OPEN. FOR BUSINESS.
Optimized UEFI Implementation on Intel® Xeon Based OCP Platform

Sarathy Jayakumar, Principal Engineer, Intel Corp
Mohan J. Kumar, Fellow, Intel Corp
UEFI-based Open Firmware
(for Intel® Xeon® Processor-based Platforms)

Platform Firmware Interface (ACPI, UEFI)

Open source Platform Package(s)

Intel Silicon Init

Platform interface tables to support OS boot
https://uefi.org

EDKII – existing upstream/open source core at
https://github.com/tianocore/edk2

UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firmware
UEFI-based Open Firm
Minimum Platform overview

**EDKII**

**GitHub**

**Intel® Firmware Support Package (FSP) Binary**

**Si API**

**Platform API**

**EDK2 API**

**Server Silicon Binaries**

**Server Board Package**

**Client Board Package**

**Ultra Mobile Board Package**

**Developer Focused**

**Easier to share and secure code**

**Consistent** boot flows and interfaces

**Approachable** across the ecosystem

**Scalable** from pre-silicon to derivatives

---

https://github.com/tianocore/edk2-platforms/tree/devel-MinPlatform/Platform/Intel/MinPlatformPkg

https://github.com/tianocore/edk2-non-os/tree/devel-MinPlatform/Silicon/Intel/PurleySiliconBinPkg

https://github.com/tianocore/edk2
1. Out of Phase processor check-in at Reset Vector
2. Multi-Socket parallel MRC
3. Early and Late I/O Init (Pre and Post Memory)
4. Platform Abstraction (PCIe HP Slots, Retimers etc)
5. Runtime Handling
Boot Time Optimization
Boot Time Optimization

ACM
- Patch load
- Basic CPU Init

SEC
- uCode
- Topology Discovery
- Routing Setup

UPI
- Topology Discovery
- Routing Setup

NEM
- DIMM Discovery
- Mem Channel Training
- ECC Init and Memory Test
- Address Map

MRC

NEM Tear Down

MP Init

Platform Init
- PCIe Enum
- And Resource Allocation
- Publish
- Mem Map,
- ACPI,
- SMBIOS

Option ROM

Boot Device Selection

OS Handoff

Boot Time Optimization Touch points

OPEN. FOR BUSINESS
Boot optimized Purley MinPlatform

Mt. Olympus Platform with 32GB of memory, SKX Processor
OS handoff to OS Load – 13sec
Boot Optimizations...

- BMC Communications
- Optimized flash settings
- Memory ECC init on cold boot only
- BDS Fast boot path

Please visit Azure Booth for more info
Proposal for reducing SMM Usage
Problem Summary

- Broadcast SMM model has served the industry well for many generations, but running into some fundamental problems
  - **Perf/QoS degradation**
    - SMM latency increases with core count.
    - Several RAS features require platform FW to monitor *every* correctable error.
  - **Firmware Complexity**
    - SMM was never designed to handle so many asynchronous events in many-core environment.
    - No concept of interrupt priority, edge/level or reentrancy
    - Discovering more race conditions as the number of RAS events and new features are added.
    - SMM handler code has crossed complexity threshold, SW can’t take advantage of HW enhancements.
  - **Security concerns**
    - SMM code is more privileged than VMM/OS and an attractive target for attackers.
OS View of SMM

- SMM Elimination strategy should not affect OS to 'Platform Abstraction' Interface.
- SW SMI Triggers are Transparent to OS
- HW SMI Triggers are Transparent to OS
Categories of SMM Handler

**Current Model**

**CATEGORY 1:**
- SW SMI that don't require SMM privileges
- (e.g. Addr xlation, NVDIMM DSMs etc., List In Back-up slide)

**CATEGORY 2:**
- SW SMI that require SMM privileges

**CATEGORY 3:**
- HW SMI and RAS Handlers that don't require priviledges

**CATEGORY 4:**
- HW SMI and RAS Handlers that require priviledges

**Proposed Model**

- ASL+PRM
- Capsule Update
- OOB / PRM
- OOB

<table>
<thead>
<tr>
<th>Category</th>
<th>Elimination Proposal</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ASL + PRM</td>
<td>e.g. Addr Translation, eLog, NVDIMM DSM etc.</td>
</tr>
<tr>
<td>2</td>
<td>Capsule Update</td>
<td>BIOS and Variable Update are the only two usages</td>
</tr>
<tr>
<td>3 and 4</td>
<td>BMC</td>
<td>Error Handling. Server implementation might choose to use BMC assistance.</td>
</tr>
</tbody>
</table>

ASL – ACPI Source Language
PRM – Protected Runtime Mechanism (Explained in Subsequent Slides)
OOB – Out of Band (e.g. BMC)
Category 1: Background on ASL and SMM Usage

• Platform Event Handling in runtime requires intimate platform knowledge
• ACPI provides mechanism for runtime platform abstraction (ASL)
• ASL is very primitive interpreted code, and very limited in scope
  • ASL is architecture neutral, hence can’t do architecture specific tasks
  • To overcome the ASL limitations, SMI is invoked to provide native code execution
  • ASL is slow, hard to develop, maintain and debug
• PRM (Protected Runtime Mechanism) provides a secure mechanism for ASL extension
  • Makes usage of ASL viable, eliminating the need to drop into SMI
  • Platform Runtime code will be carried as part of the BIOS image (tied to the platform)
Why not Driver Based Model

1. OSVs do not want to carry platform specific knowledge in the driver
2. Need intimate platform and silicon knowledge to carry out certain tasks (such as Address Translation for RAS)
3. These vary between platform generations, and even between implementations in the same generation

Examples:

1. **PSHED Plug-in**
   - Not a viable deployment model and hence ACPI abstraction was used which drops into SMI for handling complex tasks

2. **Address Translation**
   - Originally pushed to EDAC drivers
   - OSVs are pushing it back into ACPI to keep the driver generic
   - The ACPI method drops into SMI to handle complex algorithms.
What is Protected Runtime Mechanism

- PRM is a mechanism to invoke native code from ACPI context
- ASL as the landing point for SW or HW based runtime events
- ASL will invoke PRM if required (think of this as ASL Assist)

- **PRM is not a new capability**
- **It is piecing together of existing capabilities to create a mechanism.**
SMM vs PRM

Using SMM
- OS Level Software / Driver
  - ASL Methods
  - ACPI Tables
- SW SMI
  - Broadcast - Stalling all threads
  - Blackbox - Hidden from OS
  - Privileged - Unfretted access to hardware resources and memory
- HW SMI
  - h/w access
- Platform Hardware

Using PRM
- OS Level Software / Driver
  - ASL Methods
  - ACPI Tables
- ASL
  - SCI
  - Compute Offload
- Oregion
  - h/w access
- Platform Hardware
- PRM
  - Readable and Auditable

COMPLETELY wall-off from Hardware, using page tables
Readable and Auditable

OPEN. FOR BUSINESS
## SMM Vs PRM …Continued

<table>
<thead>
<tr>
<th>Feature</th>
<th>SMI</th>
<th>PRM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global/Local</td>
<td>SMI is global. All the threads are stalled.</td>
<td>Behaves like a device interrupt (SCI). Only one thread executes the handler</td>
</tr>
<tr>
<td>Privilege</td>
<td>SMM has unlimited access to system resources</td>
<td>PRM is a contained environment (Ring-0)</td>
</tr>
<tr>
<td>Resource conflict</td>
<td>Non Atomic IO (Index/Data) access</td>
<td>ASL Based access to hardware resources. OS aware.</td>
</tr>
<tr>
<td>Locked registers</td>
<td>SMM has access to all the locked registers</td>
<td>PRM can’t modify locked registers.</td>
</tr>
<tr>
<td>VMM Security</td>
<td>VMM pass-through of SW SMI leads to attack vectors</td>
<td>PRM prevents this by restricting access to memory</td>
</tr>
<tr>
<td>Blocking</td>
<td>SMM is blocking. No other interrupts can make progress</td>
<td>PRM runs in a kernel driver context. So, can be interrupted by high priority interrupts</td>
</tr>
<tr>
<td>Visibility</td>
<td>Black Box</td>
<td>Visible to OS (just link another kernel mode driver):</td>
</tr>
</tbody>
</table>
Category 2: Authenticated Variables and BIOS Update

• Authenticated Variables
  - Consumed by BIOS during boot (e.g. SMT enable/disable)
  - Use capsule update style for updating the authenticated variables
  - The RO region for Authenticated Variables gets locked out at End of DXE
  - Staged by the OS, updated on reset and BIOS acts on the settings

• BIOS Update
  - Capsule based update

Server Class systems might choose to use BMC/OOB based Variable and BIOS update based on Redfish standard.
Configure CPU so that correctable errors assert ERR0 pin instead of SMI.

ERR0 from all sockets connected to BMC GPI, configured to interrupt BMC FW.
Feedback?

- Let’s together accelerate OSF Development
- Increase adoption of MinPlatform
- Get Involved
- Feedback? How can we help
OPEN. FOR BUSINESS.