UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO

Reconfigurable Computing Applied to Latency Reduction in Control and Prediction Systems Focused on Tactile Internet

Sérgio Natan Silva

Advisor: Prof. Dr. Marcelo Augusto Costa Fernandes

Doctoral Thesis presented to the Graduate Program of Electrical and Computer Engineering of UFRN (concentration area: Computer Engineering) as part of the requirements for obtaining the title of Doctor of Science.

PPgEEC Order Number: D288
Natal, RN, January, 2021

Universidade Federal do Rio Grande do Norte - UFRN
Sistema de Bibliotecas - SISBI
Catalogação de Publicação na Fonte. UFRN - Biblioteca Central Zila Mamede

Silva, Sérgio Natan.
Reconfigurable Computing Applied to Latency Reduction in Control and Prediction Systems Focused on Tactile Internet / Sergio Natan Silva. - 2021.
125 f.: il.

Tese (doctoral thesis) - Universidade Federal do Rio Grande do Norte, Centro de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica e de Computação, Natal, RN, 2021.
Orientador: Prof. Dr. Marcelo Augusto Costa Fernandes.

1. Real-time - Doctoral thesis. 2. FPGA - Doctoral thesis. 3. Embedded systems - Doctoral thesis. 4. Tactile internet - Doctoral thesis. 5. Prediction methods - Doctoral thesis. 6. Control system - Doctoral thesis. I. Fernandes, Marcelo Augusto Costa. II. Título.

RN/UF/BCZM CDU 681.586

Elaborado por Ana Cristina Cavalcanti Tinôco - CRB-15/262

Aos meus pais, pela paciência e carinho e a minha noiva pelo apoio e motivação para conclusão deste trabalho.

Agradecimentos

Agradeço primeiramente ao Pai Celestial pela força e resiliência para conclusão deste trabalho.

Ao meu orientador, Marcelo Augusto Costa Fernandes pelos diversos ensinamentos durante a minha jornada de aprendizado. Sou muito grato por tudo.

Aos membros da banca pelas considerações e sugestões.

À minha família pelo apoio durante esta jornada na pessoa de minha avó Rita e avó Maria.

Agradeço a meu pai Carlos e a minha mãe Ilma, pela paciência e dedicação, por estarem sempre ao meu lado me aconselhando da melhor forma possível.

Agradeço também a minha noiva Fernanda, pela força e motivação e por acreditar em mim nos momentos mais difíceis.

Não podia deixar de agradecer ao meu sogro Jair e sogra Ana, bem como a Dona Rita pelo apoio na etapa final deste trabalho.

Aos colegas do LAMII e Laboratório de sistemas de comtrole por todos os momentos de descontração e estudo.

Aos demais colegas de pós-graduação, pelas críticas e sugestões.

À CAPES, pelo apoio financeiro.

Abstract

The Tactile Internet is the current technological advance for the Internet. This new paradigm enables the sending of touch information, as well as the other stimuli already sent. Thus, it is necessary to guarantee a very low latency between the devices that make up the tactile interaction. The time of propagation of information through the communication channel, processing power of local devices, the complexity of the techniques, among others, can be the cause of the system latency. This work proposes the use of dedicated hardware-based reconfigurable computing (RC) to reduce latency in control and prediction systems applied to the Tactile Internet. Two approaches are proposed to address the latency issue. In the first approach, proposes the implementation of linear and non-linear prediction techniques in RC. In this approach, prediction techniques are used to minimize the impacts caused by delays and loss of information. The second approach an intelligent control system based on Fuzzy logic in RC is proposed. The system is a Takagi - Sugeno Fuzzy-PI type controller that aims to reduce the latency associated with processing the tool's control data. The implementation uses a completely parallel strategy associated with a hybrid bit format scheme (fixed-point and floating-point). Still in this approach, two hardware projects are proposed: the first uses a single clock cycle processing architecture and the other uses a pipeline scheme. The proposals are implemented in a Field Programmable Gate Array on the Virtex 6 xc6vlx240t-1ff1156 platform. Data related to occupation and throughput associated with the target platform are presented, as well as a comparison between results through simulation and implementations in dedicated hardware. The results are superior to those presented in other studies in the literature.

Keywords: real-time, FPGA, embedded systems, tactile internet, prediction methods, control system

Resumo

A Internet Tátil é o atual avanço tecnológico para a Internet. Esse novo paradigma possibilita o envio de informações de toque, bem como os demais estímulos já antes enviados. Dessa forma, é preciso garantir uma latência muito baixa entre os dispositivos que compõem a interação tátil. Essa latência está associada ao tempo de propagação da informação pelo canal de comunicação, poder de processamento dos dispositivos locais, complexidade das técnicas em execução, entre outros. Nesse viés, este trabalho propõe o uso de hardwares dedicados baseados em computação reconfigurável (CR) para reduzir a latência em sistemas de controle e predição aplicados a Internet Tátil. São propostas duas abordagens para tratar a problemática da latência. Na primeira abordagem é proposta a implementação de técnicas de predição lineares e não-lineares em CR. Nessa abordagem as técnicas de predição são utilizadas para minimizar os impactos causados por atrasos e perda de informações. Na segunda abordagem é proposto um sistema de controle inteligente baseado em lógica Fuzzy em CR. O sistema é um controlador do tipo Takagi - Sugeno Fuzzy-PI que se propõe a reduzir a latência associada ao processamento dos dados controle da ferramenta. A implementação usa uma estratégia totalmente paralela associada a um esquema de formato de bit híbrido (ponto fixo e ponto flutuante). Ainda nesta abordagem são propostos dois projetos de hardware: o primeiro usa uma arquitetura de processamento de ciclo de clock único e o outro usa um esquema de pipeline. As propostas são implementadas em um de Field Programmable Gate Array na plataforma Virtex 6 xc6vlx240t-1ff1156. São apresentados dados relacionados a ocupação e throughput associados a plataforma alvo, bem como comparação entre resultados através de simulação e implementações em hardware dedicado. Os resultados se mostram superiores aos apresentados em outros trabalhos presentes na literatura.

Palavras-chave: tempo real, FPGA, sistemas embarcados, internet tátil, métodos de predição, sistemas de controle

Contents

List of Figures

List of Tables

List of Symbols e Nomenclatures

ANN	Artificial Neural Networks
BPM	Backpropagation Module
CPD	Cartesian Prediction Device
DSP	Digital Signal Processors
FBF	Feedback Force
FCS	Feedback Control System
FK	Master Device
FL	Fuzzy Logic
$FPGA$	Field Programmable Gate Array
FS	Fuzzy Systems
$FuzzyCS$	Fuzzy Control Systems
$H2M$	Human to Machine
HW	Hardware
IK	Inverse Kinematics
IM	Integration Module
IoT	Internet of Things
IPM	Input Processing Module
JPD	Joint Prediction Device
KFF	Kinesthetic Feedback Force
$M-FIM$	Mamdani Fuzzy Inference Machine
$M2M$	Machine to Machine

MCS Master Computational System

MD Master Device

MFG Membership Function Group

MFM Membership Function Module

ML Machine Learning

MLP Multi-Layer Perceptron

$MLP-BP$ Multi-Layer Perceptron and Backpropagation Algorithm

$MLPM$ Multi-Layer Perceptron Module

MMD Mining of Massive Datasets

MSE Mean Squared Error

NI Number of Implementations

NW Network

OFM Output Function Module

OM Operation Module

OP Operator

PID Proportional–Integral – Derivative

RC Reconfigurable Computing

$ReLU$ Multi-Layer Perceptron

$RMLP-BP$ Recurrent Multi-Layer Perceptron and Backpropagation

SCS Slave Computational System

SD Slave Device

SPD Slave Prediction Device

TI Tactile Internet

$TS-FIM$ Takagi–Sugeno Fuzzy Inference Machine

$TS-FIMM$ Takagi–Sugeno Fuzzy Inference Machine Module

$TS-FIMM-OS$ Takagi–Sugeno Fuzzy Inference Machine Module One Shot

$TS-FIMM-P$ Takagi–Sugeno Fuzzy Inference Machine Module Pipeline

Chapter 1

Introduction

The Tactile Internet (TI) is the latest technological advance on the Internet. This new communications technology guarantees very low latency, low jitter, high security, and reliability (Dohler 2015). Besides, the audiovisual stimuli already known, the TI will support the transmission of sensations of touch and performance in real-time interacting with machines and virtual reality environments (Maier et al. 2016, Ateya et al. 2019). Human-Machine Interactions (H2M) in real-time is one of the main characteristics of applications focused on TI (Ateya et al. 2019, Aijaz et al. 2015). The Tactile Internet expands the possibility of transmitting skills through the Internet (Del Re et al. 2016). In the literature, it is possible to find several applications for this new paradigm such as Virtual Reality and Augmented Industrial Automation, Games and Education (Watch & August 2014).

In this new communication environment proposed by the new Internet technology, a communication channel can be defined to emulate an environment of contact and performance interactions with very low latency. An outline of this system can be seen in Figure 1.1 which shows the presence of local computational environments interconnected with the Tactile Internet. They can use a master-slave communication structure, for example, to create the environment and thereby build the bidirectional channel. In this way, the user can use manipulation devices such as gloves and robotic arms to interact with a virtual environment or with a machine (Junior et al. 2019). The user action on these systems is converted and processed by local computing devices and subsequently transmitted to the network. The tactile internet has several guarantees of functionality, but, local systems need to be adequate to receive this new technology. Thus, local devices can be a possible barrier to the construction of an environment of tactile interaction. Given that, it is important to emphasize that the feasibility of tactile interactions are affected by the local systems and their applications.

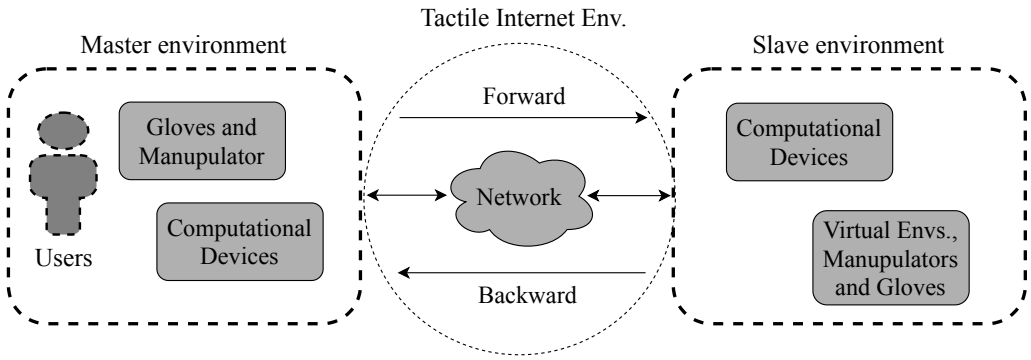

Figure 1.1: Basic model for tactile internet.

The computational systems highlighted in Figure 1.1, are responsible for executing several algorithms. These algorithms can be signal conversion algorithms, robotic control, prediction, among others. Many of these algorithms perform complex operations, that is, in each local computational system, there can be a high computing cost that can increase the processing time and so the data transmission latency. This increase in time is, in most cases, related to the increase in system latency.Besides, in cases in which the local devices are at continental distances, can be an increase in latency related to the time of propagation of information through the transmission environment. This increase in latency related to the connection can limit the time available for data processing on local systems.

Studies show that applications aimed at the tactile Internet can assume latency values between 1ms and 10ms for most cases or even values as 40ms for some cases (Fettweis 2013, Watch & August 2014, Fettweis 2014, Maier et al. 2016). Problems related to high latency are described (Fettweis 2014). The authors in Van Den Berg et al. (2017) and Wei et al. (2019) mentioned digital nausea (CyberSickness) as one of the main problems. This problem occurs when a user observing a virtual environment or a robotic element which he is manipulating has visual information that differs from tactile or motor information.

Several works already explore ways to minimize the latency generated by the situations already described. As is the case of the works (Aijaz 2016, Holland et al. 2016, Pilz et al. 2016, Dohler 2015, Simsek et al. 2016*a*) that describe possible ways to deal with the problem. But, most of the proposals presented use general-purpose processors. This approach can increase latency, and lose performance when executing complex computational techniques. Reconfigurable Computing (RC) and hardware-based systems prove to be an efficient way to develop more optimized solutions.

The Field Programmable Gate Array (FPGA) is a device widely used for RC. This technology has advantages due to its architectural flexibility and intrinsic parallelism associated. This device has a set of hardware such as memories, I/O and DSP, and high-performance features. Several works highlight great speedups using FPGAs (Coutinho et al. 2019*b*, Da Costa et al. 2019*b*, Da Silva et al. 2018, De Souza & Fernandes 2014*a*, Lopes et al. 2019*a*, Noronha et al. 2019*a*). Following this approach, this work proposes the use FPGA device to develop complex computational solutions such as the implementation of linear and non-linear techniques associated with the Tactile Internet as prediction and position control techniques.

1.1 Tactile System Model

The Figure 1.2 describes a model of a general tactile system. This system will be used as a basis for the development of this work. The structures present in computational devices can be developed in hardware. The blocks in dashed lines are the kinematics and dynamics structures related to the robotic system. However, the focus of the present work is the blocks in continuous lines in gray color containing prediction and control algorithms. The system starts with an operator, OP, responsible for transmitting stimuli to the Master Device, MD module. These stimuli are described as vectors of continuous signals, where $\mathbf{a}(n)$ which for this project can be described as a vector with in the Cartesian space described by

$$a(n) = [x^{op}(n), y^{op}(n), z^{op}(n)]. \tag{1.1}$$

where $x^{op}(n)$, $y^{op}(n)$, $z^{op}(n)$ are the variables of the spatial position of the tool (grab) after being moved by the manipulator operator.

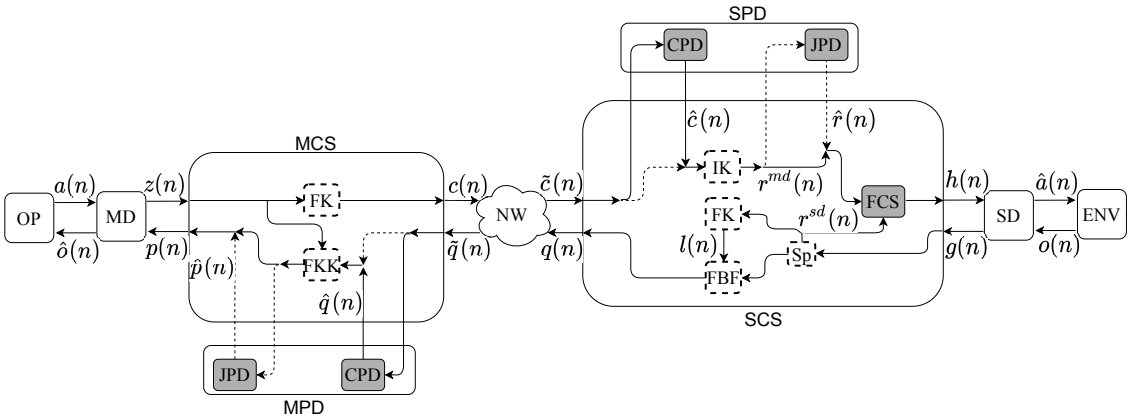

Figure 1.2: Basic model for tactile internet.

The MD is a haptic manipulator called PHANToM Omni (Silva et al. 2009*b*). Upon receiving the stimulus vector, $\mathbf{a}(n)$, PHANTHoM Omni performs an action that corresponds to the spatial stimuli received. This action is translated into discrete signals, through a set of transducers present in the device. Thus, the PHANTHoM Omni transducer set performs a transformation in the spatial coordinate signals to a discrete signal in joint coordinates called $\mathbf{z}(n)$. Where n is the n-th sample described by the sampling period, t_s, from the device. PHANToM Omni is a manipulator of three degrees of freedom, 3-DoF, which has three mobile joints (Al-Wais et al. 2016). Thus the sign $\mathbf{z}(n)$ can be expressed by

$$z(n) = [\theta_1^{md}(n), \theta_2^{md}(n).\theta_3^{md}(n)]. \tag{1.2}$$

The $\mathbf{z}(n)$ vector is then transmitted to the Master Computational System , MCS, which after receiving the signal $\mathbf{z}(n)$ transforms the data into joint coordinates for spatial coordinates. This process uses robotic kinematics algorithms called forward kinematics , FK (Junior et al. 2020). The output of this processing is a vector of discrete signals in the spatial plane called $c(n)$. These signals are then transmitted across the network. The signal $c(n)$ can be defined by

$$c(n) = [x^{mcs}(n), y^{mcs}(n), z^{mcs}(n)], \tag{1.3}$$

where $x^{mcs}(n)$, $y^{mcs}(n)$, $z^{md}(n)$ are the spatial position variables after processing the tool position.

The network, NW, is seen in this work as a system module. In this module, the signals suffer from delays and possible data loss, the $c(n)$ signal is then called $\tilde{c}(n)$, since it suffers from possible changes and is expressed by

$$\tilde{c}(n) = [x^{mcs}(n), y^{mcs}(n), z^{mcs}(n)]. \tag{1.4}$$

The module called the Slave Computational System , SCS, receive this signal vector $\tilde{c}(n)$ and it can go through two different paths. In the first path, highlighted in Figure 1.2 as a continuous line, the vector is pre-processed by the Slave Prediction Device (SPD) using Cartesian Prediction Device (CPD), which makes predictions in the Cartesian space.Thus, the module acts with the input data in the system, working with the $\tilde{c}(n)$ signs vector. In a second option, highlighted in Figure 1.2 as a dashed line, the signal is initially received by the Inverse Kinematics module, IK, which transmits a vector called $r^{mcs}(n)$ formed by $[\theta_1^{mcs}(n), \theta_2^{mcs}(n), \theta_3^{mcs}(n)]$ for other component of the SPD, called for the Joint Prediction Device (JPD), which makes predictions in the Joint space. There is little discussion in the literature about the positioning of the prediction module in a tactile system. this thesis propose the implements of the prediction for the both coordinate spaces.

The prediction module is responsible for minimizing the system latency. At this point, the module acts by making predictions in cases where the system is unable to meet the time requirements. Prediction techniques can be found in several works, such as techniques based on Linear regression, adaptive filters, Machine Learning (ML), among others for the most diverse purposes, such as market, industries, stocks, health, and communications (Gordini & Veglio 2017, Cui & Curry 2005, Kamakura et al. 2003, Srinivas et al. 2010, Bhatia et al. 2020, Shah et al. 2019, Thakur et al. 2018).

The application of predictive techniques in Tactile Internet already has its efficiency demonstrated in the works seen in (Wong et al. 2017, Briscoe et al. 2014, Ruan et al. 2018, Ruan & Wong 2018, Sakr et al. 2011, Brandi & Steinbach 2013). In the work done by the authors in (Briscoe et al. 2014), it presents a comprehensive research of techniques designed to deal with latency. One of the solutions adopted is the use of prediction techniques to minimize the impacts caused by delays and loss of information.

The output of the CPD is a vector of signals with predicted samples called $\hat{c}(n)$, which can be expressed by

$$\hat{c}(n) = [\hat{x}^{mcs}(n), \hat{y}^{mcs}(n), \hat{z}^{mcs}(n)], \tag{1.5}$$

where $\hat{x}(n)$, $\hat{y}(n)$, $\hat{z}(n)$ are the data predicted based on Cartesian information. For cases where the prediction occurs after the inverse kinematics module, the system delivers signals expressed by

$$\hat{r}(n) = [\hat{\theta}_1^{mcs}(n), \hat{\theta}_2^{mcs}(n), \hat{\theta}_3^{mcs}(n)], \tag{1.6}$$

where $\hat{\theta}_1^{mcs}(n)$, $\hat{\theta}_2^{mcs}(n)$, $\hat{\theta}_3^{mcs}(n)$ are predicted data based on information from Joint space.

Still looking at Figure 1.2, the next block in the diagram is the IK module. This module works transforming information from Cartesian space into information from Joint space. The output signal vector for this module is called $r^{mcs}(n)$, and is described by

$$r^{mcs}(n) = [\theta_1^{mcs}(n), \theta_2^{mcs}(n), \theta_3^{mcs}(n)], \tag{1.7}$$

The $r^{mcs}(n)$ signal is sent to the Feedback Control System (FCS) module. This module is responsible for performing the position control of the Slave Device , SD tool, using some control algorithm. There are several techniques for controlling the position of handling tools such as traditional techniques proportional – integral – derivative , PID, controller. The utilization of techniques based on Fuzzy Logic for position control of manipulation tools is found in the literature, but its use in controls applied in TI has not yet most been studied. The great advantage of using this type of technology is its ease in dealing with non-linear situations, unlike some classical techniques.

The FCS receives another vector of signals with the positioning of the SD tool in joint coordinates called $r^{sd}(n)$ composed of the signals $[\theta_1^{md}(n)$, $\theta_2^{md}(n)$, $\theta_3^{md}(n)]$. The control system as a function to minimize the error vector $e^{FCS}(n)$. The error vector is given by

$$e^{FCS}(n) = r^{mcs}(n) - r^{scs}(n). \tag{1.8}$$

The FCS module outputs the torque vector $h(n)$ that controls the joints of the SD manipulator. The signal vector consists of

$$h(n) = [\tau 1(n), \tau_2(n), \tau_3(n)], \tag{1.9}$$

where $\tau_1(n)$, $\tau_2(n)$, $\tau_3(n)$ are the torque values that will be applied to each of the joints of the slave manipulator. At this time the spatial position of the slave manipulator is an estimate of the position of the master manipulator. So the vector that describes this positioning can be described as follows

$$\hat{a}(n) = [\hat{x}^{op}(n), \hat{y}^{op}(n), \hat{z}^{op}(n)], \tag{1.10}$$

where $\hat{x}^{op}(n)$, $\hat{y}^{op}(n)$, $\hat{z}^{op}(n)$ are the values in cartesian coordinates of the tool position of the slave device. This ends the data flow called forward and begins the backward flow. The SD when moving interacts with the environment to which it is exposed, this allows contact with objects that are within reach.

The contact with these objects can be given by the contact force necessary to act in a certain location of the device tool. This force vector returns as feedback from the environment and are described by

$$o(n) = [F_x^{env}(n), F_y^{env}(n), F_z^{env}(n)]. \tag{1.11}$$

The SD in turn sends a vector containing the position of the tool in joint coordinates and the contact feedback called $g(n)$. This vector is split into two, where $r^{sd}(n)$, which contains the joint coordinates of the SD, is transmitted the FCS module and to the FK module. Contact feedback determines the spatial position in Cartesian coordinates of the

object with which the SD came into contact, being called $Fbc^{env}(n)$. The FK in turn performs a process analogous to that performed in the MCS, transmitting a vector of signals in Cartesian coordinates called $l(n)$, to the Feedback Force module, FBF. The $g(n)$ vector can be expressed by

$$g(n) = [r^{sd}(n)Fbc^{env}(n)], \tag{1.12}$$

where $r^{sd}(n)$ is given by

$$r^{scs}(n) = [\theta_1^{scs}(n), \theta_2^{scs}(n), \theta_3^{scs}(n)], \tag{1.13}$$

and $Fbc^{env}(n)$ is given by

$$Fbc^{env}(n) = [x^{env}(n), y^{env}(n), z^{env}(n)], \tag{1.14}$$

The vector $l(n)$ can be expressed by

$$l(n) = [x^{env}(n), y^{env}(n), z^{env}(n)]. \tag{1.15}$$

The FBF module then receives the signal vectors $l(n)$ and $Fbc^{env}(n)$ and determines the signal $q(n)$ that contains the force values acting on the tool as opposed to the actuation operator. The $q(n)$ sign is expressed by

$$q(n) = [F_x^{scs}(n), F_y^{scs}(n).F_z^{scs}(n)]. \tag{1.16}$$

The signals are then sent by the NW module in a similar way to the forward process. The vector $\tilde{q}(n)$ is received by the MCS. Thus, a prediction process similar to that of the forward flow occurs but using force values. In the CPC module, the output $\hat{q}(n)$ containing force predictions in Cartesian variables is output. The JPC module outputs the vector $\hat{p}(n)$ with the values in joint coordinates. The Kinesthetic Feedback Force (KFF) module is responsible for transforming the force values opposite direction to the tool movement of the slave device into a torque vector. Thus the vector $p(n)$ can be expressed by

$$p(n) = [\tau 1(n), \tau_2(n), \tau_3(n)]. \tag{1.17}$$

After receiving the torques, the MD passes on the touch sensation to the OP that receives the vector $\hat{o}(n)$. This vector represents the estimate of the force that the object in contact exerts on the tool of the slave device.

$$\hat{o}(n) = [\hat{F}_x^{env}(n), \hat{F}_y^{env}(n).\hat{F}_z^{env}(n)]. \tag{1.18}$$

Thus, this thesis aims to show how the implementation of techniques associated with control and prediction systems in reconfigurable computing can reduce the latency of systems aimed at tactile internet. Two works are proposed, the first of which explores the creation of dedicated hardware for position control based on fuzzy logic. The controller created is of the Fuzzy-PI type. The second work deals with the development of dedicated hardware for prediction. In this work, several prediction techniques, linear and non-linear,

were tested, such as linear regression and based on machine learning. Both works were tested to control and make predictions for a Phanthom Omni robotic manipulator.

1.2 Objectives

The objective of this thesis is to contribute to the community in the development of new proposals for Tactile Internet. The proposals are associated with the implementation in Hardware of techniques associated with control and prediction systems using reconfigurable computing, more specifically assisting in reducing latency of systems based on Tactile Internet with the use of hardware implementations.

The specific objectives of this work are:

- Develop reconfigurable hardware using parallelism techniques in FPGA of the hardware modules that implements position control algorithms associated with haptics devices;
- Develop reconfigurable hardware using parallelism techniques in FPGA of the modules that implement linear e non-linear predictions techniques.
- Demonstrate through experiments, the viability of the strategies using a robotic manipulator data.
- Demonstrate the viability of the strategies comparing with other works.

1.3 Submitted and Published Articles

- SILVA, Sérgio N. et al. Proposal of Takagi–Sugeno Fuzzy-PI Controller Hardware. Sensors, v. 20, n. 7, p. 1996, 2020.

1.4 Thesis Outline

This thesis is organized in 4 chapters, as presented in the following paragraphs. In this first chapter, an introduction was presented, in which the motivation and theme of the work are contextualized, as well as the main objectives of the research and published articles. Chapter 2 proposes the use of prediction techniques implemented in FPGA for tactile internet systems. Several prediction techniques between linear and non-linear are developed. In addition to the results of the validation of the proposed hardware modules, the post-synthesis results of each implementation will be presented, as well as comparisons with other works in the literature.

The Chapter 3 presents a Takagi - Sugeno Fuzzy-PI Controller. The system is developed in FPGA and uses a completely parallel strategy. Several bit settings for the Fuzzy-PI controller are analyzed. In addition to the results of the validation of the proposed hardware modules, the post-synthesis results of each implementation will be presented, as well as comparisons with other works in the literature.

Finally, Chapter 4 presents the final considerations, showing the conclusions about the results obtained and the possibility of future work.

Chapter 2

Prediction techniques in RC for latency reduction in IT

This chapter aims to present, analyze, and evaluate a hardware reference model that uses reconfigurable computation (RC) for prediction module that implement linear and nonlinear prediction methods. This chapter presents the implementation details, simulations, and experimental tests to validate the proposed model. Besides, analysis regarding the sample rate, throughput, latency, and area occupancy, are also presented based on post-synthesis results.

2.1 Introduction

The called Tactile Internet is the name given to the new technological paradigm for the Internet. In this new advancement in communications, it is possible to transmit the sensation of touch through the Internet, as well as video, audio, and text data (Dohler 2015). Thus, Tactile Internet systems will be responsible to providing solutions to more complex computational problems such as human-machine interactions (H2M) in real-time (Ateya et al. 2019, Aijaz et al. 2015). So, a new communication concept is inserted in which it is possible to transmit skills through the Internet (Del Re et al. 2016).In the literature, it is possible to see several applications for this new paradigm such as Virtual and Augmented Reality, Industrial Automation, Games, and Education (Watch & August 2014).

With such a wide application environment, some initial studies for tactile internet already demonstrate that it will be necessary to guarantee very low latency (Fettweis 2013, Watch & August 2014, Fettweis 2014, Maier et al. 2016). Some of these studies show that the latency for applications on the tactile Internet can be between 1 - 10ms in most cases or up to 40ms in some cases. Problems related to high latency are described in (Fettweis 2014), which shows that cybersickness is one of the main related problems (Van Den Berg et al. 2017, Wei et al. 2019).

Several works already explore ways to minimize the problems related to the latency generated by situations described, as described in the works presented by Aijaz (2016), Holland et al. (2016), Pilz et al. (2016), Dohler (2015), and Simsek et al. (2016*a*) which describes the possible ways to deal with the problem. Work like that of the authors at

(Briscoe et al. 2014) offers a comprehensive survey of techniques designed to deal with latency. One of the solutions adopted is the use of prediction techniques to minimize the impacts caused by delays and loss of information. In this case, the system works by hiding a real network latency, making predictions of the user's behavior. Hiding latency does not reduce actual latency, but it can greatly improve the quality of the user experience.

Prediction techniques have been used over the years for the most diverse purposes, such as market, industries, stocks, health, and communications (Gordini & Veglio 2017, Cui & Curry 2005, Kamakura et al. 2003, Srinivas et al. 2010, Bhatia et al. 2020, Shah et al. 2019, Thakur et al. 2018). Nowadays it is possible to find works with the most varied techniques and applied to the most variable platforms. The work in Ababei & Moghaddam (2018) shows the implementation of prediction techniques in a multicore environment. The works (Karvelis et al. 2018, Widiasari et al. 2017), show the use of prediction techniques in systems based on microcontrollers. The works presented in (Zhao et al. 2013) and (Xie et al. 2020) develop prediction techniques implemented in GPUs. But, various of the proposals seen in the literature are aimed at software implementations. Thus, the use of these approaches to implement techniques with greater computational complexity or for cases that have a large set of data to be processed, there may be an increase in the latency of the computer systems that make up the tactile link.

Some works in the literature recommend the use of reconfigurable computing-based systems such as field-programmable gate arrays (FPGAs) for implementing complex techniques (De Souza & Fernandes 2014*a*). This approach can enable a performance gain in the computer systems that constitute the tactile system it is possible to develop Hardware (HW) solutions with a specific purpose and great processing power. In the literature, it is possible to find works based on FPGA that reach 1000× speedup compared to proposals based on software (Coutinho et al. 2019*b*, Da Costa et al. 2019*b*, Da Silva et al. 2018, Lopes et al. 2019*a*, Noronha et al. 2019*a*).

Thus, this work proposes the implementation of prediction techniques in reconfigurable hardware applied to the Tactile Internet. The work proposes the implementation of linear techniques such as Zero-order prediction, and linear Regression and non-linear such as Multlayer Perceptron - Backpropagation (MLP-BP) using different bit configurations. The aim is to show that the use of prediction techniques applied to systems based on reconfigurable computing is a possible solution to reduce the impacts of latency in Tactile Internet. The implementations will be compared with software implementations with floating-point precision. Comparisons with other works in the literature show that the use of reconfiguration computing can speed up the processing speed in tactile devices.

2.2 Related Work

Some works show ways to treat latency in systems for tactile internet. For example, the study by the authors at (Wong et al. 2017) proposes to perform prediction to assist the bandwidth allocation process automatically by the server. This approach can also be seen in the work of the authors in (Ruan & Wong 2018) and (Ruan et al. 2018). However, the systems presented in these proposals are local and may not be scalable for use in more complex networks with higher traffic, as they need a set of information from all commu-

nications to perform the configuration and training of the techniques. Other authors show that linear prediction techniques are favorable for cases where there is a loss of packages or even packages with errors and that must be discarded(Sakr et al. 2011, Brandi & Steinbach 2013).

However, many of the works currently developed for Tactile Internet as well as those already mentioned, use software-level implementations. This type of approach can negatively affect the processing time of the data by the prediction techniques. Some studies show that the gains with the use of systems based on reconfigurable computing (HR), such as FPGAs, to improve communications need an accurate *feedback* as haptic system communications, as seen in (O'Malley et al. 2009, Tanaka et al. 2008, Rebello & Sriram 2004, Galvan et al. 2006). The work developed by the authors in (Oballe-Peinado et al. 2017, Hartley & Maciejowski 2013, Dorfling et al. 2019, Tu et al. 2019) aims to increase the performance of manipulative tools using FPGA-based platforms.

Thus, using implementations of prediction techniques on dedicated hardware, such as HR, can be one of the possible ways to reduce latency in computer systems. An implementation of quadratic prediction technique based on FPGA regression was developed by the authors in (Bellemare-Rousseau et al. 2020). In (Wienbrandt et al. 2019) the authors achieved gains of 1000 - 1600 $\times$ with the implementation of a technique for detecting epistasis based on logistic regression with an FPGA combined with GPU. However, few studies explore the use of linear regression applied to FPGA signal prediction. In (Carpeño et al. 2019) that propose the implementation of a probabilistic predictor in an FPGA.

The use type of HR for computationally more complex algorithms is already studied in the literature, as can be seen in the work carried out by the authors at (de Souza & Fernandes 2014*b*) which displays a set of occupation results and processing and occupation time for various RNA configurations of Functions Radial Bases. In (Zhang et al. 2015) and (Yu et al. 2015), which demonstrate the feasibility of implementing algorithms based on Deep Learning (DL) using an HR-based platform. However, there are few studies on the use of predictors in hardware applied to systems for tactile internet, but it is possible to find studies that implement ML techniques such as MLP in FPGA as in (Bahoura 2018) present an architecture for the identification of wheezing in the auscultation of lung sounds in real-time. The proposed system uses an artificial neural network of the MLP type. The training of the MLP method is offline. The implementation uses an FPGA Artix-7 at a fixed point of 36-bits. The topology of the MLP is 12 inputs, 12 neurons in the hidden layer, and 2 neurons in the output layer. The system throughput was 8.63ns.

In (Gaikwad et al. 2019) the authors present a human activity recognition system (HAR) based on hardware dedicated to smart military garments and use a multilayer perceptron algorithm (MLP) to perform the activity classification. The proposed MLP project has 7 inputs, 6 neurons in the hidden layer, 5 neurons in the output layer. Five versions of the MLP project are implemented in FPGA with the same architecture (7-6-5), but with different data precision. The analysis shows that MLP designed with 16-bit fixed-point data accuracy is more efficient in the context of classification accuracy, resource utilization, and energy consumption. The proposed MLP project has a throughput of 270ns

using about 90% of the embedded multipliers.

The work in (Zhai et al. 2016) proposed the implementation of a multi-layer perceptron (MLP) in FPGA ZYNQ SoC for the classification of gases with low latency and in real-time. An MLP with 12 inputs, 3 neurons in the hidden layer, and one neuron in the output layer is proposed. The neural network is trained offline by the Levenberg-Marquardt backpropagation algorithm. The implementation use the High Level Syntheses (HLS) on Vivado for optimize the time development. The trained weight data is done using signed fixed-point representation 24-bit total length with 20 fractional bits. For output, the layer is implemented in 16-bit total length with 14 fractional bits using the Tanh function. The hardware used was a Xilinx Zynq-7000 XC7Z010T-1CLG400 with a the throughput of the implementation was 539.7ns.

The work in (Bahoura 2016) propose the implementation of a multi-layer perceptron (MLP) in FPGA ZYNQ SoC for automatic blue whale callsclassification. An MLP with 12 inputs, 7 neurons in the hidden layer, and 3 neurons in the output layer is proposed. The neural network is trained offline by the backpropagation algorithm. The trained weight data is done using signed several fixed-point representation with 24-bit max length. For output, logistic sigmoid function. The hardware used was a Xilinx Virtex 6 XC6VLX240T and Artix-7 XC7A100T with a the throughput of the implementation of 27.89Msps and 25.24Msps.

Differently from the other works, the present article designs hardware for prediction techniques applied to tactile internet. The work also proposes the implementation of linear and non-linear prediction. Using HR as a way to reduce latency and, also, make it possible to use more complex techniques. For the linear techniques it is proposed techniques such as zero-order prediction, and based on linear regression. In these cases the work use using float-point standard IEEE 754. The work still presents four solutions for different ranges of the regression buffer. In non-linear techniques it is proposed an MLP-BP prediction technique. For these propose to use several configurations of fixed-point precision. This work also utilizes online training to update the weights of the neural network. The work uses the Phantom Omni data to validate the application of the techniques. The results showed good performance in comparison with implementations in Matlab using float-point double precision.

2.3 Proposal Description

In the new communication paradigm, it is possible to send the sensation of touch through the Internet. As in the laws of physics, there is an action and reaction process. The user interacts with a virtual environment or a physical tool over the network. Figure 1.2 in section 1.1 shows the general tactile internet system. In this Figure, two devices interact with each other. The devices can be the most diverse, such as manipulators, virtual environments, tactile or haptic gloves, among others. During the *forward* flow, the master device, MD, sends signals to the slave device, SD. Meanwhile, *backward*, the SD sends return signals (*feedback*) to the MD.

The devices have their own subsystem, master and slave subsystems. In this way, at each side of the communication process, it identifies the computer systems of the master

and the slave (MCS and SCS). These systems are responsible for performing computations related to data processing, control, robotics, prediction algorithms, among others. Each of these algorithms has an execution time that assuming a sequential algorithm of these algorithms, the total time can be given by the sum of the individual time of each algorithm.

The model that will be adopted in the present work, considers that the computational systems are constituted of diverse algorithms. Each of these algorithms adding latency to the system. Thus, the prediction process must occur in parallel with the other algorithms embedded in the MCS and SCS.This consideration aims to decouple the prediction techniques from the other algorithms and facilitates the analysis. Thus, based on the Figure 1.2, the Figure 2.1 and 2.2 illustrates a model for predictions on tactile internet systems. The model uses prediction methods in parallel with computational systems. The prediction modules, called MPD and SPD, has the same signal inputs as its respective computational systems. The path choice in this project is to calculate the Cartesian predictions before the kinesthetic and kinematics module, analogous Figure 1.2. The Signal $\tilde{\mathbf{q}}(n)$ for *MPD* and $\tilde{\mathbf{c}}(n)$ for *SPD*. The module *MPD*, upon receiving the input vector, predicts a vector called $\hat{\mathbf{q}}(n)$. This prediction has a processing time of t_{mpd}. In turn, the slave subsystem's prediction module predicts the $\hat{\mathbf{c}}(n)$ size V,vector, with a prediction processing time of t_{spd}.

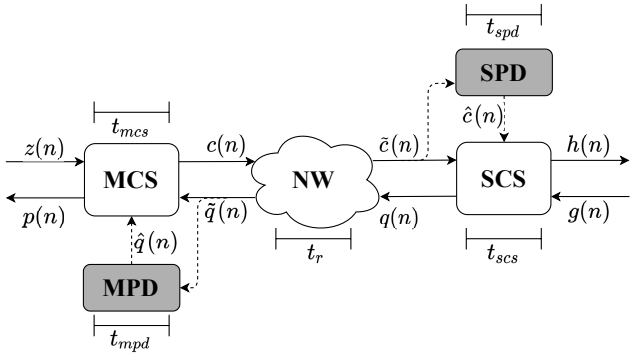

Figure 2.1: Prediction module in parallel with the computer system.

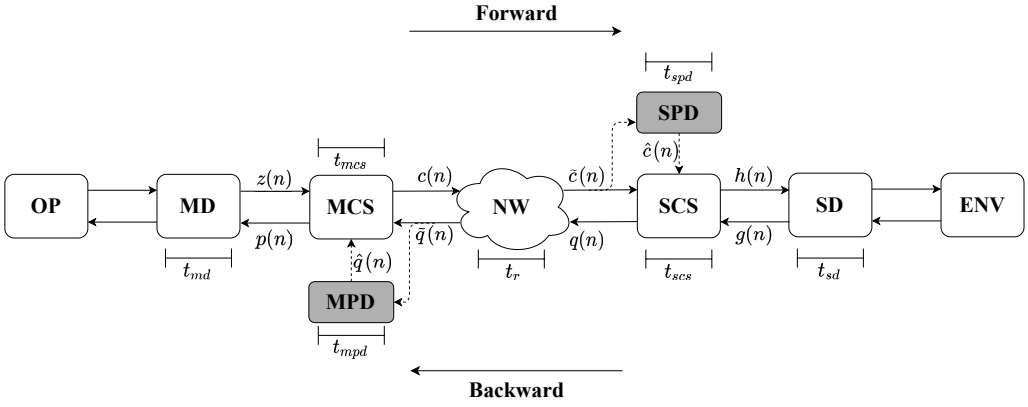

Figure 2.2: Generic Tactile Internet system with parallel prediction method.

2.4 Prediction Methods

This section describes the prediction methods used on this proposal. The methods used in this proposal are linear such as zero-order prediction, and prediction based on linear regression. Also, non-linear methods are based on ML of the Multi-Layer Perceptron and Backpropagation Algorithm (MLP-BP).

As shown in Figure 2.2, the module responsible for the predictions system can be defined in both computational systems. The module is illustrated in Figure 2.3, it is possible to see that the module can implement several different prediction methods. The prediction module can work both by computing predictions in Cartesian and joint coordinates as described in (Junior et al. 2020). In that case, parallel implementations will be necessary. The implementations may or may not be of the same technique replicated several times. In order to create a metric to define the capacity of the hardware, only the same technique was replicated. Thus, NI represents the number of implementations of this technique in parallel.

This NI value may vary according to the degrees of freedom of the virtual environment or robotic manipulator model. The methods can be the same for all cases or different techniques can be used for each of the variables analyzed such as Non-linear Prediction Methods (NLPM), Linear Prediction Methods (LPM), and Probabilistic Prediction Methods (PPM), among other. In this thesis, prediction methods based on linear and non-linear models were developed.

The system has two data streams, *forward* and *backward*, which work with the signal vectors $c(n)$ and $q(n)$. From this section, these signals will be represented in this chapter by $\upsilon(n)$ which describes the input samples for the two vectors, as well as $\hat{\upsilon}(n)$ describes the predicted samples. The bit configuration depends on the type of implementation.

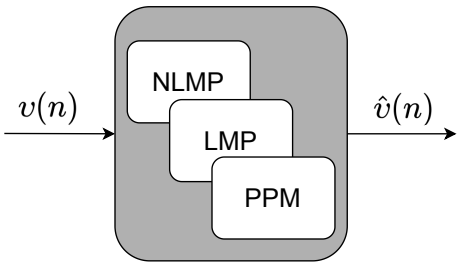

Figure 2.3: Structure of the prediction module.

2.4.1 Zero-Order Prediction

In the prediction of zero-order method transmit as a sample predicts the last real sample. The Zero-order method will replicate this sample until the system can update the value of the real sample again. The Equation 2.1 the describes this method is expressed by

$$\hat{\upsilon}(n) = \upsilon(n-1), \tag{2.1}$$

where $\upsilon(n-1)$ is the last value received and $\hat{\upsilon}(n)$ is the estimate of the real value of υ.

2.4.2 Linear Regression

Another possible prediction method is using simple linear regression. This method uses a set of past samples, defined by M to infer possible predicted data. This approach uses a set of observed pairs composed of the time marker t_m and the dependent variable,υ , that is, $(t_m(1),\upsilon(1)),(t_m(2),\upsilon_m(2)),...,(t_m(M-1),\upsilon(M-1)),(t_m(M),\upsilon(M))$. This type of regression can be defined by Equation 2.2.

$$\hat{\upsilon}(n) = \hat{\beta}_0(n) + \hat{\beta}_1(n)t_m(n), \tag{2.2}$$

where $\hat{\upsilon}(n)$ is the data that approximates a given $\upsilon(n)$. $\hat{\beta}_0(n)$ is the linear coefficient of the estimate and $\hat{\beta}_1(n)$ is the angular coefficient of this same estimate. The principle of least squares is used in the process of parameter estimation (Montgomery et al. 2012). Such coefficients can be seen in Equations 2.3 and 2.4

$$\hat{\beta}_0(n) = \bar{\upsilon}(n) - \hat{\beta}_1(n)\bar{t_m}(n), \tag{2.3}$$

where $\bar{\upsilon}(n)$ and $\bar{t_m}(n)$ are the average values of the sample variables υ and t_m that can be calculated as follows

$$\hat{\beta}_1(n) = \frac{\sum_{j=0}^{M}(t_m(n-j)-\bar{t_m}(n))(\upsilon(n-j)-\bar{\upsilon}(n))}{\sum_{j=0}^{M}(t_m(n-j)-\bar{t_m}(n))^2}. \tag{2.4}$$

2.4.3 Multi Layer Perceptron networks

Solutions based on Machine Learning are increasingly used to solve complex problems. Among which the solutions based on Artificial Neural Networks (ANN) most used. ANNs are computational techniques based on the neural structure of intelligent organisms. These structures can learn from past and current experiences. These organisms are made up of extremely complex cells, called biological neurons. The mathematical structure of the ANN, in turn, is composed of processing units, called artificial neurons. The neurons can operate in parallel and distributed (Haykin 1998). This parallelism paves the way for solutions that exploit this characteristic as systems based on FPGAs.

Architecture

The architecture of a MLP-BP is used in several applications based on neural networks (Rumelhart et al. 1986). The reason is these systems have the ability to deal with non-linearly separable problems. Equation 2.5 represents the prediction function using the Multi-Layer Perceptron MLP technique. This function use B past samples of υ to generates the $\hat{\upsilon}(n)$ value.

$$\hat{\upsilon}(n) = f\left(\upsilon_{n-1},\upsilon_{n-2},...\upsilon_{n-B}\right), \tag{2.5}$$

where $\upsilon_{n-1},\upsilon_{n-2},...\upsilon_{n-B}$ are the input values of the MLP and $\hat{\upsilon}$ is the output predicted of the MLP. Equation 2.6 presents a generic MLP with L layers, where each k-th ($k =$

$1,\ldots,L$) layer can have N_k neurons with $N_{k-1}+1$ inputs, N_{k-1} is the neurons number of the previous layer. The neurons from k-th layer process their respective input and output signals through an activation function $f_k(\bullet)$. At the n-th sample, this function is given by

$$y_i^k(n) = f_k\left(x_i^k(n)\right), \tag{2.6}$$

where $y_i^k(n)(i=1,\ldots,N_k)$ is the output for the i-th neuron in the k-th layer and $x_i^k(n)$ can be represented by

$$x_i^k(n) = \left(\sum_{j=1}^{N_k} w_{ij}^k(n) y_j^{k-1}(n)\right) - w_{i0}^k(n), \tag{2.7}$$

where $w_{ij}^k(n)$ is the synaptic weight associated with j-th input of the i-th neuron, illustrated in Figure 2.5.

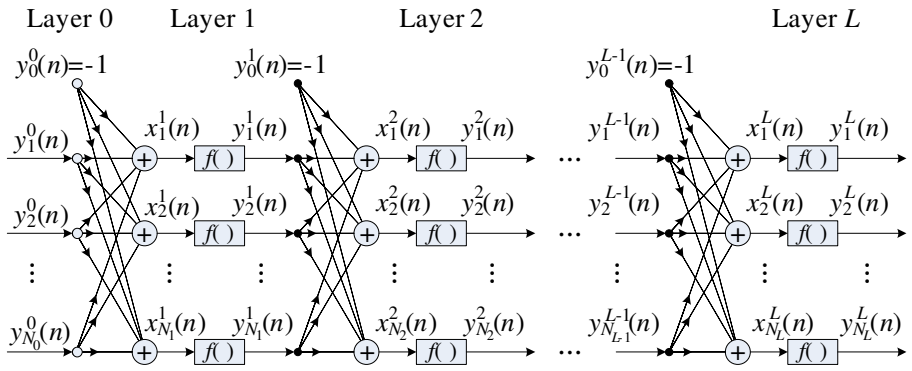

Figure 2.4: Structure of a ANN of the type MLP, with L layers.

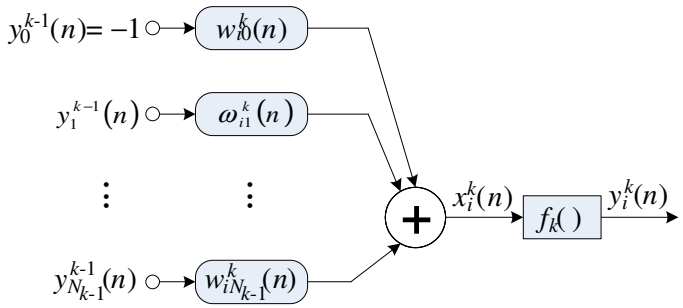

Figure 2.5: Structure of a Perceptron with $Nk-1+1$ inputs.

The function $f_k(\bullet)$ is chosen as

$$f_k(x) = max\{0,x\}, \tag{2.8}$$

named Rectified Linear Unit (ReLU) function, and the backpropagation algorithm is the training algorithm used with MLP.

Training algorithm - Backpropagation Algorithm

The weights are updated with the error gradient descent vector. At the n-th iteration,the error signal for the i-th neuron in the k-th layer is defined by

$$e_i^k(n) = \begin{cases} d_i(n) - y_i^k(n) & \text{for } k = L \\ \sum_{j=0}^{N-1} w_{ij}^{k+1}(n)\delta_i^{k+1}(n) & \text{for } k = 1 \ldots L-1 \end{cases}, \tag{2.9}$$

where $d_i(n)$ is the desired value and $\delta_j^{k+1}(n)$ is the local gradient for the i-th neuron in the $(k+1)$-th layer at the n-th iteration defined as follows:

$$\delta_i^{k+1}(n) = \begin{cases} e_i^k(n) f'(y(n)) & \text{for } k = 0 \ldots K-2, \end{cases}, \tag{2.10}$$

where $f'(y(n))$ is the derivative of the activation function.

The synaptic weights are updated according to the following equation

$$w_{ij}^k(n+1) = w_{ij}^k(n) + \eta\delta_j^k(n)y_j^k(n) + \alpha w_{ij}^k(n-1), \tag{2.11}$$

where η is the learning rate, α is the regularization term or penalty term and $w_{ij}^k(n+1)$ is the updated synaptic weight to be used in the next iteration.

2.5 Implementation Description

The structures of the hardware implementations of the prediction methods are described in this section. The hardware-implemented uses two types of precision methods. Some prediction methods use a fixed-point format with several configurations (non-linear methods). The other part using 32-bit floating-point (IEEE754) standard (linear methods).

For the linear prediction methods used the notation [F32] for fixed-point precision. Non-linear prediction methods used the notation [sT.W] for 32-bit floating-point precision. This notation indicates that the variable has in total T bits of which W bits for the fractional part and used one bit, s, to determine the sign of the variable.

2.5.1 Zero-Order Prediction

The generic structure for the hardware implementation in FPGA of the zero order linear prediction method, described in Equation 2.1, is represented in Figure 2.6. The hardware used one single register to describe the past sample of the variable $\upsilon[F32](n)$. The output is the predicted signal $\hat{\upsilon}[F32](n)$.

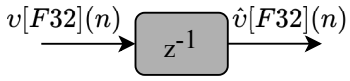

Figure 2.6: Structure of the zero order prediction method.

2.5.2 Linear Regression

The hardware structure implementation for the linear prediction technique based on linear regression was developed in FPGA following Equations 2.2, 2.3 and 2.4 already described in the section 2.4.2. The generic structures use float-point of 32-bits precision in all circuits.

The hardware structure based on Equation 2.2 for FPGA implementation is showed in Figure 2.7. The circuit has tree inputs values ($t_m[F32](n)$, $\beta_0[F32](n)$, $\beta_1[F32](n)$), and one output value ($\hat{\upsilon}[F32](n)$). The structure used one multiplier and adder to perform the equation.

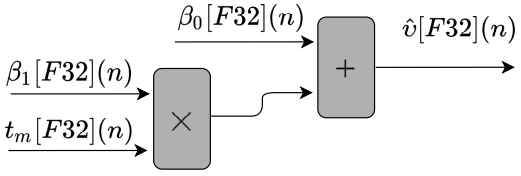

Figure 2.7: Hardware Structure for prediction using linear regression.

In Figure 2.8 it is showed the structure based on Equation 2.3. The circuit has tree inputs values ($\bar{t_m}[F32](n)$, $\beta_1[F32](n)$, $\bar{\upsilon}[F32](n)$), and one output value ($\beta_0[F32](n)$).

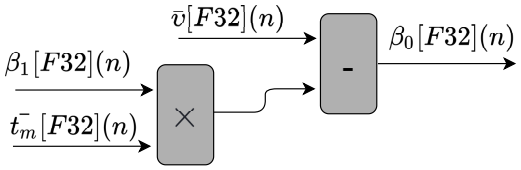

Figure 2.8: Hardware Structure for β_0 calculation in the linear regression prediction method.

The hardware structure based on Equation 2.4 is shown in Figure 2.9. In this case the structure used two multipliers, one subtractor, cascading sum (CS), and two constant values. The circuit has two inputs values ($\upsilon[F32](n)$, $\bar{\upsilon}[F32](n)$), and one output value ($\beta_1[F32](n)$).

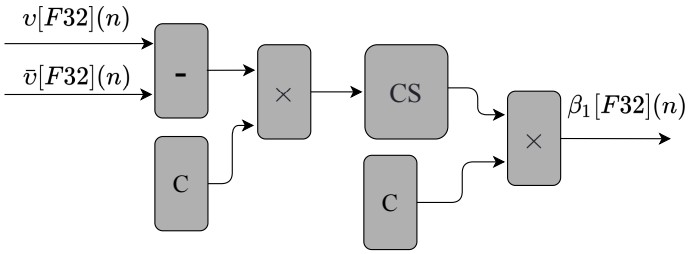

Figure 2.9: Hardware Structure for β_1 calculation in the linear regression prediction method.

The hardware generic structure used to implement the sum and mean calculate is showed in Figures 2.10 and 2.11.

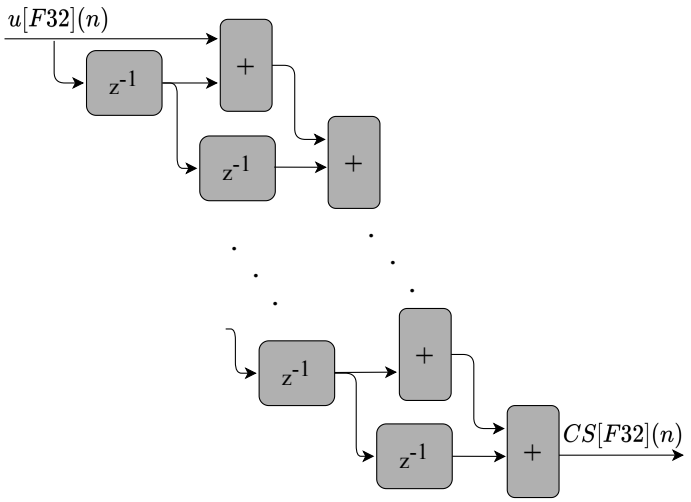

Figure 2.10: Hardware structure for cascading sum calculation.

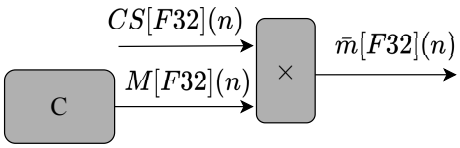

Figure 2.11: Hardware structure for mean calculation.

2.5.3 Multi Layer Perceptron

The general structure of the proposed hardware implementation for MLP-BP and RMLP-BP. This last solution uses the same structure, but one external input is change by one output signal. The architecture has two main modules called Multi-Layer Perceptron Module (MLPM) and Backpropagation Module (BPM). Figures 2.12 and 2.13

represent the hardware circuits proposed. The hardware uses a fixed-point format for all the variables. For any given variable, the notations [sT.W] show that the variable has T bits of which use W for the fractional part. The symbol "s" indicate that the variable use one bit for signed.

The circuit proposed in Figure 2.12 has B inputs of past samples of the υ variable. The proposed circuit for the second proposal has $B-1$ inputs of past samples of the υ variable. The inputs υ also are used in BPM, and samples are transmit between the modules MLPM and BPM. In the processes of transmission, the signals pass for one delay unit. The BPM also receive other signals from MLPM called $y_i^k[sT.W](n)$. These signals are the values produced at each output of the neurons. In the processes of update, the BPM sends the signals $w^k_{N_k,N_{k-1}}[sT.W](n)$ of new weight values. The BPM also has an input called $d_i[sT.W](n)$, which is the desired value for the MLP output. The MLP must then learn the behavior of a time series, therefore, the desired value is the same as the actual current sample or $d_i[sT.W](n) = \upsilon[sT.W](n)$. Subtracting this signal, $\upsilon[sT.W](n)$, from the MLP output value produces an error signal represented the variable $e[sT.W](n)$.

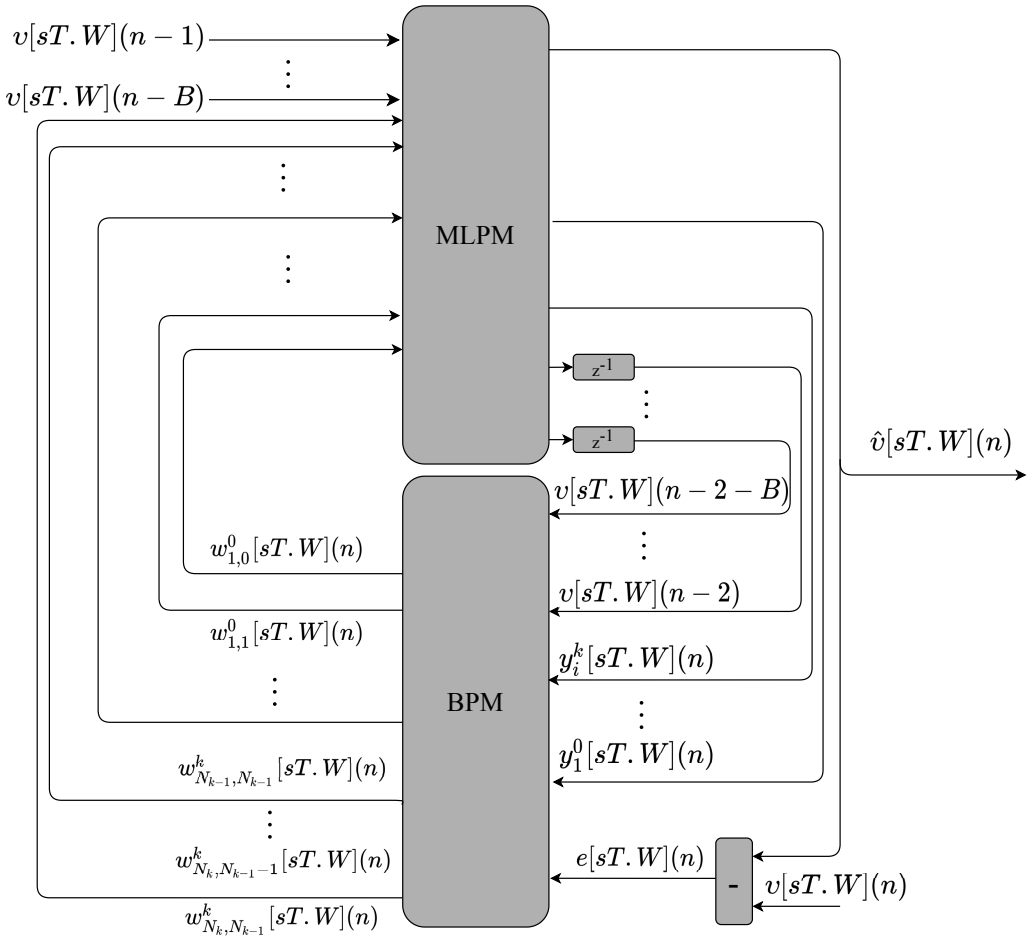

Figure 2.12: Hardware structure for MLP-BP.

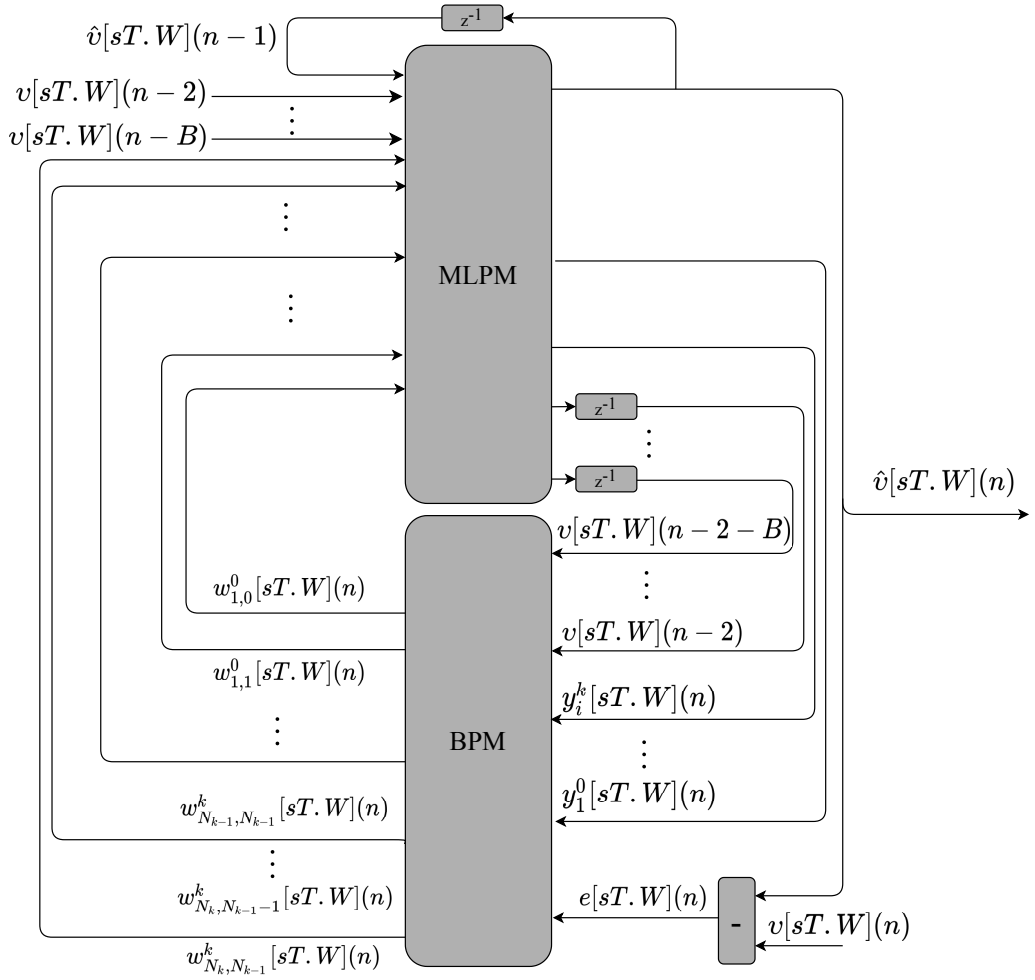

Figure 2.13: Hardware structure for RMLP-BP.

Neuron

The MLPM used various neurons for to create a network structure as is showed in the Figure 2.4. Based on Equation 2.6 and 2.7, and Figure 2.5, the neuron structure is showed in the Fig. 2.14. The model is used in the hidden layers of the network. The circuit is a semi-parallel implementation scheme for a neuron with 10 inputs values, 5 inputs of $\upsilon[sT.W](n)$ values and bias value ($\upsilon^0_0[sT.W](n)$), and 5 inputs performing the weight values $w^k_{N_k,N_{k-1}}[sT.W](n)$. The linear combination composed of adders and multipliers generates the output $x^k_i[sT.W](n)$. The signal $x^k_i[sT.W](n)$ is the input of the non-linear function called ReLU, describe in Equation 2.8. Figure 2.15 shows the generic FPGA implementation for ReLU function. In the output layer, neurons used a nonlinear activation function. The linear combination between weight and output hidden layers gives the output from the neural network.

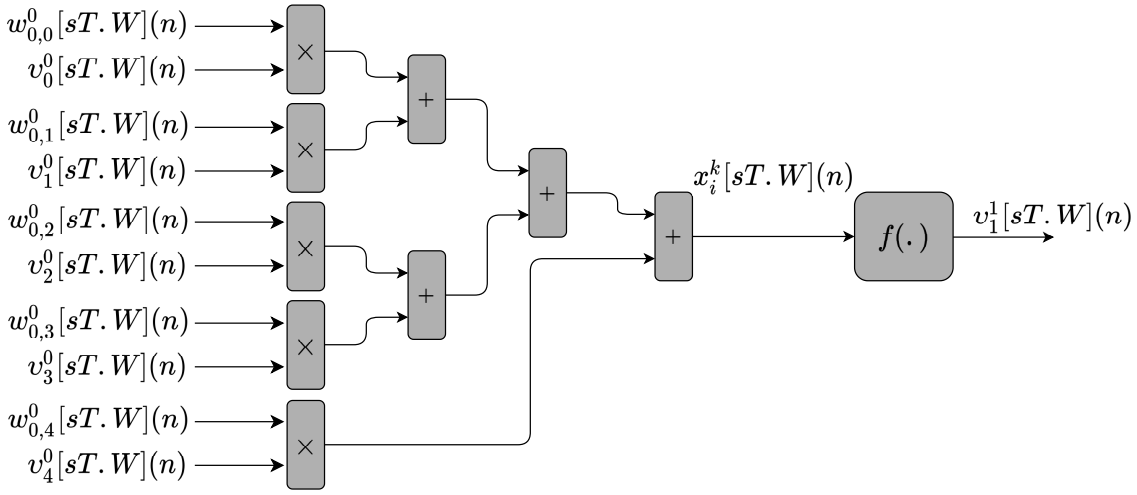

Figure 2.14: Hardware structure for the neurons.

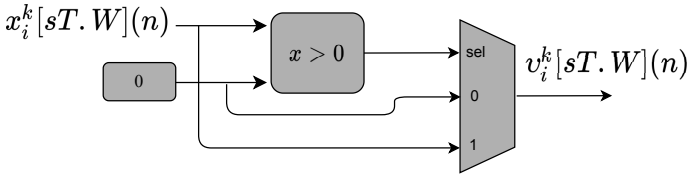

Figure 2.15: Hardware structure for ReLU function.

Backpropagation Module - BPM

The generic structure of the Backpropagation for FPGA is proposed. The gradient is calculate, according to equations 2.9 and 2.10 is presented on Figure 2.16. For the MLP output layer, the gradient is the error value itself, that is, the gradient is given by $e[sT.W](n)$. For the other layers, the gradient is calculated as shown in Figure 2.16. The circuit uses the signals $w^k_{i,j}[sT.W](n)$ and $\delta^{k+1}_j[sT.W](n)$ for calculate the output value of the gradient defined for $\delta^k_j[sT.W](n)$.

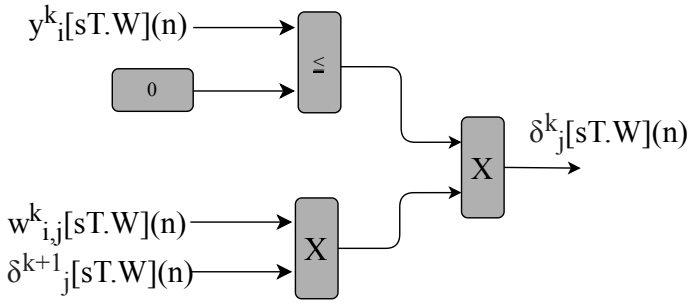

Figure 2.16: Hardware structure for hidden layers gradient.

Figure 2.17 as Equation 2.11 it is updated the weight of the MLPM. The circuit has two inputs, $y_i^k[sT.W](n)$ and $e[sT.W](n)$ for the performed the actualization the weight. The constants α and η are defined using fixed-point precision $[sT.W]$.

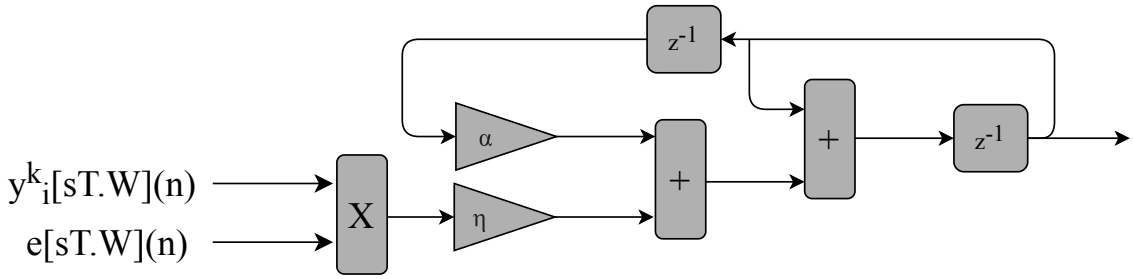

Figure 2.17: Hardware structure for update the weight.

Table 2.1 describes the parameters used in the MLP-BP and RMLP-BP implementation. The choice of parameters for training was carried out empirically.

Table 2.1: Parameters used for implementation MLP-BPand RMLP-BP technique.

Parameter	Value
Number of nodes in layers	4-4-1
Activation function	ReLU
Training Algorithm	Backpropagation
Training mode	Online mode
η	0.008
α	0.0

2.6 Synthesis Results

The synthesis results were obtained for several prediction techniques with linear techniques and non-linear techniques. In the synthesis analysis, the thesis uses the number of Embedded Registers, LUTs, and Multipliers (Digital Signal Processors (DSPs), as well as sample rate (t_s) and throughput $R_s = \frac{1}{t_s}$. All synthesis results use a FPGA Xilinx Virtex 6 (6-bits LUTs) xc6vlx240t-1ff1156. This device has $301,440$ registers, $150,720$ logical cells to be used as LUTs, and 768 multipliers.

Synthesis results are divided into two parts. The results for linear prediction techniques such as zero-order prediction, linear regression method for four values of R (1, 3, 6, and 9). These techniques use a floating-point precision of 32 bits. In the second part are presented the synthesis values for non-linear predictions based. These techniques use Artificial Neural Networks of the MLP type and backpropagation algorithm. This method was implemented for three configurations of signed fixed-point precision such as 18.14. 16.12. and 14.10.

2.6.1 Synthesis Results - Linear Prediction Techniques

Tables 2.2, 2.3 and 2.4 show the synthesis results for linear prediction techniques. Table 2.2 refers to a one implementation of the technique in hardware. Tables 2.3 and 2.4 show the occupation of the target platform area by three and six implementations of the same technique in parallel. Thus, the first column of each table highlights the analyzed prediction method or technique. NR highlights the number of Registers or flip-flops used. The PNR column the percentage of use of registers on the platform. The same is true for an NLUT column that highlights the number of LUTs and for a PNLUT column that displays your percentage of NLUT usage on the platform. The NMULT column shows the number of embedded multipliers used and your percentage of PNMULT usage in the next column. The last two columns refer to the sampling rate, t_s, displayed in nanoseconds, and the throughput, R_s, measured in mega-samples per second (Msps).

Table 2.2: Synthesis results, hardware requirement, sampling rate and throughput results for linear prediction techniques for one single implementation.

Method	NR	PR	NLUT	PNLUT	NMULT	PNMULT	ts (ns)	Rs (Msps)
Zero Order	0.00	0.00	0.00	0.00	0.00	0.00	1.42	704.23
LR (M = 1)	198.00	0.07	3440.00	2.28	9.00	1.17	40.25	24.84
LR (M = 3)	380.00	0.13	5574.00	3.70	9.00	1.17	64.50	15.50
LR (M = 6)	649.00	0.22	8870.00	5.89	9.00	1.17	104.81	9.54
LR (M = 9)	942.00	0.31	11762.00	7.80	9.00	1.17	142.16	7.03

Table 2.3: Synthesis results, hardware requirement, sampling rate and throughput results for linear prediction techniques for 3 techniques implementation in parallel.

Method	NR	PR	NLUT	PNLUT	NMULT	PNMULT	ts (ns)	Rs (Msps)
Zero Order	0.00	0.00	0.00	0.00	0.00	0.00	1.42	704.23
LR (M = 1)	529.00	0.18	9,923.00	6.58	27.00	3.52	43.53	22.97
LR (M = 3)	1,075.00	0.36	16,328.00	10.83	27.00	3.52	66.07	15.14
LR (M = 6)	1,886.00	0.63	26,159.00	17.36	27.00	3.52	118.64	8.43
LR (M = 9)	2,764.00	0.92	34,979.00	23.21	27.00	3.52	139.12	7.19

Table 2.4: Synthesis results, hardware requirement, sampling rate and throughput results for linear prediction techniques for 6 techniques implementation in parallel.

Method	NR	PR	NLUT	PNLUT	NMULT	PNMULT	ts (ns)	Rs (Msps)
Zero Order	0.00	0.00	0.00	0.00	0.00	0.00	1.42	704.23
LR (M = 1)	1,027.00	0.34	19,649.00	13.04	54.00	7.03	42.42	23.58
LR (M = 3)	2,119.00	0.70	32,457.00	21.53	54.00	7.03	66.81	14.97
LR (M = 6)	3,740.00	1.24	52,146.00	34.60	54.00	7.03	104.75	9.55
LR (M = 9)	5,497.00	1.82	69,595.00	46.18	54.00	7.03	171.32	5.84

The surfaces of the number of Registers (NR), the number of LUTs (NLUT), and throughput in function of NI and M for the prediction techniques for linear regression are presented in Figures 2.18,2.19, and 2.20, respectively. For the case of NR, the plane, $f_{\mathrm{NR}}(\mathrm{NI},\mathrm{M})$, was expressed by

$$f_{\mathrm{NR}}(\mathrm{NI},\mathrm{M}) \approx -1,439 + 510.7 \times \mathrm{NI} + 309.5 \times \mathrm{M}, \tag{2.12}$$

with a $R^2 = 0.8553$.

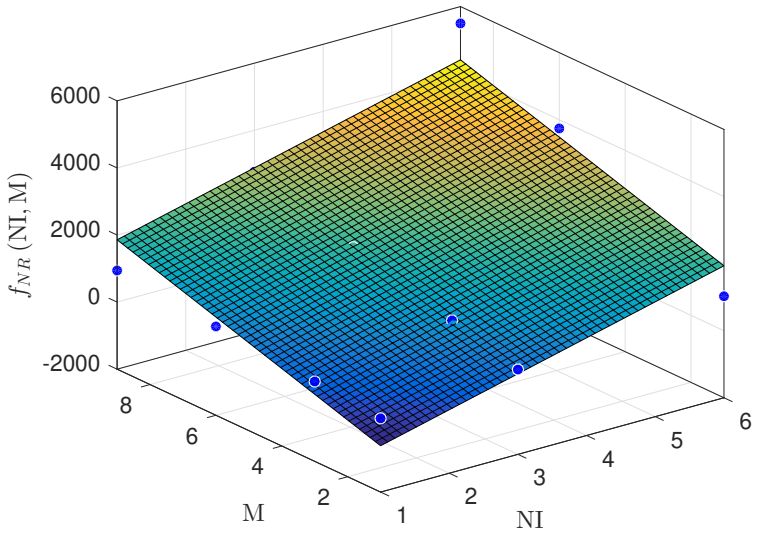

Figure 2.18: Plane, $f_{NR}(NI,M)$, found to estimate the number of Registers as a function of the number of implementations NI and M for prediction techniques Linear Regression based.

For the NLUT, the plane, $f_{\mathrm{NLUT}}(\mathrm{NI},\mathrm{M})$, was expressed by

$$f_{\mathrm{NLUT}}(\mathrm{NI},\mathrm{M}) \approx -16,360 + 7,210 \times \mathrm{NI} + 3,487 \times \mathrm{M}, \tag{2.13}$$

with a $R^2 = 0.8863$.

For throughput in Msps, a plane was found, $f_{R_s}(\mathrm{NI},\mathrm{M})$, characterized as

$$f_{R_s}(\mathrm{NI},\mathrm{M}) \approx 23.99 - 0.1368 \times \mathrm{NI} - 2.067 \times \mathrm{M}, \tag{2.14}$$

with $R^2 = 0.8980$.

The synthesis results presents for linear predictions in Tables 2.2, 2.3 and 2.4 shows that the zero-order prediction technique does not make use of any significant analyzed resources. The platform itself can handle the registration element as a signal delay using only its own clock system. The throughput value for the zero-order technique can explain the response time for input and output ports (*XILINX Virtex-6 User Guide* n.d., *XILINX Virtex-6 Data Sheet* n.d.).

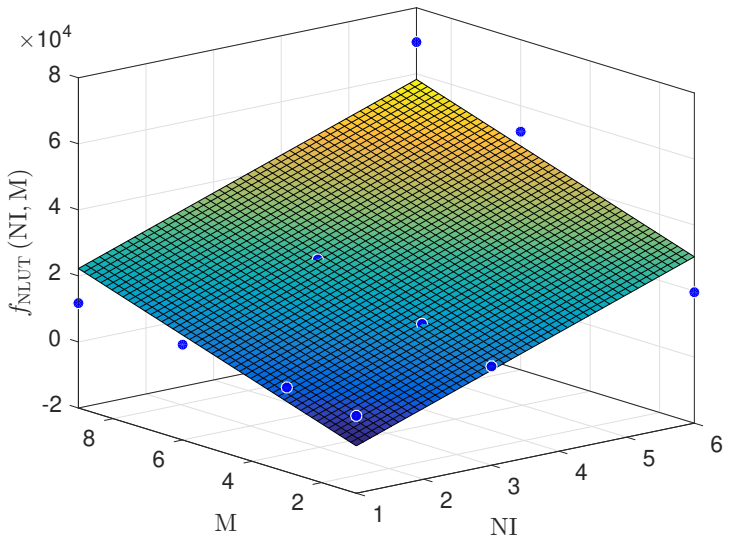

Figure 2.19: Plane, $f_{NLUT}(NI,M)$, found to estimate the number of LUTs as a function of the number of implementations NI and M for prediction techniques Linear Regression based.

The technique based on linear regression shows a linear increase in the consumption of resources as M increases and the number of implementations in the same FPGA. The t_s increases linearly to M and remains constant to the number of implementations. Observing the value of t_s, there is a significant difference in the increase in the number of samples. The main cause of the increase is the increase in the number of cascading elements to perform the method calculations, such as the cascading sum, for example. This increase in the number of elements increases the critical path of the system, thus increasing the value of t_s. Between the value of t_s for the LR (M = 1) and LR (M = 9) configuration there is an increase of 3.53×.

2.6.2 Synthesis Results - Non-Linear Prediction Techniques

The implementations related to nonlinear prediction techniques were also analyzed. The use of MLP-based techniques is common using the hyperbolic tangent function. But, using this function the percent occupation of memory elements for the target platform was of 28%. This value uses a one implementation of the technique with an MLP architecture of 4 inputs, 4 neurons in the hidden layer, 1 neurons in the output layer. With this occupancy rate, it would be unfeasible to proceed with the implementations used by the *tanh* function as a way out of the hidden neurons in the ANN. In a simple calculation the platform occupancy in memory elements would be $PNBRAM \approx 28 \times 6$ or $PNBRAM \approx 168\%$. It is then proposed to use the rectified linear unit activation function call output function or ReLU, describes in the Section 2.4 by Equation 2.8. In this new approach, the use of memories for the implementation of the function is zero.

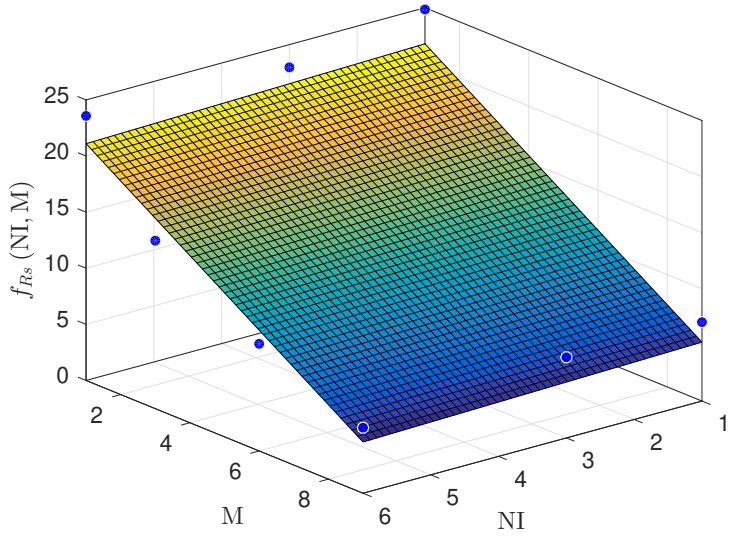

Figure 2.20: Plane, $f_{Rs}(NI,M)$, found to estimate the throughput, R_s, as a function of the number of implementations NI and M for prediction techniques Linear Regression based.

Tables 2.5 and 2.6 show the hardware occupancy results for the target platform for two implementations of non-linear prediction techniques for various configurations with fixed-point notation. The column T.W provides a fixed-point bit configuration used in each implementation. The column *NI* shows the number of implementations of the technique.

Table 2.5: Synthesis results, hardware requirement, sampling rate and throughput results for MLP prediction techniques for 1, 3 ,and 6 techniques implementation.

NI	T.W	NR	PR	NLUT	PNLUT	NMULT	PNMULT	ts (ns)	Rs (Msps)
1	18.14	547.00	0.18	8,141.00	5.40	54.00	7.03	54.24	18.44
	16.12	514.00	0.17	7,307.00	4.85	54.00	7.03	55.12	18.14
	14.10	431.00	0.14	6,575.00	4.36	54.00	7.03	52.94	18.89
3	18.14	1,695.00	0.56	24,483.00	16.24	162.00	21.09	64.42	15.52
	16.12	1,590.00	0.53	21,889.00	14.52	162.00	21.09	60.24	16.60
	14.10	1,335.00	0.44	19,729.00	13.09	162.00	21.09	55.39	18.05
6	18.14	3,390.00	1.12	48,520.00	32.19	324.00	42.19	63.95	15.64
	16.12	3,180.00	1.05	43,390.00	28.79	324.00	42.19	64.03	15.62
	14.10	2,670.00	0.89	39,718.00	26.35	324.00	42.19	61.86	16.17

Table 2.6: Synthesis results, hardware requirement, sampling rate and throughput results for RMLP prediction techniques for 1, 3 ,and 6 techniques implementation.

NI	T.W	NR	PR	NLUT	PNLUT	NMULT	PNMULT	ts (ns)	Rs (Msps)
1	18.14	565.00	0.19	8,141.00	5.40	54.00	7.03	57.01	17.54
	16.12	530.00	0.18	7303.00	4.85	54.00	7.03	55.87	17.90
	14.10	445.00	0.15	6,577.00	4.36	54.00	7.03	54.74	18.27
3	18.14	1749.00	0.58	24455.00	16.23	162.00	21.09	63.66	15.71
	16.12	1,738.00	0.58	21,885.00	14.52	162.00	21.09	56.99	17.55
	14.10	1,377.00	0.46	19,725.00	13.09	162.00	21.09	56.37	17.74
6	18.14	3,498.00	1.16	48,910.00	32.45	324.00	42.19	75.44	13.26
	16.12	3,276.00	1.09	43,786.00	29.05	324.00	42.19	63.95	15.64
	14.10	2,754.00	0.91	39,460.00	26.18	324.00	42.19	60.77	16.46

Based on the results presented on Tables 2.5 and 2.6, as well as the linear predictions results, it is possible to create surfaces to describe the behavior of the hardware occupation based on the number of implementations and the number of bits in fractional part, W, of the value in fixed point precision. The surfaces of the number of Registers (NR), the number of LUTs (NLUT), and throughput in function of NI and W for the non-linear prediction techniques are presented in Figures 2.21,2.23, 2.25, 2.22,2.24, and 2.26.

For the case of NR, the planes, $f_{\mathrm{NR}}(\mathrm{NI},\mathrm{W})$, was expressed by

$$f_{\mathrm{NR}}^{MLP}(\mathrm{NI},\mathrm{W}) \approx -1,439 + 510.7 \times \mathrm{NI} + 309.5 \times \mathrm{W}, \tag{2.15}$$

with a $R^2 = 1$ and

$$f_{\mathrm{NR}}^{RMLP}(\mathrm{NI},\mathrm{W}) \approx -1,237 + 531.4 \times \mathrm{NI} + 103 \times \mathrm{W}, \tag{2.16}$$

with a $R^2 = 0.9835$.

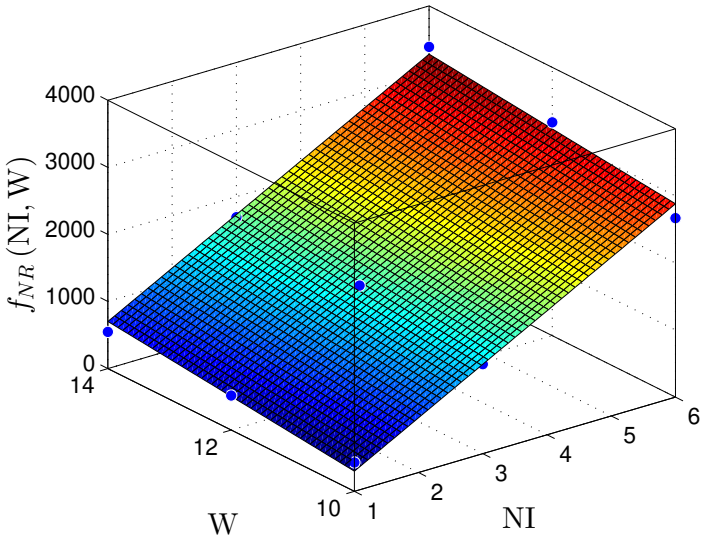

Figure 2.21: Plane, $f_{NR}(NI,W)$, found to estimate the number of Registers, *NR*, as a function of the number of implementations *NI* and the number of bits in fractional part *W* for prediction techniques MLP based.

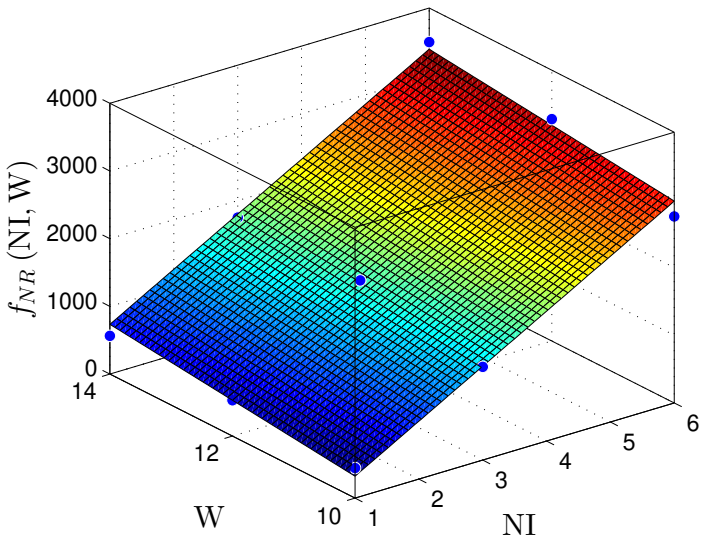

Figure 2.22: Plane, $f_{NR}(NI,W)$, found to estimate the number of Registers, *NR*, as a function of the number of implementations *NI* and the number of bits in fractional part *W* for prediction techniques RMLP based.

For the NLUT, the plane, $f_{\mathrm{NLUT}}(\mathrm{NI},\mathrm{W})$, was expressed by

$$f_{\mathrm{NLUT}}^{MLP}(\mathrm{NI},\mathrm{W}) \approx -15,050 + 7,305 \times \mathrm{NI} + 1,260 \times \mathrm{W}, \tag{2.17}$$

with a $R^2 = 0.9935$ and

$$f_{\mathrm{NLUT}}^{RMLP}(\mathrm{NI},\mathrm{W}) \approx -15,750 + 7,342 \times \mathrm{NI} + 1,312 \times \mathrm{W}, \tag{2.18}$$

with a $R^2 = 0.9899$.

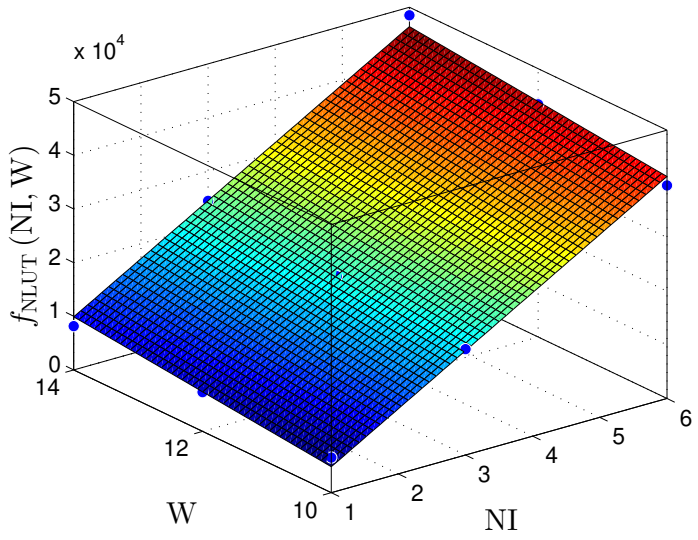

Figure 2.23: Plane, $f_{NLUT}(NI,W)$, found to estimate the number of LUTs, $NLUT$, as a function of the number of implementations NI and the number of bits in fractional part W for prediction techniques MLP based.

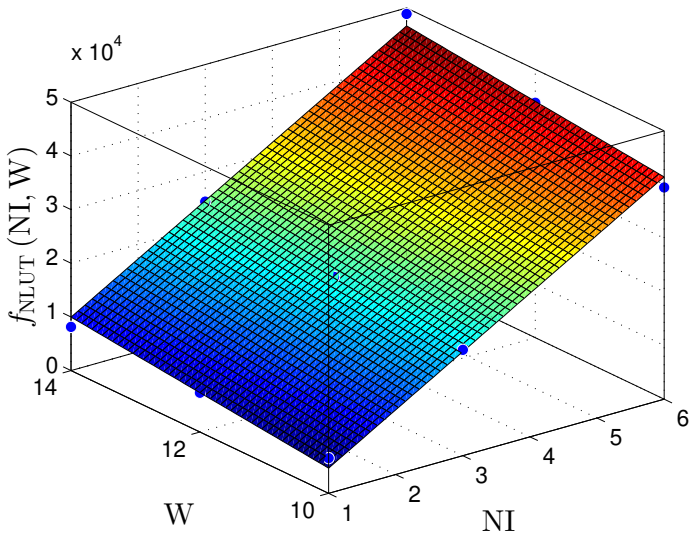

Figure 2.24: Plane, $f_{NLUT}(NI,W)$, found to estimate the number of LUTs, $NLUT$, as a function of the number of implementations NI and the number of bits in fractional part W for prediction techniques RMLP based.

For throughput in Msps, a plane was found, $f_{R_s}(\mathrm{NI},\mathrm{W})$, characterized as

$$f_{R_s}^{MLP}(\mathrm{NI},\mathrm{W}) \approx 22.25 - 0.5183 \times \mathrm{NI} - 0.2926 \times \mathrm{W}, \tag{2.19}$$

with $R^2 = 1$ and

$$f_{R_s}^{RMLP}(\mathrm{NI},\mathrm{W}) \approx 24.51 - 0.5628 \times \mathrm{NI} - 0.4966 \times \mathrm{W}, \tag{2.20}$$

with $R^2 = 8384$.

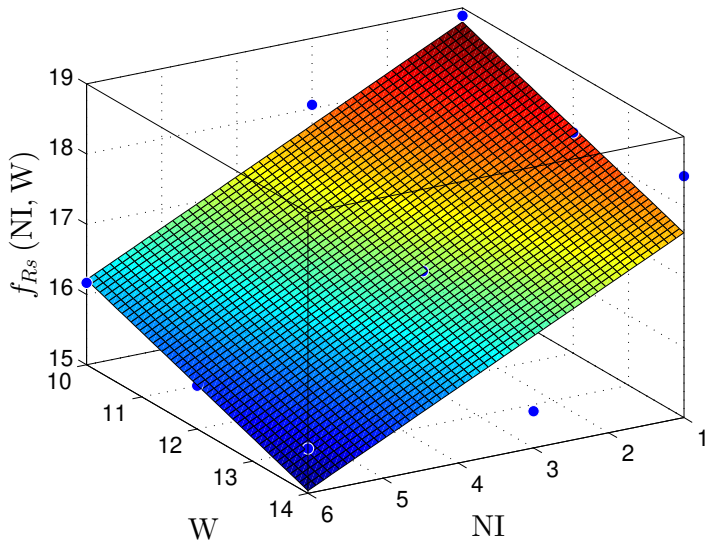

Figure 2.25: Plane, $f_{R_s}(NI,W)$, found to estimate the number of Registers, R_s, as a function of the number of implementations NI and the number of bits in fractional part W for prediction techniques MLP based.

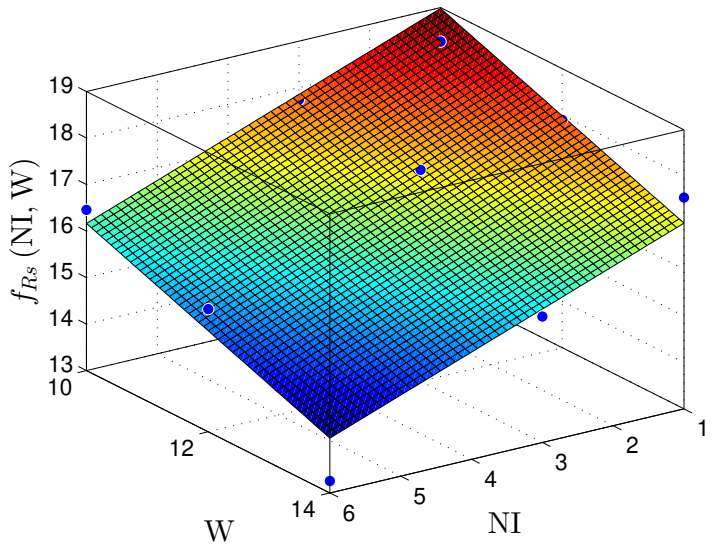

Figure 2.26: Plane, $f_{R_s}(NI,W)$, found to estimate the number of Registers, R_s, as a function of the number of implementations NI and the number of bits in fractional part W for prediction techniques RMLP based.

Tables 2.7 and 2.8 shows the hardware occupancy results for a target platform for the MLP and BP modules, MLPM and BPM, for various bit precision and number of

implementations for MLPM, excepted BPM, the implementation using 3 and 6 in parallel it was not possible. The column *NI* show the number of implementations of the hardware. The column T.W provides a fixed-point bit configuration used in each implementation. These implementations show how much each module contributes to the occupation of the platform and what are their respective throughput when separated.

Table 2.7: Synthesis results, hardware requirement, sampling rate, and throughput results for MLP module for 1, 3, and 6 number implementations.

NI	T.W	NR	PR	NLUT	PNLUT	NMULT	PNMULT	t_s(ns)	R_s(Msps)
1	18.14	0.00	0.00	1,166.00	0.77	25.00	3.26	28.69	34.86
	16.12	0.00	0.00	1,061.00	0.70	25.00	3.26	27.30	36.64
	14.10	0.00	0.00	956.00	0.63	25.00	3.26	28.41	35.20
3	18.14	0.00	0.00	3,510.00	2.33	75.00	9.77	32.65	30.63
	16.12	0.00	0.00	3,195.00	2.12	75.00	9.77	32.20	31.06
	14.10	0.00	0.00	2,880.00	1.91	75.00	9.77	30.72	32.55
6	18.14	0.00	0.00	7,020.00	4.66	150.00	19.53	34.36	29.10
	16.12	0.00	0.00	6,390.00	4.24	150.00	19.53	33.84	29.55
	14.10	0.00	0.00	5,760.00	3.82	150.00	19.53	31.81	31.44

Table 2.8: Synthesis results, hardware requirement, sampling rate, and throughput results for BP module for one implementation.

NI	T.W	NR	PR	NLUT	PNLUT	NMULT	PNMULT	t_s(ns)	R_s(Msps)
1	18.14	475.00	0.16	6,411.00	4.25	29.00	3.78	29.16	34.29
	16.12	425.00	0.14	5,651.00	3.75	29.00	3.78	25.83	38.71
	14.10	350.00	0.12	5,316.00	3.53	29.00	3.78	25.16	39.74

The synthesis results presents for non-linear predictions in Tables 2.5 and 2.6 shows that the resources decreases with the number of bits in fractional part. This is because there is a decrease in the number of bits to represent a value, therefore, the elements that store or process this value need fewer bits to perform the same task. On the other hand, the R_s increases as there is a decrease in the number of bits in the fractional part demonstration variations of $\approx$ 3.4Msps between maximum and minimum values for most cases. This opens the way for the development of solutions with flexibility in choosing the resolution used.

The throughput results, for most cases (linear and non-linear), obtained values between 14Msps and 24Msps. These throughput values, R_s, make it possible to use these solutions in problems with critical requirements, such as tactile internet applications (Simsek et al. 2016*b*, Ruan & Wong 2018, Ruan et al. 2018, Wong et al. 2017, Van Den Berg et al. 2017, Wei et al. 2019).

Synthesis results show that the hardware proposal for linear and non-linear techniques takes a small hardware space. In most of the cases the use was of less than 7%, PNMULT, in multipliers and less than 22% in LUTs, PLUT for linear implementation and $\approx$ 42%,

PNMULT, in multipliers and less than 27% in LUTs, PLUT for non-linear techniques. These results enable the use of several implementations in parallel on FPGA. In this case, enable the acceleration of several applications in massive data environments (Gupta & Hewett 2020). This result also allows the use of other robotic manipulators with more degrees of freedom or the use of other tools (Lavín-Delgado et al. 2020). The low hardware consumption allows the use in small FPGAs of low cost and consumption for applications of IoT and M2M (Jenkal et al. 2019).

By observing the implementations of the MLP neural network (MLPM) and its separate training algorithm (BPM), it is possible to analyze the resource consumption for each module. For MLP it is possible to identify that the resource consumption for 6 implementations in parallel is approximately 19.53%.

2.7 Validation Results

To validate the results of the hardware prediction methods this work used bit precision simulation tests. These simulations use a dynamic non-linear system characterized by a robotic manipulator system called Phantom Omni (Song et al. 2006, Sansanayuth et al. 2012, Silva et al. 2009*a*, Al-Wais et al. 2016). This robotic manipulator has 6-DOF (Degree of freedom), with rotational joints. The manipulator joints has only the first three joint activate, while the last three joints are not activated (Al-Wais et al. 2016). As illustrated in Figure 2.27, the device can be modeled as a 3DOF robotic manipulator with two segments L_1 and L_2. The segments are interconnected by three rotary joints angles θ_1, θ_2, and θ_3. The Phantom Omni has been widely used in literature, as presented in (Song et al. 2006, Sansanayuth et al. 2012, Silva et al. 2009*a*). Simulations used $L_1 = 0.135\,\text{mm}$, $L2 = L1$, $L3 = 0.025\,\text{mm}$, and $L4 = L1 + A$, where $A = 0.035\,\text{mm}$ as described in (Silva et al. 2009*a*).

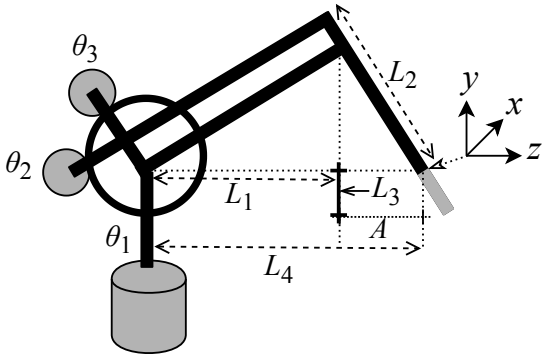

Figure 2.27: Structure of 3DOF Phantom Omni robotic manipulator.

Nonlinear, second order, ordinary differential equation used to describe the dynamics of the Phantom Omni can be expressed as

$$\mathbf{M}(\theta(t))\,\ddot{\theta}(t) + \mathbf{C}\left(\theta(t), \dot{\theta}(t)\right)\dot{\theta}(t) + \mathbf{g}(\theta(t)) - \mathbf{f}\left(\dot{\theta}(t)\right) = \tau(t) \tag{2.21}$$

where $\theta(t)$ is the vector of joints expressed as

$$\theta(t) = \begin{bmatrix} \theta_1(t) & \theta_2(t) & \theta_3(t) \end{bmatrix}^T \in \mathbb{R}^{3\times 1}, \tag{2.22}$$

τ is the vector of torques acting expressed as

$$\tau(t) = \begin{bmatrix} \tau_1(t) & \tau_2(t) & \tau_3(t) \end{bmatrix}^T \in \mathbb{R}^{3\times 1}. \tag{2.23}$$

$\mathbf{M}(\theta(t)) \in \mathbb{R}^{3\times 3}$ is the inertia matrix, $\mathbf{C}\left(\theta(t), \dot{\theta}(t)\right) \in \mathbb{R}^{3\times 3}$ is the Coriolis and centrifugal forces matrix, $\mathbf{g}(\theta(t)) \in \mathbb{R}^{3\times 1}$ represents the gravity force acting on the joints, $\theta(t)$, and the $\mathbf{f}\left(\dot{\theta}(t)\right)$ is the friction force on the joints, $\theta(t)$ (Song et al. 2006, Sansanayuth et al. 2012, Silva et al. 2009*a*, Al-Wais et al. 2016).

Figure 2.28 shows the trajectory using the angular position of the joints θ_1, θ_2, and θ_3. In this simulation the plant is the 3DOF Phantom Omni robotic manipulator. These trajectory are used with the signals input in this work.

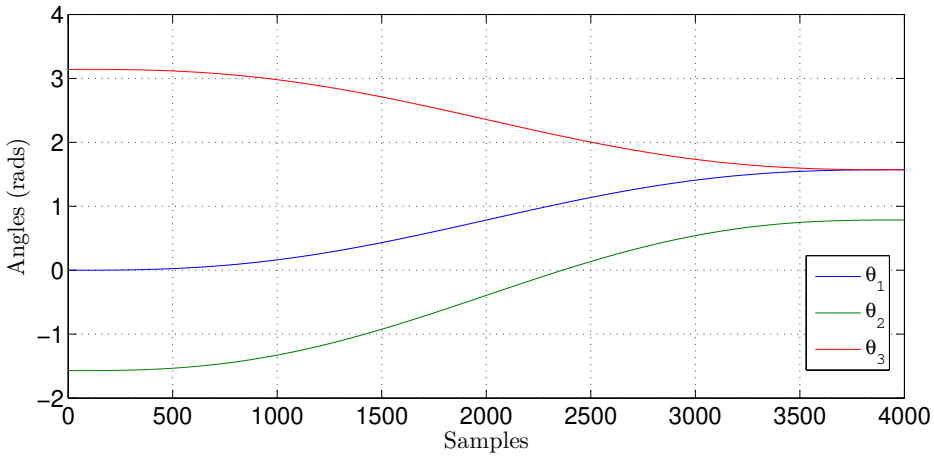

Figure 2.28: Trajectory used in simulations.

The reliability of the results was measured using error measures between the actual data and the predictions made. The analyzes are based on the mean square error. The calculation of the mean square error is defined by

$$e_{qm}(X) = \frac{1}{N_s} \sum_{i=0}^{N_s-1} (x(i) - (\hat{x})(i))^2, \tag{2.24}$$

where $E_qm(X)$ is the value of the mean square error, N_s is the number of samples, $(\hat{X})(i)$ is the estimated value of the i-th sample and $(X)(i)$ is the actual value of the i-th sample.

2.7.1 Validation Results - Linear Prediction Techniques

The validations signals using Matlab Simulink in float-point in Double precision and a proposal implementation linear prediction techniques using float-point in single precision are showed in Figures 2.29–2.33. The validation results use signal variable $\theta_1(n)$.

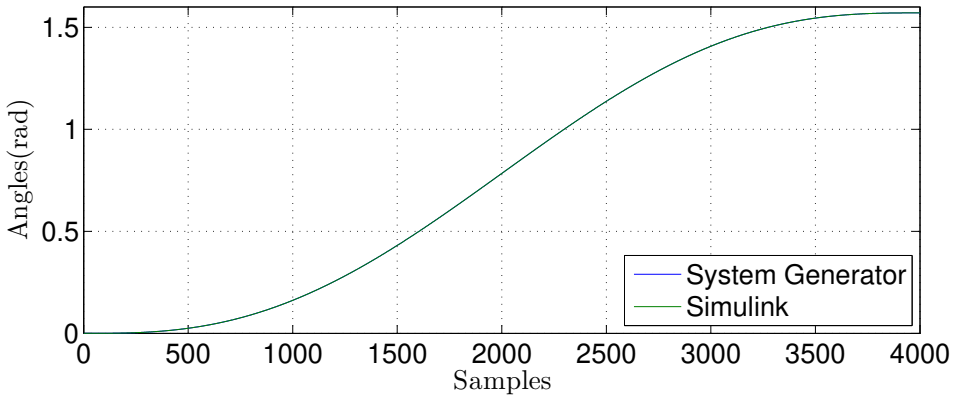

Figure 2.29: Comparison of the results simulations in Matlab Simulink and System Generation for Zero-order technique.

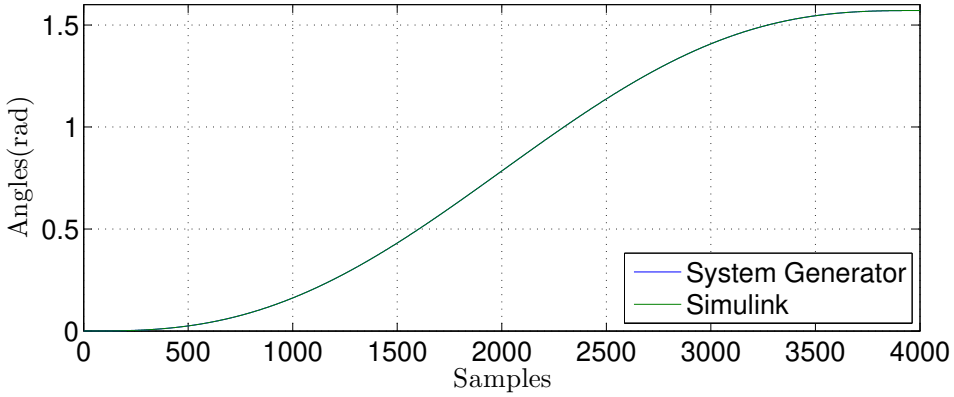

Figure 2.30: Comparison of the results simulations in Matlab Simulink and System Generation for Linear Regression technique with M = 1.

Table 2.9 show the analysis of the absolute MSE relative to the comparisons between the prediction techniques implemented in software, using 64-bit float-point (IEE754) and prediction techniques implemented in 32-bit float-point hardware. It is possible to verify that the two implementations are equivalent, that is, there is no significant difference between the two implementations. The error value is calculated using $N_s = 4,000$ data samples from each implementation.

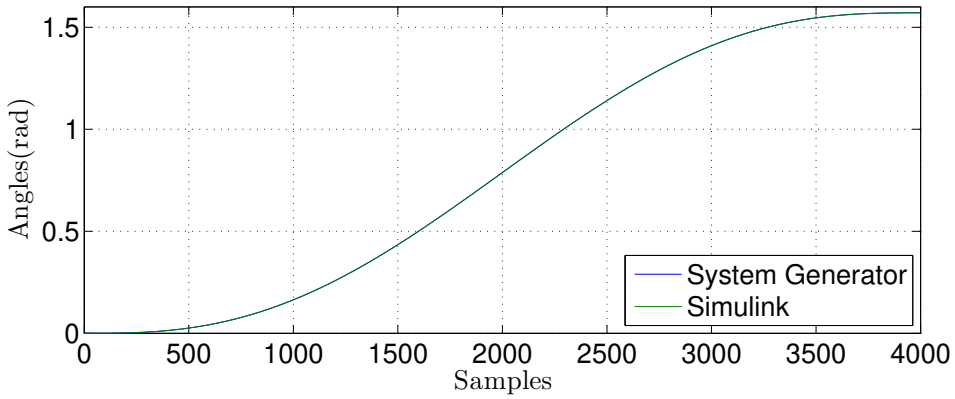

Figure 2.31: Comparison of the results simulations in Matlab Simulink and System Generation for Linear Regression technique with M = 3.

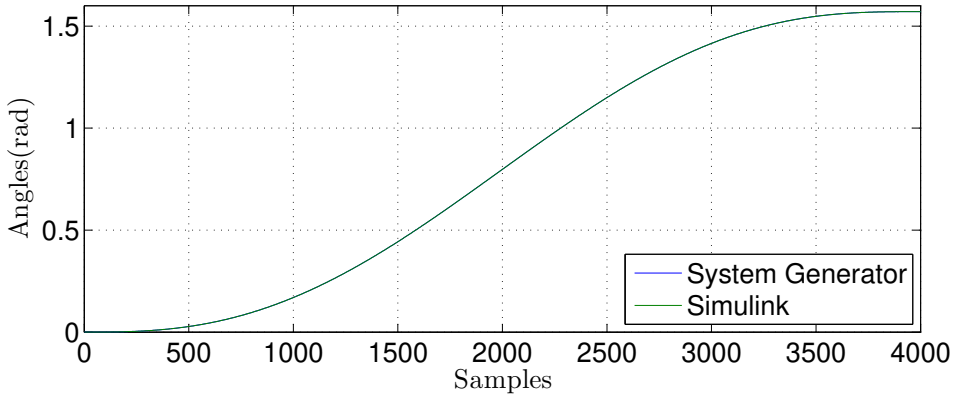

Figure 2.32: Comparison of the results simulations in Matlab Simulink and System Generation for Linear Regression technique with M = 6.

Table 2.9: Mean square error (MSE) between the software implementation and the proposed hardware implementation for linear methods.

Method	MSE
Zero Order	8.67E-13
LR (M = 1)	3.52E-12
LR (M = 3)	4.52E-11
LR (M = 6)	7.22E-10
LR (M = 9)	3.99E-10

Figure 2.34 show the analysis of mean squared error (MSE) over time for comparisons between software implementations, using 64-bit float-point (IEE754), and hardware implementation, using 32-bit float-point (IEE754). For all techniques 80 frames and 50

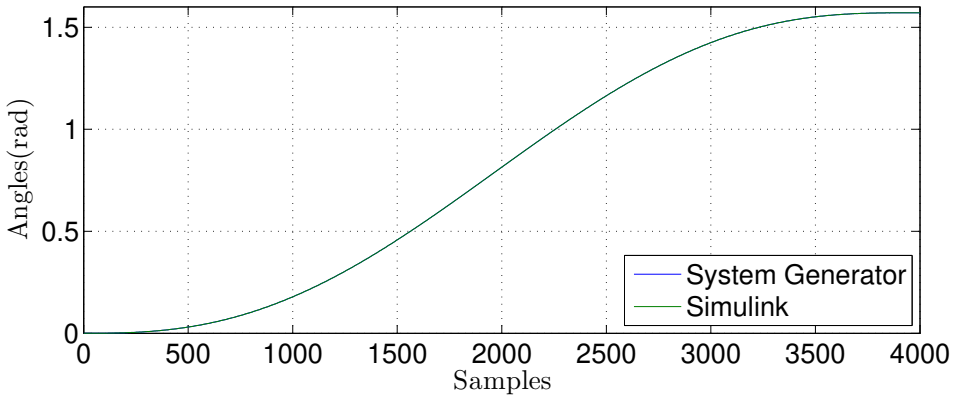

Figure 2.33: Comparison of the results simulations in Matlab Simulink and System Generation for Linear Regression technique with M = 9.

samples per frame were used to generate MSE between the proposed software model and the hardware implementation.

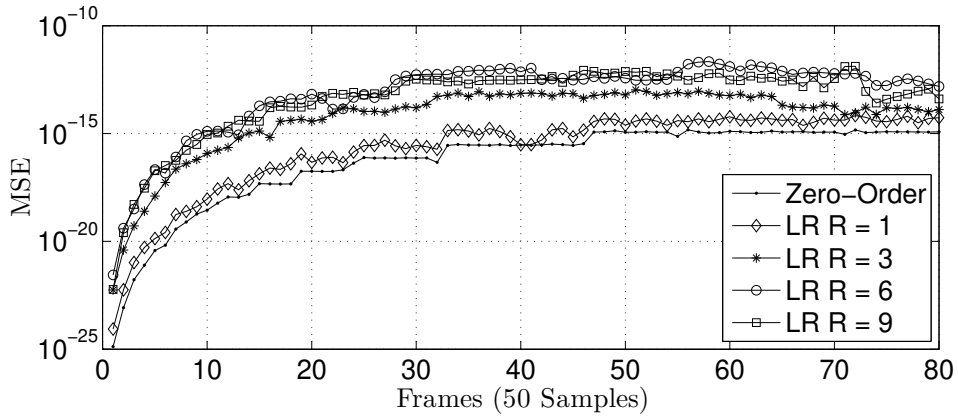

Figure 2.34: Comparison MSE values between Linear Prediction implementations.

2.7.2 Validation Results - Non-Linear Prediction Techniques

The validations signals using Matlab Simulink in float-point in Double precision and a proposal implementation for non-linear prediction techniques using float-point in single precision are showed in the Figures 2.35–2.40. The validation results use only signal variable $\theta_1(n)$ signal. The hardware validation results uses various resolutions in terms of the number of bits of the fractional part, $W = \{10, 12, 14\}$ for two non-linear techniques (MLP-BP, MLP-BP with recurrent inputs).

Figure 2.41 shows the analysis of mean squared error (MSE) over time for comparisons between software implementations of the MLP-BP based on prediction technique, using 64-bit float-point (IEE754) and its representation in hardware for three settings

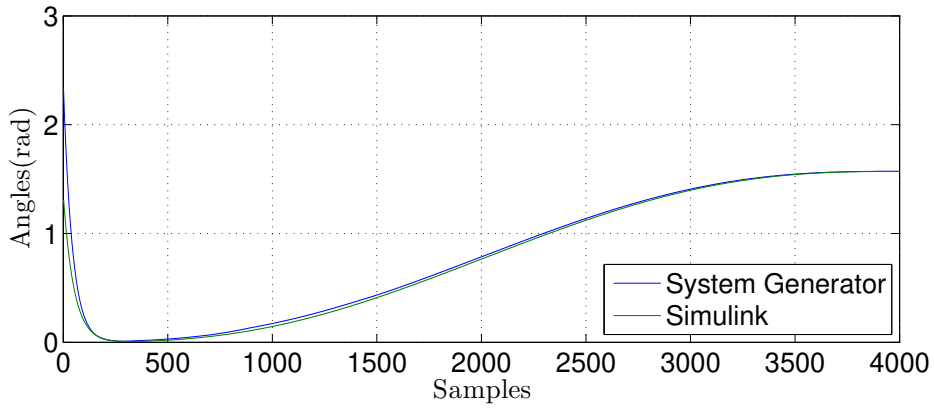

Figure 2.35: Comparison of the results simulations in Matlab Simulink and System Generation for MLP-BP normal implementation using W = 14 fractional part.

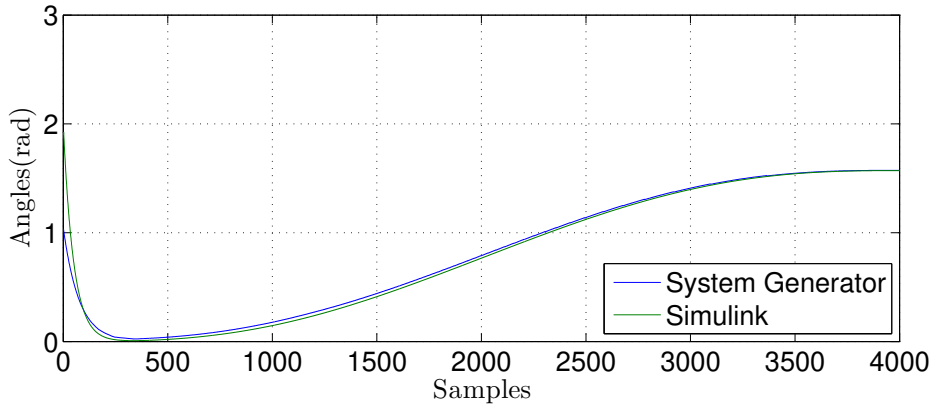

Figure 2.36: Comparison of the results simulations in Matlab Simulink and System Generation for MLP-BP normal implementation using W = 12 fractional part.

fixed-point (18.14, 16.12, 14.10). In all, 80 frames and 50 samples per frame were used to generate MSE between the proposed software model and the hardware implementation.

The results related to *MSE* are also significant. This demonstrates that hardware implementations for prediction techniques have a similar response to the 64 bit implementation even for a 14.10 bit fixed point resolution for MLP. This type demonstrates the possibility of using this type proposed in systems of low cost and limited capacity in hardware resources.

Table 2.9 shows the analysis of the absolute MSE relative to the comparisons between the non-linear prediction techniques implemented in software, using 64-bit float-point (IEE754) and prediction techniques implemented in fixed-point hardware for number of bits of the fractional part, $W = \{10, 12, 14\}$. It is possible to verify that the two implementations are equivalent, that is, there is no significant difference between the two

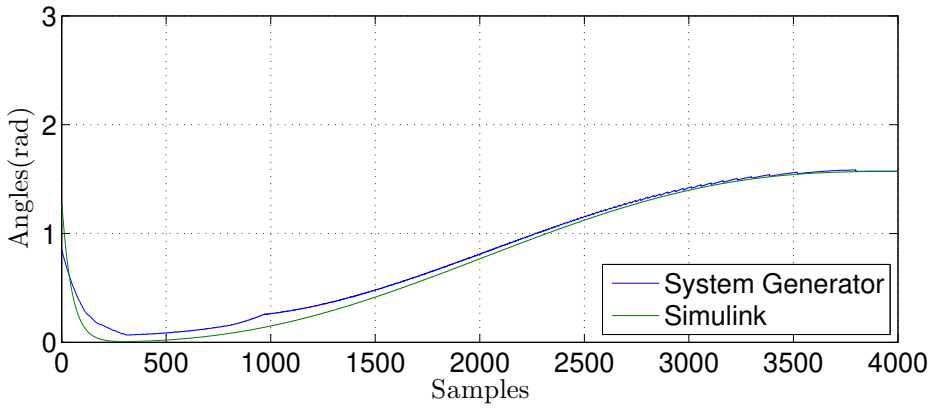

Figure 2.37: Comparison of the results simulations in Matlab Simulink and System Generation for MLP-BP normal implementation using W = 10 fractional part.

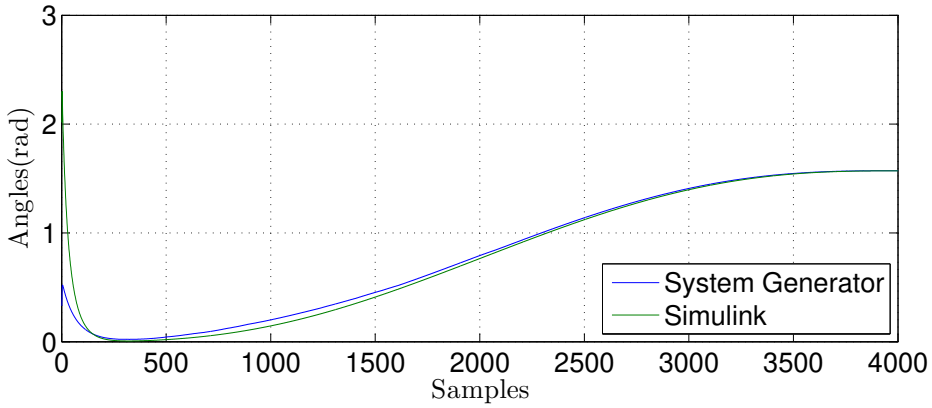

Figure 2.38: Comparison of the results simulations in Matlab Simulink and System Generation for MLP-BP recurrent implementation using W = 14 fractional part.

implementations. The error value is calculated using $N_s = 4,000$ data samples from each implementation.

Table 2.10: Mean square error (MSE) between the software implementation and the proposed hardware implementation for non- linear methods.

Method	Nbp	MSE
MLP	14	5.36E-7
	12	4.97E-6
	10	2.93E-4
RMLP	14	2.33E-8
	12	3.69E-6
	10	5.71E-4

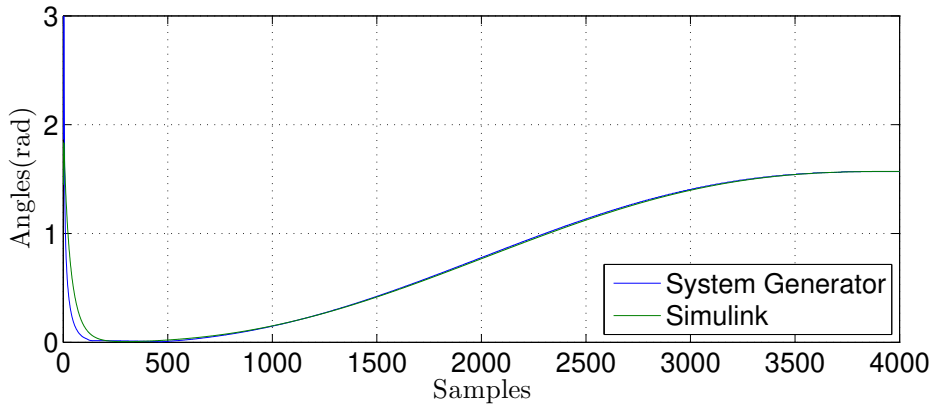

Figure 2.39: Comparison of the results simulations in Matlab Simulink and System Generation for MLP-BP recurrent implementation using W = 12 fractional part.

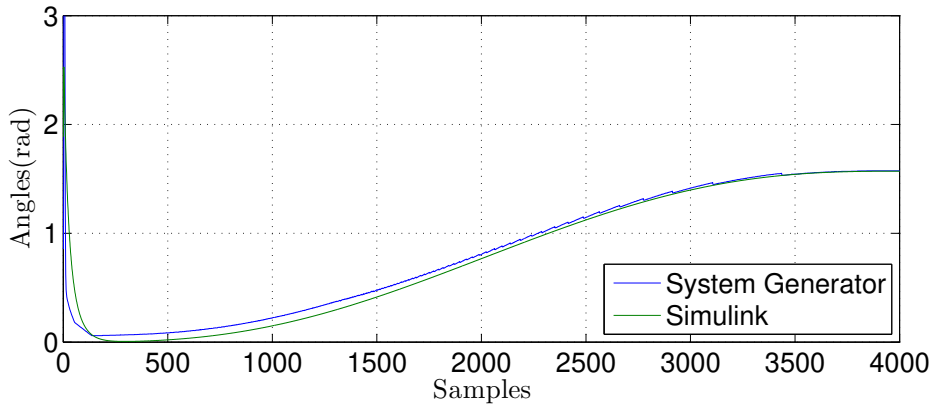

Figure 2.40: Comparison of the results simulations in Matlab Simulink and System Generation for MLP-BP recurrent implementation using W = 10 fractional part.

The MSE values are acceptable for all implementations, linear and non-linear techniques. So any technique can be chosen, the choice will be associated with the precision of the system and its size in the logic elements used.

2.8 Comparison with Other Works

2.8.1 Throughput Comparison

Table 2.11 shows a comparison with other works in the literature. Parameters like Topology, training Mode, number of implementations (NI), Data Precision (T.W), MLP Speed, MLP-BP Speed, and Throughput in Msps are showed.

The authors in (Bahoura 2018) propose an MLP with a topology of 12 inputs, 12 neurons in the hidden layer, and 2 neurons in the output layer. It works with a fixed point

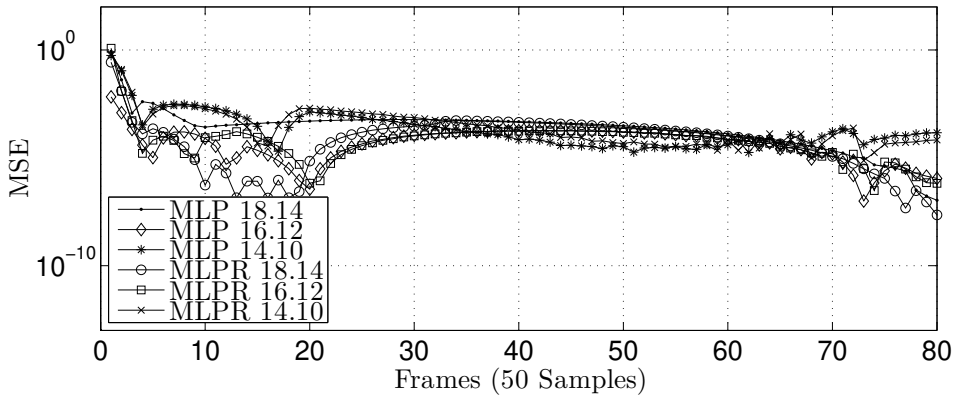

Figure 2.41: Comparison MSE values between MLP-BP implementations.

format data accuracy of 24 bits. The training of the MLP method is offline. The maximum throughput of the proposal is 113.135Msps for implementation in a FPGA Virtex 6 XC6VLX240T and 115.875Msps for implementation in a FPGA Artix-7 XC7A100T. The speedup in Msps for the MLP-BP and RMLP-BP and the MLP and RMLP proposals implemented separately are detailed in Tables 2.12 and 2.13. The high-performance factor shown in (Bahoura 2018) is related to the pipeline used in the proposed hardware design. In this approach, the critical path of the system is reduced and, consequently, gains in the maximum frequency. However, as the proposal presented in this thesis uses online training, the use of a pipeline-based project would not be feasible because the chains of delays for building this approach would hinder the accuracy of the samples for online training.

The work presented in (Gaikwad et al. 2019) proposes an MLP with 7 inputs, 6 neurons in the hidden layer, and 5 neurons in the output layer. The training of the MLP method is offline. The maximum throughput of the proposal is 3.7Msps. The reduced Throughput value is related to the number of clock cycles required to obtain a valid output. The speedup in Msps for the MLP-BP and RMLP-BP and the MLP and RMLP proposals implemented separately are detailed in Tables 2.12 and 2.13. In this case, the average speedup about the proposals presented here was $\approx 4,55$Msps.

In the work presented in (Zhai et al. 2016) propose an MLP with 12 inputs, 3 neurons in the hidden layer, and one neuron in the output layer. The training of the MLP method is offline. The maximum throughput of the proposal is 1.85Msps. The reduced Throughput value may be related to the use of High-Level Syntheses HLS. The speedup in Msps for the MLP-BP and RMLP-BP and the MLP and RMLP proposals implemented separately are detailed in Tables 2.12 and 2.13. In this case, the average speedup about the proposals presented here was ≈ 9.1Msps.

The authors in (Bahoura 2016) propose an MLP with a topology of 12 inputs, 7 neurons in the hidden layer, and 3 neurons in the output layer. It works with a fixed point format data accuracy of 24. The training of the MLP method is offline. The maximum throughput of the proposal is 27.89Msps for boosting an FPGA Virtex 6 XC6VLX240T and 25.24Msps for boosting an FPGA Artix-7 XC7A100T. The speedup in Msps for the

MLP-BP and RMLP-BP and the MLP and RMLP proposals implemented separately are detailed in the Tables 2.12 and 2.13.

Table 2.11: Throughput comparison with other works.

References	FPGA	Topology	Training Mode	NI	Data precision (T.W)	MLP Speed (MHz)	MLP-BP Speed (MHz)	Throughput (Msps)
(Bahoura 2018)	Virtex 6 XC6VLX240T	12-12-2	Off-line	1	24	113.14	-	113.135
	Artix-7 XC7A100T					115.88	-	115.875
(Gaikwad et al. 2019)	Artix-7 35T	7-6-5	Off-line	1	16.15	100.00	-	3.70
(Zhai et al. 2016)	Zynq-7000	12-3-1	Off-line	1	24.20	100.00	-	1.85
(Bahoura 2016)	Virtex 6 XC6VLX240T	12-7-3	Off-line	1	24	27.89	-	27.89
	Artix-7 XC7A100T					25.24	-	25.24
This Work MLP	Virtex 6 XC6VLX240T	4-4-1	On-line	3	18.14	34.86	18.44	18.44
					16.12	36.64	18.14	18.14
					14.10	35.20	18.89	18.89
				6	18.14	30.63	15.52	15.52
					16.12	31.06	16.60	16.60
					14.10	32.55	18.05	18.05
				1	18.14	29.10	15.64	15.64
					16.12	29.55	15.62	15.62
					14.10	31.44	16.17	16.17
This Work RMLP	Virtex 6 XC6VLX240T	4-4-1	On-line	3	18.14	34.86	17.54	17.54
					16.12	36.64	17.90	17.90
					14.10	35.20	18.27	18.27
				6	18.14	30.63	15.71	15.71
					16.12	31.06	17.55	17.55
					14.10	32.55	17.74	17.74
				1	18.14	29.10	13.26	13.26
					16.12	29.55	15.64	15.64
					14.10	31.44	16.46	16.46

Table 2.12: Speedup comparison with other works for implementations associated with backpropagation algorithm.

Prediction Techniques	FPGA	NI	Data precision (T:W)	Related Works SpeedUp (Msps) (Bahoura 2018) 1	(Bahoura 2018) 2	(Gaikwad et al. 2019)	(Zhai et al. 2016)	(Bahoura 2016) 1	(Bahoura 2016) 2
MLP-BP	Virtex 6 XC6VLX240T	1	18.14	0.16	0.16	4.98	9.96	0.66	0.73
			16.12	0.16	0.16	4.90	9.80	0.65	0.72
			14.10	0.17	0.16	5.10	10.20	0.68	0.75
		3	18.14	0.14	0.13	4.19	8.38	0.56	0.62
			16.12	0.15	0.14	4.48	8.96	0.60	0.66
			14.10	0.16	0.16	4.87	9.75	0.65	0.72
		6	18.14	0.14	0.13	4.22	8.44	0.56	0.62
			16.12	0.14	0.13	4.22	8.43	0.56	0.62
			14.10	0.14	0.14	4.36	8.73	0.58	0.64
RMLP-BP	Virtex 6 XC6VLX240T	1	18.14	0.16	0.15	4.74	9.47	0.63	0.70
			16.12	0.16	0.15	4.83	9.67	0.64	0.71
			14.10	0.16	0.16	4.93	9.87	0.66	0.72
		3	18.14	0.14	0.14	4.24	8.48	0.56	0.62
			16.12	0.16	0.15	4.74	9.48	0.63	0.70
			14.10	0.16	0.15	4.79	9.58	0.64	0.70
		6	18.14	0.12	0.11	3.58	7.16	0.48	0.53
			16.12	0.14	0.13	4.22	8.44	0.56	0.62
			14.10	0.15	0.14	4.44	8.89	0.59	0.65

Table 2.13: Speedup comparison with other works for implementations MLPM and RMLPM.

Prediction Techniques	FPGA	NI	Data precision (T.W)	Related Works				
				(Bahoura 2018) 1	(Bahoura 2018) 2	(Gaikwad et al. 2019)	(Zhai et al. 2016)	(Bahoura 2016) 1	(Bahoura 2016) 2
				SpeedUp (Msps)					
MLP	Virtex 6 XC6VLX240T	1	18.14	0.31	0.30	9.41	18.82	1.25	1.38
			16.12	0.32	0.32	9.89	19.78	1.31	1.45
			14.10	0.31	0.30	9.51	19.01	1.26	1.39
		3	18.14	0.27	0.26	8.27	16.54	1.10	1.21
			16.12	0.27	0.27	8.39	16.77	1.11	1.23
			14.10	0.29	0.28	8.79	17.58	1.17	1.29
		6	18.14	0.26	0.25	7.86	15.72	1.04	1.15
			16.12	0.26	0.26	7.98	15.96	1.06	1.17
			14.10	0.28	0.27	8.49	16.98	1.13	1.25
RMLP	Virtex 6 XC6VLX240T	1	18.14	0.31	0.30	9.41	18.82	1.25	1.38
			16.12	0.32	0.32	9.89	19.78	1.31	1.45
			14.10	0.31	0.30	9.51	19.01	1.26	1.39
		3	18.14	0.27	0.26	8.27	16.54	1.10	1.21
			16.12	0.27	0.27	8.39	16.77	1.11	1.23
			14.10	0.29	0.28	8.79	17.58	1.17	1.29
		6	18.14	0.26	0.25	7.86	15.72	1.04	1.15
			16.12	0.26	0.26	7.98	15.96	1.06	1.17
			14.10	0.28	0.27	8.49	16.98	1.13	1.25

The differences between FPGA devices have no significant influence on Throughput. The FPGAs have dedicated wires (called carry chains) between neighboring LUTs, and these circuits have a fast baud rate that allows you to combine multiple LUTs (Teubner & Woods 2013, Cui et al. 2017). Therefore, differences in the size of LUTs do not significantly affect performance.

2.8.2 Hardware Occupation Comparison

Table 2.14 shows a comparison with other works in the literature. Parameters like Topology, training Mode, number of implementations (NI), Data Precision (T.W) are described in the second to fifth columns. The sixth, seventh, eighth, ninth, and tenth columns show the number of LUTs (NLUT), the number of Registers (NR), number of multipliers (NMULT), and the number of block RAMs (NBRAM), respectively. The ratio of the hardware occupation between the proposal presented here, $N^{\text{work}}_{\text{hardware}}$, and literature works, $N^{\text{ref}}_{\text{hardware}}$. The ratio of the hardware occupation can be expressed as

$$R_{\text{occupation}} = \begin{cases} \frac{N^{\text{work}}_{\text{hardware}}}{N^{\text{ref}}_{\text{hardware}}} & \text{, for } N^{\text{work}}_{\text{hardware}} > 0 \text{ and } N^{\text{ref}}_{\text{hardware}} > 0 \\ \frac{1}{N^{\text{ref}}_{\text{hardware}}} & \text{, for } N^{\text{work}}_{\text{hardware}} = 0 \text{ and } N^{\text{ref}}_{\text{hardware}} > 0 \\ N^{\text{work}}_{\text{hardware}} & \text{, for } N^{\text{work}}_{\text{hardware}} > 0 \text{ and } N^{\text{ref}}_{\text{hardware}} = 0 \\ 1 & \text{, for } N^{\text{work}}_{\text{hardware}} = 0 \text{ and } N^{\text{ref}}_{\text{hardware}} = 0 \end{cases}, \tag{2.25}$$

where $N^{\text{work}}_{\text{hardware}}$ and $N^{\text{ref}}_{\text{hardware}}$ can be replaced by NULT, NR, NMULT, or NBRAM.

Table 2.14: Hardware occupation comparison with other works.

References	FPGA	Topology	Training Mode	NI	Data precision (T.W)	NLUT	NR	NMULT	NBRAM
1	Virtex 6 XC6VLX240T	12-12-2	Off-line	1	24	19567	21861	168	26
	Artix-7 XC7A100T					19,732	21,659	168	26
2	Artix-7 35T	7-6-5	Off-line	1	16.15	3,466	569	81	0
3	Zynq-7000	12-3-1	Off-line	1	24.20	4,032	2,863	28	2
4	Virtex 6 XC6VLX240T	12-7-3	Off-line	1	24	21,322	13,546	219	2
	Artix-7 XC7A100T					21,658	13,330	219	2
This Work MLP	Virtex 6 XC6VLX240T	4-4-1	On-line	1	18.14	8,141	547	54	0
					16.12	7,307	514	54	0
					14.10	6,575	431	54	0
				3	18.14	24,483	1,695	162	0
					16.12	21,889	1,590	162	0
					14.10	19,729	1,335	162	0
				6	18.14	48,520	3,390	324	0
					16.12	43,390	3,180	324	0
					14.10	39,718	2,670	324	0
This Work RMLP	Virtex 6 XC6VLX240T	4-4-1	On-line	1	18.14	8,141	565	54	0
					16.12	7,303	530	54	0
					14.10	6,577	445	54	0
				3	18.14	24,455	1,749	162	0
					16.12	21,885	1,738	162	0
					14.10	19,725	1,377	162	0
				6	18.14	48,310	3,498	324	0
					16.12	43,786	3,276	324	0
					14.10	39,460	2,754	324	0

The work presented in (Bahoura 2018) used a Virtex 6 XC6VLX240T e FPGA Artix-7 XC7A100T from Xilinx, and it has a hardware occupation of about 19,567 LUTs, 21,861 Registers, 168 multipliers, and 26 block RAM for Virtex 6 implementation and 19,732 LUTs, 2,1659 Registers, 168 multipliers, and 26 block RAM. The memory usage is related to the construction of the sigmoid activation function. The present work uses the ReLU function and does not require representation in memory.

In (Gaikwad et al. 2019), the work used an Artix-7 35T FPGA from Xilinx and it has a hardware occupation of about 3,466 LUTs, 569 Registers 81 Multipliers, and zero block RAM. The hardware occupation presented in (Zhai et al. 2016) used 4,032 LUTs, 2,863 Registers 28 Multipliers, and two block RAM.

The hardware structure in (Bahoura 2016) used a Virtex 6 XC6VLX240T e FPGA Artix-7 XC7A100T from Xilinx, and it has a hardware occupation of about 21,322 LUTs, 13,546 Registers, 219 multipliers, and two block RAM for Virtex 6 implementation and 21,658 LUTs, 13,330 Registers, 219 multipliers, and two block RAM. This paper uses the sigmoid with activation function.

This work implemented to various models, in this case the Tables 2.15 - 2.18 shows the proportion of occupation of this work about other works in the literature.

Table 2.15: Analysis of the Ratio Occupation for NLUT.

References	NI	Related Works					
		(Bahoura 2018) 1	(Bahoura 2018) 2	(Gaikwad et al. 2019)	(Zhai et al. 2016)	(Bahoura 2016) 1	(Bahoura 2016) 2
		Ratio Occupation					
This Work MLP-BP	1	0.42	0.41	2.35	2.02	0.38	0.38
		0.37	0.37	2.11	1.81	0.34	0.34
		0.34	0.33	1.90	1.63	0.31	0.30
	3	1.25	1.24	7.06	6.07	1.15	1.13
		1.12	1.11	6.32	5.43	1.03	1.01
		1.01	1.00	5.69	4.89	0.93	0.91
	6	2.48	2.46	14.00	12.03	2.28	2.24
		2.22	2.20	12.52	10.76	2.03	2.00
		2.03	2.01	11.46	9.85	1.86	1.83
This Work RMLP-BP	1	0.42	0.41	2.35	2.02	0.38	0.38
		0.37	0.37	2.11	1.81	0.34	0.34
		0.34	0.33	1.90	1.63	0.31	0.30
	3	1.25	1.24	7.06	6.07	1.15	1.13
		1.12	1.11	6.31	5.43	1.03	1.01
		1.01	1.00	5.69	4.89	0.93	0.91
	6	2.47	2.45	13.94	11.98	2.27	2.23
		2.24	2.22	12.63	10.86	2.05	2.02
		2.02	2.00	11.38	9.79	1.85	1.82

Table 2.16: Analysis of the Ratio Occupation for NR.

References	NI	Related Works					
		(Bahoura 2018) 1	(Bahoura 2018) 2	(Gaikwad et al. 2019)	(Zhai et al. 2016)	(Bahoura 2016) 1	(Bahoura 2016) 2
		Ratio Occupation					
This Work MLP-BP	1	0.03	0.03	0.96	0.19	0.04	0.04
		0.02	0.02	0.90	0.18	0.04	0.04
		0.02	0.02	0.76	0.15	0.03	0.03
	3	0.08	0.08	2.98	0.59	0.13	0.13
		0.07	0.07	2.79	0.56	0.12	0.12
		0.06	0.06	2.35	0.47	0.10	0.10
	6	0.16	0.16	5.96	1.18	0.25	0.25
		0.15	0.15	5.59	1.11	0.23	0.24
		0.12	0.12	4.69	0.93	0.20	0.20
This Work RMLP-BP	1	0.03	0.03	0.99	0.20	0.04	0.04
		0.02	0.02	0.93	0.19	0.04	0.04
		0.02	0.02	0.78	0.16	0.03	0.03
	3	0.08	0.08	3.07	0.61	0.13	0.13
		0.08	0.08	3.05	0.61	0.13	0.13
		0.06	0.06	2.42	0.48	0.10	0.10
	6	0.16	0.16	6.15	1.22	0.26	0.26
		0.15	0.15	5.76	1.14	0.24	0.25
		0.13	0.13	4.84	0.96	0.20	0.21

Table 2.17: Analysis of the Ratio Occupation for NMULT.

References	NI	Related Works					
		(Bahoura 2018) 1	(Bahoura 2018) 2	(Gaikwad et al. 2019)	(Zhai et al. 2016)	(Bahoura 2016) 1	(Bahoura 2016) 2
		Ratio Occupation					
This Work MLP-BP	1	0.32	0.32	0.67	1.93	0.25	0.25
		0.32	0.32	0.67	1.93	0.25	0.25
		0.32	0.32	0.67	1.93	0.25	0.25
	3	0.96	0.96	2.00	5.79	0.74	0.74
		0.96	0.96	2.00	5.79	0.74	0.74
		0.96	0.96	2.00	5.79	0.74	0.74
	6	1.93	1.93	4.00	11.57	1.48	1.48
		1.93	1.93	4.00	11.57	1.48	1.48
		1.93	1.93	4.00	11.57	1.48	1.48
This Work RMLP-BP	1	0.32	0.32	0.67	1.93	0.25	0.25
		0.32	0.32	0.67	1.93	0.25	0.25
		0.32	0.32	0.67	1.93	0.25	0.25
	3	0.96	0.96	2.00	5.79	0.74	0.74
		0.96	0.96	2.00	5.79	0.74	0.74
		0.96	0.96	2.00	5.79	0.74	0.74
	6	1.93	1.93	4.00	11.57	1.48	1.48
		1.93	1.93	4.00	11.57	1.48	1.48
		1.93	1.93	4.00	11.57	1.48	1.48

Table 2.18: Analysis of the Ratio Occupation for NBRAM.

References	NI	Related Works					
		(Bahoura 2018) 1	(Bahoura 2018) 2	(Gaikwad et al. 2019)	(Zhai et al. 2016)	(Bahoura 2016) 1	(Bahoura 2016) 2
		Ratio Occupation					
This Work MLP-BP	1	0.04	0.04	1.00	0.50	0.50	0.50
		0.04	0.04	1.00	0.50	0.50	0.50
		0.04	0.04	1.00	0.50	0.50	0.50
	3	0.04	0.04	1.00	0.50	0.50	0.50
		0.04	0.04	1.00	0.50	0.50	0.50
		0.04	0.04	1.00	0.50	0.50	0.50
	6	0.04	0.04	1.00	0.50	0.50	0.50
		0.04	0.04	1.00	0.50	0.50	0.50
		0.04	0.04	1.00	0.50	0.50	0.50
This Work RMLP-BP	1	0.04	0.04	1.00	0.50	0.50	0.50
		0.04	0.04	1.00	0.50	0.50	0.50
		0.04	0.04	1.00	0.50	0.50	0.50
	3	0.04	0.04	1.00	0.50	0.50	0.50
		0.04	0.04	1.00	0.50	0.50	0.50
		0.04	0.04	1.00	0.50	0.50	0.50
	6	0.04	0.04	1.00	0.50	0.50	0.50
		0.04	0.04	1.00	0.50	0.50	0.50
		0.04	0.04	1.00	0.50	0.50	0.50

2.8.3 Power Consumption Comparison

Tables 2.19 and 2.20 shows the dynamic power saving regarding the dynamic power. The dynamic power can be expressed as

$$P_d \propto N_g \times F_{\text{clk}} \times V_{DD}^2, \tag{2.26}$$

where N_g is the number of elements (or gates), F_{clk} is the maximum clock frequency, and V_{DD} is the supply voltage. The frequency dependence is more severe than Equation 2.26 suggests, given that the frequency at which a CMOS circuit can operate is approximately proportional to the voltage (McCool et al. 2012). Thus, the dynamic power can be expressed as

$$P_d \propto N_g \times F_{\text{clk}}^3. \tag{2.27}$$

For all comparisons, the number of elements, N_g, was calculated as

$$N_g = \text{NLUT} + \text{NR} + \text{NMULT}. \tag{2.28}$$

Based on Equation 2.27, the dynamic power saving can be expressed as

$$S_d = \frac{N_g^{\text{ref}} \times \left(F_{\text{clk}}^{\text{ref}}\right)^3}{N_g^{\text{work}} \times \left(F_{\text{clk}}^{\text{work}}\right)^3}, \tag{2.29}$$

where the N_g^{ref} and $F_{\text{clk}}^{\text{ref}}$ are the number of elements (NLUT + NR + NMULT) and the maximum clock frequency of the literature works, respectively, and the N_g^{work} and $F_{\text{clk}}^{\text{work}}$ are the number of elements (NLUT + NR + NMULT) and the maximum clock frequency of this work, respectively. Differently from the literature, the hardware proposed here uses a fully parallelization layout, and it spends a one clock cycle per sample processing. In other words, the maximum clock frequency is equivalent to the throughput, $F_{\text{clk}}^{\text{work}} \equiv R_s$.

Table 2.19: Analysis of the Frequency and Ng.

References	NI	Data precision (T.W)	F_{clk}	N_g
(Bahoura 2018)	1	24	113.14	41,596
			115.88	41,559
(Gaikwad et al. 2019)	1	16.15	100.00	4,116
(Zhai et al. 2016)	1	24.20	100.00	6,923
(Bahoura 2016)	1	24	27.89	35,087
			25.24	35,207
This Work MLP	1	18.14	18.44	8,742
		16.12	18.14	7,875
		14.10	18.89	7,060
	3	18.14	15.52	26,340
		16.12	16.60	23,641
		14.10	18.05	21,226
	6	18.14	15.64	52,234
		16.12	15.62	46,894
		14.10	16.17	42,712
This Work RMLP	1	18.14	17.54	8,760
		16.12	17.90	7,887
		14.10	18.27	7,076
	3	18.14	15.71	26,366
		16.12	17.55	23,785
		14.10	17.74	21,264
	6	18.14	13.26	52,132
		16.12	15.64	47,386
		14.10	16.46	42,538

Table 2.20: Analysis of the Dynamic Power.

		Related Works					
		(Bahoura 2018) 1	(Bahoura 2018) 2	(Gaikwad et al. 2019)	(Zhai et al. 2016)	(Bahoura 2016) 1	(Bahoura 2016) 2
		Dynamic Power					
This Work MLP-BP	1	1,099.43×	560.36×	75.13×	126.36×	13.89×	10.33×
		1,280.98×	652.89×	87.53×	147.23×	16.19×	12.03×
		1,266.09×	645.30×	86.52×	145.52×	16.00×	11.89×
	3	611.41×	311.62×	41.78×	70.27×	7.73×	5.74×
		556.89×	283.83×	38.05×	64.01×	7.04×	5.23×
		482.30×	245.82×	32.96×	55.43×	6.09×	4.53×
	6	301.57×	153.71×	20.61×	34.66×	3.81×	2.83×
		337.13×	171.83×	23.04×	38.75×	4.26×	3.17×
		333.84×	170.15×	22.81×	38.37×	4.22×	3.14×
This Work RMLP-BP	1	1,274.00×	649.33×	87.06×	146.43×	16.10×	11.97×
		1,331.81×	678.80×	91.01×	153.07×	16.83×	12.51×
		1,396.04×	711.53×	95.40×	160.45×	17.64×	13.12×
	3	589.36×	300.38×	40.27×	67.74×	7.45×	5.54×
		468.67×	238.87×	32.03×	53.87×	5.92×	4.40×
		507.39×	258.61×	34.67×	58.32×	6.41×	4.77×
	6	496.09×	252.85×	33.90×	57.02×	6.27×	4.66×
		332.46×	169.46×	22.72×	38.21×	4.20×	3.12×
		317.77×	161.96×	21.71×	36.52×	4.02×	2.99×

2.8.4 Analysis of the Comparisons

The results presented in Tables 2.11, 2.12, 2.13, **??** and **??** demonstrate that the fully parallelization strategy adopted for non-linear prediction techniques performed well with respect to speedup and a significant reduction in power consumption. In this last case, the consumption is reduced in $\approx$ 1,400 $\times$. On the other hand, the fully parallelization scheme can increase the hardware consumption, see Tables 2.14–2.18.

The mean value of speedup in Tables 2.12 and 2.13 was about $\approx 2.5\times$ in Msps for comparison using MLP and RMLP associated with BP algorithm and $\approx 4.9\times$ in for comparison using MLP and RMLP without BP module. The speedup maximum was from $19.78\times$

The hardware consumption in this thesis in comparison with other works use approximately twice more resources. But it does not use any RAM block. The efficient use of these elements can increase Throughput and reduce power consumption.

Regarding power consumption, the present work has a good performance even for 6 implementations in parallel. This result paves the way for work involving low-power devices.

2.9 Conclusion

This chapter presented several hardware proposals that can be used to reduce latency in tactile internet systems using prediction techniques. This reduction may allow local devices in conjunction with haptic devices to have a greater margin of work. Also, show the feasibility of developing these proposals using a hardware-based development strategy such as reconfigurable computing (in FPGA) to minimize the execution time of the algorithms associated with the techniques. Several linear and non-linear prediction techniques have been implemented. The developed hardware used a totally parallel implementation. For linear techniques, a design was developed using a 32-bit floating-point representation and for non-linear techniques, fixed-point implementations with different configurations (18.14, 16.12, and 14.10) were developed. All the details of the implementation were presented, as well as the results of the synthesis and the bit precision simulations. The synthesis was performed for several bit size resolutions and three different numbers of implementations in parallel (1, 3, and 6). To characterize the proposed hardware, curves were generated, using the obtained synthesis data, to predict hardware consumption and throughput for all bit sizes. Besides, comparison results were presented about throughput, hardware occupation, and energy savings with other proposals in the literature.

Chapter 3

Proposal of Takagi–Sugeno Fuzzy-PI Controller Hardware

This Chapter proposes dedicated hardware for an intelligent control system on Field Programmable Gate Array (FPGA). The intelligent system is represented as Takagi-Sugeno Fuzzy-PI controller. The implementation uses a fully parallel strategy associated with a hybrid bit format scheme (fixed-point and other floating-point). Two hardware designs are proposed; the first one uses a single clock cycle processing architecture, and the other uses a pipeline scheme. The bit accuracy was tested by simulation with a non linear control system of robotic manipulator. The area, throughput, and dynamic power consumption of the implemented hardware are used to evaluate and compare the results of this proposal. The results achieved allow that the proposal hardware can use in several applications with high-throughput, low-power and ultra-low-latency restrictions such as teleoperation of robot manipulators, tactile internet, industrial automation in industry 4.0, and others.

3.1 Introduction

Systems based on Fuzzy Logic (FL), have been used in many industrial and commercial applications such as robotics, automation, control, and classification problems. Unlike high data volume systems, such as Big Data and Mining of Massive Datasets (MMD) (Poli 2015, Nasrollahzadeh et al. 2017, Yaqoob et al. 2016), one of the great advantages of Fuzzy Logic is its ability to work with incomplete or inaccurate information.

Intelligent systems based on production rules that use Fuzzy Logic in the inference process are called in the literature Fuzzy Systems (FS) (Oviedo et al. 2004). Among the existing inference strategies, the most used, the Mamdani and the Takagi–Sugeno, are differentiated by the final stage of the inference process (Aguilar et al. 2014, Hassan et al. 2016, Boncalo et al. 2019, Bicakci 2019, Banjanovic-Mehmedovic & Husejnovic 2019, Akbatı et al. 2019, Sánchez-Solano et al. 2013, Sánchez-Solano et al. 2010, Baturone et al. 2004, Youssef et al. 2018, Khati et al. 2019, Sun et al. 2015, Deliparaschos et al. 2006, de la Cruz-Alejo et al. 2019, Huang et al. 2019, Krim et al. 2019).

The interest in the development of dedicated hardware implementing Fuzzy Systems has increased due to the demand for high-throughput, low-power, and ultra-low-latency

control systems for emerging applications such as the tactile Internet (Junior et al. 2019, Simsek et al. 2016*b*), the Internet of Things (IoT), and Industry 4.0, where the problems associated with processing, power, latency, and miniaturization are fundamental. Robotic manipulators used on the tactile internet need a high-throughput and ultra-low-latency control system, and this can be achieved with dedicated hardware (Junior et al. 2019).

The development of dedicated hardware, in addition to speeding up parallel processing, makes it possible to operate with clocks adapted to low-power consumption (Torquato & Fernandes 2019, Da Costa et al. 2019*a*, Silva et al. 2019*a*, Coutinho et al. 2019*a*, Blaiech et al. 2019, Lopes et al. 2019*b*, Noronha et al. 2019*b*). The works presented in (Chowdhury & Saha 2008, Ontiveros-Robles et al. 2016, Prado et al. 2012, Loan & Murshid 2013, Loan et al. 2013, Titinchi & Halasa 2019, Zavala & Nieto 2012, Bosque et al. 2014) propose implementations of FS on reconfigurable hardware (Field Programmable Gate Array—FPGA), showing the possibilities associated with the acceleration of fuzzy inference processes having a high degree of parallelization. Other works propose specific implementations of Fuzzy Control Systems (Fuzzy CS) using the Mamdani Fuzzy Inference Machine (M-FIM) and the Takagi–Sugeno Fuzzy Inference Machine (TS-FIM) (Aguilar et al. 2014, Hassan et al. 2016, Boncalo et al. 2019, Bicakci 2019, Banjanovic-Mehmedovic & Husejnovic 2019, Akbatı et al. 2019, Sánchez-Solano et al. 2013, Sánchez-Solano et al. 2010, Baturone et al. 2004, Youssef et al. 2018, Khati et al. 2019, Sun et al. 2015, Deliparaschos et al. 2006, de la Cruz-Alejo et al. 2019, Huang et al. 2019, Krim et al. 2019). The works presented in (Tchendjou et al. 2018, Liviu 2018, Júnior et al. 2018) propose the Takagi–Sugeno hardware acceleration for other types of application fields.

This work aims to develop a new hardware proposal for a Fuzzy-PI controller with TS-FIM. Unlike most of the works presented, this project offers a fully parallel scheme associated with a hybrid platform using fixed-point and floating-point representations. Two TS-FIM hardware modules have been proposed, the first (here called TS-FIM module one-shot) takes one sample time to execute the TS-FIM, and the second (called here the TS-FIM module pipeline) uses registers inside the TS-FIM. Two pieces of Fuzzy-PI controller hardware have been proposed, one for the TS-FIM one-shot module and another for the TS-FIM module pipeline. The proposed hardware has been implemented, tested, and validated on a Xilinx Virtex 6 FPGA (6-bit LUTs). The synthesis results, in terms of size, resources, and throughput, are presented according to the number of bits and the type of numerical precision. The physical area on the target FPGA reaches less than 7%. The implementation achieved a throughput between 10 and 18Msps (Mega samples per second), and between 490 and 882 Mflips (Mega fuzzy logic inferences per second). Validation results on a feedback control system are also presented, in which satisfactory performance has been obtained for a small number of representation bits. Comparisons of results with other proposals in the literature in terms of throughput, hardware resources, and dynamic power savings will also be presented.

3.2 Related Works

In (Chowdhury & Saha 2008), a high-performance FPGA Mamdani fuzzy processor is presented. The processor achieved a throughput of about 5 Mflips at a clock frequency of about 40 MHz, and it was designed for 256 rules and 16 inputs with 16 bits. The proposal used a semi-parallel implementation and thus reduced the number of the operations per Hz. The work presented in (Chowdhury & Saha 2008) has about $\frac{5}{40} = 0.125$ flips/Hz and the work proposed here can achieve about $\frac{256*40}{40} = 256$ flips/Hz due to the fully parallel hardware scheme used. The significant difference between throughput and operation frequency also implies a high power consumption (McCool et al. 2012). The work presented in (Ontiveros-Robles et al. 2016) uses a Mamdani inference machine and the throughput is about 48.23 Mflips. The hardware was designed to operate with eight bits, four inputs, nine rules, and one output. Similar to the work presented in (Chowdhury & Saha 2008), the proposal introduced in (Ontiveros-Robles et al. 2016) adopted a semi-parallel implementation, and this way decreased the throughput and increased power consumption. Other Mamdani implementations following the same strategy are also found in (Prado et al. 2012, Loan & Murshid 2013, Loan et al. 2013, Titinchi & Halasa 2019).

A multivariate Takagi–Sugeno fuzzy controller on FPGA is proposed in (Aguilar et al. 2014). The hardware is applied to the temperature and humidity controller for a chicken incubator, and it was projected to two inputs, six rules, and three outputs. When compared to other works, the hardware proposed in (Aguilar et al. 2014) achieved a low throughput of about 6 Mflips. A hardware accelerator architecture for a Takagi–Sugeno fuzzy controller is proposed in (Boncalo et al. 2019), and this proposal achieved a throughput about 1.56 Msps with three inputs, two outputs, and 24 bits.

In (Sánchez-Solano et al. 2013, Sánchez-Solano et al. 2010, Baturone et al. 2004), a design methodology for rapid development of fuzzy controllers on FPGAs was developed. For the case with two inputs, 35 rules and one output (vehicle parking problem), the proposed hardware achieved a maximum clock of about 66.251 MHz with 10 bits. However, the TS-FIM takes 10 clocks to complete the inference step, and this decreases the throughput, and it increases the power consumption.

The implementation presented in (Youssef et al. 2018) aims at creating a hardware scheme of a fuzzy logic controller on FPGA for the maximum power point tracking in photo-voltaic systems. The implementation takes six clock cycles over 10 MHz, and this is equivalent to a throughput of about $\frac{10\,\text{MHz}}{6} \approx 1.67$ Msps. In (Sun et al. 2015), a Mamdani fuzzy logic controller on FPGA was proposed. The hardware carries out a throughput of about 25 Mflips with two inputs, 49 rules.

The work presented in (Deliparaschos et al. 2006) implements a semi-parallel digital fuzzy logic controller on FPGA. The work achieved about 16 Msps per clock frequency of 200 MHz, that is, 0.08 Msps/MHz. On the other hand, this manuscript uses a fully parallel approach, and it achieves 1 Msps/MHz; in other words, it can execute more operations per clock cycle. In the same direction, the proposals presented in (de la Cruz-Alejo et al. 2019, Krim et al. 2019) shows a semi-parallel fuzzy control hardware with low-throughput, about 1 Msp when compared to other works.

Thus, this manuscript proposes a hardware architecture for the Fuzzy-PI control sys-

tem. Unlike the works presented in the literature, the strategy proposed here uses a fully parallel scheme associated with a hybrid use in the bit format (fixed and floating-point). After several comparisons with other implementations of the literature, the scheme proposed here showed significant gains in processing speed (throughput) and dynamic power savings.

3.3 Takagi–Sugeno Fuzzy-PI Controller

The system described in Figure 1.2 in Chapter 1 highlights the components needed to build a tactile system. The Fuzzy CS module which is responsible for the control feedback of the robotic system. The input and output variables addressed in the system being $r^{sd}(n)$, $r^{mcs}(n)$, and $h(n)$ will be recognized from this point by $y(n)$, $y_{sp}(n)$, and $r(n)$, respectively. Figure 3.1 shows the Fuzzy-PI intelligent control system operating a generic plant (Oviedo et al. 2004, Silva et al. 2019*b*, Fernandes 2016). The plant output variable , $y(t)$, is called the controlled variable (or controlled signal), and it can admit several kinds of physical measurements such as level, angular velocity, linear velocity, angle, and others depending on the plant characteristics. The controlled variable, $y(t)$, passes through a sensor that converts the physical measure into a proportional electrical signal that is discretized at a sampling rate, t_s, generating the signal, $y(n)$.

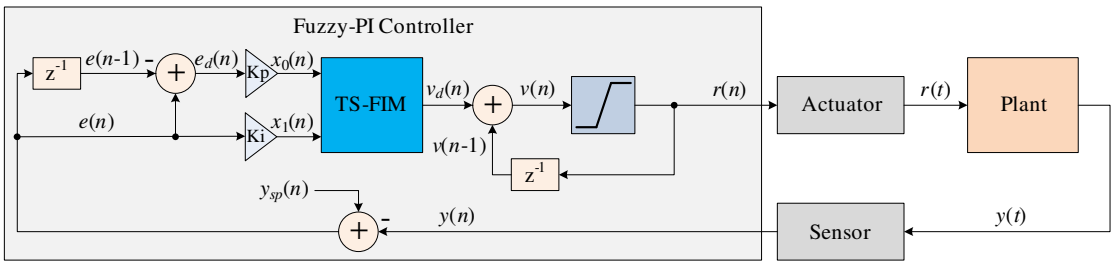

Figure 3.1: Architecture of the Fuzzy-PI feedback control system operating a generic plant.

The plant drives the kind of sensor that will be used. For level control in tanks used in industrial automation, the sensor can be characterized by the pressure sensor. For robotics applications (manipulators or mobile robotics), the sensor can be a position (capture angle information) or an encoder sensor (capture angular or linear velocity information).

In the n-th time, the Fuzzy-PI controller (see Figure 3.1) uses the signal, $y(n)$, and it calculates the error signal, $e(n)$, and difference of error, $e_d(n)$. The signal $e(n)$ is expressed by

$$e(n) = y_{sp}(n) - y(n), \tag{3.1}$$

where the $y_{sp}(n)$ is the reference signal, also called the set point variable, and the signal $e_d(n)$ is expressed by

$$e_d(n) = e(n) - e(n-1). \tag{3.2}$$

After the computation of the signals $e(n)$ and $e_d(n)$, the Fuzzy-PI controller generate the signals $x_0(n)$ and $x_1(n)$, which can be expressed as

$$x_0(n) = \text{Kp} \times e_d(n) \tag{3.3}$$

and

$$x_1(n) = \text{Ki} \times e(n). \tag{3.4}$$

The variables Kp and Ki represent the proportional and the integration gains, respectively (Oviedo et al. 2004, Silva et al. 2019*b*, Fernandes 2016). Subsequently, the signals $x_0(n)$ and $x_1(n)$ are sent to the Takagi-Sugeno fuzzy inference; called in this article Takagi-Sugeno - Fuzzy Inference Machine or TS-FIM (see Figure 3.1).

The TS-FIM is formed by three stages called fuzzification, operation of the rules (or rules evaluation), and defuzzification (the output function) (Oviedo et al. 2004). In the fuzzification, each i-th input signal $x_i(n)$ is applied to a set of F_i pertinence functions whose output can be expressed as

$$f_{i,j}(n) = \mu_{i,j}(x_i(n)) \text{ for } j = 0, \ldots, F_i, \tag{3.5}$$

where $\mu_{i,j}(\cdot)$ is the j-th membership function of the i-th input and $f_{i,j}(n)$ is the output of the fuzzification step associated with the j-th membership function and the i-th input in the n-th time.

For two inputs, $x_0(n)$ and $x_1(n)$, the TS-FIM generates a set of $F_0 + F_1$ fuzzy signals ($f_{0,j}$ and $f_{1,j}$) and these signals are processed by a set of F_0F_1 rules in the rules evaluation step. Each g-th rule can be expressed as

$$o_g = \min(f_{0,l}, f_{1,k}) \text{ for } g = 0, \ldots, (F_0F_1) - 1, \tag{3.6}$$

where $g = F_{0,l+k}$ for $(l,k) = (0,0), (0,1), \ldots, (F_0 - 1, F_1 - 1)$. Finally, the output (defuzzification) of TS-FIM, called here $v_d(n)$, can be expressed as

$$v_d(n) = \frac{a(n)}{b(n)} = \frac{\sum_{g=1}^{(F_0F_1)-1} a_g}{\sum_{g=0}^{(F_0F_1)-1} o_g} = \frac{\sum_{g=1}^{(F_0F_1)-1} o_g \times (A_g x_0(n) + B_g x_1(n) + C_g)}{\sum_{g=0}^{(F_0F_1)-1} o_g}, \tag{3.7}$$

where A_g, B_g, and C_g are parameters defined during the project (Oviedo et al. 2004). Thus, it can be said that every n-th instant TS-FIM receives as input $x_0(n)$ and $x_1(n)$ and generates as output $v(n)$, that is,

$$v_d(n) = \mathit{TSFIM}\,(x_0(n), x_1(n))\,, \tag{3.8}$$

where $\mathit{TSFIM}\,(\cdot)$ is a function that represents TS-FIM.

After the TS-FIM processing, the Fuzzy-PI controller integrates the signal $v_d(n)$ generating the signal $v(n)$ (see Figure 3.1). The signal $v_d(n)$ is the output of the Fuzzy-PI controller, and it can be expressed as

$$v(n) = v_d(n) + v(n-1). \tag{3.9}$$

This signal saturates beyond $v_{\min}$ and $v_{\max}$, generating the signal $r(n)$ that it is expressed as

$$r(n) = \begin{cases} v_{\max} & \text{for } v(n) > v_{\max} \\ v(n) & \\ v_{\min} & \text{for } v(n) < v_{\min} \end{cases}. \tag{3.10}$$

Finally, the signal $r(n)$ is sent to an actuator, which transforms the discrete signal into a continuous signal, $r(t)$, to be applied to the plant.

3.4 Hardware Proposal

The general structure of the proposed hardware implementation, is composed of three main modules called input processing (IPM), TS-FIM (TS-FIMM) and integration (IM), as shown in Figure 3.2. The hardware was developed for the most part using a fixed-point format for the variables, in which, for any given variable, the notations [uT.W] and [sT.W] indicate that the variable is formed by T bits of which W are intended for the fractional part and the symbols "u" and "s" indicate that the variable is signed or unsigned, respectively. For signed variables, the number of bits intended for the integer part is $T - W - 1$ and, for unsigned variables, the number of bits for the integer part is $T - W$.

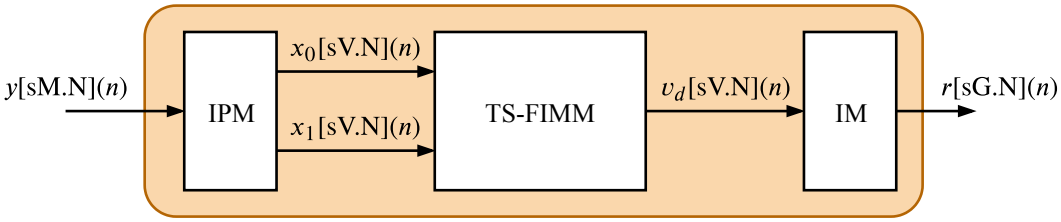

Figure 3.2: Overview of Fuzzy-PI controller proposed architecture.

3.4.1 Input Processing Module (IPM)

The IPM (shown in Figure 3.3) is responsible for processing the control signal sent by the plant to the input of the Fuzzy-PI controller. The IPM computes the Equations 3.1, 3.2, 3.3, and 3.4. The signals associated with this module were implemented with M bits, where one is reserved for the sign and N for the fractional part, so the value of M can be expressed as

$$M = N + \log_2(\lceil y_{max} \rceil) + 1, \tag{3.11}$$

where y_{max} represents the maximum value, in modulus, of the process variable, $y(n)$.

The constant parameters Kp and Ki of the gain modules (see equations 3.3 and 3.4) must be designed to maintain the output signals $x_0[\text{V.N}](n)$ and $x_1[\text{V.N}](n)$ between -1 and 1. However, it is important to note that the two gain modules can saturate the signal in $[\text{V.N}](n)$ bits after the multiplication.

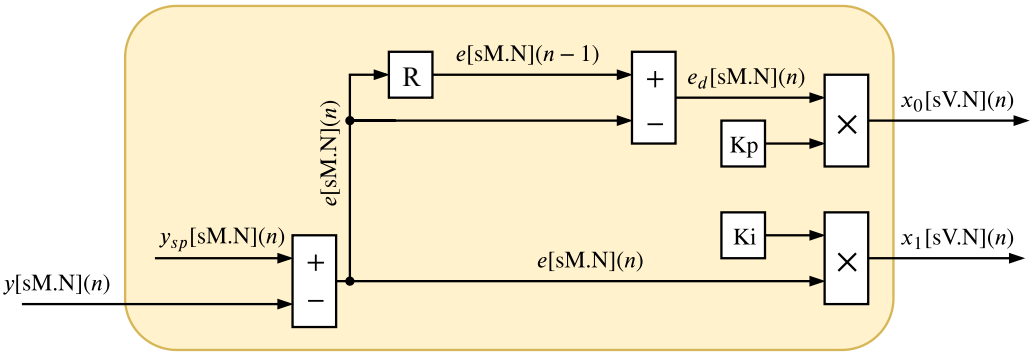

Figure 3.3: Hardware architecture of IPM.

3.4.2 TS-FIM Module (TS-FIMM)

The TS-FIMM is composed of three hardware components: Membership Function Module (MFM), Operation Module (OM), and Output Function Module (OFM). The MFM is the first module associated with TS-FIMM, and it corresponds to the fuzzification process, the OM component completes the rules evaluation phase and the OFM performs the defuzzification step (see Section 3.3). This work proposes two alternative architectures for the TS-FIMM.

The first proposed architecture, presented in Figure 3.4 and called here TS-FIMM One-Shot (TS-FIMM-OS) , performs all modules MFM, OM, and OFM in one sampling time, in other words, it takes a time interval between samples to generate the n-th output associated with the n-th input. The second one, presented in Figure 3.5 and called here TS-FIMM Pipeline (TS-FIMM-P), uses registers (blocks called R in Figure 3.4) among the input, MFM, OM, OFM, and output. The TS-FIMM-P has a latency of four sampling times to perform all modules MFM, OM, and OFM, in other words, there is a delay of four samples between the n-th output and n-th input.

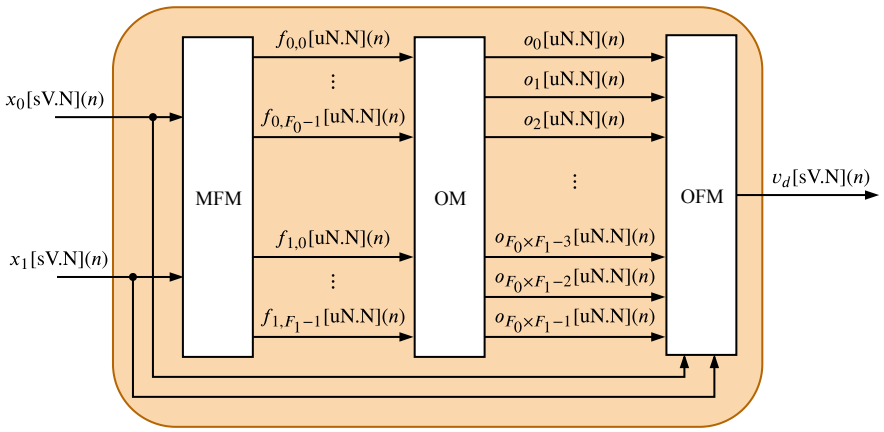

Figure 3.4: Hardware architecture of TS-FIMM One-Shot (TS-FIMM-OS).

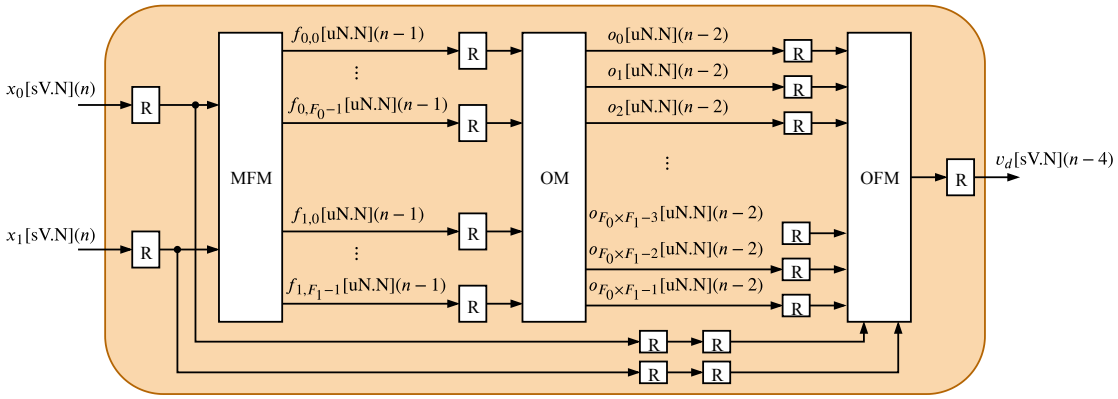

Figure 3.5: Hardware architecture of TS-FIMM Pipeline (TS-FIMM-P).

The TS-FIMM-OS will finally have a longer execution time than TS-FIMM-P because it has a longer critical path; however, the TS-FIMM-OS does not have a delay. It is important to empathize that the delay inside the feedback control can take a system to instability. Indeed, the instability degree depends on the system and how long the delay is. The instability will depend on the characteristics of the system and the size of the delay (Niculescu et al. 2010). On the other hand, the pipeline scheme associated with the TS-FIMM-P has a shorter critical path, which allows a higher throughput compared to the TS-FIMM-OS.

Membership Function Module (MFM)

In the MFM, each i-th input variable is associated with a module that collects F_i membership functions, called here the Membership Function Group (MFG). Figure 3.6 shows the i-th MFG, or MFG-i, related to the i-th input $x_i[\text{sV.N}](n)$.

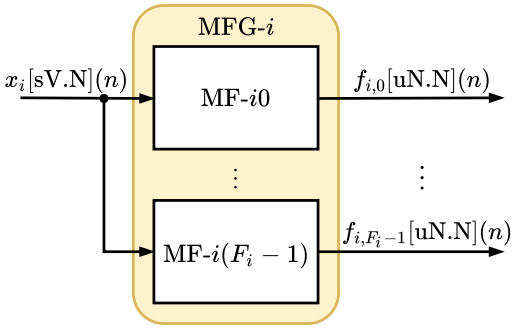

Figure 3.6: Hardware architecture of module MFG-i associated with the i-th input, $x_i[\text{sV.N}](n)$.

Each MFG-i collects F_i membership functions (see Figure 3.6) called MF-ij and each module MF-ij implements the j-th membership function associated with the i-th input, $\mu_{i,j}(x_i(n))$. Each n-th time, all membership functions MF-ij are executed in parallel producing at the output a N bits unsigned signal of type uN.N, without the integer part, called

$f_{i,j}[\text{uN.N}](n)$ (see Figure 3.6). The Fuzzy-PI controller proposed here uses $F_0 + F_1$ membership functions.

Figure 3.7 shows the membership functions implemented in the MFM. For both variables, $x_0[\text{sV.N}](n)$ and $x_1[\text{sV.N}](n)$, seven functions of pertinence were created (trapezoidal at the ends and triangular at the middle). The terms associated with the membership functions are Large Negative (LN), Moderate Negative (MN), Small Negative (SN), Zero (ZZ), Small Positive (SP), Moderate Positive (MP) and Large Positive (LP).

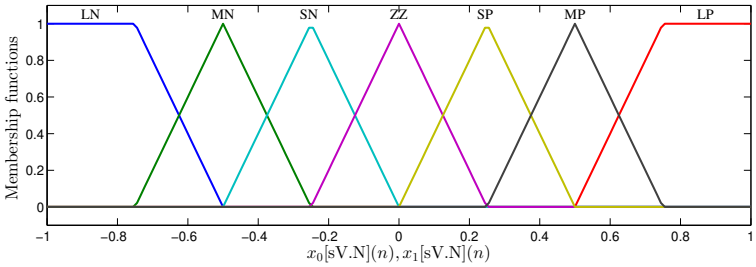

Figure 3.7: Membership functions from inputs $x_0[\text{sV.N}](n)$ and $x_1[\text{sV.N}](n)$.

Each j-th membership function associated with the i-th input was implemented directly on hardware based on the following expressions:

$$\mu_{i,j}^{RT}(x_i[\text{sV.N}](n)) = \begin{cases} 0 & \text{if } x_i[\text{sV.N}](n) > d_{i,j}[\text{sW.T}] \\ G_{i,j}^{RT}(n) & \text{if } c_{i,j}[\text{sW.T}] \leq x_i[\text{sV.N}](n) \leq d_{i,j}[\text{sW.T}], \\ 1 & \text{if } x_i[\text{sV.N}](n) < c_{i,j}[\text{sW.T}] \end{cases} \tag{3.12}$$

being $\mu_{i,j}^{RT}(\cdot)$ the trapezoidal function LP, $c_{i,j}[\text{sW.T}]$ and $d_{i,j}[\text{sW.T}]$ are constants ($c_{i,j}[\text{sW.T} < d_{i,j}[\text{sW.T}]$) and

$$G_{ij}^{RT}(n) = \frac{d_{i,j}[\text{W.T}] - x_i[\text{sV.N}](n)}{d_{i,j}[\text{W.T}] - c_{i,j}[\text{W.T}]}, \tag{3.13}$$

where W is the total number of bits of the constants of the j-th activation function associated with i-th input and T is the number of bits of the fractional part.

The values of W and T will set the resolution of the activation functions. In the implementation proposed in this work, the value of W is always expressed as $\text{W} = 2 \times \text{T} + 1$.

For the trapezoidal function LN $\mu_{i,j}^{LT}(\cdot)$, we have

$$\mu_{i,j}^{LT}(x_i[\text{sV.N}](n)) = \begin{cases} 0 & \text{if } x_i[\text{sV.N}](n) < e_{i,j}[\text{sW.T}] \\ G_{i,j}^{LT}(n) & \text{if } e_{i,j}[\text{sW.T}] \leq x_i[\text{sV.N}](n) \leq f_{i,j}[\text{sW.T}], \\ 1 & \text{if } x_i[\text{sV.N}](n) > f_{i,j}[\text{sW.T}] \end{cases} \tag{3.14}$$

with $\mu_{i,j}^{LT}(\cdot)$ the left trapezoidal function, $e_{i,j}[\text{sW.T}]$ and $f_{i,j}[\text{sW.T}]$ constants ($e_{i,j}[\text{sW.T} < f_{i,j}[\text{sW.T}]$) and

$$G_{ij}^{LT}(n) = \frac{x_i[\text{sV.N}](n) - e_{i,j}[\text{W.T}]}{f_{i,j}[\text{W.T}] - e_{i,j}[\text{W.T}]}. \tag{3.15}$$

Triangular membership functions are expressed as

$$\mu^{T}_{i,j}(x_i[\text{sV.N}](n)) = \begin{cases} \mu^{LT}_{i,j}(x_i[\text{sV.N}](n)) & \text{if } x_i[\text{sV.N}](n) < m_{i,j}[\text{sW.T}] \\ \mu^{RT}_{i,j}(x_i[\text{sV.N}](n)) & \text{if } x_i[\text{sV.N}](n) \geq m_{i,j}[\text{sW.T}] \end{cases}, \quad (3.16)$$

where $m_{i,j}[\text{sW.T}]$ is the triangle center point, that is, $m_{i,j}[\text{sW.T}] = c_{i,j}[\text{sW.T}] = f_{i,j}[\text{sW.T}]$.

Non-linear pertinence functions can be implemented with lookup tables (LUTs). Although this implementation uses only two inputs ($x_0[\text{sV.N}](n)$ and $x_1[\text{sV.N}](n)$) and seven membership functions for each input, this can be easily extended to more inputs and functions, since the entire implementation is performed in parallel.

Operation Module (OM)

The $F_0 + F_1$ outputs from the MFM module are passed to the OM module that performs all operations relative to the F_0F_1 rules, as described in Equation 3.6 in Section 3.3. Figure 3.8 details the hardware structure of one of the F_0F_1 operating modules, here called O-lk, which performs the minimum operation ("AND" connector) between the l-th membership function from input 0, $f_{0,l}[\text{nN.N}](n)$, with the k-th membership function from input 1, $f_{1,k}[\text{uN.N}](n)$ (see Equation 3.7).

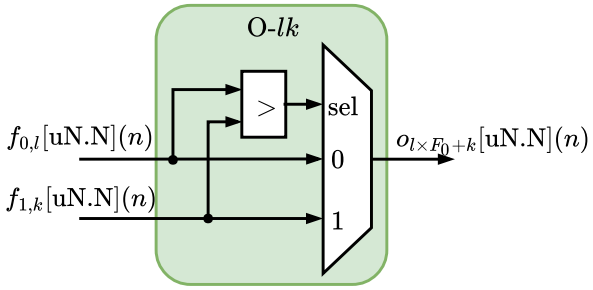

Figure 3.8: Arquitecture of the module O-lk associated with the operation between the fuzzyfied signal from the l-th membership function from input 0, $f_{0,l}[\text{nN.N}](n)$, with the k-th membership function from input 1, $f_{1,k}[\text{uN.N}](n)$ (see Equation 3.7).

Output Function Module (OFM)

The OFM, illustrated in Figure 3.9, performs the generation of the TS-FIMM output variable during the step called defuzzification. This step essentially corresponds to the implementation of the Equation 3.7 presented in Section 3.3. The blocks called NM and DM perform the numerator and denominator operations presented in Equation 3.7, respectively.

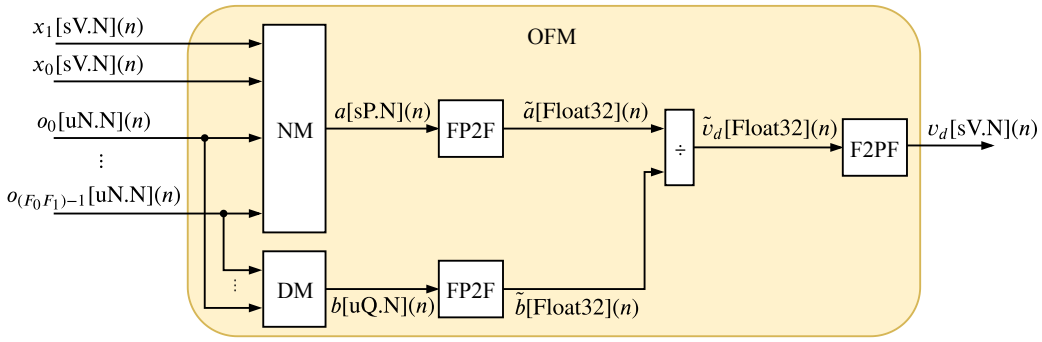

Figure 3.9: Hardware architecture of the OFM.

Figures 3.10 and 3.11 show the hardware implementation of the NM. The NM is composed of the F_0F_1 hardware components called WM-g and an adder tree structure. Each g-th WM-g, detailed in Figure 3.11, is a parallel hardware implementation of the variable a_g presented in Equation 3.7. The F_0F_1 WMs hardware components are also implemented in parallel and they generated F_0F_1 signals $a_g[\text{sH.N}](n)$ in each n-th time instant. Since $-1 < x_0[\text{V.N}](n) < 1$, $-1 < x_1[\text{V.N}](n) < 1$, $0 < o_g[\text{uN.N}](n) < 1$, $-1 < A_g < 1$, $-1 < B_g < 1$ and $-1 < C_g < 1$ for $g = 0, \ldots, F_0F_1$, the variable H can be expressed as $H = N + 3$.

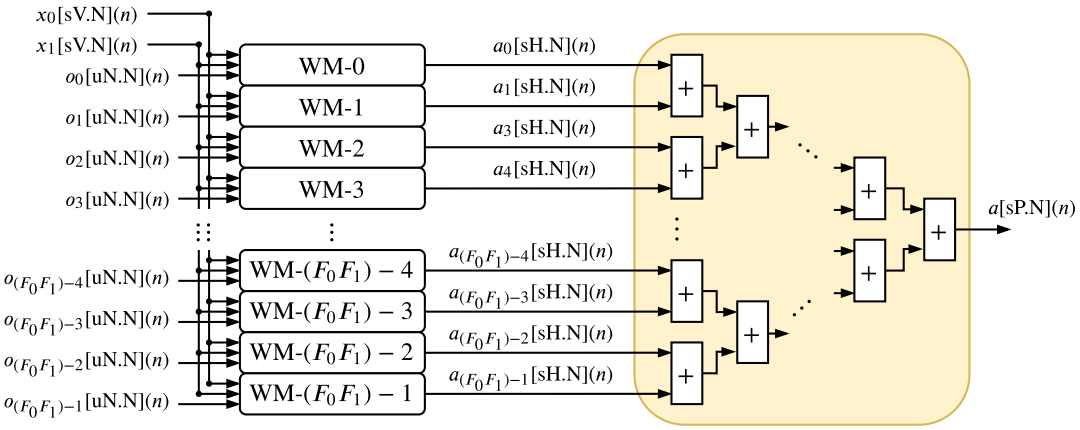

Figure 3.10: Hardware architecture of the NM.

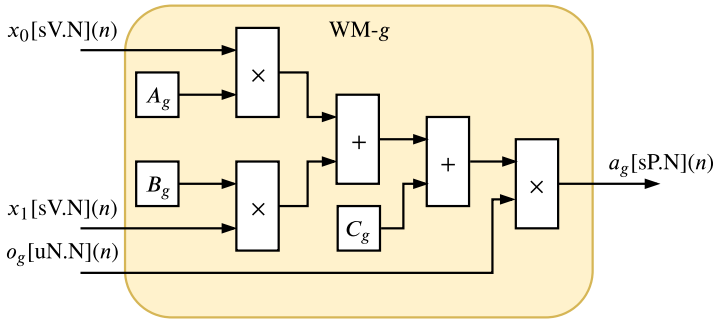

Figure 3.11: Hardware architecture of the WM-g.

The adder tree structure, illustrated in Figure 3.10, has a depth expressed as $\log_2(\lceil F_0F_1 \rceil)$; thus, the output signal $a(n)$ (see Equation 3.7) can be performed as $a[\text{sP.N}](n)$ where

$$P = H + \log_2(\lceil F_0F_1 \rceil). \tag{3.17}$$

The DM, presented in Figure 3.12, is characterized with an adder tree structure with depth also expressed as $\log_2(\lceil F_0F_1 \rceil)$. The output signal of DM can be expressed as $b[\text{sQ.N}](n)$ where

$$Q = N + \log_2(\lceil F_0F_1 \rceil) + 1. \tag{3.18}$$

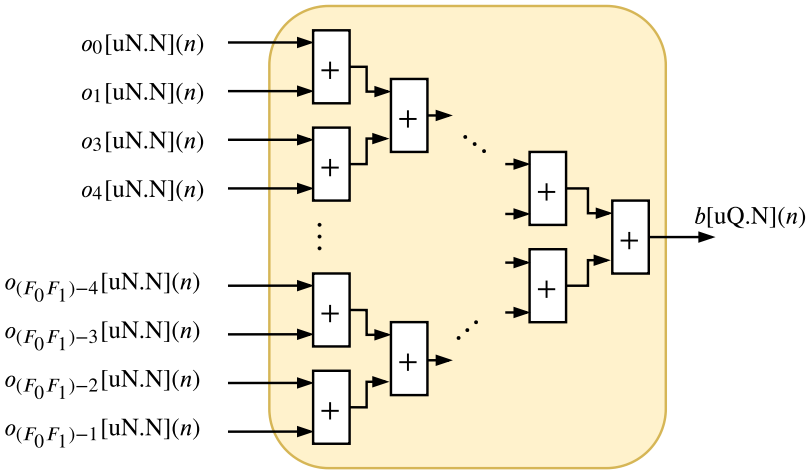

Figure 3.12: Hardware architecture of the DM.

For the division calculation, the output signals, in fixed-point, of the NM and DM modules ($a[\text{sP.N}](n)$ and $b[\text{sQ.N}](n)$ are transformed to a 32-bit floating-point (IEEE754) standard by the Fixed-point to Float (FP2F) module ($\tilde{a}[\text{Float32}](n)$ e $\tilde{b}[\text{Float32}](n)$) and after division the TS-FIMM output is converted back into fixed-point by the Float to Fixed-point (F2FP) module.

Since the TS-FIMM inputs and the values of A_g, B_g and C_g are between -1 and 1, it can be guaranteed, from Equation 3.7, that the output, $v_d[\text{sV.N}](n)$, continue normalized between -1 and 1. Thus, one can use the same input resolution, that is, N for the fractional part and $V = N + 1$ for the integer part, as shown in Figure 3.9.

3.4.3 Integration Module (IM)

The IM, shown in Figure 3.13, implements the Equation 3.9 presented in Section 3.3. This module is the last step on the Fuzzy-PI hardware, and it is composed of the accumulator with a saturation. The output signal, $r(n)$, is expressed as $r[\text{sG.N}](n)$, where

$$G = N + \log_2(\lceil v_{\max} - v_{\min} \rceil) + 1. \tag{3.19}$$

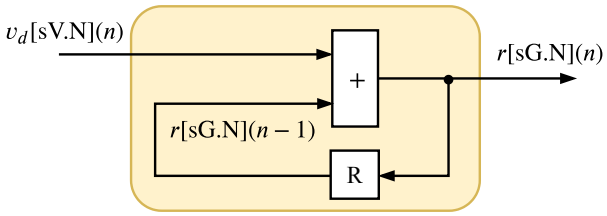

Figure 3.13: Hardware architecture of the IM.

3.5 Synthesis Results

The synthesis results were obtained for a Fuzzy-PI controller (see Figure 3.2) and also to specific modules TS-FIMM-OS (see Figure 3.4) and TS-FIMM-P (see Figure 3.5). The separate synthesis of the TS-FIMM allows for analysis of the Fuzzy inference algorithm core in the complete hardware proposal. All synthesis results used an FPGA Xilinx Virtex 6 (6-bits LUTs) xc6vlx240t-1ff1156 and that has 301,440 registers, 150,720 logical cells to be used as LUTs and 768 multipliers.

3.5.1 Synthesis Results—TS-FIMM Hardware

Tables 3.1 and 3.2 present the synthesis results related to hardware occupancy and the maximum throughput, $R_s = 1/t_s$, in Mega samples per second (Msps) of the system for several values of N and T. Tables 3.1 and 3.2 show the synthesis results associated with TS-FIMM-OS and TS-FIMM-P, respectively. The columns, NR, NLUT, and NMULT, represent the number of registers, logic cells used as LUTs, and multipliers in the hardware implemented in the FPGA, respectively. The PNR, PNLUT, and NMULT columns represent the percentage relative to the total FPGA resources.

Synthesis results show that the hardware proposal for TS-FIMM takes up a small hardware space of less than 1%, PR, in registers and less than 7% in LUTs, PLUT, of the FPGA (see Tables 3.1 and 3.2). These results enable the use of several TS-FIMM

Table 3.1: Synthesis results (hardware requirement and time) associated with TS-FIMM-OS hardware.

N	T	NR	PR	NLUT	PLUT	NMULT	PNMULT	t_s (ns)	R_s (Msps)
8	4	217	$\approx 0.07\%$	6,339	$\approx 4.21\%$	49	$\approx 6.38\%$	79.72	12.54
	6			6,381	$\approx 4.23\%$			80.95	12.35
	8			6,452	$\approx 4.28\%$			81.96	12.20
	10			6,598	$\approx 4.38\%$			83.76	11.94
10	4	259	$\approx 0.09\%$	6,772	$\approx 4.49\%$	49	$\approx 6.38\%$	84.18	11.88
	6			6,904	$\approx 4.58\%$			82.70	12.09
	8			7,331	$\approx 4.86\%$			83.94	11.91
	10			7,331	$\approx 4.86\%$			83.00	12.05
12	4	324	$\approx 0.11\%$	7,280	$\approx 4.83\%$	49	$\approx 6.38\%$	82.65	12.10
	6			7,916	$\approx 5.25\%$			83.28	12.01
	8			7,954	$\approx 5.28\%$			87.02	11.49
	10			8,147	$\approx 5.41\%$			85.99	11.63
14	4	384	$\approx 0.13\%$	8,761	$\approx 5.81\%$	49	$\approx 6.38\%$	84.12	11.89
	6			8,915	$\approx 5.91\%$			85.08	11.75
	8			8,999	$\approx 5.97\%$			86.39	11.58
	10			9,163	$\approx 6.08\%$			86.75	11.53
16	4	428	$\approx 0.14\%$	9,816	$\approx 6.51\%$	49	$\approx 6.38\%$	86.42	11.54
	6			9,990	$\approx 6.63\%$			84.80	11.79
	8			10,072	$\approx 6.68\%$			88.31	11.32
	10			10,252	$\approx 6.80\%$			88.65	11.28

implemented in parallel on FPGA, allowing for accelerating several applications in massive data environments. On the other hand, the low hardware consumption allows the use of TS-FIMM in small FPGAs of low cost and consumption for applications of IoT and M2M. Another important point to be analyzed, still in relation to the synthesis, is the linear behavior of the hardware consumption in relation to the number of bits, unlike the work presented in (de Souza & Fernandes 2014*c*), and this is important, since it makes possible the use systems with higher resolution.

The values of throughput, R_s, were very relevant, with values about 11.5 Msps for TS-FIMM-OS and values about 17 Msps for TS-FIMM-P. These values enable its application in various large volume problems for processing as presented in (Chowdhury & Saha 2008) or in problems with fast control requirements such as tactile internet applications (Simsek et al. 2016*b*, Junior et al. 2019). It is also observed that throughput has a linear behavior as a function of the number of bits.

The TS-FIMM-P with a speedup of about $1.47\times$ ($\frac{17\text{Msps}}{11.5\text{Msps}}$) is related to the TS-FIMM-OS. This speedup was driven by the critical path reduction with the pipeline scheme. However, the pipeline scheme in TS-FIMM-P used about $3.4\times$ registers (NR) more than TS-FIMM-OS.

Figures 3.14 and 3.15 show the behavior surfaces of the number of LUTs (NLUT) and throughput in function of N and T for TS-FIMM-OS, respectively. For both cases, an

Table 3.2: Synthesis results (hardware requirement and time) associated with TS-FIMM-P hardware.

N	T	NR	PR	NLUT	PLUT	NMULT	PNMULT	t_s (ns)	R_s (Msps)
8	4	746	$\approx 0.25\%$	5,326	$\approx 3.53\%$	49	$\approx 6.38\%$	56.73	17.62
	6			5,350	$\approx 3.55\%$			55.81	17.92
	8			5,422	$\approx 3.60\%$			56.18	17.80
	10			5,590	$\approx 3.71\%$			56.97	17.55
10	4	917	$\approx 0.30\%$	6,093	$\approx 4.04\%$	49	$\approx 6.38\%$	57.21	17.48
	6			6,141	$\approx 4.07\%$			57.88	17.28
	8			6,199	$\approx 4.11\%$			57.63	17.35
	10			6,317	$\approx 4.19\%$			56.72	17.63
12	4	1,113	$\approx 0.37\%$	6,910	$\approx 4.58\%$	49	$\approx 6.38\%$	57.90	17.27
	6			6,982	$\approx 4.63\%$			58.22	17.18
	8			7,016	$\approx 4.65\%$			58.60	17.06
	10			7,172	$\approx 4.76\%$			56.26	17.77
14	4	1,301	$\approx 0.43\%$	7,799	$\approx 5.17\%$	49	$\approx 6.38\%$	58.60	17.06
	6			7,823	$\approx 5.19\%$			58.22	17.18
	8			7,905	$\approx 5.24\%$			58.26	17.16
	10			8,031	$\approx 5.33\%$			60.00	16.66
16	4	1,477	$\approx 0.49\%$	8,713	$\approx 5.78\%$	49	$\approx 6.38\%$	59.43	16.83
	6			8,737	$\approx 5.80\%$			58.14	17.20
	8			8,819	$\approx 5.85\%$			57.89	17.27
	10			8,955	$\approx 5.94\%$			58.90	16.98

adjustment was made, through a regression technique, to find the plane that best matches the measured points. For the case of NLUT, the plane, $f_{\mathrm{NLUT}}(\mathrm{N},\mathrm{T})$, was expressed by

$$f_{\mathrm{NLUT}}(\mathrm{N},\mathrm{T}) \approx 1682 + 532.2 \times \mathrm{N} + 6.493 \times 10^{-13} \times \mathrm{T}, \tag{3.20}$$

with $R^2 = 0.9766$. For throughput, in Msps, a plane was found, $f_{R_s}(\mathrm{N},\mathrm{T})$, characterized as

$$f_{R_s}(\mathrm{N},\mathrm{T}) \approx 13.24 - 0.1163 \times \mathrm{N} + 3.414 \times 10^{-16} \times \mathrm{T}, \tag{3.21}$$

with $R^2 = 0.7521$.

The behavior surfaces of the number of LUTs (NLUT) and throughput in function of N and T for TS-FIMM-P are presented in Figures 3.16 and 3.17, respectively. For the case of NLUT, the plane, $f_{\mathrm{NLUT}}(\mathrm{N},\mathrm{T})$, was expressed by

$$f_{\mathrm{NLUT}}(\mathrm{N},\mathrm{T}) \approx 1171 + 491.1 \times \mathrm{N} + 4.245 \times 10^{-13} \times \mathrm{T}, \tag{3.22}$$

with a $R^2 = 0.9838$. For throughput in Msps, a plane was found, $f_{R_s}(\mathrm{N},\mathrm{T})$, characterized as

$$f_{R_s}(\mathrm{N},\mathrm{T}) \approx 18.48 - 0.09704 \times \mathrm{N} - 5.365 \times 10^{-16} \times \mathrm{T}, \tag{3.23}$$

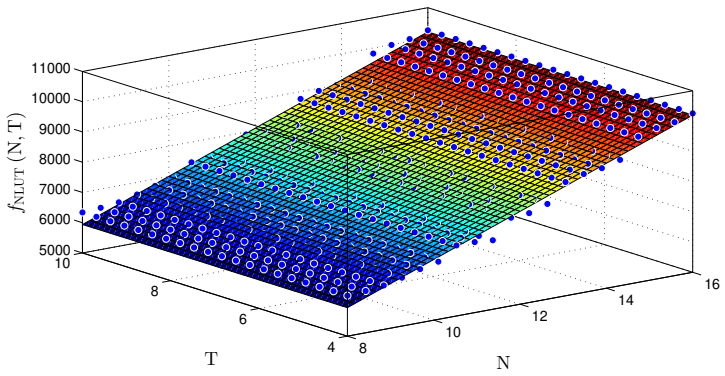

Figure 3.14: Plane, $f_{\mathrm{NLUT}}(\mathrm{N},\mathrm{T})$, found to estimate the number of LUTs as a function of the number of bits N and T for TS-FIMM-OS.

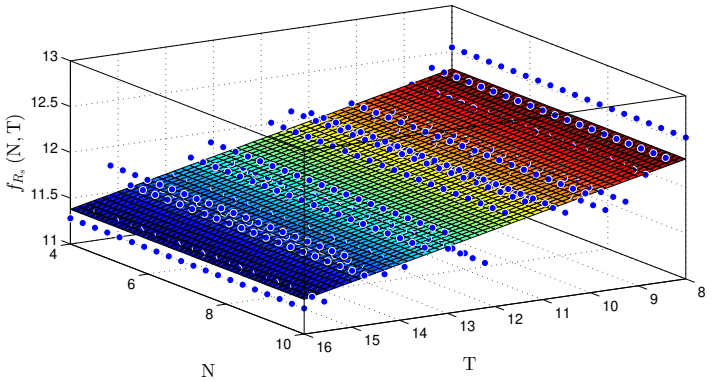

Figure 3.15: Plane, $f_{R_s}(\mathrm{N},\mathrm{T})$, found to estimate throughput, R_s, for different number of bits N and T for TS-FIMM-OS.

with $R^2 = 0.5366$.

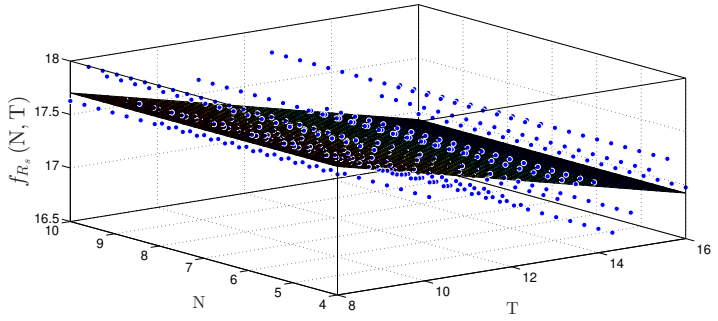

Figure 3.17: Plane, $f_{R_s}(\mathrm{N},\mathrm{T})$, found to estimate throughput, R_s, for different number of bits N and T for TS-FIMM-P.

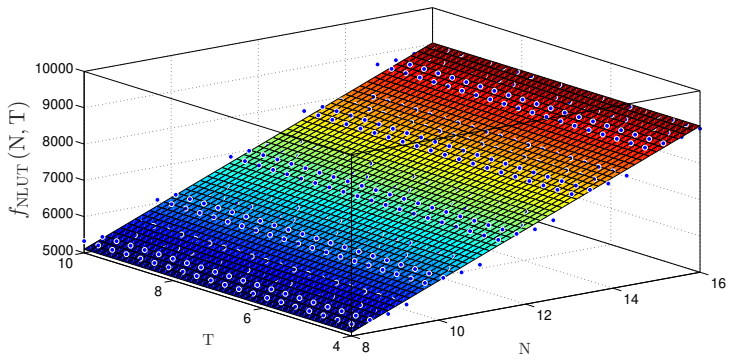

Figure 3.16: Plane, $f_{\mathrm{NLUT}}(\mathrm{N},\mathrm{T})$, found to estimate the number of LUTs as a function of the number of bits N and T for TS-FIMM-P.

3.5.2 Synthesis Results—Fuzzy-PI Controller Hardware

Tables 3.3 and 3.4 present the synthesis results related to hardware occupancy and throughput, R_s, for the Fuzzy-PI controller hardware (see Figure 3.2). The results are presented for several values of N and $T = 10$.

Table 3.3: Synthesis results (hardware requirement and time) associated with Fuzzy-PI controller hardware with TS-FIMM-OS.

N	NR	PR	NLUT	PLUT	NMULT	PNMULT	t_s (ns)	R_s (Msps)
8	261	$\approx 0.09\%$	6834	$\approx 4.53\%$	49	$\approx 6.38\%$	92.87	10.77
10	307	$\approx 0.10\%$	7331	$\approx 4.86\%$	49	$\approx 6.38\%$	98.44	10.16
12	375	$\approx 0.12\%$	8409	$\approx 5.58\%$	49	$\approx 6.38\%$	98.68	10.13
14	438	$\approx 0.15\%$	9460	$\approx 6.28\%$	49	$\approx 6.38\%$	99.98	10.00
16	488	$\approx 0.16\%$	10,595	$\approx 7.03\%$	49	$\approx 6.38\%$	104.31	9.59

Table 3.4: Synthesis results (hardware requirement and time) associated with Fuzzy-PI controller hardware with TS-FIMM-P.

N	NR	PR	NLUT	PLUT	NMULT	PNMULT	t_s (ns)	R_s (Msps)
8	790	$\approx 0.26\%$	5,826	$\approx 3.87\%$	49	$\approx 6.38\%$	66.08	15.13
10	965	$\approx 0.32\%$	6,317	$\approx 4.19\%$	49	$\approx 6.38\%$	72.16	13.86
12	1,164	$\approx 0.39\%$	7,434	$\approx 4.93\%$	49	$\approx 6.38\%$	68.95	14.50
14	1,355	$\approx 0.45\%$	8,328	$\approx 5.53\%$	49	$\approx 6.38\%$	73.23	13.66
16	1,537	$\approx 0.51\%$	9,298	$\approx 6.17\%$	49	$\approx 6.38\%$	74.56	13.41

Synthesis results, drawn on Tables 3.3 and 3.4, show that the proposed implementation requires a small fraction of hardware space, less than 1%, PR, in registers and less than 8%

in LUTs, PLUT, of the FPGA. In addition, it is possible to see the numbers of embedded multipliers, PNMULT, remained below 7%. This occupation enables the use of several Fuzzy-PI controllers in parallel in the same FPGA hardware, and this allows various control systems running in parallel on industrial applications. The low size implementation also allows the use in low cost and power consumption IoT and M2M applications. Regarding throughput, R_s, the results obtained were highly relevant, with values between 15.33, and 13.41 Msps, which enables its application in several problems with large data volume for processing as presented in (Chowdhury & Saha 2008) or in problems with fast control requirements such as tactile internet applications (Junior et al. 2019).

3.6 Validation Results

3.6.1 Validation Results—TS-FIMM Hardware

Figures 3.18 and 3.19 show the mapping between input ($x_0(n)$ and $x_1(n)$) and output $v_d(n)$ for proposed hardware and a reference implementation with Fuzzy Matlab Toolbox (License number 1080073) (MATLAB 2012), respectively. The Matlab implementation, shown in Figure 3.19, uses floating-point format with 64 bits (double precision) while in Figure 3.18 the proposed hardware-generated mapping is presented using lower resolution synthesized ($N = 8$, $V = 9$ and $T = 4$). These figures are able to present a qualitative representation of the proposed implementation, in which the obtained results are quite similar to those expected.

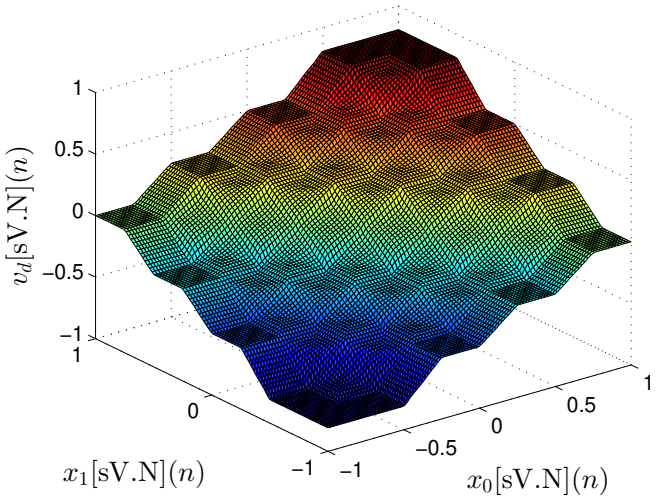

Figure 3.18: Mapping between input and output from TS-FIMM hardware using fixed-point with N = 8, V = 9 and T = 4.

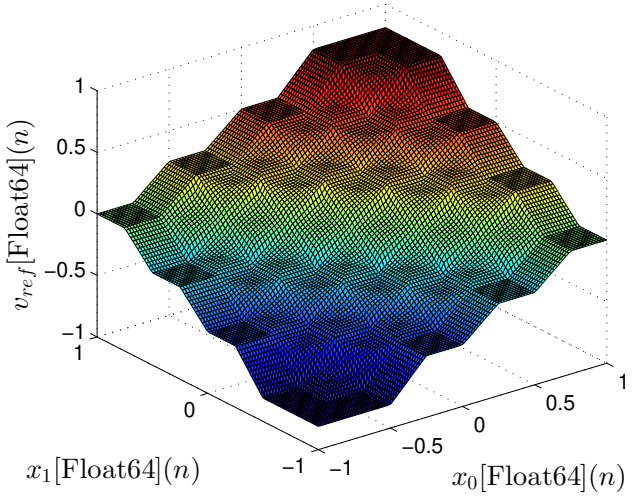

Figure 3.19: Mapping between input and ouput from TS-FIMM generated by Matlab Fuzzy Logic Toolbox using a double format.

Table 3.5 shows the mean square error (MSE) between the Fuzzy Matlab Toolbox and the proposed hardware implementation for several cases N and T. For the experiment, the calculation of MSE is expressed as

$$MSE = \frac{1}{Z}\sum_{n=0}^{Z-1}\left(v_{ref}[\text{Float64}](n) - v_d[\text{sV.N}](n)\right)^2, \tag{3.24}$$

where Z represents the number of tested points that corresponded to $10,000$ points spread evenly within the limits of the input values (-1 and 1). Figures 3.18 and 3.19 were generated with these points.

The results obtained in relation to MSE were also quite significant, showing that the TS-FIMM hardware has a response quite similar to the implementation with 64 bits even for a fixed-point resolution of 8 bits ($MSE = 2.395 \times 10^{-6}$). Another interesting fact was related to the values of T that did not significantly influence the MSE value for the pertinence functions used (see Figure 3.7) in the project. It is important to note that the implementation of TS-FIMM hardware with few bits leads to smaller hardware, low-power consumption or high-throughput values.

3.6.2 Validation Results—Fuzzy-PI Controller Hardware

In order to validate the results of the Fuzzy-PI controller in hardware, bit-precision simulation tests were performed with a nonlinear dynamic system characterized by a robotic manipulator system called the Phantom Omni (Song et al. 2006, Sansanayuth et al. 2012, Silva et al. 2009*a*, Al-Wais et al. 2016). The Phantom Omni is a 6-DOF (Degree Of Freedom) manipulator, with rotational joints. The first three joints are actuated, while the last three joints are non-actuated (Al-Wais et al. 2016), as described in Chapter 2 Section 2.7.

Table 3.5: Mean square error (MSE) between the Fuzzy Matlab Toolbox and the proposed hardware implementation for several cases N and T.

N	T	*MSE* (see Equation 3.24)
8	4 6 8 10	2.4×10^{-6}
10	4 6 8 10	1.3×10^{-7}
12	4 6 8 10	7.2×10^{-9}
14	4 6 8 10	4.9×10^{-10}
16	4 6 8 10	2.7×10^{-11}

Figure 3.20 shows the simulated system where the plant is the 3DOF Phantom Omni robotic manipulator. The controlled variables are the angular position of the joints θ_1, θ_2, and θ_3 and the actuator variables are the torques τ_1, τ_2, and τ_3. The control system has three angular position sensors and each i-th Sensor-i convert the i-th continuous angle signal, $\theta_i(t)$ to discrete angle signal, $\theta_i(n)$. There are three pieces of Fuzzy-PI hardware running in parallel and every i-th Sensor-i is connected with Fuzzy-PI hardware, Fuzzy-PI-i. Each piece of i-th Fuzzy-PI-i hardware generates the i-th discrete torques acting signal, $\tau_i(n)$, and every i-th discrete torque signal, $\tau_i(n)$, is connected to i-th actuator, Actuator-i. Finally, each i-th actuator, Actuator-i, generates the i-th continuous torque signal, $\tau_i(t)$ to the applied on the robotic manipulator. The set point variables (or reference signal) are an angular position of the joints, and they are expressed by $\theta_1^{sp}(n)$, $\theta_2^{sp}(n)$ and $\theta_3^{sp}(n)$.

Figures 3.21–3.23 present the hardware validation results for various resolutions in terms of the number of bits of the fractional part, $N = \{12, 14, 16\}$ for discrete controlled variables $\theta_1(n)$, $\theta_2(n)$ and $\theta_3(n)$, respectively. The simulation trajectory was of 10 seconds and every 2 seconds was changing. Table 3.6 shows the angle trajectory changing for set point variables $\theta_1^{sp}(n)$, $\theta_2^{sp}(n)$ and $\theta_3^{sp}(n)$. Simulations used $t_s = 1 \times 10^{-5}$, Kp $= 2000$ and Ki $= 0.1$ for each i-th Fuzzy-PI-i hardware.

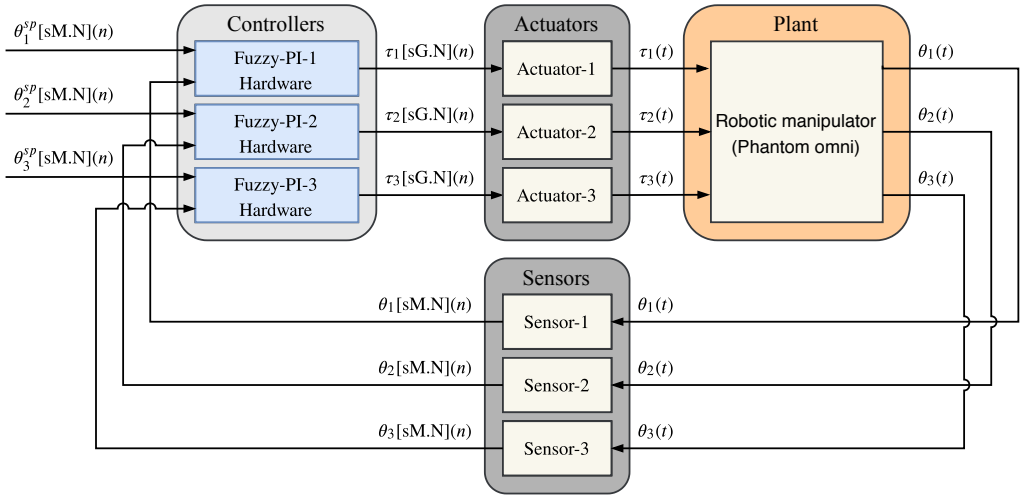

Figure 3.20: Simulated system used to validate the Fuzzy-PI hardware proposal. The plant is the 3DOF Phantom Omni robotic manipulator and there are three pieces of Fuzzy-PI hardware running in parallel.

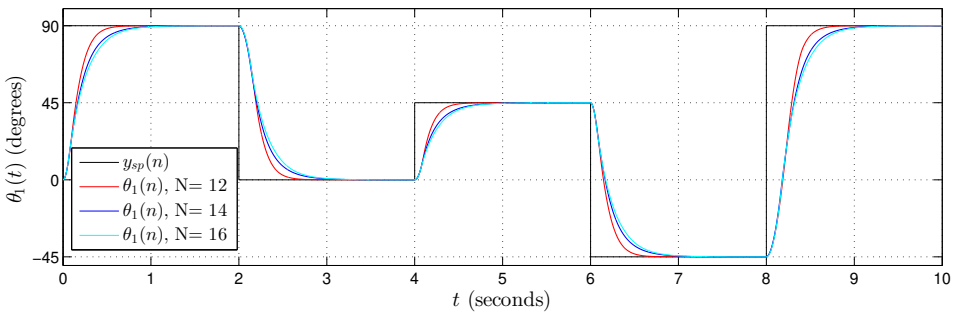

Figure 3.21: Validation results from the proposed Takagi–Sugeno Fuzzy-PI hardware. Simulation trajectory for $\theta_1(t)$ with $\theta_1(n)$ using $N = \{12, 14, 16\}$ bits in the fractional part.

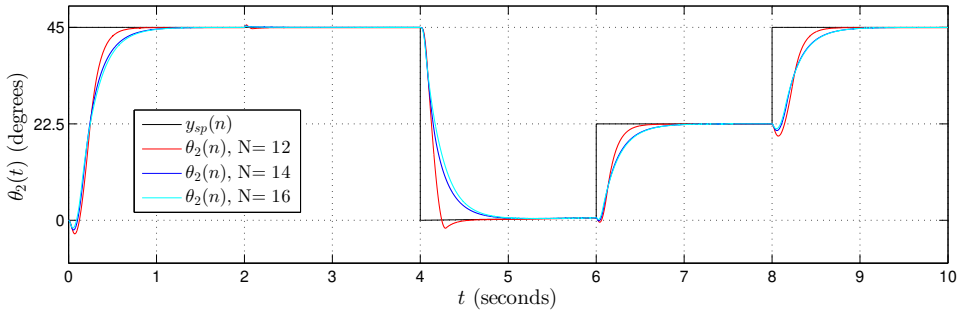

Figure 3.22: Validation results from the proposed Takagi–Sugeno Fuzzy-PI hardware. Simulation trajectory for $\theta_2(t)$ with $\theta_2(n)$ using $N = \{12, 14, 16\}$ bits in the fractional part.

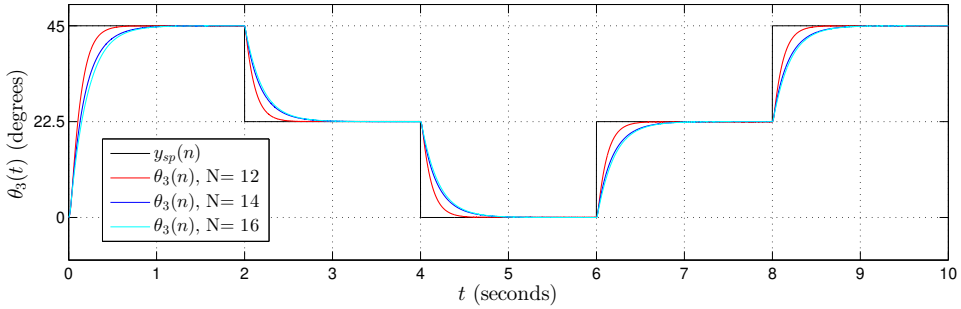

Figure 3.23: Validation results from the proposed Takagi–Sugeno Fuzzy-PI hardware. Simulation trajectory for $\theta_3(t)$ with $\theta_3(n)$ using $N = \{12, 14, 16\}$ bits in the fractional part.

Table 3.6: Angle trajectory changing for set point variables $\theta_1^{sp}(n)$, $\theta_2^{sp}(n)$ and $\theta_3^{sp}(n)$.

Set point	$0-2$ s	2 s -4 s	4 s -6 s	6 s -8 s	8 s -10 s
$\theta_1^{sp}(n)$ (Figure 3.21)	$90°$	$0°$	$45°$	$-45°$	$90°$
$\theta_2^{sp}(n)$ (Figure 3.22)	$45°$	$45°$	$0°$	$22.5°$	$45°$
$\theta_3^{sp}(n)$ (Figure 3.23)	$45°$	$22.5°$	$0°$	$22.5°$	$45°$

In the results presented in Figures 3.21–3.23, it is possible to observe that the controller followed the plant reference in all cases. Results also showed that the Takagi–Sugeno Fuzzy-PI hardware proposal has been following the reference even for a small amount of bits, that is, a low resolution.

3.7 Comparison with Other Works

3.7.1 Throughput Comparison

Table 3.7 shows a comparison with other works in the literature. Parameters like inference machine (IM) type (Takagi–Sugeno or Mamdani), number of inputs (NI), number of rules (NR), number of outputs (NO), number of bits (NB), throughput in Msps, R_s and Mflips (Mega fuzzy logic inference per second) are showed. In additional, Table 3.7 also shows the speedups (in Msps and Mflips) achieved of the TS-FIMM-OS, TS-FIMM-P, Fuzzy-PI controller with TS-FIMM-OS (Fuzzy-PI-OS) and with TS-FIMM-P (Fuzzy-PI-P) over the other works in the literature. The value in flips can be calculated as NR $\times R_s$.

In the work presented in (Sánchez-Solano et al. 2013), the results were obtained for several cases and, for one with two inputs, 35 rules, and one output (vehicle parking problem), the proposed hardware achieved a maximum clock about 66.251 MHz with 10 bits (Sánchez-Solano et al. 2010, Baturone et al. 2004). However, the FIM takes 10 clocks to complete the inference step; in other words, the hardware proposal in (Sánchez-Solano et al. 2013) achieves a throughput in Msps of about $\frac{66.251}{10} \approx 6.63$ Msps and in Mflips of about $6.63 \times 35 \approx 232.05$ Mflips. The speedup in Msps for the TS-FIMM-OS, TS-FIMM-P, Fuzzy-PI-OS, and Fuzzy-PI-P are $\frac{12.05\,\text{Msps}}{6.63\,\text{Msps}} \approx 1.82$, $\frac{17.63\,\text{Msps}}{6.63\,\text{Msps}} \approx 2.66$, $\frac{10.16\,\text{Msps}}{6.63\,\text{Msps}} \approx 1.53$, and $\frac{13.86\,\text{Msps}}{6.63\,\text{Msps}} \approx 2.09$, respectively. As the hardware proposal in this paper used 49 rules, the speedup in Mflips can be calculated as the throughput in Msps $\times \frac{49}{35}$ that is, the speedup for the TS-FIMM-OS, TS-FIMM-P, Fuzzy-PI-OS and Fuzzy-PI-P are $1.82 \times 1.4 \approx 2.55$, $2.66 \times 1.4 \approx 3.72$, $1.53 \times 1.4 \approx 2.14$, and $2.09 \times 1.4 \approx 2.93$, respectively.

The work presented in (Aguilar et al. 2014) proposes a Takagi–Sugeno fuzzy controller on FPGA with two inputs, six rules, and three outputs. The hardware achieved a throughput of about 1 Msps with 8 bits on the bus. With 8 bits, the speedup in Msps for the TS-FIMM-OS, TS-FIMM-P, Fuzzy-PI-OS, and Fuzzy-PI-P are $\frac{11.94\,\text{Msps}}{1\,\text{Msps}} \approx 11.94$, $\frac{17.55\,\text{Msps}}{1\,\text{Msps}} \approx 17.55$, $\frac{10.77\,\text{Msps}}{1\,\text{Msps}} \approx 10.77$, and $\frac{15.13\,\text{Msps}}{1\,\text{Msps}} \approx 15.13$, respectively. The speedup in Mflips is about $\frac{49}{6} \approx 8.16\times$ over the speedup in Msps.

In (Sun et al. 2015), a Mamdani fuzzy logic controller on FPGA was proposed. The hardware carries out a throughput of about 25 Mflips with two inputs, 49 rules, one output, and 16 bits. Using 16 bits, the speedup in Mflips for the TS-FIMM-OS, TS-FIMM-P, Fuzzy-PI-OS, and Fuzzy-PI-P are $\frac{11.28\times 49\,\text{Mflips}}{25\,\text{Mflips}} \approx 22.11$, $\frac{16.98\times 49\,\text{Mflips}}{25\,\text{Mflips}} \approx 33.28$, $\frac{9.59\times 49\,\text{Mflips}}{25\,\text{Mflips}} \approx 18.79$, and $\frac{13.41\times 49\,\text{Mflips}}{25\,\text{Mflips}} \approx 26.28$, respectively. As the number of rules is 49, the speedup in Msps is equal to Mflips.

The work presented in (Ontiveros-Robles et al. 2016) uses a Mamdani inference machine and the throughput in Mflips is about 48.23 Mflips. The hardware designed in (Ontiveros-Robles et al. 2016) operated with 8 bits, four inputs, nine rules, and one output. The speedup in Mflips, with 8 bits, for the TS-FIMM-OS, TS-FIMM-P, Fuzzy-PI-OS, and Fuzzy-PI-P are $\frac{11.94\times 49\,\text{Mflips}}{48.23\,\text{Mflips}} \approx 12.13$, $\frac{17.55\times 49\,\text{Mflips}}{48.23\,\text{Mflips}} \approx 17.83$, $\frac{10.77\times 49\,\text{Mflips}}{48.23\,\text{Mflips}} \approx 10.94$, and $\frac{13.41\times 49\,\text{Mflips}}{48.23\,\text{Mflips}} \approx 15.37$, respectively. The speedup in Msps is about $\frac{9}{49} \approx 0.18\times$ over the speedup in Mflips.

The hardware used in (Youssef et al. 2018) takes six clock cycles over 10 MHz (in

four states) to execute a M-IM with 16 bits. This is equivalent to a throughput of about $\frac{10\,\text{MHz}}{6} \approx 1.67\,\text{Msps}$. The scheme proposed in (Youssef et al. 2018) used two inputs, 25 rules, and one output. The speedup in Msps for the TS-FIMM-OS, TS-FIMM-P, Fuzzy-PI-OS, and Fuzzy-PI-P are $\frac{11.28\,\text{Msps}}{1.67\,\text{Msps}} \approx 6.75$, $\frac{16.98\,\text{Msps}}{1.67\,\text{Msps}} \approx 10.17$, $\frac{9.59\,\text{Msps}}{1.67\,\text{Msps}} \approx 5.74$, and $\frac{13.41\,\text{Msps}}{1.67\,\text{Msps}} \approx 8.03$, respectively. The speedup in Mflips is about $\frac{49}{25} \approx 1.96\times$ over the speedup in Msps.

The works presented in (de la Cruz-Alejo et al. 2019, Krim et al. 2019) show that a piece of hardware can achieve about 1 Msps. The work presented in (de la Cruz-Alejo et al. 2019) uses two inputs, 25 rules, one output, and 8 bits, and the designer presented in (Krim et al. 2019) was projected with three inputs, 42 rules, and one output. The speedup in Msps for the TS-FIMM-OS, TS-FIMM-P, Fuzzy-PI-OS, and Fuzzy-PI-P are equal to previously calculated values used in (Aguilar et al. 2014). The speedups in Mflips are about $\frac{49}{25} \approx 1.96\times$ and $\frac{49}{42} \approx 1.16\times$ over the speedup in Msps for works (de la Cruz-Alejo et al. 2019) and (Krim et al. 2019), respectively.

The hardware proposes in (Boncalo et al. 2019) achieved a throughput of about 1.56 Msps with three inputs, two outputs, and 24 bits. The speedup in Msps for the TS-FIMM-OS, TS-FIMM-P, Fuzzy-PI-OS, and Fuzzy-PI-P are $\frac{11.28\,\text{Msps}}{1.56\,\text{Msps}} \approx 7.23$, $\frac{16.98\,\text{Msps}}{1.56\,\text{Msps}} \approx 10.88$, $\frac{9.59\,\text{Msps}}{1.56\,\text{Msps}} \approx 6.15$, and $\frac{13.41\,\text{Msps}}{1.56\,\text{Msps}} \approx 8.59$, respectively. The fuzzy system proposed in (Boncalo et al. 2019) does not use linguistic fuzzy rules, and it cannot calculate the throughput in Mflips.

Table 3.7: Throughput comparison with other works.

References	IM	NI	NR	NO	NB	Msps	Mflips	This Work	Speedup	
									Msps	Mflips
(Sánchez-Solano et al. 2013) (2013)	TS-IM	2	35	1	10	≈ 6.63	≈ 232.05	TS-FIMM-OS	≈ 1.82×	≈ 2.55×
								TS-FIMM-P	≈ 2.66×	≈ 3.72×
								Fuzzy-PI-OS	≈ 1.53×	≈ 2.14×
								Fuzzy-PI-P	≈ 2.09×	≈ 2.93×
(Aguilar et al. 2014) (2014)	TS-IM	2	6	3	8	≈ 1.00	≈ 6.00	TS-FIMM-OS	≈ 11.94×	≈ 97.43×
								TS-FIMM-P	≈ 17.55×	≈ 143.20×
								Fuzzy-PI-OS	≈ 10.77×	≈ 87.88×
								Fuzzy-PI-P	≈ 15.13×	≈ 123.46×
(Sun et al. 2015) (2015)	M-IM	2	49	1	16	≈ 0.51	≈ 25.00	TS-FIMM-OS	≈ 22.11×	≈ 22.11×
								TS-FIMM-P	≈ 33.28×	≈ 33.28×
								Fuzzy-PI-OS	≈ 18.79×	≈ 18.79×
								Fuzzy-PI-P	≈ 26.28×	≈ 26.28×
(Ontiveros-Robles et al. 2016) (2016)	M-IM	4	9	1	8	≈ 5.36	≈ 48.23	TS-FIMM-OS	≈ 2.18×	≈ 12.13×
								TS-FIMM-P	≈ 3.20×	≈ 17.83×
								Fuzzy-PI-OS	≈ 1.97×	≈ 10.94×
								Fuzzy-PI-P	≈ 2.76×	≈ 15.37×
(Youssef et al. 2018) (2018)	M-IM	2	25	1	16	≈ 1.67	≈ 41.75	TS-FIMM-OS	≈ 6.75×	≈ 13.23×
								TS-FIMM-P	≈ 10.17×	≈ 19.93×
								Fuzzy-PI-OS	≈ 5.74×	≈ 11.25×
								Fuzzy-PI-P	≈ 8.03×	≈ 15.74×
(de la Cruz-Alejo et al. 2019) (2019)	M-IM	2	25	1	8	≈ 1.00	≈ 25.00	TS-FIMM-OS	≈ 11.94×	≈ 23.40×
								TS-FIMM-P	≈ 17.55×	≈ 34.40×
								Fuzzy-PI-OS	≈ 10.77×	≈ 21.11×
								Fuzzy-PI-P	≈ 15.13×	≈ 29.65×
(Krim et al. 2019) (2019)	M-IM	3	42	1	–	≈ 1.00	≈ 42.00	TS-FIMM-OS	≈ 11.94×	≈ 13.85×
								TS-FIMM-P	≈ 17.55×	≈ 20.36×
								Fuzzy-PI-OS	≈ 10.77×	≈ 12.49×
								Fuzzy-PI-P	≈ 15.13×	≈ 17.55×
(Boncalo et al. 2019) (2019)	TS-IM	3	–	2	24	≈ 1.56	–	TS-FIMM-OS	≈ 7.23×	–
								TS-FIMM-P	≈ 10.88×	–
								Fuzzy-PI-OS	≈ 6.15×	–
								Fuzzy-PI-P	≈ 8.59×	–

There are multiple differences between the devices used for comparison, starting with the number of bits in the LUTs (4-bits LUTs (Sánchez-Solano et al. 2013, Aguilar et al. 2014), 5-bits LUTs (Sun et al. 2015), and 6-bit LUTs (Ontiveros-Robles et al. 2016, Youssef et al. 2018, de la Cruz-Alejo et al. 2019, Krim et al. 2019, Boncalo et al. 2019)), board manufacturer (Altera (Aguilar et al. 2014, Sun et al. 2015) and Xilinx (Ontiveros-Robles et al. 2016, Youssef et al. 2018, de la Cruz-Alejo et al. 2019, Krim et al. 2019, Boncalo et al. 2019)), and families used (Spartan-3A (Aguilar et al. 2014), Cyclone-II (Sánchez-Solano et al. 2013), Arria-V GX (Sun et al. 2015), Spartan-6 (de la Cruz-Alejo et al. 2019, Krim et al. 2019), Virtex-5 (Boncalo et al. 2019), and Virtex-7 (Boncalo et al. 2019)). However, these differences have no significant influence on the throughput; the transmission rates of storage elements, such as LUTs, are in most cases of the same order of magnitude for devices using the same or similar technology. FPGAs have dedicated wires (called carry chains) between neighboring LUTs, and these circuits have a fast transmission rate that allows combining multiple LUTs (Teubner & Woods 2013, Cui et al. 2017). Therefore, differences in size of LUTs do not significantly affect throughput. Unlike the referenced works, for which most use a serial structure, in this work, we use a completely parallel approach. Thus, the design of the hardware architecture is primarily responsible for the resulting performance.

3.7.2 Hardware Occupation Comparison

Table 3.8 shows a comparison regarding the hardware occupation between the proposed hardware in this work and other literature works presented in Table 3.7. The second, third, fourth, and fifth columns show the type of FPGA, the number of logic cells (NLC), the number of multipliers (NMULT), and the number of bits in memory block RAMs (NBitsM), respectively, and the last three columns show the ratio of the hardware occupation between the proposal presented here, $N_{\text{hardware}}^{\text{work}}$, and literature works, $N_{\text{hardware}}^{\text{ref}}$, presented in Table 3.7. The ratio of the hardware occupation is described in Chapter 2, Section 2.8.2.

The work presented in (Sánchez-Solano et al. 2013) used a Spartan 3A DSP FPGA from Xilinx, and it has a hardware occupation of about 199 slices, four multipliers, and one block RAM. As this FPGA uses about 2.25 LC per slice, it used about 447 LC and it has $1,512$ K bits per block RAM. The scheme proposed in (Aguilar et al. 2014) used a Cyclone II EP2C35F672C6 FPGA from Intel, and it has a hardware occupation of about $1,622$ logic cells and 8.19 Kbits of memory. The EP2C35 FPGA has 105 block RAM and $4,096$ memory bits per block ($4,608$ bits per block including 512 parity bits).

In (Sun et al. 2015), the work assigns an Arria V GX 5AGXFB3H4F40C5NES FPGA from Intel and it has a hardware occupation of about 3248 ALMs and 6.592 Kbits of memory. The Arria V GX 5AGX has two combinational logic cells per ALM. The hardware proposed in (Ontiveros-Robles et al. 2016) employs a Spartan 6 FPGA from Xilinx, and it has a hardware occupation of about 544 LUTs and 32 multipliers. As this FPGA uses about 1.6 LC per LUT, it used about 447 LC.

The hardware presented in the manuscript (Youssef et al. 2018) utilizes a Spartan 6 FPGA from Xilinx, and it has a hardware occupation of about $1,802$ slices and five

multipliers. As this FPGA works with 6.34 LC per slice, it used about $11,425$ LC. The proposal described in (Krim et al. 2019) take advantage of Virtex 5 xc5vfx70t-3ff1136 FPGA from Xilinx, and it has a hardware occupation of about $8,195$ LUTs and 53 multipliers. As this FPGA uses about 1.6 LC per LUT, it used about $13,108$ LC. For 6-input LUT, they use the multiplier 1.6. The work presented in (Boncalo et al. 2019) used a Virtex 7 VX485T-2 FPGA from Xilinx, and it has a hardware occupation of about $1,948$ slices and 38 multipliers. As this FPGA uses about 6.4 LC per slice, it used about $12,468$ LC.

Regarding hardware utilization, the size in bits of the LUTs can have influence when comparing the NLCs in different FPGAs. Since the Virtex 6 (the FPGA board used in this work) has 6-bit LUTs, we can apply a relation factor $4/6$ to compare between 4-bit and 6-bit LUTs and $5/6$ for comparisons between 5-bit and 6-bit LUTs. In the case of 4-bit LUTs (the works presented in (Sánchez-Solano et al. 2013, Aguilar et al. 2014)), the NLC is reduced by $4/6$ and the hardware utilization ratio, $R_{\text{occupation}}$, increases by 34%. For 5-bit LUTs (the work presented in (Sun et al. 2015)), the NLC is reduced by $5/6$ and the $R_{\text{occupation}}$ increases by 17%.

Table 3.8: Hardware occupation comparison with other works.

References	FPGA	NLC	NMULT	NBitsM	This work	$R_{\text{occupation}}$ NLC	$R_{\text{occupation}}$ NMULT	$R_{\text{occupation}}$ NBitsM
(Sánchez-Solano et al. 2013) (2013)	Spartan 3A	447	4	1512 K	TS-FIMM-OS	$\approx 26.24\times$	$\approx 12.25\times$	$\approx 10^{-6}\times$
					TS-FIMM-P	$\approx 22.61\times$		
					Fuzzy-PI-OS	$\approx 26.24\times$		
					Fuzzy-PI-P	$\approx 22.61\times$		
(Aguilar et al. 2014) (2014)	Cyclone II	1,622	0	8.19 K	TS-FIMM-OS	$\approx 6.51\times$	$49\times$	$\approx 10^{-3}\times$
					TS-FIMM-P	$\approx 5.51\times$		
					Fuzzy-PI-OS	$\approx 6.74\times$		
					Fuzzy-PI-P	$\approx 5.75\times$		
(Sun et al. 2015) (2015)	Arria V GX	6,496	0	6.592 K	TS-FIMM-OS	$\approx 2.53\times$	$49\times$	$\approx 10^{-3}\times$
					TS-FIMM-P	$\approx 2.21\times$		
					Fuzzy-PI-OS	$\approx 2.61\times$		
					Fuzzy-PI-P	$\approx 2.29\times$		
(Ontiveros-Robles et al. 2016) (2016)	Spartan 6	871	32	0 K	TS-FIMM-OS	$\approx 12.13\times$	$\approx 1.53\times$	$1\times$
					TS-FIMM-P	$\approx 10.28\times$		
					Fuzzy-PI-OS	$\approx 12.56\times$		
					Fuzzy-PI-P	$\approx 10.71\times$		
(Youssef et al. 2018) (2018)	Spartan 6	11,425	5	0 K	TS-FIMM-OS	$\approx 1.44\times$	$\approx 9.8\times$	$1\times$
					TS-FIMM-P	$\approx 1.25\times$		
					Fuzzy-PI-OS	$\approx 1.48\times$		
					Fuzzy-PI-P	$\approx 1.30\times$		
(Krim et al. 2019) (2019)	Virtex 5	13,108	53	0 K	TS-FIMM-OS	$\approx 1.25\times$	$\approx 0.93\times$	$1\times$
					TS-FIMM-P	$\approx 1.09\times$		
					Fuzzy-PI-OS	$\approx 1.29\times$		
					Fuzzy-PI-P	$\approx 1.13\times$		
(Boncalo et al. 2019) (2019)	Virtex 7	12,468	38	0 K	TS-FIMM-OS	$\approx 1.32\times$	$\approx 1.29\times$	$1\times$
					TS-FIMM-P	$\approx 1.15\times$		
					Fuzzy-PI-OS	$\approx 1.36\times$		
					Fuzzy-PI-P	$\approx 1.19\times$		

3.7.3 Power Consumption Comparison

Table 3.9 shows the dynamic power saving regarding the dynamic power. Chapter 2, Section 2.8.3 presents the necessary equations to calculate the dynamic power consumption by the implementations proposed here. However, the Equation 2.28 is different. In this case N_g depends on the NLC and NMULT. Thus, the number of elements, N_g, was calculated as

$$N_g = \text{NLC} + \text{NMULT}. \tag{3.25}$$

where the N_g is the number of elements, NLC is the number logic cells. Differently from the literature, the hardware proposed here uses a fully parallelization layout, and it spends a one clock cycle per sample processing. In other words, the maximum clock frequency is equivalent to the throughput, $F_{\text{clk}}^{\text{work}} \equiv R_s$.

Table 3.9: Dynamic power comparison with other works.

References	FPGA	N_g^{ref}	$F_{\text{clk}}^{\text{ref}}$ (MHz)	This Work	N_g^{work}	$F_{\text{clk}}^{\text{work}}$ (MHz)	S_d
(Sánchez-Solano et al. 2013) (2013)	Spartan 3A	451	66.251	TS-FIMM-OS	11,779	6.63	$\approx 38.20\times$
				TS-FIMM-P	10,157		$\approx 44.30\times$
				Fuzzy-PI-OS	11,779		$\approx 38.20\times$
				Fuzzy-PI-P	10,157		$\approx 44.30\times$
(Sun et al. 2015) (2015)	Arria V GX	6,496	125	TS-FIMM-OS	16,453	0.51	$\approx 10^6\times$
				TS-FIMM-P	14,377		
				Fuzzy-PI-OS	17,001		
				Fuzzy-PI-P	14,926		
(Ontiveros-Robles et al. 2016) (2016)	Spartan 6	903	20	TS-FIMM-OS	6,598	5.36	$\approx 4.42\times$
				TS-FIMM-P	5,590		$\approx 5.22\times$
				Fuzzy-PI-OS	6,834		$\approx 4.27\times$
				Fuzzy-PI-P	5,826		$\approx 5.01\times$
(Youssef et al. 2018) (2018)	Spartan 6	11,430	10	TS-FIMM-OS	10,252	1.67	$\approx 149.16\times$
				TS-FIMM-P	8,955		$\approx 170.70\times$
				Fuzzy-PI-OS	10,595		$\approx 144.35\times$
				Fuzzy-PI-P	9,298		$\approx 164.42\times$
(Boncalo et al. 2019) (2019)	Virtex 7	12,506	150	TS-FIMM-OS	10,252	1.56	$\approx 10^5\times$
				TS-FIMM-P	8,955		
				Fuzzy-PI-OS	10,595		
				Fuzzy-PI-P	9,298		

With the exception of the Spartan-3A (presented in (Sánchez-Solano et al. 2013)), which uses 4-bit LUTs and the Arria-V GX (presented in (Sun et al. 2015)), which uses 5-bit LUTs, the other devices used for power analysis have 6-bit LUTs such as the Virtex-6. Thus, as indicated previously (see subsection 3.7.2), in the case of the Spartan-3A and the Arria-V GX, the NLC value is recalculated using a 6-bit LUT as reference. For the Spartan-3A, the NLC becomes $451 \times \frac{4}{6} \approx 301$, with a dynamic power saving of approximately $\approx 25\times$. For the Arria-V GX, the NLC becomes $6496 \times \frac{5}{6} \approx 5413$, with a dynamic power saving of approximately $\approx 10^6\times$. However, according to Equation 3.25, this reduction of NLCs will not have a significant impact on the dynamic power saving since it increases with frequency cube.

3.7.4 Analysis of the Comparisons

Results presented in Tables 3.7 and 3.9 demonstrate that the fully parallelization strategy adopted here can achieve significant speedups and power consumption reductions. On the other hand, the fully parallelization scheme can increase the hardware consumption, see Table 3.8.

The mean value of speedup was about $10.89\times$ in Msps and $30.89\times$ in Mflips (see Table 3.7) and this results are very expressive to big data and MMD applications (Poli 2015, Nasrollahzadeh et al. 2017, Yaqoob et al. 2016). High-throughput fuzzy controllers are also important for speed control systems such as tactile internet applications (Simsek et al. 2016*b*, Junior et al. 2019).

This manuscript proposal has LC resources with higher utilization than the literature proposals (Table 3.8). The mean value regarding NLC utilization was about $6.89\times$; in other words, the fuzzy hardware scheme proposed here has used $6.89\times$ more LC than the literature proposals. In the case of multipliers (NMULT), the mean value of the additional hardware was about $17.69\times$. Despite being large relative values, Tables 3.1–3.4 show that the fuzzy hardware proposals in this work expend no more than 7% of the FPGA resource. Another important aspect is the block RAM resource utilization (NBitsM). The fully parallel computing scheme proposed here does not spend clock time to access information in block RAM, and this can increase the throughput and decrease the power consumption (see (Sánchez-Solano et al. 2013), (Aguilar et al. 2014) and (Sun et al. 2015) in Tables 3.7–3.9).

The fully parallel designer allows for executing many operations per clock period, and this reduces the clock frequency operation and increases the throughput. Due to the nonlinear relationship with clock frequency operation (see Equation 2.27), this strategy permits a considerable reduction of the dynamic power consumption (see Table 3.9). The results presented in Table 3.9 show that the power saving can achieve values from 4 until 10^6 times, and these results are quite significant and enable the use of the proposed hardware here in several IoT applications.

3.8 Conclusions

This work aimed to develop a hardware dedicated to the fuzzy inference machine, the Takagi–Sugeno Fuzzy-PI controller. The developed hardware used a fully parallel implementation with fixed- and floating-point representations in distinct parts of the proposed scheme. All the details of the implementation were presented as well as the synthesis results and the bit-precision simulations. The synthesis was performed for several bit size resolutions and showed that the proposed hardware is viable for use in applications with critical processing time requirements. In order to characterize the proposed hardware, curves were generated, using the synthesis data obtained, to predict hardware consumption and throughput for all bit sizes. In addition, comparison results concerning throughput, hardware occupation, and power saving with other literature proposals were presented.

Chapter 4

Conclusion

The thesis proposed the use of dedicated hardware-based reconfigurable computing to reduce latency in control and prediction systems applied to the new Internet paradigm called the tactile internet, more specifically assisting in reducing latency of systems based on Tactile Internet with the use of hardware implementations. Initially, contextualization of the theme was developed, in which the generic model for tactile internet systems that introduced the work was illustrated. The structured model exposes details of the structured system, as well as the algorithms that make up a tactile system.

Two FPGA-based hardware proposals have been developed. The first approach proposes the implementation of linear and non-linear prediction techniques in reconfigurable computing. In this approach, it is demonstrated that prediction techniques developed in FPGA can be used to minimize the impacts caused by delays and loss of information. Several techniques have been implemented between linear and non-linear for various bit configurations. In comparison with other works, the non-linear techniques demonstrated that the full parallelization performed well with respect to speedup and a significant reduction in power consumption. The choice of not using the hyperbolic tangent as an activation function was an important factor in the consumption of RAM blocks, as well as for reducing power consumption, since the ReLU function, used here, was more computationally feasible. The throughput results obtained values between 14Msps and 24Msps between linear and non-linear techniques. These throughput values enable the use of these proposals in problems with critical time requirements such as tactile internet applications. In addition, consumption is reduced by up to $\approx 1400\times$.

In the second approach, a Takagi - Sugeno Fuzzy-PI intelligent control system was proposed, which was developed in FPGA. The implementation uses a fully parallel strategy associated with a hybrid bit format scheme (fixed-point and other floating-point). Results presented to demonstrate that the full parallelization strategy adopted here can achieve significant speedups and power consumption reductions. However, this approach has shown that it can increase the hardware area, but it has shown that the efficient use of resources such as block RAM (NBitsM) can increase system throughput, as well as decrease power consumption. The mean value of speedup was about $10.89\times$ in Msps and $30.89\times$ in Mflips. These results show that the proposed control system can be used in systems that require high-throughput for speed control systems such as tactile internet applications. It also showed that power saving can achieve values from 4 until 10^6 times, and these results are quite significant and enable the use of the proposed hardware here in

several IoT applications.

Thus, it is possible to conclude that the hardware presented in this thesis can be used in systems based on Tactile Internet since they comply with processing time requirements, massive data computing, and energy efficiency. The use of these approaches proposed here can bring benefits to the new technology in communications.

4.1 Future Works

For future work, we want to study new hardware designs that increase the performance of prediction and control techniques. In control feedback systems, test predictive control techniques and compare with the results shown here.

To study the behavior of prediction techniques implemented in hardware when subjected to a situation with high latency, signal loss, and channel noise. In this study, an index of efficiency of the prediction technique in tactile internet systems can be developed. This index would be based on how the technique deals with possible system problems while maintaining performance in speed and energy efficiency.

Propose the construction of other prediction models, such as probabilistic models, to outline a profile of techniques that can contribute to tactile systems. In this line, it is possible to create a management system for tactile internet systems. The system would be responsible for managing which prediction and/or control technique should be used in a given context. Thus, it would be possible to choose the best option in terms of speed and efficiency.

Bibliography

Ababei, Cristinel & Milad Ghorbani Moghaddam (2018), 'A survey of prediction and classification techniques in multicore processor systems', *IEEE Transactions on Parallel and Distributed Systems* **30**(5), 1184–1200.

Aguilar, A., M. Pérez, J. L. Camas, H. R. Hernández & C. Ríos (2014), Efficient design and implementation of a multivariate takagi-sugeno fuzzy controller on an fpga, *em* '2014 International Conference on Mechatronics, Electronics and Automotive Engineering', pp. 152–157.

Aijaz, A. (2016), Towards 5g-enabled tactile internet: Radio resource allocation for haptic communications, *em* '2016 IEEE Wireless Communications and Networking Conference', pp. 1–6.

Aijaz, Adnan, Mischa Dohler, A. Hamid Aghvami, Vasilis Friderikos & Magnus Frodigh (2015), 'Realizing the tactile internet: Haptic communications over next generation 5g cellular networks', *CoRR* **abs/1510.02826**.
URL: *http://arxiv.org/abs/1510.02826*

Akbatı, Onur, Hatice Didem Üzgün & Sirin Akkaya (2019), 'Hardware-in-the-loop simulation and implementation of a fuzzy logic controller with fpga: case study of a magnetic levitation system', *Transactions of the Institute of Measurement and Control* **41**(8), 2150–2159.
URL: *https://doi.org/10.1177/0142331218813425*

Al-Wais, S., S. A. Al-Samarraie, H. Abdi & S. Nahavandi (2016), Integral sliding mode controller for trajectory tracking of a phantom omni robot, *em* '2016 International Conference on Cybernetics, Robotics and Control (CRC)', pp. 6–12.

Ateya, Abdelhamied A, Mashael Khayyat, Ammar Muthanna & Andrey Koucheryavy (2019), Toward tactile internet, *em* '2019 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)', IEEE, pp. 1–7.

Bahoura, Mohammed (2016), 'Fpga implementation of blue whale calls classifier using high-level programming tool', *Electronics* **5**(1), 8.

Bahoura, Mohammed (2018), 'Fpga implementation of an automatic wheezing detection system', *Biomedical Signal Processing and Control* **46**, 76–85.

Banjanovic-Mehmedovic, L. & A. Husejnovic (2019), Fpga based hexapod robot navigation using arbitration of fuzzy logic controlled behaviors, *em* '2019 XXVII International Conference on Information, Communication and Automation Technologies (ICAT)', pp. 1–6.

Baturone, I., F. J. Moreno-Velo, S. Sanchez-Solano & A. Ollero (2004), 'Automatic design of fuzzy controllers for car-like autonomous robots', *IEEE Transactions on Fuzzy Systems* **12**(4), 447–465.

Bellemare-Rousseau, Simon, Gabriel Lachance, Shimwe-Dominique Niyonambaza, Élodie Boisselier, Mounir Bouakadoum & Amine Miled (2020), Fpga-based prediction system for neurotransmitter concentration measurement from spectrophotometry data, *em* '2020 18th IEEE International New Circuits and Systems Conference (NEWCAS)', IEEE, pp. 267–270.

Bhatia, Munish, Simranpreet Kaur, Sandeep K Sood & Veerawali Behal (2020), 'Internet of things-inspired healthcare system for urine-based diabetes prediction', *Artificial Intelligence in Medicine* **107**, 101913.

Bicakci, S. (2019), 'On the implementation of fuzzy vmc for an under actuated system', *IEEE Access* **7**, 163578–163588.

Blaiech, Ahmed Ghazi, Khaled Ben Khalifa, Carlos Valderrama, Marcelo A.C. Fernandes & Mohamed Hedi Bedoui (2019), 'A survey and taxonomy of fpga-based deep learning accelerators', *Journal of Systems Architecture* **98**, 331 – 345.
URL: *http://www.sciencedirect.com/science/article/pii/S1383762118304156*

Boncalo, O., A. Amaricai & Z. Lendek (2019), Configurable hardware accelerator architecture for a takagi-sugeno fuzzy controller, *em* '2019 22nd Euromicro Conference on Digital System Design (DSD)', pp. 96–101.

Bosque, G., I. del Campo & J. Echanobe (2014), 'Fuzzy systems, neural networks and neuro-fuzzy systems: A vision on their hardware implementation and platforms over two decades', *Engineering Applications of Artificial Intelligence* **32**, 283 – 331.
URL: *http://www.sciencedirect.com/science/article/pii/S0952197614000384*

Brandi, Fernanda & Eckehard Steinbach (2013), Prediction techniques for haptic communication and their vulnerability to packet losses, *em* 'HAVE 2013 - 2013 IEEE International Symposium on Haptic Audio-Visual Environments and Games, Proceedings', pp. 63–68.

Briscoe, Bob, Anna Brunstrom, Andreas Petlund, David Hayes, David Ros, Jyh Tsang, Stein Gjessing, Gorry Fairhurst, Carsten Griwodz & Michael Welzl (2014), 'Reducing internet latency: A survey of techniques and their merits', *IEEE Communications Surveys & Tutorials* **18**(3), 2149–2196.

Carpeño, Antonio, Mariano Ruiz, César González, Jesús Vega, Sebastián Dormido-Canto, Sergio Esquembri & Enrique Bernal (2019), 'Opencl implementation of an adaptive

disruption predictor based on a probabilistic venn classifier', *IEEE Transactions on Nuclear Science* **66**(7), 1007–1013.

Chowdhury, S. R. & H. Saha (2008), 'A high-performance fpga-based fuzzy processor architecture for medical diagnosis', *IEEE Micro* **28**(5), 38–52.

Coutinho, M. G. F., M. F. Torquato & M. A. C. Fernandes (2019*a*), 'Deep neural network hardware implementation based on stacked sparse autoencoder', *IEEE Access* **7**, 40674–40694.

Coutinho, Maria GF, Matheus F Torquato & Marcelo AC Fernandes (2019*b*), 'Deep neural network hardware implementation based on stacked sparse autoencoder', *IEEE Access* **7**, 40674–40694.

Cui, Dapeng & David Curry (2005), 'Prediction in marketing using the support vector machine', *Marketing Science* **24**(4), 595–615.

Cui, Ke, Xiangyu Li & Rihong Zhu (2017), 'A high-resolution programmable vernier delay generator based on carry chains in fpga', *Review of Scientific Instruments* **88**(6), 064703.

Da Costa, A. L. X., C. A. D. Silva, M. F. Torquato & M. A. C. Fernandes (2019*a*), 'Parallel implementation of particle swarm optimization on fpga', *IEEE Transactions on Circuits and Systems II: Express Briefs* **66**(11), 1875–1879.

Da Costa, Alexandre LX, Caroline AD Silva, Matheus F Torquato & Marcelo AC Fernandes (2019*b*), 'Parallel implementation of particle swarm optimization on fpga', *IEEE Transactions on Circuits and Systems II: Express Briefs* **66**(11), 1875–1879.

Da Silva, Lucileide MD, Matheus F Torquato & Marcelo AC Fernandes (2018), 'Parallel implementation of reinforcement learning q-learning technique for fpga', *IEEE Access* **7**, 2782–2798.

de la Cruz-Alejo, J., R. Antonio-Méndez & M. Salazar-Pereyra (2019), 'Fuzzy logic control on fpga for two axes solar tracking', *Neural Computing and Applications* **31**(7), 2469–2483.
URL: *https://doi.org/10.1007/s00521-017-3207-1*

De Souza, Alisson CD & Marcelo AC Fernandes (2014*a*), 'Parallel fixed point implementation of a radial basis function network in an fpga', *Sensors* **14**(10), 18223–18243.

de Souza, Alisson CD & Marcelo AC Fernandes (2014*b*), 'Parallel fixed point implementation of a radial basis function network in an fpga', *Sensors* **14**(10), 18223–18243.

de Souza, Alisson CD & Marcelo AC Fernandes (2014*c*), 'Parallel fixed point implementation of a radial basis function network in an fpga', *Sensors* **14**(10), 18223–18243.

Del Re, Enrico, Simone Morosi, Lorenzo Mucchi, Luca Simone Ronga & Sara Jayousi (2016), 'Future wireless systems for human bond communications', *Wireless Personal Communications* **88**(1), 39–52.

Deliparaschos, K. M., F. I. Nenedakis & S. G. Tzafestas (2006), 'Design and implementation of a fast digital fuzzy logic controller using fpga technology', *Journal of Intelligent and Robotic Systems* **45**(1), 77–96.
URL: *https://doi.org/10.1007/s10846-005-9016-2*

Dohler, M. (2015), The tactile internet iot, 5g and cloud on steroids, *em* '5G Radio Technology Seminar. Exploring Technical Challenges in the Emerging 5G Ecosystem', pp. 1–16.

Dorfling, Tinus, Hendrik du Toit Mouton, Tobias Geyer & Petros Karamanakos (2019), 'Long-horizon finite-control-set model predictive control with nonrecursive sphere decoding on an fpga', *IEEE Transactions on Power Electronics* **35**(7), 7520–7531.

Fernandes, Marcelo A.C. (2016), 'Fuzzy controller applied to electric vehicles with continuously variable transmission', *Neurocomputing* **214**, 684 – 691.

Fettweis, Gerhard P. (2013), 5G - What Will it Be : The Tactile Internet at TU Dresden, *em* 'Slides from ICC 2013 Budapest'.

Fettweis, G.P. (2014), 'The tactile internet: Applications and challenges', *Vehicular Technology Magazine, IEEE* **9**(1), 64–70.

Gaikwad, Nikhil B, Varun Tiwari, Avinash Keskar & NC Shivaprakash (2019), 'Efficient fpga implementation of multilayer perceptron for real-time human activity classification', *IEEE Access* **7**, 26696–26706.

Galvan, Stefano, Debora Botturi & Paolo Fiorini (2006), Fpga-based controller for haptic devices, *em* '2006 IEEE/RSJ International Conference on Intelligent Robots and Systems', IEEE, pp. 971–976.

Gordini, Niccolò & Valerio Veglio (2017), 'Customers churn prediction and marketing retention strategies. an application of support vector machines based on the auc parameter-selection technique in b2b e-commerce industry', *Industrial Marketing Management* **62**, 100–107.

Gupta, Vibhuti & Rattikorn Hewett (2020), 'Real-time tweet analytics using hybrid hashtags on twitter big data streams', *Information* **11**(7), 341.

Hartley, E.N. & Jm Maciejowski (2013), Predictive control for spacecraft rendezvous in an elliptical orbit using an FPGA, *em* 'European Control Conference (ECC)', pp. 1359–1364.
URL: *http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6669196*

Hassan, L. H., M. Moghavvemi, H. A. F. Almurib & K. M. Muttaqi (2016), Damping of low-frequency oscillations using takagi-sugeno fuzzy stabilizer in real-time, *em* '2016 IEEE Industry Applications Society Annual Meeting', pp. 1–7.

Haykin, Simon (1998), *Neural Networks: A Comprehensive Foundation (2nd Edition)*, 2ª edição, Prentice Hall.

Holland, O., S. Wong, V. Friderikos, Z. Qin & Y. Gao (2016), Virtualized sub-ghz transmission paired with mobile access for the tactile internet, *em* '2016 23rd International Conference on Telecommunications (ICT)', pp. 1–5.

Huang, Hsu-Chih, Chin-Wang Tao, Chen-Chia Chuang & Jing-Jun Xu (2019), 'Fpga-based mechatronic design and real-time fuzzy control with computational intelligence optimization for omni-mecanum-wheeled autonomous vehicles', *Electronics* **8**(11).
URL: *https://www.mdpi.com/2079-9292/8/11/1328*

Jenkal, Wissam, Rachid Latif, Abdelhafid Elouardi & Safa Mejhoudi (2019), Fpga implementation of the real-time adtf process using the intel-altera de1 board for ecg signal denoising, *em* '2019 4th World Conference on Complex Systems (WCCS)', IEEE, pp. 1–6.

Junior, José CVS, Matheus F Torquato, Toktam Mahmoodi, Mischa Dohler & Marcelo AC Fernandes (2020), 'Reconfigurable computing applied to latency reduction for the tactile internet', *arXiv preprint arXiv:2003.12463* .

Junior, José C. V. S., Matheus F. Torquato, Daniel H. Noronha, Sérgio N. Silva & Marcelo A. C. Fernandes (2019), 'Proposal of the tactile glove device', *Sensors* **19**(22).
URL: *https://www.mdpi.com/1424-8220/19/22/5029*

Júnior, E. I., L. Manuel Garcés Socarrás & T. C. Pimenta (2018), Design and low-cost fpga implementation of the fuzzy decision system, *em* '2018 30th International Conference on Microelectronics (ICM)', pp. 291–294.

Kamakura, Wagner A, Michel Wedel, Fernando De Rosa & Jose Afonso Mazzon (2003), 'Cross-selling through database marketing: a mixed data factor analyzer for data augmentation and prediction', *International Journal of Research in marketing* **20**(1), 45–65.

Karvelis, Petros, Theofanis-Aristofanis Michail, Daniele Mazzei, Stefanos Petsios, Andrea Bau, Gabriele Montelisciani & Chrysostomos Stylios (2018), Adopting and embedding machine learning algorithms in microcontroller for weather prediction, *em* '2018 International Conference on Intelligent Systems (IS)', IEEE, pp. 474–478.

Khati, H., R. Mellah & H. Talem (2019), Neuro-fuzzy control of a position-position teleoperation system using fpga, *em* '2019 24th International Conference on Methods and Models in Automation and Robotics (MMAR)', pp. 64–69.

Krim, Saber, Soufien Gdaim, Abdellatif Mtibaa & Mohamed Faouzi Mimouni (2019), 'Contribution of the fpgas for complex control algorithms: Sensorless dtfc with an ekf of an induction motor', *International Journal of Automation and Computing* **16**(2), 226–237.
URL: *https://doi.org/10.1007/s11633-016-1017-z*

Lavín-Delgado, JE, JE Solís-Pérez, JF Gómez-Aguilar & RF Escobar-Jiménez (2020), 'Trajectory tracking control based on non-singular fractional derivatives for the puma 560 robot arm', *Multibody System Dynamics* **50**(3), 259–303.

Liviu, T. (2018), Fpga implementation of a fuzzy rule based contrast enhancement system for real time applications, *em* '2018 22nd International Conference on System Theory, Control and Computing (ICSTCC)', pp. 117–122.

Loan, S. A. & A. M. Murshid (2013), A novel vlsi architecture of a multi membership function based max-min calculator circuit, *em* '2013 International Conference on Advanced Electronic Systems (ICAES)', pp. 74–78.

Loan, Sajad A., Asim M. Murshid, Shuja A. Abbasi & Abdul Rahman M. Alamoud (2013), 'A novel vlsi architecture for a fuzzy inference processor using gaussian-shaped membership function', *J. Intell. Fuzzy Syst.* **24**(1), 5–19.

Lopes, Felipe F, João Canas Ferreira & Marcelo AC Fernandes (2019*a*), 'Parallel implementation on fpga of support vector machines using stochastic gradient descent', *Electronics* **8**(6), 631.

Lopes, Felipe F., João Canas Ferreira & Marcelo A. C. Fernandes (2019*b*), 'Parallel implementation on fpga of support vector machines using stochastic gradient descent', *Electronics* **8**(6).
URL: *https://www.mdpi.com/2079-9292/8/6/631*

Maier, Martin, Mahfuzulhoq Chowdhury, Bhaskar Prasad Rimal & Dung Pham Van (2016), 'The tactile internet: vision, recent progress, and open challenges', *IEEE Communications Magazine* **54**(5), 138–145.

MATLAB (2012), *Matlab Fuzzy Logic Toolbox User's Guide - R2016a*, The MathWorks Inc., Natick, Massachusetts.

McCool, Michael, Arch D. Robison & James Reinders (2012), Chapter 2 - background, *em* M.McCool, A. D.Robison & J.Reinders, eds., 'Structured Parallel Programming', Morgan Kaufmann, Boston, pp. 39 – 75.
URL: *http://www.sciencedirect.com/science/article/pii/B9780124159938000025*

Montgomery, Douglas C, Elizabeth A Peck & G Geoffrey Vining (2012), *Introduction to linear regression analysis*, Vol. 821, John Wiley & Sons.

Nasrollahzadeh, Ali, Ghader Karimian & Amir Mehrafsa (2017), 'Implementation of neuro-fuzzy system with modified high performance genetic algorithm on embedded systems', *Applied Soft Computing* .
URL: *http://www.sciencedirect.com/science/article/pii/S156849461730409X*

Niculescu, Silviu-Iulian, Wim Michiels, Keqin Gu & Chaouki T. Abdallah (2010), *Delay Effects on Output Feedback Control of Dynamical Systems*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 63–84.

Noronha, Daniel H, Matheus F Torquato & Marcelo AC Fernandes (2019*a*), 'A parallel implementation of sequential minimal optimization on fpga', *Microprocessors and Microsystems* **69**, 138–151.

Noronha, Daniel H., Matheus F. Torquato & Marcelo A.C. Fernandes (2019*b*), 'A parallel implementation of sequential minimal optimization on fpga', *Microprocessors and Microsystems* **69**, 138 – 151.
URL: *http://www.sciencedirect.com/science/article/pii/S0141933118303879*

Oballe-Peinado, Oscar, José Antonio Hidalgo-López, Julián Castellanos-Ramos, José Antonio Sánchez-Durán, Rafael Navas-Gonzalez, Jaime Herran & Fernando Vidal-Verdú (2017), 'Fpga-based tactile sensor suite electronics for real-time embedded processing', *IEEE Transactions on industrial electronics* **64**(12), 9657–9665.

Ontiveros-Robles, E., J. L. Gonzalez-Vazquez, J. R. Castro & O. Castillo (2016), A hardware architecture for real-time edge detection based on interval type-2 fuzzy logic, *em* '2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)', pp. 804–810.

Oviedo, J.J.E., J.P.L. Vandewalle & V. Wertz (2004), *Fuzzy Logic, Identification and Predictive Control*, Advances in Industrial Control, Springer London.

O'Malley, Marcia K, Kevin S Sevcik & Emilie Kopp (2009), 'Improved haptic fidelity via reduced sampling period with an fpga-based real-time hardware platform', *Journal of computing and information science in engineering* **9**(1).

Pilz, J., M. Mehlhose, T. Wirth, D. Wieruch, B. Holfeld & T. Haustein (2016), A tactile internet demonstration: 1ms ultra low delay for wireless communications towards 5g, *em* '2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)', pp. 862–863.

Poli, Venkata Subba Reddy (2015), Fuzzy data mining and web intelligence, *em* 'Fuzzy Theory and Its Applications (iFUZZY), 2015 International Conference on', IEEE, pp. 74–79.

Prado, R. N. A., J. D. Melo, J. A. N. Oliveira & A. D. Dória Neto (2012), Fpga based implementation of a fuzzy neural network modular architecture for embedded systems, *em* 'The 2012 International Joint Conference on Neural Networks (IJCNN)', pp. 1–7.

Rebello, Rohan F & S Sriram (2004), 'Fpga-based high-performance computing for haptic simulation in a virtual environment', *University Chennai, India* .

Ruan, Lihua & Elaine Wong (2018), Machine intelligence in allocating bandwidth to achieve low-latency performance, *em* '2018 International Conference on Optical Network Design and Modeling (ONDM)', IEEE.

Ruan, Lihua, Sourav Mondal & Elaine Wong (2018), Machine learning based bandwidth prediction in tactile heterogeneous access networks, *em* 'IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)', IEEE.

Rumelhart, D. E., G. E. Hinton & R. J. Williams (1986), *Learning internal representations by error propagation*, MIT Press, Cambridge, MA, USA, pp. 318–362.

Sakr, Nizar, Nicolas D. Georganas & Jiying Zhao (2011), 'Human perception-based data reduction for haptic communication in Six-DoF telepresence systems', *IEEE Transactions on Instrumentation and Measurement* **60**(11), 3534–3546.

Sansanayuth, T., I. Nilkhamhang & K. Tungpimolrat (2012), Teleoperation with inverse dynamics control for phantom omni haptic device, *em* '2012 Proceedings of SICE Annual Conference (SICE)', pp. 2121–2126.

Shah, Dev, Haruna Isah & Farhana Zulkernine (2019), 'Stock market analysis: A review and taxonomy of prediction techniques', *International Journal of Financial Studies* **7**(2), 26.

Silva, A. J., O. A. D. Ramirez, V. P. Vega & J. P. O. Oliver (2009*a*), Phantom omni haptic device: Kinematic and manipulability, *em* '2009 Electronics, Robotics and Automotive Mechanics Conference (CERMA)', pp. 193–198.

Silva, Alejandro Jarillo, Omar A Domínguez Ramirez, Vicente Parra Vega & Jesus P Ordaz Oliver (2009*b*), Phantom omni haptic device: Kinematic and manipulability, *em* 'Electronics, Robotics and Automotive Mechanics Conference, 2009. CERMA'09.', IEEE, pp. 193–198.

Silva, L. M. D. Da, M. F. Torquato & M. A. C. Fernandes (2019*a*), 'Parallel implementation of reinforcement learning q-learning technique for fpga', *IEEE Access* **7**, 2782–2798.

Silva, Sérgio N., Matheus F. Torquato & Marcelo A. C. Fernandes (2019*b*), 'Comparison of binary and fuzzy logic in feedback control of dynamic systems', *International Journal of Dynamics and Control* **7**(3), 1056–1064.

Simsek, M., A. Aijaz, M. Dohler, J. Sachs & G. Fettweis (2016*a*), The 5g-enabled tactile internet: Applications, requirements, and architecture, *em* '2016 IEEE Wireless Communications and Networking Conference', pp. 1–6.

Simsek, M., A. Aijaz, M. Dohler, J. Sachs & G. Fettweis (2016*b*), The 5g-enabled tactile internet: Applications, requirements, and architecture, *em* '2016 IEEE Wireless Communications and Networking Conference', pp. 1–6.

Song, G., S. Guo & Q. Wang (2006), A tele-operation system based on haptic feedback, *em* '2006 IEEE International Conference on Information Acquisition', pp. 1127–1131.

Srinivas, K, B Kavihta Rani & A Govrdhan (2010), 'Applications of data mining techniques in healthcare and prediction of heart attacks', *International Journal on Computer Science and Engineering (IJCSE)* **2**(02), 250–255.

Sun, Yize, Shiqing Tang, Zhuo Meng, Yiman Zhao & Yunhu Yang (2015), 'A scalable accuracy fuzzy logic controller on {FPGA}', *Expert Systems with Applications* **42**(19), 6658 – 6673.

Sánchez-Solano, S., E. del Toro, M. Brox, I. Baturone & Á. Barriga (2010), A design environment for synthesis of embedded fuzzy controllers on fpgas, *em* 'International Conference on Fuzzy Systems', pp. 1–8.

Sánchez-Solano, S., M. Brox, E. del Toro, P. Brox & I. Baturone (2013), 'Model-based design methodology for rapid development of fuzzy controllers on fpgas', *IEEE Transactions on Industrial Informatics* **9**(3), 1361–1370.

Tanaka, Hiroyuki, Kouhei Ohnishi, Hiroaki Nishi, Toshikazu Kawai, Yasuhide Morikawa, Soji Ozawa & Toshiharu Furukawa (2008), 'Implementation of bilateral control system based on acceleration control using fpga for multi-dof haptic endoscopic surgery robot', *IEEE Transactions on Industrial Electronics* **56**(3), 618–627.

Tchendjou, Ghislain Takam, Emmanuel Simeu & Rshdee Alhakim (2018), 'Fuzzy logic based objective image quality assessment with fpga implementation', *Journal of Systems Architecture* **82**, 24 – 36.
URL: *http://www.sciencedirect.com/science/article/pii/S1383762116302892*

Teubner, Jens & Louis Woods (2013), *Data processing on FPGAs*, Vol. 5, Morgan & Claypool Publishers.

Thakur, Prabhat, Alok Kumar, Shweta Pandit, Ghanshyam Singh & SN Satashia (2018), 'Performance analysis of high-traffic cognitive radio communication system using hybrid spectrum access, prediction and monitoring techniques', *Wireless Networks* **24**(6), 2005–2015.

Titinchi, A. A. & N. Halasa (2019), Fpga implementation of simplified fuzzy lru replacement algorithm, *em* '2019 16th International Multi-Conference on Systems, Signals Devices (SSD)', pp. 657–662.

Torquato, Matheus F. & Marcelo A. C. Fernandes (2019), 'High-performance parallel implementation of genetic algorithm on fpga', *Circuits, Systems, and Signal Processing* **38**(9), 4014–4039.
URL: *https://doi.org/10.1007/s00034-019-01037-w*

Tu, Wencong, Guangzhao Luo, Zhe Chen, Chunqiang Liu & Longran Cui (2019), 'Fpga implementation of predictive cascaded speed and current control of pmsm drives with two-time-scale optimization', *IEEE Transactions on Industrial Informatics* **15**(9), 5276–5288.

Van Den Berg, Daniël, Rebecca Glans, Dorian De Koning, Fernando A Kuipers, Jochem Lugtenburg, Kurian Polachan, Prabhakar T Venkata, Chandramani Singh, Belma Turkovic & Bryan Van Wijk (2017), 'Challenges in haptic communications over the tactile internet', *IEEE Access* **5**, 23502–23518.

Watch, Itu-t Technology & Report August (2014), 'The Tactile Internet: ITU-T Technology Watch Report', (August), 1–18.

Wei, Xin, Qi Duan & Liang Zhou (2019), 'A qoe-driven tactile internet architecture for smart city', *IEEE Network* **34**(1), 130–136.

Widiasari, Indrastanti R, Lukito Edi Nugroho et al. (2017), Deep learning multilayer perceptron (mlp) for flood prediction model using wireless sensor network based hydrology time series data mining, *em* '2017 International Conference on Innovative and Creative Information Technology (ICITech)', IEEE, pp. 1–5.

Wienbrandt, Lars, Jan Christian Kässens, Matthias Hübenthal & David Ellinghaus (2019), '1000× faster than plink: Combined fpga and gpu accelerators for logistic regression-based detection of epistasis', *Journal of Computational Science* **30**, 183–193.

Wong, Elaine, Maluge Pubudini Imali Dias & Lihua Ruan (2017), 'Predictive resource allocation for tactile internet capable passive optical lans', *Journal of Lightwave Technology* **35**(13), 2629–2641.

Xie, Hai-Long, Qing-Hui Wang, Jian-Long Ni & Jing-Rong Li (2020), 'A gpu-based prediction and simulation method of grinding surface topography for belt grinding process', *The International Journal of Advanced Manufacturing Technology* **106**(11), 5175–5186.

XILINX Virtex-6 Data Sheet (n.d.), `https://www.xilinx.com/support/documentation/data_sheets/ds152.pdf`.

XILINX Virtex-6 User Guide (n.d.), `https://www.xilinx.com/support/documentation/user_guides/ug361.pdf`.

Yaqoob, Ibrar, Ibrahim Abaker Targio Hashem, Abdullah Gani, Salimah Mokhtar, Ejaz Ahmed, Nor Badrul Anuar & Athanasios V. Vasilakos (2016), 'Big data: From beginning to future', *International Journal of Information Management* **36**(6), 1231 – 1247.
URL: *http://www.sciencedirect.com/science/article/pii/S0268401216304753*

Youssef, Ayman, Mohammed El Telbany & Abdelhalim Zekry (2018), 'Reconfigurable generic fpga implementation of fuzzy logic controller for mppt of pv systems', *Renewable and Sustainable Energy Reviews* **82**, 1313 – 1319.
URL: *http://www.sciencedirect.com/science/article/pii/S136403211731345X*

Yu, Qi, Chao Wang, Xiang Ma, Xi Li & Xuehai Zhou (2015), A deep learning prediction process accelerator based FPGA, *em* 'Proceedings - 2015 IEEE/ACM 15th International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2015', número 500, pp. 1159–1162.

Zavala, A. H. & O. C. Nieto (2012), 'Fuzzy hardware: A retrospective and analysis', *IEEE Transactions on Fuzzy Systems* **20**(4), 623–635.

Zhai, Xiaojun, Amine Ait Si Ali, Abbes Amira & Faycal Bensaali (2016), 'Mlp neural network based gas classification system on zynq soc', *IEEE Access* **4**, 8138–8146.

Zhang, Chen, Peng Li, Guangyu Sun, Yijin Guan, Bingjun Xiao & Jason Cong (2015), Optimizing fpga-based accelerator design for deep convolutional neural networks, *em* 'Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays', ACM, pp. 161–170.

Zhao, Jun, Xiaoliang Zhu, Wei Wang & Ying Liu (2013), 'Extended kalman filter-based elman networks for industrial time series prediction with gpu acceleration', *Neurocomputing* **118**, 215–224.