

Developing Transaction Level Visualization Features For Accelerating pre-Silicon debug

Submitted By

Mrugeshkumar Janakbhai Sabalpara

14MCEC22



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INSTITUTE OF TECHNOLOGY
NIRMA UNIVERSITY**

AHMEDABAD-382481

May 2016

Developing Transaction Level Visualization Features For Accelerating pre-Silicon debug

Major Project

Submitted in fulfillment of the requirements

for the degree of

Master of Technology in Computer Science and Engineering

Submitted By

Mrugeshkumar Janakbhai Sabalpara

(14MCEC22)

Guided By

Dr. Ankit Thakkar



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INSTITUTE OF TECHNOLOGY
NIRMA UNIVERSITY
AHMEDABAD-382481

May 2016

Certificate

This is to certify that the major project entitled "**Developing Transaction Level Visualization Features For Accelerating pre-Silicon debug**" submitted by **Mrugeshkumar Janakbhai Sabalpara (Roll No: 14MCEC22)**, towards the partial fulfillment of the requirements for the award of degree of Master of Technology in Computer Science and Engineering of Nirma University, Ahmedabad, is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Dr. Ankit Thakkar
Guide & Associate Professor,
CSE Department,
Institute of Technology,
Nirma University, Ahmedabad.

Mr. Sasidhar Sunkari
DTS Group,
Intel Technology India Pvt Ltd,
Bengaluru, Karnataka.

Dr. Sanjay Garg
Professor and Head,
CSE Department,
Institute of Technology,
Nirma University, Ahmedabad.

Prof. Dr. P. N. Tekwani
Director,
Institute of Technology,
Nirma University, Ahmedabad

Statement of Originality

I, **Mrugeshkumar Janakbhai Sabalpara**, Roll. No. **14MCEC22**, give undertaking that the Major Project entitled "**Developing Transaction Level Visualization Features For Accelerating pre-Silicon debug**" submitted by me, towards the fulfillment of the requirements for the degree of Master of Technology in **Computer Science & Engineering** of Institute of Technology, Nirma University, Ahmedabad, contains no material that has been awarded for any degree or diploma in any university or school in any territory to the best of my knowledge. It is the original work carried out by me and I give assurance that no attempt of plagiarism has been made. It contains no material that is previously published or written, except where reference has been made. I understand that in the event of any similarity found subsequently with any published work or any dissertation work elsewhere; it will result in severe disciplinary action.

Signature of Student

Date:

Place:

Endorsed by
Dr. Ankit Thakkar
(Signature of Guide)

Acknowledgements

It gives me immense pleasure in expressing thanks and profound gratitude to **Dr. Ankit Thakkar**, Associate Professor, Computer Science Department, Institute of Technology, Nirma University, Ahmedabad for his valuable guidance and continual encouragement throughout this work. The appreciation and continual support he has imparted has been a great motivation to me in reaching a higher goal. His guidance has triggered and nourished my intellectual maturity that I will benefit from, for a long time to come.

I would sincerely like to thank my manager, **Mr. Sasidhar Sunkari** for granting me the opportunity to work at Intel Technology India Pvt Ltd, Bengaluru for my internship. I would also thank my whole team, who spent his valuable time to explain me the entire process and made me feel a part of the company. Also I would like to thank my entire team for the continuous guidance and support and for teaching me to learn beyond the scope of project.

It gives me an immense pleasure to thank **Dr. Sanjay Garg**, Hon'ble Head of Computer Science and Engineering Department, Institute of Technology, Nirma University, Ahmedabad for his kind support and providing basic infrastructure and healthy research environment.

A special thank you is expressed wholeheartedly to **Prof. Dr. P. N. Tekwani**, Hon'ble Director, Institute of Technology, Nirma University, Ahmedabad for the unmentionable motivation he has extended throughout course of this work.

I would also thank the Institution, all faculty members of Computer Engineering Department, Nirma University, Ahmedabad for their special attention and suggestions towards the project work.

- **Mrugeshkumar Janakbhai Sabalpara**

14MCEC22

Abstract

SoC are large designs made by combining other large designs. A typical SoC has many communication pathways and a large amount of parallel activity. In order to debug these kinds of designs effectively new approaches must be taken. We discuss the requirements for effective debug in the face of today's large SoCs, outlining real world example and making some recommendations for easier solutions. This includes transaction based debug, with "connected" transactions. While designing SoC's it also required to view details in depth of part. Zoom Network Diagram viewer is the application which will do this for us.

Abbreviations

MOC	Meta Object Compiler
TLD	Transaction Level Debug
ZNDV	Zoom Network Diagram Viewer.
SoC	System On Chip
JNI	Java Native Interface
IDE	Integrated Development Environment
PDE	Plugin Development Environment
HDL	Hardware Description Language
RTL	Register Transfer Level
IP	Intellectual Property
TLM	Transaction Level Modeling
DEMON	Debug Monitor

—

Contents

Certificate	iii
Statement of Originality	iv
Acknowledgements	v
Abstract	vi
Abbreviations	vii
List of Figures	x
1 Introduction	1
1.1 Company Profile	1
1.2 Team Profile	2
1.3 What We Do	2
2 General Overview	3
2.1 Technology Used	5
2.1.1 QT Framework	5
2.1.2 Eclipse	7
2.1.3 Java Native Interface - JNI	8
3 Transaction Level Debug	9
3.1 General	9
3.2 Overview of System On Chip Level Debug	9
3.2.1 Debug Paradigms	10
3.3 Debug Process through Transaction	14
3.4 Experiment and Development of Drag N Drop	15
3.4.1 Approach - For Drag N Drop	16
3.4.2 Custom editor - Notepad view	22
3.4.3 Email capability to send scratchpad records	23
3.5 Transaction Level Debug Details	25
4 Zoom Network Diagram Viewer	32
4.1 General	32
4.2 Detail Of Zoom Level Network Diagram Viewer	33
4.3 Zoom Network Diagram Viewer Details	34
4.4 Development Task - Zoom Network Diagram Viewer	37
4.4.1 Exploration on how external Qt application to be part of eclipse	37

4.4.2 Conclusion	43
5 Conclusion and Future Work	44
5.1 Conclusion	44
5.2 Future Work	46
Bibliography	47

List of Figures

2.1	Model-View-Delegate(MVC structure)	6
3.1	Simplified view of an SoC interface communication	14
3.2	Initially before dragging data rows from source view to scratchpad view	17
3.3	After dragging data rows from source view to scratchpad view	17
3.4	Example of color & text options	18
3.5	Example of color & text options with overlapping headers	18
3.6	Initially before dragging data rows from log view to proxy view	19
3.7	After dragging data rows from log view to proxy view	19
3.8	Initially before dragging data rows from tree view to proxy view	20
3.9	After dragging data rows from tree view to proxy View	21
3.10	Extending simple text-area to rich text editor	22
3.11	Email functionality for scratchpad records	24
3.12	TLD main window	25
3.13	TLD table view format file display with options	26
3.14	TLD tree view format file display with options	26
3.15	TLD raw log format file display with options	27
3.16	Main window for scratchpad	27
3.17	Scratchpad consisting different views with records	28
3.18	Scratchpad notepad options	29
3.19	Email functionality for scratchpad records to send debugger/engineer	30
3.20	Email records in editable format to debugger/engineer to email box	31
4.1	UML Model for Zoom Network Diagram Viewer	33
4.2	Zoom Network Diagram Viewer Visualization	34
4.3	Zoom Network Diagram Viewer Links and Blocks Connection	35
4.4	Zoom Network Diagram Viewer options available with blocks/svg	35

Chapter 1

Introduction

This chapter explains about the company profile and team profile, and the work we do.

1.1 Company Profile

Intel Corporation is an American multinational technology company. It has head-quarter in Santa Clara, California, which is started on July 18, 1968 by two dedicated scientists, Robert Noyce and Gordon Moore. They founded Intel with an idea and a vision to build products for semiconductor memory. Intel is one of the world's largest and highest valued semiconductor SoC makers. By 1971, Intel had introduced the world's first microprocessor. Which is a pioneer of the x86 series of microprocessors, which can be easily found in most personal computers. Also presented 4-bit to 64-bit microprocessors like Celeron, Pentium, Core, Xeon and Atom series, Intel has established a heritage of innovation that continues to expand its reach to the humans.

1.2 Team Profile

Team under which I worked is responsible for delivers, design and technology solutions that ensure Intel's leadership in product development and silicon technology. We Simplify the system by improving customer satisfaction by providing an interface that is minimal, obvious and clean.

Our department carries several activities :-

- Reactive
 - Support/Bugs Tracking - Provides all kind of support for systems, we develop for use. And take care that all functionality works fine.
- Proactive
 - Path-finding, Research, Development - Higher level always encourage for innovation. Also we carry out development activities for problems and deliver.

1.3 What We Do

We provide solutions for transaction verification in Pre-Si domain. Cost for doing re-engineering after post-Si is much higher than Pre-Si. Which means to do verification before Post-Si, is cost effective.

Chapter 2

General Overview

A system-on-a-chip (SoC) is a microchip which has necessary electronic circuits and parts for a system, to work properly such as one can find in a smart phone gadgets/wearable computer, on a single integrated circuit.

System-on-a-chip technology is used in increasingly complex consumer electronic devices, which are comparatively small in size . Some devices have more processing power capabilities than old desktop computers, so one can imagine. For upcoming time, one can think about SoC-equipped nano robots robots of microscopic dimensions in size might act as programmable antibodies to cure diseases, which are previously incurable . SoC video devices might be embedded in the brains of blind people, which allows them to see and SoC audio devices might allow deaf people to hear. With rapid development in SoC chips, Hand held computers with small whip antennas may someday get enough capable of browsing the Internet at Mbps speeds from any destination of the earth.

Semiconductor industrial growth advancement provided designers to make the most of from increased silicon real estate, to able to get system on a single chip. Such system-on-chip (SoC) designs are made up of many complex functional units, connected via standard bus system or customized one. Functional verification of large SoC designs presents new and interesting challenges to chip development teams. More preciously, simulation at the register transfer level (RTL) now generates tremendous amounts of signal-level data that must be tracked and analysed, for debugging of transactions, which occurs in complex structure of SoC for verification and debug process. Design validation and verification engineers can achieve significant amount of increase in their productivity by analysing

trackers generated from complex functional units.

A System On A Chip: typically uses 70 to 140 mm^2 , of silicon. It is a complete system on a chip. A system consists of memory component, peripherals and microprocessor. Processor can be a customized/assembled or standard microprocessor, or it can be a only made as media processor for sound, modem or video applications. Processors are interconnected using different mechanized techniques, inclusively with shared memories and message-passing hardware components like dedicated channels such as data buses and address buses and mailbox for system messaging. It also possible to have multiple processors and which are accompanied by other generators of bus cycles, such as DMA - direct memory access controllers. DMA controllers can be arbitrarily complex, and are really only distinguished from processors by their complete or partial lack of instruction fetching. SoCs can be easily found-able in every consumer product, like it ranges from modems to mobile phones, DVD players to iPODs and so on as you think!

2.1 Technology Used

2.1.1 QT Framework

Qt is usually a cross-platform request progress composition pertaining to computer, set in addition to mobile. Helped Systems consist of Linux, OS X, Windows, VxWorks, QNX, Google android, iOS, Rim, Sailfish OS while others.

Qt isn't a coding dialect by means of a. It is a platform prepared inside C++. A new preprocessor, the actual MOC (Meta-Object Compiler), is utilized to increase the actual C++ dialect having characteristics similar to signals and also slot machine games [1]. Before the collection phase, the actual MOC parses the original source records prepared inside Qt-extended C++ and also creates normal compliant C++ sources from their store. Hence the actual platform by itself and also applications/libraries making use of it could be compiled by just about any normal compliant C++ compiler similar to Clang, GCC, ICC, MinGW and also MSVC.

Model/View Architecture - QT

MVC[2] consists of three separations or partitioning into three different objects more preciously.

- The Model - application object
- The View - screen presentation
- The Controller - user interface reacts to user input

This object decoupling enable to reuse and provides ease to programmer as don't need to lump all things into just one component and only needed changes are needed to be done.

As you can see model and view are works very closely. Here, in between there is separation done as Model/view architecture, such that combination of objects of view and controller results in MVC architecture. Here data is separated, as how it is rendered and displayed to the end user, but concept is same as for MVC framework. By this separation one can display same data in different views and without any modification to underlying data structures. But here user can modify data as per convinces and to handle that Qt introduces the concept called delegate, Which allows to modify data when user want at anytime, user make changes to view. Here, delegate raise the signal to contacting model and telling model to change the data as per edited in view, which again displayed back to view. SO, user can able to see changes at same time. Thus, rendering of data gets easier.

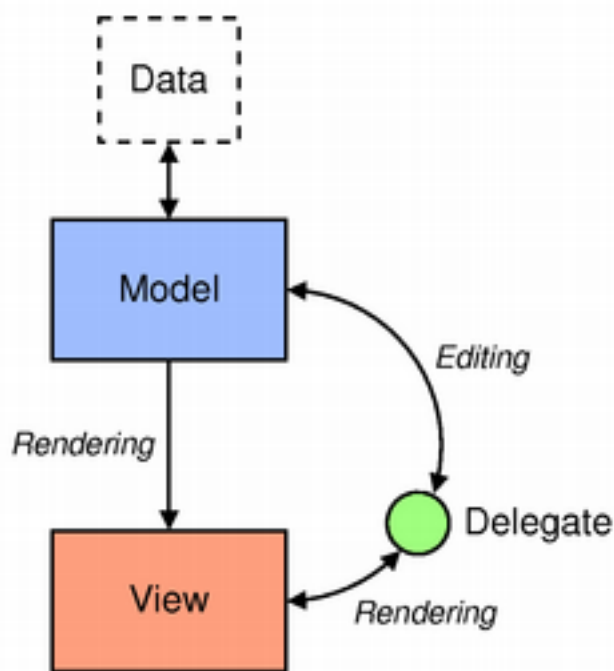


Figure 2.1: Model-View-Delegate(MVC structure)

- Models, views, and delegates are getting connected by signals and slots mechanism:
 - Signals generated from model initiates and view get to know about changes to the data done by the data source.
 - Signals generated from view provide information about the user's interaction with the items which are displayed on different views.
 - Signals generated from the delegate are get initiated when some editing done, so it generates signal and model get to know and view gets change accordingly.

2.1.2 Eclipse

The Eclipse provides the integrated plug-in development environments (IDEs) Platform and arbitrary tools. Here I provide some details to the Eclipse Platform, where it is used and from where it originated.

- Eclipse is an Open Source IDE :-
 - It is used as general purpose open platform which provides and leverage us for development of third party plug-ins, which might be written in other programming language.
 - It is well know as an Integrated Development Environment(IDE)
- It Provides tools which are useful for building, debugging, running and coding applications
- Originally Eclipse designed for Java only , but later on it added support to other languages also such as
 - C, C++, which are native languages
 - Ruby, PHP, Python, etc, which are scripting language
- In our project, I need to explore the different ways, that how Qt based application can be integrated to eclipse as a perspective, or by providing it as a plugin, so it can be installed and used as a part of eclipse environment.

- To achieve purpose of providing application as an plugin for eclipse, I need to learn about plugin development using the plug-in development environment (PDE) provides tools to build, develop, test, deploy, create and debug eclipse plug-ins and features.

2.1.3 Java Native Interface - JNI

JNI enables us like programmers to write native methods/functions to handle many situations such as when an application not written entirely in the Java programming language, means it happens that many standard Java classes are not supported. It is also used to change an existing application written in native language which need to get extension to Java to be accessible to Java applications. Many of the standard library classes relies on java native interfaces(JNI) which provide functionality to able to use those libraries in application to developers/programmers, e.g. memory organization, file I/O [3]. JNI Involves performance- and platform-sensitive API implementations in the standard collection, that allows java applications to access that operation in the safe and platform-independent manner, developed within various other dialects.

The JNI framework provides us basic method to use Java objects in the similar kind way that Java code uses these objects. A native method can create Java objects and then checks and using these objects it performs tasks. A native method can also check and use objects created by Java application program.

Chapter 3

Transaction Level Debug

3.1 General

When a large SoC is being tested, the communication on the SoC can be modeled using transactions. Communication between two points can be modeled a transaction. The collection of related transactions that make up a "complete transaction" or data transfer across the entire SoC is a transaction. In a simulation there may be hundreds of thousands or millions of transactions. This is a large dataset to use for debug. It is hard to find the transactions that are deemed "interesting".

3.2 Overview of System On Chip Level Debug

Traditional register transfer level (RTL) [4] debugging is based on effective simulation results on structural informative connectivity of the Hardware Description Language (HDL) source. It's seems to be very effective process and helpful though it is not that much helpful to reasoning about how and why questions of designer. So forth designer also already have its mirror image as how it going to be propagate in simulation and it should be work as it is defined. But when designs get more and more complex, there is a need emerges to have some automatic way for debugging.

Debugging is generally a serious undertaking for the designer along with large in addition to intricate models since these include commonly:[5]

- Heterogeneous: constructed from assorted parts maybe intellectual property (IP)[4] blocks by several major player in industry.

- Mixed: consisting of portions described at unique abstraction quantities behavior and also structural along with.
- Diverse: consists of several working out domains that will product real life connection, because sensors, transducers, digital-to-analog and/or vice-verse converters.

This incitement as well as response info utilized to work out as well as view layout behavior is additionally a big as well as various info set. Manipulating, mastering, as well as considering this specific info and it is connection along with anticipated or even desired behavior, and the designs setup (i.e., actual) behavior is usually a horrific starting. The method regarding debugging requires picking out the logic that is linked to an error, identifying the particular relevant lead to as well as result effective relationships, as well as being familiar with the way in which the planning is supposed in order to behave as well as the reason it is not conducting like that.[\[6\]](#)

3.2.1 Debug Paradigms

1. Signal level Debug[\[7\]](#)

- In signal activity need to identify the proper signal handshaking for said memory read or write operation.
- Simulation at the register transfer level (RTL)[\[4\]](#) generates tremendous amounts of signal-level data that must be tracked and analyzed, which drastically increases the complexity of the verification and debug process of SoC.

The basic paradigm is to capture the signal generated while simulating. The user configures signal probe network selects the number of signals to be simultaneously recorded in a trace buffer to analyze which further delivers to selected group - both a tracer and a debug monitor which analyzes a group of signals. Debug monitors output lines are directly connected with tracer. The debug monitor is configured to implement triggers, which enable us to start or stop recording by the tracer. A tracer can be written such a way that it gets stop or sleep mode when it gets full, or it continuously record until it gets triggered and signaled as stop. There is flag called full for trace buffer, Which is checked by debug tracer simultaneously.

2. **Assertion level debug**[\[7\]](#)

Assertion-based debug is a very popular paradigm and very tempting approach, which used successfully in pre-silicon verification. To do manual analysis of seized waveforms needs extensive efforts to check misbehavior in between of signals, where as by using assertions, it do all things automatically. Apart from assertions if we go with traditional approach, than identification of internal misbehavior takes thousands of cycles, which makes backtracking even difficult and tedious to identify root cause. In opposite of it, when an assertion gets triggered, we we gets much closer to actual error causing action, which leads to less time and circuit space also, so that by using this technique there is significant amount of reduction in space utilization can be seen. Thus assertions are shows its usefulness because they improve the serviceability of internal errors. Since most of assertions checks connections in between signals and do not rely on expected values, as in checks provides with exact match values, where it happens as generally don't know expected values in debug environment.

Assertions implemented in a local DEMON are driven by the same functional clock of the clock domain where the inputs analyzed by the assertion reside. An assertion implemented in a global DEMON works at a clock frequency higher than or equal to the frequency of the fastest clock domain providing inputs to the DEMON. All the enabled assertions can work concurrently.[7]

Because assertions are dynamically created in reconfigurable logic, many different assertions may be configured in the same infrastructure logic at different times during operation. While hundreds of assertions can be used in simulation, the number of assertions that can be configured concurrently in the fabric is limited by the fabric capacity and by the resources required to implement each assertion. However, we can take advantage of the re-usability of the re-configurable logic by partitioning the set of desired assertions into groups, and downloading one group at a time. In a system processing a non-repetitive continuous stream of data (such as music or video), this process can be automated so that each group runs in turn for a predetermined time period. Such an assertion loop provides a powerful silicon debug mechanism.

3. Transaction level Debug

- Transactions enable cross-team collaboration between hardware and software.
- They form a system design level at which many concerns can be analyzed in a fruitful and efficient manner.
- Provides the abstraction over the signal level generated data.

Cpu computer software is normally debugged from either the origin signal, or this processor instruction levels. The last option is the least expensive levels that's even now important for that programmer. Cpu instructions be construed as an all natural abstraction levels relating to the computer software along with the processor electronics. Yet (software) threads also communicate jointly by means of this interconnect, transactions. Transactions includes the reaction to processor instructions (such since load in addition to store) that result in activity on the interconnect as well as other IP blocks such since memories. Transactions for that reason really are a healthy software concerning calculation in addition to conversation. The instruction abstraction is significant for integrated hardware/software debug, while

the technique of some sort of transaction will be basic for method levels debug (i. e. numerous IP cores). This can be confirmed with the by using transaction-level modeling (TLM)[8] for interconnects in addition to SOC's.

From software engineers view , transaction level is the lowest level at which the embedded processors can be programmed by issuing read and/or write instructions. Now those read and write instructions causes transactions on the on-chip where communication infrastructure is established and, through translation to transaction commands using the appropriate communication protocol. The communication architecture transports these commands to one or more targets, which implement the actual write and/or read operation. As such, there is a natural correspondence between read and write instructions in software, and transactions within the systems communication infrastructure.

A hard-wired target is designed to respond to read and write commands on its communication interface that is connected to the systems communication infrastructure. When a read or write command is delivered to the target, the hardware designer knows how this target should react to this command. For example, when the target in question is a memory core, and the command is a write command, then the appropriate reaction of the target to the delivery of this write command is to store the commands data at the commands address.

3.3 Debug Process through Transaction

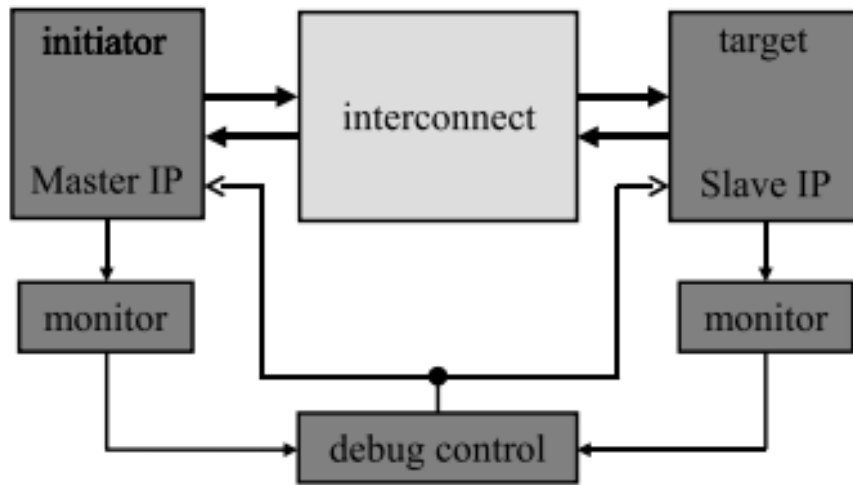


Figure 3.1: Simplified view of an SoC interface communication

- By doing debug at signal level, we need to keep track of all the signals which are generated at pin level and needs to find the connection in between them. Which is quite tedious task to do.
- While doing same thing at interface level we can capture the transaction which is initiated to perform the required operation and by getting response for the request we can say it is successful or not.
- Using higher level of abstraction by means of transactions, we can identified, whether the operation is successfully completed or not.
- Transaction logs, which are got are in row form at initial stage, and the amount of logs are also huge. Though it is lesser than, what in actual system generated. But helpful to analyze the system.
- We need to parse these logs and get required data from it. After refinement, we need to store it to database, for simplify and easy to visualizations.
- Transaction Level Debug is a perspective, which integrated into the main debug system and simplifies and automates log and trace file analysis.

- Traced data is shown in **Tabular** form , **Tree View** and **Log View** form inside the system. Which have several functionality like searching, filtering, coloring, shortcuts etc.
- We can open multiple views of different type to visualize the tracker log.
- We can also centralize the system as it can combined with different aids like verdi(Automated Debug System is an advanced open platform for debugging).

3.4 Experiment and Development of Drag N Drop

- Log files are large in size and contains several operations performed on data when request for it raise. These operations may not be logged in continuous form.Operations are Scattered in log file.

For example Any Read request is raised and request for it sent to CPU by peripheral. Before it gets completed other request also generated by other peripherals. Thus, logs for all this request will be generated as soon as they requested and stores all logs until gets completed. But due to several operations at a time it may not be show order of completion stages for an request.

- Thus, If User needs to check for specific instructions that if it is completed successfully or not than he/she needs to scroll and check operations performed on specific data, which is hectic when user needs to match some parameters for complete transaction of operation.
- For this, Need to experiment that whether a drag-gable view can created and user can drag needed rows from the log which is stored in either table or tree format in system.

3.4.1 Approach - For Drag N Drop

- Here, used Model/View Architecture as prior explained for Qt framework.
- **Previous approach consists of :-**
 - Two views
 - * Source View - Contains original data rows from table.
 - * Proxy view - Contains dragged rows from Source View.
 - Single Model
- Single model will consists of whole data, which will be applied under required rules and after getting the result, while dragging to proxy view, it will transfer only those rows which are selected from source view.
- In this approach, I used QSortFilterProxyModel, by using this model I show only those data, which are dragged to my proxy view.
- **Current implementation approach consists of :-**
 - Single view
 - Single Model
- Single model will consists of whole data, which will be applied under required rules and after getting the result, while dragging to scratch pad view, it will transfer only those rows which are selected from source view. But here I don't need to add any extra view as creating scratch pad view, only same view from where source view is getting displayed is used.

- **Case:1 - Table View**

- Before dragging the content to proxy view.

	1	2	3
1	174	175	176
2	161	162	163
3	148	149	150
4	135	136	137
5	122	123	124
6	109	110	111
7	96	97	98
8	83	84	85
9	70	71	72
10	57	58	59
11	44	45	46
12	31	32	33
13	18	19	20
14	161	162	163
15	149	150	151
16	137	138	139
17	125	126	127
18	113	114	115
19	101	102	103
20	89	90	91
21	77	78	79
22	65	66	67
23	53	54	55
24	41	42	43
25	29	30	31
26	17	18	19
27	148	149	150
28	137	138	139
29	126	127	128
30	115	116	117
31	104	105	106

Figure 3.2: Initially before dragging data rows from source view to scratchpad view

- After dragging the content to proxy view.

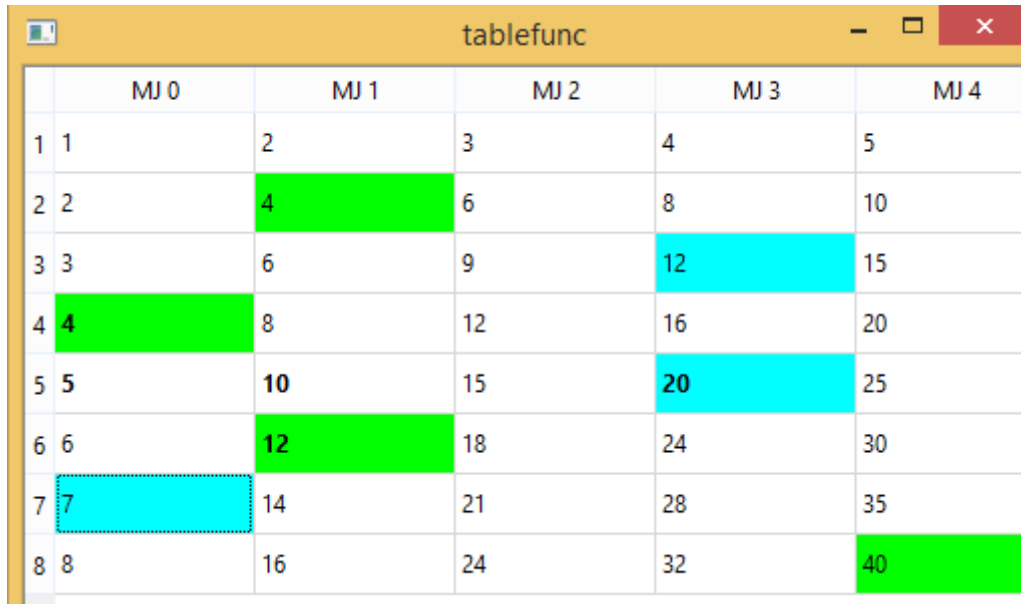
	1	2	3
1	174	175	176
3	148	149	150
6	109	110	111
16	137	138	139
19	101	102	103
21	77	78	79

	1	2	3
1	174	175	176
2	161	162	163
3	148	149	150
4	135	136	137
5	122	123	124
6	109	110	111
7	96	97	98
8	83	84	85
9	70	71	72
10	57	58	59
11	44	45	46
12	31	32	33
13	18	19	20
14	161	162	163
15	149	150	151
16	137	138	139
17	125	126	127
18	113	114	115
19	101	102	103
20	89	90	91
21	77	78	79
22	65	66	67
23	53	54	55
24	41	42	43
25	29	30	31
26	17	18	19
27	148	149	150
28	137	138	139
29	126	127	128
30	115	116	117
31	104	105	106

Figure 3.3: After dragging data rows from source view to scratchpad view

Here, features like coloring, bold also works in scratch pad view and you can go to the same data record by clicking in scratch pad view to actual view.

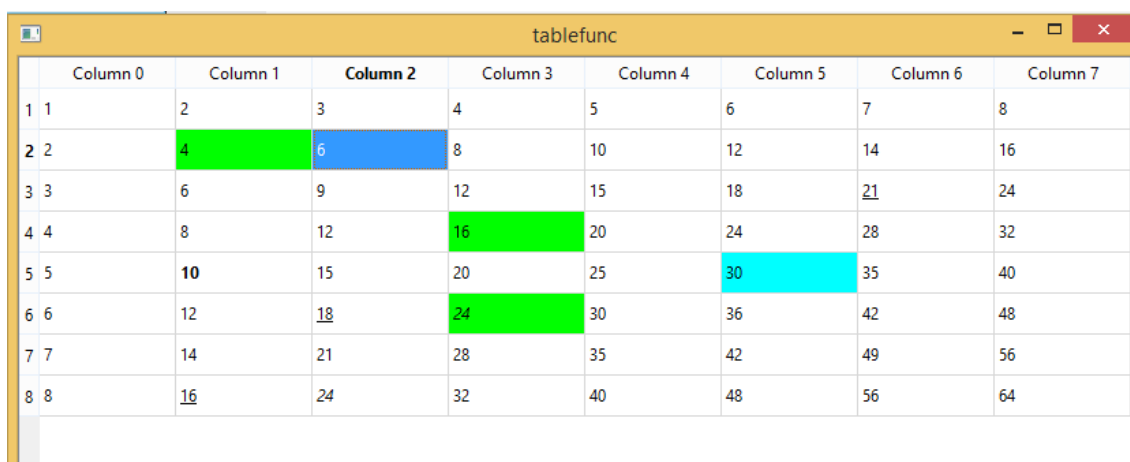
- Coloring and overlapping headers option as shown in figure 3.4 and figure 3.5.
- Coloring options with bold, italic text options.



	MJ 0	MJ 1	MJ 2	MJ 3	MJ 4
1	1	2	3	4	5
2	2	4	6	8	10
3	3	6	9	12	15
4	4	8	12	16	20
5	5	10	15	20	25
6	6	12	18	24	30
7	7	14	21	28	35
8	8	16	24	32	40

Figure 3.4: Example of color & text options

- Coloring options with bold, italic text options with demonstration of overlapping headers.



	Column 0	Column 1	Column 2	Column 3	Column 4	Column 5	Column 6	Column 7
1	1	2	3	4	5	6	7	8
2	2	4	6	8	10	12	14	16
3	3	6	9	12	15	18	21	24
4	4	8	12	16	20	24	28	32
5	5	10	15	20	25	30	35	40
6	6	12	18	24	30	36	42	48
7	7	14	21	28	35	42	49	56
8	8	16	24	32	40	48	56	64

Figure 3.5: Example of color & text options with overlapping headers

- **Case:2 - Log View**

- Before dragging the content to proxy view.

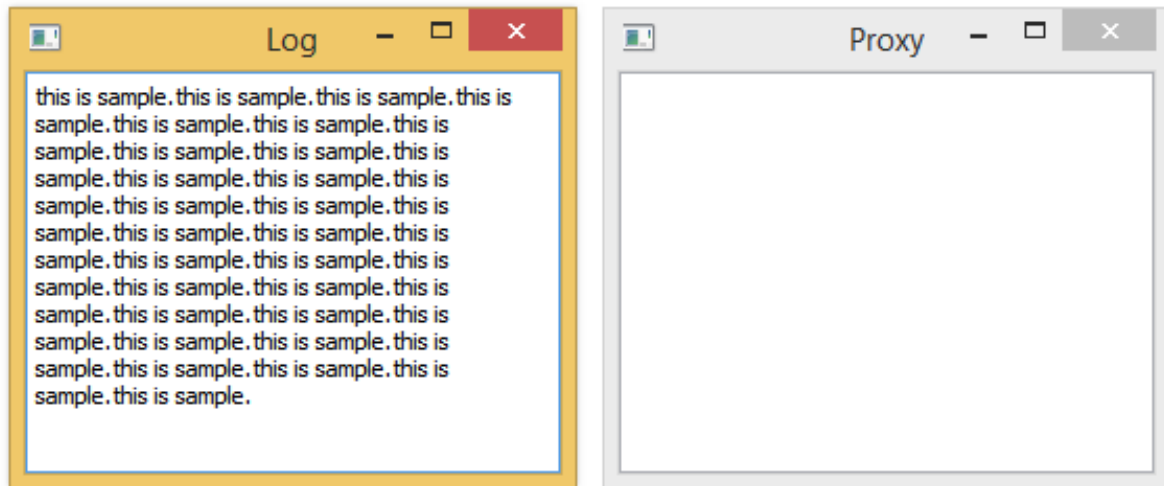


Figure 3.6: Initially before dragging data rows from log view to proxy view

- After dragging the content to proxy view.

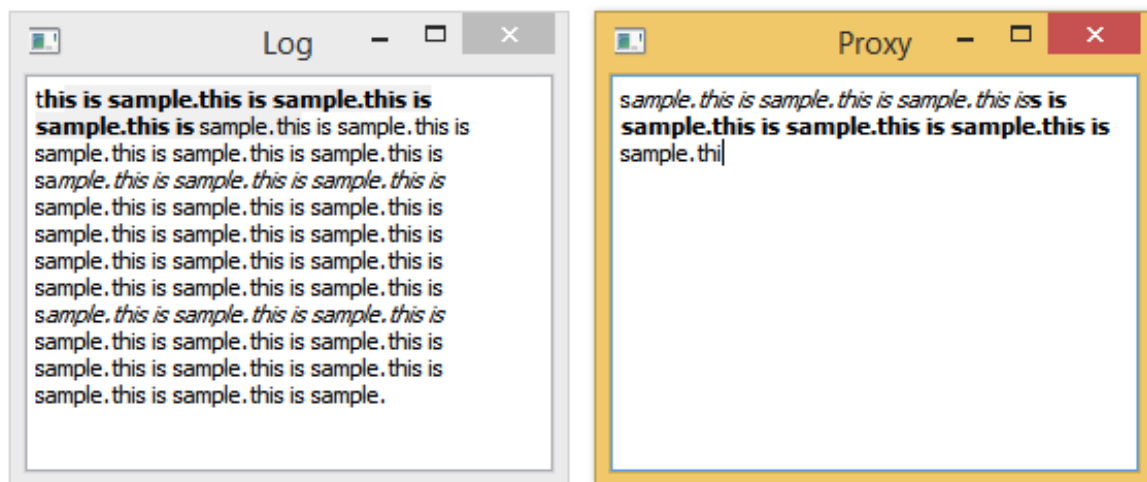


Figure 3.7: After dragging data rows from log view to proxy view

- Case:3 - Tree View
 - Before dragging the content to proxy view.

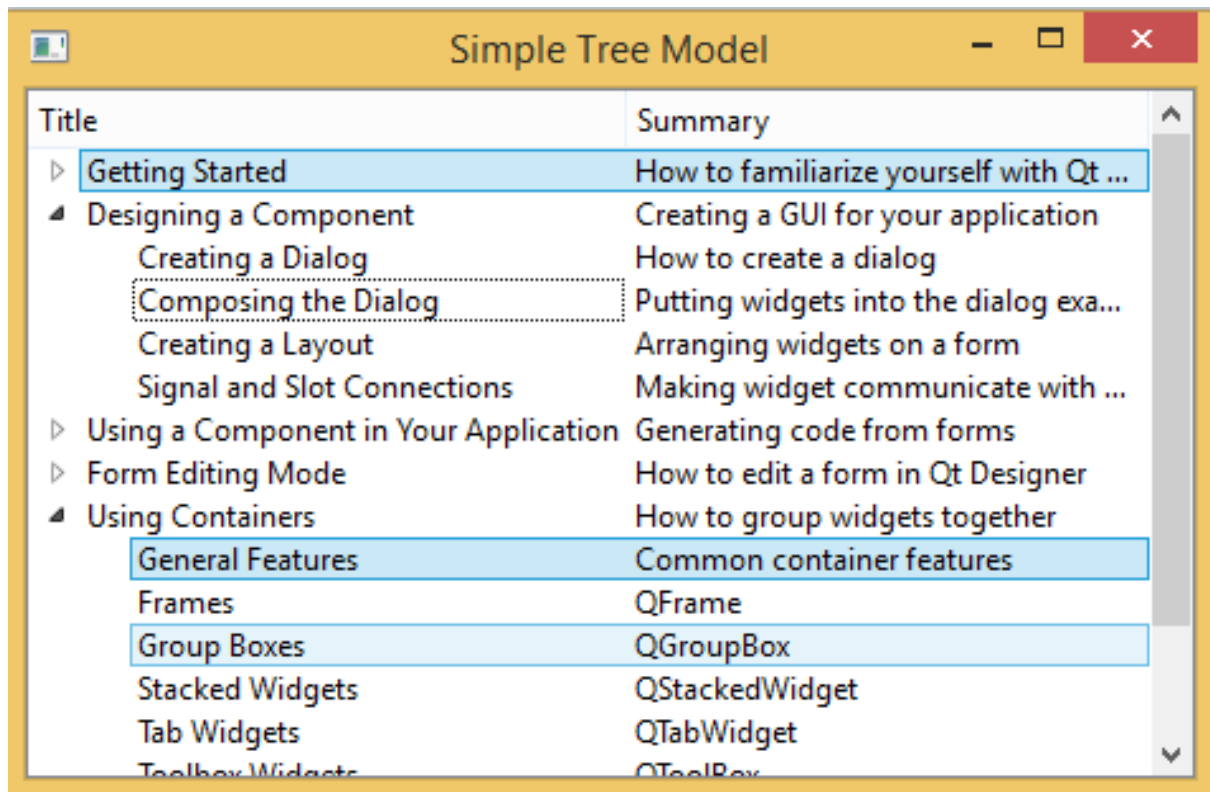
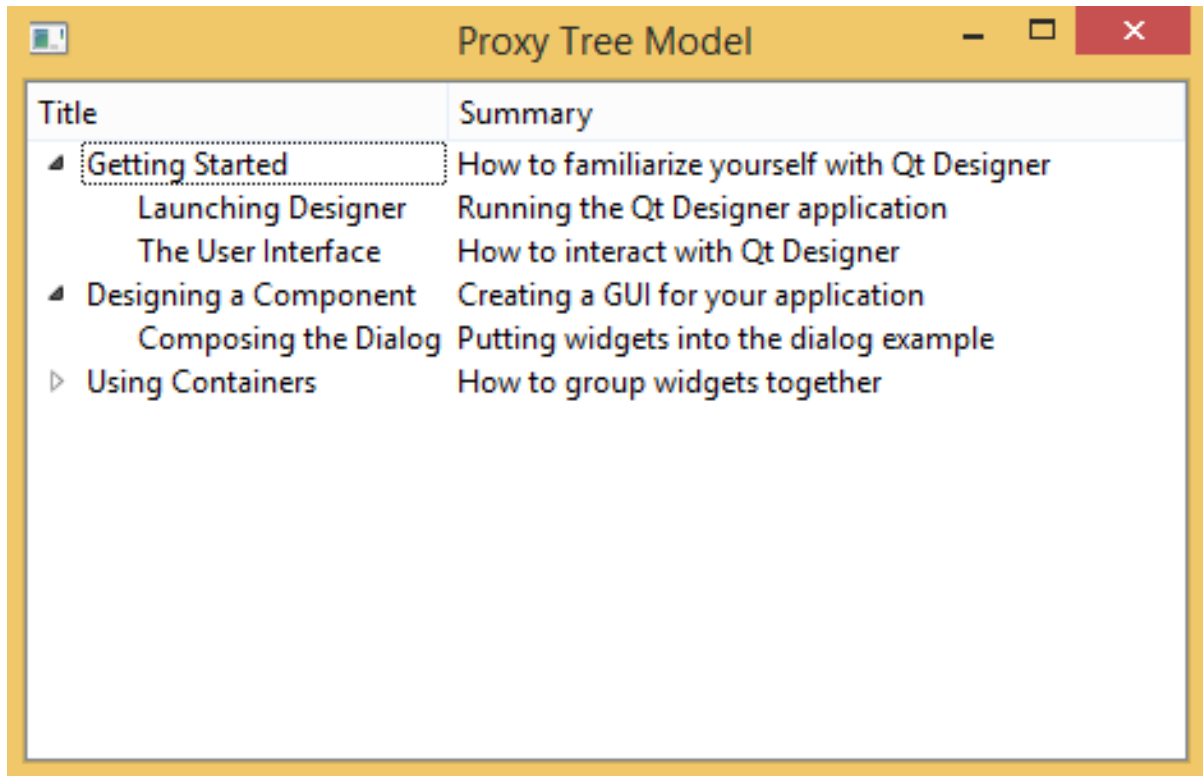


Figure 3.8: Initially before dragging data rows from tree view to proxy view

- As user can see there are some selected rows from Source tree model, wanted to be dragged to proxy tree model.
- Here, data is manipulated using `QAbstractItemModel`, which allows us to re-implement the tree model/view from scratch and give much-more handling power.

- After dragging the content to proxy view.



Title	Summary
▲ Getting Started	How to familiarize yourself with Qt Designer
Launching Designer	Running the Qt Designer application
The User Interface	How to interact with Qt Designer
▲ Designing a Component	Creating a GUI for your application
Composing the Dialog	Putting widgets into the dialog example
▷ Using Containers	How to group widgets together

Figure 3.9: After dragging data rows from tree view to proxy View

- In proxy tree model, data is not loaded in initial stage and have blank view.
- When user drag data to proxy model, it matches the rows and columns with actual data and display it on view.
- But using proxy model it takes too much time, if records are more in number.
- To remove this performance issues, I used internal pointer concept from tree modeling in Qt(cute) and regenerate tree only for dragged data on proxy model.
- Thus, user see it as dragging data and dropping data for tree view, where as in actual tree is restructured in proxy tree model.

3.4.2 Custom editor - Notepad view

- Notepad view for individual scratchpad view like Table/Tree/Log view(s).
- Here user can provide details about Table/Tree/Log view(s) content, which generally consist about different transactions.
- User can highlight very important part of text provided by him in notepad view.
- User have different options same as like a rich text editor have.
- For each view it creates separate view of notepad, so you can simply distinguish notes about respective use.
- Here content of notepad view, respectively with view(s) is sent via email.

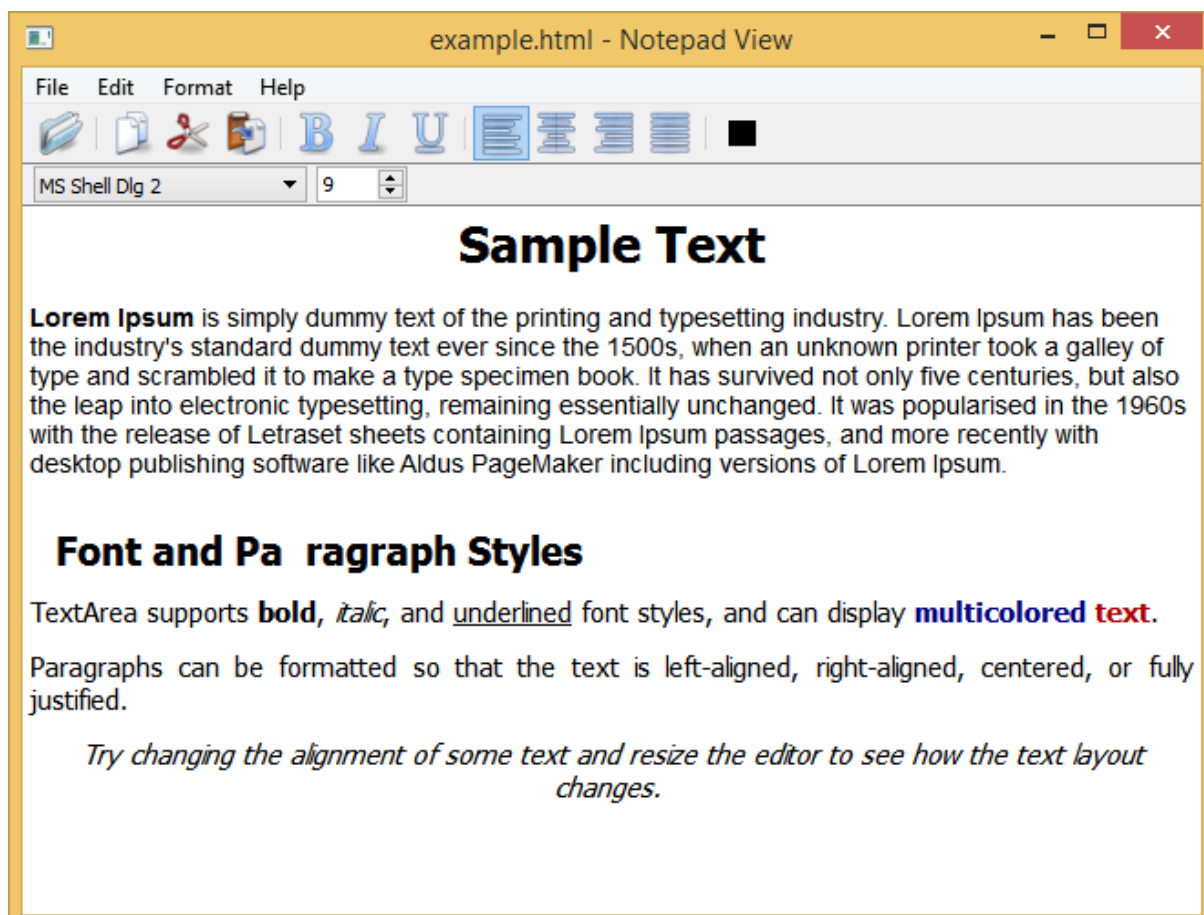
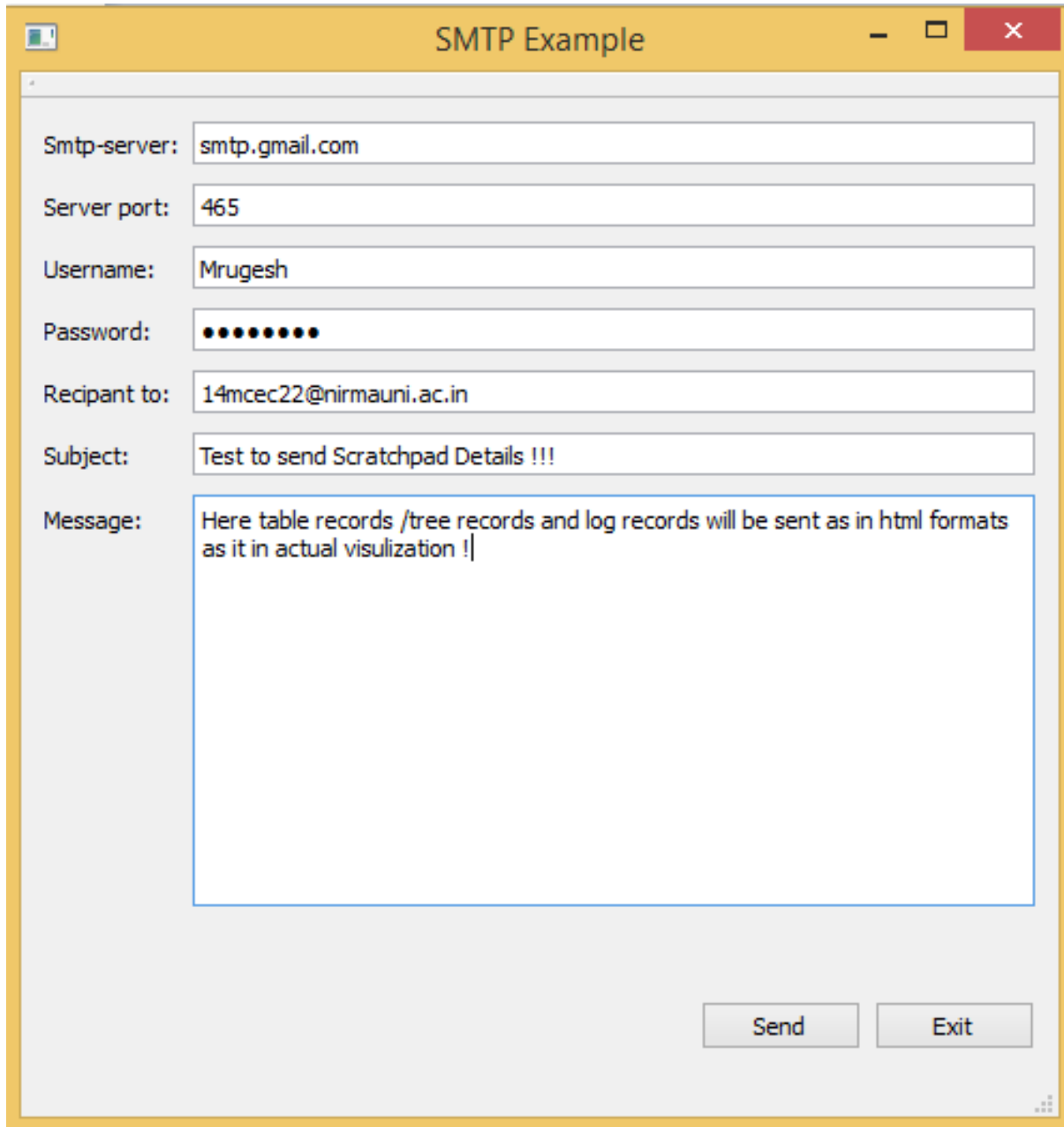


Figure 3.10: Extending simple text-area to rich text editor

3.4.3 Email capability to send scratchpad records

- Now we have email functionality to send records from scratchpad records
- Here, user can open as many views for table view, tree view, log view record.
- User can also have capabilities to do coloring, sorting, searching, define different roles also.
- Different type of roles are:-
 - Display Role
 - Decoration Role
 - Edit Role
 - Tool-tip Role
 - Status-tip Role
 - Background Role
 - Foreground Role
 - Font Role
- In email user can provide as many email id('s), as he wants by comma separated values.
- On successful email sent user will also notified by pop up box.
- If provided email Id('s) is/are improper than user get an notification to change it to correct one.

- Simple structure for email functionality is as show as below :-
- User need to configure SMTP server and needs to provided required details as shown in figure 3.6 .



The image shows a graphical user interface window titled "SMTP Example". It contains several input fields for configuring an email client:

- Smtplib-server:** A text box containing "smtp.gmail.com".
- Server port:** A text box containing "465".
- Username:** A text box containing "Mrugesh".
- Password:** A text box containing ten dots, indicating a masked password.
- Recipant to:** A text box containing "14mcec22@nirmauni.ac.in".
- Subject:** A text box containing "Test to send Scratchpad Details !!!".
- Message:** A large text area containing the text "Here table records /tree records and log records will be sent as in html formats as it in actual visulization !".

At the bottom right of the window, there are two buttons: "Send" and "Exit".

Figure 3.11: Email functionality for scratchpad records

3.5 Transaction Level Debug Details

- In this section will describe about TLD and show graphically about the solution.
- As shown in figure 3.12,when user open up TLD its looks like. In center space, where user can load different views like table/tree/log format style view which are shown in next figures.
- To load views views there are several options given in toolbar.
- To open required view format file, user needs to first go respective directory file from the provided options and select the correct file.Why?
- Because for different formats, we have different headers file and user needs to pick one correctly to load required view.

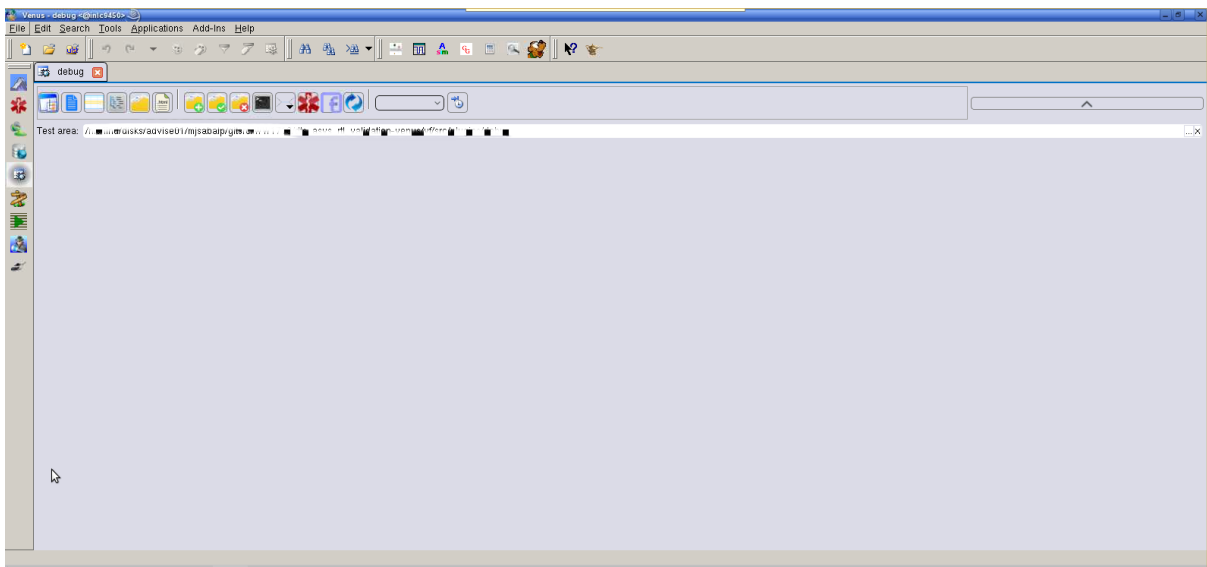


Figure 3.12: TLD main window

- Table/Tree view can show overlapping/non-overlapping headers, which changes with respect to status of the macro instructions.
- Log View is just raw text format with various functionality, thus don't have any headers.

- Below listed figures for views available and supported by TLD:-

– Pre-loaded table view - figure 3.13

Type	PId	Level	Time	uid	uIP	LIP	F	opCod	uop_name	opCod2	uop2_name	LDest	data	LSrc1	LSrc2	LSrc3	LSrc4	IPD	roID	pDst
909	-1	0	13486	N			0	0		0	0	0	0	0	0	0	0	0	15	33
910	-1	0	13866	N			0	0		0	0	0	0	0	0	0	0	0	28	2e
911	-1	0	14334	N			0	0		0	0	0	0	0	0	0	0	0	23	cb
912	-1	0	16028	N			0	4		0	0	0	0	0	0	0	0	0	ed	28
913	-1	0	20752	N			1	5		0	0	0	0	0	0	0	0	0	4d	80
914	-1	0	24234	N			0	1		0	0	0	0	0	0	0	0	0	195	a6
915	-1	0	25366	N			0	4		0	0	0	0	0	0	0	0	0	e0	b1
916	-1	0	12904	R			0	0		0	0	0	0	0	0	0	0	0	0	dd
917	0	1	3004	R			0	1		0	0	0	0	0	0	0	0	0	0	0
918	2	2	3004	R			0	1		0	0	0	0	0	0	0	0	0	0	0
919	2	2	3006	R			0	1		0	0	0	0	0	0	0	0	0	0	1
920	2	2	3032	R			0	5		0	0	0	0	0	0	0	0	0	0	2
921	2	2	3034	R			0	2		0	0	0	0	0	0	0	0	0	0	3
922	0	1	3078	R			0	5		0	0	0	0	0	0	0	0	0	0	4
923	8	2	3078	R			0	5		0	0	0	0	0	0	0	0	0	0	4
924	8	2	3080	R			0	2		0	0	0	0	0	0	0	0	0	0	5
925	0	1	3138	R			0	52L		0	0	0	0	0	0	0	0	0	0	6
926	12	2	3138	R			0	0		0	0	0	0	0	0	0	0	0	0	6
927	12	2	3140	R			0	0		0	0	0	0	0	0	0	0	0	0	7
928	0	1	3198	R			0	0		0	0	0	0	0	0	0	0	0	0	8
929	16	2	3198	R			0	0		0	0	0	0	0	0	0	0	0	0	8

Figure 3.13: TLD table view format file display with options

– Pre-loaded Tree View - figure 3.14

Type	PId	Time	ST	uid	uIP	LIP	F	opCod	uop_name	opCod2	uop2_name	slot	LDest	retData	LSrc1	LSrc2	LSrc3	LSrc4
-1	12904	R			U14		0	0		0	0	0	0	0	0	0	0	0
-1	13018	R					0	50		0	0	0	0	0	0	0	0	0
-1	13112	R					0	14		0	0	0	0	0	0	0	0	0
1539	13018	R	4m				0	0		0	0	0	0	2400	0	0	0	0
1539	13024	R	4m				0	0		0	0	0	0	0	0	0	0	0
1545	13024	R	4m				0	0		0	0	0	0	0	0	0	0	0
1545	13026	R	4m				0	0		0	0	0	0	80000006	0	0	0	0
1545	13030	R	4m				0	0		0	0	0	0	0	0	0	0	0
1545	13036	R	4m				0	0		0	0	0	1	0	0	0	0	0
1545	13040	R	4m				0	0		0	0	0	0	0	0	0	0	0
1545	13040	R	4m				0	0		0	0	0	1	0	0	0	0	0
1539	13040	R	4m				0	0		0	0	0	0	0	0	0	0	0
1539	13052	R	4m				0	0		0	0	0	0	0	0	0	0	0
1539	13052	R	4m				0	0		0	0	0	1	0	0	0	0	0
1539	13066	R	4m				0	0		0	0	0	0	0	0	0	0	0
-1	13208	R	4m				0	5m		0	0	0	0	0	a	0	0	0
-1	13210	R	4m				0	0		0	0	0	0	0	0	0	0	0
-1	13230	R	4m				0	0		0	0	0	0	0	0	0	0	0

Figure 3.14: TLD tree view format file display with options

– Pre-loaded Log View - figure 3.15

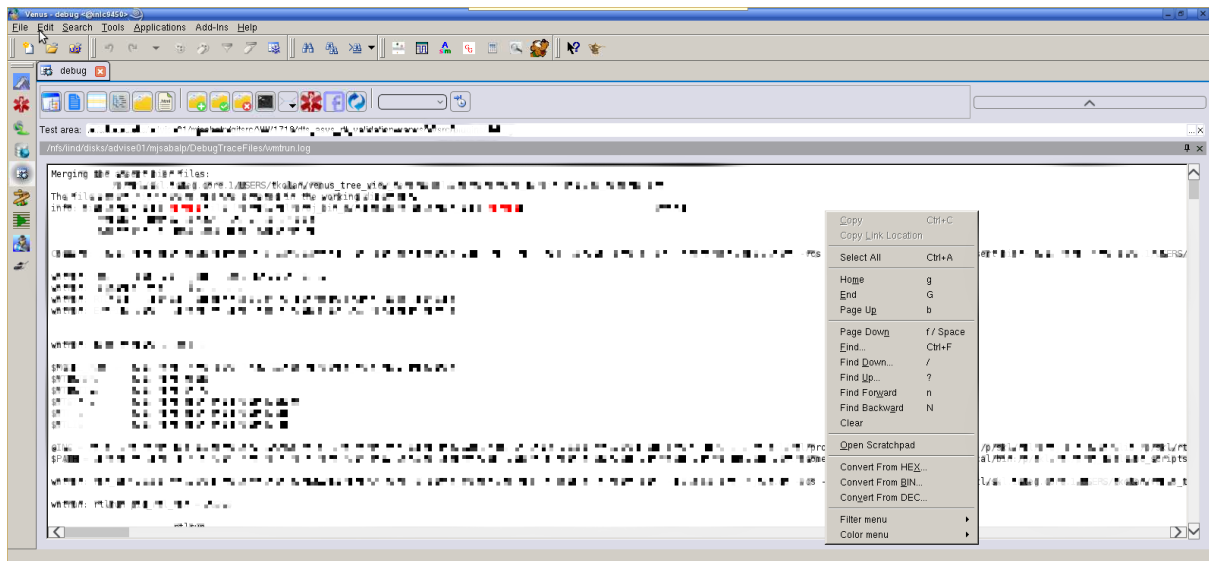


Figure 3.15: TLD raw log format file display with options

- For Each view mentioned above have context menu, which get opens on right click options.
- Here, user have option for scratchpad option called "**Open Scratchpad**". On clicking this menu there will open another window opens called "**scratchpad**".

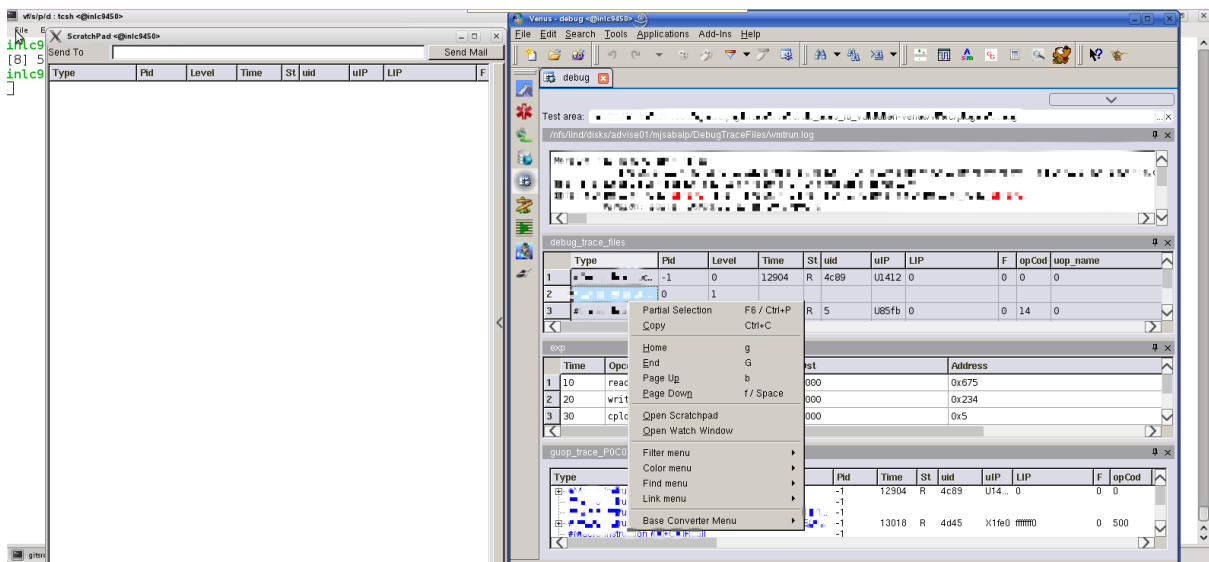


Figure 3.16: Main window for scratchpad

- Initially view doesn't consist any records but have headers form the respective view, means related to particular view from where it is get opened as shown in figure 3.16.

- As explained in various experiments about drag and drop functionality for views, user can drag small chunk of data to scratchpad view.
- Here dragging the data is allows not only in simple texture but with all styles applied to records.
- As user can see that some of the data is dragged which are not in simple text and have applied background/foreground colors by given functionality, as shown in figure 3.17.

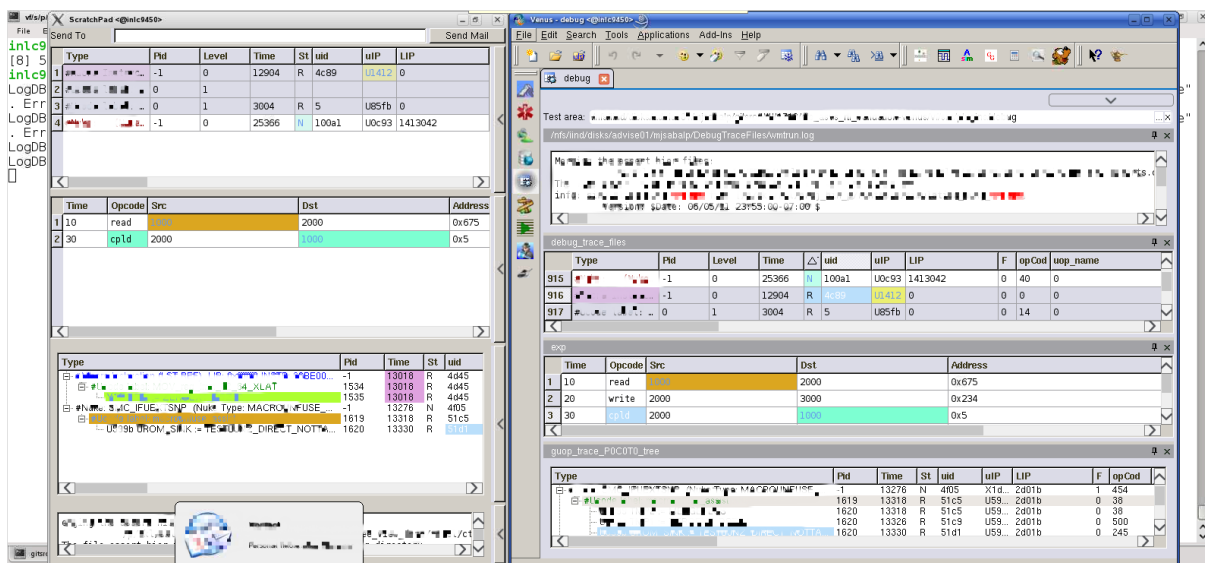


Figure 3.17: Scratchpad consisting different views with records

- In figure 3.17 user finds more option in scratchpad view to provide some additional information for respective dragged view.
- This "Notepad" option is available on the right side of each scratchpad view for each layout view (means for each table/tree/log view, opened inside scratchpad).

- When user clicks on arrow provided on right side, it gets opened as shown in figure 3.18. Here, user can add extra details required to related layout view, regarding errors, bugs, which needed to be report to debugger/engineers.
- In experiment part, where I implemented one sample editor, where functionality like:
 - Bold text
 - Italic text
 - Underline text
 - Increase/Decrease font size text
 - Color selected text
 - Clear all style of text
 - Copy text
 - Paste text
 - Select All

are captured and used in scratchpad view's notepad part as user can see in figure 3.18

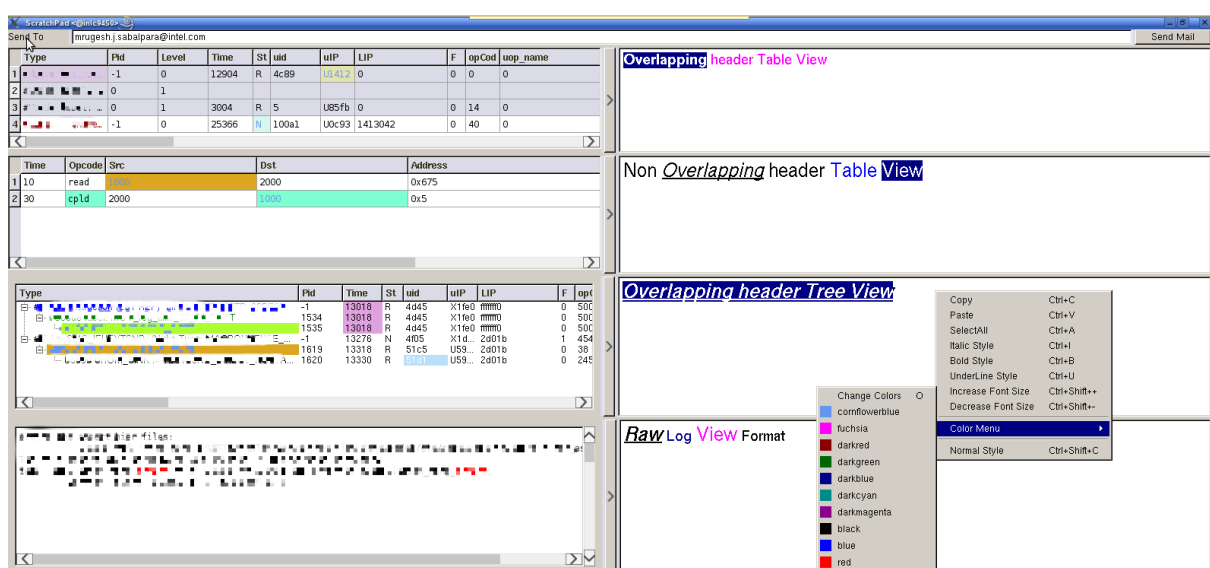


Figure 3.18: Scratchpad notepad options

- As shown and in experiment part about email functionality, here used the same concept in very generic way, where user just need to provide email id(s) to whom this email needs to be sent, which consist of information for new change requirements/enhancement/bugs.
- When user clicks on ”**Send Mail**”, email sent directly and pop up comes up which verifies the validity of email id(s) and show accordingly message on **Send Mail Successful** or **Please, Provide Correct Details**

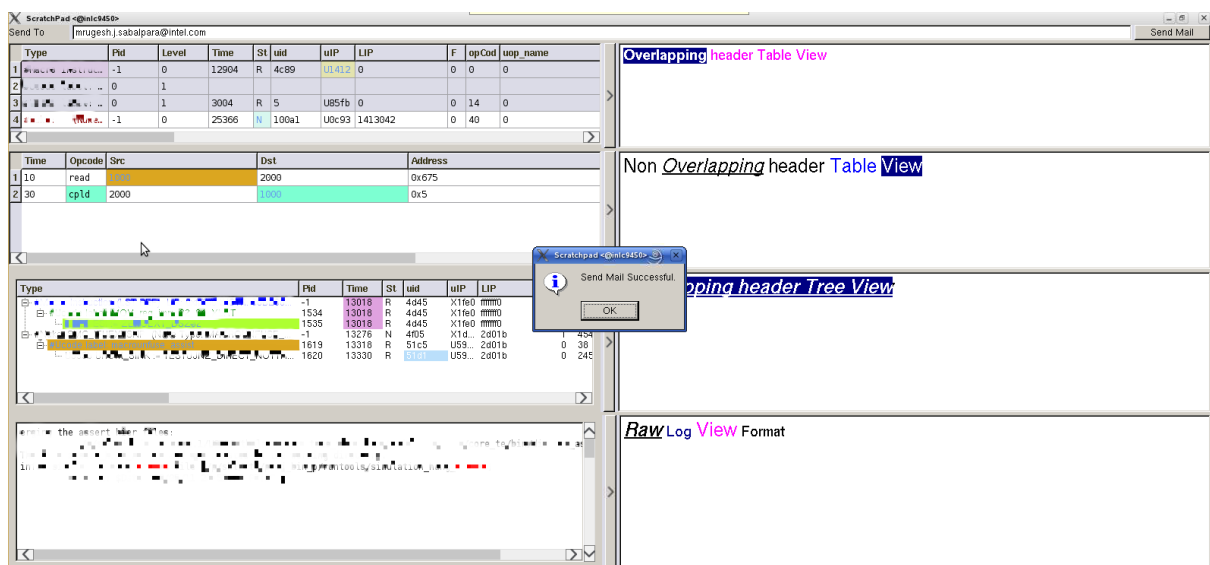


Figure 3.19: Email functionality for scratchpad records to send debugger/engineer

- Here user can also make edit of the received data records and forward it further.



Figure 3.20: Email records in editable format to debugger/engineer to email box

- To send records from table/tree/log views, I had written recursive functions, from where it gets data from each view and iterate up-to its last occurring child record, which is complex part in just showing records with all options and in proper format in email.

Chapter 4

Zoom Network Diagram Viewer

4.1 General

- Zoom Network Diagram Viewer Definition:-
 - Multi-level vector-based block diagrams of the micro architecture, implemented as an integral part of Debug perspective.
- Zoom Network Diagram Viewer perspective is an easy-to-use Transaction Level Debug-Based tool that allows:-
 - navigation
 - visualization
- Interactive Map of the design Block description and excerpts at the click of the mouse.

4.2 Detail Of Zoom Level Network Diagram Viewer

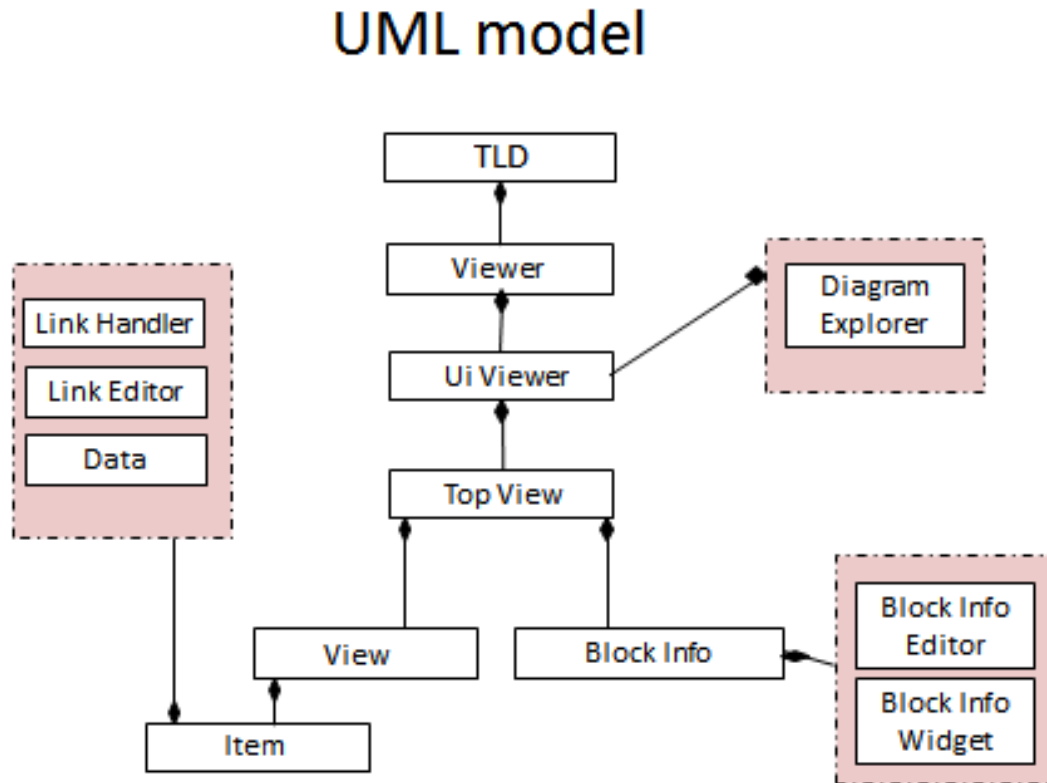


Figure 4.1: UML Model for Zoom Network Diagram Viewer

- Zoom Network Diagram Viewer provides Interactive Map of the design, as customer can see block wise structure.
- Every Block has its own description which is displayed when you click the block, it shows details.
- It can be Integrated with other familiar debug tools like Transaction Level Debug. Which Shown in figure 4.1 as at top most level it is a part of TLD.

- It helps to understand the complexity of high level architecture in easier way through visualization, thus it reduces the time and improve debug efficiency in manner.
- All the blocks in ZNDV is connected by buses as they will in physical entity, so provides better clarity about traffic flow.

4.3 Zoom Network Diagram Viewer Details

- Today ZNDV is standalone application, where as its also available with TLD perspective.
- Here, ZNDV which is a part of TLD is based on UNIX based environment, and standalone ZNDV application is windows based, which comes as a different package.
- As earlier mentioned ZNDV is used as a visualization solution for system on chip(SoC).
- Diagrams can have more than one level in-depth. Parent SoC contains blocks, and with those block another Soc can be connected. It is shown in tree hierarchy structure, so when any block is clicked and it has another svg file connected with , than it will redirected to child svg.
- Its shows the flow for SoC, that what happens on providing inputs or rather saying is showing the whole architecture for provided SoC.

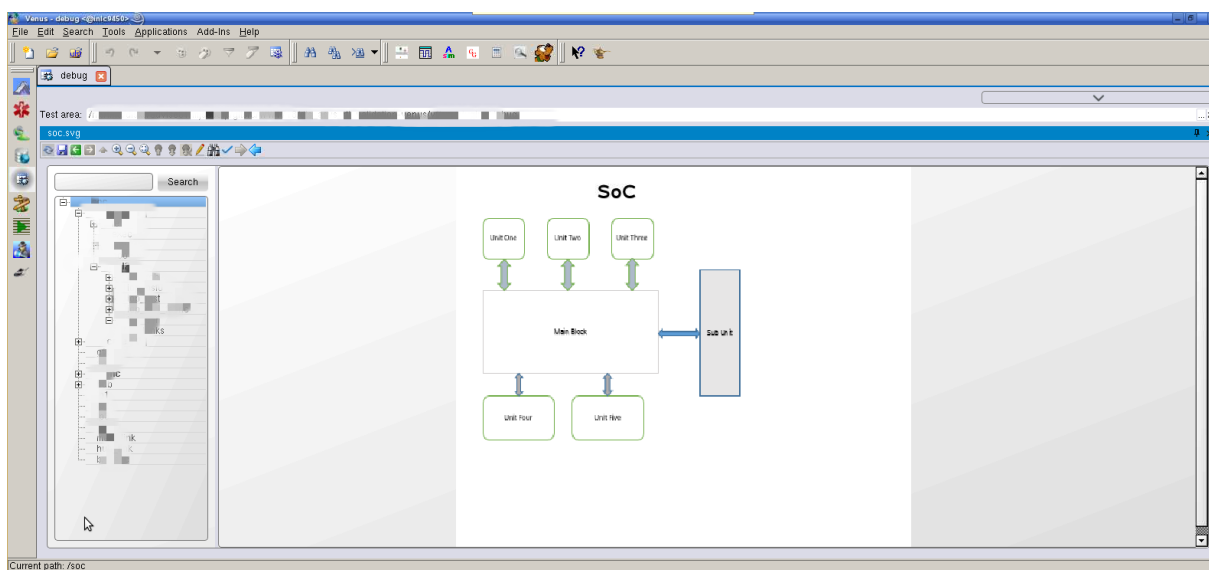
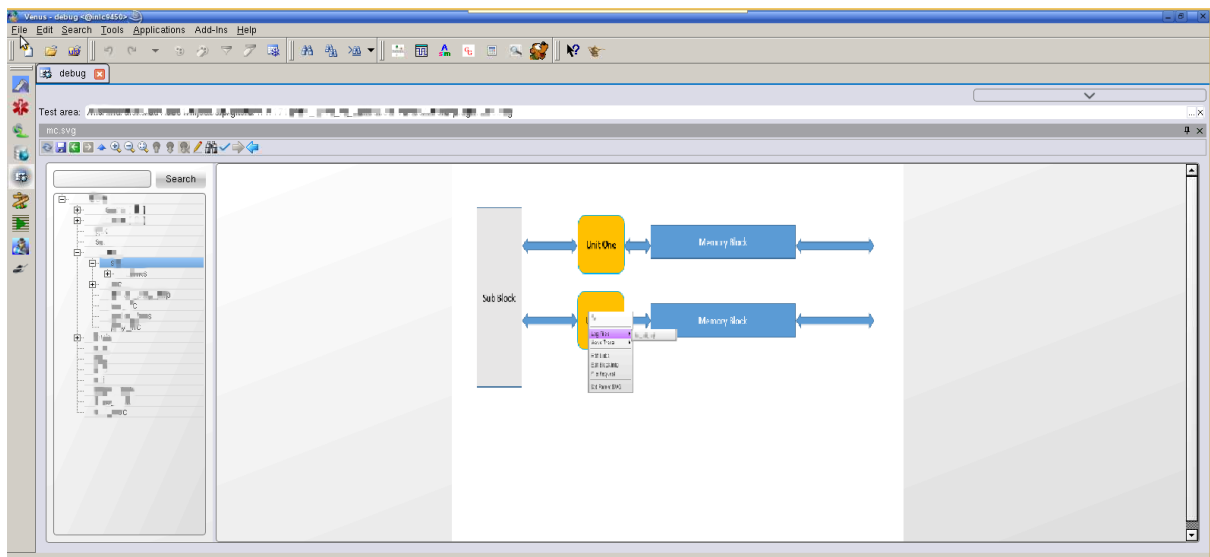


Figure 4.2: Zoom Network Diagram Viewer Visualization

-
- The screenshot displays the Venus IDE interface. At the top, there's a menu bar with options like File, Edit, Search, Tools, Applications, Add-Ins, and Help. Below the menu is a toolbar with various icons. The main workspace is divided into several panes. On the left, there's a 'Test area' pane showing a tree view of test cases. The central pane displays a diagram titled 'IOP.svg'. This diagram illustrates a system architecture with the following components and connections:
- Unit One** (yellow rounded rectangle) is connected to **Block One** (green rounded rectangle) by a bidirectional arrow.
 - Unit Three** (yellow rounded rectangle) is connected to **Block Two** (white rounded rectangle) by a bidirectional arrow.
 - Unit Two** (yellow rounded rectangle) is connected to **Sub Svg Block** (gray rounded rectangle) by a bidirectional arrow.
 - Block One** is connected to **Block Two** by a bidirectional arrow.
 - Block One** is connected to a **Sub Svg Block** (blue rounded rectangle) by a bidirectional arrow.
 - Block Two** is connected to another **Sub Svg Block** (blue rounded rectangle) by a bidirectional arrow.
 - Block Two** is connected to the **Sub Svg Block** (gray rounded rectangle) by a bidirectional arrow.
- The status bar at the bottom indicates the current path: `/soc/core [1] v02`.

- Each block has many options to add file like:-



35

- Log file
 - Documentation
 - Links
 - Block Info
 - Register Transaction Language files etc.
- Usefulness and contribution for understanding:-
 - Interactively **explore** multi-level block diagrams of the design.
 - Enable everyone to **debug** like an expert using contributed content.
 - Experts getting fewer interrupts/requests for debug help
 - Non-experts are less reliant on experts to make progress
 - Providing motivation to experts to develop debug solutions/collateral

4.4 Development Task - Zoom Network Diagram Viewer

- In system previously, it was not taking command line arguments. For this I need to write command line parser and allow user to pass command line arguments to open the svg files in Zoom Network Diagram Viewer perspective.
- Previously settings file is created on root directory, which needs to move to users temp directory.
- Implementation of utility, which allows user to know priory about not functional areas of diagram and easy to use on single click action by his/her.
- User can easily understand and correct it.
- Adding feature for searching in tree structure, thus diagrams will be shown as hierarchy and user can search related blocks.
- Previously there were ambiguities related to links as user sometimes confuses in between link(s) connect to which block(s).
- Enhanced the solution by allowing user to jump from any link to its respective blocks and also, visualize the all elements per blocks, so user easily understand the relation between blocks and document.

4.4.1 Exploration on how external Qt application to be part of eclipse

Explored Ways:-

1. **The Eclipse framework** is written in Java and the its plugin infrastructure is also built around java only, which is basically implies that if you want to write plugin than you need to use Java. There is also possibility to have plugin written in C++(or other languages native languages) and then,it can be bridged to Java by using C++ interface and avail as plugin by use of JNI (Java Native Interface).

Steps to follow:- To call Library written in C++ from Java, I Need to follow below steps:

- (a) Identification of C++ classes and methods that I want to make available from those in the library's API, from where I can make meaning full functionality to be worked
- (b) Next I need to write appropriate or more precisely equivalent Java classes, in which methods that will be implemented by the library are declared using the native keyword and left unimplemented in the Java code.
- (c) Generate stub function declarations for the JNI wrapper. This JNI wrapper contains special names given to the functions, which are implemented in both side, and these names given by generator.
- (d) Write the body of the JNI wrapper, implementing each function by converting its Java type arguments into appropriate C or C++ types, calling the native library, and converting the results back again;
- (e) Now, we are almost ready, just need to write small Java program to test it.
- (f) Build everything means Java, C++ code and get it run.

Conclusion:- Here, I need to write Java classes for equivalent C++ classes, which method declaration. So for ZNDV, we need to write equivalent Java classes and generate JNI interfaces for it.

2. **Qt Jambi:** - Qt Jambi is a Java binding of the cross-platform application framework Qt. It allows Java developers to use Qt applications in Java programming domain. Also, Qt Jambi generator can be used to create Java bindings for other Qt libraries and future versions of Qt.

Qt Jambi generator:- It is a Qt application which used to map C++ APIs into equivalent Java APIs, enables C++ programmers to easily integrate their c++ based Qt application into Java environment. The generator supports a selected subset of C++, which covers common constructs. Which generates equivalent Java API's for those common constructs, so we don't need to write extra code to convert those C++ classes . It also generates code which bridges the Java classes to the C++ classes. Based on the Java Native Interface (JNI), it ensures that method

called from Java are redirected to the corresponding functions in the C++ library.

Conclusion:-

- (a) **Installation details:** (Currently support get Ended. Found installation details but many links are broken) https://doc.qt.io/archives/qtjambi-4.5.2_01/com/trolltech/qt/qtjambi-installation.html Didn't get how to integrate Qt Jambi into eclipse.
- (b) **Qt Jambi Generator:** <http://qtjambi.org/doc/generator> Here, automatically converts C++ function calls to Java Classes and function declaration using JNI. For Qt Jambi Generator, need to write specification xmls.
- (c) **Qt Jambi Generator Example:** https://doc.qt.io/archives/qtjambi-4.5.2_01/com/trolltech/qt/qtjambi-generatorexample.html

3. **JNI Approach:** - More often than not, it's important to use indigenous rules (C/C++) to defeat your recollection management /memory in addition to functionality constraints in Java. Because, java helps indigenous rules calling performance making use of java native interface (JNI). JNI is difficult, the way it involves two dialects in addition to runtimes..[\[3\]](#)

Required knowledge of:

- (a) Java
- (b) C/C++ - the GCC Compiler
- (c) Gygwin or MinGW based on Linux or Windows
- (d) Eclipse C/C++ Development Tool (CDT)

Example:-

- **Step 1: Write a Java Class which using native language - HelloWorldJNICpp.java**

```
public class HelloWorldJNICpp {  
    static {  
        System.loadLibrary("hello"); //dll or so  
    }  
    // Native method declaration  
    private native void sayHelloWorld();  
  
    // Test Driver  
    public static void main(String[] args) {  
        new HelloWorldJNICpp().sayHelloWorld(); // Invocation  
    }  
}
```

– Compile the HelloWorldJNICpp.java into HelloWorldJNICpp.class.

* javac HelloWorldJNICpp.java

- **Step 2: Auto Create C/C++ Header file - HelloWorldJNICpp.h**

```
JNIEXPORT void JNICALL Java_HelloWorldJNICpp_sayHelloWorld(JNIEnv *, jobject)
```

– To get these generated files

* javac HelloWorldJNICpp.java

- **Step 3: Write C/C++ functionality - HelloWorldJNICppImpl.h, HelloWorldJNICppImpl.cpp, and HelloWorldJNICpp.c**

- C++ Header - "HelloWorldJNICppImpl.h"

```
#ifndef _HELLOWORLD_JNI_CPP_IMPL_H
#define _HELLOWORLD_JNI_CPP_IMPL_H

#ifdef __cplusplus
    extern "C++" {
#endif

        void sayHelloWorld ();

#ifdef __cplusplus
    }
#endif

#endif
```

- C++ functionality - "HelloWorldJNICppImpl.cpp"

```
#include "HelloWorldJNICppImpl.h"
#include <iostream>

using namespace std;

void sayHelloWorld () {
    cout << "Hello World from C++!" << endl;
    return;
}
```

- Write interface in C++ Program connecting to Java - "HelloWorldJNICpp.c"

```
#include <jni.h>

#include "HelloWorldJNICpp.h"

#include "HelloWorldJNICppImpl.h"

JNIEXPORT void JNICALL Java_HelloWorldJNICpp_sayHelloWorld
(JNIEnv *env, jobject thisObj) {
    sayHelloWorld(); // invoke C++ function
    return;
}
```

- To get this resultant file

```
* set JAVA_HOME=C:/ Program Files/ Java / jdk1.7.0_XX
* g++ -Wl,-add-stdcall-alias -I"% JAVA_HOME %/ include" -I"%JAVA_HOME%/
include win32" -shared -o hello.dll HelloWorldJNICpp.c HelloWorld-
JNICppImpl.cpp
```

- **Step 4: Execute Java class**

- java -Djava.library.path=. HelloWorldJNICpp

Conclusion:-

- JNI approach is to provide interface between java and Qt, which is C++ based framework. But to do so, we need to write JNI interfaces for all required Qt class functions, to be called from java.
- Writing JNI interface for each function of Qt class is very much time consuming. Or in other words, It is like writing every thing again in java.

4.4.2 Conclusion

Sr. No.	Approach	Description	Maintenance	Limitations
1	Write JNI interface	To communicate need to write JNI in between Qt(C++) and JAVA	Moderate	Need to write JNI interface manually for all class required to call from JAVA.
2	Qt Jambi Generator	IT takes users specification XML and internally builds Java classes for respective Qt(C++) classes	Moderate	No proper documentation available.
3	QX11Embed Container	Container binds other widget inside it.	Moderate	Need to check with sample program. Platform compatibility.
4	Launch zMAP as other window	Provide button to launch zMAP using java plugin, so opens in another window.	Less	Cant interact with different java environment

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Initial phase of internship was bit of getting ramp ups on technologies, which I didn't know priory. I got great understanding about Qt (Cute) and C++.Also I learn scripting technologies like perl, python during workshop and in working. To understand the debug solutions and working, I had many hands on exercises in initial time, and it help me a lot to understand working environment. I got good knowledge on TLD, ZNDV and back-end specifically mysql and overview of elastic, in my internship.

Leadership/Ownership:

Mentors and team member trusted me to develop full stack creation of solution as part of different debug solutions. And for it I had many experiments carried out to come up to solutions, which is currently part of TLD and ZNDV respectively. I had owned scratchpad in TLD and ZNDV as an engineer in full term internship. Also got many opportunities to talk with customers and get positive feedback's from them.

Scope/Impact/Relevance to Intel:

- I had to deliver scratchpad which supports drag and drop functionality for table/tree/log view formats, with capabilities like coloring searching, filtering, emailing. Email functionality used to send this scratchpad details to respective debugger/engineer/tool user, to ease understanding and provides information related to any kind of flaws, if any occurred in debug. By this, it is saving much time of debugger/engineer/tool user to debug tools and cost effectiveness in pre-si verifica-

tion. It also provides ease to understand small chunk using scratchpad sent as email. And also worked on UI enhancement for transaction level debug.

- While in ZNDV, there are many request for enhancement and exploration, which I carried out. Exploration about making ZNDV part of eclipse and work as a core part as written in JAVA. For it I got one month time and I provided my analysis on it and presented to team, which unfortunately turn into freeze state as there is no simpler way to do that, even I can write JNI interface, I need to write it for hundreds of classes and thousands of functions, which is very time and cost effective!
- I had request about showing links and connections for per block wise, so to remove the ambiguity in ZNDV diagrams and user can have easy understanding and also providing utilities for identify non-functional area of SoC, so user can make it correct, even if user don't have much prior information/experience.
- This all impacts majorly as one of the clients said, "**Scratchpad will be the game changer**" and ZNDV is very important tool to get knowledge of SoC level via visualization.

Lessons Learned:

I learned about day-to-day life in corpora. How actually work carry out, chances got to meet higher level people in organizations. Faced challenges, initially when I had very shallow knowledge of domain, but that's a part of learning and rapidly I compete with it.

- And for my excellent work I got :

"Recognizing Mrugesh for his commendable effort in delivering ZNDV request. Mrugesh quickly ramp up on ZNDV Code and implemented the feature with good quality. He took ZNDV request apart from his regular deliverable on scratch pad and delivered in short span of time, I would highly appreciate his multitasking abilities for making the things successful. Thank You."

Results Orientation:

- Award honors the impact I have demonstrated through:
- Assuming responsibility.
- Constructively confronting and solving problems.
- Executing flawlessly.

5.2 Future Work

- New exploration path for reducing loading time for views, using web technologies like Node js/Angular js and elastic as a back-end.
- Making visualization feature available from core - to uncore connecting with IP's, which covers everything comes under debug.
- Currently visualization part of debug solutions are very much stable and now shifting towards improvement in debug efficiency, like exploring AMBA/USB protocols, which will be next step.
- Shifting to load sharing and adopting hadoop technologies in future, so data loading and sharing will get improved and make it more efficient to use.

Bibliography

- [1] “<http://doc.qt.io/qt-5/reference-overview.html>,”
- [2] “<http://doc.qt.io/qt-4.8/model-view-programming.html>,”
- [3] “<https://www3.ntu.edu.sg/home/ehchua/programming/java/javanativeinterface.html>,”
- [4] B. Vermeulen, K. Goossensy, R. van Steedenz, and M. Bennebroekx, “Communication-centric soc debug using transactions,”
- [5] M. Abramovici, P. Bradley, K. Dwarakanath, P. Levin, G. Memmi, and D. Miller, “A reconfigurable design-for-debug infrastructure for socs,”
- [6] “<http://www.design-reuse.com/articles/12790/transaction-based-debug-of-pci-express-embedded-soc-platforms.html>,”
- [7] K. Goossens, B. Vermeulen, R. van Steeden, and M. Bennebroek, “Transaction-based communication-centric debug,”
- [8] “<http://www.design-reuse.com/articles/33664/using-transactions-to-effectively-debug-large-soc-designs.html>,”