

[Sign up](#)[Xilinx](#) / [Vitis-AI](#)[Code](#)[Issues](#) 122[Pull requests](#) 18[Actions](#)[Projects](#)[Security](#)[master](#) ▾[Vitis-AI](#) / [dsa](#) / [DPU-TRD](#) / [prj](#) / [Vivado](#) / [README.md](#)

hanxue Vitis-AI 1.3.1 Release (#318)

[History](#)

2 contributors



Raw

Blame



482 lines (325 sloc) 14.7 KB

# [Zynq UltraScale+ MPSoC DPU TRD V3.3](#) [Vivado 2020.2](#)

## [Table of Contents](#)

- [1 Revision History](#)
- [2 Overview](#)
- [3 Software Tools and System Requirements](#)
  - [3.1 Hardware](#)
  - [3.2 Software](#)
- [4 Design Files](#)
  - [Design Components](#)
- [5 Tutorials](#)
  - [5.1 Board Setup](#)
  - [5.2 Build and Run TRD Flow](#)
    - [5.2.1 Build the Hardware Design](#)
    - [5.2.2 Get Json File](#)

- [5.2.3 DPU PetaLinux BSP](#)
- [5.2.4 Run Resnet50 Example](#)
- [5.3 Configure the DPU](#)
  - [5.3.1 Modify the Frequency](#)
  - [5.3.2 Modify the Parameters](#)
- [6 Run with Vitis AI Library](#)
- [7 Known issues](#)

## 🔗 1 Revision History

---

Change Log:

V3.3 Change log:

- Supported range of conv stride from 4 to 8
- Supported Pool MaxReduce
- Supported Elew Multiply

V3.2 Change log:

- Updated IP Name
- Supported Bias-right-shift.
- Supported up-to 4 cores DPU

V3.1 Change Log:

- The first version of Vivado DPU TRD

---

## 🔗 2 Overview

---

This tutorial contains information about:

- How to set up the ZCU102 evaluation board and run the TRD.
- How to change the Configuration of DPU.

---

## 🔗 3 Software Tools and System Requirements

---

### 🔗 3.1 Hardware

Required:

- ZCU102 evaluation board
- Micro-USB cable, connected to laptop or desktop for the terminal emulator
- SD card

## 🔗 3.2 Software

Required:

- Vivado 2020.2 [Vivado Design Tools](#)
- [Vitis AI](#) to run models other than Resnet50, Optional

## 🔗 4 Design Files

### 🔗 Design Components

The top-level directory structure shows the the major design components. The TRD directory is provided with a basic README and legal notice file.

```
├── app
│   ├── img
│   └── model
├── dpu_ip
│   ├── DPUCZDX8G_v3_3_0
│   └── Vitis
├── prj
│   └── Vivado
│       ├── constrs
│       ├── doc
│       ├── dpu_petalinux_bsp
│       ├── pre-built
│       ├── scripts
│       └── base
```

## 🔗 5 Tutorials

### 🔗 5.1 Board Setup

🔗 Required:

- Connect power supply to 12V power connector.

- Connect micro-USB cable to the USB-UART connector, use the following settings for your terminal emulator:
  - Baud Rate: 115200
  - Data: 8 bit
  - Parity: None
  - Stop: 1 bit
  - Flow Control: None
- Insert SD card (FAT formatted).

#### Jumpers & Switches:

- Set boot mode to SD card:
  - Rev 1.0: SW6[4:1] - **OFF,OFF,OFF,ON**
  - Rev D2: SW6[4:1] - **ON,OFF,ON,OFF**

## 5.2 Build and Run TRD Flow

The following tutorials assume that the \$TRD\_HOME environment variable is set as given below.

```
%export TRD_HOME=<Vitis AI path>/DPU_TRD
```

 **Note:** It is recommended to follow the build steps in sequence.

### 5.2.1 Build the Hardware Design

We need install the Vivado Development Environment.

The following tutorials assume that the Vivado environment variable is set as given below.

Open a linux terminal. Set the linux as Bash mode.

```
% source <Vivado install path>/Vivado/2020.2/settings64.sh
```

The default settings of DPU is **B4096** with RAM\_USAGE\_LOW, CHANNEL\_AUGMENTATION\_ENABLE, DWCV\_ENABLE, POOL\_AVG\_ENABLE, RELU\_LEAKYRELU\_RELU6, Softmax.

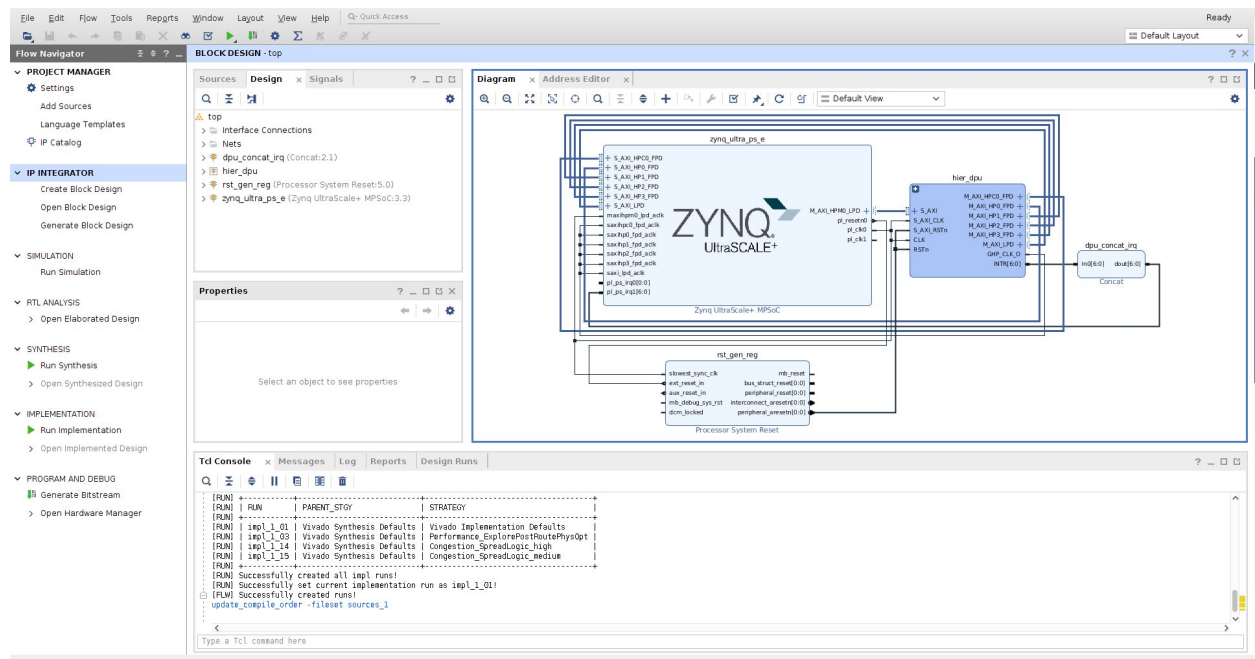
Modify the \$TRD\_HOME/prj/Vivado/scripts/trd\_prj.tcl file can change the default settings.

Build the hardware design.

```
% cd $TRD_HOME/prj/Vivado
```

```
% vivado -source scripts/trd_prj.tcl
```

After executing the script, the Vivado IPI block design comes up as shown in the below figure.

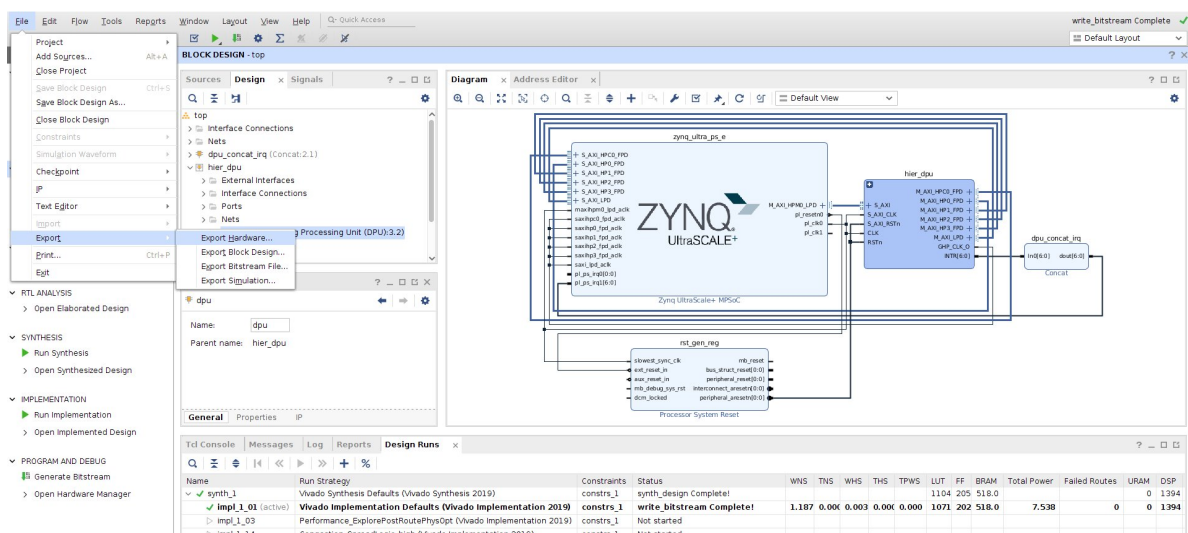


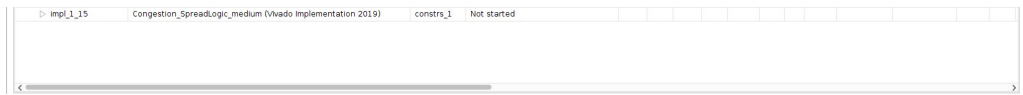
- Click on "Generate Bitstream".

Note: If the user gets any pop-up with "No implementation Results available". Click "Yes". Then, if any pop-up comes up with "Launch runs", Click "OK".

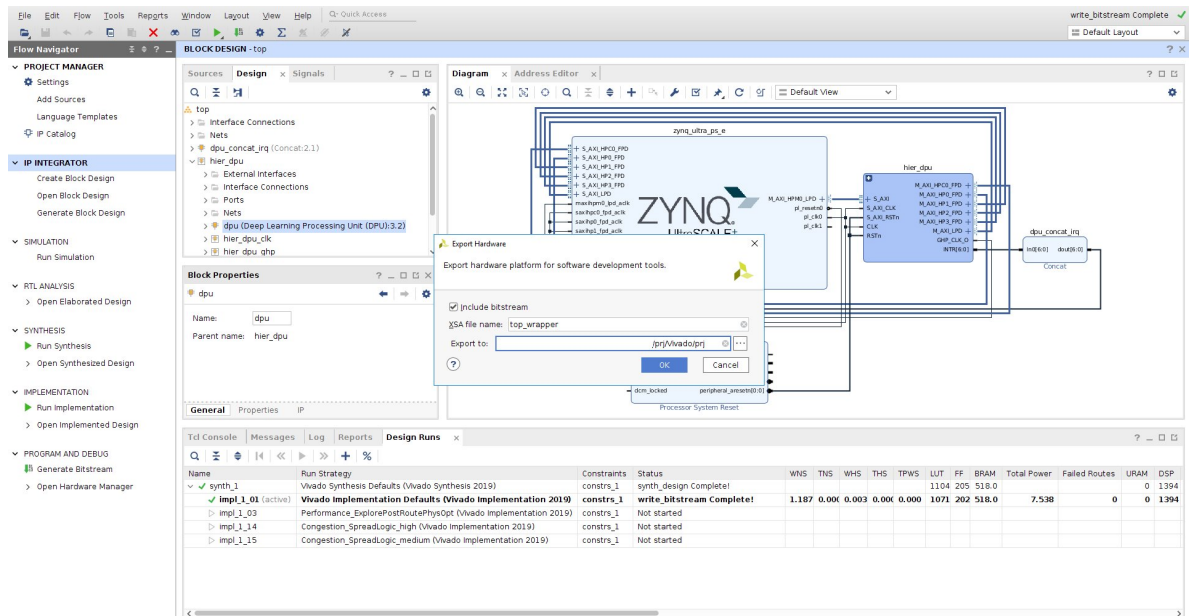
After the generation of bitstream completed.

- Go to File > Export > Export Hardware





- In the Export Hardware window select "Include bitstream" and click "OK".



The XSA file is created at \$TRD\_HOME/prj/Vivado/prj/top\_wrapper.xsa

⚡ Note: The actual results might graphically look different than the image shown

## ⚡ 5.2.2 Get Json File

Json file is an important file that needed by the VAI Compiler. The file has been created when compile by the Vivado tool. It works together with VAI Compiler to support model compilation under various DPU configurations.

The user can get the arch.json file in the following path.

\$TRD\_HOME/prj/Vivado/srcs/top/ip/top\_dpu\_0/arch.json

## ⚡ 5.2.3 DPU PetaLinux BSP

This tutorial shows how to build the Linux image and boot image using the PetaLinux build tool.

**PetaLinux Installation:** Refer to the [PetaLinux Tools Documentation](#) (UG1144) for installation.

```
% source <path/to/petalinux-installer>/settings.sh
% echo $PETALINUX
```

Post PetaLinux installation \$PETALINUX environment variable should be set.

## 🔗 Configure and build the PetaLinux project

```
% cd $TRD_HOME/prj/Vivado/dpu_petalinux_bsp
% ./download_bsp.sh
% petalinux-create -t project -s xilinx-zcu102-trd.bsp
% cd xilinx-zcu102-trd
% petalinux-config --get-hw-description=$TRD_HOME/prj/Vivado/prj/ --silentconfig
% petalinux-build
```

If the prebuilt design is wanted, please use the path for **--get-hw-description** as below.

```
% petalinux-config --get-hw-description=$TRD_HOME/prj/Vivado/pre-built/ --silentc
```

## 🔗 Create a boot image (BOOT.BIN) including FSBL, ATF, bitstream, and u-boot.

```
% cd images/linux
% petalinux-package --boot --fsbl zynqmp_fsbl.elf --u-boot u-boot.elf --pmufw pmu
```

## 🔗 5.2.4 Run Resnet50 Example

The TRD project has generated the matching model file in `$TRD_HOME/app` path as the default settings. If the user change the DPU settings. The model need to be created again.

This part is about how to run the Resnet50 example from the source code.

The user must create the SD card. Refer section "Configuring SD Card ext File System Boot" in page 65 of [ug1144](#) for PetaLinux 2020.2:

Copy the image.ub, system.dtb, boot.scr, and BOOT.BIN files in `$TRD_HOME/prj/Vivado/dpu_petalinux_bsp/xilinx-zcu102-trd/images/linux` to BOOT partition.

Extract the rootfs.ext4 file in `TRD_HOME/prj/Vivado/dpu_petalinux_bsp/xilinx-zcu102-trd/images/linux` to RootFs partition using the `dd` command. For example:

```
sudo dd if=rootfs.ext4 of=/dev/mmcblk0p2 bs=1M conv=fsync
```

Copy the folder `$TRD_HOME/app/` to RootFs partition

Insert the SD card into the FPGA and boot it in SD mode. After the Linux boot, run the

following commands in the RootFs partition:

```
% cd ./app

% cp ./model/resnet50.xmodel .

% env LD_LIBRARY_PATH=samples/lib samples/bin/resnet50 img/bellpeppe-994958.JPEG
```

Expect:

```
score[945] = 0.992235    text: bell pepper,
score[941] = 0.00315807  text: acorn squash,
score[943] = 0.00191546  text: cucumber, cuke,
score[939] = 0.000904801 text: zucchini, courgette,
score[949] = 0.00054879  text: strawberry,
```

 **Note:** The resenet50 test case can support both Vitis and Vivado flow. If you want to run other network. Please refer to the [Vitis AI Github](#) and [Vitis AI User Guide](#).

## 5.3 Configure the DPU

The DPU IP provides some user-configurable parameters to optimize resource utilization and customize different features. Different configurations can be selected for DSP slices, LUT, block RAM(BRAM), and UltraRAM utilization based on the amount of available programmable logic resources. There are also options for addition functions, such as channel augmentation, average pooling, depthwise convolution.

The TRD also support the softmax function.

For more details about the DPU, please read [DPU IP Product Guide](#)

### 5.3.1 Modify the Frequency

Modify the scripts/trd\_prj.tcl to modify the frequency of m\_axi\_dpu\_aclk. The frequency of dpu\_2x\_clk is twice of m\_axi\_dpu\_aclk.

```
dict set dict_prj dict_param DPU_CLK_MHz {325}
```

### 5.3.2 Modify the parameters

Modify the scripts/trd\_prj.tcl to modify the parameters which can also be modified on the GUI.

The TRD supports to modify the following parameters.



- DPU\_NUM
- DPU\_ARCH
- DPU\_RAM\_USAGE
- DPU\_CHN\_AUG\_ENA
- DPU\_DWCV\_ENA
- DPU\_AVG\_POOL\_ENA
- DPU\_CONV\_RELU\_TYPE
- DPU\_SFM\_NUM
- DPU\_DSP48\_USAGE
- DPU\_URAM\_PER\_DPU
- DPU\_CONV\_WP

### [DPU\\_NUM](#)

The DPU core number is set 2 as default setting.

```
dict set dict_prj dict_param DPU_NUM {2}
```

A maximum of 4 cores can be selected on DPU IP.

[Note:](#) The DPU needs lots of LUTs and RAMs. Use 3 or more DPU may cause the resource and timing issue.

### [DPU\\_ARCH](#)

Arch of DPU: The DPU IP can be configured with various convolution architectures which are related to the parallelism of the convolution unit. The architectures for the DPU IP include B512, B800, B1024, B1152, B1600, B2304, B3136, and B4096.

```
dict set dict_prj dict_param DPU_ARCH {4096}
```

[Note:](#) It relates to models. If change, must update models.

### [DPU\\_RAM\\_USAGE](#)

RAM Usage: The RAM Usage option determines the total amount of on-chip memory used in different DPU architectures, and the setting is for all the DPU cores in the DPU IP. High RAM Usage means that the on-chip memory block will be larger, allowing the DPU more flexibility to handle the intermediate data. High RAM Usage implies higher performance in each DPU core.

Low

```
dict set dict_prj dict_param DPU_RAM_USAGE {low}
```

High

```
dict set dict_prj dict_param DPU_RAM_USAGE {high}
```

## DPU\_CHN\_AUG\_ENA

Channel Augmentation: Channel augmentation is an optional feature for improving the efficiency of the DPU when handling input channels much lower than the available channel parallelism.

Enable

```
dict set dict_prj dict_param DPU_CHN_AUG_ENA {1}
```

Disable

```
dict set dict_prj dict_param DPU_CHN_AUG_ENA {0}
```

 Note: It relates to models. If change, must update models.

## DPU\_DWCV\_ENA

Depthwise Convolution: The option determines whether the Depthwise convolution operation will be performed on the DPU or not.

Enable

```
dict set dict_prj dict_param DPU_DWCV_ENA {1}
```

Disable

```
dict set dict_prj dict_param DPU_DWCV_ENA {0}
```

 Note: It relates to models. If change, must update models.

## DPU\_ELEW\_MULT\_ENA

ElementWise Multiply: The ElementWise Multiply can perform dot multiplication on


most two input feature maps. The input channel of EM(ElementWise Multiply) ranges from 1 to  $256 * \text{channel\_parallel}$ .

Enable

```
dict set dict_prj dict_param DPU_ELEW_MULT_ENA {1}
```

Disable

```
dict set dict_prj dict_param DPU_ELEW_MULT_ENA {0}
```

 Note: It relates to models. If change, must update models.

### DPU\_AVG\_POOL\_ENA

AveragePool: The option determines whether the average pooling operation will be performed on the DPU or not.

Enable

```
dict set dict_prj dict_param DPU_AVG_POOL_ENA {1}
```

Disable

```
dict set dict_prj dict_param DPU_AVG_POOL_ENA {0}
```

 Note: It relates to models. If change, must update models.

### DPU\_CONV\_RELU\_TYPE

The ReLU Type option determines which kind of ReLU function can be used in the DPU. ReLU and ReLU6 are supported by default.

RELU\_RELU6

```
dict set dict_prj dict_param DPU_CONV_RELU_TYPE {2}
```

RELU\_LEAKRELU\_RELU6

```
dict set dict_prj dict_param DPU_CONV_RELU_TYPE {3}
```

 Note: It relates to models. If change, must update models.

## DPU\_SFM\_NUM

Softmax: This option allows the softmax function to be implemented in hardware.

Only use the DPU

```
dict set dict_prj dict_param DPU_SFM_NUM {0}
```

Use the DPU and Softmax

```
dict set dict_prj dict_param DPU_SFM_NUM {1}
```

## DPU\_DSP48\_USAGE

DSP Usage: This allows you to select whether DSP48E slices will be used for accumulation in the DPU convolution module.

High

```
dict set dict_prj dict_param DPU_DSP48_USAGE {high}
```

Low

```
dict set dict_prj dict_param DPU_DSP48_USAGE {low}
```

## DPU\_URAM\_PER\_DPU

The DPU uses block RAM as the memory unit by default. For a target device with both block RAM and UltraRAM, configure the number of UltraRAM to determine how many UltraRAMs are used to replace some block RAMs. The number of UltraRAM should be set as a multiple of the number of UltraRAM required for a memory unit in the DPU. An example of block RAM and UltraRAM utilization is shown in the Summary tab section.

```
dict set dict_prj dict_param DPU_URAM_PER_DPU {0}
```

## DPU\_CONV\_WP

The DPU supports additional write-parallel acceleration for the **PointPillar models**. Modify line 49 of scripts/base/trd\_bd.tcl file can change the default settings.

The option could be set as power of 2 and up to PP/2.

For B512/B800/B1152 architectures, DPU\_CONV\_WP can be set up to 2:

```
dict set dict_prj dict_param DPU_CONV_WP {2}
```

For B1024/B1600/B2304/B3136/B4096 architectures, DPU\_CONV\_WP can be set up to 4:

```
dict set dict_prj dict_param DPU_CONV_WP {4}
```

## 🔗 6 Run with Vitis AI Library

---

For the introduction of Vitis AI Library, please refer to **Quick Start For Edge** of this page [Vitis AI Library](#)

## 🔗 7 Known issues

---

### 1.DDR QOS

When AXI HP0 port connects to DPU and use DisplayPort to display, if the QoS settings are not modified, the DisplayPort transmission may under-run, producing black frames or screen flicker intermittently during DPU running. Apart from QoS settings, increasing the read and write issuing capability (outstanding commands) of DPU connected AXI FIFO interface S\_AXI\_HPC{0, 1}\_FPD or S\_AXI\_HP{0:3}\_FPD or S\_AXI\_LPD may keep the ports busy with always some requests in the queue, which could improve DPU performance highly. [solution](#)

### 🔗 Solution

User could execute **zynqmp\_dpu\_optimize.sh** on target board to address the issue.

Copy **\$TRD\_HOME/app/dpu\_sw\_optimize.tar.gz** to target board, after linux boot-up, run:

```
% tar -zxvf dpu_sw_optimize.tar.gz

% ./dpu_sw_optimize/zynqmp/zynqmp_dpu_optimize.sh

(refer to dpu_sw_optimize/zynqmp/README.md get more info)
```

