FEDERAL UNIVERSITY OF SANTA CATARINA
TECHNOLOGICAL CENTER
ELECTRICAL AND ELECTRONICS ENGINEERING DEPARTMENT

DEVELOPMENT OF A RADIATION EXPERIMENT PAYLOAD FOR EVALUATING SDRAM MEMORIES TECHNOLOGIES IN HARSH ENVIRONMENTS

Final report for the long-term internship settled at the Laboratory of Computer Science, Robotics and Microelectronics of Montpellier

André Martins Pio de Mattos

Academic Advisor: Prof. Eduardo Augusto Bezerra, Ph.D.
Laboratory Supervisor: Prof. Luigi Dilillo, Ph.D.

Montpellier, 2021

FEDERAL UNIVERSITY OF SANTA CATARINA

LABORATORY OF COMPUTER SCIENCE, ROBOTICS AND MICROELECTRONICS OF MONTPELLIER

UNIVERSITY OF MONTPELLIER

Long-term internship completed at the Laboratory of Computer Science, Robotics and Microelectronics of Montpellier during the period of 24/02/2020 to 24/07/2020

André Martins Pio de Mattos

Prof. Eduardo Augusto Bezerra, Ph.D.

Prof. Luigi Dilillo, Ph.D.

Montpellier, 2021

Contents

1 Introduction

The increasingly market of small spacecrafts [1] is leading a rapid evolution of the technology and standards employed in these missions and pushing the limits of size, power consumption, reliability and capabilities of the embedded systems. The CubeSat, a satellite standard, and its variants are emerging as promising models for several applications in remote sensing, telecommunications, earth observation and even space exploration.

Comparing with the other standards and bulky satellites, which use high-end and tailored components and require considerable effort in development and launch, the CubeSats have a advantage due to the use of Commercial Of-The-Shelf (COTS) components and the reduced logistics involved that impacts directly in the mission cost.

Despite this advantage, the CubeSat missions do not reach all the space sector applications, due to the size scope or even a technological impracticability, reinforcing that the bulky missions still have relevance and importance in the subject. Therefore, this scenario brings a new horizon for technological advances and facilitate the widespread usage of satellites to solve challenging problems and enhance the understanding of the space.

The environment faced in these space missions is harsh due to the high temperature, pressure, radiation doses and mechanical stress conditions. These parameters achieve a critically value when considering long-term missions or high orbits. The majority of CubeSat missions follow low orbits profiles, where the conditions are less demanding, nevertheless as dangerous and onerous for development. Also, the utilization of COTS components without caution and previous analysis could lead to catastrophic failures as consequence of the creation of bottlenecks and unpredictability in the system. For this reason, several studies are being carried out to identify and quantify the vulnerabilities of these components in the space environment.

The most challenging condition to overcome is the significant amount of radiation in which these systems are exposed outside the protection of the earth magnetic field. Since the deployment from the launcher, the satellite is exposed to a wide spectrum of random radiation interactions with diverse magnitudes and consequences [2]. In order to minimize these effects, different approaches are adopted in various layers, since the hardware architecture itself to complex software correction error techniques and redundancies. Then, evaluate the vulnerability of those components due to the radiation exposure and characterize the failure threshold for these parameters are essential to determine the effort that should be made during the development. The Integrated Chip (IC) memories, in a wide variety of technologies and manufacturing nodes, are one of the most critical components affected by the high doses of radiation [3]. The high capacity of these devices, the crucial information stored and the strict requirements imposed make unfeasible, in several applications, the high-level efforts to minimize this damage due to the high performance trade-offs or even the substantial amounts of data. For this reason, other approaches are investigated, which generally leads for high-cost custom hardened devices for specific niches. Therefore, it is essential to evaluate the real venerability of these components and determine when radiation mitigation techniques are required in each application.

In this context, two types of memory could be enumerated: non-volatile and volatile. The first group offers technologies with significant robustness within radiation exposure conditions without technological enhancements, such as Flash based chips. Also, due to less performance intensive requirements, some software management techniques are feasible. However, in the volatile memories, with the pressure to attend the performance

objectives of demanding applications and the current available technologies, this group is under analysis of many studies due to its vulnerability. The most performance capable model are the Synchronous Dynamic Random-Access Memory (SDRAM) chips, which are being improved and tailored for decades in the personal computers, workstations and servers industry.

Therefore, the efforts of this work will focus on the creation of a payload capable of evaluate the radiation effects on three SDRAM memories with different manufacturing nodes. This payload will test these chips in the real harsh space environment by participating of the GOLDS-UFSC CubeSat mission. These particular SDRAM memories were previous characterized on laboratory experiments, then by exposing them to the real environments and executing the same tests routines will not only generate more results for analysis, but also provide an opportunity to assess the test methodologies themselves. Also, after collecting sufficient data to be analysed, this payload could be used to provide a meaningful health status, concerning the radiation doses which the satellite were exposed, to the entire satellite subsystems and further missions.

In order to accomplish these objectives, the payload is designed to follow the GOLDS-UFSC OBDH DaughterBoard standard [4] [5], which defines the connectors, shape and size of the board. This standard allows the utilization of the module throughout future FloripaSat [6] core[1] missions in reason of its low space occupation inside the CubeSat, being considered further as an expansion module instead of a payload experiment. Also, due to the mission limited power budget, the developed board should consider reduce power consumption and define clever power management strategies. In addition, methods for anti latch-up, a type of short circuit which can occur inside an IC, are considered in the design. Therefore, combining all these requirements, the payload architecture consists of the following modules: a control and management subsystem, operated by a System-On-a-Chip (SoC) solution Field-Programmable Gate Array (FPGA) [7]; power converters for properly voltage level supply; anti latch-up circuitry; communication and interface buses; debug module; and the SDRAM memory chips.

[1]The GOLDS-UFSC mission reuses the FloripaSat-I core modules.

2 Background

2.1 CubeSat Standard

This work proposes a scientific experiment payload for CubeSats [8], a nanosatellite [1] standard. This standard of satellites consists of specific criteria for its shape, size, and weight, which help reduce costs due to the higher feasibility for mass-production and availability of off-the-shelf parts. Besides these benefits, the standardization reduces the development time and ease the access of transportation and deployment into space.

Essentially, CubeSats are satellites composed by small cube units, which might be combined to form larger spacecrafts and to attend the mission requirements and goals. This "unit" is referred to as a 1U, then a 1U CubeSat is a 10 cm cube with a mass of approximately 1 to 1.33 kg. Figure 1 shows an illustrative diagram of the internal architecture of a 2U CubeSat, containing the core modules (power management, data handling and communication), payloads[2] (primary and secondary experiments), solar panels, antennas and others auxiliary modules.

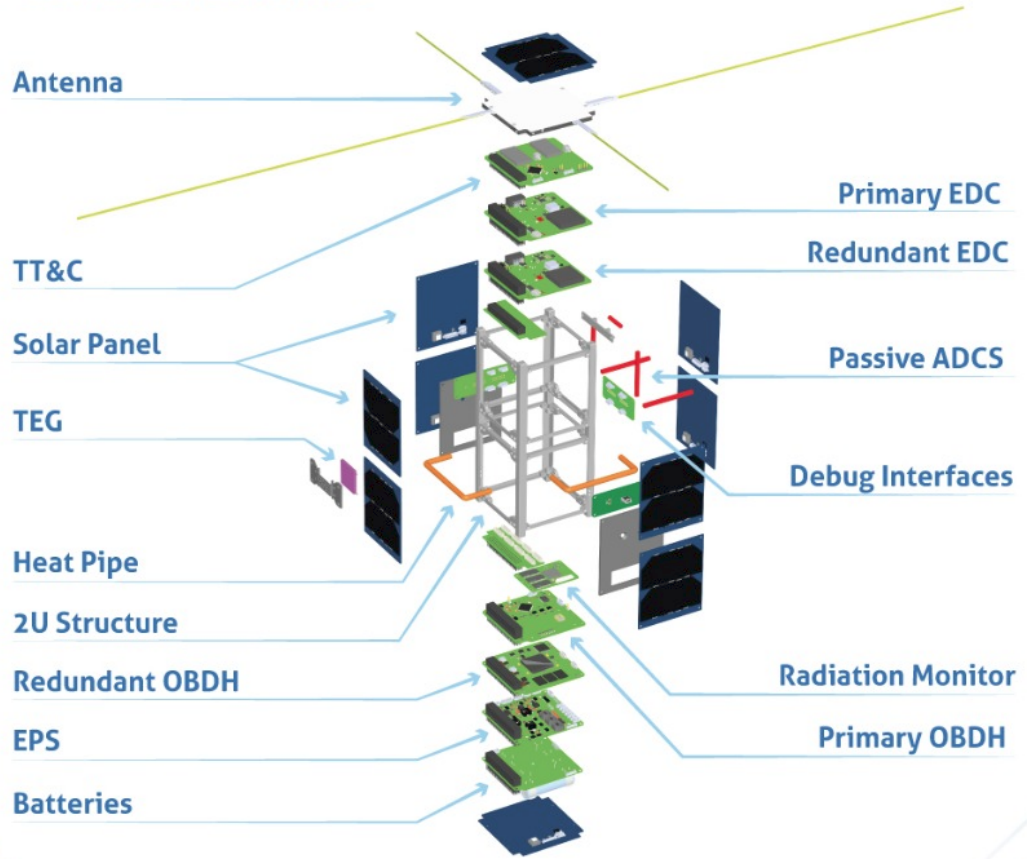


Figure 1: Conceptual exploded view of a CubeSat (GOLDS-UFSC).

[2]The "Radiation Monitor" illustrated in the diagram is the payload proposed in this work. It also presents its preliminary location inside the satellite

2.2 Radiation Effects

The space is full of engineering challenges related to the harsh environment conditions, including: mechanical stress, high temperature variations and severe radiation doses. This work focus on the last aspect, analysing its influence in electronic devices and evaluating the temporary and cumulative damaging phenomena. These effects are characterized by the Single Event Effects (SEEs), temporary, and the Total Ionizing Dose (TID) and Displacement Damage (DD), cumulative. The sensitive circuit nodes are affected or permanently damaged with the penetration of ionizing particles, which are representative outside the Earth's magnetic field protection.

The interaction with ionizing particles can induce the occurrence of instantaneous damaging events, which are characterized by the single event effects in various categories depending on the errors produced and the device components affected: Single-bit Upset (SBU), soft-error; Multiple-Bit Upset (MBU), soft-error; Multiple-Cell Upset (MCU), soft-error; Single Event Transient (SET), soft-error; Single Event Functional Interruption (SEFI), soft-error; Single Event Latch-up (SEL), hard/soft-error; Single Event Gate Rupture (SEGR), hard-error; and Single Event Burnout (SEB), hard-error. These effects might lead to soft-errors, errors that are reversible since can be mitigated by power cycles, resets, or other erase operations. In other cases, the effects induce hard-errors, leading to permanent device failure.

Both SBUs and MBUs are transient events that causes bit-flips (upsets) within a memory cell or latch, differing in the quantity of elements affected. Similarly, the MCUs are caused by the generation of two or more bit-flips within a cell array. The SETs are characterized by the collection of free carriers generated that induce a current or voltage transient that may lead to an error. The SEFIs are due to a perturbation in control signals, causing a loss of functionality and entering in a undefined state, a test mode, or a halt. The SELs occurs when a parasitic thyristor is triggered in the device structure, which generates high-current consumption and may lead to a thermal overheat in the affected region. The SEGRs occurs when a particle hit breaks the gate dielectric of a MOS transistor and the SEBs when a strike causes a destructive burnout due to high-currents.

The cumulative effects depends on the continuous exposure to radiation conditions that degrades the function of the device until its operation is definitely compromised. The TID is a long-term failure mechanism that causes shift in the transistors threshold voltage, increase of leakage currents (power consumption), and also the degradation of the gain in bipolar devices. The DD effect is the result of collisions between high-energy protons and the device that, when a minimum required energy is achieved, creates deformities in the lattice by removing the hit atom which results in different local electrical characteristics. The accumulation of this effect with energy above the displacement threshold can create a large fault cluster. For instance, it causes the reduction in bipolar transistor gain, reduction of the efficiency in solar cells, light emitting diodes and photodetectors, charge transfer inefficiency in charge coupled devices and resolution degradation in solid-state detectors.

2.3 SDRAM Memories

Synchronous dynamic random-access memories (SDRAM) are dynamic random-access memories (DRAM) whose operation is coordinated by an external supplied clock signal. The DRAM devices are a type of random access semiconductor memory that stores each

bit of data in a cell structure based on a capacitor and a transistor. The capacitor retains or releases the energy during charge and discharge events controlled through the transistor. These two states, charged and discharged, represent the two values of a bit. However, due to parasitic coupling loads, the capacitor slowly leaks off, which leads to data loss after a certain period. Then, to prevent this, the device requires an external controller to perform periodic refresh cycles that rewrites the data in the capacitors, restoring them to their original initial charge.

The DRAM memories have the disadvantage of direct influence of input signals in the internal functions since each combination of control signals determine the execution of one memory operation (e.g, read, write, and refresh). Using a synchronous interface, it is possible to use pipeline techniques that improve the memory performance in comparison with asynchronous interfaces at the same frequency of operation. Pipelining means that the device can handle new commands before the termination a previous operation processing. Then, write and read commands can be followed by other commands after a fixed number of clock cycles (latency). Also, to improve even more the access operation speed, the memory is divided into equally sized sections called banks, which allows independent operations occurring simultaneously. This architecture enables SDRAMs to achieve greater concurrency and higher data throughput than the asynchronous memories.

These devices are susceptible to harsh radiation environments and many studies are carried out to evaluate their behavior in different conditions using ground-based experiments [9]. This work, besides proposing a comparative study between different SDRAM chip generations, creates an experiment payload for in-orbit tests that might improve the ground-based evaluation methodology depending on the acquired results.

2.4 FPGA Devices

A Field-Programmable Gate Array (FPGA) is an integrated circuit designed to be reconfigurable after manufacturing, differing from its counterparts that are purpose specific and unchangeable. Then, it can be configured by the developer to perform almost any digital operations in various levels of complexity using a Hardware Description Language (HDL) and automation tools to translate the high-level commands into the physical structure combinations. The device contains an array of programmable logic blocks and a hierarchy of reconfigurable interconnects that enables the blocks to be combined for creating complex structures or merely simple logic gates (e.g., AND, OR, XOR). Also, in the majority of the devices, these blocks include memory elements that might be from simple flip-flops to more complete blocks of memory.

The FPGAs core structure can be combined with other technologies and structures to form even more flexible and ready to use devices. This mixed chips are generally referred to as System-On-a-Chip (SoC) devices [7], which internal architecture combine several elements to attend the different elements required by an application in an unique and compact solution. For instance, in this work is used a SoC solution that includes a FPGA array, an ARM processor, volatile and non-volatile memories, communication interfaces, clock sources and conditioners, independent memory controllers, and a bus interface. In subsection 3.2 the device and its subsystems are described in more depth.

Besides the architectural aspect of FGPA devices, the technology employed is important to determine its characteristics under radiation conditions, the environment analysed in this work. Despite the chosen chip not being a rad-hard device, it presents robust attributes in comparison with regular systems due to its Flash based implementation [3].

2.5 PCB Principles

The Printed Circuit Boards (PCB) are a combination of interspersed layers of conductive and non-conductive materials to form electrical circuits. These layers and electrical circuits are connected using structures called ”vias” and ”tracks”, respectively. The electronic components are attached, electrically and mechanically, in the external layers through a soldering process in their corresponding ”pads” (i.e., the designated place). The Figure 2 presents a simplified conceptual diagram of a multi-layer PCB.

Using electronic Computer-Aided Design (CAD) tools, the PCBs are deigned for production in specialized companies that use automated machinery from the board manufacturing and assembly to electrical testing. Due to this production chain, the PCBs design can incorporate high precision layouts, which allowed their evolution for complex circuits and structures. However, this progress exploiting the limit of the physical implementation transformed the development of these boards a challenging task. Some unusual and counter intuitive effects start to change the expected characteristics of the circuits, which might lead to non functional designs. For instance, when working with high-speed [3] signals, their integrity might be compromised by mismatching in length, impedance, reflections, crosstalking, and return paths.

In order to avoid these phenomena, the PCB designers use mitigation techniques and follow validated standards. In this work, the developed board required some measures to guarantee the proper operation and meet the requirements, since it is designed for space applications [10]. These techniques and patters are described in further sections and the proper justifications provided.

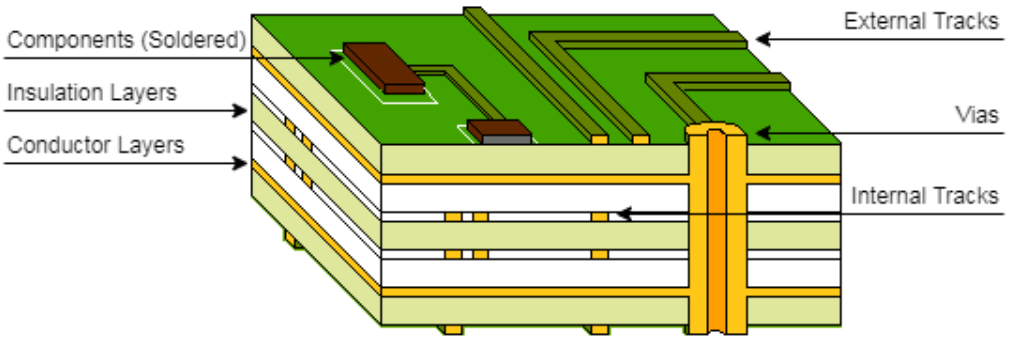


Figure 2: Conceptual PCB board.

[3] Signals with short periods of time during edge-to-edge changes, which generally appears in higher frequencies.

3 Development

The following topics will cover the entire payload development process in details, focusing on the architecture, methods, and technologies, besides the actual development steps and results. Some organizational and work activities aspects are shortly mentioned or placed as annexes, unless essential for the comprehension of the measures and approaches adopted throughout the development, since the focus of this work is the technological challenges and analysis, and the architectural proposals. The Figure 3 presents the final version (Flight Model) of the developed payload.

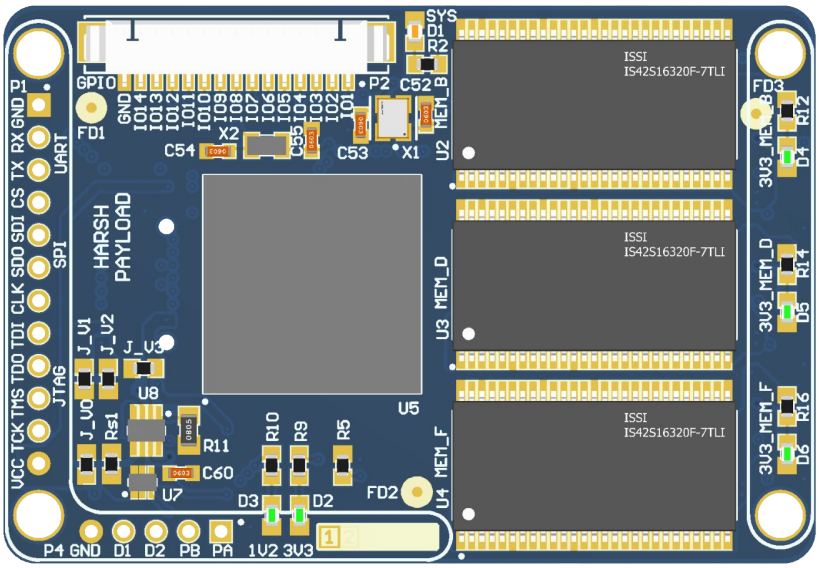


Figure 3: Preview of the developed payload.

3.1 System Overview

In order to accomplish the payload requirements, the development process reached different stages of completion: prototype phase, engineering model and flight model. These categories are a simplification of the system engineering phases definition, which generally define a deliverable and a associated review. For this reason, an overview of each stage of the system is provided to ease the understanding of the project evolution and an iterative architectural conception. The Table 1 summarizes the delivered features in each phase and overall standards compliance.

3.1.1 Prototype

During the prototype phase, the base architecture and components of the design were selected to fulfill the payload requirements. The SDRAM memories controller was the first aspect to be defined due to its importance inside the payload. This definition lead for a FPGA architecture, which could provide the necessary design flexibility and port connections, high frequencies support, and parallel execution during memory operations. Then, among different technologies and vendor options, the SmartFusion2 family from Microsemi emerged as a good trade-off between performance, power consumption, and

Features and standards	*Prototype*	*Engineering Model*	*Flight Model*
System control and management	x	x	x
Reliable radiation controller	x	x	x
Radiation hardened board		x	x
Single SDRAM memory access	x	x	x
Multiple SDRAM memory access		x	x
Static memory test algorithms		x	x
Dynamic memory test algorithms			x
Frequency test algorithms			x
Refresh rate test algorithms			x
UART communication interface	x	x	x
SPI communication interface		x	x
I2C communication interface			
CAN communication interface			
Watchdog timer protection		x	x
Low power mode operation		x	x
FSP protocol implemented		x	x
Debug logging routines	x	x	x
Experiment routines		x	x
Communication routines		x	x
System housekeeping routines		x	x
Reliable FreeRTOS instance		x	x
Firmware reliability review		x	x
Hardware reliability review			x

Table 1: Project phase progress according to the specifications.

reliability. Also, these devices have an embedded ARM Cortex-M3 processor, several useful on-chip peripherals and the advantage of being flash based, which provide a radiation hardened attribute.

Then, in order to test and validate the first design concepts, the next stage was selecting a suitable development board. Among the different options provided for this device family, the SMF2000 development kit were the best solution for the prototype conception. This board standout due to its low price, simple architecture, and all the necessary features, including a SDR SDRAM memory for the first design validations. The Figure 4 presents the board and its components highlighted: in yellow, the FTDI used to format the FPGA bitstream, debug and send log messages; in orange, the 64Mb (or 8MB) SDR SDRAM; and in red, the M2S010 SoC FPGA chip.

Regarding the delivered features of the prototype, shown in the Table 1, the base SoC design solution and the minimal system firmware were developed to attend the most critical aspects of payload, which includes the memories control and tests, simple communication routines, and usage of the internal non-volatile memory to store the firmware sources. Since the FPGA is flash based, a non-volatile memory, the bitstream is retained indefinitely and any formatting routine is necessary after system resets.

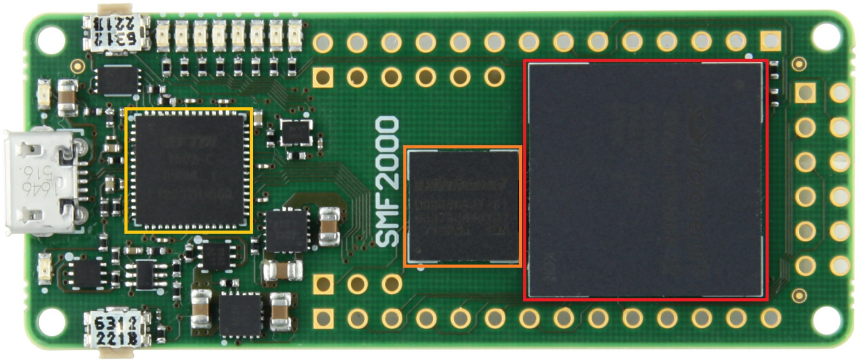


Figure 4: Development board used during the prototype phase.

3.1.2 Engineering Model

After the prototyping phase, the engineering model started to be developed, considering the previous phase review outputs. Since the first approach was using a development kit, the major external revisions focus on the FPGA SoC architecture and the software definitions. Then, the first iteration of hardware development was based on the SMF2000 and to reduce further changes, this version intended to be as close as possible of the flight model. Interfaces, connectors and the memories were the major changes from the development kit, besides the FPGA assignments and the board shape itself. The Figure 5 presents a simplified architecture diagram of the engineering model, which includes the hardware and SoC FPGA modules.

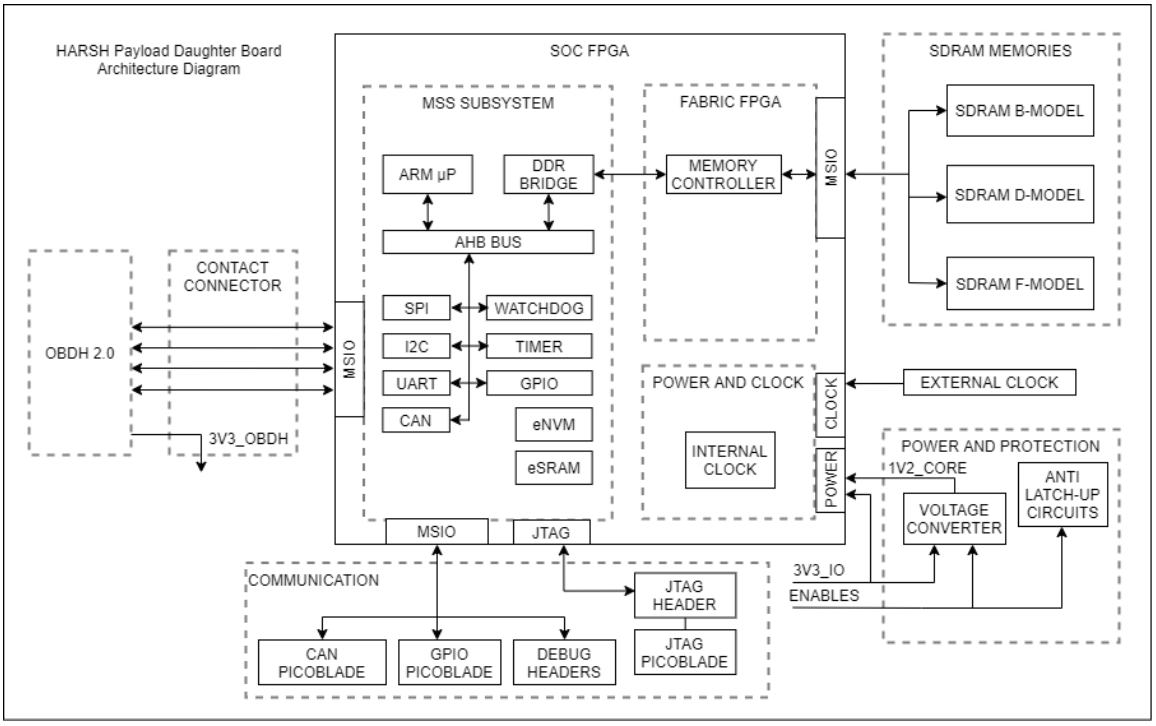


Figure 5: Simplified engineering model architecture.

In order to visualize the integration with the GOLDS-UFSC OBDH, some conceptual assemblies were performed during the development. The Figure 6 presents one of these integration setups and provide a visual understanding of both modules, their mechanical connection and electrical interface through the contact connector.

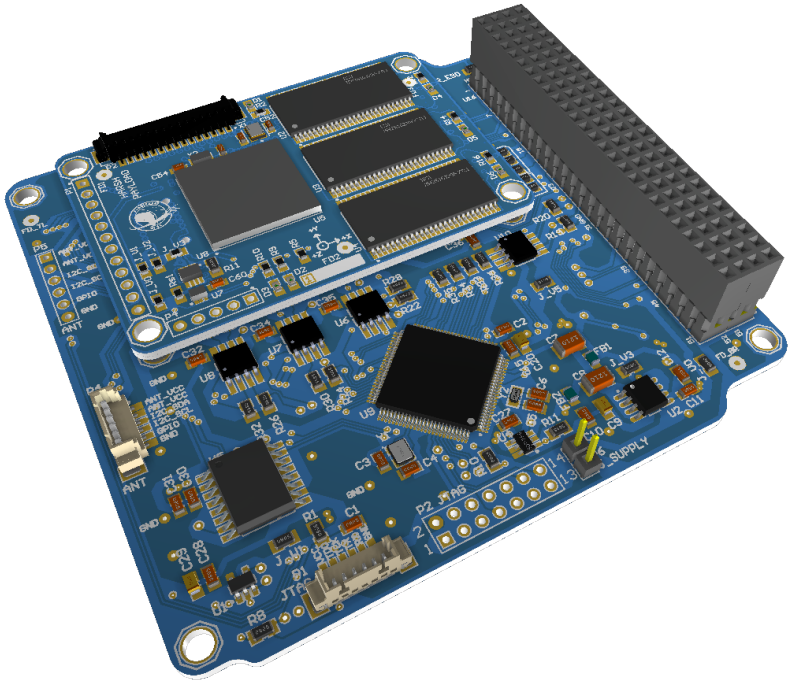


Figure 6: Conceptual payload model integrated with GOLDS-UFSC OBDH.

In summary, the engineering model review revealed that the hardware design and the SoC FPGA solution are adequate to the payload mission and requirements as a flight version. The firmware, showing the basic operation and experiments, presented satisfactory overall stability, but it was decided to refactor some routines, perform more specific tests, and improve the reliability and redundancy of some modules.

3.1.3 Flight Model

Regarding the flight qualified model, the major changes from the engineering version were the system firmware and the experiment algorithms. These modifications were important to improve the experiment execution and provide better results. Also, the integration tests revealed some weak points and non-reliable management routine implementations that might led to critical failures. Then, in order to solve these issues, the firmware was updated, tested and the design reviewed. These changes were related to the memory controller refresh rate, data acquisition and some synchronization issues. The design presented in this document covers the updated and functional implementation.

3.2 SoC FPGA Architecture

The payload core subsystem is the SoC FPGA, which provide the system control and management due to embedded microprocessor, memory controller, communication interfaces and main peripherals. The device settles the main functionalities of the payload system, then using a radiation hardened technology improves the overall system reliability and permits the analysis of the memory faults within the absence of controller errors. Also, the SmartFusion2 family is suitable for critical applications due to its intrinsic robustness, high density die, high-performance and availability of a wide variety of peripherals features. Moreover, these devices has a historic of utilization in space applications and demonstrates satisfactory overall performance and reliability. The Figure 7 provide a internal physical architecture overview of the SmartFusion2 devices. The following sections describe the SoC system implemented in the M2S010 FPGA, part of this device family, and include its simulation and test results.

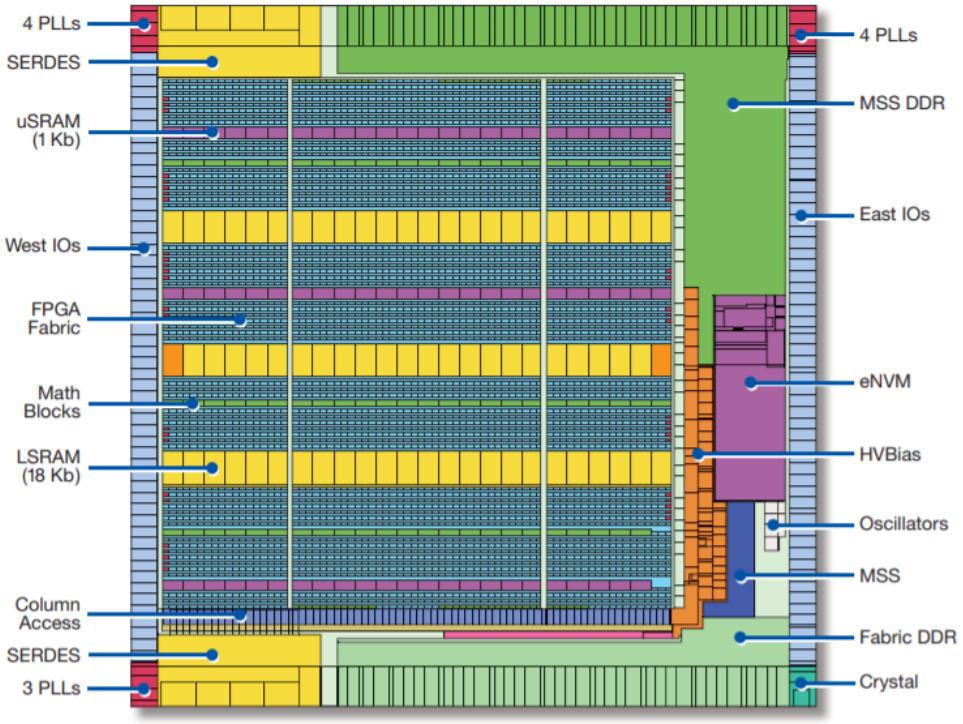


Figure 7: FPGA die overview.

3.2.1 Design Overview

The SoC architecture embedded in the FPGA chip has several features and allows each subsystem to be enabled or disabled in the design, in order to reduce the power consumption. The device is subdivided in two different parts: the Microcontroller Subsystem (MSS), with a fixed implementation, and the FPGA Fabric, where the user modules and configurations are implemented. For this project, both parts are employed to create the payload required capabilities and define the core design: controller, using the ARM Cortex-M3 processor; Soft Memory Controller(SMC) for the SDRAM management, in the FPGA Fabric section; serial communication (MMUART_0), SPI_0, I2C_0); Watchdog

timer, for system protection; RTC and timers, for synchronization; GPIO ports; Embedded Non-Volatile Memory (eNVM_0) for code sources storage; and debug, using the System Controller for JTAG interface. The following topics details each subsystem characteristics and operation. Also, the Figure 8 presents a simplified diagram of the built-in internal available modules.

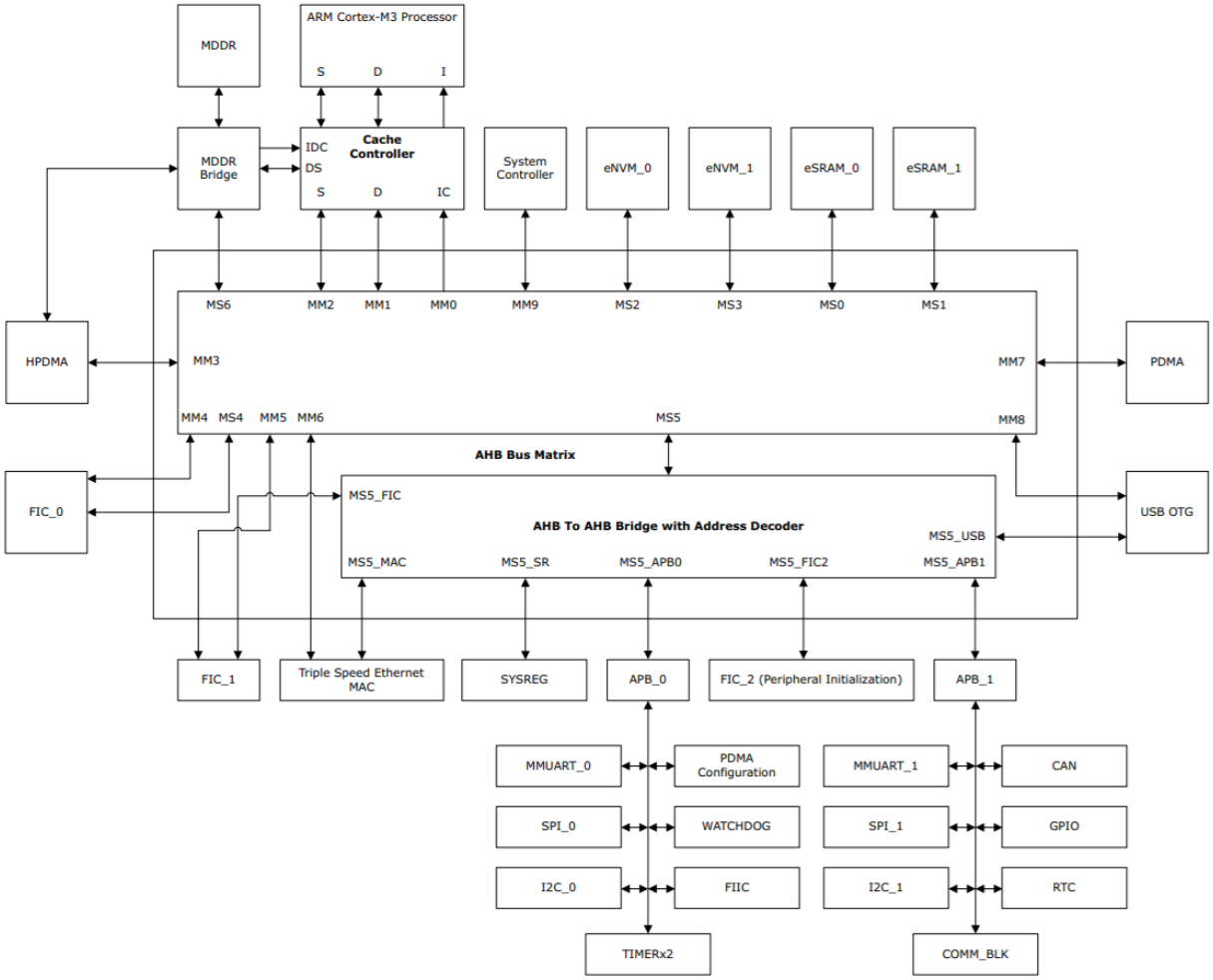


Figure 8: Internal architecture diagram overview.

Microprocessor

In the SoC architecture, the processing unit is a hard core instance of an ARM Cortex-M3 microprocessor. The unit has a low power consumption design and intend to accomplish the deeply embedded applications requirements. This processor consists in a 32-bit RISC architecture with 3-stage pipeline that provide enough processing performance for the payload routines and algorithms. In order to provide the utility features for this device, the SoC design includes a set of peripherals, which combine communication, timers, system memories, watchdog and debug modules. When disabled, these subsystems are hold in low power mode and provide customization during the development that contribute for the feasibility in power constrained applications. Also, the system uses the

ARM Advanced Microcontroller Bus Architecture (AMBA) as interface for the peripherals and the FPGA fabric devices, employing the Advanced High-performance Bus (AHB) protocol in the communications.

Memory Controller

In order to manage the SDRAM memories, a controller instantiated in the FPGA fabric is used for the external interface with these devices. In this architecture, the controller is shared among the memories since the operations campaigns are performed separately for each device and concerning timing the memories are compatible. Moreover, due to the similar internal logical architectures, despite the manufacturing differences, the memories are similar in operation and constraints. Figure 9 describes the interfaces used for the controller from the processor until the external interface. The processor, a master in the AHB bus, requests data operations for the DDR Bridge that redirect these commands for the memory controller, using a AHB to AXI interface. Then, the controller, besides the regular management operations, handles the communication and control signals through the MSIO ports to the actual memory. Since the SDRAM chips are connected in parallel in the same MSIO ports, the controller uses chip select signals to enable the correct device.

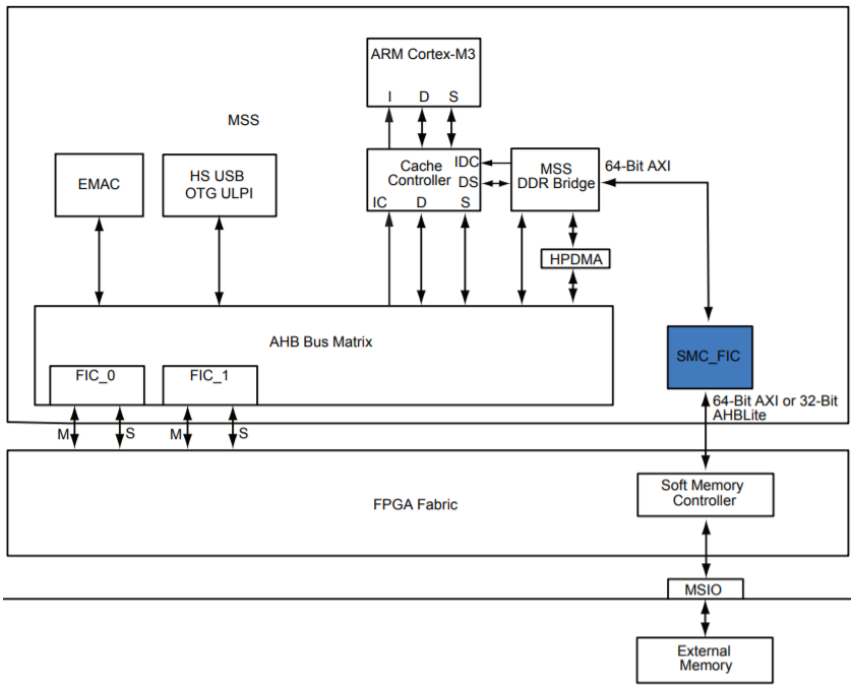


Figure 9: Memory controller interfaces overview.

This controller is part of the catalog collection provided by the SoC FPGA vendor, which provides several ported IP Cores for the fabric implementation. The used core is designed for SDR SDRAM memories and adapted to be connected to an Advanced eXtensible Interface (AXI). The Figure 10 shows a simplified diagram of this IP core. In summary, this module receive operation requests through the AXI bus and the internal controller handles this demands, generating the formatted operation frame for the actual memory controller and handling its responses for the requester. However, since the MSS subsystem (the requester) architecture provide an AHB interface, a converter is required

to properly handle the protocol exchange for an AXI bus. Also, this module is provided in the vendor catalog and do not require external control, besides the reset signal.

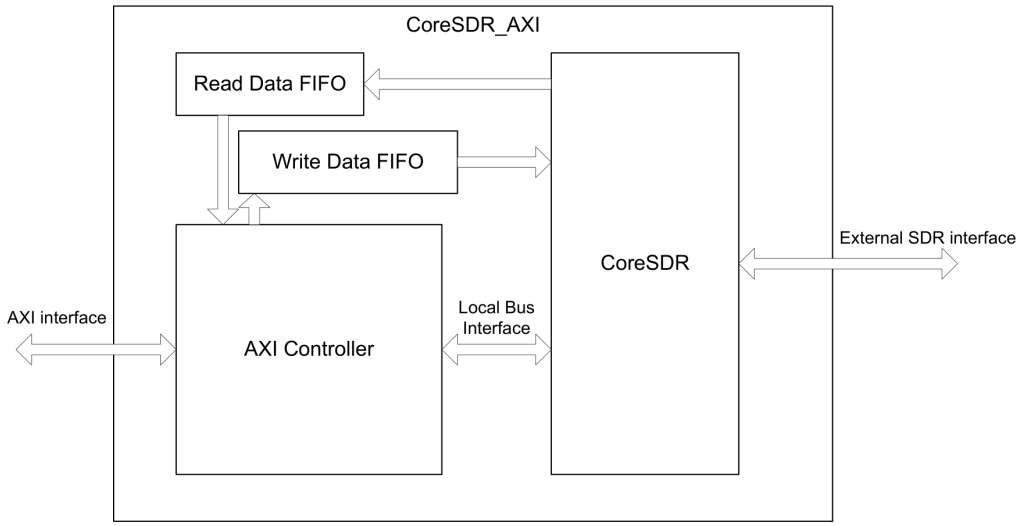


Figure 10: Core SDR wrapped for an AXI interface architecture.

The CoreSDR is the controller that manage the memory communication, generating the control signal within the time constraints. It receives the operation requests, data length and addresses from the AXI controller and the data from the write buffer in case of a write operation. The input address is structure as described in the Figure 11. This scheme is important to determine which memory should be accessed in relation to the input address, where the three most significant bits are decoded in up to eight chip selects and the others bits to set the column, row and bank. When an encoded chip select bit changes, it means that the address space of that memory ends and the next starts. This scheme allows the usage of up to eight memories without the need of additional controllers or external interfaces.

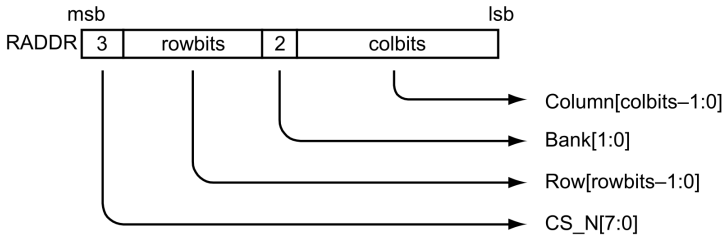


Figure 11: Core SDR Address scheme overview.

The Figure 12 shows the internal modules of the controller and the corresponding signals. The output signals are directly connected to the SDRAM memories: RAS, CAS and WE determine the requested commands; SA and BA set the target addresses; DQM is a mask for bytes within a word; CKE is used to enable the clock in the memory; CS activate the selected memory; DQ is used for input and output data; and OE is used to enable the tri-state in DQ signals. The refresh control unit is used to send periodic refresh commands, due to the SDRAM technology requirements, and the address generation module convert

the input addresses to the memory access format. The rest of the subsystems handles other functionalities: initialization, resets and timing constraints. Regarding the last item, the three memories have compatible time requirements and latency characteristics at the payload operation frequency, which were manually configured to attend the datasheet specifications.

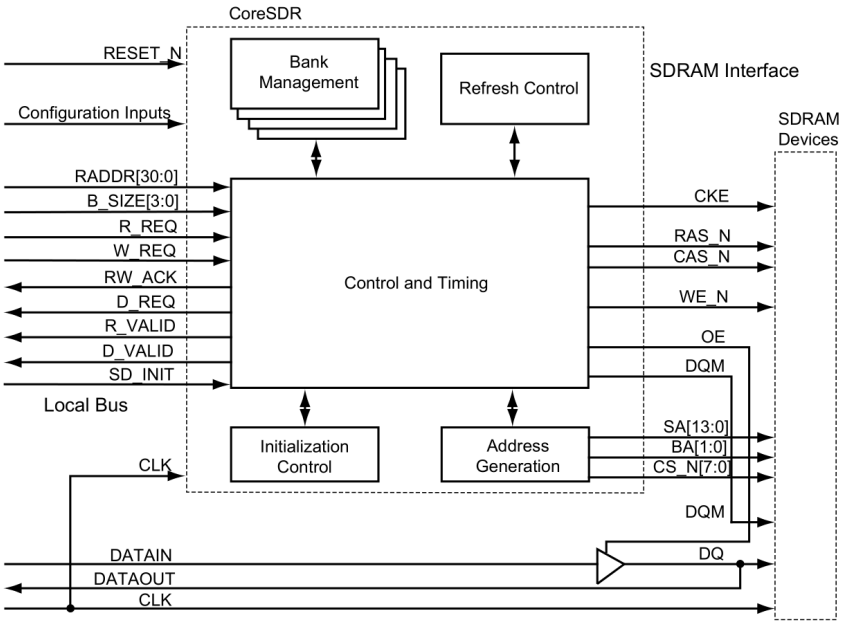


Figure 12: Core SDR architecture diagram overview.

Embedded Memories

In order to store the code and data, the SoC solution includes two different types of memories: embedded Non-Volatile Memory (eNVM), using the flash technology; and the embedded Static RAM (eSRAM). For the payload design, the eNVM is dedicated for code storage, constants and runtime variables, which provides to the system all required functions during execution and reboots. Also, since the eNVM is a flash memory, the radiation effects are reduced due to its intrinsic resilience. However, considering that the experiment memories (SDRAM) could generate many bytes of error, in case of block failures, the eSRAM is used to store these logs. This approach guarantee that the system tasks will be properly executed without memory issues due to capacity, since the potentially dangerous growth capable data is stored in a different device. In this scenario, the worst case is achieving the eSRAM memory capacity and overwriting the older results. Then, in order to avoid this problem, a system behavior change is triggered when determine memory usage levels are achieved.

The eNVM has a capacity of 256kBytes that provides a secure size for all the system applications and runtime variables without limitation issues. Moreover, the eSRAM two modules with 32kBytes each when SECDED-ON and 40kBytes each when SECDED-OFF. The single error correction, double error detection (SECDED), a extension of the Hamming code, is a technique to improve the data storage reliability, in this case on the eSRAM, which is less resilient to the radiation effects. Since this algorithm is a built-in

feature of the SoC FPGA and the capacity is handled in software, the eSRAM is used with the SECDED enabled. The Figure 13 describe the eSRAM scheme in both cases. Then, using this approach, it is possible to avoid potential problems with the system memories, allowing the payload to properly evaluate the radiation effects on the target devices, the SDRAMs.

eSRAM Block	Physical RAM4096X40 Block	Size and Address Range with SECDED ON	Size and Address Range with SECDED OFF
eSRAM_0	RAM4096X40_0	16 KB from 0x20000000 to 0x20003FFF and ECC from 0x20010000 to 0x20010FFF	16 KB from 0x20000000 to 0x20003FFF and 4 KB from 0x20010000 to 0x20010FFF
	RAM4096X40_1	16 KB from 0x20004000 to 0x20007FFF and ECC from 0x20011000 to 0x20011FFF	16 KB from 0x20004000 to 0x20007FFF and 4 KB from 0x20011000 to 0x20011FFF
eSRAM_1	RAM4096X40_2	16 KB from 0x20008000 to 0x2000BFFF and ECC from 0x20012000 to 0x20012FFF	16 KB from 0x20008000 to 0x2000BFFF and 4 KB from 0x20012000 to 0x20012FFF
	RAM4096X40_3	16 KB from 0x2000C000 to 0x2000FFFF and ECC from 0x20013000 to 0x20013FFF	16 KB from 0x20008000 to 0x20003FFF and 4 KB from 0x20013000 to 0x20013FFF

Figure 13: eSRAM memory scheme.

Serial Interfaces and IO Ports

The Microsemi FPGA SoC solution provides several peripherals in the MSS subsystem. For this payload, it is used different serial communication protocols (SPI, I2C, CAN and UART) and GPIOs to provide flexibility and redundancy. The SPI protocol, as slave, is used exclusively for communication with the GOLDS-UFSC OBDH. In this interface is used the FloripaSat Protocol (FSP), in order to attend the OBDH requirements and increase the communication reliability. Also, as redundant channels, the I2C (as slave) and CAN could be used for the same interface. This design allows compatibility with further missions of the FloripaSat core, since the payload could be used as a health monitor for radiation damage. Moreover, a UART interface is provided for system logging, which ease the development and debug processes, and a JTAG for the FPGA programming. Also, 20 GPIOs are available for parallel protocols, hardware tests or debugging, 6 IOs for the latch-up monitors and two pins for the FPGA live probes, used for system monitoring during debug.

These interfaces are available in different connectors, models and locations depending on priority and utilization: SPI, I2C and 4 GPIOs on the contact connector (interface with the OBDH); CAN, JTAG and 14 GPIOs on dedicated connectors; UART, JTAG, live probes and 2 GPIOs on dedicated connectors only available in the engineering model due to space constraints; and the 6 IOs, directly connected to the latch-up connectors.

Watchdog Timer, Interruptions and Timers

The internal watchdog timer provide a failure recovery feature for the payload system, without the necessity of an external device. This mechanism protects the payload against software errors by resetting the device and restoring the system execution from boot. Moreover, the payload uses the available timers to synchronize the peripherals execution and trigger system actions, alongside the interruption handler that manages these events.

3.3 Firmware Architecture

The CubeSat standard, although restrictive concerning definitions of external mechanical and electrical interfaces, integration guidelines and materials utilization, it does not restrain the implementation of the internal subsystems. Regarding the firmware architecture a considerable freedom allows the developers to propose their own standards using approaches and patterns from different application niches. For instance, companies tend to introduce a proprietary architecture in their platforms, creating a tailored environment of modules and development tools [11] [12] [13]. Moreover, since the CubeSat missions have a substantial contribution from the academic environment, a wide range of techniques and models are employed intending to attend the application requirements and explore novel strategies [14] [6].

However, these projects share principles and patterns inherent to critical applications, and more specifically, space niche that imply focusing on the same aspects: reliability, environment hardness, redundancy and predictability. Then, regarding the software development, using these concerns to design failure-aware systems generally lead to sophisticated and robust architectures, which increase the mission success probability. Considering the scope of this project, the following sections detail the aspects that guided the payload firmware development, describe the behavioral operation of each module and include the subsystems design.

3.3.1 Design Guidelines

Failure Protection Strategies

The payload firmware is designed to decrease failures due to system malfunctions and environmental conditions. Some of these strategies are periodic system resets, watchdog timer, hardware check routines, default parameter values and error correction algorithms in the system memories and communication interfaces. For instance, the embedded SRAM memories, designated to store the experiment results, use the Single Error Correction, Double Error Detection (SECDED), a extension of the Hamming code. Also, in the communication with the GOLDS-UFSC OBDH, a Cyclic Redundancy Check (CRC) algorithm, an error-detecting code used to detect accidental changes to raw data, is applied inside the interface protocol.

Moreover, the payload firmware in based on an operating system targeted to the embedded systems, which improves the overall reliability since the platform has inheritance of several projects in different applications, including in the space sector. Then, using this framework to develop the payload firmware, the system failures are mitigated and the architecture more qualified to the application.

Internal and External Synchronization

The system synchronization within internal constraints and with external devices is a challenge since the payload is intended to operate controlled by the OBDH and, meanwhile, perform several functions within certain time and priority limitations. Then, using the operating system framework, this requirement is accomplished using queues, tasks and semaphores and interruption handlers, which provide tools to accommodate the different constraints in a deterministic fashion. In summary, each task has an execution rate, priority and optional parameters (semaphores, initial delays and group events) that

are predictably handled by the framework scheduler. Besides this execution scheme, data should be transferred between these task without the necessity to be immediately handled, which is the fundamental strategy of the data synchronization. Also, since the system operation is controlled by the OBDH (slave mode), it is necessary a interruption handler for communications, which is non-deterministic and could cause issues. Then, to avoid this problem, a binary semaphore is used to create a link between the interruption occurrence and a handler task, which transforms the operation deterministic in terms of execution alongside rest of the system tasks.

Moreover, the communication with the OBDH has synchronization strategies due to the FloripaSat Protocol (FSP), which defines a communication standard between both modules. This protocol uses handshake operations, acknowledge responses and set a command list, besides error-detecting routines to ensure correct communication. These herein strategies and features are detailed in depth throughout the following sections, reveling the actual architecture, operation mechanisms and consequences.

Abstraction Layers

The firmware structure is based on abstraction layers, which create a separation between layers and allow a development oriented to the application itself instead of the hardware details in that feature. Also, this strategy increase the firmware readability, flexibility, organization and maintenance, due to the implementation independence from distant layers and abstraction of unnecessary information. The Figure 14 presents this layers and their hierarchy. At the bottom, it is represented the lowest level, the SoC implementation, then the first abstractions, called hardware abstraction layers due to handling with specific operation related with the processor and peripherals. At the upper levels, there are the application layers that use a high-level description approach for the implementation of the payload system functionalities and operations.

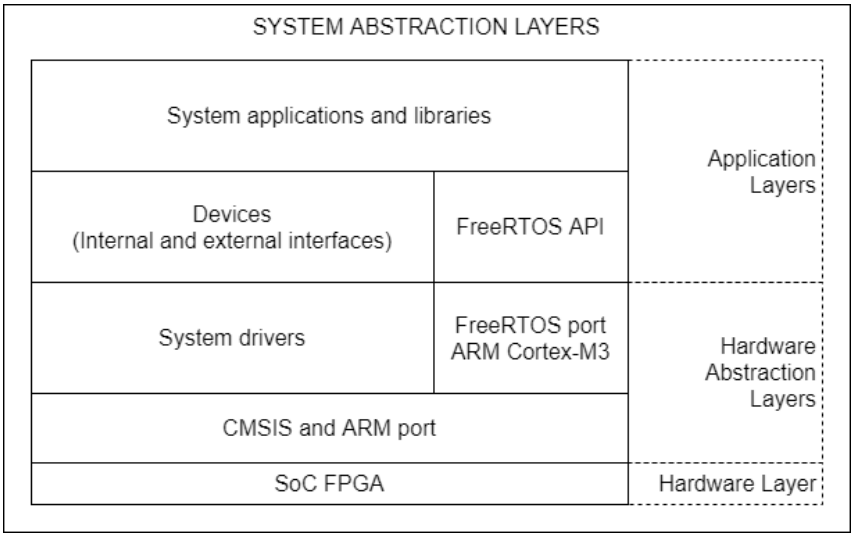


Figure 14: System abstraction layers diagram.

3.3.2 Design Overview

The firmware architecture, as herein described, use an embedded operating system that is widely used in commercial products. The FreeRTOS is a real-time operating system (RTOS) for microcontrollers and small microprocessors with proven robustness, tiny footprint, and wide device support. Then, in order to use a similar architecture as the GOLDS-UFSC OBDH, share libraries and ease the development process, the FreeRTOS was the most feasible platform for this project, besides the official port for the Microsemi SmartFusion2 SoC FPGA devices. This firmware development approach lead to the conception of functionalities translated into operating system tasks and synchronization schemes in queues and semaphores. In summary, the firmware is designed using the operating system features and structure.

Regarding the system functionality, the Figure 15 describes a simplified execution diagram, presenting the main experiment functions and their sequence. The housekeeping and system functions are omitted to evidence the major payload purpose, the experiment execution itself. After power-on and internal setup, the payload receives a command from the OBDH, prepares the experiment parameters and enables the experiment execution. Then, this data is stored in the embedded SRAM for later retrieve when a OBDH data request occurs. This loop summarizes the experiment application and the payload purpose, evaluate the SDRAM memories in the harsh environment.

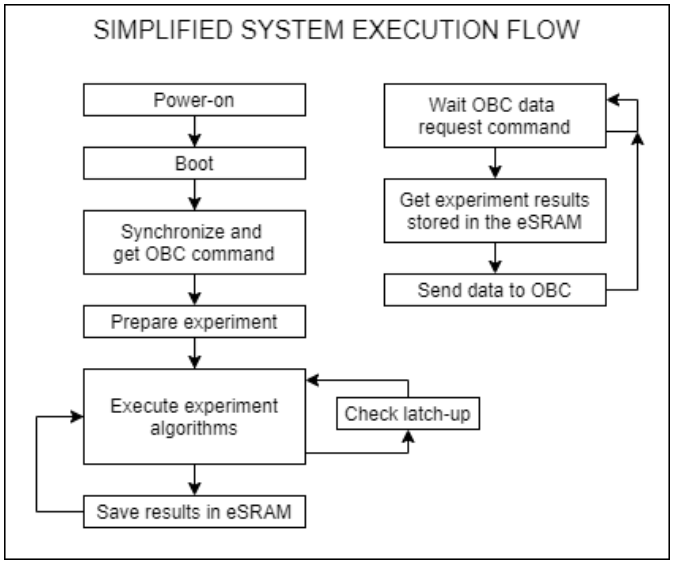


Figure 15: Simplified system execution flow diagram.

These functionalities, as herein noted, are translated in tasks, queues, semaphores and interrupt handlers, which are represented in the Figure 16. Also, some parameters are presented (priority, rate, periodicity and depth) that are detailed and justified in the following topics.

Experiment Routines

The experiment is executed through two dependent and periodic tasks: ”experiment manager task” and ”experiment runner task”. The first is responsible to handle the commands received through the communication, constantly check the latch-up monitors, and

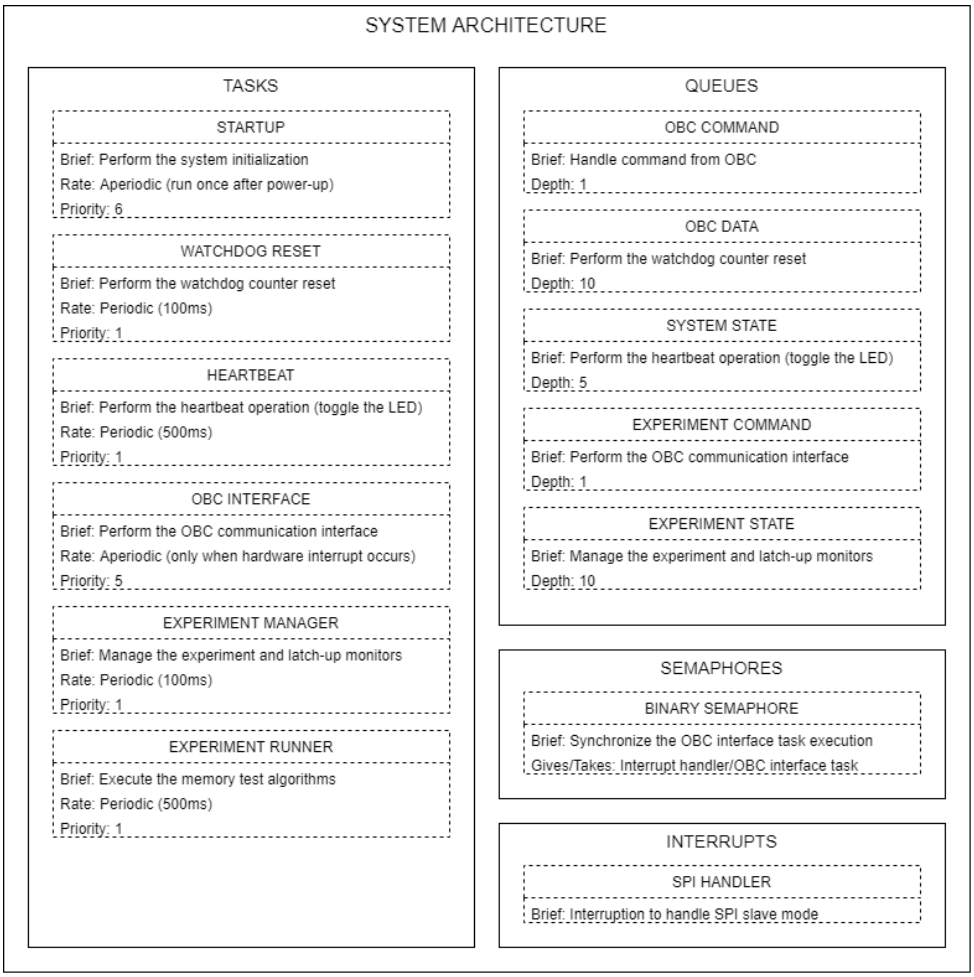


Figure 16: System architecture diagram.

coordinate the system queues. The second task coordinate the experiment execution and save the generated experiment data results. Concerning the execution flow, the manager task has a higher priority than the runner one, thereby executing despite the experiment cycle accomplished the end. This design would lead to issues if a synchronization mechanism did not coordinate the execution of one task accordingly to the other. Then, in order to fulfill this need a queue system was designed to trigger some states that are executed in key moments, for instance: reading the memory to restore the experiment results, changing the execution configuration parameters, and many other functionalities.

There are two queues directly used in the experiment routine and three indirectly, which transfer data, configuration parameters and status. The two direct queues are used for receiving the configuration from the manager to the runner task, and sending the position and size of the experiment data stored in the dedicated memory sector to the manager task. The other queues are used by the manager and communication tasks for inputting the commands from the OBC, and outputting the data and status information to the OBC.

The Figure 17 summarizes the experiment execution flow, presenting from the communication with OBC to the execution of the memory test algorithms. It starts with the command received from OBC, process the execution parameters, run the test algorithms, store data in the dedicated memory, send data status for further retrieving, read this data,

and finishes sending it to the OBC. This sequence is a simplified overview and support the visualization of the routine, but it should be analysed as two tasks not necessarily contiguous in time and using queues to trigger sequential events.

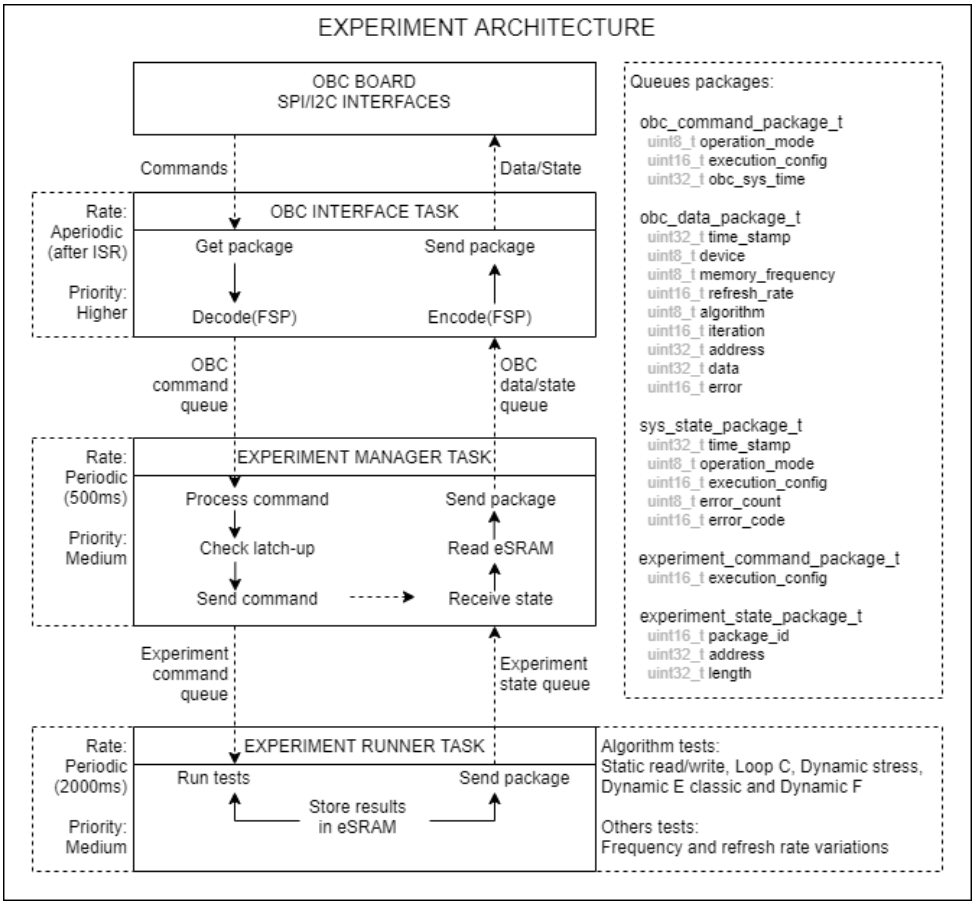


Figure 17: Experiment architecture diagram.

Housekeeping Routines

The housekeeping routines are responsible to initialize and maintain the system properly configured and operational. There is an aperiodic task referred to as ”startup” that performs the system boot once after power up, in other words, the initialization of all peripherals, interfaces, and devices. Also, there is a dedicated periodic task that ensures constant notification of properly operation for the watchdog timer and another one with the lowest priority for other periodic non essential functions (e.g., blink LED).

Communication Routines

The communication routines are handled using an aperiodic task and interruptions (one for each physical protocol). The task execution trigger is associated with the interruption since the payload is a slave in the communication from the perspective of the OBC. Then, to ensure that the communication is correctly attended, the task has the highest priority. This strategy is adopted to accomplish both requirements, attend communication requests with low latency and use the payload firmware formal application

structure (i.e., the operating system tasks). Also, this method prevents conflicts between the scheduler and manually configured interruptions, and allows the use of the operating system features safely. The Figure 18 presents the scheme adopted to receive and process the communication interactions. Also, the Figure 17 describes the interface of the communication task with the experiments routines using queues for synchronization.

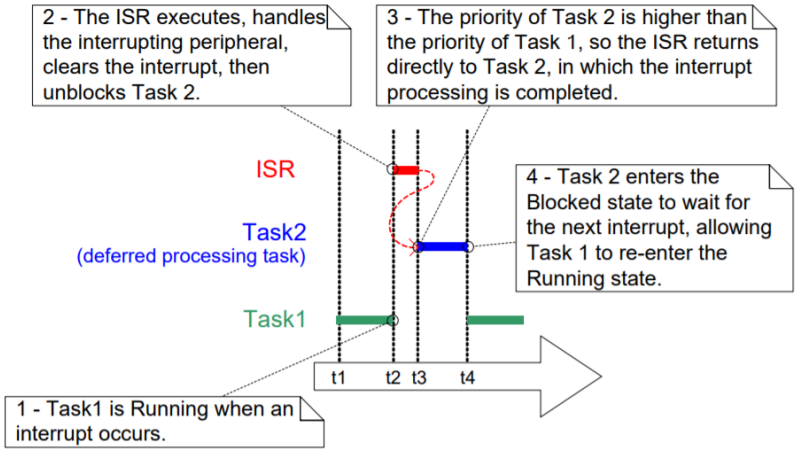


Figure 18: Communication interrupt and task architecture diagram.

Debug and Logging Routines

In order to perform debug sections and execute integration tests, there is a logging system to provide useful feedback during the development. The routines use a serial interface (UART) to output relevant steps and notify critical failures, which are easily read through a serial monitor (i.e., PuTTY or equivalent). These logs are inserted in key positions, using status flags to select between appropriate messages.

3.3.3 Memory Fault Detection Algorithms

There are two subcategories of memory detection algorithms: static, more generic and global error detection, and dynamic, which try to catch specific errors when they occurs (usually are model and technology dependent). Due to the general purpose characteristic and simplicity of the static algorithms, this work implemented two static algorithms. The first is a simple verification that performs write operations in all the memories address space with a specific value and later, after some adjustable period, it is read and compared with that value. In case of faults, the algorithm reports relevant information about the execution. The second algorithm follows a similar approach, but divide the memory in different sized portions and performs a correction attempt to ensure that the fault was a radiation functional problem and not a communication error. Despite similar, the methods have different sensibility for certain faults and are better applied in distinct scenarios. For example, the first is more prone to single event upsets than the second or the second is more reliable at the expense of execution speed.

3.4 Hardware Architecture

The payload hardware consists of a 6-layer PCB, using the GOLDS-UFSC OBDH DaughterBoard standard and following simplified space application design guidelines. In order to enumerate this patterns and constraints, the following section describes the applied techniques and particularities of the payload design. Then, each developed modules receives a brief architectural and functional description. Moreover, the tests results are analysed to evaluate the requirements accomplishment and developed functionalities.

3.4.1 Preliminary Design Analysis

Layer Stackup Planning

The payload layers scheme were specified to attend the rules described herein and to increase the feasibility of production, which uses convenient dimensions for the target manufacturer. Since the engineering model do not require space qualified boards, this scheme allowed to reduce the costs and time during this process. The Figure 19 describe the payload stackup, including: layers, dimensions and characteristics. In order to avoid tracks in the external layers, the strategy was to use the middle ones for routing, permitting the shield properties of these external layers and suitable power planes in the internal ones. Moreover, to reduce crosstalk between the signal layers, they are orthogonal concerning the routing preferential directions. Also, despite the external layers assigned for ground reference, some parts are used to handle the external supply voltages to prevent splitting the power plane, which lead to efficiency reduction as low impedance reference and increase the return paths. In summary, to correctly employ this stackup, the internal layers must have the lowest impedance (short return paths) and the external providing electromagnetic interference shielding.

Layer Name	Type	Material	Thickness (mm)	Dielectric Material	Dielectric Constant
Top Overlay	Overlay				
Top Solder	Solder Mask/...	Surface Mat...	0.01016	Solder Resist	3.5
Top-Outer-Layer (GND)	Signal	Copper	0.035		
Dielectric 1	Dielectric	Prepreg	0.1	FR-4	4.2
Top-Mid-Layer Signal	Signal	Copper	0.0175		
Dielectric 4	Dielectric	Core	0.565		4.2
Inner-Layer (GND)	Internal Plane	Copper	0.0175		
Dielectric3	Dielectric	Prepreg	0.127	FR-4	4.2
Inner-Layer (VCC)	Internal Plane	Copper	0.0175		
Dielectric 5	Dielectric	Core	0.565		4.2
Bottom-Mid-Layer Signal	Signal	Copper	0.0175		
Dielectric2	Dielectric	Prepreg	0.1	FR-4	4.2
Bottom-Outer-Layer (GND)	Signal	Copper	0.03556		
Bottom Solder	Solder Mask/...	Surface Mat...	0.01016	Solder Resist	3.5
Bottom Overlay	Overlay				
Total Thickness: 1.61788mm					

Figure 19: Stackup planning.

Placement Planning

The components placement were defined using the external connectors accessibility, considering the position in relation to the motherboard (the GOLDS-UFSC OBDH), and internal requirements. However, the positioning of the FPGA device and memory

chips were the major factor considered, since the most critical routing is between these components. Also, to ease the layout development process, the memories were placed symmetrically placed in relation to the FPGA and uniformly distributed in the board. The Figure 20 present the FPGA pin assignment to attend these requirements. It is important to note that the DDRIO and MSIOD banks were not used due to the maximum voltage levels allowed, 2.5 Volts. This assignment scheme provide three groups of signals in different directions, which ease the BGA fanout and tracks escape routes: memory control and address signals; memory data signals; and general purpose inputs and outputs.

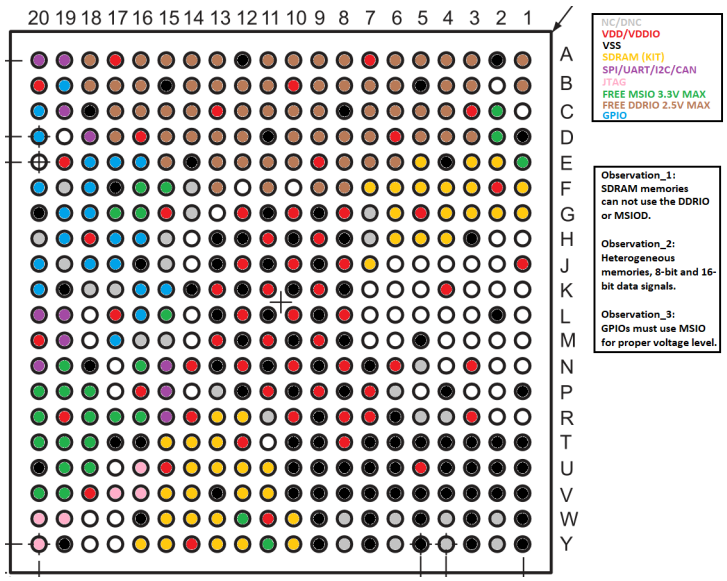


Figure 20: FPGA pinout usage.

High-Speed Design Planning

The board component interfaces are in the edge of being considered as high-speed signals due to the SDRAM memories. Then, even if a more complex approach is not required for this work, some measures were considered in the payload design to avoid issues and bad performance. The most important strategy is the properly grounding and power planes for return signals, which were herein described in the stackup planning topic. Also, an orthogonal and similar length routing strategies were adopted to avoid crosstalking and latency issues in the memory signals. The memories are placed symmetrically to provide an easier and organized pattern for routing.

The Figure 21 presents the signal layer used for tracing from the FPGA to the memory chips. The highlighted tracks are the address signals that are routed together to mitigate the length mismatch. The other signals are grouped together depending on their purpose, control or data signals.

The Figure 22 shows a similar overview of the same address signals, but presenting them in the other signal layer. The tracks are traced in the shortest routes as feasible and provide the required parallel connections, since the three memories share the same controller pins interface.

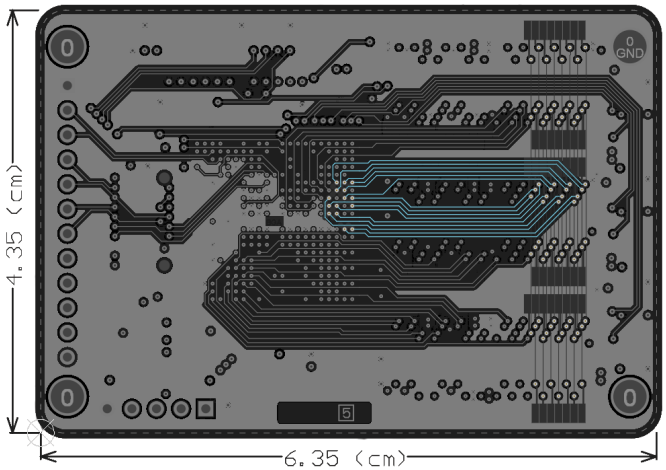


Figure 21: Top Mid-layer routing with focus on the memory address signals.

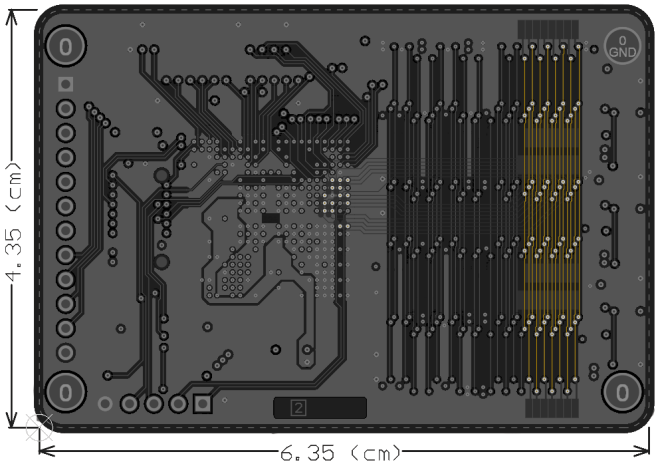


Figure 22: Bottom Mid-layer routing with focus on the memory address signals.

3.4.2 Design Overview

In summary, the payload hardware essentially consists of a controller, to manage communication and memories, and the SDRAM devices under experiment. However, some auxiliary components and modules are necessary to handle other tasks and fulfill the requirements, including: power converter, to supply the correct FPGA voltage; latch-up monitors, to prevent catastrophic system failures and provide additional data for the experiment; debug support for power supply, logging and programming; additional communication modules and connectors; and the motherboard interface connector. The Figure 23 presents the payload subsystems, internal connections and external interfaces, besides an internal overview of the SoC FPGA device implemented architecture.

In addition to the summarized power supply architecture herein presented, the Figure 24 presents in more details the power line names, voltages, and currents, besides their relation with the latch-up monitors.

The Figure 25 presents a rendered view of the board top and bottom layers with indications where each module is situated, and the Figure 26 shows the manufactured

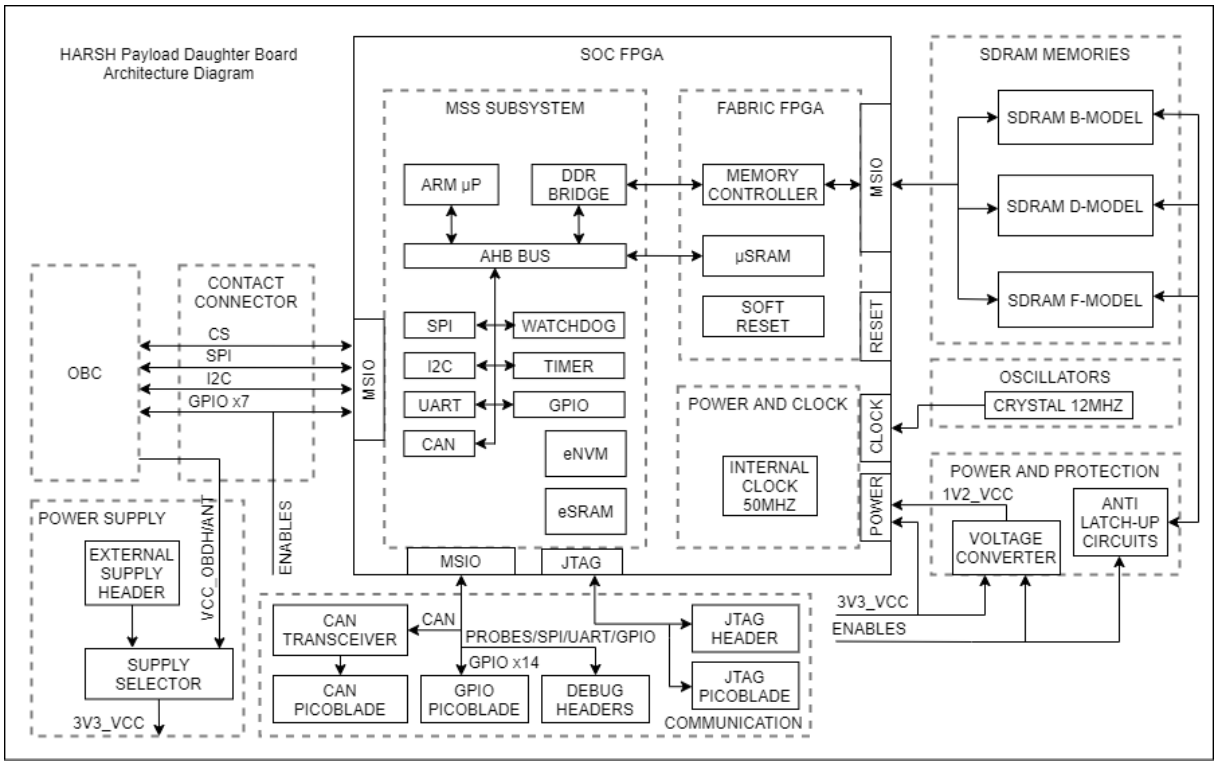


Figure 23: Hardware architecture overview.

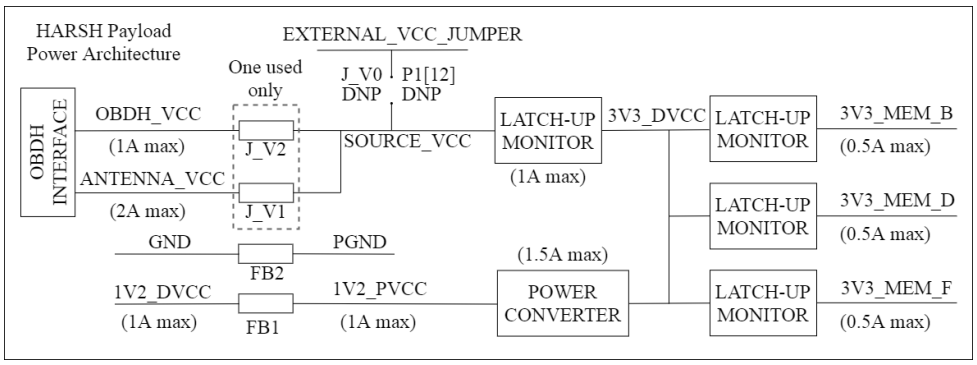


Figure 24: Power architecture overview.

board. The following topics describe these subsystems functionality and characteristics. Also, in the schematics section of the Appendix A, the schematics and layers prints are presented.

SoC FPGA Module

The FPGA device is a Ball Grid Array (BGA) component that require a fanout implementation and escape routes for the internal pins. The adopted strategy for this work is the dog-bone pattern, which is a widespread technique and attend the project necessities. The Figure 27 present the implemented pattern and escape routes and decoupling capacitor placement, both using a top view of the board. Since several FPGA pins are

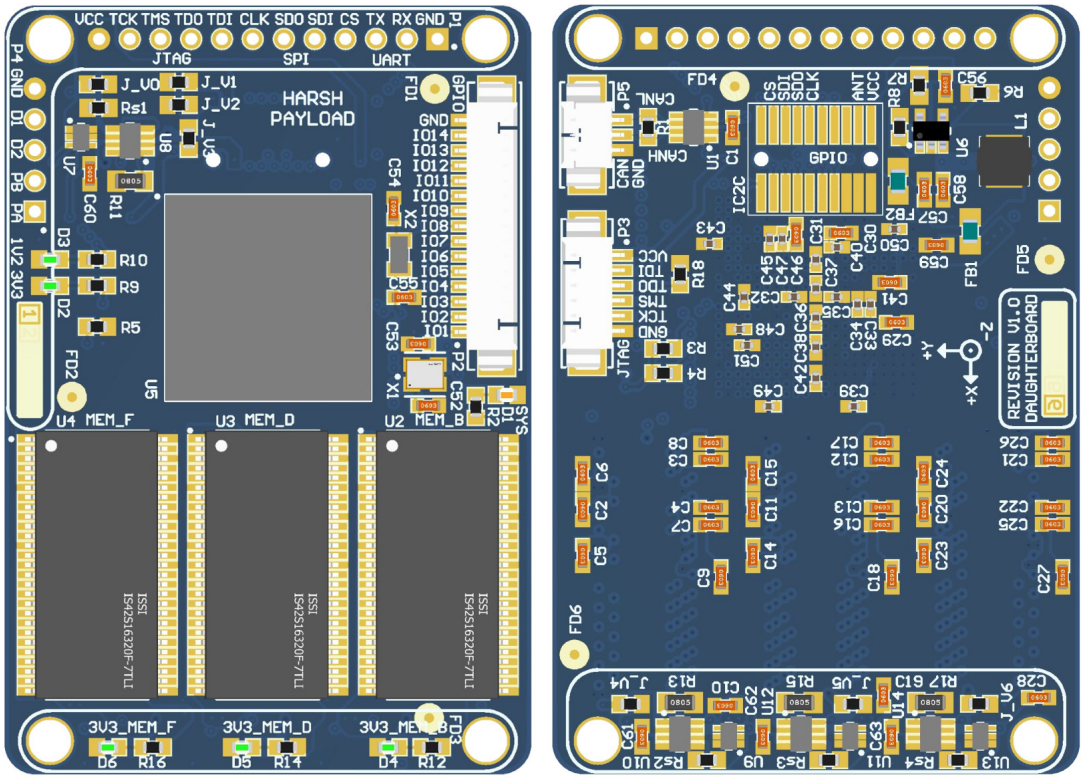


Figure 25: PCB 3D rendering overview.

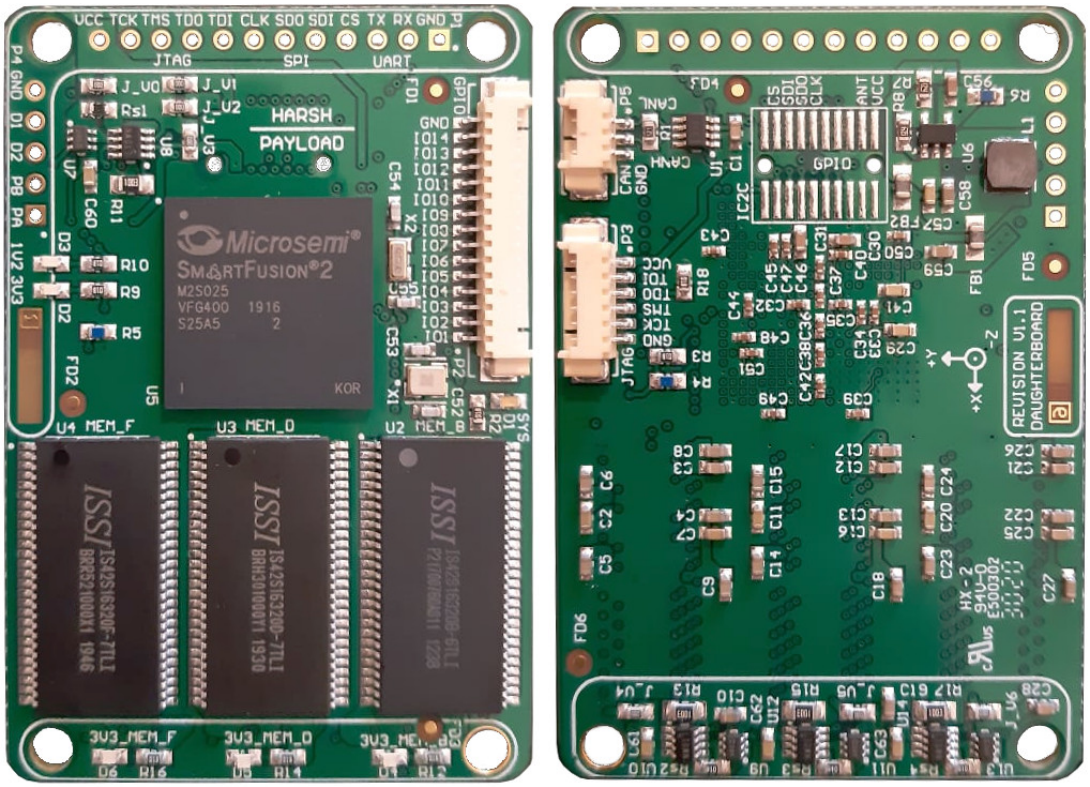


Figure 26: PCB manufactured overview.

not used and the board placement favors the routing, this implementation only required two signal layers and two power planes, which attend the payload specification.

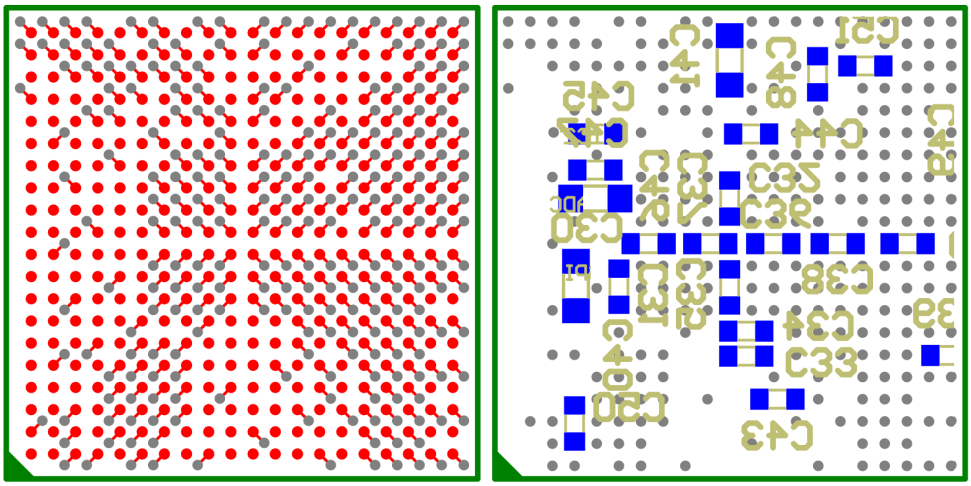


Figure 27: FPGA fanout dog-bone pattern and decoupling capacitors implementation.

Memories Module

The memories module is composed of the three experiment memory chips: IS42S16320B, IS42S16320D, and IS42S16320F. These devices are 512Mb (or 64MB) Single Data Rate (SDR) SDRAMs operating at 3.3V and up to 143MHz in a 54-pin TSOP-II packaging. A SDR device means that it can only read/write one time in a clock cycle, differing from its successors Double-Data Rate (DDR) memories that perform two operations per cycle. The selected parts configuration has a 16-bit data width and the industrial grade qualification.

The target operation frequency is 100MHz using the other parameters in the nominal indications, except for the refresh frequency (8K cycles every 64ms) that might be changed during some experiment tests. The electrical topology is a parallel connection that shares the same controller interface. This strategy was adopted since it saves the usage of several FPGA pins and the used configuration is the same.

During the board design, as presented in the Figures 25 and 26, the placement was selected to avoid any component unrelated to the memories be located on their opposite side. Also, this measure ensures that the decoupling capacitors are as close as possible of their power pins. Regarding the routing, the strategies employed were described in the subsubsection 3.4.1.

Communications and Debug Interfaces

The payload uses several communication interfaces distributed in different connectors according to the requirements or usage. There are six available interfaces for different purposes: SPI, JTAG, UART, I2C, GPIOs, and CAN. The SPI, JTAG, and UART are the main used communication protocols and the rest were added for redundancy or additional non critical features. In the same scheme, there are six different connectors that share

some interfaces: 3 picoblades, 2 debug headers, and 1 contact connector. The debug headers are not used in the flight configuration due to size limitation.

The picoblades are separately utilized for a CAN channel, a JTAG interface, and the application GPIOs. The debug headers provide an easier access for the UART logger, the application SPI, a JTAG interface, debug GPIOs, and a external power supply input for debug. The contact connector is the main interface with the OBC, which provides the application SPI, an I2C for redundant communication, OBC power inputs, and the control GPIOs.

This approach was adopted to improve the design flexibility and allow the adaptation of the payload for different scenarios. For instance, it is planned to perform ground experiments before the payload launch and also the possibility of utilization in different satellite missions.

Power Module

The power subsystem is composed of a DC converter and a simple noise filtering network. In the payload specification is stipulated that a 3.3V power line is reserved for supplying the board, which should ensure current limitation in case of a latch-up. This requirement is accomplished using a anti latch-up circuitry and through the power converter protection as a redundancy.

In order to supply the 1.2V for the FPGA chip, there is a Direct Current (DC) switching power converter (TLV62565DBVR). This step-down converter operates with high efficiency (above 85%), supplying up to 1.5A with overcurrent and thermal protections. The selected part is a 5-Pin SOT-23 chip and features an enable input.

The noise filtering network is the combination of four passive components: two ferrite inductors and two ceramic capacitors. This circuit filters the high frequency noises generated by the switching converter and received from the OBC. Then, there are 1.2V power lines and power grounds before and after the filter, which allows better signal performance for the rest of the circuits.

Latch-up Monitors

The latch-up circuitry consists of four monitors: one for the entire board and three for individually controlling the experiment memories. The circuit is based on the LTC4361ITS8-1 device that provides the required overcurrent protection through the management of a MOSFET transistor (SI1416EDH-T1-GE3). The monitor uses a SOT-8 package and the transistor chip a SOT-363-6 package, which offers a compact and efficient solution. This system is completely independent for opening the circuit in case of non nominal condition, but dependent for closing again. The memory monitors are controlled by the SoC FPGA in case of an event and the general monitor is only controlled by the OBC, since the payload is completely turned off during this condition. The system also provides a status pin that is used for detecting these events.

4 Results

In order to accomplish the payload mission, several simulation, tests and verification were performed. In this document only the most important tests are presented and detailed.

4.1 Subsystem Tests

In order to validate the first concepts, a simulation before the design synthesis was performed intending to verify mainly the SDRAM controller. The Microsemi development tool (Libero) has integration with ModelSim (a simulation software) and provide the Bus Functional Model (BFM) solution, which enables the emulation of the communication between MSS and fabric logic through the AHB. The Figure 28 demonstrates the simulation flow using the BFM scripts. The user defines the commands in the BFM script, then a compiler converts them to a vector file that is further used as an input for the testbench system, which consists of an AHB master interface, the memory controller, and the emulated memory. Since the memory controller is part of the Microsemi catalog, the sources of this testbench scheme are provided to support the early design validation. For this simulation, the script performed explicit write and read operations successively, leaving the task of setup and refresh commands to the memory controller, since it must independently execute this operations.

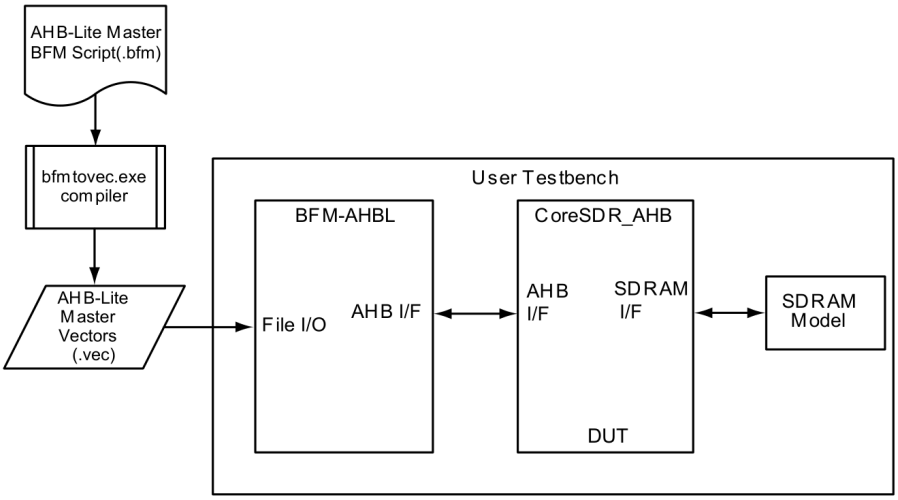


Figure 28: Memory controller simulation flow.

The Figure 29 shows a part of the simulation execution, regarding refresh, load mode, read and write commands. The parameters were selected in conformity with the datasheet nominal values.

During this design evaluation, some misunderstanding of memory parameters were detected and corrected, such as the refresh rate that was smaller than the minimal recommended. Also, this test ensured that the memories could operate in parallel without losing the timing constraints and functional requirements.

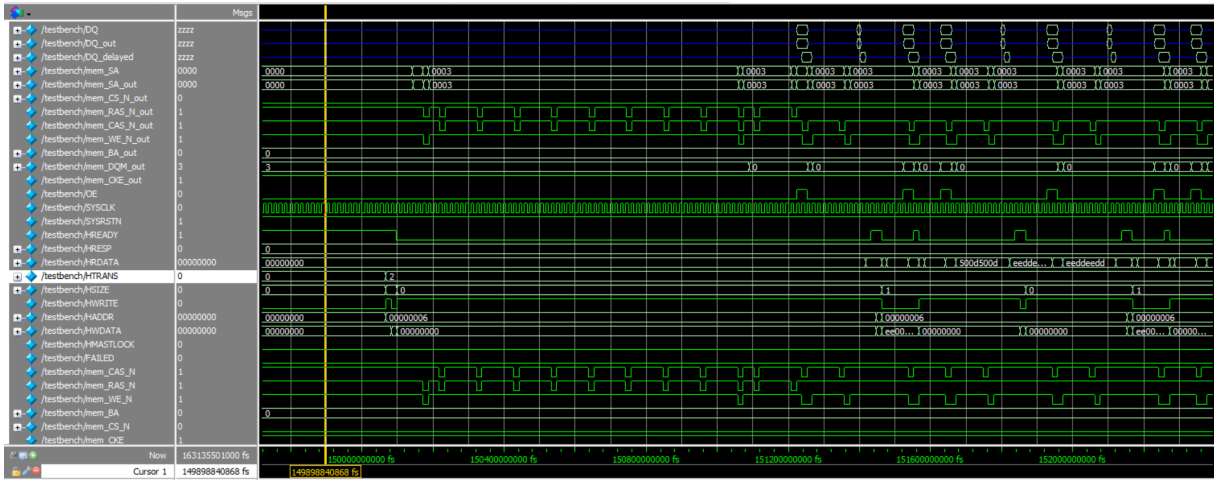


Figure 29: Memory controller simulation results.

4.2 System Tests

During the development and improvements for the flight model, it was possible to evaluate the payload under experimental radiation conditions. Two identical boards were exposed to high energy neutrons during continuous hours before presenting critical damage and reported several events. This experiment allowed improvements in the design (adjustments of the memory controller parameters for this abnormal conditions) and presented some limitations, since this type of radiation was not the main target of the experiment due to high energy exposure. As presented in Figure 30, the payload (mentioned as "HARSH") was one of the several experiments performed in this test campaign.

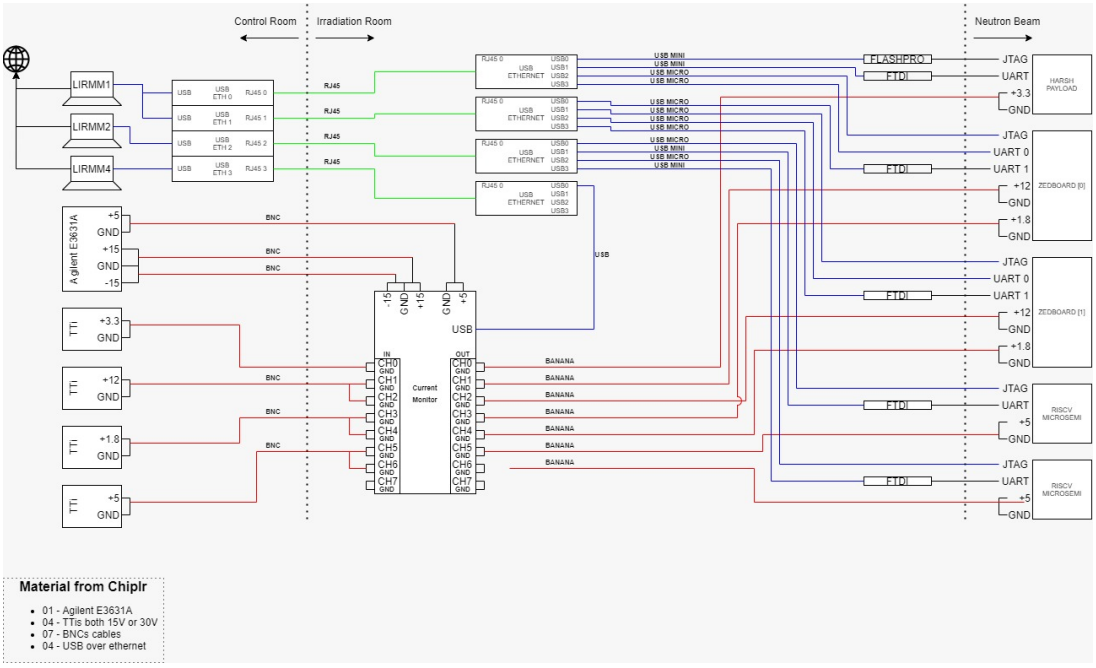


Figure 30: Radiation experiment setup (high energy neutrons beam).

The Figure 31 shows a log message example for this type of test. During the experiment, several log messages were generated and analysed to evaluate performance under radiation and produce actual radiation events, which represents the closest test setup in earth for the payload in the orbit. In this particular log it is possible to detected that there is an error in the last address of the SDRAM memory model B after performing consecutive write and read operations. Since the time difference between operations is too short, in this case there was any event detected despite the one error detected. This error is injected in the last address to verify if in that execution the algorithm is running correctly, since under radiation the device might present some defects.

```
2020-11-12 13:31:23.342274>
2020-11-12 13:31:23.342274>    ========================================================
2020-11-12 13:31:23.357913>    ==== SDRAM Test Devices=3 Size=512Mb Clock=50MHz ====
2020-11-12 13:31:23.357913>    ========================================================
2020-11-12 13:31:23.357913>
2020-11-12 13:31:23.357913>    Frame structure:
2020-11-12 13:31:23.357913>    >    64 cc aa aa aa aa dd dd dd dd lc
2020-11-12 13:31:23.357913>
2020-11-12 13:31:23.357913>    64 -> start of error frame
2020-11-12 13:31:23.357913>     c -> dynamic cycle counter
2020-11-12 13:31:23.373526>     a -> address
2020-11-12 13:31:23.373526>     d -> error data [data read XOR expected data]
2020-11-12 13:31:23.373526>     lc -> line and column code for dynamic test*
2020-11-12 13:31:23.373526>     * -> check README for lc code
2020-11-12 13:31:23.373526>
2020-11-12 13:31:23.389147>    sdram_init(): success
2020-11-12 13:31:23.389147>
2020-11-12 13:31:23.389147>    ========================================================
2020-11-12 13:31:23.389147>    Choose a test:
2020-11-12 13:31:23.389147>     0 -> Write Static
2020-11-12 13:31:23.389147>     1 -> Read Static
2020-11-12 13:31:23.389147>     2 -> Read Retention
2020-11-12 13:31:23.389147>     3 -> Loop C-
2020-11-12 13:31:23.389147>     4 -> March Dynamic Stress
2020-11-12 13:31:23.404775>     5 -> Dynamic E Classic
2020-11-12 13:31:23.404775>     6 -> Dynamic F
2020-11-12 13:31:23.404775>     7 -> Automated test
2020-11-12 13:31:23.404775>     8 -> **Define Max Error Print
2020-11-12 13:31:23.404775>     s -> **Stop Dynamic Test
2020-11-12 13:31:23.404775>
2020-11-12 13:45:14.789588>    Test: 7
2020-11-12 13:45:14.789588>    Enter data:
2020-11-12 13:46:02.012099>    0xFFFFFFFF
2020-11-12 13:46:04.288569>
2020-11-12 13:46:04.292569>    Write static 0xFFFFFFFF from address 0xA0000000 to address 0xA3FFFFFF
2020-11-12 13:46:04.305403>
2020-11-12 13:47:07.298569>    Read static 0xFFFFFFFF from address 0xA0000000 to address 0xA3FFFFFF
2020-11-12 13:47:07.408569>
2020-11-12 13:47:40.254159>    64 00 A3 FF FF FC 5A 5A 5A 5A 00
2020-11-12 13:47:42.761704>
2020-11-12 13:47:42.818555>    Errors = 1
2020-11-12 13:47:42.818555>
```

Figure 31: Log message example during the radiation tests.

4.3 System Integration

The payload was tested using different methods and configurations to achieve its stability and reliability. The first assessments started during the simulations for the memory chips using the ModelSim simulator. Then, the design was validated in the development kit through static tests in its SDRAM memory and due to similar behavioral characteristics from the memories studied in this work, it was possible to evaluate and measure important characteristics of the implementation. After this preliminary tests, using the engineering model, the payload was tested in conjunction with the OBDH and its functional operation was validated. The flight model was assessed alongside the entire satellite system and later qualified for flight. These steps are described in details in the following sections, starting

with the engineering model integration since the previous stages were presented across the document.

The integration with the engineering model occurred in three different steps: the first using a Raspberry Pi as OBC, the second with an older version of the GOLDS-UFSC OBDH (the FloripaSat-I OBDH), and the last using the GOLDS-UFSC OBDH engineering model. In order to perform the first tests, a Raspberry Pi 3 was used to emulate the OBC behavior and communication. This approach was necessary due to limited access to the actual OBDH. Despite the hardware differences, this test setup allowed the debug of the most critical communication issues and the improvement of the interface protocol.

The Figure 32 presents a command request from the OBC during tests of the first stage. It is important to note that the communication between the payload and the OBC uses the FloripaSat Protocol (FSP). In the figure is shown 8 bytes: header, destination address, source address, package type, number of paylaod packages, payload (command in this case) and the last two for CRC.

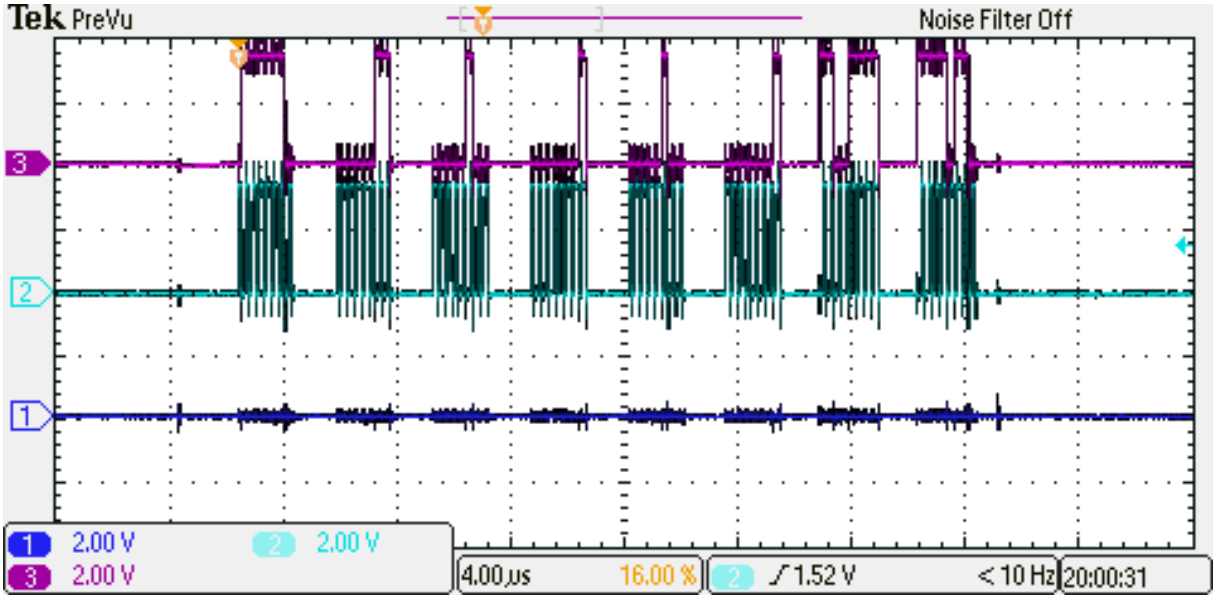


Figure 32: Example SPI command communication signals from the OBC to the payload using the FSP protocol.

The Figure 33 presents the payload acknowledgement answer due to the master request. The same package structure is observed, differing just for the type, payload content, address positions, and CRC check.

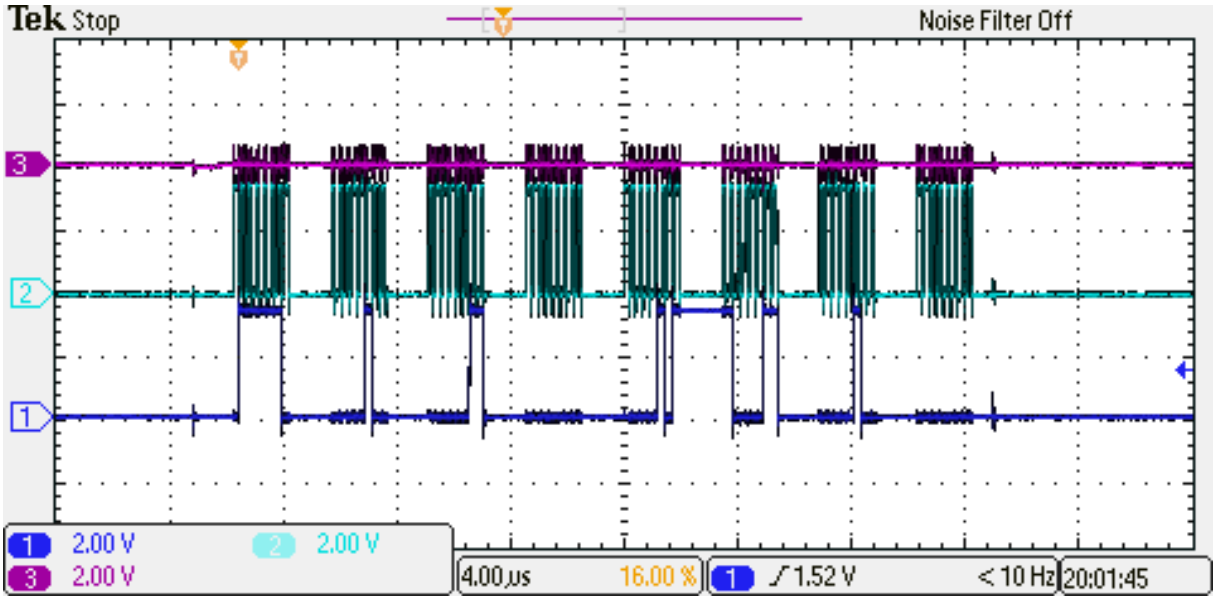


Figure 33: Example SPI acknowledgement answer communication signals from the OBC to the payload using the FSP protocol.

5 Conclusion

The development of this work lead to relevant outcomes for the research laboratory, since the payload is currently being used for different radiation assessments in the memory devices and is planned to be launched alongside the GOLDS-UFSC satellite in a complete space mission. Concerning the technologies involved in the payload, the approach and design presented some innovations in the previous similar missions since some new concepts were employed, such as the use of a RTOS for more robust verification firmware systems and the creation of a multipurpose device in relatively controlled budget.

The final verification and integration tests presented satisfactory overall performance and reliability, but revealed some bottlenecks. In firmware, the payload allowed a robust framework for future utilization and good maintenance. In hardware, the board work properly from the first production, but presented limitations in the memory frequency operation (in comparison with the maximum operation allowed). In the FPGA design, the system performed properly, but more investigation to improve the radiation resilience could increase the reliability of the experiments and its duration before presenting critical failures.

In summary, the payload accomplished its requirements and purpose, brought some innovations to the previous employed methods, presented points for improvement, and provided the necessary tools for personal development and learning experience.

References

[1] National Aeronautics and Space Administration, Mission Design Division. Small Spacecraft Technology State of the Art, NASA/TP–2015–216648/REV1, 2015.

[2] Wiley J. Larson; James R. Wertz. Space Mission Analysis and Design, 1999.

[3] Viyas Gupta. Analysis of single event radiation effects and fault mechanisms in SRAM, FRAM and NAND Flash : application to the MTCube nanosatellite project., 2017.

[4] SpaceLab. GOLDS-UFSC Mission, Acesso em: Setembro de 2020. Disponível em: <golds.ufsc.br/>.

[5] SpaceLab. FloripaSat core on-board computer, Acesso em: Dezembro de 2019. Disponível em: <github.com/spacelab-ufsc/obdh2>.

[6] SpaceLab. FloripaSat-I Mission, Acesso em: Dezembro de 2019. Disponível em: <floripasat.ufsc.br/>.

[7] Altera. What is an SoC FPGA?, 2014.

[8] NASA CubeSat Launch Initiative. Basic Concepts and Processes for First-Time CubeSat Developers, 2017.

[9] L. Luza, D. Soderstrom, H. Puchner, R. Alía, M. Letiche, A. Bosio and L. Dilillo. Effects of Thermal Neutron Irradiation on a Self-Refresh DRAM, 2020.

[10] C. Rigo, L. Seman, M. Berejuck, and E. Bezerra. Printed Circuit Board Design Methodology for Embedded Systems Targeting Space Applications, 2020.

[11] GomSpace. Command and Data handling, Acesso em: Dezembro de 2019. Disponível em: <gomspace.com/shop/subsystems/command-and-data-handling/default.aspx>.

[12] ISIS. Command and Data handling, Acesso em: Dezembro de 2019. Disponível em: <www.isispace.nl/products/command-data-handling-systems/>.

[13] EnduroSat. OnBoard Computer (OBC), Acesso em: Dezembro de 2019. Disponível em: <www.endurosat.com/cubesat-store/cubesat-obc/onboard-computer-obc/>.

[14] PolySat. PolySat Research Laboratory, Acesso em: Dezembro de 2019. Disponível em: <www.polysat.org/>.

A Payload hardware schematics

Rev	Description	Date	Author
1.0	Initial release.	30-Apr-2020	Andre M. P. Mattos
1.1	Review updates.	28-May-2020	Andre M. P. Mattos

Revision History

PCB

1_Interface.SchDoc

FIDUTIALS

FD1 FD2 FD3
FD4 FD5 FD6

MECHANICAL HOLES

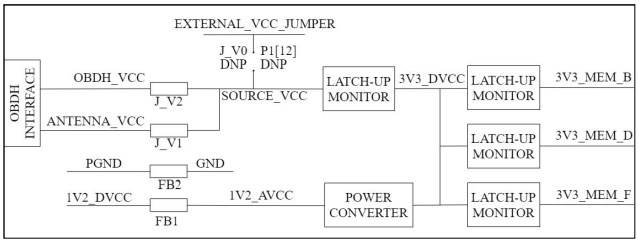
M1 M2 M3 M4
GND

PCB Elements

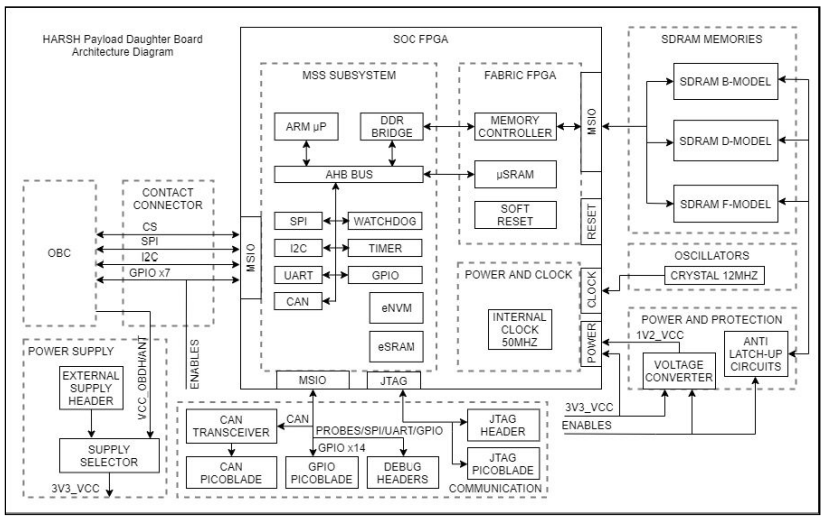
HARSH Payload Daughter Board
(HARSH_Payload) for OBDH 2.0 Hardware

- Drawn by: André Martins Pio de Mattos.
- Based on OBDH 2.0 designed by: André Martins Pio de Mattos.
- Based on DB_Memory designed by: Yan Castro de Azeredo.
- Support: Kleber Reis Gouveia Júnior.

Project Information



Power Diagram



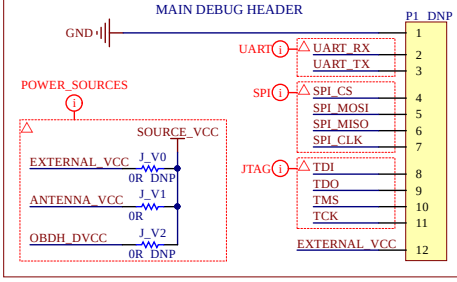
Block Diagram

Title: **HARSH Payload Achitecture**			
Size: A4	Project: HARSH_Payload.PrjPCB	Revision: 1.1	
Date: 6/1/2020	Time: 3:23:15 PM	Sheet 1 of 8	
Drawn By: Andre Martins Pio de Mattos		Model: HARSH_DB	

LIRMM - SPACERADGroup
UFSC - SpaceLab

Institutional partnership project:
research in memory radiation hardness

MAIN DEBUG HEADER



FLIGHT MODEL VERSION: (P1 header not placed)

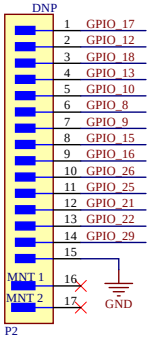
- Power supply: from FloripaSat EPS Antenna regulator place J_V1 or from FSat-II EPS OBDH regulator place J_V2 (ONLY one J_Vx placed)
- Programming: use P3 picoblade connector

OTHER VERSIONS:

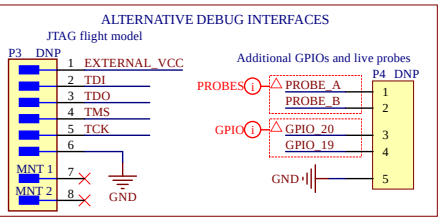
- Power supply: from FloripaSat EPS Antenna regulator place J_V1, from FSat-II EPS OBDH regulator place J_V2 or from external supply place J_V0 (ONLY one J_Vx placed or connected)
- Programming: use P1 header pins (8, 9, 10 and 11)

OBSERVATIONS:
- JTAG programming requires power supply from external supply or OBDH sources

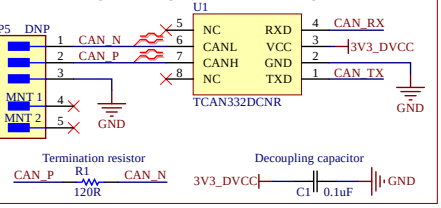
GPIO INTERFACE



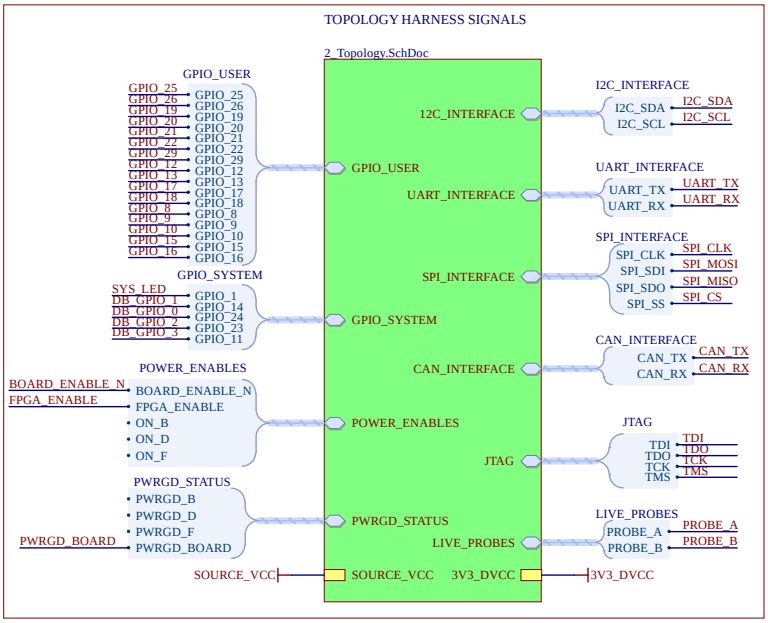
ALTERNATIVE DEBUG INTERFACES



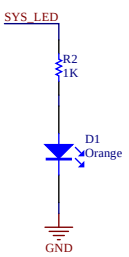
CAN TRANCEIVER AND INTERFACE



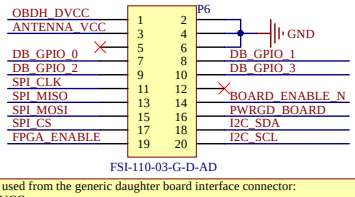
TOPOLOGY HARNESS SIGNALS



SYSTEM LED



INTERFACE FROM OBDH DAUGHTER BOARD CONNECTOR



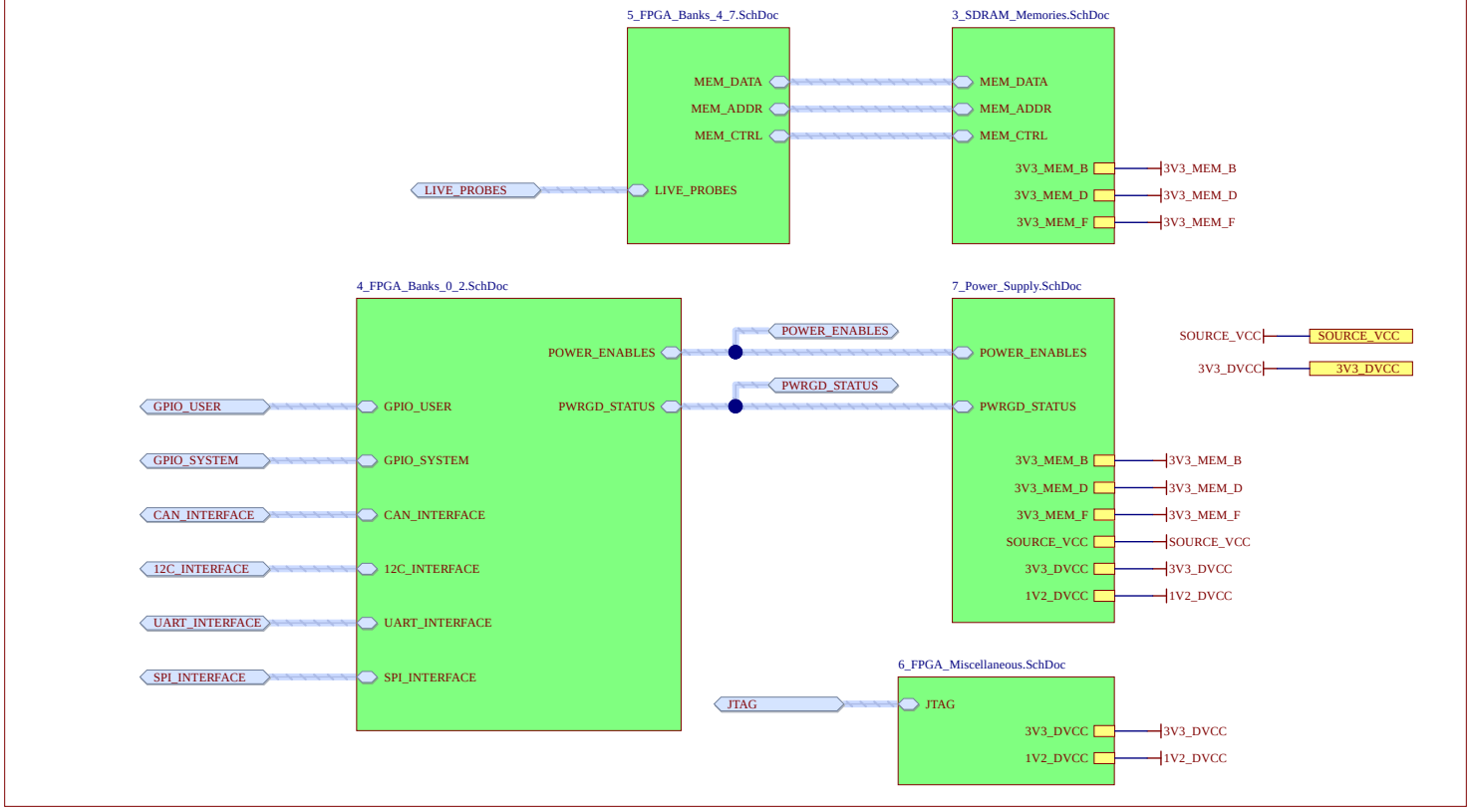
Features not used from the generic daughter board interface connector:
-BATTERY VCC
Features used differently, but compatible, in the generic daughter board interface connector:
- DB_CS2 as GPIO for FPGA enable (Active High)
- DB_ADC0 as GPIO for board "power good" status (feedback of payload power status)
- DB_ADC1 as GPIO for board enable (Active Low)

According to SAMTEC product specification, the maximum body height of the connector when pressed should be 3.28 mm.

Title: **HARSH Payload Interface**			
Size: A4	Project: HARSH_Payload.PrjPCB	Revision: 1.1	
Date: 6/1/2020	Time: 3:23:15 PM	Sheet 2 of 8	
Drawn By: Andre Martins Pio de Mattos		Model: HARSH_DB	

LIRMM - SPACERADGroup
UFSC - SpaceLab

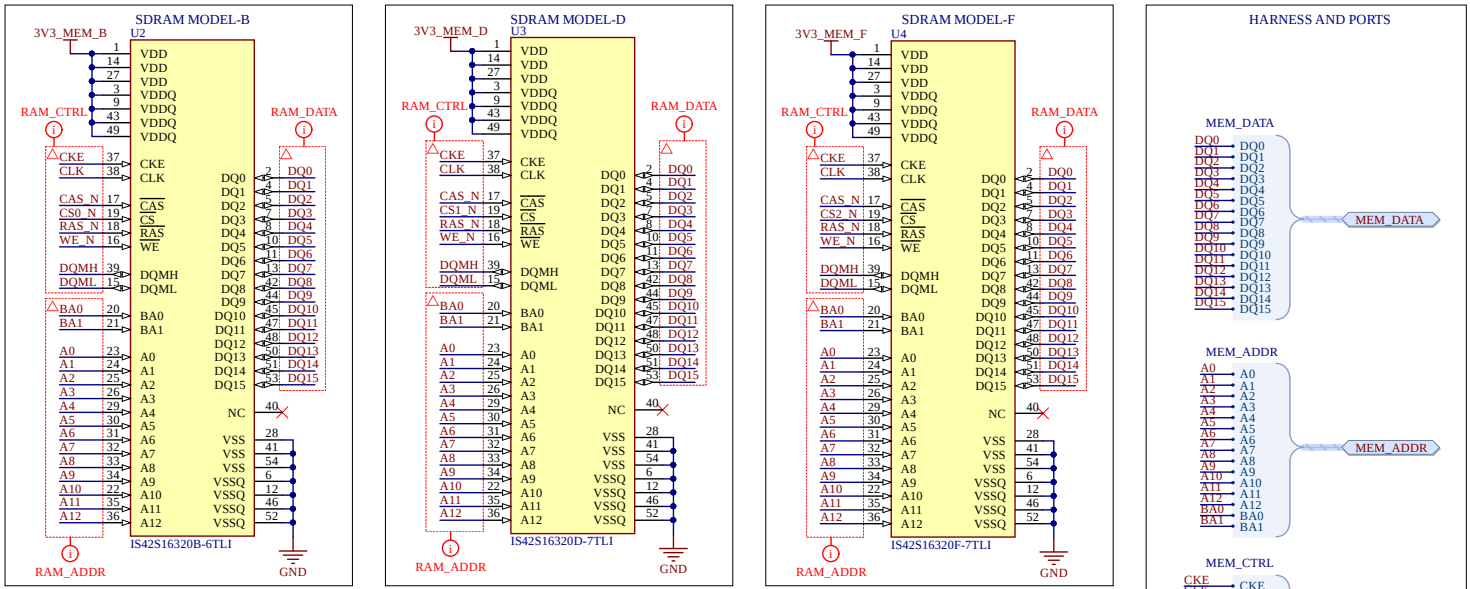
Institutional partnership project:
research in memory radiation hardness



Title: **HARSH Payload Topology**			
Size: A4	Project: HARSH_Payload.PrjPCB	Revision: 1.1	
Date: 6/1/2020	Time: 3:23:16 PM	Sheet 3 of 8	
Drawn By: Andre Martins Pio de Mattos		Model: HARSH_DB	

LIRMM - SPACERADGroup
UFSC - SpaceLab

Institutional partnership project: research in memory radiation hardness



Title: **HARSH Payload SDRAM memories**			
Size: A4	Project: HARSH_Payload.PrjPCB	Revision: 1.1	
Date: 6/1/2020	Time: 3:23:16 PM	Sheet 4 of 8	
Drawn By: Andre Martins Pio de Mattos		Model: HARSH_DB	

LIRMM - SPACERADGroup
UFSC - SpaceLab

Institutional partnership project: research in memory radiation hardness

DDRIO BANK (not used)

U5A

BANK 0

DDRIO29PB0/MDDR_ADDR14	A18
DDRIO29NB0/MDDR_ADDR15	B18
DDRIO30PB0/MDDR_ADDR12	D17
DDRIO30NB0/MDDR_ADDR13	C17
DDRIO31PB0/MDDR_ADDR10	D15
DDRIO31NB0/MDDR_ADDR11	C15
DDRIO32PB0/MDDR_ADDR8	A16
DDRIO32NB0/MDDR_ADDR9	B17
DDRIO33PB0/MDDR_ODT	B16
DDRIO33NB0/MDDR_ADDR7	C16
DDRIO34PB0/MDDR_ADDR5	B14
DDRIO34NB0/MDDR_ADDR6	A15
DDRIO35PB0/MDDR_ADDR3	C14
DDRIO35NB0/MDDR_ADDR4	C15
DDRIO36PB0/MDDR_ADDR1	A13
DDRIO36NB0/MDDR_ADDR2	A14
DDRIO37PB0/MDDR_BA2	D13
DDRIO37NB0/MDDR_ADDR0	D14
DDRIO38PB0/MDDR_BA0	E13
DDRIO38NB0/MDDR_BA1	F13
DDRIO39PB0/MDDR_CLK	B13
DDRIO39NB0/MDDR_CLK_N	B12
DDRIO40PB0/MDDR_RESET_N	E11
DDRIO40NB0/MDDR_CAS_N	E12
DDRIO41PB0/MDDR_CKE	C12
DDRIO41NB0/MDDR_CS_N	C11
DDRIO42PB0/MDDR_RAS_N	D12
DDRIO42NB0/MDDR_WE_N	E11
DDRIO43PB0/MDDR_DQ14	A11
DDRIO43NB0/MDDR_DQ15	B11
DDRIO44PB0/MDDR_DQ12	D10
DDRIO44NB0/MDDR_DQ13	E10
DDRIO45PB0/MDDR_TMATCH_0_IN	E8
DDRIO45NB0/MDDR_DM_RDQS1	F9
DDRIO46PB0/MDDR_DQS1	C9
DDRIO46NB0/MDDR_DQS1_N	B9
DDRIO47PB0/MDDR_DQ10	D9
DDRIO47NB0/MDDR_DQ11	C10
DDRIO48PB0/MDDR_DQ8	A9
DDRIO48NB0/MDDR_DQ9	A10
DDRIO49PB0/MDDR_DQ7	D7
DDRIO49NB0/MDDR_TMATCH_0_OUT	D8
DDRIO50PB0/MDDR_DQ5	A8
DDRIO50NB0/MDDR_DQ6	B8
DDRIO51PB0/MDDR_DM_RDQS0	B6
DDRIO51NB0/MDDR_DQ4	B7
DDRIO52PB0/MDDR_DQS0	C6
DDRIO52NB0/MDDR_DQS0_N	C7
DDRIO53PB0/MDDR_DQ2	A5
DDRIO53NB0/MDDR_DQ3	A6
DDRIO54PB0/MDDR_DQ0	E7
DDRIO54NB0/MDDR_DQ1	F8
DDRIO55PB0/CCC_NE0_CLKI3	E6
DDRIO55NB0	D5
DDRIO56PB0/MDDR_DQ_ECC1	A3
DDRIO56NB0/MDDR_DQ_ECC0	A4
DDRIO57PB0/MDDR_TMATCH_ECC_IN	D5
DDRIO57NB0/MDDR_DM_RDQS_ECC	B3
DDRIO58PB0/MDDR_DQS_ECC	B4
DDRIO58NB0/MDDR_DQS_ECC_N	A1
DDRIO59PB0/GB0	B1
DDRIO59NB0/GB4	C4
DDRIO60PB0/MDDR_TMATCH_ECC_OUT	D3
DDRIO60NB0/CCC_NE1_CLKI3	

M2S010-VF400

MSIO BANKS (Communication and GPIOs)

U5B

BANK 1

MSIO21PB1/GB5/USB_XCLK_C	G17
MSIO21NB1/MMUART_1_TXD/GPIO_24_B/USB_DATA2_C	G16
MSIO22PB1/GB6/MMUART_1_CLK/GPIO_25_B/USB_DATA4_C	E18 GPIO_25
MSIO22NB1/MMUART_1_RXD/GPIO_26_B/USB_DATA3_C	E18 GPIO_26
MSIO23PB1/MMUART_0_RTS/GPIO_17_B/USB_DATA5_C	E15
MSIO23NB1/MMUART_0_DTR/GPIO_18_B/USB_DATA6_C	E16
MSIO24PB1/MMUART_0_CTS/GPIO_19_B/USB_DATA7_C	E17 GPIO_19
MSIO24NB1/MMUART_0_DSR/GPIO_20_B	E16 GPIO_20
MSIO25PB1/MMUART_0_RI/GPIO_21_B	D20 GPIO_21
MSIO25NB1/MMUART_0_DCD/GPIO_22_B	C20 GPIO_22
MSI26NB1/MMUART_0_TXD/GPIO_27_B/USB_DIR_C	D18 UART_TX
MSIO27PB1/MMUART_0_RXD/GPIO_28_B/USB_STP_C	C19 UART_RX
MSIO27NB1/MMUART_0_CLK/GPIO_29_B/USB_NXT_C	B19 GPIO_29
MSIO28PB1/I2C_0_SDA/GPIO_30_B/USB_DATA0_C	A20 I2C_SDA
MSIO28NB1/I2C_0_SCL/GPIO_31_B/USB_DATA1_C	A19 I2C_SCL

M2S010-VF400

U5C

BANK 2

MSIO0PB2	V20 PWRGD_B
MSIO0NB2/USB_DATA7_B	V19 ON_B
MSIO1PB2/USB_XCLK_B	U18
MSIO1NB2/USB_DIR_B	U19 ON_D
MSIO2PB2/USB_STP_B	T18
MSIO2NB2/USB_NXT_B	T19
MSIO3PB2/USB_DATA0_B	T20 PWRGD_D
MSIO3NB2/USB_DATA1_B	R20 ON_F
MSIO4PB2/USB_DATA2_B	P20 PWRGD_F
MSIO4NB2/USB_DATA3_B	N19
MSIO5PB2/USB_DATA4_B	R17
MSIO5NB2/USB_DATA5_B	R18
MSIO6PB2/USB_DATA6_B	P18
MSIO6NB2	P19
MSIO7PB2	R16
MSIO7NB2/CAN_TX/GPIO_2_A/USB_DATA0_A	R15 CAN_TX
MSIO8PB2/CAN_RX/GPIO_3_A/USB_DATA1_A	N20 CAN_RX
MSIO8NB2/CAN_TX_EN_N/GPIO_4_A/USB_DATA2_A	M19
MSIO11PB2/CCC_NE0_CLKI1/I2C_1_SDA/GPIO_0_A/USB_DATA3_A	N16
MSIO11NB2/CCC_NE0_CLKI2/I2C_1_SCL/GPIO_1_A/USB_DATA4_A	M17 GPIO_1
MSIO12PB2/SPI_0_CLK/USB_XCLK_A	P15 SPI_CLK
MSIO12NB2/SPI_0_SDI/GPIO_5_A/USB_DIR_A	N15 SPI_SDI
MSIO13PB2/SPI_0_SDO/GPIO_6_A/USB_STP_A	L20 SPI_SDO
MSIO13NB2/SPI_0_SS0/GPIO_7_A/USB_NXT_A	L19 SPI_SS
MSIO14PB2/SPI_1_CLK	L15
MSIO14NB2/SPI_1_SDI/GPIO_11_A	L16 GPIO_11
MSIO15PB2/SPI_1_SDO/GPIO_12_A	K20 GPIO_12
MSIO15NB2/SPI_1_SS0/GPIO_13_A	J20 GPIO_13
MSIO16PB2/SPI_1_SS4/GPIO_17_A	J17 GPIO_17
MSIO16NB2/SPI_1_SS5/GPIO_18_A	J18 GPIO_18
MSIO17PB2/SPI_1_SS6/GPIO_23_A	K16 GPIO_23
MSIO17NB2/SPI_1_SS7/GPIO_24_A	K15 GPIO_24
MSIO18PB2/SPI_0_SS1/GPIO_8_A/USB_DATA5_A	H19 GPIO_8
MSIO18NB2/SPI_0_SS2/GPIO_9_A/USB_DATA6_A	G18 GPIO_9
MSIO19PB2/SPI_0_SS3/GPIO_10_A/USB_DATA7_A	H17 GPIO_10
MSIO19NB2/SPI_1_SS1/GPIO_14_A	H16 GPIO_14
MSIO20PB2/SPI_1_SS2/GPIO_15_A	G19 GPIO_15
MSIO20NB2/SPI_1_SS3/GPIO_16_A	F20 GPIO_16

M2S010-VF400

HARNESS AND PORTS

GPIO_USER

GPIO_25	GPIO_25	GPIO_USER
GPIO_26	GPIO_26	
GPIO_19	GPIO_19	
GPIO_20	GPIO_20	
GPIO_21	GPIO_21	
GPIO_22	GPIO_22	
GPIO_29	GPIO_29	
GPIO_12	GPIO_12	
GPIO_13	GPIO_13	
GPIO_17	GPIO_17	
GPIO_18	GPIO_18	
GPIO_8	GPIO_8	
GPIO_9	GPIO_9	
GPIO_10	GPIO_10	
GPIO_15	GPIO_15	
GPIO_16	GPIO_16	

GPIO_SYSTEM

GPIO_1	GPIO_1	GPIO_SYSTEM
GPIO_14	GPIO_14	
GPIO_24	GPIO_24	
GPIO_23	GPIO_23	
GPIO_11	GPIO_11	

CAN_INTERFACE

CAN_TX	CAN_TX	CAN_INTERFACE
CAN_RX	CAN_RX	

I2C_INTERFACE

I2C_SDA	I2C_SDA	I2C_INTERFACE
I2C_SCL	I2C_SCL	

UART_INTERFACE

UART_TX	UART_TX	UART_INTERFACE
UART_RX	UART_RX	

SPI_INTERFACE

SPI_CLK	SPI_CLK	SPI_INTERFACE
SPI_SDI	SPI_SDI	
SPI_SDO	SPI_SDO	
SPI_SS	SPI_SS	

PWRGD_STATUS

PWRGD_B	PWRGD_B	PWRGD_STATUS
PWRGD_D	PWRGD_D	
PWRGD_F	PWRGD_F	
	PWRGD_BOARD	

POWER_ENABLES

	BOARD_ENABLE_N	POWER_ENABLES
	FPGA_ENABLE	
ON_B	ON_B	
ON_D	ON_D	
ON_F	ON_F	

Title: HARSH Payload FPGA Banks 0_2			LIRMM - SPACERADGroup UFSC - SpaceLab
Size: A4	Project: HARSH_Payload.PrjPCB	Revision: 1.1	Institutional partnership project: research in memory radiation hardness
Date: 6/1/2020	Time: 3:23:16 PM	Sheet 5 of 8	
Drawn By: Andre Martins Pio de Mattos		Model: HARSH_DB	

MSIO BANKS (memories)

SDRAM data and control signals

U5G

BANK 7

MSIO64PB7	C1	DQ2
MSIO64NB7	C2	DQ12
MSIO67PB7	E6	DQ3
MSIO67NB7	E5	DQ13
MSIO68PB7	D2	
MSIO68NB7	E3	DQ1
MSIO70PB7	E1	
MSIO70NB7	F2	DQ0
MSIO71PB7	F3	DQ10
MSIO71NB7	F4	DQ15
MSIO74PB7	G6	DQ6
MSIO74NB7	F7	CS2_N
MSIO75PB7	G4	CS1_N
MSIO75NB7	F5	DQ7
MSIO78PB7/GB2	G2	DQ11
MSIO78NB7	G3	DQ5
MSIO79PB7/GB1	G1	DQ4
MSIO79NB7	F1	DQ14
MSIO80PB7	H5	DQML
MSIO80NB7	H4	DQMH
MSIO81PB7	J7	DQ9
MSIO81NB7	H6	DQ8

M2S010-VF400

SDRAM address and control signals

U5D

BANK 4

MSIO102PB4	Y10	A5
MSIO102NB4/CCC_NE1_CLKI0	W10	A2
MSIO103PB4/PROBE_A	Y11	PROBE_A
MSIO103NB4/PROBE_B	W12	PROBE_B
MSIO104PB4/GB3	V11	BA1
MSIO104NB4/GB7	U11	A7
MSIO105PB4/CCC_NE0_CLKI0	Y12	A11
MSIO105NB4	Y13	BA0
MSIO110PB4	V12	A9
MSIO110NB4	U12	A12
MSIO112PB4	W13	A10
MSIO112NB4	W14	A8
MSIO113PB4	U13	A0
MSIO113NB4	T13	CAS_N
MSIO114PB4	R13	RAS_N
MSIO114NB4	R12	WE_N
MSIO115PB4	V14	A3
MSIO115NB4	U14	CS0_N
MSIO116PB4	T14	CLK
MSIO116NB4	T15	CKE
MSIO117PB4	V15	A1
MSIO117NB4	Y16	
MSIO118PB4	W15	A6
MSIO118NB4	V15	A4

M2S010-VF400

HARNESS AND PORTS

MEM_DATA

DQ0	DQ0	MEM_DATA
DQ1	DQ1	
DQ2	DQ2	
DQ3	DQ3	
DQ4	DQ4	
DQ5	DQ5	
DQ6	DQ6	
DQ7	DQ7	
DQ8	DQ8	
DQ9	DQ9	
DQ10	DQ10	
DQ11	DQ11	
DQ12	DQ12	
DQ13	DQ13	
DQ14	DQ14	
DQ15	DQ15	

MEM_ADDR

A0	A0	MEM_ADDR
A1	A1	
A2	A2	
A3	A3	
A4	A4	
A5	A5	
A6	A6	
A7	A7	
A8	A8	
A9	A9	
A10	A10	
A11	A11	
A12	A12	
BA0	BA0	
BA1	BA1	

MEM_CTRL

CKE	CKE	MEM_CTRL
CLK	CLK	
CAS_N	CAS_N	
RAS_N	RAS_N	
WE_N	WE_N	
CS0_N	CS0_N	
CS1_N	CS1_N	
CS2_N	CS2_N	
DQMH	DQMH	
DQML	DQML	

LIVE_PROBES

PROBE_A	PROBE_A	LIVE_PROBES
PROBE_B	PROBE_B	

MSIOD BANKS (not used)

U5E

BANK 5

MSIOD100PB5	R1
MSIOD100NB5	R2
MSIOD101PB5	P2
MSIOD101NB5	P3

M2S010-VF400

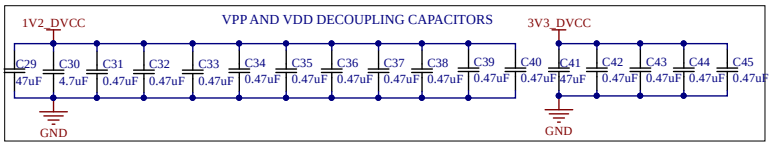
U5F

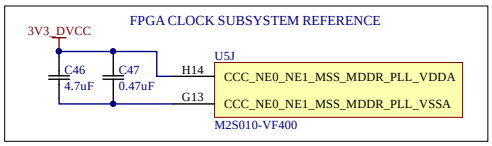
BANK 6

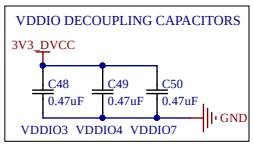
MSIOD82PB6	H1
MSIOD82NB6	H2
MSIOD83PB6	J4
MSIOD83NB6	J5
MSIOD84PB6/CCC_NE1_CLKI2	P1
MSIOD84NB6	P3
MSIOD85PB6/CCC_NE1_CLKI1	K7
MSIOD85NB6	J6
MSIOD86PB6	K5
MSIOD86NB6	K6
MSIOD87PB6	K2
MSIOD87NB6	K3
MSIOD89PB6	L6
MSIOD89NB6	L7
MSIOD90PB6	L4
MSIOD90NB6	L5
MSIOD93PB6	L1
MSIOD93NB6	K1
MSIOD94PB6	M7
MSIOD94NB6	M6
MSIOD95PB6	M3
MSIOD95NB6	L3
MSIOD97PB6	M1
MSIOD97NB6	M2
MSIOD98PB6	N4
MSIOD98NB6	M4
MSIOD99PB6	N1
MSIOD99NB6	N2

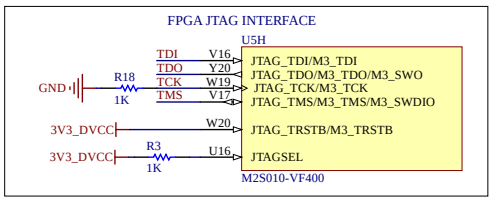
M2S010-VF400

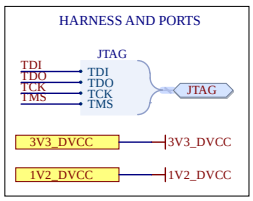
Title: HARSH Payload FPGA Banks 4_7			LIRMM - SPACERADGroup UFSC - SpaceLab
Size: A4	Project: HARSH_Payload.PrjPCB	Revision: 1.1	Institutional partnership project: research in memory radiation hardness
Date: 6/1/2020	Time: 3:23:16 PM	Sheet 6 of 8	
Drawn By: Andre Martins Pio de Mattos		Model: HARSH_DB	

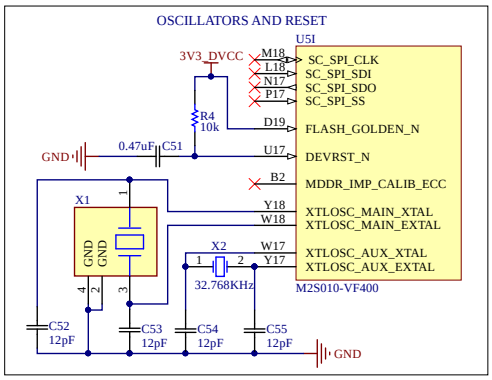


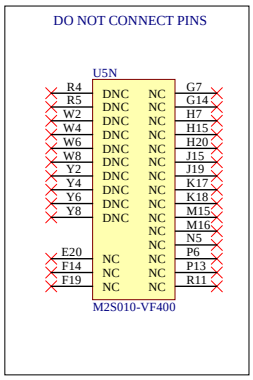


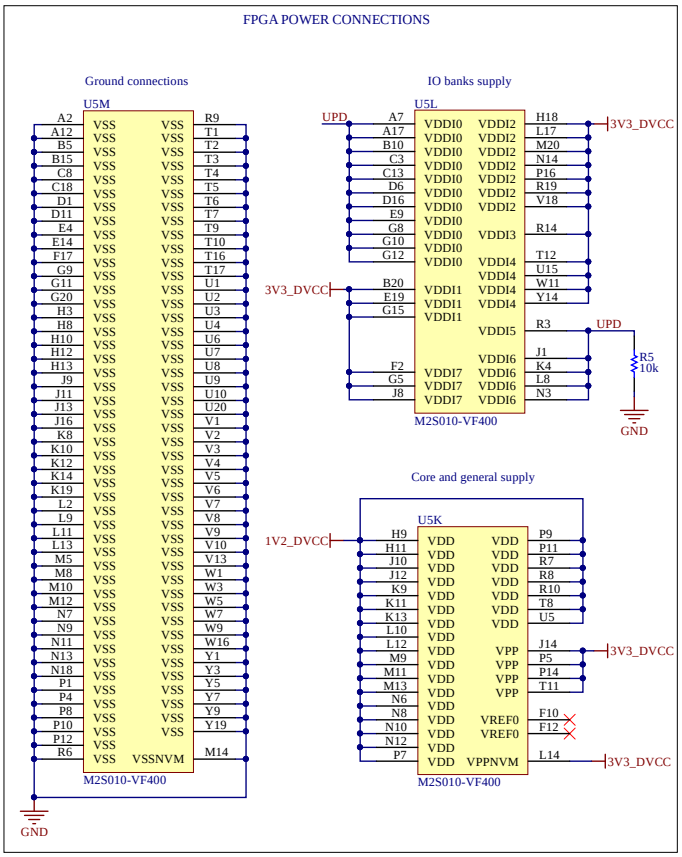






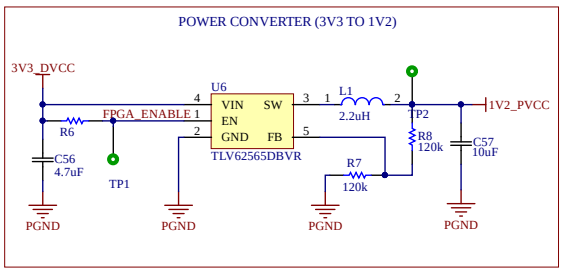


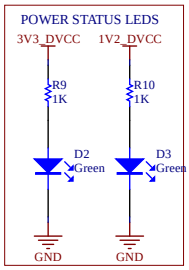


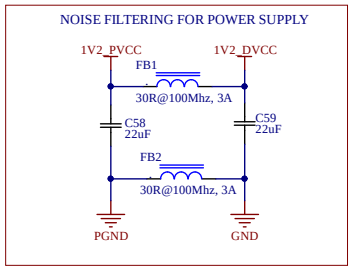


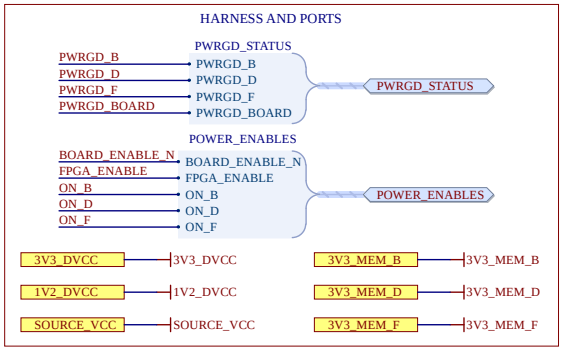
Title: **HARSH Payload FPGA Miscellaneous**			
Size: A4	Project: HARSH_Payload.PrjPCB	Revision: 1.1	
Date: 6/1/2020	Time: 3:23:17 PM	Sheet 7 of 8	
Drawn By: Andre Martins Pio de Mattos		Model: HARSH_DB	

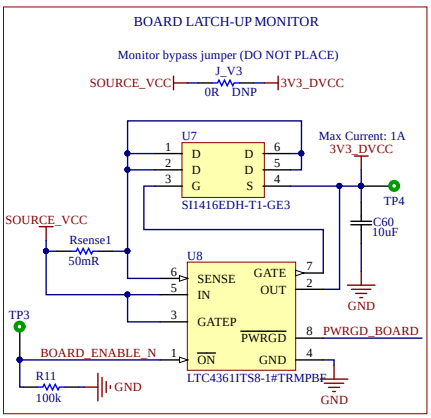
LIRMM - SPACERADGroup
UFSC - SpaceLab
Institutional partnership project: research in memory radiation hardness

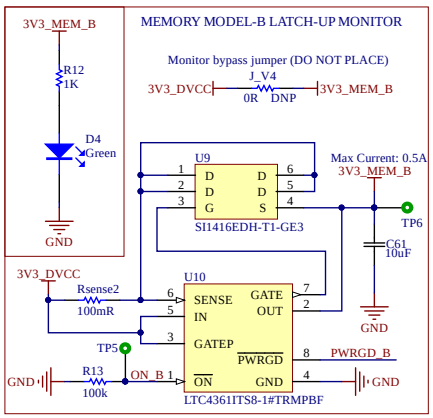


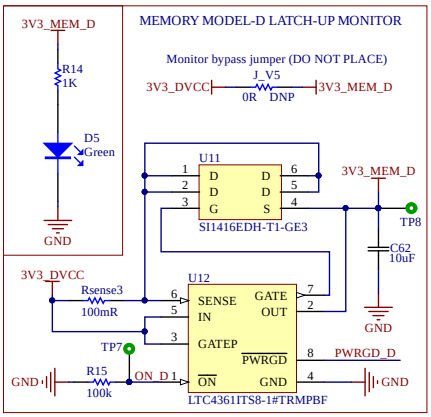


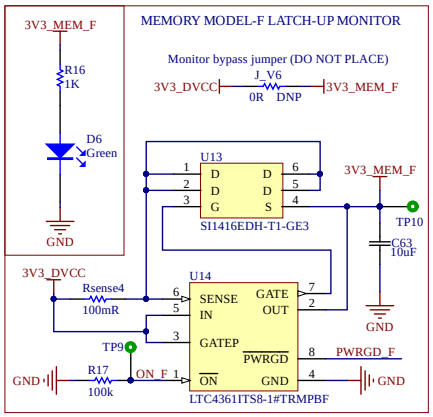






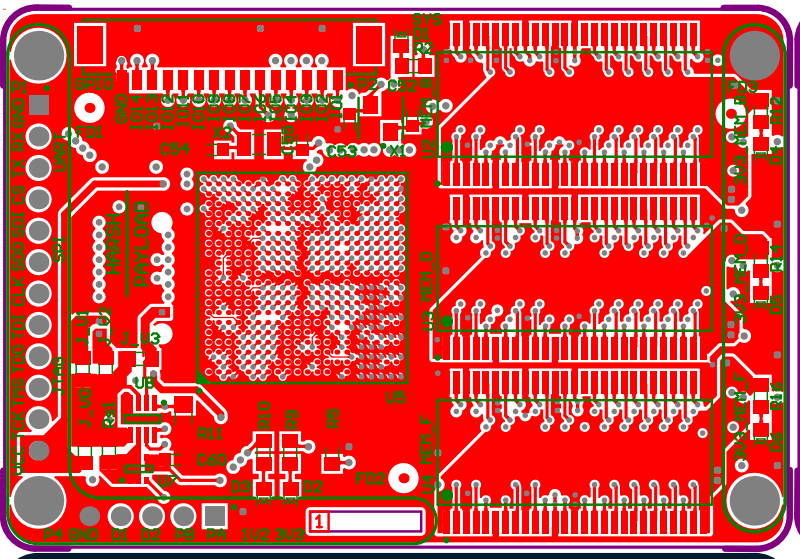


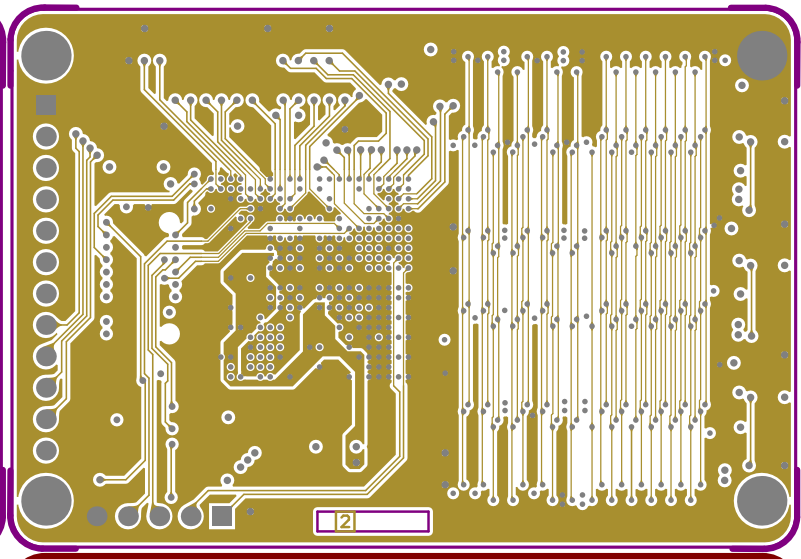


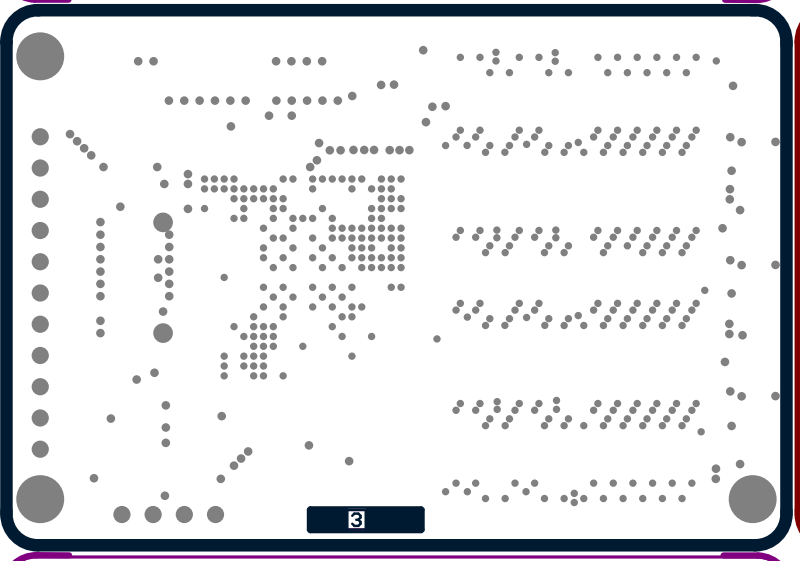


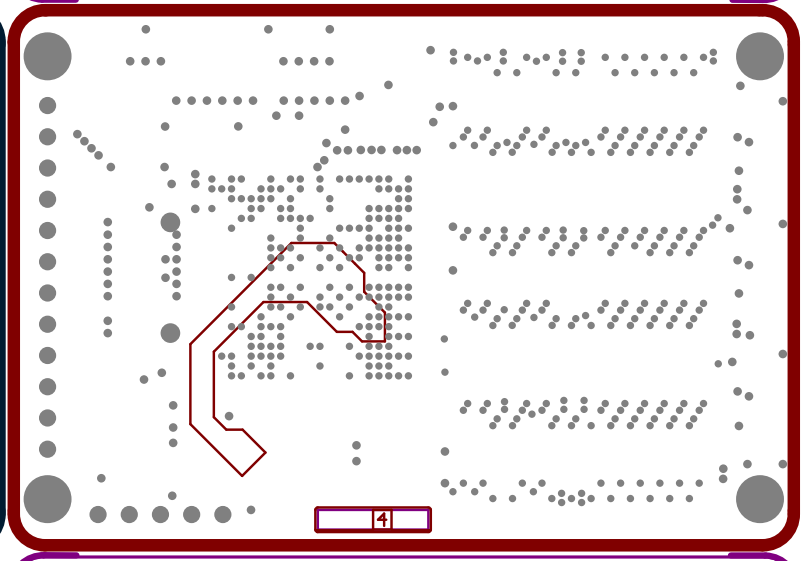
Title: **HARSH Payload Power Supply**			
Size: A4	Project: HARSH_Payload.PrjPCB	Revision: 1.1	
Date: 6/1/2020	Time: 3:23:17 PM	Sheet 8 of 8	
Drawn By: Andre Martins Pio de Mattos		Model: HARSH_DB	

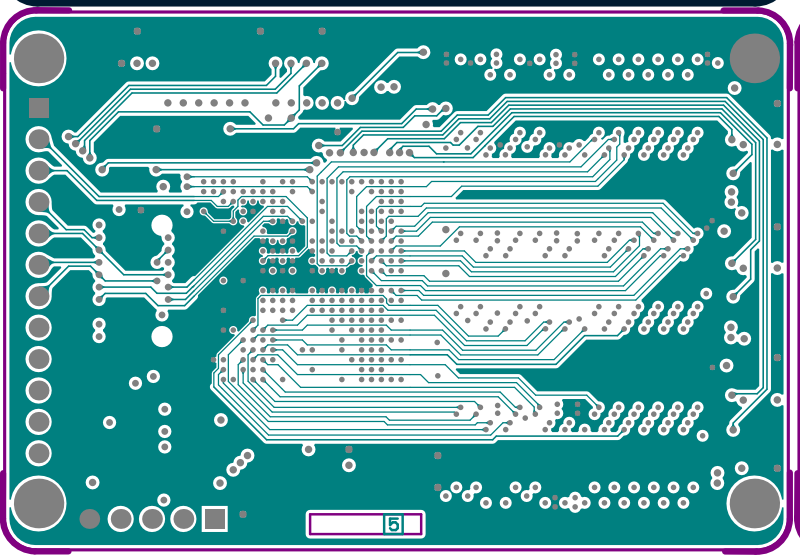
LIRMM - SPACERADGroup
UFSC - SpaceLab
Institutional partnership project: research in memory radiation hardness

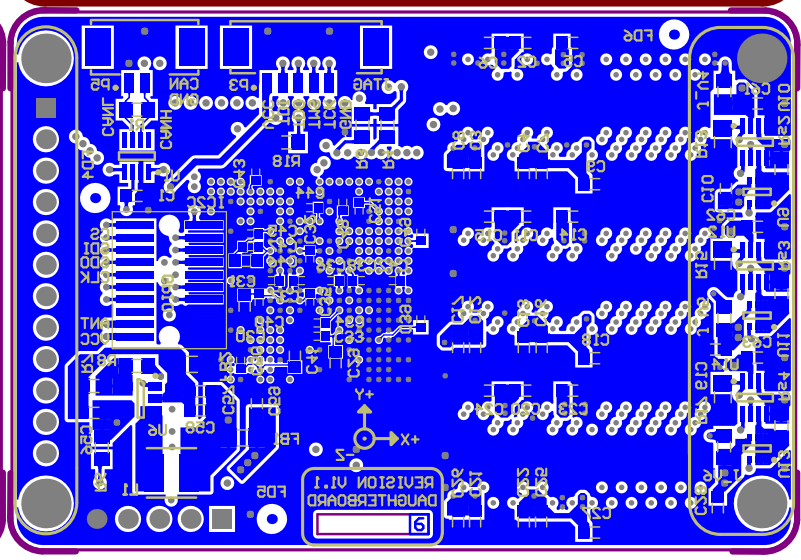

1


2


3


4


5


6