HYPERSPECTRAL MONITORING DATA PROCESSING

Sergio Montenegro, P. Behr [a], Igor Rodionov, Alexey Rodionov, Evgeniy Fedounin [b]

[a] FhG FIRST, Kekulestr 7, 12489 Berlin, Germany.
www.first.fhg.de/~sergio, www.first.fhg.de/~peter

[b] Reagent Research & Development Ctr, P.B. 192, A-190, Moscow 125190 Russia.

ABSTRACT

Nowadays exit a very high demand for hyperspectral monitoring from space for example of production, pipelines and from unknown regions. This helps to find materials and anomalies on ground. There are two main problems to be solved. First compact and sensitive detectors, and second the real time processing of the enormous data rates produced by the detectors. This problems are presented in two different papers in this conference. The detectors are presented in " Mono-Photon Technology Based Hyperspectral Systems for Remote Sensing in Russia ". The second Paper (this), describes the concept of the real time data processing of 10^10 Bytes per second of hyperspectral data using COTS technology in space.

1. INTRODUCTION

This hyperspectral camera is based on acousto-optic programmable filters, which are used for spectral selection, and set of time-coordinate sensitivity detectors to register single photons with high time-spatial resolution. This set of detectors produces 10^10 Bytes/Second which have to be processed in real time. A measurement needs 100 Seconds of integration time and produces an 1000x1000x1000 hyperspectral cube which will be proceeded by the super computing kernel.

Currently the best hyperspectral detectors (MAMA-detector) archive about 1000x1000 pixel of space resolution, 1000 spectral points and the photo-electron counting rate of 10^6 s^{-1}. «Reagent» R&D center developed an hyperspectral detector variant, which have the photo-electron counting rate greater then 10^8 s^{-1}.

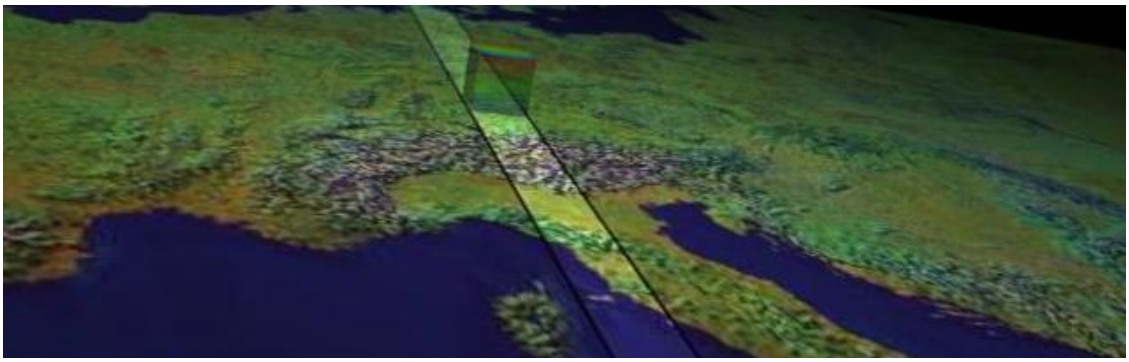

Figure 1: Scan of an hyperspectral cube

2. SUPER COMPUTING KERNEL

It requires a very high computing power to reduce the 10^10 Bytes/Sec of raw data from the Time-coordinate sensitive detectors (TCSD) to an hyperspectral cube of 1000x10000 (x,y) times 1000 points on wavelength each 100 seconds and then to classify this cube to get useful materials data with a rate of 2 Mbytes/second .

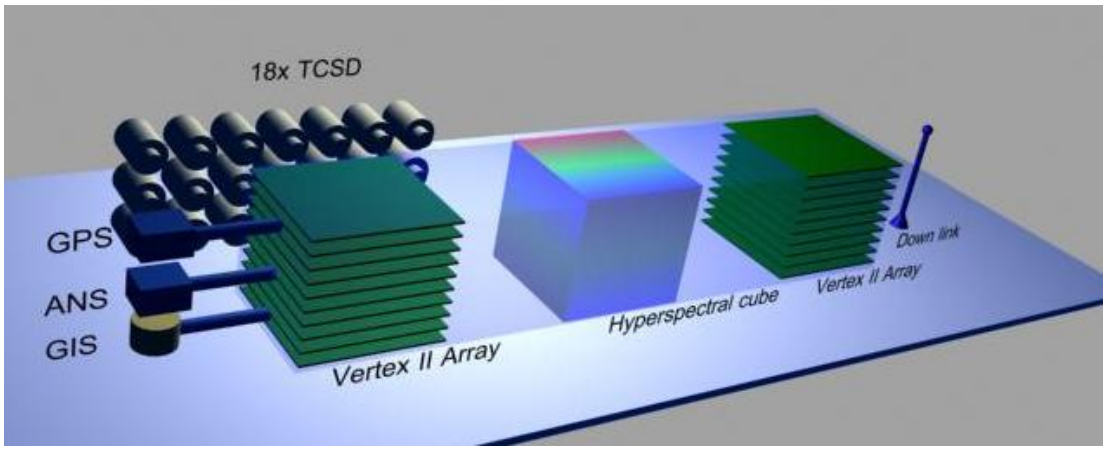

Figure 2: Detectors and Super Computing Kernel

The on-board supercomputer has to provide a performance of 3x10^10 operations/second in space! Currently there is no space computer with such a performance. We will reach this limit using an parallel array of specialized computing nodes.

Most of the required operations can be performed in a vector fashion and are the same for all pixels. For this operations the super computing kernel will implement a programmable systolic array with several parallel vector pipelines with an refined interconnection network. For control and scalar operations the super computing kernel implements an array of high performance PowerPC CPU which are connected to the interconnection network. This solution provides a very high flexibility and performance. Both data paths (vector and scalar/control) use parallelism and high frequency to get the highest possible performance.

3. HARDWARE

The super computing kernel is implemented using an array of nodes – Vertex-II-Pro chips - in a fixed, but high performance and flexible interconnection network. Inside of each node there is a PowerPC CPU (PPC) and an array of programmable logic cells

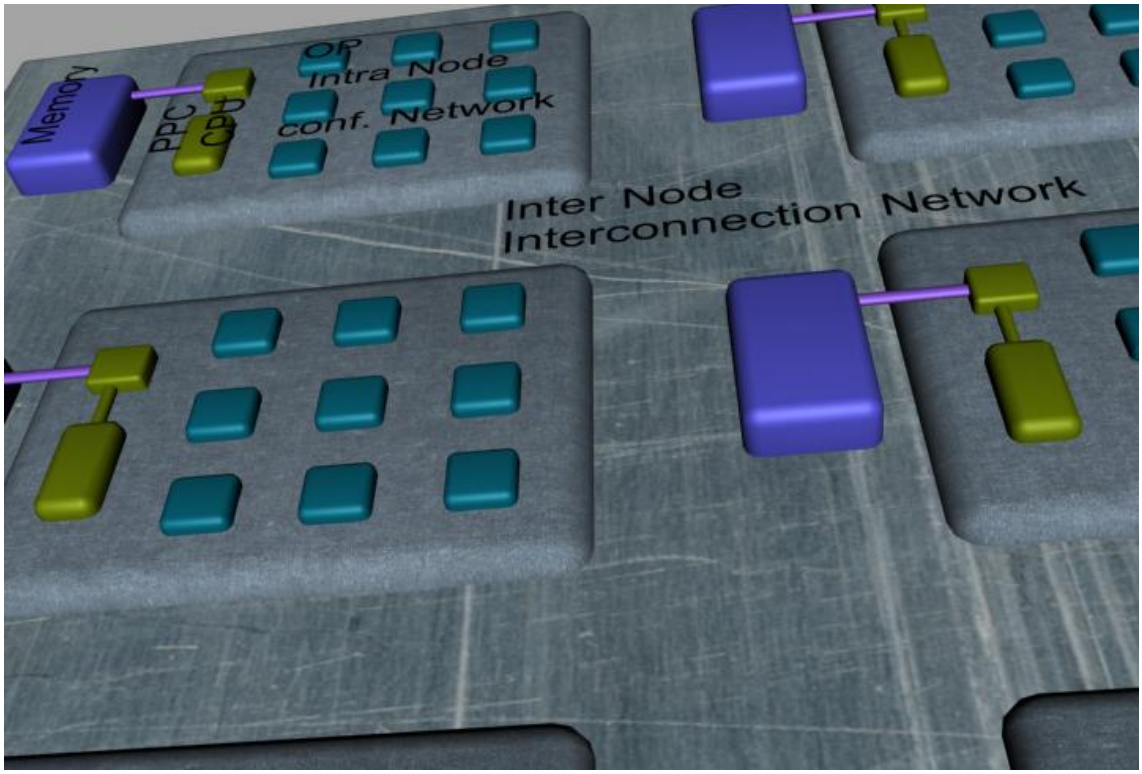

Figure 3: Array of Nodes (Vertex-Pro)

The PPC loads, configures and programs the logic cells to execute any required vector algorithm. During the execution time of the vector algorithms, the PPC controls the data flow and intermediate storage of data being processed, and it controls the interconnection network between nodes to transfer data packages as required by the algorithm. In a few words the Software in the PPCs performs the *Information Logistic* in the system.

4. SUPER COMPUTING KERNEL SOFTWARE

Before the execution starts, the user transmits (uploads) the appropriated control software and the required systolic configuration. Then the PPC will configure the sea of gates and execute the control software. During execution the super computing kernel looks like a huge systolic array with distributed real time control.

The *information logistics* - the control of the data flow throw the systolic array in each node and the packages transfer between nodes, have very tight real time requirements and need some means for communication between nodes. To achieve this function we use our micro kernel real time operating system BOSS.

BOSS provides an interface which can be extended to implement your applications, and a very simple (but effective) communication between applications in the same node. Using the BOSS communication primitives the user can implement software buses to

handle the intra node communication. In the same way like communicating with other application, each application can communicate with input/output drivers to access devices.

Figure 4: BOSS Applications and Software Buses

To be able to communicate with other applications on other nodes we implement software routers attached to the software buses. The routers take messages to applications in other nodes, build communication packages to encapsulate the messages and send the packages throw the inter-node interconnection network. On the other node another router will take the package, extract the message and send it to the target application. The applications do not need to take care about this routings.

5. BOSS: A DEPENDABLE OPEN SOURCE REAL TIME OS

BOSS targets a principle which the world forgot a long time ago: *Simplicity*. It was designed to be a dependable real time embedded operating system which can be easily certified. Due to the fact, that complexity is the first foe of safety, BOSS is intended to be as simple as possible, so it is easier to understand, to review, to use etc. Some parts of BOSS are being verified mathematically and formally using model checker and theorem proofers. With the current state of the art on formal verification, complex systems cannot be verified formally, but BOSS can be. BOSS is based on very few and simple basic functions, which can be proofed very faithfully, and these functions are used for almost every operation of the kernel. Furthermore BOSS is open-source, so that everybody can look at it and find possible errors.
Its characteristics are: multithreading , pre-emptive , priorities managements , real time, fault tolerance support , communication support , OO-design and implementation , C++ interface , Time resolution 1 microsecond , Thread switch time 3 microseconds PPC at 48 Mhz, Reaction time: under 3 microseconds PPC at 48 MHz