National Aeronautics and Space Administration

Body of Knowledge for Graphics Processing Units (GPUs)

NASA Electronic Parts and Packaging (NEPP) Program

Edward Wyrwas
Lentech, Inc
Work performed for NASA Goddard Space Flight Center
Greenbelt, Maryland

Executive Summary

Graphics Processing Units (GPU) have emerged as a proven technology that enables high performance computing and parallel processing in a small form factor. GPUs enhance the traditional computer paradigm by permitting acceleration of complex mathematics and providing the capability to perform weighted calculations, such as those in artificial intelligence systems. Despite the performance enhancements provided by this type of microprocessor, there exist tradeoffs in regards to reliability and radiation susceptibility, which may impact mission success. This report provides an insight into GPU architecture and its potential applications in space and other similar markets. It also discusses reliability, qualification, and radiation considerations for testing GPUs.

Table of Acronyms

2D	Two Dimensional
2.5D	Two and a Half Dimensional (refers to projections)
3-D	Three Dimensional
ADAS	Advanced Driver Assistance Systems
ALU	Arithmetic Logic Unit
API	Application Programming Interface
APU	Advanced Processing Unit
BGA	Ball Grid Array
BI	Burn-In
Cat5e	Category 5e (enhanced) specification
CDS	Cockpit Display System
CMOS	Complementary Metal-Oxide Semiconductor
COTS	Consumer off the Shelf
CoWoS	Chip on Wafer on Substrate
CPU	Central Processing Unit
CTE	Coefficient of Thermal Expansion
CUDA	Compute Unified Device Architecture
CUFFT	CUDA Fast Fourier Transform library
DDR	Dual Density RAM
DHCP	Dynamic Host Configuration Protocol
DRAM	Dynamic random-access memory
DUT	Device Under Test
ECC	Error Correcting Code
EGL	Embedded-System Graphics Library
EO/IR	Electro-Optics & Infrared
ES	Embedded Systems
FPGA	Field Programmable Gate Array
GCR	Galactic Cosmic Rays
GPGPU	General Purpose Graphical Processing Unit
GPU	Graphical Processing Unit
GUI	Graphical User Interface
HBM	High Bandwidth Memory
HDMI	High-Definition Multimedia Interface
HPC	High Performance Computing
HPEC	High Performance Embedded Computing

IP	Internet Protocol
LANSCE	Los Alamos Neutron Science Center
MGH	Massachusetts General Hospital
NASA	National Aeronautics and Space Agency
NRL	Naval Research Laboratory
OCM	Original Component Manufacturer
OEM	Original Electronic Manufacturer
OpenGL	Open Graphics Library
OpenCL	Open Computing Language
PCB	Printed Circuit Board
PLD	Programmable Logic Device
PTX	Parallel Thread Execution
RAM	Random Access Memory
RJ45	Registered Jack #45
RTOS	Real Time Operating System
SBST	Software Based Self-Test
SDK	Software Development Kit
SEE	Single Event Effects
SEFI	Single Event Functional Interrupts
SET	Single Event Transient
SEU	Single Event Upset
SIGINT	Signals Intelligence
SIP	System in Package
SKU	stock keeping unit
SOC	System on Chip
SOI	Silicon on Insulator
SRAM	Static Random Access Memory
SWAP	Size, Weight, and Power
TFTP	Trivial File Transfer Protocol
TSMC	Taiwan Semiconductor Manufacturing Company
TSV	Through Silicon Via
UFRGS	Federal University of Rio Grande do Sul
USB	Universal Serial Bus
WLBT	Wafer-Level Burn-in Test

Table of Contents

1 Purpose

The purpose of this document is to provide guidance on Graphics Processing Units (GPU) and their underlying technology, in regards to their potential usage in future space flight systems. This document will discuss an overview of what a GPU is, how they are used in computing systems, their susceptibility to degradation and radiation effects, due to their advanced lithography nodes, and the direction of the technology.

1.1 The Origin of the GPU

A GPU, by definition, is a graphics processing unit. It is a specialized circuit, or single-chip processor, designed to accelerate the image output in a frame buffer intended for output to a display. In the late 1990s, these types of devices were synonymously referred to as video processing devices. Initially, they were capable of rendering individual pixels within a 2-D display using triangles. It created lighting effects and transformed objects every time a video scene (single video frame) was redrawn. These are mathematically-intensive tasks which would otherwise put quite a strain on the Central Processing Unit (CPU) or other main processor in the system. Lifting this burden from the CPU frees up clock cycles that can be used for other tasks. In CPUs, the processor takes a set of instructions and computes a task. Within the CPU are several modules that perform arithmetic operations, such as multiplication of integer values, memory or register operations, and things like calculating the square root of a value. These modules are referred to as arithmetic logic units (ALU). The ALU offloads work from the main CPU logic cores. A GPU core is similar to an ALU. They are designed to accelerate specific tasks assigned to it. GPUs are very efficient at manipulating computer graphics and are generally more effective than general-purpose CPUs at matrix manipulation. GPU algorithms allow processing of large blocks of data to be done in parallel.

The first company to develop the GPU was Nvidia Inc. in 1999. The GeForce 256 GPU was capable of billions of calculations per second, could process a minimum of 10 million polygons per second, and had over 22 million transistors, compared to the 9 million found on the Pentium III, which was the leading edge CPU at the time[1]. In the last decade, GPU uses have expanded to physics, 3-D texturing, shading/shadows, Bit Coin mining, hash-code manipulation and encryption, as these tasks require repetitive or recursive mathematical functions.

1.2 New Purposes for GPUs

GPGPU[2] stands for General-Purpose computation on Graphics Processing Units*, also known as* GPU Computing. GPUs are high-performance, many-core processors capable of very high computation and data throughput. Once specially designed for computer graphics and difficult to program, today's GPUs are general-purpose parallel processors with support for accessible programming interfaces and industry-standard languages, such as C. Developers who port their applications to GPUs often achieve speedups of orders of magnitude versus optimized CPU implementations.

While similar to GPUs, microcontrollers and microprocessors are not native parallel processors. Mainstream CPUs, such as Intel's Xeon product line, have up to 24 cores which can process a hefty instruction set. Some, such as Intel's Knight's Landing[3], act as co-processors which can offload some specific instructions from the main microprocessor. Smaller cores, such as those in GPUs, containing several pipelining threads are useful when performing repetitive and similar calculations that don't require a lot of memory access or multitudes of instructions. A GPU can contain thousands of cores, with cores inside of cores, which perform a variety of small calculations concurrently.

Instruction sets have been developed such as Nvidia's Compute Unified Device Architecture (CUDA), a programming language similar to C, which allow porting of traditional code from CPU-borne operating systems to GPU hardware. A standard unified programming interface can make it a better choice than FPGA, wherein the GPU's compiled code is synthesized for a code architecture (i.e. Open CL 1.1) rather than specifically for the make of a device. Standard languages allow compilation of code similar to the x86 architecture, where code will compile on Intel, AMD and other vendor x86 devices.

Computer graphics have exponentially become more realistic over the last decade. Triangles have been replaced with meshing polygons. The graphics engine can compute shading, shadows, textures and even fluid flow. Better designs have been created for pipelines and cores. With these better ALU, it became possible for general native languages (i.e. C) to run on these devices. One such example is Nvidia's CUDA language, which was devised to offer a universal application programming interface (API) to offload code onto the GPU cores.

1.3 GPUs in Non-Space Environments

The best notable early achievements in automotive and vision systems have been seen in news media since 2010. Luxury vehicles and now economic cars offer lane detection to facilitate lane departure warnings for the driver and to augment the driver's heading control in lane keeping assist systems. Further, detection and tracking of other vehicles driving ahead was made possible using adaptive cruise control systems and forward-facing sensors. Pre-crash systems, such as GM's "Low Speed Front Automatic Braking[4]," enable braking to lessen damage if a driver reacts too slowly.

For all of these cool features to exist and work in real time, it is necessary to use multi-core processing devices or a distributed computing system to have system-level environmental data processed in real time. Parallel processing is required for visual sensors to perform real-time object detection, tracking, identification and classification. Different challenges exist to identify static and dynamic objects in a discrete way over time. Volvo[5] and other automotive companies have begun to use GPUs for their vision systems.

In stationary installations, GPUs are being used as coprocessors for accelerated computing research. Nvidia's deep learning platform is one such commercial example in which they use parallel GPUs as a neural network to teach autonomous vehicles how to "think" for themselves. Oak Ridge National Laboratory's Titan supercomputer uses GPUs to boost the parallelism of their scientific algorithms, such as simulating nuclear reactions. Multiple academic institutions employ GPU-based clusters to perform other simulations. Further, image processing from satellites can be performed much quicker, as seen in a National Oceanic and Atmospheric Administration (NOAA) evaluation. Coral Reef Watch conducted an investigation to evaluate the performance differential between a recent high-end 8-core Intel CPU and a 480-core NVIDIA GPU for two representative satellite SST data processing tasks[6]. The GPU processing shows a marked improvement (2.4 – 3.3 faster in completion time) over the CPU.

1.4 GPUs for Aerospace and Space

In aerospace, manufacturers are trying to reduce the number of embedded computer boards within an airplane. Therefore, parallel processing boards are starting to find a home in these vehicles. Aerospace vehicle displays are also a critical feature in a manned aircraft. The ARINC-661 Standard for Cockpit Display Systems (CDS) describes the definition of interactive CDS and

the communication between the pilots through the CDS itself. Display technology is diverse and advances so rapidly that by the time systems can be designed, certified and released into the market, it is often obsolete in consumer electronics terms. To prevent this in aircraft, ARINC-661 permits use of a software GPU, which helps eliminate the need to invest in lifetime buys of components which would otherwise be necessary for a unique hardware solution. In the consumer drone market, DJI provides an Nvidia TK1 GPU-powered embedded computer for "advanced image processing in the sky"[7]. For ground support, companies such as Curtiss-Wright's Defense Solutions division have introduced a rugged graphics processing unit (GPU) module for deployed radar, signals intelligence (SIGINT), and electro-optics/infrared (EO/IR) high-performance embedded computing (HPEC) applications[8].

The first satellite to include a GPU for image compression was COROT. COROT is a French national space agency (CNES)-led mission to detect exoplanets orbiting other stars and to probe the mysteries of stellar interiors. Other satellite programs have used GPUs. A primary example purpose herein is orthorectification (a process that removes the geometric distortions introduced during image capture and produces an image product that has planimetric geometry, like a map). Image projection and registration are very slow with standard CPU processing. This can delay a craft from using images to make time critical decisions where location is important. The Naval Research Laboratory (NRL) conducted work to develop an aircraft detection system[9], which uses GPUs for processing high-rate video for real-time identification of targets.

2 GPU Architecture

GPUs can be packaged in multiple ways. A few examples are provided in the following figure.

Figure 1 GPUs in Consumer Devices[10]

2.1 Cores and Pipelines

GPUs and coprocessors have smaller cores than CPUs. Smaller means that they are not configured to allow the same relative quantity of instructions as a CPU. Although smaller, the GPUs tend to have much longer pipelines. A core is the portion of a microprocessor that is the

smallest independent processing unit. The cores read instructions to perform specific actions. In traditional CPUs, the instructions are serially executed within the device accessing memory for each piece of data required for the task. In GPUs, a computational task is broken down into several similar subtasks that can be processed independently and whose results are combined afterwards, upon completion[11].

A processor pipeline is similar to an assembly line. It contains a set of serially connected processing elements. The workload of building a product gets divided into smaller sub-tasks. Each step of the build process must be performed sequentially for the product to be produced to specification. In core logic, this process is paired with clock cycles. For each clock cycle a single sub-task gets completed at any stage within the pipeline. This permits multiple instructions to come down the pipeline sequentially. While it does not decrease the processing time for a single data instruction, it increases the throughput of the system when processing that data. The pipeline is only as quick as the slowest part.

Ineffective Pipeline
With an ineffective pipeline design, the pipeline may not make the throughput faster. For example, addition takes four clock cycles and multiplication takes eight. If the order of operations dictated that you must add before the multiplication, then perform the second addition, then a total of sixteen clock cycles would be needed for the operation.

Effective Pipeline
In the same example, but with a parallel pipeline, we could compute two sets of addition in the time it takes to compute one set of multiplication. If these tasks are independent of each other, then multiplication and two bouts of addition can take place in eight cycles. If we assume the second addition operation could theoretically take place while the multiplication task is running, then we could state that this is a more effective pipeline, than described above.

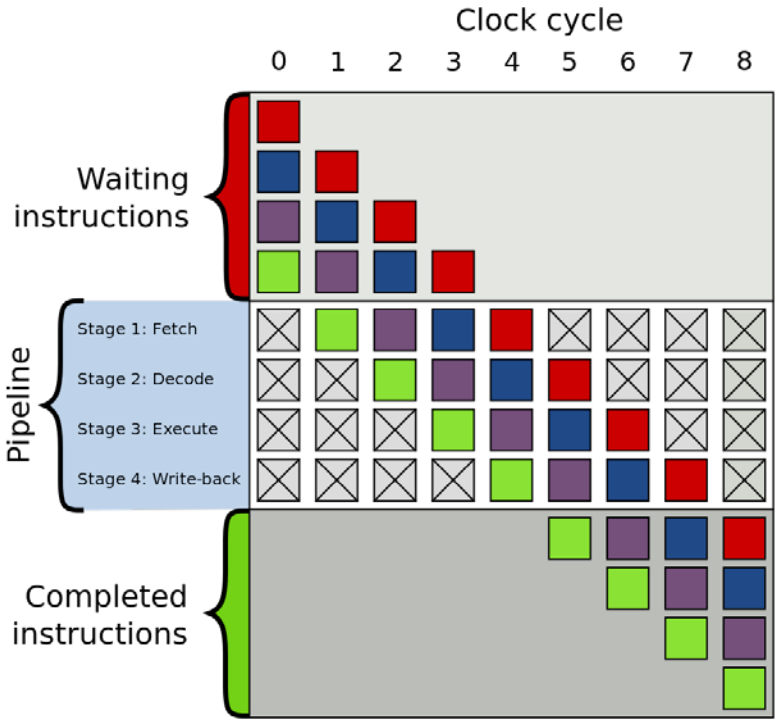

Figure 2: Illustration of Processor Pipeline[12]

In parallel processing, the cores are set up in a cell matrix. Each core must have access to a shared pool of memory such that any number of cores can perform functions on pieces of a set of or simply the same data. Keeping track of whose turn it is and whether the data is out-of-date (per other operations taking place) is integral to the system. Memory coherence is a computing phenomenon that exists when multiple processing units need to access the same shared memory. The integrity of that shared memory (typically off-chip DRAM or local SRAM cache) is quite important to avoid data corruption. As GPUs evolve, higher capacity and faster memories are required to keep things moving smoothly.

2.2 GPU Organization

To use a graphics processor, the system still needs a host CPU to run the operating system, supervise the GPU, provide access to main system memory (on board DDR), and to provide a user interface. The cores within the GPU can have multiple names, notably programmable shaders or stream processors. Typically there are multiple cores that are grouped together to form these stream processors. As such, they have shared common resources such as local memory, registers, and schedulers. GPU cores are very simple processing engines designed to execute only one instruction at a time per thread then sequentially switch to another thread once the computation is complete. Bundles of threads are called warps and typically consist of 32 threads.

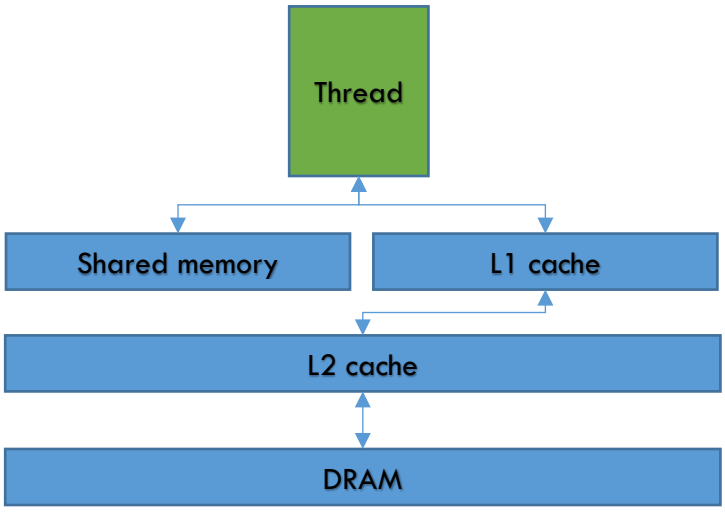

Figure 3 GPU Typical Memory Hierarchy

The memory hierarchy of a GPU is shown in Figure 3. GPUs and CPUs have different proximities to system memory. CPUs have a direct path to main memory through an integrated memory controller. Sometimes a motherboard's north bridge or system chipset will provide this functionality. GPUs must send a request to the host CPU over the bus to access this memory. Because there is a long latency inherent with this type of transaction, GPU programmers tend to use local GPU resources to mitigate performance hits. In some leading edge GPUs, a new feature called unified memory architecture permits a direct access from GPU to system memory over the bus.

2.3 GPU Physical Construction

A typical package of a GPU is a ball grid array (BGA) that is soldered to a printed circuit board (PCB). An illustration is shown in Figure 4. The GPU must be treated as a system though as it

contains the GPU die, DRAM, passive chip components and an interconnect structure. Details regarding these components are found in the appendix.

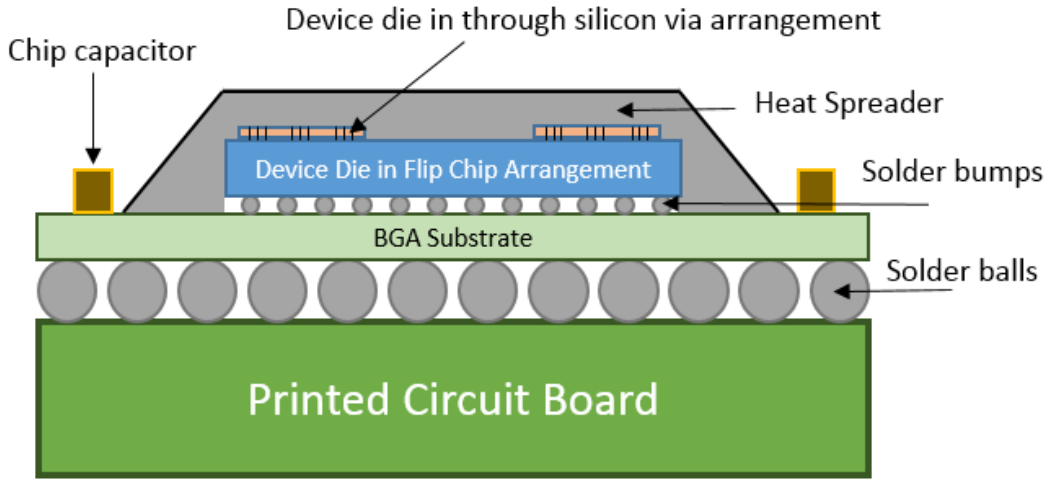

Figure 4: Illustration of Ball Grid Array Connections on GPU Device

2.4 Chip on Wafer on Substrate (CoWoS) and Stacked Die

Most of the current microchip packaging for GPUs consists of a single monolithic device in a plastic ball-grid array package. This is typical of CPUs and application processors. Traditional packaging offers minimal value to leading edge GPU applications, where bandwidth is mission critical. The best packaging seems to be no packaging or one package for multiple devices such as with system in package (SiP). The end goal of system scaling is to enable entire systems in a single package. The concept is a 3D system package based on system scaling and heterogeneous integration.

Through Silicon Vias (TSVs) make 3D integration possible. An illustration of this is provided in Figure 5. Improved performance comes from ultra-short copper interconnections between the chip and substrate. If multiple chips are to be utilized, then keeping them as close as possible will increase performance. We achieve this with 2.5D and 3D integration. This is different from multi-chip modules in that there is not an FR4* (organic, plastic) substrate to fan out its interconnections. Rather, silicon is used to keep the coefficient of thermal expansion (CTE) the same as the active die. A great example of this is Nvidia's P100[13], which contains a large GPU die and four TSV-stacked DRAM die on a silicon interposer.

* FR-4 (or FR4) is a grade designation assigned to glass-reinforced epoxy laminate sheets, tubes, rods and printed circuit boards (PCBs). *https://en.wikipedia.org/wiki/FR-4*

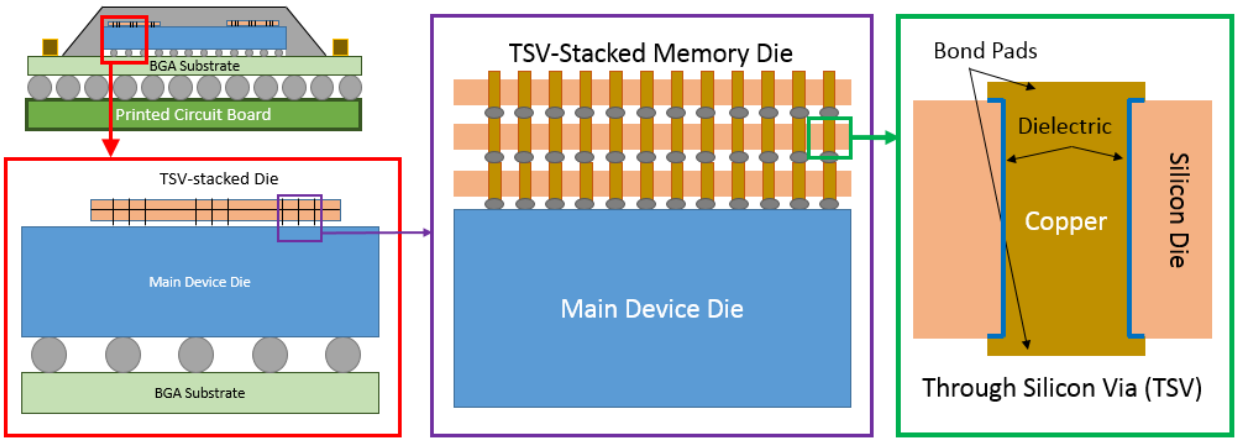

Figure 5: Stacked Die Illustration Using Through Silicon Vias (TSV)

Since 2014, GPU manufacturers have been moving the integrated dynamic random access memory (DRAM) from the circuit card of their GPU modules (aka graphics cards) to the silicon of the GPU itself. The closer the memory is to the GPU core, the quicker the operations are and lower the latency associated with memory transactions is.

More information on 2.5D and 3D packaging can be found in the Book of Knowledge for NASA Electronic Packaging Roadmap†.

2.5 High Bandwidth Memory (HBM)

When DRAM is located on the same substrate stack up as a GPU logic die, it is referred to as High Bandwidth Memory (HBM). AMD Vega's 10nm and 7nm devices are expected to have up to 32GB[14] of HBM. Nvidia's 16nm and 14nm devices have up to 16GB of HBM in their P100 and DGX-1 system. In either configuration, there are four individual stacks of TSV-connected DRAM adjacent to the GPU die.

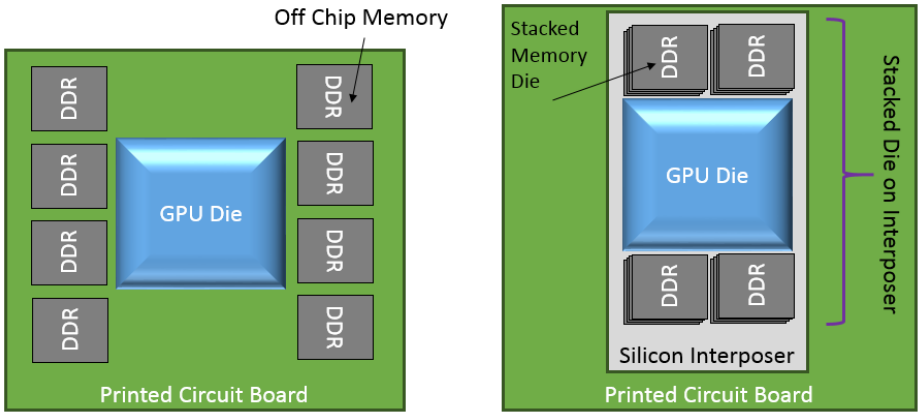

Figure 6: DDR On-Board Memory versus High Bandwidth near-Die Memory

† https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20160001770.pdf

3 Radiation Effects

The intent of radiation testing is to determine inherent radiation tolerance and sensitivities of a specific technology, microcode architecture, or device. Testing leading edge devices makes it possible to identify challenges for future radiation hardening efforts. It also allows us to investigate new failure modes and effects. It is often said in the reliability world that GPUs and other SoC/SiP devices are highly susceptible to radiation effects. Many types of radiation may affect GPGPU logic in operation due to Single Event Transients (SET)[15], corrupting the logic outcome and producing an erroneous result, and eventually being placed into the memory, thus very similar to a Single Event Upset (SEU). The radiation fault could be masked or even propagate to the output generating single or multiple silent data corruptions. The difference between both events is the former affects logic circuits and the latter memory elements. In either situation, the device can have functional interruptions or even hang.

3.1 Soft Errors

In classic CMOS technology, soft errors can cause one of multiple bits to spontaneously flip to the opposite state, due to multiple reasons such as alpha particles from package decay or cosmic rays. Historically, the errors which occur inside the memory chips themselves are almost always a result of radioactive decay. The culprit is the epoxy of the plastic chip package, which like most materials contains a few radioactive atoms. One of these minutely radioactive atoms will spontaneously decay and produce an alpha particle. Practically every material will contain a few radioactive atoms, not enough to make the material radioactive (the material is well below background levels), but they are there.

By definition, a radioactive particle will spontaneously decay. An alpha particle consists of a helium nucleus, two protons and two neutrons, having a small positive charge and a lot of kinetic energy. If such a charged particle "hits" a memory cell in the chip, the charge and the energy of the particle will typically cause the cell to change state. Whether a given memory cell will suffer this type of soft error is unpredictable, just as predicting if or when a given radioactive atom will decay is impossible. However, when you deal with enough atoms this unpredictability becomes a probability, and chip designers can predict that one of the memory cells in a chip will suffer such an error.

The issue with DRAM on GPU die is that they occupy such a large portion of the die and that they are optimized for size to hold the minimum charge necessary to keep a bit until the next refresh cycle. This makes them particularly vulnerable to various forms of radiation, such as alpha particle and neutron strikes. When the strike occurs, it can create a momentary current that flips the state of the RAM cell -- it's a function of the number of RAM cells and the resilience of each cell. The bigger the capacitor in the cell, the harder it is to flip the value. The more cells that exist, the higher the chance that a strike will affect a given chip. Further, the strike may also affect control circuitry, bit lines, etc.

3.2 Substrate Choice

Los Alamos National Laboratory has been using accelerated radiation tests at the Los Alamos Neutron Science Center (LANSCE) to characterize how processing elements respond to radiation[16]. They performed numerous tests on 90nm and 65nm Silicon and Silicon on Insulator (SOI) processors. The continuous technology scaling in integrated circuits makes them more sensitive to the effects of natural radiation such as Single Event Effects (SEEs)[17]. For this reason, physical designers are continuously searching for new methods to improve manufacturing technologies to

reduce SEE consequences. For instance, SOI technology has been proved to be less sensitive than CMOS bulk technology[18]. An evaluation of 45nm SOI multi-core processors has been conducted by university researchers from France[19]. They compare their work to 45nm bulk silicon multi-core processors[20]. In these tests, a 3-5X improvement to SEE immunity is seen when switching from bulk silicon to silicon on insulator for the same technology node.

In the Los Alamos testing, silent data corruption did occur, but was outweighed by microprocessor crashes or Single Event Functional Interrupts (SEFI). While static testing is often the basis for error rate calculations, it can be difficult to translate these errors into dynamic error rates for real-world systems. Determining the overall effect of radiation on microprocessors is not simple, as faults in the systems can remain dormant for several thousands of clock cycles before triggering an error. In addition, the operating system and the software can create noise in the system, making it difficult to determine the cause of system crashes.

3.3 Feature Scaling

Ongoing voltage scaling accompanied with feature size scaling reduces the critical charge required to flip a bit. This allows even lower-energy particles to cause soft errors. Due to these reasons, soft-error rate at 16nm is expected to be more than 100 times that at 180nm[21]. In safety-critical applications soft-data corruption may severely impact the mission. Like FPGAs, software-based hardening[22] (i.e. scrubbing and triple module redundancy) may be the only solution to mitigate radiation effects. This has been seen with the COTS Tesla C2050 GPUs designed by NVIDIA and manufactured in a 40nm technology node and will need to be confirmed as still an issue with TSMS's FinFET process (<20nm), which is currently being employed on NVIDIA's Pascal products.

3.4 Device Organization

In a GPU, a scheduler and a dispatcher are in charge of assigning a code thread to a compute unit, and synchronizing computation. When exposed to radiation, the compute units are isolated, such that a single radiation-induced event in one computing unit will only corrupt the thread assigned to it. Threads that follow the corrupted one or threads assigned to the compute units near the struck one will not be affected. Nevertheless, the corruption of shared resources, like caches, or more critical resources like the scheduler, may affect the execution of several threads, generating multiple output error[23,24].

3.5 Stacked Die

Zhang et al.[25] characterized soft error vulnerabilities across the stacked layers under 3D integration technology. They show that the outer dies can shield more than 90% particle strikes for the inner dies, which leads to a heterogeneous error rate across layers in a 3D chip. 3D integration also provides opportunities of heterogeneous integration, since different layers can be designed using different technologies[26]. Using this feature, the outer layer can be designed using Silicon-On-Insulator (SOI) technology which is more resilient to particle strikes. This helps in achieving the reliability target with significantly lower costs.

Furthermore, with the new stacked DRAM die on both Nvidia and AMD's latest 14nm offerings, it may be necessary to confirm the theory that DDR sensitivity has been seen to decrease with the shrinking of technology nodes[27]. The thought here is that although the active region is smaller, there is more active space behind it which can be susceptible to scattering effects. Total Ionizing Dose (TID) should be evaluated as well to determine the GPU's susceptibility to pipeline

corruption. The latter can be exercised by operating the same code in all pipelines, concurrently as applicable using core affinity, while comparing to a non-radiated device or a reference file using the same software.

4 Testing Considerations

There are many considerations that must be taken into account before a GPU test bench is employed. The test environment (i.e. ionizing radiation) is the primary constraint when testing and how it may affect not only the GPU device, but also other components on the circuit card is critical to building a good test setup. Other things to consider within the environment are the operating system daemons (system supervisors), physical location of software payloads (inputs) and results (outputs), integrity of data paths, and control of electrical and thermal loads. At the Device Under Test (DUT), the primary things to consider are whether the die is accessible, how much of it is accessible in 3D space (e.g. top die, bottom die, perimeter, bond wires), which functions of the DUT are independent or interdependent and whether proprietary software protocols are necessary for its operation.

4.1 Testing: Qualitative versus Quantitative

GPUs are increasingly used in mobile computing and in some embedded system applications. In the latter case safety is sometimes a key issue (e.g., in the automotive, avionics, space and biomedical domains). As an example, the Advanced Driver Assistance Systems (ADAS), which are increasingly common in cars, make an extensive usage of the images (or radar signals) coming from external cameras and sensors to detect the occurrence of dangerous situations, and trigger the activation of proper countermeasures. The high computation power required to process this data perfectly matches the characteristics of GPGPUs, and several vendors already introduced or announced specific products in this area[28].

The constraints of the mission and its test, measurement and instrumentation sensing goals need to be considered when creating the design of experiment for these components. In the quantitative sense, mathematical test vectors can be streamlined by using off-line reference files as comparison. This makes even the most minor instruction or memory upset cause havoc on the results thus making upset identification quicker. Qualitatively, however, it is much more difficult to ascertain the source of an upset or identify one taking place. A 'correct' result may contain a large degree of uncertainty, but still be the expected result. An example of qualitative testing is using a neural network on the GPU (e.g. CAFFE[29], AlexNet[30], DarkNet[31]) to identify objects within images. This type of testing can be done in a comparative way using an offline reference to determine the accuracy of the algorithm to identify the same image during irradiation. Previous work conducted by Dr. Paolo Rech from the Institute of Informatics of the Federal University of Rio Grande do Sul (UFRGS) showed that some errors can be detected with both methodologies[32].

4.2 Device Type and Software Constraints

The device software is dependent on the type of CPU and the application platform. Some combinations of operating system, hardware drivers, and application code work while others do not. The big constraints are:

1. Does it need its own operating system? Some examples are Windows, Linux, Android and Real-time operating system (RTOS)

2. Can we treat the GPU as a peripheral device and send code to it? Some GPUs accept assembly language commands, parallel thread execution (PTX), and C.
3. Can we run the same code on the previous generation and next generation of the device? It is worth noting that each release of a software development kit (SDK) from a vendor adds new features and deprecates others. The deprecation may cause algorithms to crash or behave unexpectedly.

When developing a test bench for multiple platforms, it is easiest to compile code locally on that platform to make sure that the source code libraries are compatible instead of cross compiling from one system for another. Here are some examples for developing software for cross platform use:

- Nvidia Tegra X1 – SoC ARM with embedded Linux (Nvidia's Linux for Tegra)
- Nvidia GPUs – GPU for x86 Windows and x86 Linux
- Intel Skylake Processor – IP Block for x86 Linux
- Qualcomm Adreno & Mali GPU – IP Block for ARM Linux or Android

Once these constraints are addressed, payload algorithms must be devised to both bypass the optimization features in the compiler and at the hardware level with the scheduler. Optimizations allow the GPU to predict data coming through its pipelines, apply caching features to some or all of it, and unroll certain syntax such as loops so that the GPU can more quickly process the data. These optimizations do not necessarily guarantee that the GPU is performing the algorithm as it was designed. Further, these optimizations must be bypassed to also test independent features of the GPU such as the L1 cache, shared memory, instruction pipelines, scheduler, etc. However, some pragma[‡] and other code permit toggling of the cache.

Nvidia's SDK for CUDA comes with some visual and computational algorithm samples. While this code base may change between versions or even expand, it does provide some samples that are the same across multiple versions. This allows at least a comparison point to investigate whether a new Nvidia DUT behaves similar to others within its microcode architecture, and whether new optimizations have been included and permits electrical and thermal characterization of the DUT under consistent circumstances.

Other payload types are sensor streams, computational loading models, or simply some mathematics. Some sensor stream examples include a large data set, video camera input feed or pictures within a database. The stream allows a direct input into the GPU from a peripheral device which limits memory access transactions up the memory hierarchy. Computational loading models on the other hand create a very large load on the GPU's internal memory handler and its scheduler because of the amount of data that must be processed. Models of this type tend to work most constituents of the GPU and can be used to explore issues within its state space. Mathematic models are the easiest to apply, however they can provide the least amount of beneficial data. The optimizations tend to cache and "guess ahead", which cause most of the GPU to become idle.

GPU testing requires a complex platform to arbitrate the test vectors, monitor the DUT (in multiple ways) and record data. None of these should require the DUT itself to reliably perform a task

[‡] Pragma in general terms is a programming language construct that specifies how a compiler (or other code translator) should process its input. https://en.wikipedia.org/wiki/Directive_(programming)

outside of being exercised, because if the DUT experiences a fault, any subsequent instructions will likely fail and any remnant data should be considered garbage.

4.3 Test Environment

As previously discussed, the test environment is the primary constraint when designing a test bench. If the environment has a thermal exposure range outside of the nominal operating temperatures of the device then some precautions must be made to limit thermal damage to components – by either mitigating exposure to neighboring components or possibly by only exposing the DUT to the thermal load. In a radiative environment, only exposing the DUT to the beam is critical as the other components will likely behave abnormally or fail. Shielding of electronics that control the test setup is very necessary if they are close to the beam line. In a Total Ionizing Dose (TID) test, having a separate voltage-bias circuit board to power the DUT while in the radiation chamber is often necessary to protect control electronics. A circuit board to exercise the DUT is maintained outside of the chamber. While the following subsections discuss setting up a radiation test, most of the information is transferrable to a reliability test.

4.4 Test Setup and Organization

In most beam lines, the operator area is at a different location than where the DUT is setup. In the operator area, things take place at a distance where as they normally would be on bench close by. In a proton facility, the operator area tends to be 100+ feet away which requires either patch panel for cables or very long cable runs. Signal integrity must be considered for high speed connections greater than 1 meter in length. Additionally, any interaction with the test setup must be done from the same distance. This means that any button press or switch toggle must be done with wires, relays or other mechanisms that can be controlled from a distance. The goal here is repeatability.

Electrical control is also important. While it is easy to take a COTS computer and power it on, there is very little insight into how its power supply is being loaded. Lab bench supplies such as those from Agilent or Keithley are almost always able to be controlled remotely. Some of these require significant upfront resources such as the development of LabVIEW programs or scripts to control the power supply. Other devices on the market permit the toggling of power to individual outlets on a power distribution unit or surge protector. Further, many of these options come with electrical monitoring capabilities such that the voltage and current loads of the DUT can be monitored.

The following figure provides an overview of a typical GPU test setup. The GPU is depicted by the bluish block at the top of the image.

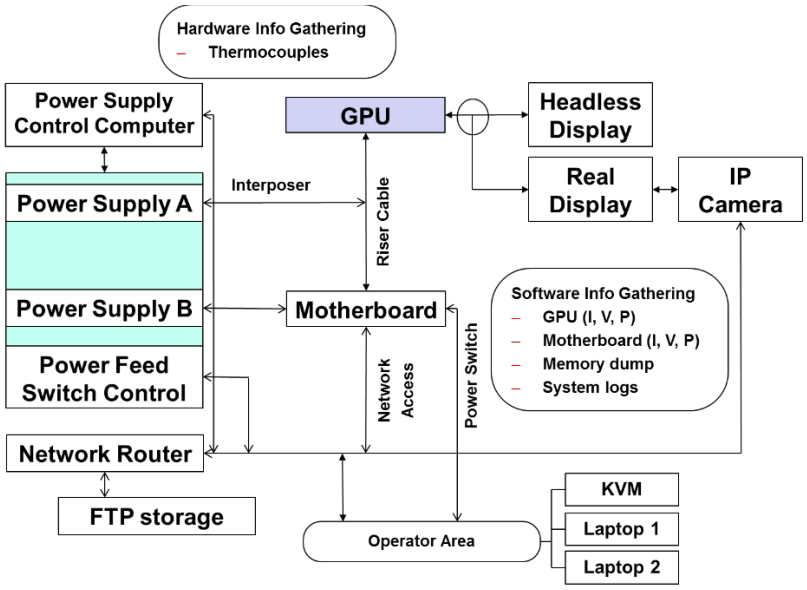

Figure 7 Typical GPU Test Setup

In this type of test setup, it is a goal to apply a monitoring capability to all sources, all interconnects and places where a 'just in case' set of eyes would be helpful. The following list of accessible nodes provides some high level guidelines into where to monitor a test.

- Network
  - Heart beat by inbound *ping* from operator area
  - Heart beat by timestamp upload via file from DUT
- Peripherals response
  - "Num lock" toggle on keyboard
  - Blinking cursor at terminal client prompt
- Visual check
  - Remotely view screen via remote desktop application (i.e. VNC)
  - Local view of screen via long cable run
  - Remote viewing of local screen using IP camera
- Electrical states
  - At the system via power supply control
  - At the DUT via interposer and monitoring circuit
- Temperature
  - At the DUT (micro environment)
  - Around the DUT (macro environment)
  - Ambient

Ping[33] is a computer network administration software utility used to test the reachability of a host on an Internet Protocol (IP) network. It measures the round-trip time for messages sent from the originating host to a destination computer that are echoed back to the source.

4.5 Test Operation Flow

Testing steps often consist of a baseline simulation or payload configuration while the DUT is not exposed to the beam. The aforementioned monitoring nodes permit the operator to capture an error occurring, subsequently stop the beam and perform an in-situ analysis of the results where

some memory dumps may be saved to an offline storage location for later post processing. The following flow chart depicts a typical test flow with system upsets and corrective action. The test continues if the DUT can be restored to its pre-irradiation functionality. This does not mean, however, that its electrical and thermal response need to be the same as pre-irradiation. Some deviation is expected from the dose rate.

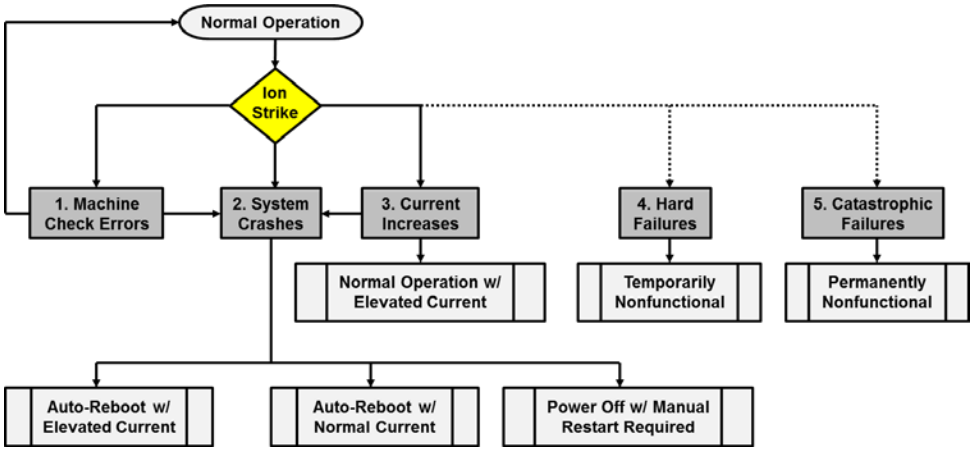

Figure 8 Test Flow Chart

4.6 Other Test Considerations

Reliability, Quality and Performance Issues

Semiconductor scaling and improvements to manufacturing has caused multiple trade-offs to occur with respect to Speed, Weight and Power (SWaP) and reliability. Details regarding challenges that occur during manufacturing, semiconductor design and component packaging are found in the appendix. These parameters are valuable when comparing different generations of devices.

Component and Performance Failures

Interactions with various component manufacturers indicated that GPU and CPU failures don't tend to take place during the useful life of a product. This lifetime tends to be eight hours per day, five days per week with a work load for 50 weeks of the year for four years. Failures were categorized into four groups: memory, controller (CPU, GPU, or microcontroller), passive component and PCB interconnects. It is worth noting that failure of any of these 'system components' will fail the GPU. Components that comprise a GPU other than the silicon die and their failure modes are discussed in the appendix.

Error Identification

Resilience is defined as the ability of a system to keep applications running and maintaining an acceptable level of service in the face of transient, intermittent, and permanent faults. The term "fault tolerance" refers to the ability of a system to continue performing its intended function properly in the face of faults. An error is the part of the total state that might lead to a failure and the failure is a transition to incorrect service. Faults can be active or inactive, depending on whether they cause errors or not.

Error identification is a process of discovering the presence of an error but without necessarily identifying which part of the system state is incorrect, and what fault caused this error. By definition, every fault causes an error. Almost always, the fault is detected by detecting the error

the fault causes. Therefore, fault detection or error detection often refers to the same thing. Latent or silent errors are errors that are not detected.

Errors that are caused by a persistent physical defect are traditionally referred to as "hard" errors. A hard error may be caused by an assembly defect such as a solder bridge or cracked solder joint, or may be the result of a defect in the silicon itself. Rewriting the memory location and retrying the read access will not eliminate a hard error. This error will continue to repeat.

Errors caused by a brief electrical disturbance are referred to as "soft" errors. Soft errors are transient and do not continue to repeat. If the soft error was the result of a disturbance during the read operation, then simply retrying the read may yield correct data. If the soft error was caused by a disturbance that upset the contents of the memory array, then rewriting the memory location will correct the error. Some soft errors require a power cycle to reset the device. Hard errors are typically detected by memory tests run at boot time.

Log Files

Many CPU and GPGPU systems, errors are logged in a register bank. Each register bank can log one error at a time; x86 architecture dictates that the hardware discard subsequent corrected errors until the bank is read and cleared by the operating system. The operating system typically reads the register bank via a polling routine executed once every few seconds. Therefore, on a processor which issues multiple memory accesses per nanosecond, millions of errors may be discarded between consecutive reads of the register bank. In GPU systems, a hardware reset tends to clear the counter. Some errors of this nature can be identified and logged in a memory dump file produced by the operating system.

The error logging architecture described above means that the console logs represent only a sample of all corrected errors that have occurred on a system. Moreover, the number of logged corrected errors is highly dependent on the polling frequency set by the operating system. For instance, if the operating system is using a 5-second polling frequency, only one error per node can be reported per 5-second interval.

In addition to automatically generated logs, it is a prudent method to have the test bench generate as many types of logs as possible for post-processing after testing. Further, all clock domains on the DUT and in the test bench should be synchronized prior to test start.

4.7 Original Component Manufacturer Qualification Testing

Initial defects[34] from manufacturing qualification are the harbinger of long term reliability. The depth of Original Component Manufacturer (OCM) testing determines how much of their production yield will be affected by latent manufacturing issues. If not enough testing was performed or not the correct testing, then the Original Electronic Manufacturer (OEM) must determine their risk through their own tests.

In an example, we consider a GPU design which was manufactured using a leading edge process. The foundry qualifies a process technology using basic shapes and structures with fundamental semiconductor properties. These test structures are what qualifies the process. Their customer provide design files of their device for a given process. The foundry has proprietary reliability models which determine the anticipated lifetime of their customer's product. The lifetime prediction is based on correlation of relative shapes of their test structures.

The best example is conductor width and Black's equation, which dictates the onset of electromigration. The lifetime assumes that no manufacturing defects or marginal design features (i.e. 90° angle where a 45° angle should be) exist in the product they are producing. When manufacturing defects occur such as dust or debris capture, these devices are screened out of the population during a wafer sort process. None of these devices are considered viable and as such are not part of the delivery yield back to their customer.

The logic core device's wafer is subjected to a Wafer-Level Burn-in and Test (WLBT)[35] process which exposes the device's power pins and some device structures such as large transistors, flip-flops, registers and multiplexers to voltage-current combinations to bin its performance corners. If the device corners fall within the envelope of the desired yield performance corners, then the device passes. Otherwise, it is either disposed of or down-selected into a lower performance bin.

A temperature Burn-In (BI) process is then conducted with an elevated temperature; this is usually around 125°C for up to 48 hours. The devices are retested afterwards at room temperature to see how far their performance deviated from the initial values. This is typically used for electromigration checks as it causes annealing of the conductors. The conductor impedance is measured beforehand and afterwards. The foundry will know what the acceptable level of deviation is based on their initial design correlation process.

For 2.D and 3D packaging, if another device type die (i.e. DRAM) is required in the final device package, then it is subjected to these processes too; unless they were tested at their own foundry. A packaging and assembler will then use other processes to stack the die. At this point, a physical device is realized and needs to be checked for functionality.

Functional tests[36] have a variety of names, such as operational-life tests, parametric or vector testing, and fault injection. Commercial fault injector equipment is not appropriate for these devices as the design description codes are held with high secrecy. Functional testing is seldom performed at temperature extremes. Instead, it is performed on a lab bench with a temperature management system which maintains its temperature to 25°C during the testing. This type of test does not necessarily stress the device as it would be stressed in the end-user application. Vector testing may be performed with patterns to test the memory hierarchy. For GPUs, an embedded code package is deployed to test basic functionality at one or multiple voltage-clock combinations. This is called a Software Based Self-Test (SBST). The final performance metric which results from this multi-stage testing methodology assigns each component a performance bin. The devices then go to the board-level OEM for assembly.

5 Currently Available Technology

This section discusses what types of devices are currently available for different market segments. The application environment or purpose of the device would dictate which type to procure.

- Native GPU (i.e. Nvidia GeForce)
- Application Processor (i.e. TI OMAP)
- Accelerator Processing Unit (i.e. AMD's Fusion APU)
- ASIC/FPGA/PLD (i.e. Xilinx Zynq UltraScale+ MPSoC)

Historically speaking, code had to be specifically written to send frame buffer commands at separate hardware. With advancements in unified languages and hardware translation, it has become possible to run native code directly on the GPU such as C. Unlike discrete or native GPU coprocessors, some GPUs take the form of an IP block or embedded engine within a System on Chip (SoC) device. The best example of this is a smart phone's SoC such as the Qualcomm Snapdragon 820 which contains a Qualcomm Adreno 530 GPU. Within this device are various functional blocks which can be exercised with software payloads. Nvidia's Jetson TX1 SoC is provided on a modular printed circuit board connected to a main carrier board by a connector. Within it are ARM CPU cores and an nVidia GPU which can be accessed similarly to the discrete GPU coprocessor. The point here is that while the packaging is different, each one of these GPUs can be tested using the same standardized code.

The Khronos group manages the specification for the OpenCL compute language while Nvidia manages its own proprietary CUDA language. The majority of devices support OpenCL code libraries. It is worth noting that Intel does not directly support OpenCL with a development kit. Instead, a software community discussion board, freedesktop.org, has produced an open source implementation of the OpenCL specification called Beignet[§]. Beignet is a generic compute oriented API for Linux which permits OpenCL programs to be run on Intel's GPUs. More information can be found at the freedesktop.org website. Other variations on how this is implemented can be found through the github.com website. A comprehensive list of OpenCL conformant products can be found at the Khronos.org website[**].

Product vendors who support OpenCL implementations include Intel, Nvidia, Apple, Arm, Qualcomm, Marvell, Vivante, MediaTek, AMD, Texas Instruments, Altera, Xilinx, PowerVR, STMicroelectronics-Ericsson, IBM and ZiiLABS.

Three types of payloads can be implemented using open standards: Neural Network, Math-Logic and Colors. The neural network is a convolutional neural network (CNN) type, which can avoid processor optimizations that recursive neural networks (RNN) primarily benefit from. Math-Logic uses mathematics and conditional logic statements to exercise memory hierarchy. The Colors payload assesses corruption in the output image presented to a monitor or display.

- Convolutional neural network (CNN) to identify land usage objects using a dataset modified from [37] for use with a "You Only Look Once" (YOLO) algorithm for object identification in still images and live stream video. The CNN was configured to be trainable on both GPU and CPU microprocessor types. Twenty one image categories were identified across the dataset. Figure 9 shows three such categories. The YOLO algorithm provides an accuracy rating and the most likely image category when presented with an image. The categories are: agricultural, airplane, baseball diamond, beach, buildings, chaparral (vegetative desert), dense residential, forest, freeway, golf course, harbor, intersection, medium residential, mobile homes, overpass, parking lot, river, runway, sparse residential, storage tanks and tennis court.
- Mathematical and logic payloads such as calculating Pi, polynomial arithmetic, Markov permutations (such as folding protein algorithms) and algebra are leveraged to fill the computational and memory components of the device while preventing hardware optimizations to manipulate the software bit-stream. These math payloads target the layers of the memory hierarchy of the device.

[§] https://www.freedesktop.org/wiki/Software/Beignet/
[**] https://www.khronos.org/conformance/adopters/conformant-products#opencl

- Graphics, texture and color rendering tests have been developed. Graphics memory tends to be directional in that it behaves as read-only. The simplest test allows monitoring of this memory by triggering a pixel color change with automatic screen compare for pixel-change identification. The most complex of these tests performs a burn-in to the rendering logic of the device. Pixel corruption or display artifacts are monitored and recorded using the test bench.

Baseball Diamond	Intersection	Dense Residential

Figure 9: Neural Net Training Images from the Land Use Dataset

In CNN networks, there are multiple knobs that can fine tune the operation of the network permitting the payload's efficacy in regards to exercising specific device structures to be scalable to the DUT which permits repeatable testing. This also allows comparable and defensible tests to take place across part numbers.

Neural networks are one type of payload that can be scaled for hardware that it supports. Unfortunately, there is not yet one neural network platform that is fully cross platform for hardware (e.g. Intel, ARM, AMD, nVidia). Therefore, a basic Math-Logic or Colors payload can test multiple generations of a device that doesn't support neural networking. OpenCL and OpenGL are open source computational and graphics libraries, respectively. Both of these standards are supported across most vendors' hardware (discrete components and embedded IP in system on chip) and are supported on all modern forms of Windows and Linux operating systems. The payloads using math or colors are also tuned to be scalable and efficient just like the configurations of neural networks. Payloads that have been designed for NASA's Electronic Part and Packaging (NEPP) Program have been compiled to be able to run within a Windows 2016, Ubuntu Linux, Linux for Tegra, and Android OS environments.

6 Summary

Semiconductor reliability is already a challenge for terrestrial applications in the realms of high performance computing (HPC), aerospace, defense and automotive electronics. Typically, these devices can be evaluated for performance marginality and long-term risks associated with semiconductor materials degradation (from nominal usage) using simulation, 1st and 2nd order approximations, or simply brute-force testing. While these methodologies have historically worked for monolithic devices, it is increasingly more difficult to assess modern CPUs, GPUs and even FPGAs. The linchpin is that not enough calendar time is available to test all possible state configurations of a device within reasonable expenditure budgets. What further exacerbates this situation is the limitations in "testing to the application environment." These challenges can be overcome, to a high degree, by a logical system-level approach to device testing using COTS computer and network components, hardware SDKs, semi-automated electrical test and

measurement equipment, and multiple software modules connecting or arbitrating the test platform.

Graphics Processing Units have emerged as a proven technology that enables high performance computing and parallel processing in a small form factor. GPUs enhance the traditional computer paradigm by permitting acceleration of complex mathematics and providing the capability to perform weighted calculations such as those in artificial intelligence systems. Despite the performance enhancements provided by this type of microprocessor, there exist tradeoffs in regards to reliability and radiation susceptibility which may impact mission success. This report provided an insight into GPU architecture and its potential applications. It also discusses reliability, qualification and radiation considerations for testing GPUs.

Testing guidelines for devices similar to GPUs or processor devices that share features with GPUs, in space or other similar electronics markets, are available from NASA Jet Propulsion Laboratory, NASA Goddard Space Flight Center and Los Alamos National Laboratories. A few notable documents are provided here for reference.

F. Irom, "Guideline for ground radiation testing of microprocessors in the space radiation environment," Jet Propulsion Laboratory, Tech. Rep. 13, 2008.

S. M. Guertin, "A guideline for SEE testing of SOC," in Single Event Effects Symposium, San Diego, CA, May 2013.

M. Berg, "Field programmable gate array (FPGA) single event effect (SEE) radiation testing," NASA/Goddard Space Flight Center, Tech. Rep., 2012.

H. Quinn, "Challenges in testing complex systems," IEEE Trans. Nucl. Sci., Vol. 61, No. 2, pp. 766-786, 2014.

7 Appendix

7.1 Electrical Connections in the Overall GPU System

Electrical or copper based links are subject to errors. However, these errors tend to be from extrinsic sources. Components used to make the electrical link are still sources of noise in a system. These errors are caused by things such as spikes, sags and surges in the power mains, electrostatic discharges, RF emissions, and cable/connector vibrations.

Work at Facebook[38] and Carnegie Mellon University had shown that errors follow a power-law distribution and a large number of errors are due to CPU sockets and channels. Non-DRAM memory failures from the memory controller and memory channel cause the majority of errors, and the hardware and software overheads to handle such errors cause a kind of denial of service attack in some servers.

Furthermore, chips per DIMM, transfer width and workload type (not necessarily CPU/memory utilization) affect reliability. Additionally, plotting against a Weibull distribution shows a decreasing hazard rate which indicates an infant mortality behavior. This may indicate that a semiconductor or PCB assembly process is the root cause of the faults. These plots are shown in Figure 10 and Figure 11.

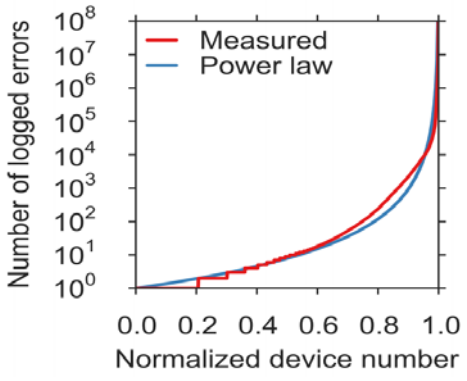

Figure 10: Memory Error Distribution – Power-Law (Facebook)

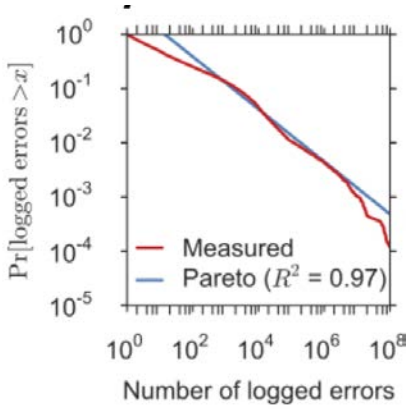

Figure 11: Memory Error Distribution – Hazard Rate (Facebook)

Another conductor topic is Through Silicon Vias (TSVs) in stacked-chip DRAM devices. A crack in one of the solder bumps connecting the stacked die, would cause an impedance change or complete isolation of that data path from the array – thus causing memory errors.

7.2 Voltage Scaling Relative to Lithography

There is a notion from the commercial market segment, that in today's systems the frequency of bit flips is no longer dominated by single-event upsets caused by radiation from space but it is increasingly attributed to fabrication miniaturization and aging of silicon. This is supported by the increasing likelihood of repeated failures in DRAM after a first failure has been observed[39]. Per Dr. Guertin at NASA's Jet Propulsion Lab, this is true for Galactic Cosmic Ray (GCR)-caused errors, at ground-level, the failure rate is not dominant (or at least should not be). But at aviation altitudes the rate goes up 100x or so, and in space the terrestrial failure rates from other sources is essentially negligible. A typically acceptable failure rate is maybe 30-100 failures per billion operating hours (unit: FIT, 1 failure per billion operating hours) and may be dominated by

problems with fabrication. At sea level, radiation may be ~20 FITs, but space failure rates, from radiation, can get up to 1e8-1e9 FITs.

Reduction in supply voltage to very close to the transistor's threshold voltage reduces power consumption, but decreases reliability of SRAM structures such as cache. Large on-chip SRAM structures such as the L2 and L3 caches are the most vulnerable to errors. In near-threshold, a cache can experience error rates that exceed 4%, rendering an unprotected (no ECC) structure virtually useless.

Memory voltages have decreased over the years, declining from 2.5V to 1.8V, 1.5V, 1.35V, and then to 1.2V as the industry has shifted to faster speed protocols. As the operating voltages decrease, the available noise margin also decreases, making it more difficult for receivers and sense amplifiers to distinguish between a 1 and a 0.

7.3 DDR Capacity in GPUs

A primary reason for increased error rates is the rapid increase in the size of memory systems. As more bits of memory are added to the system, the likelihood that any device will encounter a bit error increases. Such increases in system memory capacity are the result of the shrinking size of DRAM modules (that is, more bits can be packed on a single die and subsequently more die per package). Since 2008, DRAM capacities have increased 16x, from 512 megabits to 8 gigabits.

Further work from the Facebook-Carnegie Mellon study showed that DRAM reliability is reducing in the same rates as the increase in density. They also provide evidence that more recent DRAM cell fabrication technologies (as indicated by chip density) have substantially higher failure rates, increasing by 1.8X over the previous generation.

7.4 Rowhammer[40]

A fundamental assumption in software development is that a memory location can only be modified by processes that are permitted to write to this memory location. However, multiple studies over the last five (5) years have indicated that there are parasitic effects in DDR that can change the content of a memory cell without accessing it. This is accomplished by accessing other memory locations in a high frequency. This is called the Rowhammer bug which affects most DDR3 and DDR4 systems. Manufacturers have been quite successful in containing such effects until recently. Clearly, the fact that such failure mechanisms have become difficult to contain and that they have already slipped into the field shows that failure/error management in DRAM has become a significant problem. This problem will become even more exacerbated as DRAM technology scales down to smaller node sizes (<20nm). This is especially a concern as graphics cards contain multiple gigabytes of DDR.

While this problem seems random like cosmic rays, it is rather a bug in the way memory is accessed in an interleaved device. Address space layout randomization was a technique design to stop hackers until they started using memory spray to defeat it. Memory spray is when a large amount of memory is allocated, and then code is set up so that no matter which bits gets flipped, the resulting value will still point to something valid. The underlying issue is an analog one. It's not a bug in the code, or in the digital design of the chips, but a problem at a deeper layer.

All studies related to Rowhammer rely on the availability of a cache flush instruction in order to cause accesses to DRAM modules at a sufficiently high frequency. Memory caches can be forced

into fast cache eviction to trigger the Rowhammer bug with only regular memory accesses. This allows the Rowhammer bug to affect highly restricted and even scripting environments. Very simply the problem occurs when the memory controller under command of the software causes an ACTIVATE command to a single row address repetitively. If the physically adjacent rows have not been ACTIVATED or refreshed recently the charge from the over-ACTIVATED row leaks into the dormant adjacent rows and causes a bit to flip. Bit flips can be triggered automatically by software by flushing a memory location from the cache and reloading it. This fault mode results in corruption of a "victim row" when an "aggressor row" is opened repeatedly in a relatively short time window. Once this failure occurs a Refresh command from the memory controller solidifies the error into the memory cell. Current understanding is that the charge leakage does not permanently damage the physical memory cell which makes repeated memory tests trying to find the failing device useless.

Techniques have also been shown to flip bits without accessing the adjacent rows. Repeated reads can be sufficient to cause a disturbance. In some instances, research has shown that 540,000 read instances can be achieved in 64 milliseconds[41]. Once such study showed that frequently accessing specific memory locations can cause random bit flips in DRAM chips. 85%[42] of the DDR3 modules they examined were vulnerable. Other studies have shown this behavior exists in DDR4[43,44] memories as well, even though Targeted Refresh Row (TRR) schema have been implemented. The selection of channel, rank, bank and row is based on physical address bits.

In 2014, Samsung divulged the issue in an investor's presentation touting that its DDR4 "in-DRAM" solution is "most efficient for Rowhammer operation". One caveat is that no test data showing that DDR4 is immune to this problem has ever been published.

Pseudo Target Row Refresh (pTRR) and Target Row Refresh (TRR) are features that refresh neighboring rows when the number of accesses to one row exceeds a threshold. They have less overhead compared to doubling the refresh rate. Although TRR has been announced as implemented in all DDR4 modules, it has been removed from the final DDR4 standard. Manufacturers can still choose to implement it in their devices, but if the memory controller does not support it, it has no effect. In addition, while the TRR command is not part of the JEDEC DDR4 specification, it is part of the LPDDR4 specification which does not help servers or other high performance systems.

Error-correcting code (ECC) memory is often mentioned as a countermeasure against Rowhammer attacks. However, recent work shows that it cannot reliably protect against Rowhammer attacks. At the software level, one proposed countermeasure is the detection using hardware performance counters. The excessive number of cache references and cache hits allows to detect on-going attacks. However, this countermeasure can suffer from false positives, so it needs further evaluation before it can be brought to practice.

Rowhammer is worth discussing here in that while software security is less of an issue in a flight system than a terrestrial information technology platform, there is a high risk of unintentionally corrupting data within a flight application due to its own software. Thorough functional testing should be performed when utilizing DDR3 and DDR4 in space systems. If data becomes unintentionally corrupted due to the application, modifications can be made within the software to the memory retrieval code to force a different pattern of retrievals or write-backs.

7.5 Non-Semiconductor Components on GPUs

7.5.1 Capacitor (Buried and Chip Types)

Capacitors are found as both passive chip components on the GPU substrate and within the DRAM. In general, the problem is reduction in the thickness of the dielectric material in the capacitor. This makes it challenging to ensure the same capacitance value, given the unreliability of the ultra-thin dielectric material. For high capacitance, low voltage chip capacitors, and any multi-layer chip capacitors smaller than a 0805 package, they are often subjected to early life failures when exposed to higher temperatures. These are often found on GPUs and are subjected to the GPUs die temperature.

Issues with ceramic chip capacitors can cause changes in capacitance, increased leakage current, decreased insulation resistance, electrical short or electrical open. In most cases, these failure mechanisms will cause a catastrophic fail of the component. However, intermittent circuit behavior can also take place.

1.1 Flex cracking
Caused by depaneling, handling, insertion of connectors and mechanical fasteners, attachment of PCB or other structures within the assembly

1.2 Knit line cracking
This is a manufacturing defect

1.3 Thermal cracking
The most typical cause is thermal shock which can take place from a soldering excursion, contact with a soldering iron tip or inappropriate spacing and layout of the circuit card

1.4 Oxygen vacancy migration
Caused by thin electrode tape, a porous dielectric (ceramic) or cracks in the electrode tape

1.5 Over voltage
Caused by thin electrode tape or excessive application voltage above the specifications of the component

1.6 Dissipation factor, dV/dt
Caused by thin electrode tape, non-uniform dielectric spacing, contaminants in the dielectric (ceramic) or contaminants in the termination system of the capacitor package

1.7 Electrostatic Discharge (ESD)
Caused by handling, manufacturing, assembly and charged surfaces.

1.8 Electrode or termination corrosion
In PdAg type components, caused by atmospheric sulfur which attacks and consumes the silver

1.9 Radiation Induced Charge Loss
During radiation the capacitor leakage resistance decreases. Therefore, the time constant of the circuit will also decrease. If the capacitor is in a critical timing circuit, the timing circuit may produce errors that affect system performance[45].

1.10 Radiation Induced Conductance
The amount of radiation-induced conductivity (RIC) can vary widely with dielectric material type. For a dielectric (insulator) the band gap is large relative to room temperature thermal energies, and the valence band is full and the conduction band is empty, leaving no electrons available to participate in the conduction process. However, in the presence of ionizing radiation where the energy from the radiation that can be imparted to an

electron exceeds that of the band gap, an electron may transition to the conduction band and electrical conduction will take place[46]. This can cause a catastrophic short circuit depending on the electrode spacing within the component. Radiation and leakage current do have a relation.[47] This can be attributed to RIC.

7.5.2 Film or Chip Resistor

Issues with film resistors can cause resistance value changes including electrical opens

1.1 Electrode Crack
Caused by solder stresses, resin in the mold or an unsuitable pattern design

1.2 Solder breaking
Caused by an unsuitable soldering condition or thin/non-uniform nickel layer

1.3 Electrode corrosion
When a silver frit epoxy is used to adhere the film to the resistor substrate, the silver can be attacked by atmospheric sulfur which consumes it forming salts

1.4 Solder degradation
Caused by overloading the component package with mechanical stress or excessive thermal stress

1.5 Resistor Chipping and Cracking
Caused by depaneling of resistors, defective shapes or glass chipping

1.6 Voltage breakdown
Caused by exposure to large pulse loads or loss of inner resistor film material by other means

7.5.3 Substrate

Issues with the FR4 or organic substrate for integrated circuits are typically caused by a variation in manufacturing or assembly process

1.1 Delamination of resin to glass or to plastic

1.2 Metallization and conductor anomalies

7.5.4 Flip Chip attachment

Very small solder bumps are used to attach a flip chip die onto the component package substrate. The bumps themselves should be treated in the same way as one would evaluate typical solder balls on a ball-grid array (BGA) substrate. A flip chip orientation is the alternative to wire bonding. Seldom are both employed on the same die. However, both can be used within a system-in-package (SiP) device where multiple die are embedded within a single die-carrying package.

7.5.5 Bond Wires

When bonded wires lift from the bond pad or have a peculiar tail (in the case with stitch bonds where no tail should be present), the most probable source is poor capillary design choice in the wire bonder. Failure to get a good termination can be due to several factors during the bonding process.

Orientation of the bonding tool can be misaligned to the bond pad surface such as too low a face angle on the capillary. A larger face angle will result in a thicker cross section for second bond and a better termination because the cutting angle of the cap tip is sharper.

If the ultrasonic energy is too high while the bond force is too low, then the bond may not adhere to the surface of the bond pad and may also displace most of the bonding material. A higher force combined with lower ultrasonic energy will help this problem. Depending on the bonder, either the constant velocity or the search speed (both are different names for the same motion) effect the impact force. This is different than the bond force. The impact force provides most of the force deforming the wire. More initial deformation prior to ultrasonic exposure has a big effect on bond quality.

Aside from surface contamination, there are other wire bonding site or substrate problems that can lead to bonding failures. Common bonding site/substrate issues include excessive probe digging on bond pads, lifting of the bond pad metal, voids in the plating of the lead fingers, damage beneath the bond pad, which can lead to cratering and semiconductor nodules from metal lines or substrate processing.

Acknowledgements:
This work was performed in support of the NASA Electronic Parts and Packaging (NEPP) Program. Guidance and review was provided by NEPP's co-manager Kenneth LaBel and Steve Guertin from NASA's Jet Propulsion Lab.

REFERENCES

[1] Wikipedia.com
[2] gpgpu.org
[3] http://www.nextplatform.com/2016/06/20/intel-knights-landing-yields-big-bang-buck-jump/
[4] https://media.gm.com/media/us/en/gm/bcportal.html/currentVideoId/4631756055001/currentChannelId/Most%20Recent.gsaOff.html
[5] http://newatlas.com/nvidia-drive-px-2-in-car-supercomputer/41160/
[6] Burgess TFR., Heron SF. 2011. Computing Applications for Satellite Temperature Datasets: A Performance Evaluation of Graphics Processing Units. NOAA Technical Report NESDIS 139. NOAA/NESDIS. Silver Spring, MD. 14pp. https://data.nodc.noaa.gov/coris/library/NOAA/CRCP/other/other_crcp_publications/NOAA.Tech.Report.139.Final_12-29-11.pdf
[7] DJI website. https://store.dji.com/product/manifold
[8] http://www.intelligent-aerospace.com/articles/2016/04/curtiss-wright-debuts-rugged-dual-gpgpu-openvpx-supercomputing-module-for-deployed-radar-sigint-eo-ir.html
[9] https://www.nrl.navy.mil/itd/imda/sites/www.nrl.navy.mil.itd.imda/files/pdfs/AIAA_uav_detection_6.pdf
[10] Wyrwas, Edward. "Graphics Processor Units (GPUs)," 2017 NEPP Electronics Technology Workshop; 26-29 Jun. 2017; Greenbelt, MD; United States. GSFC-E-DAA-TN43820. https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20170006038.pdf
[11] https://en.wikipedia.org/wiki/Parallel_computing
[12] https://en.wikipedia.org/wiki/File:Pipeline,_4_stage.svg Permission is granted to copy, distribute and/or modify the image under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation.
[13] http://www.nvidia.com/object/tesla-p100.html
[14] https://www.techpowerup.com/226012/amd-vega-10-vega-20-and-vega-11-gpus-detailed

[15] Oliveira D.A.G. ; Rech P. ; Pilla L.L. ; Navaux P.O.A. ; Carro L. Gpgpus ecc efficiency and efficacy. IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems , pages 209 – 215, 2014

[16] Improving Microprocessor Reliability Through Software Mitigation Heather Quinn, Justin L. Tripp, Tom Fairbanks, Andrea Manuzzato. Los Alamos National Laboratory Document release number: LA-UR-lO-08334

[17] P.E. Dodd and L.W. Massengill, "Basic mechanisms and modeling of single-event upset in digital microelectronics," IEEE Trans. Nucl. Sci., vol. 50, pp. 583–602, June 2003.

[18] G. Gasiot, V. Ferlet-Cavrois, J. Baggio, P. Roche, P. Flatresse, A. Guyot, P. Morel, O. Bersillon, and J. du Port de Pontcharra, "SEU Sensitivity of Bulk and SOI Technologies to 14-MeV Neutrons," IEEE Trans. Nucl. Sci., vol. 49, pp. 3032–3037, Dec. 2002.

[19] Pabo Ramos, Vanessa Vargas, M. Baylac, F. Villa, S. Rey, et al.. Evaluating the SEE sensitivity of a 45nm SOI Multi-core Processor due to 14 MeV Neutrons. IEEE Transactions on Nuclear Science, Institute of Electrical and Electronics Engineers, 2016, 63 (4), pp.2193 - 2200. <10.1109/TNS.2016.2537643>. <hal-01280648>

[20] S.S. Stolt and E. Normand, "A Multicore Server SEE Cross Section Model," IEEE Trans. Nucl. Sci., vol. 59, pp. 2803–2810, Dec. 2012.

[21] S. Mittal et al., "A Survey of Techniques for Modeling and Improving Reliability of Computing Systems," IEEE TPDS, 2015.

[22] L. Pilla; P. Rech; F. Silvestri; C. Frost; P. O. A. Navaus; M. Sonza Reorda; L. Carro. Software-based hardening strategies for neutron sensitive fft algorithms on gpus. IEEE Transactions on Nuclear Science, pages 1–7, 2014.

[23] P. Rech, C. Aguiar, C. Frost, and L. Carro, "An Efficient and Experimentally Tuned Software-Based Hardening Strategy for Matrix Multiplication on GPUs", IEEE Trans. Nucl. Sci, 2013

[24] P. Rech, L. Pilla, F. Silvestri, C. Frost, M. Sonza Reorda, P. Navaux, and L. Carro, "Neutron Sensitivity and Hardening Strategies for Fast Fourier Transform on GPUs", in proceeding IEEE RADECS 2013, Oxford, UK

[25] W. Zhang et al., "Microarchitecture soft error vulnerability characterization and mitigation under 3D integration technology," in International Symposium on Microarchitecture, 2008, pp. 435–446.

[26] M. Poremba et al., "DESTINY: A Tool for Modeling Emerging 3D NVM and eDRAM caches," in DATE, 2015.

[27] I. Haque and V. Pande, "Hard Data on Soft Errors: A Large-Scale Assessment of Real-World Error Rates in GPGPU," in Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on, 2010, pp. 691–696.

[28] L. Gomez et al., "GPGPUs: How to Combine High Computational Power with High Reliability," in DATE, 2014.

[29] http://*caffe.berkeleyvision.org*

[30] http://vision.stanford.edu/teaching/cs231b_spring1415/slides/alexnet_tugce_kyunghee.pdf

[31] *https://pjreddie.com/darknet/*

[32] Modern GPUs Radiation Sensitivity Evaluation and Mitigation Through Duplication With Comparison. IEEE TRANSACTIONS ON NUCLEAR SCIENCE, VOL. 61, NO. 6, DECEMBER 2014

[33] https://en.wikipedia.org/wiki/Ping_(networking_utility)

[34] http://www.tomshardware.com/news/nvidia-volta-gv100-gpu-ai,35297.html

[35] https://bitsworkshop.org/archive/archive2002/2002s4.pdf

[36] http://www.phd-dauin.polito.it/pdfs/Mauricio%20DE%20CARVALHO_thesis.pdf

[37] Yi Yang and Shawn Newsam., "Bag-Of-Visual-Words and Spatial Extensions for Land-Use Classification," ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM GIS), 2010. The original satellite images were from the USGS National Map Urban Area Imagery collection for various urban areas in the USA. This material was based on work supported by the National Science Foundation under Grant No. 0917069

[38] Meza, J. Revisiting Memory Errors in large-Scale Production Data Centers: Analysis and Modeling of New Trends from the Field. Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Network, Rio de Janeiro, Brazil, June 2015.

[39] Hwang, A. (2012) Cosmic rays don't strike twice: understanding the nature of DRAM errors and the implications for system design. SIGARCH Comput Archit News 40(1):111–122

[40] Kim, Y. Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors. ISCA'14 (2014)

[41] http://arstechnica.com/security/2015/03/cutting-edge-hack-gives-super-user-status-by-exploiting-dram-weakness/

[42] Gruss, D. Rowhammer.js: A remote software-induced fault attack in JavaScript. Graz University of Technology, Austria
[43] Pessl, P. Reverse engineering Intel DRAM addressing and exploitation. CoRR abs/1511.08756 (2015)
[44] http://cdn.teledynelecroy.com/files/whitepapers/tuningddr4_for_power_performance.pdf
[45] http://prod.sandia.gov/techlib/access-control.cgi/2008/085577.pdf
[46] http://prod.sandia.gov/techlib/access-control.cgi/2008/085577.pdf
[47] http://hepwww.rl.ac.uk/CMSecal/Irradiation/2004/Report%20on%20Leakage%20Current.pdf