AI Engine v2.0

LogiCORE IP Product Guide

Vivado Design Suite

PG358 (v2.0) December 8, 2020

Table of Contents

Introduction

The Xilinx® LogiCORE™ AI Engine IP enables the configuration of the AI Engine Array Interface. This array is connected to the Network on Chip and to the programmable logic (PL) through tiles that are located in the AI Engine Array Interface. This IP allows the specification of the number of AXI4-Stream and memory mapped AXI interfaces with their respective width and direction and defines the clock driving the AI Engine array.

For detailed information on the Versal™ ACAP AI Engine, refer to the *Versal ACAP AI Engine Architecture Manual* (AM009).

Features

- Memory mapped AXI4 configuration interfaces.
- Memory mapped AXI4 master interfaces.
- AXI4-Stream master and slave interfaces with configurable data widths of 32, 64, or 128 bits.
- Option to enable registered interfaces to enable fast streams.
- Automatic association of programmable logic (PL) stream clocks.

IP Facts

LogiCORE™ IP Facts Table	
Core Specifics	
Supported Device Family[1]	Versal™ ACAP AI Core
Supported User Interfaces	AXI4, AXI4-Stream
Resources	N/A
Provided with Core	
Design Files	Register Transfer Level (RTL)
Example Design	Not Provided
Test Bench	Not Provided
Constraints File	Not Provided
Simulation Model	Not Provided
Supported S/W Driver	N/A
Tested Design Flows[2]	
Design Entry	Vivado® Design Suite
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Vivado Synthesis
Support	
Release Notes and Known Issues	Master Answer Record: 75675
All Vivado IP Change Logs	Master Vivado IP Change Logs: 72775
Xilinx Support web page	

Notes:

1. For a complete list of supported devices, see the Vivado® IP catalog.
2. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

XILINX

Chapter 2

Overview

Navigating Content by Design Process

Xilinx® documentation is organized around a set of standard design processes to help you find relevant content for your current development task. This document covers the following design processes:

- **Hardware, IP, and Platform Development:** Creating the PL IP blocks for the hardware platform, creating PL kernels, subsystem functional simulation, and evaluating the Vivado® timing, resource use, and power closure. Also involves developing the hardware platform for system integration. Topics in this document that apply to this design process include:
  - Port Descriptions
  - Clocking
  - Customizing and Generating the Core

Applications

The AI Engine IP core is used in applications such as:

- Application designs using the AI Engines in the AI Engine array
- Wireless and wired communications
- Machine learning
- Aerospace and defence

Licensing and Ordering

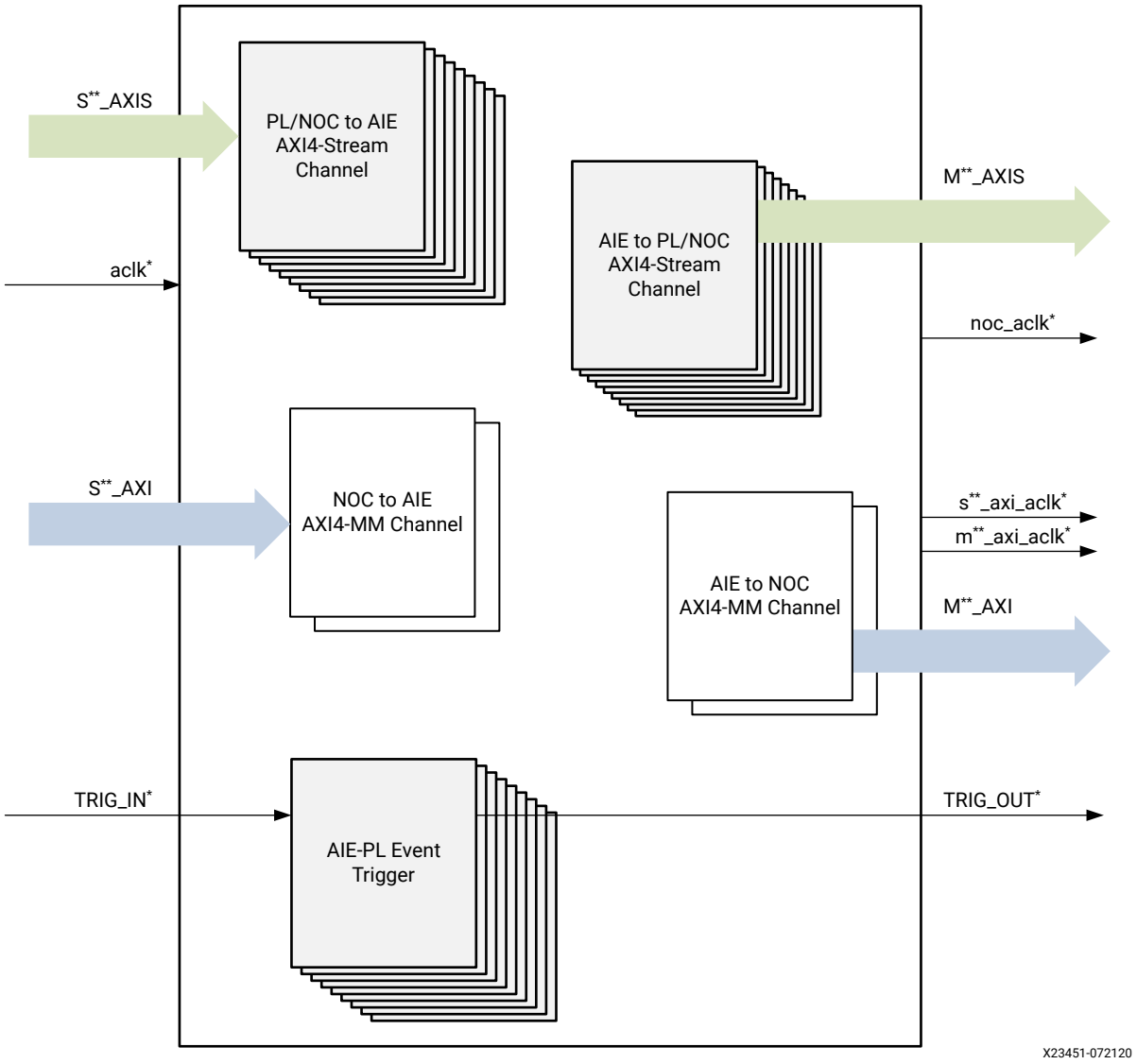
This Xilinx® LogiCORE™ IP module is provided at no additional cost with the Xilinx Vivado® Design Suite under the terms of the Xilinx End User License.

Information about 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

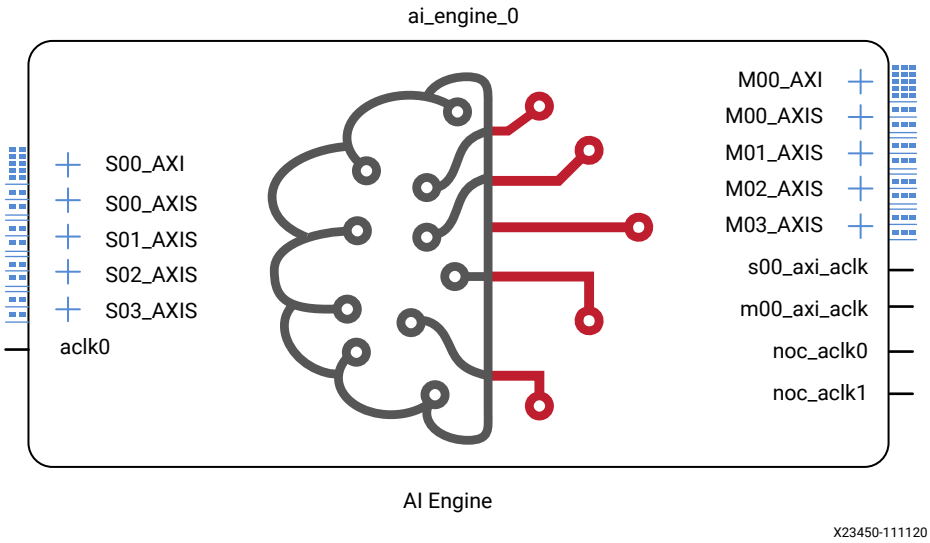
The functional block diagram of the core is shown in the following figure.

Figure 1: **Core Block Diagram**



The IP symbol of the AI Engine core is shown in the following figure:

Figure 2: **IP Symbol**



Port Descriptions

The following sections show the AXI ports and their descriptions.

M**_AXI Interface Ports

The following table shows the M**_AXI Interface Ports and their descriptions.

Table 1: **M**_AXI Interface Ports**

Port Name	I/O	Description
m**_axi_awaddr (C_M**_AXI_ADDR_WIDTH – 1: 0)	O	Write Address Channel Address Bus
m**_axi_awid (C_M**_AXI_ID_WIDTH – 1: 0)	O	Write Address Channel ID
m**_axi_awuser (C_M**_AXI_AWUSER_WIDTH – 1: 0)	O	Write Address Channel user
m**_axi_awlen[7:0]	O	Write Address Channel Burst Length. In data beats – 1.

Table 1: **M**_AXI Interface Ports** *(cont'd)*

Port Name	I/O	Description
m**_axi_awsize[2: 0]	O	Write Address Channel Burst Size. Indicates width of burst transfer. • 000b = 1-byte (8-bit wide burst) • 001b = 2 bytes (16-bit wide burst) • 010b = 4 bytes (32-bit wide burst) • 011b = 8 bytes (64-bit wide burst) • 100b = 16 bytes (128-bit wide burst) • 101b = 32 bytes (256-bit wide burst) • 110b = 64 bytes (512-bit wide burst) • 111b = 128 bytes (1,024-bit wide burst)
m**_axi_awburst[1:0]	O	Write Address Channel Burst Type. Indicates type burst. • 00b = FIXED – Fixed address • 01b = INCR – Incrementing address • 10b = WRAP – Not supported • 11b = Reserved
m**_axi_awprot[2:0]	O	Write Address Channel Protection. This is always driven with a constant output of 0010b.
m**_axi_awlock	O	Write Address Channel exclusive access.
m**_axi_awqos[3:0]	O	Write Address Channel QoS
m**_axi_awregion[3:0]	O	Write Address Channel Region
m**_axi_awcache[3:0]	O	Write Address Channel Cache
m**_axi_awvalid	O	Write Address Channel Write Address Valid. Indicates if m**_axi_awaddr is valid. • 0 = write address is not valid • 1 = write address is valid
m**_axi_awready	O	Write Address Channel Write Address Ready. Indicates target is ready to accept the write address. • 0 = target not ready to accept address • 1 = target ready to accept address
m**_axi_wdata (C_M**_AXI_DATA_WIDTH – 1:0)	O	Write Data Channel Write Data Bus
m**_axi_wstrb (C_M**_AXI_DATA_WIDTH/ 8 – 1:0)	O	Write Data Channel Write Strobe Bus. Indicates which bytes are valid in the write data bus. This value is passed from the stream side strobe bus.
m**_axi_wuser (C_M**_AXI_WUSER_WIDTH – 1: 0)	O	Write Data Channel user
m**_axi_wlast	O	Write Data Channel Last. Indicates the last data beat of a burst transfer. • 0 = Not last data beat • 1 = Last data beat
m**_axi_wvalid	O	Write Data Channel Data Valid. Indicates m**_axi_wdata is valid. • 0 = Not valid write data • 1 = Valid write data

Table 1: **M**_AXI Interface Ports** *(cont'd)*

Port Name	I/O	Description
m**_axi_wready	I	Write Data Channel Ready. Indicates the write channel target is ready to accept write data. • 0 = target is not ready • 1 = target is ready
m**_axi_bresp[1:0]	I	Write Response Channel Response. Indicates results of the write transfer. • 00b = OKAY – Normal access has been successful • 01b = EXOKAY – Not supported • 10b = SLVERR – Slave returned error on transfer • 11b = DECERR – Decode error, transfer targeted unmapped address
m**_axi_bvalid	I	Write Response Channel Response Valid. Indicates response, m**_axi_bresp, is valid. • 0 = Response is not valid • 1 = Response is valid
m**_axi_bid (C_M**_AXI_ID_WIDTH – 1: 0)	I	Write Response Channel ID
m**_axi_buser (C_M**_AXI_BUSER_WIDTH – 1: 0)	I	Write Response Channel user
m**_axi_bready	O	Write Response Channel Ready. Indicates write channel is ready to receive response. • 0 = Not ready to receive response • 1 = Ready to receive response
m**_axi_araddr (C_M**_AXI_ADDR_WIDTH – 1:0)	O	Read Address Channel Address Bus
m**_axi_arid (C_M**_AXI_ID_WIDTH – 1: 0)	O	Read Address Channel ID
m**_axi_aruser (C_M**_AXI_ARUSER_WIDTH – 1: 0)	O	Read Address Channel user
m**_axi_arlen[7:0]	O	Read Address Channel Burst Length. In data beats – 1.
m**_axi_arsize[2:0]	O	Read Address Channel Burst Size. Indicates width of burst transfer. • 000b = 1-byte (8-bit wide burst) • 001b = 2 bytes (16-bit wide burst) • 010b = 4 bytes (32-bit wide burst) • 011b = 8 bytes (64-bit wide burst) • 100b = 16 bytes (128-bit wide burst) • 101b = 32 bytes (256-bit wide burst) • 110b = 64 bytes (512-bit wide burst) • 111b = 128 bytes (1,024-bit wide burst)
m**_axi_arburst[1:0]	O	Read Address Channel Burst Type. Indicates type burst. • 00b = FIXED – Fixed address • 01b = INCR – Incrementing address • 10b = WRAP – Not supported • 11b = Reserved

Table 1: **M**_AXI Interface Ports** *(cont'd)*

Port Name	I/O	Description
m**_axi_arprot[2:0]	O	Read Address Channel Protection. This is always driven with a constant output of 0010b.
m**_axi_arcache[3:0]	O	Read Address Channel Cache
m**_axi_arlock	O	Read Address Channel exclusive access.
m**_axi_arqos[3:0]	O	Read Address Channel QoS
m**_axi_arregion[3:0]	O	Read Address Channel Region
m**_axi_arvalid	O	Read Address Channel Read Address Valid. Indicates if m**_axi_araddr is valid. 0 = write address is not valid 1 = write address is valid
m**_axi_arready	I	Read Address Channel Read Address Ready. Indicates target is ready to accept the read address. • 0 = Target not ready to accept address • 1 = Target ready to accept address
m**_axi_rdata (C_M**_AXI_DATA_WIDTH – 1:0)	I	Read Data Channel Read Data Bus
m**_axi_rid (C_M**_AXI_ID_WIDTH – 1: 0)	I	Read Data Channel ID
m**_axi_ruser (C_M**_AXI_RUSER_WIDTH – 1: 0)	I	Read Data Channel user
m**_axi_rlast	I	Read Data Channel Last. Indicates the last data beat of a burst transfer. • 0 = Not last data beat • 1 = Last data beat
m**_axi_rvalid	I	Read Data Channel Data Valid. Indicates m**_axi_rdata is valid. • 0 = Not valid write data • 1 = Valid write data
m**_axi_rready	O	Read Data Channel Ready. • 0 = Target is not ready • 1 = Target is ready
m**_axi_rresp[1:0]	I	Read Response Channel Response. Indicates results of the write transfer. • 00b = OKAY – Normal access has been successful • 01b = EXOKAY – Not supported • 10b = SLVERR – Slave returned error on transfer • 11b = DECERR – Decode error, transfer targeted unmapped address

S**_AXI Interface Ports

The following table shows the S**_AXI Interface Ports and their descriptions.

Table 2: **S**_AXI Interface Ports**

Port Name	I/O	Description
s**_axi_awaddr (C_S**_AXI_ADDR_WIDTH – 1: 0)	I	Write Address Channel Address Bus
s**_axi_awid (C_S**_AXI_ID_WIDTH – 1: 0)	I	Write Address Channel ID
s**_axi_awuser (C_S**_AXI_AWUSER_WIDTH – 1: 0)	I	Write Address Channel user
s**_axi_awlen[7:0]	I	Write Address Channel Burst Length. In data beats – 1.
s**_axi_awsize[2: 0]	I	Write Address Channel Burst Size. Indicates width of burst transfer. • 000b = 1-byte (8-bit wide burst) • 001b = 2 bytes (16-bit wide burst) • 010b = 4 bytes (32-bit wide burst) • 011b = 8 bytes (64-bit wide burst) • 100b = 16 bytes (128-bit wide burst) • 101b = 32 bytes (256-bit wide burst) • 110b = 64 bytes (512-bit wide burst) • 111b = 128 bytes (1,024-bit wide burst)
s**_axi_awburst[1:0]	I	Write Address Channel Burst Type. Indicates type burst. • 00b = FIXED – Fixed address • 01b = INCR – Incrementing address • 10b = WRAP – Not supported • 11b = Reserved
s**_axi_awprot[2:0]	I	Write Address Channel Protection. This is always driven with a constant output of 0010b.
s**_axi_awlock	I	Write Address Channel exclusive access.
s**_axi_awqos[3:0]	I	Write Address Channel QoS
s**_axi_awregion[3:0]	I	Write Address Channel Region
s**_axi_awcache[3:0]	I	Write Address Channel Cache
s**_axi_awvalid	I	Write Address Channel Write Address Valid. Indicates if m**_axi_awaddr is valid. • 0 = Write address is not valid • 1 = Write address is valid
s**_axi_awready	O	Write Address Channel Write Address Ready. Indicates target is ready to accept the write address. • 0 = Target not ready to accept address • 1 = Target ready to accept address
s**_axi_wdata (C_S**_AXI_DATA_WIDTH – 1:0)	I	Write Data Channel Write Data Bus

Table 2: **S**_AXI Interface Ports** *(cont'd)*

Port Name	I/O	Description
s**_axi_wstrb (C_S**_AXI_DATA_WIDTH/ 8 – 1:0)	I	Write Data Channel Write Strobe Bus. Indicates which bytes are valid in the write data bus. This value is passed from the stream side strobe bus.
s**_axi_wuser (C_S**_AXI_WUSER_WIDTH – 1: 0)	I	Write Data Channel user
s**_axi_wlast	I	Write Data Channel Last. Indicates the last data beat of a burst transfer. • 0 = Not last data beat • 1 = Last data beat
s**_axi_wvalid	I	Write Data Channel Data Valid. Indicates m**_axi_wdata is valid. • 0 = Not valid write data • 1 = Valid write data
s**_axi_wready	O	Write Data Channel Ready. Indicates the write channel target is ready to accept write data. • 0 = Target is not ready • 1 = Target is ready
s**_axi_bresp[1:0]	O	Write Response Channel Response. Indicates results of the write transfer. • 00b = OKAY – Normal access has been successful • 01b = EXOKAY – Not supported • 10b = SLVERR – Slave returned error on transfer • 11b = DECERR – Decode error, transfer targeted unmapped address
s**_axi_bvalid	O	Write Response Channel Response Valid. Indicates response, m**_axi_bresp, is valid. 0 = response is not valid 1 = response is valid
s**_axi_bid (C_S**_AXI_ID_WIDTH – 1: 0)	O	Write Response Channel ID
s**_axi_buser (C_S**_AXI_BUSER_WIDTH – 1: 0)	O	Write Response Channel user
m**_axi_bready	I	Write Response Channel Ready. Indicates write channel is ready to receive response. • 0 = Not ready to receive response • 1 = Ready to receive response
s**_axi_araddr (C_S**_AXI_ADDR_WIDTH – 1:0)	I	Read Address Channel Address Bus
s**_axi_arid (C_S**_AXI_ID_WIDTH – 1: 0)	I	Read Address Channel ID
s**_axi_aruser (C_S**_AXI_ARUSER_WIDTH – 1: 0)	I	Read Address Channel user
s**_axi_arlen[7:0]	I	Read Address Channel Burst Length. In data beats – 1.

Table 2: **S**_AXI Interface Ports** *(cont'd)*

Port Name	I/O	Description
s**_axi_arsize[2:0]	I	Read Address Channel Burst Size. Indicates width of burst transfer. • 000b = 1-byte (8-bit wide burst) • 001b = 2 bytes (16-bit wide burst) • 010b = 4 bytes (32-bit wide burst) • 011b = 8 bytes (64-bit wide burst) • 100b = 16 bytes (128-bit wide burst) • 101b = 32 bytes (256-bit wide burst) • 110b = 64 bytes (512-bit wide burst) • 111b = 128 bytes (1,024-bit wide burst)
s**_axi_arburst[1:0]	I	Read Address Channel Burst Type. Indicates type burst. • 00b = FIXED – Fixed address • 01b = INCR – Incrementing address • 10b = WRAP – Not supported • 11b = Reserved
s**_axi_arprot[2:0]	I	Read Address Channel Protection. This is always driven with a constant output of 0010b.
s**_axi_arcache[3:0]	I	Read Address Channel Cache
s**_axi_arlock	I	Read Address Channel exclusive access.
s**_axi_arqos[3:0]	I	Read Address Channel QoS
s**_axi_arregion[3:0]	I	Read Address Channel Region
S**_axi_arvalid	I	Read Address Channel Read Address Valid. Indicates if m**_axi_araddr is valid. • 0 = Write address is not valid • 1 = Write address is valid
s**_axi_arready	O	Read Address Channel Read Address Ready. Indicates target is ready to accept the read address. • 0 = Target not ready to accept address • 1 = Target ready to accept address
s**_axi_rdata (C_S**_AXI_DATA_WIDTH – 1:0)	O	Read Data Channel Read Data Bus
s**_axi_rid (C_S**_AXI_ID_WIDTH – 1: 0)	O	Read Data Channel ID
s**_axi_ruser (C_S**_AXI_RUSER_WIDTH – 1: 0)	O	Read Data Channel user
s**_axi_rlast	O	Read Data Channel Last. Indicates the last data beat of a burst transfer. • 0 = Not last data beat • 1 = Last data beat
s**_axi_rvalid	O	Read Data Channel Data Valid. Indicates m**_axi_rdata is valid. • 0 = Not valid write data • 1 = Valid write data

Table 2: **S**_AXI Interface Ports** *(cont'd)*

Port Name	I/O	Description
s**_axi_rready	I	Read Data Channel Ready • 0 = Target is not ready • 1 = Target is ready
s**_axi_rresp[1:0]	O	Read Response Channel Response. Indicates results of the write transfer. • 00b = OKAY – Normal access has been successful • 01b = EXOKAY – Not supported • 10b = SLVERR – Slave returned error on transfer • 11b = DECERR – Decode error, transfer targeted unmapped address

M**_AXIS Interface Ports

The following table shows the M**_AXIS Interface Ports and their descriptions.

Table 3: **M**_AXIS Interface Ports**

Port Name	I/O	Description
m**_axis_tid (C_M**_AXIS_TID_WIDTH – 1:0)	O	TID is the data stream identifier that indicates different streams of data.
m**_axis_tdest (C_M**_AXIS_TDEST_WIDTH – 1:0)	O	TDEST provides routing information for the data stream.
m**_axis_tdata (C_M**_AXIS_TDATA_WIDTH – 1:0)	O	TDATA is the primary payload that is used to provide the data that is passing across the interface. The width of the data payload is an integer number of bytes.
m**_axis_tkeep (C_M**_AXIS_TDATA_WIDTH/ 8 – 1:0)	O	TKEEP is the byte qualifier that indicates whether the content of the associated byte of TDATA is processed as part of the data stream. Associated bytes that have the TKEEP byte qualifier deasserted are null bytes and can be removed from the data stream.
m**_axis_tlast	O	TLAST indicates the boundary of a packet.
m**_axis_tready	I	TREADY indicates that the slave can accept a transfer in the current cycle.
m**_axis_tvalid	O	TVALID indicates that the master is driving a valid transfer. A transfer takes place when both TVALID and TREADY are asserted.

S**_AXIS Interface Ports

The following table shows the S**_AXIS Interface Ports and their descriptions.

Table 4: **S**_AXIS Interface Ports**

Port Name	I/O	Description
s**_axis_tid (C_M**_AXIS_TID_WIDTH – 1:0)	I	TID is the data stream identifier that indicates different streams of data.
s**_axis_tdest (C_M**_AXIS_TDEST_WIDTH – 1:0)	I	TDEST provides routing information for the data stream.
s**_axis_tdata (C_M**_AXIS_TDATA_WIDTH – 1:0)	I	TDATA is the primary payload that is used to provide the data that is passing across the interface. The width of the data payload is an integer number of bytes
s**_axis_tkeep (C_M**_AXIS_TDATA_WIDTH/ 8 – 1:0)	I	TKEEP is the byte qualifier that indicates whether the content of the associated byte of TDATA is processed as part of the data stream. Associated bytes that have the TKEEP byte qualifier deasserted are null bytes and can be removed from the data stream.
s**_axis_tlast	I	TLAST indicates the boundary of a packet.
s**_axis_tready	O	TREADY indicates that the slave can accept a transfer in the current cycle.
s**_axis_tvalid	I	TVALID indicates that the master is driving a valid transfer. A transfer takes place when both TVALID and TREADY are asserted.

Other Ports

Table 5: **Other Ports**

Port Name	I/O	Description
aclk**	I	Clock ports associated with AI Engine to PL AXI4-Stream and PL to AI Engine AXI4-Stream interfaces.
noc_aclk**	O	Clock ports associated with AI Engine to NOC AXI4-Stream and NOC to AI Engine AXI4-Stream interfaces.
m**_axis_aclk	O	Clock ports associated with M**_AXI AXI4-MM interfaces.
s**_axis_aclk	O	Clock ports associated with S**_AXI AXI4-MM interfaces.
trig_in_**	I	External event/trigger input from PL to AI Engine
trig_out_**	O	External event/trigger output from AI Engine to PL
trig_in_**_clk	I	Trigger in clock
trig_out_**_clk	I	Trigger out clock

Designing with the Core

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

General Design Guidelines

Registering Signals

To simplify timing and increase system performance in a programmable device design, keep all inputs and outputs registered between the user application and the core. This means that all inputs and outputs from the user application should come from, or connect to, a flip-flop. While registering signals might not be possible for all paths, it simplifies timing analysis and makes it easier for the Xilinx® tools to place and route the design.

Make Only Allowed Modifications

You should not modify the core. Any modifications can have adverse effects on system timing and protocol compliance. Supported user configurations of the core can only be made by selecting the options in the customization IP dialog box when the core is generated.

Clocking

The following table shows the clock parameters and their descriptions.

Table 6: **Clocks**

Clock	Description
aclk*	These clock ports are associated with AI Engine IP to PL AXI4-Stream and PL to AI Engine AXI4-Stream interfaces. One or more interfaces can be associated with one of these clock ports.
s**_axi_aclk	These clock ports are associated with S**_AXI interfaces. These are meant to be connected to a NoC IP only. Each interface is associated with one dedicated clock port and should not be used for any other purpose.

Table 6: **Clocks** *(cont'd)*

Clock	Description
m**_axi_aclk	These clock ports are associated with S**_AXI interfaces. These are meant to be connected to a NoC IP only. Each interface is associated with one dedicated clock port and should not be used for any other purpose.
noc_aclk*	These clock ports are associated with AI Engine to NoC AXI4-Stream and NoC to AI Engine AXI4-Stream interfaces. These are meant to be connected to a NoC IP only. Each interface is associated with one dedicated clock port and should not be used for any other purpose.
trig_in_*_clk	Trigger in clock.
trig_out_*_clk	Trigger out clock.

Design Flow Steps

This section 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)
- *Vivado Design Suite User Guide: Designing with IP* (UG896)
- *Vivado Design Suite User Guide: Getting Started* (UG910)
- *Vivado Design Suite User Guide: Logic Simulation* (UG900)

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) 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 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) and the *Vivado Design Suite User Guide: Getting Started* (UG910).

Figures in this chapter are illustrations of the Vivado IDE. The layout depicted here might vary from the current version.

Configuration Guide

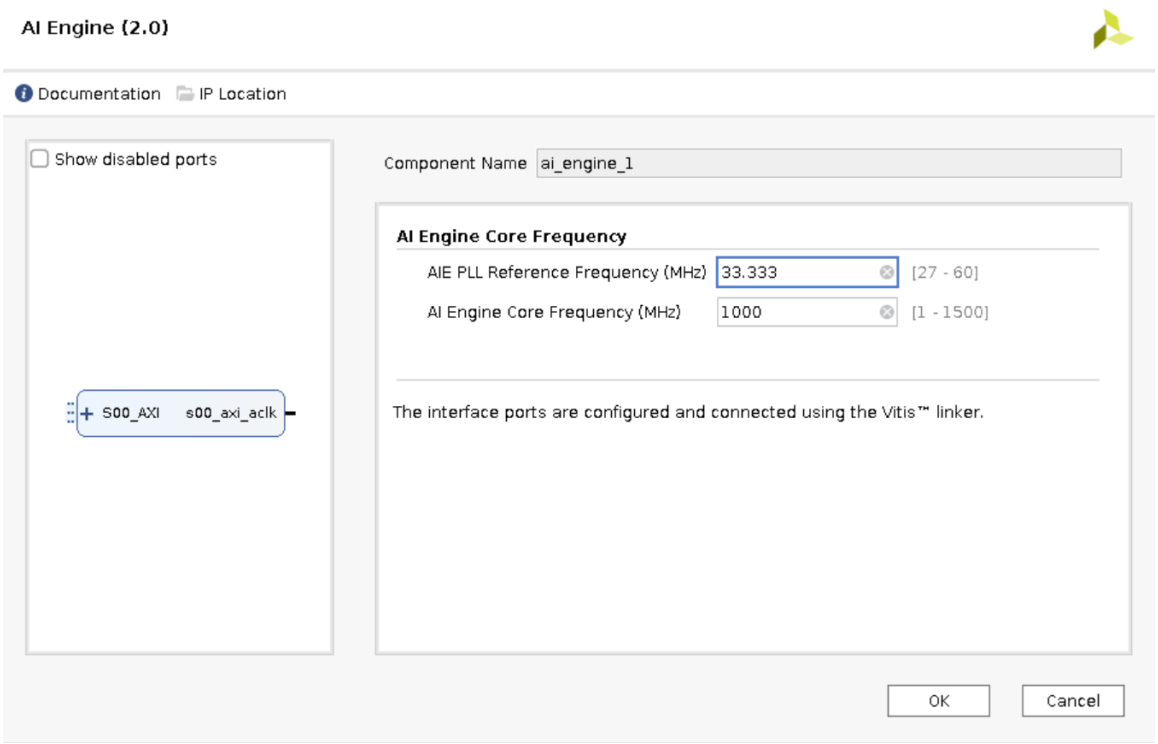
When building a platform that will use the AI Engine IP, by default, the AI Engine has only one enabled memory-mapped AXI port for configuration. The rest of the interface ports are enabled and connected using the Vitis™ software platform linker v++. Changing additional interface ports from the AI Engine IP is not supported as they could result in undesirable behavior such as compilation or runtime failures.

For more information on AI Engine programming, refer to the *Versal ACAP AI Engine Programming Environment User Guide* (UG1076) (Registration required).

Connect the input `S00_AXI` port of the AI Engine IP to the AXI NoC.

The following figure shows the available configuration settings.

Figure 3: **Settings Tab**



- **Component Name:** Shows the unique name for the AI Engine IP core and can only be changed through the Block properties in Vivado IP integrator.
- **AIE PLL Reference Frequency (MHz):** Selects the reference clock for the AI Engine IP PLL. The clock frequency depends on the **Control, Interfaces & Processing System (CIPS) IP → PS-PMC → Clock Configuration → Output Clocks → PMC Domain Clocks → Processor/Memory Clocks → HSM0 → Actual Frequency (MHz)**.

- **AI Engine Core Frequency (MHz):** Selects the AI ENGINE PLL output clock to the AI Engine IP Array.

User Parameters

The following table shows the relationship between the fields in the Vivado® IDE and the user parameters (which can be viewed in the Tcl Console).

Table 7: **User Parameters**

Vivado IDE Parameter/Value	User Parameter/Value	Default Value
AIE_CORE_REF_CTRL_FREQMHZ	1-1500 MHz	1000 MHz
AIE_REF_CLK_FREQMHZ	27-60 MHz	33.333 MHz

Output Generation

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

Constraining the Core

Required Constraints

This section is not applicable for this IP core.

Device, Package, and Speed Grade Selections

This section is not applicable for this IP core.

Clock Frequencies

This section is not applicable for this IP core.

Clock Management

This section is not applicable for this IP core.

Clock Placement

This section is not applicable for this IP core.

Banking

This section is not applicable for this IP core.

Transceiver Placement

This section is not applicable for this IP core.

I/O Standard and Placement

This section is not applicable for this IP core.

Synthesis and Implementation

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

XILINX

Appendix A

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 (DocNav) provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open 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 DocNav, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

Note: For more information on DocNav, see the Documentation Navigator page on the Xilinx website.

References

These documents provide supplemental material useful with this product guide:

1. *Versal ACAP AI Engine Programming Environment User Guide* (UG1076) (Registration required)
2. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
3. *Vivado Design Suite User Guide: Designing with IP* (UG896)
4. *Vivado Design Suite User Guide: Getting Started* (UG910)
5. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
6. *Versal ACAP AI Engine Architecture Manual* (AM009)

Revision History

The following table shows the revision history for this document.

Section	Revision Summary
12/08/2020 Version 2.0	
Initial release	N/A

Please Read: Important Legal Notices

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