Digital IC Functionality Duplication Using Neural Networks

Undergraduate Graduation Project Report Submitted in Partial Fulfillment of
the Requirements for the
Degree of Bachelor of Science of Engineering
in
The Department of Electronic & Telecommunication Engineering
University of Moratuwa

Supervisor:
Dr. S.Thayaparan

Group Members:
H.M.A.M.Bandara(180060G)
Y.A A.W.Rajakaruna(180508N)
T.M.S.Nayanajith(180417J)
P.S.C.Jayapala(180265N)

July 2023

Acknowledgments

We would like to express our sincere gratitude to our supervisor, Dr. Thayaparan, for providing continuous guidance throughout the project and for helping us develop our knowledge and skills. We are also grateful to Dr. Ranga Rodrigo for the valuable advice provided.

Furthermore, we are indebted to the staff and our department for providing us with the necessary resources and facilities, which played a significant role in the smooth execution of this project.

In conclusion, this project has been a remarkable learning experience, and we are grateful to everyone who played a part in its success.

Table of Contents

List of Figures

List of Tables

Acronyms and Abbreviations

IC Integrated Circuit

NN Neural Network

GA Genetic Algorithm

RE Reverse Engineering

ANN Artificial Neural Network

LNN Logical Neural Network

RNN Recurrent Neural Networks

DFF D-type Flip Flop

PWA Programmable Wavelet Array

ESD Electrostatic Discharge

PCB Printed Circuit Board

ABC Artificial Bee Colony

PSO Particle Swarm Optimization

ETL Extract, Transform and Load

IoT Internet of Things

RTL Register Transfer Level

SGD Stochastic Gradient Descent

GRU Gated Recurrent Unit

NLP Natural Language Processing

FPGA Field Programmable Gate Array

LSTM Long Short Term Memory

mSGA modification of Simple Genetic Algorithm

SIPO Serial Input Parallel Output

PISO Parallel Input Serial Output

LFSR Linear Feedback Shift Register

HTTP Hypertext Transfer Protocol

UART Universal Asynchronous Receiver/Transmitter

1 Introduction

This chapter provides an overview of the key elements that will be discussed in detail throughout this project. We begin by presenting the problem statement, which outlines the core challenge addressed by this project. Subsequently, the main objectives are laid out, along with a detailed plan on how these objectives will be achieved. Furthermore, we delve into the project scope to define the boundaries and limitations within which our research and development efforts are focused. The chapter then highlights the novelty and uniqueness of our project, showcasing its innovative contributions to the field. Additionally, we explore the potential applications that arise from our work, demonstrating its practical implications and real-world relevance. Lastly, to facilitate navigation and comprehension, a clear roadmap to the subsequent chapters is provided. By structuring this chapter in a coherent and organized manner, we aim to set the stage for a comprehensive understanding of the research undertaken and its significant outcomes.

1.1 The Problem Statement

Integrated circuits(ICs) are susceptible to failure while in use, just like all other circuit components. Manufacturing flaws, physical damage to the packaging, failures due to operating conditions (e.g., extreme operating temperature conditions), and electrostatic discharge (ESD) are the most common reasons for IC failure [1]. If an IC malfunctions, the desired functionality of the entire system will change, which may ultimately result in a total or partial system failure. Sometimes, it can lead to even more catastrophic damage if the faulty IC is not located and replaced as soon as possible.

Troubleshooting ICs can be challenging, especially if they have already been mounted on a Printed Circuit Board (PCB). The following methods at least allow you to fairly determine whether an IC needs to be replaced because it is faulty:

- Perform a Visual Inspection
- Check for Short-circuiting
- Test the IC Temperature
- Check the IC Voltage or Signal Waveforms

But in order to carry out any of the procedures listed above, you either need to have strong technical knowledge or support from well-qualified technicians.

Our project's objective is to develop a technique that will enable anyone, regardless of technical proficiency, to test any digital IC on a PCB circuit board for maintenance and debugging purposes. For that, we propose a failed IC localization method as a corrective maintenance mechanism. Typically, when an IC failure occurs, the system's supposed

operating modes may be altered, or an operational failure may occur. In such cases, our method can be used to identify whether the cause is an IC defect without using existing tech-savvy methods. To use our debugging approach, neither precise IC specifications nor a high level of technical expertise are necessary.

If we break down the procedure of our method it can be listed as follows:

- First, duplicate the IC functionality of a properly functioning digital IC using our mechanism. Using our proposed NN based IC function duplicator, any digital IC function can be duplicated by observing its inputs and outputs without knowing its exact specifications.
- The original digital IC can then be compared with this duplicate model during routine use or in the event of a system failure to determine whether the fault is caused by the chosen specific IC or not. This means that basically, our model serves as a reference to compare with the original IC over time or use.

1.2 Main Objectives

- Detect the IC functionality by observing the output for given inputs or by the relationship between input and output.
- Develop modeling techniques for the digital functions using NN.
- Carry out hardware implementation, and demonstrate the feasibility in a real-world application.
- Research on optimizing NN algorithms for digital functionality and do relevant publications.

1.3 Project Scope

The scope of this project can be divided into three parts. Although the primary domain revolves around neural networks (NNs), the project also encompasses hardware implementation and testing, extending its reach to include FPGA as well.

1. NN Implementation Using a High-Level Language
2. Testing, Verification, and Optimization
3. Hardware Implementation

In the first stage, we created a model for combinational circuits. We employed feed forward NNs for combinational circuits since their outputs were fixed functions of their inputs. Then, we moved on to the sequential circuits. Finally, we developed a single common re-configurable model. There was no fixed functionality between inputs and

outputs in sequential circuits. They contained feedback loops, and they had to have memory to store the data from previous iterations. In general, flip-flops were used to accomplish this in hardware. We used feedback (recurrent) NNs as the NN type for this. However, when training for longer sequences, the basic Recurrent Neural Network (RNN) suffered from vanishing gradient. So we had to go with advanced RNN versions such as LSTM, which could overcome most of the issues in basic RNNs. The final common model is then available for re-configuration as any combinational or sequential circuit.

At the first stage, the testing was done using the pre-created data sets. Data sets were created using Register Transfer Level (RTL) design software to create various logic circuits. When those circuits simulated with random input patterns, data sets were generated. For circuits, we had selected simple combinational circuits and then moved to sequential circuits such as counters, SIPO (Serial Input Parallel Output) shift registers, PISO (Parallel Input Serial Output) shift registers, and LFSR (Linear Feedback Shift Register) with up to 16 memory elements.

At the second stage, we implemented digital circuits on an FPGA and captured and transmitted its inputs and outputs data via UART (universal asynchronous receiver-transmitter) to obtain the training data. We conducted the training with real-time I/Os until the NN model converged. Additionally, we considered working on some optimization methods to improve training performance and reduce computation time.

Then, for hardware implementation, we used FPGA boards. We selected a model for a simple real-world scenario, which was a robot arm, and then successfully mimicked its functionality of the digital logic part to demonstrate the capability of our NN engine.

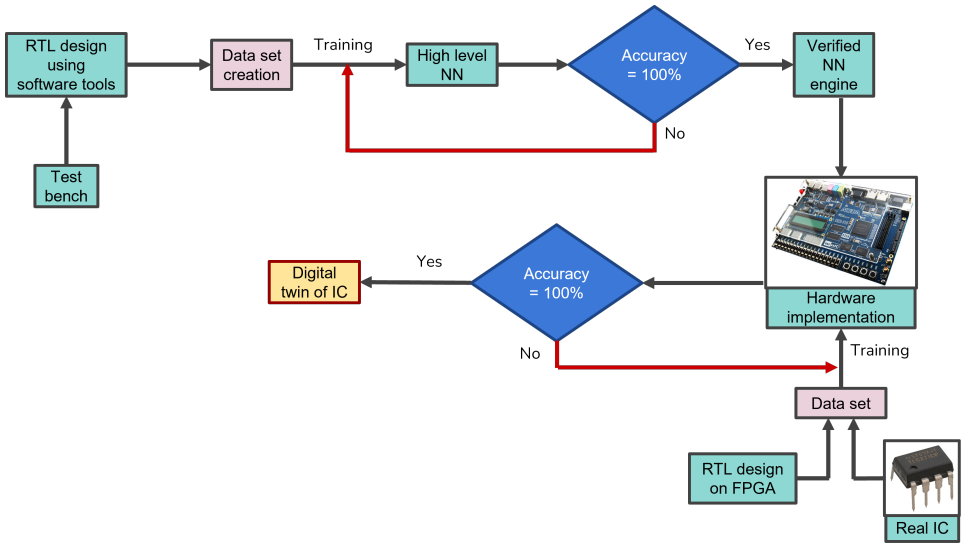


Figure 1: Project Scope

1.4 Novelty and the Uniqueness of the Project

We intended to develop an NN engine that could replicate the functionality of a digital integrated circuit (IC) by simply observing its inputs and outputs. The digital IC was expected to be on a PCB, and you could easily transport this hardware implementation to the site, where the required inputs and outputs were extracted using the tapping method. After successfully duplicating the functionality of a properly functioning IC, this could be used for a variety of purposes.

When compared to other existing methodologies, our methodology stands out for a variety of reasons. Some of them are listed below.

- Sequential Logic Functionality Duplication:
  - In terms of functionality duplication using NNs, many projects and studies were conducted previously related to combinational logic functions. However, in terms of sequential logic duplication, our method is the most comprehensive.
- Simple IC Debugging Mechanism:
  - At the moment, existing IC debugging methodologies (e.g., performing a visual inspection, checking for short-circuiting, testing the IC temperature, and checking the IC voltage or signal waveforms) are technically complex to use and will require specially trained personnel to perform. However, our method is simple enough that anyone with no prior technical knowledge can use it for IC debugging.

1.5 Potential Applications

The following are some of the potential beneficiaries of our proposed engine.

- Industrial Plants:
  - In industrial plants, there exist critical digital ICs that operate continuously. Even a minor breakdown in these ICs can result in significant losses to company profits. Therefore, our NN engine offers a solution by duplicating the functionality of these vital digital ICs as digital twins. In the event of a breakdown, these digital twins can seamlessly take over and continue operating until the replacement arrives or the necessary repairs are completed, ensuring uninterrupted productivity and minimizing downtime.
- Fault Diagnosis of Logic Circuits:
  - Initially, a digital twin is created using our NN engine by employing a 100% fault-free IC. Subsequently, this digital twin becomes a powerful tool for testing ICs and determining if they are functioning correctly. The process is user-friendly and

doesn't demand extensive technical expertise, making it easily accessible for anyone to carry out.

- IC Reverse Engineers:
  - Due to vendor competition on functionally equivalent devices, the digital IC Reverse Engineering (RE) has gained significant value in the modern product market. Engineers redraw integrated circuits and duplicate the same functionality on their products using various techniques and tools. The current method involves visually identifying the system using magnification, deriving functionality, and duplicating the design. So, instead of going through all of that trouble, they can use our mechanism to duplicate the digital IC functionality.
- IC Manufacturers:
  - Sometimes an IC on a PCB is discontinued and its specifications are unavailable. If it is necessary to manufacture a similarly functionating IC at this time, the IC manufacturers have to go to great lengths to extract the exact functionality of that IC. At a time like this, IC manufacturers can use our proposed method to duplicate the functionality of that IC and create a digital twin, and then proceed with production of a similarly functioning new IC.

1.6 Navigation to the Chapters

The rest of the report is organized into several chapters, each focusing on specific aspects of the project. The literature review is presented in Chapter 2, which provides an overview of prior research works and studies related to the project scope. Chapter 3 delves into the methodology, covering various topics such as the system architecture, hardware implementation, end-to-end system development, analysis of alternatives, and risks and risk management plan. The subsections under risks and risk Management Plan, including financial risks and technical risks, are discussed in detail, along with task delegation and time line for project management.

Chapter 4 presents the Results obtained from the project's experimentation and implementation. It includes the process of dataset creation, followed by detailed insights into the 16-Bit Counter, 16-Bit SIPO shift register, and 16-Bit PISO shift register functionalities. Furthermore, the demonstration section showcases the successful application of the project in real-world scenarios, such as controlling a Robot Arm and duplicating the functionality of a 16-Bit LFSR circuit.

In Chapter 5, the discussion and conclusions are presented. The main findings of the literature review are summarized, highlighting the significance of Recurrent Neural Networks (RNN), specifically the Basic RNN, GRU, and LSTM. The Impact of the Project is discussed, covering its local, global, and potential industrial implications. The chap-

ter concludes with key takeaways and future works, presenting opportunities for further development and enhancement of the project's capabilities.

Overall, the report provides a comprehensive exploration of the project's objectives, methodologies, results, and potential impacts, showcasing the successful application of LSTM and neural networks in digital circuit functionality duplication and presenting potential avenues for future advancements.

2 Literature Review

Reverse engineering (RE) is a critical yet time-consuming process that often requires costly specialized hardware and software [2]. Optical techniques have been used to gain a better understanding of an IC's internal workings, with practical applications in industries such as electronic manufacturing, security analysis, and intellectual property protection. In their study, Randy et al. [2] provide an overview of RE's role in the semiconductor industry, including methods such as product teardowns, system-level analysis, process analysis, and circuit extraction.

Neural networks are known for their ability to quickly process information by functioning in parallel and distributedly through a network of simple cells. Several studies have suggested methods for designing high-performance digital circuits using NNs, including Hopfield NNs for multipliers, AND gates, full adders, D-Latches with 4-bit shift registers, 4-bit asynchronous counters, and other devices [3, 4, 5].

Neural networks have also been used to simulate digital circuits and present their behaviour as software. For instance, [6] discusses how to convert digital circuits used for signal processing applications into their NN equivalents. A combinational neural-logic system that uses a Genetic Algorithm (GA) to train a network consisting primarily of neural-logic AND, OR, and NOT gates has been described in [7]. This system can be designed methodically to incorporate the characteristics of the application into the network structure to improve training and network performance. The authors of [8] suggested using a logical neural network (LNN) to design and optimize combinational logic circuits. The LNN is created and optimized for a number of truth tables using a modification of the Simple Genetic Algorithm (mSGA), where the weights and neurons of the NN represent the connections and logic operations of the digital circuit, respectively.

A wavelet network with discrete weights has been proposed in [9] to suggest a universal digital sequential circuit. The results demonstrate that the proposed circuit, which uses a Programmable Wavelet Array (PWA), can approximate the majority of general sequential logic functions. Additionally, a novel 4-bit negative edge triggered binary synchronous up/down counter using Artificial Neural Networks (ANNs) trained with hybrid algorithms is proposed in [10]. This device, which is constructed exclusively from logic gates and flip-flops, has a fast convergence rate and is less error-prone.

A method for designing neural networks that simulate sequential switching circuits using multilayer perceptrons and neural memory components has been proposed in [11]. This approach uses a single hidden layer trained using the Extract, Transform, and Load (ETL) algorithm to implement the combinational circuit and neural storage units to store internal states, resulting in a network consisting solely of neural elements. ANNs have

also been employed to model electronic circuits in the time domain using the black-box concept, as discussed in [12]. Furthermore, a new method of combinational logic circuit fault diagnosis has been presented in [13] by presenting a circuit as a diagnostic engine with its equivalent NN circuit, which takes into account the fault testability of the NN model and does not require preparation of test patterns or a fault dictionary.

In summary, the literature review has provided valuable insights into the use of neural networks (NNs) in the context of digital circuits. Reverse engineering (RE) methods have been explored for gaining a deeper understanding of ICs' internal workings, but they often involve time-consuming processes and specialized tools. However, NNs offer a promising alternative with their ability to quickly process information in parallel. Several studies have demonstrated the successful application of NNs in designing high-performance digital circuits, including multipliers, adders, shift registers, counters, and other devices. Additionally, NNs have been employed to simulate and model digital circuits, presenting their behavior as software, which can be useful for various signal processing applications. Novel methods and algorithms, such as genetic algorithms and wavelet networks, have been proposed to optimize and approximate sequential logic functions effectively. Moreover, the use of neural memory components has enabled NNs to simulate sequential switching circuits with improved efficiency. These findings highlight the potential of NNs as a powerful tool for digital circuit functionality duplication, offering an efficient and effective approach to address the challenges posed by traditional reverse engineering methods. By harnessing the capabilities of NNs, our project aims to contribute to the advancement of digital circuit design and replication, fostering innovative solutions with broader implications in various industries.

3 Methodology

This chapter delves into the core components that form the backbone of our project's execution. We commence by presenting the system architecture, providing an in-depth exploration of its design and functionalities. Subsequently, alternative methodologies for achieving the project's objectives are analyzed, allowing for a comprehensive comparison to determine the most effective approach.

To ensure a systematic and proactive approach, the chapter then addresses the identification and management of potential risks through a well-defined risk management plan. Additionally, we discuss the task delegation process, outlining the responsibilities assigned to each team member and fostering a clear understanding of their roles.

Finally, a detailed timeline is presented, offering a strategic overview of the project's schedule and milestones. By introducing the subject and elaborating on each section within this chapter, we establish a solid framework for successfully executing the project.

3.1 System Architecture

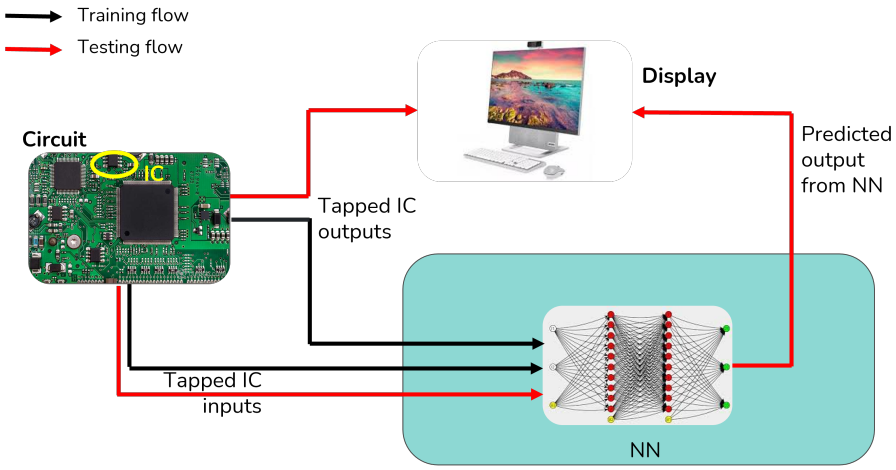


Figure 2: Architecture of the Project

The figure above depicts the final level of this project's basic architecture. For ease of understanding, this can be divided into two stages.

- Training Stage: This is indicated by the blue color flow. A training data set will be created by tapping the IC inputs and outputs on the PCB board. The NN engine will be trained over a long period of time.
- Testing Stage: This flow is indicated by the red color. Since the training is over, we now have an engine with duplicate IC functionality. For testing purposes, tapped IC

outputs are compared to NN predicted outputs. These predicted outputs are generated using Tapped IC inputs.

3.2 Hardware Implementation

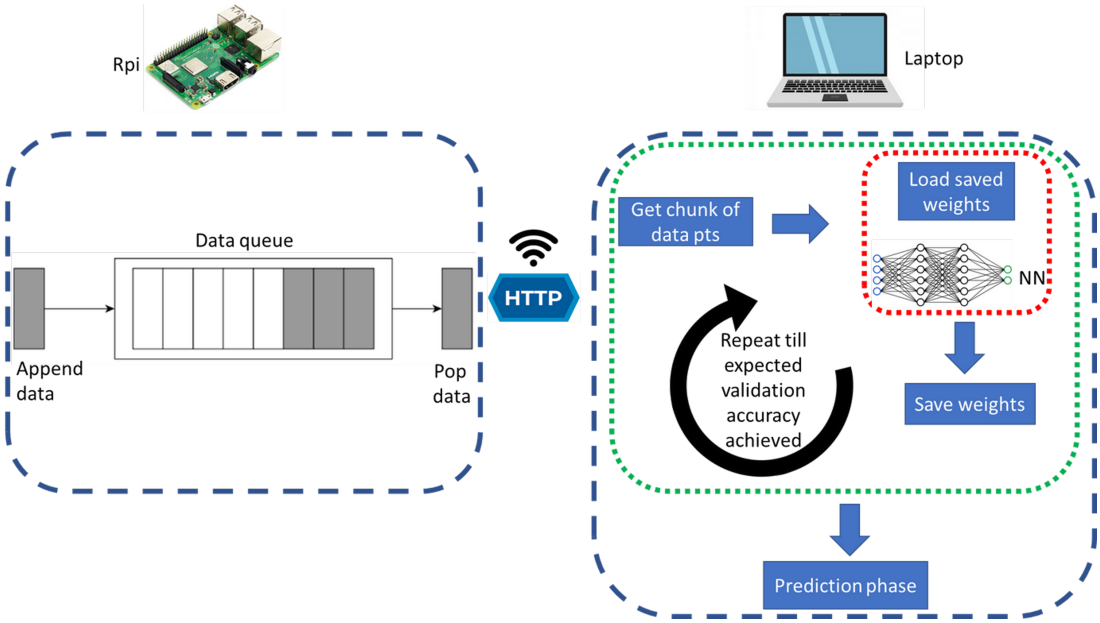


Figure 3: Simplified Hardware Implementation

We began the hardware implementation by designing a digital circuit on an FPGA and capturing its input and output data to a FIFO, also implemented on the same FPGA. These data were then transmitted to a Raspberry Pi via UART for further processing. Within the Raspberry Pi, we optimized data handling by implementing a multithreaded queue, ensuring efficient data appending and popping. Subsequently, the data were transmitted via HTTP to the laptop, where the neural network engine lies, maintaining a seamless data flow without any loss.

One of the key advantages of our system is the flexibility it offers, enabled by using HTTP for communication between the Raspberry Pi and the laptop. Upon receiving data, the neural network engine is trained and the system is locked, smoothly transitioning to the prediction phase after successfully capturing the functionality of the digital logic/circuit.

Highlighted key features of our hardware implementation include:

- Continuous data feed to the Raspberry Pi, ensuring a steady stream of information for processing.
- Efficient transmission of data from the Raspberry Pi to the laptop, facilitated by HTTP communication.

- Utilization of multithreading and data popping, optimizing data handling within the Raspberry Pi.
- Neural Network processing on the laptop, enabling powerful and accurate analysis of the captured data.

With these features, our hardware implementation delivers a robust and effective system for seamless data transfer and accurate neural network processing (Figure 3).

3.3 End-to-end system development

The development of an end-to-end system for duplicating the functionality of digital ICs using a neural network approach involves several key stages. Initially, a collection of digital circuits with observable and measurable input-output relationships is identified. This dataset of inputs and outputs is then gathered, preprocessed, rearranged, and split into training and validation sets. The chosen neural network architecture, LSTM (Long Short-Term Memory), is then implemented due to its superior performance in handling long-term dependencies. The NN is trained using the rearranged data, and hyperparameters are fine-tuned based on accuracy and loss.

The main steps in the process are as follows:

1. Identify the digital circuits to replicate: Selecting digital circuits with distinct input-output relationships is crucial. For this project, three digital sequential circuits have been chosen: Counter, serial input-parallel output (SIPO) register, and parallel input-serial output (PISO) register.
2. Collect data: Data on the inputs and outputs of the identified circuits is gathered using Quartus II 14.1 and ModelSim-Altera 10.3c. The circuits are simulated, and the obtained data is used for training the NN.
3. Preprocess the data: The datasets are rearranged to meet the requirements of the LSTM layer of the NN. Data is concatenated with previous outputs for specified time steps to create three-dimensional inputs (Figure 4).

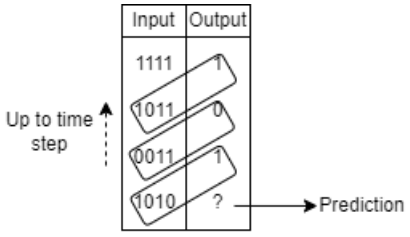


Figure 4: Data Set Prepearation

4. Splitting the data: The data is divided into training and validation sets to fine-tune hyperparameters and avoid overfitting.

5. Design the LSTM neural network architecture: The LSTM NN architecture is designed, considering the complexity of the circuit replication and modifying the architecture as necessary.
6. Train the neural network: The NN is trained using the preprocessed data, and validation accuracy and loss are monitored. Adjustments to hyperparameters are made to optimize performance.
7. Evaluate the network's performance: The trained neural network's performance is assessed using a different set of data to evaluate its capability in replicating digital circuits.
8. Hyperparameter tuning: Based on the evaluation results, hyperparameters are adjusted, or the NN architecture is modified to improve performance.
9. Develop a common neural network model: After tuning the NN and ensuring satisfactory performance, a generic neural network model is created. This model can be used to train various digital circuits, replicating their operation effectively.
10. Evaluate and refine the common neural network model: The accuracy and adaptability of the common NN model are assessed and enhanced to achieve optimal results.

By following this systematic development approach, we aim to successfully duplicate the functionality of diverse digital circuits, making our system versatile and effective in its application.

3.4 Analysis of Alternatives

Due to the high market competency in the electronic component business, vendors tend to reverse engineer benchmark products from competing vendors in order to build their own advanced products. Engineers use various techniques and tools to "redraw" integrated circuits and duplicate functionality on their products.

There are several approaches to RE integrated circuits [2]:

- Product Tear-downs: identify the product, package, internal boards, and components.
- System Level Analysis: analyze operations, functions, timing, signal paths, and interconnections.
- Process Analysis: examine the structure and materials to see how it is manufactured, and what it is made of.
- Circuit Extraction: delayer to transistor level, then extract interconnections and components to create schematics and netlists

The traditional RE process relies on optical imaging of the structure and recreating the design via a highly complex engineering process. Advanced RE labs currently use two types of imaging, optical and scanning electron microscope (SEM).

Although the above RE techniques are very powerful, they are overkill in every aspect when it comes to duplicating only the functionality of an IC.

Table 1: Comparsion Between Traditional RE and the Proposed Technique

Property	Traditional RE Techniques	IC Functionality Duplication
Complexity	High	Low
Time Consumption	High	Low
Cost	High	Low
Level of Expertise Required	High	Low

3.5 Risks and Risk Management Plan

In this section, we outline the potential risks that could impact the successful implementation of our project and present a comprehensive risk management plan to address and mitigate these challenges. We categorize the risks into two main areas: Financial Risks and Technical Risks. Under Financial Risks, we discuss the limitations of the chosen Raspberry Pi and its potential impact on data transfer speeds and system performance. We propose the utilization of more powerful development boards as an effective solution. In the Technical Risks segment, we address concerns related to our NN engine's capacity and the tapping method's ability to capture the full functionality of digital ICs. To overcome these risks, we propose enhancements to our NN engine and extended training periods. By anticipating and managing these risks proactively, we aim to ensure the successful execution of our project with optimal efficiency and accuracy.

3.5.1 Financial Risks

- The processing power of existing Raspberry Pi is limited:
  - The financial risks associated with the hardware implementation primarily revolve around the processing power limitations of the chosen Raspberry Pi (specifically, the Raspberry Pi 3b+) and its potential impact on the performance of the multithreading mechanism, particularly due to the use of HTTP. Given the limited processing power of the Raspberry Pi, issues may arise, including suboptimal data transferring speeds and bottlenecks during operation. These limitations could potentially affect the overall efficiency and responsiveness of the system, leading to compromised performance and increased processing times.

  - To mitigate these financial risks, we should consider investing in more powerful development boards as alternatives to the Raspberry Pi for hardware implementation. By opting for higher-performance development boards, we can significantly enhance the system's processing capabilities and reduce the likelihood of performance-related issues. This strategic investment in more robust hardware will result in a more reliable and efficient system, meeting the processing demands and providing a smoother user experience.

- Cost of Neural Network Training and Optimization:
  - Training and optimizing neural networks can be computationally intensive and time-consuming tasks. As we aim to develop a sophisticated neural network model for digital circuit functionality duplication, the cost of running multiple iterations of training and optimization processes may add up significantly. The expenses can arise from the need to acquire high-performance computing resources, such as powerful GPUs or cloud computing services, to expedite the training process and reduce turnaround time. Additionally, the utilization of large datasets for training purposes may require additional storage resources, further contributing to the overall cost.

  - To address this financial risk, we should carefully assess the trade-offs between computational performance and cost. Exploring cost-effective solutions, such as utilizing cloud-based machine learning platforms that offer scalable resources, can help optimize expenses while maintaining high-quality training outcomes. Furthermore, implementing efficient training algorithms and strategies, such as transfer learning or model compression, can reduce the computational load and minimize training costs. By adopting cost-conscious approaches and resource optimization techniques, we can ensure that the project remains financially viable and achieves its objectives within the allocated budget.

3.5.2 Technical Risks

- ICs with higher memory elements will not work with our NN engine:
  - Currently, our NN engine is under development to support digital ICs with up to 16 memory elements. However, this poses a technical risk as digital integrated circuits that exceed our target complexity may not be compatible with our NN engine.

  - To address this concern, we must focus on enhancing our NN engine's capabilities to support digital ICs with more memory elements. By expanding its capacity, we can ensure compatibility with a broader range of ICs, making our system more versatile and applicable to a wider variety of projects.

- The tapping method will not duplicate the IC's full functionality:

- One of the challenges we face is using the tapping method to obtain the inputs and outputs of a digital IC present on a PCB. During the training of our NN engine, we cannot guarantee that the IC is fully operational, leading to potential discrepancies in its functionality. Such variations can result in errors and impact the accuracy of our system's predictions.

- To mitigate this risk, we plan to run the training procedure of our NN engine for an extended period, allowing it to capture a more comprehensive understanding of the IC's behavior. This approach aims to increase the likelihood of accurately duplicating the IC's full functionality and mitigating potential issues that might arise during prediction.

- Potential Bottleneck with HTTP for Data Transfer:
  - In the previous section, we mentioned that HTTP is used for transmitting data between the Raspberry Pi and the laptop. While HTTP offers flexibility and ease of implementation, it may introduce a potential bottleneck in data transfer speeds, especially considering the limited processing power of the Raspberry Pi. This bottleneck could result in slower data transmission rates, affecting the overall performance of the system during multithreading.

  - To address this concern, we may explore alternative data transfer protocols or optimize the existing HTTP implementation to enhance data transfer efficiency and minimize potential delays.

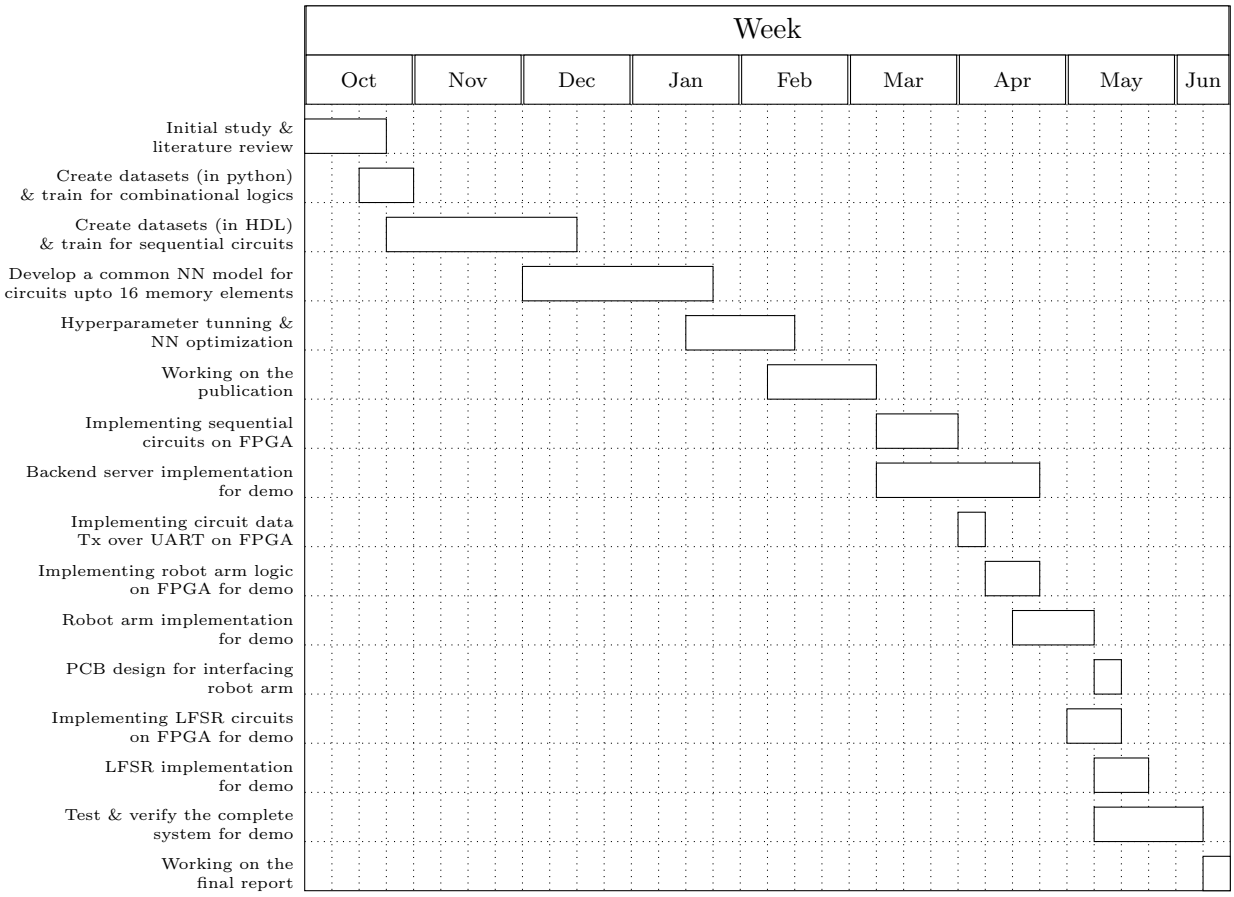
3.6 Task Delegation

Table 2: Task Delegation

Task	Anjana	Ayesh	Sahan	Supun
Initial study & literature review	✓	✓	✓	✓
Create datasets (in python) & train for combinational logics	✓	✓		
Create datasets (in HDL) & train for sequential logics	✓	✓	✓	✓
Develop a common NN model for circuits up to 16 memory elements	✓	✓	✓	✓
Hyperparameter tunning & NN optimization	✓	✓	✓	✓
Working on the publication	✓	✓	✓	✓
Implementing sequential circuits on FPGA	✓			
Backend server implementation for demo		✓		
Implementing circuit data Tx over UART on FPGA	✓			
Implementing robot arm logic on FPGA for demo	✓			
Robot arm implementation for demo			✓	
PCB design for interfacing robot arm	✓			✓
Implementing LFSR circuits on FPGA for demo	✓			
LFSR implementation for demo				✓
Test & verify the complete system for demo	✓	✓	✓	✓
Working on the final report	✓	✓	✓	✓

3.7 Time Line

Table 3: Time Line



In this chapter, we presented the architecture of our project. This is explained using the final phase of our project. Then we examined alternative methodologies for achieving our goals. The only available alternative is traditional RE techniques. However, this is vaguely related to our objectives. Then we discussed the potential risks and how to avoid them. Finally, we have displayed the task delegation and time line.

4 Results

In this chapter, we present the outcomes of our performance analysis for different neural network model architectures applied to three distinct 16-bit circuits: a counter, a shift register with SIPO, and a shift register with PISO. The dataset was meticulously designed to explore all possible iterations under the worst-case scenario. We carefully examine the impact of various parameters, including dataset size, learning rate, time step size, batch size, and number of epochs, on the models' performance.

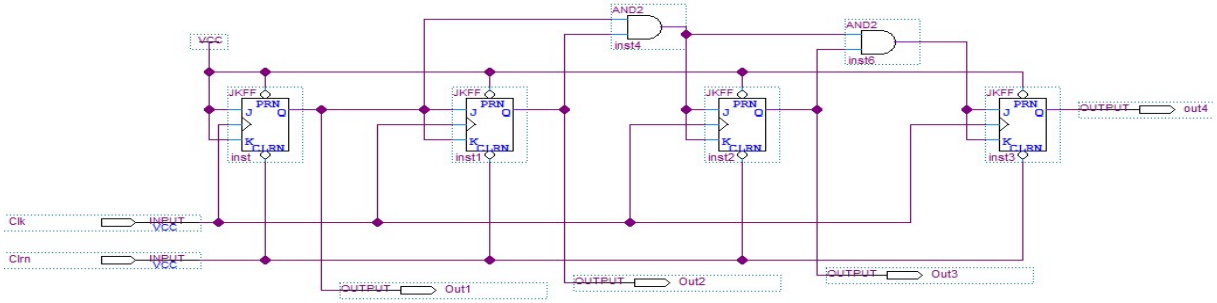


Figure 5: 4-bit Counter

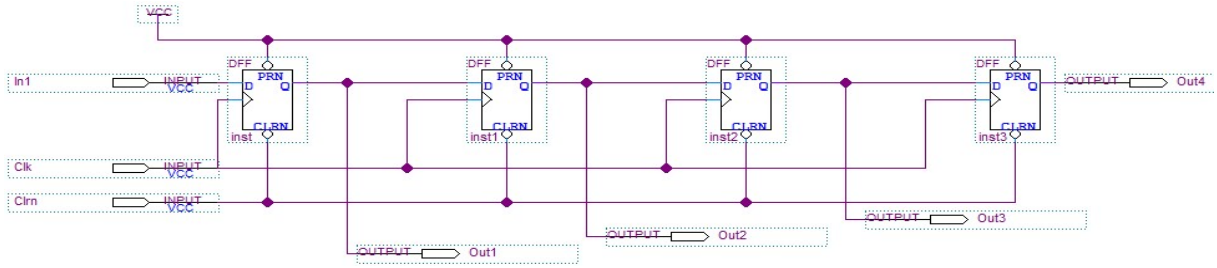


Figure 6: 4-bit SIPO Shift Register

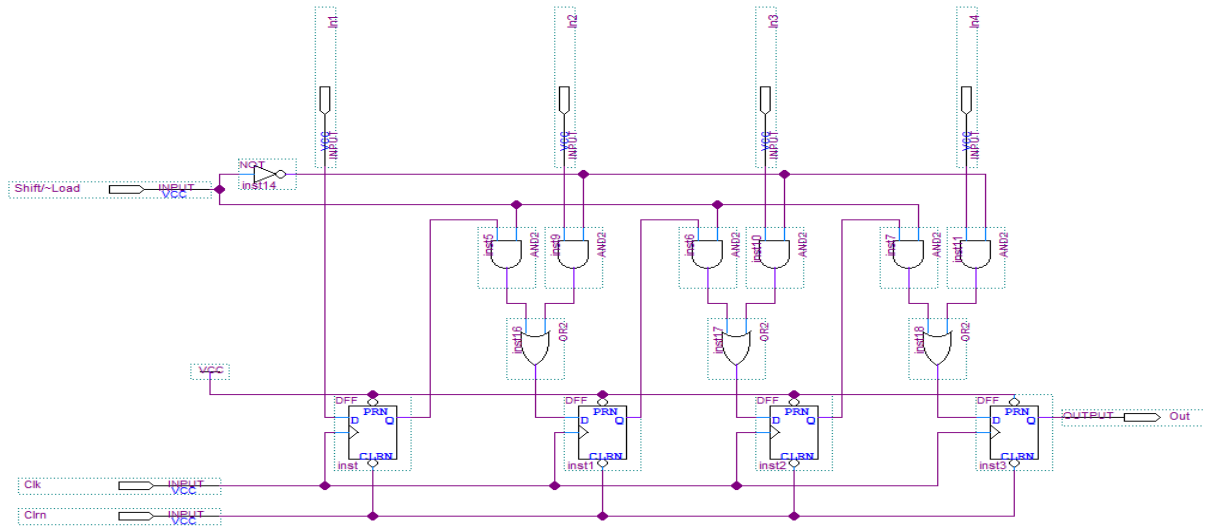


Figure 7: 4-bit PISO Shift Register

The above figures (Figure 5, Figure 6, and Figure 7). illustrates the 4-bit counter, 4-bit SIPO shift register, and 4-bit PISO shift register. These circuits served as the initial building blocks for developing our neural network (NN) model. We subsequently progressed to extend the model's capabilities to handle 16-bit versions of each of these circuits.

4.1 Dataset Creation

When the dataset was created, the following equation was used to calculate all possible iterations of the dataset that could occur under the worst-case scenario (Figure 4).

$$All\ Possible\ Iterations = (2^{in})^{time\ steps} \times 2^{out} \tag{1}$$

The above equation suggests that the ideal dataset size should be chosen by considering the number of inputs, outputs, and time step size. The following results display the performance of two top-performing neural network model architectures when applied to three distinct 16-bit circuits: a counter, a shift register with SIPO, and a shift register with PISO. Input pins were randomized during dataset creation.

4.2 16-Bit Counter

A dataset of 800,000 was used for training, and 200,000 were used for validation. The learning rate was set to 0.01 with a time step size of 1. The batch size used was 4096, and the number of epochs was 30.

Table 4: 16-Bit Counter (two hidden layers)

Parameter	**NN Architecture (LSTM, Dense)**				
	L=8	***L=16***	***L=32***	***D=32***	***D=128***
Val. Acc.	0.9723	0.9870	0.9923	0.9907	0.9957
Val. Loss	0.0864	0.0395	0.0277	0.0447	0.0158

Default parameters: LSTM=32, Dense=64.

Table 5: 16-Bit Counterr (three hidden layers)

Parameter	**NN Architecture (LSTM, Dense1, Dense2)**				
	L=16	***D1=16***	***D1=32***	***D2=32***	***D2=64***
Val. Acc.	0.9972	0.9954	0.9954	0.9934	0.9974
Val. Loss	0.0124	0.0162	0.0162	0.0375	0.0138

Default parameters: LSTM=32, Dense1=64, Dense2=128.

4.3 16-Bit SIPO Shift Register

A dataset of size 1,000,000 was used for training, and a validation set of size 100,000 was used for validation. The learning rate was set to 0.01 with a time step size of 1. The batch size used was 4096, and the number of epochs was set to 20.

Table 6: 16-Bit SIPO Shift Registerr (two hidden layers)

Parameter	**NN Architecture (LSTM, Dense)**				
	L=8	***L=16***	***L=32***	***D=32***	***D=128***
Val. Acc.	0.9919	0.9943	0.9947	0.9943	0.9948
Val. Loss	0.0356	0.0252	0.0231	0.0283	0.0214

Default parameters: LSTM=32, Dense=64.

Table 7: 16-Bit SIPO Shift Registerr (three hidden layers)

Parameter	**NN Architecture (LSTM, Dense1, Dense2)**				
	L=16	***D1=32***	***D1=64***	***D2=32***	***D2=64***
Val. Acc.	0.9946	0.9946	0.9947	0.9942	0.9946
Val. Loss	0.0225	0.0220	0.0214	0.0286	0.0230

Default parameters: LSTM=32, Dense1=64, Dense2=128.

4.4 16-Bit PISO Shift Register

A dataset of size 4,800,000 was taken for training, and 1,200,000 was taken for validation. The learning rate was set to 0.01 with a time step size of 16. The batch size used was 8192, and the number of epochs was set to 6.

Table 8: 16-Bit PISO Shift Registerr (two hidden layers)

Parameter	**NN Architecture (LSTM, Dense)**				
	L=8	***L=16***	***L=32***	***D=32***	***D=128***
Val. Acc.	0.7967	0.8998	0.9357	0.89959	0.8863
Val. Loss	0.4315	0.3012	0.2257	0.3024	0.3208

Default parameters: LSTM=16, Dense=64.

Table 9: 16-Bit PISO Shift Registerr (three hidden layers)

Parameter	**NN Architecture (LSTM, Dense1, Dense2)**				
	L=16	***D1=32***	***D1=64***	***D2=32***	***D2=64***
Val. Acc.	0.9138	0.94972	0.7961	0.9432	0.9492
Val. Loss	0.2590	0.1991	0.4315	0.2113	0.2001

Default parameters: LSTM=32, Dense1=64, Dense2=128.

Table 10: Hyper Parameter Tunning

Circuit		**Learning Rate**			**Stateful**	**Time Step Size**		**Initializer**					**Reset State**
		0.1	***0.01***	***0.001***	***False***	***1***	***8***	***GU***	***HN***	***HU***	***LN***	***LU***	***False***
16-Bit Counter	Val Acc.	0.9976	0.9948	0.9223	0.9918	0.9954	0.9984	0.9921	0.9988	0.9983	0.9953	0.9987	0.9867
	Val Loss	0.0161	0.0247	0.2434	0.0509	0.0208	0.0116	0.0487	0.0097	0.0131	0.0363	0.0100	0.0568
16-Bit SIPO Reg.	Val Acc.	0.8413	0.9945	0.9742	0.9945	0.9942	0.9945	0.9945	0.9946	0.9946	0.9945	0.9947	0.9943
	Val Loss	0.3586	0.0240	0.1213	0.0239	0.0257	0.0238	0.0231	0.0231	0.0228	0.0235	0.0229	0.0250
16-Bit PISO Reg.	Val Acc.	0.5003	0.9474	0.7095	0.9499	0.5444	0.7569	0.9450	0.9498	0.9496	0.8819	0.9498	0.8966
	Val Loss	0.6934	0.2039	0.5027	0.1987	0.6440	0.4112	0.2051	0.1990	0.1992	0.2886	0.1991	0.2772

Default parameters: LSTM=32, Dense1=64, Dense2=64, Epochs=10, Stateful=True, Time step=16, Learning Rate=0.01, Initializer=GlorotNormal, Reset state=True.
[GU:GlorotUniform, HN:HeNormal, HU:HeUniform, LN:LecunNormal, LU:LecunUnifrom]

Based on the performance results obtained above, the best model architecture is chosen to be one with one LSTM layer containing 32 nodes and two hidden dense layers, each with 64 nodes.

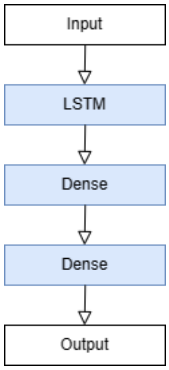


Figure 8: Selected Architecture

Then, tuning of other parameters, including hyper parameters, was initiated for each circuit using the common architecture mentioned above. Based on the findings presented in Table 10, we recommend using the aforementioned architecture with a learning rate of 0.01, HeNormal as the initializer, a time step size of 16, reset state set to True, and stateful set to True. Figure 9, Figure 10, and Table 11 show the results obtained using these recommended parameters for 10 epochs.

Table 11: Results for Best Model Parameters

	16 Bit Counter	**16 Bit SIPO**	**16 Bit PISO**
Val. Acc.	0.9948	0.9945	0.9498
Val. Loss	0.0247	0.0240	0.1190

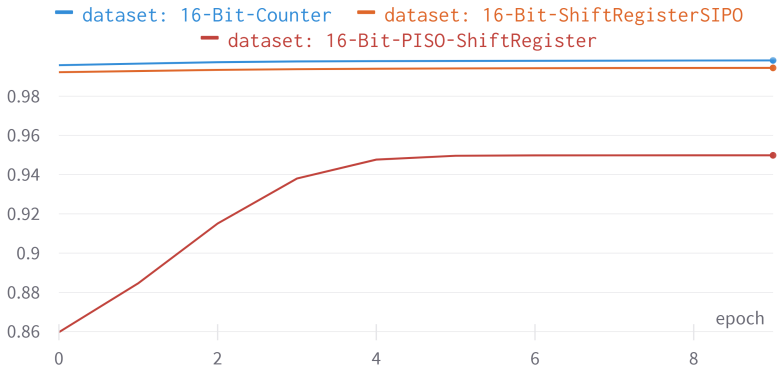


Figure 9: Validation Accuracy

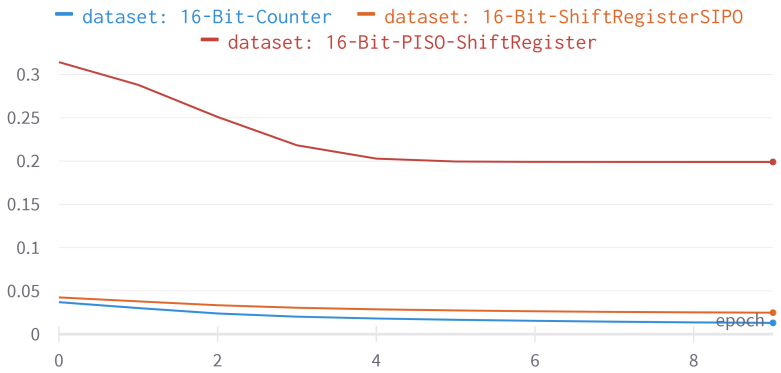


Figure 10: Validation Loss

4.5 Demonstration

In this subsection, we present two compelling demonstrations to showcase the practicality and effectiveness of our developed neural network (NN) model for duplicating digital circuits' functionalities. These demonstrations highlight the real-world applications and the capabilities of our approach in replicating the behavior of complex digital logic circuits.

4.5.1 Robot Arm

In the first demonstration, we aim to showcase a real-world scenario by selecting a servo motor arm controlled by digital logic (Figure 12). For this purpose, we implemented a

logic to control a 4-servo motor arm on an FPGA board. By capturing the input and output data from this digital logic, we transmit the data to our NN model. As depicted in Figure 11 below, our NN model successfully duplicates the functionality of the servo motor arm. Consequently, the NN model gains the ability to take on the role of the servo motor arm, performing tasks with a remarkable level of precision closely resembling that of the real digital logic circuit.

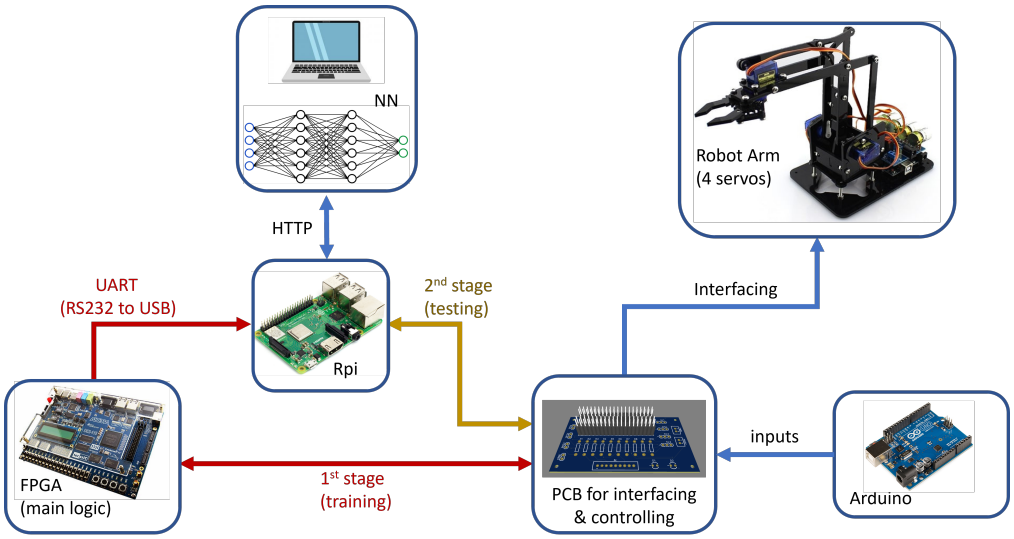


Figure 11: Robot Arm Implementation

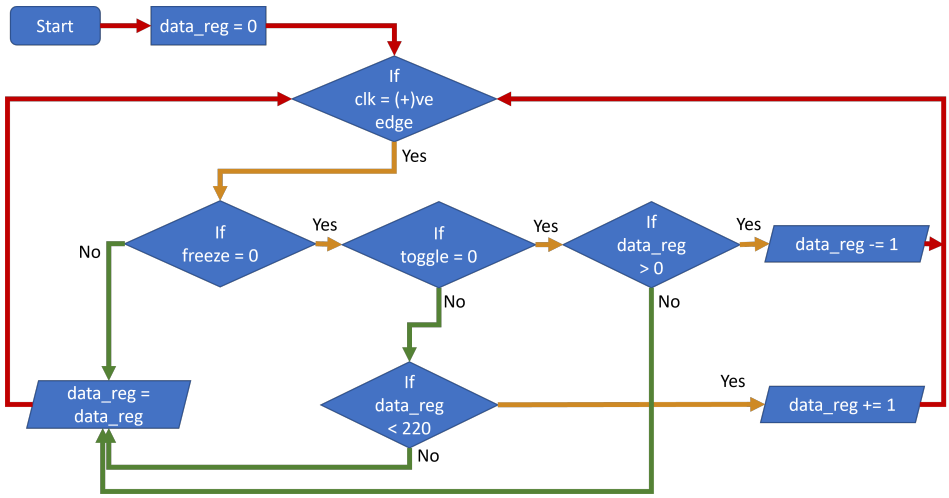


Figure 12: Robot Arm Logic

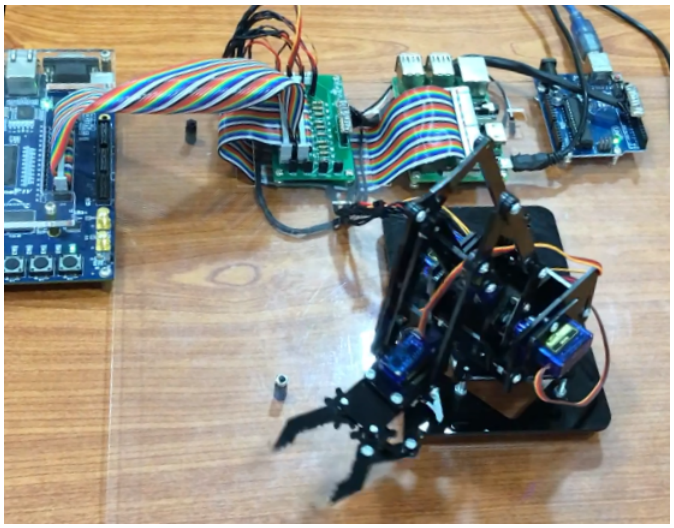


Figure 13: Robot Arm Implementation at the Demonstration

4.5.2 16-Bit LFSR

In the second demonstration, we aim to showcase the capabilities of our developed NN model in replicating the functionality of a 16-Bit LFSR (Linear Feedback Shift Register) circuit (Figure 15). To achieve this, we implemented the LFSR circuit on an FPGA board, generating input-output data that is transmitted to our NN model. Once our NN model successfully duplicates the functionality of the LFSR circuit, it can synchronously mimic the behavior and functionality of the circuit. The outputs of the replicated LFSR circuit are then displayed over an LCD display, providing visual evidence of the NN model's ability to accurately replicate the digital circuit's behavior (Figure 14).

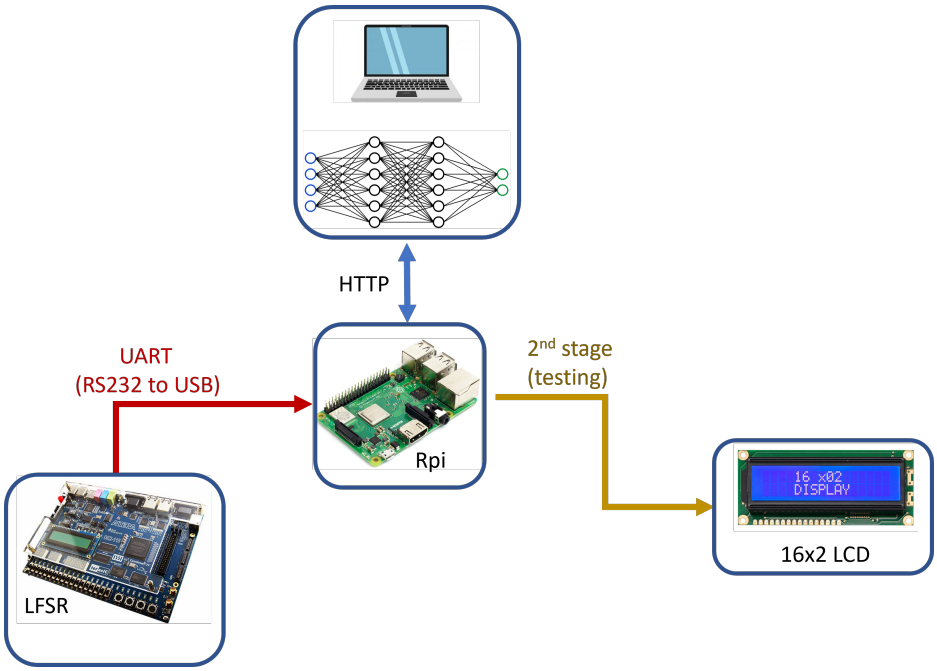


Figure 14: LFSR Implemntation

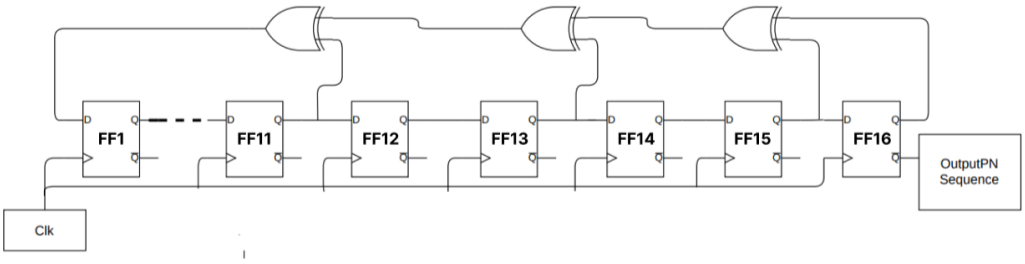


Figure 15: 16-Bit LFSR Circuit

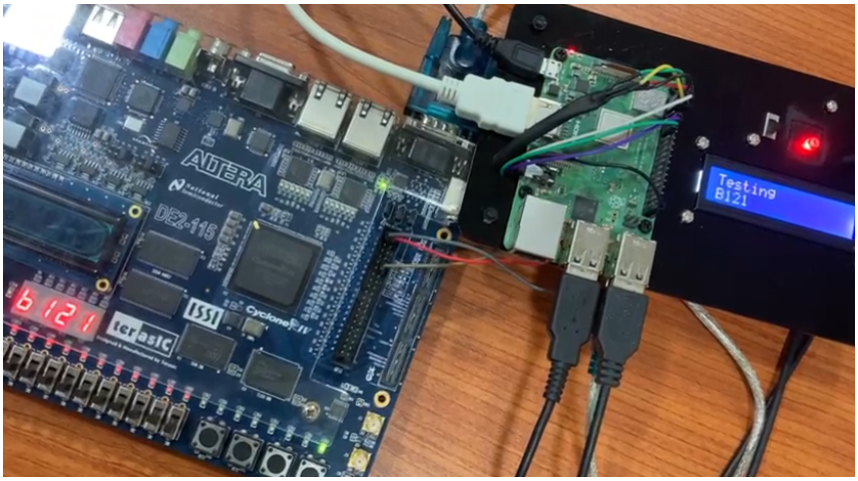


Figure 16: 16-Bit LFSR Implemntation at the Demonstration

These demonstrations serve as tangible examples of the practical applications and potentials of our developed NN model. By showcasing the successful duplication of real-world digital circuits, we affirm the effectiveness and adaptability of our approach. These demonstrations also underscore the viability of our reconfigurable NN algorithm in diverse domains, promising transformative applications in digital circuit design, testing, and replication.

5 Discussion and Conclusions

In this chapter, we present a comprehensive analysis of our research findings, discuss the implications of our work, and outline potential future directions for advancing the field of self-learning digital circuit duplication. The chapter begins with a summary of our main findings from the literature review, shedding light on the existing studies related to our project scope. We then delve into the core contributions of our research, showcasing the development of a reconfigurable neural network algorithm and hardware implementation for duplicating the functionalities of both sequential and combinational digital ICs. The main part of the chapter centers on our experimental approach and the outcomes obtained during the project's execution. We demonstrate the adaptability of our method by utilizing a common neural network model to replicate the functionalities of various circuits, showcasing its effectiveness in duplicating the entire IC functionality based on learned input-output relationships. Furthermore, we discuss the limitations of our experimental results, specifically regarding circuits with memory elements beyond our current threshold of 16. As part of our future works, we actively engage in fine-tuning hyperparameters to optimize the neural network architecture further. Our aim is to achieve enhanced accuracy and efficiency by refining these crucial components of our reconfigurable neural network algorithm. We are also enthusiastic about broadening our analysis to encompass digital sequential circuits with more than 16 memory elements, exploring the duplication of increasingly complex digital circuits. Moreover, we envision the practical impact of our research as we anticipate implementing our approach for digital IC design and testing. By leveraging the potential of our self-learning neural network algorithm, we seek to streamline IC design processes and enhance quality assurance procedures, making a meaningful contribution to the semiconductor industry and beyond. As our approach demonstrates its versatility and adaptability, we aim to make a lasting impact on the field of digital circuit duplication, opening up new avenues for innovation and progress.

5.1 Main Findings of the Literature Review

Currently, researchers have developed a variety of NN algorithms for a variety of applications. A preliminary assessment of the applicability of such networks to the implementation of digital logic circuits was performed. Each network has distinct functional characteristics.

Table 12: Neural Networks

Feedforward NN	because of the nature of its input-output direct relationship, combinational logic has been realized.
RNN-LSTM	because of its state-dependent nature, has the potential to be used in sequential logic.
Hopfield NN	have been designed high-performance digital circuits (AND Circuit, Full adder, D-Latch with 4-bit shift register, multiplier and 4 bit asynchronous counter).
ANN with ABC, PSO and GA	has been employed in the design of 4-bit negative edge triggered binary synchronous up-down counter.
Perceptron	has been used to implement combinational functions.
LNN	has been used to design and optimize combinational logic circuits.
Wavelet Network	has been used to create a universal digital sequential circuit.

There are several types of ANNs, including Feedforward NN, Radial basis function NN, CNN, RNN, Hopfield NN, GRU, and LSTM. RNN, for example, remembers the past and bases its decisions on previous inputs and iterations, which is a requirement when developing a NN engine for sequential logic circuits. GRU is a more sophisticated version of the simple recurrent cell. Because of the vanishing gradient problem in basic RNN cells, the gated concept was introduced. The use of gates improved training capability on longer sequences. LSTM cells were created as a result of further advancement. In comparison to GRU, LSTM has two additional gate structures that improve sequence training capability. Because of their superior performance, LSTM NNs were chosen to implement our NN engine.

5.1.1 Recurrent Neural Networks (RNN)

A Recurrent Neural Network (RNN) possesses the ability to retain information from previous inputs and iterations, allowing its decisions to be influenced by historical data. This characteristic is crucial when employing the algorithm to address sequential logic problems.

Next, we explore three recurrent neural network (RNN) architectures: RNN, GRU (Gated Recurrent Unit), and LSTM (Long Short-Term Memory). These architectures play a vital role in sequence learning tasks, especially when dealing with digital sequential circuits. We will examine their unique characteristics, strengths, and weaknesses to gain a deeper understanding of their suitability for various applications, particularly in digital circuit functionality duplication. By analyzing the distinctive features of each architecture, we aim to determine which one offers the most effective solution to the

challenges posed by long-term dependencies and vanishing gradient problems in sequence learning tasks.

5.1.1.1 Basic RNN

Recurrent Neural Networks (RNNs) exhibit the ability to process one or multiple input vectors and generate corresponding output vectors. Unlike regular neural networks, RNNs incorporate a "hidden" state vector, representing contextual information based on prior inputs and outputs. Consequently, the output produced by an RNN can vary depending on the sequence of previous inputs.

The key feature of RNNs lies in their inherent memory, which retains information from previous forward passes. By using the same parameters for each input and hidden layer, RNNs perform the same task on all inputs, resulting in a streamlined parameter complexity compared to other neural networks. During computations, both the input (x_t) and the previous output ($h_t - 1$) are passed through a $Tanh$ activation function [14].

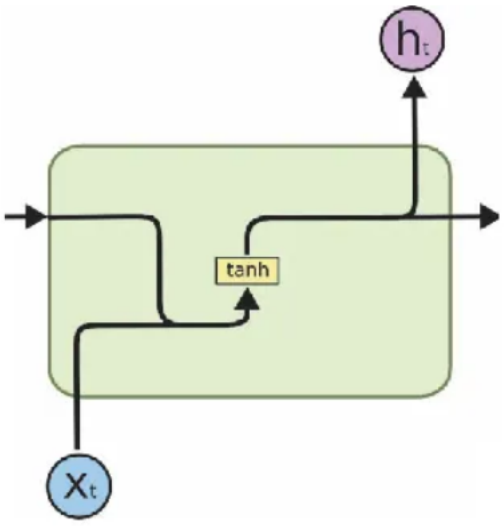


Figure 17: RNN Cell

5.1.1.2 GRU

The Gated Recurrent Unit (GRU) represents an enhanced version of the traditional simple recurrent cell. It was developed to address the vanishing gradient problem commonly encountered in basic RNN cells. By introducing the concept of gating, the GRU significantly improves the model's training capability, particularly on longer sequences.

In the GRU model, an update gate is introduced, which plays a crucial role in deciding whether to pass the previous output ($h_t - 1$) as the next cell's output (h_t) or not. Ad-

ditionally, the forget gate is incorporated, involving additional mathematical operations with a new set of weights (W) to further enhance the model's performance and learning abilities. The incorporation of these gating mechanisms empowers the GRU to capture long-range dependencies more effectively and make it a powerful tool in various sequential data tasks.

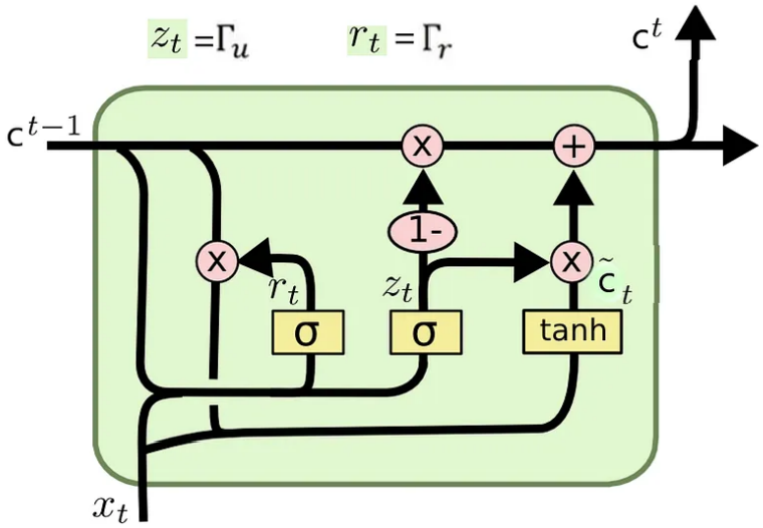


Figure 18: GRU Cell

5.1.1.3 LSTM

When dealing with digital sequential circuits, achieving functionality with long-term sequences becomes necessary. However, traditional RNN cells encounter a vanishing gradient problem, limiting their ability to learn such sequences effectively. To address this challenge, prior research introduced the gated cell approach, leading to the development of LSTM cells. In comparison to GRU, LSTM incorporates two additional gate structures, namely the forget gate and the output gate, to enhance its sequence training capabilities. The forget gate determines what information from the previous cell state should be retained or forgotten, while the output gate controls the output to the hidden state, thus influencing the next hidden state. This design enables LSTMs to address the vanishing gradient problem and handle long-term dependencies, making them well-suited for applications in digital circuit functionality duplication[15].

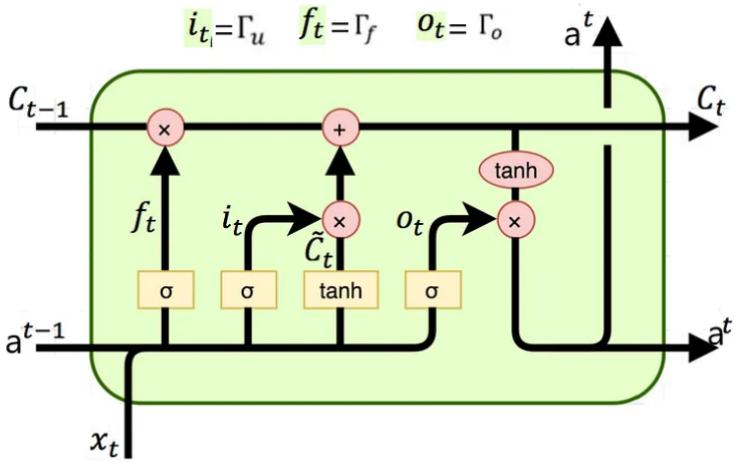


Figure 19: LSTM Cell

We have chosen LSTM (Long Short-Term Memory) for our task because of its superior ability to handle long-term dependencies and address the vanishing gradient problem effectively. Unlike traditional RNNs, LSTM incorporates additional gate structures, including the forget gate and output gate, enabling it to control the flow of information within the cell and preserve essential information over extended sequences. This unique design makes LSTM well-suited for capturing and learning complex patterns in sequential data, making it an ideal choice for our digital circuit functionality duplication project.

5.2 Impact of the Project

The successful development and implementation of our digital circuit functionality duplication project hold significant potential for both local and global impact. This section highlights the potential implications and benefits that our research may bring to the field of digital circuit design and beyond.

5.2.1 Local Impact

At the local level, our project's impact lies in its contributions to the semiconductor industry and related domains. By enabling efficient and accurate duplication of digital circuit functionalities, our reconfigurable neural network algorithm can streamline IC design processes, reducing development time and costs. This local impact extends to digital circuit testing and quality assurance, where our approach can enhance testing methodologies, ensuring the reliability and robustness of digital ICs. Moreover, our project's implementation on FPGA boards and interfacing with physical actuators, such as the servo motor arm, holds promise for advancements in robotics and automation applications at the local level, driving technological innovation and industrial growth.

5.2.2 Global Impact

On a global scale, the impact of our project is far-reaching, touching various industries that rely on digital circuits and ICs. The potential applications span from consumer electronics to communication systems, aerospace, automotive, and healthcare technologies. By simplifying digital circuit duplication and design verification processes, our approach can catalyze innovation and accelerate the development of advanced technologies worldwide. Additionally, the versatility of our neural network model and its adaptability to different circuit functionalities open avenues for applications in emerging technologies such as Internet of Things (IoT), edge computing, and artificial intelligence. As our research findings become accessible to the global research community, they can inspire further advancements in self-learning neural networks and digital circuit replication, fostering collaborative efforts in pushing the boundaries of digital circuit design and paving the way for transformative technologies.

5.2.3 Industrial Impact

Our proposed digital circuit functionality duplication process addresses a critical need in the modern product market. With vendor competition on functionally equivalent devices, the value of the digital IC duplication process has grown significantly. Our approach offers an easier and more accessible pathway for the engineering product duplication process. Traditional complex reverse engineering techniques are no longer required for IC duplication. Manufacturers can readily duplicate the functionality of an unavailable IC using our proposed method and proceed with the production of a similarly functioning new IC. This process proves invaluable in determining whether a failure was caused by a faulty IC, allowing for efficient debugging and maintenance of digital ICs on PCBs in industrial plants. The adoption of our duplication process has the potential to save time and resources on maintenance, thereby optimizing industrial operations and reducing costs. This industrial impact translates into increased productivity, improved product reliability, and enhanced competitiveness in the global market.

5.3 Conclusions

In this project, we proposed a groundbreaking reconfigurable neural network (NN) algorithm combined with a hardware implementation for self-learning the functionalities of both sequential and combinational digital ICs. Our innovative approach treats the IC's processing as a "black box," solely focusing on the input-output relationship. In contrast to existing methods, our technique has the remarkable capability to duplicate the entire IC functionality by effectively learning and replicating its input-output relationships.

Throughout our comprehensive literature review, we observed that several studies

related to our project scope have been conducted over the years. However, we found that only a limited number of them are closely relevant to our work. Moreover, a substantial majority of these studies were conducted many years ago, indicating a need for more recent advancements in this domain. Additionally, most of these studies were primarily concerned with either sequential or combinational digital ICs, lacking a comprehensive solution to address both aspects.

In response to this gap, our project represents a significant advancement as it is the first-ever implementation of a universal trainable NN engine capable of duplicating the functionalities of both sequential and combinational digital ICs. This pioneering work promises to revolutionize the field of digital circuit duplication by offering a holistic approach that overcomes the limitations of previous methods.

As demonstrated in our experimental results, we achieved exceptional accuracy rates in duplicating digital circuits, including a 16-bit counter with 99.48% accuracy, a 16-bit SIPO register with 99.45% accuracy, and a 16-bit PISO register with 94.98% accuracy. These outcomes underscore the effectiveness and potential of our reconfigurable NN algorithm and hardware implementation in accurately replicating the functionalities of diverse digital circuits.

Moving forward, our work lays the foundation for future research in this rapidly evolving field. By further expanding the dataset and training our universal NN engine on a broader range of circuits, we aim to explore the duplication of ICs with memory elements exceeding the current threshold. Through continuous refinement and advancements, our approach presents a promising avenue for enhancing the functionality and applicability of digital circuit duplication techniques.

In conclusion, this research contributes significantly to the development of self-learning digital circuit duplication methods, paving the way for transformative applications in various domains. By harnessing the power of reconfigurable neural networks and embracing a holistic approach, our project exemplifies the potential to drive innovation and progress in the realm of digital IC duplication.

5.4 Future Works

For future works, we are actively engaged in refining hyperparameters to further optimize the neural network architecture, aiming to achieve even higher levels of accuracy and efficiency. By fine-tuning these hyperparameters, we envision enhancing the overall performance and adaptability of our reconfigurable neural network algorithm.

Moreover, as part of our ongoing research, we are expanding our analysis to encompass digital sequential circuits that incorporate memory elements beyond the current threshold

of 16. This extension will enable us to investigate and successfully replicate the functionalities of more complex digital circuits, pushing the boundaries of our approach and opening up new possibilities for practical applications.

Looking ahead, we are eager to explore the implementation of our self-learning approach in digital IC design and testing. By leveraging the potential of our reconfigurable neural network algorithm, we seek to address real-world challenges in digital IC development, testing, and validation. This application-oriented approach holds immense promise for optimizing IC design processes and streamlining quality assurance procedures, thus making significant contributions to the semiconductor industry and beyond.

In summary, our commitment to ongoing research and development drives us to continuously refine and expand our approach's capabilities. As we venture into new territories, we remain dedicated to advancing the field of digital circuit duplication and exploring its practical applications across diverse industries and domains. With a focus on innovation and adaptability, we strive to make a lasting impact on the ever-evolving landscape of self-learning neural networks and digital IC replication.

References

[1] S. Holland. (2019, Dec. 4) Why ics fail and how to troubleshoot them. [Online]. Available: https://maker.pro/blog/why-ics-fail-and-how-to-troubleshoot-them

[2] R. Torrance and D. James, "The state-of-the-art in ic reverse engineering," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2009, pp. 363–381.

[3] M. Morisue and H. Koinuma, "Neural network for digital applications," in *IEEE International Symposium on Circuits and Systems,*, 1989, pp. 2189–2192 vol.3.

[4] H. Ninomtya, K. Egawa, T. Kamio, and H. Asai, "Design and implementation of neural network logic circuits with global convergence," in *Proceedings of International Conference on Neural Networks (ICNN'96)*, vol. 2. IEEE, 1996, pp. 980–985.

[5] K.-I. Sakai, M. Morisue, and H. Koinuma, "Simulation of neural network digital adder and multiplier," *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, vol. 75, no. 2, pp. 47–58, 1992.

[6] D. Sharma, "Neural network simulation of digital circuits," *International Journal of Computer Applications*, vol. 79, no. 6, pp. 7–13, 2013.

[7] H.-K. Lam and F. H. Leung, "Design and training for combinational neural-logic systems," *IEEE transactions on Industrial Electronics*, vol. 54, no. 1, pp. 612–619, 2007.

[8] D. Bhadra, T. A. Tarique, S. U. Ahmed, M. Shahjahan, and K. Murase, "An encoding technique for design and optimization of combinational logic circuit," in *2010 13th International Conference on Computer and Information Technology (ICCIT)*. IEEE, 2010, pp. 232–236.

[9] M. NISHIGAKI, "Universal digital sequential circuit using programmable wavelet array and its application," *Proc. NOLTA'95*, vol. 2, pp. 1041–1044, 1995.

[10] R. T. Yakkali and N. Raghava, "Neural network synchronous binary counter using hybrid algorithm training," *International Journal of Image, Graphics and Signal Processing*, vol. 9, no. 10, p. 38, 2017.

[11] N. Dagdee and N. Chaudhari, "Design methodology for neural network simulation of sequential circuits using neural storage elements," in *SICE'99. Proceedings of the 38th SICE Annual Conference. International Session Papers (IEEE Cat. No. 99TH8456)*. IEEE, 1999, pp. 1171–1176.

[12] M. V. Andrejević and V. B. Litovski, "Electronic circuits modeling using artificial neural networks," *Journal of Automatic Control*, vol. 13, no. 1, pp. 31–37, 2003.

[13] H. Tatsumi, Y. Murai, and S. Tokumasu, "Logic circuit diagnosis by using neural networks," in *Proceedings 31st IEEE International Symposium on Multiple-Valued Logic*. IEEE, 2001, pp. 345–350.

[14] S. A. Afshine Amidi. Cs 230 - recurrent neural networks cheat-sheet. [Online]. Available: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

[15] H. Pedamallu. (2020, Nov. 14) Rnn vs gru vs lstm. [Online]. Available: https://medium.com/analytics-vidhya/rnn-vs-gru-vs-lstm-863b0b7b1573

Appendix

The appendix provides supplementary information and resources related to our digital circuit functionality duplication project. It includes links to three key GitHub repositories, each dedicated to distinct aspects of our research and implementation. These repositories encompass the main neural network model, datasets generated for sequential logics, and FPGA implementation and interfacing details, offering comprehensive insights into our innovative approach and its practical applications.

- **Digital IC Functionality Duplication (Main Repository)**
  - This is the main repository that includes our developed neural network model for digital IC functionality duplication, as well as the final implementation of the project. Here, you can find the comprehensive codebase, documentation, and resources related to our innovative approach for duplicating the functionalities of digital circuits using a reconfigurable neural network algorithm.
- **Sequential Logic Datasets with Designs**
  - This repository houses the datasets generated for various sequential logics using Quartus and ModelSim software tools. These datasets play a crucial role in training and validating our neural network model to accurately replicate the behaviors of sequential digital circuits. The repository contains detailed information and resources related to the dataset creation process.
- **FPGA Implementation and Interfacing**
  - This repository is dedicated to the FPGA implementation and interfacing aspects of our project. It specifically focuses on the UART data transmission purpose, providing essential code and resources related to the communication between FPGA boards and other components of our system. The repository contains valuable insights into the FPGA implementation for data transmission and interfacing with other hardware components.

By accessing these repositories, you can explore the intricacies of our project, access the source code, datasets, and detailed documentation, and gain a comprehensive understanding of the techniques and tools employed to achieve digital circuit duplication using our novel neural network approach.