NXP | FTF 2016 TECHNOLOGY FORUM

i.MX 8
GRAPHICS ARCHITECTURE

FTF-DES-N1940

RAFAL MALEWSKI
HEAD OF GRAPHICS TECH ENGINEERING CENTER
MAY 18, 2016

PUBLIC USE

AGENDA

- i.MX 8 Series Scalability
- GPU Architecture
- Display Controller
- Designing for Safety
- Vision Processing

i.MX is…

SoC Scalability = Investment Re-Use

Replace the Chip ➪ Increase capability.

(Pin, Software, IP compatibility among parts)

i.MX 8 Series

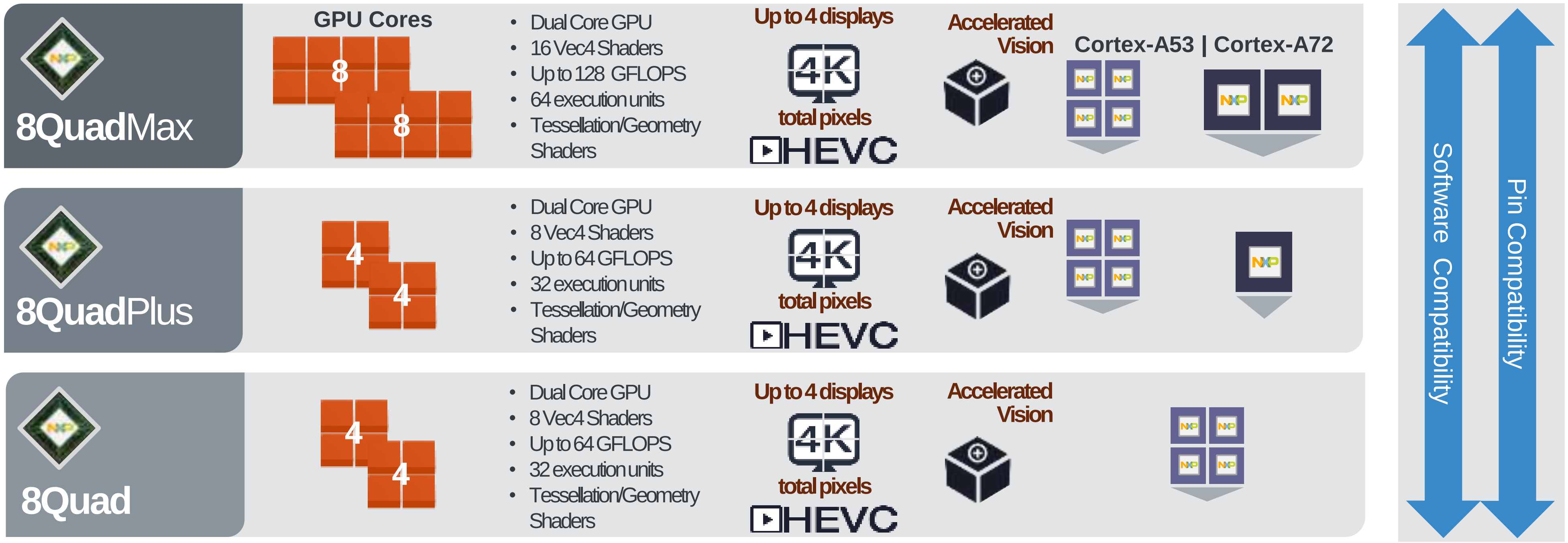
	GPU Cores		Displays		ARM Cores Cortex-A53 | Cortex-A72
8QuadMax	8 8	• Dual Core GPU • 16 Vec4 Shaders • Up to 128 GFLOPS • 64 execution units • Tessellation/Geometry Shaders	Up to 4 displays 4K total pixels HEVC	Accelerated Vision	
8QuadPlus	4 4	• Dual Core GPU • 8 Vec4 Shaders • Up to 64 GFLOPS • 32 execution units • Tessellation/Geometry Shaders	Up to 4 displays 4K total pixels HEVC	Accelerated Vision	
8Quad	4 4	• Dual Core GPU • 8 Vec4 Shaders • Up to 64 GFLOPS • 32 execution units • Tessellation/Geometry Shaders	Up to 4 displays 4K total pixels HEVC	Accelerated Vision	
8Dual	8	• Single Core GPU • 8 Vec4 Shaders • Up to 64 GFLOPS • 32 execution units • Tessellation/Geometry Shaders	Up to 2 displays 2x 1080p total pixels FULL HD 1080p HEVC	Accelerated Vision	
8DualLite	4	• Single Core GPU • 4 Vec4 Shaders • Up to 32 GFLOPS • 16 execution units • Tessellation/Geometry Shaders	Up to 2 displays 2x 1080p total pixels FULL HD 1080p HEVC	Accelerated Vision	

Software Compatibility

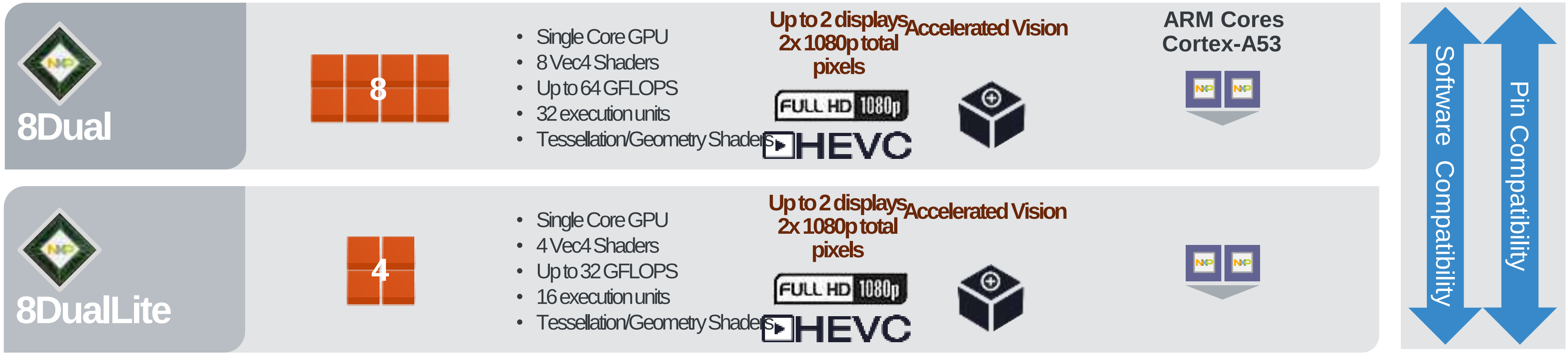
Pin Compatibility

Pin Compatibility

i.MX 8 Series – The Doubles



i.MX 8 Series – The Singles



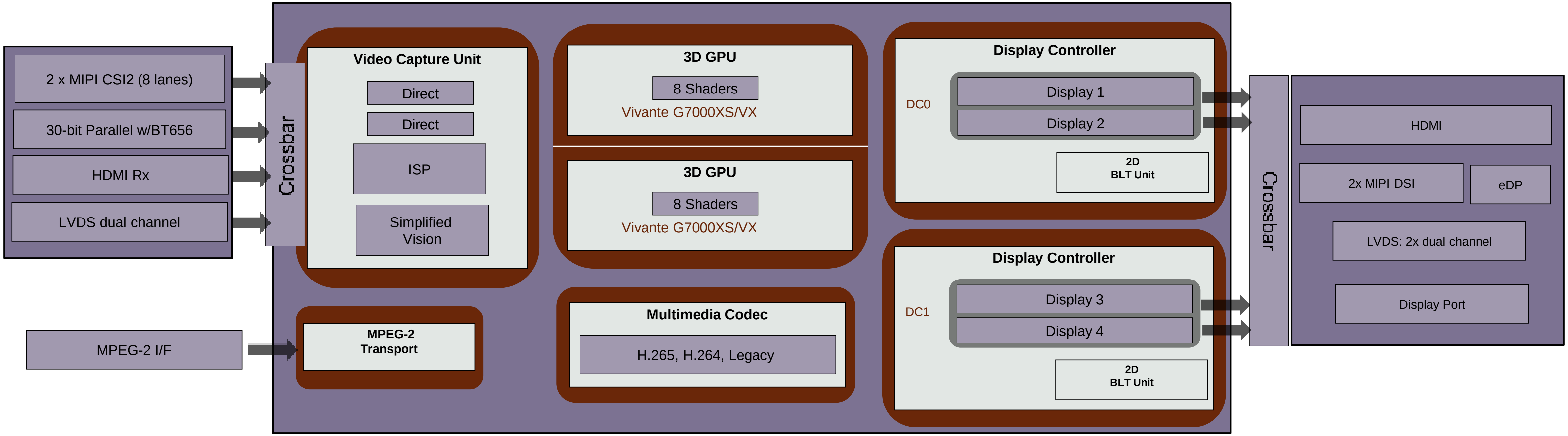
But What About the Rest of the Family Tree?

NXP Full GPU Line

Product	i.MX 6 SoloX	i.MX 6 Solo DualLite	i.MX 6 Dual Quad	i.MX 6 DualPlus QuadPlus	i.MX 8 Dual Lite*	i.MX 8 Dual*	i.MX 8 Quad* QuadPlus*	i.MX 8 QuadMax*
GPU 2D	GC400T (2D)	GC320	GC355 GC320	GC355 GC328	High Perf 2D Blit Engine	High Perf 2D Blit Engine	High Perf 2D Blit Engine	High Perf 2D Blit Engine
GPU 3D	GC400T (3D)	GC880	GC2000	GC2000+	GC7000Lite XSVX	GC7000 XSVX	x2 GC7000Lite XSVX	x2 GC7000 XSVX
# Shaders (Vec4)	1	1	4	4	4	8	4 + 4	8 + 8
Pixel Rate (Mpix/s)	180	264	1056	1188	1200	1600	1600 + 1600	1600 + 1600
Geom. Rate (MTri/s)	36	81	176	198	200	267	267 + 267	267 + 267
GFLOPS Med / High Precision	2.9 (high)	4.2 (high)	19 (high)	51.2 / 23	64 / 32	128 / 64	128 / 64	256 / 128
2D API	OpenVG 1.1[†] G2D	OpenVG 1.1[†] G2D	OpenVG 1.1 G2D	OpenVG 1.1 G2D	OpenVG 1.1[†] G2D	OpenVG 1.1[†] G2D	OpenVG 1.1[†] G2D	OpenVG 1.1[†] G2D
3D API	OGL ES 2.0	OGL ES 3.0	OGL ES 3.0	OGL ES 3.0	OGL ES 3.2	OGL ES 3.2	OGL ES 3.2	OGL ES 3.2
Compute	N/A	N/A	OCL 1.2 EP	OCL 1.2 FP	OCL 2.0	OCL 2.0	OCL 2.0	OCL 2.0
Other	2D / 3D Multithreaded	N/A	N/A	N/A	OpenVX 1.01	OpenVX 1.01	OpenVX 1.01	OpenVX 1.01

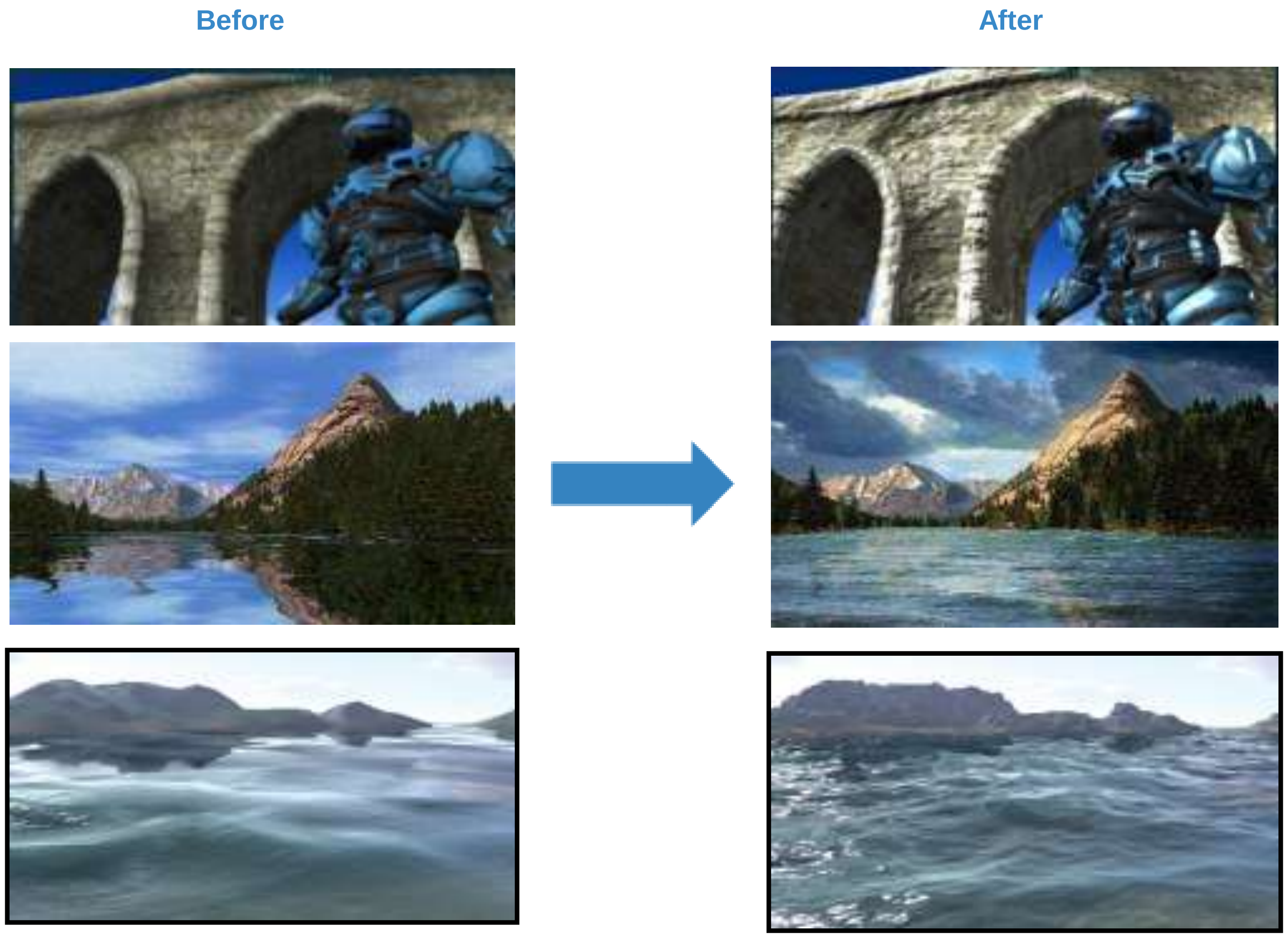
GPU

Visual Processing Complex



i.MX 8: GPU

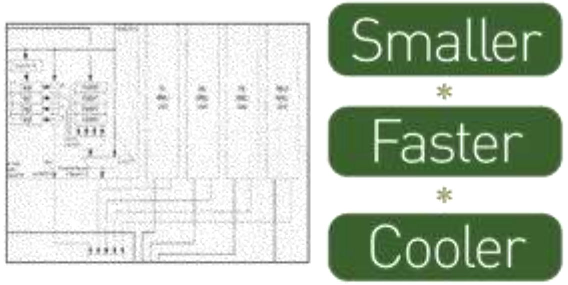
- 8x faster than i.MX 6Quad*
- Scalable Performance
  - Up to 128 GFLOPs and 64 CUs
  - Unified Shader model
- Latest Capabilities
  - Geometry Shading
  - Hardware Tessellation
  - Active power management
- Latest Standards
  - Khronos OpenGL up to Vulkan
  - Khronos OpenVX (Vision)
  - OpenCL 2.0



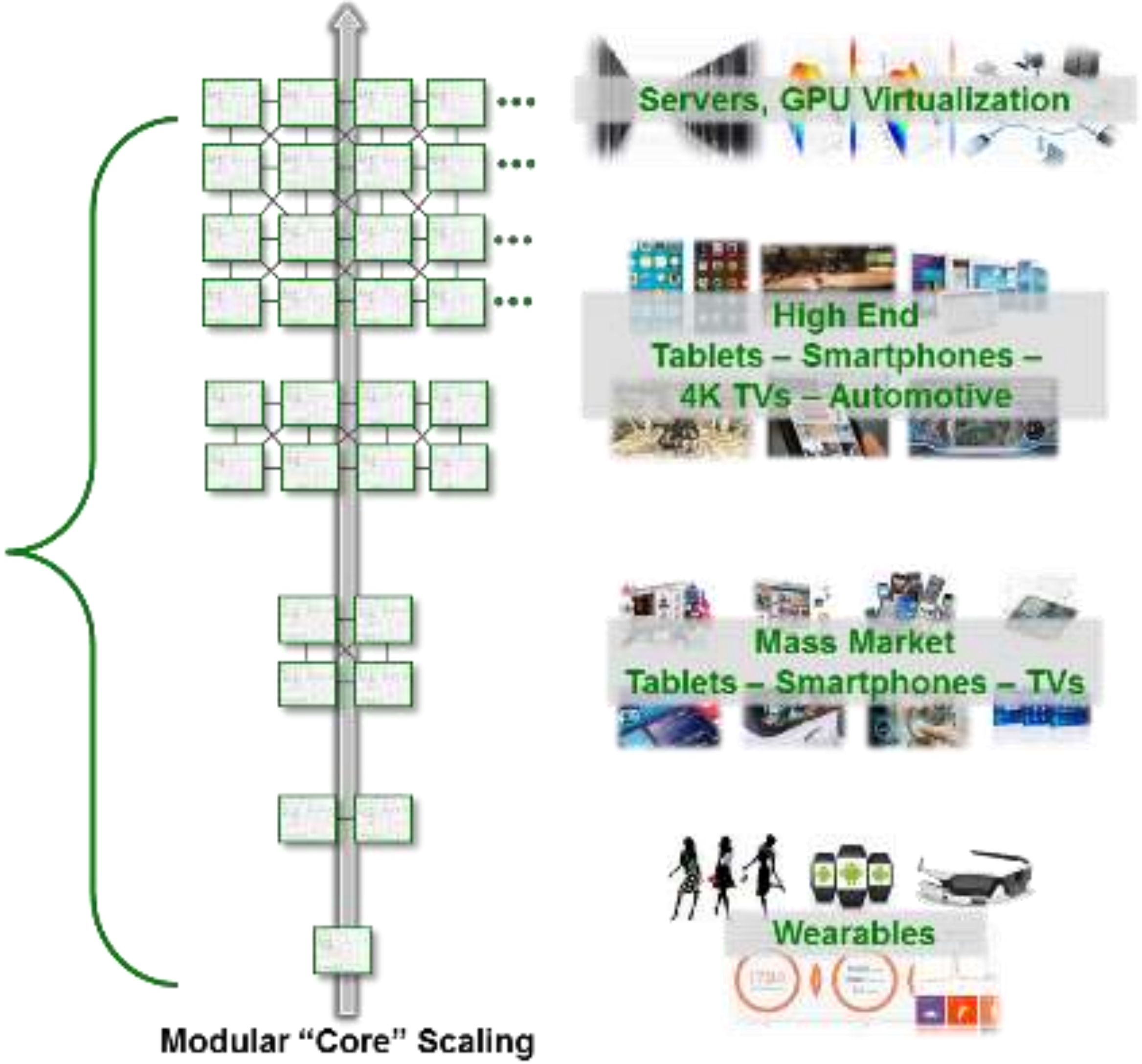
i.MX 8 GPU (GC7000 family)

*** i.MX 8QuadMax w/ dual GC7000**

GC7000 GPU Architecture

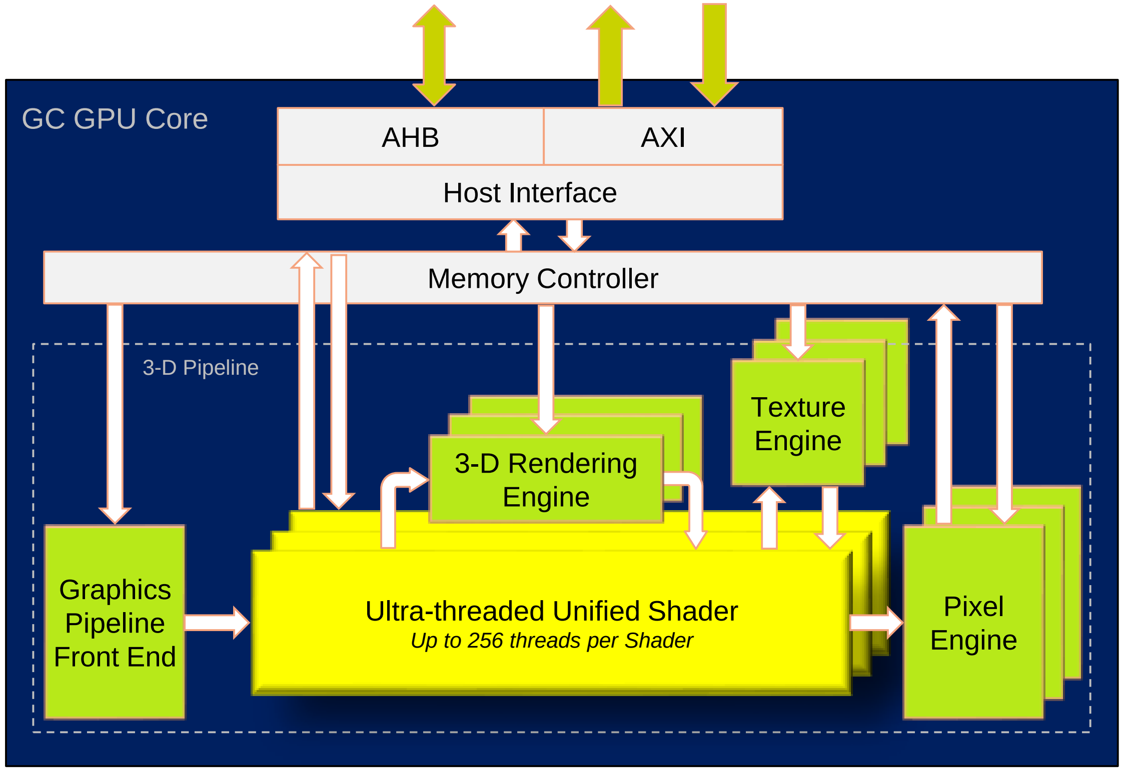


"CORE" Building Block



Scalable Ultra-Threaded Unified Shader

- Architectural Features
  - Massive data parallelism
  - Up to 16x SIMD Vec-4 shaders
  - Balanced performance/bandwidth
    - Tile rasterization
    - Many caches
    - Fast depth culling
    - Fast clear
    - Texture compression
  - Native OpenGL ES 3.2 rendering
  - High quality 4x MSAA anti-aliasing
  - Up to 256 independent threads per shader operate on discrete data in parallel



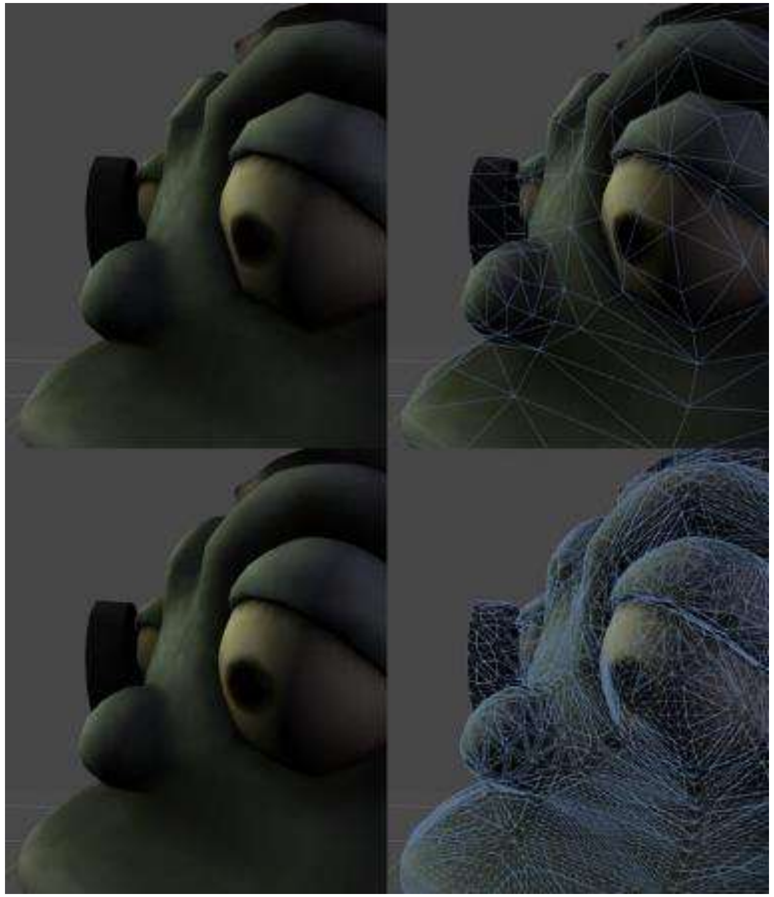
i.MX 8: 'XS' GPU

Tessellation Shading

- GPU hardware function that increases the detail of given polygons via a shader program. Polygons are subdivided into smaller and higher resolution polygons programmatically by the tessellation shader.

- Promises close to "unlimited detail" and greatly reduces 3D model sizes, memory requirements and GPU workload.

- Great for detailed maps (navigation), car or other 3D models, extremely detailed GUI designs, etc. with significantly lower starting polygon counts of the 3D content.



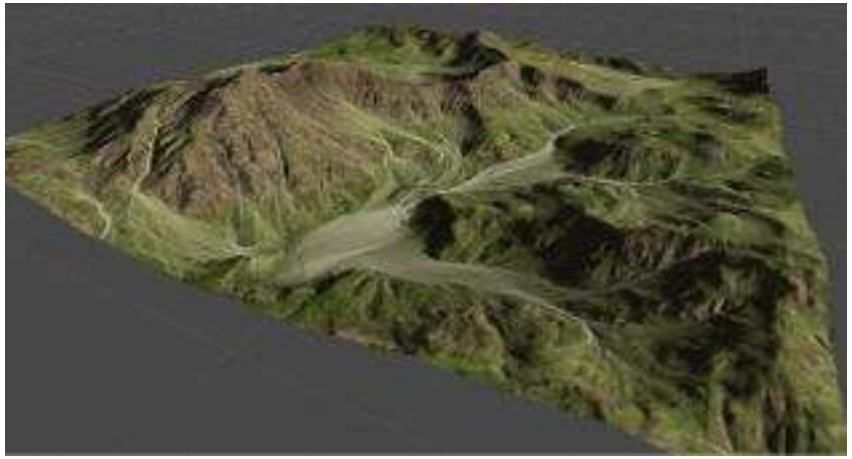
i.MX 8: ‘XS’ GPU

Geometry Shading

- GPU hardware function that can either modify existing 3D primitives that come through the “pipeline” or create NEW polygons programmatically to inject into the 3D pipeline.

- Geometry shaders run AFTER tessellation shading for almost unlimited scene flexibility.

- Unlimited use cases – but terrain generation is a key area where it is useful in automotive applications



The New New Thing

- Vulkan
  - Close to Metal API (not a replacement for OpenGL/ES)
  - No reference counting, less state machine upkeep
  - Give more control to users and less to the driver
  - Much more complicated: High Competence, High Rewards
  - Available on all hardware that is OpenGL ES 3.1 capable
  - No Tessellation yet, to make sure of widest possible distribution

Where Are Graphics API's Going?

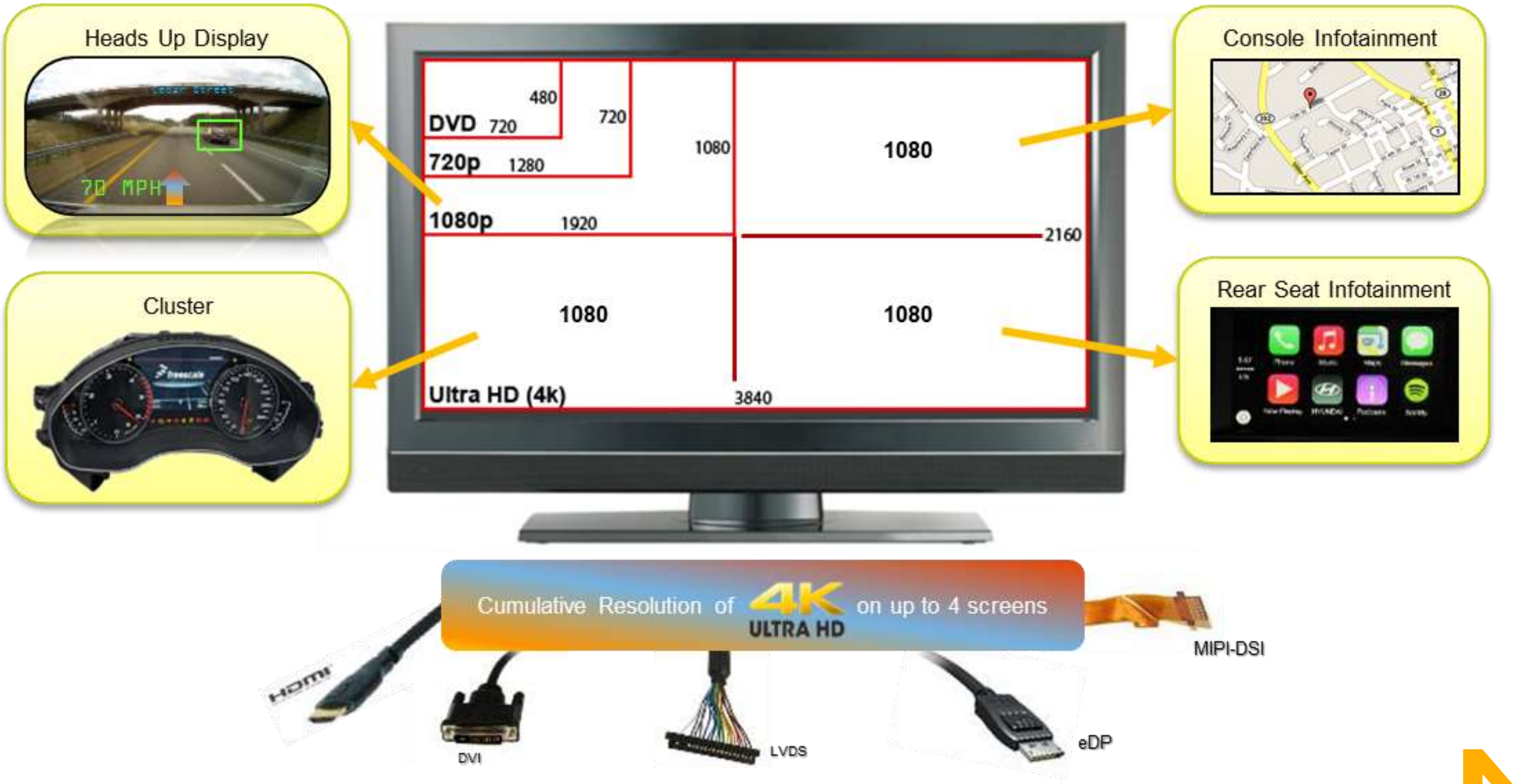
- **Vulkan** is a low-level CPU like language that is more "hands-off" in terms of fixed functionality.
  - Very efficient because it let's the user talk closer to the hardware. More dangerous… if the programmer doesn't know what they are doing.

- **Tessellation & Geometry shading** is a part of Android Expansion Pack (AEP) on OpenGL ES 3.1 and enabled in the OGL ES 3.2 standard.

- **Bare Metal Convergence** - Google, Apple, and AMD have all scrapped their proprietary plans to create a new low level graphics language. Google was going to develop "Radiance", AMD has "Mantle", Apple has "Metal", and Microsoft has DX12. Valve is funding LunarG to create an open source SDK for Vulkan with Khronos Group.

i.MX 8
DISPLAY
TECHNOLOGY

i.MX8 Display Support

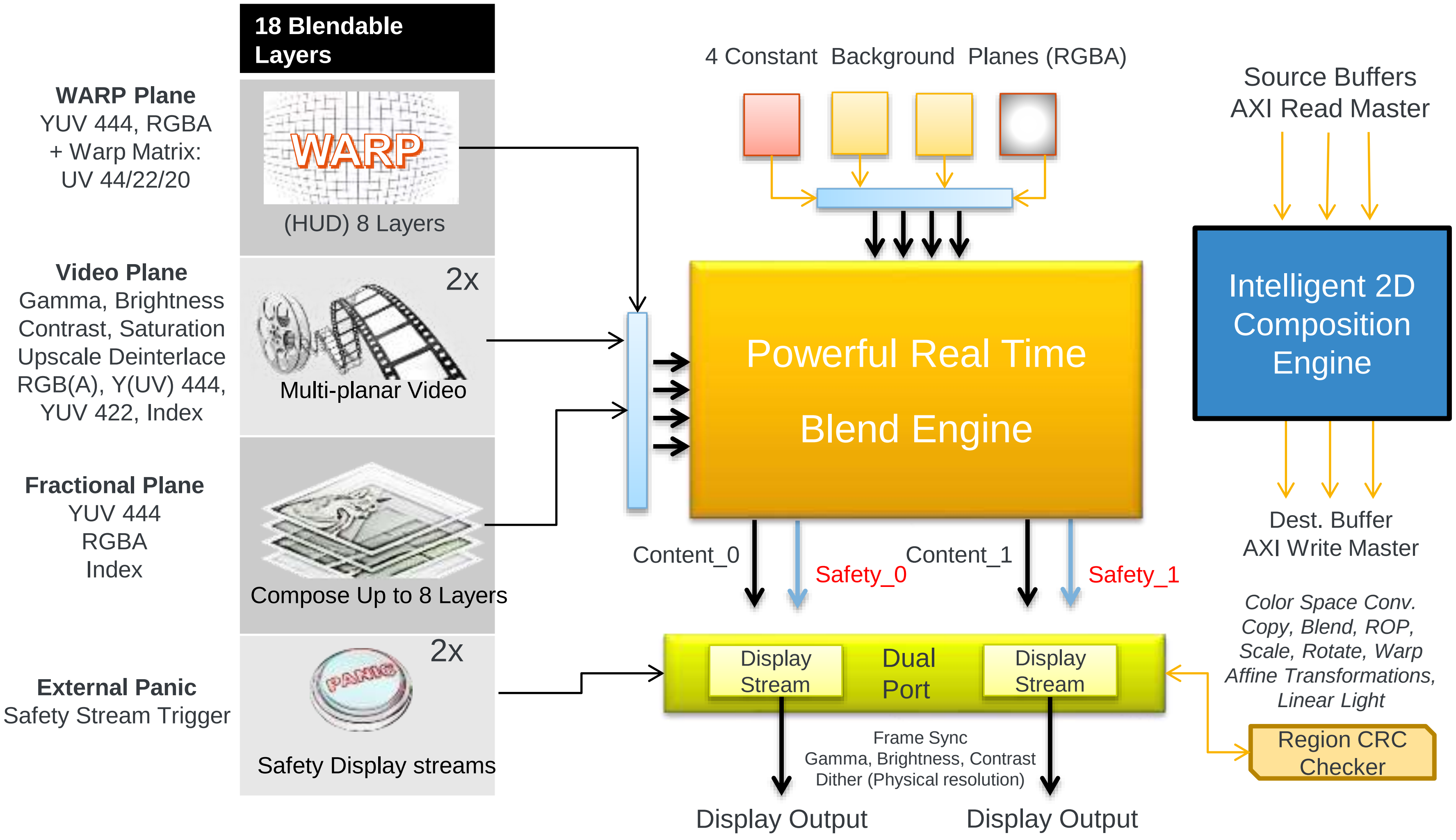


Display Output Capabilities (Dual Display Port)

300MHz pixel rate with 25% blanking distance for each port

Controller 1	Controller 2	Display Use Case Example	Total # of Displays	Max. Resolutions
☑	☒	Large Screen	1	3840 x 2160 (60 FPS) HW combined dual pixel port
☑	☒	2 Wide Screens	2	2880 x 1440 (60FPS)
☑	☑	3 Screens Wide 3 Screens Normal	3	3 @ 2880 x 960 (60FPS) or 3 @ 1280 x 1024 (120 FPS)
☑	☑	Multiple Screens Cluster, HUD, IVI	3	3840 x 720 (60FPS) 1280 x 720 (60FPS) 1920 x 1080 (60FPS)
☑	☑	Multiple Screens Cluster, HUD, IVI, RSE	4	1920 x 720 (60FPS) 1920 x 720 (60FPS) 1920 x 1080 (60FPS) 1920 x 1080 (60FPS)

i.MX 8 Display Controller



i.MX 8 Advanced Display Controller = Efficiency, Performance and Safety

- **2D Graphics Engine Support:**
  - Reduces burden on GPU: Allows the 3D GPU to be a 3D GPU
  - Plays nice with Video: Overlays native video and graphics with minimal trips to system memory
  - Saves power: 3D Engine can remain off for windowing GUI's (Android HW Composer)

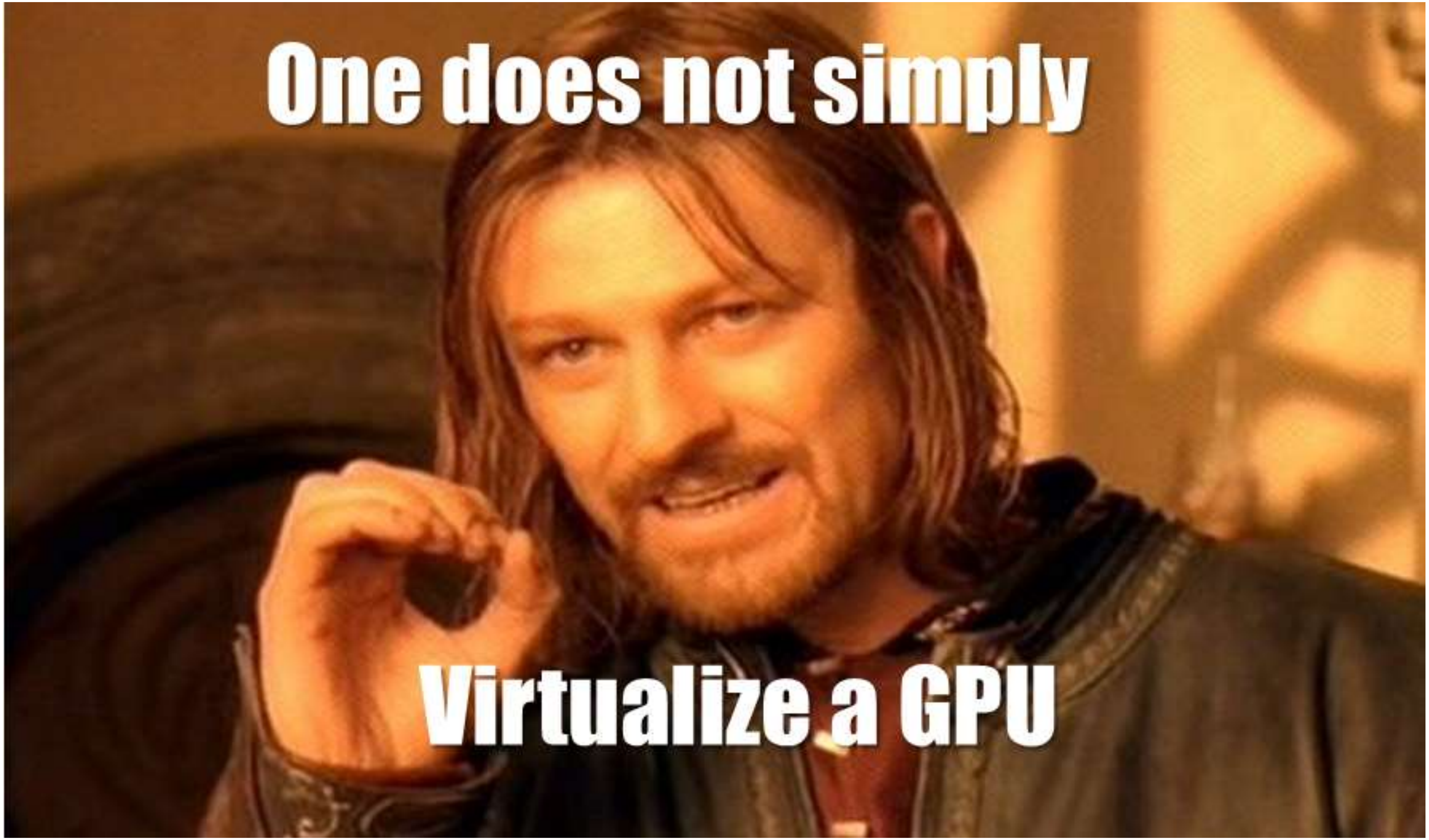
*Assumes: 2 complete sets of display processor + blitting engine.
800 Mpixels * 2 (fill-rate) + 300 Mpixels * 2 (Composition / Blend)

i.MX 8 2D Blitting Engine (32 bit)
3 Source Pipes and 1 Destination Pipe
800 Mpixels / second for fill-rate
400 Mpixels / second for copy, blend, ROP, Scale
200 Mpixels / second for rotate, warp
Matrix Filtering for Linear Color Conversion
Color LookUp Table for color buffer compression
Free angle rotation(90/180/270), flip, mirror

i.MX 8QM/QP/Q have over 2Gpixels/sec 2D Bandwidth*

i.MX 8 Display Composition Engine (32 bit)
2 Display Output Streams (independent panels)
10 bits per color component (30-bit resolution)
18 Total layers of Blend at 300 Mpixels / second
Real-time blend and Warp, no DDR passes req'd
Automatic Safety Stream Panic + Detection using CRC Matching

DESIGNING FOR SAFETY


One does not simply
Virtualize a GPU

Pitfalls of GPU Virtualization

- Increased Software Complexity
  - In the OS and kernel domains (debugging, stability)

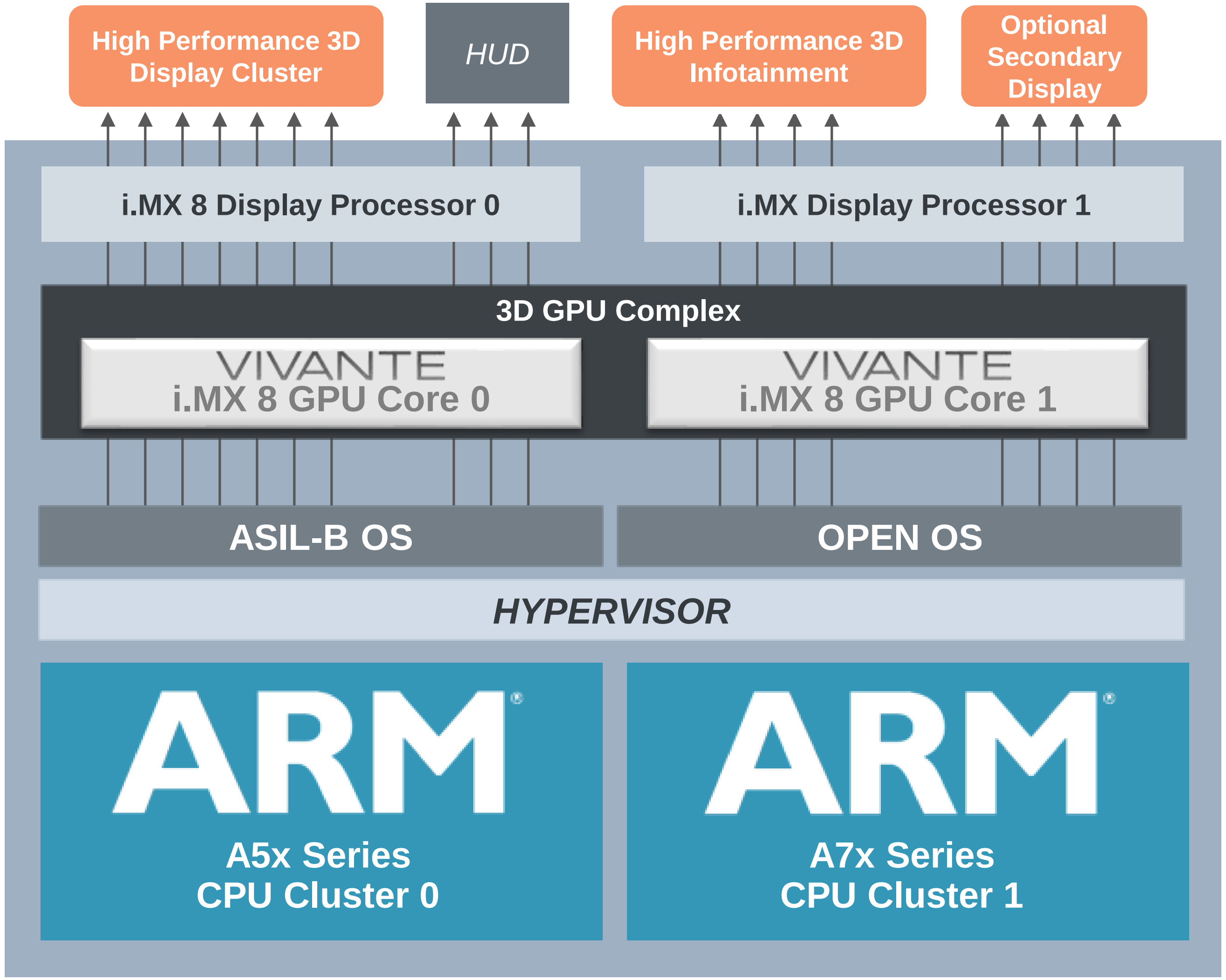
- Performance Hit
  - Virtualization overhead, especially with API trapping techniques and multiple context switching

- Monolithic GPU’s require extra HW
  - To hide the complexity of command buffer management and TDM (Time Division Multiplexing)

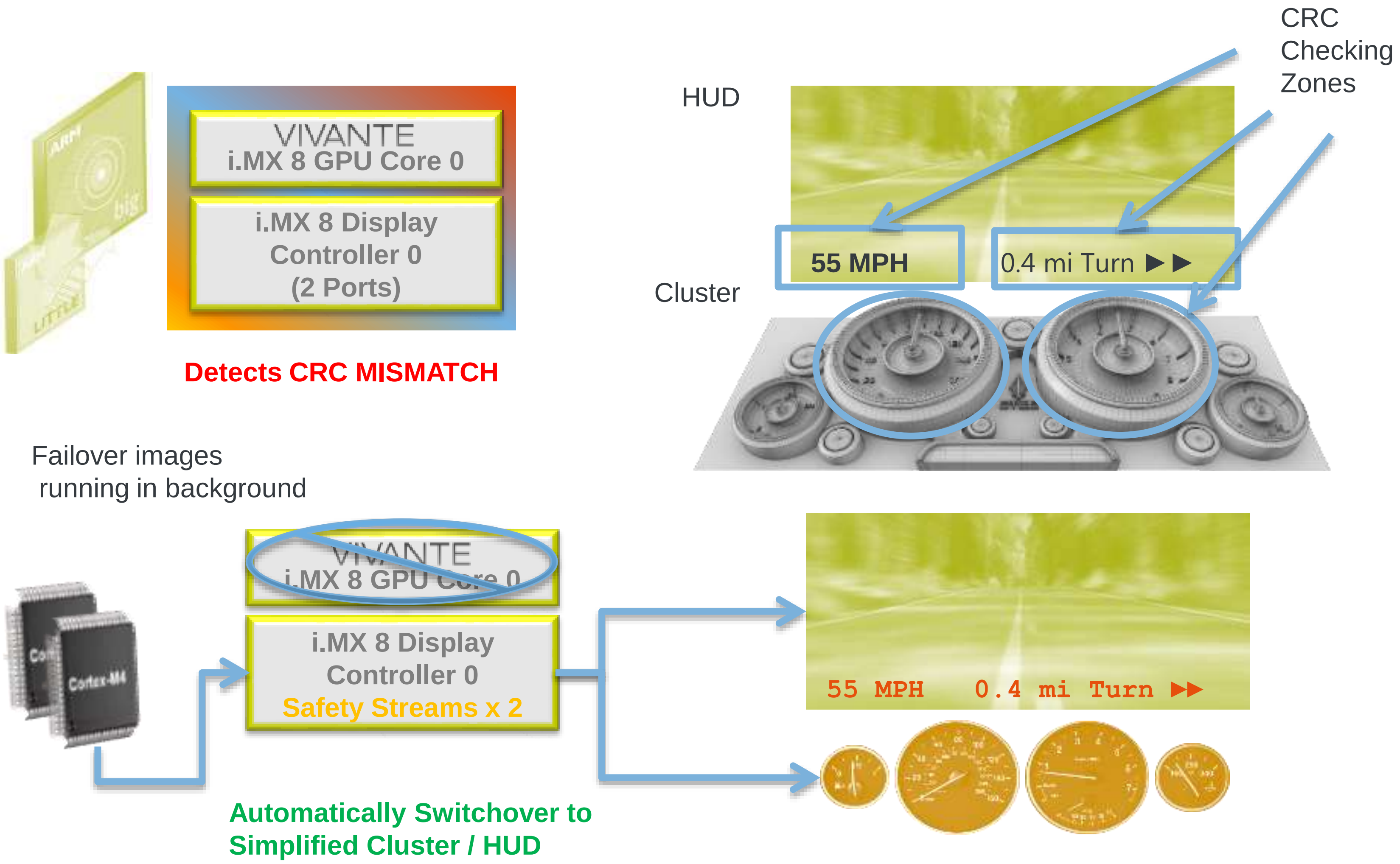
- If Monolithic GPU goes down, everything goes down
  - Means a driver bug fail in infotainment path can take down the cluster

i.MX 8 Family eCockpit Solutions

- **Advantages**
  - NXP i.MX 8 Series Vivante GPU cores are capable of independent operation and can run separate contexts - *NO COMPLICATED VIRTUALIZAION REQUIRED*
  - NXP i.MX 8 Series utilizes dual independent Display Processors with Safety Plane Support that can be accessed by independent GPU cores - *NO COMPLICATED VIRTUALIZAION REQUIRED*

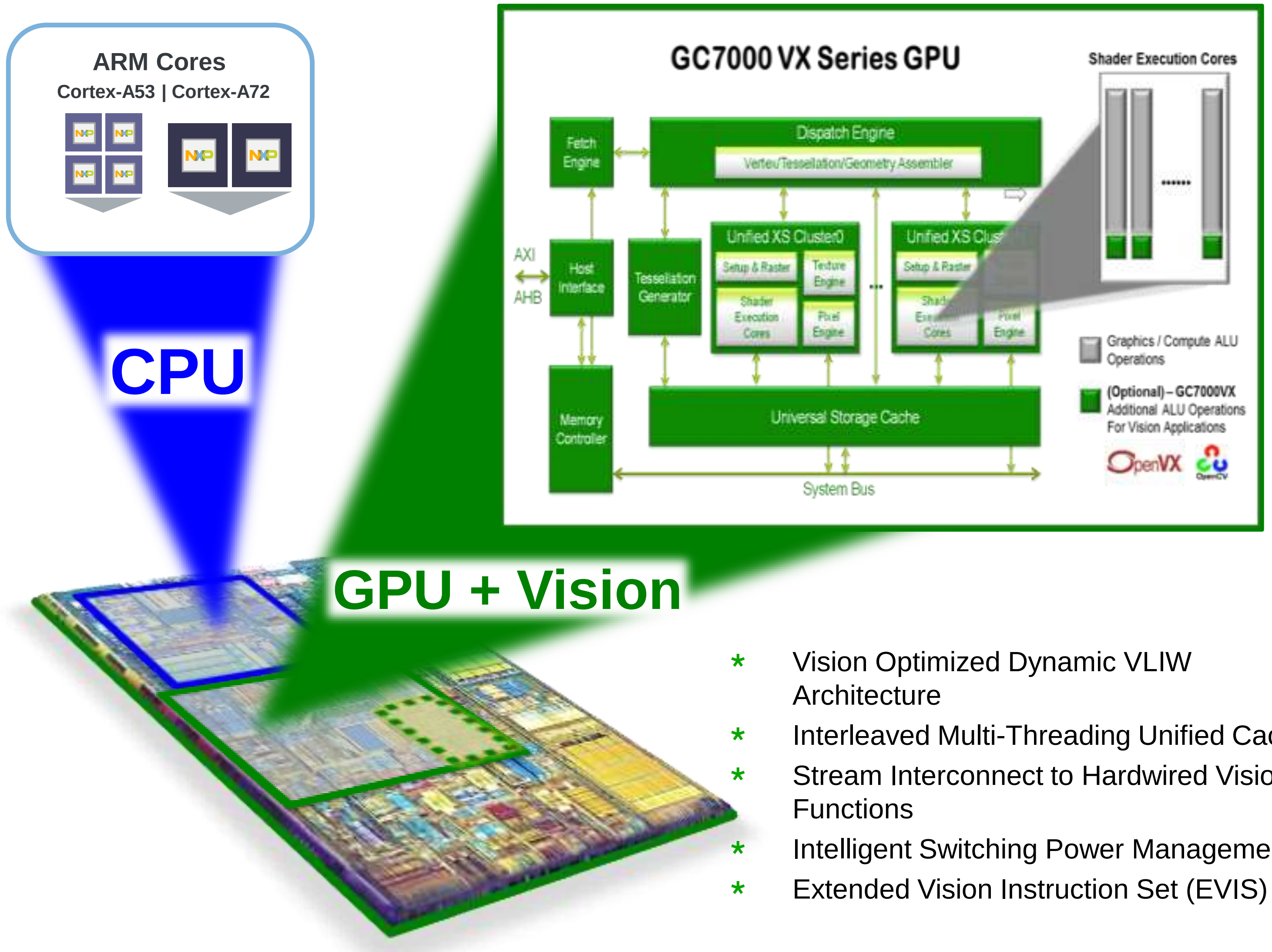


i.MX 8 e-Cockpit Cluster Safety Strategy



i.MX 8 SERIES VISION

Vision Processing ‘VX’ – the added Instruction Set



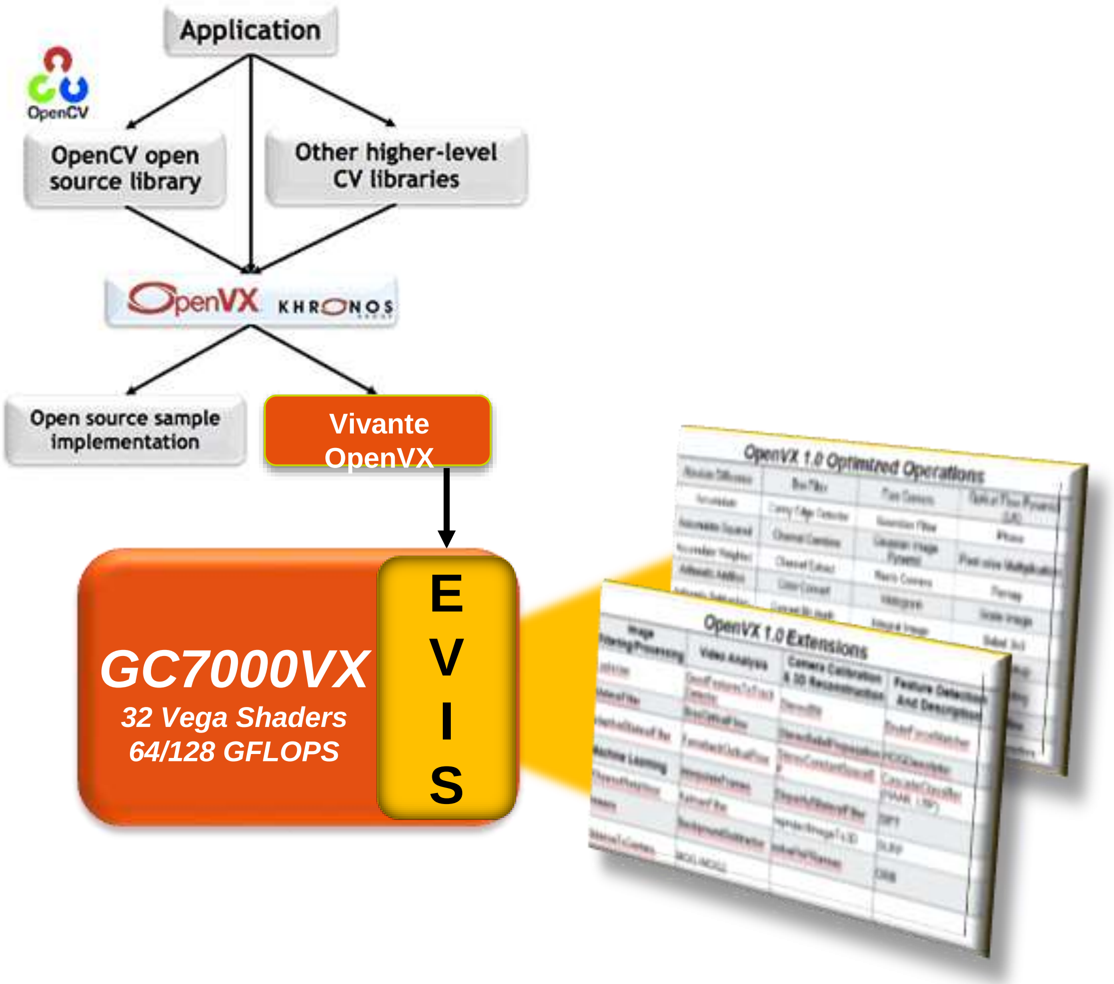
* Vision Optimized Dynamic VLIW Architecture
* Interleaved Multi-Threading Unified Cache
* Stream Interconnect to Hardwired Vision Functions
* Intelligent Switching Power Management
* Extended Vision Instruction Set (EVIS)

i.MX 8 Graphics Technology with Vision Extensions

- **OpenVX** is a Khronos royalty free vision acceleration API **designed to map OpenCV** and other higher level vision libraries into optimized hardware implementations

- **GC7000VX GPUs have EVIS™** hardware optimized GC7000 implementations and hardware extensions to **optimize the throughput of vision** acceleration via OpenVX **without an intermediate API layer**

- GC7000VX → **17x performance improvement** vs standard GPU (non VX) **implementations for vision algorithms**

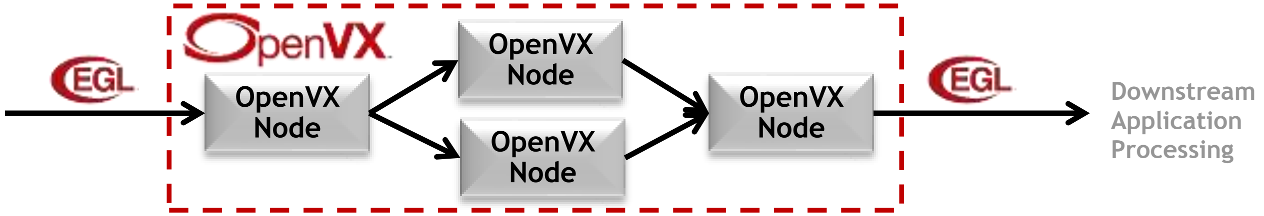


OpenVX and OpenCV are Complimentary

	OpenCV	OpenVX
Governance	**Community driven open source with no formal specification**	**Formal specification defined and implemented by hardware vendors**
Conformance	No conformance tests for consistency and every vendor implements different subset	Full conformance test suite / process creates a reliable acceleration platform
Portability	APIs can vary depending on processor	Hardware abstracted for portability
Scope	Very wide 1000s of imaging and vision functions Multiple camera APIs/interfaces	Tight focus on hardware accelerated functions for mobile vision Use external camera API
Efficiency	Memory-based architecture Each operation reads and writes memory	Graph-based execution Optimizable computation, data transfer
Use Case	Rapid experimentation	Production development & deployment

Programming OpenVX on the i.MX 8

- Directed Acyclic Graph (DAG) Framework Pipeline
  - Optimized precompiled kernels of commonly used vision processes
    - A subset of OpenCV that lends itself to HW Acceleration
      - HW Vendor can create hardened / silicon aware specialized kernels
      - App Developer can create unique shader-based kernels using OpenCL
  - OpenVX Graphs can split, join, delay, and produce callbacks depending on heuristics.
    - OpenVX Primitives include: **Images, Image Pyramids, Process Graphs, Kernels, Control Parameters**



Example OpenVX Graph

SECURE CONNECTIONS
FOR A SMARTER WORLD

ATTRIBUTION STATEMENT