Jül-4402

Zentralinstitut für Engineering, Elektronik und Analytik (ZEA)
Systeme der Elektronik (ZEA-2)

Implementation of a JTAG Verification Environment for a Complex Highly Integrated Real SoC Solution for a Neutrino Detector

Rathnakar Meka

JÜLICH
FORSCHUNGSZENTRUM

Implementation of a JTAG Verification Environment for a Complex Highly Integrated Real SoC Solution for a Neutrino Detector

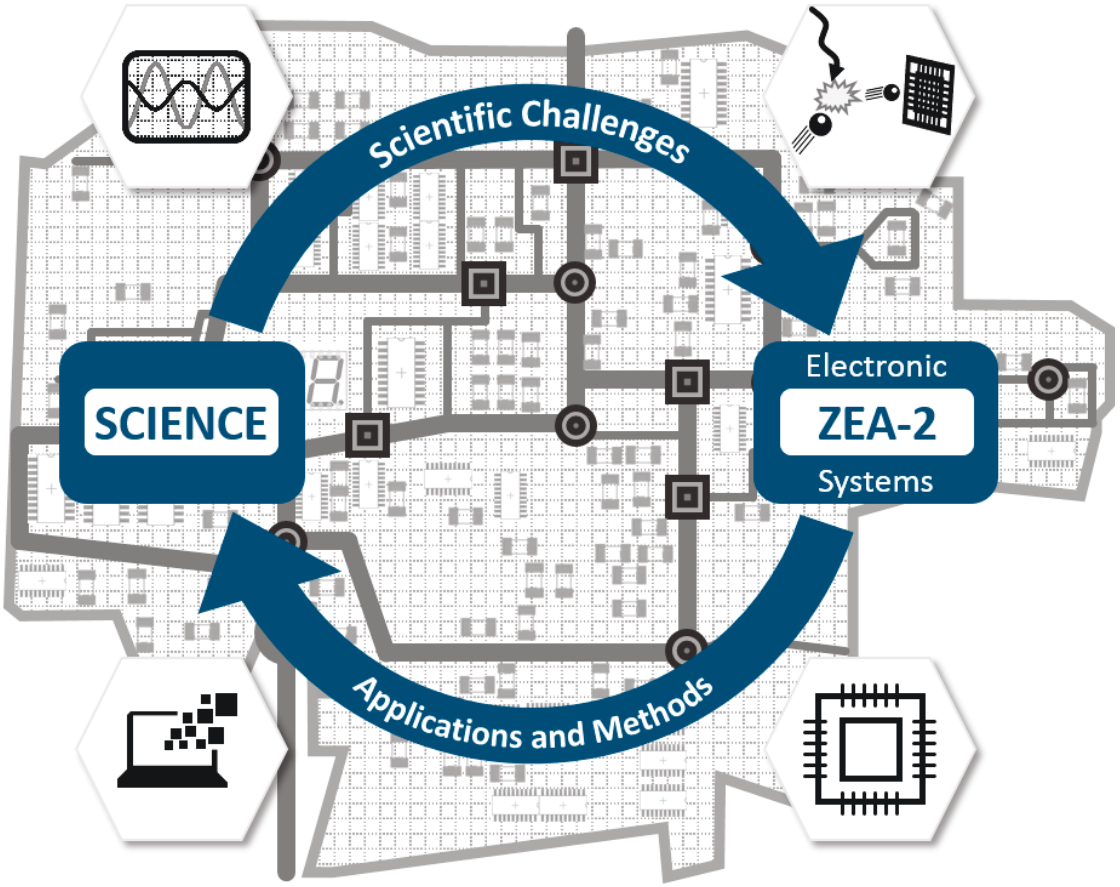
Rathnakar Meka

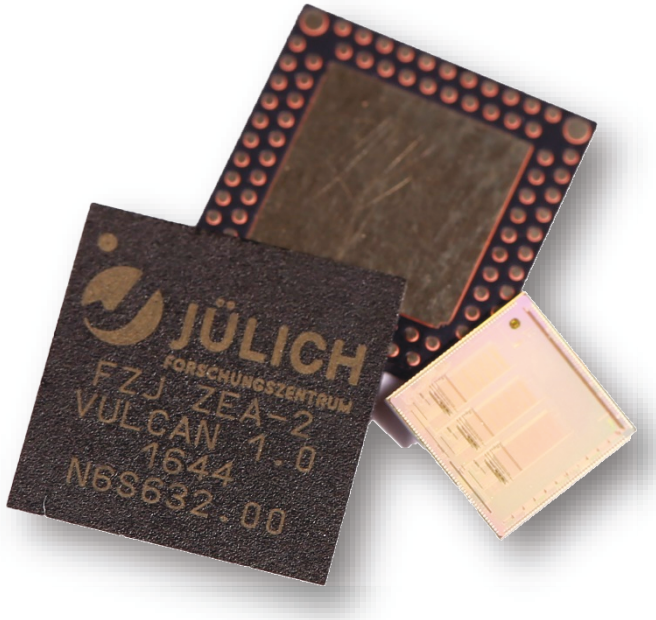
Berichte des Forschungszentrums Jülich; 4402
ISSN 0944-2952
Zentralinstitut für Engineering, Elektronik und Analytik (ZEA)
Systeme der Elektronik (ZEA-2)
Jül-4402

DE 464 (Master, Duisburg, Univ., 2017)

Vollständig frei verfügbar über das Publikationsportal des Forschungszentrums Jülich (JuSER)
unter www.fz-juelich.de/zb/openaccess

Forschungszentrum Jülich GmbH
Zentralbibliothek, Verlag
52425 Jülich
Tel.: +49 2461 61-5220
Fax: +49 2461 61-6103
E-Mail: zb-publikation@fz-juelich.de
www.fz-juelich.de/zb


Scientific Challenges
SCIENCE
Electronic
ZEA-2
Systems
Applications and Methods


JÜLICH
FORSCHUNGSZENTRUM
FZJ ZEA-2
VULCAN 1.0
1644
N68632.00

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

JÜLICH FORSCHUNGSZENTRUM

Implementation of a JTAG Verification Environment for a Complex Highly Integrated Real SoC Solution for a Neutrino Detector

thesis submitted to the faculty of engineering at
University of Duisburg-Essen
in partial fulfillment of the requirements for the degree of
Master of Science in Embedded Systems Engineering

By

RATHNAKAR MEKA
Matriculation No.: 3009223

worked at
Central Institute of Engineering, Electronics and Analytics - Electronic Systems (ZEA-2),
(Zentralinstitut für Engineering, Elektronik und Analytik, Systeme die Elektronik, (ZEA-2))
Forschungszentrum Jülich GmbH

Supervisors: **Prof. Dr.-Ing. Stefan van Waasen**
Prof. Dr.-Ing. Andreas Czylwik

Duisburg, Germany
June 2017

ACKNOWLEDGMENT

First of all, I would like to thank Prof. **Dr.-Ing. Stefan van Waasen** for proposing me such an interesting topic and supervising me in this thesis work. I would also like to take this opportunity to thank Prof. **Dr.-Ing. Andreas Czylwik** for the support in all cases.

I am very grateful to Pavithra Muralidharan, Andre Zambanini for their thoughtful comments and interesting questions about the contents through out my thesis work. Danke schoen!. They have changed my way of approach to a new task and to understand what it means to do. I would like to thank Daniel Liebau, Mario Schlösser, Roger Heil, Christian Roth and Ivan Flammia for their valuable guidance and suggestions. I am also grateful to come to **ZEA-2, Forschungszentrum Jülich GmbH** to work on this thesis and for their support.

I am thankful to my parents and brother for their love, support and prayers.

Finally, I thank all my friends for their support.

The outcome of this thesis work would not be possible without all of them.

Rathnakar Meka, June 2017

ABSTRACT

This master thesis work deals with the configuration, enabling for the verification and interpretation of the data for the application specific designed read-out Vulcan chip by means of MATLAB tool for applications.

Vulcan chip is an advanced system-on-chip (SoC) also called as an analog-to-digital unit (ADU) and it is developed by the Central Institute of Engineering, Electronics and Analytics - Electronic Systems (ZEA-2), Forschungszentrum Jülich GmbH. This chip will be used in the Jiangmen Underground Neutrino Observatory (JUNO) detector project to preprocess and digitize the analog data generated by the applied Photomultiplier Tubes (PMTs) reacting on neutrino events. A configuration system is needed to enable the verification of the designed chip and the digitized data has to be interpreted and verified. A JTAG[1] communication protocol is implemented to investigate the configuration of the chip.

At first, the theoretical fundamentals of the designed chip will be explained to provide the basic understanding of the designed chip. This includes the characteristics of the chip by means of the functionality of individual blocks, configuration interface and different data modes in the chip. Furthermore, a configuration system is implemented which will be used to configure the chip and enables it for verification. Moreover, the configuration libraries are implemented in MATLAB. These libraries enable the chip for verification and test the *write* and *read* operations of all configuration registers in the chip. The implemented configuration system and libraries successfully configures the chip and the configuration libraries will be used in further verification tests.

Secondly, the data extraction algorithms are implemented in MATLAB to analyze the data in different modes for the chip. The algorithms are tested for different data modes and the results are included in the report.

Finally, the created data extraction algorithms will be used in the automatic data acquisition and analysis environment for easy analysis of the data from the chip.

[1] JTAG-Joint Test Action Group

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

A system-on-chip (SoC) is an integrated circuit (also called as an IC or "chip"). It consists of multiple components such as connectivity, memory, analog and digital components of an electronic system mounted on a single bed with a microcontroller or a microprocessor or a DSP core at its central part. The designed SoC needs to be verified in different scenarios. The highly configurable integrated circuit, an analog-to-digital unit (ADU) also called as Vulcan chip is designed by the IC-development team at ZEA-2, Forschungszentrum Jülich GmbH. It will be used in the JUNO project (which will be discussed in section 1.2). This chip preprocesses the analog data, digitizes it and compresses the amount of data but not the quality of data. It can be configured over JTAG interface to make it easy to utilize the required blocks to perform the specified applications.

The designed chip needs a configuration system to enable the chip and to allow the verification of the chip. The configuration system provides the access to the internals of the chip and makes the functionality available and modifiable. The configuration libraries implementation in MATLAB is based on the design specifications and numerous standards as well as guidelines laid down in a joint test action group (JTAG) interface in the area of configuration of the chip. The JTAG interface is a standard serial communication protocol with a test access port (TAP) controller followed by a set of rules. The JTAG TAP port is used for JTAG control as well as providing the serial data in the chip. The configuration system should be developed in such a way that any user can do the configuration of the Vulcan chip by using this system, without any previous experience with the JTAG interface. The data extraction algorithms implementation in MATLAB to interpret the digitized data is also based on the theoretical concepts for different data modes in the chip. The developed configuration system and data extraction algorithms will be used in later stages to create automatic data acquisition and analysis environment for the chip.

1.2 JUNO Experiment

The Jiangmen Underground Neutrino Observatory (JUNO) is a neutrino detector and it is built in China territory following the Daya Bay reactor experiment. It is a multi-purpose neutrino experiment designed with a 20,000-ton linear alkyl benzene (LAB) based liquid scintillator detector of unprecedented 3% energy resolution (at 1 MeV) with 700-meter rock overburden. It is designed to determine the neutrino mass hierarchy and oscillation parameters of the reactor-neutrino energy spectrum from the Yangjiang and Taishan nuclear power plants which both are in 53 km distance and to improve the uncertainty less than 1% in several neutrino parameters [1].

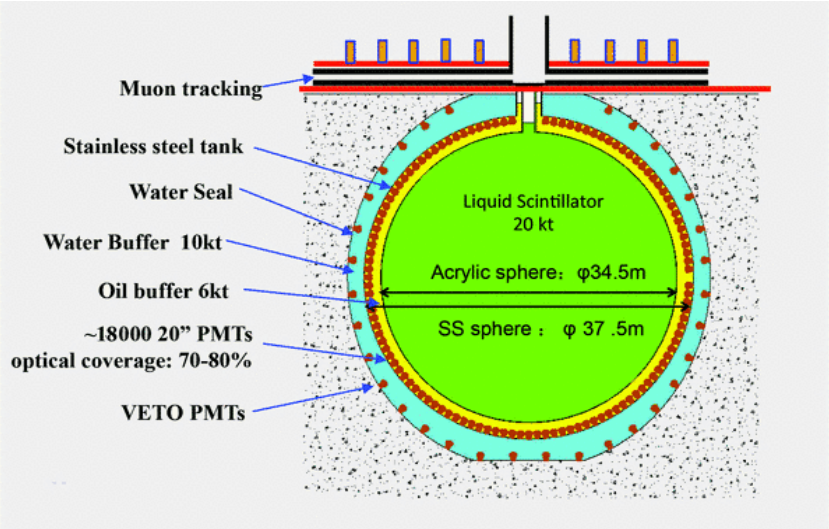


Figure 1.1: Layout of the central detection system of JUNO detector [2]

1.2.1 Scientific Goals

Neutrinos are weakly interacting particles that are created in one of three lepton flavors : electron neutrinos (ν_e), muon neutrinos (ν_μ) and tau neutrinos (ν_τ). Neutrinos change their flavors during propagation. This phenomenon is called neutrino oscillation. The probability for the oscillations of flavors is described by the PMKS matrix which contains the mixing angles (θ_{12}, θ_{23} and θ_{13}) as the main parameters [1]. The differences of squared neutrino masses, $\Delta m^2_{ij} = m^2_i - m^2_j$ (i,j= 1,2,3), have an influence on the energy spectrum of the reactor neutrinos [2].

The primary goal of JUNO detector is the determination of the neutrino mass hierarchy by absolutely measuring the energy spectrum of reactor's anti-electron neutrinos and secondary goal is the finest resolution of mixing angles [2]. Further goals are the measurements of supernova neutrinos, geo neutrinos, solar neutrinos, atmospheric neutrinos and searches for exotic physics phenomena. The precision of Δm^2_{12}, Δm^2_{32} and $sin^2\theta_{12}$ will be improved by

JUNO detector [3]. The current precision values and the estimated detector precision values for Δm^2_{12}, Δm^2_{32} and $sin^2\theta_{12}$ are listed in table 1.1.

Precision measurements		
	present	**JUNO**
Δm^2_{12}	$\sim 3\%$	$\sim 0.6\%$
Δm^2_{23}	$\sim 5\%$	$\sim 0.6\%$
$sin^2\theta_{12}$	$\sim 5\%$	$\sim 0.7\%$

Table 1.1: Precision measurement of neutrino oscillation parameters

1.2.2 Experimental Setup

JUNO will be located in Kaiping, Jiangmen, in Southern China. It is located 53 km away from both the Yangjiang and Taishan nuclear power plants which are in under construction. The total thermal power 36 GW is planned for these reactors. No other nuclear power plants are placed within 200 km distance which avoids neutrino interferences. The overburden for the experimental hall is more than 700 meters (inducing the 270 m height granite mountain which provides a good shielding) in order to reduce the muon-induced backgrounds. Experiment construction has started in 2014 and is planned to be completed in 2019. The construction includes several tunnels, an underground experiment hall, a water pool, a central detector and a muon tracking detector [1]. The overview of the JUNO detector is shown in figure 1.1.

The central detector is filled with 20 kton linear alkyl benzene (LAB) liquid scintillator. The neutrinos interact with the liquid scintillator in subsequent, the scintillator produces the scintillation light. The scintillation light is detected by 18,000 20″-photomultiplier tubes (PMTs). Based on the charge and time information from the PMT, the energy of the incident neutrinos and the interaction vertex can be reconstructed. The energy resolution, radioactivity level and technical challenges are the main involvements to choose the design concepts for the JUNO detector [2]. The energy resolution has to be better than 3% at 1 MeV to reach the expected sensitivity of mass hierarchy. The central detector will be protected by a water pool from the natural radioactivity in surrounding rocks. The selection of the PMTs with high efficiency and highly transparent liquid scintillator are some technical challenges.

1.2.3 Intelligent PMT

Photomultiplier Tube (PMT) makes use of the external photoelectric effect and are exceptional in sensitivity and in response speed [4]. A PMT is constructed as a vacuum tube which consists of a photocathode, a focusing electrode, several dynodes and an anode. Light (photons) enters into PMT strikes the photocathode and eject electrons from the photocathode into the vacuum inside. The focusing electrode accelerates these photoelectrons towards every dynode where each dynode emits more low energy electrons. Finally, the emitted electrons are collected at the anode and results in a sharp current pulse. The figure 1.2 explains the scheme of a photomultiplier tube.

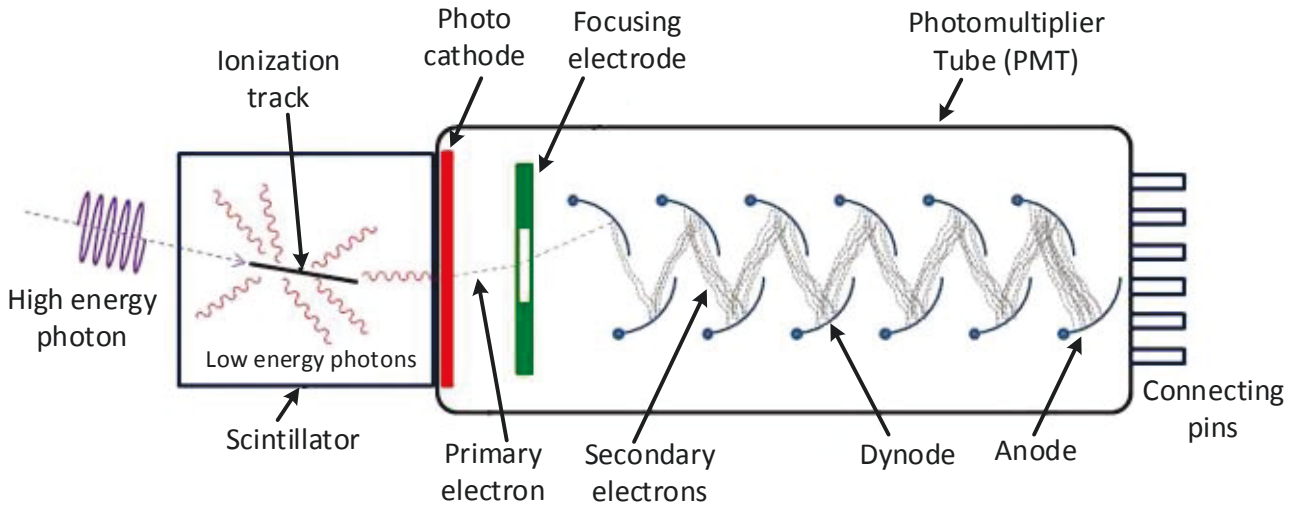


Figure 1.2: Schematic of a PMT coupled to a scintillator [5]

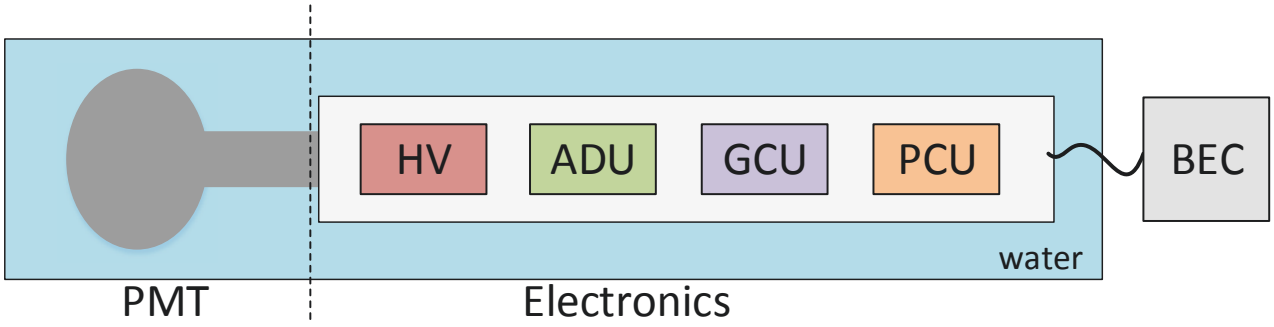


Figure 1.3: PMT with electronics

In the JUNO experiment, each PMT will be attached to a receiver chain (shown in figure 1.3). A cable connects the PMT electronics with a back end card (BEC) which is above the water level. The PMT electronics consists of a PMT base with a high voltage unit (HV), an analog-to-digital unit (ADU), a general control unit (GCU) and a power control unit (PCU). The ADU receives the current signal from each PMT in parallel and digitizes the received signal.

1.3 Vulcan (Analog-to-Digital Unit)

Vulcan consists of three single analog-to-digital converters (ADCs) assigned to three different signal ranges (shown in the figure 1.4) yielding in a large dynamic range. It consists of an analog unit (AU) and a digital control unit (CU). It has configurable operating modes to configure the chip and to test the functionality of individual blocks. This configuration can be done by using JTAG communication protocol. For configuration of the chip, a JTAG macro is inserted while designing the chip.

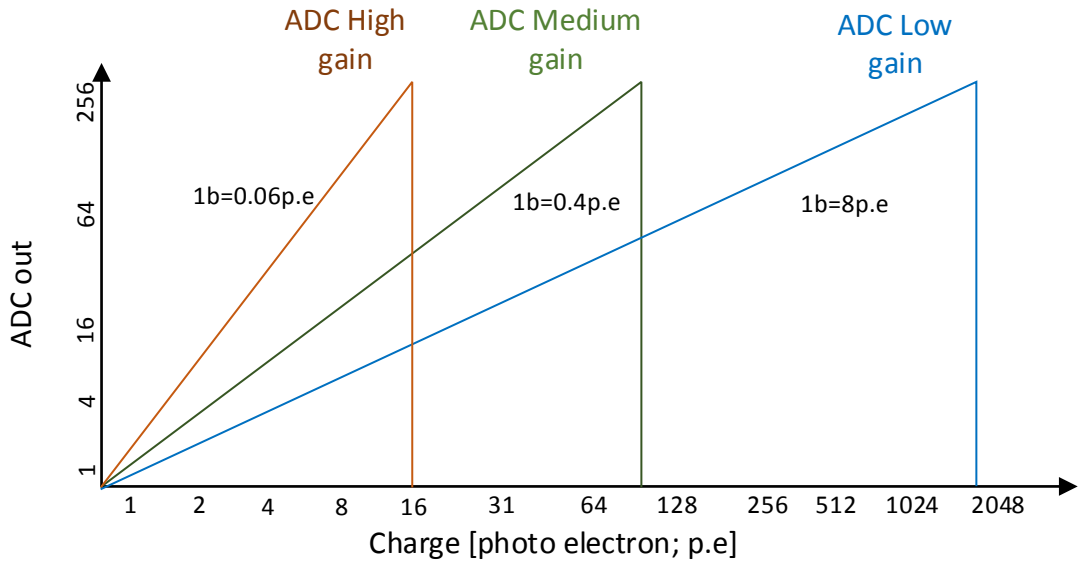


Figure 1.4: Three different ADC gain ranges in the Vulcan chip

Vulcan can process the data in different data processing modes. The processing modes are normal (PAM) mode, ADC passthrough (APT) mode, scan mode, parallel mode, serial mode and derivative mode. In this work, only the first three processing modes are considered. The concept of the Vulcan chip is explained in detail in chapter 2.

1.4 Objective of Thesis and Thesis Organization

In this thesis work, the configuration of the Vulcan chip via JTAG interface is arranged. This thesis work is divided into to two sub tasks. The first task is to create a configuration system (shown in figure 1.5) and configuration libraries which are easy to use MATLAB libraries to configure the chip via JTAG interface by using configuration registers. The *write* and *read* operations in configuration registers are done by using JTAG communication protocol. Configuration registers set the parameters for the blocks and also provide an option to modify the mode of operation. The second task is to interpret the output data from the chip and analyze the data in different data processing modes. Extraction and visualization of the data

will assist in the analysis of the data. The final aim of the thesis is to prepare the data extraction algorithms for different data processing modes using MATLAB.

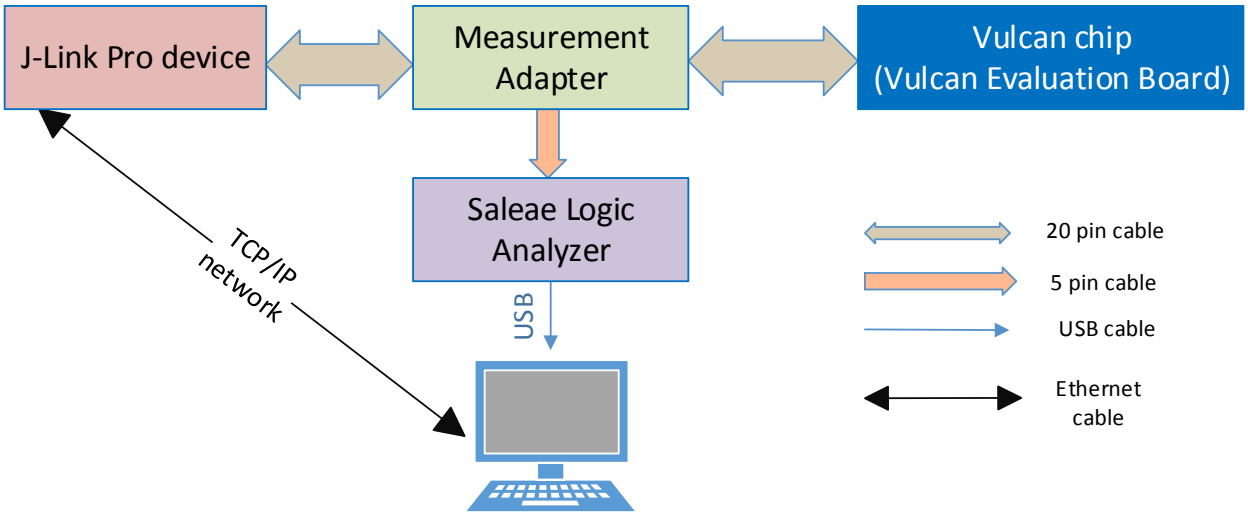


Figure 1.5: Configuration system with the JTAG communication protocol (J-Link Pro device)

This thesis work is documented as mentioned below:

- Chapter 2 explains about the architecture of Vulcan read-out chip.
- Chapter 3 presents the development of the laboratory configuration system and the configuration libraries.
- Chapter 4 shows the development of data extraction functions for the data analysis in different data modes.
- Chapter 5 draws the conclusion on this work and gives a short discussion on future work.

Chapter 2

Vulcan Read-Out Chip

This chapter describes the underlying architecture of the Vulcan read-out chip as well as the data processing modes of it.

2.1 Overview

The main task of the Vulcan read-out chip is to readout the signals from PMTs, to preprocess the analog data, to digitize the data and to transfer the digitized data to the back-end part of the data acquisition system.

Vulcan chip has the basic functions:

- Low power ADCs
- Data can be processed in different data modes
- Data compression
- JTAG communication protocol for configuration

The chip consists of a digital control unit (CU) and an analog unit (AU). The AU includes: three input chains for the same signal and a phase-locked loop (PLL). Each input chain (explained in section 2.2) includes a transimpedance amplifier (TIA), an 8 bit ADC and trigger logic. This complete architecture of the Vulcan chip is designed by ZEA-2.

The PLL generates a clock signal for the ADC, control unit and low voltage differential signaling (LVDS) drivers based on an external 32.5 MHz reference clock. The PLL makes sure that both the internal clock and the reference clock have a fixed phase. The output of an ADC sends data to the control unit, the CU writes into a data buffer and shifts it to the GCU from the data buffer.

2.2 Analog Unit (AU)

The analog unit (AU) receives the current signal from PMTs and converts it into the analog voltage signal. The analog voltage signal is then sampled by a ladder of comparators that produce the digital thermometer code. Then the thermometer code sends out data to the control unit (CU). The overview of the analog unit (AU) is shown in figure 2.1.

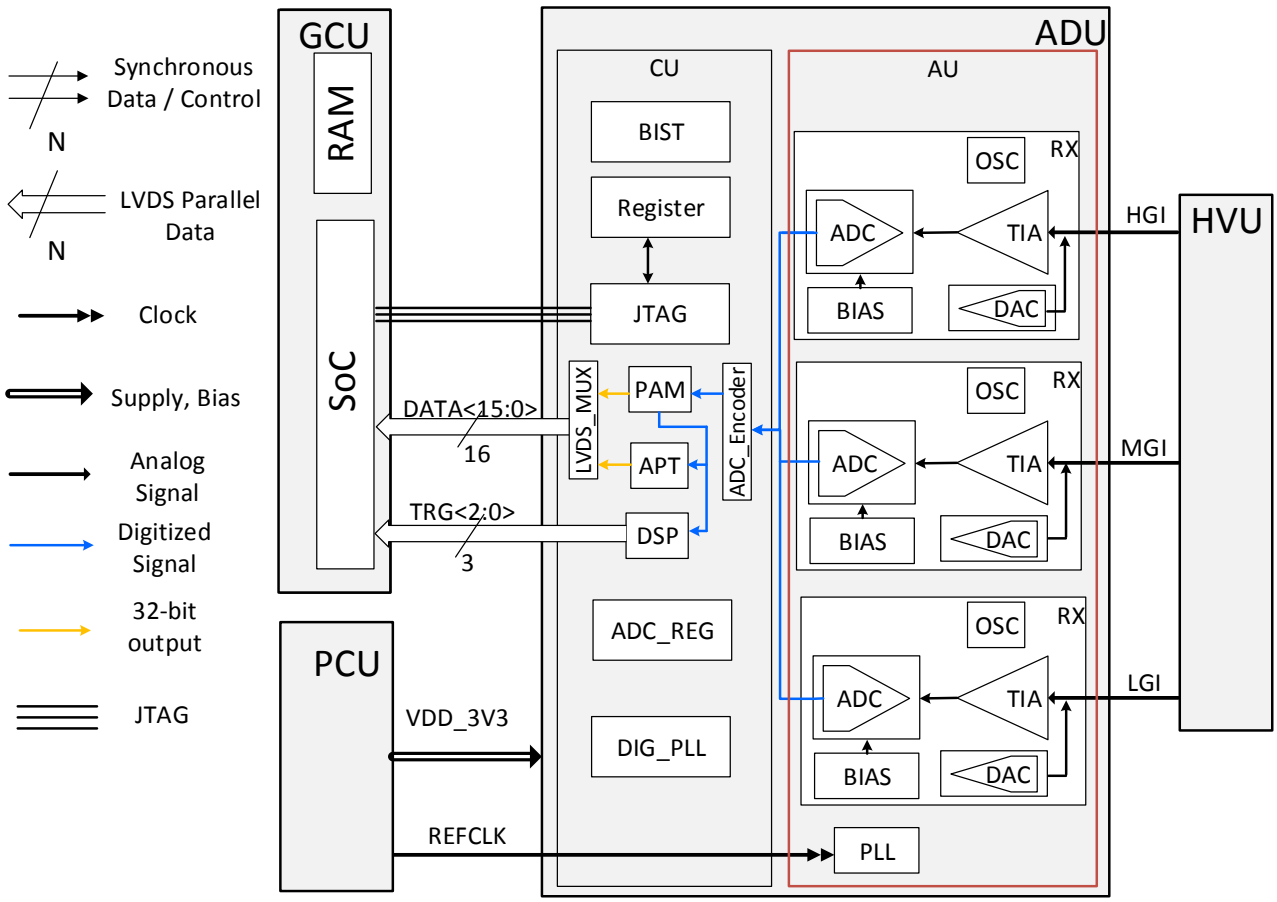


Figure 2.1: Top level of the Vulcan chip

2.2.1 Transimpedance Amplifier

A transimpedance amplifier (TIA) is a current-to-voltage converter implemented with an operational amplifier. It is also used to amplify the small signal from the photomultiplier. Thus, the amplified signal is large enough for further processing. Each transimpedance amplifier suites for an appropriate application but all have one common aspect: the requirement of a current signal into a voltage signal conversion. The figure 2.2 shows a basic block diagram of a transimpedance amplifier.

The functionality of the TIA in Vulcan is current-to-voltage conversion of the input signal (from PMTs). The converted output voltage signal is forward to the ADCs.

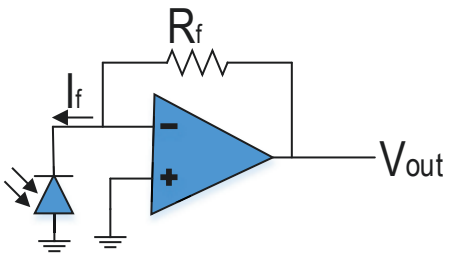


Figure 2.2: A basic block diagram of a transimpedance amplifier

2.2.2 Analog-to-Digital Converter

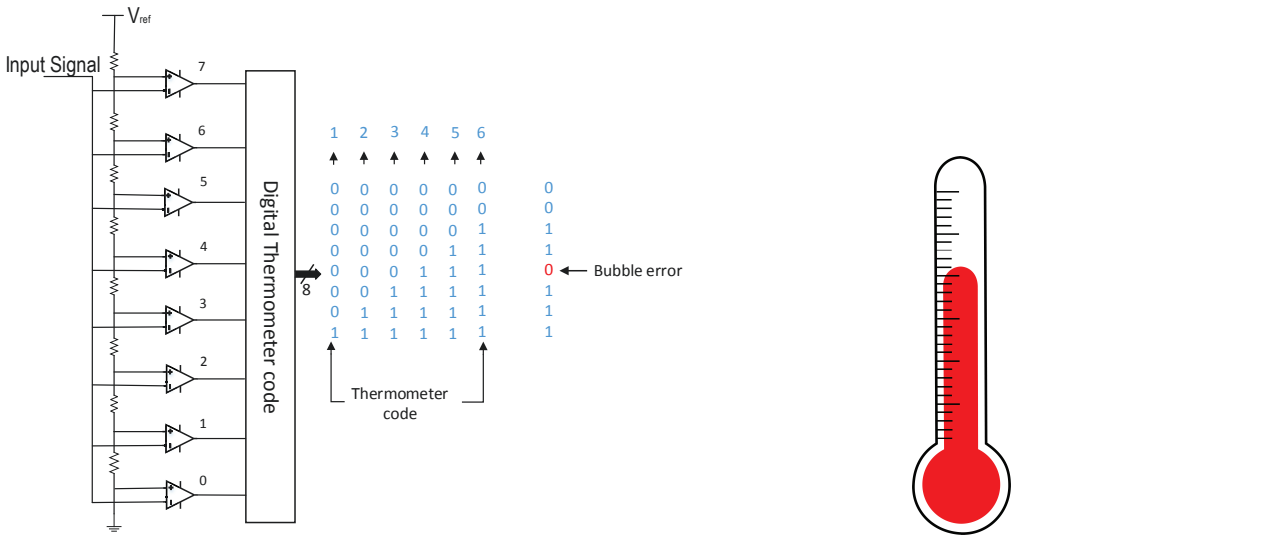
An analog-to-digital converter (ADC) is a device used for interfacing the analog and digital domains. The ADC converts an analog signal into a digital code. This conversion is done in two step process, i.e., sampling the analog signal and quantizing the sampled value.

There are several architectures of converters used for various applications. Flash, sigma delta, successive approximation (SAR) and pipelined converters are few among the types of converters. It can be characterized by the number of bits produced over the analog signal input range (8 bits in Vulcan) and the sampling rate (1 GS/s in Vulcan). The analog input signal of a converter with N-bit resolution can encode into 2^N levels. The accuracy of the measurement is limited by the resolution of a converter. The more accurate measurement needs the higher resolution (number of bits). The effective resolution of a converter is dampened by noise and it is measured by ENOB (explained in section 2.2.2.2). In the Vulcan read-out chip, the converter follows the principal of an integrating flash ADC and has a 8 bit resolution. Each converter has four internal converters with 6 bit resolution.

2.2.2.1 Flash ADC

A Flash ADC, also known as a direct-conversion converter, is the fastest way to convert an analog signal to a digital signal compared to other types of converters. It is suitable for very large bandwidth applications [6]. It is made by cascading high-speed comparators.

It uses a bank of comparators and a resistive ladder which provides the reference voltages to compare the analog input voltage with consecutive reference voltages. The circuit employs $2^N - 1$ comparators for an N-bit converter. The figure 2.3(a) illustrates a 3-bit converter. Each comparator transmits a logic "1" or "0" depends on the comparison of analog input voltage with the reference voltage applied to it. If the analog input voltage signal becomes lower than the respective comparator reference voltage level the output pattern changes from ones to zeros. When the logic "0" is interpreted in the sequence of ones, that zero is called as bubble error (shown in figure 2.3(a)) which is caused by comparator mismatches, noise or distortion. The output of the comparators is fed as an input to the digital encoder which converts the inputs into corresponding binary values [6].



(a) Flash Analog-to-Digital Converter (b) Physical Thermometer

Figure 2.3: Flash ADC with principle logic

The flash ADC approach is known as thermometer encoding. This is so named because the design is similar to the concept of mercury thermometer. The sequence of ones represents a mercury part and the sequence of zeros represents the empty space in a physical thermometer (figure 2.3(b)).

2.2.2.2 ENOB

ENOB [7] known as Effective Number of Bits or Effective Bits characterizes the real dynamic performance of an ADC. However, the real ADC circuits deals with the noise and distortion. The ENOB of an ideal is equal to the resolution of a ADC. The ENOB is based on the signal to noise ratio (SNR) equation of an ideal ADC based on a pure sinusoidal signal. The equation 2.2 is the signal-to-noise ratio (SNR) of an N-bit ADC.

$$SNR = (6.02 \times N + 1.76)\, dB \tag{2.1}$$

Where N is the resolution of an ADC. The equation is rearranged to calculate the effective number of bits out of an SNR measurement in dB.

$$ENOB = \frac{SNR - 1.76}{6.02} \tag{2.2}$$

2.2.3 Phase-Locked Loop

A phase-locked loop (PLL) is a clock signal generator and it generates an internal clock signal which is multiplied of a low frequency external reference signal. A PLL consists in general of

an oscillator, a phase detector, frequency divider and loop filter. For matching the phase of an internal clock, the oscillator constantly is adjusted on frequency to match the phase and frequency of a reference signal. The phase detector always compares the phase of an oscillator output signal with the phase of a reference signal.

The functionality of the PLL in Vulcan is generation of a 4 GHz clock signal from an external reference frequency of 32.5 MHz. This clock signal is divided down to 500 MHz and 250 MHz for ADC or LVDS and digital part respectively.

2.3 Digital Control Unit (CU)

2.3.1 Overview

In the digital control unit, the length of data samples is reduced from 256 bit to 8 bit samples and transmits the data in different data processing modes with double data rate.

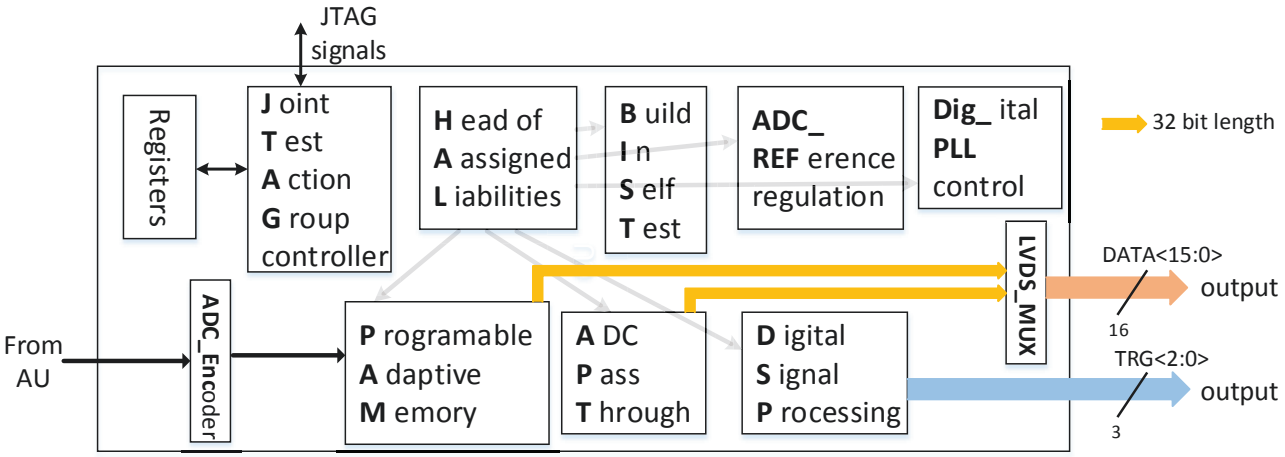


Figure 2.4: Overview of the digital control unit

In the control unit, a JTAG macro has been employed including a TAP controller which controls the central state machine by receiving commands on the JTAG TMS pin. The JTAG macro writes into and reads from the configuration registers that store the configuration of the chip. The data from all three ADCs (four samples from each ADC) transfers to the ADC-Encoder. It converts the data from thermometer code to a Gray coded data. Then, it sends the data to the Programmable Adaptive Memory (PAM) module. Vulcan has different data processing modes (explained in detail in section 2.4) for the data transmission. In normal (PAM) mode, the additional information is added to the data and stored in an internal ring buffer and then transmits to the output. In ADC pass-through (APT) mode the data transmits to the output without saving. The Digital Signal Processing (DSP) mode block is the source of trigger lines. In LVDS multiplexer (LVDS_MUX) the data processing mode among different modes can be chosen. LVDS bus with 16 bit length transmits the data with double data rate (DDR).

2.3.2 JTAG Core

Joint Test Action Group (JTAG) is an advanced DFT technique for the purpose of configuring and testing the chip. JTAG is chosen for the configuration because it is available as an IP core and it is configurable.

JTAG interface is an IEEE 1149.1 standard four-wire (and an optional reset signal) serial protocol that establishes the details of access to any chip with a JTAG port. JTAG is the name of the task force that developed the IEEE 1149.1 standard and by now also for communication standard. The number of connector signals (wires) depends on the version of JTAG. It can be two, four or five signals.

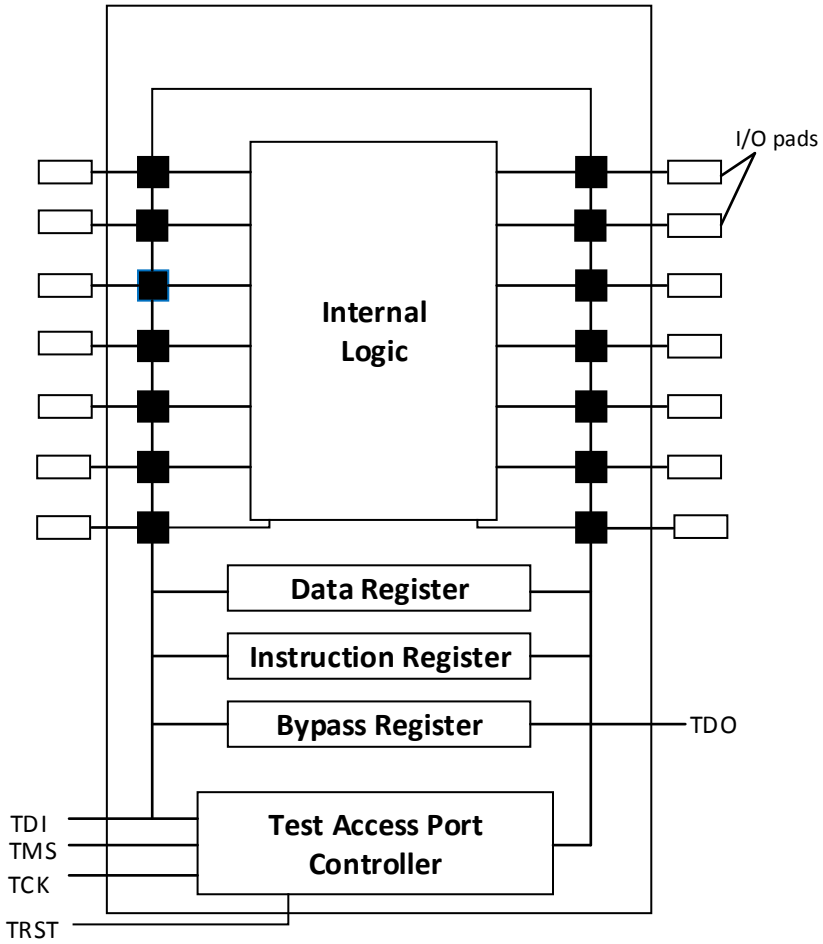


Figure 2.5: Schematic of a JTAG enabled device

The connector signals (shown in figure 2.5) are:

1. **Test Clock (TCK)**: The test clock pin provides the clock signal to run the TAP controller, which loads and unloads the instruction and data registers.

2. **Test-Mode select (TMS)**: The test mode select pin controls the test operations on the TAP controller. On the rising edge of TCK, depending on the state of TMS, a transition will be made in the TAP controller state machine.

3. **Test Data Input (TDI)**: The test data input pin is used to shift serial test instructions into instruction register and data into the data registers. TDI is clocked into the device on the rising edge of the TCK.

4. **Test Data Output (TDO)**: The test data output pin is used to shift serial test instructions from the instruction register and data out of data registers. TDO is clocked out on the falling edge of the TCK.

5. **Test Reset (TRST)**: The Test reset pin is used to reset the TAP controller.

The functionality offered by the JTAG interface is debug access and boundary scan:

- **Debug Access** is used by debugger tools to configure the chip and to access the internals of the chip. Examples: registers, memories and the system state [8].
- **Boundary Scan** is used by hardware test tools to test the circuitry connections of a device, example: on a PCB (Printed Circuit Board).

The configuration of the Vulcan chip can be done by using configuration registers which are controlled by the instructions. The JTAG interface has four predefined instructions and three custom instructions to configure the Vulcan chip. These custom JTAG instructions are provided to allow writing and reading operations in the configuration registers and access the functional blocks of the chip.

The debugger tool which has a JTAG port connects to the host by using an interface such as USB, Ethernet and connects to the target device to configure it (explained in detail in chapter 3).

2.3.2.1 TAP Controller

The schematic of the Test Action Port (TAP) is shown in figure 2.6. It contains four input signals and an output signal that are driving the circuit blocks and controls the operations (write or read) specified. The TAP facilitates a serial loading and unloading of instructions and data. The table 2.1 lists the TAP signals with a small description.

The TAP defines a set of registers and a controller that are used to define the operation of the JTAG interface. There are two types of registers associated with the interface. Each compliant device has one instruction register and one or more data registers (see in figure 2.5). The instruction register has a 3 bit length and the data register has an 8 bit length for the Vulcan

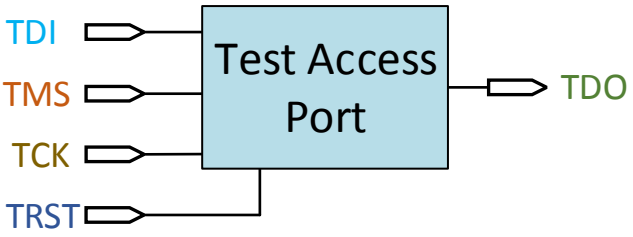


Figure 2.6: Schematic of a Test Access Port

TAP signal	Type	Description
TCK	input	Generates a JTAG Clock signal
TMS	input	Controls the TAP controller state transitions
TDI	input	Serial data from debugger tool to target
TDO	output	Serial data from target to debugger tool
TRST	input	Optional, resets the TAP controller

Table 2.1: TAP controller signals

chip. The table 2.2 lists the length of instruction register and data register for the Vulcan chip.

Type of register	Register length (no.of bits)
Instruction	3 bits
Data	8 bits

Table 2.2: Registers and bit length for the Vulcan chip

The TAP Controller (shown in figure 2.7) is a 16-state finite state machine added on the IC-die itself. It produces the internal control signals. The TAP controller is driven by TCK and TMS signals only. These two signals drive the TAP Controller as a 16-state machine to generate a clock and control signals for the instruction and data registers. A test clock rising edge and system power-up events can trigger a change of controller state.

The flow through a state machine is controlled by the value of TMS signal. The state of the TMS signal at the rising edge of TCK is responsible for determining the sequence of state transitions. The state machine is set into instruction mode (shift-IR state) by sending the sequence 01100 on JTAG TMS signal to the controller. Then, the instruction will be shifted into instruction register which is sent on the JTAG TDI input. The state machine sets into data mode by sending the sequence 1100 on JTAG TMS signal. Then, the data is shifted into data register which is sent on the JTAG TDI input.

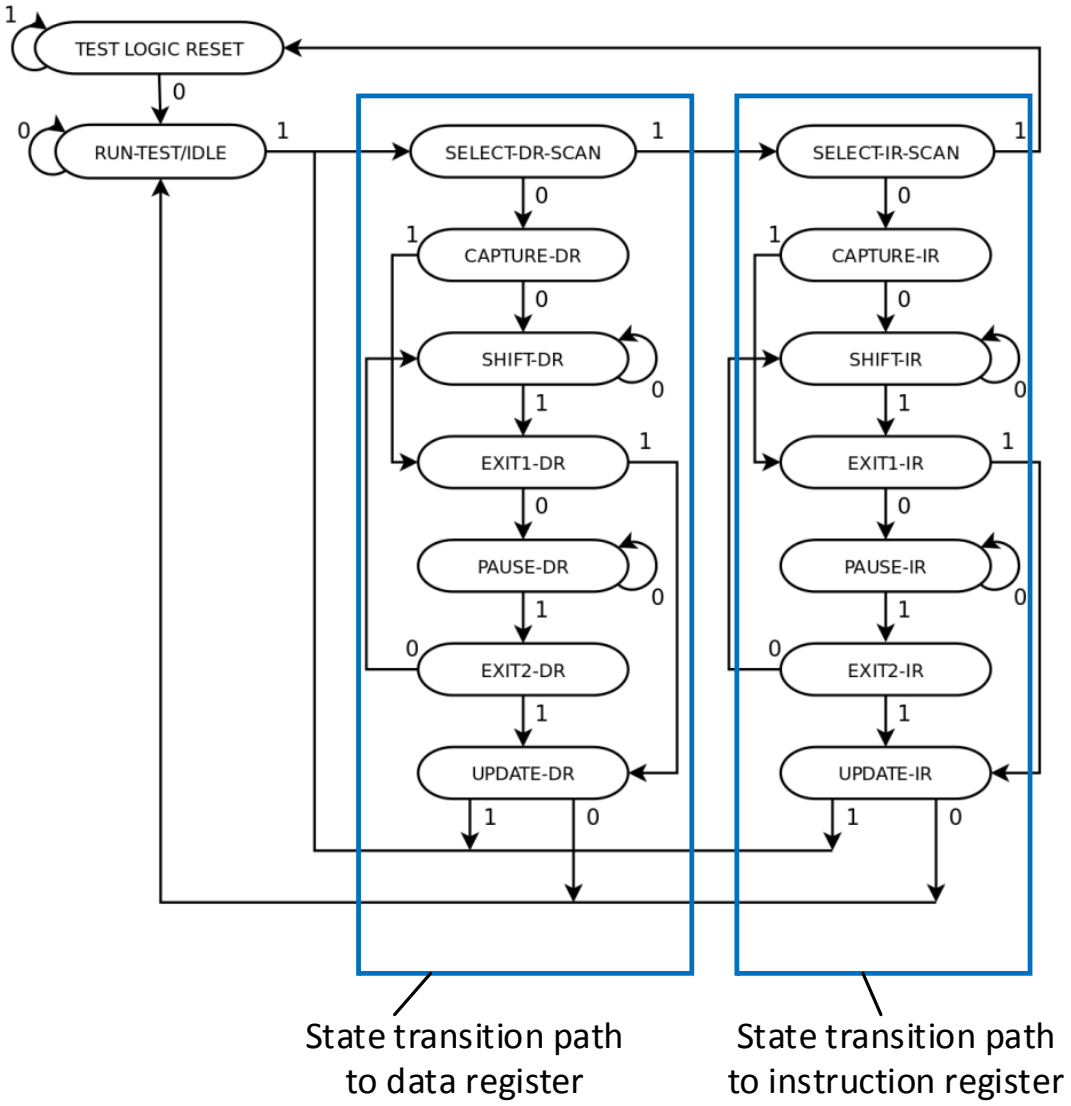


Figure 2.7: TAP Controller state machine [9]

There are two state transition paths (shown in figure 2.7) for shifting the data on TDI signal into the device,

- one for shifting in an instruction to the instruction register and,
- one for shifting data into the data register as determined by the current instruction.

2.3.2.2 JTAG Custom Instructions

The JTAG IEEE standard 1149.1 describes few instructions that can be implemented but SAMPLE, EXTEST, PRELOAD and BYPASS instructions are mandatory for all JTAG enabled devices. Custom instructions are the instructions designed for the Vulcan chip to perform the configuration operations. They comprise of loading an address (LOADADD), writing data (WRITE) and reading data (READ) instructions. As mentioned in table 2.2 all instructions have 3 bit length. The chip configuration is done by using only these instructions. All the instructions with an assigned Opcode and also a functionality performed by the JTAG interface in the chip are listed in the table 2.3.

Name of an Instruction	**Opcode**	**Description**
SAMPLE	001	Connects JTAGTDI and JTAGTDO together via a Boundary Scan Register (BSR) and read the data
EXTEST	010	Writes the data to the core (internal logic). JTAGTDI and JTAGTDO are connected via the Boundary Scan register (BSR)
PRELOAD	011	Preloads the test data into the BSR before loading an EXTEST instruction
LOADADD	100	Shifts the address of a configuration register to perform write and read operations
WRITE	101	Writes the 8-bit data into configuration register which is set by the LOADADD
READ	110	Read the 8-bit data from configuration register which is set by the LOADADD
BYPASS	111	Connects JTAGTDI and JTAGTDO together via a single bit bypass register

Table 2.3: List of instructions and an Opcode assigned for instructions

The figure 2.8 describes how the instructions and data are transmitted on JTAG signals. Every instruction and data word are transmitted on TMS, TDI and TDO signals from LSB to MSB order.

Figure 2.8: Exemplary sequences for register interaction, an x represents an undefined state. Transmission in each block is starting from the LSB to MSB

2.3.3 Configuration Register Implementation

Vulcan has 256 configuration registers but only 248 registers (81 for the digital control unit and 167 for the analog unit) are used for the Vulcan configuration and access to these registers is provided by the JTAG interface. Each register has a length of 8 bits with 8 bit unique address. For every configuration register, the LOADADD instruction has to be loaded before performing a writing or reading operation. Once the LOADADD instruction is loaded and the address of a configuration register is being set, the WRITE and READ operations will be performed for the selected configuration register. The TAP controller shift-IR state is used for instruction register to load an instruction and the shift-DR state is used for the data register to write and read the data. In this configuration, the length of an instruction register and a data register is fixed.

2.3.4 Other Blocks

2.3.4.1 BIST

The built-in-self-test (BIST) is used to test the integrated circuit by itself independent of external equipment through generating the temporary logic and analyzing the resultant response. The temporary programmable logic or rewritable memory of the BIST avoids the extra cost of an integrated circuit.

In Vulcan, the BIST generate signals either for the ADC through DAC or for the digital part and thus increases the testability. The BIST has two options to generate signals: by using stored data and by generating a programming logic. The BIST module does not have a verification option. Therefore, it is not possible to verify the Vulcan chip independent of external equipment.

2.3.4.2 ADC-Encoder

The ADC-Encoder converts the thermometer code into gray coded data by direct conversion and encode the thermometer code into binary format code by counting the number of ones in the thermometer code. This module is configured by using the configuration register.

The output of an ADC is a 256 bit thermometer code sample and it is forwarded to the encoder. The encoder is highly configurable and it has a bit manipulation scheme and a Gray coded data conversion scheme. Bit manipulation scheme can change the order of samples (6 bit ADC samples), force bits to zero, invert the bits in the sample, invert the sequence of ones and sequence of zeros in a thermometer code and change the bit order in ADC data. Gray coded data conversion scheme has two options to convert the data: by counting the number of ones (in which case there is no gray encoding) and state transition detection in the thermometer code. It sends the gray coded data to the PAM module.

2.3.4.3 PAM

In Vulcan, the programmable adaptive memory (PAM) module generates the gain mode change information along with the data samples and stores it in an internal ring buffer. Then, transfers the stored data to the output. In this module, the amount of the data (especially noise) can be compressed but not the quality of data.

In PAM module the gray coded data is converted into binary data if the output of an ADC-Encoder is a gray coded data. The normal (PAM) data mode acts as a primary data processing mode of the Vulcan chip. It receives data samples from three different ADCs in the chip. The assignment of ADC 1, 2 and 3 to high, medium and low gain is done with upper and lower threshold values set by the configuration register.

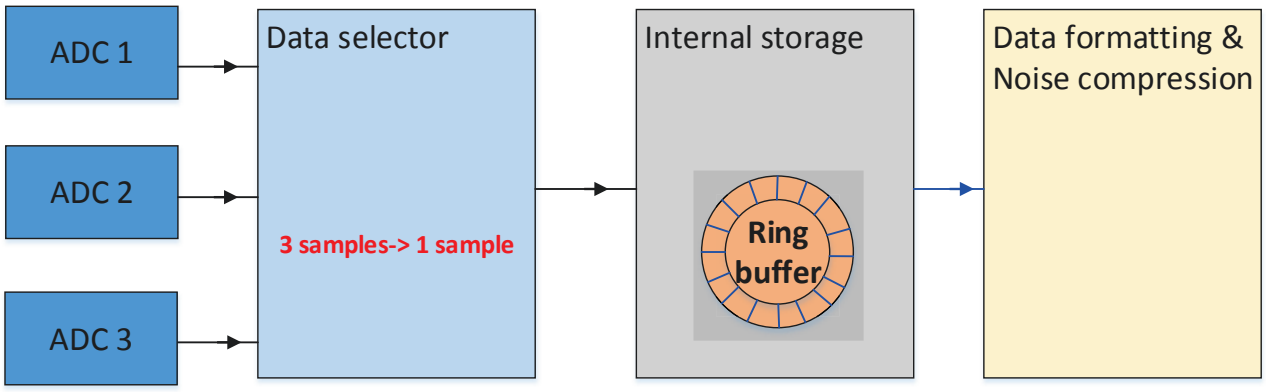


Figure 2.9: Block diagram of the Programmable Adaptive Memory

The selection of one sample among three samples of three different ADCs (shown in figure 2.9) for every sample clock is also done with the upper and lower threshold values. The highest resolution ADC sample is chosen by comparing the three ADCs. The PAM generates the header information along with the data and stores in the ring buffer. This header information consists of different gain (high, medium and low) mode changes and the system events. The data is formated based on different gain mode changes and system events. Then the formated data is send out to the output. The data formatting and noise compression are explained in section 2.4.1.

2.3.4.4 DSP

The digital signal processing (DSP) block provides the trigger output lines. There are three different trigger output lines for the Vulcan chip. The trigger generation is performed in parallel to the main data processing and it is indicated by trigger lines.

2.3.4.5 LVDS Multiplexer

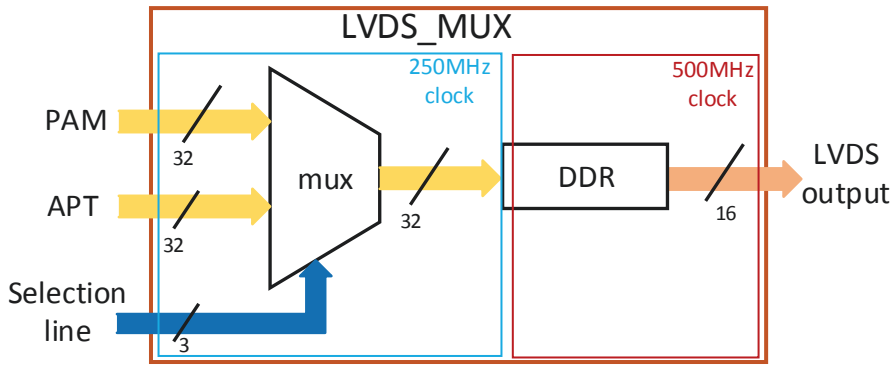


Figure 2.10: An excerpt of the LVDS multiplexer

A portion of the low-voltage differential signaling multiplexer (LVDS_MUX) is shown in figure 2.10. It selects the data which can transfer to the LVDS output. It has 8 different settings,

encoded in 3 bits (000 to 111) for different data modes (mentioned in section 1.3) and to select different possibilities like the PLL counter or a test pattern. The multiplexer gets the data from different data modes as 32 bit data samples. The data mode selection in multiplexer can be done by using a configuration register. The LVDS output bus has 16 bit data length. The parallel bus operates with double data rate (DDR) and transfers data on both the rising edge and falling edge of a clock signal.

2.4 Data Processing Modes

The ADC in analog unit drives with a 500 MHz clock and transfers the data as 256 bit samples, digital control unit drives with a 250 MHz clock and transfers the data as 32 bit samples (4× 8 bit sample) and the LVDS block drives with a 500 MHz clock and transfers the data to output as 16 bit samples in every clock cycle.

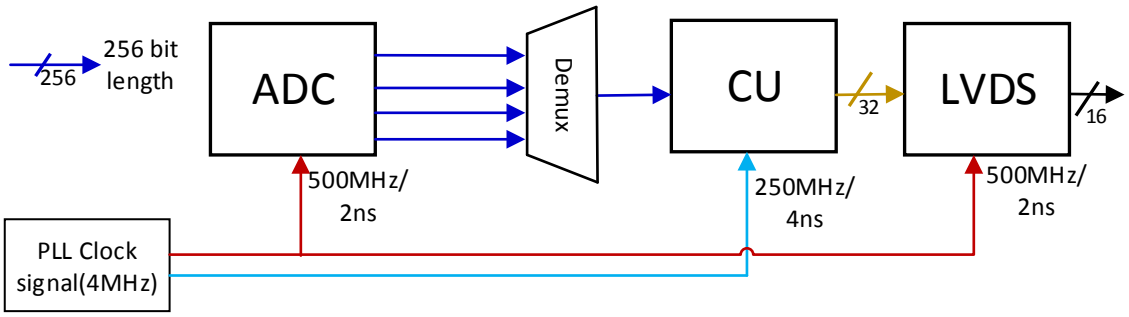


Figure 2.11: PLL clock signal division for different blocks in the chip

As mentioned in section 1.3, Vulcan can be operated in different data processing modes. The three particular modes (normal mode, APT mode and scan mode) require a 16 bit LVDS output bus (DATA0 – DATA15). Normal mode and APT mode uses the three independent trigger LVDS output lines (TRG0 – TRG2) with the LVDS output bus and scan mode uses the LVDS output bus and the first trigger line (TRG0) only. If not mentioned otherwise, data is transmitted with most significant bit (MSB) first.

2.4.1 Normal Mode

The Normal (PAM) mode is the primary data processing mode for the Vulcan chip. In this mode the data samples are transmitted along with the system events and sends out the data with the included system events. This mode is used to compress the noise samples and to reduce the amount of data but not the quality of the data.

The optimal ADC selection is explained in section 2.3.4.3. The 32 bit header information (shown in figure 2.12) consists of an information of a synchronization counter, a digital counter, a bubble counter, bus mode (gain mode change) and trigger.

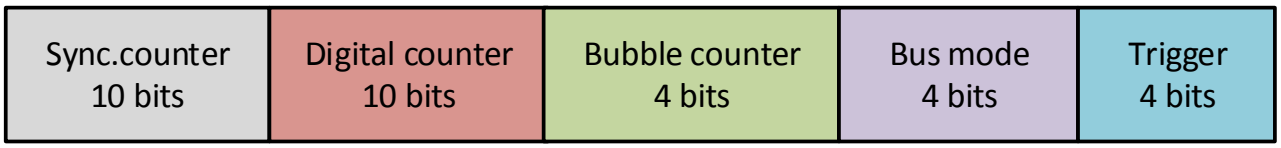


Figure 2.12: A 32 bit header information

The header information determines the source (ADC) of the data samples. The header information delays the data by two clock cycles thus increases the overhead. Therefore, an internal ring buffer is used to store the data samples and the header information until it can be sent out. While sending the data out from ring buffer a reset signal with four consecutive samples as zeros is added before header information to distinguish the header information from the data samples. Valid data samples contain at least one non-zero bit otherwise, they would be considered as a reset signal. Multiple reasons can cause the occurrence of a trigger event. The trigger event can be a gain mode change and other system events (mentioned in figure 2.12). The header information is sent out before data when the trigger event occurs. Several trigger events will fill up the ring buffer and can causes overflow. The concept of emptying the ring buffer, bus mode changes and trigger events are explained in detail in sections 2.4.1.1 and 2.4.1.2.

2.4.1.1 Bus Modes

The bus mode is a change of the gain from one to other gains (high gain, medium gain and low gain). This bus mode indicates the changed gain mode with the defined code in the header information.

Bus Mode	**Mode Description**	**Header Code**
NOISE	Transmits four LSBs of four consecutive samples (4-bits each) of the “High Gain” ADC	0001
HG	Transmits 8-bit samples of the “High Gain” ADC	0010
MG	Transmits 8-bit samples of the “Medium Gain” ADC	0011
LG	Transmits 8-bit samples of the “Low Gain” ADC	0100
NOISE50	Like NOISE but changes to MG and LG are disabled	0101
HG50	Like HG but changes to MG and LG are disabled	0110
NOISE80	Like NOISE50 but system event triggers are disabled	0111
HG80	Like HG50 but system event triggers are disabled	1000

Table 2.4: List of all implemented bus modes

The table 2.4 lists the bus mode changes and the assigned code used in the header information. The NOISE mode bus mode is selected if the signal falls below a specified threshold. In NOISE

mode the four least significant bits of four consecutive samples[1] of the "High Gain" ADC are combined as 16 bit data word and transmitted. In total 8 data samples are read from the ring buffer while it is writing only 4 samples in every clock (250 MHz) cycle. This leads to an emptying of the ring buffer. Bus mode changes may happen from any bus mode to any other bus mode. At every rising edge or falling edge of the clock, the bus mode can change to HG, MG or LG bus mode. After the Power-on-Reset, the system operates in HG bus mode. In HG bus mode, two 8-bit samples of the "High Gain" ADC are transmitted. Depending on the signal level, the bus mode may change to either MG or LG where two 8-bit samples of the "Medium Gain" or the "Low Gain" ADC are transmitted respectively.

If the bus mode change happens repeatedly the changes will fill up the ring buffer and causes the overflow. Then, the bus modes NOISE50, HG50, NOISE80 and HG80 are addressed. The bus mode changes to NOISE50 or HG50 bus mode if the ring buffer fills up 50% of it. Bus mode changes to MG and LG are disabled if the NOISE50 or HG50 bus mode is selected. If the ring buffer occupancy reaches 80%, the NOISE80 or HG80 bus mode is selected and bus mode changes to MG, LG modes and the trigger events generation are disabled. Thus, the occupancy of the ring buffer is controlled.

2.4.1.2 Triggers

The header information interrupts the data transmission when the bus mode change or any system event occurs. Multi-source trigger events (BAD, BAP, PCO and BPO triggers) combine triggers from multiple sources in one event. The respective trigger code is transmitted in header information which causes the data interruption. The table 2.5 customizes all trigger sources and the assigned trigger code used in the header information.

Trigger	**Trigger Description**	**Trigger Code**
BMC	Bus mode change	0001
DCO	Digital counter overflow due to late syn signal	0010
SYN	SYN signal detected	0011
RBU	Ring buffer underflow. Less samples available than required	0100
BAD	Bus mode change and digital counter overflow	0101
BAS	Bus mode change and syn signal detection	0110
SCO	SYN signal detection and syn counter overflow	0111
BSO	Bus mode change and syn signal detection and syn counter overflow	1000

Table 2.5: List of all considered trigger sources

[1]The reason for selecting four consecutive samples is the control unit is operated at 250 MHz clock signal and the data is transmitted as four samples in one clock cycle

Bus mode change (BMC) trigger addresses any kind of bus mode change and the new bus mode code with the BMC trigger code are provided together in the header information. At each sample (8 bit) of data, the bus mode can be changed. The BMC trigger code is generated and transmitted into header information before the data samples of a new bus mode are transmitted. If the bus mode change takes place after the transmission at the rising edge, the new trigger sequence is transmitted before the data otherwise stuff the data samples with zeros until the header information can be transmitted. Then, the non zero samples which have already been sent are replaced by zeros. The figure 2.13 explains the concept of stuffing zeros in empty samples.

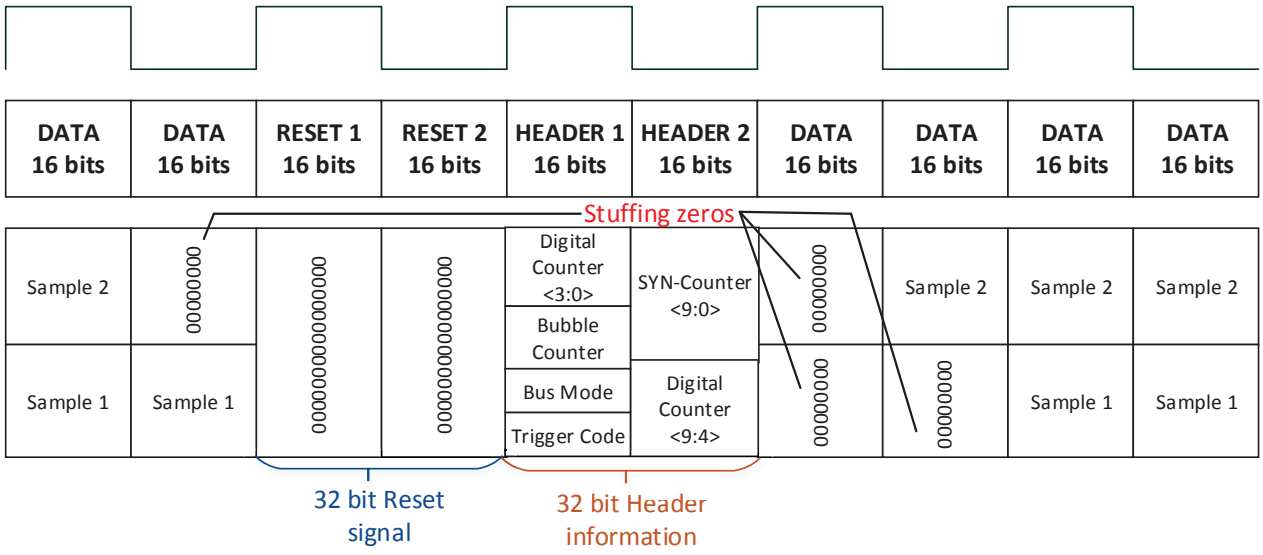


Figure 2.13: Concept of stuffing zeros in empty samples

The other information in the header is from the bubble counter, digital counter and synchronization counter. The 4-bit bubble counter provides the amount of bubble errors in the data. The timing information is provided by the 10-bit internal counter. A synchronization counter synchronizes all read-out chips and the SYN synchronization signal which is intended to be continuously pulsed resets the digital counter. A digital counter overflow (DCO) trigger is generated when the SYN synchronization signal not pulsed in time and results the digital counter overflow. The ring buffer underflow (RBU) trigger is generated when the ring buffer has fewer samples than required to form a valid data word.

2.4.2 ADC Pass-Through Mode

The ADC pass-through (APT) mode is the alternative data processing mode for the Vulcan chip. It is used primarily for debugging of the internals of the ADC and to process the data samples.

The ADC Pass-Through (APT) block gets the binary converted ADC 1, 2 and 3 samples from the PAM module. The high gain, medium gain and low gain selection for ADC 1,2 and 3 samples is done like in PAM mode (see in section 2.3.4.3). The LVDS trigger output lines (TRG2, TRG1 and TRG0) indicates three different gain sources. One among three gains (high gain, medium gain and low gain) is selected based on the gain source and transmits the data of the chosen gain along with the LVDS trigger lines. In order to avoid the ambiguous assignment of the trigger output lines, a pair of two samples has to have the same gain origin. The LVDS output lines have enough bandwidth because no additional information (trigger sequence) is transmitted along with the data samples on the LVDS (16 bit data lines) bus. The figure 2.14 illustrates the transmission of data samples along with the trigger lines.

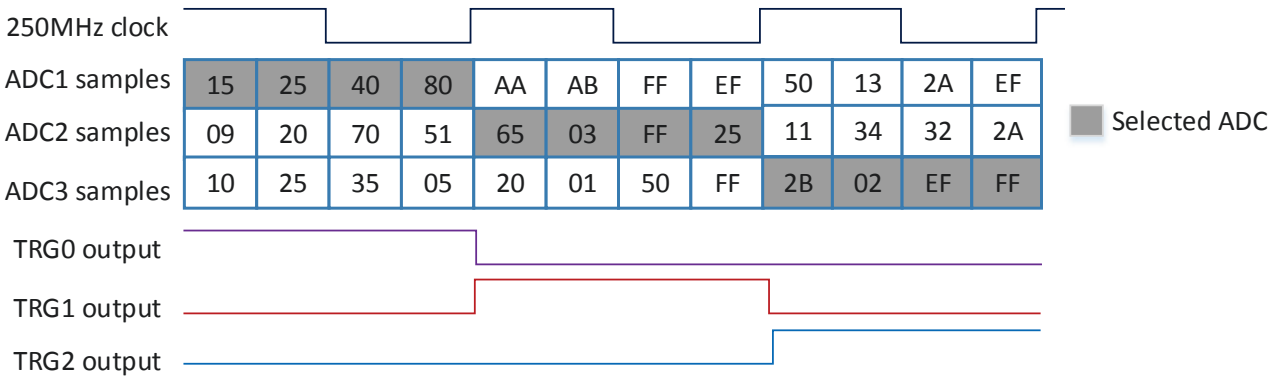


Figure 2.14: Data processing scheme in ADC Pass-Through Mode

2.4.3 Scan Mode

The scan mode is used for the debugging of the internals of the ADC and also for configuration purpose.

The figure 2.15 explains the data transmission in scan mode. This mode is also called as “debug mode”. In this mode, every four thermometer code (256 bit) samples from the ADC are buffered as a data set (4×256 bits) and are transmitted as 32 sections. Each section consists of 8 bits from all four thermometer code samples of the data set. The data sets are transmitted from all three ADCs. The control unit is operated at 250 MHz clock signal and the data is transmitted as four samples in one clock cycle. Therefore, one section out of 32 sections of the data set is transmitted in every clock cycle but the transmission of 4 (8 bit) samples in clock cycle is still same. Thus, the sample 1 and 2 are transmitted on an LVDS bus as a first LVDS word and the

sample 3 and 4 are transmitted on LVDS bus as a second LVDS word from each section of the data set. To identify the start of a new data set, the first trigger line (TRG0) is used. When the new thermometer data set is started, the trigger line TRG0 is enabled only for one clock cycle (first section of the data set).

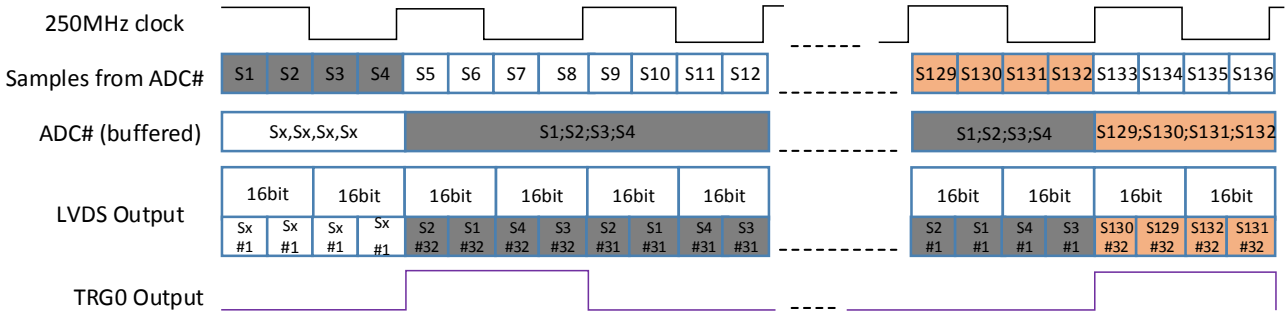


Figure 2.15: Data processing scheme in Scan Mode

The first design version of the Vulcan chip has a bug for this data mode in it. That is the TRG0 never enables for this mode. So the workaround method is introduced to overcome this issue in the design. The workaround method is explained in detail in section 4.4.2.2.

Chapter 3

Development of the Laboratory Configuration System

This chapter describes the configuration system setup to configure the Vulcan read-out chip and configuration libraries to perform *write* and *read* operations in the configuration registers. The thesis work started with the configuration system setup.

The Vulcan chip configuration is done by using a JTAG enabled device called J-Link Pro debugger from SEGGER (explained in detail in section 3.1.1) which has a JTAG interface with a TAP controller induced.

3.1 Hardware Components of the Configuration System

In this section, the proposed configuration system with all hardware support is discussed. It consists of a SEGGER J-Link Pro debugger, a measurement adapter, Saleae logic analyzer, Vulcan evaluation board and the zedboard FPGA for emulation purpose. The schematic of the hardware setup is shown in figure 3.1.

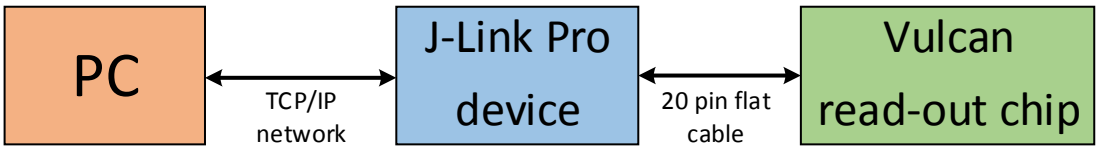


Figure 3.1: A schematic of the hardware setup

3.1.1 SEGGER J-Link Pro Debugger

J-Link Pro V4.00 is a JTAG emulator device (shown in figure 3.2) manufactured by SEGGER Microcontroller GmbH & Co. KG. It can connect to the PC by using Ethernet (TCP/IP) and

USB. J-Link Pro download speed is up to 3 MBytes/s [10]. The actual speed depends on various factors, such as JTAG clock speed and host CPU core. It can support multiple devices. It has a built-in JTAG TAP controller which is compatible with the standard 20 pin connector. The JTAG 20 pin connection pin-out is shown in figure 3.3.

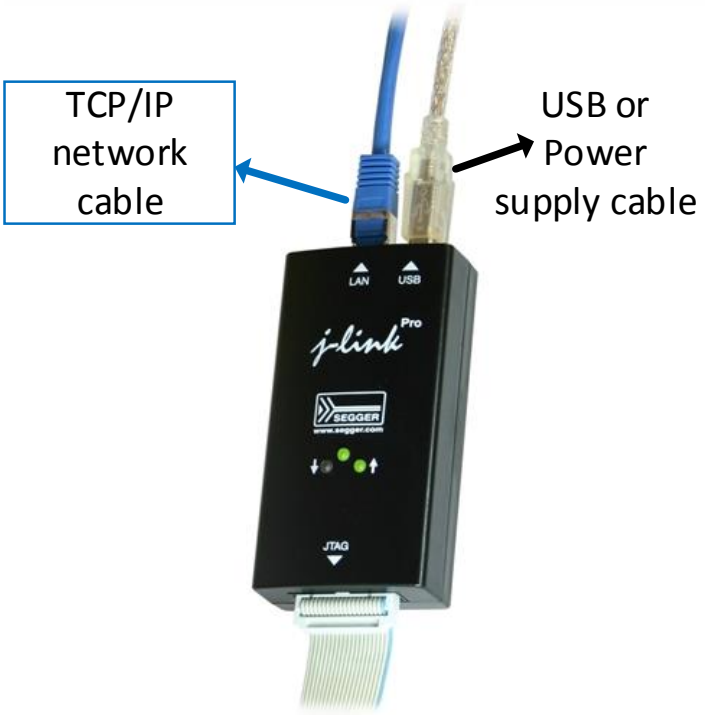


Figure 3.2: SEGGER J-Link Pro V4.00 debugger

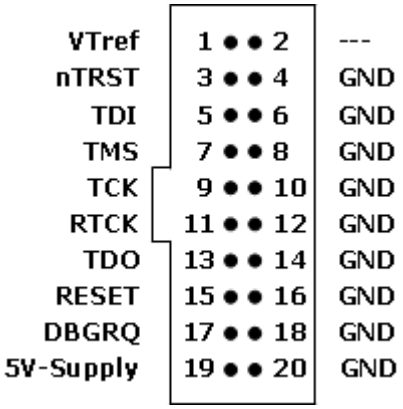


Figure 3.3: J-Link JTAG 20-pin connection pin-out

From the figure 3.3, VTref is the reference voltage for the J-Link device. TDI and TMS are the input signals. TCK is the clock signal which drives the TAP controller. TDO is the output signal, reads out the data. nTRST is the reset signal and used to reset the TAP controller. RTCK is the adaptive clock signal for the J-Link device to provide an external clock for the J-Link device.

The J-Link software development kit (SDK) is used to integrate the J-Link support into the newly created applications [11]. This integration is done by using a standard DLL or a shared library.

The J-Link SDK allows the functionality of the J-Link device for the Vulcan, such as:

- Low-level communication with the target via JTAG commands (establish and terminates the connection and set the JTAG speed).
- Reading and writing into the configuration registers.

For the Vulcan chip configuration, the J-Link Pro device is connected to the PC over Ethernet (TCP/IP) network. The device is configured with an IP address by using a J-Link configuration

tool provided by J-Link SDK. The J-Link Pro device is used for configuration of the Vulcan chip because of its libraries, drivers and compatibility. In addition, all JTAG signals and target voltage can also be measured. The observation and analysis of all JTAG signals are done by adding a measurement adapter to the setup (explained in section 3.1.2).

3.1.2 J-Link JTAG 20-Pin Measurement Adapter

The J-Link measurement adapter provides test points for the JTAG signals to check the JTAG signal integrity. The measurement adapter is connected in between the J-Link Pro device and a target device. The adapter has a standard 20-pin socket to connect with the J-Link Pro device and a 20-pin flat cable is used to connect with the target device (here the Vulcan chip).

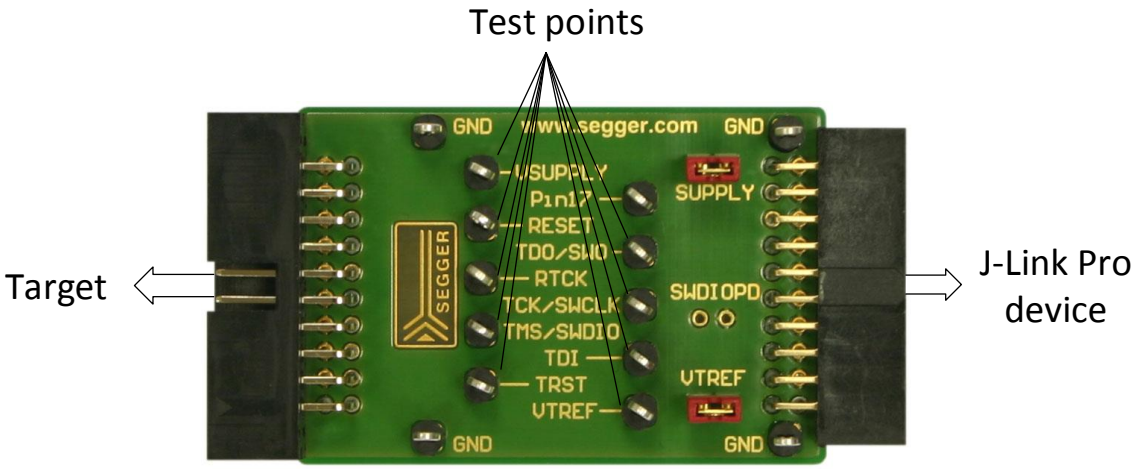


Figure 3.4: JTAG 20-pin measurement adapter

3.1.3 FPGA Evaluation Board

The ZedBoard has an ARM core processor and a FPGA. The ZedBoard FPGA is used for the emulation of the Vulcan chip. The Vulcan JTAG macro and the configuration registers has been flashed into the ZedBoard FPGA (shown in figure 3.5) to develop the *write* and *read* operations of configuration registers before Vulcan was available. The flashing is done by the IC development team and made it as available at run time. If the J-Link JTAG port connects to the ZedBoard JTAG port, it makes the communication with the ZedBoard TAP controller, not with the Vulcan JTAG macro. To avoid this, the general I/O ports of the ZedBoard are used to connect the Vulcan JTAG macro (which is flashed into ZedBoard) with the J-Link Pro device JTAG port. The I/O ports that are assigned to the JTAG signals and the pin numbers of a J-Link JTAG port are listed in table 3.1.

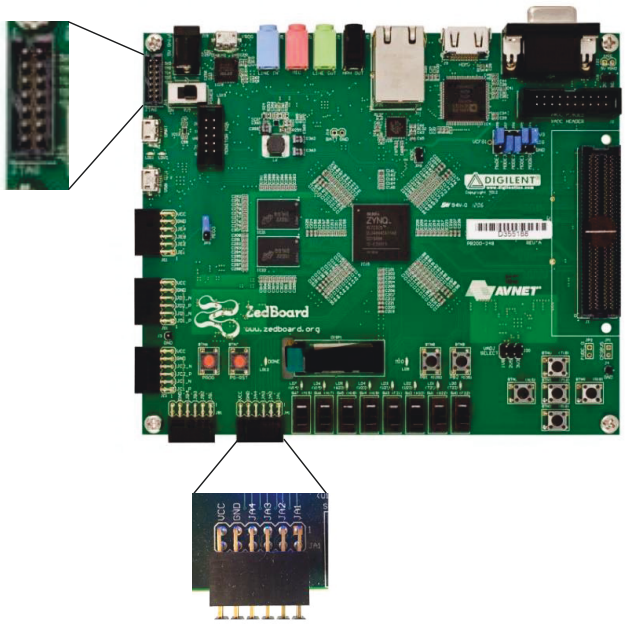
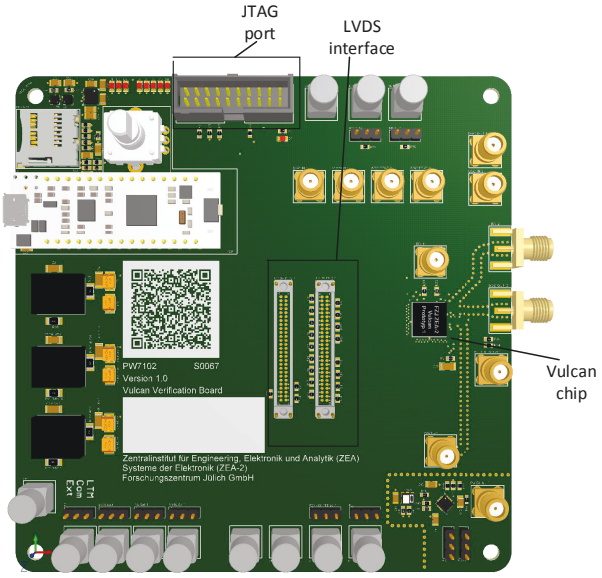


Figure 3.5: ZedBoard FPGA with Vulcan JTAG macro

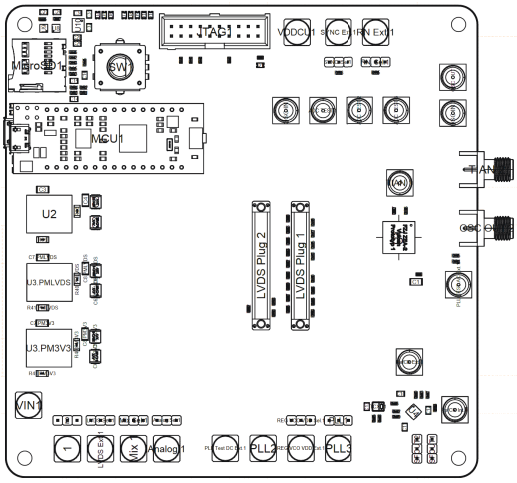
Name of the signal	zedboard I/O Pin	J-Link 20-pin
TCK	JA4	9
TRST	JA10	3
TMS	JA1	7
TDI	JA2	5
TDO	JA3	13
VTref	VCC	1
GND	GND	20

Table 3.1: List of I/O ports and J-Link pin numbers used for the J-Link connection with the zedboard

3.1.4 Vulcan Evaluation Board



(a) Vulcan evaluation board



(b) Layout of the Vulcan evaluation board. The Vulcan chip, JTAG port, LVDS plugs and minimal logic for the chip verification

Figure 3.6: Vulcan Evaluation Board with peripherals

After the emulation of the Vulcan chip, the ZedBoard FPGA is replaced by the Vulcan evaluation board (which the Vulcan chip with some minimal logic mounted on it). The figure 3.6(a) shows the Vulcan evaluation board with JTAG port, LVDS interface and the Vulcan

chip. It is a printed circuit board contains Vulcan chip, micro controller and some minimal logic required to test the Vulcan chip and it is developed at the ZEA-2: Electronic Systems. This board has a 20-pin JTAG port and it is connected to the J-Link pro device by using a 20 pin flat cable.

3.1.5 Saleae Logic Analyzer

The Saleae Logic Pro 8 USB Logic Analyzer [12] is an 8 channel logic analyzer and uses for debugging embedded applications which are acquired protocols like serial, I2C, JTAG or SPI. It is connected to the PC over USB and the Saleae Logic software allows to view and capture the signals [13]. The JTAG signals are collected into the logic analyzer through the test points of a measurement adapter (see in section 3.1.2) and recorded while configuring the Vulcan chip. An example of the recorded activity is shown in figure 3.8.

Figure 3.7: Saleae Logic Pro 8 Analyzer

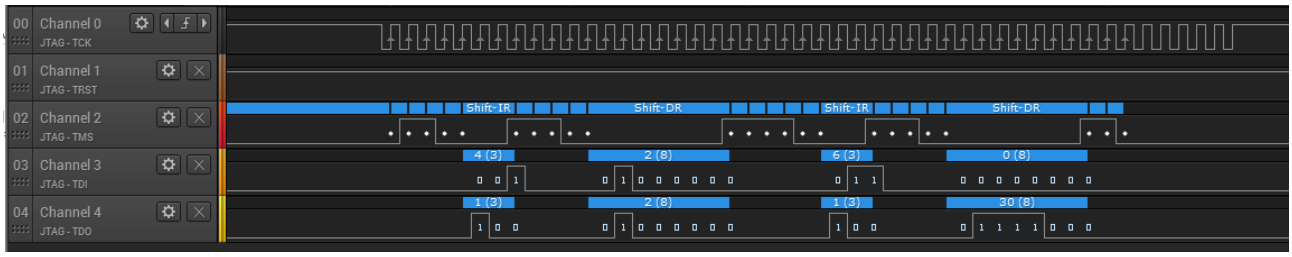


Figure 3.8: Screenshot of a logic analyzer. The response of the JTAG signals while performing write and read operations of the configuration register. The address 2 is written on TDI signal and data 30 read-out on TDO signal

3.2 Verification Test Bench Setup

The verification of the configuration registers *write* and *read* operations is done by feeding in test vectors (instruction and data) to the configuration registers of the Vulcan JTAG macro (flashed into zedboard FPGA) by using the created configuration libraries (listed in the table 3.4). With the configuration libraries, all configuration registers (81 for the digital control unit and 167 for the analog unit) *write* and *read* operations are tested. The verification is done in various aspects, such as register check, bit order and synchronization of TMS and TDI signals with TCK signal. The hardware setup of the configuration system is shown in figure 3.9 and the table 3.2 lists the hardware devices used in the configuration system.

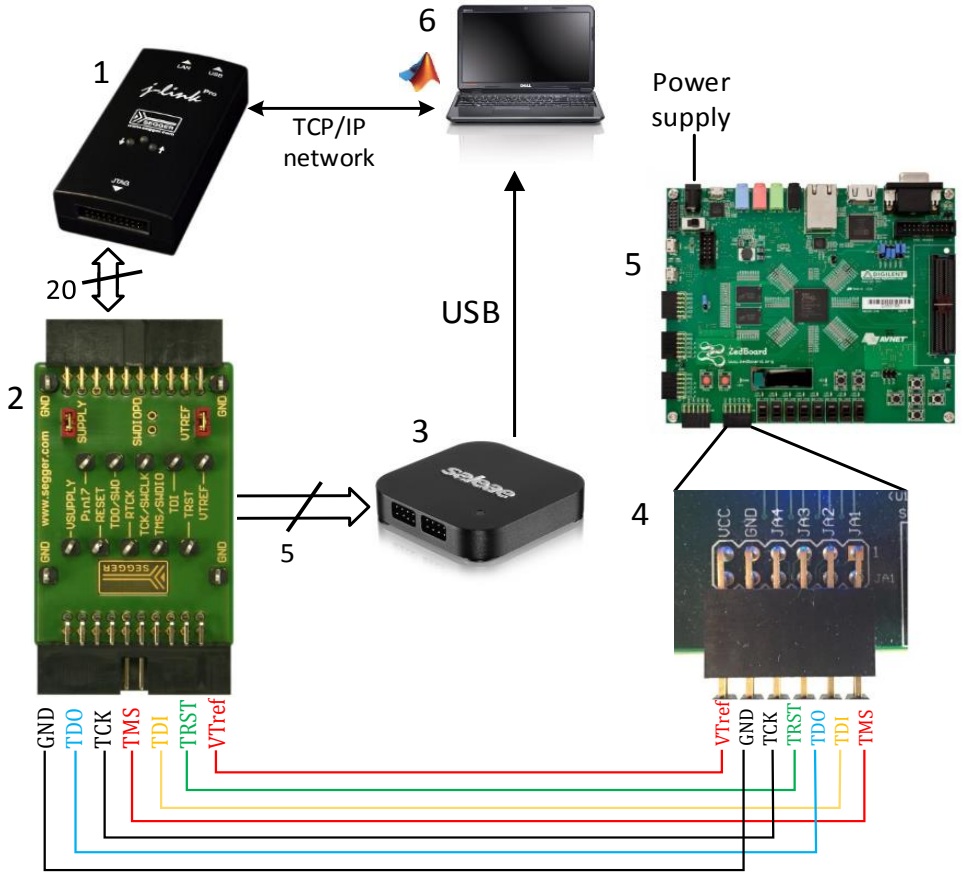


Figure 3.9: The configuration system setup

List of hardware devices	
1. J-Link Pro device	2. J-Link JTAG 20-pin measurement adapter
3. Saleae logic analyzer	4. I/O ports
5. ZedBoard FPGA (target device)	6. MATLAB software tool

Table 3.2: List of hardware devices used in the configuration system setup

3.3 Implementation into MATLAB Environment

The MATLAB is chosen as a test stand for the verification of the Vulcan chip. It has several functions that are needed for programming new applications and mathematical processing. In this work a MATLAB parsing data feature is used in analysis of the Vulcan data.

By using dedicated libraries e.g: Data Link Libraries, it is possible to establish communication with the devices (Logic analyzer, Oscilloscope) that are not directly controllable via MATLAB. The integration of a J-Link device into MATLAB workspace is done via a standard DLL and the DLL provides the C-language application programming interface (API) functions [14] to do the configuration of the chip.

3.3.1 Data Link Libraries and JTAG API Functions

The data link library (DLL) has to be loaded first, to use J-Link and the SDK in a custom application. A JLink_x64.dll library with all API functions [14] is provided by the J-Link SDK and it is used to create configuration libraries (explained in section 3.3.2) for the Vulcan chip. The DLL library is loaded into MATLAB workspace by using a **loadlibrary()** function. The API functions are loaded directly from the DLL with a **calllib()** MATLAB function.

JTAG API Functions List	
API Function	**Description of a function**
JLINK_Open	Opens the J-Link connection
JLINK_Clock	Creates a JTAG clock on TCK
JLINK_SetSpeed	Sets the speed for JTAG Communication (JLink frequency)
JLINK_Close	Terminates the J-Link connection
JLINK_JTAG_StoreInst	Stores a command (instruction) in the output buffer (see in section 3.2.4)
JLINK_JTAG_StoreData	Stores the data sequence in output buffer
JLINK_JTAG_StoreGetData	Stores data in the output buffer and retrieves TDO data from input buffer
JLINK_JTAG_GetData	Retrieves TDO data from input buffer
JLINK_JTAG_SyncBits	Writes out the data remaining in the input buffer
JLINK_JTAG_SyncBytes	Writes out the data remaining in the input buffer as Bytes

Table 3.3: List of JTAG API functions used in the configuration libraries

In order to use the JTAG API functions, the header files provided by the J-Link SDK are included with the DLL library and the API functions are defined as C-declaration functions in header files [14]. Therefore, the equivalent MATLAB notation is used when the API functions are being called. All API functions are defined in 'JLinkARMDLL.h' header file and the

'JLinkARM_Const.h' header file is used as a reference for parameters and function returned values. The 'JLink.h' header file is also used for standard call functions. The table 3.3 lists the API functions which are used to create the configuration libraries.

3.3.2 Configuration Libraries

The configuration libraries developed in MATLAB provide a generic configuration interface which enables the configuration of the Vulcan chip. The figure 3.10 explains about the flow of low level library usage to create the main configuration library. The low level library functions **send_address, write_data** and **read_data** are created: to load an instruction (LOADADD, WRITE and READ), to write data and to read data from the configuration registers respectively. With these low level functions the main configuration library functions **config_register_write, config_register_read, config_register_write_read** and **config_register_write_bulk** are created: to write an instruction and data into register, to read an instruction and data from the register, to write and read an instruction and data from the register and also for bulk instructions (multiple register addresses) along with the data (write, read and write-read) respectively. The configuration libraries are also included the operations: establish or terminate the J-Link device connection and set the J-Link JTAG speed (JTAG frequency). The **jlink_init** function is created to open (and) close a connection and to set the JTAG speed. The other function **load_dll** is created to load the DLL library with the header files into MATLAB workspace.

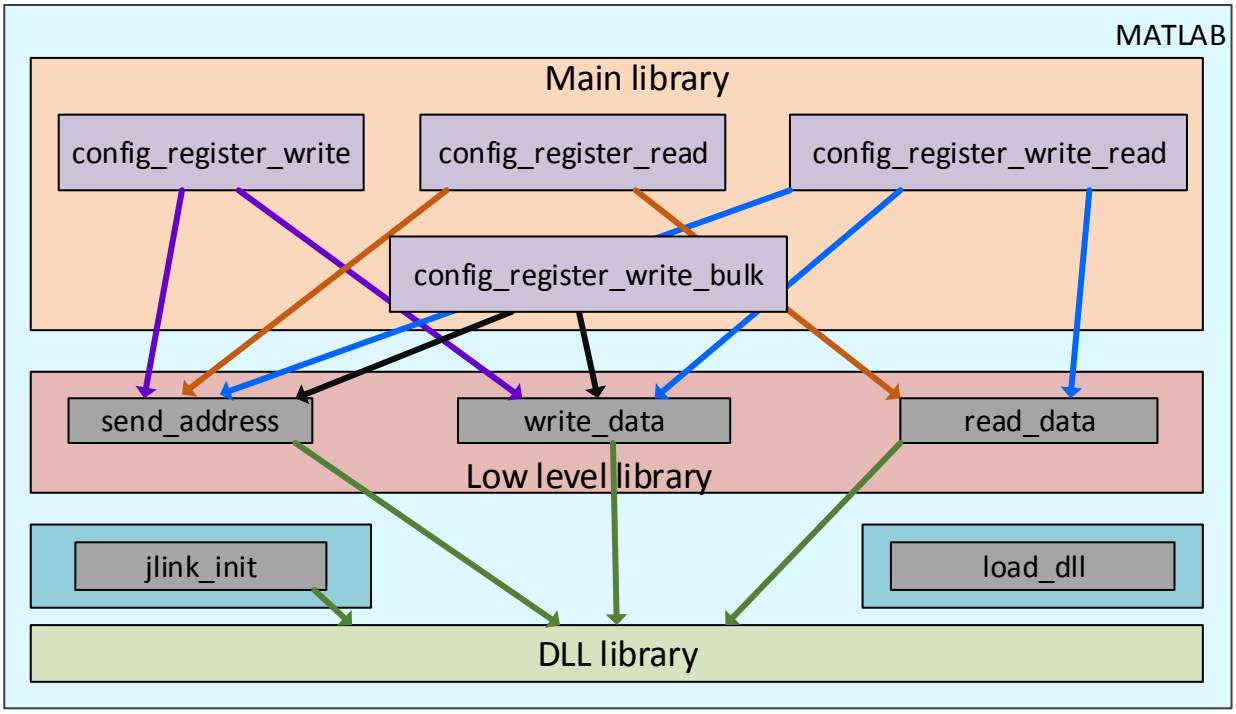


Figure 3.10: The flow of configuration libraries implementation (from top to bottom)

The table 3.4 lists all configuration libraries with a small description. With these configuration libraries, the verification team has developed the advanced libraries to verify the functionality of internal blocks of the chip.

Name of a function	**Description**
J-Link initialization library	
load_dll	loads the DLL library into MATLAB workspace
jllink_init	Opens the J-Link connection and sets the JTAG speed
Low level library	
send_address	loads the Opcode of a LOADADD instruction into instruction register and the address of the configuration register into a data register
write_data	loads the Opcode of a WRITE instruction into instruction register and writes the data value of the configuration register into a data register
read_data	loads the Opcode of a READ instruction into instruction register and reads the data value out from the configuration register
Main configuration library	
config_register_write(address,data)	uses send address and write data functions from low level library to load an address of a configuration register and to write data into configuration register.
config_register_read(address)	uses send address and read data functions from low level library to load an address of a configuration register and to read data from the configuration register.
config_register_write_read(address,data)	uses send address, write data and read data functions from low level library to load an address of a configuration register, to write data into configuration register, and to read data from the configuration register.
config_regitser_write_bulk(registers)	uses send address and write data functions from low level library to load multiple addresses of the configuration registers and to write data into respected configuration registers by looping the steps multiple times.

Table 3.4: List of created Configuration libraries

The table 3.5 lists the examples of configuration of the Vulcan chip by using configuration libraries. The Opcode for all three instructions denoted in binary code format. In config_register_write_bulk(registers) library, the registers indicate the multiple addresses and data values of configuration registers.

Configuration Library	Instruction Opcode			address	data
	LOADADD	WRITE	READ		
config_register_write(2,30)	100	101	-	2	30
config_register_read(2)	100	-	110	2	30
config_register_write_read(5,30)	100	101	110	5	30
config_register_write_bulk(registers)	100	101	-	1,2	2,1

Table 3.5: Configuration examples and information in configuration registers

A sample MATLAB script for the send_address function shows that the way of sending the LOADADD Opcode and address of the configuration register.

```
function send_address(address)
load_instr = 4 ;
instr_ptr = libpointer('uint8Ptr',load_instr);
% load an instruction to load an address of register
calllib('JLink_x64','JLINK_JTAG_StoreInst',instr_ptr,3);
load_address = address;
address_ptr = libpointer('uint8Ptr',load_address);
% load an address of the configuration register
calllib('JLink_x64','JLINK_JTAG_StoreData',address_ptr,8);
end
```

A sample MATLAB script for the write_data function shows that the way of sending a WRITE instruction Opcode and data value into the configuration register.

```
function write_data(data)
load_instr = 5;
instr_ptr = libpointer('uint8Ptr',load_instr);
% load an instruction to write data
calllib('JLink_x64','JLINK_JTAG_StoreInst',instr_ptr,3);
write_data = data;
data_ptr = libpointer('uint8Ptr',write_data);
% write the data into configuration register
calllib('JLink_x64','JLINK_JTAG_StoreData',data_ptr,8);
end
```

3.3.3 J-Link JTAG Communication

The J-Link JTAG sends the data on TMS and TDI signals with the synchronization to the TCK. It has two output buffers to store the TMS and TDI signals and an input buffer to store the TDO signal. The instruction and data sequences stored in the output buffers until

the Synchronization function JLINK_JTAG_SyncBits or JLINK_JTAG_SyncBytes is being called. The instruction and data sequences are stored in buffers from least significant bit (LSB) to most significant bit (MSB) order. The size of each buffer is 1 MByte [14].

3.3.4 Configuration Results

As explained in section 3.1.5, the JTAG signals are recorded by using Saleae logic analyzer while configuring the Vulcan chip. With the recorded activity, the information in instruction and data registers is verified. Thus, the configuration system and libraries are implemented with more reliability for the configuration of the Vulcan chip and to enable the chip for further verifications.

3.3.4.1 Loading an Address

To load an address of the configuration register, the instruction Opcode for a LOADADD is shifted into instruction register through the TDI signal. Then, the address of the configuration register is shifted into the data register through TDI signal. The JLINK_JTAG_StoreInst and JLINK_JTAG_StoreData API functions are used to drive the TAP controller TMS signal into Shift-IR and Shift-DR states respectively. The TAP controller scan for the configuration register which has the same loaded address in the Vulcan chip. The LOADADD instruction has to be loaded first, to perform write (or) read operation.

3.3.4.2 Writing Data

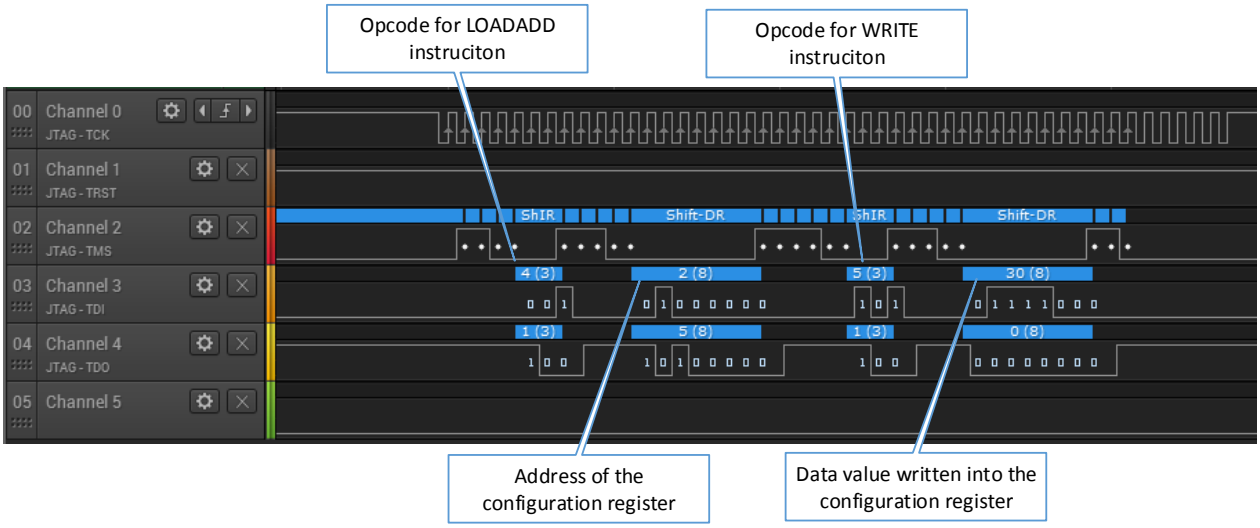


Figure 3.11: JTAG signals response of the exemplary write instruction, recorded by the Saleae logic analyzer

After loading an address of the configuration register an Opcode of a WRITE instruction is

shifted into instruction register to enable the write operation into the configuration register. The data is shifted into the data register and then writes it into the configuration register which is pointed out by the TAP controller. The *write* operation tested result of the configuration register by using the config_register_write library is shown in figure 3.11.

3.3.4.3 Reading Data

To perform the reading operation for configuration registers, first, the address of the configuration register has to be loaded (see in section 3.3.4.1). Then, the READ instruction Opcode is shifted into instruction register to enable the read operation from the configuration register. The TAP controller scan for the selected configuration register which has the same loaded address and sends out the data of selected configuration register on TDO signal. The JLINK_JTAG_StoreInst and JLINK_JTAG_StoreGetData API functions are used to drive the TAP controller TMS signal into Shift-IR and Shift-DR states respectively. The *read* operation tested result of the configuration register by using the config_register_read library is shown in figure 3.12.

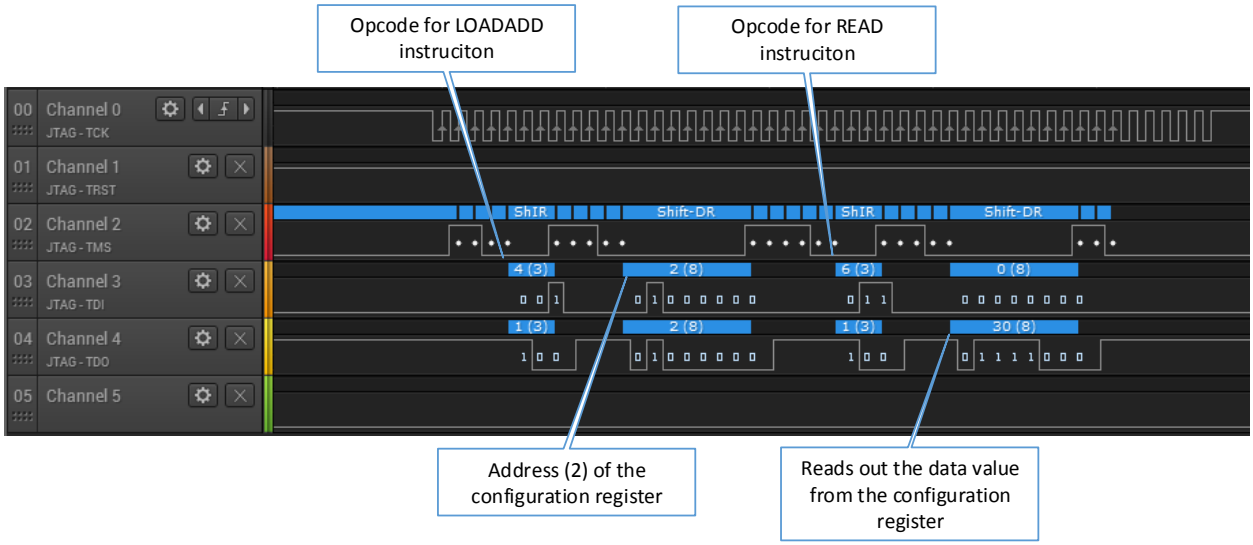


Figure 3.12: JTAG signals response of the exemplary read instruction, recorded by the Saleae logic analyzer

3.3.4.4 Writing and Reading Data

For writing and reading operations, it loads an address of a configuration register, writes the specified data into selected configuration register by using the TDI signal and reads the data from the selected configuration register on the TDO signal. These three operations are explained individually in above three sections (3.3.4.1, 3.3.4.2 and 3.3.4.3). The tested *write* and *read* operations of the configuration register by using config_register_write_read library is shown in figure 3.13.

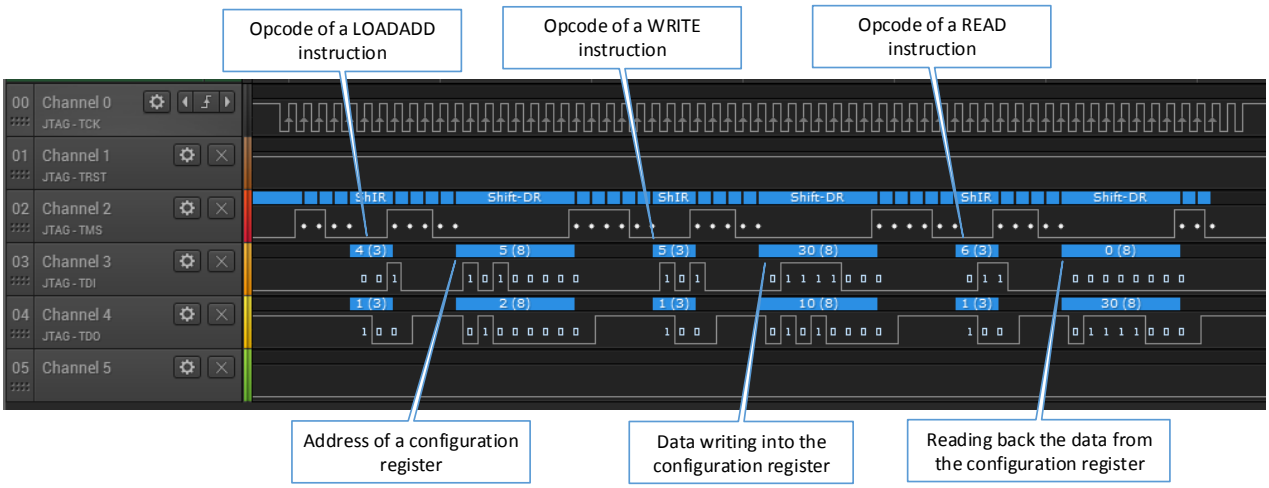


Figure 3.13: JTAG signals response of the exemplary write and read instructions, recorded by the Saleae logic analyzer

3.3.4.5 Writing Bulk Instructions and Data

The process of writing more than one instruction and data values into configuration registers at the same time is the same process like in writing data section (3.3.4.2) but the JLINK_JTAG_SyncBits function is not being called until the specified addresses and data loaded into output buffers of the J-link device. Once it is done with the loading of addresses and data values, it transfers the complete data out in a single attempt. Up to 400 instructions along with the data values can fit in the J-Link buffer. The *write* operation tested result of multiple configuration registers by using the config_register_write_bulk library is shown in figure 3.14.

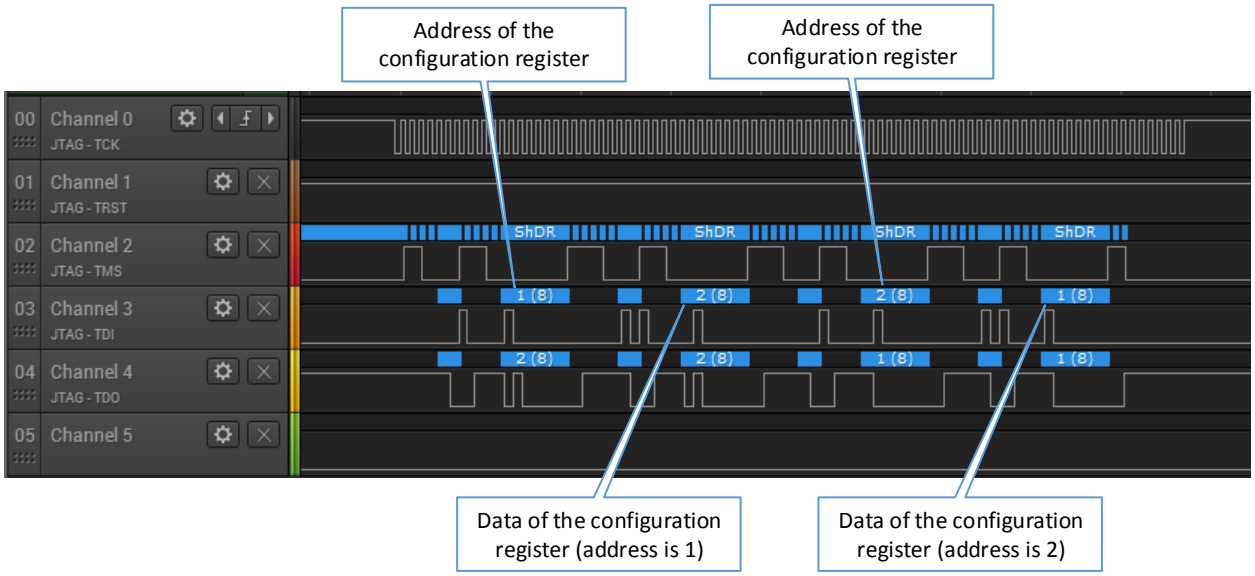


Figure 3.14: JTAG signals response of the exemplary write instruction for multiple configuration registers, recorded by the Saleae logic analyzer

3.3.4.6 Synchronization of JTAG Signals

From the tested results of all operations of the configuration registers, the periodic TCK clock signal generation and the synchronization of the TMS and TDI signals with the TCK signal are achieved. The figure 3.15 depicts the synchronization of JTAG signals.

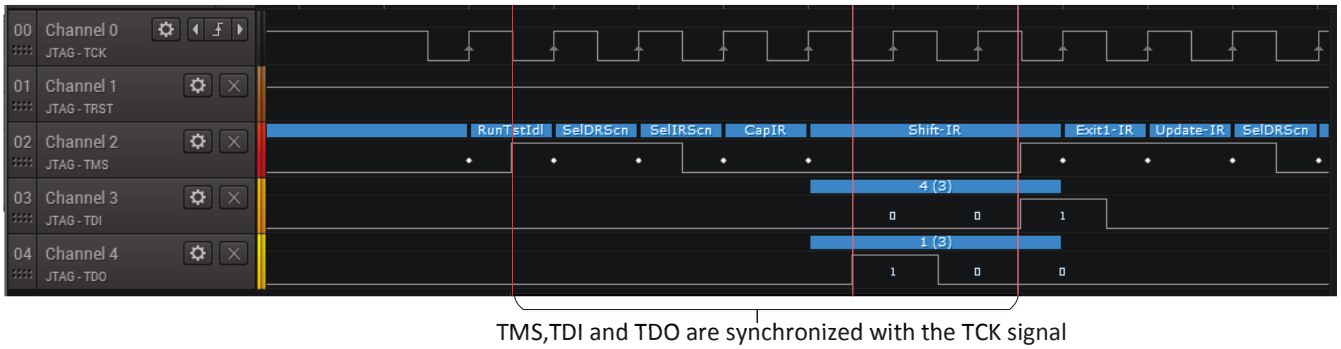


Figure 3.15: The synchronization of JTAG signals, recorded by a Saleae logic analyzer

3.4 Laboratory Results with the Vulcan

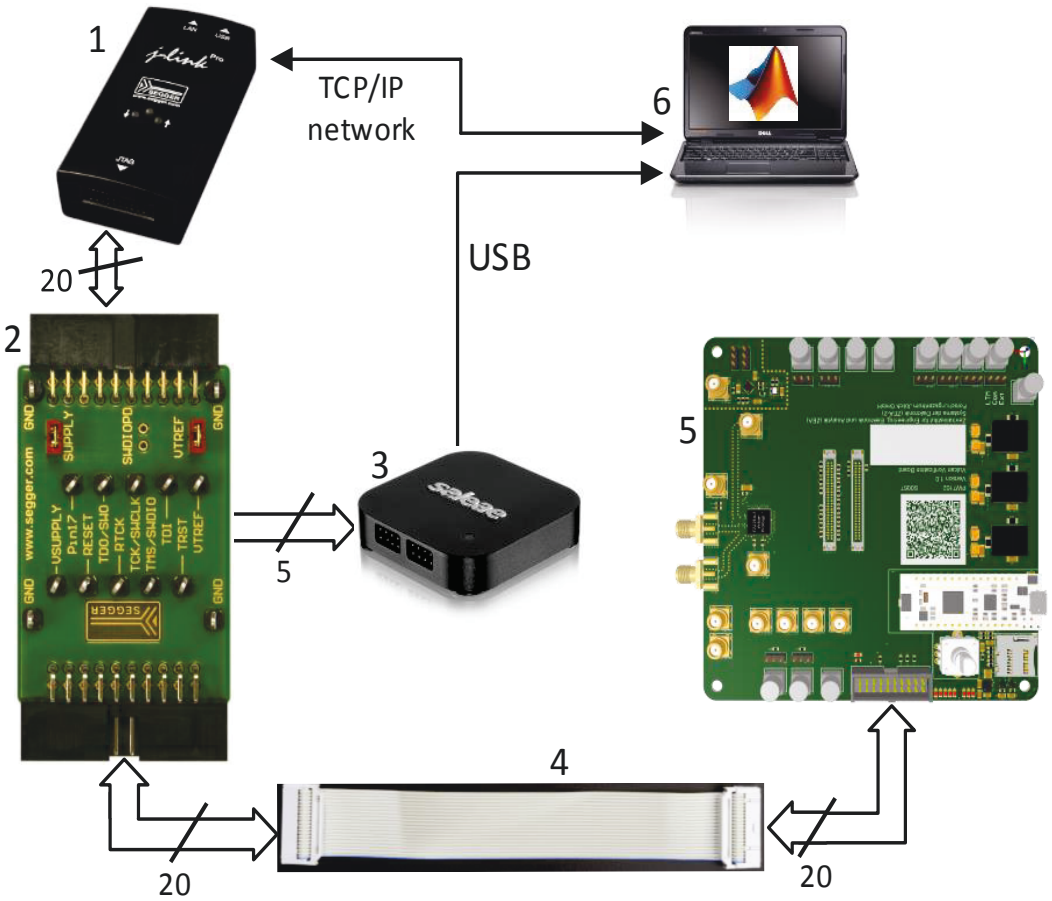


Figure 3.16: Vulcan configuration system setup

List of hardware components	
1. J-Link Pro device	2. J-Link JTAG 20-pin measurement adapter
3. Saleae logic analyzer	4. 20-pin flat cable
5. Vulcan Evaluation Board	6. MATLAB software tool

Table 3.6: List of hardware components used in the Vulcan configuration system setup

The ZedBoard FPGA is replaced by the Vulcan evaluation board (which the Vulcan chip mounted on it) in configuration setup and the updated setup is shown in figure 3.16 and the table 3.6 lists all hardware components used in the Vulcan configuration system setup. In the Vulcan configuration setup (figure 3.16), the measurement adapter directly connected to the J-Link device with 1:1 point contact. The Vulcan evaluation board JTAG port is connected to the measurement adapter by using a 20 pin flat cable.

The instruction and data are fed into configuration registers of the Vulcan chip by using the created configuration libraries (listed in the table 3.4) to test the writing and reading operations in the configuration registers. The loading an address, writing data and reading data operations of the configuration registers are tested and compared with the configuration results in section 3.3.4. An example of the tested configuration operations is shown in figure 3.17.

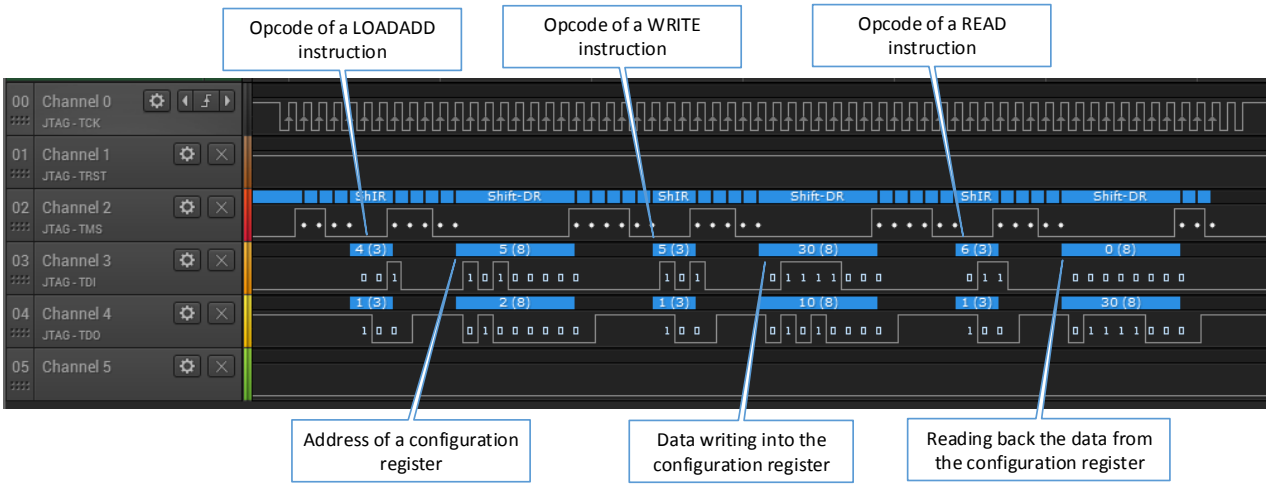


Figure 3.17: Screenshot of a Saleae logic analyzer, response of the JTAG signals recored while configuring the Vulcan chip

While testing the *w*rite and *r*ead operations in all configurations registers by using the created configuration libraries, few random data values (from 1 to 256) are failed to write into all configuration registers. The fails in registers are investigated in different aspects: the error rate over all configuration registers for all different data values and the error rate over different JTAG speeds is also calculated. The additional verification tests are done by measuring the processing time for different instructions (write,write and write-read).

3.4.1 Error Rate Results

The error rate over different configuration registers is calculated as the number of fails to write into the configuration register over the total number of tries to write into the configuration register.

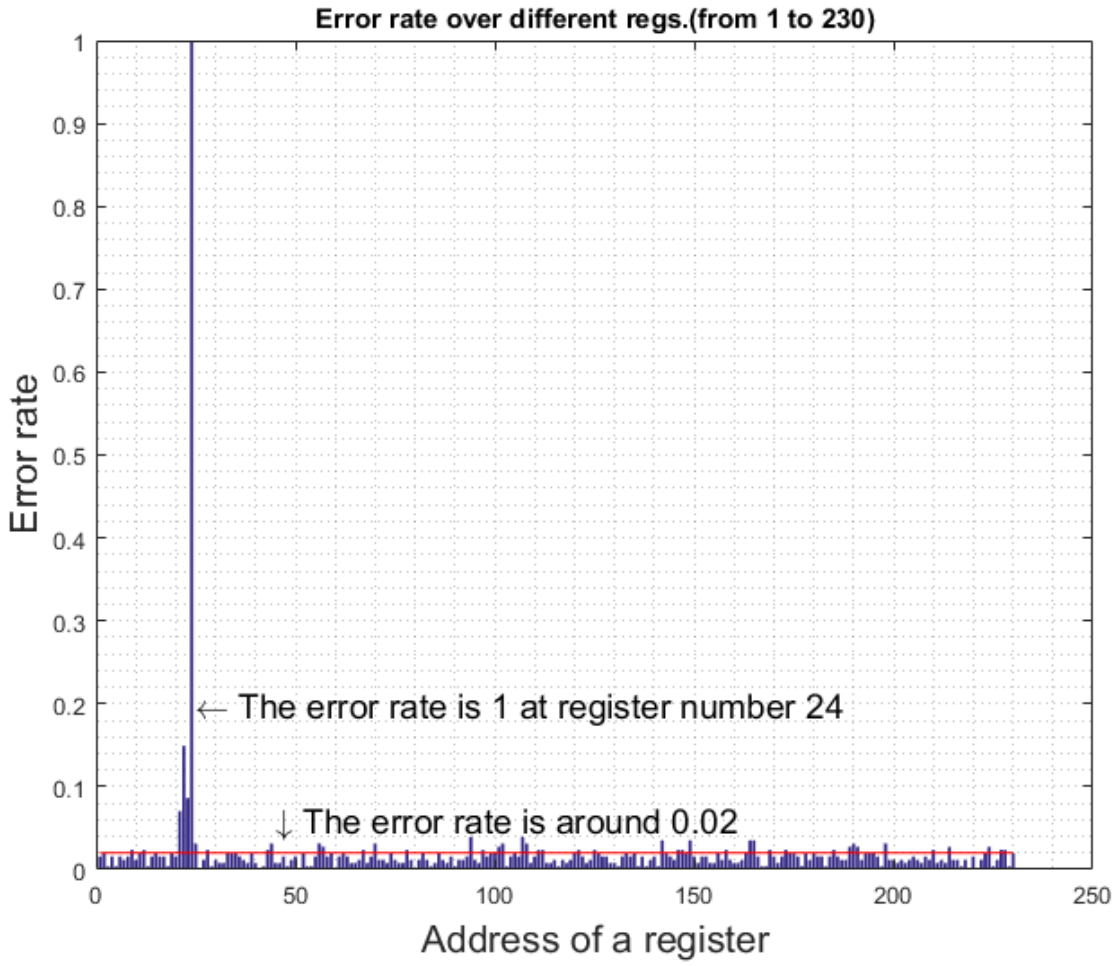


Figure 3.18: The error rate over different configuration registers

From the results in figure 3.18, the register number 24 is completely failing (100% error rate) to perform the *write* and *read* operations and the registers 21, 22 and 23 also have relatively high error rate compared to other configuration registers. The register number 41 (called conf_pam) writes the data values up to 128 only(maximum value is 255). If the data value higher than 128 is written into the register 41, it terminates the configuration process (it will not write any data value further). The error rate is not homogeneous in all registers. Thus the configuration registers are not significant of the fails in registers. Due to the time constraint of the thesis, the reason for the fails is not investigated. The possible reason could be the flip-flop fails and it could occurs because of the setup violation due to different clock phases of the JTAG and internal clock of the chip.

The error rate over different JTAG clock speeds is calculated to check the impact of the JTAG speed on the fails to perform the *write* and *read* operations in the configuration registers. The figure 3.19 draws the error rate over different JTAG speeds. From the results, it is also shown that the error rate is not significant for the fails in registers.

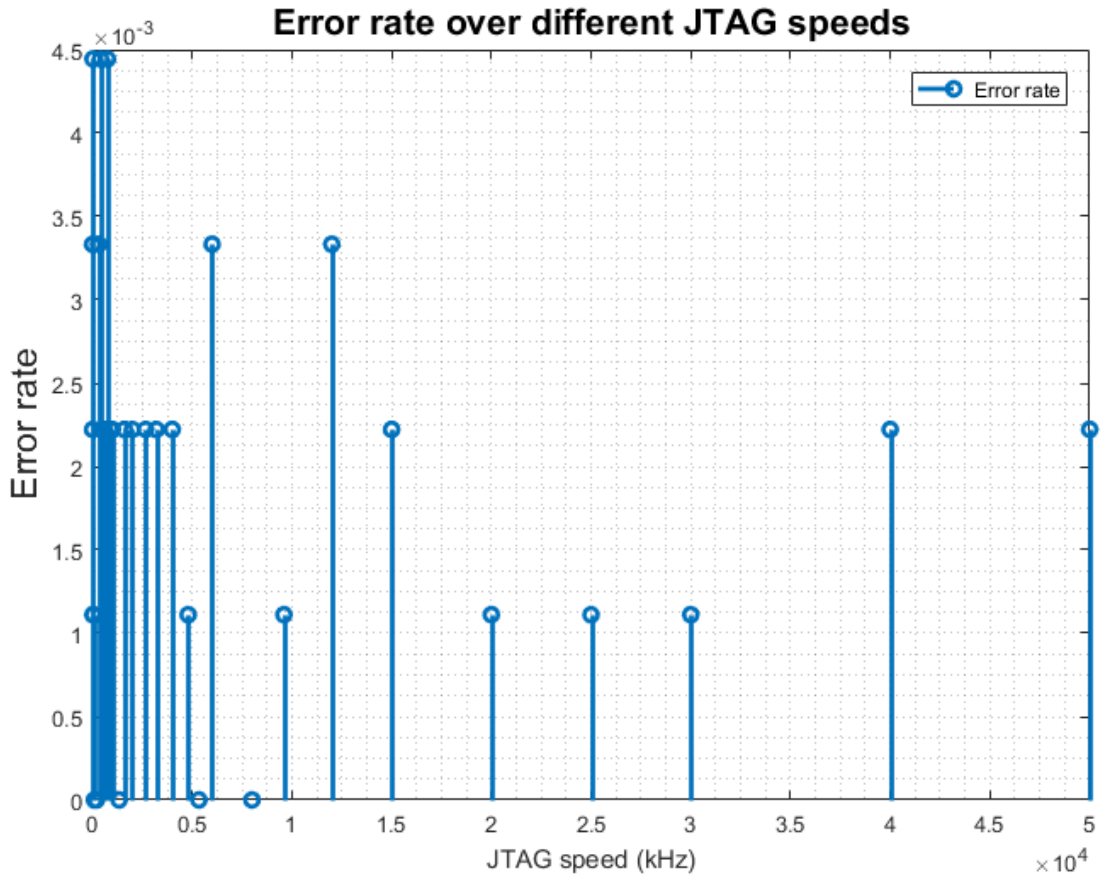
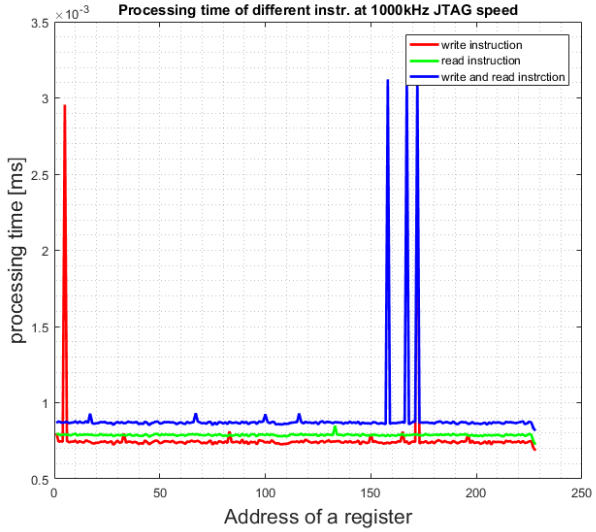


Figure 3.19: The error rate over different JTAG speeds for all configuration registers

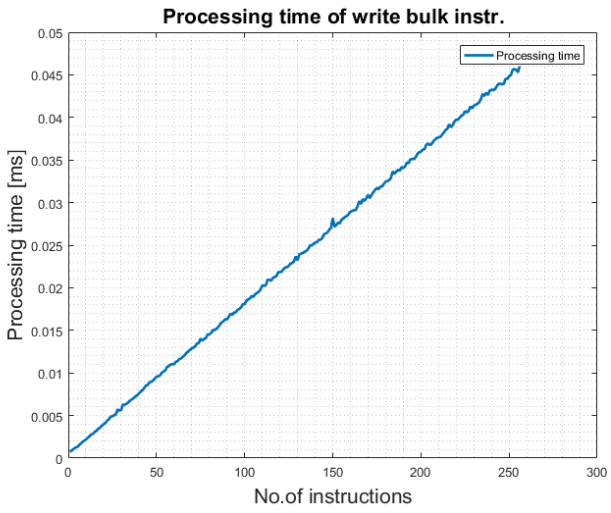
The error rate result over both aspects is concluded that the reason for the fails could be in the design of the chip. The reasons for the fails are the PLL clock was not locked and the J-Link JTAG clock is not synchronized with the PLL clock. The update_DR signal is sent along with the address and data from the JTAG TAP controller to the configuration registers block in the Vulcan chip. But the update_DR signal is some times delayed (not in phase with the digital clock). In result, the fails in registers occurred. To make the JTAG clock synchronization with the PLL clock the 625 kHz JTAG clock signal is applied externally. Thus, the issue of fails in configuration registers is solved and the Vulcan chip is successfully configured.

3.4.2 Processing Time Results

The processing time to perform *write, read* and *write-read* operations for registers (from 1 to 230) with one data value at 1000 kHz is shown in figure 3.20(a). The processing time for bulk instructions (one configuration register address and data values from 0 to 255 in an increment order) is also calculated and shown in figure fig. 3.20(b). The processing time is the time to communicate between MATLAB, Vulcan chip via the J-Link device to perform a single instruction.



(a) Processing time for different instructions



(b) Processing time of write bulk instructions

Figure 3.20: Processing time for different instructions

From the results (in figure 3.20), the processing time to write an instruction and data into configuration register is in an average of 0.74 ms, to read an instruction and data is in an average of 0.78 ms, and to write (and) read an instruction and data is around 0.86 ms. The processing time for bulk instructions is also calculated and it is proportional to the number of instructions performed. In the Vulcan chip the processing time for all configuration registers is not stable (peaks) because of the other tasks performed by the host CPU at the same time. The processing time for different instructions (*write, read* and *write-read* operations) is listed in the table 3.7.

configuration operation	processing time
Write	∼0.74 ms
Read	∼0.78 ms
Write-Read	∼0.86 ms

Table 3.7: The processing time for different instructions

Chapter 4

Development of Data Analysis Functions

In chapter 3, the implementation of the JTAG configuration system for the Vulcan chip and the configuration libraries to configure the chip are addressed. In this chapter, the data flow in the Vulcan chip and extraction of the data in different data processing modes are discussed. The data extraction algorithms are implemented to analyze the data in different data modes.

The data extraction algorithms constitute both interpretation of the data and analysis of the data. The created algorithms compose the data samples in plots with the actual data samples. The redundant data is subtracted from the data samples. The composition method of the data samples is based on the data processing modes of the chip.

4.1 Overview of the Vulcan Data Flow

The main task of the Vulcan read-out chip is to preprocess the recieved analog signals and digitize it and transmits the information to storage without loss of data. The analog unit (AU) digitizes the analog signal and sends it to the digital Control unit (CU). Then the control unit process the data in different data processing modes and transmits the data with a double data rate in selected data modes.

4.1.1 Analog Unit

A transimpedance amplifier converts the current signal which comes from the PMTs into an analog voltage signal. These signals are fed through the ADC. The converter follows the principle of a flash ADC in conversion and it has an 8-bit resolution. Thus, the number of comparators is limited to 255 ($2^8 - 1$) comparators[1]. As explained in section 2.2.2.1 the analog

[1]Number of comparators in a N-bit ADC are $2^N - 1$

input voltage is compared with consecutive reference voltages of each comparator parallel and gives an output as a logic "1"or logic "0". The output of the comparators (thermometer code) is transfer to the digital control unit. Then the digtal control unit process the data in different data modes.

4.1.2 Digital Control Unit

As mentioned in section 2.3, the data samples are transmitted as four samples in one clock cycle. In ADC_Encoder, the thermometer code is converted into the gray coded data with two options (by counting the number of ones and state transition detection in thermometer code). The gray coded data is transferred to the PAM module and it is converted into binary data (8 bit samples). Then, the ADC 1, 2 and 3 samples are assigned to high gain, medium gain and low gain by using the threshold values set by the configuration registers. In normal (PAM) data processing mode the data samples are stored in a ring buffer along with the header information and then transfered to the output. The ADC passthrough (APT) module gets the ADC 1, 2 and 3 samples from PAM module and the assignment of high gain, medium gain and low gain is done as for in the PAM mode. The gain selection is done by using gain source provided on trigger output lines. The digital signal processing (DSP) block provides the three identical trigger output lines. The data streams (4×8 bits) in different data modes (PAM mode, APT mode and scan mode) are transferred to the LVDS multiplexer. This multiplexer selects the data stream from different data modes by using the configuration register and sends the data stream to the output on LVDS bus. The LVDS bus transfers the data (16 bit) with double data rate. The figure 4.1 illustrates the data flow in a control unit.

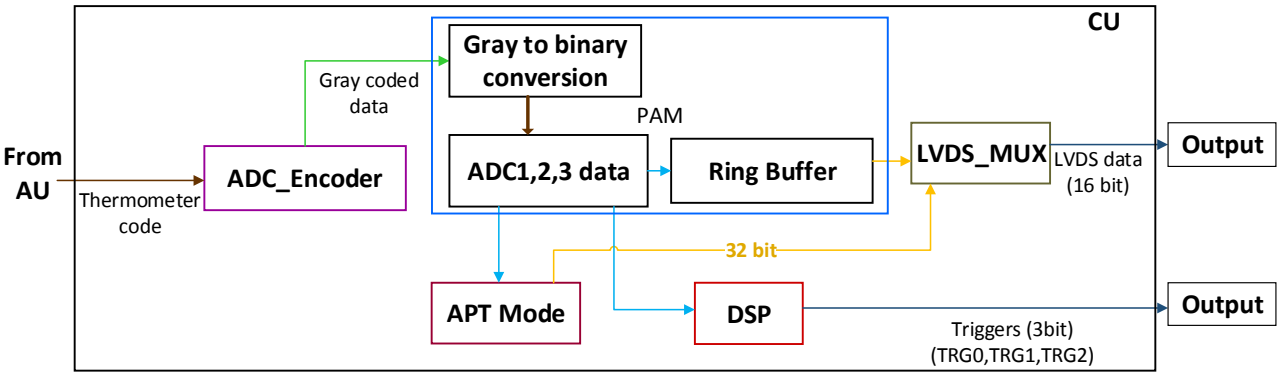


Figure 4.1: Data flow in control unit

4.2 Laboratory Setup

The output data in different data modes of the chip has to be analyzed. For the data analysis, a logic analyzer is added to the chain of hardware setup (figure 3.16) to analyze and store

the data and also to transfer the stored data to the connected PC for further analysis. The schematic of the Vulcan laboratory setup is shown in figure 4.2.

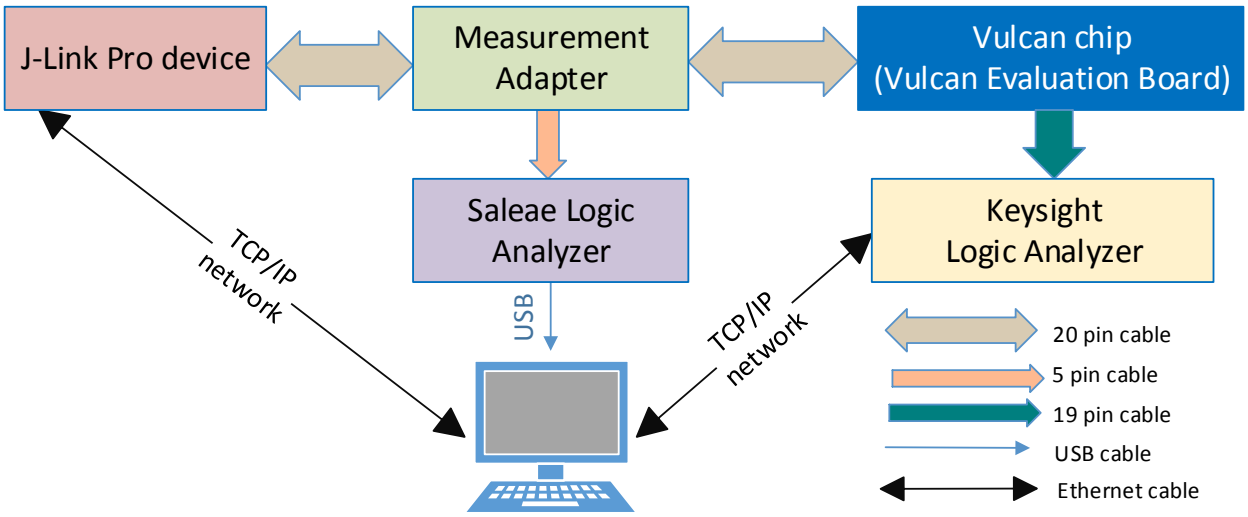


Figure 4.2: Schematic of the Vulcan laboratory setup

4.2.1 Logic Analyzer

A logic analyzer is an electronic device that captures and displays the signals from the digital system. It is useful when the analysis of the timing relationships between the captured signals in a digital system is needed. It connects to a computer over USB or Ethernet connection and delivers the captured signals to the connected computer device.

Figure 4.3: Logic Analyzer U4164A

Keysight U4164A Series modular logic analyzer [15] (shown in figure 4.3) is connected to the PC over a TCP/IP network and controlled via MATLAB. The LVDS interface (shown in 3.6(a)) of the Vulcan chip is connected to the logic analyzer pods by using 16 pin cables. The output (16 bit LVDS data) of the Vulcan chip is sent to the logic analyzer through the pods to analyze and

store the data. The stored data is transferred to the connected PC in a csv file format (explained in section 4.2.1.1). In the logic analyzer different data fields are set to the different data (LVDS data and triggers). The transfered data is used by the created extraction algorithms for further analysis.

4.2.1.1 Data Transfer Format

The Comma Separated Value (*.csv) data transfer format is used to transfer the data from the logic analyzer to the connected PC. In a CSV format the data can be recorded with different fields in a tabular form. Each field is separated by a comma (,) and it used as a data filed separator. This format is not standardized. It can also contain semicolon (;) and quotation marks (") as field separators.

4.3 Parse Data Input

Parsing data means breaking the data fields into the more readable format and assigning variables to the respected data fields by following a set of rules. Thus, it can be easily interpreted, managed and transmitted. The data fields in a CSV file are parsed into three different variables and interpreted while extracting the data.

4.4 Extraction of the Data

The data extraction algorithms are implemented for ADC passthrough (APT) mode, scan mode and normal (PAM) mode to visualize the transferred data in MATLAB. To test the created algorithms, the CSV data files for different data modes were provided by the in-house IC development team from the Vulcan simulations. Then the algorithms were tested with the actual Vulcan data.

4.4.1 Extraction of the ADC Pass-Through Mode Data

The plot for the generated data by the simulation is shown in figure 4.4 and the three trigger lines originate the source of the data samples. With this plot it is ambitious to analyze the data (source of an ADC and data sample value). To make the analysis of the data samples easier, the data samples are extracted based on the trigger lines and plotted in MATLAB by using the created data extraction algorithm.

The data along with the three trigger lines is transmitted from the logic analyzer to the connected PC. The structure of the data transfered in a CSV file is shown in the figure 4.5.

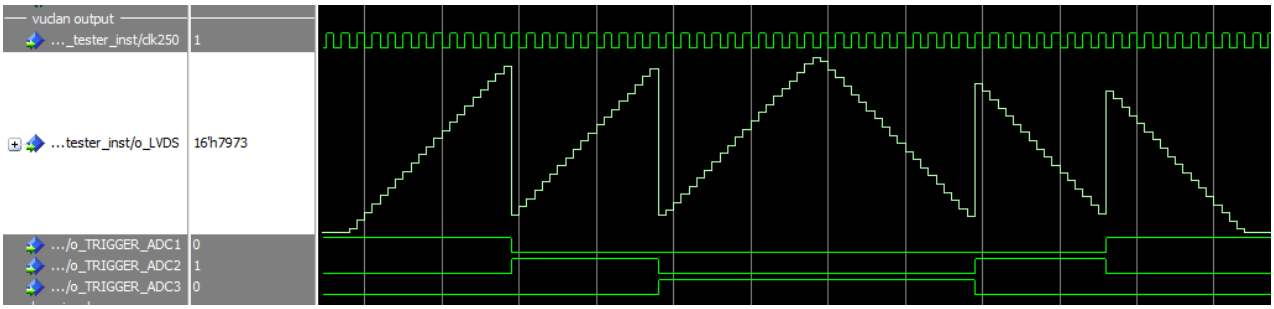


Figure 4.4: Screenshot of the ADC Pass-Through Mode data plot from the Vulcan simulation

Data samples	TRG2	TRG1	TRG0
Data value	1	0	0
Data value	0	0	1
Data value	0	0	1
Data value	0	1	0

Figure 4.5: The structure of the transfered data along with the trigger lines in a CSV file

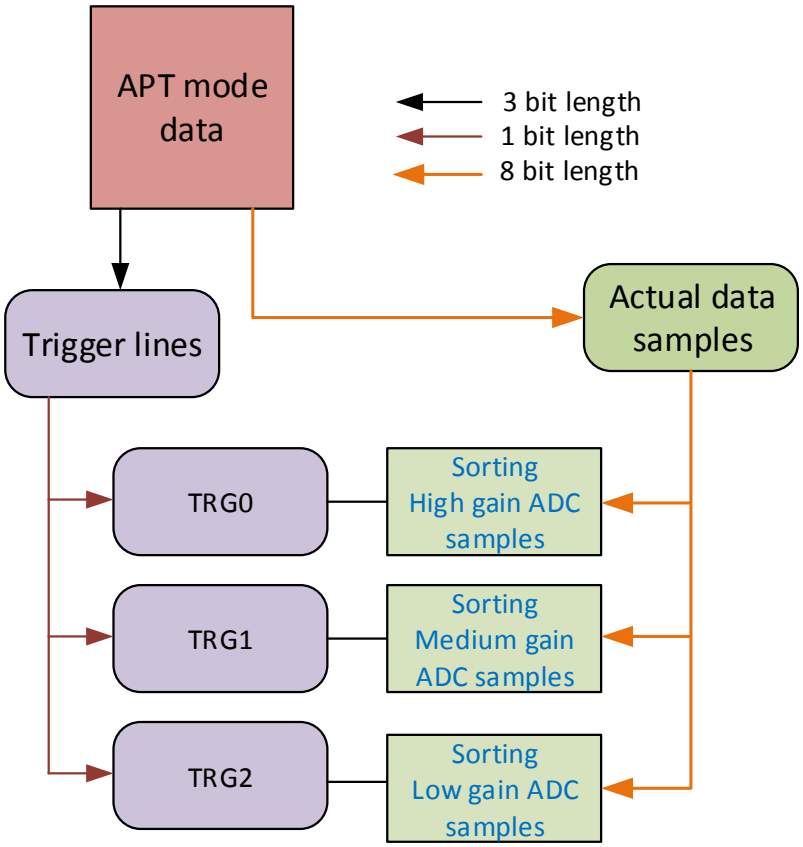


Figure 4.6: The Flow of extraction of the data in APT mode

The flow of data extraction in ADC passthrough mode is shown in the figure 4.6. In trigger data field, the three trigger bits are denoted as three different trigger lines and assigned to three different ADCs. The data samples are compared in such a way: if the first bit is high (TRG0 is 1), the data samples are from ADC1, if the second bit is high (TRG1 is 1) the data samples are from ADC2 and if the third bit (TRG2 is 1) is high the data samples are from ADC3. The data samples are sorted along with the trigger lines for three different ADCs. Only one trigger line is enabled at a time (explained in figure 2.14).

The created algorithm sorts the data and plots the sorted data along with the trigger lines. The figure 4.7 depicts the plot for the data samples extracted from the input data provided by the Vulcan simulation (by the IC development team).

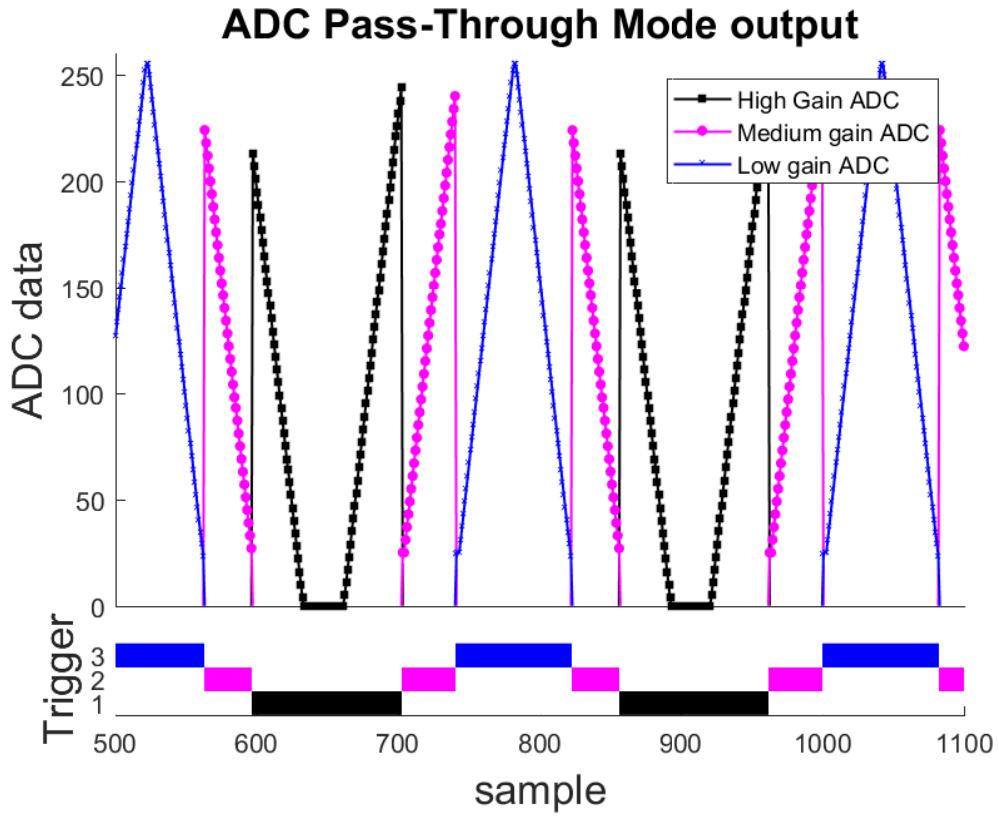


Figure 4.7: Extracted ADC Pass-Through Mode data from the Vulcan simulation

The created algorithm is also tested with the actual Vulcan data. The data transfered to the logic analyzer from the Vulcan chip is shown in figure 4.8. The transfered data is stored in the logic analyzer and sends out to the connected PC in a csv file format for further analysis of the data. From the figure only one trigger line (TRG0) is enabled and it indicates that all samples are from high gain ADC only. The developed algorithm extracts the data samples along with the trigger lines and plots the extracted data.

The figure 4.9 shows the plot for the extracted data samples from the output data of the Vulcan chip. The data samples are sorted and plotted for easy analysis. The extraction algorithm is more efficient for the analysis of large amount of data samples from the chip.

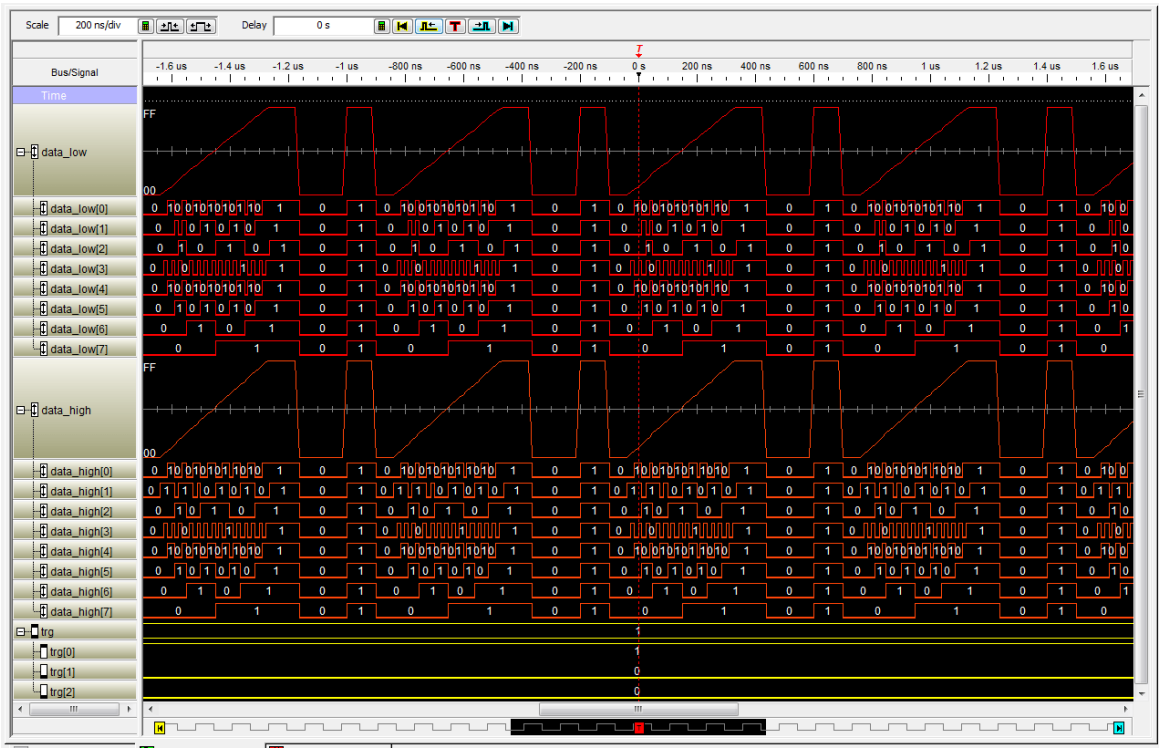


Figure 4.8: A Screenshot of the Logic analyzer for ADC Pass-Through Mode data from the actual Vulcan chip

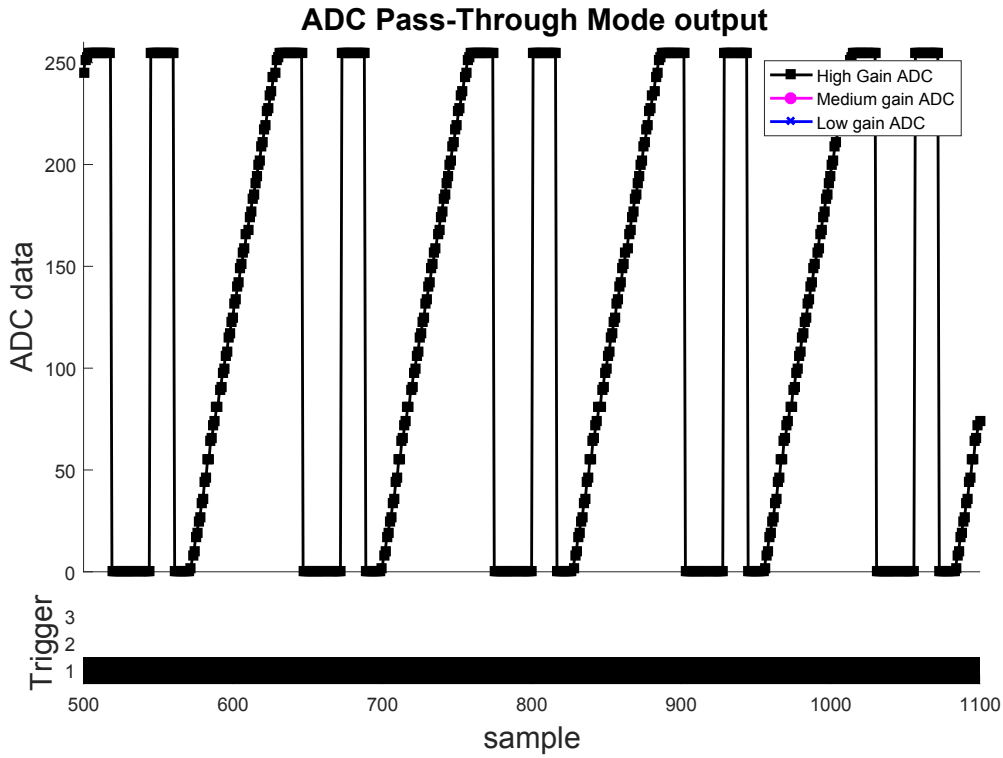


Figure 4.9: A plot for the extracted ADC Pass-Through Mode data from the actual Vulcan chip

4.4.2 Extraction of Scan Mode Data

4.4.2.1 The Intended Method

The data processing scenario in scan mode is explained in section 2.4.3. Each dataset consists of 32 sections (each section consists of four 8 bit samples). All sections in the data set are converted into binary code format. The binary converted sections are sorted as four 256 bit data words (four thermometer code samples). If the data set does not have enough samples (128 samples), the process of sorting data samples will be terminated. The sorted data sets will be plotted in a sequential order. But this mode is not working as planned because of the mismatch in bit assignment in a configuration register called conf_encoder to configure the scan mode.

Scan mode data and ADC passthrough (APT) mode data is sent to the LVDS multiplexer via the APT block. The data from one of these two modes can be selected and sent out. This selection is done by using an enable signal. Because of the mismatch in bit assignment in a configuration register the enable signal was wrongly connected. Due to a swap in the connection of enable signal, the same signal which enabled the scan mode data to be sent out, disabled the APT block. As a result, the scan mode data processing mode is disabled, the trigger line TRG0 is not enabled and no data samples are processed in scan mode. To overcome this scenario a *workaround* method is proposed and the data extraction algorithm is modified to cope with the *workaround* method.

4.4.2.2 Workaround Method

The workaround method is implemented in such a way that the data samples are transmitted always from the high gain ADC of the APT block. The synchronization pattern is generated to identify the start of the data set. The synchronization pattern generation is done in three steps. The first step is, forcing ADC samples to zero for few clock cycles. The second step is, set the first half of the thermometer code sample to sequence of ones and invert it (sequence of zeros) for the second half. The third step is, shut down the inversion and start the transmission of the actual data sets. These three steps are configured in the ADC-Encoder block. The synchronization pattern is transmitted always before the actual data to identify the start of the data set. The data sets are converted from gray coded to the binary format. After the detection of a start of the new data set, the sorting of actual data sets is done same like in intended method (see in section 4.4.2.1). The figure 4.10 shows the plot for data samples from the Vulcan simulations.

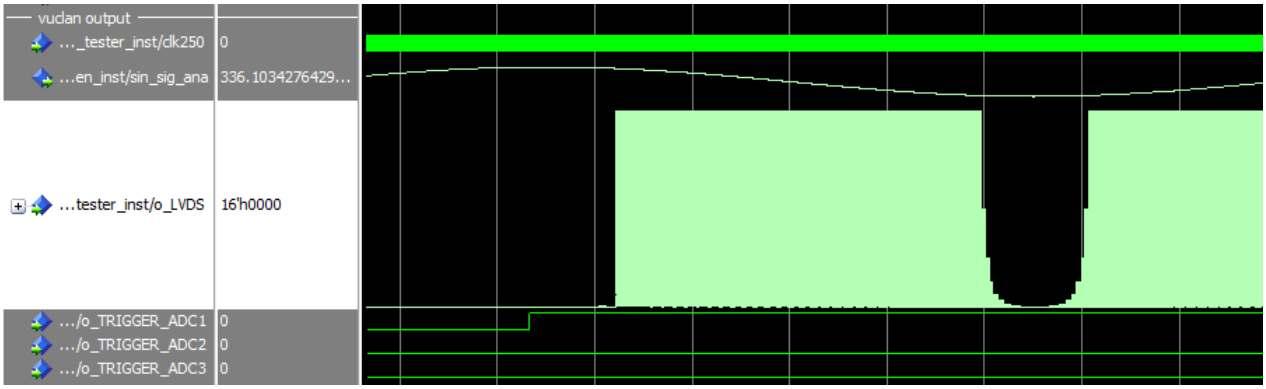


Figure 4.10: Screenshot of the Scan mode data with workaround method from the Vulcan simulation

From the figure 4.10, the data samples with 16 bit length are transmitted and it is very complicated to determine the 256 bit thermometer code samples to analyze the data. Thus, the data sets are extracted and sorted as 256 bit words (thermometer code) by using the created extraction algorithm and it plots the sorted data words in a sequential order. The figure 4.11 shows the flow of data extraction in this data mode.

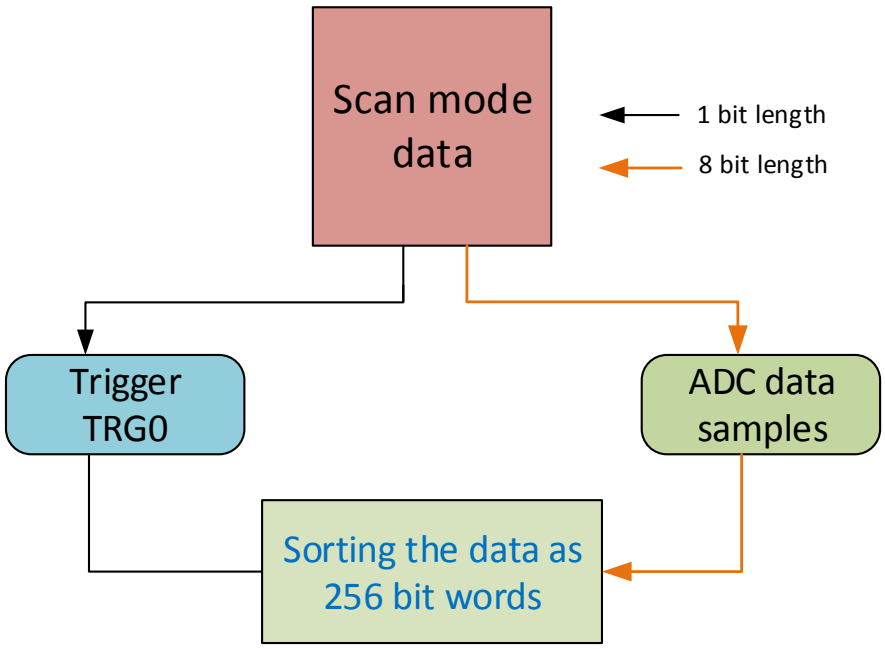


Figure 4.11: The flow of the extraction of the data in Scan mode

The extracted data sets (4× 256 bit samples) for the data samples provided by the in-house IC development team (from the Vulcan simulation) with workaround method are plotted in MATLAB and shown it in figure 4.12. The gray color in the figure represents the sequence of ones and the white color represents the sequence of zeros in a thermometer code.

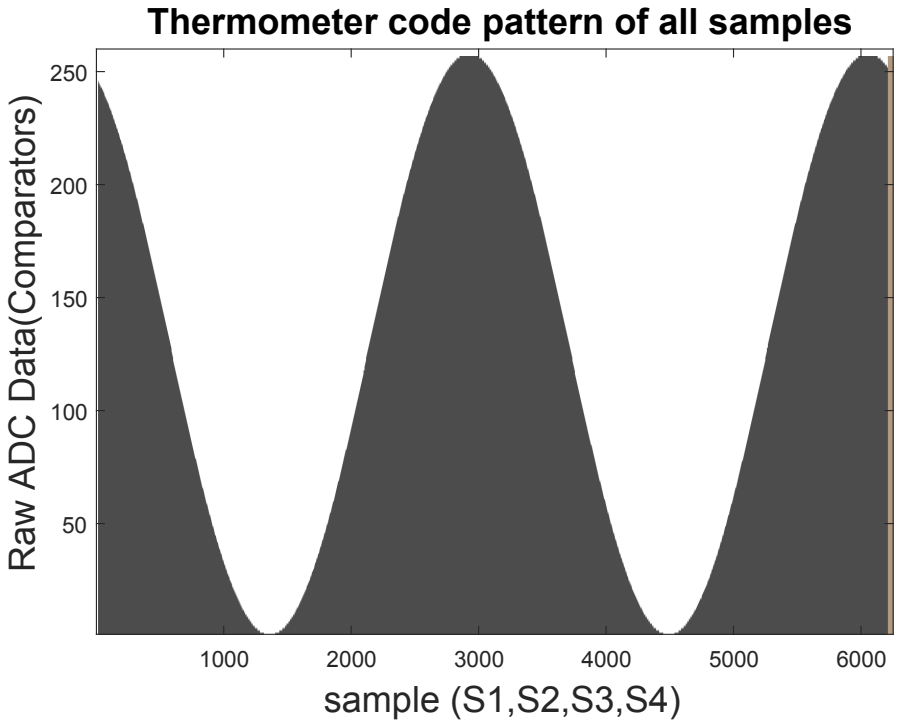


Figure 4.12: Extracted scan mode data with workaround method from the Vulcan simulation

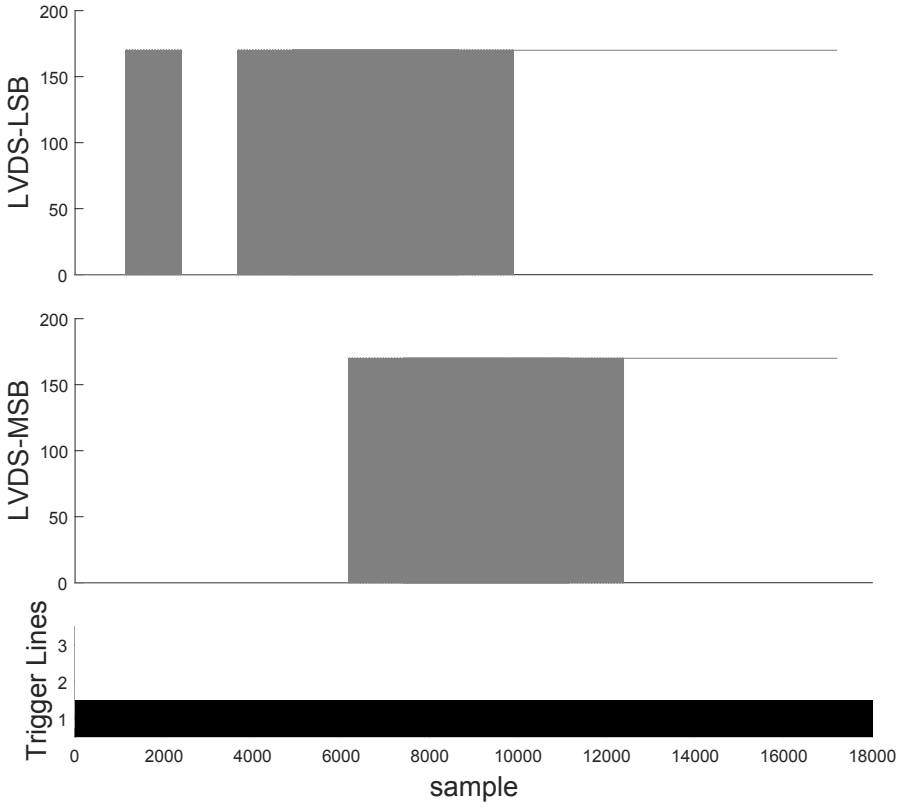


Figure 4.13: The data in scan Mode with workaround method from the actual Vulcan chip before data interpretation

The created algorithm is tested with the data from the actual Vulcan chip. The data in scan mode is transferred to the logic analyzer from the Vulcan chip. The figure 4.13 shows the transferred data before the data extraction algorithm is applied. The data samples are transmitted as 8 bit gray coded data samples (e.g: data value 170 in gray coded is equal to the 255 of the binary format data). In workaround method the data samples are always originates from the high gain ADC only. Thus, the trigger line (TRG0) is always enabled for the scan mode data processing mode.

The data samples are converted from gray coded data to binary coded data, sorted as 256 bit data words and plotted by using the created algorithm. The figure 4.14 illustrates the extracted data sets in scan mode from the Vulcan chip.

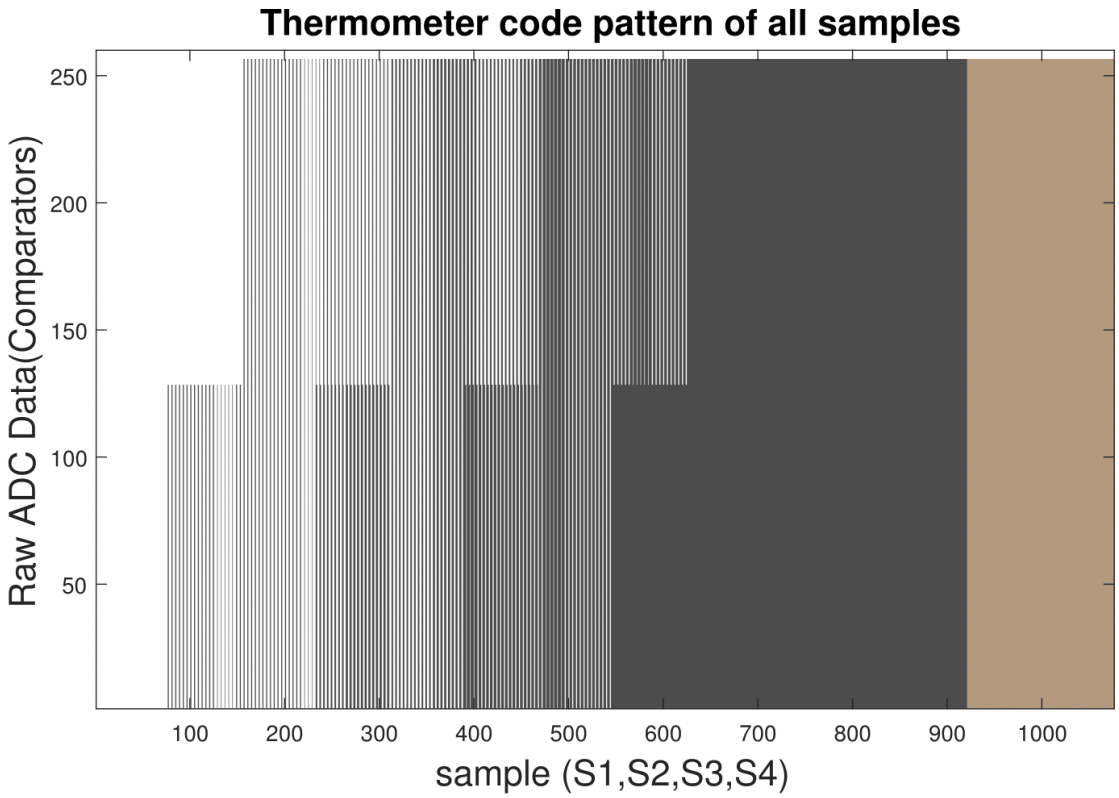


Figure 4.14: Extracted scan mode data from the Vulcan chip

4.4.3 Extraction of Normal Mode Data

The data included with the raw header data generated from the Vulcan simulation (by the IC development team) is shown in figure 4.15. The trigger lines are generated from the digital part of the chip. The extraction algorithm is developed to subtract the redundant information (trigger sequence) from the data and to plot the extracted data.

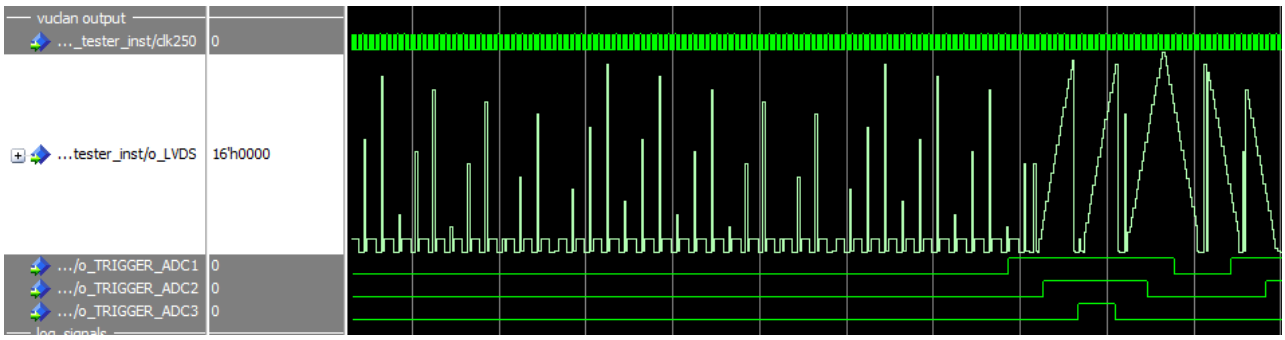


Figure 4.15: Screenshot of the normal mode data from the Vulcan simulation

In normal mode, the data is transmitted along with the trigger sequence (reset signal and header information). The figure 4.16 denotes the structure of the data in normal mode transfered from the logic analyzer to the connected PC.

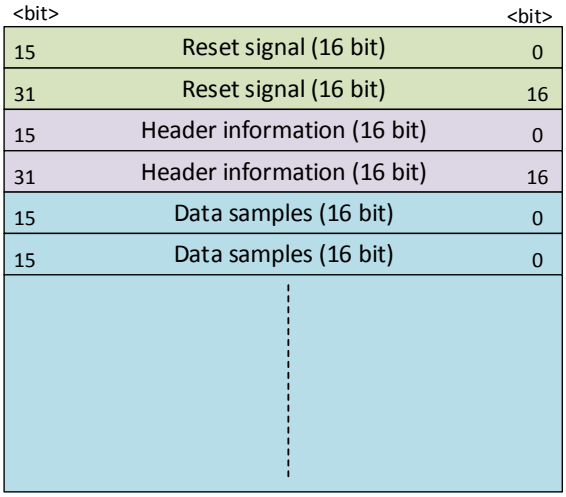


Figure 4.16: The structure of the data along with the trigger sequence in a CSV format file transfered to the connected PC

The process of data extraction in normal mode is shown in figure 4.17. After parsing the data fields into three different variables the four consecutive zero samples (reset signal) are detected from the data samples. After the reset signal detection, the next four samples are considered as the header information. After header information, the data samples until the next reset signal are considered as actual data samples. The actual data samples are extracted and sorted based on the bus mode (4 bits) and trigger (4 bits) values in the header information. The timing

information for every data sample is extracted from the digital counter. Then, the extracted data samples are plotted based on the timing information. The data samples in NOISE, HG, MG and LG bus modes are considered as normal flow state. The data samples in NOISE50, HG50, NOISE80 and HG80 bus modes are considered as overflow state. The flow state indicates the occupancy of the ring buffer. For example, the bus mode is HG50 that means the ring buffer filled 50% and the data samples are from high gain ADC.

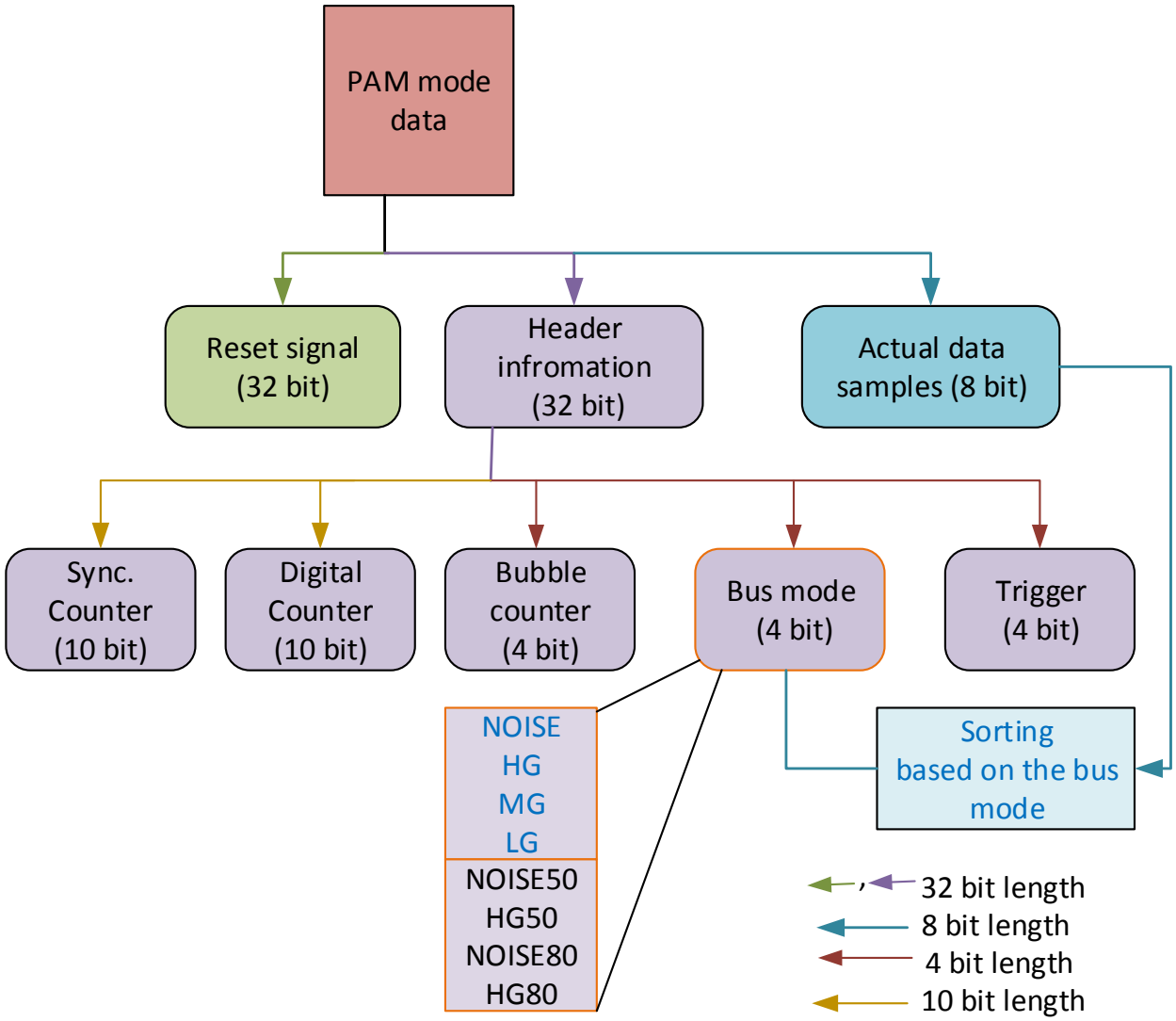


Figure 4.17: The Flow of extraction of the data in PAM mode

The extraction of the actual data samples (without trigger sequence) of the simulated data is depicted in the figure 4.18. In noise mode bus mode the 8-bit data samples are splitted into two 4-bit data samples by the extraction algorithm to emptying the ring buffer. The low gain data is scaled by 100, medium gain data is scaled by 10 and high gain data is scaled by 1 for better analysis with the plot. The flow indicator in the plot denotes the flow state of the data samples (normal flow or overflow).

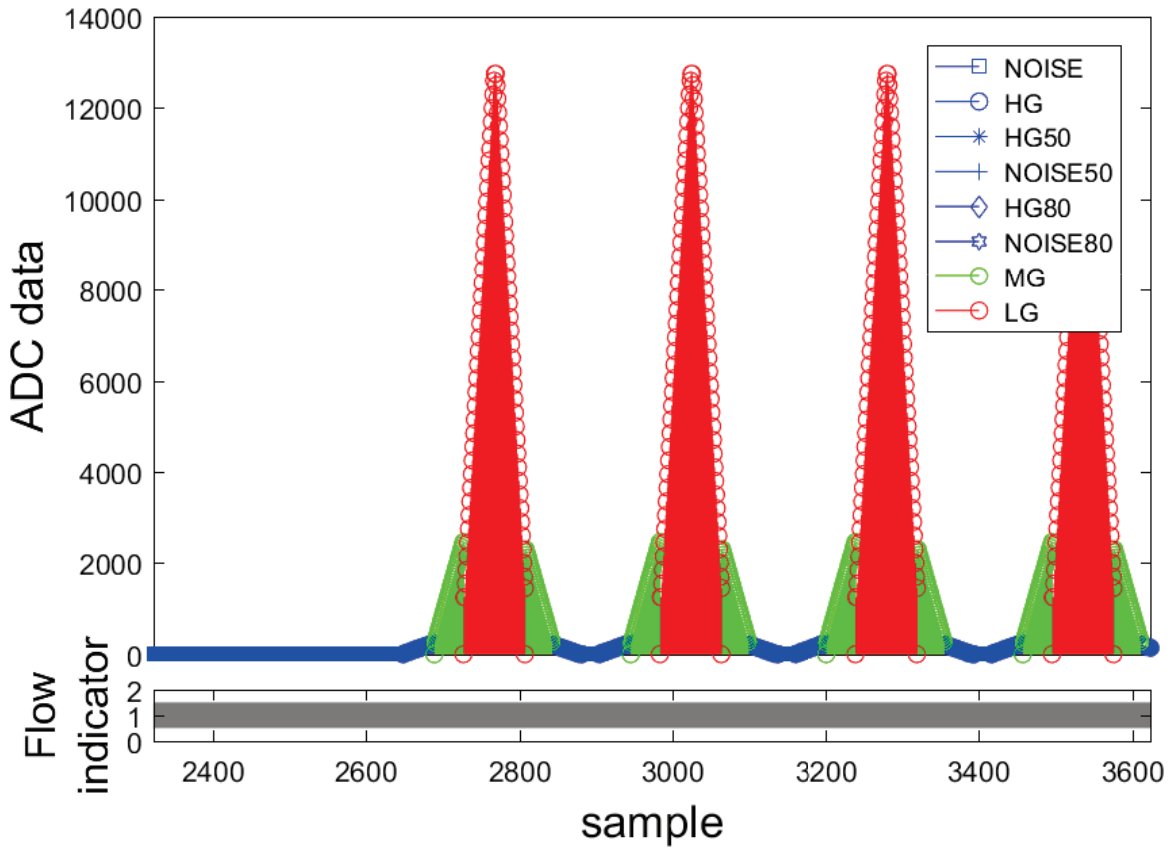


Figure 4.18: A plot of the extracted normal mode data from the Vulcan simulation

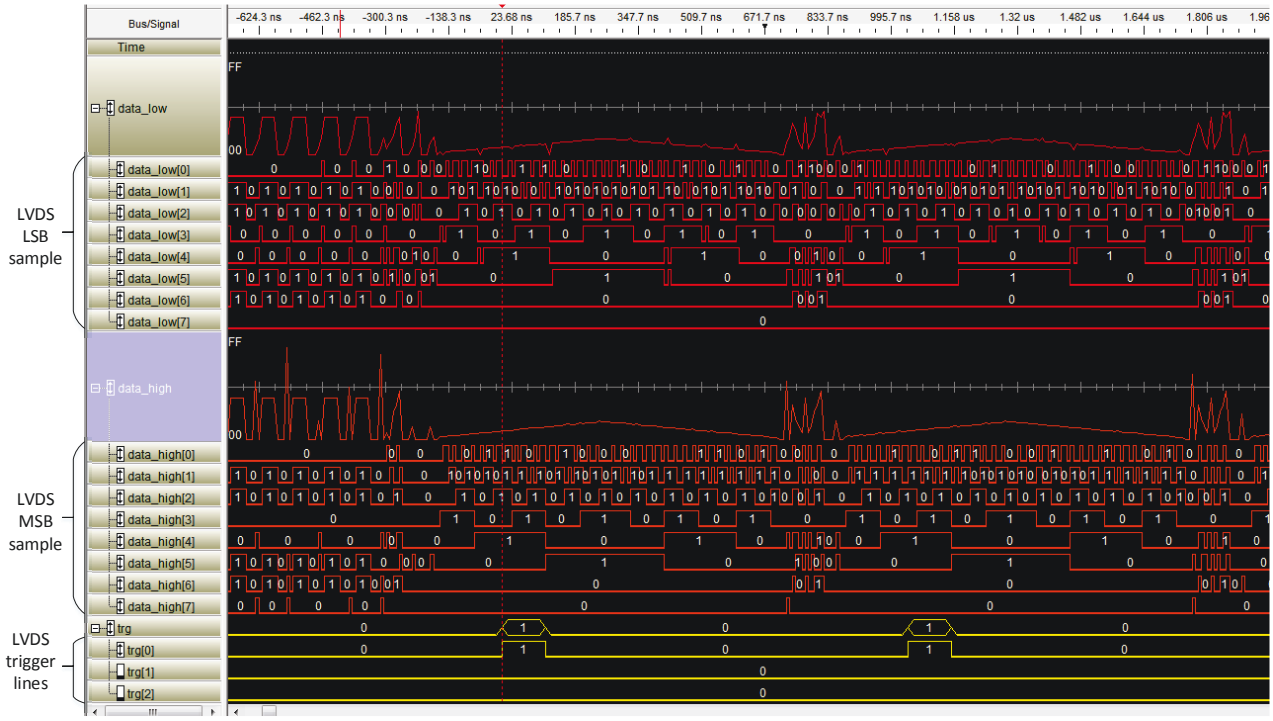


Figure 4.19: A screenshot of the PAM Mode data transferred to the Logic analyzer from the actual Vulcan chip

After the extraction of the actual data from the data provided by Vulcan simulations, the created algorithm is tested with the data from the actual Vulcan chip. The transmitted data with the raw header data in PAM processing mode to the logic analyzer from the Vulcan chip is shown in figure 4.19. The data samples are originated from the noise and high gain bus modes. The plot is deceived with the extra information along with data. The created extraction algorithm is eliminated this extra information from the data for easy analysis of the data.

The extracted actual data from the Vulcan chip is depicted in the figure 4.20. The noise bus mode data samples are splitted into two samples before sorting the data to make the ring buffer empty (explained in detailed in section 2.4.1). The HG bus mode samples are scaled by 10 for the better analysis of the data samples in the plot.

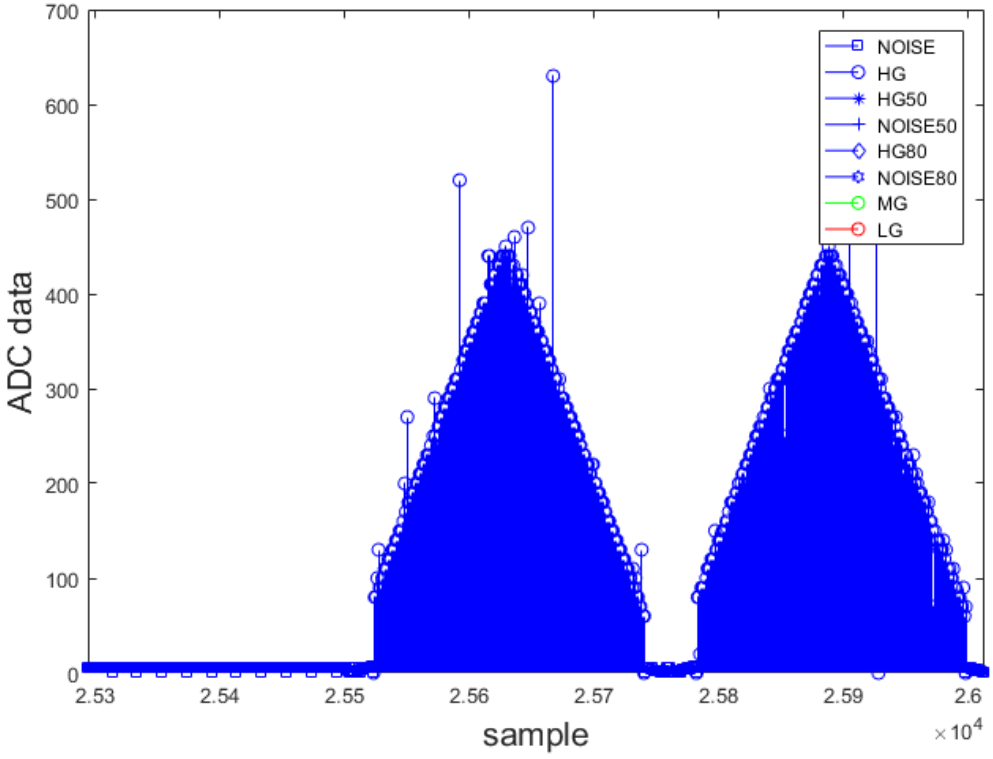


Figure 4.20: A plot for the extracted PAM mode data from the Vulcan chip

The data extraction algorithms for ADC pass-through (APT) mode, scan mode and normal (PAM) mode are implemented and tested with the data from the actual Vulcan chip. The data analysis in different data modes made easier.

The created data extraction algorithms will be used in the automatic data acquisition and analysis environment to make life easier in the analysis of the data from the chip.

Chapter 5

Conclusion and Outlook

5.1 Vulcan Chip Configuration

The aim of the work was to configure the Vulcan chip, enabling it for the verification and data interpretation in different data modes of the chip. In this work the JTAG interface is used in order to configure the Vulcan chip. The JTAG-enabled device called J-Link Pro emulator has been used to perform the JTAG operations for the configuration of the chip. Different MATLAB libraries based on custom instructions (LOADADD, WRITE and READ) have been created in order to perform *write* and *read* operations of the configuration registers in the chip. The developed functions make use of the DLL library and API functions provided by the J-Link SDK. During the development phase, a ZedBoard FPGA has been used to emulate the chip in advance. The Vulcan chip has been successfully configured by using the developed MATLAB libraries. The verification team was subsequently able to develop the verification tests with the help of the created MATLAB libraries to configure the individual blocks of the chip.

The configuration results from the actual Vulcan chip have been compared with the emulated results. It has been shown that a periodic JTAG Test Clock (TCK) signal is generated and that synchronization of the TMS, TDI and TDO signals with the TCK signal has been achieved, thus enabling correct *write* and *read* operations. The processing time for different instructions (write, read and write-read) is calculated. In Vulcan chip, a percentage of writing operations on data values (from 1 to 256) are failed in all configuration registers. The fails are investigated by calculating error rate over different aspects and solved. Thus, the Vulcan chip is successfully configured.

The configuration system which has been set up by using the JTAG interface is used for enabling the chip and for further verification tests.

A description of the principles of the JTAG communication protocol, working experience with the MATLAB and J-Link emulator was obtained through this work.

5.2 Data Extraction

The analysis of large quantity raw numeric data is a challenging task. In order to make the process more efficient, dedicated data extraction algorithms have been developed. In this work the extraction algorithms for different data modes (ADC passthrough (APT) mode, scan mode and normal (PAM) mode) have been implemented in MATLAB. These algorithms have been tested with the data generated by the Vulcan simulation and then with the actual data acquired from the Vulcan chip.

The APT mode extraction algorithm sorts the data samples from all three ADCs based on the trigger lines and plots the sorted data samples along with the trigger lines. Therefore, the analysis of data samples from different gain ADCs is made easier. The scan mode extraction algorithm sorts the data samples as 256 bit data words and plots the data words in a sequential order. Thus, the debugging of comparators in an ADC is achieved. The PAM mode extraction algorithm sorts the data samples based on the bus mode (gain changes) provided in the header information and plots the actual data samples (no trigger sequence) only. Thus, the analysis of data in more detail and the reduction in amount of data but not in the quality of data are gained.

The developed data extraction algorithms are now ready for the implementation of automatic data acquisition and data analysis environment for the Vulcan chip.

5.3 Future Work

The created configuration system with libraries and the data extraction algorithms will be used to accelerate the verification tests of the Vulcan chip.

The processing time[1] of different instructions (write, read, write-read) for all configuration registers is calculated. This processing time can be improved for different instructions by calculating the time to process different instructions only in Vulcan chip.

Enabling to get data in scan mode as for the intended method, the bit assignment issue in configuration register for the scan mode data processing mode should be overcome in the future version of the Vulcan chips.

The data extraction algorithms are implemented only for APT mode, scan mode and normal (PAM) mode. For assistance of improved data analysis the data extraction algorithms for the other three data modes (serial mode, parallel mode and derivative mode) should be implemented in MATLAB. An automatic data analysis and data acquisition environment could be developed by using the created configuration system with libraries and data extraction algorithms.

[1]Processing time is the time to communicate between MATLAB, Vulcan chip via J-Link device

Bibliography

[1] Miao He. JUNO Conceptual Design Report. 2015.
URL: https://arxiv.org/pdf/1508.07166.pdf.

[2] Yu-Feng Li. Overview of the Jiangmen Underground Neutrino Observatory (JUNO). Vol.31 (2014) 1460300 (5 Pages), DOI:10.1142/S2010194514603007. International Journal of Modern Physics: Conference Series.

[3] Neutrino oscillation parameters. [15.02.2107]
URL: http://juno.ihep.cas.cn/ATEjuno/201309/t20130912_109433.html.

[4] Photomultiplier tubes basics and applications. PMT hand book-Hamamatsu. Third edition (Edition 3a). Available at URL: https://www.hamamatsu.com/resources/pdf/etd/PMT_handbook_v3aE.pdf.

[5] Photomultiplier Tube (PMT) coupled to a Scintillator. [20.02.2017] URL: https://commons.wikimedia.org/wiki/File:PhotoMultiplierTubeAndScintillator.jpg.

[6] Tutorial 810. Understanding of Flash ADCs-maxim integrated.
URL: http://pdfserv.maximintegrated.com/en/an/TUT810.pdf

[7] Nicholas Gray. ABCs of ADCs Analog-to-Digital Converter Basics. National Semiconductor, 2003.

[8] JTAG basics. Training JTAG Interface guide. Lauterbach GmbH. Version 26 October, 2016.
URL: http://www2.lauterbach.com/pdf/training_jtag.pdf.

[9] TAP controller state machine. [05.02.2107]
URL: https://de.wikipedia.org/wiki/Joint_Test_Action_Group.

[10] J-Link or J-Trace user guide. SEGGER Microcontroller GmbH & Co. KG. Software Version 6.10a. 05 December, 2016.

[11] SEGGER J-Link Software Development Kit. User guide for the J-Link application program interface (API). Software Version 6.10n. 21 November, 2016.

[12] Saleae Logic Pro 8 software user guide. 29 December, 2014.

[13] Saleae Logic Pro 8 USB logic analyzer data sheet. 2015.

[14] SEGGER J-Link Software Development Kit. User guide for the J-Link application program interface (API). Software Version 6.10n. Chapter 4 (338 - 354). 21 November, 2016.

[15] Keysight U4164A logic and protocol analyzer user guide.

Jül-4402
Juli 2017
ISSN 0944-2952