Runtime Hardware Reconfiguration in Wireless Sensor Networks

Joel L. Wilder, Vladimir Uzelac, Aleksandar Milenković, and Emil Jovanov

Abstract— **Recent advances in sensors, low-power system-on-a-chip devices, and wireless communications, have prompted a proliferation of wireless sensor networks. As these networks require advanced integration, intensive onboard processing, and low power consumption, field programmable gate arrays (FPGAs) emerge as a technology that strikes an optimal balance between processing power, energy requirements, and flexibility. Through the power of reconfigurability, wireless sensor network designs containing reprogrammable logic can be upgraded, errors can be fixed, and limited-resource applications can be dynamically reprogrammed in the field. While powerful, this reconfiguration process can be costly in terms of labor and downtime. In order to address these issues we propose a REWISE framework (Reconfigurable Wireless Intelligent Sensor Networks) for real-time reconfiguration of the programmable logic on sensor nodes through the wireless sensor network communication infrastructure.**

I. Introduction

Recent technological advances in integrated circuits, wireless communication, and sensors have enabled a new generation of wireless sensor networks that can be used in a number of military and civil applications. Typically, a large number of miniature and inexpensive sensor nodes is deployed to monitor and control environments without human intervention for a long period of time [1], [2]. A wireless sensor network usually consists of a number of wireless sensor nodes and a base station. Various communication models can be employed, such as direct, multi-hop, or clustering.

Each sensor node typically incorporates the following: (a) one or more physical sensors capable of measuring the state of the environment; (b) actuators that affect the state of the environment; (c) a microprocessor that provides onboard processing of “raw” data from physical sensors, facilitates communication, and handles control messaging; (d) a radio interface to communicate information with neighboring nodes or a base station and eventually to the external world; (e) a power source, typically a battery. Since the nodes in a wireless sensor network must operate under severe size, weight, and power consumption constraints, important hardware and software design decisions must be made in order to meet these requirements while fulfilling system goals. One possible option for meeting these objectives is to use an application-specific integrated circuit (ASIC). ASICs are specifically designed for a target application, and can achieve the best performance, but their lack of flexibility to accommodate design changes and long design-to-fabrication cycles usually prohibit their use in a sensor node. Another option is a general-purpose processor. This type of processor provides the flexibility that the ASIC cannot offer, but at the price of reduced processing speed and power efficiency. A good compromise between hardware inflexibility and software inefficiency can be found in low-power programmable logic [3]. Reprogrammable logic can be utilized to accelerate critical paths and reduce power consumption for a wide range of signal processing algorithms and communication functions required by sensor platforms [4]. Reconfigurable sensor platforms offer flexibility and cost-effective customization before deployment and provide for the possibility of run-time reconfiguration. Furthermore, constructing sensor platforms with programmable logic and a customizable front-end may lower costs, as multiple target applications are able to share a common wireless platform.

In this paper, we introduce a REWISE framework that allows dynamic reconfiguration of sensor nodes based on a variety of situations, from changes in mission goals, needed upgrades, and bug-fixes, to accommodating environmental changes, through initiation from the base or other nodes in the network. The proposed framework can also be used as a “wireless JTAG” offering simultaneous hardware programming of multiple homogenous reconfigurable platforms. This is significant in systems where the hardware is not easily accessible or is expensive to access, resulting in more efficient maintenance cycles and reduced costs.

FPGA devices consist of an array of computational elements known as logic blocks, a set of routing elements, and a set of input/output cells, whose functionality is determined from configuration bits [5]. While the standard means for delivering configuration bits to the target FPGA is through a wired JTAG connection, other methods exist. One example describes a method for remotely reconfiguring FPGAs through the Internet [6]. This Internet Reconfigurable Logic (IRL) system includes a design

Manuscript received November 11, 2007. This work was supported in part by National Science Foundation grant IIS-0434156.

J. L. Wilder is with the Electrical and Computer Engineering Department, University of Alabama in Huntsville, Huntsville, AL 35899 USA. (phone: 256-876-5910; fax: 256-313-3717; e-mail: wilderj@eng.uah.edu).

V. Uzelac is with the Electrical and Computer Engineering Department, University of Alabama in Huntsville, Huntsville, AL 35899 USA (e-mail: uzelacv@eng.uah.edu).

A. Milenković is with the Electrical and Computer Engineering Department, University of Alabama in Huntsville, Huntsville, AL 35899 USA (e-mail: milenka@ece.uah.edu).

E. Jovanov is with the Electrical and Computer Engineering Department, University of Alabama in Huntsville, Huntsville, AL 35899 USA (e-mail: jovanov@ece.uah.edu).

containing an embedded processor and an FPGA, implementing some custom hardware function. New designs can be implemented on a host computer, and the reconfiguration payload can then be delivered to the target computer through a TCP/IP network. A second example is illustrated by Hulme *et al*, where they describe a configurable fault-tolerant processor (CFTP) that is used for spacecraft onboard processing [7]. CFTP is intended to evaluate, in various orbital regimes, different reconfiguration system-on-chip designs, such as a triple-mode redundancy, fault-tolerant circuit aimed at overcoming single-event upsets. In this example, the communication medium for passing the new FPGA configuration file is through the Internet, to a ground station, where the data is then transmitted to the orbiting satellite.

The remainder of the paper is organized as follows. Section 2 introduces the case for real-time reconfiguration through wireless sensor networks. Section 3 describes the hardware components of the wireless sensor nodes used in the REWISE testbed, and Section 4 describes the software modules and initial results of the field tests. Section 5 concludes the paper.

II. The Case for REWISE

This section of the paper describes an example REWISE system that uses reconfiguration as a design parameter. This system, as shown in Figure 1, is deployed in a military setting, behind enemy lines, where access is inherently limited, and it can be used to monitor types of traffic on local roads, capture vehicle signatures, obtain video or audio data, detect radiation sources, or monitor environmental conditions. Furthermore, signal processing of acquired sensor data is performed in order to provide data points to a base station and/or a command center. These data points can then be used to make strategic decisions for maintaining security in areas that are under surveillance.

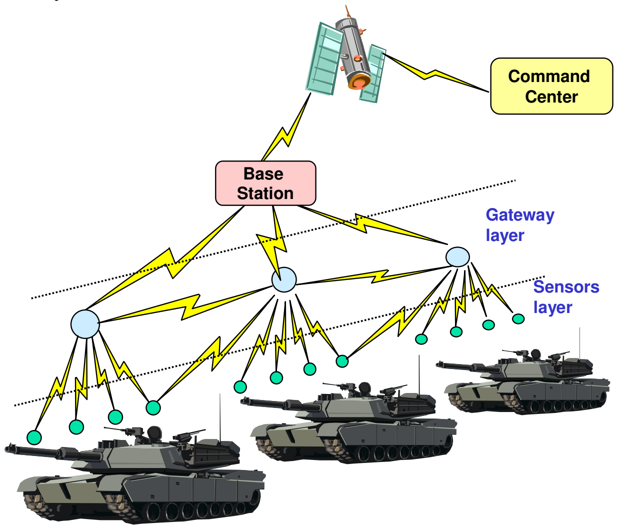

Fig. 1. A multi-tier reconfigurable wireless sensor network for surveillance and intelligent signal processing.

The system is made up of three different tiers. Since wireless communication is a fairly expensive operation in terms of power and time, this framework uses local hardware/software repositories at each layer to facilitate reconfiguration. The upper layers keep track of hardware/software configurations at the lower layers, and a command can be sent either to initiate reconfiguration from a local repository, or if the needed configuration is not present, begin downloading the required hardware/software context.

The sensor layer is at the lowest tier. A node at this level is comprised of a microprocessor, an FPGA, memory/storage, radio, an energy source, and an array of sensors capable of many different functions, such as measuring magnetic field, radiation, vibration, temperature, humidity, or gathering audio or video data. The number of nodes at the sensors layer could number into the thousands, and the sensor nodes should be so small that they can be embedded in the environment and not easily detected. As such, these nodes will have limited resources, requiring low-power operation. Since these nodes are resource-limited, the onboard storage can hold different hardware/software configuration data, which can be used to reprogram the FPGA as mission goals change (i.e., alter the function of a node from environmental monitoring to vehicle detection). Each sensor node will gather data, perform basic pre-processing, and then transmit data up to the gateway layer.

At the gateway layer, devices have more processing, storage, communication, and energy resources than at the sensors layer. These devices can be integrated with an available power grid, but this is not a requirement. A gateway node is comprised of a microprocessor, memory/storage, a radio, and possibly an FPGA, depending on processing needs. Gateway nodes could number into the hundreds, depending on the quantity of sensor nodes, and are responsible for establishing the communication network for the system. A gateway node receives data from the cluster of sensors they control and synergistically processes this data to make intelligent decisions based on mission goals. Each node at this level maintains a hardware/software repository with multiple FPGA configurations. Based on environmental conditions, the current mission, or the results of data processing, a gateway node can initiate reconfiguration at the sensors layer or download new hardware/software contexts to sensor nodes in preparation for near-term needs.

The next level up from the gateway layer is a centralized base station. The base station will have even more resources than nodes at the gateway layer and will serve as a means for providing global interface and control to the system. The base station can be under human-control or operate autonomously. All data from gateway nodes is gathered and processed at the base station for making decisions regarding mission goals. The base station can be used for initiating reconfiguration at the gateway layer or for downloading new hardware/software contexts to gateway nodes in preparation for future needs.

Finally, depending on the deployment scenario of the proposed system – perhaps the REWISE system is deployed overseas – a command center can be used to correspond with each base station through satellite communication. This would allow another means for communicating changing mission goals, and new hardware/software designs could be developed in a friendlier environment and then transferred to the system in the field. Furthermore, if the base station needed to be abandoned, then the command center would provide another level of control for initiating reconfigurations or modifying system functionality.

III. Hardware Design

The REWISE testbed that demonstrates reconfiguration through a wireless sensor network is shown in Figure 2. It consists of a base station, which is connected to a transmit mote through a serial link, a number of relay motes, and a receive mote connected to the reconfigurable sensor node platform through JTAG. The base station includes a software tool for initiating download of the configuration (XSVF file). The XSVF file is streamed serially to the transmit mote, and because of the limited resources of the hardware along the communication path, a flow-control scheme is used in order to direct the data reliably to the reconfigurable platform. The radio transceivers facilitating the wireless communication employ a networking protocol based on IEEE 802.15.4. Due to the limited range of this networking scheme, relay motes may be needed in order to traverse this path from source to destination. These relay motes will be a part of the wireless sensor network platform, and will typically employ some sensing function in addition to packet forwarding. Finally, a receive mote is connected through JTAG to the reconfigurable platform. The receive mote takes the configuration data and through a software JTAG module provides programming control of the reconfigurable device with the received XSVF file.

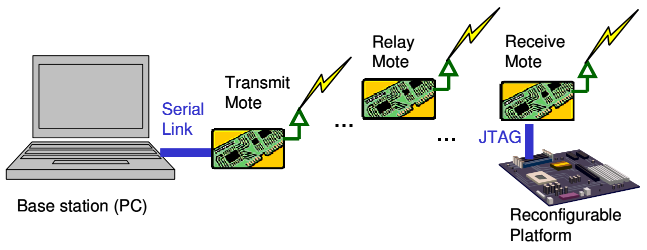

Fig. 2. The REWISE testbed.

All three types of wireless motes are based on a common platform that includes a Texas Instruments MSP430 microcontroller and a Chipcon CC2420 radio transceiver for wireless communication. The MSP430 includes a UART peripheral for communication with the base station and a synchronous peripheral interface (SPI) module for handling communication with the CC2420. The CC2420 implements the IEEE 802.15.4 networking protocol and achieves transfer rates of up to 250 kilobits per second.

On the transmit mote (Figure 3), the MSP430 uses direct memory access (DMA) transfer in order to efficiently pass data from the UART peripheral into memory buffers. These buffers constitute a buffering scheme that is used to differentiate between the two tasks of receiving data from the base station and transmitting data downstream through the radio. The MSP430 also handles the flow control for passing data successfully between the base station and the downstream motes. This is necessary because there is not enough onboard storage to allow transfer of the entire XSVF file at once, but it must be divided into smaller blocks.

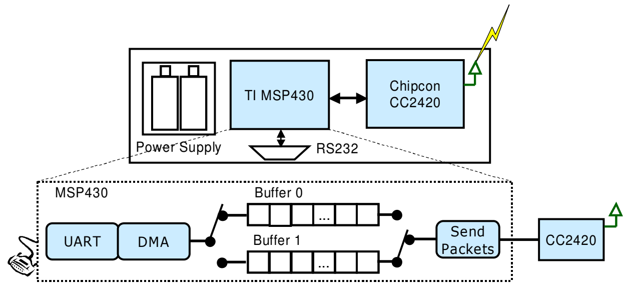

Fig. 3. Transmit mote diagram.

The receive mote (Figure 4) is connected to a reconfigurable wireless intelligent platform, which is used to both take readings and initiate control over the surrounding environment through sensors and actuators. The MSP430 on the receive mote handles the flow control responsibilities of the XSVF data stream, receiving the data from upstream nodes. It also implements the JTAG protocol for reprogramming the FPGA device.

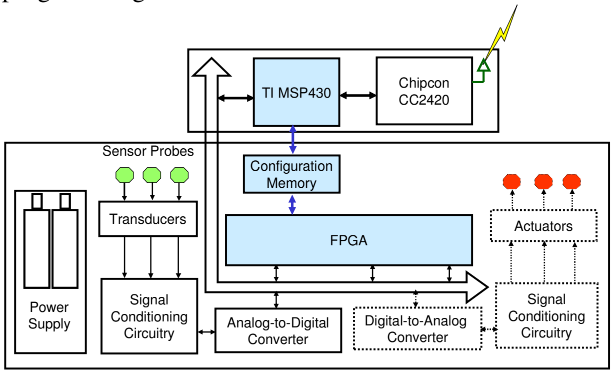
TI MSP430
Chipcon CC2420
Sensor Probes
Configuration Memory
Transducers
FPGA
Actuators
Power Supply
Signal Conditioning Circuitry
Analog-to-Digital Converter
Digital-to-Analog Converter
Signal Conditioning Circuitry

Fig. 4. Reconfigurable wireless intelligent platform with receive mote.

IV. Software Design

The software design in the REWISE testbed includes the modules for the base station, the transmit mote, and the receive mote. We start by describing the data flow that begins at the base station. This is a custom application running on a personal computer (i.e., the base station) and is responsible for initiating the reconfiguration process. This application, as described in Figure 5, takes a configuration file, which was created by design tools targeting a reprogrammable logic device and is in the compressed XSVF file format, and forwards this data serially on a byte-by-byte basis to the transmit mote. The transmit mote algorithm, as depicted in Figure 6, receives the byte sent from the base station into its MSP430 UART peripheral and subsequently feeds it into one of two memory buffers (henceforth referred to as the IN/OUT buffers) via a direct-memory access module. This process continues until one block of data (1 kB) has been sent.

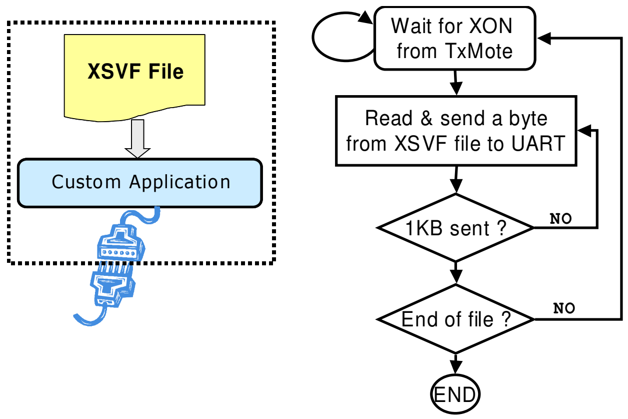

Fig. 5. Base station software application for initiating reconfiguration.

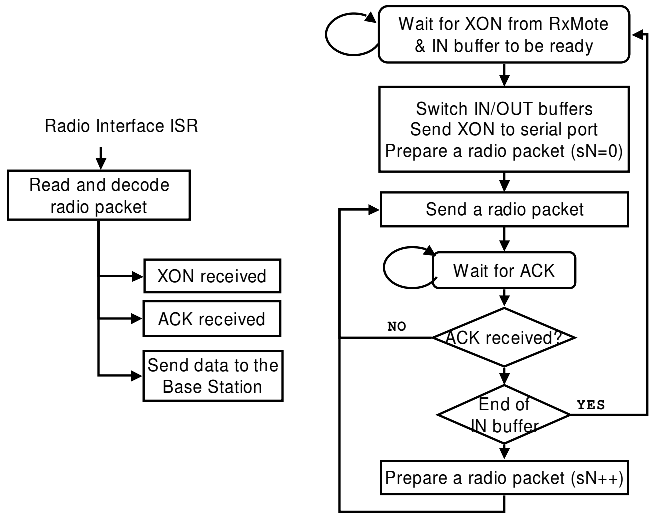
Radio Interface ISR
Read and decode radio packet
XON received
ACK received
Send data to the Base Station
Wait for XON from RxMote & IN buffer to be ready
Switch IN/OUT buffers
Send XON to serial port
Prepare a radio packet (sN=0)
Send a radio packet
Wait for ACK
NO
ACK received?
End of IN buffer
YES
Prepare a radio packet (sN++)

Fig. 6. Transmit mote algorithm.

After sending the first block of data, the base station application halts and waits to receive a transmit-on character (XON) from the transmit mote. Having received 1 kB of data from the base station, the transmit mote waits to receive an XON character from the receive mote. Upon receiving this response from the receive mote, the transmit mote switches its IN/OUT buffers and sends an XON character to the base station, requesting more data. While the base station begins sending data to the IN buffer in the transmit mote, 64 bytes of data from the OUT buffer are sent to the CC2420 radio transceiver for wireless transmission. These two tasks can occur simultaneously because the transfer of data into the memory buffer occurs via DMA, which is separate from processor control, allowing the processor to transfer data to the CC2420. The radio sends this packet of data and waits for a given period of time for an acknowledgment from a downstream mote (either a relay mote or a receive mote, as described in the REWISE testbed). If the acknowledgment is not received, then the radio packet is resent. If the acknowledgement is successfully received, the packet sequence number (sN) is incremented and the next 64 bytes from the OUT buffer is packetized and transferred to the radio in preparation for wireless transmission. This process continues on the transmit mote until the full kilobyte of data is sent from the OUT buffer.

The software application for the receive mote is provided in Figure 7. The receive mote gathers the packets that are sent from the transmit mote, as described in the previous paragraph.

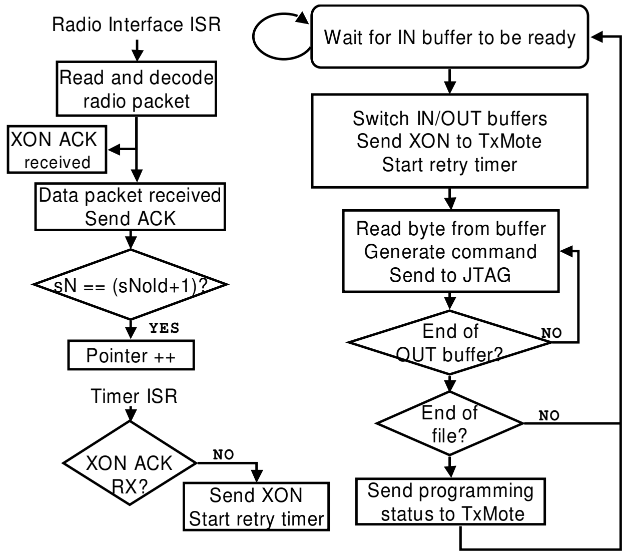

Fig. 7. Receive mote software algorithm.

Upon receiving a packet into the radio, it checks whether this is a data packet, and if so, sends the acknowledgment for that packet. At this point, the packet is decoded in order to ensure that it contains new data. This is done by checking the sequence number and making sure that this value is the same as the previously received packet's sequence number plus one. If that is the case, a pointer is incremented, and the data is transferred to one of two memory buffers (henceforth referred to as the IN/OUT buffers). If the new sequence number equals the old sequence number, then the transmit mote did not successfully receive the previous acknowledgment and this packet is discarded, while the acknowledgment is resent. This process will continue until 16 packets have been received, indicating that the IN buffer is now filled with a kilobyte of data. At this point the IN/OUT buffers are switched and an XON message is sent to the transmit mote. This XON message has the end goal of telling the base station that the next kilobyte of data can be sent.

In order to ensure that the configuration data stream flow is not cut off, the receive mote requires an acknowledgment of this XON message from the transmit mote. This is accomplished through a "retry timer". The retry timer is started as soon as the XON message is sent. Once the timer reaches a certain count, the interrupt service routine is entered and a check is performed to see if the XON message has been acknowledged. If it has not been acknowledged, then the XON message is resent in order to not cutoff the flow of data, and the retry timer is started again.

After the IN/OUT buffers have been switched, data begins to be read from the OUT buffer and sent to the JTAG configuration software module, which then controls the emulated JTAG lines on the MSP430 for the purpose of configuring the reprogrammable device. This JTAG module reads the instructions and arguments from the XSVF file, operating on successive bytes from the OUT buffer until the end is reached. Subsequently, the application returns to the state of waiting for the IN buffer to fill, and then the process is repeated until the end of the XSVF file is reached. At this point, the JTAG module provides notification for whether the FPGA was successfully reconfigured. This status is communicated to the transmit mote through the radio, which passes these results to the base station.

With the MSP430 processor on the receive node running at 8 MHz, the MSP430 processor on the transmit node running at 6 MHz, and the RS232 port on the base station configured for 115.2 kbits/sec, initial test results based on the REWISE testbed show that it takes 6 msec to transmit and receive acknowledgment of one packet containing 64 bytes worth of data. Thus, it takes 96 msec to transmit 16 packets or 1kB worth of data. This corresponds to a bandwidth of 83 kbits/sec. With a Virtex family FPGA needing an XSVF file containing 675 kB of configuration data, this requires 65 seconds to transmit. Smaller XSVF files are on the order of 45 kB, while larger devices would need around 2.7 MB. FPGAs with a greater degree of capability need more SRAM bits for programming. These values indicate file sizes for complete reconfiguration of the FPGA. However, it is now possible to only perform partial reconfiguration of a device, so this would lower the amount of data needed for transfer, and therefore decrease the overall time required for reconfiguration. Factoring in the JTAG programming time performed on the receive node, which is done in 1kB blocks (set according to the buffer size and available resources in the MSP430 processor) it takes 210 msec for 1kB worth of data transfer. This is the amount of time it takes for the transmit node to send 1kB of data until the XON message is received from the downstream node and the next 1 kB of data is ready to be sent. Thus, the effective bandwidth for transfer and JTAG programming becomes 38 kbits/sec. For an XSVF file containing 675 kB of configuration data, this requires 142 seconds to complete the programming of the downstream reconfigurable node.

V. CONCLUSIONS

FPGAs emerge as a technology of choice that strikes an optimal balance between processing power, energy requirements, and flexibility. Furthermore, leveraging the ability to reprogram FPGAs allows systems to be reconfigured after deployment, which is valuable in terms of handling upgrades, fixing bugs, accommodating modified goals and environmental changes, and maintaining a large amount of hardware in difficult-to-reach locations. To achieve these desired goals, we have proposed a REWISE framework, which consists of reconfigurable wireless intelligent sensor nodes, capable of broadening the functionality and capability of wireless sensor networks. We have given a test case that illustrates the power of a REWISE system, and we have described hardware and software components of a prototype design.

For rapid prototyping, a commercially-available hardware development board was chosen as the basis for the design. The resource limitations of the processor and the radio led to the design choices that were made, such as the transfer block size that was chosen (due to packet overhead and available memory in the MSP430), and the available speed of the point-to-point communication scheme that was used. Higher effective bandwidths and faster FPGA configuration times could be achieved with a higher-performance processor and a radio with enhanced resources. In light of these considerations, the prototype design is a positive first-step towards achieving a REWISE system and demonstrates the proof of the concept.

REFERENCES

[1] D. Culler, D. Estrin, M. Srivastava, "An Overview of Sensor Networks", IEEE Computer, pp. 41 – 49, August 2004.

[2] D. Culler, W. Hong, "Wireless Sensor Networks", Communications of the ACM, Vol. 47, No. 6, pp. 30 – 33, June 2004.

[3] Xilinx, "Spartan-3L Low Power FPGA Family", September 2005.

[4] J. Rabaey, M. Ammer, J. Silva Jr., D. Patel, S. Roundy, "PicoRadio Supports Ad Hoc Ultra-Low Power Wireless Networking", IEEE Computer, Vol. 33, No. 7, pp. 42 – 48, July 2000.

[5] K. Compton, S. Hauck, "Reconfigurable Computing: A Survey of Systems and Software", ACM Computing Surveys, Vol. 34, No. 2, pp. 171 – 210, June 2002.

[6] Xilinx, "Architecting Systems for Upgradability with Internet Reconfigurable Logic", June 2001.

[7] C.A. Hulme, H. H. Loomis, A. A. Ross, and R. Yuan, "Configurable Fault-Tolerant Processor (CFTP) for Spacecraft Onboard Processing," 2004 IEEE Aerospace Conference Proceedings, Vol. 4, pp. 2269 – 2276, March 2004.