We have used a high-performance computing system to run our code. Below I have given the exact steps and libraries that are available and that we used to run the code.

Below I have created the environments and run the code for Ariane chip design tasks while using the seed value 55. For the rest of the chip design tasks, namely Ariane133, IBM01, and IBM02, the steps are almost identical but only need to update the horizontal\_routes\_per\_micron, vertical\_routes\_per\_micron, macro\_horizontal\_routing\_allocation, and macro\_vertical\_routing\_allocation values of the placement\_util.py file inside the environment folder. These values can be obtained from the initial.plc file, which is inside the test\_data folder under the different chip design folders.

Setting up the Environments:

* mkdir Source\_Model\_Training
* cd Source\_Model\_Training
* mkdir ariane
* cd ariane
* Clone the repository
* module load python/3.10.2
* module load cudacore/.11.7.0
* module load tensorrt/8.6.1.6
* module load cudnn/8.6.0.163

Activate the virtualenv:

* virtualenv --no-download chip\_env\_ariane
* cd chip\_env\_ariane
* source bin/activate
* pip install --no-index --upgrade pip

Import and Install dm-reverb:

* wget <https://files.pythonhosted.org/packages/e5/ef/a16f50ceb44a3f6c582142d5880a482180ce2b64a426cf87d5d022db7d27/dm_reverb-0.10.0-cp310-cp310-manylinux2014_x86_64.whl>
* mv dm\_reverb-0.10.0-cp310-cp310-manylinux2014\_x86\_64.whl dm\_reverb-0.10.0-cp310-cp310-linux\_x86\_64.whl
* pip install dm\_reverb-0.10.0-cp310-cp310-linux\_x86\_64.whl

Copy the placement cost binary to .local/bin and make it executable:

* curl -O <https://storage.googleapis.com/rl-infra-public/circuit-training/placement_cost/plc_wrapper_main>
* chmod +x plc\_wrapper\_main
* mv plc\_wrapper\_main ~/.local/bin/
* setrpaths.sh --path $HOME/.local/bin

Install the Python Packages:

* pip install absl\_py==1.4.0 astunparse==1.6.3 cachetools==5.3.0 certifi==2022.12.7 charset-normalizer==3.0.1 cloudpickle==1.3.0 decorator==5.1.1 dm-reverb==0.10.0 dm\_tree==0.1.7 flatbuffers gast==0.4.0 gin-config==0.5.0 google-auth==2.16.0 google-auth-oauthlib==0.4.6 google-pasta==0.2.0 grpcio==1.47.0 gym==0.26.2 gym\_notices==0.0.8 h5py==3.7.0 idna==3.4 keras==2.11.0 keras-preprocessing==1.1.2 libclang==14.0.1 markdown==3.4.1 MarkupSafe==2.1.1 numpy==1.23.0 oauthlib==3.2.2 opt-einsum==3.3.0 packaging==23.0 pillow==9.3.0 pip==22.3.1 portpicker==1.5.2 protobuf==3.19.6 psutil==5.9.4 pyasn1==0.4.8 pyasn1-modules==0.2.8 pygame==2.1.0 requests==2.28.2 requests-oauthlib==1.3.1 rsa==4.9 scipy==1.9.3 setuptools==65.6.3 six==1.16.0 tensorboard==2.11.2 tensorboard-data-server==0.6.1 tensorboard-plugin-wit==1.8.1 tensorflow==2.11.0 tensorflow-estimator==2.11.0 tensorflow-io-gcs-filesystem==0.26.0 tensorflow-probability==0.19.0 termcolor==2.2.0 tf-estimator-nightly==2.8.0.dev2021122109 tfp-nightly==0.20.0.dev20230204 typing\_extensions==4.4.0 urllib3==1.26.14 werkzeug==2.2.2 wheel==0.38.4 wrapt==1.14.1 tk readline tzdata==2021.5 onnx onnx-graphsurgeon
* pip install --no-dependencies tf-agents==0.15.0

After that, you need to run the training from scratch script file to create the source model for the Ariane chip design task. After that, to run the fine-tuning and PMCTS scripts, you need to first run the scratch training for Ariane133, IBM01, and IBM02 to create the models for each chip design. And then you can run the fine-tuning and PMCTS script files. The steps are similar for the rest of the chip design tasks.