LatticeSCM Ethernet flexiMAC MACO Core

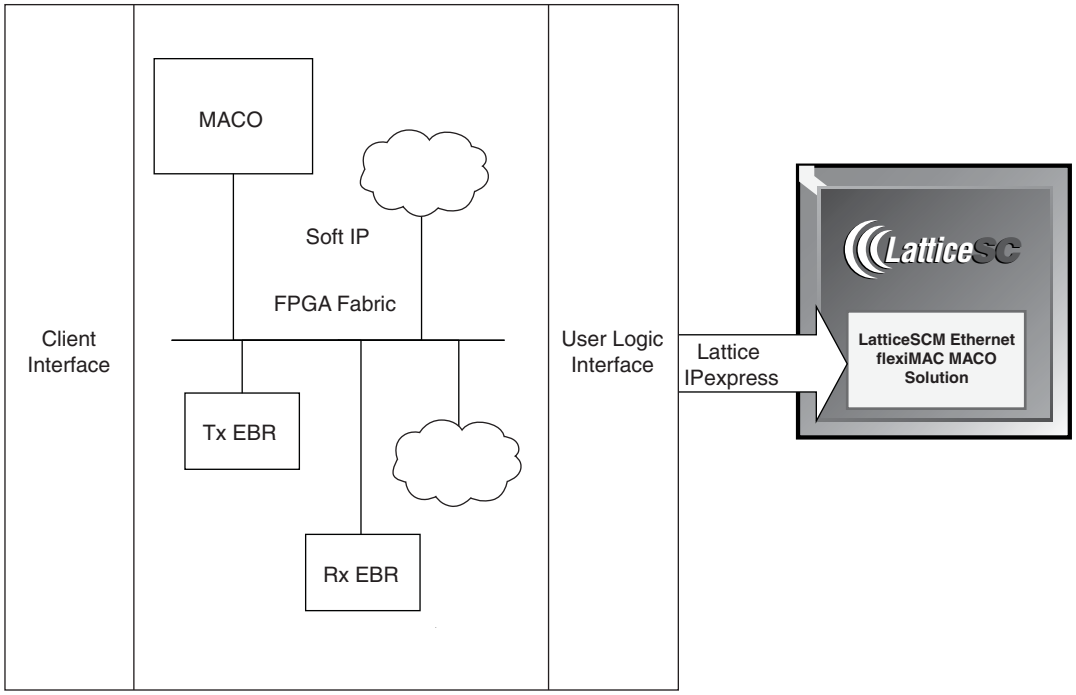
User's Guide

September 2009
ipug48_01.8

Introduction

The LatticeSCM™ Ethernet flexiMAC™ MACO™ IP core assists the FPGA designer's efforts by providing pre-tested, reusable functions that can be easily plugged in, freeing designers to focus on their unique system architecture. These blocks eliminate the need to "re-invent the wheel," by providing either an industry-standard Layer 2 flexible packet framer and parser or a Layer 1 multi-protocol functionality of the Physical Coding Sublayer (PCS) module. This proven core is optimized utilizing the LatticeSCM device's MACO architecture, resulting in fast, small cores that utilize the latest architecture to its fullest.

Figure 1. Lattice MACO IP Conceptual Diagram



Complementing the Lattice ispLEVER® software is the support to generate a number of user-customizable cores with the IPexpress™ utility. This utility helps designers input design information into a parameterized design flow. Designers can use the IPexpress software tool to help generate new configurations of this IP core. When either the 1Gb or 10Gb Ethernet IP is selected in IPexpress, the utility generates a corresponding parameter file to use with the design flow.

IPexpress, the Lattice IP configuration utility, is included as a standard feature of the ispLEVER design tools. Details regarding the use of IPexpress can be found in the IPexpress and ispLEVER online Help systems. For more information on the ispLEVER design tools, visit the Lattice web site at www.latticesemi.com/software.

The LatticeSCM Ethernet flexiMAC core is a flexible packet framer and parser that can implement Layer2 (data link layer or MAC) functionality for various standards. The flexiMAC functionality complements the Layer1 (physical layer) multi-protocol functionality of the Physical Coding Sublayer (PCS) also implemented in MACO. This yields a complete Layer1/Layer2 solution for 1/10Gb Ethernet standards. flexiMAC is targeted at the LatticeSCM family of devices.

For more information on these and other Lattice products, refer to the Lattice web site at www.latticesemi.com.

This user's guide explains the functionality of the LatticeSCM Ethernet flexiMAC core and how it can be used to provide a full Layer1/Layer2 Ethernet solution.

The LatticeSCM Ethernet flexiMAC core comes with the documentation and files listed below:

- Lattice gate level netlist
- Secured RTL simulation model
- Core instantiation template

Features

- 1Gb Ethernet MAC (full duplex only) standard support.
- 10Gb Ethernet MAC standard support including clauses 46.3.3 (Error and Fault Signalling) of IEEE Draft P802.3ae (also known as reconciliation layer)
- IP provided in encrypted netlist
- Simulation models and test benches available for free evaluation

Getting Started

Requirements to implement a MACO core in IPexpress include:

- ispLEVER version 7.0 or later
- MACO design kit
- MACO license file

For information on obtaining the above requirements, please contact your local Lattice Semiconductor sales representative.

General Description

The LatticeSCM Ethernet flexiMAC core targets a mixture of Soft IP (SIP) and Hard IP (HIP) on the LatticeSCM family of FPGAs and provides a bridging function (i.e. MAC between the client and PCS levels of the protocol stack).

The LatticeSCM Ethernet flexiMAC core is provided with implementation scripts, test benches, and documentation.

1Gb/10Gb Ethernet Functional Partitioning

The LatticeSCM Ethernet flexiMAC core implements the following transmission features:

- Appends preamble (0x55) and SFD (0xD5) to packets
- Pads packets that are less than the minimum length of 64 bytes
- Calculates and appends a CRC-32 checksum to each packet
- Enforces the minimum Inter Packet Gap (IPG) between packets
- Pauses packet transmission following reception of pause frame by Receive flexiMAC (Except the transmission of PAUSE Frames)
- Constructs and transmits pause frames when requested by the client logic
- Interfaces to FPGA EBR FIFO
- TX reconciliation layer (10Gb only)
- Generates a TX statistics vector per packet for the client logic

The LatticeSCM Ethernet flexiMAC core implements the following receiver features:

- Broadcast packet reception
- Unicast packet reception - checks the destination address of received packets to make sure that it matches the flexiMAC's address

- Multicast packet reception - hash-based multicast address packet reception (hash algorithm described later in this document)
- Checks the CRC-32 of the received packets to detect errors that occurred during transmission
- Decodes received pause frames and signal transmit flexiMAC to pause transmission
- Optionally filters PAUSE frames
- Detects control packets other than PAUSE frames
- Detects VLAN tagged packets
- Removes the pad from previously padded packets
- Interfaces to FPGA EBR FIFO
- Rx reconciliation layer (10Gb only)
- Detects protocol errors on the bus
- Generates an Rx statistics vector per packet for the client logic

Additional soft IP will be required to implement statistics gathering registers, packet counters, etc., if required. This is not part of the flexiMAC release IP (see Figure 2).

Figure 2. Ethernet SIP/HIP Partitioning

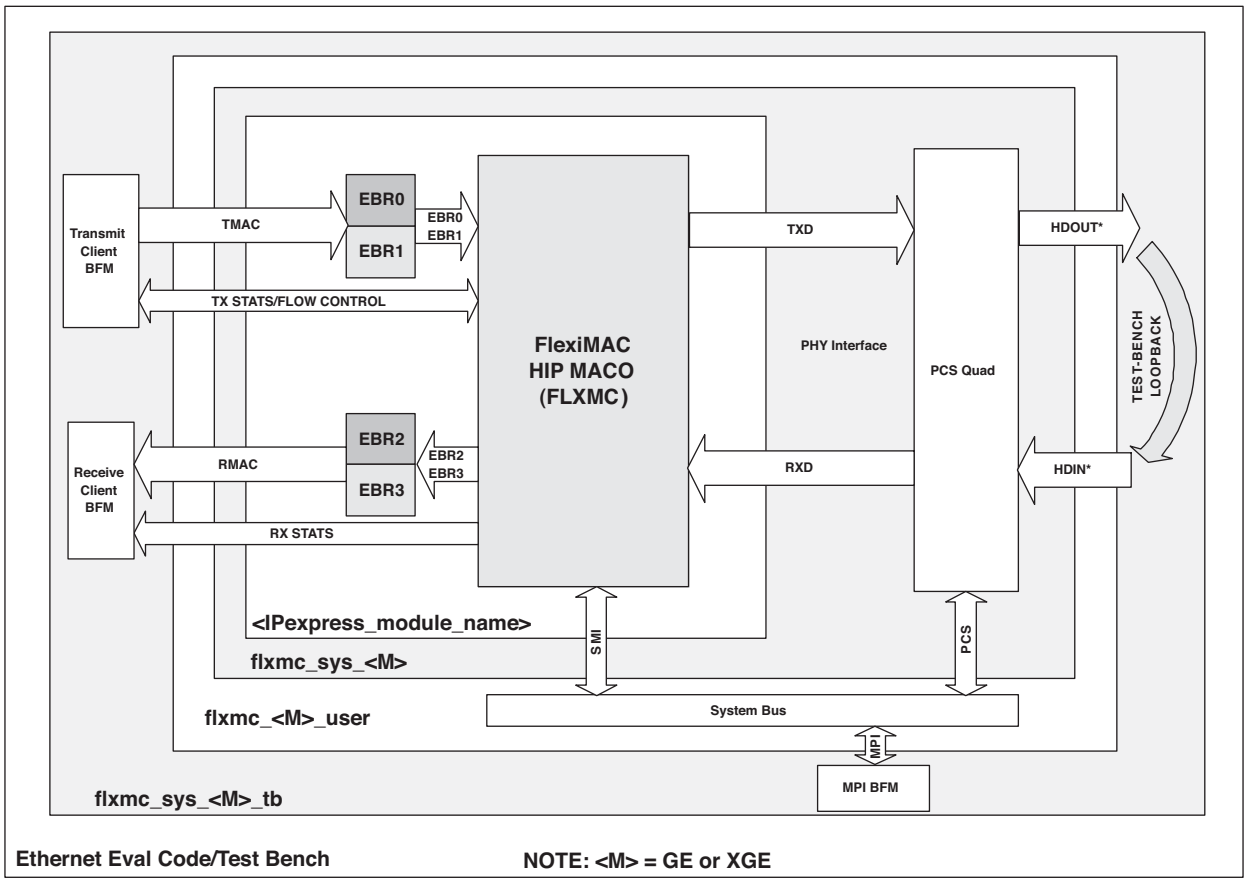


Figure 2 shows the LatticeSCM Ethernet flexiMAC core in a system context with other IP (e.g. PCS, System Bus-LatticeSCM hard IP) and user defined soft logic (e.g. client and Statistics logic). There are different levels depicted in the figure:

- <IPexpress_module_name>: this level contains the flexiMAC MACO hard IP (FLXMC) and EBR FIFOs (512 deep FIFO logic to interface to the client interface). For synthesis purposes, this level is available as a black box model, <IPexpress_module_name>_bb.v, where <IPexpress_module_name> is the module name the user selected for the core during IPexpress generation. An equivalent NGO model is available for Map, Place and Route. For simulation purposes, a different model, <IPexpress_module_name>.v (Verilog) or <IPexpress_module_name>.vhd (VHDL), is used and further expands into other behavioral simulation models (contained in flxmc_core_beh.v and other files).
- flxmc_sys_<M> (where <M>=GE or XGE): this is the complete LatticeSCM Ethernet flexiMAC core from a user perspective. This level contains, in addition to <IPexpress_module_name>, a QUAD PCS model. The QUAD PCS block (created from ispLEVER IPexpress) is either configured in 1 GE (<M>=GE) or 10 GE mode (<M>=XGE). Any user-defined solution will have to be built on top of this level.
- flxmc_<M>_user: This is a sample user solution that instantiates the LatticeSCM Ethernet flexiMAC core (flxmc_sys_<M>) and a system bus (created from ispLEVER IPexpress). This is the top level that goes through synthesis, and place and route. This system bus is generated with the following interfaces:
  - MPI master interface to a test-bench level MPI driver.
  - SMI slave interface to communicate with the FLXMC HIP SMI registers.
  - PCS slave interface to communicate with the PCS registers.
- flxmc_sys_<M>_tb: this level is used to functionally test the user evaluation design (flxmc_<M>_user) in a software simulation environment (MTI ModelSim® or Aldec® Active-HDL®). This level also contains Behavioral Functional Models (BFM) for the client receive and transmit blocks, as well as for the MPI interface driving the system bus as a master.

The next three sections describe functionally the TX, RX and Flow Control blocks of the LatticeSCM Ethernet flexiMAC core design. The focus is on the FLXMC HIP block, which contains the bulk of the Ethernet functionality.

flexiMAC TX Functional Description

Overview

The transmitter covers the data path from the EBR (client interface) through to the physical interface (i.e. to the PCS block). A summary of the basic Ethernet functionality follows:

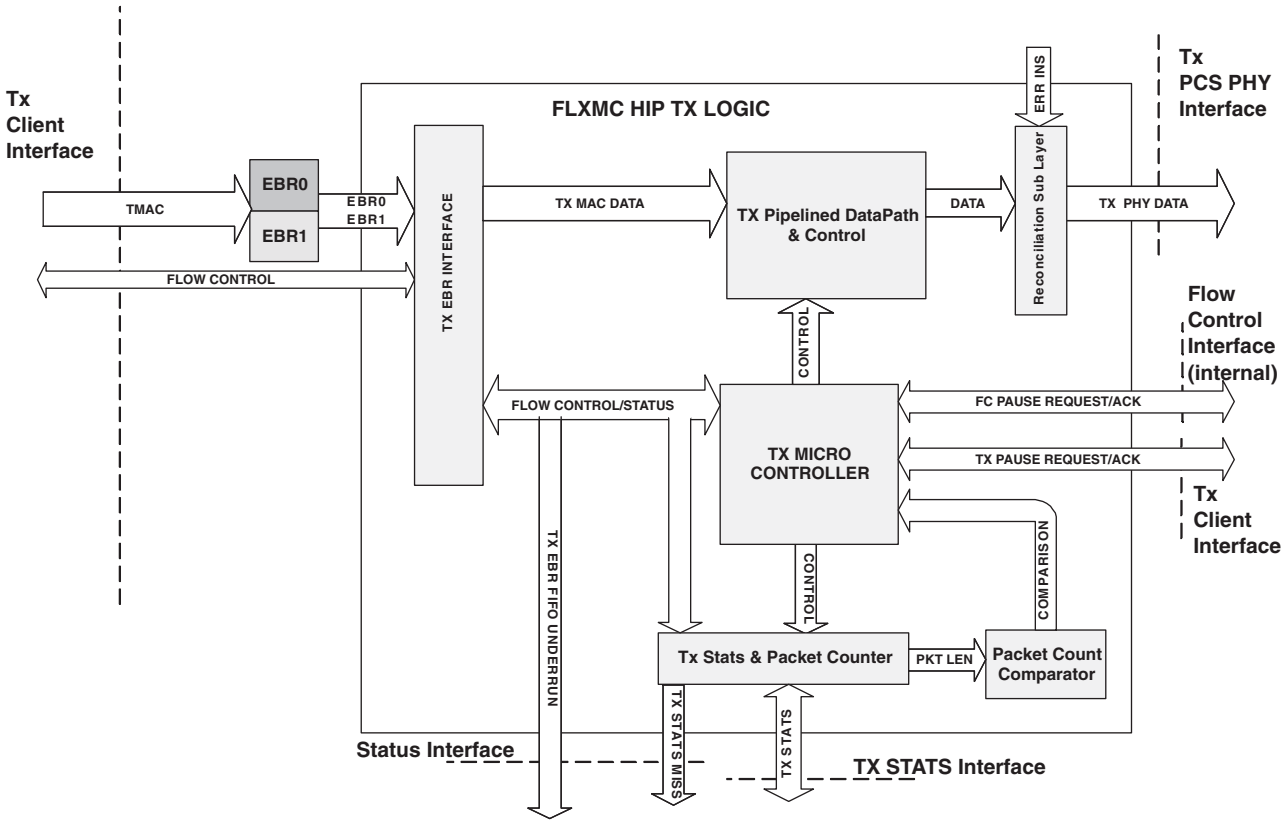
- Fetches packets from the EBR interface.
- Pads data packets less than the minimum packet length.
- Calculates and inserts the CRC-32.
- Adds framing characters (SOP / EOP).
- Enforces inter-packet gaps (IPG) rules and inserts IDLE characters between back-to-back packets in the EBR. In 10Gb Ethernet mode, it does so by adding two columns of IDLE when the terminate character (/T/) is in lane 0, and three columns of IDLE when the terminate character is in any other lane. In 1Gb Ethernet mode, it does so by inserting 15 bytes of IDLE after the last CRC byte.
- Disabled if Physical layer is down or stalled.
- Provides statistics on packet length and status of transmission.
- Stalls if Flow Control block (described later) asserts a Pause Frame.
- Constructs and transmits PAUSE frames when requested from client.

Transmit Architecture

Figure 3 shows the architecture of the TX block in the LatticeSCM Ethernet flexiMAC core.

The architecture splits between control and data path blocks. The following is a description of the major blocks inside the FLXMC HIP.

Figure 3. flexiMAC TX Architecture



TX EBR Interface Block

The EBR Interface block receives 64 bits of data from the EBR interface and multiplexes it to a 32-bit data path. It also generates a last byte position (LBP) field, which indicates the position of the last valid byte in the last (EOP) data word.

In 1Gb Ethernet mode, the data path is 8 bits wide and the interface is to a single EBR with no need for multiplexing. Please refer to the I/O Descriptions section of this document for information on the corresponding TX client interface signals.

NOTE: In 10Gb Ethernet mode any invalid bytes at the end of a packet will be overwritten with "0"s (the pad value in Ethernet). This enables the LatticeSCM Ethernet flexiMAC core to pad in 32-bit double words (DW), as the final DW of the valid data will already contain the pad value.

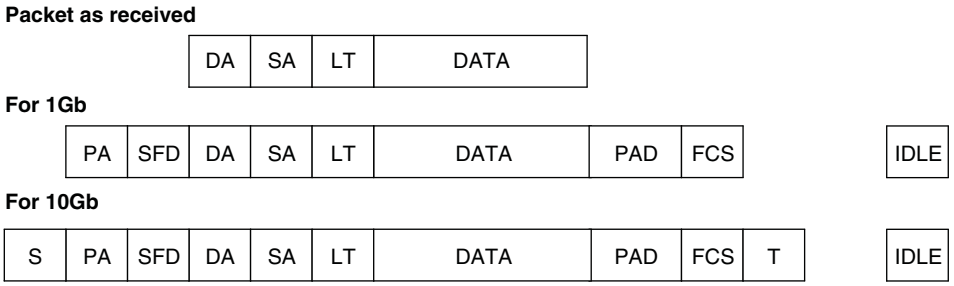
TX Pipelined Data Path and Control

The pipelined data path block performs the following functions in a series of pipelined stages:

- Padding of undersized packets
- Generation and scheduling of flow control pause frames
- Generation of the appropriate CRC to append to the end of the packet
- Data shifting to allow framing
- Protocol specific packet framing with bitstream programmable control characters
- The microcontroller generates a control word that is passed down the pipeline in order to align control to the various stages of the data pipeline

The function of the data path is to implement MAC framing and CRC checking to the packets transmitted by the MAC client. A transmitted packet is padded out to the minimum length required, if necessary, and a frame check sum (FCS) is appended. A preamble (PA) and start frame delimiter (SFD) is pre-pended. For 10Gb Ethernet, the framing is modified slightly to add the start (S) and terminate (T) characters. In between frames, IDLE characters are inserted as required. Figure 4 demonstrates framing.

Figure 4. Ethernet Mode Framing



TX CRC: The CRC is performed using the following polynomials

Table 1. CRC Polynomials for Ethernet

Protocol	CRC-32
1Gb Ethernet	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
10Gb Ethernet	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

The CRC supports 32-bit (for 10Gb E) and 8-bit (for 1Gb E) data paths. The CRC32 must also work across 16 and 24 bits as depending on the last byte position (LBP) as the last word in a packet may have 1, 2, 3 or 4 bytes valid.

Depending on how many bytes are present in the last 32-bit word, one of the 4 phases is selected, as shown in Table 2. Phases 16 and 24 are only used when an EOP has occurred and only the first 2 or 3 bytes are valid. Phase 8 is only used when an 8-bit data path is employed (i.e. 1Gb Ethernet).

Table 2. CRC Phase Selection

Mode	Phase	CRC Type
1Gb Ethernet	Phase 8	CRC 32
10Gb Ethernet (EOP = 0)	Phase 32	CRC 32
10Gb Ethernet (EOP = 1 LBP = 0)	Phase 8	CRC 32
10Gb Ethernet (EOP = 1 LBP = 1)	Phase 16	CRC 32
10Gb Ethernet (EOP = 1 LBP = 2)	Phase 24	CRC 32
10Gb Ethernet (EOP = 1 LBP = 3)	Phase 32	CRC 32

TX Framing: Based on the lane control fields from the microcontroller, the framer multiplexes data, CRC or one of the standards dependent 9-bit constants (k defined in Table 3 and Table 4) into the data path for each of the lanes.

The constants are defined below for each standard. The mapping to the mux input and the PCS code group equivalent is also shown. Along with each data word that is transmitted (e.g. to PCS), a protocol dependant control bit is also sent. For example, in 1Gb Ethernet the control bit indicates a frame is being transmitted (it is set to 0 for IDLEs).

Table 3. 10Gb Ethernet Framing

Constant	Mux Input	Control Bit Inserted	8-Bit Code Inserted by Framer	PCS Code Group Equivalent
Start	K0	1	FB	K27.7
Preamble	K1	0	55	D21.2
SFD	K2	0	D5	D21.6
TMRNT	K3	1	FD	K29.7
Idle Character	K4	1	07	K28.5 or K28.3 or K28.5

Table 4. 1Gb Ethernet Framing

Constant	Mux Input	Control Bit (tphy_txen from Section 6) Inserted	8-Bit Code Inserted by Framer	PCS Code Group Equivalent
Preamble	K1	1	55	D21.2
SFD	K2	1	D5	D21.6
Idle Character	K4	0	07	/K28.5/D5.6/ or /K28.5/D16.5/

TX Pause Frame: Note that the user client interface can also issue a pause frame on the transmit interface via the pause_req and pause_count[15:0] primary input pins. When a transmit pause frame is sent, the primary output pin pause_ack is asserted. It is the user's responsibility to de-assert pause_req upon pause_ack going high. Unless this signal is de-asserted, the transmit interface will keep sending pause frames. Note that when a pause frame is sent on the transmit side, the destination address is always (01-80-C2-00-00-01), whereas the source address is set to the smi_pause_sa[47:0] user programmable register defined in the SMI Register Interface section of this document.

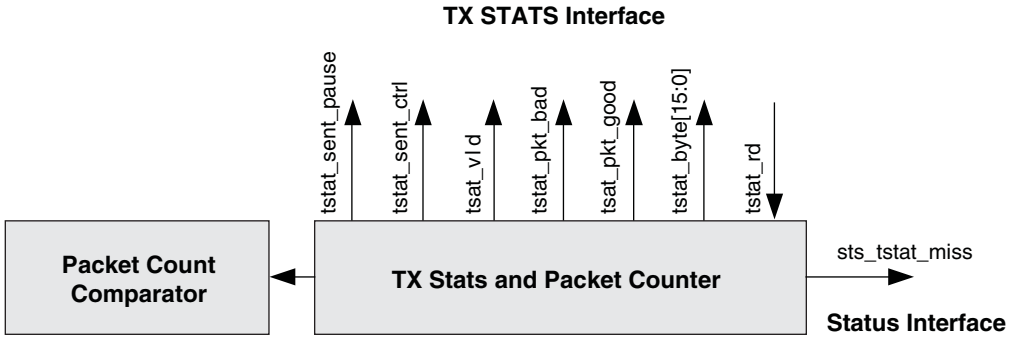
TX Microcontroller
This block contains the u-controller, which maps control inputs from the EBR interface, Flow Control Block and Control Interface and generates a control word, which is sent to the data pipeline to govern the actions of each stage of the pipeline.

Reconciliation Sublayer
When operating in 10Gb Ethernet mode this block provides support for error and fault signalling requirements of the standard

TX Stats and Packet Counter and the Packet Count Comparator
The TX stats and Packet Counter block counts the number of bytes in a packet and registers whether a packet was successfully sent. It also provides signals for the statistics bus to the client giving information on packet length and the validity of transmitted packets. It also passes the current packet length to a configurable length comparator, which indicates to the control logic if the length is less than a given threshold. This is used in Ethernet to detect short packets that need to be padded. SOP will be used to start the count and EOP to stop and hold the count.

The output of the Packet Count Comparator together with the EOP indicator control the insertion of PAD characters for Ethernet. Figure 5 shows the signals to and from the statistics and status interface.

Figure 5. TX Stats and Counters



Pause frame transmission is indicated by tstat_sent_pause and tstat_sent_ctrl signals.

The signals for the transmit flexiMAC statistics bus into the FPGA are listed in the I/O Description section of this document.

The valid signal tstat_vld is asserted after EOP has been received, indicating that the stats vector is available for the client to read. Statistics for the transmitted packet will remain valid until the client interface asserts tstat_rd or the complete transmission of the next packet. If tstat_rd is not asserted in this time, sts_tstat_miss (sent to the status bus) will be asserted following the transmission of the next packet to indicate a statistics vector was not read by the FPGA and has been missed. The FPGA is not required to generate the tstat_rd signal, and the sts_tstat_miss signal can be ignored if the FPGA logic is able to sample the statistics vector using the tstat_vld signal. The I/O Timing section of this document illustrates the TX statistics interface timing.

flexiMAC RX Functional Description

Overview

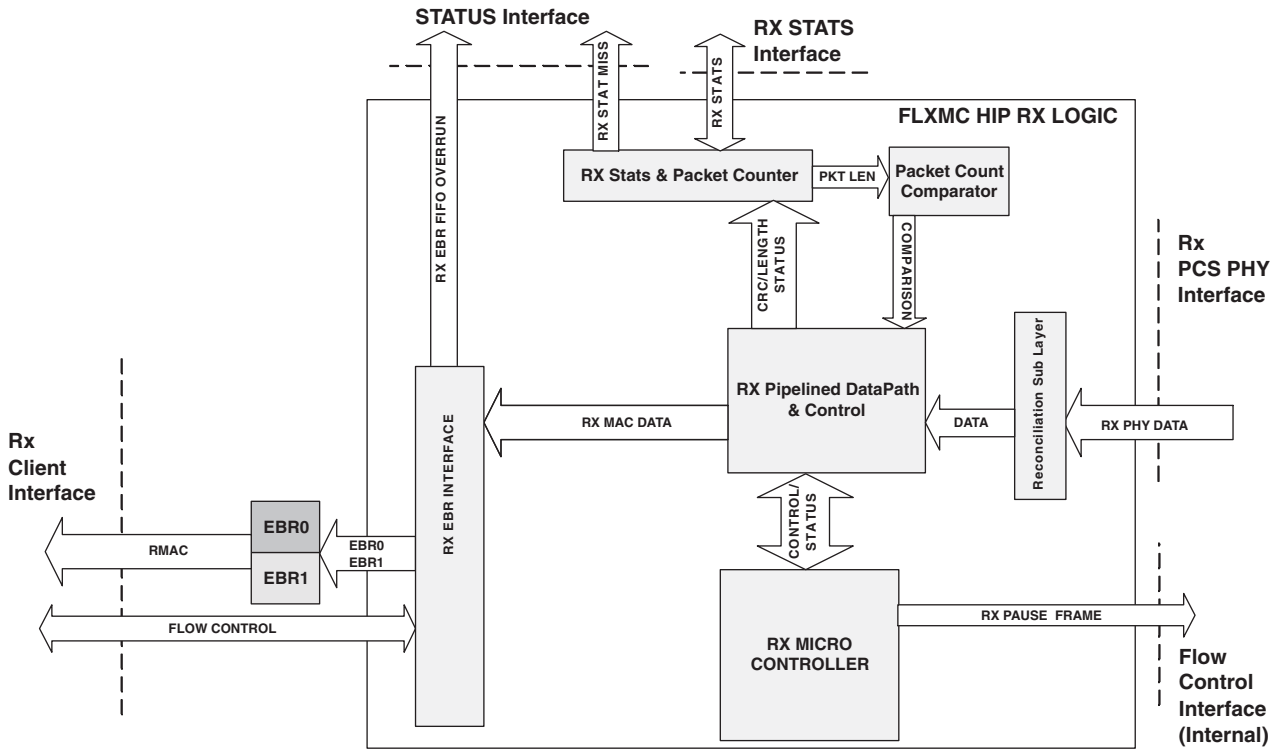
The basic function of the LatticeSCM Ethernet flexiMAC core RX path in Ethernet modes is the following:

- Parse incoming packets to extract the Destination Address (DA) field.
- Perform an address look-up on packet's DA to see if there is a destination address match for unicast, multicast or broadcast packets.
- Filter all packets for which there is no address match (unless in promiscuous mode in which case all packets are accepted).
- Detect VLAN tagged packets and indicate detection to statistics interface.
- Conduct a CRC32 check on all packets and flag CRC32 errors to the EBR interface so that the client logic can deal with errored packets.
- Detect PAUSE frames and extract and send the pause time to the Flow Control block (described later) to pause the transmission of packets (if there is no CRC32 error).
- Filter PAUSE packets after CRC check, unless the smi_ctrl_fltr_dis control bit is set (see the SMI Register Interface section).
- Remove any padding on short packets before passing it to the client logic.
- Indicate packet length, packet type (pause frames, unicast, multicast, broadcast), CRC errors, VLAN tags, etc. via the statistics bus interface.
- Treats a packet with a 0x8808 L/T field as a control frame. Any other L/T field value is interpreted as a data length indicator.

Rx Architecture

Figure 6 shows the architecture of the Rx block in the LatticeSCM Ethernet flexiMAC core. The architecture splits between control and data path blocks. The following is a description of the major blocks inside the FLXMC HIP.

Figure 6. flexiMAC Rx Architecture



RX EBR Interface Block

The RX EBR Interface block receives 32 bits of data from the flexiMAC data path and performs an interfacing and de-multiplexing function to the EBR's 64-bit data path. In 8-bit mode (1Gb Ethernet), there is no de-multiplexing.

Control data is formatted in a protocol dependent way to indicate the type of packet, the last byte position and if there was an error and so should be discarded by the client logic.

The RX EBR Interface Block is meant as a forwarding interface only; it should not be used for packet buffering. Any packet buffering should be done by the user application. During operation, if the RX EBR Interface Block is not read from fast enough, it can potentially overflow. When the FIFO is full the FIFO logic does not do flow control, all incoming data is dropped and the signal sts_rmac_fifo_overrun is asserted. If the user wishes to use a slower read clock or a throttled read mechanism, it would be best to implement another set of FIFOs in the user's application. This also makes it easer for the user logic to be implemented to generate pause packets when FIFO levels are getting too high for a slower interface.

The format of the data coming from the EBR interface is protocol dependent. Please refer to the I/O Descriptions section of this document for information on the corresponding RX client interface signals.

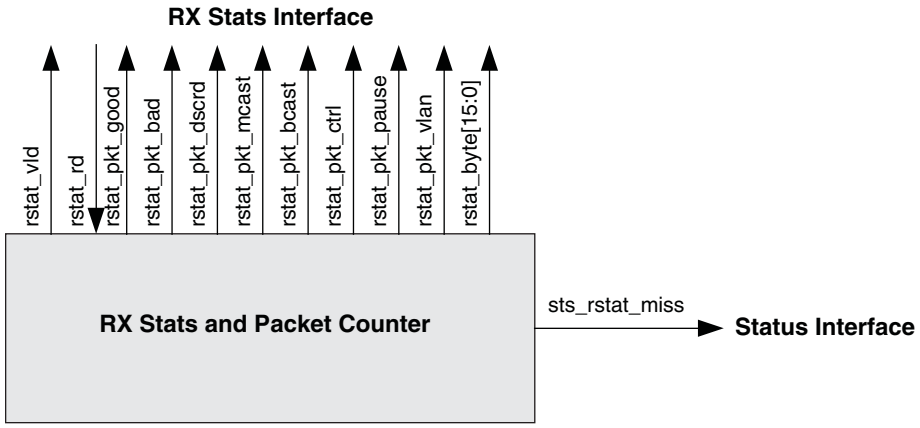
Reconciliation Sublayer

When operating in 10Gb Ethernet mode this block provides support for error and fault signalling requirements of the standard.

RX Microcontroller
This block contains the microcontroller that maps control inputs from the Rx Pipelined Data Path and Control, Flow Control Block, and control interfaces and generates a control word, which is sent to the data pipeline to govern the actions of each stage of the pipeline.

Rx Stats & Packet Counter and the Packet Count Comparator
The Rx Stats & Packet Counter block contains a byte counter for packet length. It provides signals for the statistics bus to the client giving information on packet length and the validity of transmitted packets. It also passes the current packet length to a configurable length comparator, which indicates to the control logic if the length is less than a given threshold. This signal, together with the EOP indicator, controls the removal of PAD characters for Ethernet. Figure 7 shows the signals to and from the statistics and status interface. For a description of these signals, please refer to the I/O Descriptions section of this document.

Figure 7. RX Stats Signals

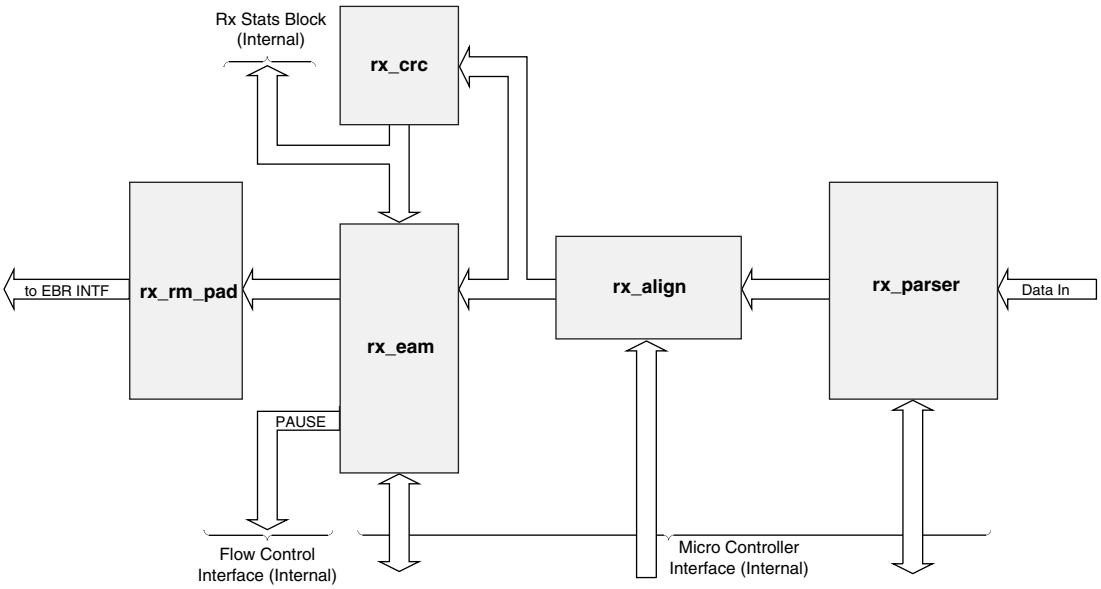


Statistics for the received packet will remain valid until the client reads the vector or the complete reception of the next packet. If rstat_rd is not asserted in time, sts_rstat_miss will be asserted following the reception of the next packet to indicate a statistics vector was not read by the FPGA and vector has been missed. The FPGA is not required to generate the rstat_rd signal, and the sts_rstat_miss signal can be ignored if the FPGA logic is able to sample the statistics vector using the rstat_vld signal. The I/O Timing section of this document illustrates the RX statistics interface timing.

Rx Pipelined Data Path and Control
The architecture for the Rx Data path block is shown in Figure 8. The functionality is described below.

The data path block interfaces to the RX EBR Interface, the RX Microcontroller, the RX Packet Counter and Comparator, the PHY interface, and the Flow Control block.

Figure 8. Rx Pipelined Data Path and Control Architecture



The basic function of the RX pipelined data path in Ethernet modes is the following:

- Detect start of packet and end of packet (rx_parser block).
- Detect VLAN Tagged Packets (rx_parser block).
- Look up packets to see if there is a destination address match for unicast, multicast or broadcast packets (rx_eam).
- Filter all packets for which there is no address match (unless in promiscuous mode in which case all packets are accepted) - (rx_eam).
- Detect PAUSE frames and send pause_time to the Flow Control block to pause the transmission of packets if there is no CRC32 error (rx_eam).
- Filter PAUSE packets after CRC check (rx_eam), unless the smi_ctrl_fltr_dis control bit is set (see the SMI Register Interface section).
- Detect VLAN-tagged packets and indicate this to the stats block (rx_parser + Microcontroller).
- Conduct a CRC32 check on all packets and flag CRC32 errors to the EBR interface so that the client logic can deal with error-ed packets (rx_crc).
- Remove the CRC field after checking.
- Remove any padding on short packets when L/T field value is smaller than data length (L/T field of 0x8808 indicates a control frame, so no pad removal occurs) before passing it to the client logic. (Microcontroller + rx_rm_pad).
- Interfaces to the RX Stats and Packet counter block to indicate packet-length, packet type (pause frames, unicast, multicast, broadcast), CRC errors, VLAN-Tags etc.

RX Parser (rx_parser): Data flowing through the parser is parsed on byte granularity. The microcontroller selects the fields for comparison. The microcontroller controls the rest of data path by sending control words down the pipelined data path.

The microcontroller controls:

- CRC checking
- Removal of padding
- Indication of type of packet (e.g. VLANS)
- Detection of special packets - Flow Control etc.
- Indication of valid data
- Indication of EOP
- Indication of SOP, etc.

Alignment Logic - rx_align: The alignment block properly aligns SOP and EOP, and valid signals with the incoming data.

Ethernet Address Match Logic (rx_eam)
Because Ethernet requires address lookups, there is a special Ethernet address match block (rx_eam), which indicates to the microcontroller if there is an address match. The CRC block checks and removes CRC32s. The rx_rm_pad block can remove parts of packets (again controlled by the microcontroller).

The Address matching logic carries out the following functions:

- Looks up Destination Address for unicast, multicast and broadcast matches.
- Deletes packets by setting valid bits to 0 if no address match is found.
- Detects PAUSE frames by looking for special DA: 01-80-C2-00-00-01.
- When a PAUSE frame is detected, it instructs the rx_rm_pad block to delete the frame after CRC checking.
- Captures the pause time and puts this on the Flow Control bus and requests the Flow Control block to initiate a pause to the TX interface if the CRC is good.
- Forms the last byte position (lbp[1:0]) on an end of packet.*
- Takes control bits from the microcontroller and pipelines them with the data as the address look-up is being performed.
- Asserts signals to the RX Stats and Packet counter block to indicate type of packet.

The first 6 bytes of the packet are used to see if this node accepts the packet or not. The packet lookup will return an address match under one of the conditions below:

The MAC is set in promiscuous mode: smi_unicast_fltr_dis is set (SMI register control bit, see the SMI Register Interface section of this document). All error-free packets will match.

For unicast addresses: The packet is a unicast packet (i.e. bit[0] of DA == 0) and the unicast address matches the 48-bit unicast destination MAC address programmed in SMI registers 0x08-0x0d (see Table 8).

For multicast addresses:

- The smi_mcast_fltr_dis (SMI register control bit, see the "SMI Register Interface" section) is set - so all multicast packets are accepted.
- The packet is multicast (i.e. bit[0] of DA == 1) and the multicast address is hashed and finds a match in the multicast table programmed in the SMI multicast filter registers. The multicast hash takes the first 6 bits of the CRC32 calculated over the first 6 bytes (the DA) of the Ethernet packet. These six bits are then used to index a unique filter bit in one of the eight multicast filter registers (see the "SMI Register Interface" section for the location of the 8 receive multicast filter registers. Each of the 8 multicast filter registers is 8 bits wide. Hence, the hash process will

index a multicast address into one of 64 possible bits in the multicast filter registers space. When a software developer wishes to accept a specific multicast address, he/she should follow the hash algorithm illustrated in the C language code of Figure 9 to determine which filter bit in the multicast registers to set. The C algorithm returns the smi_rx_mcast register index (0 to 7) as well as the bit within the register that needs to be set (0 to 7) based on a given multicast destination address input to the algorithm. Several bits can be set to accept several multicast addresses. If all 64 multicast filter register bits are set to 1, then all received multicast addresses will be passed to the FPGA client.

For broadcast packets: The packet is a broadcast packet (DA all 1s) and smi_bcast_fltr_en (SMI register control bit, see the SMI Register Interface section) is 0.

For pause packets: The packet is a PAUSE frame with DA (01-80-C2-00-00-01).

If a match is found, then packets are passed to the CRC checker. If no match is found, then the packet must be deleted and this is done by writing 0's to all valid bit fields in the data pipeline. The microcontroller will then invalidate all other words until the end of the packet.

If there is a PAUSE match then the following actions occur:

- The rx_rm_pad block is instructed to filter the packet, unless the smi_ctrl_fltr_dis control bit is set (see the SMI Register Interface section).
- The Length Type field is checked to be 88-08.
- The MAC Control Opcode field is checked to be 00-01.
- The 16-bit pause time is extracted from the Opcode parameter field and stored until the CRC is checked.
- If the CRC is good, then the pause time registered and a pause request is initiated to the Flow Control Block.
- If CRC errors are detected, then the error is flagged to the RX Stats and Packet counter block and no pause is signalled.

Figure 9. Multi-cast Bit Selection Hash Algorithm in C Language

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, unsigned char *argv[])
{

//Hexadecimal equivalent of the Flxmc CRC
//equn.
#define CRC_POLYNOMIAL 0x04c11db6

//The Multicast address is held in a 6 byte
//array
unsigned char multi_addr[6];
// variables
unsigned long int crc;
int i, j, bit;
int carry;
int register_no, register_bit;

crc = 0xffffffff;

//Input data from command line
for (j=0; j<6; j++)
{
sscanf(argv[j+1], "%x", &multi_addr[j]);
printf("%s  \n", argv[j+1]);
}

//The following loops create the 32-bit crc
//value.
//loop through each byte of the address.
for(i=0; i<6; i++)
{
//Loop through each byte bit of that byte.
for(bit=0; bit<8; bit++)
{
  //printf("multi_addr[i] %x \t t %lx \t u %lx \t", multi_addr[i], t, u);
  carry = (crc >> 31)^((multi_addr[i] & (1 << bit)) >> bit);
  crc <<= 1;
  printf("crc %lx  carry=%d\n", crc, carry);
  if (carry)
  crc = (crc ^ CRC_POLYNOMIAL) | carry;
}
}
//Extract the 6 MSBs from the CRC value,
//this six bit value is used to index a
//unique filter bit.
printf("crc %lx \n", crc);
crc >>= 26;
crc &= 0x3F;

//Find the multicast register number and
//bit of that register to set.
printf("crc %lx \n", crc);
register_no = crc >> 3;
register_bit = crc & 7;

printf ("register_no %lx register_bit %lx \n", register_no, register_bit);

  system("PAUSE");
  return 0;
}
```

Receive Remove Pad (rx_rm_pad): When instructed by the rx_eam block to strip data, this block does so and asserts the output EOP until it sees the next input EOP.

CRC Logic (rx_crc): The CRC support the standards dependant polynomials shown in Table 5.

Table 5. CRC Polynomials for Ethernet Standards

Protocol	CRC-32
1Gb Ethernet	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
10Gb Ethernet	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

The CRC supports 32-bit (for 10Gb E) and 8-bit (for 1Gb E) data paths. The CRC also work across 16 and 24 bits, depending on the last byte position (LBP) as the last word in a packet may have 1, 2, 3 or 4 bytes valid.

Depending on how many bytes are present in the last 32-bit word one of the 4 phases is selected. Phases 16 and 24 are only used when an EOP has occurred and only the first 2 or 3 bytes are valid. Phase 8 is only used when an 8-bit data path is employed (1Gb Ethernet). This is illustrated in Table 6.

Table 6. CRC Phase Selection

Mode	Phase	CRC
1Gb Ethernet	Phase 8	CRC 32
10Gb Ethernet (EOP = 0)	Phase 32	CRC 32
10Gb Ethernet (EOP = 1 LBP = 0)	Phase 8	CRC 32
10Gb Ethernet (EOP = 1 LBP = 1)	Phase 16	CRC 32
10Gb Ethernet (EOP = 1 LBP = 2)	Phase 24	CRC 32
10Gb Ethernet (EOP = 1 LBP = 3)	Phase 32	CRC 32

For a 32-bit data path, the number of valid bytes in the last word is indicated by lbp[1,0] together with EOP being asserted. The encoding is shown in Table 7.

Table 7. Last Byte Position and CRC

lbp[1:0]		EOP	Action
0	0	1	One byte valid - CRC32 complete
0	1	1	Two bytes valid - CRC32 complete
1	0	1	Three bytes valid - CRC32 complete
1	1	1	Four bytes valid - CRC32 complete

The CRC data follows the data in the packet. The nature of CRC arithmetic means that the CRC calculated over the receive data/CRC should produce a zero result, meaning that no comparison between received and calculated CRC values is required.

Flow Control

The Flow Control block performs data-link and MAC functions that are not performed in the receive flexiMAC and transmit flexiMAC blocks. This function disables transmission of Ethernet packets upon receipt of an Ethernet pause value. Figure 10 illustrates the Flow Control inputs and outputs.

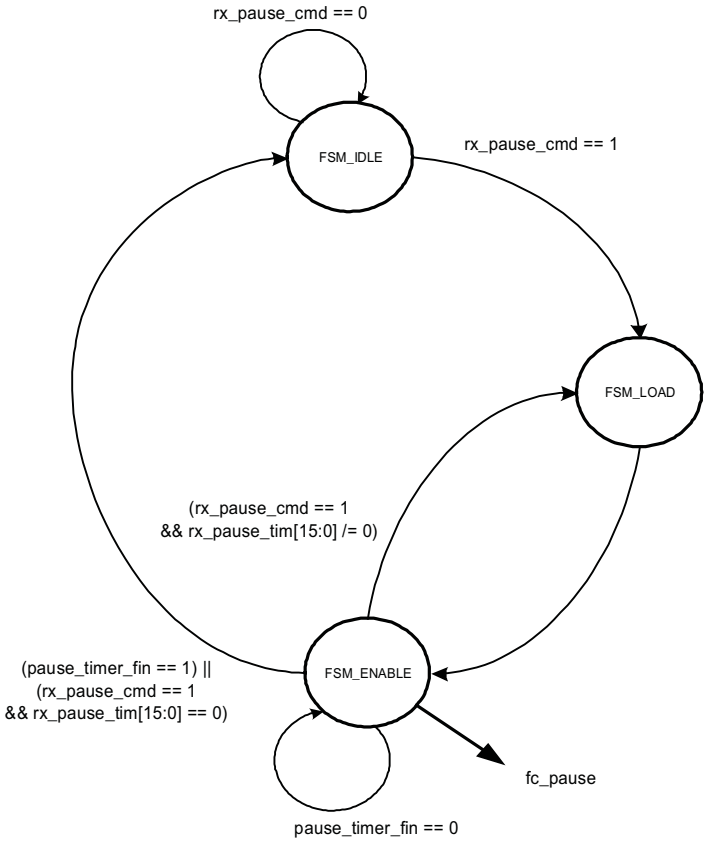
Figure 10. Flow Control Block Diagram

The Ethernet pause frame operates as follows:

- The receiver (rx_eam block) detects a pause frame and asserts a pause command, at the same time providing a pause time parameter extracted from the pause frame.
- This pause time parameter is loaded into a pause timer immediately following a pause command asserted by the receiver.
- This pause timer counts the number of bit periods determined by the pause time parameter.
- A request to pause transmission of Ethernet packets is made to the transmitter by the flow control when the pause timer has started counting.
- When the pause timer has reached the end of the pause time period, the request to pause transmission of Ethernet packets is removed, and the pause timer is reset ready for the next pause frame to be received.

The Ethernet pause command state machine state diagram is shown in Figure 11.

Figure 11. Ethernet Pause Command State Machine State Diagram



SMI Register Interface

A bank of registers is implemented internally to the flexiMAC hard IP to manage various programmable control functions. These registers are controlled by a register interface that is compatible with the LatticeSCM Serial Memory Interface (SMI). The actual base address of these registers is specified by the user by use of the SMI_OFFSET parameter and the locations in Table 8 are byte offsets of this parameter. Please refer to Lattice technical note number TN1085, *LatticeSC MPI/System Bus* for more information on SMI. Due to the way the SMI slave interface is implemented on the LatticeSCM Ethernet flexiMAC core. The SMI_OFFSET parameter must be a value equal or higher to 0x440. Also, SMI_OFFSET must be a multiple of 4. In other words, acceptable SMI_OFFSET values are 0x440, 0x480, 0x4C0,...,0x7C0 but not 0x441.

The following characteristics apply to the SMI interface on the LatticeSCM Ethernet flexiMAC core:

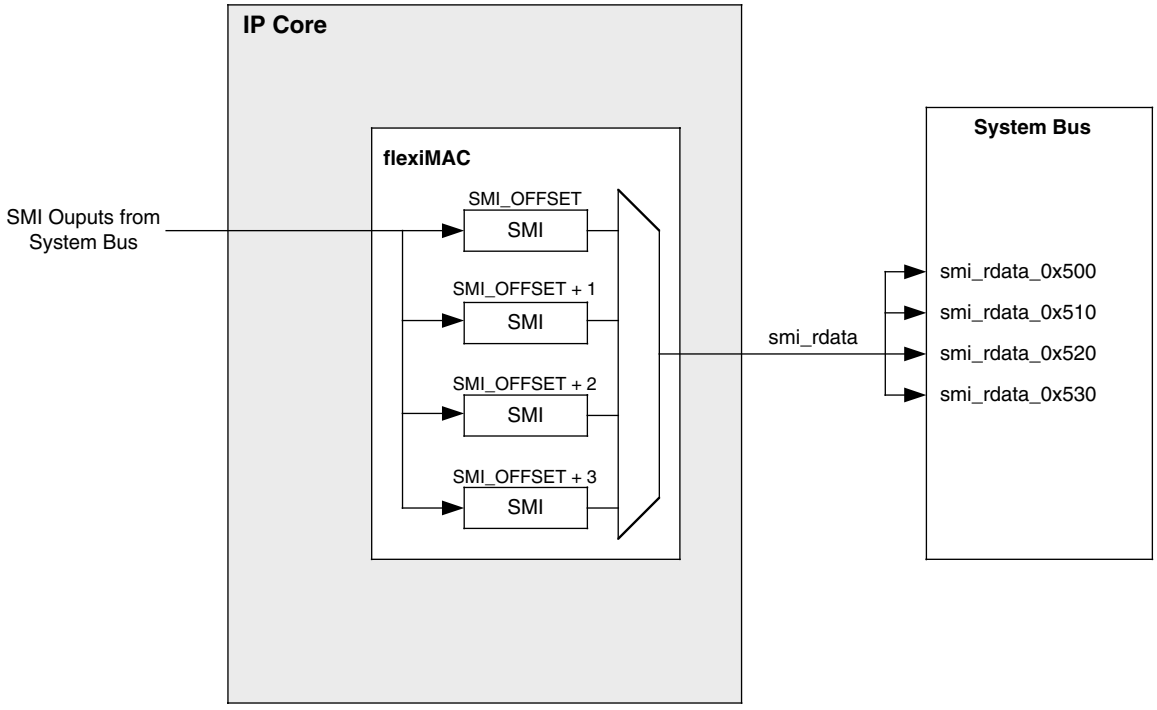
1. The SMI register logic occupies 4 SMI block addresses (each block is 16x8 bits), even through not all bits are internally used. This explains why valid SMI_OFFSET values are 0x440, 0x480, 0x4C0,...,0x7C0 but not 0x441.

2. Once an SMI_OFFSET value is assigned, the next 3 addresses cannot be used by any other SMI block. For example, if SMI_OFFSET for the LatticeSCM Ethernet flexiMAC core is set to 0x480, then 0x490, 0x4A0, 0x4B0 cannot be used by any other SMI peripheral.

3. When used with the System Bus, in order to properly read the SMI data from the flexiMAC, SRI_RDATA (from flexiMAC) has to be connected to all 4 corresponding SMI read ports on the System Bus. For example, if flexiMAC is using 0x440, the SRI_RDATA needs to connect back to system bus' SMI_RDATA_0X440, SMI_RDATA_0X450, SMI_RDATA_0X460, and SMI_RDATA_0X470 ports.

Figure 12 illustrates another example of a flexiMAC to System Bus connection. In this example, SMI_OFFSET is set to 0x500.

Figure 12. flexiMAC and System Bus Connections Example



The SMI_OFFSET must be set for both simulation and place and route design flows.

For simulation, the SMI_OFFSET can be set via defparam as shown below.

```
defparam flxmc_sys_ge.wrapper_flxmc_core.flxmc_top_sys.flxmc_top_ebr.flxmc_top_mib.SMI_
OFFSET=11'h500;
```

For implementation in ispLEVER the SMI_OFFSET should be set in the .lpf file using the ASIC preference.

```
ASIC "flxmc_sys_ge/wrapper_flxmc_core/flxmc_top_sys/flxmc_top_ebr/flxmc_top_mib" TYPE "FLXMC"
SMI_OFFSET="0x500";
```

Both simulation and implementation examples above assume the architecture shown in Figure 2, and Gigabit Ethernet mode. The top level is flxmc_ge_user. Changes in the way the flexiMAC levels are instantiated would require a change in the instantiation path for above examples.

Also note that the simulation testbench implements an SMI_OFFSET of 0x440 irrespective of the value the user defines in IPexpress or through the defparam statement. The user should choose an SMI_OFFSET of 0x440 if they wish to run the simulation testbench provided (or modify the testbench). The system bus model provided is also configured for an SMI_OFFSET of 0x440. If the user chooses a different SMI_OFFSET for implementation, then the system bus block needs to be re-generated with new SMI address ports.

Table 8 defines the register bits that are accessed via SMI.

Table 8. SMI Memory Control Registers

Byte Address Offset from SMI_OFFSET Base	Bits Order in Data Byte	Signal	Description	Static/ Dynamic*
0x00->0x07		**smi_rx_mcast0[7:0] to smi_rx_mcast7[7:0]**	**Receive multicast filter table**	**Static**
0x00	7:0	smi_rx_mcast0[7:0]	Receive multicast address register 0	Static
0x01	7:0	smi_rx_mcast1[7:0]	Receive multicast address register 1	Static
0x02	7:0	smi_rx_mcast2[7:0]	Receive multicast address register 2	Static
0x03	7:0	smi_rx_mcast3[7:0]	Receive multicast address register 3	Static
0x04	7:0	smi_rx_mcast4[7:0]	Receive multicast address register 4	Static
0x05	7:0	smi_rx_mcast5[7:0]	Receive multicast address register 5	Static
0x06	7:0	smi_rx_mcast6[7:0]	Receive multicast address register 6	Static
0x07	7:0	smi_rx_mcast7[7:0]	Receive multicast address register 7	Static
0x08->0x0d		**smi_mac_addr[47:0]**	**Sets the station unicast destination MAC address**	**Static**
0x08	7:0	smi_mac_addr[7:0]	6th received byte in destination MAC address	Static
0x09	7:0	smi_mac_addr[15:8]	5th received byte in destination MAC address	Static
0x0a	7:0	smi_mac_addr[23:16]	4th received byte in destination MAC address	Static
0x0b	7:0	smi_mac_addr[31:24]	3rd received byte in destination MAC address	Static
0x0c	7:0	smi_mac_addr[39:32]	2nd received byte in destination MAC address	Static
0x0d	7:0	smi_mac_addr[47:40]	1st received byte in destination MAC address	Static

Table 8. SMI Memory Control Registers (Continued)

Byte Address Offset from SMI_OFFSET Base	Bits Order in Data Byte	Signal	Description	Static/ Dynamic*
0x0e	0	smi_phy_loopback	Enables TPHY->RPHY loopback within the flexiMAC core	Static
	1	smi_pause_dis	Disables pause behavior of transmit MAC. Received pause frames will not cause the transmit MAC to defer transmission of the next packet until the pause timer has expired	Static
	2	smi_ctrl_fltr_dis	Disables filtering of MAC control packets (including pause packets). Setting this bit to 1 allows received PAUSE packets to be sent to the FPGA client.	Static
	3	smi_unicast_fltr_dis	Disables filtering of unicast packets (promiscuous mode). All received packets (including PAUSE packets) will be passed to the FPGA client	Static
	4	smi_mcast_fltr_dis	Disables filtering of all multicast packets. All received multicast packets will be passed to the FPGA client.	Static
	5	smi_bcast_fltr_en	Enables filtering of all broadcast packets. Received broadcast packets will not be passed to the FPGA client.	Static
	6	smi_phy_up	Control bit. Indicates physical layer is operational. The flexiMAC will not properly function unless this bit is set. One way to control this signal is by monitoring the PCS the link state machine status.	Dynamic
	7	smi_rx_reset	Resets the receive block (active high)	Dynamic
0x0f	0	smi_tx_reset	Resets the Transmit and Flow Control blocks (active high)	Dynamic
	1	smi_rsl_dis	Disables Link Fault Signaling when in 10 GbE mode. smi_rsl_dis, when set high, prevents the tx_err_ins from inserting error codes on in the TX data bus. smi_rsl_dis does not affect the rb_flt_stt bus (receive link fault status) which is always enabled in 10 GbE mode.	Static
	2:7		Unused	
0x10->0x17			Unused	
0x18->0x1D		**smi_pause_sa[47:0]**	**Transmitter pause source address**	**Static**
0x18	7:0	smi_pause_sa[7:0]	6th transmitted byte in transmitter pause source address	Static
0x19	7:0	smi_pause_sa[15:8]	5th transmitted byte in transmitter pause source address	Static
0x1A	7:0	smi_pause_sa[23:16]	4th transmitted byte in transmitter pause source address	Static
0x1B	7:0	smi_pause_sa[31:24]	3rd transmitted byte in transmitter pause source address	Static
0x1C	7:0	smi_pause_sa[39:32]	2nd transmitted byte in transmitter pause source address	Static
0x1D	7:0	smi_pause_sa[47:40]	1st transmitted byte in transmitter pause source address	Static
0x1E->0x3F			Unused	

In the table, each bit is defined as either static or dynamic. Static bits are programmed before the application of a soft reset and do not change during operation. These bits are not synchronized. Dynamic bits can change during operation and to avoid the possibility of metastability issues these bits are synchronized.

When accessing the flexiMAC SMI registers from a master interface on the system bus, a byte address in the table above can be accessed by adding the corresponding byte offset value to the SMI_OFFSET. For example, if SMI_OFFSET= 0x440, and smi_pause_sa[47:40] is to be accessed (offset=0x2D), then the address applied at the master interface (MPI or UMI address) will be 440+2D=0x46D.

Design Parameters

IspLEVER IPexpress generates design parameters automatically. These parameters are included in the flxmc_GE_param.v (1Gb Ethernet mode) and flxmc_XGE_param.v (10Gb Ethernet mode).

Either of these two parameter files has to be used in the design depending on the mode (1Gb Ethernet mode or (10Gb Ethernet mode). Most of the parameters in the parameter files are pre-set by ispLEVER IPexpress and should not be altered by the user. The only exception is SMI_OFFSET.

The user can change the SMI_OFFSET depending on where the LatticeSCM Ethernet flexiMAC core should appear in the device memory map. There are some restrictions on the possible values for SMI_OFFSET. These restrictions are detailed in the SMI Register Interface section. The default value for SMI_OFFSET in the parameter file is 0x440 (11'h440 in Verilog format).

Implementation Details

Voltage Requirements

Due to an EBR design constraint, the LatticeSCM Ethernet flexiMAC Core does not support a core voltage lower than 1.14V in 10 Gigabit Ethernet mode (312.5MHZ system clock) for -5 speed grade devices. This limitation does not exist in Gigabit Ethernet mode (125MHZ system clock).

Locating the IP

The LatticeSCM Ethernet flexiMAC IP core uses a mixture of hard and soft IP blocks to create the full design. This mixture of hard and soft IP requires the user to locate, or place, the IP core in a defined location on the device array. The hard blocks of the flexiPCS/SERDES and flexiMAC fixed locations will drive the location of the IP. Table 9 shows the site names for flexiMAC locations for each LatticeSCM device.

Table 9. flexiMAC Location Site Names

Device	Site Name
SC15	LUMACO0
SC25	LUMACO0 RUMACO0
SC40	LUMACO1 RUMACO1
SC80	LUMACO1 RUMACO1
SC115	LUMACO2 RUMACO2 LUMACO0 RUMACO0

Figure 13 provides a diagram for each of the LatticeSCM 15, 25, 40, 80, and 115 devices with site names for the MACO and flexiPCS/SERDES blocks. The MACO cores shaded in gray represent valid flexiMAC locations. Valid flexiPCS/SERDES locations are dependent on the package selection since all flexiPCS/SERDES quads are not bonded out to package pins in all packages.

The user should select the flexiPCS/SERDES quad location based on the package pinout and the location of the Ethernet interface on the board layout. Once a flexiPCS/SERDES quad has been located the closest flexiMAC location to the selected flexiPCS/SERDES location should be used.

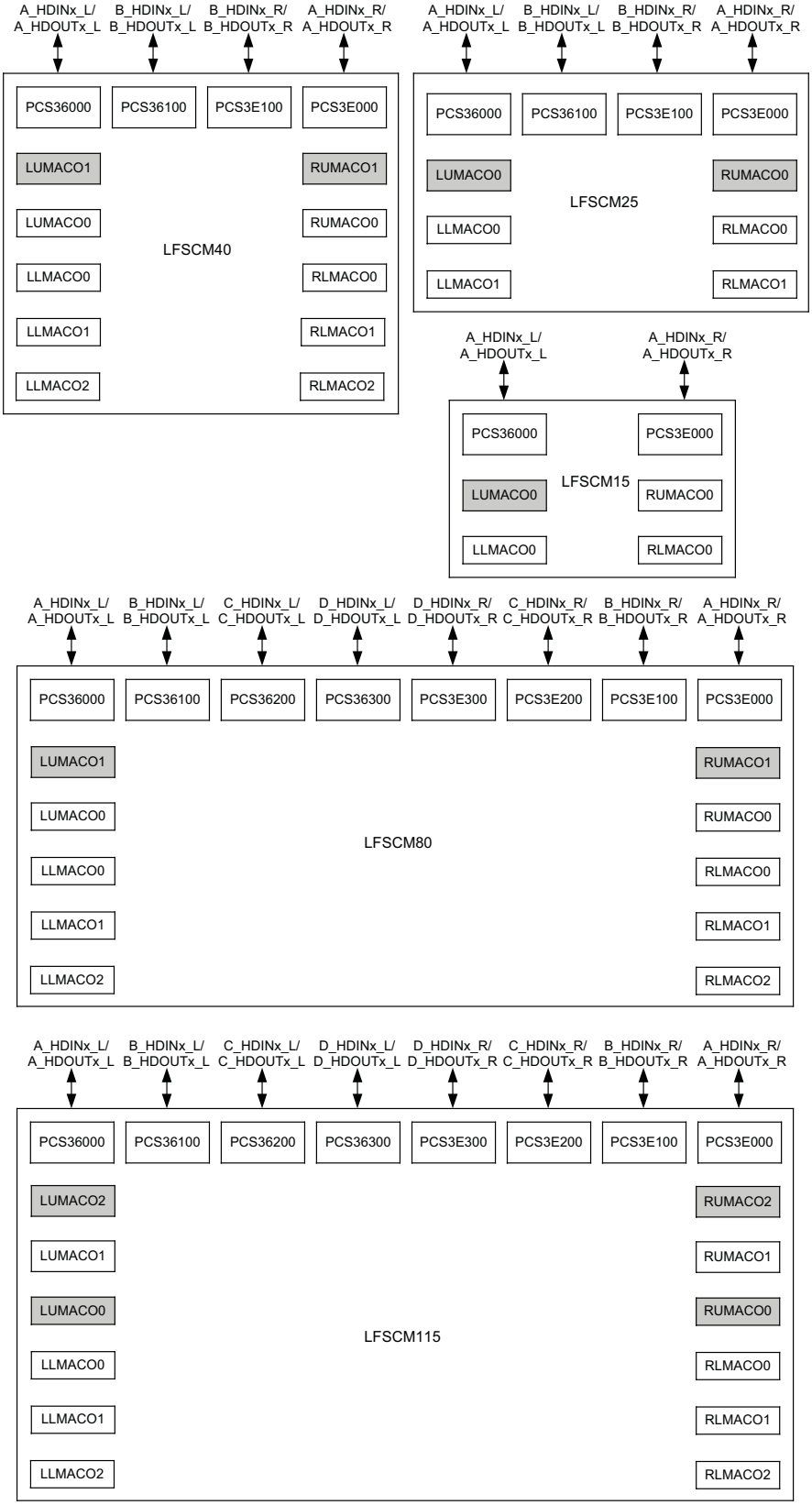
In order to locate the flexiPCS/SERDES and flexiMAC, the user must use a LOCATE preference in the .lpf file of ispLEVER. The instance of the flexiMAC and flexiPCS/SERDES block is deep inside the IP core so the complete IP hierarchy must be included in the component description. Below is an example of locating both blocks.

```
LOCATE COMP "flxmc_sys_xge/u1_xgig_eth_pcs/pcsa_inst" SITE "PCS36000";

LOCATE COMP "flxmc_sys_xge/wrapper_flxmc_core/flxmc_top_sys/flxmc_top_ebr/flxmc_top_mib"
SITE "LUMACO0";
```

The LOCATE examples above assume the architecture shown in Figure 2, and 10 Gigabit Ethernet mode. The top level is flxmc_xge_user. Changes in the way the flexiMAC levels are instantiated require a change in the instantiation path for above examples.

Figure 13. Device Arrays with flexiPCS/SERDES and MACO Sites



Note: The MACO cores shaded in gray represent valid flexiMAC locations.

I/O Descriptions

Table 10. I/O List of flxmc_sys_xge for 10 Gigabit Ethernet Mode

flxmc_sys_xge		Description
Name	**Type**	
Major Clocks and Reset		
refclkp	I	Transmit reference clock to PCS QUAD SERDES (156MHZ). Positive polarity
refclkn	I	Transmit reference clock to PCS QUAD SERDES (156MHZ). Negative polarity
ref_pclk	O	System clock from PCS QUAD FPGA interface (312MHz)
rst_n	I	Asynchronous active low reset
ff_clk_mac	I	Clock for client logic/EBR interface
Transmit EBR Interface		
tmac_data[63:0]	I	Transmit data Transmit data. [7:0] is the first byte to be transmitted
tmac_eop	I	End of Packet marker. The next word following an EOP is the first word (SOP) for the next packet.
tmac_we	I	Write enable
tmac_lbp[2:0]	I	Last byte position. Valid only when tmac_eop is asserted, 000 otherwise. [2:0]: 000 = 1 byte valid 001 = 2 bytes valid 010 = 3 bytes valid 011 = 4 bytes valid 100 = 5 bytes valid 101 = 6 bytes valid 110 = 7 bytes valid 111 = 8 bytes valid
EMB0_FULL	O	Indicates Transmit EBR FIFO is full. Also know as tmac_full. Note that the <IPexpress_module_name> module also contains a tmac_empty output port to indicate when the transmit EBR FIFO is empty.
EMB0_AMFULL	O	Indicates Transmit EBR FIFO is almost full. Also known as tmac_afull. Note that the <IPexpress_module_name> module also contains a tmac_aempty output port to indicate when the transmit EBR FIFO is almost empty.
Transmit Statistics Bus		
tstat_vld	O	Strobe to indicate that the statistics bus is valid.
tstat_rd	I	Request to read stats.
tstat_pkt_good	O	Indicates packet transmitted correctly with a valid CRC.
tstat_pkt_bad	O	Indicates packet not transmitted successfully due to TX EBR FIFO under run, resulting in a bad CRC being transmitted
tstat_byte[15:0]	O	Total length (in bytes) of transmit packet including pad bytes and 4 byte CRC.
tstat_sent_pause	O	Indicates a pause frame was transmitted as a response to a client pause frame request.
tstat_sent_ctrl	O	Indicates a control frame was transmitted.
Transmit SERDES PHY Interface		
hdoutp_0	O	Transmit high speed SERDES output – Lane 0, positive polarity
hdoutn_0	O	Transmit high speed SERDES output – Lane 0, negative polarity
hdoutp_1	O	Transmit high speed SERDES output – Lane 1, positive polarity
hdoutn_1	O	Transmit high speed SERDES output – Lane 1, negative polarity
hdoutp_2	O	Transmit high speed SERDES output – Lane 2, positive polarity
hdoutn_2	O	Transmit high speed SERDES output – Lane 2, negative polarity
hdoutp_3	O	Transmit high speed SERDES output – Lane 3, positive polarity
hdoutn_3	O	Transmit high speed SERDES output – Lane 3, negative polarity

Table 10. I/O List of flxmc_sys_xge for 10 Gigabit Ethernet Mode (Continued)

flxmc_sys_xge		Description
Name	**Type**	
Client Receive Interface		
rmac_data[63:0]	O	Data from EBR. [7:0] is the first byte received
rmac_lbp[2:0]	O	Last Byte Position. Valid only when rmac_eop is asserted. [2:0]: 000 = 1 byte valid 001 = 2 bytes valid 010 = 3 bytes valid 011 = 4 bytes valid 100 = 5 bytes valid 101 = 6 bytes valid 110 = 7 bytes valid 111 = 8 bytes valid
rmac_sop	O	Start of packet marker. Indicates a valid Ethernet SFD was received.
rmac_eop	O	End of packet marker. Indicates the end of the current Ethernet packet.
rmac_err	O	Error in packet (CRC). When a CRC error occurs, rmac_err is asserted along with rmac_eop on the rising edge of ff_clk_mac.
rmac_re	I	EBR read enable
Receive EBR Interface		
EMB2_EMPTY	O	RX EBR FIFO is empty. Also known as rmac_te.
EMB2_AMEMPTY	O	RX EBR FIFO is almost empty. Also known as rmac_ae.
Receive Statistics Bus		
rstat_vld	O	Indicates receive statistics bus valid
rstat_rd	I	Read acknowledge from client
rstat_pkt_good	O	Received packet with correct CRC
rstat_pkt_bad	O	Received packet with incorrect CRC. The flexiMAC does not discard the packet.
rstat_pkt_dscrd	O	Received packet was discarded by address filtering logic. It is not asserted when a PAUSE frame is discarded. (Note: rstat_byte does not indicate correct length of discarded packet).
rstat_pkt_mcast	O	Received multicast packet
rstat_pkt_bcast	O	Received broadcast packet
rstat_pkt_ctrl	O	Received control packet
rstat_pkt_pause	O	Received pause packet
rstat_pkt_vlan	O	Received VLAN tagged packet
rstat_byte[15:0]	O	This number represents a byte count: In 10 Gigabit Ethernet mode: • If PAUSE packet: 50 • If other than PAUSE packet: DATA +CRC(4 bytes) In Gigabit Ethernet mode: • If DATA length (in bytes) ≤ 60: 60 • If DATA length (in bytes) ≥ 61: DATA-1
Receive Data Path PHY Interface		
hdinp_0	I	Receive high speed SERDES input – Lane 0, positive polarity
hdinn_0	I	Receive high speed SERDES input – Lane 0, negative polarity
hdinp_1	I	Receive high speed SERDES input – Lane 1, positive polarity
hdinn_1	I	Receive high speed SERDES input – Lane 1, negative polarity

Table 10. I/O List of flxmc_sys_xge for 10 Gigabit Ethernet Mode (Continued)

flxmc_sys_xge		Description
Name	**Type**	
hdinp_2	I	Receive high speed SERDES input – Lane 2, positive polarity
hdinn_2	I	Receive high speed SERDES input – Lane 2, negative polarity
hdinp_3	I	Receive high speed SERDES input – Lane 3, positive polarity
hdinn_3	I	Receive high speed SERDES input – Lane 3, negative polarity
10Gb Ethernet Reconciliation		
tx_err_ins	I	Transmission errors enable. tx_err_ins will insert Error characters into all Tx lanes as per section 46.3.3.2 of 802.3ae-2002 (10 Gig reconciliation sub-layer. In the flexiMAC, the tx_err_ins is synchronized (double flopped) to the system clock. The flexiMAC will insert error columns for as many system cycles as the synchronized version of tx_err_ins is high.
rb_flt_stt[1:0]	O	Indicates Link Fault Status 00: Link Ok 01: Local Fault 10: Remote Faultrb_flt_stt[1:0] adheres to the link fault signaling state machine in figure 46-9 of the IEEE 802.3ae -2002 specification. Its possible values follow table 46-5, section 46.3.4 of the same document. It is active during the IDLE phase of the received packet but does not have any relationship to other signals on the RX client interface.
Status Interface		
sts_tstat_miss	O	Indicates last transmit statistics vector was not read and has been lost.
sts_rstat_miss	O	Indicates last receive statistics vector was not read and has been lost
sts_rmac_fifo_overrun	O	Indicates receive MAC EBR FIFO overran
sts_tmac_fifo_underrun	O	Indicates transmit MAC EBR FIFO underran. Note that when sts_tmac_fifo_underrun is asserted, a packet that underflows may look longer than the original packet since the flexiMAC stops reading from the TX EBR FIFO but does not insert any EOP characters during the underrun condition. The client side could be programmed to discard the remaining packet data and write EOP to the FIFO on an underrun condition.
sts_pause_active	O	Indicates the transmit flexiMAC is currently paused following reception of a valid pause frame
Ethernet Pause Interface		
pause_req	I	Request from Ethernet MAC client to send a pause frame
pause_count[15:0]	I	Pause time to be sent in the transmitted pause frame
pause_ack	O	Signal to indicate that the pause frame has been transmitted
SMI Interface		
smi_clk	I	SMI Clock
smi_reset_n	I	SMI Reset (active low)
smi_rd	I	SMI read strobe
smi_wr	I	SMI write strobe
smi_wdata	I	SMI serial write data
smi_addr	I	SMI address
smi_rdata	O	SMI serial read data
PCS QUAD to System Bus Interface		
pcs360_out[44:0]	I	Bus from System Bus to PCS QUAD
pcs360_in[16:0]	O	Bus from PCS QUAD to System Bus
Miscellaneous PCS QUAD Control/Status Signals		
felb	I	Active high far-end loopback enable for all four channels.
lsm_en	I	Receive link state machine enable for all four channels.
lsm_status_[0-3]	O	Per channel signal from receive link state machine indicating successful link synchronization.
mca_align_en	I	Active high multi-channel aligner enable for all four channels.

Table 10. I/O List of flxmc_sys_xge for 10 Gigabit Ethernet Mode (Continued)

flxmc_sys_xge		Description
Name	**Type**	
mca_resync	I	Active high asynchronous multi-channel aligner resynchronization signal.
mca_aligned	O	Active high signal indicating successful alignment of channels.
ctc_orun_[0-3]	O	Per channel active high flag indicator that clock tolerance compensation FIFO has overrun.
ctc_urun_[0-3]	O	Per channel active high flog indicator that clock tolerance compensation FIFO has under-run.

Table 11. I/O List of flxmc_sys_ge for 1Gb Ethernet Mode

flxmc_sys_ge		Description
Name	**Type**	
Major Clocks and Reset		
refclkp	I	Transmit reference clock to PCS QUAD SERDES (125MHZ). Positive polarity.
refclkn	I	Transmit reference clock to PCS QUAD SERDES (125MHZ). Negative polarity.
ref_pclk	O	System clock from PCS QUAD FPGA interface (125MHz)
rst_n	I	Asynchronous active low reset
Transmit Interface		
tmac_data[7:0]	I	Data
tmac_eop	I	End of packet marker. The next word following an EOP is the first word (SOP) for the next packet
tmac_we	I	Write enable for EBR FIFO
EMB0_FULL	O	Indicates Transmit EBR FIFO is full. Also know as tmac_full. Note that the <IPexpress_module_name> module also contains a tmac_empty output port to indicate when the transmit EBR FIFO is empty.
EMB0_AMFULL	O	Indicates Transmit EBR FIFO is almost full. Also known as tmac_afull. Note that the <IPexpress_module_name> module also contains a tmac_aempty output port to indicate when the transmit EBR FIFO is almost empty.
Transmit Statistics Bus		
tstat_vld	O	Strobe to indicate that the statistics bus is valid
tstat_rd	I	Request to read stats.
tstat_pkt_good	O	Indicates packet transmitted correctly with a valid CRC
tstat_pkt_bad	O	Indicates packet not transmitted successfully due to TX EBR FIFO under run, resulting in a bad CRC being transmitted
tstat_byte[15:0]	O	Total length of transmit packet including pad bytes and 4-byte CRC
tstat_sent_pause	O	Indicates a pause frame was transmitted as a response to a client pause frame request.
tstat_sent_ctrl	O	Indicates a control frame was transmitted.
Transmit SERDES PHY Interface		
hdoutp_0	O	Transmit high speed SERDES output -Lane 0, positive polarity
hdoutn_0	O	Transmit high speed SERDES output -Lane 0, negative polarity
Client Receive Interface		
rmac_data[7:0]	O	Data
rmac_sop	O	Start of packet marker
rmac_eop	O	End of packet marker
rmac_err	O	Error indicator. When a CRC error occurs, rmac_err is asserted along with rmac_eop on the rising edge of ff_clk_mac.
rmac_re	I	EBR read enable
Receive EBR Interface		
EMB2_EMPTY	O	RX EBR FIFO is empty. Also known as rmac_te.
EMB2_AMEMPTY	O	RX EBR FIFO is almost empty. Also known as rmac_ae.
Receive Statistics Bus		
rstat_vld	O	Indicates receive statistics bus valid
rstat_rd	I	Read acknowledge from client
rstat_pkt_good	O	Received packet with correct CRC
rstat_pkt_bad	O	Received packet with incorrect CRC. The flexiMAC does not discard the packet.

Table 11. I/O List of flxmc_sys_ge for 1Gb Ethernet Mode (Continued)

flxmc_sys_ge		
Name	**Type**	**Description**
rstat_pkt_dscrd	O	Received packet was discarded by address filtering logic. It is not asserted when a PAUSE frame is discarded. (Note: rstat_byte does not indicate correct length of discarded packet).
rstat_pkt_mcast	O	Received multicast packet
rstat_pkt_bcast	O	Received broadcast packet
rstat_pkt_ctrl	O	Received control packet
rstat_pkt_pause	O	Received pause packet
rstat_pkt_vlan	O	Received VLAN tagged packet
rstat_byte[15:0]	O	This number represents a byte count: In 10 Gigabit Ethernet mode: • If PAUSE packet: 50 • If other than PAUSE packet: DATA +CRC(4 bytes) In Gigabit Ethernet mode: • If DATA length (in bytes) ≤ 60: 60 • If DATA length (in bytes) ≥ 61: DATA-1
Receive Data Path PHY Interface		
hdinp_0	I	Receive high speed SERDES input-Lane 0, positive polarity
hdinn_0	I	Receive high speed SERDES input-Lane 0, negative polarity
Status Interface		
sts_tstat_miss	O	Indicates last transmit statistics vector was not read and has been lost
sts_rstat_miss	O	Indicates last receive statistics vector was not read and has been lost
sts_rmac_fifo_overrun	O	Indicates receive MAC EBR FIFO overran
sts_tmac_fifo_underrun	O	Indicates transmit MAC EBR FIFO underran
sts_pause_active	O	Indicates the transmit flexiMAC is currently paused following reception of a valid pause frame
Ethernet Pause Interface		
pause_req	I	Request from Ethernet MAC client to send a pause frame
pause_count[15:0]	I	Pause time to be sent in the transmitted pause frame
pause_ack	O	Signal to indicate that the pause frame has been transmitted
SMI Interface		
smi_clk	I	SMI Clock
smi_reset_n	I	SMI Reset (active low)
smi_rd	I	SMI read strobe
smi_wr	I	SMI write strobe
smi_wdata	I	SMI serial write data
smi_addr	I	SMI address
smi_rdata	O	SMI serial read data
PCS QUAD to System Bus Interface		
pcs360_out[44:0]	I	Bus from System Bus to PCS QUAD
pcs360_in[16:0]	0	Bus from PCS QUAD to System Bus
Miscellaneous PCS QUAD Control/Status Signals		
felb_0	I	Active high far-end loopback enable for channel 0.
lsm_en_0	I	Receive link state machine enable for channel 0.

Table 11. I/O List of flxmc_sys_ge for 1Gb Ethernet Mode (Continued)

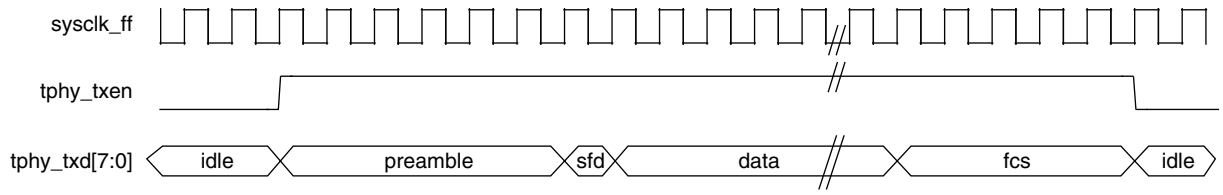
flxmc_sys_ge		Description
Name	**Type**	
lsm_status_0	O	Signal from Channel0 receive link state machine indicating successful link synchronization.
ctc_orun_0	O	Channel0 active high flag indicator that clock tolerance compensation FIFO has overrun.
ctc_urun_0	O	Channel 0 active high flog indicator that clock tolerance compensation FIFO has underrun.

I/O Timing

PHY Interface

Transmit 10Gb GMII

Figure 14. TX GMII Basic Frame Transmission



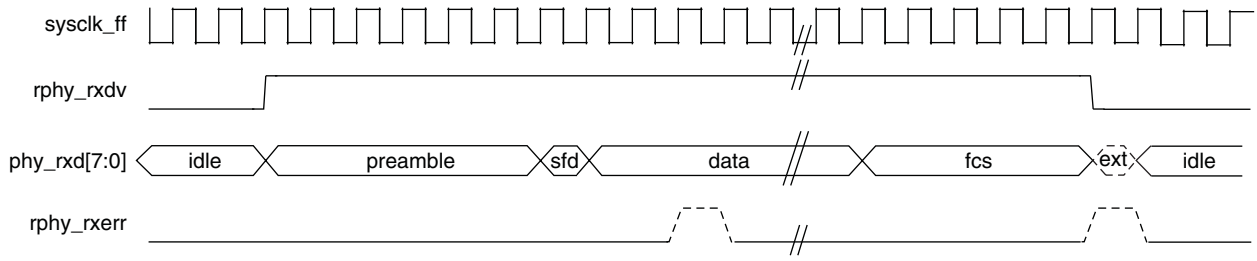
sysclk_ff: is a continuous clock that provides the timing reference for tphy_txen and tphy_txd

tphy_txen: indicates data is present on the GMII for transmission. It is asserted synchronously with the first octet of the preamble and remains asserted while all octets to be transmitted are present on the GMII. tphy_txen is negated prior to the first rising edge of sysclk_ff following the final data octet of a frame.

tphy_txd[7:0]: is an eight-bit data bus that is driven synchronously with respect to the sysclk_ff. For each sysclk_ff period in which tphy_txen is asserted data is presented on tphy_txd for transmission.

Receive GMII Interface

Figure 15. Rx GMII Basic Frame Reception



sysclk_ff: provides the timing reference for the rphy_rxdv, rphy_rxd and rphy_rxerr signals.

rphy_rxdv: is driven by the PCS to indicate presence of recovered and decoded data on the rphy_rxd bus. rphy_rxdv transitions synchronously with respect to the sysclk_ff. rphy_rxdv is asserted continuously from the first recovered octet of the frame through the final recovered octet and is negated prior to the first rising edge of sysclk_ff that follows the final octet.

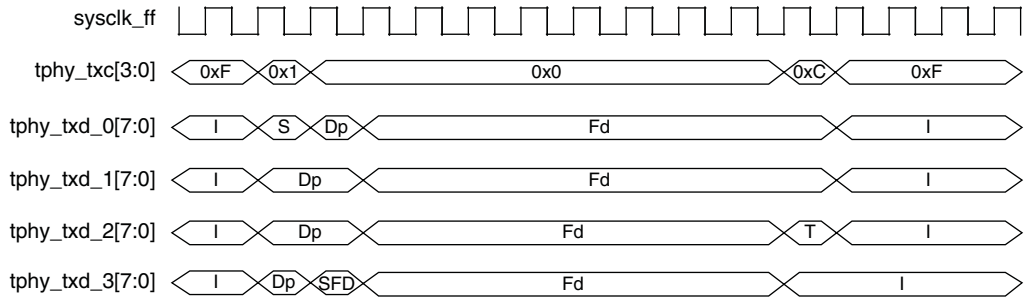
rphy_rxd[7:0]: is an eight-bit data bus that is driven by the PCS. rphy_rxd transitions synchronously with respect to sysclk_ff. For each sysclk_ff period in which rphy_rxd is asserted, rphy_rxd transfers eight bits of recovered data from the PCS. While rphy_rxdv is de-asserted, the PCS may provide a false carrier indication by asserting the rphy_rxerr signal and driving the specific value onto rphy_rxd. While rphy_rxdv is de-asserted and rphy_rxerr is asserted, a specific rphy_rxd value is used to transfer recovered carrier extend from the PCS to the Reconciliation Sublayer (RS) within the LatticeSCM Ethernet flexiMAC core. A carrier extend error is indicated by another specific value of rphy_rxd.

rphy_rxerr: is typically driven by the PCS and transitions synchronously with respect to sysclk_ff. When rphy_rxdv is asserted, rphy_rxerr is asserted for one or more sysclk_ff periods to indicate to the Reconciliation Sublayer that an error was detected somewhere in the frame presently being transferred from the PCS to the RS. As a result, the flexiMAC asserts the rmac_err signal. Driving the appropriate value on rphy_rxd while keeping rphy_rxerr asserted indicates an error within carrier extension. Assertion of rphy_rxerr when rphy_rxdv is de-asserted with specific rphy_rxd values indicates a decode of carrier extension by the PCS. While rphy_rxdv is de-asserted, the PHY may

provide a false carrier indication by asserting the rphy_rxerr signal for at least one cycle of the sysclk_ff-ff while driving the appropriate value onto rphy_rxd.

TX XGMII Interface

Figure 16. TX XGMII Basic Frame Transmission



I : Idle, S : Start, Dp : Data Preamble, SFD : Start of Frame, Fd : Frame Data, T : Terminate

sysclk_ff: provides the timing reference for the transfer of tphy_txc[3:0] and tphy_txd_[0:3] [7:0] signals to e.g. the PCS. The values of tphy_txc and tphy_txd are sampled by the PCS on the rising edge of sysclk_ff.

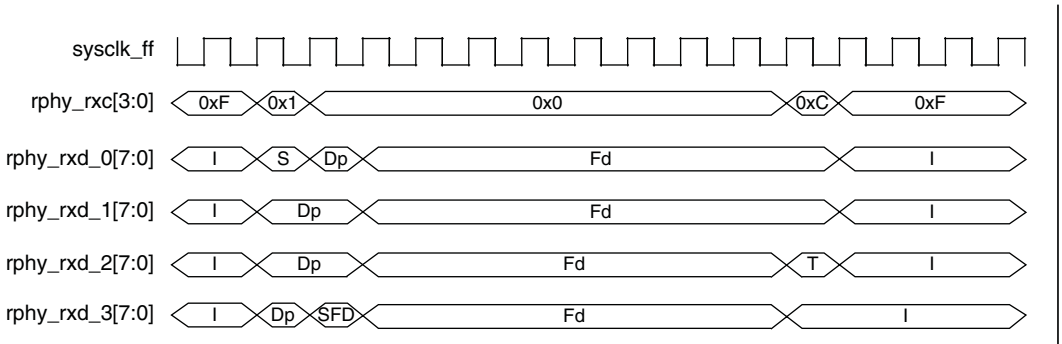
tphy_txc[3:0]: indicates that either data or control characters are present on the XGMII for transmission. The tphy_txc signal for a lane shall be de-asserted when a data octet is being sent on the corresponding lane and asserted when a control character is being sent. In the absence of errors, the tphy_txc signals are de-asserted for each octet of the preamble (except the first octet that is replaced with a start control character) and remain de-asserted while all octets to be transmitted are presented on the lanes of the XGMII. tphy_txc shall transition synchronously with respect to the rising edge of sysclk_ff.

tphy_txd: is a 32-bit data bus organized into four lanes of eight signals each. Each lane is associated with a tphy_txc signal, tphy_txd transitions synchronously with respect to the rising edge of sysclk_ff.

Assertion on a lane of appropriate tphy_txd values when tphy_txc is asserted will cause the PCS to generate code groups associated with either idle, start, terminate, sequence or error control characters. While the tphy_txc of a lane is de-asserted tphy_txd of the lane is used to request the PCS to generate code-groups corresponding to the data octet value of tphy_txd.

Rx XGMII Interface

Figure 17. Rx XGMII Basic Frame Reception



I : Idle, S : Start, Dp : Data Preamble, SFD : Start of Frame, Fd : Frame Data, T : Terminate

sysclk_ff: provides the timing reference for the transfer of the rphy_rxc[3:0] and rphy_rxd_[0:3] [7:0] signals from typically the PCS.

rphy_rxc[3:0]: indicates that the PCS is presenting either recovered and decoded data or control characters on the XGMII. The rphy_rxc signal for a lane is de-asserted when a data octet is being received on the corresponding lane and asserted when a control character is being received. In the absence of errors, the rphy_rxc signals are de-asserted by the PCS for each octet of the preamble (except the first octet that is replaced with a Start control character) and remain de-asserted while all octets to be received are presented on the lanes of the XGMII rphy_rxc are driven by the PCS and shall transition synchronously with respect to the rising edge of sysclk_ff.

rphy_rxd: is a 32-bit data bus organized into four lanes of eight signals each), that are driven typically by the PCS. Each lane is associated with an rphy_rxc signal. rphy_rxd transitions synchronously with respect to the rising edge of sysclk_ff. Assertion on a lane of appropriate rphy_rxd values when rphy_rxc is asserted indicates to the sink module the Start control character, Terminate control character, Sequence control character or Error control character that drive its mapping functions.

rphy_rxc of a lane is asserted with the appropriate Error control character encoding on rphy_rxd of the lane to indicate an error was detected somewhere in the frame presently being transferred from the PCS.

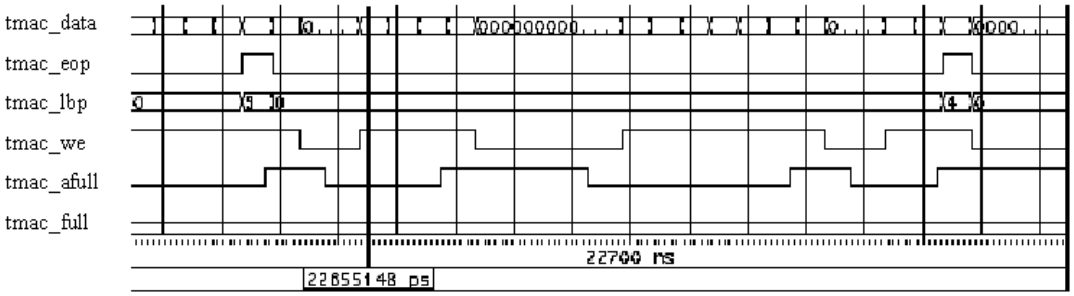
Client Interface

Transmit

Data can be written into the EBR FIFO when tmac_we is active and as long as there is room in the FIFO, i.e. tmac_full is low. The last word to be written is marked by the end of packet marker tmac_eop. At the same time the position of the last valid byte in the word is defined by the last byte position indicator tmac_lbp. For 1Gb Ethernet the data path is 8 bits and the concept of LBP is irrelevant.

Figure 18 shows TX transactions on the client Interface for a 10Gb Ethernet write operation

Figure 18. TX Client Interface for a 10Gb Ethernet Write Operation

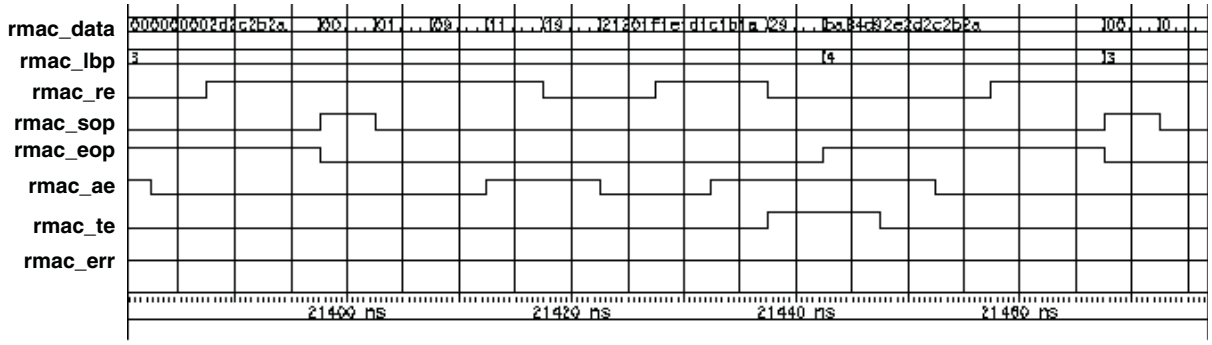


Receive

Data can be read from the EBR FIFO by asserting the read enable rmac_re and providing data is available, i.e. rmac_te (rmac_empty) is low. The start and end of the packet are marked by rmac_sop and rmac_eop respectively. In10Gb E mode the last valid byte position in the word is marked by rmac_lbp.

Figure 19 shows RX transactions on the client Interface for a 10Gb Ethernet read operation

Figure 19. Rx Client Interface for a 10Gb Ethernet Read Operation

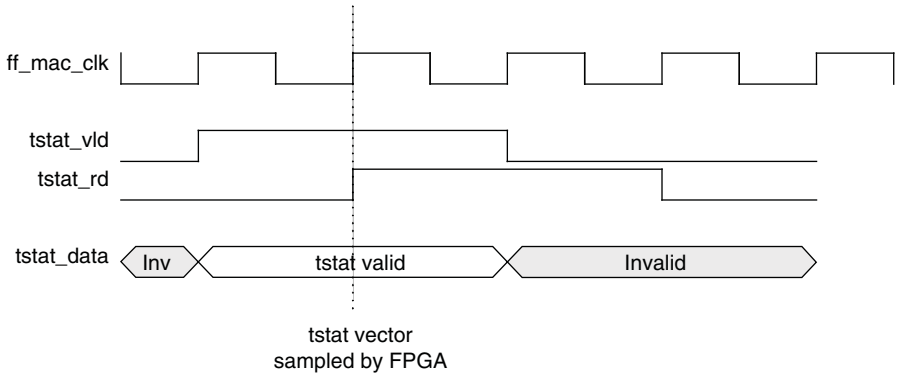


TX Stats Interface

The transmit stats valid signal tstat_vld is asserted after an EOP has been generated.

Statistics for the transmitted packet will remain valid until the client reads the vector. If tstat_rd is not asserted in time (i.e. before another packet generate a vector), sts_tstat_miss will be asserted to indicate a statistics vector was not read by the Client and vector has been missed. The client is not required to generate the tstat_rd signal, and the sts_rstat_miss signal can be ignored if the Client logic is able to sample the TX statistics vector using the tstat_vld signal.

Figure 20. TX Statistics Interface Timing Diagram

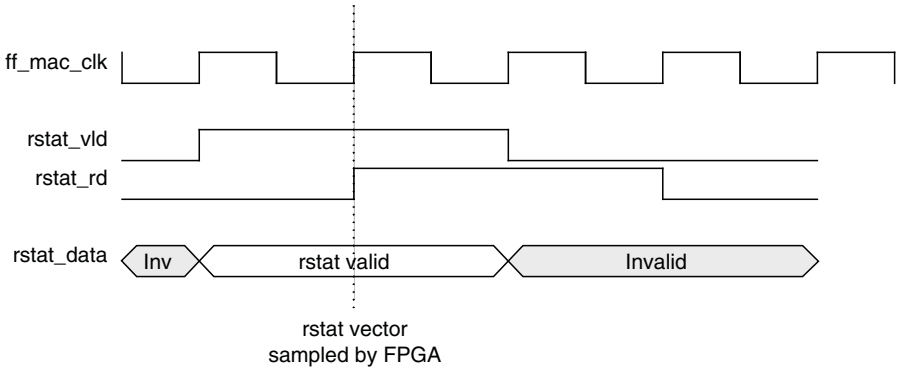


Rx Stats Interface

The read stats valid signal rstat_vld is asserted after an EOP has been received.

Statistics for the received packet will remain valid until the client reads the vector. If rstat_rd is not asserted in time, sts_rstat_miss will be asserted to indicate a statistics vector was not read by the Client and vector has been missed. The client is not required to generate the rstat_rd signal, and the sts_rstat_miss signal can be ignored if the Client logic is able to sample the RX statistics vector using the rstat_vld signal.

Figure 21. RX Statistics Interface Timing Diagram



SMI Timing

Figure 22 illustrates an SMI write transfer, whereas Figure 23 shows an SMI read transfer. The SMI_CLK is sourced from the System Bus at a fourth of the internal System Bus frequency. Each assertion on read/write strobes SMI_RD and SMI_WR will initiate a byte of data transfer (one bit per SMI_CLK cycle). SMI_ADDR[3:0] determine which of the 16 bytes of the targeted MSI interface is being accessed.

To ensure zero hold time, SMI_RD, SMI_WR, and SMI_WDATA are synchronized to the falling edge of SMI_CLK.

As shown in Figure 22, a byte write cycle starts with the assertion of a single cycle SMI_WR pulse on the falling edge of SMI_CLK along with the first SMI_WDATA bit (D00) to be written. The remaining data bits (D01 to D07) will then be written on each subsequent falling edge of SMI_CLK.

As shown in Figure 23, a byte read cycle starts with the assertion of a single cycle SMI_RD pulse on the falling edge of SMI_CLK. The first bit of SMI_RDATA (D00) is expected at the following rising edge. The remaining data bits (D01 to D07) will then be received on each subsequent rising edge of SMI_CLK.

Figure 22. SMI Write Timing Diagram

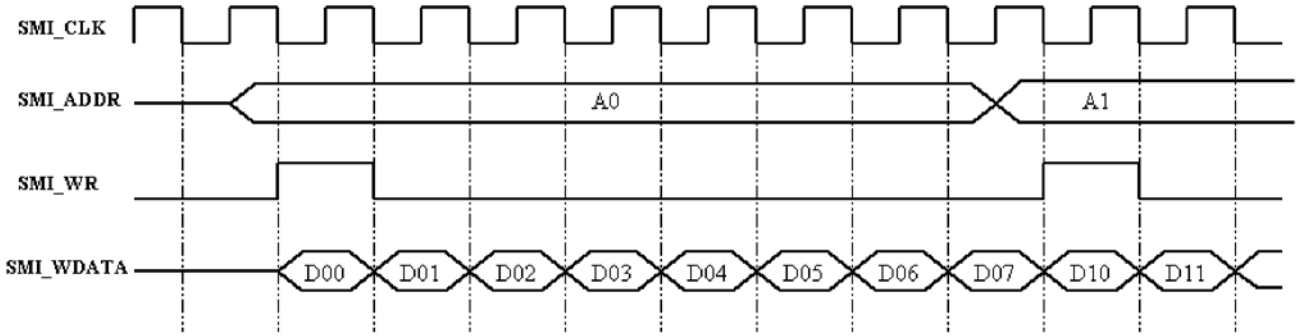
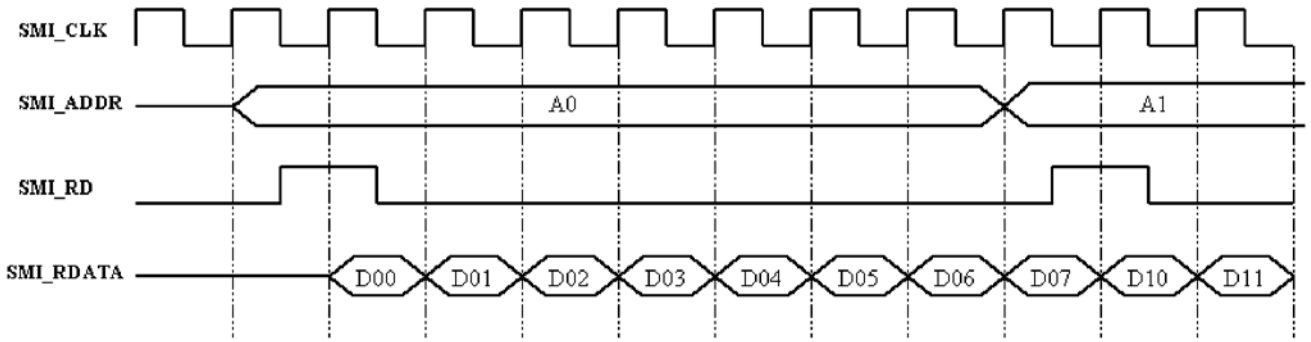


Figure 23. SMI Read Timing Diagram



Reference Information

- [1] IEEE P802.3-2002
- [2] IEEE P802.3ae-2002

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
April 2006	01.0	Initial release.
August 2006	01.1	Added minimum IPG of 16.
		Added more information and pictures on connecting the System Bus to FLXMC for a given SMI_OFFSET. Also documented how SMI_OFFSET is specified for sim/PAR.
		Corrected some information in previous user guide (loopback feature, register addresses,...).
		Clarified the effect of some of the filter control registers (for pause, multicast,..).
		Added 1.14V minimum voltage for 10 Gb Ethernet.
		Documented how MACO + PCS LOCATEs are specified.
		Documented RSTAT BYTE count information for 1 Gb and 10 Gb Ethernet.
January 2007	01.2	Explained 10Gb Ethernet 16-byte IPG requirement (Features bullets).
		Explained how IPG is enforced in the TX flexiMAC direction (flexiMAC Functional Description Overview bullets).
		Fixed typos in smi_pause_sa SMI register addresses (SMI Memory Control Registers table).
March 2007	01.3	Linked EBR voltage requirement to device speed grade (Voltage Requirements section).
June 2007	01.4	Updated "Getting Started" section.

Revision History (Continued)

Date	Version	Change Summary
June 2007 (cont.)	01.4 (cont.)	References to LatticeSC changed to LatticeSCM. Added appendix to support LatticeSCM FPGA family.
October 2007	01.5	Changed rstat_byte description in 1GBE I/O table to match that in 10GbE I/O table.
December 2007	01.6	Added RX EBR Interface Block text section.
May 2008	01.7	Added flexiMAC Location Site Names table.
		Updated Device Arrays with flexiPCS/SERDES and MACO Sites figure.
September 2009	01.8	Updated documentation to coincide with V1.4 of flexiMAC design kit: • New tmac_empty and tmac_aempty signals at core lever. • No IPG restriction in TX direction. • Explained underflow behavior. • Improved RX byte count reporting for 10 Gig mode. • Clarified control vs. data frame distinction based on L/T field. • Provided more explanation for certain statistics and reconciliation sub-layer (10 Gig) signals.

Appendix for LatticeSCM FPGAs

Table 12. Performance and Resource Utilization[1]

IPexpress User-Configurable Mode	SLICEs	LUTs	Registers	External Pins	sysMEM EBRs	f_{MAX} (MHz)
Gigabit	15	4	19	144	2	125
10 Gigabit	42	4	72	278	4	125

1. Performance and utilization characteristics are generated using Lattice's ispLEVER 7.0 software. When using this IP core in a different density, speed, or grade within the LatticeSC/M family, performance and utilization may vary.

Ordering Part Number

All MACO IP, including the Ethernet flexiMAC Core, is pre-engineered and hardwired into the MACO Structure ASIC blocks of the LatticeSCM family of parts. Each LatticeSCM device contains a different collection of MACO IP. Larger devices in the same family will have more instances of the MACO IP. Please refer to the Lattice web pages on LatticeSCM and MACO IP or see your local Lattice Sales office for more information.

All MACO IP is licensed free of charge, however a license feature line is required. See your local Lattice sales office for the license feature line.