Implementing Legacy-C Algorithms in FPGA Co-Processors for Performance Accelerated Smart Payloads

Paula J. Pingree, Lucas J. Scharenbroich*, Thomas A. Werne**
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, California 91109-8099
818-354-0587, 818-354-5322*, 818-354-3008**
Paula.J.Pingree@jpl.nasa.gov, LucasJ.Scharenbroich@jpl.nasa.gov*,
ThomasA.Werne@jpl.nasa.gov**

Christine Hartzell
Georgia Institute of Technology
Atlanta, GA 30332
Gtg733w@mail.gatech.edu

Abstract—Accurate, on-board classification of instrument data is used to increase science return by autonomously identifying regions of interest for priority transmission or generating summary products to conserve transmission bandwidth. Due to on-board processing constraints, such classification has been limited to using the simplest functions on a small subset of the full instrument data. FPGA co-processor designs for SVM[1] classifiers will lead to significant improvement in on-board classification capability and accuracy.

We implemented a SWIL[2] classifier, developed for the Hyperion instrument on the EO-1 spacecraft, on the Xilinx Virtex-4FX60 FPGA as a baseline challenge. We have taken advantage of Impulse C™, the commercially available C-to-HDL tool by Impulse Accelerated Technologies, which supports the development of highly parallel, co-designed hardware algorithms (from software) and applications. This paper describes our approach for implementing the Hyperion linear SVM on the Virtex-4FX FPGA, as well as additional experiments with increased numbers of data bands and a more sophisticated SVM kernel to show the potential for better on-board classification achieved with embedded FPGAs over current in-flight capabilities.[3,4]

[1] SVM = Support Vector Machine
[2] SWIL = Sea, Water, Ice & Land
[3] 1-4244-1488-1/08/$25.00 ©2008 IEEE
[4] IEEEAC paper#1230, Version 2, Updated 2007:12:13

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Smart Payload Motivation

On board computation has become a bottleneck for advanced science instrument and engineering capabilities. Currently available spacecraft processors have high power consumption, are expensive, require additional interface boards, and are limited in their computational capabilities. Recently developed hybrid field-programmable gate arrays (FPGAs), such as the Xilinx Virtex-4FX [1], offer the versatility of running diverse software applications on embedded processors while at the same time taking advantage of reconfigurable hardware resources all on the same chip package. These tightly coupled hardware/software co-designed systems are lower power and lower cost than general-purpose single-board computers (SBCs) [2], and promise breakthrough performance over radiation-hardened SBCs, leading to a new architecture for Smart Payload development (Table I).

Computational Platform	Performance (DMIPS)
RAD750 SBC	240
Xilinx Virtex-II Pro	450
Xilinx Virtex-4	680

TABLE I. PERFORMANCE: SBC VS. EMBEDDED FPGAS

Designs based on embedded FPGA processors also benefit from the following advantages over SBCs:

- Higher level of reuse
- Reduced risk of obsolescence
- Simplified modification and update
- Increased implementation options through modularization

We have selected the Xilinx ML410 evaluation platform (Figure 1) for development and demonstration of selected Smart Payload concepts including the SVM implementation on the Virtex-4FX FPGA.

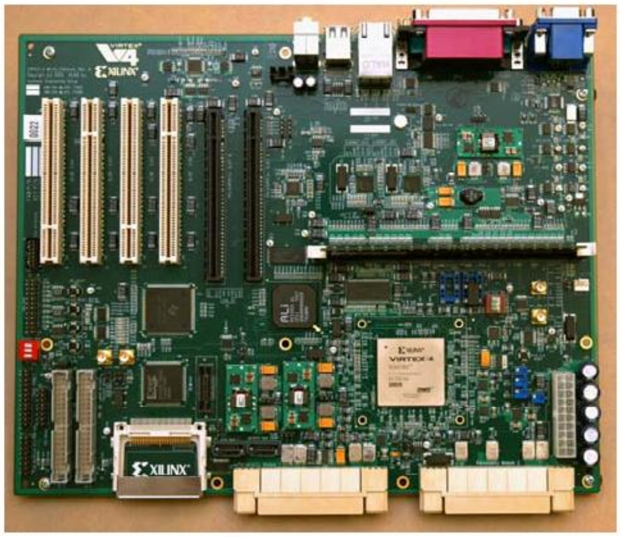


Figure 1. A good candidate for the development of a future instrument computer for space. The Xilinx ML410 evaluation board comes with the V4FX60 FPGA that features two embedded PowerPC405 processors [3].

1.2 SVMs for Hyperspectral Classification

Support Vector Machines [4] have found broad application in general machine learning and classification tasks as well as onboard remote sensing [5]. A SVM is a *maximum margin* classifier that finds a separating hyperplane between two labeled classes such that the distance to the nearest datum in each class is maximized (Figure 2). By selecting such a maximum margin hyperplane, the SVM classifier can exhibit better generalization to new data than other linear classification methods.

The goal of training a support vector machine is to learn a set of weights such that the sign of a weighted sum of dot products between the training data, x_i, and a test vector, t, will correctly predict the class of the new data vector.

$$y = \operatorname{sgn}\left(\sum_i w_i \langle x_i, t \rangle\right) \quad y \in [-1,+1]$$

SVMs also incorporate the *kernel trick* [6], which allows them to be extended from purely linear to non-linear classifiers. This trick is accomplished by formulating the training and testing algorithms in terms of dot products, <x,y>, and then replacing the dot products with a *kernel function*, K(x,y) = <□(x),□(y)>, that represents a dot product after passing the arguments through some non-linear function, □. By cleverly constructing the kernel function, the high-dimensional dot product can be computed efficiently.

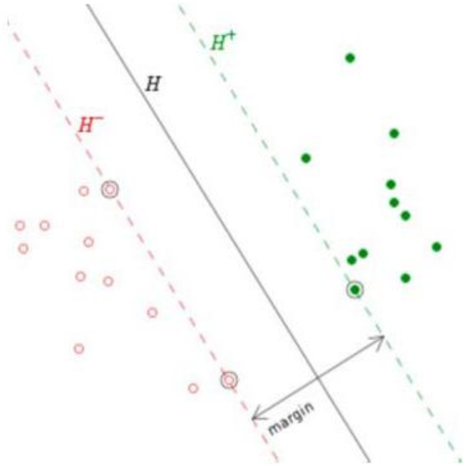


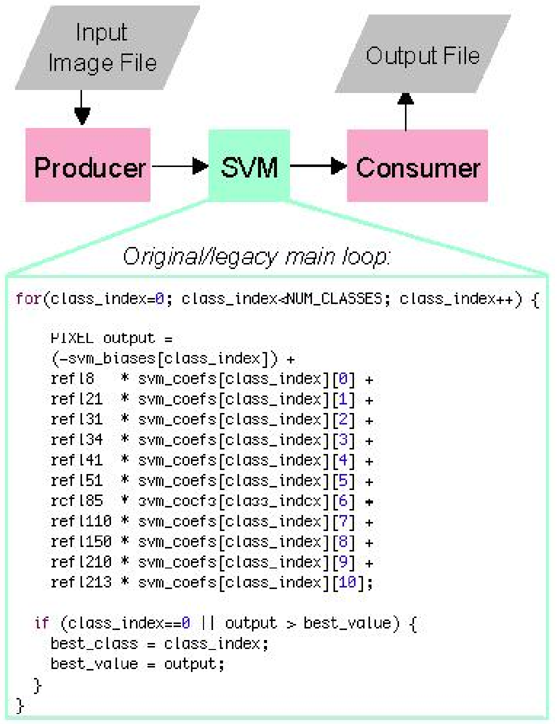
Figure 2: Maximum margin separating hyperplane between two data classes. The circled data points are the support vectors that lie on the margin.

SVMs are well suited to onboard autonomy applications. They represent a state-of-the-practice method in machine learning and have a history of reasonable performance across many domains. The property that makes SVMs particularly applicable is the asymmetry of computational effort in the training and testing stages of the algorithm. Classifying new data points requires orders of magnitude less computation than training because the process of training a SVM requires solving a quadratic optimization problem. This naively requires on the order of $O(n^3)$

operations, where n is the number of training examples. Faster algorithms that exploit the specific structure of the SVM optimization problem have been developed [7], but the training remains the primary computational bottleneck.

After a SVM is trained, many of the weights, w_i, will be equal to zero. This means that these terms can be ignored in the classification formula. The input vectors that have a corresponding non-zero weight are called *support vectors*. Even more computational savings can be realized in the case of using a linear kernel function. The weighted sum over the kernel function is associative, so all the support vectors can be collapsed into a single vector with a single weight.

Reducing the number of support vectors is key to successfully deploying a SVM classifier onboard a spacecraft where there are severe constraints on the amount of CPU resources available. Previously deployed classifiers [5] have used such reduced-set methods, but were still constrained to operate on only a subset of the available classification features. Removing such bottlenecks is critical to realizing the full potential of SVMs as an onboard autonomy tool.



```
for(class_index=0; class_index<NUM_CLASSES; class_index++) {

    PIXEL output =
    (-svm_biases[class_index]) +
    refl8   * svm_coefs[class_index][0] +
    refl21  * svm_coefs[class_index][1] +
    refl31  * svm_coefs[class_index][2] +
    refl34  * svm_coefs[class_index][3] +
    refl41  * svm_coefs[class_index][4] +
    refl51  * svm_coefs[class_index][5] +
    refl85  * svm_coefs[class_index][6] +
    refl110 * svm_coefs[class_index][7] +
    refl150 * svm_coefs[class_index][8] +
    refl210 * svm_coefs[class_index][9] +
    refl213 * svm_coefs[class_index][10];

  if (class_index==0 || output > best_value) {
    best_class = class_index;
    best_value = output;
  }
}
```

Figure 3. FPGA Co-design for the SVM Algorithm

2 SVM DEVELOPMENT FOR V4FX FPGA

JPL has developed SVM classification algorithms that can be used onboard spacecraft to identify high priority data for downlink to Earth and to provide onboard data analysis to enable rapid reaction to dynamic events. To meet NASA's science objectives these classifiers detect flooding, volcanic eruptions and sea ice break-up. Current pixel-based machine learning and instrument autonomy algorithms that have successfully detected and identified various natural phenomena are flying on computational technologies such as the RAD6000 and Mongoose V processors that have limited computing power, extremely limited active storage capabilities and are no longer considered state-of-the-art. To date, such on-board classification has been limited to using the simplest function, a linear kernel, on only a subset of the full instrument data (11 of 242 bands for Hyperion on EO-1).

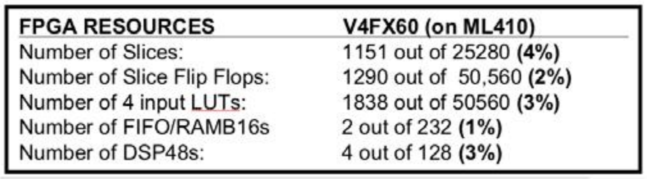
We have implemented, on the Virtex-4FX60, a linear SVM classification algorithm. This migration to a low power, high-speed FPGA computing platform adds flexibility and scalability to the system. For the FPGA-based development of the SVM, the previously software-only legacy algorithm is implemented in the FPGA hardware fabric to take advantage of high-speed parallel processing capabilities while the image file input and classification file output is managed within the embedded PowerPC processor. Figure 3 illustrates the partitioned system.

The Producer is coded in a file called sw.c. It reads an input image file containing 857,856 pixels and streams data to the SVM. The Consumer, also coded in sw.c, streams data from the SVM and writes pixel classifications (e.g., snow, water, ice, land, cloud, or unclassified) to an output file. The original legacy SVM code is put in a file called hw.c. The SVM algorithm in hw.c was transformed from C-to-HDL using the Impulse C tool set by *Impulse* [8] and simulated to validate execution of the co-designed system. The sw.c program will execute on the V4FX embedded PowerPC processor and communicate with the hardware-accelerated algorithm in the FPGA fabric.

Next the converted HDL code was synthesized for the Virtex-4FX FPGA using the Xilinx ISE & EDK development environment. Synthesis determined the V4FX resource output for the SVM algorithm to be:

- 1 Adders/Subtractors (6 bit)
- 3 Adder/Subtractor (32 bit)
- 1 Multipliers (32 bit)
- 5 Comparators (32 bit)
- 2 Floating Point Adders/Subtractors (32 bit)
- 1 Floating Point Multipliers (32 bit)

This translates to the following resource utilization for the V4FX60 device on the ML410 development platform (Table II).



FPGA RESOURCES	V4FX60 (on ML410)
Number of Slices:	1151 out of 25280 **(4%)**
Number of Slice Flip Flops:	1290 out of 50,560 **(2%)**
Number of 4 input LUTs:	1838 out of 50560 **(3%)**
Number of FIFO/RAMB16s	2 out of 232 **(1%)**
Number of DSP48s:	4 out of 128 **(3%)**

TABLE II. *IMPULSE C* RESOURCE REPORT FOR SVM.

The results of the simulation effort were presented at the 2007 NASA Space Technology Conference [9] as a preliminary report of this on-going task.

2.1 Validation

The output of the Producer-SVM-Consumer path is a file composed of a column of integers indicating the resulting class of each pixel in the image. This output file is then reformatted in Matlab^{TM} to the original pixel-wise dimensions of the image. Additionally, each class is assigned an arbitrary color and the number of pixels belonging to each class is tabulated. We can then easily calculate the percentage of pixels belonging to each class and visualize the resulting file of classified pixels.

Validation was required in two facets of this project. It was necessary to validate both the pixel classification results from the SVM and the Impulse C implementation of the SVM. We began the classification process by comparing the pixel classification percentage results to those achieved on the SVM used in the ASE on the Earth Observing-1 Satellite. The classification percentages show good agreement, particularly for the snow and water classes (Table III).

It is possible, however, for the raw percentage results to look reasonable, while the pixel classification visualization shows no resemblance to the physical features in the image. The visualizations were integral in our validation efforts. Our resulting visualizations show excellent agreement with the results from the ASE SVM (Figure 4). In addition to the qualitative comparison of the images, we also conducted a pixel-by-pixel comparison of the ASE results and our classifications. This comparison was made less accurate due to our lack of a raw classification data file for the ASE image. The pixel-by-pixel classification comparison showed that 76.8% of the pixel classifications in our results matched those of the ASE results (Figure 5 & Table III). We believe the discrepancies to be due to the differences in the training datasets of the SVMs.

In order to dismiss the possibility of errors being introduced by the Impulse C implementation of the SVM, we also wrote a conceptually identical version of the code in C and compared the resulting output to that achieved by the Impulse C implementation. The two implementations produce identical classifications on a pixel-by-pixel basis. The combination of the good agreement of our results with the ASE results as well as the independence of the results from the software platform leads us to believe that our implementation is valid.

The color key is blue = water, cyan = ice, dark purple = snow, lavender = unclassified.

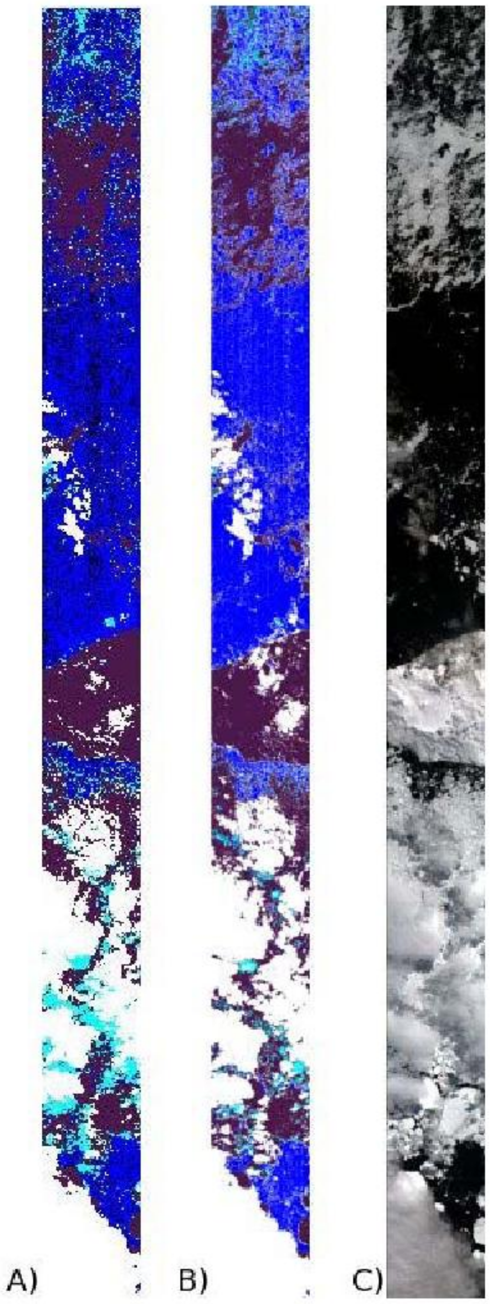


Figure 4. A comparison of the results from a) the Impulse C SVM implementation, b) the ASE SVM, and c) the original hyperspectral image.

	Percent of Pixels classified		
	ASE	Run 0716	Agreement
Snow	30.6	31.9	82.1
Water	31.0	31.1	81.7
Ice	3.0	7.3	28.8
Land	0.0	0.7	0.0
Cloud/unclassified	35.3	29.0	79.3

TABLE III. A COMPARISON OF THE PERCENTAGES OF PIXELS CLASSIFIED IN EACH CLASS BETWEEN THE ASE SVM AND OUR IMPULSE C SVM IMPLEMENTATION. THE AGREEMENT PERCENTAGES INDICATE THE PERCENT OF THE PIXELS CLASSIFIED IN EACH CLASS BY THE IMPULSE C SVM THAT WERE ASSIGNED TO THE SAME CLASS BY THE ASE SVM.

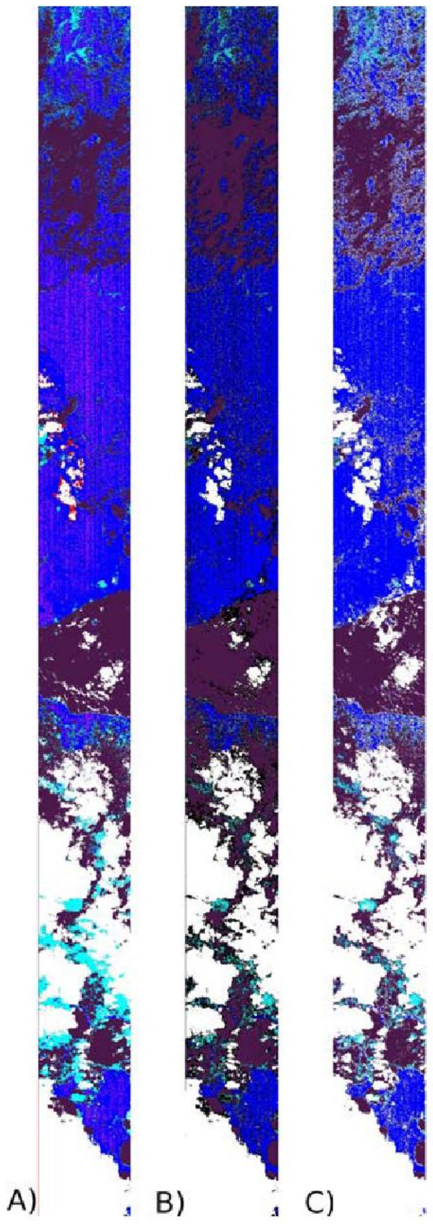


Figure 5. The black pixels in image (b) indicate the indices where the classifications of pixels (a) and (c) were not identical.

2.2 Implementation

In order to implement the SVM on the ML410 board, a minor modification to the producer/consumer model was required. The model requires the producer and consumer modules run concurrently, so the producer-hardware-consumer data flow necessitates a multi-threaded processing environment. This design allowed for simultaneous bidirectional communication between software and hardware, which permitted us to use small-depth buffers between hardware and software. In lieu of running a multi-threaded operating system on the PPC, we combined the producer and consumer functions into a single function that alternated hardware read and write operations. The communication between the software module and the hardware core was then implemented as two separate buffers (see Figure 6).

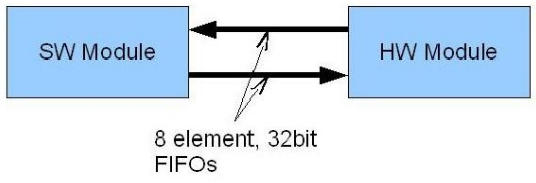


Figure 6. Hardware (HW)/Software (SW) Modules.

After this small algorithmic change, we used Impulse C to generate the hardware module. Following a few trivial changes to the software (an endian-swap, using optimized "printf" functions, etc.), the design was ready to be put onto the ML410 board. For this project we used the following board resources: a single PowerPC-405 (PPC) processor running at 100MHz, a Processor Local Bus (PLB), a 256MB DDR2 DIMM, a System ACE Compact Flash interface, an On-Chip Peripheral Bus (OPB), a PLB-to-OPB bridge, and a UART (see Figure 7). In addition, the hardware portion of the project was instantiated in the FPGA fabric.

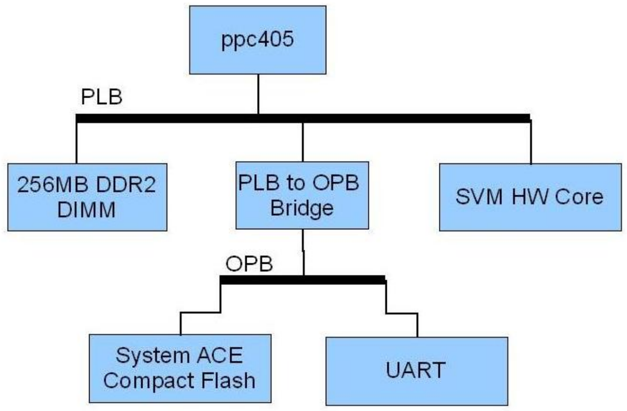


Figure 7. FPGA Hierarchy

The PPC ran the software portion of the task, which sends data to and collects data from the SVM hardware module. We chose to use the PPC instead of a Microblaze processor because the PPC can operate at triple the clock frequency of the Microblaze, and the Microblaze would be instantiated in valuable FPGA fabric, whereas the PPC exists external to the fabric. Since the 256MB DIMM is the largest source of memory on the board, we used it as main memory for the program. The PLB is a high-speed bus (compared to the OPB) that allows for fast data transfer to/from the memory and SVM core peripherals. The 16GB Compact Flash card was used to hold the input and output data files, which are too large to fit on the DIMM. The UART was used to for debugging output. The OPB is a low-speed bus that is the default interface between the processor and the System Ace controller and UART peripherals. We synthesized the design and ran it on the ML410 board. The classification output is shown in Figure 8a.

2.3 Extensions

Having successfully implemented the legacy SVM designed for Hyperion, we considered two extensions to the algorithm: using a larger number of bands with the same linear kernel SVM, and creating a new SVM with a nonlinear kernel. For the expanded linear kernel SVM, we arbitrarily selected 30 of the available 242 bands in the image. For the nonlinear kernel SVM, we used the same 11 bands as the legacy SVM with the kernel $\mathbf{K(x,y)=(<x,y> + 1)^2}$, where <x,y> is the dot product of 'x' and 'y'. Because training data was not available for the original legacy SVM, we could not generate new SVMs that would be comparable to it, so we used new training data to generate the two new SVMs then also generated a new 11-band linear-kernel SVM for comparison to the legacy SVM. See Table IV for FPGA fabric utilization percentages for each of these SVMs. Table V shows a runtime comparison of each SVM in a software-only implementation (PPC+FPU) and in the Impulse C co-designed implementation (PPC + HW).

TABLE IV. PERCENT FABRIC UTILIZATION FOR SVMS

	Linear (11 bands)	Linear (30 bands)	(2,1) Polynomial
Slices	4	8	8
Slice Flip Flops	2	2	4
4-input LUTs	3	6	6
FIFO16/RAMB16s	1	1	2
DSP48s	3	3	9

TABLE V. SVM RUN-TIME COMPARISON (IN MIN: SEC)

Classifier	PPC + FPU	PPC + HW	Speedup (raw)	Speedup (FPU/HW)
11-band Linear	2:04	0:52	1:12	2.4
30-band Linear	5:13	1:45	3:28	5.8
(2,1) Polynomial	32:24	3:11	29:13	10.2

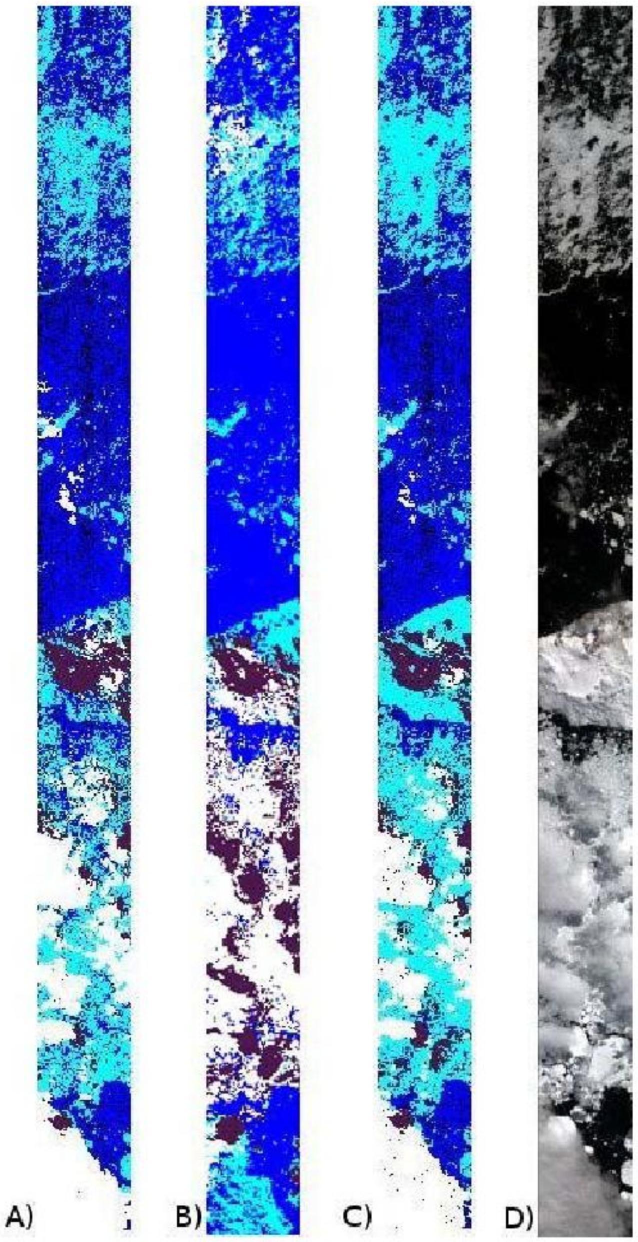


Figure 8. A comparison of the results from a) the 11-band linear SVM hardware implementation, b) the 30-band linear SVM hardware implementation, c) the polynomial SVM hardware implementation, to d) the original hyperspectral image. Color differences between image b) and the other 2 images (a & c) are due to the different bands (qty. 11 vs. 30) selected for each classification.

The hardware implementation of these SVMs produced results that agree very well with the software simulations of the algorithms. Figure 8b shows the 30-band linear SVM classification output. Figure 8c shows the polynomial SVM classification output. See Table VI for a summary of classification disagreement for each of the three SVMs. These disagreements may be due to floating-point hardware implementation differences between the FPGA hardware and the processor that ran the software simulations.

SVM Classifier	**SW/HW Implementation Difference [%]**
11-band Linear	0.34
30-band Linear	0.19
(2,1) Polynomial	1.23

TABLE VI. CLASSIFICATION DISAGREEMENT PERCENTAGE BEWTEEN SOFTWARE SIMULATION & PHYSICAL IMPLEMENTATION

3 SEU MITIGATION

Space-flight qualified FPGAs are susceptible to radiation single event upsets (SEUs), therefore this issue must be addressed for the SVM V4FX design to be flight-ready. The expected SEU rates of Rad-Hard flight processors, such as the RAD750, in a GEO environment is on the order of 1 error every 5-10 years. Expected SEU rates for the Xilinx Virtex FPGA are approximately one error per week. Recent data from similar FPGAs flown on JPL's Mars Exploration Rovers validate these predictions [10]. (It should be noted that next generation Xilinx parts such as the Virtex-4 are expected to be produced on CMOS SOI process lines, providing an order of magnitude improvement in SEU rate as well as other speed/power and radiation tolerance improvements). In order to achieve parity with Rad-Hard processors, we must reduce the SEU error rates by approximately two orders of magnitude and do this in a way that is relatively transparent to the application. Future work toward this goal could use the Xilinx Triple Modular Redundancy (TMR) Tool [11] to triplicate logic as there are sufficient remaining resources, as well as run the dual-core processors in lock-step. The simplest approach may be to include only SEU detection in the design and when detection occurs re-load the FPGA configuration file. This is a viable strategy for non-critical applications that can withstand occasional interruption for re-configuration. Partial reconfiguration is another possible solution, albeit more complex to implement, where only the effected portion of the FPGA needs to be configured.

4 CONCLUSION

FPGAs with embedded processing capabilities are demonstrating breakthrough performance previously impossible with traditional processors. This paper presented results from the synthesis of a legacy software SVM classification algorithm to the Xilinx V4FX60 FPGA platform as well as two extensions to demonstrate the increased capabilities of this implementation. Using commercially available C-to-HDL translation tools, this work was made possible under very limited funding. Hardware acceleration, of legacy software algorithms such as the described SVMs, promises to provide needed capability for more advanced on-board data processing in future science missions.

While the current method is to implement only those software classification algorithms that will fit within very constrained on-board processing resources, with embedded FPGAs such as the V4FX60, increasingly advanced SVMs may be implemented with "room to grow" in on-board resources. Our results demonstrate that our most advanced extension, the (2,1) polynomial kernel, is achieved with only 9% utilization of the FPGAs DSPs. Imagine the possibilities!

5 ACKNOWLEDGEMENTS

Abdullah Aljabri (JPL) and Charles D. Norton (JPL) for their vision and support of Smart Payload technology development.

6 REFERENCES

[1] *Virtex-4 Family Overview (DS112 v1.5)*, Xilinx Inc., San Jose, CA, 2006. Available: http://direct.xilinx.com/bvdocs/publications/ds112.pdf

[2] *MCP750 CompactPCI Host Slot Processor*, Motorola Computer Group, Temple, AZ, 2001. Available: http://www.acttechnico.com/mcp750.pdf

[3] *Xilinx Development Boards*, Xilinx Inc., San Jose, CA, 2006. Available: http://www.xilinx.com/publications/matrix/matrix_XDev_boards.pdf

[4] C. Cortez and V. Vapnik. "Support vector networks", *Machine Learning*, 20, (1995), 273 – 279.

[5] Rebecca Castano, Ngia Tang, Thomas Dogget, Steve Chien, Dominic Mazzoni, Ron Greely, Ben Cichy and Ashley Davis. "Onboard classifiers for science event detection on a remote sensing spacecraft", *Proceedings of the 12th ACM SIGKDD International conference of*

Knowledge Discovery and Data Mining, ACM Press, (2006), 845 – 851.

[6] M. Aizerman, E. Braverman and L. Rozonoer, "Theoretical foundations of the potential function method in pattern recognition learning", *Automation and Remote Control,* 25, (1964), 821 – 837.

[7] J. Platt, "*Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines*", Microsoft Research Technical Report MSR-TR-98-14, (1998).

[8] David Pellerin and Scott Thibault, *"Practical FPGA Programming in C"*, Prentice Hall, 2005. http://www.impulsec.com/

[9] P. Pingree, C. Norton, "Smart Payload Development for High Data Rate Instrument Systems", *Proceedings of the NASA Space Technology Conference (2007).*

[10] J. George, R. Koga, G. Swift, G. Allen, C. Carmichael, and C. W. Tseng, "Single Event Upsets in Xilinx Virtex-4 FPGA Devices," pre-publication paper, 2006.

[11] *Xilinx, TMRTool Fact Sheet* http://www.xilinx.com/esp/mil_aero/collateral/tmrtool_sellsheet_wr.pdf

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.

7 BIOGRAPHY

Paula Pingree is a Senior Engineer in the Instruments and Science Data Systems Division at JPL. She has been involved in the design, integration, test and operation of several JPL flight projects, the most recent being Deep Impact (DI) where she was Test Bench Manager pre-launch and Co-Lead of the Impactor Comet Encounter activity post-launch. She is presently the Electronics CogE for the Microwave Radiometer instrument on the Juno spacecraft planned for a 2011 launch to Jupiter. She also enjoys research and technology development for Smart Payloads such as this paper presents. Paula has a Bachelor of Engineering degree in Electrical Engineering from Stevens Institute of Technology in Hoboken, NJ, and a Master of Science in Electrical Engineering from California State University Northridge. She is a member of IEEE.

Lucas Scharenbroich is a Researcher in the Machine Learning and Instrument Autonomy group at JPL. He has been involved with the design, implementation, and deployment of machine learning algorithms to enable autonomous science capabilities, primarily through the Autonomous Sciencecraft Experiment (ASE). He is presently involved in the application of machine learning to automated code generation, crop yield prediction and object tracking in remote sensing data. Lucas has Bachelor of Science degrees in Electrical Engineering and Compute Science from the University of Minnesota, Duluth and a Master of Science in Information and Computer Science from the University of California, Irvine.

Thomas Werne is an Associate Engineer in the Model Based Systems Engineering and Architectures group at JPL. He is currently working on implementing FPGA-based technology for Smart Payload applications. Thomas has Bachelor of Science degrees in Electrical Engineering and Mathematics and a Master of Engineering in Electrical and Computer Engineering from Rose-Hulman Institute of Technology. He is a member of IEEE.

Christine Hartzell is an undergraduate Aerospace Engineering student at the Georgia Institute of Technology (GIT). She will be graduating in May 2008 and then pursuing a PhD in Aerospace Engineering, with a focus on Space System Design. Her research has focused on bioastronautics, instrument design methods, heat-shield design and trajectory selection in labs at GIT and at the Jet Propulsion Laboratory (JPL).