Reconfigurable Computing in Remote and Harsh Environments

Gordon Brebner

Division of Informatics
University of Edinburgh
Mayfield Road
Edinburgh EH9 3JZ
Scotland

Neil Bergmann

Cooperative Research Centre for Satellite Systems
Queensland University of Technology
GPO Box 2434
Brisbane 4001
Australia

Abstract. Reconfigurable computing based on field-programmable logic offers exciting new options to the traditional system design space, yet has still to prove itself as a means of delivering performance improvement on the desktop. It seems that apparent speed-ups require much design effort, and can then be challenged by fast microprocessors or eclipsed by current computer system architectures. In this paper, a challenging application domain is investigated — computing on a spacecraft. Here, the environment is somewhat different from the typical desktop, in that real-time constraints and power consumption constraints suggest the use of circuitry rather than programs, and the lack of physical access demands configurability of the circuitry. This application domain is of genuine contemporary interest, and the paper explains how the authors propose to apply some of their earlier reconfigurable computing research, aiming at real system improvements.

1 Introduction

Like many promising high-speed computing technologies, reconfigurable computing is struggling to gain widespread acceptance in the general desktop computing market. While many researchers have reported significant algorithm speedups with the use of FPGA technology in fairly specific applications (e.g. image processing) generally results from reconfigurable computing have been disappointing. We explore some possible reasons for this, and identify a new application domain where reconfigurable computing shows promise: computing in space[1].

As a background to the nature of this application domain, it is useful to make some initial observations about computing systems in general, and the possible roles of reconfigurable computing. Computing system components can be categorised as follows:

– transformational: these components input data, and then output results computed using the data or translated from the data, at speeds determined by the computer and at times chosen by its user;

[1] That is, computing outside the Earth's atmosphere, rather than 'computing in space' using circuits as an alternative to 'computing in time' using programs!

– interactive: these components continually interact with a human user and the environment, at a speed and in a sequence determined by the computer and its user; and
– reactive: these components continually interact with the environment, at a speed and in a sequence determined by the environment.

Performance constraints for each of these types are different, and this greatly affects the usefulness of techniques which speed up algorithm execution, such as reconfigurable computing.

A large percentage of reported reconfigurable computing applications fall into the category of transformational system components. Such components typically have a vague performance constraint, namely that results are desired 'as soon as possible', and so any performance improvement derived from reconfigurability is useful, but has to be tempered by the effort required to achieve that improvement. Experience has shown that a very large development effort is required for fairly modest performance improvement, so the use of reconfigurable computing is not compelling at present.

For interactive components, especially GUI-based interfaces, there appears to be little benefit in using reconfigurable computing technology. The operational time-beat of such components is based on human response times. In terms of computation time, much of it is spent in the uninspiring task of shuffling data to the graphics controller; other compute-intensive sub-tasks are best regarded as transformational sub-components, and the comments above apply to these.

In contrast, for reactive components, the real-time constraints imposed by the environment often involve hard deadlines, and failure to meet these can be catastrophic. In practical terms, if a purely software approach cannot meet timing deadlines, then additional hardware must be added. In such cases, the designer may be willing to expend considerable extra design effort to gain a modest performance improvement in just one small component of a system, if this means that system deadlines are met. Given this compelling need in the traditional world of software/hardware systems, it is worth exploring the domain of reactive components in general, as a fruitful area for application of reconfigurable computing techniques, in which a more flexible view of 'hardware' and 'software' is possible.

In this paper, one particular application area within the reactive domain is considered: computing in space. Consider the characteristics of an unmanned spacecraft in operation. Such a system must operate in a remote and harsh environment where no physical repairs, modifications or upgrades can be made. The only contact with the spacecraft is via a radio link, often unavailable for long periods of time, often of low bandwidth and often with high error rates.

There are some features of this application that immediately suggest the use of configurability and programmability on a fairly coarse time grain, never mind the use of subtle reconfigurable computing. This is because it is often desirable to change the functional characteristics of a spacecraft in flight, perhaps to participate in a new mission, perhaps to take advantage of improved knowledge about some phenomenon, perhaps to overcome the effect of a faulty piece of equipment.

Where functions are necessarily implemented using circuitry, such changes can only be enabled by the use of programmable logic technology, because there is no possiblity of physically swapping chips, as in a terrestrial system. For a spacecraft, all management of the reconfiguration must be handled remotely through appropriate communications protocols.

In addition to investigating the problem of 'straightforward' reconfiguration, but at great distance, there are also opportunities for exploiting dynamic reconfigurability within the spacecraft system components during normal operation, in order to cope with the unusual operating circumstances of the remote and harsh environment that is space. Such methods may prove necessary not only because of the reactive time constraints but also because of constraints on system resources. Overall, this application domain exhibits very different characteristics to most others, and so requires and justifies rather different approaches to system design. In particular, it offers an opportunity to demonstrate the practical relevance of some of the more esoteric previous reconfigurable computing research carried out by the authors and others. Lest the reader may think otherwise, it should be stressed that computing in space is no artificial application area being pushed forward as the latest potential 'killer app' for reconfigurability; this fact is evidenced by the current interest being shown by organisations such as NASA and others, e.g. [9, 14].

The rest of the paper is organised as follows. Section 2 considers the environmental characteristics in more detail, in order to motivate the novel approaches required. Then Section 3 examines reconfigurable computing techniques that are apt for the spacecraft system itself, and Section 4 examines techniques for the required communication between ground control and the spacecraft, or between spacecraft. Section 5 briefly describes the practical details involved in constructing appropriate systems, and a first experiment involving a spacecraft that is under way. Finally, Section 6 contains some conclusions about this work.

2 Environmental characteristics

First, it is worth noting that space electronics must operate in very hostile conditions: for example, high mechanical and acoustic vibration during launch, microgravity, near vacuum, and a wide temperature range. The biggest problem is the high level of ionising radiation that the electronics is subjected to, both because this radiation causes soft errors, such as a few bit-flips in memory storage each day, and also because accumulated crystal damage eventually causes permanent failure. These two characteristics mean that special techniques are required when using reconfigurable logic technology in space. Such techniques are not considered further in this paper, but are discussed in detail in [8], as are the sorts of specialised FPGA chips required. Here, the interest is in how the technology is used.

One key influence on computing in space is that there may be very small power budgets, for example, as little as 20W in total. This makes the use of lower power consumption 8-bit or 16-bit microcontrollers more attractive than higher power consumption 32-bit microprocessors, for computing using sequential programs. Until recently, the typical processing capability on spacecraft was limited to little more than sensors, solid-state data recorders and communication devices to relay the data back to ground control. The use of programmable logic opens up the possibility of adding reactive functionality, achieved through the use of programmed circuitry, that cannot be achieved with a low-grade programmed processor. Examples include the use of smart sensing instruments, with reactive processing resources attached to the sensor, and the use of intelligent filtering on sensor data. Thus, there can be benefits that would not be possible under the more relaxed constraints of the terrestrial environment, where Shand, for example, has shown that it is hard for configurable logic to compete with fast microprocessors [15]. In addition to power constraints, there are constraints on mass and volume, which limits the physical size of the technology used, something which impacts on programmable logic, microcontrollers, microprocessors and memories alike. Advances in silicon technology, and the advent of system-level integrated chips, are of assistance in this matter.

The fact that this circuitry should be programmable, rather than supplied as custom hardware, follows from the remoteness feature of the environment, as discussed in Section 1. It means that both circuit and program errors can be repaired while the spacecraft is in flight, which allows reduced mission costs by minimising risks, especially for missions with very short lead times, but long lifetimes. However, the configurability is not just of use when correcting errors, but also allows ground control to adapt spacecraft functions in response to data observed as a mission progresses. A further justification is to allow the possibility of component re-use, or partial re-use, over a number of missions. Given that only a small number of spacecraft of a particular design (possibly only one) are ever built, there is a saving on bespoke hardware costs, and also an incentive for evolutionary innovation on each mission.

As far as communication between a ground station and a spacecraft is concerned, there could hardly be a more hostile environment. The wireless radio links used are low bandwidth, will have high latencies (seconds, minutes, even hours) for distant environments, may be unavailable for extended periods depending on orbits, and may have high error rates. Therefore, from a reconfigurability point of view, any communication must be modest but include protocols that ensure some guarantee of reliability. Ideally, this communication itself must embody some degree of configurability, to allow adaptation to unexpected circumstances during a space mission. Although the quality of communication may be low, there are still real-time demands that require circuitry rather than programs, e.g. signal modulation and demodulation for soft-radio technology.

Typically, unmanned exploration spacecraft are used to sense some aspects of the physical environment, either locally (as with a magnetometer) or remotely (as with a space telescope). In the future, it is expected that space missions will involve cooperating constellations of spacecraft, combining their observations to generate a coordinated view of particular space phenomena (for example, a near-Earth comet approach). It will not be possible to control such distributed computing systems from a ground station, so it will be necessary for the spacecraft to communicate, and so cooperate, amongst themselves. This poses additional computing and communication problems.

In summary, the remote and harsh environment that spacecraft operate in offers new challenges and opportunities for reconfigurable computing: on one hand, a rationale for the use of soft circuitry; on the other hand, a rationale for the use of lightweight, but flexible and powerful, communication protocols.

3 System techniques

In order to examine the potential uses of reconfigurable computing within components of spacecraft systems, it is useful to follow the three-level view of programmability for data flow models (i.e., circuits), suggested in [6]:

1. different circuits can be executed;
2. executing circuits can be modified;
3. dynamic behaviour through choice in data flows.

In general, there seems a much stronger case for programmability at the second level for circuits than there is for programs, largely because of the relative weakness of the third level for circuits.

3.1 Execution of different circuits

The first level of programmability encompasses most of the more routine uses of configurable logic, for example, prototyping ASICs and infrequent field upgrades. This sort of coarse-grain programmability is encouraged by the majority of FPGA technologies that do not allow partial reconfiguration. In the case of spacecraft, there is a need for such programmability, but it comes at a cost. For normal operation, there may be a range of different functions required over time, and complete circuit swapping is an efficient use of the programmable logic resource, given constraints on its size. For repairing severe faults, or performing upgrades, circuit swapping will be necessary.

The penalty for complete circuit swapping arises from the need to load new configuration data into the programmable logic device. In essence, there are two possible sources of the data: storage within the spacecraft system; and storage at the ground station. In the case of data stored locally, the general interfacing and control for reconfiguration is of the same type as used in normal terrestial systems. Added constraints are that the size of memory devices may be severely limited, and that a microprocessor or microcontroller supervising the task may be less powerful. Also, either the stored data is fixed from the time of the spacecraft's launch, or has been updated from a source at the ground station. These constraints suggest that the use of circuit swapping based on locally-stored data has to be used modestly and carefully.

The situation for circuit swapping based on remotely-stored data is equally challenging, but for different reasons. Clearly, there can be an effectively infinite storage facility and relatively unlimited processing capability at the ground station. However, the problem is that configuration data then has to be sent to the spacecraft via an unhelpful communication medium, and some degree of processing will be needed at the spacecraft end to process the received data. Some of the communication issues are addressed in the next section. However, a fairly obvious conclusion is that one does not want to make circuit swapping based on remotely-stored data a frequent operation. In practice, the best solution to enable circuit swapping is to use local memory on the spacecraft system as a cache between the configurable logic device and storage at the ground station. The cache would be relatively stable, in the sense that updates would occur at frequencies measured in hours at the fastest, more likely in days or weeks.

3.2 Modification of executing circuits

Given that there are problems associated with the volume of data required to swap complete circuits, here there is a particular motivation to consider modification, rather than replacement, of circuits. This is in addition to any time-related benefits that may result from such approaches. An essential underpinning for circuit modification is the use of a programmable logic technology that allows partial configuration, ideally while the circuit is still in operation. Two general approaches to circuit modification have been studied by the authors, and others, and both look to offer real benefits in this application domain, although the physical constraints mean that attention to parsimony is crucial.

The first approach is one where there is automatic system support for circuit modification, in a manner intended to be invisible to circuit designers. This encompasses the idea of virtual circuitry (also, and rather more commonly, known as 'virtual hardware'). Here, the general idea is to allow circuits to be larger than the actual area of configurable logic that is available, in a way akin to the use of virtual memory. This is particularly attractive in an environment of limited physical resource, both in terms of circuit area but also in terms of limiting power consumption only to parts of a circuit that are currently active; clearly it is neceassary to amortise the reconfiguration power overhead over time. Many of the issues reduce to the need for some form of inexpensive placement and/or routing of circuitry dynamically at run time, and an approach to dealing with this has been suggested by one of the authors in a series of papers [1, 3, 4]. The central concern is to ensure that the overhead incurred by using a microkernel to manage the virtual circuitry is kept to the minimum. In the author's model, this means ensuring that 'Swappable Logic Units' have very regular sizes, are placed in a very regular arrangement, and are supplied with very regular routing resouces. In time, it is to be hoped that some of the crucial functionality might be incorporated into the programmable logic technology itself.

The second approach is one where there is system support for circuit modification using run-time facilities anticipated by the circuit designer, and designed harmoniously. For example, modification of executing circuits can be used to achieve an effect similar similar to parameter-passing, by 'folding' parameter values into the functionality of the circuitry. This has been studied by various authors, using various terminologies, including 'partial evaluation' for example.

It is also possible to change data flows by modifying interconnections between circuit components. The particular interest of the authors revolves around means of enabling dynamic circuit modification in a relatively dignified manner, distinct from writing binary configuration data as patches to circuitry generated by automatic plance and route tools with histories in ASIC design. Essentially, the approach revolves around circuit design and circuit modification at a uniform level of abstraction that is below the typical high level used by designers and above the typical low level used by reconfigurers. The CHASTE system of one of the authors [2] is an example of this approach in practice, as is the JBits (formerly known as JERC) system of Guccione [12]. This approach appears very promising for the sort of minimalist, precision circuit modification needs in the spacecraft application domain.

3.3 Summary

The authors strongly believe that computing in space offers a challenging, but ultimately fruitful, opportunity to demonstrate that their earlier work on soft circuitry, including virtual circuitry, has practical benefits. The third level of programmability listed above — dynamic behaviour through choice in data flows — is not considered further here, because it seems inappropriate apart from in very limited situations. Circuit size is very constrained, so it is not desirable to have 'dead' circuitry that is consuming power but not contributing to the computational data flow.

In [6], a very general view of design flows is presented, of which the three-level programmability is but one feature. Key components are control flow/data flow co-design (i.e., program/circuit co-design), algorithm/architecture co-design and softness/hardness co-design, embedded in a multiple-layer system architecture. This generality is a little grand for the necessarily parsimonious application here, but its central points have relevance. That is, the apt choice of what is best computed by programs and by circuits, the use of supportive architectures, and the commitment to hardness only when absoutely necessary.

4 Communication techniques

The essential problems involved in communication between a ground station and a satellite are taking account of high latencies, irregular availability and low bandwidth as operational constraints, together with unreliable transmission as a problem that has to be solved. The first two constraints rule out any use of communication that assumes either semi-real time or semi-continuous behaviour; the third constraint means that the amount of data transmitted has to be minimised, in particular the amount of configuration data in this case. Application-specific data compression has a significant role to play in ensuring that communication is minimised.

The unreliability problem for conventional communication channels is usually solved by acknowledgement-retransmission protocols. However, the high latencies involved here mean that this is not a good solution because of very high round-trip times for messages. A better solution is the use of error correction codes adding redundancy to the transmitted information — here, there is a tension with compression, since the redundancy uses up precious bandwidth. A feature of both error correction codes and compression is that appropriate computational power is needed at the spacecraft end of communication. Reconfigurable computing may be able to assist here, for example, circuitry for evaluating error correction codes is relatively straightforward.

In terms of application-specific compression, there is some existing work that can be built upon, although this tends to be FPGA-specific. Hauck et al [11] demonstate a method for compressing configuration data for the Xilinx XC6200 that exploits the built-in wildcarding facility allowing parallel configuration of multiple FPGA cells. Luk et al [13] discuss a method for computing differences between array configurations that exploits the partial reconfiguration capability of the XC6200. Both of these methods require non-trivial computation at the transmitter end, but no extra computation at the receiver end, which is ideal for this application.

A longer-term goal is to represent configuration information in a more compact and higher-level form, and there is initial discussion of such an approach in [5], which points to the desirability of standard 'instruction set architecture' equivalents to emerge for configurable logic. The aim is to introduce the concept of 'circlets' — a circuit analogue of program applets. It is intended that circlets will be investigated by the authors for communication in space, the basic trade-off being between compactness and portability of representation on one hand, and the computational needs of circlet interpretation on the other hand.

As well as communication protocols between ground stations and spacecraft, there is also a need for protocols for inter-spacecraft communication in multiple-craft missions. The characteristics of the communication medium are distinctly different in the latter case, and in some ways are more akin to terrestial communications. For implementing either type of communication protocol, to allow flexibility and evolution over time, the authors plan to investigate an 'active protocol' [10] approach, whereby new protocols can be dynamically implemented by assembling standard blocks that are in circuit and/or program form. This will add a 'configurable protocol' element to the overall picture of achieving adaptability at a distance.

5 Practicalities and experiments

The authors' first experiment to put these ideas into practice will involve a small low earth orbit satellite, FedSat [7], to be launched in 2000. This will include a board containing space-hardened SRAM-based FPGA chips together with control circuitry, forming the 'Adaptive Instrument Module' (AIM). The technical details of the AIM are discussed in [7]. Part of the experimentation will concern the investigation of techniques for dealing with radiation-induced errors. As far as reconfigurability is concerned, experiments will include:

- ensuring that basic communication mechanisms work;
- optimising the communication needed for reconfiguration;
- testing spacecraft-initiated dynamic reconfiguration; and
- investigating the practical usefulness of soft circuitry for achieving evolving mission requirements.

The entire reconfigurable computing set-up will be experimental so, although it will be tested as exhaustively as possible in a terrestial setting prior to launch, an essential feature will be crude hard-wired mechanisms for resetting the AIM to a known initialised state with a guaranteed capability for being configured from the ground station.

6 Conclusions

Clearly, the practical investigations associated with this work are some time off. However, this application area is not one in which researchers can do quick and early experiments in the real harsh and remote environment. What this paper has shown is that the unusual features of this environment make reconfigurable computing seem an attractive solution, in ways that are not applicable to a normal desktop setting. In particular, various reconfigurable computing ideas presented by the authors as the fruits of earlier research look like being able to make a genuine practical contribution.

Restrictions on power, mass and volume remove the microprocessor competition that typically threatens reconfigurable logic solutions which are proposed in order to meet real-time constraints. This gives a case for using soft circuitry rather than soft programs or, more likely, for using a combined soft circuitry/program approach that can be supported within the physical constraints. Limitations on circuit size makes the use of virtual circuitry an attractive possibility.

There are also challenges arising from the very limited communication interface between ground control and a spacecraft. This creates a heightened version of the 'system bus' bottleneck that threatens performance gains when reconfigurable logic chips are used alongside microprocessor chips. The solutions devised for appropriate communication will point to improved ways of within-system interfacing, in addition to coping with the difficulties of operating over very long distances in very harsh surroundings. In the future, communication between distant spacecraft themselves will also present challenges.

Acknowledgement

The support of the Australian Commonwealth Government and the Queensland State Government, through the Cooperative Research Centre for Satellite Systems, is gratefully acknowledged.

References

1. Brebner G, "A Virtual Hardware Operating System for the Xilinx XC6200", Proc. 6th International Workshop on Field-Programmable Logic and Applications, Springer LNCS 1142, 1996, pp.327–336.
2. Brebner G, "CHASTE: a Hardware-Software Co-design Testbed for the Xilinx XC6200", Proc. 4th Reconfigurable Architecture Workshop, ITpress Verlag, 1997, pp.16–23.
3. Brebner G, "The Swappable Logic Unit: a Paradigm for Virtual Hardware", Proc. 5th Annual IEEE Symposium on Custom Computing Machines, IEEE Computer Society Press 1997, pp.77–86.
4. Brebner G and A Donlin, "Runtime Reconfigurable Routing", Proc. 5th Reconfigurable Architecture Workshop, Springer LNCS 1388, 1998, pp.25–30.
5. Brebner G, "Circlets: Circuits as Applets", Proc. 6th Annual IEEE Symposium on Custom Computing Machines, IEEE Computer Society Press, 1998, pp.300–301.
6. Brebner G, "Field-Programmable Logic: Catalyst for New Computing Paradigms", Proc. 8th International Workshop on Field Programmable Logic and Applications, Springer LNCS 1482, 1998, pp.49–58.
7. Brown J, S Gardner, A Wicks, L Boland and E Graham, "The FedSat Spacecraft Design", 49th International Astronautical Congress, Melbourne, Australia, October 1998.
8. Dawood A and N Bergmann, "Enabling Technologies for the Use of Reconfigurable Computing in Space", submitted to FCCM'99.
9. Figueiredo K, K Winiecki, T Graessle and U Patel, "Study and Utilisation of Adaptive Computing in Space Applications", NASA GSFC, MAPLD Conference, September 1998.
10. Haeck S and G Brebner, "Active Protocols for Active Networks", submitted to 1st International Working Conference on Active Networks, Berlin, June/July 1999.
11. Hauck S, Z Li and E Schwabe, "Configuration Compression for the Xilinx XC6200 FPGA", Proc. 6th Annual IEEE Symposium on Custom Computing Machines, IEEE Computer Society Press, 1998, pp.138–146.
12. Lechner E and S Guccione, "The Java Environment for Reconfigurable Computing", Proc. 7th International Workshop on Field-Programmable Logic and Applications, Springer LNCS 1304, 1997, pp.284–293.
13. Luk W, N Shirazi and P Cheung, "Compilation Tools for Run-Time Reconfigurable Design", Proc. 6th Annual IEEE Symposium on Custom Computing Machines, IEEE Computer Society Press, 1997, pp.56–65.
14. Robinson S, M Caffrey and M Durham, "Reconfigurable Computer Array: the Bridge between High Speed Sensors and Low Speed Computing", Proc. 8th International Workshop on Field Programmable Logic and Applications, Springer LNCS 1482, 1998, pp.159–168.
15. Shand M, "A Case Study of Algorithm Implementation in Reconfigurable Hardware and Software", Proc. 8th International Workshop on Field Programmable Logic and Applications, Springer LNCS 1482, 1998, pp.333–343.