An Optimized Hardware System on Chip for A Support Vector Machine Classifier: A Case Study on Melanoma Detection

Shereen Moataz Afifi

A thesis submitted to
Auckland University of Technology
in fulfilment of the requirements for the degree of
Doctor of Philosophy (PhD)

2018

School of Engineering, Computer and Mathematical Sciences

Table of Contents

List of Figures

List of Tables

List of Abbreviations

FPGA	Field-Programmable Gate Array
SVM	Support Vector Machine
CAD	Computer Aided Diagnosis
HLS	High-Level Synthesis
IP	Intellectual Property
DPR	Dynamic Partially Reconfiguration
RM	Reconfigurable Module
RTR	Run-Time Reconfiguration
RP	Reconfigurable Partition
SV	Support Vector
QP	Quadratic Programming
SMO	Sequential Minimal Optimization
HDL	Hardware Description Language
HPC	High Performance Computing
GPU	Graphics Processing Unit
CPU	Central Processing Unit
SoC	System on Chip
GPP	General-Purpose-Processor
CLB	Configurable Logic Block
FSM	Finite-State Machine
CORDIC	Coordinate Rotation Digital Computer
HWS	Hybrid Working Set
SOPC	System on Programmable Chip
LS-SVM	Least Square Support Vector Machine
SOLE	Solving Linear Equations Module
KNN	K-Nearest Neighbour
PR	Partially Reconfiguration
LUT	Look-Up Table
CSD	Canonic Signed Digit
CSE	Common Sub-expression Elimination
SPW	Signal Processing Workbench
HOG	Histograms of Oriented Gradients
IPPro	Image Processing Processor
GFLOPS	Giga Floating Point Operations per Second
GOPS	Giga Operations per Second
LBP	Local Binary Pattern
SIFT	Scale-Invariant Feature Transform
BoF	Bag of Features
PL	Programmable Logic
PS	Programmable System
XDC	Xilinx Design Constraints
II	Initiation Interval
CU	Control Unit

ACP	Accelerator Coherency Port
DMA	Direct Memory Access
BB	Black-Box
IPI	IP Integrator
DCP	Design Checkpoint
PCAP	Processor Configuration Access Port
SDK	Software Development Kit
ASIC	Application Specific Integrated Circuit
VHDL	Very High Speed Integrated Circuit Hardware Description Language
RTL	Register-transfer level
BRAM	Block Random Access Memory
DSP	Digital Signal Processor
ARM	Advanced RISC (Reduced Instruction Set Computer) Machine
MHz	Megahertz
I/O	Input/Output
SD	Secure Digital
FF	Flip Flop
BUFG	Generic Global Clock Buffer
AXI	Advanced Extensible Interface
JTAG	Joint Test Action Group
RBF	Radial Basis Function

Attestation of Authorship

"I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning."

Signed: .

Date: .29/10/2018……..

Acknowledgements

Thank Allah at first, then I would like to express my sincere gratitude to my primary supervisor A/P Hamid GholamHosseini, for the continuous support of my PhD study and related research, for his patience, motivation, advice, and guidance as well as for giving me extraordinary experiences and enriching my research skills throughout the work. He provided me unflinching support and encouragement in various ways. His guidance helped me in all the time of research, different research activities and writing of this thesis as well as publications.

I would like to express my deepest thanks to my second supervisor Dr. Roopak Sinha, for his continuous support and advice to this research project. It was a privilege to have two experienced and supportive supervisors. Their advice on research as well as on my career have been priceless.

I am also grateful to Auckland University of Technology and the School of Engineering, Computer and Mathematical Sciences, particularly for granting me the Vice Chancellor Doctoral Scholarship and for the supportive and friendly staff and administrative team. In addition, I gratefully acknowledge the Graduate Research School for providing different research activities and workshops as well as miscellaneous events.

A special thanks to my lovely parents and family, especially my sincere mother who helped and supported me and my children a lot. I would like also to thank all my friends and colleagues who supported me through this long journey of study and my life in general.

Finally, I would like to thank everyone who was important to the successful realization of this thesis, as well as expressing my apology for not mentioning each one personally.

List of Publications

1. Shereen Afifi, Hamid GholamHosseini, and Roopak Sinha, "Dynamic Hardware System for Cascade SVM Classification of Melanoma," Neural Computing and Applications, pp. 1-12, August 2018.
2. Shereen Afifi, Hamid GholamHosseini, and Roopak Sinha, "FPGA Implementations of SVM Classifiers: A Review," Submitted in Neural Computing and Applications Journal in July 2018.
3. Shereen Afifi, Hamid GholamHosseini, and Roopak Sinha, "A System on Chip for Melanoma Detection Using FPGA-based SVM Classifier," Submitted in Microprocessors and Microsystems: Embedded Hardware Design Journal in July 2018.
4. Shereen Afifi, Hamid GholamHosseini, and Roopak Sinha, "SVM Classifier on Chip for Melanoma Detection," In 2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC'17), Jeju Island, Korea, July 2017.
5. Shereen Afifi, Hamid GholamHosseini, and Roopak Sinha, "A Low-Cost FPGA-based SVM Classifier for Melanoma Detection," In 2016 IEEE EMBS Conference on Biomedical Engineering and Sciences (IECBES), pp. 631-636, Malaysia, December 2016.
6. Shereen Afifi, Hamid GholamHosseini, and Roopak Sinha, "Hardware Acceleration of SVM-based Classifier for Melanoma Images," In Image and Video Technology–PSIVT 2015 Workshops: RV 2015, GPID 2013, VG 2015, EO4AS 2015, MCBMIIA 2015, and VSWS 2015, Auckland, New Zealand, November 23-27, 2015. Revised Selected Papers, F. Huang and A. Sugimoto, Eds., ed Cham: Springer International Publishing, 2016, pp. 235-245.
7. Shereen M. Afifi, Hamid GholamHosseini, and Roopak Sinha, "Hardware Implementations of SVM on FPGA: A State-of-the-Art Review of Current Practice," International Journal of Innovative Science, Engineering &Technology (IJISET), Vol. 2 Issue 11, pp. 733-752, November 2015.
8. Shereen M. Afifi, François Verdier, Cécile Belleudy, "Power Estimation Method Based on Real Measurements for Processor-based Designs on FPGA," In International Conference on Computational Science and Computational Intelligence, CSCI'14, USA, pp. 260-263, March 2014.
9. Shereen M. Afifi, Ayman M. Wahba, Abd-Elmoneim Wahdan, "Memory design for multimedia applications," M.Sc. Thesis, Arab Academy for Science, Technology and Maritime Transport, College of Engineering and Technology, Cairo, Egypt, March 2012.
10. Shereen M. Afifi, Ayman M. Wahba, Abd-Elmoneim Wahdan, "Accelerated access to visual data in multimedia applications," In International Conference on Computer Engineering and Systems, ICCES 2010, Cairo, Egypt, pp. 172-177, December 2010.
11. Shereen M. Afifi, Ayman M. Wahba, Abd-Elmoneim Wahdan, "Memory design for multimedia applications," In International Conference on Computer Design, CDES 2010, World Comp'10, USA, pp. 189-195, July 2010.

Abstract

Support Vector Machine (SVM) is a robust machine learning model used for efficient classification with high accuracy. SVM is widely utilized for online classification in various embedded applications. However, implementing the SVM classification algorithm for an embedded system or application is challenging, due to intensive and complicated computations required. This increases the importance of implementing SVM on hardware platforms for achieving high performance computing at low cost and power consumption.

Field-Programmable Gate Array (FPGA) is a powerful parallel processing reconfigurable device that is widely used for achieving essential performance of embedded systems, while effectively utilizing hardware resources, offering low cost and low power consumption. Accordingly, FPGA is a promising hardware platform for implementing an efficient embedded SVM classification system, while achieving vital embedded system constraints.

SVM has shown high accuracy for classifying melanoma (skin cancer) clinical images within a computer-aided diagnosis system used by dermatologists to detect melanoma early and save lives. This research aims to develop an optimized FPGA-based SVM classifier to be embedded within a low-cost handheld medical scanning device that runs an embedded SVM-based diagnosis system dedicated to early detection of melanoma in primary care. We aim to consider meeting significant constraints of embedded systems, while achieving efficient classification with high accuracy rate.

A hardware/software co-design for implementing an SVM classifier onto FPGA is proposed to realize melanoma detection on a chip. This SVM implementation achieves efficient melanoma classification on a recent FPGA-based hybrid platform “Zynq SoC” designed using the latest UltraFast High-Level Synthesis design methodology. The hardware implementation results demonstrate classification accuracy of 97.9% and a significant hardware acceleration rate of up to 37x with only 2.7% resource utilization and 1.69 watts for power consumption.

Furthermore, a scalable multi-core architecture is proposed to achieve multi-purpose classification on a single chip/device, which has been validated with a 2-stage cascade classifier implementation with accuracies of 98 % and 73%, to enhance melanoma detection. A simple hardware-friendly design is proposed for the building SVM core of the multi-core

architecture, aiming to reduce hardware complexity and optimize implementation results for achieving an efficient classification performance.

A novel dynamic hardware system is also proposed for implementing a cascade SVM classifier on FPGA for early melanoma detection. The hardware implementation results are optimized by using the powerful dynamic partial reconfiguration technology, where very low resource utilization of 1% slices and power consumption of 1.5 watts are achieved.

The implemented SVM classification systems on Zynq SoC using the proposed hardware designs have shown the least power consumption results among other related implementations, in addition to significantly low hardware resource utilization and processing time with significant speedups and high classification accuracy rates at low cost. Consequently, the implemented Zynq systems meet crucial embedded system constraints of high performance and low cost, resource utilization and power consumption, while achieving efficient classification with high classification accuracy, which promises realization of a cost- and energy-efficient handheld medical scanning device for early detection of melanoma.

CHAPTER 1 Introduction

1.1 Research Motivation

Computer Aided Diagnosis (CAD) systems have been widely used in clinical settings as diagnostic tools to support detection of a variety of cancers. Melanoma is considered the most dangerous form of skin cancer, which is responsible for the majority of skin cancer related deaths. New Zealand and Australia have the highest rates of melanoma in the world. Early diagnosis of melanoma could help in dramatically decreasing morbidity, mortality and treatments costs [1]. Accordingly, dermatologists are recently using image-based CAD systems as diagnostic tools to help them in decision-making and detecting melanoma at an early stage. However, such tools are very costly and only available at the dermatology clinics.

There are some smartphone/mobile applications available in the market, which are mainly used for storing images and self-monitoring. Some of these applications provide image analysis to be shared with a dermatologist for reviewing and diagnosis. Few applications provide a risk assessment that shows values of different parameters as the ABCDEs to be used for self-assessment. But, the performance of these apps in assessing melanoma risk is highly variable. In addition, none of these apps have been validated for diagnostic accuracy or utility using established research methods. So, reliance on these apps instead of medical consultation can delay the diagnosis of melanoma and harm users. Also, smartphones are not well designed for such complicated diagnosis system and sensitive application like melanoma detection. Therefore, a low-cost handheld device dedicated to early detection of melanoma is required to support primary healthcare providers.

Nevertheless, developing such devices is very challenging because of the complicated computations required by the image analysis and classification algorithms within the embedded CAD system. Also, embedded system development requires meeting challenging constraints such as real-time, low cost, limited resources and low power consumption. Parallel processor systems could be used for accelerating such complicated computations, but they are expensive and energy-intensive systems. Special purpose hardware (reconfigurable hardware) has been widely used for accelerating computations and achieving High Performance Computing (HPC) with low cost and power consumption [2]. Thus, to overcome these challenges, recent hardware advances and

technologies in reconfigurable computing should be exploited to achieve cost- and energy-efficient embedded system with high performance.

Field-Programmable Gate Array (FPGA) is a robust highly parallel processing reconfigurable device that is widely used for achieving essential performance of embedded systems, while effectively utilizing hardware resources with low cost and power consumption [3]. Interestingly, FPGAs have recently demonstrated significant performance with various applications, which outperformed other comparable platforms such as general-purpose processors and Graphics Processing Units (GPUs) [4-6]. Subsequently, FPGA is a promising hardware platform for implementing algorithms with inherent parallelism required for developing an efficient embedded CAD system for melanoma detection, offering HPC with more flexibility at low cost, while meeting challenging embedded system constraints.

1.2 Problem Statement

A reliable handheld medical scanning device featured with high performance and low-cost processing unit is required for enhancing early detection of melanoma at the primary healthcare providers, where the demand is strong in New Zealand and worldwide to decrease the mortality rate. Developing such a special device that runs complicated algorithms within an embedded CAD system is very challenging because of the critical performance, area, power, and cost constraints required to develop an embedded system.

1.3 Scope and Objectives of this Research

The CAD system for melanoma detection typically consists of four main stages including image pre-processing, segmentation, features extraction, and classification as depicted in Figure 1-1 [7]. Regarding the development of such a CAD system, the final classification stage is considered to be the most compute-intensive block compared to other stages, which needs hardware/FPGA implementation. Accordingly, this research focuses on the hardware implementation of the classification stage on FPGA, aiming to reach an efficient embedded system with high performance and low cost.

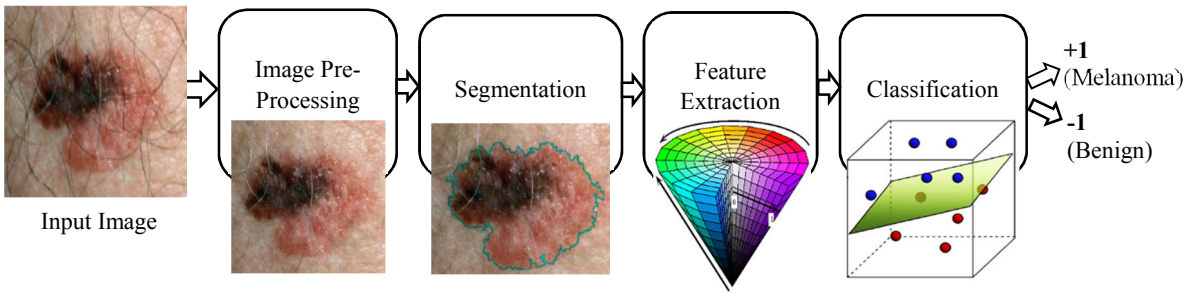

Figure 1-1. The CAD system structure.

This research is based on previous work of our research group, where suitable algorithms have been established for software-based development of different stages of the CAD system, targeting efficient melanoma detection [7]. A melanoma dataset collected from available web resources is utilized for testing and evaluation of the CAD system, which consists of benign and malignant melanoma images. Regarding the final classification stage, the Support Vector Machine (SVM) classifier showed better accuracy results for classification and diagnosis of melanoma, which is based on experimental results and performance comparisons of some tested classifiers [7]. Therefore, the SVM classifier is considered for melanoma classification in this research, for its high classification accuracy with medical images and also being a powerful machine learning feature for other applications. SVM classifier is a robust supervised machine learning tool which is widely used for different classification problems and applications, offering efficient and high classification accuracy rates.

A growing interest exists for exploiting SVMs in many embedded detection systems and various image processing applications. The SVM model is computationally expensive and time-consuming especially for large-scale problems, which raises a vital need for hardware acceleration. While software implementations of SVM produce high accuracy rates, they cannot efficiently meet embedded system constraints. This has motivated plethora of research towards implementing and accelerating SVM in hardware.

Some existing research works aimed to implement the SVM model using FPGAs by exploiting the inherent parallelism of the SVM algorithm. Different hardware architectures/designs offered HPC with a high level of parallelization. From reviewing existing implementations in the literature, we concluded that the main challenge is the difficulty of meeting important embedded system constraints of high performance, flexibility, scalability, and low area, cost, and power consumption, while achieving effective classification. Many of the current architectures and implementations did not

take these constraints into account (especially the critical power constraint that was measured for only a limited number of previous implementations). In addition, a challenging trade-off has been found that exists between meeting embedded system constraints and achieving high classification accuracy. Moreover, very limited work exists for hardware implementation of the cascade SVM classifier, in addition to using the significant FPGA-based Dynamic Partial Reconfiguration (DPR) feature (to be introduced later). Furthermore, most existing implementations in the literature were realized on old versions of FPGAs. To the best of our knowledge, no FPGA implementation of the SVM classifier exploiting the benefit of the hybrid architecture (hardware/software system) of the recent FPGA platform "Zynq System on Chip (SoC)" exists in the literature. Also, almost all previous FPGA implementations are designed utilizing the traditional time-consuming Hardware Description Language (HDL) that requires expert hardware designers. However, the modern UltraFast High-Level Synthesis (HLS) design methodology is recently recommended to simplify FPGA development [8]. Additionally, no FPGA implementation exists in the literature for classifying melanoma clinical images using SVM classifier.

Consequently, this research focuses on implementing an SVM classifier on FPGA, aiming to overcome such limitations, challenges, and research gaps identified from the performed survey study.

The ***main objective*** of this research is to propose an optimized FPGA-based SVM classifier and implement an embedded SVM classification system on a chip to be used for melanoma detection as a case study. We aim to consider meeting significant constraints of embedded systems, while achieving efficient classification with high accuracy rate, aiming to find an optimum solution for the challenging trade-off.

The ***ultimate goal*** of this study is to develop an optimized FPGA-based SVM classifier to be integrated into the full CAD system towards a reliable cost- and energy-efficient embedded system to be deployed as a handheld device, dedicated to early detection of melanoma at the primary healthcare.

1.4 Research Questions

The main research questions of this study are:

1. What is the suitable type/algorithm of the SVM classifier for a hardware-friendly implementation targeting melanoma detection?
2. How to find optimum FPGA-based design methods, features/approaches and architectures to achieve better efficiency and results?
3. What is the best trade-off (optimum solution) for achieving high classification accuracy, while meeting embedded system constraints?
4. To what extent can we achieve flexibility and scalability of a hardware design for a big data classification problem?
5. How can we realize an adaptive classification system for melanoma detection on a reconfigurable hardware/FPGA with high performance and acceptable results?
6. How can we validate the classification accuracy rate of the implemented classifier for enhancing melanoma detection?
7. Is the implemented embedded SVM classifier feasible to be integrated efficiently into a full CAD system dedicated to melanoma detection?

1.5 Original Contributions

The main contributions of this research are summarised as follows:

- To the best of our knowledge, we published the first comprehensive review/survey of the literature covering FPGA-based hardware implementations of the SVM classifier [9] (another extended paper is currently under review [10]).
- A novel hardware/software co-design is proposed for realizing an efficient embedded SVM classification system on chip (FPGA) targeting early detection of melanoma, via implementation on a recent hybrid Zynq SoC/platform using the latest (UltraFast) design methodology.
- The implemented systems on Zynq SoC meet challenging embedded system constraints by achieving high performance and low area, power consumption and cost, while realizing high classification accuracy.
- By utilizing the modern HLS design methodology and its available optimization techniques (directives), the development effort and time are reduced, whilst the embedded system design process is simplified. In addition, by applying different HLS directives, design space exploration is investigated for achieving an efficient synthesized hardware for an optimized SVM classifier implementation. Different solutions/designs are presented for balancing the existing trade-off between speed

and area, offering options for various project requirements from a low-cost design to a fast design with higher cost.

- A simple flexible IP/core-based design is presented, which is scalable and easily extendable to support multi-purpose classification (ensemble, multi-class, cascade).
- A scalable multi-core architecture based on multi-SVM core is proposed and has been validated with a 2-stage cascade SVM classifier implementation for enhancing melanoma detection. A simplified hardware-friendly design is proposed for the building SVM core of the multi-core architecture in order to reduce hardware complexity and achieve optimized results.
- A novel hardware implementation of a dynamic and adaptive cascade SVM classifier exploiting the powerful DPR feature of the FPGA is presented, targeting flexibility, scalability, applicability and adaptability, while meeting critical embedded system constraints. This dynamic DPR-based hardware system is implemented for a 2-stage cascade SVM classifier on the recent hybrid Zynq SoC using the latest UltraFast design methodology, targeting early detection of melanoma with high performance and low cost. This classifier can easily be extended to a multi-stage classifier as required, while meeting challenging embedded system constraints.
- A comparative study is presented, comparing our various proposed designs and existing related implementations in the literature, based on experimental results of the hardware implementations. The implemented classification systems on Zynq SoC show the least power consumption results among other related implementations, in addition to significantly low hardware resource utilization and processing time with significant speedups and high classification accuracy rates at low cost.

1.6 Thesis Outline

The thesis is structured into five chapters as follows:

Chapter 2: Background and Literature Review; presents an overview of the FPGA and SVM classifier. Then, a comprehensive survey of current literature is presented regarding different hardware implementations of SVM on FPGAs. It includes a critical analysis and comparison of existing works with in-depth discussions around limitations, challenges, and research gaps.

Chapter 3: Proposed Hardware Architectures, Designs and Implementations; introduces the FPGA platform and the system development tools used in the implementations. Then, all proposed designs and architectures to implement an SVM classifier on FPGA are presented. A preliminary hardware/software co-design is first proposed as an initial implementation, and then extended hardware designs are proposed to implement an efficient embedded (hardware/software) system on the FPGA for realizing an optimized SVM classifier.

Chapter 4: Experimental Results and Discussion; presents different experimental results for each of the implemented SVM classification system using each of the proposed designs explained in the previous chapter. The implemented systems are evaluated, compared and discussed based on various experimental results, aiming to find an optimum hardware solution as well as meeting critical constraints of embedded systems. In addition, the implemented classifiers on FPGA are validated for online classification of melanoma and non-melanoma samples. Moreover, a comparative study is given for comparing the implemented systems with reported implementations in the literature.

Chapter 5: Conclusions and Future Works; gives an overall conclusion of the thesis as well as highlighting the original and significant contributions. Finally, some key future research directions are suggested.

CHAPTER 2 Background and Literature Review

2.1 Introduction

In this chapter, an overview for the FPGA and SVM classifier are briefly introduced. Then, a comprehensive survey of hardware architectures and designs used for implementing SVM classifier on FPGA over the period 2010-2017 is presented. A classification of existing hardware techniques (63 reviewed articles including our published papers [11-13]) is presented, with intensive reviewing and classification of the FPGA implementation of the SVM classification phase (our point of interest). In addition, a critical analysis and comparison of existing works with in-depth discussions around limitations, challenges, and research gaps are presented.

2.2 FPGA Background

Field-Programmable Gate Array is a programmable parallel processing reconfigurable semiconductor device that is based on a matrix of Configurable Logic Blocks (CLBs) connected through programmable interconnects (Figure 2-1) [14]. FPGA is used for achieving HPC in embedded systems with efficient utilization of hardware resources. Recently, FPGAs' performance has outperformed General-Purpose-Processors (GPPs) for different applications in a growing range of areas such as computer vision, pattern recognition, image processing, digital signal processing, machine learning algorithms, bioinformatics, etc. [3, 4, 15-17].

Graphics Processing Unit (GPU) also offers an alternative platform for high performance computing [18]. Many performance comparisons of FPGA and GPU implementations of different algorithms and applications have been studied in existing literature [5, 6, 19-23]. FPGAs demonstrated superior performance in most cases, however, in some applications, GPUs slightly outperformed FPGAs [23]. The availability of open source libraries such as OpenCV helps to achieve much faster development time for GPUs than for FPGAs. However, for more complicated algorithms that use shared arrays and high memory accesses, GPUs cannot provide good performance due to memory access limitations caused by their memory architecture [5]. Although GPUs benefit from lower cost and shorter development time compared to FPGAs, they are inferior to FPGAs in terms of power consumptions (FPGA consumes approximately an order of magnitude less power [6]). Moreover, existing GPU implementations are challenging and very hard

to be mapped efficiently to the energy-efficient embedded GPUs because of the fixed hardware and limited available resources (less memory, registers, cache and cores) [24]. Accordingly, implementations on power hungry GPUs are difficult to be deployed in embedded environments, and this has motivated a move towards FPGA implementation.

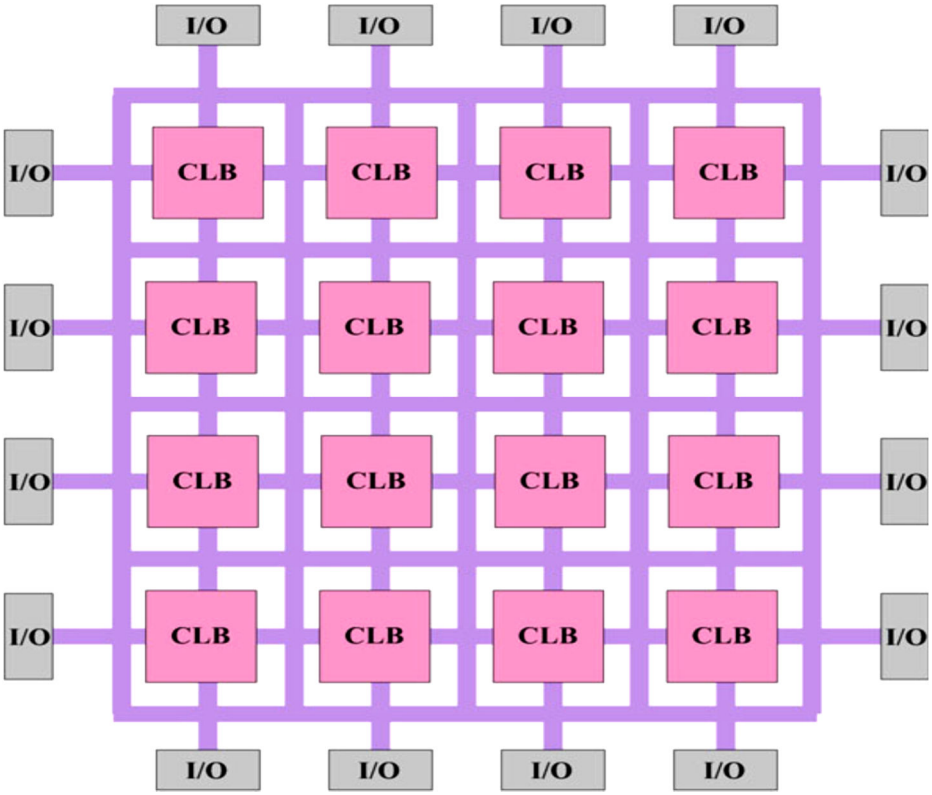

Figure 2-1. FPGA architecture consists of Configurable Logic Blocks (CLBs), Configurable I/O blocks and programmable interconnect [14].

Furthermore, modern FPGA technology encloses processor cores and different useful Intellectual Property (IP) blocks onto a single chip, offering more design simplicity and flexibility for developing high performance embedded systems and Multi-Processor Systems on Chip (SoC) [25]. Moreover, recent development tools simplify design of embedded systems by utilizing high-level languages, which reduces FPGA development effort and shorten time-to-market without relying on expert hardware designers (e.g. Xilinx Vivado HLS tool [8]). Hence, FPGA is an appropriate hardware platform for implementing an SVM classifier, whilst meeting challenging embedded system constraints.

In addition, FPGA is featured with a powerful Dynamic Partially Reconfiguration (DPR) technology that allows reconfiguring dynamically (reprogramming) selected areas on

FPGA on-the-fly, while other parts are still working (also called Run-Time Reconfiguration (RTR)) [26]. The module-based PR is widely used for various designs and applications, where Reconfigurable Modules (RMs) are reconfigured at run-time as depicted in Figure 2-2 (Reconfigurable Partition (RP) states the physical area on the FPGA designated for PR, whilst the rest remains static). DPR offers design flexibility, design space expansion, power and area savings with speedups, as well as increased productivity and scalability. Subsequently, FPGA is a promising hardware platform for implementing an embedded SVM classification system, offering high performance computing with more flexibility at low cost.

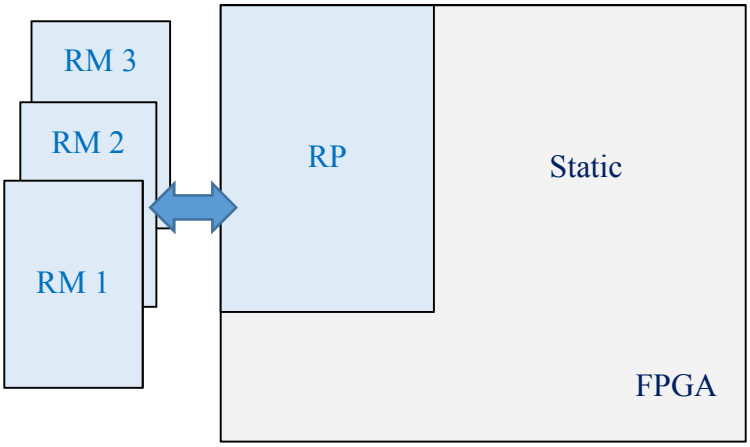

Figure 2-2. DPR architecture, where RP represents the reconfigurable partition to be reconfigured dynamically to one of the Reconfigurable Modules (RMs) at run-time.

2.3 SVM Background

Support Vector Machine (SVM) classifier is a common supervised machine learning tool which is widely used for efficient classification. SVM demonstrates high classification accuracy with numerous applications such as speech recognition, object detection, image classification, bioinformatics, medical diagnosis, etc. [27]. SVMs have shown high classification accuracy rates outperforming other popular classification algorithms in numerous cases and applications [7, 28-30]. Supervised learning machines are typically composed of two main phases, training/learning phase and classification phase. The SVM training phase constructs a model to be used for classifying any test data in the classification phase.

SVM is based on the concept of a decision boundary that separates two different classes of data in order to discriminate classes with high accuracy [31]. A separating hyperplane is constructed in the training phase by using an input training data set containing data samples. The hyperplane that best separates the samples belonging to the two classes is called a maximum-margin hyperplane that forms the decision boundary. The class samples that are on the boundary are called Support Vectors (SVs) as depicted in Figure 2-3, where SVs are encircled. These SVs obtained from training phase are then used in the classification phase for predicting the proper class of an input test data [32].

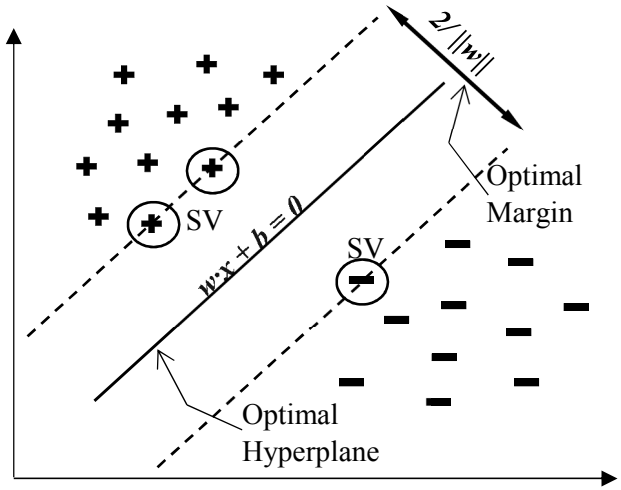

Figure 2-3. SVM separating hyperplane, where SVs are encircled to represent support vectors of the model, which are class samples on the boundary.

Consider training data labelled as (x_i , y_i), $i = 1, 2, ..., N$, $y_i \in \{-1, +1\}$, and $x_i \in R^d$. The hyperplane is the plane that separates the two classes of positive and negative samples as shown in Figure 2-3. The pattern x lies on the hyperplane in the feature space can be described by (2.1), where w is a normal vector to the hyperplane and b is a constant:

$$w \cdot x + b = 0 \tag{2.1}$$

By selecting the two hyperplanes described in (2.2) and (2.3), the data points are separated in the margin region, and the aim is to maximize the distance between them.

$$w \cdot x + b = +1 \tag{2.2}$$

$$w \cdot x + b = -1 \tag{2.3}$$

The Euclidian distance between these two hyperplanes is given as $2/||w||$ (see Figure 2-3), and so the distance $||w||$ needs to be minimized. The optimum separation hyperplane conditions can be formulated into the following expression that represents a linear SVM, minimize $||w||^2/2$ under the following constraints, which are added for all the training samples to prevent from falling into the margin:

$$w \cdot x_i + b \geq +1 \, , for \, y_i = +1 \tag{2.4}$$

$$w \cdot x_i + b \leq -1 \, , for \, y_i = -1 \tag{2.5}$$

This can be rewritten in the following equivalent form:

$$y_i \, (w \cdot x_i + b) \geq 1 \, , i = 1, \dots N \tag{2.6}$$

The optimization problem represents the minimization of a quadratic function under linear constraints (Quadratic Programming (QP)). A convenient way to solve constrained minimization problems is by using a Lagrange function, where a Lagrange multiplier α_i is assigned to each training pattern via the constraints represented in (2.6). Solving the SVM training problem by using the QP techniques is computationally expensive, especially for large high-dimensional datasets. Hence, numerous algorithms have been proposed in the literature for solving the QP problem like the Sequential Minimal Optimization (SMO) decomposition method [33]. The most important aspect of these solutions is that the complicated computation of the dot-product calculation is required for each iteration.

In many real-world classification problems, it is not possible to linearly separate the training data in the original space. So, the input space is mapped to a higher-dimensional one where a linear separation is feasible, which is a computationally expensive task, especially in large-scale problems. Therefore, SVMs go towards utilizing kernel tricks/functions $K(x_i, x)$ replacing the inner products in the optimization problem in (2.6) as in the following equation:

$$y_i \, (K(x_i, x \,) + b) \geq 1 \, , i = 1, \dots N \tag{2.7}$$

The SVM computational requirements depend on the used Kernel function. The most common kernel designs which are widely used because of their efficiency in mapping data to higher dimensional space are illustrated as follows [34]:

- Linear: $K(\vec{x}_i \, , \vec{x}) = \vec{x}_i \cdot \vec{x}$

- Polynomial: $K(\vec{x}_i, \vec{x}) = (\vec{x}_i \cdot \vec{x})^n$
- Sigmoid: $K(\vec{x}_i, \vec{x}) = \tanh(\vec{x}_i \cdot \vec{x} + \theta)$
- Gaussian Radial Basis Function (RBF): $K(\vec{x}_i, \vec{x}) = \mathrm{e}^{\left(\frac{-||\vec{x}_i - \vec{x}||^2}{2\sigma^2}\right)}$
- Hardware-friendly: $K(\vec{x}_i, \vec{x}) = 2^{-\gamma||\vec{x}_i - \vec{x}||_1}$

On the SVM classification phase, the classification function is universal and straightforward. A new data sample *x* is classified according to the output (sign) of the main decision function in (2.8). For large datasets, a massive number of complicated dot-product calculations is needed, which offers significant parallelization potential that could be exploited by parallel hardware resources as FPGAs.

$$F(x) = sign\left(\sum_{i=1}^{N} \alpha_i y_i K(\vec{x_i}, \vec{x}) + b\right) \quad (2.8)$$

SVM is originally designed for binary classification and the use for multiclass classification is more problematic, either several binary classifiers have to be built or a larger optimization problem is required. As multiclass problems are commonly encountered, many multiclass SVM classification strategies have been proposed in the literature like “one-against-all”, “one-against-one” and other methods [35].

2.4 Research Methodology of the Review Study

This review began with a comprehensive search involving thousands of publications within the initial scope of SVM implementation on FPGA. Five scientific databases were considered for the searching process: IEEE Xplore, Scopus, Google Scholar, ACM Digital Library, and ScienceDirect. The keywords used for the search task were Support Vector Machine, SVM classifier, SVM classification, FPGA, embedded system, and hardware implementation. The resulting collection of papers was narrowed down to only include conference and journal publications from 2010 onwards. This cut-off ensures a current view of technology given the significant advances in FPGA technology since 2010. Additional refinement was applied manually based on a screening of the remaining articles, so that only those work that focus on FPGA implementation of standalone SVM classifiers were considered.

Many recent works highlight the importance of real-time embedded applications in different areas like bioengineering, healthcare, digital signal processing, wireless sensor networks, multimedia, and others. Therefore, demonstrating the importance of boosting

SVM classifier to meet embedded system constraints is required to be considered and added to the current literature. There already exists a recent survey shows SVM in data mining tasks [27], regarding applications only and some of the challenges. Some of the considerable limitations of implementing the SVM model that are stated in [27] are the processing speed for the training and testing phase as well as large memory space requirements, which prove the necessity of an efficient implementation of SVM. To the best of our knowledge, there is currently no review or survey paper on FPGA-based hardware implementations of SVM targeted towards various embedded applications. Consequently, this review examines current and recent techniques in hardware implementations, targeting efficient implementations of the SVM model on FPGA.

The methodological framework of this research is defined by analysing a variety of techniques in hardware implementations on FPGA for implementing SVM classifiers. Thus, a total of 63 articles was classified into two main groups regarding hardware implementations on FPGA for the training/learning phase of the SVM as the first group (12 papers) and for the classification phase as the second group (38 papers), where training is performed offline in the software. In addition, a third group (13 papers) was also included to gather a miscellany of hardware implementations for applications that barely use the SVM classification as a part of a larger implemented algorithms covering different domains and areas. We restricted group 3 to image analysis algorithms only, as it resulted in a compact but robust set of techniques that can be qualitatively compared.

As group 2 is our main interest, we applied another methodological framework for intensively reviewing group 2, emphasizing on the implementation of the classification phase. This methodological framework is around identifying and classifying different FPGA implementations of the SVM classification phase. We proposed the following six categories of hardware implementations of SVM classifiers based on well-established hardware architectures:

A) Parallel pipelined architectures
B) Systolic array architectures
C) Dynamic partially reconfiguration-based architectures
D) Multiplier-less architectures
E) Software tool-based architectures
F) Cascaded classification-based architectures

Finally, a total of 38 papers published in the period 2010-2017 were reviewed and discussed in group 2 (including our published papers [11-13]). In addition, an extensive analysis was presented including a comprehensive comparison with in-depth discussions around limitations, challenges, and research gaps of the surveyed work. Moreover, leading research groups in the area were identified, which could be considered by hardware designers for future promising implementations.

2.5 FPGA Implementations of SVM

2.5.1 Group 1: SVM Training Phase Implementations

The training/learning phase of the SVM model has motivated many researchers to use hardware accelerators targeting a reduction in total training time. In this section, a variety of different FPGA-based architectures is presented in a systematic manner for the first group with a total number of 12 papers.

Many SVM implementations were presented in literature that were based on the common SMO decomposition method [33]. T. Kuan et al. [36] proposed a fully functional circuit design for accelerating the SVM learning phase based on SMO algorithm. The proposed architecture consists of three main circuit modules functioning for the SMO process with a memory block and a cache block that are controlled by a designed Finite-State Machine (FSM) based controller. Experimental results showed that lesser processing time was recorded from using the cache, and the recognition performance of the proposed fixed-point design (FPGA) was similar to that of the floating-point SMO running on MATLAB.

K. Cao et al. [37] proposed a parallel scalable digital architecture for training SVM based on SMO algorithm, aiming to overcome the lack of necessary flexibility in previous implemented embedded applications in the literature. A modified version of the traditional SMO algorithm [38] (to improve the efficiency of the working set selection method) was adopted in the proposed digital system, where a multiple processing units working in parallel was mapped to the error cache updating task in the algorithm. The memory size of the hardware and the number of processing units are adjustable, aiming to achieve scalable architecture for handling different sizes of the training problems. The synthesizable Verilog code adopted for FPGA synthesis was generated automatically from the Simulink Stateflow using Simulink HDL Coder. Experimental results (based on two different datasets) demonstrated that SVM training problems could be solved

effectively with inexpensive fixed-point arithmetic, offering better flexibility and scalability results.

J. Filho et al. [39] presented a dynamically reconfigurable SVM architecture for general purpose training that supports different sizes of training sets. Based on the SMO algorithm, a modular architecture was designed reaching interchanging modules by dynamic reconfiguration. The hardware-friendly kernel function proposed in [40] was adopted for the system and so the Coordinate Rotation Digital Computer (CORDIC) algorithm [41] based on shift and add operations was employed for kernel implementation. The proposed reconfigurable architecture achieved 22.38% area saving with acceptable reconfiguration time penalty. The effect of fixed-point data representation on precision and classification error was studied on three different learning benchmarks. Based on simulation results, the hardware implementations of the three benchmarks achieved acceleration factors of more than 12.53 times faster than the software implementation for the total training time.

C. Peng et al. [42] proposed a novel reconfigurable and efficient chip design for accelerating SMO-based SVM learning. Two novel methods were used in the proposed design; trimode coarse-grained reconfigurable architecture and triple finite-state-machine with dynamic scheduling, targeting improvement of the baseline design proposed in [36]. The first method amended the baseline design by proposing trimode reconfigurable architectures with parallel and pipeline computing capabilities, while the second method offered a schedule for efficient reconfiguration. Compared to the baseline design, simulation results demonstrated that the proposed design achieved improvements in area size (50% less memory usage from removing the kernel cache and 31% fewer gate counts), power consumption (17-fold improvement), and training speed (16-fold improvement) with satisfactory recognition accuracy (85%).

L. Martinez et al. [43] proposed a hardware/software architecture to accelerate SVM training phase based on SMO algorithm. The high time-consuming dot-product computation in the SMO algorithm was executed on the hardware coprocessor in parallel, while the heuristic hierarchy of SMO was implemented in software on the GPP. The bottleneck dot-product calculation was mapped to a logical AND operation and a counter to be executed in three clock cycles. The proposed coprocessor architecture achieved a speedup of 178.7x compared to a software-only implementation on a GPP.

Another hardware/software co-design system for fast SVM training based on SMO algorithm was proposed for embedded speaker identification system [44]. The modular design proposed in [36] was exploited and improved for realizing the computational bottleneck SMO training on hardware, whilst other processes including pre-processing, feature extraction and SVM-based voting analysis were implemented in software on an ARM processor (embedded C code). Also, a data-packed/unpacked mechanism was proposed to improve the efficiency of communication and data transmission between software and hardware (around 5% reduction in delivery time). Compared to an ARM embedded C code, the proposed system achieved 90% decline in training time with a slight decrease in identification rate, where about 89.9% identification rate was achieved.

A. Patel et al. [45] proposed also a hardware/software co-design system to speedup SVM training based on SMO algorithm. A fully scalable co-processor architecture was implemented on Xilinx Virtex-7 FPGA for handling kernel computations based on a proposed SMO algorithm, which effectively designed for exploiting the parallel computing power of the proposed co-processor and caching kernel columns through a grid of processing units. A speedup of 20-25x was achieved for the kernel computations on the designed co-processor, while an application speed-up of up to 15x was achieved by comparing results of experiments of the proposed system on the ADULT dataset with LIBSVM.

Other researchers implemented SVM based on different decomposition algorithms replacing the traditional SMO. A hardware/software co-design system for accelerating the SVM learning phase was presented based on another decomposition algorithm instead of the common SMO algorithm [46]. A Hybrid Working Set (HWS) algorithm was proposed based on the extended working set decomposition algorithm [47], taking advantage of cached kernel values and the fast convergence nature in order to decrease the number of iterations. A fully scalable coprocessor architecture consisting of a grid of cores was proposed to achieve parallelism for kernel computations similar to their previous proposed architecture [45]. A speed-up for kernel computations of up to 25x was achieved from the implemented coprocessor (32 cores) over a single threaded core i5 CPU. A reduction in iterations of 50% and 60% was achieved, compared to LIBSVM and SVMLight software programs due to the implemented HWS algorithm. Finally, the proposed coprocessor with the HWS algorithm achieved an application speedup of up to

15x and 23x compared to software implementations of LIBSVM and SVMLight respectively.

M. Papadonikolakis et al. [48] proposed a fully scalable heterogeneous FPGA architecture for boosting the SVM learning, which fully exploits the device parallel processing power and the dynamic range diversities of the precision requirements among training problem features. The proposed design fully utilized the custom-precision arithmetic and heterogeneous components supported by the FPGA device for handling kernel computations. The proposed architecture used parallel custom precision multipliers feeding a pipelined adder tree for the fixed-point inner products which are then interpreted into floating point format for further calculations. Experimental results demonstrated the efficiency of the proposed heterogeneous architecture, which increased with the precision diversities of the homogenous/heterogeneous datasets attributes. Also, the proposed design showed a speedup of more than 6x compared to other proposed designs.

M. Rabieah et al. [49] presented a complete FPGA-based system for boosting nonlinear SVM training by utilizing ensemble learning, in addition to a proposed cascaded multi-precision training flow that exploits FPGA reconfigurability and the training problem heterogeneity for handling large datasets. The proposed hardware module was designed for implementing Gilbert's training algorithm (simpler than SMO [50]), where numerous processing elements are used for kernel dot-product operations, each with its own on-chip memory blocks. The architecture of the processing element followed the previously proposed architecture in [48], where the kernel computation was divided between fixed point and floating point domains, and was improved by using caching and running the solution update in parallel. The proposed FPGA system achieved significant speedups compared to other CPU and GPU-based implementations across three different datasets with acceptable accuracy and lower power consumption.

M. Ning et al. [51] used a different design methodology from the previous works presented; High-Level Synthesis (HLS) for developing the hardware accelerator. A demonstration was presented to show that the hardware development time for an embedded system could be reduced by applying the HLS method using C language instead of traditional Hardware Description Language (HDL). A SOPC (System on Programmable Chip) system was designed to implement the Least Square SVM (LS-SVM) on the recent Xilinx Zynq platform using Vivado HLS tool with C language. The

LS-SVM has lower computational complexity as it solves a set of linear equations instead of a quadratic programming for standard SVM [52]. The proposed algorithm was divided into three parts; a generating Kernel Matrix module, a Solving Linear Equations Module (SOLE), and a forecasting module. The first and third modules were realized in ARM processor which also controls computing modules and data path, whilst the compute-intensive second module was implemented in programmable logic utilizing the HLS method. A speed up factor of 2-8x was achieved with 0.06% maximum relative error for trained data, compared to a Matlab-based CPU implementation for experiments with a real dataset.

S. Wang et al. [53] exploited the Run Time Reconfiguration (RTR) technology of the FPGA for boosting the online training of the LS-SVM. The proposed design was divided into two parts to be swapped using RTR and applying pipeline in the modules design, reaching high parallelization. The first part is the kernel matrix formulation where a piecewise linear interpolation method was applied. The second part is the least-square problem solving, where a modified Cholesky Decomposition was proposed, improving large memory requirements and the long latency caused by square roots operations. Experimental results illustrated speed up from 6 to 218x compared to a Xeon CPU implementation. Regarding time cost percentage analysis, the proposed architecture was shown to be suitable for large-scale problems with more than 1000 samples, due to the large reconfiguration time.

2.5.2 Critical Analysis of Group 1

The total training time is considered the main bottleneck of the overall SVM performance in real-time applications, which motivated many researchers for speeding-up this consuming task through implementation on parallel FPGA devices. This section presents a critical analysis for the previous FPGA implementations of the SVM classifier that were introduced in the previous section and summarized in Table 2-1.

Most of the proposed hardware architectures for implementing the SVM learning were based on the common SMO decomposition algorithm [36, 37, 39, 42-45], whilst an alternative algorithm was employed by one work in [46] and another simpler algorithm was implemented in [49]. Two research papers implemented the improved SVM; LS-SVM instead of the standard SVM due to its low computational complexity [51, 53]. In general, most previous work focused on boosting the whole training process including complicated kernel computations.

Various parallel digital designs were presented, where the common pipelining approach was mostly applied, reaching a high level of parallelization. Additionally, some studies employed the dynamic reconfiguration technique for the designs, aiming speed improvement with more flexibility [39, 53]. Moreover, dynamic scheduling was exploited for efficient reconfiguration in [42].

Another research work [39] adopted for their system the hardware-friendly kernel function that was proposed in [40] for reducing hardware complexity. Thus, the common Coordinate Rotation Digital Computer (CORDIC) algorithm [41] was employed for the kernel implementation, which is based on shifters and adders replacing expensive multipliers that led to a remarkable reduction in resource utilization.

Many implemented systems employed the hardware/software co-design method for running the compute intensive task of the algorithm on the FPGA as a hardware accelerator (co-processor), aiming to reach real-time SVM training [43-46, 51]. In addition, researchers in [51] used different design methodology; High-Level Synthesis (HLS) for developing the hardware co-processor instead of using the traditional HDL, where an outstanding reduction in hardware development time and effort for realizing an embedded system was emphasized.

Concerning scalability, some proposed architectures were designed to meet scalability issue, offering more flexibility for efficient usage in embedded environment [37, 45, 46, 48]. Interestingly, a unique heterogeneous FPGA architecture for fully exploiting custom-precision arithmetic and heterogeneous components of the device was proposed [48, 49], offering scalability and adaptability to the classification problem nature.

Finally regarding the achieved results, many proposed implementations for SVM training phase achieved significant speedup results which outperformed similar software implementations [39, 42-46, 49, 51, 53], where some reached acceptable accuracy with slightly rate loss [42, 44]. On the other hand, some hardware implementations gained remarkable speedup improvement compared to previous hardware designs [42, 48]. Furthermore, great area saving results were realized in [39, 42] and an improvement in power consumption was shown by [42, 49], which could meet some important embedded system constraints.

Table 2-1. Group 1: SVM Training Phase Implementations

Ref.	Training Implementation Method	FPGA Platform/ Board and Development Tool	SVM Type	Kernel Function Type	Application Domain / Dataset	Important Results
[36]	Functional circuit design	Altera Cyclone II DE2-70 Quartus II 7.2 sp3	Multiclass	Linear	Speaker recognition, SMD and FMMD speech datasets	Fully functional prototype
[37]	Parallel scalable digital architecture	Xilinx Virtex-4 (XC4VLX100) Simulink Stateflow +HDL Coder	Binary	Gaussian	Sonar dataset, Telecommunication problem dataset	-
[39]	General-purpose dynamically reconfigurable architecture, CORDIC	Xilinx Virtex-IV (XC4VLX25)	Binary	Hardware-friendly kernel	Breast Cancer, Dermatology, Tic Tac Toe benchmarks	22.38% area saving Speedup > 12.53x
[42]	Reconfigurable architecture with dynamic scheduling	Spartan c3s4000	Multiclass	Linear	Speaker recognition, SMD and FMMD speech datasets	Improvements in power, area, memory efficiency
[43]	Co-processor	XtremeDSP Virtex- IV Development Kit Xilinx ISE 9.2 ModelSIM SE 6.5	Binary	Linear	ADULT dataset	Speedup 178.7x GPP
[44]	Co-processor	Xilinx Spartan-c3s4000 Xilinx ISE 10.1	Multiclass	Linear	NIST 2010 speaker recognition database	90% less training time
[45]	Co-processor	Xilinx Virtex-7 (VC707)	Binary	Gaussian	ADULT dataset	Speedup 20-25x App speedup 15x
[46]	Co-processor	Xilinx Virtex-7 (XC7VX485T) Xilinx PlanAhead	Binary	Gaussian	Different datasets	Speedup 25x CPU App speedup 15x LIBSVM, 23x SVMLight

[48]	Heterogeneous architecture(custom-arithmetic)	Altera's Stratix III (EP3SE260)	Binary	Linear Gaussian polynomial sigmoid	Various datasets	Speedup 6x
[49]	Heterogeneous architecture(custom-arithmetic)	Xilinx board ML605 (Virtex-6 XC6VLX240T	Binary	RBF	ADULT, Forest covertype, MNIST datasets	Speedup >1 order
[51]	System on programmable chip (HLS)	Xilinx XC7Z020 Vivado HLS	LS-SVM	RBF	Electricity Demand dataset	Speedup 2-8x CPU
[53]	Run Time Reconfiguration	Xilinx ML510 (XC5VFX130T) Xilinx PlanAhead	LS-SVM	RBF	mobile communication traffic dataset	Speedup 6-218x CPU

This table summarizes all reviewed articles in group 1. For each reference in column one, the implementation method of the training phase is reported in column two, followed by the FPGA platform/board and development tools used for the implementation. Then, the type of the SVM classifier (binary/multiclass) and the kernel used are stated in columns four and five respectively, followed by the used application/dataset and finally some important results are summarized in the last column.

2.5.3 Group 2: SVM Classification Phase Implementations

Numerous techniques implemented the SVM classification stage in hardware on FPGAs. In order to implement the classification stage on hardware, SVM is first trained offline in software, and then a trained model is extracted for the FPGA implementation of the classifier. MATLAB was the mostly used platform for the SVM training stage. The 38 works reviewed and discussed in this section presented different hardware architectures and implementation techniques. We have classified the implemented architectures into six main categories, where some works applied multiple architectures and are therefore categorized into multiple categories. Summaries of the techniques belonging to each of the six categories are shown in Table 2-2, Table 2-3, Table 2-4, Table 2-5, Table 2-6 and Table 2-7 respectively.

2.5.3.1 Category A: Parallel Pipelined Architectures

The pipelining technique breaks a process into smaller stages, which allows for the concurrent execution of jobs. Pipelining can significantly boost the whole process and increases data throughput. Many implementations used the pipelined technique by exploiting the parallel processing capabilities of FPGAs, aiming to achieve efficient parallel pipelined architectures.

A fully pipelined architecture was proposed for implementing and accelerating the SVM classification, while presenting three different configurations to implement RBF,

polynomial, and sigmoid kernel functions [54]. A new processor was designed in a fully pipelined architecture by effectively exploiting embedded DSP slices and block RAMs of the FPGA. The proposed architecture used 768 DSPs and 800 BRAMs for implementing an SVM classifier, including 760 SVs. A high throughput of 2.89x106 times per second was achieved at 370.096 MHz, for classifying 128 feature dimension.

Another fully pipelined structure was proposed in [15, 21], emphasizing on a comparison between FPGA and GPU implementations. Regarding the FPGA implementation, complicated hardware modules were designed in a fully pipelined architecture using traditional HDL, whereas other basic modules, Finite-State Machines (FSMs), FIFO, and interfaces were implemented using the High-Level Language (HLL) (Impulse C). For few image pixels, the FPGA implementation was faster than the GPU and CPU implementations [21]. However, for larger number of pixels, the GPU implementation was the fastest, but it dissipated very high power, which makes it infeasible for embedded applications.

A pipelined architecture was proposed to realize a multi-purpose SVM with high flexibility in terms of input data and kernel type in [55]. A pipelined design was presented for a complete SVM core that allows dynamic selection of linear, polynomial or RBF kernel at run-time. The dot-product operation was computed with parallel embedded DSP-based MAC units and a LUT-based adder tree. The Xilinx Coordinate Rotation Digital Computer (CORDIC) [41] IP core was used to implement the exponential function. Also, two different number formats were used in the core, a fixed-point number format for the dot-product calculation, and a single precision floating point format for other computations. This flexible SVM core was simulated and verified at 50 MHz for the RBF kernel (a maximum frequency of 92 MHz might be achieved).

An FPGA pipelined design was introduced in [56], where a simplified algorithm based on posterior probability was implemented. The proposed pipelined structure exploited a LUT method for computing the sigmoid function, whilst for other computations, adders, multipliers and dividers were utilized. A mean absolute error of 10-4 order of magnitude was achieved from the FPGA implementation of the proposed simplified algorithm (fixed-point) compared to the original C algorithm (floating-point), showing little loss in the recognition rate. In addition, the computation complexity was reduced and 0.7 ms time delay was achieved, which is promising for meeting real-time constraints for the targeted application.

A hardware architecture exploiting inherent FPGA parallelism and pipelining features was proposed in [57]. A block diagram was drawn for the proposed hardware architecture using the standard single precision floating-point format. This architecture was based on using counters for address generation to different stored data in BRAMs for performing required calculations. A maximum clock frequency of 200 MHz was achieved with promising hardware results based on synthesis results, whilst 97.87% accuracy was achieved based on simulation results.

A pipelined architecture was presented in [58], aiming to reach a universal coarse-grained reconfigurable architecture that implements one of three types of machine learning tools including SVM. The proposed pipelined structure was designed as a 1D or 2D array of simple reconfigurable blocks in order to implement one of the three classifiers. For the SVM implementation, the classification process was divided into partial sums controlled by a FSM model based on reconfigurable blocks utilizing adders and multipliers (and a subtractor in case of a radial kernel). The proposed implementation gained an acceleration of 1-2 orders of magnitude, compared to a software implementation (R project-based), while achieving acceptable utilization of hardware resources (additional classification speedup experiments/comparisons were provided in [59]).

A parallel architecture with two-stage pipeline was presented in [60]. It used resource sharing for realizing a unified circuit for both linear and non-linear SVM classification. The proposed architecture included shared adders and multipliers for inner product computations required by linear and non-linear SVM. The table-driven algorithm proposed in [61] was exploited for computing the RBF kernel, aiming to boost the exponential function calculation (fixed-point), and increase accuracy. The synthesized circuit utilized 661, 261 gates at a maximum frequency of 152 MHz and achieved a speed of 33.8 fps.

Another 2-pipelined stage architecture implemented a 3-class SVM identification system [62, 63]. The first pipelined stage computes the inner-product, while the second stage calculates the summation. The proposed architecture was implemented using fixed-point number format and evaluated for different bit-lengths, demonstrating the trade-off between the identification accuracy and hardware area. A customized architecture was also presented that was based on their previous implementation in [64], which combined 2-class classifiers. As a result, an increase in the processing speed of 18% for the inner-product computations was achieved from the overlap of input data. Finally, an improved

2-pipeline stage architecture was introduced to avoid the duplication in the inner-product process, where common inner-products were calculated at the first stage (similar to the above architecture described [60]). The implementation results showed a system throughput >21.2 fps at 100 MHz [62], with sufficient identification accuracy >90% and good hardware size.

A pipelined adder was proposed to execute the accumulation operation replacing traditional adders in the main processing element, targeting acceleration of the SVM classification process [65]. The proposed pipelined adder-based processing element outperformed RCA and KS adders by 1.44 and 1.21 respectively, while utilizing few extra resources. This implementation was realized on an ASIC, however 3.5x GMACs were achieved compared to other FPGA implementations.

A pipelined digital architecture was proposed in [66], where two SVMs are working in parallel for 3-classes classification using fixed-point arithmetic. The proposed implementation was based on using subtraction, multiplication, and addition operations with comparison blocks. Pipelining with the time multiplexing technique was exploited to decrease hardware resource utilization and processing clock cycles. The implemented system showed high accuracy rates (> 81%) with low resource utilization, while processing a hyperspectral cube in 25 ms at 100 MHz with low dynamic power consumption of 67 mW.

A parallel hardware architecture was presented in [67], which used feature extraction algorithms with SVM for real-time image classification. The implemented SVM used multipliers and accumulators for parallel kernel processing using fixed-point numbers, while a Xilinx CORDIC IP was utilized for performing the exponential function. The proposed hardware system achieved 75% and 85% classification accuracy for two challenging datasets used, with 3% loss in accuracy compared to the software implementation. Also, a speedup of 5.7x was achieved compared to the software implementation, whilst moderate hardware resources were utilized with 0.25 ms processing time for an image.

Table 2-2. Category A: Parallel Pipelined Architectures

SVM Type	Ref.	Kernel Function Type	Application Domain/ Dataset	FPGA Platform/ Board	Development Tool
Binary	[54]	RBF Polynomial Sigmoid	-	Xilinx Virtex-6 (6VLX240T-FF1156)	Xilinx ISE 14.1
	[15, 21]	Gaussian	Skin classification	Xilinx Virtex-5 (LX220) Xilinx Spartan 6(XC6SLX75)	
	[55]	Linear RBF Polynomial	Pedestrian detection	Xilinx ML505 (XC5VLX110T)	-
	[58]	RBF Polynomial	18 UCI datasets	Xilinx Virtex-7	Vivado 2014.2
	[60]	RBF	-	-	-
	[65]	Linear	MNIST dataset	ASIC	Synopsys
Multiclass	[56]	Linear Sigmoid	Language Recognition	Xilinx Virtex-5	ISE 10.1
	[57]	Linear	Facial expression classification	Xilinx ML510 (Virtex-5 FXT)	ModelSim 10.1C, Xilinx ISE 14.2
	[62, 63]	Linear	Colorectal cancer detection	Altera Stratix-IV (EP4SE360F35C2)	-
	[66]	Linear	Bruised apple detection	Xilinx Zynq Z-7010	-
	[67]	RBF	Image classification Caltech-256, Belgium Traffic Sign datasets	Xilinx ML509 (Virtex 5 LX110T)	Xilinx ISE

2.5.3.2 Category B: Systolic Array Architectures

The Systolic array architecture is a configurable modular platform that combines both parallelism and pipelining techniques for enhancing computing speed. The systolic architecture is designed as an array of simple processors or Processing Elements (PEs), which provides efficient data flow and memory management. A 2D systolic array architecture is depicted in Figure 2-4 [68], which demonstrates a promising platform for achieving scalable designs and decreasing hardware complexity. Parallel FPGA was widely used for implementing the systolic array architecture for numerous applications,

especially matrix multiplication-based applications. Several SVM implementations exist in the literature that exploited the FPGA-based systolic array architecture for gaining accelerated parallel processing.

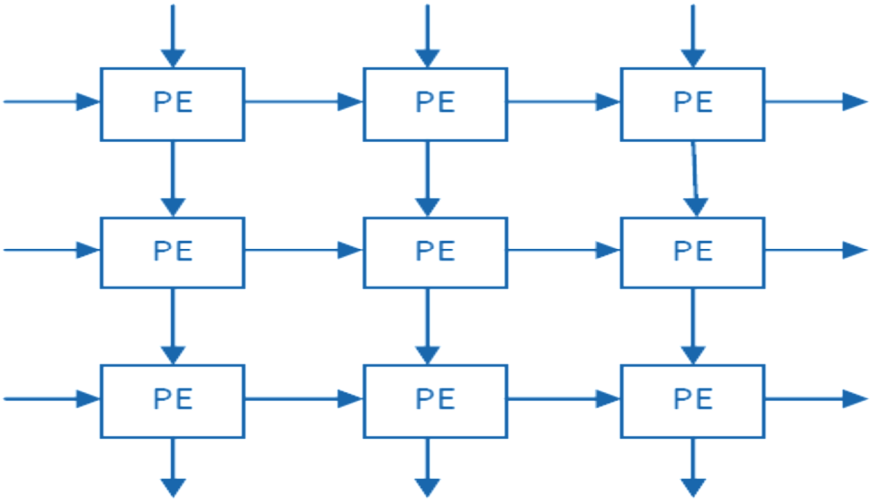

Figure 2-4. Systolic array architecture; an array of simple processors or Processing Elements (PEs) [68].

A parallel architecture proposed in [69] was based on a systolic array of PEs [70] (an initial implementation work), aiming to reach a scalable, flexible and adaptive array processing system. The proposed systolic architecture consisted of three main sections, memory control, vector processing and scalar processing. The implemented array architecture was evaluated for three types of object detection applications. A high performance of 40, 46, 122 fps was achieved for the three tested applications. Interestingly, compared to the software implementation results, no loss was achieved in the detection accuracy from the hardware implementation of the three applications (76, 77, 78%).

In addition, different implementations of systolic array architecture of PEs that was responsible for the complicated matrix multiplication required for the kernel computation were presented in [71-74]. Besides gaining advantages from using the systolic architecture, these implementations used an additional hardware technique, the partial reconfiguration technology (to be presented in the following category).

Table 2-3. Category B: Systolic Array Architectures

SVM Type	Ref.	Kernel Function Type	Application Domain/ Dataset	FPGA Platform/ Board	Development Tool
Binary	[69]	RBF Polynomial	Object detection	Xilinx ML505 (Virtex 5-LX110T)	-

2.5.3.3 Category C: DPR-based Architectures

Dynamic Partially Reconfiguration (DPR) technology allows reconfiguring dynamically selected areas on FPGA on-the-fly, while other parts are still working (also called Run-Time Reconfiguration (RTR)) [26]. Not all FPGAs support the partial reconfiguration technology. DPR offers design flexibility, design space expansion, power and area savings with speedups. However for performance-critical cases, the reconfiguration speed that is responsible for switching between different cores/modules should be taken into consideration as a significant factor. Two styles of PR exist, module-based and difference-based. The module-based PR is widely used for various designs and applications, where Reconfigurable Modules (RMs) are reconfigured at run-time as depicted in Figure 2-2. The difference-based PR is efficient for small designs and not recommended for big design changes, where a partial reconfiguration bitstream is provided to program the difference only between the two modules/designs instead of the full configuration bitstream of each module.

DPR was used to implement an SVM classifier in [71], in addition to using the systolic array architecture (category B) designed in four blocks. The kernel calculation was implemented in three sub-blocks designed in 2-pipelined stages (category A). Compared to an equivalent GPP implementation, an acceleration of up to 85x was achieved. Additionally, by applying the DPR for changing SVM parameters, a speedup of 8x was realized compared to the full chip reconfiguration.

This work was extended lately in [72], where two systolic array-based architectures were proposed to handle various dataset sizes (number of SVs is greater than the dimension and vice-versa). The two architectures gained speedups of ~61x and ~49x respectively compared to equivalent GPP implementations. Furthermore, two DPR implementations were presented as a single-core (same as in [71]) and multi-core SVM architecture (quad-core), where different copies of SVM cores with various parameters are swapped. Same speedup of 8x was achieved from using the DPR as mentioned above.

Later, Hussain et al. [73] extended their work by presenting a DPR implementation of an adaptive multi-classifier architecture that provides dynamically interchanging different copies of the SVM and K-Nearest Neighbour (KNN) classifiers with various parameters (based on their previous SVM and KNN implementations [71, 72, 75, 76]). The proposed architecture allows users to configure specific region on the FPGA to work as either SVM or KNN classifier, which could be extended in future to perform as an ensemble classifier. The multi-classifier DPR implementation showed a speedup of 8x in reconfiguration time compared to the single non-DPR classifier implementation (same value for previous implementations [71, 72]), and achieved twice as less area on FPGA as having both classifiers onto the same chip.

Beside using the systolic array architecture (category B), the difference-based partial reconfiguration technique was exploited in [74], targeting low area and power consumption. A power-aware SVM classifier prototype was realized from using the PR, as a power drop of 3 to 5% was achieved (total power = 2.021W).

Table 2-4. Category C: DPR-based Architectures

SVM Type	Ref.	Kernel Function Type	Application Domain/ Dataset	FPGA Platform/ Board	Development Tool
Binary	[71-73]	Linear	Classifying microarray/biomedical data	Xilinx ML 403(XC4VSX12)	Xilinx ISE, PlanAhead, ChipScope 12.2
Multiclass	[74]	Polynomial	Facial expression recognition	Xilinx Virtex-6 (6vlx240tff1156-2)	Xilinx ISE, Power analyzer EDA

2.5.3.4 Category D: Multiplier-less Architectures

The multiplier-less approach aims to reduce hardware complexity by avoiding the usage of the computationally intensive multipliers. Different architectures based on the multiplier-less technique were implemented in the literature. Many implementations adopted the hardware-friendly kernel function proposed in [40] that is based on a simple multiplier-free design, which offers acceptable classification performance compared to the traditional Gaussian kernel.

An embedded hardware SVM implementation was proposed in [77], which exploited the proposed hardware-friendly Kernel. A simple design of six blocks diagram was proposed

that consists of Sum of Absolute Differences (SAD) [78] and shifts operations instead of the resource-consuming multiplications. As a result, low resource utilization of 167 slices was demonstrated.

Another hardware-friendly Kernel-based architecture was presented in [79]. A SAD-based tree structure was used for the 1-norm computation between vectors, targeting clock cycles reduction and processing speed improvement. A preliminary simulation study was presented for the bit-precision of fixed-point arithmetic regarding the classification accuracy level.

An additional implementation that adopted the hardware-friendly architecture was introduced in [80] for a simple hardware design. The proposed architecture utilized the CORDIC iterative algorithm [41], which is based on exploiting shift and add operations that replaces complicated multipliers. The implemented SVM classification system utilized an external memory for storing support vectors with 2 ms speed limitation. From the hardware simulation, the implemented system demonstrated 4% error rate, while consuming 75% of the device logic. These results suggest that there is still some room for further research and improvement.

An FPGA implementation for a fast SVM was presented in [81] following the CORDIC-like algorithm implementation of the proposed hardware-friendly Kernel [40]. The proposed architecture was designed in three sub-circuits for the kernel calculations. An iterative algorithm was proposed for simplifying the CORDIC method by using adders and shifters only. The implemented system was faster than their previous CORDIC circuit implementation in [82] by a factor of 6, while very few hardware resources were utilized.

An FPGA core generator tool for embedded SVM was proposed in [83], which automatically generates an optimized hardware description for a digital implementation, meeting user requirements and the target device constraints. The proposed hardware-friendly kernel [40] was also adopted in addition to the proposal in [84], in which the bottleneck 1-norm (Manhattan norm) calculation was implemented by exploiting the parallel tree structure of the common SAD modules [78]. For the kernel implementation, three architectures were proposed. The first architecture used multiple cascade pipelined stages of Multiply-and-Accumulate (MAC) blocks (category A), in order to implement the polynomial approximation method. The second architecture employed the CORDIC-like iterative algorithm with multipliers-free implementation as in [40]. The third

architecture used the Look-Up Tables (LUTs) or memory blocks for storing kernel data. The tool was tested regarding the trade-off between latency, hardware resources and maximum clock frequency for implementing the three architectures on low-cost, intermediate-class and high-end FPGAs. The second CORDIC-like and third LUT-based architectures demonstrated higher classification figures.

A hardware architecture was presented in [85], which employed the multiplier-less kernel that is similar to the previous hardware-friendly kernel [40] and appears as the common radial kernel. The proposed architecture utilized simple shifters for implementing the multiplications, while the CORDIC algorithm was used for implementing the complicated exponential function. The proposed kernel achieved a comparable classification performance with the original radial kernel (No FPGA implementation was presented).

A multiplier-less kernel implementation for boosting SVM was presented in [86], which was based on the parallel pipelined systolic array architecture (category B). Similarly, simple shift and add operations implemented a simplified multiplier-less kernel for decreasing the hardware complexity and power consumption. In addition, a different approach was introduced by applying the CSD (Canonic Signed Digit) and CSE (Common Sub-expression Elimination) representation methods for reducing the number of required adders, which decreased the hardware complexity [87]. A comparative study was presented for the resource utilization of three different implemented classifiers (binary linear, binary non-linear and multiclass), which employed the proposed CSD-based multiplier-less kernel against the normal vector product kernel [86]. Moreover, the three implemented classifiers achieved power reductions of 1, 2.7, and 3.5% from using the traditional vector product kernel.

2.5.3.5 Category E: Software Tool-based Architectures

Most existing FPGA implementations in the literature were designed using the traditional HDL approach. However, very few implementations in the literature exploited the latest software design tools on the market, which mostly are very powerful and simplify the complicated hardware design process as well as decreasing the development time. In this section, the software tools used for the SVM implementations on FPGA are introduced as well as the proposed designs.

Table 2-5. Category D. Multiplier-less Architectures

SVM Type	Ref.	Kernel Function Type	Application Domain/ Dataset	FPGA Platform/ Board	Development Tool
Binary	[77]	Hardware-friendly	Satellite onboard/ NASA database	Xilinx Virtex-5 Spartan-3E	ModelSim
	[79]	Hardware-friendly	UCI standard dataset Breast Cancer	Altera Cyclone III	ModelSim
	[81]	Hardware-friendly	-	-	-
	[83]	Hardware-friendly	Automotive app/pedestrian detection Daimler-Chrysler dataset	Xilinx Spartan-IIE Spartan-3 Virtex-II Pro Virtex-4	-
	[85]	Digital kernel	4 UCI datasets	-	-
Multiclass	[80]	Hardware-friendly	Image recognition COIL dataset	Cyclone II(EP2C20)	-
Binary and Multiclass	[86]	Linear polynomial	Fisheriris dataset	Xilinx Virtex-7 (7vx485tffg1157-2)	Xilinx XPE 14.1

Xilinx System Generator is a common tool, which allows high-performance system modelling and automatic code generation from Matlab/Simulink [88]. The System Generator was used in [89] to design and implement a simple hardware architecture. A parallel hardware architecture was designed using a combination of serial and parallel designs of simple design blocks and functions of the System Generator. Additionally, the available CORDIC block of the tool was instantiated for implementing the complicated exponential function. Fixed point numbers were used for quantization. The simulation results reported maximum frequency of 202.840 MHz for linear classification and 1.33% error rate for nonlinear classification (98.67% accuracy). Also compared to Matlab implementation, less computation time was achieved for both implemented classifiers.

In addition, the Xilinx System Generator was utilized in [90], where a DSP slice-based processor was designed in parallel (category A). The proposed hardware design was divided into three main blocks, kernel implementation, inner products accumulation, and threshold comparison. The design employed Xilinx DSP slices, block RAMs, multipliers and exponential blocks. The simulation results demonstrated same accuracy of 93.6 % as

the software implementation with less processing time of 0.02 ms and reasonable hardware resource utilization.

The Synopsys Signal Processing Workbench (SPW) software was used in [91] to design a digital hardware implementation using available building blocks. A fixed-point number representation was used and a decision logic was utilized to implement a priority scheme required for multi-classification, targeting accuracy enhancement. The SPW generated VHDL code was analysed using the Xilinx Simulator. Compared to the software implementation, the hardware implementation achieved approximately 2.53 times speedup, while 16.2% accuracy loss was realized.

Also the Synopsys software was used in [65] (category A) to realize an ASIC implementation as discussed earlier.

The High-level Synthesis (HLS) design methodology has been recently used to decrease the FPGA development time and effort by using HLLs instead of HDLs. The Xilinx Vivado HLS tool [8] was exploited to implement a low-cost SVM IP [11-13] (our published papers) on a recent FPGA platform "Zynq SoC" with a hybrid architecture combining a processor with the FPGA in a single device [25]. An initial hardware/software co-design was implemented in [11] as an SVM accelerator to run the complicated processing task onto FPGA. It is considered the first hardware/software system implemented for SVM on FPGA/single device that exists in the current period. The proposed HLS design was successfully extended in [12] to implement an IP running full SVM algorithm onto FPGA part of the Zynq SoC. By using the available directives of the HLS tool, some hardware optimization techniques were simply applied to the proposed design as the pipelining (category A) and loop unrolling, which accelerated the original design/code by 9.9x, whilst design space exploration was investigated. The experimental results demonstrated high performance, low area and low power consumption, meeting embedded system constraints, while preserving classification accuracy rate without any loss. In addition, the implemented embedded SVM classification system on the recent hybrid Zynq SoC using the modern UltraFast HLS design methodology is a low-power system compared to other existing implementations in literature (total power=1.75W). Another similar design has been recently proposed in [13], which was based on using BRAM interfaces for passing required data instead of streaming data using the stream interface/bus with the DMA IP in [12]. Some extra resources were utilized compared to [12] and the power was slightly increased (2.7 W),

however both area and power demonstrated lesser values compared to some related implementations in the literature. Moreover, a high acceleration factor of 32x was achieved compared to a similar software implementation, whilst a 97.9% accuracy was achieved with a processing time of 11.46 μs at 250 MHz.

Table 2-6. Category E. Software Tool-based Architectures

SVM Type	Ref.	Kernel Function Type	Application Domain/ Dataset	FPGA Platform/ Board	Development Tool
Binary	[90]	RBF	Ultrasonic flaw detection	Xilinx Zynq-7000	Xilinx System Generator
	[11-13]	Linear	Melanoma detection	Xilinx Zynq-7 ZC702(XC7Z020CLG 484-1)	Xilinx Vivado 2016.1 Design Suite, Vivado-HLS
	[92]	RBF	ECG-based arrhythmia detection	Zedboard Zynq (xc7z020clg484-1)	Xilinx Vivado-HLS 2015.2, Xilinx SDK 2014.4, Petalinux 2014.4
Multiclass	[89]	Linear Gaussian	Persian handwritten digits dataset	Xilinx Virtex4 (XC4VSX35)	System Generator
	[91]	RBF	Multi-speaker phoneme recognition, TIMIT corpus	Xilinx Virtex-II (XC2V3000)	Synopsys SPW, Xilinx Simulator

Another HLS-based implementation with a hardware/software co-design was recently presented in [92], focusing on design space exploration. A systematic two-level methodology and prototype framework was proposed for realizing an efficient HLS-based SVM IP. The proposed methodology optimized first the original code structure exploiting data- and instruction-level parallelism (category A). Then, a second level optimization applied a derived set of design space pruning guidelines for refining the applied HLS directives. The proposed methodology was evaluated based on extensive analysis and validation results with a case study of ECG-based arrhythmia detection system. Experimental results showed execution latency gains of up to 98.78% compared to the default Vivado-HLS optimization of the original SVM code. A hardware/software co-design including the proposed IP was implemented on Zynq SoC for the detection

system that achieved speedups of up to 78x at 25 MHz from an equivalent software implementation.

2.5.3.6 Category F: Cascaded Classification-based Architectures

The cascaded classification architecture consists of multi classification stages (classifiers) designed in a cascading structure, targeting accelerated classification process. The cascaded scheme is depicted in Figure 2-5 [93], where the majority of data are rejected at the early stages that are based on simple classifiers with low complexity, leaving very little data to be classified in later stages that are more complex with higher accuracy. Consequently, a significant speedup could be achieved by using the cascaded classification architecture over using a single SVM classifier. This motivated implementing the cascaded architecture in hardware/FPGA to realize a real-time embedded classification system with high performance and low cost.

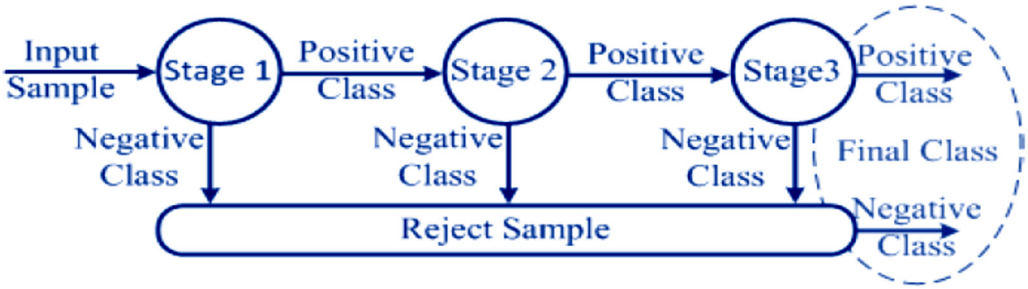

Figure 2-5. Cascade classifier scheme (3-stage cascade classifier) [93].

The first FPGA-based cascaded SVM classifier that exploited the custom-arithmetic feature of the device heterogeneous nature was presented in [94] (following their previous proposal [48] for accelerating the training phase). A parallel structure of multipliers with a pipelined adder tree was proposed for implementing the kernel computations (category A). The proposed data path was divided into two domains, fixed-point and single floating-point precision. The proposed heterogeneous architecture efficiently exploited the parallel processing power of the device heterogeneous resources and dynamic range diversities among the classification problem's features. Additionally, a two cascaded classifier scheme was implemented that combined a simple faster low-precision classifier with a complicated high-precision classifier of higher area cost. Compared to the software implementation, the implemented fully scalable heterogeneous architecture demonstrated

an acceleration of 2-3 orders of magnitude in addition to a speed-up of 7x compared to other previous implementations on FPGAs and GPUs.

As an extension to the previous work architecture [94], the FPGA reconfigurability feature was applied in [95] (category C), in order to switch from the low- to high-precision classifier in the cascade. Accordingly, higher performance was achieved, in addition to expanding the potential design space.

Another cascaded architecture was proposed in [93, 96], where a hardware reduction method was proposed to decrease area and power dissipation. This method followed the multiplier-less approach (category D) to implement the early simple cascade stages, where all multiplication operations were replaced with shift operations by rounding off the data to the nearest power of two values. A hybrid cascaded architecture (4-stage cascade classifier) was designed to combine both parallel and sequential processing (as a series of pipelined PEs (category A)) for simple early stages (three classifiers) and higher complexity SVM (one classifier) respectively. The implemented cascaded architecture demonstrated an average performance of 70 fps, whilst an acceleration of 5x was achieved compared to an implemented single parallel SVM classifier. By using the proposed hardware reduction method, resource utilization was reduced by 43% and 20% less power was consumed, while experiencing only a 0.7% loss in classification accuracy (84%).

Next, the previous work [93] was extended in [97] and [98], where a feature extraction mechanism (local binary pattern descriptors) was utilized within the proposed architecture to apply feature extraction before the final stage, aiming to enhance the detection accuracy. In [98], an additional novel response evaluation method was incorporated into the previously proposed architecture. Another type of classifiers (Neural Network) was used in the proposed response evaluation process to classify the responses of the early stages in the cascade. Accordingly, the number of tested samples was reduced before reaching the final high-complex classification stage, resulting in improving the classification speed. The proposed hybrid architecture with the proposed methods added demonstrated real-time processing of 40 fps with 80% detection accuracy [98] (similar behaviour for the accuracy loss as in [93]). Also, a reduction of 25% in area and 20% in power was achieved. Compared to their previous work [93], lower values were realized for performance and accuracy with higher area and power, however, a big test set of higher resolution images was evaluated, aiming to reach real-time embedded processing of online video classification.

Lately, a cascade SVM classifier has been introduced in [13] based on a proposed scalable multi-core architecture, which was formed from the implemented HLS-based SVM IPs (category A and E). A 2-stage cascade SVM classifier was implemented using a simplified design of the previously proposed SVM IP [13] (as described in category E section) using the control bus for passing required data replacing the BRAM interface. As a result of the simplified design, lower resource utilization and power consumption was demonstrated compared to a single SVM IP implementation, while enhancing the classification accuracy and speed (1.8 μs) as well as the diagnosis verification.

Table 2-7. Category F: Cascaded Classification Architectures

SVM Type	Ref.	Kernel Function Type	Application Domain/ Dataset	FPGA Platform/ Board	Development Tool
Binary	[94, 95]	Gaussian polynomial sigmoid	MNIST dataset	Altera's Stratix III (EP3SE260)	Altera tools
	[93, 96]	Linear polynomial	Face detection	Xilinx ML505 (Virtex 5-LX110T)	-
	[97]	Linear polynomial	Face and pedestrian detection	Xilinx Spartan-6 XC6SLX150T	-
	[98]	Linear polynomial	Face detection	Xilinx Spartan-6 XC6SLX150T	-
	[13]	Linear	Melanoma detection	Xilinx Zynq-7 ZC702(XC7Z020CLG484-1)	Xilinx Vivado 2016.1 Design Suite

2.5.4 Preliminary Analysis of Group 2

Most existing implementations targeted binary SVM classification (27 works), while only 10 works implemented multiclass classification and only one implemented both [86]. This is probably because implementing a multiclass classifier was based on combining binary classifiers, and so numerous works focused on implementing the basic binary classifier to be then extended if required. Also, various classification or detection applications have been targeted by the existing architectures/implementations, especially for different object detection. Some SVM implementations were tested using some available or specific datasets, while other implementations were general without validation for any specific application/dataset.

Implementing kernels with the inherent complicated calculations was the main challenge and focus for implementing SVM on FPGA. Some researchers used and implemented more than one type of kernels in their study. A total of 18 papers implemented the simple linear kernel, while others went for different non-linear kernels that included the basic dot-product calculation exists in the linear kernel, but with additional complexity (12 polynomial kernel and 4 sigmoid kernel existing implementations). Moreover, 14 papers implemented the Gaussian RBF kernel, which basically implements the complex exponential function. The proposed simplified hardware-friendly kernel was widely used as an alternative to the RBF for simple FPGA/hardware implementation (5 papers). Another digital kernel was proposed that is similar to the hardware-friendly kernel, but no FPGA or hardware implementation exists using that proposed kernel [85].

Most existing implementations were realized on old versions of FPGA technology using traditional design methods, even while using modern development tools. Only four researchers utilized the recent Xilinx series-7 devices [58, 66, 86, 90]. One unique research work was discriminated by exploiting the hybrid architecture of the recent Zynq-7 SoC platform and using the latest UltraFast HLS design methodology [11-13], followed by a recent study for design space exploration based on SVM implementation in [92].

Beside the standard FPGA parallel design, researchers used various hardware technique(s) for improving their SVM implementations, which was classified into six categories in this study. The standard parallel pipelined architecture (category A) was widely implemented by 24 papers. The multiplier-less approach (category D) was commonly applied in 10 implementations, which occupies high interest targeting multiplier-free architecture with hardware complexity reduction. Also, the parallel pipelined systolic array architecture (category B) was implemented in 6 papers with considerable interest for reducing multiplication complexity. Only 5 implementations exploited the powerful FPGA-based DPR feature (category C) for enhancing designs and hardware results. Moreover, only 8 existing implementations were developed by utilizing software system-design/development tools (category E), replacing the traditional design method using the HDL for decreasing FPGA development effort and time. The cascaded SVM classification (category F) was implemented on FPGA by only one research group [93-98] (6 papers) that introduced heterogeneous architecture and hybrid architecture implemented using different techniques of categories A, C, and D. Recently, another research group [13] added a cascade SVM implementation in the literature based on a

proposed multi-core architecture using different hardware techniques of categories A and E.

2.5.5 Group 3: SVM-based Applications Implementations

This group was constructed to demonstrate the usage of the SVM classification in a wide range of applications, in which the research papers focused more on the main application's implementation, rather than the classification purpose implementation as in group two. Different hardware architectures have been introduced in the literature for implementing algorithms including classification task targeting particular applications. This group consists of 13 papers to introduce some research work of different applications that deal with images (summarized in Table 2-8).

Many works for object detection have been developed on FPGAs, utilizing the SVM-based Histograms of Oriented Gradients (HOG) algorithm. A real time pedestrian detection was implemented on FPGA targeting high resolution images of 1920 x 1080 pixels to be processed at twice the pixel frequency [99]. The proposed design of the detection system was based on a time-multiplex method to construct multiscale detection, with a simply designed SVM that consists of a module for the dot-product calculation, an address decoder controlling the memory accesses, an adder for the intermediary summation, and an adder for the bias. Experimental results showed high performance of processing HD resolution images at 64 fps, which outperformed existing FPGA implementations by a factor of 4 with good resource utilization.

A real-time human detection using HOG algorithm with linear SVM classifier was implemented that based on adopting a binarization process [100]. As a result of the binarization process, all multiplication operations in the classifier were replaced with addition operations, which decreased hardware complexity. The proposed pipelined architecture achieved a processing rate of 293 fps with a detection accuracy of 1.97% miss rate and 1% false positive rate, in addition to a low power consumption rate of 353mW.

A processor-based approach was employed for implementing SVM-based HOG algorithm [101], which was based on a developed IPPro (Image Processing Processor) from Rathlin research project [102]. The designed multi-core IPPro system (coded by the help of Matlab Simulink model) achieved throughput of 2.6x compared to a handed coded

design targeting Xilinx Zed board, and 3.2x compared to relevant recent work in the literature.

An object detection processor based on a proposed simplified HOG algorithm was presented, which employed a simultaneous SVM calculation, targeting a reduction in amount of required computations [103, 104]. The proposed simultaneous SVM calculation architecture was designed as 15 parallel classification cores, where each managed 7 blocks of MAC operations and allowed the reuse of intermediate results. Accordingly, experimental results showed 99% reduction of cycle counts for the proposed SVM module compared to architecture without parallelization and pipeline. The proposed system achieved objects detection for SVGA resolution video at 72 fps with 40MHz. Also, the proposed simplified algorithm reduced required computation (from 89.2 to 2.25 GOPS (Giga Operations per Second)) with minimal memory usage and 3% decline in accuracy.

An embedded design using the HOG-SVM combination was presented for multiple object detection [105]. Single precision 32-bit floating point representation was used in the proposed implementation without making any simplification of the algorithm design, in order not to reduce the detection accuracy as occurred in some previous work. Xilinx single precision IP cores (adders and multipliers) were exploited to be designed with the longest latency in a fully pipelined structure with no need of external memory storage. Multiple binary SVM classifiers were instantiated in the proposed system to be capable of detecting multiple objects. The implemented system was capable of detecting three different objects in 640x480 images at 60 fps (outperformed the speed of the software implementation), with a computational performance of 9.47 Giga Floating Point Operations per Second (GFLOPS).

An FPGA-based hardware design for head-shoulder detection was presented [106], which was based on Local Binary Patterns (LBPs) for feature extraction and SVM for classification. In addition, foreground object detection was exploited for improving detection accuracy. A different scheme of unrolling loops required by SVM computations was used for the implementation to meet the sequential flow of data in the FPGA. And so, the SVM computation module was designed based on a FIFO, multiple multipliers, and a pipelined adder tree. The integrated system was implemented on Xilinx Virtex 6 FPGA, demonstrating real-time video stream processing of 640x480 images at 60 fps, with a 15% drop of detection accuracy.

A digital hardware architecture was presented for face detection in IR images based on LBP, SVM algorithms, and generating bounding boxes of detected faces [107]. Concerning SVM module implementation, the dot-product calculation was implemented as a pipelined sum using integer adders replacing multipliers (no DSP slices) and buffers were exploited for decreasing memory requirements. The implemented system outperformed software implementation with processing 640x480-pixels video at 313 fps with a false positive rate between 4.5 and 7.2%. Also, low resource utilization was achieved of less than 25% with 266 mW power consumed.

A hardware implementation for pedestrian detection algorithm was presented that was based on a sliding window-based SVM classifier using feature covariance matrices as descriptors [108]. The SVM classifier was designed as 15 units processing in parallel, where 147 scalar products and accumulations were computed (that are proportional to the number of the descriptor features). The implemented detection system showed encouraging results for hardware implementation with a maximum frequency of 213 MHz and 9% loss in accuracy.

A complete parallel hardware architecture targeting real-time image classification (object detection) using SVM model was implemented on FPGA, which was based on Scale-Invariant Feature Transform (SIFT) and Bag of Features (BoFs) algorithms for feature extraction [67]. Regarding SVM implementation, the proposed architecture of the SVM module was designed by using multipliers and accumulators, exploiting parallelism in computing the RBF kernel to accelerate the processing time. Also, the Xilinx IP core CORDIC block was used for implementing the exponential function (sum of sin and cos). Two different challenging datasets with high image variation were used to evaluate the proposed hardware system in which 85% and 78% classification accuracy were achieved for Caltech-256 and KUL Belgium traffic sign datasets respectively, with less than 3% loss in accuracy from the software implementation. In addition, a speedup of 5.7x was achieved for classifying 640x480 images compared to software implementation, with reasonable hardware resource utilization compared to related implementations.

C. Kyrkou et al. [109] presented a hardware architecture for real-time SVM-based object detection using a proposed depth and edge accelerated search method, targeting improvement in speedup and detection accuracy. The proposed architecture exploited the proposed SVM processing architecture in their previous work that was based on an array of processing elements [69], and the reduced set method was employed to decrease the

number of support vectors used for reducing memory utilization and increasing classification accuracy. The implemented system achieved real-time frame rates of 271, 42, and 23 fps for 320x240, 640x480 and 800x600 image sizes respectively, with a 52% decline reached for the false-positive rate.

An FPGA implementation of a CAD system for skin cancer was introduced that was based on feature extraction and SVM classification for histo-pathological images [110]. A block diagram of hardware architecture of the CAD system flow was presented, which used the RAM memory for storing input image (no hardware design details). Simulation results of the proposed system implemented on Xilinx Virtex-7 FPGA was presented for applying CAD system phases on an input test image.

2.5.6 Critical Analysis of Group 2 and 3

Regarding parallel architectures targeting acceleration of the SVM classification phase, both group two and three are analysed in this section.

Many works exploited the FPGA-based parallel systolic array architecture in their implementations for achieving high level of parallelism [69, 71, 73, 74, 86, 111], in which this architecture offers a configurable modular platform for parallel processing with efficient memory management and data flow that allows for scalable design and reduced complexity. These implementations achieved good results of classification speedups that mostly outperformed software implementations on GPPs/CPUs.

Table 2-8. Group 3: SVM-based Applications Implementations

Ref.	Classification Implementation Method	FPGA Platform/ Board and Development Tool	SVM Type	Kernel Function Type	Application Domain / Dataset	Important Results
[99]	Multiscale and time-multiplex parallel architecture	Xilinx Virtex-5 (XC5VFX200T)	Binary	Linear	Pedestrian detection INRIA dataset	64 fps
[100]	Pipelined binarization-based adder architecture	Xilinx Spartan-3e (XC3S500E)	Binary	Linear	Human detection INRIA dataset	293 fps

[101]	(IPPro) Processor-based	Xilinx Zed board (Zynq 7020) Xilinx ISE 14.6	Binary	Linear	Object detection	Throughput 3.2x
[103], 85]	Pipelined parallel architecture	Altera Cyclone IV EP4CE115	Binary	Linear	Object detection	72 fps
[105]	Fully pipeline of Xilinx single precision IP cores	Xilinx ML605 (Virtex 6 XC6VLX240T)	Multiclass	Linear	Multiple object detection	60 fps
[106]	Pipelined architecture	Xilinx Virtex 6 (XC6VLX 240T-1FF1156) Xilinx ISE 13.4	Binary	Linear	Head-shoulder detection	60 fps
[107]	Pipelined multiplierless	Xilinx Spartan-6 (XC6SLX45) Synopsys Synplify, Xilinx ISE	Binary	Linear	Face detection	313 fps
[108]	Parallel processing units	Xilinx Virtex-4 LX80	Binary	Linear	Pedestrian detection, INRIA dataset	9% accuracy loss
[67]	Parallel architecture	Xilinx ML509 (Virtex 5 LX110T) Xilinx ISE	Multiclass	RBF	Caltech-256 dataset, Belgium Traffic Sign dataset	Speedup 5.7x 3% accuracy loss
[109]	Parallel systolic array architecture	Xilinx ML505 (Virtex 5-LX110T)	Binary	Polynomial RBF	Object detection	271, 42, 23 fps
[110]	Block diagram-based	Xilinx Virtex-7 (XC7VX980) Xilinx ISE 14.2	Binary	Linear	Skin cancer detection	-

This table summarizes all reviewed articles in group 3. For each reference in column one, the implementation method of the classification phase is reported in column two, followed by the FPGA platform/board and development tools used for the implementation. Then, the type of the SVM classifier (binary/multiclass) and the kernel used are stated in columns four and five respectively, followed by the used application/dataset and finally some important results are summarized in the last column.

In addition, other works employed the FPGA-based Dynamic Partially Reconfiguration (DPR) feature that allows reconfiguring dynamically selected areas on FPGA (on-the-fly), while other parts are still working [71, 73, 74, 95, 111]. These implementations achieved more flexibility and design space expansion besides gaining speedups. Also, the

hardware implementation in [74] achieved a significant reduction in power consumption as a result of applying the difference-based partial reconfiguration technique.

Interestingly, many studies adopted the multiplier-less approach [86, 93, 96, 98, 100, 107, 112], where computational intensive multipliers required for computations are replaced with conventional adders and/or shifters in order to decrease the hardware complexity. Similarly, others [77, 79-81, 83, 85] utilized the hardware-friendly kernel function that was proposed in [113] for simplifying the hardware design by using the simple shift and add operations instead of resource consuming multiplications. Some of them employed the common CORDIC iterative algorithm for implementing the hardware-friendly kernel computations, which is also based on using only shifters and adders [80, 81, 83], while other implementations used the CORDIC algorithm for solving the exponential function of the kernel [55, 67, 85, 89]. As a result of multiplier-less implementation, significant reduction in hardware resource utilization was achieved by [77, 81, 86, 93, 96, 98], in addition to remarkable power consumption decrease demonstrated in [86, 93, 96, 98, 100, 107, 112].

Moreover, the common pipelining technique was exploited by most previous hardware designs, taking advantage of the parallel processing capabilities of the FPGA that led to throughput increase of the implemented classification process. Some researchers designed a pipeline stage for common and shared multipliers required for computations to decrease usage of duplicate multiplications [60, 62, 63], achieving lower hardware resource utilization. Additionally, some pipelined designs like in [54, 55, 105] were based on exploiting the embedded IP cores in the FPGA device for efficient resource utilization. Furthermore, the parallel pipelined (adder) tree structure was used by various designs aiming to reach processing speed improvement [55, 79, 83, 93-96, 98].

Some works [15, 21] in the surveyed papers targeted comparisons between FPGA and GPU implementations, and their performances were compared with the software. Apart from using the traditional HDL approach in many research works for designing hardware implementations, some other works used available powerful tools on the market which simplified the process of hardware designs. One of the common tools is the Xilinx System Generator that offers high-performance system modelling and automatic code generation from Matlab/Simulink, and it was exploited in [89] for their implementation.

In addition, some works have studied the quantization impact on the classification accuracy rate resulted from applying the fixed-point number formatting (bit-width precision) in the hardware implementation [58, 62, 63, 67, 79, 97]. They aimed to reduce hardware area, while maintaining high accuracy. Moreover, Afifi et al. [12] have studied the trade-off between the speed/acceleration and hardware resource utilization/area with applying some different hardware architectures (HLS optimization directives), followed by another analysis study for similar trade-off presented in [92]. As a result of applying different techniques for reducing hardware complexity and gaining acceleration, some implementations reported some loss in the classification accuracy rate [56, 67, 91, 93, 96-98]. Furthermore, many works reported classification speedup results compared to similar software implementations [13, 58, 67, 71, 72, 89-92, 94, 95], and few works presented a comparison with some related FPGA implementations in the literature [12, 13, 67, 94, 95, 97, 98].

2.6 Results Comparison and Discussion

Table 2-9 is presented in this study for reporting and comparing some results provided by selected reviewed papers (16 papers from group 2). Samples of results have been carefully extracted from the papers, while some parameters were either unclear or difficult to easily define in the proposed table or not provided/applicable. Also, some papers were excluded because of few or no hardware implementation results have been explicitly provided. This comparison includes different architectures and implementations for various SVM types and kernels implemented for different applications.

Table 2-9. Comparison of Results for the Selected Reviewed Papers

Ref.	FPGA Resources				Power (W)	FPGA	#SVs	Dim	Accuracy %	Processing time/speed	Frequency (MHz)	Category	SVM
	Slices	LUTs	BRAM	DSP									
[54]	58688	-	800	768	-	Virtex-6	760	128	-	0.9 µs	370.096	A	Binary RBF
[21]	59208	122637	2049	-	15	Virtex-5	16	-	-	0.02 s	200	A	Binary Gaussian
[55]	12674	41135	132	64	-	ML505	2048	2048	-	712.66 µs	92	A	Binary Linear
[62]	42600	250552	1001	502	-	Stratix-IV	474	500	98	47.2 ms 21.2 fps	100	A	Multiclass Linear
[66]	7228	5698	14.5	8	1.693	Zynq Z-7010	-	-	96.4	25 ms	100	A	Multiclass Linear
[67]	9646	38179	60	52	-	ML509	100	500	82	0.25 ms	50	A	Multiclass RBF
[69]	23220	8887	74	64	-	ML505	818	400	76-78	40-122 fps	100	B	Binary RBF Polynomial
[72]	1810	1705	21	21	-	ML 403	1024	20	-	7.34 µs	142.9	A B C	Binary Linear
[74]	461	461	-	7	2.021	Virtex-6	192	120	-	-	-	B C	Multiclass Polynomial

[86]	33360	33360	-	0	1.703	Virtex-7	74	-	-	-	-	B D	Multiclass
[12]	2676	2267	12	5	1.752	Zynq-7	248	27	-	141.38 µs	100	A E	Binary Linear
[89]	11589	9141	99	81	-	Virtex4	18-35	-	98.67	0.27 ms	151.286	E	Multiclass Gaussian
[90]	21305	14028	106	152	-	Zynq-7000	1024	-	93.6	0.02 ms	-	A E	Binary RBF
[91]	6373	11943	-	64	-	Virtex-II	-	-	60.6	14.18 ms	42.012	E	Multiclass RBF
[93]	13038	31854	131	59	3.2	ML505	254	400	84	70 fps	84	A D F	Cascaded Linear Polynomial
[98]	20153	35532	256	59	4.9	Spartan-6	122	400, 1062	80	40 fps	70	A D F	Cascaded Linear Polynomial
[13]	4304	3414	2	10	1.74	Zynq-7	200	27	97.9 72.5	1.8 µs	250	A E F	Cascaded Linear

The comparison is based on the table attributes of different FPGA resource utilization, power consumption in watts, used device/FPGA, number of SVs, dimensionality, classification/detection accuracy in %, processing time or/and speed in frame per second (fps), frequency in MHz, architecture category(s), and SVM type and kernel implemented.

From Table 2-9, it is clear that low reasonable hardware resource utilization was achieved in [12, 13, 72, 74], where [74] is the least. From the limited number of works that reported the power consumption results, only four implementations [12, 13, 66, 86] achieved low values of around 1.7 watts, which were realized on the latest FPGA generation, and a high power of 15 W was reported in [21]. The highest number of SVs of 2048 and highest dimensionality of 2048 were considered in [55]. Almost all introduced classification accuracies are realistic and greater than 75%, while only one [91] has achieved low rate of 60.6% with 16.2% loss in accuracy. The least processing time of 0.9 µs was achieved with a high frequency of 370 MHz in [54], while others realized different real-time values of µs and ms units (47 ms is the highest [62]). Some papers reported the processing speed in terms of frame rate, where real-time image processing was achieved from 40 fps to the highest 122 fps achieved in [69]. Finally, a variety of good acceptable results was achieved by various presented implementations, where 8 papers included more than one architecture (categories) in their implementations. Still more optimization is required to achieve the best/optimum trade-off between different parameters discussed in this comparison study.

2.7 Limitations and Challenges of Existing Works

This study reveals key limitations and existing challenges faced by the surveyed works:

- Various simplification methods were used for reducing the hardware complexity and achieving efficient/optimized hardware implementation results (such as the multiplier-less method), which affected the classification accuracy.
- Fully parallelized processing was not effectively addressed in case of implementing SVM with very large-scale/dimensionality, which requires excessive number of processing cycles when implemented over limited hardware resources.
- In case of implementing SVM with very high number of SVs for a large-scale problem/application, the memory storage and management problem was not effectively studied.
- Many SVM implementations were designed targeting specific application, which are not capable of any extensions and not easily adapted for various applications.
- Some architectures were not simply designed, which have shortage in meeting flexibility and scalability conditions of realizing an efficient embedded system.

- Difficulty of meeting all vital constraints of embedded systems such as real-time, high performance, low cost, low power and others.
- Very few implementations reported the power consumption results, which is very challenging to achieve low-power embedded system.
- Most existing implementations were implemented on old versions of FPGAs and designed using traditional methods and tools/technologies.
- The trade-off exists between classification performance/speed and hardware resource utilization needs more optimized solutions/methods.
- The primary trade-off exists between meeting embedded system constraints and preserving/maintaining classification accuracy rate.

2.8 Leading Research Groups

From reviewing the presented papers, we identify two leading research groups that are actively working on implementing SVM on FPGA.

Group 1: One unique research group is exclusively working on FPGA-based cascaded SVM classification architecture for object detection [93-98] (category F), however, we have recently implemented a cascade SVM classifier for melanoma detection in this research [13]. This research group proposed first a heterogeneous architecture of low-precision and high-precision classifiers in a cascade (the first cascade SVM on FPGA), where DPR was then applied for the switching (category C). Next, a hybrid cascaded architecture was proposed for combining parallel and sequential processing modules in a pipelined design (category A). In addition, a hardware reduction method (category D) was proposed that is based on replacing expensive multipliers with simple shift operations, achieving reduction in area and power consumption. They lately proposed an optimized architecture by adding a response evaluation method (NN classifier) to improve classification prior to the final classification stage in the cascade, boosting the overall classification. Also, they applied a feature extraction phase before the final complicated stage for improving the accuracy rate. Good results were achieved for real-time processing of high resolution images in terms of fps. But, their implemented architectures that were applied for object detection suffered from slight loss in detection accuracy rates. Also, the power consumption results that are greater than 3 watts are considered high for embedded system deployment. These also prove the existing challenging trade-off between efficient classification accuracy and meeting significant

embedded system constraints, in which they managed to present an adequate trade-off in their latest publication.

Group 2: Another research group led by H. Hussain [71-73] is working on the parallel systolic array implementation (category B), exploiting the advantage of the powerful DPR feature in the FPGA, aiming to reach flexibility and scalability (category C). They implemented various copies with different SVM parameters to fulfil user requirements at run-time and great speedup results were achieved compared to software implementations on GPP. Then, these implementations were used to realize a multi-core architecture of different SVM copies with applying DPR, which offers flexibility of changing cores on-the-fly during run-time to be used then for ensemble classification. But, no specific application was applied to employ and use the proposed architectures and so no classification accuracy was measured. They lately implemented a DPR-based adaptive multi-classifier to be used as an SVM or KNN classifier with different parameters saving more space in the device, which could be extended in the future as an ensemble classifier for classifying bioinformatics microarray data. But also, no real application was applied for verifying the usage and classification of the implemented multi-classifier. From their DPR implementations, 8x reduction in reconfiguration time was gained over reconfiguring the whole FPGA device. Finally, they gained flexibility and adaptability with good speedup results in reconfiguration time, but with no verification of the classification functioning. Also, no optimization methods were used for reducing hardware resource utilization and no power consumption measurements were presented for meeting embedded system constraints.

2.9 Concluding Remarks

This review is a unique survey study that presented and classified different hardware architectures used for implementing SVM classifiers on FPGAs. To the best of our knowledge, this is the first comprehensive survey of this topic, and it can drive future hardware implementations of SVM. In addition, it provided critical analysis and detailed comparison with discussions and identified leading research groups, limitations, challenges, and research gaps. In conclusion, the main research gap is finding an optimum solution for the challenging trade-off between achieving efficient classification accuracy and meeting important embedded system constraints of high performance, low area, cost and power/energy consumption.

Most existing research works in the literature focused on implementing the SVM classification phase online on FPGA after executing the training phase offline in software. Different FPGA-based hardware techniques have been utilized for implementing the SVM classifier [9]. Binary classifiers using linear kernel was mostly targeted for the hardware implementations. Most existing hardware designs utilized the common pipelining technique, exploiting the parallel processing capabilities of the FPGA in order to accelerate the classification task. Other hardware designs were implemented using parallel systolic array architecture and multiplier-less approach, while old versions of FPGAs were utilized for implementations without considering crucial embedded system constraints like low power consumption constraint.

From reviewing existing works in the literature, we identified a unique research group (group 1) that is exclusively working on implementing cascaded SVM classification architecture on FPGA, targeting object detection [93-98]. Their implemented architectures demonstrated real-time processing of high resolution images, but, suffered from slight loss in accuracy rates. Moreover, the power consumption results were high for embedded system deployment. Another research group led by H. Hussain [71-73] (group 2) is exploiting the FPGA-based DPR feature with implementing a parallel systolic array architecture, targeting classification of bioinformatics microarray data. They also proposed a multi-core architecture of different SVM copies. Nevertheless, no specific application was employed for the proposed architectures and also no classification accuracy was reported as well as not all implementation results were provided.

Subsequently, it is concluded that the main challenge is the difficulty of meeting important embedded system constraints of high performance, flexibility, scalability, and low levels of area, cost, and power consumption, while achieving reliable effective classification system with a high classification accuracy. Many architectures have been developed without taking into consideration critical embedded system constraints, especially the challenging constraint, low power consumption that was measured for very few numbers of previous implementations. A challenging trade-off has been observed that exists between meeting embedded system constraints and achieving efficient classification accuracy rate. Also, very limited work exists for hardware implementation of the cascade SVM classifier as well as using the powerful FPGA-based DPR technique. Additionally, most existing implementations in the literature were realized on old versions of FPGAs. To the best of our knowledge, no FPGA implementation of the

(cascade) SVM classifier exploiting the benefit of the hybrid architecture (hardware/software system) of the Zynq SoC exists in literature. Additionally, almost all previous FPGA implementations were designed utilizing the traditional time-consuming HDL that necessitates expert hardware designers. Nevertheless, the recent UltraFast HLS design methodology is newly recommended to simplify FPGA designs. Furthermore, no FPGA implementation exists in the literature for classifying melanoma clinical images using SVM classifier.

Specifically in this research, we are addressing such limitations, challenges and existing gaps identified above from this unique survey study, aiming to be identified as the third leading research group contributing to this area. Therefore, this research proposes a hardware implementation of an optimized SVM classifier on Zynq SoC by utilizing the HLS design methodology, targeting melanoma detection with high performance at low cost. In addition, a hardware implementation of a novel dynamic and adaptive cascade SVM classifier is proposed exploiting the unique DPR technology of the FPGA, targeting flexibility, scalability, applicability and adaptability, while meeting critical embedded system constraints.

CHAPTER 3 Proposed Hardware Architectures, Designs and Implementations

3.1 Introduction

In this chapter, the selected FPGA platform and system development tools used for the implementations are introduced. Then, a primary hardware/software co-design is proposed for an initial implementation of SVM on FPGA. Next, the hardware design is extended to realize a full SVM core running on FPGA using stream bus, then a similar design based on BRAMs is proposed. Finally, a multi-core architecture is proposed to implement a cascade SVM on FPGA, and then the DPR feature is applied to propose a dynamic cascade SVM architecture.

3.2 FPGA Platform and System Development Tools

The recent FPGA platform "Xilinx Zynq-7000 All Programmable System on Chip (SoC)" is chosen for our SVM implementation to exploit the newest technology and reach a powerful efficient embedded system [25]. The Zynq SoC is characterized by its hybrid architecture as depicted in Figure 3-1 [25], which significantly simplifies the embedded system development process. The hardware programmability of an FPGA is combined as a Programmable Logic (PL) with an ARM Cortex-A9 dual-core as a Processing System (PS) in a single SoC. Hence, the Zynq SoC is an ideal platform to develop a high performance smart embedded system for realizing a cost-effective medical scanning hand-held device.

The latest software tool "Xilinx Vivado Design Suite" is selected as being a powerful system-design tool that simplifies embedded system design on a single device based on integrating an FPGA within a single SoC [114]. Vivado Design Suite applies the UltraFast Design Methodology that employs Design Rule Check rules, which provide guidance on HDL code and Xilinx Design Constraints (XDC) in order to improve the quality of the design earlier in the flows and avoid problems downstream when iterations would be costlier. Also, proven templates for specific HDL code and XDC constraints that enable optimal-by-construction code are provided by the tool. The Vivado design suite offers a powerful IP and system-centric integration with fast verification, reaching significant

acceleration for design integration and implementation (see Figure 3-2 [114]). So, the Vivado tool with the UltraFast Design Methodology is exploited for our embedded system development, because of its benefits of faster compile times and convergence, more predictable results, and accelerated time-to-market.

In addition, Xilinx Vivado suite is an efficient design tool that employs the modern UltraFast High Level Synthesis (HLS) design methodology. The HLS methodology is characterized with simplifying FPGA programming via using the high-level language replacing the traditional HDL [8]. The design flow of the Vivado HLS tool is depicted in Figure 3-3 [115]. The HLS method is highly recommended as it significantly decreases the FPGA development effort and time. Consequently, the Xilinx Vivado HLS tool is utilized to implement an SVM IP to be integrated into a single SoC for reaching an online embedded classification system running on the recent Zynq SoC.

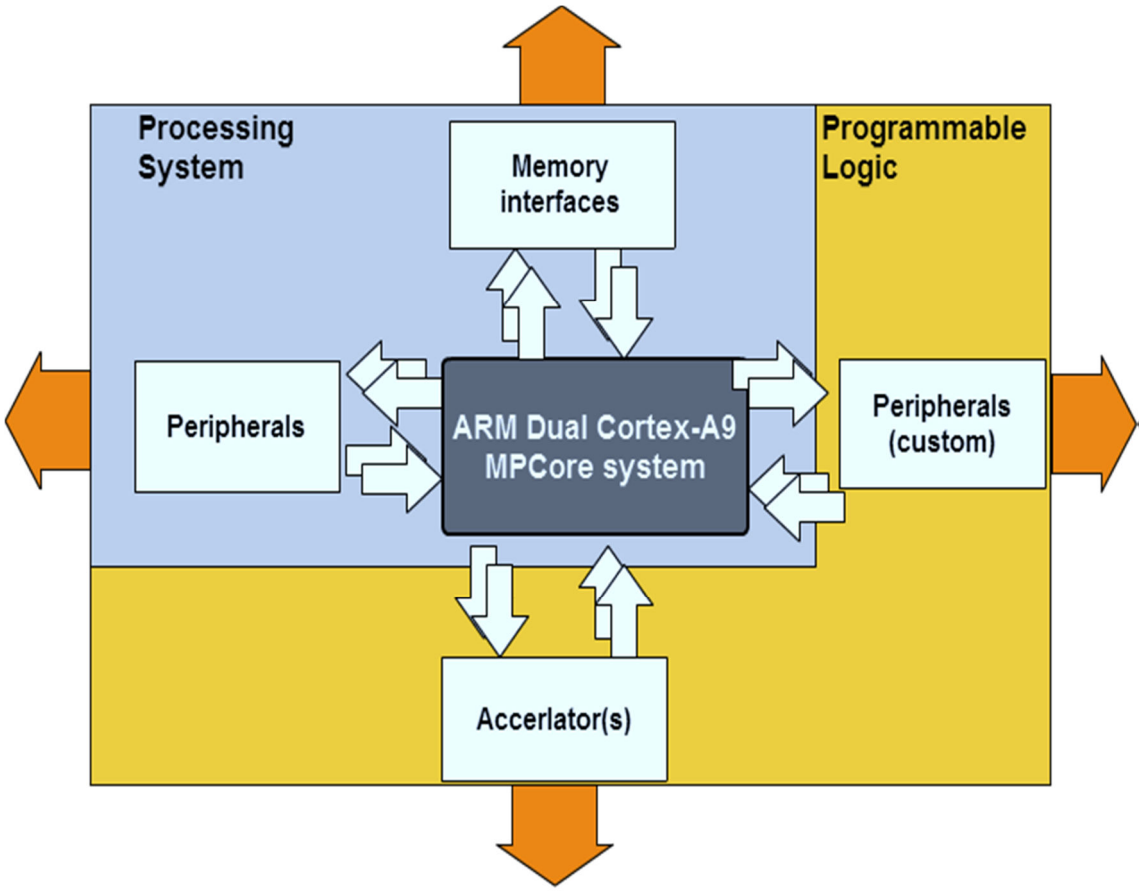

Figure 3-1. An overview of the Xilinx Zynq-7000 SoC [25].

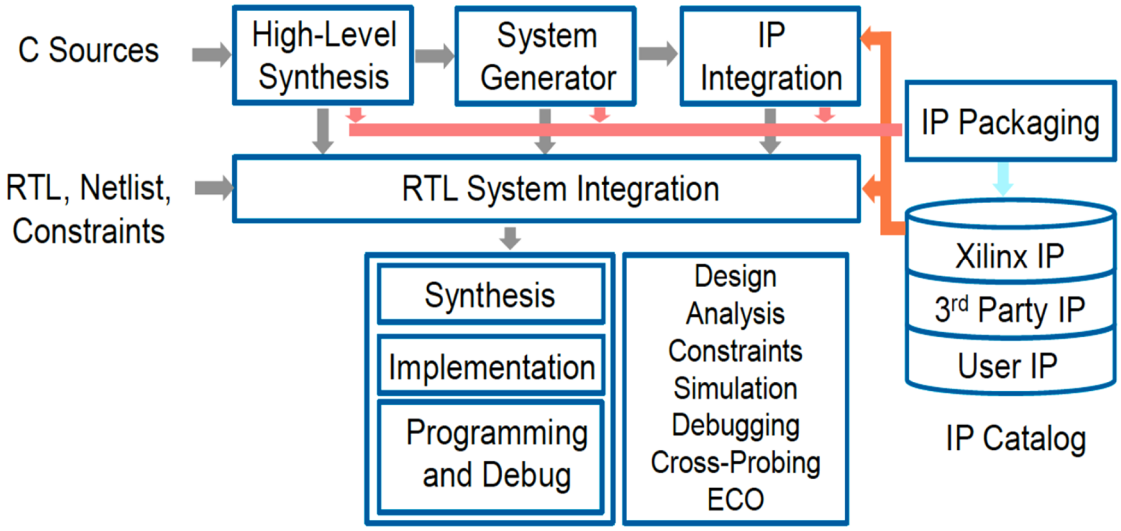

Figure 3-2. Xilinx Vivado design suite flow for system design [114].

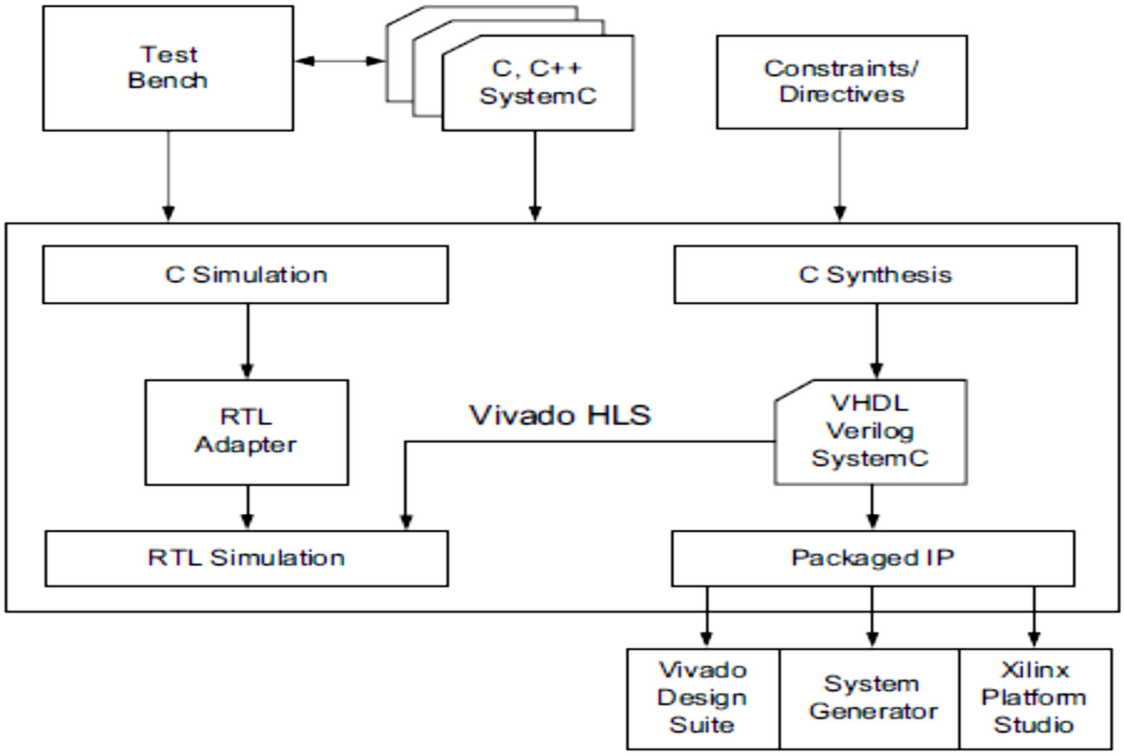

Figure 3-3. Xilinx Vivado HLS design flow [115].

3.3 Proposed Hardware/Software Co-design

A hardware/software co-design is proposed using the recent HLS design methodology to implement an HLS IP/accelerator for a binary SVM classifier with a linear kernel function

[11]. The proposed design implements the main decision function (2.8) with the linear kernel function to be rewritten as in (3.1). The most compute-intensive task of the SVM classification algorithm is implemented and accelerated onto the hardware, which is the dot-product calculation exists in the main decision function (3.1).

$$F(\vec{x}) = sign\left(\sum_{i=1}^{SV} \alpha_i y_i \left(\overrightarrow{x_i} \cdot \vec{x}\right) - b\right) \quad (3.1)$$

The hardware/software co-design technique is applied to partition the SVM algorithm efficiently between software and hardware, reaching acceleration of the accumulation of vectors multiplication part via running on hardware. Equation (3.1) is converted into two main equations (3.2) and (3.3). Equation (3.2) computes the summation of all Support Vectors (*SV*s) multiplied by the corresponding *αy* to be stored in an accumulated array/vector "*AC*" (*α, y* and *b* are parameters specified from the training phase). This computation is calculated offline by a proposed software function that implements (3.2) on software. Then, the resulted *AC* array is used in (3.3) as part of the dot-product calculation with the features of the test instance *x* to calculate the distance value, which is performed on hardware. Then, the resulted distance value from hardware is used by a proposed software function that implements the rest of calculations in the decision function (3.3) on software for classifying based on the sign value. The final result *F(x)* of the SVM classifier has two possible values equal to 1, or -1, which corresponds to melanoma class, or non-melanoma class respectively.

$$\overrightarrow{AC} = \sum_{i=1}^{SV} \alpha_i y_i \overrightarrow{x_i} \quad (3.2)$$

$$F(\vec{x}) = sign\left(\left(\overrightarrow{AC} \cdot \vec{x}\right) - b\right) \quad (3.3)$$

The structure of the proposed hardware/software system on Zynq SoC is depicted in Figure 3-4, where the HLS IP module is developed first with Vivado HLS tool and then extracted to be implemented as a hardware coprocessor onto the PL part of the Zynq SoC. By using the Vivado HLS tool with C/C++ language, a top function module is designed as an HLS IP for executing the complicated "for loop" of the dot-product calculation on hardware. However, the rest of the code is designed as a test bench (software program) running on software for testing the HLS IP and calculating the final classification decision *F(x).* This proposed software program calls the designed top-function and passes the two required vectors/arrays (*AC* and *x*) as formal parameters, after reading data of the trained

SVM model from a file and applying some processing and calculations for preparing required data.

The designed HLS IP basically implements the proposed pseudo code that is illustrated in Figure 3-5. The designed IP depends on the number of features in order to implement any SVM model. The proposed IP receives required data from the proposed software program through an input stream interface to be stored in two main arrays. The first array *array_AC* stores the accumulated vector *AC* in (3.2). The second array *array_test* holds features of the test instance *x* to be classified. The two arrays are processed to calculate the dot-product calculation for producing the distance value of the classifier. Then, the calculated distance value is finally streamed in the output stream interface to be used by the software program for generating the final SVM classification decision.

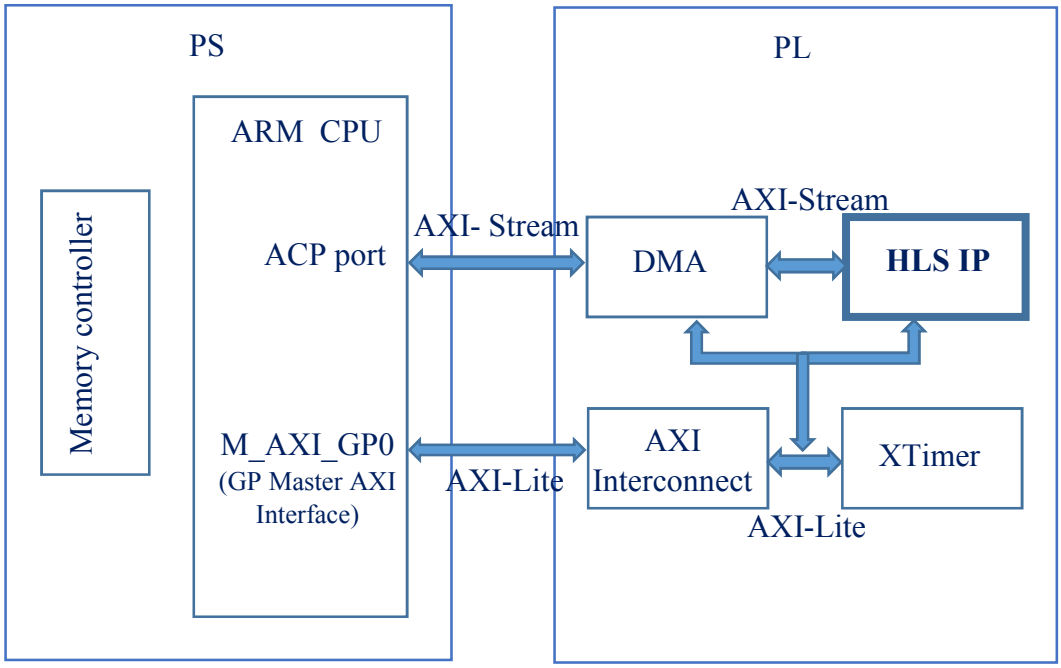

Figure 3-4. The block diagram of the proposed system on Zynq SoC.

The HLS tool provides various directives to be applied for the IP to assign different interfaces and apply other hardware techniques [115]. The AXI4-Stream interface/directive is utilized as the input/output stream interface of the designed function module for streaming the data between the ARM processor in PS part of the Zynq SoC and the hardware IP in the PL part. Additionally, the AXI-lite bus is allocated as a control bus of the module in order to control the designed IP core and other connected cores in

the system, as well as, controlling the data flow of the system through communicating with the ARM processor/PS as shown in Figure 3-4.

In addition, the HLS tool provides various optimization techniques as directives to be applied to the proposed module/IP, aiming to reach satisfactory optimization levels, while investigating design space exploration [115]. In order to optimize loops, pipelining and unrolling techniques are used for enhancing data latency and throughput. Pipelining allows concurrent execution of operations within a loop, which reduces the Initiation Interval (II) to enhance the data throughput (II is the same as the throughput and is used exclusively with pipelines). The HLS tool tends to pipeline a loop with an II of the minimum possible number of clock cycles starting with one, trying to achieve the fastest possible design.

```
Setup of IO Stream Bus
Define number of features+1 as F

Function: Classify (in_stream, out_stream) {
FOR each feature
        Read streamed data
        Convert it to float
        Store into array_AC [F]
END FOR
FOR each feature
        Read streamed data
        Convert it to float
        Store into array_test [F]
END FOR
FOR each feature
        Distance_value += array_AC [F] * array_test [F]
END FOR
Convert Distance_value to float
Write streamed data
}
```

Figure 3-5. The proposed pseudo code of the HLS IP.

The unrolling technique creates multiple independent operations instead of a single collection of operations. Unrolling the loops decreases latency, but increases hardware resource utilization, as well as, power consumption. So, loops can be partially unrolled under a defined factor, in order to improve upon area and power. Both pipelined and

unrolling designs are applied to the designed SVM HLS IP and investigated based on synthesis results, aiming to find an optimized hardware solution.

Based on HLS synthesis results, 333 clock cycles are required for processing without applying any optimization directives for the loops in the proposed function. However, 169 and 204 clock cycles are required when using unroll and pipeline directives respectively. Regarding hardware resource utilization estimated from synthesizing, 1514 FF slices, 1794 LUTs and 5 DSP48Es are utilized from using unroll directive, whilst 650 FF slices, 878 LUTs and 5 DSP48Es from using pipeline one. Actually, there is a trade-off between the data throughput and area, as by using the unroll directive more utilization of FF and LUT resources takes place than using the pipeline one. However, by using the recent Zynq SoC with large optimized size, the main aim is to increase data throughput with low resource utilization. Accordingly, the unrolling technique is exploited for further implementation, targeting throughput improvement.

After successfully synthesizing the code and applying optimization and interfaces directives, the HLS IP core for the classifier is successfully co-simulated and exported as an RTL implementation using the Vivado HLS tool. Then, the exported HLS IP is integrated into the proposed system that is designed by the help of Vivado Design tool as shown in Figure 3-4.

The HLS IP core is connected as a Zynq coprocessor in the PL part to the ARM processor in PS part through an ACP (Accelerator Coherency Port) passing by a DMA (Direct Memory Access) controller core/IP. The DMA controller IP is used to control the data transfer between the ARM processor and the HLS IP through the AXI4-Stream protocol [116]. The ACP is a 64-bit AXI slave interface on the Snoop Control Unit (CU), which provides an asynchronous cache-coherent access point directly from the Zynq PL to the PS with low latency path. Also, an AXI-Timer IP is instantiated for measuring the number of clock cycles required by the cores for performance comparisons.

Finally after synthesizing, implementing and generating the bitstream of the proposed Zynq SoC (Figure 3-4) using the Vivado tool, it is exported for the Software Development Kit (SDK) to be ready for running an application on Zynq device. The proposed software program (test bench) designed in the HLS tool is adopted to be used in the SDK tool (implemented in C) for realizing the implemented SVM classification system on the Zynq SoC.

3.4 Proposed Hardware Design of SVM HLS IP

3.4.1 Proposed SVM HLS IP on Zynq PL

Based on our previous hardware/software co-design proposed in section 3.3 [11], a hardware design extension is proposed in this section to implement a full SVM onto FPGA. In the initial implementation [11], the most compute-intensive task of the SVM classification algorithm, the dot-product calculation was implemented and accelerated onto the hardware. So, the hardware design is extended in this proposal to reach an online full SVM classifier running on the Zynq SoC, offering HPC and low cost.

Similarly by exploiting the Vivado HLS tool with C/C++ language, a top function module is designed as an HLS IP that computes the decision function (3.1), to implement a binary SVM with linear kernel on Zynq PL. The designed IP basically implements the proposed pseudo code that is illustrated in Figure 3-6. The designed IP depends on the number of SVs and features both, in order to implement any SVM model. The designed IP receives needed data via an input stream interface to be stored in three main arrays. One 2D array *array_SVs* holds features of *SVs*. The other 1D array *array_αy* has *αy* of each SV as well as *b* parameter and a third 1D array *array_test* for storing features of the test instance *x* to be classified.

For simplifying the hardware design and mapping, the required computations in (3.1) are divided into two main parts in the algorithm to implement both equations (3.2) and (3.3), aiming to reduce the hardware complexity. The first task is the summation of all SVs multiplied by the corresponding *αy* to be stored then in an accumulated array *array_ACC*. The second task is the summation of the dot-product between the accumulated array and the features of the test instance to calculate the distance value of the classifier. Then, the calculated distance value is finally classified based on the sign value to give the final SVM classifier output of the class (*th* is the threshold value determined through the validation phase). We have two possible outputs exactly equal to 1, or -1, which corresponds to melanoma class, or non-melanoma class respectively.

```
Setup of IO Stream Bus
Define number of features+1 as F and SVs+1 as SV

FOR each SV
        FOR each feature of the SV
                Read streamed data
                Convert it to float
                Store into array_SVs [SV][F]
        END FOR
END FOR
Read streamed data
Convert it to float
Store into array_αy [0] (b value)
FOR each SV
        Read streamed data
        Convert it to float
        Store into array_αy [SV]
END FOR
FOR each feature
        Read streamed data
        Convert it to float
        Store into array_test [F]
END FOR
FOR each feature
        Clear array_ACC [F]
END FOR
FOR each SV
        FOR each feature of the SV
                array_ACC [F] += array_αy [SV] * array_SVs
[SV][F]
        END FOR
END FOR
FOR each feature
        Distance_value += array_ACC [F] * array_test [F]
END FOR
Distance_value -= b
IF (Distance_value ≥ th) THEN
        RETURN 1
ELSE
        RETURN -1
END IF
```

Figure 3-6. Proposed pseudo code of the SVM algorithm.

For optimizing the formal parameters of the top function as being ports of the hardware IP, the AXI4-Stream interface is assigned as the input interface by using the Vivado HLS directives in order to stream required data to the HLS IP in Zynq PL from the ARM processor in the Zynq PS part. Also, the AXI-lite bus is assigned for the designed function for controlling the IP and data flow for the system via communicating with the ARM processor.

In order to achieve a good level of optimization, some of the different available directives in the HLS tool for optimization are applied to the top module. Both pipelined and unrolling techniques are applied to the designed SVM HLS IP for optimizing loops and investigated based on synthesis results, aiming to find an optimized hardware solution (results are provided, compared and discussed in section 4.3.2).

Concerning arrays, some performance bottlenecks are often added with array accesses. The HLS tool normally maps arrays to dual-port memories in the RTL to improve throughput [115]. The existing resource allocation directive is utilized to target a RAM resource as a single/dual-port BRAM or LUT-based RAM (registers) for an array. Also, arrays could be partitioned or reshaped by the help of the tool in order to improve memory resource implementation. Array partitioning divides an array into smaller arrays of the same resource target, which allows higher parallel access and increases throughput. The available partitioning styles are block, cyclic and complete. The block style divides the array into equally sized blocks of consecutive elements, whereas the cyclic style divides into equally sized blocks interleaving the elements. The complete style breaks the original array into individual registers [115]. These available directives are also applied to the proposed IP for more optimization and design space exploration investigation (discussed in section 4.3.2).

3.4.2 Proposed Embedded System Design on Zynq SoC

The proposed HLS IP of the SVM classifier is successfully co-simulated (RTL simulation) and exported as an RTL implementation (packaged IP), after synthesizing the designed code utilizing the Vivado HLS tool. Next, the exported HLS IP is integrated into the proposed hardware/software system on Zynq SoC as shown in Figure 3-7. Using the Vivado design suite, the ARM processor is connected with the exported HLS IP in Zynq PL using a DMA controller IP via an ACP interface to stream data on AXI-Stream bus. Besides, an AXI interconnect IP is used to connect the ARM processor with all cores/IPs

in the PL part via the control AXI-lite bus to control connected IPs and data flow in the system.

Finally, the designed Zynq SoC is exported for the SDK tool after successfully passing the synthesis, implementation and bitstream generation stages in the Vivado design tool, in order to run an application on Zynq SoC.

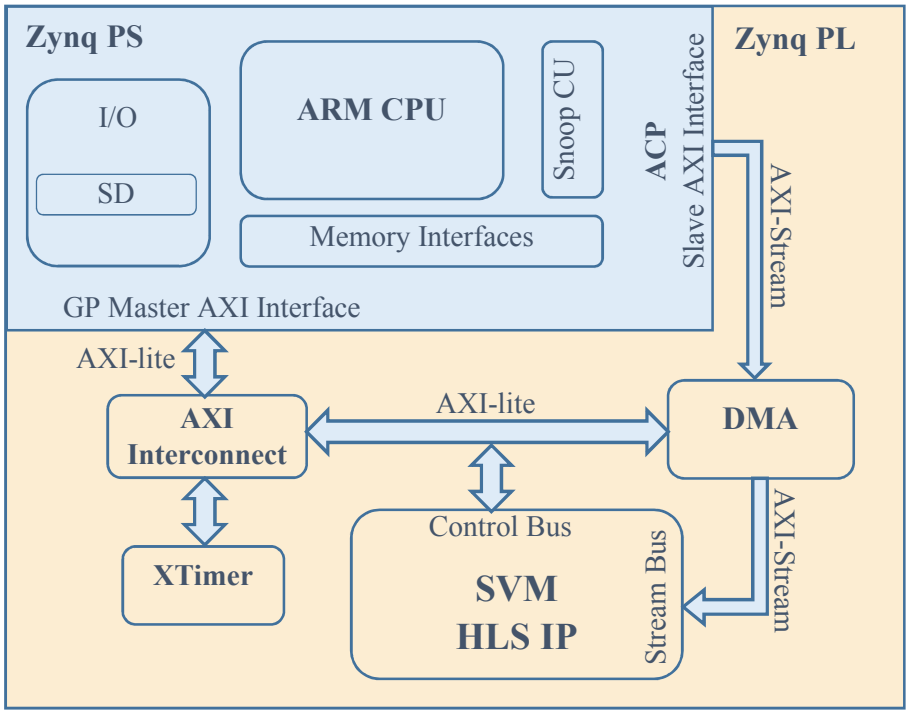

Figure 3-7. The proposed hardware/software co-design on Zynq SoC.

3.4.3 Proposed Software Design on Zynq PS

A test bench or a software program has been developed for testing and verifying the implemented classification system on Zynq, as well as realizing a full SVM IP running on the Zynq SoC. The ARM processor in the Zynq PS part is responsible for executing the test bench besides controlling the attached IPs/cores and the data flow in the system. A software program is designed and implemented in C using the SDK tool. Figure 3-8 shows the proposed algorithm of the software test bench/program that runs on the PS ARM processor.

The test instance and the parameters of the trained SVM model required for computations are read from text files saved on the SD card of the Zynq evaluation board. Three main files are read by the help of the LibXil Fat File System (FFS) library, which comprises a

file system and a glue layer to allow reading files from the SD card. The first file has the support vectors (features data) of the SVM model, and the second file includes αy for each *SV* with the *b* value. The third file keeps the test instance *x*. These three files are read Byte-by-Byte to be parsed and then saved into three arrays as the main input of the implemented SVM IP on the Zynq PL. Then, the data of the three arrays are streamed by the DMA IP through the AXI4-Stream interface to be processed by the implemented SVM IP in order to return the final classification result.

Define number of features+1 as *F* and SVs+1 as *SV*

1. Read *SVs* data Byte-by-Byte from *SVs_File* on the SD card
2. Parse data and store in *array_SVs [SV][F]*
3. Repeat steps 1 and 2 for *b* and αy data, αy*_File* and *array_*αy *[SV]*
4. Repeat steps 1 and 2 for *test* data, *test_File* and *array_test [F]*
5. Setup and calibrate the XTimer IP
6. Run the proposed SVM algorithm/function in Figure 3-6 on PS ARM
7. Measure the total execution cycles by XTimer IP for running SVM software on PS and getting the software_result
8. Initialize the DMA IP and flush the caches
9. Setup and initialize the hardware SVM HLS IP
10. Run the SVM IP on Zynq PL
11. Transfer *array_SVs [SV][F]* to the SVM IP via DMA
12. Loop and wait for DMA transfer to be done
13. Repeat steps 11 and 12 for *array_*αy *[SV]* and *array_test [F]*
14. Loop until SVM IP is done
15. Read return value hardware_result from SVM IP
16. Measure the total execution cycles by XTimer IP for running the hardware SVM IP and DMA transfers (steps 10-15)
17. Compare and print hardware_result and software_result
18. Compare run time for the hardware and software and print the acceleration factor

Figure 3-8. Proposed algorithm of the software program running on Zynq PS.

The same SVM algorithm as proposed in Figure 3-6 is executed on the ARM processor in order to compare its software result with the hardware result resulted from the implemented SVM HLS IP running on hardware. Also for comparing the performance, the XTimer IP is exploited to measure the clock cycles of running the SVM algorithm on software/PS and hardware/PL (including the hardware DMA streaming) and reports the hardware acceleration factor.

This proposed system could be easily adapted to any other trained SVM model with the same size of parameters (number of SVs and features) that targets similar applications or more general classification application. Specifically, all new data required could be easily loaded via the three designed files stored in the SD card to be processed by the implemented SVM on Zynq PL. Accordingly, the proposed SoC is feasible to achieve generality, flexibility, and adaptability.

3.5 Proposed Hardware Design of SVM HLS IP using BRAMs

In this section, a similar hardware design of the SVM HLS IP is proposed based on using BRAM interfaces instead of the stream interface used in the previous design proposed in section 3.4. This proposed HLS module has three main inputs as arrays and is divided into three functional blocks to compute the required calculations in the main decision function (3.1). The block diagram of the proposed IP is shown in Figure 3-9.

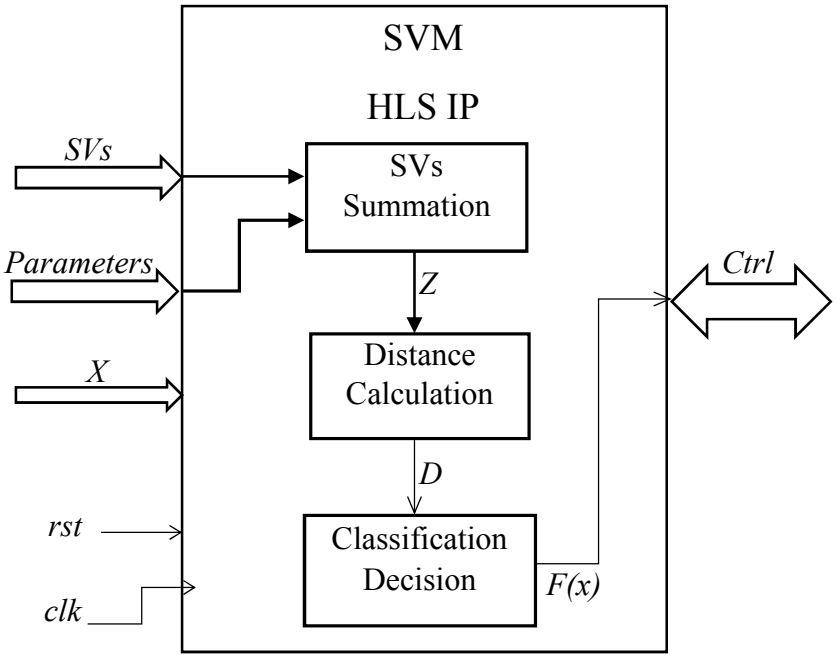

Figure 3-9. Proposed design of the SVM HLS IP.

Three input ports are designed for the arrays, which are called according to its contents as *SVs*, *Parameters*, and *X* arrays. The *SVs* is a 2D array that contains the features data of each SV. The *parameters* and *X* are 1D arrays that hold αy values of each *SV* (and the *b* value), and features of the test instance *x* respectively. Apparently, the proposed design depends on both the number of SVs and features size to implement any SVM model

(similar to the previous proposed design). To simplify the design, the main decision equation's calculations in (3.1) are divided into three main successive blocks. The first block "SVs Summation" is designed in a nested "for loop" (with the size of SVs number times features number) to compute the first part of the summation (accumulated array/vector *Z*) in the main equation as in equation (3.4).

$$\overrightarrow{Z} = \sum_{i=1}^{SV} \alpha_i \, y_i \, \overrightarrow{x_i} \tag{3.4}$$

The second block "Distance Calculation" performs the dot-product calculation between the accumulated array *Z* that outputted from the first block and the test array *X* as in (3.5) to produce the classification distance value *D*.

$$D = \overrightarrow{x} \cdot \overrightarrow{Z} \tag{3.5}$$

Finally, the parameter *b* (stored in the first place of the *parameters* array) is subtracted from the distance *D* to be then classified according to the proposed sign function in (3.6) at the final block "Classification Decision". The final classification output *F(x)* is returned through the control bus/interface *Ctrl*, which corresponds to a melanoma class (+1), benign/non-melanoma class (-1).

$$F(x) = sign\,(D - b) = \begin{cases} -1, & (D - b) < th \\ 1, & (D - b) \geq th \end{cases} \tag{3.6}$$

The HLS tool simplifies hardware design by assigning interfaces, resources, and other hardware techniques to the module through available various directives. The three input ports are each assigned as a BRAM interface (directive), as well, each input array is mapped to a single-port RAM. The control bus *Ctrl* is allocated to an AXI-lite bus, which controls the designed IP and data flow of the system through communicating with the ARM processor.

In addition, the pipelining technique is applied for each loop in order to improve the data throughput and latency. Accordingly based on the HLS synthesis results for an SVM model with 248 SVs and 27 features, the latency is reduced from 67,294 clock cycles to 8091 clock cycles, gaining a speedup of 8x from applying loop pipelining, while same number of DSPs (5) is utilized. As a disadvantage of gaining significant acceleration, extra FF and LUT resources are utilized, which reflects the well-known trade-off between speed and area. However by using the modern Zynq SoC of extensive resources, low figures of only 17% (18941) and 36% (19308) are utilized for FFs and LUTs respectively,

while gaining significant speed improvement. Consequently, the designed pipelined HLS IP is promising for achieving HPC, and low area, cost and power consumption requirements of embedded system realization for enhancing melanoma detection.

The designed HLS IP is successfully implemented and packed as an IP/RTL implementation by passing through the design flow of the HLS tool (C simulation, C synthesis, RTL co-simulation, RTL implementation). Then, the implemented HLS IP is integrated in the proposed system as depicted in Figure 3-10 that is designed by exploiting the Vivado design tool to be realized on Zynq SoC. The ARM processor in the Zynq PS part is connected to the HLS IP and other used IPs/cores in the Zynq PL via the control bus (AXI-lite) using an AXI-interconnect IP, in order to control connected IPs and data flow in the system. Three dual-port BRAMs are instantiated to be connected to the HLS IP input ports on one port each and to the ARM on the second port using AXI-BRAM-Controller IPs.

The designed SoC in Figure 3-10 is successfully implemented via the Vivado design tool stages (synthesis, implementation and bitstream generation), to be finally exported for running on Zynq by the help of the SDK tool. A similar test bench (C program) is developed for testing and verifying this implemented classifier and run it on Zynq SoC. The data required for the three arrays inputs of the HLS IP is read Byte-by-Byte from three main files to be parsed, and then written into the corresponding BRAMs. The files are stored in the SD card of the Zynq evaluation board, which hold required data for running the implemented SVM IP in order to classify a new test instance as melanoma or non-melanoma. Interestingly, this proposed system also shows scalability, flexibility, and adaptability, as any other trained SVM model can be easily implemented with the same size of the implemented model and run on Zynq by simply loading required data into the three main files.

3.6 Proposed Multi-core (Cascade SVM) Architecture

The proposed embedded system on Zynq SoC is a scalable IP-based design that is easily extended to form a multi-core architecture by adding more SVM IPs in a single device/SoC. The proposed multi-core architecture could be applied as a cascade classification. The cascade classification architecture contains multi classification stages (classifiers) that are designed in a cascading structure for accelerating the classification task (see Figure 2-5). In the next sub-section, a simplified hardware-friendly design is

proposed for the SVM HLS IP to be used to form a multi-core architecture on Zynq SoC. Next, the proposed multi-core architecture is presented with a case study of implementing a cascade SVM classification system on Zynq SoC.

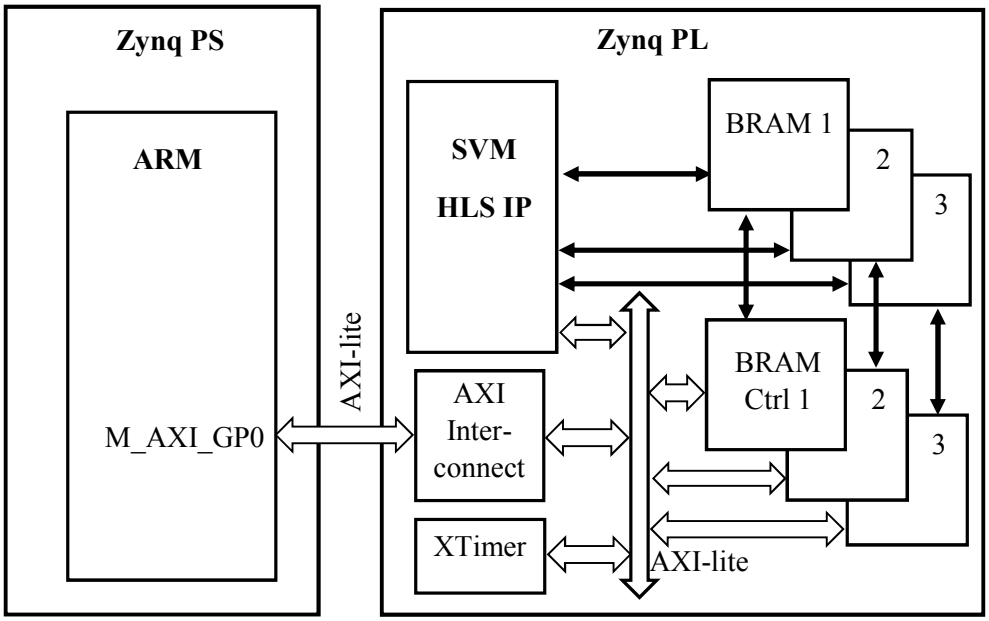

Figure 3-10. The proposed system on Zynq SoC.

3.6.1 Proposed Hardware-friendly Design of SVM HLS IP

The previous proposed hardware design of the SVM HLS IP (in section 3.5) is simplified to achieve a hardware-friendly IP to be used as a building block/core for the multi-core architecture, aiming to reduce hardware complexity and optimize implementation results. A new IP is proposed based on the designed IP in Figure 3-9, where the first block "SVs Summation" is pre-calculated (offline on software) to store the value of the accumulated vector Z (depends on trained SVM model data as in (3.4)) in the IP using an internal memory (similar to the initial design in Section 3.3 (equation 10 is implemented on software)). So, the new IP has a single input interface X (assigned to the control bus instead of a BRAM interface) and consists of the two blocks "Distance Calculation" and "Classification Decision", implementing equations (3.5) and (3.6) (see Figure 3-11).

The proposed pseudo code of the top module function for implementing the new hardware-friendly IP using the Vivado HLS tool is presented in Figure 3-12. This hardware design depends on the number of features to implement any SVM model. The designed HLS IP stores the accumulated Z array's data in the *AC_array*, which is

calculated offline using a proposed software function that implements equation (3.4) on software. The proposed IP takes a 1D array input *X* that consists of features of the test instance *x* to produce its classified class *F(x)*.

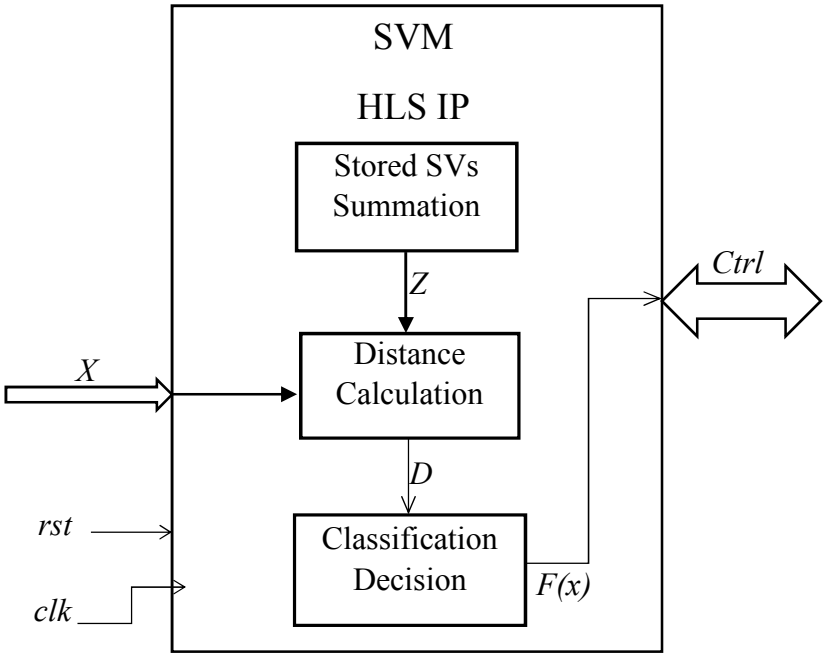

Figure 3-11. Proposed design of the SVM HLS IP.

```
Define number of features+1 as F
Function: Classify ( X [F] ) {
  Set value of "b"
  Initialize "AC_array [F] " with accumulated SVs data
  FOR each feature
        Distance_value += AC_array [F] * X [F]
  END FOR
  Distance_value - = b
  IF (Distance_value ≥ 0) THEN
      RETURN 1
  ELSE
      RETURN -1
  END IF
}
```

Figure 3-12. The pseudo code of the proposed hardware-friendly SVM HLS IP.

Using the HLS tool's directives, the AXI-lite bus is assigned for the control bus *Ctrl* of the module for controlling the designed IP core and other connected cores in the system including data flow management. For optimization, the pipelining technique is applied for the "for loop" using the HLS directives for enhancing data latency and throughput. By using the pipelining, the HLS synthesis results show latency reduction from 278 to 148 clock cycles for implementing an SVM model with 27 features. Accordingly, the speedup is nearly doubled from using the pipelining, but with some extra logic resources utilized. This founding indicates the trade-off exists between the performance and area/cost.

Similarly, after successfully synthesizing and implementing the designed code in the Vivado HLS tool, the HLS IP is exported and integrated into the proposed hardware/software system as shown in Figure 3-13. Using the Vivado design suite, the exported HLS IP is connected to the ARM processor via the AXI-lite interface passing by an AXI Interconnect IP. Finally, the implemented Zynq SoC is exported for the SDK tool to test the implemented HLS IP using an adapted software C program based on that previously proposed.

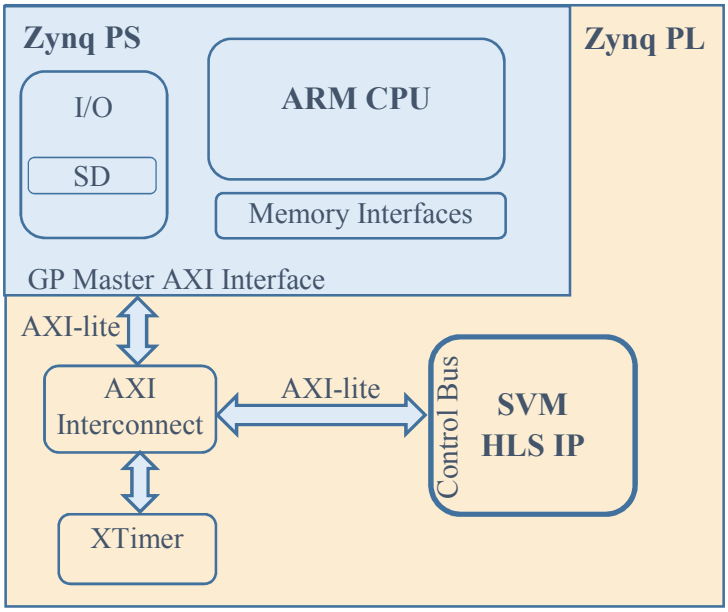

Figure 3-13. The proposed hardware/software system on Zynq SoC.

3.6.2 Proposed Multi-Core Architecture

After implementing the proposed hardware-friendly design of the monolithic SVM HLS IP, it is integrated in the proposed embedded system on Zynq SoC as in Figure 3-14, where more IPs (1 to n) can be added for design extension and scalability. By using the proposed HLS-based design, various copies of SVM IPs with different parameters/sizes could be implemented to construct a multi-core architecture that consists of n cores/IPs in a single system on chip/device. The proposed scalable multi-core architecture could be applied as an ensemble, multi-class, or cascade classifier for different classification problems and applications.

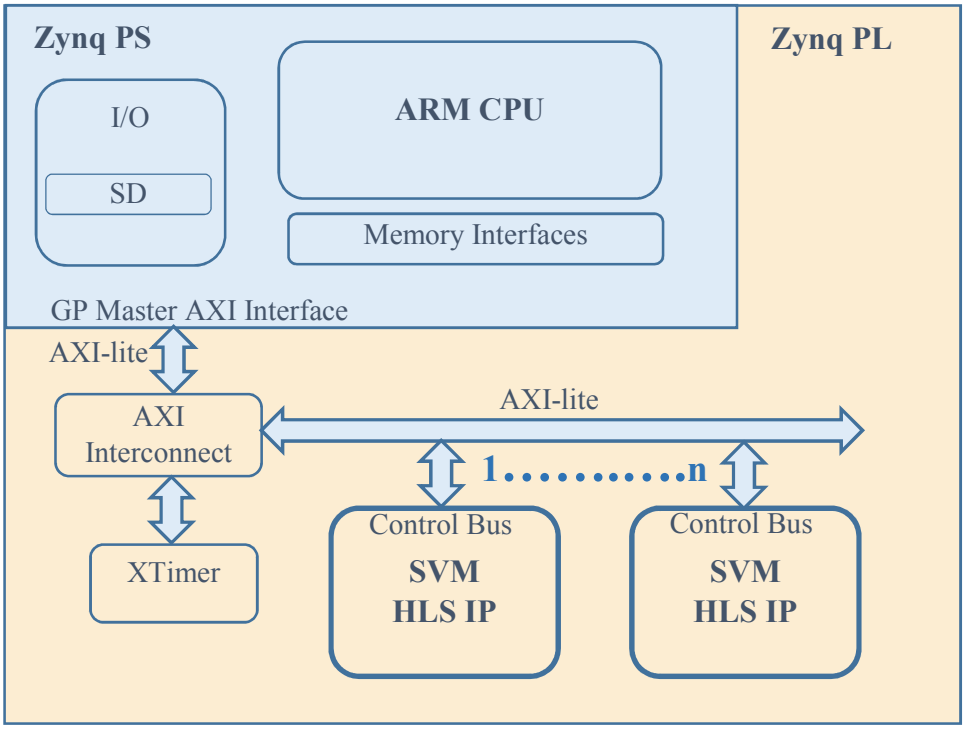

Figure 3-14. The proposed hardware/software system on Zynq SoC.

In this research, we are proposing an SVM-based cascade classifier for improving early detection of melanoma by exploiting the proposed multi-core architecture. A dual-core architecture (n=2) is proposed as our case study to be employed as a 2-stage cascaded SVM classification architecture. Figure 3-15 shows the proposed 2-stage cascade classifier that consists of two SVM HLS IPs. For enhancing early melanoma detection, the first stage in the proposed cascade is a melanoma-sensitive SVM model that is trained with a dataset of more melanoma samples, whilst the second stage is a benign-sensitive (non-melanoma-sensitive) SVM. The scenario of this cascade architecture is proposed to

verify the benign samples classification for this sensitive application, aiming to decrease the risk of false detection that harm patients by reducing the number of false negative. The test instance X is passed first by the early stage to detect melanoma as a positive sample, otherwise, it is passed to the second stage for a second verification. However, the melanoma results from the first stage should be verified by a skin cancer specialist for further diagnosis and medication.

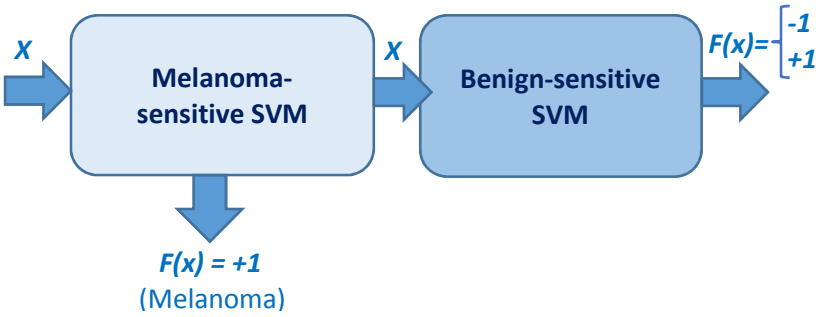

Figure 3-15. The proposed 2-stage cascade SVM classifier.

Using the HLS tool, the two SVM models are implemented to be then exported and integrated in the proposed system in Figure 3-14. Using the Vivado tool, the two exported HLS IPs are connected to the ARM processor through the AXI-lite bus to implement the 2-stage cascade SVM classifier. An application is also developed in the SDK tool (similar to the previously proposed test bench) to validate the implemented 2-stage cascade classifier on Zynq SoC. Different scenarios could be easily applied to the proposed cascaded architecture handling different classification problems and requirements. Also, the proposed 2-stage cascade classifier would be extended in the future to a multi-stage cascade classifier by simply adding more IPs, targeting performance improvement of classification.

3.7 Proposed Dynamic Cascaded SVM Architecture

The FPGA-based DPR technique is exploited to achieve a dynamic hardware system for realizing an adaptive SVM cascade classifier on Zynq SoC, targeting more flexibility and scalability, in addition to more improvements in area, power and classification performance. In order to apply the DPR feature on the proposed 2-stage cascaded architecture, a static top-level design with a single Reconfigurable Partition (RP) module/block is first proposed (RP defines the physical area on the FPGA designed for

PR, while the rest of the design remains as static logic (see Figure 2-2)). The proposed system in Figure 3-14 with one SVM HLS IP/core (n=1) is used to define the static design wrapper, while the SVM HLS IP is assigned as a RP. Then, a Black-Box (BB) is mapped onto the RP, which is defined by its hardware interfaces and ports. The BB module is then used to be reconfigured dynamically (swapped in and out) at run-time to one of the two implemented SVM IPs, melanoma- and benign-sensitive IPs.

Each IP is defined as a Reconfigurable Module (RM) for the RP that allocates the logic size based on the larger resource utilization. Each RM is to be plugged into the defined BB, where two different configurations are defined that combines the static design wrapper with each RM (a configuration designates a complete FPGA design and generates a full bitstream for the RM combined with the static logic and a partial bitstream for the RM itself). In our proposed design, the first configuration is for instantiating the first stage/IP, while the second configuration instantiates the second stage in the cascade.

In order to realize the proposed dynamic system, the design flow summarized in Figure 3-16 [117] is implemented using the Vivado design suite (IP Integrator (IPI) and Floorplanning), which starts with the HLS IP implementation and ends with generating bitstreams. It includes the top-level design synthesis and Out-Of-Context (OOC) synthesis of each RM (saved as Design Checkpoints (DCPs)), followed by implementing each defined configuration per RM and PR verification. Each of the three configurations (static with BB/RM) are designed for implementation with the aid of the Vivado Floorplanning tool, which is used to design a floorplan of the RP in the device with associated resources.

Figure 3-17 shows an image that composes of the Vivado device view for the static logic of the top-level design with BB module (left) and the two RMs, RM-M for melanoma-sensitive SVM (middle) and RM-N for non-melanoma-sensitive SVM (right) (a case study of DPR architecture (Figure 2-2)). The static resources are coloured in orange and the pblock is a purple box in the bottom-right corner that represents the RP (BB in static view), where no resources are placed as it is reserved for the RMs. The grey boxes are the ports of the RMs interfacing with the static logic. From the RMs' views, different resources are utilized for each RM that are demonstrated in cyan colour (see Appendix A for full view of the device floorplanning and implementation of the three configurations).

Finally, two bistreams are generated for each configuration, full and partial bitstreams to form a library of configuration files for configuring the device/FPGA. The full bitstream should be used for the first configuration of the FPGA to configure the whole device. Then, the partial bitstream is used to dynamically reconfigure the RP only to the corresponding RM, which has much shorter configuration time than the full bitstream.

For testing the implemented dynamic cascade classifier on Zynq SoC, the developed application in the SDK tool is modified to include dynamically reconfiguration using the generated partial bitstreams. Two configuration modes can be used for partial reconfiguration either by using JTAG or PCAP (Processor Configuration Access Port) interface. The JTAG interface is used in our experiments (as a case study), while the PCAP is to be used in the future to allow reconfiguration through the application running on the ARM processor, aiming for more time improvement.

The proposed DPR-based dynamic hardware system could be easily extended in the future to include more RMs reconfiguration for realizing an adaptive multi-stage cascade classifier, while keeping same design space. Therefore, our proposed dynamic hardware system promises to meet challenging embedded system constraints for realizing a low-cost handheld device for early detection of melanoma.

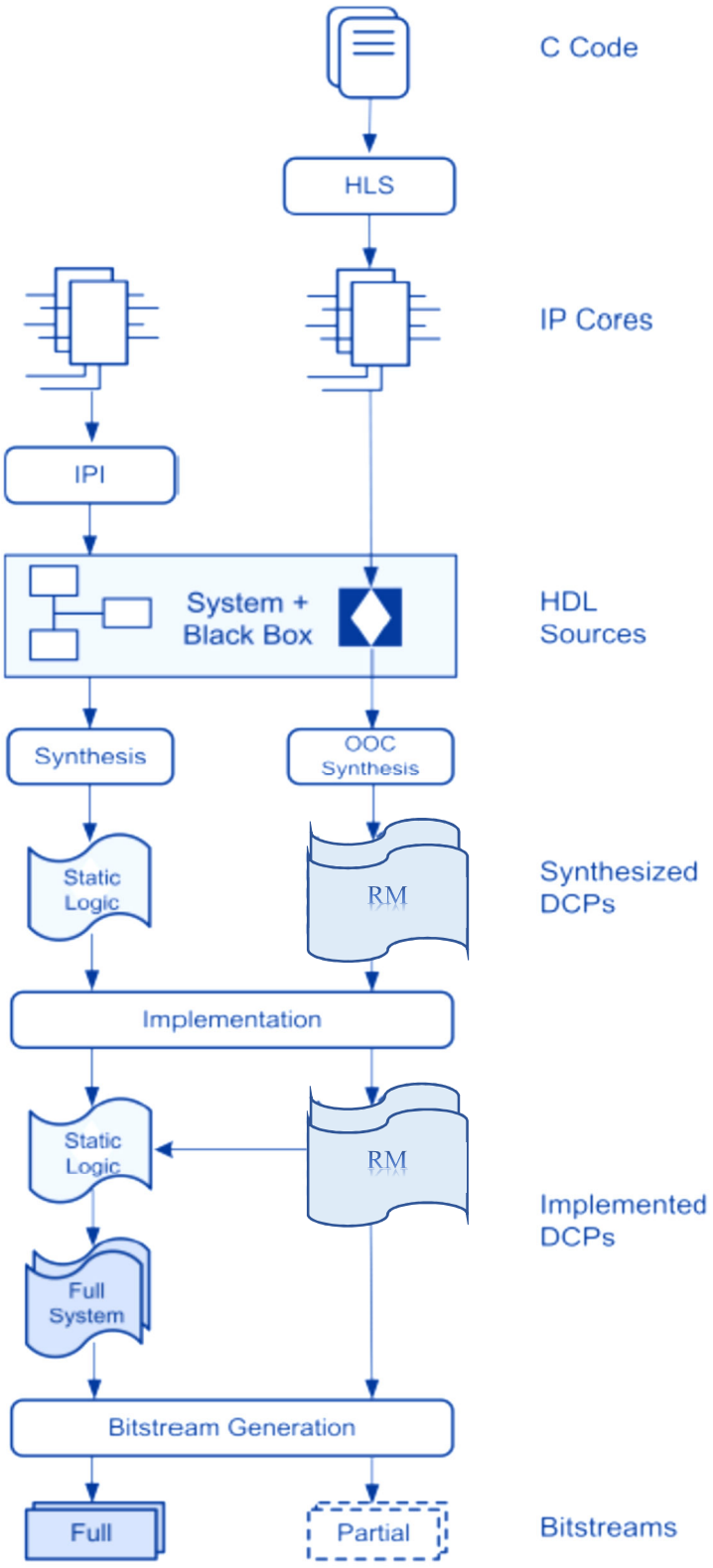

Figure 3-16. Full system design flow for PR design [117].

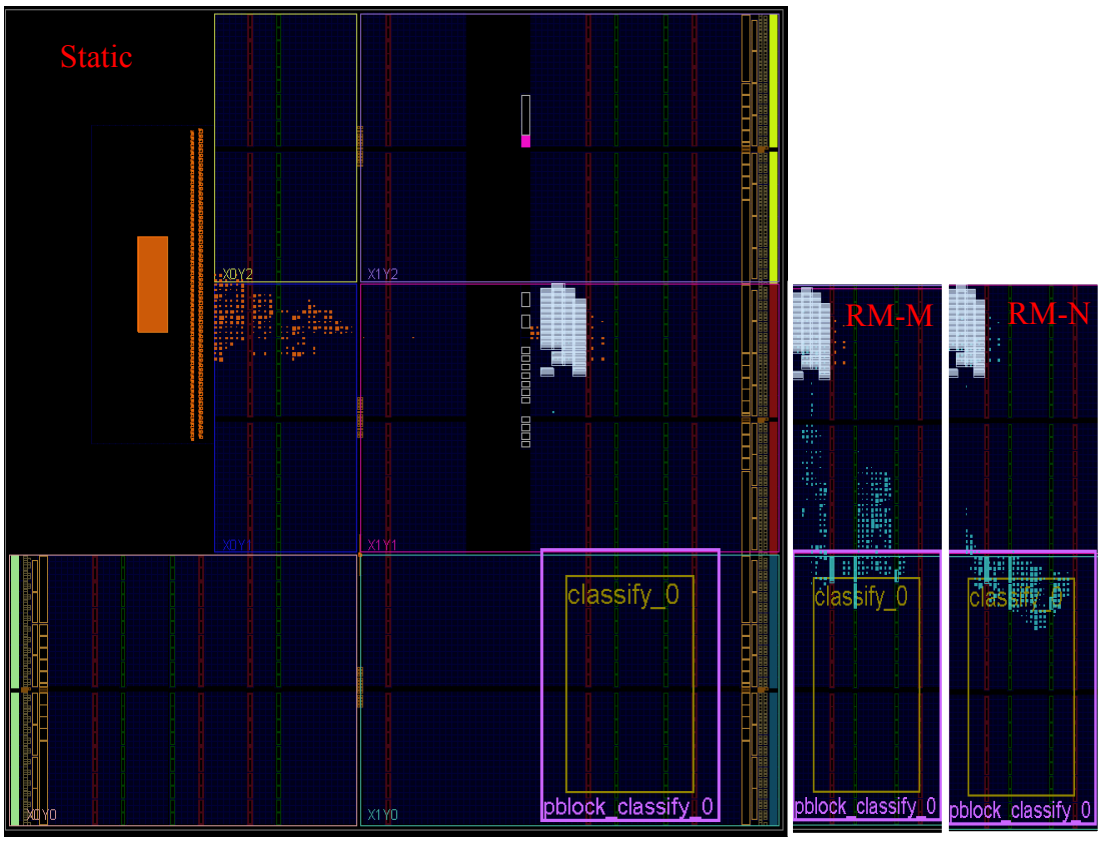

Figure 3-17. Vivado device view of the three configurations; the left side view is for the first configuration that represents the static logic of the top-level design with BB module. The two views on the right are for the two RMs: RM-M (for melanoma-sensitive SVM) and RM-N (for non-melanoma-sensitive SVM) respectively. The static resources are displayed in orange colour at the up-left corner of the static view. The pblock "pblock_classify_0" is represented in a purple box at the bottom-right corner, which defines the RP as a BB in the static configuration (to be reconfigured for each RM of the two configurations). The grey boxes represent the ports of the RMs interfacing with the static logic. In each RM's view, the corresponding resource utilization is illustrated in cyan colour.

3.8 Conclusion

This chapter presented our proposed hardware designs, aiming to implement an optimized SVM classifier for melanoma detection on the hybrid Zynq SoC using the modern HLS design methodology. First, a hardware/software co-design was proposed to realize a Zynq accelerator for the SVM classifier. Then, the hardware design was extended to achieve a full SVM IP running on hardware/FPGA. Next, a similar hardware design was proposed

based on using BRAM interface for data transfer, which replaced the stream interface in the previous design. Furthermore, a scalable multi-core architecture was proposed based on adding more SVM IPs on a single device to implement a cascade SVM classifier on Zynq SoC. A simplified design of the HLS SVM IP was proposed as a hardware-friendly design and selected to act as a building block for the multi-core architecture, in order to reduce hardware complexity and achieve optimized results. A dual-core architecture was proposed as our case study for implementing a 2-stage cascade SVM classifier for enhancing online melanoma detection on Zynq SoC. Finally, a novel dynamic hardware system that exploits the powerful DPR technology of the FPGA was proposed for realizing an adaptive, flexible and scalable embedded system of a cascade SVM classifier for efficient melanoma detection.

CHAPTER 4 Experimental Results and Discussion

4.1 Introduction

A common SVM classifier called "SVM-Light" has been studied as a case study to implement our SVM IPs for melanoma detection. The SVM-light is a robust and simple classifier, which is available in C implementation and has been used in various classification problems [118]. The modern UltraFast HLS design methodology was utilized to design and implement the SVM IPs on the Zynq SoC, which were based on the binary classification algorithm and C/C++ code of the SVM-Light. The binary classification code with linear kernel has been studied and adopted for customization in order to reach a synthesizable code capable of realization on hardware. In the designed C code, float data type was assigned for all used data and was mapped to the standard single-precision floating-point format in the hardware by the help of the Vivado HLS tool. For synthesizing, the dynamic memory allocation scheme applied in the C code for flexibility was changed for static fixed sizes targeting our case studies. As a result, the algorithm became dependent on some parameters of the SVM model (the number of features and the number of SVs), which should be defined prior the hardware implementation.

The training phase has been performed offline on software by exploiting the available SVM-Light windows application, where the default parameters and the linear kernel function were used to generate the trained SVM models. Based on our previous work within our research group for melanoma detection [7], a dataset was used for model training that consists of a total of 356 clinical images collected from available web resources, including 168 melanoma and 188 benign images. The extracted features from the image dataset were used in the training phase to generate a trained model for melanoma detection. Each image of the dataset includes a colour image of one mole with 6 mm diameter or greater. Some selected pre-processing algorithms were applied first to the lesion images for hair artifact removal and then they were manually cropped and resized to form unified images of 512x512 pixels. Next, a lesion segmentation or border detection algorithm (interactive object recognition) was employed for background removal. Finally, feature extraction schemes based on HSV colour channels were applied to the images, which generated a feature extracted dataset of 356 instances of 27 features each that was used for training the SVM models [7]. In order to achieve a higher accuracy

rate for the trained models, the cross-validation technique was utilized in the training phase. Finally after generating a trained SVM model offline, the model data was extracted to implement the trained SVM model on hardware using any of the proposed design.

The modern Xilinx Vivado 2016.1 Design Suite was utilized to design and implement our proposed system on the Xilinx XC7Z020CLG484-1 target device of the Zynq-7 ZC702 Evaluation Board. The Vivado HLS tool was used first to design and implement an SVM HLS IP. Then, the developed SVM IP was exported to be integrated into the proposed Zynq SoC (as shown in Figure 3-4) that was designed in a block diagram using the Vivado tool (the block designs are shown below for each of the corresponding proposed system). The designed Zynq system was synthesized, placed and routed, and finally the bitstream was generated to be exported for the Xilinx SDK tool to run an online classification application on Zynq.

Using the Xilinx SDK tool for each of the proposed system, an application of classifying a test instance was developed in C that implements the proposed test bench presented in the previous chapter. Some test instances of extracted features were correctly classified by all implemented SVM IPs (one at a time). In addition, the experimental results demonstrated that every hardware classification result of the implemented classifiers was exactly equal to the corresponding software classification result (to be shown and discussed below). Accordingly, the classification accuracy level could be preserved without any loss from the hardware implementation, in contrast to some existing implementations in the literature as stated in chapter 2 (Section 2.5.6). More instances would be tested in the future in order to validate the classification accuracy rate of the hardware implemented classifiers.

In the next sections, for each of the proposed designs in Chapter 3, the implemented SVM models are introduced and then experimental results and comparisons are presented and discussed.

4.2 Proposed Hardware/Software Co-design

4.2.1 Implemented SVM Model

Using the proposed hardware/software co-design described in Section 3.3, a trained SVM model was implemented for melanoma detection that was generated with 27 features and 346 SVs from the training process in SVM-light windows application. The dataset of 356

instances of 27 features was used for the training process. The Vivado block design view is shown in Figure 4-1 for implementing the proposed hardware/software system in Figure 3-4. By the aid of the Xilinx SDK tool, the proposed software program (test bench) was developed and some test instances were tested. The experimental results verified that the calculated distance value for the hardware IP ("*Distance_value"* in Figure 3-5) was the same as the software result, which shows that the classification accuracy rate was preserved with zero loss (in contrast to some previous works in the literature [56, 67, 91, 93, 96-98]).

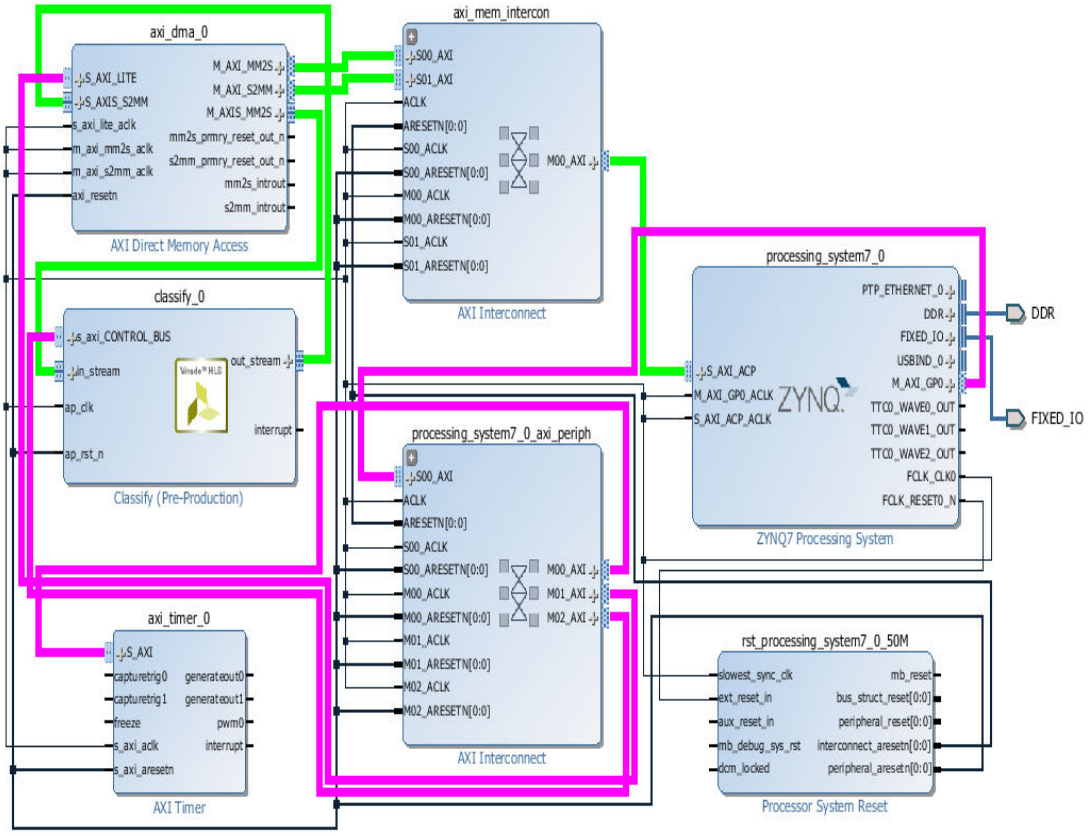

Figure 4-1. The Vivado block design view of the designed embedded system implemented on Zynq SoC, where the implemented HLS SVM IP "Classify" is connected to the Zynq7 Processing System and other IPs using AXI interface.

4.2.2 Processing Speed and Time

The AXI Timer IP core was used to compare the total computing time between running the designed code for the accumulation of multiplication process on ARM processor in

PS and on hardware IP in PL part. The total number of clock cycles for the software on ARM processor was 64,998, while for both the AXI DMA transfer and the hardware accelerator was 549 cycles. Accordingly, the acceleration factor was equal to 118.34x, which shows significant acceleration from the implemented hardware accelerator for the proposed SVM classification system. For this experiment, the ARM processor is running at 666.67 MHz, whilst the computing logic is running at only 50 MHz. However, the processing time for the hardware IP including streaming data (10.98 µs) is greatly boosted by 8.88x from the software (97.5 µs).

4.2.3 Hardware Implementation Results

The hardware resource utilization for the implementation of the proposed system on the Zynq SoC is summarized in Table 4-1. It is clear that the percentages of the resource utilization are very low, showing significant improvement in area savings. Moreover, the power consumption results reported by Vivado tool are detailed in Table 4-2. The on-chip total power consumption of 1.738 watts is considered to be small reasonable value, meeting embedded system constraints. The device static power consumed 9% of the total, whilst the remainder 91% was dissipated by the dynamic activity, mostly from the Zynq PS component, compared to other on-chip components.

4.2.4 Comparison with Related Works

Interestingly, the proposed hardware/software co-design of SVM classification of melanoma instances is to be considered as an added work in the literature that was realized on the recent hybrid Zynq SoC using the HLS design methodology. In addition, our implemented system achieved lower hardware resource utilization than some of the previous FPGA-based SVM classification implementations of different applications in the literature [54, 67, 69, 89, 93, 105]. Concerning power consumption, our implementation demonstrated lower power dissipation than other related works [21, 74, 93]. Therefore, our initial proposed system could be deployed as a real-time embedded system dedicated for melanoma detection.

Table 4-1. Device utilization summary for the implemented system on Xilinx Zynq-7 ZC702 SoC

Resource	Utilization	Available	Utilization %
Slice FF Registers	5584	106400	5.25
Slice LUTs	4373	53200	8.22
Memory LUT	173	17400	0.99
BRAM	3	140	2.14
DSP48	5	220	2.27
BUFG	1	32	3.13

Table 4-2. On-chip components power consumption summary for the implemented system on Xilinx Zynq-7 ZC702 SoC (in Watts)

On-Chip Component	Power (W)
Clocks	0.009
Logic	0.003
Signals	0.004
BRAM	0.002
DSPs	<0.001
PS7	1.565
Total Dynamic Power	1.584
Device Static Power	1.54
Total On-Chip Power	1.738

4.3 Proposed Hardware Design of SVM HLS IP

4.3.1 Implemented SVM Models

Three SVM trained models have been developed offline (using the SVM-Light windows application), to be used for the hardware implementation using the proposed design (Section 3.4), targeting melanoma detection. Three models have been generated from training the available feature dataset for melanoma with 356 instances of 27 features each.

First, the original full dataset was used to generate a trained SVM model with 346 SVs. Then, data scaling and normalization techniques were applied to the original dataset, where another trained model with 248 SVs was generated that achieved higher classification accuracy. Apart from the classification performance, these two trained models with different SVs number/size have been implemented using the proposed design. The main purpose of implementing both models was to validate the proposed design for implementing different scales/sizes, while investigating hardware experimental/implementation results, aiming to find an optimized hardware solution that meets critical embedded system constraints (scalability, flexibility, real-time, and low cost, area and power consumption).

In order to run the proposed software program/application (Figure 3-8) completely on Zynq SoC to test and validate the implemented SVM models (HLS IPs), a bigger external memory is needed at run-time to process big data required by these large-scale models. Accordingly, another third model with smaller scale has been implemented in order to be used as a case study for performance validation through running on the Zynq SoC, while the other two models were validated using simulation results only (due to the limited size of the available run-time memory). The small-scale model has 61 SVs generated from using part of the available normalized dataset in the training phase (100 melanoma and 44 benign instances) and achieved the highest accuracy rate compared to the two large-scale models for melanoma classification.

In addition, another dataset of different application ‘pattern recognition’ has been used and tested in order to validate the proposed design for general purpose classification. The used dataset selected for this case study is a standard example on the SVM-Light website [118]. The trained SVM model has 877 SVs and 9947 features, which has larger size than the trained SVM model for our application/case study “melanoma detection”. This large-scale model was fully implemented on the Zynq SoC using the previously proposed hardware/software co-design in Section 4.2 and was successfully tested and validated with some test instances for the targeted application. However for implementing such a large model using this proposed extended hardware design, an FPGA of bigger capacity with more resources is required or an additional device(s) could be co-operated to be fully implemented on hardware. Accordingly, the proposed design is capable of implementing SVM classifiers for various classification applications with variable sizes. Therefore,

generality, scalability, and applicability of classification could be achieved with the proposed design.

Finally, for our case study "melanoma detection", the three SVM trained models with different sizes (SVs' number) were implemented using the proposed design described in Section 3.4, and validated and evaluated in the following sub-sections. For convenience, the moderate-scale model with 248 SVs is denoted as "Model 1" in the rest of the thesis, while the large-scale model with 346 is denoted as "Model 2" and the small-scale model with 61 SVs is called "Model S".

Table 4-3 summarizes different parameters of the three implemented SVM models. For each model, the table shows the number of SVs generated with the number of instances used in the training dataset of 27 features each (model 1 and S were trained using normalized data).

Table 4-3. Parameters of implemented SVM Models

SVM Model	Training Dataset		SVs
	Melanoma instances	Benign instances	
Model 1	168	188	248
Model 2	168	188	346
Model S	100	44	61

The table shows the number of SVs generated with the number of instances used in the training dataset of 27 features each.

In the next sub-section, different HLS optimization techniques/directives as applied to the three models are illustrated, aiming to find the best effective designs based on synthesis results. Then in the following sub-section, the implementations of the selected HLS designs are shown for the three models, compared and evaluated based on the hardware implementation results (hardware resource utilization, power consumption, processing speed and time and classification accuracy), aiming to reach an optimized hardware solution.

4.3.2 HLS Synthesis Results

Some available optimization directives of the Vivado HLS tool have been employed and tested (as introduced in Section 3.4.1) for applying design space exploration and

optimizing the proposed HLS SVM IP design. Different experiments have been performed to investigate improving and optimizing the synthesis results by applying various optimization directives of the HLS tool, aiming to achieve an efficient and hardware-friendly design with low hardware complexity. The HLS synthesis results with the assigned directives are presented in Table 4-4 and Table 4-5 for the two large-scale models, model 1 and model 2 with 27 features each, but different SVs numbers of 248 and 346 respectively.

For the two tables, the first column displays the used directive, whilst the next columns present the synthesis results for each corresponding directive (the design latency (clock cycles), throughput/II (clock cycles), and resource utilization). The first row in the tables demonstrates the synthesis results for the default settings of the HLS tool in addition to applying the interface directives of the I/O ports mapping (AXI4 and AXI-lite as explained in Section 3.4.1). The subsequent rows are for applying alternative directives besides the interface. The resource allocation has been tested for the used arrays using BRAMs and LUTs. The pipeline or unroll technique was used for inner/most loops. It is recommended to pipeline the inner loops only in the nested loops, aiming to reach the optimum solution and allowing the HLS tool to make required scheduling quickly [115]. Additionally, the different array partition styles have been applied with the loop unrolling directive under the same factor in addition to the pipeline.

By applying different directives, the latency and throughput were significantly improved, but at the expense of using more resources. This is a justification of the existing trade-off exists between data throughput and area. The latency of the two models of 82,460 and 114,898 clock cycles were significantly decreased by a factor greater than 9 and 8 respectively. It is clear that with unrolling loops, lower latency of 9,876 and 13,698 cycles of the two models were achieved, but a higher number of resources was utilized, especially the expensive power-consuming DSPs, which were increased from 5 to 135 for both models. Also by unrolling more loops, the best latency of 8,366 cycles was successfully achieved for model 1 with utilizing more resources and using almost all available LUTs of 93%. Similarly for model 2, the least latency of 11,600 cycles was achieved, however, it is not applicable for implementation due to the excess utilization of available LUTs of 119%. However, by applying the array partitioning (cyclic style on a factor of 16), the best latency of 12,960 cycles was achieved for model 2 with fewer

resources (20 DSPs), while a good latency of 9,336 cycles was achieved for model 1 that was not less than unrolling most loops.

Table 4-4. Synthesis results of applying different directives of the HLS tool to the proposed HLS SVM IP of Model 1 (248 SVs)

Directives	**Latency (clock cycles)**	**Throughput (Initiation Interval) (clock cycles)**	**Resource Utilization**			
			BRAM	**DSP**	**FF**	**LUT**
Interfaces	82460	82461	17	5	1265	2417
Array Resource: BRAM	82460	82461	19	5	1137	2376
Array Resource: LUT	82460	82461	0	5	1393	6015
Pipeline inner loops	**14138**	**14139**	**19**	**5**	**1251**	**2477**
Pipeline most loops	14129	14130	19	10	2622	3999
Pipeline all loops	14129	14130	30	58	5460	9516
Unroll inner loops	9876	9877	28	135	13080	28039
Unroll most loops	**8366**	**8367**	**29**	**135**	**13226**	**49625**
Partial Unroll (factor 2)	78959	78960	19	5	1266	2645
Array partition (cyclic factor 2)	13858	13859	22	10	2105	3454
Array partition (cyclic factor 8)	13827	13828	17	10	6925	10103
Array partition (cyclic factor 16)	**9336**	**9337**	**16**	**20**	**11113**	**12835**
Array partition (block factor 2)	42217	42218	22	7	10916	12821
Array partition (complete)	23512	23513	27	5	23056	10218

Table 4-5. Synthesis results of applying different directives of the HLS tool to the proposed HLS SVM IP of Model 2 (346 SVs)

Directives	Latency (clock cycles)	Throughput (Initiation Interval) (clock cycles)	Resource Utilization			
			BRAM	DSP	FF	LUT
Interfaces	114898	114899	33	5	1271	2429
Pipeline inner loops	**19626**	**19627**	**35**	**5**	**1258**	**2465**
Pipeline most loops	19617	19618	35	10	2629	4048
Pipeline all loops	19617	19618	30	58	5467	9550
Unroll inner loops	**13698**	**13699**	**28**	**135**	**13919**	**29975**
Unroll most loops	11600	11601	29	135	13793	63328
Array partition (block factor 2)	59125	59126	38	7	11004	12921
Array partition (cyclic factor 2)	19248	19249	38	10	2117	3503
Array partition (cyclic factor 8)	19215	19216	40	10	6490	10027
Array partition (cyclic factor 16)	**12960**	**12961**	**28**	**20**	**11123**	**12938**
Array partition (complete)	32724	32725	27	5	29334	11276
Interfaces	114898	114899	33	5	1271	2429
Pipeline inner loops	19626	19627	35	5	1258	2465
Pipeline most loops	19617	19618	35	10	2629	4048

Regarding the pipeline technique, the least possible resource utilization with only 5 DSPs was achieved for both models with an improved latency of 14,138 and 19,617 cycles. By pipelining more loops, the latency was slightly decreased for both models with some increases in the resource utilization, reaching 10 or 58 DSPs. Accordingly, the pipelined design is considered to be more promising for a low-cost solution that offers low area and power consumption with reduced latency and high throughput.

Similarly for model S, the synthesis results are summarized in Table 4-6 for applying only the most effective directives (pipelined, unrolled and array partitioning). Optimized results were achieved for latency and resource utilization, especially with applying the pipelining and unrolling techniques.

Table 4-6. Synthesis results of applying different directives of the HLS tool to the proposed HLS SVM IP of Model S (61 SVs)

Directives	**Latency (clock cycles)**	**Throughput (Initiation Interval) (clock cycles)**	**Resource Utilization**			
			BRAM	**DSP**	**FF**	**LUT**
Interfaces	40885	40886	5	5	1656	2548
Pipeline inner loops	**3830**	**3831**	**7**	**5**	**1666**	**2511**
Pipeline all loops	3822	3823	30	105	13854	16195
Unroll inner loops	3343	3344	29	135	20626	26393
Unroll most loops	**2653**	**2654**	**27**	**135**	**19429**	**45233**
Array partition (cyclic factor 16)	4483	4484	27	19	15447	16498

Finally, design space exploration was investigated and different solutions were achieved by applying various optimization techniques, aiming to find an optimum solution for the existing trade-off between performance and area/cost. So, various designs have been implemented with different options including both a low-cost design and a faster design with higher cost, which could be selected by designers based on their project requirements. For our case study "melanoma detection", the lower cost design with fewer resources and lower power consumption is preferred, aiming to reach an efficient embedded system to be deployed as a fast and cost-effective handheld device.

4.3.3 Hardware Implementation Results

After developing the designed HLS SVM IP, it was integrated into the proposed Zynq SoC for further implementation using the Vivado tool. The Vivado block design for implementing the proposed hardware/software system in Figure 3-7 is shown in Figure 4-2. Based on the HLS synthesis results presented in the previous section, the most effective designs of the HLS IP that showed better synthesis results were selected for the Zynq SoC implementation. For the two large-scale models, the pipelined, unrolled and

array partitioned designs (bolded in Table 4-4 and Table 4-5, to be denoted as design 1, 2, and 3 respectively) have been implemented (using 100 MHz and 666.67 MHz operating frequency for PL/FPGA and PS ARM CPU respectively), for being the three best solutions for a balanced trade-off between achieving high-performance and low-area/cost. Also, model S has been implemented (using 250 MHz frequency for both FPGA and PS ARM CPU) with the pipelined and unrolled designs (bolded in Table 4-6).

4.3.3.1 Hardware Resource Utilization

The FPGA resource utilization for the three implemented systems on Zynq SoC is summarized in Table 4-7 and Table 4-8 for model 1 and 2 respectively. Also, Table 4-9 summarizes the two implemented designs of model S. The first column shows the hardware resource and then a pair of columns for each of the three designs illustrates the resource utilization value and its percentage value. The last column indicates the number of available hardware resources in the target device. It is clear that the percentages of all resource utilization in the three tables for all design implementations are very low, showing significant improvement in area savings. Regarding the large-scale models, the number of DSPs utilization is equal for the two models that is increased from 5 in design 1 to 20 and 135 for designs 3 and 2 respectively, showing the highest percentage of 61.36% for design 2 compared to other resources (similar behaviour for model S). Accordingly, design 2 is considered to be the most costly design compared to the other two with the largest area results for almost all resources. Only memory LUTs were utilized more by design 3 than the other two designs, while other resources were utilized in a moderate level between the other two designs. Design 1 demonstrated the least area results, where the max utilization is 8.57% and 14.29% of BRAMs for model 1 and 2 respectively (5.39% of LUTs for model S). Finally, design 1 is considered to be the most cost effective design for achieving an embedded system for online classification with low area and low cost.

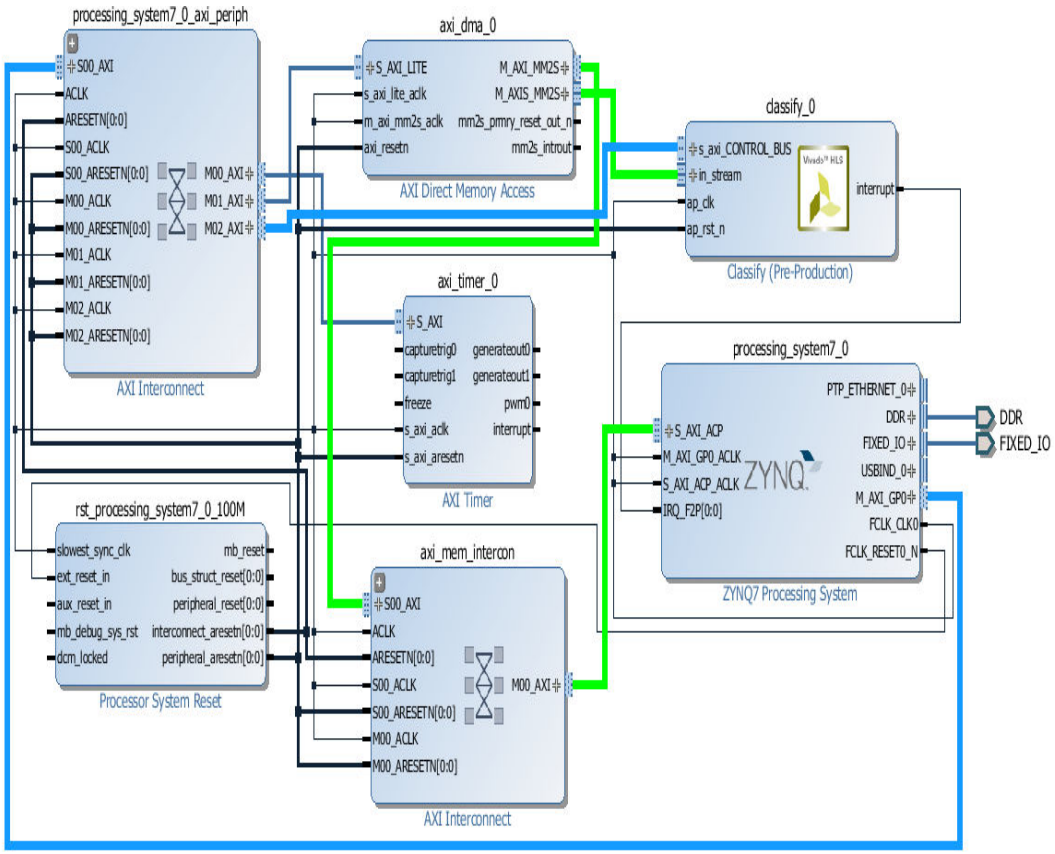

Figure 4-2. The Vivado block design view of the designed embedded system implemented on Zynq SoC, where the implemented HLS SVM IP "Classify" is connected to the Zynq7 Processing System and other IPs using AXI interface.

Table 4-7. Device utilization summary for the three implemented designs of Model 1 on Xilinx Zynq-7 ZC702 SoC

Resource	Design 1		Design 2		Design 3		Available
	Utilization	%	Utilization	%	Utilization	%	
Slice FF Registers	2891	2.72	11506	10.81	9129	8.58	106400
Slice LUTs	2566	4.82	12664	23.8	7988	15.02	53200
Memory LUT	204	1.17	169	0.97	422	2.43	17400
BRAM	12	8.57	17	12.14	10.5	7.5	140
DSP48	5	2.27	135	61.36	20	9.09	220
BUFG	1	3.13	1	3.13	1	3.13	32

Table 4-8. Device utilization summary for the three implemented designs of Model 2 on Xilinx Zynq-7 ZC702 SoC

Resource	Design 1		Design 2		Design 3		Available
	Utilization	**%**	**Utilization**	**%**	**Utilization**	**%**	
Slice FF Registers	2898	2.72	13830	13	9009	8.47	106400
Slice LUTs	2579	4.85	12808	24.08	8111	15.25	53200
Memory LUT	204	1.17	161	0.93	582	3.34	17400
BRAM	20	14.29	16.5	11.79	16.5	11.79	140
DSP48	5	2.27	135	61.36	20	9.09	220
BUFG	1	3.13	1	3.13	1	3.13	32

Table 4-9. Device utilization summary for the three implemented designs of Model S on Xilinx Zynq-7 ZC702 SoC

Resource	Design 1		Design 2		Available
	Utilization	**%**	**Utilization**	**%**	
Slice FF Registers	3332	3.13	18054	16.97	106400
Slice LUTs	2870	5.39	12371	23.25	53200
Memory LUT	212	1.22	189	1.09	17400
BRAM	6	4.29	16	11.43	140
DSP48	5	2.27	135	61.36	220
BUFG	1	3.13	1	3.13	32

4.3.3.2 Power Consumption

The power consumption results reported by Vivado tool (the confidence level is medium) for all implementations are detailed in Table 4-10 for the three models. The on-chip total power consumption of all implementations of the three models are considered to be small values, meeting the critical embedded system constraint while achieving an energy-efficient system. For the two large-scale models, the least power consumption results of 1.756 and 1.758 watts were achieved from design 1 that has the least area. Also having almost equal values of power for the two models of different sizes is promising for implementing large-scales SVMs with low power values. The device statically consumed 9% of the total power consumption, whereas the remainder 91% was consumed by the dynamic activity, mostly by the Zynq PS component (95% of total dynamic power) compared to other on-chip components. Similar figures were realized for the other implementations of the two models. For model 1, design 3 demonstrated slightly higher power than design 2. However, design 2 consumed the highest power dissipation of 2.125 watts for model 2, as a result of being the design with the highest utilization rates. The power dissipation by the DSPs was increased from less than 1% in design 1 to 5% of the total dynamic power in design 2 for model 2.

Consequently, the pipelined design 1 is capable of meeting the most challenging constraint of "low-power consumption", in addition to meeting other vital constraints of high performance, low area and low cost. Accordingly, our implementation of the proposed low-power system is so promising for the deployment in an embedded environment, aiming to reach our ultimate goal of realizing an efiicient handheld device with high performance and low cost.

For the small-scale model, both designs' implementations also showed low levels of area and power consumption. Among the other two models and designs' implementations, this model achieved the least power consumption of 1.686 watts with design 1. In addition, it is shown (from different implementation experiments) that when using less operating frequency for the ARM processor in the Zynq PL part, the power dissipation is decreased (as the dynamic power consumption is directly proportional to the clock frequency). This method could be applied in the future to optimize the power consumption of the other models.

Table 4-10. On-chip components power consumption summary for the implemented designs of each model on Xilinx Zynq-7 ZC702 SoC (in Watts)

On-Chip Component	Model 1			Model 2			Model S	
	Design 1	**Design2**	**Design 3**	**Design 1**	**Design2**	**Design 3**	**Design 1**	**Design2**
Clocks	0.012	0.027	0.029	0.011	0.064	0.029	0.034	0.072
Logic	0.006	0.015	0.03	0.006	0.083	0.027	0.018	0.015
Signals	0.008	0.025	0.039	0.008	0.123	0.036	0.027	0.042
BRAM	0.004	0.001	0.015	0.005	0.031	0.012	0.021	0.012
DSPs	0.003	0.031	0.013	0.003	0.091	0.013	0.009	0.043
PS7	1.567	1.567	1.567	1.567	1.567	1.567	1.424	1.424
Total Dynamic Power	1.601	1.666	1.693	1.601	1.959	1.684	1.533	1.609
Device Static Power	0.155	0.158	0.158	0.156	0.166	0.158	0.153	0.157
Total On-Chip Power	1.**756**	1.824	1.851	**1.758**	2.125	1.842	**1.686**	1.766

4.3.3.3 Processing Speed and Time

The AXI-Timer IP core was exploited to compare the total computing time between running the designed code (Figure 3-6) of the SVM on ARM processor in PS part and on hardware implemented in PL part of the Zynq SoC. Due to size limitation of the embedded DDR3 memory, the application/program could not run completely at the SDK tool, because of the big data size used by both models 1 and 2. Accordingly, the small-scale model S was used for evaluating the processing speed and time by using XTimer measurements from running the application on Zynq Soc, while model 1 and 2 have been evaluated based on timing simulation results.

For running the test program on Zynq SoC for model S using the SDK, only 3693 clock cycles were required for running the hardware IP including data streaming via the DMA. However, the embedded ARM processor used 77367 clock cycles as a total run time of a similar software C coded function. Therefore, a significant acceleration factor greater than 20x was achieved by using the implemented hardware accelerator/IP. The used operating frequency for both the PS Zynq/ARM processor and PL/FPGA was 250 MHz. Accordingly, the processing times of 14.77 µs and 309.47 µs were achieved for the IP and ARM respectively (approximately equal processing time (15.32 µs) for the HLS IP was estimated from the C/RTL co-simulation at the HLS tool (see Figure 3-3)). Additionally, the ARM processor could run at the most optimized option to reach 89.59 µs, whilst the acceleration factor was still significant with the order of greater than 6x. Table 4-11 (a) shows the values of number of clock cycles and processing time for running on the PL, ARM and optimized ARM at 250 MHz in addition to the speedup factors. By using the available maximum operating frequency of 250 MHz and 666.67 MHz for the PL and ARM respectively, an acceleration factor of greater than 10x was achieved regarding the total number of clock cycles, whilst >3x was achieved regarding the processing time (>2 and 1 respectively compared to the optimized ARM) (Table 4-11 (b)). In addition, the least processing time of 11.26 µs was achieved with the pipelined design of model S.

Similar acceleration values were achieved for the unrolled design of model S, acceleration factors of 20.96x and 6.07x were achieved for the IP (3690 clock cycles) over the ARM (77328 clock cycles) and optimized ARM (22398 clock cycles). At 250 MHz, the processing time of 14.76 µs was achieved for the hardware IP, while 309.3 µs and 89.59 µs were achieved for the ARM and optimized ARM respectively.

Similarly for the larger implemented pipelined model 1, the timing simulation results were used for the evaluation that have exactly equal processing times to the HLS synthesis results in Table 4-4 (also, it is shown from running model S on Zynq SoC that processing time is nearly equal to the co-simulation results). 14138 clock cycles were required by the HLS IP (141 µs processing time), while 309378 and 90585 clock cycles were required by the ARM and optimized ARM respectively, for running the same model at 100 MHz. Therefore, great speedups of 21.88x and 6.41x were achieved from the hardware accelerator compared to the embedded ARM without and with optimization, which have similar acceleration figures of model S. Similarly for the unrolled design (83.66 µs), high acceleration factors of 36.98x and 10.83x were achieved. Table 4-12 summarizes the processing times and speedups values for both designs of model 1 at 100 MHz. Consequently, a real-time embedded SVM system could be realized that is scalable and easily extended offering high performance.

Similarly for the implemented pipelined model 2 (196.26 µs), acceleration factors of 22.84x and 6.61x were achieved compared to the ARM (448353 clock cycles) and optimized ARM (129804 clock cycles). Similarly for the unrolled design (136.98 µs), high acceleration factors of 32.73x and 9.48x were achieved (Table 4-13). That's shows a promising acceleration to reach a high performance embedded classification system, while increasing scalability.

Table 4-11. Timing Summary of the implemented Model S

(a) FPGA and ARM at 250 MHz operating frequency

Computing Time		**FPGA**	**ARM**	**Optimized ARM**	**Speedup1 (ARM/ FPGA)**	**Speedup2 (Optimized ARM/ FPGA)**
Design 1	**Clock Cycles**	3693	77367	22398	**20.95**	**6.06**
	Processing Time (µs)	14.77	309.47	89.59	**20.95**	**6.06**
Design 2	**Clock Cycles**	3690	77328	22398	**20.96**	**6.07**
	Processing Time (µs)	14.76	309.31	89.59	**20.96**	**6.07**

For each implemented design, the table shows the values of number of clock cycles and processing time in µs for running the designed code on the PL/FPGA, PS ARM CPU and optimized ARM (with the most optimization option in the tool) at the corresponding operating frequency in MHz. The speedup factors are calculated for comparing the hardware runtime versus the ARM and optimized ARM processor in the last two columns.

(b) FPGA at 250 MHz and ARM at 666.67 MHz operating frequency (Design 1)

Design 1	FPGA	ARM	Optimized ARM	Speedup1 (ARM/ FPGA)	Speedup2 (Optimized ARM/ FPGA)
Clock Cycles	2815	28968	8431	10.29	2.995
Processing Time (µs)	11.26	43.45	12.65	3.86	1.12

Table 4-12. Processing Times of the implemented Model 1 at 100 MHz

Processing Time (µs)	FPGA	ARM	Optimized ARM	Speedup1 (ARM/ FPGA)	Speedup2 (Optimized ARM/ FPGA)
Design 1	141.38	3093.78	905.85	**21.88**	**6.41**
Design 2	83.66	3093.78	905.85	**36.98**	**10.83**

Table 4-13. Processing Times of the implemented Model 2 at 100 MHz

Processing Time (µs)	FPGA	ARM	Optimized ARM	Speedup1 (ARM/ FPGA)	Speedup2 (Optimized ARM/ FPGA)
Design 1	196.26	4483.53	1298.04	**22.84**	**6.61**
Design 2	136.98	4483.53	1298.04	**32.73**	**9.48**

4.3.3.4 Classification Accuracy

Using the Xilinx SDK tool, an application of classifying a test instance has been developed in C that implements the proposed test bench presented in section 3.4.3 (Figure 3-8). Some test instances of extracted features were tested to be correctly classified by all implemented SVM IPs (one at a time). Model S has been validated for online classification while running on the Zynq SoC, whilst other implemented models have been verified based on simulation results (due to memory size limitation at run-time).

Model 1 (248 SVs, 27 features) has been trained using the cross-validation method to produce a model with a good accuracy of 80.85%. In order to verify the hardware classification result of our implemented SVM IP and compare it with the software result,

we monitored the calculated distance value in the decision function (3.1) for the test instance before applying the sign function (if-else statement in pseudo code as in Figure 3-6) for determining its class (+1 or -1). By using the C/RTL co-simulation results from the HLS tool, the distance value from our implemented IP was easily compared to that value generated from the SVM-Light window application/software to be identical for all tested instances (see Figure 4-3, the percentage error is equal to ***0.000016%***, which is extremely low and almost equal to zero (generated from displaying precision approximation)). Accordingly, the percentage error is equal to zero, reserving the classification accuracy level/rate.

Same performance was achieved from the small-scale model S with a great accuracy of 97.92%, which has been validated by both simulation results and online classification results while running on the Zynq SoC. Therefore, a reliable scalable online SVM classification with a high classification accuracy could be realized with no loss in accuracy rate using our proposed design, meeting critical embedded system constraints of optimized speed, area, power and cost.

4.3.4 Comparison with Our Previous Proposed Hardware/Software Co-design

The previous hardware/software co-design [11] was proposed in order to implement the complicated dot-product calculation onto the hardware/PL, while the rest of the required calculations of the decision equation (3.1) was running on the software/PS in a single device (Zynq SoC) (Sections 3.3 and 4.2). In this proposal, the hardware design was extended to implement the whole function (full SVM) onto the PL as a Zynq coprocessor/accelerator IP. The previous design depends on the number of features, whilst the extended design depends on the number of both SVs and features in order to implement the SVM classifier.

The implementation of the previous design was realized for the SVM classifier that has the same size as model 2 (27 features and 346 SVs). So, design 1 (pipelined) and design 2 (unrolled) of model 2 were chosen in order to compare the implementation results of the previous design with this extended hardware design.

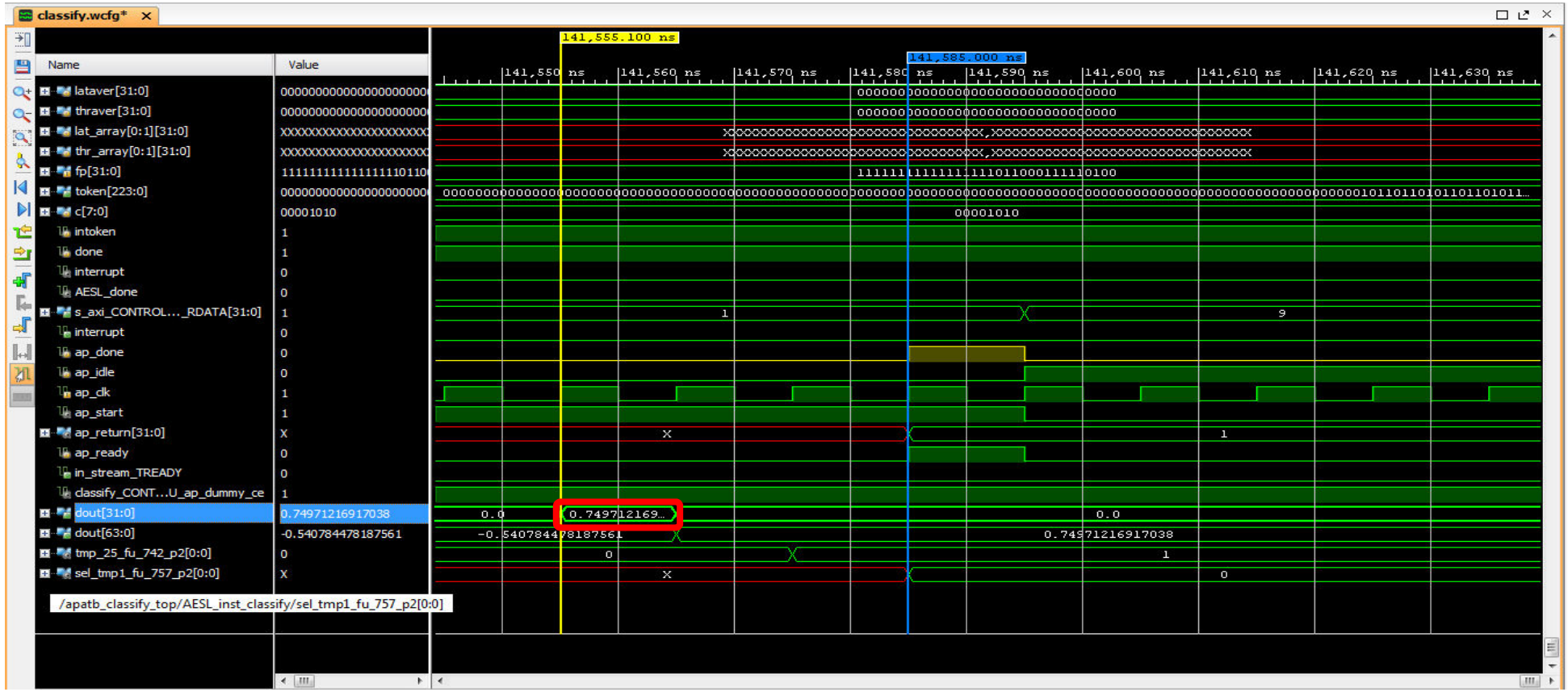

*Figure 4-3. Simulation view (timing diagram) of the implemented model 1, where the distance value (0.74971217) is displayed in "dout" signal (surrounded by a red rectangle), and compared to the true value of the software application for classification accuracy validation (%error= |true value (software value: 0.74971229)-measured value (simulation value: 0.74971217)|/true value (0.74971229) *100 = 0.000016%).*

Table 4-14 shows the implementation results of model 2 implemented on Zynq SoC using our previous hardware/software co-design with the unrolled HLS directive [11] and this proposed extended design (with the pipelined (design 1) and unrolled (design 2) directives). The extended hardware implementation of design 1 successfully extended our previous implementation with only 0.02 watts increase in total power consumption. Regarding resource utilization, only extra 17 BRAMs and 31 memory LUTs were utilized with the same number of DSPs, while other resource utilization was decreased (FFs and LUTs). Compared to design 2, an additional 0.387 watts was required than that of the previous implementation, and more resources were utilized for all resources except for Memory LUTs.

It is clear that design 1 showed more optimization in area and power results compared to design 2 for extending the previous design, where the unrolling technique was used. Accordingly, more design extension and scalability could be easily achieved with significant implementation results by using the proposed cost-effective pipelined design. Consequently, our implemented system meets the challenging constraints of low cost, area, and power dissipation, which is promising for achieving an efficient embedded system with high performance.

Table 4-14. Implementation results comparison of the implemented Model 2

Results		**Proposed Design**		**Our Previous Design [11]**
		Design 1	**Design 2**	
Resource Utilization	Slices	2898	13830	5584
	LUTs	2579	12808	4373
	Memory LUT	204	161	173
	BRAM	20	16.5	3
	DSP48	5	135	5
Power (W)		1.758	2.125	1.738

4.3.5 Comparison with Related Works

In order to compare with some relevant work, our proposed pipelined design 1 of model 1 was selected, as being the most effective design with real data size for our case study "melanoma detection". The selected model showed the best implementation results of high performance and low cost, area, and power among the implemented designs and models (apart from the small-scale model S). A general comparison is presented first, then a detailed comparison with some selected implementations is introduced.

4.3.5.1 Detection Accuracy

Some FPGA-based implementations in the literature suffered from some loss in the SVM classification accuracy as stated in Section 2.5.6. Interestingly, no loss in the classification accuracy rate was achieved from our hardware implemented models, as the calculated classification values were exactly equal to the corresponding software results, which ensures preserving the online classification accuracy rate on FPGA. That's increases the feasibility of implementing other SVM models for different applications with different sizes using our proposed design, while preserving same classification accuracy level without loss. In addition, our implemented SVM classification system showed a high detection accuracy of more than 80%, which could be used and applied in real life (>97% for model S). Precisely, our implemented system is considered to be accurate and reliable with zero loss in classification accuracy rate, in contrast to other reported implementations in existing literature [67, 91, 93, 96, 98] (have recorded some loss in accuracy).

4.3.5.2 FPGA Technology

The fact that many implementations in the literature used old versions of FPGAs and very limited used recent ones [58, 86], motivated us to use the latest FPGA technologies that are more powerful and feature-rich. Accordingly, we used the recent Xilinx Zynq-7000 SoC platform for our implementation. It also allows for a higher operating frequency, in contrast to numerous previous implementations in the literature. Besides, the modern Vivado Design Suite (2016.1 version) was exploited for our development process that applies the latest UltraFast HLS design methodology, which decreases hardware development effort, accelerates design productivity and shorten time-to-market. Consequently, our implemented system on Zynq achieved more optimized implementation results.

4.3.5.3 Processing Speed and Time

Regarding the processing speed and time, the implemented system on hardware has significantly accelerated the processing power up to 36 orders of magnitude over similar software implementation running on the embedded ARM processor. By using the recent FPGA technology that offers a higher operating frequency of 250 MHz, a processing time of 56.6 μs (33.5 μs in case of the unrolled design) was achieved, demonstrating a real-time performance (the least time of 11.26 μs was achieved for the pipelined model S). In addition, the implemented system outperformed some existing implementations regarding processing time [21, 55, 56, 67, 70, 89, 91]. Therefore, a real-time effective embedded system could be realized.

4.3.5.4 Hardware Resource Utilization

Regarding the hardware resource utilization, our proposed implementation reported very low utilization of all resources, which ensures the realization of a low power system with low cost and feasibility for extendability and scalability. Our implemented system demonstrated less resource utilization than some of the previous implementations for different applications in the literature [21, 55, 67, 69, 70, 93, 98].

4.3.5.5 Power Consumption

Interestingly, our implemented low-power system meets the most critical constraint of embedded systems "low power consumption", whereas very few of such implemented system exists in the literature. Also, it has been found that most existing implementations had not included any measurements for the power consumption. Specifically, our implementation demonstrated lower power dissipation compared to previously reported implementations [21, 74, 93, 98].

4.3.5.6 Detailed Comparison and Discussion

Some of the existing related implementations of binary SVMs were selected to be compared with our implemented model 1 and model S using the pipelined design 1 (apart from the different applications used), which is summarized in Table 4-15. Using the recent UltraFast HLS design methodology, our implementations achieved significant hardware results compared to others that used the traditional pipelined architectures and common systolic array architectures.

It is clear that lower resource utilization was demonstrated with real moderate size of SVM parameters with the linear kernel. The power consumption is significantly low (less

than 1.8 watts), compared to a very high power of 15 watts in [21], while others didn't consider this critical constraint. By using the recent FPGA platform, a higher operating frequency was used to achieve extremely less processing time than [21] and comparable time to [70] and [55]. Regarding model 1, the least processing time of 33 µs was demonstrated by using the faster unrolled design that offers slightly higher cost and area from the pipelined design, however, the lowest 11.26 µs was achieved by the small-scale model S using the cost- and energy-efficient pipelined design. Acceptable classification accuracy rate higher than 80% with zero loss was verified for our application, while others didn't validate their hardware classifiers [21, 55]. Accordingly, our implemented models on the recent hybrid Zynq SoC platform achieved optimized results for the hardware resource utilization, power consumption, detection speed and processing time with high classification accuracy rates using real data for melanoma detection.

Finally, to the best of our knowledge, our Zynq implemented embedded system using the HLS method is considered to be the first FPGA-based SVM classifier exists in the literature that targets melanoma classification. In addition, our implemented system successfully overcame most challenges exit in the literature of meeting critical embedded system constraints of high performance, flexibility, scalability, and low levels of area, cost, and power consumption, while reaching reliable effective classification system with high classification accuracy.

4.4 Proposed Hardware Design of SVM HLS IP using BRAMs

4.4.1 Implemented Models and Hardware Implementation Results

Two SVM models/IPs have been implemented based on the proposed design in Section 3.5 on Zynq SoC after extracting required data from the training phase. The Vivado block design view is shown in Figure 4-4 for implementing the proposed hardware/software system in Figure 3-10. First, the trained small-scale model "Model S" of 61 SVs and 27 features has been implemented using this proposed design (described in Section 4.3.1 and Table 4-3). The trained model has a significant accuracy of 97.92% for melanoma detection, which showed low values of hardware resource utilization with only 2 watts of total power consumption for Zynq implementation (see Table 4-16).

The device statically dissipated 8% (0.17 W) of the total power consumption, whereas the remainder 92% (1.89 W) was consumed by the dynamic activity, where mostly

dissipated by the Zynq PS component (74% (1.4 W) of total dynamic power) compared to other on-chip components. Interestingly, a significant acceleration factor of 26x was achieved compared to running an equivalent classification software processing on the embedded ARM processor at the Zynq PS part. A processing time of 11.46 µs (2865 clock cycles) was achieved by using a high operating frequency of 250 MHz offered on the modern Zynq platform, while 309.36 µs (77340 clock cycles) was achieved for the ARM processing at the same frequency.

The other large-scale model "Model 1" of 248 SVs and 27 features with 80.85% accuracy has been implemented with this proposed design (described in Section 4.3.1 and Table 4-3). Similarly, low values of resource utilization and power consumption (2.65 W) (see Table 4-16) were achieved with an acceleration factor of 32x and 39.3 µs processing time, while 1.24 ms was required by the ARM processor. Consequently, the proposed design is capable of meeting the challenging constraints of embedded systems of high performance, low area, cost and power, in addition to flexibility and scalability, which is so promising to realize a low-cost handheld device for efficient melanoma detection.

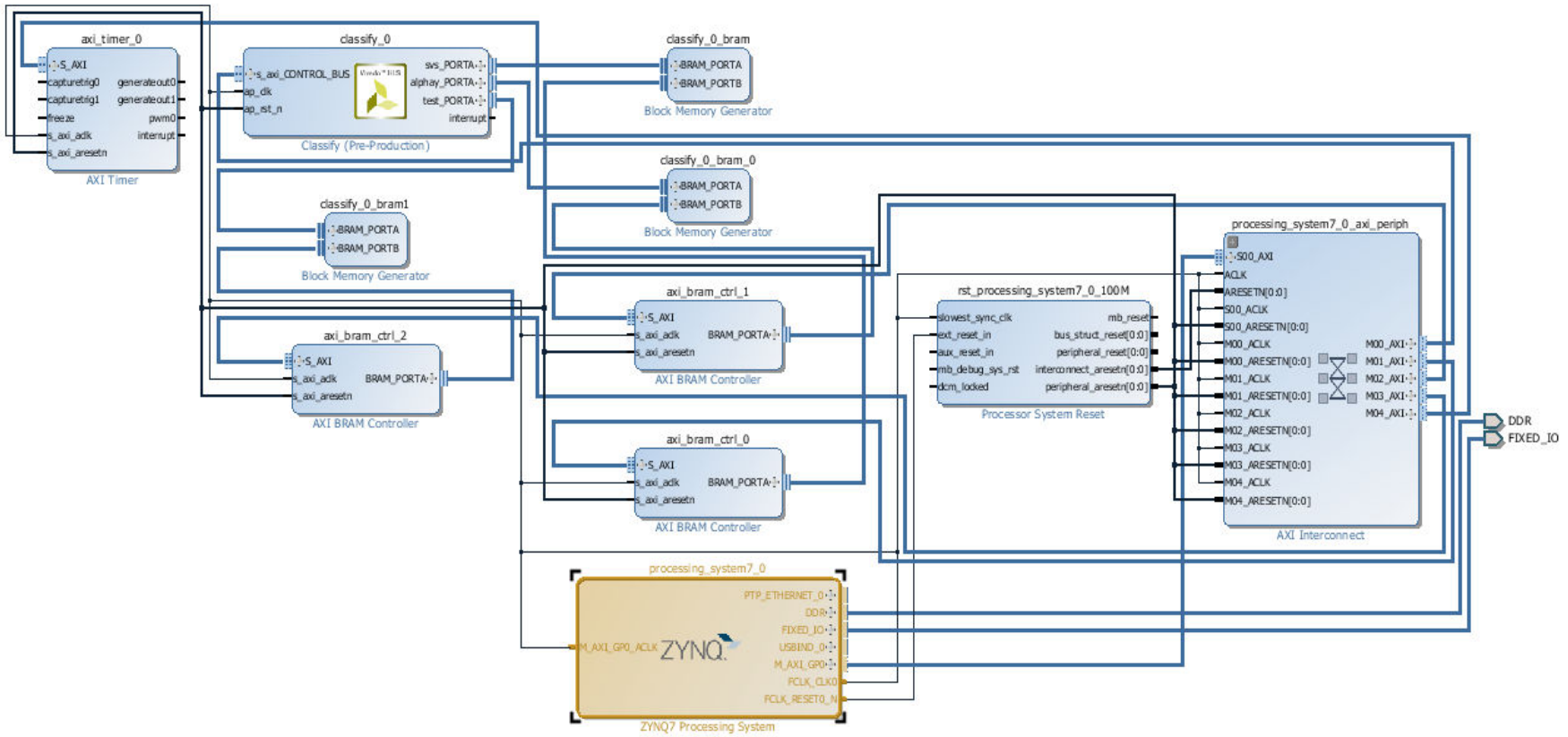

Figure 4-4. The Vivado block design view of the designed embedded system implemented on Zynq SoC, where the implemented HLS SVM IP "Classify" is connected to the Zynq7 Processing System and other IPs using AXI interface.

Table 4-15. Comparison of the implemented Model 1 and Model S with Related Works

Related Works		**Model 1**	**Model S**	**[21]**	**[69]**	**[70]**	**[55]**
FPGA Recourses	Slice FF Registers	2891	3332	59208	23220	5162	12674
	Slice LUTs	2566	2870	122637	57296	8887	41135
	BRAM	12	6	2049	83	74	132
	DSP48	5	5	N/A	40	64	64
Power (W)		1.756	1.686	15	N/A	N/A	N/A
FPGA		Zynq-7000	Zynq-7000	Virtex-5 LX220	Virtex 5-LX110T	Virtex 5-LX110T	Virtex 5-LX110T
Features size		27	27	N/A	N/A	400	512
Number of SVs		248	61	16	74-467	818	512
Application		Melanoma Detection	Melanoma Detection	Skin Classification	Object detection	Object detection	Pedestrian detection

Kernel	Linear	Linear	Gaussian	Polynomial RBF	Polynomial	Linear Polynomial RBF
Detection Accuracy	80.85%	97.92%	N/A	76-78%	88%	N/A
processing time	141, 56.6 µs (83, 33.5 unrolled)	11.26 µs (14.76 unrolled)	0.02 s	N/A	54 µs	44.84, 82.5 µs
Frequency(MHz)	100, 250	250	200	100	100	50, 92
Hardware Architecture	HLS-based pipelined (unrolled)	HLS-based pipelined (unrolled)	Fully pipelined	Systolic array	Systolic array	Pipelined

The comparison is based on the table attributes of different FPGA resource utilization, power consumption in watts, used device/FPGA, dimensionality (features size), number of SVs, the applied application, kernel function implemented, detection/classification accuracy in %, processing time, frequency in MHz, and hardware architecture/design used in the implementation.

The implemented SVM models have been validated by running the developed applications (test benches) on the Zynq SoC for online melanoma classification. Some instances were tested to be correctly classified by all implemented models as melanoma/benign class with exactly the same classification result of the software application. Accordingly, the classification accuracy level was preserved without any loss from the hardware implementation, in contrast to other reported implementations in the literature [67, 91, 93, 119]. Interestingly, our implemented SVM models showed high acceptable detection accuracy that could be used and applied in real life.

Table 4-16. Implementation results for the implemented Model S and Model 1 on Xilinx Zynq-7 ZC702 SoC

Models	**Resource Utilization (%)**					**Power (W)**
	Slices (106400)	**LUT (53200)**	**LUT-RAM (17400)**	**BRAM (140)**	**DSP (220)**	
Model S	10874 (10%)	7218 (14%)	874 (5%)	48 (34%)	5 (2%)	2.06
Model 1	30006 (28%)	17506 (33%)	2873 (17%)	48 (34%)	5 (2%)	2.65

4.4.2 Comparison with Our Previous Proposed Designs

The proposed hardware design successfully extended our initial proposed hardware/software co-design [11] (Sections 3.3 and 4.2) to implement the whole SVM classification function onto the Zynq PL(FPGA), while some extra resources were utilized (same number of DSPs) with very little increase in power dissipation (< 1 W).

Compared to the previous design (Sections 3.4 and 4.3) that is based on streaming required data via the stream interface/bus instead of using BRAM interfaces, same SVM trained models, model 1 and model S have been implemented using the BRAM-based design with applying the pipelined technique/directive to boost required processing (corresponds to Model 2 and Model 1 respectively in our published paper [13]). Table 4-17 shows the hardware implementation results of implementing the pipelined design 1 of both model 1 and model S using the previous proposed stream-based design (Table 4-7 and Table 4-9) compared to models implementations using this proposed BRAM-based design (Table 4-16) [13].

Table 4-17. Implementation results comparison of the implemented Model 1 and Model S between using this proposed BRAM-based design and the previously proposed stream-based design

Results		**Model 1**		**Model S**	
		Stream-based Design	**BRAM-based Design**	**Stream-based Design**	**BRAM-based Design**
Resource Utilization	Slices	2891	30006	3332	10874
	LUTs	2566	17506	2870	7218
	Memory LUT	204	2873	212	874
	BRAM	12	48	6	48
	DSP48	5	5	5	5
Power (W)		1.756	2.65	1.686	2.06
Processing Time (µs)		56.6	39.3	11.26	11.46
Classification Accuracy %		80.85	80.85	97.92	97.92

The stream-based design consumed less hardware resource utilization with same number of DSPs and lower power consumption for both models, while keeping the same level of the classification accuracy. This shows that our different proposed designs are promising for preserving accuracy without loss, while improving hardware implementation results for reaching an optimum solution. For the small-scale model S, the stream-based design consumed 0.4 watts less power consumption than the BRAM-based design. Also, fewer hardware resources were utilized, especially for the BRAM utilization where only 6 BRAMs were utilized, while the other design used 48 BRAMs. Similar figures of less resources and power were recorded for the large-scale model 1.

Regarding the processing time, the stream-based design spent less 0.2 µs than the other design for model S at 250 MHz, while extra 17.3 µs was required for model 1. It can be considered as another justification of the existing trade-off between performance and area/cost. Apparently, both designs showed low processing time for variable model's size, while fewer resources and less power were demonstrated by the stream-based design than

the BRAM-based design. Consequently, both designs are feasible to realize a real-time embedded classification system to be easily integrated within a cost- and energy-efficient handheld device dedicated for early detection of melanoma.

4.4.3 Comparison with Related works

The implemented SVM classifiers using the proposed BRAM-based design were remarkably accelerated onto hardware to realize real-time embedded systems, which also outperformed some existing implementations regarding processing time [21, 55, 67, 70, 91]. Moreover, the implemented models were realized onto the recent FPGA technology of the modern Zynq SoC, in contrast to most existing implementations that used old versions of FPGAs. The Zynq implemented systems achieved significantly low resource utilization and power consumption, which demonstrated lesser values than other implementations in the literature [21, 55, 67, 69, 70, 93, 119].

Finally, our Zynq implemented models achieved optimized hardware results, meeting crucial embedded system constraints with reliable classification for melanoma detection.

4.5 Proposed Multi-core Architecture (Cascade SVM Classifier) and Dynamic Cascaded SVM Architecture

Using the Vivado HLS tool, the SVM HLS IP has been developed and exported to be integrated into the proposed Zynq SoC (Figure 3-14) that was designed in a block diagram as depicted in Figure 4-5 for n=2 (2-stage cascaded architecture) using the Vivado design tool. The designed Zynq system was synthesized, placed and routed and finally the bitstream was generated to be exported for the Xilinx SDK tool to run an online classification application on Zynq for testing and verification. For the DPR-based hardware system, three configurations have been implemented as a static top-level design with BB module, RM-M (for melanoma-sensitive SVM) and RM-N (for non-melanoma-sensitive SVM) by the aid of the Vivado design suite using the floorplanning tool. For each configuration, two bitstreams were generated, full and partial bitstreams for different configuration options.

In the next sub-sections, the implemented SVM models are introduced, then different experimental results and comparisons are presented and discussed.

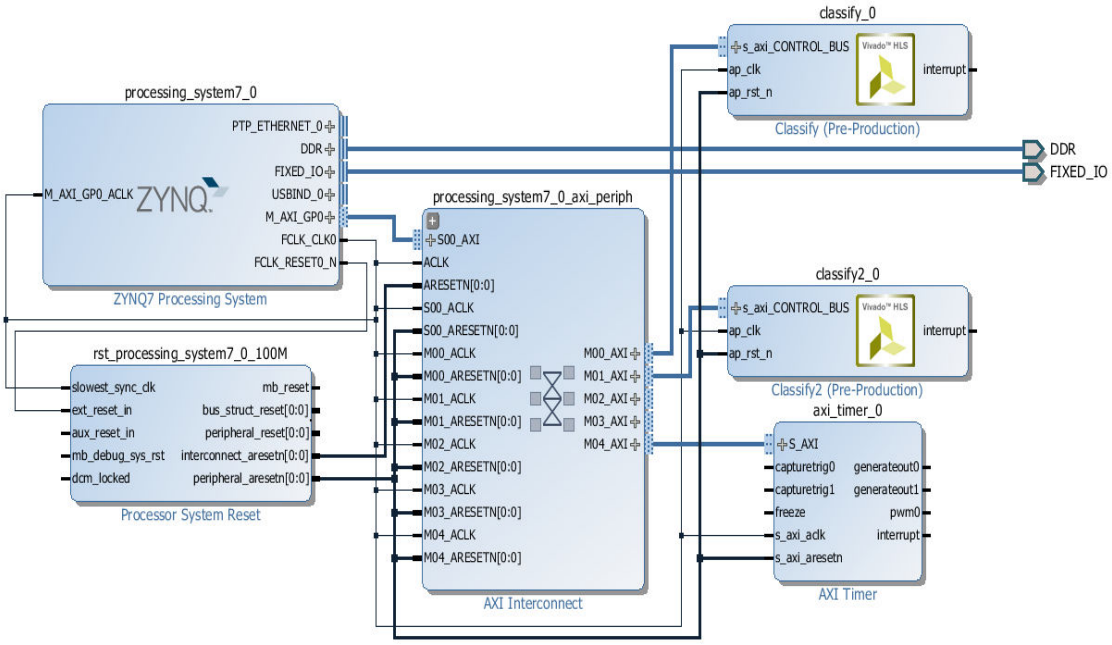

Figure 4-5. The Vivado block design view of the designed embedded system (2-stage cascaded architecture) implemented on Zynq SoC, where two hardware-friendly HLS SVM IPs "Classify" are connected to the Zynq7 Processing System and other IPs using AXI interface.

4.5.1 Implemented SVM Models

In order to implement the proposed 2-stage cascaded architecture, the used dataset was manipulated to build two different datasets in order to generate a melanoma-sensitive model as "Model M" and non-melanoma-sensitive (benign) model as "Model N". In order to achieve a higher accuracy for the trained models, data scaling and normalization techniques were applied to the original dataset, as well as, using the cross-validation technique in the training phase. Table 4-18 summarizes different parameters of the two implemented SVM models (model M is equivalent to the previous implemented small-scale model S). For each model, the table shows the number of SVs generated with the number of instances used in the training dataset of 27 features each. Finally after generating the trained SVM models offline, the models' data were extracted to implement the trained SVM model on hardware, using the proposed hardware-friendly design in Section 3.6.1 that depends on the number of features.

Table 4-18. Parameters of implemented SVM Models

SVM Model	Training Dataset		SVs	Accuracy %
	Melanoma instances	Benign instances		
Model M	100	44	61	97.92
Model N	67	144	139	72.51

The table shows the number of SVs and the classification accuracy rate % generated with the number of instances used in the training dataset of 27 features each.

4.5.2 Classification Accuracy

Using the Xilinx SDK tool, an application of classifying a test instance has been developed in C to test and validate the implemented classification systems on Zynq SoC. Similar to our previously implemented models, the test instances were correctly classified by all implemented systems using this proposed design (one at a time). Regarding the classification accuracy, the two models M and N have been trained using the cross-validation method to produce good accuracy of 97.92% and 72.51% respectively. Both models were used in the implemented cascaded scheme for improving classification performance and verifying melanoma diagnosis. The experimental results demonstrated that every hardware classification result of the implemented classifiers was exactly equal to the corresponding software classification result. In order to verify the hardware classification result of the implemented SVM IP and compare it with the software result, we monitored the calculated distance value in (3.5) for the test instance before applying the sign function ("Distance_value" in pseudo code in Figure 3-12) for determining its class (+1 or -1). By using the C/RTL co-simulation results from the HLS tool (similar to Figure 4-3), the distance value from all implemented IPs was easily compared to that value generated from the SVM-Light window application/software to be identical for all tested instances. Accordingly, the percentage error was equal to zero (similar behaviour to all previously proposed designs), reserving the classification accuracy level without any loss from the hardware implementation, in contrast to some existing implementations in the literature as stated before.

4.5.3 Processing time and Speed

From the HLS tool, the synthesis results showed processing time of 1.5 µs that is required for each model M and N using frequency of 100 MHz. For the 2-stage cascaded architecture, the processing time is nearly doubled (3 µs) if the test sample is negative (non-melanoma) and is equal to one IP's processing time otherwise. For verifying this synthesis/simulation results, the XTimer IP was connected to the designed SoC as in Figure 3-14 to measure clock cycles required for running the IP on Zynq PL through executing the developed software program (application) on the PS ARM processor using the Xilinx SDK tool. For 250 MHz, less processing time of 1.8 µs was demonstrated for the cascaded architecture [13], and a speedup of 5x was reached compared to a software implementation running on ARM processor.

4.5.4 Hardware Resource Utilization and Power Consumption

Table 4-19 summarizes the implementation results of all implemented systems/configurations (monolithic Model M and Model N, Cascade Model and Dynamic Cascade Model with RM-M and RM-N configurations) on Zynq SoC using 100 MHz frequency. Overall, all resource utilization percentages are considered significantly low, while dissipating low power of only 1.5 watts for all implemented systems, which promises more design extension and expansion for larger scale problems by using our proposed design as well as meeting critical embedded system constraints. The power consumption results were reported by Vivado tool (the confidence level is medium). The device statically dissipated 10% of the total power consumption, whereas the remainder 90% was consumed by the dynamic activity, mostly by the Zynq PS component (95% of total dynamic power) compared to other on-chip components. The implemented 2-stage cascaded architecture utilized almost double of the monolithic system-based utilization for all hardware resources and with only 2 extra memory LUTs, whilst power consumption increased by only 0.2 watts. The two monolithic models M and N utilized equal resources and power as both have the same size of 27 features. For the DPR cascade system, both RM-M and RM-N configurations utilized equal resources and power to be dynamically reconfigured at runtime, which is less than the baseline cascaded architecture and nearly equal utilization to the monolithic model (with slightly extra slices and LUTs) with only 0.1 excess power dissipation. That's shows that by using our novel DPR-based design, we can realize the cascaded scheme with efficient resource utilization and power

consumption that is nearly equal to a single SVM model, while gaining advantages of using the cascaded scheme of improving classification speed (for positive/melanoma samples) and accuracy as well as diagnosis verification.

Table 4-19. Implementation results for the implemented monolithic Model M and Model N, Cascade Model and Dynamic Cascade Model with RM-M and RM-N configurations on Xilinx Zynq-7 ZC702 SoC

SVM Model	Resource Utilization (%)					Power (W)
	Slices (106400)	LUT (53200)	LUT-RAM (17400)	BRAM (140)	DSP (220)	
Model M	1046 (1%)	858 (1.6%)	70 (0.4%)	1 (0.7%)	5 (2.3%)	1.54
Model N	1046 (1%)	856 (1.6%)	70 (0.4%)	1 (0.7%)	5 (2.3%)	1.54
Cascade Model	1785 (1.7%)	1478 (2.8%)	72 (0.4%)	2 (1.4%)	10 (4.6%)	1.56
RM-M Configuration	1050 (1%)	867 (1.6%)	70 (0.4%)	1 (0.7%)	5 (2.3%)	1.55
RM-N Configuration	1050 (1%)	862 (1.6%)	70 (0.4%)	1 (0.7%)	5 (2.3%)	1.55

4.5.5 Comparison with Our Previous Proposed Designs

Three papers have been published regarding our hardware implementation of a binary SVM classifier with linear kernel for melanoma detection on the recent hybrid Zynq SoC using the latest UltraFast HLS design methodology. Our initial hardware/software co-design [11] (Sections 3.3 and 4.2) was proposed in order to implement the complicated dot-product calculation onto the hardware/PL (similar to this proposal), and the test data is streamed using a stream interface passing by a DMA IP. Then, the hardware design was extended to implement the whole function (full SVM) on the PL as a Zynq coprocessor/accelerator in [12] (Sections 3.4 and 4.3), which depends on both numbers of SVs and features. Another similar design was proposed [13] (Sections 3.5 and 4.4), which uses BRAM interfaces to pass required data instead of using the stream interface with the DMA IP. This proposed design (Sections 3.6, 3.7 and 4.5) simplified the previous SVM IP designs to achieve a hardware-friendly design with lesser area, power and cost for design extension and realizing multi-core (cascaded) classification architecture, aiming to improve the classification performance and efficiency.

Table 4-20 shows a comparison of implementation results between our previous designs and this proposal, aiming to find an optimum solution to balance the existing trade-off between performance and cost (area and power). The implementation results (Table 4-19) in this proposal were updated to include an AXI-Timer IP in the implemented systems, in order to provide a fair comparison with our previously implemented systems that consist of the same timer IP (only used for measuring clock cycles) (extra slices and LUTs were utilized). The number of features is constant for all implemented models, equal to 27 features, while other parameters are different among them. Different optimization techniques have been applied to all proposed designs like loop unrolling and pipelining for optimizing the implementation results. The implemented hardware SVM classifiers showed speedup on hardware (Zynq PL) over software implementations on the embedded ARM processor (Zynq PS).

For the monolithic SVM, less resources were utilized with only one BRAM and same number of DSPs (5) using this proposed hardware-friendly design. In addition, the least power consumption of only 1.54 watts was achieved. Processing time was significantly decreased to 1.5 μs at 100 MHz for implementing a moderate-size model with 27 features, (regardless the number of SVs) with a high classification accuracy of 97.9 %. In addition, the energy consumption was calculated for the implemented hardware-friendly model that has the least power consumption (1.54 watts) and processing time (1.5 μs) to be 2.31 μJ, which is considered a very low value of energy consumption. Therefore, our implemented systems are capable of optimizing energy consumption for realizing a cost- and energy-efficient device.

For the 2-stage cascade classifier, less resources and power were achieved by using less frequency, while processing time was slightly increased. That's shows the trade-off of extra cost required for gaining higher speed. By using the proposed hardware-friendly design, the cascade classifier achieved lower values of area (except for DSPs which is doubled) and power than the previously implemented single SVM classifiers, while improving the classification performance and speed, in addition to the diagnosis verification.

By using the DPR feature, both resource utilization and power dissipation were significantly reduced at the cost of the configuration time (that was reduced by using the partial bitstream for reconfiguration). Accordingly with the proposed dynamic design,

more design extension and scalabiality could be easily achieved at low cost, area and power, while getting use of advantages of multi-stage cascaded architecture, aiming to enhance performance of melanoma classification and to realize a cost- and energy-efficient handheld device with high diagnosis accuracy targeting melanoma detection.

Table 4-20. Implementation results comparison of the implemented monolithic Hardware-friendly Model M and Model N and the proposed Cascade Classifier and Dynamic Cascade Classifier versus our previously proposed designs

Results		**Monolithic SVM Model**						**Cascade SVM**		
		HW/SW Co-design	**Stream-based Design Model 1**	**BRAM-based Design Model S**	**BRAM-based Design Model 1**	**Proposed Hardware-friendly Model M**	**Proposed Hardware-friendly Model N**	**[13]**	**Proposed**	**Proposed Dynamic**
Resource Utilization	Slices	5584	2891	10874	30006	1306	1302	4304	2001	1310
	LUTs	4373	2566	7218	17506	1448	1448	3414	1762	1457
	Memory LUT	173	204	874	2873	70	70	215	72	70
	BRAM	3	12	48	48	1	1	2	2	1
	DSP48	5	5	5	5	5	5	10	10	5

Power (W)	1.74	1.76	2.06	2.65	1.54	1.54	1.74	1.56	1.55
Frequency (MHz)	50	100	250	250	100	100	250	100	100
Processing Time (µs)	3.4	141.4	11.46	39.3	1.5	1.5	1.8	3	3
SVs#	346	248	61	248	61	139	61+139	61+139	61/139

For the monolithic SVM model, the HW/SW co-design represents the initial hardware/software co-design proposed in Sections 3.4 and 4.3, where the presented results are extracted from Table 4-1 and Table 4-2. The Stream-based design represents the proposed hardware design in Sections 3.4 and 4.3, where the presented results are extracted from Table 4-7 and Table 4-10 for the pipelined design 1 of model 1. The BRAM-based design represents the previous proposed design in Sections 3.5 and 4.4, where the presented results are extracted from Table 4-16 for model S and model 1. These previously proposed designs are compared to this proposed hardware-friendly design (Sections 3.6, 3.7 and 4.5), where the presented results of both models M and N are updated from Table 4-19 to include an AXI-Timer IP in the implemented systems in order to provide a fair comparison with our previously implemented systems (extra slices and LUTs are utilized). For the cascade SVM model, the last two columns represent the implemented 2-stage cascade model and the dynamic cascade model proposed in Sections 3.6, 3.7 and 4.5, where the presented results are updated from Table 4-19 after adding the AXI-Timer IP. The previous column presents an implemented 2-stage cascade model using our proposal (Section 3.6) that was implemented at higher frequency, where its results were presented in our published paper [13].

4.5.6 Comparison with Related Works

Some existing related FPGA implementations of binary SVMs with different kernel types for various applications were selected to be compared with our implemented Zynq systems of both monolithic and cascade SVM classifiers (model M, cascade model and dynamic cascade model (RM-M) in Table 4-19) that are summarized in Table 4-21 (a) and (b) respectively. Using the recent UltraFast HLS design methodology, our hardware implementations on the recent hybrid Zynq SoC achieved significant hardware results, compared to others that used old versions of FPGAs for implementing the traditional pipelined designs, multiplier-less method and common systolic array architectures. It is clear that the least resource utilization and power consumption was demonstrated by our implemented systems with real moderate size of SVM parameters with the linear kernel. The 1.5 watts of power is significantly low compared to a very high power of 15 watts in [21] and high power of 3 watts in [93], while others didn't consider this critical embedded system constraint. In addition, the least processing time of only 1.5 µs was achieved at 100 MHz operating frequency that is extremely less than others, while 0.9 µs [13] was demonstrated using 250 MHz that is equal to [54] using a higher frequency of 370 MHz. Moreover, the highest classification accuracy rate of almost 98% was demonstrated (regardless the application), while some didn't verify/validate their hardware classifiers [21, 55, 72]. Also, our proposal preserved the classification accuracy rate with zero loss compared to [93] who suffered from slightly loss in accuracy rate. Accordingly, our implemented models on the recent hybrid Zynq SoC platform achieved optimized results for the hardware resource utilization, power consumption, detection speed and processing time with high classification accuracy rates using real data for melanoma detection.

Regarding the cascaded architecture, only one research group (leading group 1) exists who worked with implementing the cascaded SVM architecture on FPGA, where only one paper [95] exists that used the DPR on a 2-stage cascaded architecture, but no implementation results are given for hardware results and power consumption. So, another implemented work of group 1 [93] that uses 4-stage cascaded architecture and without applying the DPR was selected for this comparison with our implemented 2-stage cascaded architecture. Despite using less number of features and SVs for our application "melanoma detection", we used for the two stages extremely less resources and power compared to the four stages implemented with higher number of SVs and dimension for face detection [93]. Besides, we demonstrated zero loss in classification accuracy rate,

while they suffered from 0.7% reduction in accuracy rate. Furthermore by using the DPR, more optimization in area and power was achieved, while keeping the same accuracy level and performance with extra cost of the milliseconds of the configuration time. However, the configuration time could be optimized by using the partial bitstream than using the full bitstream.

Compared to H. Hussain et al. [72] (group 2) who applied the DPR feature to their proposed architecture, we validated our proposed multi-core architecture to employ the cascaded classification for melanoma detection using real dataset, while achieving lower resources. However, they didn't apply any real application to their proposal and no results were provided for power, resources of the multi-core architecture (with or without DPR), and the classification accuracy.

Finally, to the best of our knowledge, our dynamic hardware system implemented on Zynq SoC using the HLS methodology is considered to be the first FPGA-based cascade SVM classifier exists in the literature that targets melanoma classification. In addition, our implemented system fills the main research gap exists in the literature of meeting challenging embedded system constraints of high performance, flexibility, scalability, and low cost, area and power consumption, while achieving reliable and effective classification.

Table 4-21. Implementation results comparison of the implemented monolithic Hardware-friendly Model M, 2-stage Cascade Classifier and Dynamic Cascade Classifier with related works

(a) Monolithic SVM

Ref.	FPGA Resources				Power (W)	FPGA	#SVs	Dim	Accuracy %	Processing time/speed	Frequency (MHz)	Hardware design	Kernel	Application
	Slices	LUTs	BRAM	DSP										
[54]	58688	-	800	768	-	Virtex-6	760	128	-	0.9 µs	370.096	Pipelined	RBF	-
[21]	59208	122637	2049	-	15	Virtex-5	16	-	-	0.02 s	200	Pipelined	Gaussian	Skin classification
[55]	12674	41135	132	64	-	ML505	2048	2048	-	712.66 µs	92	Pipelined	Linear	Pedestrian detection
[69]	23220	8887	74	64	-	ML505	818	400	76-78	40-122 fps	100	Systolic array	RBF Polynomial	Object detection
[70]	5162	8887	74	64	-	ML505	818	400	88	54 µs	100	Systolic array	Polynomial	Object detection

[72]	1810	1705	21	21	-	ML 403	1024	20	-	7.34 µs	142.9	DPR, Systolic array	Linear	Classifying biomedical data
Proposed Model M	**1046**	**858**	**1**	**5**	**1.54**	**Zynq-7**	**61**	**27**	**97.9**	**1.5 µs**	**100**	**Pipelined HLS-based**	**Linear**	**Melanoma detection**

(b) Cascade SVM

Ref.	FPGA Resources				Power (W)	FPGA	#SVs	Dim	Accuracy %	Processing time/speed	Frequency (MHz)	HW design	Kernel	Application
	Slices	LUTs	BRAM	DSP										
[93] 4-stage Cascade	13038	31854	131	59	3.2	ML505	254	400	84	70 fps	84	Multiplier-less	Linear Polynomial	Face detection
Proposed 2-stage Cascade	**1785**	**1478**	**2**	**10**	**1.56**	**Zynq-7**	**200**	**27**	**97.9&72.5**	**3 µs**	**100**	**Pipelined HLS-based**	**Linear**	**Melanoma detection**
Proposed Dynamic Cascade	**1050**	**867**	**1**	**5**	**1.55**	**Zynq-7**	**61/ 139**	**27**	**97.9&72.5**	**3 µs**	**100**	**DPR, Pipelined HLS-based**	**Linear**	**Melanoma detection**

The comparison is based on the table attributes of different FPGA resource utilization, power consumption in watts, used device/FPGA, number of SVs, dimensionality, classification/detection accuracy in %, processing time or speed in frame per second (fps), frequency in MHz, Hardware design methodology used, kernel function implemented, and the application.

4.6 Conclusion

This chapter presented the implementations' experiments performed for implementing our proposed hardware designs (in Chapter 3) of the SVM classifier for melanoma detection on the hybrid Zynq SoC using the modern HLS design methodology. The implemented systems have been evaluated, compared and discussed based on the experimental results, focusing on meeting critical constraints of embedded system, aiming to find an optimum hardware solution for the existing trade-offs for achieving an efficient melanoma detection. The implemented SVM classifiers on Zynq SoC have been validated for online classification of melanoma and non-melanoma samples. Additionally, a comparative study was presented for comparing the implemented systems with related implementations in the literature.

The hardware implementation results of the proposed hardware/software system demonstrated high classification accuracy of 97.9% at 11.3 µs processing time using 250 MHz operating frequency, while a significant hardware acceleration rate of up to 37x was achieved with only 2.7% resource utilization and 1.69 watts power consumption. The proposed scalable multi-core architecture was validated with a 2-stage cascade SVM classifier implementation using the proposed hardware-friendly design of the HLS SVM IP for implementing the building core of the cascaded architecture. By using the proposed simplified hardware-friendly design, the implemented cascade classifier achieved lower resource utilization (1.7% slices) and power consumption (1.56 watts) than the previously implemented monolithic SVM IPs, in addition to improvement in the classification performance.

Finally, the hardware implementation results were optimized by using the powerful DPR technology, where very low resource utilization of 1% slices and power consumption of 1.55 watts were achieved, while gaining flexibility, adaptability, scalability, and applicability. The implemented SVM classification systems on Zynq SoC using the proposed hardware designs have shown the least power consumption results among other related implementations, in addition to significantly low hardware resource utilization and processing time with significant speedups and high classification accuracy rates at low cost. Consequently, the implemented Zynq systems meet crucial embedded system constraints of high performance and low cost, resource utilization and power consumption, while achieving efficient classification with high classification accuracy,

which promises realization of a cost- and energy-efficient handheld medical scanning device for early detection of melanoma.

CHAPTER 5 Conclusions and Future Works

5.1 Introduction

The main objective of this thesis is proposing an optimized FPGA-based SVM classifier towards realizing an efficient embedded classification system, targeting early detection of melanoma as a case study. This research is a significant step towards enhancing early detection of melanoma by primary healthcare providers, who can utilize the proposed embedded classification system as part of a reliable cost- and energy-efficient handheld scanning device. This research aimed to contribute to the existing literature by considering the existing challenges, limitations and research gaps in the current literature that were identified from our novel survey study [9, 10].

In proposing the SVM classifier implementations, we also investigated and met the challenging embedded system constraints of high performance, flexibility, scalability, and low area, cost, and power consumption, while also maintaining high classification accuracy. Recent FPGA design methods, technologies, and system development tools were used for implementing the proposed designs. Different novel HLS-based designs for implementing hardware/software systems on Zynq SoC were proposed and evaluated to show improvements in the implementation results among the proposed designs and existing implementations in the literature.

This chapter presents concluding remarks for the implemented Zynq systems based on our proposed designs. It also identifies the original and significant contributions achieved. Finally, some research directions have been proposed for further research in this area.

5.2 Concluding Remarks

This novel research work [11-13] comes from arguably a third leading research group in this area in addition to two additional leading groups identified in Section 2.8. We utilized the most recent hybrid Zynq SoC platform and the modern UltraFast HLS design methodology, and developed the first HLS SVM IP/core on Zynq SoC that exists in the literature, targeting melanoma detection with high performance and low cost. A hardware/software co-design was first proposed to realize a Zynq accelerator, which was implemented in HLL (C/C++) applying available hardware directives/techniques through the HLS tool to simplify FPGA design and reduce development effort and time. The

implemented system is considered the first hardware/software SoC exists in the current period.

Next, the hardware design was extended to achieve a full SVM IP running on hardware (FPGA) and realize an efficient embedded system on Zynq SoC with high performance and low cost. Two different designs were implemented with different interfaces, to transfer the required data for processing. A stream interface using DMA protocol was applied first, and then BRAM-based interface was utilized in the proposed design. We were targeting meeting the challenging embedded system constraints, while achieving efficient classification, without any loss in the accuracy rate resulted from the hardware implementation. Experimental results demonstrated high performance, low resource utilization and power consumption, while preserving the original high accuracy level with zero loss. The implemented systems are considered as low-power and energy-efficient embedded systems that meet the critical power constraint, which showed less power dissipation than other reported implementations in the literature.

Subsequently, a scalable multi-core architecture was proposed based on adding more SVM IPs to implement a cascade SVM classifier on a single device/SoC for improving diagnosis verification. By using the proposed simplified hardware-friendly design, the implemented cascade classifier achieved lower resource utilization (1.7% slices) and power consumption (1.56 watts) than the previously implemented monolithic SVM IPs, in addition to improvement in the classification performance. Consequently, we added a new FPGA-based cascade SVM implementation in the literature besides that first implementation of group 1 (Section 2.8). Our HLS-based implementation of the cascade SVM [13] showed extremely lower hardware resource utilization and power consumption than group 1's implementation [93, 98] (Table 2-9), which indicates that our design and implementations meet the challenging embedded system constraints.

Finally, a novel dynamic (DPR-based) hardware system has been implemented for a cascade SVM classifier on Zynq SoC for melanoma detection. Using the powerful DPR feature of the FPGA, more optimization in implementation results were achieved, while gaining flexibility, adaptability, scalability, and applicability. Compared to reported implementations in the literature, we achieved significantly lesser resource utilization (1% slices) and power consumption (1.5 watts) with high performance at low cost. Consequently, the implemented system on Zynq SoC meets crucial embedded system

constraints, while achieving efficient and adaptive classification with high accuracy in addition to achieving diagnosis verification.

5.3 Contributions

The primary research contributions of this thesis are:

- The first survey of existing literature covering implementation of SVM classifiers on FPGAs had been performed and two articles [9, 10] have been developed for this unique survey. These survey articles would allow hardware designers to choose the best-fit solution for their context. This novel study contributes to the current knowledge with critical analysis and detailed comparison to identify the leading research groups, limitations, challenges, and research gaps. It was concluded that the main challenge is the difficulty of meeting important embedded system constraints of high performance, flexibility, scalability, and low resource utilisation, cost, and power consumption. In addition, the main research gap was identified by finding an optimum hardware solution for the challenging trade-off between achieving efficient classification accuracy and meeting important embedded system constraints. Accordingly, this research was conducted to overcome some of the identified challenges and limitations as well as filling some research gaps and address the proposed research questions (Section 1.4).

- The main contribution of this research is considered as proposing a novel hardware/software co-design for implementing a full SVM classifier on FPGA, and realization of an embedded SoC dedicated for melanoma detection on the recent hybrid Zynq SoC using the latest UltraFast HLS design methodology. The implemented systems on Zynq met the challenging embedded system constraints. The embedded system design was simplified, and FPGA development effort and time were decreased by using the modern UltraFast HLS design methodology, where a simple, flexible, and scalable IP-based design was proposed.

  Also by utilizing the available optimization techniques/directives in the HLS tool, design space exploration was investigated and various solutions/designs from a low-cost design to a fast design with higher cost were proposed for finding an optimum balance for the existing trade-off between speed and area. The implemented SVM classifier “model 1” on Zynq SoC with the pipelined design is considered to be an

optimized classifier for our application “melanoma detection”. This implemented model is an efficient trained model of realistic size with only ***2.7%*** slices utilized and ***1.7*** watts power consumed, while classifying was achieved at ***56 µs*** with ***80.8%*** accuracy at 250 MHz..

Interestingly, the SVM classification process was significantly accelerated on FPGA by a factor up to 37x outperforming an embedded processor, whereas 11.3 µs processing time was achieved using high operating frequency of 250 MHz. Moreover, an effective SVM classification with high accuracy of 97.9 % was realized on hardware without any loss in accuracy rate, in contrast to other existing implementations in the literature. The implemented systems on Zynq SoC are considered to be cost-effective systems that showed the least power consumption results among other related implementations, in addition to significantly low hardware resource utilization and processing time with significant speedups and high classification accuracy rates at low cost. Subsequently with our proposal, a real-time embedded system was realized that successfully overcame most challenges exist in the literature by achieving an efficient and reliable classification with high performance and low cost, area, and power consumption.

- A scalable multi-core architecture based on multi-SVM core was proposed to achieve multi-purpose classification with different scenarios (e.g. ensemble, multi-class classification). Based on experimental/implementation results of our previous proposed designs, a simplified design of the HLS SVM IP was proposed as a hardware-friendly design and selected to act as a building block for the multi-core architecture in order to reduce hardware complexity and achieve optimized results. The proposed multi-core architecture was validated with a 2-stage cascade classifier implementation based on a melanoma-sensitive SVM classifier with 98% accuracy and a benign-sensitive classifier with 73% accuracy to enhance melanoma detection.

  This implemented embedded system is the first FPGA-based cascade SVM classifier for melanoma detection in the literature that was implemented on Zynq SoC using the HLS method. In addition, it is considered a significant contribution to the literature as an additional and novel FPGA-based cascade SVM classifier implemented using the latest FPGA technology besides the first one implemented for object detection by the

first leading research group [93-98] identified in Section 2.8. Also, our implemented Zynq system showed significantly low resource utilization (1.7% slices) and power consumption (1.56 watts) with high performance at low cost, which meets critical constraints of embedded systems.

- Another original and significant contribution is the implementation of a novel dynamic (DPR-based) hardware system for a cascade SVM classifier on the recent hybrid Zynq SoC using the latest UltraFast HLS design methodology, targeting early detection of melanoma with high performance and low cost. Additionally, this novel dynamic hardware system fills the research gap by utilizing the powerful FPGA-based DPR technology. Significant optimization in implementation results was achieved from exploiting the DPR technique, while gaining flexibility, adaptability, scalability, and applicability. The implemented dynamic system on Zynq SoC is considered a novel low-power system in the literature, demonstrating extremely low resource utilization (1% slices) and power consumption (1.55 watts) than our previous implementations and existing implementations in the literature. Consequently, the proposed dynamic design meets important embedded system constraints, which promises realization of a cost- and energy-efficient handheld medical scanning device for early detection of melanoma at the primary healthcare.

5.4 Future Works

This study may lead to the following future research directions:

- The implemented embedded system of the SVM could be easily extended and adapted for different online classification applications with various sizes, targeting generality, scalability and applicability.
- The implemented binary SVM classifier could be extended to implement a multi-class classifier, in addition to implementing different types of kernels.
- More test instances would be tested in the future, in order to validate the classification accuracy rate of the implemented SVM classifiers on Zynq SoC, in addition to calculating sensitivity and specificity metrics.
- The proposed scalable IP-based hardware design implemented on Zynq SoC with the HLS design methodology could be adopted by hardware designers for implementing their embedded systems.

- Other hardware design and optimization methods/techniques (e.g. multiplier-less method, fixed-point arithmetic) could be applied and combined efficiently in the future towards more optimization in results as well as improvement of the online classification accuracy rate on hardware.
- The proposed scalable multi-core architecture (IP-based design) would be extended by adding more SVM IPs in a single device/SoC that could be applied in the future as a multi-class classifier, an ensemble classifier, or other scenarios.
- The implemented cascaded architecture can be extended to realize multi-stage cascade, by simply adding more SVM IPs, aiming to improve classification performance and diagnosis verification.
- The DPR feature could be employed to the proposed multi-core architecture to reconfigure each core (RP) in the architecture with multi RMs of various SVM copies or different classifiers, forming an adaptive and flexible system on a single SoC.
- Investigation of implementing the first three stages of the CAD system on Zynq SoC as well as investigation of integrating the implemented SVM classifier into the embedded full CAD system for realizing a low-cost handheld medical scanning device for melanoma detection or any other applications.

APPENDIX A Device Views of the DPR Design and Implementation Flow of the Proposed Dynamic Cascaded SVM Architecture

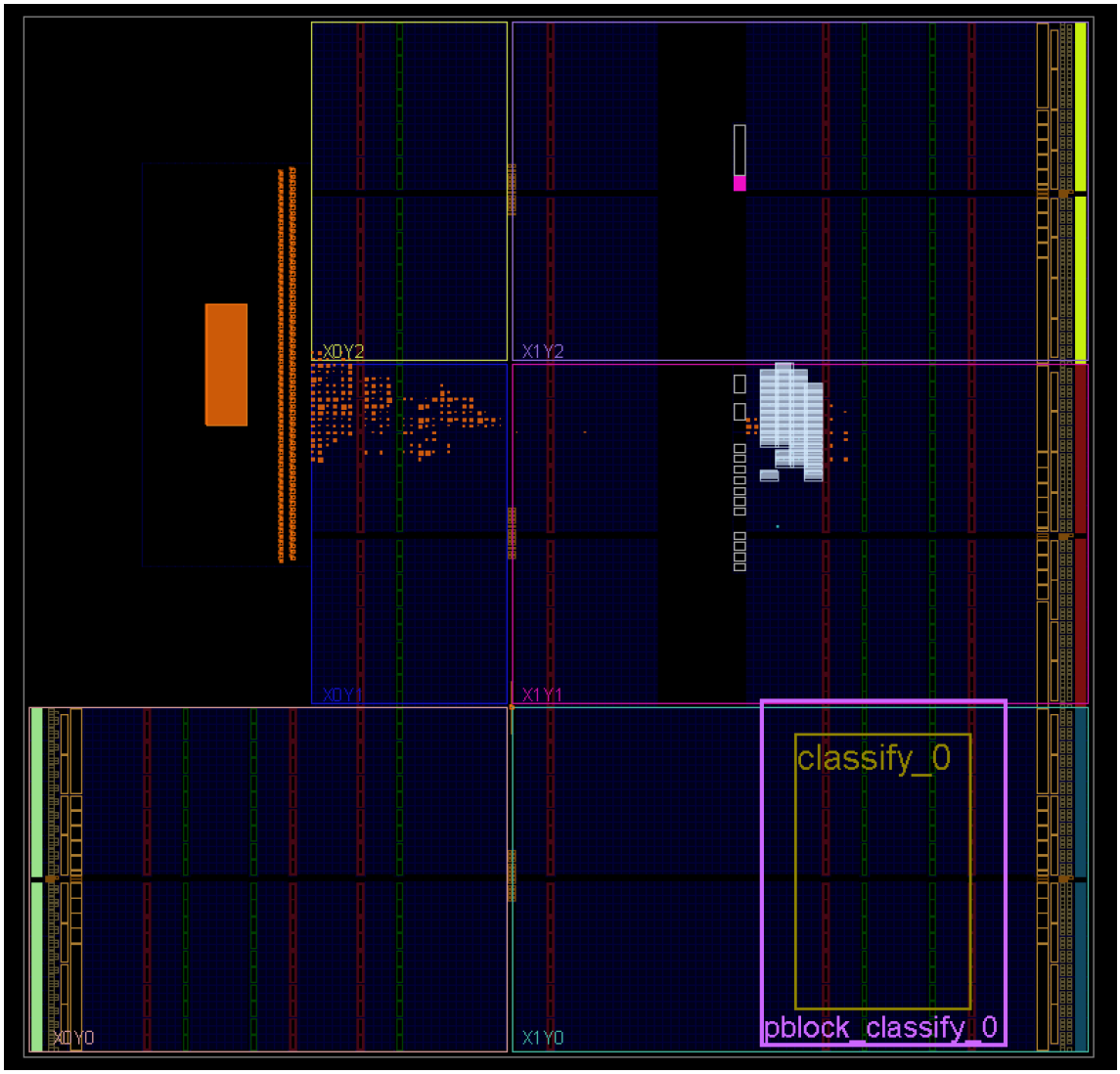

Figure A-1. Vivado device view of the first configuration that represents the static logic of the top-level design with BB module. The allocated resources are displayed in orange colour. The pblock "pblock_classify_0" is represented in a purple box at the bottom-right corner, which defines the RP as a BB in the static configuration (to be reconfigured for each RM of the two configurations). The grey boxes represent the ports of the RMs interfacing with the static logic. The static placement and routing are locked down to be used for the other configurations.

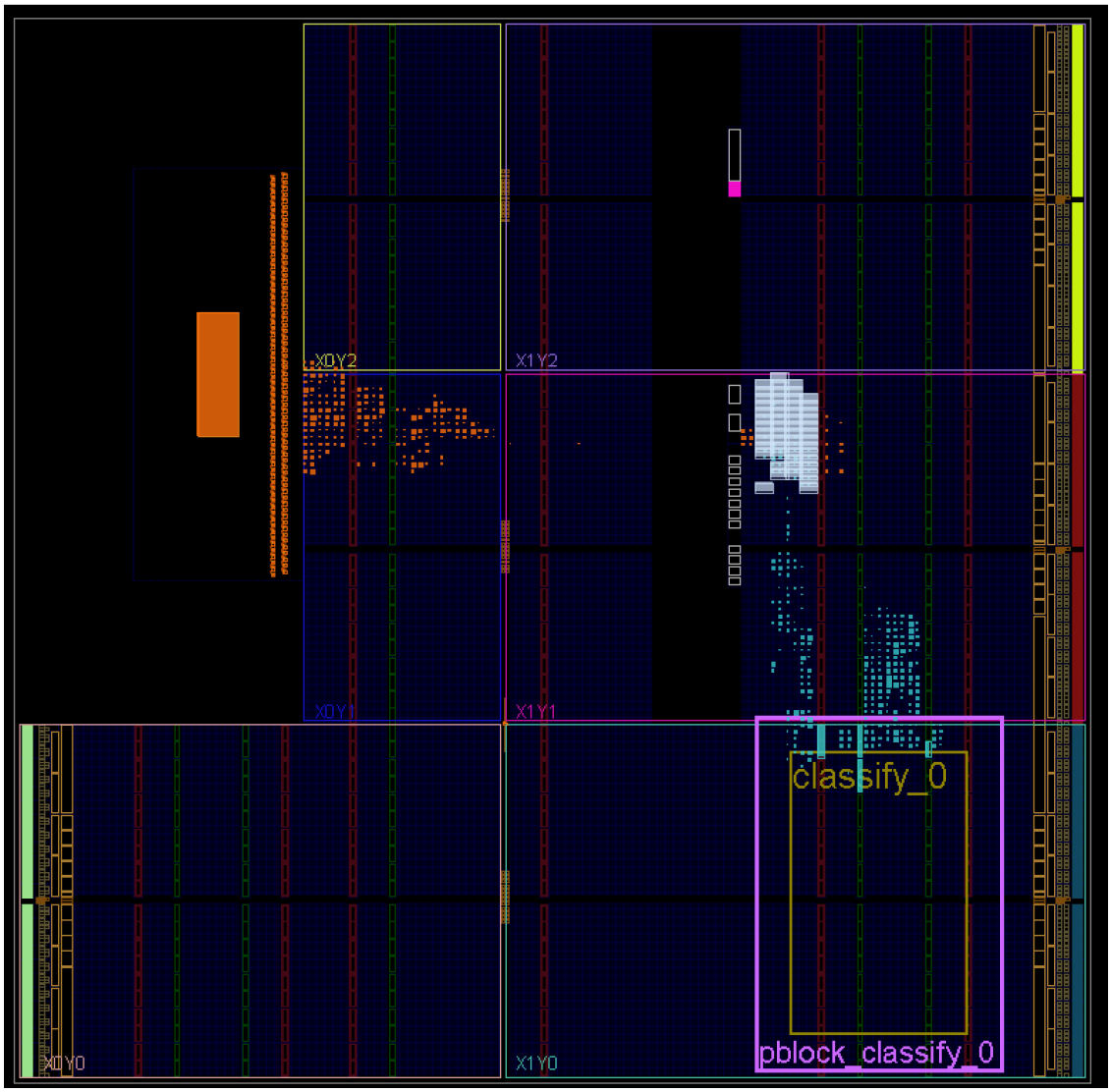

Figure A-2. Vivado device view of the RM-M configuration (for melanoma-sensitive SVM). The resource utilization is illustrated in cyan colour after placement and routing of the RM-M configuration.

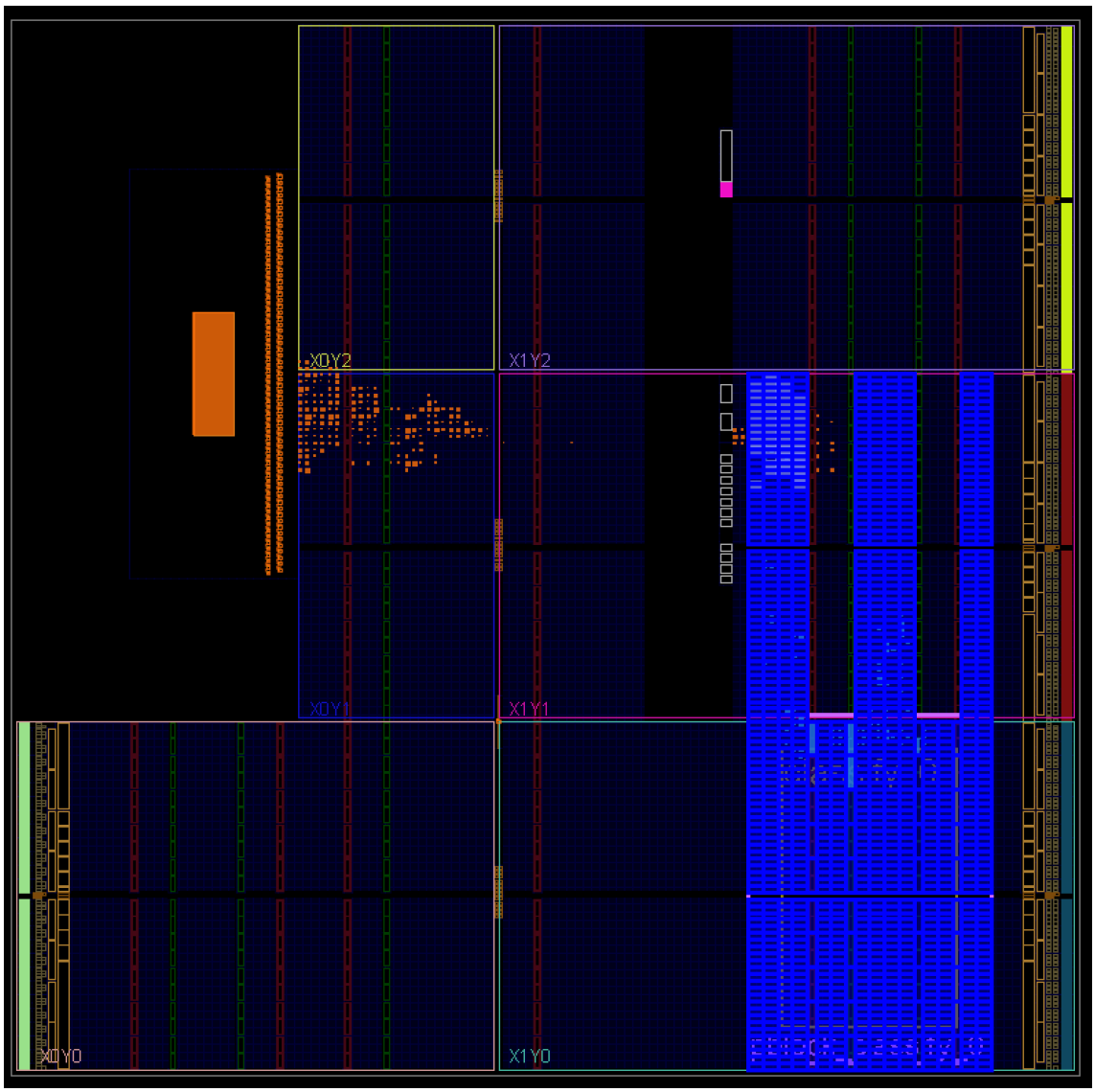

Figure A-3. The device view of the RM-M configuration, where reconfiguration partition frames are highlighted. The highlighted tiles are used in generating the partial bitstream of the RM-M.

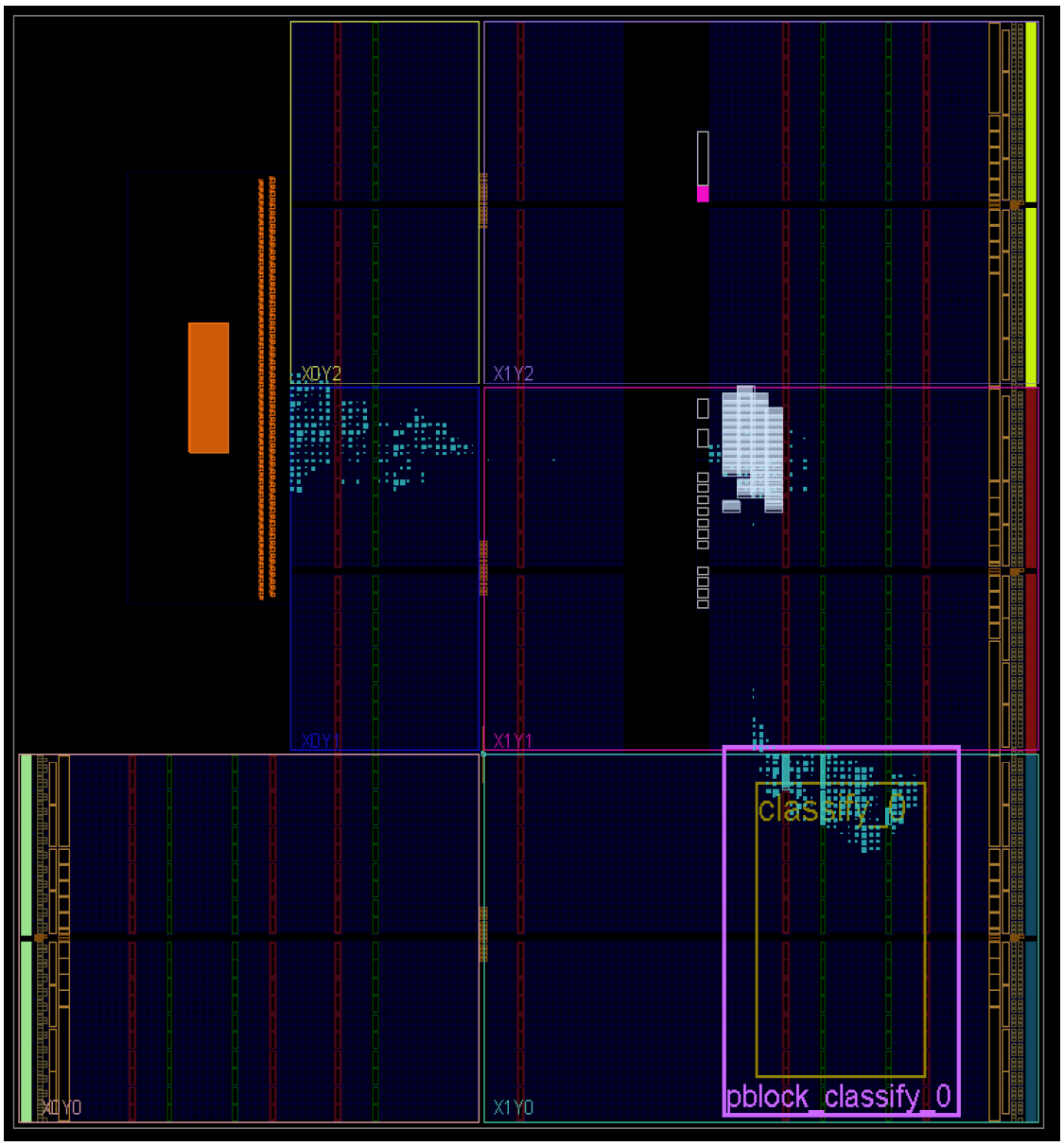

Figure A-4. Vivado device view of the RM-N configuration (for non-melanoma-sensitive SVM). The resource utilization is illustrated in cyan colour after placement and routing of the RM-N configuration.

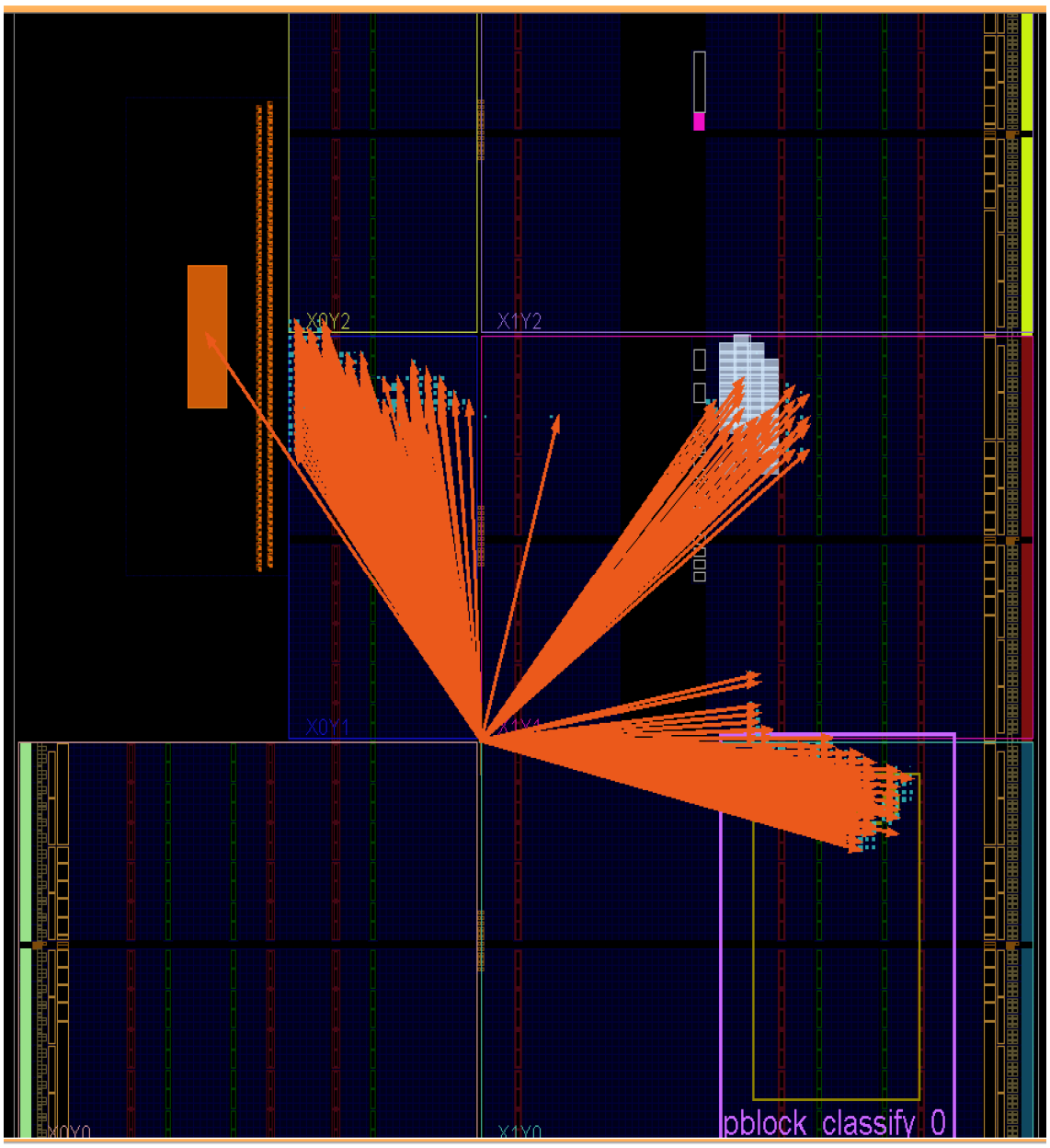

Figure A-5. Vivado device view of the RM-N configuration, where the routed nets of the static and RM-N configurations are connected in orange colour.

References

[1] S. B. Sathiya, S. S. Kumar, and A. Prabin, "A survey on recent computer-aided diagnosis of Melanoma," in *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, 2014, pp. 1387-1392.

[2] M. P. Véstias, "High-Performance Reconfigurable Computing Granularity," *Encyclopedia of Information Science and Technology,* pp. 3558-3567, 2015.

[3] T. Saegusa, T. Maruyama, and Y. Yamaguchi, "How Fast is an FPGA in Image Processing?," in *International Conference on Field Programmable Logic and Applications, FPL 2008* 2008, pp. 77-82.

[4] H. M. Hussain, K. Benkrid, and H. Seker, "The Role of FPGAs as High Performance Computing Solution to Bioinformatics and Computational Biology Data," *AIHLS2013,* p. 102, 2013.

[5] S. Asano, T. Maruyama, and Y. Yamaguchi, "Performance Comparison of FPGA, GPU and CPU in Image Processing," in *International Conference on Field Programmable Logic and Applications, 2009. FPL 2009*, 2009, pp. 126-131.

[6] J. Fowers, G. Brown, P. Cooke, and G. Stitt, "A Performance and Energy Comparison of FPGAs, GPUs, and Multicores for Sliding-Window Applications," in *Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays*, 2012, pp. 47-56.

[7] P. Sabouri, H. GholamHosseini, T. Larsson, and J. Collins, "A Cascade Classifier for Diagnosis of Melanoma in Clinical Images," in *36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2014, pp. 6748-6751.

[8] *Vivado High-Level Synthesis*. Available: http://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html

[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 (IJISET),* vol. 2, pp. 733-752, 2015.

[10] S. M. Afifi, H. GholamHosseini, and R. Sinha, "FPGA Implementations of SVM Classifiers: A Review," *Submitted in Neural Computing and Applications Journal,* 2018.

[11] S. Afifi, H. Gholamhosseini, and R. Sinha, "Hardware Acceleration of SVM-Based Classifier for Melanoma Images," presented at the Revised Selected Papers of the PSIVT 2015 Workshops on Image and Video Technology - Volume 9555, pp. 235-245, 2016.

[12] S. Afifi, H. GholamHosseini, and R. Sinha, "A Low-Cost FPGA-based SVM Classifier for Melanoma Detection," in *2016 IEEE EMBS Conference on Biomedical Engineering and Sciences (IECBES)*, 2016, pp. 631-636.

[13] S. Afifi, H. GholamHosseini, and R. Sinha, "SVM Classifier on Chip for Melanoma Detection " in *The 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC'17)*, 2017.

[14] D. H. Hoe, L. Bollepalli, and C. D. Martinez, "FPGA Fault Tolerant Arithmetic Logic: A Case Study Using Parallel-prefix Adders," *VLSI Design,* vol. 2013, p. 17, 2013.

[15] M. Wielgosz, E. Jamro, D. Zurek, and K. Wiatr, "FPGA Implementation of The Selected Parts of The Fast Image Segmentation," in *Studies in Computational Intelligence* vol. 390, ed, 2012, pp. 203-216.

[16] K. Nagarajan, B. Holland, A. D. George, K. C. Slatton, and H. Lam, "Accelerating Machine-Learning Algorithms on FPGAs using Pattern-Based Decomposition," *Journal of Signal Processing Systems,* vol. 62, pp. 43-63, 2011.

[17] L. Woods, J. Teubner, and G. Alonso, "Real-Time Pattern Matching with FPGAs," in *2011 IEEE 27th International Conference on Data Engineering (ICDE)*, 2011, pp. 1292-1295.

[18] A. Eklund, P. Dufort, D. Forsberg, and S. M. LaConte, "Medical Image Processing on The GPU–Past, Present and Future," *Medical Image Analysis,* vol. 17, pp. 1073-1094, 2013.

[19] M. Papadonikolakis, C.-S. Bouganis, and G. Constantinides, "Performance Comparison of GPU and FPGA Architectures for The SVM Training Problem," in *International Conference on Field-Programmable Technology, 2009 - FPT 2009*, 2009, pp. 388-391.

[20] B. Cope, P. Y. Cheung, W. Luk, and L. Howes, "Performance Comparison of Graphics Processors to Reconfigurable Logic: A Case Study," *IEEE Transactions on Computers,* vol. 59, pp. 433-448, 2010.

[21] M. Pietron, M. Wielgosz, D. Zurek, E. Jamro, and K. Wiatr, "Comparison of GPU And FPGA Implementation of SVM Algorithm for Fast Image Segmentation," in *Architecture of Computing Systems–ARCS 2013*, ed: Springer, 2013, pp. 292-302.

[22] S. Che, J. Li, J. W. Sheaffer, K. Skadron, and J. Lach, "Accelerating Compute-Intensive Applications with GPUs and FPGAs," in *Symposium on Application Specific Processors, 2008-SASP 2008*, 2008, pp. 101-107.

[23] E. Fykse, "Performance Comparison of GPU, DSP and FPGA Implementations of Image Processing and Computer Vision Algorithms in Embedded Systems," M.Sc. thesis, Department of Electronics and Telecommunications, Norwegian University of Science and Technology, 2013.

[24] A. Maghazeh, U. D. Bordoloi, P. Eles, and Z. Peng, "General Purpose Computing on Low-Power Embedded GPUs: Has It Come of Age?," in *2013 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIII)*, 2013, pp. 1-10.

[25] *Zynq-7000 All Programmable SoC*. Available: http://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html

[26] T. N. Sasamal and R. Prasad, "Module Based And Difference Based Implementation of Partial Reconfiguration on FPGA: A Review," *International Journal of Engineering Research and Applications (IJERA),* vol. 1, pp. 1898-1903, 2011.

[27] 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.

[28] G. M. Foody and A. Mathur, "A Relative Evaluation of Multiclass Image Classification by Support Vector Machines," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 42, pp. 1335-1343, 2004.

[29] R. Entezari-Maleki, A. Rezaei, and B. Minaei-Bidgoli, "Comparison of Classification Methods Based on The Type of Attributes and Sample Size," *Journal of Convergence Information Technology,* vol. 4, pp. 94-102, 2009.

[30] J. KIM[1], B.-S. Kim, and S. Savarese, "Comparing Image Classification Methods: K-Nearest-Neighbor and Support-Vector-Machines," *Ann Arbor,* vol. 1001, pp. 48109-2122, 2012.

[31] C. Cortes and V. Vapnik, "Support-Vector Networks," *Machine Learning,* vol. 20, pp. 273-297, 1995.

[32] V. N. Vapnik, "An Overview of Statistical Learning Theory," *IEEE Transactions on Neural Networks,* vol. 10, pp. 988-999, 1999.

[33] J. Platt, "Fast Training of Support Vector Machines Using Sequential Minimal Optimization," *Advances in Kernel Methods—Support Vector Learning,* vol. 3, 1999.

[34] C. J. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery,* vol. 2, pp. 121-167, 1998.

[35] C.-W. Hsu and C.-J. Lin, "A Comparison of Methods for Multiclass Support Vector Machines," *IEEE Transactions on Neural Networks,* vol. 13, pp. 415-425, 2002.

[36] K. Ta-Wen, W. Jhing-Fa, W. Jia-Ching, L. Po-Chuan, and G. Gaung-Hui, "VLSI Design of an SVM Learning Core on Sequential Minimal Optimization Algorithm," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 20, pp. 673-683, 2012.

[37] K.-k. Cao, H.-b. Shen, and H.-f. Chen, "A Parallel and Scalable Digital Architecture for Training Support Vector Machines," *Journal of Zhejiang University SCIENCE C,* vol. 11, pp. 620-628, 2010.

[38] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy, "Improvements to Platt's SMO Algorithm for SVM Classifier Design," *Neural Computation,* vol. 13, pp. 637-649, 2001.

[39] J. G. Filho, M. Raffo, M. Strum, and W. J. Chau, "A General-Purpose Dynamically Reconfigurable SVM," in *2010 VI Southern Programmable Logic Conference (SPL)*, 2010, pp. 107-112.

[40] D. Anguita, S. Pischiutta, S. Ridella, and D. Sterpi, "Feed-Forward Support Vector Machine without Multipliers," *IEEE Transactions on Neural Networks,* vol. 17, pp. 1328-1331, 2006.

[41] R. Andraka, "A Survey of CORDIC Algorithms for FPGA Based Computers," in *Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays*, 1998, pp. 191-200.

[42] C. H. Peng, B. W. Chen, T. W. Kuan, P. C. Lin, J. F. Wang, and N. S. Shih, "REC-STA: Reconfigurable and Efficient Chip Design With SMO-based Training Accelerator," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 22, pp. 1791-1802, 2014.

[43] L. Bustio-Martínez, R. Cumplido, J. Hernández-Palancar, and C. Feregrino-Uribe, "On the Design of a Hardware-Software Architecture for Acceleration of SVM's Training Phase," in *Advances in Pattern Recognition*, ed: Springer, 2010, pp. 281-290.

[44] W. Jhing-Fa, P. Jr-Shiang, W. Jia-Ching, L. Po-Chuan, and K. Ta-Wen, "Hardware/Software Co-design for Fast-trainable Speaker Identification System Based on SMO," in *2011 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2011, pp. 1621-1625.

[45] A. Patel, V. Sriram, and K. Varghese, "Hardware Accelerator for Support Vector Machine Training," *Training,* vol. 1, p. 2.

[46] S. Venkateshan, A. Patel, and K. Varghese, "Hybrid Working Set Algorithm for SVM Learning With a Kernel Coprocessor on FPGA," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* 2014.

[47] T. Joachims, "Advances in kernel methods," *chapter Making large-scale support vector machine learning practical,* pp. 169-184, 1999.

[48] M. Papadonikolakis and C. S. Bouganis, "A Heterogeneous FPGA Architecture for Support Vector Machine Training," in *Proceedings - IEEE Symposium on Field-Programmable Custom Computing Machines, FCCM 2010*, 2010, pp. 211-214.

[49] M. B. Rabieah and C.-S. Bouganis, "FPGA Based Nonlinear Support Vector Machine Training Using an Ensemble Learning," in *2015 25th International Conference on Field Programmable Logic and Applications (FPL)*, 2015, pp. 1-4.

[50] S. Martin, "Training Support Vector Machines Using Gilbert's Algorithm," in *Fifth IEEE International Conference on Data Mining*, 2005, p. 8 pp.

[51] M. Ning, W. Shaojun, P. Yeyong, and P. Yu, "Implementation of LS-SVM with HLS on Zynq," in *2014 International Conference on Field-Programmable Technology (FPT)*, 2014, pp. 346-349.

[52] L. Wei, Z. Chen, J. Li, and W. Xu, "Sparse and robust least squares support vector machine: a linear programming formulation," in *Grey Systems and Intelligent Services, 2007. GSIS 2007. IEEE International Conference on*, 2007, pp. 1134-1138.

[53] W. Shaojun, P. Yu, Z. Guangquan, and P. Xiyuan, "Accelerating On-Line Training of LS-SVM With Run-Time Reconfiguration," in *2011 International Conference on Field-Programmable Technology (FPT)*, 2011, pp. 1-6.

[54] Y. Ago, K. Nakano, and Y. Ito, "A Classification Processor for a Support Vector Machine with Embedded DSP Slices and Block RAMs in the FPGA," in *2013 IEEE 7th International Symposium on Embedded Multicore Socs (MCSoC)*, 2013, pp. 91-96.

[55] M. Berberich and K. Doll, "Highly Flexible FPGA-Architecture of a Support Vector Machine," in *MPC-Workshop 45*, 2014, pp. 25-32.

[56] Z. Nie, X. Zhang, and Z. Yang, "An FPGA Implementation of Multi-Class Support Vector Machine Classifier Based on Posterior Probability," in *Proceedings of 2010 3rd International Conference on Computer and Electrical Engineering (ICCEE 2010 no. 2)*, 2010.

[57] S. Saurav, S. Singh, R. Saini, and A. K. Saini, "Hardware Accelerator for Facial Expression Classification Using Linear SVM," in *SIRS*, 2015, pp. 39-50.

[58] V. S. Vranjković, R. J. R. Struharik, and L. A. Novak, "Reconfigurable Hardware for Machine Learning Applications," *Journal of Circuits, Systems and Computers,* vol. 24, 2015.

[59] V. Vranjković and R. Struharik, "Coarse-grained Reconfigurable Hardware Accelerator of Machine Learning Classifiers," in *2016 International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2016, pp. 1-5.

[60] S. Kim, S. Lee, and K. Cho, "Design of High-Performance Unified Circuit for Linear and Non-Linear SVM Classifications," *Journal of Semiconductor Technology and Science,* vol. 12, pp. 162-167, 2012.

[61] P.-T. P. Tang, "Table-driven implementation of the exponential function in IEEE floating-point arithmetic," *ACM Transactions on Mathematical Software (TOMS),* vol. 15, pp. 144-157, 1989.

[62] T. Koide, H. Anh-Tuan, T. Okamoto, S. Shigemi, T. Mishima, T. Tamaki*, et al.*, "FPGA Implementation Of Type Identifier For Colorectal Endoscopie Images With NBI Magnification," in *2014 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, 2014, pp. 651-654.

[63] S. Shigemi, T. Mishima, A.-T. Hoang, T. Koide, T. Tamaki, B. Raytchev*, et al.*, "Customizable Hardware Architecture of Support Vector Machine in CAD System for Colorectal Endoscopic Images with NBI Magnification," *SASIMI 2013 Proceedings, The 18th Workshop on Synthesis And System Integration of Mixed Information Technologies,* pp. 298-203, 2013.

[64] S. Shigemi, "An FPGA Implementation of Support Vector Machine Identifier for Colorectal Endoscopic Images with NBI Magnification," in *Proc. of the 28th International Conference on Circuits/Systems, Computers and Communications (ITC-CSCC2013)*, pp. 571-572.

[65] C. Liu, F. Qiao, X. Yang, and H. Yang, "Hardware Acceleration With Pipelined Adder For Support Vector Machine Classifier," in *2014 Fourth International Conference on Digital Information and Communication Technology and it's Applications (DICTAP)*, 2014, pp. 13-16.

[66] J. Cárdenas, M. Figueroa, and J. E. Pezoa, "A Custom Hardware Classifier for Bruised Apple Detection in Hyperspectral Images," in *SPIE Optical Engineering+ Applications*, 2015, pp. 95992K-95992K-11.

[67] M. Qasaimeh, A. Sagahyroon, and T. Shanableh, "FPGA-Based Parallel Hardware Architecture for Real-Time Image Classification," *IEEE Transactions on Computational Imaging,* vol. 1, pp. 56-70, 2015.

[68] M. Vucha and A. Rajawat, "Design and FPGA implementation of systolic array architecture for matrix multiplication," *International Journal of Computer Applications,* vol. 26, pp. 18-22, 2011.

[69] C. 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.

[70] C. Kyrkou and T. Theocharides, "SCoPE: Towards a Systolic Array for SVM Object Detection," *IEEE Embedded Systems Letters,* vol. 1, pp. 46-49, 2009.

[71] H. M. Hussain, K. Benkrid, and H. Seker, "Reconfiguration-Based Implementation of SVM Classifier on FPGA for Classifying Microarray Data," in *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2013, pp. 3058-3061.

[72] H. Hussain, K. Benkrid, and H. Şeker, "Novel Dynamic Partial Reconfiguration Implementations of The Support Vector Machine Classifier on FPGA," *Turkish Journal of Electrical Engineering and Computer Sciences,* vol. 24, pp. 3371-3387, 2016.

[73] H. M. Hussain, K. Benkrid, and H. Seker, "Dynamic Partial Reconfiguration Implementation of the SVM/KNN Multi-Classifier on FPGA for Bioinformatics Application," in *37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2015, pp. 7667-7670.

[74] R. Patil, G. Gupta, V. Sahula, and A. Mandal, "Power Aware Hardware Prototyping of Multiclass SVM Classifier Through Reconfiguration," in *2012 25th International Conference on VLSI Design (VLSID)*, 2012, pp. 62-67.

[75] H. M. Hussain, K. Benkrid, and H. Seker, "An Adaptive Implementation of A Dynamically Reconfigurable K-Nearest Neighbour Classifier on FPGA," in *NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, 2012, pp. 205-212.

[76] H. Hussain, K. Benkrid, C. Hong, and H. Seker, "An Adaptive FPGA Implementation of Multi-Core K-Nearest Neighbour Ensemble Classifier using Dynamic Partial Reconfiguration," in *22nd International Conference on Field Programmable Logic and Applications (FPL)*, 2012, pp. 627-630.

[77] A. H. M. Jallad and L. B. Mohammed, "Hardware Support Vector Machine (SVM) for Satellite on-Board Applications," in *2014 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, 2014, pp. 256-261.

[78] S. Wong, S. Vassiliadis, and S. Cotofana, "A sum of absolute differences implementation in FPGA hardware," in *Euromicro Conference, 2002. Proceedings. 28th*, 2002, pp. 183-188.

[79] X. Pan, H. Yang, L. Li, Z. Liu, and L. Hou, "FPGA Implementation of SVM Decision Function Based on Hardware-friendly Kernel," in *2013 International Conference on Computational and Information Sciences , ICCIS 2013 Proceedings*, 2013, pp. 133-136.

[80] M. Ruiz-Llata, G. Guarnizo, and M. Yébenes-Calvino, "FPGA Implementation of a Support Vector Machine for Classification and Regression," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, 2010, pp. 1-5.

[81] J. Gimeno Sarciada, H. Lamel Rivera, and M. Jiménez, "CORDIC Algorithms for SVM FPGA Implementation," in *Proceedings of SPIE - The International Society for Optical Engineering*, 2010.

[82] H. Lamela, J. Gimeno, M. Jiménez, and M. Ruiz, "Performance Evaluation of a FPGA Implementation of a Digital Rotation Support Vector Machine," in *SPIE Defense and Security Symposium*, 2008, pp. 697908-697908-8.

[83] D. Anguita, L. Carlino, A. Ghio, and S. Ridella, "A FPGA Core Generator for Embedded Classification Systems," *Journal of Circuits, Systems and Computers,* vol. 20, pp. 263-282, 2011.

[84] D. Anguita, A. Ghio, S. Pischiutta, and S. Ridella, "A support vector machine with integer parameters," *Neurocomputing,* vol. 72, pp. 480-489, 2008.

[85] V. Vranjkovic and R. Struharik, "New Architecture for SVM Classifier and Its Application To Telecommunication Problems," in *2011 19th Telecommunications Forum (TELFOR)*, 2011, pp. 1543-1545.

[86] B. Mandal, M. P. Sarma, and K. K. Sarma, "Implementation of Systolic Array Based SVM Classifier Using Multiplierless Kernel," in *2014 International Conference on Signal Processing and Integrated Networks (SPIN)*, pp. 35-39.

[87] B. Mandal, M. P. Sarma, and K. K. Sarma, "Design Of A Systolic Array Based Multiplierless Support Vector Machine Classifier," in *2014 International Conference on Signal Processing and Integrated Networks (SPIN)*, 2014, pp. 35-39.

[88] *Xilinx System Generator for DSP*. Available: https://au.mathworks.com/products/connections/product_detail/product_35567.html

[89] D. Mahmoodi, A. Soleimani, H. Khosravi, and M. Taghizadeh, "FPGA Simulation of Linear and Nonlinear Support Vector Machine," *Journal of Software Engineering and Applications,* vol. 4, pp. 320-328, 2011.

[90] Y. Jiang, K. Virupakshappa, and E. Oruklu, "FPGA Implementation of a Support Vector Machine Classifier for Ultrasonic Flaw Detection," in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2017, pp. 180-183.

[91] M. Cutajar, E. Gatt, I. Grech, O. Casha, and J. Micallef, "Hardware-based Support Vector Machine for Phoneme Classification," in *IEEE EuroCon 2013*, 2013, pp. 1701-1708.

[92] V. Tsoutsouras, K. Koliogeorgi, S. Xydis, and D. Soudris, "An Exploration Framework for Efficient High-Level Synthesis of Support Vector Machines: Case Study on ECG Arrhythmia Detection for Xilinx Zynq SoC," *Journal of Signal Processing Systems,* pp. 1-21, 2017.

[93] C. Kyrkou, T. Theocharides, and C.-S. Bouganis, "An Embedded Hardware-Efficient Architecture for Real-Time Cascade Support Vector Machine Classification," in *2013 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIII)*, 2013, pp. 129-136.

[94] M. Papadonikolakis and C.-S. Bouganis, "A Novel FPGA-based SVM Classifier," in *International Conference on Field-Programmable Technology (FPT)* 2010, pp. 283-286.

[95] M. Papadonikolakis and C. Bouganis, "Novel Cascade FPGA Accelerator for Support Vector Machines Classification," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 23, pp. 1040-1052, 2012.

[96] C. Kyrkou, T. Theocharides, and C. S. Bouganis, "A Hardware-Efficient Architecture for Embedded Real-Time Cascaded Support Vector Machines Classification," in *Proceedings of the 23rd ACM international conference on Great lakes symposium on VLSI*, 2013, pp. 341-342.

[97] C. Kyrkou, T. Theocharides, C.-S. Bouganis, and M. Polycarpou, "Boosting the Hardware-Efficiency of Cascade Support Vector Machines for Embedded Classification Applications," *International Journal of Parallel Programming,* pp. 1-27, 2017.

[98] C. Kyrkou, C.-S. Bouganis, T. Theocharides, and M. M. Polycarpou, "Embedded Hardware-Efficient Real-Time Classification with Cascade Support Vector Machines," *IEEE Transactions on Neural Networks and Learning Systems,* 2015.

[99] M. Hahnle, F. Saxen, M. Hisung, U. Brunsmann, and K. Doll, "FPGA-Based Real-Time Pedestrian Detection on High-Resolution Images," in *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2013, pp. 629-635.

[100] X. Shuai, L. Yibin, J. Zhiping, and J. Lei, "Binarization Based Implementation for Real-Time Human Detection," in *2013 23rd International Conference on Field Programmable Logic and Applications (FPL)*, 2013, pp. 1-4.

[101] C. Kelly, F. M. Siddiqui, B. Bardak, and R. Woods, "Histogram Of Oriented Gradients Front End Processing: An FPGA Based Processor Approach," in *2014 IEEE Workshop on Signal Processing Systems (SiPS)*, 2014, pp. 1-6.

[102] F. M. Siddiqui, M. Russell, B. Bardak, R. Woods, and K. Rafferty, "IPPro: FPGA based image processing processor," in *Signal Processing Systems (SiPS), 2014 IEEE Workshop on*, 2014, pp. 1-6.

[103] K. Mizuno, Y. Terachi, K. Takagi, S. Izumi, H. Kawaguchi, and M. Yoshimoto, "Architectural Study of HOG Feature Extraction Processor for Real-Time Object

Detection," in *2012 IEEE Workshop on Signal Processing Systems (SiPS)*, 2012, pp. 197-202.

[104] K. Mizuno, Y. Terachi, K. Takagi, S. Izumi, H. Kawaguchi, and M. Yoshimoto, "An FPGA Implementation of a HOG-based Object Detection Processor," *IPSJ Transactions on System LSI Design Methodology,* vol. 6, pp. 42-51, 2013.

[105] M. Komorkiewicz, M. Kluczewski, and M. Gorgon, "Floating Point HOG Implementation for Real-Time Multiple Object Detection," in *2012 22nd International Conference on Field Programmable Logic and Applications (FPL)*, 2012, pp. 711-714.

[106] T. Kryjak, M. Komorkiewicz, and M. Gorgon, "FPGA Implementation of Real-Time Head-Shoulder Detection Using Local Binary Patterns, SVM and Foreground Object Detection," in *2012 Conference on Design and Architectures for Signal and Image Processing (DASIP)*, 2012, pp. 1-8.

[107] M. Vergara, A. Wolf, and M. Figueroa, "A Texture-based Architecture for Face Detection in IR Images on an FPGA," in *Proceedings of SPIE - The International Society for Optical Engineering*, 2014.

[108] S. Martelli, D. Tosato, M. Cristani, and V. Murino, "FPGA-based Pedestrian Detection Using Array of Covariance Features," in *2011 5th ACM/IEEE International Conference on Distributed Smart Cameras, ICDSC 2011*, 2011.

[109] C. Kyrkou, C. Ttofis, and T. Theocharides, "A Hardware Architecture for Real-Time Object Detection Using Depth and Edge Information," *Transactions on Embedded Computing Systems,* vol. 13, 2013.

[110] S. S Dhar and K. Sreeraj, "FPGA Implementation of Feature Extraction Based on Histopathalogical Image and Subsequent Classification by Support Vector Machine," *IJISET-International Journal of Innovative Science, Engineering & Technology,* vol. 2, pp. 744-749, 2015.

[111] H. Hussain, K. Benkrid, and H. Seker, "Novel Dynamic Partial Reconfiguration Implementations of The Support Vector Machine Classifier on FPGA," *Turkish Journal of Electrical Engineering and Computer Sciences,* 2014.

[112] S. Xie, Y. Li, Z. Jia, and L. Ju, "Binarization-based Human Detection for Compact FPGA Implementation," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* vol. 8299 LNCS, ed, 2013, pp. 119-131.

[113] D. Anguita, S. Pischiutta, S. Ridella, and D. Sterpi, "Feed-Forward Support Vector Machine Without Multipliers," *IEEE Transactions on Neural Networks,,* vol. 17, pp. 1328-1331, 2006.

[114] *Vivado Design Suite*. Available: http://www.xilinx.com/products/design-tools/vivado.html

[115] *Vivado Design Suite User guide, High-Level Synthesis*. Available: http://www.xilinx.com/support/documentation/sw_manuals/xilinx2014_1/ug902-vivado-high-level-synthesis.pdf

[116] *Zynq-7000 All Programmable SoC Acccelerator for floating-Point Matrix Multiplication using Vivado HLS*. Available: http://www.xilinx.com/support/documentation/application_notes/xapp1170-zynq-hls.pdf

[117] *Partial Reconfiguration of a Hardware Accelerator with Vivado Design Suite for Zynq-7000 AP SoC Processor*. Available: https://www.xilinx.com/support/documentation/application_notes/xapp1231-partial-reconfig-hw-accelerator-vivado.pdf

[118] T. Joachims, *Making large-Scale SVM Learning Practical. Advances In Kernel Methods: Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (ed.)*: MIT Press, 1999.

[119] C. Kyrkou, C. Bouganis, T. Theocharides, and M. M. Polycarpou, "Embedded Hardware-Efficient Real-Time Classification With Cascade Support Vector Machines," *IEEE Transactions on Neural Networks and Learning Systems,* 2015.