Generic IP independent BIOS Signing and Parsing

Gahan Saraiya

Institute of Technology
Nirma University

March 15, 2020

Outline

1. About the Project
2. Motivation
3. BIOS: Basics
4. Requirement Specification
5. My Contribution
6. Future Scope
7. Phase 1: Study of existing architecture for hotspot

About the Project

In general to generate BIOS image (*.rom file), compilation of XYZ.c (source code) has to be done, this compilation not only involves compilation of DXE driver, PEI driver, EFI Application but also includes pre-processing checks, compression of raw files which takes huge amount of time depending on the system configuration. Implementation of this project aids in reduction of this compilation time.

Motivation: Stakeholders

- BIOS development Team
- Automation Team
- Validation Team
- Other Development Team

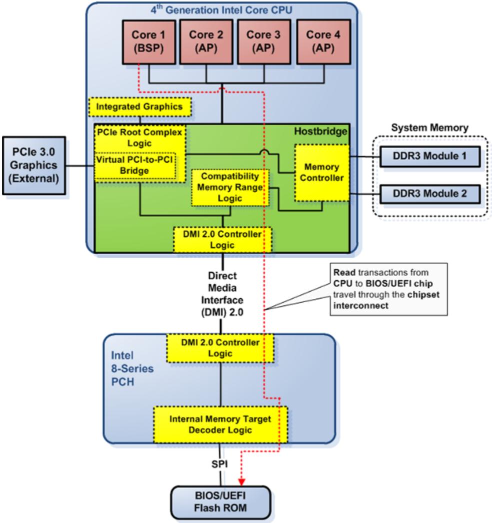
Motivation: Issues/Challenges to the industry (Towards my contribution)

- BIOS image generation: Compilation of whole source code
- More Time complexity: Compilation of source code to generate BIOS image

BIOS: Basics

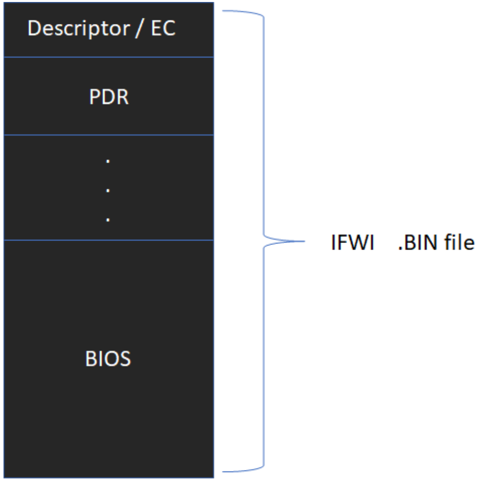
- Set of Software Routines
  - Initialize and test hardware on start
  - Provides the OS with a generic hardware abstraction
- the BIOS must do its job before your computer can load its operating system and applications

BIOS: Architecture



Read transactions from CPU to BIOS/UEFI chip travel through the chipset interconnect

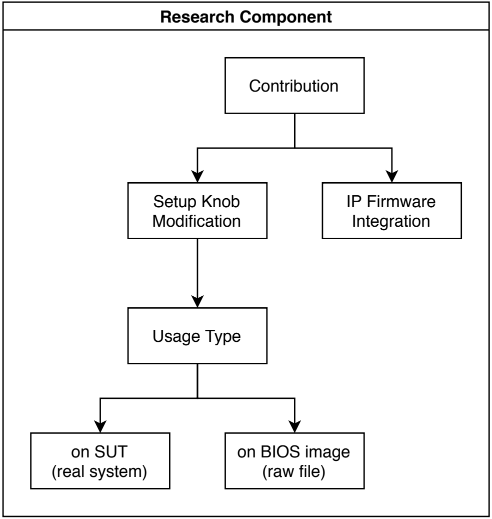
BIOS: Firmware Image



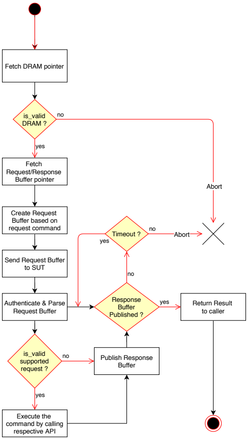
Requirement Specification

- Visual Studio C/C++ IDE
- Python 3
- Visual Studio Code
- Memory access Interfaces

My Contribution towards issues/challenges

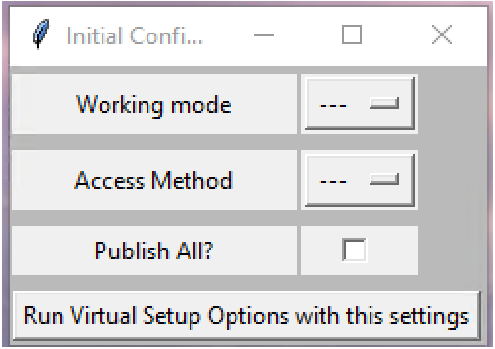


Setup Knobs Modification: Process Flow I


Fetch DRAM pointer
is_valid DRAM ?
no
yes
Fetch Request/Response Buffer pointer
Abort
Create Request Buffer based on request command
Timeout ?
no
Abort
yes
Send Request Buffer to SUT
no
Authenticate & Parse Request Buffer
Response Buffer Published ?
yes
Return Result to caller
is_valid supported request ?
no
Publish Response Buffer
yes
Execute the command by calling respective API

Setup Knobs Modification Flow on SUT

Setup Knobs Modification: Implementation Snaps I



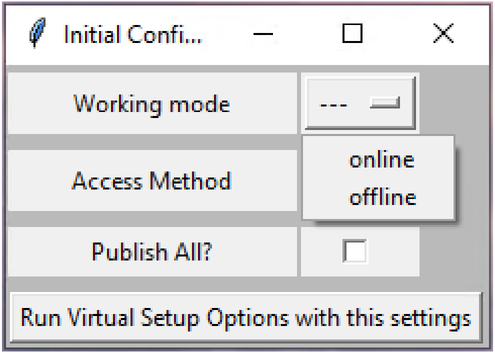
Working mode ---

Access Method ---

Publish All? □

Menu to Select initial configuration for work

Setup Knobs Modification: Implementation Snaps II



Initial Confi...

Working mode ---

online

offline

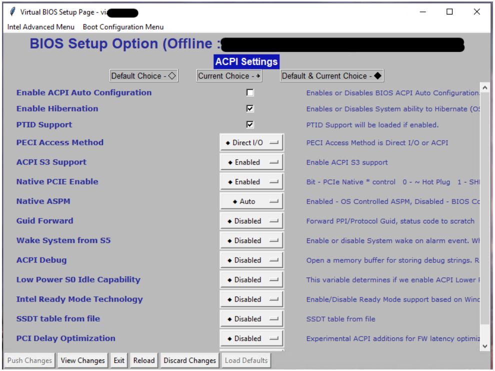
Access Method

Publish All? □

Run Virtual Setup Options with this settings

Available work mode for the system: Online and Offline

Setup Knobs Modification: Implementation Snaps III



Virtual BIOS Setup Page - vi[redacted]

BIOS Setup Option (Offline :[redacted]

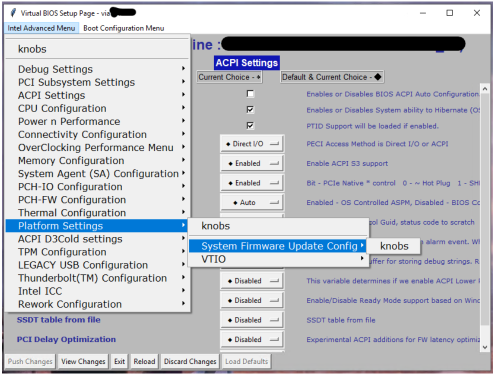
ACPI Settings

Default Choice - ◇ Current Choice - • Default & Current Choice - ◆

Enable ACPI Auto Configuration	☐	Enables or Disables BIOS ACPI Auto Configuration
Enable Hibernation	☑	Enables or Disables System ability to Hibernate (OS
PTID Support	☑	PTID Support will be loaded if enabled.
PECI Access Method	◆ Direct I/O	PECI Access Method is Direct I/O or ACPI
ACPI S3 Support	◆ Enabled	Enable ACPI S3 support
Native PCIE Enable	◆ Enabled	Bit - PCIe Native * control 0 - ~ Hot Plug 1 - SHI
Native ASPM	◆ Auto	Enabled - OS Controlled ASPM, Disabled - BIOS Co
Guid Forward	◆ Disabled	Forward PPI/Protocol Guid, status code to scratch
Wake System from S5	◆ Disabled	Enable or disable System wake on alarm event. Wh
ACPI Debug	◆ Disabled	Open a memory buffer for storing debug strings. R
Low Power S0 Idle Capability	◆ Disabled	This variable determines if we enable ACPI Lower P
Intel Ready Mode Technology	◆ Disabled	Enable/Disable Ready Mode support based on Winc
SSDT table from file	◆ Disabled	SSDT table from file
PCI Delay Optimization	◆ Disabled	Experimental ACPI additions for FW latency optimiz

Setup Options listed under ACPI Configurations

Setup Knobs Modification: Implementation Snaps IV



ine : [redacted]

ACPI Settings

Current Choice - ◆ Default & Current Choice - ◆

□ Enables or Disables BIOS ACPI Auto Configuration.

☑ Enables or Disables System ability to Hibernate (OS

☑ PTID Support will be loaded if enabled.

◆ Direct I/O PECI Access Method is Direct I/O or ACPI

◆ Enabled Enable ACPI S3 support

◆ Enabled Bit - PCIe Native * control 0 - ~ Hot Plug 1 - SHI

◆ Auto Enabled - OS Controlled ASPM, Disabled - BIOS Co

ol Guid, status code to scratch

alarm event. Wh

uffer for storing debug strings. R

◆ Disabled This variable determines if we enable ACPI Lower P

◆ Disabled Enable/Disable Ready Mode support based on Wind

SSDT table from file ◆ Disabled SSDT table from file

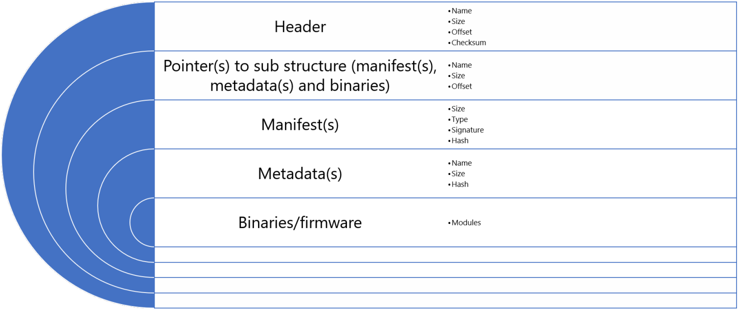
PCI Delay Optimization ◆ Disabled Experimental ACPI additions for FW latency optimiz

Navigating through BIOS setup page

Setup Knobs Modification: Outcome

- Cross platform usage
- API as a driver in BIOS Firmware
- Generic solution for usage types - on **SUT**, on **BIOS image**
- Information parsing and simulation
- Realtime sync for simulation changes
- Seamless Integration

IP firmware Integration: Structure of Module



Header	•Name •Size •Offset •Checksum
Pointer(s) to sub structure (manifest(s), metadata(s) and binaries)	•Name •Size •Offset
Manifest(s)	•Size •Type •Signature •Hash
Metadata(s)	•Name •Size •Hash
Binaries/firmware	•Modules

Proposed Structure for firmware signing

IP firmware Integration: Outcome

- Removal of IP dependency during firmware loading
- IP Subsystem :
  - Loader and Verifier
  - IP is always consumer
- Signature verification using SHA hash algorithm
- Seamless Integration of any other hash algorithm for verification
- Hardware based and Software based verification support
- Prevent common security threats
- Allow easier OEM adoption and modification based on the respective design
- Reusability/Portability of design across many IPs
- Generic design which supports any new IP integration

Future Scope

- Study of existing hotspots for automation
- Analyzing and gathering requirements of automation to hotspot
- Implementing and managing platform to keep up-to-date the user base of the framework

Phase 1: Study of existing architecture for hotspot

Phase 2: Analysis and gathering detailed information of the problem

Requirement Specification

Why JSON?

Implementation of Parsing

Flow diagram of BIOS Image parsing

Implementation of Parsing: Architecture of BIOS Image

Architecture

Implementation of Parsing: Work Flow

offset parsing next offset... 'algorithm'

Outcome

- Human Readable interpretation of BIOS Image
- GUIDs Lookup
- Verification of existence of driver/application by GUID
- Storing the image file system content by GUID
- Summarizing changes of two BIOS image

Thank You