Satellite Stem Cells: The Benefits & Overheads of Reliable, Multicellular Architectures

A. O. Erlank, C. P. Bridges
Surrey Space Centre (SSC)
University of Surrey
Guildford, Surrey, United Kingdom
{a.erlank, c.p.bridges}@surrey.ac.uk

Abstract— While small, low-cost satellites continue to increase in capability and popularity, their reliability remains a problem. Traditional techniques for increasing system reliability are well known to satellite developers, however, their implementation on low-cost satellites is often limited due to intrinsic mass, volume and budgetary restrictions. Aiming for graceful degeneration, therefore, may be a more promising route. To this end, a stem-cell-inspired, multicellular architecture is being developed using commercial-off-the-shelf components. It aims to replace a significant portion of a typical satellite’s bus avionics with a set of initially identical cells. Analogous to biological cells, the artificial cells are able to differentiate during runtime to take on a variety of tasks thanks to a set of artificial proteins. Each cell reconfigures its own proteins within the context of a system-wide distributed task management strategy. In this way, essential tasks can be maintained, even as system cells fail. This paper focuses on two hardware implementations of the stem-cell inspired architecture. The first implementation, based on a single cell, serves as the Payload Interface Computer on a CubeSat named SME-SAT. The second hardware implementation is a benchtop system composed of several cells intended to demonstrate a complete multicellular system in operation. In order to demonstrate the feasibility of these multicellular architectures, the physical attributes of the hardware implementations are compared to those of more traditional implementations and are shown to have enhanced reliability at the cost of increased power and internal bus bandwidth.

However, as is evident from several empirical studies [1,2], these cost-saving measures frequently come at the price of reduced reliability. For many academic institutions developing satellites for educational purposes, reliability is not a major concern, as hearing a single beacon from orbit is often enough to declare mission success. However, now that both commercial and scientific organisations are catching onto the potential of small, low-cost missions, novel solutions for achieving reliability with minimal overheads will be required.

To this end, an alternative system architecture, inspired by multicellular organisms, has been proposed [3,4]. The Artificial Stem Cell Architecture focuses on achieving graceful degeneracy by building complex systems out of initially identical, reconfigurable ‘cells’, instead of unique, discrete subsystems. While previous work has shown its reliability benefits through analytical derivation and simulation [5], this paper focuses on two CubeSat-scale implementations of the Stem Cell Architecture.

Not many attempts have been made to develop a generic hardware building block for satellites. Most research focuses on reuse across missions, instead of reuse on board the same satellite. For example, the CubeSat community has successfully developed a range of subsystems that are generic enough to be used for a wide range of missions [6]. Another example is the Defense Advanced Research Projects Agency (DARPA) Phoenix project which aims to develop a set of several standardised satellite building blocks which can be assembled into a functioning satellite on orbit [7]. AAC Microtec, a Swedish company, has developed a range of satellite building blocks, some of which are purposefully generic enough to be reused within the same satellite. MicroRTUs are described as: “…important building blocks for new subsystems…”, which can “… reduce the development time when they are reused” [8]. However, as far as the authors are aware, no attempts have been made to develop a satellite building block with as much potential for reuse within the same satellite as the cell described in Section 4.

The rest of this paper is structured as follows. Section 2

TABLE OF CONTENTS
1. INTRODUCTION......................................................... 1
2. A RECIPE FOR SUCCESS ......................... 2
3. BIO-INSPIRED RELIABILITY .......... 4
4. MULTICELLULAR CUBESAT AVIONICS.... 6
5. CONCLUSIONS................................. 11
REFERENCES.................................................. 11
BIOGRAPHY ........................................... 11

1. INTRODUCTION
Reliability-increasing techniques, such as component screening, modular and ad-hoc functional redundancy, extensive test campaigns and extensive use of radiation-hardened components, are common practice in the space industry. Unfortunately, many of these techniques involve significant costs and physical implementation overheads, making them undesirable amongst the growing class of small, low-cost satellites. Thus, single string architectures, based on unscreened, commercial-off-the-shelf (COTS) components have become the staple for academic and commercial teams attempting missions on tight schedules and budgets.

978-1-5090-1613-6/17/$31.00 ©2017 IEEE
describes the reliability trends seen amongst a survey of 159 CubeSats. Section 3 gives a brief overview of the Artificial Stem Cell Architecture, before Sections 4 and 5 give details of two implementations. Finally, Section 6 concludes the paper and describes planned future work.

2. A RECIPE FOR SUCCESS

CubeSat missions commonly share two characteristics which make them particularly useful for studying the problem of reduced reliability. Firstly, CubeSat missions often fall on the extreme end of the small, low-cost mission spectrum. Secondly, since many CubeSats are the product of academic institutions, information on their design and performance is readily available through mission websites and academic publications. Thus, in an attempt to derive a “recipe for success”, by finding correlations between satellite characteristics and mission success rates, a survey of all CubeSats launched before 2014 (for which information was readily available) was undertaken. Only those CubeSats that were successfully launched were included. For each of the 159 CubeSats surveyed, data was collected on the following:

Mission Success: Ranked out of 5 and based on information from Swartwout’s online CubeSat database [8]:
1. Dead on arrival in orbit (DOA)
2. Beacon/Downlink received
3. Uplink successful
4. Mission objectives partly completed
5. All mission objectives achieved

Redundancy: Out of a maximum of 3, the number of critical subsystems (OBC, EPS, COMMS) which are redundant. Any form of redundancy is allowed, as long as mission success is still achievable after primary failure.

COTS Subsystems: Out of a maximum of 3, the number of critical subsystems (OBC, EPS, COMMS), which are COTS components.

Attitude Determination and Control: Complexity of the ADCS, ranked out of 4:
0. No ADCS
1. Passive attitude control
2. Active magnetic control
3. Active magnetic + momentum wheel
4. Full 3-axis pointing control

System Architecture: Three system architectures are commonly seen amongst CubeSats:
0. OBC connects to subsystems via separate buses.
1. OBC connects to subsystems via a common, single-master bus (e.g. I2C).
2. Distributed design based on a multi-master bus.

Development Time: Measured from project start to launch. This is not an entirely accurate measure of development time, as launch delays are common. However, teams are able to perform additional testing and software development during the delay, which may enhance their chances of success.

To find correlations, the satellites were split into groups dependent of the characteristic being investigated. For example, when investigating system architecture, the satellites were split into three groups depending on whether they had a Type 0, 1 or 2 architecture. Then mission success rates were compared between these three groups. These comparisons can be seen in Figure 1a-e.

Figure 1a shows the relationship between number of redundant subsystems and mission success rates. The results are largely as expected. The data shows increasing success rates for missions with more redundancy, up to the point where two out of three of the critical subsystems are redundant. At least a mission success level of 2 (successful two-way communication) was achieved by all the satellites with two out of three redundant subsystems. Satellites containing redundancy for all three of their critical subsystems showed a slight decrease in mission success rates compared to those in the previous category, probably because of high complexity.

Figure 1b shows that approximately 35 % of the surveyed CubeSats with no COTS subsystems achieved full mission success, while approximately 16 % of the CubeSats in this category were DOA. CubeSats with a single COTS subsystem fared better, with approximately 55 % achieving full mission success and less than 10 % DOA. However, Figure 1b shows that CubeSats with higher numbers of COTS subsystems fared progressively worse. CubeSats using only COTS subsystems have a full mission success rate of only 8 %, with almost 20 % of them being DOA. This result is unexpected.

The inverse relationship between number of COTS subsystems and mission success rates is likely related to system integration. Inexperienced teams, or teams on a short deadline, may elect to buy all of their subsystems instead of designing them in-house. Due to inexperience, or time pressure, these teams then make mistakes in integration and fail to conduct sufficient system level checks, as mentioned by Swartwout [2], leading to poor performance on-orbit. Teams that developed CubeSats with few or no COTS subsystems have a better understanding of the functioning of their CubeSats. This allows them to make design decisions and revisions to better ensure the correct operation of their CubeSat. Alternatively, these teams have enough experience to know which subsystems are beyond their ability to design successfully and choose only those subsystems to buy.

The complexity of the attitude determination and control system versus mission success rates is depicted in Figure 1c. Initially ignoring the last category, corresponding to
Figure 1. CubeSat reliability survey results. In each case, mission success rates are plotted against a particular satellite parameter: number of redundant subsystems (a), number of COTS subsystems (b), ADCS complexity (c), system architecture (d), and development time (e).

satellites with a full three-axis attitude control system, some clear trends can be seen. As CubeSat ADCS complexity increases, the fraction of DOA CubeSats decreases. This result is unexpected. Once again, a likely explanation is team experience. Inexperienced design teams often elect to implement passive stabilisation systems, or forgo the ADCS entirely. Therefore, those CubeSats with more complex ADCSs were probably built by more technically able teams resulting in fewer DOA CubeSats. A second trend can also be seen: As ADCS complexity increases, mission success rates decrease. This is expected and can be explained by the fact that CubeSats with more complex ADCSs likely have more complex mission objectives. More complex mission objectives are naturally more difficult to achieve, leading to lower success rates. The exception is the final category, of CubeSats with full three-axis control systems showing high success rates. CubeSats in this category were likely built by experienced teams who understood the complexity of their missions.

Figure 1d shows that CubeSats with Type 2 system architectures had the highest success rates, while those with Type 1 system architectures fared the worst. Type 1 architectures are composed of a central OBC which is connected to the other subsystems through a central bus. A failure of the OBC renders the rest of the satellite inoperable as the subsystems are unable to communicate with each other directly. In addition, poorly designed subsystems can hang-up the bus during a failure, preventing all internal communication. Without internal communication, ground operators can do nothing to troubleshoot the situation. Due to these vulnerabilities, CubeSats with Type 1 architectures have performed poorly on orbit. Approximately 40% of the surveyed CubeSats with Type 1 architectures were never heard from once on orbit.

Figure 1e, depicting mission success versus development time, reveals an unexpected trend. It shows a decrease in mission success rates as development time increases. CubeSats developed in under a year show a mission success rate of more than 65%, while CubeSats that took six years or longer to develop show a mission success rate of under 20%. The two most probable causes of this trend are team inexperience and mission complexity. Inexperienced teams require development time to set up infrastructure and may design overly ambitious missions. On the other hand, through descoping, experienced teams can construct successful missions in relatively short timelines.

While this survey uncovered interesting trends, it largely highlights the fact that mission success is the result of a complex interaction between many factors, several of which, such as team experience, development infrastructure, and mission complexity, are difficult to measure. Simple reliability assumptions, such as COTS subsystems are more reliable, or more redundancy is always better, are shown to be false when taken in isolation. Thus, there is no simple “recipe for success” found to date.
3. BIO-INSPIRED RELIABILITY

Biological Cell

Artificial Cell

Figure 2. The differentiation process in biological and proposed artificial cells.

By extensively using unscreened COTS components and performing only limited test campaigns, low-cost satellite developers must accept that component failures during mission lifetime are a possibility. While traditional forms of redundancy could mask these failures to a certain extent, their implementation overheads are prohibitive. Thus, graceful degeneracy may be an acceptable middle ground between single-string and full system redundancy.

A system that gracefully degrades may lose capability after each successive failure, but can ensure that essential tasks remain running, even if only at reduced capacity. Graceful degeneracy is most effective if it is a core system design goal. Thus the Artificial Stem Cell (ASC) architecture was proposed with this goal in mind [3,4].

The ASC architecture was inspired by three features of biological multicellular life: differentiation, cell membranes and peer-to-peer task management.

Differentiation is the process through which cells adopt specialisations. In a simplified sense, biological organisms start out as a set of initially identical stem cells. Over time, these stem cells undergo a process known as differentiation, during which the cells physically change to take on all the specialisations required for the organism to operate as a whole. Similarly, a system based on the ASC architecture is envisioned to be based on a set of initially identical ‘black box’ subsystems, or artificial cells, which can differentiate to perform a wide variety of roles within the system. During mission lifetime, failures can be compensated for through the redifferentiation of lower priority cells.

As shown in Figure 2, the biological differentiation process is based on three main components, namely, DNA, macromolecular machinery, and proteins. The DNA stores instructions for producing all the proteins required by the organism. The macro-molecular machinery responds to internal and external conditions by reading different sections of the DNA and producing the relevant proteins. These proteins then go on to perform the majority of the cellular functions. Therefore, the set of proteins present in the cell at any one time determines its capabilities, or specialisation, and can change over time.

The ASC architecture proposes an artificial version of this process. As shown in Figure 2, in an artificial cell, the role of DNA is played by a set of non-volatile memory. A microcontroller (MCU), acting as the macromolecular machinery, responds to internal and external conditions by reading pieces of firmware from the non-volatile memory, and programming it into a set of MCUs representing blank proteins. The protein MCUs, each running different firmware, then perform various system tasks.

Cell membranes form a protective barrier around biological cells, regulating contact with the cell internals. Similarly, the proposed artificial cells interact with the outside world through specially designed generic input/output (I/O) circuitry, which allows the protein MCUs to interface to a large variety of peripherals (such as high speed data buses, analogue sensors, or high power actuators), while protecting them from unexpected electrical environments.

Finally, some form of distributed task coordination is present amongst the cells in all multicellular organisms. In complex multicellular organisms, such as mammals, coordination of bodily functions is largely centralised through the brain and nervous system. This is analogous to the central OBC and common bus architectures of many satellites. However, simpler multicellular organisms, such as sea sponges, have no central nervous system and all coordination is achieved at a cellular level through peer-to-peer communication. For reliability purposes, such a peer to peer strategy is preferred, as it removes the potential single points of failure presented by centralised control. Thus, the ASC architecture proposes a middleware, based on the concept of Agent Computing [9], which runs directly on the cells and enables peer-to-peer communication and task coordination.

The following sections of this paper describe two implementations of the ASC Architecture. The first implementation, based on a single cell, forms the Payload Interface Computer of the SME-SAT CubeSat. Due to the requirement to limit the experimental nature of this subsystem, only a subset of the ASC architecture’s features were implemented. However, the second implementation, described in Section 5, aims to demonstrate a complete CubeSat-scale multicellular system.
4. SME-SAT Payload Interface Board

Figure 3. SME-SAT CubeSat CAD designs. The SME-SAT cell is mounted above the purple CMG Array.

SME-SAT is a 3U CubeSat being developed at the Surrey Space Centre (Figure 3) [10]. SME-SAT aims to space-rate a number of commercial payloads from third-party small and medium enterprises (SMEs). These payloads include:

- Three-axis, high accuracy microelectromechanical system (MEMS) inertial measurement unit - Sensonor.
- Miniature, science-grade fluxgate magnetometer - Laboratory of Electromechanical Innovation (LEMI).
- High accuracy MEMS accelerometers - Theon Sensors.
- CubeSat star tracker - Innovative Solutions in Space (ISIS).
- Miniature control moment gyroscope (CMG) array - Surrey Space Centre.
- Smart thermal radiator tiles - MPB Communications Inc.

The original design for SME-SAT required two processing boards to designed. One would be tasked with interfacing to the majority of the third-party payloads, operating the payloads on orbit, and collecting experiment data for later transmission to the ground. The second processing board would be responsible for driving and operating the CMG array.

The 4 proteins of the SME-SAT cell are required to interface to a large variety of payloads.

During early work on the ASC architecture, it was determined that the required payload interface computer and CMG array driver boards could be ‘cellularised’.

The new design can be seen in Figure 4. A single cell composed of four proteins would perform both roles. Each protein is interfaced to a single external payload and a single CMG of the array. The proteins are networked to each other and the satellite OBC through a Controller Area Network (CAN) bus. Each protein was designed to have identical hardware, based on an ARM Cortex M0 MCU and 1 Mbit EEPROM. The new design has the following advantages:

- Simple hardware design duplicated – low-pin count MCU interfaced to a single external payload and a single CMG.
- Simplified firmware development – no multitasking operating system required, as each protein operates only a single payload.
- Graceful degeneracy in experiment output – protein failure would cause the loss of only a single payload and CMG. The others would remain unaffected.
- Flexible power consumption – individual proteins can be switched on individually as required, lowering average power consumption.

The cell-based design was developed by a single engineer over the course of a year, while the firmware was written by two engineers over six months. The flight hardware can be seen in Figure 5. It consists of a PC104-sized carrier board supporting four identical proteins.
Figure 5. The flight version of the SME-SAT cell mounted to the CMG Array, and a spare protein.

This assembly is mounted to the underside of the CMG array. Each protein operates at 48 MHz and consumes approximately 66 mW.

No ASC middleware or generic I/O circuitry was implemented to minimize the experimental nature of this critical subsystem. While the proteins are capable of communicating directly with one another, and even reprogramming one another, it is envisioned that the proteins will mostly be commanded and, if necessary, reprogrammed directly from the OBC. Thus, this cell lacks integrated macromolecular machinery.

The proposed generic I/O circuitry of the ASC architecture is replaced by the more traditional discrete, purpose-made interface ICs. These ICs are used to drive the gimbal and flywheel motors of the attached CMG, and to interface to the communication bus of the external payload. Initially, cross-strapping payloads across proteins was considered, but the added complexity of the switching circuitry was deemed unnecessarily complicated. The loss of individual payloads could be accepted, provided the others remained unaffected.

The SME-SAT mission is currently completed at the soft stack stage while awaiting environmental tests and launch.

Figure 6. CubeSpace COTS CubeSat ADCS [11].

4. MULTICELLULAR CUBESAT AVIONICS

The theoretical aim of the ASC architecture is to completely replace traditional satellite subsystems with cells. However, in practice, there is a limit to what can be ‘cellularised’. Thus, as an initial demonstration of a multicellular ASC architecture replacing traditional avionics, a benchtop prototype is being developed to replicate the capabilities of a COTS CubeSat attitude determination and control system (ADCS), named CubeADCS [11].

CubeSpace CubeADCS Solution

The CubeSpace CubeADCS is an integrated ADCS solution for CubeSats (Figure 6). An ARM Cortex M3 MCU estimates attitude using input from a MEMS gyroscope, magnetometer, six coarse sun sensors, and fine sun and horizon sensors. Attitude is controlled using three magnetoquers and a single momentum wheel. The CubeADCS can operate in various modes, including detumble, pure magnetic control and momentum-biased control.

It has flown on the two QB50 precursor missions, and is expected to fly on a number of QB50 CubeSats [11].

The CubeADCS is composed of a stack of three PC104-sized PCBs, named CubeComputer, CubeControl and CubeSense. CubeComputer contains the main processor, CubeControl contains two MCUs for interfacing to and driving the sensors and actuators, and CubeSense features two CMOS cameras for fine sun and horizon sensing. CubeComputer controls the other two boards over an I2C bus. The bundle can interface to the rest of the satellite using I2C or CAN.

Due to its range of operating modes, useful for implementing graceful redundancy, and wide range of peripherals, the CubeADCS is an interesting target for cellularisation. The rest of this section describes the design and implementation of a cell-based ADCS and compares its performance and overheads to the CubeADCS.
Cell-based ADCS Design

Advancing the SME-SAT cell design, the ADCS cell design can be seen in Figure 7a. Each cell measures 10 x 11 cm and is composed of four proteins, which are interlinked by two CAN buses and an I2C bus.

Unlike the SME-SAT cell, one protein of the ADCS cell is always required to act as the cell’s macromolecular machinery (MM). This role can be played by any of the proteins, ensuring the MM does not present a potential single point of failure.

Each protein contains a discrete I2C node, which controls power switches to various parts of the protein, can monitor for overcurrent conditions, and can place the MCU into bootloader mode. Through these I2C nodes, the MM has a level of control over the proteins on its cell.

The MM is responsible for monitoring the health of its proteins and for reprogramming them if failure or changing environmental or operating conditions call for it. The MM has access to 4 megabits of I2C EEPROM, which acts as the cell’s DNA. This EEPROM is able to store sixteen 32kB firmware images, which can be read by the MM and programmed onto the proteins. Programming is achieved over the internal CAN bus via bootloaders on the proteins.

Proteins on the same cell can communicate with each other and the MM directly using the internal CAN bus. Only the MM has access to the external, inter-cell CAN bus. Thus, in order for a protein to communicate with a protein on another cell, the packets must be routed from the internal CAN bus to the external CAN bus by the MM.

MM failure is detected by dedicated watchdog circuitry on the I2C bus. Watchdog timeout causes a hardware signal to be sent to all proteins and the activation of a CAN bus bridge. The first protein to detect the failure signal enters bootloader mode and waits to be reprogrammed into the new cell MM. The CAN bus bridge connects the internal and external CAN buses, allowing a neighbouring cell to reprogram the bootloader-state protein with the MM firmware. This process is akin to the birth of a new cell.

Unlike the SME-SAT cells, the ADCS cells contain generic I/O circuitry. Each protein has six of its own GPIO lines. The number of lines was limited by the available PCB area and pin count of the MCU. Each line can operate at 3.3 – 9V and can be used as a high speed digital input or output for interfacing to communication buses, a high current driver or sinker for directly driving actuators, or as an analogue input. The output stage is based on a half-H-bridge capable of handling up to 1A, while the input stage is based on a Schmitt trigger and instrumentation amplifier. The instrumentation amplifier can either measure analogue input signals, or the current being handled by the output stage. In addition to I/O functions, each GPIO line can also be placed into a high impedance state. This state allows peripherals to be cross-strapped between proteins.

Figure 7. Full ASC architecture cell hardware (a) and middleware (b) architectures.

Custom-developed middleware is responsible for enabling system-wide inter-cell and inter-protein cooperation. The structure of the ADCS cell middleware is shown in Figure 7b. It is based on the principle of Agent Computing, with each cell operating as its own agency [12]. The middleware is composed of two parts: a template for agents that is compiled along with user code into firmware images and eventually programmed onto the proteins, and a program which performs the MM duties and is executed on the MM protein. Both parts are based on the open source real-time operating system FreeRTOS.

To aid future compatibility with other satellites and ground stations, the middleware is based on the Federation for Intelligent Physical Agents (FIPA) Abstract Architecture [13]. As seen in Figure 7b, the MM performs the roles of the agency, offering services such as lifecycle management and a searchable directory. Unlike in most agent environments where agents are executed within a virtual machine or interpreted environment, the ADCS cell runs agents as native code on the discrete protein MCUs. This has the advantage of reduced code overhead, improved execution efficiency, and security, but at the cost of cross-platform support.
To imitate the functionality of the CubeADCS, excluding CubeSense, two ADCS cells are required, as shown in Figure 8. It was deemed that implementing CubeSense, too, was unnecessary for this research’s purposes.

Six ADCS tasks were identified and each given a priority according to three modes of operation, representing degrees of graceful degeneracy. The tasks and their priorities can be seen in Figure 8. The most basic mode, detumbling, is given a priority of 3 (highest), and requires three tasks (Torquer Control, Magnetometer Interface, and Main/Estimator) as well as magnetorquer and magnetometer peripherals. The next mode, magnetic control, is given a priority of 2 and requires two additional tasks (Orbit Propagator and Coarse Sun Sensor Interface), as well as the coarse sun sensor peripheral. Finally, the most capable mode is given a priority of 1 (lowest) and requires one additional task (Wheel Control) and the momentum wheel peripheral.

Ideally, every protein should be able to perform any of the tasks, requiring a connection to every peripheral. In reality, however, the number of generic I/O lines limits each protein to a single peripheral interface. Thus, tasks that require no peripheral can be performed by any protein, while those that require a particular peripheral can only be performed by proteins interfaced to the relevant peripheral. The high impedance mode of the generic I/O lines allows the same peripheral to be interfaced to several proteins simultaneously, allowing arbitrary levels of cross-strapping.

It is sensible to relate the level of cross-strapping to the priority of the peripheral as far as possible. In this case, the magnetometer is deemed the most important, while the momentum wheel, which is only required in the most capable operational mode, is least important.

Figure 8. Duplicating the CubeADCS functionality using two 4-protein cells with cross-strapped peripherals. Tasks can be reallocated as proteins fail.

For the cell-based design, all parts of the ADCS system are duplicated, as shown in Figure 9. At least two proteins are required, one responsible for controlling the momentum wheel and the other for controlling the magnetorquers. The reliability block diagram is maintained as per the full CubeADCS design, with each cell having a MM and MAG interface.

Figure 9. Reliability block diagrams of the CubeADCS operating in momentum-based mode (a), magnetic control/detumbling mode (b), and the equivalent cell-based design (c). C- common, S- supporting components.

Reliability Analysis

With the design of the cell-based ADCS complete, it is possible to perform simulations to compare its predicted lifetime to that of the CubeADCS.

In its most capable mode, the CubeADCS is essentially a series system, as shown in Figure 9a. CubeControl contains two MCUs, one of which drives the momentum wheel, while the other handles the magnetometer and magnetorquers [14]. The common and support blocks represent support circuitry and buses on and between PCBs.

In its two reduced modes, magnetic control and detumble, the wheel is not required. To provide some redundancy, the magnetorquer and magnetometer are controllable from either MCU. Thus, the reliability block diagram is modified to reflect this partially parallel structure, as shown in Figure 9b.

In comparison, the cell-based ADCS can be described as a modified k-out-of-n architecture, as shown in Figure 9c. However, this representation does not take into account the need for an MM on each cell and the restrictions imposed by peripheral interfaces. A thorough analytical analysis of the case involving no peripherals is given in [5]. However, to include the effects peripheral connections, Monte Carlo simulations are employed.

Figure 9 shows a comparison between the reliabilities of the CubeADCS and cell-based ADCS when operating in three different modes, based on 1000 simulations. Only two plots appear for the CubeADCS as it has identical reliability in magnetic-control and detumble modes. The failure models used in the simulation consisted of Weibull distributions for all components. The shape parameter of 0.4464 was derived from an empirical study of on-orbit satellite reliabilities [1].
The scale parameters for the MCUs/proteins, common and support components ($35e^3$, $35e^8$, $35e^9$ years) were adjusted to give the CubeADCS an expected lifetime of approximately one year (its design lifetime), while taking into account their relative complexities.

The reliability plots of Figure 9a show that when operating in their most capable modes (momentum biased), the CubeADCS is approximately 35% more reliable than the cell based ADCS after three years. However, when operating in lower modes, such as magnetic control or detumble, due to specific failures or mission requirements, the cell-based ADCS has a reliability advantage. This advantage can be increased by adding an additional cell, as shown in Figure 9b. No significant system redesign is required. The peripherals are simply cross-strapped to locations on the new cell, and the middleware will distribute tasks to the new cell as required. The three-cell ADCS is expected to be more reliable than the CubeADCS in all modes throughout its lifetime (see summary Table 1).

However, these reliability benefits come with certain overheads, which are discussed in the remainder of this section.

**Implementation Results and Overheads**

Two cells compromising the cell-based ADCS were manufactured, as seen in Figure 11. Faults can be introduced into the system using the toggle switches, which disconnect power to individual proteins, or by sending commands to the protein power switches over the internal I2C buses.

The proteins are based on ARM Cortex M0 MCUs capable of 0.84 DMIPS/MHz and a maximum clockrate of 48 MHz [15]. Since one protein per cell is devoted to MM tasks and unavailable for running user tasks, the 2-cell-based ADCS has a maximum processing capacity of $48 \times 0.84 \times 6 = 241.92$ DMIPS. In comparison, the CubeADCS has a single ARM Cortex M3 MCU clocked at 48 MHz and is capable of 1.25 DMIPS/MHz [16]. In addition, CubeControl has two 8-bit PIC18F MCUs each clocked at 11 MHz and capable of only 0.02 DMIPS/MHz [17]. Combined, the CubeADCS has a total processing capacity of 60.44 DMIPS. Therefore, the cell-based ADCS has approximately four times more processing capacity than the CubeADCS.

However, the cell-based ADCS consumes significantly more power than the CubeADCS. At full capacity, the cell-based ADCS consumes 0.78 W, while the CubeADCS consumes only 0.35 W. The power consumption breakdown can be seen in Figure 12. 32% of the power consumption is attributable to the operation of the dual CAN buses. Furthermore, with only four proteins per cell, the MM overhead is substantial, at 16%.

To reduce power consumption, the proteins can be clocked down to 12 MHz, giving the cell-based ADCS a processing capacity that is almost identical to that of the CubeADCS (60 vs 60.44 DMIPS). This reduces the power consumption by 20.51% to 0.62 W.

Each protein has 32 KB of flash for code storage and 8 KB of RAM. The middleware agent template is based on FreeRTOS V7.1.0 [18] and, with size optimization enabled, compiles to 24.9 kB (78% of flash).
The MM firmware, also based on FreeRTOS, compiles to 28.7 kB (90% of flash). Therefore, the cell-based ADCS, containing six proteins for user tasks, has a total of just 42.6 kB available for user code. However, the extensive libraries contained within the agent template, including libraries for generic I/O access and peer-to-peer communication, should allow the developer to focus on high level functionality.

Peer-to-peer communication between proteins takes two forms. The first form follows the specifications of the FIPA Agent Communication Language (ACL) [13]. While this format allows compatibility with other FIPA agents, it comes with overheads. Considering the message content of ACL messages has been restricted to 11 bytes and a typical ACL message is 70 bytes long, the ACL implementation has an 84.2% overhead. Thus, ACL messages are only used for low data rate applications, such as system reconfiguration, health monitoring and task configuration. High data rate applications can make use of traditional raw CAN frames. Provided the most significant byte of the CAN frame ID is set to zero, the CAN frame will be routed throughout the system by the cell MMs. Before reaching any protein in the system, a raw CAN frame passes through, at most, two MMs.

Figure 13 shows the internal CAN bus bandwidth overhead caused by the MM’s health checking routine, and by the reprogramming of a protein. Both the internal and external CAN buses operate at 100 kBits/s. During this experiment, protein failures (supply loss) were induced at ~40 and 110 seconds, leading to the large peaks in bus utilisation as reprogramming occurred. The failed proteins automatically revived after supply restoration at ~180 and 240 seconds, leading to another two reprogramming sessions. The smaller peaks in between reprogramming sessions are caused by routine ACL-based health checks carried out by the MM on each of the cell’s proteins. The height of these peaks is seen to decrease from a maximum at about 10 seconds, at which time all three proteins were alive, to a minimum at around 150 seconds, at which time only a single protein was alive. The health checking routine never consumes more than 4% of the bus bandwidth, while the reprogramming routine can consume up to 65%. However, it is important to note that, due to the CAN bus’s intrinsic arbitration protocol, raw CAN peer-to-peer frames will always take priority.

The repeating pattern of generic I/O circuitry can be seen along the edges of the cell PCBs in Figure 11. Each GPIO’s circuitry occupies approximately 10 x 20 mm of PCB area, using component footprints of 0805 and above. Future implementations can be minimized further by utilizing smaller surface mount components. The GPIO lines have been demonstrated successfully interfacing to I2C buses, driving DC motors, and reading analogue inputs [4]. However, the current design of the GPIO circuitry has certain limitations. Firstly, the bandwidth of the GPIO lines is limited at below 200 kHz due to parasitic capacitance. Secondly, the GPIO output circuitry suffers from voltage droop when supplying large currents, making it poorly suited for supplying high current logic devices. With an input supply of 3.3V, the GPIO circuitry outputs 3.154V at 10 mA and 2.469V at 1A. The voltage drop is due to a reverse current protection diode, MOSFET on state resistance (0.125 Ω), and current sense resistor (0.1 Ω). The voltage droop issue can be compensated for in most use cases by increasing the GPIO supply voltage.
5. CONCLUSIONS

Two implementations of the Artificial Stem Cell architecture, namely, the SME-SAT CubeSat cell and the cell-based ADCS, have been successfully implemented. The SME-SAT cell design allowed a complex piece of avionics to be designed in under a year by a single engineer. In addition, it gives SME-SAT graceful degeneracy in experimental output by removing several potential single points of failure.

The cell-based ADCS was designed to demonstrate the feasibility of a complete, CubeSat-scale multicellular system. Its performance and overheads were compared to a reference COTS ADCS solution. A comparison summary is given in Table 1. Both designs occupy approximately 200 cm$^2$ of PCB area. Advantages of the cell-based design include its enhanced reliability in lower operating modes and the ease of adding significant reliability by adding cells. However, compared to CubeADCS, the cell-based ADCS consumes ~77% more power and is predicted to be more than 30% less reliable in its full operating mode after a year.

The reliability benefits of the ASC architecture have been shown to increase with system complexity [5]. Therefore, the ASC architecture may be more suitable to low-cost satellites in the microsatellite class, where increased processing capacity is valued and power consumption is less critical.

Table 1. Summary and comparison of CubeADCS and cell-based ADCS characteristics.

<table>
<thead>
<tr>
<th></th>
<th>CubeSpace Bundle</th>
<th>2 x Cells @ 12 MHz</th>
<th>3 x Cells @ 12 MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>1yr Reliability (%)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detumble</td>
<td>85.8</td>
<td>95.2 (+11%)</td>
<td>98.8 (+15%)</td>
</tr>
<tr>
<td>Mag Control</td>
<td>66.8</td>
<td>71.4 (+6.9%)</td>
<td>97.2 (+46%)</td>
</tr>
<tr>
<td>Momentum Bias</td>
<td>66.8</td>
<td>33.4 (-50%)</td>
<td>91.9 (+38%)</td>
</tr>
<tr>
<td>Perform. (DMIPS)</td>
<td>60.44</td>
<td>60 (-0.7%)</td>
<td>90 (+50%)</td>
</tr>
<tr>
<td>Power Cons. (W)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detumble</td>
<td>0.26</td>
<td>0.41 (+58%)</td>
<td>0.56 (+115%)</td>
</tr>
<tr>
<td>Mag Control</td>
<td>0.35</td>
<td>0.55 (+57 %)</td>
<td>0.70 (+100%)</td>
</tr>
<tr>
<td>Momentum Bias</td>
<td>0.35</td>
<td>0.62 (+77%)</td>
<td>0.78 (+123%)</td>
</tr>
<tr>
<td>PCB Area (cm²)</td>
<td>~ 200</td>
<td>~ 200</td>
<td>~ 300 (+50%)</td>
</tr>
</tbody>
</table>

REFERENCES

BIOGRAPHY

Alexander O. Erlank.
(B.Eng, 2011; MEng, 2013), is a Ph.D. candidate and member of the On-Board Data Handling Group at the Surrey Space Centre. His research interests include reconfigurable embedded systems, adapting COTS hardware for space applications, and attitude determination and control. In addition, he has been involved in various CubeSat projects for which he has developed subsystems and payloads.

Dr Christopher P. Bridges.
(BEng, 2005; PhD, 2009) leads the On-Board Data Handling (OBDH) research group within Surrey Space Centre (SSC). He researches software defined radios, real-time embedded systems, agent computing, Java processing, multi-core processing in FPGAs, and astrodynamics computing methods in many spaceflight payloads. In 2013, he designed, built and still operates the UK’s first CubeSat (STRaND-1) with SSTL and now contributes towards computing hardware and software in missions with SSTL, on ESA’s ESEO mission and also the NASA-JPL/CalTech AAResT mission.