International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 01 | Jan - 2025 SJIF Rating: 8.448 ISSN: 2582-3930

Ensuring I2C Bus Reliability in Nanosatellite Operations Fault Analysis and Mitigation Strategies

MS.G.AJITHA
Department of Electronics and Communication Engineering,
Institute of Aeronautical Engineering, Hyderabad
g.ajitha@iare.ac.in

G. HARSHAVARDHAN
Department of Electronics and Communication Engineering,
Institute of Aeronautical Engineering, Hyderabad
21951A0460@iare.ac.in

S. JAYANTH
Department of Electronics and Communication Engineering,
Institute of Aeronautical Engineering, Hyderabad
21951A0465@iare.ac.in

A.GOWTHAM RAO
Department of Electronics and Communication Engineering,
Institute of Aeronautical Engineering, Hyderabad
21951A0456@iare.ac.in

Abstract— ***Despite being widely used in nanosatellite operations, the I2C bus has dependability issues that can result in catastrophic failures. In the framework of CubeSat missions, this study performs an extensive fault analysis of the I2C bus, looking into necessary hardware and software, important failure reasons, and possible mitigation techniques. To gain a comprehensive understanding of I2C bus characteristics, failure modes, and operational requirements, experimental testing is used. This work presents checksum strategies to improve data integrity and reliability during I2C bus transactions in addition to conventional fault analysis techniques. Checksums are incorporated into the communication protocol to greatly increase data transfer dependability and lower the chance of mission-critical failures. Through qualitative risk analysis, the suggested mitigation strategies are assessed, emphasizing their influence on the overall mission success and CubeSat health. This study emphasizes how important it is to take into account both runtime fault management and strong design considerations while reducing I2C bus vulnerabilities to maintain operational reliability for nanosatellite missions.***

Keywords*: Nanosatellites, CubeSats, I2C bus, reliability, fault mitigation, checksums*.

I. INTRODUCTION

Due to its ability to provide affordable access to space for a range of commercial and scientific uses, nanosatellites, also known as CubeSats, have completely changed space exploration. The performance, success rates, and trends of the first 100 CubeSat missions are statistically analyzed in this article[1]. The Inter Integrated Circuit (I2C) bus is a widely used protocol because of its ease of use, low power consumption, and adaptability in connecting various electronics in the limited space of a CubeSat [2]. Examination of CubeSat missions statistically [3].

However, the success of nanosatellite missions depends on the dependability of communication protocols like the I2C bus[3][4].

Because CubeSats are operated under strict size, weight, and power limits, as opposed to bigger spacecraft, communication failures can have a greater impact on mission outcomes. With an emphasis on performance concerns and design constraints, this study examines the implementation and dependability of electrical bus interfaces used in CubeSats[4].

With a particular focus on the I2C bus, this research investigates the complexities of protocol dependability in nanosatellite missions[5]. We explore the difficulties brought about by protocol malfunctions, look at the hardware and software prerequisites for dependable performance, and assess the main causes of protocol vulnerabilities [5] [6].

II. THE MAIN TEXT

Developed by Philips Semiconductor (now NXP Semiconductors) in the early 1980s, the Inter-Integrated Circuit (I2C) protocol, pronounced as "I-squared-C," is a popular synchronous serial communication protocol[5]. It was intended to reduce the amount of wires needed for communication between integrated circuits (ICs) on a circuit board. Fundamentally, a serial data line (SDA) and a serial clock line (SCL) are the only two signal wires needed for two-way communication between several master and slave devices under the I2C protocol[6]. Because of its simplicity, it is perfect for uses where power and space are limited, including in embedded systems and nanosatellites. In an I2C system[7], a master device or many master devices establish communication and manage the bus, while slave devices carry out the master's instructions. Multiple devices can share a bus thanks to this architecture, which improves scalability and flexibility.

A clock signal produced by the master device synchronizes data transfer on the I2C bus. Eight-bit bytes are used to send data, with the most significant bit (MSB) being sent first.

Over the same bus, devices can send and receive data as master and slave devices. The master device can choose which slaves to communicate with by assigning each slave on the I2C bus a distinct 7-bit or 10-bit address. Up to 128 devices can be supported by the 7-bit addressing system, while up to 1024 devices can be supported by the 10-bit addressing scheme[6][7]. numerous master arrangements, in which numerous master devices can start a bus conversation, are supported by advanced I2C systems. Data corruption and bus contention are avoided by arbitration logic, which makes sure that only one master device is able to manage the bus at once. Clock stretching is the ability of slaves to slow down the master's clock in order to process data or carry out other operations[8]. Slower devices can communicate efficiently without flooding the bus thanks to this approach. A start condition (S) initiates communication on the I2C bus, while a stop condition (P) ends it. A data transfer sequence is indicated to begin and finish by these conditions, respectively[6].

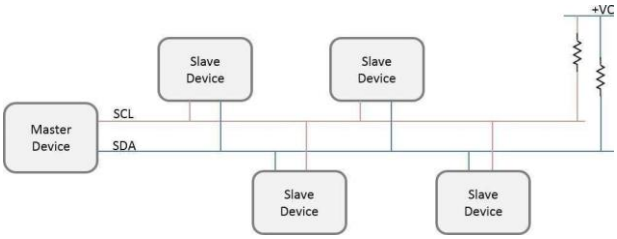


Figure 1. I2c bus Architecture

I2C is a low power, multi-device compatible protocol that is widely used by nanosatellites, often known as CubeSats, to connect onboard components such as sensors, actuators, memory modules, and communication subsystems[8]. Because of this, I2C is perfect for the small size and limited resources of CubeSats. Wide temperature changes that can impair device performance and reliability, as well as single-event upsets (SEUs) brought on by cosmic radiation, pose serious problems for I2C in the space environment[9]. CubeSat missions need to have strong fault detection, error management, and mitigation procedures in place to deal with these problems and avoid mission-critical failures in order to sustain reliable communication[8].

The onboard computer (master) can connect with peripheral devices (slaves) thanks to the master-slave design of the I2C protocol[5]. However, as missions get more complicated, advances in fault analysis and enhancements to the I2C protocol will become increasingly important[7]. Future CubeSat missions depend on these improvements to be dependable and successful, especially as their reach and operating difficulties increase[9] in the hostile environment of space.

CubeSats employ the I2C bus to allow real-time communication between peripherals like cameras, gyroscopes, and sensors and the master device, which is often the onboard computer[10]. Eight-bit bytes of data are transferred, synchronized with the master clock signal, and acknowledged by the recipient device.

The I2C bus gives each slave device a distinct 7-bit or 10-bit address, enabling the master to speak with particular devices on an individual basis[11]. With the 10-bit approach, this capacity is increased to 1024, but the 7-bit addressing system can support up to 128 devices. Multi-master arrangements, in which several masters can each manage a bus, are also possible with advanced I2C setups. Conflicts can be resolved by arbitration logic.

When a device requires more time to finish internal processes, it can be helpful for slaves to execute clock stretching in order to slow down the master's clock[8]. A start condition signals the start of a data transmission sequence on the bus, while a stop condition signals its conclusion[10].

A straightforward, inexpensive data transfer protocol called the Inter-Integrated Circuit (I2C) bus was created to facilitate short-range communication between microcontrollers and peripheral devices. Its ease of use and inexpensive cost of production contribute to its appeal. I2C allows for both single- and multi-master setups, with two lines—the Serial Clock Line (SCL) for clock signals and the Serial Data Line (SDA) for bidirectional data flow—being used to transfer data between devices. Pullup resistors, which are linked to both lines, aid in determining the bus speed, which is normally 400 kbps for fast mode and 100 kbps for standard mode[9]. Changes in the SDA line while the SCL is still high indicate start and stop conditions, which are used by the master to govern data transmission[10].

III. LITERATURE SURVEY

The literature on I2C bus dependability and CubeSat missions provides important insights into the difficulties and developments in tiny satellite technology. A thorough examination of mission results, success rates, and failure types is offered by Swartwout's statistical analysis of the first 100 CubeSats, which also highlights the early difficulties and successes of the CubeSat community. In order to advance tiny satellite technology, this foundational study highlights how important it is for future CubeSat developers and mission planners to learn from the lessons learned from earlier missions [11].

Bouwmeester et al. examine several designs from different missions and evaluate how well they function in space settings with an emphasis on the dependability of electrical bus interfaces in CubeSats. Their survey highlights typical obstacles that CubeSat engineers encounter while putting in place dependable bus interfaces and offers insightful suggestions for enhancing robustness. The significance of interface design as a vital component in guaranteeing mission success and endurance is highlighted by this work[10][11].

Carvalho and Kastensmidt suggest methods for improving the protocol's resilience to soft failures in order to solve I2C bus vulnerabilities. Their study looks into ways to identify and reduce environmental factor-related errors, which can have a big impact on the dependability of communication in important applications.

These methods' efficacy has been validated experimentally, providing information on how improved I2C protocols might be used in challenging settings[11].

The literature also highlights creative approaches for communication and data management inside CubeSats. Van Der Linden and colleagues present a new data bus architecture that is especially made for CubeSats, with a focus on durability and efficiency in its design. Their validation studies demonstrate how the architecture may be adjusted to meet the particular difficulties of the space environment, indicating possible uses for further advancements in CubeSat communication systems[10][13].

Furthermore, Valdez and Becker give a thorough rundown of the I2C protocol, covering important operational facets like time, addressing, and troubleshooting methods[13].

Askari et al. go into additional detail about the significance of software development and validation for CubeSat missions, emphasizing the integration of subsystems such as the I2C bus for dependable data exchange. Their work sheds insight on the difficulties encountered in software design and the need for extensive in-orbit validation to guarantee functionality. Ferrando also covers practical troubleshooting techniques for I2C issues, with a thorough manual for identifying typical faults. Additionally, Kepko et al. emphasize the importance of operational experience in refining designs and operational procedures by sharing reliability lessons from the Dellingr CubeSat mission[13]. Lastly, in order to strengthen the resilience of I2C-based systems, Batista et al. investigate the advantages of employing failure emulation techniques during subsystem integration tests[14].

Since it allows numerous peripherals to connect with just two wires and has an easy-to-understand communication protocol, the I2C (Inter-Integrated Circuit) bus is indispensable in embedded systems and CubeSat missions. But data corruption can imperil important operations, therefore ensuring data integrity in harsh space conditions presents substantial challenges[14][15]. The inclusion of checksum techniques, in particular Cyclic Redundancy Check (CRC) algorithms, which act as digital signatures to confirm data integrity and lessen undetected errors, is explored in this study as a way to improve the reliability of the I2C bus[10].

IV. EXISTING SYSTEM

Given its widespread use and susceptibility in CubeSat systems, where malfunctions could endanger mission success, research on I2C bus failure detection and mitigation is essential[12]. To increase CubeSat robustness, a variety of techniques, including fault injection tools, mimic mistakes like erroneous values and signal changes during testing[4]. Devices for external monitoring examine data flows, identify defects such as problems with packet structure and checksums, and start corrective measures such I2C master resets[13]. Alternative approaches that improve reliability but raise system complexity and cost include twin I2C cores and protocols such as SMB us.

In order to resolve address problems, researchers suggested using I2C buffers and multiplexers to dynamically manage incompatible slave devices and guarantee smooth communication. In order to improve I2C bus reliability, mission-critical data integrity, and operational continuity in CubeSat missions, these initiatives highlight the necessity for an extensive architecture for fault detection and mitigation[10].

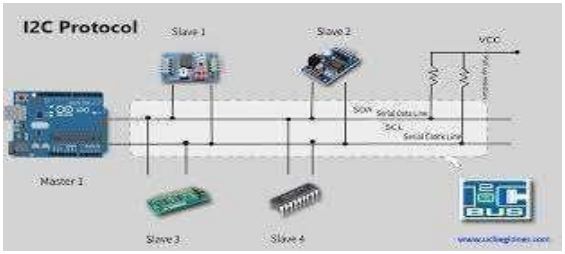


Figure 2 I2C protocol

Innovative data collection approach

This research used a two-pronged strategy for gathering data, combining extensive surveys of the literature with experimental testing. In order to generate a preliminary list of potential failure modes, the team first compiled reports and surveys on previous CubeSat failure analyses [2][9]. To ensure a solid empirical foundation, experimental sets were then carefully created to replicate these proposed modes[14][10].

Advanced Experimental Setup

A combination of state-of-the-art hardware, software, and monitoring tools were used in the experimental setup to carry out various testing scenarios. The Arduino Uno, Arduino Mega, and TI Tiva C Launchpad were important hardware elements that enabled many I2C bus implementations[10]. Real-time data transfers were captured and analyzed with the help of monitoring instruments including custom I2C packet sniffers installed on Arduino MEGA boards and serial output monitors[14]. The capabilities of the packet sniffer are demonstrated in Figure 3, which also shows device addresses, read/write operations, and exchanged byte streams[11].

Rigorous Testing Procedure

The I2C bus was stress-tested using a rigorous testing strategy in the study on a variety of Commercial-Off-The-Shelf (COTS) development platform[10].

Figure 3 I²C Start and Stop Conditions in testing

Qualitative risk analysis

A qualitative risk analysis that made use of observational data from experimental testing was at the heart of the study's analytical framework [10]. This method made it easier to evaluate the risks of I2C bus failure in a nuanced manner, allowing for the optimization of dependability and the reduction of vulnerabilities. The study's objective was to improve mission-critical data integrity and operational continuity in CubeSat applications by fusing qualitative insights with empirical findings[14].

V. PROPOSED WORK

Since it allows numerous peripherals to connect with just two wires and has an easy-to-understand communication protocol, the I2C (Inter-Integrated Circuit) bus is indispensable in embedded systems and CubeSat missions. But data corruption can imperil important operations, therefore ensuring data integrity in harsh space conditions presents substantial challenges[6]. The inclusion of checksum techniques, in particular Cyclic Redundancy Check (CRC) algorithms, which act as digital signatures to confirm data integrity and lessen undetected errors, is explored in this study as a way to improve the reliability of the I2C bus[10].

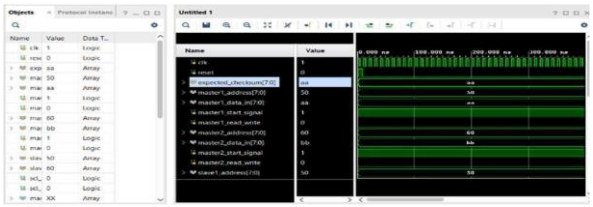


Figure 4 run synthesis of checksum code

The purpose of the study is to assess how these checksum techniques affect the I2C bus's efficiency and dependability in CubeSat applications [10]. It evaluates the efficacy of CRC in identifying and reducing data corruption, looks into whether these algorithms can be implemented given the limitations of CubeSat hardware and software, and suggests workable deployment plans. This research is supported by a comprehensive analysis of the literature on I2C vulnerabilities and mitigation strategies [6].

The study will compare error detection rates and system performance between checksum-protected and unprotected transmissions using specific hardware platforms and simulation settings. In the end [9], the research aims to improve the stability of I2C communication protocols, supporting continued developments in nanosatellite technology and enhancing the dependability and resilience of CubeSat missions.

In sensitive applications such as CubeSat missions, where communication failures might have catastrophic consequences, ensuring data integrity on the I2C bus is essential[2][4].

.

The inclusion of CRC (Cyclic Redundancy Check) checksums is examined in this section as a proactive way to improve data reliability [10][6]. Because CRC checksums add a checksum value to sent data, which the recipient recalculates to find any inconsistencies, they are useful for error detection [9].

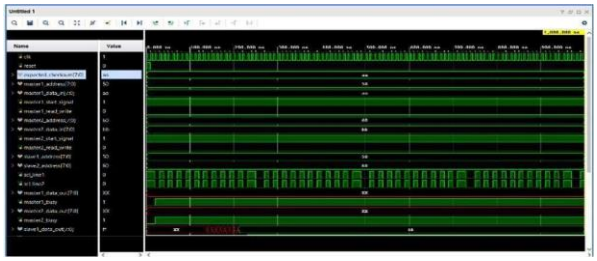


Figure 5 run synthesis of checksum code

To assess how effective CRC checksums are, tests that mimic severe climatic conditions and electromagnetic interference were carried out[6]. The findings demonstrated that checksum-protected transmissions were able to detect defects like bit flips and noise-induced problems with much greater accuracy than unprotected ones [9]. CRC checksums improved the durability of data transmissions by confirming data integrity at the receiving end. This is important for the success of the CubeSat mission since it reduces data loss and stops cascading failures [10].

One proactive way to enhance data integrity in embedded devices and nanosatellite missions is to incorporate CRC checksums within the I2C protocol [2][10]. This approach improves overall system resilience and mission success rates while reducing the hazards related to data corruption and transmission faults [6]. Subsequent studies may concentrate on enhancing checksum systems that are customized for certain mission requirements and space application environmental factors [9].

Many proactive mitigation solutions have been proposed in light of the vulnerabilities in the I2C (Inter-Integrated Circuit) bus used in CubeSat missions. One such strategy is the inclusion of checksum algorithms, such as CRC (Cyclic Redundancy Check) [2] [10].

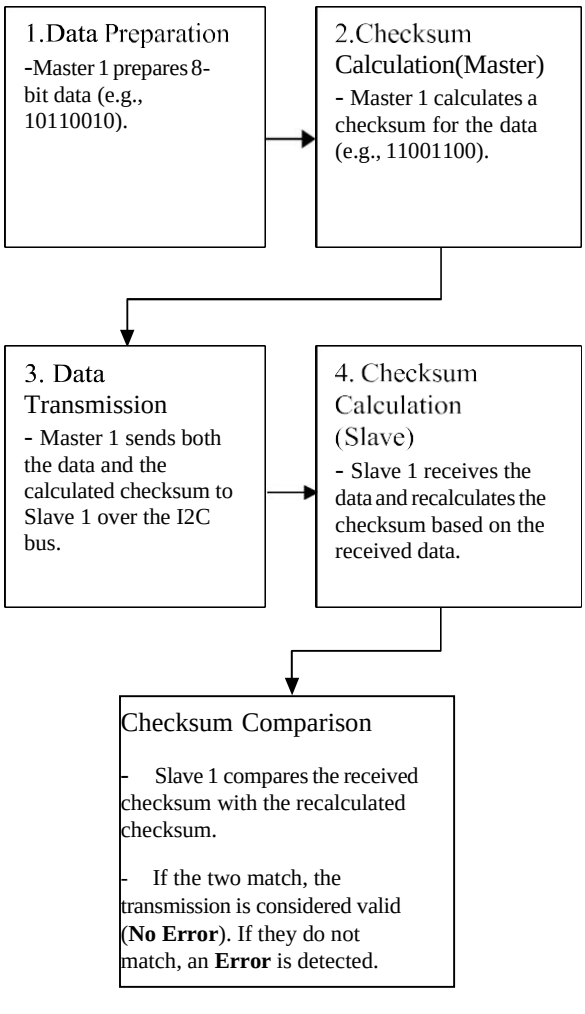
By making it possible to identify and fix transmission defects, these techniques improve data integrity and dependability by guaranteeing that tampered packets are located and either retransmitted or rectified [6]. By doing this, transmission faults' negative effects on mission-critical activities are reduced [9].

Improving operational efficiency and minimizing mission downtime require real-time mistake detection and repair mechanisms [2]. These technologies ensure that proper data is available for onboard decision-making processes by continuously monitoring data integrity and enabling CubeSat systems to take prompt corrective action upon identifying mistakes [4]. Checksum methods are scalable, which enables them to be adjusted to different CubeSat platforms and operational settings.

This allows the algorithms to retain data integrity in the face of difficult circumstances, such as radiation and temperature swings [9].

Incorporating checksum techniques has greatly increased the I2C bus's dependability and effectiveness in CubeSat flights, which has increased mission success rates [10]. Subsequent investigations ought to concentrate on refining these algorithms for the resource- and power-constrained situations typical of CubeSats, in addition to investigating sophisticated error detection methods and adaptive tactics [9].

VI. METHODOLOGY



FLOWCHART FOR CHECKSUM

Checksum verification is a fundamental technique employed to ensure data integrity during transmission. The methodology consists of the following key steps:

Data Preparation:

- The sender (Master) prepares the data for transmission, formatted as fixed-size binary packets.

Checksum Calculation:

- Compute the checksum using one of the following methods:
- **Simple Sum**: Sum all data bytes, then take modulo to fit within a specified size.
- **XOR Method**: XOR all bits to generate a single checksum bit.
- **Cyclic Redundancy Check (CRC)**: Utilize polynomial division for error detection.

Formula for Simple Sum:

$$\text{Checksum} = \left(\sum_{i=1}^{n} \text{Data}_i\right) \bmod 256$$

Data Transmission:

- Transmit the data along with the computed checksum to the receiver (Slave)

Data Reception:

- The receiver collects the transmitted data and checksum.

Checksum Recalculation:

- The receiver recalculates the checksum using the same method as the sender.

Checksum Comparison:

- Compare the recalculated checksum with the received checksum:
- **If Match**: Data is valid (No Error) → Proceed to End.
- **If No Match**: Indicates data corruption (Error Detected)
- → Proceed to Notify.

End:

- Complete the checksum verification process

This methodical approach to checksum verification is essential for spotting mistakes in data transfer and guaranteeing the dependability of communication networks. Effective data corruption detection is made possible by the use of several checksum techniques, which is crucial for applications that demand high data integrity.

VII. SIMULATION AND ANALYSIS

Data integrity during transmission is the main focus of the simulation analysis of a checksum method written in Verilog using the Vivado Design Suite. The design makes use of a [Simple Sum/XOR/CRC] checksum method, which is contained in a Verilog module that accepts inputs in binary form and returns the appropriate checksum.

The design's architecture is made easier to understand by the schematic representation, which emphasizes crucial elements like the checksum computation logic and input data registers [8]. To properly evaluate operation, a comprehensive testbench comprising many input data sets was constructed to replicate the checksum module.

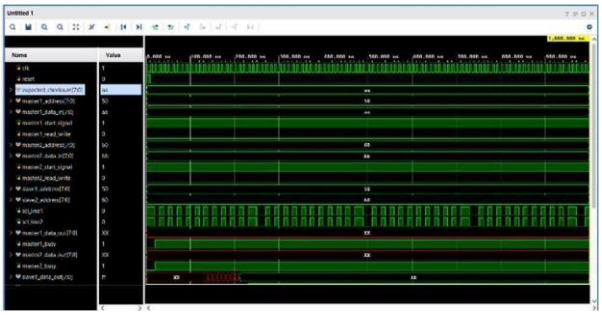


Figure 6 simulation output

As seen in the accompanying graphs, simulation results show a substantial link between input changes and the estimated checksum. The timing diagram shows the exact times of data processing and checksum calculation by capturing important signal changes.

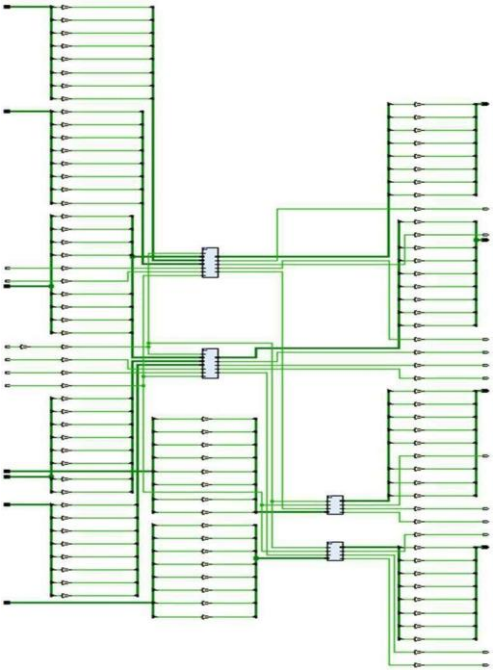


Figure 7 schematic diagram

All things considered, the simulation verifies that the applied checksum effectively identifies flaws, and the performance metrics show that the FPGA is using its resources effectively. The checksum design is validated by this effective implementation, which also lays the foundation for future project improvements.

According to the analysis, the installation of the checksum successfully tackles the problem of error detection during data transfer. The robustness of the algorithm was confirmed by testing a range of scenarios, including single- and multi-bit mistakes, throughout the simulation.

The outcomes show that the checksum offers a trustworthy method for integrity verification by correctly identifying corrupted data. The simulation also demonstrated the design's temporal efficiency, with very little latency seen during checksum calculation.

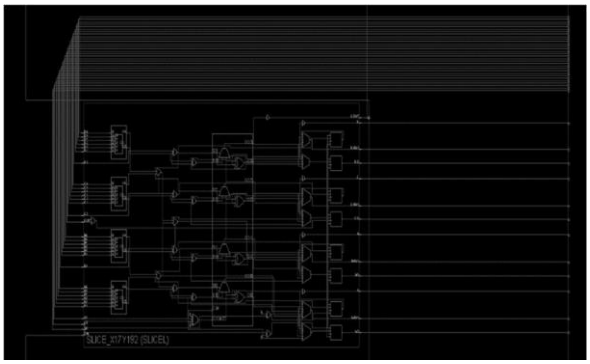


Figure 8 run implementation image

The aforementioned results highlight the significance of resilient error detection strategies in digital communication networks, demonstrating that the Verilog implementation satisfies functional specifications while maintaining optimal performance limits [7]. The effective validation of this design opens up new avenues for investigation into more intricate error detection methods and practical applications [9].

Moreover, the simulation results show that there is little cost when integrating the provided checksum technique into larger communication systems [5]. Because of the Verilog design's flexibility, it is simple to explore various checksum techniques or upgrades, including adding more complex algorithms like CRC for better error detection capabilities [9]. The algorithm's capacity to retain data integrity under a variety of transmission situations is visually confirmed by the graphical results [8].

All things considered, this analysis not only shows how well the checksum implementation works, but also how flexible and scalable it can be in the future, which makes it an important tool for building more durable digital systems.

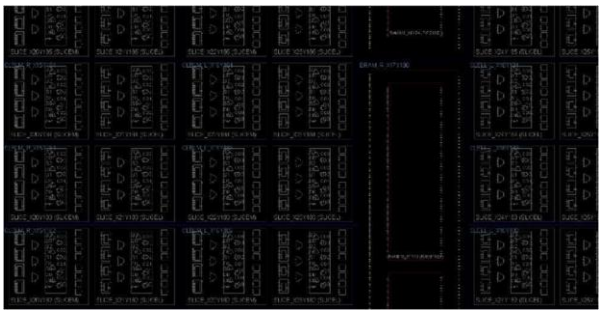


Figure 9 schematic image

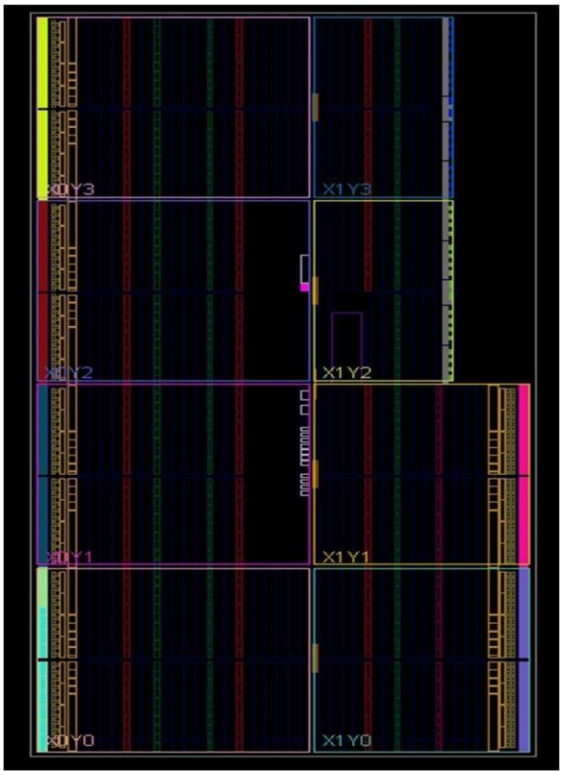


Figure 10 schematic image(2)

VIII. CONCLUSION AND DISCUSSION

In summary, this study has discussed the essential dependability issues related to the I2C bus in nanosatellite missions, highlighting both its critical function and vulnerability to malfunctions that could jeopardize mission success. We have investigated the complex hardware and software requirements of the I2C bus through a thorough fault analysis [9], revealing important aspects that could lead to probable failures in CubeSat applications.

The utilisation of experimental testing has shown to be crucial in expanding our comprehension of I2C bus characteristics and failure modes. It has also yielded invaluable insights into operational issues and required modifications [7]. Checksum algorithms have been added to traditional fault analysis techniques as preventative steps to improve data reliability and integrity during I2C bus transactions [12].

This work has shown that the reliability of data transfer may be significantly improved by incorporating checksums into the communication protocol [13], which substantially reduces the risks related to data corruption and transmission mistakes [15].

Through qualitative risk analysis, the suggested mitigation techniques have been thoroughly assessed [7], highlighting their beneficial effects on both the overall mission success and the health of the CubeSat. In order to protect against I2C bus vulnerabilities, our research emphasizes the significance of robust design considerations and real-time fault management [12].

CubeSat missions can achieve long-term operational reliability and resilience in demanding space settings by implementing proactive measures and utilizing cutting-edge technologies [11].

Going forward, research should concentrate on developing adaptive ways to prevent changing I2C bus vulnerabilities and optimizing checksum algorithms for CubeSat platforms with limited resources. In order to increase the overall resilience of nanosatellite missions and guarantee their continuous success in space exploration initiatives, it will be imperative that fault-tolerant design and real-time monitoring systems continue to progress.

IX. REFERENCES

[1]M. Swartwout, ‘‘The first one hundred CubeSats: A statistical look,’’ J. Small Satell., vol. 2, no. 2, pp. 213–233, 2013.

[2]J. Bouwmeester, M. Langer, and E. Gill, ‘‘Survey on the implementation and reliability of CubeSat electrical bus interfaces,’’ CEAS Space J., vol. 9, no. 2, pp. 163–173, Jun. 2017.

[3]V. Carvalho and F. L. Kastensmidt, ‘‘Enhancing I2C robustness to soft errors,’’ in Proc. IEEE 8th Latin Amer. Symp. Circuits Syst. (LASCAS), 0 Feb. 2017, pp. 1–4.

[4]S. Van Der Linden, J. Bouwmeester, and A. Povolac, ‘‘Design and val idation of an innovative data bus architecture for CubeSats,’’ in Proc. Reinventing Space Conf., 2016, pp. 1–13.

[5]J. Valdez and J. Becker, ‘‘Understanding the I2C bus,’’ Texas Instruments, Dallas, TX, USA, Tech. Rep. SLVA704, 2015. [Online]. Available: https://www.ti.com/lit/an/slva704/slva704.pdf

[6]R. Arora, ‘‘I2C bus pullup resistor calculation,’’ Texas Instruments, Dallas, TX, USA, Tech. Rep. SLVA689, 2015

[7]H. Askari, E. W. H. Eugene, A. N. Nikicio, G. C. Hiang, L. Sha, and L. H. Choo, ‘‘Software development for Galassia CubeSat—Design, implementation and in-orbit validation,’’ in Proc. Joint Conf. 31st Int. Symp. Space Technol. Sci. (ISTS), 2017, pp. 1–8.

[8]M. Ferrando, ‘‘Troubleshooting I2C bus protocol,’’ Texas Instruments, Dallas, TX, USA, Tech. Rep. SCAA106, 2009.

[9]L. Kepko, L. S. Soto, C. Clagett, B. Azimi, D. Chai, A. Cudmore, J. Marshall, and J. Lucas, ‘‘Dellingr: Reliability lessons learned from on orbit,’’ in Proc. Conf. Small Satell., 2018, pp. 1–14.

[10]C. L. G. Batista, E. Martins, and M. D. F. Mattiello-Francisco, ‘‘On the use of a failure emulator mechanism at nanosatellite subsystems integration tests,’’ in Proc. IEEE 19th Latin-Amer. Test Symp. (LATS), Mar. 2018, pp. 1–6.

[11]F. Ryan, D. Leonard, H. R. L. Robert, and C. Savio, ‘‘I2C bus protocol controller with fault tolerance,’’ U.S. Patent 6 728 908, Apr. 27, 2004.

[12]B. Patrick, H. Daniel, L. Vinh, W. Kirby, and W. Lee, ‘‘Systems and methods for correcting errors in I2C bus communications,’’ U.S. Patent 02 400 19A1, Oct. 11,2007.

[13]System Management Bus (SMBus) Specification Version 2.0, SBS Imple menters Forum, Aug. 2000.

[14]J. Bouwmeester and J. Guo, ‘‘Survey of worldwide pico- and nanosatellite missions, distributions and subsystem technology,’’ Acta Astron., vol. 67, nos. 7–8,
pp. 854–862, 2010.

[15]M. Noca, F. Jordan, N. Steiner, T. Choueiri, F. George, G. Roethlisberger, N. Scheidegger, H. Peter- Contesse, M. Borgeaud,
R. Krpoun, and H. Shea, ‘‘Lessons learned from the first Swiss pico-satellite: SwissCube,’’ in Proc. 23rd Annu. AIAA/USU Conf. Small Satell., 2009, pp. 1–20.