

finn框架环境配置_finn ubuntu22.04-爱代码爱编程

2024-04-14 分类: fpga开发 (/category/fpga%E5%BC%80%E5%8F%91) 深度学习 (/category/%E6%B7%B1%E5%BA%A6%E5%AD%A6%E4%B9%A0) 神经网络 (/category/%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C) 狗看了都摇头的深度学习 (/category/%E7%8B%97%E7%9C%8B%E4%BA%86%E9%83%BD%E6%91%87%E5%A4%B4%E7%9A%84%E6%B7%B1%E5%BA%A6%E5%AD%A6%E4%B9%A0)

环境: ubuntu22.04LTS、vivado2022.2

一、vivado和vitis安装 (2022.2)

在Xilinx官网 (/go?go=aHR0cHM6Ly9jaGluYS54aWxpbngY29tL3N1cHBvcnQvZG93bmxvYWQvaW5kZXguaHRtbC9jb250ZW50L3hpbGlueC96aC9kb3dubG9hZE5hdi92aXZhZG8tZGVzaWduLXRvb2xzLzlwMjlt

Mi5odG1s)下载vivado的自解压文件



📄 Xilinx Unified Installer 2022.2: Windows Self Extracting Web Installer (EXE - 209.61 MB)

MD5 SUM Value : 985168f6920c5ee2111c5d16573330e1

下载验证 ⓘ

摘要 签名 公钥

📄 赛灵思统一安装程序 (Xilinx Unified Installer 2022.2): Linux 自解压 Web 安装程序 (BIN - 271.02 MB)

MD5 SUM Value : 9bf473b6be0b8531e70fd3d5c0fe4817

下载验证 ⓘ

摘要 签名 公钥

📄 赛灵思统一安装程序 (Xilinx Unified Installer 2022.2) SFD (TAR/GZIP - 89.4 GB)

MD5 SUM Value : 4b4e84306eb631fe67d3efb469122671

下载验证 ⓘ

摘要 签名 公钥

下载完成后，在终端运行如下语句进行安装（在下载的文件目录下）

```
sudo chmod +x Xilinx_Unified_2022.2_1014_8888_Lin64.bin
sudo sh ./Xilinx_Unified_2022.2_1014_8888_Lin64.bin
```

中间安装步骤与windows相同，不再赘述

3. 配置环境，打开终端输入

```
gedit ~/.bashrc
```

在文件最后一行加入


```
source [your_path]/Xilinx/Vivado/2022.2/settings64.sh
source [your_path]/Xilinx/Vitis_HLS/2022.2/settings64.sh
```

保存文件并更新

```
source .bashrc
```

安装驱动


```
cd [your_path]/Xilinx/Vivado/2022.2/data/xicom/cable_drivers/l
sudo ./install_drivers
```



启动问题


在终端输入vivado启动时出现如下报错：

```
application-specific initialization failed: couldn't load file
```



解决办法，在终端输入以下命令安装缺失的包

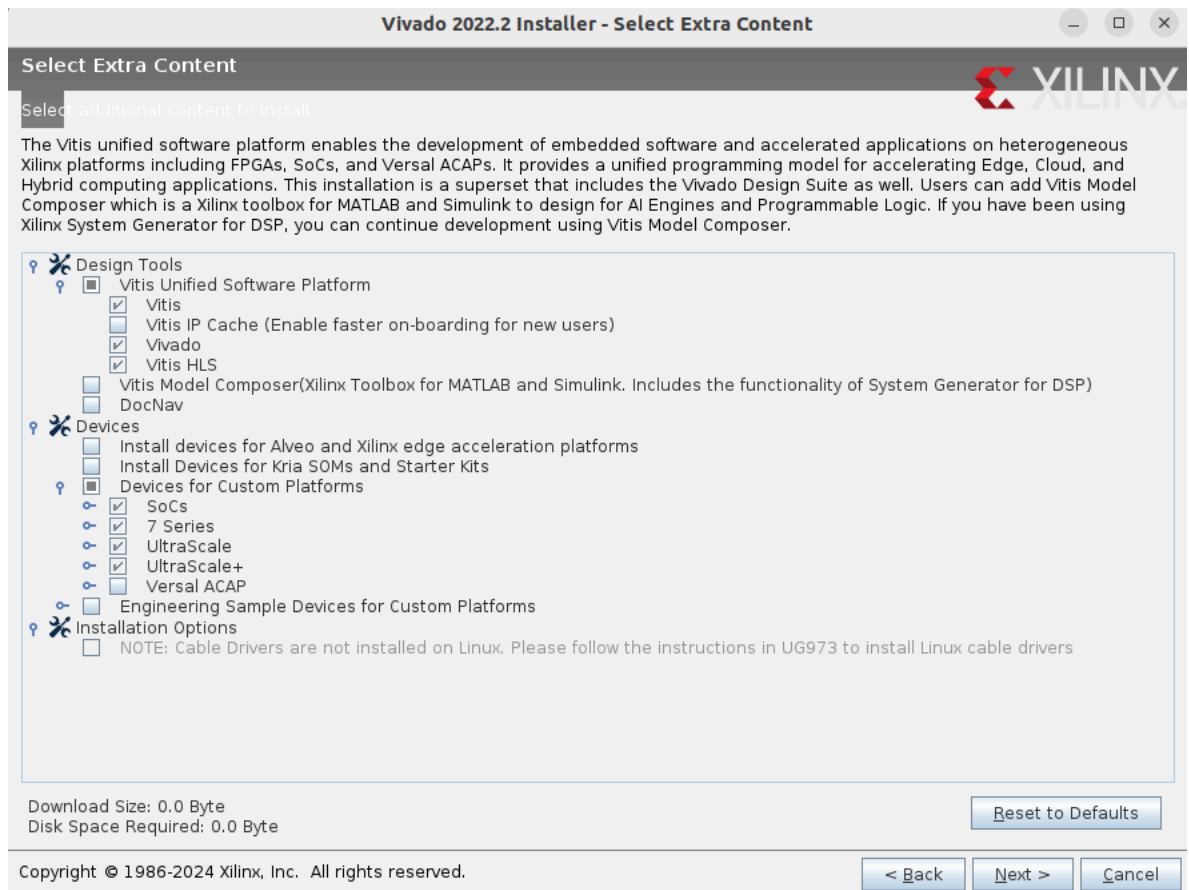
```
sudo apt update
sudo apt install libtinfo-dev
sudo ln -s /lib/x86_64-linux-gnu/libtinfo.so.6 /lib/x86_64-lin
```



license网上随便找一个导入即可

开始安装vitis，终端输入vivado，导航栏help -> Add Design Tools or Devices

10. 点击next，登录账号next，选择器件



next安装就好了，最后添加环境变量

```
source [your_path]/Xilinx/Vitis/2022.2/settings64.sh
```

二、安装Docker（无root）

打开终端、更新包索引


```
sudo apt update
```

安装依赖包

```
sudo apt install apt-transport-https ca-certificates curl gnupg
```

添加 Docker 的官方 GPG 密钥

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo
```



提示OK、安装

```
sudo add-apt-repository \  
"deb [arch=amd64] https://download.docker.com/linux/ubuntu \  
$(lsb_release -cs) \  
stable"
```

安装最新版本的 Docker Engine-Community

```
sudo apt install docker-ce
```

无root设置

创建docker组，提示已创建则直接进行下一步

```
sudo groupadd docker
```

将用户添加到docker组内

```
sudo usermod -aG docker $USER
```

激活对组的更改

```
newgrp docker
```

验证是否成功设置

```
docker run hello-world
```

出现下边界面则表明安装成功：

```
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

三、在docker上安装jupyter notebook

拉取jupyter镜像

```
sudo docker pull jupyter/scipy-notebook
```

下图表示镜像拉取完成

```
133281b5a4d5: Pull complete
bfb90c3fa999: Pull complete
333560684fd1: Pull complete
0865d1f6e9a4: Pull complete
Digest: sha256:3b37958b7b31ce94c3027d7c83c98fc16acfe166fab2de2f62ae54c50e59aed3
Status: Downloaded newer image for jupyter/scipy-notebook:x86_64-ubuntu-22.04
docker.io/jupyter/scipy-notebook:x86_64-ubuntu-22.04
lin@linpc:~$
```

2. 终端内执行

```
sudo docker images
```

可以查看docker内所有的镜像，如刚刚拉取的jupyter

```
lin@linpc:~$ sudo docker images
REPOSITORY          TAG         IMAGE ID      CREATED       SIZE
jupyter/scipy-notebook  latest     ad65fcfebde3  5 months ago  4.14GB
hello-world          latest     d2c94e258dcb  11 months ago 13.3kB
lin@linpc:~$
```

创建容器

```
sudo docker run -d -p 8888:8888 jupyter/scipy-notebook
```

查看容器是否创建成功

```
sudo docker ps
```

如下图所示，表明容器创建成功

```
lin@linpc:~$ sudo docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED
STATUS        PORTS                               NAMES
d0c2840f7976   jupyter/scipy-notebook             "tini -g -- start-no..." 4 minutes ago
Up 4 minutes (healthy)   0.0.0.0:8888->8888/tcp, :::8888->8888/tcp   vibrant_cerf
lin@linpc:~$
```

四、FINN配置

环境变量设置

直接给sudo管理员设置环境变量，因为后边执行.sh脚本文件要使用sudo权限

终端输入：

```
sudo gedit /etc/sudoers
```

找到Defaults env_reset在下方添加：

```
Defaults      env_keep += "FINN_XILINX_PATH=/home/lin/Xilinx"
Defaults      env_keep += "FINN_XILINX_VERSION=2022.2"
```

如下图所示：

```
9 Defaults      env_reset|
10 Defaults      env_keep += "FINN_XILINX_PATH=/home/lin/Xilinx"
11 Defaults      env_keep += "FINN_XILINX_VERSION=2022.2"
```

重启系统以完成设置

从存储库克隆FINN编译器，我一般放在vivado安装文件下。打开vivado设计工具存放文件夹，我的在home/lin/Xilinx，在空白处右键，选择在终端中打开，输入命令

```
git clone -b v0.10 https://github.com/Xilinx/finn.git
```


最新版本finn已更新至v0.10

4. 进入finn目录下，右键选择在终端中打开，输入命令验证是否安装成功

```
sudo bash run-docker.sh quicktest
```

运行结果：

```
-- Docs: https://docs.pytest.org/en/stable/warnings.html
= 963 passed, 16 skipped, 5 xfailed, 2 xpassed, 76237 warnings in 326.79s (0:05:26) =
```

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA (/go?go=aHR0cDovL2NyZWFiOaXZlY29tbW9ucy5vcmcvbiGljZW5zZXMuYnktc2EvNC4wLw==) 版权协议，转载请附上原文出处链接和本声明。

本文链接：https://blog.csdn.net/m0_52889836/article/details/137681930 (/go?go=aHR0cHM6Ly9ibG9nLmNzZG4ubmV0L20wXzUyODg5ODM2L2FydGljbGUvZGV0YWlscy8xMzc2ODE5MzA=)

基于vc709开发板利用ibert实现sf
p/sfp+ connectors gth收发器的
测试_vc709开发板套件-爱代码爱
编程 (/i/724865380101500)

【国产虚拟仪器】国产数据采集虚拟
仪器板卡结合labview的应用_基于la
bview的国产采集卡dll使用-爱代码爱
编程 (/i/914105380127680)

python分布式爬虫框架_python之简单Scrapy分布式爬虫的实现-爱代码爱编程
(/i/11701435465366)

2020-11-23 标签: python分布式爬虫框 (/tag/python%E5%88%86%E5%B8%83%E5%BC%8F%E7%88%AC%E8%99%AB%E6%A1%86)

分布式爬虫：爬虫共用同一个爬虫程序，即把同一个爬虫程序同时部署到多台电脑上运行，这样可以提高爬虫速度。在默认情况下，scrapy爬虫是单机爬虫，只能在一台电脑上运行，因为爬虫调度器当中的队列queue去重和set集合都是在本机上创建的，其他的电脑无法访问另外一台电脑上的内存的内容；想要让多台机器共用一个queue队列和set集合，可以让scrapy结合s

CONTINUE READING (/I/11701435465366)