GPU Verification Challenges

Zihno Jusufovic
zjusufovic@apple.com

April 9, 2020

Apple

Agenda

What's a GPU?

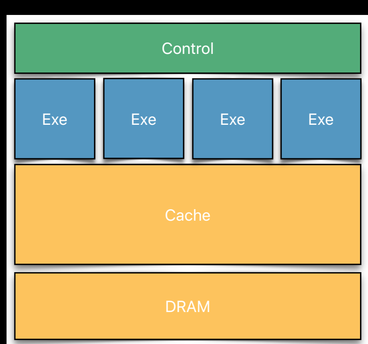
- Specialized co-processor to render 3D objects and simulate light effects
- Parallel processor leveraged for general compute, including machine learning
- Integrated HW-SW System

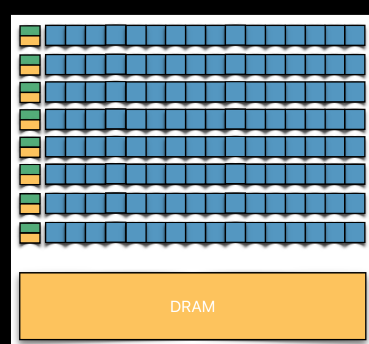
What are specific GPU DV challenges?

- Complex and evolving HW-SW features
- Rapid feature evolution
- Scaling of large GPUs
- Determining image correctness

What’s a GPU?

CPU vs GPU





CPU: Minimize *latency* of limited threads using complex control

GPU: Maximize *throughput* by scheduling many parallel threads

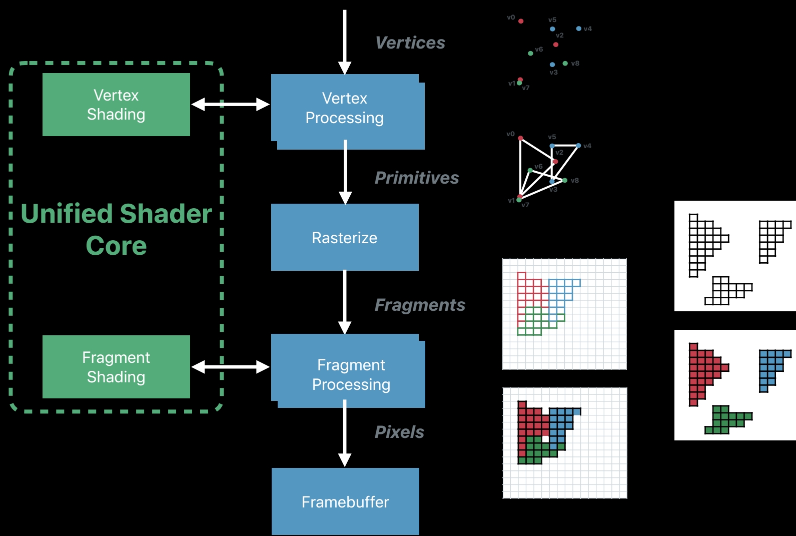
CPU vs. GPU

	CPU	GPU
Cores	Fewer	Many
Frequency	Higher	Lower
Latency	Low	Higher
Parallelism	Instruction Level	Thread Level
Registers	Fewer (Register Renaming)	Many
Speculation	Highly Leveraged (Br Prediction)	Limited (no Br Prediction)
Execution Order	Out of Order (Reorder Buffers)	In Order
Execution Units	Fewer	Many
Execution Control	Complex	Simpler
Coherency	Hardware Managed	Software Managed

GPU
Maximize Throughput, Hide Latency

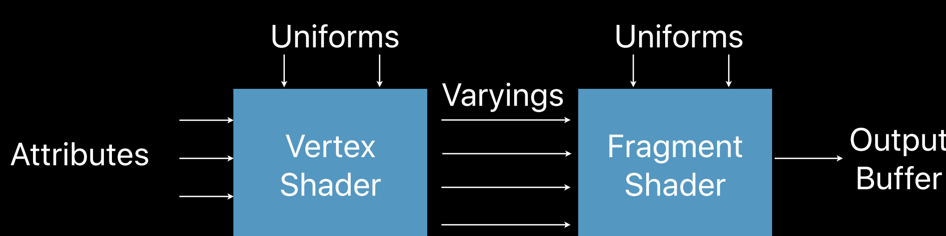
- High throughput SIMD leveraging *Thread Level Parallelism*
  - independent threads executing the same program
- *Threads* grouped into *Threadgroups* (*aka Blocks or Workgroups*)
  - shared local memory
- *Threadgroups* broken into *SIMD Groups* (aka *Warps or Wavefronts*)
  - identical, independent, lockstep programs of multiple threads
- High memory latency, high memory bandwidth
  - switch *SIMD Group* execution to maintain high shader occupancy

Simplified 3D Graphics Pipeline Diagram



Shaders

- Programs that execute on the GPU
- Written in a C-like shader language
- Compiled, linked and combined into a larger program

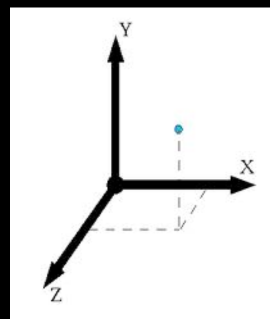


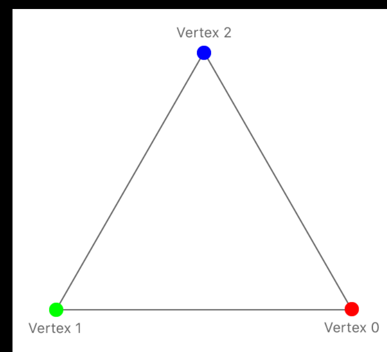
3D Full Frame

Vertex Processing

Vertex Processing
Vertex Attributes

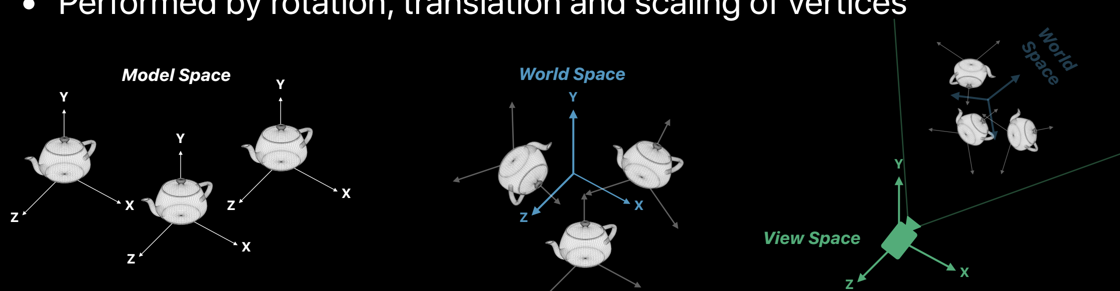
- Position Coordinates
- Color & Alpha (RGBA)
- Normals
- Texture Coordinates
- Material Properties for lighting





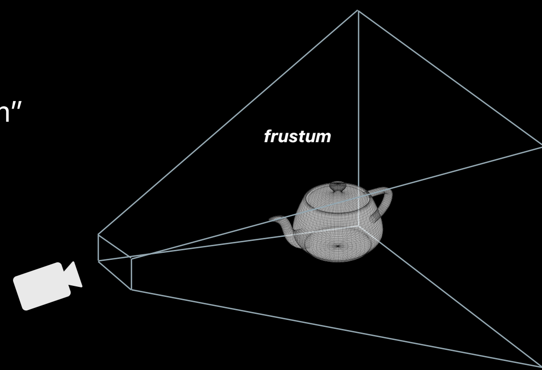
Vertex Processing
Transformation and Projection

- **Model Transform** : arrange objects in the world
- **View Transform** : position and orientation of the camera
- Performed by rotation, translation and scaling of vertices



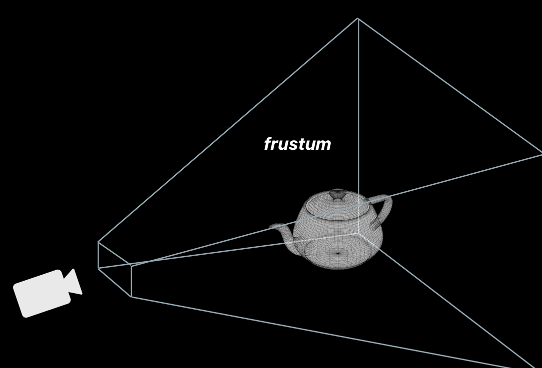
Vertex Processing
Projection Transform

- Projection space
  - set the camera's field of view, "frustum"
  - adjust focal length
  - adjust zoom factor
- Perspective projection
  - adjust for distance



Vertex Processing
Clipping and Culling

- Culling
  - remove hidden triangles due to backface, occlusion, or view frustum
- Clipping
  - clip triangles that intersect the frustum
  - interpolate the attributes to the new vertices



Vertex Processing

Key Concepts

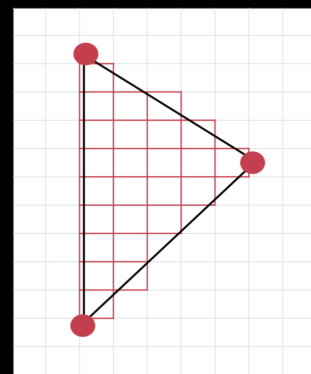
- Primitives
  - points, lines, triangles
- Vertex Shading
  - determine position, transformations, evaluate attributes
- Transformation and projection
  - rotation, translation, and scaling of geometry
  - distant objects are smaller
- Clipping and culling
  - remove offscreen and hidden geometry for efficiency

Rasterization

Rasterization

Translate 3D image to 2D

- Determine the pixels in the screen covered per triangle
- Potential pixels are called “fragments”

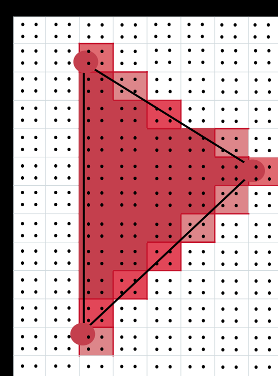


EE 382M-11, Lecture 21, 9th April 2020

Anti Aliasing

Image Quality Feature

- Anti aliasing helps reduce pixelation by blending edges
  - multiple techniques
- Multi Sampling AA (MSAA)
  - blend color for sub pixels with multi-triangle coverage
  - requires coverage mask, resolved in shader
  - single shader execution per fragment
  - complicates SIMD execution



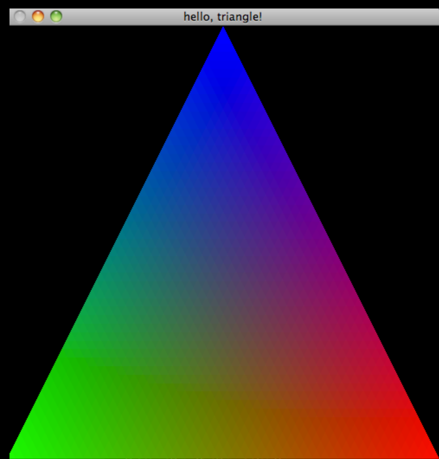
EE 382M-11, Lecture 21, 9th April 2020

Rasterization
Key Concepts

- Triangle Setup/Primitive Assembly
  - Differentials, edge equations
- Triangle traversal
  - iterate and check each sample for coverage
  - generate fragments
- Anti-Aliasing
  - multiple techniques with different trade-offs

Fragment Processing

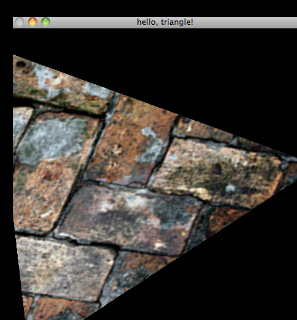
Fragment Processing
Simple Shading



Fragment Processing
Texture Mapping

- Textures are saved images or other data
  - "texels" = texture pixels

- Map texels -> pixels on geometry
  - box filter - nearest neighbor
  - bilinear - interpolate in 2D
  - bicubic - weighted sum of local texels
  - trilinear - interpolate in 3D
  - mipmaps - lower resolutions of texture



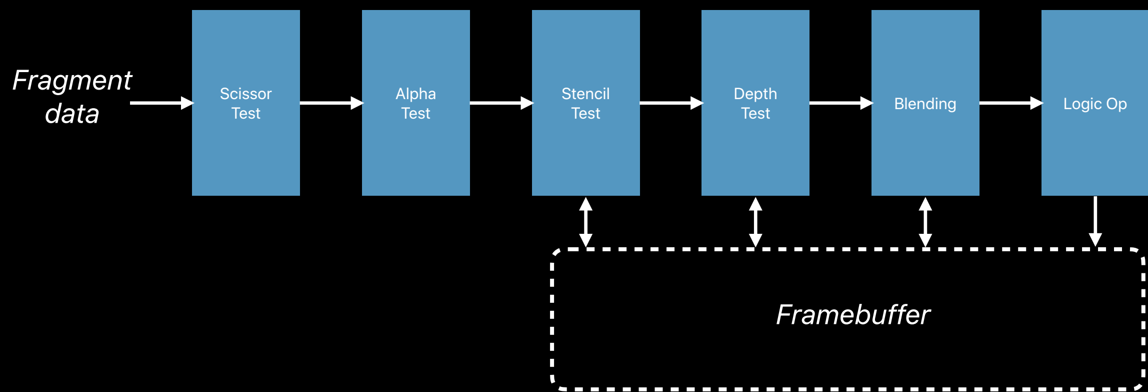
Texture Processing
Other uses for textures

- Normals
- Shadows
- Material properties

Fragment Processing
Lighting - Phong Model

- **Ambient Lighting**
  - Approximates global illumination
- **Diffuse Lighting**
  - Direct impact of light on an object
- **Specular Lighting**
  - Bright spots on shiny objets

Fragment Processing
Simple Per Fragment Tests



Fragment Processing
Key Concepts

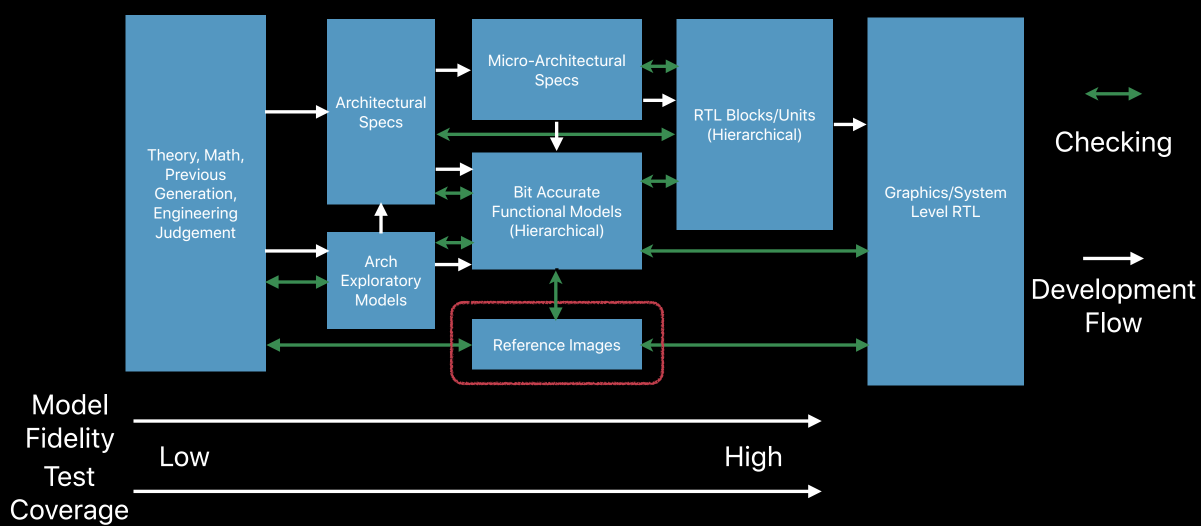
- Interpolation of fragments
  - simplest determination based on vertex attributes
- Per pixel shading computations
  - creates more complex effects than simple interpolation
- Texture Mapping
  - mapping an image onto a triangle
- Lighting
  - how surfaces respond to light and project colors
- Per Fragment tests
  - check each pixel location

GPU DV Complexity

GPU DV Challenges

- **Complex system of hardware + software**
  - Driver manages hardware directly->hardware not fault tolerant
  - Software will continue to optimize through hardware lifetime
  - API does not dictate pixel-level accuracy
- **Rapid feature evolution**
  - Graphics algorithms and techniques continuously innovated and enabled
  - GPU ISA evolves rapidly
  - Performance optimizations, workload distribution change with workload evolution
- **Full system model scaling**
  - footprint and run time
- **Determining Image Correctness**
  - many units with approximations and imprecision

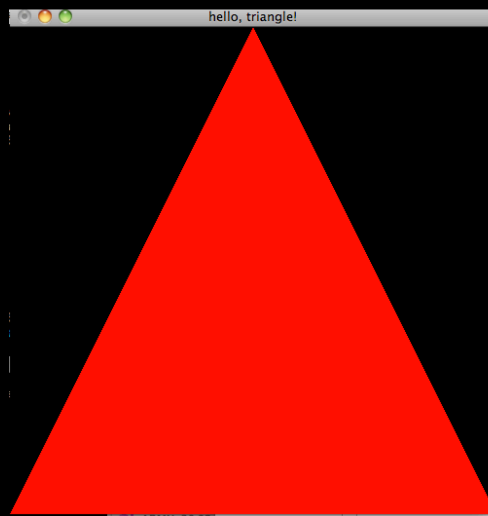
GPU Verification Flow



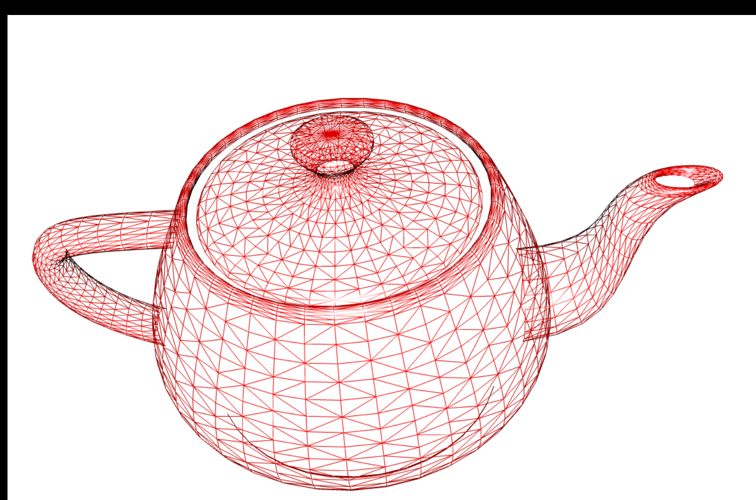
Multi-faceted RTL DV Approach

- **Hierarchical Simulation**
  - module->block->sub-system->GPU level
  - intensive directed and directed-random
- **FV Property Checking**
  - small blocks, design components, bug hunting, ECO validation
- **Emulation/FPGA**
  - Large GPU workloads & Full Frames
  - Performance and Power
- **Full Virtual System with Software**
  - ensure HW-SW validation
  - silicon readiness

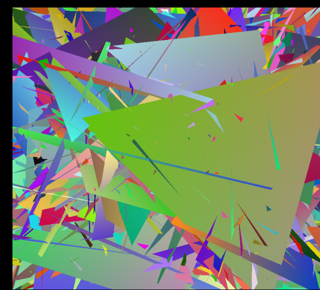
Common First Test
Hello, Triangle!



Classic 3D Graphics Model
Utah Teapot

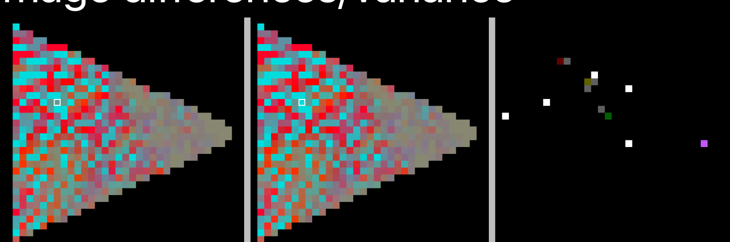


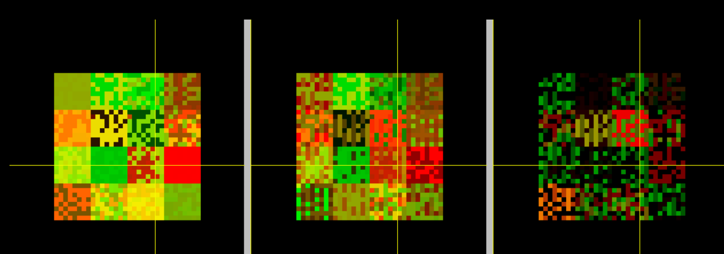
Establishing reference images



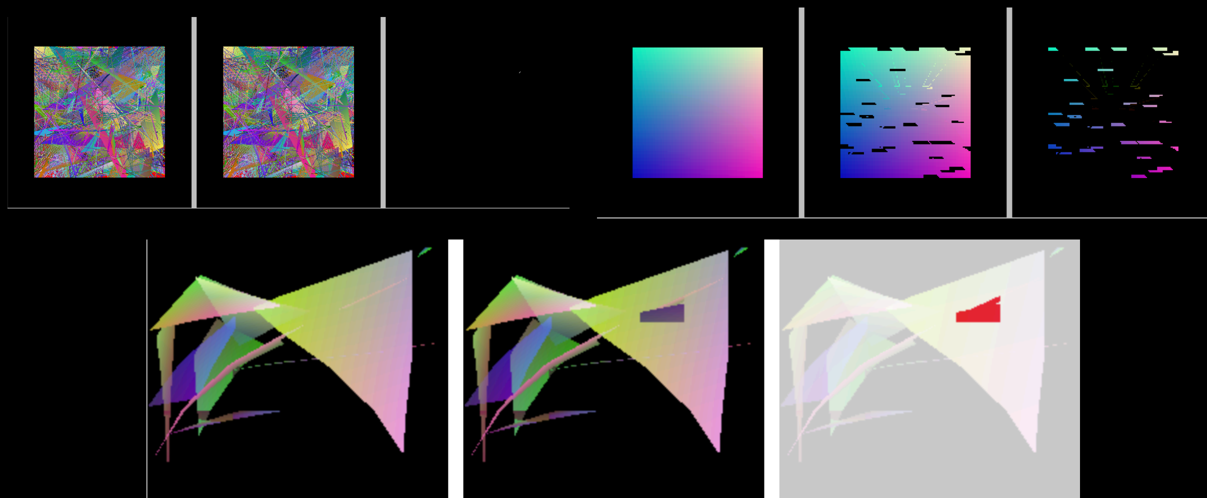
Determining Image Correctness

Image differences/variance

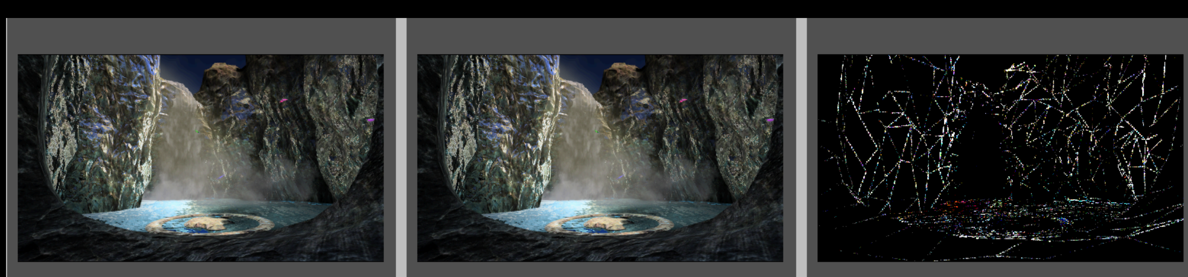




Determining Image Correctness
Image differences/variance

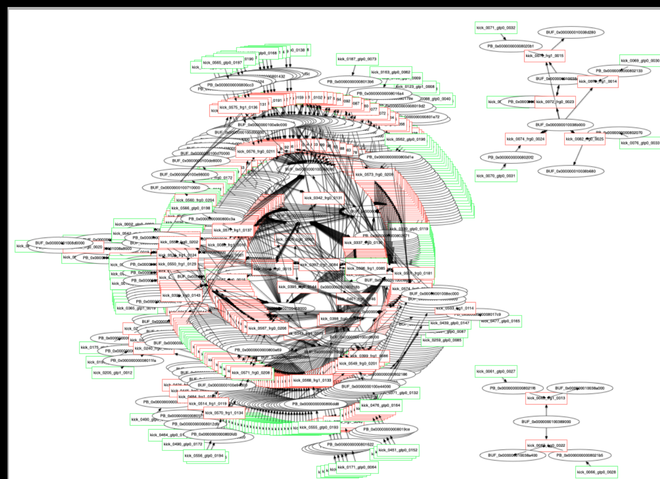


Determining Image Correctness
Image differences/variance



Debugging Images

Dependencies can be complex!



Determining Image Correctness

Key Takeaways

- **Many combined approximate techniques to create an image**
  - Image validation insures the resultant image meets expectations
- **Resulting image has subjective aspects**
  - Multiple results may be OK
  - Understanding HW/SW/application intent is critical
- **Verified hierarchically — system level results need special validation**
  - Modules, Blocks and Sub-systems verified with traditional DV, FV
  - GPU level directed-random stress with predictable results
  - Image validation focused on directed testing
  - Full system validation ensures HW-SW correctness and performance
- **Frame to frame smoothness important**

Important DV skills

- **Architecture & Micro-architecture**
  - Know how it all works and what might go wrong
- **Software Engineering**
  - Efficiency in architecting, simulating, scaling, debugging complex OOP environments
- **Probability**
  - Grit and paranoia about rare cases, but prioritize the common ones
- **Psychology**
  - Question, challenge; creativity, curiosity

EE 382M-11, Lecture 21, 9th April 2020