Considerations for GPU SEE Testing

Edward J. Wyrwas
edward.j.wyrwas@nasa.gov
301-286-5213
Lentech, Inc. work performed in support of NEPP

Acknowledgment:
This work was sponsored by:
NASA Electronic Parts and Packaging (NEPP) Program

## Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>DUT</td>
<td>Device Under Test</td>
</tr>
<tr>
<td>GPU</td>
<td>Graphics Processing Unit</td>
</tr>
<tr>
<td>MBU</td>
<td>Multi-Bit Upset</td>
</tr>
<tr>
<td>NEPP</td>
<td>NASA Electronic Parts and Packaging</td>
</tr>
<tr>
<td>PTX</td>
<td>Parallel Thread Execution</td>
</tr>
<tr>
<td>RTOS</td>
<td>Real-time Operating System</td>
</tr>
<tr>
<td>SBU</td>
<td>Single-Bit Upset</td>
</tr>
<tr>
<td>SEE</td>
<td>Single Event Effect</td>
</tr>
<tr>
<td>SEFI</td>
<td>Single Event Functional Interrupt</td>
</tr>
<tr>
<td>SEU</td>
<td>Single Event Upset</td>
</tr>
<tr>
<td>SIMD</td>
<td>Single Instruction Multiple Data</td>
</tr>
<tr>
<td>SoC</td>
<td>System on Chip</td>
</tr>
</tbody>
</table>

Outline

• GPU technology
• The setup around the test setup
• Parameter considerations
• Lessons learned
Technology

- Graphics Processing Units (GPU) & General Purpose Graphics Processing Units (GPGPU)
  - Are considered a compute device or coprocessor
  - Is not a standalone multiprocessor

- Using high-level languages, GPU-accelerated applications run the sequential part of their workload on the CPU – which is optimized for single-threaded performance – while accelerating parallel processing on the GPU.
Purpose

• GPUs are best used for single instruction-multiple data (SIMD) parallelism
  – Perfect for breaking apart a large data set into smaller pieces and processing those pieces in parallel

• Key computation pieces of mission applications can be computed using this technique
  – Sensor and science instrument input
  – Object tracking and obstacle identification
  – Algorithm convergence (neural network)
  – Image processing
  – Data compression algorithms
Device Selection

- Unfortunately, GPUs come in multiple types, acting as primary processor (SoC) and coprocessor (GPU)

Nvidia TX1 SoC

Intel Skylake Processor

Nvidia GTX 1050 GPU

Smart Phones

AMD RX460 GPU
Device Software

- Does it need its own operating system?
  - E.g. Linux, Android, RTOS

- Can we just push code at it?
  - E.g. Assembly, PTX, C

- Payload normalization
  - Can we run the same code on the previous generation
    and next generation of the device?
  - Cannot with CUDA code; can with OpenCL

Real-time Operating System (RTOS)
Parallel Thread Execution (PTX)
CUDA is a parallel computing platform and application
programming interface model created by Nvidia

Payloads

• **Visual Simulations**
  – Sample code
  – Fuzzy Donut (i.e. Furmark)

• **Sensor streams**
  – Camera feed
  – Offline video feed

• **Computational loading**
  – Scientific computing models

• **Easy Math**
  – $0 + 0 \ldots$ wait \ldots should = 0
Test Setup

- Things to consider in the test environment
  - Operating system daemons
  - Location of payload and results
  - Data paths upstream/downstream
  - Control of electrical sources
  - Temperature control (i.e. heaters) in a vacuum

- Things to consider in the device under test (DUT)
  - Is the die accessible?
  - What functional blocks are accessible?
  - Which functions are independent of each other?
  - Does it have proprietary or open software?
Test Environment

• Beam line
  – DUT testing zone where collateral damage can happen
  – Shielding for everything non-DUT

• Operator Area
  – Cables, interconnects and extenders
  – Signal integrity at a distance
  – “Everything that was done in a lab, in front of you on a bench, now must be done from a distance…”
Test Environment (Cont’d)
Arbiter Platform

- Hardware Info Gathering
  - Thermocouples
- Does not include any in-situ monitoring capabilities of the payload software

Power Supply Control Computer

Power Supply A

Power Supply B

Power Feed Switch Control

Network Router

FTP storage

Interposer

Riser Cable

GPU

Headless Display

Real Display

IP Camera

Motherboard

Network Access

Power Switch

Software Info Gathering
- GPU (I, V, P)
- Motherboard (I, V, P)
- Memory dump
- System logs

Operator Area

KVM

Laptop 1

Laptop 2

Test Environment (Cont’d)

Tripod and mounting

External power

Power injection

Arrows and circle mark locations of the lead and acrylic block fortresses
Test Environment (Cont’d)
DUT Health Status

• Accessible nodes
  – Network
    • Heart beat by inbound ping
    • Heart beat by timestamp upload
  – Peripherals response
    • “Num lock”
  – Visual check
    • Remote
    • Local
    • Local with remote viewing
  – Electrical states
Monitoring Data

Monitoring Data (Cont’d)

- Significant digits are important
- Resolution is needed for correlation
  - Faster sampling speed
  - Smaller units (µV or mV, not Volts)
Monitoring Data (Cont’d)

• Even better (albeit being a mock up):

What does a failure look like?

Request Timed Out
Destination Host Unreachable

Your PC ran into a problem and needs to restart. We’re just collecting some error info, and then we’ll restart for you. (0% complete)

If you’d like to know more, you can search online later for this error: WHEA_UNCORRECTABLE_ERROR

Failures

Latch up situations

12 V Current

Every test is another learning experience

- “Is the laser alignment jig in the beam path…”
- Nuances with controllable nodes
  - DUT power switch
  - Remote power sources
  - DUT electrical isolation from test platform
  - Thermal paths
- Improvements are always possible, but preparation time may not be as abundant
- Prioritization during development is important
  - Software payload
  - Hardware monitoring
  - Remote troubleshooting capabilities
Conclusion

- NEPP and its partners have conducted proton, neutron and heavy ion testing on several devices
  - Have captured SEUs (SBU & MBU),
  - Have seen traceable current spikes,
  - But predominately have encountered system-based SEFIs

- 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 outside of being exercised

- Progress has been made in proving out multiple ways to simulate and enumerate activity on the DUT
  - Narrowing down on a universal test bench
  - End goal is to make test code platform independent