This is a README file. In this file, we will detail the file structure, how to compile and run the code and check the output.

1 File structure

The main files to be used are the following:

1. `main.cpp` This is the main function of the AXI4-NoC by which it sends states to the RL agent and receives actions from it.

2. `interfaceCPP.cpp` and `interfacePy.py` form the data transfer interface.

3. `RLtrain_new.py` is used to train the SAC agent.

4. `RLevaluation.py` load the trained SAC agent to perform admission control in the NoC.

5. `NoC_SAC_1.zip` stores the weight parameters of the SAC model.

6. `./RecordFiles/OnlyURStransLevel_delay_20230118.py`:This written by Yizhi and it's capable of counting a lot of data. Here we've mainly used the programme to calculate the average end-to-end latency.

7. `./RecordFiles/60000_2025_epoch_0_URS_packetdelay.txt` It stored the time data from the NoC operation without admission control.
   `./RecordFiles/60000_2025_epoch_1_URS_packetdelay.txt` It stored the time data from the NoC operation with the SAC agent.

8. `FunctionDefineSwitch.h`. It stored some parameter to control some functions in `main.cpp`. What we need to used is `param_RL_SwitchOn_inMain` and `Param_RL_writeFilePakcetDelay_InMain`. If we define `param_RL_SwitchOn_inMain` the RL Switch in the `main.cpp` will be opened. And if we do not define this parameter, we will get the results without admission controller. If we define the `Param_RL_writeFilePakcetDelay_InMain`, the NoC operation results will be stored in `./RecordFiles/60000_2025_epoch_1_URS_packetdelay.txt`. Otherwise the results will be stored in `./RecordFiles/60000_2025_epoch_0_URS_packetdelay.txt`.

   In most cases, these two parameters are either both defined or both undefined. If defined simultaneously, the results of the NoC operation controlled by the SAC agent will be stored in `./RecordFiles/60000_2025_epoch_1_URS_packetdelay.txt`. Otherwise, the the results of the NoC operation without admission controller will be stored in `./RecordFiles/60000_2025_epoch_0_URS_packetdelay.txt`.

9. `Run.py` is a automatic script. It can evaluation the agent or train a new RL agent according to your requirement.

2 Running and results

2.1 Method1: Manually run

1. You need to install a C++ IDE (we used Clion here) and compile `main.cpp` into an executable file. Here, the executable `./cmake-build-debug/hello_cmake`

2. You need to configure Python 3.8 or above and install the relevant libraries, mainly the Stable-Baselines3 library. You can try this command `pip install stable-baselines3[extra]`, If the above command do not work, please visit their official website directly.

3. Since we have already trained the model `NoC_SAC_1.zip`, you can directly use the existing model to perform admission control in the NoC. Specifically, first execute `python3 RLevaluation.py`. When the terminal prints out the "reset ,begin", then excute `mian.cpp` .

4. At this point, you should pay attention to the output of `main.cpp`, when the first time to print the results as shown in Figure 1 , first interrupt `mian.cpp` and then interrupt `RLevaluation.py`. This is because the results of each evaluation are the same, the NoC's operation data has been completely recorded no longer need to evaluate the second time.

5. After recording the NoC operation data, you can execute `./RecordFiles/OnlyURStransLevel_delay_20230118.py` This will result in two histograms recording the distribution of the end-to-end latency of the signals. However, we did not use this two figures in our report. At this point you can also focus on the print message, which outputs some of the parameters we used in the final report, such as Figure 2. If you want to directly observe the average end-to-end latency after performing admission control, don't forget to change this line of code in this program `txtFileName_URS = "60000_2025_epoch_0_URS_packetdelay"` to `txtFileName_URS = "60000_2025_epoch_1_ URS_packetdelay"`.

```
average URS latency:
  : avgDelay 168NITotalDelay 45855672.000000VN: avgDelay 168NITotalDelay/VCNITotal_RespURSPakcet217.553 VCNITotal_RespURSPakcet 21077
 this need to be debugged:  VCNetworkTotal_RespURSPakcet  210779
this need to be compare  45855672.000   delayFromPeriod 45855576.000
VN: WBQ totalnumURSCount    421558   VN: WBQ allURS avgDelay   49.0
 global_respSignalNum 210779 global_injSignalHighGroupNum 9079
overall_signal_num210779  global_Packet_ID 421558
time consumption: 39.6
```

Figure 1: The output of mian.cpp

```
df mean 222 sigID                    105389.000000
overall                 245.434123
req wait                100.945763
req travel               64.298607
resp wait                35.583540
resp travel              35.996902
```

Figure 2: The output of OnlyURStransLevel_delay_20230118.py

Figure 1 shows some important parameters.For example, request signal number=210779, response signal number =210779 and the total packet number= 421558. All of this parameters show the generated packets are delivered to the corresponding destinations.

2.2   Method 2: Atomatic script

This method is more easier than previous one.

1. Step 1 is the same as the Step 1 in Method 1.

2. Step 2 is the same as the Step 2 in Method 1.

3. Define parameters `param_RL_SwitchOn_inMain` and `Param_RL_writeFilePakcetDelay_InMain`in `FunctionDefineSwitch.h` according to your experimental needs.

4. Execute the `Run.py` and select the runing mode. Input `0` means evaluation mode and input `1` means training mode. Because we have already trained the model `NoC_SAC_1.zip`, so you can just input `0` to select evaluation mode. This programm will execute the evaluation process and store the NoC operation data. And it can also stop the `main.cpp` and `RLevaluation.py` after finishing evaluation

5. Step 4 is the same as the Step 5 in Method 1.