Towards an Integrated GPU Accelerated SoC as a Flight Computer for Small Satellites

Caleb Adams, Allen Spain, Jackson Parker, Matthew Hevert, James Roach, and Dr. David Cotten
Overview

- UGA SSRL
- The MOCI Mission
- What we have done
- What we’re doing
- (if time permits) Applications
UGA SSRL

- Founded in 2016 by a team of Undergrads
  - How hard could it be?
- Faculty supported
- 2 funded cubesat missions
  - MOCI - AFRL UNP
  - SPOC - NASA USIP
- Advanced topics in remote sensing
- 5 Grad Students
The MOCI Mission

- Multiview Onboard Computational Imager
- 6U cube satellite
- ~450km orbit - ~6m GSD
- Goal is to generate 3D terrain models in near real-time
- Flying an Nvidia Tegra X2i GPU SoC
- Funded by AFRL
The CORGI Board

- **Core GPU Interface Board**
- An additional primary OBC is still needed
  - Clyde Space OBC is currently used
    - Contains ARM Cortex M3
- Designed for Cubesats
  - PC/104+ standard
- Compatible with the Nvidia TX2 and TX2
- Standard Procedures: Conformal Coating, Outgassing, Staking, etc...

The Nvidia TX2i mounted onto the UGA SSRL's CORGI board
Nvidia Jetson TX2i

- Pascal GPU - 256 CUDA cores
- ARMv8
  - Nvidia Denver 2 (dual-core)
  - ARM Cortex A57 MPCore Module (quad-core)
- 8GB LPDDR4 - 28 GB/s peak memory bandwidth
- 32GB eMMC Flash Memory
- Software enabled ECC
Minimizing the TX2i OS

- Ubuntu 16.04 LTS based
- Busybox Jetson Root FS
- JHU Dart team has solutions we are moving to
  - We used to script FS generation and dependency population
- Hardest parts are
  - maintaining all packages needed
  - maintaining CUDA compatibility
CORGI Software & Telem Monitoring

- Connects OBC to TX2i via 500Kb/s TTL UART
  - OBC will act as ‘Master’ initiating all communications to the TX2i
- API implemented on each side allowing for the OBC to send commands when needed and receive telemetry when requested
- Upon detection of anomalous behavior the OBC has the ability to hard reset the TX2i
CORGI Watchdog

- The OBC will use telemetry received from the TX2i to monitor its state.
  - If communication is lost or commanded actions take too long OBC will force a hard reset
- OBC will model the TX2i by implementing a Finite State Machine that will keep track of the TX2i state and take corrective action if necessary
Thermal Analysis

- We assume a lot of bad things here
  - TDP of 15 Watts
  - Realistic load of 7.5 Watts
- We have a TVac chamber we will kills some boards in
- Boards modeled as FR-4
- Ansys + Simplified Model
- Aluminium 6063-T5
- Carbice Space TIM used
  - Low CVCM and TML
- Goes from 160 C to 50 C with TIM and mount to frame
Lowering Power

- Easiest Way to improve Thermals!
  - For me as a software guy at least
- Shut off cores with `$sudo nvpmodel #` and set modes to minimize power
- decrease clock frequency/self throttle
- Choice of mode is design decision
- We get between 3 and 7.5 Watts doing this

<table>
<thead>
<tr>
<th>Mode</th>
<th>Name</th>
<th>Denver 2</th>
<th>ARM A57</th>
<th>GPU Hz</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Max-N</td>
<td>2</td>
<td>4</td>
<td>1.3</td>
</tr>
<tr>
<td>1</td>
<td>Max-Q</td>
<td>0</td>
<td>4</td>
<td>0.85</td>
</tr>
<tr>
<td>2</td>
<td>Max-P Core-All</td>
<td>2</td>
<td>4</td>
<td>1.12</td>
</tr>
<tr>
<td>3</td>
<td>Max-P ARM</td>
<td>0</td>
<td>4</td>
<td>1.12</td>
</tr>
<tr>
<td>4</td>
<td>Max-P Denver</td>
<td>1</td>
<td>1</td>
<td>1.12</td>
</tr>
</tbody>
</table>

TX2 / TX2i power modes
Physical Radiation Mitigation

- Dunmore Aerospace Satkit
- SIGNIFICANT future work in this section
  - Working with JHU APL, using facilities at University of Washington
- Aluminum block around the TX2i
- Cheap LEO/cubesat solutions

Aluminized Kapton shielding on a 3U face
AFC - Moving Towards a Flight Computer

- The AFC (Accelerated Flight Computer)
  - Currently a tangled mess of wires in our ESD area
- An upgraded CORGI - (this is outside the scope of MOCI mission)
- We Need to:
  - Develop Radiation Mitigation Techniques
  - Diversify Interfaces
  - Implement Watchdog
  - Add Persistent Memory
  - Improve Thermals
  - Lower The Power Consumption
## System Specs Overview

### I/O
- PC/104+, I2C, SPI, GPIO
- QSPI Expansion Header
- 2x RJ-45
- 2x USB type C
- Micro USB (FTDI)

### Nvidia Tegra X2i
- 2x Denver ARM Cortex A57
- 256 Pascal Arch GPU
- 8GB LPDDR4
- 32GB ECC support

### SmartFusion2
- ARM Cortex M3
- ARM Cortex M3 SoC FPGA
- 4x256 DDR3 Memory Bank

### Both
- 2GB (Cypress CYRS16B256)
- 1 Gb SPI Flash on SPI 0*
- 1 Gb SPI Flash on SPI 1*
Architecture Overview

- SF2 controls TX2i over eth
- Shared NAND flash between SF2 and TX2i
- Power supply through Sat stack
- Eth. is primary communications for additional Jetson modules
- SD card only for development
Integration Overview

- Bidirectional Logic shifters needed
  - convert 1.8v (CMOS) TX2i logic to 3.3v (LVTTL) SF2
- Shifters required for (onboard) serial data transfer
- Power discharge circuitry
- USB and SD card interfaces
- More in extra slides
Bootloader TMR

- Using U-Boot to add custom boot time functionality to the TX2i
  - The principle of TMR safeguards against catastrophic OS corruption
- 3 identical OS images stored in memory, along with hashes of the images
  - Hashes used to determine if an image has been corrupted by radiation
  - Hash stored in triplicate to protect against hash corruption
- If corruption is detected on all 3, bootloader will try to reconstruct a valid image
  - Uses principle of majority voting to determine which parts of the images are corrupted
  - Relies on unlikeliness of the exact same bit being corrupted on each image
Unifying Memory

- Tx2i and Smartfusion 2 both share persistent SPI flash memory
  - Shifter (MUX) required (controlled by SF2)
  - Added overall system storage
    - Radiation hardened
  - Data can be accessed when coprocessor (TX2i) is powered off
  - Sharing mutually relevant files
FPGA as a Watchdog

- FPGA contains a Finite State Machine
  - updated via regular telemetry from the system
- Will have the ability to hard or soft reset the TX2 upon detection of an anomalous state
- IO is tested against simple checks to continue
PTX overview

- PTX is a low level VM (Virtual Machine) and ISA (Instruction Set Architecture).
- Compatible with all CUDA capable GPUs
- Written like any other assembly language
- Breaks into CTAs (Cooperative Thread Arrays) = Thread Blocks
- PTX programs specify the actions of a given thread in a specific thread array
- CUDA compiles into PTX and can be used within CUDA kernels
Block PTX checkpointing

- Simple TMR type design
  - Majority Gate
- Follows finite state machine on the FPGA
- Makes the execution much slower
- Where to place PTX checkpointing is a design choice on its own
  - Last stages of pipelines ideal
  - $B_1$ and $B_2$ of smaller size
- Writing inline PTX
Adding Interfaces

- Ethernet - high speed data transfer, enables many devices on the same network in a peer to peer configuration.
- USB 2.0 (FTDI) - provides debug interface with the SF2
- USB 3.0 (Type C) - Jetson Tx2i command/control interface
  - Low profile
  - Backwards compatibility
  - Good data rate
  - Supports peripheral devices
Future Work

- Better PTX checkpointing
- Better PTX GPU memory bank monitoring
- Multi-Jetson computation
- Radiation Event Correction
- Flight!
Questions?

smallsat.uga.edu
University of Georgia
CORGI BITS

Bidirectional Logic Shift

Power switch
CORGi BITS

SD Card Interface

Power Discharge