ENEL489 Project Report

Shrey Shah | #200371776 | April 17th

ABSTRACT:

This report contains the details, findings and methodologies used to Implement the artificial neural network (ANN) based Rosseler's Chaotic System Generator on an FPGA via VHDL synthesis. Matlab was used in parallel to generate both weights, datasets and an expected solution to which the VHDL output was measured against.

ENEL489 Project Report

Shrey Shah | #200371776 | April 17th

Table of Contents

Dataset for testing and training

Use Matlab to execute the "rossler_main.m" file and populate the variables. Use the "ntstool" within Matlab to generate a network. We will leverage the "ii" and "oo" as our input and targets respectively.

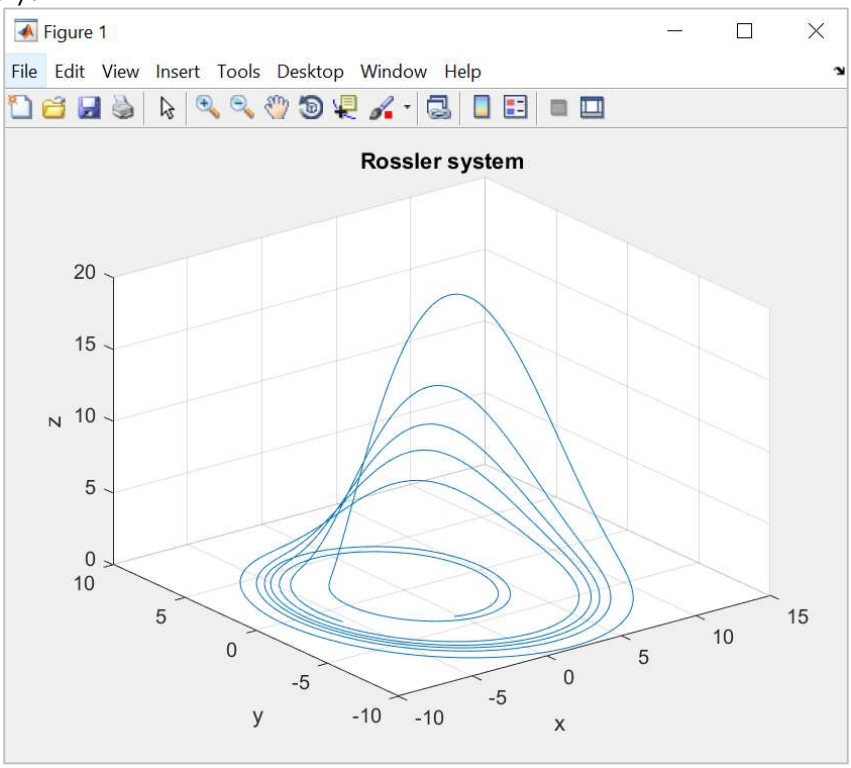
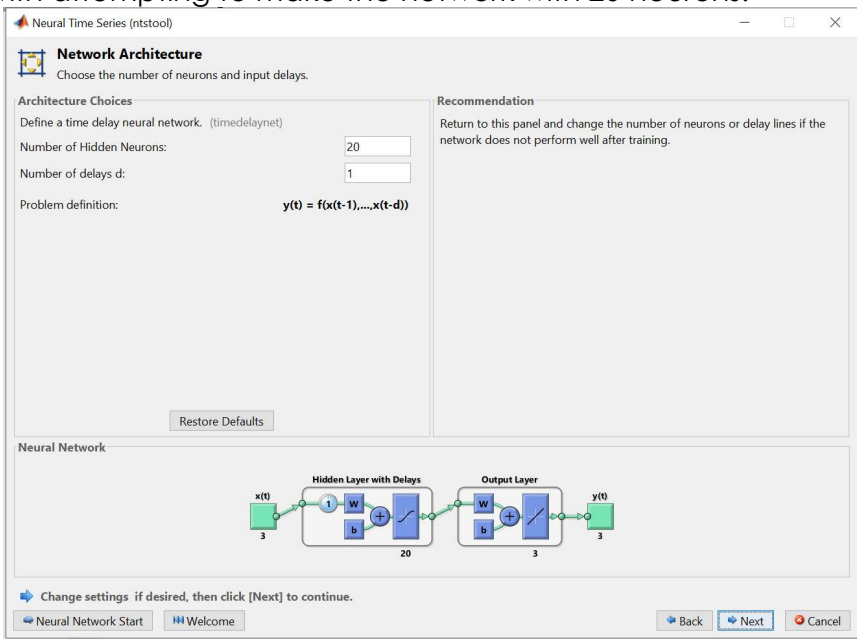


Figure 1 - Original output from Rossler System

We will first start with attempting to make the network with 20 neurons.



After training the network we can see the following iteration menu and mean squared error values.

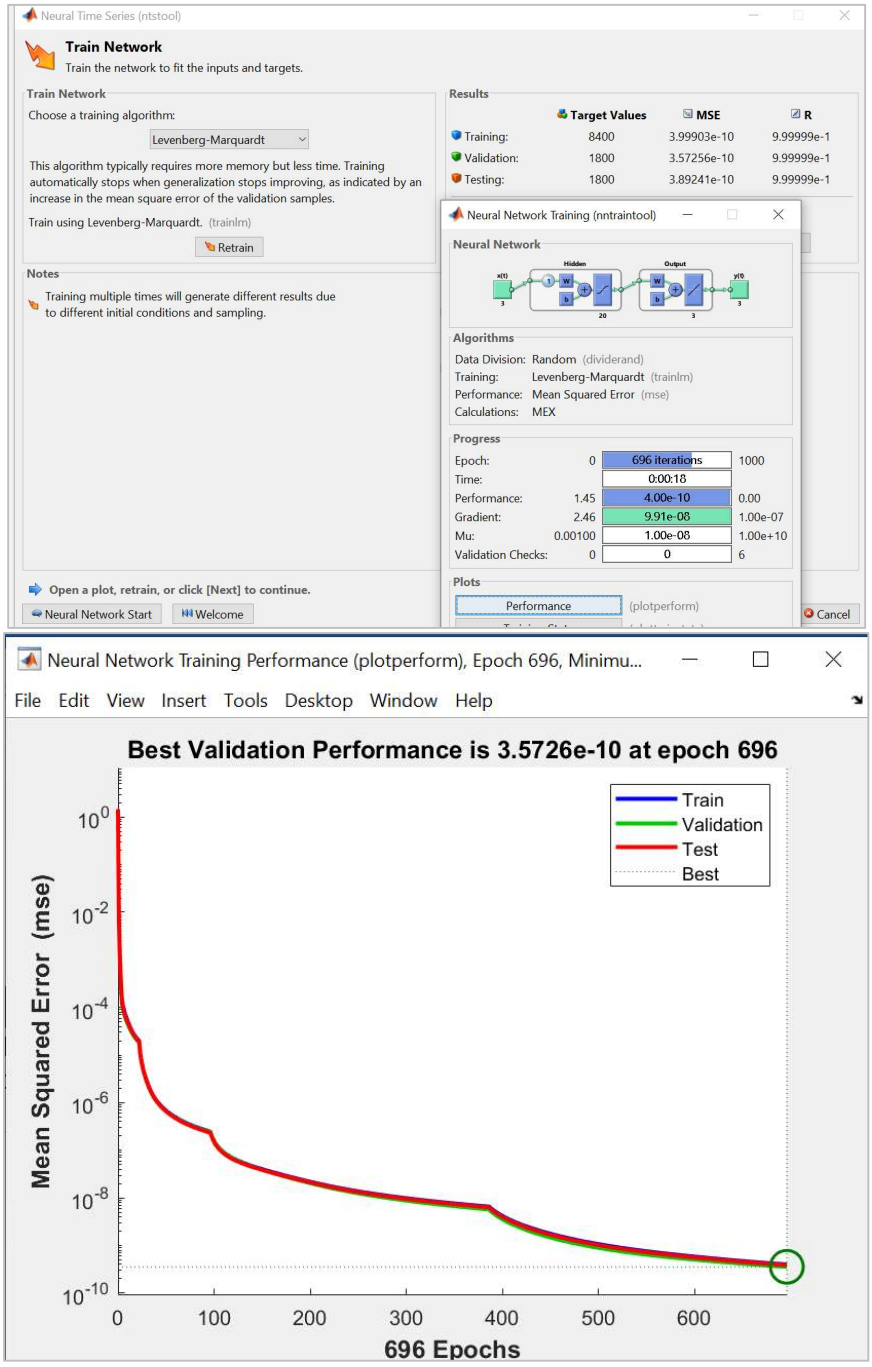
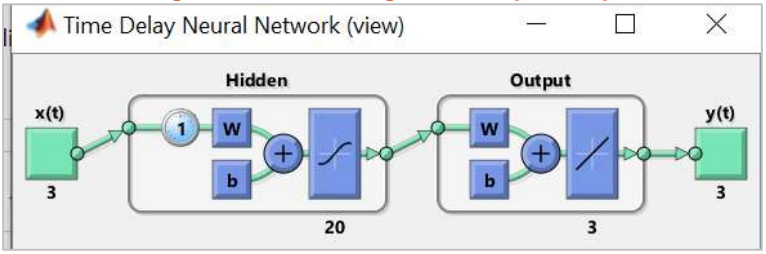


Figure 2 - NN Training and MSE (Matlab)



As we can see, testing with 20 neurons roughly gives us a Mean squared error value less than 10^-9.

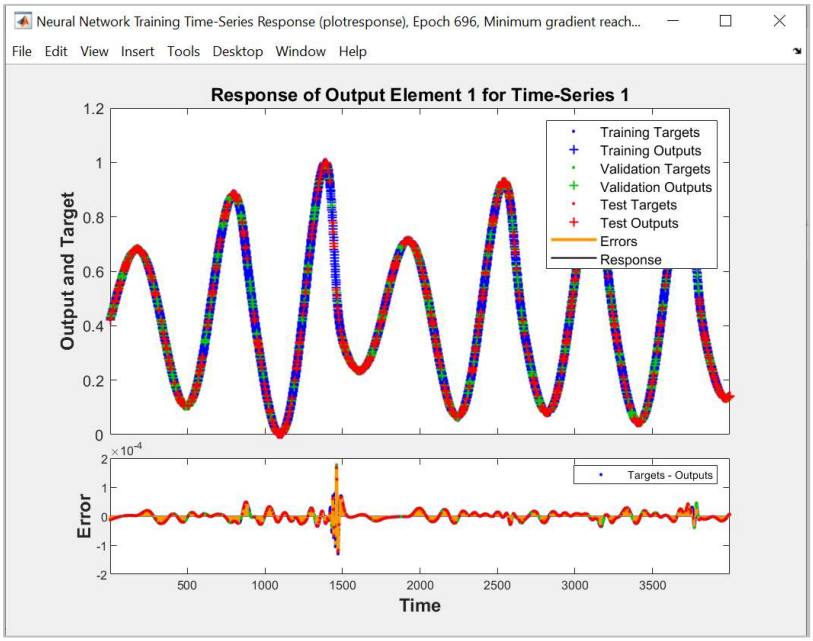


Figure 3 - Time Series Response (Matlab)

Fixed Point Format

In this test I was originally using the Q8.8 format. However, I found a lot of discrepancy in the input values generated from Rosseler function to the hardware input representation. Therefore, I switched to the 14.14 format. And since the all the values are normalized, which is essentially 14 zeroes, followed by the binary representation of the number.

Then in order to make calculations easier, I wrote a python script to convert decimal into fixed point values, by leveraging the "fxpmath" library.

Network Configuration and Hidden Layer Neurons

We chose 20 neurons for our Matlab system and the Vivado system respectively. This is because with 10 the system showed high MSE values. With 20, the Neural Network in Matlab has a MSE of less than 10^-9.

And the Vivado model has a MSE value shown in Figure 7 with calculations on 10 test samples.

RTL Design

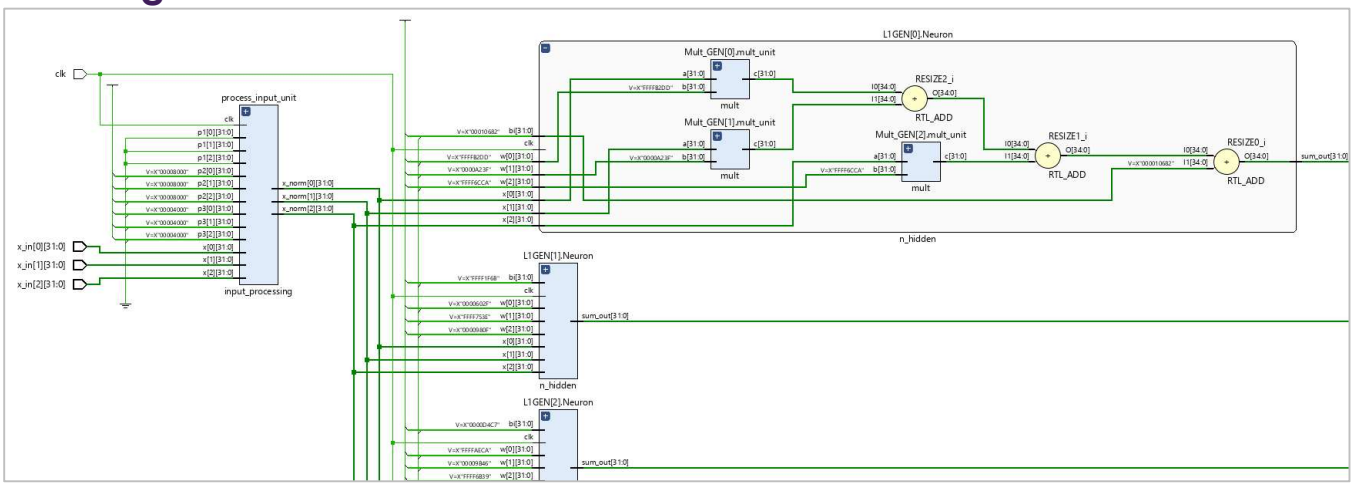


Figure 4 - RTL Design Expanded (Part 1)

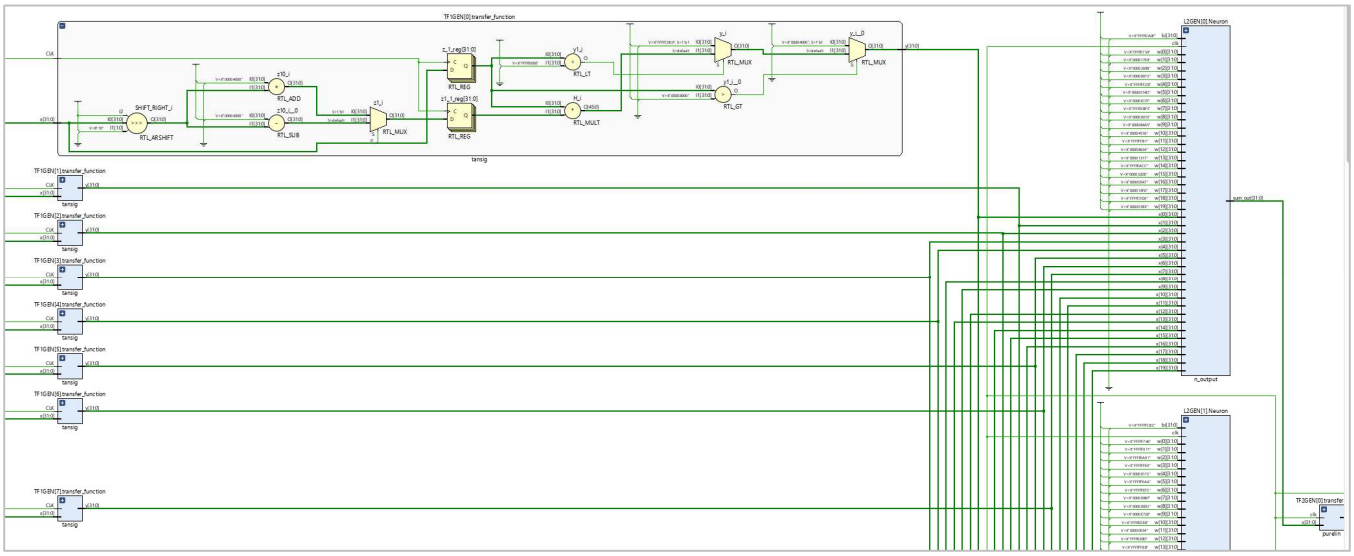


Figure 5 - RTL Design Expanded (Part 2)

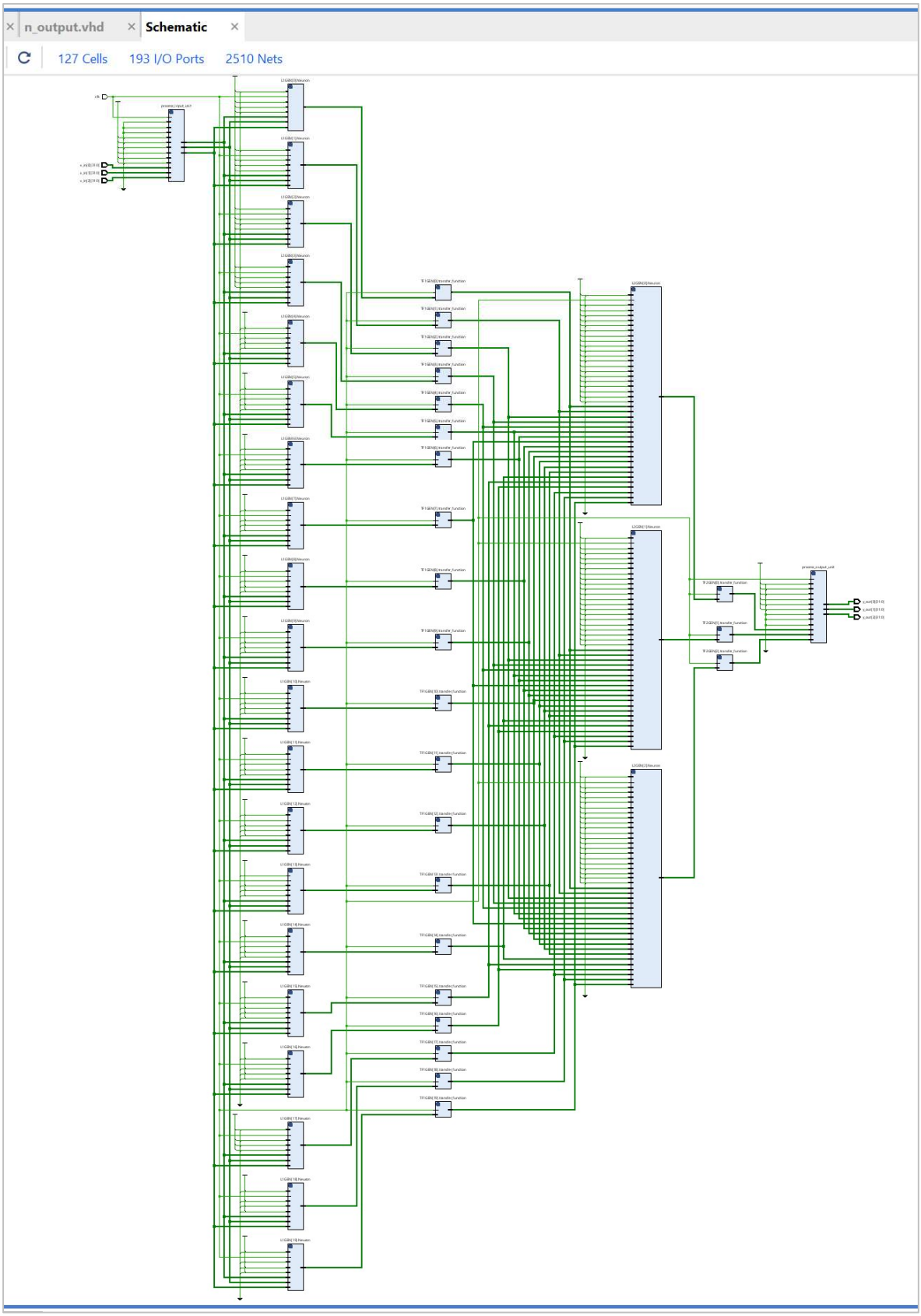


Figure 6 - RTL Design (Full View)

MSE Values

L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC
Test Runs																	
Input			VHDL Output			Expected Output			Raw Error			MSE			Overall MSE Value		
0.882852	0.508471	0.045651	0.876099	0.522278	0.065491	0.883073	0.513318	0.047155	-0.00697	0.00896	0.018335	1.5504E-04			1.3038E-04		
0.123293	0.703119	0.000186	0.130615	0.703674	0.011841	0.122212	0.699858	0.000184	0.008403	0.003816	0.011657	7.3687E-05					
0.782451	0.259461	0.007245	0.783142	0.284058	0.025146	0.785268	0.262686	0.007445	-0.00213	0.021371	0.017701	2.5819E-04					
0.459011	0.762361	0.463299	0.442749	0.768982	0.450562	0.452932	0.762974	0.44034	-0.01018	0.006008	0.010222	8.1419E-05					
0.517221	0.315519	0.001529	0.524536	0.333923	0.016968	0.519592	0.315889	0.001548	0.004944	0.018034	0.01542	1.9581E-04					
0.398016	0.937293	0.001806	0.395935	0.934326	0.01593	0.394826	0.937574	0.001756	0.001109	-0.00325	0.014174	7.0899E-05					
0.331005	0.935716	0.001028	0.330811	0.930908	0.014648	0.327837	0.935244	0.001003	0.002973	-0.00434	0.013646	7.1284E-05					
0.302678	0.930159	0.000825	0.303223	0.924744	0.014282	0.299561	0.92936	0.000806	0.003662	-0.00462	0.013476	7.2109E-05					
0.112153	0.777293	0.000178	0.118958	0.77417	0.011902	0.110409	0.774057	0.000174	0.008549	0.000113	0.011728	7.0210E-05					
0.131395	0.331188	0.000152	0.148254	0.346985	0.012939	0.133639	0.327275	0.000155	0.014615	0.01971	0.012785	2.5518E-04					

Figure 7 - MSE calculations from Vivado Simulation Results

Resource Utilization Report

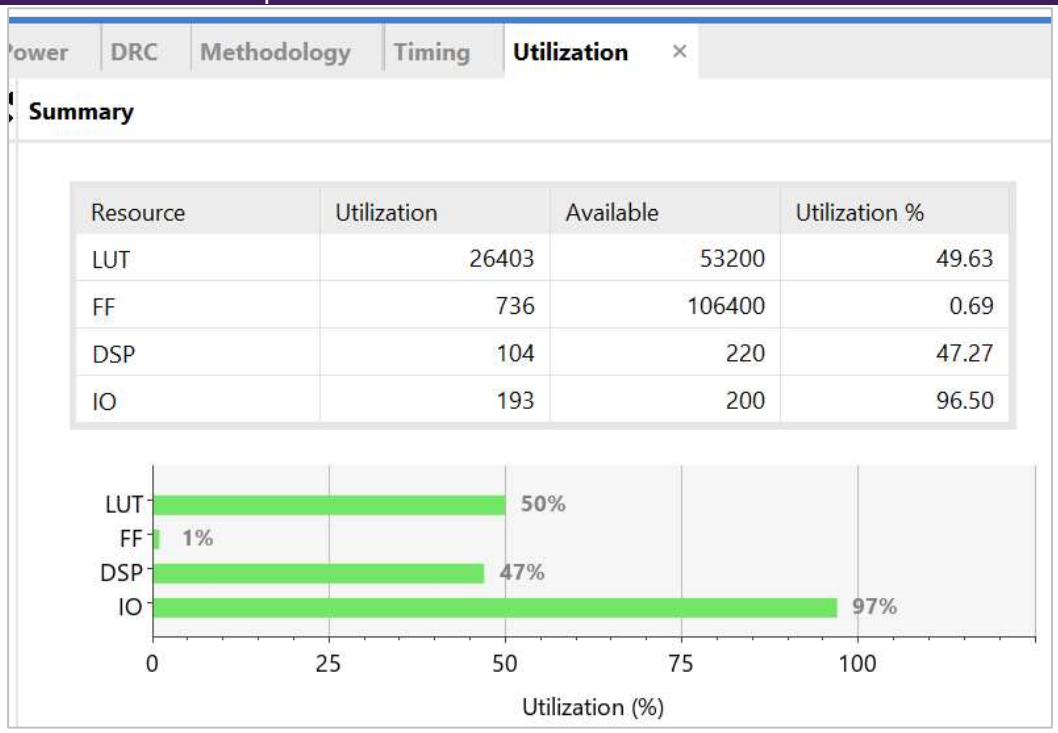


Figure 8 - Resource Utilization Summary

```
Copyright 1986-2020 Xilinx, Inc. All Rights Reserved.
----------------------------------------------------------------------------------------------------
| Tool Version : Vivado v.2020.2 (win64) Build 3064766 Wed Nov 18 09:12:45 MST 2020
| Date         : Mon Apr 19 18:38:02 2021
| Host         : HomePC-SS running 64-bit major release  (build 9200)
| Command      : report_utilization -file nn_utilization_synth.rpt -pb nn_utilizatic
| Design       : nn
| Device       : 7z020clg484-1
| Design State : Synthesized
----------------------------------------------------------------------------------------------------

Utilization Design Information

Table of Contents
-----------------
1. Slice Logic
1.1 Summary of Registers by Type
2. Memory
3. DSP
4. IO and GT Specific
5. Clocking
6. Specific Feature
7. Primitives
8. Black Boxes
9. Instantiated Netlists

1. Slice Logic
--------------

+-------------------------+-------+-------+-----------+-------+
|        Site Type        |  Used | Fixed | Available | Util% |
+-------------------------+-------+-------+-----------+-------+
| Slice LUTs*             | 27288 |     0 |     53200 | 51.29 |
|   LUT as Logic          | 27288 |     0 |     53200 | 51.29 |
|   LUT as Memory         |     0 |     0 |     17400 |  0.00 |
| Slice Registers         |   736 |     0 |    106400 |  0.69 |
|   Register as Flip Flop |   736 |     0 |    106400 |  0.69 |
|   Register as Latch     |     0 |     0 |    106400 |  0.00 |
| F7 Muxes                |     0 |     0 |     26600 |  0.00 |
| F8 Muxes                |     0 |     0 |     13300 |  0.00 |
+-------------------------+-------+-------+-----------+-------+
* Warning! The Final LUT count, after physical optimizations and full implementatior
```

Figure 9- Resource Utilization Report from Synthesis Design

Power | DRC | Methodology | Timing | **Utilization** ×

Hierarchy

Name	Slice LUTs (53200)	Slice Registers (106400)	Slice (13300)	LUT as Logic (53200)	DSPs (220)	Bonded IOB (200)	BUFGCTRL (32)
L1GEN[4].Neuron (n_hidden_13)	638	0	187	638	0	0	0
L1GEN[5].Neuron (n_hidden_14)	678	0	202	678	0	0	0
L1GEN[6].Neuron (n_hidden_15)	632	0	182	632	0	0	0
L1GEN[7].Neuron (n_hidden_16)	582	0	169	582	0	0	0
L1GEN[8].Neuron (n_hidden_17)	597	0	173	597	0	0	0
L1GEN[9].Neuron (n_hidden_18)	459	0	147	459	0	0	0
L1GEN[10].Neuron (n_hidden_0)	448	0	140	448	0	0	0
L1GEN[11].Neuron (n_hidden_1)	720	0	210	720	0	0	0
L1GEN[12].Neuron (n_hidden_2)	582	0	173	582	0	0	0
L1GEN[13].Neuron (n_hidden_3)	825	0	240	825	0	0	0
L1GEN[14].Neuron (n_hidden_4)	565	0	172	565	0	0	0
L1GEN[15].Neuron (n_hidden_5)	697	0	211	697	0	0	0
L1GEN[16].Neuron (n_hidden_6)	518	0	163	518	0	0	0
L1GEN[17].Neuron (n_hidden_7)	732	0	218	732	0	0	0
L1GEN[18].Neuron (n_hidden_8)	530	0	167	530	0	0	0
L1GEN[19].Neuron (n_hidden_9)	637	0	177	637	0	0	0
L2GEN[0].Neuron (n_output)	1939	0	1200	1939	0	0	0
L2GEN[1].Neuron (n_output_19)	1687	0	1053	1687	0	0	0
L2GEN[2].Neuron (n_output_20)	1771	0	1024	1771	0	0	0

Figure 10 - Utilization Hierarchy from Implementation

Design Specifications

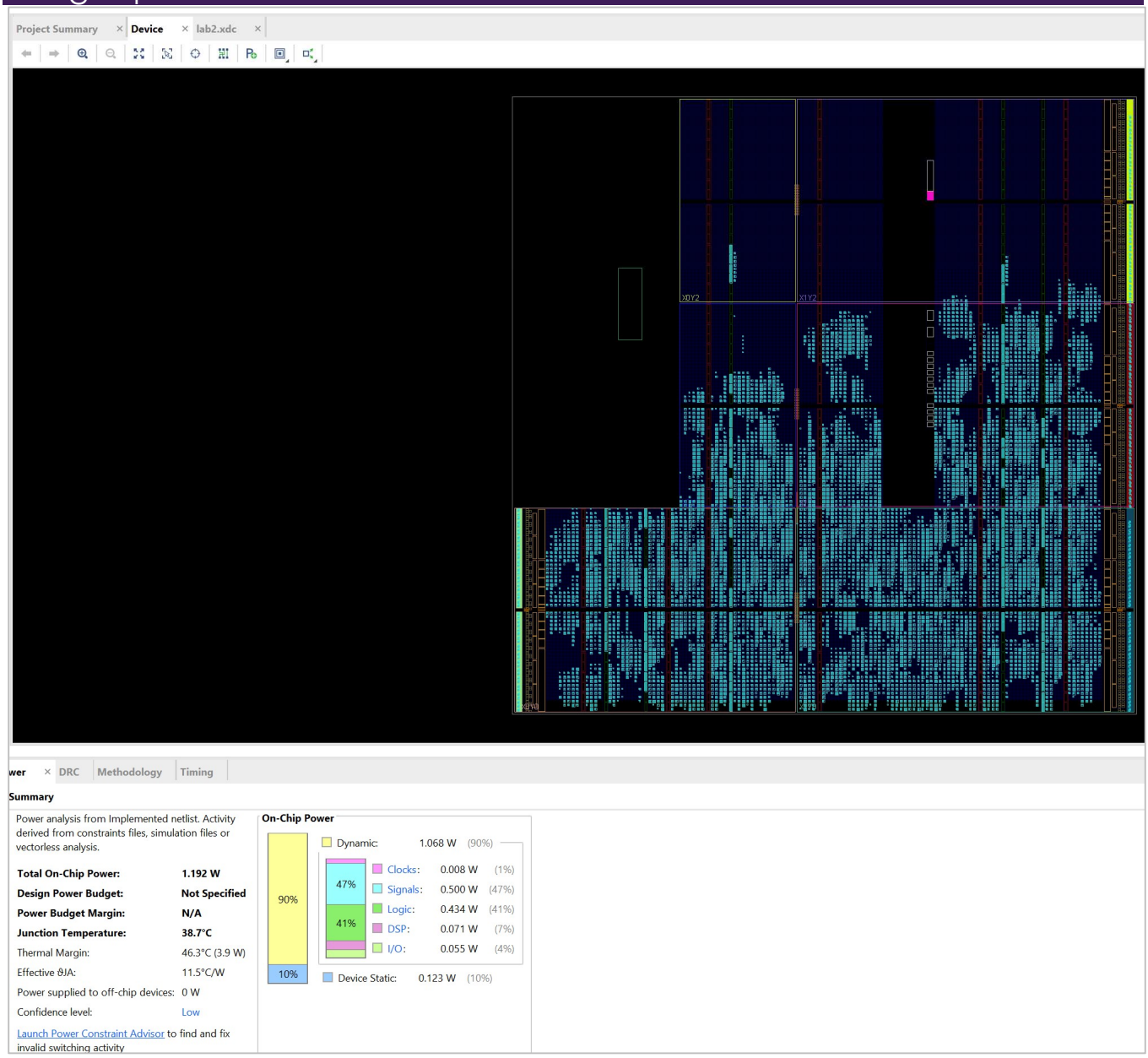


Figure 11 - Implementation View of System

Adding the constraint file allowed us to get a better value for power usage.

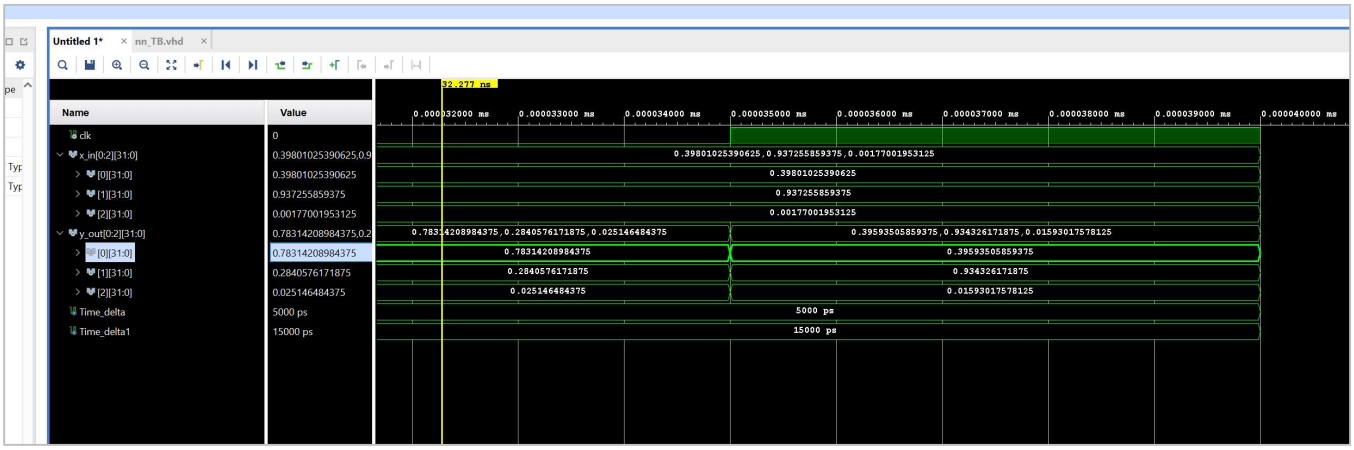


Figure 12 - Simulation Run of the code. Testing in two data sets from the ii data file

Conclusion

Limitations and Final Thoughts

The only limitations made were due to the onset of COVID-19 on the area. Since the university was closed all of the labs had to be remote. Which works well in theory but the information gap is still present. However, due to the great efforts of both our lab instructor Jin Zhang and professor Lei Zhang, I managed to learn an extensive amount of information related to the topic of FPGA system design.

I would still recommend this class to any student who is passionate and interested about hardware analysis and design. I think FPGA is a growing industry with high demand from all sides of the economy. One of the most notable would be the trading firms like Jane Street, Akuna Capital and Two Sigma all currently employ FPGA engineers to increase the processing of data into respective trading algorithms.