Evolution of the Space Shuttle Primary Avionics Software and Avionics for Shuttle Derived Launch Vehicles

Roscoe C. Ferguson
United Space Alliance

As a result of recommendation from the Augustine Panel, the direction for Human Space Flight has been altered from the original plan referred to as Constellation. NASA's Human Exploration Framework Team (HEFT) proposes the use of a Shuttle Derived Heavy Lift Launch Vehicle (SDLV) and an Orion derived spacecraft (salvaged from Constellation) to support a new flexible direction for space exploration. The SDLV must be developed within an environment of a constrained budget and a preferred fast development schedule. Thus, it has been proposed to utilize existing assets from the Shuttle Program to speed development at a lower cost. These existing assets should not only include structures such as external tanks or solid rockets, but also the Flight Software which has traditionally been a "long pole" in new development efforts.

The avionics and software for the Space Shuttle was primarily developed in the 70's and considered state of the art for that time. As one may argue that the existing avionics and flight software may be too outdated to support the new SDLV effort, this is a fallacy if they can be evolved over time into a "modern avionics" platform. The technology may be outdated, but the avionics concepts and flight software algorithms are not. The reuse of existing avionics and software also allows for the reuse of development, verification, and operations facilities. The keyword is ***evolve*** in that these assets can support the fast development of such a vehicle, but then be gradually evolved over time towards more modern platforms as budget and schedule permits. The "gold" of the flight software is the "control loop" algorithms of the vehicle. This is the Guidance, Navigation, and Control (GNC) software algorithms. This software is typically the most expensive to develop, test, and verify. Thus, the approach is to preserve the GNC flight software, while first evolving the supporting software (such as Command and Data Handling, Caution and Warning, Telemetry, etc.). This can be accomplished by gradually removing the "support software" from the legacy flight software leaving only the GNC algorithms. The "support software" could be re-developed for modern platforms, while leaving the GNC algorithms to execute on technology compatible with the legacy system. It is also possible to package the GNC algorithms into an emulated version of the original computer (via Field Programmable Gate Arrays or FPGAs), thus becoming a "GNC on a Chip" solution where it could live forever to be embedded in modern avionics platforms.

Evolution of the Space Shuttle Primary Avionics Software and Avionics for Shuttle Derived Launch Vehicles

Roscoe C. Ferguson
United Space Alliance

1. INTRODUCTION

As a result of recommendation from the Augustine Panel, the direction for Human Space Flight has been altered from the original plan referred to as Constellation. NASA's Human Exploration Framework Team (HEFT) proposes the use of a Shuttle Derived Heavy Lift Launch Vehicle (SDLV) and an Orion derived spacecraft (salvaged from Constellation) to support a new flexible direction for space exploration. The intent is to start development of the Shuttle Derived Heavy Lift Launch Vehicle (**Figure 1**) in 2011 (or shortly afterwards) with possible flights as early as 2016. The SDLV must be developed within an environment of a constrained budget and a preferred fast development schedule. Thus, it has been proposed to utilize existing assets from the Shuttle Program to speed development at a lower cost. These existing assets should not only include structures such as

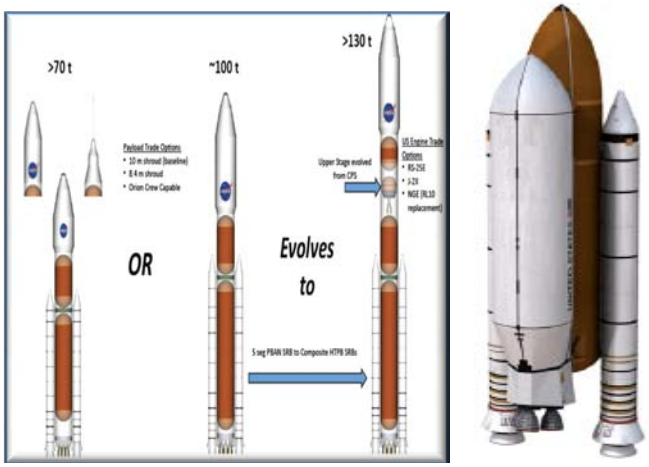

Figure 1: Shuttle Derived Launch Vehicles (Courtesy of NASA)

external tanks or solid rockets, but also the Flight Software which has traditionally been a "long pole" in new development efforts. The avionics and software for the Space Shuttle was primarily developed in the 70's and considered state of the art for that time. As one may argue that the existing avionics and flight software may be too outdated to support the new SDLV effort, this is a fallacy if they can be evolved over time into a "modern avionics" platform. The technology may be outdated, but the avionics concepts and flight software algorithms are not. The reuse of existing avionics and software also allows for the reuse of development, verification, and operations facilities. This is another hidden or forgotten cost that can over strain a budget and schedule due to challenges such as parallel development efforts.

This paper discusses an approach to ***evolve*** the current Space Shuttle avionics and software assets to support the proposed SDLV. The keyword is ***evolve*** in that these assets can support the fast development of such a vehicle, but then be gradually evolved over time towards more modern platforms as budget and schedule permits. The "gold" of the flight software is the "control loop" algorithms of the vehicle. This is the Guidance, Navigation, and Control (GNC) software algorithms. This software is typically the most expensive to develop, test, and verify. Thus, the approach is to preserve the GNC flight software, while first evolving the supporting software (such as Command and Data Handling, Caution and Warning, Telemetry, etc.). This can be accomplished by gradually removing the "support software" from the legacy flight software leaving only the GNC algorithms. The "support software" could be re-developed for modern platforms, while leaving the GNC algorithms to execute on technology compatible with the legacy system. It is also possible to package the GNC algorithms into an emulated version of the original computer (via Field Programmable Gate Arrays or FPGAs), thus becoming a "GNC on a Chip" solution where it could live forever to be embedded in modern avionics platforms.

2. OVERVIEW OF SPACE SHUTTLE AVIONICS AND PRIMARY AVIONICS SOFTWARE

2.1 Avionics

The Space Shuttle avionics system consists of multiple computers and multiple buses (**Figure 2**). The computer set consists of five AP-101S General Purpose Computers (GPCs) which are from the IBM Modular Military Computer (MMC) technology line [1].

A GPC can be separated into a CPU and Input Output Processor (IOP). Each GPC has twenty-six independent processors consisting of a CPU, Master Sequence Controller (MSC), and twenty-four Bus Control Elements (BCEs). The

MSC and BCEs are part of the IOP. The CPU performs general processing, while the IOP performs input and output processing [1].

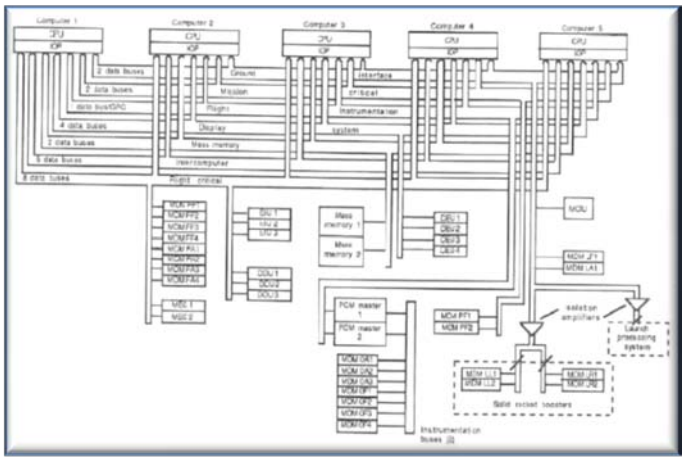

Figure 2: Space Shuttle Avionics (Courtesy of NASA)

The IOP interfaces to the twenty-four serial data buses. Twenty-three of the buses are cross strapped (available to all GPCs) while the twenty-fourth is not. The twenty-four buses are grouped into seven categories. These are Inter-computer (five buses), Mass Memory (two buses), Flight Instrumentation (five non-cross-strapped buses, one per GPC), Payload (two buses), Launch Data (two buses), Display (four buses), and Flight Critical and Sensor Control (eight buses). The eight Flight Critical and Sensor Control Buses are organized into four strings. Each string consists of two buses. The concept provides two paths to the IO devices. This renders four logical buses. However, if a bus path fails, the option is provided to switch to the alternate path via a port mode operation [1].

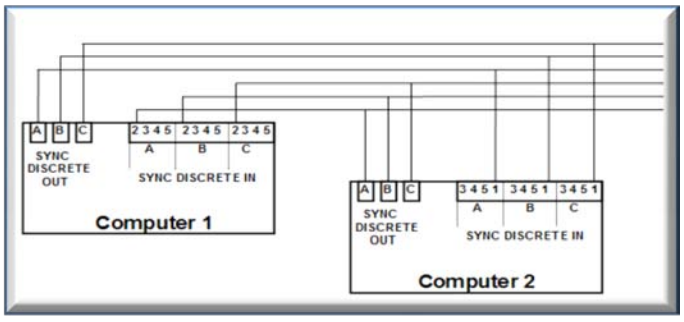

Figure 3: Synchronization Discrete Set (Courtesy of NASA)

The IOP provides a discrete interface to support input and output operations for switches and controls. The discrete set also provides a fast communication path between GPCs to support synchronization operations (**Figure 3**) [1].

2.2 Primary Avionics Flight Software (PASS)

The Primary Avionics Flight Software or PASS is the center piece of the safety critical avionics of the Space Shuttle. Without it, the vehicle is inoperable. Its failure can result in catastrophic failures resulting in the loss of life or the inability to achieving mission success [1]. The software has a proven track record of low error rates and high quality [2] [3]. These attributes were obtained via an enormous investment in all aspects of the software life cycle of continuous process improvements over 30 years.

The software is composed of system software and applications. The system software interfaces with the hardware and provides services to applications. It is an aggregate of the Flight Computer Operating System (FCOS), System Control, and User Interface. Applications provide the behavior of the vehicle [1].

FCOS is the operating system and is responsible for the low-level control over the general purpose computer (GPC). It provides support for Process Management, IO Management ("device drivers" for the IO devices), and Data Processing System (DPS) Configuration. FCOS also provides built-in support for redundancy management operations such as identical input/output and synchronization [1]. Systems Control provides support for the initialization and configuration of the system including the bus network [1].

The User Interface manages the interface between the crew and the software. Input from the crew is routed to the proper application as required. In addition to crew support, the User Interface also provides support for communication with the launch center over the Launch Data Bus (LDB) for pre-launch operations and the Mission Control Center (MCC) via Radio Frequency (RF) transmission.

Applications are composed of Guidance, Navigation, and Control (GNC), Systems Management (SM), and Vehicle Utility (VU) [8]. GNC is responsible for managing vehicle position and velocity for ascent, orbit, and entry. SM is responsible for managing and monitoring the systems of the vehicle and payloads. This includes performing fault detection, annunciation, and control of the environmental systems. VU provides support for the testing, integration, and certification of the vehicle during ground and in-flight operations. It provides both built-in self-test functions and command scripting under the control of the ground and/or crews to support vehicle testing. The command scripting is managed by VU's on-board Test Control Supervisor (TCS) logic. The TCS accepts commands one at a time or in the form of a series of commands (procedure) for which required processes are automatically sequenced to completion.

The design of the PASS is based on a layered architecture (**Figure 4**). FCOS is the bottom layer and provides an interface through Supervisor Calls (SVCs). These calls and their data parameters are encapsulated for applications using a macro interface. The macro interface supports services such as input/output request, bus configuration, overlay requests, synchronization services, process management, interrupt enable/disable, error handling, and time formatting.

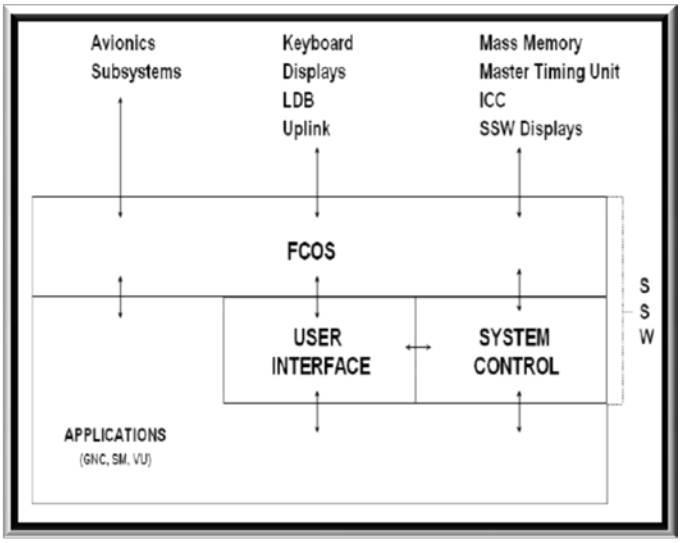

Figure 4: PASS Architecture

3. "GPC ON A CHIP" RESEARCH

From 2004-2006, research was conducted on the feasibility of implementing Space Shuttle avionics assets in Field Programmable Gate Array (FPGA) technology [4] [5] [6]. This technology allows digital electronic designs to be represented in a Hardware Description Language (HDL) where it can be synthesized to hardware components. One such component is a FPGA, which is a reprogrammable device (packaged as a chip) that can be embedded as part of electronic hardware boards. Use of such technology provides benefits such as the reduction of risk due to obsolescence and the ability to evolve hardware designs over time. Obsolescence is reduced because the digital designs are realized in a HDL that can be synthesized to new targets as technology changes over time. Designs can be evolved for the same reason. The HDL can be modified to fix hardware design flaws or incorporate new functionality and synthesized to the FPGA as needed.

This research proposed a conceptual approach where the digital design of the Space Shuttle GPC could be implemented in a HDL and synthesized to a FPGA to support the reuse of the flight software on a SDLV. At the high level, the approach would allow for the instruction set of the GPC to be executed on a FPGA. The benefits include the reuse of FSW and the existing infrastructure required for developing, testing, and verifying the FSW. Another benefit would be that the key avionics assets could become expendable whereas new GPC and/or Multiplexer/Demultiplexer (MDM) devices could be synthesized from the HDL onto FPGAs as required.

Using this approach, the system could be evolved due to the flexibility of having the legacy digital design in a HDL. For example, the design of the IOP could be modified to support varying bus technologies, while isolating the changes from the legacy FSW. The instruction set supporting IO would be unchanged, but the design could support other technology such as 1553 or RS 422. Another aspect would be that a single GPC from the legacy system could exist as one of multiple cards in the backplane of a modern avionics computer. The other boards could provide advanced functionality not part of the legacy system such as advanced failure monitoring algorithms or special interfaces to support the payloads for the SDLV. Such a platform is shown in **Figure 5** from "Use of Programmable Gate Array Technology in Future Space Avionics".

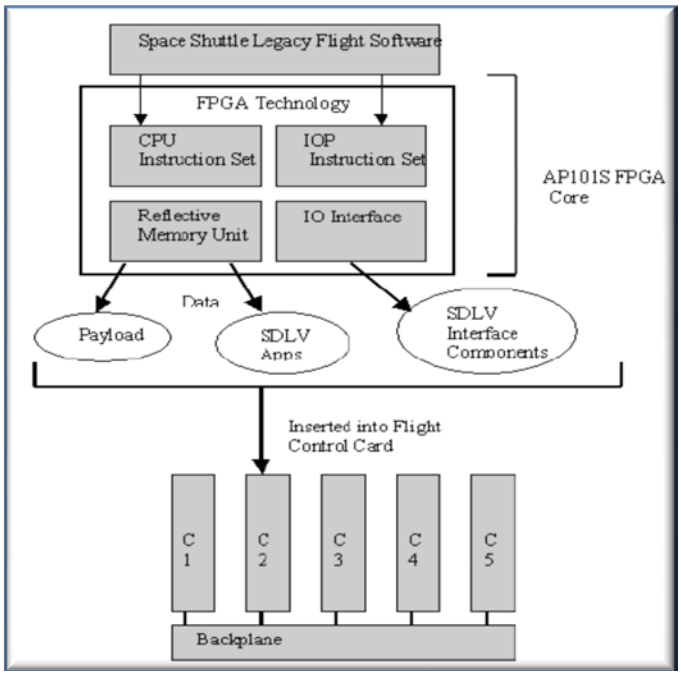

Figure 5: Legacy System in Conceptual Platform

During the research effort, the legacy Space Shuttle FSW was executed on an Altera FPGA development kit containing a prototype of a GPC on a chip. In essence, the prototype version implemented GPC functionality in a 40 mm by 40 mm package requiring less than 1 mw of power vs. the original 19.55 inches long by 7.62 inches high by 10.2 inches wide version requiring 500 w of power (**Figure 6**).

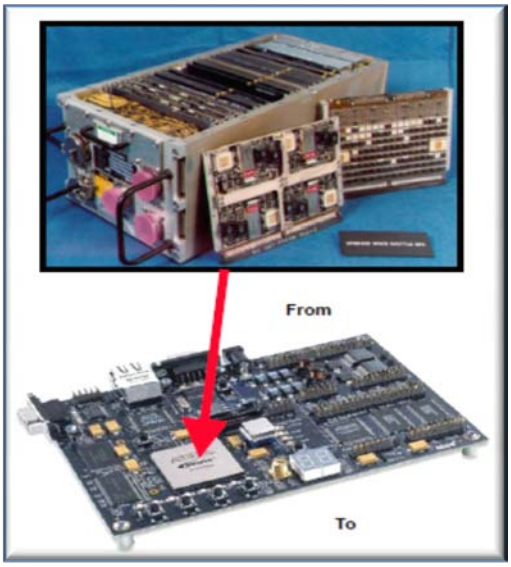

Figure 6: Prototype Repackage of GPC to FPGA

The FSW was executed on the setup in **Figure 7** which also provided support for legacy Shuttle displays (**Figure 8**).

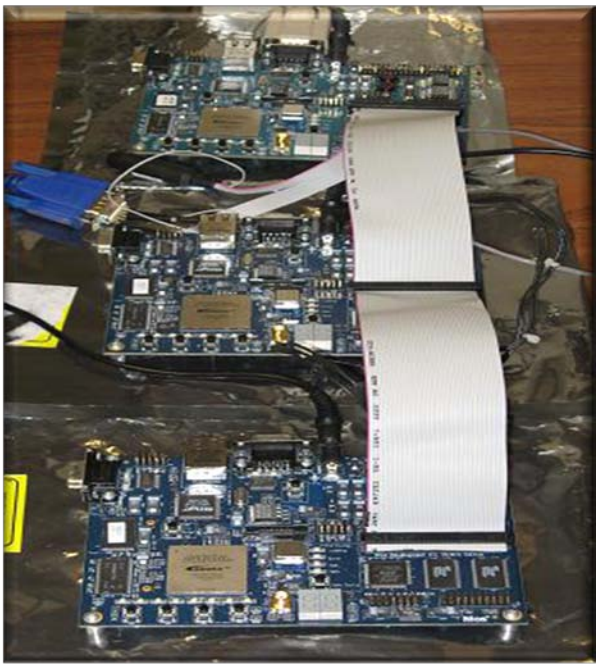

Figure 7: Prototype Setup

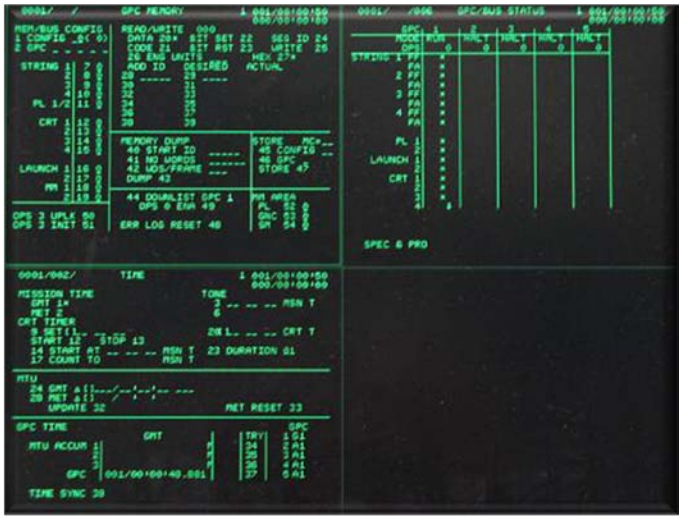

Figure 8: Output from FSW on FPGA Platform

The positive results from the research reveal that it is possible to incorporate Space Shuttle legacy hardware assets into modern avionics systems. This provides a technology insertion point to evolve the existing FSW for use in a SDLV over time. Specifically, this could be used to support the "GNC on a Chip" concept as indicated in Section 1 of this paper.

4. PASS EVOLUTION FOR THE SDLV

The use of the PASS to support the initial development of an SDLV provides a means for lower DDT&E cost. This lower cost is due to the reuse of the on-board software, the reuse of development and verification facilities, and the reuse of operations assets. The reuse of on-board software reduces the high cost of the software development lifecycle. There is a reduction in the formulation of requirements, architecture, design, testing, and verification. These are time consuming activities where each requires analysis, human collaboration, reviews, and artifacts. The ability to modify vs. create in this case can result in initial DDT&E savings. The reuse of development and verification facilities is important. Like the on-board software, these facilities must also be developed and verified. Furthermore, development of these components in parallel to the on-board software can result in chaos. Operations are another hidden cost in new development efforts. The vehicle must interface with facilities such as the Kennedy Launch Center (**Figure 9**) and Johnson Space Center (MCC). The use of the PASS allows for the reuse of the existing infrastructure supporting pre-launch operations such as the "Launch Data Bus" interface and in flight operations such as the uplink and downlink interface.

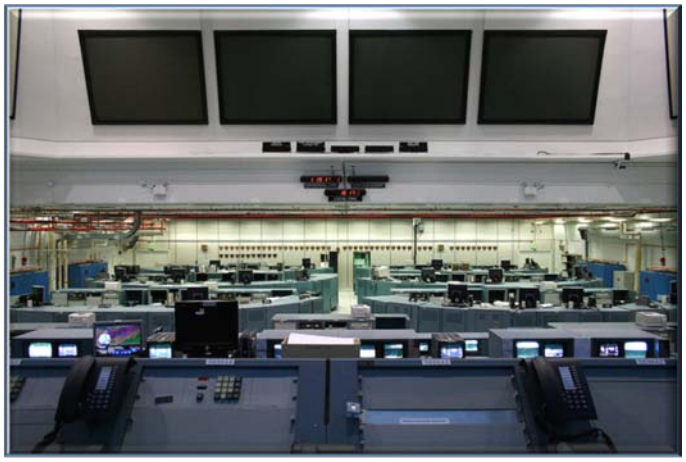

Figure 9: Kennedy Space Center-Launch Operations

It may be argued that the limitations of the legacy hardware and software may make it more feasible to start with a new solution upfront. The legacy hardware is limited to 1 Mbytes of RAM which restricts the size of applications and its support software. Thus, it would appear that this limits the ability to provide advanced functionality even if the PASS is executed on a FPGA (repackaged GPC) embedded in a modern avionics platform. Do these limitations negate the benefit of initial lower DDT&E cost in the long run? The above arguments against using the PASS in a SDLV are feasible unless it is realized that the PASS and supporting infrastructure can be ***evolved*** over time. For example, re-using legacy facilities may still involve risk and cost in maintaining legacy systems. However, the risk and cost can be reduced over time by phasing out and modernizing such systems over time as budget and schedule permits.

The GNC software is the most critical and most expensive to develop, test, and verify. Therefore, it may be prudent to evolve the PASS into a simple GNC controller that supports the dynamic flight phase of the SDLV. This solution could become a permanent fixture as a "GNC only" hardware chip on a FPGA. This way, the "control loop" could be physically isolated from other support software using a hardware solution. Once the "control loop" implementation is stable, it is the least likely of all software to be changed. There could still be a "data" interface to the "control loop" to alter behavior such as flight profiles.

The GNC controller hardware chip solution would be the end product of the natural evolution of the PASS after the initial DDT&E effort for the SDLV. The initial SDLV could use the original or slightly modified PASS to support the first few test and/or production flights. Over time, non-GNC functionality could be stripped out of the PASS and migrated to new software on modern avionics hardware. Thus, support software such as telemetry, command and data handling, and health monitoring could be implemented on new platforms separate from the "control loop" on the legacy platform. The "stripping" process could begin with the initial deliveries because the development and verification facilities are available for use upfront. This includes all the models and simulators required to test the evolving system as it is reduced in scope over time.

There are other variants to this evolved version of the PASS where the GNC controller does not have to result in a hardware chip solution. The solution could also result in the same "GNC only" software executing on an original GPC hardware platform. Thus, there could still be a separation of the "control loop" from the rest of the support software. The "control loop" software would execute in the GPC, while the support software could be implemented and executed using modern technology.

It may be argued that the "GNC only" version can be rewritten in another language which is true. One might even consider a solution where the "GNC only" version is cross compiled to other platforms via modification to the existing compiler supporting the Shuttle program. However, the primary benefit once again is the availability of the infrastructure to test and verify the software. The existing assets provide an infrastructure with flight like hardware interfaced with a production facility with modeling and simulation support. There is also a library of verification and test scripts that could be reused vs. being redeveloped. All are important for initial lower DDT&E cost. The rewritten version may require new target hardware which must be incorporated into the legacy test and verification infrastructure at an additional cost or the development of an entirely new facility at a higher cost.

5. DESIGN OF AVIONICS AND PASS PROMOTES GRADUAL EVOLUTION

The design of the avionics and PASS provides the means for evolution towards a simple GNC controller. The primary method of evolution is "pruning" or the removal of functionality. The availability of the existing development and verification facilities can aid the "pruning" process. As "pruning" occurs, the behavior of the evolved system can be compared with the legacy system using the results from existing test scripts and simulation runs. The following are the key concepts allowing for the evolution of the PASS into a reduced form.

5.1 PASS Organized into Memory Configurations

The Space Shuttle was designed to support an entire space mission (pre-paunch, ascent, orbit, and entry). Due to the memory constraints of the Shuttle GPC, the PASS was organized into memory configurations. Memory configurations are self-contained units of software that each support specific mission phases such as pre-launch, ascent,

orbit, and entry. Thus, to support the SDLV, the PASS can use the pre-launch and ascent overlays as starting points. These overlays already exclude functionality not related to a launch vehicle such as orbit and entry functionality.

5.2 Ascent GNC Software has a Simple Software Architecture

The software architecture for the Ascent GNC FSW is simple and straight forward. At the high level, it is organized as three processes that receive input, process input, and produce output. The three processes support the three basic services. These are flight control, guidance, and navigation. Flight Control is the most important and executes at the highest priority and rate. This is referred to as the High Frequency Executive (HFE). Guidance executes at the next highest priority and rate and is known as the Mid Frequency Executive (MFE). Navigation is the lowest priority and rate and is called the Low Frequency Executive (LFE). These processes are shown in **Figure 10**.

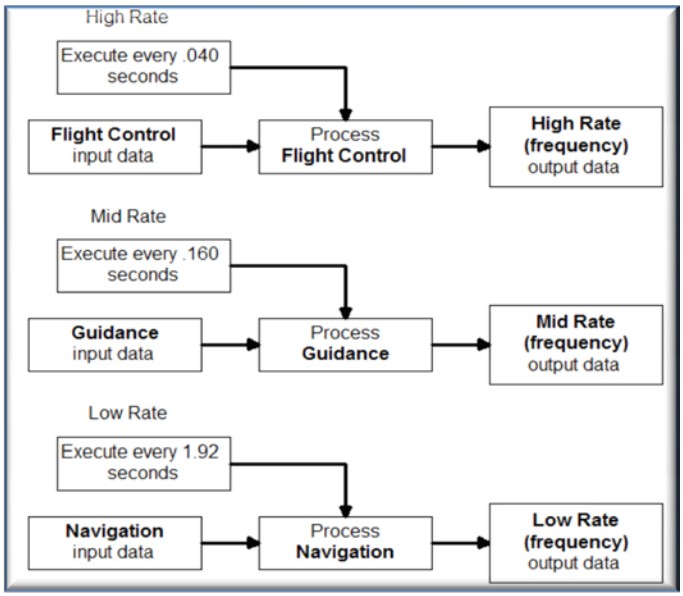

Figure 10: PASS GNC Processes

Within each of the processes, the GNC FSW is organized as an executive that invokes functions using dispatcher tables. These functions contain the pure logic supporting the mission flight profile (**Figure 11**). The importance of the executive approach is that the pure GNC algorithms are isolated from logic supporting other functionality such as user interface processing. This isolation supports the ability to effectively "prune" the software. Most of the "pruning" occurs around the algorithms and not within them. The GNC abort processing could also be "pruned" as this functionality would be obsolete due to technology such as launch abort systems for capsule based spacecraft.

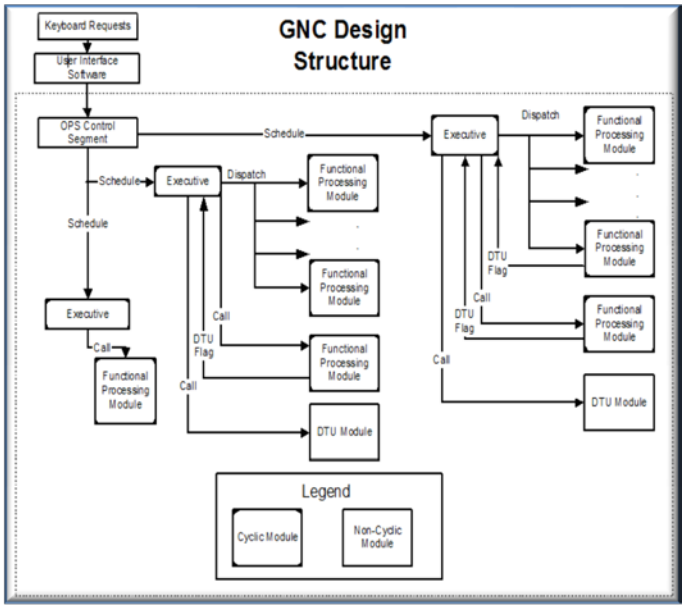

Figure 11: PASS GNC High Level Design

5.3 System Related Services are Encapsulated from Applications

The design of the PASS isolates the intricacies of the system from applications. System Software provides a well-documented interface to support services for applications. System related aspects such as real-time programming techniques (mutual exclusion) or redundancy management are encapsulated. For example, the Space Shuttle provides a redundant capability where multiple computers are configured to support two-fault tolerance for reliability (**Figure 12**). These computers are tightly synchronized and deploy mechanism to discard failing members of a set of synchronized computers. This functionality is primarily implemented using software techniques. However, the functionality is embedded within the System Software and SVC calls isolated from application logic. Thus, the application software contains a minimum amount of embedded code to support system operation.

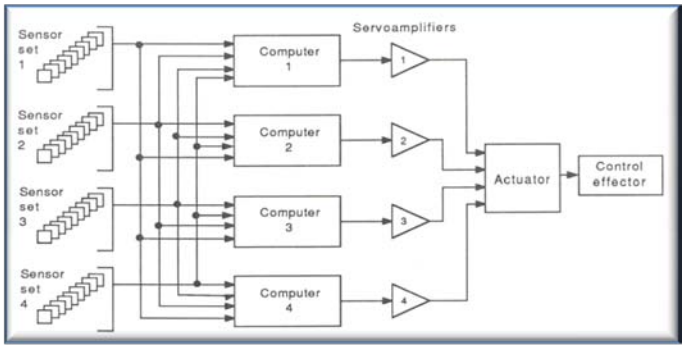

Figure 12: RM is encapsulated from Applications

The relevance of this point is that the application software interfaces with the System Software using an interface which also supports the "pruning" process. Unneeded services can be removed from applications at well-defined points.

5.4 Ascent GNC Software Receives Input and Produces Output via "Buffers"

The algorithms of the GNC application receive input by acquiring data stored in data buffers. The same is true for its output. The use of input and output data buffers provides well-defined points for data transfer to and from the applications. Once again, this provides a clean boundary for evolution. These data buffers are accessed by the Input Output Processor (IOP) of the GPC. When the IOP acquires data from sensors and other devices, that data is stored in the input buffers accessible by the algorithms. When the IOP sends output to effectors or other devices, that data is acquired from the output buffers populated by the algorithms. In essence, the buffers and IOP work like a Direct Memory Access (DMA) engine on modern day computers.

5.5 The GPC was Originally Designed as a Separate CPU and Input Output Processor

The original version of the Space Shuttle GPC had a separate CPU and Input Output Processor (IOP). The CPU and IOP was interfaced using shared memory where both the CPU and IOP could access the same memory space (up to the first 128K). The IOP provided an interface consisting of a set of registers and an instruction set. The IOP instructions for a program are stored in the shared memory space. Software in the CPU initiates IOP programs supporting both input and output. The upgraded version of the GPC combined the CPU and IOP, but maintained the shared memory interface.

The relevance of this design is that the shared memory interface and IOP instruction set provide another well-defined point to decouple the CPU applications from the IOP. The System Software could be modified to remove all traces of support for the IOP and allow another technology to simply place and gather data from buffers used by the applications. Thus, an FPGA design could only implement the CPU portion of the GPC and rely on other technologies to drive the input and output. This effectively supports migrating towards a controller that only supports the GNC application.

6. CONCEPTS OF EVOLUTION

There are many possible concepts or paths that can be taken when evolving the PASS over time. The following are a few listed in an intended order of evolution. The high level approach for each concept is provided. They are not detailed recipes, but are intended to provide direction. Although there is an implied order of evolution, the process can start at any step assuming all dependent steps from prior iterations are performed.

6.1 Step 1 - Use of Existing Avionics Hardware

The use of the existing Space Shuttle avionics hardware can potentially provide the fastest path to a 1st test or production flight. For this path, the first step towards evolving the PASS could be the removal of any user interface software supporting human interaction. This includes the removal of support for displays and keyboard interfaces. These components are not required as the SDLV is a launch vehicle. For interaction, this early system could still rely on launch data bus commanding and/or uplink commanding over RF for interaction.

The avionics software contains both "control loop" and support functionality (such as command and data handling, telemetry, or monitoring) from a slightly modified legacy software system (**Figure 13**). This would also include the redundancy management scheme supporting two-fault tolerance for safety critical operation. Actually, only a subset of the Shuttle avionics hardware may be required to support these flights as this configuration is less complex than the legacy (orbiter, external tank, solid rockets). Thus, software supporting any of the removed hardware can also be scrubbed. The development and verification activities would be performed using the legacy facilities.

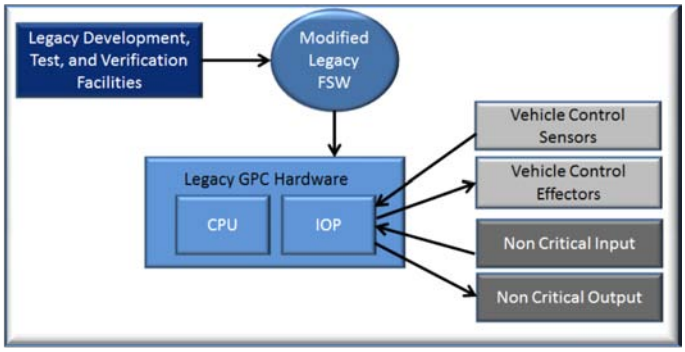

Figure 13: Modified PASS in Legacy Avionics

6.2 Step 2 - Use of New and Existing Avionics Hardware

For this phase of evolution, there is a hybrid of legacy and new avionics hardware. The legacy hardware would support the "control loop", while the new hardware would provide "support" functionality. Thus, functionality such as command and data handling, telemetry, and monitoring could be "pruned" from the legacy software and re-implemented to execute on the new hardware. The legacy PASS evolves to a "control loop" with redundancy management that executes on the legacy GPCs. The legacy system could provide data to the new system over the legacy data bus interface for insight

into its state. The new system would use this information to support monitoring and telemetry operations. There would also be a scaled down interface from the legacy to new system to support commanding capabilities.

For this point in evolution, the "control loop" executing in the legacy GPCs is responsible for acquiring all control related input and producing output for vehicle control (**Figure 14**). The new hardware supports applications for monitoring, telemetry, and command and data handling. Thus, the legacy software is further reduced due to a reduction in support functionality. This version of software can support the scenario where legacy hardware is available due to inventory or a SDLV design that allows for the recovery of avionics assets.

The development and verification activities of the "control loop" software would be performed using the legacy facilities. The development of the support software would be performed on a new platform that may require concurrent development. The concurrent development effort may be more tolerable because the support software is not as safety critical as the "control loop" software.

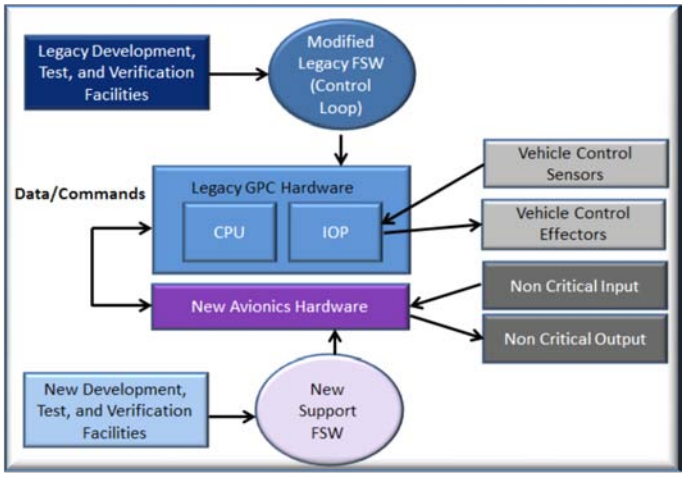

Figure 14: "Control Loop" in Legacy Avionics

6.3 Step 3 - Use of New and Replicated Avionics Hardware

At some point in time, there may not be any available legacy Shuttle hardware assets or there may be a desire to work more readily with modern technology (for avionics hardware). For this case, the replication of legacy Shuttle hardware assets such as the GPCs or Multiplexer/Demultiplexer (MDM) units could be implemented in HDL and FPGA based technologies (**Figure 15**). This can support the production of new components for the SDLV and provide the flexibility to evolve the legacy hardware to support new technologies.

The IOP of the GPC could be evolved to interface with new bus technologies, while at the same time encapsulating these changes from the flight software. The reduced version of the PASS (as a "control loop") with redundancy management would execute on the replicated hardware. The replicated hardware would still interface with the new avionics hardware as in Step 2. Furthermore, there could be additional simplifications such as the removal of legacy mass memory units and legacy initial program loads. These activities could be performed by starting from pre-stored memory images as in the legacy check-point restart approach. This would result in a simplification and reduction in the System Software.

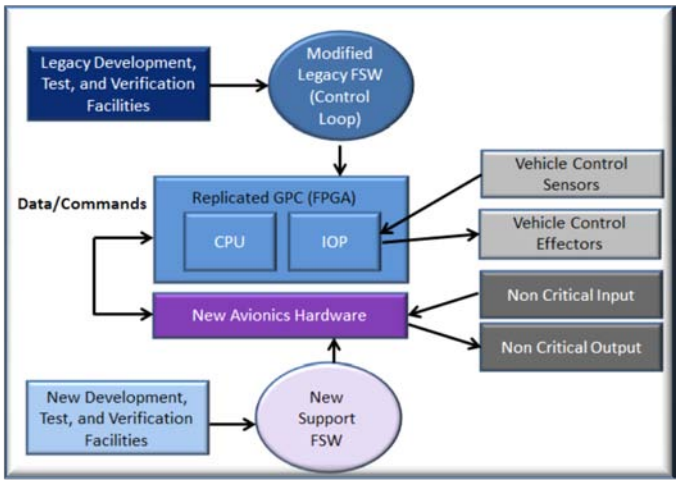

Figure 15: "Control Loop" in Replicated Avionics

6.4 Step 4 - Use of New Avionics Hardware with Embedded GNC Controller (PASS)

For this point in evolution, the design for the replicated GPC in a FPGA would remove the IOP functionality leaving only the CPU (**Figure 16**). The PASS would be further evolved where the System Software would remove all support for Input and Output, redundancy management, and other functionality not required to provide a platform that executes the GNC algorithms. Thus, it would retain a small set of functionality supporting such services as process, reduced time and event management. The PASS would become a compact solution with a small embedded RTOS. This small solution would contain the critical flight control algorithm on a chip which would be owned by the program

The new chip approach would have a shared memory interface where input can be placed into buffers to be consumed by the internal algorithms. Also, output could be acquired from these buffers. The chip could generate interrupts to indicate when to consume or provide data by the external system. Or the chip could operate as a slave to external signals or interrupts.

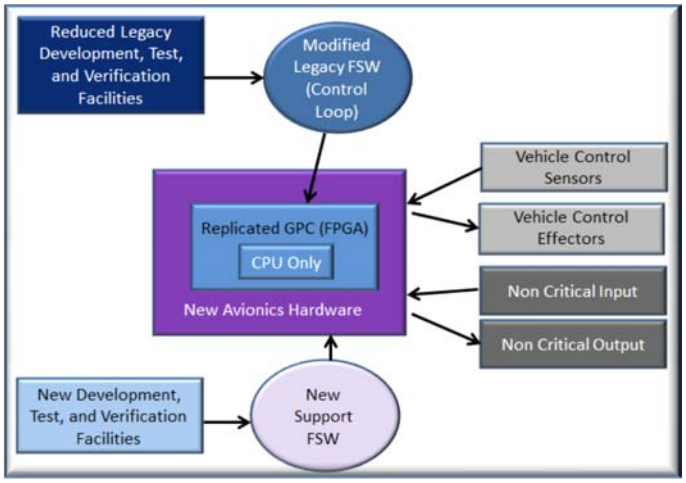

Figure 16: "Control Loop" as GNC Controller

The chip could be inserted into new avionics hardware boards using concepts such as PMC modules which allow custom electronics to be plugged into standard processor cards (**Figure 17**). The "control loop" would be isolated from the rest of the platform via a controlled shared memory interface. The new avionics platform would have to provide the redundancy scheme if required as the PASS now functions as a stand-alone GNC controller.

Figure 17: Example of PMC Solution

6.5 Step 5 - Use of New Avionics Hardware (No PASS)

At this point in evolution, there is a reduced version of the PASS that executes in an FPGA embedded in a modern avionics platform. The modern avionics platform has been matured where it supports the majority of the avionics functionality. The new development, test, and verification facilities are operational and have been maturing as well. It may be desired to phase out the PASS and any remnants of the replicated hardware. This task should be manageable as the scope of the work is isolated within an operational infrastructure.

The system could be evolved to various implementations. However, two such options are to (1) keep the "control loop" functionality in a FPGA or (2) add the "control loop" functionality with the support software. The first approach is still feasible because FPGAs are reprogrammable. Thus, a new version of the "control loop" could be supported by the existing FPGA infrastructure (**Figure 18**) and maintain the benefit of physical isolation (via controlled shared memory interface) from the rest of the system. The other option implements the functionality on the same platform as the existing support software (**Figure 19**). Both options require the update of the development, test, and verification facilities to support the control loop functionality.

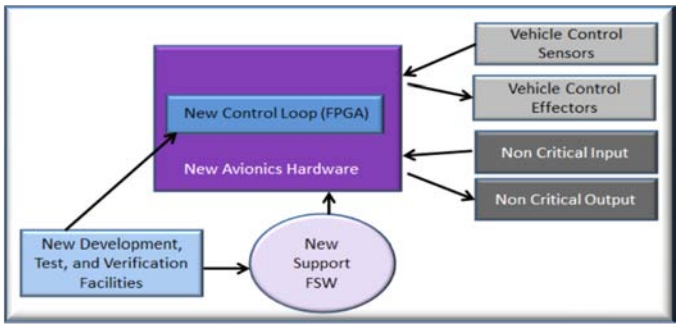

Figure 18: New Control Loop in FPGA

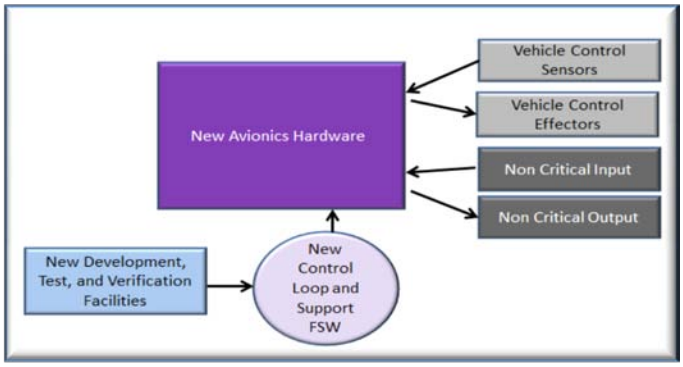

Figure 19: New Control Loop not in FPGA

7. BRIEF DISCUSSION

The plan to reuse Shuttle assets and the existing workforce is a prudent one to streamline the development of an SDLV. This lowers the initial development cost and schedule which is important under the current constraints of the environment. This provides the fastest and most affordable path to give the United States heavy lift capability. If this

reasoning is feasible for the entire launch vehicle, then it should also apply to the utilization of the existing Shuttle avionics and its flight software. Like the entire launch vehicle, this too can be evolved.

The resources and effort required to develop safety critical GNC flight software can be enormous. This requires the formulation, implementation, and verification of GNC algorithms that can control the vehicle. There is also the development of models and simulators to aid this effort. The Shuttle program has made an enormous investment over the years to develop and maintain product and facilities which have been very reliable and successful. The question is, why not make use of such an investment?

The approach to evolve the PASS towards a flight "control loop" is feasible as this is the most critical application of the legacy software. Once "pruned down", this version of the PASS should be less expensive to maintain over time until it is phased out. Skeptics argue that the PASS is written in a language that is not taught in school such as C or C++ and that there are no people out of school that can program in it. As technology progresses, every engineer has to learn to keep up. Thus, learning the language of the PASS is no different than learning the language of a new technology.

The flight "control loop" version of the PASS can execute on the legacy avionics hardware or in replicated versions of the legacy hardware. The non-flight "control loop" software could execute on other platforms. This division of effort provides the basis for initial savings and a path towards evolution. The legacy facilities already exist to support the reduction of the PASS today. There is no need for initial development or fabrication. These resources support the most risky and expensive portion of the vehicle flight software. The new technology path can start with the development of the non-flight "control loop" software. This lays the ground work to incorporate new technology and facilities over time until the legacy assets can be phased out. The PASS can evolve to a useful form that would be needed on the proposed SDLV. The evolution could be performed in phases as schedule and resources permit.

8. ACKOWLDEGEMENTS

The author would like to thank Paul Tice for his contributions to this paper.

REFERENCES

1. Hanaway, John F., Moorehead, Robert W., "Space Shuttle Avionics System", National Aeronautics and Space Administration Office of Management Scientific and Technical Information Division, Washington, DC, 1989, NASA-SP-504.
2. Baatzm E.B., "Out-of-this-World Software – The space shuttle's on-board-systems team proves that error-free programming may be tough to achieve, but isn't rocket science", CIO Magazine, February 15, 1995.
3. Fishman, Charles, "They Write the Right Stuff", Fast Company Magazine, December - January 1997, pp. 95 – 99, 104 – 106.
4. Ferguson, Roscoe, Tate, William, "Use of Programmable Gate Array Technology in Future Space Avionics", 24th Digital Avionics Systems Conference, 2005.
5. Ferguson, Roscoe, Thompson, Hiram, Smithgal, William, Tate, William, "Implementing Space Shuttle Data Processing Systems Concepts in Programmable Logic Devices", 2006 Military and Aerospace Programmable Logic Devices Conference (MAPLD), Paper 139.
6. Ferguson, Roscoe, "Replication of Space-Shuttle Computers in FPGAs and ASICs", NASA Tech Briefs, December 2008, (MSC-24141-1).