

# SYA14 - Neuromorphic Computing

## *Lab 1*

### 1 Objective

In this lab, you will learn to be familiar with Spiking Neural Network, Verilog HDL, and CAD tools.

### 2 Prerequisite

The following are the prerequisites of this exercise:

- Linux and command line tool.
- Verilog HDL.
- Modelsim with command line (the example program doesn't support GUI).
- Familiar with CAD tool to export: area cost, timing and power consumption.
- Fully connected model.
- MNIST dataset.

### 3 Spiking Neural Network Design

The source code is written in Verilog HDL. Simulation scripts using Modelsim are provided.

The SNN model is fully connected (784-48-10) for the MNIST dataset. The weights are pre-trained and converted to an 8-bit fixed point format. The structure of Verilog HDL is as follows:

- top.v: top-level file of SNN.
  - SNPC.v: neuromorphic core acts as a layer.
    - \* SNPC\_control.v: controller of SNPC.
    - \* xbar.v: crossbar of synapses.
    - \* LIF\_neuron.v: Leaky-Integrate-and-Fire neuron.

### 4 Exercise 1-1: Spiking Neural Network Design

Description: This design is about a simple spiking neural network (SNN) written in Verilog HDL using ModelSim.

## 4.1 Step 1

Download the source code from the website: <https://web-ext.u-aizu.ac.jp/misc/neuro-eng/book/NeuromorphicComputing/lab.html> (direct URL: [https://web-ext.u-aizu.ac.jp/misc/neuro-eng/book/NeuromorphicComputing/lab/SNN\\_RTL-main.zip](https://web-ext.u-aizu.ac.jp/misc/neuro-eng/book/NeuromorphicComputing/lab/SNN_RTL-main.zip))

Unzip the source code into the folder and change the directory to it. The content of the source code is as in Figure 1.

```
(base) [doanh@zxp007 SNN_RTL-main] $ ls
env.sh  README.md  RTL  SIM  TB
(base) [doanh@zxp007 SNN_RTL-main] $
```

Figure 1: The context of the source code.

- *env.sh*: environment bash file to run CAD tools. Please note this only works with the CAD server zxp007.
- *RTL*: folder of RTL code of SNN.
- *SIM*: simulation folder using Modelsim
- *TB*: folder of testbench

## 4.2 Step 2

Link tools' paths into the terminal environment using the following command.

```
source env.sh
```

The content of *env.sh* is as in Figure 2. Please note that this code only works with CAD server zxp007.

```
env.sh
~/Work/SNN_RTL-main
Save

# Design Compiler
export PATH=$PATH:$SYNOPSYS_DIR/syn/latest/bin
# IC Compiler
export PATH=$PATH:$SYNOPSYS_DIR/icc/latest/bin
# VCS
export PATH=$PATH:$SYNOPSYS_DIR/vcs-mx/latest/bin
export VCS_HOME=$SYNOPSYS_DIR/vcs-mx/latest
#setenv VCS_ARCH_OVERRIDE linux
#setenv VCS_ARCH_OVERRIDE amd64
# Raphael
export PATH=$PATH:$SYNOPSYS_DIR/taurus_X2005.10/bin
export PATH=$PATH:$SYNOPSYS_DIR/raphael_62012.06/bin
# PrimeTime
export PATH=$PATH:$SYNOPSYS_DIR/primetime/pts/P-2019.03-SP3/bin
#export PATH=$PATH:$SYNOPSYS_DIR/pts_vF-2011..12-SP1/bin
# Sentaurus
export PATH=$PATH:$SYNOPSYS_DIR/sentaurus_v6_2012.06-SP2/bin
export STDB=$HOME
# NanoTime
export PATH=$PATH:$SYNOPSYS_DIR/nt_vH-2012.12-SP1/bin

# Mentor setup
# -----
export MENTOR_DIR=/home/apps/vdec/Mentor
# Calibre
export PATH=$PATH:$MENTOR_DIR/cal.latest/bin
export MGC_HOME=$MENTOR_DIR/cal.latest

# Agilent setup
# -----
export AGILENT_DIR=/home/apps/vdec/agilent
#EMpro
export HPEESOF_DIR=$AGILENT_DIR/EMPro2012_09
export PATH=$PATH:$HPEESOF_DIR/linux_x86_64/bin
export ossSimUserSidir=$HPEESOF_DIR/idf/ads_site/si
#export PATH=/home/doanh/Work/MATLAB/R2022B/bin:$PATH
export PATH=/home/apps/vdec/Mentor/modeltech/bin:$PATH

sh Tab Width: 8 Ln 93, Col 18 INS
```

Figure 2: The context of the env.sh file.

### 4.3 Step 3

Go to the SIM (simulation) directory using the following command.

```
cd SIM
```

You can see the content of the simulation folder as in Figure 3.

```
(base) [doanh@zxp007 SIM]$ ls
_ 1_10 1_2 1_20 input output run.csh run.sh script
(base) [doanh@zxp007 SIM]$
```

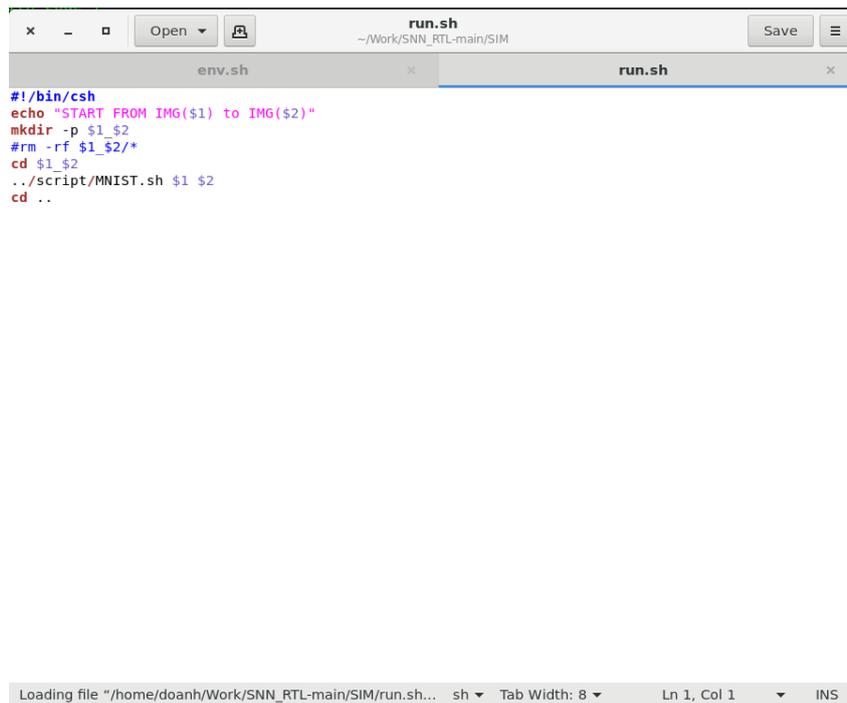
Figure 3: The context of the SIM directory.

### 4.4 Step 4

In this step, we will run the simulation. To run in bash, we need executing permission (run only once in the new system):

```
chmod +x run.sh
chmod +x script/MNIST.sh
```

We can also observe the content of the bash script as in Figure 4



```
#!/bin/csh
echo "START FROM IMG($1) to IMG($2)"
mkdir -p $1_$2
#rm -rf $1_$2/*
cd $1_$2
../script/MNIST.sh $1 $2
cd ..
```

Figure 4: The context of the running script.

Assuming we want to test image index 1 to 10, the following command is used.

```
./run.sh 1 10
```

Following is the display of the terminal with the command:

### 4.5 Step 5

The output of the simulation can be found at `SIM/output/MNIST0-10K-1-10.txt`. Figure 6 is the output example.

```

doanh@zxp007:~/Work/SNN_RTL-main/SIM
doanh@spiketrain: ~ | emacs -nw 2_fault_insertion.py | doanh@zxp007:~/Work/SNN_...
# input = ../input/RAM/WEIGHT/W_L-0000_N-0040.bin
# input = ../input/RAM/WEIGHT/W_L-0000_N-0041.bin
# input = ../input/RAM/WEIGHT/W_L-0000_N-0042.bin
# input = ../input/RAM/WEIGHT/W_L-0000_N-0043.bin
# input = ../input/RAM/WEIGHT/W_L-0000_N-0044.bin
# input = ../input/RAM/WEIGHT/W_L-0000_N-0045.bin
# input = ../input/RAM/WEIGHT/W_L-0000_N-0046.bin
# input = ../input/RAM/WEIGHT/W_L-0000_N-0047.bin
# input = ../input/RAM/WEIGHT/W_L-0001_N-0000.bin
# input = ../input/RAM/WEIGHT/W_L-0001_N-0001.bin
# input = ../input/RAM/WEIGHT/W_L-0001_N-0002.bin
# input = ../input/RAM/WEIGHT/W_L-0001_N-0003.bin
# input = ../input/RAM/WEIGHT/W_L-0001_N-0004.bin
# input = ../input/RAM/WEIGHT/W_L-0001_N-0005.bin
# input = ../input/RAM/WEIGHT/W_L-0001_N-0006.bin
# input = ../input/RAM/WEIGHT/W_L-0001_N-0007.bin
# input = ../input/RAM/WEIGHT/W_L-0001_N-0008.bin
# input = ../input/RAM/WEIGHT/W_L-0001_N-0009.bin
# Output spikes after all timestep of IMAGE(10): '{13, 0, 0, 0, 0, 2, 0, 0, 0, 0}'
# Output is number 9 with 13 spikes
# ** Note: $finish : ../TB/SNPC/TB_SNPC_MNIST.v(163)
# Time: 128247 ns Iteration: 1 Instance: /TB_SNPC_MNIST
# End time: 15:31:42 on Oct 13, 2023, Elapsed time: 0:00:02
# Errors: 0, Warnings: 0
(base) [doanh@zxp007 SIM]$

```

Figure 5: The output of the script.

```

MNIST_10K_V-1.0.txt
~/Work/SNN_RTL-main/SIM/output
env.sh | run.sh | Makefile | MNIST_10K_V-1.0.txt
1, 7, '{0, 0, 18, 0, 0, 0, 0, 0, 0, 0}'
2, 2, '{0, 0, 0, 0, 0, 0, 0, 0, 17, 0, 0}'
3, 1, '{0, 0, 0, 0, 0, 0, 0, 0, 0, 22, 0}'
4, 0, '{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 12}'
5, 4, '{0, 0, 0, 0, 0, 0, 11, 0, 0, 0, 0}'
6, 1, '{0, 0, 0, 0, 0, 0, 0, 0, 0, 22, 0}'
7, 4, '{0, 0, 0, 0, 0, 18, 0, 0, 0, 0, 0}'
8, 9, '{17, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}'
9, 6, '{0, 0, 0, 15, 4, 0, 0, 0, 0, 0, 0}'
1, -1, '{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}'
1, 7, '{0, 0, 18, 0, 0, 0, 0, 0, 0, 0, 0}'
1, 7, '{0, 0, 18, 0, 0, 0, 0, 0, 0, 0, 0}'
1, 7, '{0, 0, 18, 0, 0, 0, 0, 0, 0, 0, 0}'
1, 2, '{0, 0, 0, 0, 0, 0, 0, 0, 17, 0, 0}'
1, 1, '{0, 0, 0, 0, 0, 0, 0, 0, 0, 22, 0}'
1, 0, '{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 12}'
Plain Text | Tab Width: 8 | Ln 1, Col 1 | INS

```

Figure 6: The example of the output file.

## 5 Exercise 1-2: Area, timing, and power report of the SNN module with FPGA tool

This exercise will utilize the FPGA tool to perform the area, timing, and power.

### 5.1 Step 1: Wrap the RTL to the FPGA project

In the first step, wrap the design of `top.v` as the top module and add other modules to the FPGA project.

### 5.2 Step 2: Synthesize the top module

In the second step, synthesize the top module.

### 5.3 Step 3: Read the report

In the third step, you need to report an evaluation of the area, timing, and power consumption of the SNN module.

## 6 Submission format and Deadline

Your report should be prepared in English and should contain the following:

1. Your name, your ID, and the Lab #.
2. All reports
3. Submission format: soft copy.

---

Note: This Laboratory is designed for the book <sup>1</sup>

---

<sup>1</sup>Book: Neuromorphic Computing Principles and Organization 1st, Edition, ISBN-10: 3030925242, ISBN-13: 978-3030925246, Publisher: Springer, May 2022.