

# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnanasangama, Macche, Santibastwada Road Belagavi-590018,  
Karnataka



A  
UG PROJECT REPORT  
on

## Design and Verification of Flight Data Acquisition System using UVM

*Submitted in partial fulfillment of the requirement for the degree of*

**Bachelor of Engineering**

*in*

**Electronics & Communications Engineering - ECE**

*by*

USN : 1DS19EC084    Name: Nainshree Raj  
USN : 1DS19EC096    Name: Pallavi G  
USN : 1DS19EC099    Name: Poornima M  
USN : 1DS20EC417    Name: Lakshmi S

Under the guidance of

**Dr. Jamuna S**

Internal Project Guide

Professor, ECE Dept., DSCE,

Bengaluru-78



**Department of Electronics & Communication Engineering**

(An Autonomous College affiliated to VTU Belgaum, accredited by NBA & NAAC, Ranked by NIRF)

Shavige Malleshwara Hills, Kumaraswamy Layout,

Bengaluru-560078, Karnataka, India

**2022-23**

# Certificate

Certified that the project work entitled "Design and Verification of Flight Data Acquisition System using UVM" carried out by Nainshree Raj (1DS19EC084), Pallavi G (1DS19EC096), Poornima M (1DS19EC099), Lakshmi S (1DS20EC417) are bonafide students of the ECE Dept. of Dayananda Sagar College of Engineering, Bangalore, Karnataka, India in partial fulfillment for the award of Bachelor of Engineering in Electronics & Communication Engineering of the Visvesvaraya Technological University, Belagavi, Karnataka during the academic year 2022-23. It is certified that all corrections / suggestions indicated for project work have been incorporated in the report deposited to the ECE department, the college central library & to the university. This final year project report (**Course Code : 19EC8ICPR2**) Phase-II has been approved as it satisfies the academic requirement in respect of project work prescribed for the said degree.

---

Dept. Project Coordinators (Section incharges)  
Abhishek - Suma / Manasa / Srividya / Bindu

---

Project Guide  
Dr. Jamuna S

---

Head of the Department  
Dr. T.C. Manjunath, Ph.D. (IIT Bombay)

---

Dr. B.G. Prasad  
Principal, DSCE

## External Project Viva-Voce

Name of the project examiners (int & ext) with date :

1 : Signature : \_\_\_\_\_

2 : Signature : \_\_\_\_\_

## Declaration

Certified that the project work entitled, "**Design and Verification of Flight Data Acquisition System using UVM**" with the project work course code **19EC8ICPR2** is a bonafide work that was carried out by ourselves in partial fulfillment for the award of degree of Bachelor of Engineering in Electronics & Communication Engg. of the Visvesvaraya Technological University, Belagavi, Karnataka during the academic year 2022-23. We, the students of the project group/batch no. R-06 do hereby declare that the entire project work has been done on our own & we have not copied or duplicated any other's work or may be the extension of the works done by the earlier students. The results embedded in this UG project report has not been submitted elsewhere for the award of any type of undergraduate degree.

Student Name-1 : Nainshree Raj.

USN : 1DS19EC084

Sign : \_\_\_\_\_

Student Name-2 : Pallavi G

USN : 1DS19EC096

Sign : \_\_\_\_\_

Student Name-3 : Poornima M.

USN : 1DS19EC099

Sign : \_\_\_\_\_

Student Name-4 : Lakshmi S

USN : 1DS20EC417

Sign : \_\_\_\_\_

Date : 20/ 06 /2023

Place : Bengaluru - 78

## **Acknowledgement**

We would like to extend our thanks to Dr.T.C .Manjunath, HOD of ECE department. We would like to extend our sincere thanks to our Guide, Dr. Jamuna S, who encouraged and motivated us in developing this project and constant guidance throughout the development of this project. We take immense pleasure in expressing our sincere gratitude to our guide who despite their busy schedule made time for providing their valuable feedback on the project. We would like to extend our sincere gratitude for our project Co-ordinators- Dr.Abhishek , Dr. Suma , Prof.Manasa, Prof. Srividya, Prof.Bindu for their support. We would like to thank the academic and technical staff of the ECE Department for their support by providing the required resources. This would be incomplete without acknowledging the love and support of our parents and Friends who always stood by us at every point of life.

# Table of Contents

Title Sheet	i
Certificate	ii
Declaration	iii
Acknowledgement	iv
Table of Contents	v
List of Figures	vi
Nomenclature and Acronyms	vii
Abstract	1
Chapter 1 Introduction	2
Overview of the project work	
Background information about the project work	
Motivation obtained to take up the project work	
Problem statement of the project work	
Objectives of the project work	
Scope of the project work	
Organization of the project report	
Chapter 2 Literature Survey	8
Chapter 3 Project Details	9
Chapter 4 Simulation or Experimental Results & Discussions	20
Chapter 5 Conclusions, Future Work & Outcome of the project work	25
References	28
Appendix	30
Papers presented	49
Awards, Certificates Recognitions & Photographs	50
Plagiarism reports	52
CO-PO Mapping Justification Sheets	56
Budget Estimation Sheets	57

# List of Figures

Fig. 1 : Block-diagram of the proposed methodology	9
Fig. 2 : Flow-chart of the methodology used	10
Fig. 3 : Data Flow Diagram (DFD)	11
Fig.4: De1-Soc FPGA Board	12
Fig 5: Flight Data Acquisition System	15
Fig 6 : UVM Architecture	17
Fig 7: SPI Interface	20
Fig 8 : FIFO	20
fig 9:Verification using UVM	21
Fig 10: Memory block	21
Fig11: Memory	21
Fig 12: Log report when the test is passed	22
Fig 13: Log file output when the test is failed	22

# Nomenclature and Acronyms

## Abbreviations (Alphabetical Order) :

IEEE	Institute of Electrical & Electronics Engineers
DSCE	Dayananda Sagar College of Engineering
ECE	Electronics & Communication Engineering
UVM	Universal Verification Methodology
FDAS	Flight data acquisition System
EDA	Electronic Design Automation
HDL	Hardware Description Language
FPGA	Field Programmable Gate Array
FDR	Flight Data Recorder
ADC	Analog to Digital Convertor
DUT	Design Under Test
QAR	Quick Access Recorder
SPI	Serial Peripheral Interface
CVR	Cockpit Voice Recorder
UAV	Unmanned Aerial Vehicles





## Abstract

Avionics is an aviation industry platform that provides comprehensive solutions for flight planning, scheduling, and management. Flight data acquisition(FDA) in Avionics refers to the process of collecting and gathering flight-related information from various sources within the Avionics system. It involves the integration of data from multiple sources, such as aircraft sensors, air traffic control systems, weather information providers, and airline databases. These diverse sources contribute different types of data, including flight parameters, environmental conditions, and operational details. Overall, flight data acquisition in Avionics plays a crucial role in ensuring the availability of accurate and reliable flight-related information. It enables efficient flight planning, improves operational decision-making, and enhances safety and efficiency in the aviation industry.

The design phase involves the specification and implementation of the flight data acquisition system. This system typically consists of multiple components such as data sensors, analog-to-digital converters, data buses, and interfaces. The UVM methodology facilitates the creation of a modular and reusable verification environment by utilizing object-oriented programming and constrained-random stimulus generation techniques. The verification phase focuses on ensuring that the flight data acquisition system meets the desired functionality, performance, and reliability requirements. UVM enables the creation of test benches that incorporate verification components such as drivers, monitors, and scoreboards. These components work together to generate stimulus, monitor the behavior of the design under test, and check for functional correctness.

**Keywords:** *Flight data acquisition (FDA), Avionics, FPGA, UVM(Universal Verification Methodology), EDA(Electronic Design Automation)*

# Chapter-1

## Introduction

The Flight Data Acquisition System (FDAS) is a crucial component in aircraft systems that collects, records, and transmits flight data for analysis, and monitoring. It plays a significant role in ensuring the safety, maintenance, and performance optimization of aircraft [1]. Universal Verification Methodology (UVM) is a standardized methodology used in the verification of digital hardware designs. It is based on the hardware description language (HDL) called SystemVerilog and provides a framework for efficient and reusable verification environments. When it comes to developing a Flight Data Acquisition System using UVM, the primary focus is on verifying the functionality and correctness of the system design. UVM provides a set of classes, methods, and macros that aid in building a verification environment and testbench for the FDAS

Flight data acquisition using Field-Programmable Gate Arrays (FPGAs) and the Universal Verification Methodology (UVM) is an advanced approach to collecting and verifying flight-related data in the aviation industry. FPGAs offer programmable hardware that can be customized to perform specific tasks efficiently, while UVM provides a standardized methodology for verifying digital designs. FPGAs can handle high-speed data processing and offer flexibility in terms of interfacing with various sensors, communication protocols, and data storage units. The UVM verification methodology is employed to ensure the correctness and reliability of the FPGA-based flight data acquisition system [4].

UVM is a standardized verification framework that provides a systematic and scalable approach to verifying digital designs. It involves the creation of reusable verification components, the development of test benches, and the application of constrained-random stimulus generation techniques to thoroughly test the functionality of the design. When applied to flight data acquisition using FPGAs, UVM allows for the creation of comprehensive test environments that simulate various flight scenarios and sensor inputs. The verification components and test benches are developed to mimic the behavior of the real-world flight data acquisition system and its associated interfaces. [5]

## 1.1 Overview of the project work

The first step is to design the Flight Data Acquisition System. This includes defining the functional blocks, data acquisition modules, interfaces, and processing units required to collect, process, and transmit flight data. The design is implemented using an FPGA programming language, such as VHDL or Verilog. The UVM Verification environment is used to develop the UVM verification environment specific to the FDAS FPGA design. This involves creating UVM components, such as the testbench, test sequences, monitors, drivers, and scoreboard, tailored to the FPGA implementation and testbench development is to build the testbench infrastructure using UVM components for the FDAS FPGA design. This includes creating modules to drive input stimuli to the FDAS design and monitor its outputs.

The testbench may also incorporate virtual interfaces to interact with the FPGA design. To test sequence generation and to develop UVM test sequences that stimulate the FDAS FPGA design with a variety of test scenarios, including different flight conditions, data rates, and error cases. These sequences verify the functionality and performance of the FDAS design implemented in the FPGA. The functional coverage models using UVM constructs to track the completeness of the verification process for the FDAS FPGA design. The coverage models ensure that all aspects of the FDAS design are adequately tested and help identify any untested or under-tested areas.

UVM assertion constructs and scoreboarding techniques to verify the correctness of the FDAS FPGA design. Assertions are used to check specific conditions or properties that should hold during operation, while scoreboarding compares the expected and observed behavior of the FDAS design. The UVM-based simulation for the FDAS FPGA design and analyze the results. The simulation allows for the detection of any functional or timing errors, as well as potential issues related to the FPGA implementation. Debugging features provided by UVM can assist in identifying and fixing issues encountered during the verification process. Evaluate the verification results, generate reports, and analyze the collected flight data. The flight data obtained during the simulation can be analyzed to ensure the FDAS design meets the required specifications and standards.

## 1.2 Background information about the project work

The flight data recorder (FDR) is an crucial component of the avionics black box. Its main job is to gather and store a variety of flight metrics and system information. The FDR, which has several sensors installed throughout the aircraft, keeps track of vital flight data like altitude, airspeed, heading, vertical acceleration, inputs for the controls, engine performance, and other factors. The FDR gives investigators a thorough picture of the aircraft's performance and behavior across various flight phases by collecting these specifics [2].

Define the Requirements: Establish the flight data recorder's specifications and requirements, including the number of data channels, memory capacity, and interface protocols.

To create the data recording module: Construct a SystemVerilog module that will act as the flight data recorder's brain. Components for data storage, buffering, and acquisition should be included in this module.

Implement Data Acquisition: Create and put into use the necessary signal conditioning circuits, such as analog-to-digital converters (ADCs), as well as the data acquisition circuitry

Storage and Buffering: Before writing the obtained data to permanent storage, create a buffer to store it there temporarily. Create the memory architecture to support the flight data recorder's data rate and storage needs.

Simulation and Verification: Creating thorough test benches will enable you to mimic and validate the flight data recorder design's functioning. To guarantee the proper functionality of the recorder, test various scenarios and cases using simulated input data.

Synthesize and Implement: Synthesis tool like Xilinx Vivado or Intel Quartus, synthesize the design into the target hardware platform once it has been validated. Prepare any necessary hardware platform interfaces or peripherals.

Hardware Validation: Test the flight data recorder on the intended hardware platform to carry out hardware validation. Check that the system is operating according to the set specifications and that the recorded data corresponds to the anticipated outcomes.

### 1.3 Motivation obtained to take up the project work

- Enhancing flight safety is one of the main goals of an FDAS, as it allows for proactive action to be taken to prevent accidents, enhance maintenance procedures, and ensure the general safety of aircraft operations by identifying potential safety issues, monitoring the performance of critical systems, and detecting anomalies or deviations from expected behavior.
- Flight data, including parameters like altitude, airspeed, heading, and control inputs, can help reconstruct the sequence of events leading up to the incident and provide insights into its causes. This information is essential for determining the root causes, implementing corrective measures, and improving aviation safety. In the unfortunate event of an accident or incident, an FDAS plays a crucial role in providing valuable data for investigation and analysis purposes.
- Flight data acquisition systems require expertise in various areas, including data acquisition, sensor integration, signal processing, data storage, and communication protocols. Tackling these technical challenges can be intellectually stimulating and rewarding, allowing us to expand our skills and knowledge in different domains.
- By tracking factors like fuel consumption, engine performance, flight efficiency, and system health, operators can identify areas for optimization. This can result in fuel savings, improved operational efficiency, lower maintenance costs, and improved overall performance of the aircraft. FDAS data enables continuous monitoring and analysis of aircraft performance parameters.
- The FDAS supports maintenance activities by providing real-time and historical data about the aircraft's systems and components, which can be used for predictive maintenance, detecting potential failures or abnormalities, and scheduling maintenance activities more effectively.
- Working on a flight data acquisition system can enhance our professional profile and open doors to various career opportunities. The aviation industry is vast, and our expertise in this area can make us a desirable candidate for jobs in aerospace engineering, aviation safety, data analysis, or research and development.

## 1.4 Problem statement of the project work

The proposed system has the advantages of a simple circuit structure and high integration, which is why the problem statement for the proposed project states that the flight data recorder (FDR), also known as the "Black Box" in aviation, is an electronic device used for recording information about the electro-mechanical systems of the aircraft.

The Flight Data Acquisition System aims to improve flight safety, increase operational effectiveness, optimize maintenance procedures, adhere to regulatory standards, and support data-driven decision-making in the aviation industry in order to meet these challenges and requirements.

## 1.5 Objectives of the project work

- The primary goal of the proposed project is to develop a flight data acquisition system.
- To achieve the parallel acquisition function of all types of flight data by utilizing the parallel operation mode of FPGA.
- To research and develop a data acquisition system to acquire and store the data and analyze the data.
- To verify the system using sophisticated verification methodology-UVM.

## 1.6 Scope of the project work

- The FDAS gathers information from various sensors and systems installed on the aircraft, such as altimeters, attitude and heading reference systems (AHRS), engine monitoring systems, fuel flow sensors, navigation systems, and many others, either in real-time or at predetermined intervals.
- The gathered data is recorded and stored in a dedicated onboard recording device, commonly known as the flight data recorder (FDR) or "black box," which allows investigators to review the recorded data in the event of incidents or accidents and preserves the data for a specific amount of time, typically mandated by regulatory requirements.

- In order to enable operators and maintenance teams to remotely monitor the aircraft's performance and make informed decisions about maintenance, fuel consumption, flight planning, and other operational aspects, the FDAS may also transmit selected flight data in real-time to ground-based systems.
- The flight data recorded by the system in the event of an incident or accident provides valuable insights into the flight parameters, actions of the flight crew, and behavior of various systems during the critical phases of flight. This data aids investigators in reconstructing the sequence of events leading to an incident or accident and facilitates the identification of contributing factors.

## **1.7 Organization of the project report**

The project work we undertook is organised in the order listed below. The introductory chapter of chapter 1 gave a overview of the study. In chapter 2, a thorough assessment of the pertinent works produced by researchers and other authors is provided, along with a list of their shortcomings. Chapter 3 provides a block diagram and explanation of the project work we undertook, as well as the hardware, software, descriptions, and interfaces we used. Chapter 4 presents the findings and a discussion of the project work we undertook, as well as the applications, benefits, and limitations. The report comes to a close in chapter 5 with the conclusion, next work, and scope.

Paper presentations in conferences during the tenure of the Project Work, Paper publications in journals during the tenure of the Project Work, Hard copy of the presented conference paper / published journal paper, Awards, Recognitions in Project Fests, Photographs & any other certificates, Plagiarism Report, CO PO Mapping Sheet, Budget Estimations are presented at the end of the project report one after the other in succession.

## Chapter 2

### Literature Survey

**[1]. Author by the name Ning Jia, Hang Chen and Jun Tian in the year 2021 proposed a model on “A Design of Configurable Multi-type Flight Data Acquisition System” [1].**

The objective of this paper is to compute the acquisition task of multiple types of flight data and store it in real time. The main draw back of this paper is complexity in the design framework and requires a lot of time and effort by professional technicians to operate.

**[2]. Author by the name G.V.Jayaramaiah and Chetan Umadi in the year 2020 have published paper on “FPGA implementation of multi-protocol data acquisition system using VHDL” [4].**

The objective of this paper is to implement a parallel and serial data transfer protocols and all protocols implanted on FPGA kit and modelled using VHDL. The digital signals are provided from multichannel sensors and different ADC protocols. This system uses one-wire bus with only one pin for communication which has limited applications.

**[3]. Author by the name Ufuk Sakarya and Ibrahim Hokelek, in the year 2020 have implemented model on “Universal Verification Methodology Application of ARINC429 for Airborne Electronic Hardware Certification” [5].**

The main objective of this paper is to verify the activity by selecting commonly used data bus as a digital subsystem design under verification. The draw back of this paper is this is just a demonstration of how we can set up the verification environment.

**[4]. Author by the name Harikrishnan.K, Vishwas.H.N, Vineetha Jain K.V, Dr.Ramesh Chinthala in the year 2020 published a paper on “Sensor data acquisition and denoising using FPGA”[6].**

The main objective of this paper is to design and implement an FPGA (Field Programmable Gate Array) based DAQ (Data Acquisition) system. The design shows how an entire system can be implemented in a single FPGA fabric thereby reducing development time and cost. This paper is limited to only one sensor.



## Chapter 3

### Project Details

#### 3.1 Block diagram of the proposed system

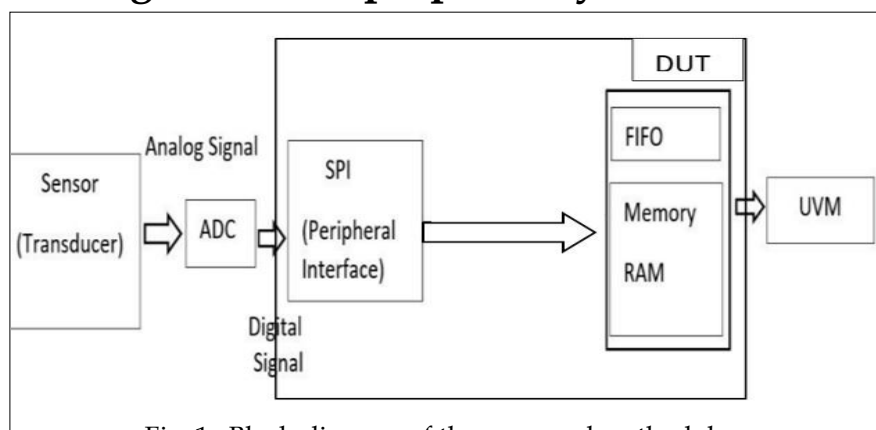


Fig. 1 : Block-diagram of the proposed methodology

Collecting data from external world such as engine sensor parameters, fuselage sensor parameters, altitude, atmosphere and other flight data and converting these signals into digital using Analog to digital converter. Interface signal module is used to collect and output various flight data, complete signal format and level conversion. FPGA send initials signal to ADC to start and waits for data. After conversion ADC sends an acknowledgement to the FPGA. FPGA reads all data and repeat the process The FPGA data acquisition module realizes the acquisition function of all kinds of flight data by using the parallel operation mode of FPGA. At last verifying the system using sophisticated verification methodology -UVM (Universal verification methodology).

UVM is open source that everyone can use the same environment for hardware design verification. The verification process is typically iterative, involving multiple rounds of simulation, formal verification, emulation, and testing until the hardware design is correct and ready for fabrication or development. The complexity of the hardware design and the level of verification required depends on the size, complexity, and criticality of the system being developed. UVM remains a widely used and supported methodology for hardware verification, providing a standardized and interoperable framework for developing reusable and scalable test benches.

### 3.2 Algorithm

In just the past several years there has been an increase in the research to evaluate and improve aircraft performance and flight characteristics. All of these efforts depend on the ability to acquire and utilize high fidelity data from a large range of sensors.

- Capturing data is a fundamental component of conducting flight research with unmanned aircraft and doing so requires the acquisition of high fidelity flight data from a large range of sensors and devices.
- The conversion of the electrical signals (Vout) from the sensor into a digital code is performed by an analog-to-digital converter (ADC) built in the FPGA.
- Acquired data are then processed by the microcontroller and the results are transmitted to a personal computer via the Universal Serial Peripheral Interface(SPI). The data from the sensor is stored in the memory.
- The Design Under Test(DUT) is verified using Universal Verification Methodology(UVM).

### 3.3 Flow-chart / Data Flow Diagram

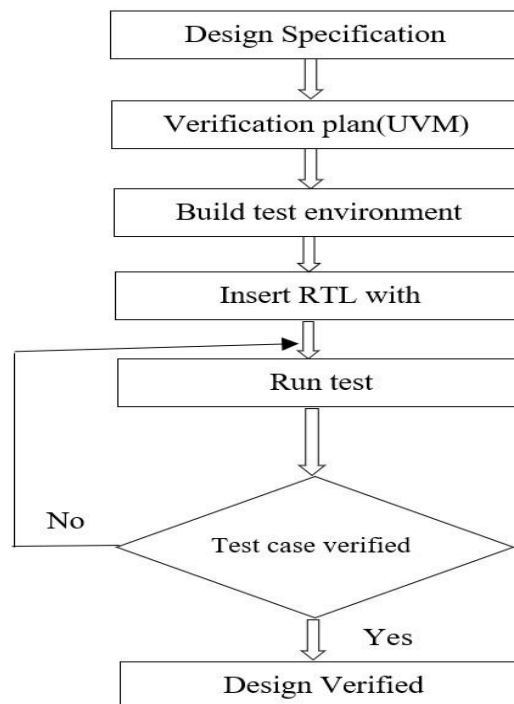


Fig. 2 : Flow-chart of the methodology used

### 3.4 Flow-chart/ Data Flow Diagram

The DFD of the working model is presented as shown in the Fig. 5.

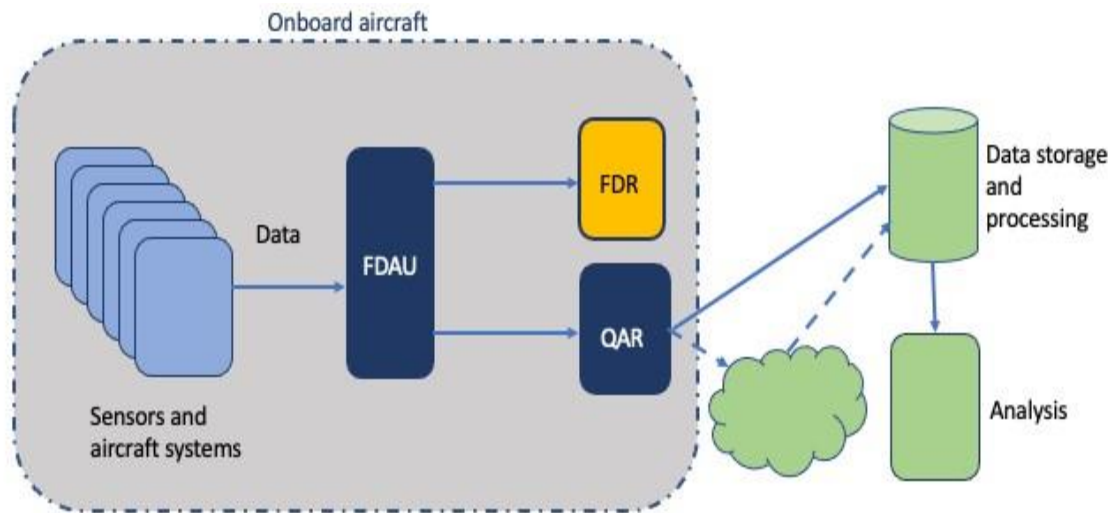


Fig. 3 : Data Flow Diagram (DFD)

Sensors are integral components of aircraft systems, playing a crucial role in monitoring various parameters and ensuring safe and efficient operation. They provide critical data that is used by the aircraft's systems for control, navigation, performance monitoring, and safety. Modern aircraft employ a vast array of sensors throughout their structure and subsystems to ensure accurate measurement, monitoring, and control of various parameters. The data collected by these sensors is utilized by onboard systems and avionics to maintain safety, optimize performance, and enhance the overall efficiency of the aircraft.

**Flight Data Source:** This represents the source of flight data, such as sensors, avionics systems, or external devices. The data from these sources is captured and sent to the Data Acquisition System.

**Data Acquisition System:** This module is responsible for acquiring the flight data from various sources. It may include data conversion, signal conditioning, and data buffering. The acquired data is then passed on to the Data Processing Module.

**Data Processing Module:** This module processes the acquired data, which may involve data filtering, calibration, synchronization, or any necessary preprocessing steps. The processed data is then forwarded to the Flight Data Storage System.

**Flight Data Storage System:** This system stores the processed flight data in a suitable format and storage medium, such as a database, file system, or cloud storage. The storage system ensures data integrity, reliability, and efficient retrieval.

**Data Analysis & Visualization Module:** This module handles the analysis and visualization of flight data. It may include functionalities like real-time monitoring, data mining, statistical analysis, and visualization tools to provide insights into the flight data for decision-making purposes.

### 3.6 Hardware used

In this section, the hardware descriptions related to the project work is presented in brief. The hardware that is used for the project work is Intel FPGA(DE1-Soc).

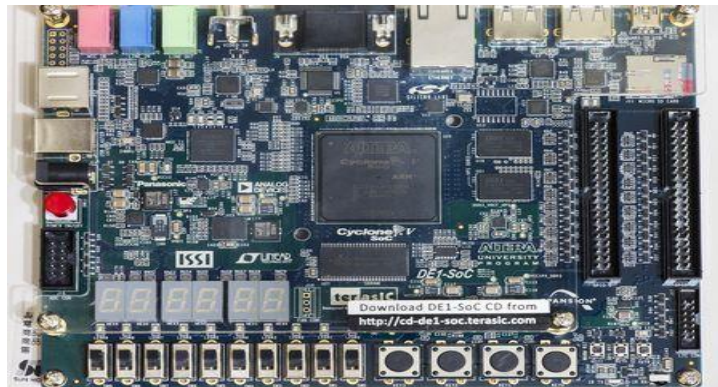


Fig.4: De1-Soc FPGA Board

The DE-1 SoC (System-on-Chip) is a development board created by Terasic that features an Altera Cyclone V FPGA (Field-Programmable Gate Array) along with an ARM Cortex-A9 dual-core processor. The DE-1 SoC combines the flexibility of an FPGA with the processing power of a microprocessor, making it suitable for a wide range of applications.

Here's a brief overview of the working of the DE-1 SoC FPGA:

- **FPGA Fabric:** The heart of the DE-1 SoC is the Cyclone V FPGA. The FPGA fabric consists of a large number of configurable logic blocks (CLBs), interconnects, and other functional elements. These components can be programmed to implement

custom digital logic circuits, including arithmetic units, memory interfaces, communication protocols, and more.

- **ARM Cortex-A9 Processor:** The DE-1 SoC also includes an ARM Cortex-A9 dual-core processor. This processor provides the capability to run a complete operating system, such as Linux, and execute high-level software applications. The Cortex-A9 processor can communicate with the FPGA fabric through various interfaces.
- **HPS (Hard Processor System):** The HPS is a combination of the ARM Cortex-A9 processor and a set of peripherals tightly integrated with the FPGA fabric. It includes memory controllers, interrupt controllers, timers, UARTs, GPIO (General-Purpose Input/Output) pins, and more. The HPS allows seamless communication between the processor and FPGA fabric.
- **Development Environment:** To program the DE-1 SoC, you can use Intel Quartus Prime, the development software provided by Intel (formerly Altera). It allows you to design and implement your custom digital logic circuits using hardware description languages like VHDL or Verilog. You can also create a software application to run on the ARM Cortex-A9 processor using standard programming languages such as C/C++.
- **Interfacing:** The DE-1 SoC board provides various physical interfaces for connecting external devices. These interfaces include USB ports, Ethernet ports, audio jacks, video interfaces, GPIO headers, and more. These interfaces allow the DE-1 SoC to interact with the outside world and connect to other devices.
- **Application Development:** With the DE-1 SoC, you can develop applications that leverage both the FPGA fabric and the ARM Cortex-A9 processor. You can create custom hardware accelerators in the FPGA fabric to offload computationally intensive tasks from the processor. The processor can handle higher-level tasks, manage the operating system, and communicate with external devices.

Overall, the DE-1 SoC provides a versatile platform for designing and implementing complex systems that require a combination of programmable logic and processing power. It enables rapid prototyping, system integration, and exploration of both hardware and software designs..

### 3.7 Software used

The software tool used for the project work is

QuestaSim - QuestaSim is part of the Questa Advanced Functional Verification Platform and is the latest tool in Mentor Graphics tool suite for Functional Verification.

QuestaSim is a powerful and widely used electronic design automation (EDA) software tool specifically designed for digital, analog, and mixed-signal simulation. Mentor, a Siemens Business, created and maintains it. Engineers may develop, validate, and debug intricate digital and mixed-signal architectures using QuestaSim's complete environment. It enables complete testing and verification of digital circuits and systems by supporting both event-driven and cycle-based simulation approaches. In addition to behavioural modelling, functional verification, performance analysis, and hardware/software co-simulation, the programme also provides a wide range of other simulation features. Waveform viewers, data visualisation, and interactive debugging interfaces are among the many debugging tools and capabilities it offers. Engineers may analyse simulation results, monitor signal behaviour, and find and correct design flaws with the help of these technologies.

QuartusPrime – To compile designs, perform timing analysis, simulate a design's reaction to different stimuli and to configure the target device with the programmer.

Field-programmable gate arrays (FPGAs) and complex programmable logic devices (CPLDs) may be designed, implemented, and programmed using Quartus Prime, a complete electronic design automation (EDA) software package created by Intel (previously Altera). For FPGA-based digital designs, it offers a full design flow from design entry to synthesis, place and route, and verification. Quartus Prime's potent synthesis and optimisation skills are one of its standout qualities. It may convert optimised gate-level representations appropriate for implementation on FPGAs from RTL (Register Transfer Level) specifications written in VHDL or Verilog. The tool uses cutting-edge algorithms to optimise the design in terms of functionality, use of space, and energy use. A powerful place and route engine is also a part of Quartus Prime, and it automatically maps the synthesised design to the destination FPGA device.

### 3.8 Working principle of the proposed system

#### A) Flight data acquisition design

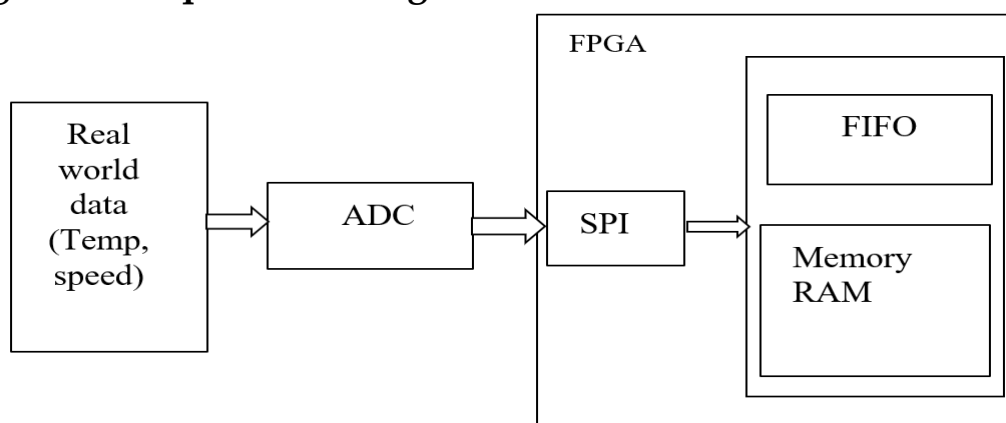


Fig 5: Flight Data Acquisition System

Flight data acquisition design aims to ensure accurate and reliable data capture, considering factors such as sensor accuracy, sampling rates, synchronization, data integrity, and system redundancy as shown in the fig1. The collected data is crucial for flight testing, aircraft certification, performance analysis, accident investigations, and ongoing aircraft health monitoring. The design of a flight data acquisition system involves selecting appropriate sensors to measure the desired parameters, designing the wiring and data buses to transmit the signals, determining the sampling rate and resolution of the data, and specifying the storage capacity and data retrieval methods. The system may utilize data recorders, data acquisition units, signal conditioning modules, and communication interfaces[1].

The system consists of ADC, interface signal module, FPGA data acquisition module, Memory storage. This system mainly comprises sensor, which collects the data from the external world. The configurable aircraft data acquisition system mainly collects engine sensor parameters, fuselage sensor parameters, atmosphere and other flight data. Analogue to Digital Converter or ADC, is a data converter which allows digital circuits to interface with the real world by converting an analogue signal into a binary code. To process the analog signal onto digital devices like as FPGA it should be first converted to digital format. After the signal conversion, data is handled using FPGA .

The Serial Peripheral Interface (SPI) is a full-duplex, synchronous protocol used as a serial data link that is standard across many microprocessors, microcontrollers. It enables

communication between microprocessors and peripherals. The SPI protocol is flexible enough to interface with numerous peripherals. Therefore, the main function of the FPGA data acquisition module is to collect many kinds of signals. FPGA, as the core of the flight data acquisition system, collects and stores the data. This system is divided into three steps: the signal processing module, FPGA data acquisition module, and data storage module.

The signal processing module in a flight data acquisition system is responsible for receiving, conditioning, and processing various sensor signals and data streams from different aircraft systems. It plays a crucial role in converting analog signals into digital format, applying necessary filters and transformations, and preparing the data for storage or transmission. The data acquisition module interfaces with a wide range of sensors and data sources throughout the aircraft. These may include airspeed sensors, altimeters, gyroscopes, accelerometers, engine parameters, control surfaces, avionics systems, and more. The signals from these sources are acquired and converted into digital form for further processing [6].

The FDR (Flight Data Recorder) which is commonly known as the "black box," is a specialized device installed on aircraft to record various parameters and flight information. It captures data such as altitude, airspeed, vertical acceleration, heading, control inputs, engine parameters, and more. The FDR typically uses solid-state memory or magnetic tape to store data, and it is designed to withstand extreme conditions, including crashes and fires, to ensure data survivability. Flight data acquisition systems are crucial for flight safety and performance monitoring. The collected data can be analyzed to identify trends, anomalies, and potential issues, enabling proactive maintenance, incident investigation [3].

## **B)Verification Plan**

Verification plan is done before verifying any project to ease the work of verification engineer. Based on the requirement, verification plan is to be built which includes list of test cases and coverage models. A verification plan defines what needs to be verified in a hardware design and then drives the verification strategy.



## Universal Verification Methodology

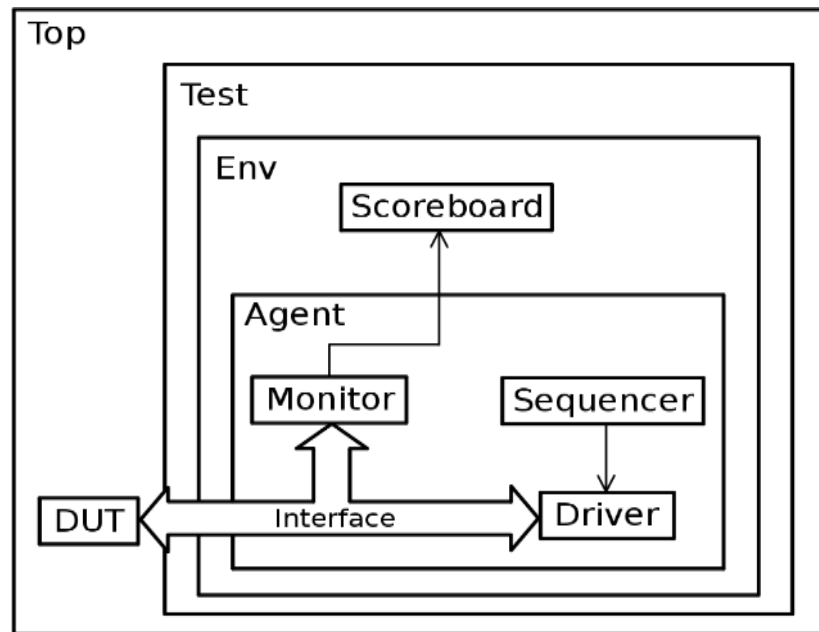


Fig 6 : UVM Architecture

Universal Verification Methodology (UVM) is a standardized methodology for functional verification of digital designs or systems. It provides a framework and set of guidelines for creating modular and reusable testbenches and verification components. UVM is widely adopted in the semiconductor industry and is supported by various electronic design automation (EDA) tools [5]. In the Universal Verification Methodology (UVM), there are several key components that work together to create a modular and reusable testbench environment. These components include:

Testbench Top: This is the main component of the testbench hierarchy. It coordinates the overall verification process and instantiates other testbench components.

Testbench Configuration: The testbench configuration specifies the desired configuration parameters for the testbench, such as clock frequency, interface configurations, or specific feature enablement.

DUT (Design Under Test) Agent: The DUT agent is responsible for interfacing with the design and translating the testbench transactions to the DUT's signals or interfaces. It contains the necessary drivers, monitors, and sequences for communication with the DUT.

Test: The test component defines a specific test scenario or use case that needs to be verified. It encapsulates the test sequence and may contain specific data and configuration information. The test component typically inherits from the `uvm_test` base class.

Environment: The environment component acts as the top-level container for the testbench. It provides the infrastructure for coordinating and controlling the verification process. The environment instantiates and connects other components, such as agents, monitors, drivers, and scoreboards. It usually inherits from the `uvm_env` base class.

Agent: An agent represents a specific interface or protocol within the design under test (DUT). It consists of multiple sub-components, each with a specific role:

Sequencer: The sequencer generates sequences of transactions or stimuli to be applied to the DUT. It controls the flow and timing of the transactions and manages sequence dependencies.

Driver: The driver receives the transactions from the sequencer and drives the stimuli onto the DUT's interface signals. It converts the transaction-level protocol into the signal-level protocol.

Monitor: The monitor component observes the DUT's interface signals, captures transaction-level information and converts it into transactions for analysis and checking in the testbench.

Scoreboard: The scoreboard compares the expected results, generated by the test or reference model, with the actual results obtained from the DUT. It checks for functional correctness and reports any discrepancies.

Sequences: Sequences represent a sequence of transactions or stimuli that are applied to the DUT. Sequences are typically generated by the sequencer and control the specific test scenario.

Sequence Items: Sequence items are the individual transactions or stimuli that make up a sequence. They encapsulate the data and control information to be applied to the DUT.

Configuration: The configuration component manages the configuration settings for the testbench and DUT. It provides a centralized location for storing and accessing configuration information.

Coverage: The coverage component tracks which parts of the design have been exercised by the testbench. It collects coverage data and generates coverage analysis reports, helping to ensure that the verification process is thorough.

Assertions: Assertions are used to define specific properties or conditions that the DUT must satisfy. They help in capturing and verifying design properties and can be used for both design and testbench verification.

These components work together to create a comprehensive UVM testbench environment for verifying digital designs. They provide a modular and scalable infrastructure that promotes reusability, maintainability, and efficient verification of the design under test.

### **Overall Conclusions of the Chapter 3**

Every FPGA will have an on-board ADC which converts analog data obtained from sensors to digital data. To send data between microcontrollers and small peripherals such as adc, shift registers, sensors we need an interface module.

We designed a Data Acquisition System in which random inputs are given and stored in memory. The design includes SPI protocol as Interface and FIFO to store data in the memory. The design act as DUT (Design under test) which is verified using UVM (Universal Verification Methodology). Each block of the UVM is designed ,which includes: UVM Interface, UVM Sequence, UVM Sequence Item, UVM Sequencer , UVM Driver , UVM Monitor , UVM Agent , UVM Scoreboard , UVM Environment UVM Test .

## Chapter-4

### Results and Discussions

Flight Data Acquisition System is designed in which random inputs are given and stored in memory. The design includes SPI protocol as Interface and FIFO to store data in the memory. The design act as DUT (Design under test) which is verified using UVM (Universal Verification Methodology). Each block of the UVM is designed, which includes: UVM Interface, Sequence, Sequence Item, Sequencer, Driver, Monitor, Agent, Scoreboard, Environment, Test. Here are the simulation results of flight data acquisition system which is verified using UVM.

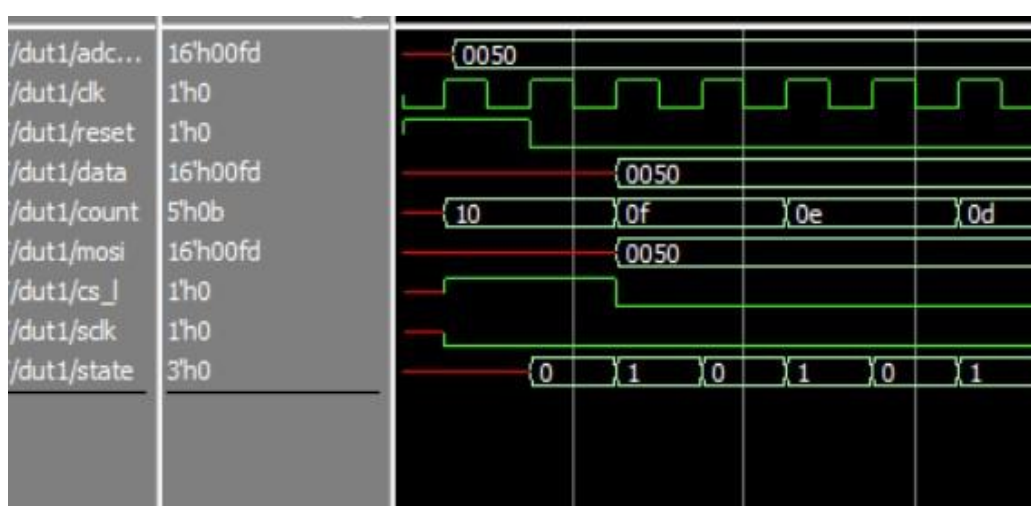


Fig 7: SPI Interface

The data from sensor is interfaced with FPGA using SPI interface and obtained results are shown in fig7.

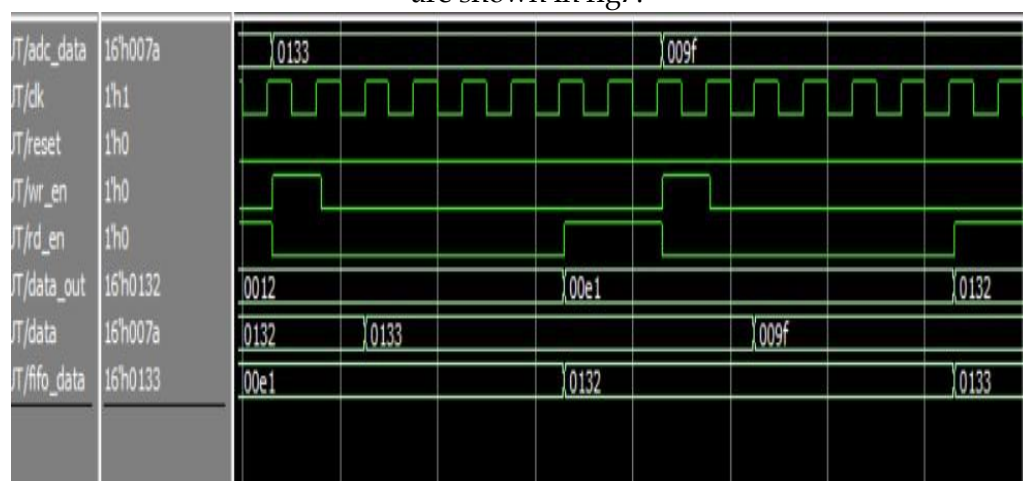


Fig 8 : FIFO

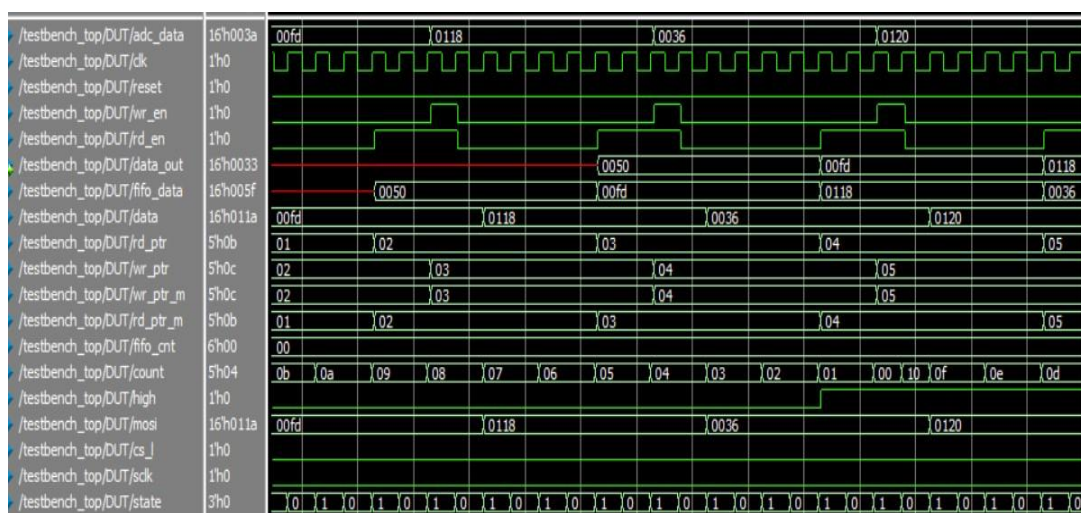


fig 9:Verification using UVM

Flight data acquisition system is verified using UVM testbench and observed waveform is as shown above in fig 9.

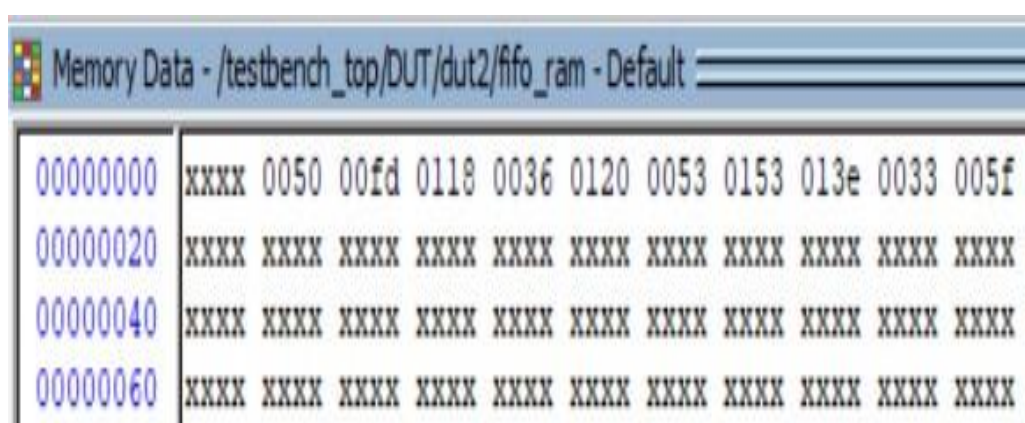


Fig 10: Memory Block

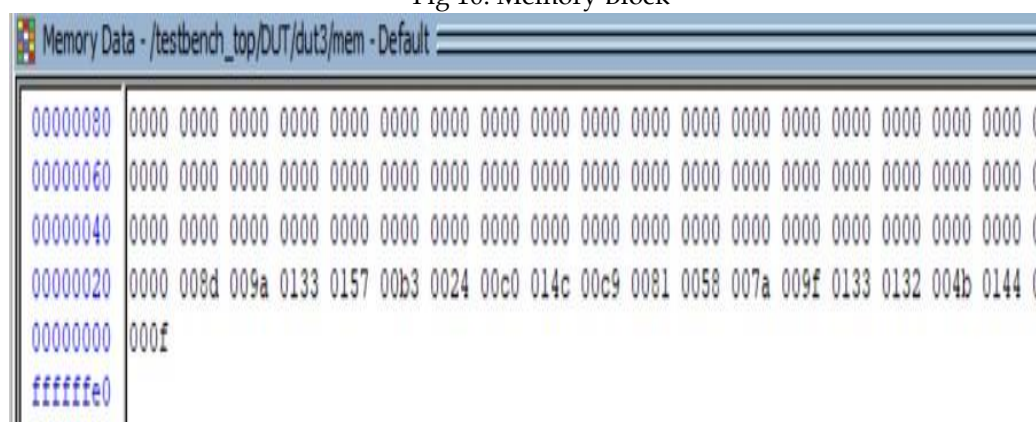


Fig11: Memory

The data processed using FPGA is stored in memory block , the above fig shows the data stored in memory during simulation.

```
# -----:RESULT:: -----
# Name      Type      Size Value
# -----:-----:
# data_seq_item data_seq_item - @736
# adc_data    integral  16  'h120
# -----:-----:
# UVM_INFO C:/questasim64_10.6c/examples/test.sv(377) @ 495: uvm_test_top.env.scb [SCOREBOARD] -----:RESULT:: -----
# UVM_INFO C:/questasim64_10.6c/examples/test.sv(378) @ 495: uvm_test_top.env.scb [ ] adc_data:120
# UVM_INFO C:/questasim64_10.6c/examples/test.sv(379) @ 495: uvm_test_top.env.scb [ ] data_out:36
# UVM_INFO C:/questasim64_10.6c/examples/test.sv(383) @ 495: uvm_test_top.env.scb [SCOREBOARD] TEST PASSED
run
# UVM_INFO @ 555: uvm_test_top.env.agnt.driver [DATA_DRIVER ] Got Transaction adc_data=13e
run
```

Fig12: Log report when the test is passed

When the test case is verified it prints as test passed in transcript window which is shown in fig 8.

```
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(215) @ 0: reporter [Questa UVM] QUESTA_UVM-1.2.3
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(217) @ 0: reporter [Questa UVM] questa_uvm::init(+struct)
# UVM_INFO @ 0: reporter [RNTST] Running test data_test...
# UVM_INFO @ 0: uvm_test_top.env.agnt [DATA_AGENT] connect_phase, Connected driver to sequencer
# UVM_INFO @ 0: uvm_test_top.env [data_ENVIRONMENT] connect_phase, Connected monitor to scoreboard
# Starting sequence spi_seq run_phase
# UVM_INFO @ 0: uvm_test_top.env.agnt.driver [DATA_DRIVER ] Got Transaction adc_data=50
# UVM_INFO @ 75: uvm_test_top.env.agnt.driver [DATA_DRIVER ] Got Transaction adc_data=fd
run
# UVM_INFO @ 155: uvm_test_top.env.agnt.driver [DATA_DRIVER ] Got Transaction adc_data=118
#
# Name      Type      Size Value
# -----:-----:
# data_seq_item data_seq_item - @683
# adc_data    integral  16  'h50
# -----:-----:
# UVM_INFO C:/questasim64_10.6c/examples/test.sv(377) @ 175: uvm_test_top.env.scb [SCOREBOARD] -----:RESULT:: -----
# UVM_INFO C:/questasim64_10.6c/examples/test.sv(378) @ 175: uvm_test_top.env.scb [ ] adc_data:50
# UVM_INFO C:/questasim64_10.6c/examples/test.sv(379) @ 175: uvm_test_top.env.scb [ ] data_out:0
# UVM_INFO C:/questasim64_10.6c/examples/test.sv(381) @ 175: uvm_test_top.env.scb [SCOREBOARD] TEST FAILED
run
# UVM_INFO @ 235: uvm_test_top.env.agnt.driver [DATA_DRIVER ] Got Transaction adc_data=36
run
# UVM_INFO @ 315: uvm_test_top.env.agnt.driver [DATA_DRIVER ] Got Transaction adc_data=120
#
# Name      Type      Size Value
```

Fig 13: Log file output when the test is failed

When the test case is not verified it prints as test failed in transcript window which is shown in fig 9.



## Applications, Advantages, Limitations

### Application/Advantages

- Modern data acquisition technology uses real-time communication and advanced networks to monitor aircraft all over the world. It is a process of converting realworld signals to the digital domain for display, storage, and analysis. They are applied in research, development, production, quality control, testing, and management.
- Flight data recorder (FDR) popularly known as “Black Box” in aviation is an electronic recording device in aircraft to facilitate the investigations of aviation accidents. This allows flight data to still be relayed to crews and inspectors on the ground in the event of an accident. The data recorded in the FDR are used to analyze the electronic systems of the aircraft for safety issues and investigation of aircraft accidents.
- Flight Safety and Accident Investigation: Improving flight safety and assisting accident investigations are two of FDAS's main uses. Analysing incidents and accidents requires access to data from an FDAS, including flight characteristics and audio recordings from the cockpit. It aids in the reconstruction of prior events, the identification of contributory components, and the discovery of the underlying causes of accidents. For enhancing aviation safety and putting preventative measures in place, this information is crucial.
- Maintenance and Health Monitoring: FDAS is crucial to maintaining and keeping track of the health of aircraft. FDAS enables maintenance staff to find abnormalities, spot possible problems, and do preventive maintenance by continually monitoring and recording flight characteristics and system health. Using the collected data for analysis makes it possible to forecast component failures, enhance maintenance plans.
- Evaluation and Optimization of Performance: FDAS data is useful for assessing and improving aircraft performance. To evaluate fuel consumption, engine performance, aerodynamic qualities, and other performance indicators, the

recorded flight data may be examined. With the use of this data, operators may see chances to save costs while also optimizing flight operations.

- **Flight Training and Simulation:** Applications utilizing FDAS data are utilized in flight training and simulation. For the purpose of pilot training, it is possible to recreate realistic flying scenarios using the recorded flight data and cockpit speech recordings. FDAS data may be used in flight simulators to precisely simulate aircraft behaviour, allowing pilots to practise different manoeuvres, emergency procedures, and unexpected circumstances in a secure setting.
- **Performance Monitoring and Trend Analysis:** At both the individual aircraft level and fleet level, FDAS data may be used for performance monitoring and trend analysis. Operators can monitor the performance of particular aircraft or whole fleets by tracking the recorded flight data over time. In order to improve the performance of the whole fleet, this analysis aids in finding deviations, tracking maintenance trends, assessing the success of operational improvements, and making data-driven choices.

## **Limitations**

- FDAS relies on sensors to gather flight data, and the sampling rate and resolution of these sensors may constrain the accuracy and granularity of the data collected. If the sampling rate is too low or the resolution is insufficient, some subtle or quick changes in flight parameters may not be accurately captured, which may affect the accuracy of analysis and system monitoring.
- FDAS typically includes a flight data recorder (FDR) that records and stores flight data for a predetermined period of time; however, the FDR's storage capacity is limited, and once the storage is full, older data may be overwritten; this limitation may present difficulties if historical flight data needs to be retrieved for analysis or investigation.
- FDAS must adhere to regulatory standards and guidelines, and ensuring that it does so can be challenging and time-consuming. Additionally, as regulations change or industry standards advance, the FDAS may need to be updated or modified on a regular basis.



## Chapter-5

# Conclusions, Future Work & Scope of Project

## Conclusions

The Flight Data Acquisition System (FDAS) plays a critical role in aviation safety and performance monitoring. It is an essential component of an aircraft's avionics system, responsible for collecting and recording various flight parameters and operational data during the entire duration of a flight. The FDAS captures information such as altitude, airspeed, engine parameters, flight control inputs, and numerous other variables that provide valuable insights into aircraft operations.

The FDAS is designed to meet stringent safety and reliability standards, ensuring accurate and real-time data acquisition under various flight conditions. It incorporates robust data storage capabilities, data retrieval mechanisms, and communication interfaces to facilitate data transfer and analysis. Advanced FDAS systems may also feature built-in diagnostic functions, self-monitoring capabilities, and automated alerts to promptly notify operators of any irregularities. The information collected by the FDAS is crucial for accident investigations, maintenance planning, and regulatory compliance. The FDAS includes a flight data recorder (FDR), sometimes referred to as the "black box" and a cockpit voice recorder (CVR). While the FDR records a wide variety of flight characteristics, the CVR records audio discussions and noises within the cockpit. In order for authorities to precisely reconstruct the events leading to an incident or accident, these recordings are essential for accident investigation, incident analysis, and pilot training.

An FDAS must have data retrieval and storing capabilities. To safely collect and preserve flight data, the system has specialised storage components. The recorded data can be accessed for post-flight analysis and investigation reasons and is normally kept on file for a set amount of time as required by regulatory agencies. Real-time data transfer from the aircraft to the ground station or repair facility is another feature of certain sophisticated FDAS deployments. With the use of this capacity, ground staff may keep an eye on flying parameters, carry out remote diagnostics, and support the flight crew as needed. Real-time data transmission facilitates effective maintenance procedures, improves situational awareness, and allows for quick decision-making.

## Scope for future works in this domain

Integration of newer and more sophisticated sensors, such as improved weather radar systems, advanced avionics, and improved engine monitoring sensors, can provide more accurate and comprehensive data, which can improve situational awareness, increase flight safety, and enable more accurate performance monitoring and analysis. Opportunities for FDAS to support these systems are created by the development of autonomous aircraft and unmanned aerial vehicles (UAVs). FDAS can play a critical role in monitoring and recording flight data for autonomous operations, providing insightful feedback for system optimization, and ensuring regulatory compliance.

Data Retrieval: An exclusive storage system for capturing and storing flight data is often included with FDAS. With the help of this storage system, it is guaranteed that the recorded data will be safely stored and be accessible for post-flight analysis and inquiry.

Real-time Data transfer: A number of FDAS systems allow for the transfer of data in realtime from the aircraft to a ground station or repair facility. This enables ground staff to keep an eye on flight parameters, conduct remote diagnostics, and help the flight crew as needed.

Data Analysis and Reporting: Following the flight, the recorded data is examined to assess the performance of the aircraft, spot any abnormalities or potential safety issues, and produce reports. The effectiveness of operations, maintenance procedures, and safety standards may all be improved using data analysis.

Accurate Incident and Accident Investigations: The FDAS is essential to this process. Investigators can use the recorded flight data, which includes flight characteristics and cockpit voice recordings, to gather precise and trustworthy information about the circumstances that led to an incident or accident. This knowledge aids in identifying the root causes, mitigating variables, and remedial activities required to stop such incidents from happening again.

Data-Driven Decision Making: FDAS data, is used by operators and aviation authorities. Identification of trends, patterns, and departures from usual operations is made possible by the study of flight data. Decision-making procedures including maintenance, operations, safety upgrades, and regulatory compliance are guided by this data.

## **Outcome of the project works**

Flight Data Acquisition System is designed in which random inputs are given and stored in memory. The design includes SPI protocol as Interface and FIFO to store data in the memory. The design act as DUT (Design under test) which is verified using UVM (Universal Verification Methodology). Each block of the UVM is designed, which includes: UVM Interface, Sequence, Sequence Item, Sequencer, Driver, Monitor, Agent, Scoreboard, Environment, Test. Here are the simulation results of flight data acquisition system which is verified using UVM

We have applied for a publication in International Journal of VLSI and Signal Processing and got Acceptance for publication from the journal.

## References

- [1] Ning Jia, Hang Chen and Jun Tian "A Design of Configurable Multi-type Flight Data Acquisition System", IEEE conference on Communication and Computing, 2021
- [2] Guojin Peng, Wei Zhang, Manting Liu "A method of Data Acquisition Network Delay Measurement for AFDX Avionics System in Flight test", 7th international conference on signal and image processing, 2022
- [3] John W. Dyer and Benjamin Douglas "Portable airborne Data Acquisition for Flight Testes", "IEEE international instrumentation and measurements Technology Conference Proceedings", 2019
- [4] G.V.Jayaramaiah and Chetan.Umadi "Fpga implementation of multiprotocol data acquisition system using vhdl" July-2020
- [5] Ufuk Sakarya and Ibrahim Hokelek "Universal Verification Methodology Application of ARINC429 for Airborne Electronic Hardware Certification", 2021.
- [6] Harikrishnan.K, Vishwas.H.N, Vineetha Jain K.V, Dr.Ramesh Chinthala, " Sensor data acquisition and de-noising using fpga", 2020.
- [7] J. AN, S. LI, et al. "Design of Architecture of Spaceflight Test Data Centers Using Cloud Platform," Journal of Spacecraft TT&C Technology, vol. 35 pp.: 137-145, 2016.
- [8] D. Laney, 3D data management: controlling data volume, velocity, and variety, META Group Research Note, 2019.
- [9] B. Zhao, Design and research of engine parameter collector system[D]. Xi'an, Northwestern Polytechnical University, 2018.
- [10] J. Zhang, G. Peng, B. Liu, W. Zhang, Hardware design of data acquisition system for a certain aircraft engine. Modern Electronics Technique," Xi'an, vol. 37, pp. 67-69, November 2014. [11] J. AN, S. LI, et al. "Design of Architecture of Spaceflight Test Data Centers Using Cloud Platform," Journal of Spacecraft TT&C Technology, vol. 35 pp.: 137-145, 2016.
- [12] D. Laney, 3D data management: controlling data volume, velocity, and variety, META Group Research Note, 2001.
- [13] B. Zhao, Design and research of engine parameter collector system[D]. Xi'an, Northwestern Polytechnical University, 2004.

- [14] J. Ma, C. Zhou. Data acquisition and processing technology. Xi'an: Xi'an Jiaotong University Press, 2001.
- [15] J. Zhang, G. Peng, B. Liu, W. Zhang, Hardware design of data acquisition system for a certain aircraft engine. Modern Electronics Technique," Xi'an, vol. 37,pp. 67-69, November 2014.
- [16] Christophersen B H, Pickell W J, et al. Small Adaptive Flight Control Systems for UAVs using FPGA/DSP Technology. Georgia Institute of Technology, 2004.
- [17] Ye Fan "FPGA-Based Data Acquisition System" 2011 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC). [18] Shi-zhen Huang and Rui-Qi Chen "FPGA- based IoT Sensor HUB" 2018 International Conference on Sensor Networks and Signal Processing (SNSP).
- [19] Swarup S. Mathurkar , Rahul B Lanjewar, Nilesh Patel and Rohit S.Somkuvwar "Smart Sensor Based Monitoring System For Agriculture Using Field Programmable Gate Array" 2014 International Conference on Circuit, Power and Computing Technologies [ICCPCT]
- [20] Shuang Bao, Hairong Yan , Qingping Chi , Zhibo Pang and Yuyin Sun "FPGABased Reconfigurable Data Acquisition System ForIndustrial Sensors" IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, VOL. 13, NO. 4, AUGUST 2017.

## Appendix

### Design Under Test(DUT):

```
module data_acq(adc_data,clk,reset,wr_en,rd_en,data_out);
input [15:0] adc_data; input clk,reset,wr_en,rd_en; output
reg[15:0] data_out; reg [15:0] data,fifo_data; reg
wr_en,rd_en;

spi dut1(.adc_data(adc_data),.clk(clk),.reset(reset),.data(data)); fifo
dut2(.data(data),.wr_en(wr_en),.rd_en(rd_en),.clk(clk),.reset(reset),.fifo_data(fifo_data));
ram
dut3(.fifo_data(fifo_data),.clk(clk),.reset(reset),.wr_en(wr_en),.rd_en(rd_en),.data_out(da
ta_out));

endmodule

module spi(adc_data,clk,reset,data);
input [15:0] adc_data; input
clk,reset; output reg [15:0] data; reg
[4:0] count; reg [15:0] mosi; reg
cs_l; reg sclk; reg [2:0] state;
always@(posedge clk)
if(reset) begin
count<=5'd16;
cs_l<=1'b1;
sclk<=1'b0;
end else
begin
case(state)
0:begin
sclk<=1'b0;
cs_l<=1'b0;
```

```
mosi<=adc_d
ata;
count<=count
-1; state<=1;
end 1:begin
sclk<=1'b0;
if(count>0)
state<=0; else
begin
count<=16;
state<=0; end
end
default:state<
=0; endcase
end assign
data=mosi;
endmodule
```

```
module fifo(data,wr_en,rd_en,clk,reset,fifo_data);
input [15:0] data; input wr_en,rd_en,reset,clk;
output reg [15:0] fifo_data;
reg [15:0] data; reg [4:0]
rd_ptr, wr_ptr; reg
[5:0]fifo_cnt; reg [15:0]
fifo_ram[128]; reg
empty,full; assign empty =
(fifo_cnt==0); assign full =
(fifo_cnt==32);
```

```
always@(data,wr_en,rd_en)
begin
```

```
        if(wr_en==1'b1)    //write into RAM
            begin
fiffo_ram[wr_ptr]=data;          end
            else if (rd_en==1'b1) // read from RAM
                begin
                    fiffo_data=fiffo_ram[rd_ptr];
end                else                begin
fiffo_data=fiffo_data;          end
end
always@(wr_en,rd_en,reset) begin:
pointer
    if( reset )
begin
wr_ptr = 0;
rd_ptr = 0;
end
    else
begin
    wr_ptr <= ((wr_en && !full) | | (wr_en && rd_en)) ? wr_ptr+1 : wr_ptr;
rd_ptr <= ((rd_en && !empty) | | (!wr_en && rd_en)) ? rd_ptr+1 : rd_ptr;
end end
always @( posedge clk ) begin:
counter
    if( reset )    fiffo_cnt
<= 0; else    begin
case ({wr_en,rd_en})
    2'b00 : fiffo_cnt <= fiffo_cnt;
    2'b01 : fiffo_cnt <= (fiffo_cnt==0) ? 0 : fiffo_cnt-1;
    2'b10 : fiffo_cnt <= (fiffo_cnt==32) ? 32 : fiffo_cnt+1;
    2'b11 : fiffo_cnt <= fiffo_cnt;
default: fiffo_cnt <= fiffo_cnt;
endcase    end end endmodule
```



```
module ram(fifo_data,clk,reset,wr_en,rd_en,data_out);
input [15:0] fifo_data; input clk,reset,wr_en,rd_en;
output reg [15:0] data_out; reg [4:0]
wr_ptr_m,rd_ptr_m; reg [15:0] mem[(2**7):0];
always@(posedge clk )    begin
if(reset==1'b1)
    begin
        for(int unsigned i=0;i<2**8;i++)
        begin
            mem[i]<=0;
        end
        data_out<=0;
    end
end
always@(wr_en,rd_en,reset) begin:
memory
    if( reset )
    begin
        wr_ptr_m = 0;
        rd_ptr_m = 0;
    end
    else
    begin
        wr_ptr_m <= ((wr_en ) || (wr_en && rd_en)) ? wr_ptr_m+1 : wr_ptr_m;
        rd_ptr_m<= ((rd_en ) || (!wr_en && rd_en)) ? rd_ptr_m+1 : rd_ptr_m;
    end
    end
    always@(fifo_data,wr_en,rd_en)
    begin
        if(wr_en==1'b1)          //write into RAM
        begin
            mem[wr_ptr_m]=fifo_data;
        end
        else if (rd_en==1'b1) // read from RAM
        begin
```

```
        data_out=mem[rd_ptr_m];
    end        else
        begin
    data_out=data_out;
    end        end
endmodule
```

**UVM Testbench:**

```
import uvm_pkg::*;
`include "uvm_macros.svh"
//-----
//    data_interface
//----- interface
data_if(input logic clk,reset);
//-----
//declaring the signals
//-----
logic wr_en; logic rd_en; logic
[15:0] adc_data; logic [15:0]
data_out;
//-----
//driver clocking block
//-----
clocking driver_cb @(posedge clk);
default input #1 output #1;
output wr_en;        output rd_en;
output adc_data; input data_out;
endclocking
//-----
//monitor clocking block
```

```
//-----
clocking monitor_cb @(posedge clk);
default input #1 output #1;  input
wr_en;  input rd_en;  input
adc_data;  input data_out;
endclocking

//-----
//driver modport
//-----
modport DRIVER (clocking driver_cb,input clk,reset);
//-----
//monitor modport
//-----
modport MONITOR (clocking monitor_cb,input clk,reset);  endinterface

//-----
// sequence item
//-----
class data_seq_item extends uvm_sequence_item;
//-----
//data and control fields
//-----
bit    wr_en; bit    rd_en;
rand bit [15:0] adc_data;    bit
[15:0] data_out;
//-----
//Utility and Field macros `uvm_object_utils_begin(data_seq_item)
`uvm_field_int(adc_data,UVM_ALL_ON)
`uvm_object_utils_end

//-----
```

```
//-----  
//Constructor  
//-----  
function new(string name = "data_seq_item");  
super.new(name); endfunction  
function string convert2string(); return  
$psprintf("adc_data=%0h ",adc_data);  
endfunction  
constraint data {adc_data<350;} endclass  
  
//======  
=  
// data_sequence - random stimulus  
//======  
=  
class data_sequence extends uvm_sequence#(data_seq_item);  
`uvm_object_utils(data_sequence)  
//-----  
//Constructor  
//-----  
function new(string name = "data_sequence");  
super.new(name); endfunction  
//-----  
// create, randomize and send the item to driver  
//-----  
task body(); data_seq_item req;  
repeat(100) begin  
    req=new();  
start_item(req);  
assert(req.randomize());
```

```
finish_item(req); end
endtask endclass

//-----
//      sequencer
//-----
class data_sequencer extends uvm_sequencer#(data_seq_item);
  `uvm_component_utils(data_sequencer)
  //-----
  //constructor
  //-----
  function new(string name, uvm_component parent);
super.new(name,parent); endfunction
  function void build_phase(uvm_phase phase);
super.build_phase(phase); endfunction
endclass

//-----
//      driver
//-----
`define DRIV_IF vif.DRIVER.driver_cb class data_driver
extends uvm_driver #(data_seq_item);
  //-----
  // Virtual Interface

  virtual data_if vif;
  `uvm_component_utils(data_driver)
  //-----
  // Constructor

  //-----
```

```
//-----  
function new (string name, uvm_component parent);  
super.new(name, parent); endfunction : new  
  
//-----  
// build phase  
//-----  
function          void          build_phase(uvm_phase          phase);  
super.build_phase(phase);  
    if(!uvm_config_db#(virtual  data_if)::get(this, "", "vif", vif))  
begin  
    `uvm_error("build_phase","driver  virtual  interface  failed");  
end  
endfunction: build_phase  
  
//-----  
// run phase  
//-----  
virtual task run_phase(uvm_phase phase);  
super.run_phase(phase);  forever begin  
data_seq_item trans;  
    seq_item_port.get_next_item(trans);  
    uvm_report_info("DATA_DRIVER          ",          $psprintf("Got          Transaction  
%s",trans.convert2string()));  
    @(posedge vif.DRIVER.clk);  
    `DRIV_IF.wr_en<=1;  
    `DRIV_IF.rd_en<=0;  
    `DRIV_IF.adc_data<=trans.adc_data;  
    //-----  
    //Shifting  
    //-----  
    @(posedge vif.DRIVER.clk);
```

```
`DRIV_IF.wr_en<=0;
`DRIV_IF.rd_en<=0;    repeat(5)
@(posedge vif.DRIVER.clk);
    `DRIV_IF.rd_en<=1;

    //-----
    //Reading
    //-----
    @(posedge vif.DRIVER.clk);
trans.data_out=`DRIV_IF.data_out;
seq_item_port.item_done();  end
endtask : run_phase
//-----
// drive - transaction level to signal level
// drives the value's from seq_item to interface signals
//----- endclass
: data_driver

//-----
//          monitor
//-----
`define MON_IF vif.MONITOR.monitor_cb class
data_monitor extends uvm_monitor;
//-----
// Virtual Interface
virtual data_if vif;
//-----
// analysis port, to send the transaction to scoreboard
//-----

//-----
```

```
uvm_analysis_port #(data_seq_item) item_collected_port;
//-----
// The following property holds the transaction information currently
// begin captured (by the collect_address_phase and data_phase methods).
//-----
`uvm_component_utils(data_monitor)

//-----
// new - constructor
//-----
function new (string name, uvm_component parent);
super.new(name, parent);
    item_collected_port = new("item_collected_port", this);
endfunction

//-----
// build_phase - getting the interface handle
//-----
function void build_phase(uvm_phase phase);
super.build_phase(phase);
    if(!uvm_config_db#(virtual data_if)::get(this, "", "vif", vif))
        `uvm_error("build_phase", "No virtual interface specified for this monitor instance")
endfunction: build_phase

//-----
// run_phase - convert the signal level activity to transaction level.
// i.e, sample the values on interface signal and assigns to transaction class fields
//-----
virtual task run_phase(uvm_phase phase); super.run_phase(phase);
```



```

    forever begin
data_seq_item trans;
trans=new();
wait(`MON_IF.wr_en==1);
    fork
        trans.adc_data=`MON_IF.adc_data;
    join
        wait(`MON_IF.wr_en==0          );
repeat(15) begin
    @(posedge vif.MONITOR.clk);
    end
    wait(`MON_IF.rd_en==1 );
    fork
        trans.data_out=`MON_IF.data_out;
    join
        item_collected_port.write(trans);
end
    endtask : run_phase endclass
: data_monitor

class fun_cov extends uvm_subscriber#(data_seq_item);
    `uvm_component_utils(fun_cov)
    data_seq_item trans; covergroup
cg;
WDATA:coverpoint trans.wr_en { bins wd[16] = {[0:2*16-1]}; }
RDATA:coverpoint trans.rd_en { bins rd[16] = {[0:2*16-1]}; } endgroup
function new(string name, uvm_component parent);
super.new(name, parent); cg = new(); endfunction
//new()
function void build_phase(uvm_phase phase);
```

---

```
trans = data_seq_item::type_id::create("trans"); endfunction
function void write(data_seq_item t); this.trans = t;
cg.sample(); endfunction endclass
```

```
//-----
//    agent
//-----
class data_agent extends uvm_agent;
//-----
// component instances
//-----
data_driver  driver;
data_sequencer sequencer;
data_monitor monitor; virtual
data_if vif;
`uvm_component_utils_begin(data_agent)
`uvm_field_object(sequencer, UVM_ALL_ON)
`uvm_field_object(driver, UVM_ALL_ON)
`uvm_field_object(monitor, UVM_ALL_ON)
`uvm_component_utils_end
//-----
// constructor
//-----
function new (string name, uvm_component parent);
super.new(name, parent); endfunction : new
//-----
// build_phase
virtual function void build_phase(uvm_phase phase);
    super.build_phase(phase);

//-----
```

---

```
monitor = data_monitor::type_id::create("monitor", this);
// creating driver and sequencer only for ACTIVE agent
driver = data_driver::type_id::create("driver", this);
sequencer = data_sequencer::type_id::create("sequencer", this);
uvm_config_db#(virtual data_if)::set(this, "seq", "vif", vif);
uvm_config_db#(virtual data_if)::set(this, "driv", "vif", vif);
uvm_config_db#(virtual data_if)::set(this, "mon", "vif", vif);
if(!uvm_config_db#(virtual data_if)::get(this, "", "vif", vif)) begin
    `uvm_error("build_phase", "agent virtual interface failed");
end

endfunction : build_phase
//-----
// connect_phase - connecting the driver and sequencer port
//-----
function void connect_phase(uvm_phase phase);
super.connect_phase(phase);
driver.seq_item_port.connect(sequencer.seq_item_export);
uvm_report_info("DATA_AGENT", "connect_phase, Connected driver to sequencer");
endfunction : connect_phase endclass : data_agent

//-----
// scoreboard
//----- class
data_scoreboard extends uvm_scoreboard;
//-----
//port to receive packets from monitor
uvm_analysis_imp#(data_seq_item, data_scoreboard) item_collected_export;

//-----
```

---

```
-  
  
`uvm_component_utils(data_scoreboard) data_seq_item  
trans;  
//-----  
// new - constructor  
//-----  
function new (string name, uvm_component parent);  
super.new(name, parent);  
item_collected_export=new("item_collected_export",this);  
endfunction  
//-----  
// build_phase - create port and initialize local memory  
//-----  
function void build_phase(uvm_phase phase);  
super.build_phase(phase); endfunction:  
build_phase  
//-----  
// write task - recives the pkt from monitor and pushes into queue  
//-----  
function void write(data_seq_item trans); trans.print();  
`uvm_info("SCOREBOARD",$sformatf("-----::RESULT:: -----"),UVM_LOW)  
  `uvm_info("", $sformatf("adc_data:%0h ",trans.adc_data),UVM_LOW)  
`uvm_info("", $sformatf("data_out:%0h ",trans.data_out),UVM_LOW)  
endfunction endclass : data_scoreboard  
  
//-----  
//          environment  
//----- class  
data_env extends uvm_env;  
  
//-----  
  
-  

```

```
// agent and scoreboard instance
//-----
data_agent    agnt;
data_scoreboard scb; virtual
data_if vif;
`uvm_component_utils(data_env)
//-----
// constructor
//-----
function new(string name, uvm_component parent);
super.new(name, parent); endfunction : new
//-----
// build_phase - crate the components
//-----
function      void      build_phase(uvm_phase      phase);
super.build_phase(phase);
    agnt = data_agent::type_id::create("agnt", this);    scb =
data_scoreboard::type_id::create("scb", this);
    uvm_config_db#(virtual data_if)::set(this, "agnt", "vif", vif);
    uvm_config_db#(virtual data_if)::set(this, "scb", "vif", vif);
    if(! uvm_config_db#(virtual data_if)::get(this, "", "vif", vif))
begin
    `uvm_error("build_phase", "Environment virtual interface failed")
end
endfunction : build_phase
//-----
// connect_phase - connecting monitor and scoreboard port
//-----
function void connect_phase(uvm_phase phase); super.connect_phase(phase);
    agnt.monitor.item_collected_port.connect(scb.item_collected_export);
uvm_report_info("data_ENVIRONMENT", "connect_phase, Connected monitor to scoreboard");
```

```
endfunction : connect_phase endclass

: data_env
    //-----
    //      test
    //-----

class data_test extends uvm_test;
    `uvm_component_utils(data_test)
    //-----
    // env instance
    //-----
    data_env env; virtual data_if vif;
    //-----
    // constructor
    //-----
    function new(string name ,uvm_component parent);
    super.new(name,parent); endfunction : new
    //-----
    // build_phase
    //-----
    function void build_phase(uvm_phase phase);
    super.build_phase(phase);    // Create the env
        env = data_env::type_id::create("env", this);
        uvm_config_db#(virtual data_if)::set(this, "env", "vif", vif);
        if(! uvm_config_db#(virtual data_if)::get(this, "", "vif", vif))
        begin
            `uvm_error("build_phase","Test virtual interface failed")
        end
    endfunction : build_phase    task
    run_phase(uvm_phase phase);
    data_sequence seq;
```

```
seq = data_sequence::type_id::create("seq",this);
phase.raise_objection(this,"starting main phase"); $display("%t
Starting sequence spi_seq run_phase",$time);
seq.start(env.agnt.sequencer);
    #500ns;
phase.drop_objection(this,"finished main phase");
endtask : run_phase endclass
```

```
//-----
//          testbench.sv
//-----
module testbench_top;
    //-----
    //clock and reset signal declaration
    //-----
    bit clk; bit reset;
    //-----
    //clock generation
    //-----
    always #5 clk = ~clk;
    //-----
    //reset Generation
    //-----
    initial begin    reset = 1;
        #15 reset =0;
    end
    //-----
    //interface instance
    //-----
    data_if intf(clk,reset);
    //-----
    //DUT instance
```

```
//-----  
data_acq DUT (  
    .clk(intf.clk),  
    .reset(intf.reset),  
    .wr_en(intf.wr_en),  
    .rd_en(intf.rd_en),  
    .adc_data(intf.adc_data),  
    .data_out(intf.data_out)  
);  
//passing the interface handle to lower heirarchy using set method  
//and enabling the wave dump  
initial begin  
    uvm_config_db#(virtual data_if)::set(uvm_root::get(),"*", "vif",intf);  
    $dumpfile("dump.vcd");  
$dumpvars; end  
//-----  
//calling test  
//-----  
initial begin  
run_test("data_test"); end  
  
endmodule
```



## Paper Presented / Publications related to the Project Work in Conferences & in Journals

We have applied for a publication in International Journal of VLSI and Signal Processing and got Acceptance for publication from the journal.

---

Editor SSRG - IJVSP

to me ▾

Mon, May 29, 1:20 PM (7 days ago) ☆ ↶

Dear Sir/Ma'am,

We are delighted to inform you that your manuscript "ID: VSP23JUNE001, Title: **DESIGN AND VERIFICATION OF FLIGHT DATA ACQUISITION SYSTEM USING UVM**" has been **"Accepted for Publication"** in SSRG-International Journal of VLSI & Signal Processing (SSRG-IJVSP)", ISSN: 2394-2584.

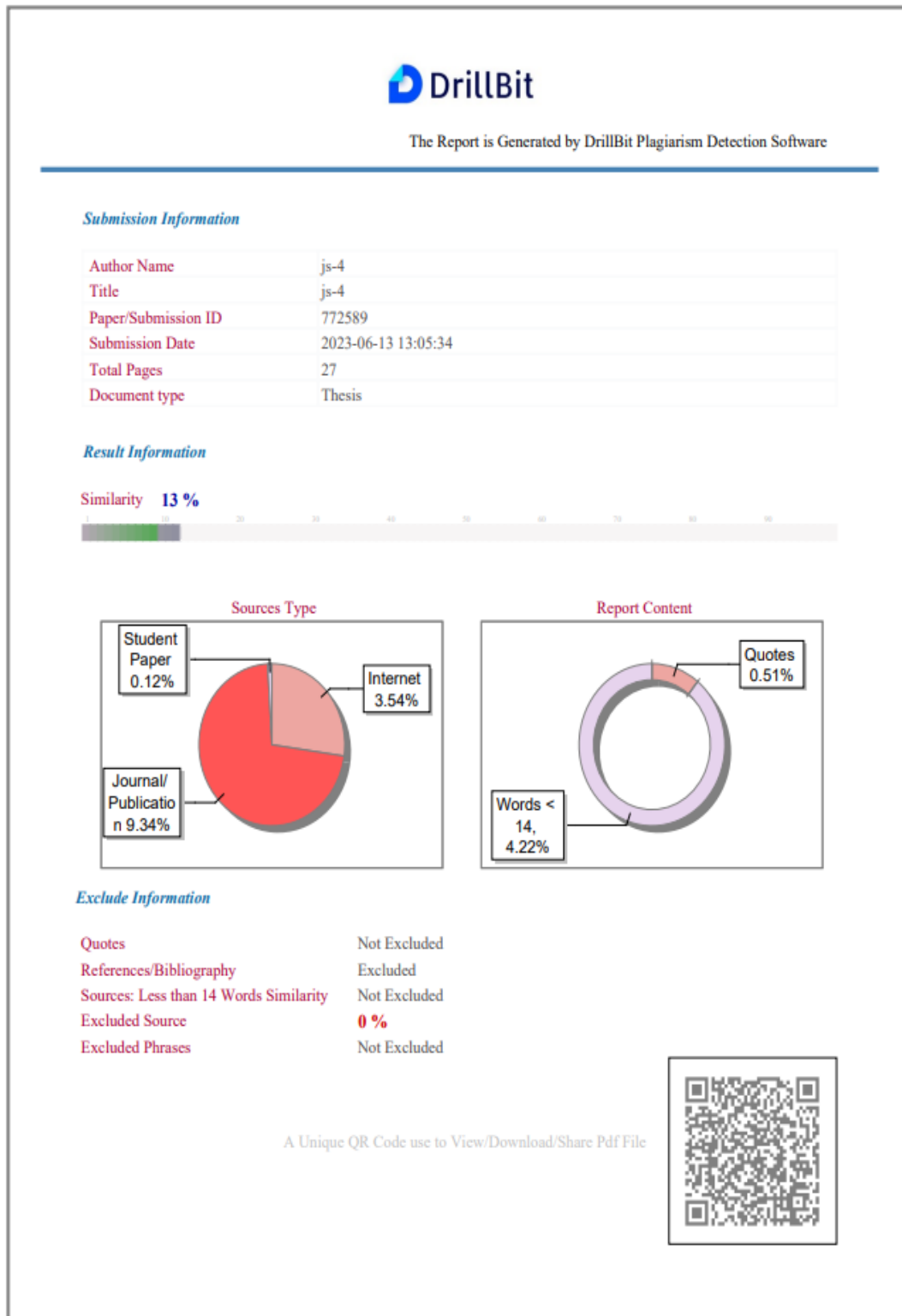
Please see the attached reviewer comments for further details about necessary revisions.


## Awards, Certificates Recognitions & Photographs





## Plagiarism reports





**DrillBit Similarity Report**

---

# 13

**SIMILARITY %**

# 40

**MATCHED SOURCES**

# B

**GRADE**

**A-Satisfactory (0-10%)**  
**B-Upgrade (11-40%)**  
**C-Poor (41-60%)**  
**D-Unacceptable (61-100%)**

---

LOCATION	MATCHED DOMAIN	%	SOURCE TYPE
1	Current Status of Flight Evaluation of DGPS-INS Hybrid Navigation System at NAL by Shingu-1994	2	Publication
2	Current Status of Flight Evaluation of DGPS-INS Hybrid Navigation System at NAL by Shingu-1994	1	Publication
3	Current Status of Flight Evaluation of DGPS-INS Hybrid Navigation System at NAL by Shingu-1994	1	Publication
4	IEEE 2011 IEEE International Conference on Signal Processing, Commun	1	Publication
5	eee.sairam.edu.in	1	Publication
6	Verification of FM9801 An Out-of-Order Microprocessor Model with Spec by Ju-2002	1	Publication
7	llibrary.co	<1	Internet Data
8	avxlive.icu	<1	Internet Data
9	Thesis Submitted to Shodhganga Repository	<1	Publication
10	www.ajol.info	<1	Publication
11	slidelegend.com	<1	Internet Data
12	moam.info	<1	Internet Data

13	Application of load monitoring in appliances energy management A review by Abubakar-2017	<1	Publication
14	www.arxiv.org	<1	Publication
15	A generic energy optimization framework for heterogeneous platforms using scalin by Gupta-2016	<1	Publication
16	Nutritional immunity transition metals at the pathogenhost interfa by Hood-2012	<1	Publication
17	www.dx.doi.org	<1	Publication
18	amnation.com	<1	Internet Data
19	Application of NX Knowledge Fusion module for the Design Automation of an Auto mo by Bernal-2012	<1	Publication
20	flex.consecoard.am	<1	Internet Data
21	Preservation and technological obsolescence Portuguese contemporary musical her by Pires-2018	<1	Publication
22	qdoc.tips	<1	Internet Data
23	Using palaeoecological and palaeoenvironmental records to guide restoration, con by Riedinger-Whitmore-2016	<1	Publication
24	www.freepatentsonline.com	<1	Internet Data
25	hoc.ilr.cornell.edu	<1	Internet Data
26	www.thefreelibrary.com	<1	Internet Data
27	Efficient computation of high-order Meixner moments for large-size signals and i by Daoui-2020	<1	Publication
28	moam.info	<1	Internet Data

29	<a href="http://www.caeaccess.org">www.caeaccess.org</a>	<1	Internet Data
30	<a href="http://www.freepatentsonline.com">www.freepatentsonline.com</a>	<1	Internet Data
31	<a href="http://allbankpapers.blogspot.com">allbankpapers.blogspot.com</a>	<1	Internet Data
32	<a href="http://biomedres.us">biomedres.us</a>	<1	Internet Data
33	Development of Integrated Data Acquisition System for Logistics Packaging Materi by Yan-2013	<1	Publication
34	<a href="http://qdoc.tips">qdoc.tips</a>	<1	Internet Data
35	Smart IGBTs for advanced distribution ignition systems	<1	Student Paper
36	The spatiality of trust Factors influencing the creation of trust and by Nilsson-2015	<1	Publication
37	<a href="http://www.bdatechbrief.com">www.bdatechbrief.com</a>	<1	Internet Data
38	<a href="http://www.jove.com">www.jove.com</a>	<1	Internet Data
39	<a href="http://www.linguee.com">www.linguee.com</a>	<1	Internet Data
40	<a href="http://www.onlinejournal.in">www.onlinejournal.in</a>	<1	Publication

## CO-PO Mapping Justification Sheets

Title of the Project: “ Design and Verification of Flight Data Acquisition System using UVM ”

### CO-PO Mapping:

Mapping of Course Outcomes to Program Outcomes and Program Specific Outcomes:

CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
CO1	3	3	2	2	2	2	2	2	2	2	2	2	2	3
CO2	3	3	3	2	2	2	3	2	2	2	2	2	2	3
CO3	3	2	3	3	3	2	3	2	2	2	2	2	2	2
CO4	2	2	2	3	3	3	2	2	2	2	2	2	2	3
CO5	2	2	2	2	2	2	3	3	3	3	2	2	2	3
CO6	1	1	1	1	2	2	2	2	2	3	3	3	2	2
CO AVG	2.3	2.1	2.1	2.1	2.3	2.1	2.5	2.1	2.1	2.3	2.1	2.1	2	2.6

\*High=3, Medium=2, Low=1

### Justification

CO1	We shall demonstrate various concept involved in VLSI
CO2	We will identify the problem and propose the possible solution through literature survey.
CO3	We will design and develop engineering solutions to problems arising in flight data recorder.
CO4	We will use Modelsim and Questasim simulation for the proposed solution and articulate the work.
CO5	The project proposes design of flight data acquisition system for the safety of aircrafts.
CO6	We shall complete the proclaimed work within stipulated time span with financial constraints.



## PO-PSO Mapping for Project – 2022-23 Batch Number: R-06

USN	Name
1DS19EC084	Nainshree Raj
1DS19EC096	Pallavi G
1DS19EC099	Poornima M
1DS20EC417	Lakshmi S

Guide Name: Dr. Jamuna S

## Justification for PO &amp; PSO mapping for Project

Project Title		Design and verification of Flight Data Acquisition System using UVM
PO ▼	Levels 3/2/1	Justification
PO1	3	We shall apply various concepts in the field of VLSI to arrive at a fail-proof design of flight data acquisition system.
PO2	3	Identifying the various issues that need to be tackled during project implementation with the help of relevant literature survey
PO3	3	Designing a suitable solution for the problem arising in flight data recorder.
PO4	3	Using research based knowledge several approaches will be analysed to simulate and rectify the problem.
PO5	3	We will make use of simulation tools like Modelsim and Questasim to exhibit the proposed design.
PO6	3	We are applying engineering practices to design data acquisition system for the safety of aircrafts.
PO7	2	We will make sure our system developed will meet the environment sustainability.
PO8	3	We are committed to the professional ethics and responsibilities during project work.
PO9	3	Functioned effectively as a team by dividing the work among the team members and implemented some modification due to team work
PO10	3	Our communication skills are improved by presenting the project and preparing report
PO11	3	We are learning about the various stages involved in project management to meet the necessary requirements
PO12	2	By laying the foundation to learn more concepts involving the design using latest VLSI techniques
PSO1	2	This project proposes a design of flight data acquisition system for the safety of aircrafts.
PSO2	2	We shall design a data acquisition system, verify the same using UVM and to implement on FPGA.

## Budget Estimation Sheets

### Budget Estimation Batch Number: R-06

USN	Name
1DS19EC084	Nainshree Raj
1DS19EC096	Pallavi G
1DS19EC099	Poornima M
1DS20EC417	Lakshmi S

### Guide Name: Dr.Jamuna S

Sl. No.	Particulars	Estimated Cost in Rs.
1	Sensors	500
2	Printing project reports	2500
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
Total		25000

