tianocore

UEFI & EDK II TRAINING

EDK II Open Board Platform Design for Intel Architecture (IA)

tianocore.org

LESSON OBJECTIVE

- Introduce Minimum Platform Architecture (MPA)
- Explain the EDK II Open board platforms infrastructure & focus areas
- Describe Intel® FSP with the EDK II open board platforms

Reference: Minimum Platform Architecture Specification

DESIRED USE CASES

– Open Source EDK II Platforms

Developers port a board without knowing UEFI

Developers enable functionality incrementally

Developers enable functionality incrementally

Make EDK II easer to use correctly?

INTRODUCING
Minimum Platform Architecture

tianocore

GOALS

Minimum Platform Architecture (MPA)

Structured	Enable developers to consistently navigate code, boot flow, and the functional results
Approachable	Enable developers to quickly produce a baseline that is extensible with minimal UEFI or EDK II knowledge
Portable	Minimize coupling between common, silicon, platform, board, and feature packages
Reusable	Enable large granularity binary reuse (FV binaries)
Testable	Enable validating the correctness of a port

Design open source EDK II Intel Architecture firmware

WHY?

Code Convergence & Consistency

System firmware (BIOS) is the largest payload in the IFWI binary image

Platform implementation is ~2-3 **million** lines+ of “C” code

Technology complexity increasing, strains firmware implementation solutions

Limited firmware engineering resources

Copy + Paste + Modify = Human Errors

tianocore

Why Move to Open Source?

Goal:

- Enable improvements in quality and security for Intel products
- Enable vertically integrated open solutions

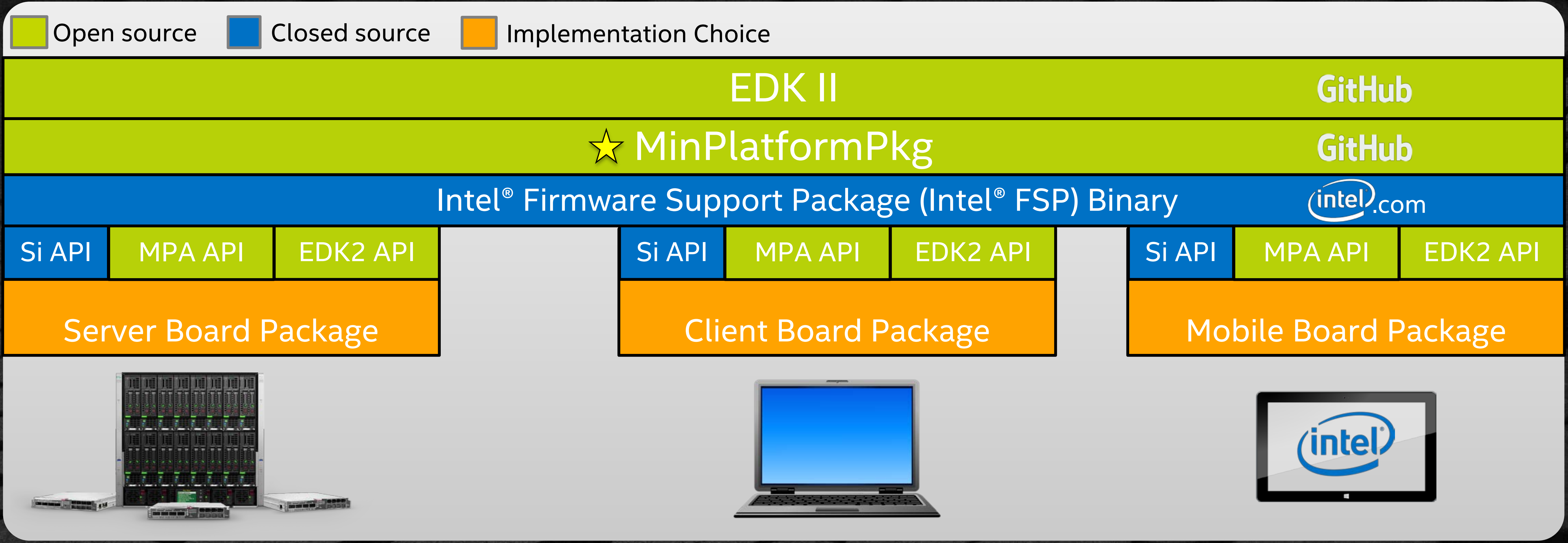
Benefits:

- Allow improved customer engagements
- Builds transparency and trust
- Reduce overhead to transition from internal to external
- Deploy fixes across the ecosystem more rapidly

Easier to access, understand, fix & optimize means improved product quality

tianocore

MinPlatform + Intel® Firmware Support Package (Intel® FSP)



🟩 Open source 🟦 Closed source 🟧 Implementation Choice

EDK II — GitHub

☆ MinPlatformPkg — GitHub

Intel® Firmware Support Package (Intel® FSP) Binary — intel.com

Si API | MPA API | EDK2 API — Server Board Package

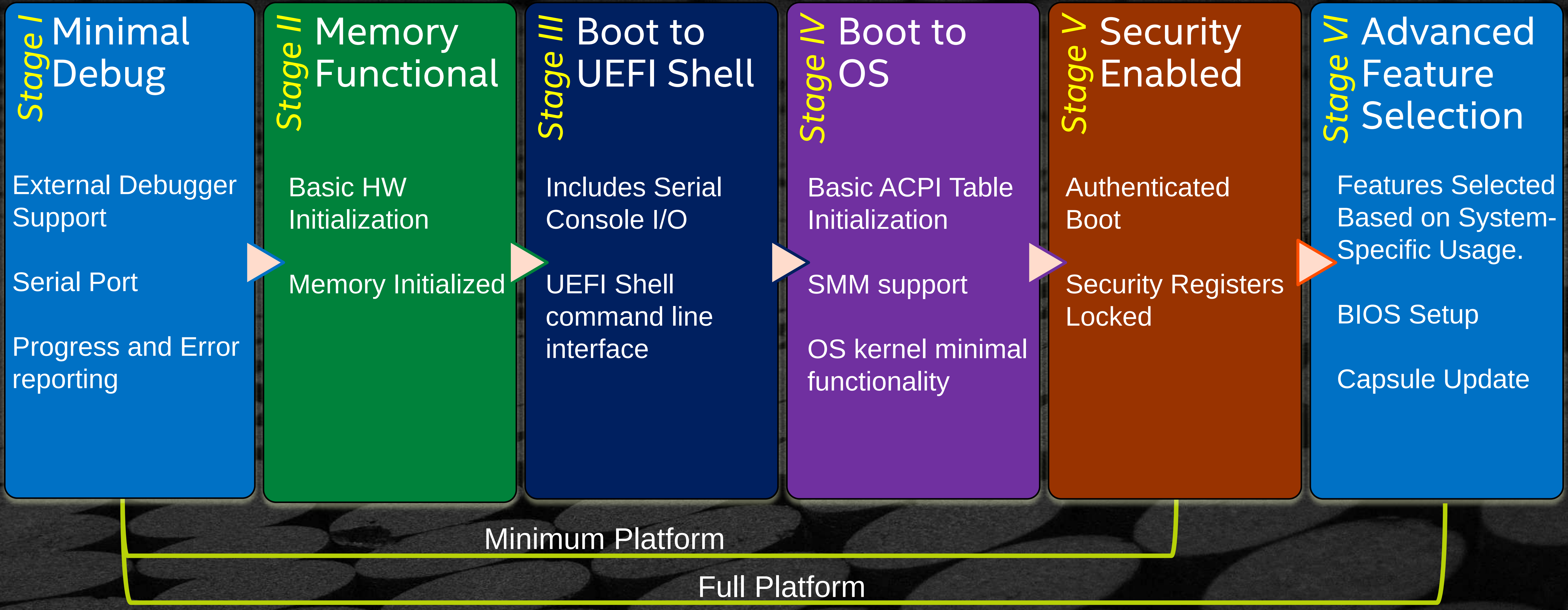
Si API | MPA API | EDK2 API — Client Board Package

Si API | MPA API | EDK2 API — Mobile Board Package

Intel Open Platform Firmware Stack - Minimum Platform

Consistent boot flows and interfaces
Approachable across the ecosystem
Scalable from pre-silicon to derivatives

What are Minimum Platform Stages?



Stage I Minimal Debug

- External Debugger Support
- Serial Port
- Progress and Error reporting

Stage II Memory Functional

- Basic HW Initialization
- Memory Initialized

Stage III Boot to UEFI Shell

- Includes Serial Console I/O
- UEFI Shell command line interface

Stage IV Boot to OS

- Basic ACPI Table Initialization
- SMM support
- OS kernel minimal functionality

Stage V Security Enabled

- Authenticated Boot
- Security Registers Locked

Stage VI Advanced Feature Selection

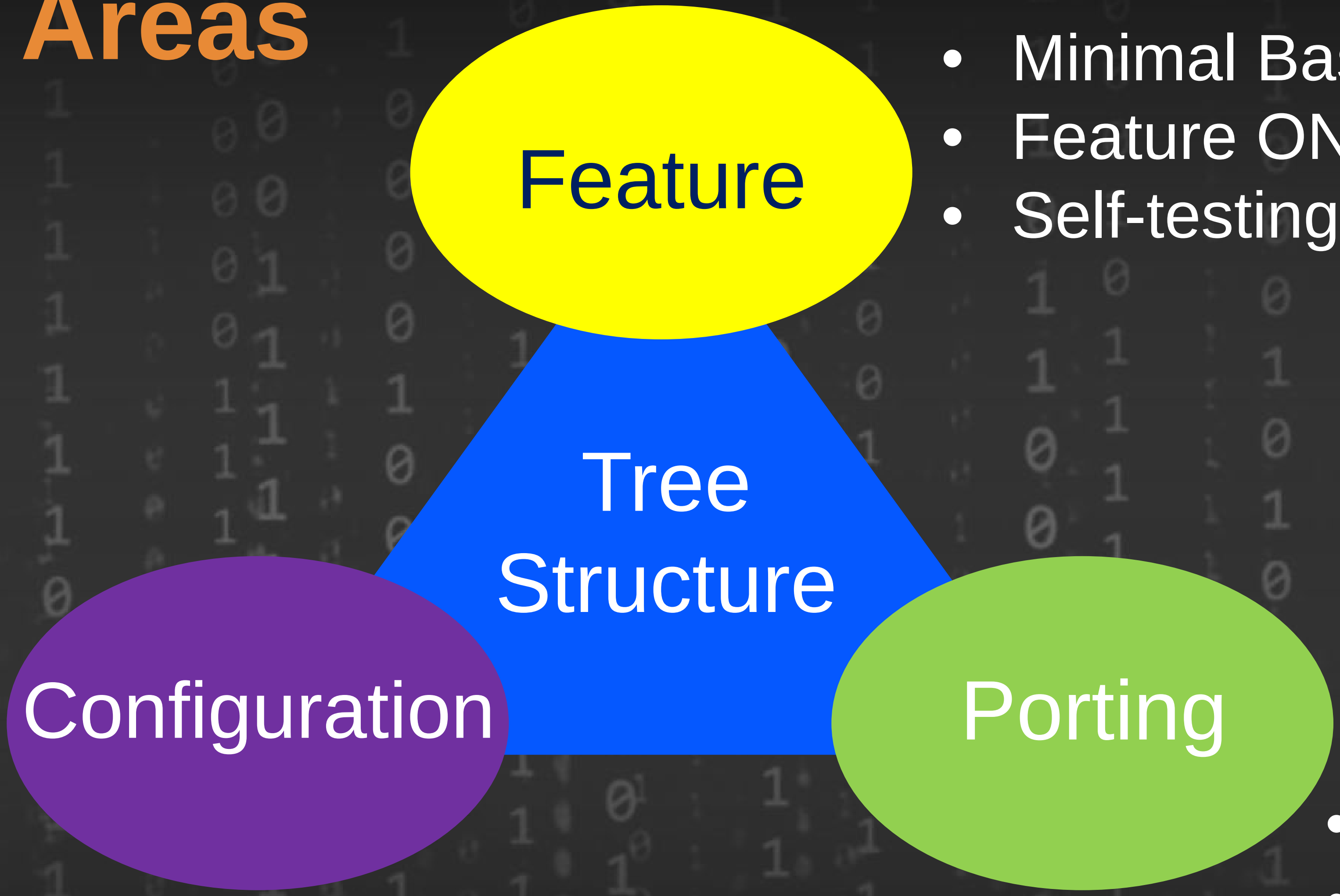
- Features Selected Based on System-Specific Usage.
- BIOS Setup
- Capsule Update

Minimum Platform

Full Platform

Stages reflect firmware development lifecycle and how a system bootstraps itself

Four Focus Areas

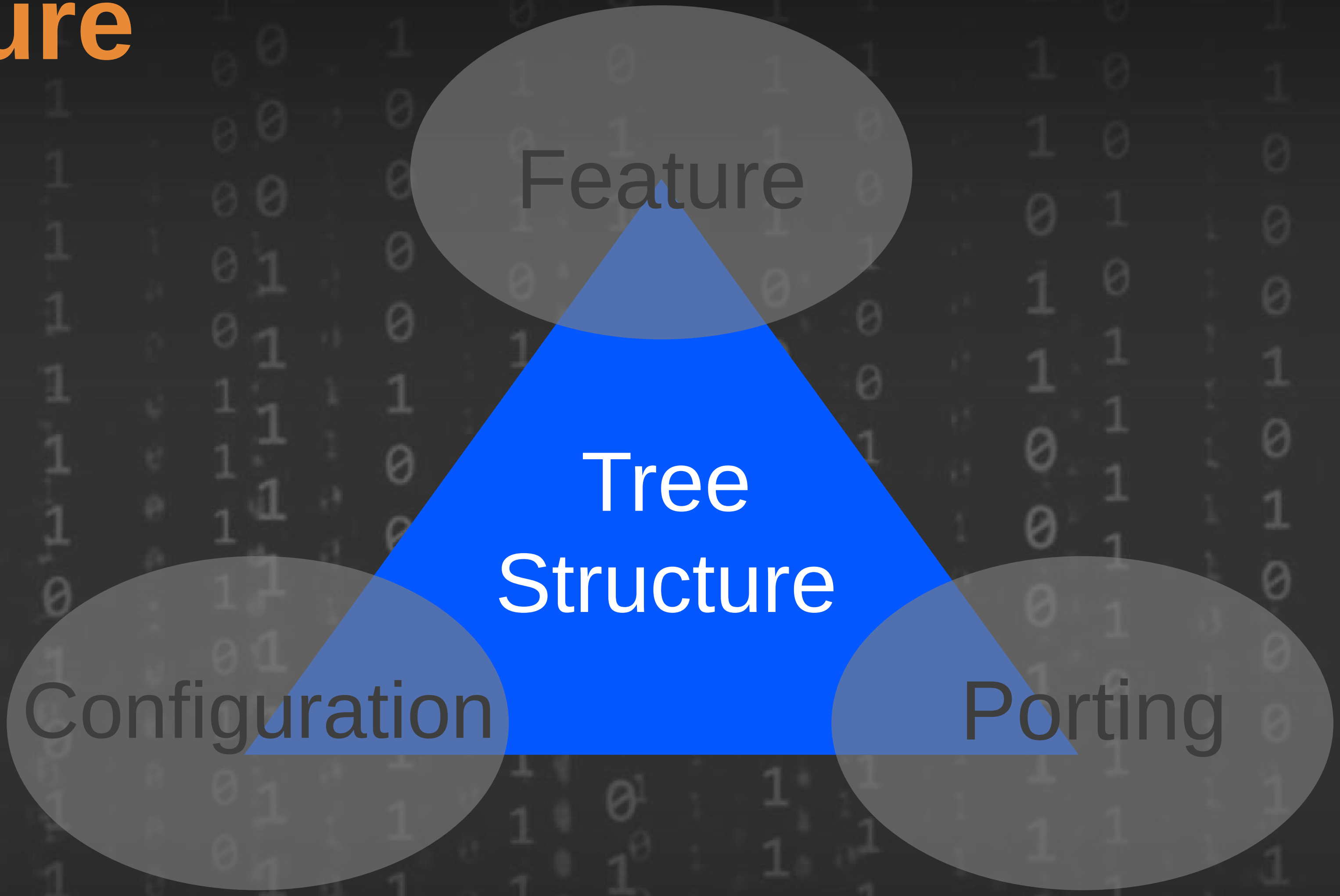


- Minimal Baseline
- Feature ON/OFF
- Self-testing

- Incremental
- Simple PCD usage model
- No setup

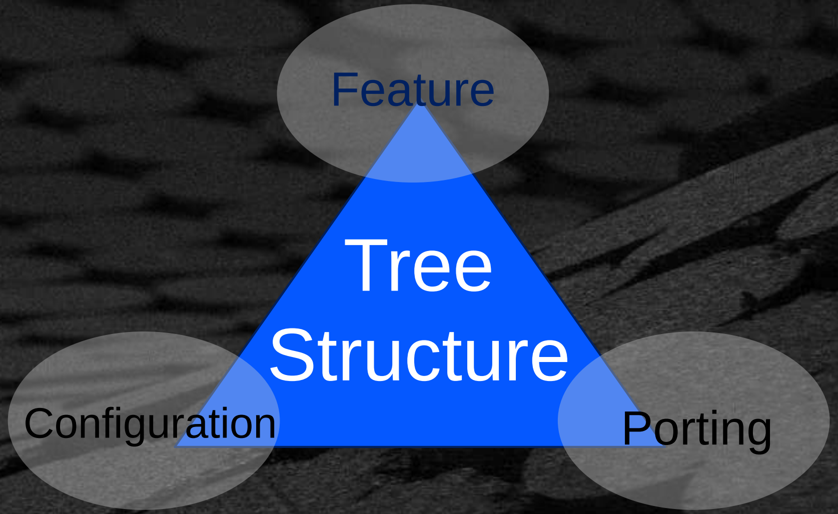
- Incremental
- Simple C libraries
- The same each time

Tree Structure



Organization

Common	• No direct HW requirements
Platform	• Enable a specific platform's capabilities.
Board	• Board specific code
Silicon	• Hardware specific code"



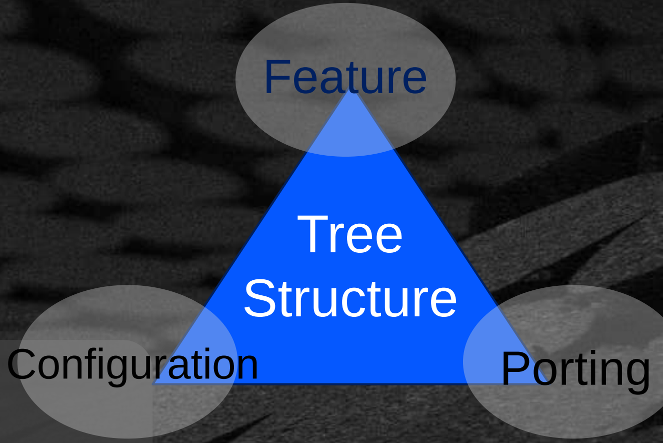
Feature

Tree Structure

Configuration

Porting

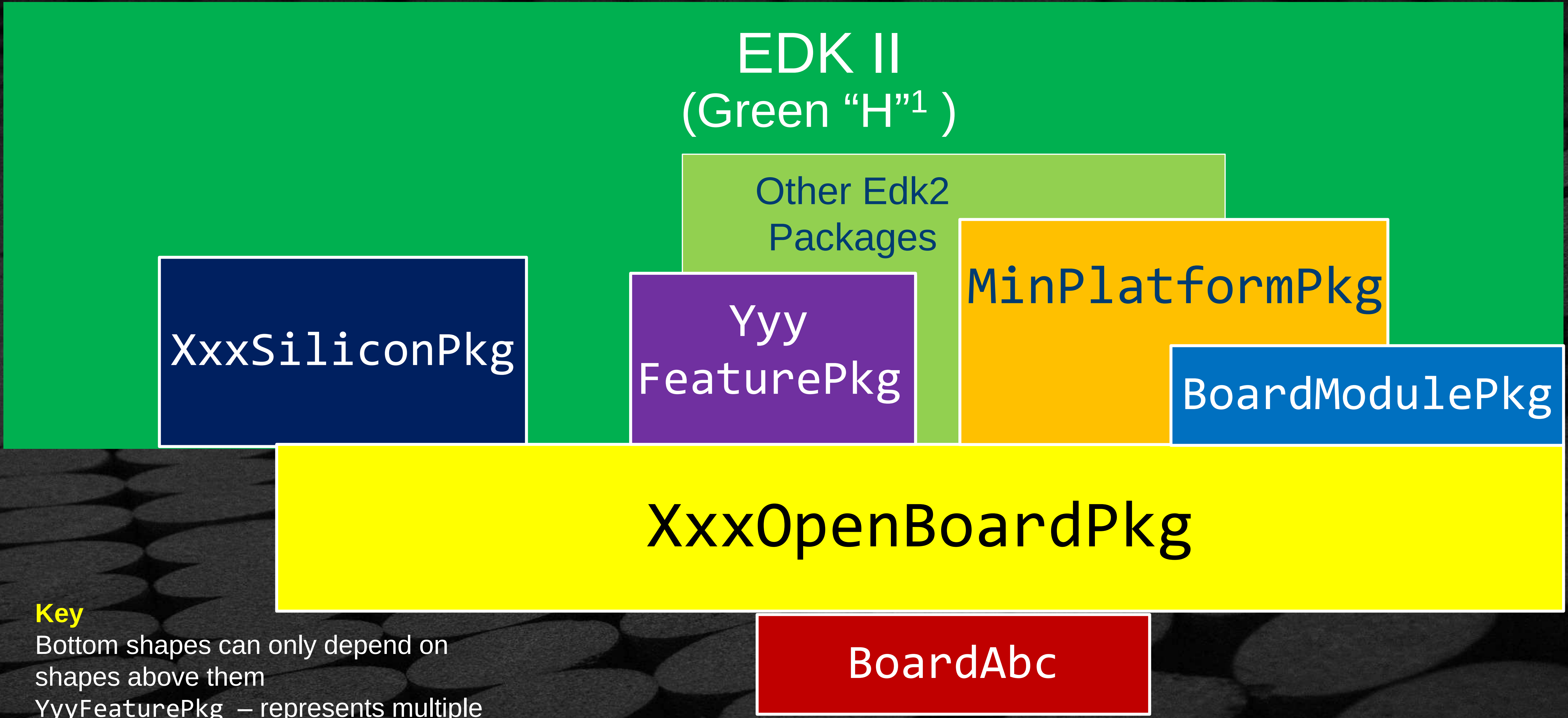
Package Organization Example



`MinPlatformPkg`	• Common - Boot flow, well defined interfaces
`XxxOpenBoardPkg`	• Platform Xxx* - board-specific details: GPIOs, memory config, audio verb tables, etc.
`XxxSiliconPkg`	• Silicon - Hardware specific code for Xxx
`YyyFeaturePkg`	• Features – Advanced functional features

*Where Xxx would be an open platform: `KabyLakeOpenBoardPkg`, `WhiskeyLakeOpenBoardPkg`, etc.

MPA Dependency Rules



Key
Bottom shapes can only depend on shapes above them
`YyyFeaturePkg` – represents multiple feature package instances that are mutually exclusive to each other

BoardAbc – directory for `OpenBoardPkg.dsc`

[1] GreenH = MdePkg + UefiCpuPkg +IntelSiliconPkg + BaseTools

Open Source EDK II Workspace

```
MyWorkSpace/
     edk2/
        - “edk2 Common”
     edk2-platforms/
        Platform/ “Platform”
           Intel/
             MinPlatformPkg/“Platform”
             BoardModulePkg/
             XxxOpenBoardPkg/ “Board”
                BoardX/ “Board Instance”
        Silicon/ “Silicon”
           Intel/
             XxxSiliconPkg/
        Features/ “any”
     edk2-non-osi/
        Silicon/
           Intel/
     FSP/“Silicon”
        . . ./
```

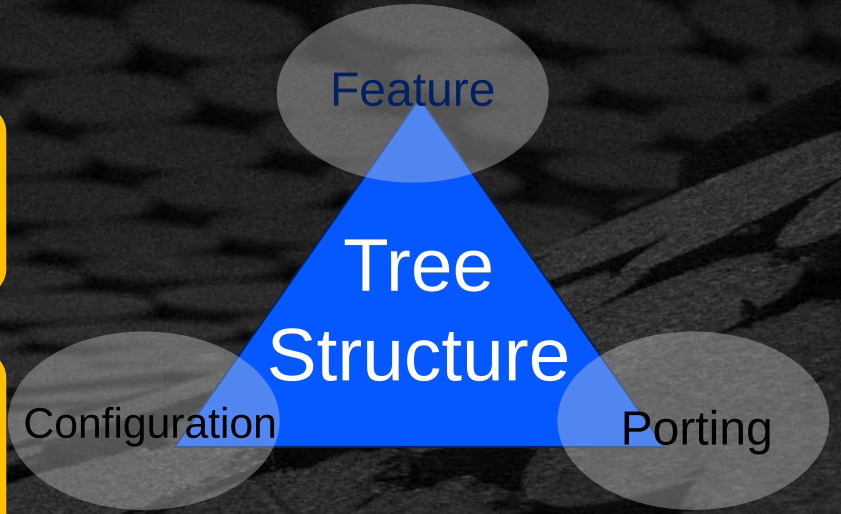
Common

Platform

Board

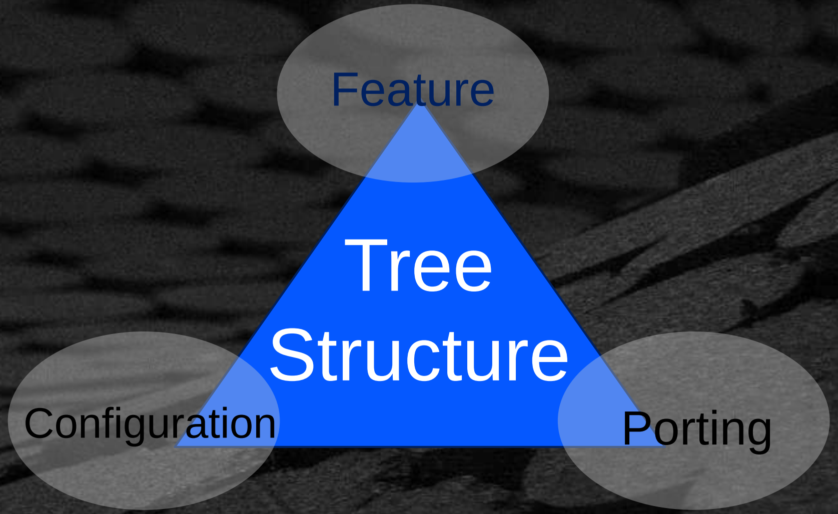
Silicon

Features



Open Board Tree Structure

```
edk2/ https://github.com/tianocore/edk2       ← Common
 . . .
edk2-platforms/ https://github.com/tianocore/edk2-platforms
  Platform/
       Intel/
            BoardModulePkg
            KabylakeOpenBoardPkg              ← Board (family)
                KabylakeRvp3                  ← Board (instance)
            MinPlatformPkg                    ← Platform (common)
  Silicon/
       Intel/
            KabylakeSiliconPkg                ← Silicon
            . . .
  Features/Intel
              AdvancedFeaturePkg              ← Features
edk2-non-osi/ https://github.com/tianocore/edk2-non-osi
    Silicon/
       Intel/
             KabylakeSiliconBinPkg            ← Silicon
             PurleySiliconBinPkg
FSP/   https://github.com/IntelFsp/FSP
       KabylakeFspBinPkg                      ← Silicon
```



Directory Description

edk2-platforms: EDK II repo includes open source platform code

- Platform folder: contains the platform specific modules by architecture
  - BoardModulePkg: generic board functionality (board Lib interfaces)
  - MinPlatformPkg: generic platform instance to control the boot flow.
  - <Generation>OpenBoardPkg: the silicon generation specific board package. All of the boards based upon this silicon generation can be located here.
- Silicon folder: contains the silicon specific modules.
  - <Generation>SiliconPkg: the silicon generation specific silicon package.
- Features/Intel folder: contains Advanced features packages.
  - <XxxFeature>Pkg: package and modules for advanced features

edk2-non-osi: EDK II repo for platform modules in binary format (ex: silicon init binaries).

- <Generation>SiliconBinPkg: It is the silicon generation specific binary package. For example, CPU Microcode or the silicon binary FVs.

Ideally, Only <Generation>OpenBoardPkg needs updating

FSP Directory Description

FSP: repo for Intel® Firmware Support Package (FSP) binaries https://github.com/intel/FSP

Platform folder Pkg: Each FSP project will be hosted in a separate directory

- ApolloLakeFspBinPkg Intel® Atom™ processor E3900 product family
- . . .
- CoffeeLakeFspBinPkg - 8th Generation Intel® Core™ processors and chipsets (formerly Coffee Lake and Whiskey Lake)
- KabylakeFspBinPkg 7th Generation Intel® Core™ processors and chipsets
  - Include
    - FSP UPD structure and related definitions used with EDK II build
  - Doc - Integration Guide .PDF documentation
  - FSP.fd - Binary to be included with flash device image
  - FSP.bsf - Configuration File with IDE configuration tool https://github.com/IntelFsp/BCT

FSP each project based on Intel Architecture

Platform Package Structure

MinPlatformPkg

```
MinPlatformPkg /
   <Basic Common Driver>/
   Include /
   Library /
   PlatformInit /
```

Platform Common Driver

Where:

- **<Basic Common Driver>**: The basic features to support OS boot, such as ACPI, flash, and FspWrapper. It also includes the basic security features such as Hardware Security Test Interface (HSTI).
- **Include**: The include file as the package interface. All interfaces defined in MinPlatformPkg.dec are put to here.
- **Library**: It only contains feature independent library, such as PeiLib. If a library is related to a feature, this library is put to <Feature>/Library folder, instead of root Library folder.
- **PlatformInit**: The common platform initialization module. There is PreMemPEI, PostMemPEI, DXE and SMM version. These modules control boot flow and provide some hook point to let board code do initialization.

Open Board Package Structure

<Generation>OpenBoardPkg

```
<Generation>OpenBoardPkg /
  <BasicCommonBoardDrivers>/
  Include /
  Library /
  Features /
     <AdvancedCommonBoardDrivers> /
  <BoardX> /
     Include/
     Library/
     <BoardSpecificDriver> /
     OpenBoardPkg.dsc
     OpenBoardPkg.fdf
```

Where:

- <BasicCommonBoardDrivers> and <AdvancedCommonBoardDrivers> designate a board generation specific feature. They need to be updated when we enable a board generation.
- <Board> contains all the board specific settings. If we need to port a new board in this generation, copy the <Board> folder and update the copy’s settings

One Feature, One directory Guideline

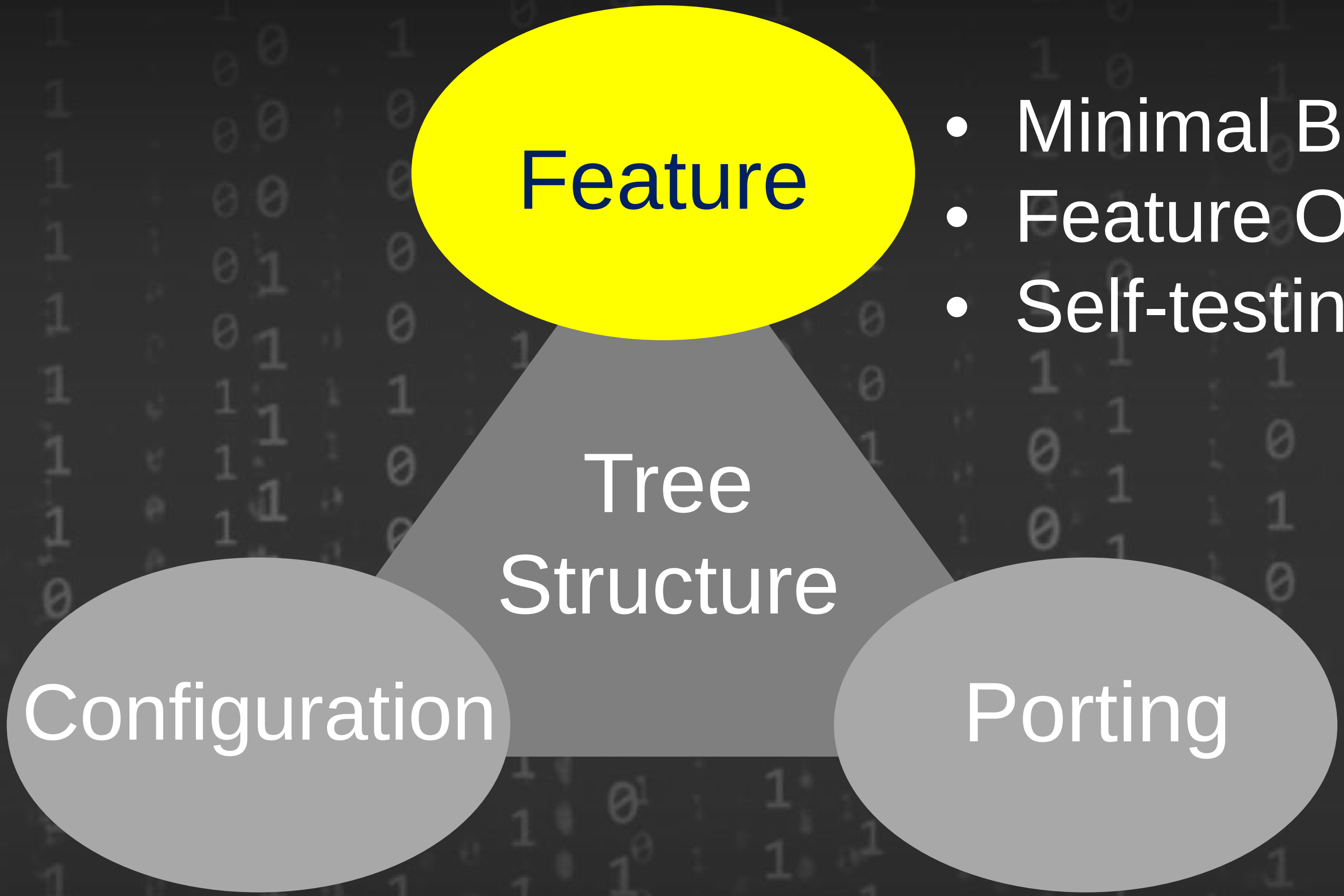
Use a hierarchical layout , KabylakeOpenBoardPkg example

```
KabylakeOpenBoardPkg /
  Acpi /
    BoardAcpiDxe  /
  FspWrapper /
    Library /
    PeiFspPolicyUpdateLib /
  Include /
  KabylakeRvp3
  Library /
    BaseEcLib /
    BaseGpioExpanderLib /
    PeiI2cAccessLib /
  Policy /
    Library /
```

```
KabylakeRvp3 / (cont.)
  Include /
  Library /
  OpenBoardPkg.dsc
  OpenBoardPkg.fdf
```

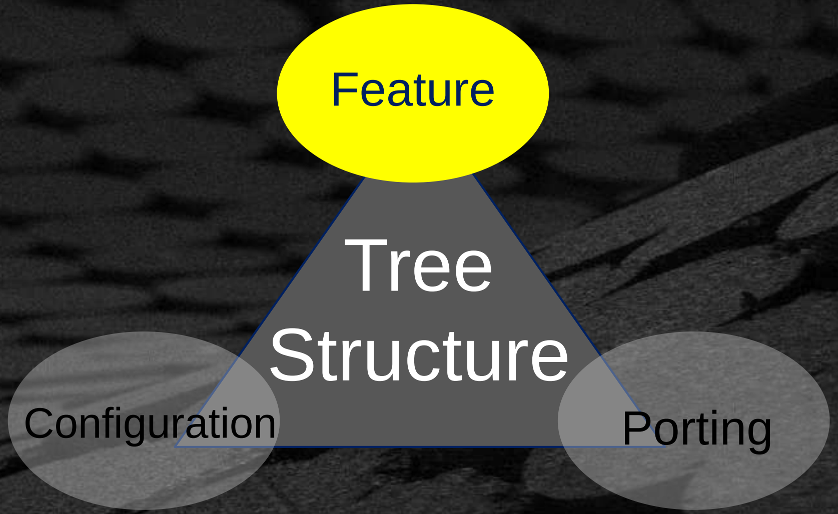
Only put the basic features into the root directory

Features



- Minimal Baseline
- Feature ON/OFF
- Self-testing

Feature Selection

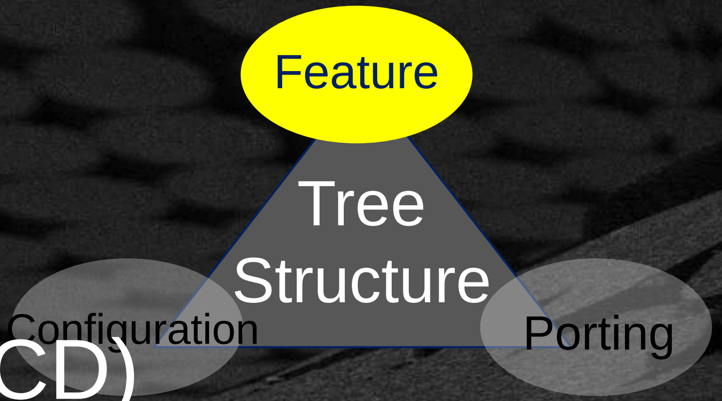


There are three phases of feature selection

Minimum	Manage Stage I-V options
Advanced Feature Selection	Add rich feature sets (Stage VI)
Optimization	Remove undesired features (Stage VII)

Select features through build, prune in binary

Minimum Platform Feature Selection



Minimum Platform

- Minimum feature selection should be exclusively implemented as Platform Configuration Database (PCD)
- Required PCD are identified in the MPA specification
- PCDs:
  - Declared with defaults in DEC files in different packages
  - Modified in DSC file for the board, if different than the default value

Silicon – FSP Integration from <Generation>FspBinPkg documentation package

All initial porting features selection should be done this way

Advanced Features- Should be:

Approachable

- Easy to identify, modify, use, and debug

Consistent

- Follow a pattern that reduces arbitrary design differences

Consolidated

- 1:1 mapping between an Intel feature and firmware feature code

Isolated

- No direct coupling between features
- More cohesion within feature code

Self-documenting

- Maintain documentation with the code – easy to find with a shared maintainer.

Self-testing

- Minimize foreseeable errors

Advanced Feature Selection Implementation

`AdvancedFeaturePkg/Include` has DSC and FDF files to include in your board’s DSC and FDF files. The order is important.

Each feature has a `PcdsFeatureFlag` defined to enable the feature.

Each feature PCD is OFF by default

Some features have more PCD configuration or library requirements.

OpenBoardPkg.dsc File

. . .

!include *^{1}…/AdvancedFeaturesPcd.dsc

gAcpiDebugFeaturePkgTokenSpaceGuid.PcdAcpiDebugFeatureEnable|TRUE

<End-of-File>

!include * ^{1}…/AdvancedFeatures.dsc

OpenBoardPkg.fdf File

[FV.FvAdvancdPreMemory]

!include * ^{1}../PreMem.fdf

[FV.FvAdvanced]

!include * ^{1} …/PostMem.fdf

. . .

* ^{1}…/ == “AdvancdFeaturePkg/Include”

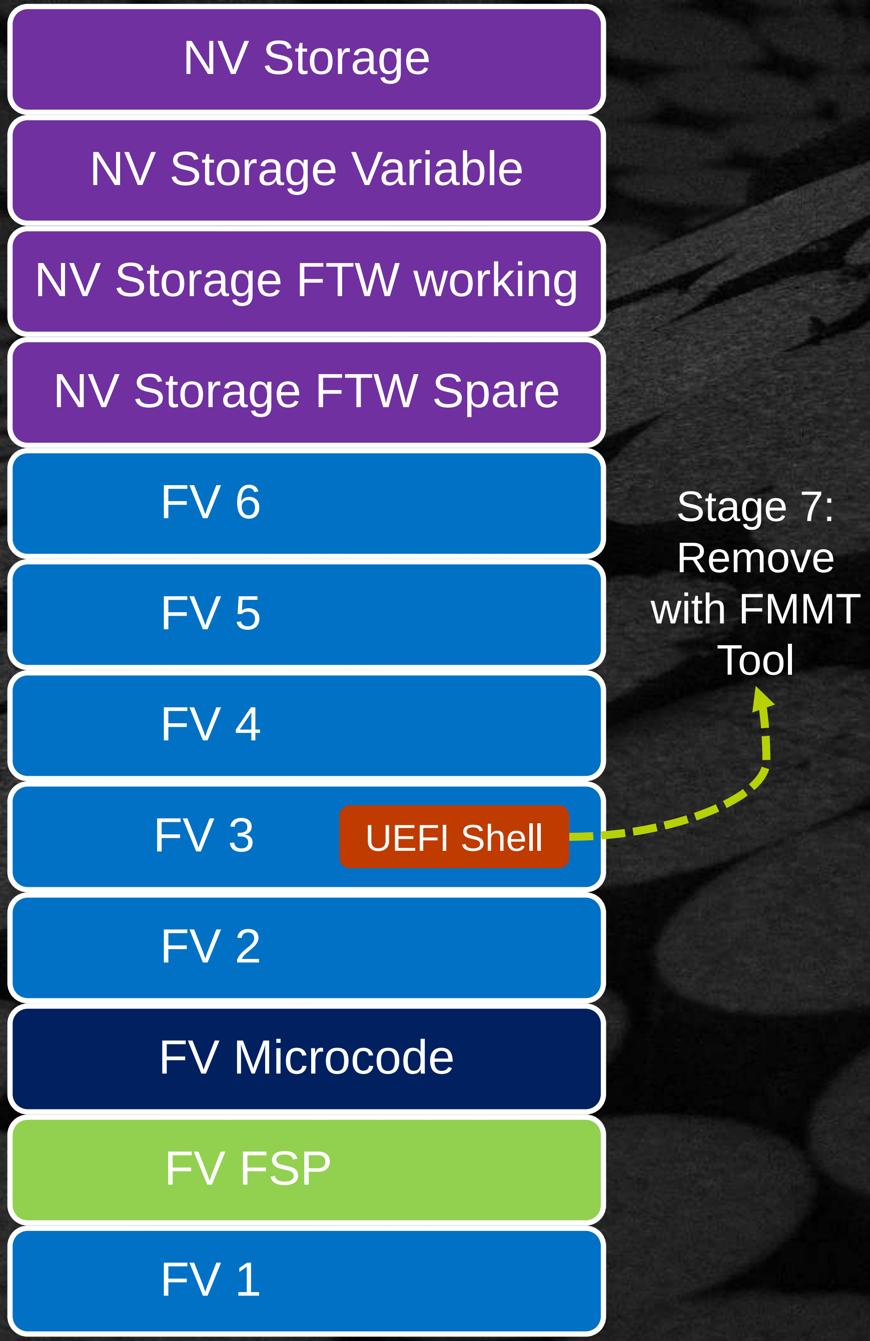
Optimization Feature Selection

Minimum Platform takes advantage of UEFI and EDK II features to enable feature selection to be done by post-processing the built binaries

Essentially, after your system is functioning well, you can remove features using the FMMT tool to remove the drivers that are included as you build up the desired functionality

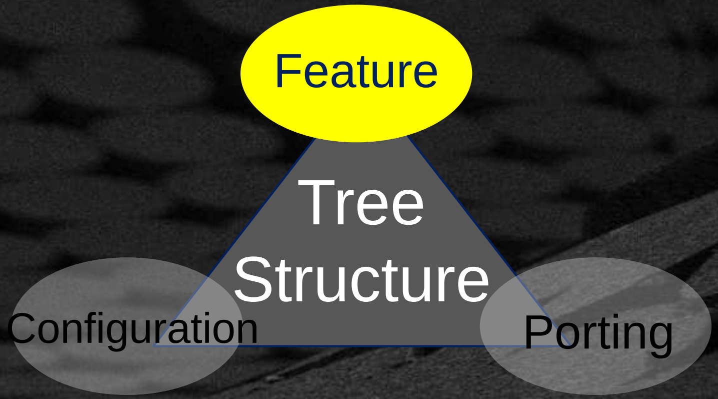
For example, if you need UEFI Shell during power-on, testing, etc. But you don't want it for final product. Minimum Platform architecture makes it easy to locate and remove the shell by post-processing the image

Link for FMMT Tool in BaseTools directory



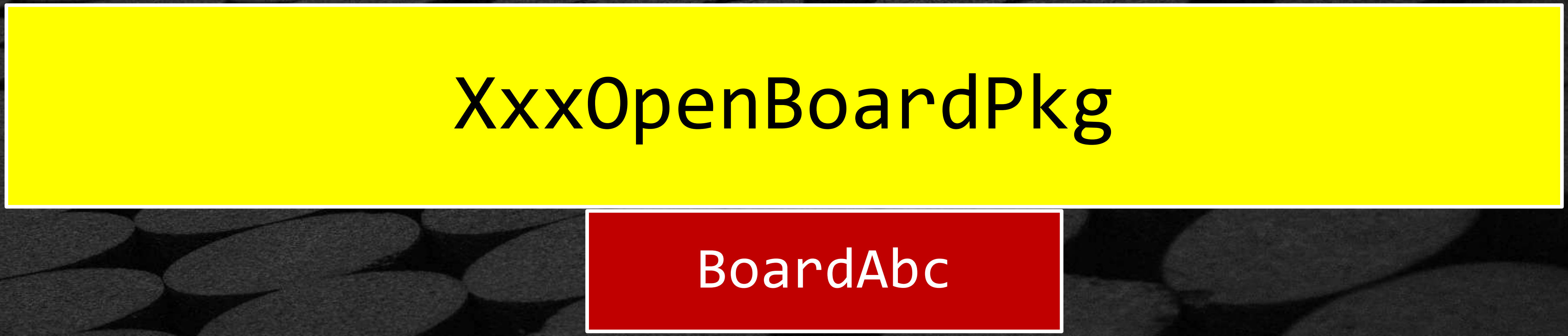
Flash Layout

Full Customization Feature Selection



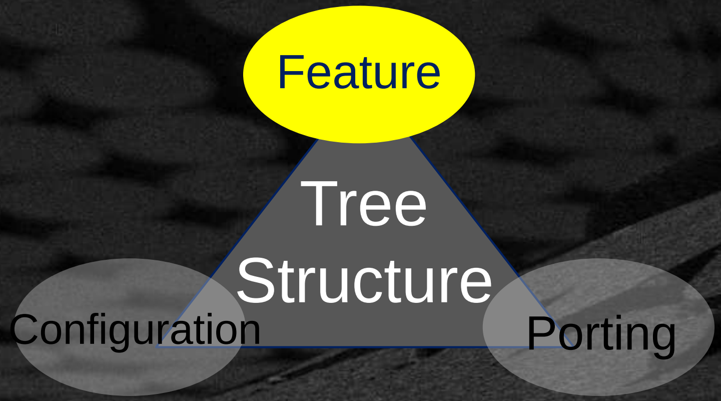
Feature modifications only at the Board / Platform DSC

Preferred modifications at Board (e.g. `BoardAbc`)



BoardAbc – directory for `OpenBoardPkg.dsc`

Features Build Enabled



Platform-Board Build Scripts

Many platforms have a script (Python or bash) to pre & post process the EDK II build process: Build Script

Example: Invoked from the `edk2-platforms/Platform/Intel`

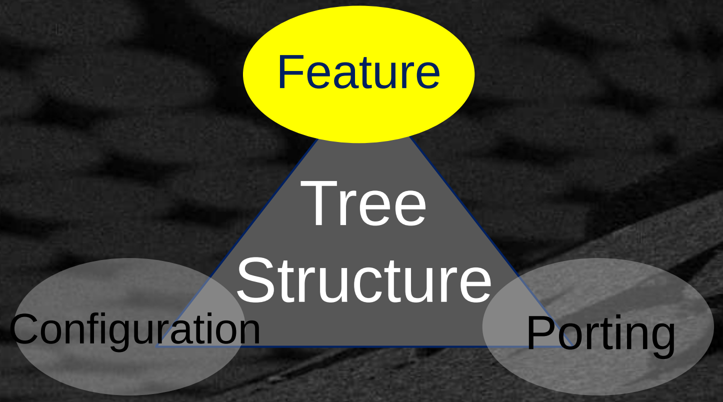
```
python build_bios.py –p <Board-name>
```

uses config file `build.cfg` from the `<Board-name>` directory

Configuration Files:

- edk2-platforms/Platform/Intel/build.cfg - default settings
- Default settings are under the DEFAULT_CONFIG section
- Override the `edk2-platforms/Platform/Intel/. . ./build.cfg` settings from each board in board specific directory

Example Build Config File



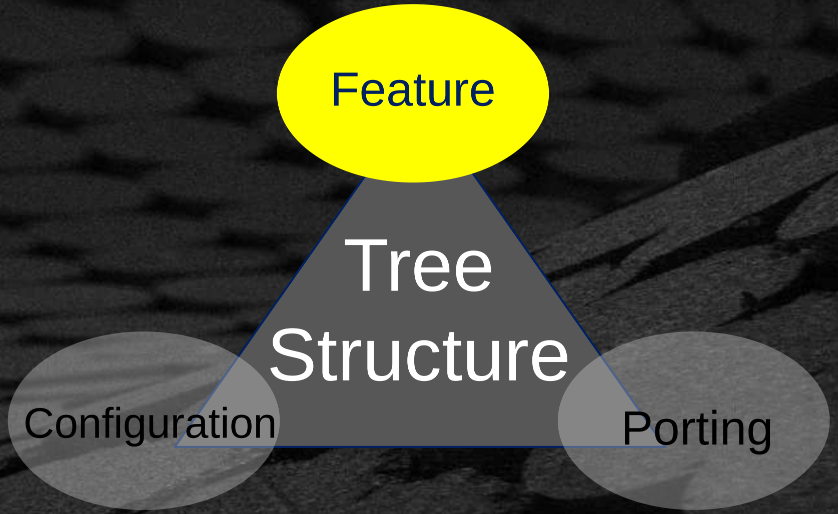
Kabylake example of Board specific settings:

```
<workspace>/edk2-platforms/Platform/Intel/KabylakeOpenBoardPkg/\
KabylakeRvp3/  build_config.cfg
```

```
[CONFIG]
WORKSPACE_PLATFORM_BIN = WORKSPACE_PLATFORM_BIN
EDK_SETUP_OPTION =
openssl_path =
PLATFORM_BOARD_PACKAGE = KabylakeOpenBoardPkg
PROJECT = KabylakeOpenBoardPkg/KabylakeRvp3
BOARD = KabylakeRvp3
FLASH_MAP_FDF = KabylakeOpenBoardPkg/Include/Fdf/FlashMapInclude.fdf
PROJECT_DSC = KabylakeOpenBoardPkg/KabylakeRvp3/OpenBoardPkg.dsc
BOARD_PKG_PCD_DSC =
KabylakeOpenBoardPkg/KabylakeRvp3/OpenBoardPkgPcd.dsc
ADDITIONAL_SCRIPTS =
KabylakeOpenBoardPkg/KabylakeRvp3/build_board.py
PrepRELEASE = DEBUG
SILENT_MODE = FALSE
...
```

Platform name & path to `build.cfg` file under `[PLATFORMS]`

Minimum Platform Stage Selection


Feature

Tree Structure

Configuration

Porting

Platform Firmware Boot Stage PCD :

OpenBoardPkgPcd.dsc

```
[PcdsFixedAtBuild]
  #
  # Please select BootStage here.
  # Stage 1 - enable debug (system deadloop after debug init)
  # Stage 2 - mem init (system deadloop after mem init)
  # Stage 3 - boot to UEFI shell only
  # Stage 4 - boot to OS
  # Stage 5 - boot to OS with security boot enabled
  # Stage 6 – Add Advanced features
  gMinPlatformPkgTokenSpaceGuid.PcdBootStage|4
```

tianocore

Minimum Platform Stage Selection

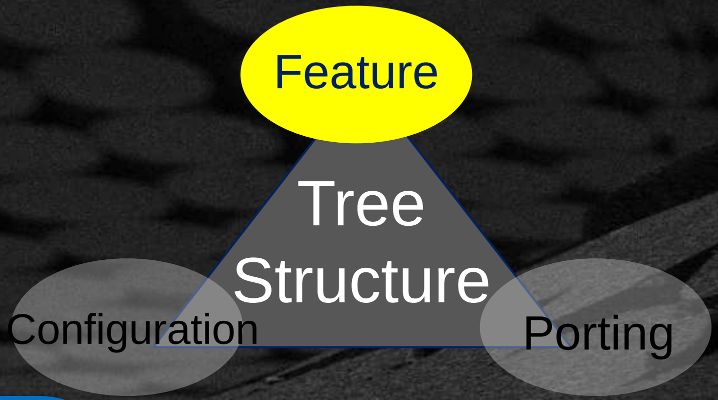
Stage selection allows developers to bring up a system in a familiar matter:

- Set to Stage I. Boot, verify serial debug capability, ready for silicon debug
- Set to Stage II. Boot, verify memory and silicon functionality
- Set to Stage III. Boot, verify board porting from shell: devices, GPIO, etc
- Set to Stage IV. Boot, verify ACPI porting, MADT, DSDT methods, etc.

Developers can exercise functionality gradually.

Verification at each stage with built in test points

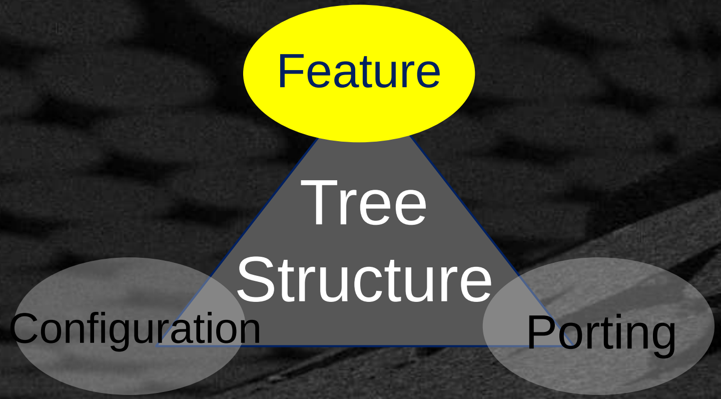
Required set of PCDs in MPA Spec



- Flash Map Config
- Debug Config
- Intel® FSP Config
- Post Memory FV
- UEFI FV
- Driver Related

- Memory Type Information
- OS FV
- Security Flash Map
- Stage 5 Features
- Advanced Feature FV

Build Control Files



DSC files

control what gets compiled and linked

FDF files

control what gets put in the system FLASH image

Where are the DSC & FDF files?

Kabylake Open Board

```
Platform/Intel/KabyLakeOpenBoardPkg/

  KabyLakeRvp3/


     OpenBoardPkgPcd.dsc    ← Modify PCD Here
     OpenBoardPkgBuildOption.dsc
     OpenBoardPkg.dsc       ← Add Features Here

     FlashMapInclude.fdf
     OpenBoardPkg.fdf       ← Add Features Here
```

```
/edk2-platforms/Platform/
  Intel/MinPlatformPkg/
    Include/
      Fdf/
      Dsc/

/edk2-platforms/Features/
  Intel/YyyAdvancedPkg/
    Include/
      Fdf/
      Dsc/
```

OpenBoardPkgPcd.dsc File Controls if feature ON or OFF

Example Kabylake Configuration .DSC file

```
[PcdsFixedAtBuild]
  #
  # Please select BootStage here.
  # Stage 1 - enable debug (system deadloop after debug init)
  # Stage 2 - mem init (system deadloop after mem init)
  # Stage 3 - boot to shell only
  # Stage 4 - boot to OS
  # Stage 5 - boot to OS with security boot enabled
  #
  gMinPlatformPkgTokenSpaceGuid.PcdBootStage|4

[PcdsFeatureFlag]
  gMinPlatformPkgTokenSpaceGuid.PcdStopAfterDebugInit|FALSE
  gMinPlatformPkgTokenSpaceGuid.PcdStopAfterMemInit|FALSE
  gMinPlatformPkgTokenSpaceGuid.PcdBootToShellOnly|FALSE
  gMinPlatformPkgTokenSpaceGuid.PcdUefiSecureBootEnable|FALSE
  gMinPlatformPkgTokenSpaceGuid.PcdTpm2Enable|FALSE

!if gMinPlatformPkgTokenSpaceGuid.PcdBootStage >= 1
  gMinPlatformPkgTokenSpaceGuid.PcdStopAfterDebugInit|TRUE
!endif
```

Link to OpenBoardPkgPcd.dsc
Confg .dsc file

Link to EDK II DSC Spec.

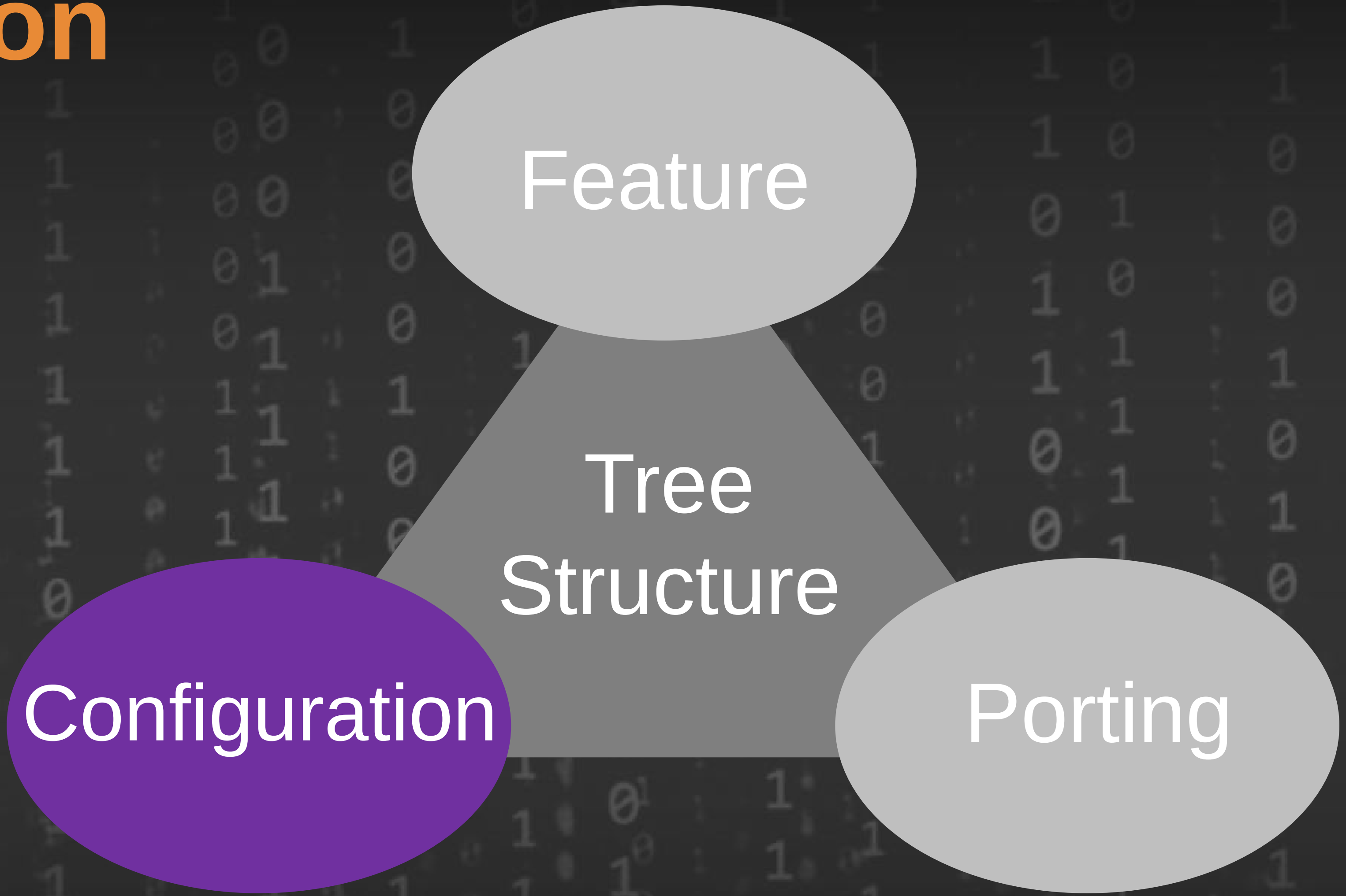
Example Kabylake .FDF file

```
[FV.FvPreMemory]
INF UefiCpuPkg/SecCore/SecCore.inf
INF MdeModulePkg/Core/Pei/PeiMain.inf
!include $(PLATFORM_PACKAGE)/Include/Fdf/CorePreMemoryInclude.fdf
INF $(PLATFORM_PACKAGE)/PlatformInit/PlatformInitPei/PlatformInitPreMem.inf
INF IntelFsp2WrapperPkg/FspmWrapperPeim/FspmWrapperPeim.inf
INF $(PLATFORM_PACKAGE)/PlatformInit/SiliconPolicyPei/SiliconPolicyPeiPreMem.inf
[FV.FvPostMemoryUncompact]
!include $(PLATFORM_PACKAGE)/Include/Fdf/CorePostMemoryInclude.fdf
# Init Board Config PCD
INF $(PLATFORM_PACKAGE)/PlatformInit/PlatformInitPei/PlatformInitPostMem.inf
INF IntelFsp2WrapperPkg/FspsWrapperPeim/FspsWrapperPeim.inf
INF $(PLATFORM_PACKAGE)/PlatformInit/SiliconPolicyPei/SiliconPolicyPeiPostMem.inf
!if gSiPkgTokenSpaceGuid.PcdPeiDisplayEnable == TRUE
FILE FREEFORM = 4ad46122-ffeb-4a52-bfb0-518cfca02db0 {
SECTION RAW = $(PLATFORM_FSP_BIN_PACKAGE)/SampleCode/Vbt/Vbt.bin
SECTION UI = "Vbt"
}
FILE FREEFORM = 7BB28B99-61BB-11D5-9A5D-0090273FC14D {
SECTION RAW = MdeModulePkg/Logo/Logo.bmp
}
```

Link to Kabylake .FDF

Link to EDK II FDF Spec

Configuration



- Incremental
- Simple PCD usage model
- No setup

Configuration Options

There might be many sources of platform configuration data.

PI PCD

Configuration Block

CMOS

UEFI Variable

Global NVS

MACRO

FSP UPD-
Silicon Policy
Hob/PPI/ Protocol

Platform signed data blob

MPA Configuration Options

Platform configuration data for Minimum Platform

PI PCD

- The PI PCD could be static data fixed at build time or dynamic data updatable at runtime.

FSP UPD- Silicon Policy Hob/PPI/ Protocol

- FSP UPD can be static default configuration, or a dynamic updatable UPD. It is policy data constructed at runtime or it can be a hook for silicon code

Global NVS

- ACPI region, passes configuration from C code to ASL code.

TIP: Use PCD Instead of UEFI Variable

UEFI Variable

```
//
// Get config from setup variable
//
VarDataSize = sizeof (SETUP_DATA);
Status = GetVariable (
     L"Setup",
     &gSetupVariableGuid,
     NULL,
     &VarDataSize,
     &mSystemConfiguration
);
```

PCD

```
//
// Get setup configuration from PCD
//
CopyMem (
     &mSystemConfiguration,
     PcdGetPtr (PcdSetupConfiguration),
     sizeof(mSystemConfiguration)
);
```

PCD Syntax Review

DEC

INF

DSC

PCD defined in the DEC file from any package

```
[Guids.common]
PcdTokenSpaceGuidName={0x914AEBE7, 0x4635, 0x459b, { 0xAA, . . .}}

 .   .   .

[Pcds...]
PcdTokenSpaceGuidName.PcdTokenName|Value[|DatumType[|MaxSize]]|Token
```

PCD usage listed in INF file for module

```
[…Pcd…]
PcdTokenSpaceGuidName.PcdTokenName|[Value]
```

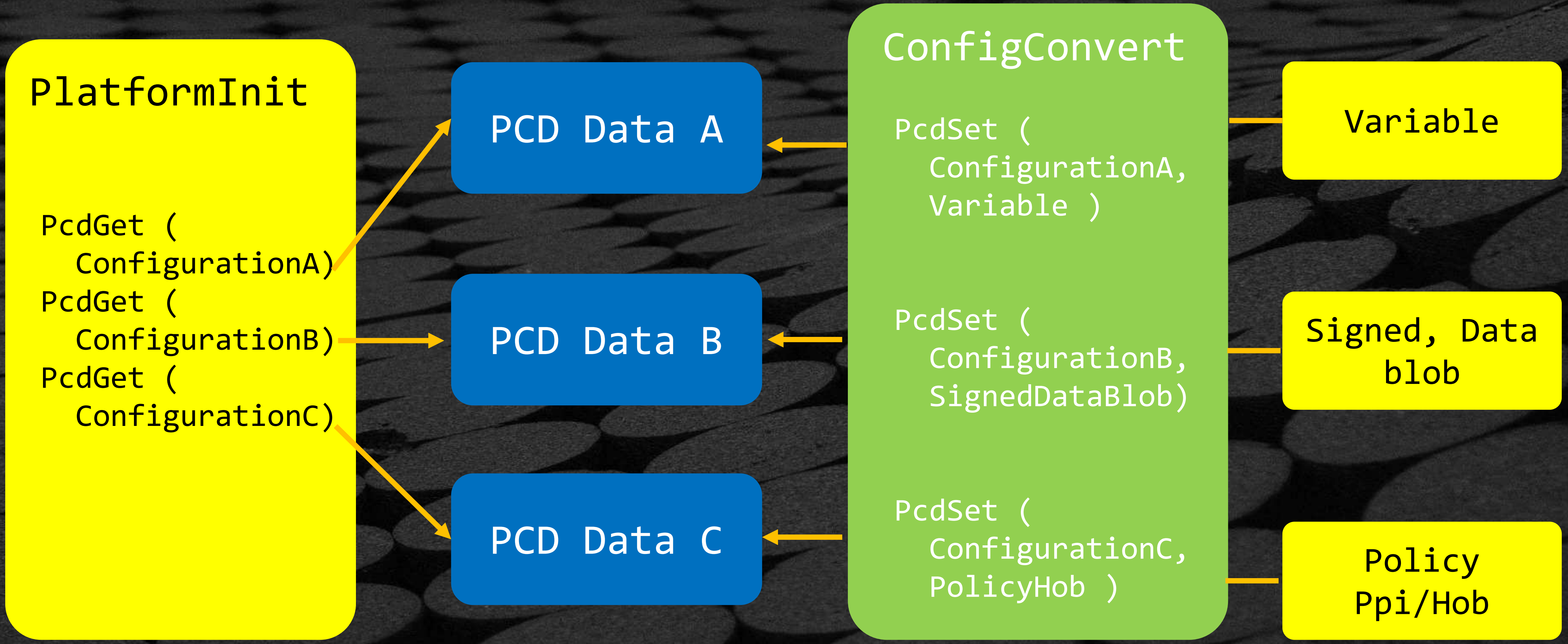
Value of PCD set in OpenBoardPkg...dsc

```
[Pcds...]
PcdTokenSpaceGuidName.PcdTokenName|Value[|DatumType[|MaximumDatumSize]]
```

How to Map PCD to Configuration Data

Using “Callback” mechanism to convert PCD to Configuration data

Platform driver should use `PcdGet()` to retrieve policy data, and `PcdSet()` to update policy data.



Source Open Source IA Firmware Platform Design Guide in EFI Developer Kit II.pdf Fig 7.

“C” Data Structure as PCDs

Example: `AdvancedFeaturePkg.dec` for SMBIOS type 0 data structure

```
gAdvancedFeaturePkgTokenSpaceGuid.PcdSmbiosType0BiosInformation| \
        {0x0}|SMBIOS_TABLE_TYPE0|0x80010000 {
    <HeaderFiles>
       IndustryStandard/SmBios.h
    <Packages>
       MdePkg/MdePkg.dec
       AdvancedFeaturePkg/AdvancedFeaturePkg.dec
}
gAdvancedFeaturePkgTokenSpaceGuid.PcdSmbiosType0BiosInformation.Vendor|0x1
gAdvancedFeaturePkgTokenSpaceGuid.PcdSmbiosType0BiosInformation.BiosVersion|0x2
gAdvancedFeaturePkgTokenSpaceGuid.PcdSmbiosType0BiosInformation.BiosSegment|0xF000
gAdvancedFeaturePkgTokenSpaceGuid.PcdSmbiosType0BiosInformation.BiosReleaseDate|0x3
gAdvancedFeaturePkgTokenSpaceGuid.PcdSmbiosType0BiosInformation.BiosSize|0xFF
gAdvancedFeaturePkgTokenSpaceGuid.PcdSmbiosType0BiosInformation.BiosCharacteristics.\
     PciIsSupported|1
gAdvancedFeaturePkgTokenSpaceGuid.PcdSmbiosType0BiosInformation.BiosCharacteristics.\
     PlugAndPlayIsSupported|1
```

Example of DSC xRef (.DEC & .h)

Purley Pkg DEC File

```
## gEfiSetupVariableGuid
OemSkuTokenSpaceGuid.PcdSetupData|{0x0}|\
SYSTEM_CONFIGURATION|0x000F0001 {
<HeaderFiles>
      Guid/SetupVariable.h
    <Packages>
      MdePkg/MdePkg.dec
      PurleyRcPkg/RcPkg.dec
      PurleySktPkg/SocketPkg.dec
      LewisburgPkg/PchRcPkg.dec
      PurleyOpenBoardPkg/PlatPkg.dec
  }
```

“C” SetupVariable.h File

```
. . .
   UINT8     FanPwmOffset;
   UINT8     WakeOnLanSupport;
   UINT8     Use1GPageTable;
   UINT8     CloudProfile;
} SYSTEM_CONFIGURATION;
```

StructureConfig.DSC File

```
gOemSkuTokenSpaceGuid.PcdSetupData.\
CloudProfile|0x0

gOemSkuTokenSpaceGuid.PcdSetupData.\
Use1GPageTable|0x1

gOemSkuTokenSpaceGuid.PcdSetupData.\
FanPwmOffset|0x0

gOemSkuTokenSpaceGuid.PcdSetupData.\
WakeOnLanSupport|0x0

. . .
```

Configuration Multi-SKU PCD – Board ID

DSC File – SKU Set at BUILD time

```
. . .
SKUID_IDENTIFIER = ?

[SkuIds]
0|DEFAULT
4|BoardX
0x42|BoardY

[PcdsDynamicDefault.common.BoardX]
gBoardModuleTokenSpaceGuid.PcdGpioPin|0x8
gBoardModuleTokenSpaceGuid.PcdGpioInitValue|\
        {0x00, 0x04, 0x02, 0x04, ...}

[PcdsDynamicDefault.common.BoardY]
gBoardModuleTokenSpaceGuid.PcdGpioPin|0x4
gBoardModuleTokenSpaceGuid.PcdGpioInitValue|\
        {0x00, 0x02, 0x01, 0x02, ...}
```

SKU PCD Set Dynamically

```
BoardXBoardDetect( VOID)
{
. . .
  if (LibPcdGetSku () != 0) {
    return EFI_SUCCESS;
  }
  if (IsBoardX ()) {
     LibPcdSetSku (BoardIdIsBoardX);
     ASSERT (LibPcdGetSku() ==
              BoardIdIsBoardX);
  }
  return EFI_SUCCESS;
}
```

Default Stores PCD – for Configuration

DSC File –

```
. . .
VPD_TOOL_GUID  = 8C3D856A-9 . . .

[DefaultStores]
0|STANDARD
1|MANUFACTURING
2|SAFE


[PcdsDynamicExVpd.common.DEFAULT]
  gEfiMdeModulePkgTokenSpaceGuid.PcdNvStoreDefaultValueBuffer|*
[PcdsDynamicEx.common.DEFAULT.STANDARD]
  gOemSkuTokenSpaceGuid.PcdSetupData.CloudProfile|0x0
  gOemSkuTokenSpaceGuid.PcdSetupData.Use1GPageTable|0x1
[PcdsDynamicEx.common.DEFAULT.MANUFACTURING]
  gOemSkuTokenSpaceGuid.PcdSetupData.CloudProfile|0x1
  gOemSkuTokenSpaceGuid.PcdSetupData.Use1GPageTable|0x0
```

- Special PCD to support the default stores concept in UEFI specification
- Can be Dynamically set

Silicon Policy Data Flow Guidelines

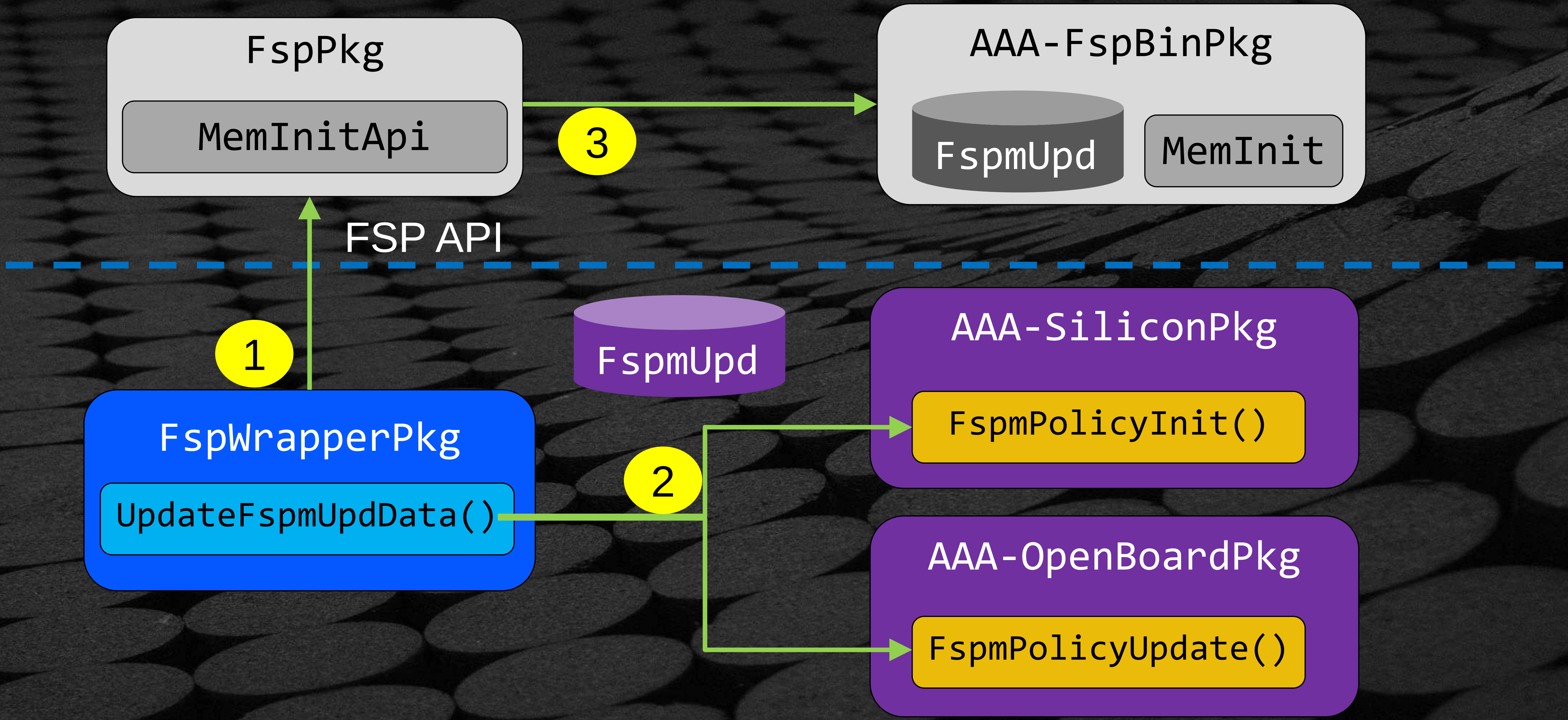
Silicon Module Provides Default Silicon Policy Data

- `Typedef` data structure

Board Module Updates the Silicon Policy Data

- PCD database, Setup Variable, Binary Blob, etc.

Example: FSP policy in MinPlatformPkg



Update Silicon Policy example

KabylakeOpenBoardPkg/FspWrapper/Library/PeiSiliconPolicyUpdateLibFsp

```
EFI_STATUS
EFIAPI
PeiFspSaPolicyUpdatePreMem (
IN OUT FSPM_UPD *FspmUpd
)
{
VOID *Buffer;
// Override MemorySpdPtr
CopyMem((VOID *)(UINTN)\
 FspmUpd->FspmConfig.MemorySpdPtr00,\
 (VOID *)(UINTN)PcdGet32 (PcdMrcSpdData), \
 PcdGet16 (PcdMrcSpdDataSize));
CopyMem((VOID *)(UINTN)\
 FspmUpd->FspmConfig.MemorySpdPtr10,\
 (VOID *)(UINTN)PcdGet32 (PcdMrcSpdData),\
 PcdGet16 (PcdMrcSpdDataSize));
```

```
 •   •   •
// Updating Dq Pins Interleaved,Rcomp Resistor &
// Rcomp Target Settings

 Buffer = (VOID *) (UINTN) PcdGet32 \
          (PcdMrcRcompTarget);
  if (Buffer) {
    CopyMem ((VOID *)\
      FspmUpd->FspmConfig.RcompTarget, \
      Buffer, 10);
  }
  return EFI_SUCCESS;
}
```

Link to file: PeiSaPolicyUpdatePrMem.c

Dynamically Set Defaults

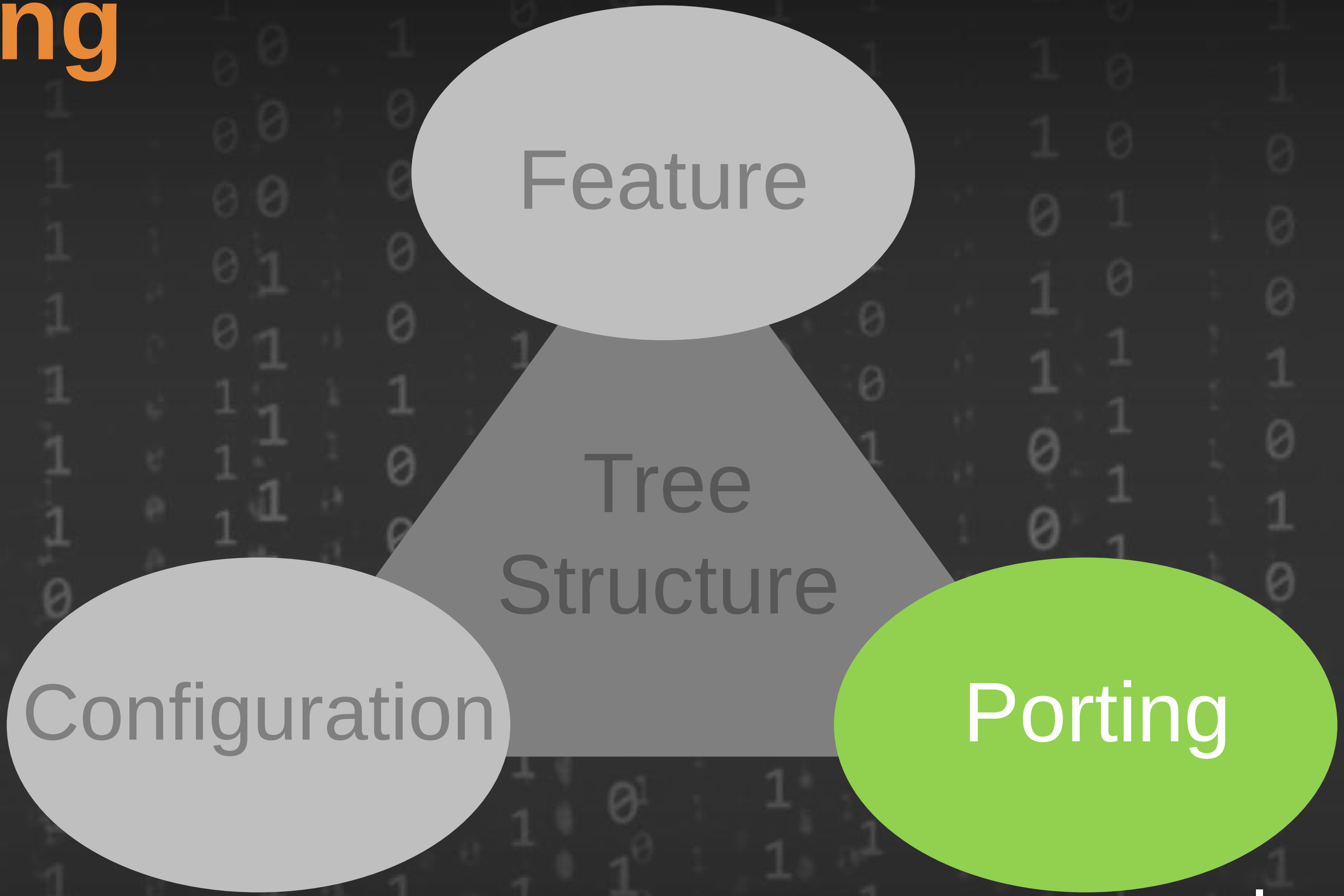
The Default Store PCD is also a dynamic PCD.

During boot, the board initialization code checks the boot mode and selects the default store.

This step must be after SetSku. Otherwise, the default setting may be wrong.

```
...
if (NeedDefaultConfig()) {
PcdSet16S (PcdSetNvStoreDefaultId, 0x0);
}
```

Board Porting



- Incremental
- Simple C libraries
- The same each time

Staged Approach by Features

- Platform Firmware Boot Stage PCD

PCD Variable:

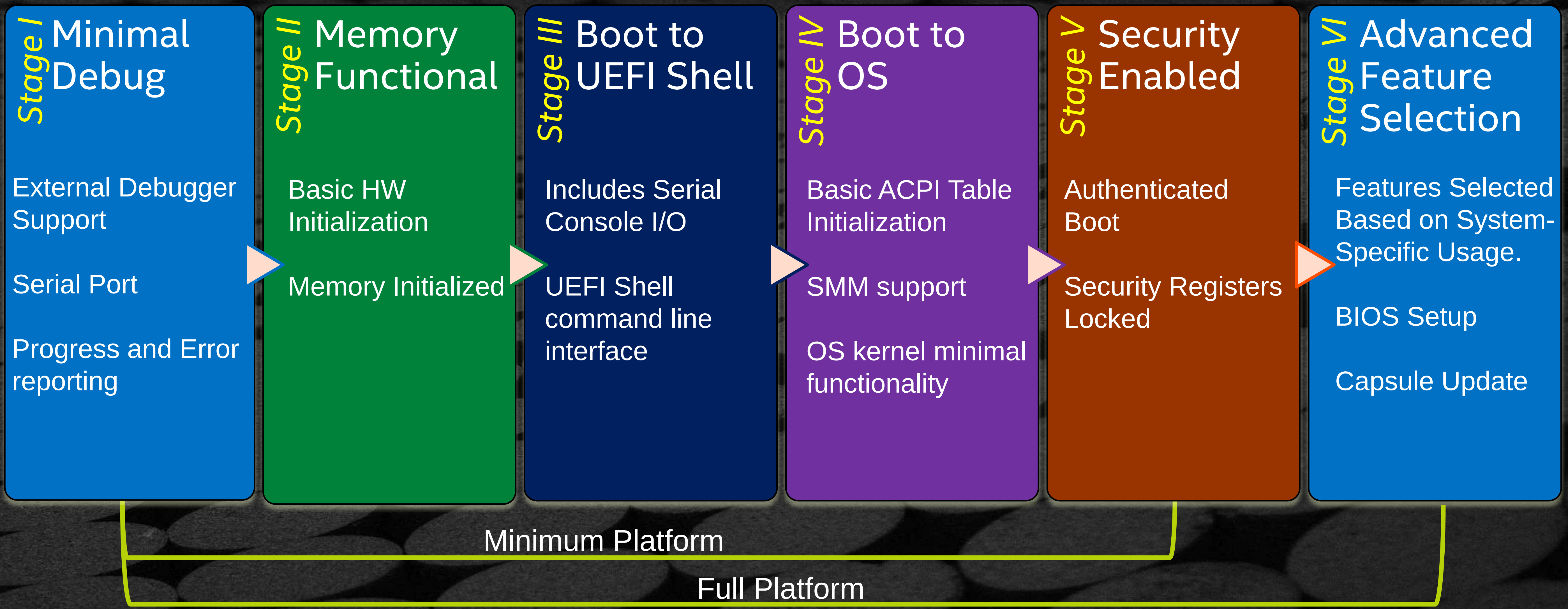
```
gPlatformModuleTokenSpaceGuid.PcdBootStage
```

Stage 1	enable debug
Stage 2	memory initialization
Stage 3	boot to UEFI shell only
Stage 4	boot to OS
Stage 5	boot to OS w/ security enabled
Stage 6	Advanced Feature Selection
Stage 7	Performance Optimizations

PCD Is tested within .FDF to see which modules to include

Minimum Platform + Intel® FSP Boot Flow

- Staged Approach



Stage I Minimal Debug

External Debugger Support

Serial Port

Progress and Error reporting

Stage II Memory Functional

Basic HW Initialization

Memory Initialized

Stage III Boot to UEFI Shell

Includes Serial Console I/O

UEFI Shell command line interface

Stage IV Boot to OS

Basic ACPI Table Initialization

SMM support

OS kernel minimal functionality

Stage V Security Enabled

Authenticated Boot

Security Registers Locked

Stage VI Advanced Feature Selection

Features Selected Based on System-Specific Usage.

BIOS Setup

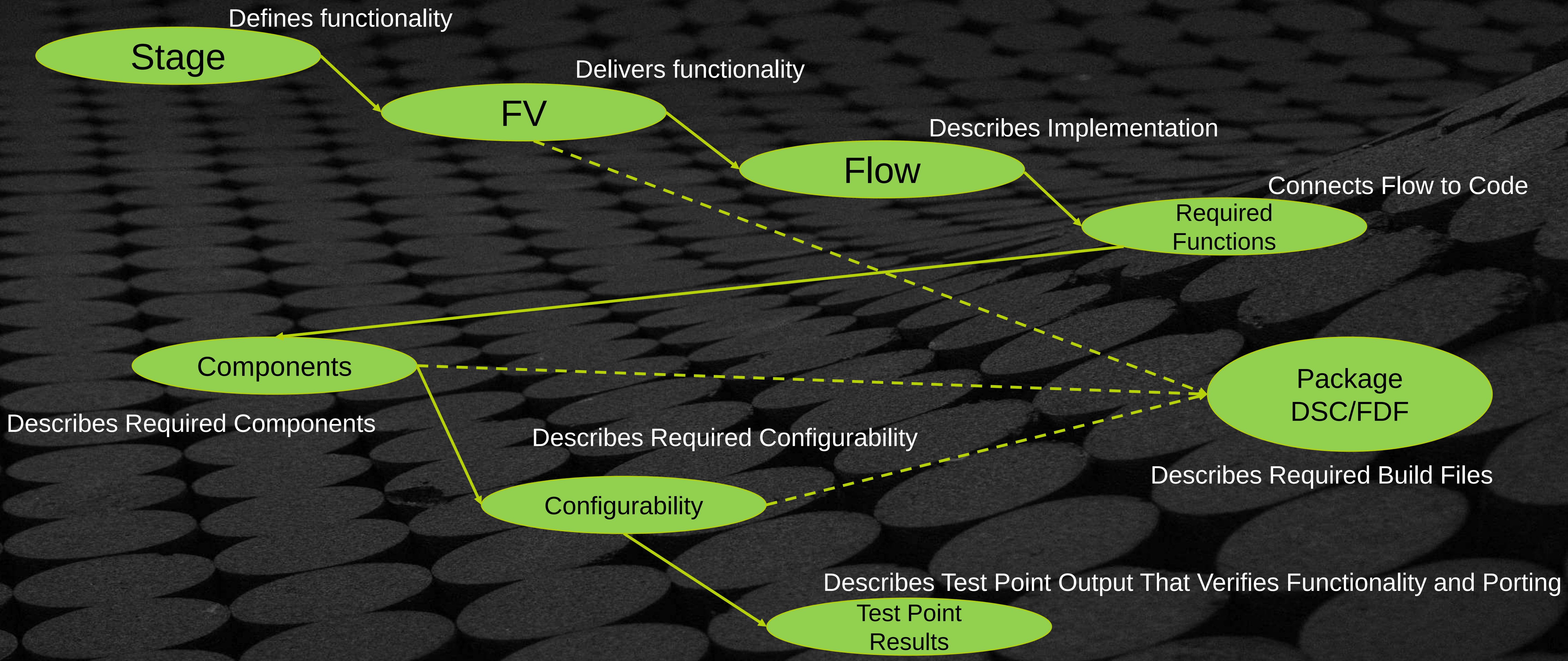
Capsule Update

Minimum Platform

Full Platform

tianocore

Stages Organize the MPA Specification



Defines functionality

Stage

Delivers functionality

FV

Describes Implementation

Flow

Connects Flow to Code

Required Functions

Components

Describes Required Components

Package DSC/FDF

Describes Required Configurability

Describes Required Build Files

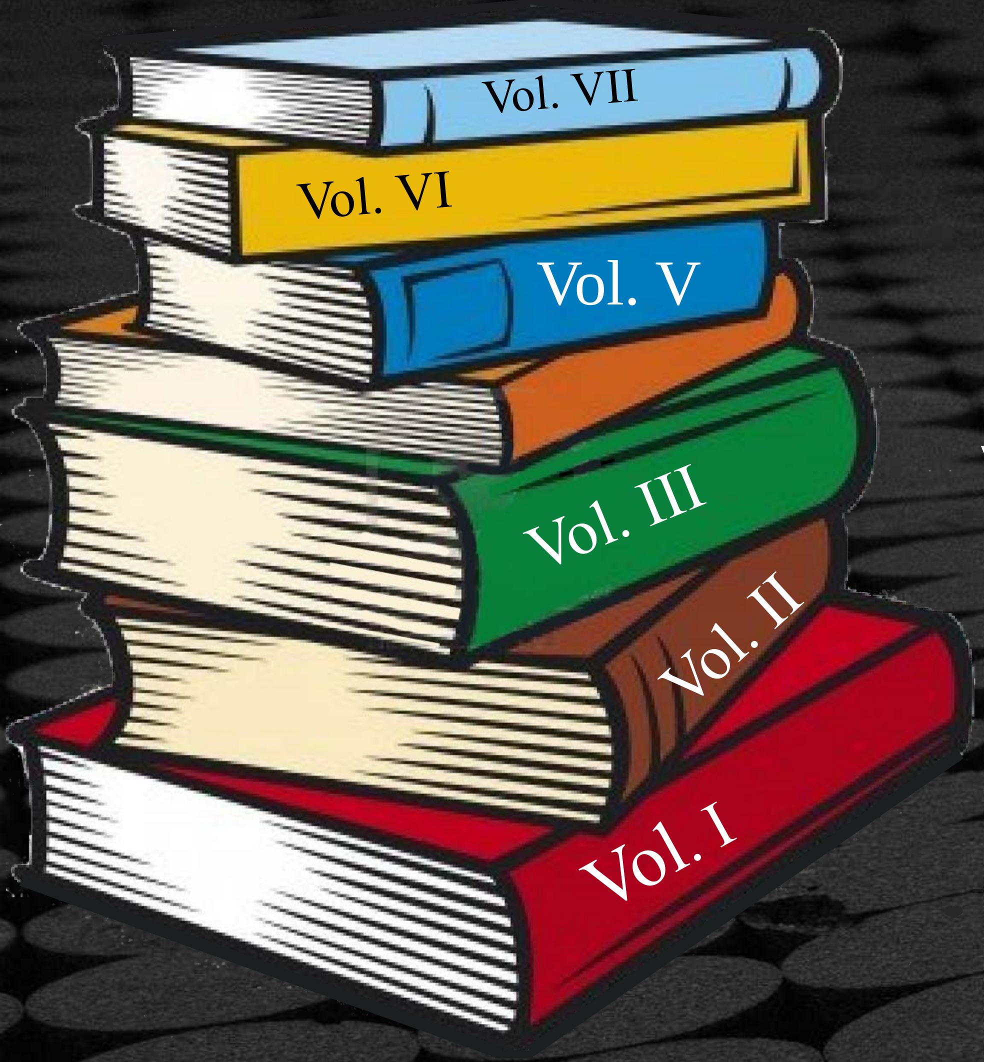
Configurability

Describes Test Point Output That Verifies Functionality and Porting

Test Point Results

Staged Approach by Features

- Firmware Volume

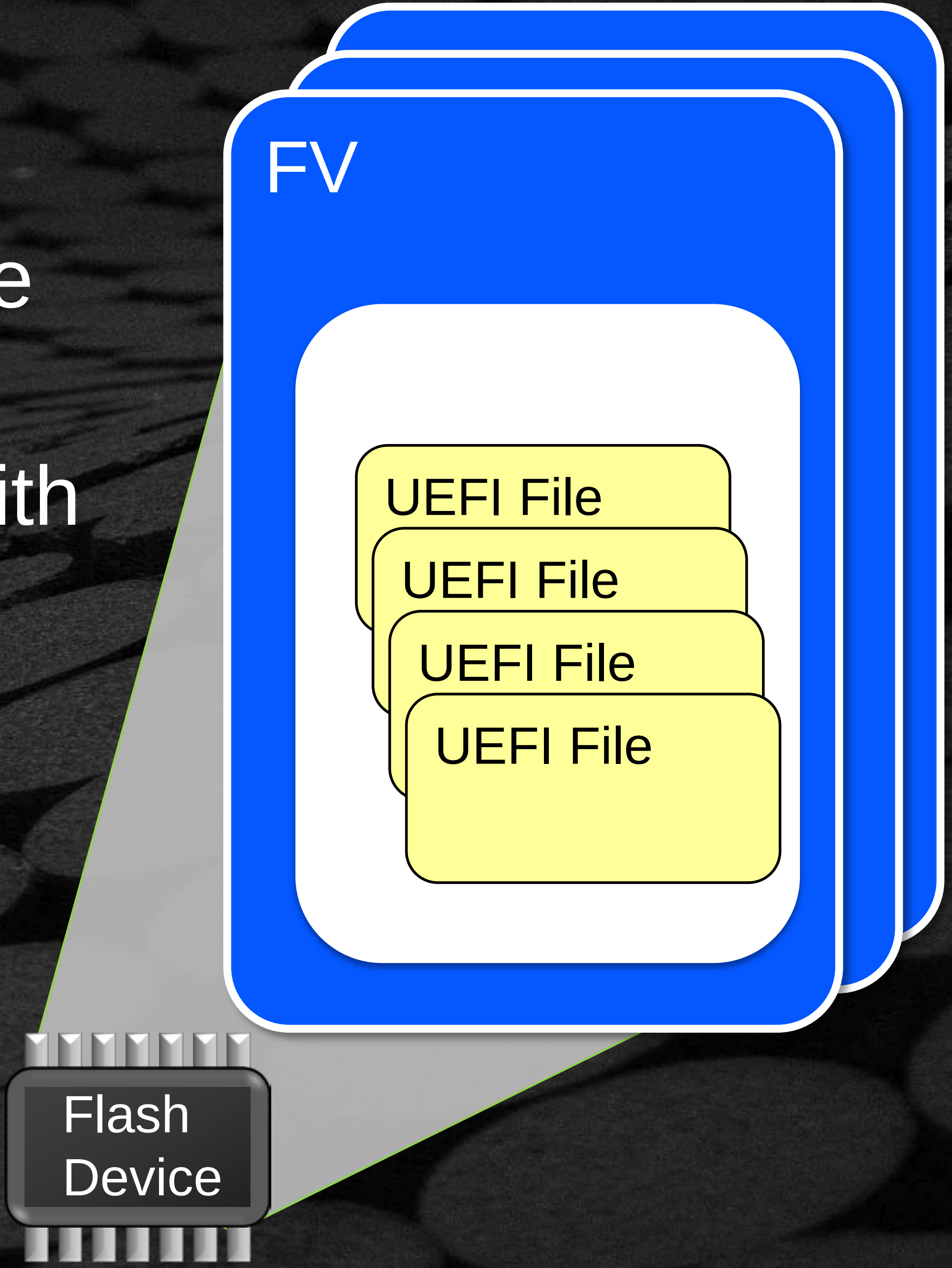


Modules organized by Firmware Volumes according to the different boot stages

UEFI Firmware Volumes (FV) - Review

Platform Initialization - Firmware Volume

- Basic storage repository for data and code is the Firmware Volume (FV)
- Each FV is organized into a file system, each with attributes
- One or more Firmware File Sections (FFS) files are combined into a FV
- Flash Device may contain one or more FVs.
- .FDF file controls the layout → .FD image(s)



PI Spec Vol 3

Standardize FV By Stages

Pre-Memory

- FvPreMemory – The PEIM dispatched before the memory initialization. Also included FSP - FV

Post Memory

- FvPostMemory – The PEIM dispatched after the memory initialization. Also included FSP - FV

UEFI Boot

- FvUefiBoot – The DXE driver supporting UEFI boot, such as boot to UEFI shell.

OS Boot

- FvOsBoot – The DXE driver supporting UEFI OS boot, such as UEFI Windows.

Security

- FvSecurity – The security related modules, such as UEFI Secure boot, TPM etc.

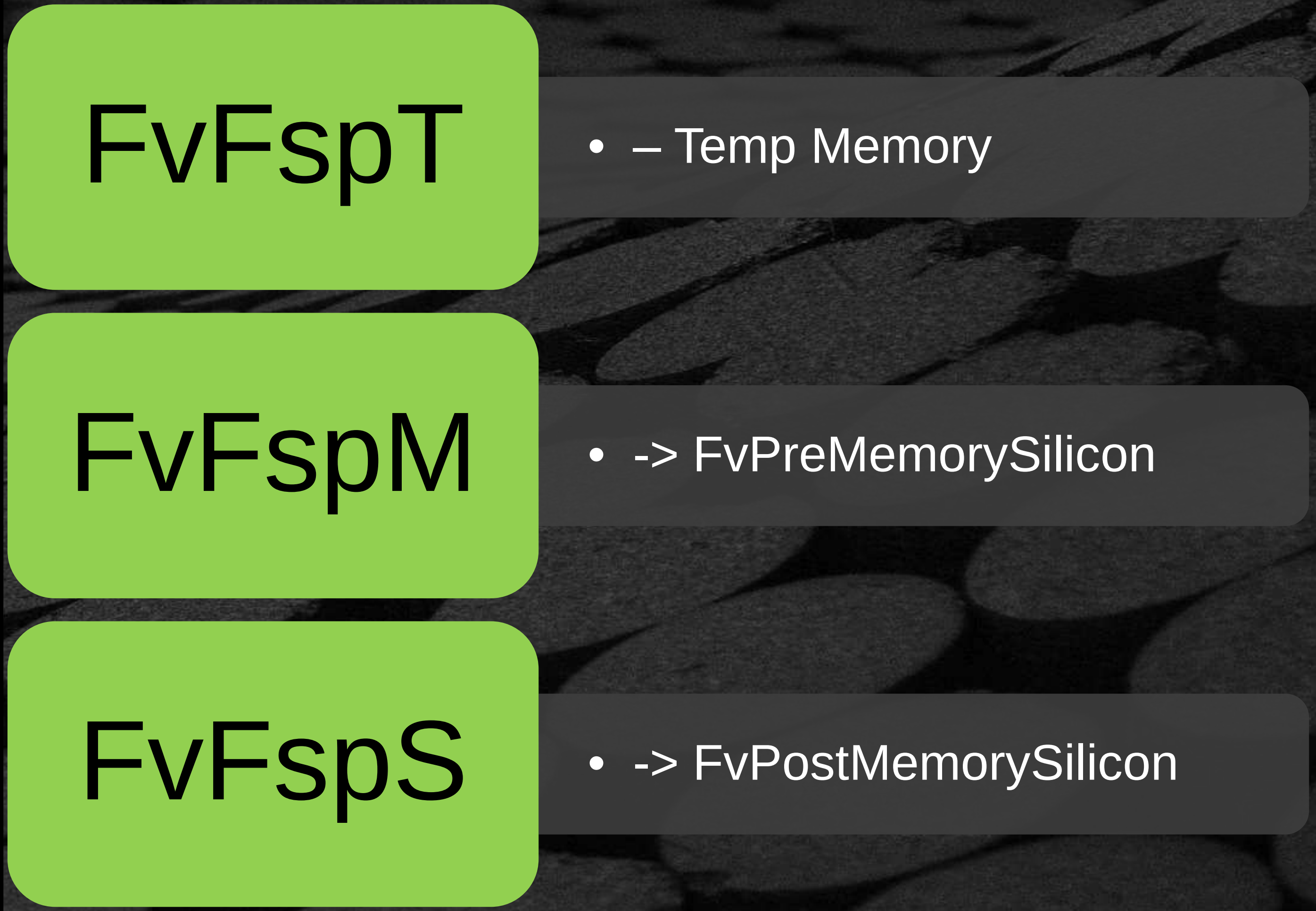
Advanced

- FvAdvanced – The advanced feature modules, such as UEFI network, IPMI etc.

Intel FSP Firmware Volumes
– created Pre-Build

```
MyWorkSpace/
      edk2/
         - “edk2 Common”
      edk2-platforms/
         Platform/Intel “Platform”
             KabyLakeOpenBoardPkg/
                include/fdf \
                    FlashMapInclude.fdf
                BoardXPkg/ “Board”
         Silicon/ “Silicon”
             Intel/MinPlatformPkg/
      edk2-non-osi/
         Silicon/Intel/
      FSP/
          BoardXPkg
             Fsp.fd
```

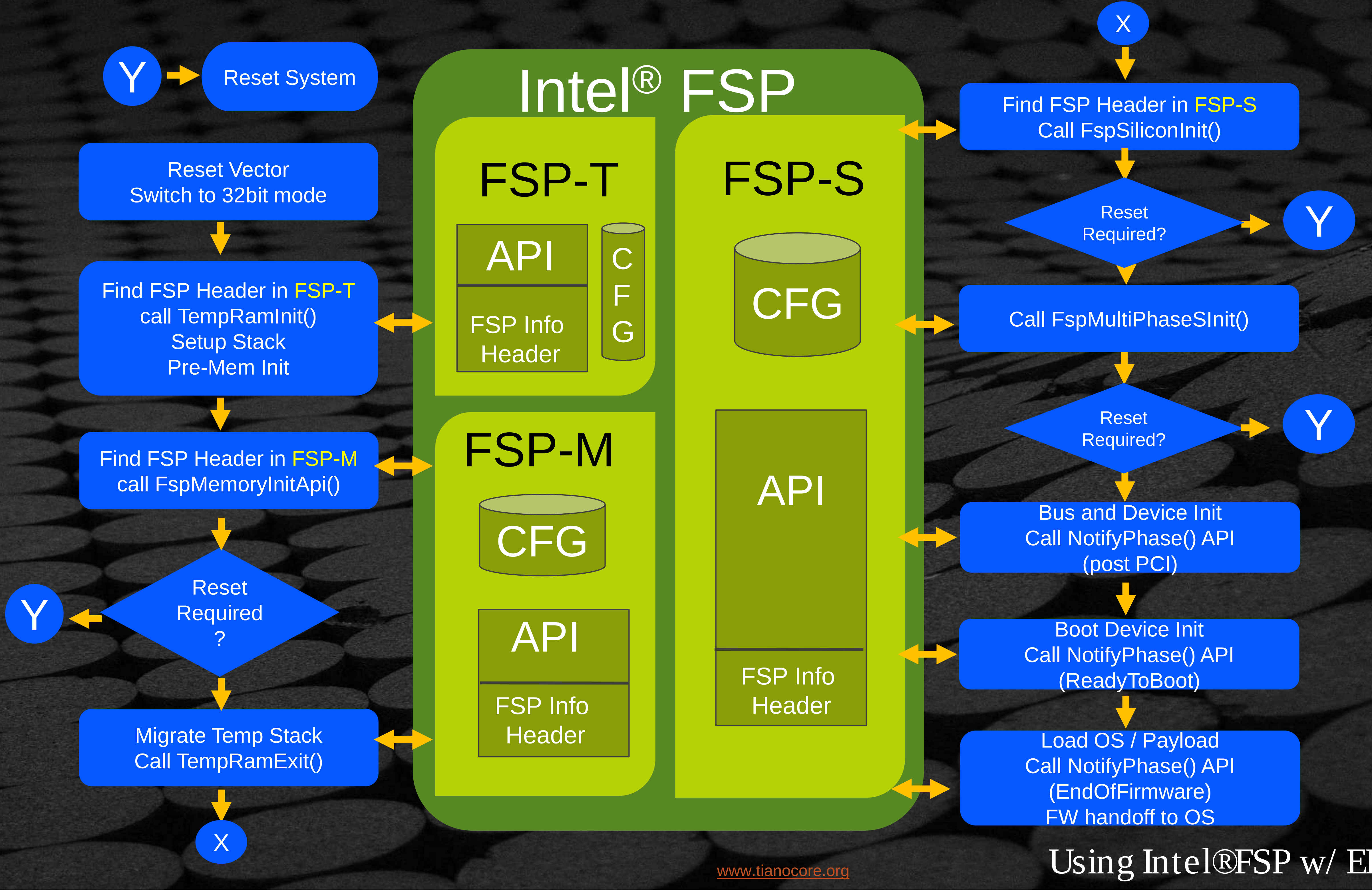
Fsp.fd Rebased for FVs



- FvFspT – – Temp Memory
- FvFspM – -> FvPreMemorySilicon
- FvFspS – -> FvPostMemorySilicon

Pre-Build w/
`RebaseAndPatchFspBinBaseAddress.py`

Intel FSP APIs in FSP Binary



Y → Reset System

Reset Vector
Switch to 32bit mode

Find FSP Header in FSP-T
call TempRamInit()
Setup Stack
Pre-Mem Init

Find FSP Header in FSP-M
call FspMemoryInitApi()

Reset Required ? → Y

Migrate Temp Stack
Call TempRamExit()

X

Intel® FSP

FSP-T: API, FSP Info Header, CFG

FSP-M: CFG, API, FSP Info Header

FSP-S: CFG, API, FSP Info Header

X

Find FSP Header in FSP-S
Call FspSiliconInit()

Reset Required? → Y

Call FspMultiPhaseSInit()

Reset Required? → Y

Bus and Device Init
Call NotifyPhase() API
(post PCI)

Boot Device Init
Call NotifyPhase() API
(ReadyToBoot)

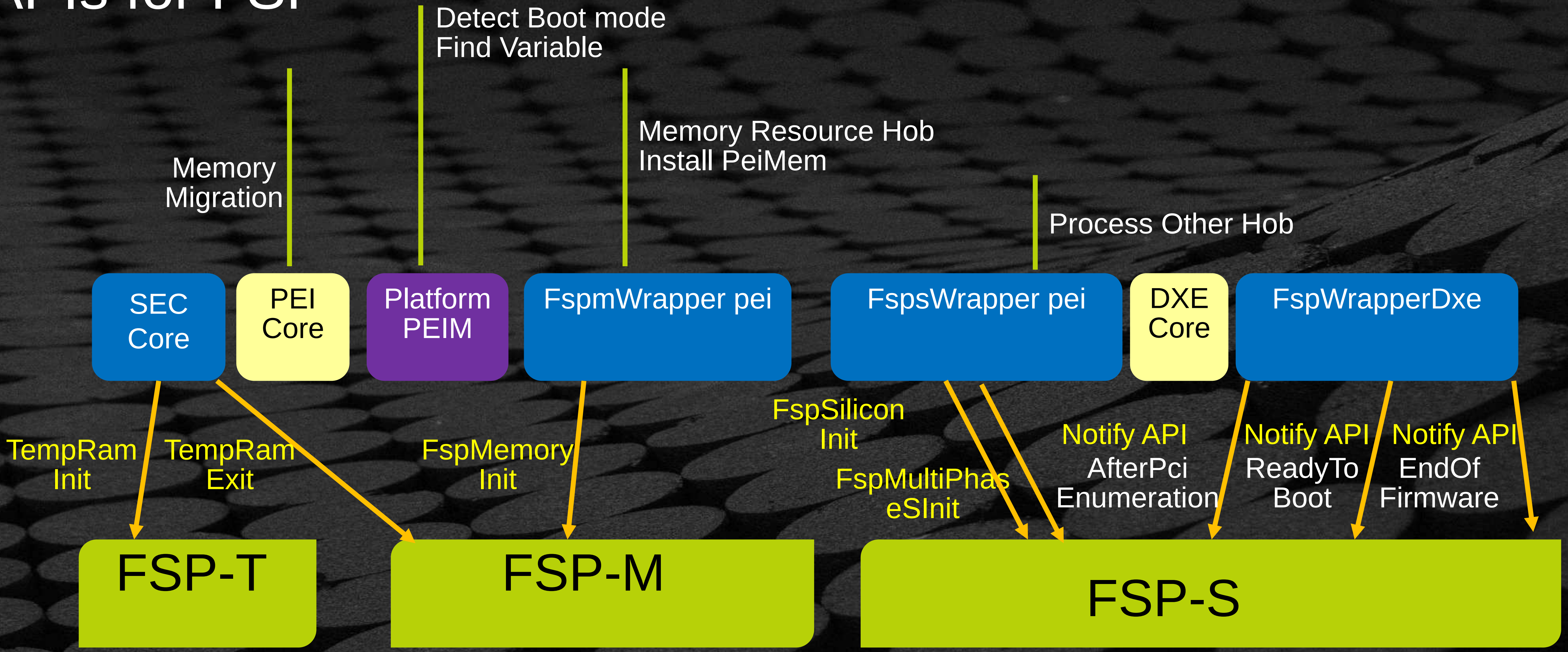
Load OS / Payload
Call NotifyPhase() API
(EndOfFirmware)
FW handoff to OS

🟩 Producer
🟦 Consumer

Using Intel®FSP w/ EDK II: PDF

Boot Flow with Intel FSP API Mode

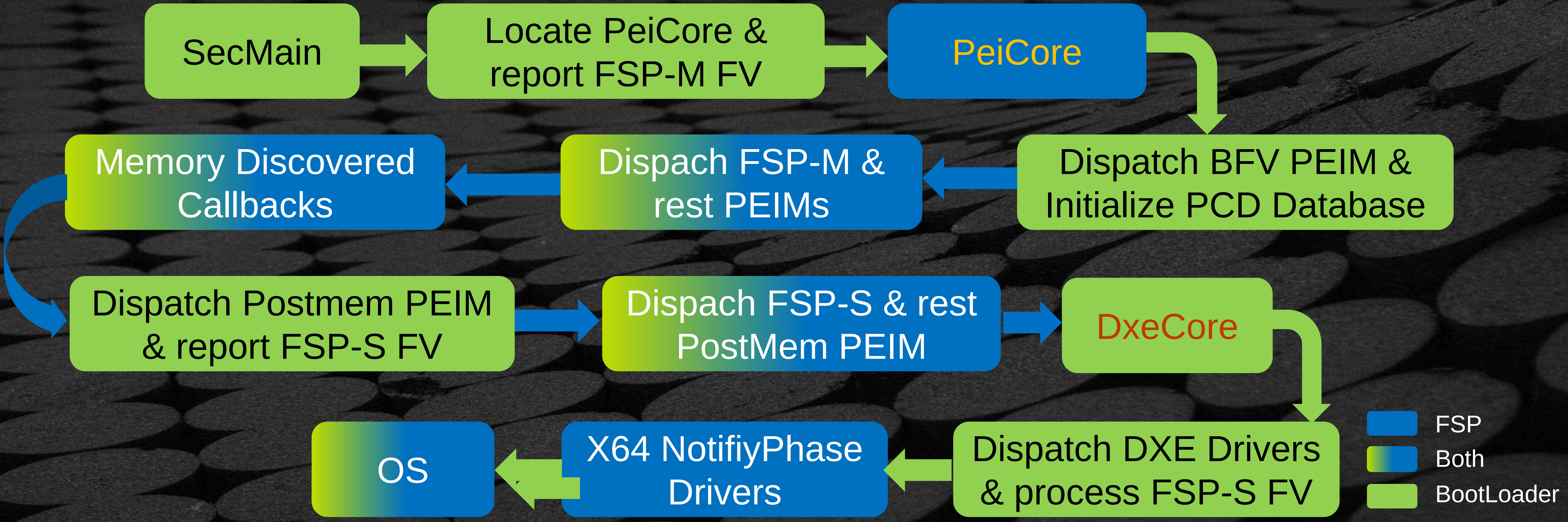
6 APIs for FSP



Original Source: Using the Intel® FSP with EDK II (2.0) Fig 4. – This now shows a 6 API added in FSP 2.2

Intel FSP 2.1 Dispatch Mode Boot Flow

`gIntelFsp2WrapperTokenSpaceGuid.PcdFspModeSelection` 0 - dispatch, 1 – API



SecMain

Locate PeiCore & report FSP-M FV

PeiCore

Dispatch BFV PEIM & Initialize PCD Database

Dispach FSP-M & rest PEIMs

Memory Discovered Callbacks

Dispatch Postmem PEIM & report FSP-S FV

Dispach FSP-S & rest PostMem PEIM

DxeCore

Dispatch DXE Drivers & process FSP-S FV

X64 NotifiyPhase Drivers

OS

🟦 FSP
▨ Both
🟩 BootLoader

FSP Spec 2.1

Dispatch Mode Interface

- Optional boot flow intended to enable Intel FSP to integrate well in to UEFI bootloader implementations.
- Conforms to UEFI & PI Specifications
- The FSP-T, FSP-M, and FSP-S are containers that expose firmware volumes (FVs) directly to the bootloader.
- UPD Mechanism to pass Config data is not needed
- PCD Database Required

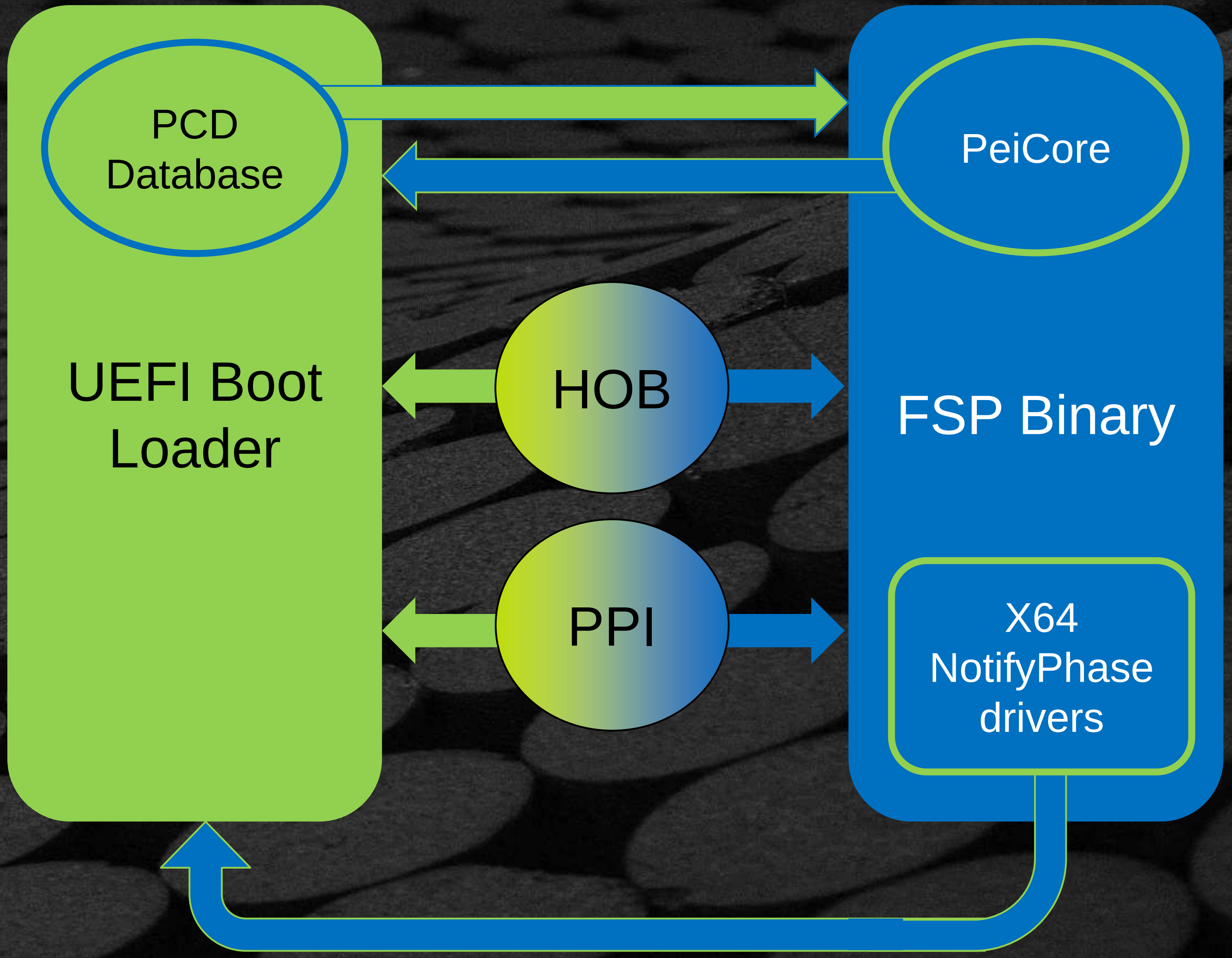


Figure 6 FSP Spec 2.2

PLATFORM HOOKS

Using EDK II Libraries

EDK II Libraries w/ Platform Hooks

DSC maps library class to library-instances

Syntax in DSC file

```
[libraryclasses]
LibraryClassName|Path/To/LibInstanceNameInstance1.inf
```

Search INF files for string: “**LIBRARY_CLASS =**”

Library Classes Section in DSC

DebugLib class example

Library Class Section

```
[LibraryClasses]
    DebugLib|MdePkg/Library/BaseDebugLibNull/BaseDebugLibNull.inf
    ...
[LibraryClasses.common.DXE_CORE]
    ...
    DebugLib|IntelFrameworkModulePkg/Library/PeiDxeDebugLibReportStatusCode/
           PeiDxeDebugLibReportStatusCode.inf
    ...
[LibraryClasses.common.DXE_SMM_DRIVER]
    DebugLib|MdePkg/Library/BaseDebugLibNull/BaseDebugLibNull.inf
```

Components Section

```
[Components]
   ...
MyPath/MyModule.inf {
<LibraryClasses>
    DebugLib|MdePkg/Library/BaseDebugLibSerialPort.inf
}
```

Platform Initialization Board Hook Modules

```
MinPlatformPkg/
  Include/
     Library/
       BoardInitLib.h
  Library/
  . . .
  PlatformInit/
    PlatformInitPei/
      PlatformInitPreMem/
      PlatformInitPostMem/
    PlatformInitDxe/
    PlatformInitSmm/
```

```
BoardDetect()
BoardDebugInit()
BoardBootModeDetect()
BoardInitBeforeMemoryInit()
BoardInitBeforeTempRamExit()
BoardInitAfterTempRamExit()
BoardInitAfterMemoryInit()
BoardInitBeforeSiliconInit()
```

PEI

```
BoardInitAfterPciEnumeration()
BoardInitReadyToBoot()
BoardInitEndOfFirmware()
```

DXE

Platform Initialization Board Hook Modules

```
MinPlatformPkg/
  . . .
  PlatformInit/
   PlatformInitPei/
    PlatformInitPreMem/






     PlatformInitPostMem/
```

```
PlatformInitPreMem/
  BoardDetect()
  BoardDebugInit()
  BoardBootModeDetect()
  BoardInitBeforeMemoryInit()
. . .
Notify call back
  BoardInitAfterMemoryInit()
```

PEI

```
PlatformInitPostMem/
  BoardInitBeforeSiliconInit()
. . .
  BoardInitAfterSiliconInit()
```

How to find the Platform Hooks: Process of Porting

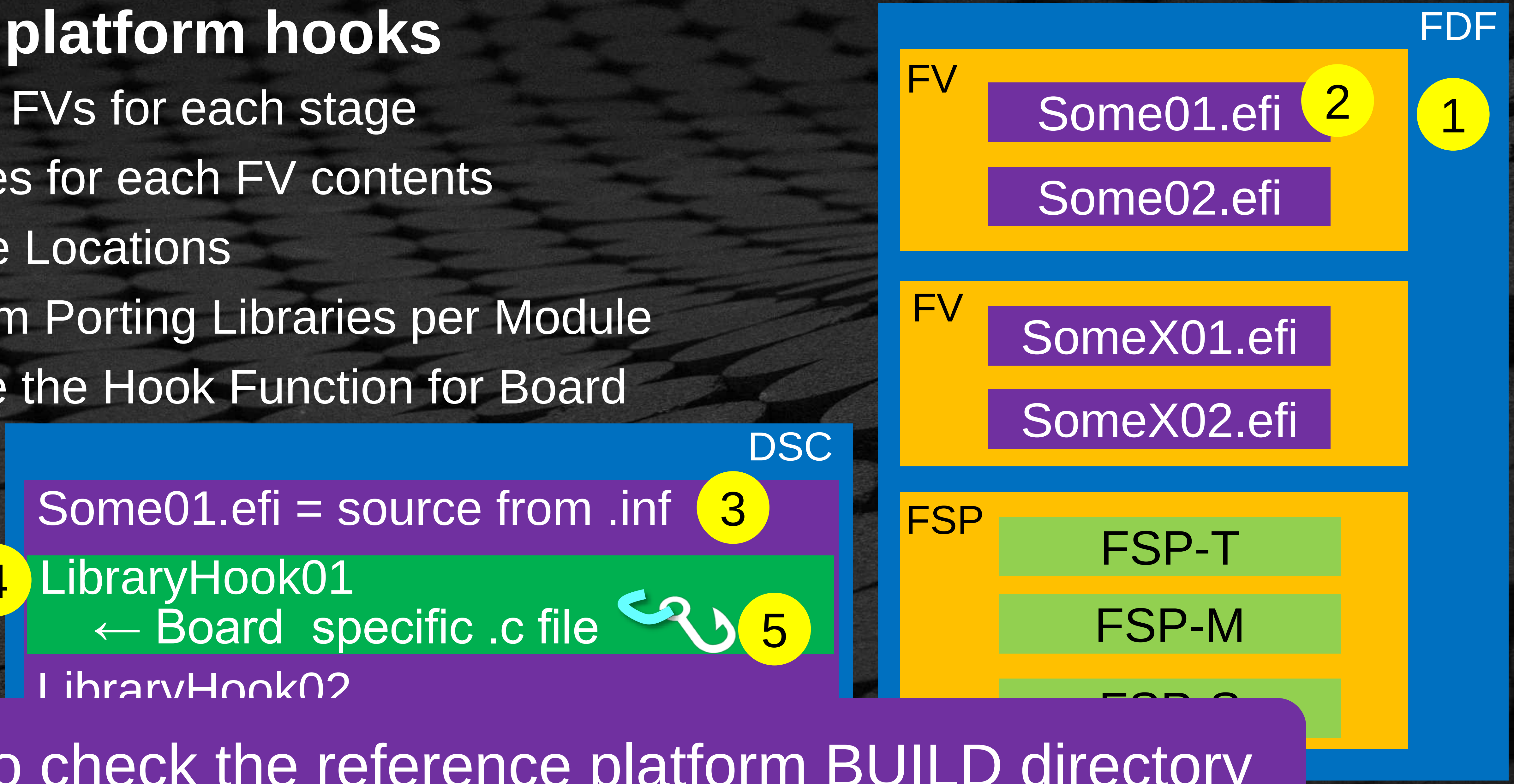
Where's the platform code

Check the Board/Platform .FDF file layout

Investigate the FDF then DSC files

Porting process per stage find and update platform hooks

- ❶ Locate FVs for each stage
- ❷ Modules for each FV contents
- ❸ Module Locations
- ❹ Platform Porting Libraries per Module
- ❺ Update the Hook Function for Board



Also check the reference platform BUILD directory

How to search for Libraries in the Workspace

1. Search the workspace .DSC files for the string of the library
2. Open the .DSC files associated with the open board platform project
3. Determine which Library is used and that should have the build path in the workspace
4. DSC file will have similar to:

```
SomeLib|Path_to_the_Library_used.inf
```

5. Verify the instance used from the Build directory

Platform Initialization Board Hook Modules - Stage 1

```
MinPlatformPkg/
  Include/
     Library/
       BoardInitLib.h ← // hooks
  Library/
  . . .
  PlatformInit/
    PlatformInitPei/
      PlatformInitPreMem/
```

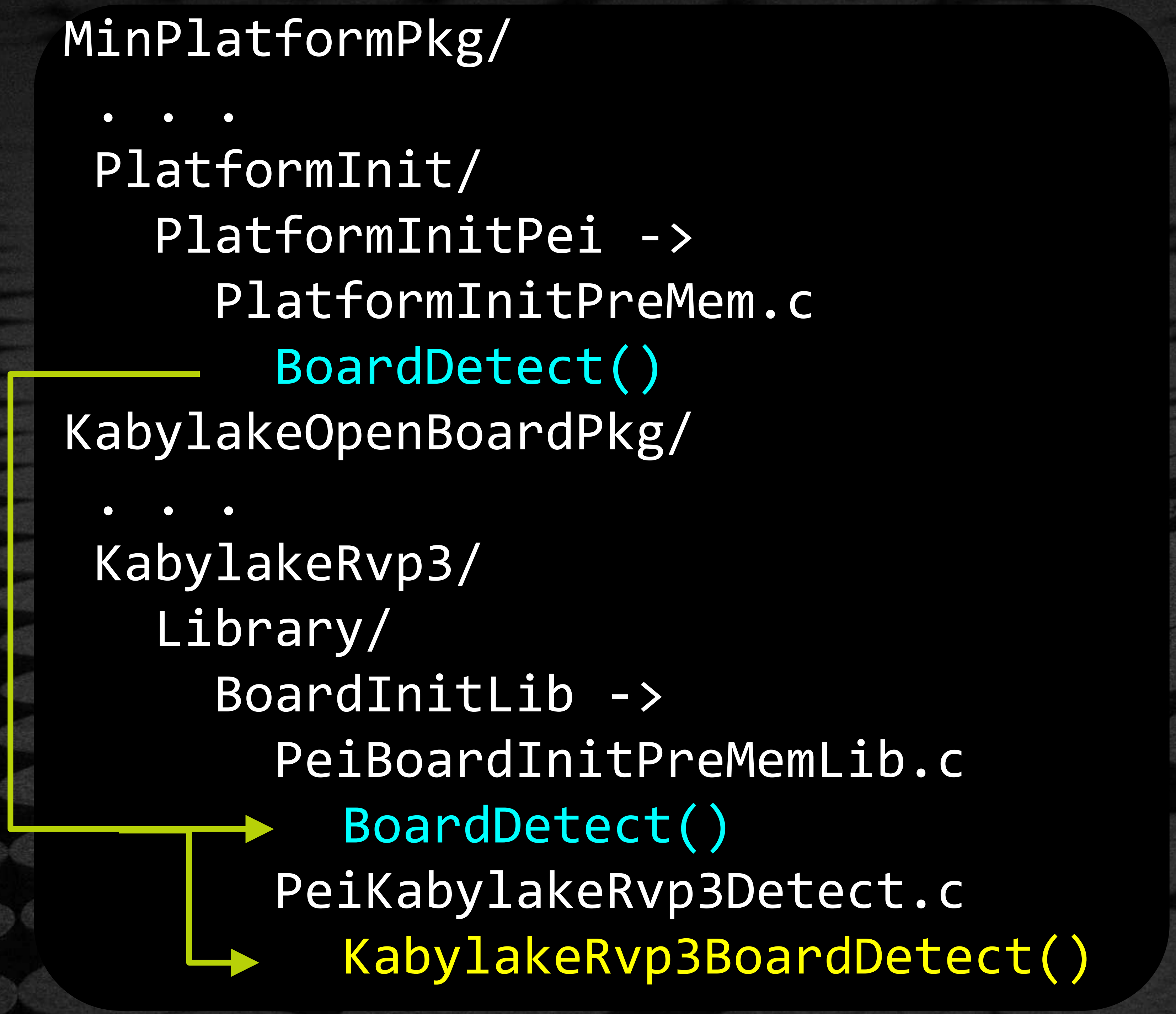
```
BoardDetect()
BoardDebugInit()
BoardBootModeDetect()
BoardInitBeforeMemoryInit()
```

Platform folder `PlatformInit` controls the platform initialization flow

Link: BoardInitLib.h

Example Hook - Board Detection

-Kabylake example



```
MinPlatformPkg/
  . . .
  PlatformInit/
    PlatformInitPei ->
      PlatformInitPreMem.c
        BoardDetect()
KabylakeOpenBoardPkg/
  . . .
  KabylakeRvp3/
    Library/
      BoardInitLib ->
        PeiBoardInitPreMemLib.c
          BoardDetect()
        PeiKabylakeRvp3Detect.c
          KabylakeRvp3BoardDetect()
```

Uses PCD Library calls to set / get Board SKU for Storing Board ID

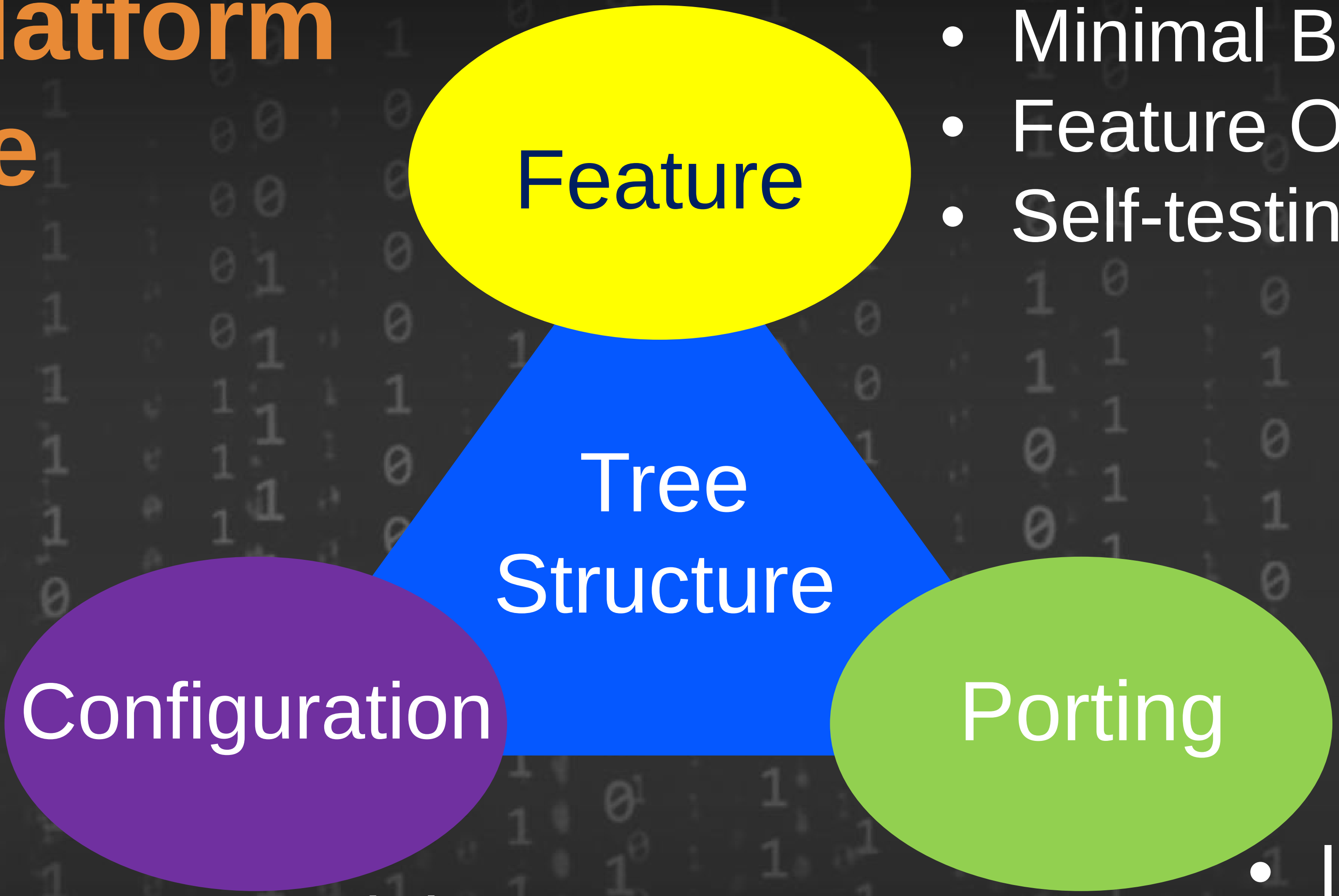
`LibPcdGetSku() & LibPcdSetSku()`

`KabylakeRvp3BoardDetect()` function reads Board ID from embedded controller (EC) using the LPC bus

`LibPcdSetSku()` stores Board ID

`LibPcdGetSku()` used from that point on

Minimum Platform Architecture Summary



- Minimal Baseline
- Feature ON/OFF
- Self-testing

- Incremental
- Simple PCD usage model
- No setup

- Incremental
- Simple C libraries
- The same each time

SUMMARY

- Minimum Platform Architecture (MPA) is an Open source Intel platform code base for use with EDK II
- EDK II Minplatform’s infrastructure focus areas: Tree, Features, Configuration & Porting
- MinPlatform uses Intel® FSP for processor, silicon and memory init & uses silicon policy guild lines for data flow

Questions?

Return to Main Training Page

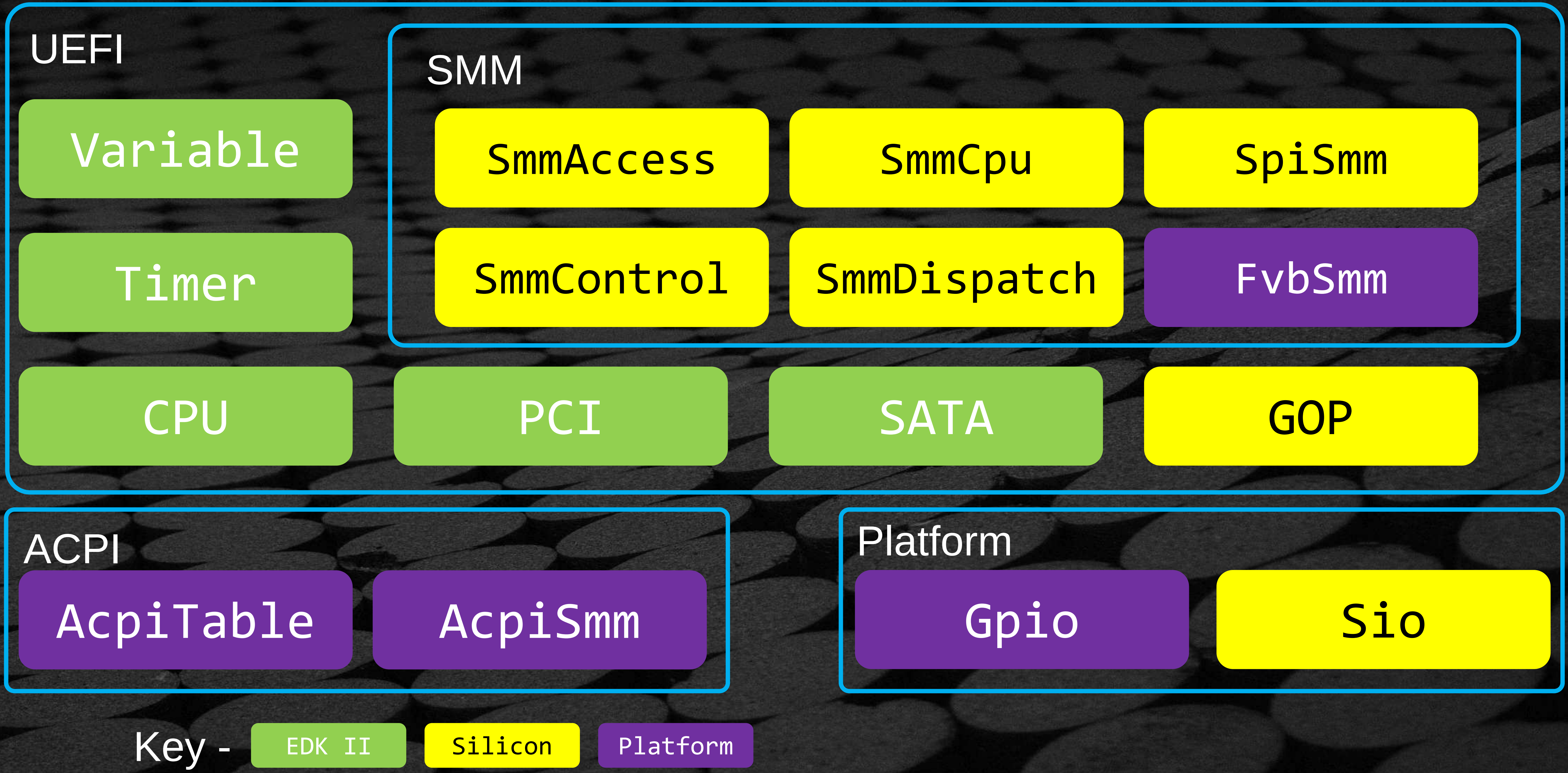
Return to Training Table of contents for next presentation link

tianocore

ACKNOWLEDGEMENTS

BACK UP

Basic Boot Components



UEFI

- 🟩 Variable
- 🟩 Timer
- 🟩 CPU
- 🟩 PCI
- 🟩 SATA
- 🟨 GOP

SMM

- 🟨 SmmAccess
- 🟨 SmmCpu
- 🟨 SpiSmm
- 🟨 SmmControl
- 🟨 SmmDispatch
- 🟪 FvbSmm

ACPI

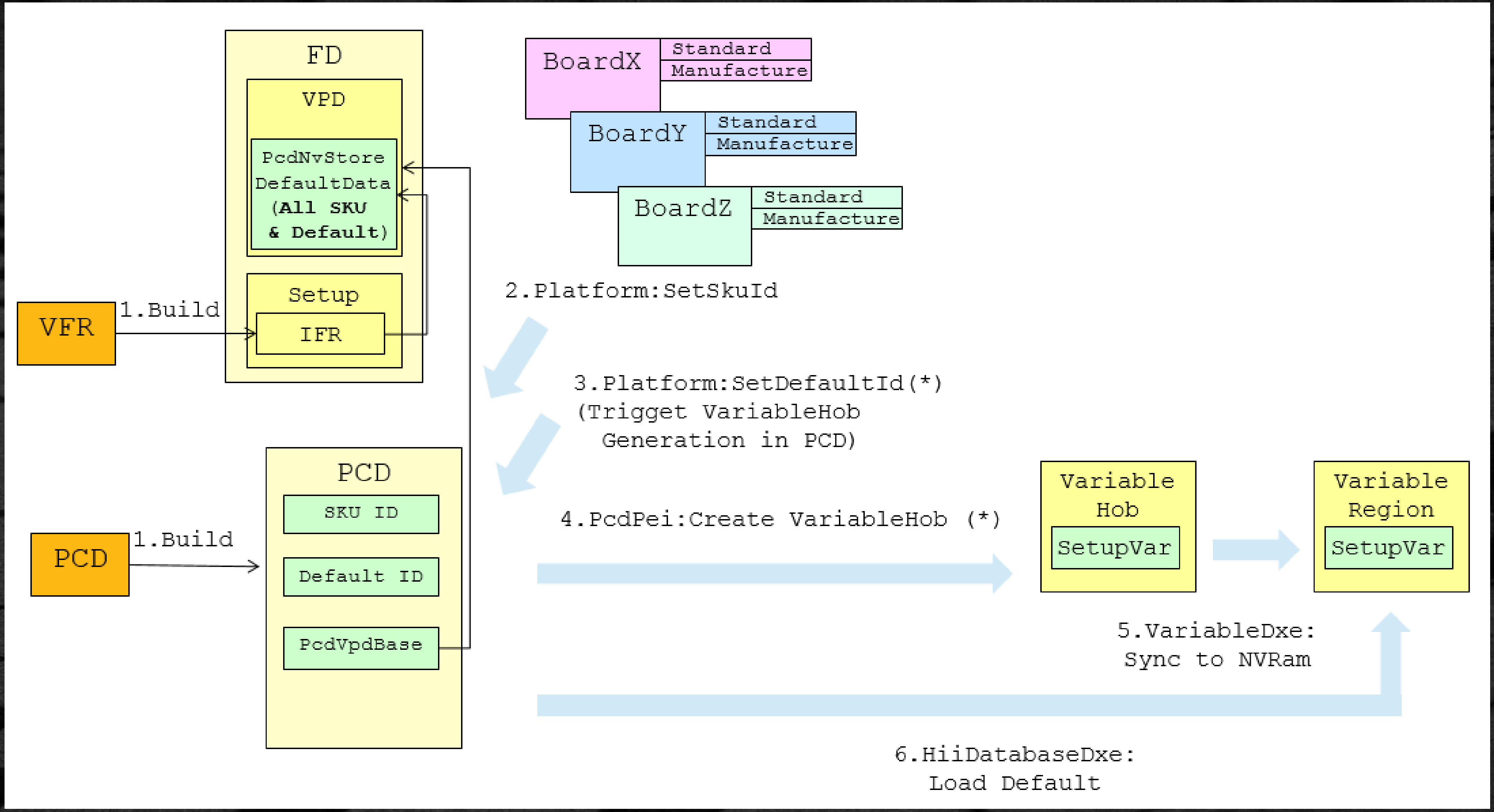
- 🟪 AcpiTable
- 🟪 AcpiSmm

Platform

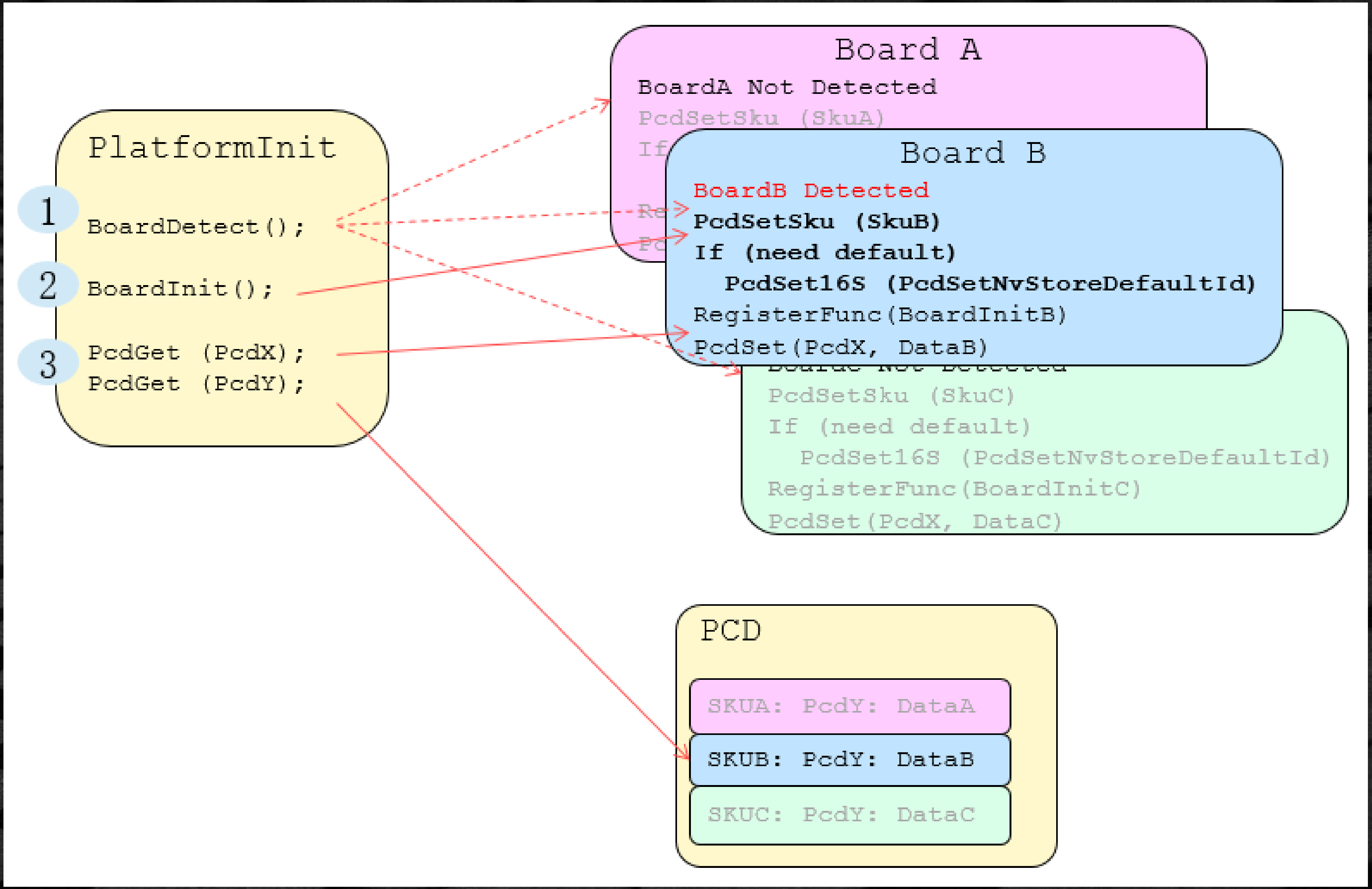
- 🟪 Gpio
- 🟨 Sio

Key - 🟩 EDK II 🟨 Silicon 🟪 Platform

Board default setup variable data



Board Detection and Initialization Flow



PlatformInit

1 BoardDetect();

2 BoardInit();

3 PcdGet (PcdX);
PcdGet (PcdY);

Board A

```
BoardA Not Detected
PcdSetSku (SkuA)
If
Re
Pc
```

Board B

```
BoardB Detected
PcdSetSku (SkuB)
If (need default)
  PcdSet16S (PcdSetNvStoreDefaultId)
RegisterFunc(BoardInitB)
PcdSet(PcdX, DataB)
```

```
BoardC Not Detected
PcdSetSku (SkuC)
If (need default)
  PcdSet16S (PcdSetNvStoreDefaultId)
RegisterFunc(BoardInitC)
PcdSet(PcdX, DataC)
```

PCD

SKUA:	PcdY:	DataA
SKUB:	PcdY:	DataB
SKUC:	PcdY:	DataC

Board Module Package Structure

BoardModulePkg

```
BoardModulePkg /
  Include /
  Library /
    BiosIdLib /
    CmosAccessib /
    PlatformCmosAccessLibNull /
```

Board Generic Functionality

Where:

- **Include**: The include file as the package interface. All interfaces defined in BoardModulePkg.dec are put to here.
- **Library**: It only contains board generic features as independent library, such as BiosIdLib and Cmos Access Lib

Advanced Feature Package

```
Features/Intel /
 AdvancedFeaturePkg /
    Include /
 XxxFeature /
    XxxFeatureSub1Pkg /
       Include /
       Library /
```

Where:

← The package interface and Includes for .DSC & .FDF files

← `Sub1Feature.dsc PostMemory.fdf PreMemory.fdf`

← Implementation of the feature as a library

The advanced features, domains such as SMBIOS table, IPMI, User Interface, Power Management