DPUCVDX8H for Convolutional Neural Networks v1.0

LogiCORE IP Product Guide

PG403 (v1.0) February 28, 2022

Xilinx is creating an environment where employees, customers, and partners feel welcome and included. To that end, we're removing non-inclusive language from our products and related collateral. We've launched an internal initiative to remove language that could exclude people or reinforce historical biases, including terms embedded in our software and IPs. You may still find examples of non-inclusive language in our older products as we work to make these changes and align with evolving industry standards. Follow this link for more information.

Table of Contents

Introduction

The Xilinx® Deep Learning Processor Unit (DPU) is a series of soft IPs dedicated to convolutional neural networks acceleration. The DPUCVDX8H is a high-performance CNN inference accelerator specially designed for Xilinx® Versal® platform, which has a large AI Engine array. The DPUCVDX8H can be quickly deployed on the VCK5000 Alveo™ data center accelerator card to instantly meet the acceleration tasks of a variety of neural networks. It runs with a set of efficiently optimized instructions and can support most convolutional neural networks, such as VGG, ResNet, GoogLeNet, YOLO, SSD, and MobileNet.

Features

The DPUCVDX8H has the following features:

- One 32-bit AXI slave interface can be connected to NoC or shell design for accessing configuration and status registers.
- One 256-bit AXI master interface connected to NoC for code fetch.
- Four 256-bit AXI master interfaces connected to NoC for model parameters loading.
- 1~8 512-bit AXI master interface connected to NoC for DPUCVDX8H operates its input/output/intermediate feature-map in DRAM.

The following are a few highlights of the DPUCVDX8H functionality:

- Configurable hardware configuration includes: Engine number
- Convolution and deconvolution
- Max pooling
- Average pooling
- ReLU, ReLU6, and Leaky ReLU
- Concat
- Elementwise-sum
- Dilation
- Reorg

- Fully connected layer
- Batch Normalization
- Split

IP Facts

LogiCORE™ IP Facts Table	
Core Specifics	
Supported Device Family	VCK5000
Supported User Interfaces	AXI4-Lite CSR Interface
Resources	See Performance and Resource Use
Provided with Core	
Design Files	XO files/connection files/AIE library file
Example Design	Verilog
Constraints File	Xilinx Design Constraints (XDC)
Supported S/W Driver	XRT
Tested Design Flows[1]	
Design Entry	Vivado® Design Suite
Simulation	N/A
Synthesis	Vivado Synthesis

Notes:

1. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

XILINX

Chapter 2

Overview

Navigating Content by Design Process

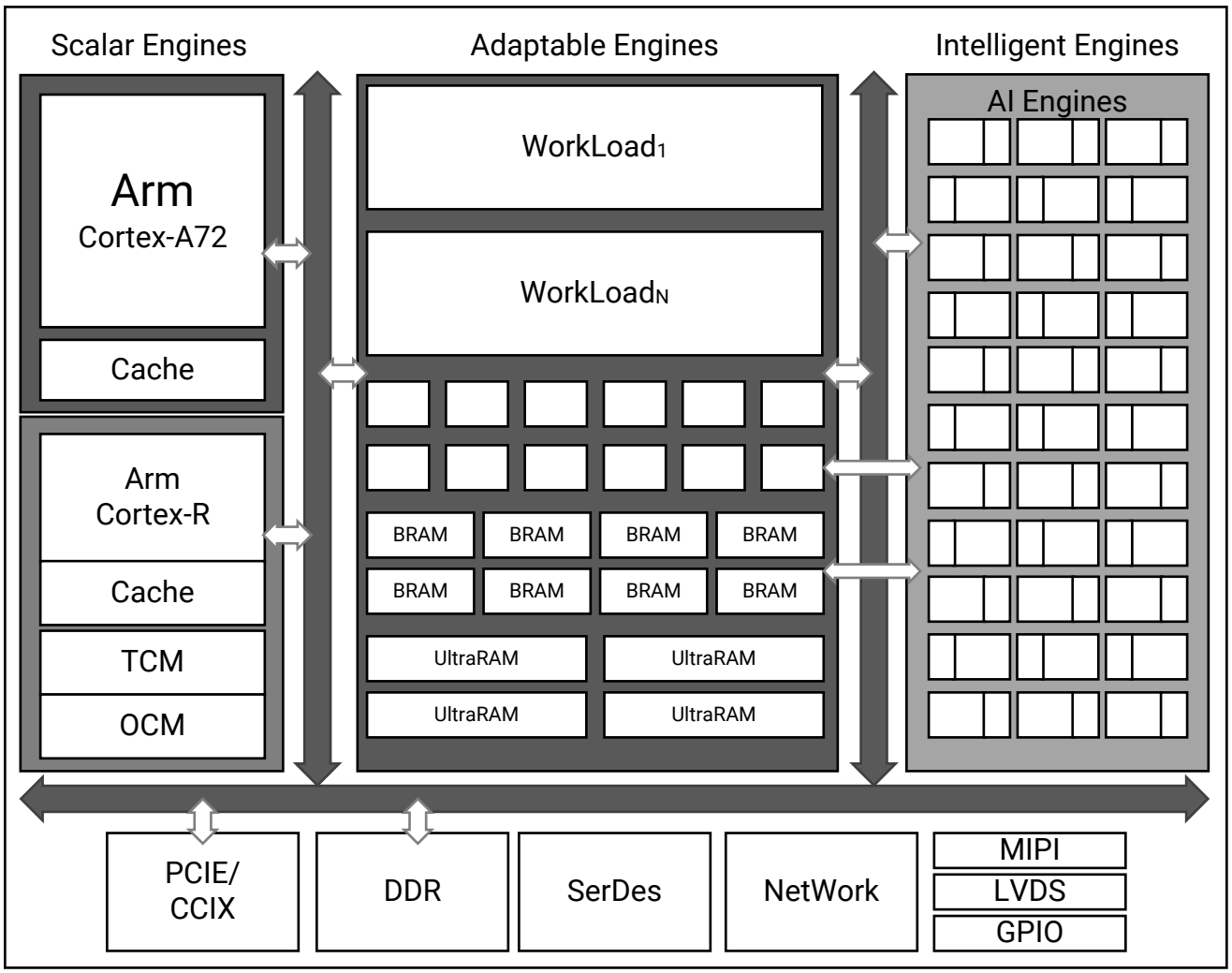
Xilinx® documentation is organized around a set of standard design processes to help you find relevant content for your current development task. All Versal® ACAP design process Design Hubs and the Design Flow Assistant materials can be found on the Xilinx.com website. This document covers the following design processes:

- **AI Engine Development:** Creating the AI Engine graph and kernels, library use, simulation debugging and profiling, and algorithm development. Also includes the integration of the PL and AI Engine kernels. Topics in this document that apply to this design process include:
  - Chapter 1: Introduction
  - Chapter 5: Development Flow
- **Hardware, IP, and Platform Development:** Creating the PL IP blocks for the hardware platform, creating PL kernels, 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:
  - Hardware Architecture
  - Port Descriptions
  - Register Space
- **System Integration and Validation:** Integrating and validating the system functional performance, including timing, resource use, and power closure. Topics in this document that apply to this design process include:
  - Chapter 5: Development Flow

Core Overview

The Xilinx® DPUCVDX8H DPU is a programmable engine optimized for convolutional neural networks, mainly for high performance applications, and it could be used to perform deep learning inference tasks such as image classification, object detection, and semantic segmentation. It is specifically designed for the Versal® platform. The architecture of the Versal platform is shown in the following figure. The Versal ACAP integrates more powerful Arm® cores, and more importantly, it integrated a new powerful programmable computing array, which is called AI Engine. It can be used as a large DSP array to accelerate general high-density computing tasks, such as 5G. The AI Engine is also optimized for machine learning tasks, compared with programmable logic, most computing tasks can get much better performance on AI Engine. The DPUCVDX8H puts most of the calculation tasks on the AI Engine, only keeping the control logic and a small amount of computation in the PL.

Figure 1: **Versal SoC FPGA**

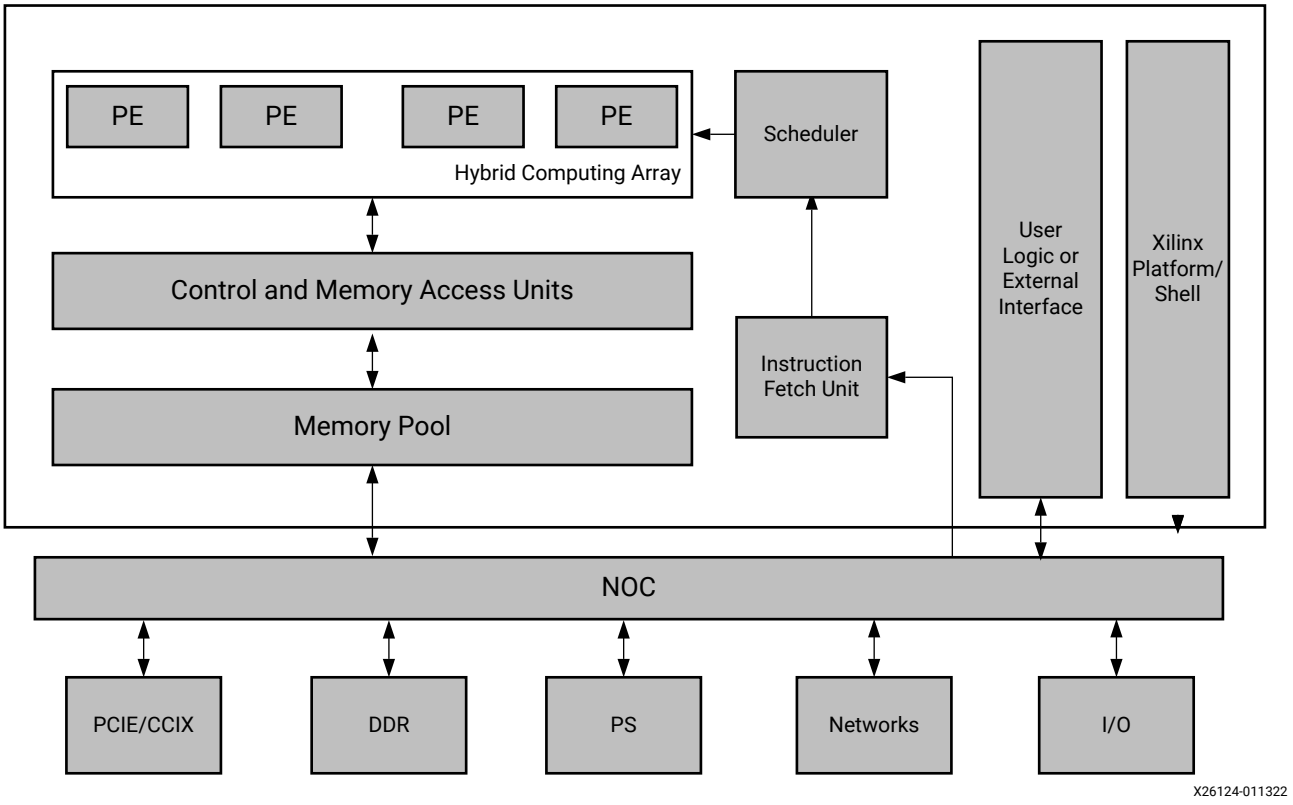


This unit includes a high-performance scheduler module, a hybrid computing array module, an instruction fetch unit module, control and memory access module, and a memory pool module. The DPU uses a specialized instruction set, which supports the efficient mapping for many convolutional neural networks. Some examples of convolutional neural networks that have been deployed include VGG, ResNet, GoogLeNet, YOLO, SSD, FPN.

The convolutional computing unit of the DPU IP is implemented on AI Engine, the control and memory access unit and memory pool are implemented in the programmable logic. The DPU IP can be connected to NoC with a standard AXI interface to access DRAM and receive external control commands. You can use Xilinx shell or self-defined logic to operate DPU, including configuration/network instruction injection/handling interrupts and data movement. To simplify development and make it easy to use, Xilinx provides a platform, shell, and related tools, to support you to integrate it into the design.

The following image shows the top-level block diagram of DPU.

Figure 2: **DPU Top-Level Block Diagram**



Development Tools

The Vitis™ unified software platform is required to integrate the DPU into your projects. Vitis software platform 2021.2 or later is required. Contact your local sales representative if the project requires an older version of the Vivado® Design Suite.

Device Resources

The DPUCVDX8H can only be deployed on the Versal platform, such as the VCK5000. For more information on resource utilization, refer to Resource Use.

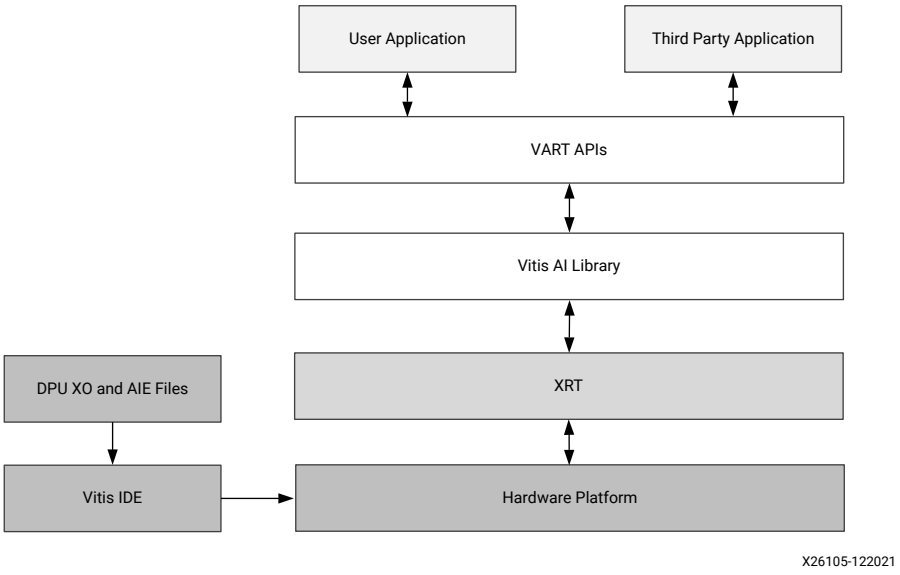
DPUCVDX8H Development Flow

The DPUCVDX8H is available with the software development stack of Vitis AI development kit. Free developer resources can be obtained from the Xilinx website.

The *Vitis AI User Guide* (UG1414) describes how to use the DPU for deploying machine learning applications with the Vitis AI tools. The development flow for DPU applications is summarized in the following steps and shown in the following figure.

1. Use the Vitis tool to generate the bitstream.
2. Download the bitstream to the target board. For instructions on how to set up a running environment for DPU applications, refer to the *Vitis AI User Guide* (UG1414).

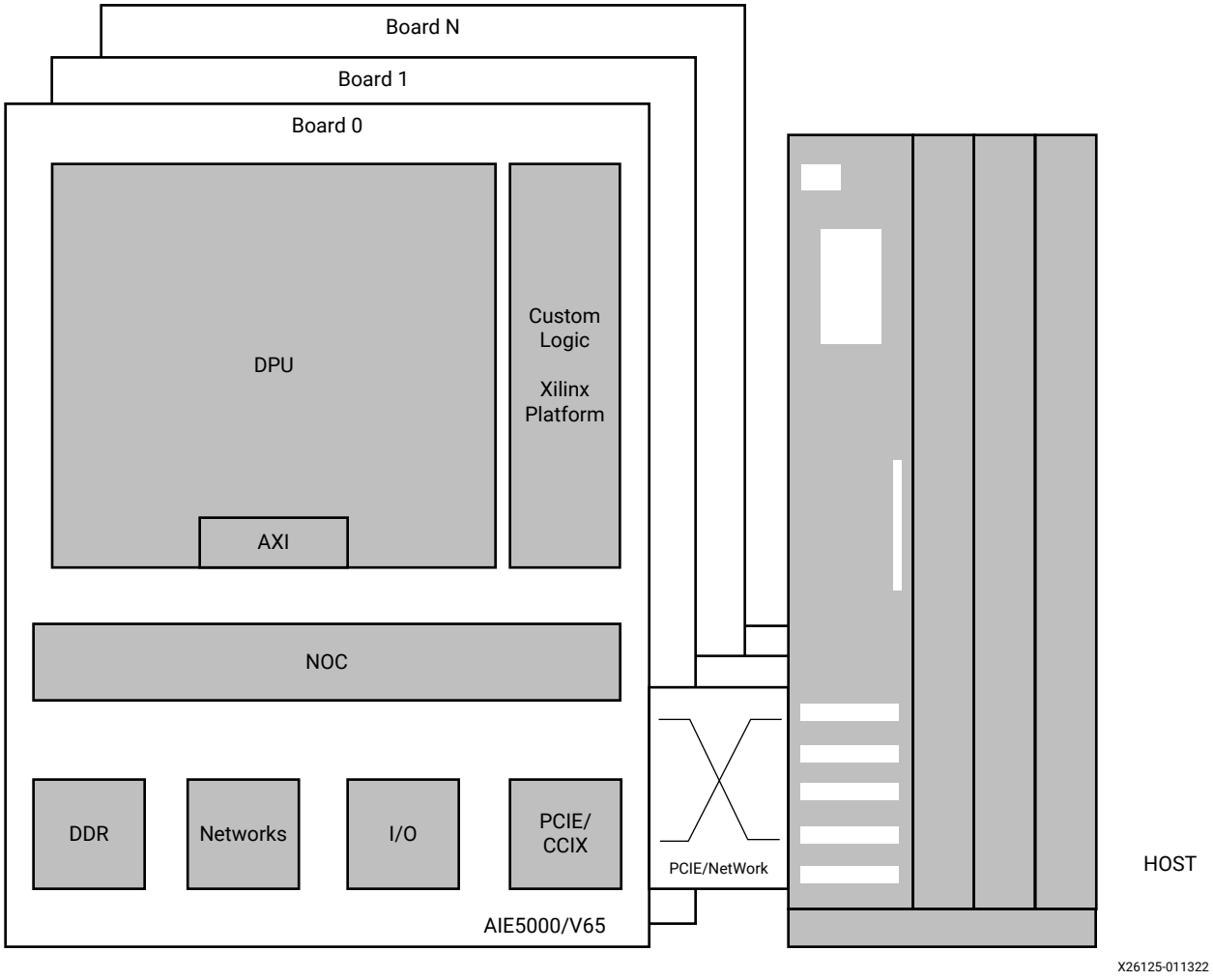
Figure 3: **Basic Development Flow**



Example System with DPUCVDX8H

The following figure shows an example system block diagram with the Xilinx VCK5000 cards which has a Versal VC1902 FPGA on card and a PCIe® or the Network port. The board can be installed into the PCIe slot or be connected to network port of the host server. The DPU IP can be integrated into the system with network on chip, and the whole system is integrated into the server through PCIe or network interconnect.

Figure 4: **Example System with Integrated DPU**

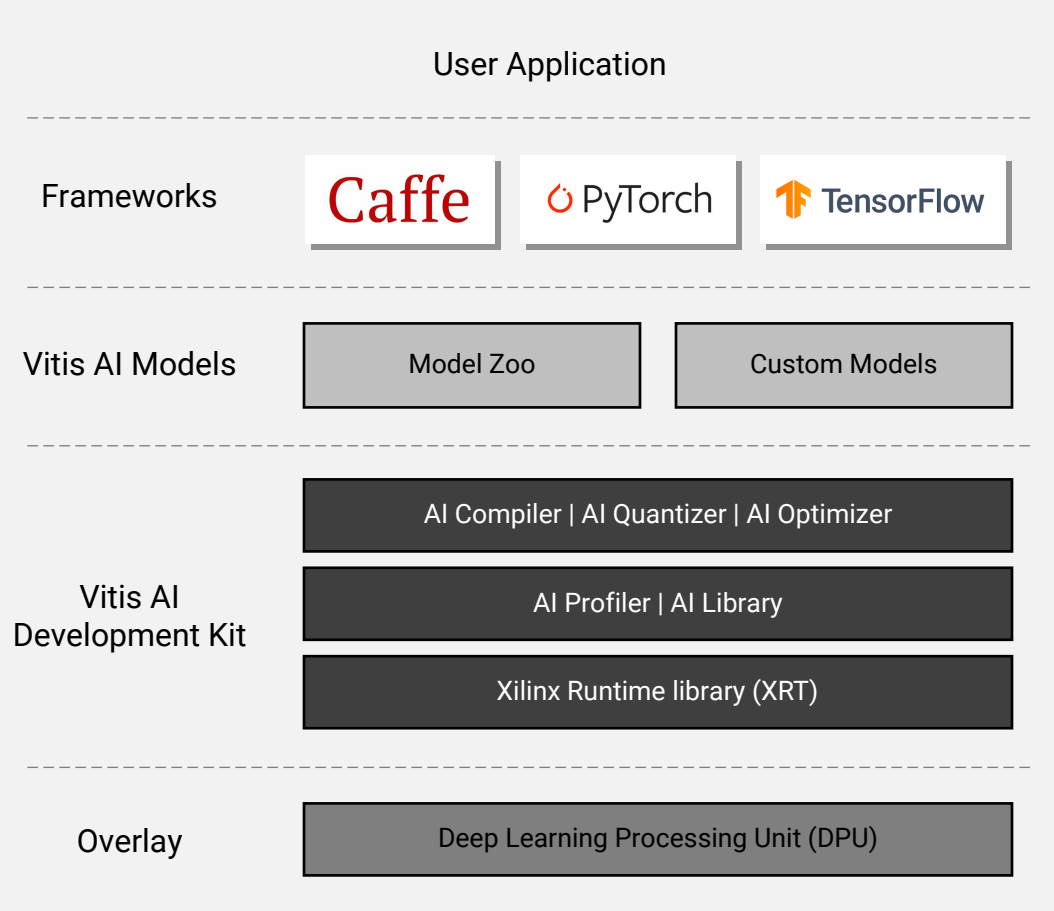


Vitis AI Development Kit

The Vitis AI development environment is used for AI inference on Xilinx hardware platforms. It consists of optimized IP cores, tools, libraries, models, and example designs.

As shown in the following figure, the Vitis AI development kit consists of AI Compiler, AI Quantizer, AI Optimizer, AI Profiler, AI Library, and Xilinx Runtime Library (XRT).

Figure 5: **Vitis AI Stack**



For more information on the Vitis AI development kit, see the *Vitis AI User Guide* (UG1414).

You can download the Vitis AI development kit for free from https://github.com/Xilinx/Vitis-AI.

Product Specification

Performance

The following table shows the peak performance of the DPU.

Table 1: **DPU_EU Computing Power on Different Devices**

Device	DPU Configuration	Frequency	Peak Performance
VCK5000	8PE	300 MHz	76.8
VCK5000	6PE	300 MHz	57.6

Resource Use

The resource utilization of various DPUCVDX8H batches configuration on VCK5000 is shown in the following table.

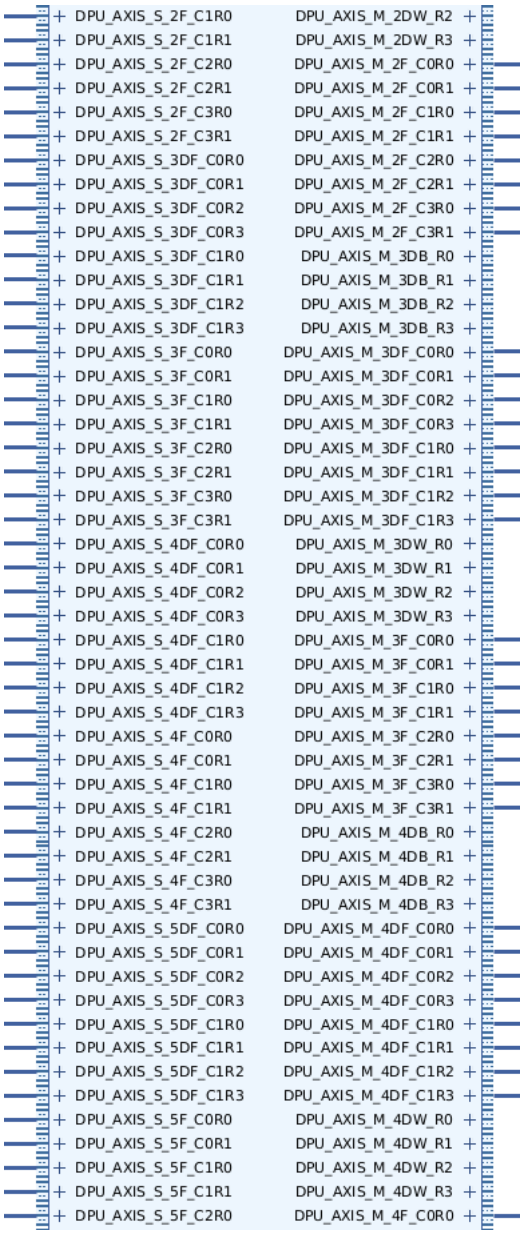
Table 2: **DPUCVDX8H Utilization**

Arch	LUT	Register	Block RAM	URAM	DSP
2 PE	235K	229K	228	222	144
4 PE	449K	433K	456	312	284
6 PE	629K	643K	780	402	424
8 PE	663K	692K	912	424	524

Port Descriptions

Some of the DPU top-level interfaces are shown in the following figure. For a detailed description of I/O ports, see Appendix A: I/O Signals .

Figure 6: **DPU IP Ports**



Register Space

The DPUCVDX8H IP implements control specific register (CSR) in programmable logic. The following tables show the DPU IP registers. These registers are accessible through the DPU_CSR_S_AXI interface.

DPU Control Registers

The DPU control registers are used to start a DPU, wait for the task to finish, and then clear the DPU status. The details of control registers are shown in the following table:

Table 3: **DPU Control Registers**

Register	Address Offset	Width	Type	Description
reg_ap_control	0x000	32	r/w	bit 0: ap_start (read/write/clear on handshake) bit 1: ap_done (read) bit 2: ap_idle (read) bit 3: ap_ready (read) bit 4: ap_continue (read/write/self clear) others: reserved
Global interrupt enable register (GIER)	0x004	32	r/w	bit 0: global interrupt enable others: reserved
IP interrupt enable register (IPIER)	0x008	32	r/w	bit 0: channel 0 (ap_done) bit 1: channel 1 (ap_ready) others: reserved
IP interrupt status register (IPISR)	0x00c	32	r/w	bit 0: channel 0 (ap_done) (read/toggle on write) bit 1: channel 1 (ap_ready) (read/toggle on write) others: reserved
reg_dpu_start	0x010	32	r/w	bit [0]: enable DPU to start
reg_finish_clr	0x018	32	r/w	bit [0]: clear reg_finish_sts
reg_finish_sts	0x080	32	r	bit [0]: indicate DPU has finished. The DPU finish signal is also output as DPU interrupt to trigger xdma or custom logic. The DPU finish is a level and asynchronous signal.

DPU Configuration Registers

The DPU configuration registers are used to indicate instruction address, common address and mean value settings.

The `reg_instr_addr` register is used to indicate the instruction address of all DPU processing engines.

The reg_base_addr register is used to indicate the address of input image and parameters for the DPU in external memory. The width of a DPU base address is 44 bits. All registers are 32 bits wide, so two registers are required to represent a 44-bit wide base address. reg_dpu0_base_addr0_l represents the lower 32 bits of base_address0 in DPU batch 0 and reg_dpu0_base_addr0_h represents the upper 12 bits of base_address0 in DPU batch 0.

There are eight groups of DPU base addresses for each DPU batch engine and thus 64 groups of DPU base addresses for up to eight DPU batch engines.

The details of configuration registers are shown in the following table:

Table 4: **DPU Configuration Registers**

Register	Address Offset	Width	Type	Description
reg_instr_addr_l	0x140	32	r/w	The lower 32 bits of instruction address of DPU. 4 KB aligned.
reg_instr_addr_h	0x144	32	r/w	The lower 12 bit in the register represent the upper 1 bit of instruction address of DPU. 4 KB aligned.
reg_engine0_base_addr_0_l	0x100	32	r/w	The lower 32 bits of base address0 of DPU engine0.
reg_engine0_base_addr_0_h	0x104	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address0 of DPU engine0.
reg_engine0_base_addr_1_l	0x108	32	r/w	The lower 32 bits of base address1 of DPU engine0.
reg_engine0_base_addr_1_h	0x10c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address1 of DPU engine0.
reg_engine0_base_addr_2_l	0x110	32	r/w	The lower 32 bits of base address2 of DPU engine0.
reg_engine0_base_addr_2_h	0x114	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address2 of DPU engine0.
reg_engine0_base_addr_3_l	0x118	32	r/w	The lower 32 bits of base address3 of DPU engine0.
reg_engine0_base_addr_3_h	0x11c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address3 of DPU engine0.
reg_engine0_base_addr_4_l	0x120	32	r/w	The lower 32 bits of base address4 of DPU engine0.
reg_engine0_base_addr_4_h	0x124	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address4 of DPU engine0.
reg_engine0_base_addr_5_l	0x128	32	r/w	The lower 32 bits of base address5 of DPU engine0.
reg_engine0_base_addr_5_h	0x12c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address5 of DPU engine0.
reg_engine0_base_addr_6_l	0x130	32	r/w	The lower 32 bits of base address6 of DPU engine0.
reg_engine0_base_addr_6_h	0x134	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address6 of DPU engine0.
reg_engine0_base_addr_7_l	0x138	32	r/w	The lower 32 bits of base address7 of DPU engine0.
reg_engine0_base_addr_7_h	0x13c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address7 of DPU engine0.
reg_engine1_base_addr_0_l	0x200	32	r/w	The lower 32 bits of base address0 of DPU engine1.

Table 4: **DPU Configuration Registers** *(cont'd)*

Register	Address Offset	Width	Type	Description
reg_engine1_base_addr_0_h	0x204	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address0 of DPU engine1.
reg_engine1_base_addr_1_l	0x208	32	r/w	The lower 32 bits of base address1 of DPU engine1.
reg_engine1_base_addr_1_h	0x20c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address1 of DPU engine1.
reg_engine1_base_addr_2_l	0x210	32	r/w	The lower 32 bits of base address2 of DPU engine1.
reg_engine1_base_addr_2_h	0x214	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address2 of DPU engine1.
reg_engine1_base_addr_3_l	0x218	32	r/w	The lower 32 bits of base address3 of DPU engine1.
reg_engine1_base_addr_3_h	0x21c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address3 of DPU engine1.
reg_engine1_base_addr_4_l	0x220	32	r/w	The lower 32 bits of base address4 of DPU engine1.
reg_engine1_base_addr_4_h	0x224	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address4 of DPU engine1.
reg_engine1_base_addr_5_l	0x228	32	r/w	The lower 32 bits of base address5 of DPU engine1.
reg_engine1_base_addr_5_h	0x22c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address5 of DPU engine1.
reg_engine1_base_addr_6_l	0x230	32	r/w	The lower 32 bits of base address6 of DPU engine1.
reg_engine1_base_addr_6_h	0x234	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address6 of DPU engine1.
reg_engine1_base_addr_7_l	0x238	32	r/w	The lower 32 bits of base address7 of DPU engine1.
reg_engine1_base_addr_7_h	0x23c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address7 of DPU engine1.
reg_engine2_base_addr_0_l	0x300	32	r/w	The lower 32 bits of base address0 of DPU engine2.
reg_engine2_base_addr_0_h	0x304	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address0 of DPU engine2.
reg_engine2_base_addr_1_l	0x308	32	r/w	The lower 32 bits of base address1 of DPU engine2.
reg_engine2_base_addr_1_h	0x30c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address1 of DPU engine2.
reg_engine2_base_addr_2_l	0x310	32	r/w	The lower 32 bits of base address2 of DPU engine2.

Table 4: **DPU Configuration Registers** *(cont'd)*

Register	Address Offset	Width	Type	Description
reg_engine2_base_addr_2_h	0x314	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address2 of DPU engine2.
reg_engine2_base_addr_3_l	0x318	32	r/w	The lower 32 bits of base address3 of DPU engine2.
reg_engine2_base_addr_3_h	0x31c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address3 of DPU engine2.
reg_engine2_base_addr_4_l	0x320	32	r/w	The lower 32 bits of base address4 of DPU engine2.
reg_engine2_base_addr_4_h	0x324	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address4 of DPU engine2.
reg_engine2_base_addr_5_l	0x328	32	r/w	The lower 32 bits of base address5 of DPU engine2.
reg_engine2_base_addr_5_h	0x32c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address5 of DPU engine2.
reg_engine2_base_addr_6_l	0x330	32	r/w	The lower 32 bits of base address6 of DPU engine2.
reg_engine2_base_addr_6_h	0x334	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address6 of DPU engine2.
reg_engine2_base_addr_7_l	0x338	32	r/w	The lower 32 bits of base address7 of DPU engine2.
reg_engine2_base_addr_7_h	0x33c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address7 of DPU engine2.
reg_engine3_base_addr_0_l	0x400	32	r/w	The lower 32 bits of base address0 of DPU engine3.
reg_engine3_base_addr_0_h	0x404	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address0 of DPU engine3.
reg_engine3_base_addr_1_l	0x408	32	r/w	The lower 32 bits of base address1 of DPU engine3.
reg_engine3_base_addr_1_h	0x40c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address1 of DPU engine3.
reg_engine3_base_addr_2_l	0x410	32	r/w	The lower 32 bits of base address2 of DPU engine3.
reg_engine3_base_addr_2_h	0x414	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address2 of DPU engine3.
reg_engine3_base_addr_3_l	0x418	32	r/w	The lower 32 bits of base address3 of DPU engine3.
reg_engine3_base_addr_3_h	0x41c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address3 of DPU engine3.
reg_engine3_base_addr_4_l	0x420	32	r/w	The lower 32 bits of base address4 of DPU engine3.

Table 4: **DPU Configuration Registers** *(cont'd)*

Register	Address Offset	Width	Type	Description
reg_engine3_base_addr_4_h	0x424	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address4 of DPU engine3.
reg_engine3_base_addr_5_l	0x428	32	r/w	The lower 32 bits of base address5 of DPU engine3.
reg_engine3_base_addr_5_h	0x42c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address5 of DPU engine3.
reg_engine3_base_addr_6_l	0x430	32	r/w	The lower 32 bits of base address6 of DPU engine3.
reg_engine3_base_addr_6_h	0x434	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address6 of DPU engine3.
reg_engine3_base_addr_7_l	0x438	32	r/w	The lower 32 bits of base address7 of DPU engine3.
reg_engine3_base_addr_7_h	0x43c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address7 of DPU engine3.
reg_engine4_base_addr_0_l	0x500	32	r/w	The lower 32 bits of base address0 of DPU engine4.
reg_engine4_base_addr_0_h	0x504	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address0 of DPU engine4.
reg_engine4_base_addr_1_l	0x508	32	r/w	The lower 32 bits of base address1 of DPU engine4.
reg_engine4_base_addr_1_h	0x50c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address1 of DPU engine4.
reg_engine4_base_addr_2_l	0x510	32	r/w	The lower 32 bits of base address2 of DPU engine4.
reg_engine4_base_addr_2_h	0x514	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address2 of DPU engine4.
reg_engine4_base_addr_3_l	0x518	32	r/w	The lower 32 bits of base address3 of DPU engine4.
reg_engine4_base_addr_3_h	0x51c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address3 of DPU engine4.
reg_engine4_base_addr_4_l	0x520	32	r/w	The lower 32 bits of base address4 of DPU engine4.
reg_engine4_base_addr_4_h	0x524	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address4 of DPU engine4.
reg_engine4_base_addr_5_l	0x528	32	r/w	The lower 32 bits of base address5 of DPU engine4.
reg_engine4_base_addr_5_h	0x52c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address5 of DPU engine4.
reg_engine4_base_addr_6_l	0x530	32	r/w	The lower 32 bits of base address6 of DPU engine4.

Table 4: **DPU Configuration Registers** *(cont'd)*

Register	Address Offset	Width	Type	Description
reg_engine4_base_addr_6_h	0x534	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address6 of DPU engine4.
reg_engine4_base_addr_7_l	0x538	32	r/w	The lower 32 bits of base address7 of DPU engine4.
reg_engine4_base_addr_7_h	0x53c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address7 of DPU engine4.
reg_engine5_base_addr_0_l	0x600	32	r/w	The lower 32 bits of base address0 of DPU engine5.
reg_engine5_base_addr_0_h	0x604	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address0 of DPU engine5.
reg_engine5_base_addr_1_l	0x608	32	r/w	The lower 32 bits of base address1 of DPU engine5.
reg_engine5_base_addr_1_h	0x60c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address1 of DPU engine5.
reg_engine5_base_addr_2_l	0x610	32	r/w	The lower 32 bits of base address2 of DPU engine5.
reg_engine5_base_addr_2_h	0x614	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address2 of DPU engine5.
reg_engine5_base_addr_3_l	0x618	32	r/w	The lower 32 bits of base address3 of DPU engine5.
reg_engine5_base_addr_3_h	0x61c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address3 of DPU engine5.
reg_engine5_base_addr_4_l	0x620	32	r/w	The lower 32 bits of base address4 of DPU engine5.
reg_engine5_base_addr_4_h	0x624	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address4 of DPU engine5.
reg_engine5_base_addr_5_l	0x628	32	r/w	The lower 32 bits of base address5 of DPU engine5.
reg_engine5_base_addr_5_h	0x62c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address5 of DPU engine5.
reg_engine5_base_addr_6_l	0x630	32	r/w	The lower 32 bits of base address6 of DPU engine5.
reg_engine5_base_addr_6_h	0x634	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address6 of DPU engine5.
reg_engine5_base_addr_7_l	0x638	32	r/w	The lower 32 bits of base address7 of DPU engine5.
reg_engine5_base_addr_7_h	0x63c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address7 of DPU engine5.
reg_engine6_base_addr_0_l	0x700	32	r/w	The lower 32 bits of base address0 of DPU engine6.

Table 4: **DPU Configuration Registers** *(cont'd)*

Register	Address Offset	Width	Type	Description
reg_engine6_base_addr_0_h	0x704	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address0 of DPU engine6.
reg_engine6_base_addr_1_l	0x708	32	r/w	The lower 32 bits of base address1 of DPU engine6.
reg_engine6_base_addr_1_h	0x70c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address1 of DPU engine6.
reg_engine6_base_addr_2_l	0x710	32	r/w	The lower 32 bits of base address2 of DPU engine6.
reg_engine6_base_addr_2_h	0x714	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address2 of DPU engine6.
reg_engine6_base_addr_3_l	0x718	32	r/w	The lower 32 bits of base address3 of DPU engine6.
reg_engine6_base_addr_3_h	0x71c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address3 of DPU engine6.
reg_engine6_base_addr_4_l	0x720	32	r/w	The lower 32 bits of base address4 of DPU engine6.
reg_engine6_base_addr_4_h	0x724	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address4 of DPU engine6.
reg_engine6_base_addr_5_l	0x728	32	r/w	The lower 32 bits of base address5 of DPU engine6.
reg_engine6_base_addr_5_h	0x72c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address5 of DPU engine6.
reg_engine6_base_addr_6_l	0x730	32	r/w	The lower 32 bits of base address6 of DPU engine6.
reg_engine6_base_addr_6_h	0x734	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address6 of DPU engine6.
reg_engine6_base_addr_7_l	0x738	32	r/w	The lower 32 bits of base address7 of DPU engine6.
reg_engine6_base_addr_7_h	0x73c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address7 of DPU engine6.
reg_engine7_base_addr_0_l	0x800	32	r/w	The lower 32 bits of base address0 of DPU engine7.
reg_engine7_base_addr_0_h	0x804	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address0 of DPU engine7.
reg_engine7_base_addr_1_l	0x808	32	r/w	The lower 32 bits of base address1 of DPU engine7.
reg_engine7_base_addr_1_h	0x80c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address1 of DPU engine7.
reg_engine7_base_addr_2_l	0x810	32	r/w	The lower 32 bits of base address2 of DPU engine7.

Table 4: **DPU Configuration Registers** *(cont'd)*

Register	Address Offset	Width	Type	Description
reg_engine7_base_addr_2_h	0x814	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address2 of DPU engine7.
reg_engine7_base_addr_3_l	0x818	32	r/w	The lower 32 bits of base address3 of DPU engine7.
reg_engine7_base_addr_3_h	0x81c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address3 of DPU engine7.
reg_engine7_base_addr_4_l	0x820	32	r/w	The lower 32 bits of base address4 of DPU engine7.
reg_engine7_base_addr_4_h	0x824	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address4 of DPU engine7.
reg_engine7_base_addr_5_l	0x828	32	r/w	The lower 32 bits of base address5 of DPU engine7.
reg_engine7_base_addr_5_h	0x82c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address5 of DPU engine7.
reg_engine7_base_addr_6_l	0x830	32	r/w	The lower 32 bits of base address6 of DPU engine7.
reg_engine7_base_addr_6_h	0x834	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address6 of DPU engine7.
reg_engine7_base_addr_7_l	0x838	32	r/w	The lower 32 bits of base address7 of DPU engine7.
reg_engine7_base_addr_7_h	0x83c	32	r/w	The lower 12 bit in the register represent the upper 1 bit of base address7 of DPU engine7.

DPU Debug Registers

The DPU debug registers are used to indicate the processing cycles. The details of debug registers are shown in the following table:

Table 5: **Reg_dpu_start**

Register	Address Offset	Width	Type	Description
reg_prof_value	0x0a8	32	r	Indicates cycle counter of DPU processing time. Saturation counting.

Interrupts

As all DPU engines work synchronously, all DPU engines generate an interrupt to signal the completion of a task. A High state on `reg_dpu_start` or `ap_start` signals the start of a DPU task. At the end of the task, the DPU generates an interrupt and bit0 in IPISR, and `reg_finish_sts` is set to 1.

To support DPU interrupt, the Interrupt Controller module implements the following registers:

- **Global Interrupt Enable Register (GIER):** Provides the master enable/disable for the interrupt output to the processor or Interrupt Controller. See the Global Interrupt Enable Register (GIER) in Table 3: DPU Control Registers for more details.
- **IP Interrupt Enable Register (IPIER):** Implements the independent interrupt enable bit for each channel. See IP Interrupt Enable (IPIER) and IP Status Registers (IPISR) in Table 3: DPU Control Registers for more details.
- **IP Interrupt Status Register (IPISR):** Implements the independent interrupt status bit for each channel. The IPISR provides Read and Toggle-On-Write access. The Toggle-On-Write mechanism allows interrupt service routines to clear multiple ISR bits using a single write transaction. The IPISR can also be manually set to generate an interrupt for testing purposes. See IP Interrupt Enable (IPIER) and IP Status Registers (IPISR) in table 2 for more details.

The interrupt should be correctly connected to the IRQ port of the PCIe® controller.

Designing with the Core

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

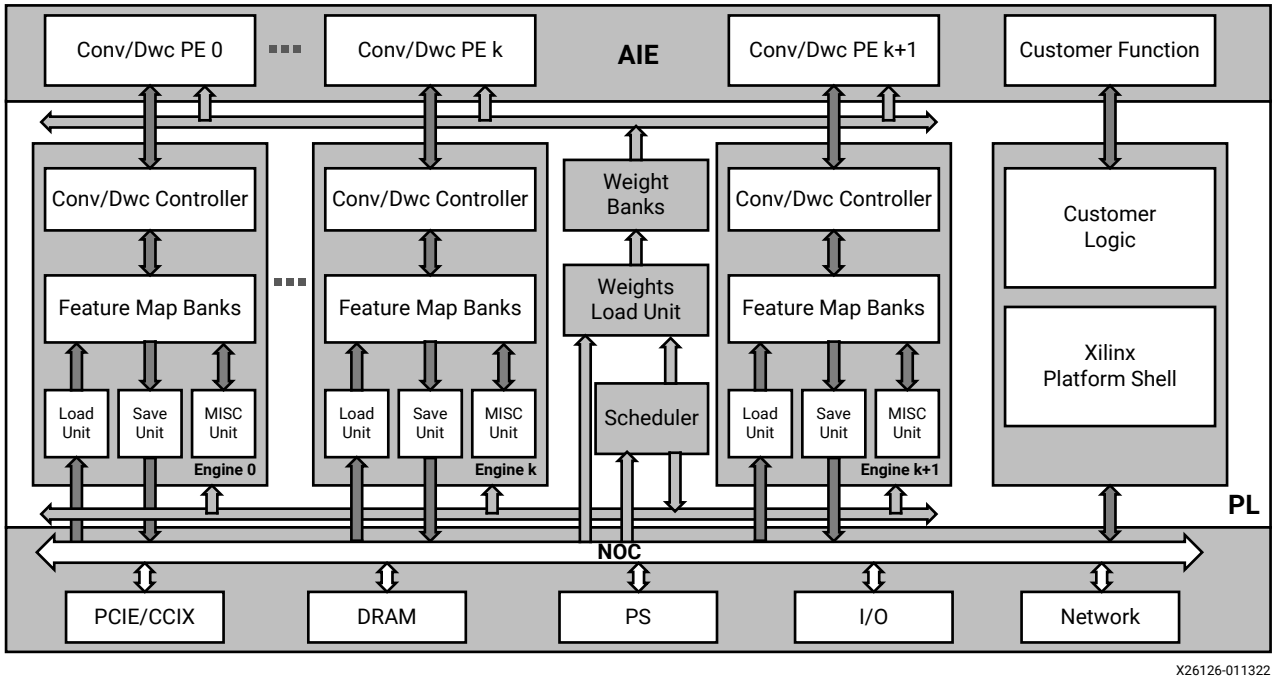
Hardware Architecture

The detailed hardware architecture of the DPUCVDX8H is shown in the following figure. Each implement could have one DPU instance, and each DPU may have two, four, six, or eight processing engines instances, the number of DPU instances depends on FPGA resource.

The Conv computing unit is implemented on AI Engine. The Conv control unit, Load unit, save unit, and MISC unit (pooling and element-wise processing) are implemented in programmable logic. All processing engines share the weight unit and scheduler unit, implemented with programmable logic. DRAM is used as system memory to store network instructions, input images, output results, and intermediate data. After bring-up, DPU fetches instructions from system memory to control the operations of the computing engine.

On-chip memory is used to buffer weights, bias, and intermediate data to achieve high throughput. Feature map banks are private to each batch engine. All processing engines share weights buffer in the same DPU instance. The data is reused as much as possible to reduce the memory bandwidth. The Conv processing engines (PE) take full advantage of the computing power of the AI Engine to get high performance.

Figure 7: **DPU Hardware Architecture**



DPUCVDX8H Feature Support

The DPU IP provides a few fixed configurations by different XO files. The configuration includes the number of processing engines, the different kernel/filter size in element-wise, and pooling.

The deep neural network features and the associated parameters supported by the DPU are shown in the following table:

Table 6: **Deep Neural Network Features and Parameters Supported by DPU**

Features	Description (channel_parallel=64, bank_depth=256)	
Convolution	Kernel Sizes	w: [1, 16] h: [1, 16]
	Strides	w: [1, 4] h: [1, 4]
	Pad_left/Pad_right	[0, (kernel_w - 1) * dilation_w]
	Pad_top/Pad_bottom	[0, (kernel_h - 1) * dilation_h]
	Input Size	kernel_w * kernel_h * ceil(input_channel / channel_parallel) <= bank_depth
	Output Size	output_channel <= 256 * channel_parallel
	Activation	ReLU, LeakyReLU, ReLU6, H-swish, H-sigmod
	Dilation	dilation * input_channel <= 256 * channel_parallell
depthwise-conv2d[1]	Kernel Sizes	W,H: {3, 5} W==H
	Strides	W: [1, 4] H: [1, 4]
	Pad_left/Pad_right	[0, (kernel_w - 1) * dilation_w + 1]
	Pad_top/Pad_bottom	[0, (kernel_h - 1) * dilation_h + 1]
	In Size	kernel_w * kernel_h * ceil(input_channel / channel_parallel) <= 4096
	Out Size	output_channel <= 256 * channel_parallel
	Activation	ReLU, ReLU6
	Dilation	dilation * input_channel <= 256 * channel_parallell
transposed-conv2d	Kernel Sizes	W: [1, 16] H: [1, 16]
	Strides	W: [1, 16] H: [1, 16]
	Pad_left/Pad_right	[1, kernel_w-1]
	Pad_top/Pad_bottom	[1, kernel_h-1]
	Out Size	output_channel <= 256 * channel_parallel
	Activation	ReLU, LeakyReLU, ReLU6, H-swish, H-sigmod
depthwise-transposed-conv2d[1]	Kernel Sizes	W,H: {6, 9, 10, 12, 15, 20}
	Strides	W: [2, 4] H: [2, 4]
	Pad_left/Pad_right	[1, kernel_w-1]
	Pad_top/Pad_bottom	[1, kernel_h-1]
	Out Size	output_channel <= 256 * channel_parallel
	Activation	ReLU, ReLU6

Table 6: **Deep Neural Network Features and Parameters Supported by DPU** *(cont'd)*

Features	Description (channel_parallel=64, bank_depth=256)	
max-pooling/ average-pooling	Kernel Sizes	2/4/6PE: W,H: [1, 8] W==H 8PE: W,H:[1,2,3,7] W==H
	Strides	W: [1, 8] H: [1, 8]
	Pad_left/Pad_right	[1, kernel_w-1]
	Pad_top/Pad_bottom	[1, kernel_h-1]
	Activation	Not supported
elementwise-sum	Input channel	input_channel <= 256 * channel_parallel
	Activation	ReLU

Notes:

1. Only 2/4/6 PE support depthwise.

Configuration Options

The DPU can be configured with predefined options, including the DPU processing engine number by different XO files. These options are used to meet performance and resources.

DPU Processing Engines

Different XO files can select two, four, six, or eight processing engines. Multiple engines can achieve higher performance. Consequently, it consumes more programmable logic resources. For eight engines in one DPU, depthwise-conv is not supported, pool kernel size = 4,5,6, and 8 is not supported.

Development Flow

Customizing and Generating the Bitstream Files with the Vitis Software Platform

The following sections describe the development flow on how to use the DPU IP with the Vitis™ IDE:

Input Files

To use the AI Engine design files and RTL files in program logic region, three types of files are provided:

- **libadf.a:** contains the compiled AI Engine design graph. For more information, refer to the *Versal ACAP AI Engine Programming Environment User Guide* (UG1076).
- **xo file:** Vitis kernel file which contains the RTL code files used in program logic.
- **Connection file:** contains the AXI4-Stream connection between the AI Engine and program logic design and also the connection between the program logic design and the NoC.

Vitis Flow

Send all the input files to the `v++` command to generate the `xclbin`. For more information, refer to the example design on GitHub.

I/O Signals

The DPU I/O signals are listed and described in the following table.

Table 7: **DPU Signal Description**

Signal Name	Interface Type	Width	I/O	Description
s_axi_control	Memory mapped AXI slave interface	32	I/O	32-bit memory mapped AXI interface for registers.
ap_clk	Clock	1	I	Input clock used for DPU general logic. The range is from 100 MHz to 300 MHz.
ap_clk_2	Clock	1	I	Unused
ap_rst_n	Reset	1	I	Active-Low reset for DPU general logic.
ap_rst_n_2	Reset	1	I	Unused
DPU_AXI_I	Memory mapped AXI master interface	256	I/O	256-bit memory mapped AXI interface for DPU instructions.
DPU_AXI_W0	Memory mapped AXI master interface	256	I/O	256-bit memory mapped AXI interface for DPU parameters.
DPU_AXI_W1	Memory mapped AXI master interface	256	I/O	256-bit memory mapped AXI interface for DPU parameters.
DPU_AXI_W2	Memory mapped AXI master interface	256	I/O	256-bit memory mapped AXI interface for DPU parameters.
DPU_AXI_W3	Memory mapped AXI master interface	256	I/O	256-bit memory mapped AXI interface for DPU parameters.
DPU_AXI_0	Memory mapped AXI master interface	512	I/O	512-bit memory mapped AXI interface for DPU engine0.
DPU_AXI_1	Memory mapped AXI master interface	512	I/O	512-bit memory mapped AXI interface for DPU engine1.
DPU_AXI_2	Memory mapped AXI master interface	512	I/O	512-bit memory mapped AXI interface for DPU engine2.
DPU_AXI_3	Memory mapped AXI master interface	512	I/O	512-bit memory mapped AXI interface for DPU engine3.
DPU_AXI_4	Memory mapped AXI master interface	512	I/O	512-bit memory mapped AXI interface for DPU engine4.
DPU_AXI_5	Memory mapped AXI master interface	512	I/O	512-bit memory mapped AXI interface for DPU engine5.
DPU_AXI_6	Memory mapped AXI master interface	512	I/O	512-bit memory mapped AXI interface for DPU engine6.
DPU_AXI_7	Memory mapped AXI master interface	512	I/O	512-bit memory mapped AXI interface for DPU engine7.
DPU_AXIS_M_0F_C0R0	Memory mapped AXI master interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.

Table 7: **DPU Signal Description** *(cont'd)*

Signal Name	Interface Type	Width	I/O	Description
DPU_AXIS_M_0F_C1R0	Memory mapped AXI master interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.
DPU_AXIS_M_0F_C2R0	Memory mapped AXI master interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.
DPU_AXIS_M_0F_C3R0	Memory mapped AXI master interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.
DPU_AXIS_M_0F_C0R1	Memory mapped AXI master interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.
DPU_AXIS_M_0F_C1R1	Memory mapped AXI master interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.
DPU_AXIS_M_0F_C2R1	Memory mapped AXI master interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.
DPU_AXIS_M_0F_C3R1	Memory mapped AXI master interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.
DPU_AXIS_M_0W_C0R0	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.
DPU_AXIS_M_0W_C1R0	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.
DPU_AXIS_M_0W_C2R0	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.
DPU_AXIS_M_0W_C3R0	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.
DPU_AXIS_M_0W_C0R1	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.
DPU_AXIS_M_0W_C1R1	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.
DPU_AXIS_M_0W_C2R1	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.
DPU_AXIS_M_0W_C3R1	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.
DPU_AXIS_M_0W_C0R2	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.
DPU_AXIS_M_0W_C1R2	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.
DPU_AXIS_M_0W_C2R2	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.
DPU_AXIS_M_0W_C3R2	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.
DPU_AXIS_M_0W_C0R3	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.
DPU_AXIS_M_0W_C1R3	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.
DPU_AXIS_M_0W_C2R3	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.
DPU_AXIS_M_0W_C3R3	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.
DPU_AXIS_M_0B_C0	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV bias.

Table 7: **DPU Signal Description** *(cont'd)*

Signal Name	Interface Type	Width	I/O	Description
DPU_AXIS_M_0B_C1	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV bias.
DPU_AXIS_M_0B_C2	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV bias.
DPU_AXIS_M_0B_C3	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV bias.
DPU_AXIS_S_0F_C0R0	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.
DPU_AXIS_S_0F_C1R0	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.
DPU_AXIS_S_0F_C2R0	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.
DPU_AXIS_S_0F_C3R0	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.
DPU_AXIS_S_0F_C0R1	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.
DPU_AXIS_S_0F_C1R1	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.
DPU_AXIS_S_0F_C2R1	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.
DPU_AXIS_S_0F_C3R1	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.
DPU_AXIS_M_0DF_C0R0	Memory mapped AXImaster interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.
DPU_AXIS_M_0DF_C0R1	Memory mapped AXI master interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.
DPU_AXIS_M_0DF_C0R2	Memory mapped AXI master interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.
DPU_AXIS_M_0DF_C0R3	Memory mapped AXI master interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.
DPU_AXIS_M_0DF_C1R0	Memory mapped AXI master interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.
DPU_AXIS_M_0DF_C1R1	Memory mapped AXI master interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.
DPU_AXIS_M_0DF_C1R2	Memory mapped AXI master interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.
DPU_AXIS_M_0DF_C1R3	Memory mapped AXI master interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.
DPU_AXIS_M_0DW_R0	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for DWC weights.
DPU_AXIS_M_0DW_R1	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for DWC weights.
DPU_AXIS_M_0DW_R2	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for DWC weights.
DPU_AXIS_M_0DW_R3	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for DWC weights.
DPU_AXIS_M_0DB_R0	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for DWC bias.

Table 7: **DPU Signal Description** *(cont'd)*

Signal Name	Interface Type	Width	I/O	Description
DPU_AXIS_M_0DB_R1	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for DWC bias.
DPU_AXIS_M_0DB_R2	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for DWC bias.
DPU_AXIS_M_0DB_R3	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for DWC bias.
DPU_AXIS_S_0DF_C0R0	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.
DPU_AXIS_S_0DF_C0R1	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.
DPU_AXIS_S_0DF_C0R2	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.
DPU_AXIS_S_0DF_C0R3	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.
DPU_AXIS_S_0DF_C1R0	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.
DPU_AXIS_S_0DF_C1R1	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.
DPU_AXIS_S_0DF_C1R2	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.
DPU_AXIS_S_0DF_C1R3	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.
DPU_AXIS_M_1F_C0R0	Memory mapped AXI master interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.
DPU_AXIS_M_1F_C1R0	Memory mapped AXI master interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.
DPU_AXIS_M_1F_C2R0	Memory mapped AXI master interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.
DPU_AXIS_M_1F_C3R0	Memory mapped AXI master interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.
DPU_AXIS_M_1F_C0R1	Memory mapped AXI master interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.
DPU_AXIS_M_1F_C1R1	Memory mapped AXI master interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.
DPU_AXIS_M_1F_C2R1	Memory mapped AXI master interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.
DPU_AXIS_M_1F_C3R1	Memory mapped AXI master interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.
DPU_AXIS_M_1W_C0R0	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.
DPU_AXIS_M_1W_C1R0	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.
DPU_AXIS_M_1W_C2R0	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.
DPU_AXIS_M_1W_C3R0	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.
DPU_AXIS_M_1W_C0R1	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.

Table 7: **DPU Signal Description** *(cont'd)*

Signal Name	Interface Type	Width	I/O	Description
DPU_AXIS_M_1W_C1R1	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.
DPU_AXIS_M_1W_C2R1	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.
DPU_AXIS_M_1W_C3R1	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.
DPU_AXIS_M_1W_C0R2	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.
DPU_AXIS_M_1W_C1R2	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.
DPU_AXIS_M_1W_C2R2	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.
DPU_AXIS_M_1W_C3R2	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.
DPU_AXIS_M_1W_C0R3	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.
DPU_AXIS_M_1W_C1R3	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.
DPU_AXIS_M_1W_C2R3	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.
DPU_AXIS_M_1W_C3R3	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV weights.
DPU_AXIS_M_1B_C0	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV bias.
DPU_AXIS_M_1B_C1	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV bias.
DPU_AXIS_M_1B_C2	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV bias.
DPU_AXIS_M_1B_C3	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for CONV bias.
DPU_AXIS_S_1F_C0R0	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.
DPU_AXIS_S_1F_C1R0	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.
DPU_AXIS_S_1F_C2R0	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.
DPU_AXIS_S_1F_C3R0	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.
DPU_AXIS_S_1F_C0R1	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.
DPU_AXIS_S_1F_C1R1	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.
DPU_AXIS_S_1F_C2R1	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.
DPU_AXIS_S_1F_C3R1	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for CONV feature map.
DPU_AXIS_M_1DF_C0R0	Memory mapped AXI master interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.

Table 7: **DPU Signal Description** *(cont'd)*

Signal Name	Interface Type	Width	I/O	Description
DPU_AXIS_M_1DF_C0R1	Memory mapped AXI master interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.
DPU_AXIS_M_1DF_C0R2	Memory mapped AXI master interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.
DPU_AXIS_M_1DF_C0R3	Memory mapped AXI master interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.
DPU_AXIS_M_1DF_C1R0	Memory mapped AXI master interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.
DPU_AXIS_M_1DF_C1R1	Memory mapped AXI master interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.
DPU_AXIS_M_1DF_C1R2	Memory mapped AXI master interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.
DPU_AXIS_M_1DF_C1R3	Memory mapped AXI master interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.
DPU_AXIS_M_1DW_R0	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for DWC weights.
DPU_AXIS_M_1DW_R1	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for DWC weights.
DPU_AXIS_M_1DW_R2	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for DWC weights.
DPU_AXIS_M_1DW_R3	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for DWC weights.
DPU_AXIS_M_1DB_R0	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for DWC bias.
DPU_AXIS_M_1DB_R1	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for DWC bias.
DPU_AXIS_M_1DB_R2	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for DWC bias.
DPU_AXIS_M_1DB_R3	Memory mapped AXI master interface	64	I/O	64-bit memory mapped AXI4-Stream interface for DWC bias.
DPU_AXIS_S_1DF_C0R0	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.
DPU_AXIS_S_1DF_C0R1	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.
DPU_AXIS_S_1DF_C0R2	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.
DPU_AXIS_S_1DF_C0R3	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.
DPU_AXIS_S_1DF_C1R0	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.
DPU_AXIS_S_1DF_C1R1	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.
DPU_AXIS_S_1DF_C1R2	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.
DPU_AXIS_S_1DF_C1R3	Memory mapped AXI slave interface	128	I/O	128-bit memory mapped AXI4-Stream interface for DWC feature map.
interrupt	Interrupt	1	O	Active-High interrupt output from DPU.

***Note*:** For a detailed connection guide with AI Engine, refer to Chapter 5: Development Flow.

XILINX

Appendix B

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 guide:

1. *Vitis AI User Guide* (UG1414)
2. *Versal ACAP AI Engine Programming Environment User Guide* (UG1076)

Revision History

The following table shows the revision history for this document.

Section	Revision Summary
02/28/2022 Version 1.0	
Initial release.	N/A

Please Read: Important Legal Notices

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