Design & Development of automated validation framework for Intel Graphics display driver

Prepared By :

Nirav Patel

12MCEC21

Internal Guide

Dr. Madhuri Bhavsar

Nirma University

External Guide

Mr. Praveen Vasudevan

Intel Technology India Ltd.

Department Of Computer Science And Engineering

Institute Of Technology

Nirma University

Ahmedabad

May-2014

Design & Development of automated validation framework for INTEL Graphics display driver

Major Project

Submitted in partial fulfillment of the requirements

For the degree of

Master of Technology in Computer Science and Engineering

PREPARED BY :

Nirav Patel

12MCEC21

Internal Guide

Dr. Madhuri Bhavsar

Nirma University

External Guide

Mr. Praveen Vasudevan

Intel Technology India Ltd.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INSTITUTE OF TECHNOLOGY

NIRMA UNIVERSITY

AHMEDABAD

DECLARATION

This is to certify that,

I, **Nirav Patel**, **12MCEC21**, a student of semester IV M.Tech CSE, Nirma University, Ahmedabad , hereby declare that the project work **Design & Development of automated validation framework for INTEL Graphics display driver** has been carried out by me under the guidance of Mr. Praveen Vasudevan , Intel Technology India Private Limited, Bangalore and Dr. Madhuri Bhavsar, Computer Science and Engineering, Nirma University, Ahmedabad. This Project has been submitted in the partial fulfillment of the requirements for the award of degree Master of Technology (M.Tech.) in Computer Science and Engineering, Nirma University, Ahmedabad during the year 2013 - 2014.

I have not submitted this work in full or part to any viable University or Institution for the grant of any possible degree.

Nirav Patel(12MCEC21)

CERTIFICATE

This is to certify that the Major Project entitled **Design & Development of automated validation framework for INTEL Graphics display driver** submitted by **Nirav Patel(12MCEC21)**, towards the partial fulfillment of the requirements for the degree of Master of Technology in Computer Science Engineering of Nirma University of Science and Technology, 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, have not been submitted to any other university or institution for award of any degree or diploma.

Mr. Praveen Vasudevan
External Guide,
Intel Technology India Ltd.

Dr. Madhuri Bhavsar
Internal Guide,
Nirma University

Mr. Sridharan Manjunadh
Project Manager,
Intel Technology India Ltd.

Prof. Vijay Ukani
PG Coordinator - CSE,
Nirma University

Dr. Ketan Kotecha
Director,
Nirma University

Dr. Sanjay Garg
HOD - CSE,
Nirma University

ACKNOWLEDGEMENT

First and foremost, sincere thanks to **Mr. Sridharan Manjunadh** Manager, Intel Technology India Private Limited, Bangalore. I enjoyed his vast knowledge and owe him lots of gratitude for having a profound impact on this report.

I would like to thank my Mentor, **Mr. Praveen Vasudevan**, Intel Technology India Private Limited, Bangalore for his valuable guidance. Throughout the training, he has given me much valuable advice on project work. Without him, this project work would never have been completed.

I would also like to thank my Internel guide **Dr. Madhuri Bhavsar**, Institute of Technology, Nirma University, Ahmedabad for her valuable guidance.

I would also like to thank **Dr. K Kotecha**, Director, Institute of Technology, Nirma University, Ahmedabad and **Dr. Sanjay Garg**, HOD - CSE , Institute of Technology, Nirma University, Ahmedabad for providing me an opportunity to get an internship at Intel Technology India Private Limited, Bangalore.

I would like to thank my all faculty members for providing encouragement, exchanging knowledge during my post-graduate program.

I also owe my colleagues in the Intel, special thanks for helping me on this path and for making project at Intel more enjoyable.

Nirav Patel(12MCEC21)

Contents

List of Figures

Abstract

Graphics is unavoidable part of computer.With increase of different architectures and features of graphics,Graphics Drivers are also becoming complex . To meet this pace, driver validation also needs to be updated. Software validation is an important means to ensure software quality and to improve software reliability. With the volume and complexity of software applications continue to increase, people's requirements on software quality are also rising; the function of software validation in the process of software development has become more and more important, while software validation effort also appears to be increasingly difficult.

The proposal is to define a systematic approach to achieve optimization and efficiency improvement by considering overall validation process.Test content optimization and Test content Automation for Intel Graphics Display Driver so as to achieve better Quality end products.

The focus is on the Automation of Intel Graphics Display Driver validation. Currently these types of tests are executed manually so they need more manual efforts, time and accuracy is also not guaranteed.The proposal is to automate the validation process or test related to Intel Graphics Display driver. By using different h/w, S/w tools and algorithms, the validation process can be automated which can reduce the manual efforts, reduce the cycle time and increase accuracy.This automation includes objective assessment of different graphics driver features.

Chapter 1

Introduction to Computer Graphics and Graphics Driver

The term computer graphics includes almost everything on computers that is not text or sound. Today almost every computer can do some graphics, and people have even come to expect to control their computer through icons and pictures rather than just by typing.

1.1 Graphics Overview

Computer Graphics covers the following areas:

- **2D** : Mainly concerned about 2-dimentional representation of 2D images.
- **3D** : 2D representation of 3D models which are rendered dynamically.
- **Media** : Encoding, Decoding, Content Protection, Transcoding.
- **Display** : All these 3 scenarios result in pixel updates in frame buffer. Display engine takes care of streaming from Frame buffer to end display port(s) .

Most of these tasks involve SW/HW interaction therefor OS needs a VBIOS & graphics driver to assist in most of these activities.

1.2 Graphics Terminologies

- **Pixel** : Atomic unit of display in memory or on a display. It has two main properties: Position and Color.

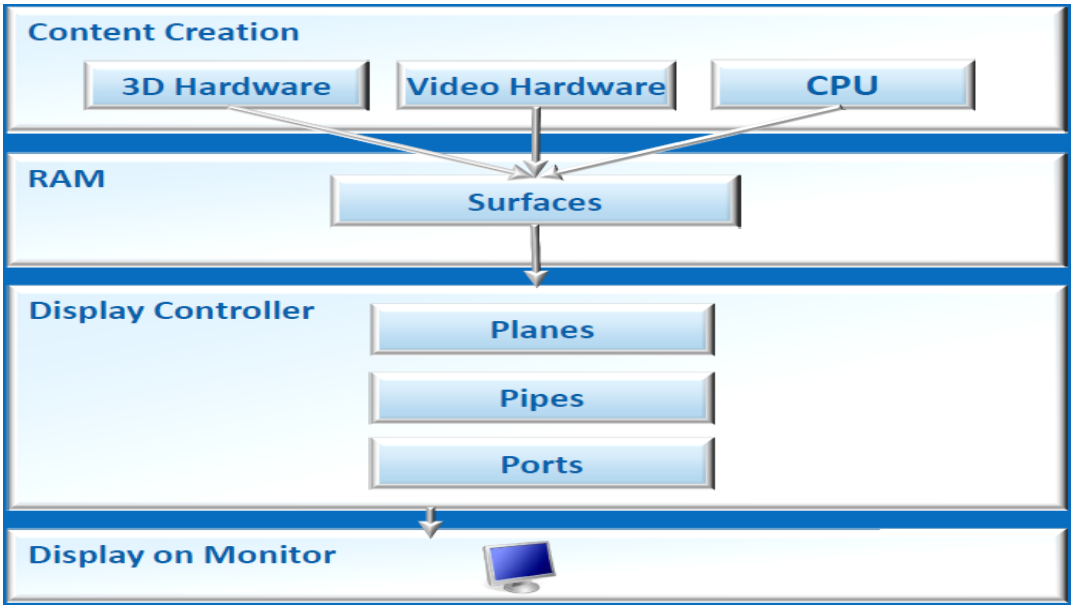

Figure 1.1: Graphics Overview
[7]

- **Resolution** : Width and height dimensions of the display in pixels. Eg: 1024x768, 1280x720 etc.
- **Refresh Rate** : Rate at which display needs to be refreshed (e.g. 60Hz, 75Hz etc.).
- **Color depth** : Number of bits in the one pixel.
- **Frame Buffer** : An allocation of memory that stores every pixel that is to be processed and/or displayed.
- **Aspect Ratio** : Relationship between width of the physical display and its height. Eg: 4:3 (1024x768), 16:9 (1600x900)
- **Pixel/Dot Clock** : How quickly a single dot/pixel can be produced on the screen. Pixel Clock = Htotal * Vtotal * RR
- **Frame Per Second** : Rate at which GPU produces unique consecutive images/frames in 1 second.
- **Display Engine/Controller** : Output component of a GPU
  - Presents text and graphical data to display monitors

- Display driver sends frame Buffer data to the monitor through a cable
- Controls, adjusts, modifies adapter output for monitor
- May also control, adjust or modify the receiving monitor itself

1.3 Display Controller

- **Planes** : A display plane can be thought of as a rectangular image delivered in time to the pipe . A display plane has characteristics such as source (location in memory), size (X x Y), position ((0,0) for primary or (X,0) for secondary extended display), format (RGB, Colordepth) & associated destination pipe (which pipe the plane is assigned to).

- **Pipes** : Its about Time Provide output pixels, at the right time to the Ports The blender is synchronizing and mixing the pixels data from all the display planes.
  Base Plane (ex: 800x600@16bpp)
  + Overlay (ex: 160x120 YUV)
  + Cursor (ex: 64x64@32bpp)
  Panel fitting Color correction

- **Ports- Display Buses** :
  - The output of the display controller
  - These are the places where the data finally appears to devices outside the graphics device
  - Ports can be internal or external (like LVDS or HDMI)
  - Port has characteristics such as signals (Hsync, Vsync), Format (RGB) etc.
  - Handles power management of the external devices

1.4 Display Interfaces

- **Analog Display Connector** :
  - VGA (Video Graphics Array) analog display technology

- **Digital Display Connectors** :

- Display Port Royalty free digital display specification
- HDMI (High Definition Multimedia Interface) - High definition digital display plus audio
- DVI (Digital Visual Interface) Digital display

- **Mobile Display Connectors** :
  - LVDS (Low-voltage Differential Signaling) : low voltage interface for mobile, usually internal to the PC
  - Embedded Display Port : Low power Display Port for mobile or all-in-one, internal to the PC
  - MIPI (Mobile Industry Peripheral Interface) : Interface for mobile devices such as mobile phones, tablets & netbooks

1.5 Display Modes

- **One Display** :
  - SD : Single Display Mode
    Only one display is active at a time

- **Two Displays** :
  - DDC : Dual Display Clone
    Two displays are active at a time and same content is displayed on both the displays
  - ED : Extended Mode
    Two displays are active and one display work as Primary display and other as secondary which is extended part of primary display

- **Three Displays** :
  - TDC : Tri Displays Clone
    Three displays are active at a time and same content is displayed on all the displays

– TED : Tri Extended Mode

Three displays are active and one display work as Primary display and others are extended part of primary display

1.6 Graphics driver role

The graphics driver interfaces between the operating system and the GPU hardware. Each graphics driver is written for specific graphics hardware and effectively translates the task at hand to be executed most efficiently on the available graphics hardware.

Graphics hardware might support a specific feature, but if there are no graphics drivers to support it, the feature is useless.

1.7 Graphics Driver Architecture

Grpahics Driver has major 3 components

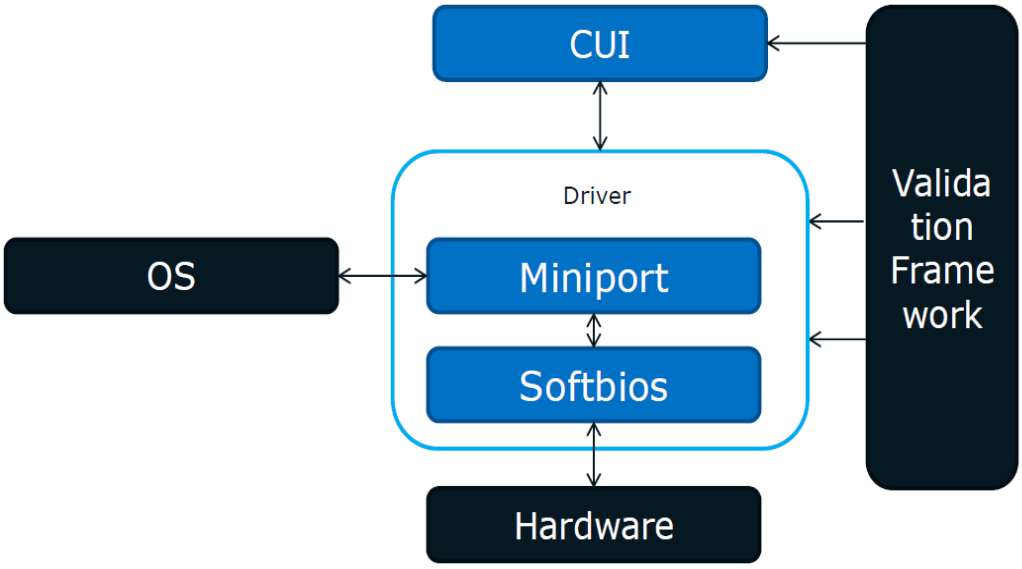

Figure 1.2: Graphics Driver Architecture
[7]

- **CUI** : Common User Interface

  This is the only user Interface module of Driver which is OEM customizable and provide localization support.

- **Miniport** : This component of driver is responsible for adapter initialization, interrupts, MMIO mapping etc. Entry point for all IOCTL calls from OS
- **SoftBios** : Low level Display Component Manages Display Pipe, DPLL, Planes and Ports . It manages capability enumeration of display side Handles Mode Set, display devices, I2C/GMBus etc.

Driver can run in two different Modes

- **User Mode** : Runs in the non-privileged processor mode in which other application code executes. User-mode drivers cannot gain access to system data except by calling the Win32 API which, in turn, calls system services Different structures, different entry points, and different system interfaces.
  E.g. USB drivers, Vista graphics UMD etc.
- **Kernel Mode** : Runs as part of the operating system in a privileged environment Kernel-mode drivers can perform certain protected operations and can access system structures that user-mode drivers cannot access.
  E.g. Miniport drivers

Chapter 2

Post Silicon Validation and Automation

In the process of development of a semiconductor integrated circuit the Post-silicon validation[5] is last mile. The pre silicon validation process is done through different devices in virtual environment with emulation and simulation tools. But in the post silicon validation actual tools and devices used for run the tests, original and real time setup boards are validated with analyzer and tools.

Big semiconductor companies spent large amount of money for making new designs and components.that is the "sunk costs" of design. In parallel it is imperative that new design should reach in market within customer window. The little delay in this delivery of product costs a large amount of money.So the post silicon validation is most important part in successful design.

Similarity and dependability issues are discovered and determined early, and the results are utilized to further refine Intel's configuration and assembling devices and methodologies. As a result, the quality and dependability of Intel stages and parts have enhanced consistently, even as execution and multifaceted nature have kept on climbing[5]

There are essentially five stages in Intel's far reaching post silicon approval system:

- Stage 1: System Validation

- Stage 2: Analog Validation
- Stage 3: Compatibility Validation
- Stage 4: Software Validation
- Stage 5: Product Qualification

2.1 System Validation

System acceptance puts the genuine part through a thorough suite of tests in a genuine nature's domain. The test suites are connected to the Intel CPU, chipset also design subsystem, recurrence and temperature conditions are strengthened to test execution at the great corners of the segment's particulars.

- **CPUs** : Both deliberate and irregular tests are utilized to blanket profound information space and escalated coasting point requests. Framework approval tests for the Intel Pentium 4 processor over a great sample of the extension and power of the methodology:
- **Chipsets** : All chipset characteristics are tried utilizing custom-fabricated framework acceptance sheets and test cards running custom programming to stretch every interface of the chipset. Execution parameters are pushed as far as possible on all cards and transports simultaneously, to accept execution limits and to confirm transport similarity.
- **Graphics** : Particular apparatuses and test suites are utilized for acceptance testing of the representation subsystem in all Intel chipsets with coordinated illustrations. Extraordinary test pictures are made to guarantee a thorough pattern for robotized testing. In the event that a test reports even a single wrong bit, the main driver is controlled by an acceptance design, so that the issue could be completely determined to guarantee exceptional visual quality with no deformities.

2.2 Analog Validation

As PC execution requests keep on climbing, the electrical honesty of processor and chipsets is crucial to guarantee dependable operation at high frequencies. Intel's simple acceptance testing discovers disappointments that can happen in only trillionths of

a second. Segments are pushed to disappointment at the extremes of temperature, voltage and Recurrence. Issues are determined, and discoveries are imparted to plan and processing designs so as to enhance Intel's outline and generation systems.

- **Circuit Marginality Validation (CMV)** : Throughout CMV, the test suites that were utilized throughout presilicon recreation of Intel processors are connected once more, yet this time with a concentrate on unwavering quality and execution under great working conditions. Both business and custom tests are used to test voltage, recurrence and temperature extremes, and to guarantee solid operation inside the item's particulars.

- **Analog Integrity Engineering (AIE)** : AIE tests the trustworthiness of the chipset, to verify the whole stage is electrically hearty. Execution is accepted under a wide mixed bag of most detrimental possibility situations. The electrical power of the chipset and processor is accepted under true situations.

2.3 Compatibility Validation

A key playing point of Intel structural planning is its wide similarity with outsider hardware parts, programming provisions and working frameworks. Intel segments what's more stages experience exhaustive similarity, anxiety and concurrency testing with in excess of 20 working frameworks, various motherboards and include cards, more than 150 peripherals, and more than 400 requisitions. OS testing incorporates different variants of Microsoft Windows, Netware, SCO, Unixware and Linux. Provision testing incorporates huge numbers of the world's most well known business and sight and sound projects, also as various recreations, industry benchmarks and industry fittings tests.

Gigantic record exchanges and telecasts test execution and information coherency utilizing a extensive variety of conventions, including Ethernet, Fast Ethernet, Gigabit Ethernet, and Fiber Channel. Notwithstanding these well-known fittings and programming items and conventions, the segment is tried with extraordinarily planned cards that push test parameters past accepted breaking points, to guarantee predominant execution under most detrimental possibility- conditions. For instance, mixed media activity is expanded to the data transfer capacity point of confinement of the PCI transport, to check that sound and feature signs don't split under top workloads.

2.4 Software Validation

Intel creates all center programming segments for its PC and server stages. This incorporates the BIOS and the drivers that are utilized for representation, stockpiling and LAN network. All around this procedure, programming acceptance is hard coordinated with equipment approval to guarantee that fittings and programming segments work easily together. This is vital to approve that the aggregate stage will convey top execution and unwavering quality in the greatest conceivable extent of situations

2.5 Product Qualification

This is a last stage in Validation cycle, in this stage all the subsystems like representation, PCI gadgets are united on stage and the conduct of silicon is tried. The similarity and interoperability of all segments on a stage is approved furthermore any client issues will be determined. Throughout this last stage, segment abilities are likewise contrasted and current end-client desires. In the event that an item effectively passes this testing, it is primed to enter the commercial center.[6]

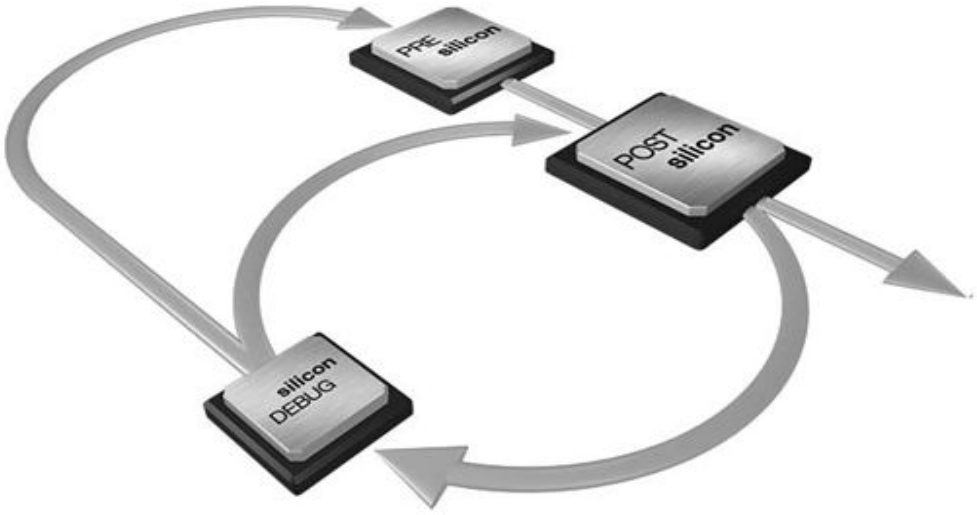

Figure 2.1: Silicon Validation
[5]

2.6 Role of Automation Validation

Programming approval is an essential stage in programming life cycle. Quality approval specifically decides the stable operation of programming items. It is well realized that the time it now, prolonged and exhausting much for programming manual acceptance with high work power and effectively presented counterfeit error.therefore it is basic to the mechanization test now. Computerized acceptance engineering in programming approval is to be further enhanced, as work heap of programming acceptance is exceptionally large(accounting for something like 40-50 rate of general advancement cycle), of which the vast majority of the work applies to computerization, so the test change will bring extremely huge results to cost, quality and cycle of the entire programming activities improvement work. As a rule, it is through the improvement of the mechanization apparatuses and the execution of the approval scripts for computerization acceptance to attain the reason for programming quality assessment.

- **Drawbacks of traditional software validation**
  - The project process is difficult to control
  - Difficult to control project risk
  - Project development costs exceed budget
- **In practice, we have to move for automated validation to decrease validation engineers manual efforts**
  - Automation validation saves 80% of efforts of validation engineers which ultimately
    - Improve test efficiency
    - Reduce cost of test
    - Enable to cover more aspects of validation

2.7 Automation Process

- **Test tool selection**

  Test Tool determination generally relies on upon the innovation the Application

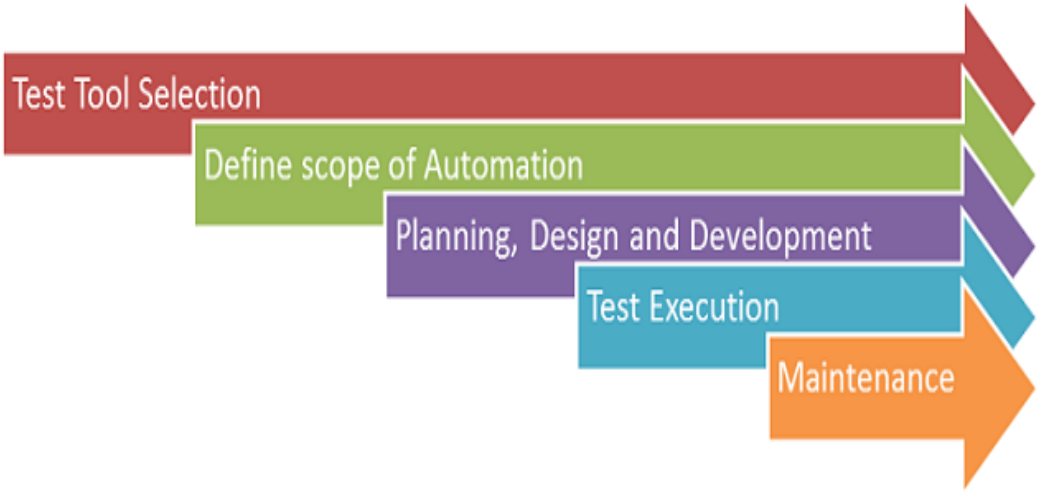

Figure 2.2: Automation Process
[1]

Under Test is based on. It's a great thought to direct Proof of Concept of Tool on Automation

- **Define the scope of Automation**
  scope of mechanization is the region of your Application Under Test which will be mechanized. Emulating focuses help focus scope:
  - Features that are essential for the business
  - Scenarios which have extensive measure of information
  - Common functionalities crosswise over requisitions
  - Technical possibility
  - Extent to which business parts are reused
  - Complexity of experiments
  - Ability to utilize the same experiments for cross Platform testing
- **Planning, Design and Development**
  during this stage you make Automation procedure & plan, which holds succeeding subtle elements

- Automation apparatuses selected
- Framework configuration and its characteristics
- In-Scope and Out-of-extension things of computerization
- Automation proving ground readiness
- Schedule and Timeline of scripting and execution
- Complexity of experiments
- Deliverables of mechanization testing

- **Test Execution**

  automation Scripts are executed throughout this stage. The scripts need information test information before they are situated to run. Once executed they give itemized test reports. Execution could be performed utilizing the mechanization apparatus straightforwardly or through the Test Management instrument which will conjure the robotization device.

- **Maintenance**

  as new functionalities are added to the System Under Test with progressive cycles, Automation Scripts need to be included, surveyed and kept up for each one discharge cycle. Support gets important to enhance viability of Automation Scripts.

Chapter 3

Literature Survey

Exploring the Use of a Test Automation Framework [1], provides the proof that As technology continues to grow and become more complex, software testers will be faced with tougher challenges to fully test the software product within the time given to them. To keep up with this trend, testers must consistently look for ways to improve their testing practices. A test automation framework is a tool that can help a tester efficiently develop end-to-end automated test solutions. So before any organization decides to commit to develop a test automation framework, it would be wise to first explore its use within your testing environment to ensure that it is suitable to your software testing needs.

Examination of Software Automation Test Protocol [2], shows that In practice, we need to consolidate characteristics of customary manual testing and computerized testing, play their qualities, so mechanized testing procedures and apparatuses function as a weapon in the hands of analyzers, discharge them from tedious work, centering time and vitality on intricate work obliging savvy judgment and other new test cases.

In expansion, we must keep away from compare computerized testing and analyzers together, not approaching excessively for mechanized testing. We might set a fitting computerized testing discernment, have an acceptable comprehension that manual testing is an influential supplement to mechanized testing and not supplant the status of the analyzer. Any single innovation or operation propels can't freely guarantee a substantial scale of upgrades to software development effectiveness, soundness and practicality in a short time. Mechanized testing is additionally an amassing of experience, slow method-

ology, programming analyzers can't be required to robotize all the tests in a short time.

Effective mechanized testing obliges creating suitable robotized test project, and sensible computerized testing is the first venture in the achievement execution of the computerized testing system. Just by completely considering the danger of execution of computerized testing, assets and targets, it is conceivable to create mechanized testing procedures for your own, and eventually enhance test efficiency and decrease expense of test. Each one undertakings with testing division or occupied with testing business, ought to learn abroad and household progressed testing background, allude to mainstream industry-standard, discover their own particular group's trying routines and models to make more terrific social value.

Methods for Agile Software Testing Automation: An Industrial Experience [3], presents the experience in performing methods for test computerization in coordinated programming advancement Scrum. We could watch the impact of deft values on group association and test robotization. These encounters demonstrated that it is possible to extend cooperation in coordinated effort to settle issues, scan for coordinated testing apparatuses and keeping the administration and association of a product testing and scrum process.

The computerization testing is an asset to robotize dull undertakings, record programming, diminishes expense of venture utilizing open source test devices, and errand assignment in little parts. Then again, the software engineers' state of mind in these situations are extremely paramount to examination new methodologies and apparatuses. A few lessons educated could be concentrated and they could help other programming designers when performing dexterous test robotization method in a nature.

Chapter 4

Scope of the Project

- Reduce Cycle time of test cases.
- Reduce Manual efforts in executing test cases.
- Increase the accuracy of test results.
- Increase productivity.

Chapter 5

Technology

5.1 Frameworks

- **Microsoft UI Automation**

  Microsoft UI Automation is the new availability skeleton for Microsoft Windows, accessible on all working frameworks that help Windows Presentation Foundation (WPF). UI Automation gives automatic access to most client interface (UI) components on the desktop, empowering assistive innovation items, for example, screen book lovers to give data about the UI to end clients and to control the UI by means other than standard info. UI Automation additionally permits robotized test scripts to communicate with the UI.

- **.NET Framework**

  .net Framework is a product system created by Microsoft that runs essential on Microsoft Windows. It incorporates a vast library and gives dialect interoperability over a few programming dialects. programs composed for .NET Framework execute in a product environment , known as the Common Language Runtime (CLR), a provision virtual machine that gives administrations, for example, security, memory administration, and special case taking care of. The class library and the CLR together constitute .NET Framework.

5.2 Tools

- **UI Spy**

  UISpy is a Microsoft tool for viewing the UiAutomation element tree, and perform-

ing actions on elements. This tree is almost the same as the Twin ElementTree. UISpy can be a useful tool for determining the structure of your target application and determining the Criteria you should use to identify elements

- **Ranorex**

  ranorex is a GUI test robotization structure for testing of desktop, electronic and portable requisitions.

  Ranorex is given by Ranorex Gmbh, a product improvement organization for creative programming test robotization results. Ranorex does not have a scripting dialect of its own, rather utilizing standard programming dialects, for example, C# and Vb.net as a base.

  Main characteristics

  - GUI Object Recognition - Ranorexpath for recognizable proof/separating of numerous types of GUI components.

    ranorex Spy gives the mapping data of GUI components to their Ranorexpath statement.

  - Object-based Capture/Replay usefulness (called Ranorex Recorder), which gives viable recordings by means of an activities table editor,transforms recorded movements into C# and Vb.net code and creates nitty gritty report records for speedy blunder discovery.

  - Test Automation Library for .Net

  - Test Development Environment (called Ranorex Studio) makes code fulfillment, debugging and test undertaking administration conceivable.

  - Flexible Test Automation Interface: Test Automation inside particular test situations is conceivable. Test suites with Ranorex brings about .EXE documents for straightforward incorporation into existing situations, for example, test administration instruments, nonstop reconciliation courses of action or group execution scenarios.they could be effortlessly run by propelling the .EXE record from the charge line.

Chapter 6

Design of Display Automation Framework

- **Objective**

  The objective of this framework is to verify quality of Intel graphics driver display component. Use of this framework determines different quality metrics on different configurations and systems and on the basis of these quality metrics the quality of graphics driver display component verifies. The configurations and systems are selected as it cover all types of end user experiences.

- **Automation Model**

  - Possible Input :

    Automation tests can have manual inputs or it can also be leveraged as plug and play functionality

  - Expected Output :

    Logs , Reports or execution time can be possible output of automation tests.

- **Interaction with graphics driver**

  Framework interacts with graphics driver using

  - Driver's API's
  - OS API's
  - Using Test Players , same as user interacts with driver

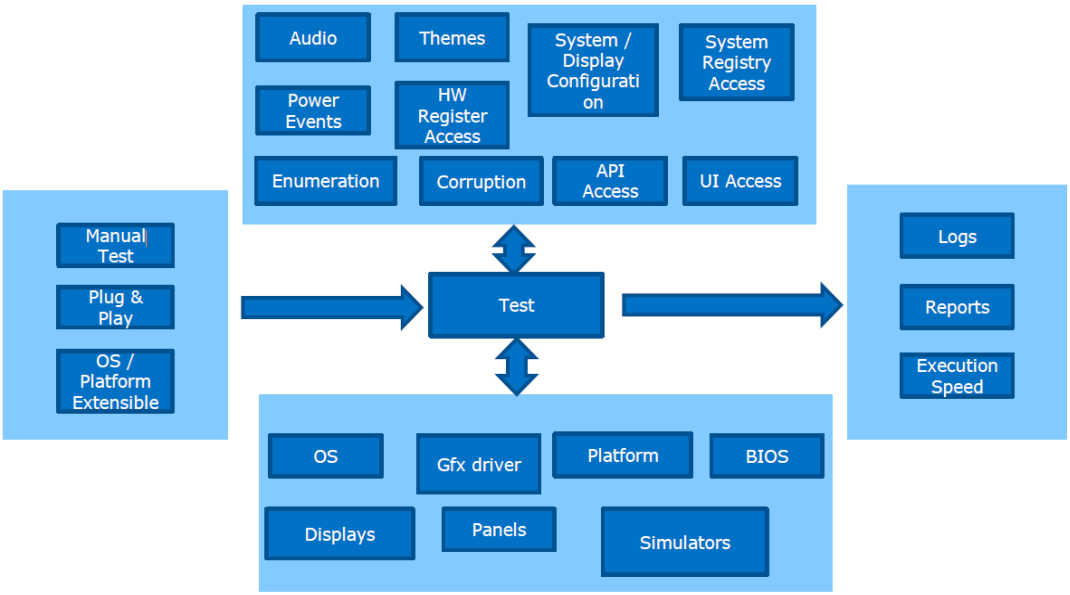

Figure 6.1: Automation Model
[6]

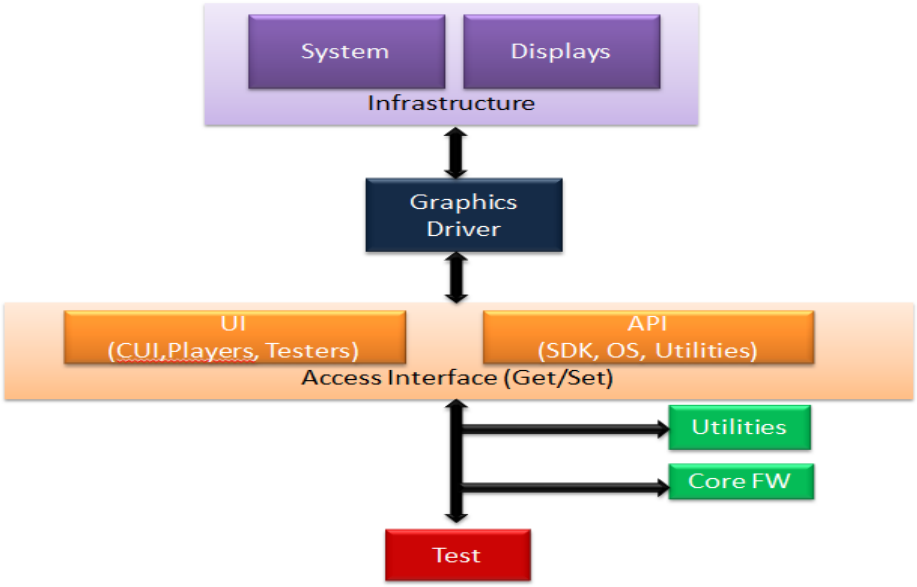

Figure 6.2: Framework interaction with Driver
[7]

Chapter 7

Features Implemented

Following are some of the features implemented in Automation Framework

7.1 Display switch

- **Description**

  Display configuration like Single display , Double display clone, Extended display and so on can be changed using graphics driver and also using OS Page . Users frequently change these configurations which is known as Display switch feature(Switching between the displays)

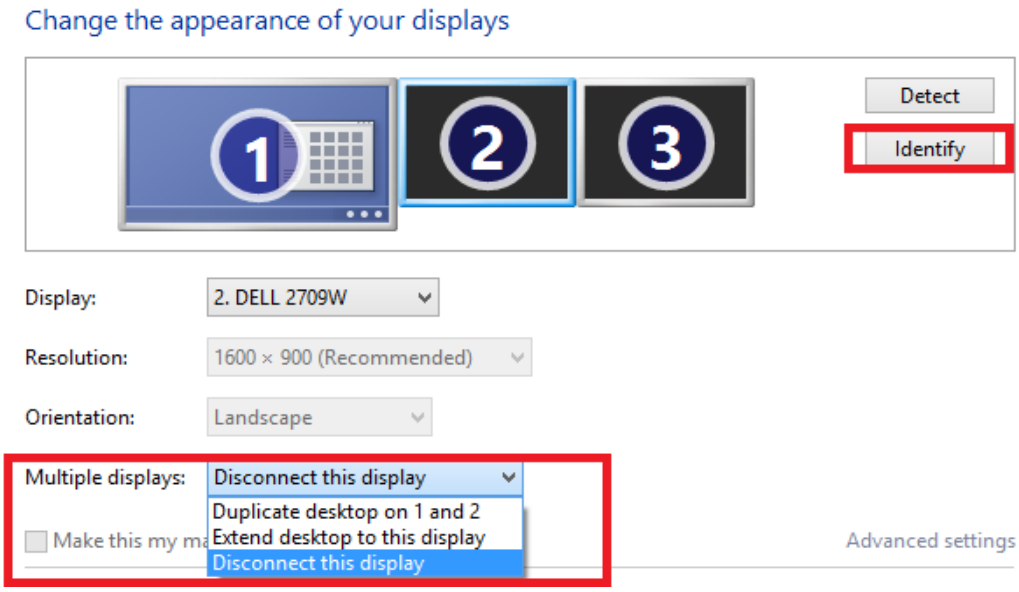

Figure 7.1: OS Screen resolution page options to switch between display configurations [7]

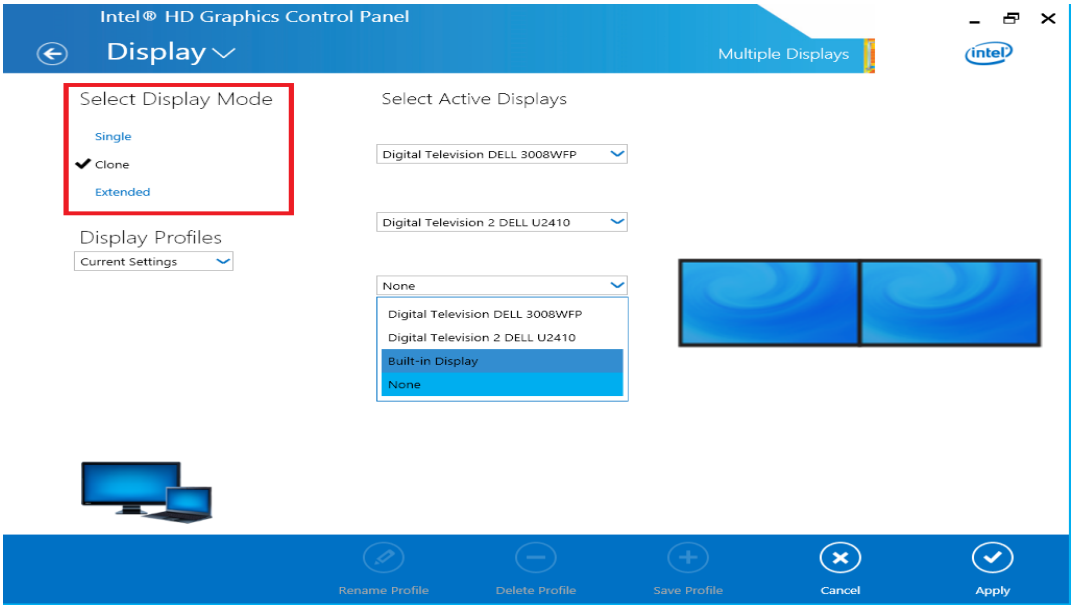

Figure 7.2: CUI page options to switch between display configurations [7]

- **Expected Output**

  The expectation is that driver should handle any display configuration change request from OS and CUI should immediately reflect the new configuration if configuration change is successful.

- **Result**

  - Feature is successfully implemented in Framework. Automation framework is capable to validate this feature for any INTEL platform with any flavour of Windows OS.
  - Validated 4 versions of driver with HDMI , DP , VGA connectors , No bug found till date

7.2 Mode Change

- **Description**

  Combination of Resolution, Refresh rate, Bpp is know as Mode(x,y,rr,bpp). Mode can be changed using driver CUI page or OS page. Verification of modes listed by CUI and OS are in synchronization is needed.

- **Expected Output**

  Modes (x,y,rr,bpp) listed in CUI should match with the modes listed in OS Page and CUI page should show the same resolution,RR,bpp that was applied using OS Page

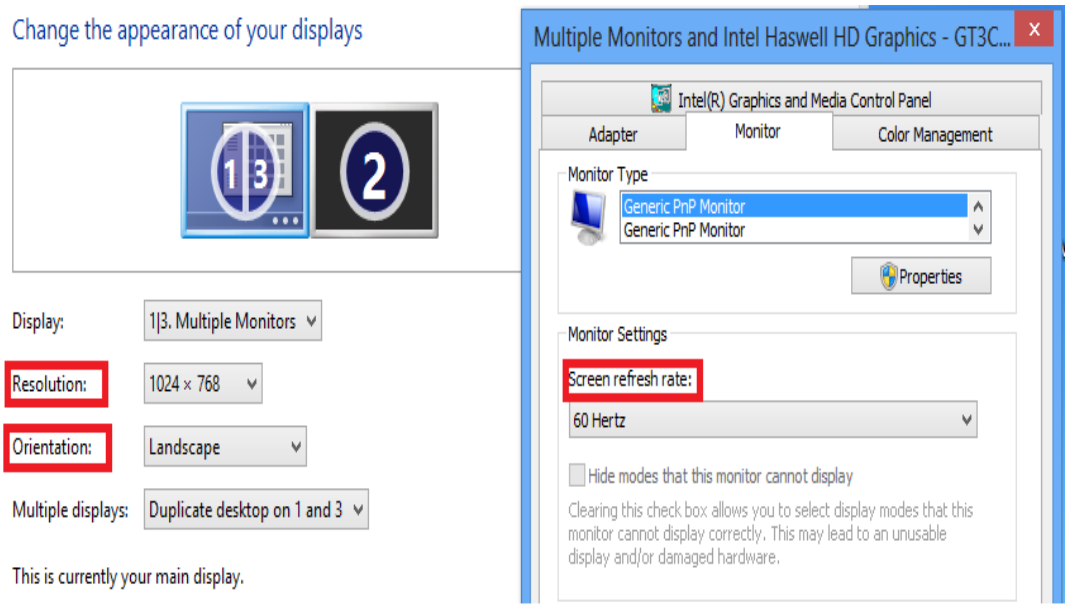

Figure 7.3: OS Screen resolution page options to change Modes [7]

- **Result**
  - Feature is successfully implemented in Framework. Automation framework is capable to validate this feature for any INTEL platform with any flavour of Windows OS.
  - Bug reported for driver version 15.32 - for some Resolution, CUI SDK API is failing to set Mode listed by OS (RR is missing)
  - Bug Type - Heisenbug

7.3 Deep Color

- **Description**
  - Deep color is a term used to describe a gamut comprising a billion or more colors

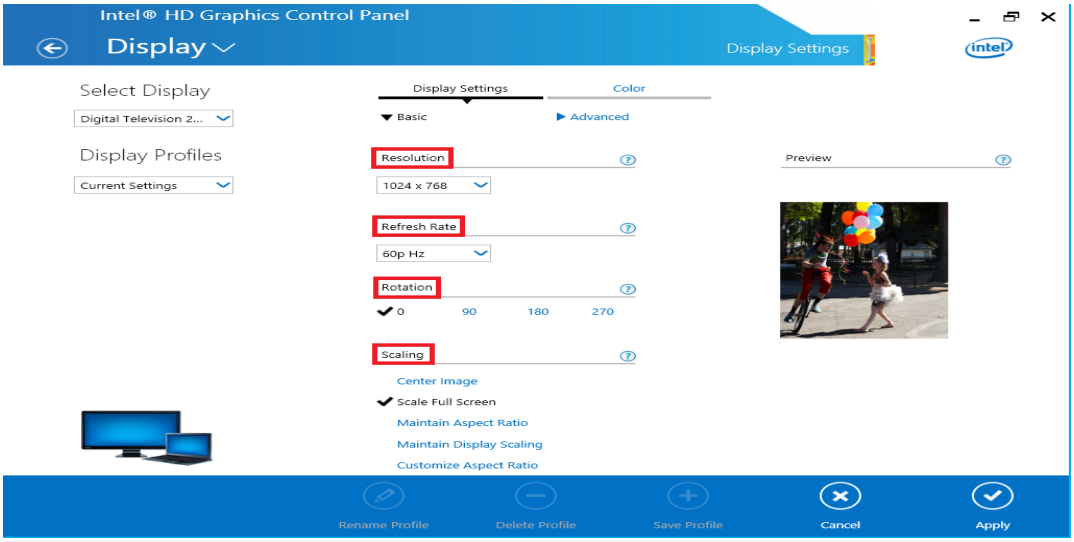

Figure 7.4: CUI page options to change Modes [7]

- Types of Color format:
  - 8 bits per color(24bit color format)
  - 10 bits per color(30bit color format)
  - 12 bits per color(36bit color format)
  - 16 bits per color(48bit color format)
- Color depths greater than 24 bits are defined as Deep Color.
- Deep color is supported for RGB 4:4:4 and YUV 4:4:4 pixel data
- Panel must support Deep color to use this feature.
- 8 bit color/24 bits per pixel:
  -256 levels of gradation
  -256 x 256 x 256 = 16,777,216 possible colors:17 Million colors.
- 10 bit color/30 bits per pixel:
  -1024 levels of gradation
  -1024 x 1024 x 1024 = 1,073,741,824 possible colors: 1 billion colors
- **Extended display identification data (EDID)** is a data structure provided by a digital display to describe its capabilities to a video source (e.g. graphics card or set-top box).

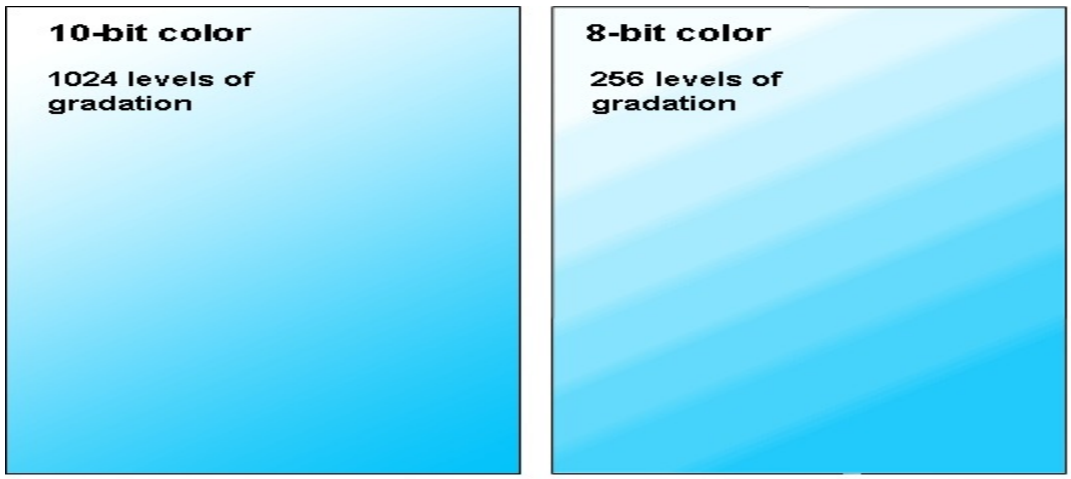

Figure 7.5: Comparasion 8 bit color/24 bits per pixel Vs 10 bit color/30 bits per pixel [7]

- **Expected Output**

  DeepColor supported panel must enable this feature after running Deepcolor enable Applicaion on that panel or in clone mode and not supported panel must be in the same state

- **Result**

  - Feature is successfully implemented in Framework. Automation framework is capable to validate this feature for any INTEL platform with any flavour of Windows OS with all Display Mode Combination
  - 10 OEMs' Deepcolor supported panels are validated , No bug found till date

7.4 LPSP

- **Description**

  - LPSP : Low power state performance
  - LPSP is Power feature of Graphics driver
  - LPSP is

    Enable : EDP display is active

    Disable : Other than EDP display is active

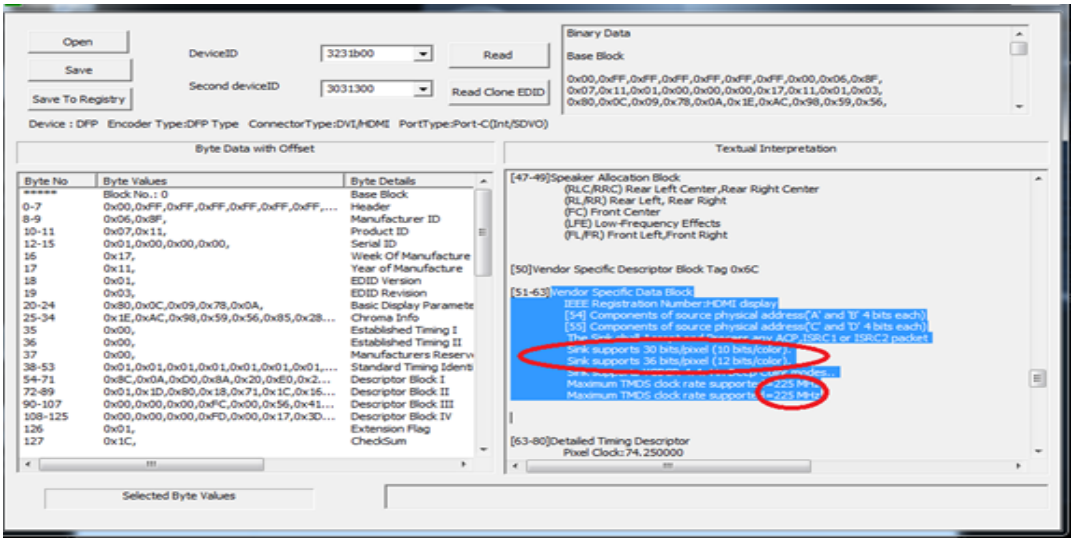

Figure 7.6: Read the EDID of the HDMI panel to know DeepColor Support [7]

Display driver enable LPSP to improve power performance of system by disabling Power well in case system is running with EDP display (System must be running with Native mode of Haswall platform) .

- **Expected Output**

  The expectation is that driver should enable LPSP feature (turn off power well component) when system is running in SD - EDP configuration with native mode for Haswall platform and any mode for Broadwell platform

- **Result**

  - Feature is successfully implemented in Framework. Automation framework is capable to validate this feature for any INTEL platform with any flavour of Windows OS.
  - Vlidated LPSP feature for 50 Panels from different OEMs . No bug found till date

7.5 Scaling

- **Description**

  Display Driver provides different scaling options like

  - Center Image

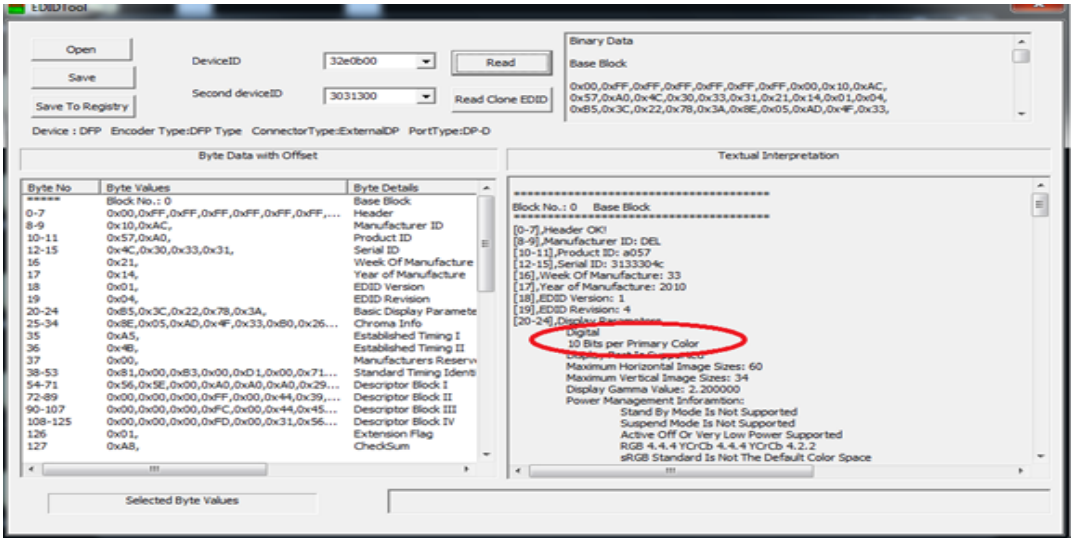

Figure 7.7: Read the EDID of the DP panel to know DeepColor Support [7]

- Scale Full Screen
- Maintain Aspect Ratio
- Maintain Display Scaling
- Custom Aspect Ratio : CAR

Driver programs registers for Scaling change ,
For CAR scaling

- Resolution X and Y may not be the same as applied(Max 12% deviation is expected)
- Driver computes new values for resolution and program registers

- **Expected Output**

  The expectation is that driver should program Source and Target Registers correctly . In case of CAR Scaling , resolution deviation must be less than or equal to 12

- **Result**
  - Feature is successfully implemented in Framework. Automation framework is capable to validate this feature for any INTEL platform with any flavour of Windows OS.

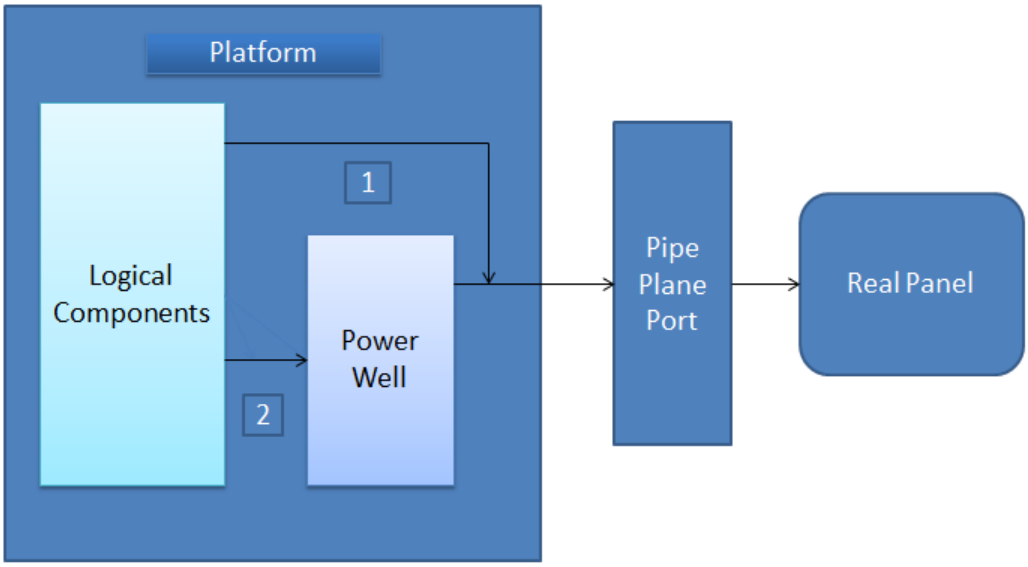

Figure 7.8: INTEL Architecture to understand LPSP
[7]

- Bug reported for CAR scaling option : Resolution deviation is not in the permitted limit for driver version 15.36 (Broadwell platform only)
- Bug Type - Bohrbug

7.6 Corruption check

- **Description**

  While performing any operation Display Driver may create frame buffer data which are corrupted. Corruption check mechanisum is high priority demand for driver validation.

  Automation framework is provided with corruption check mechanisum which can be used with any feature.

  There are two methods for corruption check

  - CRC Engine

    Intel chipset has inbuilt funtionality to calculate CRC(Cyclic redundent code) using CRC Engine . This CRC is XOR of all pixel values in framebuffer data. Computed CRC is stored in register. Automation framework is using this value as golden value . Golden value is used later on to compare with run time image data to identify corruption

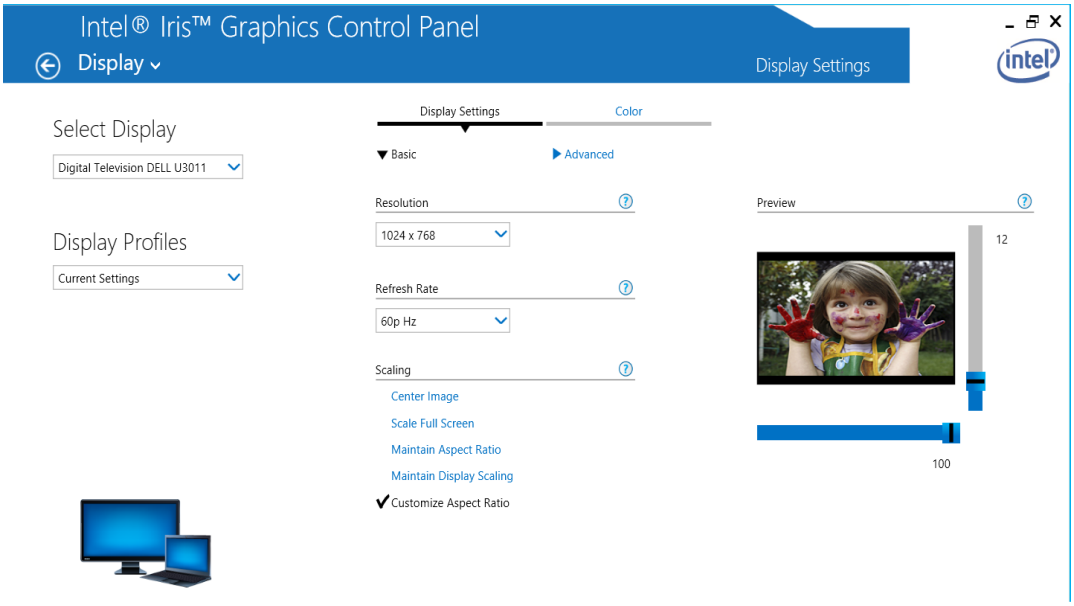

Figure 7.9: Scaling options in CUI
[7]

- DVMU - Display simulator

  DVMU is simulator for real displays , developed by INTEL platform engineering team. DVMU provides CRC value for image through API. Automation framework is using this API to get CRC of any image which is used to identify corruption

- **Expected Output**

  The expectation is that driver should not create corrupted frame buffer data .

- **Result**

  - Corruption check is successfully implemented in Framework. Automation framework is capable to check corruption along with driver features for any INTEL platform with any flavour of Windows OS .
  - 15 different Driver versions are validated till date, Bug is reported for Haswell driver version 15.33 (Driver development team has accepted driver corruption)
  - Bug Type - Bohrbug

7.7 Achievement

- Use of Automation Framework validate above features in few hours with precision which takes days by manual validation
- Types of Bugs found
  - Heisenbug : Software bug that seems to disappear or alter its behaviour when one attempts to study it
  - Bohrbug : Do not change their behaviour and are relatively easily detected , 'good, solid bug'
- Automation framework has improved reliability

  Reliability can be majored by MTBF ,

  MTBF = MTTF + MTTR [7]

  Where ,

  MTBF = Mean time between failure

  MTTF = Mean time to failure

  MTTR = Mean time to repair

  Use of automation framework has highlighted bugs which are fixed quickly and reduced MTTF and improved reliability
- Automation framework has improved validation coverage

  Adding new feature to current version of software , needs to write new test cases.

  Count of new test cases = (N/R) * T [7],

  Where ,

  N = probability of occurrence of new operations for new release of the software,

  R = probability of occurrence of used operations in the current release

  T = number of all previously used test cases

  for graphics driver new release ,

  R = 0.9

  N = 0.2

  T(Manual) = 20

  T(Automated framework) = 200

  Count(Manual) = (0.9 / 0.2) * 20 = 90

Count(with automation framework) = (0.9 / 0.2) * 200 = 900

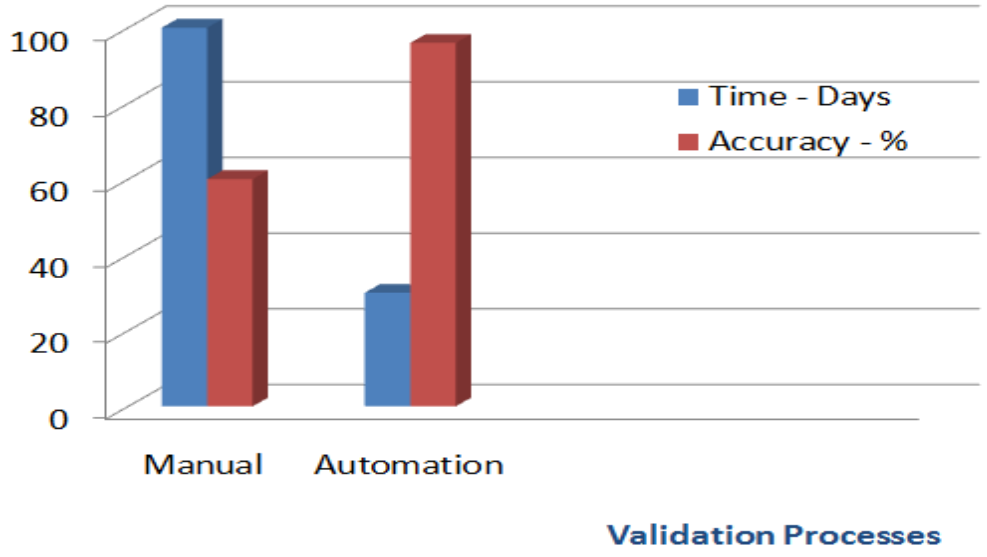

Figure 7.10: Automation framework achievement for LPSP [7]

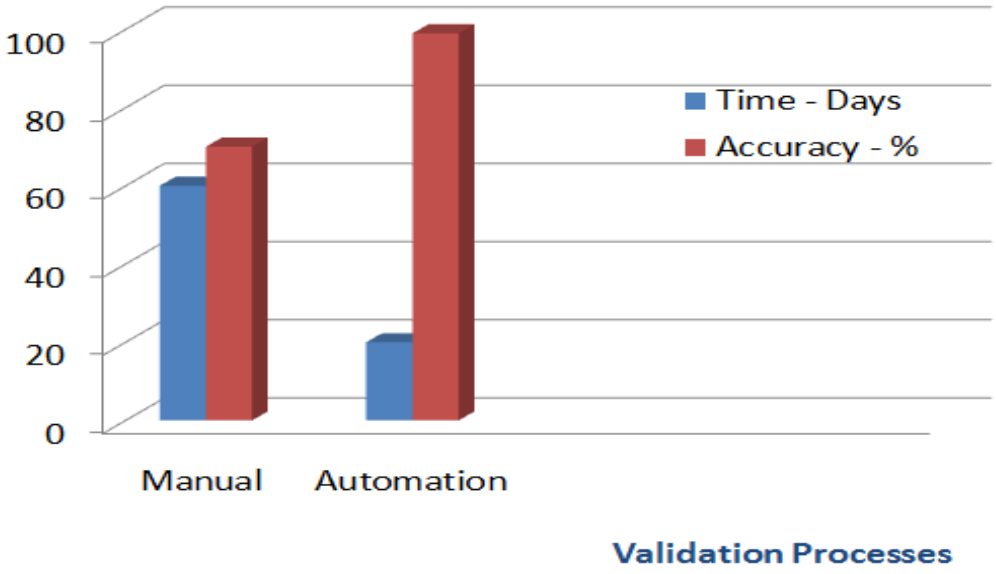

Figure 7.11: Automation framework achievement for Scaling [7]

Chapter 8

Conclusion

This report includes overview of Post silicon validation, framework for automation of Graphics display driver validation.

Automation of Graphics Display Driver Validation is very useful for Post silicon Validation. By usage of framework the validation becomes easier. By using this framework tedious manual efforts can be reduced, the automation can be run on different number of configurations for validation. Also using automation of graphics display driver validation gives more accurate and fast results with different quality matrices.

Bibliography

[1] Alex Cervantes, "Exploring the Use of a Test Automation Framework"

[2] XiangFeng Meng "Analysis of Software Automation Test Protocol " ,2011 International Conference on Electronic & Mechanical Engineering and Information Technology

[3] Eliane Collins , Arilo Dias-Neto, "Strategies for Agile Software Testing Automation: An Industrial Experience",2012 IEEE 36th International Conference on Computer Software and Applications Workshops

[4] Jeff Bisgrove,Tom Jones ,"Integrated Test Facility (ITQ-Automation Testing to Support Intels Manufacturing Output"

[5] Intel Developers,Intel Platform and Component Validation - Whitepaper,Intel Technology India Pvt. Ltd.

[6] Intel Developers,2012 client Product Requirement Document(PRD),Intel Technology India Pvt. Ltd.

[7] Other Intel Sources.