A Low Complexity Image Compression Solution for Onboard Space Applications

Antonio Lopes F.
National Institute for Space Research
Divisão de Eletrônica Embarcada
Av. dos Astronautas 1758, S. J. dos Campos
SP - Brazil
55 12 3208-6812
alopes@dea.inpe.br

Roberto d'Amore
Instituto Tecnológico de Aeronáutica
Divisão de Engenharia Eletrônica
Pça. Mal. Eduardo Gomes 50, S. J. dos Campos
SP - Brazil
55 12 3947-6876
damore@ita.br

ABSTRACT
In this work, a real time hardware data compression solution for raster scan cameras, to be onboard the next generation of Remote Sensing Brazilian satellites, is proposed. The options for image data compression methods are briefly covered to substantiate the choice: a low complexity implementation based on JPEG-LS, near-lossless compression algorithm, which can be synthesized in a single electronic device. JPEG-LS performance in terms of compression rates and loss is evaluated by processing remote sensing rough images through in-house developed software tool. Finally, the description and results of an implementation in FPGA are presented and compared to other works, emphasizing the differences related to application requirements.

Categories and Subject Descriptors
B.7.1 [**Integrated Circuits**]: Types and Design Styles – *Algorithms implemented in hardware.*

General Terms
Algorithms, Performance, Design, Experimentation, Verification.

Keywords
Space applications, On board, Image, Compression.

1. INTRODUCTION
The upgrading performance of new generations of Remote Sensing Instruments for Earth observation is in the sense of higher spatial, radiometric and spectral resolutions. In consequence, additional on-board mass memory, downlink data rate and power consumption have to be considered for real time operation. Furthermore, available x-band microwave link for Earth Exploration-Satellite Service (EESS) ruled by Radiocommunication Sector of International Telecommunication Union (ITU-R) have stringent limitations in power and bandwidth, frequently shared by different instruments in satellites.

SBCCI'10, September 6–9, 2010, São Paulo, Brazil.

Data compression is the usual solution to save data storage, data rate (or transmission time for low rate links), bandwidth and power resources at the cost that data compression algorithms are complex for onboard software or hardware implementations, and compressed data are more susceptible to errors. On board satellite compression implementation options in use [6] are based on software, hardware or in a both software and hardware solution. Complex and massive data processing can be achieved with the use of software based solutions like DSP or a high speed CPU, for example. The spreading use of hardware implementations, based on ASICs and more recently on FPGAs, is mainly due to its relatively higher throughput. By mixing software and hardware implementation the best of both worlds can be explored.

Budget, technology and commercial restrictions on qualified components are the guide lines on which the work of searching for a solution among the options for compression methods and feasible hardware implementation possibilities were carried on, in detriment of more software demanding implementation options. JPEG-LS is a mature compression algorithm and a commercial intellectual property (IP) core is available in an enclosed and general purpose application version [10]. Related works in the area have been carried out with good results for throughput but with no resources of JPEG-LS near-lossless capabilities [4], or with excessive area usage [3], and without any mention to error propagation mitigation. Although JPEG-LS near-lossless mode compression results are worse than the obtained by other lossy compression methods [4], it offers a good trade-off between performance and complexity. Moreover, differences in application requirements, like fault-tolerant design and control under space application point of view, justify a customized solution.

In this work a real time hardware data compression solution for raster scan remote sensing cameras is proposed. The compression method is a low complexity implementation based on JPEG-LS, near-lossless compression algorithm, to be synthesized in a single electronic device.

In Section 2 of this paper, image compression basis and the most common methods for onboard satellite systems, including JPEG-LS standard, are briefly covered. In Section 3, the performance of JPEG-LS for remote sensing images is evaluated by software test results; the proposed hardware implementation is described; and implementation results are presented. In Section 4, final conclusions are exposed.

2. IMAGE COMPRESSION METHODS

Basic concepts explored in data compression methods are entropy, correlation and compression rate. The entropy figure of a memoryless or uncorrelated source is related to redundancy of its information contents so that the higher the entropy the lesser the possibilities of compression. Correlation is a characteristic of sources with memory where there is some kind of data interdependency. In the case of image data, the compression is evaluated also by the resulting distortion or the peak signal to noise ratio (PSNR) figure, obtained from the mean square error between original and the recovered image after encoding/decoding processes.

Remote sensing imaging data present some characteristics that contribute to compression: low entropy, caused by atmospheric effects that reduces contrast and dynamic range and results in a concentration of signal at mid levels; strong correlation between different regions of an image and between different spectral bands.

An image compression process consists, basically, of a decorrelator to map image information into a lower correlation space, and an entropy coder to optimize output information entropy. Most of the onboard compression systems in use [6] are employing prediction or transform decorrelation techniques. In predictive differential methods, an estimate of the pixel to be encoded is obtained from previous adjacent pixels. The difference between the pixel and the prediction, or the error, is then encoded through an entropy coder. In the decompression process the pixel is recovered by adding the prediction to the decoded error. The Consultative Committee for Space Data Systems - Lossless Data Compression (CCSDS 121.0-B-1 Recommendation), JPEG lossless [7] and JPEG-LS standards are examples of predictive differential methods.

In the transform based methods, the range of options for reversible decorrelating transforms and coders is large and still being explored. JPEG standard [7] is based on Discrete Cosine Transform (DCT). JPEG2000, Set Partitioning in Hierarchical Trees (SPIHT), and CCSDS Image Data Compression (CCSDS 122.0-B-1 Recommendation) are based on Discrete Wavelet Transform (DWT). Lossy, good PSNR, and high compression rate results can be achieved with transform based methods but, on the other hand, the transforms are multi pass, complex operations, requiring storage resources for image blocks during processing.

JPEG-LS international standard [8] defines a set of lossless or near-lossless compression methods for coding continuous-tone, gray-scale, or color digital still images. Low Complexity Lossless Compression for Images (LOCO-I) algorithm [5] is the basis of JPEG-LS standard. A simplified block diagram of LOCO-I is represented in Figure 1. An important control parameter of this algorithm is δ, named NEAR in JPEG-LS standard, which defines the amount of near-lossless compression ($\delta = 0$ for lossless compression). The modeling approach is based on the notion of "context", where for each sample x a context is determined from gradients calculated by using the neighborhood reconstructed samples a, b, c, and d (refer to Figure 1). Each context is represented by an integer number that indexes four variables referred here as context mode variables (CMV). In low entropy, or flat regions of the image, if the context estimates that neighbor samples are identical (or nearly identical, in near-lossless coding) then the run mode is selected, otherwise regular mode is selected. In regular mode, a predictor combines the reconstructed samples, a, b, c, and d, to form a prediction of x. The prediction error is computed, as the difference between the sample x and its predicted value, and then corrected by a context-dependent term to compensate for systematic biases in prediction. In the case of near-lossless coding, the prediction error is quantized. Finally, the error is then encoded using a context dependent Golomb coding.

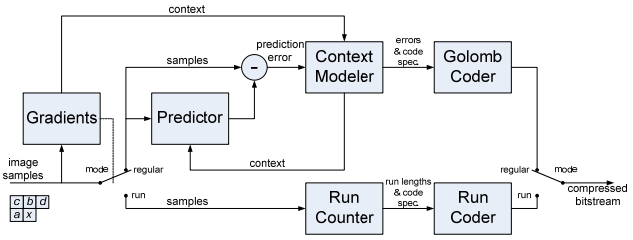


Figure 1. LOCO-I Algorithm Simplified Block Diagram.

In run mode, the encoder seeks, starting at x, for a sequence of consecutive samples identical or nearly identical to the reconstructed sample a. The run ends when this condition is no more verified or by the end of the current line, whichever comes first. The length information, which also specifies one of the two run-ending alternatives, is then encoded using a procedure extended from Golomb coding, with improved performance and adaptability.

Main drawback of predictive differential methods is the susceptibility to error propagation. Processing errors, like single-event upsets (SEU) in the cumulative context variables, or transmission errors will be propagated until the end of a scan. Besides SEU mitigation and error detection and correction (EDAC) methods, error propagation can be contained by the exploiting restart intervals. In this scheme, preview in JPEG-LS standard, the process is restarted more frequently along a scan and the transmitted data is broken in small independent packets. Transmission error propagation containment results in a decrease in the compression factor.

JPEG-LS is a one pass method that uses integer operations so that memory requirements are limited to a single image line. In transform based methods the same image block is processed more than once through floating point operations resulting in increased storage requirements. Due to differences in end application requirements, a direct comparison between JPEG-LS and JPEG 2000 hardware implementation examples should be made with caution. A commercial JPEG 2000 IP core for FPGA implementation uses 5 times more logic elements and needs 100 times more memory than a JPEG-LS one [10]. Although the first one is for video processing and the second is for raster scan images processing, image characteristics and rates are similar.

3. JPEG-LS ANALYSIS, IMPLEMENTATION AND RESULTS

3.1 Performance Analysis for Remote Sensing Images

C++ software implementations of JPEG-LS encoder and decoder were developed to evaluate compression performance and to explore the influence of restart intervals. Software tool resources

allow to select NEAR figure, from 0 to 9, and to break the image in independent blocks, by defining the number of lines per restart interval. In addition, the PSNR between the original and the recovered image after encoding/decoding processes is also evaluated. The test images were gathered by the CCD camera of China-Brazil Environment Resources Satellite (CBERS-2B). The four spectral bands, B1 - blue, B2 - green, B3 - red and B4 - near infrared, are available in rough data, 5812 x 5812 pixels image size, 8-bit pixel, TIFF files. To give an idea of the visual features, due to the compression and decompression processes, a detail of one of the images is shown in Figure 2.a together with its reconstructions.

Performance was evaluated by processing 20 different test images without restarting the compression process. The worse case lossless and near-lossless compression and respective PSNR results are listed in Table 1. The results for NEAR = 0 are in accordance with the expectations [6] and the mean near-lossless performance, for NEAR = 1, results in a compression better than 4:1 and a PSNR better than 50 dB. Eventual reductions in the compression rate can be handled by an output buffer at the risk of buffer overload. To overcome output buffer overload, one solution is to switch the NEAR as a parameter to increase or decrease compression rate along the process, as preview in JPEG-LS standard, part 2 [9].

Table 1. JPEG-LS Worse Case Compression for Different NEAR Values

		Compression Factor			PSNR (dB)		
NEAR		**0**	**1**	**2**	**0**	**1**	**2**
BAND	**B1**	2.3	3,7	4.7	loss less	51.4	45.9
	B2	2.3	3.7	4.7		51.4	45.9
	B3	2.3	3.5	4.5		51.4	46.2
	B4	2.3	3.6	4.6		51.4	46.1

Band B3 of Table 1 was chose to evaluate the effect of different restart interval extensions in the compression performance and the results are listed in Table 2. The small reduction in the compression factor, below 3% for a 24 line restart interval, is an indicative of the potential of this method to avoid error propagation.

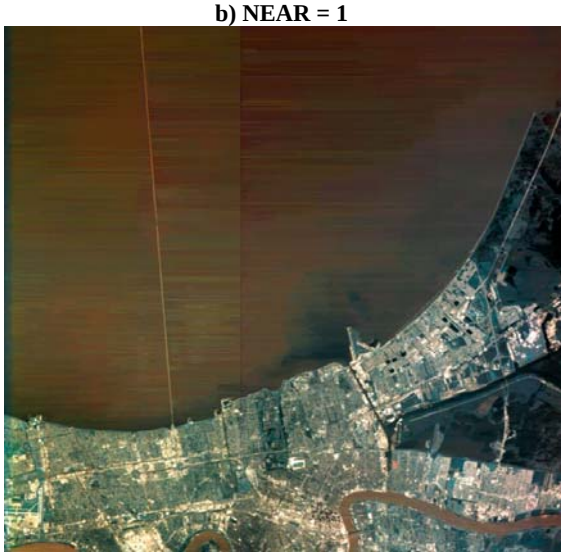
Table 2. JPEG-LS Performance for Different Restart Interval Extensions (B3 of Table 1)

		Compression Factor			PSNR (dB)		
NEAR		**0**	**1**	**2**	**0**	**1**	**2**
Lines per Restart Interval	**955**	2.28	3,54	4.48	loss less	51.3	46.2
	145	2.27	3.52	4.46		51.4	46.2
	24	2.23	3.45	4.37		51.4	46.2
	5	2.10	3.19	4.02		51.4	46.2

Part of the compression rate obtained must be spent in raising transmission error resilience by using EDAC. The use of a Reed-Solomon RS(255,223) EDAC would raise tolerance to burst errors with a 7% reduction in compression.

a) Original Image

b) NEAR = 1



c) NEAR = 2

Figure 2. Original and Reconstructed Images Detail.

3.2 Hardware Implementation Description

The hardware implementation of the JPEG-LS encoder in FPGA is based on a VHDL description of the algorithm. Referring to the architecture block diagram of Figure 3, the compression and encoding is achieved through two concurrent processes: the main process, where the core of the algorithm is executed; and the Golomb/Run Length (RL) encoder process. The reconstructed samples of one line are stored in the Line Buffer, to be used by the predictor in the next line. To save the time spent under CMV reinitialization and allow an immediate restarting of the compression process, two CMV arrays are employed and while one is being used by the compressor process, the other is reinitialized. Because of similarities between Golomb and RL encoding, the same process can be used with minor modifications. To adapt random throughput variations, main and encoding processes are interfaced through a FIFO. For test purposes, the Output Data Control process has two operation modes: one is saving the encoded data to the memory, and the other mode is checking encoded data against memory contents. By stopping the compression process, encoded data memory contents can be uploaded or downloaded.

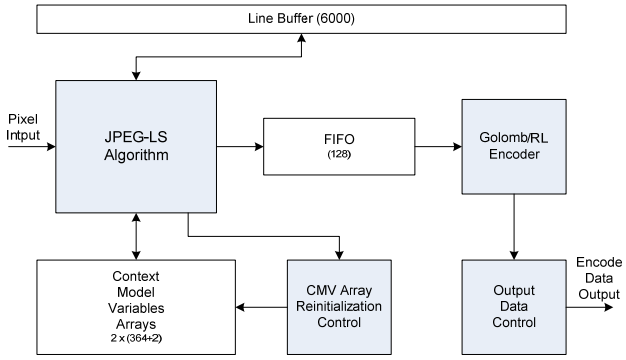


Figure 3. JPEG-LS VHDL Description Architecture.

The main process is a 16 state machine represented in the flowchart of Figure 4. Two compression levels are implemented: lossless (NEAR = 0) and near-lossless (NEAR = 1), selectable at VHDL compilation level. A resume of the distribution of the algorithm operations along Regular Mode states (steps) are listed in Table 3. The three Initial Context steps are executed at the beginning of each line. New pixel samples are gathered in steps Regular Mode 0 and Run Mode 2 and the reconstructed values are saved to the Line Buffer in Regular Mode 2 and Run Mode 1 steps. Part of the Run-length encoding is executed inside the Run Mode loop and Run-length < J residues are encoded in the Run Code steps. Run-length code information, including the J variable, is transferred to the Encoder Process through the FIFO, in place of the mapped error (*MErrval*) and Golomb coding (*k*) variables. Run interruption sample processing is distributed through steps Run Mode 1 (Context determination), Run Code 0 (Prediction evaluation), Regular Mode 2 and 3.

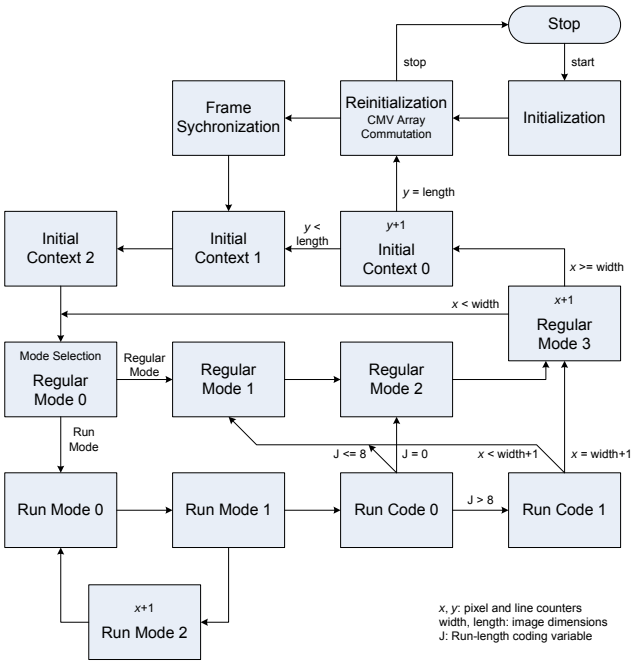


Figure 4. Main Process Flow Chart.

Table 3. Regular Mode Operation Distribution along Main Process Steps

Main Process Step	Algorithm Operation
Regular Mode 0	Gradient calculation Context determination Mode selection Prediction evaluation
Regular Mode 1	Prediction correction Golomb cod. variable computation (*k*)
Regular Mode 2	Error calculation Error quantization and reconstructed value evaluation (NEAR=1) Error mapping (*MErrval*) CMV update FIFO ← *k*, *MErrval*
Regular Mode 3	Bias computation Context update

3.3 Hardware Implementation Results

Memory requirements for 8-bit pixel and 6000 pixel width, two CMV arrays, and a 128 "word" FIFO are listed in Table 4. In general, memory blocks in FPGA are accessed through binary ports and the final requirement is about 50% higher than the total in Table 4. The Run loop is composed of three steps and, in general, run mode is faster than regular mode, in consequence, the throughput is limited by regular mode mean period, which is just above 4 steps. As one step corresponds to one state of the main state machine which in turn corresponds to one clock cycle, then a 12 Mpix/s rate is achievable with a 50 MHz clock.

To evaluate hardware implementation performance a Cyclone III EP3C25 FPGA, with 24,624 logic elements and 608,256 RAM bits, was employed. Real time operation is tested by using a high speed multi I/O board to feed image data to the FPGA board. The resulting encoded data is verified continuously in reference to previous loaded memory contents. The FPGA synthesized compressor is able to process one spectral band in blocks of 5812 x 90 pixels, 8-bit pixel, NEAR = 1, at a rate of 12 Mpix/s, with a 50 MHz clock frequency and using about 10% of device logic elements and 20% of device RAM bits.

Table 4. Memory Requirements

Function	Data	Extension	Total (kbits)
Line Buffer	8-bit pixel	6000	48
CMV Array	6-bit *N* 9-bit *B* and *C* 11-bit *A* 6-bit *Nn*	366 364 366 2	2 x 12.8
FIFO	4-bit *k* 8-bit *MErrval* 5-bit J	128 128 128	2.2
		Total	75.8

Referring to Table 5, a comparison with similar works has to be made with caution, due to the lack of a standard area measurement, different manufacturer's technology and incomplete implementation details. Both references [3, 4] have as main target the throughput, which is achieved by parallelizing process blocks, in detriment of area usage. The solution of reference [4], without near-lossless resources, uses about two times the proposed area (logic cells), and the solution of reference [3], with near-lossless resources, uses about 17 times the area (equivalent gates) of the ASIC implementation of reference [4]. Based on the Remote Sensing cameras MUX [11] and AWFI [12] under development by Brazilian National Institute for Space Research (INPE), which delivers image data at 5 and 3.8Mpix/s per spectral band, respectively, the proposed implementation fulfills preliminary throughput expectations and includes resources of restarting the process, to limit error propagation.

Table 5. Comparison against Other Results

Technology	Logic Area	Memory Usage (bits)	Operating Frequency (MHz)	Throughput (Mpixels/s)
Cyclone III EP3C25 (present work)	2,375 (10%) log. cells	13 x 9k (20%)	50	12
Virtex-E XCV 1600E [4]*	4,929 (15%) log. cells	8 x 4k (6%)	66.9	66.9
ASIC 0.18 um [4]**	27,681 eq. gates	23,887	183	183
Virtex-II Pro XC2VP30 [3]***	473,862 eq. gates	50,655	90	90

* Lossless only implementation, 640 or 1024 pixels per image line (not clearly specified).
** Lossless only implementation, 1024 pixels per image line.
*** Lossless and near-lossless implementation, 1024 pixels per image line.

The successful implementation results, combined with the algorithm analysis, indicate the potential offered by the proposed solution for onboard data compression and establishes the basis on which the next steps of exploration, in the sense of a functional and reliable end item hardware, will be carried on.

4. CONCLUSION

JPEG-LS relative efficiency in terms of compression rate, loss, complexity and memory requirements, is very attractive to hardware implementation purposes. Depending on application needs, compressions better than 2:1, in lossless mode, or average compressions exceeding 4:1 with a PSNR better than 50 dB, can be achieved. Comparing to other methods [6] JPEG-LS is an excellent choice for lossless applications. Under lossy applications approach, the performance of modern DWT transform based methods is superior but, under hardware requirements point of view, their complexity is certainly a major drawback.

To improve fault-tolerance, mitigation techniques shall be designed, implemented and evaluated [2]. In consequence, more FPGA resources will be necessary, and that is why basic project area savings are important. Additional work shall be carried on packetization process in which spectral band data are mixed in formatted frames with addition of header, EDAC and stuffing data to maintain a constant output data rate [1].

The main problem in the development of space qualified hardware for remote sensing image data compression is not the compression method itself but the hardware in which the data will be processed. The use of low complexity solution is a range spreading factor in terms of device options, as space qualified ASICs or FPGAs, or even commercial off-the-shelf (COTS) FPGAs, which can be strategically decisive in the near future.

5. ACKNOWLEDGMENTS

Our thanks to INPE's DPI fellows, in special to Jeferson Arcanjo, for the test image files.

6. REFERENCES

[1] Almeida, G.M., Bezerra, E.A., Cargnini, L.V., Fagundes, R.D.R., and Mesquita, D.G. 2007. A Reed-Solomon algorithm for FPGA area optimization in space applications. *Proceedings of the 2nd NASA/ESA Conference on Adaptive Hardware and Systems* (Edinburgh, Scotland, Aug. 2007), 243-249. DOI= http://doi.ieeecomputersociety.org/10.1109/AHS.2007.17.

[2] Kastensmidt, F.L., Neuberger, G., Carro, L., and Reis, R. 2004. Designing and Testing Fault-Tolerant Techniques for SRAM-based FPGAs. In *Proceedings of the 1st Conference on Computing Frontiers* (Ischia, Italy, Apr. 2004), 419-432. DOI= http://doi.acm.org/10.1145/977091.977150.

[3] Lei, J., Li, Y., Kong, F., and Wu, C. 2008. A New Pipelined VLSI Architecture for JPEG-LS Compression Algoritm. In *Proceedings of SPIE* 7084 (2008). DOI= http://dx.doi.org/10.1117/12.794543.

[4] Papadonikolakis, M.E., Kakarountas, A.P., Goutis, C.E. 2008. Efficient high-performance implementation of JPEG-LS encoder. *J. Real-Time Image Proc.* 3 (Dec. 2008), 303-310: http://www.springerlink.com/content/e25538m6m8kv7472/

[5] Weinberger, M.J., Seroussi, G., and Sapiro, G. 2000. The LOCO-I Lossless Image Compression Algorithm: Principles and Standardization into JPEG-LS. *IEEE Trans. Image Processing* 9 (Aug. 2000), 1309-1324. DOI= http://dx.doi.org/10.1109/83.855427

[6] Yu, G., Vladimirova, T., and Sweeting, M.N. 2009. Image compression systems on board satellites. *Acta Astronautica*. 64 (May-Jun. 2009), 988-1005. DOI= http://dx.doi.org/10.1016/j.actaastro.2008.12.006.

[7] ITU T.81. 1992. *Digital Compression and Coding of Continuous-Tone Images - Requirements and Guidelines*. ITU CCITT recommendation (Sep. 1992): http://www.itu.int/rec/T-REC-T.81-199209-I/en.

[8] ISO/IEC 14495-1:1999. Information technology -- Lossless and near-lossless coding of continuous tone still images: Baseline. *JPEG-LS standard part-1* (Dec. 1999): http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=22397.

[9] ISO/IEC 14495-2:2003. Information technology -- Lossless and near-lossless coding of continuous tone still images: Extensions. *JPEG-LS standard part-2* (Apr. 2003): http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=37700.

[10] CAST. *JPEG 2000 and Lossless JPEG IP Core Specifications.* Retrieved June 4, 2010, from IP Provider CAST, Inc.: http://www.cast-inc.com/ip-cores/index.html.

[11] RBN-HDS-0014/03. 2006. *CBERS 3&4 Multispectral Camera (MUX) Subsystem Specification*. Instituto Nacional de Pesquisas Espaciais (INPE) (S.J. dos Campos, Apr. 2006).

[12] A823100-SPC-01 Rev 01. 2008. *Advanced Wide Field Imaging Camera (AWFI) Subsystem Specification*. Instituto Nacional de Pesquisas Espaciais (INPE) (S.J. dos Campos, Aug. 2008).