

## Introduction

The AXI4-Stream FIFO core allows memory mapped access to a AXI4-Stream interface. The core can be used to interface to the AXI Ethernet without the complexity or resource utilization of using DMA.

The principal operation of this core allows the write or read of data packets to or from a device without any concern over the AXI4-Stream interface signalling. The management of the AXI4-Stream interfaces is transparent to the user.

## Features

- 32-bit AXI4-Lite slave interface
- Independent configurable internal TX and RX data FIFOs
- Full duplex operation.
- Supports AXI Ethernet basic mode only
- Provides interrupts for error and status conditions

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family <sup>(1)</sup>	Zynq™-7000, Artix™-7, Virtex®-7, Kintex™-7, Virtex-6, Spartan®-6
Supported User Interfaces	AXI4-Lite , AXI4-Stream
Resources	See <a href="#">Table 16</a> and <a href="#">Table 17</a>
Provided with Core	
Documentation	Product Specification
Design Files	VHDL
Example Design	Not Provided
Test Bench	Not Provided
Constraints File	Not Provided
Simulation Model	N/A
Supported S/W Driver <sup>(3)</sup>	Standalone
Tested Design Flows <sup>(4)</sup>	
Design Entry	Vivado™ Design Suite 2012.2 <sup>(5)</sup> ISE™ Design Suite Embedded Edition 14.2
Simulation	Mentor Graphics ModelSim
Synthesis	Vivado Synthesis XST 14.2
Support	
Provided by Xilinx @ <a href="http://www.xilinx.com/support">www.xilinx.com/support</a>	

### Notes:

1. For a complete listing of supported devices, see the [release notes](#) for this core.
2. Supported in ISE Design Suite Embedded Edition implementations only.
3. Standalone driver details can be found in the EDK or SDK directory (<install\_directory>/doc/usenglish/xilinx\_drivers.htm). Linux OS and driver support information is available from [//wiki.xilinx.com](http://wiki.xilinx.com).
4. For a listing of the supported tool versions, see the [Xilinx Design Tools: Release Notes Guide](#).
5. Supports only 7 series devices.

## Overview

Figure 1 shows the major components in the AXI4-Stream FIFO core — an AXI Interface block with an AXI4-Lite Slave interface, Interrupt Controller, a Registers module, a Receive Control Module, a Transmit Control Module, a Receive FIFO for the receive data and length, and a Transmit FIFO for the transmit data and the length.

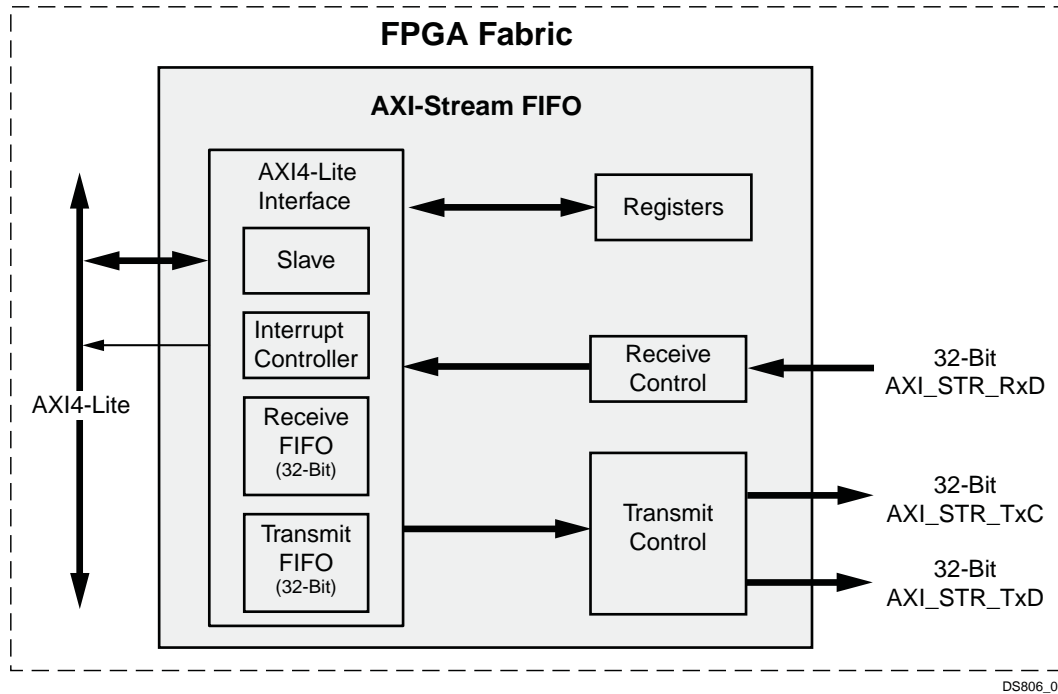


Figure 1: AXI4-Stream FIFO Core Block Diagram

## I/O Signals

The AXI4-Stream FIFO core uses a *transparent bus* EDK format to simplify the connection of signals between the AXI4-Stream interface of the core and AXI4-Stream interfaces of other IP such as the AXI Ethernet core. This technique allows the EDK tools to automatically connect the AXI signals as an entire group.

When using AXI4-Stream FIFO core with the AXI Ethernet core, connect the three AXI4-Stream interfaces listed:

1. AXI\_STR\_TXD - AXI4-Stream Transmit Data
2. AXI\_STR\_TXC - AXI4-Stream Transmit Control
3. AXI\_STR\_RXD - AXI4-Stream Receive Data

In addition, to support the transmit protocol of the AXI Ethernet core, add the AXI\_STR\_TxC interface. This interface does not provide any actual data related to the transmit packet data.

The AXI4-Stream FIFO core uses one clock from the AXI4-Lite interface for all clock inputs. When the AXI Ethernet core is used with the AXI4-Stream FIFO core, all the AXI Stream input clocks of the AXI Ethernet core must use the same clock.

Table 1: I/O Signals

Signal Name	Direction	Description
<b>Top Level System Signals</b>		
Interrupt	O	System Interrupt
<b>AXI4-Lite Signals</b>		
S_AXI_ACLK	I	AXI Clock
S_AXI_ARESETN	I	AXI Reset, active LOW
S_AXI_AWADDR(C_SAXI_ADDR_WIDTH-1:0)	I	AXI Write address
S_AXI_AWVALID	I	AXI address valid
S_AXI_AWREADY	O	AXI address ready
S_AXI_WDATA(C_SAXI_DATA_WIDTH-1:0)	I	Write data
S_AXI_WSTRB((C_SAXI_DATA_WIDTH/8)-1:0)	I	Write strobes
S_AXI_WVALID	I	Write valid
S_AXI_WREADY	O	Write ready
S_AXI_BRESP(1:0)	I	Write response
S_AXI_BREADY	O	Write response ready
S_AXI_ARADDR(C_SAXI_ADDR_WIDTH-1:0)	I	Read address
S_AXI_ARVALID	I	Read address valid
S_AXI_ARREADY	O	Read address ready
S_AXI_RDATA(C_SAXI_DATA_WIDTH-1:0)	I	Read data
S_AXI_RRESP(1:0)	I	Read response
S_AXI_RVALID	I	Read valid
S_AXI_RREADY	O	Read ready
<b>AXI4-Stream Transmit Data Channel Signals</b>		
AXI_STR_TXD_ACLK	I	Clock for the AXI4-Stream Transmit data interface
mm2s_prmry_reset_out_n	O	Reset for the AXI4-Stream Transmit data interface
AXI_STR_TXD_TVALID	O	Transmit data channel valid
AXI_STR_TXD_TREADY	I	Transmit data channel ready
AXI_STR_TXD_TLAST	O	Transmit data channel last word
AXI_STR_TXD_TKEEP (C_SAXI_DATA_WIDTH/8-1:0)	O	Transmit data channel byte strobes
AXI_STR_TXD_TDEST(3:0)	O	Transmit data channel destination
AXI_STR_TXD_TDATA(C_SAXI_DATA_WIDTH-1:0)	O	Transmit data channel data
<b>AXI4-Stream Transmit Control Channel Signals</b>		
AXI_STR_TXC_ACLK	I	Clock for the AXI4-Stream Transmit Control interface
mm2s_cntrl_reset_out_n	O	Reset for the AXI4-Stream Transmit Control interface
AXI_STR_TXC_TVALID	O	Transmit Control channel valid
AXI_STR_TXC_TREADY	I	Transmit Control channel ready
AXI_STR_TXC_TLAST	O	Transmit Control channel last word

Table 1: (Cont'd) I/O Signals (Cont'd)

Signal Name	Direction	Description
AXI_STR_TXC_T_KEEP (C_SAXI_DATA_WIDTH/8-1:0)	O	Transmit Control channel byte strobes
AXI_STR_TXC_TDATA (C_SAXI_DATA_WIDTH-1:0)	O	Transmit Control channel control
<b>AXI4-Stream Receive Data Channel Signals</b>		
AXI_STR_RXD_ACLK	I	Clock for the AXI4-Stream Receive Data interface
s2mm_prmry_reset_out_n	O	Reset for the AXI4-Stream Receive Data interface
AXI_STR_RXD_TVALID	I	Receive Data channel valid
AXI_STR_RXD_TREADY	O	Receive Data channel ready
AXI_STR_RXD_TLAST	I	Receive Data channel last word
AXI_STR_RXD_TKEEP (C_SAXI_DATA_WIDTH/8-1:0)	I	Receive Data channel byte strobes
AXI_STR_RXD_RDEST(3:0)	I	Receive data channel destination
AXI_STR_RXD_TDATA (C_SAXI_DATA_WIDTH-1:0)	I	Receive Data channel data

## Design Parameters

To allow the user to generate a AXI4-Stream FIFO core that is tailored for their system, certain features are parameterizable, thereby resulting in a design that utilizes only the resources required by their system and runs at the best possible performance. The features that are parameterizable in the AXI4-Stream FIFO core design are shown in [Table 2](#).

## Inferred Parameters

In addition to the parameters listed in [Table 2](#), there are parameters that are inferred for each AXI interface in the EDK tools. Through the design, these EDK-inferred parameters control the behavior of the AXI Interconnect. For a complete list of the interconnect settings related to the AXI interface, see the [DS768 AXI Interconnect Specification Data Sheet](#).

Table 2: Design Parameters

Feature/Description	Parameter Name	Allowable Values	Default Values	VHDL Type
Device family	C_FAMILY	zynq7000, artix7, virtex7, kintex7, virtex6, spartan6	virtex7	string
Width of all ID signals across the interface. This is auto computed by the tools	C_SAXI_ID_WIDTH	Integer	4	Integer
Address width of the interface. Must be 32.	C_SAXI_ADDR_WIDTH	32	32	Integer
Data width of the interface.	C_SAXI_DATA_WIDTH	32	32	Integer
Base address of the IP <sup>(1)</sup>	C_SAXI_BASEADDR	0x0 - 0xfffffeff	0xfffffeff	Bit32
High address of the IP <sup>(1)</sup>	C_SAXI_HIGHADDR	0x00000fff - 0xffffffff	0xffffffff	Bit32
TX FIFO Depth. FIFO size is C_TX_FIFO_DEPTH*C_SAXI_DATA_WIDTH/8	C_TX_FIFO_DEPTH	512, 1024, 2048, 4096	512	Integer

Table 2: (Cont'd) Design Parameters (Cont'd)

Feature/Description	Parameter Name	Allowable Values	Default Values	VHDL Type
RX FIFO Depth. FIFO size is $C\_RX\_FIFO\_DEPTH \times C\_S\_AXI\_DATA\_WIDTH/8$	C_RX_FIFO_DEPTH	512, 1024, 2048, 4096	512	Integer
TX FIFO Programmable FULL threshold value.	C_TX_FIFO_PF_THRESHOLD	10 - $C\_TX\_FIFO\_DEPTH-2$	510	Integer
TX FIFO Programmable EMPTY threshold value.	C_TX_FIFO_PE_THRESHOLD	2 - $C\_TX\_FIFO\_DEPTH-10$	2	Integer
RX FIFO Programmable FULL threshold value.	C_RX_FIFO_PF_THRESHOLD	10 - $C\_RX\_FIFO\_DEPTH-2$	510	Integer
RX FIFO Programmable EMPTY threshold value.	C_TX_FIFO_PE_THRESHOLD	2 - $C\_TX\_FIFO\_DEPTH-10$	2	Integer

**Notes:**

- HIGHADDR is required to be at least BASEADDR + 4095 because AXI requires a core to have a minimum 4k address space. For example: C\_SAXI\_BASEADDR = 0x00000000 and C\_SAXI\_HIGHADDR = 0x00000FFF. The range specified by BASEADDR and HIGHADDR must be a power of 2 in size, and must have BASEADDR aligned to the size.

Figure 2 shows the AXI4-Stream FIFO core connected to the AXI Ethernet core.

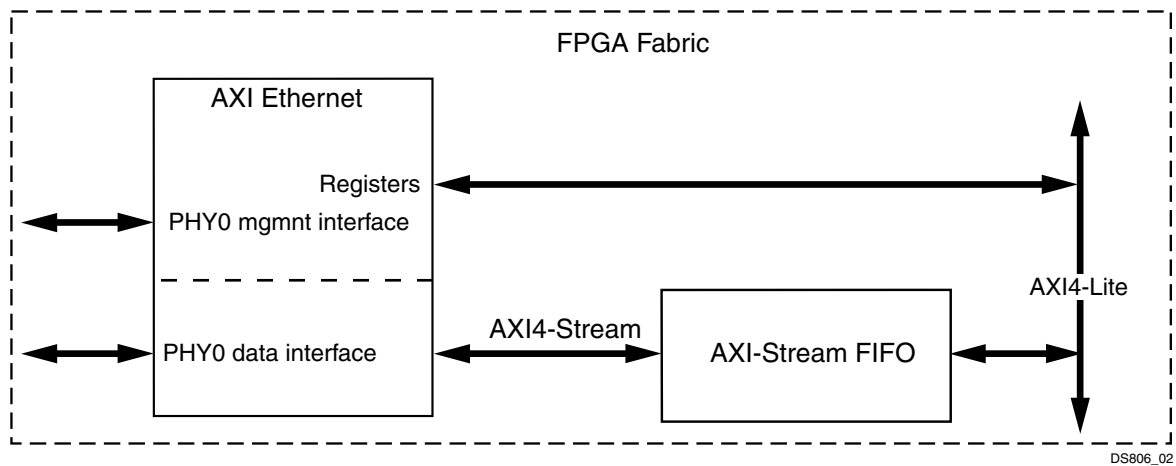


Figure 2: AXI4-Stream FIFO Connected to an AXI Ethernet Core

Figure 3 shows a partial code segment from an EDK MHS file. The file shows an example connection between AXI4-Stream FIFO cores and an AXI Ethernet core.

```
BEGIN axi_fifo_mm_s
  PARAMETER INSTANCE = ETHERNET_fifo
  PARAMETER HW_VER = 2.01.a
  PARAMETER C_INTERCONNECT_S_AXI_MASTERS = microblaze_0.M_AXI_DP
  PARAMETER C_BASEADDR = 0x71a00000
  PARAMETER C_HIGHADDR = 0x71a0ffff
  BUS_INTERFACE S_AXI = axi_interconnect_memory_mapped_lite_0
  BUS_INTERFACE AXI_STR_RXD = ETHERNET_fifo_rxd
  BUS_INTERFACE AXI_STR_TXD = ETHERNET_fifo_txd
  BUS_INTERFACE AXI_STR_TXC = ETHERNET_fifo_txc
  PORT S_AXI_ACLK = clk_100_0000MHzMMCM0
  PORT INTERRUPT = axi_fifo_INTERRUPT
  PORT AXI_STR_TXD_ACLK = clk_100_0000MHzMMCM0
  PORT AXI_STR_TXC_ACLK = clk_100_0000MHzMMCM0
  PORT AXI_STR_RXD_ACLK = clk_100_0000MHzMMCM0
END

BEGIN axi_ethernet
...
  BUS_INTERFACE S_AXI = axi_interconnect_memory_mapped_lite_0
  BUS_INTERFACE AXI_STR_RXD = ETHERNET_fifo_rxd
  BUS_INTERFACE AXI_STR_TXD = ETHERNET_fifo_txd
  BUS_INTERFACE AXI_STR_TXC = ETHERNET_fifo_txc
...
  PORT PHY_RST_N = ETHERNET_PHY_RST_N
  PORT GTX_CLK = clk_125_0000MHz
  PORT INTERRUPT = ETHERNET_INTERRUPT
  PORT AXI_STR_RXS_TREADY = net_vcc
  PORT AXI_STR_TXC_ACLK = clk_100_0000MHzMMCM0
  PORT AXI_STR_TXD_ACLK = clk_100_0000MHzMMCM0
  PORT AXI_STR_RXS_ACLK = clk_100_0000MHzMMCM0
  PORT AXI_STR_RXD_ACLK = clk_100_0000MHzMMCM0
END
```

DS806\_03

Figure 3: EDK MHS File Code Segment

## Registers Definition

The AXI4-Stream FIFO core contains the registers listed in [Table 3](#).

**Table 3: Register Names and Descriptions**

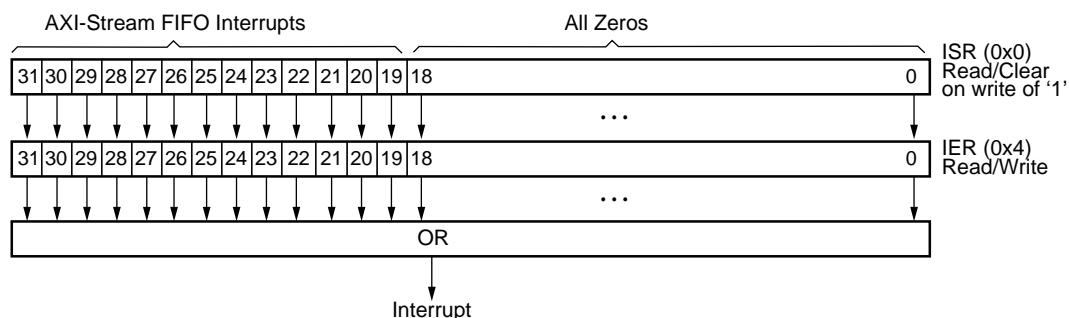
Register Name	AXI Address	Access
Interrupt Status Register (ISR)	C_SAXI_BASEADDR + 0x0	Read/Clear on Write <sup>(1)</sup>
Interrupt Enable Register (IER))	C_SAXI_BASEADDR + 0x4	Read/Write
Transmit data FIFO reset (TDFR)	C_SAXI_BASEADDR + 0x8	Write <sup>(2)</sup>
Transmit data FIFO Vacancy (TDFV)	C_SAXI_BASEADDR + 0xC	Read
Transmit data FIFO 32bit wide data write port (TDFD)	C_SAXI_BASEADDR + 0x10	Write
Transmit Length FIFO (TLF)	C_SAXI_BASEADDR + 0x14	Write
Receive data FIFO reset (RDFR)	C_SAXI_BASEADDR + 0x18	Write <sup>(2)</sup>
Receive data FIFO Occupancy (RDFO)	C_SAXI_BASEADDR + 0x1C	Read
Receive data FIFO 32bit wide data read port (RDFD)	C_SAXI_BASEADDR + 0x20	Read
Receive Length FIFO (RLF)	C_SAXI_BASEADDR + 0x24	Read
AXI4-Stream reset (SRR)	C_SAXI_BASEADDR + 0x28	Write <sup>(2)</sup>
Transmit Destination Register (TDR)	C_SAXI_BASEADDR + 0x2C	Write
Receive Destination Register (RDR)	C_SAXI_BASEADDR + 0x30	Read
Reserved	C_SAXI_BASEADDR + 0x34 to C_SAXI_BASEADDR + 0x3C	N/A <sup>(3)</sup>

### Notes:

1. The latched interruptible condition is cleared by writing a 1 to that bit location. Writing a 1 to a bit location that is 0 has no effect. Likewise, writing a 0 to a bit location that is 1 has no effect. Multiple bits may be cleared in a single write.
2. Reset if written with 0xA5.
3. If read, these registers will return 0x0. Writing these registers will have no effect.

## Interrupt Interface

The interrupt signals generated by the AXI4-Stream FIFO core are managed by the ISR and IER registers. The ISR is combined with the IER register to define the interrupt interface of the AXI4-Stream FIFO core. An overview diagram of the interrupt control structure is show in [Figure 4](#).



**Figure 4: Interrupt Control Structure**

### Interrupt Status Register (ISR)

The Interrupt Status Register is shown in [Figure 5](#). The Interrupt Status register uses one bit to represent each internal interruptible condition.

Once an interruptible condition occurs, it will be captured in this register (represented as the corresponding bit being set to 1) even if the condition changes. The latched interruptible condition is cleared by writing a 1 to its bit location. Writing a 1 to a bit location that is 0 has no effect. Likewise, writing a 0 to a bit location that is 1 has no effect. Multiple bits may be cleared in a single write.

For any bit set in the Interrupt Status Register, a corresponding bit must be also set in the Interrupt Enable Register for the IP2INTC\_Irpt signal to be driven active High out of the AXI4-Stream FIFO core.

The Interrupt Status Register bit definitions are detailed in [Table 4](#).

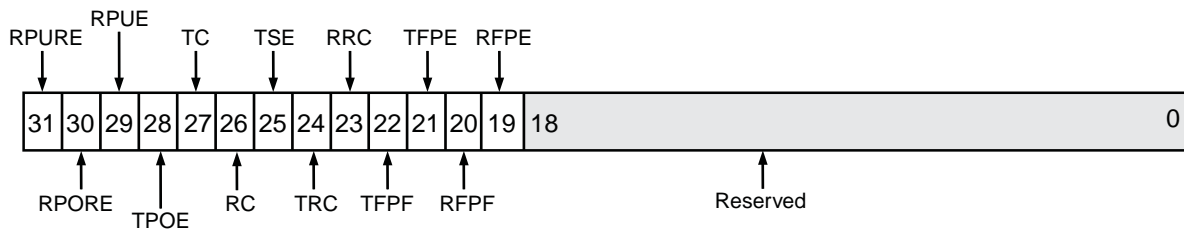


Figure 5: Interrupt Status Register (offset 0x0)

Table 4: Interrupt Status Register Bit Definitions

Bit(s)	Name	Core Access	Reset Value	Description
0-18	Reserved	Read	0x0	<b>Reserved:</b> These bits are reserved for future definition and will always return all zeros.
19	RFPE	Read/Clear on Write of "1"	0	<b>Receive FIFO Programmable Empty:</b> Generated when the difference of read and write pointers of receive FIFO reaches the programmable EMPTY threshold value. '0' = No interrupt pending '1' = Interrupt pending
20	RFPF	Read/Clear on Write of "1"	0	<b>Receive FIFO Programmable Full:</b> This interrupt is generated when the difference of read and write pointers of receive FIFO reaches the programmable FULL threshold value. '0' = No interrupt pending '1' = Interrupt pending
21	TFPE	Read/Clear on Write of "1"	0	<b>Transmit FIFO Programmable Empty:</b> This interrupt is generated when the difference of read and write pointers of transmit FIFO reaches the programmable EMPTY threshold value. '0' = No interrupt pending '1' = Interrupt pending
22	TFPF	Read/Clear on Write of "1"	0	<b>Transmit FIFO Programmable Full:</b> This interrupt is generated when the difference of read and write pointers of transmit FIFO reaches the programmable FULL threshold value. '0' = No interrupt pending '1' = Interrupt pending
23	RRC	Read/Clear on Write of "1"	0	<b>Receive Reset Complete:</b> This interrupt indicates that a reset of the receive logic has completed. "0" = No interrupt pending. "1" = Interrupt pending.
24	TRC	Read/Clear on Write of "1"	0	<b>Transmit Reset Complete:</b> This interrupt indicates that a reset of the transmit logic has completed. "0" = No interrupt pending. "1" = Interrupt pending.



Table 4: (Cont'd) Interrupt Status Register Bit Definitions (Cont'd)

Bit(s)	Name	Core Access	Reset Value	Description
25	TSE	Read/Clear on Write of "1"	0	<b>Transmit Size Error:</b> This interrupt is generated if the number of 32-bit words (including partial words in the count) written to the transmit data FIFO does not match the value written to the transmit length register (bytes) divided by 4 and rounded up to the higher integer value for trailing byte fractions. Interrupts occur only for mismatch of word count (including partial words). Interrupts do not occur due to mismatch of byte count. "0" = No interrupt pending. "1" = Interrupt pending.
26	RC	Read/Clear on Write of "1"	0	<b>Receive Complete:</b> Indicates that at least one successful receive has completed and that the receive packet data and packet data length is available. This signal is not set for unsuccessful receives. This interrupt may represent more than one packet received, so it is important to check the receive data FIFO occupancy value to determine if additional receive packets are ready to be processed. "0" = No interrupt pending. "1" = Interrupt pending.
27	TC	Read/Clear on Write of "1"	0	<b>Transmit Complete:</b> Indicates that at least one transmit has completed. "0" = No interrupt pending. "1" = Interrupt pending.
28	TPOE	Read/Clear on Write of "1"	0	<b>Transmit Packet Overrun Error:</b> This interrupt is generated if an attempt is made to write to the transmit data FIFO when it is full. A reset of the transmit logic is required to recover. "0" = No interrupt pending. "1" = Interrupt pending.
29	RPUE	Read/Clear on Write of "1"	0	<b>Receive Packet Underrun Error:</b> This interrupt occurs when an attempt is made to read the receive FIFO when it is empty. The data read is not valid. A reset of the receive logic is required to recover. "0" = No interrupt pending. "1" = Interrupt pending.
30	RPORE	Read/Clear on Write of "1"	0	<b>Receive Packet Overrun Read Error:</b> This interrupt occurs when more words are read from the receive data FIFO than are in the packet being processed. Even though the FIFO may not be empty, the read has gone beyond the current packet and removed data from the next packet. A reset of the receive logic is required to recover. "0" = No interrupt pending. "1" = Interrupt pending.
31	RPURE	Read/Clear on Write of "1"	0	<b>Receive Packet Underrun Read Error:</b> This interrupt occurs when an attempt is made to read the receive length register when it is empty. The data read is not valid. A reset of the receive logic is required to recover. "0" = No interrupt pending. "1" = Interrupt pending.

## Interrupt Enable Register (IER)

The Interrupt Enable Register shown in Figure 6 determines which interrupt sources in the Interrupt Status Register are allowed to generate interrupts. Setting to "1" in a bit location enables the related interrupt from being propagated, while a value of "0" disables it.

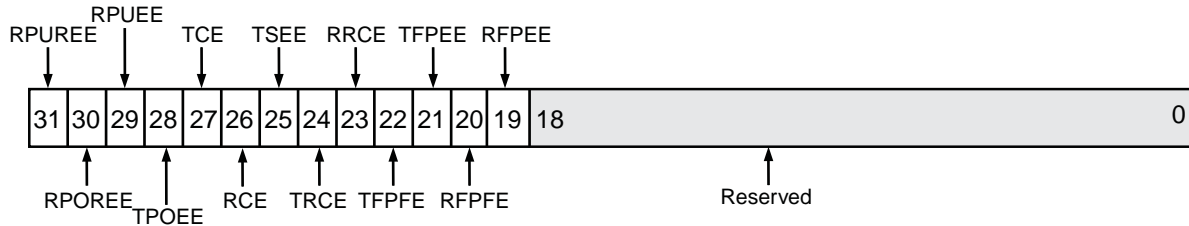
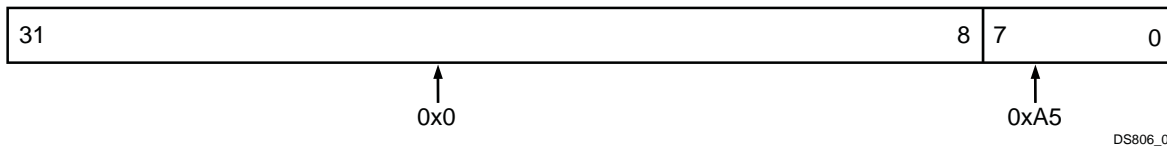


Figure 6: Interrupt Enable Register (offset 0x4)

## Transmit Data FIFO Reset Register (TDFR)

The Transmit Data FIFO Reset Register shown in Figure 7 is not an actual register, but is instead a write-only address, which when written with a specific value, generates a reset for the Transmit Data FIFO. This reset will not occur until transmit activity on the TX AXI Stream has completed. The reset can occur only during inactive times on the TX AXI Stream and will affect only the transmit circuitry in this core, thereby preventing the core on the other end of the AXI4-Stream from receiving a partial packet which could potentially cause a failure condition in the latter core.

Because of this mode of operation, it is possible that if the AXI4-Stream becomes unresponsive during an AXI4-Stream transaction, a reset will never occur. For example, this might occur while waiting for the destination ready to go active in the middle of a transfer. In such cases it is necessary to use both the AXI4-Stream Reset and the S\_AXI\_ARESETN reset.



DS806\_07

Figure 7: Transmit Data FIFO Reset Register (offset 0x8)

Table 5: Transmit Data FIFO Reset Register Bit Definitions

Bit(s)	Name	Core Access	Reset Value	Description
31-0	Reset Key	Write	N/A	<b>Reset Write Value.</b> "0x000000A5" - Generate a reset. Others - No effect.

## Transmit Data FIFO Vacancy Register (TDFV)

The Transmit Data FIFO Vacancy Register shown in Figure 8 is a read-only register that gives the vacancy status of the Transmit Data FIFO.

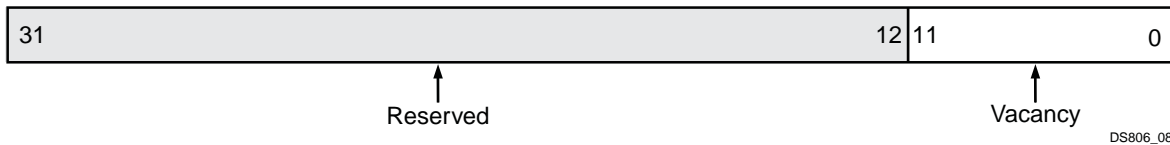


Figure 8: Transmit Data FIFO Vacancy Register (offset 0xC)

## Transmit Data FIFO Data Write Port (TDFD)

The Transmit Data FIFO Data Write Port shown in Figure 9 is a 32-bit wide address location for writing data into the Transmit Data FIFO. The smallest packet that may be transmitted is four 32-bit words (include partial final word) which corresponds to 13 to 16 bytes. The maximum packet that may be transmitted is limited by the size of the FIFO which is 510 (include partial final word) words or 2037 to 2040 bytes.

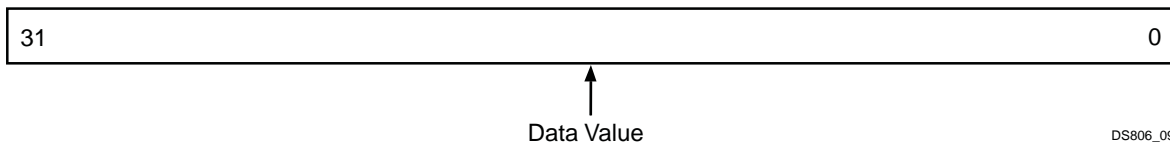


Figure 9: Transmit Data FIFO Data Write Port (offset 0x10)

Table 6: Transmit Data FIFO Data Write Port Bit Definitions

Bit(s)	Name	Core Access	Reset Value	Description
31-0	Write Data Value	Write	N/A	Transmit Data FIFO Write Value.

## Receive Data FIFO Reset Register (RDFR)

The Receive Data FIFO Reset Register shown in Figure 10 is not an actual register but, rather a write-only address, which when written with a specific value, generates a reset for the Receive Data FIFO.

This reset will not occur until receive activity on the RX AXI Stream has completed. Only during inactive times on the RX AXI Stream can a reset occur. It will affect only the receive circuitry in this core. This prevents the core on the other end of the AXI4-Stream from transmitting a partial packet which may cause failure condition in that core.

Because of this mode of operation, it is possible that if the AXI4-Stream interface becomes unresponsive during an AXI4-Stream transaction, that the reset will never occur. An example transaction is if a packet is received over the AXI4-Stream that exceeds the FIFO size of this core causing the core's destination ready to become inactive in the middle of a transfer. In this case, an S\_AXI\_ARESETN reset is needed.

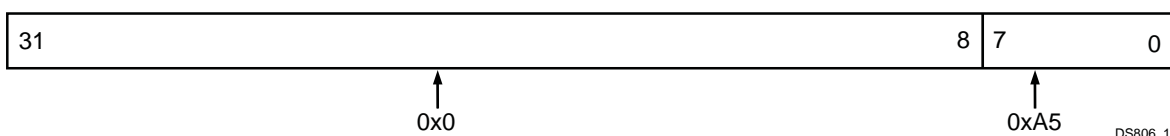


Figure 10: Receive Data FIFO Reset Register (offset 0x18)

Table 7: Receive Data FIFO Reset Register Bit Definitions

Bit(s)	Name	Core Access	Reset Value	Description
31-0	Reset Key	Write	N/A	<b>Reset Write Value:</b> "0x000000A5" - Generate a reset. Others - No effect.

## Receive Data FIFO Occupancy Register (RDFO)

The Receive Data FIFO Occupancy Register shown in Figure 11 is a read-only register that gives the occupancy status of the Receive Data FIFO.

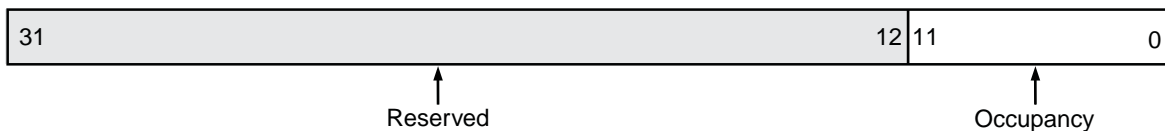


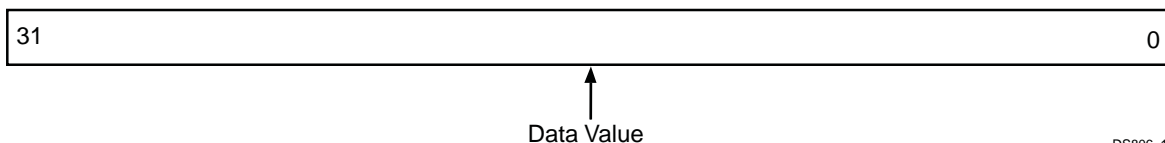
Figure 11: Receive Data FIFO Occupancy Register (offset 0x1C)

Table 8: Receive Data FIFO Occupancy Register Bit Definitions

Bit(s)	Name	Core Access	Reset Value	Description
11-0	Occupancy	Read	0x0	<b>Receive Data FIFO Occupancy:</b> This is the unsigned value reflecting a current snapshot of the number of 32-bit wide locations in use for data storage in the receive Data FIFO memory core. This value is only updated after a packet is successfully received, and therefore can be used to determine if a receive packet is ready to be processed when a non-0 value is read.
31-12	Reserved	Read	0x0	<b>Reserved:</b> These bits are reserved for future definition and will always return all zeros.

## Receive Data FIFO Data Read Port (RDFO)

The Receive Data FIFO Data Read Port shown in Figure 12 is a 32-bit wide address location for reading data from the Receive Data FIFO. The smallest packet that may be received is a four 32-bit words packet which corresponds to 13 to 16 bytes. The maximum packet that may be received is limited by the size of the FIFO which is 510 words or 2037 to 2040 bytes.



DS806\_12

Figure 12: Receive Data FIFO Data Read Port (offset 0x20)

Table 9: Receive Data FIFO Data Read Port Bit Definitions

Bit(s)	Name	Core Access	Reset Value	Description
31-0	Read Data Value	Read	N/A	Receive Data FIFO Read Value.

## Transmit Length Register (TLR)

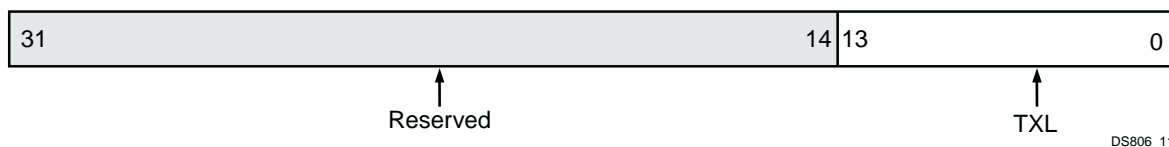
The Transmit Length Register shown in Figure 13 is used to store packet length values (the number of bytes in the packet) corresponding to valid packets ready for transmit. The data for the packet is stored in the transmit Data

FIFO. The data is written to the AXI4-Stream FIFO core over the AXI4 interface, typically by a processor or DMA core such the Central DMA (CDMA). When presenting a transmit packet to the AXI4-Stream FIFO core, write the packet data to the Transmit Data FIFO first, then write the length of the packet into the TLR.

Once the packet length is written to the TLR, it is automatically moved to the Transmit Data FIFO with the packet data freeing up the TLR for another value. The packet length must be written to the TLR after the packet data is written to the transmit data FIFO. It is **not** valid to write data for multiple packets to the transmit data FIFO before writing the packet length values.

The action of writing to the Transmit Length Register is used by the AXI4-Stream FIFO core to initiate the processing of transmit packets across the AXI4-Stream interface. This action continues until all of the TLR values that have been stored are processed.

The width of the TLR is wide enough to support packets up to 2048 bytes in length. The smallest packet that may be transmitted is four 32-bit words (include partial final word) which corresponds to 13 to 16 bytes. The maximum packet that may be transmitted is limited by the size of the FIFO which is 510 (include partial final word) words or 2037 to 2040 bytes.



DS806\_11

Figure 13: Transmit Length Register (offset 0x14)

Table 10: Transmit Length Register Bit Definitions

Bit(s)	Name	Core Access	Reset Value	Description
31-14	Reserved	N/A	0x0	<b>Reserved:</b> These bits are reserved for future definition and will always return all zeros.
13-0	TXL	Write	0x0	<b>Transmit Length:</b> The number of bytes of the corresponding transmit packet stored in the transmit data FIFO.

## Receive Length Register (RLR)

The receive length register shown in Figure 14 is used to retrieve packet length values (the number of bytes in the packet) corresponding to valid packets received. The data for the packet is stored in the Receive Data FIFO.

The length is written by the AXI4-Stream FIFO core when the packet is received across the RX AXI4-Stream interface and the receive data FIFO has the adequate number of locations to ensure that all of the packet data has been saved.

The RLR should only be read when a receive packet is available for processing (the receive occupancy is not zero). Once the RLR is read, the receive packet data should be read from the receive data FIFO before the RLR is read again.

The RLR values are stored in the receive data FIFO by the AXI4-Stream FIFO core with the data of each packet. The RLR value for the subsequent packet to be processed is moved to the RLR when the previous RLR value has been read.

This register is wide enough to support packets up to 2048 bytes in length. The smallest packet that may be received is four 32-bit words (include partial final word) which corresponds to 13 to 16 bytes. The maximum packet that may be received is limited by the size of the FIFO and is 510 words (include partial final word) or 2037 to 2040 bytes.

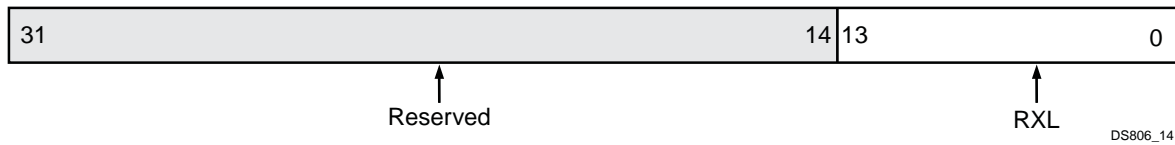


Figure 14: Receive Length Register (offset 0x24)

Table 11: Receive Length Register Bit Definitions

Bit(s)	Name	Core Access	Reset Value	Description
31-14	Reserved	Read	0x0	<b>Reserved:</b> These bits are reserved for future definition and will always return all zeros.
13-0	RXL	Read	0x0	<b>Receive Length:</b> The number of bytes of the corresponding receive data stored in the receive data FIFO.

## AXI4-Stream Reset Register (SRR)

The AXI4-Stream Register shown in Figure 15 is not an actual register. It is a write-only address, which when written with a specific value, generates an immediate reset for the entire core as well as driving a reset on the external outputs, s2mm\_prmry\_reset\_out\_n, mm2s\_prmry\_reset\_out\_n, and mm2s\_cntrl\_reset\_out\_n, which can be used to reset the core on the other end of the AXI4-Stream.

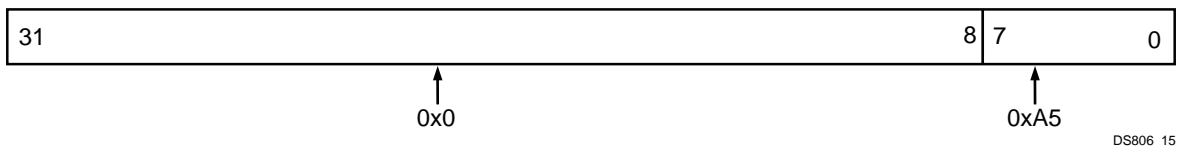


Figure 15: AXI4-Stream Reset Register (offset 0x28)

Table 12: AXI4-Stream Reset Register Bit Definitions

Bit(s)	Name	Core Access	Reset Value	Description
31-0	Reset Key	Write	N/A	Reset Write Value: "0x000000A5" - Generate a reset. Others - No effect.

## Transmit Destination Register (TDR)

The Transmit Destination Register shown in Figure 16 stores the destination address corresponding to the packet to be transmitted. When presenting a transmit packet to the AXI4-Stream FIFO core, write the destination address into TDR first, write the packet data to the Transmit Data FIFO next, and then write the length of the packet into the TLR.

The destination address must be written to the TDR before the packet data is written to the transmit data FIFO. Writing data for multiple packets to the transmit data FIFO before writing the destination address values is not a valid sequence.

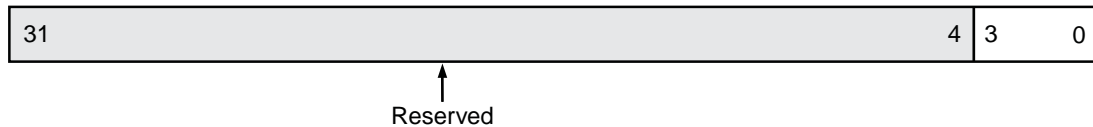


Figure 16: Transmit Destination Register (offset 0x2C)

Table 13: Transmit Destination Register Bit Definitions

Bit(s)	Name	Core Access	Reset Value	Description
31-4	Reserved	N/A	0x0	<b>Reserved:</b> These bits are reserved for future definition and will always return all zeros.
3-0	TDEST	Write	0x0	<b>Transmit Destination:</b> The destination address of the transmit packet stored in the transmit data FIFO.

## Receive Destination Register (RDR)

The Receive Destination Register shown in Figure 17 retrieves the destination address corresponding to the valid packet received.

The RDR should only be read when a receive packet is available for processing (the receive occupancy is not zero). Once the RDR is read, the receive packet data should be read from the receive data FIFO before the RDR is read again. The RDR values are stored in the receive data FIFO by the AXI4-Stream FIFO core with the data of each packet. The RDR value for the subsequent packet to be processed is moved to the RDR when the previous RDR value has been read.

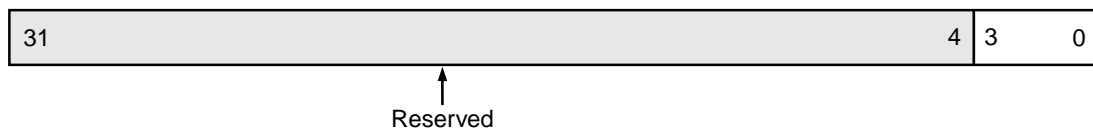


Figure 17: Receive Destination Register (offset 0x30)

Table 14: Receive Destination Register Bit Definitions

Bit(s)	Name	Core Access	Reset Value	Description
31-4	Reserved	N/A	0x0	<b>Reserved:</b> These bits are reserved for future definition and will always return all zeros.
3-0	RDEST	Read	0x0	<b>Receive Destination:</b> The destination address of the receive packet stored in the receive data FIFO.

## Reserved Registers

Reading from reserved registers will return zeros and writing to reserved registers will have no effect. However, any accesses to address offset 0x40 and above causes undefined results.

## Basic Usage

The AXI4-Stream FIFO core was designed to provide memory-mapped access to an AXI4-Stream interface connected to other IP, such as the AXI Ethernet core. Systems must be built through the Embedded Development Kit (EDK) to attach the AXI4-Stream FIFO core, AXI Ethernet core, processor, memory, interconnect the buses, clocking, and additional embedded components.

This section briefly describes the operation of the AXI4-Stream FIFO core through register accesses using the AXI Ethernet core as an example.

Packets will automatically be transmitted by writing packet data to the transmit data FIFO followed by writing a length in the TLR.

Receiving packets functions in a manner similar to transmission, except that the steps are reversed. Receiving the ISR receive complete interrupt or polling of the receive data FIFO occupancy register (when not zero) indicates the reception of a packet. Reading the RLR provides the packet length in bytes. Given the number of received packet bytes, the appropriate number of reads of the Read data FIFO will provide the packet data.

Table 15 illustrates a power-up read of the registers followed by a transmission and reception of a single packet. See the register definitions for further information and options.

Table 15: Sample TX and RX Usage

Register	Access	Value	Activity
<b>Power-up Read of Register Values</b>			
ISR	Read Word	0x01800000	Read interrupt status register
ISR	Write Word	0xFFFFFFFF	Write to clear reset done interrupt bits
ISR	Read Word	0x00000000	Read interrupt status register
IER	Read Word	0x00000000	Read interrupt enable register
TDFV	Read Word	0x000001FE	Read the transmit FIFO vacancy
RDFO	Read Word	0x00000000	Read the receive FIFO occupancy
TDR	Write Word	0x00000002	Transmit Destination address (0x2 = destination device address is 2)
<b>Transmit a Packet</b>			
TXFIFO_DATA	Write Word	0xFFFFFFFF	4 bytes of data
TXFIFO_DATA	Write Word	0x12345678	4 bytes of data
TXFIFO_DATA	Write Word	0x00010203	4 bytes of data
TXFIFO_DATA	Write Word	0x08090A0B	4 bytes of data
TXFIFO_DATA	Write Word	0x10111213	4 bytes of data
TXFIFO_DATA	Write Word	0x18191A1B	4 bytes of data
TXFIFO_DATA	Write Word	0x20212223	4 bytes of data
TXFIFO_DATA	Write Word	0x28292A2B	4 bytes of data
TDFV	Read Word	0x000001F6	Read the transmit FIFO vacancy
TLR	Write Word	0x00000020	Transmit length (0x20 = 32bytes), this starts transmission
ISR	Read Word	0x08000000	A typical value after Tx Complete is indicated by interrupt
ISR	Write Word	0xFFFFFFFF	Write to clear reset done interrupt bits
ISR	Read Word	0x00000000	Read interrupt status register
TDFV	Read Word	0x000001FE	Read the transmit FIFO vacancy
<b>Receive a Packet</b>			
ISR	Read Word	0x04000000	A typical value after Rx Complete is indicated by interrupt
ISR	Write Word	0xFFFFFFFF	Write to clear reset done interrupt bits
ISR	Read Word	0x00000000	Read interrupt status register
RDFO	Read Word	0x00000008	Read the receive FIFO occupancy



Table 15: Sample TX and RX Usage (Cont'd)

Register	Access	Value	Activity
RLR	Read Word	0x00000020	Receive length (0x20 =32 bytes) indicates number of bytes to read
RDR	Read Word	0x00000002	Receive Destination address (0x2 = destination device address is 2)
RDFO	Read Word	0x00000008	Read the receive FIFO occupancy
RXFIFO_DATA	Read Word	0xFFFFFFFF	4 bytes of data
RXFIFO_DATA	Read Word	0x12345678	4 bytes of data
RXFIFO_DATA	Read Word	0x00010203	4 bytes of data
RXFIFO_DATA	Read Word	0x08090A0B	4 bytes of data
RXFIFO_DATA	Read Word	0x10111213	4 bytes of data
RXFIFO_DATA	Read Word	0x18191A1B	4 bytes of data
RXFIFO_DATA	Read Word	0x20212223	4 bytes of data
RXFIFO_DATA	Read Word	0x28292A2B	4 bytes of packet data and CRC value
RDFO	Read Word	0x00000000	Read the receive FIFO occupancy (no further receive packets to process)

## Design Implementation

### Design Tools

The AXI4-Stream FIFO core design is implemented using VHDL code. Xilinx XST is the synthesis tool used for synthesizing the core.

### Target Technology

The target technology is an FPGA listed in the supported device family field of the [LogiCORE IP Facts Table](#).

### Device Utilization and Performance Benchmarks

Because the AXI4-Stream FIFO core will be used with other design pieces in the FPGA, the utilization and timing numbers reported in this section are just estimates. As the AXI4-Stream FIFO core is combined with other pieces of the FPGA design, the utilization of FPGA resources and timing of the AXI4-Stream FIFO core design will vary from the results reported here. The core benchmarks are shown in [Table 16](#) for a Virtex-6 FPGA and in [Table 17](#) for the Spartan-6 FPGA.

Table 16: Performance and Resource Utilization Benchmarks for the Virtex-6 FPGA

Device Resources				F <sub>MAX</sub> (MHz)
Block RAMs	Slices	Slice Flip- Flops	LUTs	F <sub>MAX</sub>
2	452	957	876	150

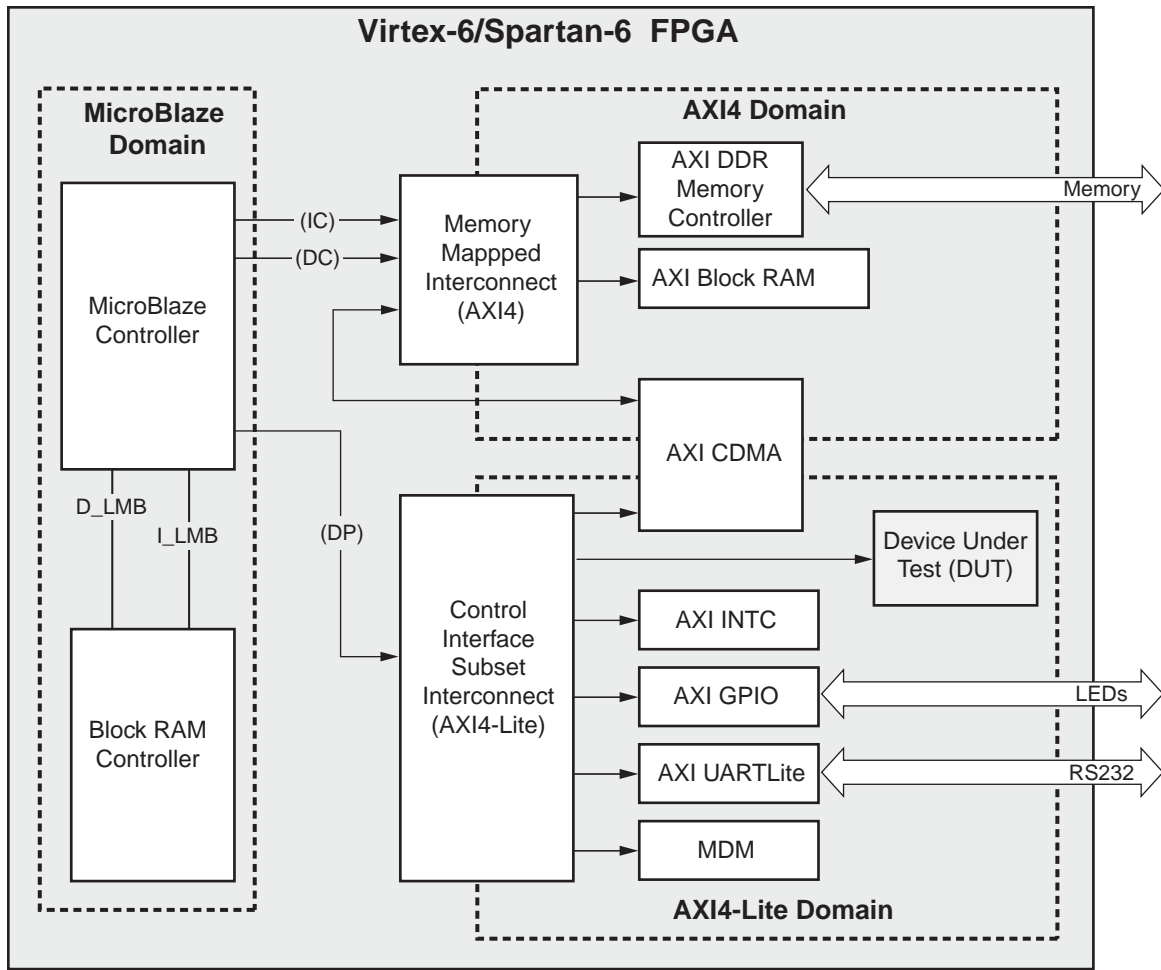
Table 17: Performance and Resource Utilization Benchmarks for the Spartan-6 FPGA

Device Resources				F <sub>MAX</sub> (MHz)
Block RAMs	Slices	Slice Flip- Flops	LUTs	F <sub>MAX</sub>
2	452	957	876	100

## System Performance

To measure the system performance ( $F_{MAX}$ ) of the AXI4-Stream FIFO core, it was added as the Device Under Test (DUT) to a Virtex-6 FPGA system as shown in [Figure 18](#).

Because the core is used with other design modules in the FPGA, the utilization and timing numbers reported in this section are estimates only. When this core is combined with other designs in the system, the utilization of FPGA resources and timing of the design will vary from the results reported here.



DS790\_13

Figure 18: Virtex-6 and Spartan-6 FPGA System with the AXI4-Stream FIFO core as the DUT

The target FPGA was then filled with logic to drive the LUT and BRAM utilization to approximately 70% and the I/O utilization to approximately 80%. Using the default tool options and the slowest speed grade for the target FPGA, the resulting target  $F_{MAX}$  numbers are shown in Table 18.

Table 18: System Performance

Target FPGA	Target $F_{MAX}$ (MHz)		
	AXI4	AXI4-Lite	MicroBlaze
xc6slx45t <sup>(1)</sup>	90 MHz	120 MHz	80
xc6vlx240t <sup>(2)</sup>	135 MHz	180 MHz	135

**Notes:**

1. Spartan-6 LUT utilization: 60%; BRAM utilization: 70%; IO utilization: 80%; MicroBlaze not AXI4 interconnect; AXI4 interconnect configured with a single clock of 120MHz.
2. Virtex-6 LUT utilization: 70%; BRAM utilization: 70%; IO utilization: 80%.

The target  $F_{MAX}$  is influenced by the exact system and is provided for guidance. It is not a guaranteed value across all systems.

## Reference Documents

1. DS562, *AXI Slave Burst*
2. DS160, *Spartan-6 Family Overview*
3. DS150, *Virtex-6 Family Overview*
4. AXI4 AMBA Protocol Version 2.0 Specification
5. DS180, *7 Series FPGAs Overview*

## 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 at no additional cost with the Xilinx Vivado Design Suite and ISE Design Suite Embedded Edition software under the terms of the [Xilinx End User License](#).

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](#).

## Revision History

Date	Version	Description of Revisions
9/21/10	1.0	Xilinx initial release.
3/1/11	1.1	Updated to v2.00a for the 13.1 release.
6/22/11	1.2	Updated for the 13.2 release; added 7 series family support.
7/25/12	2.0	Added support for Vivado Design Suite. Added <a href="#">Transmit Destination Register (TDR)</a> and <a href="#">Receive Destination Register (RDR)</a> . Added FIFO depth and threshold parameters to <a href="#">Table 2</a> .

## Notice of Disclaimer

Xilinx is providing this product documentation, hereinafter "Information," to you "AS IS" with no warranty of any kind, express or implied. Xilinx makes no representation that the Information, or any particular implementation thereof, is free from any claims of infringement. You are responsible for obtaining any rights you may require for any implementation based on the Information. All specifications are subject to change without notice. XILINX EXPRESSLY DISCLAIMS ANY WARRANTY WHATSOEVER WITH RESPECT TO THE ADEQUACY OF THE INFORMATION OR ANY IMPLEMENTATION BASED THEREON, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OR REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE FROM CLAIMS OF INFRINGEMENT AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Except as stated herein, none of the Information may be copied, reproduced, distributed, republished, downloaded, displayed, posted, or transmitted in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx.