Sensory Stream Data Mining on Chip

Yang Cai, Carnegie Mellon University, ycai@cmu.edu
Yong X. Hu, NASA Langley Research Center, yongxiang.hu-1@nasa.gov

Abstract

Mining physical properties from real-time sensor stream data is important to the atmospheric studies, ecology and oceanography. An FPGA-based reconfigurable sensory stream data mining processor is presented in this paper. The processor is based on Generalized Non-Linear Regression algorithm and trained with radiative transfer simulations and observations for autonomous detection of satellite measurement signatures and retrievals of atmospheric physical properties. The results show that the embedded computing approach is faster than traditional computing methods in orders of magnitude. The data mining processor is able to automatically adapt to multi-platform sensors, eliminate redundant algorithm development and provide a vehicle for autonomous onboard image analysis and physics-based data compressions.

1 Introduction

Sensory data fusion and analysis from multiple satellite platforms have been increasingly important tasks in Atmospheric science research. Unfortunately, seamless multi-platform data analysis tools currently do not exist because of difficulties in algorithm development and huge cost. For example, developing an ensemble of operational algorithms for a single instrument, such as CERES and MODIS, took about a decade and cost millions. As a result, these Earth observing missions have not fully achieved their capabilities. With the growing number of satellites and improving spatial and spectral resolutions, users have been coping with massive data on daily basis.

The objective of this study is to build an embedded computing processor for onboard physical property retrieving, which would significantly reduce the cost of physical inversion for combined observations. With this tool, scientists can save their time from repeatedly learning and developing complicated mathematical and computational models, such as the Rodgers iterative methods [7].

2 Sensory Data Mining

Given physical properties \([\mathbf{P}]\) of the atmosphere, most multi-spectral observations \([\mathbf{O}]\) can be simulated \([\mathbf{O} = f(\mathbf{P})]\), thanks to known physics models (also called “forward models”). Physical-inversion is the process of retrieving physical properties from observations \([\mathbf{P} = f^{-1}(\mathbf{O})]\). Physical inversion is a key component of an Earth observing mission. For example, near sea surface wind-speed (W) can be retrieved from the TRMM Microwave Imager (TMI) brightness temperatures (\(\mathbf{B}\)). For given wind speed, brightness temperatures can be estimated from physics models \([\mathbf{B} = f(W)]\). The simplest inversion to retrieve wind speed \(W\) from observation \(\mathbf{B} \ [W = f^{-1}(\mathbf{B})]\) is a linear regression. For given physical properties, we will generate a huge library of high spectral resolution radiative transfer calculations for UV, visible, near IR, thermal IR and microwave wavelengths. To create the database, we would use radiative transfer models such as MODTRAN for gas absorption and DISORT for multiple scattering.
Most remote sensing related inverse problems are \textit{nonlinear} in nature. A generalized nonlinear regression (GNR) method is used specifically for high-performance computing implementation. Using GNR, the wind speed \( W \) can be derived from microwave measurements \( \mathbf{B} \) as:

\[
W(\bar{B}) = \frac{\sum_{i=1}^{N} \hat{W}_i D_i}{\sum_{i=1}^{N} D_i}
\]

\[
D_i = \exp\left[ -\frac{1}{2} \sum_{j=1}^{N} \left( \frac{\hat{B}_{ji} - \hat{B}_j}{\rho \sigma_i} \right)^2 \right]
\]

where \( \hat{W}_i \) and \( \hat{B}_{ji} \) are wind speed and microwave brightness temperatures from forward model simulations and previous retrieval results. \( \rho \) is a correlation factor between the brightness temperature of channel \( j \) (\( \hat{B}_j \)) and the wind speed, and \( \sigma_i \) is measurement error of \( \hat{B}_j \). Based on radial-bases neural networks, GNR is a non-parametric estimation method. Retrievals using GNR are as straightforward as linear regressions but yield more accurate results. GNR has been tested on other remote sensing applications as well [9]. With all the existing physics models, we can produce equivalent \( \hat{W}_i \) and \( \hat{B}_{ji} \) for all instruments from forward simulation. Comparing with other inverse methods, GNR is more universal since it does not require \textit{a priori} information. However, GNR is not necessary an ideal candidate for high performance parallel computations. To improve the parallelism of the algorithm, we modify the GNR to its subset, the Radial Basis Function (RBF) model, which is simpler and easy to parallelize.

\[
W(\bar{B}) = W_0 + \sum_{i=1}^{k} W_i D_i
\]

\[
D_i = \exp\left[ \frac{\text{dist}(\hat{B}_i, \bar{B})^2}{2\sigma_i^2} \right]
\]

Where each \( \hat{B}_i \) is a kernel center and where \( \text{dist}(\cdot) \) is a Euclidean distance calculation. The kernel function \( D_i \) is defined so that it decreases as the distance between \( B_u \) and \( B_{i,i} \) increases. Here \( k \) is a user defined constant that specifies the number of kernel functions to be included. The Gaussian function \( D_i \) is centered at the point \( \hat{B}_i \) with some variance \( \sigma_i^2 \). The function provides a global approximation to the target function, represented by a linear combination of many local kernel functions. The value for any given kernel function is non-negligible only when the input \( \bar{B} \) falls into the region defined by its particular center and width. Thus, the network can be viewed as a smooth linear combination of many local approximations to the target function. The key advantage of RBF networks is that they contain only summation of kernel functions rather than compounded calculation so that RBF networks are easier to be parallelized. In addition, they can be trained much more efficiently with genetic algorithms.

3 System Architecture

A prototype of the data-mining-on-chip was constructed on the NI PXI-7831R FPGA prototyping board. The FPGA Vertex II 1000 contains 11,520 logic cells, 720 Kbits Block RAM, and 40 embedded 18x18 multipliers. This board is primarily designed for implementation of custom electronic logic probes. It can be programmed using the tools provided by National Instruments. These tools convert LabVIEW VI block-diagram code into VHDL, which is then compiled to a layout for the FPGA using the standard Xilinx toolset. With this system, rapid prototyping of a solution and alteration of the number of parallel channels can be done without significant development time. In our design, the computation intensive portion of the multi-spectral image
classification algorithm resides on the calculations within the processor. The user interface, data storage and I/O, and adaptive coprocessor initialization and operation are performed on the host computer. We have implemented the parallelism algorithms at two levels: First, on-chip parallelism, where we use Labview™ “Parallel While Loop” to execute the parallel modules. Second, we put multiple FPGA boards on the PXI bus where they can run in parallel.

![Basic design for GNR implementation on FPGA](image)

### Fig. 1 Basic design for GNR implementation on FPGA

#### 4 Results

We have the following preliminary results through our benchmark experiments: 1) we found that FPGA over-performance Pentium at least two to three orders of magnitude in terms of speed. For example, for RG model, the FPGA uses 39 ns (with 10 MHz clock speed). Pentium uses between 1000 ns and 2000 ns (with 1 GHz clock speed). 2) we also found that the fixed point Radial Basis Function algorithm demonstrated ideal parallelism as the number of simultaneous basis compares increases.

![Computing Time (left, 2 Inputs and 100 Basis) and Resources Utilization (right)](image)

**Fig. 2** Computing Time (left, 2 Inputs and 100 Basis) and Resources Utilization (right)

Table 1 shows a comparison of the processing time required to compute the value of one set of inputs $\overline{B}$. While significant advantage is seen in the addition of an extra iterating loop and the unrolling of the dimension calculation loop into two parallel channels, no performance improvement is seen with the further addition of parallel channels to calculate $D_j$. This is likely because the Memory Extension Utility
creates modules that are shared, and accessor controls are used to regulate entry into the modules from multiple parallel data sources. The initialization of more independent memory modules may further improve performance at the cost of more RAM used and more redundant copies of the lookup tables in memory. One limitation in the design of the GNR equation was the number of multiplier units available on the Xilinx FPGA. On the Xilinx FPGA, multiplication is executed by the use of a pre-constructed 18x18 MULT unit on the chip; This unit is useful because multiplication is an expensive operation in terms of number of gates used. However, with only forty of these units on the chip the amount of multiplication allowed in the GNR calculation was extremely limited. Potential applications of this generalized FPGA-based neural network processor include: multi-platform combined retrievals; universal application of atmospheric remote sensing; real-time data analysis and atmospheric corrections for ocean and land image processing; improvements in data processing capabilities of Atmospheric Science Data Centers (ASDC). An example of accelerating atmospheric retrievals in ASDC: currently ASDC host computer has all the input “neurons”. The I/O of generalized inversion FPGA processor would be linked to ASDC using bi-directional “shared memory”.

Table 1: Resource utilization and processing time for the GNR equation with multiple parallel loops and sub-loops

<table>
<thead>
<tr>
<th># of loops</th>
<th># of sub-loops</th>
<th>Slices</th>
<th>Slice Flip</th>
<th>4 input LUTs</th>
<th>MULT 18x18s</th>
<th>Equivalent gate count</th>
<th>Calculation time (ticks)</th>
<th>calc time (microsecs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>917</td>
<td>1217</td>
<td>1282</td>
<td>2</td>
<td>420053</td>
<td>4500</td>
<td>112.5</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1013</td>
<td>1347</td>
<td>1316</td>
<td>3</td>
<td>425510</td>
<td>2949</td>
<td>73.725</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3074</td>
<td>4183</td>
<td>4001</td>
<td>6</td>
<td>477586</td>
<td>1715</td>
<td>42.875</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2473</td>
<td>3519</td>
<td>2948</td>
<td>12</td>
<td>492296</td>
<td>1569</td>
<td>39.225</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>2936</td>
<td>4211</td>
<td>3457</td>
<td>15</td>
<td>514038</td>
<td>1579</td>
<td>39.475</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>5118</td>
<td>7626</td>
<td>6045</td>
<td>30</td>
<td>622895</td>
<td>1578</td>
<td>39.45</td>
</tr>
<tr>
<td>Maximum on FPGA</td>
<td></td>
<td>5120</td>
<td>10240</td>
<td>10240</td>
<td>40</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5 Conclusions

First, we found that the data mining processor is able to automatically adapt to multi-platform sensors, eliminate redundant algorithm development and provide a vehicle for autonomous onboard image analysis and physics-based data compressions. Second, we found that FPGA over-performance Pentium at least two to three orders of magnitude in terms of speed. Third, we found that the fixed point Radial Basis Function algorithm demonstrated ideal parallelism as the number of simultaneous basis compares increases. Finally, we found the bottlenecks of the FPGA-based computing is the data I/O. How to get data in and out of the FPGA is on a critical path in terms of speed. Currently we use PXI bus that contains only 16-bit parallel channels. It can be saturated when the data exchanges are frequent. For high-speed data exchange, we plan to build our own board with a matrix of inter-connected FPGAs and use the available data I/O lines on FPGA chips for inter-chip data communication. In addition, from the implementation experience of FPGA, we have realized the importance of FPGA-oriented design. To make things work right and efficiently, we would redesign algorithms that are easy to be parallelized and easy to be implemented at the gate level. It may create a new direction of computational algorithm design.
Acknowledgement

We would like to thank Jim, Chuck and Newton of National Instruments for his comments and supports. The work was sponsored in part by NASA grants from ESTO-AIST and Creativity and Innovation Initiative.

References

15. Xilinx Products, 2003