www.czasopisma.pan.pl PAN www.journals.pan.pl

INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2022, VOL. 68, NO. 3, PP. 609-617
Manuscript received October 12, 2021; revised August, 2022. DOI: 10.24425/ijet.2022.141280

An Efficient Classification of Hyperspectral Remotely Sensed Data Using Support Vector Machine

Mahendra H N, and Mallikarjunaswamy S

***Abstract*—This work present an efficient hardware architecture of Support Vector Machine (SVM) for the classification of Hyperspectral remotely sensed data using High Level Synthesis (HLS) method. The high classification time and power consumption in traditional classification of remotely sensed data is the main motivation for this work. Therefore presented work helps to classify the remotely sensed data in real-time and to take immediate action during the natural disaster. An embedded based SVM is designed and implemented on Zynq SoC for classification of hyperspectral images. The data set of remotely sensed data are tested on different platforms and the performance is compared with existing works. Novelty in our proposed work is extend the HLS based FPGA implantation to the onboard classification system in remote sensing. The experimental results for selected data set from different class shows that our architecture on Zynq 7000 implementation generates a delay of 11.26 µs and power consumption of 1.7 Watts, which is extremely better as compared to other Field Programmable Gate Array (FPGA) implementation using Hardware description Language (HDL) and Central Processing Unit (CPU) implementation.**

***Keywords*—Support Vector Machine (SVM); Central Processing Unit (CPU); Digital Signal Processor (DSP); Field Programmable Gate Array (FPGA); High Level Synthesis (HLS); Hardware description Language (HDL)**

I. Introduction

Hyperspectral image classification aims to assign each pixel of an image to a corresponding class label based on the spatial and spectral features. The Field Programmable Gate Array (FPGA) based classification enables us to classify remotely sensed (RS) data in real-time, therefore it can be extended to use in onboard classification at satellite platform to overcome the issues of limited downlink bandwidth [1]. The available bandwidth between satellite and ground station cannot be sufficient to transfer remotely sensed data to the ground station because of its large in size. Therefore onboard classification will resolve this issue and it meets the real-timeclassification requirements i.e $3x10^{10}$ operation/second [2]. In recent days, we can find significant research on the above challenges. The Graphical Processing Unit (GPU) was used for processing the hyperspectral images and it gives better performance, but its high power consumption is not a good choice for onboard application [3]. In onboard applications, FPGA offers superior performance by providing low power consumption [4] and low hardware utilization [5]. Therefore, this work aims to develop the FPGA based Support Vector Machine (SVM) for classification of hyperspectral RS data.

In remote sensing, image classification is an important task [6]. The supervised machine learning tool such as Support Vector Machine (SVM) classifier is widely used for classification to get maximum accuracy. SVM exhibits high classification accuracy in many applications such as image classification, object detection, speech recognition, and medical diagnosis [7]. In hyperspectral classification, SVM performs better accuracy than other classifiers [8]. The supervised learning machines performing the function in two different phases, learning or training phase and classification phase. In the training phase, the SVM develops the model for classification on given test of data based on the Support Vectors (SVs). These trained SVs are further used to find the class of data during the classification phase.

In many applications, the modeling of SVM is time consuming and computationally expensive. The software based implementations of SVM produce high accuracy rates, but it cannot meet the real-time requirements. To meet the real-time constraints in remote sensing, a dedicated hardware need to be design and implemented with low power consumption and resource utilization. FPGA is a parallel processing reconfigurable device to meet the constraints of low power consumption and resource utilization in real time applications [9]. Therefore, FPGA is an appropriate choice to meet the real time requirements. In this paper, we propose an efficient hardware architecture for implementing the Support Vector Machine (SVM) on FPGA for Hyperspectral image classification.

This research was supported by JSS Academy of TechnicalEducation Bangalore, affiliated to Visvesvaraya Technological University, Belagavi and VGST sponsored renewable energy laboratory for grant of financial assistance

Mahendra H N is with Department of Electronics and Communication Engineering, JSS Academy of Technical Education Bengaluru and Affiliated to Visvesvaraya Technological University, Belagavi, India.(email-mahendrahn@jssateb.ac.in)

Mallikarjunaswamy S, is with Department of Electronics and Communication Engineering, JSS Academy of Technical Education Bengaluru and Affiliated to Visvesvaraya Technological University, Belagavi, India. (email-mallikarjunaswamys@jssateb.ac.in)

II. RELATED WORK

In the literature many researchers have implemented the FPGA based architectures for realizing the SVM classification for real-time application [9] [10]. The main constraints such as high performance, power consumption, scalability, flexibility, area, and low cost are to be considered to achieve the effective classification. These constraints are not effectively considered in the current architectures and implementations, that too power consumption becomes critical for onboard classification in remote sensing. Therefore in this section, some of the existing work on FPGA-based architectures for realizing the SVM classification in different fields are reviewed.

The hardware architecture for radial basis function-based SVM for implementation of the exponential function is presented [11]. Sumeet Saurav et al. [12] have designed the FPGA based architecture to implement the multi-class linear SVM for the classification of individual facial expression. A Novel SVM architecture is presented [13] to over the issues of data linear dependencies on the number of the Support Vectors. In industrial ultrasound applications to detect ultrasonic A-scan signals, an FPGA based SVM classifier is designed [14]. An implementation of SVMs on FPGA is also extended to Logarithmic Number Systems [15]. Also in studies such as mixed-precision [16], systolic architectures [17], and a coprocessor [18] have focused on the FPGA based binary SVM classifier design.

To the best of our knowledge, no SVM implementation on FPGA exists in the literature that has not to use the combination of hardware and software hybrid architecture for classification of hyperspectral RS images. Hence, a recent platform of FPGA "Zynq System on Chip (SoC)" is the best choice for the implementation of hybrid architecture. Also, most of the implementation makes use of the traditional Hardware Description Language (HDL), which requires an expert hardware developer and highly time-consuming. Therefore, the Ultra-Fast High-Level Synthesis (HLS) methodology has been explored to simplify the design in the development of FPGA [23].

In our previous study [10] several challenges, limitations, and research gaps in real-time processing and classification of remotely sensed have been addressed. Consequently, this work aims to implement the SVM on FPGA and find the best for the onboard application. An SVM architecture implemented based on hardware/software co-design is compared with existing works.

III. PROPOSED DESIGN FOR SVM IMPLEMENTATION

A. System Development Tools and FPGA Platform

In our work to implement the hybrid architecture on FPGA, the software tool "Xilinx Vivado Design Suite" has been used. With the help of this tool, the designer can develop an FPGA within a single SoC [24]. The tool also support to UltraFast HLS design methodology, therefore it becomes more power full in digital design. This methodology is based on High-Level Language (HLL) for simplifying FPGA programming by replacing the traditional HDL [24] and reduces the FPGA development time and effort.

The “Xilinx Zynq-7000 All Programmable System on Chip (SoC)” FPGA platform has been used to implement our proposed SVM classifier. The hybrid architecture is characterized by Zynq SoC, which makes the process of embedded system development significantly simple. An ARM and FPGA are combined as a Processing System (PS) and Programmable Logic (PL) respectively.

B. Proposed SVM HLS Implementation on Zynq PL

A binary SVM classifier architecture is designed by utilizing the recent design methodology of HLS and incorporating a linear kernel function. The SVM classification algorithm is implemented by HLS design, which classifies sample x of test data in the main decision function. The equation (1) describes the decision function based on the parameters b, y and α that are determined during training phase [26].

$$f(x) = sign(\sum_{i=1}^{SV} \alpha i \; yi \; (\overrightarrow{xi} \, . \overrightarrow{x}) - b) \tag{1}$$

The decision function (1) is computed in C/C++ language using the Vivado HLS tool. To reduce complexity in hardware design the equation (1) is divided into 3 main equation as follows.

$$\overrightarrow{AC} = \sum_{i=1}^{sv} \alpha i \; yi \; \overrightarrow{xi} \tag{2}$$

$$D = \overrightarrow{AC} . \vec{x} \tag{3}$$

$$F(X) = \; sign(D - b) = \begin{Bmatrix} -1, (D - b) < th \\ 1, (D - b) \geq th \end{Bmatrix} \tag{4}$$

In the developed algorithm, three main tasks are considered to perform the required calculation, and which are mapped to the corresponding equation (2), (3), and (4). The first task performing the function of equation (2), which stores the summing of all multiplied SVs in an accumulated array (AC).

The dot product between the accumulated array and test instance of equation (2) is performed in the second task to calculate the classification distance value (D). In the last task based on the sign value, the calculated distance number is classified ($F(x)$), and threshold value *th* is calculated through the validation phase. The outcome of this process has two possibilities either 1 or -1, which are corresponding to class 1 or class 2. The flow diagram of the proposed SVM HLS Implementation is shown in Fig. 1.

The hardware/software co-design based architecture is developed on Zynq SoC to implement the proposed SVM is shown in Fig. 2. In implementing the SVM classifier, the proposed architecture depends on the size of both the SVs and features. The needed input data to the design is received through the input stream interface and stored in three main memory. The features of SVs are stored in one 2D array, corresponding *ay* of each SV in a 1D array, and test instance features data are stored in another 1D array.

The HLS tool effectively helps to develop the Intellectual Property (IP) for the proposed SVM classifier on Zynq SoC [25], by providing the various directives. The stream directive AXI4 is used as an input stream to streaming the data between the designed HLS IP in the PL slice and PS (ARM CPU) on Zynq SoC. Also, the AXI-lite bus is used as a control bus to control the data flow through the ARM CPU and controlling the designed IP.

In the proposed architecture, the HLS IP module is designed with the help of Vivado HLS tool and implemented on PL part of the Zynq platform. To perform multiplication process within the loop, HLS tool is used with C language. The rest of the code is about reading the data from the file of the trained model and doing some processing for preparing data.

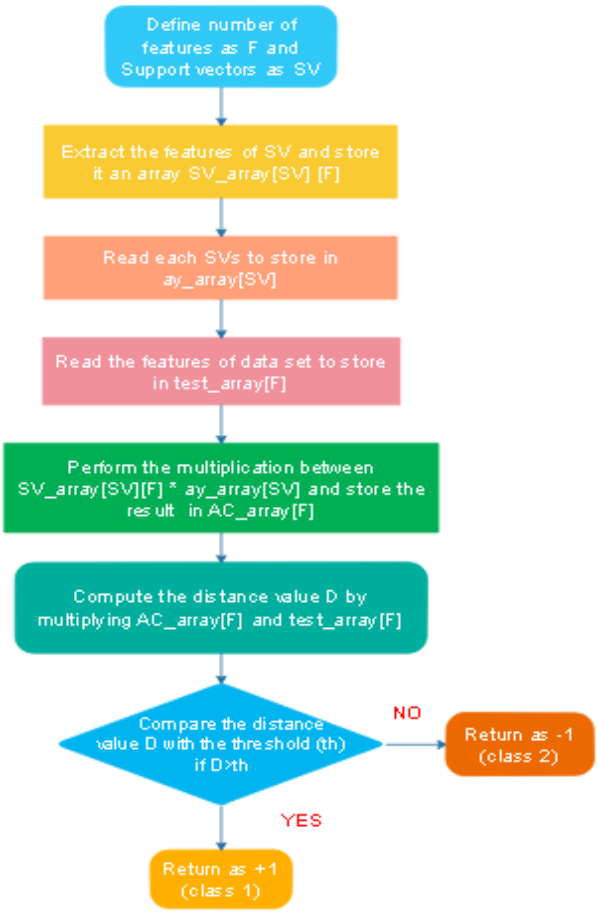

Fig. 1. Flow diagram of Proposed SVM HLS Implementation

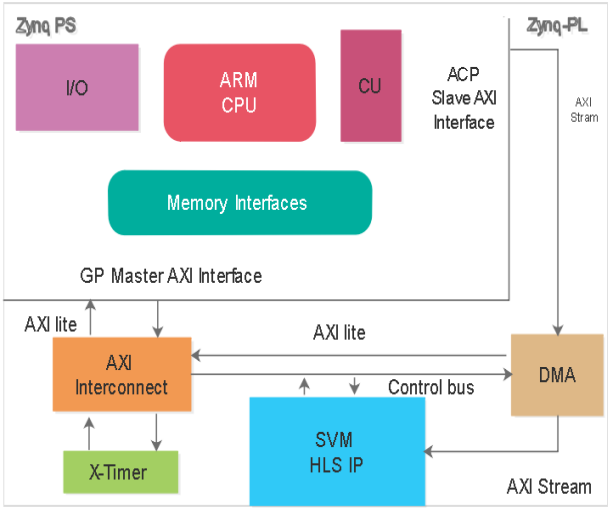

Fig. 2. The Proposed Architecture for Implementation on Zynq SoC

The top function of the parameters are optimized as being ports of the hardware IP using Vivado HLS Directives with the help of AXI4-Stream interface. This helps the two vectors to stream the data for the multiplication into the HLS IP. The AXI-lite bus is assigned to control the communication between the designed function for controlling the IP core and ARM processor in PS part of the Zynq platform as shown in Figure 2. The vectors of the double data type are replaced by float to generate an AXI4-Stream interface for streaming data between the hardware coprocessor in PL and ARM CPU in PS. Also, to support HLS stream protocol the source file is changed to a CPP file instead of C. As a result to reach high parallelism minor changes in the code is done using the Vivado HLS tool.

A good level of optimization is critical for real time application, hence to achieve high optimization the HLS tool available directives were applied for the top module. For the accumulation of multiplication process pipelining and unrolling the loop directives were assigned and tested to increase the data throughput rate. The loops in the proposed function takes around 333 clock cycles without applying any optimization directives. But with directives of pipeline and unroll it takes 169 and 204 clock cycles respectively.

C. Implementation of Zynq SoC for Proposed Embedded System Design

Once synthesizing the code and applying interfaces directives, the Vivado HLS tool is used to co-simulate and export as an RTL implementation of the HLS IP for the classifier. Then by using Vivado Design tool, integrate the exported HLS IP into a proposed system as shown in Fig. 3.

In the PL part of the proposed system Zynq coprocessor is implemented by connecting core of HLS IP with the ARM processor in PS part through Accelerator Coherency Port (ACP) using Direct Memory Access (DMA). The data transfer communication between HLS IP and ARM processor through the AXI4-Stream protocol is controlled by DMA controller IP. In the control unit the ACP is a 64-bit AXI slave interface, which provides a low latency asynchronous cache-coherent access point from the Zynq PL to the PS. Also, measuring the number of clock cycle required for the core comparison is instantiated by the AXI-Timer. At last it was exported for the Software Development Kit (SDK) after synthesizing, generating and implementing the bit tream of the proposed design for running the application on Zynq device.

D. Implementation of Zynq SoC for Proposed Software Design

To verify and test the implemented SVM classifier on the Zynq SoC, a stimulus/test bench or software program has been developed. The execution of the test bench is the responsibility of the PS ARM CPU. The required software program or test bench for testing is developed and implemented in C using the SDK tool. A proposed software program or test bench runs on the ARM processor of PS is shown in Fig. 4. The required trained parameters of the SVM model and test instance for computation are imported from three main files saved in the SD card of Zynq. The support vectors are stored in the first file, and *ay* of each SV with the value of *b* is stored in the second file. The test data are stored in the third file. Next, for further processing, all imported data are stored in three main arrays to stream to the Zynq.

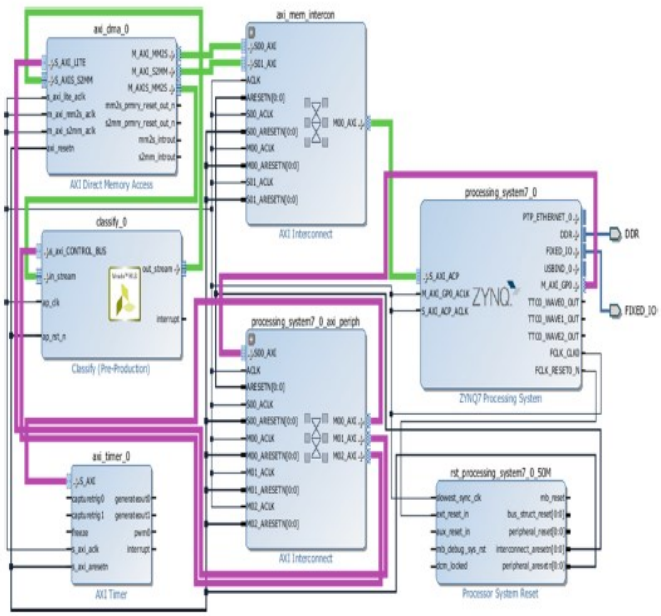

Fig. 3. Proposed system implemented on Zynq

Defined F as number of features+1 and SV+1 has support vectors

1. On the File_SVs read SVs data in byte wise
2. Store in an array of SVs_array[SV][F]
3. Repeat the first two steps for ay and b, ay_array[SV] and File_test
4. Repeat the first two steps for test_array[F] and File_test
5. Setup XTimer IP and calibrate
6. The proposed algorithm shown in Figure 1 runs on PS ARM
7. Calculate the XTimer total execution cycle
8. DMA IP initialized
9. SVM HLS IP hardware is initialized
10. SVM IP runs on the Zynq PL
11. Using DMA transfer SVs_array to the SVM IP
12. Waiting for transfer done by DMA within a loop
13. Step 11 and 12 are repeated for test_array[F] and ay_array[SV]
14. Execute SVM IP done in a loop
15. Hardware results read from SVM IP
16. DMA transfer and hardware of SVM IP running total execution time measured by XTimer
17. Compare the results of result_software and result_hardware
18. Also, compare the run time of software and hardware implementation.

Fig. 4. Zynq for running software program using proposed algorithm.

To compare the results of software and hardware implementation, the proposed SVM algorithm shown in Fig. 1 is implemented on the ARM processor and the obtained results are compared with SVM HLS IP running on hardware. The XTimer IP, counts the clock cycle running of the SVM algorithm in software/PS and hardware/PL for performance comparison. This proposed system can support adopting any trained SVM model with the parameters of the same size and feasible to achieve flexibility, generality, and adaptability.

IV. EXPERIMETAL RESULTS

A. Experimental Setup

To implement our proposed hardware design on the Evaluation Board of Zynq-7 ZC702, the Xilinx Vivado 2016.1 Design Suite is used. First using a Vivado HLS tool SVM HLS IP is developed, then it is integrated with the design Zynq SoC using the Vivado tool as shown in Figure 2. The bit stream of the designed Zynq system is generated after synthesis, place, and route for the Xilinx SDK tool. This tool helps to runs the application on onboard for classification process on Zynq.

B. Data Set

The experiment is conducted using two HSI data sets of the Airborne Visible Infra-Red Imaging Spectrometer (AVIRIS). The first data set contain pixels 145 x 145 image of southwestern part of the Indian subcontinent of Mysuru district and the second data set contains 512 x 217 of pixels image collected from Mysuru city. These data images contain 224 bands for every pixel with a scan rate of 12Hz and produce 677 pixels for every scan with the sampling rate of approximately.

In this data set only 9 spectral bands out of 224 are considered for training and classification, because of the information redundancy and water absorption. In this work, we consider 6 classes, and 346 pixels are used as test vectors for training binary SVM classifiers.

C. Synthesis Results of HLS

The proposed HLS IP design is optimized using available directives of the Vivado HLS tool. By applying different optimization directives, several experiments were conducted to improve and optimize the synthesis results. This process will help to design low complexity and friendly hardware. The assigned directives of corresponding HLS synthesis results were presented in Table I. The first column in Table 1 lists the used directives, followed by synthesis results of HLS is presented in column two, which includes design latency, throughput, and resource utilization. The HLS tool default setting synthesis results is presented in the first row of the table, which includes I/O ports interface directives mapping (AXI4 and AXI). In subsequent rows, alternative directives interface were tabulated. Also, for the used array the resource allocation is tested such as RAM and LUT. The concept of the pipeline is introduced for inner or most loops, aiming to get the optimum solution and helps the tool to make quick scheduling (Vivado Design Suite). Also, different style of array partition is applied with the directive of loop unrolling.

TABLE I
DIFFERENT DIRECTIVE S SYNTHESIS RESULTS IN PROPOSED HLS IP

Directives	Throughput	Latency	LUT	DSP	BRAM	FF
Pipeline inner loops	**19627**	**19626**	**2465**	**5**	**35**	**1258**
Interfaces	114899	114898	2429	5	33	1271
Pipeline all loops	19618	19617	9550	58	30	5467
Pipeline most loops	19618	19617	4048	10	35	2629
Unroll most loops	11601	11600	63328	135	29	13793
Unroll inner loops	**13699**	**13698**	**29975**	**135**	**28**	**13919**
Array partition Complete	32725	32724	11276	5	27	29334
Array partition Cyclic factor 2	19249	19248	3503	10	38	2117
Array partition Block factor 2	59126	59125	12921	7	38	13793
Array partition Cyclic factor 16	**12961**	**12960**	**12938**	**20**	**28**	**11123**
Array partition Cyclic factor 8	19216	19215	10027	10	40	6490

The throughput and latency of the proposed HLS IP design were significantly improved by applying various optimization directives with the basic interface directives. The latency of the trained model with the interfaces of pipeline inner loop directives 114,898 clock cycles (first row) was considerably decreased by a factor of *8x* by utilizing the extra resources. However, unroll inner loops provide a lower latency of 13,698 cycles by utilizing or allocating extra resources such as DSP slices, which is increased from 5 to 135. In the case of unroll, most loops directive gives the lowest latency of 11,600, but it is not recommended for the implementation because of its excess of LUT utilization up to 119%, which was significantly reduced in unroll inner loops to 92%. However, to moderate between the percentage of LUT utilization and reaching lower latency array partition is applied, which provides a latency of 12960 with the resource utilization of 20 DSP.

By analyzing the different directives and their performance concerning latency, throughput, and resource utilization, the pipeline inner loops directives have considerably decreased the latency with some increases in the utilization of resources. Therefore, the pipelined design is considerably best solution for low power consumption and area (a small number of DSP and LUT) with high throughput and reduced latency.

D. Results of Hardware Implementation

The Vivado tool has been used to integrate the designed HLS SVM IP into the Zynq SoC. Based on the synthesis results of HLS presented in the previous section, the designs which are exhibits better results were selected for the implementation of Zynq SoC. The designs unrolled, pipelined, and array partitioned shown in Table 1 were selected as the three best solutions for the implementation (with an operating frequency of 100 MHz for PL/FPGA and 666.67 for ARM CPU) to balance the trade-off between low cost/area and high performance.

1) Analysis of Resources Utilization

The resource utilization of FPGA for the three implemented designs on Zynq is tabulated in Table II. The available hardware resources are listed in the first column and in further columns utilization of resources is listed. In the last column, the target device available hardware resources are listed. In all three design implementation, the percentage of resource utilization is very low, hence it shows the area of saving in implementation.

TABLE II
RESOURCE UTILIZATION OF IMPLEMENTED DESIGNS ON ZYNQ SOC

On-Chip Component	Available	Unrolled	Pipelined	Array Partitioned
Slice LUTs	53200	12808	2579	8111
Slice FF Registers	106400	13830	2898	9009
Memory LUT	17400	161	204	582
BRAM	140	16.5	20	16.5
DSP48	220	135	5	20

2) Analysis of Power Consumption

The power consumption of all three design implementation has been analysed using the Vivado tool and corresponding results are reported in Table III. From the results, it is clear that the power consumption of all three design is a considerably small value, hence it will meet the real-time constraints of an embedded system. The pipelined design consumed the least power of 1.758 watts, compared to designs of unrolled and array partitioned.

TABLE III
POWER CONSUMPTION OF THREE DESIGNS

#	Unrolled	Pipelined	Array Partitioned
On-Chip Power (Watts)	2.125	1.758	1.842

3) Analysis of Processing Time

The computing time of designed SVM code on the hardware-implemented in PL part and ARM processor in the PS part of the Zynq SoC is exploited using AXI-Timer IP core shown in Fig. 2. The operating frequency of 100 MHz is used to compute the processing time of both the PS part of the ARM processor and the PL part of hardware implementation. Accordingly, the processing time of designs is tabulated in Table IV.

TABLE IV
PROCESSING TIME COMPARISON OF THREE DESIGNS

#	FPGA	ARM	Optimized FPGA/ARM
Clock Cycles	2815	28968	10.29
Pipelined (µs)	11.26	43.45	3.86
Unrolled (µs)	14.77	309.47	20.95
Array Partitioned(µs)	14.76	309.31	20.96

4) Analysis of Classification Accuracy

An accuracy assessment is an important parameter to analyses the effectiveness of designed HLS SVM IP. The classification results of hardware implementation are compared with software implementation to assess the performance of the classifier. To analyze the performance of developed HLS SVM IP, the confusion/error matrix needs to construct and it can be used to calculating the overall classification accuracy. Therefore, for each design, the confusion/error matrix is developed for the performance analysis. The confusion matrix for the pipelined design is constructed with the 6 classes and 346 test vectors. The test vector is compared pixel by pixel with the remotely sensed data, then agreement and disagreement are arranged in the cells of the error matrix. The total column in the matrix gives how many reference test data are compared with each class and the row total gives the how many pixels are classified to each class. Correctly classified pixels can be found by adding diagonal elements of the error matrix. Table V shows a typical confusion matrix of 6 classes and 346 reference test vectors of HLS SVM IP.

Table VI shows a confusion matrix of software implementation for 6 classes and 346 reference test vectors**.** The classification accuracy rate of our hardware implementation has got 97.61% and the corresponding software implementation has achieved 98.57% is tabulated in Table VII. The difference of overall classification accuracy is within 1% for the same data set, hence our implemented HLS SVM IP can be used for real-time classification without compromising with accuracy rate.

TABLE V
CONFUSION MATRIX OF HLS SVM IP.

Classified Data / Classes	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Row Total
Class 1	52	1	0	0	0	0	53
Class 2	0	44	0	0	0	0	44
Class 3	0	0	32	0	1	0	33
Class 4	0	0	1	27	0	0	28
Class 5	0	0	0	1	29	0	30
Class 6	1	0	0	0	0	21	22
Column Total	53	45	33	28	30	21	210

TABLE VI
CONFUSION MATRIX OF SOFTWARE IMPLEMENTATION.

Classified Data / Classes	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Row Total
Class 1	52	0	0	0	0	0	52
Class 2	0	45	0	0	0	0	45
Class 3	0	0	33	0	1	0	34
Class 4	0	0	0	27	0	0	27
Class 5	0	0	0	1	29	0	30
Class 6	1	0	0	0	0	21	22
Column Total	53	45	33	28	30	21	210

TABLE VII
ACCURACY COMPARISON OF HLS SVM IP AND SOFTWARE IMPLEMENTATION

#	Training Data set	Support Vectors	Accuracy
HLS SVM IP	210	346	97.61 %
Software Implementation	210	346	98.57 %

TABLE VIII
COMPARISON OF PIPELINED DESIGN WITH OTHER RESEARCHERS

Relate Work	Ours	Sen Ma et al (2018) [36]	Kumar et al (2017) [37]	Qasaimeh et al (2015) [38]	Paoletti et al. (2020) [35]	Jiang et al (2017) [33]	Kyrku et al (2016) [34]	Ningma et al (2016) [27]	Wang et al (2016) [30]	Nitish et al (2017) [32]	Abelardo et al (2019) [31]
Slice LUTs	2,579 /53200	NA	2443/ 69120	16,138/ 69120	NA	14,028/5 3200	35,532/ 92152	NA	234666/ 262400	62,522/ 218,600	11122/ 53200
Slice FF Registers	2,898/ 106400	NA	2332/ 69120	7,924 /69120	NA	21,305/1 06400	20,153/ 184304	NA	443688 / 524800	81,135/ 437,200	15225/ 106400
BRAM	20/140	NA	19/148	60/148	NA	106/140	256/268	NA	1715/ 2567	157/545	8/140
DSP48	5/220	NA	24/64	53/64	NA	152/220	59/180	NA	1680/ 1963	111/900	34/220
Number of SVs	346	NA	8	98	NA	94	122	50	100	NA	NA
Kernal	Linear	NA	Linear	Linear	Linear	Linear	Linear	Linear	Linear	NA	Linear
Classifier	SVM	MLC	SVM	SVM	SVM	SVM	SVM	SVM	SVM	Haar cascaded	SVM
Hardware Architecture	HLS-based Pipelined	HLS based pipelined	HDL based	HDL based	GPU based	MATLAB Simulink	MAT LAB	HDL based	HDL based	HLS-based	HLS based
Frequency (MHz)(	100	100	100	100	200	100	70	100	100	100	200
Power (W)	1.758	NA	N/A	N/A	NA	NA	NA	3.9	26.3	NA	2.04
Processing Time	11.26 µs	3.0 s	NA	25ms	89.62ms	0.02ms	40 FPS	25.8 µs	0.99 s	38 ms	2.20ms
Application	remote sensing	Big image data	Pattern recognition system	Object detection	remote sensing	Industrial applications	Face detection	remote sensing	remote sensing	Face detection	Medical Images
Accuracy	97.61 %	NA	NA	97.54 %	84.25%	97%	80 %	97.7%	97.8%	NA	NA
FPGA	Zynq-7000	Kintex - AC-510	Virtex 5-LX110T	Virtex 5-LX110T	GPU	Zynq-7000	Spartan-6	ZynqXC7Z020	Altera Stratix V	ARM CPU and Zynq-7000	Zynq-7000

TABLE IX
PROCESSING TIME, POWER CONSUMPTION, AND ARCHITECTURE DIFFERENCE OF DIFFERENT PLATFORM

#	Ours	NingMa et al (2016) [27]	Wang et al (2016)	Kumar et al (2017) [37]	Nitish et al (2017) [32]	NingMa et al (2016) [27]	NingMa et al (2016) [27]	NingMa et al (2016) [27]	NingMa et al (2016) [27]	Paoletti et al. (2020) [35]
Hardware Architecture	HLS based	HDL based	HDL based	HDL based	HLS based	DSP	ARM	PC	Xeons	GPU based
FPGA	Zynq 7000	ZynqXC7Z020	Altera Stratix V	Virtex 5-LX110T	Zynq-7000	NA	NA	NA	NA	NA
Application	Remote sensing	Remote sensing	Remote sensing	Pattern recognition system	Face detection	Remote sensing	Remote sensing	Remote sensing	Remote sensing	Remote sensing
Power (W)	1.758	3.9	26.3	NA	NA	16.0	3.3	103.8	95	NA
Time	11.26 µs	25.8 µs	0.99S	NA	38ms	65.7 µs	1321.2µs	216.3 µs	14.1 µs	89.62ms
Energy(mJ/pixel)	0.019	0.1	1	NA	NA	1.1	4.3	22.4	1.33	NA

V. Comparison and Discussion

This work focus on developing the optimized hardware for SVM classifier to realize the real-time classification system for remote sensing application. To compare the performance of our work, the result of different platform implementations have been considered in different applications.

The pipelined design of our implementation has been considered for the comparison with the related work is shown in Table VIII. Also, our HLS based pipelined design is compared with the work related to Hyperspectral data classification on a different platform is shown in Table IX.

To evaluate the performance of the proposed design model, we compare the processing time and power consumption of various platforms as shown in Table IX. In (Ning ma et al 2016) the traditional HDL based implementation on Zynq-XCz020 has generated the delay of 25.8 µs/pixel and consumes the power of 3.9 Watts, which is twice compared to our HLS based design. Also, the other processors such as DSP and ARM have been used in microsatellites experimental for low-cost space missions. Therefore, we compare our HLS SVM IP with these processors and commercially available processors in terms of processing time and power consumption. The processors consider for result analysis includes DSP TMS320C6678, ARM Cortex A9, and Xeons E5-2650. An analysis of power consumption and processing time from Table 9 shows that our pipelined design implementation on Zynq-7000 has to offer extremely better performance as compared to other platforms. The pipelined design on Zynq-7000 gets the speed up of 2x, 5x, 100x, 20x compared to Zynq-XC7Z020, DSP, ARM, and PC respectively. Also, the energy-saving of our pipelined design extremely good as compared to other implementation listed in Table 9. The power consumption of our pipelined design has consumed the least power as compared to other designs, therefore our proposed design is well suited for onboard classification of remotely sensed data and fulfils the requirement of real-time classification.

CONCLUSIONS

This paper proposes an implementation of an SVM classifier on FPGA using hardware/software co-design and an embedded SoC is realized for onboard classification of hyperspectral data on the Zynq 7000 SoC using HLS design methodology. Our Zynq implemented system satisfies the constraints of an embedded system, with high performance in terms of power consumption, area, and speed. In this work, the utilization of HLS design and directives (optimization techniques) helps in achieving a high classification accuracy rate of 97.61%. The proposed system is cost-effective since it uses only 2.7% of slices and power consumption of 1.7 watts. Also the processing time our proposed system is 11.3 *µs,* which is extremely better compared to existing work in the literature. These results show the proposed system meets the requirements of real-time classification and it is suited for onboard hyperspectral image classification system.

Acknowledgment

The authors would like to thank, JSS Academy of Technical Education, Bengaluru, Visvesvaraya Technological University (VTU), Belagavi and vision group on science and technology (VGST) Karnataka Fund for Infrastructure Strengthening in Science & Technology Level –2 (JSSATEB) for all the support and encouragement provided by them to take up this research work and publish this paper.

References

1. Hennes Henniger, Stefan Seyfarth, and Erhard Diedrich, “Analysis and Comparison of New Downlink Technologies for Earth Observation Satellites”, *Radio Engineering,* vol. 25, No.1, pp.1-6, 2016. http://doi.org/10.13164/re.2016.0001
2. Montenegro S, Rodionov I, Rodionov A, and Fedounin E, “Hyperspectral monitoring data processing”, *International Symposium on Small Satellites for Earth Observation*, ISBN 3-89685 569-7 4, 2003.
3. S. Lucana, M. Enrico, V. Raffaele, “Highly-Parallel GPU Architecture for Lossy Hyperspectral Image Compression”, *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing (J-STARS),* vol. 6, No. 2, pp. 670-68, 2013. https://doi.org/10.1109/JSTARS.2013.2247975
4. S Bernabe, S. Lopez, A. Plaza, “FPGA Design of an Automatic Target Generation Process for Hyperspectral Image Analysis”, *IEEE 17th International Conference on Parallel and Distributed Systems*, pp. 1010-1015, 2011. http://doi.org/10.1109/ICPADS.2011.64
5. Dequan Liu, Guoqing Zhou, Jingjin Huang, Rongting Zhang, Lei Shu, Xiang Zhou, and Chun Sheng Xin, “On-Board Geo referencing Using FPGA-Based Optimized Second-Order Polynomial Equation”, *Remote Sens.,* vol. 11, pp.1-28, 2019. http://doi.org/10.3390/rs11020124
6. D. Lu, and Q. Weng, “A survey of image classification methods and techniques for improving classification performance”, *International Journal of Remote Sensing,* Vol. 28, No. 5, pp. 823–870, 2007. https://doi.org/10.1080/01431160600746456
7. J. Nayak, B. Naik, and H. Behera, “A Comprehensive Survey on Support Vector Machine in Data Mining Tasks: Applications & Challenges,” *International Journal of Database Theory and Application,* vol. 8, pp. 169-186, 2015. http://doi.org/10.14257/ijdta.2015.8.1.18
8. G. Camps Valls, L. Bruzzone, “Kernel-based methods for hyperspectral image classification*”, IEEE Transaction on Geoscience and Remote Sensing,* vol. 43, No. 6, pp.1351-1362, 2005. https://doi.org/10.1109/TGRS.2005.846154
9. S. M. Afifi, H. GholamHosseini, and R. Sinha, “Hardware Implementations of SVM on FPGA: A State-of-the-Art Review of Current Practice”, *International Journal of Innovative Science, Engineering & Technology,* vol. 2, pp. 733-752, 2015.
10. Mahendra HN, Mallikarjunaswamy S, Siddesh GK, Komala M, Sharmila N, “Evolution of real-time onboard processing and classification of remotely sensed data”, *Indian Journal of Science and Technology,* vol. 13, pp. 2010-2020, 2020. https://doi.org/10.17485/IJST/v13i20.459
11. Shradha Gupta, Sumeet Saurav, Sanjay Singh, Anil K Saini, Ravi Saini, “VLSI Architecture of Exponential Block for Non-Linear SVM Classification”, *IEEE International Conference on Advances in Computing, Communications and Informatics*, pp.128-132, 2015. https://doi.org/10.1109/ICACCI.2015.7275662
12. Sumeet Saurav, Ravi Saini, and Sanjay Singh, “FPGA Based Implementation of Linear SVM for Facial Expression Classification”, *IEEE International Conference on Advances in Computing, Communications and Informatics*, pp. 766-773, 2018. https://doi.org/10.1109/ICACCI.2018.8554645
13. Markos Papadonikolakis, Christos-Savvas Bouganis, “A Novel FPGA-based SVM Classifier”, *IEEE International Conference on Field Programmable Technology*, pp.283-290, 2010. https://doi.org/10.1109/FPT.2010.5681485
14. Yiyue Jiang, Kushal Virupakshappa and Erdal Oruklu, 2017, “FPGA Implementation of a Support Vector Machine Classifier for Ultrasonic Flaw Detection”, *IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, pp. 180-183, Aug. 2017. https://doi.org/10.1109/MWSCAS.2017.8052890
15. Kevin M. Irick, Michael DeBole, Vijaykrishnan Narayanan, and Aman Gayasen, “A hardware efficient support vector machine architecture for FPGA”, *IEEE Symposium on Field-Programmable Custom Computing Machines*, pp. 304-305, Apr. 2008. https://doi.org/10.1109/FCCM.2008.40

16. Papadonikolakis, M. and Bouganis C S, "Novel cascade FPGA accelerator for sup- port vector machines classification", *IEEE Transactions on Neural Networks and Learning Systems,* pp.1040-1052, 2012. https://doi.org/10.1109/TNNLS.2012.2196446
17. Kyrkou, C. and Theocharides, T, "SCoPE: Towards a systolic array for SVM object detection", *IEEE Embedded Systems Letters,* pp. 46-49, 2009. http://doi.org/10.1109/LES.2009.2034709
18. Cadambi S., Igor D, Venkata J, Murugan S, Eric C, Srimat T, H.P. Graf, "A massively parallel FPGA-based coprocessor for support vector machines", *IEEE Symposium on Field Programmable Custom Computing Machines*, pp.115-122, Apr. 2009. https://doi.org/10.1109/FCCM.2009.34
19. Berberich M, and Doll K, "Highly flexible FPGA-architecture of a support vector machine", *In: MPC workshop*, pp. 25–32, 2014.
20. C. Kyrkou and T. Theocharides, "SCoPE: Towards a Systolic Array for SVM Object Detection," *IEEE Embedded Systems Letters,* vol. 1, pp. 46-49, 2009. http://doi.org/10.1109/LES.2009.2034709
21. Christos Kyrkou, and T Theocharides, "A Parallel Hardware Architecture for Real-Time Object Detection with Support Vector Machines," *IEEE Transactions on Computers,* vol. 61, pp.831-842, 2012. https://doi.org/10.1109/TC.2011.113
22. M. Pietron, M. Wielgosz, D. Zurek, E. Jamro, and K. Wiatr, "Comparison of GPU and FPGA Implementation of SVM Algorithm for Fast Image Segmentation," *Architecture of Computing Systems–ARCS*, -Springer, pp. 292-302, 2013. https://doi.org/10.1007/978-3-642-36424-2_25
23. Vivado High-Level Synthesis, Available: http://www.xilinx.com/products/designtools/vivado/integration/esl-design.html
24. Vivado Design Suite Available: http://www.xilinx.com/products/design-tools/vivado.html
25. Zynq-7000 All Programmable SoC. Available: http://www.xilinx.com/products/silicondevices/soc/zynq-7000.html.
26. Boser, Guyon, Vapnik, "Support Vector Machines (SVMs)", *Intelligent Systems and Signal Processing in Power Engineering,* pp. 161-226, 1992.
27. Ning Ma, Shaojun Wang, Syed Mohsin Ali, Xiuhai Cui and Yu Peng, "High Efficiency On-Board Hyperspectral Image Classification with Zynq SoC", *MATEC Web of Conferences,* 2016. https://doi.org/10.1051/matecconf/20164505001
28. Mahendra H N, Shivakumar B R, Praveen J, "Pixel-based Classification of Multispectral Remotely Sensed Data Using Support Vector Machine Classifier", *International Journal Of Innovative Research In Electrical, Electronics, Instrumentation And Control Engineering,* vol. 3, pp. 94-98, Apr. 2015. http://doi.org/10.1109/IACC.2016.20
29. Mahendra H N, Mallikarjunaswamy S, Rekha V, Puspalatha V, Sharmila N, "Performance Analysis of Different Classifier for Remote Sensing Application", *International Journal of Engineering and Advanced Technology,* vol. 9, pp. 7153-7158, Oct. 2019. https://doi.org/10.35940/ijeat.a1879.109119
30. Shaojun Wang, Xinyu Niu, Ning Ma, Wayne Luk, Philip Leong, and Yu Peng, "A Scalable Dataflow Accelerator for Real Time Onboard Hyperspectral Image Classification" *Applied Reconfigurable Computing: 12th International Symposium*, ARC 2016 http://doi.org/10.1007/978-3-319-30481-6_9
31. Abelardo Baez, Himar Fabelo, Samuel Ortega, Giordana Florimbi, Emanuele Torti , Abian Hernandez, Francesco Leporati, Giovanni Danese, Gustavo M. Callico and Roberto Sarmiento, "High-Level Synthesis of Multiclass SVM Using Code Refactoring to Classify Brain Cancer from Hyperspectral Images", MDPI Electronics, 2019. https://doi.org/10.3390/electronics8121494
32. Nitish Srivastava, Steve Dai, Rajit Manohar, and Zhiru Zhang, "Accelerating Face Detection on Programmable SoC Using C-Based Synthesis", *Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 195-200, February 2017. https://doi.org/10.1145/3020078.3021753
33. Yiyue Jiang, Kushal Virupakshappa and Erdal Oruklu, "FPGA Implementation of a Support Vector Machine Classifier for Ultrasonic Flaw Detection", *IEEE 60th International Midwest Symposium on Circuits and Systems* (MWSCAS), 2017 https://doi.org/10.1109/MWSCAS.2017.8052890
34. Christos Kyrkou, Christos-Savvas Bouganis, Theocharis Theocharides, and Marios M. Polycarpou, "Embedded Hardware-Efficient Real-Time Classification With Cascade Support Vector Machines", *IEEE Transactions On Neural Networks And Learning Systems*, Vol. 27, No. 1, January 2016. https://doi.org/10.1109/TNNLS.2015.2428738
35. Mercedes E. Paoletti , Juan M. Haut, Xuanwen Tao, Javier Plaza Miguel and Antonio Plaza, "A New GPU Implementation of Support Vector Machines for Fast Hyperspectral Image Classification", *MDPI Remote sensing Journal*, Feb. 2020. https://doi.org/10.3390/rs12081257
36. Sen Ma, Xuan Shi, and David Andrews, "Parallelizing maximum likelihood classification (MLC) for supervised image classification by pipelined thread approach through high-level synthesis (HLS) on FPGA cluster", *BIG EARTH DATA, Taylor & Francis Group* VOL. 2, NO. 2, pp. 144–158, 2018. https://doi.org/10.1080/20964471.2018.1470249
37. Santosh Kumar, Manikandan J and VK Agrawal, "Hardware Implementation of Support Vector Machine Classifier using Reconfigurable Architecture", *IEEE International Conference on Advances in Computing, Communications and Informatics* (ICACCI), 2017. https://doi.org/10.1109/ICACCI.2017.8125814
38. Murad Qasaimeh, Assim Sagahyroon, and Tamer Shanableh, "FPGA-based Parallel Hardware Architecture for Real-Time Image Classification", *IEEE Transactions on Computational Imaging*, Volume 1, Issue 1, March 2015. https://doi.org/10.1109/TCI.2015.2424077