Value always 0	R	0
4.15, 4.14	Package Identifier	Value always 0	R	0
4.24.[15:13]	Reserved	Value always 0	R	0
4.24.12	PHY XGXS Lane Alignment (not supported)	TX alignment status is available in the PCS core register	R	0 0
4.24.11	Pattern Testing ability	0 = Not able to generate pattern.	R	0
4.24.10	PHY XGXS has loop back capability	1 = Has loop back capability	R	1
4.24.[9:4]	Reserved	Value always 0	R	0
4.24.3	Lane 3 Sync (not supported)	Status is available from the PCS core	R	0 0
4.24.2	Lane 2 Sync (not supported)	Status is available from the PCS core	R	0 0
4.24.1	Lane 1 Sync (not supported)	Status is available from the PCS core	R	0 0

Table 2-8. XAUI IP Core Registers (Continued)

Bit(s)	Name	Description	R/W	Reset Value
4.24.0	Lane 0 Sync (not supported)	Status is available from the PCS core	R	0 0
4.25.15:3	Reserved	Value always 0	R	0
4.25.2	Receive test pattern enable	0 = Receive test pattern not enabled	R	0
4.25.1:0	Test pattern select	00	R	00
4.8000.[15:0]	MCA Sync Request	Writing any value to this register triggers the MCA sync request. This register always read as 0.	R/W	0
4.8001.15	MCA Sync Status	This bit provides MCA synchronization status.	R/W	0
4.8002.[15:8]	Unused	Bit [15:8] are unused	R/W	0
4.8002.[7:0]	GPO	This field controls the General Purpose Outputs (GPO) when the MDIO is enabled. It is intended to control optional ports of the PCS core.	R/W	0
4.8003.[15:8]	Unused	Bit [15:8] are unused	R/W	0
4.8003.[7:0]	GPI	This field senses the state of the General Purpose Inputs (GPI) when the MDIO is enabled. It is intended to sense the status control of the PCS core.	R/W	0

Table 2-9. XAUI Vendor Specific Registers 4.9xxxh

| Bit(s)	Name	Description	R/W	Reset Value
4.9xxxx.[15:8]	Unused	Bit [15:8] are unused	R/W	0
4.9xxxx.[7:0]	PCS register access	Bit [7:0] are directly mapped to sci_data port. Address [5:0] are directly mapped to sci_address port [5:0]. Address bit [8:6] are used to decode which SCI quad is being selected. [8:6] = 0 SCI0 is selected [8:6] = 1 SCI1 is selected [8:6] = 2 SCI2 is selected [8:6] = 3 SCI3 is selected [8:6] = 4 SCI AUX is selected	R/W	0

Input/Output Timing

XGMII Specifications

Clause 46 if IEEE 802.3-2005 specifies HSTL1 I/O with a 1.5V output buffer supply voltage for all XGMII signals. The HSTL1 specifications comply with EIA/JEDEC standard EIA/JESD8-6 using Class I output buffers with output impedance greater than 38¾ to ensure acceptable overshoot and undershoot performance in an unterminated interconnection. The parametric values for HSTL XGII signals are given in Table 2-10. The HSTL termination scheme is shown in Figure 2-11. Timing requirements for chip-to-chip XGMII signals are shown in Figure 2-12.

Table 2-10. XGMII DC and AC Specifications

Parameter	Condition	Min.	Typ.	Max.	Units
VDDIO	-	1.4	1.5	1.8	V
VREF	-	0.68	0.75	0.9	V
VTT1	-	-	0.75	-	V
VIH	-	VREF + 100mV	0.85	VDDIO + 0.3	V
VIL	-	-0.3	0.65	VREF - 100mV	V
VOH2	IOH > 8mA	1	1.1	-	V
VOL	IOL > -8mA	-	-	0.4	V

Figure 2-11. HSTL1 Circuit Topology
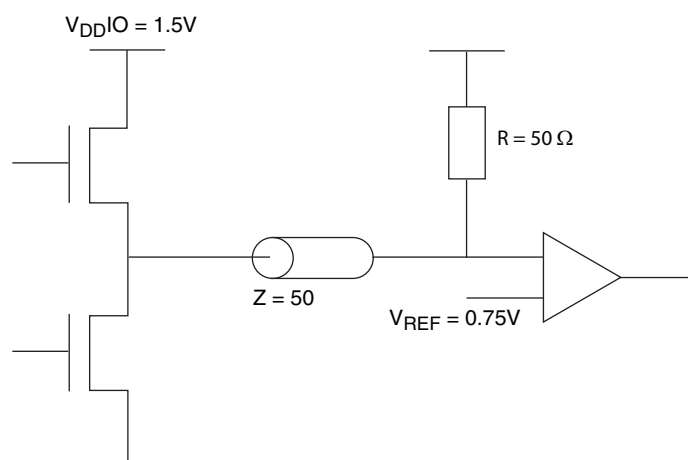
Figure 2-12. XGMII Timing Parameters

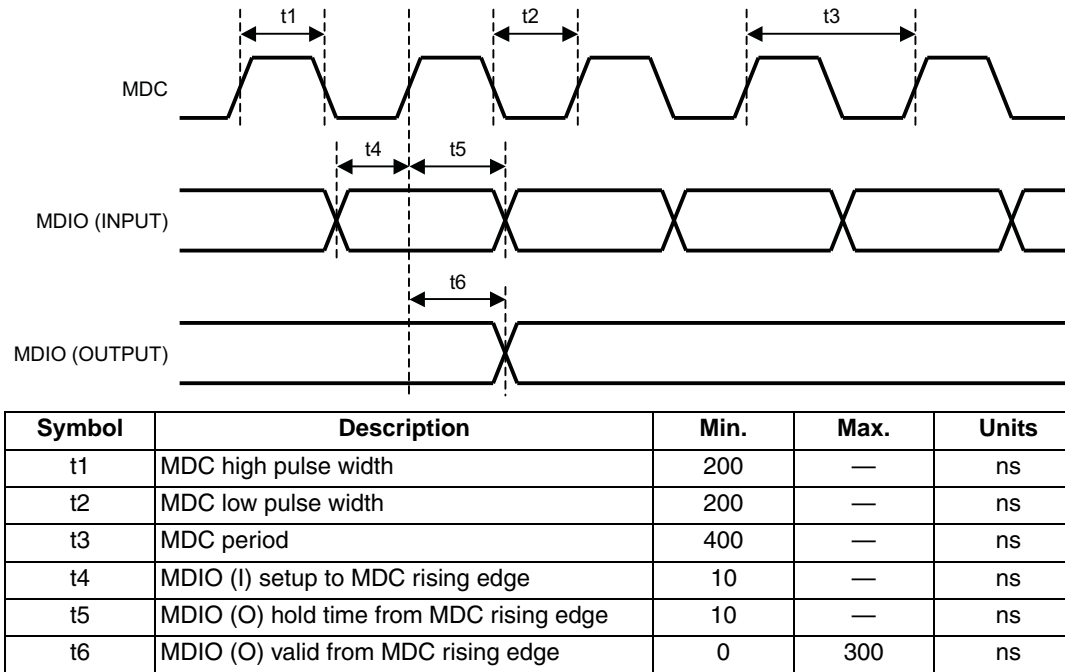| Symbol	Driver	Receiver	Units
t_{SETUP}	960	480	ps
t_{HOLD}	960	480	ps
t_{PWMIN}	2.5	-	ns
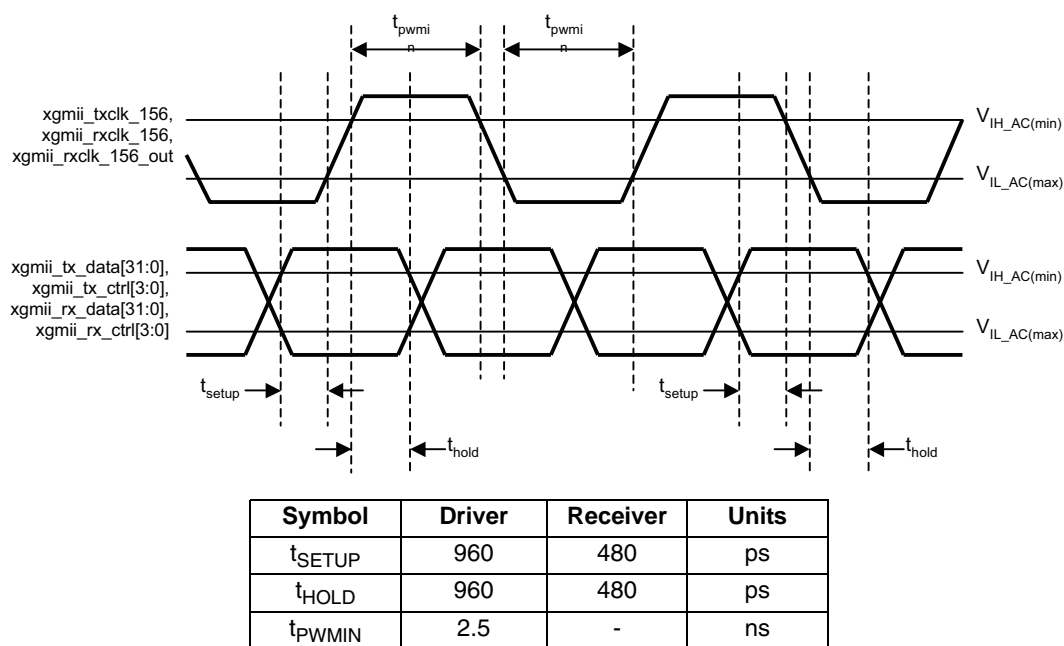
XAUI Specifications

Refer to the LatticeECP2M and LatticeECP3 PCS specifications for a complete XAUI timing requirements.

MDIO Specifications

The electrical specifications of the MDIO signals conform to Clause 45.4 of IEEE 802.3-2005.

Figure 2-13. MDIO Timing

Symbol	Description	Min.	Max.	Units
t1	MDC high pulse width	200	—	ns
t2	MDC low pulse width	200	—	ns
t3	MDC period	400	—	ns
t4	MDIO (I) setup to MDC rising edge	10	—	ns
t5	MDIO (O) hold time from MDC rising edge	10	—	ns
t6	MDIO (O) valid from MDC rising edge	0	300	ns

Parameter Settings

The IPexpress™ tool is used to create IP and architectural modules in the Diamond and ispLEVER software. Refer to "IP Core Generation" on page 27 for a description on how to generate the IP.

Table 3-1 provides the list of user configurable parameters for the XAUI IP core. The parameter settings are specified using the XAUI IP core Configuration GUI in IPexpress.

Table 3-1. XAUI IP Configuration Parameters

Parameter	Range/Options	Default
Configuration Options	Tx Slip Buffer/Rx Slip Buffer/MDIO	Tx Slip Buffer/ Rx Slip Buffer
Generation Options	Behavioral Model/Netlist [.ngo]/ Evaluation Directory	Behavioral Model/Netlist [.ngo]/ Evaluation Directory
Support Synplify, Support Precision, Support ModelSim, Support ALDEC	Enable, Disable	Enable

XAUI IP Configuration Dialog Box

Figure 3-1 shows the XAUI IP core configuration dialog box.

Figure 3-1. XAUI IP Configuration Dialog Box
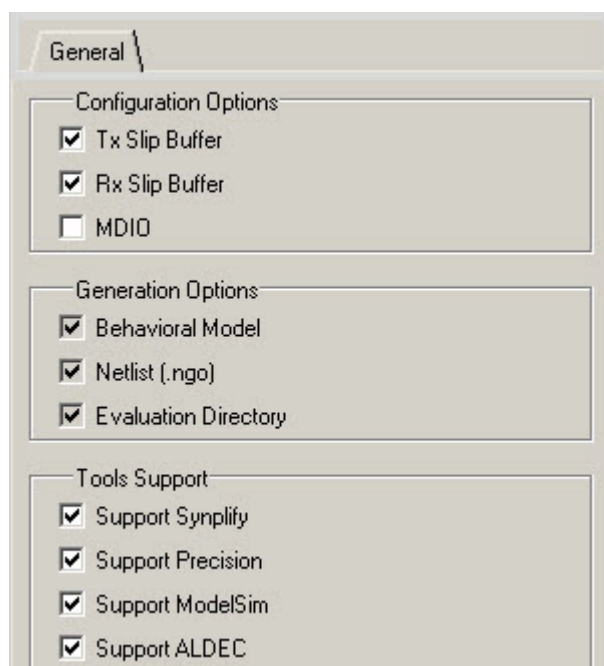
Parameter Descriptions

This section describes the available parameters for the XAUI IP core.

Tx Slip Buffer

This option allows the user to include a slip buffer in the XAUI transmit direction for clock tolerance compensation (enabled by default).

Rx Slip Buffer

This option allows the user to include a slip buffer in the XAUI receive direction for clock tolerance compensation (enabled by default).

MDIO

This option allows the user to include an MDIO interface providing access to XAUI IP core internal registers (disabled by default).

Behavioral Model

This option creates behavioral simulation model for IP core (always enabled).

Netlist [.ngo]

This option creates a netlist (.ngo file) for IP core (always enabled).

Evaluation Directory

This option creates an evaluation directory for IP core that includes simulation and implementation example projects (always enabled).

Tools Support

The XAUI IP core evaluation capability supports multiple synthesis and simulation tool flows. These options allow the user to select desired tool support.

IP Core Generation

This chapter provides information on licensing the XAUI IP core, generating the core using the Diamond or isp-LEVER software IPexpress tool, running functional simulation, and including the core in a top-level design. The Lattice XAUI IP core can be used in LatticeECP3 and LatticeECP2M device families.

Licensing the IP Core

An IP license is required to enable full, unrestricted use of the XAUI IP core in a complete, top-level design. An IP license that specifies the IP core (XAUI) and device family (ECP3 or ECP2M) is required to enable full use of the XAUI IP core in LatticeECP2M or LatticeECP3. Instructions on how to obtain licenses for Lattice IP cores are given at:

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

Users may download and generate the XAUI IP core for Lattice ECP3 or LatticeECP2M and fully evaluate the core through functional simulation and implementation (synthesis, map, place and route) without an IP license. The XAUI IP core 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 32 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 XAUI IP core is available for download from the Lattice IP Server using the IPexpress tool in Diamond or isp-LEVER. The IP files are automatically installed using ispUPDATE technology in any customer-specified directory. After the IP core has been installed, it will be available in the IPexpress GUI dialog box shown in Figure 4-1.

The IPexpress tool GUI dialog box for the XAUI 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 loaded.

- **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. LatticeECP2M, 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)
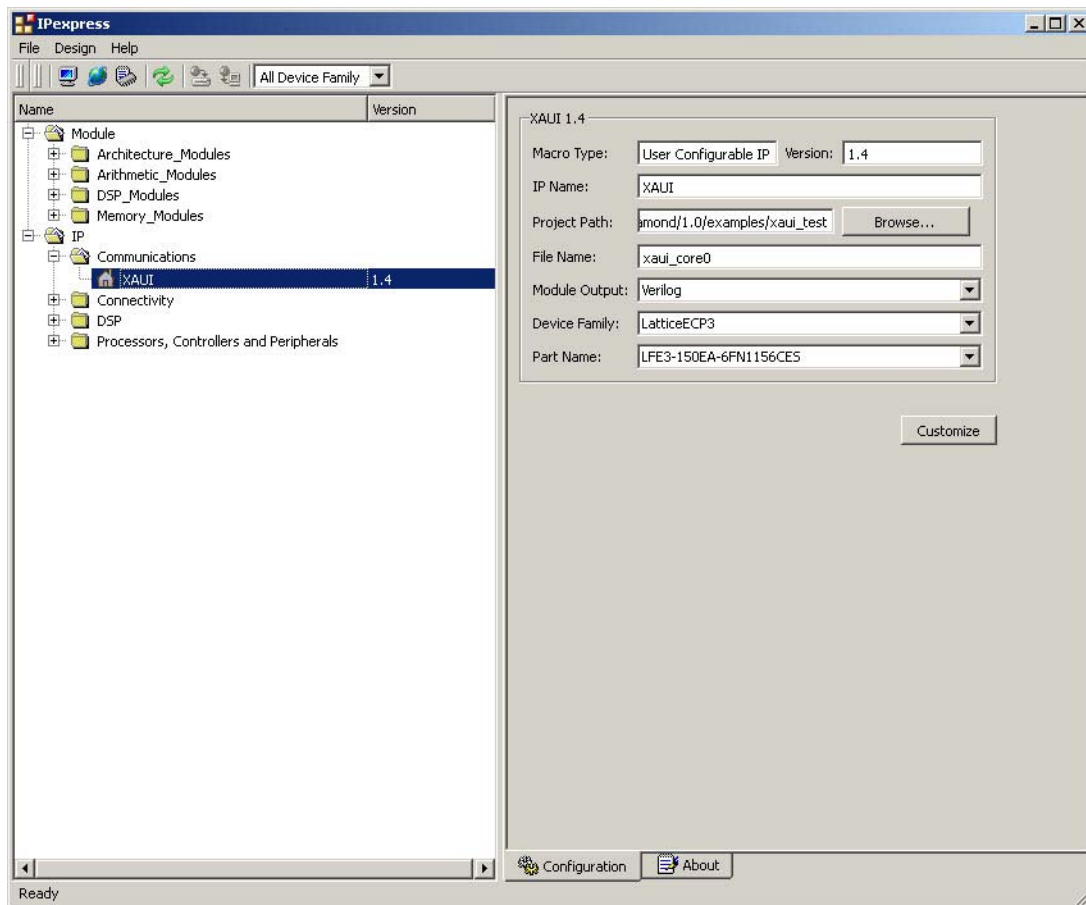
Note that if the IPexpress tool is called from within an existing project, Project Path, Module Output (Design Entry in ispLEVER), Device Family and Part Name default to the specified project parameters. Refer to the IPexpress tool online help for further information.

To create a custom configuration, the user clicks the **Customize** button in the IPexpress tool dialog box to display the XAUI SDRAM 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 25 for more information on the XAUI parameter settings.

Figure 4-2. Configuration GUI (Diamond Version)
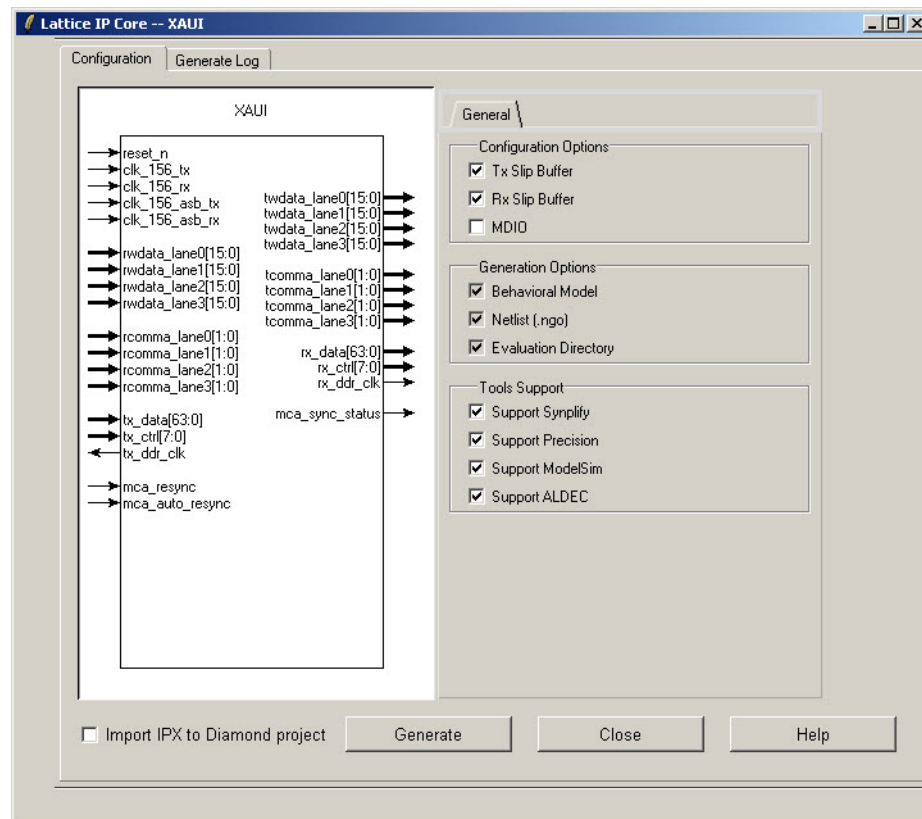
IPexpress-Created Files and Top Level Directory Structure

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.

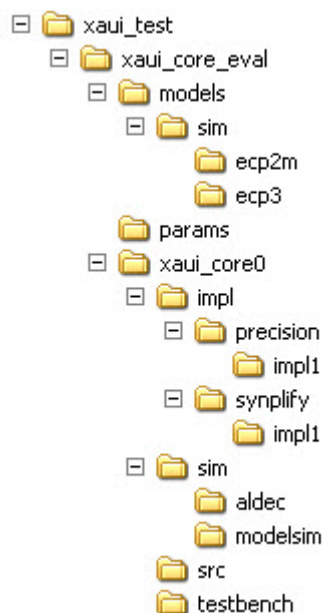*Figure 4-3. XAUI Core Directory Structure*

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 specified in the IPexpress tool.

Table 4-1. File List

File	Description
<username>.lpc	This file contains the IPexpress tool 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.
<username>.ngo	This file provides the synthesized IP core.
*<username>*_bb.v/.vhd	This file provides the synthesis black box for the user's synthesis.
*<username>*_inst.v/.vhd	This file provides an instance template for the PCI IP core.
*<username>*_beh.v/.vhd	This file provides the front-end simulation library for the PCI IP core.

Table 4-2 provides a list of key additional files providing IP core generation status information and command line generation capability are generated in the user's project directory.

Table 4-2. Additional Files

File	Description
*<username>*_generate.tcl	This file is created when the GUI "Generate" button is pushed. This file may be run from command line.
*<username>*_generate.log	This is the synthesis and map log file.
*<username>*_gen.log	This is the IPexpress IP generation log file

The *\xaui_core_eval* and subtending directories provide files supporting XAUI IP core evaluation. The *\models* directory shown in Figure 4-3 contains files/folders with content that is constant for all configurations of the XAUI IP core. The `\<username>` subfolder (`\XAUI_core0` in this example) contains files/folders with content specific to the username configuration.

The `\xaui_core_eval` directory is created by IPexpress the first time the core is generated and updated each time the core is regenerated. A `\<username>` directory is created by IPexpress 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. `\xaui_core0`, `\xaui_core1`, etc.

Instantiating the Core

The generated XAUI IP core package includes black-box (*<username>*_bb.v) and instance (*<username>*_inst.v) templates that can be used to instantiate the core in a top-level design. Two example RTL top-level reference source files are provided in `\<project_dir>\xaui_core_eval\<username>\src`.

The top-level file *<username>*_eval.v is the same top-level file that is used in the simulation model described in the next section. Designers may use this top-level reference as a template for the top level of their design. Included in *<username>*_eval.v sample packet generation and checking capabilities at the XGMII interface.

The top-level file *<username>*_only_top.v supports the ability to implement just the XAUI IP core itself. This design is intended only to provide an accurate indication of the device utilization associated with the XAUI IP core and should not be used as an actual implementation example.

To instantiate this IP core using the Linux platform, users must manually add one environment variable named "SYNPLIFY" to indicate the installation path of the OEM Synplify tool in the local environment file. For example, `SYNPLIFY=/tools/local/isptools7_2/isptools/synplify_linux`.

Running Functional Simulation

The functional simulation includes a configuration-specific behavioral model of the XAUI IP core (*<username>*_beh.v) that is instantiated in an FPGA top level along with user-side (XGMII) packet generation and checking capabilities. The top-level file supporting ModelSim simulation is provided in `\<project_dir>\xaui_core_eval\<username>\src`. This FPGA top is instantiated in an evaluation test-bench provided in `\<project_dir>\xaui_core_eval\<username>\testbench`.

Users may run the evaluation simulation by doing the following:

1. Open ModelSim.

2. Under the **File** tab, select **Change Directory** and choose folder
   `<project_dir>\xaui_core_eval\<username>\sim\modelsim`.

3. Under the **Tools** tab, select **Execute Macro** and execute one of the ModelSim "do" scripts shown.

The simulation waveform results will be displayed in the ModelSim Wave window.

The top-level file supporting Aldec Active-HDL simulation is provided in `\<project_dir>\xaui_core\<username>\sim\aldec`. This is the same top-level that is used for ModelSim simulation.

Users may run the Aldec evaluation simulation by doing the following:

1. Open Active-HDL.

2. Under the console tab change the directory to
   `\<project_dir>\xaui_core_eval\<username>\sim\aldec`

3. Execute the Active-HDL "do" scripts shown.

The simulation waveform results will be displayed in the Active-HDL Wave window.

Synthesizing and Implementing the Core in a Top-Level Design

The XAUI IP core itself is synthesized and is provided in NGO format when the core is generated. Users may synthesize the core in their own top-level design by instantiating the core in their top level as described previously and then synthesizing the entire design with either Synplify or Precision RTL Synthesis (Synopsys Synplify is included with Diamond and ispLEVER. Precision RTL Synthesis is available separately from Mentor Graphics).

Two example RTL top-level configurations supporting the XAUI IP core top-level synthesis and implementation are provided with the XAUI IP core in `\<project_dir>\xaui_core_eval\<username>\impl`.

The top-level file xaui_core_eval_only_top.v provided in `\<project_dir>\eval\<username>\src\rtl\top` supports the ability to implement just the XAUI IP core. This design is intended only to provide an accurate indication of the device utilization associated with the core itself and should not be used as an actual implementation example.

The top-level file *<username>*_eval.v provided in `\<project_dir>\eval\<username>\src` supports the ability to instantiate, simulate, map, place and route the XAUI IP core in an example design that include packet generation and checking modules at the XGMII inter-face. This is the same configuration that is used in the evaluation simulation capability described previously.

Push-button implementation of both top-level configurations is supported via the Diamond or ispLEVER project files, *<username>*_core_only_eval.syn and *<username>*_eval.syn. These files are located in `\<project_dir>\xaui_core_eval\<username>\impl\(synplify or precision)`, for implementation using either the Synplify synthesis tool or Precision synthesis tool, respectively.

For the Linux platform, the top-level synthesis must be run separately with a synthesis tool such as Synplify Pro, since Diamond and ispLEVER for Linux only accept an EDIF entry. For the core-only project, the synthesis tcl file *<username>*_core_only.tcl is generated in
`\<project_dir>\xaui_core_eval\<username>\impl\synplify`. The user can run this tcl script to synthesize the core_only top-level files in the above directory.

For the reference project, *<username>*_eval.tcl will be generated in the
`\<project_dir>\xaui_core_eval\<username>\impl\synplify` directory. The user can run this tcl script to synthesize the reference top_level files in the above directory.

To use this project file in Diamond:

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

2. Browse to
   `\<project_dir>\xaui_core_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>\xaui_core_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 XAUI 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.

Support Resources

This chapter contains information about Lattice Technical Support, additional references, and document revision history.

Lattice Technical Support

There are a number of ways to receive technical support.

Online Forums

The first place to look is Lattice Forums (http://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 & 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

References

- IEEE Std. 802.3-2005, *Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, December 9, 2005.
- TN1084, *LatticeSC SERDES Jitter*

LatticeECP3

- HB1009, *LatticeECP3 Family Handbook*

LatticeECP2M

- HB1003, *LatticeECP2M Family Handbook*

Revision History

Date	Document Version	IP Core Version	Change Summary
June 2007	01.0	1.0	Initial release.
June 2008	01.1	1.1	Title changed from "10Gb Ethernet Attachment Unit Interface (XAUI) IP Core User's Guide" to "XAUI IP Core User's Guide".
			Updated Appendix for LatticeECP2M FPGAs.
June 2009	01.2	1.2	Added support for LatticeECP3 FPGA family.
November 2009	01.3	1.3	Updated utilization tables with ispLEVER 8.0.
June 2010	01.4	1.3	Divided document into chapters. Added table of contents.
			Added Quick Facts table in Chapter 1, "Introduction."
			Added new content in Chapter 4, "IP Core Generation."
September 2010	01.5	1.4	Added support for Diamond software throughout.
January 2012	01.6	1.6	IP Core I/O and Signal Definitions table – Added lines for rx_fifo_empty and rx_fifo_full to the With Optional Rx Slip Buffer section. Added lines for tx_fifo_empty and tx_fifo_full to the With Optional Tx Slip Buffer section.

Resource Utilization

This appendix gives resource utilization information for Lattice FPGAs using the XAUI IP core.

IPexpress is the Lattice IP configuration utility, and is included as a standard feature of the Diamond and ispLEVER design tools. Details regarding the usage of IPexpress can be found in the IPexpress and Diamond or ispLEVER help system. For more information on the Diamond or ispLEVER design tools, visit the Lattice web site at: www.latticesemi.com/software.

LatticeECP2M FPGAs

Table A-1. Performance and Resource Utilization[1]

	Configuration			Utilization			
Device	**TX Slip Buffer**	**RX Slip Buffer**	**MDIO**	**SLICEs**	**LUTs**	**Registers**	**EBRs**
LFE2M20E-6F256CES	No	No	No	1351	1813	1545	0
LFE2M20E-6F256CES	No	No	Yes	1447	1951	1685	0
LFE2M35E-6F484CES	No	Yes	No	1632	2137	1989	2
LFE2M50E-6F672CES	Yes	No	No	1727	2264	2033	2
LFE2M70E-6F900CES	Yes	Yes	Yes	2160	2801	2641	4

1. Performance and utilization data are generated using Lattice Diamond 1.0 and Synplify Pro for Lattice D-2009.12L software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeECP2M family.

Ordering Part Number

The Ordering Part Number (OPN) for all configurations of the XAUI IP core targeting LatticeECP2M devices is XAUI-PM-U1.

Jitter and XAUI Compliance

The Lattice XAUI IP core for the LatticeECP2M device family offers compatibility with XAUI solutions and is functionally compliant to XAUI standard specification IEEE 802.3-2005. XAUI specification limits total jitter to 112ps (0.35UI at 320 picoseconds/Unit Interval). LatticeECP2M exhibits transmit jitter within XAUI specification under typical conditions. Worst case transmit jitter considered over all temperature, voltage and process corners may fall outside XAUI specification. However, the LatticeECP2M based XAUI solution has been found to be compatible with many systems. See TN1084, *LatticeSC SERDES Jitter*, for applicable jitter specification.

XAUI specification requires the receive side to have the ability to operate with 208ps (0.65UI) of jitter. The Lattice solution for XAUI receive operates well within that specification.

LatticeECP3 FPGAs

Table A-2. Performance and Resource Utilization[1]

	Configuration			Utilization			
Device	**TX Slip Buffer**	**RX Slip Buffer**	**MDIO**	**SLICEs**	**LUTs**	**Registers**	**EBRs**
LFE3-35E-7FN484CES	No	No	No	1194	1674	1498	0
LFE3-70E-7FN672CES	No	Yes	No	1529	2077	1980	2
LFE3-150E-7 FN1156CES	Yes	Yes	No	1880	2510	2467	4

1. Performance and utilization data are generated using Lattice Diamond 1.0 and Synplify Pro for Lattice D-2009.12L software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeECP3 family.

Ordering Part Number

The Ordering Part Number (OPN) for the XAUI IP core targeting LatticeECP3 devices is XAUI-E3-U1.

Данный компонент на территории Российской Федерации

Вы можете приобрести в компании MosChip.

Для оперативного оформления запроса Вам необходимо перейти по данной ссылке:

http://moschip.ru/get-element

   Вы можете разместить у нас заказ для любого Вашего проекта, будь то серийное   производство или разработка единичного прибора.

В нашем ассортименте представлены ведущие мировые производители активных и пассивных электронных компонентов.

Нашей специализацией является поставка электронной компонентной базы двойного назначения, продукции таких производителей как XILINX, Intel (ex.ALTERA), Vicor, Microchip, Texas Instruments, Analog Devices, Mini-Circuits, Amphenol, Glenair.

Сотрудничество с глобальными дистрибьюторами электронных компонентов, предоставляет возможность заказывать и получать с международных складов практически любой перечень компонентов в оптимальные для Вас сроки.

На всех этапах разработки и производства наши партнеры могут получить квалифицированную поддержку опытных инженеров.

Система менеджмента качества компании отвечает требованиям в соответствии с ГОСТ Р ИСО 9001, ГОСТ РВ 0015-002 и ЭС РД 009

Офис по работе с юридическими лицами:

105318, г.Москва, ул.Щербаковская д.3, офис 1107, 1118, ДЦ «Щербаковский»

Телефон: +7 495 668-12-70 (многоканальный)

Факс: +7 495 668-12-70 (доб.304)

E-mail: info@moschip.ru

Skype отдела продаж:
moschip.ru   moschip.ru_6
moschip.ru_4   moschip.ru_9