- DISCONTINUED PRODUCT -

Generic Framing Procedure v2.1

DS303 April 25, 2008 **Product Specification**

Introduction

The LogiCORE™ IP Generic Framing Procedure (GFP) core is a fully verified protocol encapsulation/de-encapsulation engine enabling efficient transport of LAN/SAN client protocols over SONET/SDH-based networks.

Features

- Fully implements the *ITU-T G.7041 GFP Specification*
- Supports frame-mapped, transparent, and mixed mode operation
- Supports both null and linear frames, enabling 1 to 256 channels
- Operates at up to OC-48 (32-bit interface) or OC-192 (64-bit interface) data rates
- Supports both client data frames and client management frames
- Optional host interface for access to control and status registers in circuit
- Offers standardized user interfaces
  - LocalLink interfaces for transfer of data
  - Device Control Register (DCR) host interface
- Fully configurable using the Xilinx® CORE Generator™ tool
- Available under terms of the SignOnce™ IP Site License

LogiCORE IP Facts					
Core Specifics					
Supported Device Families	Virtex®-5,Virtex-4 Virtex-II Pro, Virtex-II, Spartan®-3, Spartan-3E				
Virtex-5 Devices					
Resources Used	**I/O[1]**	**LUTs**	**FFs**	**Block RAMs**	**Slices[2]**
MAP Core (32-bit)	N/A	1828-4368	1361-3673	0-1	467-1810
MAP Core (64-bit)	N/A	3518-6147	2242-4229	0-1	946-2237
UNMAP Core (32-bit)	N/A	1102-3828	989-3589	0-9	556-1632
UNMAP Core (64-bit)	N/A	1747-6725	1838-5443	0-15	639-2593
Virtex-4, Virtex-II Pro, Virtex-II, Spartan-3, and Spartan-3E Devices					
Resources Used	**I/O[1]**	**LUTs**	**FFs**	**Block RAMs**	**Slices[2]**
MAP Core (32-bit)	N/A	1352-5060	1076-4100	0-1	841-2781
MAP Core (64-bit)	N/A	2308-7528	1842-4647	0-1	1624-4234
UNMAP Core (32-bit)	N/A	447-4902	515-4098	0-9	400-2931
UNMAP Core (64-bit)	N/A	3383-9331	1840-6358	0-15	2241-5379
Provided with Core					
Documentation	Product Specification • Getting Started Guide • User Guide •				
Design File Formats	VHDL, Verilog				
Constraints File	UCF				
Verification	VHDL, Verilog Test Bench				
Instantiation Template	VHDL, Verilog				
Design Tool Requirements					
Xilinx Implementation Tools	ISE® 10.1				
Simulation	Mentor Graphics® ModelSim® v6.3c				
Synthesis	XST ISE, Synplicity® Synplify®				
Support					
Provided by Xilinx, Inc. @ www.xilinx.com/support.					

1. All I/O internal to the FPGA.
2. Slice counts obtained with area groups placed on each core.

Applications

The GFP core can be used in many applications to enable efficient transport of LAN/SAN client protocols over SONET/SDH networks. The following sections describe common applications using additional Xilinx IP cores and reference designs.

Frame-Mapped Mode (Ethernet)

Figure 1 displays a typical application of transmitting Ethernet over SONET/SDH. This example uses the Xilinx 1000BASE-X PCS/PMA and Xilinx 1-Gigabit Ethernet MAC for the client interface. The user logic interfaces between the MAC and the GFP core, accumulating packets of data on a per-channel basis. The GFP core is configured in frame-mapped mode, with linear extension headers enabled. This allows the GFP core to time-multiplex many channels into a single GFP stream. The GFP core interfaces to the Xilinx SPI-4.2 core, which connects to an external framer.

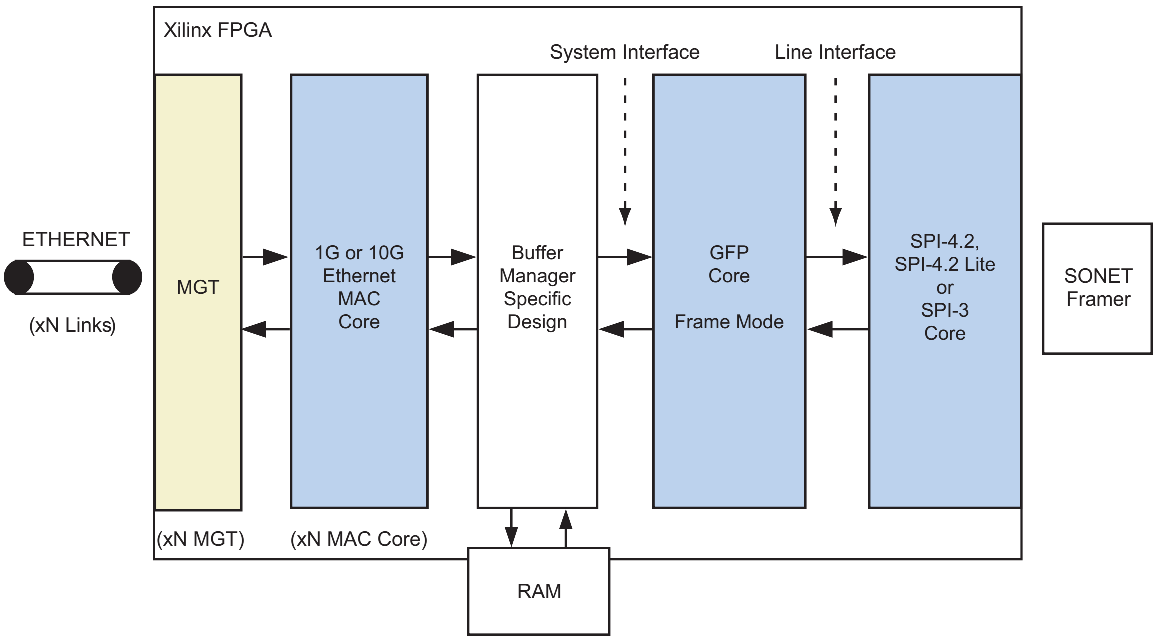

Figure 1: **Transmitting Ethernet over SONET/SDH**

Transparent Mode (Fibre Channel)

Figure 2 illustrates a typical application of transmitting Fibre Channel over SONET/SDH. This example uses XAPP759, *Configurable Physical Coding Sublayer* (CPCS) for the client interface. The CPCS can be configured to support a variety of protocols, including Fibre Channel, ESCON, FICON, and 1-Gigabit Ethernet. In this case, the CPCS design is configured for Fibre Channel. The user logic interfaces between the CPCS and the GFP core, performing Fibre channel flow control (spoofing), as well as rate adaptation and buffering on a per-channel basis.

The GFP core is configured in transparent mode, with linear extension headers enabled, allowing the GFP core to time-multiplex many channels into a single GFP stream. The GFP core interfaces to the Xilinx SPI-4.2 core, which connects to an external framer.

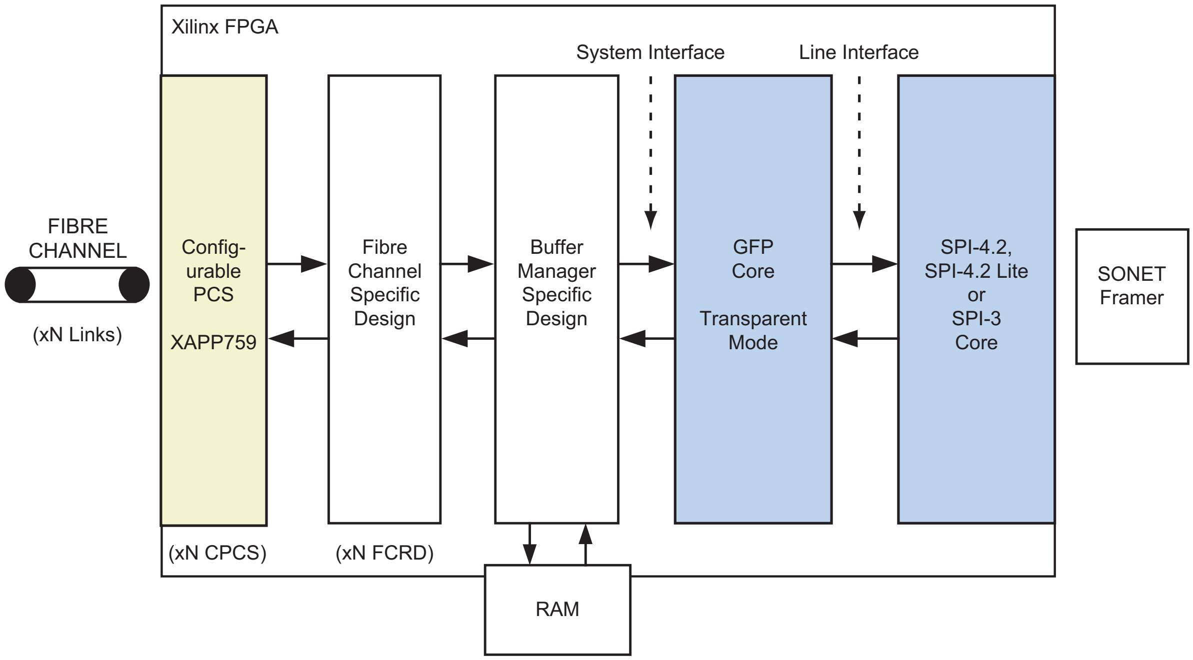

Figure 2: **Transmitting Fibre Channel over SONET/SDH**

Feature Summary

Operating Modes

The GFP core can be configured for frame-mapped, transparent, or mixed mode operation. When both frame-mapped and transparent operations are required, the core statically supports transmitting either frame-mapped or transparent frames, and can switch between modes *in situ* using the host interface. When only frame-mapped or transparent functionality is required, the unused logic is automatically removed for optimal resource utilization.

Frame-mapped

When configured for frame-mapped mode, the GFP core supports all frame-mapped protocols including PPP, Ethernet, and RPR as defined in the *ITU-T GFP Specification*. The MAC functionality of removing the appropriate headers, such as flag and escape characters for PPP or preamble and start-of-frame for Ethernet, must be performed outside of the GFP core. The core accepts complete, contiguous frames of data for encapsulation and de-encapsulation. The frame-mapped core also handles all corrupted frames by correctly terminating the frames according to the *ITU-T GFP Specification*.

Transparent

When configured for transparent mode, the GFP core supports all transparent protocols including Fibre Channel, ESCON, FICON, Gigabit Ethernet, and OVB ASI, as defined in the *ITU-T G.7041 GFP Specification.* It accepts 8b data as input (8b/10b encoding and decoding must be performed outside of the GFP core). The data is encoded/decoded into 64b/65b block codes as specified in the *ITU-T G.7041 GFP Specification*. Illegal 8b/10b control words are mapped to a special 10B_ERR control character in the 64b/65 block code. The GFP core also supports the insertion and removal of special 65B_PAD characters to allow for rate adaptation. The core generates superblocks (composed of eight 64b/65b block codes) with a CRC-16.

Mixed Mode

Mixed mode enables the GFP core to process both frame-mapped and transparent mode frames, and supports all protocols defined in the ITU-T GFP Specification. Using either the CORE Generator or the host interface, the core can be configured to support both modes of frame transmission simultaneously, on a per-channel basis. If the host interface is enabled, the core configuration can also be changed in circuit.

Support for Multiple Channels

The GFP core supports up to 256 channels using the channel ID (CID) field in the linear extension header. Through the CORE Generator graphical user interface (GUI), up to 10 channels can be configured for different protocols, and channel-specific register settings (see <RD Red>"MAP Configuration Space" on page 11 for detailed information). Additional channels beyond 10 (up to 256) are supported and use the same configuration as channel 0.

Operates at OC-48 and OC-192 Rates

The GFP core supports OC-48 (2.5 Gbps) and OC-192 (10 Gbps) data rates. For a 32-bit data bus at OC-48, the internal logic runs at up to 100 MHz. For a 64-bit data bus at OC-192, the internal logic runs at approximately 156 Mhz.

Frame Types

The GFP core supports two client frame types: client data frames and client management frames. Client data frames contain user payload data; client management frames indicate client signal-fail information. Idle frames are automatically generated by the core when the input bandwidth is insufficient to fill the bandwidth requirements of the output transport network.

Host Interface

The GFP core provides a host interface for access to control and status registers. Inclusion of this interface is optional and allows configuration of the core in situ. To minimize the resource utilization of the core, the user can choose to generate a GFP core without a host interface.

Support for Corrupted Frames

Errors can be optionally injected into the MAP core for any CRC calculation as well as any scrambling function. An interrupt mechanism (and interrupt mask) is used in each core to indicate when an unexpected event occurs. All these configuration options are controlled through the host interface.

Overview

The GFP core is comprised of two separate cores that enable the transmission and reception of data to or from a SONET/SDH network:

- The MAP core receives client network protocol data via the system interface, encapsulates the data with the GFP protocol, and transmits the GFP encapsulated frame via the line interface.
- The UNMAP core receives GFP encapsulated data via the line interface and de-maps the GFP frames to extract client network protocol data to be transmitted via the system interface.

Each core is comprised of three interfaces:

- **System interface**. Transmits data between the user's client interface and the GFP core. The data on the system interface is the client network protocol data, such as Ethernet or Fibre Channel.
- **Line interface**. Transmits data between the GFP core and the user's line interface (SONET/SDH). The data on the line interface is GFP encapsulated data, such as frame-mapped Ethernet, or transparent mode Fibre Channel.
- **Host interface**. Provides access to the GFP core status and control registers *in situ*.

Figure 3 illustrates the MAP and UNMAP cores.

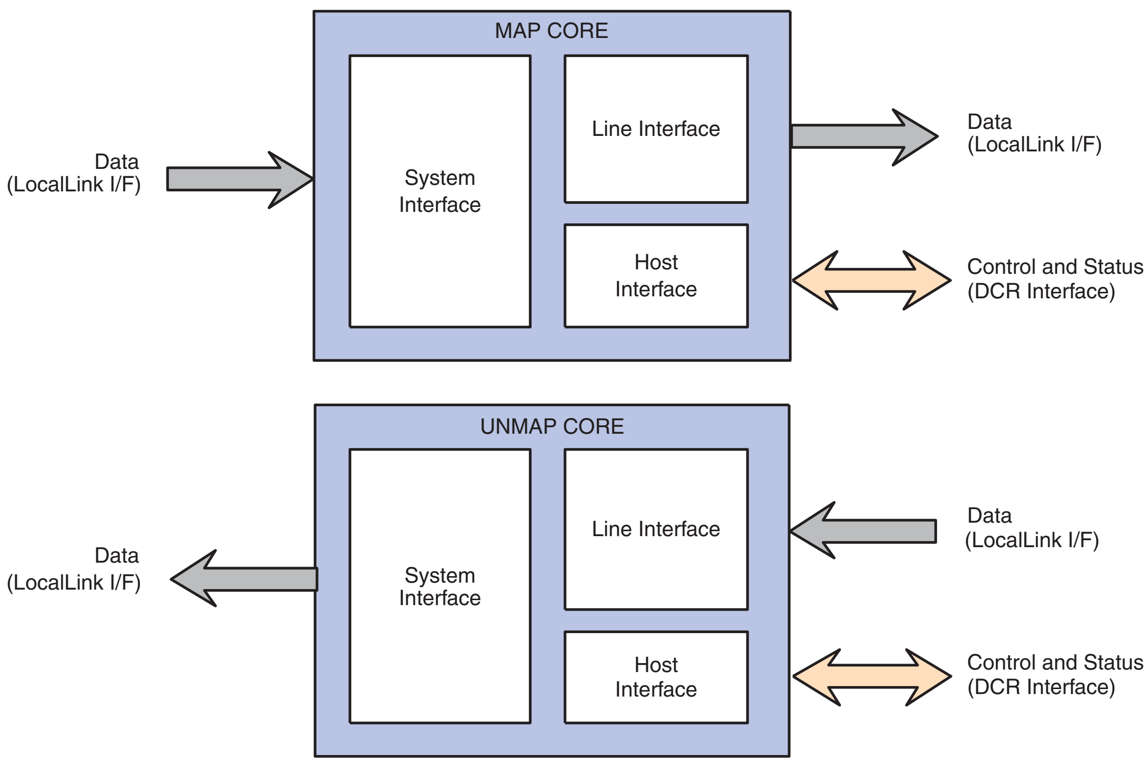

Figure 3: **GFP Core Block Diagram**

Core Interfaces

This section describes the interface signals of the GFP core. The MAP and UNMAP cores utilize the following top-level signal interfaces:

- Common interface
- System interface
- Line interface
- Host interface

Note that a signal ending in _N is active low; otherwise, the signal is active high. All signals apply to both frame-mapped and transparent mode unless otherwise noted.

MAP Core Interfaces

Figure 4 displays the MAP core interfaces. All signals are defined in their respective sections following the illustration.

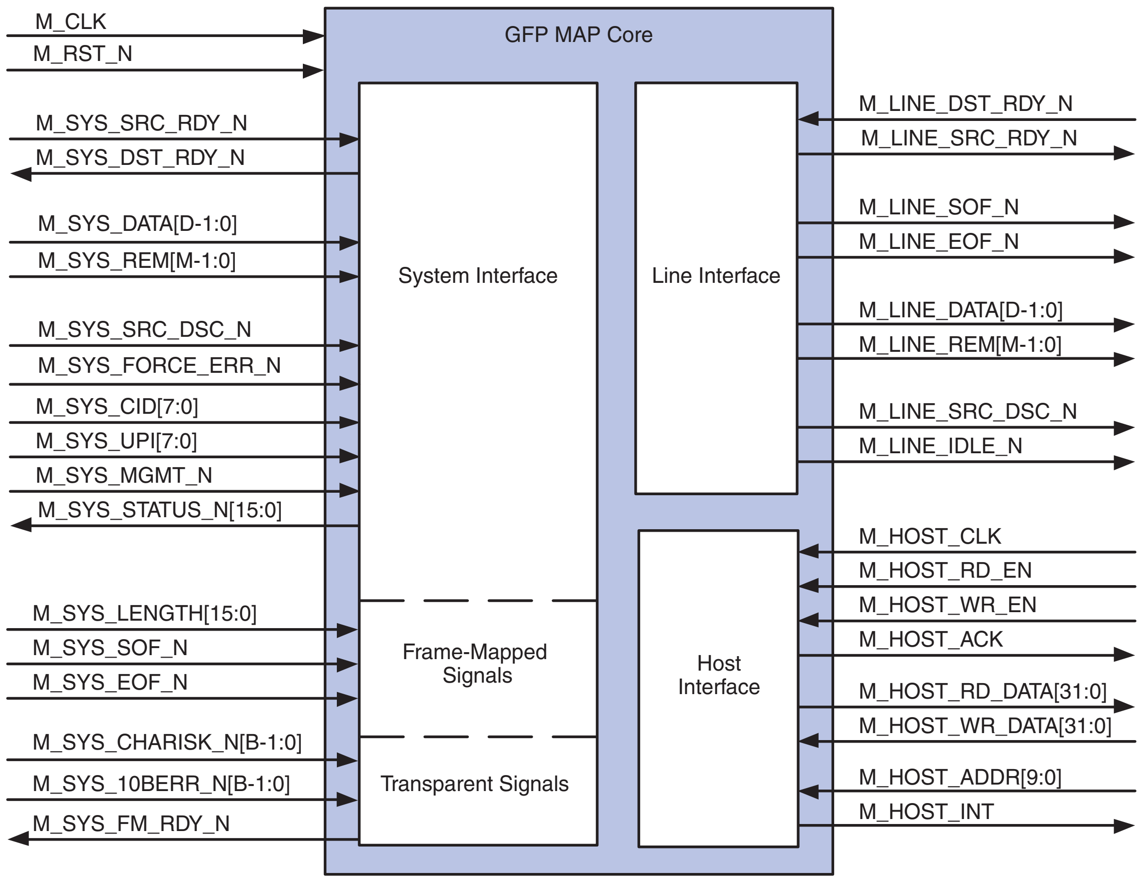

Figure 4: **MAP Core Interfaces**

Table 1 describes the relationship between the data bus width and additional signals. The MAP core supports both a 32-bit and a 64-bit interface. The width of many bus signals depends on the interface width selected (32 or 64 bits).

Table 1: **MAP Core Bus Widths**

Data Bus Width (D) (*_DATA)	Remainder Width (M) (*_REM)	Data Byte Width (B) (*_10BERR_N, *_CHARISK_N)
32	2	4
64	3	8

Common Interface

Table 2 defines the signals common to the entire MAP core. The MAP reset signal (M_RST_N) causes a hard reset of the entire core (core logic and host interface). This signal is an asynchronous input which is synchronized internally in the core before being used. The initial hardware reset should be generated by the user. Subsequent resets may be asserted by using the M_RST_N pin (for a complete core reset), or by driving the specific reset register in the MAP host interface (for core logic reset only).

Table 2: **MAP Common Interface**

Name	Direction	Description
M_CLK	Input	**MAP Clock**: All system and line interface operations are synchronous to this clock.
M_RST_N	Input	**MAP Reset**: Asynchronous reset that resets both the MAP core logic and MAP host interface.

System Interface (M_SYS)

Table 3 describes the MAP system interface signals (M_SYS). The MAP system interface connects to the client side of the system and implements the Xilinx LocalLink standard, providing a simple, flexible way to receive frames. The system interface consists of a unidirectional data bus with control signals that allow the user application to stall or discontinue data. The system interface signals are divided into three categories: signals common to both frame-mapped and transparent mode, signals specific to frame-mapped mode, and signals specific to transparent mode, as shown in Figure 4.

Table 3: **MAP System Interface**

Name	Direction	Description
Common to frame-mapped and transparent modes		
M_SYS_SRC_RDY_N	Input	**Write Enable** (Source Ready): Indicates a word presented by the client is valid (not accepted until M_SYS_DST_RDY_N is also asserted).
M_SYS_DST_RDY_N	Output	**Write Accepted** (Destination Ready): Indicates a word presented by the client will be accepted (if M_SYS_SRC_RDY_N is also asserted).
M_SYS_DATA[D-1:0]	Input	**Data Bus**: Client-side data to be encapsulated into a GFP frame. If Ethernet is being transferred (frame-mapped mode), then the first word sent should be the destination address. In transparent mode, the data will be raw 8b data.

Table 3: **MAP System Interface** ***(Cont'd)***

Name	Direction	Description
M_SYS_REM[M-1:0]	Input	**Data Remainder**: The number of valid bytes in M_SYS_DATA is M_SYS_REM + 1, and is MSB justified. In frame-mapped mode, this is only valid when M_SYS_EOF_N is asserted. In transparent mode, this can be asserted at any time (core will insert 65B_PAD words to fill gaps if necessary). Example: 32-bit data bus: REM = "00" => DATA[31:24] valid REM = "01" => DATA[31:16] valid REM = "10" => DATA[31:8] valid REM = "11" => DATA[31:0] valid
M_SYS_SRC_DSC_N	Input	**Discontinue Frame**: Causes the frame currently in transit to terminate (once the specified length is reached) and invert its FCS. In frame-mapped mode, this signal is only valid when M_SYS_EOF_N is asserted.
M_SYS_FORCE_ERR_N	Input	**Force Error**: Inserts errors as specified by the MAP host register GFP_ERR. This signal should be asserted for the entire frame in which an error is to be inserted.
M_SYS_CID[7:0]	Input	**Channel Identifier**: Indicates the channel ID for linear frames. This signal is only present if linear extension headers are enabled.
M_SYS_UPI[7:0]	Input	**User Payload Identifier**: Indicates the payload type for the current data frame. When M_SYS_MGMT_N is asserted, this signal indicates the type of management frame to send.
M_SYS_MGMT_N	Input	**Management Frame Indicator**: Generates a client management frame to be transmitted on the line interface. When asserted, M_SYS_UPI will be used to indicate the type of client signal fail (CSF).
M_SYS_STATUS_N[15:0]	Output	**Status Bus**: Provides real-time status and errors. [15:6] = reserved [5] = Loss of client signal management frame transmitted due to CSF timer expiration [4] = Loss of character synchronization management frame transmitted due to CSF timer expiration [3:2] = reserved [1] = Invalid K character written in the system interface (mapped to 10B_ERR) [0] = PLI mismatch
Frame-mapped mode only		
M_SYS_LENGTH[15:0]	Input	**Payload Length**: Indicates the length of the current frame in bytes. This is only active when M_SYS_SOF_N is asserted.
M_SYS_SOF_N	Input	**Start of Frame**: Indicates the beginning of either a data or management frame.
M_SYS_EOF_N	Input	**End of Frame**: Indicates the end of either a data or management frame.

Table 3: **MAP System Interface *(Cont'd)***

Name	Direction	Description
Transparent mode only		
M_SYS_CHARISK_N[B-1:0]	Input	**K Character Indicator**: Indicates that the corresponding byte of M_SYS_DATA is a 8b/10b K character, not a data character. This signal is ignored for a given byte lane if M_SYS_REM indicates that the byte is not valid.
M_SYS_10BERR_N[B-1:0]	Input	**Insert 10B_ERR**: Indicates that the corresponding byte of M_SYS_DATA should be replaced with the 10B_ERR code. This signal is ignored for a given byte lane if M_SYS_REM indicates that the byte is not valid. This signal overrides M_SYS_CHARISK_N if both signals are simultaneously asserted.
M_SYS_FM_RDY_N	Output	**Frame Ready**: Look-ahead signal indicating that the next transparent frame will begin in no fewer than 4 cycles (in 32-bit) or 2 cycles (in 64-bit). See the *GFP User Guide* for details.

Line Interface (M_LINE)

Table 4 describes the MAP line interface signals (`M_LINE`). The MAP line interface utilizes the Xilinx LocalLink interface for a simple yet flexible medium for transmitting frames. It consists of a unidirectional data bus with additional control signals.

Table 4: **MAP Line Interface**

Name	Direction	Description
M_LINE_DST_RDY_N	Input	**Read Enable** (Destination Ready): Indicates a word is accepted this cycle (not accepted until M_LINE_SRC_RDY_N is also asserted).
M_LINE_SRC_RDY_N	Output	**Read Valid** (Source Ready): Indicates a word presented by the line interface is valid (not read until M_LINE_DST_RDY_N is also asserted).
M_LINE_SOF_N	Output	**Start of Frame**: Indicates the beginning of a frame.
M_LINE_EOF_N	Output	**End of Frame**: Indicates the end of a frame.
M_LINE_DATA[D-1:0]	Output	**Data Bus**: GFP encapsulated data transmitted on the line interface.
M_LINE_REM[M-1:0]	Output	**Data Remainder**: The number of valid bytes in M_LINE_DATA is M_LINE_REM + 1, and is MSB justified. This signal is only valid when M_LINE_EOF_N is asserted. See M_SYS_REM for valid byte mappings.
M_LINE_SRC_DSC_N	Output	**Frame Discontinue**: Indicates the current GFP frame contains an error.
M_LINE_IDLE_N	Output	**Idle Indicator**: Indicates the current GFP frame is an idle frame, and can be dropped by the user if necessary. This signal is asserted for the entire idle frame.

Host Interface (M_HOST)

Table 5 describes the Map host interface signals (`M_HOST`). The MAP host interface to the user application utilizes the DCR bus for direct connection to the PowerPC™, Microblaze™, or other microprocessors. It consists of separate read and write data buses with a shared address bus. Note that the host interface is optional, and may be removed to conserve resources if *in situ* access to control registers is not required.

Table 5: **MAP Host Interface**

Name	Direction	Description
M_HOST_CLK	Input	**Host Clock**: All host interface signals are synchronous to this clock. Note that this signal is optional, as the host interface can be configured to be synchronous to M_CLK.
M_HOST_RD_EN	Input	**Read Enable**: Indicates a read access to the register addressed on M_HOST_ADDR.
M_HOST_WR_EN	Input	**Write Enable**: Indicates a write access to the register addressed on M_HOST_ADDR.
M_HOST_ACK	Output	**Acknowledge**: Both read and write requests to the host interface are acknowledged through this signal. Following a read request, indicates that the data on M_HOST_RD_DATA is valid. Following a write request, indicates that the data on M_HOST_WR_DATA was accepted.
M_HOST_RD_DATA[31:0]	Output	**Read Data**: Data read from the address M_HOST_ADDR, which is valid when a read request (M_HOST_RD_EN) followed by an acknowledge (M_HOST_ACK) is asserted.
M_HOST_WR_DATA[31:0]	Input	**Write Data**: Data to be written into the register specified by the address M_HOST_ADDR. The write is acknowledged on M_HOST_ACK when the write succeeded.
M_HOST_ADDR[9:0]	Input	**Register Address**: Bus used to specify the address being accessed either for a read or write.
M_HOST_INT	Output	**Interrupt**: Indicates that an interrupt condition has been detected, and will continue to be driven until the interrupt is cleared (by writing to MAP GFP_INT).

MAP Configuration Space

The MAP core supports a host interface for access to control and status registers *in situ*. The MAP core provides both general and channel-specific registers.

- <RD Red>Table 6 describes the MAP core general registers.
- <RD Red>Table 7 describes the MAP core channel-specific registers, which enable the user to customize specific parameters on a per-channel basis.

Table 6: **MAP Core General Registers**

Register Name	Type	Description	DCR Offset
GFP_VERSION	R/W	[31] = Core reset	0x0
	R	[30:0] = Core version number	
GFP_CTRL		[31:18] = Reserved	0x1
	R/W	[17] = Disable scrambling of header	
	R/W	[16] = Disable scrambling of payload	
		[15:4] = Reserved	
	R/W	[3:0] = Upper 4 bits of Client Signal Fail (CSF) counter	
GFP_ERR		[31:8] = Reserved	0x2
	R/W	[7] = Payload scrambling error insertion	
	R/W	[6] = Location of superblock CRC error - (1) insert CRC error in all superblocks - (0) insert CRC error in first superblock	
	R/W	[5] = Superblock CRC error insertion	
	R/W	[4] = FCS error insertion	
	R/W	[3] = Core header scrambling error insertion	
	R/W	[2] = cHEC error insertion	
	R/W	[1] = tHEC error insertion	
	R/W	[0] = eHEC error insertion	
GFP_INTMASK		[31:2] = Reserved	0x3
	R/W	[1] = Enable interrupt for invalid K characters received	
	R/W	[0] = Enable interrupt for PLI length mismatch	
GFP_INT[1]		[31:2] = Reserved	0x4
	R/W	[1] = Invalid K character received	
	R/W	[0] = PLI length mismatch	

1. All interrupts are cleared by performing a write operations to this register.

The MAP core also contains channel-specific registers, which enables the core to customize specific parameters on a per-channel basis (see Table 7). Note that the channel notation refers to channel-specific configurations, where the number of channels is specified in the GUI (x is 0 to number of channels -1).

Table 7: **MAP Core Channel-Specific Registers**

Register Name	Type	Description	DCR Offset
CHANx_GFP_REGISTER_A		[31:28] = Reserved	0x8 + 8n (n=0...9)
	R/W	[27] = Core Configuration - Transparent mode(1) - Frame-mapped mode (0)	
	R/W	[26] = Enable FCS	
	R/W	[25] = Use UPI from REGISTER_A[23:16]	
	R/W	[24] = Use length from REGISTER_A[15:0]	
	R/W	[23:16] = Data UPI	
	R/W	[15:0] = Length	
CHANx_GFP_REGISTER_B		[31:18] = Reserved	0x9 + 8n (n=0...9)
	R/W	[17] = Send loss of client signal management frame when CSF timer expires	
	R/W	[16] = Send loss of character synchronization management frame when CSF timer expires	
	R/W	[15:8] = Spare field	
	R/W	[7:0] = Channel ID (CID) for this channel (alias)	

UNMAP Core Interfaces

Figure 5 displays the UNMAP core interfaces. All interfaces, and their associated signals, are defined in their respective sections following the illustration.

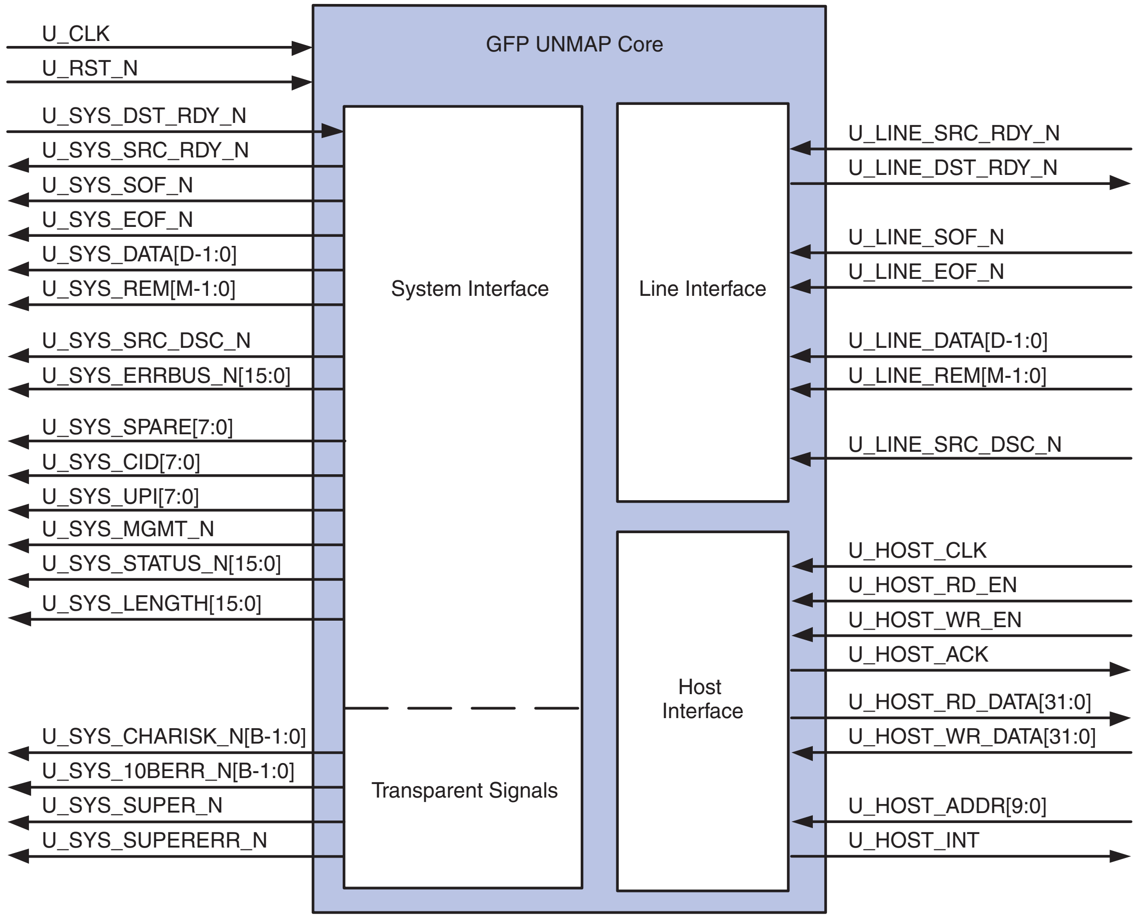

Figure 5: **UNMAP Core Interfaces**

Table 8 defines the relationship between the data bus width and additional signals. The UNMAP core supports both a 32-bit and a 64-bit interface. The bus widths of several signals, detailed below, depend on the interface width selected.

Table 8: **UNMAP Core Bus Widths**

Data Bus Width (D) (*_DATA)	Remainder Width (M) (*_REM)	Data Byte Width (B) (*_10BERR_N, *_CHARISK_N)
32	2	4
64	3	8

Common Interface

Table 9 describes the common interface. The UNMAP reset signal (`U_RST_N`) causes a hard reset of the entire core (core logic and host interface). This signal is an asynchronous input which is synchronized internally in the core before being used. The initial hardware reset should be generated by the user.

Subsequent resets may be asserted by using the U_RST_N pin (for complete core resets), or by driving the appropriate register in the UNMAP host interface (for core logic reset only).

Table 9: **MAP Common Interface**

Name	Direction	Description
U_CLK	Input	**UNMAP Clock:** All system and line interface operations are synchronous to this clock.
U_RST_N	Input	**UNMAP Reset:** Asynchronous reset which resets both the UNMAP core logic and UNMAP host interface.

System Interface (U_SYS)

Table 10 describes the UNMAP system interface signals (U_SYS). The UNMAP system interface connects to the client side of the system and implements the Xilinx LocalLink standard, providing a simple, flexible way to transmit frames. The system interface consists of a unidirectional data bus with control signals that provide the user application with real-time status about the current frame. The system interface signals are divided into three categories: signals common to both frame-mapped and transparent mode, signals specific to frame-mapped mode, and signals specific to transparent mode.

Table 10: **UNMAP System Interface**

Name	Direction	Description
Common to frame-mapped and transparent modes		
U_SYS_DST_RDY_N	Input	**Read Enable** (Destination Ready)**:** Indicates a word is accepted this cycle (not accepted until U_SYS_SRC_RDY_N is also asserted).
U_SYS_SRC_RDY_N	Output	**Read Valid** (Source Ready)**:** Indicates a word presented by the system interface is valid (not read until U_SYS_DST_RDY_N is also asserted).
U_SYS_SOF_N	Output	**Start of Frame:** Indicates the beginning of a frame.
U_SYS_EOF_N	Output	**End of Frame:** Indicates the end of a frame.
U_SYS_DATA[D-1:0]	Output	**Data Bus:** Client network protocol data that has been extracted from the GFP frame. If Ethernet is being transferred (frame-mapped mode), the first word sent will be the destination address. In transparent mode, the data will be 8b/10b decoded data, which is either data or control/error (depending on the values of U_SYS_CHARISK_N and U_SYS_10BERR_N).
U_SYS_REM[M-1:0]	Output	**Data Remainder:** The number of valid bytes in U_SYS_DATA is U_SYS_REM + 1, and is MSB justified. This signal is only valid when U_SYS_EOF_N is asserted. Example: 32-bit data bus: REM = "00" => DATA[31:24] valid REM = "01" => DATA[31:16] valid REM = "10" => DATA[31:8] valid REM = "11" => DATA[31:0] valid
U_SYS_SRC_DSC_N	Output	**Frame Discontinue:** Indicates the current frame contains an error. This type of error will be asserted on U_SYS_ERRBUS_N.

Table 10: **UNMAP System Interface *(Cont'd)***

Name	Direction	Description
U_SYS_ERRBUS_N[15:0]	Output	**Error Condition:** Provides real-time errors and status for the current frame. The following errors are reported (bits 6, 2:0) if they are enabled in the host interface (UNMAP GFP_CTRL[3:0]). Bits 3-5 are always reported. [15:8] = reserved [7] = Transparent frame did not end on a superblock boundary [6] = FCS error [5] = cHEC corrected [4] = tHEC corrected [3] = eHEC corrected [2] = cHEC error [1] = tHEC error [0] = eHEC error
U_SYS_SPARE[7:0]	Output	**Spare Field:** Indicates the value of the spare field of a linear frame. This signal is present when linear extension headers are enabled.
U_SYS_CID[7:0]	Output	**Channel Identifier:** Indicates the channel ID for linear frames. This signal is present when linear extension headers are enabled.
U_SYS_UPI[7:0]	Output	**User Payload Identifier:** Indicates the payload type for the current data frame. When U_SYS_MGMT_N is asserted, this signal indicates the type of management frame received. See the *GFP User Guide* for detailed UPI definitions.
U_SYS_MGMT_N	Output	**Management Frame Indicator:** Indicates a client management frame was received. When asserted, U_SYS_UPI indicates the type of client signal fail (CSF).
U_SYS_STATUS_N[15:0]	Output	**Status Bus:** Provides real-time status. [15:5] = reserved [4] = System FIFO almost full [3] = reserved Synchronization Status**:** Indicates the current status of the frame delineation state machine (one-cold encoding: [2] = SYNC [1] = PRESYNC [0] = HUNT
U_SYS_LENGTH[15:0]	Output	**Payload Length:** Indicates the length of the current frame in bytes.
Transparent signals		
U_SYS_CHARISK_N[B-1:0]	Output	**K Character Indicator:** Indicates that the corresponding byte of U_SYS_DATA is a 8b/10b K character, not a data character. This signal should be ignored for a given byte lane if U_SYS_REM indicates that the byte is not valid.
U_SYS_10BERR_N[B-1:0]	Output	**Insert 10B_ERR:** Indicates that the corresponding byte of U_SYS_DATA is an illegal 8b/10b word. This signal should be ignored for a given byte lane if U_SYS_REM indicates that the byte is not valid.
U_SYS_SUPER_N	Output	**Superblock Indication:** Asserted at the start of every new superblock for one clock cycle.
U_SYS_SUPERERR_N	Output	**Superblock CRC Error Indication:** Asserted for one clock cycle at the end of a superblock which contains a CRC-16 error.

Line Interface (U_LINE)

Table 11 describes the UNMAP line interface signals (U_LINE). The UNMAP line interface utilizes the Xilinx LocalLink standard, providing a simple, flexible way to receive frames. It consists of a unidirectional data bus with control signals.

Table 11: **UNMAP Line Interface**

Name	Direction	Description
U_LINE_SRC_RDY_N	Input	**Write Enable**: Indicates a word presented by the client is valid (not accepted until U_SYS_DST_RDY_N is also asserted).
U_LINE_DST_RDY_N	Output	**Write Accepted**: Indicates a word presented by the client will be accepted (if U_SYS_SRC_RDY_N is also asserted).
U_LINE_SOF_N	Input	**Start of Frame**: Indicates the beginning of a new GFP frame. This signal is only valid if No Hunting is selected for frame delineation.
U_LINE_EOF_N	Input	**End of Frame**: Indicates the end of the current GFP frame. This signal is only valid if No Hunting is selected for frame delineation.
U_LINE_DATA[D-1:0]	Input	**Data Bus**: Receives GFP encapsulated frames.
U_LINE_REM[M-1:0]	Input	**Data Remainder**: The number of valid bytes in U_LINE_DATA is U_LINE_REM + 1, and is MSB justified. This signal is only valid when U_LINE_EOF_N is asserted. Refer to U_SYS_REM for valid byte mappings. This signal is only valid if No Hunting is selected for frame delineation.
U_LINE_SRC_DSC_N	Input	**Discontinue Frame**: Indicates the current GFP frame contains an error. This signal is only valid if No Hunting is selected for frame delineation.

Host Interface (U_HOST)

Table 12 describes the host interface signals (U_HOST). The UNMAP host interface to the user application utilizes the DCR bus for direct connection to the PowerPC, Microblaze, or other microprocessor. It consists of separate read and write data buses with a shared address bus. Note that the host interface is optional, and may be removed to conserve resources or if *in situ* access to control registers is not required.

Table 12: **UNMAP Host Interface**

Name	Direction	Description
U_HOST_CLK	Input	**Host Clock**: All host interface signals are synchronous to this clock. Note that this signal is optional, as the host interface can be configured to be synchronous to U_CLK.
U_HOST_RD_EN	Input	**Read Enable**: Indicates a read access to the register addressed on U_HOST_ADDR.
U_HOST_WR_EN	Input	**Write Enable**: Indicates a write access to the register addressed on U_HOST_ADDR.
U_HOST_ACK	Output	**Acknowledge**: Both read and write requests to the host interface are acknowledged through this signal. Following a read request, this signal indicates that the data on U_HOST_RD_DATA is valid. Following a write request, this signal indicates that the data on U_HOST_WR_DATA was accepted.
U_HOST_RD_DATA[31:0]	Output	**Read Data**: Data read from the address U_HOST_ADDR, which is valid when a read request (U_HOST_RD_EN) followed by an acknowledge (U_HOST_ACK) is asserted.
U_HOST_WR_DATA[31:0]	Input	**Write Data**: Data to be written into the register specified by the address U_HOST_ADDR. The write is acknowledged on U_HOST_ACK when the write succeeded.
U_HOST_ADDR[9:0]	Input	**Register Address**: Bus used to specify the address being accessed either for a read or write.
U_HOST_INT	Output	**Interrupt**: Indicates that an interrupt condition has been detected. This signal will be driven until the interrupt is cleared (by writing to GFP_INT).

UNMAP Configuration Space

The UNMAP core supports a host interface for access to control and status registers *in situ*. The UNMAP core supports only general registers, which are defined in Table 13.

Table 13: **UNMAP Core General Registers**

Register Name	Type	Description	DCR Offset
GFP_VERSION	R/W	[31] = Core reset	0x0
	R	[30:0] = Core version number	
GFP_CTRL		[31:28] = Reserved	0x1
	R/W	[27:24] = Number of cHEC matches required during the PRESYNC state to synchronize the core (delta, minimum=1)	
		[23] = Reserved	
	R/W	[22] = Disable extension header error check (eHEC) correction	
	R/W	[21] = Disable type field error check (tHEC) correction	
	R/W	[20] = Disable core header error check (cHEC) correction	
	R/W	[19] = Disable descrambling of header	
	R/W	[18] = Disable descrambling of payload	
		[17:5] = Reserved	
	R/W	[4] = Disable error reporting of superblock CRC error	
	R/W	[3] = Disable error reporting of FCS error	
	R/W	[2] = Disable error reporting of cHEC error	
	R/W	[1] = Disable error reporting of tHEC error	
	R/W	[0] = Disable error reporting of eHEC error	
GFP_FIXERR[1]		[31:27] = Reserved	0x2
	R/W	[26] = cHEC error corrected	
	R/W	[25] = tHEC error corrected	
	R/W	[24] = eHEC error corrected	
	R/W	[23:16] = CID of corrected cHEC frame	
	R/W	[15:8] = CID of corrected tHEC frame	
	R/W	[7:0] = CID of corrected eHEC frame	
GFP_INTMASK		[31:5] = Reserved	0x3
	R/W	[4] = Enable interrupt for superblock CRC error	
	R/W	[3] = Enable interrupt for FCS error	
	R/W	[2] = Enable interrupt for cHEC error	
	R/W	[1] = Enable interrupt for tHEC error	
	R/W	[0] = Enable interrupt for eHEC error	

Table 13: **UNMAP Core General Registers *(Cont'd)***

Register Name	Type	Description	DCR Offset
GFP_INT[2]		[31:5] = Reserved	0x4
	R/W	[4] = Superblock CRC error	
	R/W	[3] = FCS error	
	R/W	[2] = cHEC error	
	R/W	[1] = tHEC error	
	R/W	[0] = eHEC error	

1. This register holds the first value detected until cleared. All errors are cleared by performing a write operation to this register.
2. All interrupts are cleared by performing a write operation to this register.

Resource Utilization and Performance

The GFP core can be generated in numerous configurations to provide maximum flexibility for specific user applications. For this reason, resource utilization heavily depends on the core configuration selected by the user. Table 14 and 15 provide sample resource utilization numbers for the GFP core. All cores in the tables are generated with 10 channels and all features enabled; reducing the channel count and/or disabling features reduces the core's resource requirements.

The most effective method for determining resource requirements for a user application is to generate a core with the required feature set, and then run the provided implementation script. See the UG151, *GFP Getting Started Guide* for details.

Table 14: **MAP Core Resource Utilization**

Configuration	32-bit Data Bus Width		64-bit Data Bus Width	
	Slices	RAMB16	Slices	RAMB16
Frame-Mapped Mode	1635	0	2608	0
Transparent Mode	2794	1	3753	1
Mixed Mode	3075	1	3884	1
Host Interface	(+) 350	N/A	(+) 350	N/A

Note: The MAP core resource utilization is configured in 10-channel with all features on/enabled.

Table 15: **UNMAP Core Resource Utilization**

Configuration	32-bit Data Bus Width		64-bit Data Bus Width	
	Slices	RAMB16	Slices	RAMB16
Frame-Mapped Mode	1404	5	3301	10
Transparent Mode	2801	9	5249	15
Mixed Mode	2801	9	5249	15
Host Interface	(+) 130	N/A	(+) 130	N/A

Note: The UNMAP core resource utilization is configured in 10-channel with all features on/enabled.

The GFP core can be implemented targeting four FPGA families, enabling the user to choose the family best suited for their specific application. The maximum operating frequency of the GFP core depends on the selected target device. Table 16 defines the maximum operating frequencies of the GFP core. Note that the user constraints file (UCF) delivered with the core is set to a default value that may not match the maximum performance numbers defined in Table 16; however, the user can adjust the constraints as described in "Chapter 7, Constraining the Core" of the *GFP User Guide*.

Table 16: **Maximum Operating Frequencies (MHz)**

FPGA Family	Speed Grade	32-bit Data Bus Width	64-bit Data Bus Width
Virtex-4	-10	180	170
	-11	200	200
Virtex-II Pro	-5	175	140
	-6	190	160
	-7	210	180
Virtex-II	-4	140	120
	-5	160	130
	-6	180	150
Spartan-3	-4	110	95
	-5	130	110

Verification

Xilinx has verified the GFP core in a proprietary test environment, using an internally developed bus functional model (BFM). Thousands of test vectors have been generated and verified, including both correct and errored frames. For details on the verification of the GFP core, see the *GFP User Guide*.

References

1. UG151, *GFP Getting Started Guide*
2. UG152, *GFP User Guide*
3. *ITU-T G.7041 GFP Specification*

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 SignOnce IP Site License. A free evaluation version of the module is provided with the Xilinx CORE Generator, which lets you assess the core functionality and demonstrates the various interfaces of the core in simulation. To access the full functionality of the core, including FPGA bitstream generation, a full license must be obtained from Xilinx.

Please contact your local Xilinx sales representative for pricing and availability of additional Xilinx LogiCORE IP modules and software. Information about additional Xilinx solutions is available on the Xilinx IP Center.

Revision History

Date	Version	Revision
9/30/04	1.1	Xilinx initial release.
11/11/04	1.2	Updated 64-bit information and added resource utilization/core performance data.
4/28/05	1.3	Updated to ISE 7.1i SP1, core v1.3, support for Spartan-3E devices.
1/18/06	1.4	Update for ISE 8.1i.
4/25/08	2.1	Update for ISE v10.1 and added support for Virtex-5 devices.

09/29/09 - This is the final publication. No content was changed.

Notice of Disclaimer