

# NYCU-EE IC LAB - Spring 2023

## Lab04 Exercise

### Design: Simple Recurrent Neural Network

#### Data Preparation

1. Extract test data from TA's directory:

```
% tar xvf ~iclabta01/Lab04.tar
```

2. The extracted LAB directory contains:

a. **00\_TESTBED**

b. **01\_RTL**

c. **02\_SYN**

d. **03\_GATE**

#### Design Description

A **Recurrent Neural Network (RNN)** is a class of artificial neural networks where connections between nodes form a directed or undirected graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. Derived from feedforward neural networks, RNNs can use their internal state (memory) to process variable length sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition. Recurrent neural networks are theoretically Turing complete and can run arbitrary programs to process arbitrary sequences of inputs.

In this exercise, you are asked to implement the simple RNN model as Fig1.

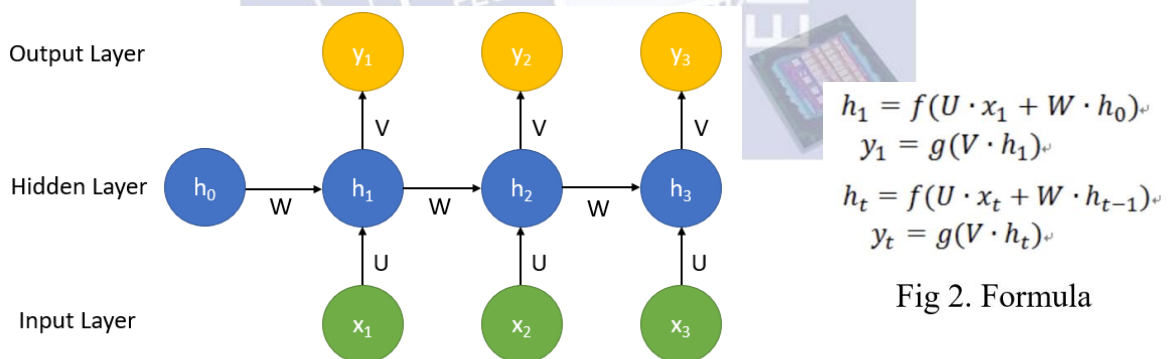


Fig 2. Formula

Fig 1. simple RNN

$$f(x) = \max(0.1x, x)$$

**Leaky ReLU**  
 $\max(0.1x, x)$

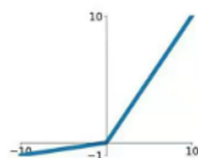


Fig 3.1. The Leaky ReLU activation function

$$g(x) = \sigma(x)$$

**Sigmoid**

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

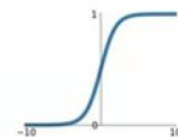


Fig 3.2. The sigmoid activation function

$$U = \begin{bmatrix} u_{00} & u_{01} & u_{02} \\ u_{10} & u_{11} & u_{12} \\ u_{20} & u_{21} & u_{22} \end{bmatrix} \quad W = \begin{bmatrix} w_{00} & w_{01} & w_{02} \\ w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \end{bmatrix} \quad V = \begin{bmatrix} v_{00} & v_{01} & v_{02} \\ v_{10} & v_{11} & v_{12} \\ v_{20} & v_{21} & v_{22} \end{bmatrix}$$

Fig 4. Weight matrix

$$x_1 = \begin{bmatrix} x_{10} \\ x_{11} \\ x_{12} \end{bmatrix} \quad x_2 = \begin{bmatrix} x_{20} \\ x_{21} \\ x_{22} \end{bmatrix} \quad x_3 = \begin{bmatrix} x_{30} \\ x_{31} \\ x_{32} \end{bmatrix}$$

Fig 5. Input x vector

$$h_0 = \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \end{bmatrix} \quad y_1 = \begin{bmatrix} y_{10} \\ y_{11} \\ y_{12} \end{bmatrix} \quad y_2 = \begin{bmatrix} y_{20} \\ y_{21} \\ y_{22} \end{bmatrix} \quad y_3 = \begin{bmatrix} y_{30} \\ y_{31} \\ y_{32} \end{bmatrix}$$

Fig 6.  $h_0$  vector

Fig 7. Output y vector

$$\begin{bmatrix} u_{00} & u_{01} & u_{02} \\ u_{10} & u_{11} & u_{12} \\ u_{20} & u_{21} & u_{22} \end{bmatrix} * \begin{bmatrix} x_{10} \\ x_{11} \\ x_{12} \end{bmatrix} = \begin{bmatrix} u_{00} * x_{10} + u_{01} * x_{11} + u_{02} * x_{12} \\ u_{10} * x_{10} + u_{11} * x_{11} + u_{12} * x_{12} \\ u_{20} * x_{10} + u_{21} * x_{11} + u_{22} * x_{12} \end{bmatrix}$$

Fig 8. Example of matrix operation

## ● Description

$x_1, x_2, x_3, h_0, h_1, h_2, h_3, y_1, y_2, y_3$  are all 3\*1 vectors, such as in Figs. 5, 6, and 7.  $U, V$ , and  $W$  are all 3\*3 matrices, as shown in Fig. 4. When  $in\_valid$  is pulled high, you will get continuous input from  $x_1$  to  $x_2$  and finally to  $x_3$ , that is to say, the sequence is  $x_{10} \rightarrow x_{11} \rightarrow x_{12} \rightarrow x_{20} \rightarrow x_{21} \rightarrow x_{22} \rightarrow x_{30} \rightarrow x_{31} \rightarrow x_{32}$ . At the same time, you will get the input  $h_0$  by the sequence  $h_{00} \rightarrow h_{01} \rightarrow h_{02}$ . When  $in\_valid$  is pulled high, you will also get the corresponding weights, and their input is in accordance with the **raster scan order** method (from left to right, from top to bottom, such as  $u_{00} \rightarrow u_{01} \rightarrow u_{02} \rightarrow u_{10} \rightarrow u_{11} \rightarrow u_{12} \rightarrow u_{20} \rightarrow u_{21} \rightarrow u_{22}$ ,  $W, V$  are also the same method). First, refer to Fig. 1 and operate according to the formula of Fig 2,  $U * x_1 + W * h_0$ , a 3\*1 vector will be obtained. Next, we use the **Leaky RELU** activation function for the 3 elements in this vector, which can be referring to Fig. 3.1, and the  $h_1$  vector of the middle hidden layer can be obtained. Next,  $V * h_1$  will get another 3\*1 vector, and then use the **Sigmoid** activation function on the 3 elements of this vector to get the  $y_1$  vector of the output layer. Finally,  $y_1, y_2, y_3$  are the final output we want, and the output should also follow  $y_1$  to  $y_2$  and finally to  $y_3$ , that is to say, the order is  $y_{10} \rightarrow y_{11} \rightarrow y_{12} \rightarrow y_{20} \rightarrow y_{21} \rightarrow y_{22} \rightarrow y_{30} \rightarrow y_{31} \rightarrow y_{32}$ . For detailed flow chart and formulas, please refer to Figs. 1 to 8. **It should be noted that  $h_1$  and  $h_2$  will be put in the next level as input.**

## Inputs and Outputs

The following are the definitions of input signals

Input Signals	Bit Width	Definition
clk	1	Clock.
rst_n	1	Asynchronous active-low reset.

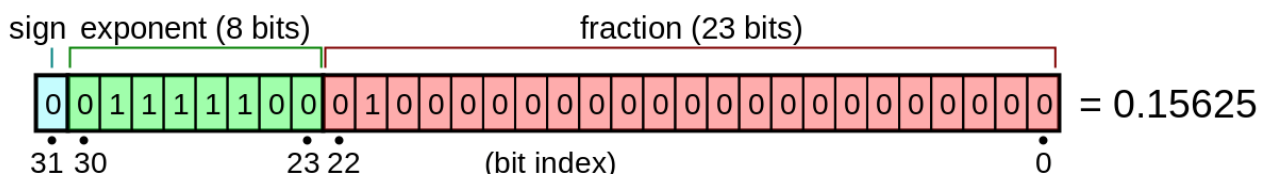
in_valid	1	High when all input is valid.
weight_u, weight_w, weight_v	32	The input weight signals. There are 9 signals for each weights. The arithmetic representation follows the IEEE-754 floating number format.
data_x	32	The input x signals. There are 3*3=9 signals (x <sub>1</sub> : first 3 signals, x <sub>2</sub> : middle 3 signals, x <sub>3</sub> : last 3 signals). The arithmetic representation follows the IEEE-754 floating number format.
data_h	32	The input h <sub>0</sub> signals. There are 3 signals. The arithmetic representation follows the IEEE-754 floating number format.

The following are the definitions of output signals

Output Signals	Bit Width	Definition
out_valid	1	High when out is valid.
out	32	The output y signals. There are 3*3=9 signals (y <sub>1</sub> : first 3 signals, y <sub>2</sub> : middle 3 signals, y <sub>3</sub> : last 3 signals). The arithmetic representation follows the IEEE-754 floating number format.

Each time you output the result, our pattern will check the correctness of it. Basically, if you follow the formulas and use IEEE floating point number IP, you should get same result as our answer.

**However, we release the constraint; you may have an error under 0.0005 for the result after converting to float number. This means that we will convert your output from binary format**



$$\text{sign} = +1$$

$$\text{exponent} = (-127) + 124 = -3$$

$$\text{fraction} = 1 + 2^{-2} = 1.25$$

$$\text{value} = (+1) \times 1.25 \times 2^{-3} = +0.15625$$

into real float number, and compare with our answer. Error will be calculated by '(golden-ans)/golden' and get its absolute value. If the error is higher than the value, you will fail this lab.

Binary form: 00111101101100101010000001000101

IEEE floating number: 0.08721975

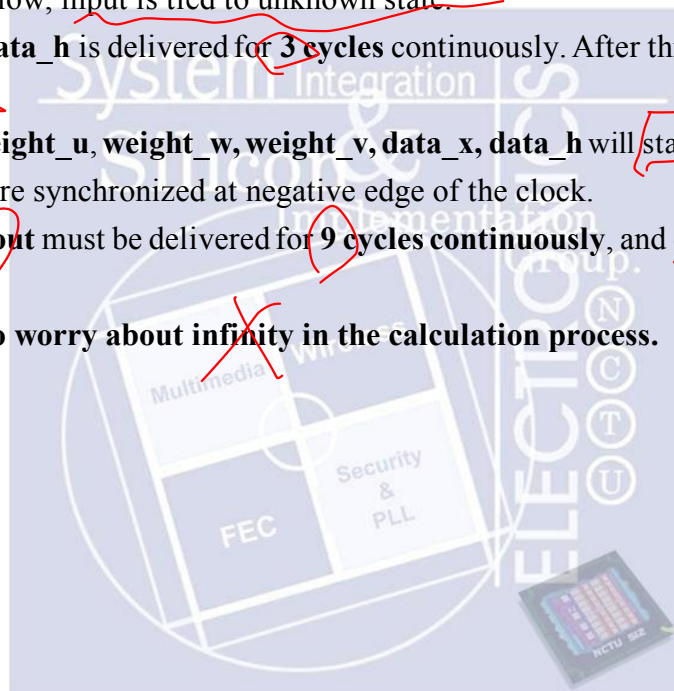
nWave:



nWave will round the number for display, but the computation will not be affected. The following numbers are all from nWave, thus the computations are performed by IPs in Verilog.

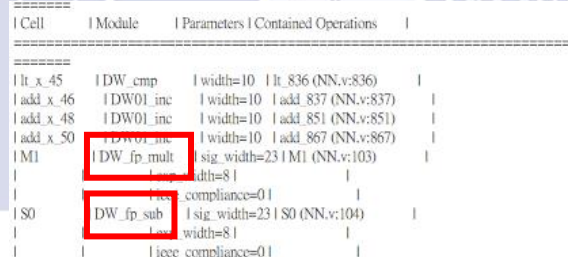
※  $e-02 = 10^{-2}$

1. The input signal **weight\_u**, **weight\_w**, **weight\_v**, **data\_x** are delivered for **9 cycles** continuously. When **in\_valid** is low, input is tied to unknown state.
2. The input signal **data\_h** is delivered for **3 cycles** continuously. After three cycles, input is tied to unknown state.
3. The input signal **weight\_u**, **weight\_w**, **weight\_v**, **data\_x**, **data\_h** will start to be given at same time.
4. All input signals are synchronized at negative edge of the clock.
5. The output signal **out** must be delivered for **9 cycles continuously**, and **out\_valid** should be **high** simultaneously.
6. You don't need to worry about infinity in the calculation process.



## Specifications

1. Top module name: NN (design file name: NN.v)
2. **You have to check an error under 0.0005 for the result after converting to float number. If the error is higher than the value, you will fail this lab.**
3. **It is asynchronous reset and active-low architecture. If you use synchronous reset (considering reset after clock starting) in your design, you may fail to reset signals.**
4. The reset signal (rst\_n) would be given only once at the beginning of simulation. All output signals should be reset after the reset signal is asserted.
5. The **out** should be reset after your **out\_valid** is pulled down.
6. **The latency is the clock cycles between the first falling edge of the in\_valid and the last rising edge of out\_valid. The rising edge of out\_valid should arrive within 100 cycles after the falling edge of in\_valid.**
7. The area is limited in **2500000**. Also, the synthesis time should be less than **1.5** hours.
8. You can adjust your clock period by yourself, but the maximum period is **50 ns**. The precision of clock period is 0.1, for example, 4.5 is allowed, 4.55 is not allowed.
9. The input delay is set to **0.5\*(clock period)**.
10. The output delay is set to **0.5\*(clock period)**, and the output loading is set to **0.05**.
11. The synthesis result of data type **cannot** include any **latches**.
12. After synthesis, you can check NN.area and NN.timing. The area report is valid when the slack in the end of timing report should be **non-negative (MET)**.
13. **In this lab, you must use at least one IEEE floating point number IP from Designware. We will check it at NN.resource in 02\_SYN/Report/. The example shows in following figure.**



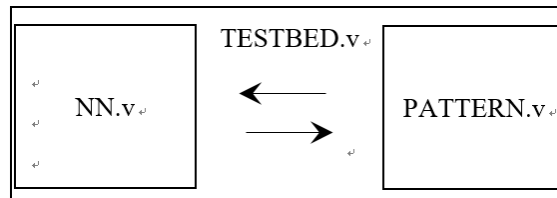
Cell	Module	Parameters	Contained Operations
lit_x_45	DW_cmp	width=10	lit_836 (NN.v:836)
add_x_46	DW01_inc	width=10	add_837 (NN.v:837)
add_x_48	DW01_inc	width=10	add_851 (NN.v:851)
add_x_50	DW01_inc	width=10	add_867 (NN.v:867)
M1	DW_fp_mult	sig_width=23	M1 (NN.v:103)
		width=8	
		piece.compliance=0	
S0	DW_fp_sub	sig_width=23	S0 (NN.v:104)
		width=8	
		piece.compliance=0	

## Grading Policy

1. Function Validity: 70%
2. Performance: 30 %
  - Area \* Computation time: 30%
  - Computation time = Latency \* clock cycle time

## Block diagram





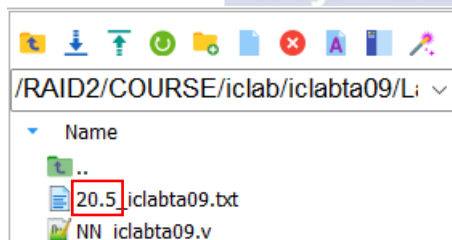
## Note

### 1. Please submit following files under 09\_SUBMIT before 12:00 at noon on March. 20:

- **NN.v**
- If uploaded files **violate the naming rule**, you will get **5 deduct points**.

• In this lab, you can adjust your clock cycle time. **Consequently, make sure to key in your clock cycle time after the command like the figure below**. It's means that the TA will demo your design under this clock cycle time.

```
[iclabta09@ee01 09_SUBMIT]$ ./01_submit 20.5
```



- The 2nd demo deadline is **12:00 at noon on March.22**.
- Check whether there is any wire / reg / submodule name called “error”, “fail”, “pass”, “congratulation”, “latch”, “DW\_fp”, if you used, you will fail the lab.

### 2. Template folders and reference commands:

01\_RTL/ (RTL simulation) **./01\_run**

02\_SYN/ (Synthesis) **./01\_run\_dc**

(Check if there is any **latch** in your design in **syn.log**)

(Check the timing of design in /Report/NN.timing)

03\_GATE / (Gate-level simulation) **./01\_run**

✖ **You should make sure the three clock period values identical in 00\_TESTBED/Pattern.v && /02\_SYN/syn.tcl:**

```

`ifdef RTL
    `include "NN.v"
    `define CYCLE_TIME 50
`endif
`ifdef GATE
    `include "NN_SYN.v"
    `define CYCLE_TIME 50
`endif
  
```

```

#=====
# Global Parameters
#=====
set DESIGN "NN"
set CLK_period 50.0
  
```

## Sample Waveform

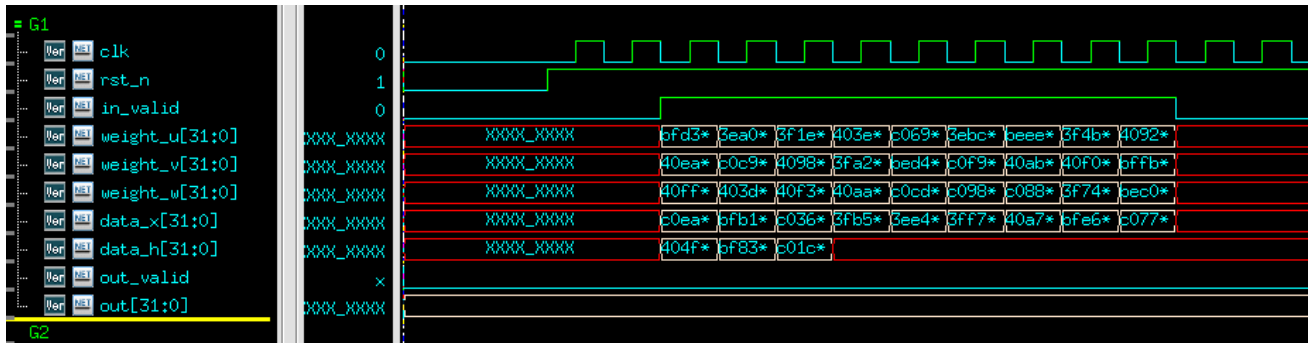


Fig1. Input waveform

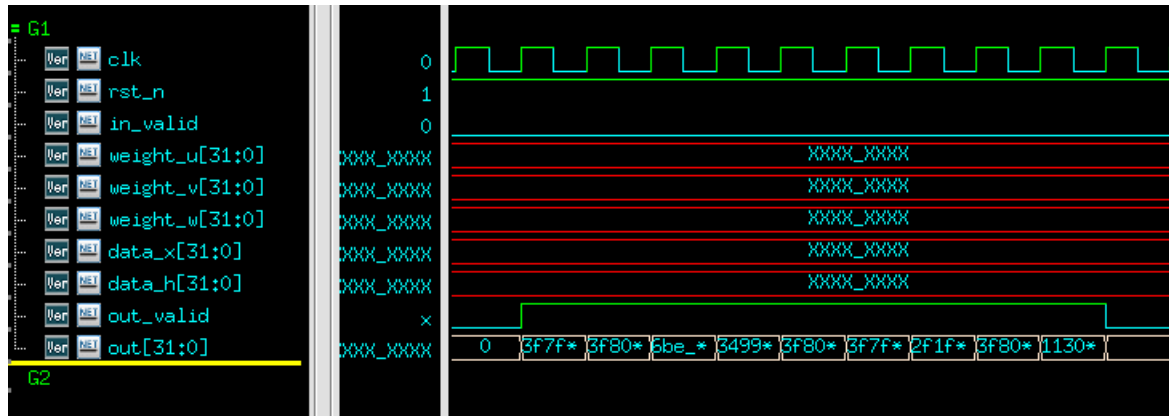


Fig2. Output waveform

