DPU for Convolutional Neural Network v1.0

DPU IP Product Guide

PG338 (v1.0) February 28, 2019

Revision History

The following table shows the revision history for this document.

Section	Revision Summary
02/28/2019 Version 1.0	
Initial release	N/A

Table of Contents

IP Facts

Introduction

The Xilinx® Deep Learning Processor Unit (DPU) is a configurable engine dedicated for convolutional neural network. The computing parallelism can be configured according to the selected device and application. It includes a set of efficiently optimized instructions. It can support most convolutional neural networks, such as VGG, ResNet, GoogLeNet, YOLO, SSD, MobileNet, FPN, etc.

Features

- *One slave AXI interface for accessing configuration and status registers.*
- *One master interface for accessing instructions.*
- *Supports configurable AXI master interface with 64 or 128 bits for accessing data.*
- *Supports individual configuration of each channel.*
- *Supports optional interrupt request generation.*
- *Some highlights of DPU functionality include:*
    - *Configurable hardware architecture includes: B512, B800, B1024, B1152, B1600, B2304, B3136, and B4096*
    - *Configurable core number up to three*
    - *Convolution and deconvolution*
    - *Max pooling*
    - *ReLu and Leaky ReLu*
    - *Concat*
    - *Elementwise*
    - *Dilation*
    - *Reorg*
    - *Fully connected layer*
    - *Batch Normalization*
    - *Split*

DPU IP Facts Table	
Core Specifics	
Supported Device Family	Zynq®-7000 SoC and UltraScale+™ MPSoC Family
Supported User Interfaces	Memory-mapped AXI interfaces
Resources	See Chapter 3: DPU Configuration
Provided with Core	
Design Files	Encrypted RTL
Example Design	Verilog
Constraint File	Xilinx Design Constraints (XDC)
Supported S/W Driver	Included in PetaLinux
Tested Design Flows	
Design Entry	Vivado® Design Suite
Simulation	N/A
Synthesis	Vivado Synthesis
Supported	
Provided by Xilinx at the Xilinx Support web page	

Notes:

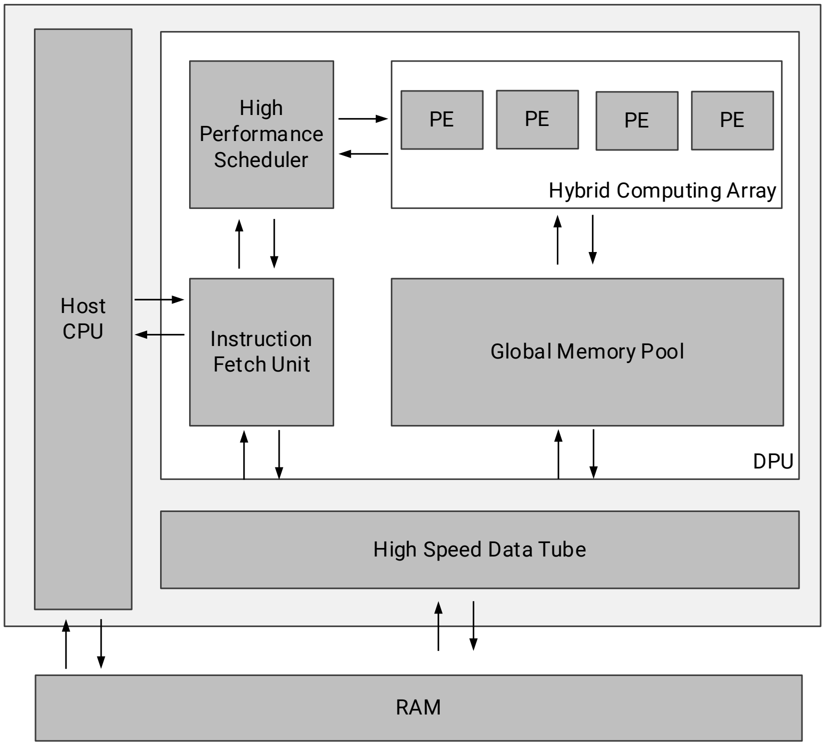
1. Linux OS and driver support information are available from DPU TRD or DNNDK.
2. If the requirement is on Zynq-7000 SoC, contact your local FAE.
3. For the supported versions of the tools, see the *Vivado Design Suite User Guide: Release Notes Installation, and Licensing* (UG973).

Introduction

The Xilinx® Deep Learning Processor Unit (DPU) is a programmable engine dedicated for convolutional neural network. The unit contains register configure module, data controller module, and convolution computing module. There is a specialized instruction set for DPU, which enables DPU to work efficiently for many convolutional neural networks. The deployed convolutional neural network in DPU includes VGG, ResNet, GoogLeNet, YOLO, SSD, MobileNet, FPN, etc.

The DPU IP can be integrated as a block in the programmable logic (PL) of the selected Zynq®-7000 SoC and Zynq UltraScale™+ MPSoC devices with direct connections to the processing system (PS). To use DPU, you should prepare the instructions and input image data in the specific memory address that DPU can access. The DPU operation also requires the application processing unit (APU) to service interrupts to coordinate data transfer.

The top-level block diagram of DPU is shown in Figure 1.



X22327-022019

Figure 1: Top-Level Block Diagram

Development Tools

Use the Xilinx Vivado Design Suite to integrate DPU into your own project. Vivado Design Suite 2018.2 or later version is recommended. Previous versions of Vivado can also be supported. For requests, contact your sales representative.

Device Resources

The DPU logic resource is optimized and scalable across Xilinx UltraScale+ MPSoC and Zynq-7000 devices. For the detailed resource utilization, refer to Chapter 3: DPU Configuration.

How to Run DPU

The DPU operation depends on the driver which is included in the Xilinx Deep Neural Network Development Kit (DNNDK) toolchain.

You can download the free developer resources from the Xilinx website: https://www.xilinx.com/products/design-tools/ai-inference/ai-developer-hub.html#edge

Refer to the *DNNDK User Guide* (UG1327) to obtain an essential guide on how to run a DPU with DNNDK tools. The basic development flow is shown in Figure 2. First, use Vivado to generate the bitstream. Then, download the bitstream to the target board and install the DPU driver. For instructions on how to install the DPU driver and dependent libraries, refer to the *DNNDK User Guide* (UG1327).

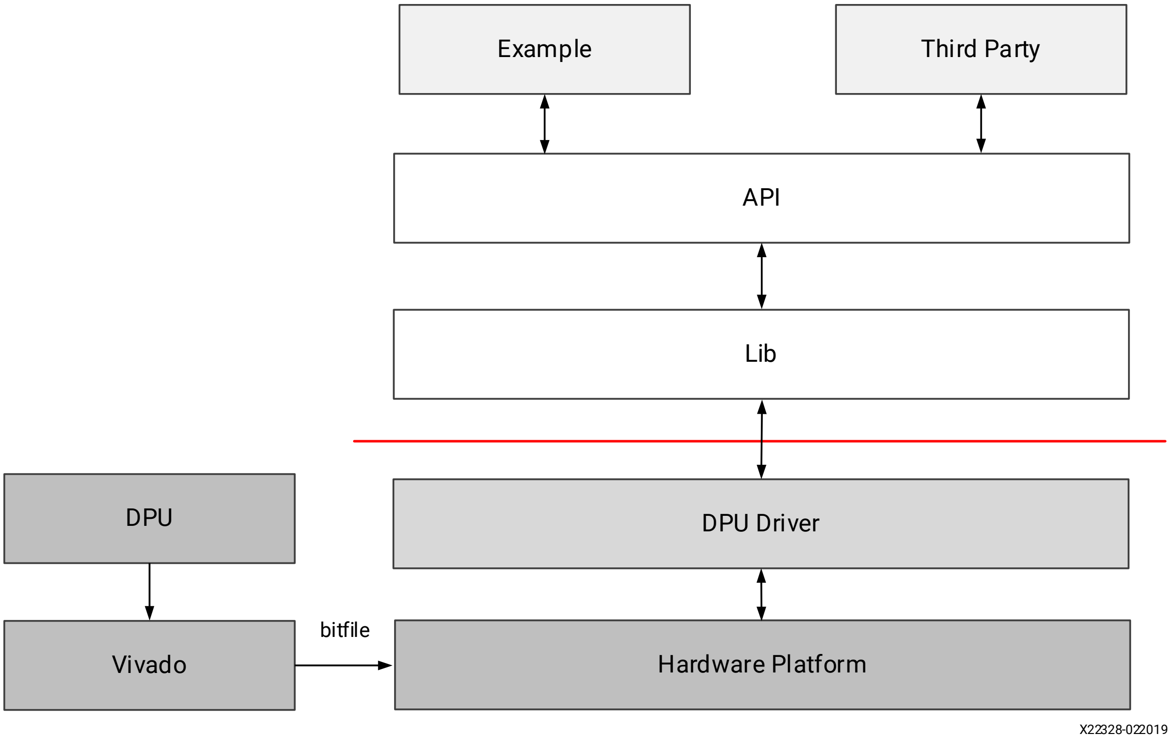


Figure 2: Basic Development Flow

Example System with DPU

The figure below shows an example system block diagram with the Xilinx UltraScale+ MPSoC using a camera input. DPU is integrated into the system through AXI interconnect to perform deep learning inference tasks such as image classification, object detection, and semantic segmentation.

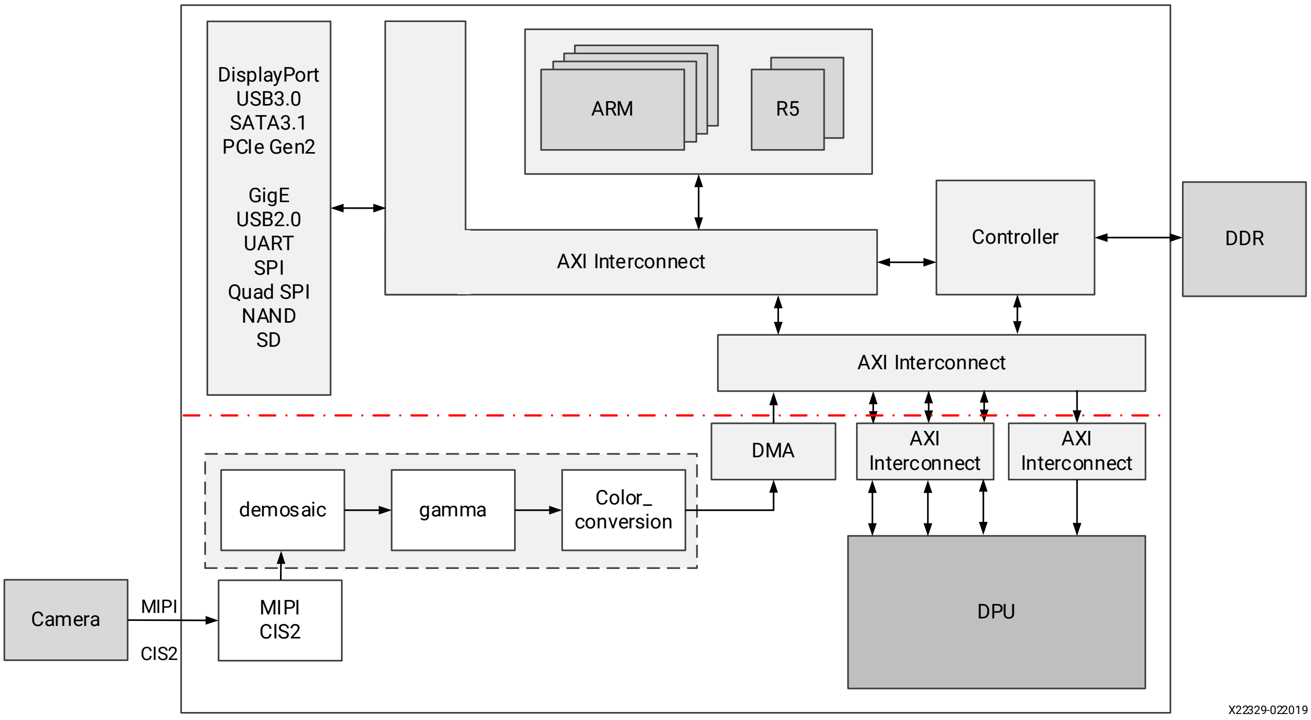


Figure 3: Example System with Integrated DPU

DNNDK

Deep Neural Network Development Kit (DNNDK) is a full-stack deep learning toolchain for inference with the DPU.

As shown in Figure 4, DNNDK is composed of Deep Compression Tool (DECENT), Deep Neural Network Compiler (DNNC), Neural Network Runtime (N2Cube), and DPU Profiler.

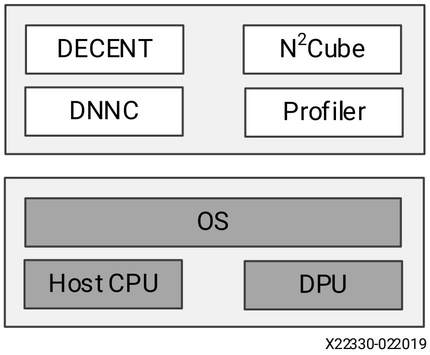


Figure 4: DNNDK Toolchain

The instructions of DPU are generated offline with DNNDK. Figure 5 illustrates the hierarchy of executing deep learning applications on the target hardware platform with DPU.

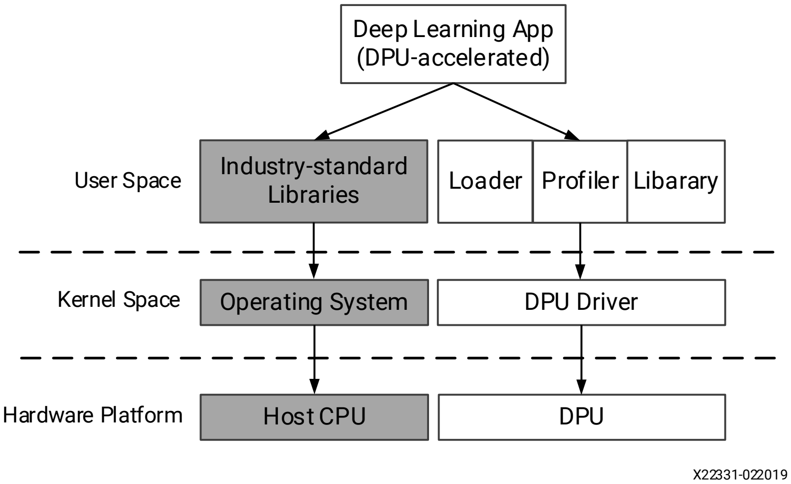


Figure 5: Application Execution Hierarchy

Licensing and Ordering Information

This IP module is provided at no additional cost under the terms of the Xilinx End User License.

Information about this and other IP modules is available at the Xilinx Intellectual Property page. For information on pricing and availability of other Xilinx IP modules and tools, contact your local Xilinx sales representative.

Hardware Architecture

The detailed hardware architecture of DPU is shown in Figure 6. After start-up, DPU fetches instructions from the off-chip memory and parses instructions to operate the computing engine. The instructions are generated by the DNNDK compiler where substantial optimizations have been performed.

To improve the efficiency, abundant on-chip memory in Xilinx® devices is used to buffer the intermediate data, input, and output data. The data is reused as much as possible to reduce the memory bandwidth. Deep pipelined design is used for the computing engine. Like other accelerators, the computational arrays (PE) take full advantage of the fine-grained building blocks, which includes multiplier, adder, accumulator, etc. in Xilinx devices.

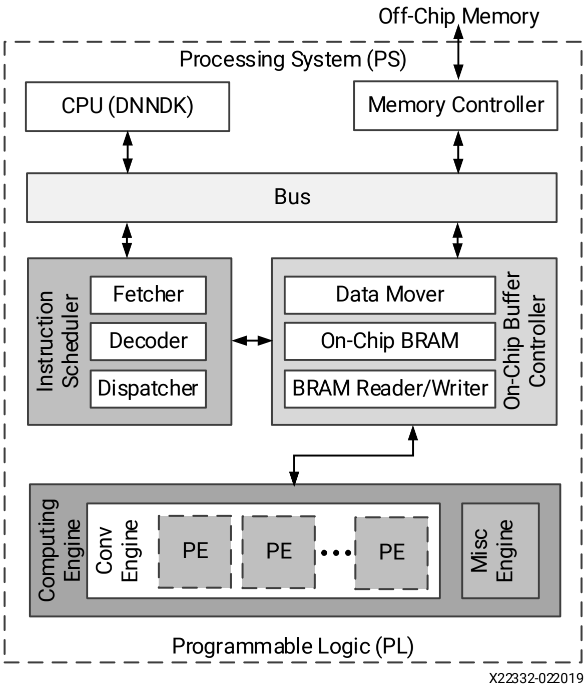


Figure 6: DPU Hardware Architecture

DSP with Enhanced Utilization (DPU_EU)

In the previous DPU version, the general logic and DSP slices work in the same clock domain, though technically the latter can run at a higher frequency. To enhance the utilization of DSP slices in DPU, the advanced DPU_EU version was designed.

The EU in "DPU_EU" means enhanced utilization of DSP slices. DSP Double Data Rate (DDR) technique is used to improve the performance achieved with the device. Therefore, two input clocks for DPU is needed, one for general logic, and the other for DSP slices. The difference between DPU and DPU_EU is shown in Figure 7.

All DPU mentioned in this document refer to DPU_EU, unless otherwise specified.

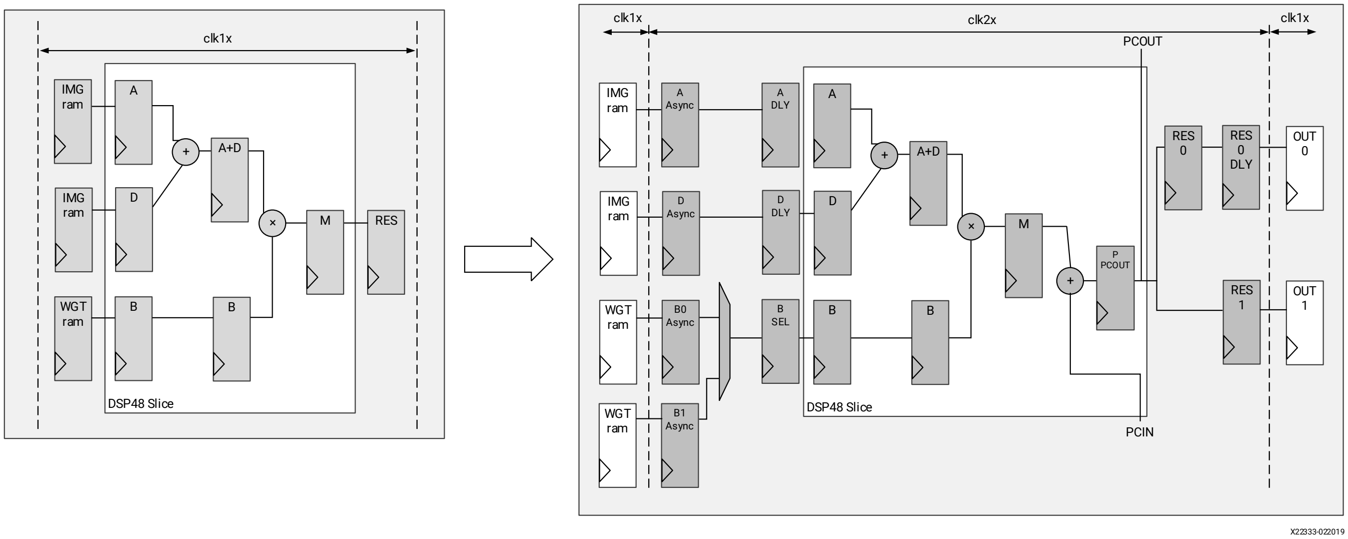


Figure 7: Difference between DPU and DPU_EU

Port Descriptions

The DPU top-level interfaces are shown in Figure 8.

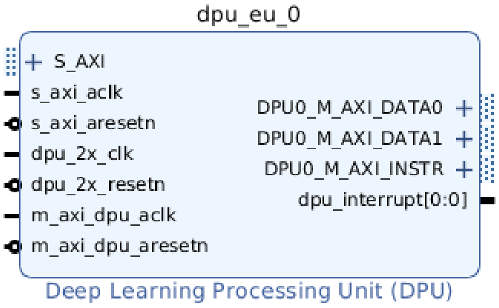


Figure 8: DPU_EU IP Port

The DPU I/O signals are listed and described in Table 1.

Table 1: DPU Signal Description

Singla Name	Interface Type	Width	I/O	Description
S_AXI	Memory mapped AXI slave interface	32	I/O	32-bit Memory mapped AXI interface for registers.
s_axi_aclk	Clock	1	I	AXI clock input for S_AXI
s_axi_aresetn	Reset	1	I	Active-Low reset for S_AXI
dpu_2x_clk	Clock	1	I	Input clock used for DSP unit in DPU. The frequency is two times of m_axi_dpu_aclk.
dpu_2x_resetn	Reset	1	I	Active-Low reset for DSP unit
m_axi_dpu_aclk	Clock	1	I	Input clock used for DPU general logic.
m_axi_dpu_aresetn	Reset	1	I	Active-Low reset for DPU general logic
DPUx_M_AXI_INSTR	Memory mapped AXI master interface	32	I/O	32-bit Memory mapped AXI interface for instruction of DPU.
DPUx_M_AXI_DATA0	Memory mapped AXI master interface	128	I/O	128-bit Memory mapped AXI interface for DPU data fetch.
DPUx_M_AXI_DATA1	Memory mapped AXI master interface	128	I/O	128-bit Memory mapped AXI interface for DPU data fetch.
dpu_interrupt	Interrupt	1~3	O	Active-High interrupt output from DPU. The data width is decided by the DPU number.

Notes:

1. If only input ports are needed, you can edit the ports in the block diagram and declare at the port interface level.

Register Space

The DPU IP implements registers in the programmable logic. Table 2 shows the DPU IP registers. These registers are accessible from the host CPU through the S_AXI interface.

Reg_dpu_reset

The reg_dpu_reset register controls the resets of all DPU cores integrated in the DPU IP. The lower three bits of this register control the reset of up to three DPU cores respectively. All the reset signals are active-High. The details of reg_dpu_reset is shown in Table 2.

Table 2: Reg_dpu_reset

Register	Address Offset	Width	Type	Description
Reg_dpu_reset	0x004	32	R/W	[0] – reset of DPU core 0 [1] – reset of DPU core 1 [2] – reset of DPU core 2

Reg_dpu_isr

The reg_dpu_isr register represents the interrupt status of all DPU cores integrated in the DPU IP. The lower three bits of this register shows the interrupt status of up to three DPU cores respectively. The details of reg_dpu_irq is shown in Table 3.

Table 3: Reg_dpu_isr

Register	Address Offset	Width	Type	Description
Reg_dpu_isr	0x608	32	R	[0] – interrupt status of DPU core 0 [1] – interrupt status of DPU core 1 [2] – interrupt status of DPU core 2

Reg_dpu_start

The reg_dpu_start register is the start signal for DPU core. There is one start register for each DPU core. The details of reg_dpu_start is shown in Table 4.

Table 4: Reg_dpu_start

Register	Address Offset	Width	Type	Description
Reg_dpu0_start	0x220	32	R/W	Control the start-up of DPU core0.
Reg_dpu1_start	0x320	32	R/W	Control the start-up of DPU core1.
Reg_dpu2_start	0x420	32	R/W	Control the start-up of DPU core2.

Reg_dpu_instr_addr

The reg_dpu_instr_addr register is used to indicate the instruction address of DPU core. There are three registers which are reg_dpu0_instr_addr, reg_dpu1_instr_addr, and reg_dpu2_instr_addr. The details of reg_dpu_instr_addr are shown in Table 5.

Table 5: Reg_dpu_instr_addr

Register	Address Offset	Width	Type	Description
Reg_dpu0_instr_addr	0x20c	32	R/W	[0] –The instruction start address in external memory for DPU core0.
Reg_dpu1_instr_addr	0x30c	32	R/W	[0] –The instruction start address in external memory for DPU core1.
Reg_dpu2_instr_addr	0x40c	32	R/W	[0] –The instruction start address in external memory for DPU core2.

Reg_dpu_base_addr

The reg_dpu_base_addr register is used to indicate the address of input image and parameters for DPU calculation in the external memory. The width of dpu_base_addr is 40 bits so it can support an address space up to 1 TB. All registers are 32 bits wide, so two registers are required to represent a 40-bit wide dpu_base_addr. The reg_dpu0_base_addr0_l represents the lower 32 bits of the base address0 in DPU core0, and the reg_dpu0_base_addr0_h represents the upper eight bits of the base address0 in DPU core0.

There are eight groups of DPU base address for each DPU core and in total 24 groups of DPU base address for up to three DPU cores. The details of reg_dpu_base_addr are shown in Table 6.

Table 6: Reg_dpu_base_addr

Register	Address Offset	Width	Type	Description
Reg_dpu0_base_addr0_l	0x224	32	R/W	The lower 32 bits of the base address0 of DPU core0.
Reg_dpu0_base_addr0_h	0x228	32	R/W	The lower 8 bits in the register represent the upper 8 bits of the base address0 of DPU core0.
Reg_dpu0_base_addr1_l	0x22C	32	R/W	The lower 32 bits of the base address1 of DPU core0.
Reg_dpu0_base_addr1_h	0x230	32	R/W	The lower 8 bits in the register represent the upper 8 bits of the base address1 of DPU core0.
Reg_dpu0_base_addr2_l	0x234	32	R/W	The lower 32 bits of the base address2 of DPU core0.

Reg_dpu0_base_addr2_h	0x238	32	R/W	The lower 8 bits in the register represent the upper 8 bits of the base address2 of DPU core0.
Reg_dpu0_base_addr3_l	0x23C	32	R/W	The lower 32 bits of the base address3 of DPU core0.
Reg_dpu0_base_addr3_h	0x240	32	R/W	The lower 8 bits in the register represent the upper 8 bits of the base address3 of DPU core0.
Reg_dpu0_base_addr4_l	0x244	32	R/W	The lower 32 bits of the base address0 of DPU core0.
Reg_dpu0_base_addr4_h	0x248	32	R/W	The lower 8 bits in the register represent the upper 8 bits of the base address0 of DPU core0.
Reg_dpu0_base_addr5_l	0x24C	32	R/W	The lower 32 bits of the base address1 of DPU core0.
Reg_dpu0_base_addr5_h	0x250	32	R/W	The lower 8 bits in the register represent the upper 8 bits of the base address1 of DPU core0.
Reg_dpu0_base_addr6_l	0x254	32	R/W	The lower 32 bits of the base address2 of DPU core0.
Reg_dpu0_base_addr6_h	0x258	32	R/W	The lower 8 bits in the register represent the upper 8 bits of the base address2 of DPU core0.
Reg_dpu0_base_addr7_l	0x25C	32	R/W	The lower 32 bits of the base address3 of DPU core0.
Reg_dpu0_base_addr7_h	0x260	32	R/W	The lower 8 bits in the register represent the upper 8 bits of the base address3 of DPU core0.
Reg_dpu1_base_addr0_l	0x324	32	R/W	The lower 32 bits of the base address0 of DPU core1.
Reg_dpu1_base_addr0_h	0x328	32	R/W	The lower 8 bits in the register represent the upper 8 bits of the base address0 of DPU core1.
Reg_dpu1_base_addr1_l	0x32C	32	R/W	The lower 32 bits of the base address1 of DPU core1.
Reg_dpu1_base_addr1_h	0x330	32	R/W	The lower 8 bits in the register represent the upper 8 bits of the base address1 of DPU core1.
Reg_dpu1_base_addr2_l	0x334	32	R/W	The lower 32 bits of the base address2 of DPU core1.
Reg_dpu1_base_addr2_h	0x338	32	R/W	The lower 8 bits in the register represent the upper 8 bits of the base address2 of DPU core1.
Reg_dpu1_base_addr3_l	0x33C	32	R/W	The lower 32 bits of the base address3 of DPU core1.
Reg_dpu1_base_addr3_h	0x340	32	R/W	The lower 8 bits in the register represent the upper 8 bits of the base address3 of DPU core1.

Reg_dpu1_base_addr4_l	0x344	32	R/W	The lower 32 bits of the base address0 of DPU core1.
Reg_dpu1_base_addr4_h	0x348	32	R/W	The lower 8 bits in the register represent the upper 8 bits of the base address0 of DPU core1.
Reg_dpu1_base_addr5_l	0x34C	32	R/W	The lower 32 bits of the base address1 of DPU core1.
Reg_dpu1_base_addr5_h	0x350	32	R/W	The lower 8 bits in the register represent the upper 8 bits of the base address1 of DPU core1.
Reg_dpu1_base_addr6_l	0x354	32	R/W	The lower 32 bits of the base address2 of DPU core1.
Reg_dpu1_base_addr6_h	0x358	32	R/W	The lower 8 bits in the register represent the upper 8 bits of the base address2 of DPU core1.
Reg_dpu1_base_addr7_l	0x35C	32	R/W	The lower 32 bits of the base address3 of DPU core1.
Reg_dpu1_base_addr7_h	0x360	32	R/W	The lower 8 bits in the register represent the upper 8 bits of the base address3 of DPU core1.
Reg_dpu2_base_addr1_l	0x42C	32	R/W	The lower 32 bits of the base address1 of DPU core2.
Reg_dpu2_base_addr1_h	0x430	32	R/W	The lower 8 bits in the register represent the upper 8 bits of the base address1 of DPU core2.
Reg_dpu2_base_addr2_l	0x434	32	R/W	The lower 32 bits of the base address2 of DPU core2.
Reg_dpu2_base_addr2_h	0x438	32	R/W	The lower 8 bits in the register represent the upper 8 bits of the base address2 of DPU core2.
Reg_dpu2_base_addr3_l	0x43C	32	R/W	The lower 32 bits of the base address3 of DPU core2.
Reg_dpu2_base_addr3_h	0x440	32	R/W	The lower 8 bits in the register represent the upper 8 bits of the base address3 of DPU core2.
Reg_dpu2_base_addr4_l	0x444	32	R/W	The lower 32 bits of the base address0 of DPU core2.
Reg_dpu2_base_addr4_h	0x448	32	R/W	The lower 8 bits in the register represent the upper 8 bits of the base address0 of DPU core2.
Reg_dpu2_base_addr5_l	0x44C	32	R/W	The lower 32 bits of the base address1 of DPU core2.
Reg_dpu2_base_addr5_h	0x450	32	R/W	The lower 8 bits in the register represent the upper 8 bits of the base address1 of DPU core2.
Reg_dpu2_base_addr6_l	0x454	32	R/W	The lower 32 bits of the base address2 of DPU core2.

Reg_dpu2_base_addr6_h	0x458	32	R/W	The lower 8 bits in the register represent the upper 8 bits of the base address2 of DPU core2.
Reg_dpu2_base_addr7_l	0x45C	32	R/W	The lower 32 bits of the base address3 of DPU core2.
Reg_dpu2_base_addr7_h	0x460	32	R/W	The lower 8 bits in the register represent the upper 8 bits of the base address3 of DPU core2.

Interrupts

Upon the completion of one DPU task, an interrupt from DPU occurs to signal the completion of the task. The active-High of the Reg_dpu0_start means the start of a DPU task for DPU core0. At the end of the DPU task, DPU sends an interrupt and one bit in the register Reg_dpu_isr is set to 1. The position of the active bit in the Reg_dpu_isr depends on the number of DPU cores. For example, when the DPU core1 finishes a task while the DPU core 0 is still working, the Reg_dpu_isr is set as 2, the lowest bit is 0, and the lower bit is 1.

The data width of dpu_interrupt is determined by the number of DPU cores. When the parameter of DPU_NUM is set to 2, it means the DPU IP is integrated with two DPU cores, and the data width of the dpu_interrupt signal is two bits. The lower bit represents the DPU core 0 interrupt and the higher bit represents the DPU core1 interrupt.

The interrupt connection between the DPU and PS is described in the Device Tree file, which indicates the interrupt number of DPU connected to the PS. The reference connection is shown as Figure 9.

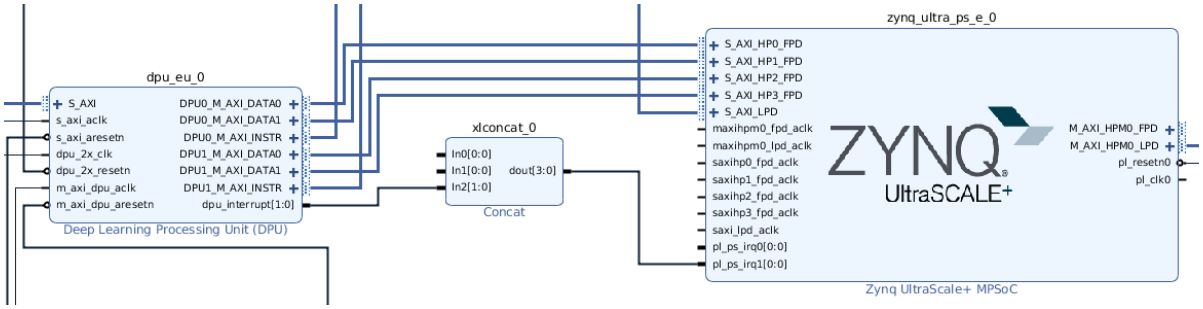


Figure 9: Reference Connection for DPU Interrupt

Notes:

1. If DPU is working with the DNNDK package, you should connect the dpu_interrupt at the 10th bit in the irq signal of PS. For example, if the DPU_NUM is set as 2, the 2-bit dpu_interrupt should connect with irq10 and irq11 of PS.
2. irq7~irq0 corresponds to pl_ps_irq0[7:0].
3. irq15~irq8 corresponds to pl_ps_irq1[7:0].

Introduction

The DPU IP provides some user-configurable parameters to optimize the resources or the support of different features. You can select different configurations to use on the preferred DSP slices, LUT, block RAM, and UltraRAM utilization based on the programmable logic resources that are allowed. There is also an option to determine how many DPU cores will be used.

The deep neural network features and the associated parameters supported by DPU is shown here.

Table 7: Deep Neural Network Features and Parameters Supported by DPU

Features	Description	
Convolution	Kernel Sizes	W: 1-16 H: 1-16
	Strides	W: 1-4 H:1-4
	Padding_w	1: kernel_w-1
	Padding_h	1: kernel_h-1
	Input Size	Arbitrary
	Input Channel	1 – 256*channel_parallel
	Output Channel	1 – 256*channel_parallel
	Activation	ReLU & LeakyReLU
	Dilation	dilation * input_channel <= 256 * channel_parallel && stride_w == 1 && stride_h == 1
Deconvolution	Kernel Sizes	W: 1-16 H: 1-16
	Stride_w	stride_w * output_channel <= 256 * channel_parallel
	Stride_h	Arbitrary
	Padding_w	1: kernel_w-1
	Padding_h	1: kernel_h-1
	Input Size	Arbitrary
	Input Channel	1 – 256 * channel_parallel
	Output Channel	1 – 256 * channel_parallel
	Activation	ReLU & LeakyReLU
Max Pooling	Kernel Sizes	W: 1-16 H: 1-16
	Strides	W: 1-4 H:1-4
	Padding	W: 1-4 H: 1-4
Element Wise	Input channel	1 – 256*channel_parallel

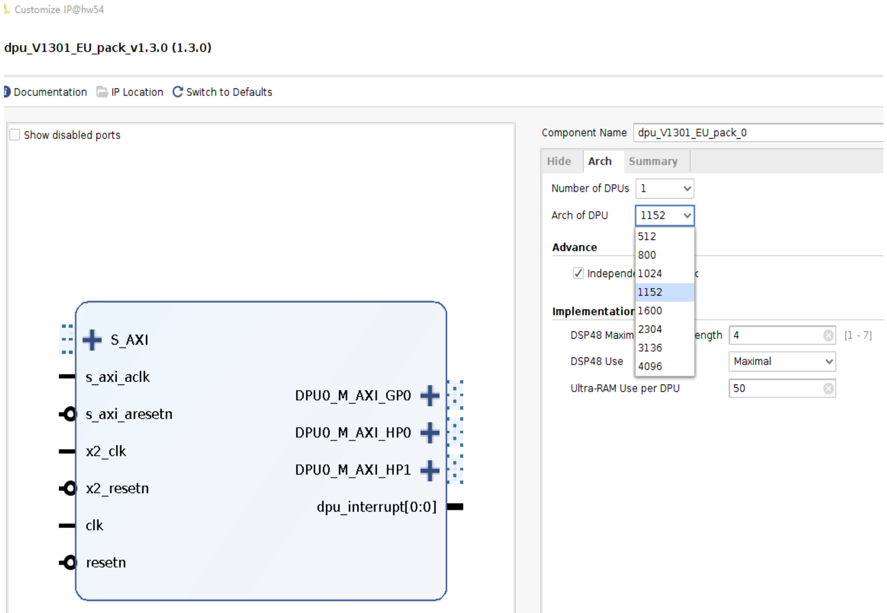
	Input size	arbitrary
Concat	Output channel	1 – 256*channel_parallel
Reorg	Strides	stride * stride * input_channel <= 256 * channel_parallel
FC	Input_channel	Input_channel <= 2048*channel_parallel
	Output_channel	Arbitrary

Notes:

1. The parameter channel_parallel is determined by the DPU configuration. For example, the channel_parallel of DPU-B1152 is 12, the channel_parallel of DPU-B4096 is 16.

Configuration Options

You can configure the DPU with some predefined options which includes DPU core number, DPU convolution architecture, DSP cascade, DSP usage, and UltraRAM usage. These options enable the DPU IP configurable in terms of DSP slice, LUT, block RAM, and UltraRAM utilization. Figure 10 shows the DPU configuration.



Customize IP@hw54

dpu_V1301_EU_pack_v1.3.0 (1.3.0)

Component Name: dpu_V1301_EU_pack_0

Number of DPUs: 1

Arch of DPU: 1152

☑ Independe

DSP48 Maxim... ength: 4 [1 - 7]

DSP48 Use: Maximal

Ultra-RAM Use per DPU: 50

Figure 10: DPU Configuration

DPU Core Number

You can use up to three DPU cores can be included in one IP. Multiple DPU cores can be used to achieve higher performance. Consequently, it consumes more programmable logic resource.

If the requirement is to integrate more than three cores, send the request to a Xilinx sales representative.

DPU Convolution Architecture

The DPU IP can be configured with different convolution architectures which is related to the parallelism of the convolution unit. The optional architecture for DPU IP includes B512, B800, B1024, B1152, B1600, B2304, B3136, and B4096.

There are three dimensions of parallelism in the DPU convolution architecture - pixel parallelism, input channel parallelism, and output channel parallelism. The input channel parallelism is always equal to the output channel parallelism. The different convolution architecture requires different programmable logic resource. The larger convolution architecture can achieve higher performance with more resources. The parallelism for different convolution architecture is listed in Table 8.

Table 8: Parallelism for Different Convolution Architecture

Convolution Architecture	Pixel Parallelism (PP)	Input Channel Parallelism (ICP)	Output Channel Parallelism (OCP)	Peak Ops (operations/per clk)
B512	4	8	8	512
B800	4	10	10	800
B1024	8	8	8	1024
B1152	4	12	12	1150
B1600	8	10	10	1600
B2304	8	12	12	2304
B3136	8	14	14	3136
B4096	8	16	16	4096

Notes:

1. In each clock cycle, the convolution array finishes a multiplication and an accumulation, which are two operations. So, the peak operations per cycle is equal to PP*ICP*OCP*2.

DSP Cascade

You can select the maximal length of DSP48E slice cascade chain. Typically, the larger cascade length indicates less logic resources, but it might lead to worse timing. The smaller cascade length might use more fabric resources which is not economical for small devices. Xilinx recommends selecting the mid-value, which is 4, in the first iteration and adjust the value if the timing is not met.

DSP Usage

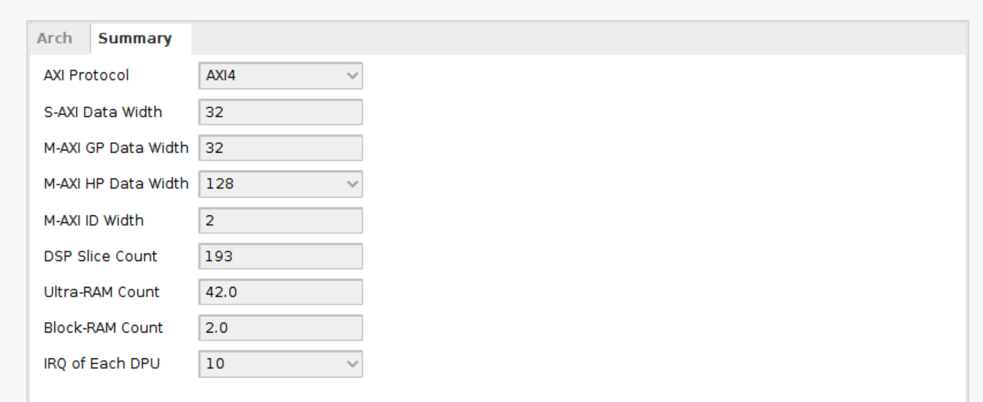
You can select whether DSP48E slices are used for the accumulation in the DPU convolution module. If the low DSP usage is selected, the DPU IP will use DSP slices for multiplication only in the convolution. In the high DSP usage mode, the DSP slice will be used for both multiplication and accumulation. As a result, the high DSP usage consumes more DSP slices and less LUT. The difference of logic utilization between high and low DSP usage is shown in Table 9. The data is tested on the Xilinx ZCU102 platform without Depthwise Conv, Average Pooling, Relu6, and Leaky Relu features.

Table 9: Resources of Different DSP Usage

High DSP Usage					Low DSP Usage				
Arch	**LUT**	**Register**	**BRAM**	**DSP**	**Arch**	**LUT**	**Register**	**BRAM**	**DSP**
B512	20177	31782	69.5	98	B512	20759	33572	69.5	66
B800	20617	35065	87	142	B800	21050	33752	87	102
B1024	27377	46241	101.5	194	B1024	29155	49823	101.5	130
B1152	28698	46906	117.5	194	B1152	30043	49588	117.5	146
B1600	30877	56267	123	282	B1600	33130	60739	123	202
B2304	34379	67481	161.5	386	B2304	37055	72850	161.5	290
B3136	38555	79867	203.5	506	B3136	41714	86132	203.5	394
B4096	40865	92630	249.5	642	B4096	44583	99791	249.5	514

UltraRAM

Some Zynq MPSoC devices have both block RAM (BRAM) and UltraRAM as the on-chip memory. The DPU IP uses BRAM by default, however, you can use UltraRAM to replace BRAM according to the available memory resources. Note the number filled in the UltraRAM row in Figure 10 might not be the actual utilization, because the minimum cache unit in DPU needs two or three UltraRAM depending on the different DPU architecture. The final utilization is shown in Figure 11.



AXI Protocol	AXI4
S-AXI Data Width	32
M-AXI GP Data Width	32
M-AXI HP Data Width	128
M-AXI ID Width	2
DSP Slice Count	193
Ultra-RAM Count	42.0
Block-RAM Count	2.0
IRQ of Each DPU	10

Figure 11: Summary Page of DPU Configuration

DPU Performance on Different Devices

Table 10 shows the peak performance of the DPU on different devices.

Table 10: DPU_EU Performance (GOPs) on Different Device

Device	DPU Configuration	Frequency (MHz)	Peak Performance
Z7020	B1152x1	200	230 Gops
ZU2	B1152x1	370	426 Gops
ZU3	B2304x1	370	852 Gops
ZU5	B4096x1	350	1.4 Tops
ZU7EV	B4096x2	330	2.7 Tops
ZU9	B4096x3	333	4.1 Tops

Performance of Different Models

In this section, the performance of several models is given for reference. The result was measured on the Xilinx ZCU102 board with 3x B4096_EU cores at 333 MHz (DSP slices ran at 666 MHz) and DNNDK v2.08, shown in Table 11.

Table 11: Performance of Different Models

Network Model	Workload (Gops per image)	Input Image Resolution	Frame per second (FPS)
Inception-v1	3.2	224*224	405
ResNet50	7.7	224*224	175
SqueezeNet	0.698	224*224	1048
Tiny-YOLO	6.97	448*448	220
YOLO-V2	82.5	640*640	24
Pruned YOLO-V2	18.4	640*640	120
YOLO-V3	53.7	512*256	43
Pruned YOLO-V3-	4	512*256	115

Notes:

1. The pruned models were generated by the Xilinx pruning tool.

I/O Bandwidth Requirements

When different neural networks run in the DPU, the I/O bandwidth requirement is different. Even the I/O bandwidth requirement of different layers in one neural network are different. The I/O bandwidth requirements for some neural networks, averaged by layer have been tested with one DPU core running at full speed. The peak and average I/O bandwidth requirements of three different neural networks are shown in Table 12. The table only shows the number of two commonly used DPU (B1152 and B4096). Note that when multiple DPU cores run in parallel, each core might not be able to run at full speed due to the limitation of I/O bandwidth.

Table 12: I/O Bandwidth Requirements for DPU-B1152 and DPU-B4096

Network Model	DPU-B1152		DPU-4096	
	Peak (MB/s)	Average (MB/s)	Peak (MB/s)	Average (MB/s)
Inception-v1	1704	890	4626	2474
Resnet50	2052	1017	5298	3132
SSD	1516	684	5724	2049
Pruned YOLO-V3	2076	986	6453	3290

If one DPU core needs to run at full speed, the peak I/O bandwidth required shall be met. The I/O bandwidth is mainly used for accessing data though the AXI master interfaces (Dpu0_M_AXI_DATA0 and Dpu0_M_AXI_DATA1).

Introduction

There are three clock domains in the DPU IP: the register, the data controller, and the computation unit. The three input clocks can be configured by you depending on the requirements. Therefore, the corresponding reset for the three input clocks shall be configured correctly.

Clock Domain

Figure 12 shows the three clock domains.

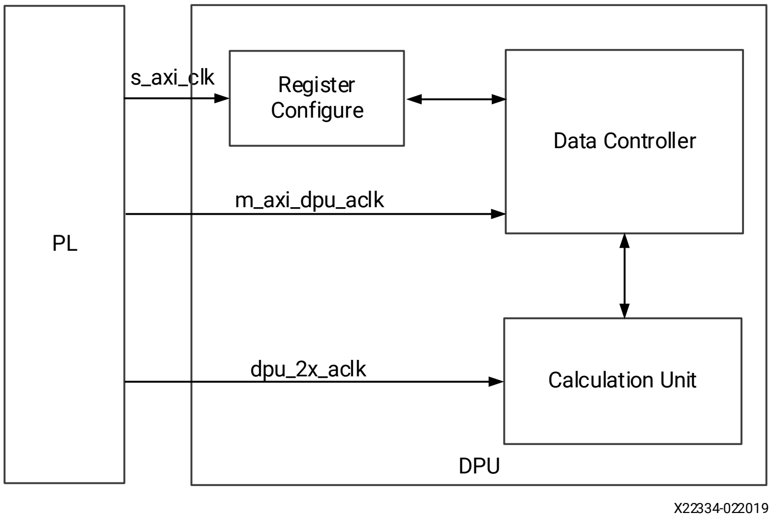


Figure 12: Clock Domain in DPU

Register Clock

The input s_axi_clk is used for the register configure module. This module receives the DPU configure data though the S_AXI interface and the related clock of S_AXI is s_axi_clock. The register for DPU configure is updated at a very low frequency and most of those registers are configured at the start of a task. Xilinx recommends that the frequency of the s_axi_clock is set as 100 MHz.

Data Controller Clock

The primary function of the data controller module is to schedule the data flow in the DPU IP. The data controller module works with m_axi_dpu_aclk. The data transfer between DPU and external memory happens in the data controller clock domain, so m_axi_dpu_aclk is also the AXI clock for the AXI_MM master interface in the DPU IP. You should connect m_axi_dpu_aclk with the AXI_MM master clock.

Computation Clock

The DSP slides in the computation unit module are in the dpu_2x_clk domain, which doubles the clock frequency of the data controller module. Therefore, the frequency of the dpu_2x_clk should be twice the m_axi_dpu_aclk. Furthermore, the two related clocks must be edge-aligned.

Reference Clock Generation

There are three input clocks for the DPU in which the frequency of the dpu_2x_clk should be two times the m_axi_dpu_aclk and the two clocks must be synchronous to meet the timing closure. The recommended circuit design is shown in Figure 13.

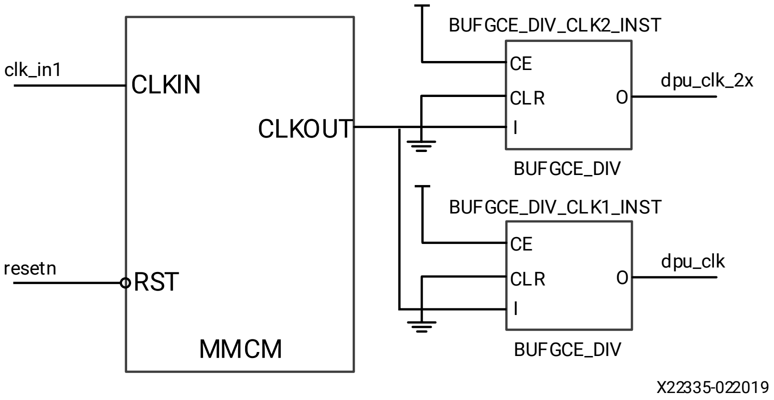
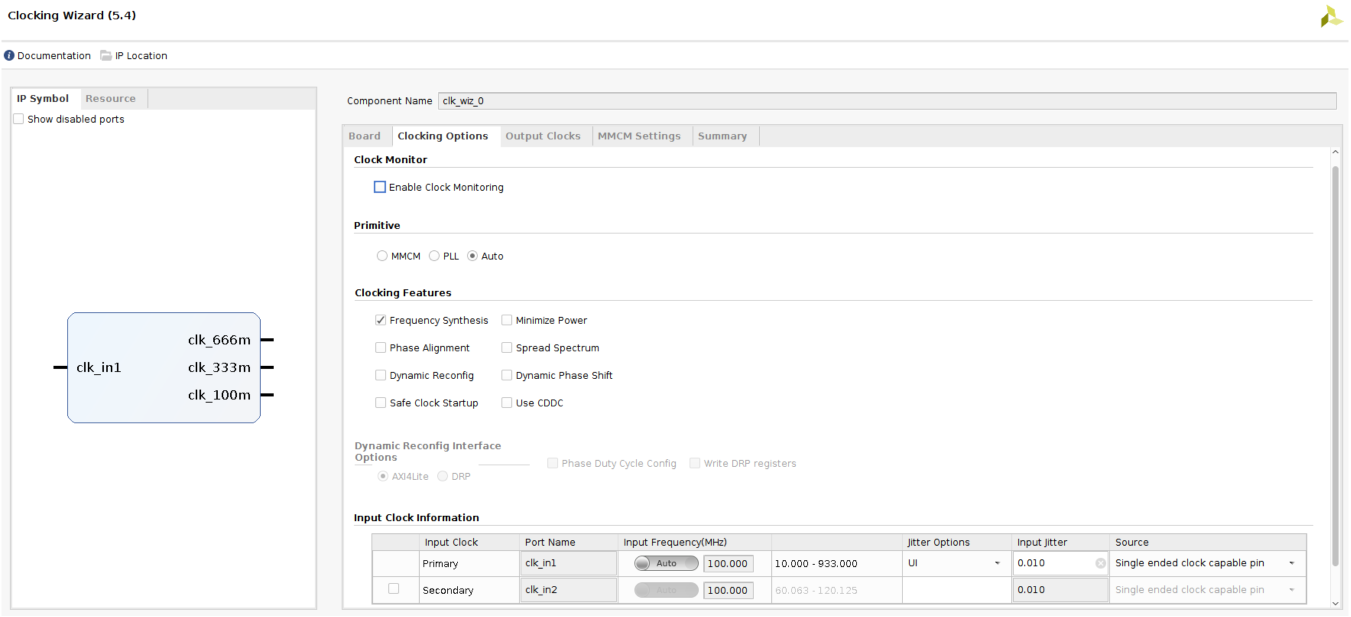


Figure 13: Reference Circuit

You can instantiate an MMCM and two BUFGCE_DIV to design this circuit. The frequency of clk_in1 is arbitrary and the frequency of output clock CLKOUT in the MMCM should be the frequency of dpu_clk_2x. The BUFGCE_DIV_CLK1_INST obtains the clock of whichever frequency is half of the dpu_clk_2x. The dpu_clk and dpu_clk_2x are generated by the same clock, so they are synchronous. The two BUFGCE_DIVs enable the skew between the two clocks to significantly decrease, which helps with timing closure.

Using Clock Wizard

You can instantiate a clock wizard IP to generate the above circuit. In this reference design, the frequency of s_axi_aclk is set to 100 MHz and m_axi_dpu_aclk is set to 333 MHz, which is also the maximum frequency of AXI-HP interfaces in Xilinx UltraScale+ MPSoC devices. Therefore, the frequency of the dpu_2x_clk should be set to 666 MHz. The recommended configuration of the Clocking Options tab is shown in Figure 14. Note that the parameter of the Primitive must be set to Auto.



Clocking Wizard (5.4)

□ Show disabled ports

Component Name: clk_wiz_0

Board | Clocking Options | Output Clocks | MMCM Settings | Summary

Clock Monitor

□ Enable Clock Monitoring

Primitive

○ MMCM ○ PLL ● Auto

Clocking Features

☑ Frequency Synthesis	□ Minimize Power
□ Phase Alignment	□ Spread Spectrum
□ Dynamic Reconfig	□ Dynamic Phase Shift
□ Safe Clock Startup	□ Use CDDC

Dynamic Reconfig Interface Options

● AXI4Lite ○ DRP □ Phase Duty Cycle Config □ Write DRP registers

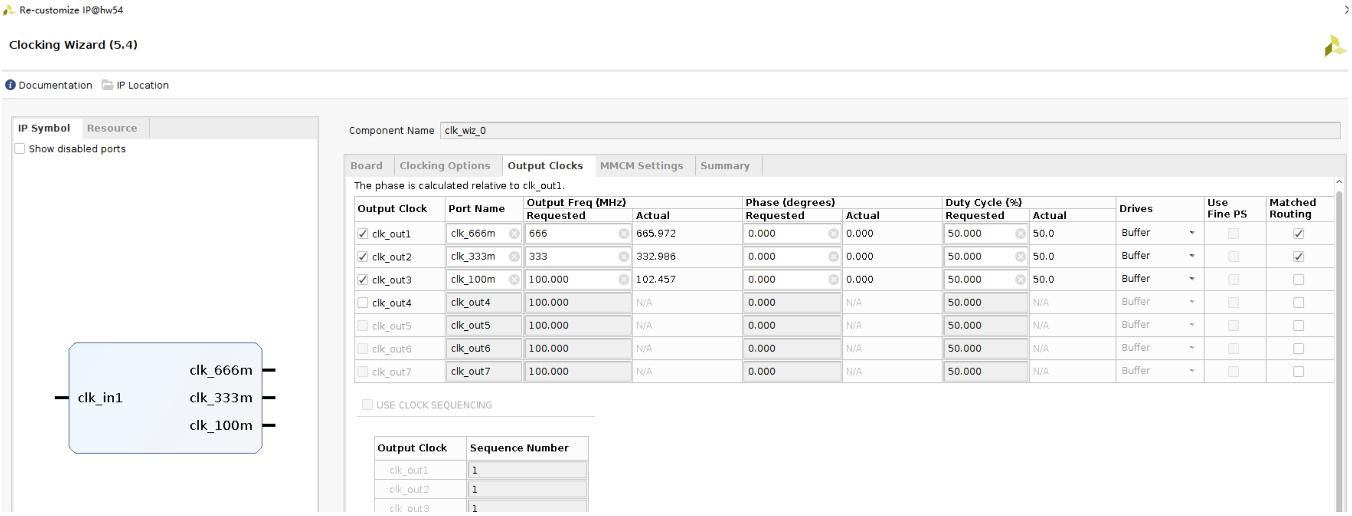
Input Clock Information

	Input Clock	Port Name	Input Frequency(MHz)			Jitter Options	Input Jitter	Source
	Primary	clk_in1	Auto	100.000	10.000 - 933.000	UI	0.010	Single ended clock capable pin
□	Secondary	clk_in2	Auto	100.000	60.063 - 120.125		0.010	Single ended clock capable pin

Figure 14: Recommended Clocking Options of Clock Wizard

Matched Routing

Select the Matched Routing for the m_axi_dpu_aclk and dpu_2x_clk in the Output Clocks tab of the Clock Wizard IP. When the Matched Routing setting enables the two clocks that are both generated through a BUFGCE_DIV, the skew between the two clocks has significantly decreased. The related configuration is shown in Figure 15.



Clocking Wizard (5.4)

□ Show disabled ports

Component Name: clk_wiz_0

Board | Clocking Options | Output Clocks | MMCM Settings | Summary

The phase is calculated relative to clk_out1.

Output Clock	Port Name	Output Freq (MHz)		Phase (degrees)		Duty Cycle (%)		Drives	Use Fine PS	Matched Routing
		Requested	Actual	Requested	Actual	Requested	Actual			
☑ clk_out1	clk_666m	666	665.972	0.000	0.000	50.000	50.0	Buffer	□	☑
☑ clk_out2	clk_333m	333	332.986	0.000	0.000	50.000	50.0	Buffer	□	☑
☑ clk_out3	clk_100m	100.000	102.457	0.000	0.000	50.000	50.0	Buffer	□	□
□ clk_out4	clk_out4	100.000	N/A	0.000	N/A	50.000	N/A	Buffer	□	□
□ clk_out5	clk_out5	100.000	N/A	0.000	N/A	50.000	N/A	Buffer	□	□
□ clk_out6	clk_out6	100.000	N/A	0.000	N/A	50.000	N/A	Buffer	□	□
□ clk_out7	clk_out7	100.000	N/A	0.000	N/A	50.000	N/A	Buffer	□	□

□ USE CLOCK SEQUENCING

Output Clock	Sequence Number
clk_out1	1
clk_out2	1
clk_out3	1

Figure 15: Matched Routing in Clock Wizard

Reset

There are three input clocks for the DPU IP, each of which has a corresponding reset. You must guarantee each pair of clocks and resets is generated in a synchronous clock domain. If the related clocks and resets are not matched, the DPU might not work properly. A recommended solution is to instantiate a Processor System Reset IP to generate a matched reset for each clock. The reference design is shown in Figure 16.

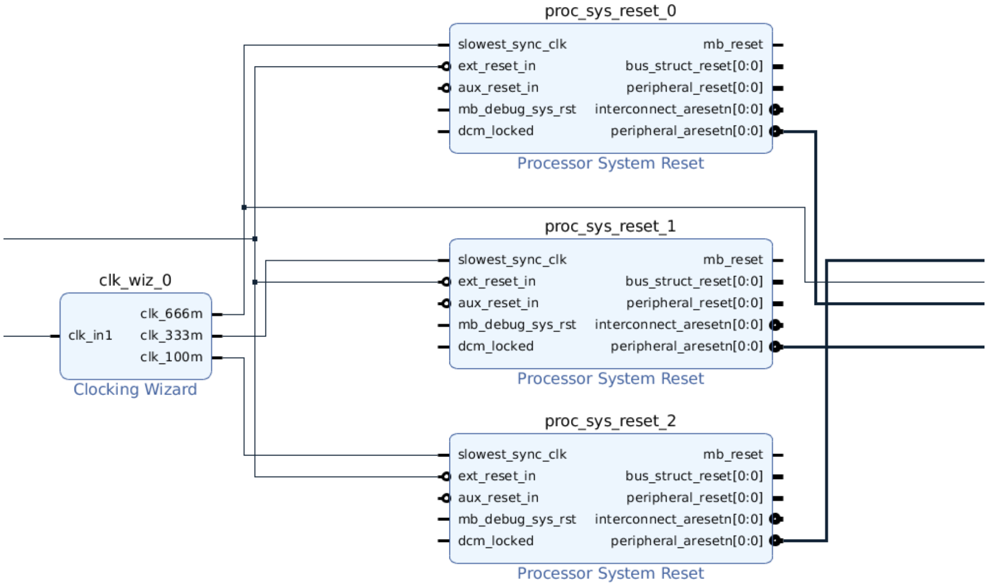


Figure 16: Reference Design for Resets

Customizing and Generating the Core in MPSoC

The following sections describe the development flow on how to use the DPU IP with the Vivado® Design Suite:

- Add DPU IP into Repository
- Add DPU IP into Block Design
- Configure DPU Parameters
- Connect DPU with a Processing System in the Xilinx SoC
- Assign Register Address for DPU
- Generate Bitstream
- Generate BOOT.BIN

Add DPU IP into Repository

In the Vivado GUI, click **Project Manager** > **IP Catalog**. In the IP Catalog tab, right-click and select **Add Repository** (Figure 17), then select the location of the DPU IP. This will appear in the IP Catalog page (Figure 18).

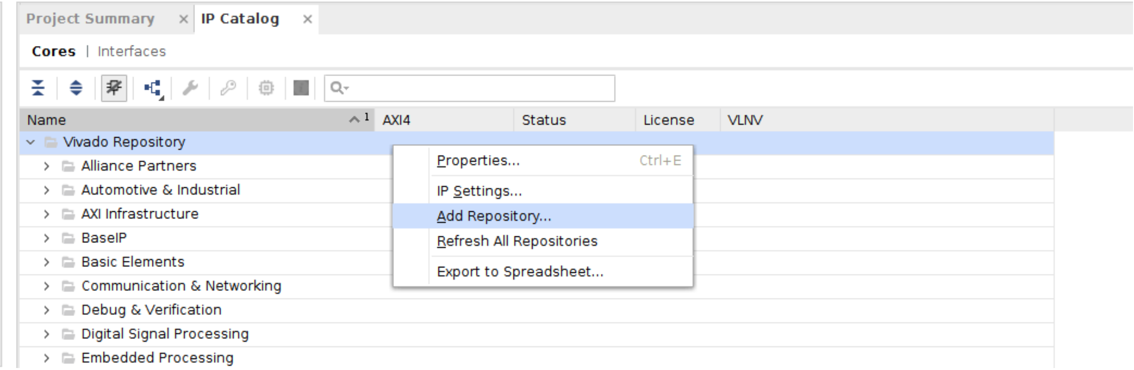
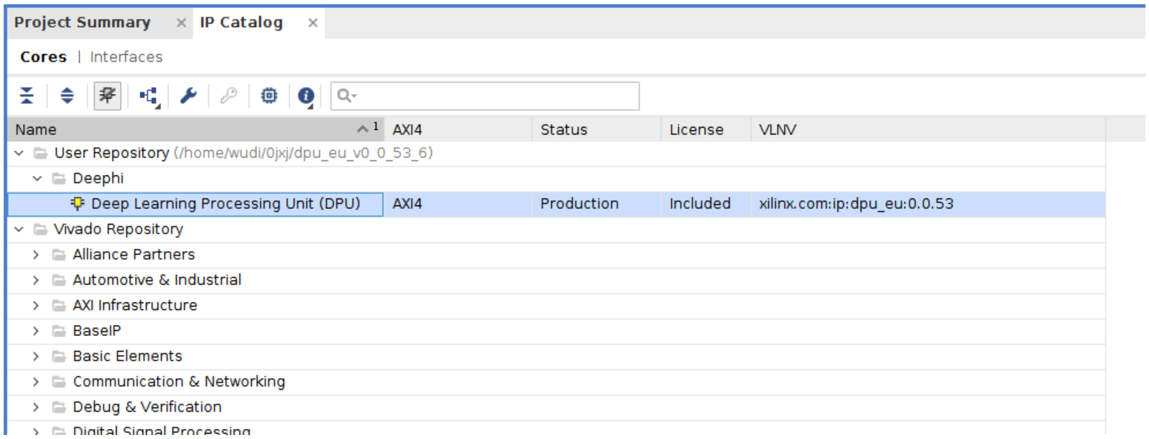


Figure 17: Add Repository



Name	AXI4	Status	License	VLNV
User Repository (/home/wudi/0jxj/dpu_eu_v0_0_53_6)				
Deephi				
Deep Learning Processing Unit (DPU)	AXI4	Production	Included	xilinx.com:ip:dpu_eu:0.0.53
Vivado Repository				
Alliance Partners				
Automotive & Industrial				
AXI Infrastructure				
BaseIP				
Basic Elements				
Communication & Networking				
Debug & Verification				
Digital Signal Processing				

Figure 18: DPU IP in Repository

Add DPU IP into Block Design

Search DPU IP in the block design interface and add DPU IP into the block design. The procedure is shown in Figure 19 and Figure 20.

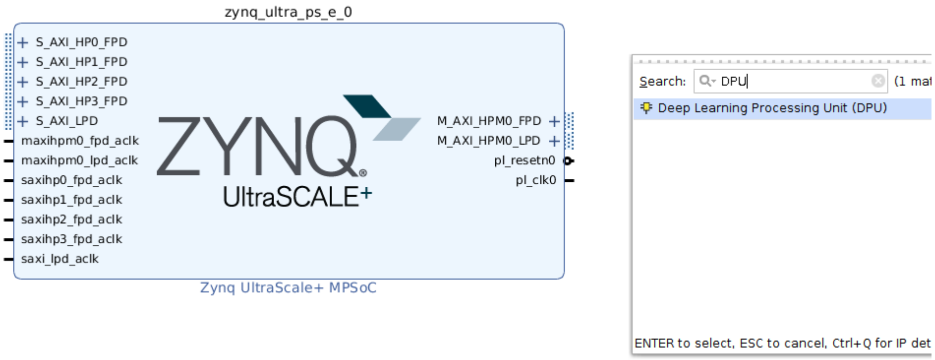


Figure 19: Search DPU IP

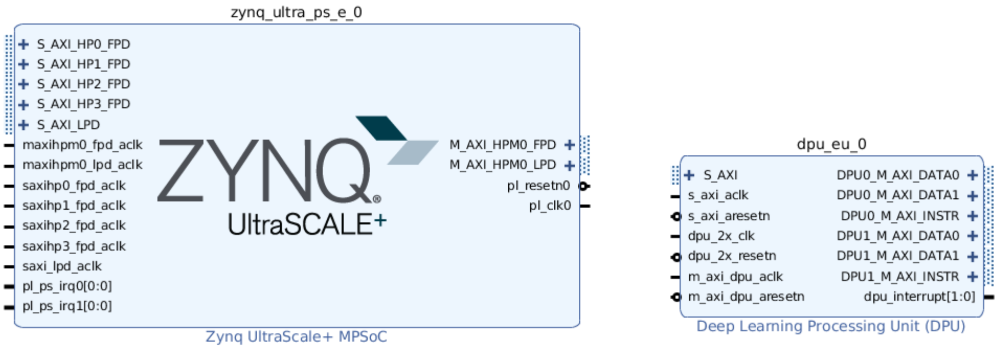
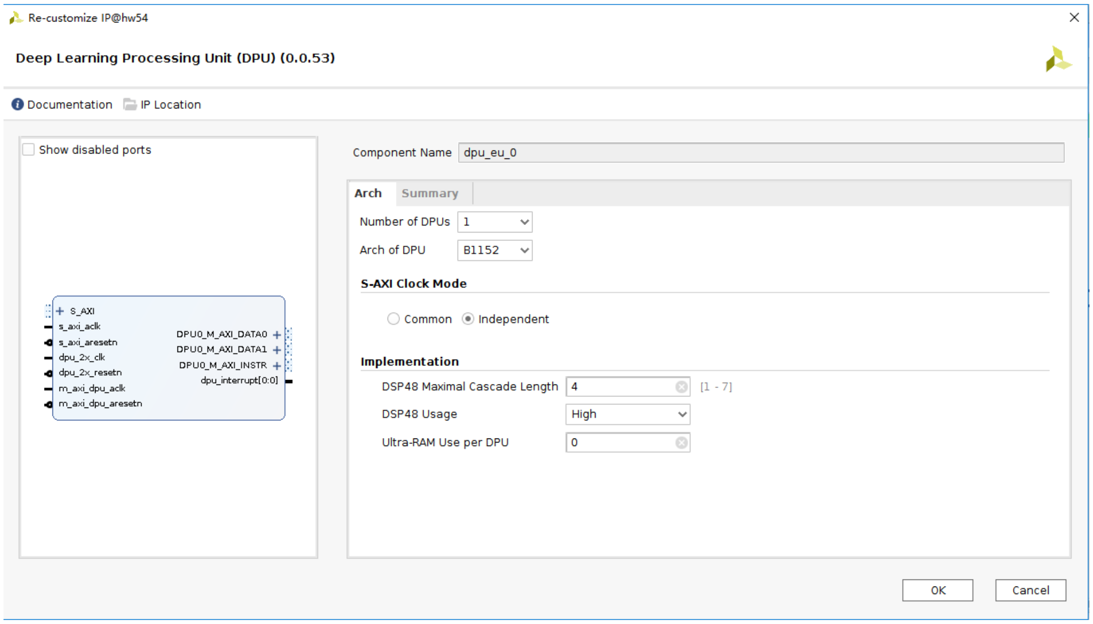


Figure 20: Add DPU IP into Block Design

Configure DPU Parameters

You can configure the DPU IP as shown in Figure 21. The details about these parameters can be found in Chapter 3: DPU Configuration.



Component Name: dpu_eu_0

Number of DPUs: 1

Arch of DPU: B1152

S-AXI Clock Mode

○ Common ● Independent

Implementation

DSP48 Maximal Cascade Length: 4 [1 - 7]

DSP48 Usage: High

Ultra-RAM Use per DPU: 0

Figure 21: Configure DPU

Connect DPU with a Processing System in the Xilinx SoC

No matter how many DPU cores are configured, there is only one slave interface in the DPU IP. Each DPU core has three master interfaces, one for instruction fetch and the other two for data fetch. The number of master interfaces in the DPU IP depends on the DPU_NUM parameter.

You can connect the DPU to a processing system (PS) with any kind of interconnections. You must ensure the DPU can correctly access the DDR memory space. Generally, an AXI data transaction passes an Interconnect IP, the delay of data transaction will increase. The delay of the data transmission between the DPU and the Interconnect will reduce the DPU performance. Therefore, Xilinx recommends that each data fetch master interface in the DPU connects to the PS through a direct connection rather than through an AXI Interconnect IP. The reference connection between the DPU and PS in the Xilinx UltraScale+ MPSoC is shown as Figure 22.

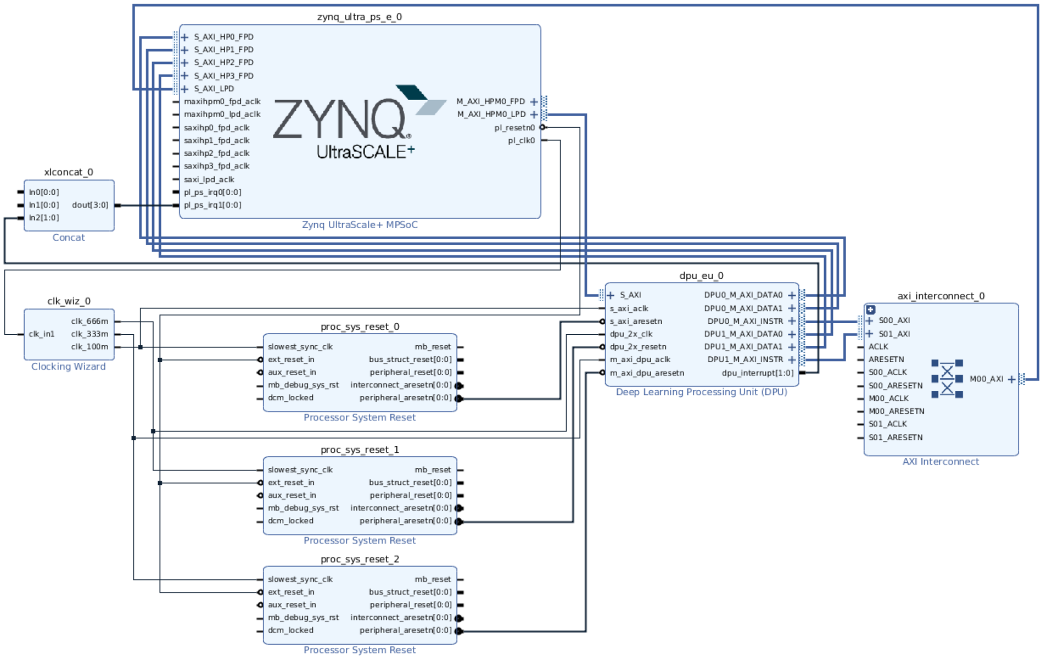


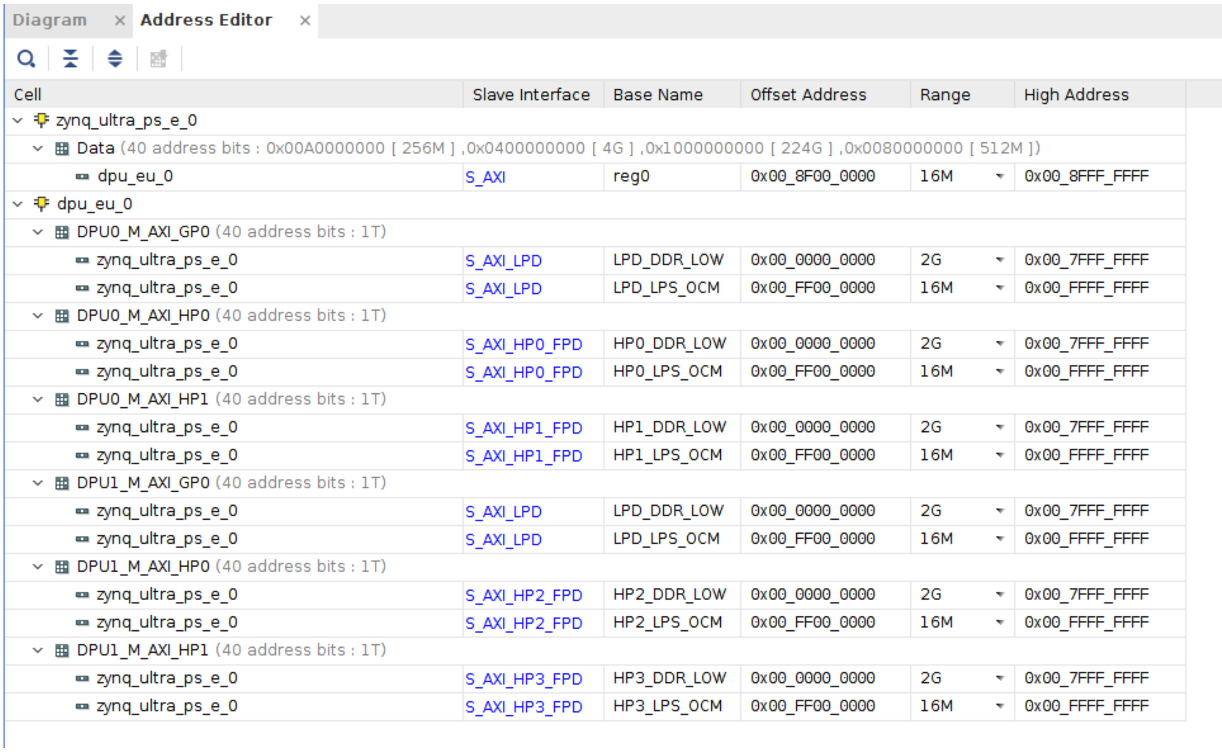
Figure 22: Connect DPU with PS of MPSoC

Assign Register Address for DPU

When the DPU connection is complete, the next step is to assign the register address of the AXI slave interface. The minimum space needed for the DPU is 16 MB. The DPU slave interface can be assigned to any starting address accessible by the host CPU.

Note: *When building a custom system with the pre-built Linux environment in the DNNDK package, the DPU slave interface must be connected to the M_AXI_HPM0_LPD PS Master and the DPU base address must be set to 0x8F00_000 with a range of 16 MB. The DPU register address in the driver and device tree file in the DNNDK package is fixed at 0x8F00_0000. If the address in the driver and device tree file is the same as the address assigned in Vivado, you can connect the DPU slave interface to any master interface in the PS and allocate any address for the DPU.*

The reference address assignments of the DPU with the DNNDK package are shown in Figure 23.

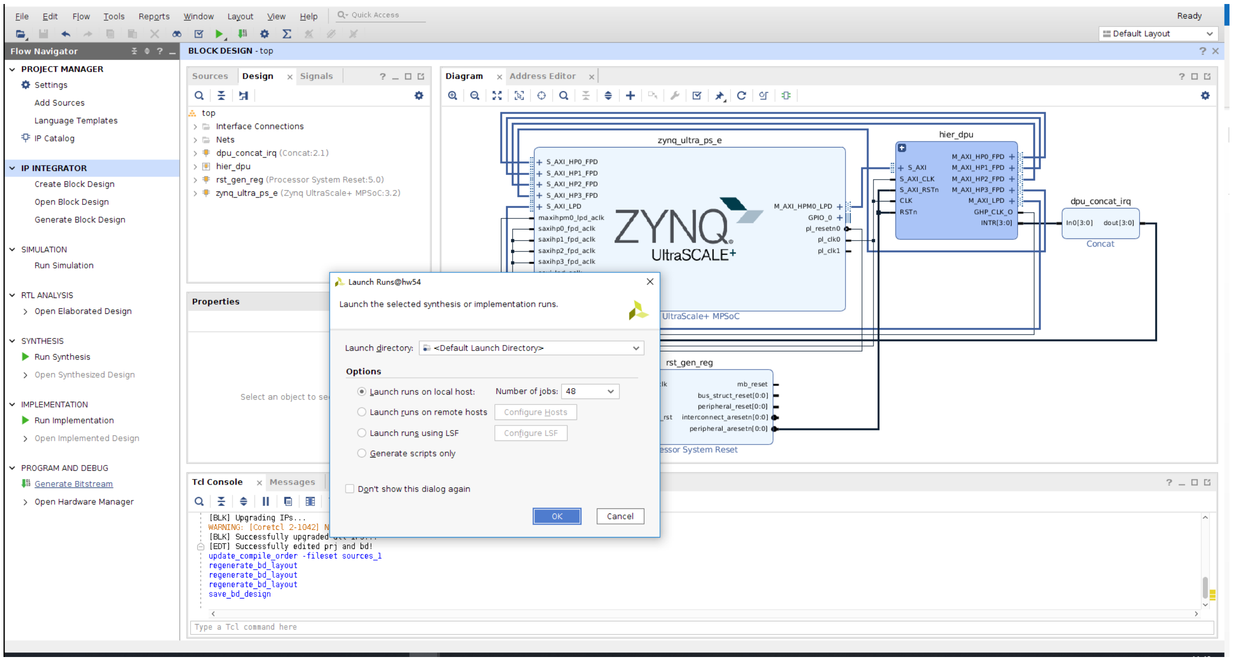


Cell	Slave Interface	Base Name	Offset Address	Range	High Address
zynq_ultra_ps_e_0					
Data (40 address bits : 0x00A0000000 [256M] ,0x0400000000 [4G] ,0x1000000000 [224G] ,0x0080000000 [512M])					
dpu_eu_0	S_AXI	reg0	0x00_8F00_0000	16M	0x00_8FFF_FFFF
dpu_eu_0					
DPU0_M_AXI_GP0 (40 address bits : 1T)					
zynq_ultra_ps_e_0	S_AXI_LPD	LPD_DDR_LOW	0x00_0000_0000	2G	0x00_7FFF_FFFF
zynq_ultra_ps_e_0	S_AXI_LPD	LPD_LPS_OCM	0x00_FF00_0000	16M	0x00_FFFF_FFFF
DPU0_M_AXI_HP0 (40 address bits : 1T)					
zynq_ultra_ps_e_0	S_AXI_HP0_FPD	HP0_DDR_LOW	0x00_0000_0000	2G	0x00_7FFF_FFFF
zynq_ultra_ps_e_0	S_AXI_HP0_FPD	HP0_LPS_OCM	0x00_FF00_0000	16M	0x00_FFFF_FFFF
DPU0_M_AXI_HP1 (40 address bits : 1T)					
zynq_ultra_ps_e_0	S_AXI_HP1_FPD	HP1_DDR_LOW	0x00_0000_0000	2G	0x00_7FFF_FFFF
zynq_ultra_ps_e_0	S_AXI_HP1_FPD	HP1_LPS_OCM	0x00_FF00_0000	16M	0x00_FFFF_FFFF
DPU1_M_AXI_GP0 (40 address bits : 1T)					
zynq_ultra_ps_e_0	S_AXI_LPD	LPD_DDR_LOW	0x00_0000_0000	2G	0x00_7FFF_FFFF
zynq_ultra_ps_e_0	S_AXI_LPD	LPD_LPS_OCM	0x00_FF00_0000	16M	0x00_FFFF_FFFF
DPU1_M_AXI_HP0 (40 address bits : 1T)					
zynq_ultra_ps_e_0	S_AXI_HP2_FPD	HP2_DDR_LOW	0x00_0000_0000	2G	0x00_7FFF_FFFF
zynq_ultra_ps_e_0	S_AXI_HP2_FPD	HP2_LPS_OCM	0x00_FF00_0000	16M	0x00_FFFF_FFFF
DPU1_M_AXI_HP1 (40 address bits : 1T)					
zynq_ultra_ps_e_0	S_AXI_HP3_FPD	HP3_DDR_LOW	0x00_0000_0000	2G	0x00_7FFF_FFFF
zynq_ultra_ps_e_0	S_AXI_HP3_FPD	HP3_LPS_OCM	0x00_FF00_0000	16M	0x00_FFFF_FFFF

Figure 23: Assign DPU Address

Generate Bitstream

Click **Generate Bitstream** in Vivado shown in Figure 24.



Launch Runs@hw54

Launch the selected synthesis or implementation runs.

Launch directory: <Default Launch Directory>

Options

- ● Launch runs on local host: Number of jobs: 48
- ○ Launch runs on remote hosts [Configure Hosts]
- ○ Launch runs using LSF [Configure LSF]
- ○ Generate scripts only

□ Don't show this dialog again

[OK] [Cancel]

```
[BLK] Upgrading IPs...
WARNING: [Coretcl 2-1042] N
[BLK] Successfully upgraded
[EDT] Successfully edited prj and bd!
update_compile_order -fileset sources_1
regenerate_bd_layout
regenerate_bd_layout
regenerate_bd_layout
save_bd_design
```

Figure 24: Generate Bitstream

Generate BOOT.BIN

You can use the Vivado SDK or PetaLinux to generate the BOOT.BIN file. For boot image creation using the Vivado SDK, refer to the *Zynq UltraScale+ MPSoC Embedded Design Tutorial* (UG1209). For PetaLinux, use the *PetaLinux Tools Documentation Reference Guide* (UG1144).

References

These documents provide supplemental material useful with this product guide:

1. *DNNDK User Guide* (UG1327)
2. *Zynq UltraScale+ MPSoC Embedded Design Tutorial* (UG1209)
3. *PetaLinux Tools Documentation Reference Guide* (UG1144)

Please Read: Important Legal Notices

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

AUTOMOTIVE APPLICATIONS DISCLAIMER

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