LatticeCORE

SGMII and Gb Ethernet PCS IP Core User's Guide

April 2014
IPUG60_02.1

Table of Contents

Introduction

The Lattice SGMII and Gb Ethernet PCS IP core implements the PCS functions of both the Cisco SGMII and the IEEE 802.3z (1000BaseX) specifications. The PCS mode is pin selectable. This IP core may be used in bridging applications and/or PHY implementations.

The Serial Gigabit Media Independent Interface (SGMII) is a connection bus for Ethernet Media Access Controllers (MACs) and Physical Layer Devices (PHYs) defined by Cisco Systems. It replaces the classic 22-wire GMII connection with a low pin count, 4-pair, differential SGMII connection. The classic GMII interface defined in the IEEE802.3 specification is strictly for Gigabit rate operation. However, the Cisco SGMII specification defines a method for operating 10 Mbps and 100 Mbps MACs over the interface. Moreover, the Cisco SGMII specification is comprised of more than just a bus interface definition; it defines a bridging function between SGMII and GMII buses.

These applications can be completely implemented in ECP5™ and LatticeECP3™ Field Programmable Gate Array (FPGA) devices. As an example, Lattice has developed a reference design for a complete SGMII-to-(G)MII bridge. This reference design is included with the SGMII and Gb Ethernet PCS IP Core package and is described in detail in "SGMII-to-(G)MII Reference Design" on page 33.

This document describes the IP core's operation, and provides instructions for generating the core through Lattice's IPexpress™ tool, and for instantiating, synthesizing, and simulating the core.

Quick Facts

Table 1-1 gives quick facts about the Lattice SGMII and Gb Ethernet PCS core.

Table 1-1. Lattice SGMII and Gb Ethernet PCS IP Core Quick Facts

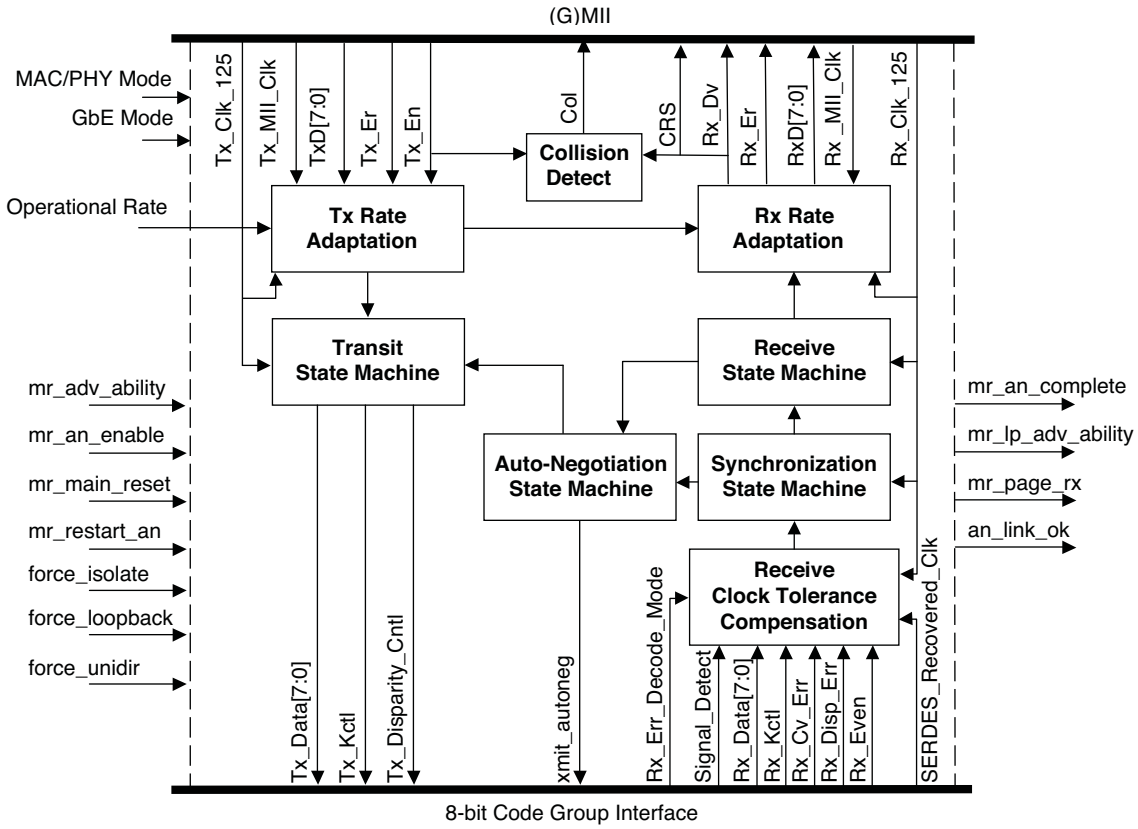
		SGMII and Gb Ethernet IP Configuration	
		Across All IP Configurations	
Core Requirements	FPGA Families Supported	ECP5, LatticeECP3	
	Minimal Device Needed	LFE5UM-25F-6MG285C	LFE3-17E-7FN256C
Resource Utilization	Targeted Devices	LFE5UM-85F-7BG756C	LFE3-70EA-7FN484C
	Data Path Width	8	
	LUTs	800 to 1000	
	sysMEM EBRs	0 to 1	
	Registers	800 to 1000	
Design Tool Support	Lattice Implementation	Lattice Diamond® 3.2	
	Synthesis	Synopsys® Synplify Pro® for Lattice I-2013.09L-SP1	
		Mentor Graphics® Precision® RTL	
	Simulation	Aldec® Active-HDL™ 9.3 Lattice Edition	
		Mentor Graphics® ModelSim® SE (Verilog only)	

Features

- Implements PCS functions of the Cisco SGMII Specification, Revision 1.8
- Implements PCS functions for IEEE 802.3z (1000BaseX)
- Dynamically selects SGMII/1000BaseX PCS operation
- Supports MAC or PHY mode for SGMII auto-negotiation
- Supports (G)MII data rates of 1G bps, 100 Mbps, 10 Mbps
- Provides Management Interface Port for control and maintenance
- Includes Easy Connect option for seamless integration with Lattice's Tri-Speed MAC (TSMAC) IP core

The SGMII and Gb Ethernet PCS IP core converts GMII frames into 8-bit code groups in both transmit and receive directions and performs auto-negotiation with a link partner as described in the Cisco SGMII and IEEE 802.3z specifications. The IP core's block diagram is shown in Figure 2-1.

Figure 2-1. SGMII and Gb Ethernet PCS IP Core Block Diagram



An example of how this core may be used in implementing a complete SGMII-to-(G)MII bridge is shown in Figure 2-2.

Figure 2-2. SGMII to (G)MII Bridge Reference Design

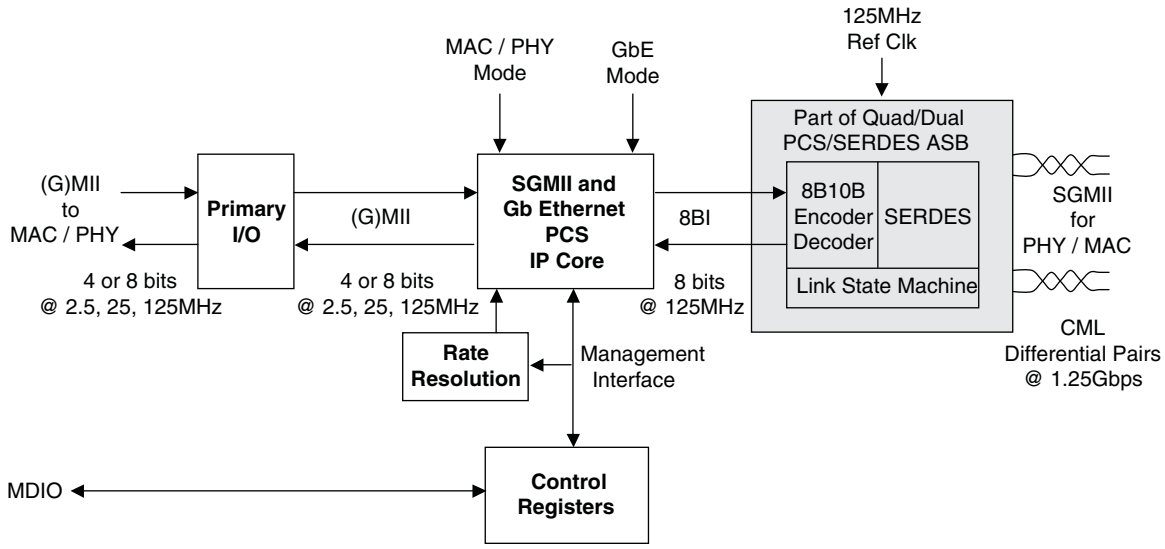
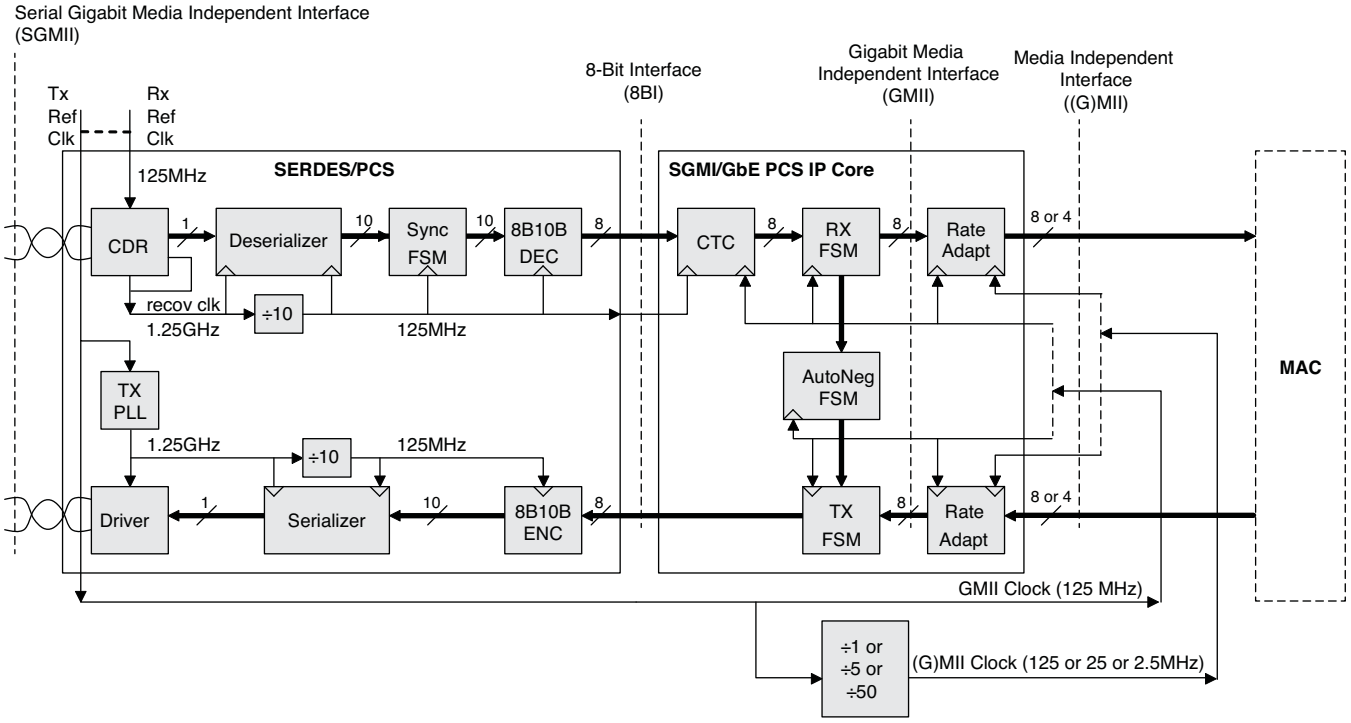


Figure 2-3 shows typical clock network connections between the IP core, the SERDES, and a MAC.

Figure 2-3. Typical SGMII Clocking Network



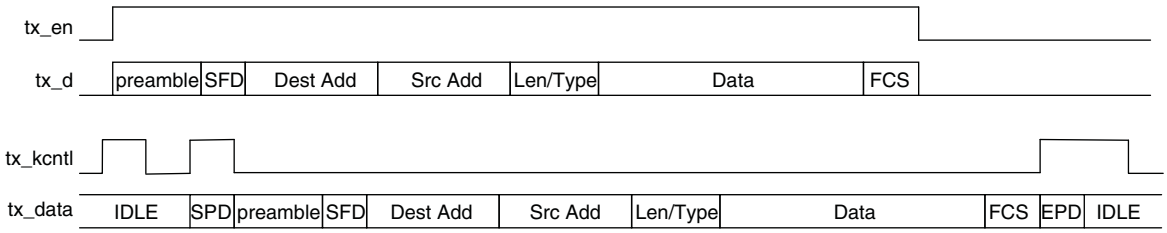
Transmit Rate Adaptation

This block adjusts the byte-per-byte data rate such that the output rate is always 1Gbps. For the case where the incoming (G)MII operates at 1Gbps, there is no data rate alteration. The incoming data is 8 bits wide clocked at 125 MHz and the outgoing data is also 8 bits wide clocked at 125 MHz. For the case where the incoming (G)MII operates at 100 Mbps, each incoming data byte is replicated ten times on the outgoing port. The incoming data is 4 bits wide clocked at 25 MHz clock and the outgoing data is 8 bits wide clocked at 125 MHz. The case for 10 Mbps (G)MII operation is similar except that data bytes are replicated 100 times and the incoming clock rate is 2.5 MHz.

Transmit State Machine

The transmit state machine implements the transmit functions described in clause 36 of the IEEE802.3 specification. The state machine's main purpose is to convert GMII data frames into code groups. A typical timing diagram for this circuit block is shown in Figure 2-4. Note that the state machine in this IP core does not fully implement the conversion to 10-bit code groups as specified in the IEEE802.3 specification. Instead, partial conversion to 8-bit code groups is performed. A separate encoder in the Lattice SERDES completes the full conversion to 10-bit code groups.

Figure 2-4. Typical Transmit Timing Diagram



Soft Receive Clock Tolerance Compensation (CTC) Circuit

This block allows the receive path to compensate for slight frequency offsets between two clocks with a nominal frequency of 125 MHz. One timing source is the recovered clock from the SERDES Rx physical link. The other timing source is the locally generated Rx clock. As long as the two clock frequencies are within acceptable limits, the compensation circuit can maintain data path integrity. The circuit employs a 1024-byte deep FIFO with programmable high and low watermarks. When the FIFO occupancy is below the low watermark and the datastream is within an inter-packet region, FIFO reading is halted and two idle-2 ordered sets are inserted into the datastream. This action compensates for situations where the local clock is faster than the recovered clock. When the FIFO occupancy is above the high watermark and the datastream is within an inter-packet region, FIFO writing is inhibited for a period of one idle ordered set. This action compensates for situations where the local clock is slower than the recovered clock.

Note that the amount of timing drift between the two CTC clocks is dependent upon the size of the frequency offset and the size of the ethernet frame. Larger frame sizes produce larger timing drifts; larger frequency offsets produce larger timing drifts. For example, a +/- 100 ppm offset for a 1500 byte frame produces a 2.4 ns time drift. Increasing the frame size to 15,000 bytes produces a 24 ns time drift. The characteristics of the system design affect the compensation requirements for the CTC.

To address some of these issues, the IP core has three CTC implementations.

- A dynamic mode.
- A static mode.
- A “none” mode.

The user chooses the desired CTC mode when the IP core is generated using the IPexpress tool.

The dynamic CTC mode is intended for use where the IP core is expected to utilize all three of the possible (G)MII data rates (1 Gbps, 100 Mbps, 10 Mbps). The dynamic mode CTC automatically changes its FIFO thresholds to match the (G)MII data rate. These thresholds are hard-coded, and cannot be changed by the user. The thresholds are 16/32 for 1 Gbps, 34/68 for 100 Mbps, 340/640 for 10 Mbps. These settings provide the following compensation performance for a 9600 byte frame: +/- 729 ppm for 1Gbps; +/- 166 ppm for 100 Mbps; and +/- 176 ppm for 10 Mbps.

The static CTC mode is intended for use where the IP core is expected to operate at only one (G)MII data rate (1 Gbps, or 100 Mbps, or 10 Mbps). The user is able to choose the FIFO thresholds when the IP core is generated using the IPexpress tool. The default FIFO thresholds are 16/32, providing +/- 729 ppm compensation for a 9600 byte frame at (G)MII data rate of 1 Gbps. However, the user can alter these thresholds to provide other compensation capabilities.

The “none” CTC mode is intended for cases where a CTC function within the IP core is unwanted. The “none” mode utilizes a 16-byte deep FIFO to provide clean data transfers between the RX recovered clock, and the local RX clock. These two clocks may have arbitrary phase relationships with respect to each other; however the clocks must be at exactly the same frequency (0 ppm offset).

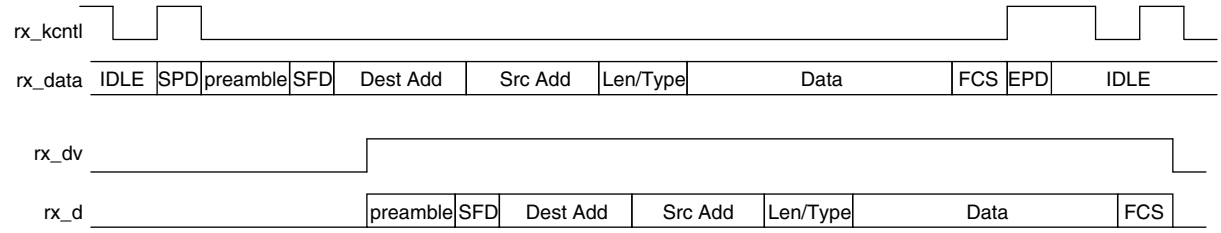
Synchronization State Machine

The synchronization state machine implements the alignment functions described in clause 36 of the IEEE802.3 specification. The state machine's main purpose is to determine whether incoming code groups are properly aligned. Once alignment is attained, proper code groups are passed to the receive state machine. If alignment is lost for an extended period, an auto-negotiation restart is triggered.

Receive State Machine

The receive state machine implements the receive functions described in clause 36 of the IEEE802.3 specification. The state machine's main purpose is to convert code groups into GMII data frames. A typical timing diagram for this circuit block is shown in Figure 2-5. Note that the state machine in this IP core does not fully implement the conversion from 10-bit code groups as specified in the IEEE802.3 specification. Instead, partial conversion from 8-bit code groups is performed. A separate decoder in the Lattice SERDES performs 10-bit to 8-bit code group conversions.

Figure 2-5. Typical Receive Timing Diagram



Receive Rate Adaptation

This block's function is similar to the transmit rate adaptation block, except that it operates in reverse. The incoming data rate is always 1Gbps. The outgoing data rate is reduced by factors of 1x, 10x, or 100x for (G)MII rates of 1Gbps, 100 Mbps, and 10 Mbps respectively.

Auto-Negotiation State Machine

The auto-negotiation state machine implements the link configuration functions described in clause 37 of IEEE802.3 specification. However, the Cisco SGMII specification defines several changes (summarized below). This IP core will operate in adherence to either specification, based on the setting of an external “mode” pin. Please consult the two specifications for a detailed description of auto-negotiation operation. From a high-level view, the primary auto-negotiation functions are: testing the physical link for proper operation, and passing link configuration information between entities sitting on both sides of the link.

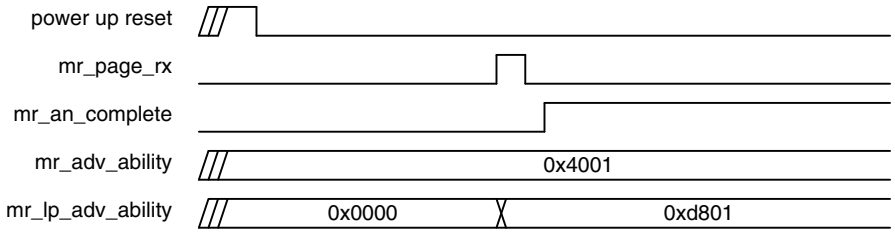
The major Cisco SGMII auto-negotiation modifications include:

- Decreases link timer interval from 10 msec to 1.6 msec.
- Redefines “link ability” bit assignments (see Table 2-1).
- Eliminates the need to pass link ability information from MAC to PHY.
- Adds a new condition that forces a restart on the PHY side whenever the PHY link abilities change.

A rough flow of auto-negotiation operation is as follows:

- An event triggers the process (e.g. reset, loss of sync).
- Both link entities recognize that the process has started and begin transmitting their abilities.
- The link timer expires.
- Both link entities attempt to verify reception of link partner abilities. Once reception is achieved, the entity transmits an acknowledge signal.
- The link timer expires.
- Both link entities verify receipt of link partner acknowledgement and then begin transmitting IDLE code groups.
- The link timer expires.
- Both link entities verify reception of IDLE code groups from link partner. Then the process enters the LINK-OK state and normal PCS operation can begin. The machine will remain in the LINK-OK state until a re-start event is detected.

Figure 2-6. Typical Auto-Negotiation Timing Diagram for MAC-Side Entity



IP core signal ports associated with the auto-negotiation function are listed in Table 2-1.

Table 2-1. IP Core Signal Ports Associated with Auto-Negotiation

	Bit Number	SGMII PHY Mode	SGMII MAC Mode	GbE Mode
mr_adv_ability[15:0]	15	Link Status	0	Next Page
	14	Acknowledge (controlled by autoneg state machine)	1	Acknowledge (controlled by autoneg state machine)
	13	0	0	Remote Fault[1]
	12	Duplex Mode 1=full; 0=half	0	Remote Fault[0]
	11	(G)MII Speed[1] 11=rsvd; 10=1Gbps	0	0
	10	(G)MII Speed[0] 01=100Mbps; 00=10Mbps	0	0
	9	0	0	0
	8	0	0	Pause[1]
	7	0	0	Pause[0]
	6	0	0	Half Duplex
	5	0	0	Full Duplex
	4	0	0	0
	3	0	0	0
	2	0	0	0
	1	0	0	0
	0	1	1	0
mr_lp_adv_ability[15:0]	Displays ability data received by entity (from link partner)			
mr_an_enable	Active high signal; enables auto-negotiation operation			
mr_restart_an	Active high signal; forces restart of auto-negotiation			
mr_page_rx	Active high signal, asserts while state machine is in “Complete Acknowledge” state			
mr_an_complete	Active high signal, asserts while state machine is in “Link OK” state			
sgmii_mode	Controls auto-negotiation behavior. 0 = MAC-side operation; 1 = PHY-side operation			
debug_link_timer_short	Controls duration of link interval timer. 0 = Timer interval is normal (1.6 msec/SGMII 10msec/1000BaseX) 1 = Timer interval is 2 µsec (debug). This signal should be tied to logic-0 when synthesizing the IP core so that the timer operates normally. However, for simulation, the signal can be tied to logic high to speed up completion of the auto-negotiation portion of the simulation.			

Collision Detect

The collision detect signal is the logical-and of the tx_en and rx_dv GMII signals.

Carrier Sense

The carrier sense signal is identical to the rx_dv GMII signal.

Data Path Latency Specifications

Latency is the time delay taken to propagate signals between two points in the data path. This section provides latency specifications for the transmit and receive data paths of the SGMII IP core. The latency values are found by measuring the delay time between the leading edge of the SFD symbol of an ethernet frame, as it passes through the data path. For example, to specify the latency of the RX-CTC circuit, a continuous burst of constant-width ethernet frames, with constant 12-clock-cycle inter frame gaps are applied to the head-end of the RX data path. Then the latency is found by measuring the difference between the leading edge of the SFD symbol into and out-of the CTC circuit. The latency specifications are given in units of clock-cycles (125 MHz clock). All latency specifications are provided for the 1Gbps (G)MII data rate.

Referring back to Figure 2-3, it is shown that the two major elements in the IP core's transmit data path include the transmit-rate-adaptation-block and the transmit-state-machine-block. Both blocks provide constant delays, regardless of ethernet frame size, clock accuracy, and inter frame gap. There is no latency variation due to these factors. Table 2-2 shows the transmit data path latencies for the IP core's two (G)MII configurations – classic style and TSMAC easy connect style. Other IP core modes such as MAC/PHY, GbE/SGMII, etc. have no influence on transmit data path latency.

Table 2-2. Transmit Data Path Latency Specifications

IP Core GMII Configuration	TX FSM Latency	TX Rate Adapt Latency	TX Total Latency
Classic GMII	3	9	12
Easy Connect GMII	3	2	5

Referring back to Figure 2-3, it is shown that the three major elements in the IP core's receive data path include the RX-CTC-block, the RX-state-machine-block, and the RX-rate-adaptation-block. The last two blocks provide constant delays; there is no frame-to-frame latency variation. However, the RX-CTC block is affected by frame-size, clock accuracy, and FIFO-thresholds. The RX-CTC-block does exhibit frame-to-frame latency variation. Table 2-3 shows the RX data path latencies for various frequency accuracies, frame sizes, and (G)MII physical configurations. Note however that all specifications are given with Dynamic/Static CTC FIFO thresholds of 16-low; 32-high (these are the default threshold settings).

Table 2-3. Receive Data Path Latency Specifications

Frequency Offset Between Sender and Receiver	IP Core GMII Configuration	RX CTC Expected Number of Clock Slips per Frame	RX CTC Number of Clock Cycles Added/Dropped per Frame		RX CTC Latency		RX FSM Latency		RX Rate Adapt Latency		RX Total Latency	
			Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.
64-Byte Frames												
+200 ppm	Classic GMII	-0.03	0	2_{add}	24	26	6	6	8	8	38	40
0 ppm	Classic GMII	0	0	0	27	27	6	6	8	8	41	41
-200 ppm	Classic GMII	+0.03	0	2_{drop}	32	34	6	6	8	8	46	48
+200 ppm	Easy Connect GMII	-0.03	0	2_{add}	24	26	6	6	2	2	32	34
0 ppm	Easy Connect GMII	0	0	0	27	27	6	6	2	2	35	35
-200 ppm	Easy Connect GMII	+0.03	0	2_{drop}	32	34	6	6	2	2	40	42
1500-Byte Frames												
+200 ppm	Classic GMII	-0.3	0	2_{add}	24	26	6	6	8	8	38	40
0 ppm	Classic GMII	0	0	0	27	27	6	6	8	8	41	41
-200 ppm	Classic GMII	+0.3	0	2_{drop}	32	34	6	6	8	8	46	48
+200 ppm	Easy Connect GMII	-0.3	0	2_{add}	24	26	6	6	2	2	32	34
0 ppm	Easy Connect GMII	0	0	0	27	27	6	6	2	2	35	35
-200 ppm	Easy Connect GMII	+0.3	0	2_{drop}	32	34	6	6	2	2	40	42

Table 2-3. Receive Data Path Latency Specifications (Continued)

Frequency Offset Between Sender and Receiver	IP Core GMII Configuration	RX CTC Expected Number of Clock Slips per Frame		RX CTC Number of Clock Cycles Added/Dropped per Frame		RX CTC Latency		RX FSM Latency		RX Rate Adapt Latency		RX Total Latency	
				Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.
15,000-Byte Frames													
+200 ppm	Classic GMII	-3.0		2_{add}	4_{add}	25	27	6	6	8	8	39	41
0 ppm	Classic GMII	0		0	0	27	27	6	6	8	8	41	41
-200 ppm	Classic GMII	+3.0		2_{drop}	4_{drop}	31	33	6	6	8	8	45	47
+200 ppm	Easy Connect GMII	-3.0		2_{add}	4_{add}	25	27	6	6	2	2	33	35
0 ppm	Easy Connect GMII	0		0	0	27	27	6	6	2	2	35	35
-200 ppm	Easy Connect GMII	+3.0		2_{drop}	4_{drop}	31	33	6	6	2	2	39	41
Random Frame Sizes From 64 to 15,000 Bytes, Random PPM Offset From -200 to +200													
Random ppm	Classic GMII	-3.0 min.	+3.0 max.	4_{drop}	4_{add}	24	34	6	6	8	8	38	48
Random ppm	Easy Connect GMII	-3.0 min.	+3.0 max.	4_{drop}	4_{add}	24	34	6	6	2	2	32	42

Signal Descriptions

Table 2-4 shows a detailed listing of all the SGMII IP core I/O signals.

Table 2-4. SGMII and Gb Ethernet PCS IP Core Input and Output Signals

Signal Name	I/O	Signal Description
Clock Signals		
tx_clk_125	In	**Transmit 125MHz Clock** - 125 MHz clock for transmit state machine. Incoming data from the transmit rate adaptation block data is sampled on rising edge of this clock. Outgoing 8-bit code group transmit data is launched on the rising edge of this clock.
tx_mii_clk	In	**Transmit MII Clock** - 125 MHz, 25 MHz, or 2.5 MHz clock for incoming (G)MII transmit data. Data is sampled on the rising edge of this clock. Note, this port is only present when the IP core is generated using the “classic” (G)MII option.
tx_clock_enable_source	Out	**Transmit Clock Enable Source** - This signal is only present when the IP core is generated using the “TSMAC Easy Connect” (G)MII option. This signal is used in combination with the transmit 125 MHz clock to regulate the flow of transmit (G)MII data. The signal is generated by the transmit rate adaptation block. This clock enable should be tied to the transmit section of the Lattice MAC that sends transmit Ethernet frames to the SGMII and Gb Ethernet PCS IP core. This clock enable should also be tied to the clock enable “sink” of the SGMII and Gb Ethernet PCS IP core. This clock enable’s behavior is controlled by the setting of the operational_rate pins of the IP core. For 1Gbps operation, the clock enable is constantly high. For 100 Mbps operation, the clock enable is high for one-out-of-ten 125 MHz clock cycles. For 10 Mbps operation, the clock enable is high for one-out-of-one-hundred 125 MHz clock cycles.
tx_clock_enable_sink	In	**Transmit Clock Enable Sink** - This signal is only present when the IP core is generated using the “TSMAC Easy Connect” (G)MII option. This signal is used in combination with the transmit 125 MHz clock to regulate the flow of transmit (G)MII data. When the clock enable is high and the transmit clock edge rises, (G)MII data is sampled.
rx_clk_125	In	**Receive 125MHz Clock** - 125 MHz clock for synchronization and receive state machines. Incoming code groups from the receive CTC block data are sampled on rising edge of this clock. Outgoing GMII data is launched on the rising edge of this clock.
rx_mii_clk	In	**Receive MII Clock** - 125 MHz, 25 MHz, or 2.5 MHz clock for outgoing (G)MII receive data. Data is launched on the rising edge of this clock. Note, this port is only present when the IP core is generated using the “classic” (G)MII option.
rx_clock_enable_source	Out	**Receive Clock Enable Source** - This signal is similar to the tx_clock_enable_source described above, except that it is used for the receive data path. Note that this signal is only present when the IP core is generated using the “TSMAC Easy Connect” (G)MII option.
rx_clock_enable_sink	In	**Receive Clock Enable Sink** - This signal is only present when the IP core is generated using the “TSMAC Easy Connect” (G)MII option. This signal is used in combination with the receive 125 MHz clock to regulate the flow of receive (G)MII data. When the clock enable is high and the receive clock edge rises, (G)MII data is launched.

Table 2-4. SGMII and Gb Ethernet PCS IP Core Input and Output Signals (Continued)

Signal Name	I/O	Signal Description
SERDES_recovered_clk	In	**SERDES Recovered Clock** -125 MHz clock recovered from receive side of SERDES. This signal is synchronous with the 8-bit code groups received by the SERDES and should be used to clock receive data between the SERDES and the IP core. The IP core samples incoming 8-bit code groups on the rising edge of this clock.
GMII Signals		
tx_d[7:0]	In	**Transmit Data** - Incoming (G)MII data. Note, this port's behavior varies depending on the (G)MII option used when generating the IP core. For "classic" mode, when the (G)MII data rate is 1Gbps, all 8 bits of tx_d are valid. However, for 100 Mbps and 10 Mbps, only bits 3:0 of tx_d are valid. For the "TSMAC Easy Connect" mode all 8 bits of tx_d are valid for all (G)MII data rates (1Gbps, 100 Mbps, 10 Mbps).
tx_en	In	**Transmit Enable** - Active high signal, asserts when incoming data is valid.
tx_er	In	**Transmit Error** - Active high signal used to denote transmission errors and carrier extension on incoming (G)MII data port.
rx_d[7:0]	Out	**Receive Data** - Outgoing (G)MII data. Note, this port's behavior varies depending on the (G)MII option used when generating the IP core. For "classic" mode, when the (G)MII data rate is 1Gbps, all 8 bits of rx_d are valid. However, for 100 Mbps and 10 Mbps, only bits 3:0 of rx_d are valid. For the "TSMAC Easy Connect" mode all 8 bits of rx_d are valid for all (G)MII data rates (1Gbps, 100 Mbps, 10 Mbps).
rx_dv	Out	**Receive Data Valid** - Active high signal, asserts when outgoing data is valid.
rx_er	Out	**Receive Error** - Active high signal used to denote transmission errors and carrier extension on outgoing (G)MII data port.
col	Out	**Collision Detect** - Active high signal, asserts when tx_en and rx_dv are active at the same time.
crs	Out	**Carrier Sense Detect** - Active high signal, asserts when rx_dv is high.
8-Bit Code Group Signals		
tx_data[7:0]	Out	**8b Transmit Data** - 8-bit code group data after passing through transmit state machine. Note: the transmit 8-bit code group interface operates differently in the LatticeECP2M and LatticeSC FPGA families. For LatticeECP2M, the port truly behaves as an 8-bit code group interface where data is tx_data[7:0], and tx_kcntl and tx_disparity_cntl are control signals for the 8-bit to 10-bit encoder within the LatticeECP2M SERDES. However for LatticeSC, the port behaves as a 10-bit code group interface, with signals tx_kcntl and tx_disparity_cntl forming bits 8 and 9 of the 10-bit interface. In addition, the 8-bit to 10-bit encoder resides within the IP core; and consequently the encoder within the LatticeSC SERDES is bypassed. Note that even though there are functional differences between the LatticeECP2M and LatticeSC implementations, the IPcore/SERDES connectivity is the same for both implementations - tx_data[7:0] of IP core connects to tx_data[7:0] of SERDES - tx_kcntl of IP core connects to tx_kcntl of SERDES - tx_disparity_cntl connects to tx_disparity_cntl of SERDES.
tx_kcntl	Out	**8b Transmit K Control** - Denotes whether current code group is data or control. 1 = control, 0 = data.
tx_disparity_cntl	Out	**8b Transmit Disparity Control** - Used to influence the disparity state of an idle code group at the beginning of an inter-packet gap. 1 = Insert /I1/ if inter-packet gap begins with positive disparity, otherwise insert /I2/; 0 = normal, insert /I2/.
rx_data[7:0]	In	**8b Receive Data** - 8-bit code group data presented to the receive state machine.
rx_kcntl	In	**8b Receive K Control** - Denotes whether current code group is data or control. 1 = control, 0 = data.

Table 2-4. SGMII and Gb Ethernet PCS IP Core Input and Output Signals (Continued)

Signal Name	I/O	Signal Description
rx_err_decode_mode	In	**Receive Error Control Mode** - The embedded SERDES block of the LatticeECP2M FPGAs has two modes of interpreting errors - decoded and normal. In decoded mode, the three signals (rx_even, rx_cv_err, rx_disp_err) are used to decode one of eight error conditions. In decoded mode, the IP core responds to the following errors: 100 - Coding Violation Error 111 - Disparity Error All other error codes are ignored by the IP core. In normal mode, the three error signals (rx_even, rx_cv_err, rx_disp_err) behave normally. The rx_err_decode_mode signal should be set high for decode-mode, and low for normal mode. When using LatticeSC devices, the pin should always be tied low.
rx_even	In	**Rx Even** - This signal is only used when error decoding mode is active. Otherwise, the signal should be tied low.
rx_cv_err	In	**Rx Coding Violation Error** - In normal mode, an active high signal denoting a coding violation error in the receive data path. In decode mode, used to decode 1 of 8 error conditions.
rx_disp_err	In	**Rx Disparity Error** - In normal mode, an active high signal denoting a disparity error in the receive data path. In decode mode, used to decode 1 of 8 error conditions.
signal_detect	In	**Signal Detect** - Denotes status of SERDES Rx physical link. 1 = signal is good; 0 = loss of receive signal.
rx_compensation_err	Out	**Receive CTC Compensation Error** - This active high signal asserts when the receive CTC FIFO either underflows or overflows.
ctc_drop_flag	Out	**CTC Drop Indication** – Active high signal that asserts when the CTC logic drops an idle-code-group from the RX data stream.
ctc_add_flag	Out	**CTC Add Indication** – Active high signal that asserts when the CTC logic adds an idle-code group to the RX data stream.
xmit_autoneg	Out	**Auto-negotiation XMIT Status** – This signal should be tied to the “xmit_chn” pin of the SERDES. When the signal is high, it forces the RX data path of the SERDES to periodically insert idle-code groups into the SERDES RX data stream. This ensures that the SERDES CTC has opportunities to rate-adapt for ppm offsets. The PCS IP core drives this signal high when the auto-negotiation process is running. At all other times, the PCS IP core drives this signal low. Use of this signal is required if the designer chooses to use the CTC function embedded within the SERDES (instead of using the soft CTC function within the SGMII/GbE PCS IP core). Otherwise, the use of this signal is optional.
Management Signals		
mr_adv_ability[15:0]	In	**Advertised Ability** - Configuration status transmitted by PCS during auto-negotiation process.
mr_an_enable	In	**Auto-Negotiation Enable** - Active high signal that enables auto-negotiation state machine to function.
mr_main_reset	In	**Main Reset** - Active high signal that forces all PCS state machines to reset.
mr_restart_an	In	**Auto-Negotiation Restart** - Active high signal that forces auto-negotiation process to restart.
mr_an_complete	Out	**Auto-Negotiation Complete** - Active high signal that indicates that the auto-negotiation process is completed.
mr_lp_adv_ability[15:0]	Out	**Link Partner Advertised Ability** - Configuration status received from partner PCS entity during the auto-negotiation process. The bit definitions are the same as described above for the mr_adv_ability port.
mr_page_rx	Out	**Auto-Negotiation Page Received** - Active high signal that asserts while the auto-negotiation state machine is in the “Complete_Acknowledge” state.

Table 2-4. SGMII and Gb Ethernet PCS IP Core Input and Output Signals (Continued)

Signal Name	I/O	Signal Description
force_isolate	In	**Force PCS Isolate** – Active high signal that isolates the PCS. When asserted, the RX direction forces the (G)MII port to all zeros, regardless of the condition of the incoming 1.25Gbps serial data stream. In the TX direction, the condition of the incoming (G)MII port is ignored. The TX PCS behaves as though the (G)MII TX input port was forced to all zeros. Note, however, that the isolate function does not produce any electrical isolation – such as tri-stating of the (G)MII RX outputs of the IP core. When the signal is deasserted (low), the PCS isolation functions are deactivated. The use of this signal is optional. If the user chooses not to use the isolate function, then this signal should be tied low.
force_loopback	In	**Force PCS Loopback** – Active high signal that activates the PCS loopback function. When asserted, the 8-bit code-group output of the transmit state machine is looped back to the 8-bit code-group input of the receive state machine. When deasserted, the loopback function is deactivated. The use of this signal is optional. If the user chooses not to use the loopback function, then this signal should be tied low.
force_unidir	In	**Force PCS Unidirectional Mode** – Active high signal that activates the PCS unidirectional mode. When asserted, the transmit state machine path between the TX (G)MII input and the TX 8-bit code-group output will remain operational, regardless of what happens on the RX data path. (Normally RX loss of sync, invalid code-group reception, auto-negotiation restarts can force the transmit state machine to temporarily ignore inputs from the TX (G)MII port). When deasserted, the unidirectional mode is deactivated. The use of this signal is optional. If the user chooses not to use the unidirectional function, then this signal should be tied low.
an_link_ok	Out	**Auto-Negotiation Link Status OK** – Active high signal that indicates that the link is ok. The signal is driven by the auto-negotiation state machine. When auto-negotiation is enabled, the signal asserts when the state machine is in the LINK_OK state. If auto-negotiation is disabled, the signal asserts when the state machine is in the AN_DISABLE_LINK_OK state (see IEEE 802.3 figure 37-6). This signal is intended to be used to produce the “Link Status” signal as required by IEEE 802.3, Status Register 1, Bit D2 (see IEEE 802.3 paragraph 22.2.4.2.13)
Miscellaneous Signals		
rst_n	In	**Reset** - Active low global reset.
sgmii_mode	In	**SGMII Mode** – Controls the behavior of the auto-negotiation process when the core is operating in SGMII mode. 0 = operates as MAC-side entity, 1 = operates as PHY-side entity.
gbe_mode	In	**Gigabit Ethernet Mode** – Controls the core’s PCS function. 0 = operates as SGMII PCS, 1 = operates as Gigabit Ethernet PCS (1000BaseX)
operational_rate[1:0]	In	**Operational Rate** – When the core operates in SGMII PCS mode, this port controls the regulation rate of the rate adaptation circuit blocks as follows: 10 = 1G bps Rate 01 = 100 Mbps Rate 00 = 10 Mbps Rate Note in Gigabit Ethernet PCS mode, the rate adaptation blocks always operates at the 1Gbps rate, regardless of the settings on the operational_rate control pins.
debug_link_timer_short	In	**Debug Link Timer Mode** – Active high signal that forces the auto-negotiation link timer to run much faster than normal. This mode is provided for debug purposes (e.g., allowing simulations to run through the auto-negotiation process much faster than normal).

The IPexpress tool is used to create IP and architectural modules in the Diamond software. Refer to "IP Core Generation" on page 19 for a description on how to generate the IP.

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

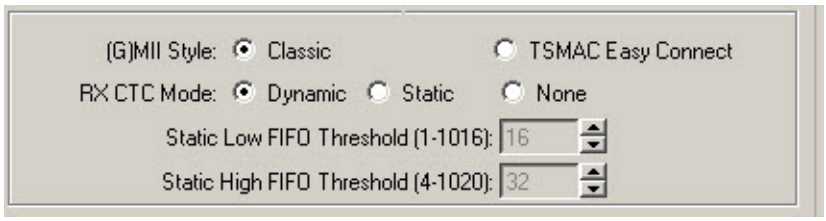
Table 3-1. SGMII and Gb Ethernet PCS IP Core Configuration Parameters

Parameter	Range/Options	Default
(G)MII Style	Classic/ TSMAC Easy Connect	Classic
RX CTC Mode	Dynamic/ Static or None	Dynamic
Static CTC FIFO Low Threshold	3-1010	16
Static CTC FIFO High Threshold	13-1020	32

General Tab

Figure 3-1 shows the contents of the SGMII and Gb Ethernet Options in the IPexpress Tool.

Figure 3-1. SGMII and Gb Ethernet Options in the IPexpress Tool



The SGMII and Gb Ethernet IP options of the following parameters:

(G)MII Style

This parameter affects the behavior and implementation of the (G)MII port. In "Classic" mode, the (G)MII data port is 8 bits wide. All 8 bits are used for 1Gbps operation. Only the lower 4 bits are used for 100Mbs and 10 Mbps operation. A separate MII clock is used to synchronize the (G)MII data. The MII clock frequency varies with the (G)MII data rate: 125 MHz for 1 Gbps, 25 MHz for 100 Mbps, 2.5 MHz for 10 Mbps. For the "TSMAC Easy Connect" mode, the (G)MII data port is 8 bits wide; and all 8 bits are used, regardless of the (G)MII data rate. A single 125 MHz clock is used to synchronize (G)MII data; and a clock enable is used to regulate the (G)MII data rate.

RX CTC Mode

This parameter controls the behavior of the CTC block. In dynamic mode, the CTC FIFO thresholds are automatically changed, based upon the current operational rate of the rate adaptation blocks. Optimal thresholds are internally chosen for the three data rates (1Gbps, 100 Mbps, 10 Mbps). In static mode, the user manually chooses the CTC FIFO thresholds, and these thresholds remain fixed. In "none" mode, the CTC function is replaced by a shallow FIFO that facilitates clock domain crossing between the recovered SERDES clock and the local IP core receive-side 125 MHz clock.

Static CTC FIFO Low Threshold

When RX CTC Static mode is chosen, this parameter specifies FIFO low (almost empty) threshold.

Static CTC FIFO High Threshold

When RX CTC Static mode is chosen, this parameter specifies FIFO high (almost full) threshold

Guidelines for Calculating Static CTC FIFO Thresholds

The FIFO thresholds directly affect the compensation capability of the CTC circuit. The thresholds should be set to accommodate the transmission characteristics of the system into which the IP core is used. The critical system characteristics for setting the FIFO thresholds are maximum ppm offset, maximum frame size, and (G)MII data rate. The following procedure can be used to calculate the Static CTC FIFO thresholds.

1. Calculate the maximum clock slip between maximum-sized frames, rounded up to the nearest integer,

$$M = \frac{2N\delta}{R} \times 10^{-3}$$

where:

- N = maximum ethernet frame size (bytes)
- δ = maximum ppm offset (e.g. 100 for +/- 100 ppm offset; 200 for +/- 200 ppm offset, etc.)
- R = (G)MII data rate (e.g. 1000 for 1Gbps, 100 for 100 Mbps, 10 for 10Mbps)

2. Calculate the low FIFO threshold. Note this equation computes the minimum required threshold. However, using a larger value will provide additional design margin.

$$THRESHOLD_{LOW} = M + 2$$

3. Calculate the high FIFO threshold. Note this equation computes the minimum required threshold. However, using a larger value will provide additional design margin..

$$THRESHOLD_{HIGH} = THRESHOLD_{LOW} + 10$$

4. Calculate the minimum acceptable FIFO full level. This value must be less than 1024 -- the maximum space allocated for the FIFO. If the computed full level is greater than 1024, then the static CTC will not meet the specified design constraints..

$$THRESHOLD_{FULL} = THRESHOLD_{HIGH} + M + 2$$

Table 3-2 provides minimum CTC static threshold calculations for six different system specifications.

Table 3-2. Minimum CTC Static Threshold Calculations

	100	100	100	100	100	100	Max ppm Offset (+/-)
	1500	1500	1500	9600	9600	9600	Max Frame Size (bytes)
	1000	100	10	1000	100	10	(G)MII Data Rate (Mbps)
M	1	3	30	2	20	193	
$Threshold_{LOW}$	3	5	32	4	22	195	
$Threshold_{HIGH}$	13	15	42	14	32	205	
$Threshold_{FULL}$	16	20	74	18	54	400	

IP Core Generation in IPexpress

This chapter provides information on how to generate the Lattice SGMII IP core using the IPexpress tool in the Diamond software, and how to include the core in a top-level design.

Licensing the IP Core

An IP core- and device-specific license is required to enable full, unrestricted use of the SGMII IP core in a complete, top-level design. 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 SGMII IP core and fully evaluate the core through functional simulation and implementation (synthesis, map, place and route) without an IP license. The SGMII 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 26 for further details. However, a license is required to enable timing simulation, to open the design in the Diamond EPIC tool, and to generate bitstreams that do not include the hardware evaluation timeout limitation.

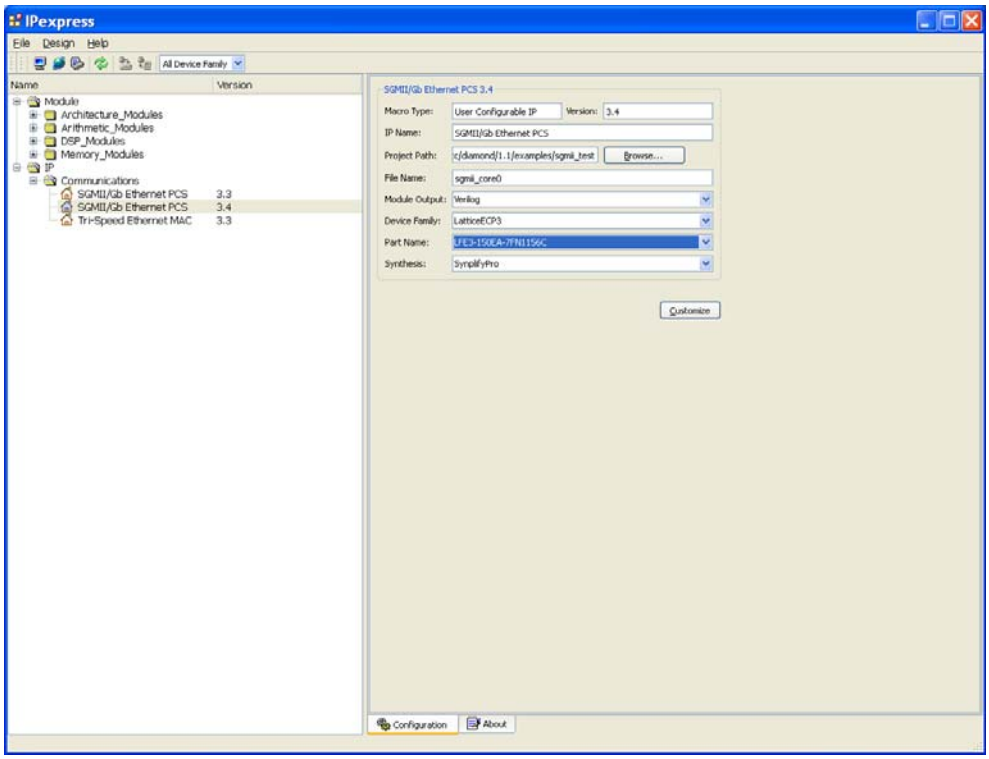
Getting Started

The SGMII and Gb Ethernet PCS 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 IPexpress tool GUI dialog box for the SGMII and Gb Ethernet PCS 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.
- **Module Output** – Verilog or VHDL.
- **Device Family** – Device family to which IP is to be targeted (e.g. LatticeECP3, ECP5, 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)

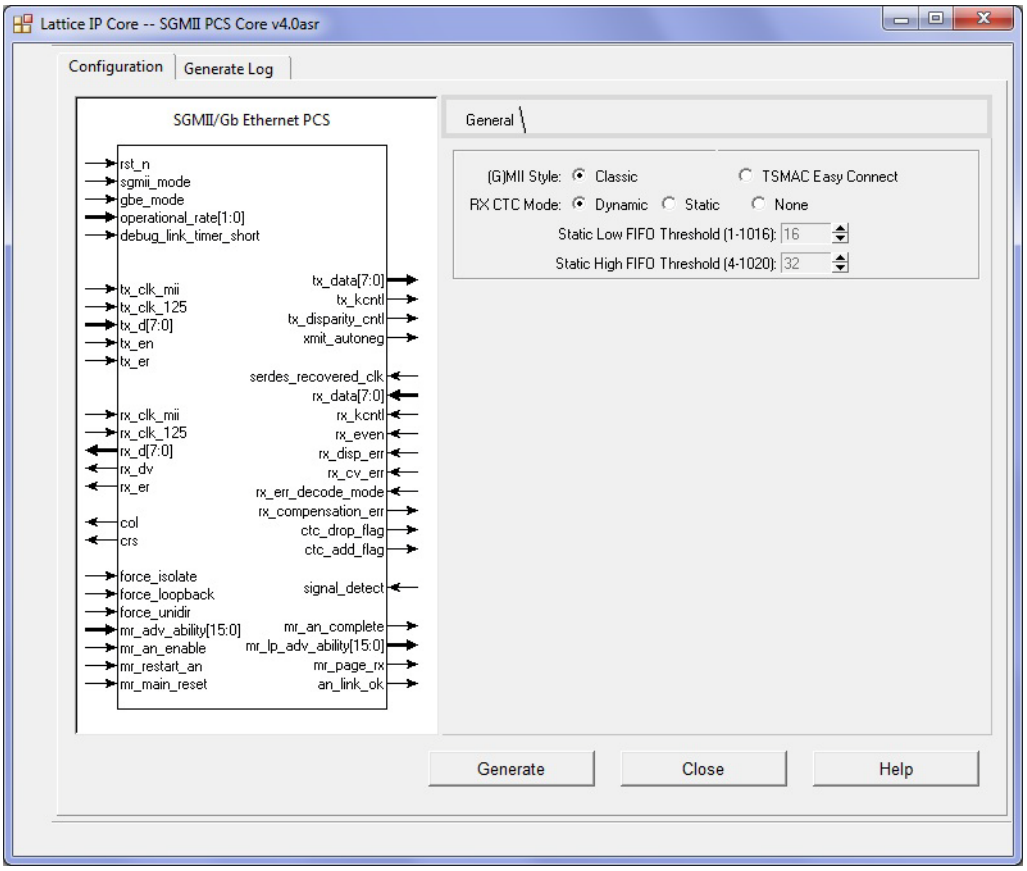


Note that if the IPexpress tool is called from within an existing project, Project Path, Module Output, Device Family and Part Name default to the specified project parameters. Refer to the IPexpress tool online help for further information.

To create a custom configuration:

1. Click the **Customize** button in the IPexpress tool dialog box to display the SGMII IP core Configuration GUI, as shown in Figure 4-2.
2. Select the IP parameter options specific to your application. Refer to "Parameter Settings" on page 17 for more information on the SGMII and Gb Ethernet PCS IP core parameter settings.

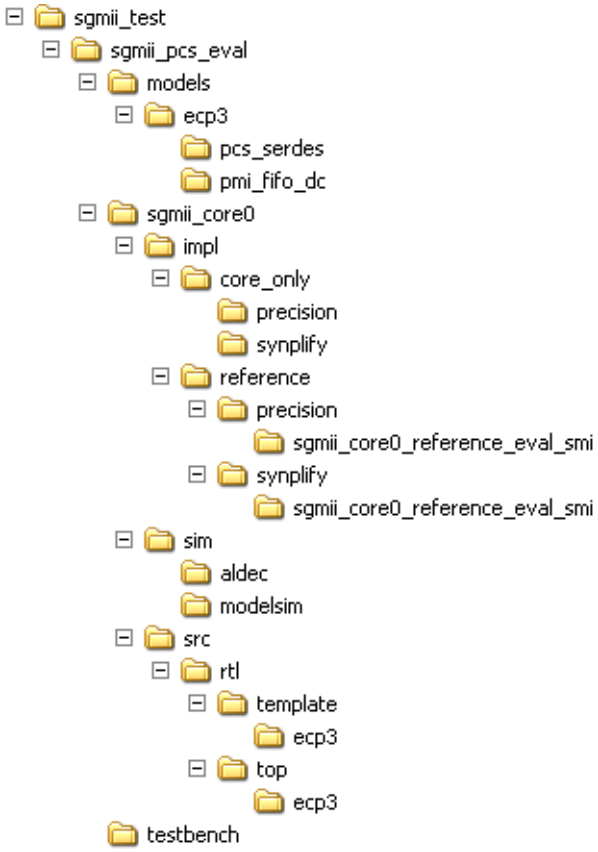
Figure 4-2. Configuration Dialog Box (Diamond Version)



IPexpress-Created Files and Top Level Directory Structure

When the user clicks the **Generate** button, 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. SGMII and Gb Ethernet PCS IP Core Generated 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 during the IPexpress tool generation.

Table 4-1 provides a list of key files created by the IPexpress tool. The names of most of the created files are customized to the user's module name specified in the IPexpress tool. The files shown in Table 4-1 are all of the files necessary to implement and verify the SGMII and Gb Ethernet PCS IP core in a top-level design.

Table 4-1. File List

File	Description
<*username*>_inst.v	This file provides an instance template for the IP.
<*username*>_bb.v	This file provides the synthesis black box for the user's synthesis.
<*username*>_beh.v	This file provides a behavioral simulation model for the IP core.
<*username*>.ngo	This file provides the synthesized IP core.
<*username*>.lpc	This file contains the IPexpress tool options used to recreate or modify the core in the IPexpress tool.

Table 4-1. File List (Continued)

File	Description
<*username*>.ipx	IPexpress package file (Diamond only). This is a container that 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 this file to the associated Diamond project.
pmi_*.ngo	One or more files implementing synthesized memory modules used in the IP core.

The following additional files providing IP core generation status information are also generated in the "Project Path" directory:

- **<username>_generate.log** – IPexpress synthesis and EDIF2NGD log file.
- **<username>_gen.log** – IPexpress IP generation log file.

The **\<sgmii_pcs_eval>** and subtending directories shown in Figure 4-3 provide files supporting SGMII and Gb Ethernet PCS core evaluation. The **\<sgmii_pcs_eval>** directory contains files/folders with content that is constant for all configurations of the SGMII and Gb Ethernet PCS. The **\<***username***>** subfolder (**\sgmii_core0** in this example) contains files/folders with content specific to the username configuration.

The **\sgmii_pcs_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. **\<my_core_0>**, **\<my_core_1>**, etc.

Instantiating the Core

The generated SGMII and Gb Ethernet PCS 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***>\sgmii_pcs_eval\<***username***>\src\rtl\top\<technology>**.

The top-level file top_smi.v (top_hb.v when TSMAC Easy Connect mode is chosen) implements the SGMII-to-(G)MII reference design described in "SGMII-to-(G)MII Reference Design" on page 33. Verilog source files associated with the reference design are located in the following directory: **\<***project_dir***>\sgmii_pcs_eval\<***username***>\src\rtl\template\<technology>**.

The top-level file top_pcs_core_only.v supports the ability to implement just the SGMII and Gb Ethernet PCS core by itself. This design is intended only to provide an indication of the device utilization associated with the SGMII and Gb Ethernet PCS IP core; and it should not be used as useful example of a design application.

Using the SERDES with the SGMII IP Core

Note that most applications for the SGMII IP core require use of the FPGA SERDES block. The reference design (see "SGMII-to-(G)MII Reference Design" on page 33) demonstrates how the SERDES is used with the SGMII IP core. However, note that the reference design only demonstrates one implementation – a single SERDES, assigned to channel 0. If your application requires a different SERDES configuration, for example utilizing a different channel number, or utilizing multiple channels, then you cannot use the SERDES module provided in the reference design. Instead, you must generate an appropriate SERDES module with the configuration settings required for your application. The SERDES module can be generated using IPexpress. Please see TN1176, *LatticeECP3 SERDES/PCS Usage Guide* for details about configuring the SERDES.

The following are a few key SERDES settings that must be chosen for SGMII applications:

- ECP5:
  - Channel Protocol: SGMII
  - CTC block: DISABLED (Default is DISABLED)
- LatticeECP3:
  - Channel Protocol: SGMII
  - CTC block: DISABLED (Default is DISABLED)

*IPexpress only allows the default CTC settings to be chosen. You must manually alter these defaults CTC settings by altering the SERDES auto-config file generated by IPexpress. For example, with LatticeECP2M, open the auto-config file with a text editor. Search for a line containing the CTC setting (e.g. CH0_CTC_BYP). Change the word "NORMAL" to the word "BYPASS".

Using CTC with the SGMII IP Core

Note that both the SERDES and the SGMII IP core have clock tolerance compensation (CTC) circuits. So there may be some confusion about which CTC circuit should be used with the IP core. Lattice recommends that the SERDES CTC should be turned-off, and the SGMII IP core CTC should be turned-on. Designers should not choose the case where both the SERDES CTC and the IP core CTC are turned-on at the same time; unpredictable behavior may result.

The enable/disable CTC circuit settings are specified through the IPexpress tool when the SERDES and/or SGMII IP core are generated.

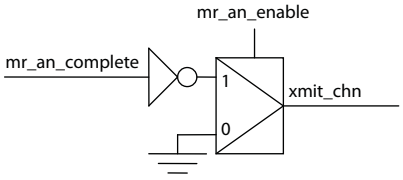
There may be some cases where the designer chooses to disable the CTC within the IP core (SGMII CTC mode = “none”). This mode was intended for use in synchronous systems, where the SERDES recovered clock and the local RX clock have no ppm offset. The CTC of the SERDES should also be disabled for such systems. This configuration will provide reliable operation between the SERDES and IP core.

However, there may be cases where the designer chooses to disable the IP core CTC circuit within an asynchronous system (where there is a ppm offset between SERDES recovered clock and the local RX clock). In this case the user may intend to utilize the SERDES CTC. It should be noted that there are some limitations when using the SERDES CTC with the IP core. For the LatticeSC FPGA family, it is not possible to utilize the SERDES CTC with the IP core because the SERDES permanently disables the CTC when the Generic 8B10B SERDES mode is selected. However, it is possible to utilize the SERDES CTC with the IP core within the LatticeECP3 and ECP5 FPGA families. But even in these three FPGA families, there are limitations on how the SERDES CTC may be used with the IP core. The restriction is associated with auto-negotiation. The SERDES CTC requires that idle-code-groups are occasionally added to the RX data path to maintain proper CTC operation. This happens normally during the inter-packet gaps of normal ethernet frame transmission. However, during auto-negotiation idle-code-group transmissions are not always present. To overcome this problem, the SERDES CTC provides an input pin (labeled “xmit_chn” in LatticeECP3 and ECP5 devices). When this pin is driven high, it occasionally inserts idle-code-groups into the RX data path. If the IP core is allowed to run auto-negotiation, then the SERDES “xmit_chn” pin must be driven “high” throughout most of the auto-negotiation process. After the auto-negotiation process completes, the “xmit_chn” pin should be driven “low.” If auto-negotiation never runs, then the SERDES “xmit_chn” pin can be driven permanently driven “low.”

For SGMII IP core versions 3.4 and higher, the IP core provides a signal labelled "xmit_autoneg" that can be tied to the SERDES "xmit_chn" pin to satisfy the required insertion of idle code groups while auto negotiation runs.

For SGMII IP core versions 3.3 and lower, the IP core does not provide a direct signal to drive the “xmit_chn” SERDES pin. However, the designer can create an appropriate control signal for driving “xmit_chn” by combining two IP core output signals as shown in Figure 4-4.

Figure 4-4. Combining Two IP Core Output Signals



Running Functional Simulation

The functional simulation model generated in the “Project Path” root directory (<username>_beh.v) may be instantiated in the user's testbench for evaluation in the context of their design application. Lattice does not provide a testbench for evaluating this IP core in isolation. However, a functional simulation capability is provided in which <username>_beh.v is instantiated in the SGMII-to-(G)MII reference design described in “SGMII-to-(G)MII Reference Design” on page 33. This FPGA top is instantiated in a testbench provided in **\<*project_dir*>\sgmii_pcs_eval\testbench**.

Users may run the eval simulation by doing the following.

Using Aldec Active-HDL:

1. Open Active-HDL.

2. Under the Tools tab, select **Execute Macro**.

3. Browse to folder **\<*project_dir*>\sgmii_pcs_eval\<*username*>\sim\aldec** and execute one of the "do" scripts shown.

Using Mentor Graphics ModelSim

1. Open ModelSim.

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

3. Under the Tools tab, select **Execute Macro** and execute the ModelSim “do” script shown.

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

Synthesizing and Implementing the Core in a Top-Level Design

The SGMII and Gb Ethernet PCS IP core itself is synthesized and provided in NGO format when the core is generated through IPexpress. You may 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 following text describes the evaluation implementation flow for Windows platforms. The flow for Linux and UNIX platforms is described in the Readme file included with the IP core.

As described previously, the top-level file top_pcs_core_only.v provided in **\<*project_dir*>\sgmii_pcs_eval\<*username*>\src\rtl\top** supports the ability to implement the SGMII and Gb Ethernet PCS core in isolation. Push-button implementation of this top level design is supported via the project file <*username*>_core_only_eval.ldf located in **\<*project_dir*>\sgmii_pcs_eval\<*username*>\impl**.

To use this project file in Diamond:

1. Choose **File > Open > Project**.
2. Browse to \<*project_dir*>**sgmii_pcs_eval**\<*username*>**impl\core_only\synplify (or precision)** in the Open Project dialog box.
3. Select and open <*username*>_core_only_eval.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.

Push-button implementation is also supported for the SGMII-to-(G)MII reference design specified by the top-level file top_smi.v described previously. This capability is supported via the Diamond project file located in \<*project_dir*>**sgmii_pcs_eval**\<*username*>**impl\reference\synplify**. The flow for implementing the reference design is equivalent to the flow for implementing the core-only design described above.

Hardware Evaluation

The SGMII and Gb Ethernet PCS 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 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.

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**.

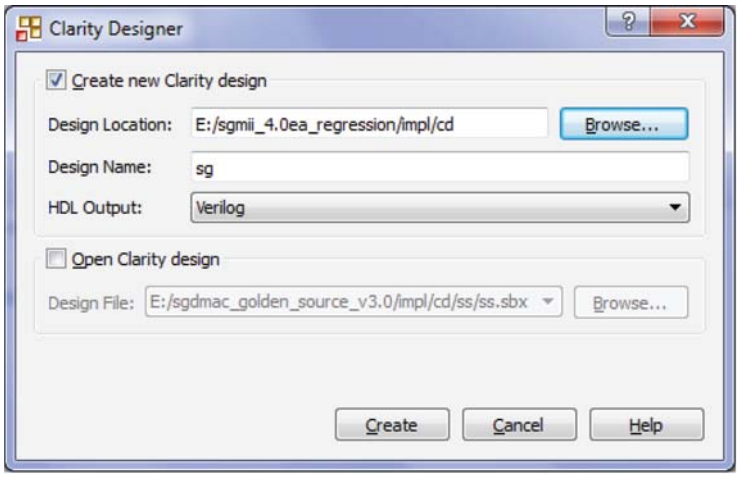
The IPexpress package file (.ipx) supported by Diamond holds references to all of the elements of the generated IP core required to support simulation, synthesis and implementation. The IP core may be included in a user's design by importing the .ipx file to the associated Diamond project. To change 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.

IP Core Generation in Clarity Designer

Getting Started

The first step in generating an IP Core in Clarity Designer is to start a project in Diamond software with the ECP5 device. Clicking the Clarity Designer button opens the Clarity Designer tool.

Figure 4-5. Starting a Project in Clarity Designer



To generate a specific IP core configuration the user specifies:

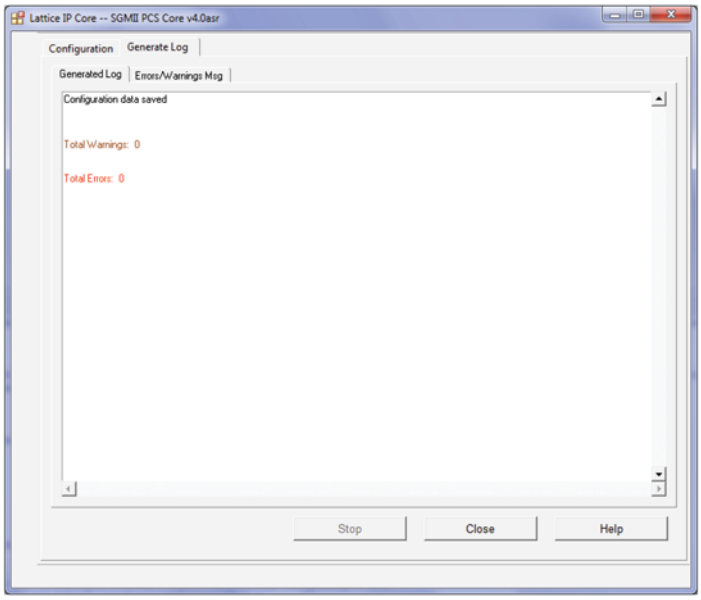
- **Instance Path** – Path to the directory where the generated IP files will be located.
- **Instance Name** – "username" designation given to the generated IP core and corresponding folders and files.
- **Module Output** – Verilog or VHDL.
- **Device Family** – Device family to which IP is to be targeted.
- **Part Name** – Specific targeted part within the selected device family.

Note that because the Clarity Designer tool must be called from within an existing project, Project Path, Module Output, 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:

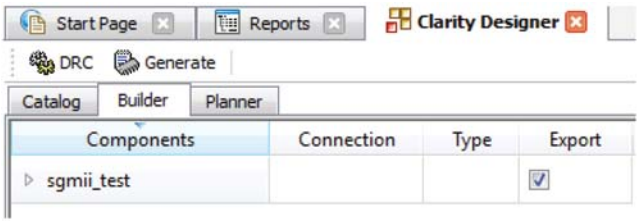
1. Click the **Customize** button in the Clarity Designer dialog box to display the SGMII IP core Configuration GUI, as shown in Figure 4-2.
2. Select the IP parameter options specific to your application. Refer to "Parameter Settings" on page 17 for more information on the SGMII IP core parameter settings.
3. After setting the parameters, click **Customize**.
4. A dialog box, shown in Figure 4-6, displays logs, errors and warnings. Click **Close**.

Figure 4-6. Clarity Designer Generate Log Tab



5. The Clarity Designer Builder tab, shown in Figure 4-7, opens.

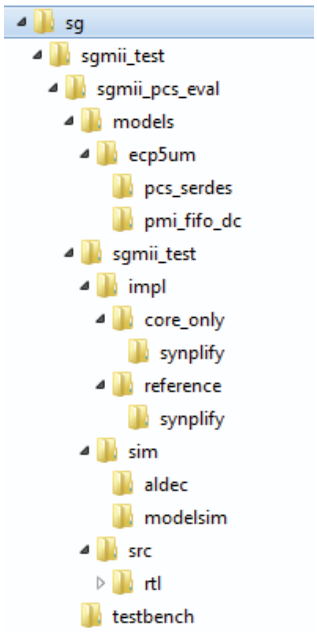
Figure 4-7. Clarity Designer Builder Tab



Clarity Designer Created Files and Top Level Directory Structure

The directory structure of the generated files is shown in Figure 4-8.

Figure 4-8. ECP5 SGMII IP Core Directory Structure



The design flow for IP created with the Clarity Designer tool uses post-synthesized modules (NGO) for synthesis and a protected model for simulation. The post-synthesized module are customized and created during the Clarity Designer tool generation.

Table 4-2 provides a list of key files and directories created by the Clarity Designer tool and how they are used. The Clarity Designer 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 Clarity Designer tool.

Table 4-2. File List

File	Description
<*username*>.v	This file provides the IP core wrapper.
<*username*>_inst.v	This file provides an instance template for the IP.
<*username*>_bb.v	This file provides the synthesis black box for the user's synthesis.
<*username*>_beh.v	This file provides a behavioral simulation model for the IP core.
<*username*>_core.ngo	This file provides the synthesized IP core.
<*username*>_pcs.ngo	This file provides the synthesized PCS logic.
<*username*>.lpc	This file contains the IPexpress tool options used to recreate or modify the core in the IPexpress tool.
<*username*>.ipx	IPexpress package file (Diamond only). This is a container that 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 this file to the associated Diamond project.
pmi_*.ngo	One or more files implementing synthesized memory modules used in the IP core.

The following additional files, which provides IP core generation status information, are also generated in the Project Path directory:

- **<username>_generate.log** – Clarity synthesis and EDIF2NGD log file.
- **<username>_gen.log** – Clarity IP generation log file.

Simulation Evaluation

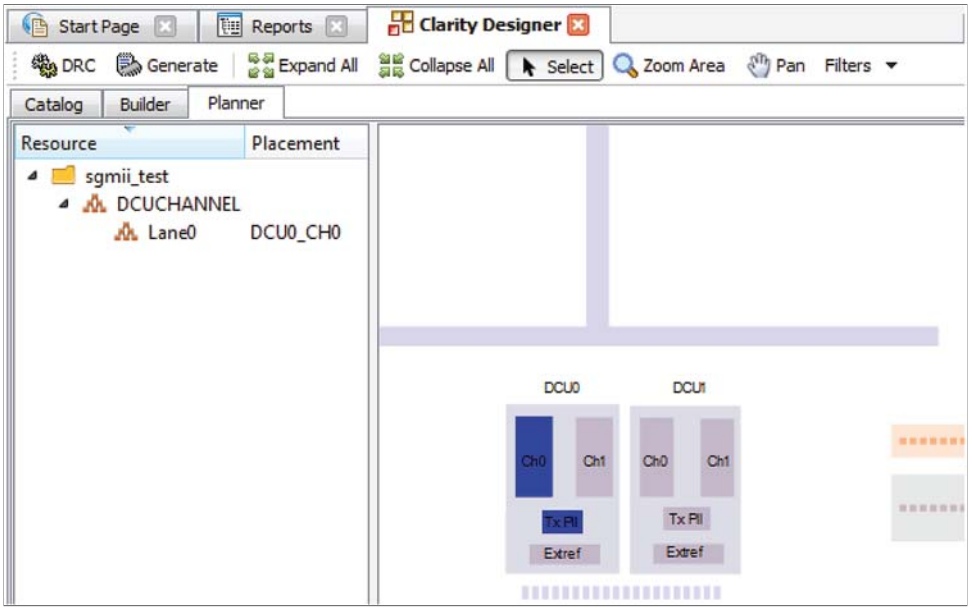
Please refer to "Simulation Evaluation" on page 30 for details.

IP Core Implementation

After completing the Configuration step, perform the procedure below. Please refer to Figure 4-10.

1. Click the **Planner** tab.

Figure 4-9. Planner Tab in Clarity Designer

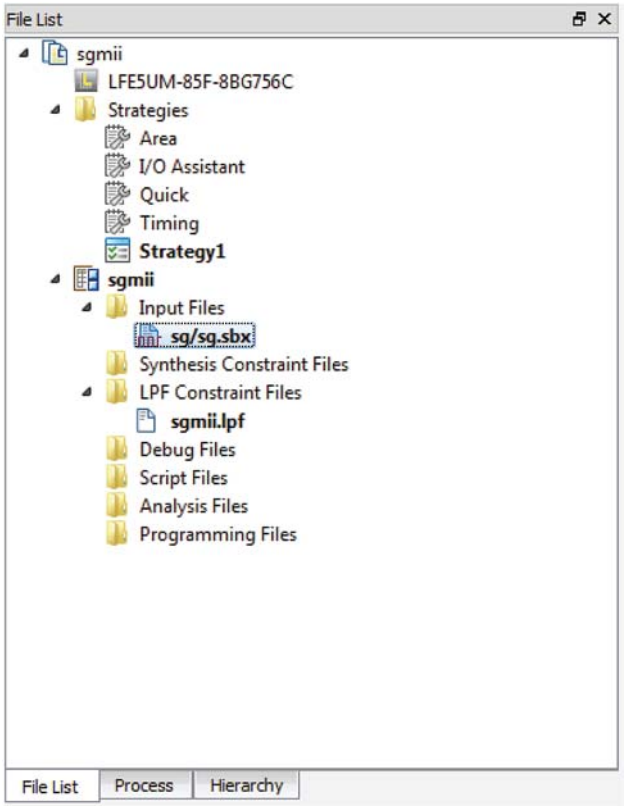


2. Expand the configured IP instance to Lane level.
3. Drag the lanes of the IP to the DCU channel(s).
4. Click the **Generate** button to create the Clarity Designer (.sbx) file.

Clarity Designer (.sbx) files can be used in design projects such as an HDL file or an IPexpress generated (.ipx) file. A key difference between IPexpress generated files and Clarity Designer generated files is that the latter may contain not only a single block but multiple modules or IP blocks and may represent a subsystem. In IPexpress, the process generates a single module or IP. This is a one step process since an IPexpress file can only contain one module or IP. In Clarity Designer, saving a file is a separate step. Modules or IP are configured and multiple modules or IP can optionally be added within the same file. Additionally, since building and planning can also be done, saving the file and generating the blocks may be performed later.

After the Generate step is completed, the ".sbx" file is automatically added to current Diamond Project Input Files list as shown in Figure 4-10.

Figure 4-10. File List in Report Dialog Box



After this step, click **Process** at the bottom of window, then double-click **Place & Route Design** to Start PAR. This is similar to a standard Diamond PAR flow.

Regenerating/Recreating the IP Core

By *regenerating* an IP core with the Clarity Designer tool, you can modify any of the options specific to an existing IP instance. By *recreating* an IP core with Clarity Designer tool, you can create (and modify if needed) a new IP instance with an existing LPC/IPX configuration file.

Regenerating an IP Core in Clarity Designer Tool

To regenerate an IP core in Clarity Designer:

1. In the Clarity Designer Builder tab, right-click on the existing IP instance and choose **Config**.
2. In the module dialog box, choose the desired options.

   For more information about the options, click **Help**. You may also click the **About** tab in the Clarity Designer window for links to technical notes and user guides. The 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 needs.
3. Click **Configure**.

Recreating an IP Core in Clarity Designer Tool

To recreate an IP core in Clarity Designer:

1. In Clarity Designer click the Catalog tab.
2. Click the **Import IP** tab (at the bottom of the view).
3. Click **Browse**.
4. In the Open IPX File dialog box, browse to the .ipx or .lpc file of the module. Use the .ipx if it is available.
5. Click **Open**.
6. Type in a name for Target Instance. Note that this instance name should not be the same as any of the existing IP instances in the current Clarity Designer project.
7. Click **Import**. The module's dialog box opens.
8. In the dialog box, choose desired options.

   For more information about the options, click **Help**. You may also check the **About** tab in the Clarity Designer window for links to technical notes and user guides. The IP may come with additional information.

   As the options change, the schematic diagram of the module changes to show the ports and the device resources the module needs.
9. Click **Configure**.

This chapter provides application support information for the SGMII and Gb Ethernet PCS IP Core.

SGMII-to-(G)MII Reference Design

This section describes the operation of an SGMII-to-(G)MII bridge, using Lattice's SGMII and Gb Ethernet PCS IP Core. This bridge demonstrates how the IP core can be used in a bridging application and as a starting point for developing your own custom bridge design.

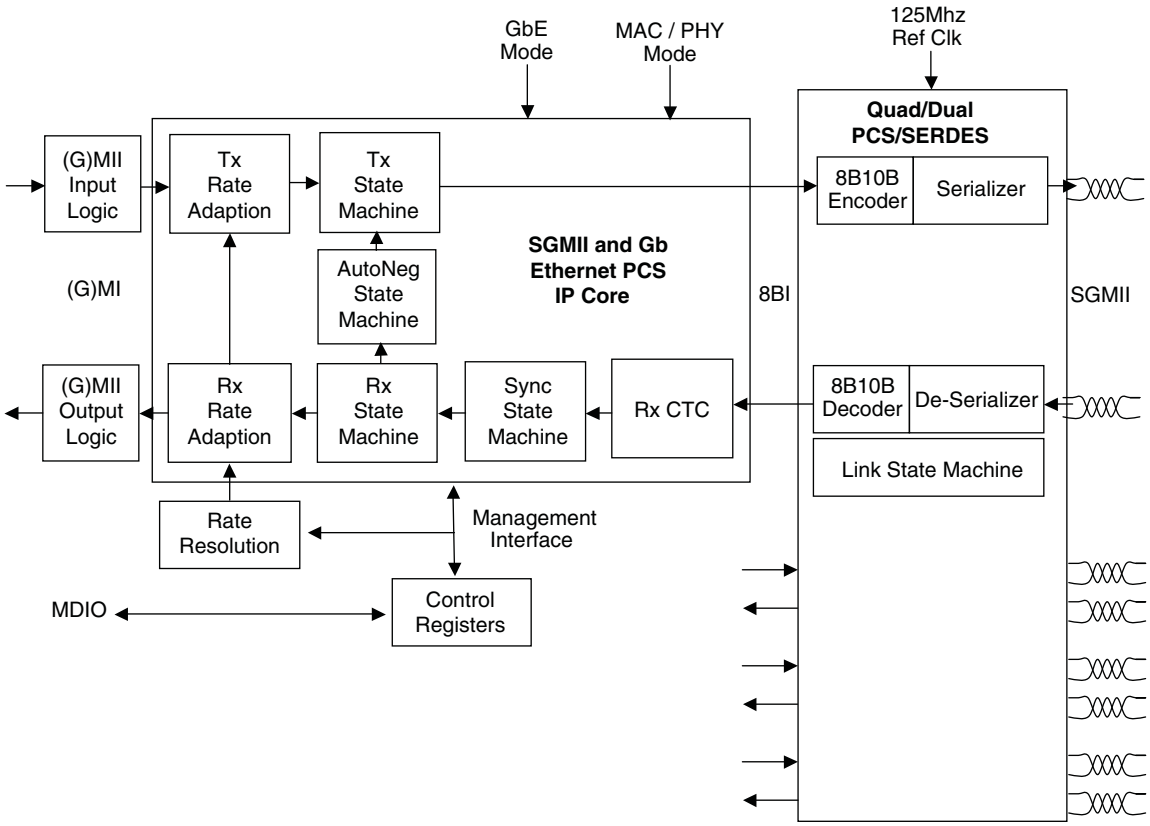
Features

- Tri-Speed (G)MII Interface operating at 10 Mbs, 100 Mbs, or 1G bps
- Differential, CML, SGMII Port operating at 1.25 Gbps
- Pin selectable SGMII MAC/PHY mode
- Dynamically switchable SGMII/1000BaseX PCS mode
- Management registers accessible through MDIO

Detailed Description

The block diagram of the SGMII-to-(G)MII Bridge is shown in Figure 5-1.

Figure 5-1. Bridge with SMII Control Interface



The SGMII and Gb Ethernet PCS IP Core

The IP core performs the data channel encoding/decoding, auto-negotiation, and rate adaptation functions described earlier in the main section of this document.

PCS/SERDES

This block is an embedded circuit function within the FPGA architecture. It provides this application with a 1.25Gbps SERDES function, and 8B10B data encoder/decoder functions.

Rate Resolution

This block is used to drive the operational_rate port of the IP core, thereby controlling the data flow rate of the rate adaptation blocks within the IP core. Operational_Rate is a 2-bit vector, where 10=1 Gbps; 01=100 Mbps; 00=10 Mbps. Table 5-1 shows how rates are resolved by this circuit block.

Table 5-1. Rate Resolution Function

GBE Mode 0=SGMII PCS 1=1000BaseX PCS	SGMII Mode 0=MAC 1=PHY	Auto Negotiation Enable	IP Core's Advertised Rate (sent)	IP Core's Link Partner Rate (recv)	Rate when AutoNeg Disabled	Rate Resolution Output
x	x	0	x	x	DIS[1:0]	DIS[1:0]
0	0	1	x	LP[1:0]		LP[1:0]
0	1	1	AR[1:0]	x		AR[1:0]
1	x	x	x	x		'10' (1 Gbps)

Control Registers

The control register block contains five of the management registers specified in IEEE 802.3, Clause 37 – Control, Status, Auto Negotiation Advertisement, Link Partner Ability, Auto Negotiation Expansion, and Extended Status. Note that the register set implementation varies, dependent upon which (G)MII interface style is used to generate the IP core within ipExpress. For the classic-GMII style, the register set is read/written through a serial management interface (SMI) or MDIO as specified in IEEE 802.3, Clause 22. For the TSMAC-Easy-Connect GMII style, the register set is read/written through the parallel Host-Bus interface – the same parallel interface that the Lattice TSMAC uses to access TSMAC IP core registers.

The register map addresses and register descriptions for both implementation styles are shown below.

Table 5-2. Register Map for SMI Registers

Address	Mode	Register Name
0x0	R/W	Control Register
0x1	R	Status Register
0x4	R/W	Advertised Ability
0x5	R	Link Partner Ability
0x6	R	Auto Negotiation Expansion Register
0xF	R	Extended Status Register

Table 5-3. Description of Control Register (Address 0x0 - SMI)

Data Bit	Name	Mode	Description
15	Reset	R/W	1=Reset (self clearing) 0=normal
14	Loopback	R/W	1=Loopback 0= normal
13	Speed Selection[0]	R/W or Stuck-at-0	Combined with bit D6 to form 2-bit vector Speed Selection [1:0] = 11 = reserved Speed Selection [1:0] = 10 = 1Gbps Speed Selection [1:0] = 01 = 100Mbps Speed Selection [1:0] = 00 = 10Mbps In GbE Mode, Speed Selection [1:0] is stuck at “10” = 1Gbps. In SGMII Mode, the Speed Selection[1:0] bits can be written to any value. However, the specified speed only affects the (G)MII data rate when auto-negotiation is disabled. Otherwise, these control bits have no effect. The control of (G)MII data rate when auto-negotiation is enabled is managed by bits [11:10] of the advertised ability vector.
12	Auto Neg Enable	R/W	1=Enable, 0=Disable
11	Power Down	R/W	1=Power Down, 0=Power Up
10	Isolate	R/W	1=Isolate, 0=Normal
9	Restart Auto Neg	R/W	1=Restart (self clearing), 0=Normal
8	Duplex Mode	R/W	1=Full Duplex, 0=Half Duplex Note that the setting of this bit has no effect on the operation of the PCS channel. The PCS channel is always a 4-wire interface with separate TX and RX data paths.
7	Collision Test	Stuck-at-0	1=Enable Test, 0=Normal
6	Speed Selection[1]	R/W or Stuck-at-1	Combined with bit D13 to form the 2-bit vector Speed Selection [1:0]
5	Unidirectional	R/W	1=Unidirectional, 0=Normal
4	reserved	Stuck-at-0	
3	reserved	Stuck-at-0	
2	reserved	Stuck-at-0	
1	reserved	Stuck-at-0	
0	reserved	Stuck-at-0	

Table 5-4. Description of Status Register (Address 0x1 - SMI)

Data Bit	Name	Mode	Description
15	100BASE-T4	Stuck-at-0	0=not supported
14	100BASE-X Full Duplex	Stuck-at-0	0=not supported
13	100BASE-X Half Duplex	Stuck-at-0	0=not supported
12	10 Mbps Full Duplex	Stuck-at-0	0=not supported
11	10 Mbps Half Duplex	Stuck-at-0	0=not supported
10	100BASE-T2 Full Duplex	Stuck-at-0	0=not supported
9	100BASE-T2 Half Duplex	Stuck-at-0	0=not supported
8	Extended Status	Stuck-at-1	1=supported
7	Unidirectional Capability	R	1=supported, 0=not supported
6	MF Preamble Suppress	Stuck-at-0	0=not supported
5	Auto Neg Complete	R	1=complete, 0=not complete
4	Remote Fault	Stuck-at-0	0=not supported
3	Auto Neg Ability	Stuck-at-1	1=supported
2	Link Status	R	1=Link Up, 0=Link Down (Latch-on-zero, Clear-on-read)

Data Bit	Name	Mode	Description
1	Jabber Detect	Stuck-at-0	0=not supported
0	Extended Capability	Stuck-at-0	0=not supported

Table 5-5. Description of Advertised Ability Register (Address 0x4 - SMI)

Data Bit	PCS=GbE		PCS=SGMII-PHY-Side		PCS=SGMII-MAC-Side	
	Mode	Name	Mode	Name	Mode	Name
15	R/W	Next Page	R/W	Link Status	R	0
14	R/W	Acknowledge	R/W	Acknowledge	R	1
13	R/W	Remote Fault[1]	R/W	0	R	0
12	R/W	Remote Fault[0]	R/W	Duplex Mode	R	0
11	R/W	0	R/W	Speed[1]	R	0
10	R/W	0	R/W	Speed[0]	R	0
9	R/W	Pause[1]	R/W	0	R	0
8	R/W	Pause[0]	R/W	0	R	0
7	R/W	Half Duplex	R/W	0	R	0
6	R/W	Full Duplex	R/W	0	R	0
5	R/W	0	R/W	0	R	0
4	R/W	0	R/W	0	R	0
3	R/W	0	R/W	0	R	0
2	R/W	0	R/W	0	R	0
1	R/W	0	R/W	0	R	0
0	R/W	0	R/W	1	R	1

Table 5-6. Description of Link Partner Ability Register (Address 0x5 - SMI)

Data Bit	PCS=GbE		PCS=SGMII	
	Mode	Name	Mode	Name
15	R	Next Page	R	Link Status
14	R	Acknowledge	R	Acknowledge
13	R	Remote Fault[1]	R	0
12	R	Remote Fault[0]	R	Duplex Mode
11	R	0	R	Speed[1]
10	R	0	R	Speed[0]
9	R	Pause[1]	R	0
8	R	Pause[0]	R	0
7	R	Half Duplex	R	0
6	R	Full Duplex	R	0
5	R	0	R	0
4	R	0	R	0
3	R	0	R	0
2	R	0	R	0
1	R	0	R	0
0	R	0	R	1

Table 5-7. Description of Auto Negotiation Expansion Register (Address 0x6 - SMI)

Data Bit	Name	Mode	Description
15:3	Reserved	Stuck-at-0	Reserved
2	Next Page Able	Stuck-at-0	0=not supported
1	Page Received	R	1=received, 0=not received latch on 1, clear on read
0	Reserved	Stuck-at-0	Reserved

Table 5-8. Description of Extended Status Register (Address 0xF - SMI)

Data Bit	Name	Mode	Description
15	1000BASE-X Full Duplex	Stuck-at-1	1 = Supported
14	1000BASE-X Half Duplex	Stuck-at-0	0 = Not supported
13	1000BASE-T Full Duplex	Stuck-at-0	0 = Not supported
12	1000BASE-T Half Duplex	Stuck-at-0	0 = Not supported
11:0	Reserved	Stuck-at-0	Reserved

Table 5-9. Register Map for Host Bus Registers

Address	Mode	Register Name
0x0	R/W	Control Register - Low
0x1	R/W	Control Register - High
0x2	R	Status Register - Low
0x3	R	Status Register - High
0x8	R/W	Advertised Ability - Low
0x9	R/W	Advertised Ability - High
0xA	R	Link Partner Ability - Low
0xB	R	Link Partner Ability - High
0xC	R	Auto Negotiation Expansion Register - Low
0xD	R	Auto Negotiation Expansion Register - High
0x1E	R	Extended Status Register - Low
0x1F	R	Extended Status Register - High

Table 5-10. Description of Control Register

Data Bit	Name	Mode	Description
Control Register High, Address 0x1 - Host Bus			
7	Reset	R/W	1 = Reset (self clearing), 0 = Normal
6	Loopback	R/W	1 = Loopback, 0 = Normal
5	Speed Selection[0]	R/W or Stuck-at-0	Combined with bit D6 in Control Register Low to form a 2-bit vector: Speed Selection [1:0] = 11 = Reserved Speed Selection [1:0] = 10 = 1Gbps Speed Selection [1:0] = 01 = 100Mbps Speed Selection [1:0] = 00 = 10Mbps In GbE Mode, Speed Selection [1:0] is stuck at “10” = 1Gbps In SGMII Mode, the Speed Selection[1:0] bits can be written to any value. However, the specified speed only affects the (G)MII data rate when auto-negotiation is disabled. Otherwise, these control bits have no effect. The control of (G)MII data rate when auto-negotiation is enabled is managed by bits [11:10] of the advertised ability vector.
4	Auto Neg Enable	R/W	1=Enable, 0=Disable
3	Power Down	R/W	1=Power Down, 0=Power Up
2	Isolate	R/W	1=Isolate, 0=Normal
1	Restart Auto Neg	R/W	1=Restart (self clearing), 0=Normal
0	Duplex Mode	R/W	1=Full Duplex, 0=Half Duplex Note that the setting of this bit has no effect on the operation of the PCS channel. The PCS channel is always a 4-wire interface with separate TX and RX data paths.
Control Register Low Address 0x0 - Host Bus			
7	Collision Test	Stuck-at-0	1=Enable Test 0=Normal
6	Speed Selection[1]	R/W or Stuck-at-1	Combined with bit D5 in Control Register High to form the 2-bit vector Speed Selection [1:0]
5	Unidirectional	R/W	1=Unidirectional, 0=Normal
4	Reserved	Stuck-at-0	
3	Reserved	Stuck-at-0	
2	Reserved	Stuck-at-0	
1	Reserved	Stuck-at-0	
0	Reserved	Stuck-at-0	

Table 5-11. Description of Status Register

Data Bit	Name	Mode	Description
Status Register High Address 0x3 - Host Bus			
7	100BASE-T4	Stuck-at-0	0=not supported
6	100BASE-X Full Duplex	Stuck-at-0	0=not supported
5	100BASE-X Half Duplex	Stuck-at-0	0=not supported
4	10 Mbps Full Duplex	Stuck-at-0	0=not supported
3	10 Mbps Half Duplex	Stuck-at-0	0=not supported
2	100BASE-T2 Full Duplex	Stuck-at-0	0=not supported
1	100BASE-T2 Half Duplex	Stuck-at-0	0=not supported
0	Extended Status	Stuck-at-1	1=supported
Status Register Low Address 0x2 - Host Bus			
7	Unidirectional Capability	R	1=supported, 0=not supported

Data Bit	Name	Mode	Description
6	MF Preamble Suppress	Stuck-at-0	0=not supported
5	Auto Neg Complete	R	1=complete, 0=not complete
4	Remote Fault	Stuck-at-0	0=not supported
3	Auto Neg Ability	Stuck-at-1	1=supported
2	Link Status	R	1=Link Up, 0=Link Down Latch-on-zero Clear-on-read
1	Jabber Detect	Stuck-at-0	0=not supported
0	Extended Capability	Stuck-at-0	0=not supported

Table 5-12. Description of Advertised Ability Register

Data Bit	PCS=GbE		PCS=SGMII-PHY-Side		PCS=SGMII-MAC-Side	
	Mode	Name	Mode	Name	Mode	Name
Advertised Ability Register High, Address 0x9 - Host Bus						
7	R/W	Next Page	R/W	Link Status	R	0
6	R/W	Acknowledge	R/W	Acknowledge	R	1
5	R/W	Remote Fault[1]	R/W	0	R	0
4	R/W	Remote Fault[0]	R/W	Duplex Mode	R	0
3	R/W	0	R/W	Speed[1]	R	0
2	R/W	0	R/W	Speed[0]	R	0
1	R/W	Pause[1]	R/W	0	R	0
0	R/W	Pause[0]	R/W	0	R	0
Advertised Ability Register Low, Address 0x8 - Host Bus						
7	R/W	Half Duplex	R/W	0	R	0
6	R/W	Full Duplex	R/W	0	R	0
5	R/W	0	R/W	0	R	0
4	R/W	0	R/W	0	R	0
3	R/W	0	R/W	0	R	0
2	R/W	0	R/W	0	R	0
1	R/W	0	R/W	0	R	0
0	R/W	0	R/W	1	R	1

Table 5-13. Description of Link Partner Ability Register

Data Bit	PCS=GbE		PCS=SGMII	
	Mode	Name	Mode	Name
Link Partner Ability Register High, Address 0xB - Host Bus				
7	R	Next Page	R	Link Status
6	R	Acknowledge	R	Acknowledge
5	R	Remote Fault[1]	R	0
4	R	Remote Fault[0]	R	Duplex Mode
3	R	0	R	Speed[1]
2	R	0	R	Speed[0]
1	R	Pause[1]	R	0
0	R	Pause[0]	R	0
Link Partner Ability Register Low, Address 0xA - Host Bus				
7	R	Half Duplex	R	0
6	R	Full Duplex	R	0

Data Bit	PCS=GbE		PCS=SGMII	
	Mode	Name	Mode	Name
5	R	0	R	0
4	R	0	R	0
3	R	0	R	0
2	R	0	R	0
1	R	0	R	0
0	R	0	R	1

Table 5-14. Description of Auto Negotiation Expansion Register

Data Bit	Name	Mode	Description
Auto Negotiation Expansion Register High, Address 0xD - Host Bus			
7:0	Reserved	Stuck-at-0	Reserved
Auto Negotiation Expansion Register High, Address 0xC - Host Bus			
7:3	Reserved	Stuck-at-0	Reserved
2	Next Page Able	Stuck-at-0	0=not supported
1	Page Received	R	1=received, 0=not received Latch on 1, clear on read
0	Reserved	Stuck-at-0	Reserved

Table 5-15. Description of Extended Status Register

Data Bit	Name	Mode	Description
Extended Status Register High, Address 0x1F - Host Bus			
7	1000BASE-X Full Duplex	Stuck-at-1	1=supported
6	1000BASE-X Half Duplex	Stuck-at-0	0=not supported
5	1000BASE-T Full Duplex	Stuck-at-0	0=not supported
4	1000BASE-T Half Duplex	Stuck-at-0	0=not supported
3:0	Reserved	Stuck-at-0	Reserved
Extended Status Register Low, Address 0x1E - Host Bus			
7:0	Reserved	Stuck-at-0	Reserved

(G)MII I/O Logic

The I/O logic consists of I/O flip-flops and buffers for moving (G)MII data into and out of the FPGA.

Signal Descriptions

Table 5-16 shows a detailed listing of all the reference design I/O signals.

Table 5-16. Input and Output Signals for (G)MII-to-SGMII Bridge

Signal Name	I/O	Description
125MHz Reference Clock Signals		
in_clk_ref	In	**Inbound Reference Clock** - 125MHz clock source for the transmit data path. It is used by the GMII side of the Tx rate-adapt logic, the tx state machine, the 8b10b encoder, and the SERDES serializer.
out_clk_ref	In	**Outbound Reference Clock** - 125MHz clock source for the receive data path. It is used by the GMII side of the Rx rate adapt logic, the Rx state machine, the Sync state machine, 8b10b decoder, and the SERDES de-serializer.
MII Signals		
data_in_mii[7:0]	In	**Inbound MII Data** - Incoming (G)MII data. It can be either MII data (10 Mbps or 100 Mbps) or GMII data (1Gbps). When used for MII data, only bits 3:0 are valid. Bits 7:4 should be tied to fixed-state logic levels. When used for GMII data, all eight bits are valid.
en_in_mii	In	**Inbound MII Enable** - Active high signal, asserts when incoming data is valid.
err_in_mii	In	**Inbound MII Error** - Active high signal, used to denote transmission errors and carrier extension on incoming (G)MII data port.
data_out_mii[7:0]	Out	**Outbound MII Data** - Outgoing (G)MII data. It can be either MII data (10 Mbps or 100 Mbps) or GMII data (1Gbps). When used for MII data, only bits 3:0 are valid. Bits 7:4 are always driven to logic-low levels. When used for GMII data, all eight bits are valid.
dv_out_mii	Out	**Outbound MII Data Valid** - Active high signal, asserts when outgoing data is valid.
err_out_mii	Out	**Outbound MII Error** - Active high signal, used to denote transmission errors and carrier extension on outgoing (G)MII data port.
col_out_mii	Out	**Outbound MII Collision Detect** - Active high signal, asserts when data valid signals are simultaneously active at the transmit and receive data paths of the GMII port of the Lattice IP core.
crs_out_mii	Out	**Outbound MII Carrier Sense Detect** - Active high signal, asserts when data valid signal is high at receive data path of the GMII port of the Lattice IP core.
SERDES Signals		
HDOUTP0	Out	**Outbound SGMII(P) Signal** - P-sense portion of differential SERDES transmit signal.
HDOUTN0	Out	**Outbound SGMII(N) Signal** - N-sense portion of differential SERDES transmit signal.
HDINP0	In	**Inbound SGMII(P) Signal** - P-sense portion of differential SERDES receive signal.
HDINN0	In	**Inbound SGMII(N) Signal** - N-sense portion of differential SERDES receive signal.
Miscellaneous Signals		
rst_n	In	**Reset Active low global reset**
sgmii_mode	In	**SGMII Mode** - Controls the behavior of the auto-negotiation process when the IP core operates in the SGMII PCS mode. 0=operates as MAC-side entity 1=operates as PHY-side entity
gbe_mode	In	**GbE Mode** - Controls IP core's PCS function. 1 = operates as 1000BaseX PCS. 0 = operates as SGMII PCS.
Implementation Specific Signals for Classic (G)MII Configuration		
MII Clock Signals		
in_clk_mii	In	**Inbound MII Clock.** MII clock source for the transmit data path. It is used by the MII input logic and the MII side of the Tx rate adapt logic. The frequency of this clock is dependent on the type of data driven into the MII data port. For 10Mbps data, the clock frequency is 2.5MHz; for 100Mbps data, the clock frequency is 25MHz; for 1Gbps data, the clock frequency is 125MHz.
out_clk_mii	In	**Outbound MII Clock.** MII clock source for the receive data path. It is used by the MII output logic and the MII side of the rx rate adapt logic. The frequency of this clock is dependent on the type of data driven into the MII data port. For 10Mbps data, the clock frequency is 2.5MHz; for 100Mbps data, the clock frequency is 25MHz; for 1Gbps data, the clock frequency is 125MHz.

Table 5-16. Input and Output Signals for (G)MII-to-SGMII Bridge (Continued)

Signal Name	I/O	Description
Management Register Interface Signals		
mdc	In	**Management Data Clock.** Clock source for the serial management interface. The IEEE 802.3 specification (clause 22) dictates that the maximum frequency for this clock is 2.5 MHz. However, in this bridge design the clock can be conservatively driven as fast as 50 MHz.
mdio	In/Out	**Management Data Input/Output.** Bi-directional signal used to read/write management registers.
port_id[4:0]	In	**Port Identification Address.** Used to define the binary address of this management node. The value used here corresponds to the PHY-ADD portion of the management frame format (specified in IEEE 802.3, clause 22).
Implementation Specific Signals TSMAC Easy Connect (G)MII Configuration		
MII Clock Signals		
(in_clk_ref)	In	**Inbound Reference Clock –** In the TSMAC Easy Connect configuration, there is no separate transmit clock provided for the (G)MII port. Instead, the “in_clk_ref” signal (125 MHz) listed above in this table is used to clock data and control signals into the transmit (G)MII port.
in_ce_source	Out	**Inbound Clock Enable Source –** This signal is used in combination with the in_clk_ref clock to regulate the flow of transmit (G)MII data. The signal is generated by the transmit rate adaptation block. This clock enable should be tied to the transmit section of the Lattice MAC that sends transmit Ethernet frames to the SGMII and Gb Ethernet PCS IP core. This clock enable should also be tied to the clock enable “sink” of the SGMII and Gb Ethernet PCS IP core. This clock enable’s behavior is controlled by the setting of the operational_rate pins of the IP core. For 1Gbps operation, the clock enable is constantly high. For 100 Mbps operation, the clock enable is high for one-out-of-ten 125 MHz clock cycles. For 10 Mbps operation, the clock enable is high for one-out-of-one-hundred 125 MHz clock cycles.
in_ce_sink	In	**Inbound Clock Enable Sink** - This signal is used in combination with the in_clk_ref clock to regulate the flow of transmit (G)MII data. When the in_ce_sink signal is high and the in_clk_ref edge rises, (G)MII data is sampled.
(out_clk_ref)	In	**Outbound Reference Clock -** In the TSMAC Easy Connect configuration, there is no separate receive clock provided for the (G)MII port. Instead, the “out_clk_ref” signal (125 MHz) listed above in this table is used to clock data and control signals out of the receive (G)MII port.
out_ce_source	Out	**Outbound Clock Enable Source** - This signal is similar to the in_ce_source signal described above, except that it is used for the receive data path.
out_ce_sink	In	**Outbound Clock Enable Sink** - This signal is used in combination with the out_clk_ref clock to regulate the flow of receive (G)MII data. When the out_ce_sink signal is high and the out_clk_ref edge rises, (G)MII data is launched.
Management Interface Signals[1]		
hclk	In	**Host Clock.** This is the Host Bus clock, and is used to clock the Host Bus interface.
hcs_n	In	**Host Chip Select.** This is an active low signal used to select management registers for read/write operations.
hwrite_n	In	**Host Write.** This active low signal is used to write data to the selected register.
haddr[5:0]	In	**Host Address.** This addresses one of the management registers.
hdatain[7:0]	In	**Host Data Input.** The CPU writes to the management registers through this data bus.
hdataout[7:0]	Out	**Host Data Output.** The CPU reads the management registers through this data bus.
hready_n	Out	**Host Ready. T**his is an active low signal used to indicate the end of transfer. For write operations, hready_n is asserted after data is accepted (written). For read operations, hready_n is asserted when data on the hdataout bus is ready to be driven out to the CPU.

1. Note for the TSMAC Easy Connect (G)MII configuration, the register interface implementation is identical to the host interface used for the Lattice TSMAC. Therefore, the management registers of the SGMII and Gb Ethernet PCS IP core are accessed in the identical manner that TSMAC registers are accessed. Consult IPUG51, *Tri-Speed Ethernet MAC User’s Guide*, for host interface timing details.

Core Validation

The Lattice SGMII and Gb Ethernet PCS IP core has been validated through interoperability tests with an external Marvell 8E1111 PHY device. An external ethernet protocol generator/analyzer, such as the Spirent Smartbits tester, was used to exercise auto-negotiation, and drive/monitor ethernet traffic through the Marvel PHY and SGMII/GbE IP core test circuit.

Refer to the following Lattice interoperability technical note:

- TN1197 - *LatticeECP3 Marvell SGMII Physical/MAC Layer Interoperability*

Chapter 7:

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.

E-mail Support

techsupport@latticesemi.com

Local Support

Contact your nearest Lattice Sales Office.

Internet

www.latticesemi.com

References

The following documents provide more technical information regarding this IP core:

- Serial-GMII Specification, Revision 1.8, November 2, 2005, Cisco Systems
- IEEE 802.3-2002 Specification

LatticeECP3

- HB1009, LatticeECP3 Family Handbook
- TN1176, LatticeECP3 SERDES/PCS Usage Guide
- TN1197, LatticeECP3 Marvell SGMII Physical/MAC Layer Interoperability
- IPUG51, Tri-Speed Ethernet MAC User's Guide

ECP5

- HB1012, ECP5 Family Handbook

Revision History

Date	Document Version	IP Core Version	Change Summary
September 2006	01.0	1.0	Initial release.
November 2006	01.1	2.0	Added support for LatticeECP2M family.
April 2007	01.2	2.0	Added rx_compensation_err signal to the SGMII IP Core Input and Output Signals table (8-Bit Code Group Signals section).
August 2008	01.3	3.0	Updated to include changes introduced by release 3.0 of IP core.
April 2009	01.4	3.1	Added support for LatticeECP3 family.
June 2009	01.5	3.2	Updated to include information on updating the PCS and implementing multiple PCS blocks.
September 2009	01.6	3.2	Corrected description for gbe_mode signal in the Input and Output Signals for (G)MII-to-SGMII Bridge table.
June 2010	01.7	3.3	Added support for Diamond software.
			Divided the document into chapters and added table of contents.
			Added Table 1-1 in Chapter 1.
			Added Figure 2-3.
			Updated Table 5-16.
			Added Chapter 6, “Core Validation.”
August 2010	01.8	3.3	Updated “Soft Receive Clock Tolerance Compensation (CTC) Circuit” on page 8.
			Added “Guidelines for Calculating Static CTC FIFO Thresholds ” on page 18.
			Updated “Using CTC with the SGMII IP Core” on page 24.
March 2011	01.9	3.4	Update document for new IP core version 3.4.
October 2011	02.0	3.4	Updated “IP Core Signal Ports Associated with Auto-Negotiation” on page 11.
April 2014	02.1	4.0	Added support for Diamond 3.2.
			Added support for ECP5 device family.
			Removed references to LatticeECP2M™ and LatticeSC™ device families.
			Updated corporate logo.
			Updated “Lattice Technical Support” on page 44.

Resource Utilization

This appendix gives resource utilization information for Lattice FPGAs using the SGMII and Gb Ethernet PCS core. The IP configurations shown in this chapter were generated using the IPexpress software tool. IPexpress is the Lattice IP configuration utility, and is included as a standard feature of the Diamond design tools. Details regarding the usage of IPexpress can be found in the IPexpress and Diamond help systems. For more information on the Diamond design tools, visit the Lattice web site at: www.latticesemi.com/software.

LatticeECP3 FPGAs

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

Configuration				SLICEs	LUTs	Registers	EBRs	f_{MAX}[2] (MHz)
GMII Style	RX CTC Mode	FIFO Low Threshold	FIFO High Threshold					
Classic	None	—	—	747	877	898	0	125
Classic	Static	16	32	839	1000	1007	1	125
Easy Connect	Static	240	260	709	848	851	1	125
Easy Connect	Dynamic	—	—	729	869	882	1	125

1. Performance and utilization data are generated targeting an LFE3-70EA-7FN484C device using Lattice Diamond 3.2 and Synplify Pro I.2013.09L-SP1 software. Performance may vary when using a different software version or targeting a different device density or speed grade within the LatticeECP3 family.
2. The SGMII requires operation at 125 MHz, therefore a higher frequency is not stated. However this core can easily attain frequencies above 140 MHz in a LatticeECP3 speed grade 7 device.

Supplied Netlist Configurations

The Ordering Part Number (OPN) for the SGMII and Gb Ethernet PCS core targeting LatticeECP3 devices is GBE-SGMII-E3-U1.

ECP5 FPGAs

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

Configuration				SLICEs	LUTs	Registers	EBRs	f_{MAX}[2] (MHz)
GMII Style	RX CTC Mode	FIFO Low Threshold	FIFO High Threshold					
Classic	None	—	—	750	879	898	0	125
Classic	Static	16	32	837	1002	1007	1	125
Easy Connect	Static	240	260	709	851	851	1	125
Easy Connect	Dynamic	—	—	745	897	882	1	125

1. Performance and utilization data are generated targeting an LFE5UM-85F-7MG756C device using Lattice Diamond 3.2 and Synplify Pro I.2013.09L-SP1 software. Performance may vary when using a different software version or targeting a different device density or speed grade within the ECP5 family.
2. The SGMII requires operation at 125 MHz, therefore a higher frequency is not stated. However this core can easily attain frequencies above 140 MHz in a ECP5 speed grade 7 device.

Supplied Netlist Configurations

The Ordering Part Numbers (OPNs) for the SGMII and Gb Ethernet PCS core targeting ECP5 devices are GBE-SGMII-E5-U and GBE-SGMII-E5-UT.

X-ON Electronics

Largest Supplier of Electrical and Electronic Components

Click to view similar products for Development Software ***category:***

Click to view products by Lattice ***manufacturer:***

Other Similar products are found below :

RAPPID-560XBSW RAPPID-567XFSW DG-ACC-NET-CD SRP004001-01 SW006021-1NH SW163052 SYSWINEV21 Core429-SA SW500006-HPA CWP-BASIC-FL W128E13 CWP-PRO-FL SYSMACSE210L SYSMACSE203L AD-CCES-NODE-1 NT-ZJCAT1-EV4 CWA-BASIC-FL RAPPID-567XKSW CWA-STANDARD-R SW89CN0-ZCC CWA-LS-DVLPR-NL VDSP-21XX-PCFLOAT RAPPID-563XMSW IPS-EMBEDDED SWR-DRD-L-01 SDAWIR-4532-01 SYSMAC-SE201L MPROG-PRO535E AFLCF-08-LX-CE060-R21 WS02-CFSC1-EV3-UP SYSMAC-STUDIO-EIPCPLR LIB-PL-PC-N-1YR-DISKID SYSMACSE2XXL LS1043A-SWSP-PRM 1120270005 1120270006 MIKROBASIC PRO FOR FT90X (USB DONGLE) MIKROC PRO FOR AVR (USB DONGLE LICENSE) MIKROC PRO FOR FT90X (USB DONGLE) MIKROBASIC PRO FOR AVR (USB DONGLE LICEN MIKROBASIC PRO FOR FT90X MIKROC PRO FOR DSPIC30/33 (USB DONGLE LI MIKROC PRO FOR FT90X MIKROC PRO FOR PIC32 (USB DONGLE LICENSE 52202-588 MIKROPASCAL PRO FOR ARM (USB DONGLE LICE MIKROPASCAL PRO FOR FT90X MIKROPASCAL PRO FOR FT90X (USB DONGLE) MIKROPASCAL PRO FOR PIC32 (USB DONGLE LI SW006021-2H