1. Inference:

Description: The "inference" directory contains the inference process code (without backward propagation).
① t10k-images-idx3-ubyte: images for CNN testing (Low dimension is 28 X 28. If images are in other dimensions, we will scale it or modify the network structure and we will send you the new structure after we know the dimension of your pictures.)
② t10k-images-idx1-ubyte: labels corresponding to the test images in ①.
③ trained_data directory: consists of the pre-trained weights and bias of each layer, which is extracted from the training model (in the train folder) with backward propagation.
④ forward directory: contains the VHDL and Verilog code generated by Xilinx Vivado HLS.

How to run:
① the C++ code in the "inference_only" directory can be compiled and run in Linux. We use gcc/g++ version 7.4.0 on Ubuntu 18.04.1:

g++ *.cpp (compile)
./a.out (execution)

② the VHDL/Verilog code needs to be executed in the Xilinx Vivado HLS. We use Vivado HLS 2019.1 on Windows 10:

1). Open the code in HLS:

Xilinx design tools → Vivado HLS 2019.1 → Open project (as shown in Fig 1) select the forward folder, then click OK.

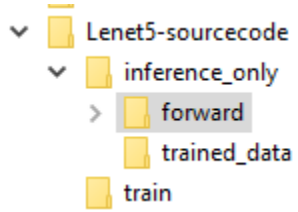


Fig. 1 Open the VHDL/Verilog design in HLS.

After click OK, the project will be opened, the target device is xc7a100t-csg324-1. You can change it to other devices by using the solution setting under solution.

2). Run Simulation:

Solution → C/RTL Co-simulation (as shown in Fig. 2), choose VHDL or Verilog here and click OK to start the simulation. We have simulated the VHDL code in our experiment. (Note: This process took about 2 hours on my computer.)

If there is a device change or setting change, re-synthesis (which generates the RTL code) needs to be conducted before simulation (i.e., Solution → Run C Synthesis →Active Solution). Then run the co-simulation and check if the generated RTL code contains any logic error.

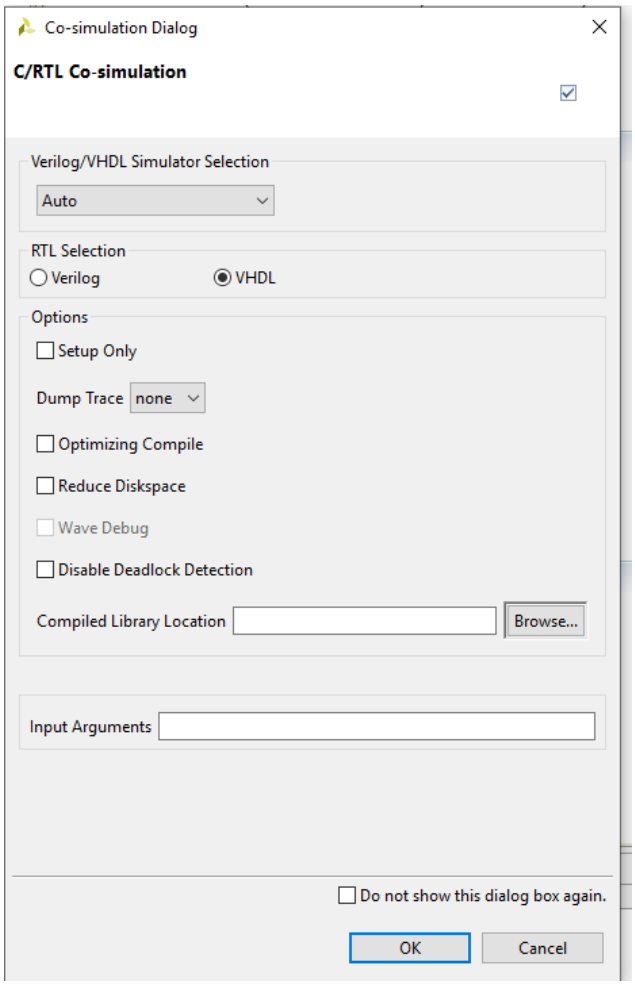


Fig. 2 C/RTL Co-simulation interface.

---

2. Training

Description: The "train" directory contains the LeNet-5 code for training and inference (both forward and backward propagation). This network has been trained with 60000 images, and tested with 10000 images. Its accuracy is about 80%.
① train-images-idx3-ubyte: the training image dataset which consists of 60000 training images.
② train-labels-idx1-ubyte: labels corresponding to the training images in ①.
③ weight and bias files are generated after the training process, and we use them for inference. The weight and bias used in the inference folder are generated here.

The training can be performed off-line. Then we do real-time inference on the FPGA board using the parameters generated in training.

How to run:

① The C++ code in the “train” directory can be executed in Linux. We use gcc/g++ version 7.4.0 on Ubuntu 18.04.1:

g++ *.cpp (compile)
./a.out (execution)

It will output the error rate. We calculate the accuracy = 100% - error rate.

---

3. C1_hardcode:

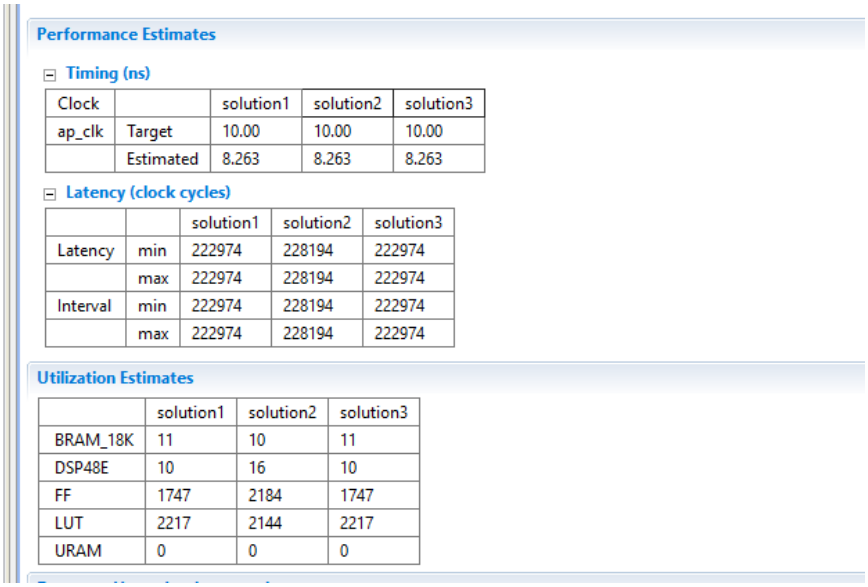
Description: The “C1_hardcode” directory contains the LeNet-5 convolutional 1 layer code. This code is used to test the performance improvement from difference unrolling techniques. ①Three solutions in the secondary “C1_hardcode” directory: Solution 1 does not use any improvement technique. Solution 2 applies an unrolling method proposed in a paper. Solution 3 uses our proposed unrolling technique.

How to run:

① The C++ code in the “C1_hardcode” directory can be executed in Linux. We use gcc/g++ version 7.4.0 on Ubuntu 18.04.1:

g++ *.cpp (compile)
./a.out (execution)

② the VHDL/Verilog code is executed using Xilinx Vivado HLS. We use Vivado HLS 2019.1 on Windows 10. In C1_hardcode, we run the C/RTL Co-simulation as mentioned before. We can also compare the latency and resource consumption by clicking Project, then compare reports, and selecting Solution 1, Solution 2, and Solution 3 sequentially. The comparison result is shown in Fig 3.



Performance Estimates

Timing (ns)

Clock		solution1	solution2	solution3
ap_clk	Target	10.00	10.00	10.00
	Estimated	8.263	8.263	8.263

Latency (clock cycles)

		solution1	solution2	solution3
Latency	min	222974	228194	222974
	max	222974	228194	222974
Interval	min	222974	228194	222974
	max	222974	228194	222974

Utilization Estimates

	solution1	solution2	solution3
BRAM_18K	11	10	11
DSP48E	10	16	10
FF	1747	2184	1747
LUT	2217	2144	2217
URAM	0	0	0

Fig. 3 Comparison Report.