HB0784
Handbook
CoreXAUI v2.0

Microsemi
Power Matters.

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Fax: +1 (949) 215-4996
Email: sales.support@microsemi.com
www.microsemi.com

About Microsemi

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for aerospace & defense, communications, data center and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; enterprise storage and communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, California, and has approximately 4,800 employees globally. Learn more at www.microsemi.com.

Contents

Figures

Tables

1 Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the most current publication.

1.1 Revision 1.0

Revision 1.0 is the first publication of this document. Created for CoreXAUI v2.0.

2 Introduction

CoreXAUI core provides 10Gb Ethernet Extended Sublayer (XGXS) capabilities in PolarFire FPGA fabric. This IP core connects to PolarFire Transceiver (PF_XCVR) in PCS mode using the built-in 8B10B Encoding/Decoding and Comma Detector Logic. This IP core provides support for standard XAUI and Reduced XAUI (RXAUI), which use four or two transceiver lanes respectfully to achieve the required 10Gbps data rate.

XAUI uses four transceiver lanes at 3.125 GHz to achieve the 10Gbps data rate. This is achieved using 16-bit data per PCS lane at 156.25MHz. RXAUI uses two transceiver lanes at 6.25 GHz to achieve the 10Gbps data rate. This is achieved using 32-bit data per PCS lane at 156.25MHz.

2.1 Features

CoreXAUI v2.0 has the following features:

- Operations frequency of 156.25MHz
- Pseudorandom Idle Insertion using PRBS X7 + X3 + 1
- XAUI (Idle Character) Encoding
- Support 64bit @ SDR in transmitter and receiver path
- Will use PF_XCVR built-in 8B10B Encoder
- Will use PF_XCVR built-in 8B10B Decoder and COMMA word aligner
- Support multichannel alignment and lane deskew in receiver path
- XAUI (Idle Character) Decoding
- Support for 10Gbps data rate across four lanes of SERDES running at 3.125 GHz
- Support for 10Gbps data rate across two lanes of SERDES running at 6.25 GHz (that is, support for RXAUI)

2.2 Core Versions

This handbook applies to CoreXAUI v2.0. The release notes provided with the core list known discrepancies between this handbook and the core release associated with the release notes.

2.3 Supported Families

- PolarFire™

2.4 Utilization and Performance

CoreXAUI has been implemented in the following Microsemi device families. A summary of the implementation data for CoreXAUI is listed in Table 1.

Table 1 • **CoreXAUI Utilization and Performance**

	Tiles			Utilization		Performance
Family	**Sequential**	**Combinatorial**	**Total**	**Device**	**Total %**	**MHz**
PolarFire	2328	3652	4980	MPF300T_ES	1.68	156.25

3 Functional Description

CoreXAUI supports 64-bit XGMII at single data rate. It connects to a TX/RX XGMII Client and to the Transceiver through the PCS Interface. Supports XAUI (16-bit per lane) or RXAUI (32-bit per lane) data path configuration. RXAUI configuration complies with the Dune Networks specification by maintaining 8b10b encoding disparity per RXAUI physical lane.

The transmitter path consists of five main blocks:

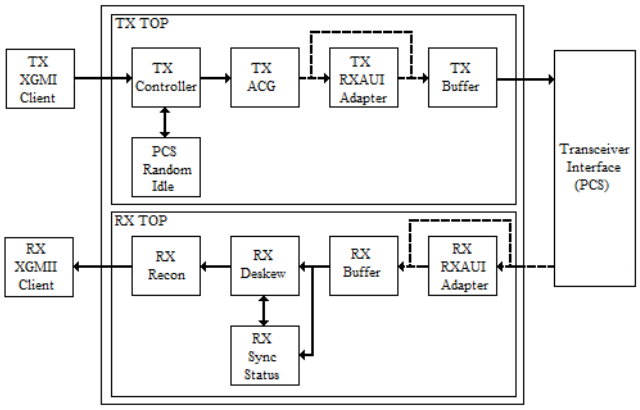
1. Tx Controller
2. TX ACG
3. TX RXAUI Adapter
4. TX Buffer
5. PCS Random Idle.

The receiver path consists of five main blocks:

1. RX RXAUI Adapter
2. RX Buffer
3. RX Deskew
4. RX Recon
5. Rx Sync Status

The CoreXAUI system level diagram is as shown in Figure 1.

Figure 1 • **CoreXAUI Block Diagram**



3.1 CoreXAUI Blocks

CoreXAUI contains the following blocks:

3.1.1 TX Controller

This block controls the transmission of XGMII data and control information. It recognizes the XGMII Start, Terminate, Idle, and Sequence control characters used to determine whether packet, Idle or link status is being sourced by the XGMII and informs TX ACG whether to transmit data or special characters in all or individual octets.

3.1.2 TX ACG

This block controls the transmission of all code groups. The Tx Controller informs this block whether to transmit data or special characters in all or individual octets.

3.1.3 TX RXAUI Adapter

This block provides a XAUI to RXAUI adapter in the transmitter path. When XAUI_MODE is configured as RXAUI, this block will take XAUI transmit data and multiplex it over two PCS lanes rather than four PCS lanes, which occurs when XAUI_MODE is configured as XAUI.

3.1.4 TX Buffer

This block buffer transmitter data to provide a clock domain crossing between XGMII transmit clock domain and PCS transmit domain. A small FIFO is used to achieve successful clock domain crossing.

3.1.5 PCS Random Idle

This block provides the PCS pseudorandom idle character insertion to the Tx Controller using the polynomial X7 + X3 + 1. The purpose of randomizing the idle sequence is to reduce electromagnetic interface (EMI) idle transmission. The Align, Sync, and Skip code group selection is based on bit-0 of the pseudorandom generated data being even/odd and a 5-bit down counter, which is used to control the Align code group spacing.

3.1.6 RX RXAUI Adapter

This block provides a RXAUI to XAUI adapter in the receiver path. When XAUI_MODE is configured as RXAUI, this block will take two PCS lanes and demultiplex them into XAUI received data rather than demultiplexing four PCS lanes, which occurs when XAUI_MODE is configured as XAUI.

3.1.7 RX Buffer

This block buffer receiver data to provide a clock domain crossing between XGMII receive clock domain and PCS receive domain. A small FIFO is used to achieve successful clock domain crossing.

3.1.8 RX Deskew

This block provide PCS lane deskew of the received data. This is achieved by monitoring for the Align code group in each lane and deskewing the lanes, so that, they all align to this code group. Deskew is successful when four occurrences of Align code group appears on each lane in the same clock cycle. RX_STATUS will assert when deskew alignment has been successful. Deskew process will retrigger when four or more occurrences of Align code group appear on each lane in the different clock cycle and RX_STATUS will deassert to indicate deskew alignment has been lost.

3.1.9 Rx Sync Status

This block provides the sync status to the Rx Deskew block. By monitoring the 8b10b error signals and the decoded data/control signals on each lane, this block determines whether the receive channel is ready for operation or if failure in the channel has occurred (for example, wrong word alignment), which means the channel should be reset. The LANE[n]_PCS_ARST_N signal will assert when failure occurs on an individual lane to reset the 8b10b logic and COMMA word aligner in the PolarFire transceiver.

3.1.10 RX Recon

This block reconstructs the octets back into XGMII data and control. It monitors XGMII Terminate, Idle, Error control characters and will produce errors if invalid 8b10b data is received.

4 Core Structure

4.1 XGMII Structure

The XGMII is composed of independent transmit and receive paths. Each direction uses 64-bit data (TXD & RXD) and 8-bit control (TXC & RXC), which both have their own clock domain (TX_CLK & RX_CLK). The data and control are organized into four PCS lanes when XAUI_MODE is configured as XAUI and are organized into two PCS lanes when XAUI_MODE is configured as RXAUI. Each PCS lane has its own clock domain (LANE[n]_[TX|RX]_CLK) and a small FIFO is used as a clock domain crossing buffer.

When XAUI_MODE is configured as XAUI, the first octet is aligned to lane 0, the second octet to lane 1, the third octet to lane 2, the fourth octet to lane 3, the fifth octet to lane 0 and so on. When XAUI_MODE is configured as RXAUI, the number of lanes are reduced by doubling the data & control widths and concatenating XAUI lane 0 & lane 1 onto RXAUI lane 0 and XAUI lane 2 & lane 3 onto RXAUI lane 1, this is how the reduction of lanes is achieved. RXAUI configuration complies with the Dune Networks specification by maintaining 8b10b encoding disparity per RXAUI physical lane.

CoreXAUI transmitter/receiver lane associations for the XGMII data /control muxing when configured in XAUI mode is described in Table 2:

Table 2 • **XAUI Mode: XGMII Muxing & Lane Association**

XGMII Data	XGMII Control	PCS Lane
[39:32], [7:0]	[4], [0]	0
[47:40], [15:8]	[5], [1]	1
[55:48], [23:16]	[6], [2]	2
[63:56], [31:24]	[7], [3]	3

CoreXAUI transmitter/receiver lane associations for the XGMII data /control muxing when configured in RXAUI mode is described in Table 3:

Table 3 • **RXAUI Mode: XGMII Muxing & Lane Association**

XGMII Data	XGMII Control	PCS Lane
[47:40], [39:32], [15:8], [7:0]	[5], [4], [1], [0]	0
[63:56], [55:48], [31:24], [23:16]	[7], [6], [3], [2]	1

CoreXAUI Permissible XGMII encodings of TXC/RXC and TXD/RXD are described in Table 4:

Table 4 • **Permissible XGMII encodings**

TXC / RXC	TXD / RXD	Description
0	00 through FF	Normal data transmission
1	07	Idle
1	9C	Sequence (only valid in lane 0)
1	FB	Start (only valid in lane 0)
1	FD	Terminate
1	FE	Transmit error propagation

The code mapping for XGMII to/from 8b10b encoding/decoding are described in Table 5:

Table 5 • **Code Mapping for XGMII to/from 8b10b**

Control Character	Notation	XGMII Control Code	8b10b Code
IDLE	/I/	0x07	Any one for the following: K28.0 (0x1C) K28.3 (0x7C) K28.5 (0xBC)
START	/S/	0xFB	K27.7 (0xFB)
TERMINATE	/T/	0xFD	K29.7 (0xFD)
ERROR	/E/	0xFE	K30.7(0xFE)
SEQUENCE	/Q/	0x9C	K28.4 (0x9C)
DATA	/d/	00 through FF	00 through FF **Note:** Normal data transmission when control signal for octet equal zero

Link faults are reported through the sequence order sets and can appear in the upper or lower 32-bits of the received data, the sequence order sets are described in Table 6:

Table 6 • **Sequence Order Sets**

4bit Control	32bit Data	Description
0x0001	0x010009C	Local Fault
0x0001	0x020009C	Remote Fault
0x0001	0x030009C	Link Interruption

4.2 Clocking

TX_CLK: This is a continuous clock used for operation at 10Gb/s. It provides the timing reference for the transfer of the 64-bit TXD and 8-bit TXC signals and is sampled on the rising edge. Its frequency shall be 156.25MHz ±0.01% of the MAC transmit data rate.

RX_CLK: This is a continuous clock used for operation at 10Gb/s. It provides the timing reference for the transfer of the 64-bit RXD and 8-bit RXC signals and is sampled on the rising edge. Its frequency shall be 156.25MHz ±0.01% of the MAC received data rate.

LANE[n]_TX_CLK: This is a continuous reference clock from the PolarFire transceiver and one is included per transmitter lane (that is, [n] = 0 to 3). It provides the timing reference for the LANE[n]_TX* signals and is sampled on the rising edge. Its frequency shall be 156.25MHz ±0.01% of the MAC transmit data rate.

LANE[n]_RX_CLK: This is a continuous reference clock from the PolarFire transceiver and one is included per receiver lane (that is, [n] = 0 to 3). It provides the timing reference for the LANE[n]_RX* signals and is sampled on the rising edge. Its frequency shall be 156.25MHz ±0.01% of the MAC received data rate.

5 Tool Flows

5.1 Licensing

CoreXAUI clear RTL is license locked and the obfuscated RTL is freely available.

5.1.1 Obfuscated

- Complete RTL code is provided for the core, enabling the core to be instantiated with SmartDesign.
- Simulation, Synthesis, and Layout can be performed with Libero software. The RTL code for the core is obfuscated using the IP encryption (encryptP1735.pl) solution.

5.1.2 RTL

Complete RTL source code is provided for the core.

5.2 SmartDesign

Note: CoreXAUI is compatible with Libero System-on-Chip (SoC) and Libero System-on-Chip (SoC) PolarFire. Unless specified otherwise, this document uses the name Libero to identify Libero SoC and Libero SoC PolarFire.

CoreXAUI is pre-installed in the SmartDesign IP deployment design environment or downloaded from the online repository. Figure 2 shows an example instantiated.

Figure 2 • **XAUI Configuration**

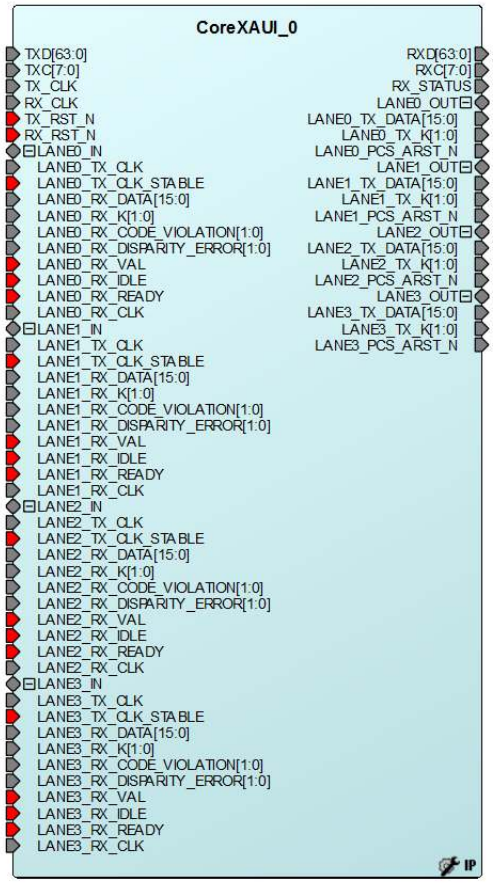
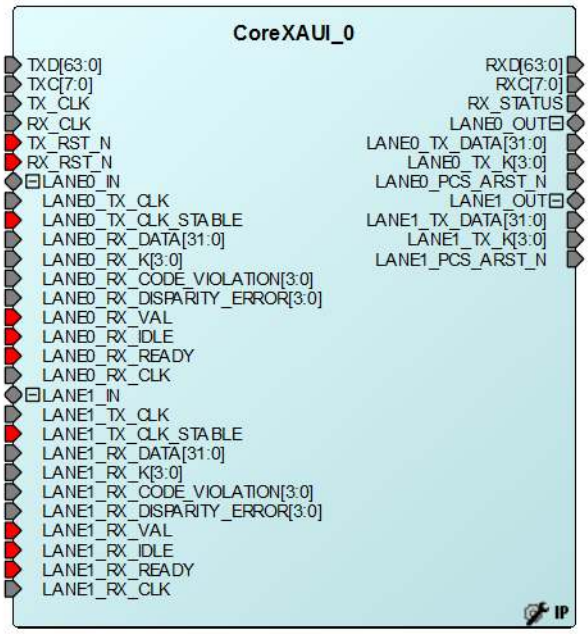
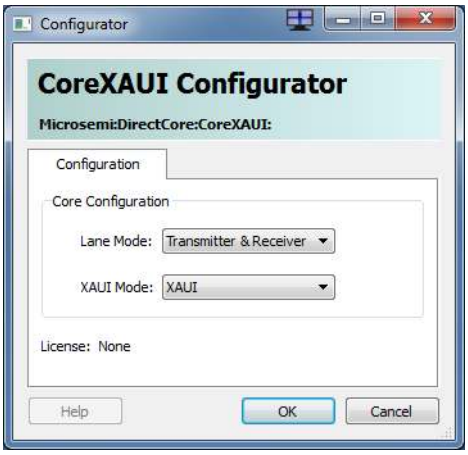


Figure 3 • **RXAUI Configuration**



The core can be configured using the configuration GUI within SmartDesign, as shown in Figure 4.

Figure 4 • **SmartDesign Configuration GUI**



5.3 Simulation Flows

The user testbench for CoreXAUI is included in all releases.

To run simulations, right-click the canvas, and select **Generate Design**.

When SmartDesign generates the design files, it will install the user testbench files.

To run the user testbench, Set the design root to the CoreXAUI instantiation in the Libero SoC design hierarchy pane, right click **Simulate** in the Libero SoC Design Flow window under Verify Pre-Synthesized Design and select **Open Interactively**. This invokes

ModelSim and automatically runs the simulation.

5.4 Synthesis in Libero

After setting the design root appropriately for your design, click the **Synthesis** icon in Libero SoC. The Synthesis window appears, displaying the Synplicity® project. Set Synplicity to use the Verilog 2001 standard if Verilog is being used. To run Synthesis, click the **Run** icon.

5.5 Place-and-Route in Libero

After setting the design root appropriately for your design, and after running Synthesis, click the Layout icon in Libero SoC to invoke Designer. CoreXAUI requires no special place-and-route settings.

5.6 Constraints

As the CoreXAUI is a high-speed IP core, it is important to provide it with constraints in order to Place-and-Route a Libero Soc design without timing violations.

5.6.1 Synthesis Constraints

```
###==== CLOCKS
create_clock -name {LANE0_TX_CLK} -period 6.4 -waveform {0 3.2 } [ get_ports
{ LANE0_TX_CLK } ]
create_clock -name {LANE1_TX_CLK} -period 6.4 -waveform {0 3.2 } [ get_ports
{ LANE1_TX_CLK } ]
create_clock -name {LANE2_TX_CLK} -period 6.4 -waveform {0 3.2 } [ get_ports
{ LANE2_TX_CLK } ]
create_clock -name {LANE3_TX_CLK} -period 6.4 -waveform {0 3.2 } [ get_ports
{ LANE3_TX_CLK } ]
create_clock -name {LANE0_RX_CLK} -period 6.4 -waveform {0 3.2 } [ get_ports
{ LANE0_RX_CLK } ]
create_clock -name {LANE1_RX_CLK} -period 6.4 -waveform {0 3.2 } [ get_ports
{ LANE1_RX_CLK } ]
create_clock -name {LANE2_RX_CLK} -period 6.4 -waveform {0 3.2 } [ get_ports
{ LANE2_RX_CLK } ]
create_clock -name {LANE3_RX_CLK} -period 6.4 -waveform {0 3.2 } [ get_ports
{ LANE3_RX_CLK } ]
create_clock -name { RX_CLK} -period 6.4 -waveform {0 3.2 } [ get_ports RX_CLK
} ]
create_clock -name {TX_CLK} -period 6.4 -waveform {0 3.2 } [ get_ports { TX_CLK
} ]
```

Note: These synthesis constraints are for when LANE_MODE = 2 and XAUI_MODE = 0, all other configurations will be a reduced version of these synthesis constraints since numerous clocks are removed in various configurations.

6 Core Interfaces

The following interfaces will be available for CoreXAUI:

- PCS Fabric Interface
- XGMII Interface

6.1 Configuration Parameters

CoreXAUI parameters/generics are described in Table 7.

Table 7 • **CoreXAUI Parameters/Generics Descriptions**

Name	Range	Default	Description
FAMILY	26	26	Family: 26: PolarFire
XAUI_MODE	0 - 1	0	XAUI Mode 0: XAUI - 4 lanes 1: RXAUI - Reduced XAUI (that is, 2 lanes)
LANE_MODE	0-2	2	Lane Mode 0: Transmitter Only 1: Receiver Only 2: Transmitter & Receiver

6.2 I/O Signals

CoreXAUI will have the following PCS ports available:

Table 8 • **I/O Signals**

Name	Width	Direction	Description
LANE[n]_TX_CLK	1	IN	LANE[n]_IN Where n can be 0, 1, 2, or 3 depending on the number of configured lanes.
LANE[n]_TX_CLK_STABLE	1	IN	
LANE[n]_RX_DATA	16 or 32	IN	
LANE[n]_RX_K	2 or 4	IN	
LANE[n]_RX_CODE_VIOLATION	2 or 4	IN	
LANE[n]_RX_DISPARITY_ERROR	2 or 4	IN	
LANE[n]_RX_VAL	1	IN	
LANE[n]_RX_IDLE	1	IN	
LANE[n]_RX_READY	1	IN	
LANE[n]_RX_CLK	1	IN	
LANE[n]_TX_DATA	16 or 32	OUT	LANE[n]_OUT Where n can be 0, 1, 2, or 3 depending on the number of configured lanes.
LANE[n]_TX_K	2 or 4	OUT	
LANE[n]_PCS_ARST_N	1	OUT	

CoreXAUI will have the following Fabric ports available as described in Table 9:

Table 9 • **Fabric Ports**

Name	Width	Direction	Description
TXD	64	IN	Transmitter Data
TXC	8	IN	Transmitter Control
TX_CLK	1	IN	Transmitter Clock
TX_RST_N	1	IN	Transmitter Reset: This is an active low signal.
RXD	64	OUT	Receiver Data
RXC	8	OUT	Receiver Control
RX_STATUS	1	OUT	Receiver Status - Asserts high when then the XAUI link state machine and alignment is done.
RX_CLK	1	IN	Receiver Clock
RX_RST_N	1	IN	Receiver Reset: This is an active low signal.

7 Timing Diagrams

7.1 XGMII Timing Diagrams

Figure 5 shows the timing diagram for normal packet transmission:

Note: Dp: Preamble Data Octet, SFD: Start Frame Delimiter

Figure 5 • **Normal Packet Transmission**

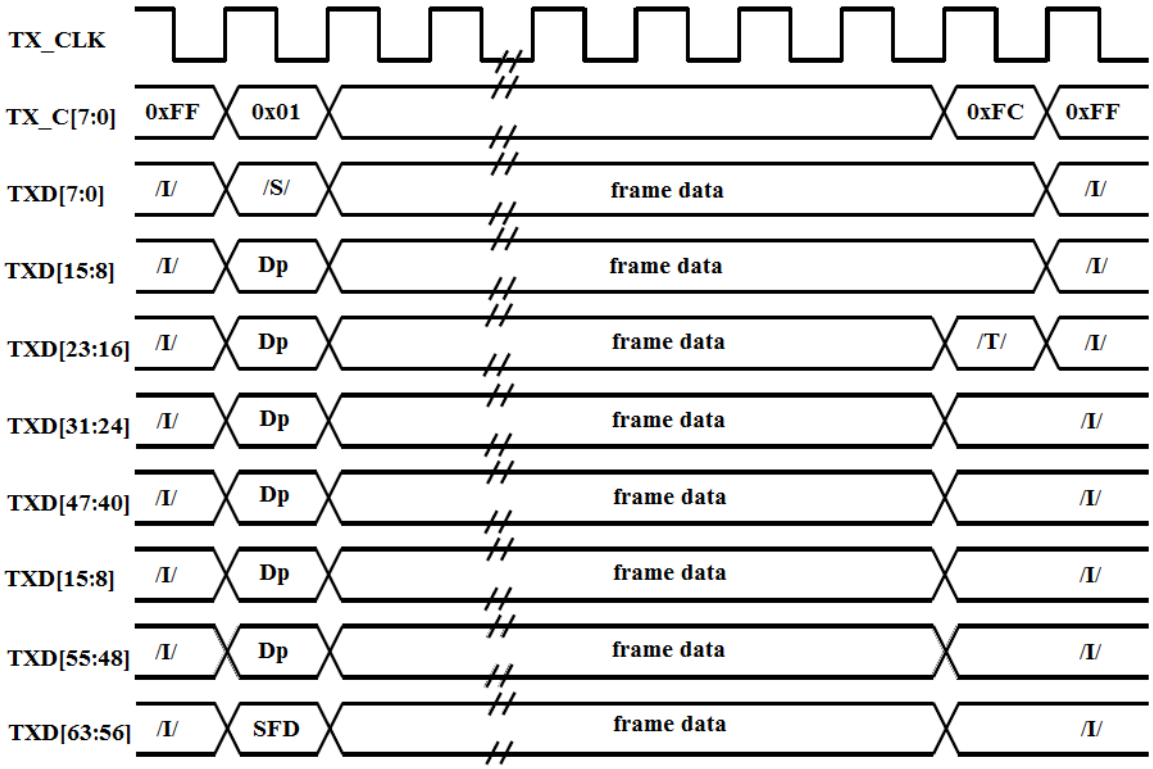


Figure 6 shows the timing diagram for transmit error propagation:

Note: Dp: Preamble Data Octet, SFD: Start Frame Delimiter

Figure 6 • **Transmit Error Propagation**

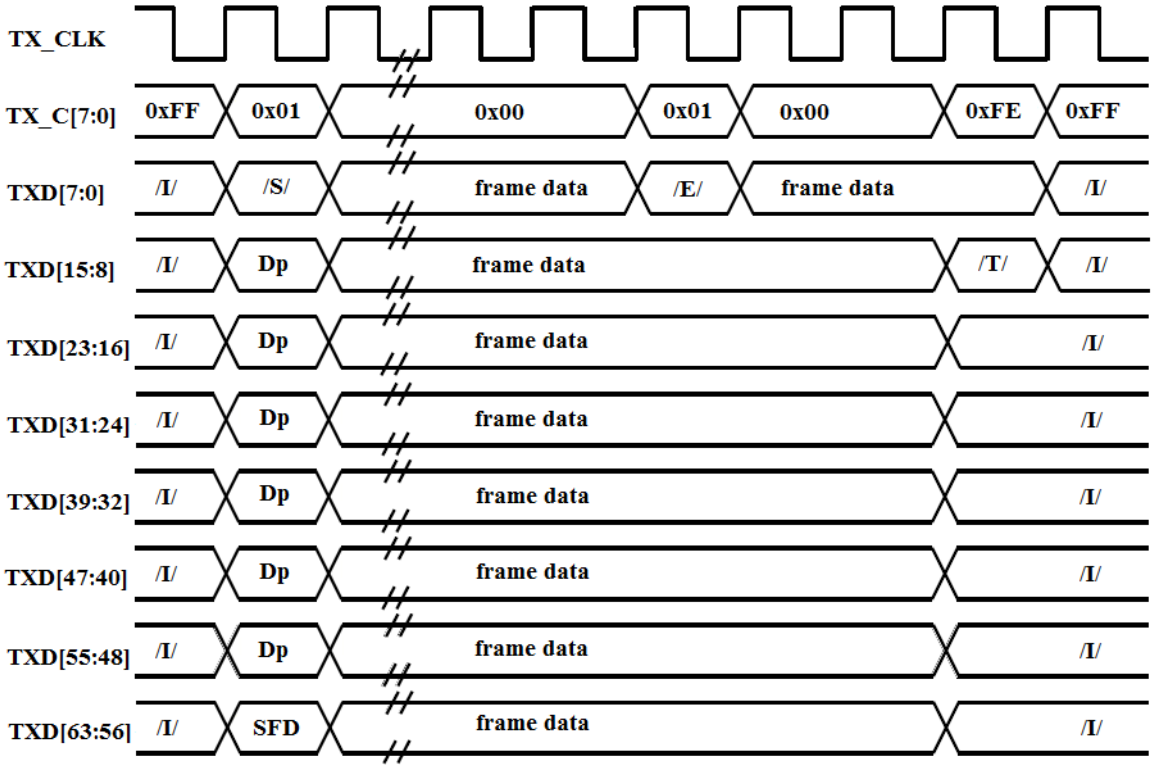


Figure 7 shows the timing diagram for basic frame reception:

Note: Dp: Preamble Data Octet, SFD: Start Frame Delimiter

Figure 7 • **Basic Frame Reception**

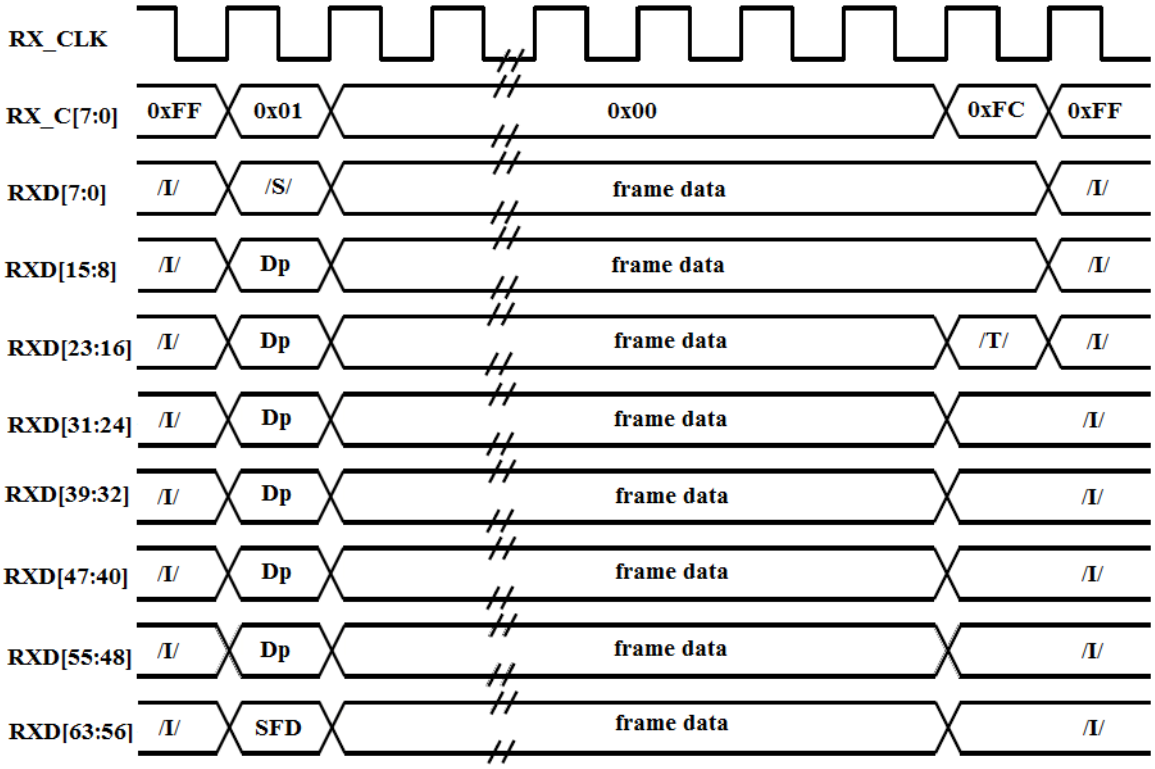
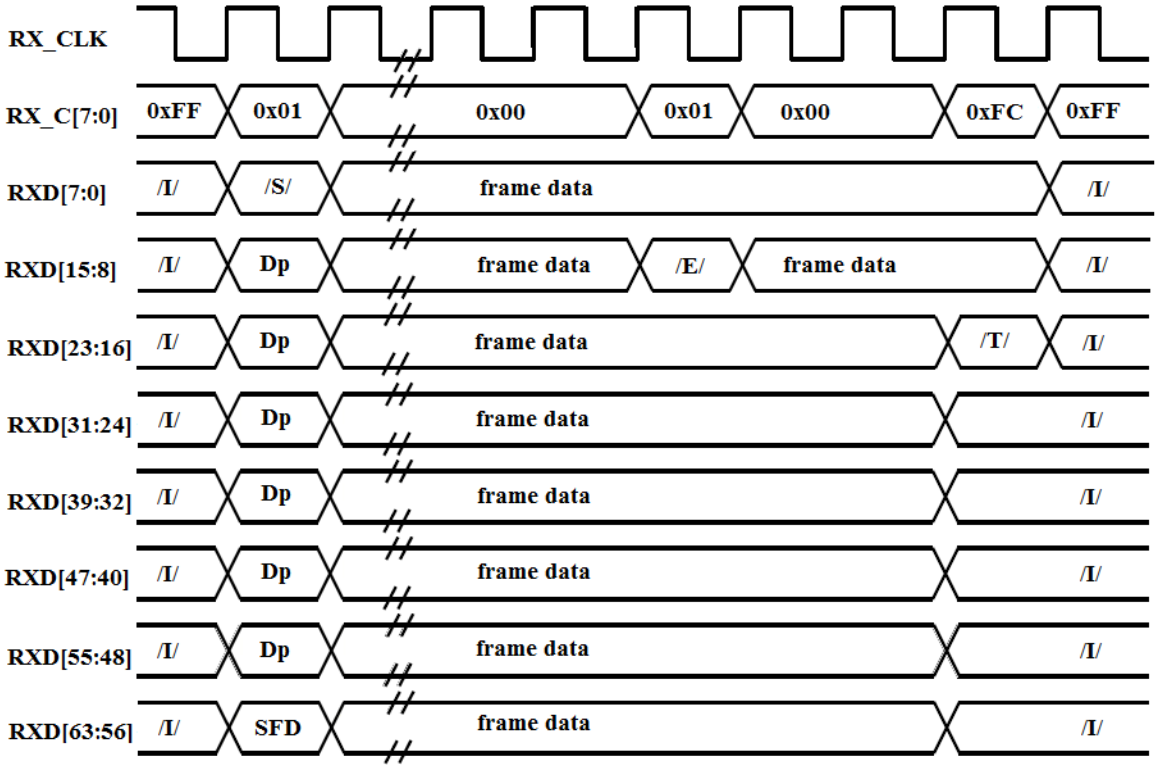


Figure 8 shows the timing diagram for reception with error:

Note: Dp: Preamble Data Octet, SFD: Start Frame Delimiter

Figure 8 • **Reception With Error**

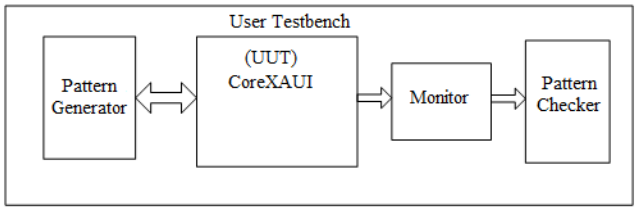


8 Testbench Operation

8.1 User Testbench

The CoreXAUI user testbench gives an example of how to use the core.

Figure 9 • **User Testbench**



The simulation testbench shown in Figure 9 includes an instantiation of the CoreXAUI macro, data generation, and data monitor and checker. The purpose of the testbench is to test the functionality of the core by inputting known data, monitoring the output, and checking for expected results.

The core is delivered with a simple simulation test-bench. This test-bench is purely delivered as a vehicle to get started with using the core, and is not an attempt at an exhaustive testbench.

9 Ordering Information

9.1 Ordering Codes

CoreXAUI can be ordered through your local Microsemi sales representative. It should be ordered using the following number scheme: CoreXAUI -XX, where XX is listed in Table 10.

Table 10 • **Ordering Codes**

XX	Description
RM	RTL multi-use multiple-site license