

ENHANCED FPGA IMPLEMENTATIONS OF
COMMUNICATION PROTOCOL RECEIVERS
WITH EARLY ANOMALY DETECTION
CAPABILITIES



NAVEED IQBAL
09-242242-007

BAHRIA UNIVERSITY ISLAMABAD

ENHANCED FPGA IMPLEMENTATIONS OF COMMUNICATION PROTOCOL RECEIVERS WITH EARLY ANOMALY DETECTION CAPABILITIES



NAVEED IQBAL
09-242242-007

A thesis submitted in fulfilment of the
requirements for the award of the degree of
Master of Science (Computer Engineering)

Department of Computer Engineering

BAHRIA UNIVERSITY ISLAMABAD

AUGUST 2026

MS-13 Thesis Completion Certificate

Student Name: **Naveed Iqbal** Registration Number: **97824**

Program of Study: **Master of Science (Computer Engineering)**

Thesis Title: **Enhanced FPGA Implementations of Communication Protocol Receivers with Early Anomaly Detection Capabilities**

It is to certify that the above student's thesis has been completed to my satisfaction and, to my belief, its standard is appropriate for submission for evaluation. I have also conducted plagiarism test of this thesis using HEC prescribed software and found similarity index at **10%** that is within the permissible set by the HEC. for MS/MPhil/PhD.

I have also found the thesis in a format recognized by the BU for MS/MPhil/PhD thesis.



Principle Supervisor's Signature: _____

Principle Supervisor's Name: **Dr. Syed Khawar Hussain Shah**

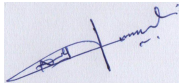
August 20, 2026

MS-14A

Author's Declaration

I, **Naveed Iqbal** hereby state that my MS thesis titled **“Enhanced FPGA Implementations of Communication Protocol Receivers with Early Anomaly Detection Capabilities”** is my own work and has not been submitted previously by me for taking any degree from **“Bahria University, Islamabad”** or anywhere else in the country / world.

At any time if my statement is found to be incorrect even after my Graduate the university has the right to withdraw cancel my MS degree.



Naveed Iqbal

09-242242-007

August 20, 2026

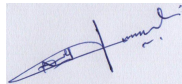
MS-14B Plagiarism Undertaking

I, **Naveed Iqbal** solemnly declare that research work presented in the thesis titled

Enhanced FPGA Implementations of Communication Protocol Receivers with Early Anomaly Detection Capabilities

is solely my research work with no significant contribution from any other person. Small contribution / help whenever taken has been duly acknowledged and that complete thesis has been written by me.

I undertake that if I am found guilty of any formal plagiarism in the above titled thesis even after award of MS degree, the university reserves the right to withdraw / revoke my MS degree and HEC and the university has the right to publish my name on HEC / University Website on which name of students who submitted plagiarised thesis are placed.



Naveed Iqbal

09-242242-007

August 20, 2026

DEDICATION

Dedicated to every Sincere worker driven by a passion for improvement and continuous desire for excellence.

ACKNOWLEDGMENTS

I am thankful to the One True Guide, the Unseen, for guiding my path throughout the Journey.

I would like to thank my supervisor, Dr. Syed Khawar Hussain Shah, for his constant encouragement throughout this research, as well as my coordinators, Dr. Kamran Saeed and Mr. Mubashir, for their support.

Finally, I am thankful to my family for always being there for me.

ABSTRACT

Communication protocols such as UART, and SPI remain widely used across embedded systems, avionics, automotive electronics, and industrial equipment. While these protocols are mature and reliable, their conventional receiver implementations generally report only basic errors (e.g., parity, framing, or synchronization faults). Such mechanisms are insufficient for detecting subtle or intermittent anomalies caused by transmitter-side inconsistencies, hardware degradations, signal integrity variations, or protocol-timing deviations. These irregularities often remain hidden in the testing phase. With advancements in Field-Programmable Gate Arrays (FPGAs), there is increasing potential to embed intelligent diagnostic capabilities directly within communication protocol receivers. This thesis proposes an enhanced approach of implementation and with integrated Self-Monitoring Receiver Implementation (SMRI) framework, a novel FPGA based approach that improves the existing receivers and integrates practical anomaly detection mechanisms into the receiver architecture itself. The SMRI framework is designed to identify issues that may not yet result in incorrect data reception but indicate an increased probability of failure under specific conditions. The proposed research evaluates this framework using commonly used communication standard UART, and also give a brief touch to its use in SPI Protocol, demonstrating how builtin anomaly detection can enhance system validation, reliability, and predictive fault monitoring during early testing phases.

TABLE OF CONTENTS

DECLARATION	ii
DEDICATION	iv
ACKNOWLEDGMENTS	v
ABSTRACT	vi
TABLE OF CONTENTS	vii
LIST OF TABLES	x
LIST OF FIGURES	xi
LIST OF ABBREVIATIONS	xii
1 INTRODUCTION	1
1.1 Introduction	1
1.2 Research Gap	2
1.3 Problem Statement	2
1.4 Research Questions	3
1.5 Research Objectives	3
1.6 Research Contribution	4
1.7 Thesis Organization	5
2 TECHNICAL BACKGROUND AND LITERATURE REVIEW	7
2.1 UART Protocol	8
2.2 UART Transmitter	8
2.3 UART Receiver	8
2.4 UART Example	9
2.5 Technical Details of the Problem in Existing UART Implementations	10
2.6 SPI Protocol	11
2.7 Core Signal Lines	11

2.8	How It Works	11
2.9	Technical Details of the Problem in Existing SPI Implementations	12
2.10	Literature Review	14
2.11	Summary Table of the Reviewed Papers	19
2.12	Summary and Research Gap	23
3	METHODOLOGY	25
3.1	System Architecture Overview for the UART Protocol	26
3.2	The Proposed Mechanism for the UART	27
3.3	Self-Monitoring and Glitch Detection Logic for the UART	28
3.4	Anomaly generator for the UART	29
3.5	Traditional UART Receivers Implementation	29
3.6	System Architecture Overview for the SPI Protocol	30
3.7	The proposed Mechanism for the SPI protocol based communication	31
3.8	Development of the simple SPI master code for proof of concept	31
3.9	Development of the simple SPI slave code for proof of concept	32
3.10	Development of the Specialized SPI slave code for testing the new features in simulations	32
3.11	Self-Monitoring and Anomaly Detection Logic for SPI	33
3.12	Short Summary	35
3.13	Conclusions	36
4	EXPERIMENTS AND RESULTS	37
4.1	SPI based Master Simulations	38
4.2	Simulations details of the UART protocol based communication	38
4.3	Implementations of UART protocol based communicating modules	47
4.4	Conclusions	49

5	CONCLUSIONS & PERSPECTIVES	51
5.1	Conclusion	51
5.2	Research Perspective	52
	REFERENCES	53

LIST OF TABLES

2.1	Summary of the reviewed literature.	19
4.1	Summarized Results of testing in simulations of UART protocol based communication	46
4.2	Summarized Results of testing in hardware	48
4.3	Resources Summary	49

LIST OF FIGURES

2.1	UART example waveform	9
2.2	UART Transmit waveform	10
2.3	SPI waveform	12
2.4	SPI Transition region	14
3.1	Block Diagram for Testing UART	27
3.2	Block Diagram of Self Monitoring	29
3.3	Block Diagram for Testing SPI	30
3.4	Block Diagram for Testing SPI Having Anomalies	32
3.5	SPI Clock Cycle Area Division	34
4.1	UART Transmitter Waveform showing noise insertion at 10th byte	39
4.2	UART Transmitter Waveform showing noise insertion at 20th byte	40
4.3	UART Transmitter Waveform showing noise insertion at 30th byte	41
4.4	UART Transmitter Waveform showing noise insertion at 40th byte	42
4.5	UART Transmitter Waveform showing noise insertion at 50th byte	43
4.6	UART Receiver Waveform showing detection of short pulse	44

LIST OF ABBREVIATIONS

FPGA	Field Programmable Gate Array
UART	Universal Asynchronous Receiver/Transmitter
SPI	Serial Peripheral Interface
SMRI	Self Monitoring Receiver Implementation
IC	Integrated Circuit
TB	Test Bench
UUT	Unit Under Test
VGA	Video Graphics Array
BIST	Built In Self Test
LFSR	Linear Feedback Shift Register
I2C	Inter Integrated Circuit
CAN	Controller Area Network
ECU	Electronic Control Unit
FIFO	First In First Out
RF	Radio Frequency

CHAPTER 1

INTRODUCTION

1.1 Introduction

Data communication across on-chip, off-chip, and module boundaries is one of the basic functionalities of modern digital systems. There are many protocols which are used to exchange information but, in this thesis, we will focus on UART (Universal Asynchronous Receiver Transmitter) and SPI (Serial Peripheral Interface). Both these protocols are old, easy to understand and implement. Most of the technical people who have worked on data exchange between modules will be familiar with these protocols and may understand the core idea proposed in this thesis. For this reason, these two protocols are chosen. The same ideas may be used in any other protocol receiver. Existing FPGA based receiver IP cores for UART and SPI are written with traditional approaches which may be enhanced. Their error reporting probability is very low, they can report parity mismatches, framing faults etc. only if error occurs at a specific time. In hardware deployments, transmission lines rarely fail instantly. Physical layer problems such as impedance mismatches, clock drift, or even abnormal waveform due to bug at transmitter side firmware if exists, can be detected at an early stage. These appear as duty cycle distortion, edge jitter, phase shifts or high frequency glitches or dips but still satisfy standard digital sampling thresholds. Such anomalies which are not detected in lab testing but may appear in the field

which may cause unwanted consequences. Our proposed design is able to detect anomalies at the blind spots even if they are not disturbing the original data. Additionally, these receivers will be more robust to decode the data correctly as compare to the existing schemes.

1.2 Research Gap

Current receivers are implemented only to receive data. There are no measures taken to report some simple anomalies whose probability is high and detection is very simple. In short, these receivers fail to identify:

- Non-critical timing variations
- Transmitter side implementation bugs
- Short term noise induced deviations
- Miss understanding while reading data sheet especially in case of SPI timing calculations

Gap: Lack of intelligent, FPGA-based enhanced anomaly detection systems across UART, SPI.

1.3 Problem Statement

Current FPGA based designs of UART and SPI receivers lack the functionality to detect the already present anomaly at an early stage. Also, it is common practice that if a module passes initial testing, people skip its detailed testing which requires additional hardware like oscilloscopes and high-speed logic analyzers. Our provided solution integrates diagnostic hardware directly into the UART and SPI receiver to detect abnormality at the blind spots even if the data received is correct.

1.4 Research Questions

To address the limitations identified in existing receiver implementations, this study investigates how to add very simple checks to existing receivers to fill this gap. Consequently, the following specific sub-questions are addressed:

- How can FPGA-based receivers detect some anomalies even with correct data reception?
- What diagnostic parameters can detect early signs of degradation or instability?
- How to integrate these mechanisms into UART and SPI with minimal hardware overhead?
- How effective is SMRI vs. conventional error-reporting schemes?

1.5 Research Objectives

To address the limitations identified in existing receiver implementations along with possible and easy but valuable enhancements, the primary objective of this thesis is to design, develop, and validate a framework capable of identifying anomalies in the communication line. To achieve this goal, the study focuses on the following specific research objectives:

- To critically evaluate current receiver designs in order to pinpoint their fundamental limitations and vulnerabilities.
- To develop a novel SMRI framework.
- To develop an FPGA-based transmitter capable of actively injecting anomalies for testing purposes.
- To implement the proposed SMRI receivers and validate it's all features.

- To directly compare the performance with existing implemented modules.

1.6 Research Contribution

To address the limitations in the existing implementations of receivers, this thesis introduces the Self-Monitoring Receiver Implementation (SMRI) framework for FPGA-based receiver architectures that will have additional functionality of detecting abnormality along with the standard data reception. UART and SPI receivers are implemented here as an example, however the idea may be extended to any other communication receiver.

The primary contributions of this work are:

- The SMRI Framework Architecture: Introducing a FPGA-based design approach that tracks edge placement, sampling margins, and timing jitter without interrupting standard data decoding.
- Advanced Data Decoding: Introducing new techniques to increase the probability of receiving correct data in some cases where the existing schemes are failing to decode correct data.
- UART-Receiver Design: Complete RTL implementations of SMRI-enabled receiver for UART Protocol which will be helpful to understand the concept and use it on any other communication protocol receiver.
- SPI-Receiver Design: Complete RTL implementations of SMRI-enabled receiver for SPI Protocol which will be helpful to understand the concept and use it on any other communication protocol receiver.
- Transmitter side code for Fault-Injection: A custom, Verilog-based transmitter code capable of injecting controlled fault in order to analyze the behavior of our new receiver and testing both the existing receivers and our designed receiver.

- **Simulation and Implementation Results:** Analyze and share results of simulations and implementations. Also providing tips to include such functionalities in any protocol receiver.

It is also clarified that early anomaly detection means identification of existing signal anomalies before they impact data integrity, a critical vulnerability that traditional UART receiver designs fail to report.

1.7 Thesis Organization

This thesis is organized into the following five chapters:

- **Chapter 1:** This chapter provides a brief introduction to the research work done. The research gap is also highlighted here, along with the problem statement. The research questions and objectives are derived, and at the end, the core contributions of this work are explained briefly.
- **Chapter 2:** This chapter is divided into two main sections: technical background and literature review. The first section provides foundational knowledge of the UART protocol, its main modules, and the technical background of the existing problem illustrated with an example. It further details the SPI protocol, its core signals, and its operational mechanism. The second section provides a comprehensive review of the existing work available in the literature.
- **Chapter 3: Methodology:** This chapter presents the overview of system architecture along with the details of each module used. It explains the whole setup and the proposed mechanism for UART protocol using new scheme along with details of designing of the Self-Monitoring Receiver Implementation (SMRI) framework. It also provides information including the detection of timing anomaly modules and the custom fault injection transmitter. Moreover, this also provides the technical details regarding embedding the proposed monitoring design in SPI protocol implementations.

- Chapter 4: Experiments and Results: This chapter presents the experiments and testing performed. The results obtained after simulating the new design and its comparison with the existing designs are also presented in this chapter. Further, the results obtained after hardware implementations are shown which are summarized in tabular form with the main focus on anomaly detection accuracy.
- Chapter 5: Conclusions & Perspectives: This final chapter synthesizes the overall outcomes of the study by summarizing the primary contributions and major achievements of this research. Highlights how this work advances or improves upon existing solutions in the literature. Furthermore, this chapter provides a forward-looking perspective on the significance of the results and explores future research directions. Specifically, it discusses how the proposed methodologies can be expanded, adapted, or applied to other communication protocols and related frameworks.

CHAPTER 2

TECHNICAL BACKGROUND AND LITERATURE REVIEW

This chapter is on the technical background of both the UART and the SPI protocol together with review of the literature. The following topics are addressed in this chapter.

- Brief explanation of the UART Protocol
- Brief explanation of the UART Transmitter
- Brief explanation of the UART Receiver
- Basic UART example along with waveform
- Technical details of the problem in existing UART implementations
- Brief explanation of SPI Protocol
- Core signals of SPI Protocol
- How SPI Works
- Technical details of the problem in existing SPI implementations
- Literature review
- Comparison table of reviewed papers
- Conclusion

2.1 UART Protocol

Universal Asynchronous Receive Transmit is a protocol to exchange data between two entities. It has normally two modules, a transmitter and a receiver. For pure data communication, it needs only one line at transmitter side for transmission and one line at receiver side for reception. Additional pins may be required for ground etc. A high logic is used to transmit '1' while low logic is used to transmit '0'.

2.2 UART Transmitter

The working principle of UART Transmitter side is as follows:

- Although data is transmitted bit by bit but as per protocol requirements, a single data packet is consist of 8 bit data byte along one Start and one Stop bit.
- The transmitter side adds a start bit '0' at the start and a stop bit '1' at the end of each byte.
- The transmitter side transmits data bit by bit starting with the start bit, then the data byte starting with the least significant bit of the data byte and finally the stop bit.
- Each bit duration is the same as calculated by the data rate.

2.3 UART Receiver

The working principle of UART Receiver side is as follows:

- The receiver receives data bit by bit to form a byte.
- In idle state, it is in wait of a low logic.
- The very first low logic is considered as the start of each byte and is then used to calculate the timing of each bit for valid detection.

- The received byte is concluded when the 10th bit received is confirmed to be a high logic. This ensures the proper timing up to some extent.

2.4 UART Example

For example, the user wants to transmit a data byte which is 0x5D (8-bits in hexadecimal format). The process will be as follows:

- The binary of 0x5D is 01011101
- The transmitter side will keep the transmitting pin a low logic for single bit duration to start transmission.
- Then it will transmit the data bit one by one starting with least significant bit which is '1' here in this example.
- At the end, a logic one is transmitted as a stop bit.

The waveform for the above example is shown in following figure [2.1](#)

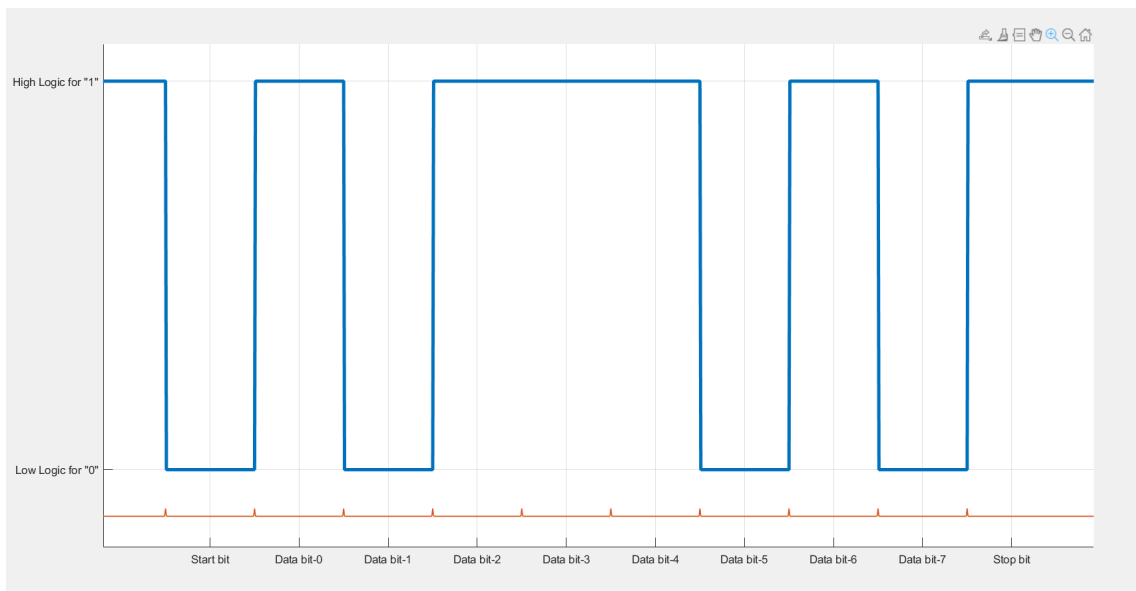


Figure 2.1: UART example waveform

2.5 Technical Details of the Problem in Existing UART Implementations

Suppose we are working with UART protocol, and the technical specifications are as follows:

- Clock frequency is 16MHz
- UART data rate is 1mbps
- This means that a single byte will be transmitted in 10us (Including Start and Stop bits)
- A single bit time will be 1us
- 1us is equivalent to 16 clock cycles of our main clock

Suppose the transmitter side is transmitting any single data bit which is '0' represented with a 'low' level of voltage. Its correct and the supposed incorrect waveforms are as given below in figure 2.2

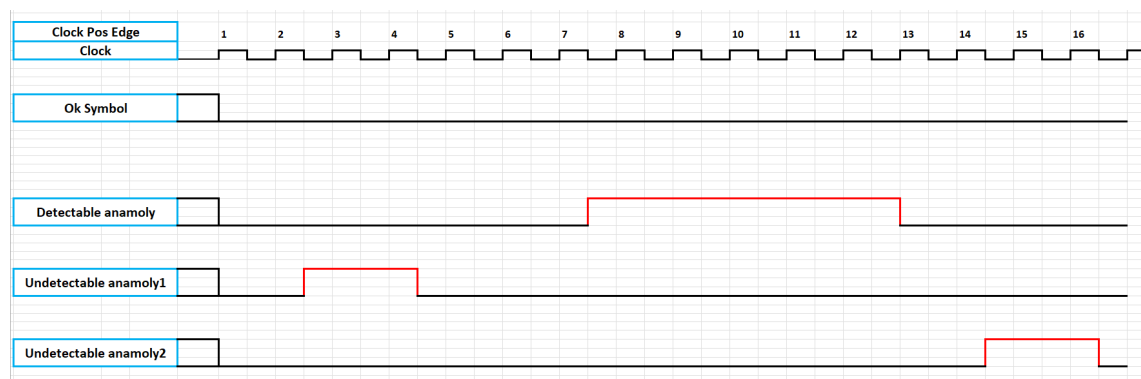


Figure 2.2: UART Transmit waveform

In the above figure, one correct and three incorrect wave forms for the single transmitted bit are given. Now the existing receivers use various techniques to receive a single bit which are as follows:

- Some receivers sample the incoming bit at 50 percent of the bit time for example 8th clock cycle.

- Some receivers sample the 8th, 9th and 10th bit and perform a majority voting technique.
- But none of the receivers basically look at the initial 6 or last 4 cycles which are referred as blind spots in this work.

2.6 SPI Protocol

Serial Peripheral Interface is a full duplex protocol that defines how two entities can exchange data with one another in a synchronous mode. It is mostly used for off-chip communication between IC's. For example a board having Microcontroller or FPGA as main processing element while it has many other IC's used for acquiring different data and provide it to main processing unit. So a standard protocol is needed to perform this exchange. SPI is one of the simple, mostly used protocol for such purposed.

2.7 Core Signal Lines

- SCLK : Serial Clock which is generated by the master.
- MOSI: Data from the master to the slave.
- MISO: Data back from the slave to the master.
- CS: Chip Select used as an enable signal.

2.8 How It Works

- The master pulls the target peripheral's Chip Select line low to begin communication.
- SCLK is started.
- Data is placed on their respective output lines simultaneously with one clock edge and is sampled on the other edge of the clock.

- The master pulls the Chip Select line high to end the communication.

Although there are different modes in which SPI protocol may work. For drawing a sample waveform, let suppose both master and slave will update their data out on positive edge of the clock while both will record the incoming bit on the negative edge of the clock. Then for a very simple 3 bits transfer, the waveform will be as follows: 2.3

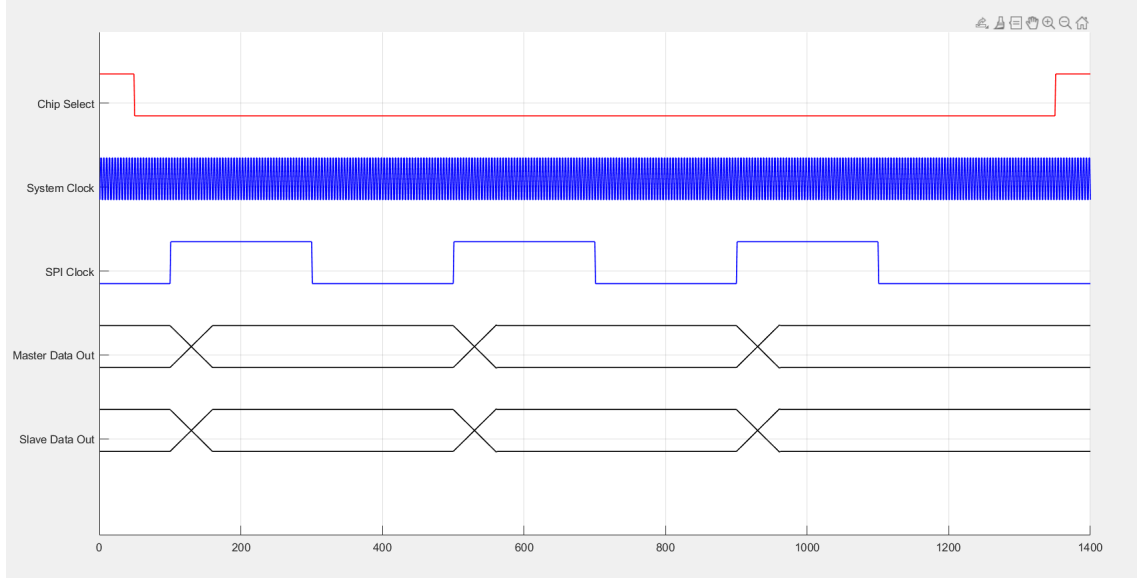


Figure 2.3: SPI waveform

2.9 Technical Details of the Problem in Existing SPI Implementations

In the same way while implementing SPI receivers, the existing schemes only sample the data on the SPI clock edge which is normally much slower than the system main clock. So the main clock can be used to look for any anomalies during the whole clock cycle of SPI clock.

Let us suppose that the implementation of SPI Main (master) in an FPGA is required in order to communicate with the peripherals available on the same board which will act as secondary module. Then the steps to implement the main module will be as follows:

- Identification of the SPI mode for the secondary module.

- Selection of frequency for SPI clock.
- Understanding the timings diagram for reliable implementation.
- Implementation of basic functions.
- Identification of the timing windows where the transition is allowed and where it is not allowed.
- Inserting checks using system clock to detect transitions in the timing window where they were not allowed
- Developing methods to record and report such anomalies.

For example, Our system clock is 100MHz and the secondary module details as per datasheet are as follows:

- Both main and secondary module will out the outgoing data on positive edge of the clock
- Both main and secondary module will sample the incoming data on negative edge of the clock
- The frequency of SPI is required to be 1MHz
- The maximum propagation delay of the clock signal to reach the secondary module is 50ns
- The maximum propagation delay of the clock signal to reach the secondary module is 50ns

Then the regions where the transition is allowed and where it is not allowed along with other details will be as given in the following figure [2.4](#)

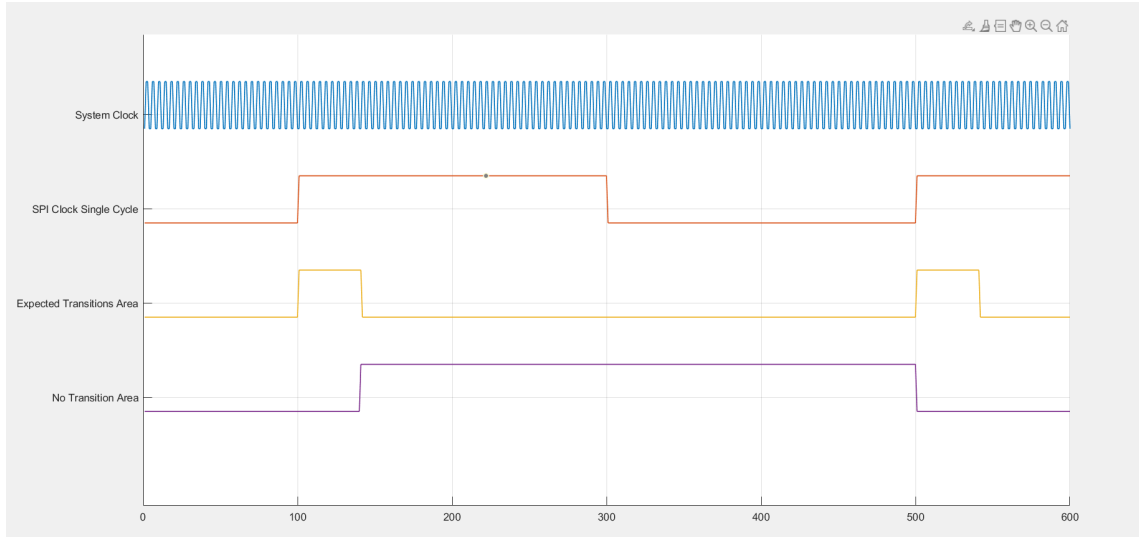


Figure 2.4: SPI Transition region

In the above figure, System clock, SPI single clock cycle, Area where transition is expected and the area where transition is not allowed are highlighted and drawn. It is also mentioned that some minor delays are not considered for simplicity. This figure is drawn only to explain the idea at an abstract level.

2.10 Literature Review

A lot of papers were reviewed during this thesis. Some of them are short listed to make the summary of literature review.

Ali et al. (2023) [1] highlighted that data interruption during serial transmission causes communication delays, slower data transfers, and higher interfacing overheads. These issues stem from inappropriate buffer sizes at both the transmitter and receiver ends. To resolve this, the authors proposed a dynamic buffer size adjustment method. In this approach, the transmitter configures its buffer size based on current requirements and communicates these new settings to the receiver, allowing the receiver to adjust its buffer accordingly. The authors introduced two modes of configuration, “Instantiation” and “Runtime” corresponding to static and dynamic parameterization,

respectively. Furthermore, the authors have also presented the ASIC implementation of the proposed modified UART IP Core.

Li (2024) [2] has highlighted the significance of UART protocol and demonstrated efficient implementation of UART protocol on FPGA. The author has proposed a technique to change the data rate at run time in order to avoid the problems related to data rate due to very long or very short distances or other unknown parameters. To deal with this the author has proposed a technique to set the data rate. However, for this technique both the transmitter side and receiver side should be implemented in customized way. The author has also demonstrated the principle for multibyte transmission.

Xiang et al. (2023) [3] have talked about the plain implementation of UART protocol. This implementation is then connected with top level modules to be used for VGA display purposes. The focus is on the modularity of modules during design.

Xing (2024) [4] has highlighted the simplicity and significance of UART protocol. The author has also talked about the limitations of some available IC's when faster data rates are required. Further plain UART transmitter, receiver and baud rate generators are implemented and tested. At the receiver side, the sample is taken at the central interval of the bit duration. The author has also provided the waveforms captured after simulation.

Yi-yuan (2011) [5] has explained in detail how UART protocol works, what are the main components of UART and how they work together. The design has also relied on method of sampling the incoming bit at central interval. Moreover, the work has considered the chances of misjudging an incoming byte due to very short duration glitch in the incoming signal, so it is ensured to also verify the starting bit at 50% of the bit duration before

considering it a proper start. In this way an attempt is made to avoid starting reception due to noise or other unwanted effects.

Saha et al. (2014) [6] have demonstrated the importance of Serial Peripheral Interface along with the concern of complexity of the circuits due to advancement of technology. In order to address the issues related to such complexities and reduce testing cost, the authors have proposed a Built In Self Test (BIST) hardware which will be able to transmit a fixed data and then receive it in order to confirm its correctness. Moreover, the concept of control bits is used to program the feature of the Serial Peripheral Interface module. These bits are used for selecting the clock rate, mode of SPI, and the order of bits such that it decides which bit goes out first etc. The design to test the BIST feature, Random pattern generator and Linear Feedback Shift Registers are used. The simulations and implementation results are also presented by the authors.

Qiang et al. (2020) [7] have explained the Serial Peripheral Interface in general and talked about its significance. The core signals used usually by any SPI protocol based module are also discussed. The authors have also explained how this protocol works. What are the different four modes along with drawing waveform for all modes. Moreover the simulations and implementations results are shown. A state machine based approach is used for implementation. The authors have also tried to simplify the states of state machine. Overall, this work was about the basic implementation of module for communication with any other SPI protocol based module.

Pahlevi et al. (2018) [8] have discussed both UART and SPI protocols. The study is mainly focused on the comparison of the communication speed achievable by the specific micro-controller Atmega328 and the FPGA the authors used.

Rio et al. (2014) [9] have discussed two communication protocols used for exchanging data which are SPI and I2C. The study is mainly focused on the basic implementation of these both protocols on FPGA. The authors have also shared the waveforms obtained after testing.

Liu (2024) [10] have demonstrated the importance of Serial Peripheral Interface and highlighted that the traditional SPI hardware implementations often lack flexibility in clock frequency, clock polarity (CPOL), and clock phase (CPHA). These limitations cause increase in development time and cost. Also the adaptability in various applications is last. The author has proposed a design to overcome such limitations and focused on implementation that will be able to support multiple modes of communication using SPI protocol. The author has also provided the feature of dynamically selecting SPI clock frequency. The feature of selecting the data width is also added thus improving flexibility regarding data width selection. The authors have also performed detailed testing in different modes of SPI and provided waveform.

Kopparapu et al. (2025) [11] The authors have briefly discussed architecture of FPGA. UART protocol basic introduction is also mentioned. And a brief introduction of temperature sensor is illustrated.

Ganeswara Rao et al. (2024) [12] The authors have demonstrated a low power design for UART protocol based communication specifically for use in smart homes. The authors have initially provided basics of UART protocol and then demonstrated clock gating technique. The authors have provided RTL schematic and Power analysis.

Raju et al. (2025) [13] The authors have highlighted the significance of Inter Integrated Circuits, Serial Peripheral Interface, and Universal Asynchronous Receive Transmit Protocol. Then the authors have demonstrated a

technique for Multi Protocol Conversion.

Penurkar et al. (2025) [14] The authors have highlighted the significance of serial communication over parallel communication. Design and implementation of Single Master Single Slave Serial Peripheral Interface (SPI) on FPGA of is demonstrated.

Xie (2024) [15] The author has talked about evolvement of UART and demonstrated the design of transceiver using finite state machines. Moreover, the author has highlighted the features like different baud rates and data formats.

Maity et al. (2024) [16] In this work, the authors have demonstrated the design and development of the Serial Peripheral Interface on FPGA. The resources summary and synthesis results are also provided.

D'Silva et al. (2025) [17] In this work, the authors have pointed towards the use of multiple protocols in a single system. The authors have proposed an FPGA based implementation of a Controller Area Network (CAN) to Serial Peripheral Interface (SPI) bridge, which allows the interfacing of legacy CAN based modules of the Electronic Control Units (ECUs) with other micro controller units that act as the SPI slaves.

Trehan et al. (2024) [18] In this paper, the authors have worked on The Multi Protocol Conversion Unit and have used Serial Peripheral Interface and Inter Integrated Circuit protocols.

Sonwane et al. (2023) [19] The authors have highlighted the significance of UART and have worked on its design and development on FPGA. A First In First Out memory buffer is used in this work to prevent data losses.

HariPriya et al. (2022) [20] The design of UART is demonstrated with the focus on minimizing energy usage.

Kumar (2021) [21] The author has worked on UART for RF modules.

Beyond these main texts, a broader body of literature was thoroughly reviewed to ensure a complete understanding of the topic. However, many of these extra studies yielded heavily overlapping conclusions and backed up existing ideas without introducing new theoretical frameworks or different empirical views. To maintain a sharp focus on the main research problem and avoid repetitive talk, these studies are acknowledged as part of the wider scholarly agreement but are not analyzed in individual detail within this chapter.

2.11 Summary Table of the Reviewed Papers

Table 2.1 provides a structured look at the literature reviewed so far, offering a quick reference for the current state of research. The table briefly describe the main technical contributions provided so far. This comparative view helps pinpoint the specific technical gaps that the remainder of this work addresses.

Table 2.1: Summary of the reviewed literature.

Reference	Focus Area
Ali et al. (2023) [1]	Focused on the issues related to inappropriate buffer sizes chosen initially. Worked on dynamically adjustments of buffer size at transmitter side and then communicate these settings to the receiver side to adjust the buffer size accordingly.

Continued on next page

Table 2.1 continued from previous page

Reference	Focus Area
Li (2024) [2]	Demonstrated efficient implementation of UART protocol with the main focus on adaptive baud rate generation
Xiang et al. (2023) [3]	Demonstrated implementaion of basic UART protocol with the focus on modularity which can then be used for VGA image display.
Xing (2024) [4]	FPGA based plain implementation of UART transmitter and Receiver. Sample is taken in the center of bit time as mostly done in the traditional implementations.
Yi-yuan (2011) [5]	provides plain implementation while improving start bit detection mechanism up to some extent
Saha et al. (2014) [6]	A Built In Self Test (BIST) hardware is proposed which will be able to transmit a fixed data and then receive it in order to confirm its correctness. Moreover, the concept of control bits is used to program the feature of the Serial Peripheral Interface module. These bits are used for selecting the clock rate, mode of SPI, and the order of bits such that it decides which bit goes out first etc.
Qiang et al. (2020) [7]	presented the basic implementation of Serial Peripheral Interface using state machine. Modes of the SPI are also discussed. The authors have also tried to simplify the states of state machine.
Pahlevi et al. (2018) [8]	The authors have discussed both UART and SPI protocols. Comparison of the communication speed achievable by the specific micro-controller Atmega328 and the FPGA is done.

Continued on next page

Table 2.1 continued from previous page

Reference	Focus Area
Rio et al. (2014) [9]	The authors have discussed two communication protocols used for exchanging data which are SPI and I2C with the focus on the basic implementation of these both protocols on FPGA.
Liu (2024) [10]	The author has proposed a design that will be able to support multiple modes of communication using SPI protocol, provided the feature of dynamically selecting SPI clock frequency and the feature of selecting the data width is also added thus improving flexibility regarding data width selection.
Kopparapu et al. (2025) [11]	briefly discussed architecture of FPGA along with the UART protocol basic introduction and a brief introduction of temperature sensor.
Ganeswara Rao et al. (2024) [12]	The authors have demonstrated a low power design for UART protocol based communication specifically for use in smart homes. The authors have initially provided basics of UART protocol and then demonstrated clock gating technique. The authors have provided RTL schematic and Power analysis.
Raju et al. (2025) [13]	The authors have highlighted the significance of Inter Integrated Circuits, Serial Peripheral Interface, and Universal Asynchronous Receive Transmit Protocol. Then the authors have demonstrated a technique for Multi Protocol Conversion.

Continued on next page

Table 2.1 continued from previous page

Reference	Focus Area
Penurkar et al. (2025) [14]	The authors have highlighted the significance of serial communication over parallel communication. Design and implementation of Single Master Single Slave Serial Peripheral Interface (SPI) on FPGA of is demonstrated.
Xie (2024) [15]	The author has talked about evolvement of UART and demonstrated the design of transceiver using finite state machines. Moreover, the author has highlighted the features like different baud rates and data formats.
Maity et al. (2024) [16]	In this work, the authors have demonstrated the design and development of the Serial Peripheral Interface on FPGA. The resources summary and synthesis results are also provided.
D'Silva et al. (2025) [17]	In this work, the authors have pointed towards the use of multiple protocols in a single system. The authors have proposed an FPGA based implementation of a Controller Area Network (CAN) to Serial Peripheral Interface (SPI) bridge, which allows the interfacing of legacy CAN based modules of the Electronic Control Units (ECUs) with other micro controller units that act as the SPI slaves.
Trehan et al. (2024) [18]	In this paper, the authors have worked on The Multi Protocol Conversion Unit and have used Serial Peripheral Interface and Inter Integrated Circuit protocols.

Continued on next page

Table 2.1 continued from previous page

Reference	Focus Area
Sonwane et al. (2023) [19]	The authors have highlighted the significance of UART and have worked on its design and development on FPGA. A First In First Out memory buffer is used in this work to prevent data losses.
Haripriya et al. (2022) [20]	The design of UART is demonstrated with the focus on minimizing energy usage.
Kumar (2021) [21]	The author has worked on UART for RF modules.

It is again clarified that in addition to the primary literature discussed above, a wide range of secondary sources was examined to map the wider landscape of the field. Because these additional studies largely copied known methods and offered parallel results, they did not provide the unique conceptual turning points necessary to advance this specific inquiry. Their details have therefore been left out to keep the discussion short and centered strictly on the most influential viewpoints.

2.12 Summary and Research Gap

The literature reviewed in this chapter demonstrates that previous research on UART and SPI protocol implementations has primarily centered on several core functional and performance features. While these baseline metrics are critical for standard operating conditions, they often overlook edge case timing anomalies. In practical applications, these anomalies frequently arise from hardware vulnerabilities, subtle implementation flaws, or misinterpretations of complex datasheet specifications, a risk that is especially pronounced during SPI interfacing. Specifically, previous research has focused on:

- Developing baseline implementations of the UART protocol for basic communication.
- Creating dynamic baud rate generation mechanisms to improve code configurability and flexibility.
- Improving the modularity of hardware designs to enhance code readability as well as aiding in its reusability.
- Managing buffer size constraints at both the transmitter and receiver ends to avoid data losses.
- Implementing additional codes to improve frame level integrity checks to detect data corruption.
- Developing baseline implementations of the SPI master side code in FPGA for interfacing with the slave modules.
- Supporting multiple modes of SPI communication to improve the ability to communicate with different slave devices.
- Integrating features for dynamically selecting the SPI clock frequency based on operational needs.
- Enhancing data width selection to improve flexibility thus allowing configuration of frame sizes.

Despite these advancements, existing literature has largely overlooked the potential of scanning the entire bit time, even with the availability of faster crystal oscillators on modern FPGA boards. Most designs still rely on traditional single-point sampling, which fails to leverage the higher clock frequencies offered by modern hardware. This thesis addresses this specific research gap by introducing a comprehensive bit-time scanning mechanism that enhances sampling reliability and overall communication robustness. Crucially, these proposed features are straightforward to integrate into existing digital designs without requiring extensive architectural modifications.

CHAPTER 3

METHODOLOGY

This chapter details the research design and mechanism of the proposed implementations of UART and SPI protocol. It also provides details of the work done to implement complementary modules. The chapter is divided into two main parts as follows.

- The first part is related to the all work done for UART protocol based communication in a system.
- The second part is related to the all work done for SPI protocol based communication in a system.

The following core topics are addressed in this chapter.

- System architecture overview for the UART protocol
- The proposed mechanism for the UART protocol
- Development of the enhanced UART receiver which will also have Self monitoring and glitch detection logic
- Development of UART transmitter having the capability to inject noise
- Development of traditional UART receivers to compare the behavior of all receivers
- Development of test bench for integrating and simulating the UART modules.

- System architecture overview for the SPI protocol based communication
- The proposed mechanism for the SPI protocol based communication
- Development of the simple SPI master code for proof of concept
- Development of the simple SPI slave code for proof of concept
- Development of the Specialized SPI slave code for testing the new features in simulations.
- Self-Monitoring and Anomaly Detection Logic for the SPI protocol based module

3.1 System Architecture Overview for the UART Protocol

The system consists of testing environment (both in simulation as well as in hardware) having one UART transmitter along with two traditional and one proposed UART receivers. The architecture is shown in the following Figure [3.1](#)

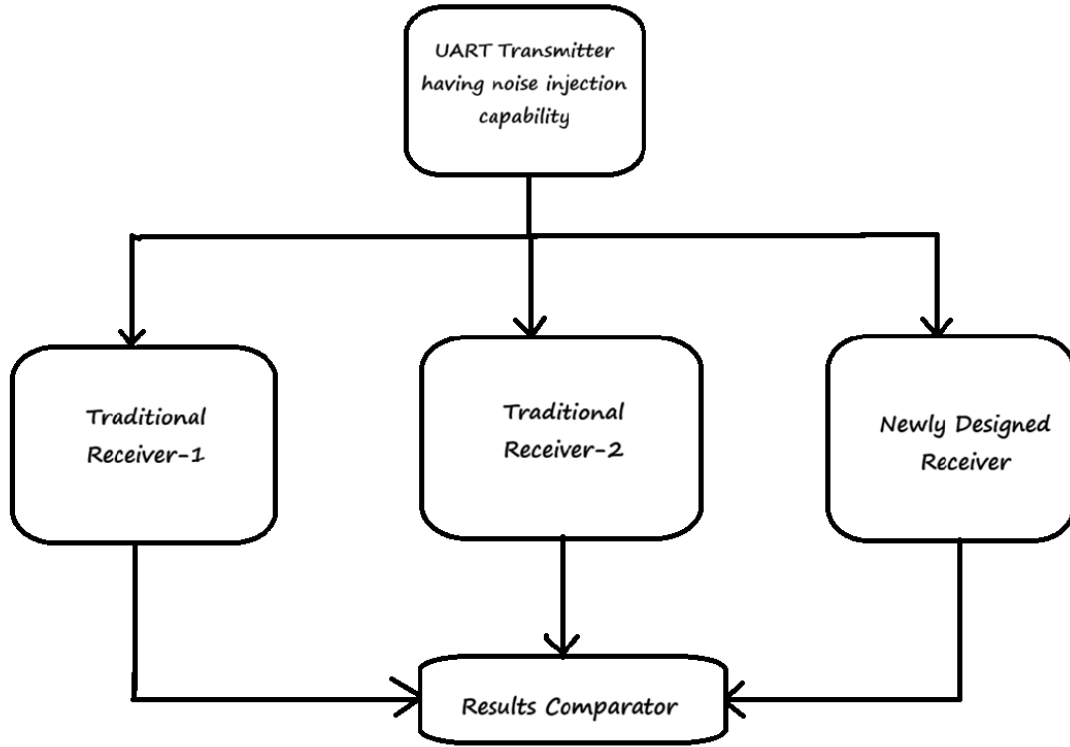


Figure 3.1: Block Diagram for Testing UART

Details of the components are as follows:

- UART trasnmitter developed in verilog having the capability to inject noise. The noise is fixed to see the exact behavior
- Traditional UART Receiver-1 developed in verilog which takes sample near the centre of bit duaration
- Traditional UART Receiver-2 developed in verilog which takes 3 samples near the centre of bit duaration and perform a majority voting
- Proposed UART Receiver developed in the style of SMRI in verilog which analyze the signal at its full capacity and perform a majority voting for decision as well as use counters to record anomalies

3.2 The Proposed Mechanism for the UART

The SMRI framework builds diagnostic capability into the receiver by over-sampling input lines and monitoring the incoming data stream thoroughly.

Rather than taking a single or three samples at the bit midpoint, SMRI used a continuous counter to record all zeros and ones within that single bit time. It then decided about the incoming bit to be one or zero. If number of ones were greater than number of zeros, it was considered as one else it was considered as zero. At the same time, if it detected a high frequency signal, it incremented the counter used for anomalies. Hence it detected the most accurate bit and at the same time also reported the unwanted glitches or dips.

3.3 Self-Monitoring and Glitch Detection Logic for the UART

Besides implementation of UART receiver with an improved style, two independent blocks were used to monitor the line status which are as follows:

- **Short Pulse Counter:** It was used to detect all the bytes having pulses whose duration is less than 75% of the bit time.
- **Low Accuracy Decisions Counter:** It takes decision exactly at bit boundaries when it expects the signal to be stable for at-least 75 percent time of the bit duration. It captured the anomalies when the bit duration was greater than 75% but there was a drift in the clock or bug at the transmitter side to calculate the bit duration.

The 75% threshold is just for anomaly detection, not for deciding binary data. The receiver still uses the standard 50% mark to read 0s and 1s. We established the 75% threshold to monitor timing margins.

Any pulse of duration less than 75% of the bit time will be declared as short pulse. Additionally, if a signal transition drifts by more than 25% of the bit time, it signals a major clock or data rate issue, triggering a warning before the data actually breaks down at the 50% mark. Moreover, this 75% value isn't hard coded; it is a fully customizable parameter in our design so users can change the percentage based on their environment.

The block diagram for Self Monitoring logic is shown in the following Figure 3.2

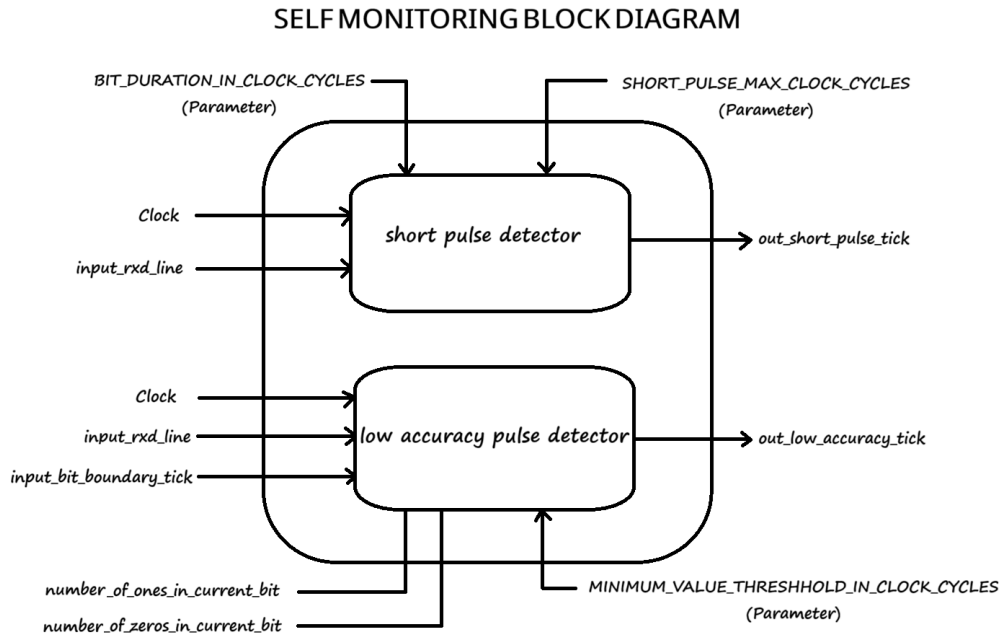


Figure 3.2: Block Diagram of Self Monitoring

3.4 Anomaly generator for the UART

To properly evaluate the SMRI receivers, a custom transmitter was implemented in Verilog. It was having the capability to inject clean and controlled anomalies into the output signal. The output was connected to the newly designed receiver and existing receivers simultaneously. The behavior of all receivers was analyzed and their details and results will be shared in the next chapter.

3.5 Traditional UART Receivers Implementation

Two traditional receivers were developed for experiments. The details were as follows:

- Traditional Receiver-1 was detecting start of the transfer by a high to low transition and taking sample exactly at central interval.

- Traditional Receiver-2 was detecting start of the transfer by a high to low transition and recording 3 samples near 50% duration of the bit and then performed majority voting in order to decode the bit received.

3.6 System Architecture Overview for the SPI Protocol

The system consists of testing environment (both in simulation as well as in hardware) having SPI protocol based master module written in a very simple way just for testing purposes but having the proposed feature along with an SPI protocol based slave module having the capability to violate timings whenever and wherever it is configured to for violation. The architecture is shown in the following figure [3.3](#)

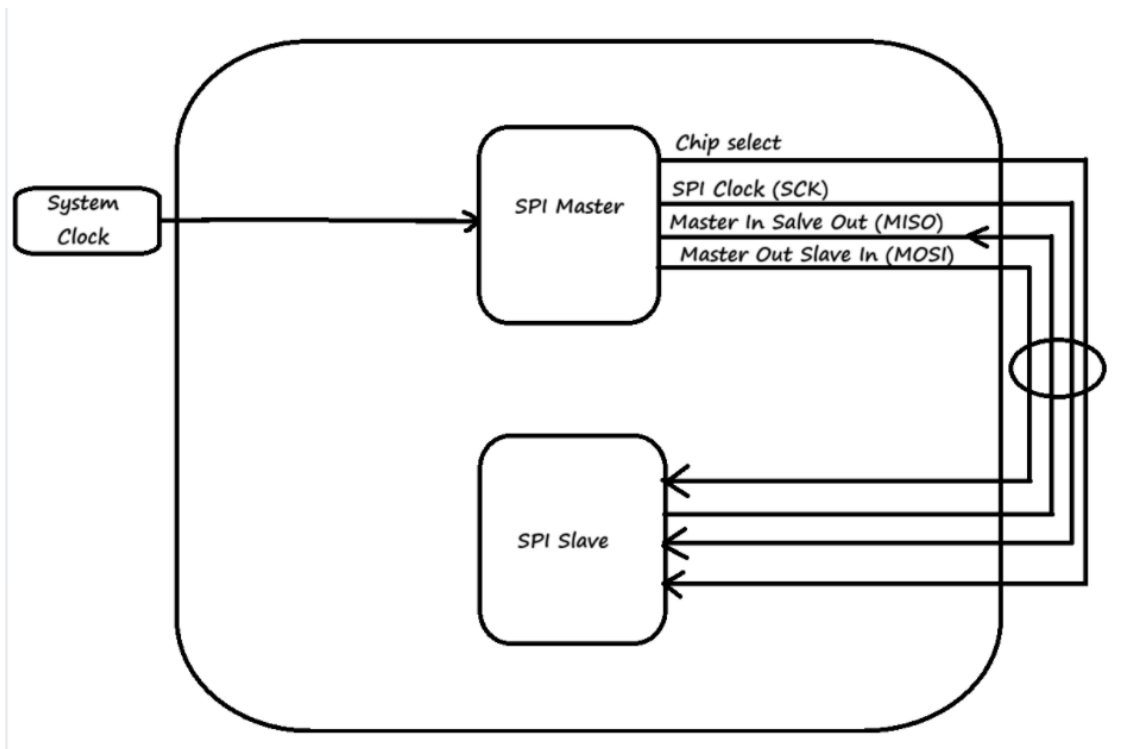


Figure 3.3: Block Diagram for Testing SPI

Details of the components are as follows:

- System clock is shown in the figure to clarify that this clock is given only to SPI protocol based master module.

- SPI protocol based master is shown which is responsible for transaction initiation and reporting errors also.
- SPI protocol based slave is shown which is developed to provide data to the master whenever it is asked to.

It is also mentioned that in real world scenarios, most of the times, the master and the slave are not on the same chip but, here in this case, the set up was simplified just for the sake of testing and verification. Although the delays were inserted to simulate the real world delays.

3.7 The proposed Mechanism for the SPI protocol based communication

In the same way as UART, the whole clock cycle of the SPI clock was scanned using main clock and reported anomaly when the transition came very near to sampling edge or just after sampling edge. Although data was sampled using spi clock, the main clock was just used in parallel just to analyze the transition time.

3.8 Development of the simple SPI master code for proof of concept

A simple SPI protocol based master code was developed for testing. The proposed SMRI framework was also added to this simple code. The newly added block was just generating simple one clock cycle pulse whenever it detected a transition in the area where it was not allowed. Further, counters were used to count the anomalies per transaction or per bit as required.

3.9 Development of the simple SPI slave code for proof of concept

A simple SPI protocol based slave code was developed for testing. It was just capable of responding to the master whenever it was asked.

3.10 Development of the Specialized SPI slave code for testing the new features in simulations

A specialized SPI protocol based slave code was designed and developed which was having the capability to delay a transition in simulations. It was just capable of inserting delay in response time whenever and wherever it was programmed to do so. The new block diagram was as follows [3.4](#)

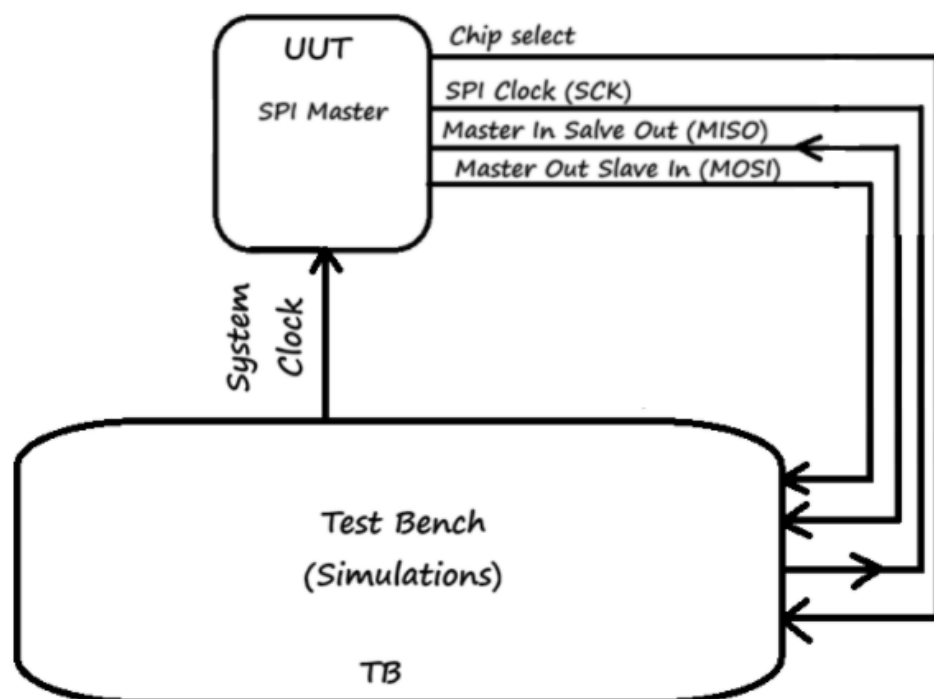


Figure 3.4: Block Diagram for Testing SPI Having Anomalies

3.11 Self-Monitoring and Anomaly Detection Logic for SPI

Since self monitoring and anomaly detection are not requirements of standard SPI protocol, it is always better to develop an independent logic in FPGA for such purposes in order to avoid complex designs changes. For this reason, to develop an independent block to monitor the incoming data line, the following steps were performed:

- The whole SPI clock cycle duration was analyzed and divided into the following two categories.
 - The area where normal transition was allowed
 - The area where normal transition was not allowed
- An additional signal was used to indicate that on going transfer is occurring and it has entered to the area where transition was not allowed
- A one clock cycle pulse was generated when the transition was detected in that area

In order to provide the details about how to divide the SPI clock duration into two categories, an example waveform is provided in the following figure [3.5](#)

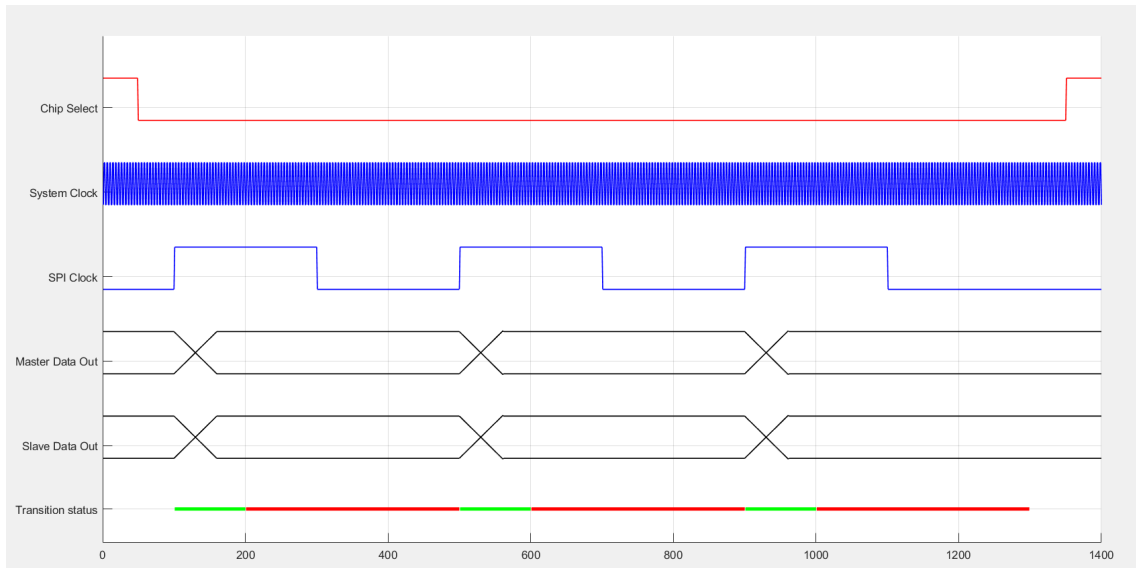


Figure 3.5: SPI Clock Cycle Area Division

In the above figure, The transition status line is drawn such that the allowed area for transition is represented with green color and the area where transition is not allowed is drawn with red color. The area where no line is visible is considered as "do not care" area so that the anomaly counter may be incremented only when the transition is detected in the area where it is not allowed.

After this logic was inserted, additional simple block that was inserted to obtain clear results which are as follows.

- Counter was used to count all unwanted transitions
- Counter was used to increment at bit boundary if there were one or more invalid unwanted transitions
- Counter was used that was only incremented after complete transaction if there were one or more unwanted transitions captured.

These methods are suggested to keep the additional blocks simple and effective.

3.12 Short Summary

This summary is being provided for the technical individuals who are already familiar with design and development of various communication protocols, and they just need summarized hints to include these advanced monitoring features in their designs. So, no matter which protocol is used for communication, the fundamental steps to include monitoring techniques will be as follows:

- Read the data sheet or any protocol document for the required communication.
- Identify the idle state of a signal
- Identify the maximum toggling frequency of the signal when communication is active
- Identify the moments when transitions are allowed and when they are not allowed
- Try to detect any transition where they are not allowed at all, for example, in idle state.
- Try to detect transition in the interval where the signal should remain stable within a communication window.
- One may also simply detect the pulses whose width is less than the shortest possible width defined by that protocol.
- Use one clock cycle pulses to detect such anomalies and use single bits to report these to avoid extra hardware and power usage. However, these things depend on resources available and information needed for reporting.
- Keep these detections simple and independent of the main logic so that the design may remain simple.

3.13 Conclusions

To conclude this chapter, it detailed the design and verification process for the enhanced UART receiver along with the design and development of the other modules needed for testing and verification. The modules were coded in Verilog with a focus on improving data reliability, reducing issues due to noise, and fixing timing issues. It also presented the changes in SPI receivers in detail. The methodology covered most of things needed for design and development of the proposed functionalities along with the suggestions to keep it simple, clean, and meaningful. Now that the design framework and testing steps are fully established, Chapter 4 will present the experiments, testing, simulation and implementation results.

CHAPTER 4

EXPERIMENTS AND RESULTS

This chapter presents the testing performed during this research. The designed and developed firmware as discussed in chapter 3 were integrated and a test bench was developed to perform testing and analyze results. This chapter also has two main parts as follows.

- The first part is related to the all testing done and results obtained for the UART protocol based communication in a system.
- The second part is related to the all work done for SPI protocol based communication in a system.

The following points are addressed in this chapter.

- Micro level details of the transmitter designed and implemented specially for testing the newly designed receiver for the communication based on the UART protocol.
- Screen shots of the waveform taken after inserting noise at multiple locations within a bit time.
- Suggestion regarding how to embed the proposed logic in the existing UART protocol based receivers for anomaly reporting.
- Results in tabular form summarized after the testing in simulations of UART protocol based communication.

- Results in tabular form summarized after the testing in implementations of UART protocol based communication.
- Conclusions

4.1 SPI based Master Simulations

The newly developed SPI based Master code was tested only in simulations. The test bench generated responses having transitions both in the "Transition Allowed" region and the "Transition Not Allowed" region. The master side Self-Monitoring successfully detected the transition that occurred in the "Transition Not Allowed" region. The detailed testing was performed for UART protocol which are discussed in the next sections.

4.2 Simulations details of the UART protocol based communication

The transmitter was designed in such a way that it was transmitting 100 bytes. It was having the capability to inject noise in selected bit of the selected byte. In this selected example for simulation, noise injection details were as follows:

- Data byte fixed to 8 bits hexadecimal AA
- Fourth bit was selected for noise which was 1 bit 1 originally.
- Total bit duration was 1000ns
- Noisy pulse duration was 50 to 200 nanoseconds, such that the bit became low for the said duration.
- At 10th byte, when the said bit duration was started. It became low for 125ns after approximately 125ns
- At 20th byte, when the said bit duration was started. It became low for 125ns after approximately 312ns

- At 30th byte, when the said bit duration was started. It became low for 125ns after approximately 500ns
- At 40th byte, when the said bit duration was started. It became low for approximately 62ns after approximately 687ns
- At 50th byte, when the said bit duration was started. It became low for approximately 62ns after approximately 812ns

Waveform showing noise insertion at 10th byte is as follows [4.1](#)

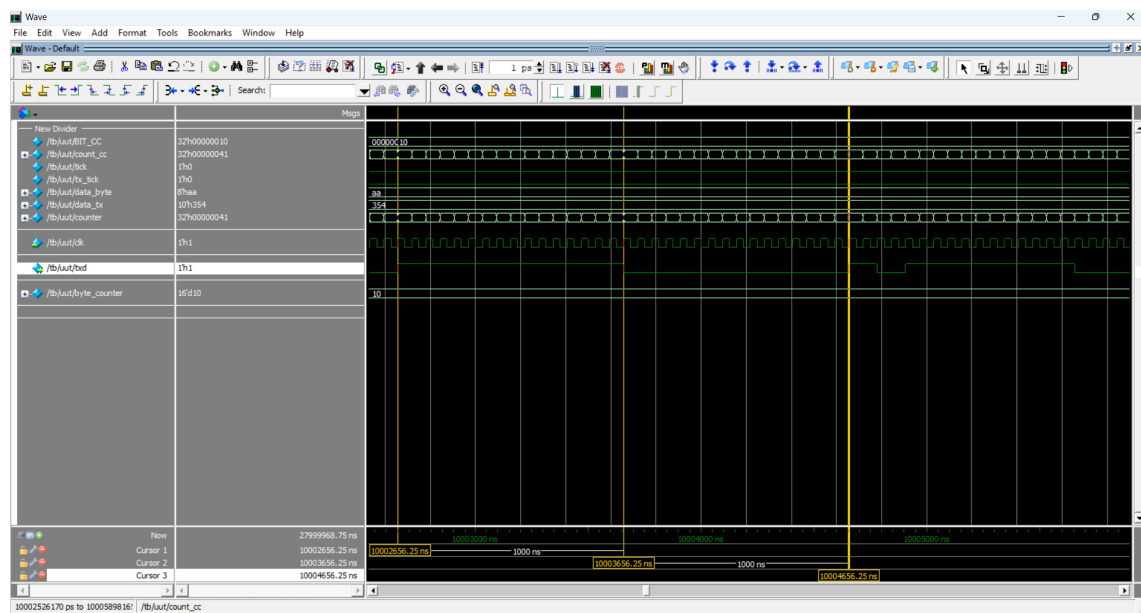


Figure 4.1: UART Transmitter Waveform showing noise insertion at 10th byte

Detail description of the signals shown in the above figure are as follows:
Clk is the clock used which is 16 megahertz.

BIT CC denotes duration of a single bit in the form of clock cycles, since here we are using a clock of sixteen megahertz and a data rate of 1 mega bit per second, so the duration of the single bit will be equal to 16 clock cycles which is equals to 1000 nanoseconds.

Data byte denotes the data transmitting currently which is 8 bits.

Data tx (10 bits) denotes the data symbol which is 8-bit data along with start and stop bit.

Byte Counter denotes the byte currently transmitting. As shown, it is the 10th byte transmitting currently.

Moreover, the noise pulse of two clock cycles is also shown causing a dip in the transmitting signal for two clock cycles. The noise is inserted after 125 nanoseconds of the bit start time and the duration of noise is 125 nanoseconds. The data is normal for all other bits of the current byte.

Waveform showing noise insertion at 20th byte is as follows [4.2](#)

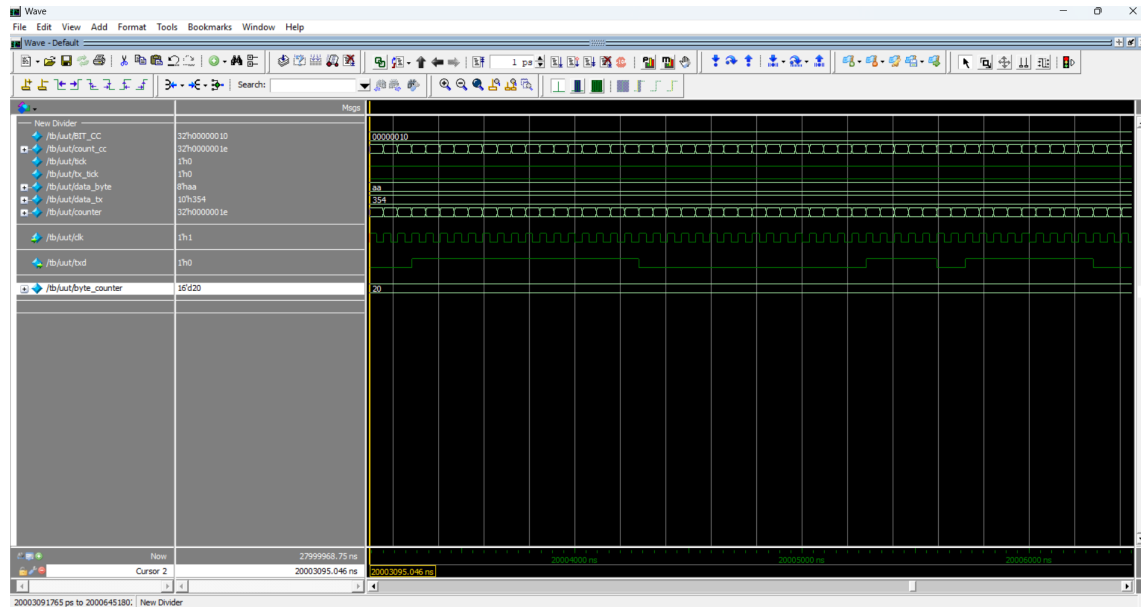


Figure 4.2: UART Transmitter Waveform showing noise insertion at 20th byte

Detail description of the signals shown in the above figure are as follows:
Clk is the clock used which is 16 megahertz.

BIT CC denotes duration of a single bit in the form of clock cycles, since here we are using a clock of sixteen megahertz and a data rate of 1 mega bit per second, so the duration of the single bit will be equal to 16 clock cycles which is equals to 1000 nanoseconds.

Data byte denotes the data transmitting currently which is 8 bits.

Data tx (10 bits) denotes the data symbol which is 8-bit data along with start and stop bit.

Byte Counter denotes the byte currently transmitting. As shown, it is the 20th byte transmitting currently.

Moreover, the noise pulse of two clock cycles is also shown causing a dip in the transmitting signal for two clock cycles. The noise is inserted after

approximately 312 nanoseconds of the bit start time and the duration of noise is 125 nanoseconds. The data is normal for all other bits of the current byte.

Waveform showing noise insertion at 30th byte is as follows [4.3](#)

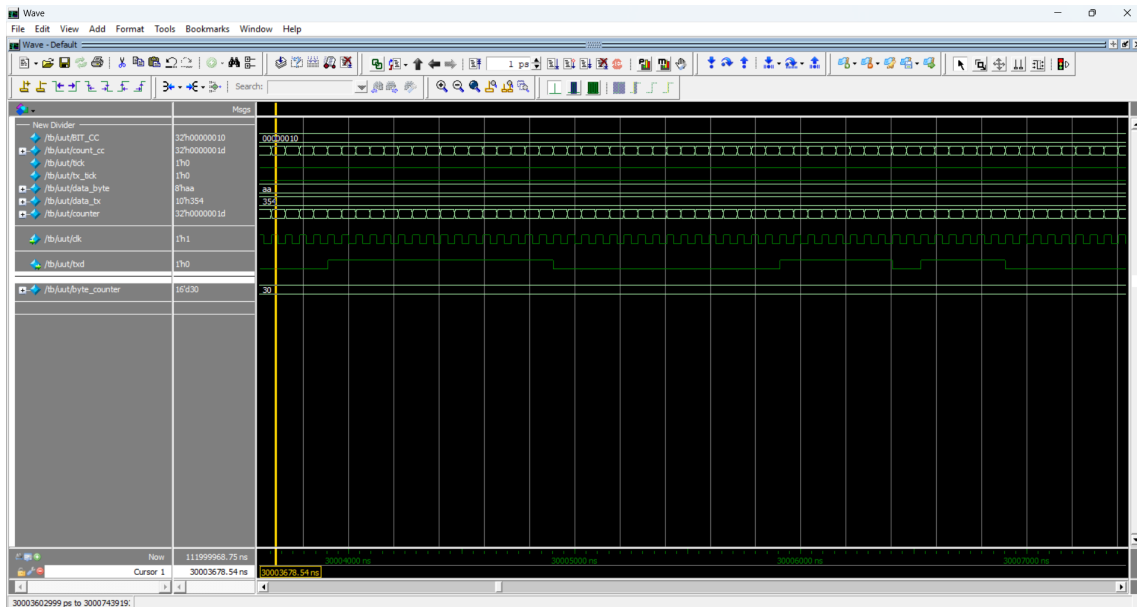


Figure 4.3: UART Transmitter Waveform showing noise insertion at 30th byte

Detail description of the signals shown in the above figure are as follows:
Clk is the clock used which is 16 megahertz.

BIT CC denotes duration of a single bit in the form of clock cycles, since here we are using a clock of sixteen megahertz and a data rate of 1 mega bit per second, so the duration of the single bit will be equal to 16 clock cycles which is equals to 1000 nanoseconds.

Data byte denotes the data transmitting currently which is 8 bits.

Data tx (10 bits) denotes the data symbol which is 8-bit data along with start and stop bit.

Byte Counter denotes the byte currently transmitting. As shown, it is the 30th byte transmitting currently.

Moreover, the noise pulse of two clock cycles is also shown causing a dip in the transmitting signal for two clock cycles. The noise is inserted after approximately 500 nanoseconds of the bit start time and the duration of

noise is 125 nanoseconds. The data is normal for all other bits of the current byte.

Waveform showing noise insertion at 40th byte is as follows [4.4](#)

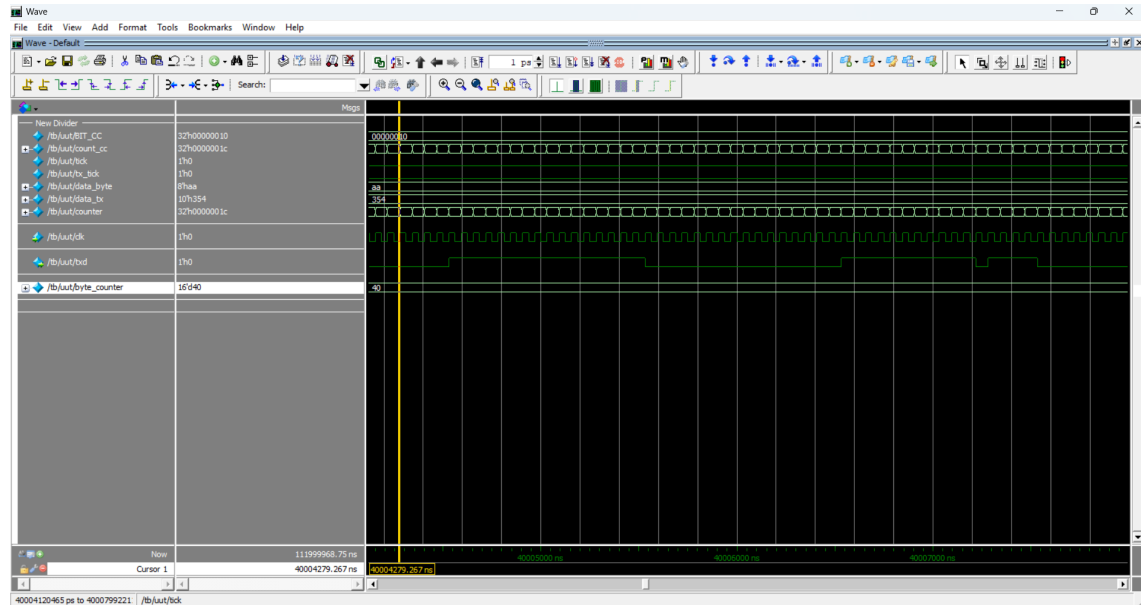


Figure 4.4: UART Transmitter Waveform showing noise insertion at 40th byte

Detail description of the signals shown in the above figure are as follows:
Clk is the clock used which is 16 megahertz.

BIT CC denotes duration of a single bit in the form of clock cycles, since here we are using a clock of sixteen megahertz and a data rate of 1 mega bit per second, so the duration of the single bit will be equal to 16 clock cycles which is equals to 1000 nanoseconds.

Data byte denotes the data transmitting currently which is 8 bits.

Data tx (10 bits) denotes the data symbol which is 8-bit data along with start and stop bit.

Byte Counter denotes the byte currently transmitting. As shown, it is the 40th byte transmitting currently.

Moreover, the noise pulse of two clock cycles is also shown causing a dip in the transmitting signal for one clock cycles. The noise is inserted after approximately 687 nanoseconds of the bit start time and the duration of noise is approximately 62 nanoseconds. The data is normal for all other

bits of the current byte.

Waveform showing noise insertion at 50th byte is as follows [4.5](#)

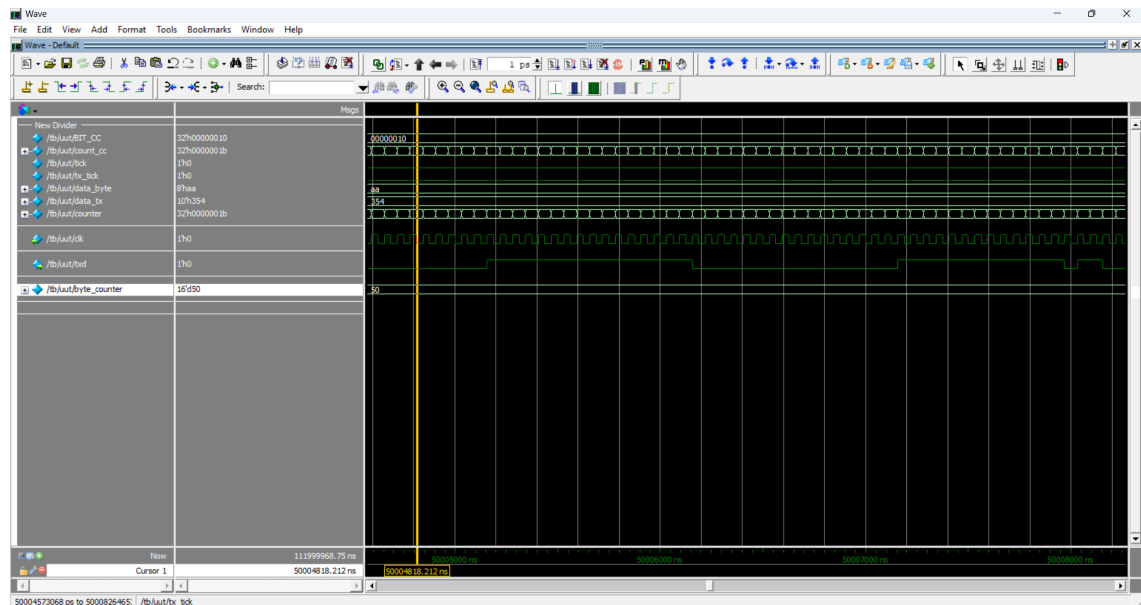


Figure 4.5: UART Transmitter Waveform showing noise insertion at 50th byte

Detail description of the signals shown in the above figure are as follows:
Clk is the clock used which is 16 megahertz.

BIT CC denotes duration of a single bit in the form of clock cycles, since here we are using a clock of sixteen megahertz and a data rate of 1 mega bit per second, so the duration of the single bit will be equal to 16 clock cycles which is equals to 1000 nanoseconds.

Data byte denotes the data transmitting currently which is 8 bits.

Data tx (10 bits) denotes the data symbol which is 8-bit data along with start and stop bit.

Byte Counter denotes the byte currently transmitting. As shown, it is the 50th byte transmitting currently.

Moreover, the noise pulse of two clock cycles is also shown causing a dip in the transmitting signal for one clock cycles. The noise is inserted after approximately 812 nanoseconds of the bit start time and the duration of noise is approximately 62 nanoseconds. The data is normal for all other

- Increment the counter at byte boundary if the requirement is to count number of bytes where short pulses occurred.

The short pulses counts shown in the summarized results represent the number of bytes having short pulses. In this way, the number of short pulses detected were exactly equal to the number of bytes where noise was injected.

This simple method is capable of providing detailed information about the noise detected. However, the only important information for the designer and developers is to know whether some anomaly exist or not. Once this information is obtained, the remaining troubleshooting is not a big issue.

It is also mentioned that if there only limited bits are available for additional information, the designer may report only in a single associated with each byte to report the anomaly.

In some cases, a single bit can be allocated as a whole just to report that the anomaly has occurred at least once. In this case, it is suggested to always to not set this bit at power up, since there is always a possibility of glitches or spikes when the module is powering up or going down.

The remaining waveforms are not shown since the noise location was more or less same in the remaining noisy bytes.

The overall results of traditional receiver-1, traditional receiver-2 and the newly developed receiver are summarized in the following table.

Experimental Metric	Traditional Receiver 1 (Deciding only on central sample)	Traditional Receiver 2 (Deciding on central 3 samples)	Newly designed Receiver
Total Transmitted Bytes	100	100	100
Noisy Bytes Injected	10	10	10
Noise Pulse Width	Short Pulses at various locations	Short Pulses at various locations	Short Pulses at various locations
Successfully Decoded Bytes	98	99	100
Glitches/Anomalies Detected	9 (Blind)	0 (Blind)	10/10
Data Corruption	2%	1%	0%
Glitch Detection Accuracy (%)	0%	0%	100%

Table 4.1: Summarized Results of testing in simulations of UART protocol based communication

It has been verified that the newly designed receiver successfully received all the data correctly and at the same time reported the anomalies so that the problems causing these analogies may be mitigated promptly while the traditional receiver have detected either the data correctly or incorrectly in some cases and have no functionality of reporting anomalies.

4.3 Implementations of UART protocol based communicating modules

In the same way all these modules were synthesized and implemented in FPGA and tested. The details were as follows:

- Since the activities may performed very faster in hardware as compared to simulations. Therefore, the testing samples were increased from 100 Bytes to 1000000 Bytes.
- Glitches having variable width were also introduced to check its behavior.
- The results were recorded in a frame and transmitted serially outside FPGA.
- The results were recorded in the PC.

The results are summarized in the following table.

Experimental Metric	Traditional Receiver 1 (Deciding only on central sample)	Traditional Receiver 2 (Deciding on central 3 samples)	Newly designed Receiver
Total Transmitted Bytes	1000000	1000000	1000000
Noise Pulses of 5ns each (0.5% of the bit duration)	14000	14000	14000
Noise Pulses of 50ns each (5% of the bit duration)	14000	14000	14000
Noise Pulses of 70ns each (7% of the bit duration)	14000	14000	14000
Noise Pulses of 150ns each (15% of the bit duration)	15000	15000	15000
Noise Pulses of 200ns each (20% of the bit duration)	15000	15000	15000
Noise Pulses of 400ns each (40% of the bit duration)	14000	14000	14000
Noise Pulses of 800ns each (80% of the bit duration)	14000	14000	14000
Total Noisy Bytes	100000	100000	100000
Noisy Bytes (%)	10%	10%	10%
Successfully Decoded Bytes	931201	966427	986000
Glitches/Anomalies Detected	0 (Blind)	0 (Blind)	96714
Data Corruption	6.8%	3.35%	1.4%
Glitch Detection Accuracy (%)	0%	0%	96.7%

Table 4.2: Summarized Results of testing in hardware

It is also mentioned that the noise injected was strictly controlled to analyze the exact behavior of the newly designed receiver and the existing receivers.

Because doing so ensures our thinking to be aligned with the possible cases and the associated outcomes. Also in this way, one can verify the behavior. The results were exactly the same as expected. However, in real world scenarios, the results may vary.

The resource utilization is summarized as follows:

Design	LUT Utilization	Flip-Flop Utilization	Power
Traditional Receiver	19	40	0.063 W
Enhanced Receiver (Without Self Monitoring)	46	45	0.0634 W
Enhanced Receiver (With Self Monitoring)	58	54	0.064 W

Table 4.3: Resources Summary

As shown in the above table, it has negligible impact on power as compared to its usefulness. Also it is mentioned that modern FPGA's have thousands of LUT tables available and millions of Flip Flops so the impact on Flip Flops and LUTs is also negligible. It is further mentioned that these figures may further be reduced depending on the requirements.

4.4 Conclusions

In conclusion, this chapter successfully evaluated the newly designed UART receiver through a series of systematic experiments. The results demonstrate that the existing receivers lack these enhancements which are very useful and simple to integrate in existing firmware. The quantitative outcomes establish a clear baseline for understanding the existing shortcomings and highlighting the efficacy of the proposed approach.

Overall, the data presented herein fulfill the experimental objectives of

this study and lay the groundwork for the comprehensive analysis and theoretical implications discussed in the following chapter.

CHAPTER 5

CONCLUSIONS & PERSPECTIVES

This final chapter concludes the thesis by reviewing the limitations of traditional FPGA UART receivers and presenting options for improvement. The research demonstrated that standard implementations lack the anomaly detection capabilities. While these errors can be corrected if the noise is within certain limits. The following sections summarize the work done and outline perspectives for future.

5.1 Conclusion

This thesis successfully proposed, designed and implemented a Self Monitoring Receivers Implementation which has the capability to successfully detect the anomalies in the receiving line. Unlike traditional receivers, it was also able to decode the correct data when the noise glitch did not cross some threshold. It also detected some of the anomalies where actual data recovery was impossible but it reported that it has detected some abnormal transitions and the data has low accuracy. The traditional and newly developed receivers were tested with feeding same data line to all receivers simultaneously. The results were found satisfactory with almost negligible overhead in hardware resources. The traditional receivers failed to decode data correctly in some cases and also missed some anomalies already present while the new receiver tried to correct data up to some extent and reported almost 100% controlled errors.

Moreover, details on how to embed this functionality in SPI communication

are also provided.

5.2 Research Perspective

Looking at the bigger picture, the methodology developed in this thesis provides possibilities for its use in many ways, some of which are as follows:

- **Early Warning Systems for Industry:** Traditional receivers currently working in industries only report an error when the data is corrupted. By adding this self monitoring logic, devices can warn the user that a wire is getting noisy even before the data corruption. This allows for proactive maintenance in factories and critical systems.
- **Use of Artificial Intelligence:** The core idea may be generalized using AI algorithms to automatically calculate the ranges for anomalies for any data rates and clock frequencies
- **Expanding to Other Protocols:** The core idea of using high-frequency oversampling to catch tiny glitches does not have to stop at UART. This exact same approach can be used to develop independent monitoring blocks while developing other commonly used communication standards, such as I²C, or CAN bus. No additional crystals are required for such purposes since modern FPGAs card are having crystals having frequency in several Mega Hertz while the communication protocols normally operate at much lower speeds than crystal frequency.

REFERENCES

- [1] Ali, M. A., Pasha, M. A., Masud, S. and Al Ghazali, F. FPGA Design of a Reconfigurable UART IP Core and its SoC Implementation for IoT Applications. 2023 25th International Multitopic Conference (INMIC). IEEE. 2023. 1–6. Cited on pp. 14 and 19.
- [2] Li, Z. Design, refinement, and enhancement of FPGA-implemented UART circuitry. Applied and Computational Engineering. 2024. 37: 259–265. Cited on pp. 15 and 20.
- [3] Xiang, R., Wang, J., Chen, B. and Lyu, P. FPGA-based UART Transmission for VGA Image Display. Proceedings of the 2023 International Conference on Advances in Artificial Intelligence and Applications. 2023. 201–207. Cited on pp. 15 and 20.
- [4] Xing, Y. Hardware circuit implementation of transmitter and receiver based on UART protocol. Applied and Computational Engineering. 2024. 72: 204–210. Cited on pp. 15 and 20.
- [5] Fang, Y.-y. and Chen, X.-j. Design and Simulation of UART Serial Communication Module Based on VHDL. 2011 3rd International Workshop on Intelligent Systems and Applications. 2011. 1–4. Cited on pp. 15 and 20.
- [6] Saha, S., Rahman, M. A. and Thakur, A. Design and implementation of SPI bus protocol with Built-in-self-test capability over FPGA. 2014 International Conference on Electrical Engineering and Information Communication Technology. 2014. 1–6. Cited on pp. 16 and 20.
- [7] Qiang, J., Gu, Y. and Chen, G. FPGA Implementation of SPI Bus Communication Based on State Machine Method. Journal of

- Physics: Conference Series. 2020. 1449(1): 012027. [Online]. Available: <https://doi.org/10.1088/1742-6596/1449/1/012027> Cited on pp. 16 and 20.
- [8] Pahlevi, R. R., Putrada S., A. G. and Abdurohman, M. Fast UART and SPI Protocol for Scalable IoT Platform. 2018 6th International Conference on Information and Communication Technology (ICoICT). 2018. 239–244. Cited on pp. 16 and 20.
- [9] Río, T., Oliva, L. and Juárez-Gracia, A. Implementation of the communication protocols SPI and I2C using a FPGA by the HDL-Verilog language. Research in Computing Science. 2014. 75: 31–41. Cited on pp. 17 and 21.
- [10] Liu, Z. The Hardware Implementation of a Register-configurable SPI Interface. Science and Technology of Engineering, Chemistry and Environmental Protection. 2024. 1. Cited on pp. 17 and 21.
- [11] Kopparapu, J. and Rambabu, S. An Effective FPGA Based Real Time System for Measuring Temperature and Light Intensity Using UART Protocol. 2025 6th International Conference on Mobile Computing and Sustainable Informatics (ICMCSI). 2025. 439–443. Cited on pp. 17 and 21.
- [12] Ganeswara Rao, M. V., Sumalatha, M., Krishna, K. H., Jaiswai, N., Kumar, P. R. and Himasri, P. S. S. S. Design and development of FPGA Architecture for Low-Power UART for Smart Homes. 2024 OPJU International Technology Conference (OTCON) on Smart Computing for Innovation and Advancement in Industry 4.0. 2024. 1–6. Cited on pp. 17 and 21.
- [13] Raju, S. S., Wu, S. and Wang, N. Design and Implementation of a Multi-Protocol Converter Supporting SPI, I2C, and UART Interfaces. 2025 8th International Conference on Information Communication

- and Signal Processing (ICICSP). 2025. 677–681. Cited on pp. 17 and 21.
- [14] Penurkar, S., Kumar, A., Srivastava, S. and Nakhate, S. Design and Implementation of Single Master Single Slave Serial Peripheral Interface (SPI) on FPGA. 2025 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS). 2025. 1–7. Cited on pp. 18 and 22.
- [15] Xie, R. Design and Simulation of UART Protocol Based on FPGA. 2024 6th International Conference on Applied Machine Learning (ICAML). 2024. 551–557. Cited on pp. 18 and 22.
- [16] Maity, A., Samanta, P. K., De, B. P., Dash, S. K., Bhowmik, W. and Bakshi, A. FPGA Implementation of Serial Peripheral Interface Transceiver Module. 2024 International Conference on Computer, Electrical Communication Engineering (ICCECE). 2024. 1–5. Cited on pp. 18 and 22.
- [17] D'Silva, E. J. and Yadav, S. Implementation of a CAN-to-SPI Protocol Bridge for Automotive ECUs Using FPGA. 2025 9th International Conference on Inventive Systems and Control (ICISC). 2025. 1819–1822. Cited on pp. 18 and 22.
- [18] Trehan, M., Kumar, P. and Gaur, N. Multi-Protocol Conversion Unit Implementation on a FPGA. 2024 2nd World Conference on Communication Computing (WCONF). 2024. 1–7. Cited on pp. 18 and 22.
- [19] Sonwane, R., Deshmukh, A. and Choudhary, S. UART (Universal Asynchronous Receiver Transmitter) for Serial Data Communication: Design and Implementation on FPGA Platform. 2023 2nd International Conference on Futuristic Technologies (INCOFT). 2023. 1–4. Cited on pp. 18 and 23.

- [20] Haripriya, D., Kumar, K., Shrivastava, A., Al-Khafaji, H. M. R., Moyal, V. and Singh, S. K. Energy-Efficient UART Design on FPGA Using Dynamic Voltage Scaling for Green Communication in Industrial Sector. *Wireless Communications and Mobile Computing*. 2022. 2022(1): 4336647. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1155/2022/4336647> Cited on pp. 19 and 23.
- [21] Kumar, V., Rastogi, A. and Tomar, V. Implementation of UART Design for RF Modules Using Different FPGA Technologies. *IOP Conference Series: Materials Science and Engineering*. 2021. 1116(1): 012131. [Online]. Available: <https://doi.org/10.1088/1757-899X/1116/1/012131> Cited on pp. 19 and 23.