System Design Document

For

Ingenion Telemetry Web Server

Team members:
Ryan Flinchum
Morgan Smith
Thomas Swenson
Ricky Nelson
Jack Capuano
Conrad Prisby
Hamilton Henneberg

Version/Author	**Date**	**Changes**
v1.0 / Hamilton Henneberg	9/22/23	Creation of Document Initialization of 1 - 2.3
v1.1 / Jack Capuano	9/29/23	Review & Revision
v1.2 / Thomas Swenson	9/29/23	Final Review
v2.1 / Ryan Flinchum	10/31/23	● Added Context Diagram ● Added Use case diagram ● Addressed all TA feedback ● Added diagram descriptions for every diagram
v3.0 / Morgan Smith	11/29/23	Modified 1.2 Added Glossary Table Added to 1.2.1
v3.1/ Ryan Flinchum	12/1/23	● Added description for Context diagram ● Added FPGA Hardware Block design and description ● Revised Purpose and Scope ● Revised Design Constraints section ● Added Level 0 Data Flow Diagram and description
v3.2 / Hamilton Henneberg	12/1/23	Created UI, Integration, and Class Diagrams Corrected sections 3, 4, and 5 based on previous feedback finalized all sections and figures
v1.1.1 / Ryan Flinchum	2/5/24	Updated necessary information

TABLE OF CONTENT

SYSTEM DESIGN DOCUMENT

1 INTRODUCTION

As the aerospace industry continues to integrate more deeply with advanced software systems, the field of satellite technology is experiencing unprecedented growth in both complexity and significance. The essence of contemporary satellite operations hinges on the seamless acquisition and management of telemetry data, a task that grows more challenging with the escalation of satellite system intricacies. In this milieu, the necessity for an advanced telemetry web server is both clear and pressing—the need for a system that not only captures and displays telemetry data but also enhances the ease of access for engineers during the testing phase is paramount. The Ingenion Telemetry Web Server project is conceived to fulfill this need.

The project addresses the intricacies of telemetry data management by proposing a sophisticated web server designed to streamline the interaction between telemetry systems and their operators. The Ingenion Telemetry Web Server aims to elevate the functionality of existing Ground Support Equipment (GSE) systems, providing engineers with an intuitive and interactive platform to monitor, control, and validate satellite hardware. This document introduces the Ingenion Telemetry Web Server project, highlighting its inception, development, and the envisioned scope, ultimately detailing a system that stands as a beacon of innovation in the telemetry and satellite testing arena.

1.1 Purpose and Scope

The Ingenion Telemetry Web Server is purpose-built to serve as an integral tool in the testing and validation of satellite hardware, providing a real-time telemetry data interface that is both robust and user-friendly. Its scope is defined by the imperative to offer engineers a system that not only reliably processes telemetry data but also does so in a manner that facilitates ease of access and interaction. The server is intricately designed to augment the capabilities of existing Total Verification Systems (TVS), used extensively by space agencies, with an enhanced range of features that cater to modern satellite testing requirements.

This includes the strategic integration of a Xilinx MicroBlaze soft-core CPU within an Artix 7 FPGA, alongside the deployment of FreeRTOS—an open-source real-time operating system. These integrations ensure that the Ingenion Telemetry Web Server not only hosts an interactive web server but also establishes TCP over Ethernet connections for seamless data management. While the server is adept at handling complex telemetry data, it is crafted with a focus on simplicity and user-centric design, ensuring that even non-specialist users can operate it with minimal training.

It is important to note that while the Ingenion Telemetry Web Server is comprehensive in its functionalities, it is not intended to serve as a development platform for telemetry algorithms. Instead, it is a facilitator, enabling the display and management of telemetry data derived from satellite testing operations. The development of a graphical user interface (GUI) is not within the scope of this project, as the system is designed to complement existing interfaces. The onus of ensuring compatibility of telemetry

algorithms with this web server rests with the users, aligning with the project's goal of offering a streamlined and efficient telemetry data management system. The project is focused on individual telemetry streams, providing a specialized solution that does not extend to broader data collection methods or the complexities of managing drone swarms or other such entities.

1.2 Project Executive Summary

1.2.1 System Overview

This section of the System Design Document provides a high-level description of the project from a management perspective and outlines the framework within which the conceptual system design was developed.

Our objective for this phase of the Ingenion project is to focus on the software development for enhancing the Total Verification System (TVS), a critical testing tool used by NASA's Goddard Space Flight Center for simulating, testing, and verifying satellite hardware components. With the FPGA hardware already developed, our attention shifts to refining and expanding the system's software capabilities.This project will consist of the integration of FreeRTOS, an open-source real-time operating system, on the Xilinx MicroBlaze soft-core CPU implemented on the Artix 7 FPGA. In addition to the FreeRTOS implementation, we will develop an interactive web server hosted on the FPGA. The web server will facilitate connections to external computers using TCP over Ethernet, enabling the system to read and display telemetry data from peripherals including GPIO, SPI, UART, and I2C.

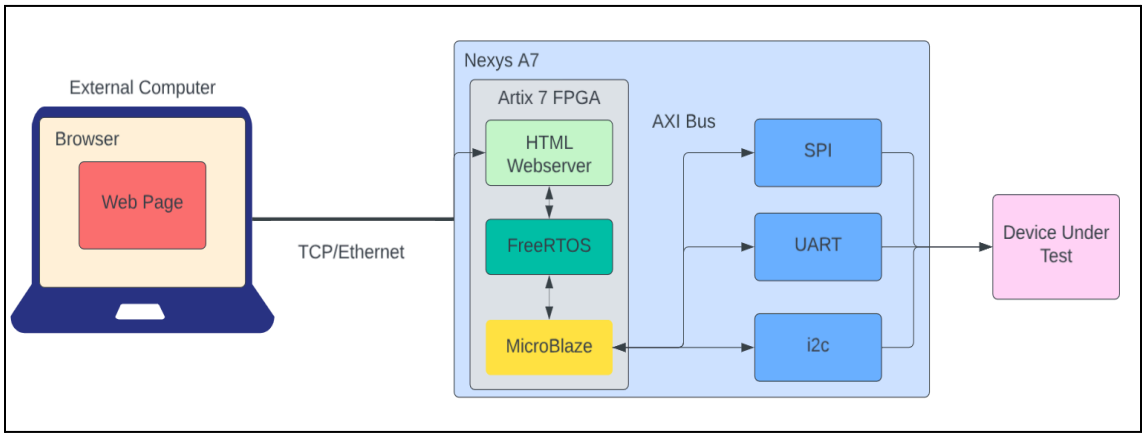


Figure 1.2.1: High-level System Architecture Diagram

As pictured in Figure 1.2.1, The Telemetry Web Server project operates in an environment containing an External Computer, a Nexys A7 FPGA and a DUT. Through ethernet and the TCP protocol the External Computer connects with the HTML web server which operates within FreeRTOS on the Microblaze soft CPU. The Web Server provides additional functionalities to the External user such as the ability to monitor,

send, or receive data sent over the AXI bus. This data that will be intercepted/generated is telemetry data involved in the testing and communications that occur on the DUT.

1.2.2 Design Constraints

This section delves into the specific limitations and assumptions that have shaped the design of the Ingenion Telemetry Web Server. It highlights the critical trade-offs considered during the project's development and elaborates on how these have directed the design choices, potentially introducing constraints.

Resource Allocation: A primary constraint has been the limited availability of FPGA resources. The project had to be designed within the confines of a single FPGA unit's logic cells, memory, and processing capabilities. This limitation necessitated a careful balance between desired features and available resources. For instance, implementing additional parallel processing capabilities was restricted by the number of logic gates available, leading to prioritization based on system requirements.

Compatibility: The system had to be fully compatible with the existing Total Verification System (TVS) and external computer systems used in current satellite testing setups. This prerequisite limited the choice of development tools and languages to those already supported by the TVS, potentially excluding more modern or efficient alternatives. Additionally, the interface protocols were constrained to those already in use by the TVS to avoid extensive reconfiguration of the existing system.

Time, Budget, Labor: With the ambitious goal of completing the project within a single academic semester, the team faced significant time constraints. The zero-budget parameter further restricted the ability to procure additional development tools or external expertise. The manpower was fixed at seven team members, which imposed a cap on the workload distribution and necessitated a highly efficient task management strategy.

Our project closely follows the architectural blueprint of the ARTY FreeRTOS Web Server. This adherence has inherently shaped our design choices, limiting modifications that could be made to the system’s structure and functionality. The necessity to remain compatible with the established architecture has influenced our system's overall flexibility in terms of hardware and software customization.

1.2.3 Future Contingencies

This section anticipates potential contingencies that may arise during the system design and development process. Contingencies refer to unforeseen circumstances or challenges that could alter the project's direction. It also addresses potential workarounds or alternative plans to mitigate the impact of these contingencies.

Interface Agreements: If interface agreements are difficult to establish, contingency plans will involve identifying alternative communication protocols or temporary solutions.

Architectural Stability: In the event that architectural stability issues arise with the chosen FPGA or CPU platform, the project team will explore alternative hardware options or adapt the design to mitigate instability.

Project Scale: If, during the course of the project, it is determined that the scope is either too large or too small to align with project objectives, contingency plans will be activated to scale the project accordingly. This may involve expanding or reducing features, adjusting resource allocation, or revisiting project goals to ensure alignment with project size and complexity.

1.3 Project References

This section provides a bibliography of key project references and deliverables that have been produced before this point.

"Arty Free rtos Web Server." ARTY FreeRTOS Web Server - Xilinx Wiki - Confluence, xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18841844/ARTY+FreeRTOS+Web+Server. Accessed 29 Sept. 2023.

"Microblaze Soft Processor Core." AMD, www.xilinx.com/products/design-tools/microblaze.html. Accessed 29 Sept. 2023.

"Microblaze Webserver Demo for the Arty Evaluation Kit." AMD, www.xilinx.com/video/fpga/microblaze-webserver-demo.html. Accessed 29 Sept. 2023.

"The LXI Primer." Lxistandard.Org, www.lxistandard.org/Documents/LXI%20Primer/The%20LXI%20Primer%20Final.pdf. Accessed 29 Sept. 2023.

1.4 Glossary

Term	Description
Ingenion Telemetry Web Server	The software component responsible for facilitating communication and interaction between the Digilent Nexys A7 development board, Xilinx Artix 7 FPGA, MicroBlaze soft-core CPU running FreeRTOS, and external client devices via Ethernet. It is a

	critical part of the Total Verification System (TVS) project.
Total Verification System (TVS) Project	A project aimed at enhancing satellite hardware testing by integrating advanced features and improving telemetry data management.
Digilent Nexys A7	A development board used in the project for interfacing with satellite hardware components.
Xilinx Artix 7 FPGA	A Field-Programmable Gate Array (FPGA) from Xilinx used to host the MicroBlaze soft-core CPU and execute the software.
FreeRTOS	An open-source real-time operating system used for running tasks on the MicroBlaze CPU.
Ethernet	A network communication protocol used for connecting the Nexys A7 board to external client devices.
TCP/IP	Transmission Control Protocol/Internet Protocol, the suite of communication protocols used for transmitting data over networks, including the internet.
User Interface (UI)	The graphical interface that allows end-users to interact with and control the telemetry system.
Telemetry Data	Data collected from satellite hardware components, used for monitoring and control.
External Computer	A device connected to the telemetry system via Ethernet for data exchange and management.

Real-Time Web Server	A web server that provides immediate and interactive access to telemetry data and system controls.
Communication Protocols	Standards like Ethernet and TCP/IP used for reliable data exchange within the system.
Xilinx MicroBlaze CPU	The soft-core CPU integrated into the Artix 7 FPGA for processing tasks.
HTTP Protocol: Hypertext Transfer Protocol	Used for communication between web browsers and the web server.
WebSockets	A communication protocol that enables real-time, bidirectional communication between the web server and connected clients.
IEEE 802.3	A standard for Ethernet communication.
HTTP POST Requests	A method for sending data from a user to a web server.
System Features	Key functionalities or capabilities of the telemetry web server, such as communication link establishment, user web interface, and telemetry data parsing and display.

2 SYSTEM ARCHITECTURE

2.1 System Hardware Architecture

This section will describe the overall hardware architecture of the Ingenion Telemetry Webserver. This will include an overview of the system's hardware components and their interconnections.

The hardware architecture of the Ingenion Telemetry Webserver comprises various components essential for its operation. These components include:

Digilent Nexys A7 Development Board: Initially, the system is developed and tested on this development board, which provides the necessary hardware interfaces and connectivity options.

Xilinx Artix 7 FPGA: The FPGA on the Nexys A7, serves as the central processing unit for the system. It houses the Xilinx MicroBlaze soft-core CPU and provides programmable logic for custom functions.

MicroBlaze Soft-Core CPU: This CPU core is implemented on the FPGA and serves as the system's primary processor. It runs the FreeRTOS operating system and manages system tasks.

Ethernet Interface: The system uses an Ethernet connection for communication with external computers and data transfer.

AXI Bus: The onboard AXI bus facilitates communication with various peripherals and data sources within the FPGA.

Peripheral Devices: These may include sensors, memory modules, and other hardware elements necessary for system functionality.

2.2 System Software Architecture

This section will describe the overall software architecture of the Ingenion Telemetry Webserver. This includes a breakdown of software modules, computer languages used, and any computer-aided software engineering tools employed in the development process. Structured organization diagrams and object-oriented diagrams will be used to illustrate the software architecture.

The software architecture of the Ingenion Telemetry Web Server consists of various software modules, functions, and components that collectively enable system functionality. These include:

FreeRTOS: This open-source real-time operating system runs on the MicroBlaze CPU, providing task scheduling and management.

Web Server Software: The software responsible for hosting the interactive web server, enabling user interactions and data display.

Communication Protocols: Modules for handling TCP/IP communication over Ethernet for external computer connectivity.

Telemetry Data Processing: Software components responsible for reading, processing, and displaying telemetry data from the AXI bus and peripheral devices.

User Interface: Modules for creating the human-machine interface through which users

interact with the system.

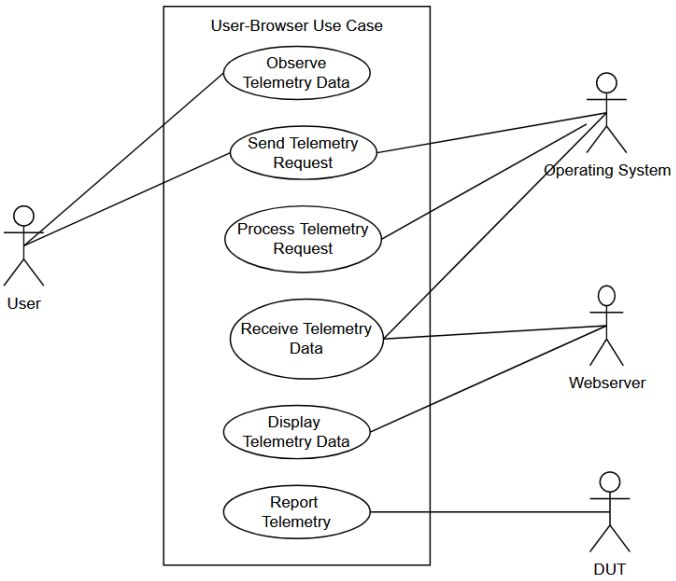


Figure 3.1.1: Ingenion Telemetry Server Use Case Diagram

The use-case diagram pictured in figure 3.1.1 illustrates the use cases within the system's architecture. As is depicted, the user has the ability to observe always-reported telemetry data, or send requests for telemetry. The Operating system, or software running on the operating system has the ability to receive the user's specific telemetry request, to process this request, and to receive telemetry data. The display of this telemetry data is handled by the Webserver which displays telemetry data that is reported by the DUT. Lastly, the design under test (DUT) reports telemetry data to the system always and this data is the foundation upon which this Telemetry Web Server is built.

2.3 Internal Communications Architecture

This section will provide an overview of the internal communications architecture within the Ingenion Telemetry Web Server system. This includes describing the communication pathways and protocols used for data exchange within the system, as well as any specific architectures implemented.

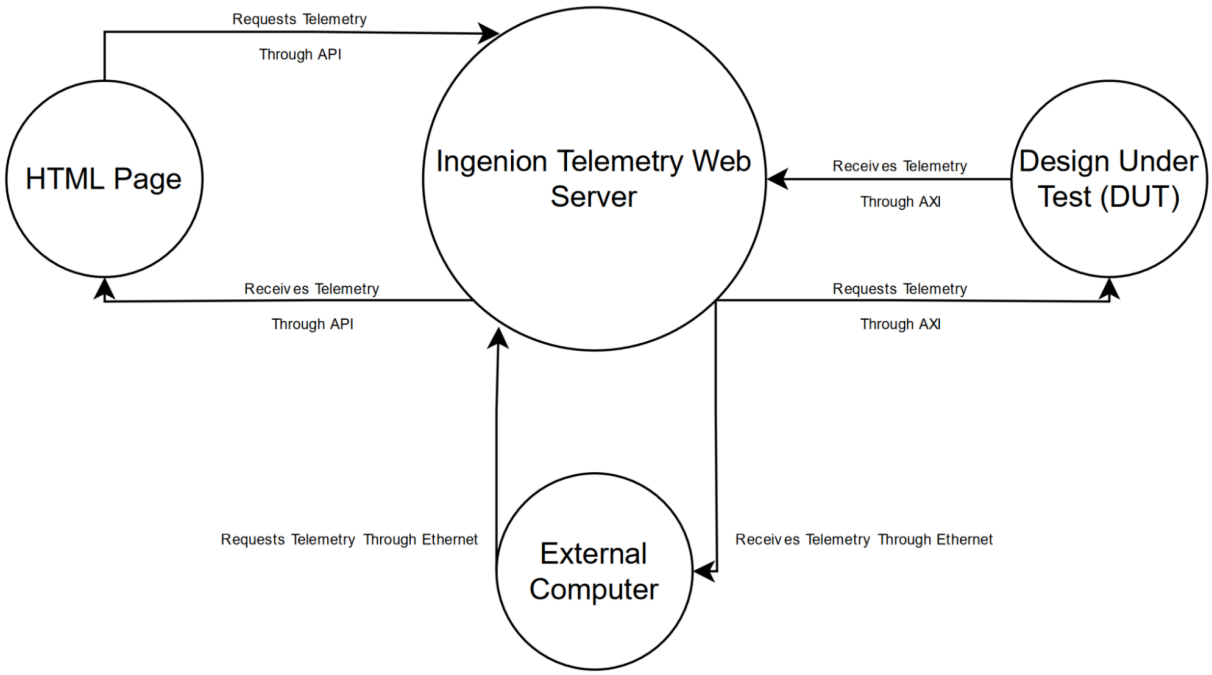


Figure 2.3.1: High-level Context Diagram

As pictured in Figure 2.3.1 the Telemetry Web Server serves as the main component within the project. This Bulb is meant to represent the FPGA design which is detailed by Figure 4.1.1. The purpose of this design is to facilitate the flow of telemetry between the DUT, External Computer, and the HTML page.

For example, the external may request telemetry from the Web Server. This request will then be processed and routed to the correct AXI endpoint within the DUT. Once received the DUT will send Telemetry to the Web Server. This Telemetry will then be sent to the External Computer.

Ethernet (TCP/IP): The system leverages Ethernet connectivity for high-speed data exchange between the Ingenion Telemetry Web Server and the External Computer. It employs the TCP/IP protocol suite to establish reliable communication channels.

AXI Bus Communication: Internally, the AXI (Advanced eXtensible Interface) bus serves as a primary communication pathway for data transfer between the MicroBlaze CPU operating on the Ingenion Telemetry Web Server and the Design under test.

Telemetry API: The HTML web server will obtain telemetry data from the system through the use of a telemetry API which is developed by our team.

3 HUMAN-MACHINE INTERFACE

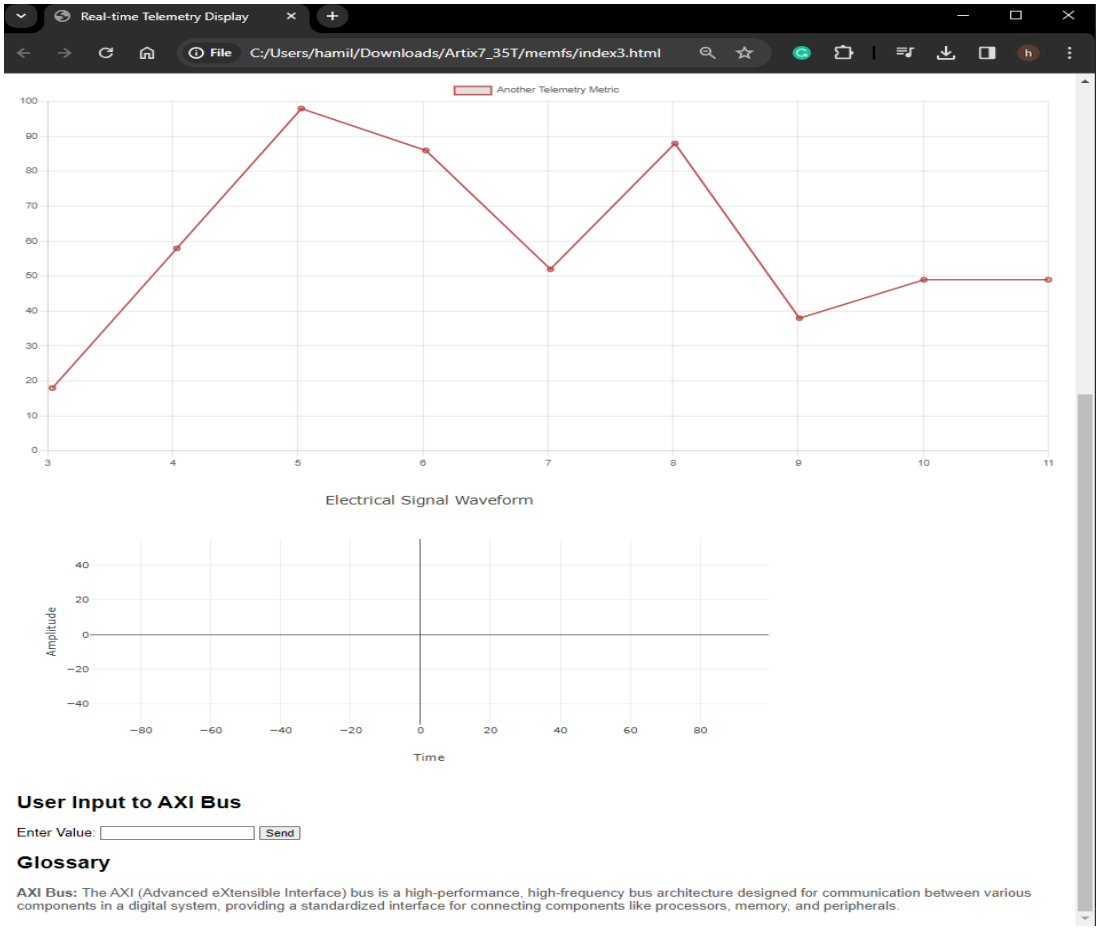


Figure 3.1: UI Example

The web page, designated as "Real-time Telemetry Display," illustrates a user-friendly interface featuring live telemetry data visualization. In this example (Figure 3.1), two dynamic line charts provide real-time insights into AXI bus metrics, supplemented by an electrical signal waveform plot. The charts dynamically update every 2 seconds, offering a snapshot of the system's performance. Additionally, a user input section allows users to contribute values directly to the AXI bus, enhancing the page's interactivity. This UI example combines diverse visualizations and user input capabilities for a concise and interactive telemetry monitoring experience.

3.1 Inputs

This section provides a detailed design of the system and subsystem inputs, focusing on the user/operator interaction. Inputs encompass user-initiated actions, commands, and data that are processed by the system. Below are the components to be addressed in this section:

User Commands: Users interact with the system by providing commands via the HTML web interface. Commands can be issued for the following purposes: capturing telemetry from specific AXI addresses, sending data to specific AXI addresses, configuring communication settings such as baud rate and ethernet speeds.

Data Entry: For the User interface, users will be required to input which AXI addresses they want to access via a text box on the HTML web page to retrieve data from this address.

3.2 Outputs

This section will focus on the primary system output of the enhanced Total Verification System (TVS) project, which is the real-time telemetry display on the web server. This display serves the purpose of providing users with live telemetry data from satellite hardware components.

Contents: The real-time telemetry display presents live telemetry data from the onboard AXI bus and other peripherals in a user-friendly web interface. A definition of the AXI bus is detailed within our glossary. The AXI bus will be connected to the DUTs and outputs will consist of test-based telemetry data from these designs under test.

4 DETAILED DESIGN

4.1 Hardware Detailed Design

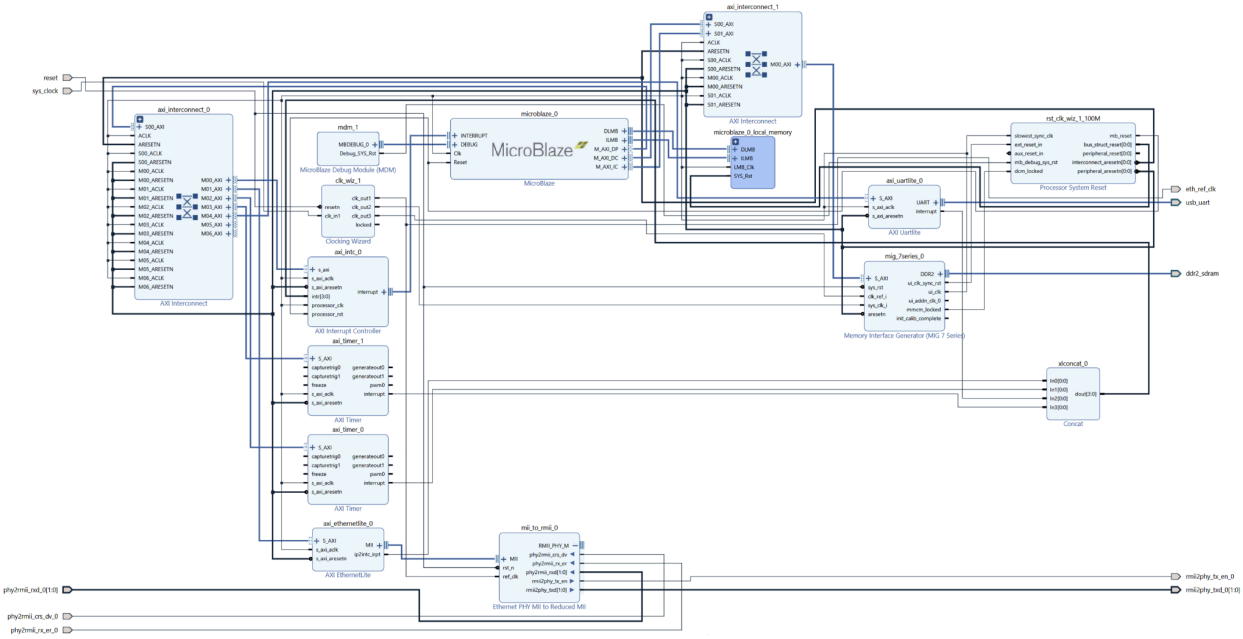


Figure 4.1.1: FPGA Hardware Block design

The block design architecture depicted in Figure 4.1.1 represents the complex hardware design for the telemetry web server. This architecture is primarily built around the Xilinx MicroBlaze processor, a versatile and customizable soft-core processor designed for Xilinx Field-Programmable Gate Arrays (FPGAs).

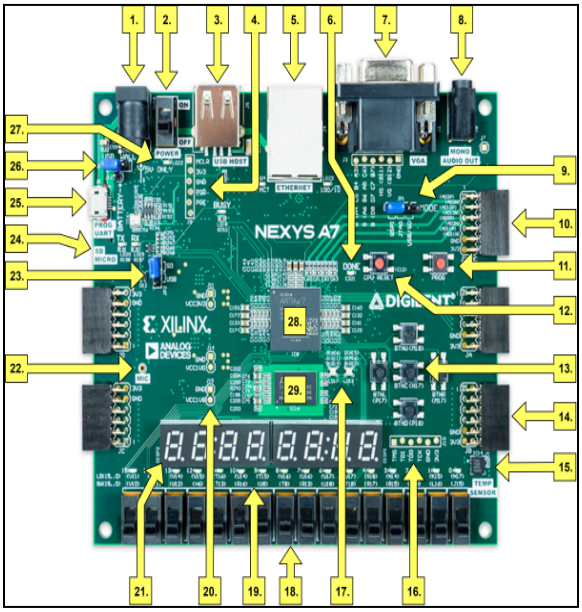
At the heart of the design is the MicroBlaze, which interfaces with several key components via the AXI interconnect—a system bus designed by ARM to connect different parts of a chip or system efficiently. The AXI interconnect facilitates communication between the MicroBlaze and various peripherals and memory blocks, including the DDR2 memory module, represented by the mig_7series component, and local memory resources (DLMB and ILMB for data and instruction respectively). These memory modules are essential for the storage and retrieval of the code and data necessary for the operation of the telemetry web server.

Peripheral components, such as timers (axi_timer_0 and axi_timer_1), an Ethernet interface (axi_ethernetlite_0), and UART interface (axi_uartlite_0), are integrated into the design to provide a range of functionalities. The Ethernet interface enables network connectivity, essential for the telemetry data's transmission and reception to and from external computers. In contrast, the UART interface facilitates serial communication, possibly with other onboard hardware or for console management. The timers may serve to manage time-driven events within the system, crucial for tasks that require precise timing, such as sampling data at specific intervals.

The design also includes a Debug Module (MDM), a Clocking Wizard for clock management, and a Processor System Reset block, indicating a robust approach to system stability and error handling. The inclusion of a Concat block hints at the need for combining multiple signals into a single bus, which is often used for effective signal management and routing within FPGAs.

Overall, the block design architecture outlines a sophisticated system intended to provide real-time telemetry data services, with a strong emphasis on modularity, expandability, and robust control mechanisms. The use of MicroBlaze as a central processing unit allows for flexible software development and deployment, essential for a telemetry web server operating within the demanding environment of satellite hardware testing and verification.

Graphical Representation:



Callout	Component Description	Callout	Component Description
1	Power jack	16	JTAG port for (optional) external cable
2	Power switch	17	Tri-color (RGB) LEDs
3	USB host connector	18	Slide switches (16)
4	PIC24 programming port (factory use)	19	LEDs (16)
5	Ethernet connector	20	Power supply test point(s)
6	FPGA programming done LED	21	Eight digit 7-seg display
7	VGA connector	22	Microphone
8	Audio connector	23	External configuration jumper (SD / USB)
9	Programming mode jumper	24	MicroSD card slot
10	Analog signal Pmod port (XADC)	25	Shared UART/ JTAG USB port
11	FPGA configuration reset button	26	Power select jumper and battery header
12	CPU reset button (for soft cores)	27	Power-good LED
13	Five pushbuttons	28	Xilinx Artix-7 FPGA
14	Pmod port(s)	29	DDR2 memory
15	Temperature sensor		

Power input requirements:
Powered from USB or any 4.5V-5.5V external power source

Connector specifications:
-USB-JTAG programming circuitry
-USB-UART bridge
-USB HID Host for mice, keyboards and memory sticks
-10/100 Ethernet PHY
-Pmod connector for XADC signals
-Four Pmod connectors providing 32 total FPGA I/O
-12-bit VGA output
-PWM audio output
-PDM microphone

Memory:
-128MiB DDR2
-Serial Flash
-microSD card slot

Processor:
Artix-7 FPGA:
-15,850 Programmable logic slices, each with four 6-input LUTs and 8 flip-flops (*8,150 slices)
-4,860 Kbits of fast block RAM (*2,700 Kbits)
-Six clock management tiles, each with phase-locked loop (PLL)
-240 DSP slices (*120 DSPs)
-Internal clock speeds exceeding 450 MHz
-Dual-channel, 1 MSPS internal analog-digital converter (XADC)

User interfaces:
-16 Switches
-16 LEDs
-Two RGB LEDs
-Two 4-digit 7-segment displays

Additional Information:
For more detailed specifications, pin assignments, and usage guidelines for the Digilent Nexys A7 board, please refer to the official board documentation and user manual provided by Digilent.

4.2 Software Detailed Design

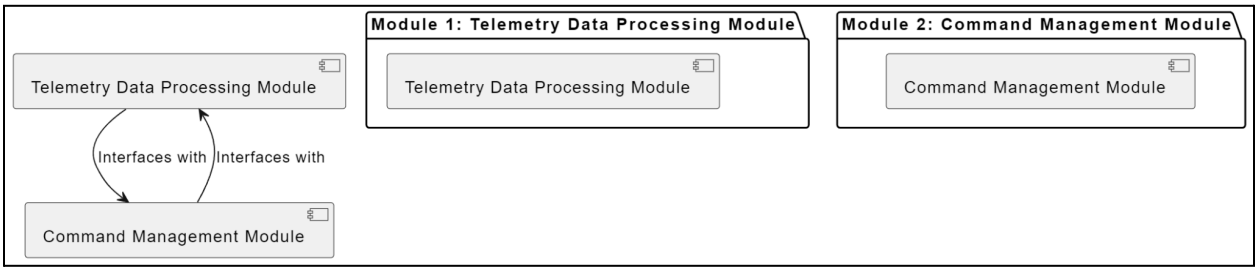


Figure 4.2.1: depicts the Telemetry Data Processing Module, orchestrating the collection and processing of AXI bus data, supported by advanced algorithms for web server presentation. Simultaneously, the Command Management Module, featured in the same figure, validates and executes authorized user commands, implementing stringent security measures.

This section provides detailed module designs for the software components of the Ingenion Telemetry Web Server project. Each module's design includes a narrative description, its function, conditions of use, processing logic, interfaces, security requirements, algorithms used, data structures, graphical representations of processing and control flow, and data entry/output graphics where applicable. Industry-standard module specification practices are followed. Additional diagrams will be included as project scope is refined.

Module 1 Telemetry Data Processing Module:

Narrative Description:

Function: The Telemetry Data Processing Module manages the collection and processing of telemetry data from various sources.
Conditions of Use: Scheduled for execution at specified intervals or upon request.
Processing Logic: Collects data from the AXI bus and peripheral devices, processes it for display, and stores it in appropriate formats.
Interfaces: Interfaces with the AXI bus, peripheral devices, and the web server for data presentation.
Security Requirements: Implements access controls to ensure only authorized users can initiate data processing.
Algorithms: Utilizes data sampling and signal processing algorithms for telemetry data processing.
Data Elements and Structures:

Input: Telemetry Data Streams
Output: Processed Telemetry Data, Graphical Representations
Graphical Representation:

Module 2: Command Management Module

Narrative Description:

Function: The Command Management Module is responsible for receiving and processing commands from authorized users and forwarding them to the appropriate subsystems or components for execution.
Conditions of Use: Accessed when authorized users need to send commands to control or configure the system.
Processing Logic: Validates user authorization, interprets received commands, and routes them to the relevant subsystems or modules for execution.
Interfaces: Interfaces with user input from the web interface, validates command syntax, and communicates with subsystems via appropriate interfaces.
Security Requirements: Implements strict access controls to ensure that only authorized users can send commands. Additionally, enforces command validation to prevent unauthorized or malicious commands.
Algorithms: Utilizes algorithms for command parsing, syntax checking, and routing.
Data Elements and Structures:

Input: User-Generated Commands
Output: Command Execution Status, Responses to User

These detailed module designs provide a comprehensive view of the system's functionality, logic, data flow, and user interfaces, ensuring a clear understanding of each module's role within the Ingenion Telemetry Webserver project.

4.3 Internal Communications Detailed Design

This section will outline the detailed design of internal communications within the enhanced Total Verification System (TVS) project. Internal communications are essential for the exchange of information, commands, and data among system components.

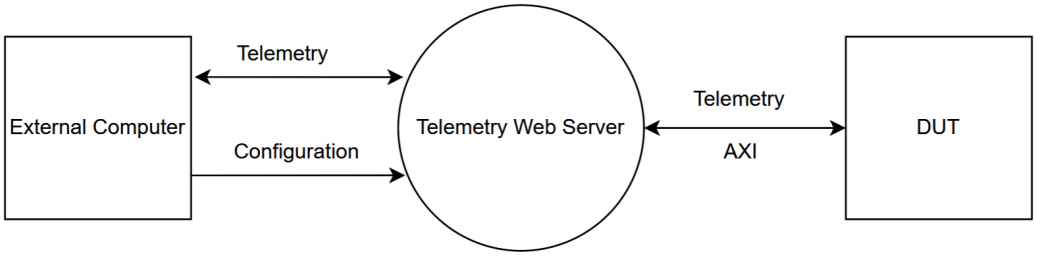


Figure 4.3.1: Level 0 Data Flow Diagram

As seen in Figure 4.3.1 the communications crucial to our design will consist of Telemetry being sent and received by the DUT, the web server itself, and an external computer. The External computer will additionally be able to send settings configurations to the Telemetry Web Server.

The internal communication network for the TVS project will consist of the following components:

Web Server: The Digilent Nexys A7 board, with FreeRTOS running on a MicroBlaze CPU programmed onto its Artix 7 FPGA, acting as a web server, will serve telemetry data to connected clients.

External Computer: A single external computer or device will act as the client, connecting to the web server, via an ethernet connection, to access telemetry data.

For the web server's internal communication, no specific bus timing requirements or bus control mechanisms are necessary. Communication will occur over Ethernet using TCP/IP, which automatically handles timing and packet control.

Data exchange between the DUT and the Nexys A7 will primarily use the standard data formats for the telemetry data depending on the connection, such as UART, i2c, SPI, etc., and then will be transmitted to the web server in a structured format such as JSON or XML for ease of parsing and display.

5 EXTERNAL INTERFACES

5.1 Interface Architecture

This section will provide a comprehensive description of the interface architecture for the Ingenion Telemetry Webserver project. This includes detailing the interfaces between the system being developed and other systems, as well as the architecture and protocols employed. A diagram will be provided to illustrate the communication paths between this system and each of the other systems, aligning with the context diagrams in Section 1.2.1. If necessary, subsections will be used to address specific interfaces being implemented.

The interface architecture of the Ingenion Telemetry Webserver project encompasses various communication pathways and protocols that enable interaction with external systems and components. Key elements of the interface architecture include:

Ethernet Communication: The primary means of communication between the Ingenion Telemetry Webserver and external systems is through Ethernet connections. This enables data transfer and command exchange between the web server system and external computers.

5.2 Interface Detailed Design

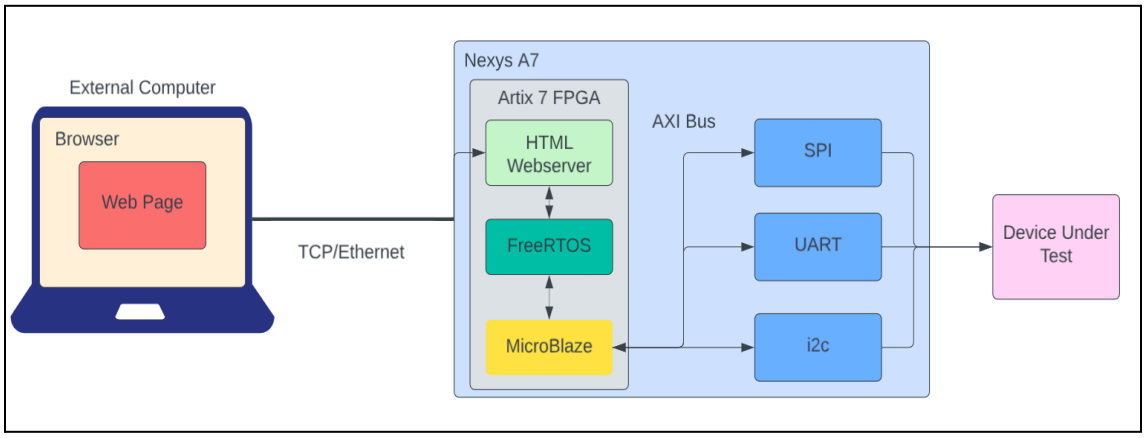


Figure 5.2.1 System Connectivity Overview: This diagram provides a representation of the system's architecture, highlighting the bidirectional data flow between the DUT, Nexys A7 containing the various components, and the external computer.

This section will provide detailed design specifications for each interface that facilitates information exchange between the Ingenion Telemetry Webserver and external systems. This includes data format requirements, hand-shaking protocols, error reporting formats, graphical representations of data flow, and query and response descriptions. Each interface will be addressed separately, ensuring clarity and precision in interface design.

Data Format Requirements:
The data exchanged over Ethernet will adhere to standard data formats such as JSON (JavaScript Object Notation) for ease of parsing and compatibility with web technologies. Data may need to be converted into JSON format before transmission or processed upon reception.

Hand-shaking Protocols:
The hand-shaking between the web server and external systems will involve a request-response model. Handshake messages will include TCP requests and responses. The format will follow TCP protocol standards, including headers, status codes, and payload data. Timing for message exchange will be within specified response time limits to ensure timely communication.

Error Reporting:
Error reports will be communicated using standard TCP error status codes (e.g., 4xx or 5xx) in response messages. Error details will be included in the response body, allowing for proper error handling and logging. Critical errors may trigger operator notifications through alarm flags.

Graphical Representation:
A diagram (not shown here) will illustrate the connectivity between the web server and external systems, showing the direction of data flow. Arrows will depict data transmission pathways and the systems involved.

Query and Response:
Queries sent to the web server will be structured as TCP GET requests with specific endpoints and query parameters. Responses will be provided in JSON format, containing requested data or acknowledgments.

Data Format Requirements:
Communication with the Digilent Nexys A7 development board will utilize specific data formats compatible with FPGA programming and control. Data may need to be converted into binary or other formats suitable for FPGA interfacing.

Hand-shaking Protocols:
The hand-shaking with the Digilent Nexys A7 will involve predefined control commands and acknowledgment signals. Timing for handshakes will adhere to FPGA interface specifications.

Error Reporting:
Errors related to FPGA interfacing will be logged and may trigger system notifications. Detailed error reports will be retained in system logs for analysis and debugging.

Graphical Representation:
A separate diagram will illustrate the connectivity between the web server and the Digilent Nexys A7 development board, indicating the flow of commands and responses.

Query and Response:
Queries to the Digilent Nexys A7 will consist of control commands sent over dedicated communication channels. Responses will include acknowledgments or status updates.

For each interface, the detailed design specifications ensure that data exchange is accurately formatted, transmitted, and received and that hand-shaking, error handling, and communication protocols are well-defined and compliant with the respective systems being interfaced. Any formal Interface Control Documents (ICDs) relevant to these interfaces will be referenced as needed to maintain consistency and alignment with established standards.

6 SYSTEM INTEGRITY CONTROLS

Sensitive systems, particularly those handling information that, if compromised, could affect State programs or individual privacy, require robust integrity controls. These controls are essential for maintaining the confidentiality, availability, and accuracy of critical data. The following outlines the minimum levels of control and specifications that should be implemented to ensure the integrity of the Ingenion Telemetry Webserver system.

Control and Reporting: Develop audit procedures to control, report, and retain audit information. This includes tracking user activities, system events, and changes to critical data.
Retention Period: Define a retention period for audit logs and reports, ensuring that historical

data is retained for compliance, analysis, and investigation purposes.
Application Audit Trails:

Dynamic Auditing: Implement dynamic auditing mechanisms that record retrieval access to designated critical data. Audit trails should capture details such as user identity, timestamp, accessed data, and actions performed.
Standard Tables for Data Validation:

Data Validation: Utilize standard reference tables and validation rules for data fields. These tables should be used to validate data entered or received by the system to ensure accuracy and consistency.
Verification Processes:

Data Modification Controls: Establish robust verification processes for additions, deletions, or updates of critical data. Such processes should include validation checks, approval workflows, and logging of changes.
Audit Information Identification:

User Identification: Ensure that all audit information is associated with the user's identification, allowing traceability of actions to specific individuals.

Network Terminal Identification: Record the network terminal or device used for the actions performed in the audit trail.

Date and Time Stamps: Include date and time stamps in audit logs to accurately track when events occurred.

Data Accessed or Changed: Document the specific data accessed or changed during user actions to provide a complete audit trail.