Reconfigurable Computing Concepts for Space Missions: Universal Modular Spares

M. Clinton Patrick
NASA/MSFC/EV43
Marshall Space Flight Center, AL 35812
256-544-3836
clint.patrick@nasa.gov

Abstract—Computer hardware systems for control, data collection and other purposes will once more be crucial resources in NASA's upcoming space missions. Compulsion to provide these resources within mission payload requirements, with hardiness to operate for extended periods under potentially harsh conditions in off-World environments, is daunting enough without considering the possibility of doing so with conventional electronics. This paper examines some ideas and options, and proposes some initial approaches, for design of reconfigurable computing resources offering true modularity, universal compatibility, and unprecedented flexibility to service all forms and needs of mission infrastructure. [1] [2]

TABLE OF CONTENTS

1. INTRODUCTION

Reconfigurable Computing (RC) research oversight at Marshall Space Flight Center (MSFC) has since 2006 fallen under the Radiation-Hardened Electronics for Space Environments (RHESE) project. This is partly because of a need to provide avionics systems that are more robust in harsher off-World environments, and in part simply because the approach to providing systems for Space should logically be an integrated one.

Reconfigurability Defined

To gain a contextual understanding of reconfigurability, it may be most useful to contrast it with what it is not: it is not reprogrammability, although it can possess characteristics of reprogrammability. In a conventional serial system, based on a central processing unit (CPU), changes are effected by modifying commands (or OP codes) for specific operations to be executed, along with their order of execution, all in fixed hardware resources. That is, the CPU has portions of its circuitry which are dedicated to executing particular algorithms in particular ways to ensure specific outputs result when specific inputs are applied. Furthermore, in most cases only one copy of each algorithm exists in one particular location on a physical device – a situation which cannot be modified without designing and producing a new device.

In contrast, reconfigurable computing places circuitry for specific behavior where needed, and as needed. In other words, multiple copies of each piece of circuitry may be plugged into data flow paths where needed, with no circuitry wasted on unused algorithms. If a new algorithm is needed that was not previously realized in hardware, available resources may be configured – or freed and then reused – to meet that need.

2. CORE CONCEPTS IN RC

Reconfigurability in the Space Context

Regardless of technical intricacies, it is good to keep an eye turned toward potential benefits. To that end, an example is in order – one very relevant to Space flight.

Most conventional Space-related systems utilize a host of custom avionics boxes for each of the tasks and stages of a mission. Engine controllers operate during launch and ascent, Emergency Detection System (EDS) hardware handles potential abort conditions and responses, flight computers calculate, monitor and correct trajectories, specialized electronic boxes oversee and control long-range and short-range stages of automated/autonomous rendezvous and docking (AR&D), computers in landing vehicles handle descent and landing operations, surface rovers perform all manner of closed-loop control, and so on. For most of these processes, a particular computing resource is used for a limited period and then shuts down; in many instances, periods of operation do not even overlap.

Suppose, then, a single computing resource – or a small number of such resources distributed throughout a vehicle – could oversee many of the independent processes or limited subsets of overlapping processes listed above. During launch, engine and abort operations are maintained; in extraterrestrial travel, navigation functions are performed;

[2] IEEEAC paper #1503, Version 3, Updated December 16, 2007

from long-range, radar is wired in to guide the vehicle to close proximity with an orbital facility; camera systems are then configured in to monitor and execute docking procedures; then the same camera systems are used during de-orbit and landing. At this point, computer and camera systems from the vehicle can be transferred to a surface-roving vehicle, might be held as a spare for any system, or could instead be used for life support in a habitat module.

The direct impact in payload reduction alone from this technology should be phenomenal. Shared enclosure space, power consumption savings, and reduced complexity are further potential payoffs.

Other basic concepts stem from this very basic idea. These include what will here be called: Levels, Granularity, and Complexities of Reconfigurability. Also, RC efforts at NASA have come to be separated into three primary areas of concentration and objective technology developments: Interface Modularity, Processing Modularity, and Fault Mitigation. Each of these concepts will be touched upon below; note, however, that because of a very real degree of inseparability of the ideas, much overlap can be expected in their descriptions, which follow.

For more conceptual and historical perspective on RC technologies in general, see [1].

Levels of Reconfigurability

Levels of reconfigurability are, very briefly, a gauge of the physical strata at which an RC system is reconfigurable. These are currently delineated from macro to micro-levels of definition as: Box, Board, Chip, and Gate levels.

Granularity

The degree to which an RC system is reconfigurable may also be distinguished by a rough estimate of granularity, a term more widely recognized both inside and outside the RC field. For example, systems composed primarily of banks of interconnected field-programmable gate array (FPGA) units (often called "FPGA fabric" or more generally "reconfigurable fabric") can be reconfigured at the logic gate level, and thus are considered fine-grained. In contrast, a system based on Field-Programmable Node Arrays (FPNA) or Field-Programmable Object Arrays (FPOA) does not have such low-level reconfigurability and thus ranges from medium-grained to coarse-grained. A system based only on board-level reconfiguration falls squarely in the coarse-grained category[2].

Complexities of Reconfigurability

This account outlines four different complexities of reconfigurability: Basic, Physical Spares, Automatic, and Autonomous Reconfigurability.

Basic Reconfigurability—This is the heart of RC research: leveraging a core ability to modify algorithms and interfaces directly in hardware to address various changing demands. This typically centers either on timesharing the same hardware resource in a given system for many different purposes, or on upgrading hardware functionality, after production, as necessary.

Physical Spares Reconfigurability—Modular RC spares enable use of identical hardware components in different systems. Crucial to this is the ability to leverage basic reconfigurability of a given computing resource for widely varying operations in fundamentally different target systems. One very important appeal of this is the resulting potential for reduction of types of flight spares and further savings in payload weight. An additional very significant benefit is substantial savings in man-rating or other qualification costs for flight hardware, supposing flight qualification is made general enough for basically unlimited physical (primarily meaning mechanical) reconfiguration.

Automatic Reconfigurability—Automatic reconfigurability provides options, either through direct commands or pre-determined procedures, to modify the architecture and behavior of given circuitry to carry out significantly different computing functions in different contexts.

Autonomous Reconfigurability—This is the most complex – and most desirable – ability an RC system may possess. It is the capacity to reconfigure without external direction or oversight, and drives more at self-adaptation than the previous three complexities. While this does not exclude selective application of predetermined algorithms for effecting change, it primarily designates evolutionary techniques utilizing intelligent self-adaptation.

As can be seen, each of these complexities relies heavily upon successful implementation of prior ones. No one level of complexity can exist without the underlying abilities. Progress in development of technology will follow in step-wise fashion: first, with modules that can be manually reconfigured for different tasks in a given system; then versions that can be manually reconfigured for use in multiple systems; then versions that will automatically reconfigure based upon prompts from the system in which they are currently installed, with capability built along the way to timeshare the computing resource among multiple tasks; then with features added systematically to enable fault checking and mitigation, self-repair, and advanced autonomous reconfiguration and adaptive behavior.

Interface Modularity

Interface Modularity, or External Modularity, is that portion of a reconfigurable system most critical to its functionality as a universally modular piece of hardware. This feature makes it possible to interchange computing resources from different vehicles or applications, each possibly with entirely different communication and interfacing schemes.

Because much of the current capability in RC is centered upon use of FPGAs and related programmable logic devices (PLDs), it is fortunate some devices are beginning to

include embedded or downloadable (some from third-party vendors) capabilities to match various standard protocols; take, for example, provisions for Ethernet and RocketIO in the Virtex-5 product from Xilinx, along with modules for interfacing through RS-422, RS-485, and other communication standards.

A new effort capitalizes on this concept: the Universal Reconfigurable Translator Module (URTM) is being built to demonstrate at least three different serial bus conventions in a single package, with capability to reconfigure interfaces as various applications require [3].

An exciting aspect reconfigurability brings into the picture is apparent on realization that a given interconnection signal need not be tied to a specific communication protocol or even a specific location. A set of wires dedicated to carrying, for example, sixteen bits of parallel data may just as easily be reconfigured and rerouted in a few moments to sixteen individual serial communication links. Or spare links may be used to repair individual subsets of a communication channel as needed. Using FPGAs or other devices as simple switching networks, the very routing of the signals becomes almost infinitely changeable.

Processing Modularity

This has in the recent past been called either Internal Modularity or Spares Modularity. The former term seems ambiguous, while the latter better applies to a complete RC system including its interfacing modules.

Core ability to modify functionality of a system stems from this aspect of basic reconfigurability, as shown in Figure 1. The chief feature of interest here is a capacity to fundamentally modify underlying logic circuits and their internal interconnections. Regardless of the specific means by which reconfiguration is achieved, this must be flexible to some extent at the sub-board, sub-chip, or circuit level. It must also take place primarily as a substantial change in hardware itself. To clarify this point: a conventional system based upon a central-processing unit (CPU) is admittedly adaptable to accomplishment of different tasks, but its reconfiguration is accomplished – with few if any exceptions in today's systems – by modifying software only.

To be truly modular and universal, RC technology must be capable of realizing either massively-parallel circuits or instruction-based serial processes interchangeably. RC systems may very well incorporate CPU-based technologies in various forms, in either reconfigurable or non-reconfigurable, embedded varieties. In some instances, such an arrangement may prove more efficient than raw logic implementations.

Radiation Tolerance and Fault Mitigation

At some point, RC will be applied to harsh environments such as Space, with consequent related requirements for environmental tolerance, error detection and resolution. Conceptually, this objective is very diverse, but may be divided into three camps as follows: relatively conventional fault detection and mitigation benefits inherent in RC, similar benefits that require intentional exploitation, and entirely new possibilities emerging specifically because of RC technology.

Inherent Fault Mitigation Benefits of RC—First, because the technology is inherently more distributed than a conventional CPU-based computing device, RC devices should also be inherently less likely to exhibit negative behavior or sustain catastrophic damage in the event of radiation events. Since all of the computing activity in a CPU must pass through the central processing point's "bottleneck," radiation damage at that one point would practically guarantee complete loss of device functionality. With the more distributed RC approach, this particular central vulnerability does not exist.

Another benefit of RC technology seems almost serendipitous. Highly parallelized hardware often has lower system clock speeds, realizing total throughput advantages through execution

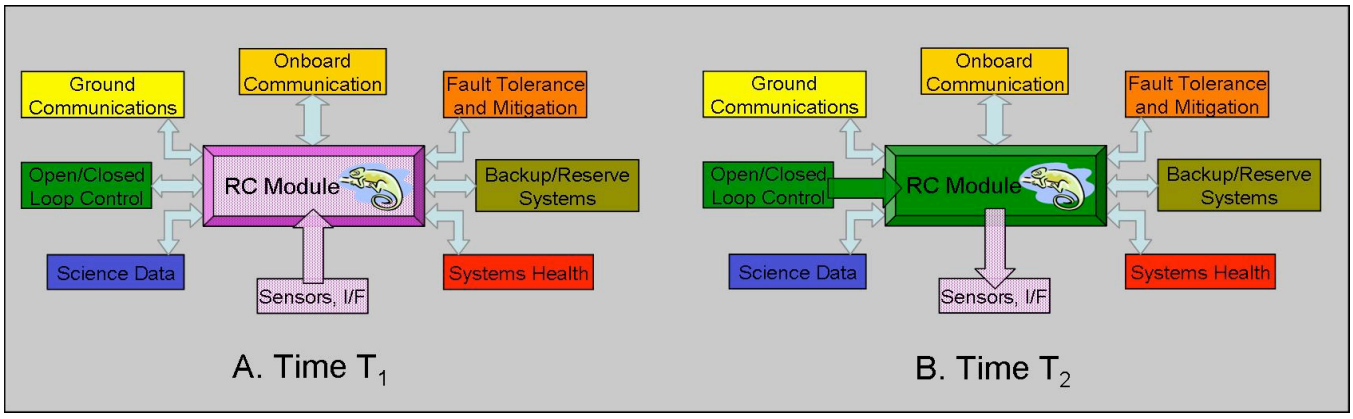

Figure 1 – RC Modules: Building Blocks for RC Processor Modularity
Different colors/shades illustrate modular function interchangeability, with return arrow paths indicating modified data, algorithms, or other functionality being stored for future use. The chameleon icon represents these RC modules.

of massive numbers of operations simultaneously. Because circuits with lower clock speeds tend to naturally have more radiation tolerance, fewer EMI issues, and better robustness in general, this can be considered an altogether agreeable bonus.

RC Techniques for Fault Mitigation— The idea of Hardware Swapping, or simply utilizing modular computational resources interchangeably in multiple systems, extends system maintenance matters. Computer resources flown in harsh environments or for exceptionally long periods may be repaired by physically swapping subsystems or entire systems as needed; modularity of the hardware simplifies logistics for its provision in an efficient manner. This is a low-hanging fruit, and as such will be demonstrated in the near future, likely much sooner than may be expected of other fault mitigation concepts.

In the event of permanent damage to particular points in a circuit, resources can be relocated, with reconfiguration around the damage. This brings up concepts of an "RC as hard disk" approach and "circuit paging." The former marks off bad portions of a circuit, once identified, cutting them from reserve resources much as is done with bad memory blocks in a hard disk controller. The latter involves keeping one tested copy of circuitry on standby, in order to allow periodically swapping out and running diagnostics on the operational copy.

Fault Mitigation Benefits Unique to RC—A few new possibilities exist solely because of the advent of RC. One of these is perhaps the most novel concept in this paper: fused sub-circuits.

In conventional electronic systems, catastrophic events such as power-to-ground shorts generally trigger a domino effect progressing through failures at the device, board, and box levels, typically ending when a power breaker is thrown. Unfortunately, this also leaves the system with no functionality at all.

Proposed here is the concept of fusing or otherwise protecting subsets of a device to enable isolation of failed portions. Failure of subsections of the device might result in a system reset, but upon recovery the system should be capable of replacing lost functionality in reserve locations and continuing as before.

Finally, it is possible current practices of multi-string redundancy and voting may one day become obsolete because of RC. By flagging significant decisions of single-string processes, recreating conditions and relevant circuitry, and double- or triple-checking results, redundancy may be created in a temporally displaced fashion rather than simultaneously in replicated circuitry. Substantial circuit complexity and system development effort may thus be eliminated without adversely impacting safety and reliability. And since hardware voting often eventually requires a final set of single-string circuitry or very complex alternative schemes, RC may provide welcome relief from current vulnerabilities.

Any number of sources for single-point failures might exist in future RC devices. While mitigation techniques will address many of these, it should be interesting to see what other new ideas are developed to advance the state of the art.

3. APPROACHES

Crewed Flight

Requirements and considerations for various approaches depend upon particular applications. One primary distinction is of crewed verses uncrewed missions. An argument here is that for crewed systems reconfigurations will be attended, and can thus involve modular, universal electronic units replaced by hand. In the case of uncrewed missions, human interaction will be limited to remote reconfiguration only of installed hardware; the argument that robotic units could perform physical replacements will be neglected for now.

The idea of Hardware Swapping, or direct utilization of Reconfigurable Spares, applies most readily to crewed exploration. This is obvious when one considers availability of a human to pull spare hardware out of inventory or swap compatible hardware from an inactive or less critical system.

Consider the number and variety of vehicles, habitats, and other infrastructure required to accomplish launch, staging, interplanetary operations, AR&D, landing, and operation of remote outposts. It follows that a ready supply of modular spare parts and cannibalized parts could be maintained, assuming a philosophy of Spares Modularity is established now and followed throughout planning and development, and beyond.

Differentiation should be made among spares pulled from other unused or spent or less-critical systems, spares held in un-powered status in other equipment, and spares drawn from storage. In long-term mission applications, it is important to consider circuit lifetimes based on these differences: basic in-service time, time spent installed un-powered or in powered standby status, standard shelf life in exposure to ambient radiation and other hazards, and un-powered storage in shielded vaults. With the capability to physically replace hardware assemblies from a reduced stock, the last option is made much more feasible, however costly the shielding is in terms of payload weight and volume impacts.

Qualification of systems for crewed flight typically is more involved and more expensive. However, given adoption of a universal modular computing resource for both crewed and uncrewed applications and hardware-qualified for both, this concern is minimized.

Uncrewed Operations

Without a direct human presence, ability to make changes to systems must obviously be much more highly automated. As noted, provision of robotic systems to carry out

equipment repair and replacement in lieu of human intervention is a possibility; however, this carries its own set of tradeoffs which will not be argued here.

Humans can still intervene remotely, by uploading revisions or commanding implementation of preset reconfiguration options. This approach works up to a point, depending upon the distance of separation of humans from the system and the skill of designers in predicting contingencies; it becomes an unpalatable option as communication delays become increasingly significant.

One avenue under consideration would apply adaptive or evolutionary techniques. This includes but is not limited to neural networks and genetic algorithms. It is recognized these are controversial technologies, especially in application to space-qualified systems, but their potential should not be overlooked. Furthermore, these approaches are much more likely to be accepted first for use in uncrewed systems than for human-rated ones.

With inability or reduced ability to physically swap spares, the various fault mitigation options mentioned above become much more desirable. Lacking direct ability for hardware swapping, circuit lifetime issues become much more pressing on long-term missions. It thus becomes crucial to consider various techniques for error detection, failure detection, and mitigation: in essence, an arsenal of self-monitoring and self-healing tools.

Concepts in Modularity and RC Implementation

Modularity necessary to realize much of the technological advancement proposed can be implemented in a widely varied number of ways. This section touches on basic philosophies adopted so far in approaching the problem.

Collection of RC modules into useable RC fabric (as illustrated, for example, in Figure 2.) will always require craftiness on the part of designers. The usual tradeoffs must be made: between performance and power consumption, for example. In addition, interconnection of modules may be accomplished in so many different ways as to be prohibitive. Here, a major tradeoff must be made in local interconnection paths amongst individual modules vs. connections necessary for a system-wide bus. Note again that what is represented as bus interconnections here may not necessarily be interpreted as classic bus wires; rather, it is possible for interconnections between modules and clusters to also be reconfigurable resources.

Very closely related developments in FPNA/FPOA technologies are followed with great interest. These should offer tradeoffs in granularity verses ease of implementation in the near term, where fine-grained solutions will likely call for many more years of development prior to delivery of marketable products.

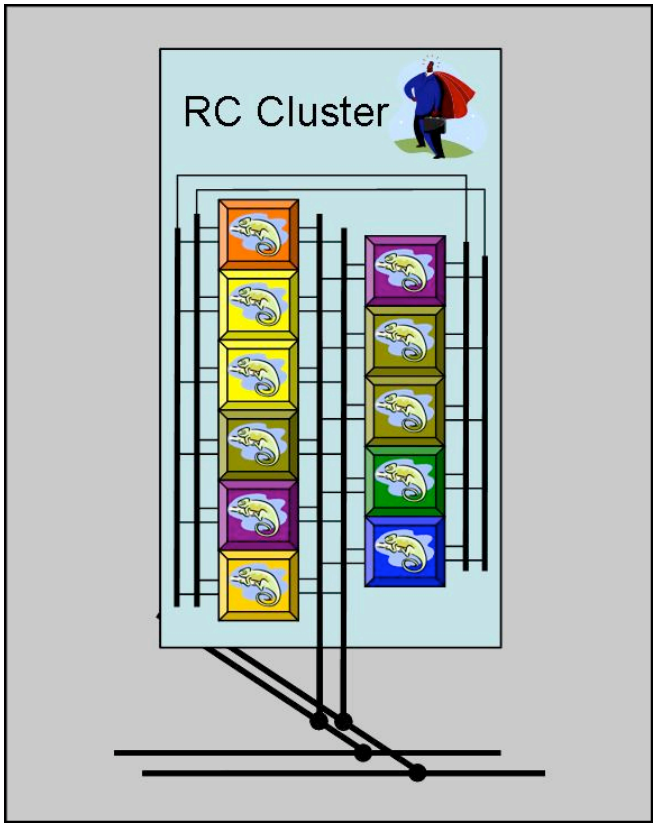

Figure 2 – Building an RC Fabric
With multiple reconfigurable modules, multiple interconnection paths, and redundant busses. Colors/shades indicate different subsystem applications and spare resources. Note that subsystems may cross boundaries or even share modules.

In currently emerging devices, nodes are clustered in sets having a desired representation of various computing strengths. In some cases, the distribution of capabilities is customizable prior to production of target devices. Most nodes are capable of addressing any number of specific applications or algorithms; for example, one type of node may easily work ground communication, on-board communication, and general system interfacing tasks. Or the node might be specialized to a particular type of computation. The specific application might take only a portion of one node in one cluster, or could be distributed across multiple nodes and clusters. The development environment for programming and controlling such a device, or networks of them, is one major challenge in delivering FPNA technology.

Given the assumption RC modules are in fact modular in nature, it follows that repair of failed hardware may be accomplished by replacement of subsets of systems rather than entire systems (Figure 3).

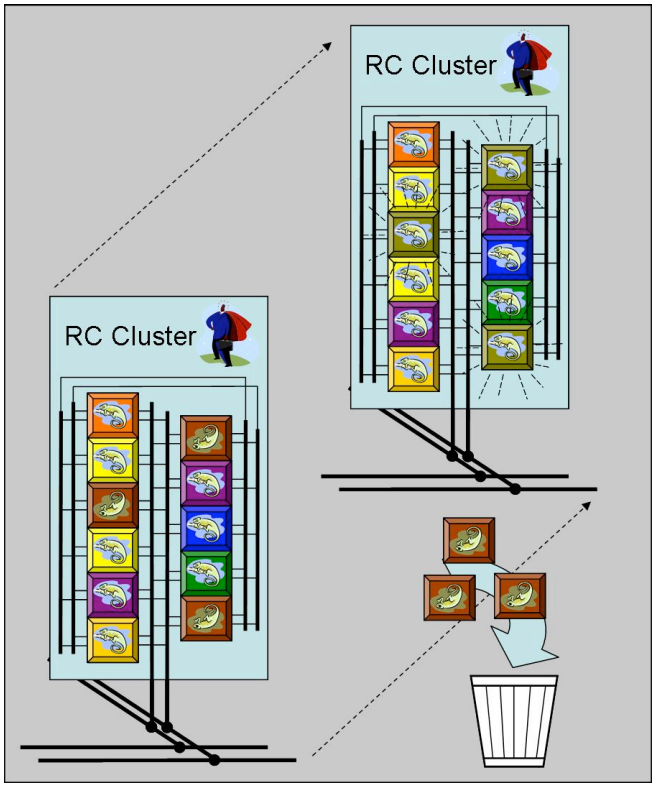

Figure 3 – Physical System Repair
Dead modules may be routed around until no further spares are available

System-Level Approaches

Decisions must be made on such issues as whether to realize entire systems in RC hardware, or to only use RC as supplemental components of conventional systems. Note that in either case, interfacing of primary or backup systems alike must accommodate changes in access to peripheral data acquisition and sensor systems.

Not surprisingly, a variety of plans may be pursued in realizing functional system definition with RC technology. The following figures illustrate conceptually a few of these. The underlying assumption is that initially RC might be adopted only on a very limited basis, but with continued use and familiarity would eventually become all-inclusive.

Because conventional systems usually consist of custom, dedicated electronic boxes connected directly to sensors and other interfaced resources, as Figure 4 shows, applying an

RC system as a modular spare in such an environment might not be as straightforward as hoped. In the event of failure in a primary module, access to these resources would most likely be compromised.

For this reason, it might be necessary to invest in preplanned modifications to system integration strategies. One such change could involve bussing sensor lines and other resources along with regular data bus signals, in order to enable access to their resources by any subsystems requiring them upon reconfiguration (Figure 5).

Ultimately, an RC system could be implemented as a universal resource, as depicted in Figure 6. Here, the term "universal" might be considered multifaceted; that is, the RC system is not only able to replicate any onboard subsystem, but may also be used effectively in nearly all forms of infrastructure. The primary drawback is that this universality must be designed in ahead of time for such broad physical and functional compatibility to be possible.

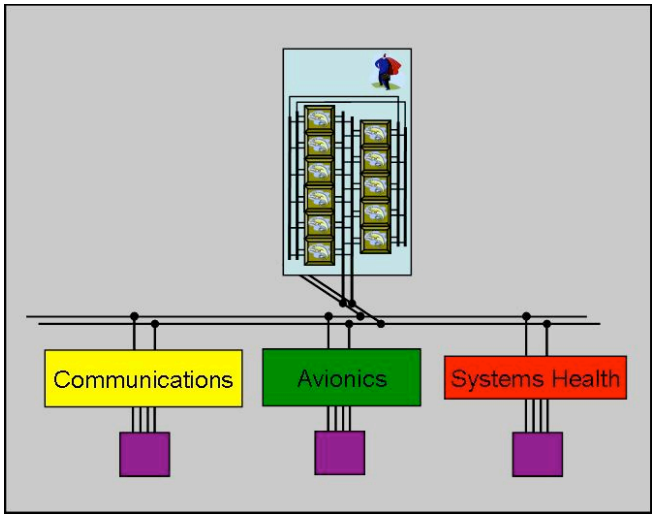

Figure 4 – RC Cluster in a Conventional System
With Conventional Direct Peripheral Interfacing

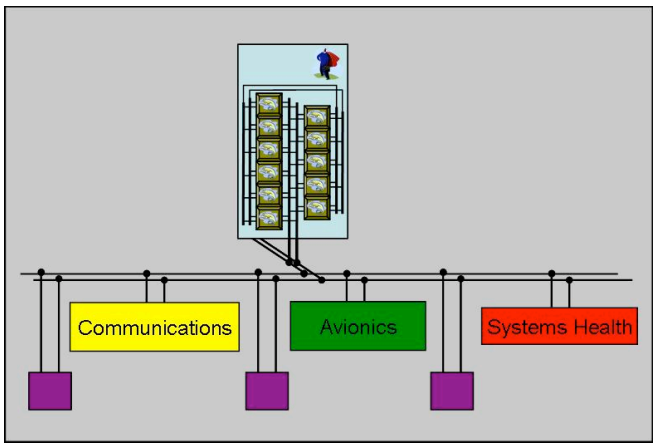

Figure 5 – RC in a Modified Conventional System
Bussed peripheral interfacing enables access of backup system and ensures full recovery upon primary system failure(s)

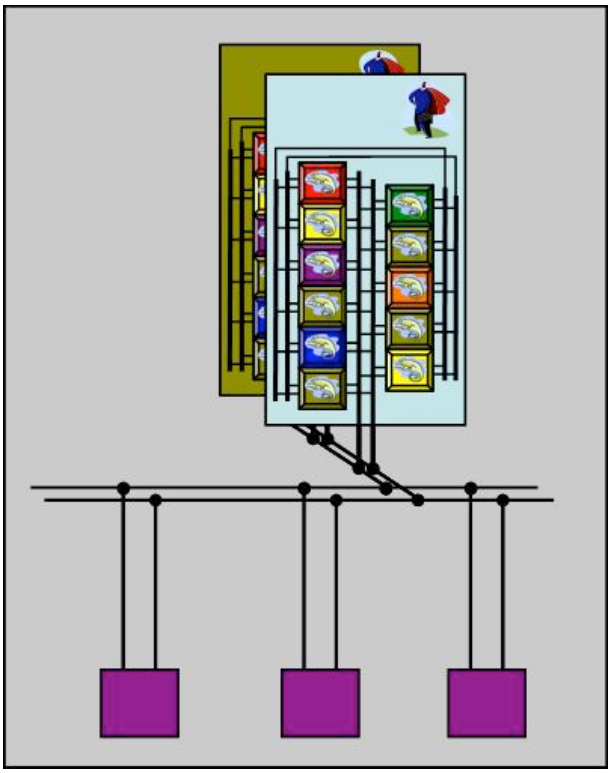

Figure 6 – RC as a Complete System Resource
Additional cluster represents expandable modular reserve circuit capacity

Application Demonstrations

A number of applications have been targeted for demonstration in systems under consideration for impending research. These applications have been chosen for relevance to a wide variety of efforts, in order to showcase flexibility of developing RC systems and to aid in researchers' grasp of concepts.

At this time some of the applications are general while some are more specific. Replication of microcontroller, microprocessor, and digital signal processor capabilities are examples of the former. The latter includes demonstrations of closed-loop control as with a motor, motor controller, and shaft position encoder; video or other image acquisition and processing subsystems; and software-defined radio.

The URTM effort mentioned previously is directed at demonstration of RC-based interface modularity

Further efforts will provide examples of data handling, science data analysis, general number crunching, trajectory projections, or other capabilities relevant to upcoming missions. These will be directed at demonstration of processing modularity.

Integration of interface modularity and processing modularity demonstration efforts within NASA is planned to take place in mid to late 2008. An overall roadmap of the projected RC efforts is shown in Figure 7.

4. PRACTICAL MATTERS

RC technology hasn't yet landed with full force in the real world. Its potential is enormous, and will continue to grow and evolve as the technology begins to mature. In the more distant future, it is conceivable RC systems will displace or even replace the software-based operating system, and software itself, as we know it.

In the meantime, there are some very real problems in implementation. At the forefront is the FPGA technology that essentially started the whole RC effort. While these have exceptional effectiveness and flexibility in limited applications, when incorporated into fabrics their configuration, programming and use become correspondingly more complex. Reprogramming FPGAs while they run – often called partial reconfiguration or run-time reconfiguration – is of immense interest to modern technologists; however, this process is still relatively slow, in that it cannot yet be accomplished between system clock cycles. Furthermore, densities of logic possible at this time are still limited and power consumption is an especially problematic issue. On a basis of performance vs. power alone, FPGAs currently prove to be a poor choice for protracted systems – especially in applications for Space.

On the other end of this spectrum, Application-Specific Integrated Circuits (ASICs) often have remarkable power and space characteristics, but prove too expensive to produce. And a further key point is that they typically can't be reconfigured at all.

Most devices available for research today have little or no accommodation designed in for radiation-related issues. That means few are radiation tolerant, let alone radiation hardened.

For the time being, emerging technologies in field-programmable node arrays (FPNA) or field-programmable object arrays (FPOA) deserve attention. These promise an impressive tradeoff of FPGA flexibility with ASIC speed and low power consumption. Redundant nodes or objects promote such ideas as self-checking, circuit reserves, and self-healing needed for fault-tolerant architectures and systems required to perform throughout long-duration missions.

5. CONCLUSIONS

The relatively new field of RC promises entire sets of new tools for addressing needs of the Space community for avionics and other computing capabilities.

RC has a theoretically unlimited potential for modularity. On a very basic level, modularity makes a lot of sense. Hardware development, flight qualification, square parts logistics, and many other aspects of electronic provisioning become vastly simplified if they only need be done once for multiple systems.

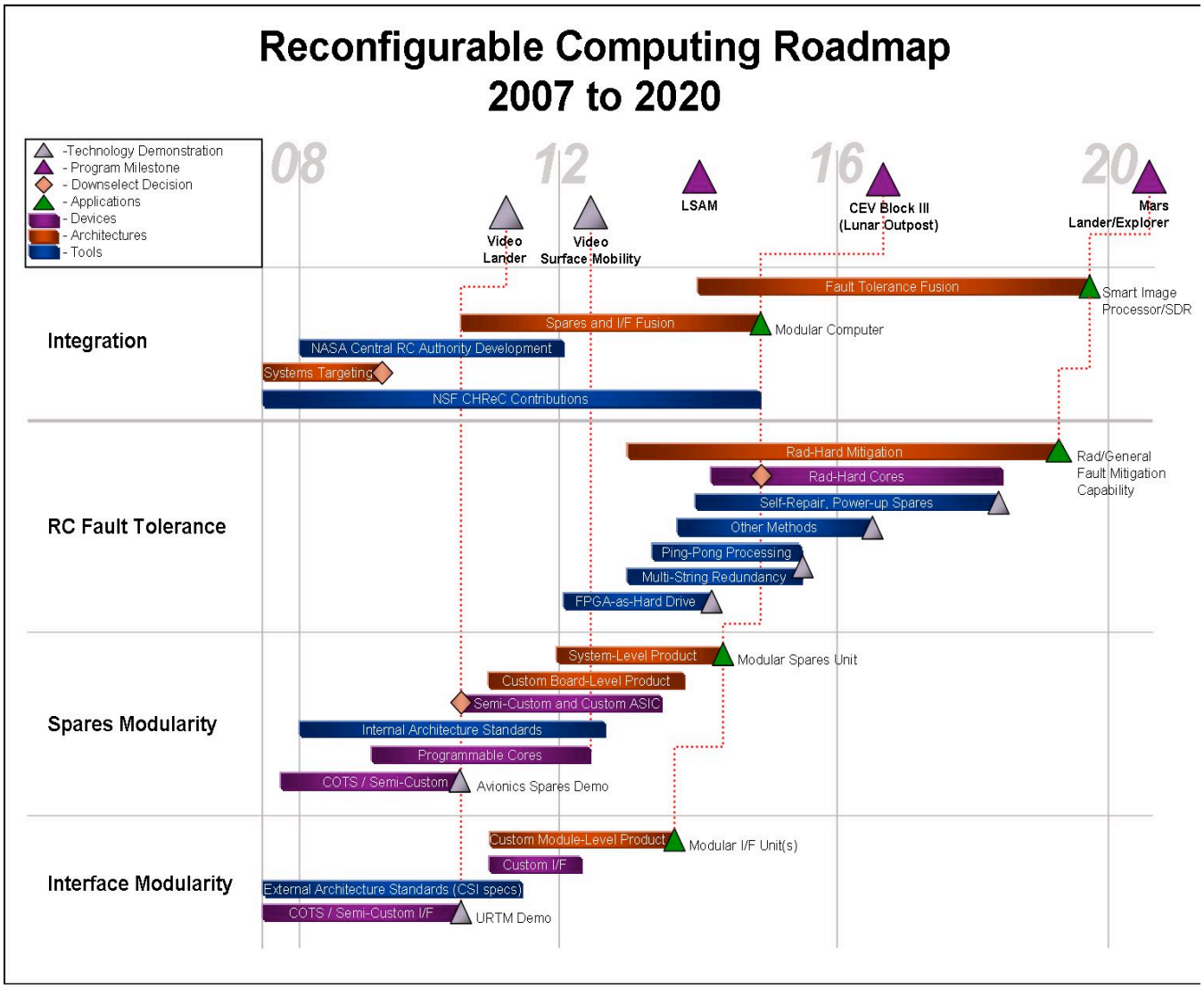

Figure 7 – NASA RHESE RC Roadmap

Nevertheless, it must be admitted implementation of this vision will not be simple. Many systems exist now partly because the many people building them have different ideas about how such things should be done. This technological inertia is typically hard to overcome; so when an entirely new paradigm is considered, whole layers of additional resistance can and should be expected.

Still, the potential for much higher efficiency is tantalizing: universal spares, and the accompanying savings in space, power, payload weight, design time, flight qualification effort, and other costs, are expected to draw enough interest in the near future to move the revolution along to its next exciting stage.

REFERENCES

[1] Robert L. Jones and Robert F. Hodson, *A Roadmap for the Development of Reconfigurable Computing for Space*. Version 2.0, March 2007. Internal NASA document (currently unpublished); contact this paper's author for access.

[2] Clive "Max" Maxfield. *The Design Warrior's Guide to FPGA*. Burlington, MA: Newnes, 2004.

[3] NASA contract number NNL07AA25C.

BIOGRAPHY

Clint Patrick is an Electronics Engineer at MSFC. He has been with NASA more than twenty-six years, working in electronic data systems; circuit design, layout and production; FPGAs; system integration; system software development; spectroscopy; artificial neural networks; and reconfigurable computing. He is currently NASA's RC task lead for the RHESE project.