APB3 to AXI4 Lite Converter Core User Guide

UG-CORE-APB3-AXI4LITE-v3.3
February 2023
www.efinixinc.com

Contents

Introduction

The APB3 to AXI4 Lite Converter core translates APB3 transactions into AXI4-Lite transactions. The core functions as a slave on the APB3 interface and as a master on the AXI4-Lite interface. The APB3 to AXI4 Lite Converter core supports a single write transfer or read transfer at a time.

Use the IP Manager to select IP, customize it, and generate files. The APB3 to AXI4 Lite Converter 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

- Complies with AMBA AXI4-Lite specification (ARM IHI 0022D) and the APB3 specification (ARM IHI 0024B)
- 32-bit slave on a 32-bit APB3 interface
- 32-bit master on a 32-bit AXI4-Lite interface
- Supports no wait write, wait write, no wait read, and wait read operations
- Verilog HDL RTL and simulation testbench
- Includes example designs targeting the Trion® T20 BGA256 Development Board and Titanium Ti60 F225 Development Board

FPGA Support

The APB3 to AXI4 Lite Converter core supports all Trion® and Titanium FPGAs.

Resource Utilization and Performance

Note: The resources and performance values provided are just guidance and change depending on the device resource utilization, design congestion, and user design.

Titanium Resource Utilization and Performance

FPGA	Address Width (bit)	Logic/ Adders	Flipflops	Memory Blocks	DSP48 Blocks	f_{MAX} (MHz)(1)	Efinity® Version(2)
Ti60 F225 C4	10	29	100	0	0	656	2021.2
	16	30	112	0	0	578	
	32	29	144	0	0	580	

Trion Resource Utilization and Performance

FPGA	Address Width (bit)	Logic Utilization (LUTs)	Registers	Memory Blocks	Multipliers	f_{MAX} (MHz)(1)	Efinity® Version(2)
T20 BGA256 C4	10	36	100	0	0	265	2021.1
	16	36	112	0	0	289	
	32	36	144	0	0	289	

---

(1) Using default parameter settings.
(2) Using Verilog HDL.

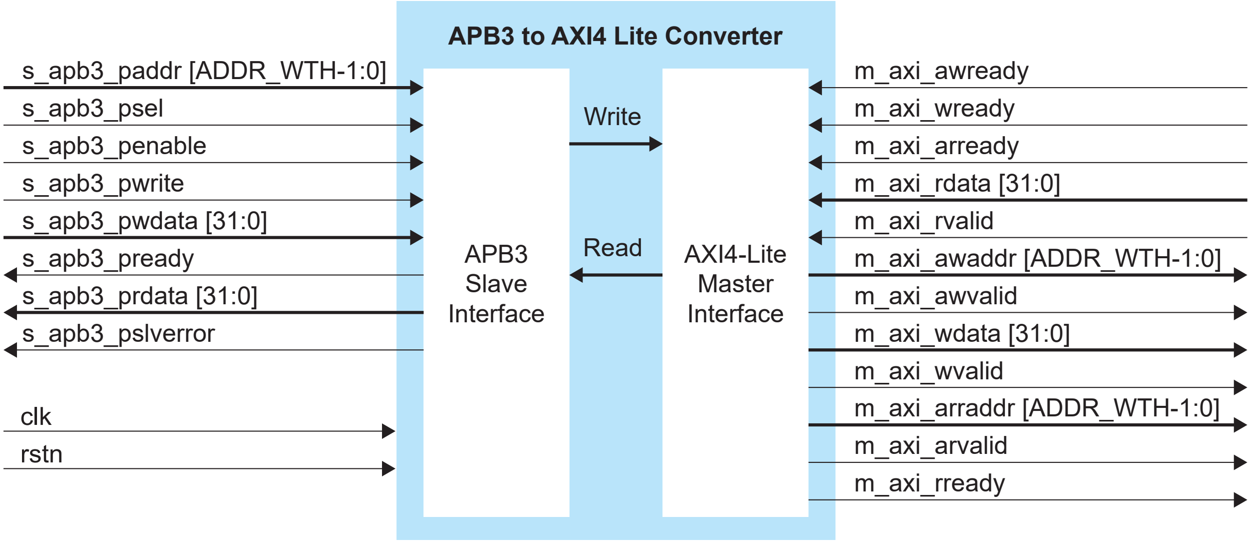
Functional Description

The APB3 to AXI4 Lite Converter core includes an error detection feature. In the case of timeout (128 clock cycles), that is the AXI4-Lite interface is not responding, the APB3 slave interface asserts the `s_apb3_pslverror` signal high.

Note: Efinix® recommends that you set the delay clock cycle not more than 123 when using wait write or wait read operation. This allows at least five clock cycles for the write or read operation to complete before reaching the timeout.

- *APB3 Slave interface*—Provides a bidirectional slave interface to the core. The APB3 data bus widths are always fixed at 32 bits, while address width can vary from 2 to 32 bits. The APB3 interface performs write transfers when the `s_apb3_pwrite` signal is high and read transfers when the `s_apb3_pwrite` signal is low.
- *AXI4 Master Interface*—Provides the AXI4-lite master interface to the core. The data bus widths are fixed at 32 bits, while the address is equal to the APB3 address width.

Figure 1: APB3 to AXI4 Lite Converter System Block Diagram



Ports

Table 1: APB3 Ports

Name	Direction	Description
s_apb3_paddr[ADDR_WTH-1:0]	Input	APB3 address bus. It can be up to 32 bits wide and is driven by the APB3 master device. Write and read operations share the same address bus
s_apb3_psel	Input	APB3 master device generates this signal to each peripheral bus slave. This signal must always be high for write or read operations.
s_apb3_penable	Input	Enables the second and subsequent cycles of an APB3 transfers.
s_apb3_pwrite	Input	0: Read transfer 1: Write transfer
s_apb3_pwdata[31:0]	Input	APB3 write data bus.
s_apb3_pready	Output	Logic high indicates that the slave is ready to receive the next transfer.
s_apb3_prdata[31:0]	Output	APB3 read data bus.
s_apb3_pslverror	Output	Logic high indicates that there is no response from the AXI4 interface.

Table 2: AXI4-Lite Ports

Name	Direction	Description
m_axi_awready	Input	Indicates that the channel is signaling valid write address and control information.
m_axi_wready	Input	Indicates that the slave can accept the write data.
m_axi_arready	Input	Indicates that the slave is ready to accept an address and associated control signals.
m_axi_rdata[31:0]	Input	Content of the read data.
m_axi_rvalid	Input	Indicates that the channel is signaling valid read address and control information.
m_axi_awaddr[ADDR_WTH-1:0]	Output	Copy address of the APB3 on write transfer.
m_axi_awvalid	Output	Indicates that the valid read data is available.
m_axi_wdata[31:0]	Output	Data to be written.
m_axi_wvalid	Output	Indicates that the channel is signaling valid write address and control information.
m_axi_arraddr[ADDR_WTH-1:0]	Output	Copy address of the APB3 on read transfer.
m_axi_arvalid	Output	Indicates that a valid read data is available.
m_axi_rready	Output	Indicates that the channel is signaling the required read data. This signal is constantly high.

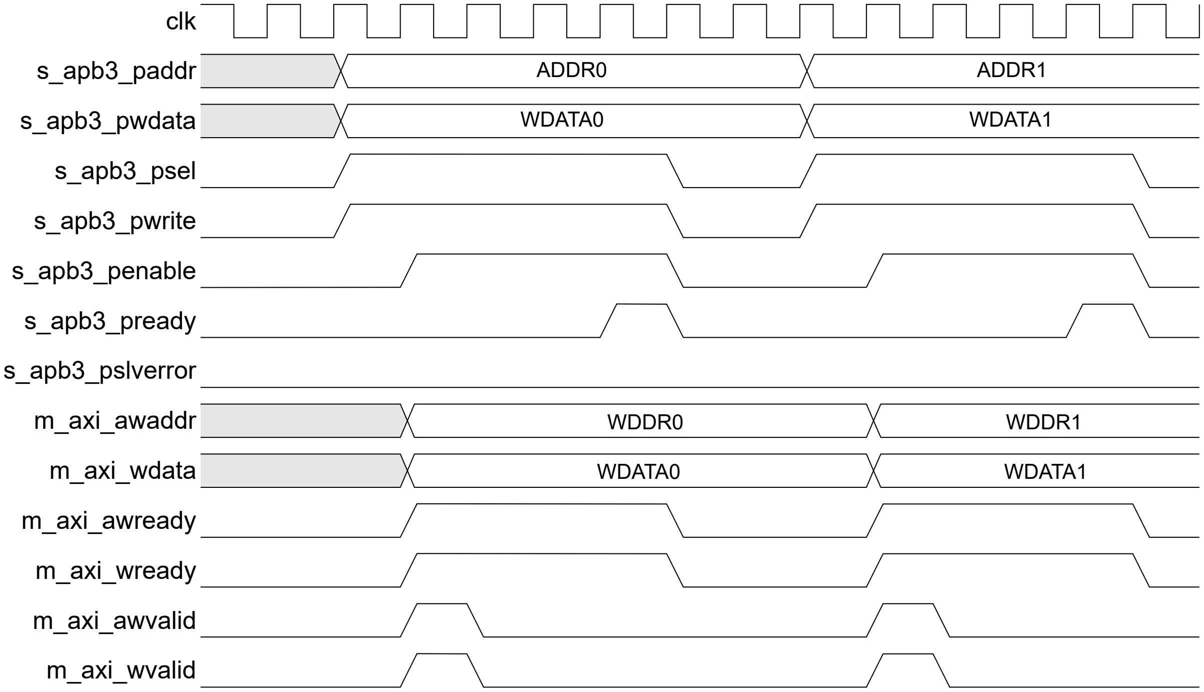
Table 3: Clock and Reset Ports

Name	Direction	Description
clk	Input	All signals are synchronous to this clock.
rstn	Input	Synchronous reset signal that initializes all internal pointers and output flags.

APB3 to AXI4 Lite Converter Operations

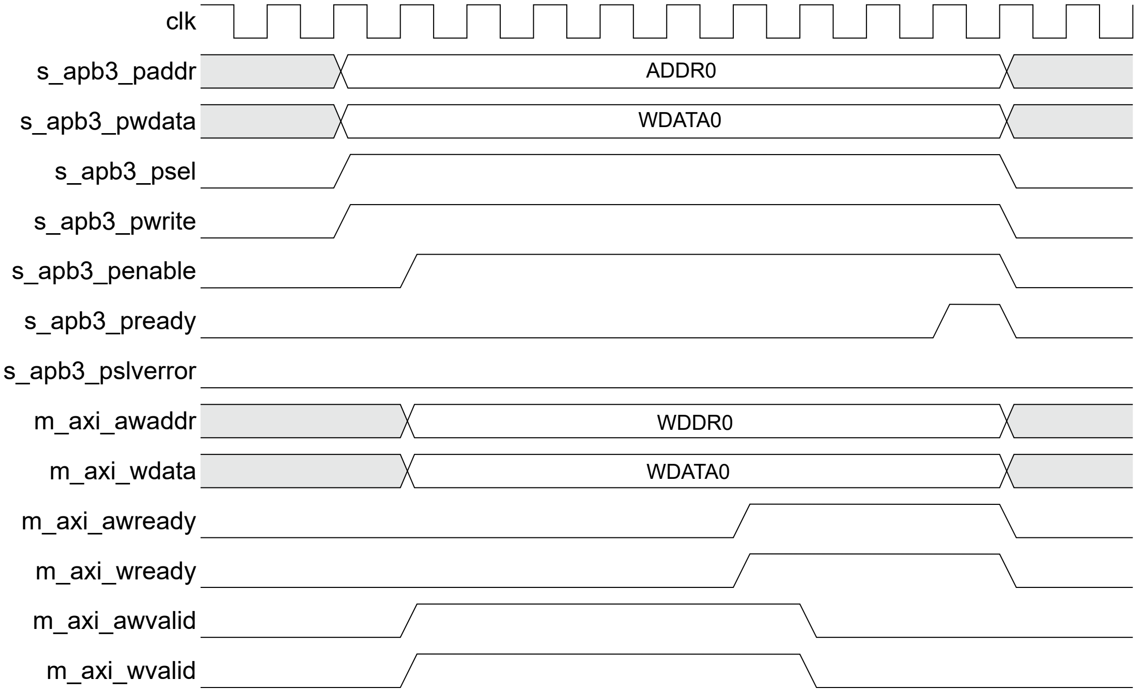
Assert the `s_apb3_penable` one clock cycle after asserting `s_apb3_psel` and `s_apb3_pwrite` to start the write operation. The write operation is complete when the `s_apb3_pready` toggles to high then low, five cycles after `s_apb3_psel` and `s_apb3_pwrite` go high.

Figure 2: Write Operation Waveform (ADDR_WTH = 10)



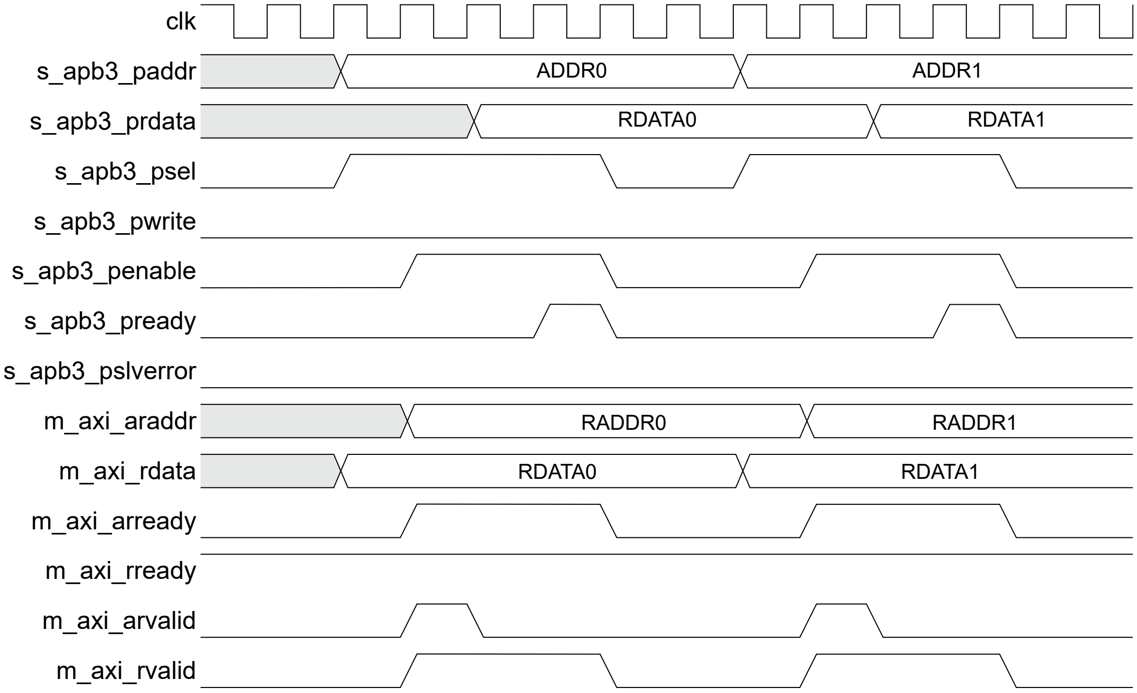
Assert the `s_apb3_penable` one clock cycle after asserting `s_apb3_psel` and `s_apb3_pwrite` to start the wait write operation. Then, assert the `m_axi_awready` and `m_axi_wready` signals after the number of wait delay cycles. The write operation is complete when the `s_apb3_pready` toggles to high then low, five plus the wait delay cycles after `s_apb3_psel` and `s_apb3_pwrite` go high.

Figure 3: Wait Write Operation Waveform (ADDR_WTH = 10, wait 5 clock cycle)



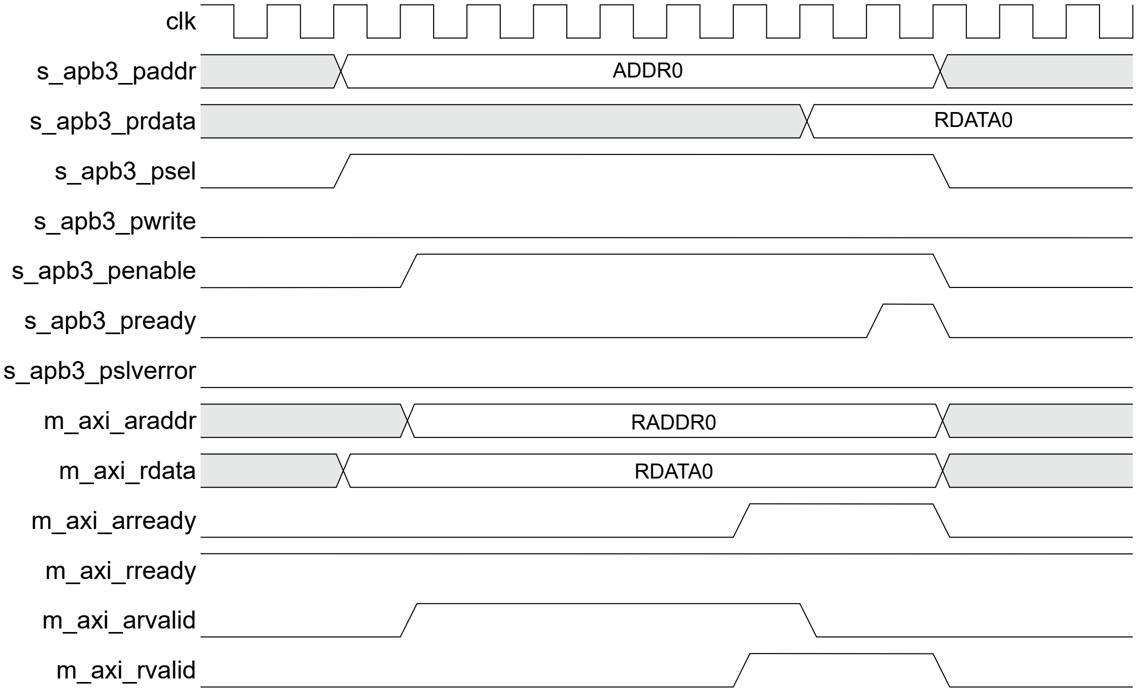
Assert the `s_apb3_penable` one clock cycle after asserting `s_apb3_psel` to start the read operation. The read operation is complete when the `s_apb3_pready` toggles to high then low, four cycles after `s_apb3_psel` goes high.

Figure 4: Read Operation Waveform (ADDR_WTH = 10)



Assert the `s_apb3_penable` one clock cycle after asserting `s_apb3_psel` to start the read operation. Then, assert the `m_axi_arready` signal after the number of wait delay cycles. The read operation is complete when the `s_apb3_pready` toggles to high then low, four plus the wait delay cycles after `s_apb3_psel` goes high.

Figure 5: Read Write Operation Waveform (ADDR_WTH = 10, wait 5 clock cycle)



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 a 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 an IP 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 IP core's user guide or on-line help.
5. (Optional) In the **Deliverables** tab, specify whether to generate an IP core example design targeting an Efinix® development board and/or testbench. For SoCs, you can also optionally generate embedded software example code. 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.

Note: Refer to the IP Manager chapter of the Efinity® Software User Guide for more information about the Efinity® IP Manager.

Customizing the APB3 to AXI4 Lite Converter

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 4: APB3 to AXI4 Lite Converter Core Parameter

Parameters	Options	Description
Address Width	2 - 32	Defines the APB3 and AXI4-Lite address width. Default = 10

APB3 to AXI4 Lite Converter Example Design

You can choose to generate the example design when generating the core in the IP Manager Configuration window. Compile the example design project and download the **.hex** or **.bit** file to your board.

Important: Efinix tested the example design generated with the default parameter options only.

The example design targets the Trion® T20 BGA256 Development Board and Titanium Ti60 F225 Development Board. This design continuously performs write and read operations. The design then compares the data from each operation. The design displays the operation type and flags if it detects an error in the data or address of an operation through the board's LEDs.

- Trion® T20 BGA256 Development Board—If there are no errors, the LEDs blink from LEDs D2, and D3 to LED D2 continuously.
- Titanium Ti60 F225 Development Board—If there are no errors, the LEDs blink from LEDs D16 blue and D17 blue to LED D16 green continuously.

Figure 6: APB3 to AXI4 Lite Converter Core Example Design

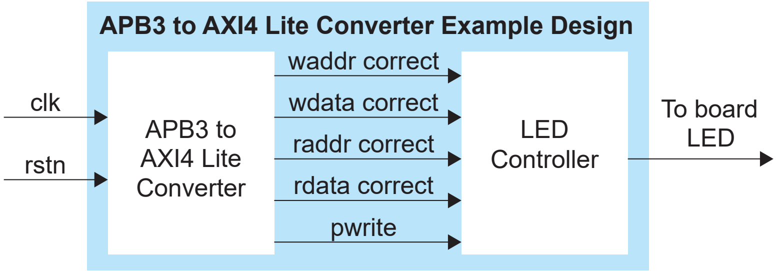


Table 5: Trion® Example Design Implementation

FPGA	LUTs	Registers	Memory Blocks	Multipliers	f_{MAX} (MHz)[3]	Efinity® Version[4]
T20 BGA256 C4	82	53	0	0	171	2020.1

Table 6: Titanium Example Design Implementation

FPGA	Logic and Adders	Flip-flops	Memory Blocks	DSP48 Blocks	f_{MAX} (MHz)[3]	Efinity® Version[4]
Ti60 F225 C4	242	248	0	0	390	2021.2

APB3 to AXI4 Lite Converter 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 in your computer to use this script.

The testbench includes several tests that use either randomly generated data or user input data. The testbench also includes a comparator to indicate a pass/fail for the read and write operations.

Table 7: Test Types and Input Type

Test	Description	Input Type
Write	Single write test	Random or user input
Read	Single read test	Random or user input
Continuous Write	8 (default) cycles write test	Random
Continuous Read	8 (default) cycles read test	Random
Write Wait	Single wait write test	User input
Read Wait	Single wait read test	User input
Write then Read	Single write then read test	User input
Error	Error test	User input

After running the write simulation, the test prints the following message:

```
80000- PASS! the output write address is correct
80000- PASS! the output write data is correct
```

After running the read simulation, the test prints the following message:

```
480000- PASS! the output read address is correct
510000- PASS! the output read data is correct
```

---

[3] Using default parameter settings.
[4] Using Verilog HDL.

After running the error simulation, the test prints the following message:

```
5980000- PASS! the signal is correct on State Error
```

Revision History

Table 8: Revision History

Date	Version	Description
February 2023	3.3	Added note about the resource and performance values in the resource and utilization table are for guidance only.
January 2022	3.2	Updated resource utilization table. (DOC-700)
October 2021	3.1	Added note to state that the f_{MAX} in Resource Utilization and Performance, and Example Design Implementation tables were based on default parameter settings. Updated design example target board to production Titanium Ti60 F225 Development Board and updated Resource Utilization and Performance, and Example Design Implementation tables. (DOC-553)
June 2021	3.0	Added note about including all **.v** generated in testbench folder is required for simulation. Updated resource utilization and performance table. Updated example design output and implementation table. Added support for Titanium FPGAs and example design for Titanium Ti60 F225 Development Board. Updated for Efinity v2021.1.
December 2020	2.0	Update core name to APB3 to AXI4 Lite Converter. Updated user guide for Efinix® IP Manager which includes added IP Manager topics, updated parameters, and user guide structure.
September 2020	1.0	Initial release.