Enabling RAS on Xeon OCP Platforms Using Intel FSP and Coreboot
Enabling RAS on Xeon OCP Platforms Using Intel FSP and Coreboot

Ruth Li, Software Architect, Intel
Agenda

- This talk is part of Intel® FSP exploration to support diversified boot frameworks along with Open system firmware initiative
- The topic depicted technical solution to enable Runtime RAS service under Coreboot framework via FSP API interface.

- Open System Firmware and FSP Overview
- Problem in Silicon specific SMM/Runtime modules working with FSP
- Solution in Runtime RAS support with FSP
- Demo of ‘RAS Enabling in FSP with Coreboot’
Open System Firmware and FSP Overview

OPEN SOURCE EDKII

EDKII:
https://github.com/tianocore/edk2

EDKII Minimum Platform Spec:
https://edk2-docs.gitbook.io/edk-ii-minimum-platform-specification

EDKII MinPlatform/OpenBoard:
https://github.com/tianocore/edk2-platforms/tree/master/Platform/Intel/MinPlatformPk

REDISTRIBUTED Firmware Binary

• FSP(Firmware Support Package)
  ✓ Latest spec: FSP 2.3
  ✓ Supporting UEFI/Coreboot/Slim Bootloader
• Microcode patch, Security ACM
• Mgmt.: SPS FW, Ignition FW

OPEN POSSIBILITIES.
In **UEFI Flavor**, when working with Intel® FSP
- CPU, Mem, Chipset initialization are executed in PEI phase, enclosed in Intel® FSP
- Silicon Specific SMM/Runtime modules are implemented in Close-source, Dispatched in DXE phase

In **non-UEFI flavor**, if integrating with bootloaders (e.g., coreboot, Slimboot):
- CPU, Mem, Chipset initialization are supported with Intel® FSP Binary
- UEFI DXE/SMM module CAN’T be Seamlessly integrated.

**Problem/Issue**
- No Unified Redistributable Silicon SMM/Runtime modules to support diversified bootloader for Open System Firmware Solution
Reliability, Availability and Serviceability (RAS)

RAS Play important Role in modern Server/Cloud Service
- Unexpected Downtime is Disruptive and Expensive - $5,000 for one-minute outage!!!
- Minimum Reliability Requirement is Rising – 4-nines, and 5-nines

RAS Stack Topology
- Boot-time: Initialize RAS driver and install Runtime/SMM RAS handler
- Run-time: When HW error happened
  - Firmware: RAS SMM handler process the error and Failure Telemetry
  - OS/Hypervisor: Further Error handling and Read error log etc.

Gap with Intel® FSP Support
- No Runtime/SMM Support

Solution to enable RAS within Intel®
FSP binary
Solution to enable Standalone MM/RAS with Intel® FSP

Solution Components

- **RAS modules**: Convert DXE/SMM RAS driver into PEIM and Standalone MM
- **MM Core/Framework**: support launching Standalone MM modules
- **Late MM loader**: dispatch OEM MM modules in later boot phase
- **Runtime MM update**: Introduce Runtime Update Framework to receive new MM module entered in OS time
- **Encapsulate Standalone MM Core, RAS module (PEIM, Standalone), MM Late loader, Runtime update framework into Intel® FSP binary**

*During Boot-time: Initialize MM core in early Silicon-Init phase, dispatch RAS driver from FSP*
**RAS Demo Flow**

**Demo Configuration**
- Demo use cases: RAS Address Translation Handler

**Flash Image include**
- FSP Binary (with API support) + Coreboot as boot loader
- Linux Boot used as pre-OS payload

**Ubuntu as Target Production OS**

---

**Demo Flow:** System Boot successfully, and doing checkpoint in Boot time and Runtime stage

**During Boot Time**
1. Confirm ACPI Table exposed as expectation in Coreboot
2. Confirm ACPI Table exposed as expectation in Linuxboot

**Runtime Stage**
3. Confirm ACPI Table is exposed correctly
   - Triger Software SMI, Address Translation Handler was triggered as expectation

---

**Intel® FSP**

1. Add Required PEIM drivers
2. Add Standalone MM Core
3. Add Required Standalone MM drivers for RAS

**HOB**

1. Parse HOB and Publish ACPI for RAS Address Translation Handler

**Core-boot**

**LinuxBoot (u-root + bzImage)**

1. ACPI Table (for RAS) exposed.
2. Address Translation Handler can be triggered in runtime via SMI

**Linux OS**
By FSP: 1). SMI handler is installed. 2). HOB is Exposed HOB to boot loader

In Coreboot: Received from HOB, and build ACPI record

In Linuxboot: Acpi table is exposed

In Ubuntu: SMI handler is triggered, ACPI table is installed
Call to Action

Get involved into Open Compute Project

Open System Firmware: https://www.opencompute.org/projects/open-system-firmware

Engage with Intel on FSP and Open Firmware Development


Intel EDKII: https://github.com/tianocore/edk2
Thank you!