AMD Common Silicon Firmware Module

Satish Rai, Changhwa Lee

September 27, 2007

Purpose of this presentation

To help understand how the firmware for silicon components (Processor, Northbridge and Southbridge) delivered to support PI and UEFI

Provide a mechanism to support Independent BIOS Vendors with a better interoperability for future silicon components

Why common firmware is important

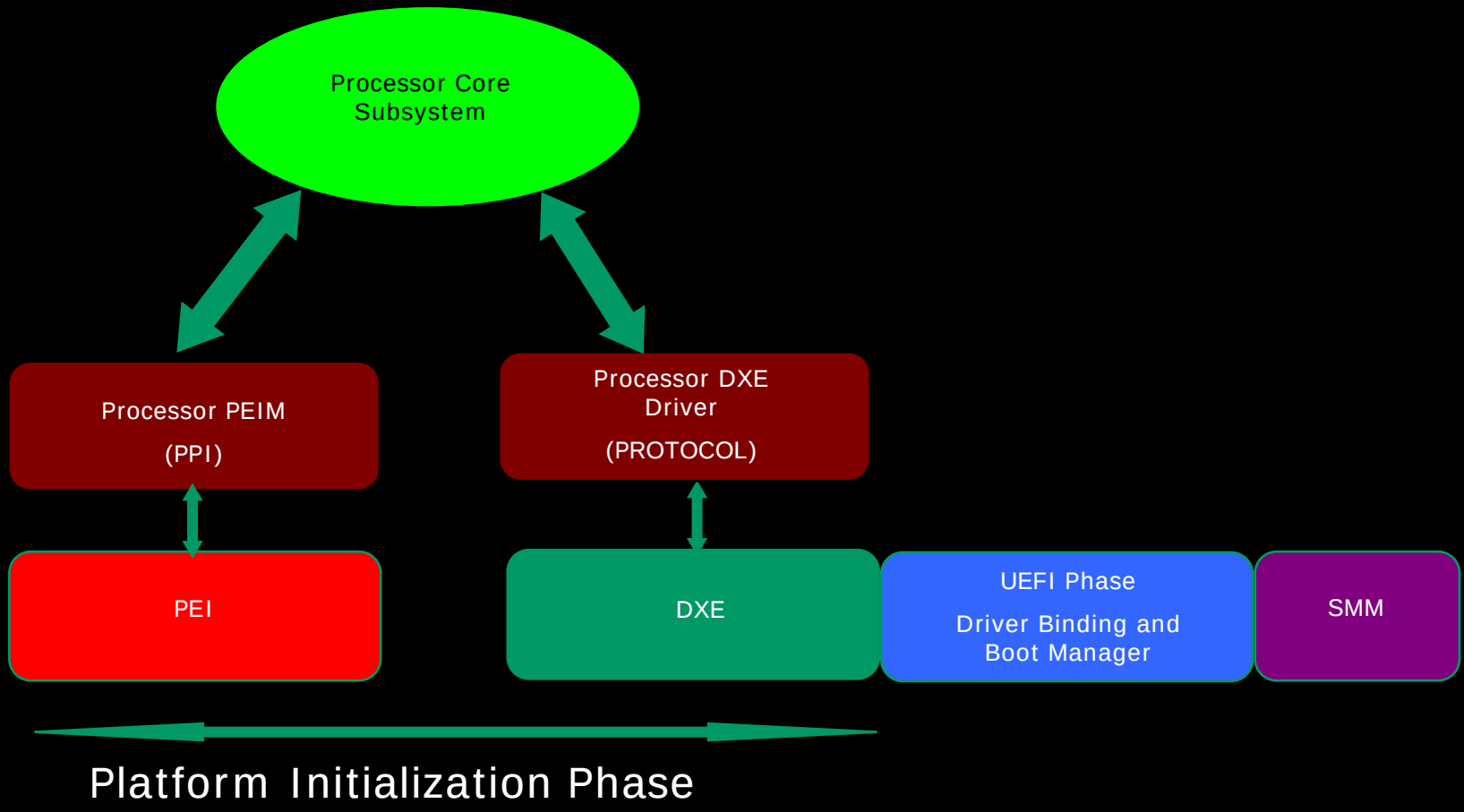
- Time to market
  - IBV and OEM/ODM can get the code at the same time to continue their development and hence expedite the product completion
- Ease of Porting
  - The standardizing the interface reduces or eliminates porting time
  - Hide the complexities associated with any new silicon features
- Code Quality and Maintainability
  - Every silicon may bring newer method for initializing feature set
  - Common module hides the complexities behind standardized interface.
  - Fully validated code on AMD reference platform
  - Easy for debugging and post production support

AMD Silicon Components

The Silicon components indicated here consists of

- Processor Core Subsystem
- Chipset

Common Firmware Interface (Processor)



AMD Processor Core Subsystem

AMD Processor core subsystem consist of following components

-CPU Core

-Hyper Transport

-Memory Controller

Functionality of Processor Core Subsystem

- CPU Initialization
  - Sets up the Cache As RAM
  - Initializes CPU Core i.e. initializing BSP and APs.
  - APs remain in Halt State through out the POST unless it requires any initialization
  - Create Processor specific information i.e.. AMD PowerNow™ information
- Hyper Transport Initialization
  - Initializes the Coherent Links between Processors
  - Initializes the Non Coherent Links between Processor and Chipsets

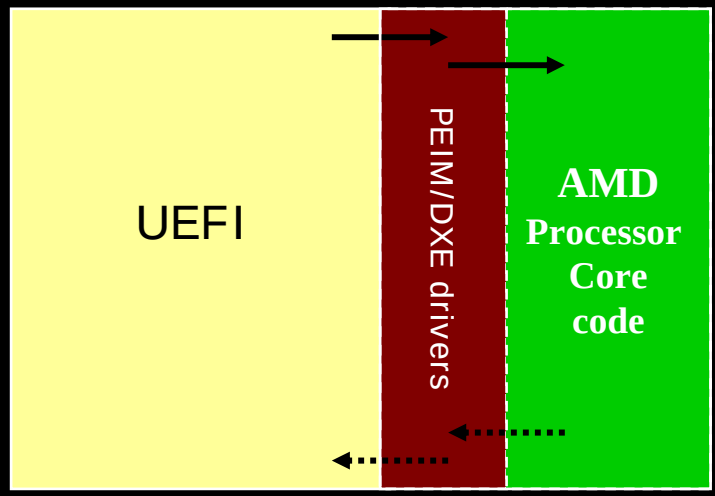
(Cont.)

Functionality of Processor Core Subsystem

- Memory Controller Initialization
  - Performs Memory Controller Init
  - Performs DRAM Controller Init
  - Creates System Memory Map
  - Sets up the cacheability for system memory
-

Calling Interface (Processor Core)

- PEIM and DXE drivers call out to Agesa Ppi and Agesa Protocol depending on phase of execution to access the services for Processor Subsystem



Example of Calling Interface

```
typedef struct _AGESA_PPI_PPI {
 CPU_INITIALIZE_RESET       CpuInitializeReset;
 CPU_LOAD_UCODE             CpuLoadUcode;
 CPU_VALIDATE_UCODE         CpuValidateUcode;
 CPU_HT_INITIALIZE          CpuHtInitialize;
 ..
 ..
} AGESA_PPI;
```

Example of Calling Interface

```
 EFI STATUS  Status

AGESA_PPI   *AgesaPpi;

(*PeiServcies).LocatePpi (PeServices,

          ..

        &AgesaPpiGuid,

        AgesaPpi

);

AgesaPpi->InitializeReset()
```

Example of Calling Interface

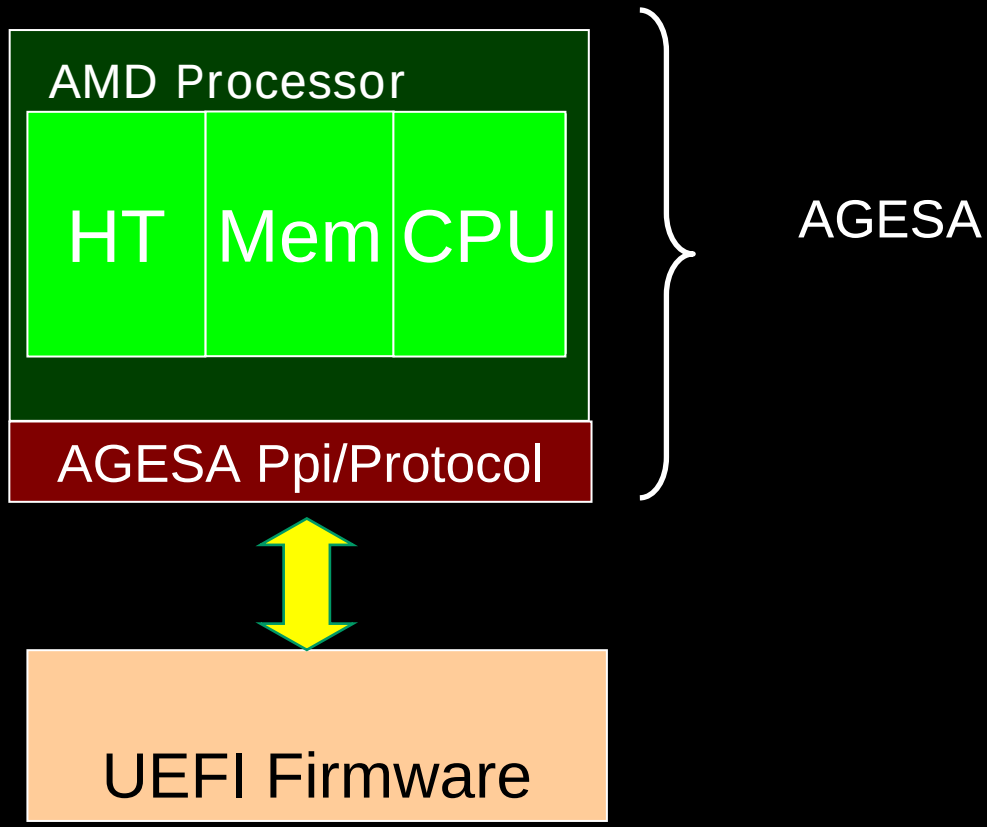
```
typedef struct _AGESA_DXE_PROTOCOL {
 CPU_PSTATE_CREATE_ACPITABLES  CpuPstateCreateAcpitables;
  CPU_PSTATE_GATHER_DATA       CpuPstateGatherData;
  CPU_ACPI_SRAT                CpuAcpiSrat;
  CPU_INITIALIZE_LATE          CpuInitializeLate;
 ..
} AGESA_PROTOCOL;
```

Example of Calling Interface

```
EFI_STATUS  Status

AGESA_PROTOCOL   *AgesaProtocol;

Status = gBS->LocateProtocol (&gAgesaDxeProtocolGuid, NULL,
&AgesaDxeProtocolPtr);

AgesaProtocol->CpuPstateCreateAcpitables ();
```

What does AMD deliver

- AMD delivers the Processor Core Subsystem as AGESA. It also supplies the PEIM and DXE Drivers which produces AGESA Ppi and AGESA Protocol



What is “AGESA”

A – AMD – Make it easy for customers to ship AMD processor based systems; reducing time, cost, training

G – Generic – It works in any customer’s development environment

E – Encapsulated – OOP term meaning all knowledge about the processor is contained in this package

S – Software – applying software development methodologies to be modular and maximize reusable code

A – Architecture – designed for the long term, spanning many processor families and reducing customer changes

Components of AGESA

- Hyper Transport Component
  - HT component initialization gets called out for each link.
  - There is a separate call out for Frequency and Width
  - Code is generic to support multi node and multi link
  - OEM Hooks are called before and after cHT and ncHT initialization

(Cont.)

Components of AGESA

- CPU Components
  - Provides services to be used by IBV through PPI and Protocols
  - Supports
    - Cache As Ram Initialization
    - CPUID
    - Platform Type (UMA, non UMA)
    - ACPI objects i.e. _PSS, _XPSS, _PST, _PSD
    - ACPI Tables
    - BrandID and MSR Initialization
    - Microcode Patch Loader
    - Processor Power State Information
    - SMM Initialization

Components of AGESA

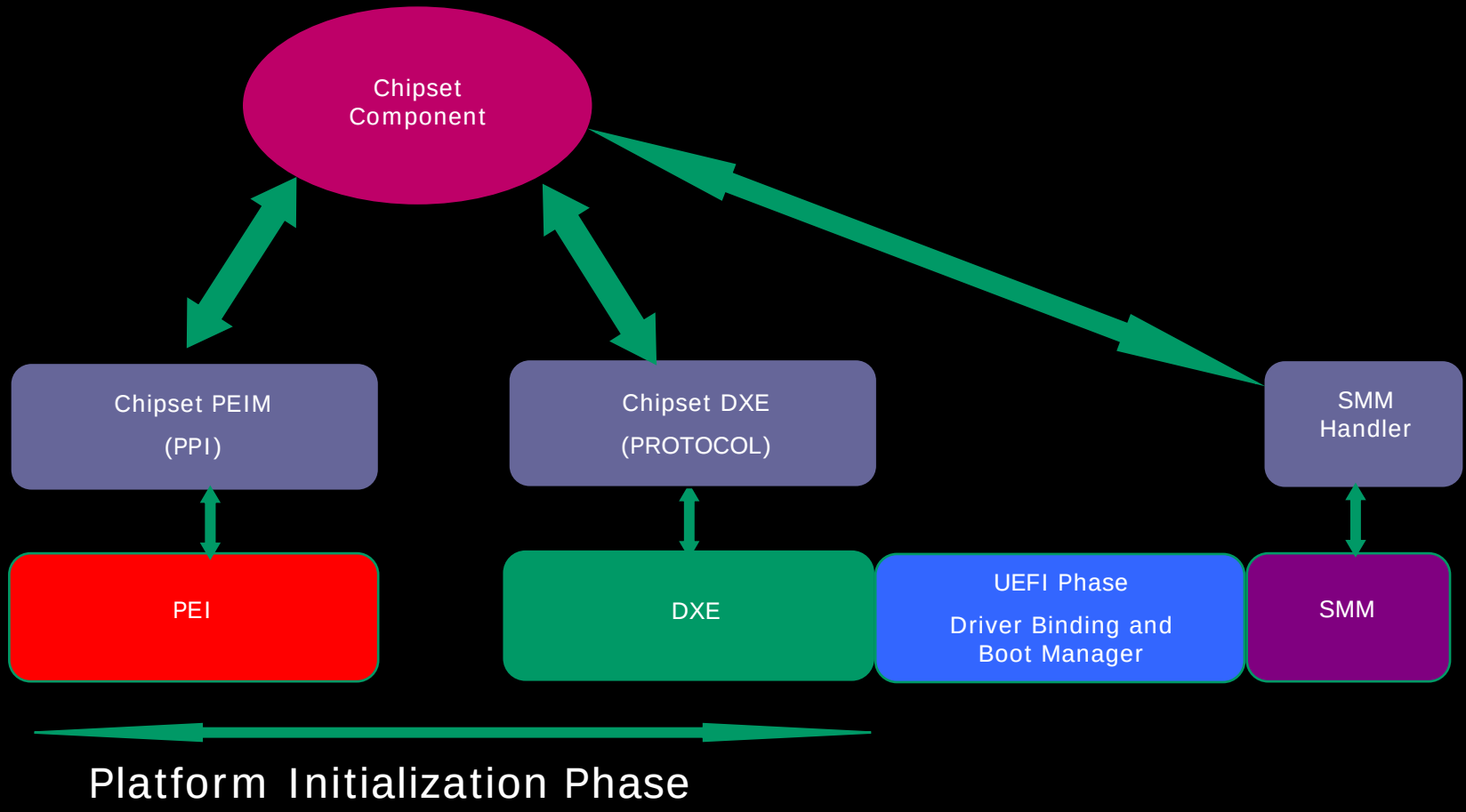
- Memory Controller
  - Handles memory initialization during normal POST and S3 Resume
  - Provides single entry point for Memory Controller Init and DRAM Controller Init.
  - Callback to support
    - User configuration through NVRAM Read/Write
    - SPD Read
    - Setting of platform specific timing value.

AGESA Packaging

AMD Smarter Choice

- AGESA Package, currently contains source code for processor core subsystem and associated PPI and PROTOCOLS along with implemented PEIM and DXE Drivers (both DXE as well as SMM phase)

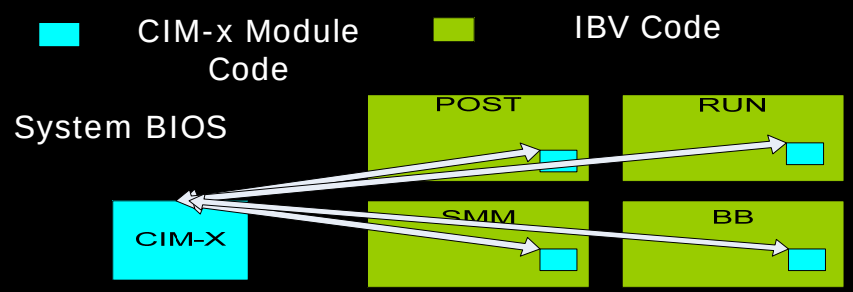
Common Firmware Interface (Chipset)

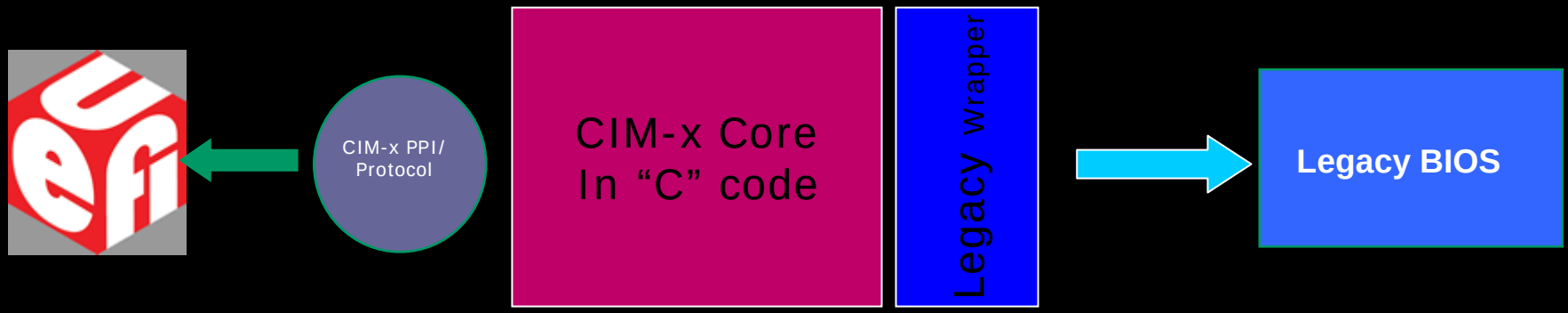


Chipset Components

- Chipset component is called CIM-x
- The CIM-x components are distributed as
  - CIM-x NBB1 (Boot Block), NBB2 (Normal Boot)
  - CIM-x SBB1 (Boot Block), SBB2 (Normal Boot)
- CIM-x NB and CIM-x SB are distributed independently to support system design based upon customers requirement

CIM-x Enabling Model





CIM-x Feature Summary

AMD Smarter Choice

- Same architecture for Northbridge and Southbridge

- Most of the code written in C to support both the world i.e.. UEFI and Legacy

- IBVs calls CIM-x modules through Interface and Gate files in legacy BIOS world.

- UEFI BIOS uses PPI or PROTOCOL (UEFI interface) to execute CIM-x C files.

- Interface available to support POST, Runtime, SMM and BootBlock for both the world

- Brings consistency in providing solution to both the world without compromising on quality.

CIM-x Model For UEFI

- Core Code for CIM-x in “C”
- PPI and PROTOCOL to access the core code
- CIM-x PEIMs and Drivers provides access to CIM-x core code for any PPI and DXE modules
- Packaging scheme is common to both CIM-x Northbridge and CIM-x Southbridge

Example of Calling Interface (CIM-x Northbridge)

```
typedef struct _PEI_AMD_NB_PPI {
  PEI_NB_POWERON_RESET     NbPowerOnInit;
  PEI_NB_HT_INIT           NbHtInit;
  PEI_NB_PCIE_BASEADDRESS  NbPcieBaseSet;
} PEI_AMD_NB_PPI;

extern EFI_GUID gPeiAmdNbPpiGuid;
```

Example of Calling Interface (CIM-x Northbridge)

```
EFI_STATUS  Status

PEI_AMD_NB_PPI *AmdNbPpi;

(*PeiServcies).LocatePpi (PeServices,

          ..

        & gPeiAmdNbPpiGuid,

        &AmdNbPpi

);

AmdNbPpi->NbPowerOnInit()
```

Example of Calling Interface (CIM-x Northbridge)

```
typedef struct _EFI_AMD_NB_CIM_X_PROTOCOL {
  NBCFG                                 NbPolicy;
  PWMCFG                                PwmPolicy;
  EFI_AMD_NB_INIT                       NbInitPciEarly;
  EFI_AMD_NB_INIT                       NbInitEarlyPost;
  EFI_AMD_NB_INIT                       NbInitLatePost;
  EFI_AMD_NB_INIT                       NbInitPciELate;
  EFI_PCI_ROOT_BRIDGE_IO_PROTOCOL *PciRootBridgeIo;
} EFI_AMD_NB_CIM_X_PROTOCOL;



extern EFI_GUID gEfiAmdNbCimXProtocolGuid;;
```

Example of Calling Interface (CIM-x Northbridge)

```
EFI_STATUS  Status

EFI_AMD_NB_CIM_X_PROTOCOL *NbCimXProtocolPtr;

Status = gBS->LocateProtocol (& gEfiAmdNbCimXProtocolGuid,
NULL, &NbCimXProtocolPtr);

NbCimXProtocolPtr->PciRoorBridgeIo ();
```

Example of Calling Interface (CIM-x Southbridge)

```
typedef struct _PEI_AMD_SB_PPI {

  PEI_SB_POWERON_RESET     SbPowerOnInit;

} PEI_AMD_SB_PPI;


extern EFI_GUID gPeiAmdSbPpiGuid;
```

Example of Calling Interface (CIM-x Southbridge)

```
EFI_STATUS  Status

PEI_AMD_SB_PPI *AmdSbPpi;

(*PeiServcies).LocatePpi (PeServices,

          ..

        & gPeiAmdSbPpiGuid,

        &AmdSbPpi

);

AmdSbPpi->SbPowerOnInit()
```

Example of Calling Interface (CIM-x Southbridge)

```
typedef struct _EFI_AMD_SB_CIM_X_PROTOCOL {
  SBCFG                                 SbPolicy;
  EFI_AMD_SB_INIT                       BeforePciInit;
  EFI_AMD_SB_INIT                       AfterPciInit;
  EFI_AMD_SB_INIT                       LatePostInit;
  EFI_AMD_SB_INIT                       BeforePciRestoreInit;
  EFI_AMD_SB_INIT                       AfterPciRestoreInit;
} EFI_AMD_NB_CIM_X_PROTOCOL;


extern EFI_GUID gEfiAmdSbCimXProtocolGuid;;
```

Example of Calling Interface (CIM-x Southbridge)

```
EFI_STATUS  Status

EFI_AMD_SB_CIM_X_PROTOCOL *SbCimXProtocolPtr;

Status = gBS->LocateProtocol (& gEfiAmdSbCimXProtocolGuid,
NULL, &SbCimXProtocolPtr);

SbCimXProtocolPtr->BeforePciInit ();
```

Summary

AMD Smarter Choice

• The Processor Core Subsystem code is available as AGESA

• The chipsets (Northbridge and Southbridge) code is available as CIM-x

• AMD Provides all the necessary infrastructure to enable UEFI on AMD Silicon

In a nutshell.....

AMD is committed to support UEFI

Questions and Answers

Q&A

Trademark Attribution