iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 5 **Issue:** VII **Month of publication:** July 2017

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:6.887
Volume 5 Issue VII, July 2017- Available at www.ijraset.com

Graphics Output Protocol (GOP) Driver for UEFI

Reethambari S V[1], Dr D Seshachalam[2]
[1]*Department of ECE BMS College of Engineering,Bangalore, India.*
[2]*Professor and former HOD, Dept of ECEBMS College of Engineering,Bangalore, India.*

Abstract: ***The BIOS (Basic Input/Output System and also known as the System BIOS, ROM BIOS or PC BIOS) is a type of firmware used to perform hardware initialization during the booting process on IBM PC compatible computers, and to provide runtime services for operating systems and programs[1]. Unified Extensible Firmware Interface (UEFI) was designed as a successor to BIOS, aiming to address its technical shortcomings. As of 2014, new PC hardware predominantly ships with UEFI firmware. The Unified Extensible Firmware Interface (UEFI) is a specification that defines a software interface between an operating system and platform firmware. UEFI replaces the Basic Input/Output System (BIOS) firmware interface originally present in all IBM PC-compatible personal computers. UEFI driver is a Loadable Image loaded by UEFI loader. These drivers may consume or produce protocols. The GOP (Graphics Output Protocol) Driver is part of the UEFI boot time drivers responsible for bring up the display during bios boot. This driver enables logo display during bios boot time. This paper has a GOP device driver written for an Intel's IoT Android platform which is responsible for display control until the operating system and in turn the display controller of the system gains the control.***

Keyword: BIOS, Unified Extensible Firmware Interface, GOP, Protocol, PCI, Simics, display.

I. INTRODUCTION

The GOP driver is a replacement for legacy video BIOS and enables the use of UEFI pre-boot firmware without CSM. The GOP driver can be 32-bit, 64-bit, or IA-64 with no binary compatibility. UEFI pre-boot firmware architecture (32/64-bit) must match the GOP driver architecture (32/64-bit). The Intel Embedded Graphics Drivers' GOP driver can either be fast boot (speed optimized and platform specific) or generic (platform agnostic for selective platforms).

EFI defines two types of services: boot services and runtime services. Boot services are available only while the firmware owns the platform (i.e., before the ExitBootServices call), and they include text and graphical consoles on various devices, and bus, block and file services. Runtime services are still accessible while the operating system is running; they include services such as date, time and NVRAM access. In addition, the *Graphics Output Protocol* (GOP) provides limited runtime services support. The operating system is permitted to directly write to the frame buffer provided by GOP during runtime mode. However, the ability to change video modes is lost after transitioning to runtime services mode until the OS graphics driver is loaded [2]. This paper includes a GOP driver written for an IoT's platform using the development kit EDK II which is responsible for the display during booting process until the operating system gains control of the display and invoke display devices.

II. BLOCK DIAGRAM

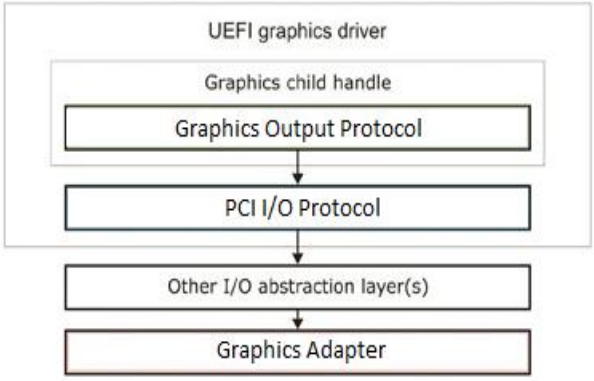


Fig 1 GOP driver

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:6.887
Volume 5 Issue VII, July 2017- Available at www.ijraset.com

III. SOFTWARE USED

A. SIMICS

Simics is a full-system simulator used to run unchanged production binaries of the target hardware at high-performance speeds. Simics was originally developed by the Swedish Institute of Computer Science (SICS), and then spun off to Virtutech for commercial development in 1998. Virtutech was acquired by Intel in 2010 and Simics is now marketed through Intel's subsidiary Wind River Systems.

Simics can simulate systems such as Alpha, x86-64, IA-64, ARM, MIPS (32- and 64-bit), MSP430, PowerPC (32- and 64-bit), POWER, SPARC-V8 and V9, and x86 CPUs. The current version of Simics is 5 and it is available for Microsoft Windows and Linux host platforms.

B. EDK II

The EFI Developer Kit (EDK) is an Open Source release of the Framework Foundations, defined in the Framework Core Interface Specifications (CIS), plus a set of sample drivers and three sample targets implemented for the Nt32, UNIX, and DUET platforms. In addition to Open Sourcing the Framework Foundation code, the EDK allows for the development, debugging, and testing of EFI and DXE drivers, Option ROMs, and pre-Boot applications.

C. Visual Studio

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft [5]. It is used to develop computer programs for Microsoft Windows, as well as web sites, web applications and web services. Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native code and managed code.

IV. GOP DRIVER

The EFI specification defined a UGA (Universal Graphic Adapter) protocol as a way to support device-independent graphics[3]. UEFI did not include UGA and replaced it with GOP (Graphics Output Protocol), with the explicit goal of removing VGA hardware dependencies. The two are similar.

Table 1 gives a quick comparison of GOP and video BIOS:

Table 1 Difference between Video BIOS and GOP

VIDEO BIOS	GOP
64 KB limit. 16-bit execution	No 64 KB limit. 32-bit protected mode
CSM is needed with UEFI system firmware.	No need for CSM. Speed optimized (fast boot).
The VBIOS works with both 32- and 64-bit architectures.	The UEFI pre-boot firmware architecture must match the GOP driver.

The GOP Driver is part of the UEFI boot time drivers responsible for bring up the display during bios boot. This driver enables logo display during bios boot time[4].

The GOP driver interacts with the PCI Driver which is responsible for enumerating the PCI devices such as graphics. For each controller detected on the PCI driver, it installs a PCI IO Protocol that get used by the underlying child driver for implementing its services.

The UEFI graphics driver (GOP Driver) is responsible for the Display unit functionality. It acts as a bus driver for the display subsystem and creates child devices for each of the output interfaces discovered by it[4]. It uses the PCI IO protocol for implementing the GOP protocol that is installed on each of the output child devices created by it.

Fig 2 although has both DSI and DP output interfaces are shown together as child devices, only one is active.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:6.887
Volume 5 Issue VII, July 2017- Available at www.ijraset.com

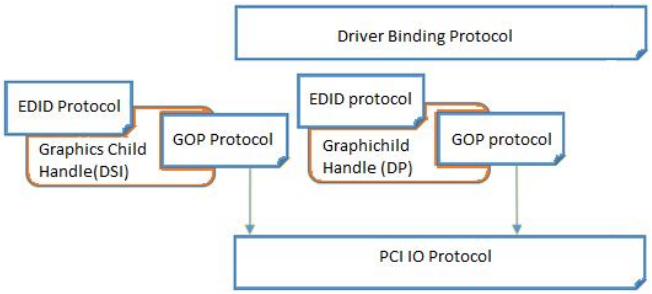


Fig 2 GOP driver implementation

A. The GOP Driver Implements the QueryMode(), SetMode(), Blt() Services of the Protocol.

In order to reduce boot time the GOP driver need not enumerate all the available modes during boot time. This enumeration can be deferred to the linux driver based on requirements for boot time. This driver also implements Device Path Protocol, Driver binding protocol interfaces. For each output interface the Graphics Output Protocol and Device Path Protocol needs to be implemented.

Depending on the need to support EDID panels, the driver will need to support the EDID Discovered protocol. The EDID data needs to be installed on the output interface.

The EDID active protocol is installed with the available EDID data on the output interface. The EDID override protocol is the alternate mechanism to supply EDID data if the EDID Discovered Protocol is unavailable. This should take care of support for EDID less panels.

1) *QueryMode():* This function is supposed to enumerate all the modes that can be supported for a given display panel. Typically this is the intersection of the modes that are supported by the Display controller and the given Panel. To cater to boot time optimizations we can consider enumerating only single mode while leaving the rest of the modes to be discovered by the driver.

2) *SetMode() :* This function is used the select a mode from the list enumerated by QueryMode.

3) *Blt() :* This function is used to copy contents from offscreen buffer to the display buffer.

V. METHODOLOGY

Most graphics controllers are PCI controllers, and this implies that UEFI Drivers for graphics controllers are typically PCI drivers. If a device is intended to be used as a graphics console output device while UEFI firmware is active, then a UEFI Driver must be implemented that produces the Graphics Output Protocol. The graphics controller must either directly support or be able to emulate the following operations:

A. Block transfer to fill a region of the frame buffer
B. Block transfer from system memory to region of frame buffer
C. Block transfer from region of frame buffer to system memory
D. Block transfer between two regions of the frame buffer
E. Query attached display devices for EDID information
F. Set the supported graphics modes that is intersection of modes that the graphics controller supports and the display device supports.

EDK II uses the services of a Graphics Output Protocol and bitmap fonts to produce the Simple Text Output Protocol. This means if a Graphics Output Protocol is produced by a UEFI Driver, then the frame buffer managed by that UEFI Driver can be used as a text console device without having to implement the Simple Text Output Protocol in the UEFI Driver for the graphics controller.

UEFI Drivers for graphics controllers are typically more sensitive to the EBC virtual machine interpreter overheads, so it is critical that the performance guidelines are followed for a UEFI Driver for a graphics controller that is compiled for EBC to have good performance. UEFI Drivers for graphics controllers typically follow the UEFI driver model. Some graphics controllers have a single output controller, and other may have multiple output controllers. In both cases, a child handle must be created for each output controller, which means UEFI Drivers for graphics controllers are always either Bus Drivers or Hybrid Drivers. They are never Device Drivers. UEFI Drivers for graphics controllers are chip-specific because of the requirement to initialize and manage the graphics device.

UEFI drivers that manage graphics controllers typically follow the UEFI Driver Model because the devices are typically on industry standard busses such as PCI. However, it is possible to implement UEFI drivers for graphics controllers that are not on industry standard busses. In these cases, a Root Bridge Driver implementation that produces a handle for each output controller in the driver entry point may be more appropriate than a UEFI Driver Model implementation. Graphics controllers that are connected to a single output device are the simplest type of UEFI graphics driver. They produce a single child \ handle and attach both Device Path and Graphics Output protocols onto that handle. They need a single data structure to manage the device. An example of a single output graphics driver stack is shown below.

VI. INTEGRATION

A. Platform Requirements

The platform firmware must meet the following requirements for GOP Driver integration:

1) Platform firmware must be compliant to UEFI 2.1 or later.
2) Platform must enumerate and initialize the graphics device
3) Platform must allocate sufficient graphics frame buffer memory required to support the native mode resolution of the integrated display.
4) The platform must produce the standard EFI_PCI_IO_PROTOCOL and as well as the EFI_DEVICE_PATH_PROTOCOL on the graphics device handle. Additionally, the platform must produce PLATFORM_GOP_POLICY_PROTOCOL.
5) The platform firmware must not launch the legacy Video BIOS.

The GOP Driver solution comprises the following files

Table 2 GOP driver files

File Name	Description	Format
GopDriver.efi	The GOP driver binary	Uncompressed PE/COFF image
Vbt.bin	Contains Video BIOS Table (VBT) data	Raw binary
Vbt.bsf	BMP script file. Required for modifying Vbt.bin using BMP tool.	Text

B. Integration Steps

1) Customize the VBT data file Vbt.bin as per platform requirements and the corresponding BSF file.
2) Integrate Vbt.bin and GopDriver.efi files into the platform firmware image. The process of accomplishing this step is determined by the platform implementer, specific to the platform firmware implementation.

C. GOP Driver Protocols

The GOP driver follows the UEFI Driver model and produces the following protocols:

1) *Efi_Driver_Binding_Protocol:* The GOP Driver produces the driving binding protocol on its image handle.
2) *efi_driver_binding_protocol.supported()*: In the Supported() call, the driver uses the device path protocol and PCI IO protocol to determine if it supports the controller handle. The RemainingDevicePath must be NULL, a valid ADR type or ‘end of device path’.
3) *Efi_Driver_Binding_Protocol.Start():* In the Start() call, the GOP driver initializes the graphics hardware, produces child handles and installs the required protocols on the child handles. If the child handle cannot be created, then EFI_UNSUPPORTED is returned. In this case the firmware should call Start with NULL as the RemainingDevicePath parameter value, to allow the driver start a default display[4]. If there are no displays connected, then **Start()** returns EFI_SUCCESS for NULL value.The following are the protocols installed on child controller handles
4) *Efi_Device_Path_Protocol:* All the child controller handles that are produced have EFI_DEVICE_PATH_PROTOCOL installed. The device path is produced by appending the ADR node to the device path of the graphics controller.

5) *Efi_Graphics_Output_Protocol:* The GOP driver produces a single instance of the protocol and is installed on the child controller, determined by the RemainingDevicePath.

Table 3 EFI_GRAPHICS_OUTPUT_PROTOCOL

RemainingDevicePath	Action
Valid ACPI Address	GOP installed on corresponding child handle
NULL	GOP installed on a default child handle determined by the GOP driver.

6) *Efi_Edid_Discovered_Protocol:* If the child controller is a single display, then EFI_EDID_DISCOVERED_PROTOCOL is installed. If the display is a local flat panel without EDID, then the GOP Driver constructs an EDID based on the timing details of this panel configured in VBT.
7) *Efi_Edid_Active_Protocol* If the child controller is the GOP device, then EFI_EDID_ACTIVE_PROTOCOL is the protocol installed. EFI_EDID_OVERRIDE_PROTOCOL is not used by GOP driver
8) *Efi_Component_Name2_Protocol:* The GOP Driver installs the component name protocol on its image handle. English is the only language supported.

VII. RESULTS

A. Building BIOS

The following command is used to build BIOS in SPI mode. This successfully builds the BIOS from the source code and creates spi.bin file which is a binary image file. The stitching process generates the stitch file (binary image) from all the source files and library files.

C:\UEFI>BuildImage.bat /c /x6 /nmrc /vp /spi SRVL Debug

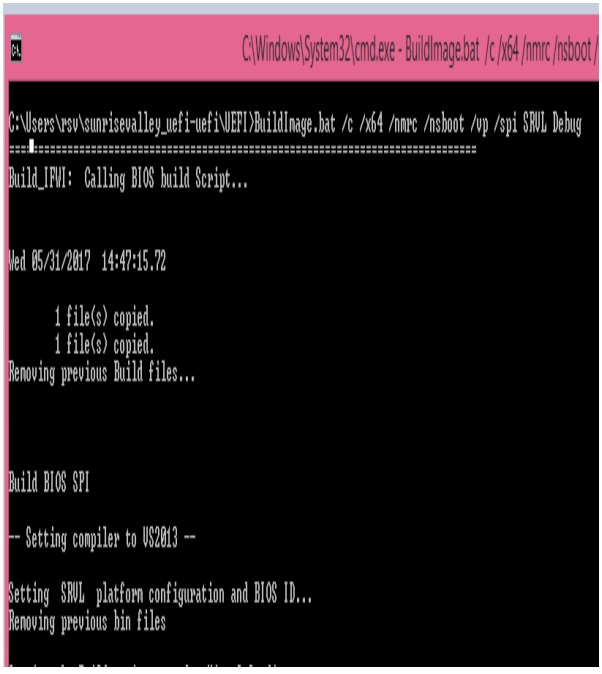


Fig 3 Command executed to build BIOS in SPI mode

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:6.887
Volume 5 Issue VII, July 2017- Available at www.ijraset.com

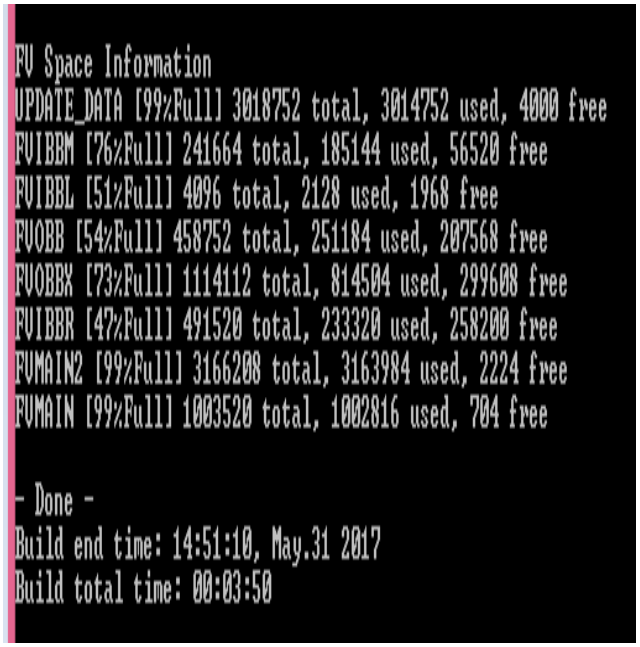


Fig 4 Command window showing "BIOS building is done"

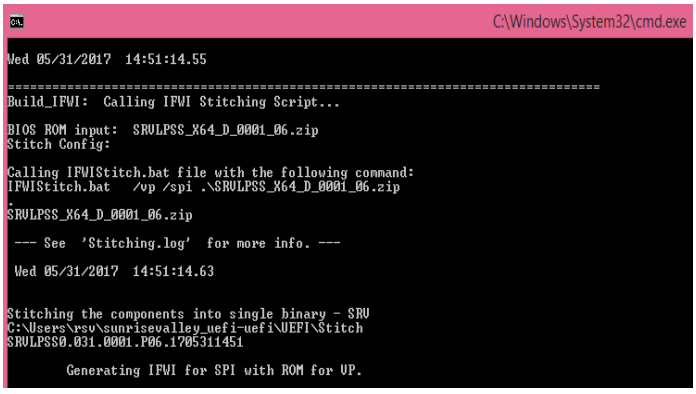


Fig 5 Command window showing that stitching process has begun

Fig 6 Command window showing that binary file is successfully generated

B. Simulating in Simics

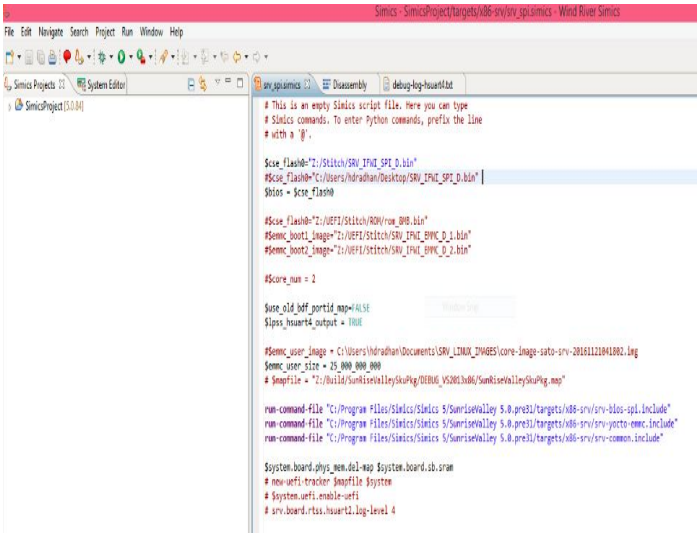


Fig 7 Simics window with debug script for SPI

The generated binary image is to be simulated for the platform. A script file is written for invoking all the drivers in SPI mode.

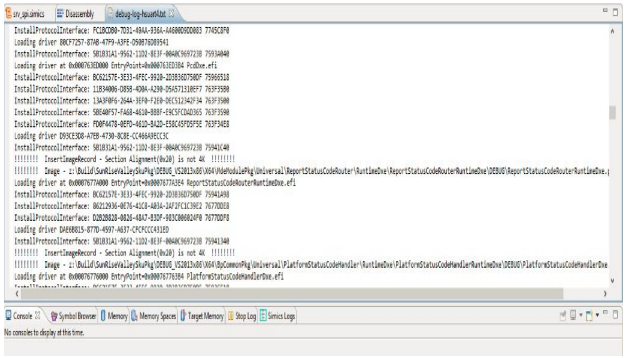


Fig 8 Simics debug window for simulating the binary file generated

The debug file is generated when run on the Simics. This has list of all the drivers invoked and protocols installed. The log file is hsuart.txt.

```
Loading driver at 0x000763B9000 EntryPoint=0x000763B93B4 Gop.efi
InstallProtocolInterface: BC62157E-3E33-4FEC-9920-2D3B36D750DF 75932318
GOP DRIVERInstallProtocolInterface: 9042A9DE-23DC-4A38-96FB-7ADED080516A 763BBA08
InstallProtocolInterface: BD8C1056-9F36-44EC-92A8-A6337F817986 0
InstallProtocolInterface: 763B9614-0000-0000-3096-3B7600000000 75932998
InstallProtocolInterface: A8838948-0000-4800-85C0-79184C8B4368 75932D98
InstallProtocolInterface: 75932998-0000-0000-D0FD-437700000000 7745C780
InstallProtocolInterface: 75932D98-0000-0000-0000-000000000000 7745C780
```

Fig 9 Invoking of GOP driver

The above diagram shows the invoking of GOP driver. The Gop.efi (binary image file of GOP source code) is invoked and various protocols like

1) Efi_driving_binding_protocol
2) Efi_device_path_protocol
3) Efi_graphics_output_protocol
4) Efi_edid_discovered_protocol
5) Efi_edid_active_protocol
6) Efi_component_name_protocol

It can be seen that the user readable name GOP DRIVER is displayed as EFI_COMPONENT_NAME_PROTOCOL is getting installed. All the above mentioned protocols have been listed out in log file by mentioning their respective GUIDs. After successful simulation, the binary image file is dumped on to the platform board.

VIII. CONCLUSION AND FUTURE WORK

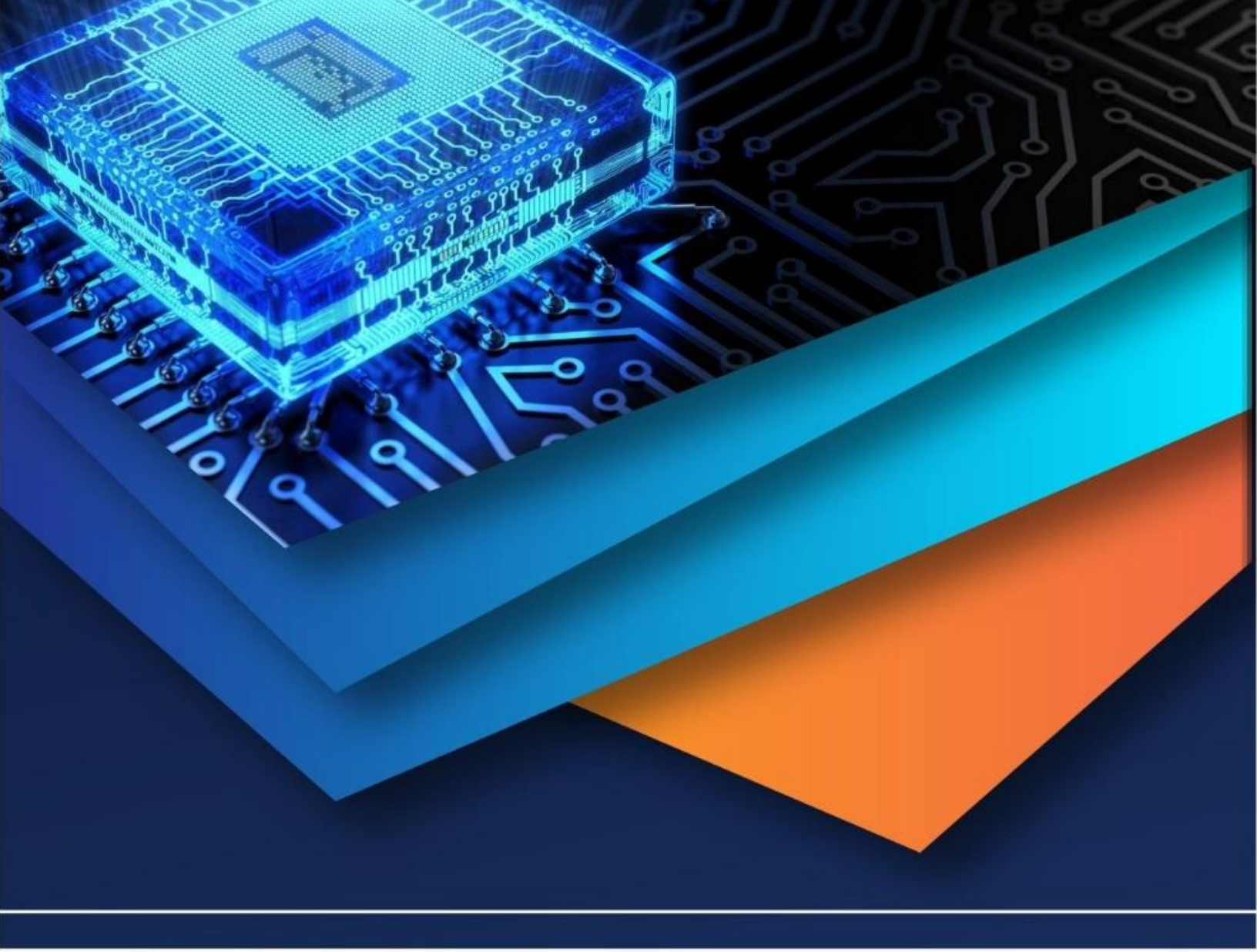
The paper involves understanding of BIOS, booting process, UEFI, architectural flow, UEFI drivers, driver models and GOP driver. Simulation tools like Simics and Visual studio are used to build BIOS, develop and debug for Intel's IoT platform.

Having knowledge of all the above mentioned concepts and writing and invoking a simple driver, the Graphics Output Protocol (GOP) is written for Intel's IoT platform which is responsible for graphics display during booting process until the OS gains the control.

As a part of further improvisation, the GOP driver can be further developed for providing options like brightness, contrast, color pixels etc. to the user during the booting time through the interface.

REFERENCES

[1] Muhammad Irfan Afzal Butt, "BIOS integrity, an advanced persistent threat", IEEE Information Assurance and Cyber Security (CIACS), 2014 Conference on,2014.
[2] Rahul Khanna, Fadi Zuhayri, Murugasamy Nachimuthu, "Unified extensible firmware interface: An innovative approach to DRAM power control", IEEE Energy Aware Computing (ICEAC), 2014 International Conference on 2014.
[3] Rahul Khanna, Fadi Zuhayri, Christian Le, "Unified extensible firmware interface: An innovative infrastructure for power/thermal autonomics", IEEE EnergyAware Computing (ICEAC), 2014 International Conference on 2014
[4] Vincent Zimmer, Michael Krau, "Establishing the root of trust", Unified Extensible Firmware Interface Forum, August 201
[5] Sven Amann, Sebastian Proksch, Sarah Nadi, "A Study of Visual Studio Usage in Practice", Software Analysis, Evolution, and Reengineering (SANER), 2016 IEEE 23rd International Conference on 2016,2016, DOI: 10.1109/SANER.2016.39



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY