

Software/Hardware Reconfigurable Network Processor for Space Networks

Clement G. Lee
Jet Propulsion Laboratory
California Institute of Technology
818-354-5587
clement@arcadia.jpl.nasa.gov

Andrew A. Gray, Jeffrey M. Srinivasan, Yong J. Chong,
Allen H. Farrington, Kenneth J. Peters, Valerie M. Stanton

Abstract – We present an overview of long-life reconfigurable processor technologies and of a specific architecture for implementing a *software reconfigurable (software-defined) network processor* for space applications. A prototype of the software defined reconfigurable processor described here is currently operating in the laboratory at the Jet Propulsion Laboratory. The reconfigurable processor performs the functions of the physical layer (software radio), namely modulation, demodulation, pulse-shaping, error correction coding and decoding, as well as the data link layer, network layer, transport layer, and application layer science processing. The primary motivations behind the space-based software reconfigurable network processor are the following:

- To enable rapid-prototyping and rapid space-qualified implementations of communications, navigation, and science signal processing functions.
- Providing long-life communications infrastructure enabled by on-orbit processor reconfiguration.
- Providing greatly improved science instrumentation and processing capabilities through on-orbit *science-driven reconfiguration*.

I. Introduction

This work extends numerous advances in commercial industry as well as military software radio developments [1-5] to space-based radios and network processing. Such radios are software-defined while the implementation of the radio and other network functions are generally performed in combinations of the following software-defined processors: generic software processors, field programmable gate arrays (FPGAs), digital signal processors (DSPs), as well as tradition digital and mixed-signal applications specific integrated circuits (ASICs) and discrete analog-circuits. The development of such radios and the network processor presented here, require defining the correct combination of the processing methods outlined above.

We first introduce the software reconfigurable paradigm as appropriate to address the challenges of developing long life space based communications infrastructure. Then we discuss a specific reconfigurable architecture developed to achieve the benefits of the reconfigurable paradigm. To make use of the processor we provide an overview of the appropriate design methodologies that should be used; object-oriented design of hardware and software algorithms is a key aspect of these design methods. Such methods will empower the design engineer with tools required for reconfiguration, rapid prototyping and implementation of a wide variety of signal processing functions.

In particular dynamic partial reconfiguration implies that an object oriented design method should be followed. Dynamic partial reconfiguration of the network processor will greatly increase the value of the processor within the network. For example the processor's ability to perform a very large number of temporally separated (time-multiplexed) functions will dramatically increase network elasticity and compatibility (see Fig. 1).

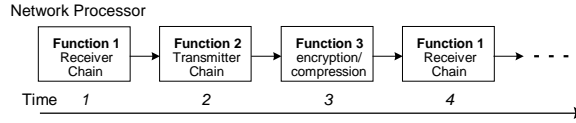


Figure 1. Time multiplexed processor functions

Finally we give examples of systems in space that will make use of the benefits of the processor and discuss the performance of the prototype currently operating in the laboratory at the Jet Propulsion Laboratory, and highlight future areas of development.

II. Software Reconfigurable Processor Technologies

A. Software Reconfigurable Paradigm

Many of the motivations for a space-based reconfigurable processor are similar to those driving reconfigurable processor efforts in private industry, in particular the cell-phone industry. Like cellular phones, space-based processors need long life and during this lifetime, diverse applications arise. These potential applications cannot be anticipated at product/mission launch. Moreover, the value of adapting to these unpredictable needs is extremely high, driving the need for reconfigurability [1].

The reconfigurable processor architecture, a composite of the processors listed previously, is determined by making trades between complexity, cost, development time, mass and size, flexibility, power consumption, and reliability to achieve system requirements. Given the variety of processors available in the commercial sector and the varying development platforms, these trades are extraordinarily complex.

We present high-level design paradigms and a generic reconfigurable processor architecture for providing tremendous flexibility, which in this instance leads to long life, concurrent mission reconfigurability, and rapid prototyping of a wide variety of signal processing functions.

The high-level description of the functions of the space-based reconfigurable network processor is that it serves as communications infrastructure, science instrument and science data processor, and navigation infrastructure. Figure 2 is a conceptual illustration of these functions and how they are implemented. The network processor is represented as conventional network layers (layers 1–4, and 7 of the OSI network model).

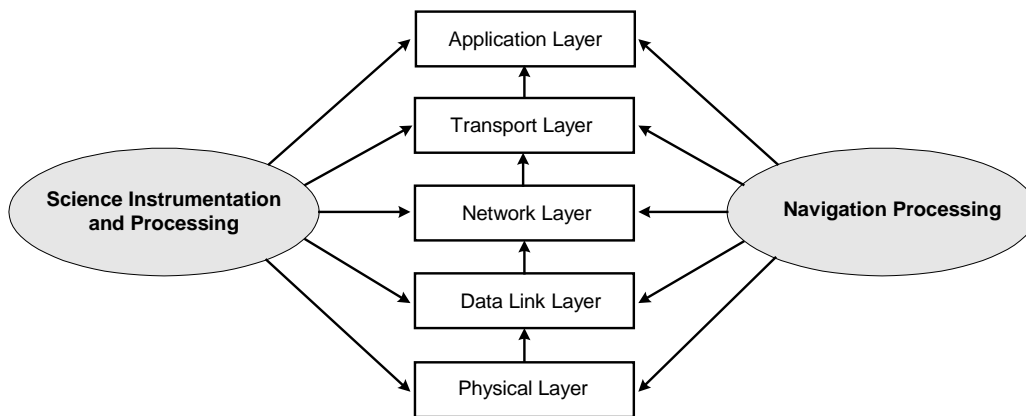


Figure 2. Model of Science, Navigation, and Communications Processing

Figure 3 illustrates a possible paradigm for communications, navigation, and science requirements developed by scientist and mission planners, to be integrated into the network processor during an operational mission. With reconfigurable processor technology, network communications and navigation may be improved during a mission. Of great importance from a science perspective is the fact that science processing may be modified by mission planners as a result of the science data acquired during a mission.

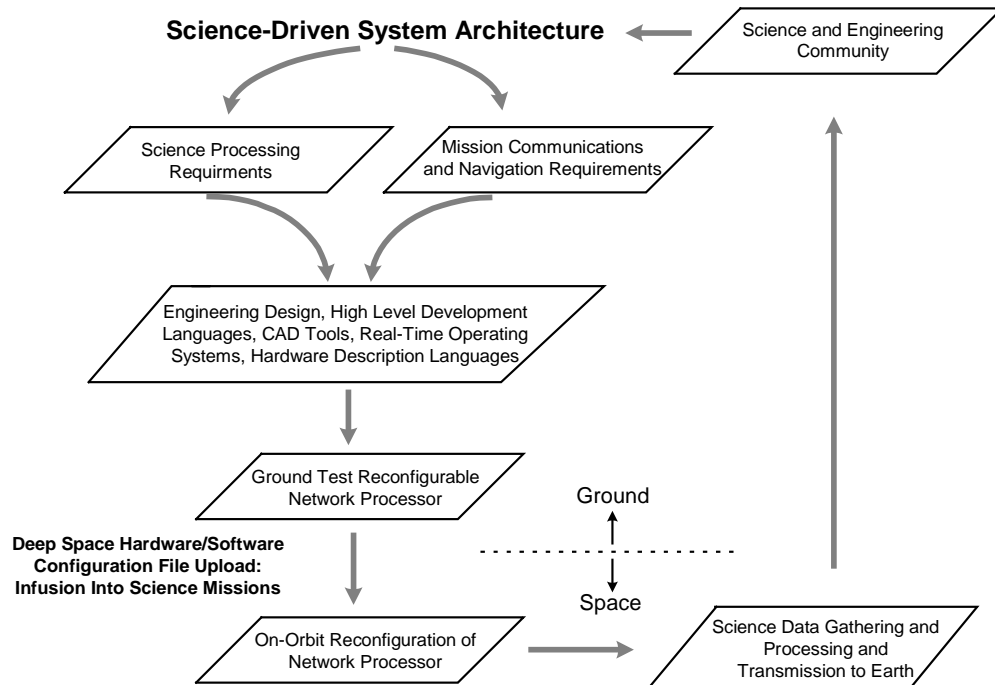


Figure 3. On-Orbit Reconfiguration of Network Processor

B. Reconfiguration Examples

Example: Science Instrument Processing

The argument for the need for on-orbit science processing enhancements is very compelling. The science goals and therefore the exact processing desired may change after mission launch. An example of this was the desire to change the way occultation measurements were made in the *CHAMP* mission after launch. *Mars Scout* atmospheric occultation measurements may also reveal that changes should be made in the way these measurements are made.

The concept of the reconfigurable processor as a science-defined processor may be extended to include time-varying science processing. Due to the reconfigurable nature of the network processor it may fill the role of a large number of temporally separated science instrument processors.

Example: Physical Layer Software Radio

The deep space communications channel has very unique problems compared to terrestrial communications. Primarily, the channel distances involved are often many orders of magnitude larger than those of terrestrial communications. This makes power efficient transmission of information critical. Power efficiency may be increased through the use of error-control coding. Turbo-codes and low-density parity check codes currently offer performance approaching the Shannon limit on channel capacity [6]. The advantages these modern error-

control codes exhibit over more conventional codes are well known and are on the order of many decibels in required transmit power savings. Unfortunately the optimal decoders for many such codes are computationally intensive, making them prohibitively complicated to implement for high data rates. There is little doubt in the academic community that decoders for these codes will be developed with significantly less complex implementations. In other words, although it would be a significant undertaking to implement a turbo decoder for data rates in excess of a few megabits per second in today's space qualified processing technology, in a few years it is probable that simplified decoders will be developed to be readily implemented in this processing technology. Implementing such decoders in a processor years after launch is possible with the paradigm illustrated in Figure 3.

C. Reconfigurable Architecture

In deciding on a reconfigurable architecture, several classes of processors were considered: ASIC, FPGA, DSP, and microprocessors. No single processor class is superior to another for all applications; the application will dictate the selection of the processor type. Figure 4 is an abstract depiction of the advantages and disadvantages of each processor class [8,9].

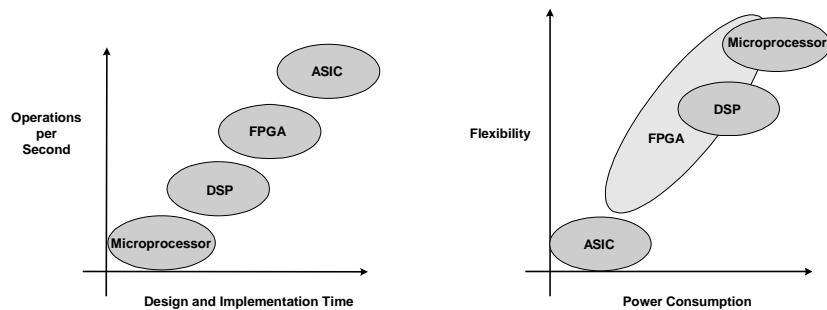


Figure 4. Properties of different processors

The primary applications here are long life science driven satellite and space exploration networks. Thus, flexibility and computational power motivated the decision to use a reduced instruction set computer (RISC) microprocessor and an FPGA. Figure 5 illustrates a conceptual picture of the network layers implemented across a reconfigurable processor. Note that the configuration of the software and hardware processors is defined by software control. This hybrid combination allows for complex floating/fixed point processing from the RISC as well as high-speed parallel and/or pipeline processing from the FPGA. And both the RISC microprocessor and the FPGA are reprogrammable.

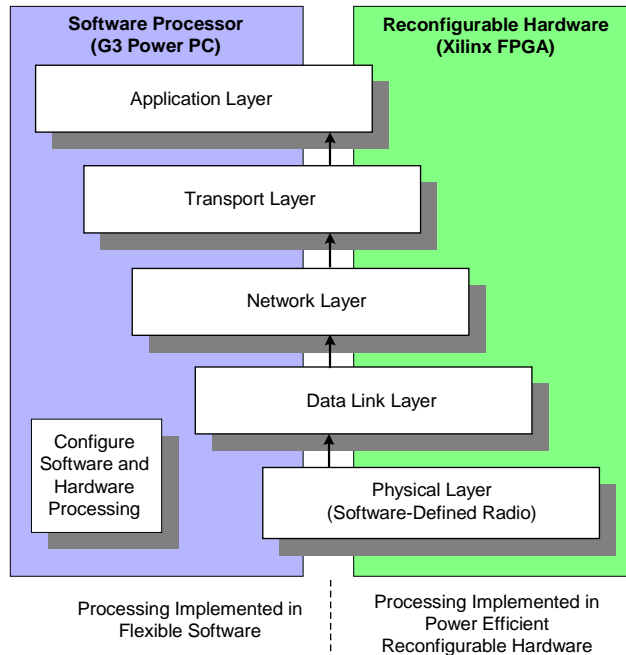


Figure 5. Network Layers Implemented in a Reconfigurable Processor

The hardware platform for the reconfigurable network processor consists of a Xilinx Virtex XCV1000 field programmable gated array (FPGA) on an Alpha Data ADM-XRC daughter card and an MCP750 board. The Xilinx Virtex part was chosen for: ability to partially reconfigure the FPGA, tools support, immunity to latch ups, availability of radiation tolerant parts, and usage in another mission, the Mars Exploration Rover (MER). The MCP750 board has a Motorola 750 MHz PowerPC processor. The PowerPC was chosen for its high MIPS/Watt ratio and availability of a radiation tolerant chip. The MPC750 board was chosen for: driver support, the board support package, RTOS software support, etc. The Alpha Data ADM-XRC card is the hardware interface between the MCP750 card and the Xilinx Virtex FPGA. The Alpha Data card also has a user input/output interface for debugging. The ADM-XRC's PCI Mezzanine connector (PMC) provides ease of use with industry standard connections.

With a microprocessor in the architecture, a decision must be made on the use and type of an operating system (OS). Since this network processor will be responsible for dealing with communications, navigation, and science processing simultaneously, an OS is necessary for multi-tasking. Integrity is the real time operating system (RTOS) chosen for this platform. Integrity was developed by Green Hills Software Corp. for aerospace customers. Integrity was chosen for many reasons. Integrity's hardware memory protection provides secure tasks, secure device drivers, and secure inter-process communications. Integrity has pre-emptive multi-tasking, making it a true real-time scheduler. Integrity uses virtual memory and has field upgrade and debugging capabilities. Green Hills provides excellent vendor support as well as complete front-end and back-end tools.

D. Reconfigurable Design Methodology

Hybrid software and reconfigurable hardware design calls for a unified object-oriented and hierarchical design paradigm encompassing:

1. *Software*: object oriented design in C++ for a microprocessor.
2. *Reconfigurable hardware or "firmware"*: object oriented hardware description language (HDL) code, e.g. Verilog

3. *Interface*: object oriented software interface drivers and/or HDL decode/encode interface modules.

The network processor architecture was developed to allow the algorithms and signal processing of various layers to be accomplished in a logical way: with some functions logically being implemented in software and some logically in reconfigurable hardware. There has been little work on broadly applicable design methods, which allow the design engineer to define such objects in the context of a reconfigurable hardware/software processor. A primary problem is that although both hardware and software development are generally understood as logically separate entities or classes of objects, there are relatively few techniques that apply to the design of systems as a whole, and particularly for the instance when partial reconfiguration with time constraints is desired. The goal is to develop design tools that facilitate this logical placement of processing.

The interface class is really a combination of software and hardware processing (classes). Rather than include an entire signal processing module however, we minimize the logical extent of the module. This is because interface objects exist primarily to solve multirate data input/output timing issues and data format conversions that exist on the boundary between hardware and software. An example object could be a data buffer (either in hardware or software) between hardware output software input.

Coad and Yourdon [11-13] present a set of quality design principles based on the following parameters, that result in better, “maintainable” object oriented software designs:

- *Coupling*: Interaction coupling between classes should be kept low by reducing the complexity of message connection and decreasing the number of messages that can be sent and received by an individual object. Inheritance coupling between classes should be high.
- *Cohesion*: A service in a class should carry out one and only one function. The attributes and services should be highly cohesive. A specialization should actually portray a sensible specialization.
- *Clarity of design*: A consistent vocabulary should be used. The names in the model should closely correspond to the names of the concepts being modeled. The responsibilities of a class should be clearly defined and adhered to. The responsibilities of any class should be limited in scope.
- *Generalization-Specialization depth*: It is important not to create specialization classes which are conceptually not a real specialization, e.g. created for the sake of reuse.
- *Keeping objects and classes simple*: Excessive numbers of attributes in a class should be avoided – an average of one or two attributes for each service in a class is usually all that is required. “Fuzzy” class definitions should be avoided. A class should map to a type of entity in a problem description. All definitions should be clear, concise, and comprehensive.

Maintainable means that the code is easily understandable and modifiable. These design principles will lead to fine granularity libraries that can be easily reused. Dynamic on-board linking of software code is analogous to partial reconfiguration of an FPGA. In both cases, one or many modules of software or firmware can be uploaded to change the functionality of the network processor. Software does not need to be recompiled. Object oriented and hierarchical design is essential to accomplish partial reconfiguration or reprogramming of the network processor.

Signal Processing Worksystem (SPW) was chosen as the tool for high level signal processing design, development, and simulation for several major reasons. SPW allows for modular/object oriented and hierarchical designs. SPW is a commercial off the shelf (COTS) product as the tool contains pre-generated signal processing code or modules. By incorporating these modules into the design, time can be saved. Most importantly, SPW was chosen because it could be used to

generate Verilog, the hardware description language (HDL). This allowed us to allocate fewer resources on hardware development.

FPGA design is done at high level using SPW. SPW generates object oriented HDL. Then Xilinx tools convert the HDL into an FPGA configuration file or bit file. Xilinx provides many tools to simulate the HDL designs as well as tools to analyze timing issues.

Some software designs were implemented in SPW for simulation purposes. Other software designs are implemented in C++, using Green Hill's tools, Mutli builder. This tool is a compiler, debugger, and an editor. Green Hills also provides an RTOS simulator as well as an event analyzer to aid in embedded development.

III. Hardware Prototype of a Reconfigurable Network Processor

A prototype, we call the NavaTyrr, of the reconfigurable processor developed using the preceding architecture and design methodologies is currently operating in the laboratory. To date the physical, data link, and transport layers have been implemented and demonstrated. The physical layer consists of a reconfigurable binary phase shift keying (BPSK) demodulator, the data link layer consists of an implementation of the majority of the Proximity-1 protocol draft recommendation by the Consultative Committee for Space Data Systems (CCSDS), and the transport layer consists of a commercial transport layer protocol. The processor has been demonstrated in a two-way communications link with data rates as high as 1 megabit per second per channel.

A. Physical Layer BPSK Radio

The physical layer of our prototype includes a binary phase shift keying (BPSK) radio. The BPSK radio was designed with many programmable parameters, in addition to being implemented on a reconfigurable platform. This BPSK radio can be programmed to transmit and receive data rates between 1 Kbps and 4Mbps, using non-return to zero (NRZ) or Manchester (Bi-Phase L) pulse wave form. The carrier frequency of the transceiver is also programmable up to 4MHz. The current receiver consists of a carrier tracking loop (Costas loop) and a symbol tracking loop (Data Transition Tracking Loop (DTTL)). Both tracking loops have 2nd order loop filters, with programmable normalized loop bandwidths ranging from 0.1 to 0.0001 (400K to 0.1 Hz). A lock detector measures the power in the inphase and the quadrature arms of the Costas loop to determine if the carrier tracking loop is locked.

With both software and hardware available, we have the flexibility to implement more complex and computation intensive algorithms. Thus, we decided to implement an open loop software FFT acquisition algorithm. A digital automatic gain control (AGC) supplements a broad band analog AGC. The analog AGC will scale the noisy signal down to fit within the range of the analog-to-digital converter (ADC). The digital AGC must scale the signal back up after the wide band noise has been filtered out. Error correction coding and decoding schemes can be implemented in either hardware or software. This offers a much wider range of coding options. The NavaTyrr uses a convolutional code that is required for the space data link protocol in use.

In simulation, the physical layer performance of the BPSK receiver was measured both by the loop SNR of each tracking loop and the BER. The performance metric of the hardware receiver is the bit error rate (BER), as loop SNR is difficult to measure accurately. Figure 6 shows the preliminary performance of the physical layer.

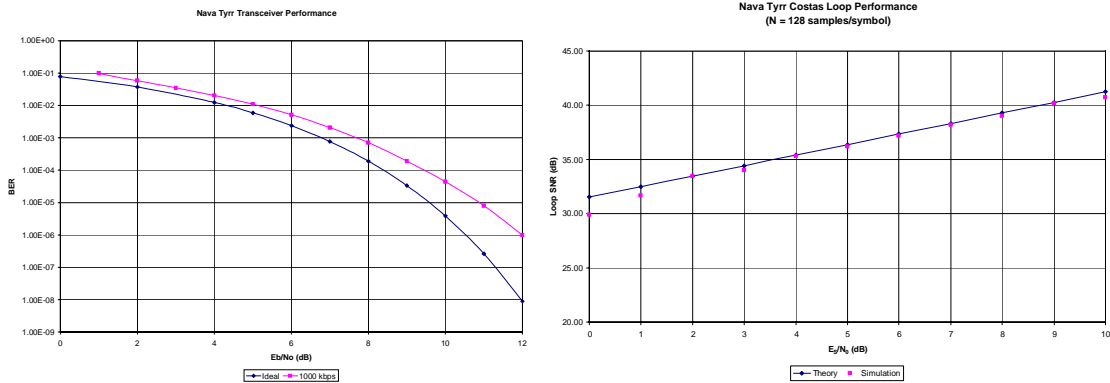


Figure 6. Physical layer performance

B. Data Link Layer and Network Layer: Proximity-1 Protocol

The Proximity-1 space link protocol, developed by the Consultative Committee for Space Data Systems (CCSDS) [16], specifies the Physical Layer, the Medium Access control sublayer, and is the Data Link Layer protocol intended for space communications around Mars. Proximity-1 primarily emphasizes point to point connection, while there are plans to extend Proximity-1 to one-to-many connections. Proximity-1 specifies both reliable and non-reliable modes. Proximity-1 requires a full duplex link. The NavaTyrr prototype currently contains the most complete implementation of a Proximity-1 link. Our Proximity-1 is implemented in both software and hardware, with the majority of its functionality implemented in the microprocessor. As part of the Data Link Layer, Proximity-1 acts as software controller of the hardware, turning on/off parts of the transceiver when necessary, changing data rates, managing the data, etc. Hardware buffers are used to send and receive data to and from software for Proximity-1 to manage. The buffers are necessary to allow the much slower software interrupt to keep up with the high data rates. An Attached Synchronization Marker (ASM) is appended to the beginning of each frame to determine frame synchronization. The ASM and the buffers are implemented in hardware.

The Ethernet bridge is the next network layer of the OSI model. The Ethernet bridge is implemented entirely in software. The Ethernet bridge acts as a driver to interface data packetized under Proximity-1 protocol to data packetized for TCP/IP protocol. The application layer of the prototype is Apple's Quicktime software. Fortunately for us, the other layers of the OSI model have already been implemented by commercial plug and play software, allowing TCP/IP to transfer information from a Quicktime server to a Quicktime client.

C. Network Performance of the Prototype

A lot of knowledge was gained from the preliminary performance results for the NavaTyrr. The initial design choices were motivated by a tight schedule. The implementation of the buffers, for example takes advantage of the Xilinx's circulating dual port RAM. Therefore, the buffers were very easy to implement. The buffers were sized at 4 kilobytes for the highest data rate and the software interrupt rate. Since any data must go through the entire half buffer, the network latency of the NavaTyrr is quite high. Software reads/writes to the buffers after one half has been read/written. A future implementation of this will involve a FIFO buffer scheme that will allow data to be written to the front of the buffers based on priority. The effective data rate across our reconfigurable network processor is approximately one half of the actual data rate. This includes the ethernet bridge and the Proximity-1 overhead. The overhead from Proximity-1

alone gave us 90% of the true data rate. Optimal software should output an effective data rate much closer to the actual data rate.

IV. Future Technology Developments

Future NASA science missions plan to use the reconfigurable network processor developed at JPL. These missions include Space Technology 5 (ST-5) in 2003 (as part of the *Constellation Communication and Navigation Transceiver, CCNT*); the *StarLight* instrument for Autonomous Formation Flyer (AFF) in 2006; and the *Neige* experiments on Mars premier orbiter in 2005. The reconfigurable processor may also be appropriate for Mars scout missions in 2007 as well as future Mars network payloads.

Other technology developments that are ongoing and will be part of the flight development efforts of the network processor include:

- 1) Improved partial dynamic reconfiguration.
- 2) Improved scrubbing techniques.
- 3) Multiple simultaneous data channels.
- 4) The use of bandwidth efficient modulations.
- 5) Simultaneous navigation and communications using both spread spectrum and pilot channel techniques.

V. Conclusion

We have provided a brief overview of software reconfigurable processor technologies and the paradigms used in development of the prototype software reconfigurable network processor operating in the laboratory at JPL. The authors believe reconfigurable processor technology leads directly to a tremendous increase in the diversity of applications of signal processing for science benefits as compared to traditional processor technologies, and can provide long-life infrastructure support. Mission concurrent reconfiguration enables multi-mission support, reconfigurable communications and navigation infrastructure, and science instrumentation and processing improvements. Examples of missions planning to use the reconfigurable architecture of Figure 5 and variations developed at JPL include Space Technology 5 (ST-5) in '03, the Starlight instrument for Autonomous Formation Flyer in '06, and the Neige experiments on Mars Premier orbiter in '05. The reconfigurable processor may also be appropriate for Mars Scout missions in '07 as well as future Mars Network payloads.

References

- [1] N.J. Drew, M.M. Dillinger, "Evolution toward reconfigurable user equipment," *IEEE Communications Magazine*, vol. 39, no. 2, Feb. 2001.
- [2] E. Buracchini, "The software radio concept," *IEEE Communications Magazine*, vol. 38, no. 9, Sept. 2000.
- [3] W.H.W. Tuttlebee, "Software-defined radio: Facets of a developing technology," *IEEE Personal Communications*, vol. 6, no. 2, pp. 38–44, April 1999.
- [4] J. Mitola III, "Software radio architecture: A mathematical perspective," *Selected Areas in Communications, IEEE Journal on*, vol. 17, no. 4, pp. 514–538, April 1999.
- [5] M.S. Cummings, S. Haruyama, "FPGA in the software radio," *IEEE Communications Magazine*, vol. 37 no. 2, pp. 108–112, Feb. 1999.
- [6] M. K. Simon, S. Hinedi, W. Lindsey, *Digital Communication Techniques*, PTR Prentice Hall, Englewood Cliffs, New Jersey, 1995
- [6] A. Stoica, R. Zebulum, D. Keymeulen, R. Tawel, T. Daud, and A. Thakoor, "Reconfigurable VLSI architectures for evolvable hardware: From experimental field programmable transistor

- arrays to evolution-oriented chips," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 9, no. 1, pp. 227–232, Feb. 2001.
- [7] E. Ramsden, "The ispPAC family of reconfigurable analog circuits," *Proc. of The Third NASA/DoD Workshop on Evolvable Hardware*, pp. 176–181, 2001.
- [8] P.P. Gelsinger, "Microprocessors for the new millennium: Challenges, opportunities, and new frontiers," *Proc. IEEE Intl. Solid-State Circuits Conf.*, pp. 22–25, 2001.
- [9] J. Eyre and J. Bier, "The evolution of DSP processors," *IEEE Signal Processing Magazine*, vol. 17, no. 2, pp. 43–51, March 2000.
- [10] O. Mencer, M. Platzner, M. Morf, and M. Flynn, "Object-oriented domain specific compilers for programming FPGAs," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 9, no. 1, February 2001.
- [11] P. Coad and E. Yourdon, *Object-Oriented Analysis*, Prentice-Hall, 1991.
- [12] P. Coad and E. Yourdon, *Object-Oriented Design*, Prentice-Hall, 1991.
- [13] L.C. Briand, C. Bunse, and J.W. Daly, "A controlled experiment for evaluating quality guidelines on the maintainability of object-oriented designs," *IEEE Trans. on Software Engineering*, vol. 27, no. 6, June 2001.
- [14] W. Nebel and G. Schumacher, "Object-oriented hardware modeling - where to apply and what are the object?," *Proc. EuroDAC'96*, pp. 428-433, Geneva, Switzerland, 1996.
- [15] P.N. Green, M.D. Edwards, "Object oriented development for reconfigurable embedded systems," *IEE Proceedings on Computer and Digital Technology*, vol. 147, no. 3, May 2000.
- [16] CCSDS 211.0-R-2: *Proximity-1 Space Link Protocol*, Red Book, Issue 2, January 2000.
<http://ccsds.org/documents/pdf/CCSDS-211.0-R-2.pdf>