

**PONTIFICAL CATHOLIC UNIVERSITY OF RIO GRANDE DO SUL
SCHOOL OF TECHNOLOGY
COMPUTER ENGINEERING**

**DIGITAL SYSTEMS
INSTRUMENTATION APPLIED
TO THE DEBUGGING OF
DIGITAL INTERFACES IN THE
SOC-WIMED**

**BERNARDO FARINON
DIOGO OGOCHI**

Bachelor Thesis submitted to the Pontifical
Catholic University of Rio Grande do Sul
in partial fulfillment of the requirements
for the degree of Bachelor in Computer
Engineering.

Advisor: Prof. Fernando Gehm Moraes

**Porto Alegre
2026**

ACKNOWLEDGMENTS

We would like to express our sincere gratitude to everyone who contributed, directly or indirectly, to the completion of this work.

We are especially grateful to our advisors and professors for their guidance, knowledge, encouragement, and continuous support throughout this project. Their expertise and valuable feedback were essential to the development of this research.

We also thank our colleagues for their collaboration, valuable discussions, and support throughout the development of this work.

Finally, we express our heartfelt appreciation to our families and friends for their constant encouragement, patience, and unwavering support throughout our academic journey.

INSTRUMENTAÇÃO DE SISTEMAS DIGITAIS APLICADA À DEPURAÇÃO DE INTERFACES DIGITAIS DO SOC-WIMED

RESUMO

A depuração de sistemas e interfaces digitais requer métodos capazes de correlacionar sinais elétricos e eventos temporais. Alguns métodos de validação, como simulação, permitem verificar módulos isolados e identificar falhas lógicas durante o desenvolvimento. Entretanto, a execução em hardware real pode revelar problemas de integração, temporização e comunicação que não aparecem em ambiente simulado. Em sistemas em chip (SoC) implementados em FPGA ou ASIC, essa tarefa torna-se especialmente relevante, pois falhas de integração podem resultar em protocolos de comunicação não funcionais, erros de temporização, configuração incorreta de periféricos e comportamento inesperado do sistema. Nesses cenários, a utilização de ferramentas de instrumentação torna-se indispensável para a validação do sistema, permitindo analisar sinais digitais, protocolos de comunicação e identificar inconsistências entre o comportamento esperado e os resultados obtidos experimentalmente. Equipamentos de bancada, como analisadores lógicos e osciloscópios, possibilitam a captura de sinais e a análise temporal, auxiliando no processo de depuração e na identificação de falhas de integração e de comunicação. Este trabalho apresenta uma contribuição para a depuração do SoC-WiMed, com foco na construção de um ambiente de observação e análise, utilizando o analisador lógico Agilent 1672G, e propõe estudos de caso envolvendo os protocolos SPI e UART. Adicionalmente, descreve-se uma infraestrutura baseada em scripts em Python para organização de dados experimentais. Como resultado, o trabalho produziu uma metodologia reproduzível para a instrumentação de sinais digitais e a análise funcional.

Palavras-Chave: Depuração. Analisador lógico. Interfaces digitais. Instrumentação. Sistemas em chip (SoC).

DIGITAL SYSTEMS INSTRUMENTATION APPLIED TO THE DEBUGGING OF DIGITAL INTERFACES IN THE SOC-WIMED

ABSTRACT

Debugging digital systems and interfaces requires methods capable of correlating electrical signals and temporal events. Validation methods, such as simulation, allow isolated modules to be verified and logical faults to be identified during development. However, execution on real hardware may reveal integration, timing, and communication issues that do not appear in simulated environments. In systems-on-chip (SoC) implemented on FPGA or ASIC platforms, this task becomes especially relevant, since integration failures may result in non-functional communication protocols, timing errors, incorrect peripheral configurations, and unexpected system behavior. In these scenarios, instrumentation tools are essential for system validation, enabling analysis of digital signals and communication protocols and the identification of inconsistencies between expected and observed behavior. Instruments such as logic analyzers and oscilloscopes enable signal acquisition and temporal analysis, assisting in debugging and identifying integration and communication failures. This work contributes to debugging the SoC-WiMed by developing an observation and analysis environment using the Agilent 1672G logic analyzer, including case studies of the SPI and UART protocols. In addition, a Python-based infrastructure is described to organize experimental data. As a result, the work produced a reproducible methodology for digital signal instrumentation and functional analysis.

Keywords: Debugging. Logic analyzer. Digital interfaces. Instrumentation. Systems-on-chip (SoC).

LIST OF FIGURES

2.1	Configuration menu representation and physical setup.	14
2.2	Menu Field options in Analyzer mode.	15
2.3	Format Machine interface showing bus naming and bit assignment using asterisks (*) and dots (.).	17
2.4	Physical probe connections from the analyzer's Pod 1 to the FPGA prototyping board interfaces.	18
2.5	Trigger Menu screen highlighting the term definition cell matrix.	20
2.6	Sequence Level pop-up window configured with custom terms for experimental validation.	21
2.7	Conceptual workflow highlighting the criteria for selecting between Timing and State analysis modes.	22
2.8	Analyzer Configuration menu illustrating Machine 1 mapped to the Timing type.	24
2.9	Format Machine menu illustrating a renamed label and the active 250 MHz Timing Acquisition Mode status bar.	25
2.10	Channel assignment matrix interface showing active bit mapping using asterisks (*).	26
2.11	Trigger configuration menu showing a defined term and its assigned hexadecimal condition.	27
2.12	Timing Sequence Level pop-up window configured to trigger on term <i>a</i> with a minimum duration filter of 16 ns.	29
2.13	Waveform menu interface showing digital signal transitions, time markers, and hex bus decode lines at the marker position.	30
2.14	Analyzer Configuration menu illustrating Machine 1 mapped to the State type and displaying channel activity indicators along the J-clock channel.	31
2.15	Format Machine menu in State Mode illustrating the customized SPI label and the active 150 MHz master clock status indicator assigned to the falling edge of the J-clock line.	32
2.16	Trigger configuration menu in State Mode showing a customized term defined under the active synchronous bus label.	33
2.17	Sequence Level 1 pop-up menu configuration screen illustrating the assignment of term <i>b</i> as the primary state sequence condition.	34

2.18	Sequence Level 2 pop-up menu configuration screen illustrating the command to store all subsequent synchronous states once the initial frame boundary is triggered.	35
2.19	State Listing display interface demonstrating the indexed table format centered around the zero-point trigger marker.	37
2.20	Format menu configuration screen illustrating the creation of the custom PATGEN label and deactivation of background groups.	38
2.21	Channel assignment matrix highlighting the specific four-pin asterisk (*) selection on Pod A1 mapped to the PATGEN bus.	39
2.22	Pattern Generator Format Menu interface showing the customized PATGEN label mapped to the lower bits of Pod B4 alongside an internal clock source configuration.	40
2.23	Pattern Generator Sequence editor showing the completed multi-line vector program compiled in binary format.	41
2.24	Timing analyzer waveform view displaying the captured multi-channel walking ones pattern generated by the independent output module.	42
2.25	System menu interface demonstrating the disk operations and storage medium selection options.	43
2.26	Configuration storage workspace configured for loading and storing instrument settings via the target file fields.	44
2.27	Print Disk submenu overlay configured to export complete trace spreadsheets in ASCII format.	45
2.28	LAN configuration sub-panel displaying the network parameter assignments and connection pathways.	46
2.29	Terminal environment showing the active host-to-instrument FTP communication session and data format initialization.	47
2.30	SPI transmission captured using the Agilent 1672G. The MOSI, SCLK, and CS signals can be observed, with a separate CS activation preceding each transmitted byte.	48
2.31	UART transmission captured in State Analysis mode.	49
2.32	Pattern Generator sequence programmed in the Agilent 1672G. The displayed vector sequence was used to generate a predefined digital waveform.	51
2.33	Waveform captured by the Agilent 1672G after connecting the Pattern Generator output to a logic analyzer input channel.	51
2.34	Workflow of the Python support software.	52
2.35	SPI waveform reconstructed by the Python support software, including hexadecimal and ASCII decoding of the transmitted data.	54

3.1	SoC-WiMed block diagram. In the AXI4-Lite bus, green boxes represent master interfaces, and yellow boxes represent slave interfaces.	56
4.1	Overview of the UART transmission generated by the SoC-WiMed platform and captured using the Agilent 1672G logic analyzer.	59
4.2	Detailed view of a UART frame transmitted by the SoC-WiMed platform, showing the start bit, data field, and stop bit used during character transmission.....	59
4.3	UART waveform after protocol decoding, showing the reconstructed data values associated with the captured transmission.	60

CONTENTS

1	INTRODUCTION	9
1.1	TEST EQUIPMENT	10
1.2	OBJECTIVES	12
1.3	DOCUMENT ORGANIZATION	12
2	LOGIC ANALYZER AGILENT 1672G	13
2.1	OPERATION PROCEDURES	13
2.1.1	ANALYZER INTERFACE	14
2.1.2	TRIGGER CONFIGURATION	19
2.1.3	ACQUISITION MODES	21
2.1.4	PATTERN GENERATOR	37
2.1.5	FILE MANAGEMENT	42
2.1.6	LAN INTERFACE AND FTP TRANSFER	45
2.2	CASE STUDIES	47
2.2.1	SPI PROTOCOL DEBUGGING	47
2.2.2	UART PROTOCOL DEBUGGING	49
2.2.3	DIGITAL SIGNAL INJECTION	50
2.2.4	PYTHON SUPPORT SOFTWARE	52
3	SOC-WIMED PLATFORM	55
3.1	SOC-WIMED OVERVIEW	55
4	DEBUGGING THE SOC-WIMED	57
4.1	EXPERIMENTAL SETUP	57
4.2	UART INTERFACE VALIDATION	58
5	CONCLUSION AND FUTURE WORK	62

1. INTRODUCTION

The development of modern digital systems combines various hardware and software components within a single platform. In System-on-Chip (SoC) architectures, processors, memories, buses, and communication peripherals must work together, making system validation and debugging more complex.

In the SoC-Wimed project, this integration includes a RISC-V-based processor, communication peripherals, internal memory, and modules supporting the Zephyr RTOS. The work uses FPGA prototyping and experimental validation before ASIC implementation. Issues found during testing can affect future integrated circuit implementations.

Simulation tools are commonly used in digital system development to validate isolated modules and check hardware behavior. However, running the system on actual hardware can uncover problems that do not show up in simulations, such as timing violations, synchronization errors, bus conflicts, and issues with communication interfaces. These challenges become harder to spot as the number of integrated modules increases.

In this situation, digital instrumentation becomes an important part of the experimental validation process. During hardware execution, instrumentation helps increase system observability by enabling direct monitoring of signals, interfaces, and timing behavior from the prototype. It also improves controllability, since specific test conditions can be created through reset signals, external clock sources, or controlled data transmission. As formalized by Bushnell and Agrawal in Chapter 6 of their work [[Bushnell and Agrawal, 2000](#)], these resources are essential to simplify the analysis of communication and synchronization problems by overcoming the inherent difficulty of observing and controlling internal circuit states.

In FPGA-based platforms, it is also possible to use internal debugging tools such as Integrated Logic Analyzers (ILA) [[AMD/Xilinx, 2023](#)]. Another possibility is the use of laboratory instruments connected directly to the prototype. The logic analyzer captures digital signals over time, allowing clock, control, and data signals to be analyzed together during system execution. The multimeter is mainly used for static electrical measurements. The oscilloscope allows voltage variations and timing behavior to be observed. The signal generator can provide external signals and generate electrical waveforms with specific characteristics for testing interfaces and system inputs.

This work contributes to the experimental debugging of the SoC-Wimed by using the Agilent 1672G logic analyzer on an FPGA-based platform. It includes case studies of the system's digital interfaces and employs Python scripts to organize and analyze experimental data. The goal is not to replace simulation but to add another step to the SoC validation process by observing signals in the physical prototype.

1.1 Test Equipment

I **Multimeter.** The multimeter is used to measure voltage, current, resistance, and continuity. Before acquiring signals with the logic analyzer, it is important to verify that the FPGA board and external modules operate at the expected voltage levels and have proper ground references. The instrument also allows continuity testing and resistance measurements, helping identify connection problems involving cables, connectors, probes, and test points used during the experiments.

On FPGA platforms, simple connection issues may prevent communication interfaces from operating correctly. Missing ground connections, faulty probes, or incorrectly connected signals may introduce failures that make debugging unfeasible. For this reason, basic electrical verification procedures are performed before signal acquisition procedures.

The multimeter is limited to static electrical measurements and cannot properly represent fast digital transitions or communication activity between interfaces. Analyses involving timing behavior, synchronization, or protocol validation require instruments capable of observing signals over time, such as oscilloscopes and logic analyzers. In this work, the multimeter is used as a support instrument during the preparation and verification of the experimental setup.

II **Logic Analyzer.** The logic analyzer [[Rippel, 2024](#); [Tektronix, Inc., 2009](#)] is the main instrument used in this work. It is designed to capture and analyze digital signals by interpreting voltage levels based on a configurable threshold, enabling visualization of logical transitions between binary states. Unlike an oscilloscope, which provides a detailed analog representation of electrical signals, the logic analyzer focuses on the logical interpretation of multiple digital channels simultaneously, making it suitable for debugging digital systems, buses, finite state machines, and serial communication interfaces.

This capability allows the validation of communication protocols such as UART and SPI. In UART communication, the analyzer can be used to verify transmitted data and the protocol structure, including start and stop bits. For SPI communication, signals such as SCLK, MOSI, MISO, and CS can be observed simultaneously, enabling the verification of synchronization, clock polarity, and protocol behavior.

Another important feature of the logic analyzer is the trigger mechanism. Trigger conditions allow signal acquisition to start only when a specific event occurs, improving the observation of relevant transactions. For example, in UART communication, the transmission line remains high while idle and transitions to low during the start bit. A falling-edge trigger configured on the TX signal allows the capture process to begin at

the start of a transmission. More advanced trigger conditions can also be configured to detect specific sequences or complex signal events.

- III **Signal Generator.** The signal generator is used to produce external stimuli for the system under test. Although the instrument is commonly associated with analog signal generation, it can also generate square waves, clock pulses, and simple digital sequences, making it useful for debugging digital systems. As a result, the signal generator can be used to test digital inputs and emulate communication or control signals between peripherals.

Within the context of the SoC-Wimed platform, a signal generator could be used as a complementary debugging instrument by emulating communication protocols such as UART or SPI. Instead of relying on an external peripheral, predefined signal sequences could be generated to stimulate the communication interface under controlled conditions, allowing protocol timing, data reception, and error handling to be evaluated.

Although this approach was not employed during the experimental validation presented in this work, it represents a useful strategy for future debugging activities involving communication interfaces.

The signal generator does not replace real peripherals, but it allows isolated and controlled tests to be performed. When communication failures occur, the external peripheral is not always the source of the problem. By isolating variables and generating signals independently, it becomes easier to reduce environmental interference and identify failure sources during debugging.

- IV **Oscilloscope.** The oscilloscope is used to observe electrical signals over time, allowing analysis of characteristics such as amplitude, frequency, stability, and waveform shape equipment [Tektronix, 2011]. Unlike the logic analyzer, which interprets digital signals only as logical states, the oscilloscope displays the complete analog behavior of the electrical signal. This makes it possible to identify noise, oscillations, instability, and signal integrity problems that may not be visible through logical analysis alone.

In this work, the oscilloscope is primarily used as a support instrument for the initial validation of signals and the verification of voltage levels in the analyzed interfaces. Its use also helps to ensure safe operating conditions for the FPGA board and connected laboratory equipment during the experiments.

Although the oscilloscope is useful for electrical and temporal signal analysis, it does not replace the logic analyzer for debugging complex digital protocols. The simultaneous observation of multiple channels and the logical interpretation of communication protocols make the logic analyzer more suitable for the functional validation procedures used in this work.

1.2 Objectives

This work aims to contribute to the experimental validation and debugging of the SoC-Wimed. For this purpose, a digital instrumentation methodology focused on analyzing UART and SPI serial interfaces is applied and documented, using the Agilent 1672G logic analyzer as the primary instrumentation tool.

To reach the strategic objective, the following specific goals are set:

I Study the Agilent 1672G logic analyzer.

Study the operation of the Agilent 1672G logic analyzer, including timing and state analysis modes, trigger configuration, and digital channel setup.

II Develop UART and SPI debugging case studies.

Develop case studies for UART and SPI debugging, including analyzer configuration, expected data patterns, and interpretation of captured signals.

III Develop support scripts.

Develop Python scripts to assist in the organization and analysis of captured experimental data.

IV Apply the debugging methodology to the SoC-Wimed.

Apply the proposed debugging methodology to the SoC-Wimed platform, with an emphasis on validating communication between the computer and the FPGA through the UART interface.

V Produce technical documentation.

Produce technical and reproducible documentation, including signal tables, screenshots, connection diagrams, and analysis of experimental results.

1.3 Document Organization

This work is organized as follows. Chapter 1 presents the motivation, objectives, and scope of this work. Chapter 2 introduces the Agilent 1672G logic analyzer and discusses the instrumentation concepts required for the experimental activities. Chapter 3 presents an overview of the SoC-Wimed platform, including its main architectural components and communication interfaces relevant to the debugging process. Chapter 4 presents the experimental validation of the SoC-Wimed UART interface using the Agilent 1672G logic analyzer and the Python support software developed during this work. Chapter 5 concludes this work and discusses possible directions for future developments.

2. LOGIC ANALYZER AGILENT 1672G

The Agilent 1672G is a logic analyzer designed to acquire and analyze digital signals in embedded systems, processors, and communication interfaces [Agilent Technologies, 2002, 2003, 2005]. The instrument provides 64 independent data channels and 4 dedicated clock/qualifier channels, enabling simultaneous observation of multiple signals during system execution. This capability is particularly useful when debugging communication protocols, where data, control, and synchronization signals must be analyzed together.

The instrument supports two main acquisition modes: Timing Analysis and State Analysis. In Timing Analysis mode, signals are sampled asynchronously, enabling observation of transitions, pulse widths, glitches, and timing relationships between signals. In this configuration, the Agilent 1672G can operate at sampling rates of up to 250 MHz with all 64 channels or up to 500 MHz in half-channel mode. In State Analysis mode, signal acquisition is synchronized with the clock of the circuit under test, supporting frequencies of up to 150 MHz. This mode is particularly suitable for observing synchronous buses and monitoring the execution flow of digital systems.

Another important feature is its memory capacity and trigger sequencing capabilities. The instrument provides up to 64K samples per channel in its standard configuration and supports memory expansion options and complex trigger conditions. The trigger system allows data acquisition to begin only when predefined events occur, reducing the amount of irrelevant data collected and simplifying debugging. Although triggers were mainly used in this work to capture UART and SPI communication events, the analyzer can also be configured for more advanced applications, such as monitoring finite-state machine (FSM) transitions, validating control sequences, and analyzing handshake events between digital modules.

The Agilent 1672G also supports configurable threshold-voltage settings that convert measured voltage levels into digital logic states. This feature allows the analyzer to be adapted to different logic families and operating voltages, increasing the reliability of signal interpretation and reducing errors caused by voltage-level incompatibilities between the analyzer and the circuit under test.

2.1 OPERATION PROCEDURES

Below, the primary operational characteristics of the analyzer are discussed, which were necessary for conducting the experiments within the scope of this work. Knowing the operational characteristics of the analyzer is necessary for a correct interpretation of the acquisition and debugging procedures, which will be discussed later in the case stud-

ies section. The operational characteristics of the device were acquired by studying the manufacturer's official technical manuals [Agilent Technologies, 2002, 2003] and through practical experimentation with the analyzer in the laboratory. Furthermore, to demonstrate the procedures performed and to aid the understanding of the interface and settings chosen, screenshots from the analyzer interface are provided below.

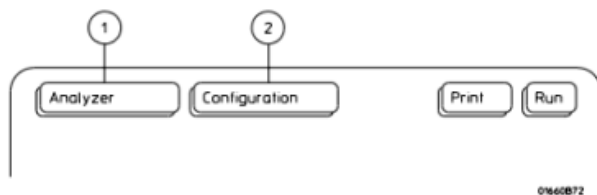
2.1.1 Analyzer Interface

2.1.1.1 Initial Screen and Mode Field

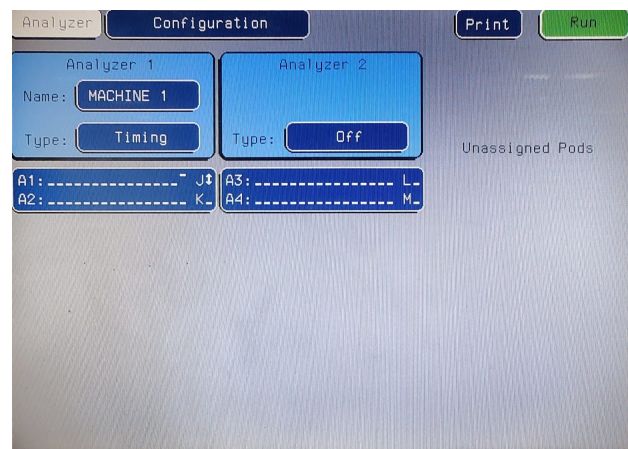
After powering up the analyzer, once the system has finished loading, the analyzer menu appears as the first screen. This menu is the main means of interaction between the user and the instrument. It centralizes all analyzer functions, allowing the user to configure acquisitions, visualize signals, and access the equipment tools.

Menu can be identified by two fields in the upper left corner. The leftmost field displays “Analyzer”. This field is referred to as the *Mode Field* because it controls which sets of menus are accessible. The second field, to its right, provides access to menus in the selected mode and is therefore called the *Menu Field*. For example, in Analyzer mode, the analyzer menus are accessed through the Menu Field. Menus are referred to by the titles that appear in the Mode and Menu Fields.

Figure 2.1a shows the top of the first screen. The Mode Field, item 1, displays “Analyzer”. The Menu Field, item 2, displays “Configuration”. Since menus are identified by the titles in these two fields, this menu is referred to as the Analyzer Configuration menu. Figure 2.1b shows the interface and a real image of the analyzer.



(a) Menu format sketch.



(b) Physical menu configuration setup.

Figure 2.1: Configuration menu representation and physical setup.

2.1.1.2 Menu Field

Right next to the Mode Field is the *Menu Field*, a selector whose contents change according to the current context. While the Mode Field defines the overall operating state of the global workspace, determining whether the instrument operates in a system utility or measurement environment, the Menu Field updates the available options to match the selected mode. Users can access this area via either the rotary control knob or the front-panel navigation keys, allowing them to switch between different working windows. This hierarchical organization keeps the user interface uncluttered by hiding irrelevant functions and enables seamless transitions between channel allocation, trigger configuration, and raw data visualization.

To illustrate the interaction with the configuration interface, Figure 2.2 highlights the dropdown structure of the Menu Field when the instrument operates in Analyzer mode. In this configuration, the user can navigate through the different stages of the acquisition setup.

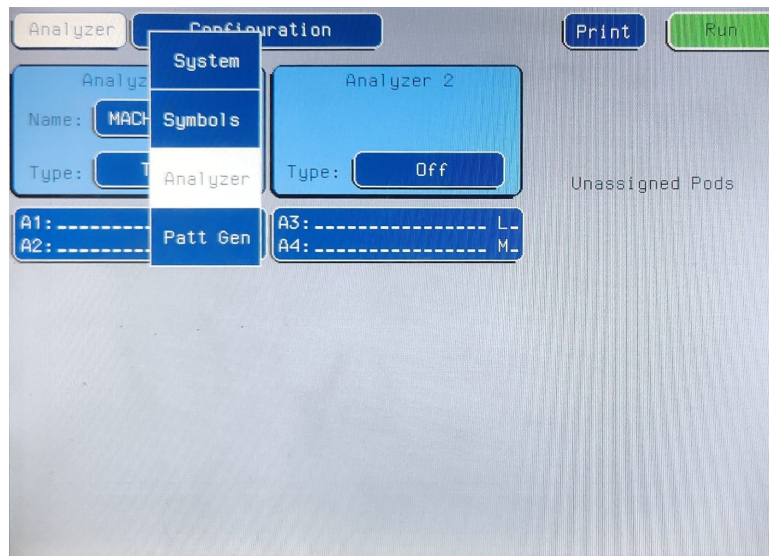


Figure 2.2: Menu Field options in Analyzer mode.

2.1.1.3 System Menus

Selecting “System” in the Mode Field activates the main Menu Field, which consists of five subfields. Each subfield provides access to system-wide configuration parameters and operational functions associated with the Agilent 1672G platform architecture:

- **Hard Disk:** Provides access to nonvolatile disk management functions, including directory navigation, file management, and storage-related configuration settings.
- **Flexible Disk:** Supports low-level operations involving external storage media, including formatting and file management through the external 3.5-inch disk drive.

- **External I/O:** Configures communication settings for the standard Agilent interfaces, including GP-IB, RS-232-C, and LAN. Functions range from interface addressing to printer configuration and external reset control.
- **Utilities:** Provides access to the real-time clock settings, firmware management functions, and display adjustments for improved CRT visualization.
- **Test:** Displays the modules currently installed in the system, reports software version information, and initiates internal hardware self-tests to verify proper channel and probe calibration.

2.1.1.4 Analyzer Menus

When the Mode Field is set to “Analyzer”, the instrument configures its internal subsystems for digital signal acquisition and analysis. The Menu Field then updates its contents to provide access to measurement-specific environments. Some of these menus are available in all analyzer configurations, whereas others depend on the selected acquisition mode, such as state, timing, or Systems Performance Analysis (SPA).

- **Configuration:** Available in all analyzer modes, this menu associates the physical probe pods with the current measurement session and defines the default analyzer operating configuration.
- **Format:** Configures custom bus labels, signal polarities, clock edge selection, and the electrical threshold characteristics of the connected probe pods.
- **Trigger:** Provides access to the multilevel logic sequencing engine, allowing the definition of complex trigger conditions based on logic patterns, signal transitions, and conditional events. These capabilities enable highly selective acquisition of bus activity by capturing data only when predefined conditions are satisfied.
- **Listing:** Displays captured transactions as a sequential state listing. This view is particularly useful for examining bus activity and tracing instruction execution in microprocessor-based systems.
- **Compare and Mixed Display:** Advanced analysis tools that either compare two active state listings against a stored reference dataset (*Compare*) or simultaneously display state listings and timing waveforms on a common time base (*Mixed Display*).
- **Waveform:** Displays the logic level transitions of captured signals as digital waveforms along a horizontal time axis. This view is primarily intended for asynchronous signal analysis and diagnostic work, including propagation delay measurements and the identification of digital glitches.
- **Chart and SPA:** High level analysis tools. The *Chart* view plots numerical bus values over time, while *SPA* (Systems Performance Analysis) provides advanced performance metrics and graphical analyses of the target system’s behavior.

2.1.1.5 Navigation Flow and Channel Labeling

The operation of the Agilent 1672G follows a hierarchical configuration workflow. Typically, the user first selects the operating domain via the *Mode Field*, then selects the desired workspace using the *Menu Field*, and finally configures the relevant parameters on the selected page. To improve signal readability during complex measurement sessions, the instrument allows channels to be organized into logical groups and assigned custom labels through the *Format* menu. These labels are automatically propagated to all analysis views, including the *Listing* and *Waveform* displays, ensuring a consistent representation of signals throughout the measurement process.

By default, newly created channel groups are assigned sequential names ranging from *Bus1* to *Bus126*. The user may rename any group by selecting its label field, invoking the *Modify Label* function, and entering an alphanumeric identifier of up to six characters using the front panel entry controls. Individual channels are then assigned to the selected bus through an interactive mapping interface. Using the navigation controls, the user selects a physical channel and toggles its assignment status. An asterisk (*) indicates that the channel is included in the selected logical group, whereas a dot (.) denotes an unassigned channel. Pressing the *Select* key changes the assignment state, and selecting *Done* confirms the configuration.

The assignment of physical channels to logical signal groups is performed entirely within the *Format* menu, the primary configuration environment for signal organization. As illustrated in Figure 2.3, users can define custom bus labels, such as protocol-specific groups for UART or SPI signals, and assign individual channels to each group.

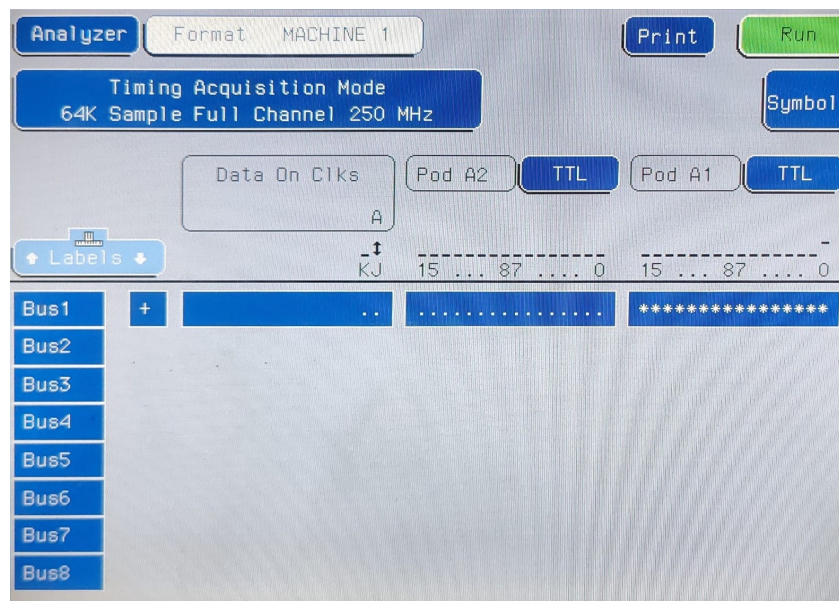


Figure 2.3: Format Machine interface showing bus naming and bit assignment using asterisks (*) and dots (·).

The graphical interface employs a simple notation to indicate channel membership. An asterisk (*) marks a channel that is currently assigned to the selected bus, while a dot (·) identifies a channel that is excluded from the group. Assignments can be modified by navigating to the desired bit position with the rotary knob and pressing the front panel *Select* key.

2.1.1.6 Target System Channel Mapping

Experimental validation of the SoC-Wimed platform required a dedicated channel-mapping scheme capable of simultaneously capturing the system's primary digital interfaces with the analyzer's *Pod 1*, as shown in Figure 2.4. To simplify laboratory wiring and optimize signal organization within the analyzer, the monitored signals were divided into distinct logical groups:

- **Asynchronous Domain (UART):** This group was used to capture processor diagnostic messages, including software-generated *print* outputs. Channel 0 was assigned to the UART transmission line (TX), while Channel J was configured as the timing reference signal required by the analyzer for signal acquisition.
- **Synchronous Communication Session (SPI):** For SPI measurements, a dedicated logical group labeled *SPI* was created. Channel 0 was assigned to the serial data line (MOSI or MISO, depending on the measurement configuration), Channel 1 was assigned to the Chip Select (CS) signal, and Channel J was configured as the SPI clock input.
- **Unassigned Channels:** All remaining channels within *Pod 1* that were not required for a given experiment were left unassigned. In the analyzer interface, these channels are represented by dots (·). Restricting channel assignments to only the required signals reduces configuration complexity and prevents unused inputs from occupying analyzer resources during acquisition.

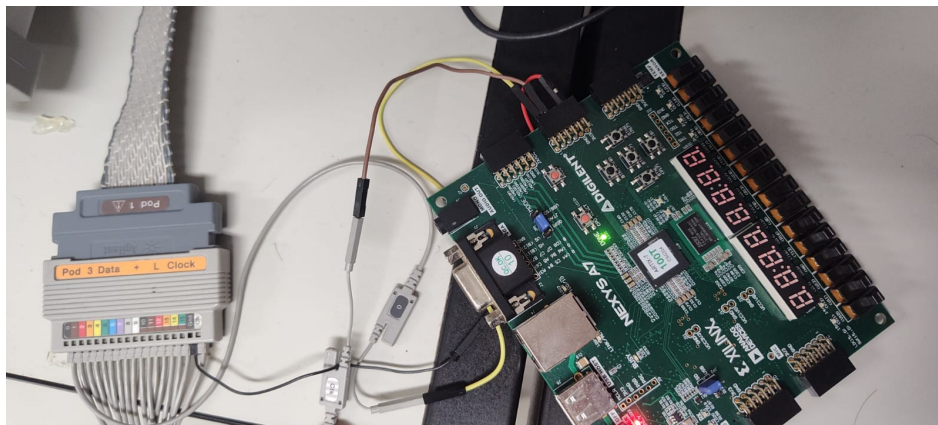


Figure 2.4: Physical probe connections from the analyzer's Pod 1 to the FPGA prototyping board interfaces.

2.1.2 Trigger Configuration

2.1.2.1 Trigger Basics

The trigger system serves as a hardware filtering mechanism, allowing the analyzer to isolate a specific digital event of interest from the surrounding background activity. Since the logic analyzer continuously records incoming signals into a circular memory buffer, the trigger condition defines the instant at which data acquisition is considered significant. This mechanism enables the analyzer to retain both pre-trigger and post-trigger information, providing context for the captured event.

The instrument includes predefined trigger templates that depend on the selected acquisition mode. In the default state analysis configuration, the trigger is set to *"While storing 'anystate' TRIGGER on 'a' occurring 1 time, Store 'anystate'"*. In the default timing analysis configuration, the trigger condition is *"TRIGGER on 'a' > 8 ns"*. Although these templates allow immediate operation, most measurement tasks require defining application-specific trigger conditions. This customization is achieved by assigning hardware resources to trigger terms and defining the corresponding signal patterns.

2.1.2.2 Trigger Menu

The Trigger menu becomes available when *Analyzer* is selected in the Mode Field and *Trigger* is selected in the Menu Field. Upon startup, the instrument configures Analyzer 1 as Machine 1 and automatically assigns the available trigger resources. By default, Analyzer 2 remains inactive.

Analyzer 2 can be enabled through the Trigger menu by selecting an entry from the *Terms* field and assigning the required hardware resources. Resource allocation is managed through the *Resource Term Assignment* menu, which allows trigger resources to be redistributed among the available trigger terms.

Trigger conditions are created by selecting a trigger term, such as *a*, and specifying a pattern for the corresponding signal label. The analyzer accepts binary values, hexadecimal values, or the symbol "X" to represent a don't care condition. Once the configuration is confirmed with the *Done* command, the pattern is added to the active trigger definition. This capability enables the analyzer to detect predefined signal combinations and initiate data acquisition when the specified conditions are satisfied.

Figure 2.5 presents the Trigger menu configured with example patterns assigned to trigger term *a*.

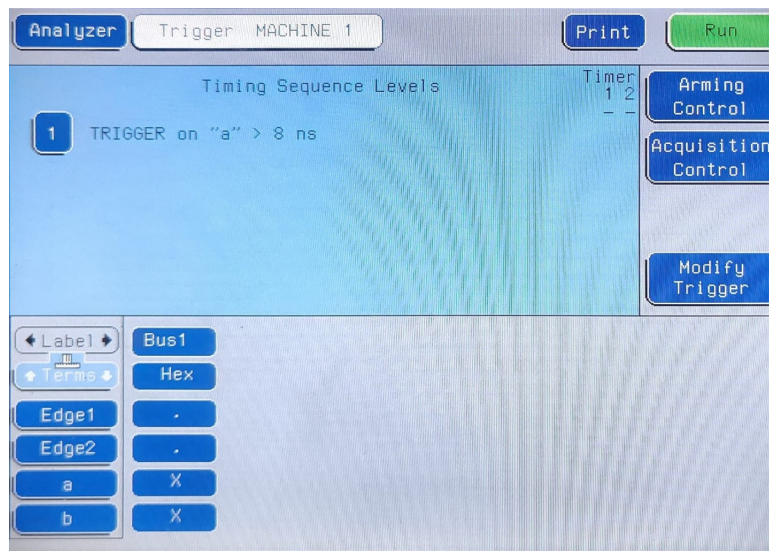


Figure 2.5: Trigger Menu screen highlighting the term definition cell matrix.

2.1.2.3 Trigger Sequence

Beyond simple edge and pattern detection, the Agilent 1672G provides a sequence-based trigger engine that enables defining multi-stage acquisition conditions. Sequential trigger configurations are accessed through the sequence level indicators located along the left side of the Trigger menu. Selecting a sequence level opens the *Sequence Level* window, which provides three primary functions: *Insert Level*, *Select New Function*, and *Delete Level*. These features allow the construction of conditional and event-driven trigger sequences.

- **Sequence Level Addition:** A new sequence stage can be inserted by selecting *Insert Level* and specifying whether the stage should be placed *Before* or *After* an existing level. The instrument then displays the *Trigger Function Library*, which contains 12 state-based and 13 timing-based functions that can be assigned to the new level.
- **Function Modification:** Existing sequence stages can be reconfigured through the *Select New Function* option. This mechanism allows the user to replace the trigger logic at the sequence level without having to recreate the entire trigger structure. The corresponding graphical representation and parameter fields are updated automatically.
- **Conditional and Storage Controls:** Trigger terms can be assigned to functional expressions such as “*While storing*”, “*TRIGGER on*”, and “*Else on*”. These expressions define the conditions governing data acquisition, trigger generation, storage qualification, and sequence progression.

2.1.2.4 Practical Trigger Examples

During the experimental validation of the SoC-Wimed platform, dedicated trigger configurations were developed for each monitored communication interface. In the UART measurement setup, trigger term *a* was assigned to the *UART* signal group and configured

to detect a falling edge on the *TX* line. This configuration enabled the analyzer to trigger on the UART start bit, which occurs when the transmission line transitions from its idle high state (1) to a low state (0), marking the beginning of a data frame.

For SPI measurements, a two-level sequential trigger configuration was employed. The first sequence level monitored the *CS* (Chip Select) signal and required a transition to its active low state (0). Upon detection of this condition, the trigger sequence advanced to the second level, where the SPI data signals (*MOSI/MISO*) were evaluated. This configuration synchronized the acquisition process with the start of an SPI transaction, allowing detailed observation of the first transmitted data byte and the subsequent communication activity. The Level 2 trigger configuration used during the experimental validation is shown in Figure 2.6.

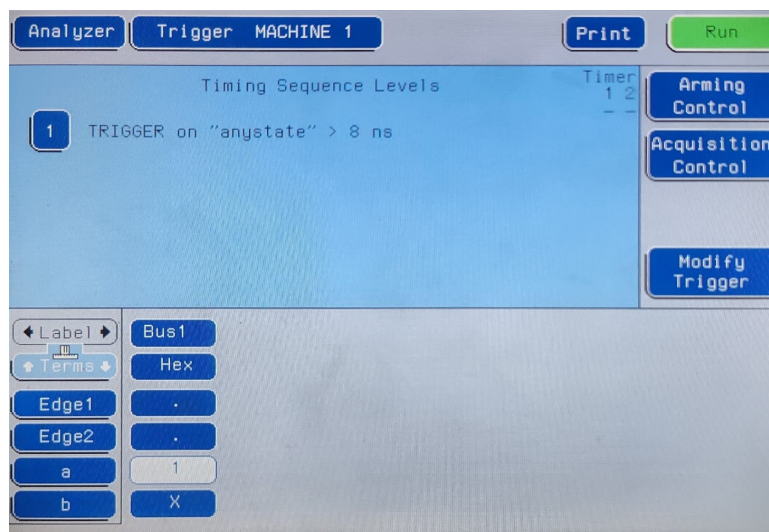


Figure 2.6: Sequence Level pop-up window configured with custom terms for experimental validation.

2.1.3 Acquisition Modes

Figure 2.7 shows the two supported acquisition modes: *Timing Analysis* and *State Analysis*. The choice between these modes depends on the characteristics of the digital interface under investigation, particularly on whether signal activity is interpreted with respect to an external clock signal or must be reconstructed from the analyzer's internal time base.

For interfaces that do not provide a dedicated clock signal, such as the asynchronous *UART* bus used to transmit processor diagnostic messages, the analyzer typically operates in *Timing Analysis* mode. In this configuration, signal transitions are sampled using the instrument's internal high-frequency timing reference, enabling reconstruction and analysis of the interface's temporal behavior.

In contrast, synchronous interfaces such as *SPI* provide an explicit clock signal that defines when data should be sampled. In these cases, the analyzer can operate in *State Analysis* mode, using the external clock signal as the acquisition reference. Data are then sampled only on the selected clock edge, ensuring that captured values correspond directly to the logical states observed by the target system. This approach provides a more accurate representation of protocol transactions and simplifies the interpretation of synchronous communications.

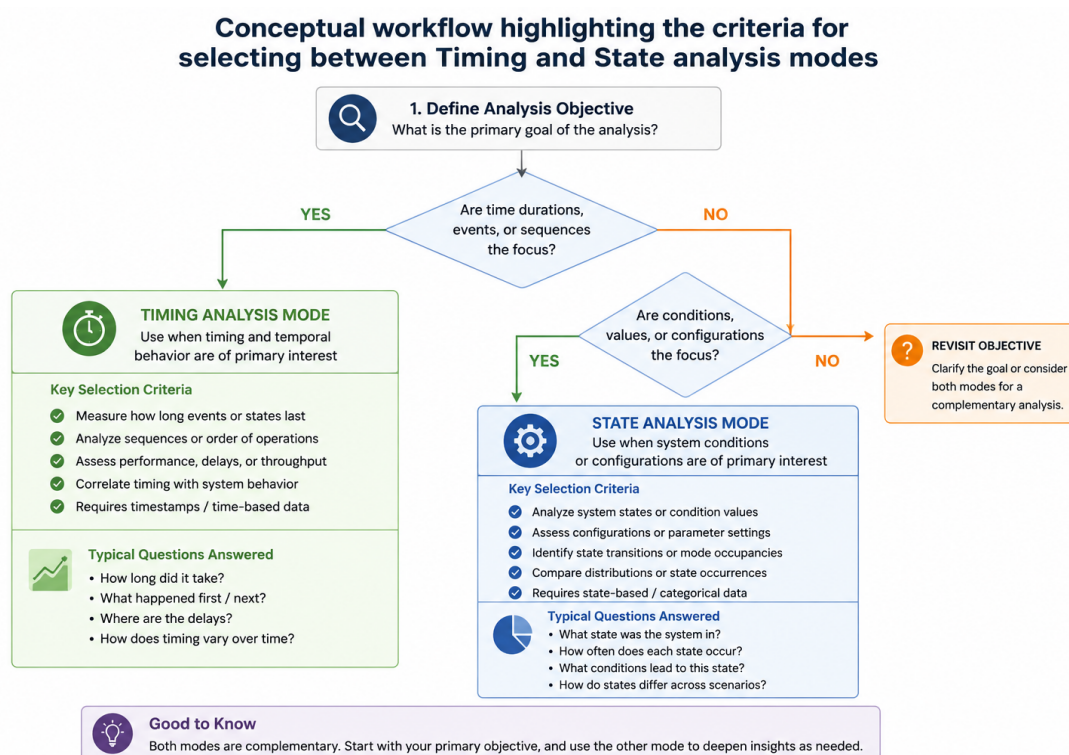


Figure 2.7: Conceptual workflow highlighting the criteria for selecting between Timing and State analysis modes.

2.1.3.1 Timing Analysis Pathway (Internal Clock)

Timing analysis consists of sampling and storing digital signals at uniform time intervals using the Agilent 1672G's internal clock source. In this acquisition mode, the analyzer operates independently of the target system's timing signals and records logic transitions according to its own sampling rate. As a result, timing analysis provides a time-based representation of signal activity, similar in principle to the operation of a digital oscilloscope.

Despite this similarity, important architectural differences distinguish a logic analyzer from a conventional oscilloscope:

- **Channel Count:** A logic analyzer supports a substantially larger number of input channels, up to 64 simultaneous digital inputs in the Agilent 1672G. This capability enables the

concurrent observation of wide parallel buses and multiple digital interfaces that would be impractical to monitor with a typical oscilloscope.

- **Voltage Resolution:** Oscilloscopes measure and display analog voltage waveforms with high vertical resolution. In contrast, a logic analyzer converts each sampled signal into a binary value. The acquisition circuitry compares the input voltage to a predefined threshold, classifying samples above the threshold as a logic high (1) and below as a logic low (0).

In this work, Timing Analysis mode was initially used to verify the operation of the asynchronous UART interface employed for processor diagnostics and software debugging. Since UART communication does not rely on a dedicated clock signal shared between the transmitter and receiver, synchronization is achieved through a predefined baud rate. Consequently, signal integrity and protocol timing must be evaluated directly from the temporal characteristics of the transmitted waveform.

The initial Timing Analysis acquisitions confirmed the correct operation of the UART interface. However, reliable decoding required a high sampling frequency, resulting in a large number of samples per transmitted bit and increased memory consumption, which limited the duration of the acquisitions. For this reason, the remaining experiments were performed using State Analysis mode with an externally generated sampling clock. By providing this external clock, State Analysis enabled longer acquisitions while maintaining sufficient temporal resolution for reliable UART decoding.

The timing diagrams generated by the Agilent 1672G enabled direct observation of the UART transmission line (TX), allowing individual bit intervals to be measured and compared against the expected baud rate configuration. By examining the durations of the start bit, data bits, and stop bit, it was possible to verify that the communication parameters were correctly configured and that the transmitted frames conformed to the UART protocol specification.

2.1.3.2 Timing Mode Hardware Setup

The timing analysis workspace is accessed by configuring the analyzer for Timing Analysis mode. Upon startup, the Agilent 1672G automatically displays the Configuration menu. If Analyzer 1 (Machine 1) has not yet been assigned a timing acquisition task, the mode can be configured through the Analyzer Configuration interface as follows:

1. Press the front panel *Config* key to open the Analyzer Configuration workspace.
2. Navigate to the *Type* field within the Analyzer 1 configuration panel and press the *Select* key.
3. Choose the *Timing* option from the configuration menu and confirm the selection with the *Select* key.

Once the configuration is complete, the analyzer status line indicates that the instrument is operating in Timing Analysis mode. As shown in Figure 2.8, this confirmation verifies that the acquisition engine is ready to perform time-based signal measurements.

An additional feature of the Agilent 1672G is the presence of activity indicators associated with the configured probe pods. These indicators provide a visual representation of signal transitions detected on the monitored channels. When a signal crosses the configured logic threshold and changes state, the corresponding activity indicator responds dynamically, providing immediate feedback that the input channel is receiving valid digital activity.

In the experimental setup developed for the SoC-Wimed platform, these indicators served as a preliminary diagnostic tool. Before configuring trigger conditions or initiating acquisitions, the activity markers allowed rapid verification that UART traffic was reaching the analyzer inputs, thereby confirming the integrity of the physical connections and the correct operation of the communication interface.

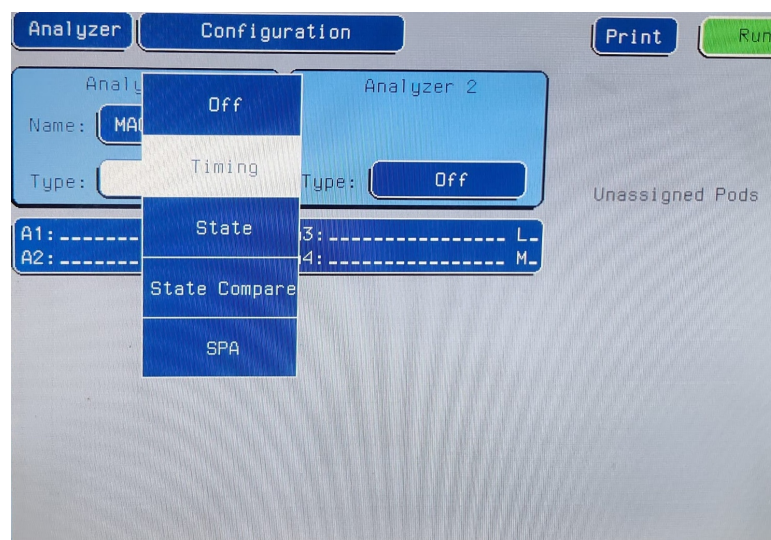


Figure 2.8: Analyzer Configuration menu illustrating Machine 1 mapped to the Timing type.

2.1.3.3 Label Customization and Sampling Rate Verification

Signal labels are used to organize physical channels into logical groups, referred to as buses, thereby simplifying signal identification and interpretation during acquisition and analysis. In Timing Analysis mode, custom labels can be assigned through the following procedure:

1. Press the *Format* key on the instrument front panel to open the signal mapping interface.
2. Navigate to the desired default label field, such as *Bus1*, and press the *Select* key.

3. Enter the desired alphanumeric label, for example, *UART* or *TCOUNT*, using the front panel entry controls. Existing characters may be edited using the backspace key, the navigation keys, or the rotary control knob.
4. Press the *Done* key to confirm and store the new label assignment.

As shown in Figure 2.9, the header area of the configuration interface also provides acquisition status information. In Timing Analysis mode, this region displays the active acquisition configuration, including the number of enabled channels and the current sampling frequency. In the configuration used throughout this work, the analyzer operated in full-channel acquisition mode at a sampling rate of 250 MHz, as indicated by the status message “64C Sample Full Channel 250 MHz”.

For the SoC-Wired validation experiments, verifying the sampling rate was essential to ensure accurate observation of UART communications. Since UART transmissions rely on fixed bit periods defined by the selected baud rate, a 250 MHz sampling clock provides a sampling interval of 4 ns, resulting in a substantial oversampling ratio relative to the UART bit duration. This oversampling capability enables accurate bit-timing measurement, facilitates detection of timing irregularities, and reduces uncertainty in identifying signal transitions within the acquired waveform.

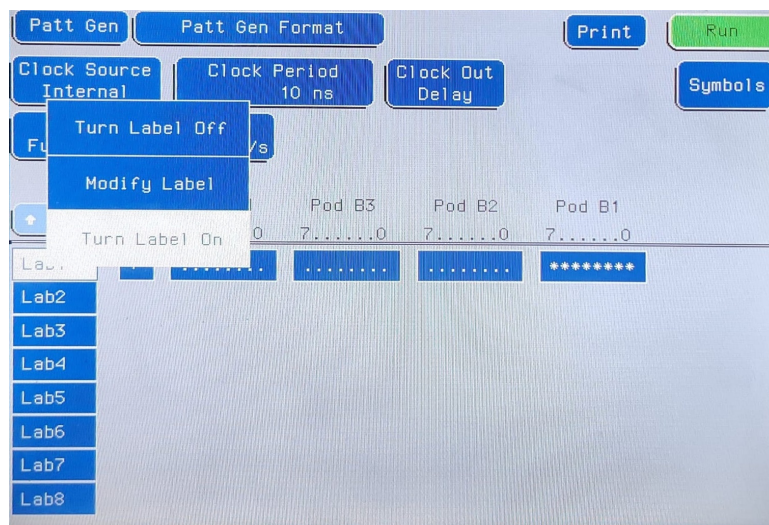


Figure 2.9: Format Machine menu illustrating a renamed label and the active 250 MHz Timing Acquisition Mode status bar.

2.1.3.4 Channel Assignment Reconfiguration

After creating a custom label within the *Format* menu, the corresponding physical channels must be assigned to that logical group. As illustrated in Figure 2.10, the default channel configuration can be modified through the front panel controls to match the interface under investigation:

1. Navigate to the graphical representation of the desired probe pod, located to the right of the selected label, and press the *Select* key.
2. Press the *Clear Entry* key to remove any existing channel assignments. All previously assigned channels are then displayed as unassigned dots (·).
3. Use the rotary control knob to position the cursor over the desired channel location within the channel assignment matrix.
4. Assign the selected channel to the logical group by marking the corresponding position with an asterisk (*).
5. Press the *Done* key to confirm and store the channel mapping.

The resulting display, shown in Figure 2.10, illustrates a configuration in which the lower eight channels of Pod A1 are grouped under a single logical label.

To experimentally validate the SoC-Wired platform, this procedure was used to isolate the signals associated with the UART communication interface. After creating the *UART* label, all default channel assignments were cleared, and only the channels connected to the UART signals were assigned to the group. In the final configuration, physical channels 0 and 1 were mapped to the logical label, allowing the analyzer to focus exclusively on the relevant communication signals. Restricting the acquisition to the required channels simplified waveform interpretation and reduced the amount of irrelevant information displayed during debugging and protocol verification.

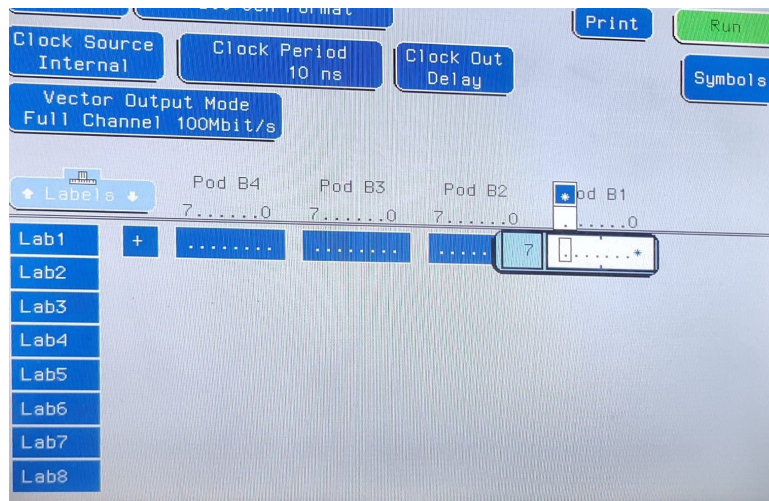


Figure 2.10: Channel assignment matrix interface showing active bit mapping using asterisks (*).

2.1.3.5 Timing Trigger Term Definition

To capture a specific communication event, the analyzer must be configured with an appropriate trigger condition that defines the point of interest within the acquisition. Trig-

ger terms are the fundamental elements of the trigger system, enabling signal patterns, logic states, or timing conditions to be associated with the trigger engine. These conditions determine when the analyzer identifies a significant event and establishes the trigger reference point within the acquisition buffer.

The procedure used to assign a trigger condition to a trigger term in Timing Analysis mode is summarized below and illustrated in Figure 2.11:

1. Press the *Trigger* key on the front panel to open the trigger configuration workspace.
2. Navigate to the label associated with the monitored signal group. Position the cursor on the radix field beneath the custom *UART* label, press *Select*, choose the desired representation format (*Hex* or *Binary*), and confirm the selection.
3. Move the cursor to the row corresponding to the desired trigger term, such as *a*, and enter the required trigger value using the front panel keypad. Confirm the entry by pressing the *Done* key.

As shown in Figure 2.11, this procedure associates the selected signal pattern with trigger term *a*, making it available to the analyzer's trigger sequencing system.

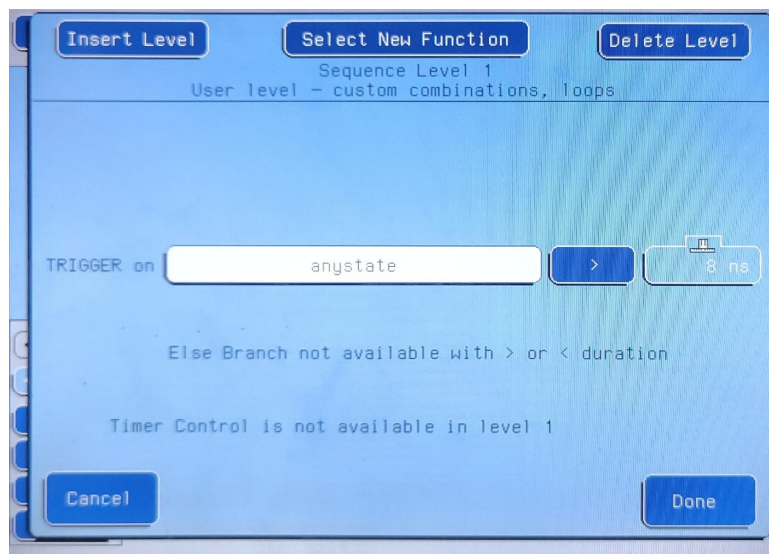


Figure 2.11: Trigger configuration menu showing a defined term and its assigned hexadecimal condition.

Although the instrument documentation commonly demonstrates trigger assignment using multi-bit buses and hexadecimal patterns, the SoC-Wimed validation setup required a substantially simpler configuration. The monitored UART interface consisted of only a few active signals, and the trigger term was associated with the UART transmission channel to detect the start of a communication frame. In the configuration used during testing, the trigger condition was defined to detect the UART start bit transition, allowing the acquisition to be aligned with the onset of data transmission.

As indicated in Figure 2.11, the trigger summary reports the condition “TRIGGER on $a > 8 \text{ ns}$ ”. In this context, trigger term a was associated with the UART transmission signal, and the timing qualification ensured that the detected event satisfied the specified duration requirement before being recognized by the trigger engine. This configuration enabled reliable capture of UART communication activity while minimizing false triggers caused by short transient events or signal noise.

2.1.3.6 Timing Trigger Specification and Filter Glitch Tuning

Once the trigger terms have been defined, the trigger qualification criteria must be configured to specify how long a signal condition must remain valid before the analyzer recognizes it as a trigger event. In Timing Analysis mode, this duration requirement is configured through the sequence level settings, as illustrated in Figure 2.12.

The configuration procedure is summarized as follows:

1. Open the *Trigger* menu and navigate to the desired sequence level within the *Timing Sequence Levels* panel. Select the corresponding level, for example, Level 1, to open the sequence configuration window.
2. Position the cursor in the field immediately to the right of the *Trigger On* entry and press *Select*.
3. Choose the previously defined trigger term, such as a , and confirm the selection.
4. Navigate to the duration qualification field located to the right of the comparison operator ($>$) and specify the required minimum duration, for example, 16 ns .
5. Press *Done* to store the configuration and exit the sequence level setup window.

As shown in Figure 2.12, the duration qualification serves as a filtering mechanism that prevents the trigger system from responding to extremely short signal disturbances. A trigger event is generated only if the specified logic condition remains valid for at least the configured time interval. This requirement reduces the likelihood of false triggers caused by transient effects such as ringing, reflections, metastable transitions, or electrical noise.

During the validation of the SoC-Wired communication interfaces, this feature proved particularly useful when monitoring asynchronous UART traffic. By requiring a signal transition to remain valid for a minimum duration before being accepted as a trigger event, the analyzer could distinguish genuine protocol activity from short transient disturbances. This qualification improved the reliability of trigger detection and ensured that acquisitions were aligned with valid UART communication events rather than spurious signal fluctuations.

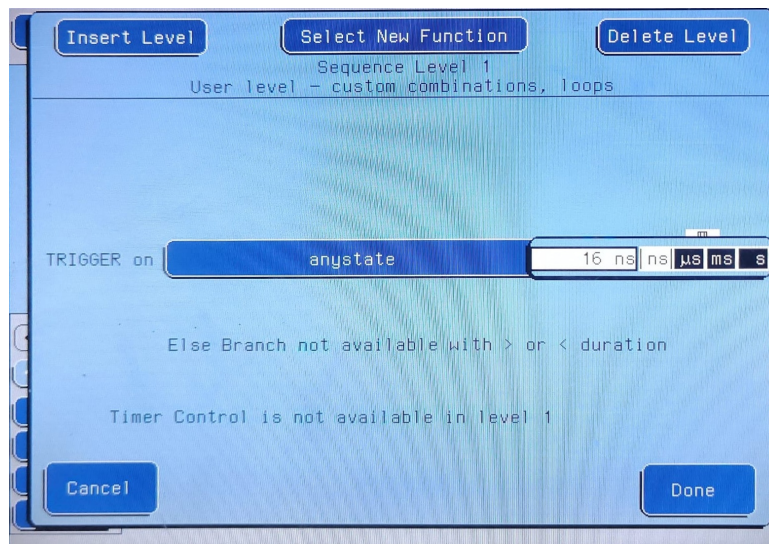


Figure 2.12: Timing Sequence Level pop-up window configured to trigger on term *a* with a minimum duration filter of 16 ns.

2.1.3.7 Execution and Waveform Analysis

With the timing parameters defined and the acquisition sequence configured, capture can begin. Pressing the physical Run button activates monitoring of the selected channels. When the programmed hardware trigger condition is met, the logic analyzer stores the captured data in a circular buffer and automatically switches the interface to Waveform mode, allowing the operator to analyze the captured signal waveforms. Three primary controls are available for viewing and manipulating waveform data (see Figure 2.13):

- **Scaling the Waveform** – Highlighting the “sec/Div” field in the Waveform menu and rotating the control clockwise zooms in on signal activity, enabling a closer examination of waveform details. Further rotation progressively increases the magnification. Rotating the control counterclockwise zooms out, allowing a larger portion of the capture memory, or the entire capture, to be viewed.
- **Scrolling Through the Waveform** – Highlighting the “Delay” field and rotating the control clockwise or counterclockwise allows the user to navigate through the captured data and review samples recorded before or after the trigger event.
- **Measuring a Time Interval** – Highlighting the “Markers Off” field and pressing the physical “Select” button enables the display of horizontal timing markers. After highlighting the “Trig to X” field, rotating the control adjusts the X marker position. As the X marker crosses digital transitions, the corresponding bus value is displayed in hexadecimal format at the marker location in real time.

To support the examination of the SoC Wimed Processor, the analysis environment was used to probe the programmed UART interface. The capture was configured to trigger on the falling edge of the TX signal, and the sec/Div control was used to zoom in on individual bits. Using the timing markers, specifically the X marker, the time interval between signal

transitions was measured accurately. These measurements enabled the calculation of the actual baud rate and verification of the UART output timing. This verification is necessary to ensure reliable communication and correct logging of transmitted data.

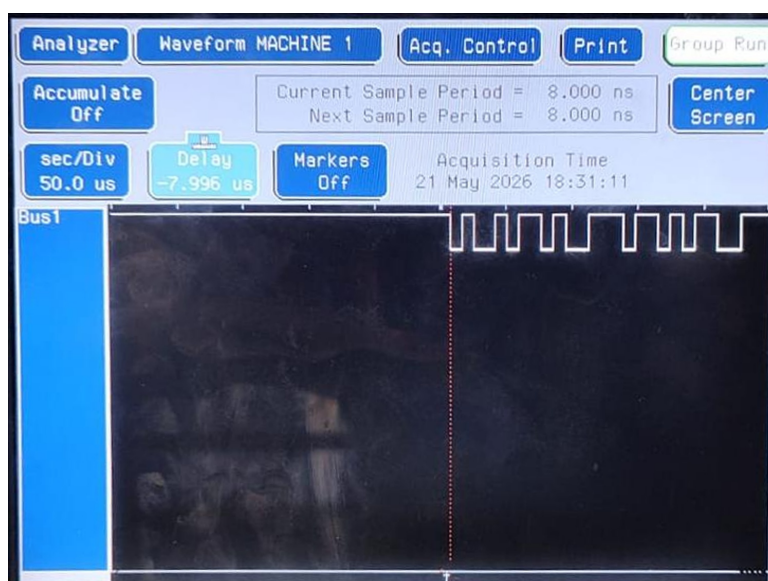


Figure 2.13: Waveform menu interface showing digital signal transitions, time markers, and hex bus decode lines at the marker position.

2.1.3.8 State Analysis Pathway (External Clock)

State analysis captures and stores digital information synchronously with an external clock signal supplied by the system under test. In this mode, the Agilent 1672G changes its acquisition strategy from periodic time-based sampling to capturing data on specific transitions of the target system's clock signal. This approach is suitable for protocols that require synchronous access to a data bus, such as the Serial Peripheral Interface (SPI). Data on an SPI bus is sampled on defined edges of the SCLK signal, with the specific sampling edge determined by the configured SPI clock phase.

2.1.3.9 State Mode Hardware Configuration Setup

Synchronous, clock-based analysis is enabled through the analyzer control interface by performing the following steps:

1. Press the dedicated *Config* key on the front panel to open the analyzer configuration menu.
2. Using the front panel cursor controls, navigate within the Analyzer 1 configuration area to the field adjacent to the *Type* setting, then press the *Select* key.

3. From the displayed pop-up menu, select *State* and press the *Select* key again to confirm the selection.

After the configuration is applied, as shown in Figure 2.14, the analyzer status bar indicates that the acquisition engine is operating in State analysis mode.

As an initial validation step, activity indicators, displayed as dynamic vertical arrows next to the assigned channels, can be used to verify the presence of a signal. In State mode, these indicators confirm that signal transitions are crossing the configured threshold voltage. During the initial evaluation of the SoC Wimed platform, activity indicators were observed on the lower channels of Pod A1, including the J clock signal channel. The presence of these indicators confirmed that both clock and data activity were successfully detected by the analyzer probe before runtime acquisition was initiated.

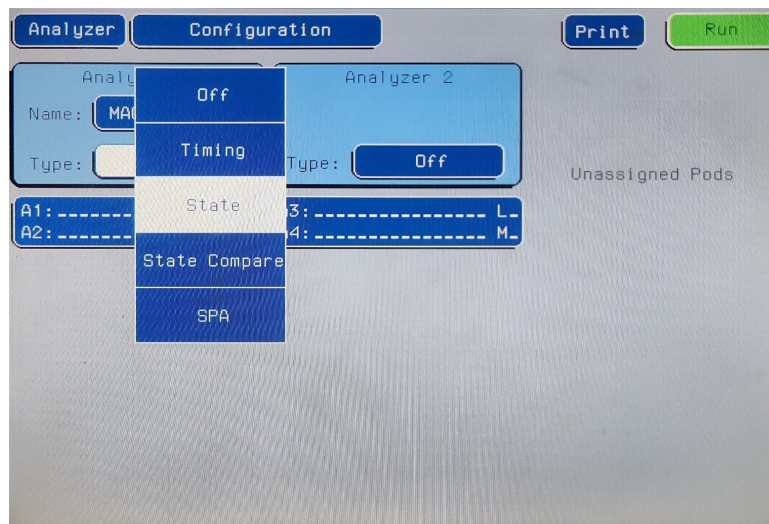


Figure 2.14: Analyzer Configuration menu illustrating Machine 1 mapped to the State type and displaying channel activity indicators along the J-clock channel.

2.1.3.10 State Label Customization and System Clock Verification

In State mode, labels group physical channels into logical buses, enabling synchronized analysis of related signals. To align the analyzer configuration with the debug interface of the target hardware, the default label assignments can be modified using the following procedure:

1. Press the *Format MENU* button on the front panel to open the signal mapping display. Using the front panel arrow keys, highlight the *Bus2* field within the Labels section and press the *Select* key.
2. Using the front panel alphanumeric controls, enter the desired label name for the synchronous bus, such as *SPI*, and then press the *Done* button to confirm the entry.

3. To conserve acquisition resources and simplify the display, disable any unused default label groups, such as *TCOUNT* or *BCOUNT*, by selecting the corresponding label and choosing the *Turn Label Off* option.

As shown in Figure 2.15, this procedure simplifies the configuration display and clearly identifies the active State mode settings in the main interface window. In the illustrated configuration, the Agilent 1672G state acquisition engine is configured to operate at its maximum synchronous clock frequency of 150 MHz with a capture depth of 64K samples, indicated by the status message “150 MHz/64K State”. Because State mode relies entirely on an externally supplied clock to control data acquisition, this specification provides sufficient bandwidth for the SoC’s Wimed SPI bus clock frequency, enabling reliable capture of synchronous bus activity.

The master clock setting indicates that data is sampled on the falling edge of the J clock signal ($J \downarrow$), which is connected to the SPI serial clock (SCLK) line on the development board. This configuration ensures that all captured data is synchronized to the bus clock and accurately reflects the timing relationships present on the SPI interface.

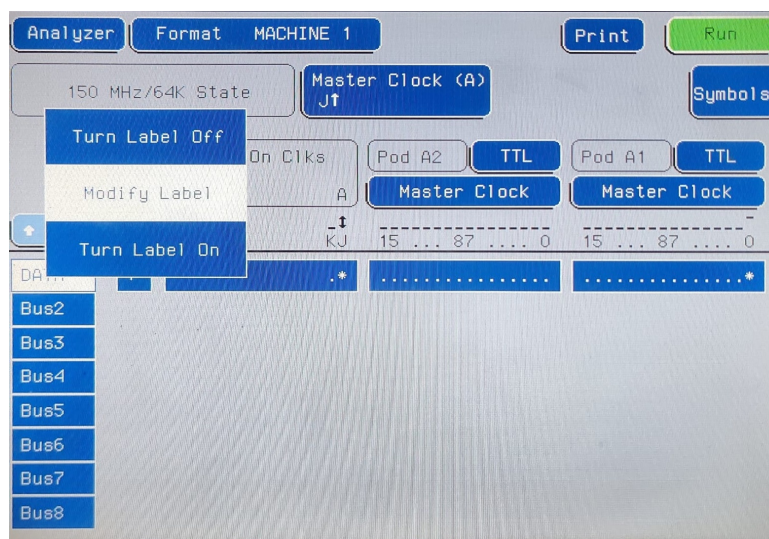


Figure 2.15: Format Machine menu in State Mode illustrating the customized SPI label and the active 150 MHz master clock status indicator assigned to the falling edge of the J-clock line.

2.1.3.11 State Trigger Term Definition

To enable the state analyzer to isolate and capture synchronous transactions of interest, trigger terms must be defined to specify the conditions under which the circular capture buffer is preserved. Trigger terms act as hardware-evaluated conditions that monitor incoming bus activity for specific patterns or values. When a configured condition is satisfied, the analyzer records the relevant transaction data and makes it available for further analysis.

To associate a specific transaction pattern or value with a trigger term in the State mode interface, the procedure shown in Figure 2.16 should be followed:

1. Press the *Trigger MENU* key on the front panel to open the sequence configuration and trigger term assignment interface.
2. Using the navigation controls, highlight the numeric base field located beneath the custom *SPI* label and press the *Select* key to display the available formatting options.
3. Select *Hex*, or another desired numerical representation, and press the *Select* key to confirm the input format.
4. Navigate to the data entry field corresponding to the desired trigger term, for example, term *b*, under the appropriate label column. Enter the required hexadecimal value using the front panel keypad and press the *Done* key to store the trigger criterion.

Following this configuration, as illustrated in Figure 2.16, the specified value becomes associated with the selected trigger term and is made available to the analyzer's multi-level sequencing engine.

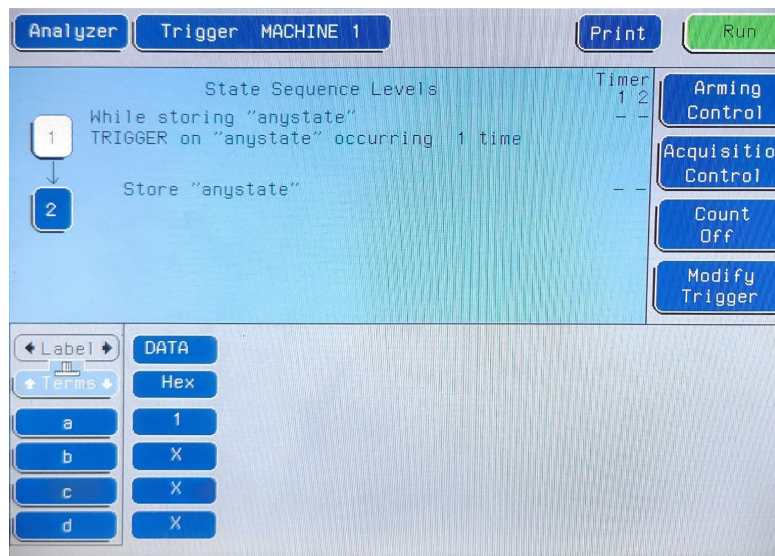


Figure 2.16: Trigger configuration menu in State Mode showing a customized term defined under the active synchronous bus label.

To evaluate the SoC Wimed platform, this procedure was used to define trigger conditions for the SPI bus. Rather than triggering on generic register values or broad bus activity patterns, term *b* was configured to monitor all three active signals grouped under the *SPI* label. Hexadecimal bit masks were selected to represent the active-low chip select (CS) signal and the corresponding initial states of the MOSI and MISO lines. This configuration enabled the state analyzer to detect the precise start of an SPI transaction. As a result, acquisition was restricted to relevant bus activity, reducing unnecessary captures and improving the efficient use of available capture memory.

2.1.3.12 Setting Up Level 1 of the State Trigger Specification

In State mode, event-driven acquisition is implemented by defining a sequence of trigger levels that the hardware acquisition engine evaluates before committing the final capture to memory. This sequencing mechanism allows the analyzer to monitor a series of bus events and advance through predefined states until the desired trigger condition is satisfied. According to the instrument documentation, the configuration of Level 1 is performed using the following procedure:

1. In the Trigger menu, highlight the field labeled *1* within the State Sequence Levels section and press the *Select* key to open the Level 1 configuration window.
2. Highlight the value field associated with the *TRIGGER on* command, press the *Select* key, and choose the desired trigger term, such as term *b*, from the available options.
3. Press the *Done* key twice to confirm the selection and close the Level 1 configuration window.

As shown in Figure 2.17, this configuration causes the analyzer to store all samples that satisfy the selected storage qualification condition, for example “anystate”, from the moment the *Run* key is pressed until the trigger condition associated with term *b* is detected.

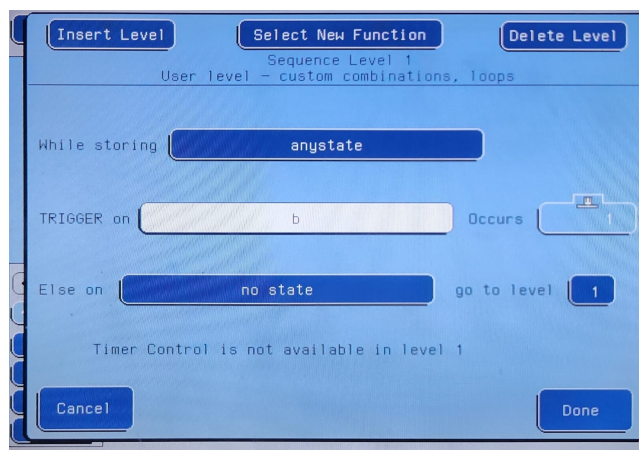


Figure 2.17: Sequence Level 1 pop-up menu configuration screen illustrating the assignment of term *b* as the primary state sequence condition.

For the evaluation of the SoC Wimed platform, the sequence engine was configured to identify specific SPI communication events rather than broader system states. Consequently, Level 1 was programmed to detect the start of an SPI transaction by the occurrence of the term *b*, which was defined using the three signals grouped under the *SPI* label. This trigger condition corresponds to asserting the chip select signal together with the initial states of the MOSI and MISO lines. By identifying this event boundary, the analyzer can accurately detect the start of a communication frame and advance to Level 2, where the next stage of the trigger sequence is evaluated.

2.1.3.13 Setting Up Level 2 of the State Trigger Specification

In Sequence Level 1, the analyzer's multi-level trigger engine monitors and records bus activity while searching for the event that marks the beginning of an SPI communication frame. Once this trigger condition is detected, Sequence Level 2 is entered, during which the acquisition engine continues to record subsequent bus activity. This arrangement allows the analyzer to capture the complete transaction following the identified frame boundary while preserving the relevant pre-trigger context recorded during Level 1.

Configuring Sequence Level 2 involves the following procedure:

1. Highlight the 2 field within the State Sequence Levels section of the Trigger menu and press the *Select* key to open the Sequence Level 2 configuration window.
2. Highlight the field adjacent to the *Store* action and press the *Select* key. From the displayed menu, select the *anystate* option to record all valid synchronous bus states, then press the *Done* key to confirm the selection.
3. Press the *Done* key once more to close the Sequence Level 2 configuration window and return to the main trigger menu.

As shown in Figure 2.18, the State Trigger Specification is now fully defined as a two-stage trigger sequence. Level 1 instructs the analyzer to record qualified bus activity while searching for the trigger condition associated with term *b* (While storing 'anystate', TRIGGER on 'b' 1 time). Once this condition is satisfied, Level 2 continues to record qualified bus activity according to the command 'Store 'anystate'.

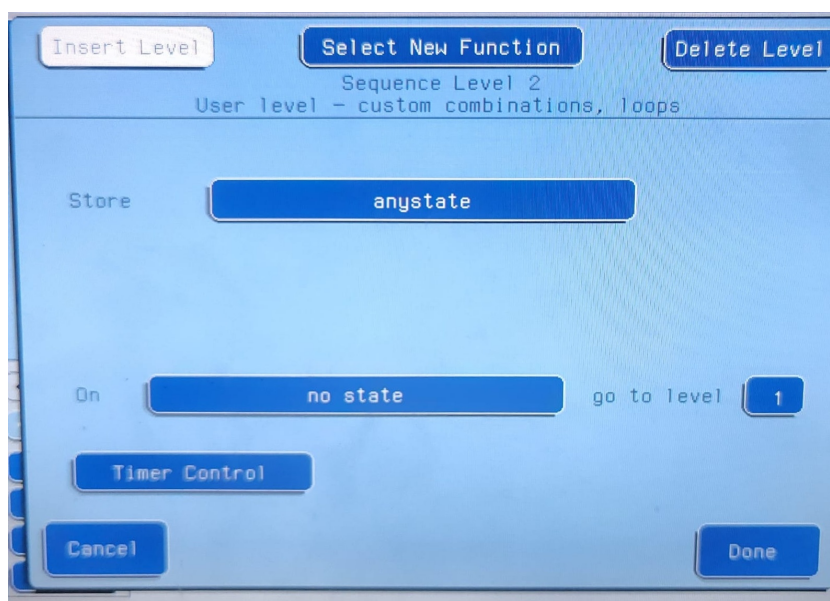


Figure 2.18: Sequence Level 2 pop-up menu configuration screen illustrating the command to store all subsequent synchronous states once the initial frame boundary is triggered.

For evaluating the SoC Wimed platform, this two-level trigger sequence was used to maximize the use of the available capture memory when analyzing the three-signal SPI interface. After Level 1 detects the beginning of an SPI transaction through the assertion of the active low chip select (CS) signal and the corresponding MOSI and MISO states defined by term *b*, control advances immediately to Level 2. The analyzer then continues storing all qualified synchronous bus states, allowing the remainder of the capture memory to be dedicated to the SPI transaction of interest. This approach increases the proportion of captured samples associated with active communication while reducing the memory overhead from idle bus activity.

2.1.3.14 Running State Acquisition and Listing Analysis

The final step before examining the captured state data is to execute the acquisition sequence and review the resulting state listing generated by the analyzer. According to the operating procedure described in the Agilent 1672G user documentation, state-based acquisitions are analyzed using the following steps:

1. Press the front panel *Run* button to initiate synchronous state acquisition.
2. Examine the *State Listing* display. The trigger event is positioned at line 0 within the listing and represents the point at which the configured trigger condition was satisfied. Entries with negative line numbers correspond to states captured before the trigger event, whereas entries with positive line numbers correspond to states captured after the trigger event.
3. If an alternative numerical representation is required, highlight the *Hex* field beneath the custom *SPI* label and select the desired format, such as *Decimal* or *Binary*, from the displayed menu.

As shown in Figure 2.19, the *State Listing* provides a tabular representation of the captured synchronous bus states, allowing each transaction to be correlated with its position relative to the trigger event. The captured data can be navigated using either the rotary control or the paging keys on the front panel.

The *State Listing* is particularly useful for verifying SPI communications during hardware validation. Because acquisition is synchronized to the falling edge of the SCLK signal through the J clock input, each entry in the listing corresponds to a valid clock-qualified bus state. This allows hexadecimal SPI values to be examined directly without requiring interpretation of individual waveform transitions. Consequently, protocol activity can be verified more efficiently than in a timing-based acquisition, where signal values must often be reconstructed from waveform data.

The State Analyzer also provides a *Waveform* display mode for visual inspection of captured signals. However, the functionality and navigation in this mode are largely similar to those previously described for Timing mode analysis. As a result, the *State Listing* remains the primary interface for examining synchronous protocol transactions, while the waveform view serves as a complementary tool for investigating timing relationships when required.

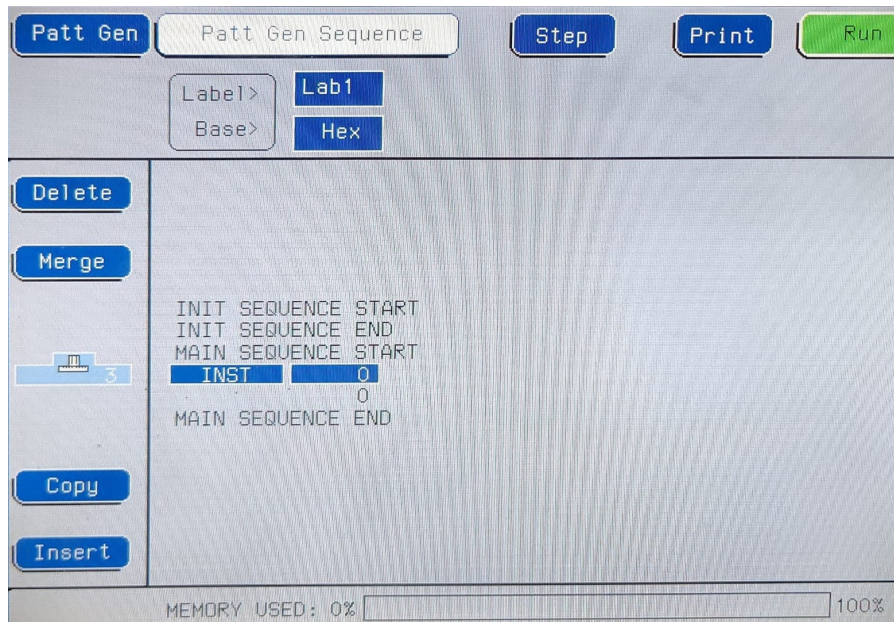


Figure 2.19: State Listing display interface demonstrating the indexed table format centered around the zero-point trigger marker.

2.1.4 Pattern Generator

In addition to recording and analyzing digital signals, it is often desirable to generate controlled stimulus patterns to evaluate system behavior under specific operating conditions. To support this capability, some logic analyzers incorporate an integrated pattern-generation function that provides programmable digital outputs to drive the system under test while simultaneously monitoring its response.

For general diagnostic applications, this functionality can be used to generate pre-defined test patterns, such as walking ones sequences, while performing concurrent timing analysis to evaluate signal integrity and verify correct system operation.

For characterizing test platform and development kit, this feature provides a controlled mechanism to investigate system behavior across a range of operating conditions. By injecting programmable digital stimuli through the logic analyzer, the system can be exercised using carefully defined input sequences. Depending on the configuration, the logic analyzer can emulate aspects of SPI master or slave behavior, enabling the generation of representative communication transactions for validation.

This capability allows synthesized SPI sequences to be used to explore protocol edge cases, evaluate transaction boundary handling, and assess the robustness of on-chip communication logic. Furthermore, controlled variations in signal timing, including clock jitter or timing offsets, can be applied to assess the tolerance of synchronization structures and communication interfaces to timing perturbations. Such testing provides valuable insight into the operational limits and reliability of the hardware design under non-ideal conditions.

2.1.4.1 Pattern Generator Interface Configuration and Triggering

To evaluate the external stimulus generation capabilities of the Agilent instrumentation platform, the integrated pattern generator is configured alongside the existing logic analyzer signal assignments. Rather than repeating the previously described configuration procedures for signal acquisition, the required settings are consolidated in the analyzer's Format menu by defining a dedicated workspace for pattern generation and monitoring.

As with the other analyzer configurations, the default label *Bus2* within the Format menu is renamed to *PATGEN*, as shown in Figure 2.20. Unused default labels, such as *TCOUNT*, are disabled using the *Turn Label Off* option to simplify the display and reduce unnecessary resource usage. The signal assignment matrix is then modified so that channels 8 through 11 on Pod A1 are grouped under the *PATGEN* label using the front panel asterisk (*) channel selection mechanism, as illustrated in Figure 2.21. This four-channel group provides a direct loopback connection that allows the logic analyzer to monitor the signals generated by the pattern generator and verify their correct operation.

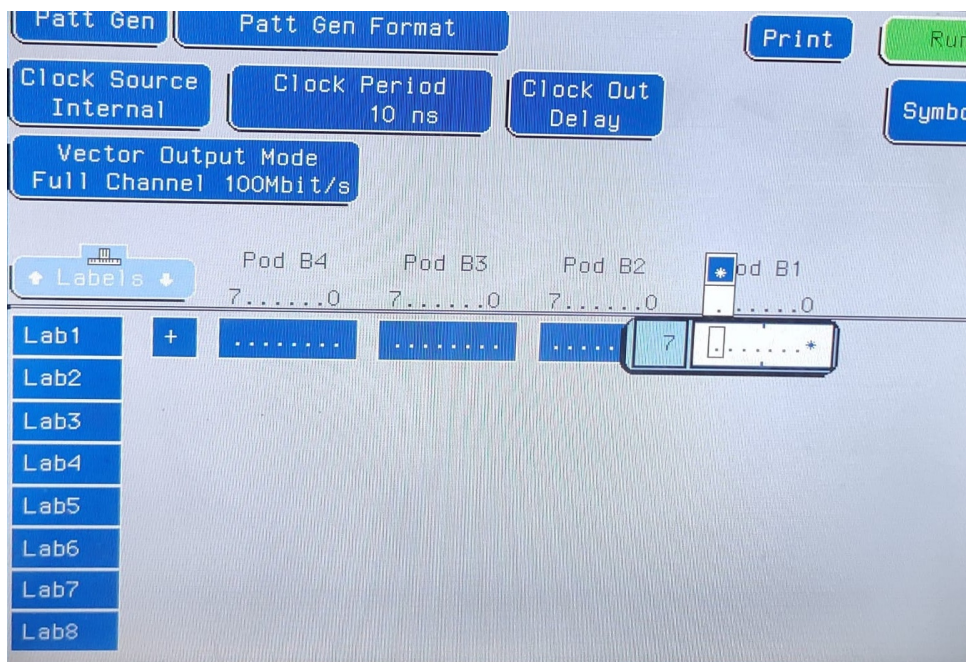


Figure 2.20: Format menu configuration screen illustrating the creation of the custom PAT-GEN label and deactivation of background groups.

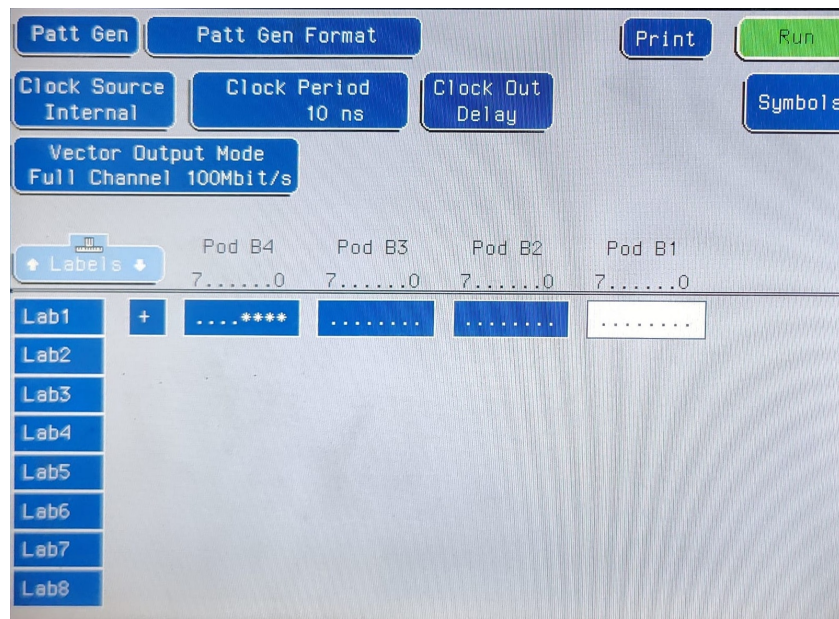


Figure 2.21: Channel assignment matrix highlighting the specific four-pin asterisk (*) selection on Pod A1 mapped to the PATGEN bus.

2.1.4.2 Hardware Architecture and Clock Source Configuration

To initialize the stimulus generation system, control must be transferred from the logic analyzer interface to the pattern generator subsystem. Rather than configuring each output channel individually, the pattern generator allows related signals to be grouped into a dedicated output bus, simplifying the definition of stimulus vectors that will be applied to the device under test (DUT).

Pattern generator configuration begins within the Format menu by defining a custom output group and assigning it the label *PATGEN*. The channel assignment is then restricted to the lower four output channels, corresponding to channels 0 through 3 on Pod B4, using the front-panel controls and the asterisk (*) channel-selection mechanism. To prevent unintended signal interactions and ensure that only the required outputs are driven, the channel assignments associated with Pods B3, B2, and B1 are cleared using the *Clear Entry* function. This configuration ensures that only the designated output channels participate in stimulus generation.

Accurate stimulus generation also requires the output vectors to be synchronized to a defined clock source. Within the pattern generator configuration menu, the *Clock Source* parameter is set to *Internal*, causing the module to generate output transitions according to its internal timing reference. For the evaluation of the SoC Wimed platform, the output clock period was configured to 10 ns, corresponding to a vector generation rate of 100 MHz. This fixed timing reference enables the pattern generator to produce a deterministic sequence of digital output vectors with consistent timing characteristics, facilitating the validation of the target system's synchronous input circuitry and communication interfaces.

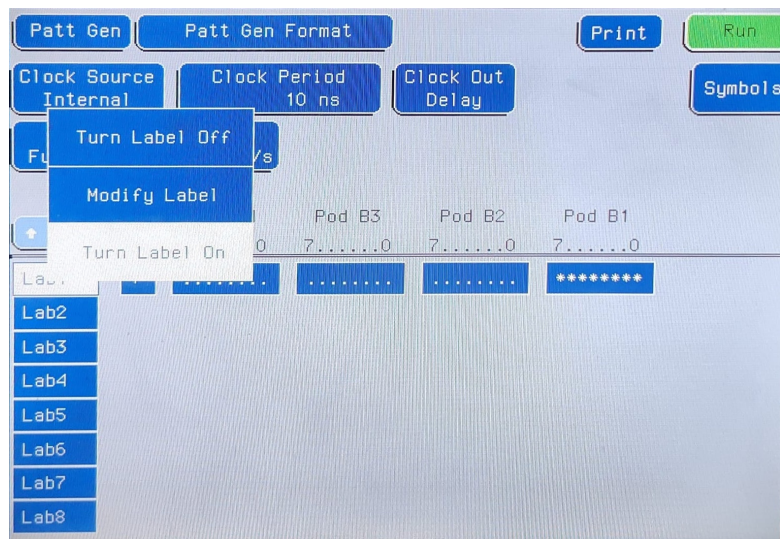


Figure 2.22: Pattern Generator Format Menu interface showing the customized PATGEN label mapped to the lower bits of Pod B4 alongside an internal clock source configuration.

2.1.4.3 Vector Sequence Programming and Pattern Generation

The generation of a digital stimulus sequence is accomplished by programming a series of output vectors that are executed by the pattern generator. According to the instrument interface, the user transitions from the pattern generator configuration settings to the sequence programming environment by selecting the *Patt Gen Format* field and opening the *Patt Gen Sequence* editor. This editor provides a vector-based programming interface in which each output state is stored directly in the pattern generator's memory and executed sequentially during operation. The procedure for creating a stimulus program is as follows:

1. Within the sequence editor, configure the data display format by selecting the numeric base field associated with the *PATGEN* column and choosing *Binary*. This representation facilitates direct programming of individual output bits.
2. Position the cursor at the first programmable vector entry immediately below the sequence start marker.
3. Use the *Insert* key to add additional vector entries as required, constructing the desired sequence of output states.

As illustrated in Figure 2.23, a simple example is implementing a walking ones test pattern across the four output channels. In this sequence, a single logic high is shifted through successive bit positions, producing vectors such as 0001, 0010, 0100, and 1000. This pattern is widely used for validating digital interfaces because it exercises each signal line individually while maintaining a known relationship between adjacent channels.

For characterizing the SoC Wimed platform, the walking-ones sequence provides a controlled, repeatable stimulus to verify the correct operation of the digital input path.

By replacing unpredictable application traffic with a deterministic test pattern, each input channel can be evaluated independently while operating at the configured vector rate. This approach facilitates the detection of signal integrity issues, including unintended coupling between neighboring traces, timing skew between channels, and errors introduced during high-speed data capture. Consequently, the walking ones pattern serves as an effective diagnostic tool for validating the integrity of the board-level interconnects and the associated input circuitry.

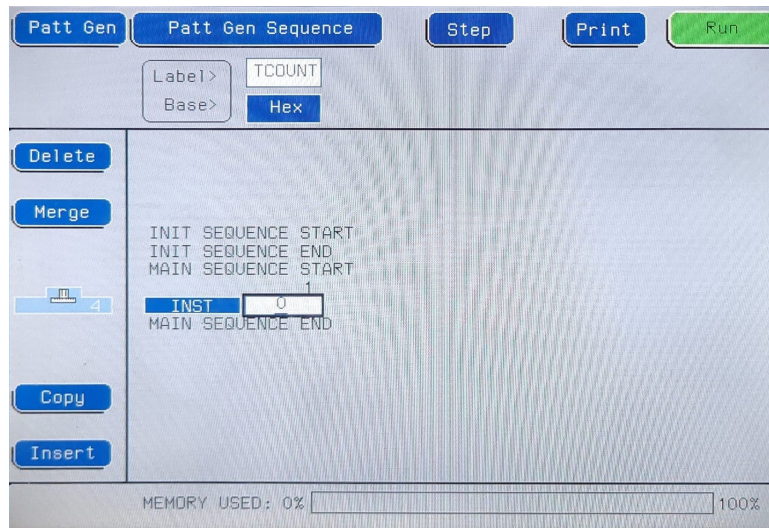


Figure 2.23: Pattern Generator Sequence editor showing the completed multi-line vector program compiled in binary format.

2.1.4.4 Pattern Generator Execution and Waveform Verification

The final verification of the stimulus-response chain was performed by executing the generated vector sequences on the pattern generator and observing the resulting signal stream using the waveform acquisition module of the timing analyzer. Rather than navigating through the instrument menus step by step, the stimulation and acquisition process was initiated through a direct operating sequence that enabled both instruments in rapid succession:

1. Start continuous pattern generation by pressing the physical blue *SHIFT* key followed by the *Run* key on the pattern generator front panel. This operation causes the programmed bit sequence to repeat continuously.
2. Return the display to the timing analyzer workspace and execute a single waveform acquisition using the *Single* capture function within the *Run* menu. Adjust the horizontal time scale (*sec/Div*) to 20 ns to obtain a convenient view of the captured signals.

As shown in Figure 2.24, the captured timing trace displays four synchronized digital signals. All acquired channels grouped under the *PATGEN* label transition through

identical high and low logic states, reproducing the programmed binary pattern exactly as expected.

This procedure served as a preliminary loopback validation test. Its purpose was to verify the physical connections, familiarize the operator with the instrumentation, and confirm correct signal routing through the testbench before performing measurements on the target integrated circuit. Because the pattern generator produces waveforms according to its internal clock cycles and the timing analyzer acquires data at a sampling rate of 250 MHz, any timing skew or signal integrity issues would appear as timing offsets, pulse distortions, or inconsistencies between channels.

No observable skew, distortion, or channel mismatch was detected among the four acquired signals. These results confirm correct instrument configuration, proper electrical connectivity, and the integrity of the measurement probes.

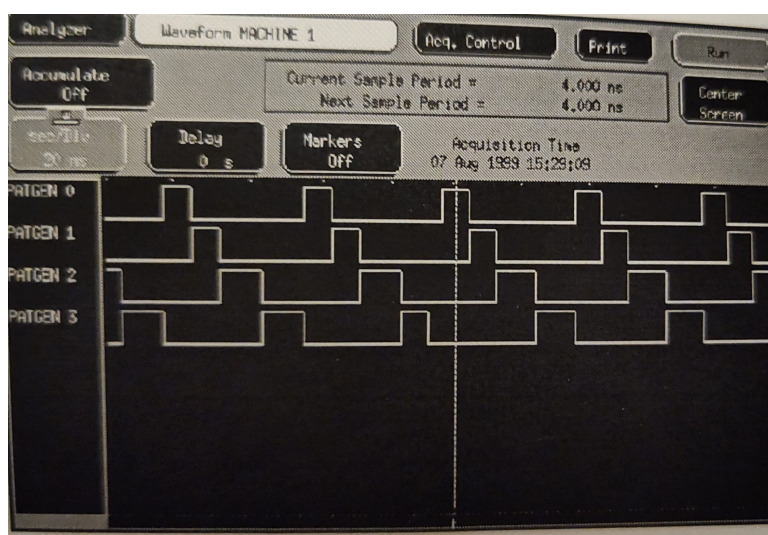


Figure 2.24: Timing analyzer waveform view displaying the captured multi-channel walking ones pattern generated by the independent output module.

2.1.5 File Management

The ability to connect the logic analyzer to an external computer extends its functionality beyond standalone operation by enabling configuration backups, automated instrument control, and advanced post-acquisition data analysis. The Agilent 1672G platform provides several data transfer and communication options to support these capabilities. Available external interfaces include GPIB, RS-232-C, and Ethernet LAN connectivity.

For local data storage and convenient transfer of measurement data to an MS-DOS compatible personal computer, the instrument incorporates an integrated 3.5-inch floppy disk drive capable of reading and writing floppy disks formatted with the MS-DOS file system.

The instrument's file management system supports storing and transferring measurement configurations, acquired waveform data, and full-screen images. These files can be archived on floppy disks and subsequently analyzed or restored on compatible systems.

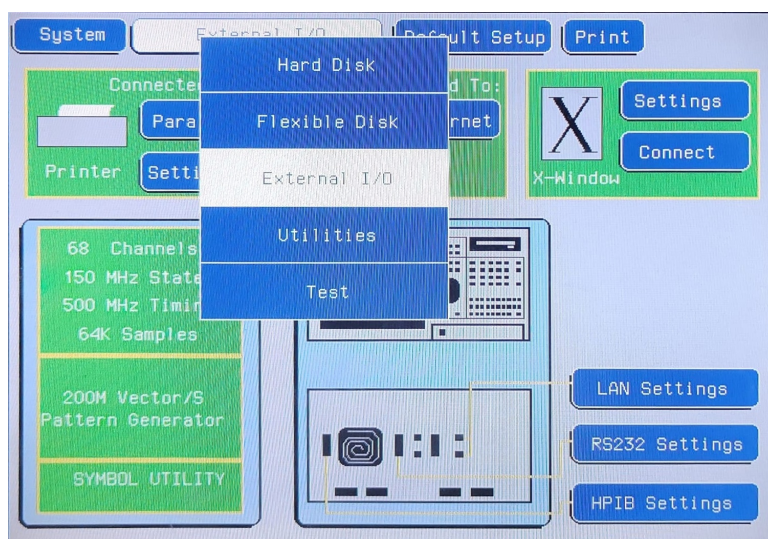


Figure 2.25: System menu interface demonstrating the disk operations and storage medium selection options.

2.1.5.1 Save and Load Configuration Routines

To support repeatable measurement procedures and the management of hardware configurations, the instrument allows configuration data to be saved to and restored from either the internal hard disk or the floppy disk drive. The standard procedures for storing and loading configuration files are outlined below:

1. **Saving a Configuration:** Access either the *System Hard Disk* or *System Flexible Disk* menu. Within the main *System* menu, set the action field to *Store*. Select *Analyzer*, *System*, or *All* as the configuration domain to determine which hardware settings will be saved. Specify the destination file using the *to file* field, optionally enter a file description, and press the physical *Execute* key to complete the operation.
2. **Loading a Configuration:** Access either the *System Hard Disk* or *System Flexible Disk* menu and set the action field within the main *System* menu to *Load*. Select the target module from the available options and specify the source file either by entering its name directly or by selecting it with the rotary control using the *from file* field. Press the *Execute* key to load the configuration. The instrument automatically filters compatible files according to the selected hardware domain. Analyzer configuration files are identified by the 167xan_config designation, whereas complete system configuration files use the 167x_cnfg designation.

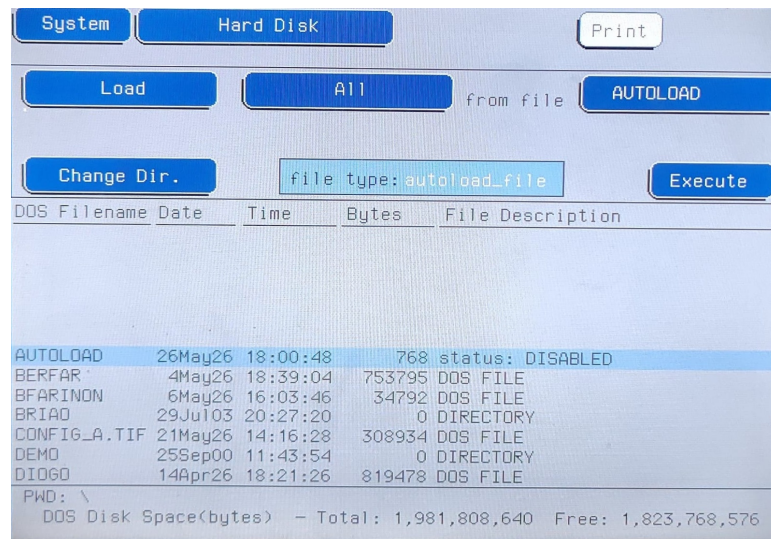


Figure 2.26: Configuration storage workspace configured for loading and storing instrument settings via the target file fields.

2.1.5.2 ASCII Trace Export and Tabular Data Extraction

When an acquisition contains textual data, such as a directory listing or the complete *State Tracking Matrix*, the information can be exported as an ASCII text file. Unlike a conventional screen capture, this method records the entire dataset rather than only the portion currently visible on the display. Consequently, all acquired entries stored within the acquisition buffer, including data located outside the visible viewing area, are preserved in the exported file.

To export the data, insert a floppy disk and display the desired trace or data listing on the screen. Press the physical *PRINT* key on the front panel to open the print menu, then select the *Print Disk* option from the lower menu bar. Within the resulting configuration panel, specify the destination file name, set the *Output Format* field to ASCII (ALL), and select *Flexible Disk* as the output destination. Press the *Execute* field to initiate the transfer. The instrument then writes the complete acquisition dataset to the floppy disk in ASCII format.

2.1.5.3 Educational Role in Pre-Validation Diagnostics

Consistent with the scope of this work, the file management capabilities described in this section were not employed during the formal data collection and physical characterization of the SoC-Wimed platform. Instead, the procedures for managing configuration files and exporting textual data were used primarily during the project's initial familiarization phase.

Understanding how the Agilent 1672G stores configuration and acquisition data, together with a general awareness of its operating system and file format constraints, provided valuable insight into the instrument's architecture and operational workflow. This knowledge

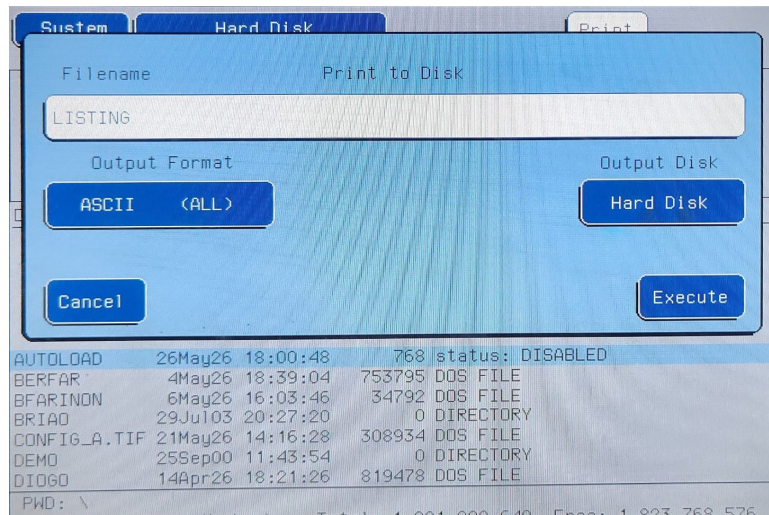


Figure 2.27: Print Disk submenu overlay configured to export complete trace spreadsheets in ASCII format.

enabled the students to become familiar with the analyzer interface and to perform routine instrument management tasks with confidence.

As a result, the students were able to focus on the subsequent validation activities without requiring additional training or automated support mechanisms. The experience gained during this preparatory phase contributed to the reliable execution of the formal measurement campaigns conducted on the target SoC device.

2.1.6 LAN Interface and FTP Transfer

Another feature of the Agilent 1672G platform is its integrated Ethernet LAN interface. This capability enables data transfer from the analyzer to an external computer without an intermediate storage medium, such as a floppy disk. Once a network connection has been established, the analyzer supports three remote operating environments: *FTP (File Transfer Protocol)*, *Telnet*, and *X Window*.

The *Telnet* and *X Window* environments provide remote access to the instrument and facilitate activities such as instrument configuration, software development, and remote operation. However, these environments do not offer the same level of convenience for file exchange as dedicated file transfer services. For this reason, *FTP* serves as the primary mechanism for transferring files between the analyzer and external computer systems.

Using *FTP*, configuration files, acquisition data, and other stored information can be transferred directly between the analyzer and host computers. This capability simplifies data management and eliminates the need for removable storage media during routine laboratory operation.

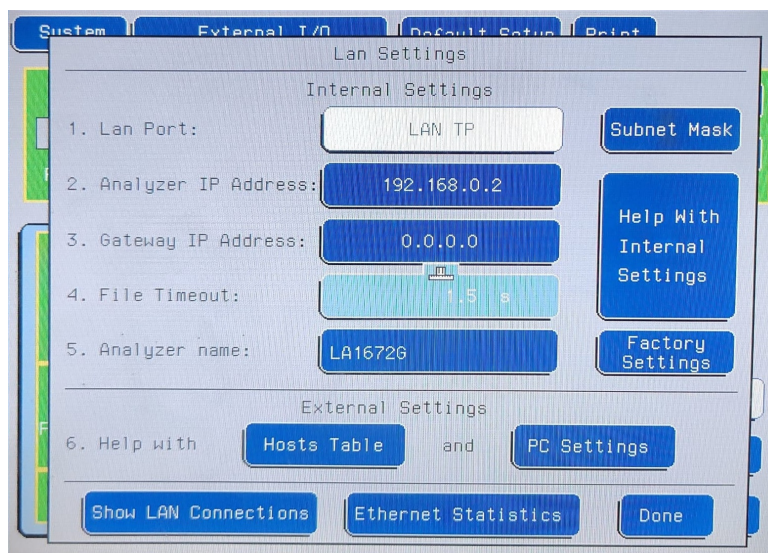
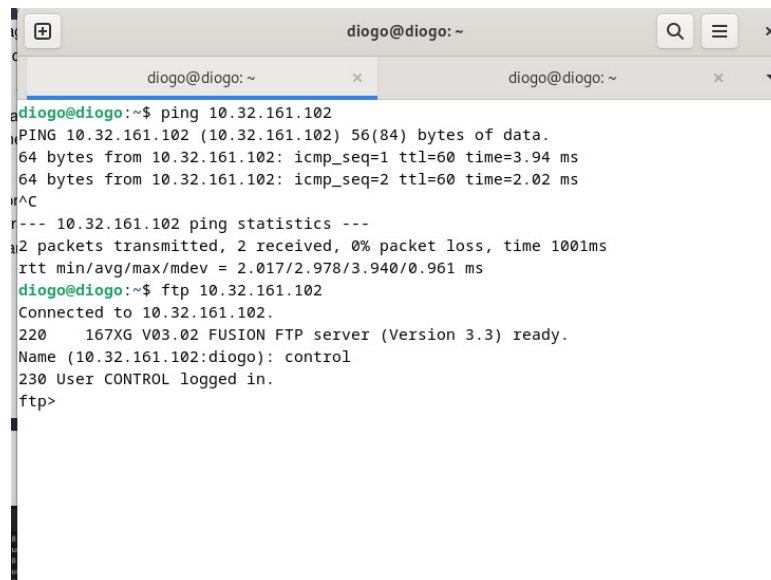


Figure 2.28: LAN configuration sub-panel displaying the network parameter assignments and connection pathways.

2.1.6.1 Network File Transfer Procedures via FTP

To transfer files using the FTP subsystem, the following procedure can be performed from a remote host computer:

1. Verify that FTP services are available on the host computer and establish a physical Ethernet connection between the host system and the logic analyzer.
2. Initiate an FTP session by connecting to the logic analyzer using its assigned IP address or hostname.
3. At the authentication prompt, enter the appropriate access credential according to the intended operation. The data login provides read access to stored files, whereas the control login permits file management operations, including uploading files to the analyzer and modifying its stored data.
4. Before transferring files, switch the FTP session from ASCII mode to binary mode by issuing the command `BIN`. Binary transfer mode ensures that non-text files, such as compressed trace files and other binary data, are transferred without modification or corruption.
5. Navigate the analyzer's directory structure using the standard FTP file management commands. Files can then be transferred using conventional FTP operations, such as `get` to download files from the analyzer and `put` to upload files from the host computer to the instrument, as shown in Figure 2.29.



```

diogo@diogo: ~
diogo@diogo: ~
diogo@diogo:~$ ping 10.32.161.102
PING 10.32.161.102 (10.32.161.102) 56(84) bytes of data.
64 bytes from 10.32.161.102: icmp_seq=1 ttl=60 time=3.94 ms
64 bytes from 10.32.161.102: icmp_seq=2 ttl=60 time=2.02 ms
^C
--- 10.32.161.102 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 2.017/2.978/3.940/0.961 ms
diogo@diogo:~$ ftp 10.32.161.102
Connected to 10.32.161.102.
220 167XG V03.02 FUSION FTP server (Version 3.3) ready.
Name (10.32.161.102:diogo): control
230 User CONTROL logged in.
ftp>

```

Figure 2.29: Terminal environment showing the active host-to-instrument FTP communication session and data format initialization.

2.1.6.2 Interface with Automated Protocol Analysis Framework

In contrast to the local storage mechanisms described previously, which were not used during the formal characterization campaign, the Ethernet network interface and FTP transfer capabilities were integral components of the SoC-WiMed validation workflow. During testing, raw acquisition data stored in the analyzer's internal memory was retrieved and transferred over a LAN connection to a host computer for subsequent processing.

This transfer pipeline was designed to supply data directly to an automated validation framework implemented in Python, which is described in detail in the following section. By enabling direct network-based transfer of acquisition data, the workflow eliminated the need for intermediate removable storage media and significantly reduced the overhead associated with manual data handling.

As a result, captured trace data could be delivered from the logic analyzer to the validation environment, where automated analysis routines processed the acquired bitstreams and compared them with the expected system behavior.

2.2 Case Studies

2.2.1 SPI Protocol Debugging

The Serial Peripheral Interface (SPI) is a synchronous communication protocol commonly used in embedded systems. Communication is performed through a clock signal

and dedicated data lines. In the implementation used in this work, the signals observed were SCLK, MOSI, and CS.

To validate the logic analyzer's use for SPI communications, a simple SPI transmitter was implemented on the FPGA. The module periodically transmitted a predefined sequence of characters, allowing the captured data to be compared with the expected values.

The Agilent 1672G digital probes were connected to the SPI signals. The CS signal was used as the trigger source since it indicates the beginning of a transaction. Whenever CS changed to its active state, the analyzer started the acquisition process. This reduced the amount of idle data stored and made it easier to locate individual transmissions.

Figure 2.30 shows an example of an SPI transmission captured using the Agilent 1672G. The clock signal (SCLK), transmitted data (MOSI), and chip select signal (CS) can be observed simultaneously. The activation of the CS signal marks the beginning of each SPI transaction and was used as the trigger condition during signal acquisition.

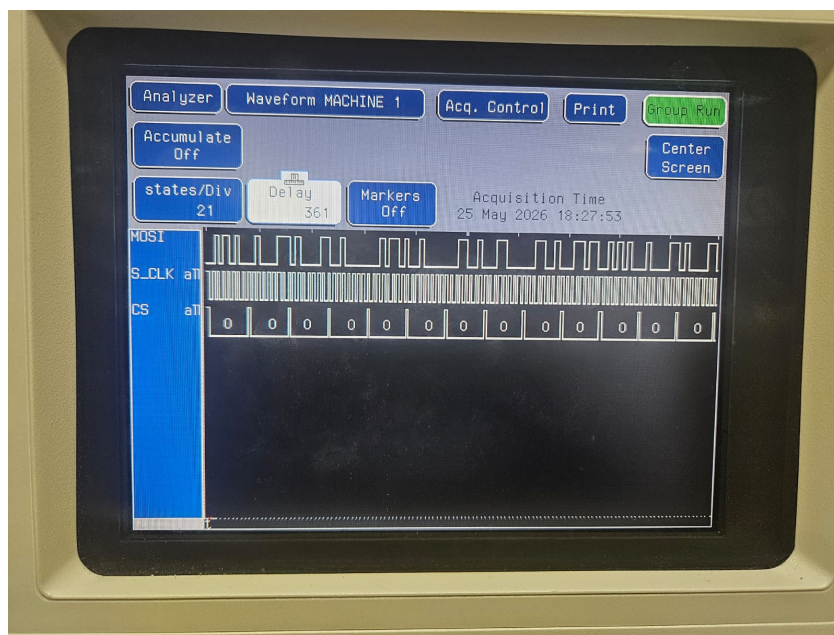


Figure 2.30: SPI transmission captured using the Agilent 1672G. The MOSI, SCLK, and CS signals can be observed, with a separate CS activation preceding each transmitted byte.

In the implemented design, each transmitted character was associated with an independent SPI transaction. As a result, the CS signal was asserted once for every transmitted byte. This behavior can be observed directly in Figure 2.30, where multiple CS activations delimit consecutive transmissions. The clear separation between transactions simplified the identification of individual characters and later served as a reference during the analysis of the captured data.

The captured values matched the expected transmission sequence, confirming the correct operation of the SPI module and the analyzer configuration used during the experiment.

2.2.2 UART Protocol Debugging

The Universal Asynchronous Receiver Transmitter (UART) is a serial communication protocol that does not use a shared clock signal between transmitter and receiver. Instead, communication is based on a previously configured baud rate. A UART frame is composed of a start bit, a sequence of data bits, and one or more stop bits.

For this experiment, a UART transmitter was implemented in the FPGA and configured to periodically send a predefined message. Since the transmitted content was known beforehand, the captured data could be compared directly with the expected message.

The TX signal was connected to one of the digital channels of the Agilent 1672G. A falling-edge trigger was configured on this signal because the UART line remains at logic high while idle and transitions to logic low at the beginning of a transmission. This allowed the analyzer to start acquisition near the beginning of each frame.

Initial captures were performed using Timing Analysis mode. In this configuration, the analyzer samples the signal continuously at a fixed frequency. Since the sampling frequency is much higher than the UART baud rate, a large number of samples are generated for each transmitted bit. While this provides a detailed view of the waveform, it also consumes memory quickly during long acquisitions.

Figure 2.31 presents an example of a UART transmission captured using State Analysis mode. The upper waveform corresponds to the UART TX signal, while the lower waveform represents the external clock provided by the FPGA and used only for acquisition purposes.

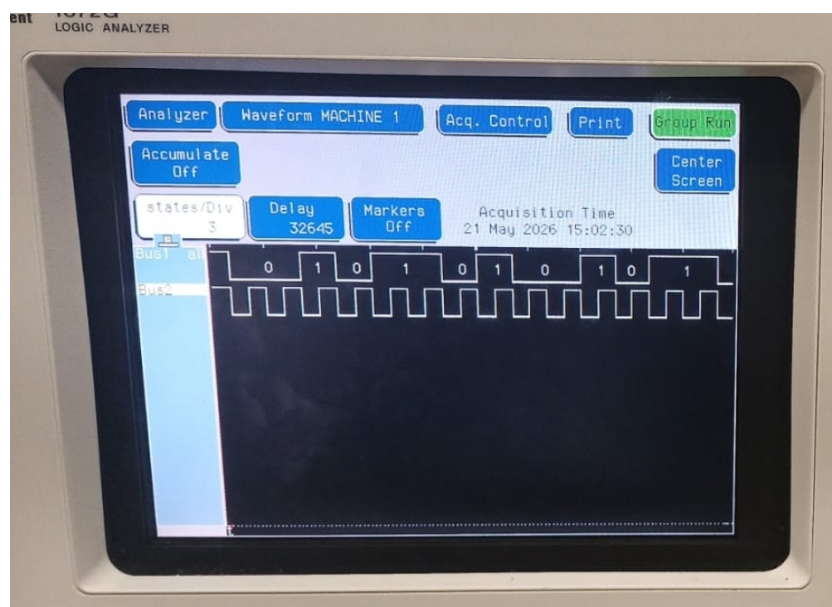


Figure 2.31: UART transmission captured in State Analysis mode.

Unlike SPI, UART communication does not require a shared clock signal between transmitter and receiver. Synchronization is achieved through the configured baud rate and the detection of the start bit. Therefore, the clock shown in Figure 2.31 is not part of the UART protocol itself. Its only purpose is to provide a sampling reference for the logic analyzer.

Using State Analysis mode allowed the analyzer to acquire data only on clock events rather than continuously sampling the signal. This significantly reduced the number of stored samples while preserving the information required to reconstruct the transmitted bytes. By analyzing the captured sequence, it was possible to identify the transmitted bits and verify the UART interface's correct operation.

The use of an externally generated acquisition clock was useful for long UART captures, as it reduced memory consumption compared to Timing Analysis mode while still preserving the information required for protocol decoding.

2.2.3 Digital Signal Injection

In addition to signal acquisition capabilities, the Agilent 1672G may be equipped with a Pattern Generator module, enabling it to generate programmable digital signals. Unlike conventional signal generators that focus on analog waveforms, the Pattern Generator is designed to create digital sequences composed of logic states, transitions, and timing relationships defined by the user.

The generated patterns are described through a sequence of vectors, where each vector defines the logic levels applied to the output channels during a given clock period. The instrument supports loops, repetition of signal sequences, and programmable execution flow, enabling the generation of complex digital stimuli. This capability can be used to emulate communication interfaces, control signals, and other digital events without requiring additional hardware.

As part of the study of the Agilent 1672G features, a simple experiment was performed using the Pattern Generator module. A predefined sequence of logic values was programmed through the Pattern Generator interface, as shown in Figure 2.32. The generated output was then connected to one of the logic analyzer's input channels, allowing the instrument's signal to be captured and analyzed by the same Agilent 1672G unit.

This configuration created a closed-loop experiment in which the Pattern Generator served as the signal source, while the logic analyzer served as the acquisition instrument. The objective was not to validate an external circuit, but rather to understand the configuration process and verify the relationship between the programmed sequence and the acquired waveform.

Figure 2.32 shows the sequence programmed into the Pattern Generator. Figure 2.33 presents the corresponding waveform captured by the logic analyzer after connecting the generator output to an acquisition channel. The captured transitions matched the programmed pattern, confirming the correct operation of both the Pattern Generator and the acquisition subsystem.

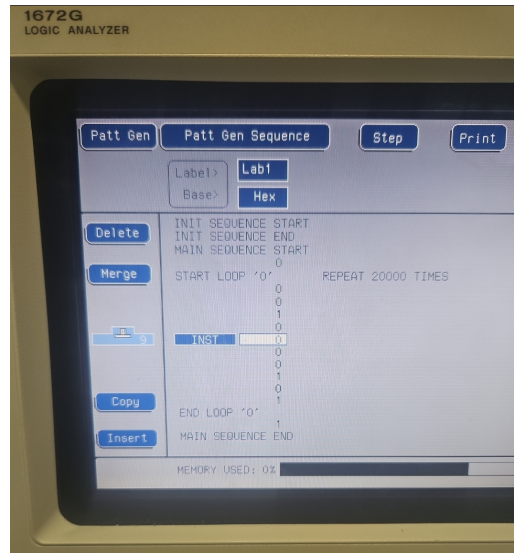


Figure 2.32: Pattern Generator sequence programmed in the Agilent 1672G. The displayed vector sequence was used to generate a predefined digital waveform.

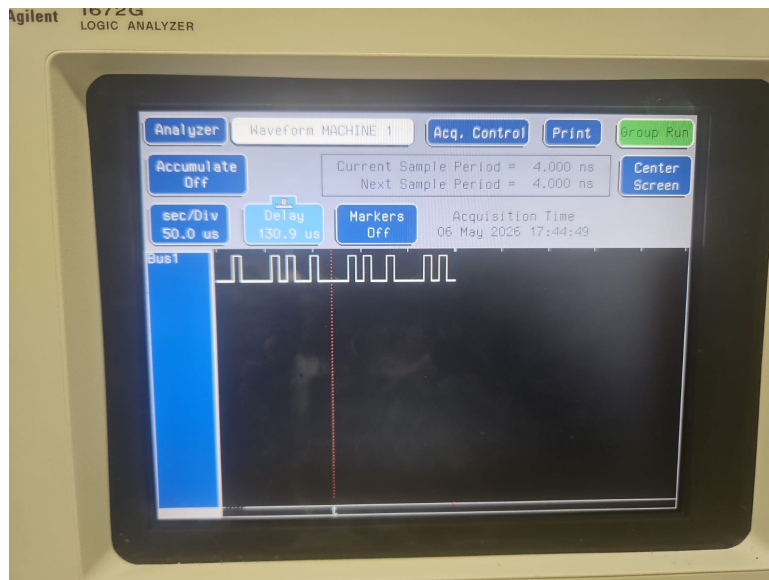


Figure 2.33: Waveform captured by the Agilent 1672G after connecting the Pattern Generator output to a logic analyzer input channel.

Although the Pattern Generator was not used during the debugging activities described later in this work, the experiment provided practical experience with the signal generation resources available in the Agilent 1672G. It also helped establish a better understanding of the relationship between generated and acquired digital signals, which is important when using instrumentation for debugging and validation.

2.2.4 Python Support Software

During the experimental campaign, the Agilent 1672G logic analyzer was configured to export captured acquisitions in state-listing format. Although this representation preserves all information required for signal analysis, manual interpretation becomes increasingly impractical as the length of the captured binary sequences grows. To address this limitation, a collection of Python scripts was developed to automate the decoding and validation process.

The Python support software developed during this work is available as an open-source repository to facilitate reproducibility and future extensions of the proposed methodology and can be accessed at our [GitHub repository](#).

Figure 2.34 summarizes the workflow adopted by the Python support software. After reading the state-listing file exported by the logic analyzer, the software extracts the captured logic states and forwards them to protocol-specific decoding routines. Depending on the selected protocol, the corresponding decoder reconstructs the transmitted bytes and converts them into hexadecimal and ASCII representations.

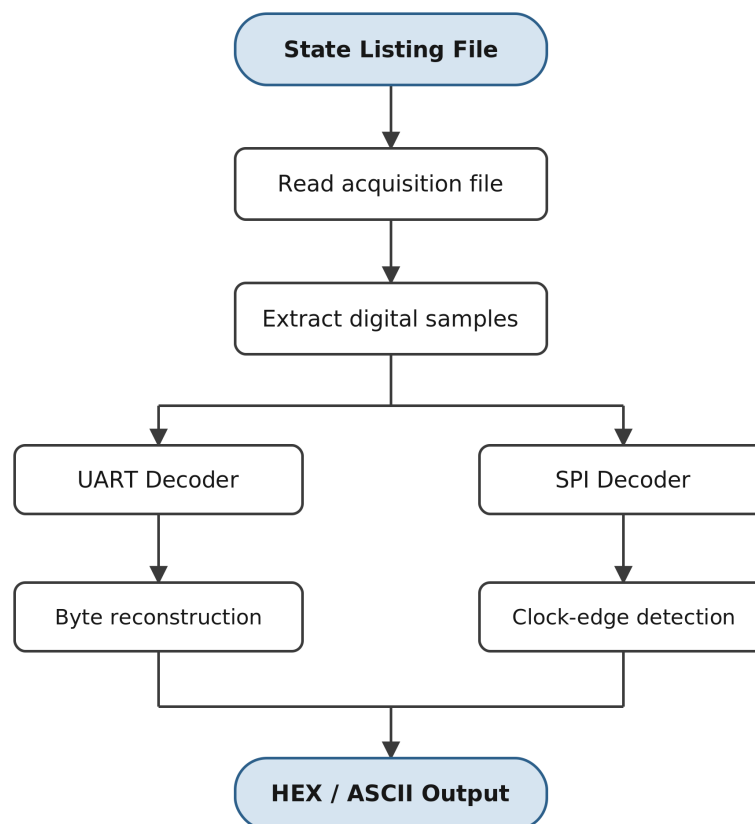


Figure 2.34: Workflow of the Python support software.

Although both UART and SPI captures were processed using the same software framework, the decoding procedures differ according to the characteristics of each protocol.

For UART communication, byte reconstruction is based on timing information. After detecting the start bit, the software uses the configured baud rate and acquisition sampling frequency to calculate the number of captured samples per bit period. The decoder then samples the signal near the center of each bit interval in order to maximize reliability and reduce the influence of signal transitions. Since UART transmissions are typically performed using a least-significant-bit-first (LSB-first) format, the recovered bits must be assembled according to their transmission order before reconstructing the transmitted byte. The resulting byte stream is converted to hexadecimal and ASCII, as shown in Listing 2.1, allowing messages transmitted by the FPGA to be displayed directly in a human-readable format.

```
MACHINE 1 - State Listing
```

```
Label > MOSI CS S_CLK
```

```
Base > Hex He Hex
```

```
96      0    1    1
```

```
97      0    0    0
```

```
98      0    0    1
```

```
99      1    0    0
```

```
100     1    0    1
```

```
...
```

```
141     1    0    0
```

```
142     1    0    1
```

```
143     1    0    0
```

Listing 2.1: Excerpt of the state-listing file exported by the Agilent 1672G logic analyzer and used as input for the Python decoding software.

SPI communication employs a different strategy because data transmission is synchronized by a dedicated clock signal. Instead of relying on calculated sampling instants, the decoder monitors clock transitions and samples the MOSI signal at the appropriate clock edge. The Chip Select (CS) signal is used to delimit individual transactions and determine byte boundaries. In the SPI implementation evaluated in this work, each transmitted character was encapsulated in an individual SPI transaction. Experimental analysis showed that data was transmitted using an MSB-first format and sampled on the falling edge of the serial clock. The recovered bits are grouped into bytes and converted to hexadecimal and ASCII values, allowing direct interpretation of the transmitted message.

Several versions of the software were developed throughout the project. Initial implementations focused on extracting binary values from acquisition files and supporting manual verification of captured data. Subsequent revisions introduced automated bit grouping,

byte reconstruction, ASCII conversion, protocol-aware decoding, and support for multiple acquisition configurations. These enhancements significantly reduced the effort required to analyze logic analyzer captures and improved the efficiency and reliability of UART and SPI communication validation.

Figure 2.35 illustrates the SPI analysis process performed by the developed software. The logic states exported by the analyzer were reconstructed into a waveform representation and subsequently decoded into hexadecimal and ASCII values. In the illustrated example, the transmitted message was successfully reconstructed as “TCC - 2026”, confirming the correct operation of the SPI communication interface and validating the effectiveness of the proposed methodology.

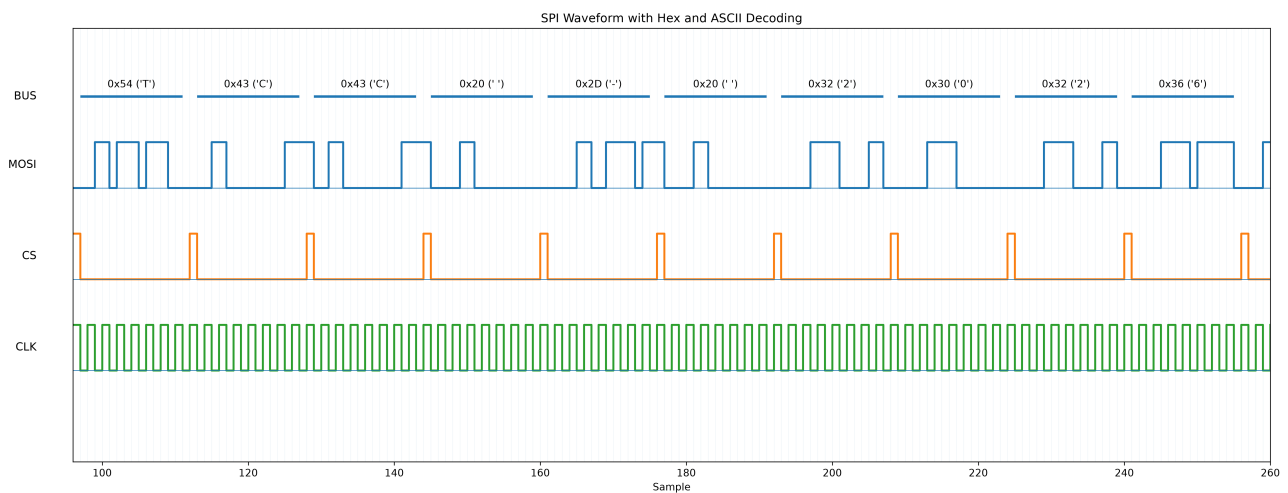


Figure 2.35: SPI waveform reconstructed by the Python support software, including hexadecimal and ASCII decoding of the transmitted data.

3. SOC-WIMED PLATFORM

This chapter presents an overview of the SoC-WiMed platform used throughout this work. The discussion focuses on the architectural components and communication interfaces that are most relevant to the debugging activities performed during the experimental validation process.

The SoC-WiMed is a RISC-V based System-on-Chip developed within the WiMed project. The platform integrates processing, memory resources, communication peripherals, interrupt management modules, and support infrastructure required for embedded software execution. One of its main characteristics is the adoption of an FPGA-based prototyping flow before ASIC implementation, allowing hardware and software components to be validated under realistic operating conditions.

From the perspective of this work, FPGA prototyping provides an opportunity to observe the behavior of the system while it executes real applications. Although simulation remains an important validation tool, execution on physical hardware allows communication interfaces, synchronization mechanisms, and interactions between hardware and software to be evaluated under conditions that are closer to the final system.

Because many development and debugging tasks depend on communication between the FPGA and a host computer, particular attention is given to the UART interface. This communication channel is used during software loading, system monitoring, debugging activities, and data exchange throughout the development process. For this reason, UART communication was selected as the initial target of the instrumentation methodology presented in this work.

The following section presents an overview of the SoC-WiMed architecture and highlights the modules most relevant to the debugging activities described later in this work.

3.1 SoC-WiMed Overview

One of the main objectives of the platform is to support an FPGA-to-ASIC development flow. In this approach, the system is first implemented and validated on an FPGA, where functionality can be verified under real operating conditions. Once the design reaches a sufficient level of maturity, the same architecture can be adapted for ASIC implementation. This development strategy reduces integration risks and allows problems to be identified before circuit fabrication.

Figure 3.1 presents the architecture of the RS5-SoC. It is important to highlight that all modules were developed in-house, without third-party components. There are five groups of modules:

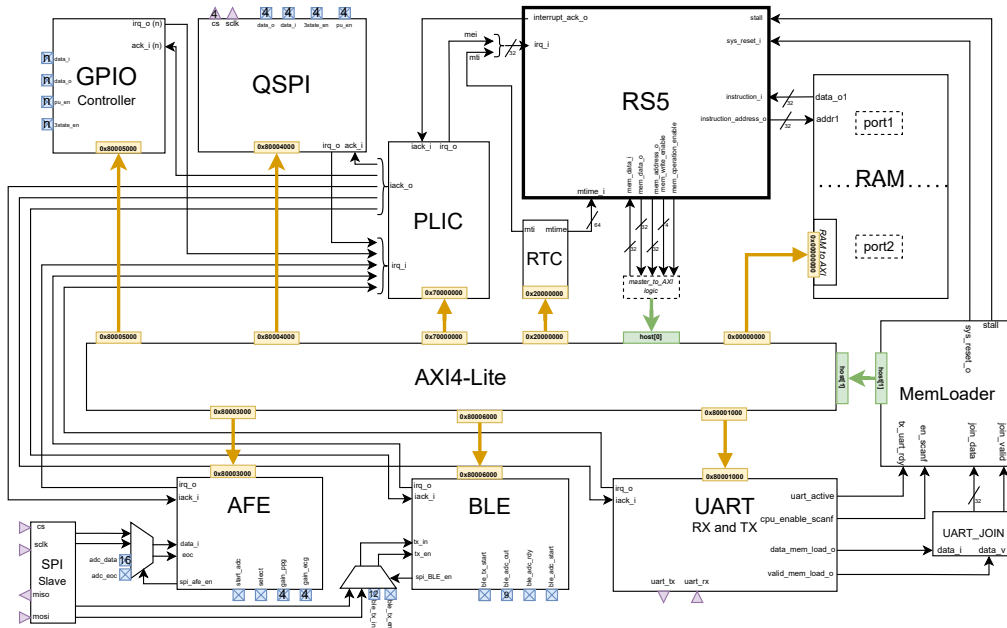


Figure 3.1: SoC-WiMed block diagram. In the AXI4-Lite bus, green boxes represent master interfaces, and yellow boxes represent slave interfaces.

1. **Computing subsystem:** includes the RS5 processor [Nunes et al. \[2024\]](#), a dual-port 64 KB RAM, a real-time clock (RTC), and a Platform-Level Interrupt Controller (PLIC);
2. **MCU peripherals:** GPIO, QSPI, SPI, and UART modules;
3. **WiMed SoC-related peripherals:** analog front-end (AFE), which receives data from ADCs, and the BLE module, which is the interface with the Bluetooth block;
4. **Multi-master AXI4-Lite bus** [ARM \[2025\]](#): the processor and the MemLoader can act as bus masters, with the MemLoader assigned a higher priority;
5. **MemLoader:** receives commands from an external host processor via the UART module.

The MCU peripherals enable communication with external devices and support experimental validation activities. From the perspective of this work, the UART interface is particularly relevant because it serves as the primary communication channel between the FPGA implementation of the SoC-WiMed and a host computer.

Another important component is the MemLoader module. This module receives commands via the UART interface and enables software applications to be loaded into the SoC's internal memory. As a result, UART communication is involved not only in debugging but also in the platform's normal development workflow.

Because UART is used during program loading, software execution, system monitoring, and debugging, its correct operation is essential for the development process. Communication failures may prevent software from being loaded correctly, make debugging information unavailable, or compromise the interaction between the host computer and the FPGA. For these reasons, UART communication was selected as the initial target of the debugging methodology presented in this work.

4. DEBUGGING THE SOC-WIMED

This chapter presents the application of the instrumentation methodology developed in this work to the SoC-WiMed platform. The Agilent 1672G logic analyzer was used to observe, capture, and analyze UART communication associated with software execution and system interactions.

UART communication plays a fundamental role in the SoC-WiMed architecture. In addition to providing a communication channel between the platform and a host computer, the interface is used by the MemLoader module, which receives binary program data through UART, stores it in memory, and subsequently releases the RS5 processor for program execution. Consequently, communication failures on this interface can compromise both software loading and system validation procedures.

The experiments presented in this chapter focus on observing and analyzing UART signals generated during the operation of the SoC-WiMed platform, which is implemented on an FPGA. By capturing communication signals directly from the FPGA pins, it was possible to verify correct protocol operation, analyze transmitted data, and assess the effectiveness of external instrumentation as a debugging and validation tool for the platform.

4.1 Experimental Setup

The experiments were conducted using the SoC-WiMed platform implemented on a Nexys A7 FPGA development board. A simple bare-metal application running on the RS5 processor was used to generate UART traffic during system operation.

For this purpose, a *Fibonacci* application was loaded into the platform through the MemLoader infrastructure. The application transmits textual messages through the UART peripheral using standard `printf()` calls, enabling complete software-generated communications to be observed directly at the hardware level. The transmitted messages include a prompt requesting the number of iterations and the corresponding Fibonacci value computed by the processor.

To make the UART signal accessible to the logic analyzer, the FPGA top-level design was modified to route the UART transmit signal to an external *Pmod* connector while preserving its original connection to the onboard USB-to-UART interface. This functionality was implemented by assigning the internal UART transmit signal to an additional output port connected to the selected *Pmod* header.

An auxiliary clock signal was also generated within the FPGA using a frequency divider. The divider produced a 1 MHz clock, which was exported through a second *Pmod* pin and used as the external sampling reference for acquisitions performed in State Anal-

ysis mode. The Agilent 1672G was configured to acquire samples on both the rising and falling edges of this clock, resulting in an effective sampling frequency of 2 MHz. This approach provided sufficient temporal resolution for reliable UART decoding while significantly reducing memory consumption compared to Timing Analysis mode.

The FPGA pin assignments were specified in the XDC constraints file. The UART transmit signal was routed to pin D4 of the Nexys A7 *Pmod* interface, and the external acquisition clock to pin D18. Both signals were connected to the Agilent 1672G logic analyzer through digital probe connections.

This configuration allowed the UART transmission generated by the RS5 processor to be observed simultaneously through both the serial terminal and the logic analyzer. Consequently, the software's output could be directly correlated with the captured digital waveforms, facilitating the validation of both the communication interface and the acquisition methodology.

4.2 UART Interface Validation

After configuring the acquisition setup, UART transmissions from the SoC-WiMed platform were captured and analyzed with the Agilent 1672G logic analyzer.

A falling-edge trigger was configured on the UART transmit (*TX*) signal. Because the UART line remains at logic high during idle periods and transitions to logic low at the beginning of the start bit, this trigger configuration allowed the analyzer to initiate acquisition near the start of each transmission.

Figure 4.1 presents an overview of the UART activity generated by the SoC-WiMed platform during software execution. Although individual UART frames cannot be readily distinguished at this time scale, the capture clearly demonstrates periodic data transmission through the communication interface. The observed activity confirms that the processor is executing the application and generating the expected serial output.

The captured waveform was used to evaluate the timing characteristics of the UART interface. The duration of a single bit period was measured and compared with the value corresponding to a baud rate of 115200 bit/s. For this configuration, the expected bit period is approximately 8.68 μ s. The measured bit duration agreed with the expected value, indicating that the UART timing parameters implemented in the SoC-WiMed platform were operating correctly. This result confirms that the clock generation and baud rate configuration used by the UART peripheral produced the expected transmission timing.

A detailed examination of the captured waveform enabled the identification of the individual components of a UART frame, including the start bit, data field, and stop bit. The

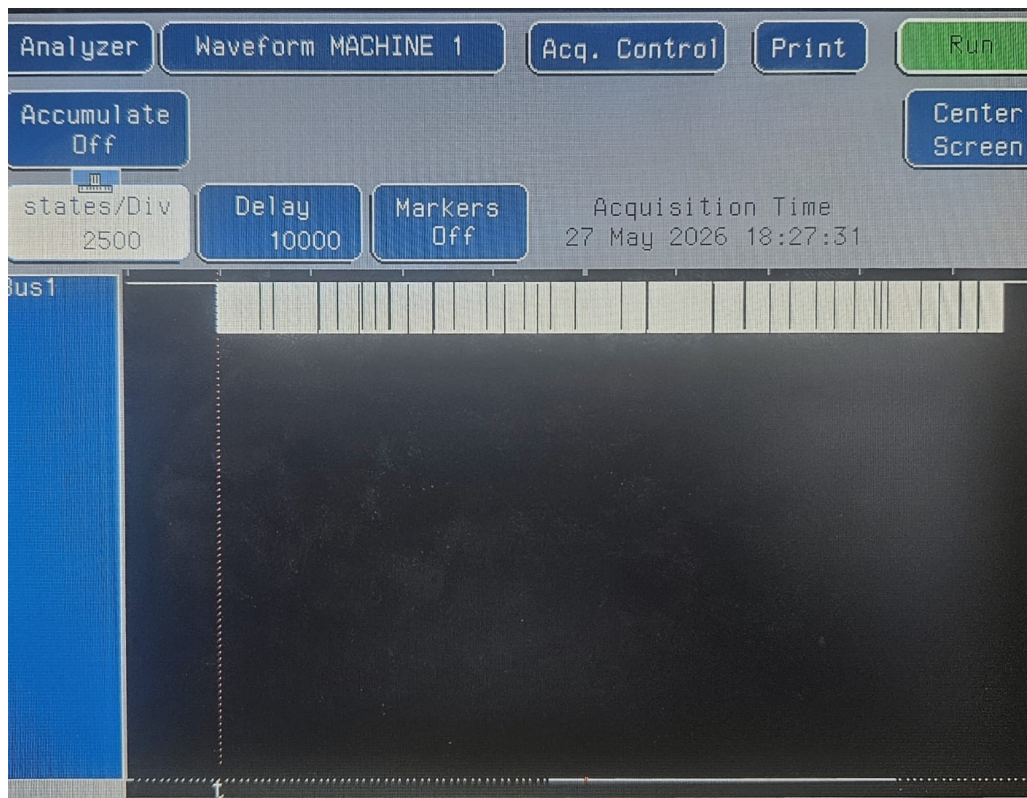


Figure 4.1: Overview of the UART transmission generated by the SoC-WiMed platform and captured using the Agilent 1672G logic analyzer.

observed frame structure was consistent with the expected UART protocol format, providing additional evidence of successful communication.

Figure 4.2 presents a view of one transmitted frame. The structure is consistent with the expected UART format, allowing the individual bits to be identified and the transmitted byte to be reconstructed.

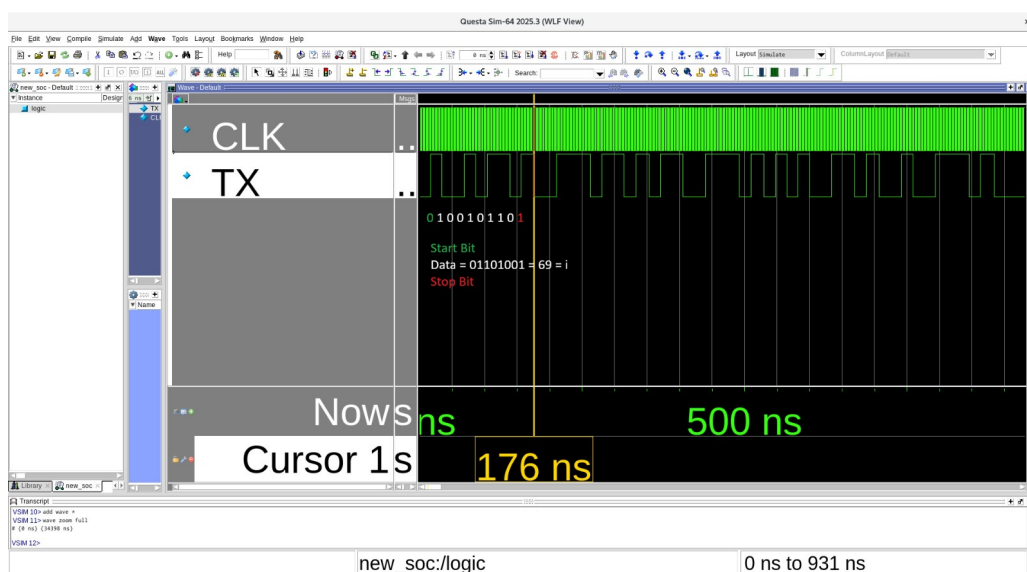


Figure 4.2: Detailed view of a UART frame transmitted by the SoC-WiMed platform, showing the start bit, data field, and stop bit used during character transmission.

To simplify the analysis, the captured information was also processed using the Python-based software developed during this work. The script reads the files exported by the logic analyzer, calculates the number of samples per bit, reconstructs the transmitted bytes, and converts the recovered data to hexadecimal and ASCII.

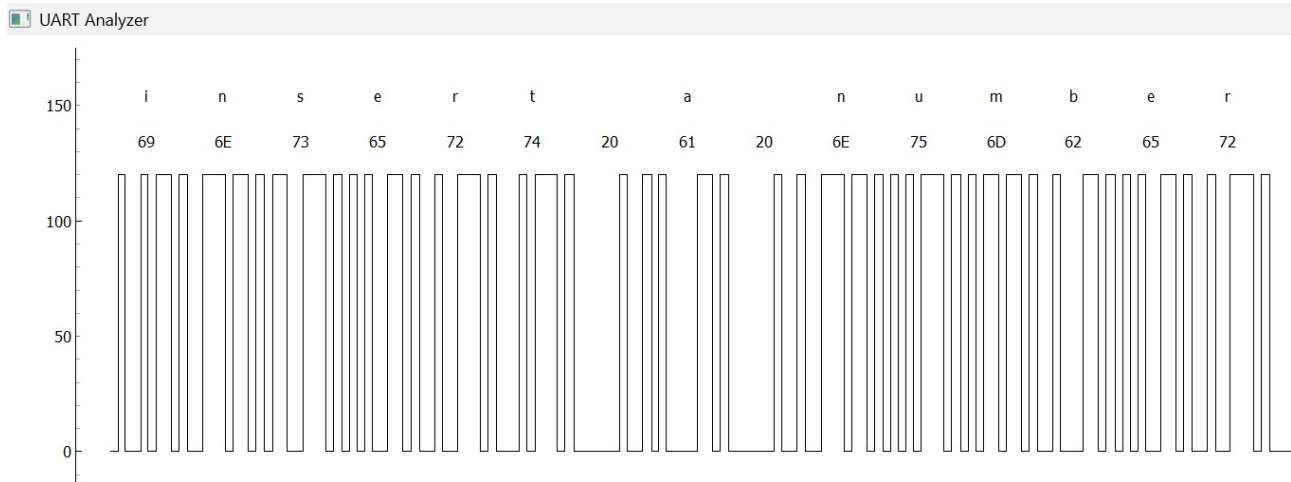


Figure 4.3: UART waveform after protocol decoding, showing the reconstructed data values associated with the captured transmission.

Figure 4.3 shows the result of the decoding process applied to the acquired waveform. The decoded values correspond to the information generated by the application running on the RS5 processor, confirming that the communication was correctly captured and interpreted.

The decoder determines the number of samples corresponding to a single UART bit from the ratio between the effective acquisition sampling rate and the configured baud rate. The FPGA provides a 1 MHz external sampling clock to the Agilent 1672G. During State Analysis, however, the analyzer records one sample on each clock edge, resulting in two captured samples for every clock period. Consequently, although the external reference clock operates at 1 MHz, the exported state-listing contains an effective sampling rate of 2 MSamples/s. Therefore, the number of samples corresponding to one UART bit is given by

$$\text{Samples/bit} = \frac{2\,000\,000}{115\,200} \approx 17.36.$$

This value is used by the decoder to determine the sampling position near the center of each bit interval, where the logic level is least affected by signal transitions, improving the robustness of the reconstruction process.

Considering the effective acquisition rate of 2 MHz, the number of samples available for each transmitted bit decreases proportionally as the UART baud rate increases. For example, acquisitions at 230400 bit/s and 460800 bit/s would provide approximately 8.68 and 4.34 samples per bit, respectively, while 921600 bit/s would reduce this value to ap-

proximately 2.17 samples per bit. Although these higher baud rates were not experimentally evaluated in this work, the proposed methodology can be applied provided that a sufficient number of samples per bit is maintained for reliable decoding.

Listing 4.1 presents an example of the software output after processing a UART acquisition. The program calculates the sampling ratio, reconstructs the transmitted bytes, and converts them into a human-readable message. The recovered text matches the output produced by the Fibonacci application, providing additional verification of both the UART communication and the decoding methodology.

```
Samples/bit = 17.36

Message HEX:
69 6E 73 65 72 74 20 61 20 6E 75 6D 62 65 72 ...

Message ASCII:
insert a number to set the amount of iterations
for the fibonnacci sequence:?
The fibonnacci value of 8 iterations is 21
```

Listing 4.1: Output generated by the Python UART decoding software.

The results obtained from the waveform inspection and software-assisted decoding confirmed the correct operation of the UART interface implemented in the SoC-WiMed platform. The measured timing characteristics were consistent with the configured baud rate, the observed frame structure matched the expected UART format, and the recovered messages corresponded to the output generated by the application executing on the RS5 processor. These observations demonstrate the effectiveness of the Agilent 1672G as a tool for protocol validation and debugging in FPGA-based SoC platforms.

5. CONCLUSION AND FUTURE WORK

The development of FPGA-based systems requires more than functional verification through simulation. Although simulation remains an essential step of the design process, system execution on physical hardware frequently exposes integration, communication, and timing issues that are difficult to identify during earlier development phases. Consequently, signal instrumentation represents an important resource for system validation and debugging.

This work investigated the use of the Agilent 1672G logic analyzer to validate and analyze digital communication interfaces in FPGA-based systems. A series of experiments involving UART and SPI communications was conducted to develop practical expertise in signal acquisition, trigger configuration, timing analysis, state analysis, and protocol interpretation. Additional laboratory instruments, including a multimeter, an oscilloscope, and a signal generator, were used to support the preparation and verification of the experimental environment.

The initial experimental activities focused on simple UART and SPI implementations developed on an FPGA platform. These experiments established the signal acquisition procedures used throughout the project and provided a controlled environment for learning to operate and configure the logic analyzer. In parallel, Python-based software tools were developed to automate the interpretation of captured data, enabling the reconstruction and analysis of UART and SPI transmissions with reduced manual effort.

The knowledge acquired during these preliminary studies was subsequently applied to the SoC-WiMed platform. UART communications generated by software executing on the RS5 processor were captured directly from FPGA pins and analyzed using the Agilent 1672G. The resulting measurements enabled verification of the UART frame structure, bit timing, and transmitted data, demonstrating the effectiveness of external instrumentation in observing and validating communication events during system operation.

An important outcome of this work was the development of a practical tutorial on using the Agilent 1672G for digital system debugging. The tutorial documents equipment configuration procedures, probe connections, trigger setup, timing and state analysis techniques, UART and SPI case studies, and the Python support tools developed during the project. By consolidating this information into a structured reference, the work provides a reproducible methodology that future students and researchers can adopt, reducing reliance on informal knowledge transfer and facilitating validation and debugging in FPGA-based systems.

Future work may extend the proposed methodology to other peripherals available in the SoC-WiMed platform, including GPIO, QSPI, and interrupt-related mechanisms. The integration of logic analyzer acquisitions with automated testing frameworks may also be

investigated to further reduce manual effort in system validation. Another direction involves comparing external instrumentation techniques with FPGA-embedded debugging resources, such as Integrated Logic Analyzers (ILAs), to evaluate the advantages and limitations of each approach. Finally, the procedures and tools developed throughout this work may be adapted to other FPGA-based platforms and applied to subsequent stages of FPGA-to-ASIC development flows.

REFERENCES

- Agilent Technologies (2002). *Agilent Technologies 1670G Series Logic Analyzers User's Guide*. <https://www.calstatela.edu/sites/default/files/1672g.pdf>.
- Agilent Technologies (2003). *Agilent Technologies 1670G Series Logic Analyzers Service Guide*. http://bitsavers.org/test_equipment/hp/166x_167x/01670-97023_1670G_Service_Guide_200311.pdf.
- Agilent Technologies (2005). Agilent 1670G Series Benchtop Logic Analyzers — Technical Data. Product Data Sheet, Pub. No. 5968-6421E. <https://www.keysight.com/us/en/assets/7018-08998/data-sheets-archived/5968-6421.pdf>.
- AMD/Xilinx (2023). *Vivado Design Suite User Guide: Programming and Debugging*. Xilinx, Inc. UG908.
- ARM (2025). AXI4 and AXI4-Lite interfaces. <https://developer.arm.com/documentation/dui0534/b/Parameter-Descriptions/Interface/AXI4-and-AXI4-Lite-interfaces/>.
- Bushnell, M. L. and Agrawal, V. D. (2000). *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. Kluwer Academic Publishers, Boston.
- Nunes, W., Dal Zotto, A., Borges, C., and Moraes, F. G. (2024). RS5: An Integrated Hardware and Software Ecosystem for RISC-V Embedded Systems. In *LASCAS*, pages 1–5. <https://doi.org/10.1109/LASCAS60203.2024.10506171>.
- Rippel, J. (2024). *Logic Analyzers in Practice*. Elektor. <https://www.elektor.com/logic-analyzers-in-practice>.
- Tektronix (2011). *XYZs of Oscilloscopes: Primer*. Tektronix, Inc. <https://www.tek.com/en/documents/primer/xyzs-oscilloscopes-primer>.
- Tektronix, Inc. (2009). *Logic Analyzer Fundamentals*. <https://engineering.case.edu/sites/default/files/Tektronix%20Logic%20Analyzer%20-%20Fundamentals.pdf>.