Development of a Satellite Sensor Network for Future Space Missions

Tanya Vladimirova, Xiaofeng Wu and Christopher P. Bridges
Surrey Space Centre
Department of Electronic Engineering
University of Surrey, Guildford, UK, GU2 7XH
{T.Vladimirova, X.Wu, C.P.Bridges}@surrey.ac.uk

Abstract—Future spacecraft are envisioned as autonomous, miniature, intelligent and massively distributed systems. At the Surrey Space Centre, a research project is currently under investigation, which aims to develop a picosatellite sensor network using the CubeSat platform. The proposed satellite sensor network will be used to demonstrate technology advances in space, including modified IEEE 802.11 wireless standard for inter-satellite links (ISL), distributed computing for computationally intensive on-board signal processing, and reconfigurable system-on-a-chip (SoC) design. [1,2]

TABLE OF CONTENTS

1. INTRODUCTION

The research presented in this paper is part of the ESPACENET project [1], which aims to develop enabling technologies for future distributed space architectures based on flexible, reconfigurable, evolvable, and intelligent multi-spacecraft sensing networks. "Evolvable Networks of Intelligent and Secure Integrated and Distributed Reconfigurable System-On-a-Chip Sensor Nodes for Aerospace Based Monitoring and Diagnostics", or ESPACENET for short, is funded by the UK research council EPSRC. It targets the development of a robust satellite sensor network based on flexible *picosatellite* nodes or *piconodes*.

Future spacecraft are envisioned as autonomous, miniature, intelligent and massively distributed space systems [2]. The Satellite Sensor Network is proposed, which applies the concept of terrestrial wireless sensor networks to low Earth orbit (LEO) spacecraft. The concept of satellite sensor networks can be applied to many space missions. Some examples are the following:

- realising co-orbiting assistants/inspectors of larger mother ships;
- providing continuous Earth coverage for multi-point remote sensing, monitoring or communications at low cost in LEO;
- providing continuous communications for multiple low-powered surface vehicles around the Moon, Mars and other planets or asteroids.

The research carried out at Surrey Space Centre is aimed at both node and network levels. At the node level, picosatellites based on the CubeSat [3] bus are being built. For fast prototyping, commercial-off-the-shelf (COTS) components/boards are chosen, including the flight module, MHX transceiver and satellite chassis from Pumpkin [4], power module from Clyde-Space [5], and a GPS module from SSTL [6]. The payload subsystem needs to be customized to perform the proposed satellite mission objective. At the network level, a wireless COTS protocol for inter-satellite communication is designed based on the terrestrial IEEE 802.11 standard [7]. A multi-objective optimization algorithm is used to optimize signal routing within the satellite network [8]. Computationally intensive tasks for virtual satellite missions are selected, which are to be processed using a distributed approach [9].

The remaining paper is organized as follows. Section 2 introduces the proposed satellite mission. Section 3 focuses on the network-level architecture, including inter-satellite links, routing algorithms and distributed computing. Section 4 presents some simulation results. Section 5 looks into the detailed payload hardware architecture and Section 6 concludes the paper.

2. SATELLITE MISSION

As a practical output of the ESPACENET project we aim to prototype a picosatellite based on the CubeSat platform and commercial-off-the-shelves (COTS) components for a distributed satellite mission consisting of three such picosatellites in LEO. Two launch opportunities are available to us: 1. Via the CubeSat program [10], in which more than 60 universities take part. 2. Via Surrey Satellite

[1] 1-4244-1488-1/08/$25.00 ©2008 IEEE.
[2] IEEEAC paper #1263, Version 4, Updated October 26, 2007

Technology Limited (SSTL), a spin-off company of the University of Surrey, who have indicated that they would be willing to provide a flying opportunity on one of their future micro-satellites, which could act as a mothership releasing the picosatellites in space at particular times. Exact orbital parameters would be determined as the project progresses with the launch provider.

The mission objective is to demonstrate and test novel technologies resulting from the project. The following technologies, which are under development at present, will be demonstrated:

1. Inter-satellite links capable of supporting standard wireless protocols

2. Distributed computing over a network of 3 satellites, which is capable to support three applications:

    a. Simple collaborative imaging including compression/decompression and image processing tasks, e.g. sending of an image or part of an image to another satellite for processing; pattern recognition on the whole image, provided by all the three satellites, etc.

    b. Measurement and processing of the electromagnetic spectrum of the atmosphere using a reconfigurable MEMS antenna [11].

    c. Running and computing of multi-objective evolutionary optimization algorithms (MOEA) for signal routing.

3. Secure processing at node and network level

4. Reconfigurable computing in space based on a system-on-a-chip (SoC) on-board controller implemented in an FPGA.

3. Design of a Satellite Sensor Network

The ESPACENET project aims to develop an ad-hoc network with picosatellites in LEO. The mass of such satellites is less than 1 *kg*. Sensing and on-board computing are also expected within the satellite network. Fig. 1 shows the network architecture of the satellite cluster. The master node is a larger picosatellite or a microsatellite; and all the slave nodes are picosatellites. Within one cluster, the master satellite serves as the cluster head, and controls the other slave satellites. All the communications between the satellite clusters, and the ground station take place via the master satellite. The slave satellites together with the master satellite are grouped as an ad-hoc network, which allows peer-to-peer communication between two nodes.

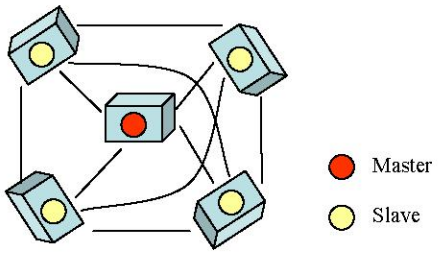


Figure 1. A satellite cluster with master and slave nodes

The satellites within a cluster should work together to perform distributed satellite missions. And in order to support these missions, several research topics are investigated, including inter-satellite link (ISL), distributed computing, and multi-objective optimization for signal routings.

Inter-satellite Links

The inter-satellite link is established via wireless communication protocols and based on the terrestrial IEEE 802.11 standard. However the IEEE 802.11 standard normally supports a typical communication range within 300m. For space, the separation between satellites should be more than at least 1 *Km*. Hence the protocol must be modified to meet this requirement.

The main access technique of IEEE 802.11 is called Distributed Coordination Function (DCF), which is based on the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) and a random back-off interval following any busy medium condition. The CSMA/CA protocol specifies that for a node to transmit, it will first sense the medium around it to determine whether another node is transmitting. If the node in question finds that the medium is free, then that node will begin transmission. The CSMA/CA algorithm also mandates a delay between contiguous frame sequences. A transmitting node will make sure that the medium is idle for this required duration of time. Nodes will each select a random back-off interval to initialize a counter, and will decrement this counter while the medium remains idle. Once a node has determined that the medium is idle, short inter-frame space (SIFS) and DCF inter-frame space (DIFS), are defined, to provide delays between sending and receiving request to send (RTS), clear to send (CTS), DATA and Acknowledgement (ACK) packets.

The IEEE 802.11 standard defines SIFS 10 μS, and DIFS 50 μS. However, these timings are only meaningful for indoor use of 300m range. For a minimum 15 *Km* communication range the Medium Access Control (MAC) inter-frame timing for control and data transfer need to be re-defined for an inter-satellite link [12]. The IEEE 802.11 standard defines the timing parameters according to the following equations [13]:

$$SIFS = RxRFDelay + RxPLCPDelay +$$

$$MacProcessingDelay + RxTxTurnaroundTime,$$
$$SlotTime = CCATime + TxTxTurnaroundTime + AirPropagationTime + MacProcessingTime,$$
$$DIFS = SIFS + 2 * SlotTime,$$
$$AckTimeout = frameTXtime + AirPropagationTime + SIFS + AckTXtime + AirPropagationTime. \quad (1)$$

From equations (1), the DIFS, SlotTime, and AckTimeout depend on the AirPropagationTime. By increasing the distance between nodes, the AirPropagationTime increases and all the parameters have to be redefined. Table 1 compares the parameter between the IEEE 802.11 standard timing and the inter-satellite link timing with 15 *Km* communication range.

Table 1. Comparison of inter-frame timing of the IEEE 802.11 standard and the inter-satellite link

Parameters	IEEE 802.11 (μS)	Inter-satellite link (μS)
SlotTime	20	75
SIFS	10	33
DIFS	50	160
CCATime	<15	15
RxTxTurnaroundTime	<5	5
AirPropagationTime	1	55
MacProcessingTime	0	0
AckTimeout	1	1729

Distributed Computing

Picosatellites have very limited resources in terms of power, processor speed, and memory. It is not viable to perform computationally intensive tasks like multi-objective optimization on a single node. However the processing capability of individual picosatellites could significantly be improved by distributing such tasks to multiple nodes.

Distributed computing requires faster development cycles, decreased effort, and greater software reuse, which motivates the creation and use of *middleware* and middleware-based architectures [14]. Middleware is a software layer that resides between the application programs and the underlying operating systems, including network protocol stacks. Network middleware mediates between an application program and a network as illustrated in Fig. 2. It manages the interaction between applications across heterogeneous computing nodes.

Agents are a programming abstraction, which is used for middleware design in networked systems. By thinking behaviourally (assigning an agent a 'role'), communication interactions and autonomous actions become easier to realise. This allows them to work 'proactively' and 'reactively' to their environment and to any given task. They can be proactive when finding new communications routes in a networked environment and reactive to disconnections, low bandwidths or high latencies [15].

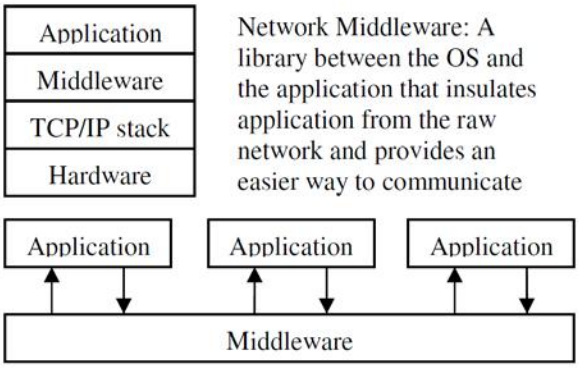


Figure 2. Network middleware structure

There are many potential benefits to using Agents on-board spacecraft. These include (but not exhaustively) the following:

- Increasing the level autonomy through decomposition of high-level goals;
- Increasing the flexibility and adaptability of flight software through dynamic agent uploads
- Improving the reliability of spacecraft and clusters of spacecraft by incorporating fault detection at both high and low levels thus reducing the need for large ground support organizations.

In a satellite network mobile software Agents could be used for the following:

- **Deployment, Upgrading and Maintenance:** The exploitation of mobile code to support software deployment and maintenance is very useful in a highly distributed environment. Mobile code paradigms would allow the action of configuration of new software and hardware units, such as Intellectual Property (IP) cores in a FPGA, or rebuilding of an application with new parameters locally at each satellite. Proactive reconfiguration by comparing both hardware characteristics and software features for configuring and installation could be automated. Functionality could be added or removed dependant on the satellite's resources.

- **Customisation of Services:** Agents on board satellites could provide a flexible service for both server and client roles dependant on their current hardware and software status. This allows Agents to be reactive in their communications to other satellites.

- **Disconnected Operations:** Without an attitude determination and control system, the picosatellites could be tumbling and moving in differing orbits making intersatellite communication difficult.

Unreliable links with low-bandwidth and low-reliability require new methods for allowing flexibility in the system. Improving granularity in the client-server paradigm would allow a greater number of operations but would increase local resource use, complexity and reduce flexibility for change.

- **Improved Fault Tolerance:** Traditional client-server systems request data and services from remote environments which can then be executed locally. But if a return result is only partially complete, there are problems in addressing the state and possible recovery. As Agents encapsulate all state information in a single component, the information can be traced and recovered. Destructive space environment effects could be mitigated using Agents to detect hardware or even software errors.

The JADE-LEAP [16] software environment is adopted for mobile agent development and is used extensively to develop new Agent systems and novel applications for resource constrained devices using wireless links in Java. To accommodate the agent computing environment, a Java Virtual Machine (JVM) for embedded devices is typically required, which can serve as a communication medium between various heterogeneous platforms. However the Java programming language is not the best choice for real-time critical systems. To enable real-time functionality on board satellite systems the use of a Java specific processor called Java Optimized Processor (JOP) [17] is considered.

An FPGA-based SoC design is under development incorporating the JOP processor with the LEON3 SPARC V8 processor [18] as a secondary master processor core. The interface between LEON3 and JOP is achieved via the AMBA2-AHB bus as shown in Fig. 3. Benefits of utilizing JOP in an FPGA include moving the JVM software to hardware, reducing the memory footprint and enabling Java applications, such as Agents, for real-time applications.

Signal Routing

The data captured by sensors on board the slave satellites are processed and then they are sent to the ground station via the master satellite. Satellites performing a distributed satellite mission would also be required to carry out computationally intensive processing in addition to inter-node communication. Two network optimization objectives are considered: transmission power and time. The transmission power is required to guarantee reliable communications.

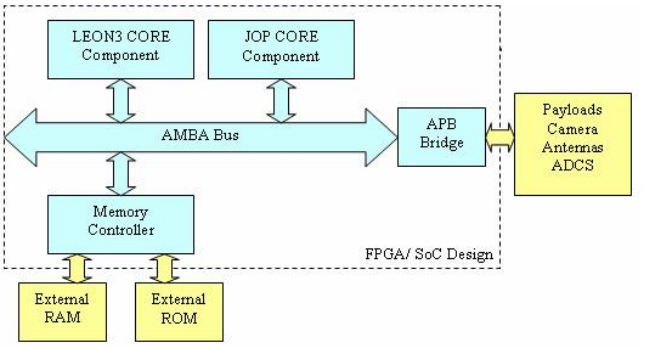


Figure 3. LEON3 and JOP integration system level

The well known Friis free space propagation model [19] defines the relationship between the received and the transmitted power. It concludes that by multi-hopping the sensed data will significantly reduce the power for communication. The transmission time for multi-hopping in a *n*-satellite network is given by the propagation delay and the satellite queuing and processing delay, which is determined by the amount of buffers, queuing discipline, servers and the packet arrival rate [8]. These two objectives are competing to each other. On the one hand more hopping satellites will save more power; on the other hand the end-to-end time delay will be increased.

MOEA is aimed at providing a set of Pareto-optimal routes, in which both objectives are optimized. The MOEA starts with a parent population, each individual containing a route from the starting node to the destination node, i.e. its chromosome. Each parent is randomly mated with one another to produce two children (crossover). The children are then mutated with certain probability. The objective functions of the children are calculated, and a fitness value is assigned to the parent and children. The individuals with best fitness are then passed onto the next generation. The process continues until the maximum number of generations is reached. In the end an optimized route is obtained. However nodes that are not working temporarily are stored in a vector. Any chromosome that contains such node will be considered dead and discarded during evolution.

4. SIMULATION RESULTS

Simulations were performed to determine how useful a task or service distribution would be on a network with varying latencies on all OSI layers - from the hardware layer in a LEON3 processor up to the application layer. As an example, the JPEG2000 image compression encoder was targeted for a 48 mega-pixel satellite image. Some experimental variables include:

- Worst-case hardware switching delay = 1.258 *ns* (Clock Report of LEON3 on XC3s-1500 FPGA)
- No. of nodes = 3

- MAC access delay = 2.049 *ms* [13]
- Service delay (variable) = 1 *ns* to 1 *s*
- Propagation through free space

$$D_{Propagation} = \frac{d}{c} = \frac{10{,}000}{2.99792458 \times 10^{8}} = 3.33 \times 10^{-5} s$$

- WiFi (IEEE 802.11b) Variables:

  No. of transmissions = 3 (also variable),

  Packet sizes = 1500 of 2346 bits, Channels = 14
- Image Size: 7507 x 6399 pixels, File size: 50.826 to 6.386 MB

The 'round trip delay' results in Fig.4 show that task distribution can reduce the time to complete a task, even over a distributed satellite network but there is also a limit on the number of nodes the task is distributed to before the intersatellite link latencies become too great.

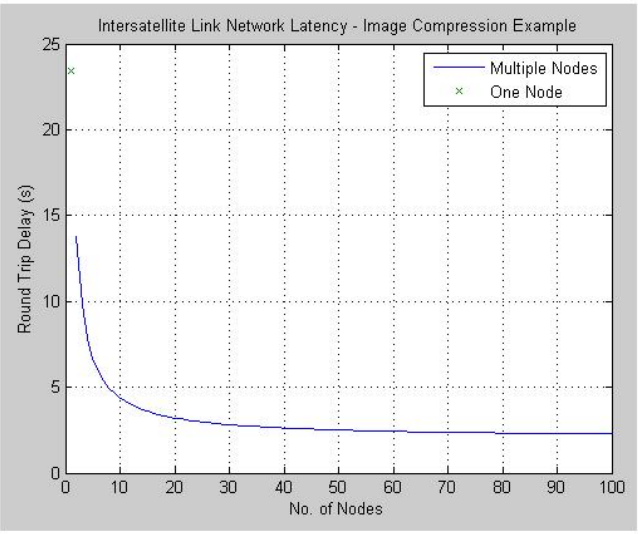


Figure 4. Round trip delay of a satellite network

JADE-LEAP was successfully employed to communicate over an ad-hoc IEEE 802.11 wireless link with a number of protocols, specifically TCP/IP and UDP, using multiple laptops. A much preferable scheme used for satellites is UDP, as described in [20] because it is better suited for 'store-and-forward' communications. UDP has a number of advantages: it is faster (a dropped packet, in this case, is preferred to a delayed packet); it can take advantage of JADE's Agent Communication Language (ACL) for acknowledgements and finally supports packet broadcast and multicasting.

As a fast demonstrator, JADE-LEAP was also used to distribute the JPEG2000 encoder service using the ACL scheme for agent registration and negotiation. Consider the scenario in a distributed satellite system where a satellite has an image to compress but does not have the service stored locally. It then beacons out to neighbouring satellites in the network to find if another satellite has that service. One satellite does and then wraps the service in a 'Mobile Agent', transfers itself across to the imaging satellite, executes that service and then returns to the original satellite it came from.

Wireshark [21], a network protocol analyser, is used to capture live network data throughputs in our network using the JADE-LEAP system and mobility of Agents. Wireshark is typically used to troubleshoot network problems, examine security problems and learn/debug protocol implementations, which makes it ideal for this analysis. We started with running the described scenario using JADE-LEAP's default protocol: TCP/IP. The throughput results are shown in Fig. 5 for varying different sizes of Agents from 12 KB down to 2.7 KB, where the black coloured graph denotes the TCP throughput and the red - the throughput of the Remote Method Invocation (RMI) mobile Agent.

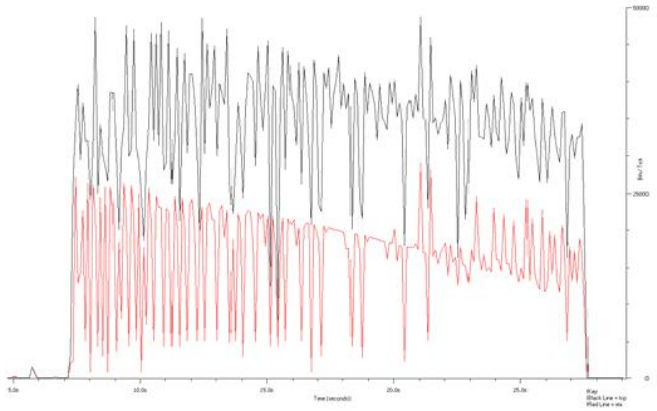
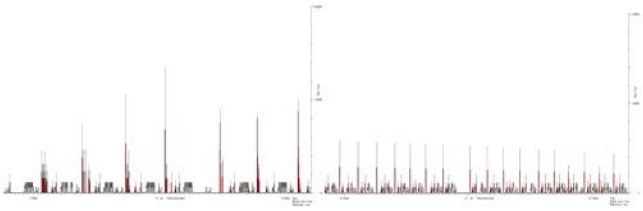


Figure 5. TCP throughput (bits/tick)

Agent payload size affects the throughput, where the smaller Agent has a lower data rate requirement: from 300.72 kbps with a 2.7 KB Agent to 409.6 kbps for a 12 KB Agent. The results in Fig. 5 also show that TCP utilizes much of the throughput for fault tolerance and reliability in its acknowledgement scheme which reduces the total payload data throughput over the network. This is confirmed by similar graphs in Fig. 6 where Agents of different sizes have different throughputs and completion times.



(a) (b)

Figure 6. TCP throughput analysis; a) Time 7.5 to 8.5s and b) 26.5 to 27.5s

At this point in time, there is no complete UDP transport service in JADE-LEAP and an internal communications peer (ICP) is required to plug-in to JADE-LEAP's interfaces.

5. PICOSATELLITE DESIGN

CubeSat Bus

The CubeSat platform is a 10x10x10 *cm* standard bus structure weighing at 1 *kg*, which is compatible with the PC/104 format. The CubeSat bus also comes in double (2U) and triple unit (3U) sizes to conform to the P-POD deployment mechanism. The CubeSat Kit platform has sold over 30 kits since the first launch of CubeSat in 2003 to researchers and companies in the small satellite field because it provides a standard COTS solution to develop new technologies. CubeSat advantages include a space ready structure, a reliable 'flight module' as the main on-board computer (OBC) based on the low power TI MSP430 microcontroller and many other CubeSat designs to learn from. Foreseeable problems include limited power budget for communicating to ground or operating payloads, with typical CubeSats having between 1 – 2 *W* of total satellite power.

The CubeSat development kit and the proposed picosatellite model are shown in Fig. 7. The Flight Module and the IEEE 802.11 PC/104 board are attached to the development board through the CubeSat kit bus.

Figure 7. CubeSat development kit and the proposed picosatellite

The OBC is to be a MSP430 microcontroller with an embedded reduced operating system, called Salvo. An MHX transceiver is available from Microhard Systems [22]. The transceiver can plug into the flight module directly. The CubeSat also accepts the 9XStream series of 910 *MHz* and 2.4 *GHz* transceivers from MaxStream which successfully flew on NASA's GeneSat-1 [23]. A CubeSat power subsystem commercially available from Clyde-Space includes Electrical Power System (EPS), battery and solar arrays. The 1U power board can provide 2.4 *W* (sunlit average) [5]. Three boards are in a process of development in house – a communications board to support wireless IEEE802.11 ISL, an FPGA board for distributed computing and a camera board.

The development of effective space networks requires the characteristics of a WSN reliability, robustness, flexibility and redundancy. Research into all current CubeSat missions shows that reliability and simplicity with the whole satellite is a key requirement to ensure success, whilst having more complex systems as separate payloads. To ensure reliability in space, we propose a satellite bus architecture that treats our new technology systems as payloads, as shown in Fig. 8.

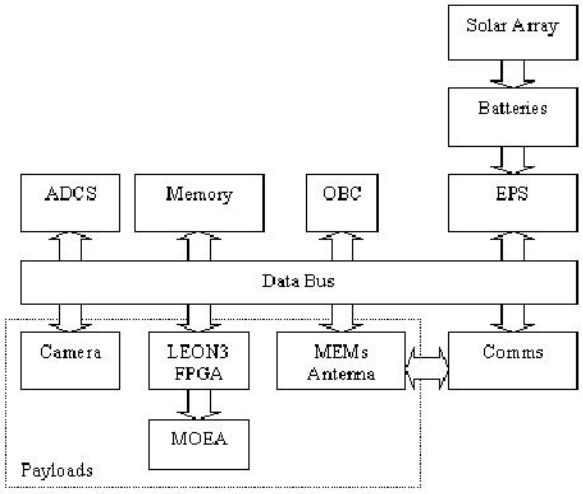


Figure 8. ESPACENET satellite architecture

Payload

The picosatellite nodes are to be computationally able to run MOEA on board to optimise the networks ability to perform the mission e.g. make decisions from complex algorithms on resource allocation. This will require additional hardware (CPU, memory, etc) and software (OS, applications) resources to be provided in the CubeSat constrained structure. Because this is a 'technology demonstration' mission, we design the payload including all the necessary components for distributed computing.

Fig. 9 shows the hardware architecture of the payload, consisting of three modules for satellite missions. The Global Positioning System (GPS) is used to determine the

satellite attitude and position in orbit. GPS can have many benefits for a distributed computing environment in space, especially with respect to cluster management, formation flying and collision avoidance. A space-enhanced GPS module can be provided by our spin-off company SSTL. Reconfigurable MEMS antenna is currently being built by our academic partner University of Edinburgh. The payload high-performance computing is based on a system-on-a-chip design, implementing the LEON3 processor and other associated peripherals. The peripheral IP cores include image compression, wireless MAC/PHY for inter-satellite link, image encryption. All these IP cores are currently under development at the Surrey Space Centre.

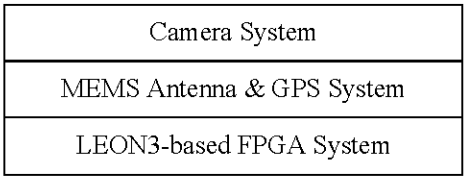


Figure 9. The payload architecture

The payload is controlled by the main on-board controller, which is the MSP430 microcontroller, via the I^2C bus. While the satellites are running in orbit, these modules can be turned on/off in according to the remaining power and mission requirements.

System-on-a-Chip Design

The payload computer and its peripherals are designed as a SoC in order to improve the hardware efficiency in terms of power, speed and area. Our payload computer includes a high-performance SPARC V8 processor LEON3 [18]. The SPARC V8 is a standard RISC architecture with typical features like large number of registers and few and simple instruction formats. However, the LEON3 IP core is more than a SPARC compatible CPU. It is also equipped with various peripherals that interconnect through two types of the AMBA bus (AHB and APB). Hence the SoC design is centric on the AMBA bus. All the IP cores can be added or removed from the bus.

The LEON3 processor supports many operating systems, including RTEMS, eCos, embedded Linux, and etc. For our project, RTEMS is chosen because it has a much smaller footprint; runs faster; and provides a multiprocessor manager [24]. All these features are essential for distributed computing within a picosatellite sensor network.

Image compression and multi-objective evolutionary optimization algorithms are computationally intensive. Our objective is to realise these algorithms as high-speed implementations. The algorithms can be run as software programs, but this approach is not sufficiently fast, which is an important requirement for real-time applications. The most efficient approach is to implement the computationally intensive algorithms as hardware accelerators. For example, we aim to implement the image compression algorithm as a hardware IP core and integrate as a peripheral to the LEON processor.

Another approach is to implement some functionality of the algorithms as an IP core, and to run the remaining functionality as software on the processor. We can achieve an optimal software-hardware split by profiling the algorithms. Here we take the MOEA as an example. The MOEA is initially programmed in C. It is running with the RTEMS real-time operating system on the LEON3 processor. The LEON3 is clocked at 40 MHz. There are lots of 'multiply' (MUL) and 'divide' (DIV) operations in the MOEA algorithm. We configure the LEON3 processor with three options: soft 'MUL/DIV' option, hard 'MUL/DIV' option (IP core), and floating point unit. The soft 'MUL/DIV' option can emulate 'multiply' and 'divide' operations with software. The hard 'MUL/DIV' option provides hard wired 'multiply' and 'divide' operations with the LEON3 processor. The floating point unit together with the 'MUL/DIV' IP core can realize the operations without any soft means. The C program of the MOEA algorithm is compiled using the RTEMS cross compiler. Table 2 shows the compilation results.

Table 2. Profiling results of MOEA for LEON3

Options	Time (mS)	'MUL/DIV' (mS)	Ratio (%)
Soft 'MUL/DIV'	1.54	0.159	10.32
Hard 'MUL/DIV'	0.96	0.038	3.96
Floating point unit	0.19	0	0

In Table 2, the second column is the execution time of MOEA. The third column is the time spent on the 'MUL/DIV' operation in the algorithm. It is obvious that the soft implementation of the 'MUL/DIV' is the slowest approach. When a hard 'MUL/DIV' is implemented, there are still 'MUL/DIV' operations after compilation. This is because the float point unit is not implemented, and all the fixed-point 'MUL/DIV' operations will be removed, but the floating 'MUL/DIV' operations will be running as software. This is confirmed by the case when the floating point unit is integrated, where no soft 'MUL/DIV' operations are needed any more. Also we find that the total speed of running the program is increased by integrating the hardware IP core. With the hard 'MUL/DIV', the execution time is 62% of the soft 'MUL/DIV' time. However when both the floating point unit and the hard 'MUL/DIV' are present, it is only 12%.

Partial Run-Time Reconfiguration (FPGA)

The proposed payload computer is implemented in an SRAM-based FPGA. FPGAs provide flexibility of design;

shorter time-to-market; lower cost; reconfigurability etc., which makes them suitable for use in small satellite on-board systems. Many SRAM-based FPGAs, such as Xilinx Virtex FPGAs, support partial run-time reconfiguration (RTR) [25]. In this case, the FPGAs can be reconfigured during run-time introducing changes to specific parts of the design only.

A disadvantage of SRAM-based devices is that they are vulnerable to the high levels of radiation in the space environment [26]. Heavy ions from cosmic rays can easily deposit enough charge in or near an SRAM cell to cause a single-bit error, or single event upset (SEU). Because SRAM FPGAs store their logic configuration in SRAM switches, they are susceptible to configuration upsets, meaning that the routing and functionality of the circuit can be corrupted. We take advantage of partial run-time reconfiguration to mitigate the SEU effect by repairing the areas affected by soft failures.

The FPGAs can accommodate not only soft IP cores, hard IP cores are also available. For example Xilinx FPGAs provide on-chip hard-wired cores like Block SelectRAM (BRAM) and multipliers. Hard processor IP cores (PowerPC) are also available in some Virtex II Pro and Virtex IV FPGAs. Starting from the Virtex II series, Xilinx Virtex FPGAs have integrated an internal configuration access port (ICAP) into the programmable fabric, which enables the user to write software programs that modify the circuit structure and functionality at run-time for an embedded processor. The ICAP is, in fact, a subset of the SelectMAP interface [27], which is used to configure Xilinx FPGAs.

Application-specific soft IP cores are being developed at the Surrey Space Centre to serve the requirements of the satellite mission, for example a Direct Memory Access Controller (DMAC), a MAC controller to support wireless inter-satellite links, a compression core and a fault-tolerant encryption core for image processing, etc.. Fig. 10 shows a diagram of the SoC architecture.

The on-chip peripheral bus (OPB) is used to connect all the ICAP modules. The ICAP is connected to the LEON3 processor via the OPB-to-AHB bridge. Once the FPGA is initially configured, the ICAP is used as an interface to reconfigure the FPGA. An ICAP device driver is included in the embedded Linux operating system [28], which needs to be adapted to RTEMS. The control logic for reading and writing data to the ICAP is implemented in the LEON3 processor as a software driver. The BRAM is used as a configuration cache. Because Virtex II FPGAs support reconfiguration only by frames, the BRAM must be big enough to hold one frame each time. The configuration bitstream of each SoC component can be stored on board in a Flash memory. The bitstream of a new or upgraded SoC component can be uploaded

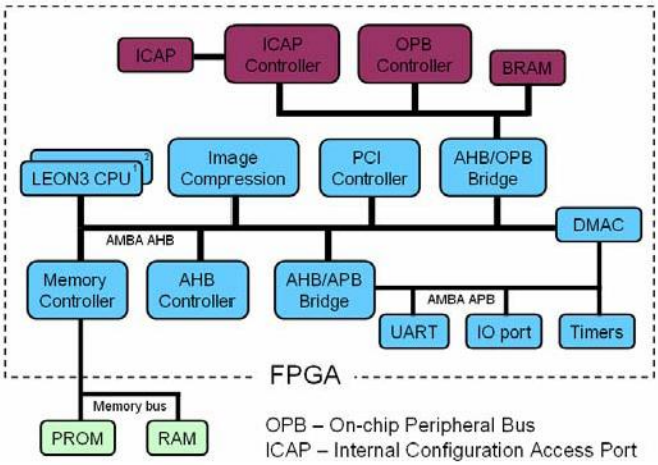


Figure 11. The Reconfigurable SoC architecture of the payload computer

6. CONCLUSIONS

In this paper we present a satellite sensor network for future satellite missions based on the CubeSat picosatellite platform. At network level a multi-objective optimization algorithm is used to optimise the signal routing. To perform satellite tasks, distributed computing over the satellite network is proposed. Design methodology for designing the payload hardware and software at node level is detailed. Image processing algorithms are implemented as peripheral cores of the LEON3 processor. Partial run-time reconfiguration is proposed to mitigate radiation effects in a system-on-a-chip design, which is implemented using SRAM-based FPGAs. A multi-agent software system is proposed to support inter-node communication based on IEEE 802.11 wireless connectivity.

It is envisaged that satellite sensor networks could be used to monitor the Earth with a global coverage. The high-performance FPGA device will enable fast on-board image analysis, so that processing results rather than raw data are sent to the ground station. Combined with the distributed computing capability this can provide a low cost approach to implementation of emergency response systems for detection and monitoring from space.

ACKNOWLEDGEMENTS

This research is sponsored by EPSRC under grant EP/C546318/01.

REFERENCES

[1] N.Haridas, E.Yang, A.T.Erdogan, T.Arslan, N.Barton, A.J.Walton, J.S.Thompson, A.Stoica, T.Vladimirova, X.Wu, K.D. McDonald-Maier, W.G.J. Howells. "ESPACENET: A Joint Project for Evolvable and Reconfigurable Sensor Networks with Application to Aerospace–Based Monitoring and Diagnostics", *Proceedings of 6th International Conference on Recent Advances in Soft Computing (RASC2006)*, Ed. K.Sirlantzis, pp. 410-415, 10-12 July 2006, Canterbury.

[2] T. Vladimirova, X. Wu, K. Sidibeh, D. Barnhart and A.-H. Jallad. "Enabling Technologies for Distributed Picosatellite Missions in LEO", Proceedings of *1st NASA/ESA Conference on Adaptive Hardware and Systems (AHS 2006)*, pp. 330-337, 15-18 June, Istanbul, Turkey, IEEE Computer Society.

[3] CubeSatKit, http://www.cubesatkit.com

[4] Pumpkin Inc, http://www.pumpkininc.com

[5] Clyde Space, http://www.clydespace.com

[6] SSTL, SSTL SGR-05 series space GPS receivers, http://www.sstl.co.uk/documents/Subsys_SGR05_HQ%5B1%5D.pdf

[7] ANSI/IEEE 802.11 Standard, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 1999.

[8] X. Wu, T. Vladimirova and K.Sidibeh. "Signal Routing in a Satellite Sensor Network Using Optimisation Algorithms", to appear in *Proceedings of IEEE Aerospace Conference 2008*, Big Sky, USA.

[9] T. Vladimirova, X. Wu, A.-H. Jallad, and C. P. Bridges, "Distributed Computing in Reconfigurable Picosatellite Networks", *Proceedings of NASA/ESA Conference on Adaptive Hardware and Systems (AHS 2007)*, Edinburgh, August 2007.

[10] CubeSat Community, http://cubesat.calpoly.edu/

[11] N. Haridas, A. El-Rayis, A. T. Erdogan, and T. Arslan, "Multi-frequency antenna design for space based reconfigurable satellite sensor node", *Proceedings of NASA/ESA Conference on Adaptive Hardware and Systems (AHS 2007)*, Edinburgh, August 2007.

[12] K. Sidibeh, T. Vladimirova, A. Baker. "Adapting the IEEE 802.11 Protocol for Inter-Satellite Links in LEO Constellations", *Proceedings of Wireless for Space Applications Workshop*, 10-13 July 2006, ESTEC/ESA, Noordwijk, The Netherlands.

[13] T. Vladimirova, C. P. Bridges, K. Sidibeh, D. Barnhart, G. Prassinos, A.H. Jallad, X. Wu, J.Paul, A. Baker, K. Maynard, "ESA Wireless Study, Technical Note 1", Surrey Space Centre, University of Surrey, UK.

[14] A.-H. Jallad and T. Vladimirova. "Distributed Computing for Formation Flying Missions" *Proceedings of ACS/IEEE International Conference on Computer Systems and Applications, AICCSA-06*, 8-11 March 2006, Dubai/Shariah, UAE.

[15] C. P. Bridges and T. Vladimirova. "Autonomous Software Agents in Wireless Embedded Systems", *3rd UK Embedded Forum (UKEF'07)*, University of Durham, April 2007.

[16] JADE Whitepaper, http://jade.tilab.com/papers/2003/WhitePaperJADEEXP.pdf

[17] M. Schoeberl, "A Java Processor Architecture for Embedded Real-Time Systems", *Journal of Systems Architecture (2007)*, 21st June 2007.

[18] J.Gaisler. GRLIB IP Library User's Manual (Version 1.0.4). Gaisler Research, 2005.

[19] H.T. Friis, A note on a simple transmission formula. Proc. IRE, 34, 1946.

[20] L. Wood, C. Peoples et al. "TCP's protocol radius: the distance where timers prevent communication," *Proceedings of 3rd International Workshop on Satellite and Space Communications (IWSSC '07)*, September 2007.

[21] Wireshark Website, http://www.wireshark.org

[22] Microhard Systems, http://microhardcorp.com

[23] Ignacio A. Mas, Christopher A. Kitts. "A Flight-Proven 2.4GHz ISM Band COTS Communications System for Small Satellites", *Proceedings of 21st Annual AIAA/USU Conference on Small Satellites 2007*, Utah, USA.

[24] J. Gaisler, RCC User's Manual. Gaisler Research, 2006.

[25] Xilinx, Two Flows for Partial Reconfiguration: Module Based or Difference Based. Application Note. http://www.xilinx.com/bvdocs/appnotes/xapp290.pdf.

[26] M. Ohlsson, P. Dyreklev, K. Johansson, and P. Alfke. "Neutron Single Event Upsets in SRAM-based FPGAs", *Proceedings of IEEE Nuclear and Space Radiation Effects Conference (NSREC'1998)*, 1998, Newport Beach, California, USA.

[27] B.Blodget, P.James-Roxby, E.Keller, S.McMillan, and P.Sundararajan. "A Self-reconfiguration Platform", *Proceeding of 13th International Conference on Field-Programmable Logic and Applications, FPL'2003*, pp. 565-574. 2003, Lisbon, Portugal.

[28] J.A.Williams, and N.W.Bergmann. "Embedded Linux as a Platform for Dynamically Self-Reconfiguring Systems-On-Chip", *Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA 2004)*, 2004, Las Vegas, Nevada, USA.

BIOGRAPHY

Tanya Vladimirova, MEng, MSc, PhD, CEng, MIET, MIEEE, is a Senior Lecturer at the Department of Electronic Engineering, University of Surrey, UK. She is a member of the Surrey Space Centre, where she leads the VLSI Design and Embedded Systems research group. Her research interests are in the areas of FPGA/ASIC design, wireless sensor networks, image processing for embedded systems, distributed computing and artificial intelligence.

Xiaofeng Wu, PhD, MIET, is a member of the VLSI Design and Embedded Systems group at the Surrey Space Centre, University of Surrey. He is currently working on the ESPACENET project, which is funded by the research council EPSRC.

Christopher P. Bridges, BEng, Student-MIET, is a 2nd year PhD Student in the VLSI Design and Embedded Systems group at Surrey Space Centre, University of Surrey. His research is in distributed computing using software agents for satellite systems and multi-core design for FPGAs.