TI | Developer Conference

February 28-March 2, 2006 • Dallas, TX

Accelerating Innovation with the DaVinci^{TM} Software Code and Programming Model

Raj Pawate
Distinguished Member Technical Staff
Software Applications Manager
Houston, Texas

Agenda

- Introduction
  - Bare deck Silicon to *Silicon with Component SW*
  - Challenges to building a Video Product
- What software content does TI provide?
  - Codecs, drivers (Functionality)
  - Abstraction (easy to use, plumbing, infrastructure)
- What is the Software Architecture?
  - How are the different software components related?
  - What is the programming model?
- Details of the three layers
  - Application layer (APL)
  - Input-Output layer (IOL)
  - Signal Processing layer (SPL)
- TI Software Product Portfolio for DM6446/3
- Conclusion

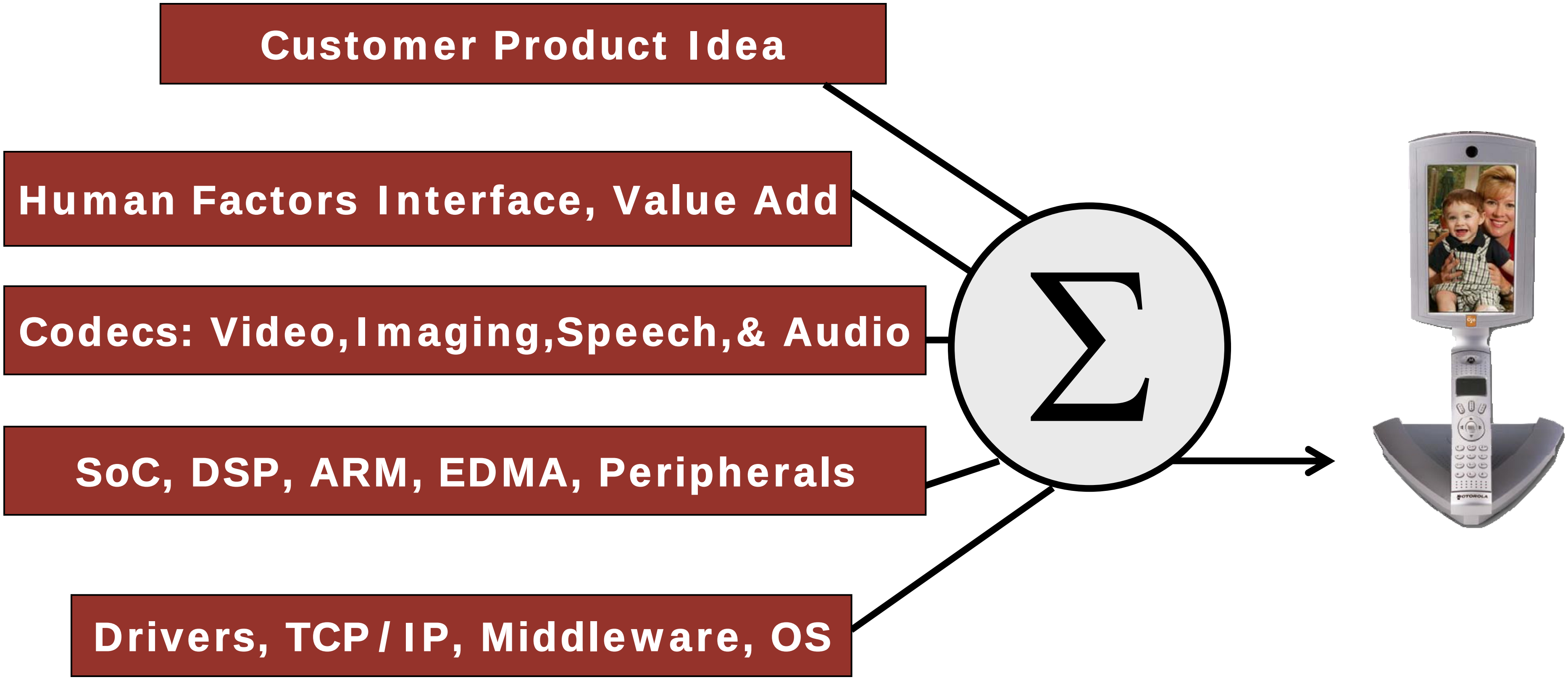
Acronyms

API	Application Programming Interface
APL	Application Layer
EPSI	Easy Peripheral Software Interface
HWAL	Hardware Adaptation Layer
IOL	Input-output Layer
OSAL	Operating System Adaptation Layer
RPC	Remote Procedure Call
SPL	Signal Processing Layer
VISA	Video, Imaging, Speech and Audio
xDM	xDAIS for Digital Media

Transitioning from Bare-deck Silicon to Silicon with Component SW

	Silicon	Silcon + SW
Device	✓	✓
EVM	✓	✓
Tools	✓	✓
Pretested Component-ware *(windows, walls)*		✓
Drivers		✓
Codecs		✓
Pretested subsystem-ware *(floor plans)*		✓
Codec combos		✓
Integrated drivers in OS		✓
Ease of Use and Rules *(building codes)*		✓
Rules for replacing components		✓
APIs, Framework *(Abstraction ware)*		✓

Software Challenges to Building a Video Product



Customer Product Idea

Human Factors Interface, Value Add

Codecs: Video,Imaging,Speech,& Audio

SoC, DSP, ARM, EDMA, Peripherals

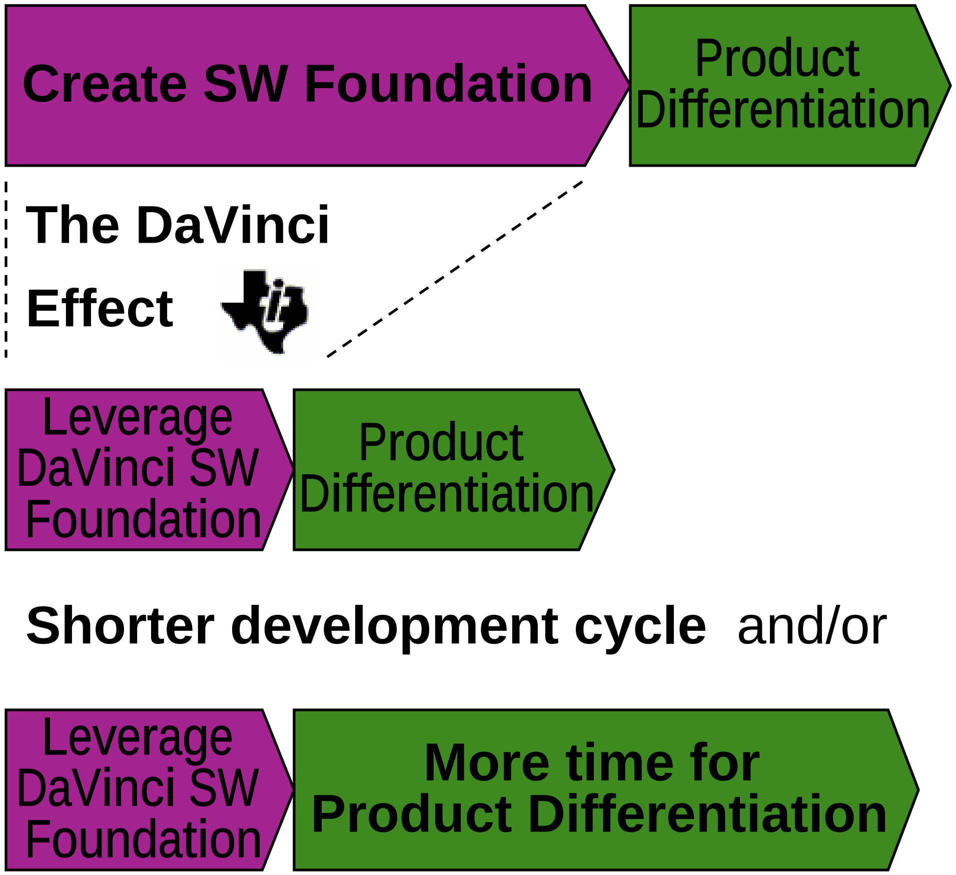
Drivers, TCP/IP, Middleware, OS

- Requires expertise in a variety of different domains
- Several man years to have a hardened codebase

Goal: Accelerate Time to Market

New Product Idea

Standard multi-media product development



Create SW Foundation

Product Differentiation

The DaVinci Effect

Leverage DaVinci SW Foundation

Product Differentiation

Shorter development cycle and/or

Leverage DaVinci SW Foundation

More time for Product Differentiation

Agenda

- **Introduction**
  - Bare deck Silicon to *Silicon with Component SW*
  - Challenges to building a Video Product
- **What software content does TI provide?**
  - Codecs, drivers (Functionality)
  - Abstraction (easy to use, plumbing, infrastructure)
- **What is the Software Architecture?**
  - How are the different software components related?
  - What is the programming model?
- **Details of the three layers**
  - Application layer (APL)
  - Input-Output layer (IOL)
  - Signal Processing layer (SPL)
- **TI Software Product Portfolio for DM6446/3**
- **Conclusion**

MSOffice12

Developer Conference

DaVinci™ Software Offerings Optimized for Efficient Innovation

Operating Systems & Device Drivers

- Linux OS preported to device
- Input output drivers tightly integrated into OS
  - Configurable
  - Robust, tested with EPSI APIs

Published Multimedia Application Programming Interfaces (APIs)

- Industry-recognized APIs
- DaVinci APIs (VISA, EPSI, xDM)

Codec Engine

- Codec abstraction
- Interprocessor communication
- DSP/BIOS™

Multimedia Codecs

- H.264
- MPEG4
- H.263
- WMV9
- VC1
- MPEG2
- JPEG
- AAC+

- AAC
- WMA9
- WMA8
- MP3
- G.711
- G.728
- G.723.1
- G.729ab

Signal Processing Libraries

- Codec Kernels
- FIR, IIR

Slide 8

MSOffice12 Replace this with old version

, 10/27/2005

TI Developer Conference

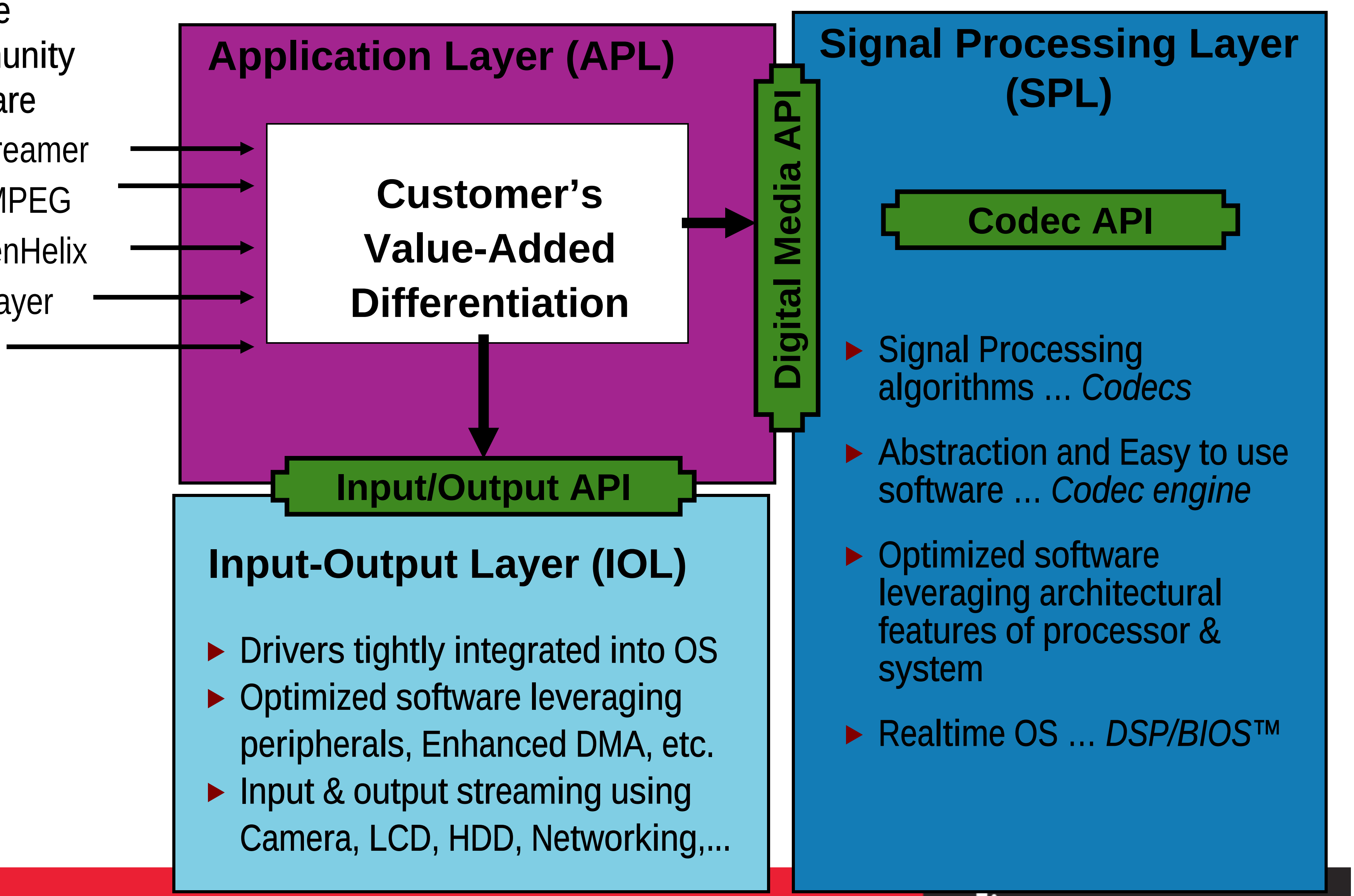
Agenda

- **Introduction**
  - Bare deck Silicon to *Silicon with Component SW*
  - Challenges to building a Video Product
- **What software content does TI provide?**
  - Codecs, drivers (Functionality)
  - Abstraction (easy to use, plumbing, infrastructure)
- **What is the Software Architecture?**
  - How are the different software components related?
  - What is the programming model?
- **Details of the three layers**
  - Application layer (APL)
  - Input-Output layer (IOL)
  - Signal Processing layer (SPL)
- **TI Software Product Portfolio for DM6446/3**
- **Conclusion**

DM644x™ Software Architecture

Linux Open Source Community Software

- Gstreamer
- FFMPEG
- OpenHelix
- Mplayer
- etc



Application Layer (APL)

Customer's Value-Added Differentiation

Digital Media API

Input/Output API

Input-Output Layer (IOL)

- Drivers tightly integrated into OS
- Optimized software leveraging peripherals, Enhanced DMA, etc.
- Input & output streaming using Camera, LCD, HDD, Networking,...

Signal Processing Layer (SPL)

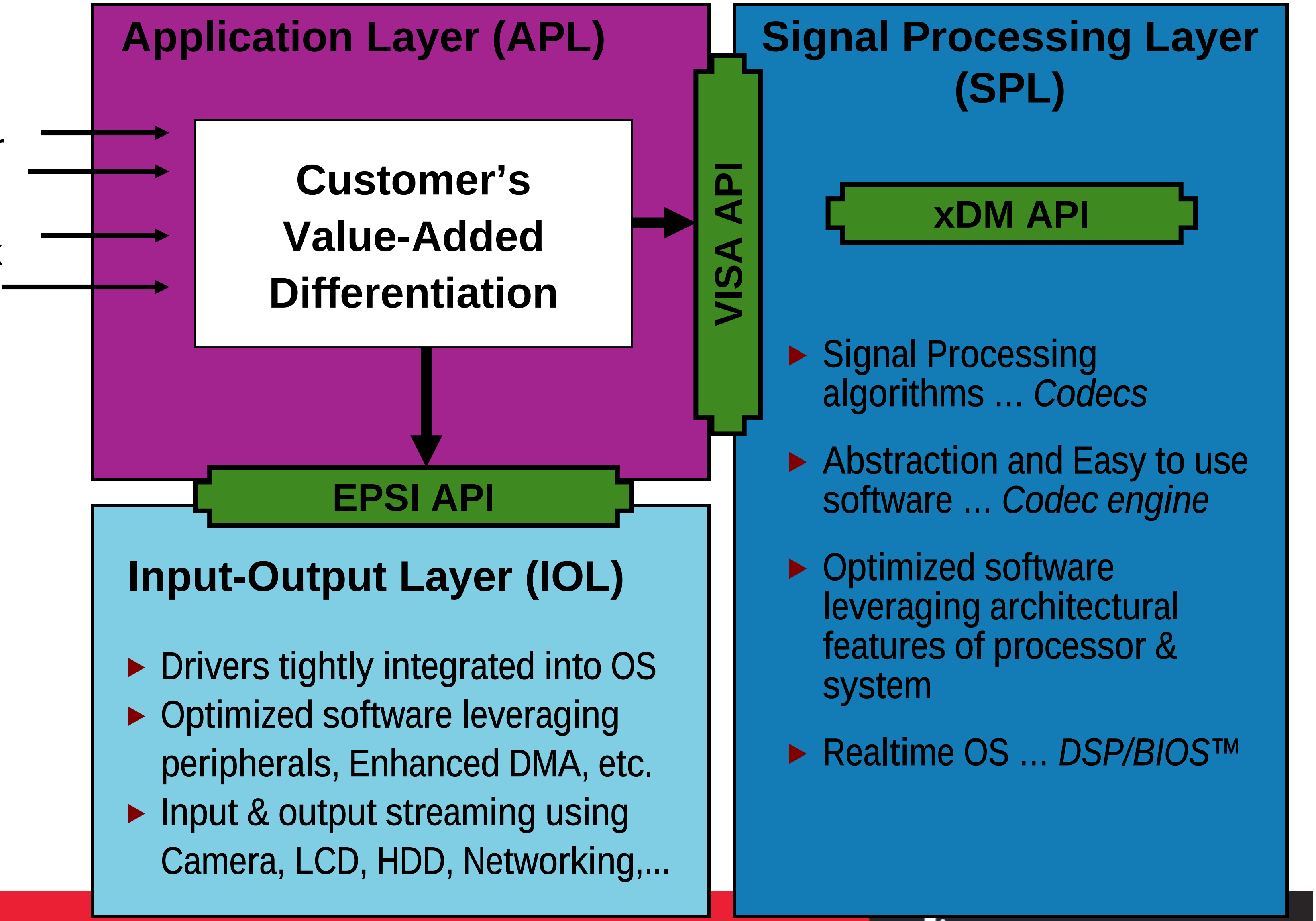
Codec API

- Signal Processing algorithms … *Codecs*
- Abstraction and Easy to use software … *Codec engine*
- Optimized software leveraging architectural features of processor & system
- Realtime OS … *DSP/BIOS™*

DM644x™ Software Architecture

Linux Open Source Community Software

- Gstreamer
- FFMPEG
- OpenHelix
- Mplayer
- etc



Application Layer (APL)

Customer's Value-Added Differentiation

VISA API

EPSI API

Input-Output Layer (IOL)

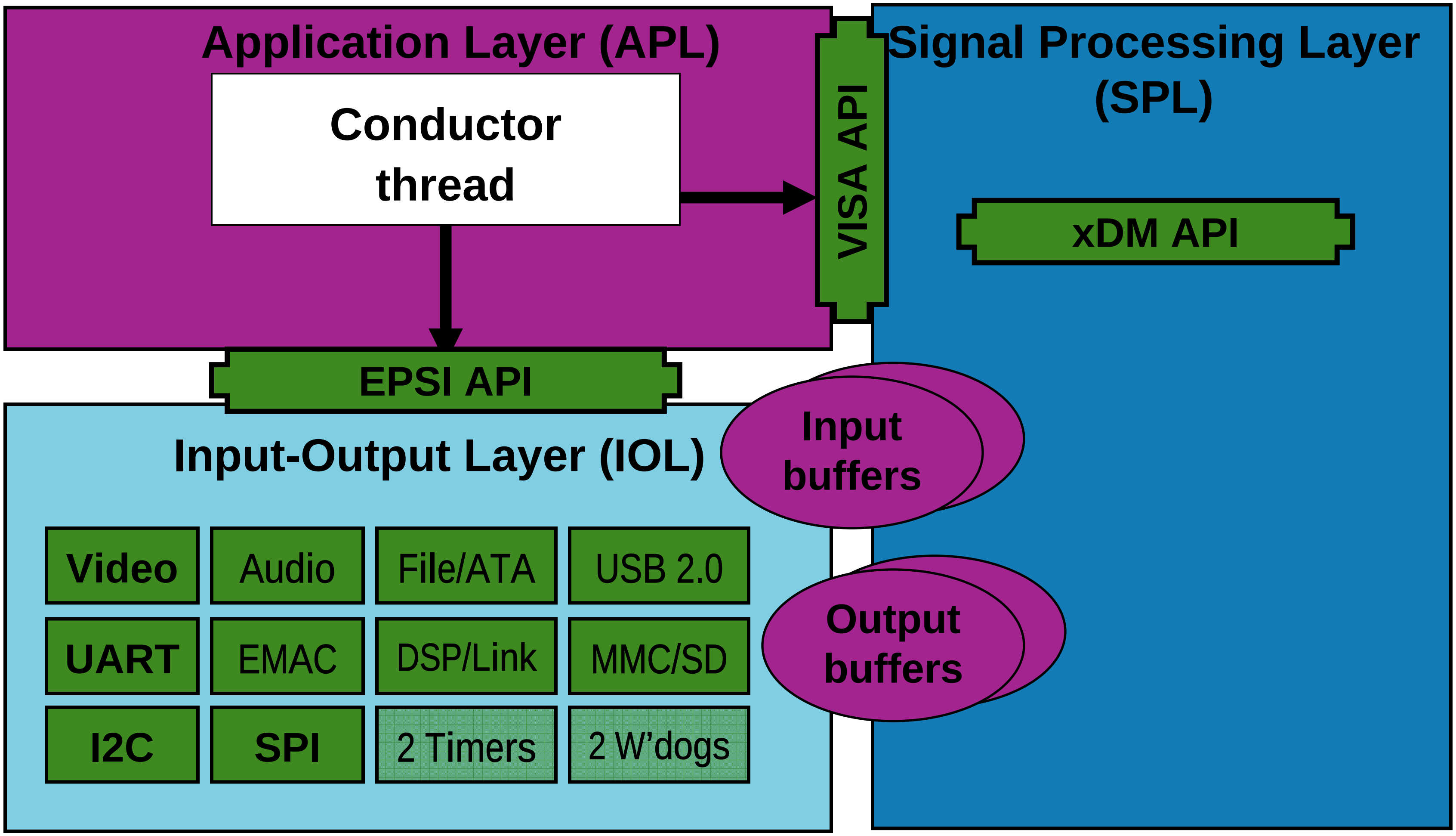
- Drivers tightly integrated into OS
- Optimized software leveraging peripherals, Enhanced DMA, etc.
- Input & output streaming using Camera, LCD, HDD, Networking,...

Signal Processing Layer (SPL)

xDM API

- Signal Processing algorithms … *Codecs*
- Abstraction and Easy to use software … *Codec engine*
- Optimized software leveraging architectural features of processor & system
- Realtime OS … *DSP/BIOS™*

Input Output Layer (IOL)



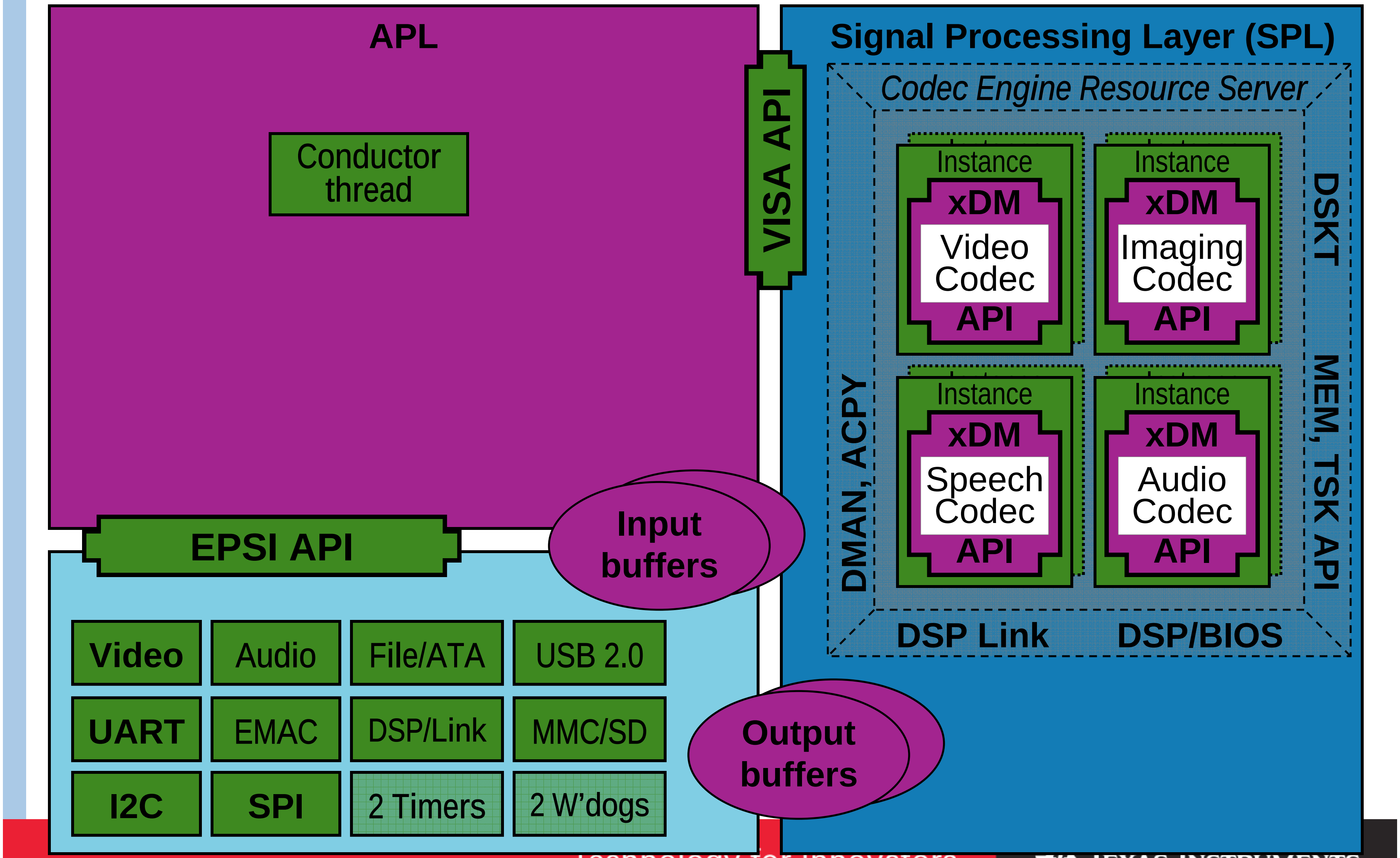
EPSI APIs: open **read** **write** close

Linux APIs….V4L2 for Video, OSS for Audio, Speech, …

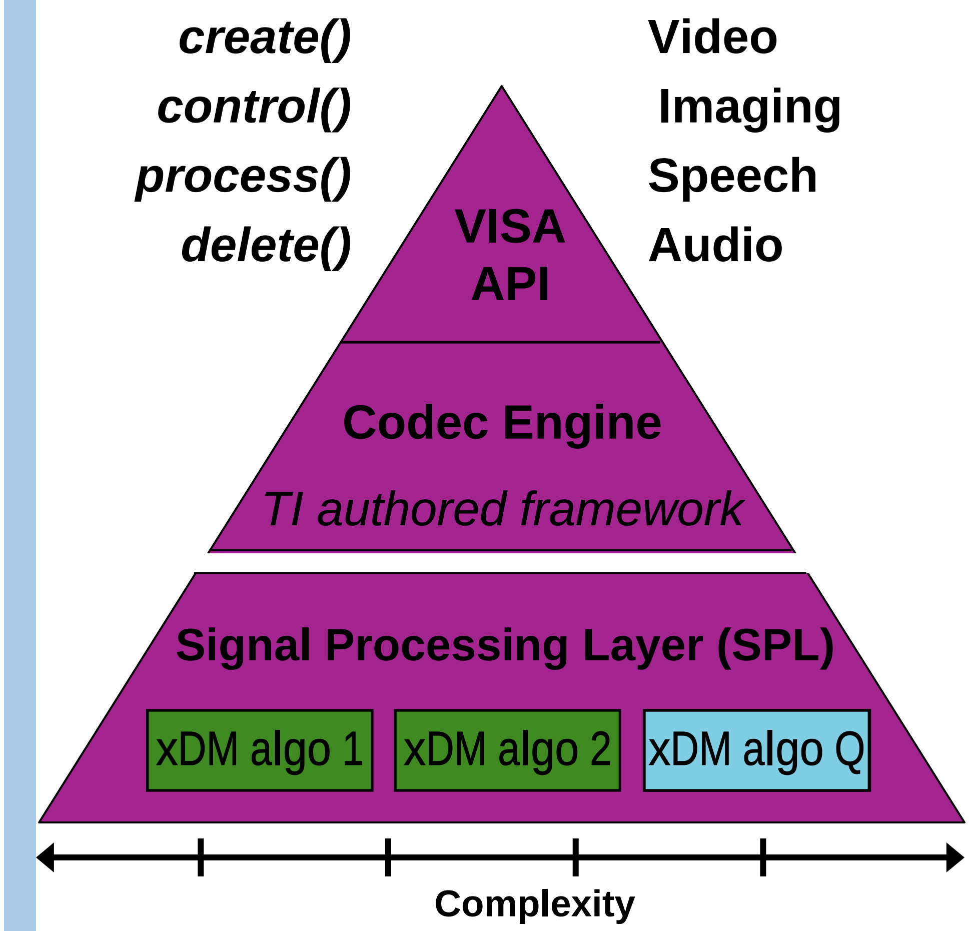
No overhead in passing buffers… **only pointers are passed**

Optimized, robust (tested) drivers leveraging SoC features, EDMA, etc

Signal Processing Layer (SPL)



VISA



create()
control()
process()
delete()

VISA API

Video
Imaging
Speech
Audio

Codec Engine

TI authored framework

Signal Processing Layer (SPL)

xDM algo 1
xDM algo 2
xDM algo Q

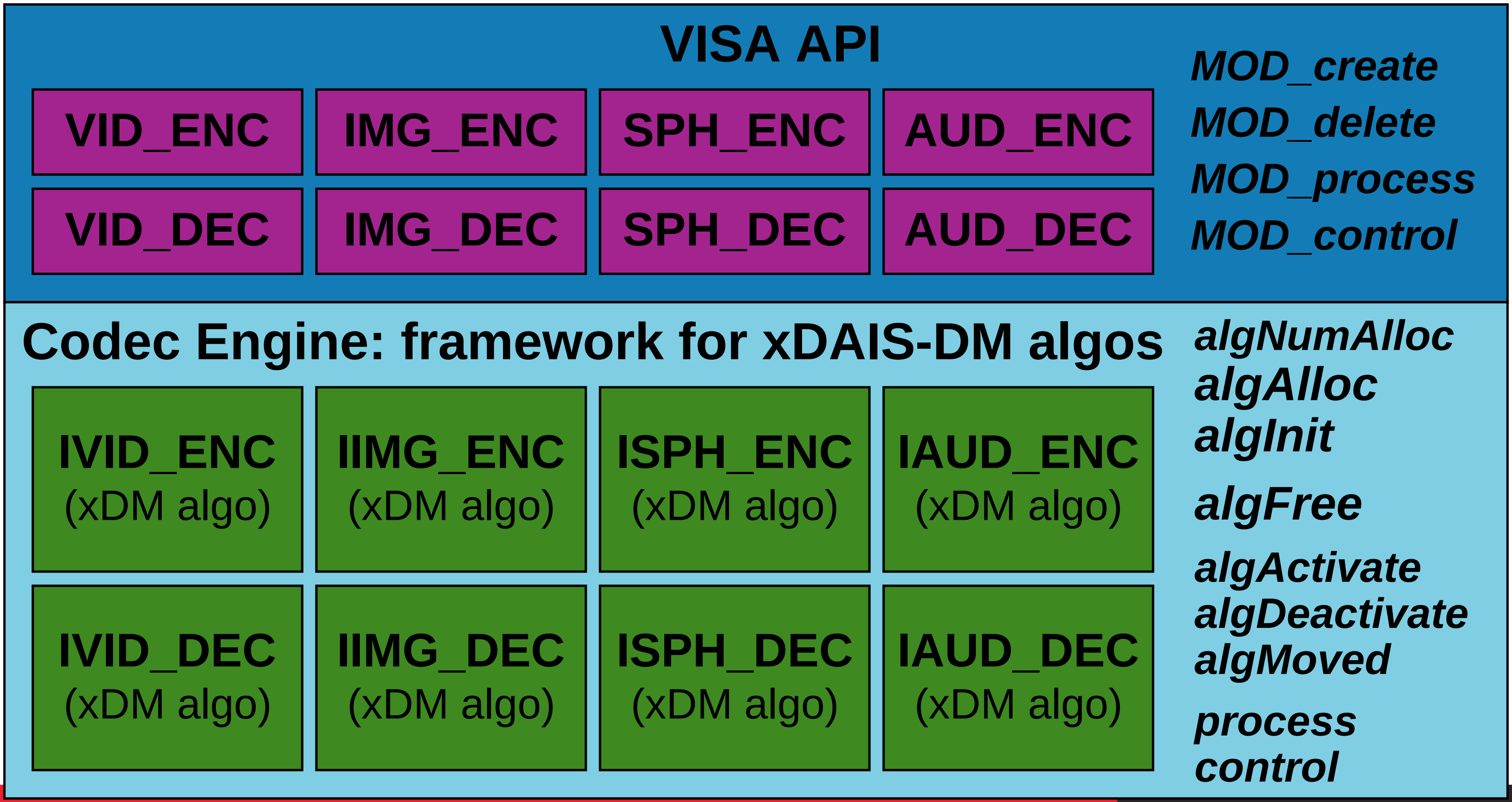
Complexity

Reducing dozens of API to 4 sets

- Complexities of the Signal Processing Layer “SPL” are abstracted via the Codec Engine and VISA API
- VISA API are the user interface to the Codec Engine
- VISA = 4 processing domains :
  Video Imaging Speech Audio
- Separate API set for encode and decode
- Thus, a total of 8 API classes:
  `VIDENC IMGENC SPHENC AUDENC`
  `VIDDEC IMGDEC SPHDEC AUDDEC`
- Key API in each set (where “xxx” is one of the groups above):
  `xxx_create    xxx_delete`
  `xxx_process   xxx_control`
- The experienced DSP programmer can employ a ready-made Signal Processing Layer, create an SPL from packaged or ‘raw’ xDM algos, or author their own algos depending on their needs and skills with DSP

VISA Abstracts Details of xDM Algos

- **Application author controls algos via high level VISA API**
- **xDAIS-DM (xDM) algorithms implement an enhanced xDAIS interface**
- **Codec Engine is a *framework* that implements VISA fxns on xDM algos**
  - *eg*: MOD_create() = algNumAlloc() + algAlloc() + MEM_alloc() + algInit()



VISA API

VID_ENC **IMG_ENC** **SPH_ENC** **AUD_ENC**

VID_DEC **IMG_DEC** **SPH_DEC** **AUD_DEC**

MOD_create
MOD_delete
MOD_process
MOD_control

Codec Engine: framework for xDAIS-DM algos

IVID_ENC (xDM algo) **IIMG_ENC** (xDM algo) **ISPH_ENC** (xDM algo) **IAUD_ENC** (xDM algo)

IVID_DEC (xDM algo) **IIMG_DEC** (xDM algo) **ISPH_DEC** (xDM algo) **IAUD_DEC** (xDM algo)

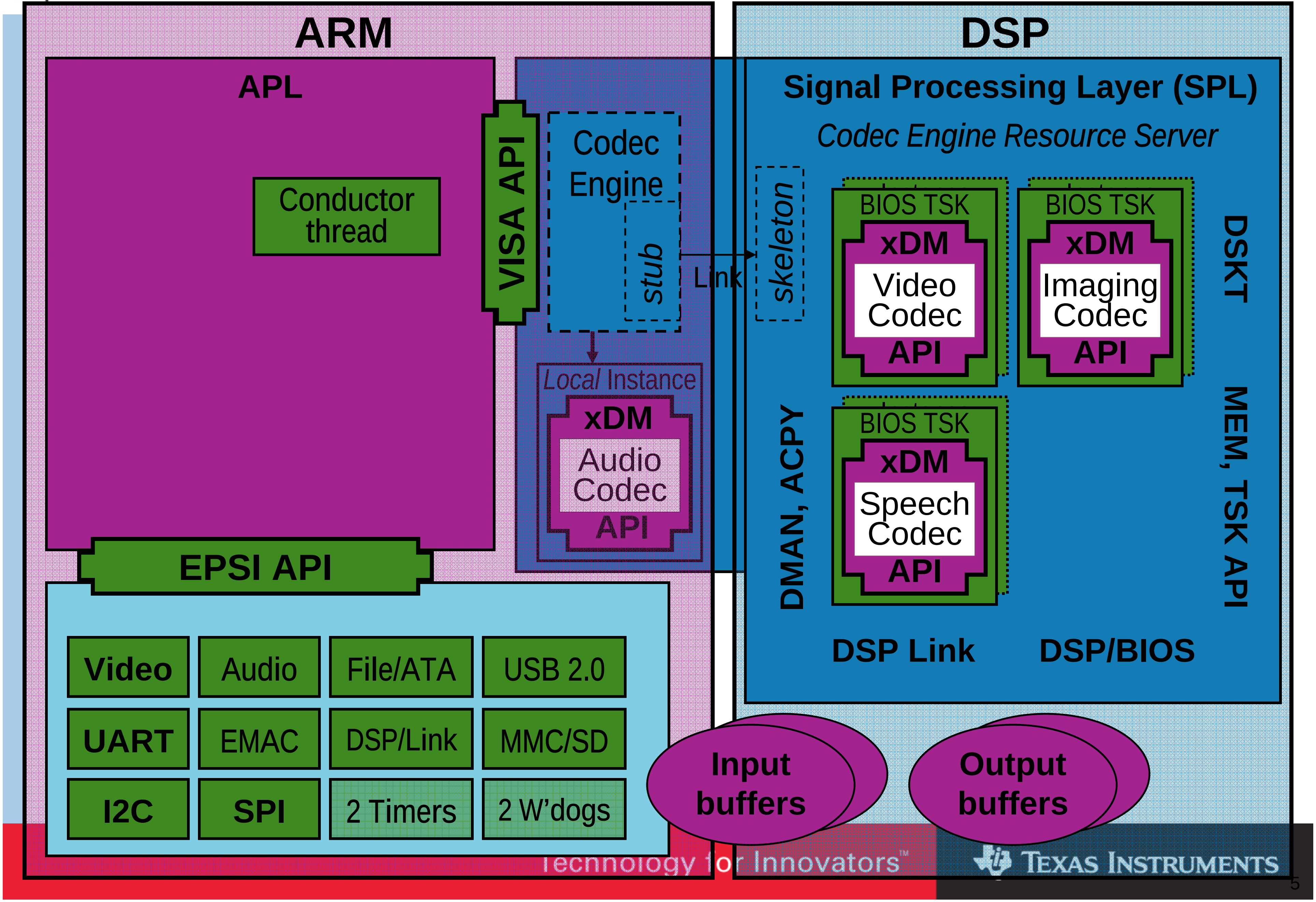
algNumAlloc
algAlloc
algInit

algFree

algActivate
algDeactivate
algMoved

process
control

Mapping Software to Hardware: Crossing Processor boundaries w/- Remote Procedure Calls (RPC)



ARM

APL

Conductor thread

VISA API

Codec Engine

stub

Link

Local Instance

xDM

Audio Codec

API

EPSI API

Video

Audio

File/ATA

USB 2.0

UART

EMAC

DSP/Link

MMC/SD

I2C

SPI

2 Timers

2 W'dogs

DSP

Signal Processing Layer (SPL)

Codec Engine Resource Server

skeleton

BIOS TSK

xDM

Video Codec

API

BIOS TSK

xDM

Imaging Codec

API

BIOS TSK

xDM

Speech Codec

API

DSKT

MEM, TSK API

DMAN, ACPY

DSP Link

DSP/BIOS

Input buffers

Output buffers

VISA Benefits

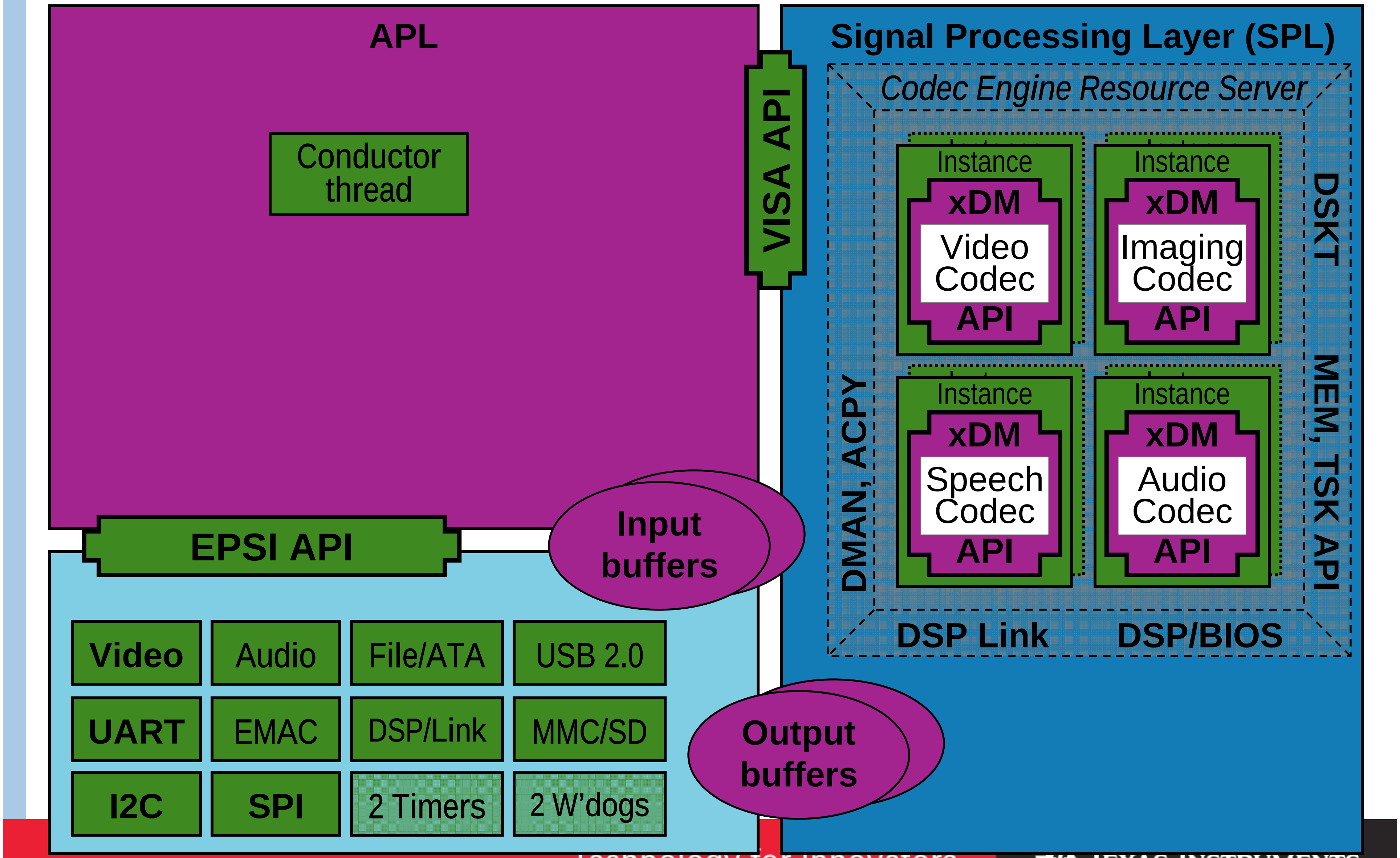
Application Author Benefits

- App author enjoys benefits of signal processing layer without need to comprehend the complexities of the DSP algo or underlying hardware
- Application author uses only *one* API for a given media engine class
- Changing codec within the class involves *no* changes to app level code
- All media engine classes have a similar look and feel
- Adapting any app code to other engines and API is very straight forward
- Example apps that use VISA to manage xDM codecs provided by TI
- Customers can create multimedia frameworks that will leverage VISA API
- VISA contains hooks allowing additional functionalities within codecs
- Authoring app code, multimedia frameworks & end equipment expertise is what customers do best, and want to focus on - VISA optimizes this

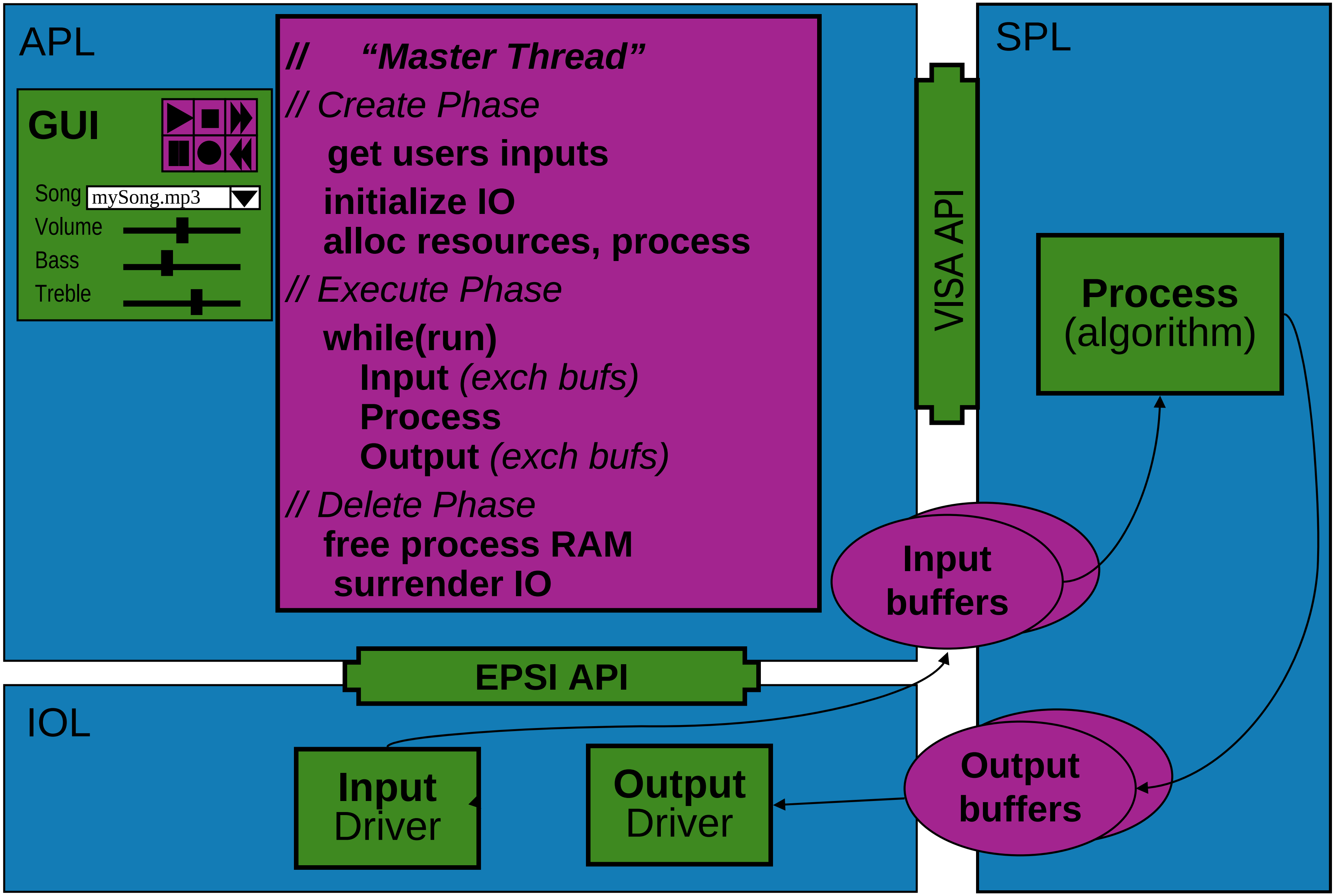
Algorithm Author Benefits

- Codec engine authors have a known standard to write to
- Codec authors need have no knowledge of the end application
- Codecs can be sold more readily, since they are easy to apply widely
- Each class contains the information necessary for that type of media
- VISA, and xDAIS-DM, build on xDAIS – an established algo interface
- Tools exist today to adapt algos to xDAIS, and may include –DM soon (?)

Application Layer (APL)



APL Conductor Thread



APL

GUI

Song mySong.mp3

Volume

Bass

Treble

```
//    "Master Thread"
// Create Phase
    get users inputs
    initialize IO
    alloc resources, process
// Execute Phase
    while(run)
        Input (exch bufs)
        Process
        Output (exch bufs)
// Delete Phase
    free process RAM
    surrender IO
```

VISA API

SPL

Process (algorithm)

Input buffers

Output buffers

EPSI API

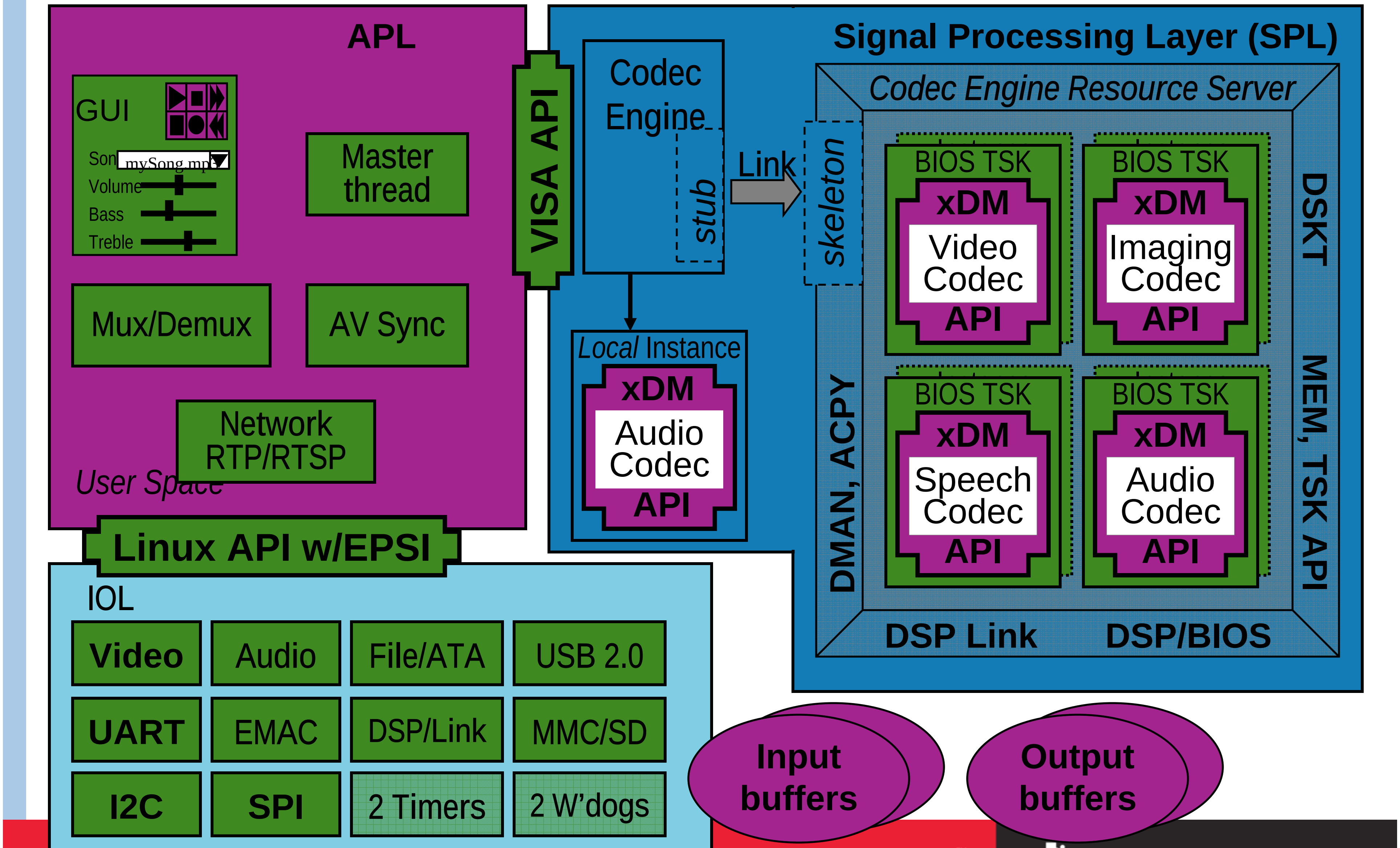
IOL

Input Driver

Output Driver

EPSI APIs: open **read write** close **VISA APIs:** create **process control** delete

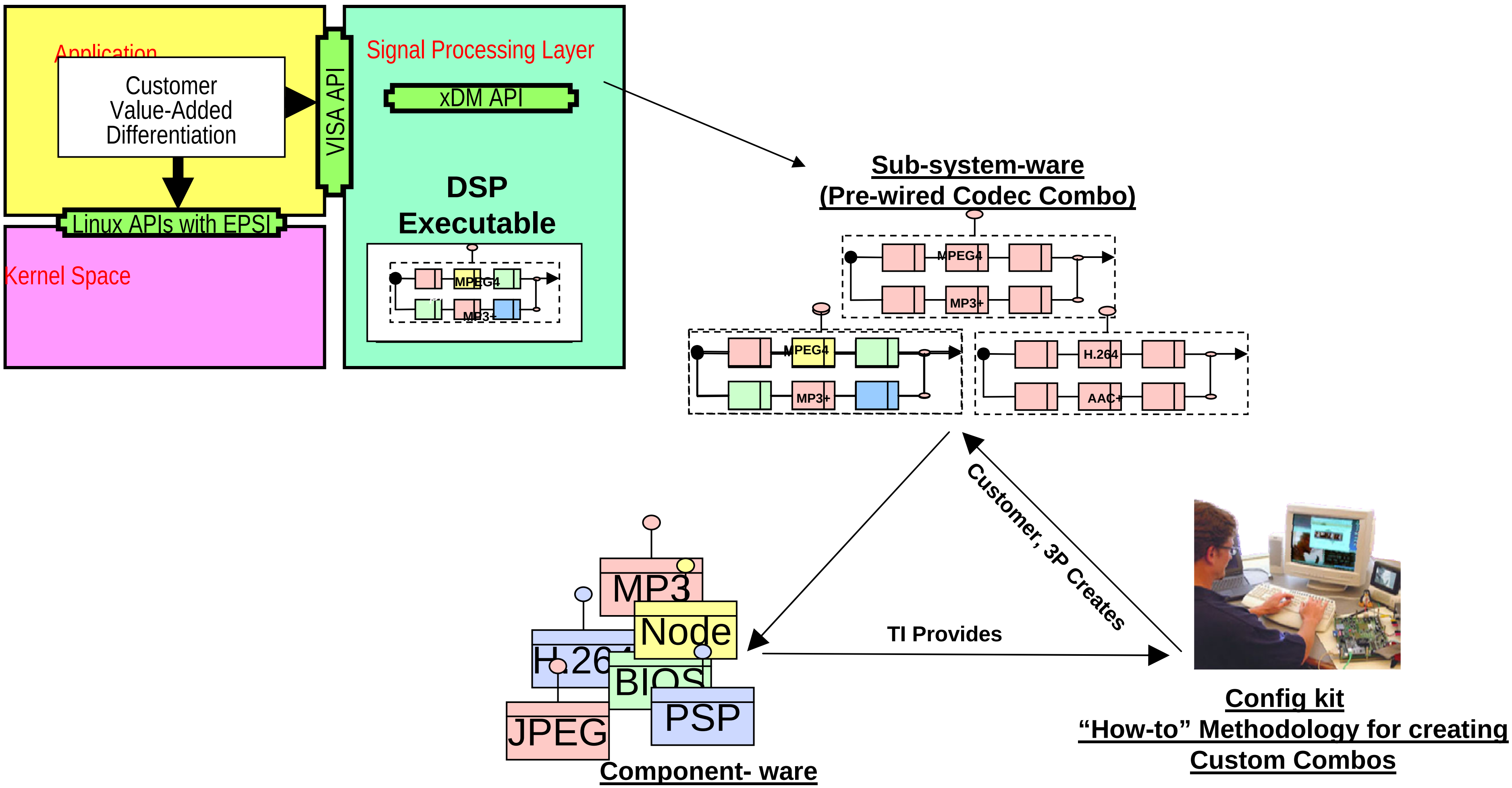
DaVinci Software Architecture on DM6446



DaVinci Programmers Model

- Three layers ... Application layer, IO layer, and Signal Processing layer
- Signal Processing layer
  - presents VISA APIs to all other layers
  - implement codecs using xDM APIs,
  - implements all other algorithms using xDAIS APIs
  - *buffer based processing, decoupled from all other layers*
  - delivered as
    - .lib for uniprocessor SoCs,
    - .out for multiprocessor SoCs
- Input output layer
  - presents EPSI APIs to all other layers
  - implements peripheral drivers
  - generates an interrupt to APL whenever a buffer is full
  - buffers in shared memory, only pointers are passed
- Application layer
  - implements the conductor thread, GUI, middleware,etc.
  - orchestrates all input and output streams to other layers
  - *interfaces with the other layers as* ***built-in library functions***

Simplified Embedded Video



Application

Customer Value-Added Differentiation

Linux APIs with EPSI

Kernel Space

VISA API

Signal Processing Layer

xDM API

DSP Executable

Sub-system-ware (Pre-wired Codec Combo)

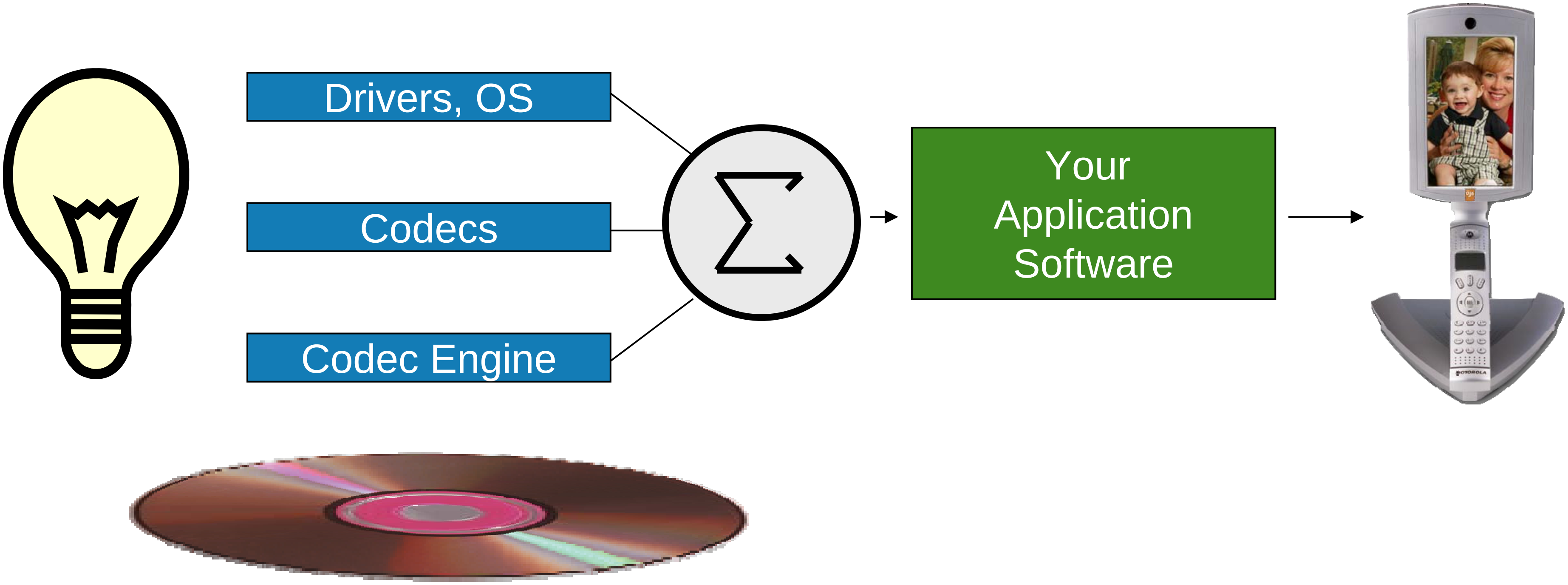
Customer, 3P Creates

TI Provides

Component- ware

Config kit “How-to” Methodology for creating Custom Combos

Conclusion: Accelerating Video Innovation From Idea to Realization



Drivers, OS

Codecs

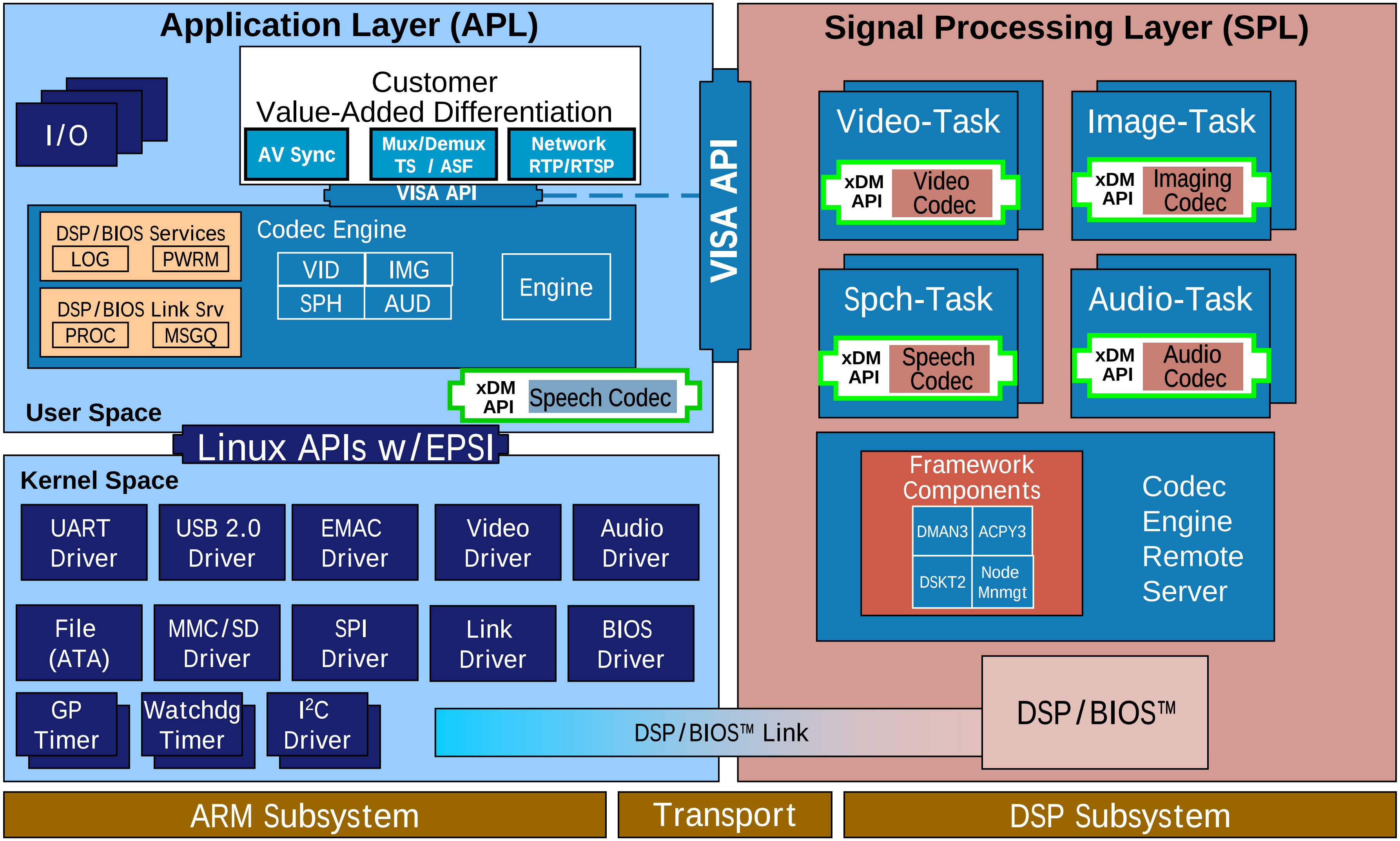
Codec Engine

Your Application Software

TI Software Platform

TI Developer Conference

DM644x™ Software Stack



Technology for Innovators™

Texas Instruments