Enabling Runtime RAS for Xeon Platforms using Open System Firmware

Daocheng “Amos” Bu, Firmware Engineer (Intel)
Anjaneya “Reddy” Chagam, Sr. Principal Engineer (Intel)
Notices & Disclaimers

Intel technologies may require enabled hardware, software or service activation. Your costs and results may vary.

No product or component can be absolutely secure.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. For more complete information about performance and benchmark results, visit http://www.intel.com/benchmarks.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit http://www.intel.com/benchmarks.

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Cost reduction scenarios described are intended as examples of how a given Intel-based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.
Agenda

- Background
- FSP-SMM API
- FSP-SMM binary layout
- SMM traditional Mode VS. Standalone Mode
- FSP-SMM flow
- OSF(coreboot)→FSP-SMM
- Traditional SMM transition to MM
- Converged SMM Solution (Future)
Background

- Current FSP2.3 does not include SMM related function, bootloaders own SMM foundation and feature itself.
- However, for some SMM functions like server RAS runtime features, it is not easy for bootloader (like coreboot/SlimBoot) to support it.
- It will be a solution for non-EDKII bootloader with FSP-SMM to support SMM foundation and features.
FSP-SMM API

- The Intel® FSP binary follows the UEFI Platform Initialization Firmware Volume Specification format.
- Split into 4 separate FVs
- FSP-SMM: SMM initialization phase (FSP owns SMRAM)
  - Primary purpose of this phase is to provide a collection of silicon SMI handlers that provide value-add services that a bootloader can use.
  - `FspSmmlnit()`: Initialize SMM Foundation, dispatch all Standalone MM drivers, build SMM environment, install all SMI handlers.
FSP-SMM binary layout

FSP TOP
- FSP-T
  - Temp RAM Phase Data
  - FSP_INFO_HEADER - T
- FSP-M
  - Memory Init Phase Data
  - FSP_INFO_HEADER - M
- FSP-S
  - Silicon Init Phase Data
  - FSP_INFO_HEADER - S
  - FSP-I
  - SMM Init Phase Data
  - FSP_INFO_HEADER - SMM

FSP BASE

SRAM / IBB-L
- FSP-T
  - Temp RAM Phase Data
  - FSP_INFO_HEADER - T

CAR / IBB-M
- FSP-M
  - Memory Init Phase Data
  - FSP_INFO_HEADER - M

Memory / IBB-R
- FSP-S
  - Silicon Init Phase Data
  - FSP_INFO_HEADER - S

SMRAM / IBB-R
- FSP-I
  - SMM Init Phase Data
  - FSP_INFO_HEADER - S
SMM Traditional Mode

- DXE Core
- Security Arch Dxe
- Traditional SmmIpl
- FV Protocol
- Security Protocol
- LOAD
- SmmFoundation
- Traditional SmmCpu
- SmmAccess Dxe
- MpService Dxe
- SMM_ACCESS Protocol
- MP_SERVICE Protocol
Standalone Mode

Standalone: SMM will not run DXE code.
**FSP-SMM Boot Flow**

1. **Bootloader**
   - Call (FSP-SMM base + FspSMMInitEntryOffset)

2. **Standalone SmmIpl Entry**
   - Open SMRAM
   - Load SmmFoundation from Flash to SMRAM
   - Call SMM Foundation
   - Close SMRAM

3. **Standalone SmmFoundation Entry**
   - Initialize SMM Services
   - Dispatch Drivers In StandaloneSmm FV
   - Return Back to IPL

4. **Standalone SmmCpu Entry**
   - Multiprocessor Information Collection
   - SMBASE Calculation
   - SMM Rebase
   - SMM Setting
   - Now SMI Enabled

**FSP-SMM**
## FSP-SMM Modules changes

<table>
<thead>
<tr>
<th>FV</th>
<th>Status</th>
<th>Build target</th>
<th>Driver name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSP-S FV</td>
<td>Existed</td>
<td>IA32</td>
<td>SmmAccessPei.Inf</td>
<td>Provide access SMM Ram ppi</td>
</tr>
<tr>
<td>FSP-S FV</td>
<td>Existed</td>
<td>IA32</td>
<td>CpuMpPei.inf</td>
<td>Provide MP service ppi</td>
</tr>
<tr>
<td>FSP-SMM FV</td>
<td>New</td>
<td>IA32</td>
<td>SmmControlPei.inf</td>
<td>Provide smi trigger/clear ppi</td>
</tr>
<tr>
<td>FSP-SMM FV</td>
<td>New</td>
<td>IA32</td>
<td>CpuMpInfoPei.inf</td>
<td>Provide MP info hob</td>
</tr>
<tr>
<td>FSP-SMM FV</td>
<td>New</td>
<td>IA32</td>
<td>StandaloneMmplpiPei.inf</td>
<td>Load MMCore to SMM Ram</td>
</tr>
<tr>
<td>FSP-SMM FV</td>
<td>New</td>
<td>X64</td>
<td>StandaloneMmCore.inf</td>
<td>Dispatch all MM driver and register all handlers</td>
</tr>
<tr>
<td>FSP-SMM FV</td>
<td>New</td>
<td>X64</td>
<td>PISmmCpuStandaloneSmm.inf</td>
<td>Build CPU SMM enviroment</td>
</tr>
<tr>
<td>FSP-SMM FV</td>
<td>New</td>
<td>X64</td>
<td>CpuIo2StandaloneMm.inf</td>
<td>Provide CPU io/memory access protocol</td>
</tr>
<tr>
<td>FSP-SMM FV</td>
<td>New</td>
<td>X64</td>
<td>PchSmiDispatcherServer.inf</td>
<td>Install sw smi handler for verify</td>
</tr>
</tbody>
</table>
coreboot FSP-SMM (boot) integration

1. FspSiliconInit (w/ SmmInitEn=1)
2. Silicon Init
3. FspSmmInit
4. Report FV
5. Execute RAS PEIMs
   Setup MM foundation
   Execute RAS MM drivers
6. IEH init
7. Hobs w/ RAS ACPI data
8. Install ACPI, Pass E820, RSDP etc
9. Driver consumes the RAS ACPI
FSP-SMM (runtime) Execution

1. Inject PCIE UCE

2. SMI

3. IEH error handler
   *WheaLog handler to update HEST table*

4. NMI

5. Invoke GHES NMI handler
   *Dump HEST and panic*

Hardware error from APEI Generic Hardware Error Source: 1
- Event severity: fatal
- Error 0, type: fatal
- `section_type` = PCIe error
- `port_type` = 4, root port
- `device_id` = 0000:63:02.0 slot: 7
- `vendor_id` = 0x8086, `device_id` = 0x0db0
- `aer_uncor_status` = 0x00040000, `aer_uncor_mask` = 0x00180020
- `aer_uncor_severity` = 0x00463010
- TLP header: 40000000 0000000f fdaff040 12345678

---

**Runtime**

**Intel Confidential**
Traditional SMM transition to Stanalone MM

- **Traditional SMM foundation**
  - DXE_SMM Driver
    - DXE Init Code
    - RAS Functionality SMM Init Code
    - RAS SMM Runtime Code

- **Hybrid SMM foundation**
  - DXE_SMM Driver
    - DXE Init Code
    - RAS Functionality SMM Init Code

- **Standalone MM Driver**
  - RAS MM Runtime Code

- **Coreboot Bootloader**
  - FspSmmInit

**Steps:**
1. FSP-SMM
2. Standalone MM foundation
3. RAS Functionality MM Init Code
4. RAS MM Runtime Code
5. Further separate For FSP-SMM

Separate two parts for standalone MM drivers

Reuse Directly
Thanks & QA

• Xeon Server coreboot credits:
  - Meta, Byte Dance, Google, Amazon, Intel
  - 9Elements, SysPro Consulting
  - Wiwynn, Quanta, Inspur

• Reference link:
  - https://github.com/universalscalablefirmware
  - 644852_2.3_Firmware-Support-Package-External-Architecture-Specification
  - FSP 2.4 EAS is coming..