UNIVERSITÀ DI PARMA

UNIVERSITÀ DEGLI STUDI DI PARMA

DOTTORATO DI RICERCA IN
"TECNOLOGIE DELL'INFORMAZIONE"

CICLO XXXIII

Advanced FPGA-Based Systems Design Techniques for Wearable Sensor Applications

Coordinatore:
Chiar.mo Prof. Marco Locatelli

Relatore:
Chiar.ma Prof.ssa Ilaria De Munari
Tutor:
Chiar.ma Prof.ssa Valentina Bianchi

Dottorando: Marco Bassoli

Anni Accademici 2017/2018 - 2019/2020

Dedicated to
My Family

Contents

List of Figures

List of Tables

Introduction

Among the trending topics of the last years, the *Internet of Things* (IoT) is one of those gathering the most attention [1]. In its golden-rule implementation, this paradigm pursues the ubiquitous exchange of information between every kind of object, even those we nowadays perceive as the most irrelevant. Within this vision, the amount of data collected and produced by our actions should allow for building a better society, in which new services can be introduced to foster tailored healthcare and increasing well-being [2].

Although the final implementation of this conception may be perceived as utopian by most, we are slowly getting in touch with it in our daily lives. In recent years, the *Smart Home* concept has been extended from the simple automation and automatic control of the home appliances to more complex management of the user interaction with several sensors and actuators deployed in the home environment in order to pursue the users' well-being and energy sustainability [3, 4, 5, 6, 7, 8, 9, 10, 11]. Besides, the amount of data produced by users inside their own home environment can enable useful applications in health management, such as the early detection of behavioral trends and anomalies otherwise impossible to identify [12, 13, 14, 15, 9, 16, 17].

This new scenario requires a new generation of devices to be developed [18], aiming at gathering relevant information on how and when the user interacts with the home environment. One of the best ways to accomplish this task is to exploit both *environmental* (e.g. armchair sensors to monitor inactivity periods, toilet proximity sensors, etc.) and *wearable* (e.g. a device to be worn at the waist) sensors. In addition,

if those devices are able to send the collected data over the Internet, data can be processed and accessed by the various professionals involved (e.g. medical doctors, caregivers, relatives, and the users themselves) to give the best user-centered service.

At the D2Lab of the University of Parma, a new home monitoring system has been developed to enable the behavioral analysis of the users inside their homes. Its working principle relies on a set of battery-powered sensors connected to an Internet cloud service (Figure 1). The sensors are responsible for gathering information about the user's actions inside the home and, at the end of the day, for sending it on the cloud for the analytics. It is important to notice that the system is not conceived for real-time reporting, but instead, it aims at providing a set of daily, weekly or monthly behavioral analyses to the user (or to who is in charge for her/him). This is required to highlight variations on user's macro- and long-term patterns (e.g. a decrease in mobility), and to act with specific prevention programs.

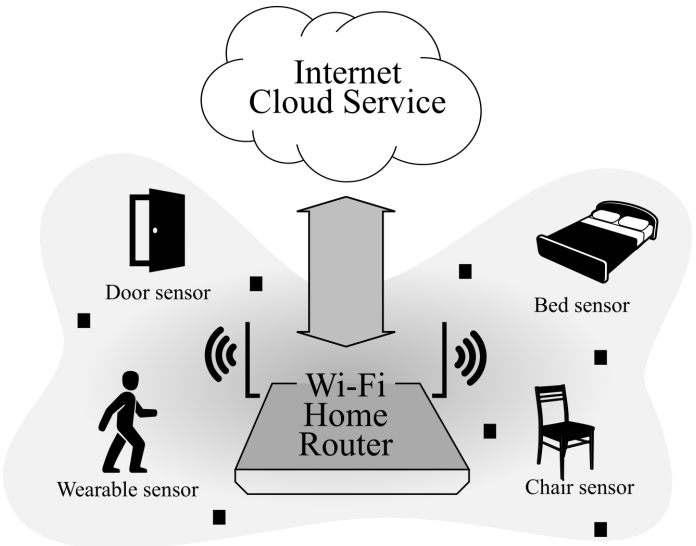


Figure 1: Sensors system for behavioral analysis.

Differently from traditional systems featuring low-power, low-cost wireless transmission protocols (e.g. Zigbee [19, 20, 21]), this system relies on Wi-Fi connectivity. If on the one hand the low-power protocols simplify the energy management of the

IoT nodes, on the other hand they require completely new and dedicated infrastructure in each home in which the system is to be deployed, since, typically, a network compliant with these standards is not already there. Moreover, these low-power protocols feature a short area coverage and require a set of repeaters and range extenders to operate across multiple house rooms. These problems can be addressed by using the Wi-Fi protocol, which features a wider area coverage and whose deployment is simplified by exploiting a standard Wi-Fi home router (quite often already present in the homes) for the Internet connection. The result is a more scalable and less expensive final implementation.

This, of course, comes at the expense of increased power consumption, which may harm the sensors' battery lifetime. For this reason, dedicated sensor platforms have been designed at the D2Lab for low-power operation even on Wi-Fi connectivity. An extensive account for the exploited design methodologies is given in [22], where it is shown that the obtained battery lifetimes are suitable for practical exploitation.

Based on such studies, a complete set of specific environmental sensors have been developed, providing expressive information relevant to behavioral inference [3]: a magnetic contact, to sense the opening/closing of drawers or doors; an armchair/bed pressure sensor to detect the presence of the user on chairs or beds; a Passive InfraRed (PIR) sensor to reveal the presence of a person inside a room, calibrated to exclude small-size pets and animals (e.g. cats or small dogs); a toilet proximity sensor, to catch the user's interactions with the toilet service. The data collected by the sensors are sent to an IoT cloud platform for processing and visualization.

Moreover, the capabilities of the system have been extended with the introduction of a wearable sensor for Human Activity Recognition (HAR) to enrich the behavioral analysis with activities difficult or impossible to detect with environmental sensors [23, 24]. The design of the sensor required the same energy-aware techniques to be used, with particular care to the on-off cycles of the Wi-Fi radio. Typically, HAR algorithms rely on high sample rates of the motion sensors (e.g. 50Hz [25]), whose data need to be processed to extract the activity of the user. In this specific case, if all the computation part was on the cloud, the Wi-Fi radio would be continuously

uploading the stream of data, negatively affecting the battery lifetime. For this reason, provided that the energy required for computational activity is much less than that dissipated by the radio, a part of the computation needs to be moved on the wearable device to send on the cloud just the final activity of the user. In particular, a machine learning algorithm has been selected and studied for this application.

This is the context of this thesis, which is focused on the study and further development of the aforementioned behavioral analysis ecosystem. The aim is to build a system capable of addressing the main trade-offs of the state-of-the-art, namely behavioral analysis complexity and battery power efficiency. To reach this goal, a performance assessment and optimization of the environmental sensors have been carried out, and the design and development of the wearable sensor have been performed.

After the first part of related works (chapter 1) and background (Chapter 2), a work on the set of environmental sensors has been carried out in Chapter 3 for performance assessment. Then, in Chapter 4, the design of the wearable sensor for HAR is presented, which led to the study of the algorithm and the hardware for the application.

Chapter 1

Background

This Chapter is a collection of the theoretical notions this thesis relies on. The concepts described here will then be exploited for the experimental parts of this work.

1.1 Support Vector Machine Algorithms

A Support Vector Machine (SVM) is a machine learning algorithm that, in its simplest form, can solve a binary classification problem using two or more input values called *features* [26]. For example, let us imagine having to distinguish if an object is "an apple" or "not an apple" based on the weight, height, width, and color of the object. In this case, we can plan to use this set of information as the *features* (i.e. the input) of an SVM algorithm to obtain at the output the *classification* of the object in the "apple" or "not apple" class.

To formalize this concept, let us take the case of two generic input features x_1 and x_2 of Figure 1.1a (i.e. the *features space*). The aim of the algorithm is to provide an output label $\hat{y} = +1$ or $\hat{y} = -1$ whether the current input features vector x is of the same set of the black or the white dots, respectively (Figure 1.1b). In other words, it tries to *classify* the new input x relying on the information already known by the algorithm, i.e. the black and white dots provided in the *training* phase.

To do so, the algorithm makes use of a threshold line as shown in Figure 1.1c.

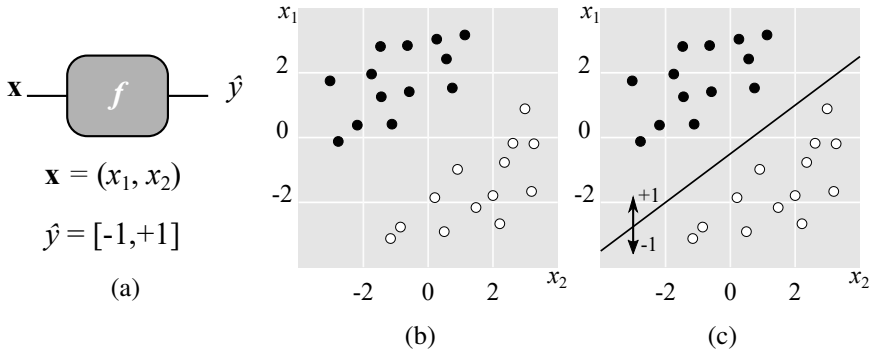


Figure 1.1: Support Vector Machine linear example: (a) input-output function, (b) features space, (c) features space with threshold line.

If the features input vector $\mathbf{x} = [x_1, x_2]$ is above the line in the features space, it is classified as +1, if it is below the line, it is classified as -1. This is exactly the aim of such systems: to classify an input into the proper output class.

Diving deeply into mathematical details, when dealing with a linear classifier as in the example, the equation used for the function f is

$$f(\mathbf{x}) = sign(\langle \mathbf{w}, \mathbf{x} \rangle + b) = sign\left(\sum_i w_i \cdot x_i + b\right), \tag{1.1}$$

where $\langle \mathbf{w}, \mathbf{x} \rangle$ is the dot product of the input features vector $\mathbf{x}$ and a vector of weights $\mathbf{w}$, and b is a bias term. In two dimensions, this equation is exactly the equation describing a line on a plane and, indeed, it is the line defined in the features space. At every new input $\mathbf{x}$, the equation is solved and the *sign* function detects whether the input point is below or above such line.

As shown in Figure 1.1c, the key aspect for a proper classification is the position of that line, which must separate exactly the points already present on the plane. For this reason, the training phase is used to find the best equation for f.

During the training phase, the training algorithm iteratively aims at finding the best $\mathbf{w}$ and the b which maximize the *margin* between the points already known by the system (i.e. the training set) and the line, as shown in Figure 1.2. The set of

weights **w** defining the separation of the features space are called *support vectors*.

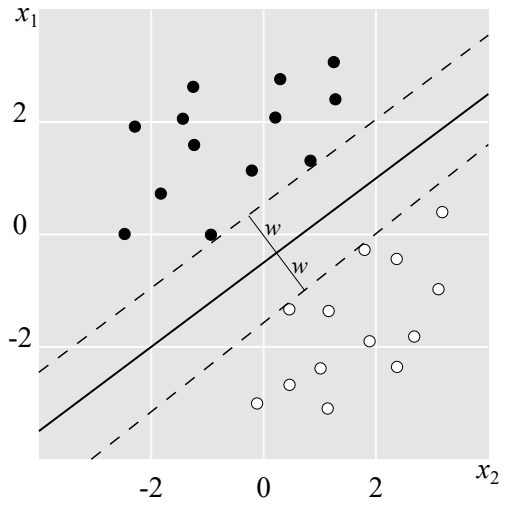


Figure 1.2: Best fitting threshold line.

Those techniques allow us to define the so-called Linear SVM classifier.

1.1.1 Kernelized Support Vector Machine

The Linear SVM classifier is efficient enough until the classification problem is simple and the feature space is linearly separable or close to linearly separable. However, in reality, many classification problems are not that easy and it may be difficult (if not impossible) to perform the optimum separation.

In such cases, the solution to the classification problem can still be achieved by using a *transformation* of the features space in order to switch to a new space where the classes can easily be separated. The transformation is performed by applying a function, called *kernel*, to be used in place of Equation 1.1 with

$$f(x) = sign(\langle \mathbf{w}, k(\mathbf{x}, \mathbf{w}') \rangle + b), \tag{1.2}$$

where **w'** is the weights vector transformed in the kernel space [27], and $k(\mathbf{x}, \mathbf{w}')$ is the kernel function.

Several equations are used in literature for the kernel function depending on the problem to solve. For reference, here are reported the most frequently employed:

- Polynomial: $k(\mathbf{x},\mathbf{w}') = (\langle \mathbf{w}',\mathbf{x}\rangle + c)^n$, where n is the polynomial degree;
- Gaussian or Radial Basis: $k(\mathbf{x},\mathbf{w}') = exp(-\gamma\|\mathbf{x}-\mathbf{w}'\|^2)$, where γ is a positive, non-zero constant;
- Sigmoid: $k(\mathbf{x},\mathbf{w}') = \tanh(\gamma\cdot\mathbf{x}^T\cdot\mathbf{w}'+b)$.

To better understand the effect of this solution, let us take a single-feature example of the top Figure 1.3a.

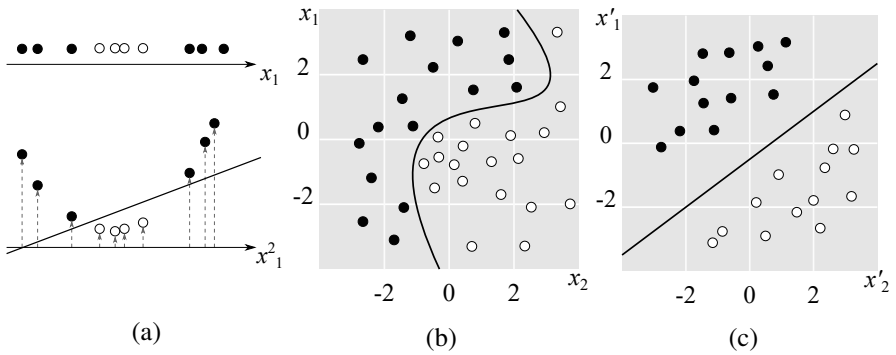


Figure 1.3: Kernelized Support Vector Machine example: (a) single-feature linear (top) and kernelized (bottom) example, (b) original input feature space with kernelized threshold, (c) kernelized input feature space.

In this case, it is clear that a linear separation (which should be done using a vertical line) of the black and white classes is not possible. However, the solution here can still be obtained by transforming the feature space as the bottom Figure 1.3a, in which the x_1 axis is squared to introduce a new feature x_1^2. With this new configuration, the points of the features space rely on a paraboloid instead of a straight line, and the separation turns to be feasible with the same techniques of a linear classifier.

This concept can also be extended to the 2-features example of the previous Section. Figure 1.5b shows again a features space impossible to be linearly separated. However, by using a *cubic polynomial kernel* $k(\mathbf{x},\mathbf{w}') = (\langle \mathbf{w}',\mathbf{x}\rangle + c)^3$, the original features space can be separated and transformed in the new features space of Figure 1.3c.

When using a kernel function to move to another features space, it is suggested to normalize the features using the information acquired during the training phase. In fact, any small out-of-scale which would not affect a linear SVM algorithm, may produce considerable effects in a kernelized implementation due to the amplification introduced by the kernel function.

This means applying the following standard score statistical normalization equation:

$$\mathbf{x}_{norm} = \frac{\mathbf{x} - \mu}{\sigma \cdot s}, \tag{1.3}$$

where μ is a vector of mean values, σ is a vector of standard deviation values and s is a vector of scale values. The output $\mathbf{x}_{norm}$ is the normalized features vector to be used in place of $\mathbf{x}$ in Equation 1.2.

1.1.2 Multiclass Problem

In the real world, it frequently happens that more than two classes need to be distinguished at the output. Although the base SVM algorithm only deals with binary classification problems, multiclass problems can be managed by combining more than one SVM algorithms together. If m output classes have to be classified, two techniques are possible: *one-to-one*, requiring $m(m-1)/2$ binary classifiers, and *one-to-rest*, requiring m binary classifiers.

The one-to-one approach performs the classification in pairs and, in the end, the class with the higher score is used as the final output. The one-to-rest approach instead uses the classification of one class among all the rest of the dataset, actually classifying the class or the non-class. The class which has obtained the highest score is then selected as the final output.

The usage of one technique in place of the other depends on the application requirements.

1.2 Model-Based Design

This Section is about the model-based technique, which has been used as the workflow for the development of the SVM core described in Chapter 3.

The design flow of dedicated hardware architectures is traditionally done by starting with a handwritten Hardware Description Language (HDL) and, after subsequent testing and verification methods, the system is implemented on the destination platform (e.g. MCU or FPGA). However, when dealing with complex systems or in large-scale projects this can be a challenging task, especially if the documentation and requirements collection phases address totally different topics than the hardware design and implementation ones. In fact, in such cases, the workflow must be flexible enough to allow information to flow from the design part to the software/firmware part, and vice versa.

The common way to achieve this task is to produce a set of document-based requirements (which may also be carried out by a separate design team) to be used in the final implementation part (which may also be carried out by a separate software/firmware team). Then, after the implementation, tests are carried out to verify if the requirements were fully and correctly designed for the application. In case of a testing failure, the process goes *backward* to the implementation phase and, if no issues are detected, it goes back further to the design phase. Then, after the corrections, the whole *forward* flow is repeated until both the requirements and the implementation are free of issues.

This trial-and-error approach may be enough for simple or standalone applications, however, if the complexity of the system grows, more design iterations are to be expected, and the waste of time of this methodology can turn to be consistent [28, 29]. As proved by different works [30, 31], dealing with model-based frameworks can help all the phases involved in the process. On the one side, the design phase can benefit from the advanced testing tools offered in the design environment,

easing the requirements simulation and verification, and experimenting with virtually limitless ideas and designs. On the other side, the implementation phase can benefit from the code generation tools usually included in such model-based frameworks, which can automatically translate the functionality of the design phase to a verified low-level code (i.e. C, VHDL, etc.). This can eliminate the time-consuming, error-prone handwriting phase.

1.3 Posit Numbers Theory

In this Section, the Posit numeric format is introduced to understand the theoretical background on which the final part of Chapter 3 relies on.

The Posit format is a numeric data type conceived as a drop-in replacement of the IEEE 754 floating-point standard and which aims at providing an increased dynamic range with the same representation bits [32].

Compared to the IEEE 754 standard, using a Posit representation gives several advantages, such as a reduced accuracy loss, a higher dynamic range and, a Not-a-Real (NaR) notation which substitutes the infinity to avoid overflows or underflows [33].

Posit numbers have the following general binary representation:

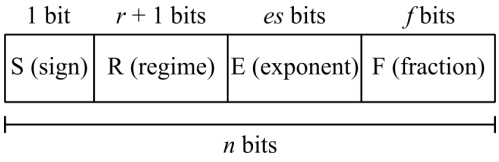


Figure 1.4: Bit-field representation of a Posit number.

where S, R, E, and F are the sign, regime, exponent, and fraction fields, respectively, with their lengths reported on top. The total length of the Posit representation is indicated with n.

The sign field S is always the most significant bit of the Posit bit-string. Its meaning matches the IEEE 754 format, with a 0 and 1 indicating a positive or negative sign, respectively.

The regime field is a sequence of bits starting with r bits of the same value (0 or 1) terminated by a bit of opposite value. As an example, a regime bit sequence can be “000001”, “1110”, “01”, etc. To be used in computations, this field needs to be converted to an integer value with the following equation:

$$k = \begin{cases} -r, & if\ string\ starts\ with\ 0 \\ r-1, & if\ string\ starts\ with\ 1 \end{cases}. \tag{1.4}$$

where k is the integer regime value. The exponent field is different from the exponent definition of the IEEE 754 format. To extract the equivalent exponent value to be used in computations, the following equation needs to be used:

$$exp = 2^{es+k} \cdot e \tag{1.5}$$

where exp is the integer exponent value and e is the integer value represented in the E field. In other words, the exponent exp is the result of the concatenation of k as the high part and e as the low part. In this thesis, e will refer to the Posit exponent bit-field value and exp to refer to the converted value to be used in the computations shown in the following Sections.

The fraction field has the same definition as in the IEEE 754 format: by adding the hidden 1 at the most significant bit, the fraction is directly represented as a 1.F fixed-point number. The real decimal value N of the Posit number can be found with the equation

$$N = (-1)^S \cdot 2^{exp} \cdot 1.F \tag{1.6}$$

When referring to the Posit numbering system, the notation used embeds information about the size of n and es fixed fields:

$$Posit(n, es) \tag{1.7}$$

As an example, the notation Posit(8,1) refers to a Posit format with numbers of 8 bits, of which 1 bit is for the exponent e.

1.3.1 Regime Variable Length

The key concept of Posit numbers is the regime field R, which has been designed to have a *variable* bit length *r*. Since the Posit total size *n* is fixed, having a longer sequence of regime bits translates in "stealing" bits from the exponent *e* and the fraction fields, which are then pushed to the right and outside of the Posit representation. In the extreme case, the regime field can extend to push out both the exponent *e* and the fraction, which are then considered filled with zeros. This property allows us to "tune" the accuracy of the binary representation of the number by acting on the regime string.

Let us consider two cases of real numbers representation: a number close to 1 and a number close to the maximum value allowed by the dynamic range. In the first case, the exponent *exp* is not needed and all the relevant information will be carried by the fraction part. Hence, a high number of fraction bits is required. However, in the second case, the situation is opposite because this time the relevant information is carried by the exponential notation. The fraction part can be neglected and all the bits should be dedicated to the exponent *exp* part.

If the number of total bits *n* is fixed, the Posit numbers can achieve better performance than the IEEE 754 format in both situations because they can move the bit depth where more information is needed thanks to the variable size of the regime field (which concurs to the final *exp* value as shown in Equation 1.5).

Another outcome of the Posit format is the resource saving. If the project constraint of the application is the accuracy and the dynamic range, the Posit numbers can nearly reach IEEE 754 performance while needing half of the representation bits, as shown in [34]. The result is that, if using Posit numbers, the computational load can be lowered while still having a comparable representation quality.

1.3.2 Special Cases

As aforementioned, the Posit standard also identifies two special cases: *NaR* and *Zero*. They have special encodings which must be detected to produce the correct result during operations. In particular, the NaR case is encoded as a Posit string starting

with 1 and followed by all 0 and the Zero case is a string of all zeros.

Chapter 2

Related Works

In this Chapter, the state-of-the-art related to this thesis work is presented.

2.1 Smart Home Systems

The concept of a Smart Home has evolved remarkably over the past few decades. Initially, the Smart Home coincided with Home Automation intended as technological solutions applied to the home environment to automatically manage some situations (e.g., open doors/curtains, control thermostat) and detect dangerous events (e.g., fire, flood), freeing the user from manual control [35]. Now, the evolution of ICT (Information and Communication Technologies) has allowed for the addition of many advanced features to smart homes over time, extending the possible applications. Among the advancements, the monitoring of human activity in the home environment has particular importance. Modern systems exploit human monitoring mainly to improve the energy management of the building [36, 37, 38, 39, 40, 41]. However, data related to the users and how they live in their own home environment can find straightforward applications in health management, allowing for early detection of behavioral trends and anomalies possibly relevant to one's well-being [12, 13, 14, 15, 9, 16, 17].

To accomplish this task, some new devices have to be developed [18] aimed at

considering the interaction of the user with the home environment (e.g. armchair sensors to monitor inactivity periods, toilet proximity sensors, etc.). These devices must be able to send data over the Internet so that, once processed, they can be accessed by the various professionals involved, for example, medical doctors, caregivers, relatives, and the users themselves.

Applications based on such a system have been presented in past works. For example, an infrastructure based on a Passive InfraRed (PIR), bed and temperature sensors were reported in [42]. In this case, the data transmission was based on the X10 wireless protocol, and a home server was exploited to process data and to send analyses and alerts to a third person (i.e. the clinician) through emails. Another monitoring system based on temperature, pressure, PIR sensors, and actuators was described in [43]. Sensors communicated with a local computer by means of a wireless ZigBee protocol. Data were processed to monitor the activities of elders, and the results determined which actions to take. The system proposed in [44] leans on a mix of ZigBee and Power Line Communication (PLC) transmission protocols. The architecture was conceived with distinct ZigBee Wireless Sensor Networks (WSNs) in each room, which communicate with a central management station through a PLC. An infrastructure exploiting a custom wireless protocol, the so-called Wellness protocol, was presented in [45]. The implementation was based on temperature, pressure sensors, PIR sensors, a manual alert button, and actuators. Another example of such a system was CARDEA (Computer-Aided, Rule-based Domestic Environment Assistant), developed at the University of Parma and specifically aimed at behavioral analysis. Originally based on the Ethernet protocol [46, 47], the system's strength is flexibility. It is conceived to integrate different kinds of sensors and to support smart interfaces [48, 49], in order to tailor the system's functions to the specific users' needs. By supporting a wireless protocol (i.e. IEEE 802.15.4/ZigBee) as well, further kinds of devices were introduced; the most remarkable is the wearable sensor MuSA (MUltiSensor Assistant) [50, 51, 52, 53], designed for user motion analysis (e.g. fall monitoring) and for localization and identification purposes [54, 8].

Most of the systems presented in the literature based their connectivity on wireless protocols featuring low-cost hardware and low power consumption (e.g. ZigBee

[19, 20, 21]). Coupling the low power required by a ZigBee transmission with recent developments in the field of energy harvesting, it is now possible to develop battery-less IoT nodes [55, 56]. However, powering the ZigBee protocol tasks for prolonged periods of time (i.e., >10 s) with energy harvesters is not recommended [57]. Some advanced functions (e.g. localization or identification [58]), useful in Smart Home systems conceived for the continuous monitoring of the individuals and for behavioral analysis, are likely to be prevented. Moreover, the disadvantage of the ZigBee (or low-power equivalents) approach is mainly the necessity of completely new and dedicated infrastructure in each home in which the system is going to be deployed, since, typically, a network compliant with these standards is not already present. Furthermore, a gateway device has to be considered as an interface between the home protocol and the Ethernet, in addition to a number of range extenders due to the low range featured by ZigBee [59].

In the D2Lab research group of Electronic Engineering at the University of Parma, a new home monitoring system entirely based on Wi-Fi connectivity has been developed, in a fashion strictly compliant with the Internet of Thing (IoT) paradigm. Sensors connect to the Internet through a standard Wi-Fi home router, which is quite often already present in the homes. Some manufacturers are already producing chips for routers able to support Wi-Fi, Bluetooth, and ZigBee protocols in parallel (e.g., Qualcomm QCA4020 and QCA4024 [60]). These solutions could simplify the adoption of multiple protocols in the near future, but do not eliminate the need for building a complete infrastructure for the ZigBee sensors and in particular for deploying range extenders in the environment. On the contrary, Wi-Fi features a much wider range than ZigBee, so that no (or fewer) network extenders are needed, simplifying the overall approach, lowering the costs, and making the whole system more scalable. These characteristics open the system to the market, making it particularly attractive to consumers. This, of course, comes at the expense of increased power consumption, which may harm battery lifetime. Therefore, dedicated sensor platforms have been carefully designed at D2Lab for low-power operation composed of both environmental and wearable sensors. The general system architecture is sketched in Figure 2.1. It includes a set of sensing devices, a Wi-Fi router, and services installed on a cloud

environment. Currently, it exploits the IBM Bluemix Watson IoT platform (although other cloud services are supported, e.g. Amazon AWS, Microsoft Azure, MathWorks Thingspeak, etc.), which offers a dashboard that enables a handy visualization of sensors information, as shown in Figure 2.2.

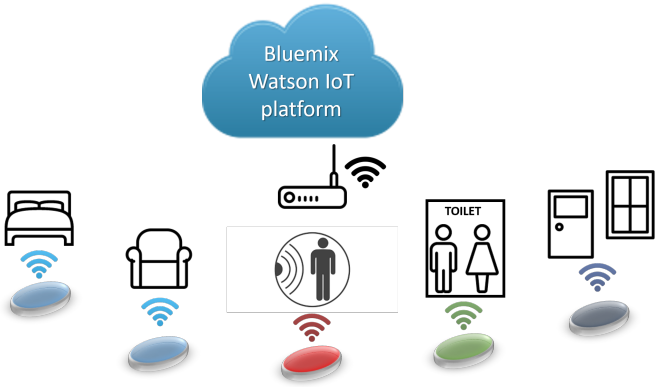


Figure 2.1: Wi-Fi system architecture.

2.2 State of the Art in HAR

Nowadays, human activity recognition is a subject of research interest because of the benefits it can bring in several fields, i.e. sport, health, security, etc. Several research groups realized and tested novel approaches to achieve better performance and/or lower costs mainly following two paths: hardware and firmware/software solutions. These are two sides of the same coin because the optimization of one gives positive results in the development of the other and vice versa.

In this Section, the state-of-the-art is reported to figure out the current advancements on this topic. Starting from the wider point of view of HAR, the research is organized to end with a closer look at HAR with wearable devices. In this journey, a view on embedded hardware and firmware/software solutions is also given to have an idea of how the literature is moving on these fields.

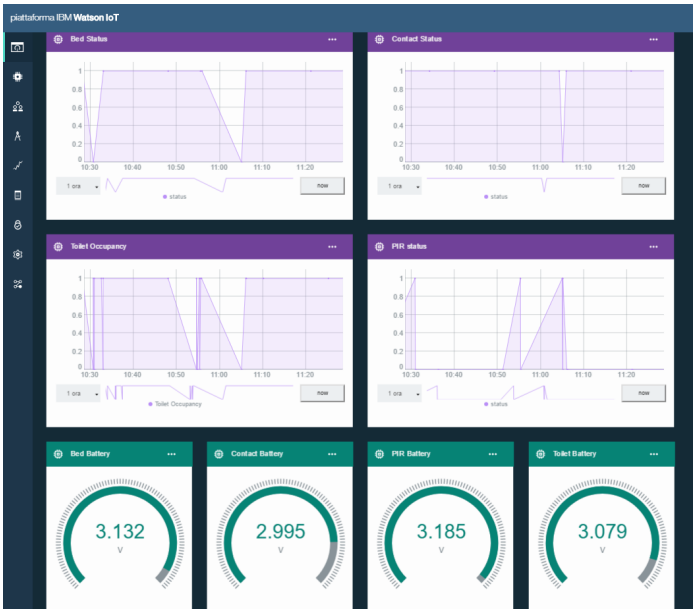


Figure 2.2: IBM Bluemix Watson dashboard reporting Wi-Fi sensors activity and battery level.

As a starting point, the best way to have an idea of the landscape is to analyze review works on this subject. In [61], the team resumes the state-of-the-art in the activity detection and classification field exploiting wearable devices. The study is mainly focused on the hardware type (accelerometer, gyroscope, magnetometer, camera, audio, electrocardiography (ECG), photoplethysmographic (PPG), etc., along with their location), algorithms employed (decision tree, SVM, K-Nearest Neighbor (KNN), etc.) and activities recognized (sitting, standing, walking, etc.). The result is an identification of a set of research works that have been summarized and classified by type of sensor (accelerometer/gyroscope/magnetometer, camera, or hybrid), number of used sensors, number of subjects used for testing, and type of classification learner employed (threshold, unsupervised, supervised).

In [62], Hooshmand et al. review the state-of-the-art algorithms used to lossily compress bio-signals with the aim to evaluate the best approach to realize efficient

energy saving in wearable devices. The study considers 1-D bio-signals such as ECG, heart or respiratory (RESP) rates, and PPG. It presents novel approaches based on online dictionaries, elucidating their operating principles and providing a quantitative assessment of compression, reconstruction, and energy consumption performance of all schemes. In the end, the team considers the most efficient schemes the ones which allow for reducing the signal size up to 100 times, entailing similar reductions in the energy demand while still keeping the reconstruction error within 4% of the peak-to-peak signal amplitude.

2.2.1 Hardware Solutions

The hardware side of a HAR system can be identified by both the sensing parts involved in raw data collection and the processing parts dedicated to data processing. Depending on the final applications, these two components can reside on the same device (e.g. MuSA [52], a wearable sensor that signals a detected fall) or can be two or more distinct devices (e.g. SmartShoe [63], a shoe that sends raw data to the smartphone for energy expenditure calculation). In the second case, the hardware dedicated to the processing phase can be treated as ideal, typically being a system with a high amount of processing power to make this computational effort negligible. The first case is instead the most challenging one, as it is the configuration that provides ubiquity but, at the same time, forces to deal with physical size, battery lifetime, and computing capability.

In literature, several systems are presented, each one proposing the better trade-offs for the application.

In [64], a monitoring system based on muscular activity is presented. This is commonly done by electromyography (EMG) systems, a precise method but too sensitive to environmental disturbances. Here, the new exploited method relies on air-pressure sensors and air-bladders. Since the change of the air pressure can be more robustly measured compared with the change of skin electric signals, the proposed sensing method is useful for mobile devices due to its great signal-to-noise ratio (SNR) and fast response time. In the paper, the research group verified the performance of the system by comparing it with an EMG signal and motion sensor.

A totally different approach was adopted in [65], in which a monitoring system able to perform activity recognition by monitoring changes in Wi-Fi signals (hence without a device that needs to be worn) is presented. The method exploits a Channel State Information (CSI)-based Human Activity Recognition and Monitoring system, called CARM. This is a merge of two models: the first is based on a CSI-speed model that quantifies the relation between CSI dynamics and human movement speeds, the second is a CSI-activity model that quantifies the relation between human movement speeds and human activities. This system has been tested with the implementation on commercial Wi-Fi devices and, after the analysis of True Positives (*TP*), True Negatives (*TN*), False Positives (*FP*), and False Negatives (*FN*), the group achieved the recognition accuracy of Equation 2.1.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} = 96\% \tag{2.1}$$

Similarly to [65], the work presented in [66] is about PAWS (Passive Human Activity Recognition Based on WiFi Ambient Signals) in [66], an indoor localization system. It uses the RSSI fingerprint of different activities to perform activity recognition in a Wi-Fi environment. To reach the goal, a novel fusion algorithm has been specifically designed based on the classification tree, resulting in an outperform of three other well-known classifiers in terms of accuracy and complexity: NaiveBayes, Bagging, and KNN. The group presented a first prototype version for tests in real-life conditions.

In [67], the work focuses again on device-free wireless activity recognition and localization with the aim to simplify the characterization of the target, a key aspect to increase the accuracy of the recognition system. The solution here was to apply a sparse auto-encoder network algorithm for wireless features discrimination and then, with the selected features, a softmax-regression-based machine learning framework was used. This enables the system to simultaneously perform location, activity, and gesture recognition with an accuracy of 85%. This value is proved to be higher than what was obtained by systems using traditional handcraft features.

Another indoor device-free activity recognition technique has been studied in [68]. Leveraging the subject and analyzing the non-line-of-sight (NLOS) radio fre-

quency signals, a KNN classifier algorithm has been used on the packet receive rate to recognize the activity of a person. They also found that the radio frequency of the signal is proportional to the height of the person, so to extract that information too. With this method, the group achieved an average accuracy increase of 44% compared to the fixed-sample RSSI-based method and an accuracy of 76.6% in distinguishing the height of the moving target.

Wearable shoes (SmartShoe) are used instead in [69], which were equipped with pressure sensors and an accelerometer to analyze user movements. In the paper, the group focuses on a comparison between two classifiers (SVM and Multilayer Perceptron (MLP)) used for activity recognition, both with rejection. They find out a benefit using MLP, reaching a 99.8%±0.1% of accuracy.

As a development phase, in [63] the same group explores the feasibility of a system in which data acquired by the shoes are sent to a smartphone for the computation of the Energy Expenditure (EE) and to perform activity recognition. To test the best algorithm to use, they make a comparison between three classifiers: SVM, Multinomial Logistic Discrimination (MLD), and MLP. Tests have been carried out on 15 subjects and the result is that, despite the same accuracy given by all three methods (~95%), MLD and MLP require less running time and memory by a factor of more than 10^3 for activity recognition. No significant differences are present for EE.

With a similar sensing element (shoe) but with different addressees, [70] shows HAR applied to caregivers during their work. The aim is to detect possible overexertion injuries caused by awkward postures during patients handling. The group uses a smart wearable device (named Smart Insole 2.0) and a novel spatio-temporal warping pattern-recognizing framework. The results validation is done by a pilot study on eight subjects and an overall classification accuracy of 91.7% is achieved.

Another change in hardware occurs in [71], where the focus is on heterogeneous features learning for RGB-depth (RGB-D) activity recognition using a dedicated video camera. The group extracted similar hidden structures coming from each of the four channels and applied a joint learning model to explore shared and feature-specific components. The uniform system has been called "joint heterogeneous features learning (JOULE) and has been proved to successfully recognize human activ-

ity, focusing on human-objects interactions.

Coming to smartphones, [72] shows a performance analysis of an activity recognition system based on modern mobile phones. By analyzing sensory data sequences (form accelerometer, gyroscope, light sensor, proximity sensor, barometer, and GPS) of 10 subjects, the group assigns an F-score number to categorize performances under various placement settings (arm, hand, pocket, and waist). With the additional implementation of four different multi-class classifiers (linear and Radial Basis Function (RBF) kernel SVM, Random Forest, and KNN) they make a complete comparison of the techniques based on smartphones hardware.

A mixed approach between device-free and wearables is used in [73], in which a study of an activity recognition system based on Passive Radio-Frequency Identification (RFID) tags is presented. The idea is to use several RFID tags (which can be embedded on the clothes) and an RFID reader with two antennas, all worn by the user. This system is proved to be able to detect human motion by analyzing RSSI signals between tags and antennas. Data processing is performed in a dedicated online environment that provides the result with a latency of 5 seconds. The outcome is a non-invasive set of objects that, after two weeks of tests on four subjects, gave an accuracy of 93.6%.

The same RFID technology with a different application is used in [74]. The aim here is to perform activity recognition based on objects tracking, hence human-object interaction. The system relies on one or more RFID tags placed on the object to track, and several antennas placed around the room. Tracking is performed exploiting antennas signals trilateration based on RSSI information. In particular, a Gaussian Mean Weighting Filter is used to correct RSSI instability and Elliptic Trilateration is used to decrease the error given by the antenna not being omnidirectional. Results show an average activity identification of 93.3%.

A peculiar HAR system is then presented in [75], which introduces a floor embedding pressure sensors. They use piezoelectric sensors to get pressure information proportional to the user's gait and, with this particular technology, large areas such as train stations, hospitals, or shops can be covered. After some signal conditioning hardware, that set of information is sent to a central unit for data processing. They

demonstrated the ability to recognize footsteps, jumps, sitting down into or standing up from a chair, and impact of objects with an accuracy of 100% using Pearson Product-Momentum Correlation Coefficients as a classifier.

In [52], the MuSA wearable sensor is introduced, with the ability to perform localization, identification and, exploiting its on-board sensors, fall detection, heartbeat, and breathing rate. The sensor is thought to be worn at belt or chest and, with the other sensors included in CARDEA environment [76], provides an ambient assisted living to support interaction between users and caregivers.

2.2.2 Firmware and Software

In addition to hardware elements intended to sample and process data, a HAR system needs efficient algorithms to obtain the result. Raw data cannot directly provide information about human activity but must be modified by one or more algorithms dedicated and possibly optimized to retrieve certain information. Research on this topic teems of excellent works aimed to find the best approach to extract the information of interest, regardless of whether the goal is the computational speed, the accuracy, or the limited hardware.

In [77], the system performs activity recognition exploiting an accelerometer placed in different places: waist and ankle. Exploiting a ZigBee connection, the aim of this study is to find an optimized algorithm for this low-performance hardware and to increase independence from the sensor position. For data processing, the research group uses a novel Ensemble Empirical Mode Decomposition (EEMD) to identify the set of features and cooperation Game theory for features selection (GTFS). In the end, a consistent increase in accuracy is shown using the EEMD features and, using the GTFS, a significant reduction in the needed features is obtained.

A different HAR algorithm able to work with few labeled activity data and additional information from users (i.e. weight, height, etc.) is presented in [78]. So doing, the system can identify the fitness of others' models relying on a small amount of labeled data from the new user. Once placed the new user in the proper shared activity model, the algorithm uses Bayesian networks and SVMs to perform activity recognition. The result is an increase in accuracy (83.4% compared to 77.3% of indi-

vidual and standard population models). Another aspect arising from this study is that this new method provides better results than methods based on users' demographic information.

A camera-based approach is used in [79], which describes a Fuzzy Segmentation and Recognition algorithm applied to video recordings of user's actions. The main focus is to be able to correctly recognize activities with gradual transitions between them. To do so, the algorithm performs a segmentation of the video into a sequence of non-overlapping blocks, each one lasting a short period of time. Then, multivariable time series are formed by concatenating block-level human activity summaries that are computed using topic models over local spatio-temporal features extracted from each block. After encoding the fuzzy event through Fuzzy Segmentation, the system can correctly apply an activity label with transitions. Results based on six datasets give an accuracy between 42.6% and 78.9%.

The work presented in [80] introduces the Interest Point features as an improvement in video input sequential modeling for action/activity recognition. To obtain the result, the algorithm first splits the video sequence into short overlapping segments of fixed size through a sliding window technique. As the second step, each segment is described by a local Bag Of Word (BOW) of the histogram of Interest Points (or their trajectories) enclosed in the segment. As the last step, a first-layer SVM classifier converts each BOW into a vector of class conditional probabilities that are then given as input to a second SVM (hidden conditional random field) for actual recognition. A comparison with state-of-the-art algorithms on three public datasets shows that this system can outperform or match each leader.

Referring to the same video activity recognition, [81] proposes a system that can recognize incoming visual content based on the previously experienced activities. The high-level activity is parsed into consecutive sub-activities, and a context cluster is built to model the temporal relations. The semantic attributes of the sub-activity are organized by a concept hierarchy. The dynamical evolution of the brain memory is mimicked to allow the input information to decay and reinforce, providing a natural way to maintain data and save computational time. Performances are tested by comparing this algorithm and traditional incremental learning method [82] on three

public datasets (CAD-120, MHOI, and OPPORTUNITY). Results demonstrate the same accuracy with less time cost.

With the addition of depth information on the camera's signals, the work described in [83] shows a novel framework to be applied for HAR. The techniques employed are first an extension of the surface normal to polynomial by assembling local neighboring hypersurface normals to jointly characterize local motion and shape information and then a supernormal vector (SVN) scheme to aggregate low-level polynomials into a discriminative representation (such as a simplified Fisher kernel representation). To capture the spatial layout and temporal order, a subdivision of the depth video into a set of space-time cells is performed using an adaptive spatio-temporal pyramid. To make a comparison, a linear SVN solver is used on four public datasets and results are compared to state-of-the-art public HAR algorithms. An increase in accuracy of up to 3% is obtained.

In [84], the aim is to study the optimum classification method of accelerometer signals for Activity Recognition (AR) and Movement Recognition (MR) applied to rehabilitation and elderly monitoring. The work exploits an accelerometer of a smartphone to recognize several activities (idle, still, walking, running, going up/down the stairs, or cycling) and movements (arm circles, arm presses, arm twist, curls, seaweed, and shoulder rolls). As a comparison, SVM, decision trees, and dynamic time warping classifiers are taken into consideration. Results show an accuracy above 90% in AR and above 99% in MR using SVM-based approaches.

Referring to Ambient Assisted Living (AAL) field, [85] focuses on the difficult task of estimating the required window for online sensory data streams (coming from a variety of sensors) to recognize a specific activity. The proposed solution is to split the problem into two phases. First, an Offline Phase is performed. Starting from an annotated training set and with the most relevant feature selected (in the study this is chosen among a number of activations of each sensor, activation duration of each sensor, number of activated sensors for each activity, and the location of the sensor), a matrix containing the best fitting sensor for each activity is calculated using Information Gain attribute evaluation [86]. The second step is an Online Phase. Here, an optimum window is calculated by checking if the current active sensor is contained in

the best fitting matrix. The phase continues then with the extraction of the features to be used in the subsequent multi-class classification. After all these steps, the activity is given as result.

In the work presented in [87], the group proposes a bidirectional feature to be used during the features extraction phase of activity recognition. Because of the two components of a bidirectional feature, the problem can be treated as a second-order tensor. For this reason, the research group defines a new tensor-based feature selection method named Tensor Manifold Discriminant Projections (TMDP). It simultaneously applies an optimization criterion that can directly process the tensor spectral analysis problem, extracts local rank information by finding a tensor subspace that preserves the rank order information of the within-class input samples, and extracts discriminant information by maximizing the sum of distances between every sample and their interclass sample mean. Using Principal Component Analysis (PCA), the vectorized tensor feature is transformed to the ultimate vector feature representation, making TMDP robust to the noise by PCA's noise-filtering role. The algorithm is then compared to other traditional vector-based feature selection methods such as LDA, LPP, PCA, DLA, GMSS, HMSS, etc. Results show that TMDP outperforms in each proposed comparison.

Then, in [88], the research team investigates the use of template matching for recognizing sports activities using one single accelerometer worn at the wrist. This work differs from the others because of its limited sensing hardware (a single accelerometer) and the final generalized application target in terms of subject, sensor, and measurement-axis. Relating to this last aspect, the aim is to evaluate the ability of a template-matching classification algorithm to generalize on a population of overweight subjects. The system is evaluated by processing data collected in a gym environment, and results are compared with four popular classifiers: Decision Trees, Naive Bayesian, Logistic Regression, and Artificial Neural Network. The low accuracy obtained with the best template-matching metric is imputed to the lower accuracy in classifying cross trainer and rowing activities.

2.2.3 Wearable Devices

Among HAR systems, those exploiting wearable wireless devices are the most challenging. Differently from fixed sensors, a wearable sensor faces some constraints: 1) size and weight must be so to make the device less invasive as possible; 2) battery lifetime must be long enough to not force the user to frequent recharges; 3) the accuracy must be comparable with other fixed devices.

It is clear that each point is constraining each other (e.g. a lower size may lead to a lower battery capacity and so to less lifetime) so an application-related trade-off has to be found after context evaluation.

Referring to point 2), several research activities have been done to demonstrate how that goal can be reached by acting on both the hardware and firmware side.

Optimization for Battery Lifetime

One of the main key points of a wearable device is the battery lifetime. To provide the right user experience, a wearable device must be able to work continuously as much as possible, so as to avoid the user charging (and not using) it too frequently. The first and easier solution is to increase the battery capacity. This is a good practice until the object becomes too big and heavy to carry. Excluding the battery capacity increase, some ways to achieve the result have been tested by several research groups.

In [89], a Wi-Fi wearable system suitable for ambient assisted living is shown. The challenge here is to build a Wi-Fi-based system (as to be able to exploit the existing home wireless and internet user infrastructure) capable to provide ambient assisted living features. This work focuses on the energy management of the Wi-Fi module, a communication system that lacks strong energy policies and born to support higher-speed transfer rates and wider area coverage when compared to systems typically employed in home environments (e.g. Zigbee, Bluetooth, etc.). Thanks to a deep study of both hardware and firmware techniques, the group obtains a prototype wearable device able to match the operational lifetime of sensors relying on the aforementioned energy-aware communications systems.

A different approach has been used in [90], where the energy saving is achieved

by dynamically set the sampling rate of the sensible element (an accelerometer) to reduce the transmitted packets over time. By evaluating the motion magnitude of the sensor, the algorithm is able to set a high sample rate if the sensor is moving or a low sample rate if it is steady. The lower sample rate configuration is defined as guaranteeing a minimum and predefined activity recognition accuracy. It is clear that this operating mode brings a trade-off between accuracy and energy saving, with the proposed method shown to have an overall energy saving of 45.9% when compared with the same device continuously streaming at the maximum sampling frequency (12 Hz). The observed accuracy loss in the activity recognition is 0.41%.

Another dynamic power solution has been identified in [91], this time on smart wearable systems used in infants' sleep monitoring. The technique is based on a Transmission Power Control (TPC) mechanism that changes its characteristics according to the scenario of operation. It uses Inertial Measurements Unit (IMU) sensors to determine the position of the infant and, based on that, predicts the current state of the channel by evaluating RSSI signals. In the end, a comparison is done considering other state-of-the-art TPC algorithms and different infant positions. When compared with the same device at full transmission power (0 dBm), the proposed TPC mechanism reaches a maximum energy saving of up to 47% depending on the infant position.

In [92], the Authors introduce NEON, a wearable fall detector based on a low power MCU (Texas Instruments MSP480), two accelerometers, a barometer, an RF transceiver, and a single 3V, 1Ah coin cell battery. In addition, the paper reports a study on the classification features to be selected to build an energy-efficient fall detection wearable device. The group analyses a large set of data coming from a single accelerometer collecting data at 50 Hz and related to 10 young healthy volunteers, each one performing predefined actions to be classified as falls or non-falls. Those data have been then processed to find the best energy-efficient features (in terms of computational power required by the MCU in the calculation) to be used in the binary decision tree algorithm of choice. After the features selection, the system has been assessed and proved to have a power saving of 75.3% when using four out of ten of the most used features for this task in literature. The reduced number of features

affected the system with an error rate of 7.1% in the recognition accuracy.

The same NEON wearable device was exploited in [93] to propose further energy-saving techniques, which are somehow similar to [91]. In addition, to describe the right low-power hardware, the system here also benefits of particular accelerometers configuration: one accelerometer serves as the "static" sensor (ACCsm, sampling in low-power at 6 Hz) and the other one is used as the "dynamic" sensor (ACCfd, sampling in the power-hungry 50 Hz frequency). If the system is in a stationary mode, only ACCsm is on and, once ACCsm senses an intense motion, ACCfd is also turned on to catch a possible fall. This technique, combined with a computation-efficient features extraction to be used in the SVM classifier, led to a system with an estimated battery lifetime of 1,125 days.

Coming to running analysis systems, in [94] the team presented Gazelle, a compact, lightweight, accurate, and highly energy-efficient wearable device born to perform online running analysis to prevent injuries. The main aspect of this device is the energy consumption, kept low thanks to the proposed Sparse Adaptive Sensing (SAS) technique. Similarly to [93], the system exploits two accelerometers, one featuring low performance and low energy consumption, and one with opposite characteristics. With the first one always on, the second one is triggered only on significant motion. Once turned on, the second accelerometer's energy consumption is still kept under control by adapting its sample rate to the actual activity. Since 2014, Gazelle has been used in real-world testing by over 100 runners (of different skill-level) and data show an accuracy of 97.7% and an average energy saving of 83.6% when compared with the same device recognizing the same activities and constantly sampling the high-performance accelerometer at 200 Hz. The system is reported to reach more than 200 days of continuous high-precision operation using a single coin-cell battery.

In [95], the authors propose the usage of multiple cells battery pack as the power source with a scheduling algorithm to decide which cell has to work based on the remaining energy. Differently from other battery management systems, in this case, the selection of the operating cell is done using the kinetic battery model (KiBaM) instead of a linear one. Experimental results show that in real conditions the battery is not constantly providing energy but behaves differently depending on the working

state of the sensor. In particular, an increase in battery voltage is observed during low-load periods. This behavior is not included in linear battery models and this leads to errors in battery charge evaluations. Using KiBaM algorithm (i.e. a better active cell selection), simulations show an increase of 75% in battery lifetime when compared to sequential scheduling.

A deep dive into the energy accumulation technology is presented in [96], where the group analyses non-ideal properties of batteries and how these properties may impact power-performance trade-offs in wearable computing applications. The study covers both ideal and non-ideal aspects of batteries such as "C" rating and recovery. Using Doyle's Li-ion battery discharge model during simulations, the group ends up with a summary of key points to consider during battery lifetime estimation, dimensioning, and usage.

Chapter 3

Environmental Sensors

This part of thesis was aimed at the assessment of the performance of the environmental sensors by means of a series of tests carried out both in a laboratory and in a real home environment. In detail, four sensors were involved in the tests (Magnetic Contact, Armchair/Bed, Passive InfraRed (PIR), and Toilet proximity sensors) and, in the end, measurements about the power consumption have been performed and presented.

Moreover, during the tests, the sensors' data were logged to carry out a possible activity profile of the users in order to test the system performance. Although the aim of this work is not to develop and demonstrate a new behavioral analysis method, this was useful to assess the new hardware architecture conceived to collect data usable by behavioral analysis models developed elsewhere [42, 97, 98, 99]. These tests show the usability and convenience of data collected and demonstrate that it is possible to take advantage of the flexibility of the Wi-Fi platform while eliminating costs given by the need for additional hardware (i.e. home servers and routers, adopted in [16, 17, 18, 42]).

In detail, the environmental Wi-Fi sensors under test are:

- Armchair (or Bed) occupancy sensor, to monitor inactivity periods or sleep disorders;
- Passive InfraRed (PIR), to monitor the movements inside the house;

- Toilet proximity sensor, to monitor the toilet accesses;
- Magnetic contact, to detect door and windows opening/closing. The same device can be used to monitor interaction with other meaningful objects (e.g. a cupboard door, to monitor feeding habits, or the medicine cabinet, for monitoring compliance with therapy prescriptions).

All sensors are directly and individually connected to the Internet through the Wi-Fi home router. The commissioning procedure is very simple and relies on the Wi-Fi Protected Setup (WPS) standard. No technical skill is required so that consumers themselves could manage the installation procedure. Data are sent to the Watson IoT platform, inside the IBM Bluemix cloud services, via MQTT (Message Queue Telemetry Transport) protocol with a Quality of Service (QoS) equal to 2. that the message is received by the broker once and once only. The payload of the message is a string in a JSON (JavaScript Object Notation) format, reporting the sensor status. The device firmware, nevertheless, is suitable for connecting to other platforms as well (e.g., Amazon AWS (Amazon Web Services), Microsoft Azure, Thingspeak, etc.).

A Wi-Fi-certified system-on-chip (SoC) (CC3200, Texas Instruments, Dallas, TX, USA [100]) was chosen as the core of all sensors. It integrates an 80 MHz clock MCU (MicroController Unit) with a 32-bit architecture and a network processor (compliant with the IEEE 802.11b/g/n network radio protocol).

For prototyping purposes, commercial development boards were exploited as the motherboard for all the devices. By connecting the same motherboard to different sensing devices, different sensors were built, sharing the architecture, thus reducing development costs. The sensors are shown in Figure 3.1. All the devices can be powered from an AC (Alternating Current) outlet (through a standard USB port). This possibility eliminates the need to control and replace the batteries. However, the sensors' position is very important to efficiently monitor the users' activities. This position has to be independent of where the outlets have been set up in the house, but, at the same time, it is necessary to avoid wires in the room or to modify the existing home electrical system. For this reason, the devices are also conceived for

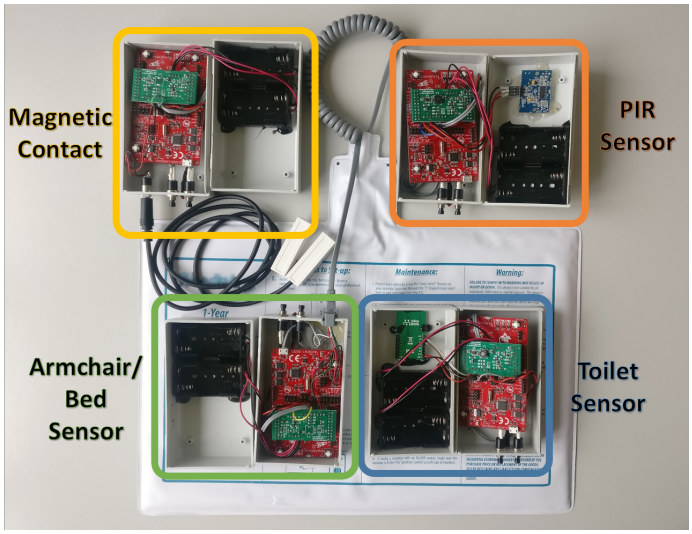


Figure 3.1: The sensors prototypes. PIR = Passive InfraRed.

battery-powered operation. In order to preserve battery lifetime, careful management of the energy budget was implemented. In particular, simple sensors do not need to fully exploit the Wi-Fi data transmission rate, since a limited quantity of data has to be exchanged over time. So, taking advantage of such an overabundant data rate, sleep phases are introduced in the operating cycle. The devices alternate active (data processing and transmission) and sleep intervals. Among available low power modes of the MCU, the Low-Power Deep Sleep (LPDS) and the HIBERNATE modes were exploited. In LPDS mode, the MCU stops its main clock while the radio module is active and maintains the connection with the network. The device is more reactive to external interrupts and features low wake-up times. In HIBERNATE mode, the radio section is also turned off. Hence, HIBERNATE is the most effective in power saving, but, when awakening from hibernation, the device needs to re-connect to the network. Wake-up times are in the order of tens of seconds, so hibernation is convenient when the sleep time is long enough. In our previous study [22], it was shown that the overall power performance of LPDS mode is better than HIBERNATE whenever a large enough frequency of messages (higher than 150 messages/day) is to be managed.

Considering the fact that all the sensors need to send a keep-alive message (every 15 minutes, in the application at hand), hibernation turns out to be practical whenever only a few tens of actual "events" per day are expected to occur. Among the sensors presented in this first implementation of the Wi-Fi system, this characteristic may fit with Armchair/Bed sensors, Toilet sensors, and the several uses of the Magnetic contact. On the other hand, LPDS mode may be a better fit for higher frequency events, as it happens for the PIR sensor. Based on such general assumptions, in the following specific implementation, details are given for each device.

- Armchair/Bed sensor: In [101], a characterization of three different chair sensor elements is presented: a strain gauge, a mechanical switch, and a vibration sensor. Higher detection accuracy is reported for the mechanical switch, but this kind of sensor is more difficult to integrate into an ordinary home chair/bed since it is not straightforward to cover the whole sitting area. For our application, the strain gauge seems the most indicated choice. Indeed, resistive pressure pads are being commercialized which can be placed under the bed mattress or over the chair seat. These pads have been selected as sensing elements. The resistance change is assessed through a simple voltage divider, which drives a binary threshold comparator. This, in turn, generates an interrupt to wake up the MCU. As mentioned, given the relatively low expected number of events, the HIBERNATE sleep mode is exploited during idle phases.

- Magnetic Contact: The sensing element is a reed switch, coupled to a magnet. When the two components are close to each other, the switch opens. This configuration is particularly convenient when drawers/doors are supposed to stay closed most of the time. Since the reed switch is open when they are closed, the sensor drains no current while in this state. Interrupts are generated to signal both transitions (close to open, open to close). In this case, the selection of sleep mode depends on the actual device function. If, for instance, applied to the home main door, the HIBERNATE mode seems to be more effective again.

- Toilet proximity sensor: The sensing element is a distance/proximity sensor. The sensor's purpose is to allow for counting toilet visits (which may be rele-

vant to many medical conditions), distinguishing them from generic bathroom presence (due to washing, for instance), which could be assessed through a PIR sensor. A short-enough, personalized reading range is therefore needed to cope with the actual placement of the sensor itself. The device has a reading range from 10 cm to 150 cm. The analog reading is fed to a comparator, which generates an interrupt signal to the system core. In our experiments, we found that calibrating the sensor threshold at 65 cm distance was effective in discriminating toilet actual usage from generic bathroom presence. The nature of this sensor should make it suitable to exploit the benefits of the HIBERNATE mode. However, due to the high current consumption of the sensing element during its on-state, the system core needs to power it on only when a distance reading is needed, keeping it in the off-state otherwise. In our tests, we found that a period of 3 seconds for power off followed by 0.5 s of power on gives the optimal system reactivity. Since the sensing element duty cycle is driven by the MCU, LPDS mode had to be adopted.

- PIR sensor: the sensing device is a standard passive-infrared motion sensor. It requires a supply voltage in the range of 3.3–5 V. In order to allow the system to be powered by AA batteries (as a common feature of all devices), a boost DC-DC regulator (direct current to direct current power converter that steps up voltage) has been added. The device already has embedded converters and provides a digital output signal, which is used to wake up the system core whenever a movement is detected. To avoid communication overload, PIR data are filtered on board; only status changes are transmitted, besides keep-alive messages. In the envisaged scenario, however, this results in a number of daily messages exceeding the above-mentioned threshold, which makes the adoption of hibernation not suitable and makes the adoption of LPDS mode preferable.

3.1 Power Consumption Analysis

To assess the devices, field tests were carried out for two months. For standby battery-lifetime evaluation, a full set of sensors was installed in a laboratory environment and

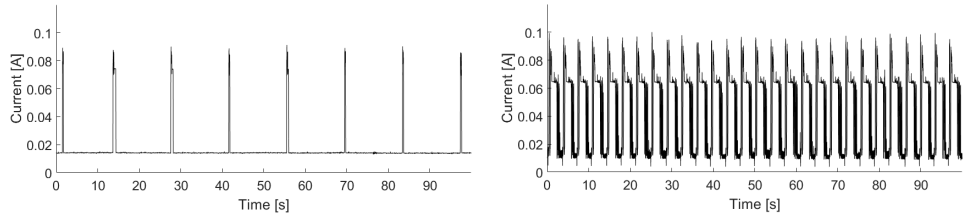
configured as to not catch any event caused by user interactions. With this setup, the aim was to only evaluate the power consumption caused by the keep-alive events sent by the sensors every 15 minutes. In addition, in order to estimate the battery lifetime in a working condition similar to the real one, another full set of sensors was installed in a real home environment inhabited by a family. In this case, each sensor was deployed accordingly to its purpose:

- Armchair sensor: Sensor pad placed on a lunchroom chair.
- Magnetic contact: On the bedroom door.
- Toilet sensor: Inside the bathroom to sense toilet interactions.
- PIR sensor: Inside the bedroom.

In a previous work [22], preliminary laboratory tests about the sensors' power consumption were carried out with the assumption that both the connections to the Wi-Fi network and to the Internet were reasonably stable. In real operation, this is not always true, and here, in order to consider possible disruptions that a consumer may experience, two main issues were evaluated:

- Sensor unable to connect to the Wi-Fi network: The system, after power on, is configured to search for the previously known network (through the WPS procedure) until connection succeeds; if the network connection is broken (e.g., because of a blackout), the device performs a reboot. The system always experiences an energy-expensive boot-loop until the network is restored;
- Sensor connected to the Wi-Fi network but without active Internet access (e.g., because of Internet provider issues): This situation arises only after the previous condition is met. After power on, the device searches and connects to the known network; it tries to establish a connection with the online cloud. If the procedure fails (e.g., because the Internet connection is lost), a reboot is performed. The result is the same, with the system always in an energy-expensive boot-loop until the internet connection is restored.

To better understand the consequences of this behavior on the actual current consumption, each scenario was reproduced, and current measurements were carried out using a digital oscilloscope with a bandwidth of 350 MHz, an ADC (Analog to Digital Converter) resolution of 12 bit, and a maximum sample rate of 1.25 GS/s. To measure the sensors' current, a probe with a bandwidth of 50 MHz and a minimum sensitivity of 10 mA/div was exploited. Results are shown in Figure 3.2. In Figure 3.2a, the device settles at an almost constant current absorption of 14.2 mA while waiting for the network connection, with an average 71.4 mA current while searching for a Wi-Fi connection. In Figure 3.2b, phases in which the system is not using the radio module (during reboot) with a 101 mA peak current absorption can be identified, as well as a 74.3 mA peak current when the Wi-Fi connection succeeded and tried to establish the connection to the cloud. From the analysis of those data, a total mean current of



(a) (b)

Figure 3.2: Current absorption measurements: (a) scenario with missing Wi-Fi network connection; (b) scenario with sensor connected to Wi-Fi but without internet access.

17.3 mA can be extracted for the first scenario and a total mean current of 34.5 mA for the second one. This behavior can significantly degrade sensors' energy performance. To overcome this problem, and hence to extend battery lifetime, the device is forced to follow a duty-cycled sequence of searching and sleeping phases, as shown in Figure 3.3. The device is allowed to check for the presence of the Wi-Fi network or the Internet connectivity for a maximum period of 10 s. After this time, if the task fails, the system enters in a HIBERNATE sleep mode for 60 s to save energy and preserve battery charge. After the sleeping phase, a new search phase begins. This

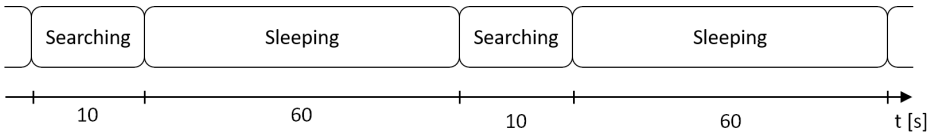
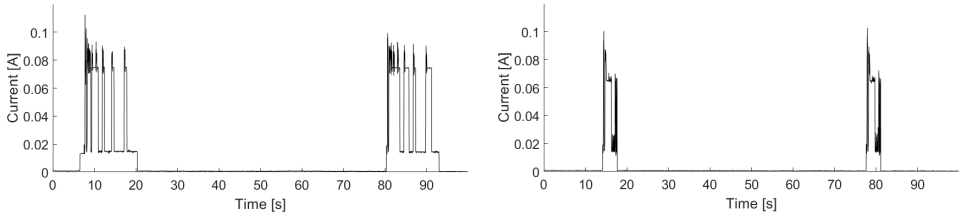


Figure 3.3: Working phases for both Wi-Fi and Internet absence scenarios.

behavior has been tested and compared with the standard one for the same scenarios. The new current measurements are shown in Figure 3.4. After a larger initial current



(a) (b)

Figure 3.4: Current absorption measurements of sensors running with the introduced sleep periods: (a) scenario with missing Wi-Fi network connection; (b) scenario with sensor connected to Wi-Fi but without internet access.

peak needed by the device to boot up, some current peaks are experienced during the Wi-Fi/Internet check phase and they reach the minimum in the sleep phase. In this case, the total mean currents over the period are 7.3 mA in the first scenario and 3.1 mA in the second one. Due to the unpredictability of these situations, the advantage in terms of battery lifetime is not quantifiable; however, with this method, the system is able to reduce the energy consumption by 42% in the case of Wi-Fi network absence and by 91% in the case of Internet connectivity absence. Hence, this behavior has been adopted for all sensors, and the actual battery lifetime during real operation has been evaluated. As stated before, the sensors' energy performance was evaluated in both laboratory and home environments. All sensors were powered with parallel-series of four AA LR6 alkaline batteries. In the laboratory test, due to the different activity profiles expected in a public environment, only the standby power consump-

tion was tested. The only signals sent were periodic (15 min) keep-alive messages. In the real home environment, instead, the sensors were stressed during the day by three different members of a family. After two months of testing, only the toilet sensor ran out of battery: For the other sensors, the laboratory set shows a residual charge of about 46%, while in the home environment this value is lower at 34%. It is worth considering that the battery lifetime in a real home environment strongly depends on the actual interaction with the sensors (e.g., the Magnetic contact can be stressed by more than one user, affecting the expected daily events and the sensor energetic performance). Moreover, as expected, the toilet sensor has a higher energy consumption because of the usage of LPDS mode and due to the higher power consumption of the sensing element. For this reason, the toilet battery lifetime was 30 days.

3.2 System Performance Analysis

To evaluate the system accuracy, sensitivity, and specificity, one user manually took note of his interactions with the sensors for the two months of tests. These data were compared with the events generated by the sensors and automatically collected into the cloud platform. This analysis could effectively be done, in this scenario, only with the chair sensor, in fact, data for the Magnetic contact, Toilet, and PIR sensors cannot be univocally assigned to a specific person, in an environment populated by more than one end-user. To resolve this situation a user identification mechanism should be considered [58, 102]. In the case of the chair sensor, a total of 218 events were detected as follows: 99 True Positives (*TP*), 109 True Negatives (*TN*), 0 False Positives (*FP*), and 10 False Negatives (*FN*). Then, the accuracy, sensitivity, and specificity are carried out [103]:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} = 95\% \tag{3.1}$$

$$Sensitivity = \frac{TP}{TP+FN} = 91\% \tag{3.2}$$

$$Specificity = \frac{TN}{TN+FP} = 100\% \tag{3.3}$$

where *TP*, *TN*, *FP*, and *FN* are defined as reported in Table 3.1. The results pre-

Symbol	Position of the User	Where the User is Identified
TP	User seated on the chair	User detected as seated on the chair
TN	User not seated on the chair	User not detected as seated on the chair
FP	User not seated on the chair	User detected as seated on the chair
FN	User seated on the chair	User not detected as seated on the chair

Table 3.1: Definition of True Positives (*TP*), True Negatives (*TN*), False Positives (*FP*), and False Negatives (*FN*).

sented show that the sensor is definitely capable of providing occupancy information in a real context. In [101], the reported averaged accuracy for a resistive strain gauge was 94%. This value is compatible with these results and a safe conclusion is that the performance of the sensors is attributable to the resistive pressure pad. Nowadays, the pressure pad is the best choice due to its commercial availability and because it does not require specialized installation. Better performance could be obtained by replacing the sensing element, without affecting the system architecture. Some elaborations on the data acquired from all the sensors have been performed to demonstrate the possibility of identifying patterns in the user's activity. For each sensor, two different cases have been considered:

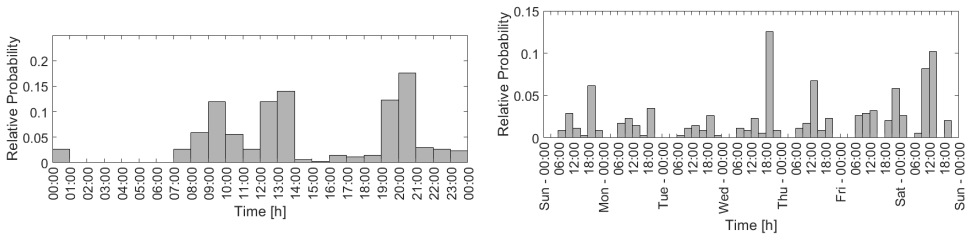
- Daily activity: the distribution of the user-sensor interactions during a 24 h period was analyzed. In Figure 3.5a, Figure 3.6a, Figure 3.7a and Figure 3.8a the probability of an event in a certain hour of the day is plotted. Values were obtained through the analysis of the activity per hour of every test day. The relative probability was calculated with the equation

$$\nu = \frac{c_i}{N} \tag{3.4}$$

where c_i is the number of events that occurred in the hour, and N is the total number of events observed during the test.

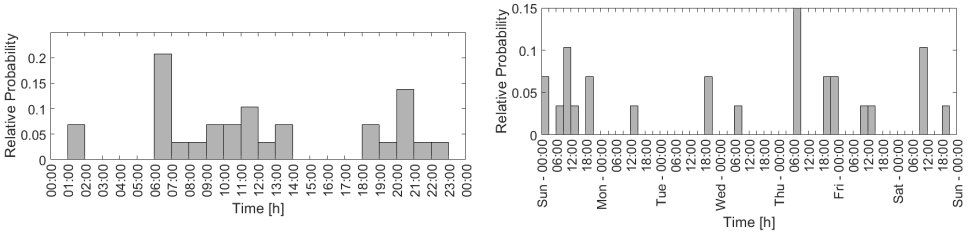
- Weekly activity: the interaction was assessed during a week, counting the events registered in different three-hour intervals in the same day of the week (Figure 3.5b, Figure 3.6b, Figure 3.7b and Figure 3.8b).

Although the aim of this work was not to present a complete behavioral analysis module, simple trend information might be extracted from the results, demonstrating data convenience. For instance, a high lunchroom chair interaction can be seen during meals; the door openings are concentrated during the weekend, when the user is usually not at work. This fact can be confirmed by the PIR weekly graphs, in which the activity distribution shows a higher presence starting from Friday. Looking at the Toilet sensor activity, a periodic usage can be seen around mealtimes and in the early morning after waking up. Moreover, as expected, no sensors recorded activities during night-time, when the user is supposed to be sleeping.



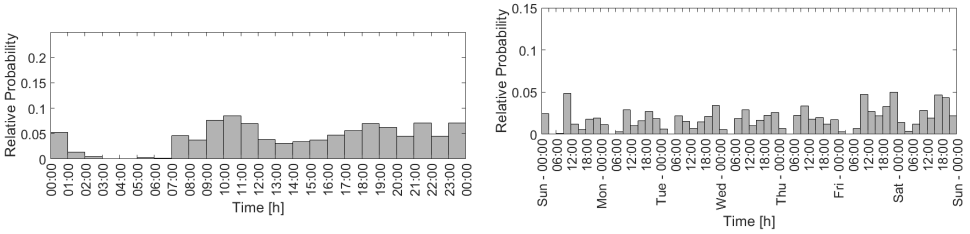
(a) (b)

Figure 3.5: Armchair sensor activity during the day (a) and during the week (b).



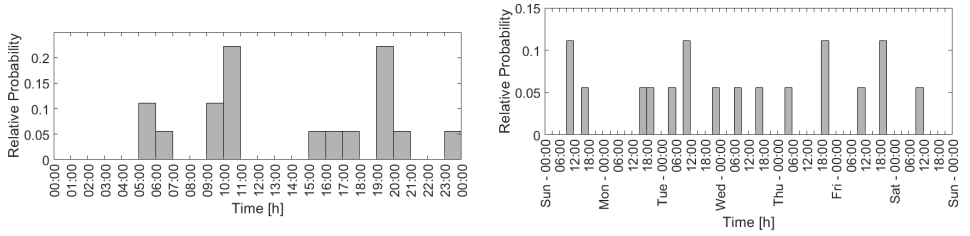
(a) (b)

Figure 3.6: Magnetic contact (door sensor) activity during the day (a) and during the week (b).



(a) (b)

Figure 3.7: PIR sensor activity during the day (a) and during the week (b).



(a) (b)

Figure 3.8: Toilet sensor activity during the day (a) and during the week (b).

Chapter 4

Wearable Sensor for Human Activity Recognition

The addition of a wearable device inside the ecosystem presented in Chapter 2 can introduce behavioral analysis difficult or impossible to perform with environmental sensors alone [23, 24]. The human activity recognition (HAR) plays here an important role: it is indeed recognized that an active lifestyle is the basis for a healthy life [104]. Therefore, users' lifestyle can be assessed by monitoring the amount of daily activity and, eventually, building-up a behavioral model which in turn can be useful for the early detection of anomalies possibly relevant to well-being [104]. Moreover, to construct a precise behavioral profile, it is of utmost importance to accurately assess the type of the user's activity (e.g. walking, climbing stairs up/down, etc.). For example, a user could continue to move regularly (e.g. walking) but starting to avoid more difficult movements (e.g. climbing stairs). This behavior could signal an increase in fatigue, which may indicate a possible deterioration in the health conditions worthy of further study.

Therefore, a first prototype has been developed at the D2Lab as a platform to experiment and test with the new device.

4.1 Wearable Sensor Prototype

The prototype data acquisition system is composed by a Wi-Fi wearable sensor sending collected data to the cloud service in an IoT compliant vision. The prototype of the wearable sensor (Figure 4.1) is based on the MPU9250 integrated inertial measurement unit (IMU, with a 3D accelerometer, gyroscope, and magnetometer) connected to a development board (LaunchPadXL board) including the CC3200 system-on-chip (SoC) by Texas Instruments, which integrates the ARM Cortex-M4 MicroController Unit (MCU). It features a 32-bit architecture at 80 MHz clock and a network pro-

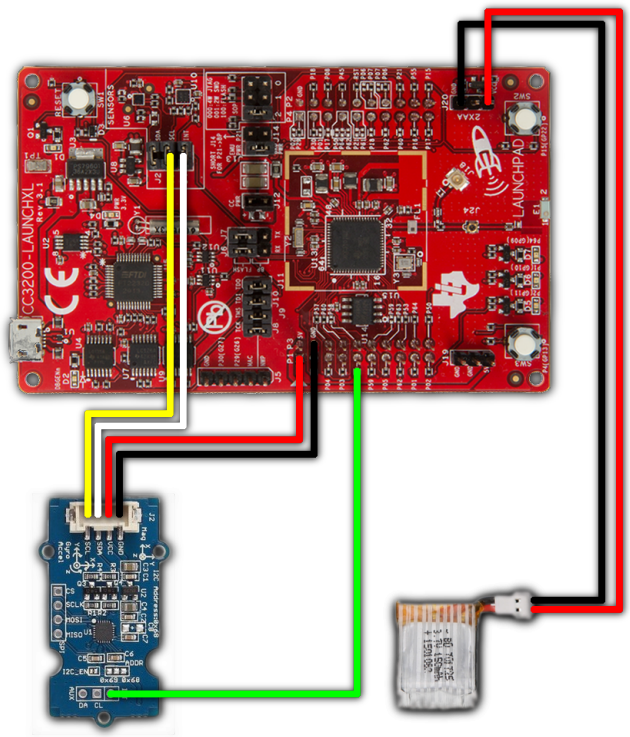


Figure 4.1: The WiFi wearable sensor including the LaunchPadXL board and the inertial measurement unit (IMU) board.

cessor compliant with the IEEE 802.11b/g/n network protocol radio. From the er-

gonomic point of view, a more compact board redesign is planned.

The sensors' full-scales are programmable and a sensing acceleration up to ± 8 g and a magnetometer sensitivity of ± 4800 μT have been set, while the gyroscope was set on a full-scale range of ± 250 °/s. All the sensors have been configured with a sampling rate of 50 Hz. The physical sensor output consists of an ordered list of values (3D output from accelerometer, gyroscope, and magnetometer) corresponding to signed 16-bit values per axis for a total of 18 bytes.

A key issue in device usability is the network configuration procedure: the wearable sensor is directly connected to the Internet through the Wi-Fi home router. The commissioning procedure is very simple and relies on Wi-Fi Protected Setup (WPS) standard: the user is asked to push a button on the device, and a single LED signaling pattern has been implemented to provide feedback. No technical skill is required, so that the user himself could manage the installation procedure. Data are sent to the online Watson IoT platform, inside the IBM Bluemix cloud services, via MQTT protocol with a Quality of Service (QoS) equal to 2: this ensures the necessary reliability to the transmission process, since the protocol itself guarantees that the message is received by the broker once and once only. The payload of the message is a string in a JSON format, reporting sensors' status. The device firmware, nevertheless, is suitable for connecting to other platforms as well (e.g. Amazon AWS, Microsoft Azure, Thingspeak, etc.). Each wearable device features its own unique ID, so that the association to the cloud environment can be managed by the service provider; this results in a truly "plug-and-play" approach.

The collected data are in general not significant in themselves, but they need interpretation to infer meaningful information (e.g., information about user actual activity such as walking, sitting, etc.). We may think to transmit every sensor sample to the cloud, for subsequent processing and interpretation. This implies that the radio section, the most energy-hungry part of the sensor, is always active reducing the battery lifetime. In order to prove the validity of the whole approach, sensor energy consumption has to be considered. In order to account for actual use scenarios, a low capacity battery (Li-Ion 4.2 V, 500 mAh) has been used (battery capacity is limited due to ergonomic constraints: size and weight).

The analysis of human motion features requires a sampling rate as high as 50 samples per second [25]: assuming a sampling frequency of 50 Hz, a full 9 degrees-of-freedom datum sent as soon as sampled and a battery of 500 mAH, it can be demonstrated [89] that a lifetime shorter than 8 hours could be obtained. This prevents such an approach to be actually usable in a real-life context given the high power absorption of the Wi-Fi radio in a continuous streaming configuration.

For this reason, the prototype was not conceived for daily battery operation. In fact, to guarantee a Wi-Fi working behavior comparable to low-power protocols (e.g. Zigbee), special design techniques are needed, which include careful management of the on-off cycles of the radio module [22]. Hence, IMU's data stream can not be sent directly to the Internet cloud platform in daily usage, but it needs to be processed by an algorithm inside the device to have a lower data throughput at the output.

4.2 The Human Activity Dataset

The wearable CC3200 Wi-Fi platform has been used to build a dataset of the activities to be used in the design of the algorithm to be employed onboard for HAR. The data collection phase involved 15 subjects (12 males and 3 females aging between 25 and 50 years) repeating the same test 9 times on different days. For testing purposes, two bytes have been added to the IMU samples, indicating the identification number of the user, and a code related to the activity actually performed. The latter has been obtained by asking the user to follow a specified protocol during the test (i.e. the activities sequence) and to press a button available on the wearable sensor at every change in the activity. The push of the button triggers a counter in the device firmware: every activity the subject was performing was then identified with the corresponding counter number. No noise filtering was carried out on the data. In details, Table 4.1) reports the total set of activities of the dataset, and, in Table 4.2 the activities sequence for a single user. The resulting dataset is a collection of the raw IMU output (triaxial accelerometer, gyroscope, and magnetometer values) stored in the form of timeseries, as shown in Figure 4.2.

Activity ID	Activity
1	Walking
2	Stand
3	Sitting-down
4	Stay seated
5	Standing-up
6	Running
7	Climbing stairs down
8	Climbing stairs up
9	Lie-down

Table 4.1: Activity set.

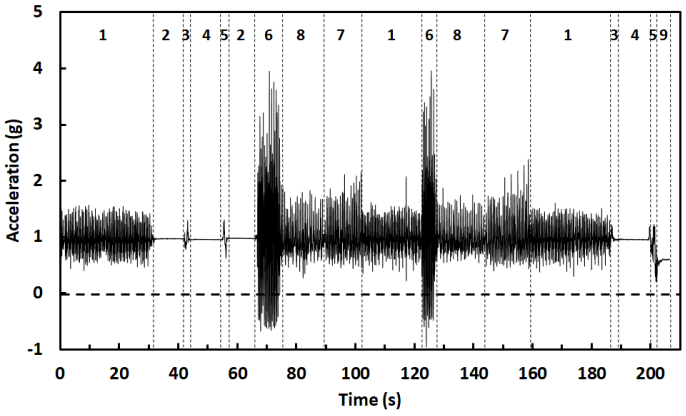


Figure 4.2: Example of the accelerometer x-axis reading. The activities performed during data acquisition have been highlighted, reporting the corresponding activity ID.

Counter	Activity ID	Activity
1	1	Walking
2	2	Stand
3	3	Sitting-down
4	4	Stay seated
5	5	Standing-up
6	2	Stand
7	6	Running
8	8	Climbing stairs up
9	7	Climbing stairs down
10	1	Walking
11	6	Running
12	8	Climbing stairs up
13	7	Climbing stairs down
14	1	Walking
15	3	Sitting-down
16	1	Stay seated
17	1	Standing-up
18	1	Lie-down

Table 4.2: Activity sequence for each acquisition.

4.3 Algorithm

The dataset presented in the previous Section has been used in collaboration with the Computer Engineering research group of the University of Parma to design a deep learning, Neural Network (NN) algorithm (Convolutional Neural Network [105]).

In parallel, the work of this thesis focused on the hardware implementation of a core function which can be employed both as a Support Vector Machine (SVM) algorithm and/or as the activation function of a NN. As shown in Figure 4.3, a NN algorithm is composed of *nodes* organized in *layers*. Each node features an *activation function* i, h, or o which, as proved by several works [106, 107], can be customized to achieve the desired results.

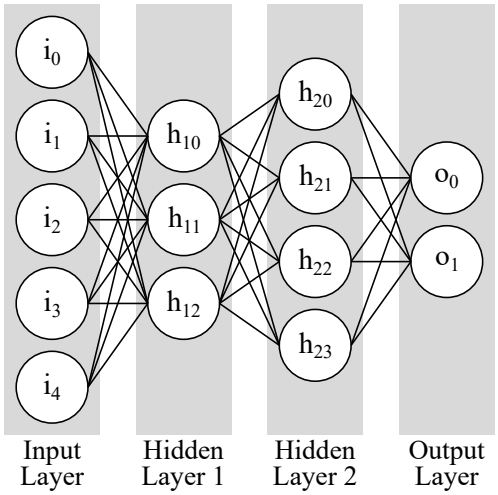


Figure 4.3: Example of a neural network with two hidden layers and: five nodes at the input layer, three nodes at the first hidden layer, four nodes at the second hidden layer, and two nodes at the output layer.

For the purpose of this work, the implementation of the core in a full SVM is important because it allows for gathering information also on the implementation of the core as a node of a NN for further evaluations on the work of [105].

4.3.1 Training and Inference Computations

In general, machine learning algorithms are composed of a training and an inference phase, with the former to be considered as the most computationally expensive since it needs to process the entire (typically wide) training set. For this reason, the machine learning algorithm to be employed for HAR is conceived to have the training phase on the cloud and the inference phase on the wearable device. This working principle allows the system to fit for any new user with the following procedure. The first time the user wears the device, it is configured in *training mode* and she/he is guided through a sequence of the 7 activities to be performed (e.g. Table 4.2). During the training phase, the collected raw IMU data are sent to the cloud platform, which hosts the training algorithm and generates the coefficients for the inference phase. The coefficients are then downloaded and saved into the wearable device, which, from now on, is in *inference mode*, executes the inference algorithm, and sends the results to the cloud for the behavioral analyses.

This enables for keeping the wearable device compact and dedicated to the inference phase, which is the most frequently required compared to the lump sum training phase.

4.3.2 Core SVM Selection

The selection of the core SVM algorithm has been performed with the MathWorks® Classification Learner® tool, in which the training results of the decision tree, discriminant analysis, SVM, KNN and naive Bayes supervised classification learners [108] have been tested and compared. According to classification learning theory, a set of features of the dataset samples has been calculated on the dataset of Section 4.2 to be used as the input of the algorithm. In particular, the features employed are the 9 statistical parameters reported in Table 4.4 and calculated over a window of 1.28 s (which means 64 samples at 50 Hz for each sensor), as found in literature [25]. Since the IMU has 3 sensors with 3 axes each, the total number of input features for the algorithm is 81.

The dataset has been split to reserve the 70% to the training set and 30% to the

	Feature
1	Average
2	Mode
3	Median
4	Standard Deviation
5	Variance
6	Minimum
7	Maximum
8	Root Mean Square (RMS)
9	Interquartile Range (IQR)

Table 4.3: Features used as the SVM input.

test set, and a training session has been issued for the interesting algorithms, i.e. the SVM family ones. As shown in Table 4.4, results show the cubic SVM as the best performing algorithm for this application.

Algorithm		Accuracy
Family	Version	[%]
SVM	Linear	80.9
SVM	Quadratic	90
SVM	Cubic	91.7
SVM	Fine Gaussian	80.1
SVM	Medium Gaussian	88.6
SVM	Coarse Gaussian	78.1

Table 4.4: Comparison of the training of the SVM algorithms.

The selected Cubic SVM algorithm has been then refined by appending a post-processing phase to exclude incompatible activity sequences. The applied rules follow the matrix of Figure 4.4 and allowed for a slight increase in the accuracy value to 93.2%.

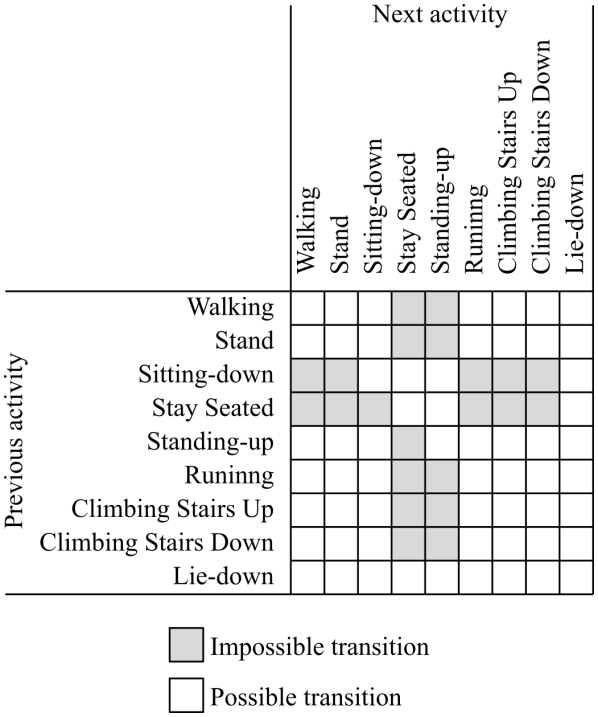


Figure 4.4: Matrix identifying the possible/impossible transitions to be used in the post-processing algorithm phase.

4.3.3 Core SVM Analysis

The output of the Classification Learner® has then been analyzed to identify the basic elements to be used in the hardware implementation phase.

The first thing which has been investigated is the approach used by the Classification Learner ® to solve the multiclass problem, which has been found to be the *one-to-one* (Section 1.1). This means the system makes use of 36 binary classifiers (called *Classification Learners*) to produce 36 output labels +1 or -1. Then, such outputs are merged in the last phase, called *Final Evaluation*, which takes care of producing the final output of the system, i.e. the activity the user is performing. The

equation defining the model is

$$\mathbf{d} = \left| \sum_{i=1}^{36} (1 - y_i \cdot CodingMatrix_{ij}) \right|, \tag{4.1}$$

where y_i is the output of the i-th binary learner, *CodingMatrix* is a 36×9 matrix of weights calculated during the training phase, and $\mathbf{d}$ is the 9-element vector holding the result, i.e. a score value indicating how much a certain class is likely to be assigned for the input. Hence, the final output label (i.e. a number from 1 to 9 according to Table 4.1) is the index of $\mathbf{d}$ where the minimum value is stored:

$$n = index_{\mathbf{d}} \left[\mathbf{d} = min\left(\mathbf{d}\right) \right], \tag{4.2}$$

where n is the final output of the system, i.e. the activity the user is performing.

A deep look at the binary learners was also required for the next hardware implementation. This part of the system is characterized by 36 similar objects (described by Equation 1.2) to which a polynomial cubic kernel is applied (Section 4.3.2):

$$\begin{cases} y = sign(\langle \mathbf{w}, k(\mathbf{x}, \mathbf{w}') \rangle + 1) \\ k(\mathbf{x}, \mathbf{w}') = (\langle \mathbf{x}, \mathbf{w}' \rangle + 1)^3 \end{cases}. \tag{4.3}$$

By merging all the equations of the system, it is possible to write the equation describing a single binary learner as

$$y = sign(\langle \mathbf{w}, k(\mathbf{x}, \mathbf{w}') \rangle + 1) = sign\left(\sum_{i=1}^{81} \left(\sum_{j=1}^{N} \left(x_j \cdot w'_j \right)^3 + 1 \right) \cdot w_i + 1 \right), \tag{4.4}$$

where N is the the number of support vectors calculated in the training phase, which changes for each binary learner.

4.3.4 Features Calculation

As described in Section 4.3.2, a full SVM algorithm does not take as input the raw data of the IMU introduced in Section 4.1 but needs a preprocessing phase to calculate the *features* of Table 4.3.

This is done by applying the following equations at each axis of the accelerometer, gyroscope and magnetometer:

$$
\begin{aligned}
&\text{Average:} && avg = \frac{\sum_{i=1}^{N_S} X_i}{N_S} \\
&\text{Mode:} && mode = \text{most frequent sample value} \\
&\text{Median:} && median = \frac{X_{sorted}\left[\frac{N_S}{2}\right] + X_{sorted}\left[\frac{N_S}{2}+1\right]}{2} \\
&\text{Standard Deviation:} && std = \sqrt{\frac{\sum_{i=1}^{N_S}(X_i - avg)^2}{N_S - 1}} \\
&\text{Variance:} && var = std^2 \\
&\text{Minimum:} && min = \min(\mathbf{X}) \\
&\text{Maximum:} && max = \max(\mathbf{X}) \\
&\text{RMS:} && RMS = \sqrt{\frac{\sum_{i=1}^{N_S} X_i^2}{N_S}} \\
&\text{IQR:} && IQR = Q_3 - Q_1 = X_{sorted}\left[3 \cdot \frac{N_S+1}{4}\right] - X_{sorted}\left[\frac{N_S+1}{4}\right]
\end{aligned}
\tag{4.5}
$$

where $\mathbf{X}$ is the input vector of N_S IMU samples (which is 64, as mentioned in Section 4.3.2) and $\mathbf{X}_{sorted}$ is the same input vector with the elements sorted in ascending order.

The result is an input features vector $\mathbf{x}$ of 81 elements, as shown in Figure 4.5.

4.4 Hardware

The state-of-the-art analysis of 2.2 highlights a fragmented scenario, in which a high number of different solutions have been designed for HAR.

The first conclusion which can be drawn is an inverse proportionality in the accuracy and the number of classes to be recognized, an aspect usually related to the complexity of the algorithm employed. In fact, most of the proposed systems are implemented on an end-device that needs high-speed computation to be able to catch all the human movements under study. Since the final platform is usually an embedded device that must be worn or, at least, near to the user, it usually has poor computing power compared to existing high-end devices. Moreover, such devices are often

	Average	Mode	Median	Standard Deviation	Variance	Minimum	Maximum	RMS	IQR
Accelerometer X	1	2	3	4	5	6	7	8	9
Accelerometer Y	10	11	12	13	14	15	16	17	18
Accelerometer Z	19	20	21	22	23	24	25	26	27
Gyroscope X	28	29	30	31	32	33	34	35	36
Gyroscope Y	37	38	39	40	41	42	43	44	45
Gyroscope Z	46	47	48	49	50	51	52	53	54
Magnetomter X	55	56	57	58	59	60	61	62	63
Magnetomter Y	64	65	66	67	68	69	70	71	72
Magnetomter Z	73	74	75	76	77	78	79	80	81

Figure 4.5: Representation of the features vector **x**. The numbers are the indexes of the elements inside the vector.

battery-powered and require careful power management for energy saving. These constraints force the designers to rely on solutions based either on low-performance algorithms or on low-performance hardware. The final effect is the same: a decrease in the accuracy or in the number of classes of the application.

Another aspect highlighted in the research is the polarization of the devices employed for the applications, which are all based on a MicroController Unit (MCU). If on the one hand these solutions offer the best trade-off between cost, power-saving, and design easiness (as opposed to ASICs, for example), on the other hand it may be the bottleneck on the application performance.

This is the reason why, in this work, a Field Programmable Gate Array (FPGA) was selected to implement the system. In recent years, FPGA producers have started to focus on costs and power reduction of their products because of the incoming market demand [109], and this can be an opportunity to explore new solutions. If on the one side the FPGA design faces an increase in complexity due to the hardware

development, on the other side it gives us the possibility to create optimized and potentially better performing architectures compared to traditional MCUs. All these aspects may allow hosting more refined and complex algorithms to keep high both the accuracy and the classes to be recognized.

4.4.1 Support Vector Machine Modeling

The development of the SVM algorithm has been carried out relying on the model-based design described in Section 1.2, and the MathWorks® suite has been exploited for this purpose. In detail, Simulink® has been used to design, prototype, simulate, and test the algorithm used for HAR, and the HDL Coder® tool has been used to automatically generate and test the VHDL code.

The resulting flow is reported in Figure 4.6, where the basic elements of the features computation (Section 4.3.4) and the SVM (Section 4.3.2) are depicted.

After having identified the proper algorithm (Section 4.3.3), a first modeling phase has been performed. In this step, the mathematical algorithm has been first designed to validate the numerical behavior of all the involved parts, and then, in a second modeling phase (Section 4.4.2), a more hardware-friendly model has been designed. This allowed for performing the code generation step through the HDL Coder tool and to get a VHDL code to be used for the implementation. The last phase was to perform the timing simulation and implementation on the target FPGA.

The result of the first modeling step is the block diagram depicted in Figure 4.7, where the vector of the 576 elements (64 samples $\times$ 3 sensors $\times$ 3 axis) $\mathbf{X}$ is used as input, and the output is the predicted class n.

Each binary learner conforms to the model architecture of Figure 4.8, in which, after a normalization phase, the kernel and decision functions are executed.

The details of the underlying blocks are reported in the following Subections.

Features

According to Section 4.3.4, the features calculation relies on the system of Equations 4.5. The resulting modeled architecture is reported in Figure 4.9, in which every

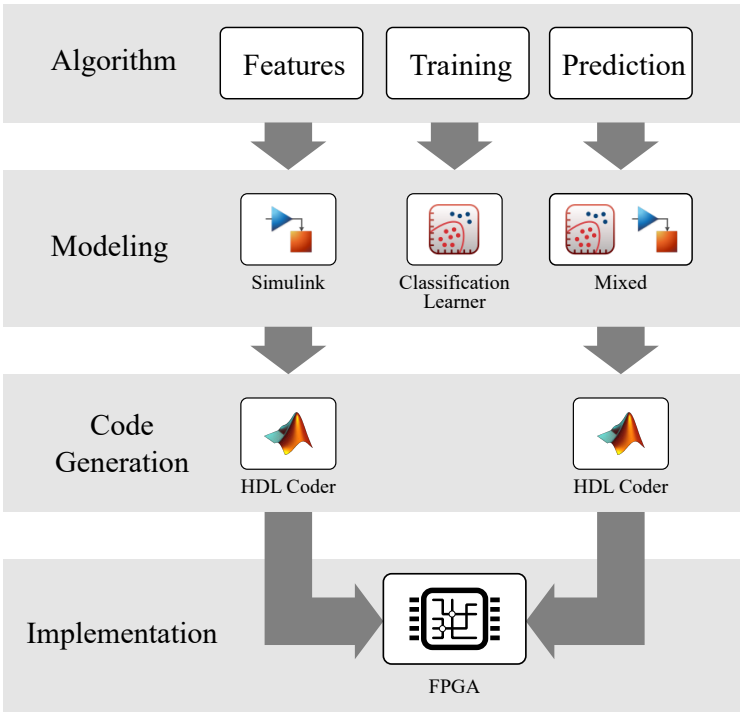


Figure 4.6: Model-based development workflow.

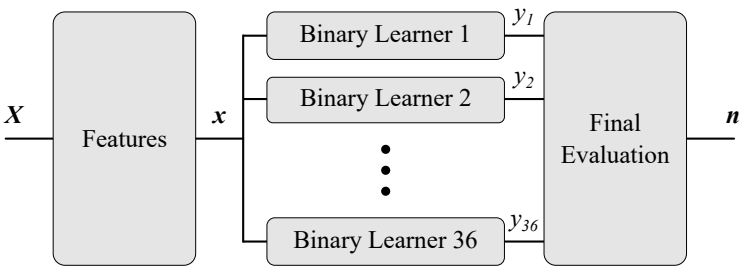


Figure 4.7: Full activity recognition algorithm model.

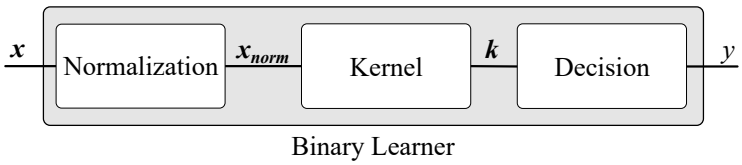


Figure 4.8: Single binary learner model.

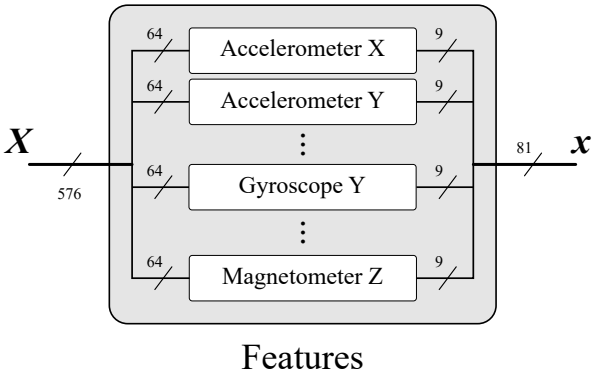


Figure 4.9: Overview of the features calculation modeling.

channel is processed with a dedicated block.

As shown in Figure 4.10, the system takes as input the 576 elements, which are then split into its 64 samples and given to the proper processing block. The blocks dedicated to the elaboration of the single channels have all the same design, reported in Figure 4.10 using the accelerometer's X-axis as reference.

Binary Learner

The binary learner system is composed of the parts of Figure 4.8, which are replicated 36 times to perform the one-to-one classification. According to the theory of Section 1.1, it holds a first normalization phase, the kernel function, and then the decision phase. In the following paragraphs, each block is presented in detail.

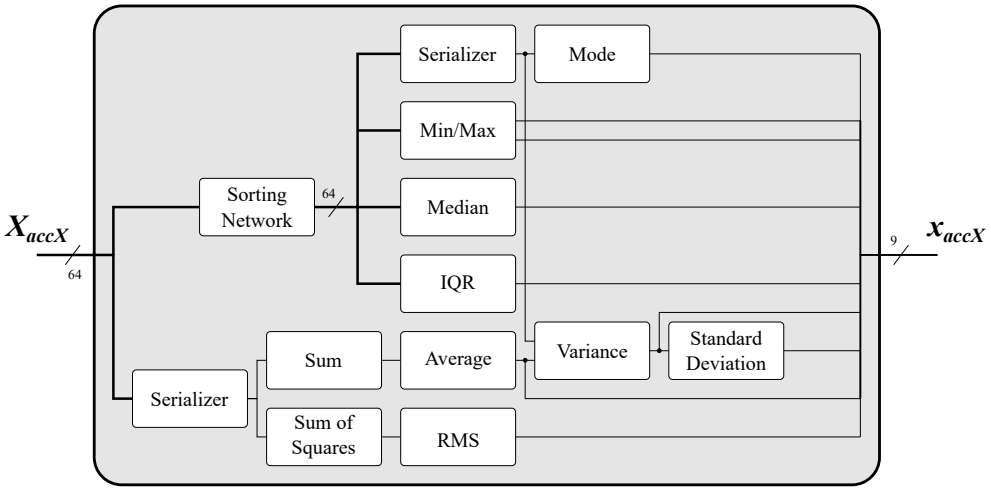


Figure 4.10: Modeling example of features computation for the accelerometer's X axis, i.e. one of the 9 input channels.

Normalization The normalization part is used to limit the out-of-scale occurrences which may be introduced by the kernel function, as explained in Section 1.1.1. The modeling of this part has been carried out by implementing Equation 1.3, hence producing the diagram of Figure 4.11.

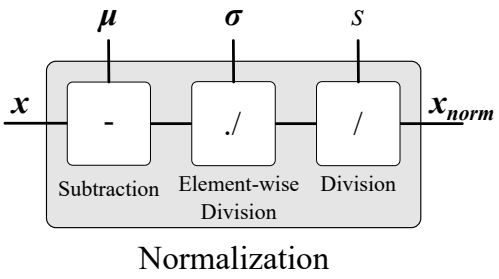


Figure 4.11: Modeling of one Binary Learner normalization phase.

Kernel In Figure 4.12, the modeling of the kernel function is depicted. It is worth noting here that the result vector $\boldsymbol{k}$ has a length N which is different for every binary learner. In fact, considering that the transformed support vectors $\boldsymbol{w'}$ represent a matrix of $81 \times N$ elements and that the X_{norm} vector is of 81 elements, the result of the first element-wise multiplication is again a matrix of $81 \times N$ elements but the subsequent summation enables a switch to the N dimension. Hence, from here on, the size of the bus is of N elements straight to the output $\boldsymbol{k}$.

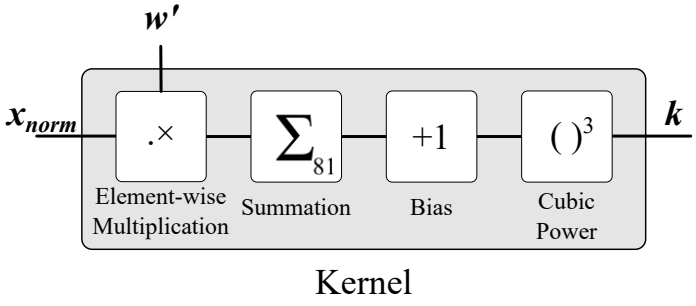


Figure 4.12: Modeling of one Binary Learner kernel phase.

Decision The Decision phase is depicted in Figure 4.13. Similarly to the kernel part, here is another switch from the dimension N to 1 because of the summation, which this time collapses all the vector into a single value. After the sign function, the output has a value of +1 or -1, according to the binary classification theory of

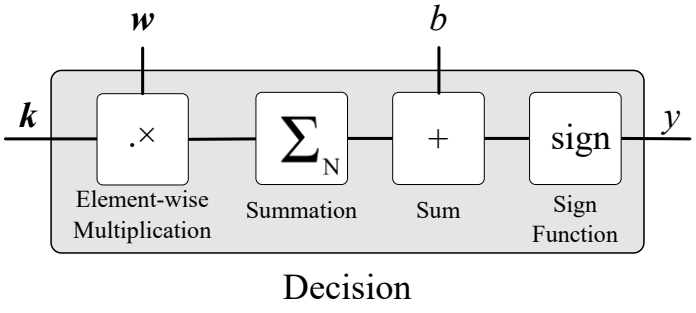


Figure 4.13: Modeling of one Binary Learner decision phase.

Section 1.1.

Final Evaluation

The Final Evaluation block takes as input a 36-element vector holding all the +1 or -1 outputs of the 36 binary learners. As for the Kernel and Decision blocks, here the summation is responsible for another switch in the dimension of the bus. Since the *CodingMatrix* is a coefficients matrix of 9×36 elements, the result of the element-wise multiplication is again a 9×36 matrix. However, the summation performs the sum of the rows of the matrix, giving a 9-element vector as the result.

This vector is then given to the *idx* function, which looks for the index of the minimum element and produces the output *n*, i.e. the label of the activity the user is performing.

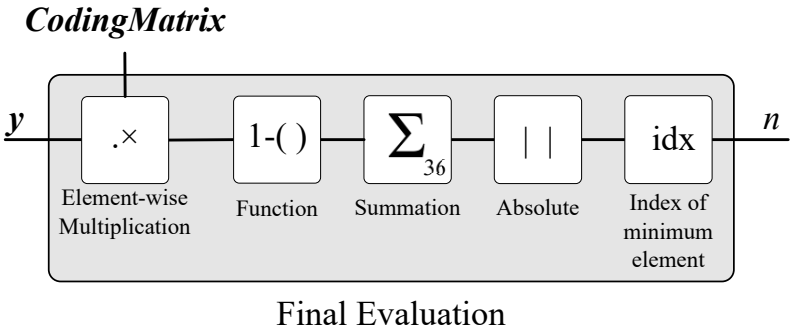


Figure 4.14: Modeling of the Final Evaluation phase of the system.

4.4.2 Support Vector Machine Hardware Design

After testing the functionality of the aforementioned model in Simulink, the next step was to define a hardware-friendly model to be converted in VHDL code with the HDL Coder. Since the target now is the final FPGA implementation, it is unlikely to have a system like the one modeled right now to be hosted in hardware.

The first obstacle would be the size of the resulting architecture. In fact, as shown in Figure 4.7, 36 identical elements should be implemented to perform the task, each

of them with a series of mathematical operations which require non-trivial architectures (e.g. the divisions of the Normalization phase). Moreover, as will be discussed in detail in Section 4.5, the system has been designed to work with an IEEE 754 32-bit standard to ensure a high output numerical quality. Since the complexity of floating-point arithmetic is consistently heavier than an integer or fixed-point one, the warning given by the high number of mathematical operations becomes stronger.

The second obstacle would be the memory cells required to host all the coefficients resulting from the training phase, which have a size of 8.3 Megabytes when stored in binary format on a PC's hard disk. This would not be a problem if using a high-end FPGA but, given the size and energy constraints of the application, the landscape here restricts to a low- or mid-end device. For such devices, this requirement is out of specifications [110, 111].

For this reason, the benefits of the model-based design have been exploited to experiment with a new model with the same exact functionality as the previous one, but more hardware-friendly and targeted at FPGA implementation. The overview of the result can be seen in Figure 4.15.

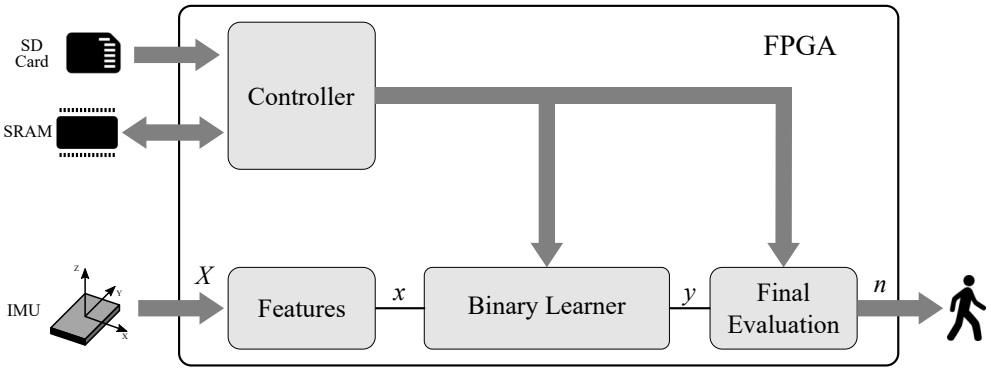


Figure 4.15: Architecture of the final system to be implemented in the FPGA.

The problem of the memory usage has been addressed by introducing a Secure Digital (SD) Card to host and retain the coefficients of the training phase, and a volatile Static Random Access Memory (SRAM) to be used as high-speed storage during normal operations. Moreover, a bus has been introduced to allow the coeffi-

cients to reach the proper areas during the computations of the Binary Learner and the Final Evaluation blocks.

The details of each element are described in the following Subsections.

Memory Controller

As aforementioned, a set of external memories has been introduced to avoid exceeding FPGA's internal resources usage. As shown in Figure 4.16, the system was designed with two types of memories:

- SD Card: it is used as a non-volatile memory to retain the coefficients during power-off. However, it is not suitable for normal operation due to its high read access time.
- SRAM: given its low read access time, it is used as the work memory to feed the Binary Learner and Final Evaluation blocks with the proper coefficients.

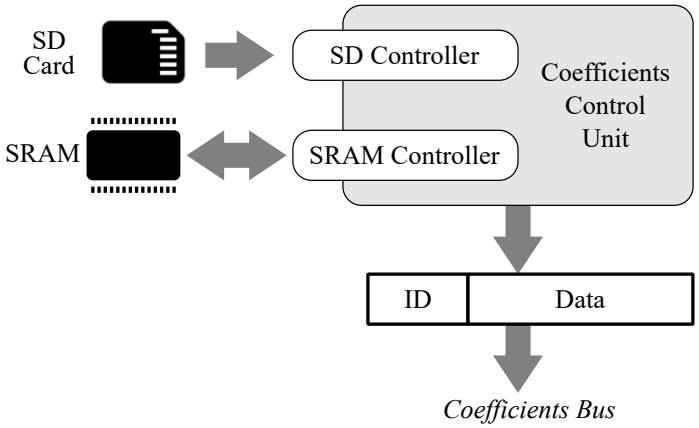


Figure 4.16: Memory Controller architecture.

The controller is used to manage the interactions between the two memories and the *Coefficients Bus*. At startup, the controller pulls the coefficients from the SD Card and writes them in the SRAM to prepare the system. After this phase, the controller

reads the data from the SRAM and pushes it on the Coefficients Bus, which is directly connected to the rest of the architecture.

As will be seen later on, the organization of the coefficients inside the memories is not random and it has been defined to allow the system to operate correctly. Each word of the memory is composed of an *ID* and a *Data* field, as shown in Figure 4.16. The ID is an identification value automatically interpreted by the targets to understand which coefficient is the Data value related to.

Features

The model of the features block engineered for FPGA implementation is reported in Figure 4.17.

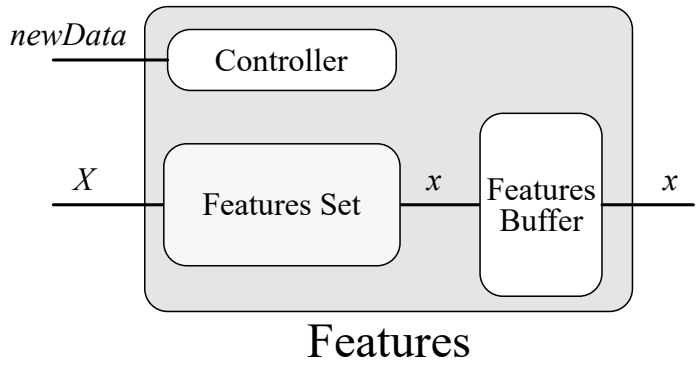


Figure 4.17: Features calculation final architecture.

The first difference from the previous model is the elimination of the 9 parallel blocks, which are here replaced by a single block equivalent to the one of Figure 4.10. At every new input, the *newData* bit signals the Controller to take 64 elements of the 576-element input vector X to compute the features' calculation for that channel. When the 9-feature result is ready, it is written in the first 9 words of the 81-word Features Buffer, and the next 64-element input chunk is used to produce the features for the next channel. After 9 cycles, the Features Buffer is full, and x can be used for the rest of the system.

This solution allows for using a single features-generation architecture in place of 9.

Binary Learner

Similar to the Features part, the 36 Binary Learners are here replaced with a single object iterated 36 times to produce the results. The internal architecture is depicted in Figure 4.18 and it is mostly untouched with respect to the previously modeled version.

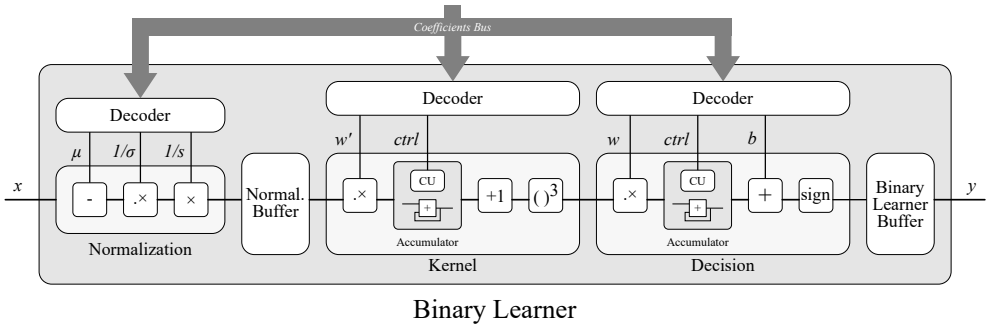


Figure 4.18: Binary Learner final architecture.

The main differences are the Deciders (needed as the interfaces with the Coefficients Bus) and two intermediate buffers holding the partial results of the affected vectors. The summation parts are here converted to accumulators to have more hardware-friendly architectures. Then, in the Normalization phase, the coefficients of σ and s have been pre-processed and stored inside the memory with its inverse. This allows us to replace the division operations with a less expensive multiplication without affecting the rest of the system.

The Decoder blocks are used here to interpret the Coefficient Bus message, and to understand if and where the current coefficient is to be used in one of its underlying blocks. In addition, since the Binary Learner has to be iterated 36 times, the Decoders are also responsible for resetting the buffers and accumulators when a new Binary Learner computation has started. This is done, again, by listening to the Coefficients

Bus to detect when a set of coefficients has switched to a set of the next Binary Learner.

For example, let us imagine that all the 1/s vector elements of the Normalization are encoded in memory with ID = 5 for the Binary Learner 1 and with ID = 12 for the Binary Learner 2 (Figure 4.7). In this case, the first time the Decoder detects an ID = 5 on the bus, it feeds the Data value to the Normalization multiplier and saves the result in the first cell of the Normal Buffer. Then, when a new ID = 5 arrives, the Decoder routes Data again to the multiplier and saves the result in the *second* cell of the Normalization Buffer. After some cycles, the ID on the bus is ID = 12. This time the Decoder senses a new set is started and, in addition to routing Data to the multiplier, it resets the Normalization Buffer and writes the new result in the *first* position, meaning a new Binary Learner computation has started.

One concern here may be: is it safe for the Decoder to flush the Normalization Buffer without knowing if the next block (i.e. the Kernel) has actually processed those data? The answer is: yes, thanks to a proper ID scheduling inside the memory. In fact, if the order in which the IDs are fed to the blocks have the proper sequence, the data can flow from the input to the output without collisions and the Decoders can ignore the state of each other.

Final Evaluation

The Final Evaluation block is reported in Figure 4.19 and its basic elements rely on the same concepts of the Binary Learner.

The *idx* function is modeled as a register updating each time the input is minor then the saved value, and a counter to keep track of the index of the saved element. The result is a number that is the label of the activity the user is performing.

4.4.3 FPGA Implementation

The hardware-friendly modeled architecture was then processed by the HDL Coder tool to generate the VHDL for implementation. After this, the proprietary Integrated Development Environment (IDE) of the target device is required to be able to produce

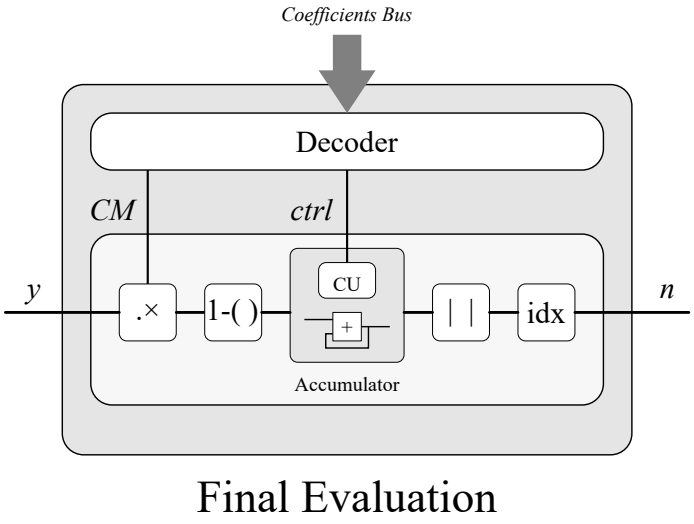


Figure 4.19: Final Evaluation architecture. Here, the *CodingMatrix* is reported ad *CM*.

the bitstream and program the hardware. For this part of the work, an Altera Cyclone IV FPGA was selected as the target device, and the Altera Quartus II IDE was used to interact with the device.

After importing the VHDL code generated by the HDL Coder in Quartus, the implementation workflow has been carried out, and the results are reported in Table 4.5.

The next step was to verify the timing of the circuit, which must fit the requirement of 100 MHz to guarantee the whole system to produce the activity at the output

Parameter	Resources Usage
Logic Elements	88%
Multipliers	100%
Memory	15%

Table 4.5: Implementation results on a Altera Cyclone IV.

in less than 1.28 s, i.e. the time window of the input samples (Section 4.3.2).

For this reason, a timing simulation has been carried out to verify the requirement and the results demonstrated several violations during the processing, which led to wrong output values. To solve this issue, several solutions have been taken into account.

The first one may be to add pipeline stages inside the architecture to break the critical paths and to increase the slack of the circuit. However, this approach introduces additional issues to the system because of the presence of accumulators. This solution has been investigated and addressed in Section 4.5.

The second alternative may be to reduce the overall complexity of the system, for example switching from the IEEE 754 32-bit floating-point arithmetic to fixed-point or integer. However, this leads to a degradation of the numerical accuracy of the system and introduces a trade-off between circuit lightness and numerical quality (hence output accuracy). This solution has been investigated and a search for alternatives has been carried out in Section 4.6.

4.5 Hardware Investigation I: Pipelined Accumulators

A solution identified to solve the timing issues of the system is the introduction of pipeline stages to break the critical paths. However, as aforementioned, this collides with a potential performance loss introduced by the accumulators. In fact, among the SVM blocks employed in the dedicated Simulink design, the accumulator is one of the most frequently used.

The simplest accumulator architecture can be designed by using an adder in which the first input receives the operand element and the second input is the feedback of the output [112]. It is worth noting that general FPGA-based SVM architectures deal with high numeric dynamic ranges; thus, they are based on floating-point arithmetic, as this is the best solution with data with this requirement [113]. A typical approach when dealing with hardware floating-point arithmetic is to introduce pipelined architectures to reduce the critical path timing, potentially increasing the system clock frequency [114]. When used in the simple accumulator architecture

described before, pipelined floating-point adders become critical. In fact, new input should be presented only when the output of the last addition can be fed back to ensure correct operation and avoid data hazards independently of the number and the length of the input vectors [112]. This would limit the applicability of the system, and different accumulator architectures should be individuated. However, when the boundary conditions allow this solution to be exploited, the latency T (expressed in number of clock cycles) of the whole accumulator to process an input vector of n elements is $T = np$, where p is the length of the adder pipeline.

Considering model-based designs and, in particular, the Simulink environment, several accumulator blocks are already available. However, some of them are not suitable for the specific application due to incompatibility with the HDL Coder workflow (such as the Cumulative Sum block) or with the Floating-Point HDL library (as for the *Multiply-Accumulate* block). Some others (*Sum of Elements* and *Matrix Sum*) implement the HDL code as a binary tree or a linear chain of floating-point adders presenting all the input elements in parallel—since the complexity of these solutions grows with the number of inputs to accumulate, the amount of resources used when these blocks are exploited in this field should be evaluated [115]. Hence, in this thesis work, a possible alternative Simulink model for an accumulation circuit based on a floating-point pipelined adder and fully compatible with the HDL Coder workflow is presented.

To select the architecture to be implemented, a review of the state-of-the-art accumulation circuits was carried out. In reference [116], Luo and Martonosi present the architecture of an accumulator in which a floating-point pipelined adder is broken down into its mathematical operations (i.e., the ones involving the sign, mantissa, and exponent) and an internal feedback loop is introduced to embed the accumulation feature. This solution is reported as providing a minimal accumulation latency of $T = p + (n+1) + t_{norm}$, where t_{norm} is the combinational logic delay of the last accumulation part of the architecture. However, the exact latency value cannot be determined a priori because it is strongly dependent on the target hardware architecture [116].

A similar approach is used by Nagar and Bakos in reference [117], in which the

accumulation latency is independent of the hardware implementation, and t_{norm} can be considered as a one clock cycle. However, as for [116], the model-based implementation of this solution implies the additional development of a new adder architecture in order to consider the required modification.

In [115], Zhuo et al. present two main architectures based on standard floating-point adders: the fully compacted binary tree (FCBT) and the single strided adder (SSA). FCBT is an accumulator derived from a binary adder tree in which the first level is replaced by one buffer and a single adder, and the rest of the levels are replaced by an additional adder shared by $\lceil \log n \rceil - 1$ buffers. With the proper control logic, the system can perform the accumulation in $T \leq 3n + (p-1)\lceil \log n \rceil - 3$ for $n < n_{max}$, where n_{max} is the maximum input vector length that the system has been designed to work with. Because two different floating-point adders were deployed, this solution turned out to be undesirable, as it requires large area resources [115].

To overcome this issue and to remove the n_{max} limitation, the SSA architecture has been introduced. This architecture is based on a single adder, two buffers, and a control logic. With this system, the latency has proven to be $T \leq n + 2p^2$.

A different set of architectures are based on the work presented in [118]. Here, an implementation of an accumulator based on a standard pipelined floating-point adder is described. The input data vector is split into two different buffers and, at each clock cycle, one element from each buffer is given to the adder operands. Then, after p cycles, the vector of adder results is split into two halves again, which serve as the new input elements. This procedure is repeated until no other couples of operands are present in the buffers, meaning the accumulation has ended and the result is ready. This architecture was found to produce the accumulation result in $T = (p-1)\lceil \log n \rceil + 3(n-1)$. The main limitation of this system is that only one input vector can be accumulated at a time, resulting in the fact that the subsequent vectors must wait for the current result to be produced before they can be processed.

The time and the resources needed to perform the accumulation are reduced in [119]. Compared to the work presented in [118], the input buffers are substituted by two multiplexers at the input of the adder. One multiplexer can switch between the input vector and a register holding the adder output, and the other can switch

between a constant value and the direct adder output. With the proper control of the multiplexers and the register, the time needed to compute the accumulation is improved for $n>p$. Then, the resulted latency is

$$T = \begin{cases} (p-1)\lceil \log n \rceil + 3(n-1) & ,n \leq p \\ n + (p-1)\lceil \log p \rceil + 4(p-1) & ,n > p \end{cases}. \tag{4.6}$$

From this work, the total accumulation time of this circuits is found to be

$$T = T_f + T_m + T_d, \tag{4.7}$$

where T_f is the time needed for all the input elements to get inside the accumulator (feed phase), T_m is the time needed to process all the partial results given by the couples of adder input operands (merging phase), and T_d is the time needed for the last result to exit the adder pipeline (drain phase). It can be shown that this formula is applicable to every accumulator based on the architecture presented in [119]. Moreover, it can be easily observed that $T_f = n$ and $T_d = p-1$.

In reference [120], an improved control algorithm (i.e., asymmetric method (AM)) for the merging time is presented. In this case, the merging time was found as

$$T_m^{AM} = \begin{cases} n\lceil \log n \rceil - 2^{\lceil \log n \rceil} + n + (k-n)\lceil \log n \rceil & ,n < p \\ p\lceil \log p \rceil - 2^{\lceil \log p \rceil} + p & ,n \geq p \end{cases}, \tag{4.8}$$

which shortened the total accumulation time by $3(p-1)$. In reference [121], a modified AM is proposed, with an improvement for every $n < 2p$:

$$T_m^{MA}(n) = \begin{cases} T_m^{AM}(n) - p & ,n \leq p \\ T_m^{AM}(n) - p + 1 & ,n = p+1 \\ T_m^{AM}(n)\lfloor n/2 \rfloor - D\lfloor n/2 \rfloor & ,p+1 < n < 2p \\ T_m^{AM}(n) & ,n \geq 2p \end{cases}, \tag{4.9}$$

where D is a displacement function that compensates the irregular merging pattern that characterizes the control logic. In reference [122], Tai et al. propose a modified version of [121], introducing the delayed buffering (DB) algorithm, in which the

control logic can further reduce the merging time T_m^{DB} in respect to T_m^{MA} for certain input set lengths:

$$T_m^{DB}(n) = \begin{cases} p\lceil \log n \rceil + 2^{\lceil \log n \rceil} + n - p & , n \leq p \\ p\lceil \log p \rceil + 2^{\lceil \log p \rceil} + n - p & , n = p+1 \\ pL - 2^L + \lfloor G \rfloor - D + 1 & , n > p+1 \end{cases}, \tag{4.10}$$

where L and G are functions of n and p, which, as the D function, compensate for the irregular merging pattern.

In [123], a solution requiring a variable number of adders is presented—this increased the reuse and portability of the accumulator, but with a higher occupied area. For example, for the area-efficient modular fully pipelined architecture (AeMFPA), two adders are required. More recently, reference [124] presented an accumulator circuit that can simultaneously add multiple independent vectors; however, the input buffer size is dependent on the number of the inputs, limiting the portability over different applications. Finally, in reference [125], a more flexible solution is reported—the core of the idea is a new state-based method (SBM) algorithm, a scheduling strategy for buffer management aiming at lower latency and smaller area.

In Table 1, a summary of the performance of the mentioned architectures is reported, along with some practical examples that were computed considering an adder latency of $p = 11$, as the latency of Simulink floating point adder intellectual property (IP), and an input length of $n = 15$, as well as an adder latency of $p = 14$ and $n = 16$ for comparison with the solution reported in [125]. The system presented in [122] offers the lowest latency for the accumulation of an input set of data. In this case, the total accumulation time depends on the input vector length with respect to the pipelined adder latency as expressed in Equation 4.11.

$$T = T_f + T_m^{DB} + T_d = \begin{cases} n + p - 1 + p\lceil \log n \rceil + 2^{\lceil \log n \rceil} + n - p, & n \leq p \\ n + p - 1 + p\lceil \log p \rceil + 2^{\lceil \log p \rceil} + n - p, & n = p+1 \\ n + p - 1 + pL - 2^L + \lfloor G \rfloor - D + 1, & n > p+1 \end{cases}. \tag{4.11}$$

This model was exploited in the proposed model-based implementation and in the SVM kernel. It was fully tested in an FPGA implementation and, to validate the results, it was compared with the simple iterative accumulator solution [112], SBM

Method	sadf		
	Generic	p=11, n=15	p=14, n=16
SSA[115]	$\leq n+2p^2$	257	408
FCBT[115]	$\leq 3n+(p-1)\lceil \log n \rceil$	85	100
AM[120]	$n+p-1+T_m^{AM}$	64	83
MA[121]	$n+p-1+T_m^{(}MA)$	58	71
A2eMFPA[123]	$n+p\lceil \log p+2 \rceil$	81	100
[124]	$n+T_m^{AM}+\lceil p/2 \rceil$	60	104
SBM[125]	Not available	-	75
DB[122]	$n+p-1+T_m^{DB}$	57	71

Table 4.6: State-of-the-art hardware accumulator architectures..

[125], and the built-in *Sum of Elements* Simulink block. Then, the proposed accumulator was used in a model-based implementation of the SVM kernel function.

4.5.1 Model-Based Single-Set DB

The proposed Simulink model is shown in Figure 4.20. To design the proposed model, basic Simulink blocks have been used. However, since in Simulink a specific block modeling an adder with latency is missing, an *Adder With Latency* block has been created as a cascade of an adder and a delay block. This approach also allows us to configure the adder's latency with a customizable value of p. The rest of the architecture features three *Switch* blocks (*R_Switch*, *A_Switch* and *B_Switch*) and three main Logic blocks (*External Signaling Logic*, *Main Control Logic* and *Adder Supervisor Logic*). The *Switch* blocks are used as routing elements and their behavior is equivalent to the Register Transfer Level (RTL) multiplexer element. With this configuration, the *Register* can be shared by both A and B operands. Moreover, as a control logic rule, the input data can only be used as operand A while operand B comes from the feedback path each time the adder output is valid. The *Logic* blocks are subsystems that produce the control signals for the entire architecture. In detail:

- *Main Control Logic*: it is the core control unit of the system. As explained in [122], it controls the data path of the input data stream, the Register and the adder to avoid data collisions and data loss. The detailed operation of the logic is reported in Table 4.7 and an execution example is shown in Table 4.8;

- *External Signaling Logic*: it is the logic dedicated to the management of the *data_last* input flag and to produce the *result_ready* output flag. The output can be considered ready when all the input conditions are verified: *data_last* raised by the user, internal adder pipeline empty (meaning no other operands are to be processed) and last adder result placed in the *Register*. The first condition is evaluated by capturing the user *data_valid* assertion through a Set-Reset (S-R) Flip-Flop (FF), the second is directly given by the *pipeline_empty* signal from the *Adder Supervisor Logic* and the third is evaluated by verifying whether the *R_sel* bus is equal to one. The FF S-R is reset by the *result_ready* signal delayed by one clock cycle (*result_ready'*), so to set the system ready for the next streaming accumulation. The circuit dedicated to this task is shown in Figure 4.21a;

- *Adder Supervisor Logic*: by checking if a new couple of inputs are presented to the adder, it notifies if any data is inside the pipeline. In addition, it signals when a sum operation has been completed and the adder output is valid. The internal logic is shown in Figure 4.21b. The *new_input* bit signal goes high each time a new couple of operands is presented to the adder and it is used as the input of the shift register represented by the FF1, FF2, ..., FFp, with p the length of the internal adder pipeline. When the *sum_valid* bit goes high, p clock cycles are elapsed, meaning the addition result is ready. Moreover, if the *pipeline_empty* bit is low, no new operands have been presented in the last p clock cycles, i.e. the internal adder pipeline is empty.

In Table 4.8, an example of the running algorithm is shown with the internal adder latency configured to be 2 clock cycles.

For simplicity, in this use case, four data elements are read, one every clock cycle, while the adder is a two-stage pipeline operator. At cycle 0, the first element is

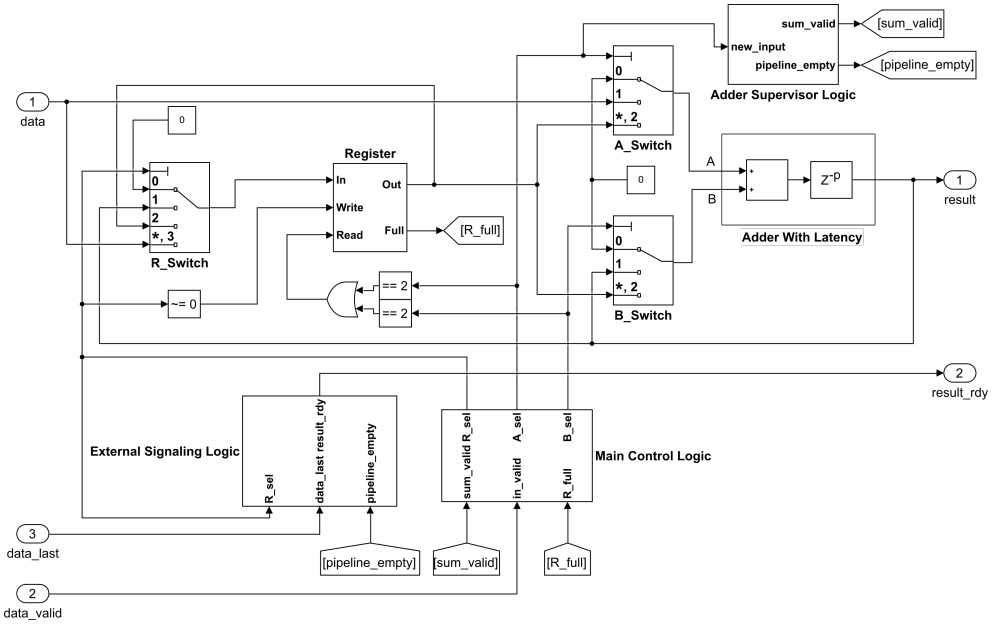


Figure 4.20: The Simulink accumulator model based on the work of [119]. In this example, the system has been configured to model a pipelined accumulator with an adder latency of p clock cycles.

	Condition	Behavior
1	Input valid	Input in register R
2	Adder output valid, data in register R	Adder output fed back to adder input, register R value to adder input
3	Input valid, data in register R	Input directly to adder, register R value to adder input
4	Input valid, adder output valid	Adder output fed back to adder input, Input directly to the adder
5	Input valid, adder output valid, data in register R	Adder output fed back to adder input, Input directly to adder, register R holding data

Table 4.7: Main Control Logic working behavior.

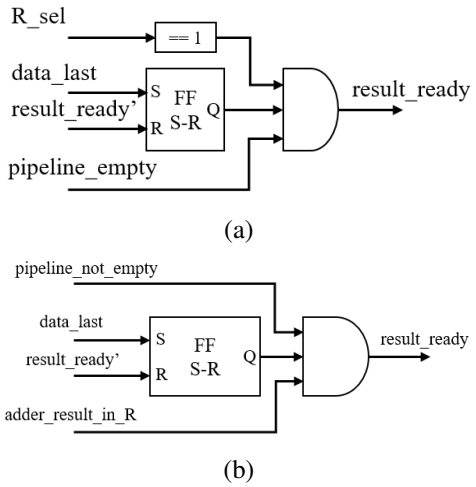


Figure 4.21: (a) External Signaling Logic function; (b) Adder Supervisor Logic function.

Cyc.	Data	A	B	R	Result
0	X_1			X_1	
1	X_2	X_2	X_1		
2	X_3			X_3	
3	X_4	X_4	X_1+X_2	X_3	X_1+X_2
4				X_3	
5		X_3	$X_1+X_2+X_4$		$X_1+X_2+X_4$
6					
7					$\sum_{i=1}^{4} X_i$

Table 4.8: Main Control Logic working behavior.

presented. Since the adder produces a valid output only with a pair of input operands, the element is stored in the *Register*. At the next cycle, a new input data is ready and now the two operands can be pushed in the adder pipeline. The working mode repeats these steps until the first sum is generated by the adder, here at cycle 3. In this situation, the *Register* is already storing a value (X_3), so the control logic pushes into the adder the new incoming input together with the sum just generated. The *Register* is set in a hold state. At cycle 5, when a new couple of data is available, the adder is fed with the value stored in *Register* and the last generated sum. After two clock cycles (i.e. the adder pipeline latency), the final accumulation value becomes valid.

Results

The presented model has been compared with Xilinx Floating-point accumulator Intellectual Property (IP) core for FPGA implementation. To have comparable results, both architectures have been configured to have a total accumulator latency of 30 clock cycles. For the Simulink model, this means using an adder pipeline latency p of 11 clock cycles and an input streaming length n of 5 values, as found by using the DB architecture equation of Table 4.6.

In Figure 4.22, a Simulink example of an input stream of 5 random floating-point values in the range -100 to 100 is reported. As shown, the input flags *data_valid* and *data_last* are attached to the input stream to notify whether the value is valid and the last. After the *data_last* flag has been asserted and the whole system finishes its internal processing, the *ouput_ready* flag is raised for one clock cycle. This notifies the user about the result readiness.

To test the HDL Coder compatibility, a non-target-specific VHDL code generation has been carried out for an architecture based on the 32-bit floating-point format. The generate code has then been imported in Vivado software and, after synthesis and implementation elaborations for a Xilinx Artix-7 XC7A100T-CSG324 FPGA target device, results have been reported in Table 4.9. Both systems perform the same data processing: accumulation of a 32-bit floating-point input stream, with a total latency of 30 clock cycles and an input of 5 streaming values.

As shown, the presented model features lower resources usage than Xilinx IP im-

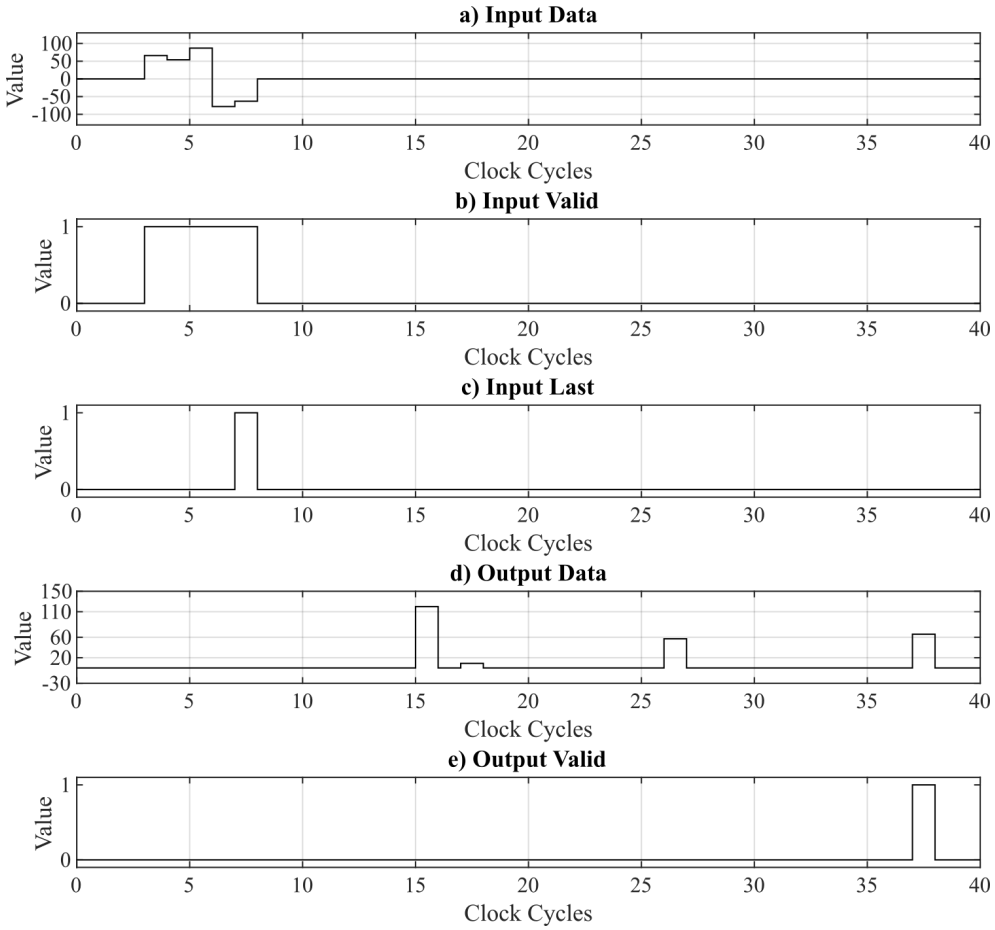


Figure 4.22: Example of an execution of the accumulator model: (a) Input data values; (b) External data valid input signal (*data_valid*); (c) External input signal to notify the last value of the set (*data_last*); (d) Output value (*result*); (e) Internally generated output signal to notify the output is valid (result_*ready*).

	Presented accumulator	IP accumulator
Slice LUTs	635	3275
Slice Registers	723	3067

Table 4.9: Post-implementation resources usage report generated by Xilinx Vivado.

plementation. This result was expected because the internal fixed-point accumulator of the IP had to be configured to match the full data range and precision of the 32-bit floating-point format.

4.5.2 Model-Based Multiple-Set DB

In reference [122], two versions of the DB algorithm with different input processing properties are described. The first one, the single-set DB, is able to process one input vector at a time and its model-based design and test have been described in the previous Section. The main drawback of such a solution is that if more than a single vector has to be accumulated, each vector has to wait for the result of the previous one before being processed.

The second algorithm is the multi-set DB, which is able to process a continuous stream of input vectors without the need to wait for the output results to be produced. Because the data is processed in a streaming fashion in the SVM context, in this thesis the focus is on the multi-set DB version, although it required a more complex design with respect to the single-set design. In Figure 4.23, the proposed Simulink model-based accumulator is shown.

As for the Single-Set version, the design of the Multiple-Set accumulator model entirely relies on basic Simulink blocks and, even if the working principle is almost entirely similar, the architecture requires consistent modifications.

The core of the architecture is the *Adder With Latency* block, which processes floating-point inputs and has already been introduced in the previous section. The remainder of the architecture features two multiplexers (modeled with Simulink Switch blocks *A_Switch* and *B_Switch*); two multiple-word registers (*Input Buffer (IBUF)*

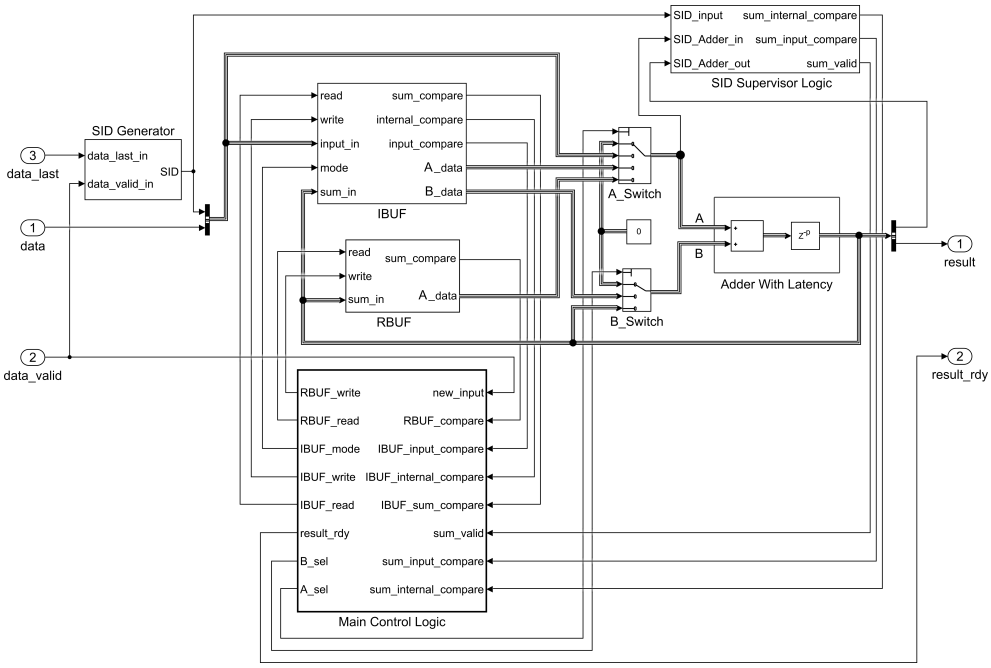


Figure 4.23: Simulink multi-set delayed buffering (DB) accumulator implementation.

and *Result Buffer (RBUF)*); and a control logic that is composed of three blocks: *Set IDentification (SID) Generator*, *Adder Supervisor Logic*, and *Main Control Logic*. The *SID Generator* takes care of the tagging of the input elements to track each element of different sets. Each time the *data_last* flag is asserted together with the *data_valid* flag, the SID value is increased by one and it is merged into the internal bus together with the data value. Thus, all the data in the operands path are a pair of input data and a SID. The size of the counter (i.e., the maximum value of the SID) can be precisely set by knowing that, as found in [122], there cannot be more than $\lceil 5p/3 \rceil$ sets at the same time inside the architecture. The Adder Supervisor Logic, instead, tracks all the SIDs to notify whether the current adder output is of the same set of any other set inside the adder pipeline (*sum_internal_compare*) or the current adder output is of the same set of the input (*sum_input_compare*) or, finally, a new adder output is produced (*sum_valid*). To achieve this result, the internal architecture exploits comparators and simple logic functions. As for the Single-Set version, the *Main Control Logic* is the core control unit of the system. The model-based implementation relies on the pseudocode of Table 4.7 and exploits full combinational logic. The inputs of the logic function are flags indicating relevant events (if new data are available, if a new sum is ready, etc.) and, at the output, produce the configuration setups for all the involved elements (i.e., IBUF, RBUF, A and B working modes, and when the output accumulation is ready). No sequential logic was used for this block, resulting in an output update rate independent of the system clock.

As mentioned, in a Multi-Set DB, two different buffers are needed. The IBUF buffer stores all the input elements that cannot enter the adder immediately because a couple of the same set (i.e., with the same SID) is not yet available. It is composed of an array of memory cells and two controllers, one for read and one for write operations. The model-based architecture is shown in Figure 4.24.

The memory cells array can store the data values along with their SIDs. The model-based implementation of this part exploits the *For Each* Simulink subsystem, which can scale and replicate its internal architecture (i.e., the single memory cell, in this case) based on a parameter N. According to the SSDB version, N was set to $\lceil p/2 \rceil$ to guarantee no storage overflow. When a write operation is issued, the write

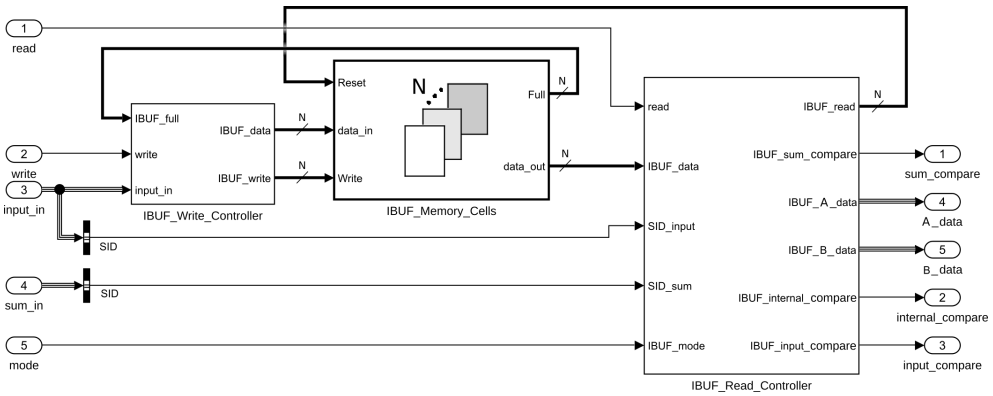


Figure 4.24: Architecture of the *Input Buffer (IBUF)* block.

controller drives the input data to the first empty cell of the array by setting the proper *IBUF_write* array value to 1.

The read controller takes as input the content of all the cells, the SIDs of the input and the sum, the mode of operation, and the read flag. When a read operation is requested by setting the read bit, the content of one or two cells are presented at the *A_data* and *B_data* ports depending on the *mode* input signal. When mode is equal to 1, one value is read and is assigned to *A_data*, which is managed by the A-Switch (Figure 4.23) accordingly to the main control logic combinational function, and eventually set as the input of the adder. In this case, *B_data* value is kept un-set, leaving the other input of the internal accumulator adder to be driven by the main control logic through the *B_Switch* (Figure 4.23). If *mode* is equal to 2, a pair of data of the same set has to be read. The controller logic automatically selects the pair having the same and oldest SID, giving priority to the oldest accumulation result production. The read data are assigned to *A_data* and *B_data*. When mode is equal to 3, the behavior is specular to mode 1: the single read value is assigned to *B_data*, and *A_data* is not used.

The remainder of the output signals serve as inputs for the *Main Control Logic* and are produced by combinational logic. In particular, the *sum_compare* and *input_compare* signals are produced by looking for the cells with the same *SID_sum*

and the *SID_input*. The *internal_compare* signal is computed by comparing the internal content of the memory cells and it is used to notify if two or more memory cells hold data of the same set.

The purpose of the RBUF is quite similar, except that it holds all the adder outputs that cannot be re-introduced yet as inputs, as there are not a couple of operands with the same tag to be processed already. The implementation results in a subset of the architecture of IBUF. In fact, its input signals are only the read, write, and *data_in* values and its output signals are the *sum_compare* and *A_data* values. For this architecture, the value of N was set to $\lceil 2p/3 \rceil$.

4.5.3 Stand-Alone Model-Based Accumulator Implementation

To assess the performance of the proposed accumulator, Simulink simulations were carried out. The standard Simulink IP adder was exploited in the accumulator architecture. This block featured an internal latency of p = 11 cycles. Two mathematical series were exploited as input, and the correct accumulation results were evaluated. In particular, the inputs were the Euler's number e and the Leibniz π mathematical approximation series, defined as

$$e = \sum_{k=0}^{\infty} \frac{1}{k!}, \tag{4.12}$$

$$\pi = \sum_{k=0}^{\infty} \frac{4 \cdot (-1)^k}{2k+1}. \tag{4.13}$$

The series was generated in MATLAB environment as a 50-element vector for the Euler's number series and as a 200-element vector for the Leibniz π series to obtain an approximation error lower than 0.5%. Then, the two vectors were imported in Simulink with the From Workspace block and presented as input to the accumulator. In Figure 4.25, the results of the accumulation of two input vectors are shown.

The two vector series were presented to the input of the accumulator as a data stream, Euler's series first (at t = 0μs), and then the Leibniz series (at t = 0.5μs) (Figure 4.25a). The last element of a single vector was highlighted by the *data_last* signal (Figure 4.25b). The *data_valid* signal was high until a valid data is given to

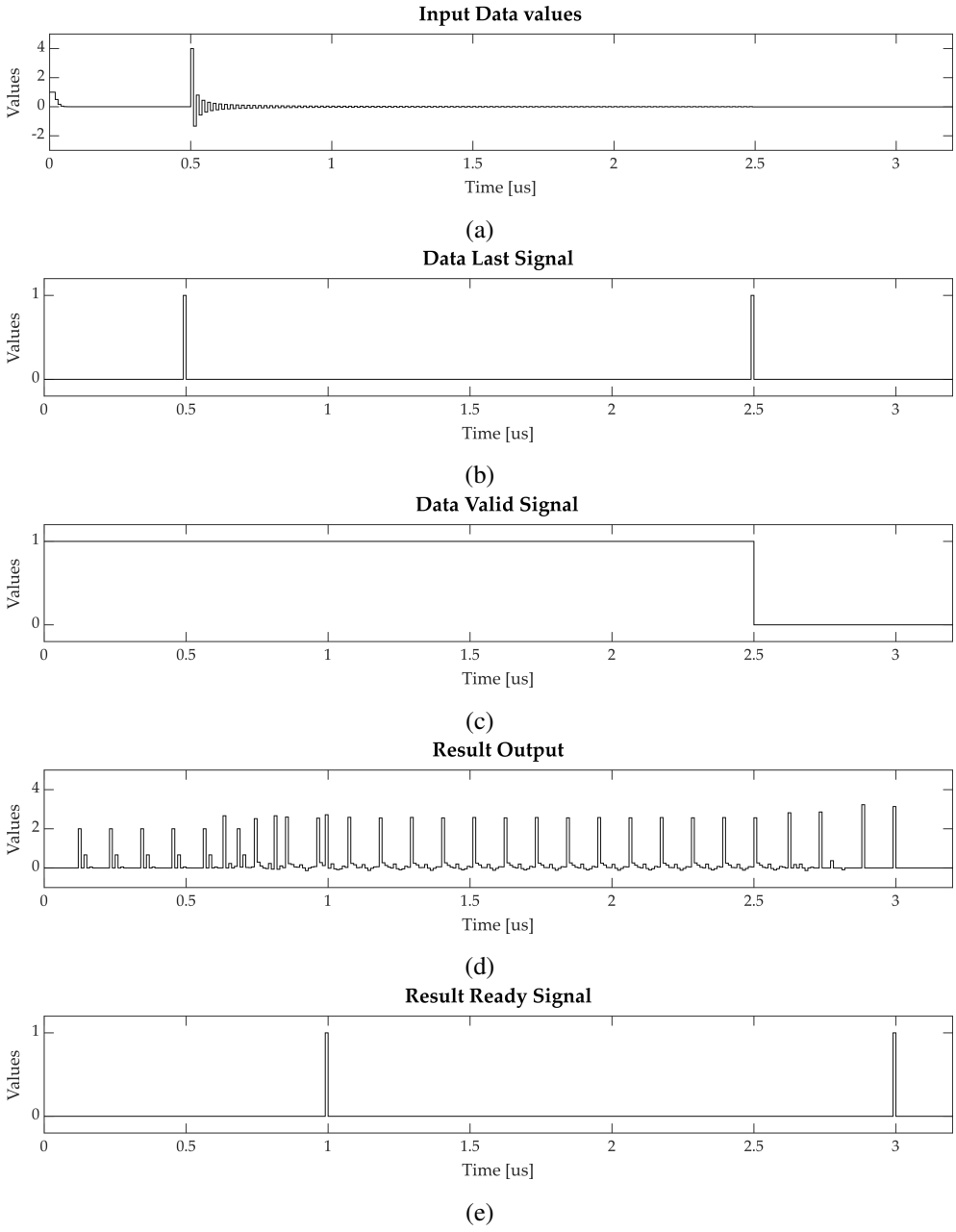


Figure 4.25: Input and output signals of the multi-set DB accumulator in Simulink simulation: (a) Input vector values, (b) *Data_last* signal, (c) *Data_valid* signal, (d) Accumulator output value, and (e) *Result_ready* signal.

the accumulator input (Figure 4.25c). The accumulation outputs (Figure 4.25d) were evaluated when the *result_ready* signal was asserted (Figure 4.25e). From this simulation, the correctness of the results can be assessed. From Equation (4.11), considering a latency of $p = 11$ cycles and a simulation step of 10 ns, the first result was produced 99 cycles after the first input element. Then, the second result was produced 200 cycles after the first result. These time intervals can be verified from Figure 4.25.

Once the functionality of the accumulator architecture was verified, VHDL code was automatically generated from the Simulink model and used to synthesize the circuits in Xilinx Vivado software. Here, results were evaluated in terms of FPGA look-up tables (LUTs), flip-flops (FFs), and (Digital Signal Processor) DSP usage, and maximum achievable clock frequency. For this purpose and to show portability, two different platforms were considered: a Xilinx Artix-7 XC7A100T FPGA device along with Xilinx Vivado 2019.1 software and an Altera Cyclone 10 LP 10CL010 with Quartus 19.1. All the simulations and timing results were carried out considering a clock frequency of 100 MHz. In these experiments, a stream of 200 vectors, each one of 100 elements, was considered to highlight the capability of the models to process subsequent vectors in a short timeframe, without the need for complex input synchronization logic.

The performance of the proposed accumulator model was compared to that of the available Simulink solution. The Simulink IP block takes as input a set of data in parallel to perform the sum. If the input values are fed serially, an input buffer is needed to host all the elements. The time needed for this buffering stage is equal to the length of the input stream, and the length of the buffer represents the maximum vector length the system can accumulate. This, in a VHDL implementation, limits the input streaming vector length. During the VHDL generation process, the accumulator architecture is designed as a binary tree adder or a linear adder chain. For this comparison, the input buffer was set to 100 samples and the architecture to the one offering the lowest implementation resource usage, i.e., the linear adder chain. In Table 4.10, the post-implementation results for Xilinx are reported, whereas data for Altera are shown in Table 4.11.

As shown in Tables 4.10 and 4.11, the proposed accumulator outperformed Simulink

	Proposed Model	Simulink IP
Slice LUTs	1643	40198
Slice Registers	1239	33450
DSPs	0	0
BRAM	0	0
Fmax (MHz)	105	109
Latency (cycles)	49	989

Table 4.10: Simulink accumulator resource usage, maximum frequency, and latency on Xilinx Artix 7.

	Proposed Model	Simulink IP
Logic Elements	2483	47430
DSPs	0	0
Memory (bits)	154	436648
Fmax (MHz)	108	N.A.
Latency (cycles)	49	989

Table 4.11: Simulink accumulator resource usage, maximum frequency, and latency on Altera Cyclone 10 LP.

IP in both area and time. In particular, from the area point of view, in Xilinx implementation, the proposed model used 2.6% of available LUTs and 0.97% of slice registers, whereas Simulink IP used 63.4% and 26.4%, respectively. Moreover, in the Altera implementation, although the number of logic elements (LEs) of the proposed accumulator corresponded to 24%, the Simulink IP could not be implemented; in fact, the occupied area saturated the resources, resulting in a 460% quantity of logic elements. For this reason, the achievable maximum frequency was not reported in this case. The advantage of the new model over the available IP appears evident. It is worth noting that, despite the low area occupied, the proposed solution does not require DSP slices, resulting in the independence of the presence of these blocks in the selected platform, enhancing portability.

To evaluate the performance of the proposed model-based accumulator, once generated, in respect to other solutions, some comparisons were made with other accumulator architectures and available IPs: iterative accumulator, single-set DB, SBM, and Vivado floating-point accumulator IP. These architectures, Vivado IP in particular, are not suitable for automatic code generation; however, these data can give some information about the applicability of the whole process and can confirm the choice of the architecture. In Table 4.12, the Xilinx Artix 7 post-implementation results of the compared accumulator architectures are reported.

The Xilinx Floating-Point IP is made of a fixed-point accumulator wrapped by floating-point conversions at the input and the output stages. To support the full precision and range of the 32 bits floating-point format, the internal fixed-point accumulator register must be correctly configured. The DSP slice usage was disabled to make a fair comparison with the proposed model. Moreover, the architecture optimization was set to produce the lowest latency—with this configuration, the internal fixed-point adder latency value resulted in 23 cycles. As shown in Table 4.12, the proposed model occupied less than a half in the area, without a significant difference in maximum frequency. Furthermore, the achievable frequency of the proposed accumulator is compatible with the maximum frequency allowed by the target FPGA.

Regarding the comparison with other architectures presented in the literature, the proposed accumulator outperforms the iterative and the single-set DB architectures

	Proposed Accumulator	Single-Set DB [126]	Iterative [112]	SBM [125]	Vivado IP
Logic Elements	1643	749	658	1411	3245
Slice registers	1239	811	534	1027	3120
DSPs	0	0	0	0	0
BRAM	0	8.5	8.5	0	0
Fmax (MHz)	105	112	126	102	134
Latency (cycles)	49	9800	200,200	54	23

Table 4.12: Post implementation accumulator resource usage, maximum frequency, and latency on Xilinx Artix 7 FPGA.

in terms of latency needed to produce the result. It is important to note that this difference arises from the fact that the selected architecture is designed to process a stream of consecutive vectors, whereas both the iterative and the single-set DB solutions do not have this capability. The greater the number of vectors to be processed, the greater the latency associated with the latter two architectures. Furthermore, the buffers used for the management of the inputs synchronization must be carefully designed by considering the size of the vectors stream. SBM architecture performs well in terms of the occupied area. As a percentage, it occupies the 2.3% of the available LUTs and the 0.8% of the available slice registers. However, these numbers are close enough to that observed for the proposed model. The same goes for the maximum frequency, with a slight advantage for the selected accumulator. The new model also presents good results in latency, confirming the correct choice of the architecture also compared to the newer solutions presented in the literature.

4.5.4 SVM Kernel Implementation

In order to evaluate the accumulation circuits described thus far, they were exploited in the development of the model-based design of the SVM cubic kernel. Figure 4.26

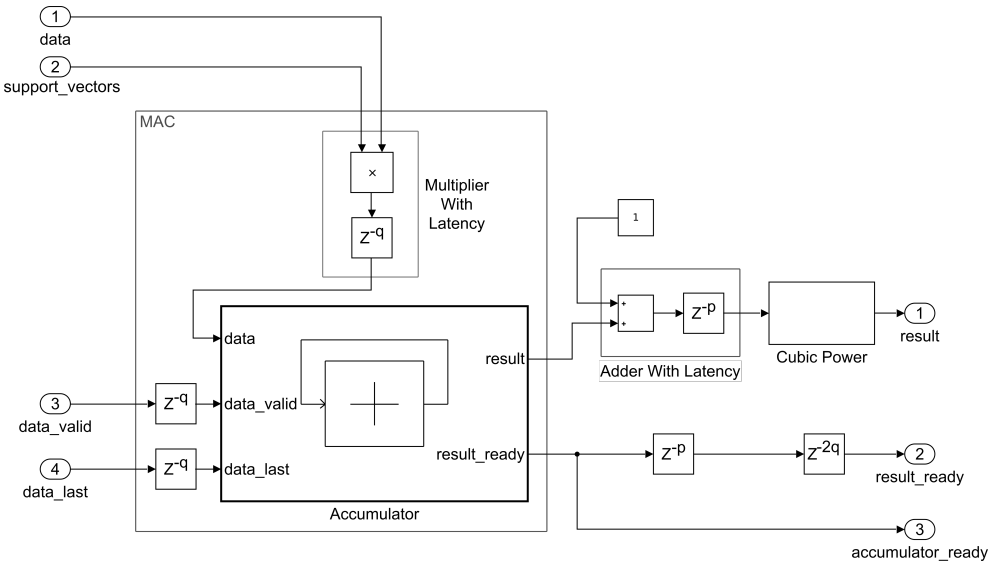


Figure 4.26: Simulink cubic kernel implementation.

shows the model-based kernel designed using the pipelined accumulator. It embeds a multiplier, an accumulator, and a cubic power block. It was tested with different architectures for the accumulator, as explained in the following sections.

All the data flowing inside the kernel are in IEEE 754 floating-point 32 bit format. As for the adder block described earlier, the multiplier block was also modeled with a Simulink floating-point IP cascaded with a delay block in order to take into account the introduced latency (q). To synchronize all the data paths, several delay lines were introduced to compensate for the latency of the mathematical operations. For example, as the cubic power block is implemented as a cascade of two multipliers, the required synchronizing delay on the *result_rdy* signal is twice the delay of a single multiplier ($2q$). The input scheduling should be tailored according to the selected accumulator architecture. In this experiment, a single support vector of 207×81 elements related to a single binary problem was selected and used as *support_vectors* input of the presented kernel architecture. The same statistical elaboration was applied to data in the test set—one vector of 81 elements, representing one instance of

the test set, was exploited as the *data* input of the kernel architecture.

The kernel function was designed as a model-based block in Simulink. For the accumulation process, the comparison has been carried out against the Simulink IP. An HDL code was generated and implemented on the same Xilinx Artix-7 FPGA exploited for the stand-alone accumulators, with a clock frequency of 100 MHz.

In this practical evaluation, other than Simulink and VHDL post-implementation simulations, measurements on hardware implementation were performed. Simulink simulations were performed to compare the proposed model latency with the ones of the kernel implementation with Simulink IP. In both the implementations, standard Simulink floating-point adder and multiplier IP were exploited. Similar to the adder IP, which had an already mentioned latency of $p = 11$, the internal latency of the multiplier IP was found as $q = 6$.

In Figure 4.27, Simulink simulations are shown, in which the *result_ready* signals are plotted. The dashed line refers to the time taken to complete the processing of the dot product of the whole 207×81 support vectors and the 1×81 data vector. In the case of the proposed model (Figure 4.27a), the time needed to accumulate the first vector at the input was equal to 161 cycles. Then, 206×81 cycles were needed for the remaining vectors. Considering a clock frequency of 100 MHz, this corresponded to 168.5 μs.

In the case of Simulink IP (Figure 4.27b), with an input stream of 81 elements and $p = 11$, a total time of 891 cycles were required to obtain the correct accumulation, along with 111 cycles for the remainder of the kernel operations, starting from when the first element was available. Hence, the kernel processing for the first vector took 1002 cycles, and then 206×81 cycles were needed to complete the processing, corresponding to 176.9 μs.

The kernel models' VHDL codes were automatically generated, and performance was evaluated in Vivado environment in terms of resource usage and maximum achievable frequency. Moreover, the latencies resulting from Simulink were verified in the Vivado post-implementation timing simulations. A busy signal was configured in order to be high from the first element presented at the input to the last kernel output produced. Examples are shown in Figure 4.28.

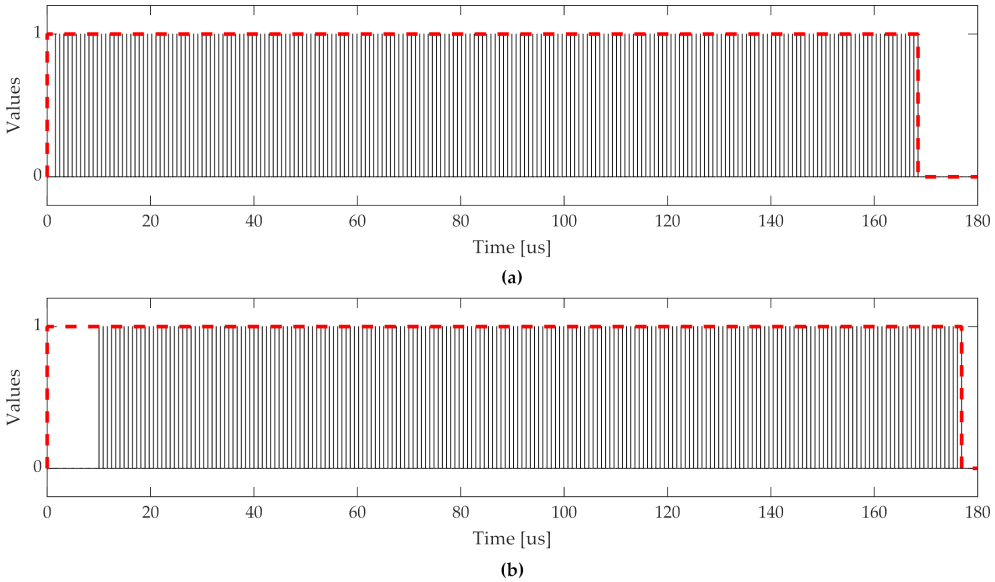


Figure 4.27: Kernel performance in Simulink simulations: (a) Kernel with proposed accumulator, (b) Kernel with Simulink IP accumulator.

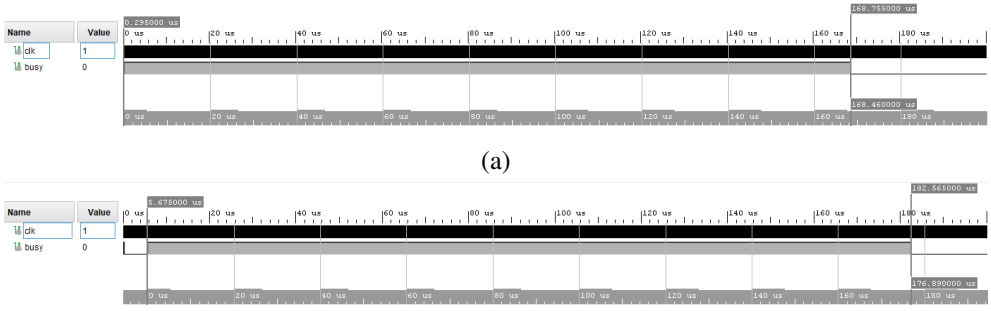


Figure 4.28: Xilinx Vivado post-implementation results of the kernel with (a) Kernel with proposed accumulator, (b) Kernel with Simulink IP accumulator.

	Proposed Accumulator	Simulink IP
Slice LUTs	3266	41354
Slice Registers	2791	34700
DSPs	0	0
BRAM	0	0
Fmax (MHz)	106	106
Latency (cycles)	161	1002

Table 4.13: Post implementation kernel resource usage and maximum frequency on Xilinx Artix 7 FPGA.

Performance in terms of resources usage, maximum achievable frequency, and latency are summarized in Table 4.13.

The resulted latencies confirmed the Simulink simulations and the results on the stand-alone accumulators. The proposed model definitely performed better in terms of occupied area—it used only 5.2% of the available LUTs and 2.2% of the available registers for the whole kernel. Contrarily, the Simulink IP appeared critical in this context, with 65% and 27% of the LUTs and registers, respectively. Considering that many other logic blocks need to be instantiated together with the kernel in a complete SVM implementation, the proposed solution appears a possible valid approach in this context. Moreover, it is worth noting that in wearable sensors, low power consumption has particular relevance. With the technology advancement in the FPGA field, as already mentioned, many low power models have been made available and can be exploited in this context, even considering floating-point arithmetic [127]. The lowest power platforms have generally a low number of resources available; for this reason, the occupied area aspect is of utmost importance in these kinds of applications. Although the maximum operating frequency was the same for both solutions, the resulting latency for the proposed model was definitely lower.

To further confirm the simulation values, the FPGA was configured with the generated code and the performance was measured directly on hardware. The busy sig-

nals were measured using a Tektronix MSO 2024 oscilloscope. Figure 4.29 shows the experimental setup.

Results are reported in Figures 4.30 and 4.31, which are related to the proposed architecture and the Simulink IP, respectively.

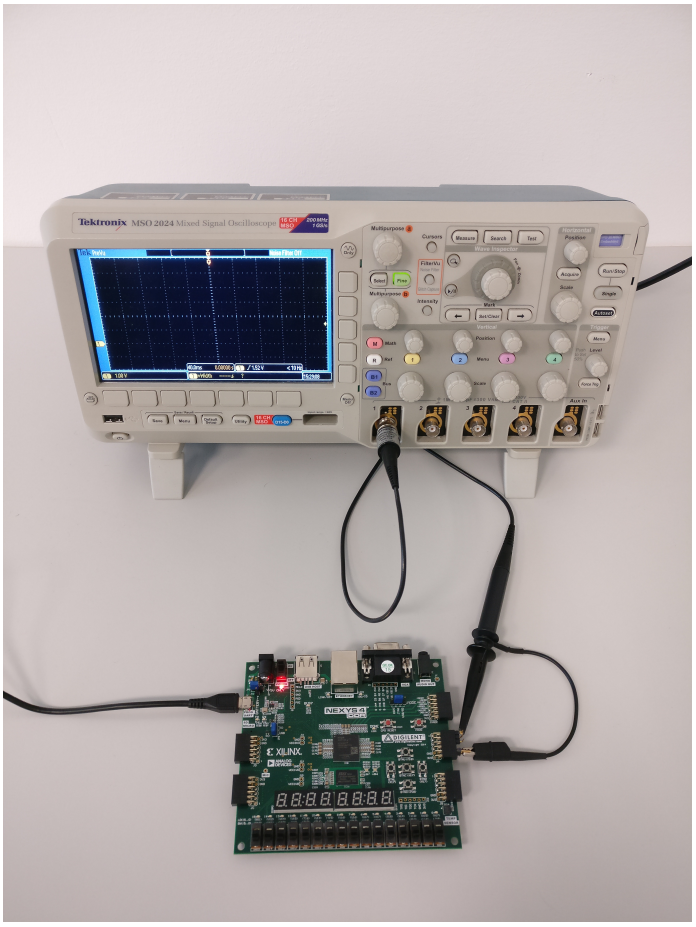


Figure 4.29: Experimental setup for the hardware measurement.

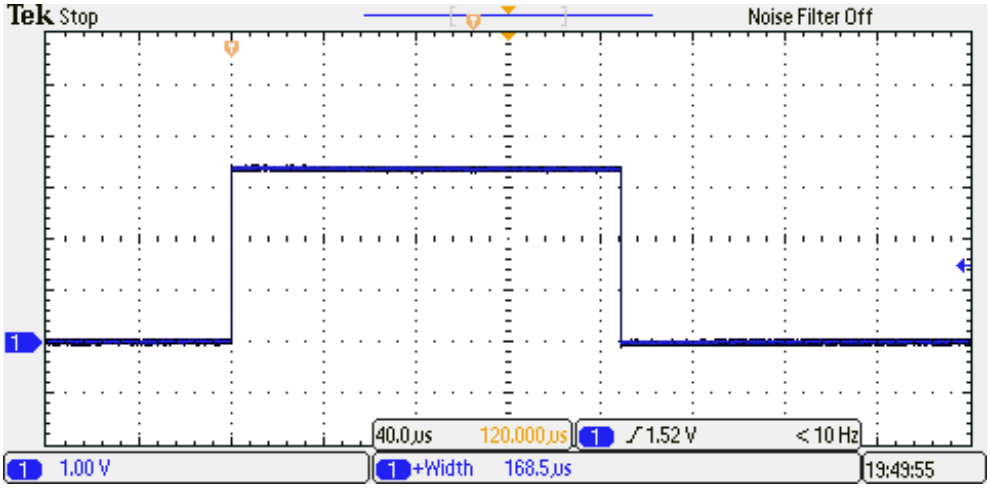


Figure 4.30: Measurement of the processing time of the kernel with the proposed accumulator implemented on the FPGA.

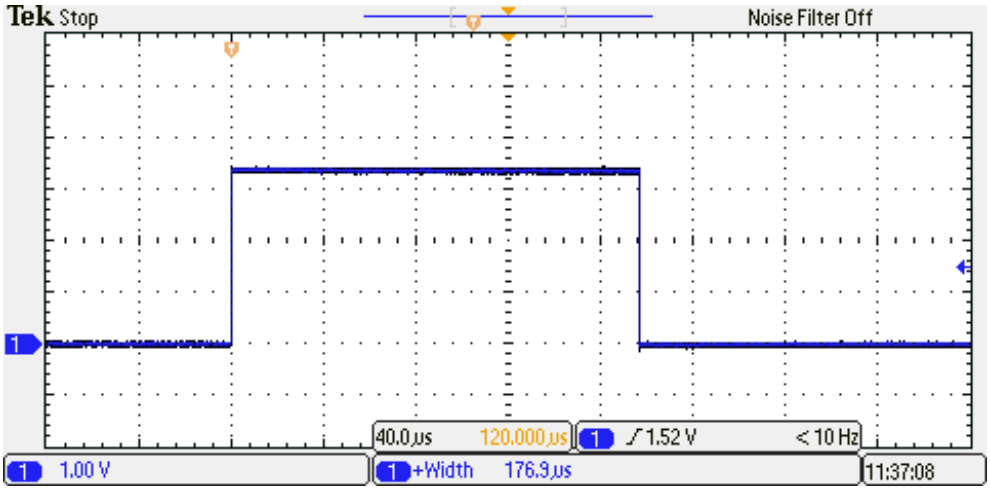


Figure 4.31: Measurement of the processing time of the kernel with Simulink IP implemented on the FPGA.

Measurements confirm the latencies of the simulations and the correctness of the

result. The difference between the two processing times was of about 8.4 μs, which corresponds to 840 clock cycles. As shown in Table 4.13, this result corresponds with the difference in latency of the two solutions.

4.6 Hardware Investigation II: Posit Numeric Format

Another solution which has been investigated to solve the timing issues of the system is the numerical representation used to binary-encode numbers across the architecture.

In binary real numbers representation, the trade-off between the complexity and accuracy given by fixed-and IEEE 754 floating-point formats is a well-known topic [127]. While the fixed-point format can process real numbers with architectures as simple as the one involved in integer arithmetic, floating-point numbers [128] require dedicated and larger architectures which leverages the overall system complexity. On the other hand, the floating-point architecture consistently extends the dynamic range of the representation, reducing the errors and potentially satisfying the increasing accuracy requirement of limited resources devices (e.g. as the IoT edge nodes).

Several works tried to challenge this problem with design techniques based on traditional floating-point architectures optimized for those incoming fields [129, 130, 131] but, nowadays, this trade-off is supposed to be less critical with the introduction of the new Posit numerical representation [33]. As shown in Section 1.3, the outcome of this approach is an increase in the output quality of the computation system, whose result can be more accurate and better matching the decimal real number to be represented. Moreover, low-performance devices can take advantage of the Posit format to produce accurate results with lower resources than the IEEE 754 format. As an example, if an IEEE 754 32-bits dynamic range is required for the application, about the same can be achieved with a 16-bit Posit format. This almost halves the resources needed by the system, potentially enabling the computation on devices with strong hardware constraints.

Posit mathematical operations rely on the same floating-point structures, plus few additional steps (called decoding and encoding phases) required to interpret the

binary encoded number. In other words, as described in detail later, a Posit mathematical operator can be seen as a floating-point operator wrapped by a decoding and an encoding phase, which determines an increase in the hardware complexity. However, this increase is found to be balanced by the save in resources given by the lower bits required to represent the number [132, 133].

4.6.1 Architecture of a Posit Multiplier

The typical operations involved in the multiplication of Posit numbers are depicted in Figure 4.32.

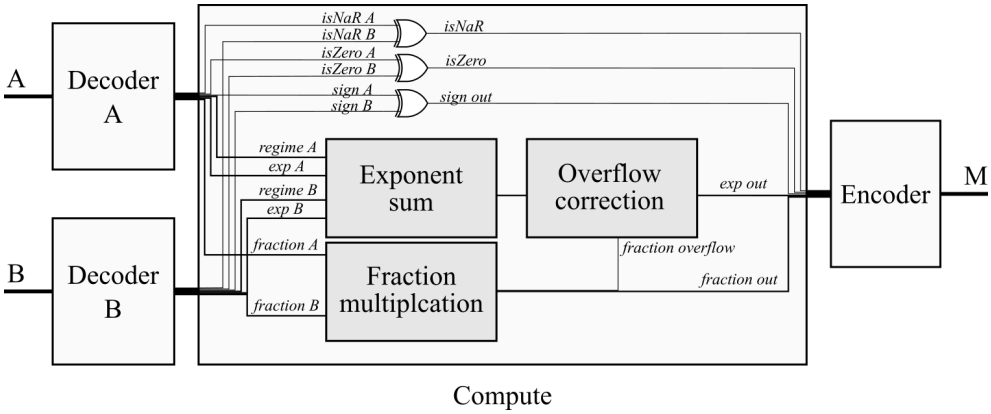


Figure 4.32: Posit multiplier basic elements.

Two numbers A and B are first processed in the decoding phase, which extracts the elements encoded in the Posit format: the sign, the regime, the exponent e and the fraction. Moreover, the decoders also gather information about the special cases of the Posit format, which are the NaR and Zero values.

After the decoding phase, three operations can be performed in parallel in the Compute block: the special cases evaluation, the exponents' sum and the fractions' multiplication. All those steps will be described in detail later.

The results of these operations are then passed to the last encoding phase. Here the chunks of information obtained in the previous phase are packed together to pro-

duce a number which fits the correct Posit format.

Since Posit is a relatively new field, the majority of the works found in literature focuses on designing architectures which provide the minimum set of characteristics to make the multiplier work and the modifications remain limited [134, 135, 136, 137]. However, some works start to introduce variants to improve the performance in terms of energy consumption [138], data fusion for Multiply-Accumulate units or chained operations in general [32, 34, 139, 140], and reconfigurability [141].

4.6.2 Model-Based Multiplier Implementation

The proposed implementation of the Posit multiplier relies on the same workflow used before: Simulink in combination with its HDL Coder automatic code generation tool for FPGA implementation.

Decoder The decoding phase used for implementation is derived from the work presented in [134], which consists of the architecture of Figure 4.33 for an example of a Posit(8,1) number.

The first step of the system is to separate the sign (located in the 8th bit) from the rest of the bits. The remaining 7 bits are then two's complemented if the number representation was negative (hence, if the sign bit was "1"). After being sign-corrected, the bit string holds the regime starting from the 7th bit, and the *exp* and the fraction starting from an unknown position which depends on the length of the regime field.

The length of the regime field is extracted in the central path of the architecture, which involves the Leading One Detector (LOD) presented in [134]. The parametrized recursive pseudocode of the system is reported in Table 4.14.

The output of the LOD is then used for fraction and exponent e extractions, and for the regime integer value computation. The fraction and the exponent e are embedded in the Posit bits after the regime bits. Hence, once known the length of the regime bits, the two values can be extracted by shifting and cutting the input bits. In addition, a leading 1 needs to be appended at the fraction's most significant bit to match the fixed-point representation of the value.

```
     LOD #(N) (in[N-1:0], K[S-1:0], vld)
1    GENERATE
2      IF( N == 2 )
3        vld = |(in), K = !in[1] & in[0]
4      ELSIF ( N & (N - 1))
5        LOD #(1 « S)({1 « S {1'b0}} | in, K, vld)
6      ELSE
7        K_L[S-2:0], K_H[S-2:0], vld_L, vld_H
8        LOD #(N » 1) (in[(N » 1) - 1:0], K_L, vld_L)
9        LOD #(N » 1) (in[N - 1:N » 1], K_H, vld_H)
10       vld = vld_L | vld_L
11       K = vld_H ? {1'b0, K_H} : {vld_L,K_L}
12   ENDGENERATE
```

Table 4.14: Pseudocode of the LOD algorithm developed in [134]. N is the bit-size of the input *in*, S is equal to $\log(N)$, K is the output value, and *vld* is the control signal used in the cascade of the blocks.

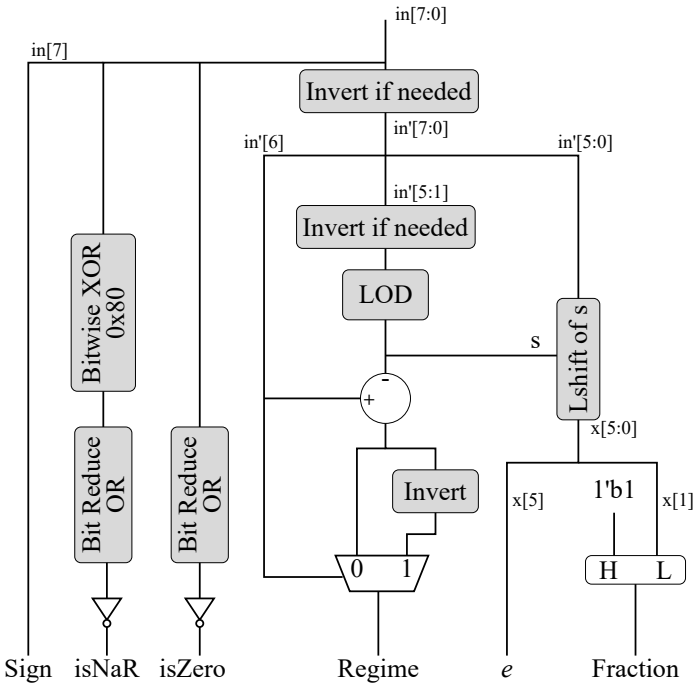


Figure 4.33: Posit multiplier decoding block.

The regime value output is produced by processing the LOD output according to Equation 1.4, hence by evaluating if the 7th input bit was 1 or 0 and acting by adjusting the count value with a +1 and a two's complement if needed.

In a separate path, the isNaR and isZero flags are generated by testing if the input Posit number matches the special values for NaR and Zero.

Encoder

The encoding phase is responsible of repacking in a form compatible to the Posit standard, and its architecture is depicted in the Figure 4.34, with reference to the usual Posit(8,1) format:

The fraction coming from the multiplication stage is already formatted and ready to be included in the last part of the Posit string, hence no processing is required. On

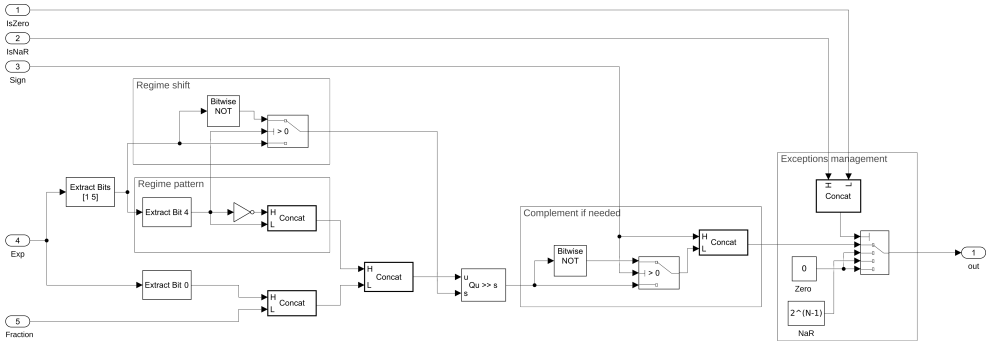


Figure 4.34: Posit multiplier encoding block.

the other hand, the exponent *exp* value produced by the adder is a combination of the regime and the exponent *e* values required by the Posit format.

The separation is immediate for the exponent *e* part, which is directly the least significant bit of the strings. The regime integer value can also be directly extracted by the rest of the bits, but it requires additional processing to be converted from an integer value to a sequence of leading bits and a termination bit. This is performed by defining the regime pattern (i.e. generating the correct first leading bit and the correct termination bit, “01” or “10”) and how many times the first leading bit must be replicated to build the full regime bit-string. The first part is performed by the “regime pattern” area, while the second part is performed by the “regime shift” area. The result of the “regime pattern” area is appended to the fraction and exponent *e* string. Then, this string is arithmetic shifted right by the value computed in the “regime shift” area to replicate the most significant bit the required number of times.

The final steps perform a negation of the string based on the value of the sign bit, its insertion on top of the string, and a gating phase which overwrites the output value if one or both the input Posit numbers were Zero or NaR.

Exponent *exp* sum and fraction multiplication

In this work, two novel versions of Posit multipliers have been designed for this part, both relying on the concepts introduced by the *Vedic multiplication*. The first one is

based on a traditional Vedic architecture for the multiplier part. The second is based on a modified Vedic version specifically designed for the Posit multiplication. In fact, as will be seen, it is able to perform both the fraction multiplication and the exponent *exp* sum with a unique, synergistic architecture.

Vedic multiplication The Vedic multiplication is a technique for integer binary numbers which relies on the "vertical and crosswise" (or Urdhva-Tiryakbhyam) algorithm [142]. In general, the multiplication of two binary numbers A and B involves two steps: the Partial Products Generation (PPG) and the sum of such partial products. The PPG part of the Vedic multiplier is performed with a sliding window which splits the input binary strings into substrings. These substrings are then bitwise processed with and AND logic function in a crosswise fashion, as shown in the example of Figure 4.35.

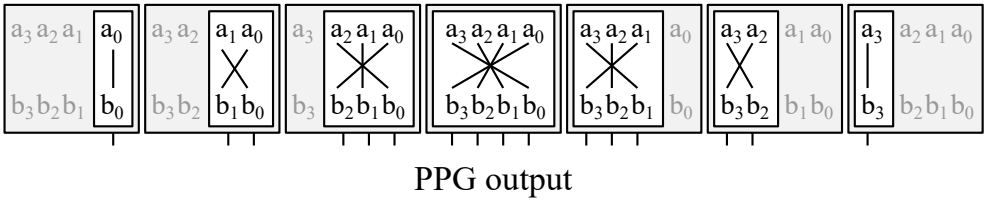


Figure 4.35: Partial product generation with Vedic "vertical and crosswise" pattern.

The architecture can be clearly split in independent sections, each one producing the respective partial products. The number of sections depends on the number of bits of the inputs A and B. The concept of Figure 4.35 is translated to the blocks of Figure 4.36.

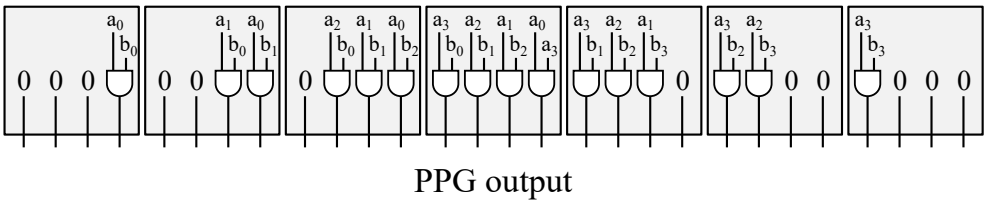


Figure 4.36: Vedic PPG with AND gates.

The PPG outputs needs then to be summed using a binary adder. The best performing adder architecture found in literature for Vedic multipliers is the compressor circuit ([143], with particular reference to the 4:2 version [144]) paired with a Full Adder (FA).

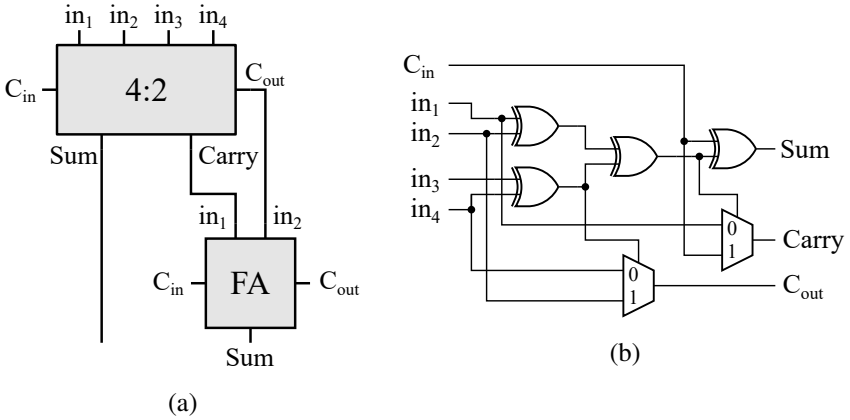


Figure 4.37: Vedic adder elements: (a) 4:2 compressor coupled with a FA, (b) circuit of a 4:2 compressor.

With reference to the example of Figure 4.37, one compressor and full adder couple is needed for each section to compute the output multiplication result. Therefore, the final Vedic architecture turns to be organized in columns as shown in Figure 4.38.

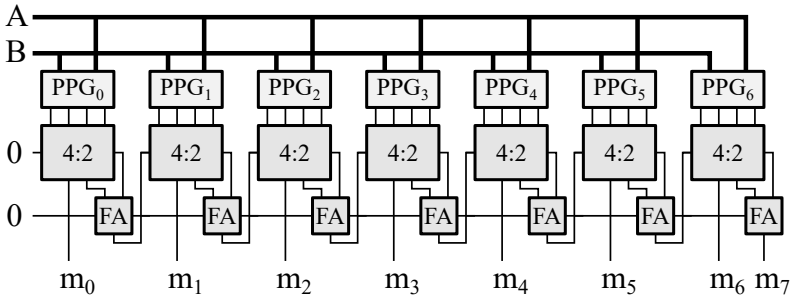


Figure 4.38: Complete 4-bits Vedic multiplier.

Each column is called stage, and the number of stages depends on the number of

PPG sections, hence on the number of input bits. In particular, the required number of stages is equal to $2n - 1$, where n is the number of the operands input bits.

If the number of input bits increases, the adder part must scale accordingly. However, thanks to the modularity of the whole Vedic approach, it is possible to keep the same 4:2 compressors and FAs as base blocks and change their number and position. Two examples of stages are reported in Figure 4.39.

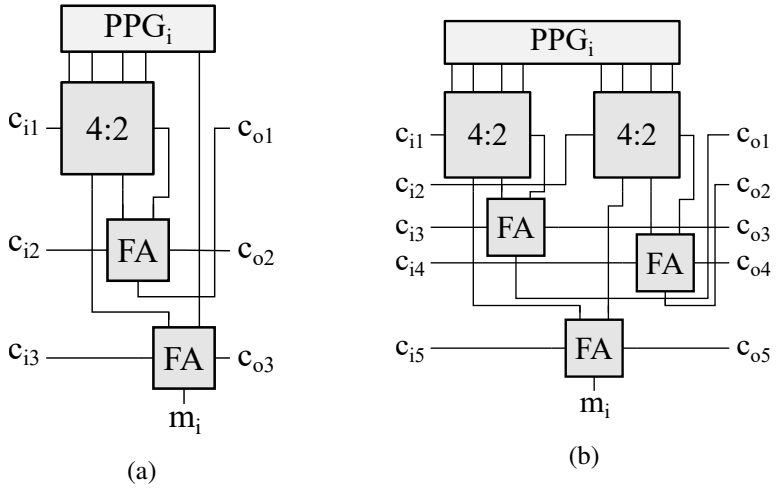


Figure 4.39: Example of stages for (a) 5-bit and (b) 8-bit Vedic architectures.

Standard Posit Multiplier Integration In this solution, the fraction, sign, exponent *exp*, regime and exceptions computation are reported in Figure 4.40.

The system is made of three paths which can be mostly executed in parallel, except for the fraction overflow bit interfering with the "exp management" path. The output sign follows the classical multiplication rule, which is represented by the XOR logical function. The Zero and NaR output conditions are set if at least one of the inputs holds those values. Hence, the logical function mapping this behavior is the OR. The priority of Zero or NaR is defined in the Encoder final part interfacing with the Posit output. The exponent *exp* result is performed by the integer sum of the inputs, which, according to the Posit format, are the concatenation of the regime and

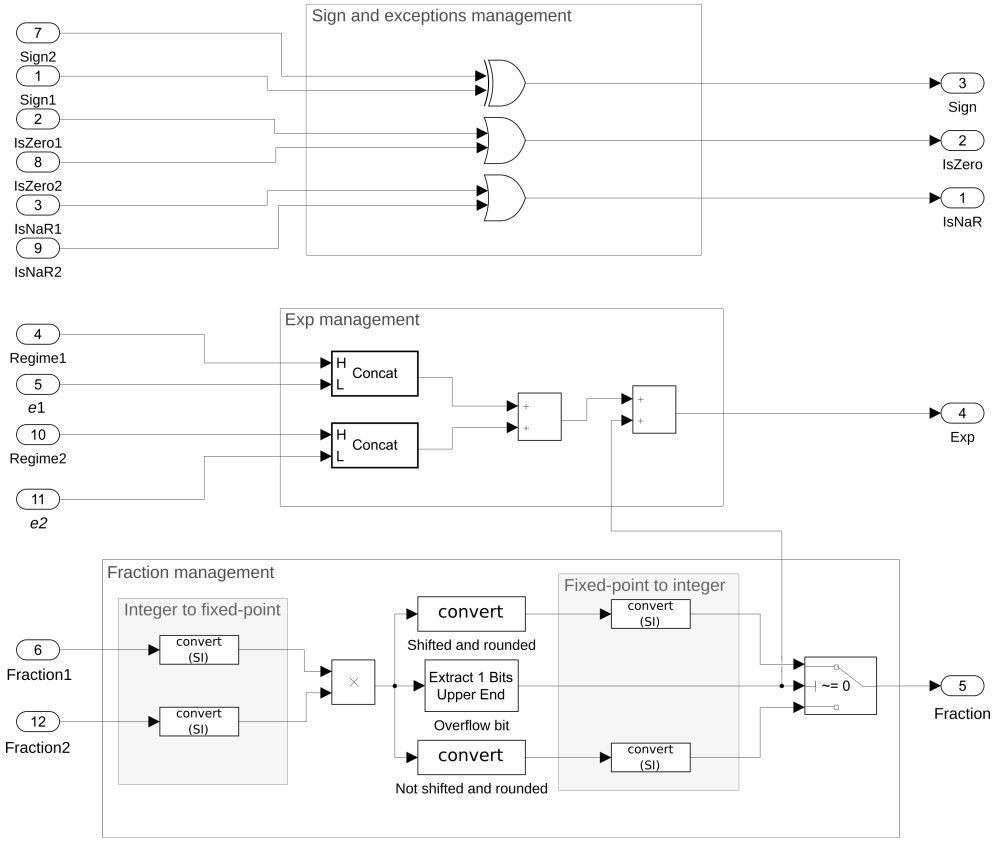


Figure 4.40: Posit multiplier exponent *exp* sum, fraction multiplication, and exception and sign management.

the exponent *e* as the high and low part, respectively. The computation must also take into account the fraction overflow. If it occurred, the fraction is right-shifted of one position and the overflow is transferred to the exponent exp with an increment of one.

The fraction computation is performed by multiplying the input values and then managing the rounding and the overflow of the result. In Figure 4.40, two conversion stages are introduced, with two elements at the input and two elements before the routing switch to create a central area in which all the data are encoded as fixed-point. This is done to exploit the Simulink rounding features included in the fixed-

point domain. In fact, the output of the multiplier can be rounded and/or shifted by simply tuning the properties of the *Convert* blocks without the need of additional design effort. Moreover, this approach gives flexibility when choosing the desired rounding mode since the block allows for selecting among ceiling, convergent, floor, nearest, round, simplest or zero rounding modes.

The fraction multiplication here is performed exploiting the traditional Vedic architecture. In Figure 4.41, an example of fraction multiplication of a Posit(8,1) system is depicted.

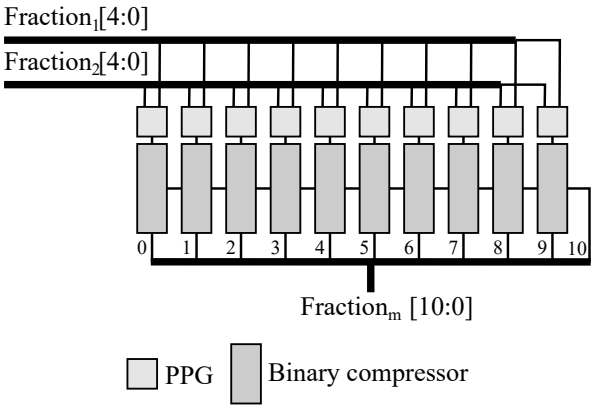


Figure 4.41: 5-bits Vedic multiplier basic blocks to be used for a Posit(8,1) multiplier system.

As shown, a 5-bit Vedic integer multiplier is used to perform the multiplication in a Posit(8,1) arithmetic. This is a direct outcome of the Posit rules reported in Section 1.3.

Proposed Posit Multiplier Integration With this solution, the fraction multiplication and the exponent *exp* sum are performed in the same block. Hence, the design of the compute block is the one of Figure 4.42.

All the components have the same function as the ones of Figure 4.41, except the modified Vedic block, which features the architecture of Figure 4.43.

The design refers to the same Posit(8,1) example and, as in Figure 4.41, the frac-

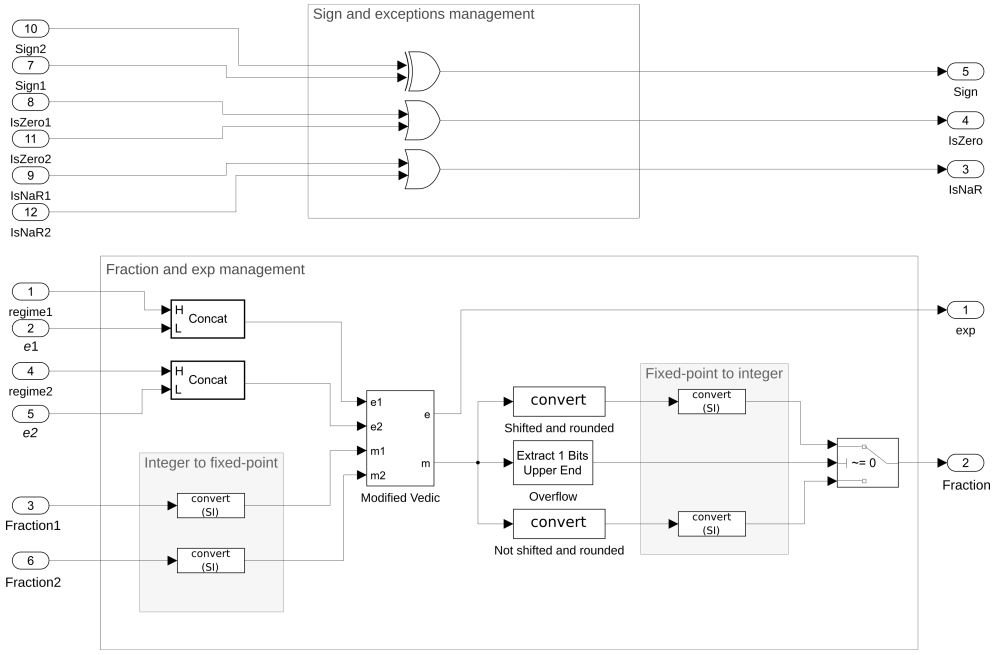


Figure 4.42: Posit multiplier exponent *exp* sum, fraction multiplication, and exception and sign management.

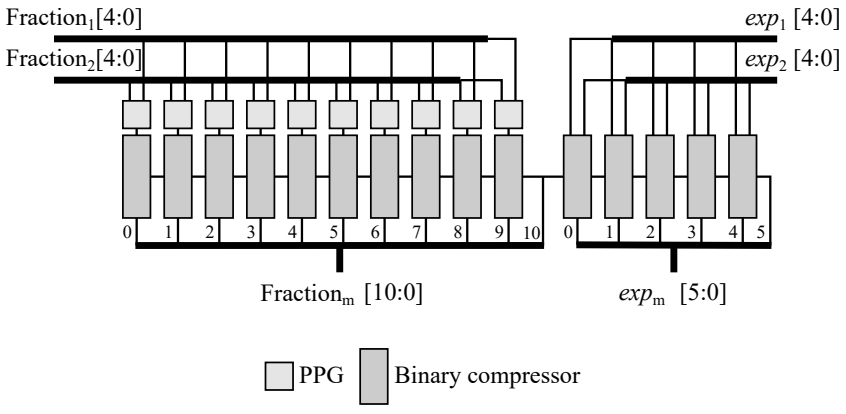


Figure 4.43: Modified Vedic multiplier-adder, designed for 5-bit input fractions and 5-bit input exponents *exp*.

tion and the exponent *exp* input numbers directly derives from Equation 1.5.

This modified Vedic block exploits the modularity of the stage-based Vedic approach to extend the architecture and preserving the modularity. In fact, in the example, five additional stages purged by the PPG part are appended to the right of the multiplier architecture to add the sum functionality needed for the exponent *exp* processing. Moreover, this solution also embeds the overflow detection and correction, which is provided by the connection of the carry chains.

4.6.3 Methodology

To assess the performance of the system, the model has been exported in VHDL through the HDL Coder tool and implemented in the Xilinx Vivado design suite. After the implementation process, the metrics of area usage and data timing have been evaluated in terms of Look-Up Tables (LUTs) usage and logic worst path timing, which is the time of the path passing through the highest number of LUTs. In this evaluation, the time contribution given by the interconnections between LUTs has not been considered since it is highly susceptible to contextual implementation conditions. In fact, it is rare to find implementations of a multiplier as a single component in a hardware platform, and a practical use case is needed to evaluate the system as a whole, including the routing. Here, to assess the single multiplier component performance, the only parts taken into account are the ones which are always present independently of the surrounding system, i.e. the LUTs and their associated timing.

The proposed Posit multiplier has been designed using Simulink basic blocks, which are then translated to VHDL through the HDL Coder tool. For comparison, three additional architectures based on typical multiplier and adder algorithms are introduced to demonstrate the increase in performance of the proposed solution. In particular, the Decoder and Encoder parts are kept the same, and the central computational part has been changed by introducing one multiplier based on the Booth radix-4, one based on the Simulink default implementation and one based on a standard Vedic block.

The modified Vedic model described in the previous Section is the only one able to process the fraction and the exponent *exp* in the same block. Hence, in all the other

systems used for this comparison, the exponent *exp* is processed using the default Simulink *Sum* block. During the code generation, the HDL Coder converts this block with the "+" VHDL operator, which is in turn mapped with a predefined Intellectual Property (IP) architecture in Vivado.

Moreover, for further assessment, another set of architectures have been designed based on the same algorithms but with pipelined techniques. The aim of this experiment was to identify the critical paths of the systems and to evaluate the maximum performance achievable by reducing them. According to the theory, the high modularity of the Vedic solutions is expected to ease the pipeline design, hence, to reach better performance.

Combinatorial designs

The first set of tests was carried out on fully combinatorial architectures designed as follows.

Booth radix-4 multiplier This version is based on the default Simulink adder for the exponent *exp* sum and on the Booth radix-4 multiplier [145] for the fractions, which is shown in Figure 4.44.

The system has been implemented in VHDL (hence without the Simulink modeling) following the typical Booth radix-4 multiplier algorithm. At first, one of the operands (B, in this case) is processed by the recoding phase. Then, the A operand is processed with the recoded B operand in the PPG block. As the final step, the PPG output is summed in the Carry Propagate Adder (CPA) block to produce the multiplier result M.

Simulink default When dealing with code generation, Simulink is able to offer default implementations for all the supported blocks. As for the adder, the multiply block is mapped with the standard VHDL operator "*" when exported with the HDL Coder tool. Vivado interprets this operator through its IP multiplier embedded in the design suite.

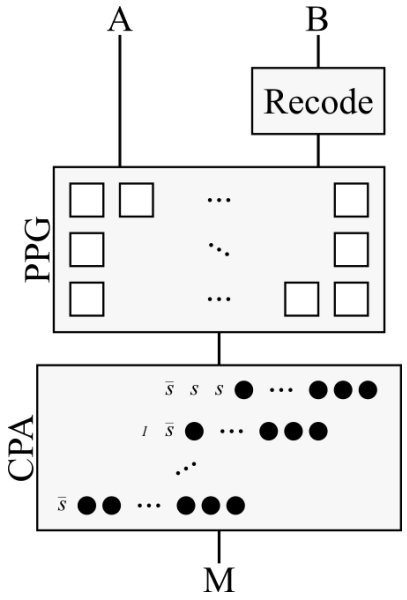


Figure 4.44: Booth radix-4 multiplier basic blocks.

Vedic multiplier This version relies on the default adder for the exponent *exp* sum and on the traditional Vedic multiplier for the fraction multiplication. The model-based architecture of this solution is the one presented in Figure 4.40 and, as found in [144], an improvement in performance is expected compared to the previous solutions.

Proposed solution This is the solution embedding the fraction multiplication and the exponent *exp* sum in the same, Vedic-based architecture. It refers to the blocks reported in Figure 4.42 and Figure 4.43.

Pipelined designs

The first step to designing the pipelines was to divide the architectures with registers to isolate and check the contributions of Decoder, Compute and Encoder parts, as shown in Figure 4.45.

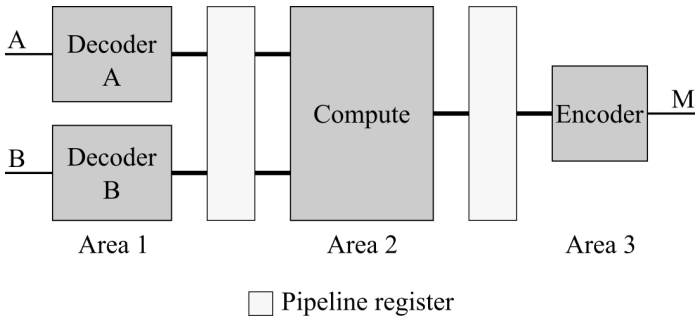


Figure 4.45: Posit multiplier pipeline subdivision to isolate Decoder, Compute, and Encoder areas.

The modifications have been performed in Simulink following the basic pipeline techniques [146] and, after the high-level simulations, the model has been exported to Vivado, in which the implementation has been issued.

Then, further pipeline insertion has been performed to break down the Compute architecture, as shown in Figure 4.46.

As will be seen, here is where the Vedic architectures show their advantages. Thanks to the high modularity of the stage-based infrastructure, the process of distributing the pipeline registers is straightforward and requires less effort than the other solutions in finding the optimal positions.

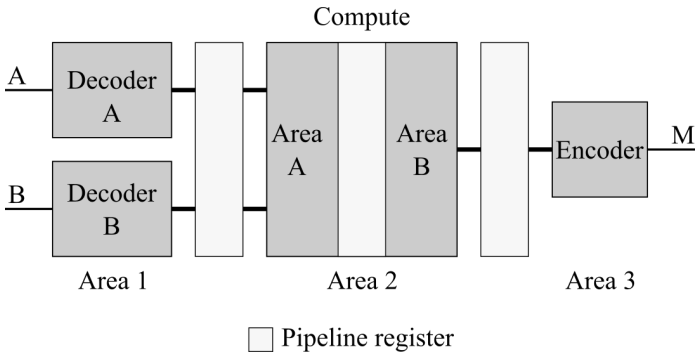


Figure 4.46: Posit multiplier pipeline subdivision to isolate decoder, compute and encoder areas, and to break down the compute area.

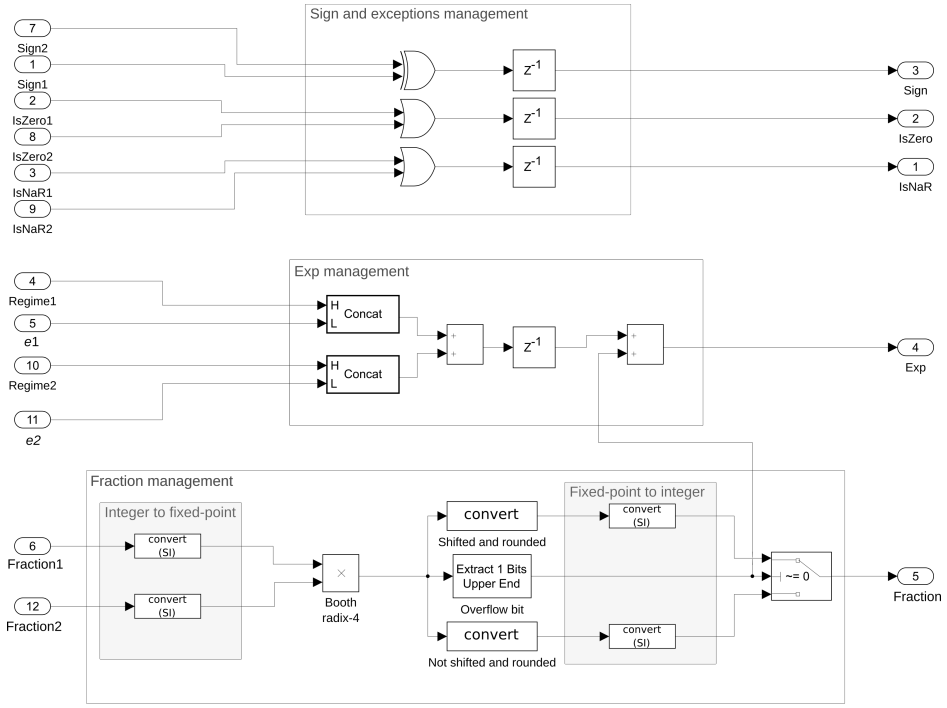


Figure 4.47: Pipelined compute block when the Booth radix-4 multiplier is used. The pipeline registers are modeled in Simulink with the *Delay* block, here represented with the z^{-1} symbol.

Booth radix-4 multiplier The pipeline break-down of the Compute block has been performed by inserting pipeline registers at the block level (Figure 4.47) and inside the Booth multiplier (Figure 4.48).

Moving those registers back or forth of this position determines a decrease in performance, hence this configuration is considered to be the best performing for this architecture.

Simulink default Here, the only design margin allowed is at the Compute block level, since the *Multiply* Simulink block architecture can not be manually pipelined,

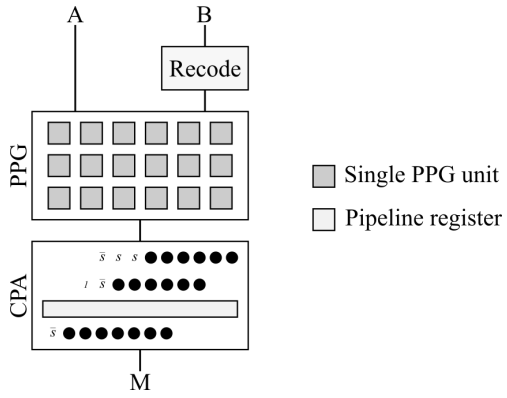


Figure 4.48: Pipelined Booth radix-4 architecture basic blocks.

as for the "*" VHDL operator. The only possible configuration of the pipeline is the one of Figure 4.49.

Vedic multiplier Here, the Compute block has the same pipeline registers as in Figure 4.47 and the Vedic multiplier is internally broken as in Figure 4.50.

Thanks to modularity, it is easy to find the optimal position in which to insert the pipeline registers. Here, the Vedic architecture is not broken at the center because of the other logic (rounding and switch) which is present in the path of the fraction, as shown in Figure 4.47. The *center* to be considered is the center of that path, not the center of the Vedic circuit alone.

Proposed solution In this case, the exponent *exp* is embedded inside the modified Vedic block, hence the Compute block architecture is the one of Figure 4.51.

The modified Vedic block is the one carrying all the heavy logic given by the fraction multiplication and exponent *exp* sum. This means that the pipeline can be inserted more efficiently because both the multiplier and the adder have the same modular architecture. The resulting circuit is the one of Figure 4.52.

This time, the optimal position to break the architecture is two stages further to the right than in Figure 4.50 because of the exponent *exp* architecture, which is

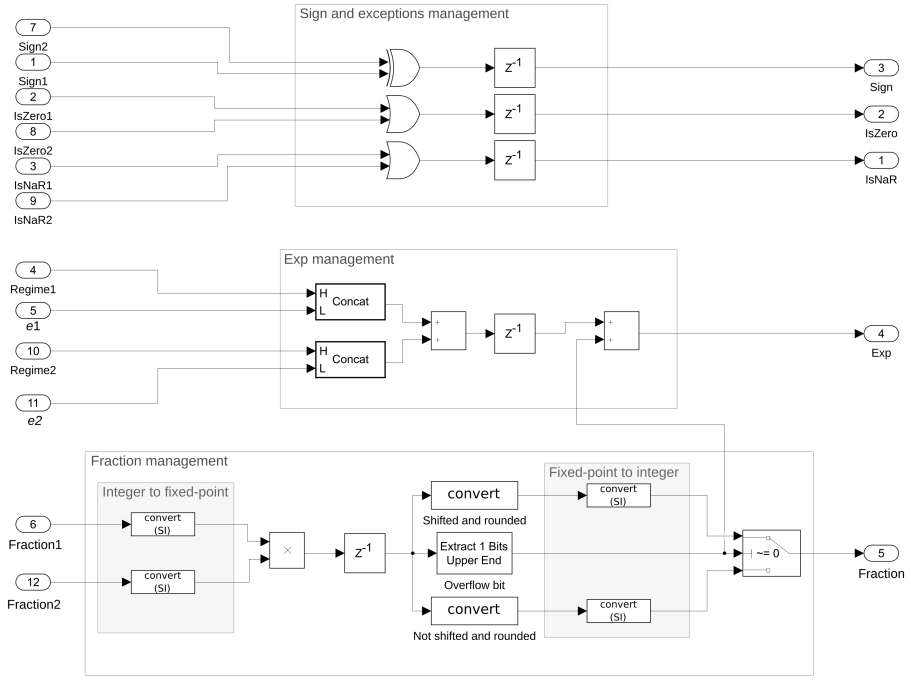


Figure 4.49: Pipelined compute block when the Simulink default multiplier is used.

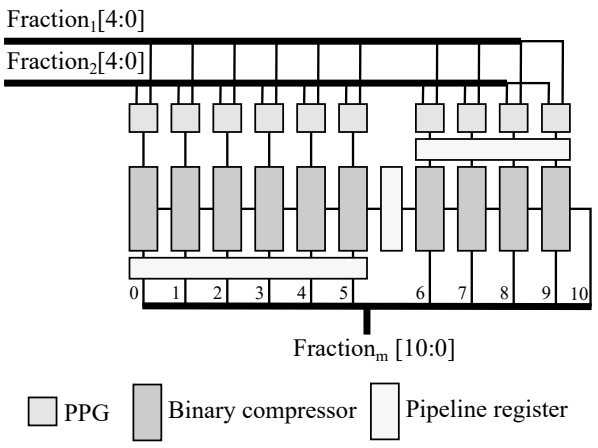


Figure 4.50: Pipelined Vedic multiplier.

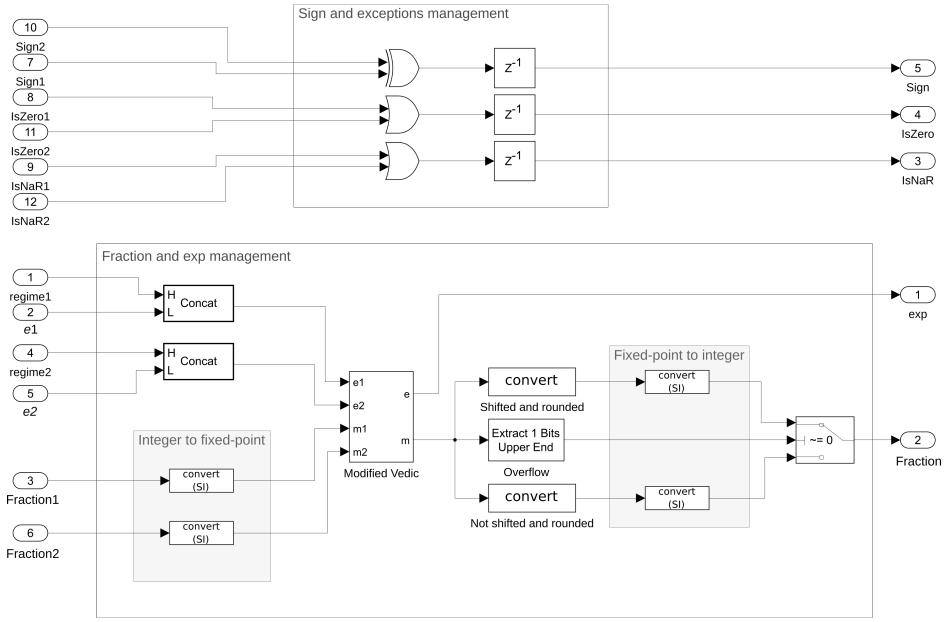


Figure 4.51: Pipelined compute block when the Modified Vedic multiplier-adder is used.

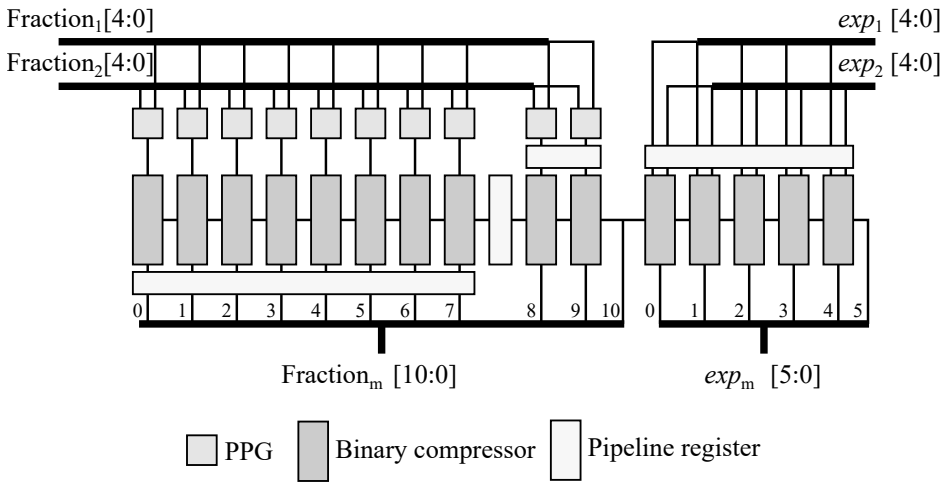


Figure 4.52: Pipelined Modified Vedic multiplier-adder.

Element	A	B	C	D
Fraction multiplier	Booth radix-4	Simulink default	Vedic	Vedic
Exponent *exp* sum	Simulink default	Simulink default	Simulink default	Vedic-based

Table 4.15: Summary of the compared algorithms to perform Posit multiplication.

Parameter	A	B	C	D
LUTs	113	104	123	122
T_{logic} [ns]	3.8	4.3	2.9	2.7
AT	520.6	550.4	426.3	394.2

Table 4.16: Implementation result of the combinatorial versions.

appended to the fraction multiplier and lengthens the carry path.

4.6.4 Results

The obtained variants differing by computing algorithms are summarized in Table 4.15.

The Posit numeric format chosen for the tests is the Posit(8,1) and the implementation has been performed on a Xilinx Artix 7 FPGA. The first set of results are related to the combinatorial implementation of the circuits. The implementation results are shown in Table 4.16.

As expected by the findings of [144], the classical Vedic implementation (C) has better performance than the default Simulink (B) and Booth (A) implementations. However, integrating the fraction and exponent *exp* computations in the same architecture (D) demonstrates to reach better performance when observing the area-time product.

After this assessment, a first pipeline evaluation has been performed to identify

Parameter	A	B	C	D
LUTs	113	104	123	121
FFs	63	63	63	63
$T1_{logic}$ [ns]	1.2	1.2	1.2	1.2
$T2_{logic}$ [ns]	2.6	2.9	2.1	2.1
$T3_{logic}$ [ns]	1.2	1.2	1.2	1.2
AT	457.6	484.3	390.6	386.4

Table 4.17: Implementation result of the 2-stage pipeline versions.

Parameter	A	B	C	D
LUTs	113	104	123	121
FFs	88	82	86	87
$T1_{logic}$ [ns]	1.2	1.2	1.2	1.2
$T2A_{logic}$ [ns]	1.5	2.8	1.3	1.4
$T2B_{logic}$ [ns]	1.3	1.1	1.5	1.0
$T3_{logic}$ [ns]	1.2	1.2	1.2	1.2
AT	312	534.8	306	285.6

Table 4.18: Implementation result of the 3-stage pipeline versions.

the most time-critical elements of each system. Hence, two pipeline stages have been introduced to isolate the decoding, the compute and the encoding parts. The results are shown in Table 4.17.

The sections of the Decoders and the Encoder are equal in terms of delay time, and the central Compute part is the one having the worst path. Hence, a breakdown of the Compute part has been carried out to further evaluate the performance. Results of the newly obtained pipelined circuits are shown in Table 4.18.

The major performance improvement is obtained with the Vedic-based versions as a direct consequence of the ability to design a well-balanced pipeline architecture.

Posit Format	Pipeline Stages
Posit(8,0)	1, 2, 3
Posit(8,1)	1, 2, 3
Posit(16,1)	1, 2, 3
Posit(16,2)	1, 2, 3
Posit(32,1)	1, 2, 3, 4, 5
Posit(32,2)	1, 2, 3, 4, 5

Table 4.19: Posit formats and pipeline stages taken into consideration for the comparison.

The best performing version is the Proposed solution (D), in which the merging of the fraction multiplication and exponent *exp* sum in the same architecture allows for a better critical path reduction.

As a next evaluation step, the comparison has been further extended taking as reference the proposed solution (i.e. the best performing) and the Booth Radix-4 implementation. In particular, the solutions have been compared in terms of Posit format and pipeline stages, as reported in Table 4.19, and results are reported in Figure 4.53.

The proposed solution features better area-time performance in the majority of cases, with a maximum increase of 26.3% in the case of the Posit(8,1).

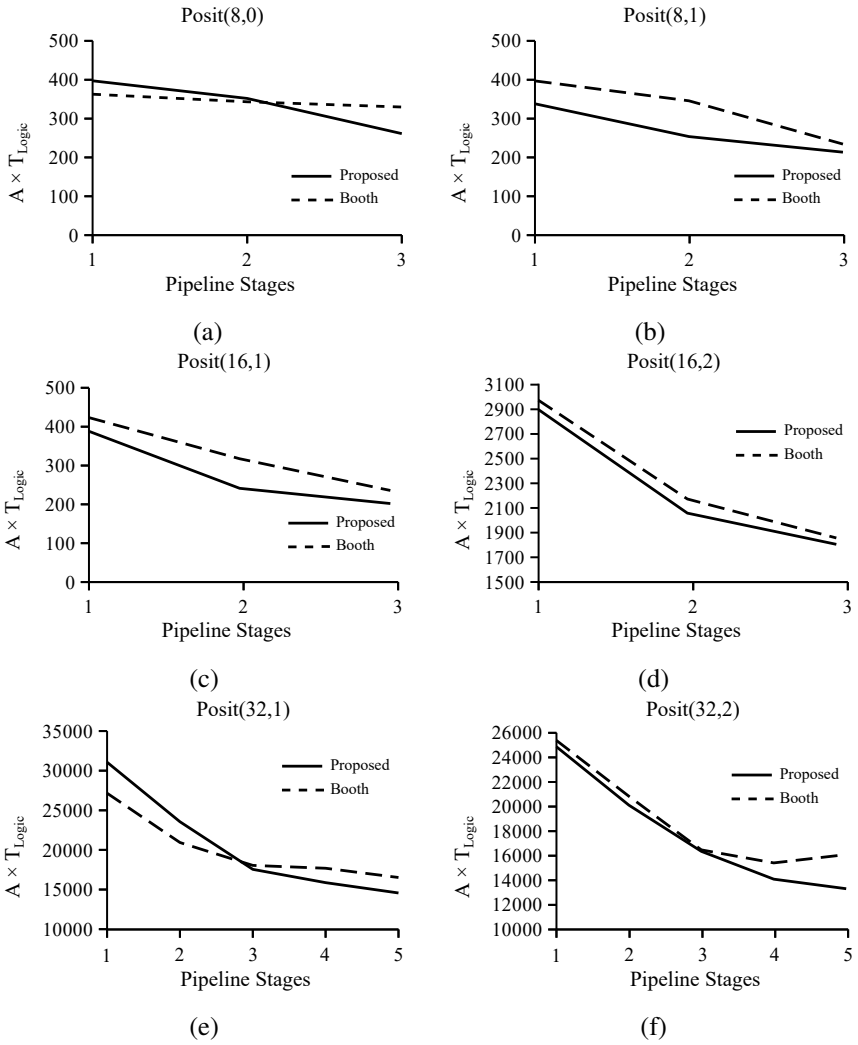


Figure 4.53: Comparison of the Posit multiplier with the proposed and the Booth solutions.

Conclusion

The work presented in this thesis is related to the development and analysis of a set of Wi-Fi IoT sensors to be used in a Smart Home context. This solution offers an alternative to traditional architectures, typically based on ZigBee connectivity. This kind of network is not typically available in the home environment, and this causes problems in the practical deployment, mainly due to the necessity of the set-up of a new and complete network system for every installation.

In the new Wi-Fi IoT vision, the sensors are connected to the Internet through a standard Wi-Fi router, in an IoT compliant fashion, without the need for additional dedicated home gateways to ensure connectivity or the range extenders often needed when using other standard protocols. Hence, a complete set of new devices conceived to acquire data related to users' behavior has been designed and developed.

The first part of this thesis is about the performance assessment of two sets of Wi-Fi sensors composed of an Armchair sensor, a Magnetic contact, a Toilet proximity sensor, and a PIR sensor, which were run for two months. In fact, since the principal drawback of the Wi-Fi connectivity is the higher power consumption, low power design strategies have been considered and applied to demonstrate the possibility of using this architecture in the context of Smart Homes for behavioral monitoring. The first set of sensors was used to evaluate stand-by battery lifetime while the second one was placed in a real home environment, inhabited by a family, to evaluate real battery lifetime. After the two months of testing, only the Toilet proximity sensor ran out of battery. For the other sensors, the laboratory set showed a residual charge of about 46%, while in the home environment this value was 34%. Furthermore, in some

cases, the network could undergo connectivity issues; in these situations, dedicated operating cycles were introduced, instead of the standard ones, to reduce the impact on the battery lifetime. This lead to a reduction in the current absorption of 42% in the case of Wi-Fi network absence and 91% in the case of a lack of Internet connectivity.

Moreover, in order to validate the possibility of using this system as a hardware platform for behavioral analyses, a possible activity profile of an occupant has been extracted from the set of sensors installed in the home environment. A user was assigned to take note of each interaction with the sensors for two months to test the system performance. The accuracy, sensitivity, and specificity were evaluated for the chair sensor (whose interaction, in our environment, can definitely be attributable to a single person) resulting in 95%, 91%, and 100%, respectively. These results show that the developed sensor is definitely capable of providing occupancy information in a real context.

The next part of the thesis focused on the development of a wearable sensor to be introduced in the same ecosystem of the environmental sensors to enable the human activity recognition of the user. The work started from a prototype already defined [22], which has been used as the base platform to introduce the recognition of 7 activities (walking, stand, sitting-down, stay seated, standing-up, running, climbing stairs down, climbing stairs up, lie-down) in the system.

For the activity recognition, a core algorithm has been identified which can be used in both a Support Vector Machine (SVM) classifier and, in general, as the activation function of neural networks algorithms nodes. This allowed for testing the system for the final implementation and, in addition, to gather information for a further implementation of a Convolutional Neural Network (CNN) developed in collaboration with the Computer Engineering research group of the University of Parma.

For the selection of the SVM algorithm in which to introduce the core for the study, different training phases have been carried with the MathWorks® Classification Leaner® tool to compare the performance of several SVM versions, and the Cubic SVM algorithm demonstrated the best accuracy of 93.2%.

Then, a state-of-the-art analysis has been carried out to figure the current advancements in terms of algorithms accuracy, number of recognized activities, and

hardware employed for HAR. The result highlighted difficulties in keeping high both the accuracy and the number of recognized activities, which appears to be the result of a trade-off between systems cost, size, and power saving. Another outcome of the research is the polarization of the hardware, which relies on MCU devices. This led to the selection of an FPGA device as the computational device, which, at the price of a more complex design phase, allows for a more optimized and potentially performing architecture than the one offered by traditional MCUs.

The selected algorithm has been then elaborated for FPGA implementation exploiting the model-based design approach, which allows the designer to focus on the test and prototyping of the functionality in the design phase, and the technical concerns in the final implementation phase. The expected result is an overall improvement in both the design and the testing of the system.

In the first step, the SVM has been modeled to test the functionality of the algorithm, and the basic elements of the system have been identified and evaluated. Then, exploiting the flexibility of the model-based approach, a second version of the model has been designed taking into consideration the final implementation, i.e. the FPGA device. This allowed for ending up with a mixed combinatorial and sequential model which has then been processed by the HDL Coder® tool to generate the final VHDL code.

The VHDL code has been imported in the Altera Quartus II IDE for the implementation in an Altera Cyclone IV FPGA. As a result, the system has been implemented with a resource usage of 88% of the Logic Elements, 100% of the Multipliers, and 15% of the Memory blocks. After this, a VHDL timing simulation has been carried out to evaluate the correctness of the output value. The result of this step turned to be negative, with the verification of the output always ending with a failure.

Two solutions have been identified as the candidates to solve this issue. The first one is the introduction of pipeline stages inside the architecture to break the critical timing paths and to speed up the internal signals. However, this practice introduced a new issue related to the accumulators (i.e. the objects performing the summation of a vector) architectures. The simplest version of an accumulator is made by a sum element in which the first input receives the operand element and the second input is

the feedback of the output. In such a case, if the sum element produces the result in one clock cycle, the accumulation result is simply delayed by one clock cycle after the last input element. However, if the sum element has an internal pipeline of p stages, the input elements frequency must be scaled to f_{CK}/p, where f_{CK} is the clock frequency.

Going back to the SVM of this work, this would mean slowing down every accumulator input, reducing the overall system throughput. In addition, this effect is amplified in the case of cascades of multiple accumulators, which is the case of the design presented in this thesis (e.g. Figure 4.18).

For this reason, a state-of-the-art analysis was carried out to look for a solution, which was identified as the Delay Buffering accumulator. The model-based design approach has been applied to this part of the system and the functionality of the proposed accumulator was first tested with behavioral simulation in the Simulink environment. The tests were carried out using two mathematical series vectors as input, and the results show the correct output accumulation values for both of them. Then, the VDHL code was automatically generated and performance was assessed with post-implementation timing simulations on two different target FPGAs, a Xilinx Artix 7 and an Altera Cyclone 10 LP, in order to demonstrate portability. Results were compared with the available Simulink IP supporting HDL code generation, demonstrating a significant reduction of about 95% in both area and time. Other solutions presented in the literature [126, 112, 125] and the Vivado IP were compared, as well as demonstrating the applicability of the HDL code generation process and confirming the choice of architecture. To frame the accumulator performance in a practical context, it has been evaluated inside of the polynomial cubic kernel of the SVM core. Additionally, in this context, better performance was confirmed, greatly reducing the occupied area and making the solution particularly attractive for implementation in the context of the wearable sensor. The simulation results were also validated with hardware measurements on the target FPGA.

The second solution evaluated to solve the timing issues is a replacement of the IEEE 754 32-bit floating-point format used for the computations inside the system. This numerical format is needed when dealing with data with high numerical dynamic

ranges, since the usage of classical integer or fixed-point formats would determine a loss in accuracy during computations. Hence, in the SVM of this work (and for the majority of machine learning algorithms in general), the floating-point format seems to be the only feasible solution.

However, in recent years a new numeric format named Posit has been proposed as a drop-in replacement of the IEEE 754 standard. Among its characteristics, it features higher dynamic range and accuracy, which allows for building systems with performance similar to floating-point but with fewer bits. As a result, this can reduce the overall complexity of the architecture.

For this reason, a study on the Posit has been carried out using the model-based design approach on the Posit multiplier as a use case. Several architectures have been presented in the literature for this object and, in this work, an improvement to the state-of-the-art has been introduced with a modified version of the multiplier.

A Posit multiplier is composed of three main elements: a decoding part, a computational part, and an encoding part. In detail, the computational part is equivalent to the IEEE 745 one: the sum of the exponents, the multiplication of the fractions, and the management of the sign and the special cases. In this work, the exponents' sum and the fractions' multiplication parts have been merged in a single, synergistic part by using a Vedic-based architecture.

The Vedic architecture is at the heart of the Vedic multiplier for integer numbers. Like the majority of binary integer multipliers, it is made of a partial product generation (PPG) part and an adder part. In the Vedic multiplier, the PPG is made by using a "vertical and crosswise" pattern, whose results are summed up in the adder using a series of 4:2 compressors paired with Full-adders. The main advantage of this architecture is the modularity, since the PPG and the 4:2 compressors are configured in independent *columns*.

This modularity has been exploited and introduced in the Posit multiplier to replace the exponents' sum and fractions' multiplication with a unique block. The architecture features a traditional Vedic multiplier for the fractions and an adder built as an extension of the 4:2 compressors columns for the exponents' sum.

The proposed Posit multiplier has been compared with a traditional solution

based on a Booth Radix-4 multiplier for the fractions. The comparison has been carried out in terms of Posit numbers bit length (i.e. 8, 16, and 32 bits) and pipeline stages introduced inside the system (i.e. from 1 to 3 for the 8 and 16 bits, and 1 to 5 for the 32 bits) and results show a performance improvement of up to 26.3% of the area-time product of the proposed solutions.

In the future development of the system, an integration of the solutions explored to solve the VHDL timing failure is to be performed to evaluate the effects inside the SVM core algorithm. This integration process is expected to be facilitated by the model-based approach which has been at the base of all the thesis work.

Once tested and validated the core system, the next step is the implementation evaluation of the CNN produced in collaboration with the Computer Engineering team, which demonstrated a recognition accuracy of 97% in the training phase. The process is expected to take advantage of the solutions found in this work and, again, the execution of the workflow of Figure 4.6 can be exploited to reach the hardware implementation. At this point, it is possible to compare the two systems to identify the one to adopt for further developments inside the Smart Home ecosystem.

Bibliography

[1] Reding D.F. and Eaton J. Science & Technology Trends 2020-2040. Technical Report 1, NATO Science & Technology Organization, 2020.

[2] Kevin Ashton et al. That 'internet of things' thing. *RFID journal*, 22(7):97–114, 2009.

[3] De Munari I. Bassoli M., Bianchi V. A plug and play iot wi-fi smart home system for human monitoring. *Electronics*, 7:200, 2018.

[4] Carutasu N. L. Pîrjan A. Petrosanu D. M., Carutasu G. A review of the recent developments in integrating machine learning models with sensor devices in the smart buildings sector with a view to attaining enhanced sensing, energy efficiency, and optimal building management. *Energies*, 12:4745, 2019.

[5] Molnar A. J. Bocicor M. I. Cuesta-Frau D. Molina-Picó A. Goga N. Marin I., Vasilateanu A. I-light - intelligent luminaire based platform for home monitoring and assisted living. *Electron*, 7:220, 2018.

[6] Ghavami M. Dudley S. Rana S. P., Dey M. Signature inspired home environments monitoring system using ir-uwb technology. *Sensors*, 19:385, 2019.

[7] Pandya S. Ren H. Akbarzadeh S. Mukhopadhyay S. C. Chen C. Gope P. Chouhan A. Chen W. Ghayvat H., Awais M. Smart aging system. uncovering the hidden wellness parameter for well-being monitoring and anomaly detection. *Sensors*, 19:766, 2019.

[8] De Munari I. Ciampolini P. Guerra C., Bianchi V. Cardeagate: Low-cost zigbee-based localization and identification for aal purposes. *In Proceedings of the IEEE International Instrumentation and Measurement Technology Conference (I2MTC), Pisa, Italy, 11-14*, 2015:245–249, May 2015.

[9] F. Grossi, G. Matrella, I. De Munari, and P. Ciampolini. A flexible home automation system applied to elderly care. In *2007 Digest of Technical Papers International Conference on Consumer Electronics*. IEEE, January 2007.

[10] Ribeiro B. Cardoso A. Leitao J., Gil P. A survey on home energy management. *IEEE Access*, 8:5699–5722, 2020.

[11] Kim H. Son H. A pilot study to test the feasibility of a home mobility monitoring system in community-dwelling older adults. *Int. J. Environ. Res. Public Health*, 16:1512, 2019.

[12] Silvia Ceccacci, Andrea Generosi, Luca Giraldi, and Maura Mengoni. An user-centered approach to design smart systems for people with dementia. In *2017 IEEE 7th International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*. IEEE, September 2017.

[13] Debajyoti Pal, Suree Funilkul, Nipon Charoenkitkarn, and Prasert Kanthamanon. Internet-of-things and smart homes for elderly healthcare: An end user perspective. *IEEE Access*, 6:10483–10496, 2018.

[14] S. Vechet, J. Hrbacek, and J. Krejsa. Environmental data analysis for learning behavioral patterns in smart homes. In *2016 17th International Conference on Mechatronics - Mechatronika (ME)*, pages 1–6, 2016.

[15] Himanshu Thapliyal, Rajdeep Kumar Nath, and Saraju P. Mohanty. Smart home environment for mild cognitive impairment population: Solutions to improve care and quality of life. *IEEE Consumer Electronics Magazine*, 7(1):68–76, January 2018.

[16] Sergio Saponara, Massimiliano Donati, Luca Fanucci, and Alessio Celli. An embedded sensing and communication platform, and a healthcare model for remote monitoring of chronic diseases. *Electronics*, 5(4):47, August 2016.

[17] Dai Sasakawa, Naoki Honma, Takeshi Nakayama, and Shoichi Iizuka. Human posture identification using a MIMO array. *Electronics*, 7(3):37, March 2018.

[18] Losardo Agostino, Grossi Ferdinando, Matrella Guido, De Munari Ilaria, and Ciampolini Paolo. Exploiting aal environment for behavioral analysis. *Assistive Technology Research Series*, 33(Assistive Technology: From Research to Practice):1121–1125, 2013.

[19] Khusvinder Gill, Shuang-Hua Yang, Fang Yao, and Xin Lu. A zigbee-based home automation system. *IEEE Transactions on Consumer Electronics*, 55(2):422–430, May 2009.

[20] Kwang il Hwang, Byoung-Jo Choi, and Seok hoon Kang. Enhanced self-configuration scheme for a robust ZigBee-based home automation. *IEEE Transactions on Consumer Electronics*, 56(2):583–590, May 2010.

[21] Imran Zualkernan, A. Al-ali, Mustafa Jabbar, Imad Zabalawi, and Ahmed Wasfy. InfoPods: Zigbee-based remote information monitoring devices for smart-homes. *IEEE Transactions on Consumer Electronics*, 55(3):1221–1226, August 2009.

[22] Marco Bassoli, Valentina Bianchi, Ilaria De Munari, and Paolo Ciampolini. An IoT approach for an AAL wi-fi-based monitoring system. *IEEE Transactions on Instrumentation and Measurement*, 66(12):3200–3209, December 2017.

[23] V. Bianchi C. Guerra M. Bassoli I. De Munari and P. Ciampolini. The helicopter project: Wireless sensor network for multi-user behavioral monitoring. In *Proc. Int*, pages 1445–1454. Conf. Eng. Technol. Innov. Eng. Technol. Innov. Manag. Beyond New Challenges New Approaches (ICE/ITMC, January 2018.

[24] S. C. Mukhopadhyay. Wearable sensors for human activity monitoring: A review. *IEEE Sensors J.*, 15(3):1321–1330, March 2015.

[25] A. Khan N. Hammerla S. Mellor and T. Plotz. Optimising sampling rates for accelerometer-based human activity recognition. *Pattern Recognit. Lett.*, 73:33–40, April 2016.

[26] Ingo Steinwart and Andreas Christmann. *Support vector machines*. Springer Science & Business Media, 2008.

[27] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. Adaptive Computation and Machine Learning series, 2018.

[28] MATLAB. Model-Based Design for Embedded Control Systems. 2019.

[29] Rob Doorakkers. Model Based Definition: nice-to-have or must-have for an automated, flawless industry 4.0 production environment? Technical report, 2019.

[30] S. Perry. Model based design needs high level synthesis - a collection of high level synthesis techniques to improve productivity and quality of results for model based electronic design. *in*, 2009:1202–1207, 2009.

[31] Lee Y. Sorchini Z. Mignogna A. Agirman I. Kim H. Choe J. M., Arnedo L. Model-based design and dsp code generation using simulink® for power electronics applications. In *Proceedings of the 10th International Conference on Power Electronics and ECCE Asia (ICPE 2019-ECCE Asia)*, pages 27–30, Korea, pp. 923–926, May 2019. Busan.

[32] Zachariah Carmichael, Hamed F. Langroudi, Char Khazanov, Jeffrey Lillie, John L. Gustafson, and Dhireesha Kudithipudi. Deep positron: A deep neural network using the posit number system. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, March 2019.

[33] J. L. Gustafson and I. Yonemoto. Beating floating point at its own game: Posit arithmetic. *Supercomputing Frontiers and Innovations*, 4(2), June 2017.

[34] Hao Zhang, Jiongrui He, and Seok-Bum Ko. Efficient posit multiply-accumulate unit generator for deep learning applications. In *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, May 2019.

[35] C. Yerrapragada and P.S. Fisher. Voice controlled smart house. In *IEEE 1993 International Conference on Consumer Electronics Digest of Technical Papers*. IEEE.

[36] Huang-Chia Shih. Automatic building monitoring and commissioning via human behavior recognition. In *2016 IEEE 5th Global Conference on Consumer Electronics*. IEEE, October 2016.

[37] Il kyu Hwang, Dae sung Lee, and Jin wook Baek. Home network configuring scheme for all electric appliances using ZigBee-based integrated remote controller. *IEEE Transactions on Consumer Electronics*, 55(3):1300–1307, August 2009.

[38] Cheng-Yi Chen, Chien-Yuan Liu, Chiu-Chan Kuo, and Cheng-Fu Yang. Web-based remote control of a building's electrical power, green power generation and environmental system using a distributive microcontroller. *Micromachines*, 8(8):241, August 2017.

[39] Gabriele Lobaccaro, Salvatore Carlucci, and Erica Löfström. A review of systems and technologies for smart homes and smart grids. *Energies*, 9(5):348, May 2016.

[40] Giovanni Brusco, Alessandro Burgio, Daniele Menniti, Anna Pinnarelli, Nicola Sorrentino, and Luigi Scarcello. An energy box in a cloud-based architecture for autonomous demand response of prosumers and prosumages. *Electronics*, 6(4):98, November 2017.

[41] Iván Froiz-Míguez, Tiago Fernández-Caramés, Paula Fraga-Lamas, and Luis Castedo. Design, implementation and practical evaluation of an IoT home automation system for fog computing applications based on MQTT and ZigBee-WiFi sensor nodes. *Sensors*, 18(8):2660, August 2018.

[42] Marjorie Skubic, Rainer Dane Guevara, and Marilyn Rantz. Automated health alerts using in-home sensor data for embedded health assessment. *IEEE Journal of Translational Engineering in Health and Medicine*, 3:1–11, 2015.

[43] Anindya Nag and Subhas Chandra Mukhopadhyay. Occupancy detection at smart home using real-time dynamic thresholding of flexiforce sensor. *IEEE Sensors Journal*, 15(8):4457–4463, August 2015.

[44] Mingfu Li and Hung-Ju Lin. Design and implementation of smart home control systems based on wireless sensor networks and power line communications. *IEEE Transactions on Industrial Electronics*, 62(7):4430–4442, July 2015.

[45] Hemant Ghayvat, Jie Liu, Subhas Chandra Mukhopadhyay, and Xiang Gui. Wellness sensor networks: A proposal and implementation for smart home for assisted living. *IEEE Sensors Journal*, 15(12):7341–7348, December 2015.

[46] Matrella G. De Munari I. Ciampolini P. Grossi F., Bianchi V. Internet-based home monitoring and control. *Assistive Technology Research Series*, 25(2009):309–313, 2009.

[47] Grossi F. Matrella G. De Munari I. Ciampolini P. Losardo A., Bianchi V. Web-enabled home assistive tools. *Assistive technology research series*, 29:448–455, 2011.

[48] Valentina Bianchi, Ferdinando Grossi, Ilaria De Munari, and Paolo Ciampolini. Multi-modal interaction in aal systems. *Assistive Technology Research Series*, 29:440–447, 2011.

[49] De Munari I. Ciampolini P. Mora N., Bianchi V. Simple and efficient methods for steady state visual evoked potential detection in bci embedded system. In *2014 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 2044–2048. IEEE, 2014.

[50] Matrella G. De Munari I. Ciampolini P. Bianchi V., Grossi F. A wireless sensor platform for assistive technology applications. In *2008 11th EUROMICRO Conference on Digital System Design Architectures, Methods and Tools*, pages 809–816. IEEE, 2008.

[51] Valentina Bianchi, Claudio Guerra, Ilaria De Munari, and Paolo Ciampolini. A wearable sensor for AAL-based continuous monitoring. pages 383–394, 2016.

[52] V. Bianchi, F. Grossi, I. De Munari, and P. Ciampolini. Musa: A multisensor wearable device for aal. In *2011 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 375–380. 2011.

[53] Valentina Bianchi, Ferdinando Grossi, Ilaria De Munari, and Paolo Ciampolini. Multi sensor assistant: A multisensor wearable device for ambient assisted living. volume 2, pages 70–75. American Scientific Publishers, March 2012.

[54] F. Montalto, C. Guerra, V. Bianchi, I. De Munari, and P. Ciampolini. MuSA: Wearable multi sensor assistant for human activity recognition and indoor localization. pages 81–92, 2015.

[55] Jae Yong Cho, Kyung-Bum Kim, Hamid Jabbar, Jeong Sin Woo, Jung Hwan Ahn, Won Seop Hwang, Se Yeong Jeong, Haimoon Cheong, Hong Hee Yoo, and Tae Hyun Sung. Design of optimized cantilever form of a piezoelectric energy harvesting system for a wireless remote switch. *Sensors and Actuators A: Physical*, 280:340–349, September 2018.

[56] Shishir Mallick, Al-Zadid Sultan Bin Habib, Abu Shakil Ahmed, and Sk. Shariful Alam. Performance appraisal of wireless energy harvesting in IoT. In *2017*

3rd International Conference on Electrical Information and Communication Technology (EICT). IEEE, December 2017.

[57] Valentina Bianchi, Paolo Ciampolini, and Ilaria De Munari. RSSI-based indoor localization and identification for ZigBee wireless sensor networks in smart homes. *IEEE Transactions on Instrumentation and Measurement*, 68(2):566–575, February 2019.

[58] J. Pedro Amaro, Rui Cortesão, Fernando J.T.E. Ferreira, and Jorge Landeck. Device and operation mechanism for non-beacon IEEE802.15.4/zigbee nodes running on harvested energy. *Ad Hoc Networks*, 26:50–68, March 2015.

[59] Shadi Al-Sarawi, Mohammed Anbar, Kamal Alieyan, and Mahmood Alzubaidi. Internet of things (IoT) communication protocols: Review. In *2017 8th International Conference on Information Technology (ICIT)*. IEEE, May 2017.

[60] 802.15.4 SoC Qualcomm Overview QCA4020 Multi-mode Wi-Fi, Bluetooth. https://www.qualcomm.com/products/qca4020.

[61] M. Cornacchia, K. Ozcan, Y. Zheng, and S. Velipasalar. A survey on activity detection and classification using wearable sensors. *IEEE Sensors Journal*, 17(2):386–403, 2017.

[62] M. Hooshmand, D. Zordan, D. Del Testa, E. Grisan, and M. Rossi. Boosting the battery life of wearables for health monitoring through the compression of biosignals. *IEEE Internet of Things Journal*, 4(5):1647–1662, 2017.

[63] E. Sazonov, N. Hegde, R. C. Browning, E. L. Melanson, and N. A. Sazonova. Posture and activity recognition and energy expenditure estimation in a wearable platform. *IEEE Journal of Biomedical and Health Informatics*, 19(4):1339–1346, 2015.

[64] P. Jung, G. Lim, S. Kim, and K. Kong. A wearable gesture recognition device for detecting muscular activities based on air-pressure sensors. *IEEE Transactions on Industrial Informatics*, 11(2):485–494, 2015.

[65] W. Wang, A. X. Liu, M. Shahzad, K. Ling, and S. Lu. Device-free human activity recognition using commercial wifi devices. *IEEE Journal on Selected Areas in Communications*, 35(5):1118–1131, 2017.

[66] Y. Gu, F. Ren, and J. Li. Paws: Passive human activity recognition based on wifi ambient signals. *IEEE Internet of Things Journal*, 3(5):796–805, 2016.

[67] J. Wang, X. Zhang, Q. Gao, H. Yue, and H. Wang. Device-free wireless localization and activity recognition: A deep learning approach. *IEEE Transactions on Vehicular Technology*, 66(7):6258–6267, 2017.

[68] X. Huang and M. Dai. Indoor device-free activity recognition based on radio signal. *IEEE Transactions on Vehicular Technology*, 66(6):5316–5329, 2017.

[69] W. Tang and E. S. Sazonov. Highly accurate recognition of human postures and activities through classification with rejection. *IEEE Journal of Biomedical and Health Informatics*, 18(1):309–315, 2014.

[70] F. Lin, A. Wang, L. Cavuoto, and W. Xu. Toward unobtrusive patient handling activity recognition for injury reduction among at-risk caregivers. *IEEE Journal of Biomedical and Health Informatics*, 21(3):682–695, 2017.

[71] J. Hu, W. Zheng, J. Lai, and J. Zhang. Jointly learning heterogeneous features for rgb-d activity recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(11):2186–2200, 2017.

[72] Y. Chen and C. Shen. Performance analysis of smartphone-sensor behavior for human activity recognition. *IEEE Access*, 5:3095–3110, 2017.

[73] L. Wang, T. Gu, X. Tao, and J. Lu. Toward a wearable rfid system for real-time activity recognition using radio patterns. *IEEE Transactions on Mobile Computing*, 16(1):228–242, 2017.

[74] D. Fortin-Simard, J. Bilodeau, K. Bouchard, S. Gaboury, B. Bouchard, and A. Bouzouane. Exploiting passive rfid technology for activity recognition in smart homes. *IEEE Intelligent Systems*, 30(4):7–15, 2015.

[75] R. Serra, D. Knittel, P. Di Croce, and R. Peres. Activity recognition with smart polymer floor sensor: Application to human footstep recognition. *IEEE Sensors Journal*, 16(14):5757–5775, 2016.

[76] M. Bassoli V. Bianchi and I. De Munari. A plug and play iot wi-fi smart home system for human monitoring. *Electronics*, 7:9, September 2018.

[77] Z. Wang, D. Wu, J. Chen, A. Ghoneim, and M. A. Hossain. A triaxial accelerometer-based human activity recognition via eemd-based features and game-theory-based feature selection. *IEEE Sensors Journal*, 16(9):3198–3207, 2016.

[78] J. Hong, J. Ramos, and A. K. Dey. Toward personalized activity recognition systems with a semipopulation approach. *IEEE Transactions on Human-Machine Systems*, 46(1):101–112, 2016.

[79] H. Zhang, W. Zhou, and L. E. Parker. Fuzzy temporal segmentation and probabilistic recognition of continuous human daily activities. *IEEE Transactions on Human-Machine Systems*, 45(5):598–611, 2015.

[80] M. Selmi, M. A. El-Yacoubi, and B. Dorizzi. Two-layer discriminative model for human activity recognition. *IET Computer Vision*, 10(4):273–278, 2016.

[81] L. Wang, X. Zhao, Y. Si, L. Cao, and Y. Liu. Context-associative hierarchical memory model for human activity recognition and prediction. *IEEE Transactions on Multimedia*, 19(3):646–659, 2017.

[82] David W. Aha, Dennis Kibler, and Marc K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, January 1991.

[83] X. Yang and Y. Tian. Super normal vector for human activity recognition with depth cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(5):1028–1039, 2017.

[84] I. Bisio, A. Delfino, F. Lavagetto, and A. Sciarrone. Enabling iot for in-home rehabilitation: Accelerometer signals classification methods for activity and movement recognition. *IEEE Internet of Things Journal*, 4(1):135–146, 2017.

[85] F. A. Machot, A. H. Mosa, M. Ali, and K. Kyamakya. Activity recognition in sensor data streams for active and assisted living environments. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(10):2933–2945, 2018.

[86] Jasmina Novakovic. Using information gain attribute evaluation to classify sonar targets. In *17th Telecommunications forum TELFOR*, pages 1351–1354, 2009.

[87] Y. Guo, D. Tao, J. Cheng, A. Dougherty, Y. Li, K. Yue, and B. Zhang. Tensor manifold discriminant projections for acceleration-based human activity recognition. *IEEE Transactions on Multimedia*, 18(10):1977–1987, 2016.

[88] J. Margarito, R. Helaoui, A. M. Bianchi, F. Sartor, and A. G. Bonomi. User-independent recognition of sports activities from a single wrist-worn accelerometer: A template-matching-based approach. *IEEE Transactions on Biomedical Engineering*, 63(4):788–796, 2016.

[89] Valentina Bianchi, Claudio Guerra, Marco Bassoli, Ilaria De Munari, and Paolo Ciampolini. The HELICOPTER project: Wireless sensor network for multi-user behavioral monitoring. In *2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC)*. IEEE, June 2017.

[90] H. Rezaie and M. Ghassemian. An adaptive algorithm to improve energy efficiency in wearable activity recognition systems. *IEEE Sensors Journal*, 17(16):5315–5323, 2017.

[91] D. Fernandes, A. G. Ferreira, S. Branco, R. Abrishambaf, H. Carvalho, J. Mendes, J. Cabral, and A. Rocha. Energy saving mechanism for a smart wearable system: Monitoring infants during the sleep. In *2016 IEEE International Conference on Industrial Technology (ICIT)*, pages 1932–1937, 2016.

[92] C. Wang, S. J. Redmond, W. Lu, M. C. Stevens, S. R. Lord, and N. H. Lovell. Selecting power-efficient signal features for a low-power fall detector. *IEEE Transactions on Biomedical Engineering*, 64(11):2729–2736, 2017.

[93] C. Wang, W. Lu, S. J. Redmond, M. C. Stevens, S. R. Lord, and N. H. Lovell. A low-power fall detector balancing sensitivity and false alarm rate. *IEEE Journal of Biomedical and Health Informatics*, 22(6):1929–1937, 2018.

[94] Q. Liu, J. Williamson, K. Li, W. Mohrman, Q. Lv, R. P. Dick, and L. Shang. Gazelle: Energy-efficient wearable analysis for running. *IEEE Transactions on Mobile Computing*, 16(9):2531–2544, 2017.

[95] R. Jin, Z. Che, Z. Wang, M. Zhu, and L. Wang. Battery optimal scheduling based on energy balance in wireless sensor networks. *IET Wireless Sensor Systems*, 5(6):277–282, 2015.

[96] T. L. Martin and D. P. Siewiorek. Non-ideal battery properties and low power operation in wearable computing. In *Digest of Papers. Third International Symposium on Wearable Computers*, pages 101–106, 1999.

[97] Sorin-Aurel Moraru, Liviu Pemiu, Dominic Mircea Kristaly, Delia A. Ungureanu, Florin Sandu, and Adrian A. Mosoi. Integrating wireless sensors into cloud systems for ambient assisted living. July 2017.

[98] Dominic Mircea Kristaly, Sorin-Aurel Moraru, Vlad Stefan Petre, Ciprian Adrian Parvan, Delia Elisabeta Ungureanu, and Adrian Alexandru Mosoi. A solution for mobile computing in a cloud environment for ambient assisted living. In *2018 26th Mediterranean Conference on Control and Automation (MED)*. IEEE, June 2018.

[99] Paula Lago, Claudia Roncancio, and Claudia Jiménez-Guarín. Learning and managing context enriched behavior patterns in smart homes. *Future Generation Computer Systems*, 91:191–205, February 2019.

[100] Cc3200 SimpleLink Wi-Fi® and. *and Internet-of-Things solution*. a Single-Chip Wireless MCU | TI.com Available online.

[101] Timilehin Labeodan, Kennedy Aduda, Wim Zeiler, and Frank Hoving. Experimental evaluation of the performance of chair sensors in an office space for occupancy detection and occupancy-driven control. *Energy and Buildings*, 111:195–206, January 2016.

[102] Valentina Bianchi, Claudio Guerra, Marco Bassoli, Ilaria De Munari, and Paolo Ciampolini. The HELICOPTER project: Wireless sensor network for multi-user behavioral monitoring. In *2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC)*. IEEE, June 2017.

[103] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, June 2006.

[104] D. S. Kehler et al. A systematic review of the association between sedentary behaviors with frailty? *Exp. Gerontol.*, 114:1–12, December 2018.

[105] Lombardo G. Fornacciari P. Mordonini M. De Munari I. Bianchi V., Bassoli M. Iot wearable sensor and deep learning: An integrated approach for personalized human activity recognition in a smart home environment. *IEEE Internet Things Journal*, 6:8553–8562, 2019.

[106] Yichuan Tang. Deep learning using linear support vector machines. *arXiv preprint arXiv:1306.0239*, 2013.

[107] Abien Fred Agarap. An architecture combining convolutional neural network (cnn) and support vector machine (svm) for image classification. *arXiv preprint arXiv:1712.03541v2*, 2019.

[108] Richard O Duda, Peter E Hart, et al. *Pattern classification*. John Wiley & Sons, 2006.

[109] FPGA chips are coming on fast in the race to accelerate AI | VentureBeat. https://venturebeat.com/2020/12/10/fpga-chips-are-coming-on-fast-in-the-race-to-accelerate-ai/.

[110] Xilinx FPGAs & 3D ICs. https://www.xilinx.com/products/silicon-devices/fpga.html.

[111] Intel FPGA Products. https://www.intel.com/content/www/us/en/products/programmable/fpga.html.

[112] Creţ O. Tudoran R. De Dinechin F., Pasca B. An fpga-specific approach to floating-point accumulation and sum-of-products. In *Proceedings of the 2008 International Conference on Field-Programmable Technology (ICFPT 2008)*, pages 8–10, Taiwan, pp. 33–40, December 2008. Taipei.

[113] Oberman S. F. Flynn M. J. Advanced computer arithmetic design; wiley: New york, ny, usa. 2001.

[114] Flynn M. J. Hallin T. G. Pipelining of arithmetic functions. *IEEE Trans. Comput*, 21:880–886, 1972.

[115] L. Zhuo, G. R. Morris, and V. K. Prasanna. High-performance reduction circuits using deeply pipelined operators on fpgas. *IEEE Transactions on Parallel and Distributed Systems*, 18:10, October 2007.

[116] Martonosi M. Luo Z. Accelerating pipelined integer and floating-point accumulations in configurable hardware with delayed addition techniques. *IEEE Trans. Comput*, 49:208–218, 2000.

[117] Bakos J. D. Nagar K. K. A high-performance double precision accumulator. In *Proceedings of the 2009 International Conference on Field-Programmable Technology (FPT'09)*, pages 9–11, NSW, Australia, pp. 500-503, December 2009. Sydney.

[118] Kuck D. J. *The Structure of Computers and Computations*. Wiley: New York, USA, 1978.

[119] Kogge P. M. *The Architecture of Pipelined Computers;*. Hemisphere Pub Corp., Washington DC, USA, 1981.

[120] Hwang K. Ni L. M. Vector-reduction techniques for arithmetic pipelines. *IEEE Trans. Comput C-*, 34:404–411, 1985.

[121] H. J. Sips and H. Lin. An improved vector-reduction method. *IEEE Transactions on Computers*, 40:2, 1991.

[122] Y.-G. Tai, C.-T. D. Lo, and K. Psarris. Accelerating matrix operations with improved deeply pipelined vector reduction. *IEEE Transactions on Parallel and Distributed Systems*, 23:2, February 2012.

[123] M. Huang and D. Andrews. Modular design of fully pipelined reduction circuits on fpgas. *IEEE Transactions on Parallel and Distributed Systems*, 24:9, September 2013.

[124] Huang Y. H. A Wei M. Tag based vector reduction circuit. In *In Proceedings of the IEEE High Performance Extreme Computing Conference (HPEC 2015)*, pages 15–173, MA, USA, September 2015. Waltham.

[125] Yin T. Zheng Y. Chen J. Tang L., Cai G. A resource consumption and performance overhead optimized reduction circuit on fpgas. In *Proceedings of the 2019 International Conference on Field-Programmable Technology (ICFPT)*, pages 9–13, China, pp. 287–290, December 2019. Tianjin.

[126] De Munari I. Bassoli M., Bianchi V. A simulink model-based design of a floating-point pipelined accumulator with hdl coder compatibility for fpga implementation. *Appl. Electron Pervading Ind. Environ. Soc. ApplePies Lect. Notes Electr. Eng. In Press*, 2019:1–9, 2019.

[127] C. Inacio and D. Ombres. The DSP decision: fixed point or floating? *IEEE Spectrum*, 33(9):72–74, September 1996.

[128] IEEE. Ieee standard for floating-point arithmetic. August 2008.

[129] Marco Bassoli, Valentina Bianchi, and Ilaria De Munari. A model-based design floating-point accumulator. case of study: FPGA implementation of a support vector machine kernel function. *Sensors*, 20(5):1362, March 2020.

[130] R.V.K. Pillai, D. Al-Khalili, and A.J. Al-Khalili. Evaluation of 1's complement arithmetic for the implementation low power CMOS floating point adders. In *CCECE '97. Canadian Conference on Electrical and Computer Engineering. Engineering Innovation: Voyage of Discovery. Conference Proceedings*. IEEE.

[131] Shilpa Kukati, D.V Sujana, Shruthi Udaykumar, P. Jayakrishnan, and R. Dhanabal. Design and implementation of low power floating point arithmetic unit. In *2013 International Conference on Green Computing, Communication and Conservation of Energy (ICGCE)*. IEEE, December 2013.

[132] Yohann Uguen, Luc Forget, and Florent de Dinechin. Evaluating the hardware cost of the posit number system. In *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, September 2019.

[133] E. Ternovoy, Mikhail G. Popov, Dmitrii V. Kaleev, Yurii V. Savchenko, and Alexey L. Pereverzev. Comparative analysis of floating-point accuracy of IEEE 754 and posit standards. In *2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*. IEEE, January 2020.

[134] Manish Kumar Jaiswal and Hayden K.-H. So. PACoGen: A hardware posit arithmetic core generator. *IEEE Access*, 7:74586–74601, 2019.

[135] Artur Podobas and Satoshi Matsuoka. Hardware implementation of POSITs and their application in FPGAs. In *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, May 2018.

[136] Manish Kumar Jaiswal and Hayden K.-H So. Universal number posit arithmetic generator on FPGA. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, March 2018.

[137] Rohit Chaurasiya, John Gustafson, Rahul Shrestha, Jonathan Neudorfer, Sangeeth Nambiar, Kaustav Niyogi, Farhad Merchant, and Rainer Leupers. Parameterized posit arithmetic hardware generator. In *2018 IEEE 36th International Conference on Computer Design (ICCD)*. IEEE, October 2018.

[138] Hao Zhang and Seok-Bum Ko. Design of power efficient posit multiplier. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 67(5):861–865, May 2020.

[139] Aneesh Raveendran, Sandra Jean, J. Mervin, D. Vivian, and David Selvakumar. A novel parametrized fused division and square-root POSIT arithmetic architecture. In *2020 33rd International Conference on VLSI Design and 2020 19th International Conference on Embedded Systems (VLSID)*. IEEE, January 2020.

[140] Nuno Neves, Pedro Tomas, and Nuno Roma. Dynamic fused multiply-accumulate posit unit with variable exponent size for low-precision DSP applications. In *2020 IEEE Workshop on Signal Processing Systems (SiPS)*. IEEE, October 2020.

[141] Souradip Sarkar, Purushotham Murugappa Velayuthan, and Manil Dev Gomony. A reconfigurable architecture for posit arithmetic. In *2019 22nd Euromicro Conference on Digital System Design (DSD)*. IEEE, August 2019.

[142] K. D. Rao, P. V. Muralikrishna, and C. Gangadhar. Fpga implementation of 32 bit complex floating point multiplier using vedic real multipliers with minimum path delay. December 2018.

[143] Josmin Thomas, R. Pushpangadan, and S Jinesh. Comparative study of performance vedic multiplier on the basis of adders used. In *2015 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE)*. IEEE, December 2015.

[144] Valentina Bianchi and Ilaria De Munari. A modular vedic multiplier architecture for model-based design and deployment on FPGA platforms. *Microprocessors and Microsystems*, 76:103106, July 2020.

[145] A. D. Booth. A signed binary multiplication technique. *Q. J. Mech. Appl. Math*, 4(2):236–240, January 1951.

[146] Eduardo Boemo. Pipelining on FPGAs: A tutorial. In *2019 X Southern Conference on Programmable Logic (SPL)*. IEEE, April 2019.

Notes

Part of the presented work has been already published by the same author in the following papers:

- Bianchi, V., Bassoli, M., Lombardo, G., Fornacciari, P., Mordonini, M., and De Munari, I. (2019). IoT Wearable Sensor and Deep Learning: An Integrated Approach for Personalized Human Activity Recognition in a Smart Home Environment. IEEE Internet of Things Journal, 6(5), 8553–8562.

  https://doi.org/10.1109/JIOT.2019.2920283

- Bassoli, M., Bianchi, V., De Munari, I., and Ciampolini, P. (2017). An IoT approach for an AAL Wi-Fi-based monitoring system. IEEE Transactions on Instrumentation and Measurement, 66(12), 3200–3209.

  https://doi.org/10.1109/TIM.2017.2753458

- Bassoli, M., Bianchi, V., and De Munari, I. (2018). A plug and play IoT Wi-Fi smart home system for human monitoring. Electronics (Switzerland), 7(9), 200.

  https://doi.org/10.3390/electronics7090200

- Bassoli, M., Bianchi, V., and De Munari, I. (2019). A Simulink Model-based Design of a Floating-point Pipelined Accumulator with HDL Coder Compatibility for FPGA Implementation. Applepies (Applications in Electronics Pervading Industry, Environment and Society) International Conference, 1–9.

  https://doi.org/10.1007/978-3-030-37277-4_19

- Bassoli, M., Bianchi, V., and De Munari, I. (2020). A Model-Based Design Floating-Point Accumulator. Case of Study: FPGA Implementation of a Support Vector Machine Kernel Function. Sensors, 20(5), 1362.

  https://doi.org/10.3390/s20051362

Acknowledgments

There are moments in your life in which you take a breath, turn your head back and see the footprints you have left behind on your path. For me, this is one of such moments and I found myself nicely surprised by the number of people, friends, and colleagues who have been relevant in defining what and where I am right now. In particular, I would like to express my gratitude:

To my guides, Ilaria De Munari and Valentina Bianchi, for constantly providing me with suggestions, advice, and guidelines since my BSc degree back in 2013. This work would not have been possible without you, literally, so thank you.

To Paolo Ciampolini and Guido Matrella, who were anytime ready to share some help, to give trust, and to mentor me in difficult moments.

To my colleague and friend Niccolò Mora, who is always glad to share some laugh in front of a coffee cup. Thank you also for being by my side in my first academic international experience, making such an event less unfamiliar.

To my friend Andrea Bettati, always ready to dive with me in any kind of insane technical and technological conversation. Few of them had a real-world outcome (e.g. the open-source FPGA), but striving is the key.

To Jonatan, Filippo, Daniele, Leonardo, Mattia, and all the students who populated the D2Lab over time. It was real fun to design the craziest projects, initiatives, and ideas with you, so thank you all.

Alla fine, ma non per ultimi, alla mia Famiglia e a Laura, che mi hanno sempre ed instancabilmente supportato, incoraggiato e consigliato lungo tutto il percorso.

Senza il vostro aiuto non avrei mai reggiunto questo traguardo.