LogiCORE IP XPS MOST NIC (v1.01a)

DS638 April 19, 2010 **Product Specification**

Introduction

The LogiCORE Media Oriented Systems Transport (MOST®) Network Interface Controller (NIC) core is a controller designed to the *MOST Specification revision 2.4*. When combined with the Xilinx Automotive solution and embedded processing, the MOST NIC core allows the user to take advantage of the MOST open standard network by providing a higher level of customization in a scalable, flexible design solution.

Features

- Operates in both master and slave modes
- Full bandwidth sustained transfers (24 Mbps)
- Supports full control channel bandwidth utilizing two transmit and receive buffers
- Synchronous, asynchronous, and control channels with 4 to 60 bytes of synchronous data per frame
- Flexible user interfaces
  - 32 bit PLB interface allows use in Xilinx EDK
  - LocalLink-based streaming port for real-time data access
- Physical to logical channel mapping performed in hardware with 16 transmit and 16 receive logical channels
- Parameterizable
  - Provision to select master/slave or slave-only
  - Word count triggers for both egress and ingress channel buffers
- Full support of error and status notification
- Internal loopback mode for diagnostic applications
- Support for Ring Break Diagnosis (RBD) test
- MicroBlaze™ and PowerPC® (405 and 440) processor drivers, and network services from MOCEAN Laboratories, AB (MOCEAN)
- Low-cost recovery PLL available from Integrated Device Technology, Inc. (IDT)
- Available through the Xilinx COREGenerator™ tool

LogiCORE™ IP Facts	
Core Specifics	
Supported Device Family	Spartan®-3A/3A DSP, Spartan-3, Spartan-3E, Automotive Spartan 3/3E/3A/3A DSP, Spartan-6, Virtex®-4 /4Q/4QV
Resources Used	
See Table 72,	
Provided with Core	
Documentation	Product Specification
Design File Formats	VHDL
Constraints File	EDK TCL Generated
Verification	N/A
Instantiation Template	EDK
Design Tool Requirements	
Xilinx Implementation Tools	ISE® 12.1
Verification	ModelSim PE/SE 6.5c and above
Simulation	ModelSim PE/SE 6.5c and above
Synthesis	XST
Support	
Provided by Xilinx, Inc.	

Functional Description

MOST, a growing standard for automotive multimedia networks, is a low-cost, fiber-optic based network that provides integration for passenger infotainment networks. The Xilinx MOST NIC core, in conjunction with the Xilinx Automotive solution and embedded processing, allows designers to take advantage of the MOST open standard network by providing a higher level of customization in a scalable and flexible design solution. The MOST NIC core is fully validated using independent system testing.

The MOST NIC core provides unique support for real-time access to the synchronous portion of the MOST frame through the LocalLink interface (a Xilinx standardized user interface), allowing designs to take advantage of powerful parallel-processing capabilities of the Xilinx FPGA family. The core is delivered with a 32- bit PLB bus suitable for both standalone and embedded applications. A complete hardware and software solution is realized when used in conjunction with a MicroBlaze soft processor or a hard PowerPC processor solution, drivers, and MOCEAN Network Services.

Feature Summary

The following subsections list the key features of the MOST NIC core.

Master/Slave Configuration

The MOST NIC core can be configured for master/slave (`C_OPMODE = 0`) or slave-only (`C_OPMODE = 1`) mode operation. The master/slave core can be used either as a master or as a slave depending on the user application. When only slave functionality is required, the unused logic is removed automatically for optimal resource utilization in the slave-only core.

- **Master/Slave**. When configured for master/slave mode, the MOST NIC core supports full-ring master capabilities including clock generation and control message initiation. All slave operations are also fully supported. An external clock source is required for clock generation
- **Slave-only**. Slave-only is used for the majority of MOST controllers. The slave mode supports the transmission and reception of all MOST data types (synchronous, asynchronous, and control). The clock is recovered from the incoming MOST stream. An external PLL is required for clock recovery and data alignment

Full Synchronous Channel

- The MOST frame data is received on a time slot (or physical channel) basis. The MOST NIC supports a range of 4 through 60 time slots of synchronous data. The Synchronous Boundary Descriptor (SBD) sets the boundary between the synchronous and asynchronous data in 4 byte increments, or quadlets
- The total amount of synchronous and asynchronous data available in a frame is 60 bytes. For this reason, the asynchronous portion of the frame is 60 bytes minus the synchronous data (`60-(SBD*4)` bytes)

The remainder of the usable data in the MOST frame (2 bytes) is reserved for control data.

Streaming Port

The MOST NIC core supports a streaming port that provides access to the synchronous data portion of the MOST stream in real time without requiring processing by the host. This port allows both *ingress* (write to the NIC) and *egress* (read from the NIC) transactions. Both transmit and receive data can be accessed by logical channel (see "Logical Channel Mapping" on page 3 for more information). The data is available *per byte* for this Xilinx LocalLink interface.

- **Receive**: The Receive Streaming port can be used to optionally intercept and/or insert data into the MOST NIC receive path. This allows for additional processing of the data where the external module can act as a sink, source, or a preprocessor of the stream, to offload operations from the host
- **Transmit**: The Transmit Streaming port can be optionally used to intercept and/or insert data into the MOST NIC transmit path. This allows for additional processing of the data where the external module can act as a sink, source, or a preprocessor of the stream, offloading operations from the host
- **Register Control**: The MOST NIC core contains dedicated memory-mapped register space for the streaming port. Any accesses to these locations is forwarded to the streaming port. The external module can then implement control registers, accessible through the MOST NIC core. Typical applications for the streaming port are synchronous data encryption and decryption

Logical Channel Mapping

Audio or video data is spread over several synchronous time slots, and as such, the grouping of related time slots into a logical channel is efficient for software processing. The MOST NIC core groups physical time slot data into logical channel data under user control and automatically synchronizes the multiple time slots on a per-frame basis. The MOST NIC core contains 32 logical channel buffers: 16 reserved for transmit operations and 16 for receive operations. A logical channel can aggregate from one to 60 time slots in any of the synchronous physical channels or can be mapped to the asynchronous channel.

For the receive path, four logical channels are dedicated to streaming port ingress access, four logical channels for streaming port egress, and eight logical channels for PLB receive. Similarly, for the transmit path, four logical channels are dedicated to streaming port ingress access, four logical channels for streaming port egress, and 8 logical channels for PLB transmit. Logical channel 0 is reserved for asynchronous data in both directions.

Each logical channel buffer is capable of storing up to 32 words of 32 bits each. All buffers can assert interrupts to an external microprocessor to indicate that the buffer requires processing. The word count at which the interrupt is asserted is user-configurable.

The MOST NIC core provides a host interface for access to control and status registers. The interface is a 32-bit wide PLB bus. The PLB bus is also used to transfer transmit data and receive MOST frame data.

Loopback

The transmit path can be internally connected to the MOST controller receive path for diagnostic tests, allowing users to receive all frames transmitted. Users can validate that the configuration of the core and the data path are as expected before its insertion on the MOST ring.

Applications

The MOST NIC core can be used in many applications to interface to a MOST network. The core can function as a standalone MOST interface or be combined with other LogiCORE IP and EDK cores to build a variety of embedded systems. The MOST NIC solution allows flexible partitioning between software and hardware functions.

Applications include:

- A customized, scalable MOST network controller that unburdens the host processor from network- specific overhead which is ideal for applications demanding maximum host application performance

- Augmentation to the MOST NIC core with custom IP to process the MOST streaming data with no host processor overhead

The following sections describe common MOST NIC applications in conjunction with additional Xilinx IP cores.

Basic MOST Controller

A typical application for a basic MOST network interface using the MOST NIC and MicroBlaze processor cores is shown in Figure 1. The MicroBlaze processor core contains the MOST driver and the MOCEAN Network Services stack. The MOST NIC core supports transmission and reception of synchronous data, asynchronous data, and control messages.

Synchronous and asynchronous data are aggregated and mapped by the core from physical channels (synchronous channel time slots of one byte each or asynchronous data) into logical channels. Audio transmission is distributed among several discrete *quadlets* on the MOST bus. Grouping these related quadlets into logical channels allows software to process the data more efficiently. In this application, a variety of data types are supported for transmission and reception on the MOST network including audio, video, and control types.

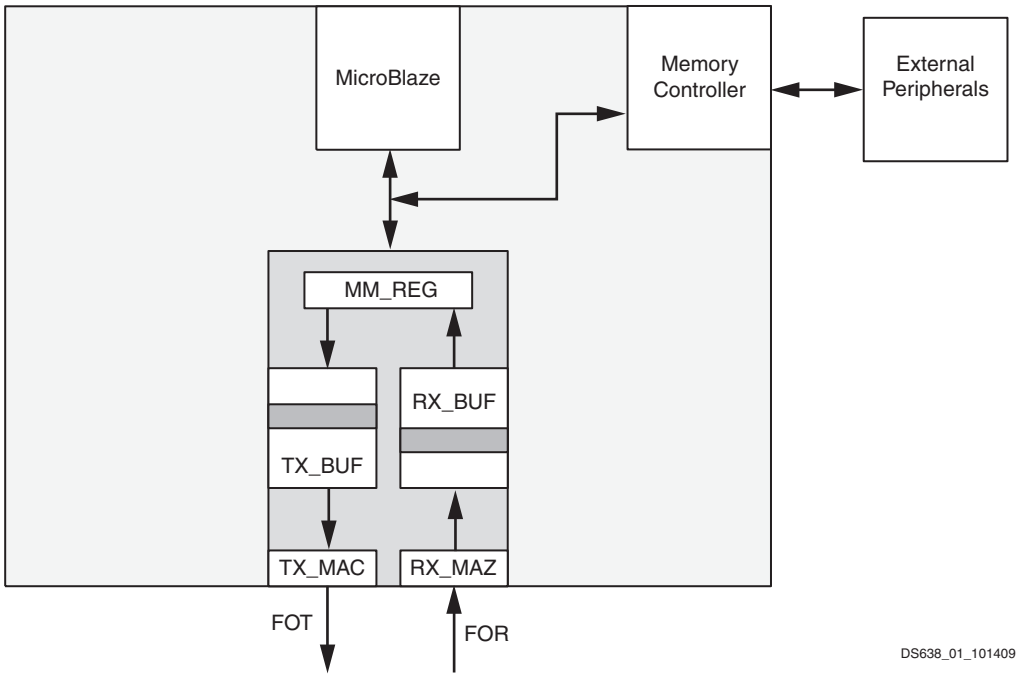

Figure 1: **Basic MOST NIC Configuration**

Audio / Video Encryption and Decryption

An application involving the MOST NIC core streaming port is shown in Figure 2. The streaming port allows an external core to process the synchronous data portion of the MOST stream in real time. Synchronous channel data can be accessed over the streaming port on a logic channel basis. In this example, a decryption core accesses received synchronous data using the streaming port. After decryption is performed, the decrypted data is written to the MOST NIC core receive buffer using the streaming port.

The received data is then accessed using the user interface and processed using software running on the MicroBlaze and (or) Power PC processors. Similarly, transmit data is encrypted by an encryption core using the streaming port. The encryption core reads unencrypted synchronous transmit data, encrypts it, and then writes the encrypted data to the MOST NIC core transmit buffer using the streaming port. The encrypted data is then queued for transmission on the MOST ring.

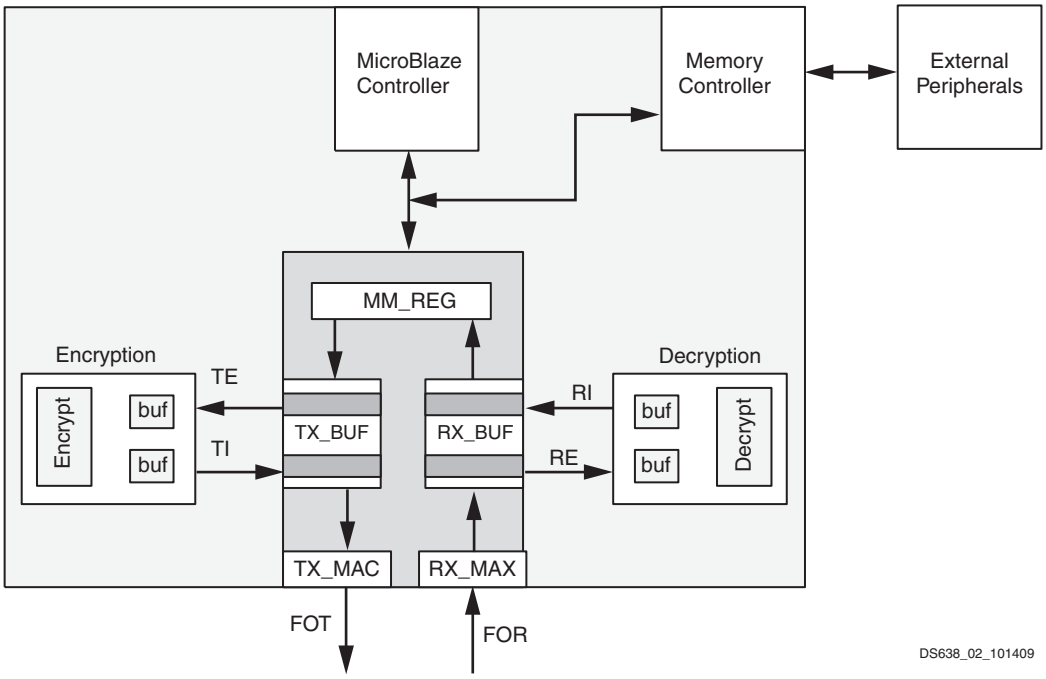

Figure 2: **Encryption and Decryption Using the Streaming Port**

Standalone MOST NIC

The following figure illustrates a standalone MOST NIC. This example uses the MOST NIC core, MicroBlaze and (or) PowerPC processor cores, Centralized DMA controller, I^2C, and S/PDIF cores. The MOST NIC transmits and receives synchronous, asynchronous, and control messages to and from the

MOST network. Data is organized on a logical channel basis and transferred between system buffers using the 32-bit PLB bus.

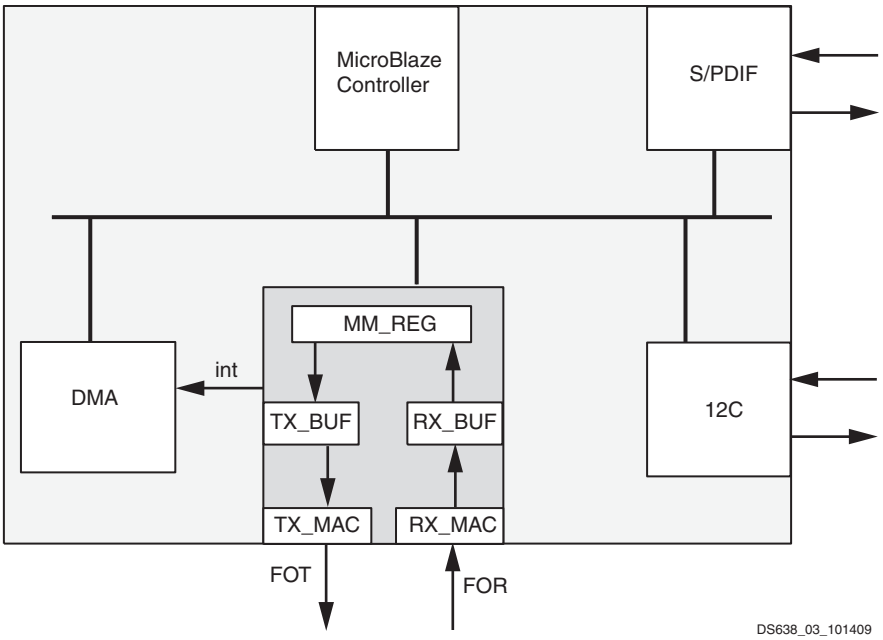

Figure 3: **Standalone MOST NIC**

Synchronous data is synchronized and sourced from external multimedia devices through the S/PDIF core. Control data is transferred to and from external devices through the I^2C core. The Centralized DMA controller manages the routing of data between system memory, the MOST NIC, S/PDIF and I^2C cores.

Functional Overview

The MOST NIC core is a full-featured soft IP core incorporating all necessary logic to interface to a MOST ring. The core supports the transmission and reception of synchronous, asynchronous, and control data to and from the MOST ring.

The following subsections describe:

- "Module Architecture"
- "Receive Routing Engine"
- "Receive Buffer"
- "Transmit Buffer"
- "Transmit Routing Engine"
- "Transmit MAC"
- "Transmit Bypass"
- "Common Routing Table"
- "Control Processor"
- "Memory Mapped Registers"

Module Architecture

The major sub-modules of the MOST NIC are shown in Figure 4.

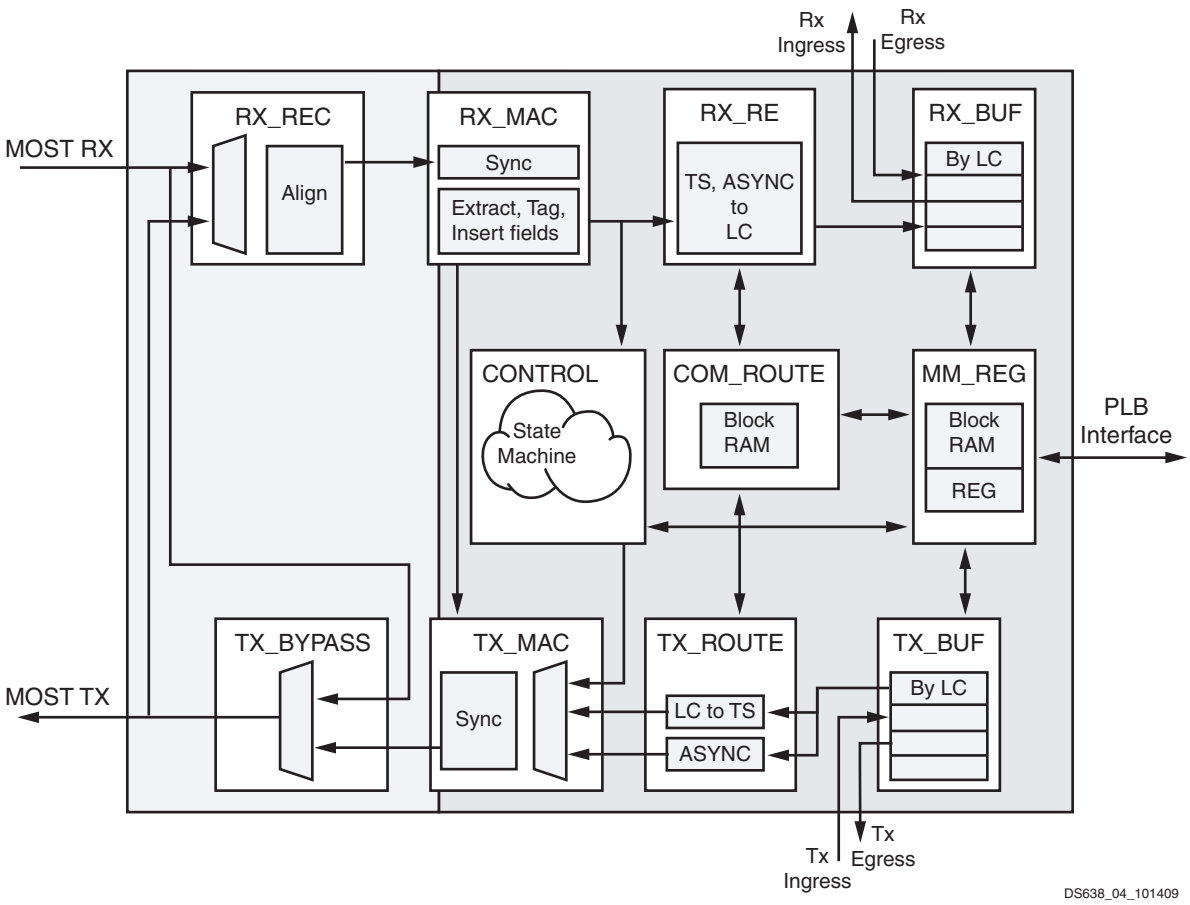

Figure 4: **MOST NIC Block Diagram**

Receive Recovery

The Receive Recovery (RX_REC) module realigns the incoming MOST data to the recovered and corrected MOST clock from the off-chip PLL.

Receive MAC

The Receive MAC (RX_MAC) module decodes the received MOST frame. The RX_MAC module has several functions including frame decode, serial-to-parallel conversion, and timing decode.

Receive Routing Engine

The Receive Routing Engine (RX_RE) receives the decoded MOST frame and allocates data to the appropriate buffer for further processing. The RX_RE module utilizes a shared lookup table which contains the following information:

- Indication of which data to keep and which to discard
- Mapping of received synchronous time slot data to logical channels

Asynchronous and synchronous data is written to the appropriate location in the receive buffer (RX_BUF) based on the look up table.

Receive Buffer

The Receive Buffer (RX_BUF) contains sufficient storage for synchronous and asynchronous receive data. Data is organized on a logical channel basis.

Transmit Buffer

The Transmit Buffer (TX_BUF) contains sufficient storage for synchronous and asynchronous transmit data. Data is organized on a logical channel basis.

Transmit Routing Engine

The Transmit Routing Engine (TX_RE) maps logical channels to time slots.

Transmit MAC

The Transmit MAC (TX_MAC) module encodes synchronous, asynchronous and control data into the transmit MOST frame. This module has several functions including frame encode and parallel to serial conversion.

Transmit Bypass

The Transmit Bypass (TX_BYPASS) module muxes in the receive serial stream, if the MOST NIC core is operating in a bypass mode.

Common Routing Table

This Common Routing Table (COM_ROUTE) module is a common RAM resource shared by both the receive and transmit paths as a look-up table.

Control Processor

The control processor (CONTROL) handles control messages. Processing of control messages is contained within the MOST NIC core. This sub-module decodes the control messages and also generates a suitable response to received control messages. Any received normal message is forwarded through the Memory Mapped Register to the external microprocessor.

The control processor can hold up to two receive messages and up to two transmit messages in addition to the most recent transmit message with the response received on the ring. Any messages that require external processing are forwarded via the Memory Mapped Register interface. These messages are processed using an interrupt driven register interface.

Memory Mapped Registers

The Memory Mapped Register (MM_REG) module contains all registers for the control and to determine the status of the MOST NIC. All registers are 32 bits wide.

Clocking and Reset

Clocking

The MOST Controller contains three input clocks: MOST Clocks (MOST_PLL_CLK, MOST_COM_CLK) and PLB Clock (SPLB_Clk). The following conditions apply to clock frequencies:

- MOST_PLL_CLK and MOST_COM_CLK can be either 45.1584 MHz or 49.152 MHz and are frequency- locked to each other

- SPLB_Clk is asynchronous to the MOST clocks, and may have a minimum clock frequency of 0.7 times the MOST_PLL_CLK. The maximum frequency will depend on the device selected, but all devices will allow at least 75 MHz. The requirements for the DCM must also be met
- An external clock source is required for MOST_PLL_CLK and MOST_COM_CLK in Master mode
- An external clock and data recovery PLL is required for MOST_PLL_CLK operation
- The streaming port clock (STR_CLK) is an output and is identical to the PLB Clock, where the core forwards this clock on behalf of the user
- MOST_RX is sampled at negative edge of MOST_PLL_CLK

Reset Mechanism

Two reset mechanisms are provided for the MOST NIC: the SPLB_Rst input is a hard reset (Table 1 on page 12), and a software-controlled soft reset is provided through a user configuration register. Both the hard and soft resets cause the logic within the MOST NIC to return to the default state.

Hard Reset

Assertion of SPLB_Rst causes the logic within the MOST NIC to return to the default state. Configuration registers which are not located in block RAM are returned to their default values as indicated in the memory map description. Read/Write transactions cannot be performed while the SPLB_Rst input is asserted.

Soft Reset

Assertion of the soft reset signal causes all logic within the MOST NIC to return to the default state. Configuration registers not located in block RAM with the exception of the soft reset control register are returned to their default values as indicated in the memory map description. Read/Write PLB transactions can be performed starting at the next valid transaction window.

Core Interfaces

This section describes the interface signals of the MOST NIC core. The major MOST NIC interfaces include the following:

- MOST Ring Interface
- Host Interface
- Streaming Port Interface

MOST NIC Interfaces

The following figure illustrates the MOST NIC interfaces. All signals are defined in their respective sections following the illustration.

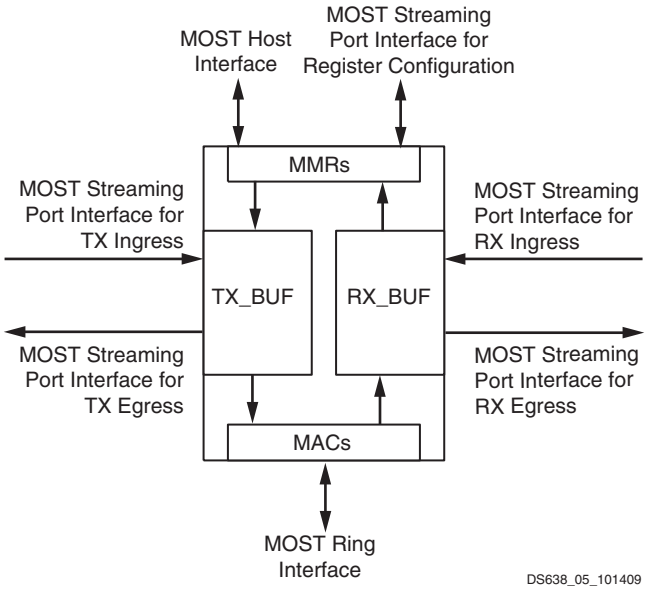

Figure 5: **MOST Core Interfaces**

MOST Ring Interface

Table 1 on page 12 defines the signals (MOST_) interfacing to the MOST ring via an external PHY and DCM.

Host Interface

Table 1 on page 12 defines the MOST host interface signals (`PLB_*/Sln_*`). The MOST NIC contains a 32 bit PLB slave interface bus to interface to the PowerPC, MicroBlaze, or other microprocessors. There are separate read and write data buses with a shared address bus. The host interface is used to access control registers as well as to transmit and receive data.

Byte Enable Functionality

The MOST NIC core implements a superset of the standard PLB interface with regards to byte enables. All PLB accesses are considered long word aligned accesses, that is, bits 0 and 1 of the address are ignored, and assumed to be 00b. The byte enable signals are used to qualify which byte(s) are selected within the long word access as follows:

- If `PLB_BE[0]` is asserted, `PLB_wrDBus[0:7]` are selected for access
- If `PLB_BE[1]` is asserted, `PLB_wrDBus[8:15]` are selected for access
- If `PLB_BE[2]` is asserted, `PLB_wrDBus[16:23]` are selected for access
- if `PLB_BE[3]` is asserted, `PLB_wrDBus[24:31]` are selected for access

The `PLB_BE[0:3]` bus can be driven with any value from 0x0 to 0xF

Burst Support

The PLB interface accepts both burst and single cycle transactions to the control registers. However, only the TX Buffer (TXBUFF), RX Buffer (RXBUFF), Common Routing Table (CRT), and Master Allocation Tables (MAT) provide optimized pipelined data acknowledgement for burst requests. Burst requests to all other registers are treated as consecutive single-cycle access

Streaming Port

Table 1 on page 12 defines the MOST NIC steaming port signals (STR_*). The streaming port has five main components: the Memory Map Register and the four Data Transfer interfaces.

Memory Map Register

A subset of the MOST NIC memory map is reserved for the streaming port. Access to these locations are forwarded to the streaming port. An external module interface to the streaming port can then implement control register specific to that module (but accessed through the MOST NIC memory map). This group also contains an interrupt port such that the external module can assert an interrupt source within the MOST NIC.

Data Transfer

The MOST NIC core includes one receive and one transmit streaming port based on the LocalLink interface for data transfer. The streaming ports are used to access synchronous channel data in the transmit and receive FIFOs on a logical channel basis. Both the transmit and receive streaming ports have ingress and egress capabilities.

As a general rule, the logical channel width is 4 bits. However, for the streaming ports, the logical channels are split by port type. Internal to the core, all egress logical channel values have a prefix of 11b and all ingress logical channel values have a prefix of 10b.

The streaming port contains interrupt signals to indicate logical channel buffer word count status. The word count interrupt signals indicate when the logical channel buffers have enough data to be processed, as defined by the full and empty word count parameters in Table 4 on page 19. The interrupt is a one-clock wide active high pulse synchronous to the STR_CLK.

The interrupts are generated by this controller and processed as required by the external module. In the case of egress buffers, an interrupt is asserted when there is sufficient data in the FIFO that can be read by the streaming port. In the case of ingress buffers, an interrupt is asserted when there is sufficient free space in the FIFO to allow for additional write data from the streaming port.

Timing diagrams for Streaming Port Read and Streaming Port Write are shown in Figure 11 on page 25 and Figure 12 on page 57 respectively.

I/O Signals

The following table describes the I/O signals for the XPS MOST NIC.

Table 1: **XPS MOST NIC I/O Signal Descriptions**

Signal Name	Signal Direction	Description
MOST Ring Interface		
MOST_TX	Output	**Transmit MOST Data**: The serial bit stream transmitted onto the MOST ring.
MOST_RX	Input	**Receive MOST Data**: The serial bit stream received from the MOST ring.
MOST_COM_CLK	Input	**MOST Common Clock**: The clock used internally to the core. For a slave-only core, the MOST common clock can be connected to MOST_PLL_CLK. For a master/slave core, the source when in Master mode is the external crystal, and the source when in slave mode is MOST_PLL_CLK.
MOST_PLL_CLK	Input	**MOST PLL Clock**: The clock recovered or created by the external PLL. Received data is sampled with this clock.
MOST_EXT_NRESET	Output	**MOST External Not-Reset**: This active low signal can be used to drive a reset externally to the core. It is suggested to connect this to the reset of the PLL, in order to reset the PLL directly from software.
MOST_EXT_BYPASS	Output	**MOST External Bypass**: This signal can be used to control the state of the external PLL by placing it into bypass mode. This signal is not related to the programming of the MODE select register.
MOST_FOR_STATUS	Input	**Fiber Optic Receiver Status**: Indicates availability of the external FOR.
MOST_PLL_LOCK	Input	**PLL Lock**: The external PLL lock status negates this signal to indicate the receive clock is locked. This connection is not mandatory as the core examines the quality of the input clock to detect a loss of lock.
MOST_MASTER	Output	**MOST Master**: Indicates if this node is acting as the master. This is used to control the input of the off chip PLL to be either the RX line (for acting slaves) or an off chip crystal (for acting masters). This is applicable for master configured nodes as well as slaves in Ring Break Diagnostics acting as a master.
MOST Host Interface		
SPLB_Clk	Input	**Clock**: All host interface signals are synchronous to this clock.
SPLB_Rst	Input	**Reset**: The MOST core is reset to the default state upon assertion of this signal.
MOST_Irpt	Output	**Interrupt**: Asserted to indicate a MOST Controller Status interrupt condition to the microprocessor or interrupt controller.
BUFFER_Irpt	Output	**Interrupt**: Asserted to indicate a Buffer interrupt condition to the microprocessor or interrupt controller.
PLB Master Interface Signals		
PLB_PAValid	Input	**Select**: Indicates an active read or write access. This signal qualifies all bus inputs from the PLB master.
PLB_RNW	Input	**Read not Write**: ‘1’ --> Read access; ‘0’ --> Write access.

Table 1: **XPS MOST NIC I/O Signal Descriptions *(Cont'd)***

Signal Name	Signal Direction	Description
PLB_rdBurst	Input	**Read Burst Transfer**: Indicates that the transfer being performed will be followed with a transfer to the next sequential address in the same direction, read. For any given burst transaction, PLB_rdBurst needs to be asserted for every data except the last
PLB_wrBurst	Input	**Burst Transfer**: Indicates that the transfer being performed will be followed with a transfer to the next sequential address in the same direction, write. For any given burst transaction, PLB_wrBurst needs to be asserted for every data except the last
PLB_ABus[0:31]	Input	**Address**: Bus used to specify the address being accessed either for a read or write.
PLB_wrDBus[0:31]	Input	**Write Data Bus**: Data to be written to the address specified by either PLB_ABus (for single transfer) or generated address (for line, sequential and fixed length burst). The write is acknowledged by Sl_wrDAck and Sl_wrComp when complete.
PLB_BE[0:3]	Input	**Byte Enable**: Selects which byte lane of the data bus is being accessed.
PLB_size[0:3]	Input	**Transfer Size**: Indicates the size of requested transfer. "0000" --> Single transfer "1010" --> Burst transfer Others --> Not supported.
PLB_masterID[0:3]	Input	**Master Identification**: Indicates the identification of the master of the current transfer. Sl_MBusy, Sl_MRdErr and Sl_MWrErr must be driven using this identification.
PLB_type[0:2]	Input	**Transfer Type**: Indicates the type of transfer requested. Supported transfer type is memory transfer ("000").
PLB_Msize[0:1]	Input	**Master Size**: Indicates the data bus width of the associated master. "00" --> 32 bit master "01" --> 64-bit master "10" --> 128-bit master "11" --> 256-bit master. This is not supported by Xilinx
PLB Slave Interface Signals		
Sl_addrAck	Output	**Address Acknowledge**: When asserted, indicates that the address is latched. This is delayed PLB_PAValid generated as pulse. All the PLB input signals must latched as they will not be available after Sl-addrAck is asserted.
Sl_SSize[0 : 1]	Output	**Write Data Acknowledge**: Indicates the slave size. This is always "00" as the slave supported is 32 bit.
Sl_wrDAck	Output	**Write Data Acknowledge**: When asserted, indicates that the data currently on the PLB_wrDBus is no longer required
Sl_wrComp	Output	**Write Data Complete**: When asserted, indicates the end of the current write transfer
Sl_rdDBus[0 : C_SPLB_DWIDTH - 1]	Output	**Read Data**: Data to be written to the address specified by either PLB_ABus (for single transfer) or generated address (for line, sequential and fixed length burst). Data is valid when a read request followed by an acknowledge (Sl_rdDAck) is asserted. Data is mirrored for 64 and 128 data width.
Sl_rdWdAddr[0:3]	Output	**Read Word Address**: Indicates the word address within the line of data requested of a data word transferred as part of a read line transfer.

Table 1: **XPS MOST NIC I/O Signal Descriptions *(Cont'd)***

Signal Name	Signal Direction	Description
Sl_rdDAck	Output	**Read Data Acknowledge**: When asserted, indicates that the data currently on the Sl_rdDBus is valid.
Sl_rdComp	Output	**Read Data Complete**: When asserted, indicates that the read transfer is either complete or will be complete in the next clock cycle.
Sl_MBusy[0 : C_SPLB_NUM_MASTERS - 1]	Output	**Busy**: Indicates that the slave is busy in performing read or write transfer. Only Sl_MBusy[PLB_masterID] must be asserted and remaining bits must be driven zero.
Sl_MWrErr[0 : C_SPLB_NUM_MASTERS - 1]	Output	**Write Error**: Indicates that the write transfer has encountered an error. Only Sl_MWrErr[PLB_masterID] must be asserted and remaining bits must be driven zero.
Sl_MRdErr[0 : C_SPLB_NUM_MASTERS - 1]	Output	**Read Error**: Indicates that the read transfer has encountered an error. Only Sl_MRdErr[PLB_masterID] must be asserted and remaining bits must be driven zero.
Unused PLB Signals		
PLB_UABus[0:31]	Input	Signal is available in port list for compliance purpose, but ignored internally.
PLB_SAValid	Input	Signal is available in port list for compliance purpose, but ignored internally.
PLB_rdPrim	Input	Signal is available in port list for compliance purpose, but ignored internally.
PLB_abort	Input	Signal is available in port list for compliance purpose, but ignored internally.
PLB_busLock	Input	Signal is available in port list for compliance purpose, but ignored internally.
PLB_TAttribute[0:15]	Input	Signal is available in port list for compliance purpose, but ignored internally.
PLB_lockerr	Input	Signal is available in port list for compliance purpose, but ignored internally.
PLB_wrPendReq	Input	Signal is available in port list for compliance purpose, but ignored internally.
PLB_rdPendReq	Input	Signal is available in port list for compliance purpose, but ignored internally.
PLB_wrPendPri[0:1]	Input	Signal is available in port list for compliance purpose, but ignored internally.
PLB_rdPendPri[0:1]	Input	Signal is available in port list for compliance purpose, but ignored internally.
PLB_reqPri[0:1]	Input	Signal is available in port list for compliance purpose, but ignored internally.
Sl_wait	Output	Signal is available in port list for compliance purpose, but ignored internally.
Sl_rearbitrate	Output	Signal is available in port list for compliance purpose, but ignored internally
Sl_wrBTerm	Output	Signal is available in port list for compliance purpose, but ignored internally.

Table 1: **XPS MOST NIC I/O Signal Descriptions *(Cont'd)***

Signal Name	Signal Direction	Description
SI_rdBTerm	Output	Signal is available in port list for compliance purpose, but ignored internally.
SI_rdWdAddr[0:3]	Output	Signal is available in port list for compliance purpose, but ignored internally.
SI_MIRQ[0:C_SPLB_NUM_MASTERS-1]	Output	Signal is available in port list for compliance purpose, but driven with zero.
Common to All Interfaces		
STR_CLK	Output	**Clock:** All signals on the streaming port are synchronous to this clock.
STR_RST	Output	**Reset:** An active high reset signal that is asserted when ever the MOST core is reset (by either hard or soft reset).
Memory map		
STR_MMR_REQ	Output	**Memory Map Request:** Assertion indicates an active read or write to the streaming port memory mapped signals. The REQ signal qualifies all other streaming port memory map signals. The REQ signal is deasserted when the transfer is complete (STR_MMR_ACK) sampled high.
STR_MMR_RNW	Output	**Memory Map Read / Write:** Indicates the direction of the memory map transfer. Logic 1 indicates reads. Logic 0 indicates writes. Asserted coincident with STR_MMR_REQ. The STR_MMR_RNW signal must remain constant during the memory map access.
STR_MMR_ADDR[0:7]	Output	**Memory Map Address:** The address being written to or read from.
STR_MMR_BE[0:3]	Output	**Memory Map Byte Enable:** Selects which byte lane of the data bus is being accessed.
STR_MMR_WDATA[0:31]	Output	**Memory Map Write Data:** The data being written to the streaming port. Asserted coincident with STR_MMR_REQ. The STR_MMR_WDATA signals will remain constant during the memory map access. This signal is only valid when MMR_RNW = 0.
STR_MMR_RDATA[0:31]	Input	**Memory Map Read Data:** Data read from the streaming port external module. Data is valid when a read request followed by an acknowledgment STR_MMR_ACK is asserted.
STR_MMR_ACK	Input	**Memory Map Acknowledge:** Asserted by the target to indicate completion of a write or read cycle. When asserted the cycle is complete and returned read data is latched. The STR_MMR_REQ signal is deasserted immediately after assertion of ACK to complete the transaction.
STR_MMR_INT	Input	**Interrupt In:** The target asserts this signal for 1 clock duration to set the MOST interrupt status register bit assigned to the streaming port. If enabled this will cause an interrupt to the external microprocessor. The STR_MMR_INT signal must be asserted for 1 clock duration only.
Data Transfer: Receive Streaming Port-Egress		
STR_RE_LC[0:1]	Input	**Receive Egress Logical Channel:** Logical channel for the data being requested.
STR_RE_DATA [0:7]	Output	**Receive Egress Data:** Data read from to the logical channel selected on STR_RE_LC. Data is valid when a read request followed by an acknowledgment by requestor (SRC_RDY) and target (DST_RDY) is asserted.

Table 1: **XPS MOST NIC I/O Signal Descriptions *(Cont'd)***

Signal Name	Signal Direction	Description
STR_RE_BIF_AVAIL[0:3]	Output	**Receive Egress Buffer Interface Available**: When asserted, data is available in the given logical channel buffer. This signal is deasserted when the core is not enabled and triggers low in the event of the logical channel flush from the Host Interface.
STR_RE_SRC_RDY	Output	**Receive Egress Source Ready:** Read data on the STR_RE_DATA is available and valid.
STR_RE_DST_RDY	Input	**Receive Egress Destination Ready:** The requestor is ready for data for this logical channel (STR_RE_LC).
STR_RE_WCINT[0:3]	Output	**Receive Egress Interrupt:** Asserted for 1 clock when the receive egress logical channel buffer has crossed the word count as configured by the full word count parameter (FWC). There is one signal for each receive read logical channel.
Data Transfer: Receive Streaming Port-Ingress		
STR_RI_LC[0:1]	Input	**Receive Ingress Logical Channel:** Logical channel for the data being written.
STR_RI_DATA[0:7]	Input	**Receive Ingress Data:** Data being written the logical channel selected on STR_RI_LC. Data is accepted when a write request followed by an acknowledgment by requestor (SRC_RDY) and target (DST_RDY) is asserted.
STR_RI_BIF_AVAIL[0:3]	Output	**Receive Ingress Buffer Interface Available**: When asserted, data is available in the given logical channel buffer. This signal is deasserted when the core is not enabled and triggers low in the event of the logical channel flush from the Host Interface.
STR_RI_SRC_RDY	Input	**Receive Ingress Source Ready:** Write data on the STR_RI_DATA is available and valid.
STR_RI_DST_RDY	Output	**Receive Ingress Destination Ready:** The target has accepted the write data for this logical channel (STR_RI_LC).
STR_RI_WCINT[0:3]	Output	**Receive Ingress Interrupt:** Asserted for 1 clock when the receive ingress logical channel buffer has crossed the word count as configured by the empty word count parameter (EWC). There is one signal for each receive write logical channel.
Data Transfer: Transmit Streaming Port-Egress		
STR_TE_LC[0:1]	Input	**Transmit Egress Logical Channel:** Logical channel for the data being requested.
STR_TE_DATA [0:7]	Output	**Transmit Egress Data:** Data read from the logical channel selected on STR_TE_LC. Data is valid when a read request followed by an acknowledgment by requestor (SRC_RDY) and target (DST_RDY) is asserted.
STR_TE_BIF_AVAIL[0:3]	Output	**Transmit Egress Buffer Interface Available**: When asserted, data is available in the given logical channel buffer. This signal is deasserted when the core is not enabled and triggers low in the event of the logical channel flush from the Host Interface.
STR_TE_SRC_RDY	Output	**Transmit Egress Source Ready:** Read data on the STR_TE_DATA is available and valid.
STR_TE_DST_RDY	Input	**Transmit Egress Destination Ready:** The requestor is ready for data for this logical channel (STR_TE_LC).

Table 1: **XPS MOST NIC I/O Signal Descriptions *(Cont'd)***

Signal Name	Signal Direction	Description
STR_TE_WCINT[0:3]	Output	**Transmit Egress Interrupt:** Asserted for 1 clock when the transmit egress logical channel buffer has crossed the word count as configured by the full word count parameter (FWC). There is one signal for each transmit read logical channel.
Data Transfer: Transmit Streaming Port-Ingress		
STR_TI_LC[0:1]	Input	**Transmit Ingress Logical Channel:** Logical channel for the data being written.
STR_TI_DATA[0:7]	Input	**Transmit Ingress Data:** Data being written to the logical channel selected on STR_TI_LC. Data is accepted when a write request followed by an acknowledgment by requestor (SRC_RDY) and target (DST_RDY) is asserted.
STR_TI_BIF_AVAIL[0:3]	Output	**Transmit Ingress Buffer Interface Available**: When asserted, data is available in the given logical channel buffer. This signal is deasserted when the core is not enabled and triggers low in the event of the logical channel flush from the Host Interface.
STR_TI_SRC_RDY	Input	**Transmit Ingress Source Ready:** Write data on the STR_TI_DATA is available and valid.
STR_TI_DST_RDY	Output	**Transmit Ingress Destination Ready:** The target has accepted the write data for this logical channel (STR_TI_LC).
STR_TI_WCINT[0:3]	Output	**Transmit Ingress Interrupt:** Asserted for 1 clock when the transmit ingress logical channel buffer has crossed the word count as configured by the empty word count parameter (EWC). There is one signal for each transmit write logical channel.

Design Parameters

The XPS MOST NIC is configurable through a set of design parameters shown in Table 2, Table 3 and Table 4.

PLB Parameters

Table 2 lists the PLB specific parameters of XPS MOST NIC controller.

Table 2: **PLB Parameters**

Name	Values	Default	Description
C_FAMILY		spartan3, spartan3e, spartan3a, spartan3adsp, aspartan3, aspartan3e, aspartan3a, aspartan3adsp,spartan6, virtex4, qvirtex4, qvvirtex4, virtex5, virtex5fx, virtex6, virtex6cx	Target FPGA family
C_BASEADDR	'xxxxxxxxxxxxxxx0000000000000000' where x can be use controlled	0x00000000	Base Address of the XPS MOST NIC Controller
C_HIGHADDR	'xxxxxxxxxxxxxxxx1111111111111111' where x is identical to the value in C_BASEADDR	0x0000FFFF	High Address of the XPS MOST NIC Controller
C_SPLB_MID_WIDTH	log2(C_SPLB_NUM_MASTERS)	1	PLB master ID width
C_SPLB_NUM_MASTERS	1-16	1	Number of PLB Masters
C_SPLB_DWIDTH	32, 64, 128	32	Data width of XPS MOST NIC controller and PLB bus

Operating Modes

The core supports two configuration modes: Full master/slave, and slave with pseudo-master modes only (ring break diagnostic and loopback modes). The following table provides the description for the operating mode parameter.

Table 3: **Operating Mode Parameter**

Name	Values	Default	Description
C_OPMODE	0, 1	1	0 : Full Master/Slave mode 1: Slave-only mode

Word Count Triggers

The MOST NIC core supports 16 logical channel buffers for the receive and transmit directions, for a total of 32 logical channel buffers. In each direction, eight logical channels are dedicated for PLB, four for streaming ingress, and four for streaming egress. All buffers can assert interrupts to an external microprocessor to indicate that the buffer requires processing. The word count at which the interrupt is asserted is configurable, as defined in Table 4.

Streaming ports also assert a trigger to assist in processing for any logic directly connected to the port. Word counts can be chosen separately for ingress buffers and egress buffers. All ingress buffers use the same word counts, and all egress buffers use the same word counts. The available interrupts for each

group of logical channel are also illustrated in he same word counts. The available interrupts for each group of logical channel are also illustrated in Figure 6.

Table 4: **Word Count Parameters**

Name	Possible Values	Default	Description
C_FWC	8, 16	16	**Full Word Count**. The received word count at which an interrupt is generated.
C_EWC	16, 8	16	**Empty Word Count**. The available transmit word count at which an interrupt is generated.

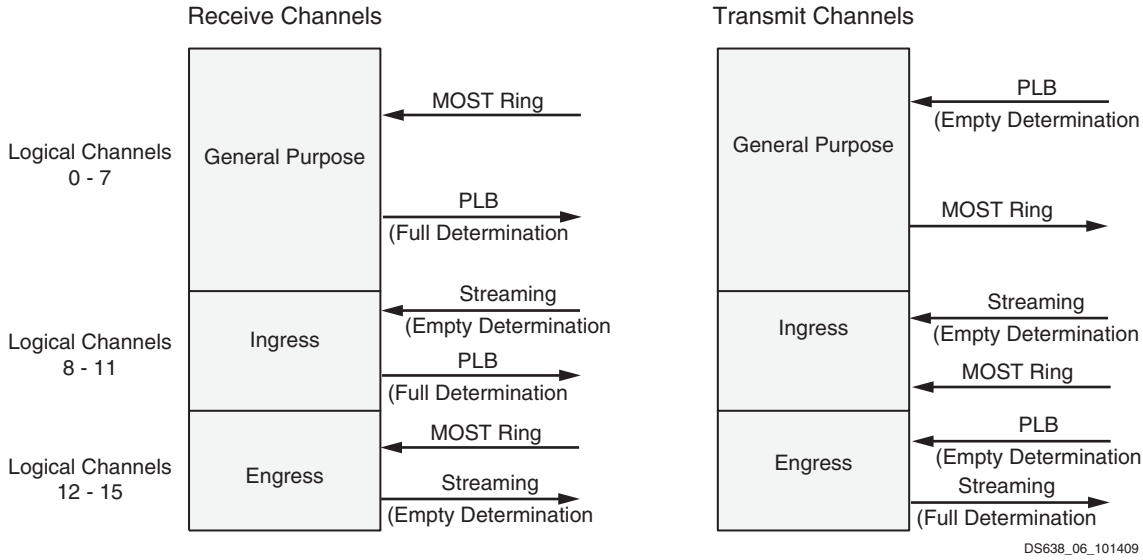

Figure 6: **Logical Channel Support**

Ingress / Egress

Ingress and egress are defined from the reference point of the core. An *ingress* transaction is synonymous with a *write* from the user interface to the core; an *egress* transaction is synonymous with a *read* to the user interface from the core. The following table summarizes the user interface access types (as illustrated in Figure 6 on page 19), and identifies them as either ingress or egress buffers.

Table 5: **Ingress / Egress Buffer Mapping**

Access	Buffer Type	Word Count
RX PLB Read	Egress	Full word count
RX Streaming Read	Egress	Full word count
RX Streaming Write	Ingress	Empty word count
TX PLB Write	Ingress	Empty word count
TX Streaming Read	Egress	Full word count
TX Streaming Write	Ingress	Empty word count

Full Word Count

An interrupt is generated every time a total of C_FWC words have been filled in the egress buffer. Setting the C_FWC to 8 generates an interrupt when eight words (32 bytes) have been written to the

buffer by the source. Another interrupt is generated when an additional eight words are written by the source, or an end-of- packet is received for asynchronous data. On initialization, all egress buffers are empty, and as a result, no interrupts are generated until the appropriate data is written.

Empty Word Count

An interrupt is generated every time a total of C_EWC have been removed in the ingress buffer. Setting the C_EWC to 8 generates an interrupt when there is room for at least eight words (32 bytes) to be written to the ingress buffer. When the interrupt is asserted, there is an assumption that the external processor will write eight words to the ingress buffer. In essence, eight locations of the buffer are now reserved and are no longer considered unused. When or if there are eight more unused word locations, an additional interrupt is generated.

On initialization, all ingress buffers are empty. The user application must prime the ingress buffer with data, and then enable the logical channel. Interrupts are then generated when there is C_EWC words free in the buffer. After the interrupt is cleared, an additional interrupt can be generated depending on the setting of C_EWC and the number of unused word locations remaining.

PLB Support

The MOST NIC is an PLB slave. The address of the MOST NIC can be configured using the XPS MOST NIC GUI.

Table 6: **PLB Base Address Parameter**

Name	Possible Values	Default Value	Description
C_BASEADDR	Valid address range: 0x00000000 - 0xffff0000, where bits 18-31 must be 0b	h'0000_0000	Base Address for the core
C_HIGHADDR	Valid address range: C_BASEADDR + 0x0000ffff	h'0000_ffff	High Address for the core

Parameter - Port Dependencies

N/A

Register Bit Ordering

All registers use big-endian bit ordering where bit-0 is MSB and bit-31 is LSB. The following table shows the bit ordering.

Table 7: **Register Bit Ordering**

0	1	2	. . . . .	29	30	31
MSB			. . . . .			LSB

Register Descriptions

The following table defines the MOST Controller registers, which are 32 bits wide and represented in big endian format, where bit 0 is the Most Significant Bit (MSB) on left, and bit 31 is the Least Significant Bit (LSB) on the right, that is, 0:31.

Reserved Bits

Reads to reserved bits or unused bits return zero(s). Writes to read-only registers are ignored. Reads from write-only registers return zero(s).

Byte Enable Support

Byte enable support is provided to the locations indicated in the following table. Locations that do not support byte enables ignore the PLB_BE[0:3] signals and provide 32 bit writes and reads. Locations that do support byte enables qualify writes with the assertion of PLB_BE[0:3]. Read access ignores the PLB_BE[0:3] signal and provides 32 bit read data.

Table 8: **Xilinx MOST Controller Register**

Register Name	Address	Size (Words)	Access	Byte Enable Support
Reserved for Future Use				
Reserved	C_BASEADDR + 0x0000 - C_BASEADDR + 0x00FC	64	Read only	N
MOST Controller Registers				
Version Register (VR)	C_BASEADDR + 0x0100	1	Read only	N
Soft Reset Register (SRR)	C_BASEADDR + 0x0104	1	Read/write	Y
Mode Select Register (MSR)	C_BASEADDR + 0x0108	1	Read/write	Y
Status Register (SR)	C_BASEADDR + 0x010C	1	Read only	N
Channel Control Register (CCR)	C_BASEADDR + 0x0110	1	Read/write	Y
Maximum and Current Position and Delay Register (MCPDR)	C_BASEADDR + 0x0114	1	Read only	N
Logical Address Register (LAR)	C_BASEADDR + 0x0118	1	Read/write	Y
Alternate Address Register (AAR)	C_BASEADDR + 0x011C	1	Read/write	Y
Group Address Register (GAR)	C_BASEADDR + 0x0120	1	Read/write	Y
Flush Control Register (FCR)	C_BASEADDR + 0x0124	1	Write only	Y
TX Buffer Error Register (TXBER)	C_BASEADDR + 0x0128	1	Read only	N
RX Buffer Error Register (RXBER)	C_BASEADDR + 0x012C	1	Read only	N
MOST Control Processing				
Message Retry Count and Delay Register (MRCDR)	C_BASEADDR + 0x0130	1	Read/write	Y
Control TX Status (CTXS)	C_BASEADDR + 0x0134	1	Read/write	Y
Control RX Status (CRXS)	C_BASEADDR + 0x0138	1	Read only	N
Control TX FIFO (CTXFIFO)	C_BASEADDR + 0x013C	1	Write only	N
Control TX Response FIFO (CTXRESFIFO)	C_BASEADDR + 0x0140	1	Read only	N
Control RX FIFO (CRXFIFO)	C_BASEADDR + 0x0144	1	Read only	N
Reserved	C_BASEADDR + 0x0148	1	Read only	N
Interrupt Status / Control				

Table 8: **Xilinx MOST Controller Register *(Cont'd)***

Register Name	Address	Size (Words)	Access	Byte Enable Support
MOST Interrupt Status Register (MISR)	C_BASEADDR + 0x0150	1	Read/write	Y
MOST Interrupt Pending Register (MIPR)	C_BASEADDR + 0x0154	1	Read only	N
MOST Interrupt Enable Register (MIER)	C_BASEADDR + 0x0158	1	Read/write	Y
MOST Interrupt Clear Register (MICR)	C_BASEADDR + 0x015C	1	Write only	Y
Buffer Interrupt Status Register (BISR)	C_BASEADDR + 0x0160	1	Read/write	Y
Buffer Interrupt Pending Register (BIPR)	C_BASEADDR + 0x0164	1	Read only	N
Buffer Interrupt Enable Register (BIER)	C_BASEADDR + 0x0168	1	Read/write	Y
Buffer Interrupt Clear Register (BICR)	C_BASEADDR + 0x016C	1	Write only	Y
Reserved	C_BASEADDR + 0x0170	36	Read only	N
Routing Table				
Common Routing Table (CRT)	C_BASEADDR + 0x0200	32	Read/Write	N
Logical Channel Enable (LCE)	C_BASEADDR + 0x0280	1	Read/Write	Y
Reserved	C_BASEADDR + 0x0284	7	Read Only	N
Slave Active Register 1 (SAR1)	C_BASEADDR + 0x02A0	1	Read Only	N
Slave Active Register 2 (SAR2)	C_BASEADDR + 0x02A4	1	Read Only	N
Reserved	C_BASEADDR + 0x02A8	6	Read Only	N
Master Allocation Table (MAT)	C_BASEADDR + 0x02C0	16	Read Only	N
Streaming Port MMR				
Streaming Port MMR (SMMR)	C_BASEADDR + 0x0300	64	Read/Write	Y
MOST PLL Control Registers				
PLL Reference Count Register (PRCR)	C_BASEADDR + 0x0400	1	Read/Write	Y
External PLL Reset Count (EPRC)	C_BASEADDR + 0x0404	1	Read Only	N
Reserved				
Reserved	C_BASEADDR + 0x0408	766	Read Only	N
Transmit/Receive Buffer				
Transmit Buffer (TXBUFF)	C_BASEADDR + 0x1000	1024	Write Only	N
Receive Buffer (RXBUFF)	C_BASEADDR + 0x2000	1024	Read Only	N
Reserved				
Reserved	C_BASEADDR + 0x3000 - C_BASEADDR + 0x3FFC	1024	Read Only	N

MOST Controller Registers

Version Register

The Version Register (VR) (Table 9) indicates capability and version information about the core.

Table 9: **Version Register**

Bit(s)	Name	Access	Default Value	Description
0 - 15	RSVD	Read Only	0	**Reserved:** Reads return zero and writes have no effect.
16-23	CAP	Read Only	0x01	**Capability:** The capabilities of the core are included in binary-encoded format. The capability for this version reads 0x01.
24 -31	VER	Read Only	0x11	**Version:** The version of the core is included in binary encoded format. The version for this release reads 0x11.

Soft Reset Register

The Soft Reset Register (SRR) is used to place the MOST controller in reset. Table 10 on page 24 describes the SRR bits.

- Writing a 1 to SRST bit of the SRR places the MOST controller in reset. All registers (with the exception of the SRR) are reset. The MOST controller can and will be held in reset indefinitely as long as SRST is a 1'
- Writes of 1 to SRST take priority over writes of MENA. For example, a write of 11b to the SRR forces SRST to 1 and MENA to 0
- When SRST is asserted, the MOST core is initialized to bypass mode

Many configuration bits can be changed only when the MENA bit is negated. When MENA is negated, the core defaults to bypass mode as a slave. To enable operation, a logic 0 must be written to the SRST bit and a logic 1 to the MENA bit.

The typical programming sequence is:

- Program SRST to 1. This resets all registers to their default state, including the MENA bit. The controller is in slave, bypass mode
- Program SRST and MENA to 0. This takes the core out of reset, but allows the user to program control registers
- Configure PRCR based on the equation:

  `SAMPLE_POINT X SPLB_Clk (MHz) / MOST_PLL_CLK (MHz),`

  where `SAMPLE_POINT` = 0xA4 (constant)

  Refer to "PLL Reference Count Register" on page 48 for PRCR programming.
- Program control registers to the appropriate values
- Program SRST to 0, and MENA to 1. The MOST controller can now participate in active bus communications in the selected mode. Many control registers cannot be written when MENA is 1. Any writes to these registers will have no affect

MENA can be deasserted at any point, but this also forces the MOST controller into a bypass mode. In addition all buffers are flushed, and all logical channels disabled.

Table 10: **Soft Reset Register**

Bit(s)	Name	Access	Default Value	Description
0 - 29	RSVD	Read/Write	0	**Reserved:** Reads return zero and writes have no effect.
30	MENA	Read/Write	0	**MOST Enable:** The MOST controller is enabled when this bit is asserted. When negated operational and configuration modes may be set. The MOST controller will be operational when MENA is 1. When set to 0, the MOST controller is forced to the bypass mode as a slave and the controller is disabled This bit is reset to 0 on assertion of SRST.
31	SRST	Read/Write	0	**Soft Reset:** The MOST controller is reset if this bit is set to 1. When asserted, all MOST controller registers, (except the SRST) are reset. When negated, operation of the MOST controller resumes - subject to the status of MENA

Mode Select Register

The Mode Select Register (MSR) configures the operating mode of the controller. This register can be read back to determine that it has been programmed correctly. However, to determine the actual state of the core, the Status Register (SR) must be read. Table 11 on page 25 describes the MSR bits.

- The MNSLV, RBDT, BYPASS, and LBACK bits can only be written when the MENA bit in the SRR is negated. When MENA bit is asserted, the MSR register is sampled, and the appropriate mode is enabled
- Master/Slave operation is configured via the MNSLV bit
- The RBDT, BYPASS and LBACK bits are mutually exclusive. Only one of these bits may be set at a time. Normal mode is enabled when RBDT, BYPASS, and LBACK are all 0. If more than one bit is set, BYPASS takes priority, and then LBACK
- A node configured as a master node (`MNSLV = 1`) cannot be put into BYPASS mode. To place the core into BYPASS, the core will need to configured to be a slave node (`MNSLV = 0`)
- When MENA is changed from a 1 to a 0 the RBDT, BYPASS, and LBACK bits are initialized to the default states (BYPASS mode)
- In Normal mode, the MOST controller participates in normal network operation

The default state of the MOST controller is BYPASS mode as a slave.

Table 11: **Mode Select Register**

Bit(s)	Name	Access	Default Value	Description
0 - 17	RSVD	Read Only	0	**Reserved:** Reads return zero and writes have no effect.
18	EBYPASS	Read/Write	1	**External bypass:** The value in this bit is forwarded to the MOST_EXT_BYPASS pin in order to drive a bypass state to external circuitry, such as a PLL. This bit is not related to the BYPASS mode of the core (bit 30 of this register).
19	TCTL	Read/Write	0	**Transmit control synchronization override:** Allows transmission of normal control messages for a master when the ring is not stable. 1 = enables transmission of normal messages even when node is not block synchronized 0 = regular operation This bit can be written only when MENA in the SRR is negated. This bit can be asserted only when MNSLV is also asserted.
20	ENRST	Read/Write	1	**External not-reset:** The value in this bit is forwarded to the MOST_EXT_NRESET pin in order to drive a reset to external circuitry, such as a PLL.
21- 24	RSVD	Read Only	0	**Reserved:** Reads return zero and writes have no effect.
25	DAZ	Read/Write	0	**Drive all Zeros:** Forces the transmit pin to be 1'b0. 1 = drive all Zero's on the tx pin. 0 = drive normal data
26	STRON	Read/Write	0	**Stream On:** Indicates if there is an external module attached to the streaming port 1 = external module is attached to the streaming port. 0 = external module is not attached to the streaming port If an external module is not attached to the streaming port, then all MMR access to the streaming port will be terminated by the MOST controller directly. This bit can be written only when MENA in the SRR is negated
27	MNSLV	Read/Write	0	**Master - Not Slave:** This bit selects the master or slave functionality. Writes to this bit position have no effect if the operating mode is Slave Only unless RBDT is asserted. 1 = set the controller to Master mode 0 = set the controller to Slave mode This bit can be written only when MENA in the SRR is negated
28	RBDT	Read/Write	0	**Ring Break Diagnosis Test Mode Selection:** RBD test mode select bit. 1 = enables the core for an RBD test mode. 0 = does not enable the core for an RBD test mode This bit can be written only when MENA in the SRR is negated.When MNSLV is also asserted in a slave-only core, this will force the core into a pseudo-master mode, which will provide valid traffic on the MOST_TX line. For all other programming, this bit will have no effect on the operation of the core.

Table 11: **Mode Select Register *(Cont'd)***

Bit(s)	Name	Access	Default Value	Description
29	LBACK	Read/Write	0	**Loopback Selection:** The Loopback mode select bit. 1 = enable Loopback mode 0 = disable Loopback mode This bit can be written only when MENA in the SRR is negated
30	BYPASS	Read/Write	1	**Bypass mode Selection:** The Bypass mode select bit. 1 = enable Bypass Mode 0 = disable Bypass Mode. This bit can be written only when MENA in the SRR is negated and the core is configured as a Slave (MNSLV = 0) Bypass mode is also forced active when MENA is reset to0
31	RSVD	Read Only	0	**Reserved:** Reads return zero and writes have no effect.

Transmit Control Synchronization Override

Control messages can be sent or received only when the node is at least Block synchronized. For a Master node, this does not happen until all nodes on the rings are also synchronized. However, a Master node initiates the preamble generation for the ring. If this mode is set, the synchronization state of the present Master node will be ignored, and the node may transmit the request portion of a normal control message.

Note: Even with this setting active, all transmissions are guaranteed to fail because an acknowledgement for the message will never be received.

Ring Break Diagnosis Test Mode

The Ring Break Diagnosis (RBD) mode allows the user diagnose a fatal error in the network which prevents the propagation of light through the network. Four conditions cause a fatal error:

1. The device has an insufficient or no distribution voltage
2. An optical receiver is defective
3. An optical transmitter is defective
4. The optical connection between the transmitter and the receiver is interrupted

Bypass Mode

In Bypass mode, the signal received at the input (`MOST_RX`) port is forwarded to the output (`MOST_TX`) port. As long as the node is in BYPASS mode, it is transparent to the network (that is, the position count is not incremented) and this node does not transmit any frames. However, the node receives *any* frame, stores the position and delay value, and performs parity checking. Bypass mode is also set when MENA=0.

Loopback Mode

In Loopback mode, the MOST controller transmits a frame onto the MOST bus. *Any* transmitted frame is looped back internally to the core to the RX line and is acknowledged if it is a control message. It does not participate in normal bus communication and does not receive any frames transmitted by other MOST nodes.

Normal Mode

In Normal mode, the MOST controller participates in bus communication by transmitting and receiving frames. The Drive All Zeros (DAZ) control bit overrides the output of the TX pin in any operational mode. When this bit is set, the TX output is forced to '0'.

Status Register (SR)

The following table shows the Status Register which provides the current status of the MOST controller.

Table 12: **Status Register**

Bit(s)	Name	Access	Default Value	Description
0 - 17	RSVD	Read Only	0	**Reserved:** Reads return zero and writes have no effect.
18	EBYPASS	Read Only	1	**External bypass:** The current value of the MOST_EXT_BYPASS pin.
19	TCTL	Read Only	0	**Transmit control synchronization override:** Indicates that transmission of normal control messages enabled for any synchronization state 1 = transmission of normal messages is enabled even when node is not block synchronized 0 = regular operation
20	ENRST	Read Only	1	**External not-reset:** The current value of the MOST_EXT_NRESET pin.
21	DAZ	Read Only	0	**Drive all zeros**: Indicates the DAZ bit is set 1 = DAZ is set, and as such the TX output has been forced to '0'. 0 = The DAZ bit is not set
22	SBLCK	Read Only	0	**Super Block Lock**: Indicates synchronization to the super block 1 = controller is phase and frequency locked to the bit stream and synchronized to the super block 0 = controller is not synchronized to super block
23	BLCK	Read Only	0	**Block Lock:** Indicates synchronization to the block 1 = controller is phase and frequency locked to the bit stream and synchronized to the block 0 = the controller is not synchronized to block
24	FLCK	Read Only	0	**Frame Lock:** Indicates synchronization to the frame 1 = controller is phase and frequency locked to the bit stream and synchronized to the frame 0 = controller is not phase and frequency locked to the bit stream and not synchronized to the frame
25	FOS	Read Only	0	**Fibre Optic Status:** Indicates the signal being received on the MOST_FOR_STATUS pin.
26	STRON	Read Only	0	**Stream On:** Indicates if there is an external module attached to the streaming port 1 = external module is attached to the streaming port. 0 = external module is not attached to the streaming port

Table 12: **Status Register *(Cont'd)***

Bit(s)	Name	Access	Default Value	Description
27	MNSLV	Read Only	1	**Master/Not slave:** Indicates that the MOST controller is master or slave 1 = MOST controller is configured as a master 0 = MOST controller is configured as a slave
28	RBDT	Read Only	0	**Ring Break Diagnosis Test:** Indicates that the MOST controller is in RBD test mode 1 = in RBD test mode 0 = not in RBD test mode
29	LBACK	Read Only	0	**Loopback:** Indicates that the MOST controller is in Loopback mode 1 = in Loopback mode 0 = not in Loopback mode
30	BYPASS	Read Only	1	**Bypass:** Indicates that the MOST controller is in Bypass mode 1 = in Bypass mode 0 = not in Bypass mode
31	RSVD	Read Only	0	**Reserved:** Reads return zero and writes have no effect.

Channel Control Register

The following table shows the Channel Control Register (CCR) which configures information for the synchronous, asynchronous, and control channels.

- The SBD bits control the portion of the MOST frame dedicated to synchronous and asynchronous data. This value determines the boundary between the synchronous and asynchronous data area in the frame
- For example, a value of 10 indicates that there are 10 quadlets (10*4 bytes) of synchronous data and 5 quadlets (5*4 bytes) of asynchronous data in the frame
- The SBD value may change at any time during operation. However, after a change of the synchronous boundary, the synchronous channel resource mechanism becomes invalid and has to be reset by the reception of a *deallocate all* message
- The minimum legal value for SBD is 1 (0x1). Writes of 0 are treated as a write of 1
- The SBD bits can be written only when this node is programmed as a master. The value read from the register will always be the active SBD state in the ring. If this node is acting as a master, this will be the value previously programmed. But when the core is not enabled, the value read will always be the default value. If this node is acting as a slave, it will be the SBD received from the ring

Table 13: **Channel Control Register**

Bit(s)	Name	Access	Default Value	Description
0 - 27	RSVD	Read/Write	0	**Reserved:** Reads to return zero and writes have no effect.
28 - 31	SBD	Read/Write	6	**Synchronous Boundary Descriptor:** Reads indicates the boundary between the synchronous and asynchronous data areas in the frame. The SBD bits can be written only when this node is configured as a master as set by MNSLV in MSR. When the core is enabled as a Master, this will automatically take effect

Maximum and Current Position and Delay Register

The following table shows the Maximum and Current Position and Delay Register (MCPDR) which provides the maximum number of devices (position) and the maximum delay value of the MOST network. For a slave node, this maximum position and delay are derived from the Network Information Channel on the MOST ring. As a result, this register takes some time to stabilize at initialization and after every unlock event. It also provides the current position and delay value of the device.

Table 14: **Maximum and Current Position and Delay Register**

Bit(s)	Name	Access	Default Value	Description
0	RSVD	Read Only	0	**Reserved:** Reads return zero and writes have no effect.
1 - 7	MAXDELAY	Read Only	0	**Maximum Delay Value:** Indicates the total number of node/frame delays for synchronous data within the MOST network.
8	RSVD	Read Only	0	**Reserved:** Reads return zero and writes have no effect.
9- 15	MAXPSTN	Read Only	0	**Maximum Position Value:** Indicates the total number of active nodes within the MOST network.
16	RSVD	Read Only	0	**Reserved:** Reads return zero and writes have no effect.
17 - 23	NODEDELAY	Read Only	0	**Node Delay Value:** Indicates the number of node delay from the timing master downstream to this node, for synchronous data. This value will be 0 in case of timing master.
24	RSVD	Read Only	0	**Reserved:** Reads return zero and writes have no effect.
25 - 31	NODEPSTN	Read Only	0	**Node Position:** Indicates the physical position of the node in the MOST network. All the active slave nodes will increment this value by 1. This value will be 0 in case of timing master.

Logical Address Register

The following table shows the Logical Address Register (LAR) which indicates the logical address of this node, which is used by all nodes to address a single node, and is unique even if there are multiple devices of the same type. The logical address is always used as source address in the control message header, but can also be used as destination address in the asynchronous packet header. Valid ranges are

0x0001-0x02FF and 0x0500-0xFFFF. No checking of a valid LAR is performed in hardware. Users must ensure this register is programmed with a valid value.

Table 15: **Logical Address Register Description**

Bit(s)	Name	Access	Default Value	Description
0 - 15	RSVD	Read/Write	0	**Reserved:** Reads return zero and writes have no effect.
16 - 31	LAR	Read/Write	0x0FFF	**Logical Address:** Configures the logical address of the node in the MOST ring

Alternate Address Register

The following table shows the Alternate Address Register (AAR) which indicates the alternate address of this node, and is used by all nodes to address a single node. This address is unique, even if there are multiple devices of same type. The alternate address is set by the user and can be given any value in the range 0x0000–0xFFFF. The alternate address can be used as both source address and destination address in the asynchronous packet header fields.

Table 16: **Alternate Address Register**

Bit(s)	Name	Access	Default Value	Description
0 - 15	RSVD	Read/Write	0	**Reserved:** Reads return zero and writes have no effect.
16 - 31	AAR	Read/Write	0x0000	**Alternate Address:** Configures the alternate address of the node in the MOST ring.

Group Address Register

The following table shows the Group Address Register (GAR) which indicates the group address of this node and provides access to a group of devices. The group address can be set by the application and is typically used to control several devices of the same type (for example, active speakers).

The grouping of devices must be established during definition and initialization of the system. Groups can be dynamically built by modifying the group address. The group address can be any value in the range 0x0300–0x03C7 or 0x03C9–0x03FF. The default address (0x03C8) is reserved for multicast; the user should change the GAR from this default value.

Table 17: **Group Address Register**

Bit(s)	Name	Access	Default Value	Description
0 - 15	RSVD	Read/Write	0	**Reserved:** Reads return zero and writes have no effect.
16 - 23	GAR (high)	Read Only	0x03	**Group Address High Byte:** Configures the group address of the node in the MOST ring- hard coded to group address range of 0x03
24 - 31	GAR (low)	Read/Write	0xC8	**Group Address Low Byte:** Configures the group address of the node in the MOST ring, and can be set to any value.

Note: The high byte of GAR may not be modified by the user because all group addresses must start with 0x03.

Flush Control Register

The following table shows the Flush Control Register (FCR) which allows the synchronous, asynchronous, and control buffers to be flushed.

Table 18: **Flush Control Register Description**

Bit(s)	Name	Access	Default Value	Description
0 - 23	RSVD	Write Only	0	**Reserved:** Reads return zero and writes have no effect.
24-27	FCHN	Write Only	0	**Flush Channel:** Configures the logical channel to be flushed.
28	FDIR	Write Only	0	**Flush Direction**. 1 = Flush the transmit buffer 0 = Flush the receive buffer
29	RSVD	Write Only	0	**Reserved:** Reads return zero and writes have no effect.
30	FLCB	Write Only	0	**Flush Logical Channel Buffer:** Enables a logical channel buffer to be flushed. 1 = Flush the logical channel buffer as indicated by FDIR and FCHN. 0 = Do not flush a logical channel buffer
31	FCTB	Write Only	0	**Flush Control Buffer:** Enables a control buffer to be flushed. 1 = Flush the control buffer as indicated by FDIR 0 = Do not flush a control buffer This bit needs to be written only once with a 1 to flush the appropriate buffer. The bit will automatically return to 0

Transmit Buffer Error Register

The Transmit Buffer Error Register (TXBER) indicates the transmit logical channel buffers with underflow or overflow errors. Table 19 and Table 20 on page 32 summarize and describe the register bits.

- TXBER bits are cleared when the appropriate buffer has been flushed via the FCR. The bits are set on subsequent underflow or overflow conditions
- When an error condition sets any of the TXBER bits, the TXBUFFER bit is set in the MOST interrupt status register, if enabled

Table 19: **Transmit Buffer Error Register Bits**

0	1	2	3	4	5	6	7
TXOLC15	TXOLC14	TXOLC13	TXOLC12	TXOLC11	TXOLC10	TXOLC9	TXOLC8
8	**9**	**10**	**11**	**12**	**13**	**14**	**15**
TXOLC7	TXOLC6	TXOLC5	TXOLC4	TXOLC3	TXOLC2	TXOLC1	TXOLC0
16	**17**	**18**	**19**	**20**	**21**	**22**	**23**
TXULC15	TXULC14	TXULC13	TXULC12	TXULC11	TXULC10	TXULC9	TXULC8
24	**25**	**26**	**27**	**28**	**29**	**30**	**31**
TXULC7	TXULC6	TXULC5	TXULC4	TXULC3	TXULC2	TXULC1	TXULC0

Table 20: **Transmit Buffer Error Register**

Bit(s)	Name	Access	Default Value	Description
0 - 15	TXOLC_*N*_	Read Only	0	**Transmit Overflow Logical Channel:** Indicates a logic channel has overflowed. 1 = The indicated *(_N_)* logical channel has overflowed 0 = The indicated *(_N_)* logical channel has not overflowed.
16 - 31	TXULC_*N*_	Read Only	0	**Transmit Underflow Logical Channel:** Indicates a logic channel has underflowed. 1 = The indicated *(_N_)* logical channel has underflowed 0 = The indicated *(_N_)* logical channel has not underflowed.

Receive Buffer Error Register

The Receive Buffer Error Register (RXBER) indicates which receive logical channel buffers have underflow or overflow errors. Table 21 and Table 22 on page 32 summarize and describe the register bits.

- RXBER bits are cleared when the appropriate buffer has been flushed (via the FCR). The bits are set on subsequent underflow or overflow conditions
- When an error condition sets any of the RXBER bits, the RXBUFERR bit is set in the MOST interrupt status register, if enabled

Table 21: **Receive Buffer Error Register Bits**

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
RXOLC15	RXOLC14	RXOLC13	RXOLC12	RXOLC11	RXOLC10	RXOLC9	RXOLC8
8	**9**	**10**	**11**	**12**	**13**	**14**	**15**
RXOLC7	RXOLC6	RXOLC5	RXOLC4	RXOLC3	RXOLC2	RXOLC1	RXOLC0
16	**17**	**18**	**19**	**20**	**21**	**22**	**23**
RXULC15	RXULC14	RXULC13	RXULC12	RXULC11	RXULC10	RXULC9	RXULC8
24	**25**	**26**	**27**	**28**	**29**	**30**	**31**
RXULC7	RXULC6	RXULC5	RXULC4	RXULC3	RXULC2	RXULC1	RXULC0

Table 22: **Receive Buffer Error Register**

Bit(s)	Name	Access	Default Value	Description
0 - 15	RXOLC_*N*_	Read Only	0	**Receive Overflow Logical Channel:** Indicates a logic channel has overflowed. 1 = The indicated *(_N_)* logical channel has overflowed 0 = The indicated *(_N_)* logical channel has not overflowed.
16 - 31	RXULC_*N*_	Read Only	0	**Receive Underflow Logical Channel:** Indicates a logic channel has underflowed. 1 = The indicated *(_N_)* logical channel has underflowed 0 = The indicated *(_N_)* logical channel has not underflowed.

MOST Control Processing

Message Retry Count and Delay Register

The following tables shows the Message Retry Count and Delay Register (MRCDR) which contains the retry count and delay value for the re-transmission of the control messages. The count value configures how many times the re-transmission should take place and the delay value configures the gap between the re-transmission. This register should not be changed during the transmission of a control message, as the changes may not propagate to the active message.

Table 23: **Message Retry Count and Delay Register Description**

Bit(s)	Name	Access	Default Value	Description
0 - 15	RSVD	Read/Write	0	**Reserved**: Reads return zero and writes have no effect.
16 - 23	DLYVALUE	Read/Write	1	**Delay Value**: Configures the gap between the re-transmission of control messages. The time unit is always measured as blocks within the control channel. Valid values are 0x00 to 0xFF.
24 - 31	RTRYCNT	Read/Write	1	**Retry Count**: Configures the number of re-transmission. Valid values are 0x00 to 0xFF.

Control Transmit Status Register

The following table shows the Control Transmit Status (CTXS) register which contains status and control for the transmission of control messages.

Table 24: **Control Transmit Status Register Bit Description**

Bit(s)	Name	Access	Default Value	Description
0 - 29	RSVD	Read/Write	0	**Reserved:** Reads return zero and writes have no effect.
30	CTXOCC	Read	0	**Transmit Occupancy:** Indicates the number of control messages currently queued for transmission in the CTXFIFO. The Control module can also queue 1 message in progress. Users should not add any control message if this register indicates '1.'
31	CNEWTX	Write	0	**New Transmit Message:** Write a '1' to indicate that a new transmit message has been added to the transmit control message buffer. Reads always return'0.'

Control Receive Status Register

The following tables shows the Control Receive Status (CRXS) register which contains status and control for the transmission of control messages.

Table 25: **Control Receive Status Register Bit Description**

Bit(s)	Name	Access	Default Value	Description
0 - 29	RSVD	Read Only	0	**Reserved:** Reads return zero and writes have no effect.
30	CRXOCC	Read Only	0	**Receive Occupancy:** Indicates if there is any control message currently queued for reception. Users should not try to read any control message if this register indicates 0. Users should not try to read any control message if this register indicates '0'
31	CNEWRX	Read Only	0	**New Receive Message:** Indicate that a new receive message was received since the last read to this register. This bit is cleared on reads. It is set when a receive message is successfully received from the MOST ring and added to the CRXFIFO

Control Transmit FIFO

The Control Transmit FIFO (CTXFIFO) register (Table 26) is a write buffer used to add control messages for transmission.

Table 26: **Control Transmit FIFO Bit Description**

Bit(s)	Name	Access	Default Value	Description
0 - 31	TXMSG	Write Only	Not initialized	**Transmit Message:** Transmit messages are added to the control transmit message FIFO by writing to this registers

Control messages can be two to six words in length. The control transmit message FIFO contains sufficient storage to hold one complete message. The control module within the core will also queue a message, for a total storage capacity for two messages. The FIFO is accessed by writing words sequentially to the CTXFIFO, where all reserved bits must be written with zeroes. The order in which words of the message must be added are as follows.

Control Transmit Message Structure

The first write contains the destination and source addresses. Destination Address (DSTA) is the destination of the control message. Source Address (SRCA) is the source of the message and must be set to the current logical address of the core.

Table 27: **Control TX MSG: First Word**

0 - 15	16-31
DSTA	SRCA

The second write contains the message priority, type, and the first three bytes of the message. The bytes are added to the Control Transmit FIFO in network order.

The Message Type (MSGT) is a 3-bit binary encoded message type. The user can write any type of control message to the CTXFIFO. Bytes 0-N are written with control fields or data as necessary. The

remaining fields in the second word contain the first three bytes of the message, as indicated in the following table.

Table 28: **Control TX MSG: Second Word**

0	1 - 4	5 - 7	8 - 15	16- 23	24 - 31
RSVD	PRIOR	MSGT	Byte0	Byte1	Byte2

The third to sixth write contains the remaining bytes of the message. Only as many bytes as are required need to be written to the CTXFIFO.

Table 29: **Control TX MSG: Third through Sixth Word**

0 - 7	8 - 15	16- 23	24 - 31
ByteN	ByteN+1	ByteN+2	ByteN+3

Control Message Encoding and Response

Any type of control message can be sent from user software using CTXFIFO. Responses to transmitted messages are returned in the CTXRESFIFO. Any messages which receive a NAK in the transmission field, or are otherwise errored, are automatically retried by hardware and subject to the retry values in the MRCDR registers.

The CTLTXFAIL interrupt is asserted if the counters expire and the message was not successfully transmitted. The following table describes the message type encoding and response of both the CTLTKOK interrupt and the returned receive message.

Table 30: **Message Type Encoding and Actions**

MSGT	Message Type	CTLTXOK Assertion	CTLTXFAIL Assertion	Response
3'b000	Normal	On successful ACK received from the destination.	ACK not received and message retried as dictated by MRCDR	The original message and transmission status are returned in the CTXRESFIFO
3'b001	Remote Read Request	On successful reception of the read response	Response not received and message retried as dictated by MRCDR	The remote read response is returned as a received control message in the CTXRESFIFO. The message in the CTXRESFIFO includes the original request, response, and transmissions status.
3'b010	Remote Write Request	On successful ACK received from the destination	ACK not received and message retried as dictated by MRCDR	The original message and transmission status are returned in the CTXRESFIFO
3'b011	Allocation Request	On the successful reception of an allocation response, and successful transmission of the ACK from this node.	Response not received and message retried as dictated by MRCDR	The allocation response is returned as a received control message in the CTXRESFIFO. The message in the CTXRESFIFO will include the original request, response as well as the transmission status.

Table 30: **Message Type Encoding and Actions *(Cont'd)***

MSGT	Message Type	CTLTXOK Assertion	CTLTXFAIL Assertion	Response
3'b100	Deallocation Request	On the successful reception of a deallocation response, and successful transmission of the ACK from this node.	Response not received and message retried as dictated by MRCDR	The deallocation response is returned as a received control message in the CTXRESFIFO. The message in the CTXRESFIFO will include the original request, the response as well as the transmission status.
3'b101	Get Source Request	On successful ACK on ring.	ACK not received and message retried as dictated by MRCDR	The Get Source Response is returned as a received control message in the CTXRESFIFO. The message in the CTXRESFIFO will include the original request, response and transmission status
3'b110, 3'b111	Reserved - do not use	n/a	n/a	n/a

Note: Non-legal encodings are not checked by hardware, likely resulting in a corrupted MOST ring.

Control Transmit Response FIFO

The following table shows the Control Transmit Response FIFO (CTXRESFIFO) register which is a read buffer used to read transmit message responses and status.

Table 31: **Control Transmit Response FIFO Bit Description**

Bit(s)	Name	Access	Default Value	Description
0 - 31	CTXRES	Read Only	Not initialized	**Transmit Message Response:** The original transmit message, response, and status is read via this register.

Control messages can be from two to six words in length. A seventh word is used to return the transmission status. The control transmit response FIFO contains sufficient storage to hold one complete message, response, and status. The buffer is accessed by reading words sequentially from the CTXRESFIFO.

This FIFO contains response messages received from destination nodes as a result of a transmitted control message from this node. The received message includes the original request, the response containing both the message answer as well as checksum, and transmission status indicating transmission success or failure. Responses are received for allocation, deallocation, remote read, and get source control messages.

All seven words must be read for each message, even if the message contains fewer than seven words, and the seventh word always contains the transmission status. The order in which words of the message must be read is defined in Table 32 through Table 35.

Control Transmit Response Structure

The first read contains the destination and source addresses. Destination Address (DSTA) is the destination of the control message. The Source Address (SRCA) is the source address of the message,

which will be the current LAR of the core if the message was composed properly for the Control Transmit FIFO.

Table 32: **Control TX Response MSG: First Word**

0 - 15	16-31
DSTA	SRCA

The second read contains the message type and the first three bytes of the message. The remaining fields in the second word contain the first three bytes of the message, as indicated in the table.

Table 33: **Control TX Response MSG: Second Word**

0 - 5	5 - 7	8 - 15	16- 23	24 - 31
RSVD	MSGT	Byte0	Byte1	Byte2

The third to sixth read contain the remaining bytes of the message. All six words must be read.

Table 34: **Control TX Response MSG: Third to Sixth Word**

0 - 7	8 - 15	16- 23	24 - 31
ByteN	ByteN+1	ByteN+2	ByteN+3

The seventh word always contains the transmission status. Transmit messages are tried until successful, as programmed in the MRCDR register. However, when the MRCDR counter expires, and a failed message with the failed transmission status for the last message attempt is returned as indicated by a value not equal to 0x1010 for the TSTAT field, the response portion of the message may not be completely readable due to the error condition causing failure.

Table 35: **Control TX Response MSG: Seventh Word**

0 -15	16- 31
RSVD	TSTAT

Control Receive FIFO

The Control Receive FIFO (CRXFIFO) register is a read buffer used to read received normal control messages. Control messages are six words in length. The control receive message FIFO contains sufficient storage to hold two complete messages, including the received checksum. The buffer is accessed by reading words sequentially from the CRXFIFO. Received control messages are control messages sourced from other nodes targeted to this node. All six words must be read for each message. The order in which words of the message must be read is described in Table 36 through Table 39.

Table 36: **Control Receive FIFO Bit Description**

Bit	Name	Access	Default Value	Description
0 - 31	RXMSG	Read Only	Not initialized	**Receive Message:** Receive messages are read from the control receive message queue by reading from this register

Control Receive Message Structure

The first read contains the destination and source addresses. Destination Address (DSTA) is the destination of the control message–either Broadcast or the current Groupcast Address, Logical

Address, or Physical Address of this node. The Source Address (SRCA) is the address of the source of the message

Table 37: **Control RX MSG: First Word**

0 - 15	16-31
DSTA	SRCA

.The second read contains the message type and the first three bytes of the message. The Message Type (MSGT) is a 3-bit binary encoded message type. Table 30 describes the binary encoding of the messages. The remaining fields in the second word contain the first three bytes of the message as indicated in Table 38.

Table 38: **Control RX MSG: Second Word**

0 - 5	5 - 7	8 - 15	16- 23	24 - 31
RSVD	MSGT	Byte0	Byte1	Byte2

The third to sixth read contain the remaining bytes of the message. All six words must be read. Only Normal control messages will be passed along to the application level. All others are automatically processed by the hardware. Allocation, Deallocation, and Get Source messages are all fully supported. Remote Read messages received from other nodes are acknowledged, but zeroes are returned. Remote Write messages received from other nodes are acknowledged with *a message type not supported* response because any data content is ignored.

Table 39: **Control RX Response MSG: Third to Sixth Word**

0 - 7	8 - 15	16- 23	24 - 31
ByteN	ByteN+1	ByteN+2	ByteN+3

Interrupt Control and Status Registers

The MOST controller has two interrupt signals: `MOST_irpt` is asserted based on the MOST Interrupt status, and `BUFFER_irpt` is asserted based on the Buffer interrupt status.

MOST Interrupt Control and Status Registers

MOST interrupts are controlled using four registers:

- **Status (MISR):** Indicates the interrupt status bits. These bits are set and cleared regardless of the status of the corresponding bit in the Interrupt Enable Register (`MIER`)
- **Pending (MIPR)**: Indicates the interrupts that are actually indicated to software. In effect, bit wise it is the MISR register ANDed with the MIER register
- **Enable (MIER)**: Determines if an interrupt is generated or not
- **Clear (MICR)**: Clears the status bits in the MISR and MIPR register when a 1 is written to the corresponding bit in the Interrupt Clear Register (MICR)

Triggering MOST Interrupts

Two conditions cause the MOST Interrupt signal to assert:

- If a bit in the MISR is 1 and the corresponding bit in the MIER is 1
- Changing an MIER bit from 0 to 1 when the corresponding bit in the MISR is already 1

An interrupt is generated and the Pending Register bit (MIPR) is set if the corresponding mask bit in the Enable Register (MIER) is set.

Clearing MOST Interrupts

Two conditions cause the MOST Interrupt signal to deassert:

- Clearing a bit with a value of 1 in the MISR/MIPR (by writing a 1 to the corresponding bit in the MICR)
- Changing an MIER bit from 1 to 0 when the corresponding bit in the MISR is 1

When deassertion and assertion conditions occur simultaneously, the MOST line is deasserted first, and then reasserted if the assertion condition remains true.

Common Bit Map

Because the Status, Pending, Enable, and Clear registers have common mapping between registers, only a single mapping is depicted. Each has a unique register and address map, as indicated in Table 8 on page 21.

Each bit has a suffix:

- **Status**: INT
- **Pending**: PEN
- **Enable**: ENA
- **Clear**: CLR

Table 40: **MOST Interrupt Register Bit Positions**

0	1	2	3	4	5	6	7
RSVD	RSVD	RSVD	RSVD	RSVD	STRINT	RXBUFERR	TXBUFERR
8	**9**	**10**	**11**	**12**	**13**	**14**	**15**
RXASYNEOP	TXASYNEOP	CTLRXNAK	CTLRXNEW	CTLTXFREE	CTLTXFAIL	CTLTXOK	CTLARBLST
16	**17**	**18**	**19**	**20**	**21**	**22**	**23**
ARXCRCERR	RSVD	PERR	BPCV	MDLYCHG	MPOSCHG	DLYCHG	POSCHG
24	**25**	**26**	**27**	**28**	**29**	**30**	**31**
RSVD	LONCHG	SBDCHG	SYNCCHG	RSVD	RWP	RSVD	RSVD

Table 41: **MOST Interrupt Register**

Bit(s)	Name	Description
0 - 4	RSVD	**Reserved:** Reads return zero and writes have no effect.
5	STRINT	**Streaming Interrupt:** Indicates assertion of the streaming port interrupt signal strm_irpt
6	RXBUFERR	**RX Buffer Error:** Indicates that a receive buffer has encountered an underflow or overflow error. The RXBER contains each LC buffer status.
7	TXBUFERR	**TX Buffer Error:** Indicates that a transmit buffer has encountered an underflow of overflow error. The TXBER contains each LC buffer status.

Table 41: **MOST Interrupt Register *(Cont'd)***

Bit(s)	Name	Description
8	RXASYNEOP	**RX ASYNC End of Packet:** Indicates the RX asynchronous channel has received the end of the packet.
9	TXASYNEOP	**TX ASYNC End of Packet:** Indicates the TX asynchronous channel has transmitted the end of packet
10	CTLRXNAK	**Control Message Not Acknowledged:** Indicates a receive message could not be ACKed because the Control Message Receive FIFO was full.
11	CTLRXNEW	**Control Message Reception New:** Indicates a new receive control message has been received into the Control Message Receive FIFO
12	CTLTXFREE	**Control Message Transmission Free:** Indicates that a control message has just been de-queued from the Control Message Transmit FIFO. A user can now add another transmit control message.
12	CTLTXFREE	**Control Message Transmission Free:** Indicates that a control message has just been de-queued from the Control Message Transmit FIFO. A user can now add another transmit control message.
13	CTLTXFAIL	**Control Message Transmission Failed:** Indicates that the control message transmission failed (expired retry).(from the user Transmit Control Message FIFO)
14	CTLTXOK	**Control Message Transmission Completed:** Indicates that a control message transmission successfully completed - Ack received. (from the user Transmit Control Message FIFO)
15	CTLARBLST	**Control Message Arb Lost:** Indicates that arbitration to transmit a new control message was lost.
16	ACRCRXERR	**Async Receive CRC Error**: Indicates that a CRC error occurred during the reception of a an asynchronous message. (note that the message is still received)
17	RSVD	**Reserved:** Reads return zero and writes have no effect.
18	PERR	**Parity Error :** Indicates that a parity error occurred during reception.
19	BPCV	**Bi-phase Coding Violation Occurred:** Indicates that a coding violation has occurred during reception (outside of a preamble).
20	MDLYCHG	**Max Delay Change:** Indicates a change in maximum delay has occurred.
21	MPOSCHG	**Max Position Change:** Indicates a change in the maximum position has occurred
22	DLYCHG	**Delay Change:** Indicates a change in delay has occurred
23	POSCHG	**Position Change:** Indicates a change in position has occurred.
24	RSVD	**Reserved:** Reads return zero and writes have no effect.
25	LONCHG	**LON Change:** Indicates a change in the LON has occurred.
26	SBDCHG	**SBD Change:** Indicates a change in the SBD has occurred.
27	SYNCCHG	**SYNC Change:** Indicates a change in the frame, block or super block sync has occurred. The Status Register (SR) indicates the synchronization level obtained.
28	RSVD	**Reserved:** Reads return zero and writes have no effect.
29	RWP	**Received Wake-up Preamble:** Indicates that the MOST controller has received a wake-up preamble.
30-31	RSVD	**Reserved:** Reads return zero and writes have no effect.

Interrupt bits in the MISR can be cleared by writing to the Interrupt Clear Register (MICR) or by directly writing the appropriate bit with 0. For all bits in the MISR, a set condition takes priority over the clear condition, and the bit remains 1.

The MISR is a read/write register whose bits can be directly set and cleared for diagnostic purposes. All bits are initialized to 0 on the assertion of reset.

MOST Interrupt Pending Register

The MOST Interrupt Pending Register (MIPR) indicates the interrupts that are actually indicated to software. In effect, it is the MISR register bit wise ANDed with the MIER. The MIPR is a read-only register; writes have no effect. All bits are initialized to 0 on the assertion of reset.

MOST Interrupt Enable Register

The MOST Interrupt Enable Register (MIER) is used to enable or disable the generation of the interrupts. Interrupts are enabled by setting the appropriate bit in the MIER. MIER is a read/write register. All bits are initialized to 0 on the assertion of reset.

MOST Interrupt Clear Register

The MOST Interrupt Clear Register (MICR) is used to clear interrupt status bits. Writes of 1 clear the appropriate bit in the MOST Interrupt Status Register (MISR) and correspondingly in the MIPR.

The MICR is a write-1-to-clear register. Users must write a 1 to the appropriate bit position to clear the indicated interrupt. Reads return 0 in each bit position. All bits are initialized to 0 on the assertion of reset.

Buffer Interrupt Control and Status Registers

Buffer interrupts are controlled through four registers:

- **Status (BISR)**: Indicates the interrupt status bits which are set and cleared regardless of the status of the corresponding bit in the Interrupt Enable Register (BIER)
- **Pending (BIPR):** Indicates the interrupts that are actually indicated to software. In effect, it is the BISR register bit-wise ANDed with the BIER
- **Enable (BIER):** Determines if an interrupt is generated or not
- **Clear (BICR):** Clears the status bits in the BISR and BIPR register when a 1 is written to the corresponding bit in the Interrupt Clear Register (BICR)

Triggering Buffer Interrupts

Two conditions cause the Buffer Interrupt signal to assert:

- If a bit in the BISR is 1 and the corresponding bit in the BIER is 1
- Changing an BIER bit from 0 to 1 when the corresponding bit in the BISR is already 1

An interrupt is generated and the Pending Register bit (BIPR) is set if the corresponding mask bit in the Enable Register (BIER) is set.

Clearing Buffer Interrupts

Two conditions cause the Buffer Interrupt signal to deassert:

- Clearing a bit in the BISR/BIPR that is 1 (by writing a 1 to the corresponding bit in the MICR)
- Changing an BIER bit from 1 to 0 when the corresponding bit in the BISR is 1

When deassertion and assertion conditions occur simultaneously, the Buffer interrupt line is deasserted first, and then reasserted if the assertion condition remains true.

Common Bit Map

As the Status, Pending, Enable and Clear register have common mapping between registers, only a single mapping is depicted. Each has a unique register and address map as defined in table Table 8 on page 21.

Each bit has a suffix:

- **Interrupt Status**: INT
- **Pending**: PEN
- **Enable**: ENA
- **Clear**: CLR

The following table shows the BIR bits:

Table 42: **Buffer Interrupt Register bits**

0	1	2	3	4	5	6	7
TXLC15	TXLC14	TXLC13	TXLC12	RSVD	RSVD	RSVD	RSVD
8	**9**	**10**	**11**	**12**	**13**	**14**	**15**
TXLC7	TXLC6	TXLC5	TXLC4	TXLC3	TXLC2	TXLC1	TXLC0
16	**17**	**18**	**19**	**20**	**21**	**22**	**23**
RSVD	RSVD	RSVD	RSVD	RXLC11	RXLC10	RXLC9	RXLC8
24	**25**	**26**	**27**	**28**	**29**	**30**	**31**
RXLC7	RXLC6	RXLC5	RXLC4	RXLC3	RXLC2	RXLC1	RXLC0

Buffer Interrupt Status Register

The Buffer Interrupt Status Register (BISR) contains bits that are set when a specific interrupt condition occurs. If the corresponding mask bit in the Buffer Interrupt Enable Register (BIER) is set, `BUFFER_irpt` is generated. Interrupt bits in the BISR can be cleared by writing to the Interrupt Clear Register (BICR) or by directly writing the appropriate bit with a 0. For all bits in the BISR, a set condition takes priority over the clear condition, and the bit remains 1.

A Buffer Interrupt bit is set when the threshold for the logical channel buffer indicated is crossed and that particular logical channel is enabled (LCE =1). For example, with the FW parameter set to 1/4 for a receive buffer, a buffer interrupt is asserted when the buffer receives 8, 16, 24, or 32 words, and that logical channel is enabled through the corresponding LCE bit. Because there is no read access of the RX egress channels or write access to the TX ingress channels through the Host Interface, these bits are reserved.

BISR is a read/write register and bits can be directly set and cleared under user control for diagnostic purposes. All bits are initialized to 0 on the assertion of reset.

Buffer Interrupt Pending Register

The Buffer Interrupt Pending register (BIPR) indicates the interrupts that are actually indicated to software. In effect it is the BISR register bit wise ANDed with the BIER register. BIPR is a read-only register; writes have no effect. All bits are initialized to 0 on the assertion of reset.

Buffer Interrupt Enable Register

The Buffer Interrupt Enable Register (BIER) is used to enable or disable the generation of the interrupts. Interrupts are enabled by setting the appropriate bit in the BIER. BIER is a read/write register. All bits are initialized to 0 on the assertion of reset.

Buffer Interrupt Clear Register

The Buffer Interrupt Clear Register (BICR) is used to clear interrupt status bits. Writes of 1 clear the appropriate bit in the Buffer Interrupt Status Register (BISR) and the BIPR

BICR is a write-1-to-clear register. Users must write a 1 to the appropriate bit position to clear the indicated interrupt; reads return 0 in each bit position. All bits are initialized to 0 on the assertion of reset.

Routing Table

Common Routing Table

The Common Routing Table (CRT) is used to configure the mapping of time slots to logical channels for both the transmit and receive data paths of the MOST controller.

The MOST frame contains 60 bytes of synchronous or asynchronous data. Up to 60 bytes can be defined as synchronous time slots; the remaining bytes belong to the asynchronous channel. Each word in the routing table controls the mapping of four time slots. Logical channel number 0 is reserved for asynchronous data and as such can not be used for any synchronous time slots. Mapping a time slot to a transmit egress channel or a receive ingress channel is not permitted because there no connection from these channels to a ring. If this is done, the core either transmits or receives corrupted data; however, any frames remain well-formed.

The CRT is stored in internal RAM, and for this reason is not initialized on reset. The following table describes the Transmit Routing table memory map.

Table 43: **Common Routing Table**

Time Slots	Direction	Address	Size (Words)	Access
0:3	Transmit	C_BASEADDR + 0x0200	1	Read/Write
4:7	Transmit	C_BASEADDR + 0x0204	1	Read/Write
8:11	Transmit	C_BASEADDR + 0x0208	1	Read/Write
12:15	Transmit	C_BASEADDR + 0x020C	1	Read/Write
16:19	Transmit	C_BASEADDR + 0x0210	1	Read/Write
20:23	Transmit	C_BASEADDR + 0x0214	1	Read/Write
24:27	Transmit	C_BASEADDR + 0x0218	1	Read/Write
28:31	Transmit	C_BASEADDR + 0x021C	1	Read/Write
32:35	Transmit	C_BASEADDR + 0x0220	1	Read/Write
36:39	Transmit	C_BASEADDR + 0x0224	1	Read/Write
40:43	Transmit	C_BASEADDR + 0x0228	1	Read/Write
44:47	Transmit	C_BASEADDR + 0x022C	1	Read/Write
48:51	Transmit	C_BASEADDR + 0x0230	1	Read/Write

Table 43: **Common Routing Table *(Cont'd)***

Time Slots	Direction	Address	Size (Words)	Access
52:55	Transmit	C_BASEADDR + 0x0234	1	Read/Write
56:59	Transmit	C_BASEADDR + 0x0238	1	Read/Write
Async	Transmit	C_BASEADDR + 0x023C	1	Read/Write
0:3	Receive	C_BASEADDR + 0x0240	1	Read/Write
4:7	Receive	C_BASEADDR + 0x0244	1	Read/Write
8:11	Receive	C_BASEADDR + 0x0248	1	Read/Write
12:15	Receive	C_BASEADDR + 0x024C	1	Read/Write
16:19	Receive	C_BASEADDR + 0x0250	1	Read/Write
20:23	Receive	C_BASEADDR + 0x0254	1	Read/Write
24:27	Receive	C_BASEADDR + 0x0258	1	Read/Write
28:31	Receive	C_BASEADDR + 0x025C	1	Read/Write
32:35	Receive	C_BASEADDR + 0x0260	1	Read/Write
36:39	Receive	C_BASEADDR + 0x0264	1	Read/Write
40:43	Receive	C_BASEADDR + 0x0268	1	Read/Write
44:47	Receive	C_BASEADDR + 0x026C	1	Read/Write
48:51	Receive	C_BASEADDR + 0x0270	1	Read/Write
52:55	Receive	C_BASEADDR + 0x0274	1	Read/Write
56:59	Receive	C_BASEADDR + 0x0278	1	Read/Write
Async	Receive	C_BASEADDR + 0x027C	1	Read/Write

Synchronous Routing Table Word Organization

The following table identifies how synchronous time slots are organized within each word.

Table 44: **Synchronous Routing Table Word Organization**

0 - 7	8 - 15	16- 23	24 - 31
Time slot N	Time slot N+1	Time slot N+2	Time slot N+3

Synchronous Routing Table Time Slot/Async Organization

The following table identifies how synchronous time slots are configured within each byte.

Table 45: **Synchronous Routing Table Bits**

Bit	Name	Access	Defaul t Value	Description
0	Enable	Read/Write	0	Enable: Controls whether or not the MOST controller receives or transmits on this time slot. 1 = MOST controller uses this time slot 0 = MOST controller does not use this timeslot
1-3	RSVD	Read/Write	0	Reserved: Reads to return zero and writes have no effect
4-7	LC#	Read/Write	0	Logical Channel Number: Indicates which logical channel the timeslot is mapped to. May not be 0x0.

Asynchronous Routing Word Organization

The following table identifies how the asynchronous channel is organized within each word.

Table 46: **Asynchronous Routing Table Word Organization**

0 - 7	8 - 15	16- 23	24 - 31
Async	RSVD	RSVD	RSVD

Asynchronous Routing Table Organization

The following table shows how the asynchronous byte is configured.

Table 47: **Asynchronous Routing Table Bit**

Bit	Name	Access	Default Value	Description
0	Enable	Read/Write	0	**Enable:** Controls whether or not the MOST controller receives or transmits on the asynchronous channel. 1 = MOST controller uses the async channel 0 = MOST controller does not use the async channel
1-3	RSVD	Read/Write	0	**Reserved:** Reads to return zero and writes have no effect
4-7	LC#	Read/Write	0	**Logical Channel Number:** Indicates which logical channel the async channel is mapped to. Must be 0x0.

Logical Channel Enable

The following tables show the Logical Channel Enable (LCE) register which is used to enable or disable channels on a logical channel basis. Writes to the LCE take effect at the next frame boundary; reads indicate the current state of the logical channels.

Table 48: **Logical Channel Enable Register Bits**

0	1	2	3	4	5	6	7
TXLCE15	TXLCE14	TXLCE13	TXLCE12	TXLCE11	TXLCE10	TXLCE9	TXLCE8
8	**9**	**10**	**11**	**12**	**13**	**14**	**15**
TXLCE7	TXLCE6	TXLCE5	TXLCE4	TXLCE3	TXLCE2	TXLCE1	TXLCE0
16	**17**	**18**	**19**	**20**	**21**	**22**	**23**
RXLCE15	RXLCE14	RXLCE13	RXLCE12	RXLCE11	RXLCE10	RXLCE9	RXLCE8
24	**25**	**26**	**27**	**28**	**29**	**30**	**31**
RXLCE7	RXLCE6	RXLCE5	RXLCE4	RXLCE3	RXLCE2	RXLCE1	RXLCE0

Table 49: **Logical Channel Enable Bit Description**

Bit	Name	Access	Default Value	Description
0 - 31	(R/T)XLCE_*N*_	Read/Write	0	**Enable Logical Channel:** Enables or disables a logical channel. 1 = Enable the indicated *(_N_)* logical channel 0 = Disable the indicated *(_N_)* logical channel.

Slave Active Register 1

The following tables show the Slave Active Register (SAR1) which reflects the active bits set by slave nodes during the transmission of the Network Information Channel on the MOST ring. As a result, this register takes some time to stabilize at initialization and after every unlock event. Not all bits are accurate. The accuracy depends on the position of the node within the ring.

Table 50: **Slave Active Register 1-bit Description**

Bit(s)	Name	Access	Default Value	Description
0 - 31	ACT_*N*_	Read Only	Not initialized	**Active:** Slave reports timeslot active status. 1 = Time slot indicated *(_N_)* is active 0 = Timeslot indicated *(_N_)* is not active, or not reported yet, due to owning node being downstream.

Table 51: **Slave Active Register 1 Bits**

0	1	2	3	4	5	6	7
ACT31	ACT30	ACT29	ACT28	ACT27	ACT26	ACT25	ACT24
8	**9**	**10**	**11**	**12**	**13**	**14**	**15**
ACT23	ACT22	ACT21	ACT20	ACT19	ACT18	ACT17	ACT16
16	**17**	**18**	**19**	**20**	**21**	**22**	**23**
ACT15	ACT14	ACT13	ACT12	ACT11	ACT10	ACT9	ACT8
24	**25**	**26**	**27**	**28**	**29**	**30**	**31**
ACT7	ACT6	ACT5	ACT4	ACT3	ACT2	ACT1	ACT0

Slave Active Register 2

The following tables show Slave Active Register (SAR2) which reflects the active bits set by slave nodes during the transmission of the Network Information Channel on the MOST Ring, as a result, this register will take some time to stabilize at initialization and after every unlock event. Not all bits are necessarily accurate, the accuracy depends on the position of the node within the ring.

Table 52: **Slave Active Register 2-bit Description**

Bit(s)	Name	Access	Default Value	Description
0 - 31	ACT_*N*_	Read Only	Not initialized	**Active:** Slave reports timeslot active status. 1 = Time slot indicated *(_N_)* is active 0 = Timeslot indicated *(_N_)* is not active, or not reported yet, due to owning node being downstream.

Table 53: **Slave Active Register 2 Bits**

0	1	2	3	4	5	6	7
RSVD	RSVD	RSVD	RSVD	ACT59	ACT58	ACT57	ACT56
8	**9**	**10**	**11**	**12**	**13**	**14**	**15**
ACT55	ACT54	ACT53	ACT52	ACT51	ACT50	ACT49	ACT48
16	**17**	**18**	**19**	**20**	**21**	**22**	**23**
ACT47	ACT46	ACT45	ACT44	ACT43	ACT42	ACT41	ACT40
24	**25**	**26**	**27**	**28**	**29**	**30**	**31**
ACT39	ACT38	ACT37	ACT36	ACT35	ACT34	ACT33	ACT32

Master Allocation Table

The Master Allocation Table (MAT) is used to read the assignments of synchronous time slots to various controllers on the ring. This table is valid only when the core is in Master mode. The table is not cleared when changing the operating mode from Master to Slave. If this is a requirement, send a self-addressed message to the node deallocating all the channels, or apply a soft reset to the core to ensure an empty table. Self-deallocation of all channels can also be used if there is change in synchronous boundary descriptor.

Each word in the table reports the mapping of four time slots. The last word (0x02FC) is unused and the value returned will be indeterminate.

The MAT is stored in internal RAM, and for this reason is not initialized on reset, though it will be loaded with unallocated tags as the core starts operation.Time slots are configured as follows for each byte. For an allocated time slot, the connection label is the first time slot of that allocation cluster. For an unallocated time slot, the connection label is the unallocated tag. The following table describes the memory map.

Table 54: **Master Allocation Table**

Time Slots	Address	Size (Words)	Access
0:3	C_BASEADDR + 0x02C0	1	Read Only
4:7	C_BASEADDR + 0x02C4	1	Read Only
8:11	C_BASEADDR + 0x02C8	1	Read Only
12:15	C_BASEADDR + 0x02CC	1	Read Only
16:19	C_BASEADDR + 0x02D0	1	Read Only
20:23	C_BASEADDR + 0x02D4	1	Read Only
24:27	C_BASEADDR + 0x02D8	1	Read Only
28:31	C_BASEADDR + 0x02DC	1	Read Only
32:35	C_BASEADDR + 0x02E0	1	Read Only
36:39	C_BASEADDR + 0x02E4	1	Read Only
40:43	C_BASEADDR + 0x02E8	1	Read Only
44:47	C_BASEADDR + 0x02EC	1	Read Only
48:51	C_BASEADDR + 0x02F0	1	Read Only

Table 54: **Master Allocation Table *(Cont'd)***

Time Slots	Address	Size (Words)	Access
52:55	C_BASEADDR + 0x02F4	1	Read Only
56:59	C_BASEADDR + 0x02F8	1	Read Only
Unused	C_BASEADDR + 0x02FC	1	Read Only

Master Allocation Table Word Organization

Time slots are organized as follows within each word.

Table 55: **Master Allocation Table Word Organization**

0 - 7	8 - 15	16- 23	24 - 31
Time slot N	Time slot N+1	Time slot N+2	Time slot N+3

Master Allocation Table Time slot Organization

Time slots are configured as follows within each byte.

Table 56: **Routing Table Time Slot Organization**

0	1:7
Allocated	Connection Label (6:0)

Streaming Port MMR

The Streaming Port MMR bus (STR_MMR) is the portion of the memory map which is dedicated to external modules connected to the streaming ports. Any accesses to this portion of the memory mapped are forwarded to the streaming port via the STR_MMR.

Writes from the PLB bus to the STR_MMR bus are not posted. As such, the Streaming PORT MMR acknowledge must be asserted within 12 cycles.

If the STRON bit in the MSR is set to 0, MMR accesses to the streaming port are disabled. The MOST controller will ACK PLB access in this circumstance and return zeroes and the Read data, and discard Write data.

MOST PLL Control Registers

PLL Reference Count Register

This register sets the reference count value used by the PLL clock monitoring logic to evaluate the quality of the clock generated by the PLL. This logic is active immediately when the core is brought out of reset. If a problem was detected, the core will automatically reset the external PLL through assertion of MOST_PLL_NRESET. The circuit evaluates the incoming clock at regular intervals to ensure that the external PLL is still generating the good quality clock. The value needs to be programmed based on the frequency of `SPLB_Clk` according to the formula below:

`SAMPLE_POINT` X `SPLB_Clk` (MHz) / `MOST_PLL_CLK` (MHz), rounded to the closest integer, where `SAMPLE_POINT` = 0xA4 (constant).

The default value of PRCR is 0x20. This register should be programmed when MENA is zero.

Example:

When SPLB_Clk = 50 MHz, MOST_PLL_CLK = 45.12 MHz, then PLL Reference Count is 181.73, ceiled to 182 (0x0B6).

When SPLB_Clk = 70 MHz, MOST_PLL_CLK = 45.12 MHz, PLL Reference Count is 254.43, floored to 254 (0x0FE).

Table 57: **PLL Reference Count Register Organization**

Bit(s)	Name	Access	Default value	22:31
0 - 21	RSVD	Read/Write	0	**Reserved:** Reads return zero and writes have no effect.
22 - 31	PRC	Read/Write	32	**PLL Reference Count:** Indicates the reference count value used by the PLL correction logic to evaluate the quality of the clock generated by the PLL. This should be programmed when MENA is zero.

External PLL Reset Count

This register indicates the number of times the external PLL was reset. This counter resets when a hard or soft reset is applied. This register resets only when the core is disabled. Disabling MENA also resets PLL reset count register. The maximum value of this counter is 255 and does not roll over.

Table 58: **External PLL Reset Count Organization**

Bit(s)	Name	Access	Default value	22:31
0 - 23	RSVD	Read/Write	0	**Reserved:** Reads return zero and writes have no effect.
24 - 31	EPRC	Read	0	**External PLL Reset Count:** Indicates the number of times Reset was applied to external PLL by the external PLL logic when the quality of the reference clock is not good. The maximum value is 255 and does not roll over.

Table 59: **External PLL Reset Count Organization**

0-23	24:31
Reserved	External PLL reset count(7:0)

Transmit / Receive Buffer

Transmit Buffer

This portion of the memory map allows for FIFO based access to the transmit buffers. Sufficient memory map address space is provided to allow random access to the transmit buffer or to allow incrementing of addresses in the case of a burst transaction. However, the core is currently implemented to provide FIFO based access to the transmit buffer.

The transmit buffer is organized on a logical channel basis. The transmit buffer supports a total of 16 logical channels. Each buffer contains 32 words. To ease in programming, the Logical Channel Number forms the 3rd Most Significant Nibble of the Transmit Buffer Address. As such the address space supports 64 words per logical channel, where an access to any word in the space is forwarded to the

appropriate channel, even though the underlying memory content is only 32 words deep. The following table describes the Transmit Buffer memory map.

Table 60: **Transmit Buffer Memory Map**

Logical Channel	Function	Address	Size (words)	Access
0x0	PLB access	C_BASEADDR + 0x1000	64	PLB Write; MOST Read
0x1	PLB access	C_BASEADDR + 0x1100	64	PLB Write; MOST Read
0x2	PLB access	C_BASEADDR + 0x1200	64	PLB Write; MOST Read
0x3	PLB access	C_BASEADDR + 0x1300	64	PLB Write; MOST Read
0x4	PLB access	C_BASEADDR + 0x1400	64	PLB Write; MOST Read
0x5	PLB access	C_BASEADDR + 0x1500	64	PLB Write; MOST Read
0x6	PLB access	C_BASEADDR + 0x1600	64	PLB Write; MOST Read
0x7	PLB access	C_BASEADDR + 0x1700	64	PLB Write; MOST Read
0x8	Streaming Ingress	C_BASEADDR + 0x1800	64	STR Write; MOST Read
0x9	Streaming Ingress	C_BASEADDR + 0x1900	64	STR Write; MOST Read
0xA	Streaming Ingress	C_BASEADDR + 0x1A00	64	STR Write; MOST Read
0xB	Streaming Ingress	C_BASEADDR + 0x1B00	64	STR Write, MOST Read
0xC	PLB access/ Streaming Egress	C_BASEADDR + 0x1C00	64	PLB Write; STR Read
0xD	PLB access/ Streaming Egress	C_BASEADDR + 0x1D00	64	PLB Write; STR Read
0xE	PLB access/ Streaming Egress	C_BASEADDR + 0x1E00	64	PLB Write; STR Read
0xF	PLB access/ Streaming Egress	C_BASEADDR + 0x1F00	64	PLB Write; STR Read

Accessing the Transmit Buffer

The logical channel transmit buffers must be accessed as a FIFO; that is, each word for transmission must be written to the appropriate FIFO in sequential order. The lowest 8 bits of the address are ignored when accessing the transmit buffer. The only bits of significance are the logical channel bits.

The bytes within each word are ordered as follows.

Table 61: **Transmit Buffer Byte Organization**

0 - 7	8 - 15	16- 23	24 - 31
Byte N	Byte N+1	Byte N+2	Byte N+3

The first word written to the transmit buffer will be the first word transmitted on the MOST ring.

Transmit Synchronous Message Format

Synchronous data has no header information; the content is raw data. As such the transmit buffer is written with data ordered as indicated in Table 61 on page 50. A user always writes data in words. The last word may contain less than 4 bytes of actual data. In such a case the user should pad unused data locations with `0x00` in order to mute the channel on completion of transmission.

Transmit Asynchronous Message Format

Asynchronous data contains a header field for the destination address, length and source address, followed by data. The format is as follows:

Table 62: **Asynchronous Message: First Word (Header)**

0-15	16-23	24-31
Destination Address	Length	RSVD

The first word of the Asynchronous message contains the destination address and length. The hardware will only transmit as many bytes as indicated by length. Length is calculated in the following way:

Length (in bytes) = (Packet Length -1)*4 +2

Table 63: **Asynchronous Message: Second Word (Header)**

0-15	16-31
Source Address	RSVD

The second word of the transmit asynchronous message contains the Source address.

Table 64: **Asynchronous Message: Third to Nth Word**

0 - 7	8 - 15	16- 23	24 - 31
Data Byte N	Data Byte N+1	Data Byte N+2	Data Byte N+3

The 3rd to Nth word contains the actual data of the transmit message. A user always writes data in words. The last word may contain less than 4 bytes of actual data. In such a case the user should pad unused data locations with `0x00`.

The asynchronous message CRC is calculated and appended automatically by hardware.

The TXBER will indicate any overflow or underflow error conditions on a logical channel basis. If enabled, the TXBUFERR in the MISR will be asserted on any TX buffer errors.

Receive Buffer

This portion of the memory map allows for FIFO-based access to the receive buffers. Sufficient memory map address space is provided to allow random access to the receive buffer. However, the core is currently implemented to provide FIFO based access to the receive buffer.

The receive buffer is organized on a logical channel basis. The receive buffer supports a total of 16 logical channels. Each buffer contains 32 words. To ease in programming the Logical Channel Number forms the 3rd Most Significant Nibble of the Receive Buffer Address. As such the address space supports 64 words per logical channel, where an access to any word in the space is forwarded to the appropriate channel, even though the underlying memory content is only 32 words deep. The following table defines the Receive Buffer Memory Map.

Table 65: **Receive Buffer Memory Map**

Logical Channel	Function	Address	Size (words)	Access
0x0	PLB access	C_BASEADDR + 0x2000	64	MOST Write/PLB Read
0x1	PLB access	C_BASEADDR + 0x2100	64	MOST Write/PLB Read
0x2	PLB access	C_BASEADDR + 0x2200	64	MOST Write/PLB Read
0x3	PLB access	C_BASEADDR + 0x2300	64	MOST Write/PLB Read
0x4	PLB access	C_BASEADDR + 0x2400	64	MOST Write/PLB Read
0x5	PLB access	C_BASEADDR + 0x2500	64	MOST Write/PLB Read
0x6	PLB access	C_BASEADDR + 0x2600	64	MOST Write/PLB Read
0x7	PLB access	C_BASEADDR + 0x2700	64	MOST Write/PLB Read
0x8	PLB access/ Streaming Ingress	C_BASEADDR + 0x2800	64	STR Write/PLB Read
0x9	PLB access/ Streaming Ingress	C_BASEADDR + 0x2900	64	STR Write/PLB Read
0xA	PLB access/ Streaming Ingress	C_BASEADDR + 0x2A00	64	STR Write/PLB Read
0xB	PLB access/ Streaming Ingress	C_BASEADDR + 0x2B00	64	STR Write. PLB Read
0xC	Streaming Egress	C_BASEADDR + 0x2C00	64	MOST Write. STR Read
0xD	Streaming Egress	C_BASEADDR + 0x2D00	64	MOST Write. STR Read
0xE	Streaming Egress	C_BASEADDR + 0x2E00	64	MOST Write. STR Read
0xF	Streaming Egress	C_BASEADDR + 0x2F00	64	MOST Write. STR Read

Accessing the Receive Buffer

The logical channel receive buffers must be accessed as a FIFO; that is, each word for reception must be read from the appropriate FIFO in sequential order. The lowest 8 bits of the address are ignored when accessing the receive buffer. The only bits of significance are the logical channel bits. The following table defines the byte organization of the Receive Buffer. The first word read from the receive buffer is the first word received on the MOST ring.

Table 66: **Receive Buffer Byte Organization**

0 - 7	8 - 15	16- 23	24 - 31
Byte N	Byte N+1	Byte N+2	Byte N+3

Receive Synchronous Message Format

Synchronous data has no header information. The content is raw data. As such the receive buffer is read with data ordered as indicated in Table 66. A user always reads data in words. The last word can contain less than 4 bytes of actual data.

Receive Asynchronous Message Format

Asynchronous data contains a header field for the destination address, length, source address and address mode, followed by data, followed by the CRC, and including a status field. The following table defines the format.

Table 67: **Asynchronous Message: First Word (Header)**

0-15	16-23	24-31
Destination Address	Length	RSVD

The first word of the receive asynchronous message contains the Destination address and Length. The hardware will only receive as many bytes as indicated by Length. The second word of the receive asynchronous message contains the Source address.

Table 68: **Asynchronous Message: Second Word (Header)**

0-15	16-31
Source Address	RSVD

The 3rd to Nth word contains the actual data of the receive message. A user always reads data in words. The last word may contain less than 4 bytes of actual data. In such a case the unused data locations will be padded with 0x00.

Table 69: **Asynchronous Message: Third to Nth Word**

0 - 7	8 - 15	16- 23	24 - 31
Data Byte N	Data Byte N+1	Data Byte N+2	Data Byte N+3

The last word contains the status for the message. CRC checking is done by hardware and reported. Status is encoded as defined in Table 70 and Table 71.

Table 70: **Asynchronous Message: Last Word**

0 - 3	4 - 7	8 - 31
RSVD	Status	RSVD

Table 71: **Status Encoding**

Value	Meaning
0000b	RX message is valid
0001b	RX message has a CRC error
001xb	RX message has a length error
0100b - 1111b	reserved - not used

The RXBER indicates any overflow or underflow error conditions on a logical channel basis. If enabled, the RXBUFERR in the MISR will be asserted on any RX buffer errors.

Interrupt Processing for Receive Asynchronous Data

Receive Asynchronous data can be shorter than the Full Word Count trigger. However, a user will want to be notified and receive this packet of data before the full word count threshold is crossed by receiving subsequent asynchronous packets.

When the end of an asynchronous packet is reached, the RXASYNEOP interrupt is triggered. The internal logic pads the buffer up to the next word count location in the buffer, and the word count logic is also reset.

The Logical Channel Buffer interrupt for an asynchronous buffer is triggered only when the word count threshold is crossed. The RXASYNEOP interrupt is triggered on every end of asynchronous packet reception.

The user only has to read the valid data of the packet. After the user reads the last word (which contains the status), the buffer is automatically cleared up to the next word count boundary.

Timing Diagrams

PLB single read

Figure 7 shows the timing interface for single read transfer. `Sl_addrAck`, `Sl_rdDAck`, and `Sl_rdComp` must be asserted within 16 PLB clock cycles after the assertion of `PLB_PAValid`, including the clock cycle where the `PLB_PAValid` is asserted.

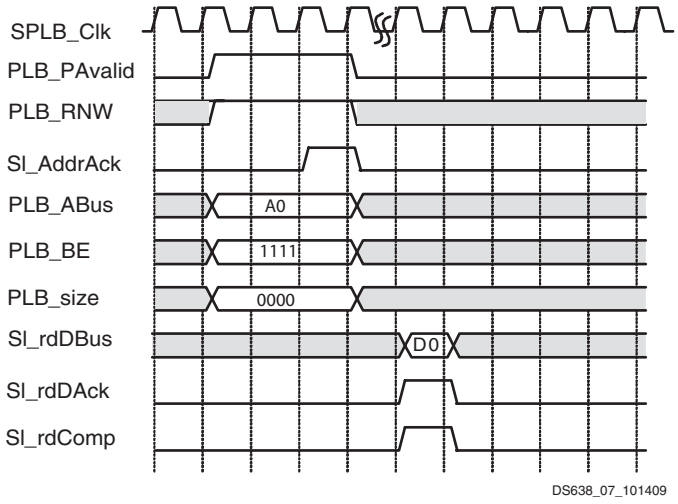

Figure 7: **Single Read Transfer**

PLB single write

Figure 8 shows the timing interface for single write transfer. The `Sl_addrAck`, `Sl_wrDAck`, and `Sl_wrComp` must be asserted within 16 PLB clock cycles after the assertion of `PLB_PAValid`, including the clock cycle where the `PLB_PAValid` is asserted.

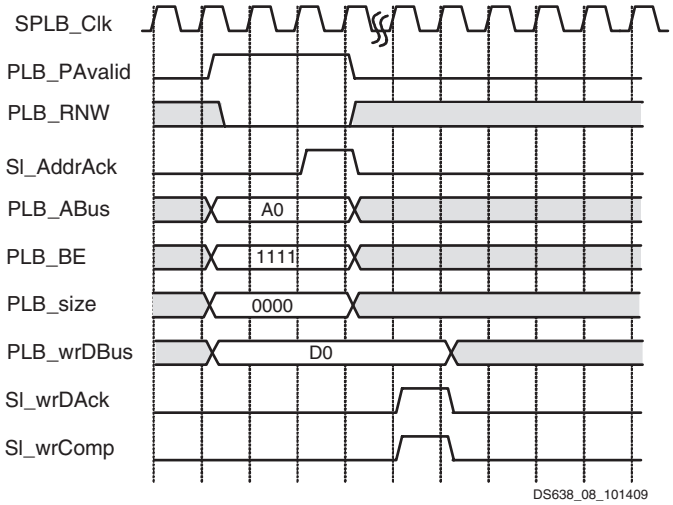

Figure 8: **Single Write Transfer**

PLB burst read

Figure 9 shows the timing interface for burst read transfer. The `Sl_addrAck` must be asserted within 16 PLB clock cycles after the assertion of `PLB_PAValid` (including the clock cycle where the `PLB_PAValid` is asserted).

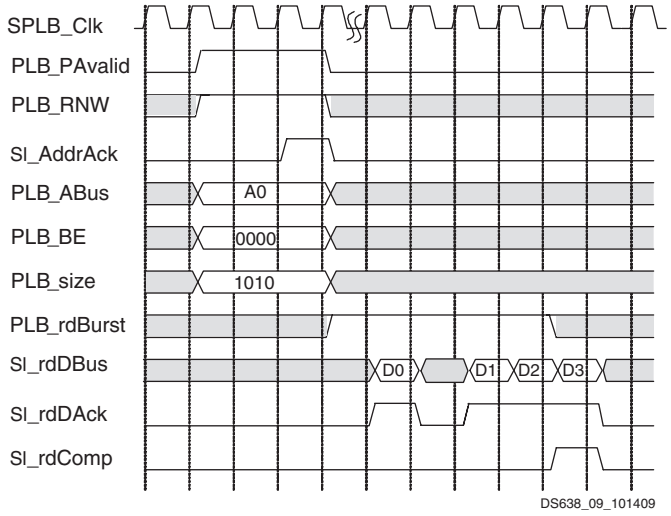

Figure 9: **Burst Read Transfer**

This timing diagram shows where the read data is continuously given in every clock. However, throttled burst is allowed where the data need not be provided in each clock. In this case, DUT may take few clock cycles between two data.

PLB burst write

Figure 10 shows the timing interface for burst write transfer. The `Sl_addrAck` must be asserted within 16 PLB clock cycles after the assertion of `PLB_PAValid` (including the clock cycle where the `PLB_PAValid` is asserted).

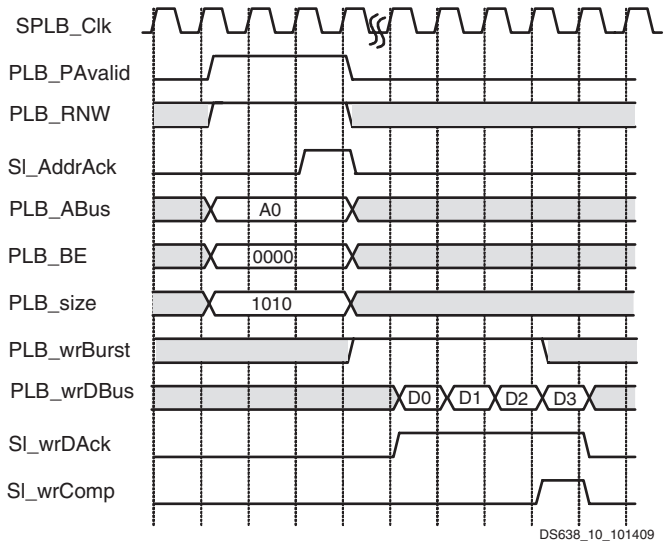

Figure 10: **Burst Write Transfer**

Streaming Port Receive Egress (Read from Streaming Port)

The streaming ports use LocalLink signaling, where either the core or the external requesting interface are free to negate ready, indicating that a transmission is stalled for this cycle. Only when both source and destination ready are asserted does a transfer occur. The user can change the logical channel at any time as long `*_DST_RDY` is deasserted the previous cycle, although this can result in less than full throughput. Typically, the number of bytes read should match four times the word count as configured by the full word count parameter (`FWC`). Figure 11 illustrates two example transactions where the external requestor stalls the access on logical channel 0 and the MOST NIC stalls the access on logical channel.

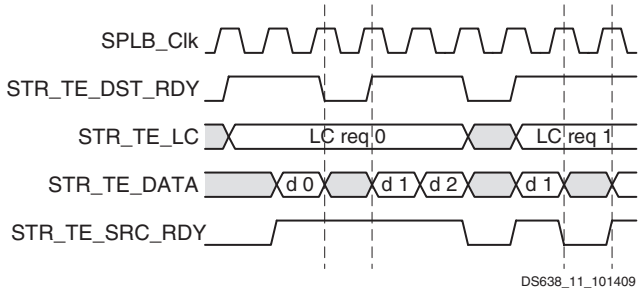

Figure 11: **Receive Streaming Port Read**

Streaming Port Receive Ingress (Write to Streaming Port)

The write port typically, should have a number of bytes written to match four times the word count, as configured by the empty word count parameter (EWC). An application can monitor the standard word count flags to trigger this access, or, alternatively, the hardware component can continue accessing data

from any given channel, which the core provides as long as data exists. The following figure illustrates sample transactions where the external requestor is stalling on logical channel 0, and the MOST NIC is stalling on logical channel 1.

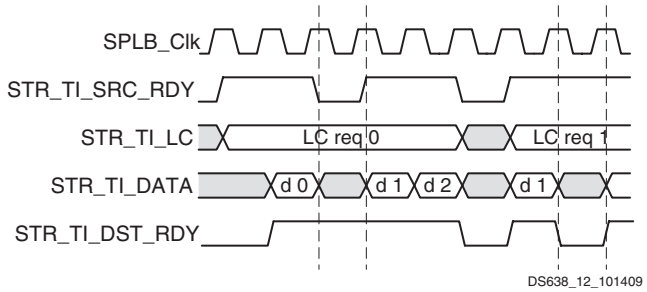

Figure 12: Receive Streaming Port Write

Streaming Port Transmit Egress and Ingress

The operation of the Transmit Egress streaming port is similar to the Receive Egress streaming port. The operation of the Transmit Ingress streaming port is similar to the Receive Ingress streaming port.

Programming the Core

This section describes how to program the MOST controller for a variety of operations.

Configuring the Controller

For a Ring (Slave)

After a hard or soft reset, follow these steps to configure the controller for a ring (slave):

1. Ensure the core is reset by writing a '1' to the SRST bit. This resets the core and forces MENA to be negated. This can be read back via the SRR.
2. Write a '0' to the SRST bit, and a '0' to the MENA to take the core out of reset, but allow configuration of the core.
3. Configure the controller for initial Bypass mode by writing '0' to each of MNSLV, RBDT, and LBACK, and '1' into BYPASS.
4. Program the PRCR based on the PLB clock frequency and PLL clock frequency.
5. Program the logical address in the LAR.
6. Program the alternate address in the AAR.
7. Program the group address in the GAR.
8. Program the message retry count in the MRCDR.
9. Confirm the status of the controller by reading the SR, which reports the current state of the core. To read back what was programmed to the core, read MSR.
10. Enable appropriate interrupts by writing '1' to locations as needed in the MIER
11. Clear all MOST interrupts by writing 0xFFFF to the MICR
12. Enable the controller by writing a '0' to SRST and '1' to MENA. The core then operates in Bypass mode, which allows synchronization without disturbing the ring.

13. Via interrupts or polling, confirm full synchronization is achieved by reading the SR. The SBLCK, BLCK, and FLCK bits indicate the synchronization level.
14. After synchronization is achieved, disable the controller again by writing '0' into both SRST and MENA.
15. Configure the controller to Normal mode by writing '0' to each of MNSLV, RBDT, LBACK, and BYPASS.
16. Re-enable the controller by writing a '0' to SRST and '1' to MENA. The core then starts Normal operations.
17. Read the value for the synchronous boundary descriptor in the CCR.
18. Read the value for the delay and position registers in the MCPDR.
19. The controller is now ready to send and transmit control, synchronous, and asynchronous data.

For a Ring (Master)

After either a hard or soft reset, follow these steps to configure the controller for a ring (master):

1. Ensure the core is reset by writing a '1' to the SRST bit. This resets the core and forces MENA to be negated. This can be read back via the SRR.
2. Write a '0' to the SRST bit, and a '0' to the MENA to take the core out of reset, but allow configuration of the core.
3. Configure the controller for Normal mode by writing '0' to each of RBDT, and LBACK, and BYPASS, and '1' into MNSLV.
4. Program the PRCR based on the PLB clock frequency and PLL clock frequency.
5. Program the logical address in the LAR.
6. Program the alternate address in the AAR.
7. Program the group address in the GAR.
8. Program the message retry count in the MRCDR.
9. Program the desired synchronous boundary descriptor into CCR.
10. Enable appropriate interrupts by writing '1' to locations as needed in the MIER
11. Clear all MOST interrupts by writing 0xFFFF to the MICR.
12. Enable the controller by writing a '0' to SRST and '1' to MENA. The core then operates in Normal mode.
13. Confirm the status of the controller by reading the SR, which reports the current state of the core. To read back what was programmed to the core, read MSR.
14. Via interrupts or polling confirm full synchronization of the ring is achieved by reading the SR. The SBLCK, BLCK, and FLCK bits indicate the synchronization level.
15. The controller is now ready to send and transmit control, synchronous, and asynchronous data.

Configuring a Transmit Synchronous Channel

To configure a logical channel for transmission, the following sequence occurs:

1. Send an allocation request transmit message with the number of synchronous timeslots required.
2. Receive the allocation response message. The response indicates the first eight synchronous time slots available, although the user is only granted the number of timeslots requested in Step 1.
3. A user then maps the time slots granted to logical channels. For this step, one logical channel is used for one time slot. (In this example, the logical channel two is used for time slot five). If the

logical channel buffer was used to transmit prior to the new allocation, it must be flushed to clear any stale data and to reset to a clean state.

4. Enable the logical channel buffer interrupts. Program TXLC2 in the BIER.
5. Program the routing table. Program time slot five (Address C_BASEADDR + 0x0204) to be set to logical channel two by programming the appropriate byte with 0x02.
6. Prime the logical channel buffer as described below.
7. Enable the logical channel by writing a 1 to TXELC2 in the Logical Channel Enable Register (LCE). At the next frame boundary, the controller starts to transmit data from the logical channel buffer on to the MOST ring.
8. Queue data for transmission by writing logical channel two's transmit buffer as buffer interrupts are received.

Transmitting Synchronous or Asynchronous Data

To transmit data in conjunction with the configuration sequence above, use the following sequence:

1. After reset, initialize the configuration and interrupt control registers as required.
2. Enable the core for Normal mode, and assert MENA in the SRR.
3. Allocate channels for synchronous data via sending allocation control messages.
4. Program the routing table (CRT) as required.
5. Prime the Logical Channel Buffer with data. Ideally the user would fill the entire buffer with valid transmit data. At a minimum, at least an empty word's count worth of data should be written to the logical channel buffer, or an entire asynchronous packet when the packet is smaller than this value.
6. Enable the appropriate logical channel enable (LCE) as required
7. At the next Start of Frame, the data in the transmit logical channel buffer starts to be consumed. An appropriate interrupt from the BISR is asserted depending on the number of free locations in the transmit buffer. If the user primed the buffer with a full 32 words, an interrupt is not generated until at least a word's count of data had been transmitted.
8. The Interrupt routine should then do the following:
    a. Clear the interrupt by writing the BICR.
    a. Write data to the appropriate transmit buffer. Table 6, page 20 indicates the address to use. All accesses must be word accesses. As many words should be written to satisfy the logical channel word count settings (see Table 4, page 19). If the C_EWC is set to '16,' then at least 16 words should be written. If the C_EWC is to '8,' then at least 8 words should be written (or until the end of an asynchronous packet).
    a. In the case of the asynchronous channel 0, the controller arbitrates on the ring to send the asynchronous packet after the first two words of the packet are available. The controller also automatically pads out the last frame in the packet, and triggers ATXASYNEOP to alert the application that the packet has exited the core.
9. A new interrupt is generated when the C_EWC threshold is reached again.

Note: If a bit in the MISR is asserted, and a second interrupt for that bit is triggered, the second interrupt will be hidden in the first. Failure to service interrupts when they are first triggered will result in a memory leak for that buffer.

Receiving Synchronous Data

To receive data on a synchronous logical channel, the following sequence occurs:

1. After reset, initialize the configuration and interrupt control registers as required
2. onEnable the core for Normal mode, and assert MENA in the SRR.
3. Allocate channels via sending allocation control messages.
4. Program the routing table (CRT) as required.
5. Enable the appropriate logical channel enable (LCE) as required. If the logical channel buffer was used to transmit prior to this new allocation it must be flushed
6. When the threshold of the logical channel buffer has been satisfied, the appropriate interrupt from the BISR is asserted
7. The Interrupt routine should then do the following:
   a. Clear the interrupt by writing the BICR.
   a. Read data from the appropriate receive buffer. Table 6, page 20 indicates the address to use. All accesses must be word accesses. As many words should be read to satisfy the logical channel word count settings (see Table 4, page 19). If the C_FWC is set to '16,' 16 words should be read, if set to '8,' 8 words.
8. A new interrupt is generated when the C_FWC threshold is reached again.

Receiving Asynchronous Data

The following sequence is used to receive packets on the asynchronous logical channel:

1. After reset, initialize the configuration and interrupt control registers as required
2. Enable the core for Normal mode, and assert MENA in the SRR
3. Program the routing table (CRT) as required
4. Enable the appropriate logical channel enable (LCE) as required
5. RXASYNEOP will be triggered when the end of packet has been reached. The appropriate interrupt from the BISR will be asserted when the threshold of the logical channel buffer has been satisfied
6. The Interrupt routine should then do the following:
   a. Clear the relevant interrupt by writing the BICR or MICR
   b. Read data from the appropriate receive buffer. The address to be used is indicated in Table 6, page 20. All accesses must be word accesses.
   c. The user will know the total length of the asynchronous receive packet from the length field in the first word of the packet. The user must keep a running count as to how many words have actually been received.
   d. If at least C_FWC words remain, then the user should read as many words as necessary to satisfy the logical channel word count settings (see Table 6, page 20). If the C_FWC is set to '16,' then 16 words should be read. If the C_FWC is to 8, then 8 words should be read.
   e. If less than C_FWC words remain, then the user should only read as many words as are necessary to satisfy the packet length, in addition to the read of the last (status) word.
7. A new interrupt will be generated when the C_FWC threshold or the end of packet is reached again. The threshold will be reset for every packet.

Flushing a Synchronous or Asynchronous Buffer

To Remove the Data in the Current Stream

The following sequence is used to flush a buffer when the data in the buffer is no longer desired:

1. Determine the current active channels by reading the logical channel enable (LCE) register.
2. Write the returned value back to the LCE by setting the bit for the desired logical to 0
3. Flush the appropriate logical channel buffers by writing the Flush Control Register (FCR) with the appropriate Logical Channel Number (FCHN), direction (FDIR) and the flush bit (FLCB). This will reset the pointers in the buffer and reset the word count values in that the channel. For asynchronous data in receive logical channel 0, the flush needs to occur after receiving the ASYNCEOP interrupt to ensure that the incoming packet is not partially flushed.
4. If it is a transmit logical channel, prime the buffer as described above in Transmitting Synchronous or Asynchronous Data
5. Determine the current active channels by reading the logical channel enable (LCE) register.
6. Write the returned value back to the LCE by setting the bit for the desired logical to '1'

Due to an Overflow or Underflow Condition

The following sequence is used to flush a buffer due to an underflow or overflow condition

1. After reset, initialize the configuration and interrupt control registers as required
1. Enable the core for Normal mode, and assert MENA in the SRR
1. Allocate channels via sending allocation control messages
1. Program the routing table (CRT) as required
1. Enable the appropriate logical channel enable (LCE) as required
1. Enable the RXBUFERR and TXBUFERR bits in the Most Interrupt Enable Register (MIER)
1. The appropriate interrupt from the BISR will be asserted when either an underflow or overflow condition has occurred
1. The Interrupt routine should then do the following:
    a. Read the TXBER for Transmit buffer issues, and the RXBER for Receive buffer issues. These registers will indicate which logical channel buffer overflowed or underflowed.
    b. Disable the appropriate logical channel by setting the corresponding bit in the Logical Channel Enable Register (LCE) to '0'.
    c. Flush the appropriate logical channel buffers by writing the Flush Control Register (FCR) with the appropriate Logical Channel number (FCHN), direction (FDIR) and the flush bit (FLCB). This will reset the pointers in the buffer and reset the corresponding bit in the TXBER or RXBER register
    d. Clear the interrupt by writing the MICR
2. As it is likely that the error condition may have been triggered by an error condition in the ring, it is recommended that the user deallocate all timeslots for the logical channel, and restart the sequence of configuring a logical channel.

Using the Streaming Ports

When using the streaming port interfaces, the following considerations occur:

1. First, the user must configure the logical channel for the streaming port
   a. If the data is being received from the MOST ring or transmitted to the MOST ring, follow the sequence to configure the logical channels as above. The interrupt procedure for the OPB interface does not need to be followed, because the OPB interface will never see the data transfers on this interface. However, any priming procedures must still be followed
   b. If the data is being received from the OPB interface or transmitted to the OPB interface, all that needs to be done is to enable the correct bit in the logical channel enable (LCE), because no data is seen on the ring. However, the interrupt procedure for data reception/transmission on the OPB interface must still be followed.
2. When the user asserts `OPB_Rst`, this is passed to the user as `STR_RST`.
3. When a buffer is flushed for any reason, this notification is passed along to the user as rising edge transition on `STR_<interface>_BIF_AVAIL`, where there is a specific bit for each logical channel. The core will not transfer any data when this signal is low.
4. For the TX and RX datapaths, the ingress and egress channels share common resources. The means that only the ingress or the egress may be active at any given time. For both datapaths, ingress has higher priority over egress, which ensures that neither interface will be starved when both are active.
5. When changing the logical channel on an egress port, one cycle of dead time where `<interface>_DST_RDY` is deasserted is required.
6. While the signaling for these interfaces is LocalLink, word count interrupts are provided to the user for use in determine full or empty status in the buffers. `<interface>_SRC_RDY` and `<interface>_DST_RDY` are deasserted when the core is unable to accept/transmit data. This determination is combined across logical channels, which means for example, that a full ingress buffer will block any ingress buffer.

Control Message Procedure

Transmit Message Procedure

After a controller has been configured, communication between the controllers is required to obtain information about configuration and to send/receive control messages to request allocation time slots, and so forth.

To add new Control Transmit Messages, the following sequence occurs:

1. Poll the Control Transmit Status Register (CTXS) occupancy bits (CTXOCC) until the occupancy is '0,' which indicates that there is room for a new transmit control message.
2. Write as many words as necessary in the order described above to the Control Transmit FIFO (CTXFIFO).
3. When all words have been written to the CTXFIFO, set the new message bit (CNEWTX) in the Control Transmit Status Register (CTXS). This will queue the transmit message for transmission on the MOST ring. The occupancy bit (CTXOCC) is set to '1' after the CNEWTX bit is written.
4. An interrupt (CTLTKFREE) is asserted after a message is de-queued from the Control Transmit FIFO. The CTXOCC is deasserted at the same time.
5. The user receives either a CTLTXOK interrupt (if successful) or a CTLTXFAIL interrupt (if not successful) to indicate the success of message transmission. Only one of these two interrupts is asserted for each transmit message. The core automatically attempts to retransmit messages according to the gap and retry programming set by the user in MRCDR.

6. On completion, the entire message is returned to the user, including the original request, response, and transmission status in the CTXRESFIFO

Note: No overflow notification or protection is provided for writes to a full Control Transmit Message FIFO. Writes to a full FIFO will corrupt the existing contents and may corrupt the state of the control channel in the MOST ring. This also implies that in case of back-to-back transmissions of control messages, the user must read the contents from the CTXRESFIFO within 2 frames of receiving the CTLTXOK interrupt for the first message or this content will be overwritten.

Response Message Procedure

To read a response for a Control Transmit Message, the following sequence occurs:

1. Either the CTLTXOK and CTLTXFAIL interrupt are asserted on completion of the transmit message and reception of any response and status. A CTLTXFAIL interrupt is only asserted after the message transmission has been retried subject to the value programmed in the MRCDR register.
2. Read seven words in the order defined above from the Transmit Response FIFO (CTXRESFIFO). A valid message has a value of 0x1010 in the TSTAT field, and the entire message is useful. A failed message has a TSTAT value set to the transmission status of the received message for the last transmit attempt (not equal to 0x1010), and the response portion may be missing.

Note: No underflow notification or protection is provided for reads to an empty Transmit Response FIFO Reads to an empty FIFO corrupt future transmit response reception.

Receive Message Procedure

To read new normal Control Receive Messages, the following sequence is used. Note that low-level messages, for example, allocation and deallocation, are directly handled by the core, and do not go to the user.

1. A CTLRXNEW interrupt is asserted on reception of each new receive control message. The CRXOCC is set to '1' at the same time. Wait for the interrupt indicating a new Control Message has been received.
2. Optionally, read the Control Receive Status Register (CRXS) occupancy bits (CRXOCC) to ensure the occupancy is not '0'. This indicates the number of received control messages to receive
3. Read six words in the order described above to the Control Receive FIFO (CRXFIFO)
4. When all words have been read from the CRXFIFO, the occupancy is automatically decremented. If another message was received in the time it took to read the first message and clear the interrupt, the occupancy automatically accounts for this. If the occupancy bit (CRXOCC) is still '1,' another message is available in the CRXFIFO. If CRXOCC is '0,' no more messages are available.
5. If the CRXFIFOs are not serviced by the user, the core will not acknowledge (NAK) messages it receives until room is made available.

Note: No underflow notification or protection is provided for reads to an empty Control Receive Message FIFO. Reads to an empty FIFO will corrupt future message reception.

Flushing a Control Buffer due to a Deadlock Condition

The following sequence is used to flush a control buffer. In general, flushing of control buffers is not recommended, however, if a deadlock state occurs where a very low priority message is not able to win arbitration, this method can be used to allow the high priority message to be transmitted.

1. The user must set a software timer to determine that this condition will be necessary
2. The user must then wait for an interrupt. A CTLTXOK interrupt will mean that the message was finally transmitted. A CTLTXFAIL will indicate that the retry count has been exceeded, and the message has been flushed automatically. Nothing more needs to be done for these first two

conditions. A CTLARBLST will indicate that this low priority message has failed again- the rest of the steps must be followed.

3. The user must flush the transmit control buffers by writing the Flush Control Register (FCR) with the appropriate Logical Channel Number (FCHN), direction (FDIR) and the flush bit (FLCB). This event must occur within a few frames of receiving the CTLARBLST to ensure that the ring is not corrupted.
4. All messages written to the control buffers will then be flushed. The user must reload the messages in the new order, with the high priority message first.

Controlling the State of Light on the Ring

Stopping the Transmission of Light on the Ring

The following sequence is used to ensure that there is no light on an inactive ring

1. At some point the Application layer will decide to stop transmitting data.
2. Ensure there is no more TX activity in the core- deallocate all channels, and cease writing to the transmit buffers (logical channel and control), which will minimize the number of active nodes in the device.
3. To force zero transmission at any time, the DAZ bit can be set in any mode with the exception of BYPASS.

Starting the Reception of Light from the Ring

The following sequence is used to enter into Normal mode from an inactive ring

1. At some point, the FOT will detect light, and this will trigger the LONCHG interrupt in the MISR.
1. The user can then use the correct sequence in configuring the core to regain synchronization.

Starting the Transmission of Light to the Ring

The following sequence is used to start traffic on an inactive ring

1. Force the core to master or a pseudo-master by setting MNSLV to '1,' BYPASS and LBACK to '0,' and RBDT to '0' for a Master/Slave core and to '1' for a Slave only core.
2. The core will start sending empty frames. Eventually, when the ring is complete, the FOT will detect light and this will trigger the LONCHG interrupt.
3. The user can then use the correct sequence in configuring the core to get back into normal mode for resynchronization.

Configuring the Controller for Test Modes

Ring Break Diagnostics

The procedure for Ring Break Testing is specified in the MOST Specification. The core is enabled to assist the user in this mode.

1. Ensure the core is reset by writing a '1' to the SRST bit. This will reset the core and force MENA to be negated. This can be read back via the SRR.
2. Write a '0' to the SRST bit, and a '0' to the MENA to take the core out of reset, but allow configuration of the core.
3. For a full Master/Slave core or a slave-only, program MNSLV to the desired setting for Ring Break. In addition, program RBDT to '1'. Program BYPASS and LBACK to '0.'
4. Enable the core by writing '1' to MENA and keeping SRST set to '0.'

5. Detect whether the ring is connected by looking for the LON interrupt. For a full Master/Slave core, complete reception is enabled. For a slave-only core, corrupt data will be received, but with the LON interrupt the user will know that there could be reception from that point.

For Loopback

For any asynchronous or synchronous data, a loopback configuration requires no special considerations. Any control message type with a single-address value can be sent to itself with the core responding appropriately to the message and triggering all of the correct interrupts in the system. Note that because the node acts as both upstream and downstream from itself, the sequence on the transmit side may not reflect exactly a regular transmission to another node. However, the stream of data received by the NIC will always look consistent. The following sequence should be followed after a reset (either hard or soft).

1. Ensure the core is reset by writing a '1' to the SRST bit. This will reset the core and force MENA to be negated. This can be read-back via the SRR.
2. Write a '0' to the SRST bit, and a '0' to the MENA to take the core out of reset, but allow configuration of the core.
3. Configure the controller for immediate transmission by writing '0' to each of BYPASS and RBDT, the desired mode into MNSLV, and '1' into LBACK. This must be done because the core will not be able to lock onto an incoming stream unless the core is also transmitting a stream. Note that the transmit output can be observed on the `MOST_TX` signal, which is routed internally in the core to the `MOST_RX` line. No external optical cables are required.
4. Program the PRCR based on the PLB clock frequency and PLL clock frequency.
5. Program the logical address in the LAR.
6. Program the alternate address in the AAR.
7. Program the group address in the GAR.
8. Program the message retry count in the MRCDR.
9. Confirm the status of the controller by reading the SR, which reports the current state of the core. To read-back what was programmed to the core, read MSR.
10. Enable appropriate interrupts by writing '1' to locations as needed in the MIER
11. Clear all MOST interrupts by writing 0xFFFF to the MICR
12. Enable the controller by writing a '0' to SRST and '1' to MENA. If the core is a configured as a master, operation will be similar. If the core is configured as a slave, the core will operate in pseudo-master mode, which will initially send empty frames in order to provide an incoming data stream for synchronization. Master-specific behavior, other than preamble generation and field resetting to all zeroes, will not be available in the data stream of a pseudo-master.
13. Via interrupts or polling confirm full synchronization is achieved by reading the SR. The SBLCK, BLCK and FLCK bits will indicate the synchronization level.
14. Read the value for the synchronous boundary descriptor in the CCR.
15. Read the value for the delay and position registers in the MCPDR.
16. The controller is now ready to send and transmit control, synchronous and asynchronous data.

Special Considerations

Disabling MENA for an Active Core

While the user may change the MENA bit at any time, deasserting MENA while the core is in active mode does have some ramifications for core operation that are listed below:

1. All buffers will be flushed. This includes the TX and RX buffers for synchronous and asynchronous data, as well as both directions of control buffers.
2. The core will automatically be forced into a bypass state, such that the programming does not corrupt the ring. Once MENA is reasserted, the original programming will be reactivated. If this needs to be checked, the user can read the MSR for the programmed state of the core, and the SR register for the actual state of the core.
3. The synchronization state machines will proceed as normal.
4. All programmed configuration will stay active. This includes such registers as the address programming and retry configuration as well as the routing table. If the state of these registers needs to be 'reset,' the user will need to do this.
5. The streaming port interface will become disabled.
6. Disabling MENA also resets the External PLL reset count (EPRC) register.

SBD Change

A change in the Synchronous Boundary Descriptor has multiple effects in the core. SBD changes happen when the value is changed in the CCR register as a Master, or when an SBDCHG interrupt is detected as a Slave. The following steps should be handled by the user application to ensure that this event does not corrupt the ring.

1. If the MOST NIC is configured as a Master Node, the user should send a deallocate all message to itself. This will ensure that the allocation table stays within the bounds of the current descriptor. MAT registers will be out of date until this occurs, and SAR registers will be out of date until updated by the ring.
2. All nodes should reallocate their synchronous data transmission channels.
3. The user should flush the asynchronous buffer (LC 0), because some messages may be in transmit, and an SBD event will stop the transmission. Any pending messages should be retransmitted.
4. The user should reprogram the CRT to take into affect the new synchronous channel timeslots.

Multicast Addressing for Control Messages

While multiple addressing modes of broadcast and groupcast are supported by the NIC, consider the following:

1. Only message types of Normal, Remote Write, and Get Source are supported. If the NIC receives a message with a multicast address that does not match one of these types, it is dropped silently.
2. The NIC never responds to a multicast address for a message it has sent. For this reason, in test modes such as loopback, these messages never result in a successful transmission.

Using the Core in Loopback

When using the core in loopback mode, the user software processes both the receive and transmit interrupts at the same time. If the MOST NIC is used in a design where the system clock (`OPB_Clk`) is slow, there may be difficultly servicing the normal rate interrupts.

1. For asynchronous packets, the lower the SBD value, the more asynchronous data valid in any given frame. In this condition, the user may want to keep the length of the packet down to a value close to the depth of the buffer to prevent underflow/overflow conditions in the buffers when the user routine cannot service both the RX and TX data paths at the same rate.
2. For synchronous data, the user should not try to enable all of the timeslots, particularly on the same logical channel.
3. For control data, there are no limitations. When the RX buffers are filled to capacity, the core automatically retransmits failing TX packets. However, because the node acts as both upstream

and downstream to itself, the sequence on the transmit side may not reflect the exact transmission sequence to a different node on the ring. Even so, the stream of data received by the MOST NIC will always look consistent.

Standalone Core Generation using XPS tool

GUI Mode

1. Open XPS tool and select a blank project.
2. Drag and drop the XPS MOST v1.01a from IP Catalog. Double click the core to open GUI and select required core settings.
3. Select Hardware->Generate Netlist.

Command Line Mode

1. Select the required core settings in the MHS file (system.mhs).
2. In the command window, type

```
platgen -p <device_name> -lang vhdl system.mhs
```

Ex: `platgen -p xc3s1600efg320-5 -lang vhdl system.mhs`

In both the above modes of operation,

- The netlist will be generated in implementation directory with the name xps_most_nic_0_wrapper.ngc.
- The wrapper file will be generated in hdl directory with the name xps_most_nic_0_wrapper.vhd.

Design Constraints

Timing Constraints

The core can have up to three clock domains:

- `SPLB_Clk` for the main system clock for the core
- `MOST_COM_CLK` and `MOST_PLL_CLK` for the MAC transmitter and receiver

These clock nets and the signals within the core that cross these clock domains must be constrained appropriately in a UCF.

PERIOD Constraints for Clock Nets

SPLB_Clk

The clock provided to `SPLB_Clk` must be constrained to the appropriate frequency. The frequency allowed will vary across devices, but will be at least 75 MHz for all implementations. In addition, `SPLB_Clk` must be at least 7/10 of the frequency of `MOST_PLL_CLK`.

The following UCF syntax shows the necessary constraints being applied to the `SPLB_Clk` signal for a 50 MHz clock:

Set the system clock period constraints

```
NET "SPLB_Clk" TNM_NET = “SPLB_Clk”;
TIMESPEC “TS_SPLB_clock” = PERIOD "SPLB_clock" 20000 ps HIGH 50 %;
```

MOST_COM_CLK and MOST_PLL_CLK

The clock provided to MOST_*_CLK must be constrained to the desired management interface operating frequency, where the two clocks are frequency locked to one another. In addition, there are requirements set on the two clocks depending on the mode of the core, that are fully depicted in the clock wrapper file provided in the example design. For a full master/slave core, there is a BUFGMUX to select the correct source for the received data. For a slave-only core, the two clocks are connected.

Note: For normal slave operation, the clock should be tied to the recovery clock. For slave operation in loopback mode, the clock should be tied to the crystal clock. In order to switch between normal and loopback modes, the master/slave circuit can be used to change these sources while in operation.

The following UCF syntax shows a 45.158 MHz (CD audio rate) period constraint being applied to the MOST_PLL_CLK and MOST_COM_CLK:

Set the MOST PLL clock period constraints

```
NET "MOST_PLL_CLK" TNM_NET = "MOST_PLL_CLK";
TIMESPEC "TS_MOST_PLL_CLOCK" = PERIOD "MOST_PLL_CLOCK" 22144.274 ps HIGH
50 %;
```

Set the MOST COM clock period constraints

```
NET "MOST_COM_CLK" TNM_NET = "MOST_COM_CLK";
TIMESPEC "TS_MOST_COM_CLOCK" = PERIOD "MOST_COM_CLOCK" 22144.274 ps HIGH
50 %;
```

The following constraint is recommended to be added in the UCF for constraining the datapath delay between SPLB_Clk and MOST_COM_CLK:

Set the constraint for datapath delay between MOST COM clock and SPLB_Clk

TIMESPEC "TS_MOST_CCLK_SPLB" = FROM "MOST_COM_CLK" TO "SPLB_Clk" 10 ns DATAPATHONLY;

```
TIMESPEC "TS_SPLB_MOST_CCLK" = FROM "SPLB_Clk"   TO "MOST_COM_CLK" 10 ns
DATAPATHONLY;
```

The following offset constraints for the Tx and Rx paths are recommended to be specified in the UCF file:

NET "*most_*MOST_RX*" TNM = PADS:RX_PATH;

NET "*most_*MOST_TX*" TNM = PADS:TX_PATH;

NET "*most_*MOST_?X*" TNM = PADS:MOST_PATH;

TIMEGRP "RX_PATH" OFFSET = IN 6 ns before "MOST_PLL_CLK";

TIMEGRP "TX_PATH" OFFSET = OUT 22 ns after "MOST_COM_CLK";

TIMESPEC "TS_BYPASS" = FROM "RX_PATH" TO "TX_PATH" 15 ns;

Design Implementation

Target Technology

The target technology is an FPGA listed in the Supported Device Family field of the LogiCORE IP Facts table. The device used must have the following attributes:

- Large enough to accommodate the core.
- Contains a sufficient number of IOBs.

Device Utilization and Performance Benchmarks

Because the xps_most_nic core will be used with other design modules in the FPGA, the utilization and timing numbers reported in this section are estimates only. When the XPS MOST NIC core is combined with other designs in the system, the utilization of FPGA resources and timing of the XPS MOST NIC design will vary from the results reported here. The XPS MOST NIC resource utilization for various parameter combinations measured with Spartan-3 as the target device are detailed in the following table.

Table 72: **Performance and Resource Utilization Benchmarks on Spartan-3 (xc3s1500-fg456-4)**

Parameter Values (Other parameters at default values)			Device Resources				F_{MAX} (MHz)
C_OPMODE	C_FWC	C_EWC	Slices	Slice Flip-Flops	4-input LUTs	BRAMs	PLB F_{MAX}
0	16	16	3137	2644	4617	6	83.633
0	16	8	3136	2644	4613	6	80.879
0	8	16	3136	2644	4613	6	83.94
0	8	8	3135	2644	4608	6	85.10
1	16	16	3033	2595	4431	6	87.596
1	16	8	3032	2595	4427	6	85.091
1	8	16	3031	2595	4427	6	82.095
1	8	8	3030	2595	4422	6	86.437

Specification Exceptions

N/A

Reference Documents

1. MOST Specification revisions 2.4 and 2.5
2. IBM 128-bit Processor Local Bus Architecture Specifications version 4.6
3. LocalLink Specification (To access the LocalLink specification, the user must register on the LocalLink product page.)
   http://www.xilinx.com/xlnx/xebiz/designResources/ip_product_details.jsp?iLanguageID=1&sSecondaryNavPick=&key=LocalLink_UserInterface

Support

Xilinx provides technical support for this LogiCORE product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled *DO NOT MODIFY*.

Ordering Information

This Xilinx LogiCORE IP module is provided under the terms of the Xilinx Core Site License. The core is generated using the Xilinx ISE Embedded Edition software (EDK). For full access to all core functionality in simulation and in hardware, you must purchase a license for the core. Please contact your local Xilinx sales representative for information on pricing and availability of Xilinx LogiCORE IP.

For more information, please visit the XPS MOST NIC product web page.

Information about this and other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page. For information on pricing and availability of other Xilinx LogiCORE modules and software, please contact your local Xilinx sales representative.

This Embedded IP module is provided under the terms of the Xilinx Core Site License. A free evaluation version of the core is included with the ISE Design Suite Embedded Edition software. Use the Xilinx Platform Studio application (XPS) included with the Embedded Edition software to instantiate and use this core.

For full access to all core functionality in simulation and in hardware, you must purchase a license for the core. Please contact your local Xilinx sales representative for information on pricing and availability of Xilinx LogiCORE IP.

Revision History

Date	Version	Description of Revisions
10/30/07	1.0	Initial Xilinx release.
4/18/08	1.0.1	In LogiCORE table, changed XA designator to Automotive Spartan-3.
6/17/08	1.0.2	Updated the data sheet for xps_most_nic_v1_00_b changes, added 2 new registers (PRCR and EPRC), updated the programming sequence, added slice registers, BRAMs in Resource Utilization Table, corrected number of Slices which was indicating number of Slice registers in xps_most_nic_v1_00_a datasheet.
7/25/08	1.1	Added QPro Virtex®-4 Hi-Rel and QPro Virtex-4 Rad Tolerant support.
7/28/08	1.1.1	Updated description for PRCR.
7/30/08	1.1.2	Added offset constraints to Design Constraints section.
04/24/09	1.2	Added 'Programming the Core' and 'Standalone Core Generation' sections; replaced references to supported device families and tool name(s) with hyperlink to PDF file.
4/19/10	1.3	Updated to 12.1; listed supported devices families in LogiCORE Table; updated images to graphic standard; converted to new DS template; added ordering information.

Notice of Disclaimer