本周汇报

1.9-1.12

◎ 李宏毅机器学习

EP1 EP2

基础概念：

……

软件安装与开发环境搭建：

Anaconda

Jupyter Notebook

Pytorch

Tensorflow

Keras

…

Mnist.py

D: > Mnist > Mnist.py > ...

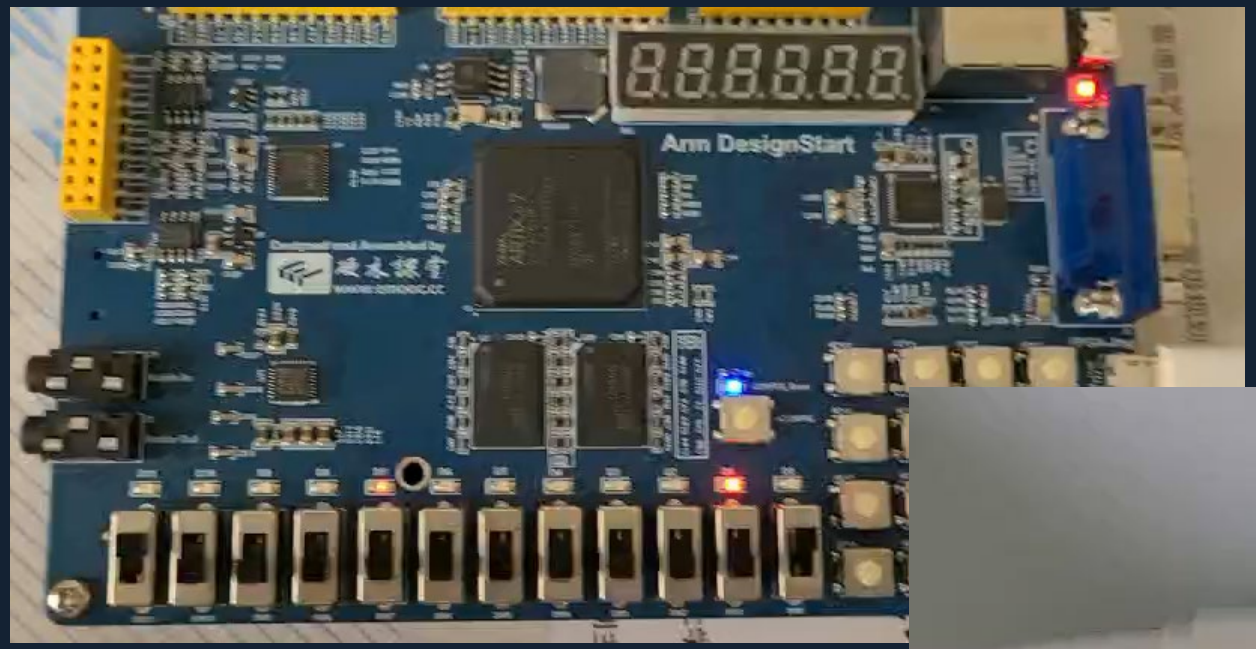
```
        network.add(layers.Dense(10, activation='softmax'))
        return network
network = LeNet()
network.compile(optimizer=RMSprop(lr=0.001), loss='categorical_crossentropy',
#print(network.summary())

#将数据铺开成一维向量 把二维矩阵28*28矩阵压缩成1*724像素值的向量
train_images = train_images.reshape((60000, 28, 28, 1)).astype('float') / 255
test_images = test_images.reshape((10000, 28, 28, 1)).astype('float') / 255
train_labels = to_categorical(train_labels) #这是一个十分类问题，最后只有是个神
test_labels = to_categorical(test_labels)

# 训练网络，用fit函数，epochs表示训练多少个回合， batch_size表示每次训练给多大的
network.fit(train_images, train_labels, epochs=10, batch_size=128, verbose=2)
```

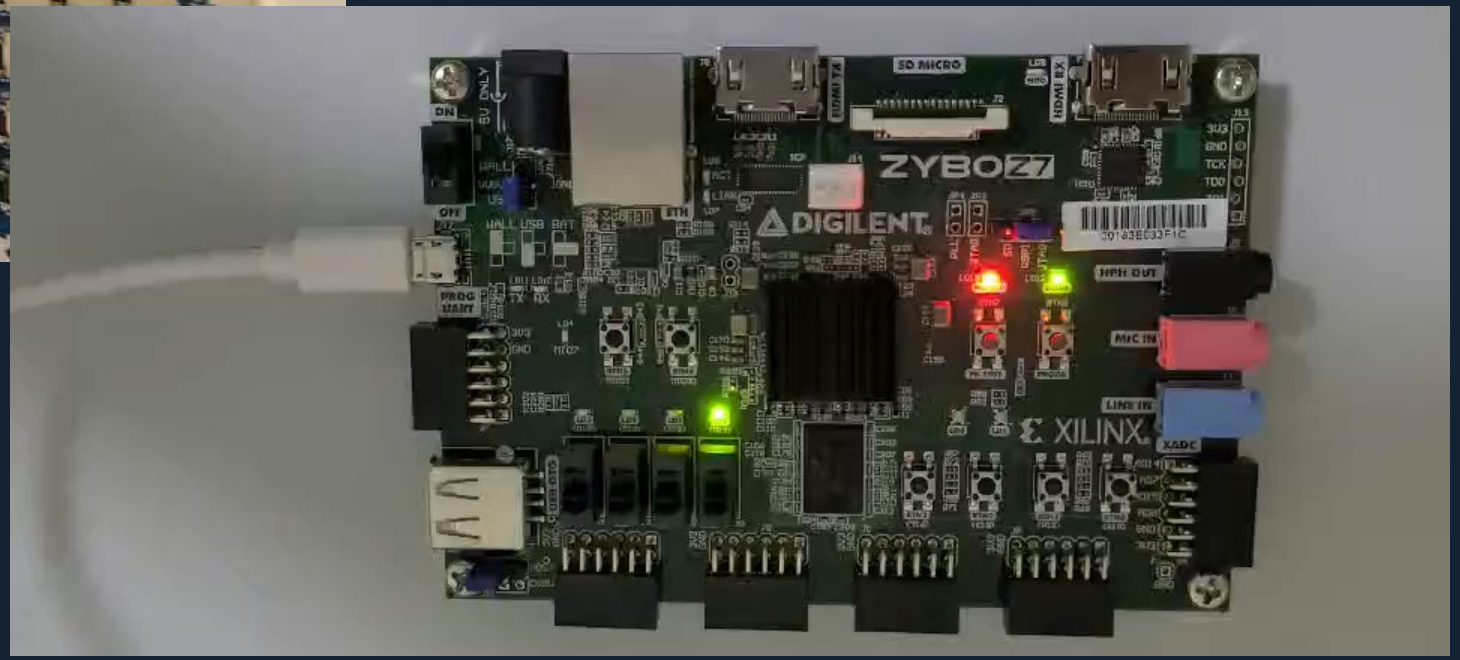
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
2021-01-12 09:43:23.355461: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:11
Epoch 1/10
469/469 - 6s - loss: 0.3317 - accuracy: 0.9004
Epoch 2/10
469/469 - 5s - loss: 0.0923 - accuracy: 0.9715
Epoch 3/10
469/469 - 5s - loss: 0.0597 - accuracy: 0.9819
Epoch 4/10
469/469 - 5s - loss: 0.0467 - accuracy: 0.9859
Epoch 5/10
469/469 - 5s - loss: 0.0371 - accuracy: 0.9888
Epoch 6/10
469/469 - 5s - loss: 0.0311 - accuracy: 0.9904
Epoch 7/10
469/469 - 5s - loss: 0.0258 - accuracy: 0.9915
Epoch 8/10
469/469 - 5s - loss: 0.0225 - accuracy: 0.9930
Epoch 9/10
469/469 - 5s - loss: 0.0195 - accuracy: 0.9939
Epoch 10/10
469/469 - 5s - loss: 0.0168 - accuracy: 0.9945
313/313 [==============================] - 1s 2ms/step - loss: 0.0319 - accuracy: 0.9902
test_loss: 0.03188302740454674     test_accuracy: 0.9901999831199646
```

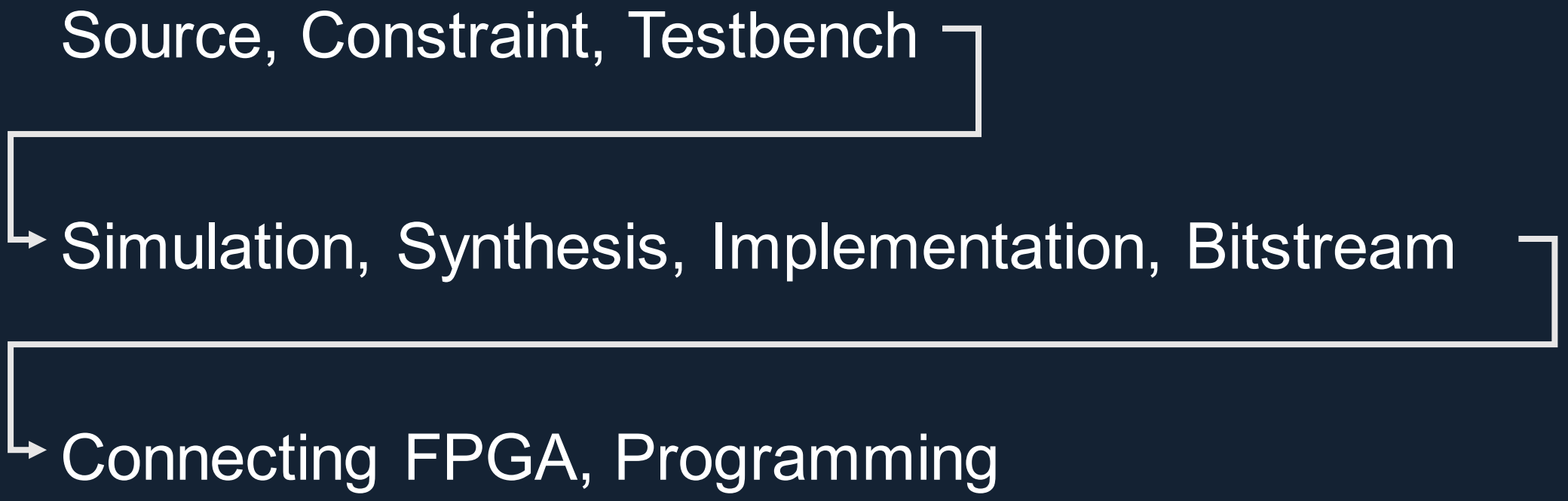
开发板

两个简单实验，用来熟悉流程





◎ Vivado 基础操作

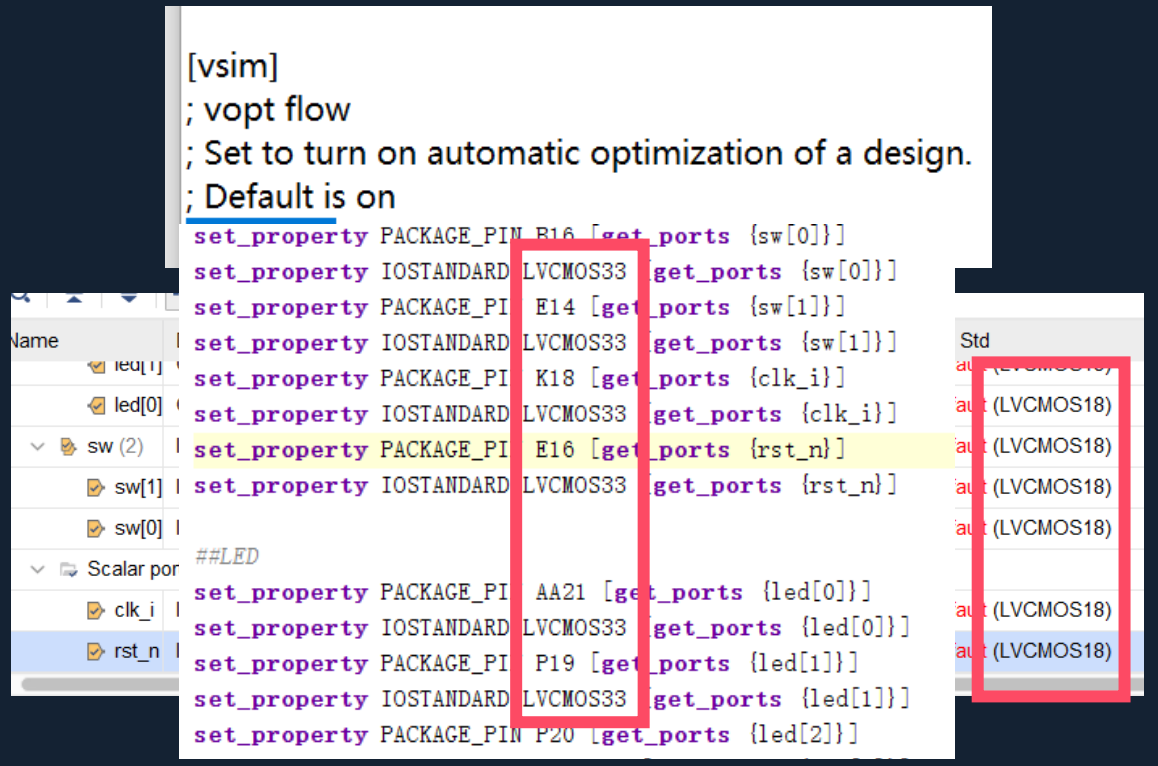

Source, Constraint, Testbench
Simulation, Synthesis, Implementation, Bitstream
Connecting FPGA, Programming

◎ Vivado 问题解决

与Modelsim联合仿真时卡住

无法产生比特流

CPU利用率过低



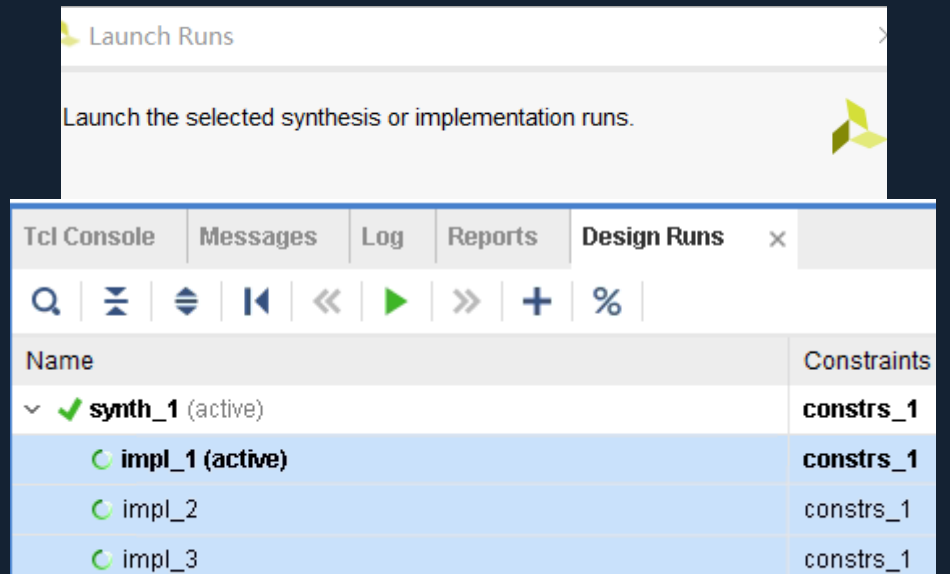

```
get_param  general.maxThreads
2
set_param general.maxThreads 16
16
get_param  general.maxThreads
16
```

Jobs vs Threads

Jobs: 同时在跑的runs的数量

ug904 P27

Use the **Number of jobs** drop-down menu to define the number of local processors to use when launching multiple runs simultaneously.



Threads: 使用的CPU线程数

ug904 P7

Multithreading with the Vivado Tools

On multiprocessor systems, Vivado tools use multithreading to speed up certain processes, including DRC reporting, static timing analysis, placement, and routing. The maximum number of simultaneous threads varies, depending on the number of processors and task. The maximum number of threads by task is:

- DRC reporting: 8
- Static timing analysis: 8
- Placement: 8
- Routing: 8
- Physical optimization: 8

The default number of maximum simultaneous threads is based on the OS. For Windows systems, the limit is 2; for Linux systems the default is 8. The limit can be changed using a parameter called `general.maxThreads`. To change the limit use the following Tcl command:

```
Vivado% set_param general.maxThreads <new limit>
```

where the new limit must be an integer from 1 to 8, inclusive.

◎ 第二周计划

1.13~1.19

- 数电前3章 & Verilog基础语法
- 计组前2章
- 开始C & Python的学习
- Vivado & HLS: ug888 ug892 ug998
- 李宏毅: Regression
- 《深度学习》数学基础

Thank you