

Mission Operation for Reconfigurable Spacecraft

Savio Chau, Adans Ko, Kar-Ming Cheung

Jet Propulsion Laboratory, California Institute of Technology,
4800 Oak Grove Dr., Pasadena, CA 91109, USA.

Abstract

Nowadays, flight qualified Field Programmable Gate Array (FPGA) contains millions of gates. This technology enables many new applications in mission operations such as in-flight reconfiguration for fault recovery, in-flight reconfiguration for resource planning in different mission phases, multiple science objectives spacecraft, and technology upgrade during long life missions. On the other hand, the new applications will also create many new challenges for mission operations. These challenges include how to reconfigure the hardware of the spacecraft in flight, how to transmit the large reconfiguration file to the spacecraft, and how to verify the reconfiguration is executed correctly. This paper will examine the issues and solutions to these challenges. First, a brief survey of the current FPGA technology and the process to reconfigure these devices will be provided. Second, the avionics architectural support to facilitate the in-flight reconfiguration will be discussed. Third, the mission operation procedures and uplink/downlink process to reconfigure the spacecraft in-orbit will be described. The main concern of these procedures is the safety of the spacecraft during and after the reconfiguration. Finally, a set of commands and telemetry required to execute the in-flight hardware reconfiguration will be proposed.

1. Introduction

Reconfigurable technology, especially the Field Programmable Gate Array (FPGA), allows a design to be changed relatively easily even after it has been integrated into the system. In addition, since FPGA is relatively low cost and can be "programmed" individually, it is much more cost effective for small quantity production. For these reasons, the FPGA is an attractive technology for space system designs. Previously, FPGA technology was not seriously considered for space applications because of their radiation tolerance did not meet the requirements of most flight missions. However, some radiation tolerant FPGAs that are suitable for a large number of space applications are available nowadays. It can be expected that the radiation-hardened FPGA technology would be further improved in the future.

On the other hand, the current application of FPGA in space systems is only limited to the prototyping or replacement of application specific integrated circuits (ASIC) in avionics systems. In reality, the FPGA technology can enable many new applications in space that require in-orbit or in-flight reconfiguration. Examples of these new applications include:

1. In-flight reconfiguration to correct hardware design error after launch.
2. In-flight reconfiguration for fault recovery: an avionics system can include some generic spare FPGAs that can be "programmed" to replace multiple types of failed components, so that the amount of redundant components required by fault tolerance can be reduced.
3. In-flight reconfiguration in different mission phases for resource planning: by changing the amount of hardware, it is possible to trade performance with power in different mission phases. For example, the computing workload of the flight processor during the cruise phase of a mission is very low. Therefore, the FPGA can be configured as a moderate performance controller to stand in for the processor. This would allow the main flight processor to be powered off and thus reduce the power consumption of the system. On the other hand, during mission critical phases, the FPGA can wake up the flight processor for general operations while reconfigures itself as a piece of special purpose hardware to take over the execution of high demand software. Since hardware usually has much better performance/power ratio, this technique can boost the system performance but only increase the power consumption modestly.
4. In-flight reconfiguration to enable the same hardware to be used for multiple purposes: many missions in the future might desire to visit several destinations to carry out multiple science objectives at different time. This is not practical if each science objective requires a different set of hardware, but it is very feasible if the same set of hardware can be reconfigured for multiple purposes.
5. Technology Upgrade During Long Life Missions: with the advances in power generation technologies, many missions in the future would be capable to carry out ultra-long life missions such as visiting the edge of the Solar System. One challenge to mission operations these missions create is the maintenance of a crew who would understand the technologies that would become obsolete toward the end of the missions. By including several large spare FPGAs, it is at least possible to replace the original processor with newer generation processors, so that the maintenance of the software would be much easier.

While the FPGA technology would enable many new applications, these applications will also create new challenges for mission operations. First of all, the process to reconfigure the on-board FPGAs from ground has not been fully understood. Second, robust process must be developed to safeguard against errors in hardware reconfiguration. Third, mission operation traditionally assumes the hardware design is well verified. However, as the hardware is now reconfigurable, this assumption is no longer valid. This will create more uncertainty in diagnostics when problem occurs. Finally, modern FPGA usually contains millions of gates, so a configuration data file also has several megabytes. Many deep space missions have data rate of only a fraction of kilobits per second. Sending such large data file will take up significant uplink time.

This paper will examine the issues and solutions to these challenges. First, a brief survey of the current reconfigurable technology and the techniques to program these devices will be provided. Second,

the avionics architectural support to facilitate the in-flight reconfiguration will be discussed. Third, the mission operation procedures and uplink/downlink process to reconfigure the spacecraft in-orbit will be described. The main concern of these procedures is the safety of the spacecraft during and after the reconfiguration. Fourth, a set of commands and telemetry required to execute the in-flight hardware reconfiguration will be proposed. Finally, new technologies in mission operation and telecommunication to support in-flight reconfigurable spacecraft will be recommended.

2. Survey of the Reconfigurable Technologies

Nowadays, the term “reconfigurable devices” is almost synonymous with FPGA. In reality, there are many types of reconfigurable devices exist. In fact, even EEPROM and flash memory can be considered reconfigurable devices. However, this study only focuses on reconfigurable devices for implementing complex logics. These devices can be categorized as follows.

- *PLD* — a programmable logic device (PLD) is an integrated circuit that can be configured by the end user by placing the chip into a special programming unit. Some PLDs can also be configured “in-system”.
- *PLA* — a Programmable Logic Array (PLA) is a relatively small PLD that contains two levels of logic, an AND-plane and an OR-plane, where both levels are programmable
- *PAL* — a Programmable Array Logic (PAL) is a relatively small PLD that has a programmable AND-plane followed by a fixed OR-plane
- *CPLD* — a more Complex PLD that consists of an arrangement of multiple simple PLD-like blocks on a single chip.
- *FPGA* — a Field-Programmable Gate Array is a PLD featuring a general structure that allows very high logic capacity. Whereas CPLDs feature logic resources with a wide number of inputs (AND planes), FPGAs offer more narrow logic resources. FPGAs also offer a higher ratio of flip-flops to logic resources than do CPLDs.

In comparison, FPGA has been making more advances than other types of devices. Modern FPGA contains millions of gates and even incorporates embedded processor cores. Therefore, FPGA has become the dominating reconfigurable technology for space application. Table 1 lists some of the state-of-the-art FPGAs. Most of the FPGAs are designed for ground applications only. However, a few of them, such as those offered by Xilinx and Actel are suitable for flight missions that have moderate radiation environments.

Among the types of FPGAs shown in Table 1, the anti-fuse technology is not suitable for in-flight reconfiguration since it can only be written once. Either the SRAM or EEPROM based FPGA are suitable. The EEPROM FPGA has an advantage over the SRAM FPGA in that its configuration memory is non-volatile, so that the configuration of the FPGA does not have to be reloaded every time the system is

powered up. Unfortunately, there is no flight qualified EEPROM based FPGA available at this time. Therefore, this paper will only focus on the use of SRAM based FPGAs.

Table 1: FPGA Survey Results

Supplier	State-of-the-Art Devices	Number of System Gates	Connection Technology	Embedded Processor Core	Embedded SRAM	Flight Parts Availability
Xilinx	Virtex II Pro (XC2VP125)	5 M gates	SRAM	4 Power PC 405	10 Mbits	Projected 200K TID No latch up at LET=100 TID>100K, LET >125
	Virtex II (XC2V8000)	8 M gates	SRAM	No	3 Mbits	
	XQVR1000	1.1 M gates	SRAM	No	131 Kbits	
Actel	AX2000	2 M gates	Anti-fuse	No	295 Kbits	No No TID<100K, LET>40
	APA1000	1 M gates	Flash	No	198 Kbits	
	RT54SX72S	108 K gates	Anti-fuse	No	None	
Altera	Excalibur (EP20K1000K)	1.77 M gates	Anti-fuse	1 ARM922T	327 Kbits	Not yet, but interested
Lattice	ispXPGA 1200	1.25 M gates	SRAM + on-chip EEPROM	No	414 Kbits	Not yet, but interested
Cypress	CY8C2X443 family (mixed signal)	8 digital blocks 12 analog blocks		8-bit Harvard Architecture MC8 Microcontroller	256 bytes SRAM 16 Kbytes flash	No
Atmel	AT40KAL	50 K gates	SRAM	No	18.4 Kbits	No, but Atmel offers rad-hard ASICs
Quicklogic	Eclipse Family QL7180	662 K gates	ViaLink (fuse like)	No, but has 18 embedded ALUs	83 Kbits	No
Lucent	ATT3000 and ORCA series	< 6 K gates too small to consider	SRAM			

3. Techniques to Reconfigure FPGAs

A SRAM based FPGA contains a large number of unconnected logic unit called **configurable logic block (CLB)**. The connections among the CLBs are controlled by switches or multiplexers, which in turn are controlled by SRAM cells built into the FPGA. This is depicted in Figure 1. The array of SRAM cells used for configuring the connection switches in the FGPA is called the **configuration memory**. The functions of each CLB can also be configured by the configuration memory. The content of the configuration memory is called a **configuration file**.

In order to configure a SRAM based FPGA to perform the desired functions, its configuration memory has to be loaded with a configuration file at system start up. The techniques to load the configuration file into FPGA can be classified into three categories: Master Mode, Slave Mode, and Boundary Scan Mode.

These modes are selected by setting some mode pins on the FPGA. For example, the Xilinx Virtex family FPGAs have three such mode pin [1].

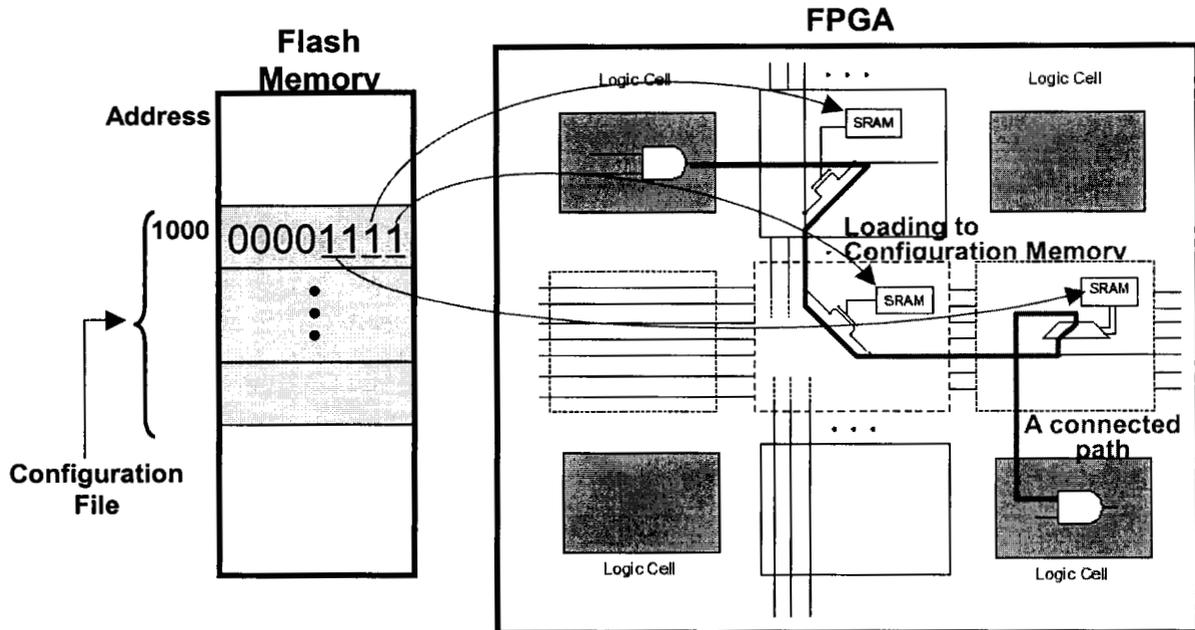


Figure 1: Configuration of FPGA

Master Mode

In this mode, the configuration file is permanently stored in a non-volatile flash memory. The FPGA has built-in control circuit that provides clock and control signals to read the flash memory. If the FPGA is set to this mode, the FPGA automatically reads the flash memory upon reset and then puts the configuration file data into its configuration memory. An example of a FPGA in Master Mode setting is shown in Figure 2 [1].

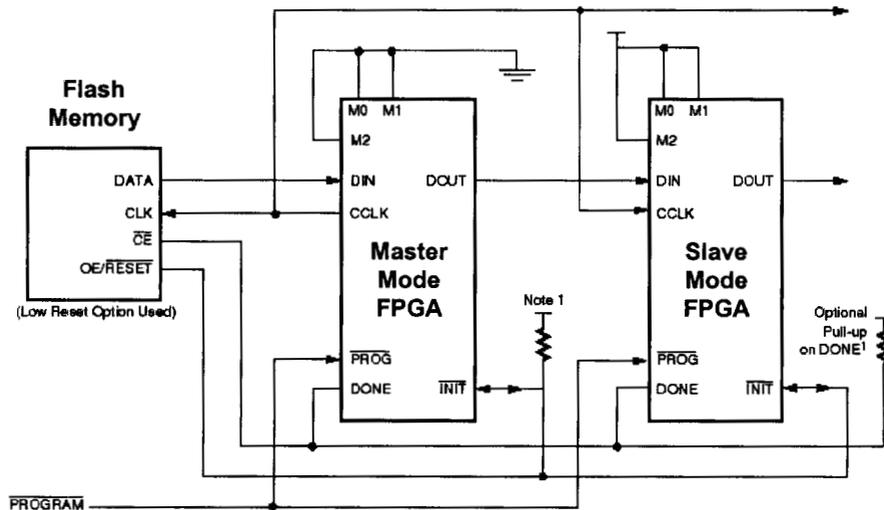


Figure 2: FPGAs in Master and Slave Mode Setting

Figure 2 shows that the configuration file is loaded into the Master Mode FPGA in bit-serial data stream. In other settings, the FPGA can also be configured with parallel data stream. The Master Mode configuration using parallel data stream is called the **Master SelectMAP Mode** in the Xilinx Virtex family FPGAs [1]. The SelectMAP Mode doesn't allow the daisy chain setting as shown in Figure 2 and thus requires each FPGA has to have its own flash memory. However, the major advantage of the SelectMAP mode is that it allows the configuration memory to be read for verification. For this reason, the Master Mode is actually referring to the Master SelectMAP Mode hereafter.

Slave Mode

In this mode, the configuration file data can be either permanently stored in a flash memory or supplied by an external source such as a processor or another Master Mode FPGA. Upon reset, the external source either loads the configuration file directly into the FPGA configuration memory, or drives the FPGA to read the flash memory with a clock signal. The right most FPGA in Figure 2 is an example of a Slave Mode FPGA and how it can be configured by a Master Mode FGPA.

The Xilinx Virtex FPGA family also allows parallel data stream to load the FPGA in Slave Mode. The Slave Mode configuration using parallel data stream is called the **Slave SelectMAP Mode** in the Xilinx Virtex family FPGAs [1]. The Slave SelectMAP Mode also has to have its own flash memory and allows the configuration memory to be read for verification. The Slave aster Mode is also referring to the Slave SelectMAP Mode hereafter.

Boundary Scan Mode

In this mode, the configuration file is loaded from the flash memory to the FPGA through the boundary scan, which can load any internal register or memory in the FPGA by shifting a serial-bit stream through a standard interface called JTAG (i.e., IEEE 1149.2 standard). JTAG was originally designed for testing, but it has been used for many other purposes nowadays due to its capability to load data into internal registers of a chip. However, since this mode requires specialized hardware and software support that are not available in space. Therefore, this mode will not be considered any further in this paper.

4. Architectural Support for In-Flight Hardware Reconfiguration

The aforementioned techniques of reconfiguring FPGA are usually done in ground systems. In order to extend those techniques to reconfigure FPGA in space, additional architectural supports in the avionics system are required. These support are described in the following.

First, in order to reconfigure the FPGAs in space, they have to be SRAM or EEPROM based. As mentioned before, flight qualified EEPROM based FPGAs are not available at this time, so the SRAM based FPGAs is the only option. The configuration memory of the FPGA can be loaded either by the flight processor or from a non-volatile memory device such as flash memory. However, for those FPGA that has the potential to implement critical functions of the spacecraft, it is more desirable that their

configuration memory can be loaded without the need of a processor. For those FPGAs that implement non-critical functions, their configuration memory can be loaded with the help of the flight processor every time when they start up. In other words, the FPGAs that might implement critical functions should use the Master Mode configuration (or at least should have Master Mode as an option), and the FPGAs that implements non-critical functions can use the Slave Mode configuration.

Second, for those FPGA that use Master Mode configurations, their flash memories have to have direct access from the Telecommunication System, so that the configuration file can be loaded to their flash memories directly from the ground (direct from Earth link) or from another spacecraft (relay link). This requires that there is a robust end-to-end data delivery mechanism, and the telecommunication system to have the capability to directly interface with the flash memories, to handle the writing of data into the flash memories, and to jumpstart the avionics system to perform some basic functions. Recent advances in lower layer link protocols [3][4] enable reliable communications using autonomous retransmission, and allow interoperability among spacecraft and between spacecraft and Earth. The flight communication system has to be able to be directly commanded from the ground or from another spacecraft, and be equipped with microprocessors and onboard memory that can be extended to perform the basic interfacing functions with the FPGAs (and other functions) in the avionic system.

Third, the Telecommunication System also has to have capability to read back the configuration memories of the FPGAs under the ground commands. This will allow the mission operation to verify if the configuration memories have been loaded correctly. The implementation of this capability is similar to that of loading the flash memory.

Finally, since changing the hardware of a spacecraft is a critical operation. Therefore, safeguard has to be built into this process in the mission operation. A technique called the Guarded Upgrade, which is similar to the Guarded Software Upgrade technique [2], will be employed to provide this safeguard. Details of this technique can be found in [2] and an example of the guarded upgrade operation will be discussed in the following.

An example of a space system that has these recommended architectural supports for in-flight hardware reconfiguration is shown in Figure 3.

5. Procedures to Reconfigure Spacecraft Hardware In-Flight

This section will examine the procedure to reconfigure the FPGAs in a flight system using the Guarded Upgrade technique. This procedure can safeguard the hardware reconfiguration but it requires the system to have blank spare FPGAs. This is illustrated in Figure 3. The spare FPGAs cannot output any data or signals to the I/O or system memory. The Guarded Upgrade procedure is explained as follows.

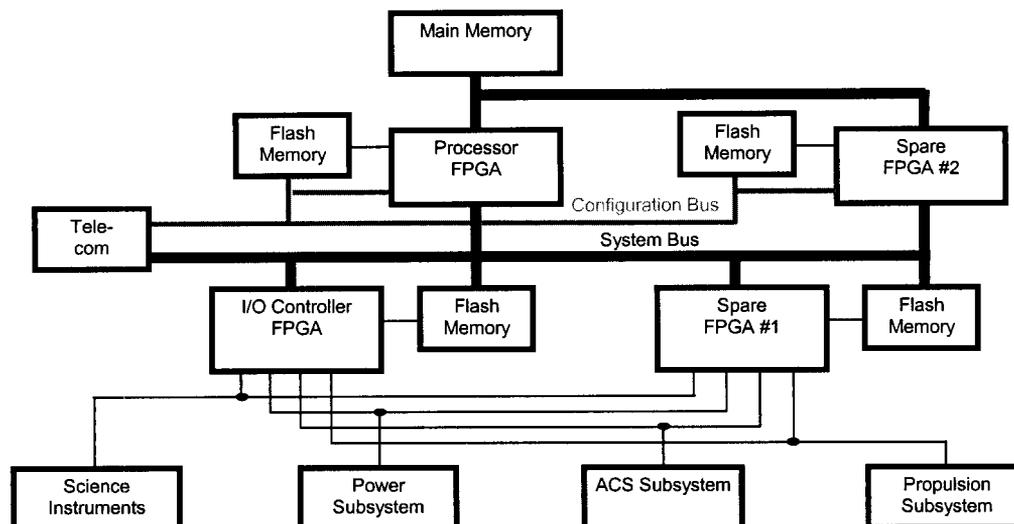


Figure 3: In-Flight Hardware Reconfigurable Flight System

During the flight mission, when the occasion arises that in-flight hardware reconfiguration has become necessary, the ground operator will send a command to the spacecraft to prepare for the loading of one of the spare FPGA. This command will specify which FPGA will be loaded. Then the configuration file is sent from the ground and the Telecommunication System will load the file to the configuration memory of the specified FPGA. Thus the spare FPGA is now configured to perform the required functions.

If the newly configured FPGA is to perform non-critical functions, then its I/O pins can be activated by a hardware command so that it can start to receive inputs and produce output signals. The ground operator can optionally use a normal command to read out, downlink, and verify the content of the configuration memory of the FPGA before it is activated.

If the newly configured FPGA has to perform a critical function (e.g., attitude control, power distribution etc.), then the Guarded Upgrade procedure has to be used. In most of the cases, the critical function may have already been implemented somewhere on the spacecraft. Some of the reasons for that critical function to be moved into one of the spare FPGA are as follows. First, the critical function may have been implemented by another FPGA but a design error has been found and need to be corrected. Second, the critical function may have been implemented by software and then has to be moved into hardware to meet performance requirements. Third, the hardware design of the critical function has to be updated. For example, the flight processor has to be updated to a newer generation.

In all these cases, the Guarded Upgrade procedure can use the old implementation as a reference. Hence, the newly configured FPGA will first be put into an observation mode, in which the FPGA will receive all the inputs and execute the function. However, its outputs will not control any devices or write

into any system memory. Instead, its outputs are used to compare with the outputs of the old implementation. The comparison can take place internally in the FPGA by temporarily converting its output pins into input pins. This can be done by a hardware command. By comparing with the old implementation, most functions of the FPGA can be verified except for the differences between the old and the new implementations. The differences can be verified by on-board simulation with the processor, which produces the necessary inputs and then reads out the output registers of the FPGA for verification.

The newly configured FPGA can be promoted to the active role and start to control I/O devices and writing to system memories when all of its functions are verified. The old implementation can first be put into backup mode in case the new FPGA fails unexpectedly. The old implementation can retire or removed from the system when the new FPGA has been operating flawlessly for some period of time. The promotion of the new FPGA and retirement of the old implementation can be achieved either by normal or hardware commands.

If the newly configured FPGA has to implement a new critical function that does not exist on the spacecraft, then its operations have to be verified by on-board simulation, where the FPGA will receive real inputs and perform the critical function. However, its outputs will only be read by the flight processor but will not control any I/O devices or write into any system memory. The advantage of on-board simulation over ground-based simulation is that the inputs are much more realistic. After the operations of the new FPGA are fully verified, then the outputs of the FPGA can be activated by a normal or hardware command.

If the flight processor is working properly, the spare FPGA can be configured with the Slave Mode. However, for those spare FPGAs that have the potential to be configured to perform critical functions, they should have the options of using either the Slave Mode or Master Mode for configuration. For those FPGAs that have the capability to replace the flight processor, it is critical that they should use the Master Mode configuration. The main difference between the Slave Mode and the Master Mode is that the Slave Mode only requires normal commands to initiate the configuration while the Master Mode has to use hardware commands to do that.

6. Recommended Commands In-Flight Hardware Reconfiguration

The above description has highlighted some of the commands required to support in-flight hardware reconfiguration. These commands can be grouped into two categories, the normal commands and the hardware commands. Following is a list of the recommended commands and their parameters (in parenthesis). The functions of these commands can be found in the description of the In-Flight Hardware Reconfiguration Procedures

Normal Commands:

- Load configuration memory (FPGA ID)
- Load FPGA with slave mode (FPGA ID)
- Downlink FPGA configuration memory data (FPGA ID, starting address, range)
- Set FPGA to observation state (FPGA ID)
- Set FPGA to active state (FPGA ID)
- Reset FPGA (FPGA ID)
- Set FPGA to spare state (FPGA ID)

Hardware Commands

- Load configuration memory (FPGA ID)
- Load FPGA with master mode (FPGA ID)
- Downlink FPGA configuration memory data (FPGA ID, starting address, range)
- Reset FPGA (FPGA ID)

7. Conclusion and Future Work

This paper has highlighted many potential space applications of the FPGAs, especially the in-flight hardware reconfiguration in future flight missions. The architectural support to facilitate in-flight hardware reconfiguration and the procedure to perform in-flight hardware reconfiguration were also discussed in details. A set of commands for in-flight hardware reconfiguration has also been proposed. However, the uplink procedure to send these commands and the downlink procedure to read back the configuration memories need further development. The Telecommunication System designs to support the decoding of these reconfiguration commands have to be developed. The architectural support for in-flight hardware reconfiguration and the validation/verification of the reconfiguration procedure also need to be further refined.

8. Acknowledgment

The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References:

- [1] Xilinx Inc., Virtex-II Pro™ Platform FPGA Documentation, page 87, March 2002
- [2] A. Tai, L. Alkalai, S. Chau, Performance Evaluation, vol. 44, page 211 (2001)
- [3] Proximity-1 Space Link Protocol, Recommendation for Space Data System Standards, CCSDS 211.0-R-2. Red Book. Issue 2. Washington, D.C., CCSDS, January 2000.
- [4] X.25 Recommendation (10/96) Interface between Data Terminal Equipment (DTE) and Data Circuit-terminating Equipment (DCE) for terminals operating in the packet mode and connected to public data networks by dedicated circuit, ITU-T.