AXI Data FIFO Core User Guide

UG-CORE-AXI-DATA-FIFO-v1.0
June 2023
www.efinixinc.com

Contents

Introduction

The AXI Data FIFO core provides data buffering for both read and write channels to help prevent stalls, increase throughput, and cross clock domains. It has up to four optional internal FIFOs, one for each channel except the write response channel which is always directly fed through from the MI (Master Interface) to the SI (Slave Interface). You can enable or disable the FIFOs in the read or write address and data channels.

Use the IP Manager to select IP, customize it, and generate files. The AXI Data FIFO core has an interactive wizard to help you set parameters. The wizard also has options to create a testbench and/or example design targeting an Efinix® development board.

Features

- Supports AXI3 and AXI4 protocols
- Individually configurable Read and Write datapaths
- 32 and 512 deep FIFO for each channel (excluding write response channel)
- Optional packet FIFO operation to avoid full/empty stalls in the middle of burst
- Verilog HDL RTL and simulation testbench

Device Support

Table 1: AXI Data FIFO Core Device Support

FPGA Family	Supported Device
Trion	All
Titanium	All

Resource Utilization and Performance

> **Note:** The resources and performance values provided are based on some of the supported FPGAs. These values are just guidance and change depending on the device resource utilization, design congestion, and user design.

Table 2: AXI Data FIFO Resource Utilization and Performance

Generated with Verilog HDL.

FPGA	Packet Mode	Logic Elements[1]	Memory Block	DSP Block	f_{MAX} (MHz)	Efinity Version
T20 BGA256	Yes	330/19728 (2%)	16 / 204 (8%)	0/36 (0%)	300	2023.1
	No	136/19728 (<1%)	8 / 204 (4%)	0/36 (0%)	100	2023.1
Ti60 F225	Yes	330/60800 (<1%)	37/256 (14%)	0/160 (0%)	400	2023.1
	No	330/60800 (<1%)	4/256 (2%)	0/160 (0%)	125	2023.1

Release Notes

You can refer to the IP Core Release Notes for more information about the IP core changes. The IP Core Release Notes is available in the **Efinity Downloads** page under each Efinity software release version.

> **Note:** You must be logged in to the Support Portal to view the IP Core Release Notes.

---

[1] Logic, Adders, Flipflops, etc.

Functional Description

The AXI Data FIFO consist of a FIFO for each of the channel in the AXI protocol (except for the write response channel). There are two modes of data FIFO that can be configured for each of the write and read paths:

- Block RAM based FIFO—Supported for 32 or 512-deep FIFO (data channel only)
- Block RAM based Packet FIFO—Supported for 512-deep FIFO only (Write/Read Address FIFO)

Figure 1: AXI Data FIFO Core Block Diagram
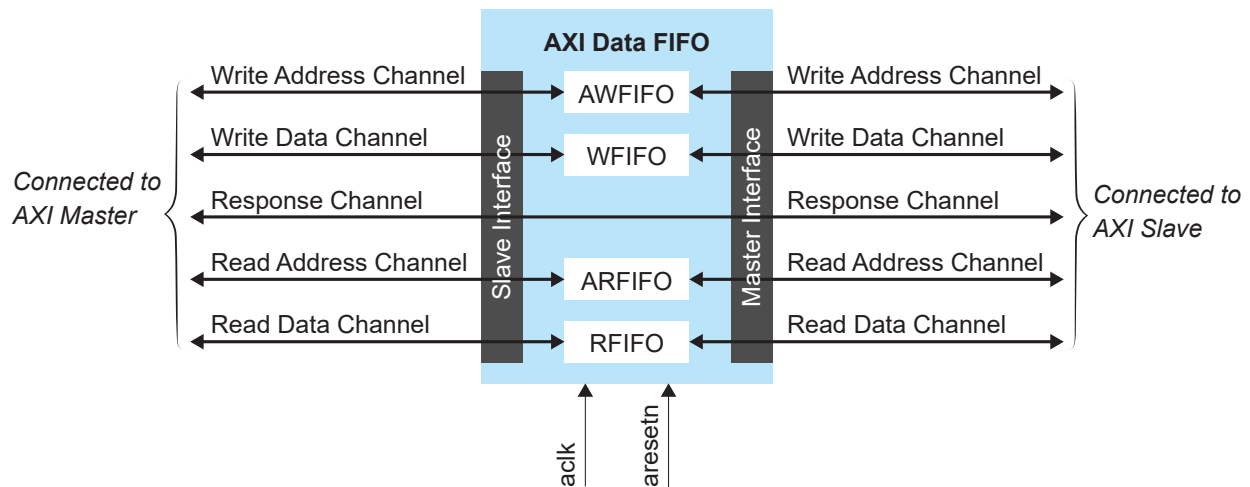
The packet FIFO mode (Write/Read Address FIFO) is utilized to prevent full/empty stalls during bursts. Alongside the 512-deep FIFO on the data channel, packet mode also incorporates a 32-deep FIFO on the corresponding address channel. When write packet mode is enabled, the write transaction on the AW channel experiences a delay until the complete write data burst (concluded with WLAST) has been received on the SI. This delay ensures that stalling caused by a slow write data source is avoided. Similarly, when read packet mode is enabled, the read transaction on the AR channel is delayed until the FIFO has sufficient vacancy to store the entire burst according to ARLEN. The term *vacancy* refers to the remaining free space in the read channel FIFO that has not been committed by previously issued AR commands. This approach prevents stalling due to a slow read destination.

Ports

Table 3: Global Interface

Port	Direction	Description
aclk	Input	Core clock.
aresetn	Input	Active-low asynchronous reset.

Table 4: Slave Interface Write Address Channel

Port	Direction	Description
s_axi_awvalid	Input	Write address channel valid.
s_axi_awaddr [ADDR_WIDTH-1:0]	Input	Write address channel address.
s_axi_awprot [2:0]	Input	Write address channel protect.
s_axi_awid [ID_WIDTH-1:0]	Input	Write address channel transaction ID.
s_axi_awburst [1:0]	Input	Write address channel burst type.
s_axi_awlen [7:0]	Input	Write address channel burst length.
s_axi_awsize [2:0]	Input	Write address channel transfer size.
s_axi_awcache [3:0]	Input	Write address channel cache encoding.
s_axi_awqos [3:0]	Input	Write address channel QoS.
s_axi_awuser [AWUSER_WIDTH-1:0]	Input	Write address channel user-defined signals.
s_axi_awlock [1:0]	Input	Write address channel locked transaction.
s_axi_awregion [3:0]	Input	Write address channel region identifier. Not applicable to AXI3.
s_axi_awready	Output	Write address channel ready.

Table 5: Slave Interface Write Data Channel

Port	Direction	Description
s_axi_wvalid	Input	Write data channel valid.
s_axi_wdata [DATA_WDITH-1:0]	Input	Write data channel data.
s_axi_wstrb [DATA_WIDTH/8-1:0]	Input	Write data channel strobe. Single bit represents data byte.
s_axi_wlast	Input	Write data channel data beat.
s_axi_wuser [WUSER_WIDTH-1:0]	Input	Write data channel user-defined signals.
s_axi_wid [ID_WIDTH-1:0]	Input	Write data channel transaction ID.
s_axi_wready	Output	Write data channel ready.

Table 6: Slave Interface Write Response Channel

Port	Direction	Description
s_axi_bready	Input	Write response channel ready.
s_axi_bresp [1:0]	Output	Write response channel response.
s_axi_bvalid	Output	Write response channel valid.
s_axi_bid [ID_WIDTH-1:0]	Output	Write response channel transaction ID.
s_axi_buser [BUSER_WIDTH-1:0]	Output	Write response channel user-defined signals.

Table 7: Slave Interface Read Address Channel

Port	Direction	Description
s_axi_arvalid	Input	Read address channel valid.
s_axi_araddr [ADDR_WIDTH-1:0]	Input	Read address channel address.
s_axi_arprot [2:0]	Input	Read address channel protect.
s_axi_arid [ID_WIDTH-1:0]	Input	Read address channel transaction ID.
s_axi_arburst [1:0]	Input	Read address channel burst type.
s_axi_arlen [7:0]	Input	Read address channel burst length.
s_axi_arsize [2:0]	Input	Read address channel size.
s_axi_arcache [3:0]	Input	Read address channel cache encoding.
s_axi_arqos[3:0]	Input	Read address channel QoS.
s_axi_aruser [ARUSER_WIDTH-1:0]	Input	Read address channel user-defined signals.
s_axi_arlock [1:0]	Input	Read address channel locked transaction.
s_axi_arregion [3:0]	Input	Read address channel region identifier. Not applicable to AXI3.
s_axi_arready	Output	Read address channel ready.

Table 8: Slave Interface Read Data Channel

Port	Direction	Description
s_axi_rready	Input	Read data channel ready.
s_axi_rid [ID_WIDTH-1:0]	Output	Read data channel transaction ID.
s_axi_rdata [DATA_WDITH-1:0]	Output	Read data channel data.
s_axi_rresp [1:0]	Output	Read data channel read response.
s_axi_rvalid	Output	Read data channel valid.
s_axi_rlast	Output	Read data channel last data beat.
s_axi_ruser [RUSER_WIDTH-1:0]	Output	Read data channel user-defined channel.

Table 9: Master Interface Write Address Channel

Port	Direction	Description
m_axi_awvalid	Output	Write address channel valid.
m_axi_awaddr [ADDR_WIDTH-1:0]	Output	Write address channel address.
m_axi_awprot [2:0]	Output	Write address channel protect.
m_axi_awid [ID_WIDTH-1:0]	Output	Write address channel transaction ID.
m_axi_awburst [1:0]	Output	Write address channel burst type.
m_axi_awlen [7:0]	Output	Write address channel transaction length.
m_axi_awsize [2:0]	Output	Write address channel transfer size.
m_axi_awcache [3:0]	Output	Write address channel cache encoding.
m_axi_awqos [3:0]	Output	Write address channel QoS.
m_axi_awuser [AWUSER_WIDTH-1:0]	Output	Write address channel user-defined signal.
m_axi_awlock [1:0]	Output	Write address channel locked.
m_axi_awregion [3:0]	Output	Write address channel region identifier. Not applicable to AXI3.
m_axi_awready	Input	Write address channel ready.

Table 10: Master Interface Write Data Channel

Port	Direction	Description
m_axi_wvalid	Output	Write data channel valid.
m_axi_wdata [DATA_WDITH-1:0]	Output	Write data channel data.
m_axi_wstrb [DATA_WDITH/8-1:0]	Output	Write data channel strobe. Single bit represents data byte.
m_axi_wlast	Output	Write data channel last data beat.
m_axi_wuser [WUSER_WIDTH-1:0]	Output	Write data channel user-defined signals.
m_axi_wid [ID_WIDTH-1:0]	Output	Write data channel transaction ID.
m_axi_wready	Input	Write data channel ready.

Table 11: Master Interface Write Response Channel

Port	Direction	Description
m_axi_bready	Output	Write response channel ready.
m_axi_bresp [1:0]	Input	Write response channel response.
m_axi_bvalid	Input	Write response channel valid.
m_axi_bid [ID_WIDTH-1:0]	Input	Write response channel transaction ID.
m_axi_buser [BUSER_WIDTH-1:0]	Input	Write response channel user-defined signals.

Table 12: Master Interface Read Address Channel

Port	Direction	Description
m_axi_arvalid	Output	Read address channel valid.
m_axi_araddr [ADDR_WIDTH-1:0]	Output	Read address channel address.
m_axi_arprot [2:0]	Output	Read address channel protect.
m_axi_arid [ID_WIDTH-1:0]	Output	Read address channel transaction ID.
m_axi_arburst [1:0]	Output	Read address channel burst type.
m_axi_arlen [7:0]	Output	Read address channel burst length.
m_axi_arsize [2:0]	Output	Read address channel transfer size.
m_axi_arcache [3:0]	Output	Read address channel cache encoding.
m_axi_arqos [3:0]	Output	Read address channel QoS.
m_axi_aruser	Output	Read address channel user define signals.
m_axi_arlock [1:0]	Output	Read address channel locked.
m_axi_arregion [3:0]	Output	Read address channel region identifier.
m_axi_arready	Input	Read address channel ready.

Table 13: Master Interface Read Data Channel

Port	Direction	Description
m_axi_rready	Output	Read data channel ready.
m_axi_rid [ID_WIDTH-1:0]	Input	Read data channel transaction ID.
m_axi_rdata [DATA_WDITH-1:0]	Input	Read data channel data.
m_axi_rresp [1:0]	Input	Read data channel response.
m_axi_rvalid	Input	Read data channel valid.
m_axi_rlast	Input	Read data channel last data beat.
m_axi_ruser [RUSER_WIDTH-1:0]	Input	Read data channel user define signals.

Packet Mode Operations

You can enable the packet mode by enabling the **Write Address FIFO** or **Read Address FIFO** parameter in the IP Manager. When enabled, you will observe a 2-clock cycle delay between the slave and the master address channel. The write and read address channel acts like a pass through when the packet mode is disabled

The following figures describe timing sequences on the AXI write and read address channels with the address FIFO enabled and disabled.
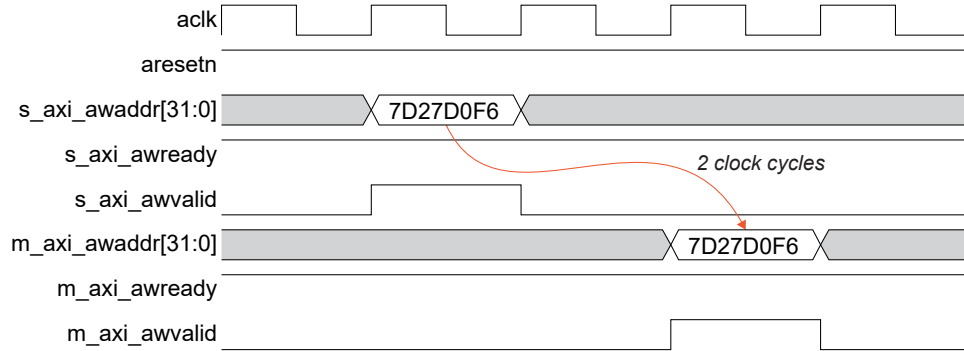
Figure 2: Write Address FIFO Enabled

Figure 3: Write Address FIFO Disabled

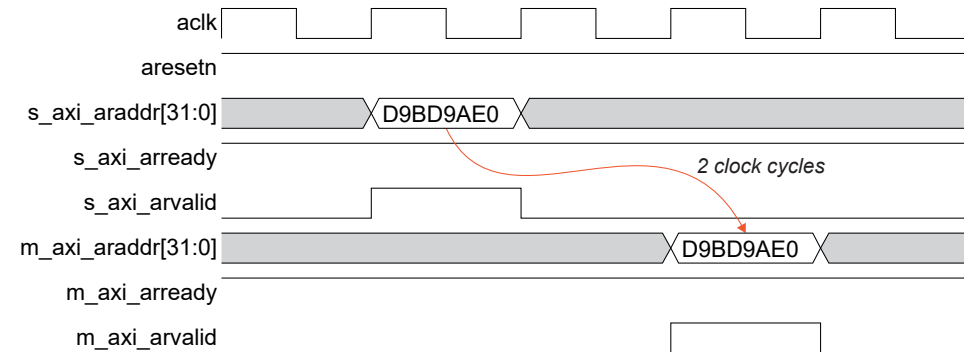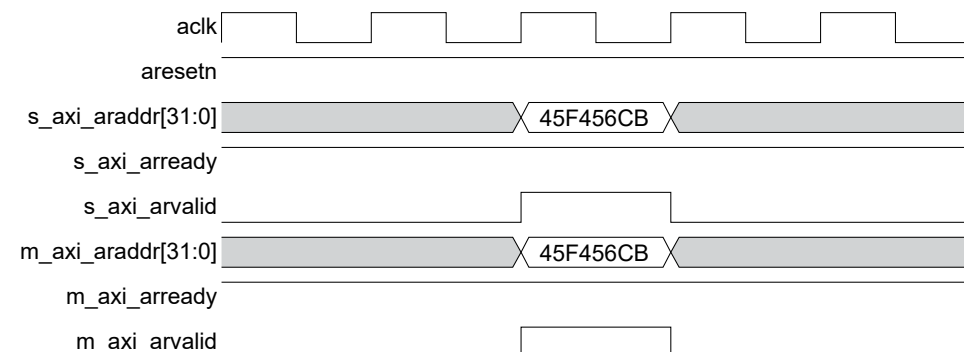*Figure 4: Read Address FIFO Enabled*

Figure 5: Read Address FIFO Disabled

IP Manager

The Efinity® IP Manager is an interactive wizard that helps you customize and generate Efinix® IP cores. The IP Manager performs validation checks on the parameters you set to ensure that your selections are valid. When you generate the IP core, you can optionally generate an example design targeting an Efinix development board and/or a testbench. This wizard is helpful in situations in which you use several IP cores, multiple instances of an IP core with different parameters, or the same IP core for different projects.

> **Note:** Not all Efinix IP cores include an example design or a testbench.

Generating the AXI Data FIFO Core with the IP Manager

The following steps explain how to customize an IP core with the IP Configuration wizard.

1. Open the IP Catalog.
2. Choose **AXI Infrastructure** > **AXI Data FIFO** core and click **Next**. The **IP Configuration** wizard opens.
3. Enter the module name in the **Module Name** box.

> **Note:** You cannot generate the core without a module name.

4. Customize the IP core using the options shown in the wizard. For detailed information on the options, refer to the *Customizing the AXI Data FIFO* section.
5. (Optional) In the **Deliverables** tab, specify whether to generate an IP core example design targeting an Efinix® development board and/or testbench. These options are turned on by default.
6. (Optional) In the **Summary** tab, review your selections.
7. Click **Generate** to generate the IP core and other selected deliverables.
8. In the **Review configuration generation** dialog box, click **Generate**. The Console in the **Summary** tab shows the generation status.

> **Note:** You can disable the **Review configuration generation** dialog box by turning off the **Show Confirmation Box** option in the wizard.

9. When generation finishes, the wizard displays the **Generation Success** dialog box. Click **OK** to close the wizard.

The wizard adds the IP to your project and displays it under **IP** in the Project pane.

Generated Files

The IP Manager generates these files and directories:
- **<module name>_define.vh**—Contains the customized parameters.
- **<module name>_tmpl.v**—Verilog HDL instantiation template.
- **<module name>_tmpl.vhd**—VHDL instantiation template.
- **<module name>.v**—IP source code.
- **settings.json**—Configuration file.
- **<kit name>_devkit**—Has generated RTL, example design, and Efinity® project targeting a specific development board.
- **Testbench**—Contains generated RTL and testbench files.

Customizing the AXI Data FIFO

The core has parameters so you can customize its function. You set the parameters in the General tab of the core's IP Configuration window.

Table 14: AXI Data FIFO Parameters

Parameters	Option	Description
PROTOCOL	AXI4, AXI3	Select the AXI protocol. Default: AXI4
READ WRITE MODE	READ_WRITE, READ, WRITE	Select the AXI Data FIFO mode. Default: READ_WRITE
Address Width	1 - 64	Set the address channel address width. Default: 32
Data Width	32, 64 ,128, 256, 512, 1024	Select the write/read channel data width. Default: 32
ID width	0 - 32	Set the ID transaction width. Default: 4
User Write Address Width	0 - 1024	Set the user write address width. Default: 0
User Read Address Width	0 - 1024	Set the user read address width. Default: 0
User Write Width	0 - 1024	Set the user write width. Default: 0
User Read Width	0 - 1024	Set the user read width. Default: 0
User Response Width	0 - 1024	Set the write response width. Default: 0
Write FIFO Depth	0, 32, 512	Select the write channel FIFO depth. Default: 512
Write Address FIFO	Enable, Disable	Enable or disable the write address FIFO. Enabling this triggers the packet FIFO mode. Available only when **Write FIFO Depth** is 512. Default: Disable
Read FIFO Depth	0, 32, 512	Select the read channel FIFO depth. Default: 512
Read Address FIFO	Enable, Disable	Enable or disable the read address FIFO. Enabling this triggers the packet FIFO mode. Available only when **Read FIFO Depth** is 512. Default: Disable

AXI Data FIFO Testbench

You can choose to generate the testbench when generating the core in the IP Manager Configuration window.

Note: You must include all **.v** files generated in the **/testbench** directory in your simulation.

Efinix provides a simulation script for you to run the testbench quickly using the Modelsim software. To run the Modelsim testbench script, run `vsim -do modelsim.do` in a terminal application. You must have Modelsim installed on your computer to use this script.

The testbench is set up like an example design, consisting of the following blocks:

- *Custom Master*—Main driver and sequencer in the testbench
- *Custom Slave*—Provides necessary feedback such as read data and handshake signals to the master through the DUT
- *Generator*—Randomly generates the data used for all transactions made by both the custom master and slave
- *Checker*—Probes all DUT inputs and outputs, internally performs similar logic to the DUT to predict the outputs, and asserts `test_pass` signal if the test is completed without any prediction mismatches with the DUT outputs
- *DUT*—AXI Data FIFO design under test

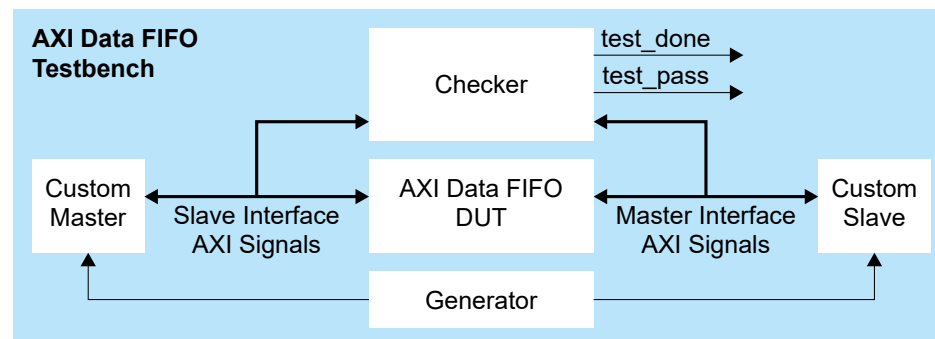*Figure 6: AXI Data FIFO Top-Level Block Diagram*

Table 15: Testbench Files

File	Description
tb.v	Testbench to generate clock and pass down parameters to top.v.
top.v	Top-level wrapper for the DUT environment.
efx_axi_data_fifo_checker.v	Probe the DUT's input and outputs signals to make sure that the data transferred through the data FIFO is consistent with the expected output.
generator.v	Generates random data for the master and slave to be used as output for simulation.
efx_axi_master.v	A state machine that drives the scenarios used for simulation.
efx_axi_slave.v	A slave that acts like a memory and giving response based on the AXI protocol.
efx_crc32.v	CRC32 module used by the traffic generator and checker.
efx_fifo_top.*<simulator>*.v	Encrypted FIFO module.
<user_define_name>.v	Generated encrypted AXI Data FIFO core file based on user configuration in Efinity IP Manager.

Revision History

Table 16: Revision History

Date	Version	Description
June 2023	1.0	Initial release.