EMBEDEEP
壹 普 智 慧

EEP-TPU FPGA IP 评估版
使用手册

eep-ug007 (v0.1.0)

2022-11-14

厦门壹普智慧科技有限公司

修订历史：

版本	日期	描述	作者
0.1.0	2022-11-14	初始版本	周 xx

目　　录

1、概述.....4
2、IP 工程文件包的说明.....5
3、Vivado 工程搭建.....6
3.1 linux 系统.....6
3.2 Windows 系统.....7
4、TPU IP 裸机测试.....8
4.1 硬件连接.....8
4.2 vitis 工程搭建.....8
4.3 TPU 测试结果演示.....11
5、嵌入式 linux 使用说明.....16
6、附录.....17
6.1 神经网络模型的编译与转换.....17
6.2 FPGA 资源使用情况.....18

1、概述

EEP TPU 是由**厦门壹普智慧科技有限公司**（EMBEDEEP）自主研发的人工智能张量处理器。EEP-TPU 张量处理器拥有众多创新特性，采用数据流计算架构，可为神经网络推理提供高效率、高性能、高灵活性的数据计算；可提供 FP16 和 INT8 精度的计算，支持混合精度计算模式；可支持 SoC 模式或 Standalone 模式，对主处理器性能及操作系统环境无依赖，可适配多种场景下的多种系统；配合自主研发的异构计算软件系统与编译优化开发环境，可支持多种主流深度学习框架，包括 CAFFE、Darknet、Pytorch、ONNX、NCNN 等；支持分类、检测、分割等多种图像算法，包括 ResNet、Inception、MobileNet、SqueezeNet、YOLOV3、YOLOV4、YOLOV5、SSD、ICNet 等。

为了方便用户对 EEP TPU IP 有更直接的性能评估及使用，我们为免费提供了基于 Xilinx zynqMPSoC FPGA 的评估版 IP（TPU 可连续工作 1~2 小时）。用户可以使用评估版 IP 基于提供的设计 demo，快速搭建带有 EEP TPU 的 FPGA 工程，并结合相应的裸机程序，直观的体验如何在神经网络推理应用中使用 EEP TPU。

2、IP 工程文件包的说明

在评估版 IP 的工程文件包中，我们提供了基于 ZU15EG FPGA 开发板的 Vivado 工程。包括带有 EEP TPU IP Vivado 工程的脚本文件、FPGA 管脚约束文件、评估版 IP 以及与 IP 匹配的编译器。该工程可用于运行测试 TPU 功能的裸机源码程序。更为详细的目录介绍，可以参见工程目录中的 excel 文档——“目录说明.xlsx”。

注：工程目录是基于米联客 MZU15A FPGA 开发板开发的，用户如果改用其他 ZynqMP SoC FPGA，需要修改对应的 FPGA 管脚约束和 zynqMP IP 设置，EEP TPU IP 的调用及连接方法是一致的。

3、Vivado 工程搭建

图 3.1 所示为 Vivado IPI 工具中搭建的用于 TPU IP 测试的 SoC 系统框图。系统包含：

（1）DVP 信号接收 IP（EEP_DVP_Top），该 DVP IP 可以采集 DVP 摄像头数据并存放于 DDR；

（2）TPU IP（EEP_TPU），该 TPU IP 可进行神经网络推理计算；

（3）zynqMP，用于系统程序运行等工作。

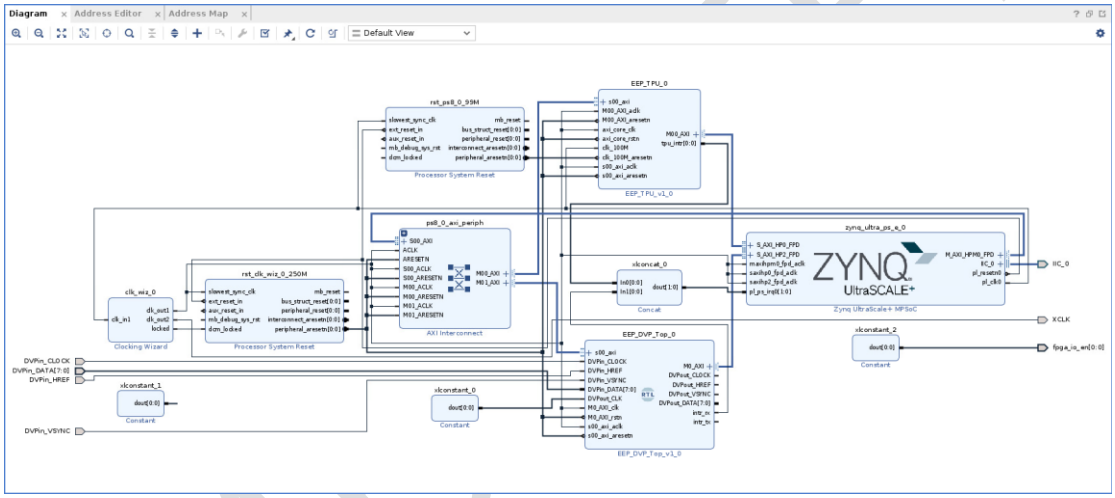


图 3.1

3.1 linux 系统

在 linux 系统下，用户可以打开终端，进入 IP 工程目录。在 script 目录中，修改 create_prj.sh 中[Vivado install path]vivado 的安装路径，然后执行命令：

```
$ ./create_prj.sh system_rtl_MZU15A_TPU_IP_DVP_DP_v202101.tcl
```

脚本会自动生成 vivado 工程，进行综合实现，生成 bit 及相关的 xsa 文件。

3.2 Windows 系统

在 Windows 系统下，打开 Vivado 工具，在 tcl console 中执行命令：

source system_rtl_MZU15A_TPU_IP_DVP_DP_v202101.tcl

注：在 tcl 文件开头，有变量 PRJNAME、PRJPATH、IP_PATH、constr_PATH，需要根据实际情况修改相应的路径名称。

执行上述命令后，同样也会生成 vivado 工程及相关的 bit 文件等。

4、TPU IP 裸机测试

4.1 硬件连接

基于米联客 MZU15A 开发套件，硬件连接如下图 4.1 所示。

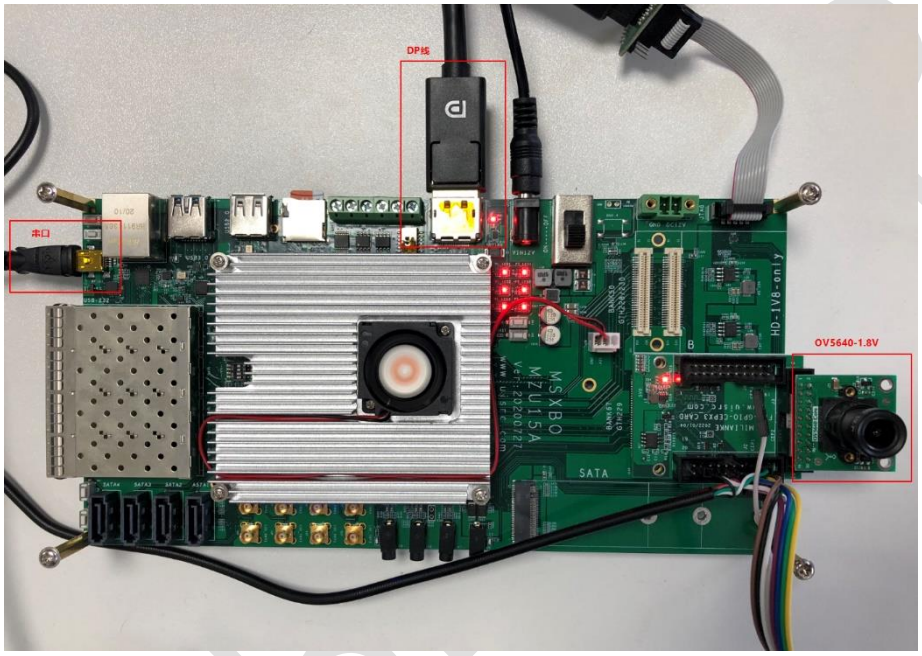


图 4.1 硬件连接

4.2 vitis 工程搭建

打开 Xilinx Vitis 2021.1，选择 xsa 文件所在的文件夹作为 workspace。

注：xsa 文件位于 hardware/xsa 目录。

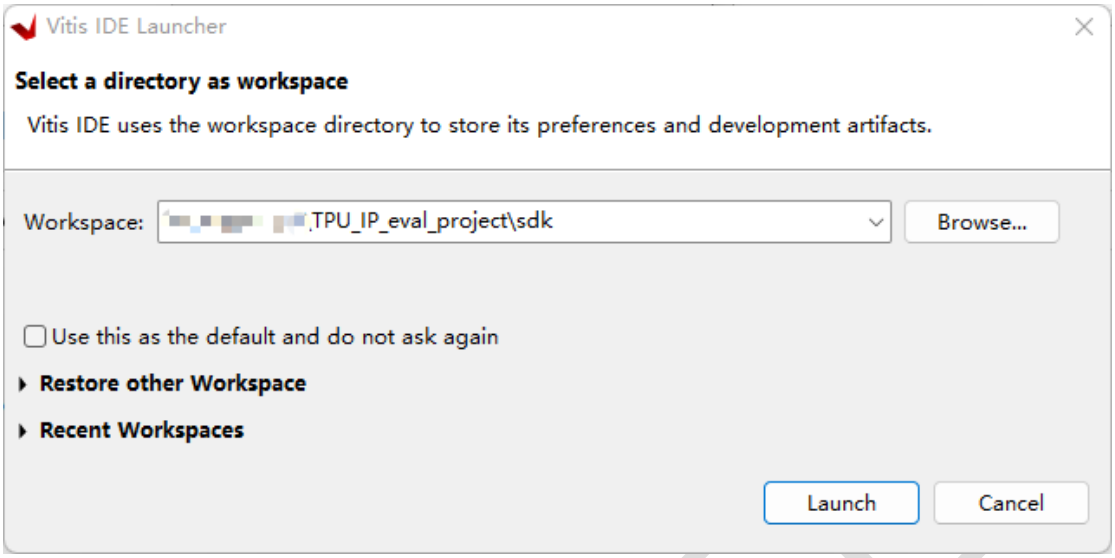


图 4.2 搭建流程图 1

File->New->Application Project…->Next，选择 xsa 文件。

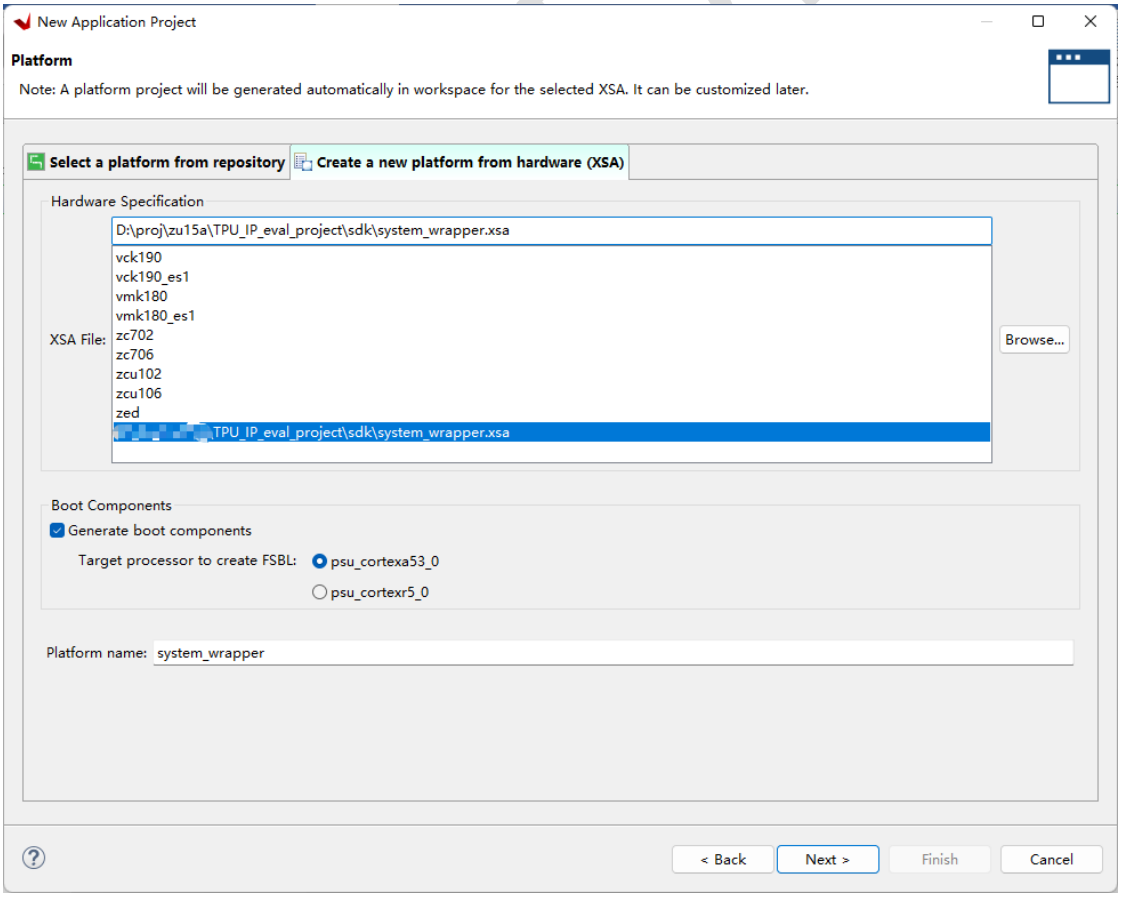


图 4.3 搭建流程图 2

输入工程名，Next->Next，创建一个空的 C++工程，完成工程搭建。

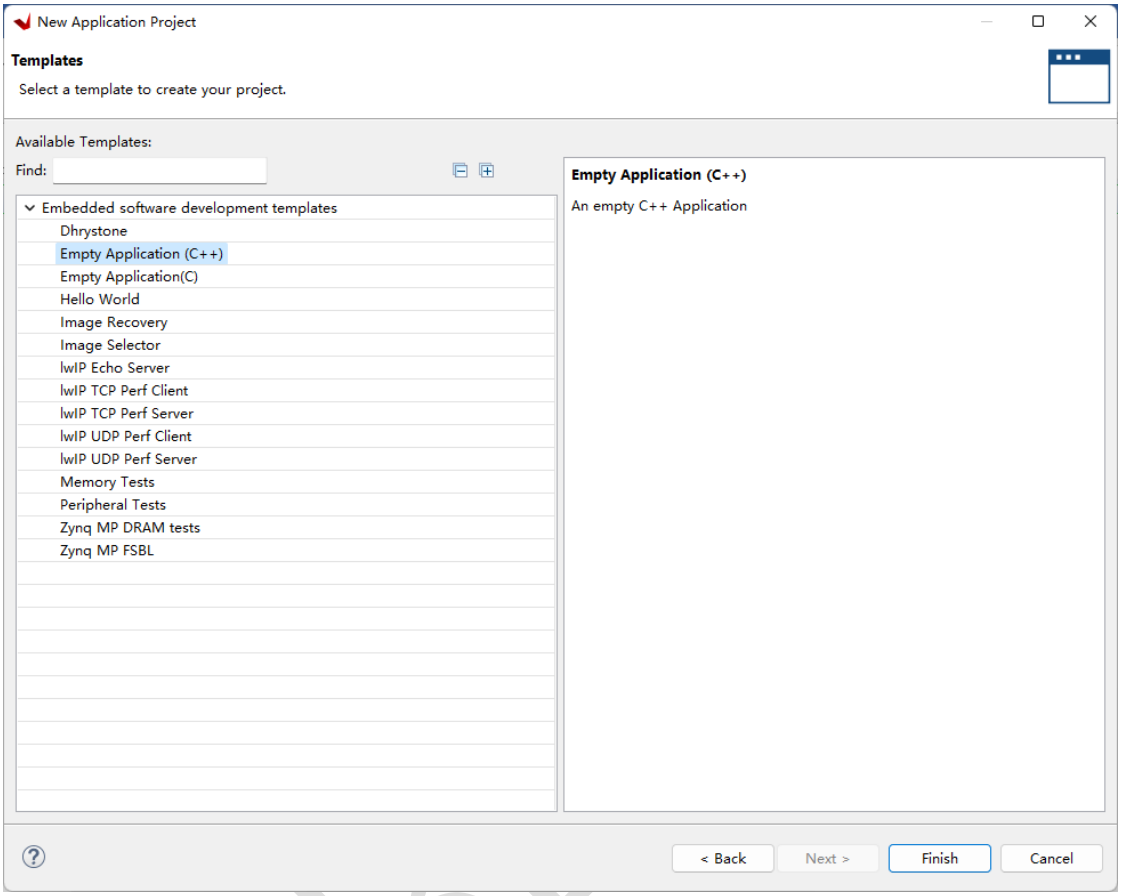


图 4.4 搭建流程图 3

（1）增加 xilffs 文件系统支持

选择 platform.sqr 修改 BSP 的设置，增加 xilffs 文件系统支持，然后重新编译 system_wrapper。

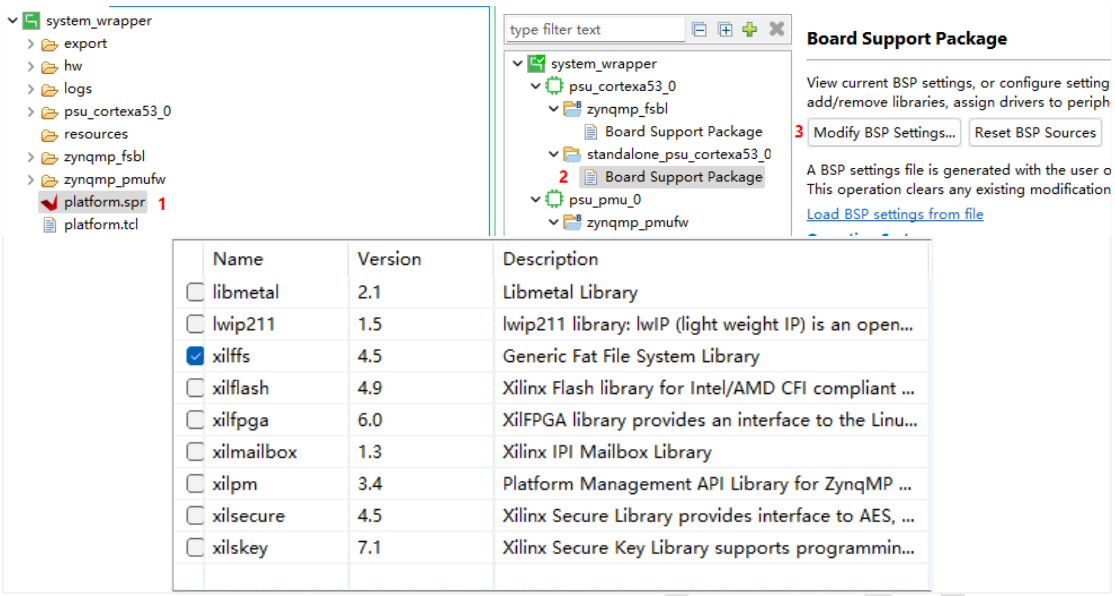


图 4.5 xilffs 配置

（2）设置堆大小

通过右键点击“Generate Linker Script”按钮设置堆大小，设置值为 10000 KB，实际大小可根据应用程序做调整。

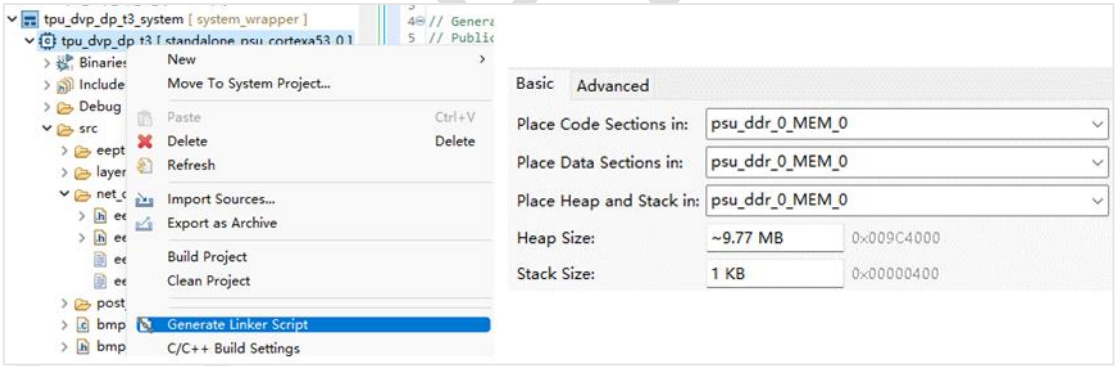


图 4.6 Heap 设置

将我们提供的 sdk/standalone/src 文件夹中的文件拷贝到搭建的 sdk 工程 src 文件夹中，即完成工程的搭建。

4.3 TPU 测试结果演示

程序的加载可以通过 JTAG 下载或者用我们预编译好的 BOOT.bin 写入 SD 卡，通过 SD 卡启动加载 FPGA。

（1）JTAG 下载

将 eepnet.mem 和 eepinput.mem 文件复制到 SD 卡，启动模式设置成 JTAG，即模式开关设置成“ON ON ON ON”。在 Xilinx Vitis 中右键工程->Debug As->Launch Hardware(Single Application Debug)，完成 bit 和 elf 文件的下载，在 Debug 界面 run 程序。

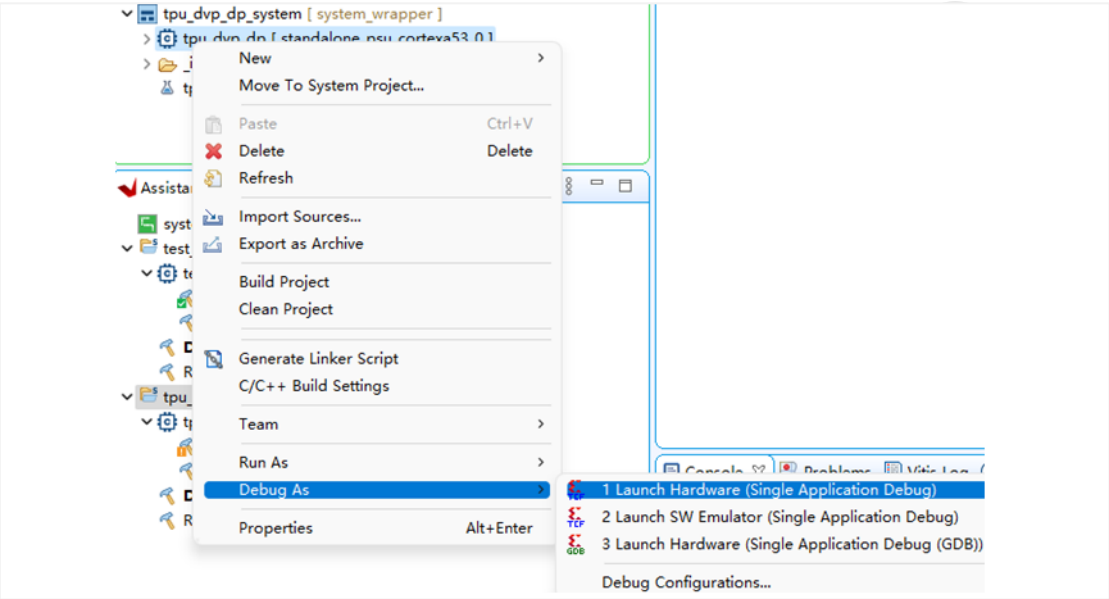


图 4.7 程序运行

（2）SD 卡启动

将 demo/boot 文件夹中的 BOOT.bin、eepnet.mem 和 eepinput.mem 文件复制到 SD 卡，启动模式设置成 SD 卡启动，即模式开关设置成“ON ON OFF OFF”。

开发板上电后，可以从串口终端中看到如下打印信息：

```
Enter the Number:Xilinx Zynq MP First Stage Boot Loader
Release 2021.1   Nov 18 2022  -  16:24:22
PMU-FW is not running, certain applications may not be supported.
Loading program ......
HPD event .......... ! Connected.
Lane count =    2
Link rate =     10

Starting Training...
          ! Training succeeded.
DONE!
.......... HPD event
output cnt = 2
out[0]: hwaddr 0x320f44c0, shape: 1,255,13,13
out[1]: hwaddr 0x321096c0, shape: 1,255,26,26
in: hwaddr 0x31bac4c0, shape: 1,3,416,416
mem_base = 0x31000000
tpureg_addr = 0xa0000000
hwbase0 = 0x31000000
hwbase1 = 0x31bac4c0
hwbase2 = 0x3215dec0
hwbase3 = 0x320f44c0
memsize = 0x01bcde00
addr_alg = 0x31b8c400
TPU hardware version: 0x00810b32 (0.8.4-r11-p50)
Loading data
Load Net Data From SD Card...
Load Net Done
################################################
Choose Feature to Test:
1: Get 1 Frame
2: Forward Result
3: Save Image to SD Card
4: Read Test Image
5: Run Demo
0: Exit
Enter the Number:
```

图 4.8 初始菜单

此时，DP 显示器显示图 4.9 中的初始界面，表示 DP 显示器可以正常显示。

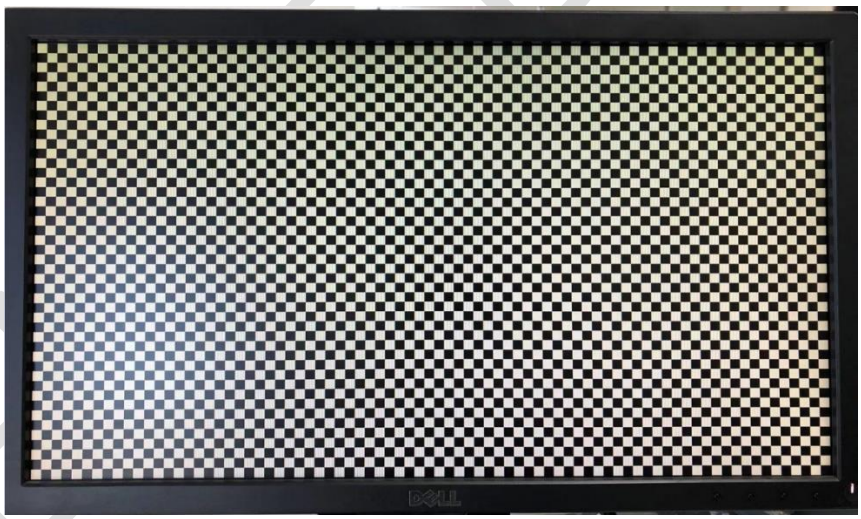


图 4.9 初始显示图像

测试菜单说明：

（1）通过 DVP 采集一帧图像并显示；

（2）推理采集的图像；

（3）保存图像数据到 SD 卡；

（4）读取测试图片数据（数据保存在 SD 卡，用于测试推理是否正确）；

（5）连续 DVP 采集图像并推理。

测试项：

（1）在串口菜单中先输入“4”从 SD 卡加载测试图像数据，再输入“2”推理测试图像，可以看到图 4.10 所示串口打印结果：

```
Choose Feature to Test:
1: Get 1 Frame
2: Forward Result
3: Save Image to SD Card
4: Read Test Image
5: Run Demo
0: Exit
Enter the Number:4
################################################
Choose Feature to Test:
1: Get 1 Frame
2: Forward Result
3: Save Image to SD Card
4: Read Test Image
5: Run Demo
0: Exit
Enter the Number:2
Forwarding...
read output[0]: addr 0x320f44c0, shape: 255,13,13
epmat_size = 0x15200(86528)
read output[1]: addr 0x321096c0, shape: 255,26,26
epmat_size = 0x54800(346112)
forward time is 38679 us
detection output time is 62124 us
Obj[0]:  8(     truck)  0.910327; At (0.007392,0.276477)  0.251595 x 0.515154
Obj[1]:  1(    person)  0.846338; At (0.857661,0.324374)  0.899836 x 0.472777
Obj[2]: 18(     horse)  0.821468; At (0.443775,0.124901)  0.829803 x 0.885698
Obj[3]: 17(       dog)  0.600459; At (0.283588,0.547477)  0.388209 x 0.892048
Obj[4]:  1(    person)  0.325858; At (0.487803,0.033946)  0.695681 x 0.658211
```

图 4.10 测试图片推理

（2）在串口菜单中先输入 1 通过 DVP 采集一帧图像，再输入 2 推理采集的图像并打印结果。

```
################################################
Choose Feature to Test:
1: Get 1 Frame
2: Forward Result
3: Save Image to SD Card
4: Read Test Image
5: Run Demo
0: Exit
Enter the Number:1
Image Capture Done
################################################
Choose Feature to Test:
1: Get 1 Frame
2: Forward Result
3: Save Image to SD Card
4: Read Test Image
5: Run Demo
0: Exit
Enter the Number:2
Forwarding...
read output[0]: addr 0x320f44c0, shape: 255,13,13
epmat_size = 0x15200(86528)
read output[1]: addr 0x321096c0, shape: 255,26,26
epmat_size = 0x54800(346112)
forward time is 38684 us
detection output time is 62145 us
Obj[0]:  1(    person)  0.808453; At (0.007016,0.121290)  0.850179 x 1.035409
```

图 4.11 实际采图推理

（3）在串口菜单中输入 5，使 DVP 连续采集图像并推理。

5、嵌入式 linux 使用说明

已经编译好的 BOOT.BIN 及 image.ub 文件存放在文件夹 hardware/BOOTbin 中，通过该文件可以启动嵌入式 linux 系统。在 linux 系统中，用户可以对 TPU 进行相应的测试（注：该 TPU 为评估版本，故每隔一小时需要重新启动一下系统）。在 linux 系统下，TPU 的使用说明请参考 doc 目录下 eep-ug050 和 eep-ug053 文档，其对应的 demo 程序存放于 sdk 目录中。

6、附录

6.1 神经网络模型的编译与转换

参考编译器的使用文档，将神经网络模型编译成 TPU 可执行的 bin 文件。再通过 eepBincvt 工具，将 bin 文件转换成相关的 C 语言头文件及模型二进制文件。这些头文件和二进制模型文件是裸机测试所需的网络模型及相关配置信息。

（1）神经网络模型的编译

运行 sdk/standalone/net_model/scripts/b_yolo4tiny.sh 脚本，在 sdk/standalone/net_model/binRoot/yolov4tiny 文件夹中生成 eeptpu_s2.pub.bin.

（2）神经网络模型的转换

Bin 文件转换的目的是将编译器生成的 bin 文件转换成适用于裸机程序运行的文件格式。eepBinCvt 运行环境为 Ubuntu 18.04。

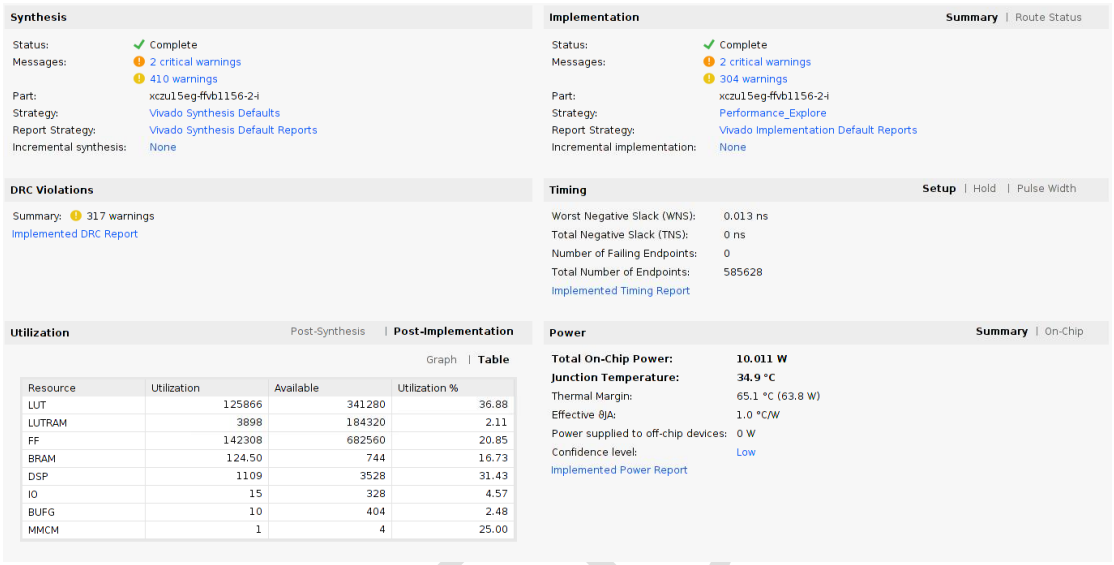
运行 sdk/standalone/net_model/scripts/cvt.sh 脚本，会在当前目录生成 eepinput.h, eepinput.mem, eepnet.h, eepnet.mem 四个文件。将.h 文件复制到 src/net_data 文件夹中，将.mem 文件复制到 SD 卡，应用程序从 SD 卡中加载神经网络网络模型数据到 DDR 存储器。

```
(base) [redacted]NNmodel/eepBinCvt_v2.1.0# ./cvt.sh
./eepBinCvt --bin ../scripts/binRoot/yolov4tiny/eeptpu_s2.pub.bin --input ../models/images/ssd/004545.bmp --outp
ut header

[ eepBinCvt v2.1.0 ]

Bin file: ../scripts/binRoot/yolov4tiny/eeptpu_s2.pub.bin
Input file: ../models/images/ssd/004545.bmp
Base address: 0x00000000
Output type: header
Generating header file...
Bin file is 'public-bin'.
Bin use compiler v2.3.13
Write net data to: ./eepnet.h
Write input data to: ./eepinput.h
Generate ok: cost 834.332 ms

./eepBinCvt --bin ../scripts/binRoot/yolov4tiny/eeptpu_s2.pub.bin --input ../models/images/ssd/004545.bmp --outp
ut mem

[ eepBinCvt v2.1.0 ]

Bin file: ../scripts/binRoot/yolov4tiny/eeptpu_s2.pub.bin
Input file: ../models/images/ssd/004545.bmp
Base address: 0x00000000
Output type: mem
Generating mem file...
Bin file is 'public-bin'.
Bin use compiler v2.3.13
Write ini to: ./eepnet.ini
Write net data to: ./eepnet.mem
Write input data to: ./eepinput.mem
Generate ok: cost 28.615 ms
```

图 6.1 模型编译示例

6.2 FPGA 资源使用情况

在 FPGA 器件 xczu15eg-ffvb1156-2-i 中实现 TPU v3+设计，FPGA 资源占用情况如图 6.2 所示：



Synthesis

Status:	Complete
Messages:	2 critical warnings
	410 warnings
Part:	xczu15eg-ffvb1156-2-i
Strategy:	Vivado Synthesis Defaults
Report Strategy:	Vivado Synthesis Default Reports
Incremental synthesis:	None

Implementation Summary | Route Status

Status:	Complete
Messages:	2 critical warnings
	304 warnings
Part:	xczu15eg-ffvb1156-2-i
Strategy:	Performance_Explore
Report Strategy:	Vivado Implementation Default Reports
Incremental implementation:	None

DRC Violations

Summary: 317 warnings

Implemented DRC Report

Timing Setup | Hold | Pulse Width

Worst Negative Slack (WNS):	0.013 ns
Total Negative Slack (TNS):	0 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	585628

Implemented Timing Report

Utilization Post-Synthesis | Post-Implementation

Graph | Table

Resource	Utilization	Available	Utilization %
LUT	125866	341280	36.88
LUTRAM	3898	184320	2.11
FF	142308	682560	20.85
BRAM	124.50	744	16.73
DSP	1109	3528	31.43
IO	15	328	4.57
BUFG	10	404	2.48
MMCM	1	4	25.00

Power Summary | On-Chip

Total On-Chip Power:	**10.011 W**
Junction Temperature:	**34.9 °C**
Thermal Margin:	65.1 °C (63.8 W)
Effective θJA:	1.0 °C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Implemented Power Report

图 6.2 FPGA 资源占用情况