Protocol Controller Configurations for Self-checking Processor Pair for long orbital missions

Thejas T *, Ranjani K †, Manju C.R. ‡ and Jayalekshmy L§
Flight Computers Group, Vikram Sarabhai Space Centre Trivandrum, Kerala, India, 695022

Orbital space missions lasting for a long duration require a robust computing system that can withstand extreme environments, especially high levels of ionizing radiation. A self-checking pair based on an indigenous processor is proposed as the onboard computer to provide radiation tolerance. The 1553B bus is commonly used for communication in space systems. The protocol functions in the onboard computers are carried out by a separate protocol control chip. As the outputs from only one processor are routed to the bus, the processors in the pair can be configured in two ways – either both as bus controllers or one as bus controller and one as bus monitor. The 1553B protocol controllers in the processor pair were initialized as per this requirement and the complete software was burned into the program memory in a prototype board. Simulation runs with standard flight inputs proved that the final outputs produced in both the cases match exactly. Based on the results and further analysis, it was decided to opt for the BC-BM configuration.

I. Introduction

Long duration orbital missions have a high chance of being exposed to radiation during the orbital phase and it is possible that the electronic systems in the launch vehicle could be affected by single event effects (SEE) [1]. The various categories of SEE, such as single event upset (SEU), single event latch-up (SEL), single event functional interrupt (SEFI), multi-bit upsets (MBU) and others can corrupt the FPGA, RAM, and registers in microprocessors and other devices. Suitable radiation mitigation techniques need to be included in both the hardware and software of all sub-systems in such missions.

In typical launch vehicles, a reliable bus such as MIL-STD-1553B [2] is used for communication. It is a centralized command-response protocol, with an onboard computer (OBC) acting as the master or bus controller (BC) and the remaining systems functioning as remote terminals (RTs). Long-duration missions require a robust computing system that can work in extreme environments, especially high levels of ionizing radiation. To mitigate the effects of SEE, diverse techniques are adopted in orbital missions, including spatial/ temporal redundancy, use of parity and error-correcting

*Engineer, thejas_t@vssc.gov.in,
†Engineer, ranjani141@gmail.com
‡Division Head, Flight Software and integration Divison, cr_manju@vssc.gov.in
§Group Director,Flight Computers Group, l_jayalekshmy@vssc.gov.in

codes, periodic scrubbing, reset/ power-cycling and circuit-level protections.

To combat SEE, a self-checking pair of processors is generally used in various reusable rockets and orbital systems, thus providing spatial and temporal redundancy. A supervisory system, generally an FPGA, compares the outputs of the processors in the pair to ensure the correctness of data. A self-checking pair based on an indigenous processor is proposed as the onboard computer to provide radiation tolerance. In this system, both the microprocessors execute the embedded software independently in a time-synced manner. Externally, the two processors in the pair function as a single system and output only a single set of data, aided by the FPGA.

The 1553B protocol supports only a single bus controller or master on each bus. The protocol functions in launch vehicle OBCs are carried out by a separate protocol control chip (PCC), like the Advanced Communication Engine (ACE) from DDC [3]. In the case of the self-checking pair, each processor in the pair contains such a chip, but the outputs from only one chip are routed to the bus by the FPGA. Therefore, the processors in the pair can be configured in two ways – either both as bus controllers (BC) or one as bus controller and one as bus monitor (BM). The BM can monitor all transactions on the 1553B bus and select the input data required for its computations.

In this paper, the suitability of the above configurations is assessed. Towards this, a prototype board with two 16-bit processor cores and an FPGA was used for evaluation. The onboard software for both these configurations was developed and programmed in the processors. Simulation runs were taken with the flight profile and initialization data of an actual launch vehicle mission. The output data from both the processors in the two configurations was extracted and compared. Based on the results and analysis, it was decided to opt for the BC-BM configuration for our requirement. The details of both the designs, including the configuration of the protocol controllers, software development and analysis of output data, are presented.

The organization of this paper is as follows: the related literature in the field of self-checking processor pair is explored in Section II. In Section III, the essential background on the hardware board architecture, 1553B protocol controller and the launch vehicle software is presented. The combinations proposed for the protocol controllers in the two processors of a self-checking pair are explained in Section IV. The testing carried out, along with the results, constitute Section V and the conclusions are listed in Section VI.

II. Related Work

In [4], the authors present a survey of fault-tolerant platforms, emphasizing the importance of lock-step dual processor architectures in automotive safety-critical applications, where synchronized processors detect faults through comparator circuits. Other configurations like loosely-synchronized dual processors and Triple Modular Redundancy (TMR) are also discussed, highlighting the trade-offs in performance and fault tolerance. Floridia et. al. [5] propose an on-line self-test mechanism for dual-core lockstep System-on-Chips (SoCs), focusing on efficient fault coverage for comparators through a hybrid architecture. In [6], Phil Koopman discusses fault tolerant systems, focusing on dual self

checking pairs which consist of two redundant systems, each with two cores that are used to check for inconsistencies. The two cores acting as a self checking pair help in preventing errors and the redundant system helps ensure that there are no outages.

The author in [7] describes tandem computers which comprise multiple processors, memories and disks connected by dual bus system and are independent. Processes are replicated and done in primary and backup processors such that the backup processor takes over in case of failure in primary. Another architecture that is explained is Stratus, which uses two cores in each processor board which are driven in lock-step, both cores running the same code and finally the outputs are compared. If any inconsistency is detected the board stops and it also carries out diagnostics. There are two processor boards running the same functions so that when one board is in repair, the other works without interruption.

Michael Holguin et.al.[8] elaborate on an emergency detection system (EDS) for commercial crew launches which notifies the crew when there is a degrading condition and issues an abort signal. The EDS architecture contains two processing units which work in parallel with self checking pairs of processors. Klecka et.al. [9] patented a processing unit with two processors with one master processor and a shadow processor which is used only for comparison. If there is any inconsistency, the master processor will determine the cause, save the state and if recovery is possible, it will reset and reinitialize to the saved state in both the cores. The work in [10] presents a data processing system with two cores in lock-step mode. The second core is designed to shadow the first processor so that faults are detected by comparison.

Dual core processors in self-checking pairs have been discussed in existing literature but their configuration has not been studied in detail. In particular, the mode of communication of the two cores with the system bus is not described in the above related work. The main focus of our work is the mode of communication of the two processors in the self-checking pair on the external bus, for transmission and reception of data. This is done by appropriate configuration of the protocol controllers.

III. BACKGROUND

A. Prototype Board Architecture

The prototype board consists of two indigenous 16-bit processors (P1 and P2) and associated peripherals. The processor consists of the Arithmetic and Logic Unit (ALU), the control logic and the control memory. The maximum frequency of operation is 75 MHz. It is a 16-bit floating-point processor with high-level language (Ada) support. The processor can address 232K words and the memory capacity is subdivided into 200K words of EEPROM and 32K of RAM. The flight application code resides in EEPROM whereas data is stored in SRAM. The total SRAM area is divided into local SRAM for individual processors and a shared RAM area for data exchange between processors. Eight interrupts and five memory mapped I/O ports are also available. The processor is interfaced to the MIL-STD-1553B bus through the protocol controller core logic embedded in the FPGA. The basic architecture of the board is shown in Fig 1

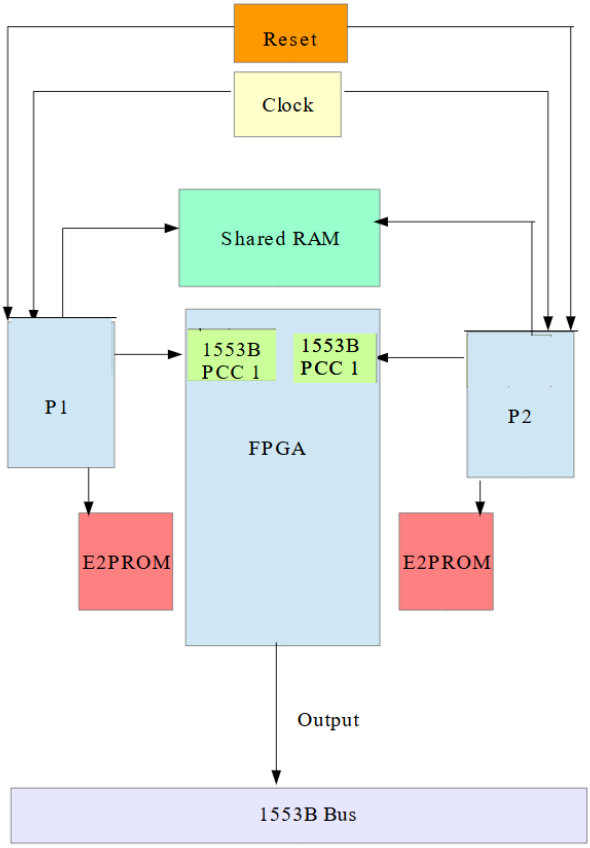

Fig. 1 Architecture of prototype board

B. 1553B Protocol Controller

Generally, a 1553B protocol controller[3] provides the interfacing between a microprocessor and a 1553B bus. It supports Bus Controller (BC), Remote Terminal (RT) and Bus Monitor (BM) modes of operation. It contains internal address latches and data buffers that can transmit and receive so as to interface with the bus. The protocol controller implements three types of bus monitor modes: a word monitor, a selective message monitor, and a combined RT/selective monitor. It has a memory mapped software interface to the host processor, with 17 internal registers. The interrupt mask register is used to enable and disable interrupt requests for various conditions. Configuration registers are used to select the mode of operation to BC/BM/RT. The start register is provided for command type functions, such as software reset.

The memory organization of the Protocol Controller Chip (PCC) is such that messages are stored in message blocks for BC/BM modes while data blocks store data words in RT mode. For each message block there is a 4 word descriptor. The first word, block status word, is written by the controller's protocol logic at the beginning and end of each message. The block status word contains bits relating to message status, completion, validity etc. and is the first level of check carried out on a 1553B message. The second word is the time tag word, which indicates the start and end times of each

message. The third word is the inter-message gap time and the fourth word is the message block address for BC. In BC mode, the message block contains command word, data words and status word. Each message is initiated by the command word and the response of the RT is through the status word. This is checked to ensure a valid acknowledgment from the RT.

The format of the block status word and status word is shown in Fig 2 and Fig 3, respectively.

BIT	DESCRIPTION
15(MSB)	REMOTE TERMINAL ADDRESS BIT 4
14	REMOTE TERMINAL ADDRESS BIT 3
13	REMOTE TERMINAL ADDRESS BIT 2
12	REMOTE TERMINAL ADDRESS BIT 1
11	REMOTE TERMINAL ADDRESS BIT 0
10	MESSAGE ERROR
9	INSTRUMENTATION
8	SERVICE REQUEST
7	RESERVED
6	RESERVED
5	RESERVED
4	BROADCAST COMMAND RECEIVED
3	BUSY
2	SUBSYSTEM FLAG
1	DYNAMIC BUS CONTROL ACCEPTANCE
0(LSB)	TERMINAL FLAG

Fig. 2 Status Word

BIT	DESCRIPTION
15(MSB)	EOM
14	SOM
13	CHANNEL B/$\overline{A}$
12	ERROR FLAG
11	STATUS SET
10	FORMAT ERROR
9	NO RESPONSE TIMEOUT
8	LOOP TEST FAIL
7	MASKED STATUS SET
6	RETRY COUNT 1
5	RETRY COUNT 0
4	GOOD DATA BLOCK TRANSFER
3	WRONG STATUS ADDRESS/NO GAP
2	WORD COUNT ERROR
1	INCORRECT SYNC TYPE
0(LSB)	INVALID WORD

Fig. 3 Block Status Word

C. Launch Vehicle Software

The launch vehicle software consists of a real-time operating system (RTOS) and other software components with specific functions, referred to as application software. RTOS software is the system software that manages the resources available in the processor and ensures the health of the processor and its peripherals through periodic health checks. It schedules the application tasks such as navigation, guidance, sequencing and control software as per the requirement. It provides necessary communication interface with each RT periodically at defined time slots. In the Indian launch vehicle context, each of the application software components is developed by separate design teams. These are then

integrated with the RTOS and the complete software is assembled and linked with the user library to generate the final executable files.

IV. Proposed Configurations

As described in Section I, the self checking pair works by executing the same code in both the processors so that the outputs can be compared to check for inconsistencies. The two processors in the pair are referred to as P1 and P2 in this paper. In this section, the details of the software in the two processors for both the proposed configurations are explained.

A. Bus Controller – Bus Controller (BC-BC)Configuration

Each of the two processors in the prototype board contains a 1553B Protocol Controller Chip (PCC).In the first option, hereafter referred to as the BC-BC configuration, the PCCs in both processors are configured as bus controllers. The software programmed in P1 and P2 is exactly same in this configuration. The output data from only P1 are routed to the 1553B bus. In order to monitor the health and outputs from P2, the essential parameters are written to the shared RAM area by P2. This data is read by P1 and posted in telemetry for analysis and monitoring.

The timeline for the initializations carried out in the onboard computer for a benchmark launch vehicle mission is depicted in Fig 4. In both the processors, subsequent to power-on/ reset, the PCCs are initialized to bus controller mode at the start. This is done by issuing a soft reset and then setting the various configuration registers to the values corresponding to BC mode of operation. Then all the stack memory locations in the PCC RAM are cleared. After this initialization, message blocks are created for the 1553B communication with the RTs as per the timing schedule for the mission.

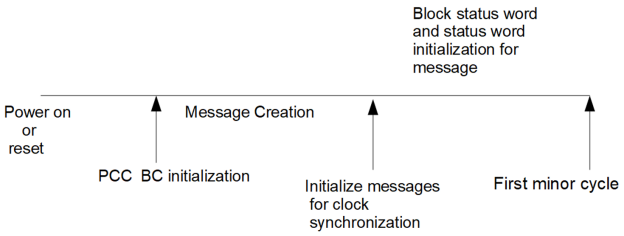

Fig. 4 Flight Initialization time-line for BC-BC configuration

The 1553B messages for clock synchronization are then initiated by the BC so that all the processor-based systems in the avionics configuration begin their first frames in a synchronized manner. Further to this, initialization of block status words and status words for all the messages is carried out. A trigger for start of BC mode of operation is given to PCC by writing a pattern to a configuration register. This sets in motion the sequence of messages for the RTs on the bus. After all the initializations are completed, the first frame of real-time execution is commenced. The real-time interrupt is generated by the hardware timer in the processor on completion of every minor frame (typically 20 ms). As

part of the timer interrupt service routine in every 20 ms, the PCC is re-initialized in bus controller mode, the soft reset is issued for the start of a fresh cycle and the BC start trigger is generated. The sequence of initializations to be done for bus controller function in every frame is shown in Fig 5.

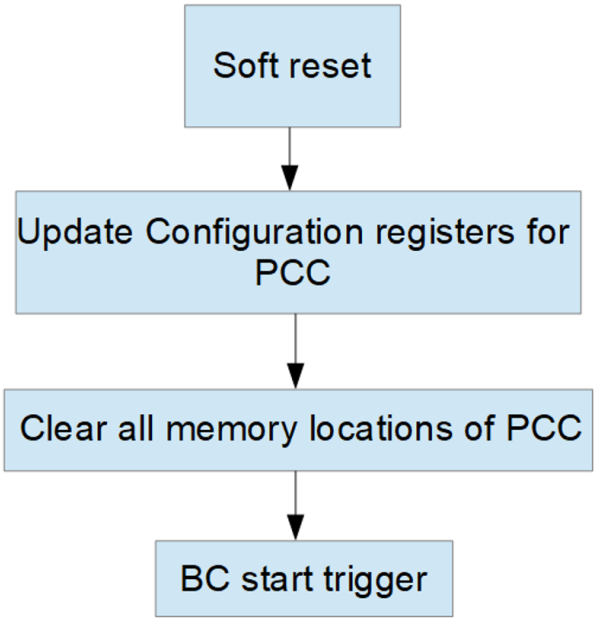

Fig. 5 Flowchart of Bus Controller initialization

B. Bus Controller- Bus Monitor (BC-BM) Configuration

In the second option, hereafter referred to as the BC-BM configuration, the PCC in P1 is configured as a bus controller and P2 operates as a bus monitor (BM). The bus monitor does not initiate any messages; it monitors all transactions on the bus and captures the required input data. These data are used for the computations in P2. The outputs generated by P2 are compared with those of P1 operating as BC, thus completing the self-checking pair. The software programmed in P1 remains the same as in Option1, as it is functioning as a bus controller in this case as well. On the other hand, P2 is to be configured to work as a bus monitor (BM) calling for changes in the initialization process and memory map.

The configuration for bus monitor operation is done by issuing a soft reset and then setting the various configuration registers to the values corresponding to BM mode. Then all the stack memory locations in the PCC RAM are cleared. After this initialization, the data stack pointer and command stack pointer are set to the beginning of the respective stacks. A lookup table is used as the reference for deciding which messages on the bus need to be monitored. In our experiment, the lookup table contents are set in such a way as to monitor all the messages on the bus. All the message buffers are then cleared. A trigger for start of BM mode of operation is given to PCC by writing a pattern to a configuration register. The BM then starts capturing the data traffic on the bus.

As part of the timer interrupt service routine every 20 ms, the PCC is re-initialized in bus monitor mode, the soft reset is issued for the start of a fresh cycle and the BM start trigger is generated. The sequence of initializations to be done for bus monitor function in every frame is shown in Fig.6.The memory map in PCC is different for the bus monitor and bus controller configurations. Therefore, changes are made in the software for P2 so that the processor reads the messages from the correct location as per the BM memory map. The code required for filling the data words for BC-RT/ broadcast messages is removed as it is irrelevant in this case. As in the BC-BC case, health parameters and functional outputs from P2 are obtained in telemetry through the shared RAM.

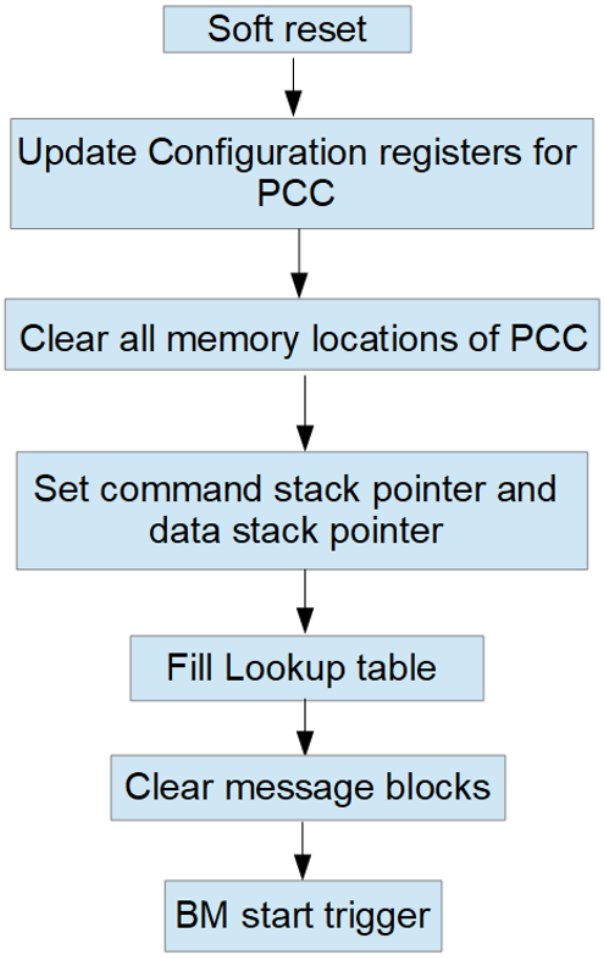

Fig. 6 Flowchart for Bus Monitor initialization

C. Testing and Results

To compare the two possible configurations and to analyze the performance, simulation tests were carried out with the benchmark flight software of a typical launch vehicle mission. The real time OS in both cases was integrated with all the application software. The corresponding binaries were programmed in the EEPROM of P1 and P2 on the prototype

board. The system was then tested using a stand-alone test-bed where the messages from the different RTs are simulated. The final, critical outputs from the integrated flight software are the sequencing commands and control commands to be posted to actuators. These outputs from both P1 and P2 were recorded and checked to confirm their correctness and one-on-one matching.

It was seen that no error flags are set in the runs with both the configurations. The sequencing and control commands generated by both processors showed an exact match in every frame in both BC-BC and BC-BM combinations. The sequencing commands are listed in Table 1 and the three control outputs are shown in Fig 7. As observed from the table and the plots, the control command outputs available as 16 bit integer values from both p1 and p2 processors were exactly matching to the least significant bits. This demonstrated that the bus controller – bus controller combination as well as the bus controller – bus monitor combination is fully capable of working as a self-checking pair.

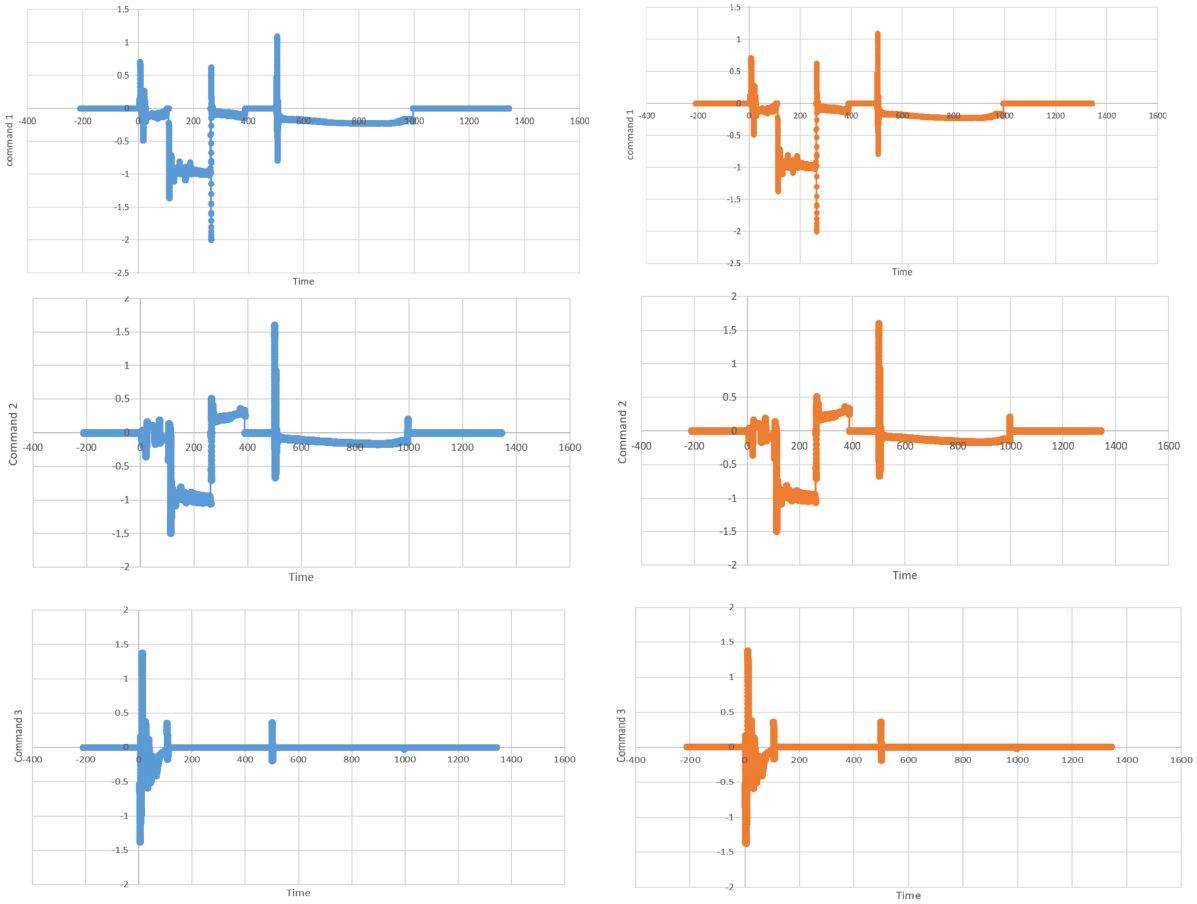

Fig. 7 Control outputs: BC-BC configuration (Blue) and BC-BM configuration (Orange)

To arrive at a final configuration for the long duration missions, the overall software architecture for the two options was compared (Table 2). In BC-BC configuration both processors attempt to initiate the transmission of messages on the bus by issuing the trigger for start of BC operation. In this case, P2 receives the required data from the bus, but this

data corresponds to the response from the RTs to the commands from P1. This may be considered a protocol violation in the 1553B standard. There is also a possibility that any error in P2, such as non-initiation of commands, could be missed, as P2 continues to receive messages. These are two disadvantages in the BC-BC system.

Sequencing Command	**Time of Issue (s) (BC-BC)**	**Time of Issue (s) (BC-BM)**
Reset	0.0	0.0
Vehicle lift-off	0.02	0.02
Stage 1 boosters separated	0.42	0.42
Stage 2 ignition	101.58	101.58
Stage 2 separation and stage 3 ignition	104.86	104.86
Start of coasting phase	259.20	259.20
Stage 3 separation and stage 4 ignition	383.94	383.94
Satellite separation	488.94	488.94
Orbit change using thrusters	994.26	994.26

Table 1 Sequencing outputs: BC-BC and BC-BM configurations

On the other hand, in the BC-BM option, only P1 initiates the transmission of messages on the bus. P2 acts a silent observer and recipient and then uses the data for the execution of tasks. One minor drawback is the extent of software changes between the two processors in the pair. Nevertheless, the BC-BM option is superior, as it does not violate the basic command-response philosophy of the 1553B protocol in any way. Also, there is no possibility of an error in either of the processors going undetected. Considering these two factors, it was decided to adopt the second option, the BC-BM configuration, for our system.

Feature	**BC-BC**	**BC-BM**
Message initiation	Both P1 and P2 initiate messages through BC start trigger and commands	Only P1 initiates messages through BC start trigger and commands
Data reception	P2 receives data corresponding to the response for commands from P1	P2 monitors all transactions on the bus and captures the data
Software similarity	Same software in P1 and P2	Software is different in P1 and P2

Table 2 Comparison of BC-BC and BC-BM Configurations

V. Conclusion

As the number and frequency of missions to space continue to rise rapidly, the use of sustainable, reusable rockets and modules becomes significant. This calls for implementation of mitigation measures to combat radiation, as the avionics systems are particularly susceptible to ionizing radiation in orbit. The idea of a self-checking pair of processors is widely adopted in various space vehicles and satellites, as an effective means to tolerate the adverse effects of radiation.

Normally, both the processors in a self-checking pair execute the same software with the same set of inputs, in step with each other and generate outputs. The outputs are matched by a supervisory system before posting to actuators. In this work, two different configurations were studied for the processors in the self checking pair -Bus Controller- Bus Controller and Bus Controller- Bus Monitor. The 1553B protocol controllers in the processor pair were initialized as per this requirement and the complete software was burned into the EEPROM in a prototype board.Simulation runs with standard flight inputs proved that the final outputs produced in both the cases match exactly.

The BC-BM combination is the preferred option for our application as it follows the 1553B command-response protocol to the digit and minimizes the likelihood of undetected errors. Prior to the actual mission, the possible optimizations in the code are to be explored. In addition, multiple tests and simulations are to be executed at stand-alone, sub-system and system levels to ensure the robustness of this new configuration.

References

[1] Mezger, B. W., "Single Event Effects," , 2024. URL `https://seds.nl/files/Single_Event_Effect_Benjamin_Mezger_PGMICRO.pdf`.

[2] *MIL-STD-1553 Designer's Guide Sixth Edition*, Data Device Corporation (DDC), 1998.

[3] *ACE/Mini-ACE Series BC/RT/MT Advanced Communication Engine Integrated 1553 Terminal User's Guide*, Data Device Corporation (DDC), 2010.

[4] Baleani, M., Ferrari, A., Mangeruca, L., Sangiovanni-Vincentelli, A., Peri, M., and Pezzini, S., "Fault-tolerant platforms for automotive safety-critical applications," *Proceedings of the 2003 International Conference on Compilers, Architecture and Synthesis for Embedded Systems*, Association for Computing Machinery, New York, NY, USA, 2003, p. 170–177. https://doi.org/10.1145/951710.951734, URL https://doi.org/10.1145/951710.951734.

[5] Floridia, A., and Sanchez, E., "On-line self-test mechanism for Dual-Core Lockstep System-on-Chips," *Microelectronics Reliability*, Vol. 112, 2020, p. 113770. https://doi.org/https://doi.org/10.1016/j.microrel.2020.113770.

[6] Koopman, P., "Better Embedded System SW: Self-Monitoring and Single Points of Failure," , 2014. URL https://betterembsw.blogspot.com/2014/04/self-monitoring-and-single-points-of.html.

[7] Nørvag, K., "An Introduction to Fault-Tolerant Systems",," *IDI Technical Report 6/99*, Vol. ISSN 0802-6394, July 2000.

[8] Michael Holguin, G. H., and Mingee, R. (eds.), *Commercial Crew Launch Emergency Detection System The Key Technology for Human Rating EELV*, AIAA SPACE 2010 Conference Exposition, 2010. https://doi.org/https://doi.org/10.2514/6.2010-8670.

[9] Klecka, J. S., Bruckert, W. F., and Jardine, R. L., "Error self-checking and recovery using lock-step processor pair architecture," , 1998. URL https://patents.google.com/patent/US6393582B1.

[10] Refaeli, J., Amedeo, N. H.-C., and Woodrum, L. A., "DATA PROCESSING SYSTEM HAVING LOCKSTEP OPERATION," , 2019. URL https://data.epo.org/publication-server/rest/v1.0/publication-dates/20190605/patents/EP3493062NWA2/document.pdf.