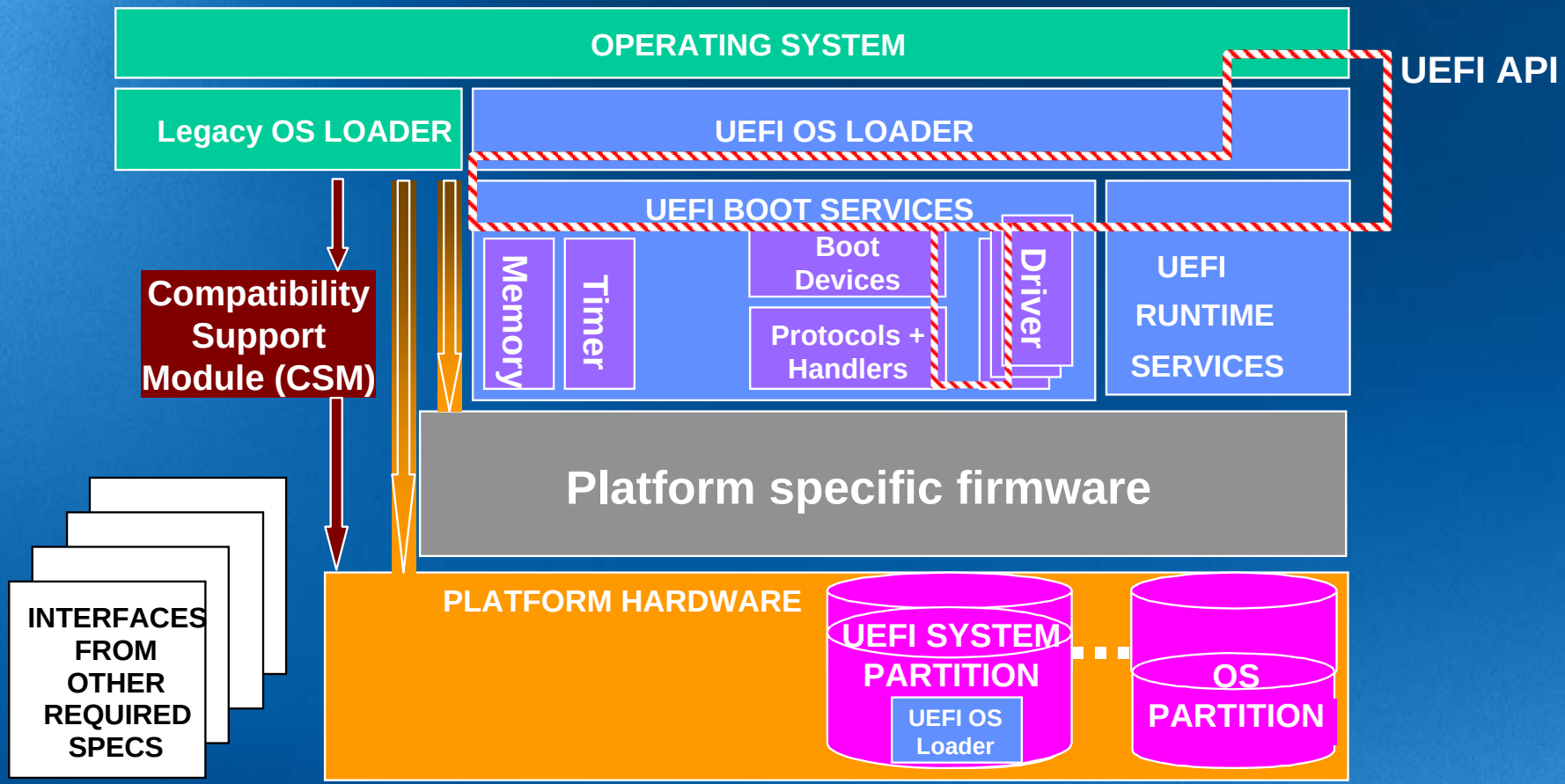
Benefits of UEFI in Manufacturing and Test

Harry Hsiung
Technical Marketing Engineer
Intel Corporation

Agenda

- UEFI in Manufacturing Environment
- Example Use of UEFI in Manufacturing
- Test Advantages through UEFI
- Industry Benefits and Support

UEFI Layered Implementation



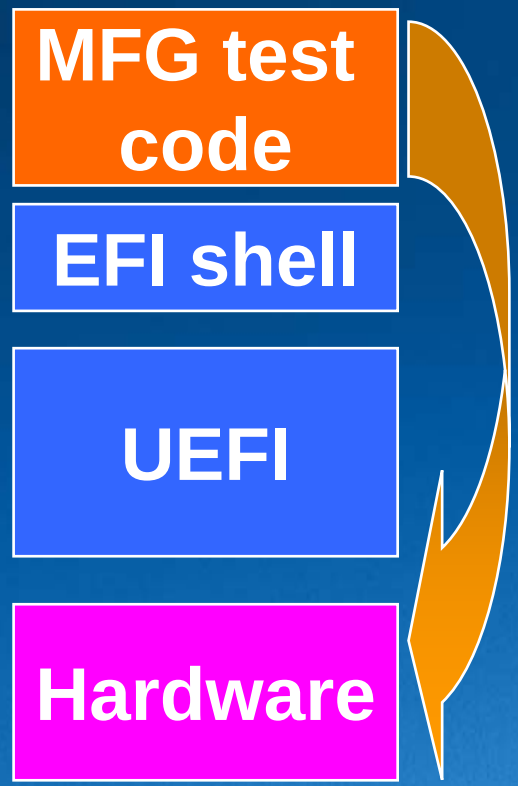
Relationship of Manufacturing Use of UEFI versus OS boot

Normal OS Boot

- OS Loader
- UEFI
- Hardware

- MFG environment migrated to EFI
- MFG test code loaded by UEFI
- UEFI test code directly interfaces with the HW.

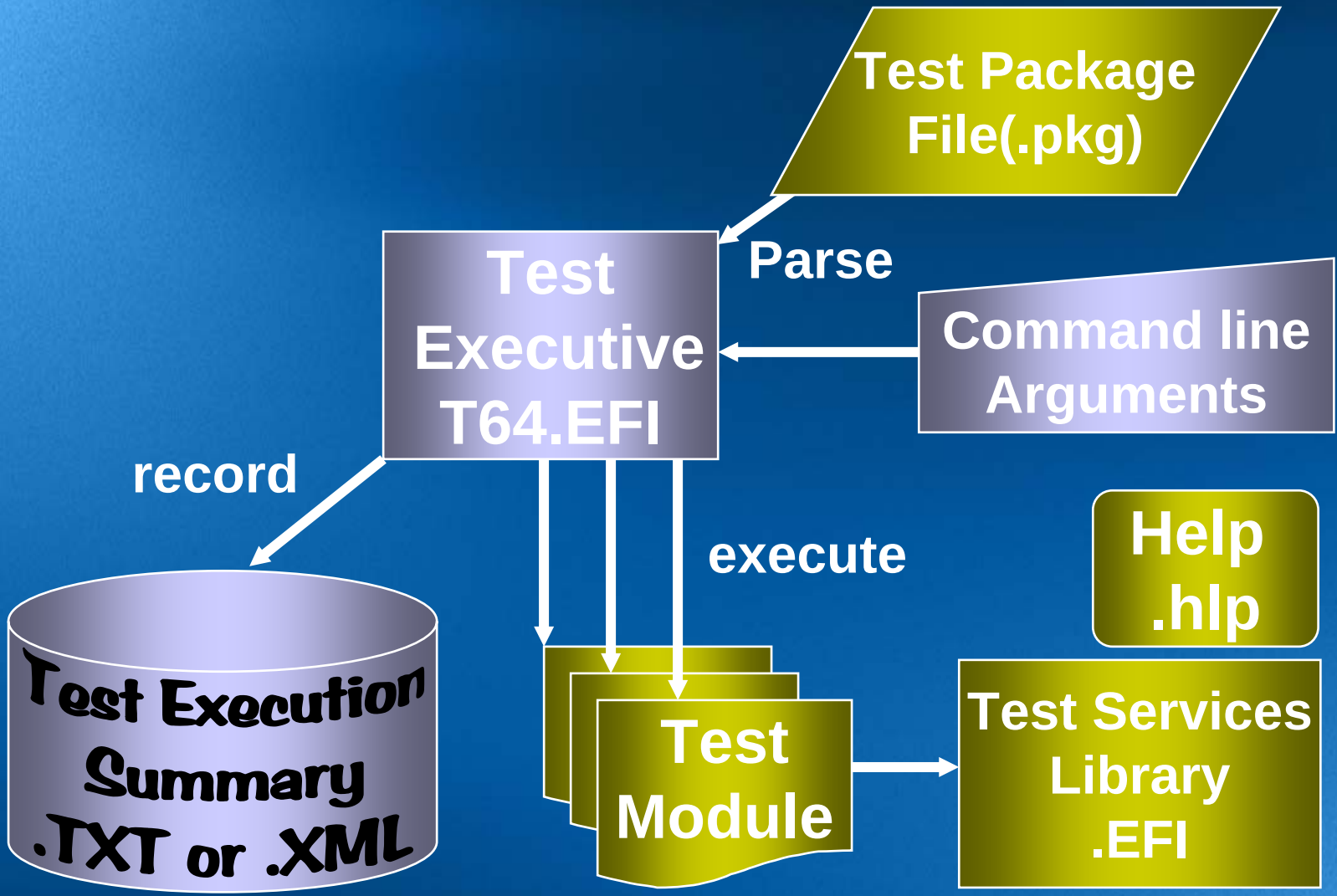
factory



Factory Flow

- Boot to network over PXE
  - Initial hardware power-up test with EFI MTA
  - Run within Shell with python scripts
- Configure cards, Nvram and Drives
  - From EFI shell using EFI 1.10 driver Configuration protocols and EFI based utilities (EFI config utility)
- Boot to WinPE*
  - System test and hardware verification for Windows
  - Pull configuration desired for system from RIS over LAN
  - Reseal installation at factory

Manufacturing Test Automation (MTA) of UEFI platform



EFI Toolkit “Tools for the Factory”

- EFI shell
- Common Utilities (edit, pci, dmem, mm drivers, dmpstore, dblk, devices, etc.)
- Libc
- EFI Lib
- Network Stack (Tcpip, Dhcp,Ftp, Ping etc)
- Configuration of system with Nvram variables and boot manager
  - Automatic loading of drivers not in flash
  - Executing test environment directly without unnecessary loading of unused drivers.
- Python script interpreter

UEFI Driver Diagnostics

- UEFI Driver Model Drivers (OptionROM code) can have built in diagnostics
- Each driver can expose a 4 tiered test
  - Standard test
  - Extended test
  - Manufacturing test
  - Maximum test
- IHV silicon/card provider should have Diagnostic protocol as part of UEFI driver

UEFI Driver Configuration

- UEFI Driver Model Drivers (OptionROM code) may have configuration protocol
- Each driver can have different configs
  - Set options
  - Validate options
  - Force defaults
- IHV silicon/card provider should have configuration protocol as part of UEFI driver

Advantages of using UEFI interface

- Loading of test environment via UEFI
  - Unnecessary to write additional code to load test code
  - Reuse of UEFI drivers (from Silicon providers and EDK)
  - Use EFI shell (interactive)
  - EFI shell freely available with source on Tianocore.org
  - Written in high level C code
  - Test environment the same across architectures
- Flat memory model
  - Memory can be tested with less interference
  - All memory is available (UEFI is relocatable)
- Full control of the system
  - Very low driver interference
  - No interrupts required except for EFI timer
  - UEFI test software has direct contact with hardware
    - CPU in Physical mode with no address translation
    - No OS kernel interference for I/O or memory access
    - Only one core used, other cores in Sipi rendevous loop

Test Time advantages (Motherboard)

- Reduced pretest time (booting to UEFI)
  - Need only bare minimum drivers to load or run EFI Shell
  - Unnecessary to load drivers not used to load test environment
- Faster test time
  - i.e. boot from LAN(pxeboot) with built in lan drivers
- Requires fewer reboots (can be 0)
- Run stop or disconnect on UEFI drivers used during test instead of reboot
- Can still boot to OS
  - Just load the UEFI OS Loader from shell or boot manager

Test portability

- Test code can compile for all platforms
  - IA-32, IA-64, Intel® 64
- UEFI test environment can be identical between architectures (cross compiled)

Test Binary Locations

- EFI has built in network stack
- EFI images can run from:
  - The firmware image
  - Local Media (HDD, USB, CD\DVD ROM)
  - Network share
- Choosing non local media:
  - No impact on customer image
  - No risk of corrupting files
  - Station re-imaging is eliminated

Example: ACD CPV using UEFI

- Impact of testing AHCI with UEFI driver
  - 10,658 capacity improvement per module
  - 32K capacity improvement per 3 modules
  - One module cost 1.5$ + floor space. Estimated savings >2M$
  - Factory has no floor space to add more module

EFI Standard Test Vision

- EFI Standard Test Interface
  - Industry standard interface for a “test module” to communicate with a “test engine”
    - Leverage validation product
    - Eliminate duplicate work
  - Tests from multiple companies interoperate
    - IHVs provide tests for their components
    - OEMs provide tests for their boards

EFI Standard Test Vision

- Currently being worked through UEFI Forum
  - Influence your future
  - Everyone is invited
    - some are contributors to UEFI already
- Open source test executive
  - Allows for customization:
    - Configuration information
    - Data collection
    - Operator interactions

Wrap-Up: Modularity & Servers

- Code base reuse and modularity enables faster time to market with new hardware platforms
- PIWG architecture enables make vs. buy
  - PEI, DXE, Drivers, etc…
- Internal company code sharing opportunities
  - Across diverse architectures – e.g., embedded and server
- Maximize flexibility in product choice points

Framework supports fast TTM and cross-architecture code sharing.

13

- Legacy: Requires specialized knowledge
  - Expertise gained through experience
- EFI: Fast-track enabled
  - EFI Specification Set
  - Sample Code
  - Training

EFI eliminates Legacy deficiencies

Developer thoughts

Portable Source Code and Executables

- Drivers written in C, no assembler needed or allowed
- EFI Byte Code (EBC) executables - architecture-neutral
- Flat memory model
- Standardized compression algorithm
- Debug using native code (IA32, IPF)

Web Resources

- www.UEFI.org
  - UEFI Specification
- www.TianoCore.org
  - Open Source (EFI Developers Kit, Shell, etc…)
  - Mailing lists for help and reporting issues
- www.Intel.com/Technology/Framework
  - Intel's Framework Specs
- www.Intel.com/Technology/EFI
  - Intel's EFI website


intel®
Leap ahead™