

NASA and Blue Origin's Flight Assessment of Precision Landing Algorithms Computing Performance

David Rutishauser¹ and Gavin Mendeck²

NASA Johnson Space Center, Houston, Texas, 77058, United States

Ray Ramadorai³

Blue Origin, Kent, WA 98032, United States

John Prothro⁴

Avenue Technologies and Commodities Inc., Ponca City, Oklahoma, 74604, United States

Thadeus Fleming⁵

Odyssey Space Research, Houston, Texas, 77058, United States

Peter Fidelman⁶

Blue Origin, Kent, WA 98032, United States

NASA's Safe and Precise Landing - Integrated Capabilities Evolution (SPLICE) project continues NASA's work in the development and testing of technologies for Precision Landing and Hazard Avoidance (PL&HA). This paper presents results characterizing how SPLICE flight software utilizes the shared computing resources of the Descent Landing Computer (DLC), one of the PL&HA technologies under development. The SPLICE technologies are being tested as an integrated payload on Blue Origin's New Shepard suborbital vehicle. The results presented in this paper are measured by applications running in and with the flight software both in flight, and in a high-fidelity Hardware-in-the-Loop (HWIL) simulation environment. Linux utilities to measure performance are also executed from the command line in the HWIL configuration. Performance measurements of the SPLICE workloads executing on the DLC provide insight on how efficiently the software is utilizing the DLC resources. Examples of how these measurements have guided improvements in the flight code are presented. In addition, the DLC uses a commercial processor as a surrogate for NASA's High-Performance Spaceflight Computing (HPSC) processor. This work provides insight on how an HPSC system may perform delivering PL&HA capabilities on a future mission. The measurements also can be used to infer architectural requirements for PL&HA capabilities, informing the HPSC project and other flight computer development efforts. Examples of the measurements collected include processor utilization, I/O bandwidth, cache and branch misses, and application profiles.

¹ SPLICE Avionics Lead, Avionic Systems Division.

² SPLICE Flight Software Lead, Aerosciences and Flight Mechanics Division.

³ Principal Technologist Avionics.

⁴ Software Engineer, Avionic Systems Division, NASA JSC.

⁵ Software Engineer, Aeroscience and Flight Mechanics Division, NASA JSC.

⁶ Software Engineer, Advanced Development Programs.

I.Nomenclature

DDL	= De-orbit Descent and Landing
DLC	= Descent and Landing Computer
DQG	= Dual Quaternion Guidance
HDA	= Hazard Detection and Avoidance
HPSC	= High Performance Space Computer
IMU	= Inertial Measurement Unit
LIDAR	= Laser Imaging, Detection and Ranging
MPSOC	= Multi-Processor System on a Chip
PL&HA	= Precision Landing and Hazard Avoidance
RAM	= Random Access Memory
SBC	= Single-Board Computer
SIMD	= Single-Instruction, Multiple Data
SPLICE	= Safe and Precise Landing Integrated Capabilities Evolution
TRN	= Terrain Relative Navigation

II. Introduction

The NASA/Blue Origin 2018 De-orbit Descent and Landing (DDL) Tipping Point includes two flights on the New Shepard launch vehicle with NASA and commercial sensor payloads [1]. Incorporating results from these sensors into real-time guidance and navigation solutions requires a high-performance computer. For the test flights on the New Shepard vehicle, NASA has developed a Descent and Landing Computer (DLC) as the compute platform to integrate the Safe and Precise Landing - Integrated Capabilities Evolution (SPLICE) project technologies under test. Today's DLC is a surrogate unit based on the Xilinx Ultrascale+ Multi-Processor System on a Chip (MPSoC) [2] and does not meet environmental requirements for long duration spaceflight. Future SPLICE development will replace today's surrogate DLC with space-grade hardware such as the High-Performance Spaceflight Computing (HPSC) processor. Multicore or other high-performance processing such as Single-Instruction, Multiple Data (SIMD) is becoming increasingly relevant to capabilities planned for future space missions, including Precision Landing & Hazard Avoidance (PL&HA). An understanding of how the SPLICE algorithms utilize the shared resources of the DLC, during integrated operation in flight, provides valuable insight on the SPLICE payload performance. These measurements not only reveal where performance optimizations are most necessary, but also inform the selection of space-grade hardware.

This paper describes the study approach, results and conclusions associated with the SPLICE DLC performance evaluation. Section III gives a brief description of the system's hardware and software architecture and the algorithms that it executes. Section IV describes the performance metrics collected during the DLC's ground and flight testing and the methodology by which they were collected. Section V provides a summary of the results. Section VI contextualizes the results and describes how they have informed improvements to system performance. Finally, we offer conclusions in section VII.

III.System Description

The DLC system includes a high-performance computer, camera, navigation Doppler LIDAR, and an IMU for measuring attitude rates and lateral accelerations. A Hazard Detection LIDAR will be added for future test flights. The purpose of the DLC's computer is to execute SPLICE algorithm technologies under development, such as extended Kalman Filter guidance, Dual Quaternion Guidance (DQG), and Terrain Relative Navigation (TRN). Hazard Detection and Avoidance (HDA) will be added as a future capability. The DLC and SPLICE algorithms under test are introduced in more detail in [3]. Figure 1 shows the main components of the DLC and its interfaces.

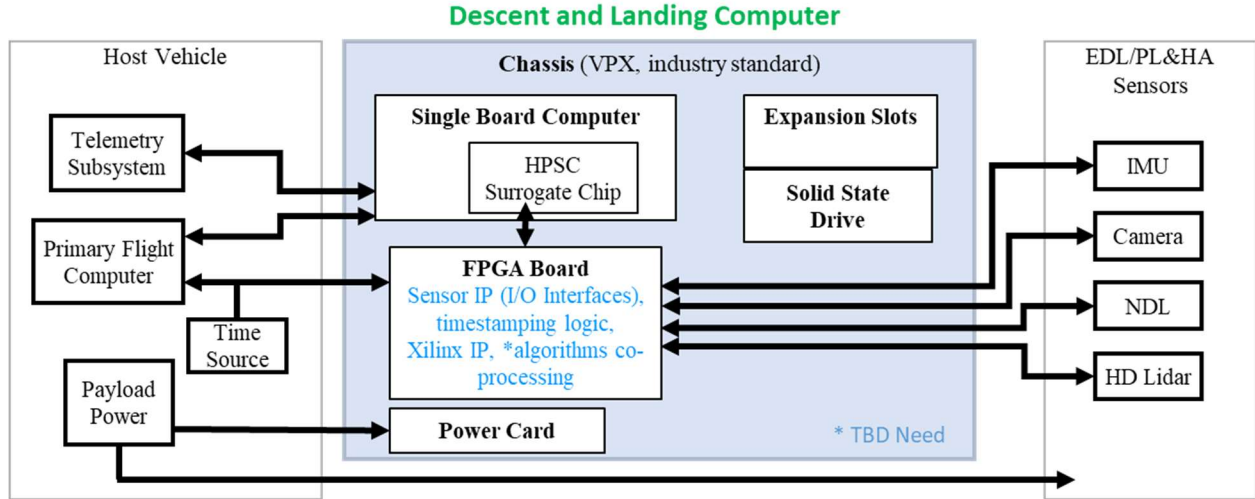


Fig. 1 SPLICE flight test system configuration.

In Figure 1, the SPLICE flight software runs on the Single Board Computer HPSC Surrogate chip (Xilinx Ultrascale+ MPSoC). Typical industry benchmarks, such as Coremark, have been run on the Ultrascale+ MPSoC and are not included in this paper [4]. The metrics reported here are specific to the SPLICE applications. The SPLICE algorithms and other flight software runs in NASA's core Flight System (cFS) framework [5]. This framework uses a structure of "apps" for software services or functions, e.g. a navigation algorithm. Apps are spawned as individual threads, that the underlying Linux operating system allocates to the four processing cores available on the MPSoC chip. Other than this coarse level of parallel operations, and some multithreading found in the TRN libraries, the SPLICE software is not written for parallel execution. Each of the four cores on the MPSoC chip also has a NEON Single-Instruction, Multiple Data (SIMD) co-processor. The NEON vectorizing compiler was enabled for the SPLICE software tested.

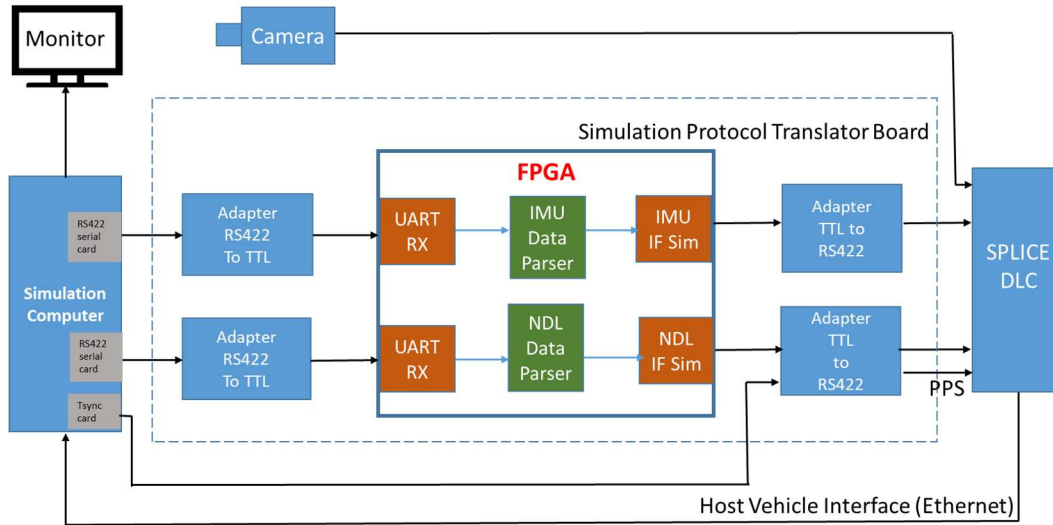


Fig. 2 SPLICE HWIL test system configuration.

The Hardware-in-the-Loop (HWIL) configuration used for the ground data collection is shown in Figure 2. As shown, the TRN camera provides a sensor in the loop and is presented a monitor with images driven by the simulation computer. IMU and NDL sensor models in the simulation provide serial outputs to a protocol translator that provides formatting and electrical translation to present the DLC with these inputs indistinguishable from the real sensor. More details about the HWIL can be found in [5].

The SPLICE system was integrated with Blue Origin’s New Shepard Propulsion Module (PM), which makes a soft propulsive landing on a pad after launch. The PM flight profile provides a relevant analog to the Entry, Descent, and Landing (EDL) profiles for which the SPLICE payload is designed to provide PL&HA capability. Figure 3 shows the locations of the SPLICE components on the top of the PM.

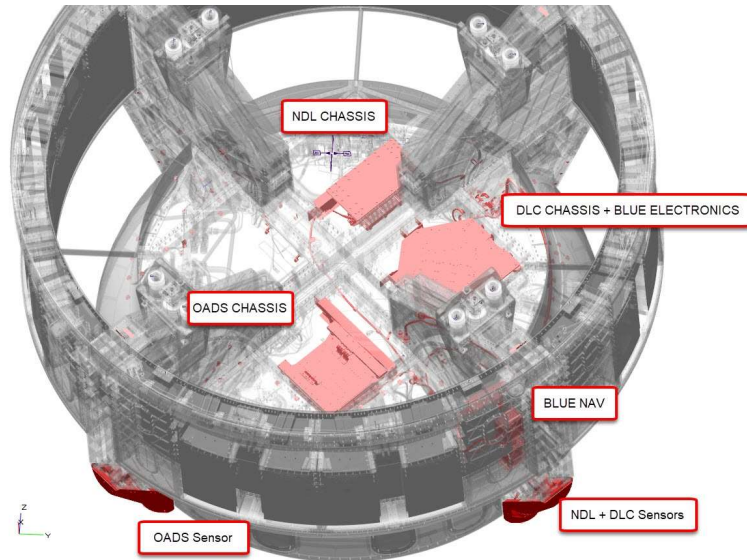


Fig. 3 Locations of the SPLICE system components on the New Shepard PM.

IV. Performance Metrics

This paper compares performance measurements collected during Flight 2 against measurements collected in twelve runs in the HWIL test configuration using the Tipping Point flight 2 DLC build. Measurements were collected by manually executed Linux command line tools, by standalone Linux applications running independently of the flight software, and by the flight software’s monitoring features. Table 1 shows a run matrix detailing the measurements collected. Each run included the flight software’s processor utilization measurement in the Health and Status (*cFS-HS*) application, a before and after *top* and *df* snapshot, and the standalone application *multi-perf*. The command-line utilities *mpstat* and *iostat*, the in-house *threadmon* standalone application, and *perf record* call application profiles were alternatively used in different runs. This was done to reduce overhead and interference from the measurement utilities.

Table 1. Performance Measurement Summary

		Quality	HWIL: Blue-2 FSW & Blue-1 data									FLIGHT Blue-2	HWIL: Blue-2 FSW & data		
Measurement	Tool		1a	2a	3	4	5	6	7	8	9		1	2	4
processor utilization	CFS HS, (1/sec)	●	x	x	x	x	x	x	x	x	x	x	x	x	
	mpstat	●			x										
	threadmon (1/sec)	●		x										x	
	top (before/after)	◐	x	x	x	x	x	x	x	x	x	x	x	x	
SSD usage	df (before/after)	◐	x	x	x	x	x	x	x	x	x	x	x	x	
SSD bandwidth	iostat	●				x									
DDR usage	CFS HS, (1/sec)	◐										x	x	x	
	OS (meminfo)	◐	x	x	x	x	x	x	x	x	x	x	x	x	
DDR bandwidth		○													
Cache misses	multi-perf, (1/sec)	●	x	x	x	x	x	x	x	x	x	x	x	x	
Cache references	multi-perf, (1/sec)	●	x	x	x	x	x	x	x	x	x	x	x	x	
Branch misses	multi-perf, (1/sec)	●	x	x	x	x	x	x	x	x	x	x	x	x	
Branch references	multi-perf, (1/sec)	●	x	x	x	x	x	x	x	x	x	x	x	x	
Instructions	multi-perf, (1/sec)	●	x	x	x	x	x	x	x	x	x	x	x	x	
Cycles	multi-perf, (1/sec)	●	x	x	x	x	x	x	x	x	x	x	x	x	
Context switches	multi-perf, (1/sec)	●	x	x	x	x	x	x	x	x	x				
Application profile	perf record	●				all DQG TRN NLU NLP									

V. Results

Figure 4 (a) shows the top-level processor utilization as measured in the HWIL configuration, measured at 1 Hz using *cFS-HS*. As shown, the stacked percent utilization for four processors remains below half capacity for almost the entire run. The DLC has significant reserve processing potential beyond running the SPLICE software. As described later in the paper, this occurs because the SPLICE software does not yet make efficient use of the hardware resources. Applications with high CPU usage have not yet been redesigned to better leverage the multiple cores. It is also observed that the load across the four cores is not evenly balanced, which reflects the small amount and coarse parallelization in the SPLICE software. Since last year’s publication [3], we have gained the ability to measure resource utilization per thread, thus getting a deeper look at the resource consumption of each SPLICE algorithm. Per-thread instrumentation helps us to select which algorithms to optimize and to quantify the speedup achieved. Figure 4 (b) shows the thread-specific processor utilization for each of the SPLICE algorithms of interest measured by *threadmon*. Because we observed periods of high processor utilization for the DQG and TRN algorithms, we focused on speeding up those algorithms. Figure 4 (b) shows that the navigation algorithms (NLP and NLU) used a very low amount of processor time compared to the other algorithms. Consequently, they were not candidates for further optimization.

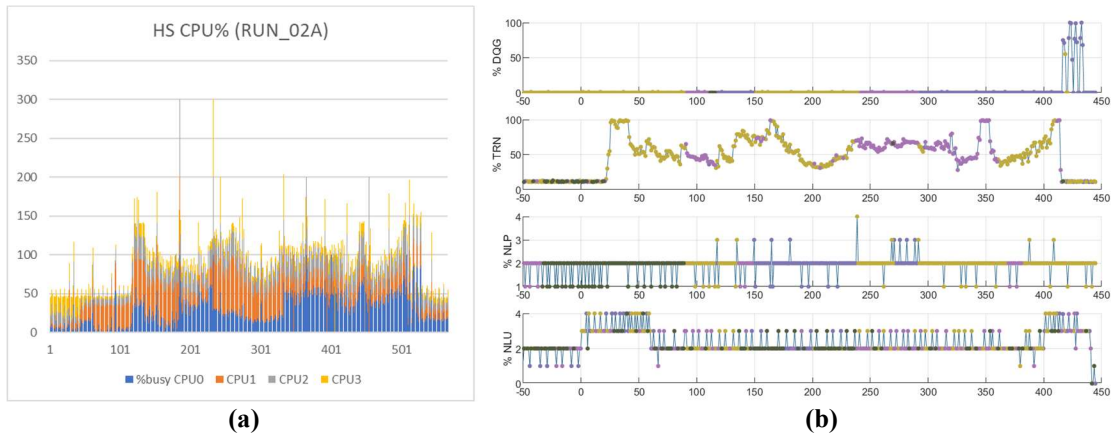


Fig. 4 Processor utilization, HWIL configuration (a) top level, and (b) specific algorithms.

Comparing top-level processor utilization (Figure 4(a)) with thread-specific processor utilization (Figure 4(b)) shows processor spikes that do not directly originate from any of the SPLICE algorithms. These have been observed consistently on every run where thread-specific versus top-level processor utilization was collected. Spikes in RUN_03 are shown in Figure 5 as a representative example. It was observed the spikes are smaller when the TRN camera frame rate is reduced, suggesting it is tied to handling of the imagery. No adverse effect on system timing has been found so investigating the processor spikes has not been prioritized.



Fig. 5 Spike features in processor utilization.

Figure 6 is a comparison of the utilization measured on the flight and two of the HWIL runs. Examining the relative utilization between the processors shows different allocations of the load between CPU cores, but very similar load profiles between simulation and flight. This is evidence that trades and analysis can be done in the HWIL configuration with confidence that the results will be reflected in flight experience.

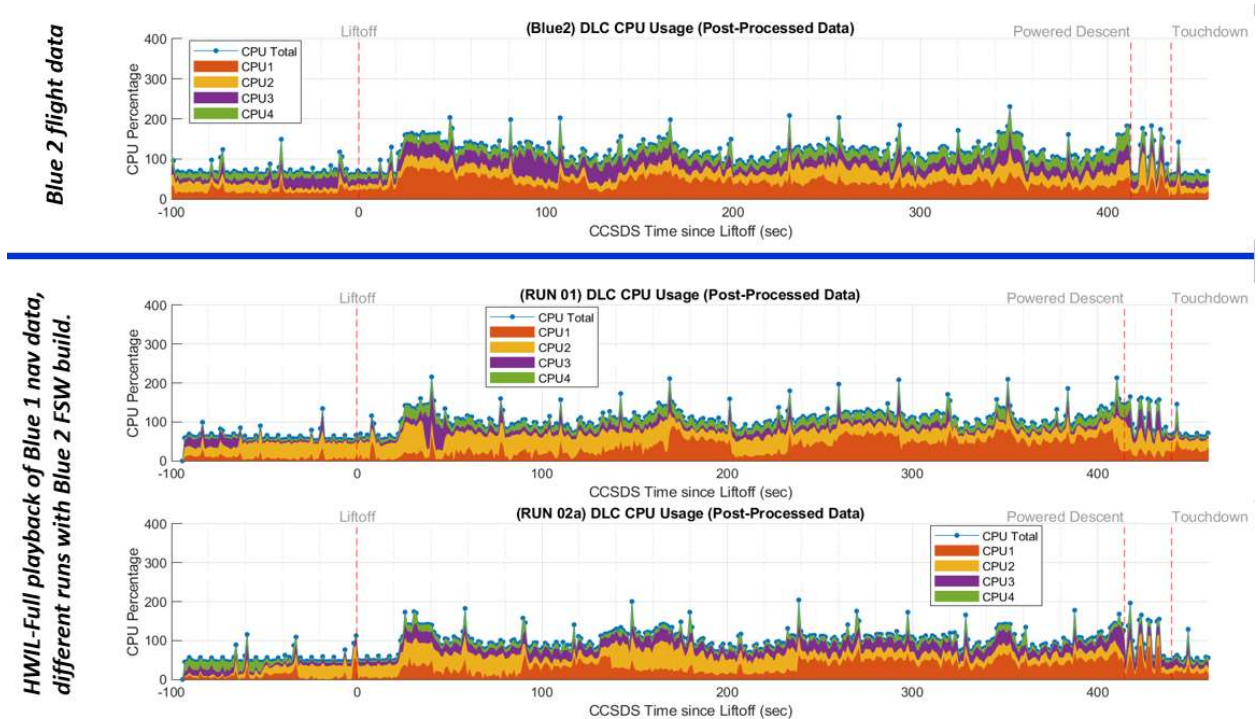


Fig. 6 Comparing Flight CPU Usage to HWIL Test Runs.

Figure 7 shows the measurements collected in simulation and flight from the standalone *multi-perf* application. For each flight software application, *multi-perf* measures cache and branch misses, execution cycles, and instruction count. These measurements are taken for individual application threads. In the figure, HWIL-Lite is the Blue Origin Flight 1 flight software load and Flight 1 flight profile and TRN images read off disk by the software. This best represents Flight 1 software performance. HWIL-Full is the Flight 2 flight software load with Flight 1 flight profile and TRN camera processing displayed images (configuration shown in Figure 2). This represents Flight 2 software performance prior to the flight. HWIL-Emu is the post-flight assessment of the Flight 2 software load with the Flight 2 flight data and profile, and TRN camera images fed via a camera emulator to the DLC.

As shown in Figure 7, there are low cache misses for all GNC applications. This implies the DLC cache sizes are adequate for these applications. There are also low instructions per cycle for DQG, which suggests room for improvement. There is also much similarity between the flight and HWIL-Full Multi-perf results. This implies that optimizations and trades can be evaluated in the simulation configuration with high confidence that the results will extend to flight.

		HWIL-Lite	HWIL-Full	FLIGHT	HWIL-Emu
FSW Build:		~Blue-1 FSW	Blue-2 FSW	Blue-2 FSW	Blue-2 FSW
Data:		Blue-1	Blue-1	Blue-2	Blue-2
Category					
App Tasks Completed Within Time Allotted	DQG				
	TRN				
	NLU				
	NLP				
App CPU Usage over Call, Average	DQG	85%	60%	n/a	40%
	TRN	100%	70%	n/a	70%
	NLU	<3%	2%	n/a	2%
	NLP	<3%	2%	n/a	2%
Instructions / Cycles, Average	DQG	0.5	0.4	0.4	0.4
	TRN	0.6	0.8	0.8	0.8
	NLU	0.7	0.7	0.7	0.7
	NLP	0.5	0.4	0.4	0.4
% Cache Miss, Average	DQG	2%	3%	3%	2%
	TRN	1%	2%	2%	2%
	NLU	1%	1%	1%	1%
	NLP	3%	3%	3%	3%
% Branch Miss, Average	DQG	9%	7%	7%	7%
	TRN	6%	4%	4%	4%
	NLU	8%	4%	4%	4%
	NLP	16%	17%	16%	17%

Fig. 7 Comparing Flight Multi-perf to HWIL.

Call graph profiles for the SPLICE algorithms show that the execution environment does not introduce significant overhead, meaning that the performance bottleneck is the algorithms themselves. The algorithms are still bottlenecked on processor single-threaded performance and there is no obvious source of further speedup besides optimizing and parallelizing the algorithms themselves. Parallelization is the most promising direction because the surrogate DLC is multi-core, as is every other radiation tolerant processor in its performance category, and because FPGAs are also best at accelerating parallel workloads. Beyond parallelization, offloading parts of the computation to the FPGA may also be necessary to hit rate targets, especially for DQG. Increasing the vectorization on the Ultrascale+ MPSoC also has good potential for increasing performance.

The memory usage is shown in Figure 8. Memory usage was low throughout the entire flight, indicating the DLC's 4 GB of onboard RAM is more than sufficient for the SPLICE algorithms. Memory usage trended upward during the first 200 seconds of flight but was flat for the rest of the flight. This rules out any substantial memory leaks, but the variability should be investigated to better understand whether the memory allocation is occurring in the SPLICE algorithms themselves, in other processes running on the DLC, or in the underlying operating system.

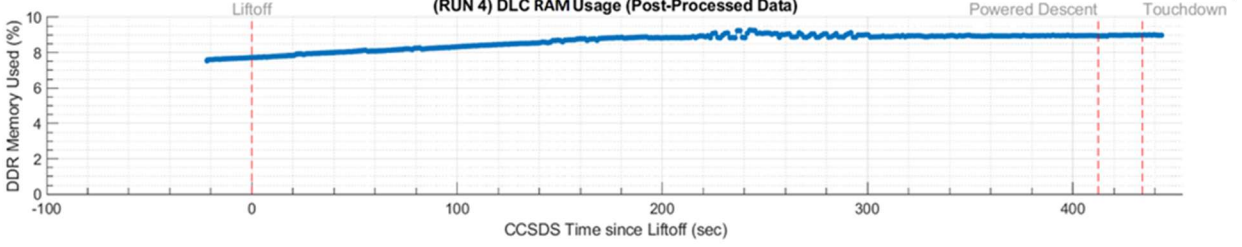


Fig. 8 RAM usage.

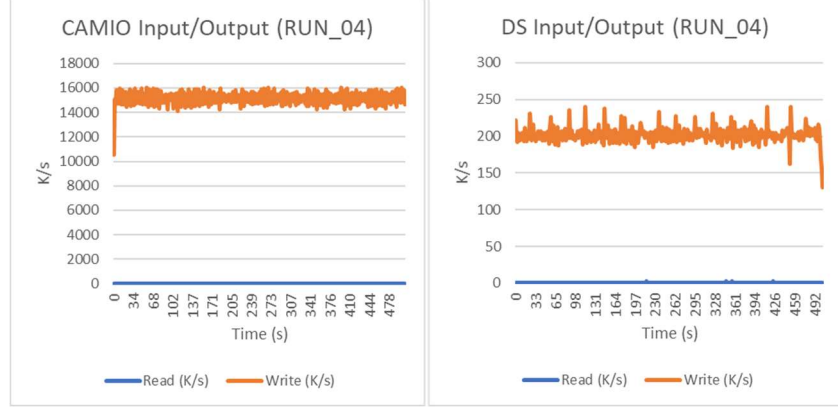


Fig. 9 Measurements of I/O bandwidth.

Figure 9 shows I/O bandwidth measurements collected with the *iotop* utility in the HWIL configuration. CAMIO is the cFS application that reads and saves the TRN camera image data, and represents the highest bandwidth at 16 Mbyte/sec. Non-image data is recorded using the Data Storage (DS) application, which uses a factor of 80 less bandwidth than CAMIO. These two applications account for >99.9% of all disk writing during the test run.

VI. Discussion

Flight software updates between two rounds of HWIL testing impacted functionality, CPU utilization, and *multi-perf* metrics. Multiple DQG improvements were deployed and tested to reduce processing time to achieve successful powered guidance solutions. Optimizations were also made to the TRN code with the intent of reducing its processor utilization. Results show that these optimizations have been effective: while the algorithm still saturates a CPU core at 100% utilization for some portion of the mission, the duration of the saturation has decreased. TRN's increased efficiency is also visible in other statistics, with some other improvements being fewer cycles and fewer missed branches. The performance metrics informed these software improvements that were included in the Flight 2 build. Not shown in the paper are the call graph profiles produced for the SPLICE algorithms of interest using the *perfrecord* utility. These profiles are valuable for identifying areas where the code is spending the most time, which represent targets for optimization that will produce the greatest performance improvements. These profiles are collected for a particular thread of interest, which is executing not standalone but in its integrated environment, competing for the shared resources of the DLC. This provides a profile of execution that accurately reflects the flight environment. The demonstrated similarity between the performance metrics in flight and simulation allows for confident continued use of the HWIL to observe the effects of ongoing optimization efforts. These efforts will include examination of how successful the vectorizing compiler is with a particular code, and attempts to increase the level of vectorization. Opportunities to parallelize the code will also be identified and tested.

VII. Conclusion

The results presented in this paper represent the current state of ongoing cycles of performance optimization for the SPLICE hardware and software. The tools developed to do the performance analysis provided the needed insight into the core performance, memory performance, and non-volatile storage performance and enabled the most fruitful areas for optimization of the various algorithms running on the DLC. The DQG and TRN algorithms currently consume the full single threaded compute resource during critical execution windows. Analysis of the results suggest that these algorithms would benefit from further optimization increasing the execution rate as well as providing resources for the addition of new algorithms such as hazard avoidance. Hazard avoidance sensors and algorithms will be added to the next version of the DLC. Performance data from the flight and the HWIL suggest that high utilization of compute resources by the different algorithms do not substantially overlap in time meaning that both will benefit from parallelization. This result suggests that multi-core compute resources on the DLC are sufficient for these kind of landing algorithms. The team will continue working with algorithm and software developers to appraise the performance improvements made so far and to determine what additional software changes are most likely to yield further speedups. These results will also influence the selection of space-grade DLC hardware compatible with the SPLICE algorithms' processing requirements. The performance insights gained in this work informs future PL&HA implementations.

Acknowledgements

This work was performed under Space Act Agreement # ACO-2019-051 between NASA and Blue Origin.

References

- [1] R. Sostaric, J. Carson, S. Pedrotty, D. Rutishauser, T. Tse, J. Sooknanan, F. Amzajerian, A. Cianciolo, C. Kuhl, B. Blair, G. Chen, and P. Chen, "The SPLICE Project: Safe and Precise Landing Technology Development and Testing", Proceedings of the AIAA 2021 SciTech Conference, (virtual), January 2021.
- [2] Xilinx MPSoC Web site: <https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html>. Accessed: 2021-11-09.
- [3] D. Rutishauser, R. Ramadorai, J. Prothro, T. Fleming, and P. Fidelman, "NASA and Blue Origin Collaborative Assessment of Precision Landing Algorithms and Computing", Proceedings of the AIAA 2021 SciTech Conference, (virtual), January 2021.
- [4] Zynq Ultrascale+ MPSoC – System Performance Modelling: <https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18842421/Zynq+UltraScale+MPSoC+-+System+Performance+Modelling>. Accessed: 2021-11-09.
- [5] D.K. Rutishauser and T. Tse, "Hardware-in-the-Loop Testing for Suborbital Flights of the Safe and Precise Landing Integrated Capabilities Evolution (SPLICE) Project Technologies", Proceedings of the AIAA 2020 SciTech Conference, Orlando, FL, January 2020.