

Initial Development of Embedded Low-Power Parallel Processing for On-Orbit Spacecraft Anomaly Management

Christopher Kitts, Paul Toledo, Mike Rasay
 Robotic Systems Laboratory, Santa Clara University
 500 El Camino Real, Santa Clara CA 95053; 408.554.4382
 ckitts@scu.edu, PToledo@scu.edu, rrasay@scu.edu

ABSTRACT

Maintaining the health of a spacecraft is a critical element of satellite operations, typically driving operations costs and often requiring standing armies of highly trained operations staff. An important aspect of this work is anomaly management (AM), which is the detection, diagnosis and resolution of anomalous conditions. Research in this area ranges from the development of robust reasoning techniques to the design of highly-performance flight processors able to implement these techniques on-orbit. Our recent work in this area focuses on the composition of advanced model-based reasoning (MBR) algorithms for AM that can be efficiently executed on a new generation of low-power, low-cost multi-core embedded processors. These processors provide a parallel-processing capability that can potentially revolutionize the performance of highly accurate but deliberative and computationally expensive MBR algorithms while still being suitable for space vehicle applications. In this paper, we will describe our latest theoretical and algorithmic contributions to MBR-based AM. We will discuss how our successfully demonstrated algorithms are being recast for parallel processing, and how these newly formed algorithms are being prototyped using new low-power multi-core embedded processors. Finally, we will discuss testbeds for this technology ranging from ground engineering units to simple student-based flight experiments.

INTRODUCTION

Since an embedded processor was used as the flight computer for Apollo, the aerospace industry has struggled to balance the incorporation of new, higher performance computing technologies with the risks associated with non-heritage equipment, the space environment, escalating on-board complexity. Today, on-board computers have found wide application in tasks such as payload processing, configuration management, sensor estimation, and active control of components for attitude, thermal, power, and communications processes. Of course, the flexibility afforded by software makes the use of flight computers a compelling solution for tasks involving tasks command and data handling and feedback control.

Some missions demand high levels of autonomy given challenges such as the need for rapid decision-making or the need to robustly operate in unpredictable and highly dynamic environments. Typical missions falling into this category include planetary probes, highly responsive platforms capable of responding to transient events, and space vehicles providing life support functions.

The selection of flight computers for a space system often involves trade studies regarding the computing

architecture and its effect on cost, performance, ease of integration, and reliability. One of the significant choices has been the selection either a centralized or a distributed computing architecture.¹ In a centralized architecture, a single flight computer typically performs all computing functions and is connected to subsystems in a star configuration. While this can simplify the logical design through the concentration of computing in a single component, it is this very concentration that often leads to drawbacks in terms of reliability, flexibility, and ease of integration. In distributed computing architectures, computing components are located throughout the system, and component connectivity often takes other forms, such as a linear bus.

Significant previous work at Santa Clara University (SCU) has explored the benefits and challenges of distributed computing architectures. This has led to the development of the Emerald Protocol Suite (EPS), a set of standards for data/communications/power, as well as hardware and software implementations of this standard in the form of the AVR-SAT dCDH (distributed command and data handling) flight computing system.² In partnership with collaborators at the University of Texas at Austin, a suite of AVR-SAT computers form the dCDH system for the two FASTRAC nanosatellites

currently scheduled to launch in 2009 through the University Nanosatellite Program.³ Other academic partners, such as Washington University in St. Louis (WUSTL), have found great value in this system and are incorporating it into their small spacecraft as well.⁴ In a demonstration at the 2006 AIAA/USU Conference on Small Satellites, SCU and WUSTL conducted live demonstrations of rapid integration of new components as well as the arbitrary cross-strapping of two spacecraft, capabilities enabled by the use of standards such as EPS and the incorporation of plug-and-play like capabilities.⁵

While SCU faculty, staff and students continue to explore the benefits of dCDH technologies, we are also searching for new computing technologies that have the potential to enable new capabilities and cost-effective solutions. One such technology is parallel processing.

PARALLEL PROCESSING

Parallel computing is the simultaneous use of multiple processors to solve a computational problem. In this computing approach, a software program is divided into discrete functional parts, and the discrete series of instructions for each part are executed concurrently on different CPUs.⁶

The primary benefit of parallel computing is the potential performance improvement in the speed of a computing task given the use of multiple processors. Amdahl's law provides a prediction for the theoretical maximum speedup that can be achieved.⁷ For systems whose processes are completely independent, the optimal parallelization speedup scales with the number of processors. In practical applications, a portion of the computing task cannot be computed in a parallel fashion and/or computational overhead is required to manage the parallel computation. In such cases, near optimal speedup is achieved for small numbers of processors (ie. 6 processors) and stabilizes into a constant value for larger number of processors (ie. 1024 processors). Figure 1 displays the amount of speedup achievable for a varying number of processors and with different fractions of the computing task that can be parallelized.

Computers that leverage parallel processing come in the form of Shared Memory Processors (SMP) and Distributed Networks. SMP architectures behave as one coherent region under the management of a single operating system. The discrete parts of the computational problem are assigned to specific processors. Distributed Networks, also known as "clusters" are made up of multiple stand-alone computers connected on a common network. A master node usually acts as the controller and assigns the

discrete parts to other nodes. In industry, these architectures are used in automotive applications, aerospace studies, chemical and pharmaceutical studies, electronics, energy research, geophysics and oil applications, and weather prediction. Systems vary from 16 processor engineering workstations, 1024 processor supercomputers, and even distributed hybrids. Personal computer graphics cards are another example, with these cards using as many as 256 parallel processors in order to perform functions such as pixel rendering. It is important to note that these applications are systems on the ground so power demands and thermal loads are easily accommodated.

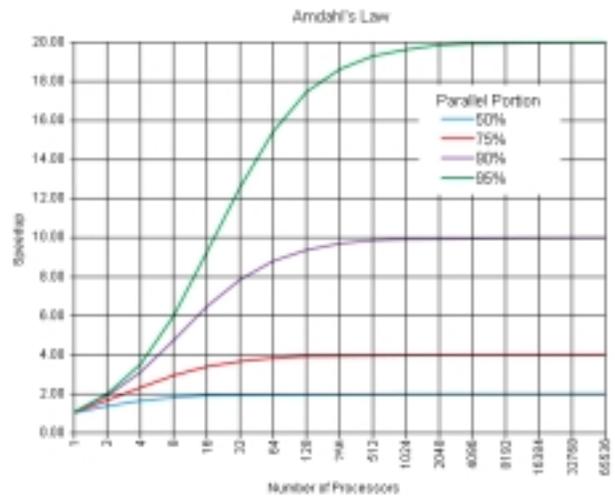


Figure 1: Parallel Processing Performance Improvement⁸

Only recently have commercial low-power parallel processors become available for embedded applications. The design and exploitation of these multi-core systems have stimulated new analyses regarding performance implications for varying core hardware design and for symmetric, asymmetric, and dynamic processing architectures.⁹

Faculty, staff and students have been exploring the use of these processors for a variety of applications on robotic platforms that operate across the range of land, sea, air and space domains, with diverse sponsors ranging from the National Science Foundation to BMW. Two specific examples of such systems are the Parallax Propeller microcontroller and the IntellaSys SEAForth Scalable Embedded Array processor.

Parallax Propeller Microcontroller

Shown in Figure 2, the Parallax 8-processor Propeller microcontroller that supports simultaneous independent

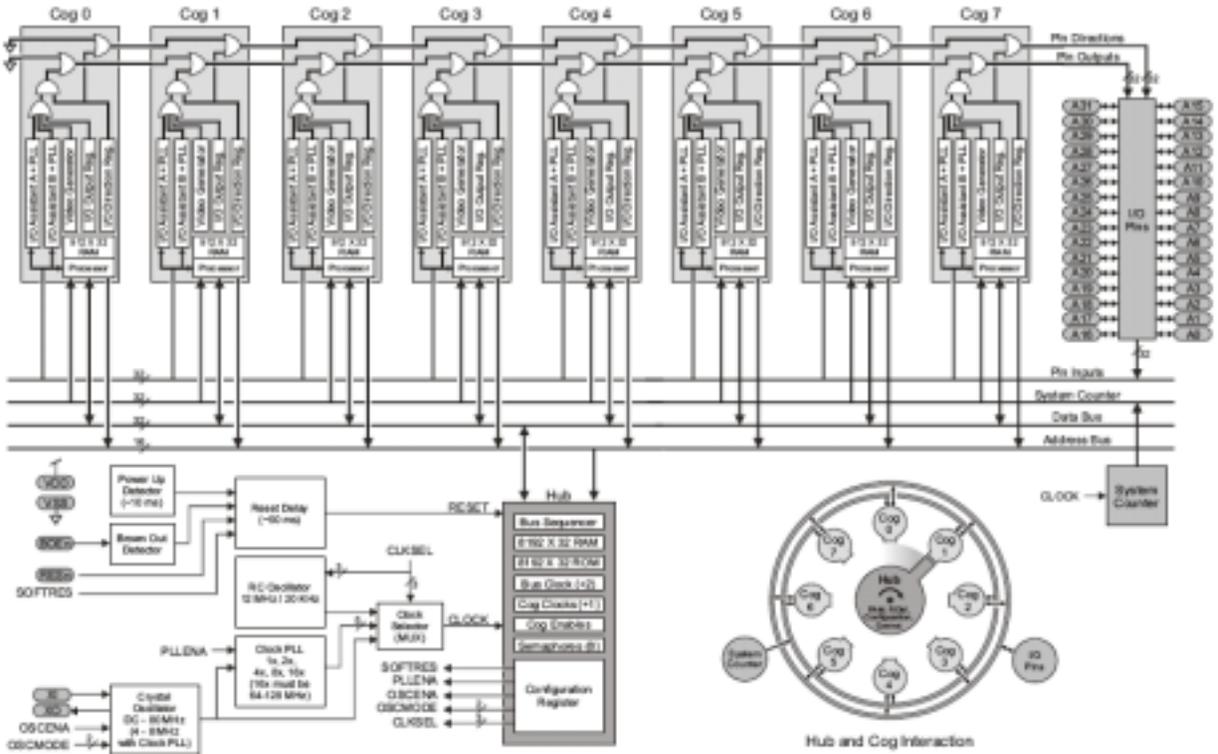


Figure 2: Parallax 8-Core Propeller Microcontroller¹⁰

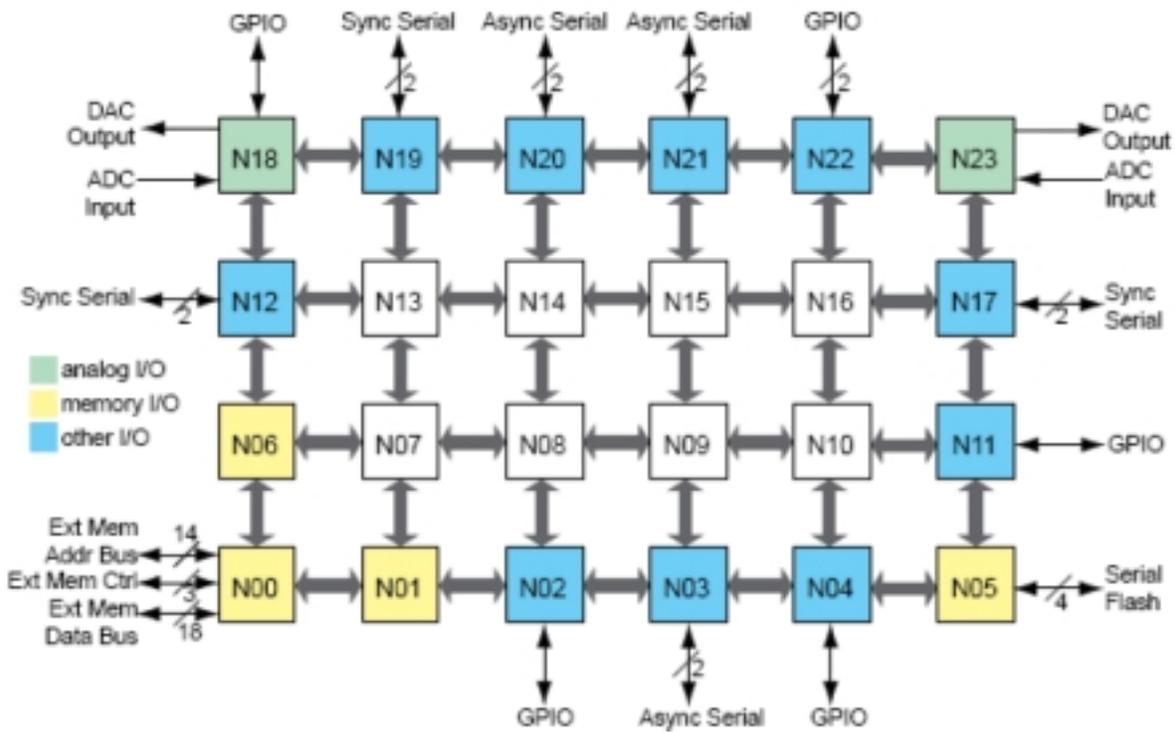


Figure 3: Block Diagram of IntellaSys 24-Core SEAFORTH-24A Microprocessor¹²

or cooperative processing on each processors while sharing common resources.¹⁰ Each core is a 32-bit processor with 2 kB of RAM running at 3.3V and a maximum 5 mA current. Each core can compute at a speed of 20 MIPS, giving the overall microcontroller an execution speed of up to 160 MIPS. Shared resources, allocated via semaphores when necessary, include 64 kB of RAM and ROM, 32 i/o pins, and a system clock. The processors may be programmed in assembly language or in a new higher-level language known as Spin. Development of embedded applications is facilitated by direct support for a keyboard, mouse, and VGAmomitor.

Although the SCU team is planning to make use of higher performing parallel processors with more cores, the current availability and shallow learning curve of this microcontroller, combined with Parallax's outstanding educational support, have made its use quite valuable to our initial prototyping efforts.

IntellaSys SEAFORTH Processor

Shown in Figure 3, the IntellaSys SEAFORTH-24A is an array of 24 core processors that together can deliver up to 18 BIPS of computing power.¹¹ Each processor is an 18-bit stack oriented computer with its own dedicated 64 words of both RAM and ROM. Cores have i/o to include SPI, serial, parallel, A/D and D/A, and general purpose single-pins; cores can also be programmed to provide i/o services to other processors by supporting I²C, synchronous or asynchronous serial communications, or other protocols. The cores consume a maximum power of 9 mA and can be placed in a sleep mode that draws only 5 μW. Processors are programmed using the VentureFORTH language, which serves as the native execution language for the cores.

The SCU research team has been working with IntellaSys for the past year in order to learn the architecture and programming techniques specific to the SEAFORTH processor as well as to specify and review layouts for SEAFORTH-based boards that are optimized for specific applications involving collaborators at BMW.¹³

SPACECRAFT ANOMALY MANAGEMENT

Anomalies are unexpected conditions that occur in a functional engineering system.¹⁴ They must be detected, diagnosed and resolved in order to maintain the system in its functional role. Managing anomalies in space systems is particularly challenging given their complexity and their remote orbital environment. Many of the common reasoning approaches used for anomaly management, such as expert systems, have

poor performance due to their informal knowledge base, their lack of comprehensive and systematic evaluation, and their inability to assert causal relationships.

Recent work in causal, model-based reasoning and its application to systems ranging from automobiles to spacecraft has demonstrated the ability of this technology to provide precise conjectures regarding the anomalous state of a complex engineering system. As shown in Figure 4, detection is the result of comparing observations (OBS) of the real engineering system with predictions based on a simulation model that has been configured (CNFG) in the same manner. If the observations and predictions are consistent, then the system is judged to be nominal; otherwise the inconsistency is flagged as the symptom of an anomaly, and formal diagnosis and resolution activities commence.

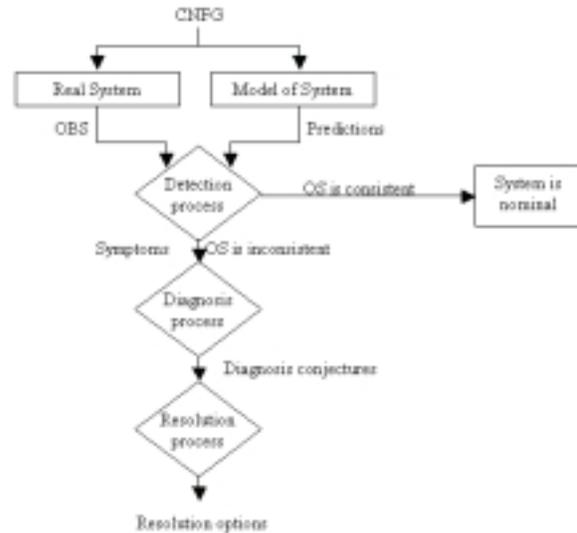


Figure 4: Process Flow for Model-Based Anomaly Management¹⁵

Our previous work in this field has established a comprehensive theoretical framework for anomaly management reasoning based on fundamental descriptions of the design and intended operation of a system. This theory formally defines several distinct but potentially interacting anomalies, termed faults, hazards, and misconfigurations. We have also developed computational algorithms based on this theory; these algorithms rely on a computational technique known constraint relaxation and are posed in a control theoretic manner.^{14,15} Furthermore, we have developed software that implements these algorithms in the form of Matlab/ Simulink programs, and we have

experimentally verified their functionality. Finally, we have validated the use of this conceptual framework through ground-based execution of the algorithms in order to manage anomalies throughout the end-to-end space system for several spacecraft missions, ranging from the university-class Sapphire satellite to NASA Ames Research Center small satellites such as GeneSat-1 and PharmaSat.¹⁴⁻¹⁹⁸

While this approach to anomaly management has been shown to achieve very high performance compared to other diagnostic reasoning techniques, its barrier for implementation in embedded environments is the level of computation required for the diagnosis and resolution processes. A core element of these processes involves the simulation of system behavior over a range of alternate system states. The individual simulations for each such case may be computationally decoupled thereby allowing, at least in principle, for a parallel version of the algorithms to be developed and implemented. It is this conjecture that has motivated the development of the work reported in this paper.

EXPERIMENTAL TESTBED

To explore the implementation, verification, and validation of a parallelized anomaly management process, we have initiated the design of such a system.

A Simple, Controlled Target

While we have demonstrated advanced model-based anomaly management for a variety of complex, distributed engineering systems, we generally start the exploration of new technical approaches in this area with a very simple target engineering system: the full digital adder as diagramed in Figure 5. This system has served as one of the benchmark target systems in this research area for several decades. This is due to a design that is simple and yet oddly challenging to troubleshoot in an exhaustive manner. Characteristics that make this design particular interesting and relevant include its upstream and downstream branching, its configuration-dependent observability, and its one-to-many mapping of symptoms to possible root causes of the problem. We note that even for a simple system such as this, it is rare for a human analyst to routinely produce a correct, complete set of diagnosis conjectures.

For our experimentation, an enhanced form of this circuit is used in order to provide characteristics that allow our anomaly management algorithms to demonstrate their full capabilities. These include additions such as component redundancy; furthermore, modifications have been made to support the ability to electronically inject anomalies such as component

failures, hazardous conditions such as elevated temperatures, or incorrect configuration inputs.

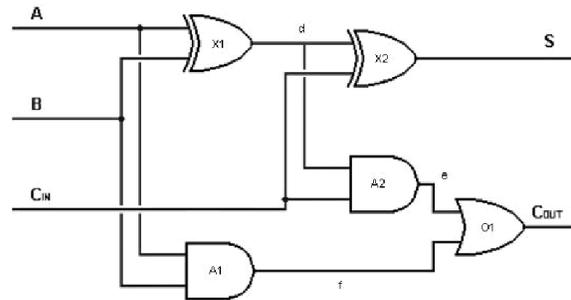


Figure 5: A Full Digital Adder.

Anomaly Management Testbed Hardware

The hardware architecture of the testbed is shown in Figure 6. The adder circuit is controlled and monitored by a parallel processor; in the current first iteration, a Parallax Propeller microcontroller, shown in Figure 7, is being used. This multi-core computer executes an initial iteration of the parallelized anomaly diagnosis algorithm that has been previously verified and validated.¹⁴ Figure 8 shows the adder and parallel processor together during test. The parallel processor interfaces with an SCU AVR-SAT distributed command and data handling module, shown in Figure 9, in order to provide it with SCU's standard satellite/robotic avionics interface, thereby allowing the testbed to be used for both laboratory verification and field/space demonstrations.² Figure 6 also shows that the parallel processor interfaces with an auxiliary interface board. The purpose of this board is to support an interface with other engineering systems. Its use for one particular test will be described later in this paper.

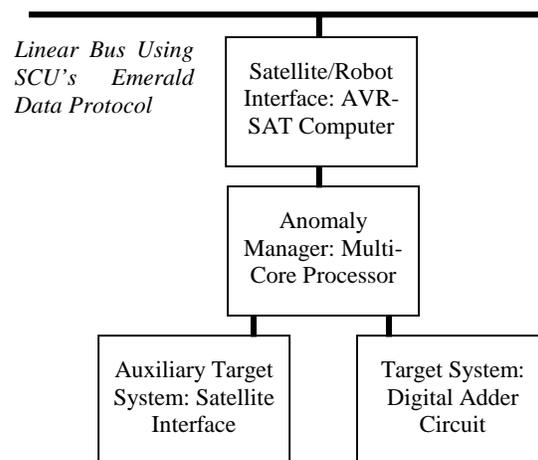


Figure 6: Testbed Hardware Architecture



Figure 7: Parallax Propeller Development Board

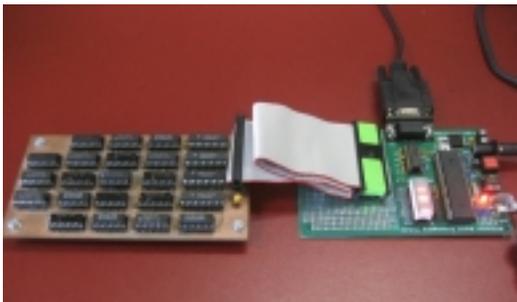


Figure 8: Adder and Parallel Processor During Test



Figure 9: SCU AVR-SAT Flight Computer

Anomaly Management Software

In the current iteration of the anomaly management software, the software performs the following functions:

- a) interface with the system being analyzed in order to acquire telemetry,

- b) model the engineering system being analyzed (in this case, the digital adder),
- c) simulate the engineering system in order to produce predictions of behavior,
- d) perform the detection process by checking the level of consistency between observations of the real system and predictions of the simulation,
- e) perform a diagnosis algorithm when symptoms are observed based on any consistencies found in step (d).

A simple example of how models are computed when using the Parallax Propeller and the SPIN programming language is provided in Figure 10. In this example, the code first checks to see whether the A or B unit for the X1 gate (which is a Boolean XOR) is selected. In either case the component has two input values, an internal state, an output, and the XOR behavior. Similar logic applies to all of the components throughout the adder model. For simplicity, output statements have been extracted. Behavioral modeling such as this is quite straightforward for simple digital logic components; however, the technique has been extended to far more behaviorally complex components and physical processes.¹⁴⁻¹⁹

The detection algorithm has been implemented for the AVR-SAT processor, and as of this writing is being ported for execution on the parallel processor.

The diagnosis software will be implemented following detection. Keeping Amdahl’s law in mind, our design approach is to assign processes across multiple processors to increase speedup. The Propeller will allow the software functions to be divided between multiple cogs in order to execute the anomaly management process. It is very similar to the using the SMP schema. This is more critical for the diagnosis process, which is typically more computationally expensive than detection.

The anomaly manager provides the following software elements, as illustrated in Figure 11.

- The Command and Control Executive utilizes a single processor. It is responsible for the overall software flow and communicates with the AVR-SAT computer. It also assigns symptoms to available processors. For example, if the detection process identifies 4 symptoms, command and control will assign each symptom to an individual processor for execution of the diagnosis algorithm.

```

`compute XOR 1 Logic Gate
PUB computeXOR1

IF NOT (conf.getXOR1_select)
    comp[1].setInputValue(1,conf.getAdder_In1)
    comp[1].setInputValue(2,conf.getAdder_In2)
    comp[1].setStateValue(30)
    comp[1].setOutputValue(comp[1].getInputValue(1)^comp[1].getInputValue(2))

ELSEIF (conf.getXOR1_select)
    comp[2].setInputValue(1,conf.getAdder_In1)
    comp[2].setInputValue(2,conf.getAdder_In2)
    comp[2].setStateValue(30)
    comp[2].setOutputValue(comp[2].getInputValue(1)^comp[2].getInputValue(2))

return

```

Figure 10: Example of Behavioral Modeling using the SPIN Language for an XOR gate

- The Real System Interface utilizes a single processor. It is responsible for configuring the digital adder circuit, reading the digital adder circuit, and in the future, injecting anomalies in the satellite system.
- Shared Memory is visible to all processors. The symptoms list of asserted anomalies found during the detection process are stored here. This enables the command and control executive to read the shared memory and assign execution of the detection process to the necessary processor.
- The Detection Process utilizes a single processor. It computes the expected system state and checks for any inconsistency between the model and real system. Any inconsistencies are added to a symptoms list in shared memory.
- The Diagnosis Process can utilize multiple processors concurrently. It is used to identify valid diagnoses that result in re-establishing consistency between observations and our estimate of the system's anomaly state.

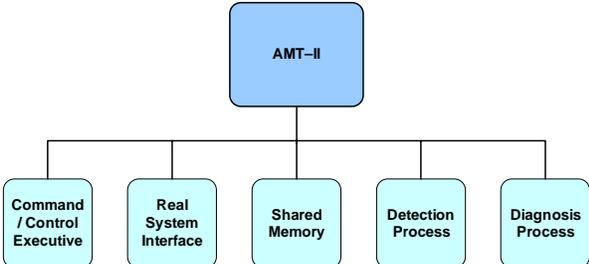


Figure 11: Anomaly Manager Software Elements

Figure 12 illustrates the high-level software flow being implemented for execution on the Parallax Propeller microcontroller.

First, the entire system is initialized and observed. For the simulated system, the system configuration is used

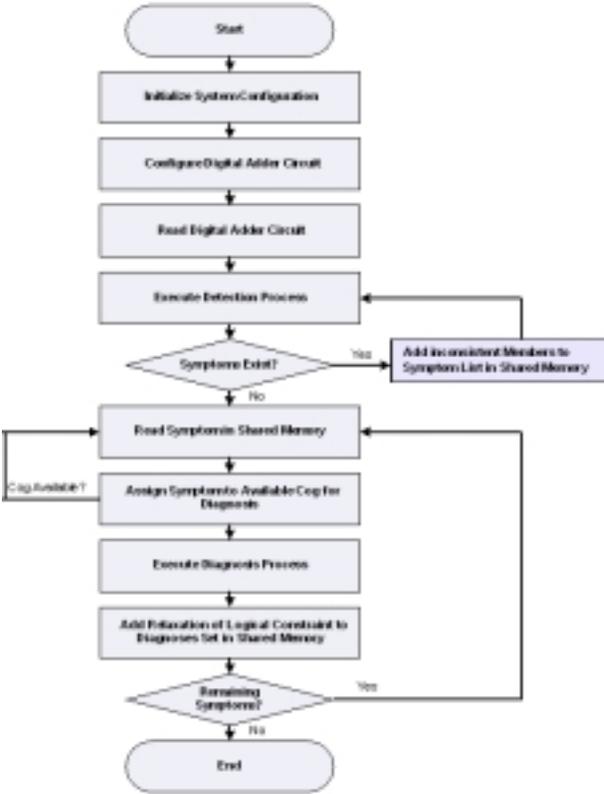


Figure 12: Anomaly Manager Software Flow

to initialize the model inputs, and system values within the model are computed in order to generate the ultimate outputs that can be compared to telemetry from the actual system. For the real system, the system configuration is used to initialize the physical inputs where are set in order for the circuit to compute the result as per its design.

Second, the detection process is executed. The detection algorithm checks for any inconsistencies between the simulated and observed values. Inconsistencies are added to a symptoms list in shared memory. If no symptoms are added to the list then the system is NOMINAL, otherwise the system is ANOMOLOUS.

Third the diagnosis process is executed. Anomaly diagnosis only occurs when the system is ANOMOLOUS. If the system is ANOMOLOUS, the program checks the symptom list in shared memory. Command and Control assigns the symptom to the next available processor for diagnosis and the diagnosis process is immediately executed. If more symptoms exist, each will be assigned to the next available processor and the diagnosis process is executed in parallel. This repeats as long as processors are available. It is possible that diagnosis of 6 different symptoms can occur concurrently on 6 processors. When the observed system becomes consistent with the predicted attribute values in a particular processor, the processor adds that logical constraint to the set of possible diagnoses in shared memory. The processor then becomes available for the next symptom in the shared memory list.

Flight Experiment Prototype

The anomaly management testbed described is being used for ground test in order to explore alternate parallel processing designs and to verify their functionality. Once this is done, the parallel implementation can be applied to a number of engineering systems for which SCU routinely conducts anomaly management operations.

Of particular interest, however, is the potential to demonstrate this technology in space on-board a student-built spacecraft. The design of the current testbed has been developed to support just such a possibility given the use of the AVR-SAT avionics module, which is the standard computational interface for any payload or subsystem incorporated into SCU student-developed spacecraft.

In particular, the current testbed is being baselined for a student-developed spacecraft currently being prototyped through the Air Force Office of Scientific

Research's University Nanosatellite Program (AFOSR); it is noted that the testbed is compact and functionally benign, making it a candidate for other launch opportunities if they should evolve.

In the current nanosatellite project, known as OBSIDIAN (Orbiting Biological Study using In-situ Diagnostics Implemented via an Autonomous Nanosatellite), the anomaly management testbed serves as a payload, addressing AFOSR's interest in autonomous control and advanced computing technologies for space.²⁰ A photo of OBSIDIAN is shown in Figure 13.

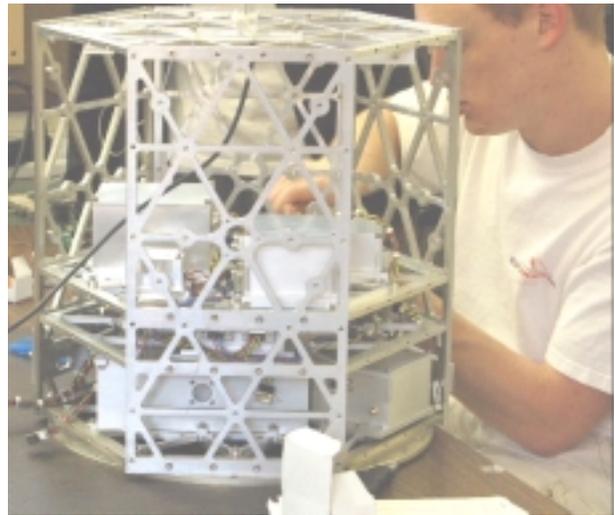


Figure 13: The OBSIDIAN Nanosatellite

We note that the act of performing anomaly management on a digital adder circuit while in space, in and of itself, does not add much to verifying algorithm performance, since such performance is suitably demonstrated on the ground. However, even though it is a simple system, demonstrating this capability given the design of our testbed will allow us to perform what we believe will be the first controlled and properly validated demonstration of this technology while in space. This is the case given that we have the ability to inject anomalies in a controlled manner and that we can operationally conduct this experiment in a double blind manner such that performance data can be collected using well-established research norms.

Beyond this, however, we are working to support an expanded, less-controlled but more interesting demonstration of the same technology. As indicated in Figure 6, the parallel processor interfaces with a secondary breakout board. For OBSIDIAN, this breakout board will be used to inject simulated

anomalies into the spacecraft at large; in effect, we will be intentionally breaking this satellite when in orbit! Current plans include simulating several anomalies, such as a) a communication system failure via the deliberate disabling of a transceiver component connection, and b) an arbitrary component fault via the deliberate disabling of power. The anomaly manager will have a simple model of spacecraft functionality and will use this to perform anomaly management using the same algorithms verified through use with the digital adder. Overall, this demonstration will provide a way to collect technology validation data in a high-risk, operational environment, a feat that is particularly suited to student-based projects such as those supported by the University Nanosatellite Program.

SUMMARY

Parallel processing holds a promise for significant performance improvement for computing tasks that can be parallelized. The diagnosis and resolution tasks associated with space system anomaly management are examples of processes where we believe that this can be effectively done. The ability to perform such computations with a low-power computer suitable for embedded systems applications opens the door to considering the use of such systems as on-board anomaly managers for small spacecraft. Our initial work presented in this paper has reviewed this motivation and described our initial work in developing a simple experimental testbed for exploring the cost-effective use of this technology.

ACKNOWLEDGEMENTS

The authors thank those students and collaborators who have been involved in our efforts to conceptualize and develop the use of embedded parallel processors for a wide variety of applications and testbeds, to include Mr. Jeff Ota, Wolfgang Appuhn, and Mr. Will Nguyen.

Portions of this work have been developed under the auspices of the University Nanosatellite Program, funded by the Air Force Office of Scientific Research through Grant No. FA9550-07-1-0097. Additional uses of this technology, which have informed this work, have also been funded by BMW as well as by the National Science Foundation under Grant No. CNS-0619940. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the U.S. Air Force, BMW, or the National Science Foundation.

REFERENCES

1. Palmintier, B., The Emerald Protocol Suite: Design and Implementation of a Modular,

Distributed Architecture for Small Satellite Command, Telemetry, and Power Systems. Stanford University Engineer Degree Thesis. Advisors: T. Kenny, C. Kitts, R. Twiggs, E. Carryer. June 2004.

2. Palmintier, B., C. Kitts, P. Stang, and M. Swartwout, "A Distributed Control Architecture for Small Satellite and Multi-Spacecraft Missions," Proceedings of the 16th AIAA/USU Conference on Small Satellites, Logan Utah, August 2002.
3. Holt, G., S. Stewart, J. Mauldin, T. Campbell, P. Eckho, H. Elmasri, B. Evans, M. Garg, J. Greenbaum, M. Linford, M. Poole, E.G. Lightsey, L.L. Raja, and T. Ebinuma. "Relative Navigation, Microdischarge Plasma Thruster, and Distributed Communications Experiments on the FASTRAC Mission." Proceedings of the 17th AIAA/USU Conference on Small Satellites, Logan Utah, August 2003.
4. Swartwout, M., C. Kitts, P. Stang, and E.G. Lightsey, "A Standardized, Distributed Computing Architecture: Results from Three Universities," Proceedings of the 19th AIAA/USU Conference on Small Satellites, Logan Utah, August 2005.
5. Rogers-Marcovitz, F. and P. Williams, "Dallas EEPROM Equipment Profile for Rapid Integration and Automatic System Modeling," Proceedings of the 5th AIAA Responsive Space Workshop, Los Angeles California, April 2007.
6. Barney, B., "Introduction to Parallel Computing," LLNL UCRL-MI-133316, September 2007.
7. Hennessy, J. L., and D. A. Patterson, Computer Architecture: A Quantitative Approach. Morgan Kaufmann, 3rd Edition, 2003.
8. ____, "Amdahl's Law," Wikipedia, 2008.
9. Hill, M.D. and M.R. Marty, "Amdahl's Law in the Multicore Era," IEEE Computer, July 2008.
10. ____, "Propeller P8X32A DataSheet: 8-Cog Multiprocessor Microcontroller," Parallax Inc, 2007.
11. ____, SEAForth-24A Scalable Embedded Array Processor, IntellaSys PB012908 Product Brief (Preliminary version). 2007.
12. ____, "SEAForth-24A Device Data Sheet, Preliminary Version 1.2," IntellaSys, 2008.
13. Chartier, A., C. Hwuang, and J. Lupu, An Enhanced Automobile Computing Architecture Exploiting Embedded Parallel Processors, Advisor: C. Kitts, Santa Clara University

- Undergraduate Thesis, Departments of Electrical and Computer Engineering, in draft.
14. Kitts, C., "Managing Space System Anomalies Using First Principles Reasoning." IEEE Robotics & Automation Magazine, Sp. Issue on Automation Science, v 13 no 4, December 2006.
 15. Kitts, C., Theory and Experiments in Model-Based Space System Anomaly Management. Stanford University Ph.D. Dissertation, 2005.
 16. Schuet, D., and C. Kitts, "A Distributed Satellite Operations Testbed for Anomaly Management Experimentation," Proceedings of the 3rd AIAA "Unmanned Unlimited" Systems, Technologies and Operations Conference, Chicago, IL, September 2004.
 17. Kitts, C., and R. Rasay, "Model-Based Anomaly Management for Small Spacecraft Missions," Proceedings of the 21st Annual AIAA/USU Conf on Small Satellites, Logan UT, 2007.
 18. Rasay, R., A Graphical Model-Based Reasoning Analysis Environment for Space System Anomaly Management. Advisor: C. Kitts. Santa Clara University Masters Thesis, June 2007.
 19. Minelli, G., Model-Based Reasoning Applied to Biological Spacecraft Payloads for Anomaly Management. Advisor: C. Kitts. Santa Clara University Masters Thesis, in draft.
 20. Kitts, C., "The OBSIDIAN Nanosatellite," SCU Technical Report submitted to the Air Force Office of Scientific Research, 2008.