

# Mailbox v2.1

## *LogiCORE IP Product Guide*

**Vivado Design Suite**

**PG114 October 4, 2017**

# Table of Contents

## IP Facts

### Chapter 1: Overview

Feature Summary .....	6
Licensing and Ordering .....	6

### Chapter 2: Product Specification

Standards .....	7
Performance .....	7
Resource Utilization .....	8
Port Descriptions .....	8
Register Space .....	12

### Chapter 3: Designing with the Core

General Design Guidelines .....	20
Clocking .....	20
Resets .....	20
Protocol Description .....	21

### Chapter 4: Design Flow Steps

Customizing and Generating the Core .....	22
Constraining the Core .....	26
Simulation .....	27
Synthesis and Implementation .....	27

### Appendix A: Migrating and Upgrading

Migrating to the Vivado Design Suite .....	28
Upgrading in the Vivado Design Suite .....	28

### Appendix B: Debugging

Finding Help on Xilinx.com .....	29
Debug Tools .....	30
Simulation Debug .....	31

Hardware Debug ..... 31  
Interface Debug ..... 32

**Appendix C: Application Software Development**

Device Drivers ..... 33

**Appendix D: Additional Resources and Legal Notices**

Xilinx Resources ..... 34  
Documentation Navigator and Design Hubs ..... 34  
References ..... 35  
Revision History ..... 35  
Please Read: Important Legal Notices ..... 36

## Introduction

In a multiprocessor environment, the processors need to communicate data with each other. The easiest method is to set up inter-processor communication through a mailbox. The Mailbox core features a bidirectional communication channel between two processors. The Mailbox core can be connected to the processor either through AXI4-Lite or AXI4-Stream interfaces. The interfaces are available for connection to any IP that supports them, for example MicroBlaze™.

## Features

- Supports AXI4-Lite and AXI4-Stream independently on each of the ports
- Configurable depth of mailbox
- Configurable interrupt thresholds and maskable interrupts
- Configurable synchronous or asynchronous operation
- Bidirectional communication

LogiCORE™ IP Facts Table	
<b>Core Specifics</b>	
Supported Device Family <sup>(1)</sup>	UltraScale+™ UltraScale™ Zynq®-7000 All Programmable SoC 7 Series
Supported User Interfaces	AXI4-Lite, AXI4-Stream
Resources	<a href="#">Performance and Resource Utilization web page</a>
<b>Provided with Core</b>	
Design Files	Vivado: RTL
Example Design	Not Provided
Test Bench	Not Provided
Constraints File	Not Provided
Simulation Model	VHDL Behavioral
Supported S/W Driver <sup>(2)</sup>	Standalone driver: mbox
<b>Tested Design Flows<sup>(3)</sup></b>	
Design Entry	Vivado® Design Suite
Simulation	For supported simulators, see the <a href="#">Xilinx Design Tools: Release Notes Guide</a>
Synthesis	Vivado Synthesis
<b>Support</b>	
Provided by Xilinx at the <a href="#">Xilinx Support web page</a>	

### Notes:

1. For a complete listing of supported devices, see the Vivado IP catalog.
2. Standalone driver details can be found in the SDK directory (<install\_directory>/SDK/<release>/data/embeddedsw/doc/xilinx\_drivers.htm). Linux OS and driver support information is available from the [Xilinx Wiki page](#).
3. For the supported versions of the tools, see the [Xilinx Design Tools: Release Notes Guide](#).

## Overview

The Mailbox core is used for bidirectional inter-processor communication. A mailbox is a link between two otherwise separate processor systems. Other multi-port IP blocks, such as a memory controller, can also be shared by the two sub systems.

In addition to sending the actual data between processors, the Mailbox core can be used to generate interrupts between the processors.

The Mailbox core in a typical AXI4-Lite system is shown in the top-level block diagram in [Figure 1-1](#). The AXI4-Stream option has the Mailbox core interface connected directly to a master with no bus in between.

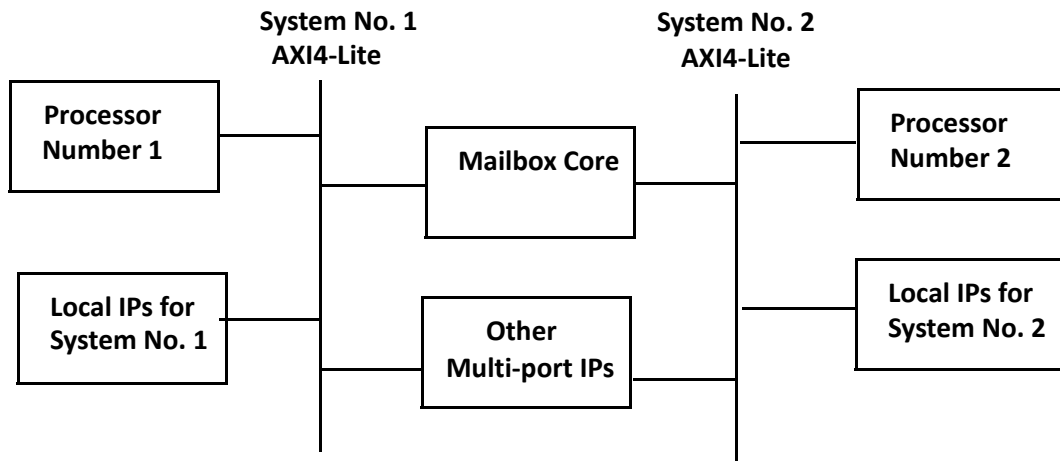


Figure 1-1: Mailbox Core in an AXI4-Lite System

---

## Feature Summary

### Bus Interfaces

The Mailbox core has two bus interfaces to access the internal resources, usually connected to different processors in a multi-processor system. Both interfaces can be independently configured to use an AXI4-Lite or AXI4-Stream interface.

### Registers

The Mailbox core provides several types of registers, available with the AXI4-Lite bus interface, to exchange information and handle interrupts:

- Read and Write Data registers, which provide the primary way to transfer data with the Mailbox core. These registers act as a FIFO, to allow data transfers from one processor (writing to the FIFO) to the other (reading from the FIFO). The FIFO size can be configured to hold from 16 up to 8192 values.
- Status and control registers, to determine FIFO and interrupt threshold status.
- Interrupt registers, which control the behavior of interrupts, in particular FIFO fill thresholds to determine when an interrupt is generated.

### Streaming Access

When using the AXI4-Stream bus interface, data transfer FIFOs are available to read from or write to an interface. It is possible to check if the FIFO is full before writing or empty before reading, by using a non-blocking test instruction (for example, `tnput` or `tnget`).

---

## Licensing and Ordering

This Xilinx® LogiCORE™ IP module is provided at no additional cost with the Xilinx Vivado® Design Suite 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 about pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your [local Xilinx sales representative](#).

# Product Specification

## Standards

The Mailbox core adheres to the *ARM® AMBA AXI and ACE Protocol Specification* [Ref 1].

The Mailbox core adheres to the *ARM AMBA AXI4-Stream Protocol Specification* [Ref 2].

## Performance

The frequency and latency of the Mailbox core are optimized for use with MicroBlaze™. This means that the frequency targets are aligned to MicroBlaze targets.

## Maximum Frequencies

For details about performance, visit [Performance and Resource Utilization](#).

## Latency and Throughput

The latency and throughput of accesses to the Mailbox core FIFO depends on the bus interface. The latency for each interface when reading or writing, as well as the throughput, is shown in [Table 2-1](#), according to the parameter settings affecting the measurements.

Table 2-1: Latency and Throughput

Bus Interface	Read Latency (clock cycles)		Write Latency (clock cycles)		Throughput (clock cycles/word)	
	Minimum	Typical	Minimum	Typical	Minimum	Typical
<b>Synchronous Distributed RAM (C_ASYNC_CLKS = 0, C_IMPL_STYPE = 0):</b>						
AXI4-Lite	3	12	3	12	6	24
AXI4-Stream	1	10	1	10	2	20
<b>Synchronous Block RAM (C_ASYNC_CLKS = 0, C_IMPL_STYPE = 1):</b>						
AXI4-Lite	3	12	4	13	7	25
AXI4-Stream	1	10	1	11	2	21

Table 2-1: Latency and Throughput (Cont'd)

Bus Interface	Read Latency (clock cycles)		Write Latency (clock cycles)		Throughput (clock cycles/word)	
	Minimum	Typical	Minimum	Typical	Minimum	Typical
<b>Asynchronous Distributed RAM (C_ASYNC_CLKS = 1, C_IMPL_STYPE = 0):</b>						
AXI4-Lite	3	12	3	12	10	28
AXI4-Stream	1	10	1	10	2	24
<b>Asynchronous Block RAM (C_ASYNC_CLKS = 1, C_IMPL_STYPE = 1):</b>						
AXI4-Lite	3	12	4	13	11	29
AXI4-Stream	1	10	1	11	2	25

The minimum number only takes into account the effect of the hardware implementation, whereas the typical value also accounts for the typical default software driver overhead.

The throughput denotes the time for one write access followed by one read access. If several writes and reads are performed, they might partly overlap. As the number of accesses increases, this overlap causes the throughput to approach the access latency.

When using the AXI4-Stream interface, the software can be optimized by reading or writing several words in sequence, but in this case care must be taken to avoid stalls due to intermediate instructions. The stream instructions must be consecutive to achieve the minimum latency.

## Resource Utilization

For details about resource utilization, visit [Performance and Resource Utilization](#).

## Port Descriptions

The Mailbox core has two interfaces that are used to connect to the rest of the system. Both interfaces can be independently configured to use the AXI4-Lite or AXI4-Stream interface. The signal descriptions are included in three tables:

1. The AXI4-Lite signals are described in [Table 2-2](#).
2. The AXI4-Stream signals are described in [Table 2-3](#).
3. The common signals are described in [Table 2-4](#).

All signals in [Table 2-2](#) through [Table 2-4](#) apply to both interface sides; <x> denotes the interface number, which can be 0 or 1.



Table 2-2: AXI4-Lite I/O Signal Description

Port	Signal Name	Interface	I/O	Initial State	Description
<b>System Signals</b>					
P43	S<x>_AXI_ACLK	System	I	-	AXI clock
P44	S<x>_AXI_ARESETN	System	I	-	AXI reset, active-Low
<b>AXI Write Address Channel Signals</b>					
P45	S<x>_AXI_AWADDR[C_S<x>_AXI_ADDR_WIDTH-1:0]	AXI	I	-	AXI write address. The write address bus gives the address of the write transaction.
P46	S<x>_AXI_AWVALID	AXI	I	-	Write address valid. This signal indicates that valid write address is available.
P47	S<x>_AXI_AWREADY	AXI	O	0	Write address ready. This signal indicates that the slave is ready to accept an address.
<b>AXI Write Channel Signals</b>					
P48	S<x>_AXI_WDATA[C_S<x>_AXI_DATA_WIDTH - 1: 0]	AXI	I	-	Write data
P49	S<x>_AXI_WSTB[C_S<x>_AXI_DATA_WIDTH/8-1:0] <sup>(1)</sup>	AXI	I	-	Write strobes. This signal indicates which byte lanes to update in memory. <sup>(1)</sup>
P50	S<x>_AXI_WVALID	AXI	I	-	Write valid. This signal indicates that valid write data and strobes are available.
P51	S<x>_AXI_WREADY	AXI	O	0	Write ready. This signal indicates that the slave can accept the write data.
<b>AXI Write Response Channel Signals</b>					
P52	S<x>_AXI_BRESP[1:0]	AXI	O	0x0	Write response. This signal indicates the status of the write transaction. 00 - OKAY 10 - SLVERR 11 - DECERR
P53	S<x>_AXI_BVALID	AXI	O	0	Write response valid. This signal indicates that a valid write response is available.
P54	S<x>_AXI_BREADY	AXI	I	-	Response ready. This signal indicates that the master can accept the response information.
<b>AXI Read Address Channel Signals</b>					
P55	S<x>_AXI_ARADDR[C_S<x>_AXI_ADDR_WIDTH -1:0]	AXI	I	-	Read address. The read address bus gives the address of a read transaction.

Table 2-2: AXI4-Lite I/O Signal Description (Cont'd)

Port	Signal Name	Interface	I/O	Initial State	Description
P56	S<x>_AXI_ARVALID	AXI	I	-	Read address valid. This signal indicates, when High, that the read address is valid and remains stable until the address acknowledge signal, S<x>_AXI_ARREADY, is High.
P57	S<x>_AXI_ARREADY	AXI	O	1	Read address ready. This signal indicates that the slave is ready to accept an address.
<b>AXI Read Data Channel Signals</b>					
P58	S<x>_AXI_RDATA[C_S<x>_AXI_DATA_WIDTH -1:0]	AXI	O	0x0	Read data
P59	S<x>_AXI_RRESP[1:0]	AXI	O	0x0	Read response. This signal indicates the status of the read transfer. 00 - OKAY 10 - SLVERR 11 - DECERR
P60	S<x>_AXI_RVALID	AXI	O	0	Read valid. This signal indicates that the required read data is available and the read transfer can complete
P61	S<x>_AXI_RREADY	AXI	I	-	Read ready. This signal indicates that the master can accept the read data and response information

**Notes:**

1. This signal is not used. The Mailbox core assumes that all byte lanes are active.

Table 2-3: AXI4-Stream I/O Signal Description

Port	Signal Name	Interface	I/O	Initial State	Description
<b>System Signals</b>					
P62	S<x>_AXIS_ACLK	System	I	-	AXI clock
P63	M<x>_AXIS_ACLK	System	I	-	AXI clock
<b>AXI Slave Channel Signals</b>					
P64	S<x>_AXIS_TDATA[C_S<x>_AXIS_DATA_WIDTH - 1: 0]	AXIS	I	-	Data
P65	S<x>_AXIS_TLAST	AXIS	I	-	Last data flag, indicates that this is the last word.
P66	S<x>_AXIS_TVALID	AXIS	I	-	Data valid. This signal indicates that valid data and last flag are available.
P67	S<x>_AXIS_TREADY	AXIS	O	0	Data ready. This signal indicates that the slave can accept the data.

Table 2-3: AXI4-Stream I/O Signal Description (Cont'd)

Port	Signal Name	Interface	I/O	Initial State	Description
<b>AXI Master Channel Signals</b>					
P68	M<x>_AXIS_TDATA[C_M<x>_>_AXIS_DATA_WIDTH-1:0]	AXIS	O	0x0	Data
P69	M<x>_AXIS_TLAST	AXIS	O	0	Last data flag, indicates that this is the last word.
P70	M<x>_AXIS_TVALID	AXIS	O	0	Data valid. This signal indicates that valid data and last flag are available.
P71	M<x>_AXIS_TREADY	AXIS	I	-	Data ready. This signal indicates that the slave can accept the data.

Table 2-4: Mailbox Common I/O Signal Description

Port	Signal Name	Interface	I/O	Initial State	Description
<b>Common Interface Signals</b>					
P82	FSL_Clk	System	I	N/A	This is the input clock to the Mailbox core when used in synchronous FIFO mode (C_ASYNC_CLKS = 0) and both interfaces are AXI4-Stream based (C_INTERCONNECT_PORT_<x> = 4). The FSL_Clk is in this case used to clock the core, in all other cases the internal Mailbox core clock is automatically derived from S<x>_AXI_ACLK.
P83	SYS_Rst	System	I	N/A	External system reset. This signal is only required when both interfaces are configured to be streaming interfaces (AXI4-Stream). If any AXI4-Lite interface is available this signal is optional.
<b>Common Signals</b>					
P85	Interrupt_0	System	O	0	Interrupt signal that data is available at interface 0
P86	Interrupt_1	System	O	0	Interrupt signal that data is available at interface 1

## Register Space

Each interface of the Mailbox core has the same set of information registers. The information at each interface is not identical but rather localized for that interface because the communication is bidirectional.

Table 2-5 shows all the Mailbox core registers and their addresses for AXI4-Lite case. Much of the information can be acquired for the AXI4-Stream case with the use of S<x>\_AXIS\_TREADY and M<x>\_AXIS\_TVALID.

Table 2-5: Mailbox Registers

Base Address + Offset (hex)	Register Name	Access Type	Default Value (hex)	Description
BASEADDR + 0x0	WRDATA	Write	N/A	Write Data address. Write only.
BASEADDR + 0x4	Reserved	N/A	N/A	Reserved for future use
BASEADDR + 0x8	RDDATA	Read	N/A	Read Data address. Read only
BASEADDR + 0xC	Reserved	N/A	N/A	Reserved for future use
BASEADDR + 0x10	STATUS	Read	0x1	Status flags for Mailbox core. Read only.
BASEADDR + 0x14	ERROR	Read	0x0	Error flags, clear on read. Read only.
BASEADDR + 0x18	SIT	-	-	Send Interrupt Threshold. Read/Write
BASEADDR + 0x1C	RIT	-	-	Receive Interrupt Threshold. Read/Write
BASEADDR + 0x20	IS	-	-	Interrupt Status register. Read/Write
BASEADDR + 0x24	IE	-	-	Interrupt Enable register. Read/Write
BASEADDR + 0x28	IP	-	-	Interrupt Pending register. Read only
BASEADDR + 0x2C	CTRL	Write	N/A	Control Register. Write only.
BASEADDR + 0x30	Reserved	-	-	Reserved for future use
BASEADDR + 0x34	Reserved	-	-	Reserved for future use
BASEADDR + 0x38	Reserved	-	-	Reserved for future use
BASEADDR + 0x3C	Reserved	-	-	Reserved for future use

## Write Data Register (WRDATA)

Writing to this register results in the data being transferred to the RDDATA register at the other interface. Trying to write while the full flag is set results in an error and the FULL\_ERROR bit is set. The register is write only and a read request issued to WRDATA is ignored. Bit assignment in the WRDATA register is described in [Table 2-7](#).

Table 2-6: Write Data Register

WRDATA	
31	0

Table 2-7: Mailbox Write Data Register Bit Definitions

Bit(s)	Name	Core Access	Reset Value	Description
31-0	WRDATA	Write	-	Write register to send data to the other interface

## Mailbox Read Data Register (RDDATA)

Reading from this register pops one value from the mail FIFO. Trying to read while the empty flag is set results in an error and the EMPTY\_ERROR bit is set. The register is read only and a write request issued to RDDATA is ignored. Bit assignment in the RDDATA register is described in [Table 2-9](#).

Table 2-8: Read Data Register

RDDATA	
31	0

Table 2-9: Mailbox Read Data Register Bit Definitions

Bit(s)	Name	Core Access	Reset Value	Description
31-0	RDDATA	Read	-	Read register to get data word sent from the other interface

## Mailbox Status Register (STATUS)

The Mailbox Status Register contains the current status of the Mailbox core. The register is read only and a write request issued to STATUS is ignored. Bit assignment in the STATUS register is described in [Table 2-11](#).

Table 2-10: Status Register

Reserved				RTA	STA	Full	Empty
31	4	3	2	1	0		

Table 2-11: Mailbox Status Register Bit Definitions

Bit(s)	Name	Core Access	Reset Value	Description
31-4	Reserved			Reserved for future use
3	RTA	Read	0	Receive Threshold Active indicates the current FIFO status of this interface in the receive direction 0 = The receive FIFO level is less than or equal to the RIT threshold 1 = The receive FIFO level is greater than the RIT threshold
2	STA	Read	0	Send Threshold Active indicates the current FIFO status of this interface in the send direction 0 = The send FIFO level is greater than the SIT threshold 1 = The send FIFO level is less than or equal to the SIT threshold
1	Full	Read	0	Indicates the current status of this interface in the send direction 0 = There is room for more data 1 = The FIFO is full; any attempts to write data are ignored and an error is generated
0	Empty	Read	1	Indicates the current status of this interface in the receive direction 0 = There is data available 1 = The FIFO is empty, any attempts to read data are ignored and an error is generated

## Mailbox Error Register (ERROR)

The Mailbox Error Register contains the error flags for AXI4-Lite accesses from this interface. The error register is cleared at read, this means that all bits are sticky and that they indicate any errors that occurred since last time the error register was read. The register is read only and a write request issued to ERROR is ignored. Bit assignment in the ERROR register is described in [Table 2-13](#).

Table 2-12: Error Register

Reserved		Full Error	Empty Error
31	2	1	0

Table 2-13: Mailbox Error Register Bit Definitions

Bit(s)	Name	Core Access	Reset Value	Description
31–2	Reserved			Reserved for future use
1	Full Error	Read	0	Indicates if there has been any attempts to write to the WRDATA register while the Full flag was asserted since the error register was last read 0 = No error has occurred 1 = One or more attempts to write while Mailbox FIFO is full
0	Empty Error	Read	0	Indicates if there has been any attempts to read from the RDDATA register while the Empty flag was asserted since the error register was last read 0 = No error has occurred 1 = One or more attempts to read while Mailbox FIFO is empty

## Mailbox Send Interrupt Threshold Register (SIT)

The Mailbox Send Interrupt Threshold Register contains the interrupt threshold for this interface in the send direction. Depending on the send FIFO data level writing a new SIT can cause a rising edge on STA that can generate a STI interrupt if it is enabled in the IE register. Bit assignment in the SIT register is described in [Table 2-15](#).

Table 2-14: SIT Register

		SIT
31	Log2(C_MAILBOX_DEPTH)-1	0

Table 2-15: Mailbox SIT Register Bit Definitions

Bit(s)	Name	Core Access	Reset Value	Description
Log2(C_MAILBOX_DEPTH)	SIT	Read/Write	0	Lower Log2(C_MAILBOX_DEPTH) bits used, right justified to bit 0

## Mailbox Receive Interrupt Threshold Register (RIT)

The Mailbox Receive Interrupt Threshold Register contains the interrupt threshold for this interface in the receive direction. Depending on the receive FIFO data level writing a new RIT can cause a rising edge on RTA that can generate a RTI interrupt if it is enabled in the IE register. Bit assignment in the RIT register is described in [Table 2-17](#).

Table 2-16: RIT Register

	RIT
31	Log2(C_MAILBOX_DEPTH)-1      0

Table 2-17: Mailbox RIT Register Bit Definitions

Bit(s)	Name	Core Access	Reset Value	Description
Log2(C_MAILBOX_DEPTH)	RIT	Read/Write	0	Lower Log2(C_MAILBOX_DEPTH) bits used, right justified to bit 0

## Mailbox Interrupt Status Register (IS)

The Mailbox Interrupt Status Register contains the current interrupt status for this interface. There are three types of interrupts that can be generated. Mailbox Error interrupt are generated when any of the bits in the ERROR register is set. The other two interrupts are FIFO related: RTI is generated for a rising edge on the RTA bit in the STATUS register and STI that is generated for a rising edge on the STA STATUS register bit. RTI and STI are used to indicate that it is time to read from or write to the FIFOs to avoid any stalls in the data flow. Bit assignment in the IS register is described in [Table 2-19](#).

Table 2-18: IS Register

	Reserved	ERR	RTI	STI
31	3	2	1	0

Table 2-19: Mailbox IS Register Bit Definitions

Bit(s)	Name	Core Access	Reset Value	Description
31-3	Reserved			Reserved for future use
2	ERR	Read/Write	0	Mailbox Error Interrupt Status for this interface. Values for read: 0 = No interrupt event has occurred. 1 = A Mailbox error has occurred. Values for write: 0 = No change 1 = Acknowledge and clear the interrupt if it is active



Table 2-19: Mailbox IS Register Bit Definitions (Cont'd)

Bit(s)	Name	Core Access	Reset Value	Description
1	RTI	Read/Write	0	Mailbox Receive Threshold Interrupt pending status for this interface. Values for read: 0 = No interrupt event has occurred. 1 = Data level in the receive FIFO has caused a RTI. Values for write: 0 = No change 1 = Acknowledge and clear the interrupt if it is active
0	STI	Read/Write	0	Mailbox Send Threshold Interrupt pending status for this interface. Values for read: 0 = No interrupt event has occurred. 1 = Data level in the send FIFO has caused a STI. Values for write: 0 = No change 1 = Acknowledge and clear the interrupt if it is active

## Mailbox Interrupt Enable Register (IE)

The Mailbox Interrupt Enable Register contains the mask for the allowed interrupts on this interface. Bit assignment in the IE register is described in [Table 2-21](#).

Table 2-20: IE Register

Reserved			ERR	RTI	STI
31	3	2	1	0	

Table 2-21: Mailbox IE Register Bit Definitions

Bit(s)	Name	Core Access	Reset Value	Description
31-3	Reserved			Reserved for future use
2	ERR	Read/Write	0	Mailbox Error Interrupt Enable for this interface 0 = ERR interrupt is disabled 1 = ERR interrupt is enabled
1	RTI	Read/Write	0	Mailbox Receive Threshold Interrupt Enable for this interface 0 = RTI interrupt is disabled 1 = RTI interrupt is enabled
0	STI	Read/Write	0	Mailbox Send Threshold Interrupt Enable for this interface 0 = STI interrupt is disabled 1 = STI interrupt is enabled

## Mailbox Interrupt Pending Register (IP)

The Mailbox Interrupt Pending Register contains the currently pending interrupts from this interface. It is a read only register generated by performing a bitwise AND between the IS and IE registers. A write request issued to the IP is ignored. Bit assignment in the IP register is described in [Table 2-23](#). All the bits in this register are OR'd together to generate the interrupt output signal for this interface. When an interrupt has been serviced it is acknowledged by writing the corresponding bit to the IS Register.

Table 2-22: IP Register

Reserved		ERR	RTI	STI
31	3	2	1	0

Table 2-23: Mailbox IP Register Bit Definitions

Bit(s)	Name	Core Access	Reset Value	Description
31-3	Reserved			Reserved for future use
2	ERR	Read	0	Mailbox Error Interrupt Pending status for this interface 0 = No pending interrupt 1 = Pending interrupt for Mailbox errors
1	RTI	Read	0	Mailbox Receive Threshold Interrupt Pending status for this interface 0 = No pending interrupt 1 = Pending interrupt for data level in receive FIFO
0	STI	Read	0	Mailbox Send Threshold Interrupt Pending status for this interface 0 = No pending interrupt 1 = Pending interrupt for data level in send FIFO

## Mailbox Control Register (CTRL)

The Mailbox Control Register is used to clear (reset) the Receive and Send FIFO from each interface. This can be useful to ensure that no stale data remains in the FIFO, for example when resetting or restarting software on a processor connected to the Mailbox. It is a write only register. A read request issued to the CTRL register is ignored. Bit assignment in the CTRL register is described in [Table 2-25](#).

Table 2-24: CTRL Register

Reserved		CRF	CSF
31	2	1	0

Table 2-25: Mailbox CTRL Register Bit Definitions

Bit(s)	Name	Core Access	Reset Value	Description
31–2	Reserved			Reserved for future use
1	CRF	Write	-	Clear Mailbox receive FIFO for this interface 0 = Do nothing 1 = Clear receive FIFO
0	CSF	Write	-	Clear Mailbox send FIFO for this interface 0 = Do nothing 1 = Clear send FIFO

# Designing with the Core

---

## General Design Guidelines

This chapter includes guidelines and additional information to facilitate designing with the core.

---

## Clocking

The `Sn_AXI_ACLK` ( $n = 0, 1$ ) input is only used when the AXI4-Lite interconnect is used. Then it should normally be connected to the same clock as the interconnect.

The `Mn_AXIS_ACLK` or `Sn_AXIS_ACLK` ( $n = 0, 1$ ) are only used when AXI4-Stream is used. Then they should be connected to the corresponding stream clock.

With synchronous operation (`C_ASYNC_CLKS = 0`), the two clock inputs used must both be connected to the same clock signal in all the cases above.

The `FSL_Clk` input is only used with synchronous operation (`C_ASYNC_CLKS = 0`) and when both interfaces use AXI4-Stream. Then it should be connected to the common AXI4-Stream clock signal.

---

## Resets

The `Sn_AXI_ARESETN` ( $n = 0, 1$ ) input is only used when the AXI4-Lite interconnect is used. Then it should normally be connected to the same reset as the interconnect.

The `SYS_Rst` input is necessary when both interfaces use AXI4-Stream, because the streaming interfaces do not have dedicated resets.

All enabled reset signals are treated equally and reset the entire Mailbox core, including any AXI4-Lite interfaces. With asynchronous operation (`C_ASYNC_CLKS = 1`) synchronization of the reset signals to the different clock domains is handled automatically internally.

---



**IMPORTANT:** *It is recommended that the reset signals are asserted for at least 16 clock cycles of the slowest clock connected to the Mailbox core.*

---

## Protocol Description

See the ARM® AMBA® *AXI and ACE Protocol Specification* [Ref 1] for a description of the AXI4-Lite protocol.

See the *ARM AMBA AXI4-Stream Protocol Specification* [Ref 2] for a description of the AXI4-Stream protocol.

# Design Flow Steps

This chapter describes customizing and generating the core, constraining the core, and the simulation, synthesis and implementation steps that are specific to this IP core. More detailed information about the standard Vivado® design flows and the IP integrator can be found in the following Vivado Design Suite user guides:

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 3]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 4]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 5]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 6]

---

## Customizing and Generating the Core

This section includes information about using Xilinx tools to customize and generate the core in the Vivado Design Suite.

If you are customizing and generating the core in the Vivado IP integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 3] for detailed information. IP integrator might auto-compute certain configuration values when validating or generating the design. To check whether the values do change, see the description of the parameter in this chapter. To view the parameter value, run the `validate_bd_design` command in the Tcl console.

You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

1. Select the IP from the Vivado IP catalog.
2. Double-click the selected IP or select the **Customize IP** command from the toolbar or right-click menu.

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 4] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 5].

**Note:** Figures in this chapter are illustrations of the Vivado Integrated Design Environment (IDE). The layout depicted here might vary from the current version.

This chapter includes information about using Xilinx tools to customize and generate the core in the Vivado® Design Suite.

The Mailbox core parameters are divided into two categories: System and Mailbox. When using the Vivado IP integrator feature, the addresses are auto-generated.

The User tab configuration screen is shown in [Figure 4-1](#).

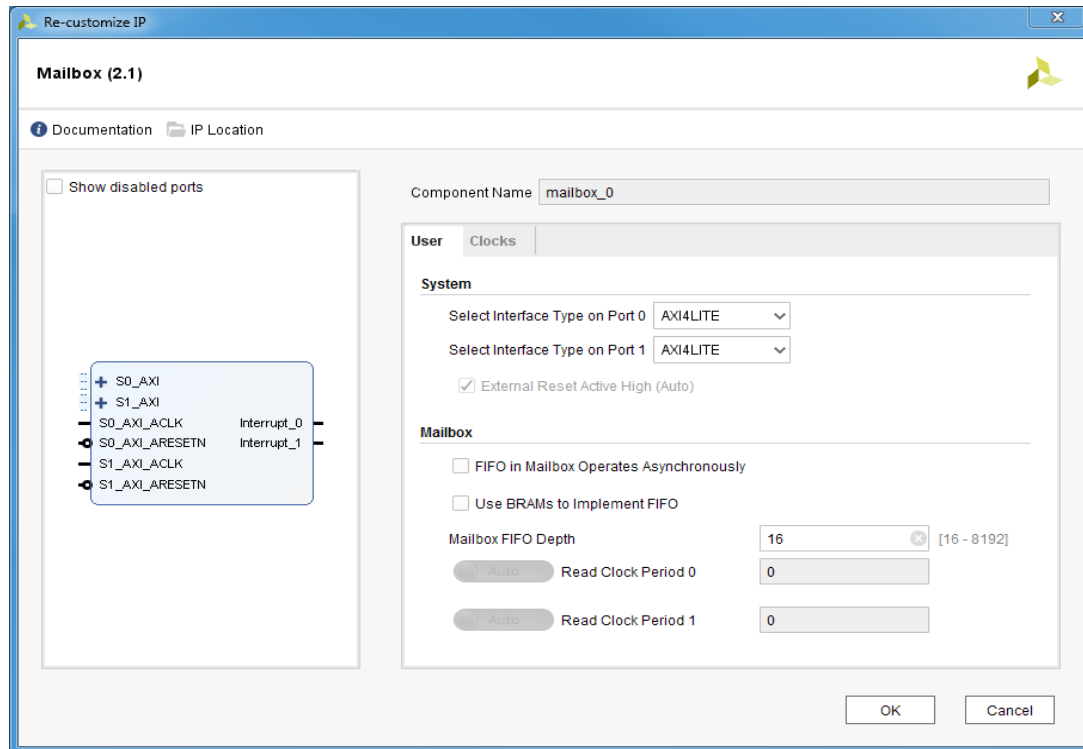


Figure 4-1: Configuration Screen

- **Select Interface Type** - Sets the bus interface on both ports to either AXI4-Lite or AXI4-Stream.
- **External Reset Active High** - Sets the reset polarity. Auto-generated by the tool.
- **FIFO in Mailbox Operates Asynchronously** - Enables asynchronous operation, when the clocks of the two interfaces are not identical.
- **Use BRAMs to Implement FIFO** - A mask indicating which address bits the LMB BRAM Interface Controller takes into account when decoding an access.
- **Mailbox FIFO Depth** - Sets the number of words available in the FIFO, from 16 to 8192.
- **Read Clock Period** - Sets the clock period in picoseconds for asynchronous operation.

The Clocks parameter tab is shown in [Figure 4-2](#).

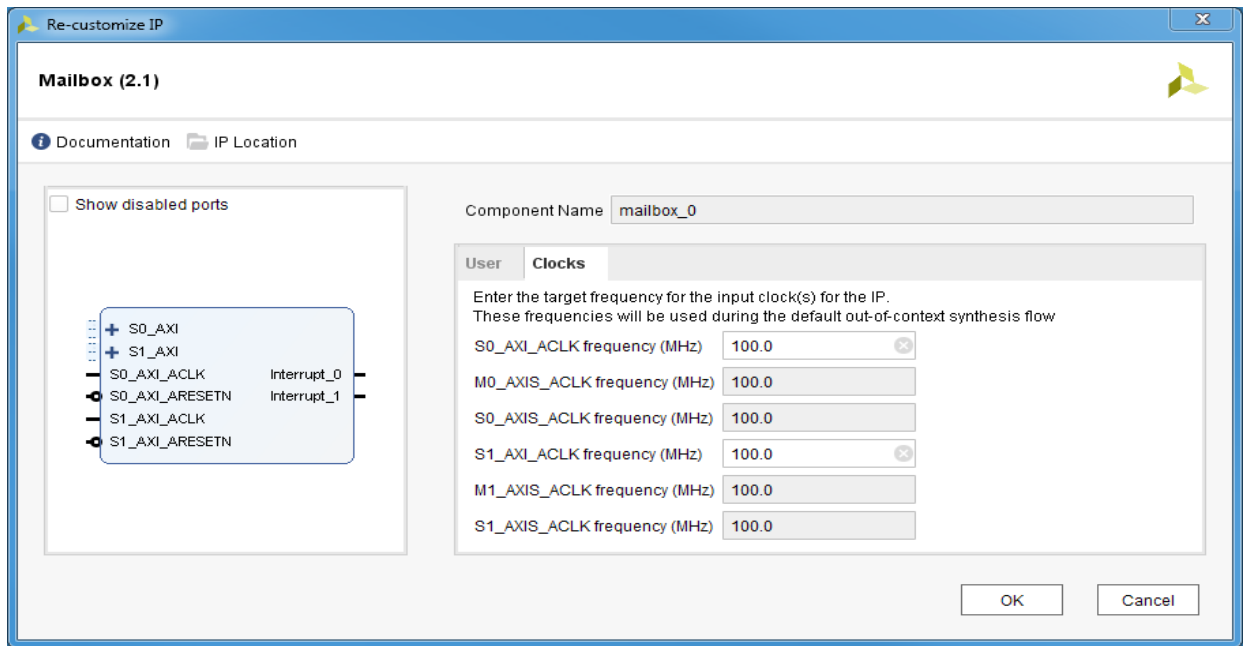


Figure 4-2: Clocks Parameter Tab

- **Sn\_AXI\_ACLK frequency (MHz)** - Sets the frequency for the AXI4 slave interface clock (n = 0–1)
- **Mn\_AXIS\_ACLK frequency (MHz)** - Sets the frequency for the AXI4-Stream master interface clock (n = 0–1)
- **Sn\_AXIS\_ACLK frequency (MHz)** - Sets the frequency for the AXI4-Stream slave interface clock (n = 0–1)

## Parameter Values

To obtain a Mailbox core that is uniquely tailored for the system, certain features can be parameterized in the core design. This allows you to configure a design that uses the resources required by the system only and that operates with the best possible performance. The features that can be parameterized in the design are as shown in [Table 4-1](#).

Table 4-1: Mailbox Design Parameters

Generic	Feature/Description	Parameter Name	Allowable Values	Default Value	VHDL Type
<b>System Parameters</b>					
G1	Target FPGA family	C_FAMILY	Supported architectures	virtex7	string
G2	Level of external reset	C_EXT_RESET_HIGH	0 or 1	1	integer



Table 4-1: Mailbox Design Parameters (Cont'd)

Generic	Feature/Description	Parameter Name	Allowable Values	Default Value	VHDL Type
<b>Mailbox Parameters</b>					
G20	Specify if interfaces are synchronous or asynchronous	C_ASYNC_CLKS	0–1	0	integer
G21	Number of synchronization FF for each clock domain crossing	C_NUM_SYNC_FF	0–8	2	integer
G22	Use BRAMs to implement FIFO	C_IMPL_STYLE	0–1	1	integer
G23	Select interface type to be used on port 0: 2 - AXI4-Lite 4 - AXI4-Stream	C_INTERCONNECT_PORT_0	2, 4	0	integer
G24	Select interface type to be used on port 1: 2 - AXI4-Lite 4 - AXI4-Stream	C_INTERCONNECT_PORT_1	2, 4	0	integer
G25	FIFO depth of mailbox	C_MAILBOX_DEPTH	16–8192	16	integer
G26	Read Clock period for interface 0 when asynchronous LUTRAM is used (in ps)	C_READ_CLOCK_PERIOD_0	>0 when enabled	0	integer
G27	Read Clock period for interface 1 when asynchronous LUTRAM is used (in ps)	C_READ_CLOCK_PERIOD_0	>0 when enabled	0	integer

## User Parameters

Table 4-2 shows the relationship between the fields in the Vivado IDE and the User Parameters (which can be viewed in the Tcl console).

Table 4-2: Vivado IDE Parameter to User Parameter Relationship

Vivado IDE Parameter	User Parameter	Default Value
Select Interface Type on Port 0	C_INTERCONNECT_PORT_0	2
Select Interface Type on Port 1	C_INTERCONNECT_PORT_1	2
FIFO in Mailbox Operates Asynchronously	C_ASYNC_CLKS	0
Use BRAMs to Implement FIFO	C_IMPL_STYLE	0
Mailbox FIFO Depth	C_MAILBOX_DEPTH	16
Read Clock Period 0	C_READ_CLOCK_PERIOD_0	0
Read Clock Period 1	C_READ_CLOCK_PERIOD_1	0

## Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 4].

---

## Constraining the Core

This section contains information about constraining the core in the Vivado Design Suite.

### Required Constraints

There are no required constraints for this core.

### Device, Package, and Speed Grade Selections

There are no Device, Package or Speed Grade requirements for this core.

### Clock Frequencies

There are no specific clock frequency requirements for this core.

### Clock Management

The Mailbox core can either be fully synchronous with all clocked elements clocked by the same physical clock, or asynchronous with different clocks on the two connected bus interfaces.

With an asynchronous configuration, the parameter C\_ASYNC\_CLKS (FIFO in Mailbox Operates Asynchronously) must be set manually, as well as the read clock period in picoseconds for each bus interface using the two parameters C\_READ\_CLOCK\_PERIOD\_0 (Read Clock Period 0) and C\_READ\_CLOCK\_PERIOD\_1 (Read Clock Period 0).

To operate properly when connected to MicroBlaze™, the corresponding bus interface clock must be the same as the MicroBlaze CLK.

### Clock Placement

There are no specific Clock placement requirements for this core.

### Banking

There are no specific Banking rules for this core.

### Transceiver Placement

There are no Transceiver Placement requirements for this core.

## I/O Standard and Placement

There are no specific I/O standards and placement requirements for this core.

---

## Simulation

For comprehensive information about Vivado simulation components, as well as information about using supported third-party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 6].



---

**IMPORTANT:** For cores targeting 7 series or Zynq-7000 devices, UNIFAST libraries are not supported. Xilinx IP is tested and qualified with UNISIM libraries only.

---

## Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 4].

# Migrating and Upgrading

This appendix contains information about migrating a design from the ISE® Design Suite to the Vivado® Design Suite, and for upgrading to a more recent version of the IP core. For customers upgrading in the Vivado Design Suite, important details (where applicable) about any port changes and other impact to user logic are included.

---

## Migrating to the Vivado Design Suite

For information about migrating to the Vivado Design Suite, see the *ISE to Vivado Design Suite Migration Guide* (UG911) [\[Ref 7\]](#).

---

## Upgrading in the Vivado Design Suite

This section provides information about any changes to the user logic or port designations that take place when you upgrade to a more current version of this IP core in the Vivado Design Suite.

# Debugging

This appendix includes details about resources available on the Xilinx Support website and debugging tools.

---

## Finding Help on Xilinx.com

To help in the design and debug process when using the Mailbox core, the [Xilinx Support web page](#) contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support.

### Documentation

This product guide is the main document associated with the Mailboxcore. This guide, along with documentation related to all products that aid in the design process, can be found on the [Xilinx Support web page](#) or by using the Xilinx® Documentation Navigator.

Download the Xilinx Documentation Navigator from the [Downloads page](#). For more information about this tool and the features available, open the online help after installation.

### Answer Records

Answer Records include information about commonly encountered problems, helpful information on how to resolve these problems, and any known issues with a Xilinx product. Answer Records are created and maintained daily ensuring that users have access to the most accurate information available.

Answer Records for this core can be located by using the Search Support box on the main [Xilinx support web page](#). To maximize your search results, use proper keywords such as

- Product name
- Tool message(s)
- Summary of the issue encountered

A filter search is available after results are returned to further target the results.

## Master Answer Record for the Mailbox Core

AR: [54415](#)

### Technical Support

Xilinx provides technical support at the [Xilinx Support web page](#) for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support if you do any of the following:

- Implement the solution in devices that are not defined in the documentation.
- Customize the solution beyond that allowed in the product documentation.
- Change any section of the design labeled DO NOT MODIFY.

To contact Xilinx Technical Support, navigate to the [Xilinx Support web page](#).

---

### Debug Tools

The main tool available to address Mailbox design issues is the Vivado® Design Suite debug feature.

#### Vivado Design Suite Debug Feature

The Vivado Design Suite debug feature inserts logic analyzer and virtual I/O cores directly into your design. The debug feature also allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature in the Vivado IDE is used for logic debugging and validation of a design running in Xilinx devices.

The Vivado logic analyzer is used with the logic debug IP cores, including:

- ILA 2.0 (and later versions)
- VIO 2.0 (and later versions)

See the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [\[Ref 8\]](#).

### Reference Boards

All 7 series, UltraScale and UltraScale+ Xilinx development boards support the Mailbox core. These boards can be used to prototype designs and establish that the core can communicate with the system.

---

## Simulation Debug

The simulation debug flow for QuestaSim is described below. A similar approach can be used with other simulators.

- Check for the latest supported versions of QuestaSim in the [Xilinx Design Tools: Release Notes Guide](#). Is this version being used? If not, update to this version.
- If using Verilog, do you have a mixed mode simulation license? If not, obtain a mixed-mode license.
- Ensure that the proper libraries are compiled and mapped. In the Vivado Design Suite **Flow > Simulation Settings** can be used to define the libraries.
- Have you associated the intended software program for the MicroBlaze™ processor with the simulation? Use the command **Tools > Associate ELF Files** in Vivado Design Suite.
- When observing the traffic on the interfaces connected to the Mailbox core, see the timing in the relevant specification:
  - For AXI4-Lite, see the *AMBA® AXI and ACE Protocol Specification* [Ref 1].
  - For AXI4-Stream, see the *AMBA 4 AXI4-Stream Protocol Specification* [Ref 2].

---

## Hardware Debug

Hardware issues can range from link bring-up to problems seen after hours of testing. This section provides debug steps for common issues. The Vivado debug feature is a valuable resource to use in hardware debug. The signal names mentioned in the following sections can be probed using the debug feature to debug specific problems.

Many of these common issues can also be applied to debugging design simulations.

### General Checks

Ensure that all the timing constraints for the core were properly incorporated from the example design and that all constraints were met during implementation.

- Does it work in post-place and route timing simulation? If problems are seen in hardware but not in timing simulation, this could indicate a PCB issue. Ensure that all clock sources are active and clean.
- If using MMCMs in the design, ensure that all MMCMs have obtained lock by monitoring the `locked` port.

---

## Interface Debug

### AXI4-Lite Interfaces

Read from a register that does not have all 0s as a default to verify that the interface is functional. Output `Sn_AXI_ARREADY` asserts when the read address is valid, and output `Sn_AXI_RVALID` asserts when the read data/response is valid, where *n* is the interface number (0 or 1). If the interface is unresponsive, ensure that the following conditions are met:

- The `Sn_AXI_ACLK` input is connected and toggling.
- The interface is not being held in reset, and `Sn_AXI_ARESETN` is an active-Low reset.
- The common core reset is not active, and `SYS_Rst` is an active-High reset.
- If the simulation has been run, verify in simulation and/or a Vivado debug feature capture that the waveform is correct for accessing the AXI4-Lite interface.

### AXI4-Stream Interfaces

If data is not being transmitted or received, check the following conditions:

- If transmit `Mn_AXIS_TREADY` is stuck Low following the `Mn_AXIS_TVALID` input being asserted, the core cannot send data.
- If the receive `Sn_AXIS_TVALID` is stuck Low, the core is not receiving data.
- Check that the `Mn_AXIS_CLK` and `Sn_AXIS_CLK` inputs are connected and toggling.
- Check that the common core reset is not active, and `SYS_Rst` is an active-High reset.
- Check that the AXI4-Stream waveforms are being followed
- Check core configuration.



# Application Software Development

---

## Device Drivers

The Mailbox core is supported by the mbox driver, included with Xilinx® Software Development Kit.

# Additional Resources and Legal Notices

---

## Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see [Xilinx Support](#).

---

## Documentation Navigator and Design Hubs

Xilinx® Documentation Navigator provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open the Xilinx Documentation Navigator (DocNav):

- From the Vivado® IDE, select **Help > Documentation and Tutorials**.
- On Windows, select **Start > All Programs > Xilinx Design Tools > DocNav**.
- At the Linux command prompt, enter `docnav`.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In the Xilinx Documentation Navigator, click the **Design Hubs View** tab.
- On the Xilinx website, see the [Design Hubs](#) page.

**Note:** For more information on Documentation Navigator, see the [Documentation Navigator](#) page on the Xilinx website.

## References

These documents provide supplemental material useful with this product guide:

1. AMBA® AXI and ACE Protocol Specification ([ARM IHI 0022E](#)), registration required.
2. AMBA® AXI4-Stream Protocol Specification ([ARM IHI 0051A](#)), registration required.
3. *Vivado® Design Suite User Guide: Designing IP Subsystems using IP Integrator* ([UG994](#))
4. *Vivado Design Suite User Guide: Designing with IP* ([UG896](#))
5. *Vivado Design Suite User Guide: Getting Started* ([UG910](#))
6. *Vivado Design Suite User Guide: Logic Simulation* ([UG900](#))
7. *ISE® to Vivado Design Suite Migration Guide* ([UG911](#))
8. *Vivado Design Suite User Guide: Programming and Debugging* ([UG908](#))

## Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/04/2017	2.1	Added description of FIFO reset functionality
11/30/2016	2.1	Reset description updated.
10/05/2016	2.1	Updated Xilinx Automotive Applications Disclaimer.
11/18/2015	2.1	Added support for UltraScale+ families.
06/24/2015	2.1	Moved performance and resource utilization data to the web.
04/02/2014	2.1	Added C_NUM_SYNC_FF to control number of synchronization FF.
03/20/2013	1.0	This Product Guide replaces PG088. There are no documentation changes for this release.

---

## Please Read: Important Legal Notices

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at <https://www.xilinx.com/legal.htm#tos>; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at <https://www.xilinx.com/legal.htm#tos>.

### **AUTOMOTIVE APPLICATIONS DISCLAIMER**

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.

© Copyright 2013–2017 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. AMBA, AMBA Designer, ARM, ARM1176JZ-S, CoreSight, Cortex, PrimeCell, and MPCore are trademarks of ARM in the EU and other countries. . All other trademarks are the property of their respective owners.