A Scalable Dataflow Accelerator for Real Time Onboard Hyperspectral Image Classification

Shaojun Wang[1,2], Xinyu Niu[2], Ning Ma[1], Wayne Luk[2], Philip Leong[3], and Yu Peng[1]

[1] Harbin Institute of Technology, Harbin 150008, China
wangsj@hit.edu.cn
[2] Imperial College London, London SW7 2NZ, UK
[3] University of Sydney, Sydney 2006, Australia

Abstract. Real-time hyperspectral image classification is a necessary primitive in many remotely sensed image analysis applications. Previous work has shown that Support Vector Machines (SVMs) can achieve high classification accuracy, but unfortunately it is very computationally expensive.This paper presents a scalable dataflow accelerator on FPGA for real-time SVM classification of hyperspectral images.To address data dependencies, we adapt multi-class classifier based on Hamming distance. The architecture is scalable to high problem dimensionality and available hardware resources. Implementation results show that the FPGA design achieves speedups of 26x, 1335x, 66x and 14x compared with implementations on ZYNQ, ARM, DSP and Xeon processors. Moreover, one to two orders of magnitude reduction in power consumption is achieved for the AVRIS hyperspectral image datasets.

1 Introduction

Hyperspectral image (HSI) classification aims to assign a categorical class label to each pixel in an image, according to the corresponding spectral and/or spatial features[1]. In satellite onboard processing, real-time HSI classification can significantly reduce download bandwidth and storage requirements, as well as enable greater autonomy due to improved real-time decision making ability. Moreover, improved processing speeds are necessary to match the higher spectral, spatial and temporal resolutions associated with improved sensors.

While there is a clear need for real-time HSI classification, it is challenging to meet both the required computational ($\approx 3 \times 10^{10}$ operations/second [11]) and power (< 20 W) constraints. In this paper we propose an FPGA-based SVM processor which fully meets these requirements. To the best of our knowledge, this is the first reported system that addresses this challenge.

Support vector machines (SVM) are a supervised non-linear machine learning technique which can effectively deal with the Hughes phenomenon [10], caused

This work was partially supported by the Fundamental Research Funds for the Central Universities(Grant No. HIT.NSRIF.201615)and China Scholarship Council.

by the high spectral dimensionality of HSI data. For this reason it has been widely used for HSI classification [3]. SVM classification is computationally intensive task, with computational complexity being linear with the number of support vectors (SV), the dimensionality of SVs, and the dimensionality of the problem [13]. Furthermore, a multi-class classifier is required for most remote sensing applications. In this paper, we focus on acceleration of the classification phase, and assume that training has been performed off-line.

FPGAs have been widely used to accelerate applications in a number of different fields usually achieving low power consumption[8]. Due to its importance, several FPGA-based implementation of SVMs have been reported using techniques such as Logarithmic Number Systems [4], Cascade SVM [13, 16], systolic architectures [4, 6], mixed-precision [14], coprocessor [2], and data flow architectures. Most of these studies focused on binary classifiers and were tailored to special applications. This work addresses the multi class classification problem in which strategies for dealing with data dependencies between binary classifiers are explored.

This paper proposes a scalable SVM multi-class classifier accelerator for HSI classification which achieves real-time on-board classifications under strict power and volume constraints. The main contributions are:

- A scalable accelerator architecture which utilises dataflow programming technology to maximize performance.
- Models to predict and optimise the proposed architecture.
- An implementation of the accelerator on a Maxeler MPC-X1000 dataflow node. The runtime, energy consumption and classification accuracy are evaluated and compared to ARM, ZYN, DSP and Xeon on real HSI datasets.

2 Background

Hyperspectral Images are typically represented as a data cube [1], $Z \in R^{n_1 \times n_2 \times n_b}$, with spatial information collected in the X-Y plane containing $n_1 \times n_2$ pixels, and spectral information represented in the Z-direction with n_b spectral bands.

Each pixel can be represented as a vector $z \in R^{n_b}$ in spectral space. Similar materials on the earth's surface have similar spectral feature, making the pixels separable. Multi-class classifiers are built from multiple binary classifiers and strategies can be parallel or hierarchical. Parallel approaches usually provide higher accuracy and less data dependencies, but require more binary classifier instances compared to hierarchical approaches [10]. Thus they are more suitable for FPGA based implementation. Parallel approaches label a new sample according to the result of a discriminant function whose inputs are the output of several parallel binary classifiers. The One-Against-One (OAO) parallel strategy usually provides higher accuracy when used with a proper discriminant function such as Hamming distance.

In OAO, a K class classification uses $K\left(K-1\right)/2$ binary classifiers to compute all pair-wise values. Each binary classifier is trained with the same number of samples from two different classes. After training, the classification of a new pixel vector,$z \in R^{n_{b1}}$, involves the process as shown in algorithm 1.

Algorithm 1 Multi class SVM classifier with Hamming Distance as discriminant function

```
1: input a test pixel $z$;
2: for Iteration $j$ from 1 to $K(K-1)/2$ do
3:     compute the $j^{th}$ Hamming Code bit with SVM binary classifier as
           $$R\_code(j) = sign\left[\sum_{i=1}^{l} \alpha_{(j,i)} K(z, x_{(j,i)}) + b_j)\right] \quad (1)$$
4: end for
5: for Iteration $j$ from 1 to $K$ do
6:     compute the Hamming distance with each class's mask and identifying code as
           $$T\_res(j) = (R\_Code\&mask(j)) \oplus I\_code(j) \quad (2)$$
7:     for Iteration $i$ from 1 to $K(K-1)/2$ do
8:         accumulate the total none zero bits as hamming distance
               $$H(j) = (T\_res_{(j,i)} == 1)?H(j) + 1 : H(j) \quad (3)$$
9:     end for
10: end for
11: label the pixel with the index of $minH(j)$
```

For each new test data, $K(K-1)/2$ binary classifiers can generate the corresponding $K(K-1)/2$ bit Hamming code, using equation(1), in which $x_{(j,i)} \in R^{n_{b1}}$ is the i^{th}support vector in the j^{th} SVM binary classifier, n_{b1} is the support vectors's dimension which is usually much less than the number of spectral bands n_b, l is the total number of support vectors in each SVM binary classifier, $\alpha_{(j,i)}$ is the i^{th} Lagrange multiplier in j^{th} SVM binary classifier, b_j is a real constant, and $K(z, x_{(j,i)})$ is the kernel function. In this work, we employ the widely-used radial basis function (RBF) kernel:

$$K(z, x_j) = exp\left\{- \parallel z - x_j \parallel_2^2 / \sigma^2\right\} \quad (4)$$

where σ is the width parameter. The values of the hyperparameters, σ, α and b are ascertained by cross validation during training.

Each class also has an identifying code formed in the training process. By computing the Hamming distances between the test data's Hamming code and each classes' according to equation(2) and (3), the test data is labelled with the class with the minimum corresponding Hamming distance.

In equation(2), $mask(j)$ and $I\_Code(j)$ are the mask code and identifying code of the j^{th} class. Table 1 shows an example in which these codes are generated for a 4 class classification problem. In this example, we use $C_4^2 = 6$ binary classifiers, with the binary classifier for classes 1 and 2 being labelled as $1vs2$. The outputs can have values of 1, 0, and x, where 1 and 0 indicate whether the processed datum is in this class, and x indicates the output is not related to this class. In Table 1, the outputs of $2vs3$, $2vs4$, and $3vs4$ are labelled x for class 1, since class 1 is not used in these classifiers. To support efficient hardware operators, the mask code sets the bit that corresponds to x to be 0 to only use relevant outputs, and the identifying code contains the classifier outputs.

Table 1. The identifying and mask codes for a 4-classification problem

	1vs2	1vs3	1vs4	2vs3	2vs4	3vs4	identifying code	mask code
Class 1	1	1	1	x	x	x	111000	111000
Class 2	0	x	x	1	1	x	000110	100110
Class 3	x	0	x	0	x	1	000001	010101
Class 4	x	0	0	x	0	0	000000	001011

3 Accelerator Architecture

3.1 Architecture Overview

The data flow accelerator architecture is shown in Fig. 1. We implement a data flow engine (DFE) on an FPGA chip. The DFE takes newly sampled data, and outputs classification results to the decision system or downlink system of satellites. Several binary classification kernel (BC kernel) groups are instantiated on the FPGA, each of which can generate the Hamming code for an image pixel. With the Hamming distance kernel following each BC kernel group, each pixel is assigned a class label. The collection kernel combines the results of the Hamming distance kernel to a certain bit width and finally outputs the results for all the pixels processed in the DFE.

The number of BC kernel groups in the DFE can be adjusted according the classification problem size and the hardware resources available on the FPGA. The whole system is scalable and flexible, and can fit different application. Moreover, no data dependencies exist between different BC kernels. All BC kernel groups operate simultaneously to achieve the best performance under memory and interface bandwidth constraints.

3.2 Memory and Computation Data Flow

As shown in Fig.1, different BC kernel groups have separate image data RAM and share preload SVM model ROMs. A single DFE contains M BC kernel groups, and M image data RAMs are instantiated. The preload SVM model ROMs store the parameters for each binary SVM classifier that comprise the support vectors and corresponding alpha parameter. For each binary SVM model, a support vector (SV) ROM and corresponding Alpha parameter ROM are

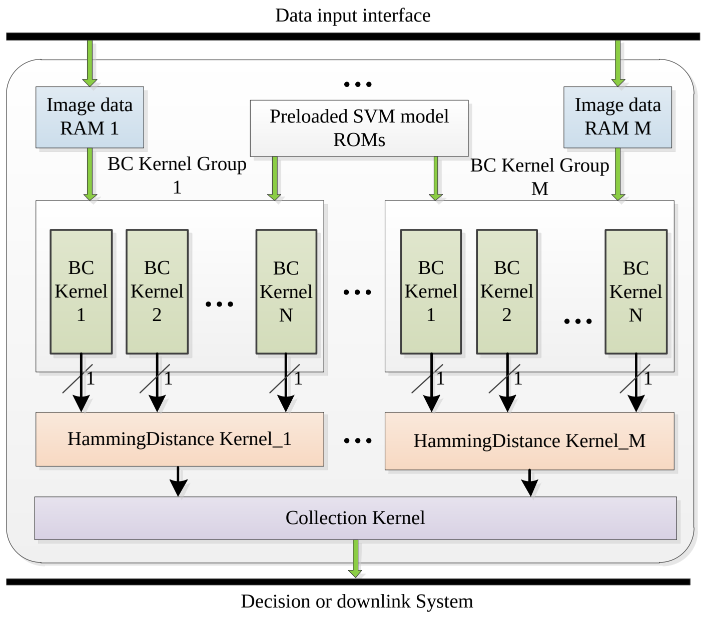

Fig. 1. Architecture of multi class classifier for HSI classification

instantiated. For the BC kernel group containing N BC kernels, N SV ROMs and N Alpha ROMs are needed. The data in each SV ROM and each Alpha ROM are shared by M BC kernel with the same index in M BC kernel groups. All the data are stored as 32 bit single precision values for compatibility with software, but computations are in fixpoint(16,16) to save DSP resources and decrease computing latency. The data flow in DFE is shown in Fig.2 In Fig. 2, $z(i,j)$ is the j^{th} element of i^{th} test pixel vector, $SV\_t(i,j)$ is the j^{th} element of i^{th} support vector in t^{th} binary classifier, $Alpha\_t(i)$ is the i^{th} Lagrange multiplier in the t^{th} binary classifier, l is the total number of support vectors for each binary classifier and n_{b1} is the dimension of support vector.

As shown in Fig.2, M BC kernel groups, containing total $M \times N$ BC kernels, work simultaneously. Each kernel inputs one element of the support vector and one element of the test pixel vector in each clock tick. So M test pixels can get their corresponding N bits hamming codes in $l \times n_{b1}$ ticks.

3.3 Design Models

In this section, we analyze the performance of the aforementioned architecture. We use a resource specification file to indicate the number of available resources in the target platform. The available resources include on-chip logic resources and off-chip bandwidth resources.

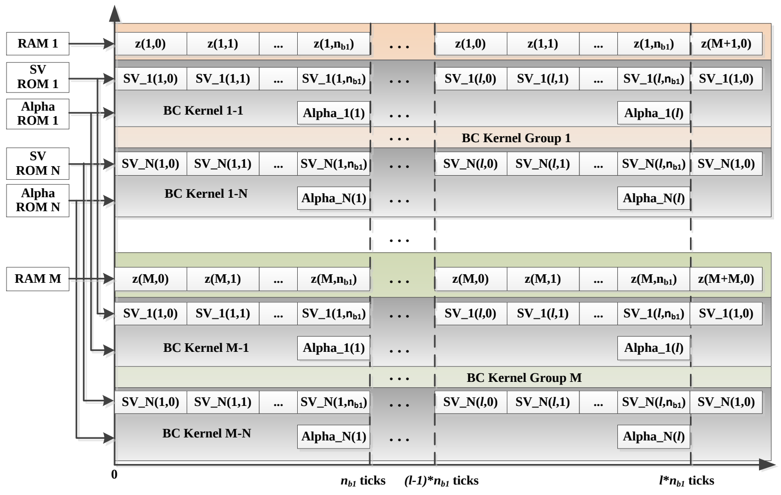

Fig. 2. Data flow in DFE with M classifiers and N BC kernel in each classifier

In order to eliminate performance bottlenecks, we develop design models to estimate resource usage based on application variables and design parameters, as listed in Table 2.

Table 2. Variables and parameters used by design models.

Design Parameters	
na_i	number of arithmetic operators of type i
$r_{i,L/F/D}$	resource usage of a type i arithmetic operator on LUTs, FFs, or DSPs
r_M	number of bits can be stored in a BRAM
$A_{L/F/D/M/BW}$	available logic, memory, and memory bandwidth resources
Application variables	
K	the number of classes
N	the number of BC kernels in each group and equals to $K\,(K-1)\,/2$
M	the number of BC kernel Groups in single DFE
l	the number of support vectors for each binary classifier
n_{b1}	the dimension of each support vector
Clk_Fren	the system clock frequency for the kernels

Bandwidth Analysis. ROM initialization is one-time process which is prior to the computation process and has no strict bandwidth requirements.

The off-chip data transfer only involves the data interface (PCIe, Ethernet, 1394) through which CPU or Spectral meter transfers test pixel data to RAMs and reads back classification results during computation process.

According to Fig. 2, M single precision pixel vectors should be transferred to RAMs in one clock in each first n_{b1} clock ticks starting from each integer times of $l \times n_{b1}$ clock ticks. The peak bandwidth is $32 \times M \times Clk\_Fren$. The pixel vector data will be stored in RAM and repeatedly used in total $l \times n_{b1}$ clocks.

The BC kernel group needs $l \times n_{b1}$ clock cycles to process one pixel, and the result is 4bits (for the maximum 16 classes situation), even M BC kernel groups work simultaneously, the output bandwidth is $4 \times M/(l \times n_{b1})$ bits/s. Therefore, we express the total bandwidth requirements (bits/s) as:

$$BW = 32 \times M \times Clk\_Fren + 4 \times M/(l \times n_{b1}) \tag{5}$$

Hardware Utilisation Analysis. The logic cells, on-chip memory and DSP blocks are the main resources consumed in the design. We list the data-path structure for BC kernels in Fig. 3. Most of the on-chip resources are consumed by arithmetic operators in BC kernels.

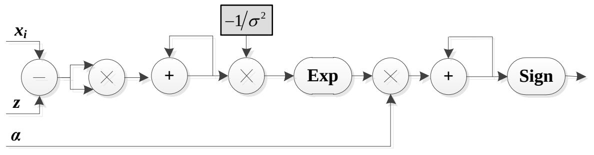

Fig. 3. Data path of binary classifier kernel

The proposed architecture contains $N \times M$ BC kernels. We express the on-chip logic resource usage as

$$R_{L/F/D} = \sum_{i \in \odot = +,-,*,\ldots} na_i \cdot r_{i,L/F/D} \cdot N \cdot M \tag{6}$$

where na_i indicates the number of arithmetic operators of type i, and $r_{i,L/F/D}$ indicates the resources (LUTs, FFs, or DSPs) consumed by the operator. As an example, as shown in Fig. 3, a BC kernel contains 3 multipliers ($na_\times = 3$) and each multiplier uses 1 DSP block ($r_{\times,D} = 1$).

The direct on-chip memory requirements involve the SV ROMs, Alpha ROMs and RAMs. We map these memory into Block RAMs (BRAMs) in FPGAs.

In Table 3, we list the number of bits the total on-chip memory architecture needs to store, and use r_M to indicate the memory capacity of a BRAM. As shown in the table, the number of total memory blocks can be expressed as:

$$R_M = 32 \times \lceil (N \times l \times n_{b1})/r_M + N \times l/r_M + (M \times n_{b1})/r_M \rceil \tag{7}$$

Performance Model As shown in Fig. 1, a DFE can process M pixels in $l \times n_{b1}$ clock cycles when the bandwidth and hardware requirement are both satisfied.

Table 3. Memory requirements

	SVROM	Alpha ROM	RAM
Width(bits)	32	32	32
Depth	$l \times n_{b1}$	l	n_{b1}
Number	N	N	M
Total(bits)	$32 \times N \times l \times n_{b1}$	$32 \times N \times l$	$32 \times M \times n_{b1}$
Block Memory	$\lceil(32 \times N \times l \times n_{b1})/r_M\rceil$	$\lceil(32 \times N \times l)/r_M\rceil$	$\lceil(32 \times M \times n_{b1})/r_M\rceil$

The system performance is $M \times Clk\_Fren/(l \times n_{b1})$ pixel/s. For real time image processing, the sampling rate S must satisfy the following formula.

$$M \times Clk\_Fren/(l \times n_{b1}) \geq S \tag{8}$$

s.t.

- $32 \times M \times Clk\_Fren + 4 \times M/(l \times n_{b1}) \leq A_{BW}$
- $R_{L/F/D} = \sum_{i \in \odot = +,-,*,\ldots} na_i \cdot r_{i,L/F/D} \cdot N \cdot M \leq A_{L/F/D}$
- $R_M = 32 \times \lceil(N \times l \times n_{b1})/r_M\rceil + \lceil N \times l/r_M\rceil + \lceil(M \times n_{b1})/r_M\rceil \leq A_M$

where A_{BW} indicates the available off-chip memory bandwidth, and $A_{L/F/D}$ and A_M indicates the available on-chip logic and memory resources.

4 Experiments and Results

In this section the performance of the proposed accelerator is compared to systems using radiation hardened and state-of-the-art commercial multi core CPUs.

4.1 Experimental Setup

The accelerator is implemented on a Maxeler MAX4 DFE which is equipped with an Altera Stratix V 5SGSMD8N2F45C2 FPGA. Although the DFE board is connected to a server via PCIe, our accelerator can operate without an external server, thus making it more suitable for space applications. We used the MAXJ language to express the accelerator design. The MaxCompiler maps the design to the FPGA and provides APIs for the host application running on the CPU.

The HSI data sets used are the well-known Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS) Northwestern Indiana scene and Salinas Valley scene. These images data contain 224 spectral bands. However, because of water absorption and information redundance, only a few spectral bands, e.g. 9 in this study, are used for training and classification. Both images contain 16 classes.

4.2 Classification Accuracy and Hardware Occupation

LS-SVM[⋆] is adopted in training phase to keep the number equality of support vectors in each binary classifier. We evaluate the overall accuracy of our accelerator in the aforementioned two image data sets. For both images, we try to

[⋆] The source code can be found from http://www.esat.kuleuven.be/sista/lssvmlab/

classify 6 classes from totally 16 classes. Each binary classifier is trained with 100 samples containing 9 spectral bands, and 15 binary classifiers are used to realize the 6 class classification problem. 540 pixels in each image are used as the test pixel vectors. The overall accuracy and comparison with some recent research are shown in Table 4.

Table 4. Overall accuracy(OA) comparison

Methods	OA on first image(%)	OA on second image(%)
Approach in this paper	98.3	97.8
ANN based Adaboost[15]	98.02	-
MLRsub[5]	92.5	-
HA-PSO-SVM[17]	98.2	-
SdA[7]	91.9	95.5

From Table 4, the overall accuracy of the multi classifier based on Hamming distance is almost the same or better than other methods on these two data sets. We did not realize 16 classes because the other 9 classes did not have enough labeled samples for training. Many approaches have been proposed to solve this problem [17]. These methods are implemented during the training phase, and can be combined with the Hamming distance method of this study during the classification phase. The only problem is that we need more BC kernels in a group to realize multi classification.

Eight BC kernel groups each containing 15 BC kernels, 8 Hamming Distance kernels, 8 RAM kernels, 15 SV ROM kernels and 15 Alpha ROM kernels and one Collection kernel are instantiated in a single DFE. The target operating frequency is set to 120 MHz. The hardware utilisation after map and routing using Altera Quartus II 13.1 tool set is shown as in Table 5.

Table 5. FPGA resource utilizationm

Resources	Logics	FFs	DSPs	Block Memory
Used	234666	443688	1680	1715
Avaiable	262400	524800	1963	2567
Utilization	89.43%	84.55%	85.58%	66.81%

From Table 5, we can see that hardware resources are almost fully utilized and Utilization is balanced between logic, flip-flops, DSPs and memory.

4.3 Performance Comparison with Other Processors

Radiation harden processors are the traditional option for satellite on-board computers, and performance of the most advanced space grade CPU, e.g. RAD750 from BAE systems, is just about 400DMIPS@200MHz. Other processors, such as ARM and DSP are also used in some low cost space missions, especially in some experimental micro satellites. In this context, we compare the run time and energy consumption between our DFE accelerator and some available commercial processors whose performances are similar with space grade CPUs. These processors include ZYNQ XC7Z020, ARM Cortex A9 and TMS320C6678 DSP. Xilinx ZYNQ XC7Z020 chip runs at 100MHz frequency. Six binary SVM classifiers are instantiated in the PL part with HLS design and computation flow and data transfer management are performed with the ARM processor. The ARM Cortex-A9 processor runs at 666.7MHz, and employs the Vector Floating Point Unit and 32KB Cache to speed up the computing. TMS320C6678 DSP is one of the most advanced multicore DSP chips from Texas Instruments. It runs at 1GHz and 8 cores are programmed in parallel. One million test pixel vectors are used for the evaluation. We also compare the performance of the accelerator with state-of-the-art dual Intel Xeon E5-2650 CPUs to demonstrate its performance advancement. The Xeon CPUs has 12 cores and 12 threads parallel programming are designed with OpenMP library. Eight millions test pixel vectors are used for duel CPUs performance evaluation. The detailed result is shown as in table 6.

Table 6. Runtime and energy consumption comparison

Platform	ZYNQ	ARM	DSP	Xeons	DFE
T(μs/pixel)	25.8	1321.2	65.8	14.1	0.99
Power (W)	3.9	3.3	16	95	26.3
E(mJ/pixel)	0.1	4.3	1.05	1.33	0.03
Speedup	26.0	1334.5	66.4	14.2	1

From Table 6, the accelerator on DFE gets 26x, 1334.5x, 66.4x and 14.2x speed up compared to the ZYNQ, ARM, DSP and Xeon processors respectively, while consuming two orders of magnitude less energy than the ARM, DSP and Xeon processors. However, less hardware resource in radiations harden FPGA chips and other reliability related measures, such as triple module redundancy(TMR) and configuration scrubbing, can decrease the performance in real space application compared to this study. The most advanced space grade FPGA chip, Xilinx Virtex-5QV XQR5VFX130, has about 1/7 hardware resources of the FPGA chip in this research. Taking the extra hardware consumption for TMR into account, the performance in space grade FPGA may decrease about 21 times compared to FPGA in this paper. Nevertheless more than 60x and 3x speed up compared to ARM and DSP respectively can be achieved.

4.4 Performance Comparison with Design Model

To evaluate the accuracy of the design model proposed in section 3, we compare the real performance of accelerator with the theoretical performance from design model in terms of the average processing time for one pixel as shown in Fig.4.

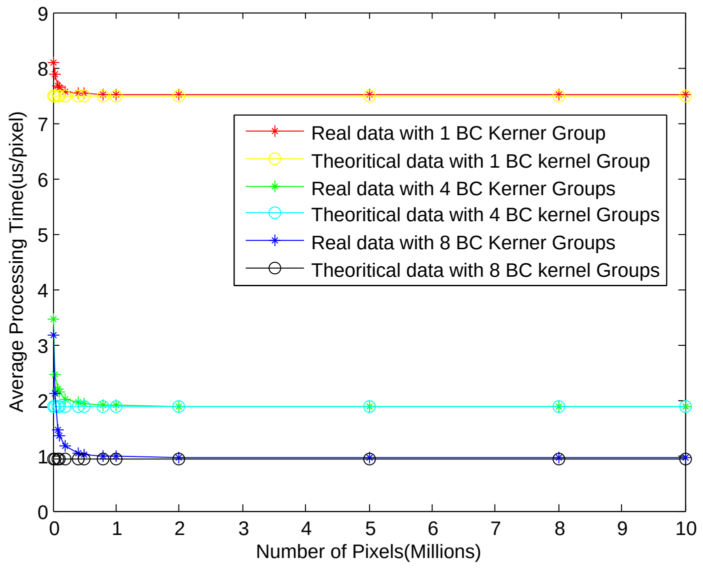

Fig. 4. Performance comparison between the models and real implementation

From Fig.4, the real performances are similar to the theoretical results especially when the amount of pixels are larger than 1M. When the pixels are less than 1M, the real performance is lower than theoretical value. It's caused by kernel initialization which is a relatively fixed period but has more influences on the average performance for small datasets. The results demonstrate the accuracy of the design model is good enough for performance prediction and optimization.

5 Conclusion

This paper proposes a novel accelerator architecture for real time hyperspectral image classification which is a bottleneck for on-board hyperspectral image analysis. The Hamming distance based SVM is adopted as the multi class classifier which provides high accuracy and avoids data dependency between different binary classifiers. The accelerator uses the advantages of dataflow programming to achieve high performance and can be easily scaled to fit different applications. The accelerator is implemented on the Maxeler MAX4 DFE board. Experimental results on real HSI data sets show that Hamming Distance based multi classes SVM classifier can achieve higher or equal accuracy compared to other approaches, with 26x, 1334.5x,66.4x and 14.2x speed up over ZYNQ, ARM, D-SP and Xeon processors, while consuming one or two orders of magnitude lower energy. These results show for the first time, the feasibility of real-time on-board HSI classification. Future work involves accelerating the multi class classifiers with feature extraction, and extending the accelerator architecture to other state-of-the art classifiers, such as convolutional neural networks.

References

1. Bioucas-dias, J.M. et al. Hyperspectral Remote Sensing Data Analysis and Future Challenges. IEEE Geoscience and Remote Sensing Magazine. 6, 6-36 (2013)
2. Cadambi, S., Igor, D., et al. A massively parallel FPGA-based coprocessor for support vector machines. Proceedings - IEEE Symposium on Field Programmable Custom Computing Machines, FCCM 2009. 115-122(2009)
3. Gustavo, C., Davis, T., et al. Advances in hyperspectral image classification: Earth monitoring with statistical learning methods. IEEE Signal Processing Magazine. 31, 1, 45-54 (2014)
4. Irick, K.M. et al. A hardware efficient support vector machine architecture for F-PGA. Proceedings of the 16th IEEE Symposium on Field-Programmable Custom Computing Machines, FCCM'08. 304-305(2008)
5. Khodadadzadeh, M. et al. A New Framework for Hyperspectral Image Classification Using Multiple Spectral and Spatial Features. IEEE Geoscience and Remote Sensing Symposium. 4628-4631(2014)
6. Kyrkou, C. and Theocharides, T. SCoPE: Towards a systolic array for SVM object detection. IEEE Embedded Systems Letters. 1, 2, 46-49(2009)
7. Liu, Y. et al. Hyperspectral classification via deep networks and superpixel segmentation. International Journal of Remote Sensing. 36, 13, 3459-3482(2015)
8. Lopez, S. et al. The promise of reconfigurable computing for hyperspectral imaging onboard systems: A review and trends. Proceedings of the IEEE. 101, 3, 698-722 (2013)
9. Mandal, B. Design of a Systolic Array based Multiplierless Support Vector Machine Classifier. 2014 International Conference on Signal Processing and Ingegrated Networks. 2, 35-39(2014)
10. Melgani, F. and Bruzzone, L. Classification of hyperspectral remote sensing images with support vector machines. Geoscience and Remote Sensing, IEEE Transactions on. 42, 8, 1778C1790(2004)
11. Montenegro, S. et al. Hyperspectral monitoring data processing. ISBN 3-89685-569-7, 1-4(2003)
12. Papadonikolakis, M. et al. Performance comparison of GPU and FPGA architectures for the SVM training problem. Proceedings of the 2009 International Conference on Field-Programmable Technology, FPT'09. 388-391 (2009)
13. Papadonikolakis, M. and Bouganis, C.S. Novel cascade FPGA accelerator for support vector machines classification. IEEE Transactions on Neural Networks and Learning Systems. 23, 7, 1040-1052 (2012)
14. Papadonikolakis, M. and Bouganis, C.S. A Heterogeneous FPGA Architecture for Support Vector Machine Training. Field-Programmable Custom Computing Machines (FCCM), 2010 18th IEEE Annual International Symposium on. 6-9(2010)
15. Sami, Q. et al. Neural Network based Adaboosting Approach for Hyperspectral Data Classification. 2011 International Conference on Computer Science and Network Technolgoy. 241-245(2011)
16. Theocharides, T. and Member, S. Embedded Hardware-Efficient Real-Time Vector Machines. IEEE Transactions on Neural Networks and Learning Systems. 1-14(2015)
17. Xue, Z. et al. Harmonic Analysis for Hyperspectral Image Classification Integrated With PSO Optimized SVM. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. 7, 6, 2131-2146(2014)