sensors

MDPI

Article

An FPGA-Based Neuro-Fuzzy Sensor for Personalized Driving Assistance

Óscar Mata-Carballeira *, Jon Gutiérrez-Zaballa, Inés del Campo and Victoria Martínez

Department of Electricity and Electronics, Faculty of Science and Technology, University of the Basque Country UPV/EHU, 48940 Leioa, Spain; jongutierrez@bizkaia.eu (J.G.-Z.); ines.delcampo@ehu.eus (I.d.C.); victoria.martinez@ehu.eus (V.M.)

* Correspondence: oscar.mata@ehu.eus; Tel.: +34-94-601-3238

Received: 30 June 2019; Accepted: 15 September 2019; Published: 17 September 2019

Abstract: Advanced driving-assistance systems (ADAS) are intended to automatize driver tasks, as well as improve driving and vehicle safety. This work proposes an intelligent neuro-fuzzy sensor for driving style (DS) recognition, suitable for ADAS enhancement. The development of the driving style intelligent sensor uses naturalistic driving data from the SHRP2 study, which includes data from a CAN bus, inertial measurement unit, and front radar. The system has been successfully implemented using a field-programmable gate array (FPGA) device of the Xilinx Zynq programmable system-on-chip (PSoC). It can mimic the typical timing parameters of a group of drivers as well as tune these typical parameters to model individual DSs. The neuro-fuzzy intelligent sensor provides high-speed real-time active ADAS implementation and is able to personalize its behavior into safe margins without driver intervention. In particular, the personalization procedure of the time headway (THW) parameter for an ACC in steady car following was developed, achieving a performance of 0.53 microseconds. This performance fulfilled the requirements of cutting-edge active ADAS specifications.

Keywords: advanced driving assistance systems (ADAS); safety and comfort; driving style; unsupervised clustering; k-means; adaptive neuro-fuzzy inference system (ANFIS); field-programmable gate array (FPGA); programmable system-on-chip (PSoC)

1. Introduction

Currently, manufactured cars rely on a myriad of sensors to measure many internal and external variables that could influence the handling behavior of the automobile, as well as additional parameters, such as visibility and occupant comfort. Depending on their sophistication level, sensors can be classified ranging from simple sensors that directly measure single physical parameters (e.g., ambient light sensors and temperature sensors) to complex intelligent sensors, which determine parameters of the surrounding environment through wide spectrum signals (e.g., radio frequency/radar and light/video); besides measuring, they perform data processing and are enabled to carry out actuations. Whereas intelligent sensors make use of data of a different nature underneath, in which complex and nonlinear behaviors are codified; data-mining techniques used jointly with machine learning (ML) algorithms have shown adequate performance for modeling this hidden information. As intelligent sensors often rely on complex sensors and sensor fusion techniques, the data processing power they need can only be provided by high-performance computational platforms such as microprocessors, graphics-processing units (GPUs), or field-programmable gate arrays (FPGAs). In particular, FPGA-based implementations stand out due to the extremely high operational frequencies and low power consumption they can achieve, even for complex, multilayered algorithms [1]. In the context of the automotive field, intelligent sensors are key components of current assistance systems.

A long time has passed since automobiles were simple mechanical systems. With the development of the first driving-assistance systems (DAS) in the 1980s, cars introduced electronics in daily driving, turning them into complex mechatronic systems [2]. Thus, they have been gradually fitted with several simple electronic sensors, such as angular speed sensors, torque sensors, accelerometers, and gyroscopes, which together with mechatronic drives have derived safety systems such as the anti-lock braking system (ABS) [3], traction-control system (TCS) [4], and electronic stability program (ESP) [5]. These systems, before being marketed as optional equipment in top tier models, have drastically improved automotive safety. This improvement led to the obligation of fitting these systems in every new car in the United States of America (USA) since 2011 [6] and in the European Union (EU) since 2014 [7].

In the field of driver comfort, simple speed sensors play an important role in conventional feedback systems, such as cruise control (CC) solutions that relieve drivers from the mental load of keeping speed in a steady state during long rides in highways and motorways. However, despite CC systems contributing to reducing drivers' fatigue during long trips, they may bring about dangerous situations, especially rear-end collisions [8]. These situations occur because conventional CC systems are not aware of the distance from the preceding vehicle, that is, they only take into account the internal variables of the car. To solve these problems, external world data-based systems, such as adaptive cruise control (ACC), arose as an alternative to provide enhanced safety and comfort functionalities. ACC relies on real-time measurements of the distance to the preceding vehicle to keep the time gap and avoid rear-end collisions, thus increasing road safety [9]. Systems like ACC are based on complex intelligent sensors that measure external parameters through high-bandwidth signals such as radar, LIDAR, or video. These kinds of enhanced systems are known as advanced driving-assistance systems (ADAS) [2].

ADAS rely on a continuous stream of data from multiple sensors that measure internal and external variables to provide advanced functionalities [10]. They can be classified into two categories taking into account their actuation level: passive and active ADAS. The former provides advice or information to the driver. Examples of passive ADAS are blind-spot sensors that use ultrasound sensors to detect obstacles in the blind spot of the rear-view mirrors. Collision-avoidance systems use radar, and seldom LIDAR or video, to detect potential front collisions, warning the driver (forward-collision warning (FCW)) [11]. Lane-departure warning systems (LDW) [12] detect lane marks by video signals and inform the driver about lane-departure events. Finally, traffic sign identification/recognition systems (TSI/TSR) are able to detect speed limits displaying warnings in the dashboard of the vehicle. The latter, active ADAS, can perform actions on the car, and includes systems such as autonomous emergency braking (AEB) [11], which can automatically stop the car. Lane-keeping assistance (LKA) [12] is a step forward from LDW systems; it can correct the trajectory of the vehicle by autosteering manoeuvres to avoid unintended lane changes. Other examples of active ADAS are the above introduced ACC and automatic speed assistants (ASA), which, based on TSI/TSR and jointly acting with a global positioning system (GPS) signal, automatically apply the speed limit of the road to the car [13].

The aforementioned systems have been demonstrated to improve safety in cars since they relieve drivers from some of the most safety-critical tasks [14]. The conjunction of all of these intelligent sensors, together with high-speed wireless communications, have allowed car-makers for the first time to develop intelligent vehicles with automated driving systems [15]. The acceptance of ADAS by drivers mainly depends on the engineering factors of the system, predefined by technicians and implied by system functional specifications [16]. Thus, for the longitudinal control of vehicles, the time gap with the precedent vehicle, known as the time headway (THW), is monitored by means of radar sensors, considering the longitudinal area of safe travel in relation to the predefined stop distance parameter limits set by the manufacturer. However, despite these systems being very easy to use and contribute to improving road safety, acceptance by the motorists has been limited [17]. This lack of acceptance is caused by drivers perceiving the car as not as natural and stable as expected, despite it

being programmed to be as conservative and safe as possible. Furthermore, FCW systems' conservative THW safety parameters may make some drivers feel distrust on the systems and annoyance, as they return an excessive number of warnings, making motorists prone to dismiss this advice or even to disengage the system [18]. This lack of use causes their effect on global sinistrality rates to not be as good as previously foreseen, despite ACC systems having been gradually installed in new production cars and demonstrated to contribute in reducing accidents' severity, even eliminating them [19].

Personalization Approaches in ADAS

Several car-makers have introduced a slight level of ADAS customization by allowing users to manually select the THW parameter from a knob or a lever in the steering wheel, always within the pre-engineered parameters. However, despite a group of drivers appreciating the freedom that manually adjustable parameters provide them in automatic longitudinal control modes; this personalization process, including the operation of the system itself, can be complicated for other drivers not so familiarized with control systems in automobiles. Therefore, it seems reasonable to introduce personalization strategies that need no driver intervention to make the adoption of these systems easier for that sector of the automotive community [20]. ADAS personalization embeds characteristics of motorists' driving style into the system. The driving style (DS) is the manner the driver operates a vehicle in terms of steering, acceleration and braking, and how this driver relates to the other ones in terms of predictability and aggressiveness [21]. There exist two approaches to personalize ADAS depending on DS: individual-based and group-based.

Individual-based personalization strategies try to reproduce or identify the DS of a given individual using ML techniques or mathematical models. Within this scope, in the works by the authors of [22,23], ACC was adapted to individual drivers in real time based on DS observations, achieved by the recursive least-squares (RLS)-based fitting of a linear car model. The model reproduced the time gaps observed in a short manual-driving session (learning mode) and mimicked these learned time gaps when the personalized ACC was enabled (running mode). In the work by the authors of [24], an ACC was developed with the same approach as previous, but introducing a forgetting factor with the learning of the driver parameters occurring when the driver is manually controlling the vehicle while following a lead vehicle. A different approach was chosen in the work by the authors of [25], where a learning, hidden Markov model-based driver model, combined with a model predictive control (MPC) algorithm, was used to create personalized driver assistance able to imitate different DSs. Regarding FCW/AEB, in the work by the authors of [26], the personal DS was statistically modeled to estimate driver-specific probability distribution of danger level to determine the activation threshold of the system. Individual-based strategies have the main advantage of entirely mimicking the DS but, despite this feature being very desirable, it generally requires intensive computation, hard to be achieved in real-time. Moreover, the modeled behaviors would require safety verification since not all drivers handle vehicles in a correct way. To mitigate these drawbacks, group-based approaches have emerged.

Group-based personalization strategies locate drivers in a small number of representative DSs for which a control strategy is implemented. In the work by the authors of [27], a group-based approach of the driver's ACC preferred time gap is presented. The drivers were clustered to create three general driver profiles to be used, together with demographic information, to predict the gap by using a regression model and decision trees. The authors of [28] describe a stop-and-go ACC system that groups drivers into three clusters depending on their DS, with the cluster membership determining the reference acceleration profile to adjust the ACC controller. Recently, in the work by the authors of [29], a support-vector-machine (SVM)-based approach was used to classify driving behavior into two different clusters in order to select a personalization parameter for an ACC. These strategies, despite not entirely mimicking DS, are computationally more efficient, requiring less computation since they work on a previously offline-trained classification algorithm, allowing online, real-time

computation. On the other hand, as they represent a class-averaged DS, they can easily be validated and verified to always operate in safe margins.

In this work, a hybrid personalization strategy for DS modeling is proposed. To perform system development, data from a subset of real-world trips of the Strategic Highway Research Program (SHRP2)'s naturalistic driving study (NDS) [30] are used. First, a group-based technique was used with the aim of building a three-cluster DS classifier. Then, each the clusters was approximated by means of an adaptive neuro-fuzzy inference system (ANFIS) obtaining identification rates higher than 85.7% for the three clusters. Finally, an individual-based algorithm was used to adapt the behavior of the group to a particular driver. The whole system was successfully implemented using an FPGA device of the Xilinx's Zynq programmable system-on-chip (PSoC). The system can mimic the typical timing parameters of a group of drivers as well as tune these typical parameters to model individual DSs. The neuro-fuzzy intelligent sensor provided high speed for real-time active ADAS implementation and could personalize its behavior into safe margins without driver intervention. In particular, the personalization procedure of the THW parameter for an ACC in steady car-following scenarios is described.

The remainder of this paper is organized as follows. Section 2 provides an outline of the proposed DS personalization system. In Section 3, the driving style characterization methods in car-following scenarios are presented. The neuro-fuzzy modeling approach of driving style groups is provided in Sections 4 and 5 exposes the implementation of the FPGA-based intelligent sensor and provides experiment results for a personalized ACC in a steady car-following situation. Finally, concluding remarks are summarized in Section 6.

2. Outline of Driving-Style Personalization System

The intelligent sensor developed in this work relies in two well-differentiated stages: an offline design stage and the online in-car operation stage. The sequence of tasks involved in the offline design stage are depicted in Figure 1.

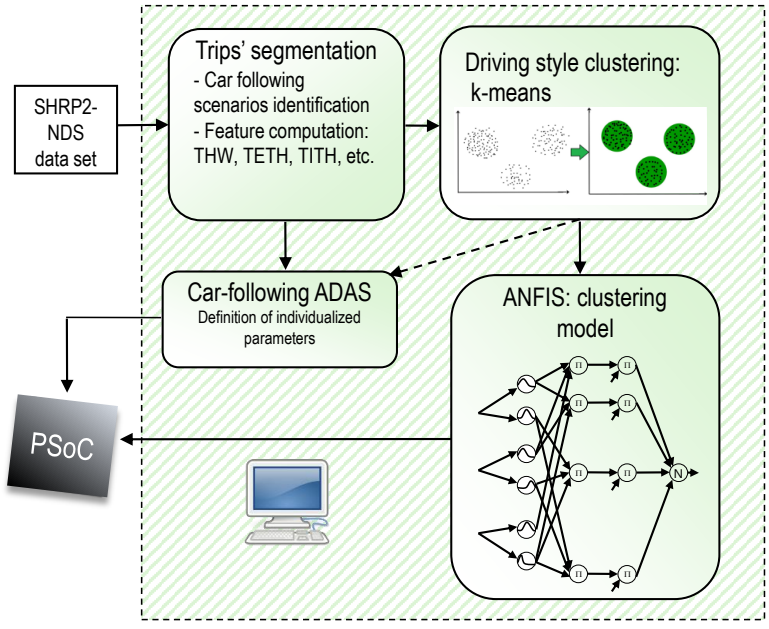

Figure 1. Offline sequence of tasks involved in the design and development of a neuro-fuzzy sensor for advanced driving-assistance system (ADAS) personalization.

The first task comprises the segmentation of SHRP2-NDS trips with the aim of selecting car-following scenarios, and the computation of a set of meaningful car-following parameters or features for each segment of the trip (e.g., THW, time-exposed THW (TETH) and time-integrated

THW (TITH) [19], which will be defined later in Section 3.2.1). Next, an unsupervised clustering technique, the k-means algorithm, was used to group driving styles into a number of clusters in car-following circumstances. Then, an ANFIS was trained in order to develop a high-performance model of the DS classifier. The main advantage of using an ANFIS-like model is that it was suitable for the development of high-speed parallel-hardware architectures, allowing in-car DS classification for ADAS personalization in the online stage. Thus, the ANFIS model was implemented using an FPGA of the Xilinx Zynq device family. More precisely, it is a PSoC that integrates microprocessors and their peripherals with programmable logic.

The FPGA-based driving style classifier acted as an intelligent sensor able to adapt the ADAS response to driver preferences in longitudinal car-following scenarios. In particular, a steady car-following application was developed: a personalized ACC.

Figure 2 depicts a block diagram of the FPGA-based implementation of the intelligent sensor for personalized ADAS. We used a Xilinx Zynq 7000 family PSoC [31]. The proposed system was composed of two main modules: the software partition, implemented on the processing system (PS), and the hardware partition, implemented on the programmable-logic (PL) section of the PSoC. Note that the PS consisted of a dual-core ARM Cortex A9 microprocessor, whereas the PL comprised typical FPGA resources such as logic blocks, interconnections, digital signal processing (DSP) cores, and random access memory (RAM) blocks. Additionally, the PSoC system had several interfacing hard modules for field buses (Ethernet, CAN-bus, etc.). Considering the characteristics of PS and PL, the distribution of the tasks to be performed by the proposed system is explained in the following lines.

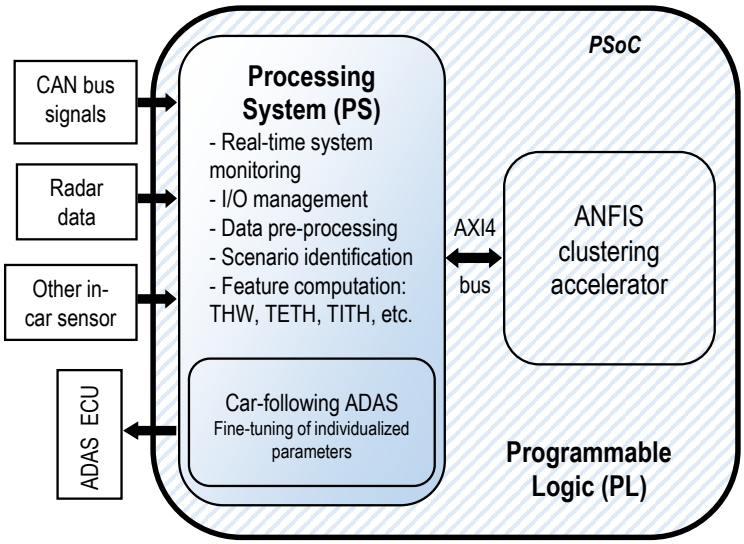

Figure 2. Block diagram of the field-programmable gate array (FPGA)-based intelligent sensor for online car-following ADAS.

The PS, apart from performing global system monitoring, was connected to the vehicle systems through field buses, managing the I/O interface of the vehicle's systems with the proposed solution in this work. Therefore, the PS was responsible for capturing the input data from both the radar sensor and the standard information from the CAN-bus. Regarding input data signals, the system is fed with the distance with the preceding vehicle and relative speed between the host vehicle and the preceding one (both from the radar sensor), and with the host vehicle speed (standard information from the CAN-bus). With those signals, the PS computes the driving features that allow to perform DS classification. These features, stated in Figure 2, are computed according to Equations (1), (3) and (4) from Section 3.2.1. Inside the PSoC, once the features have been computed, they are sent from the PS to the PL by means of an internal bus, on which the entire PSoC architecture is based: the Advanced eXtensible Interface 4.0 (AXI4-bus). According to features sent from the PS to the PL through the AXI4 bus, the VHDL-based PL-implemented ANFIS clustering accelerator classifies DS and sends the

classification results back to the PS through the AXI4-bus. With the received classification results, the PS computes the ACC personalization parameter and, finally, sends it through the vehicle's CAN-bus to its electronic control unit (ECU), responsible for the ACC.

Thus, in summary, and according to Figure 2, the PS executes the tasks of interfacing with the vehicle buses to collect the input data, computes selected identification features, sends them to the ANFIS accelerator deployed in the PL through AXI4 bus, collects the outputs of the accelerator, computes the personalized ACC parameters, and sends them to the ACC module. On the other hand, as it is computationally intensive, the PL implements an ANFIS-based classification that needs to be executed as fast as possible. Using this architecture, in Section 5, an ACC application with personalized behavior is presented.

3. Driving Style Characterization in Car-Following Scenarios

Many research works have been conducted in the field of intelligent vehicles, and their sensors, to understand how drivers and traffic behave and to determine which sensors are suitable for each situation. Most of these studies rely on cars instrumented with CAN-bus loggers, inertial measurement units (IMUs), radars, lidars, and video cameras to collect meaningful data. There exist two main branches in driving studies: non-naturalistic and naturalistic.

Non-naturalistic studies, such as NU-Drive [32], UYANIK [33], and UTDrive [34] make use of dedicated instrumented cars, which simplifies data collection and logistics by increasing the number and complexity of the boarded sensors. The selected human subjects (motorists) do not drive their own cars, so the collected data may not reflect real driving situations. This approach also involves simulator-based road-safety studies, self-report studies, statistical analysis, and authority-investigated crashes. However, despite the fact that these methods have greatly contributed to understanding how road users behave and which factors involving crashes are the most important, they do not reflect completely realistic situations. Therefore, NDSs have been conducted to compile data that faithfully reflects driver behavior in every-day traffic situations. NDSs, pioneered by the Virginia Tech Transportation Institute (VTTI) [35], are the most recent trend in traffic safety and ADAS research.

Meaningful examples of NDSs are UDRIVE [36], carried out in the EU, and the 100-Car NDS [37] and SHRP2-NDS [38], carried out by the VTTI. These studies focused on collecting data from drivers (human subjects) in their own vehicles and environment in everyday trips without interfering in any normal behavioral patterns, that is to say, with no experiment control. Thus, NDSs allow for the observance of normal driver situations, providing much better feedback to correctly understand drivers' behavior in normal, unguided traffic situations, as participants do not have the feeling of being involved in an experiment. Consequently, NDSs are more useful to customize real ADAS. In the following, the SHRP2-NDS, used in this work, is briefly introduced.

3.1. SHRP2-NDS Description

The main objective of the SHRP2 project is to address the influence of driver performance and behavior on traffic safety. This involves understanding the way the driver handles and adapts to a vehicle, roadway characteristics, traffic lights, signs, infrastructure, and other environmental features. SHRP2-NDS offers two key advantages: detailed and accurate precrash information, including objective information about driving behavior, and exposure information, including the frequency of behaviors in normal driving. To take part in this study, the participants' vehicles are checked for their suitability to fit the systems used in the NDS. Next, while the data acquisition system and instrumentation are installed, the participant serves several driver-assessment tests as well as medical examinations for a total of 2–3 h.

The VTTI developed a custom data acquisition system for the SHRP2-NDS [38]. This system, whose main blocks are displayed in Figure 3, was manufactured by American Computer Development Inc. and includes a forward radar, four video cameras (with a forward-facing one, color, and wide-angle view), an IMU with XYZ accelerometers and gyroscopes, vehicle network (CAN bus) data logging,

GPS-based location, computer vision-based lane tracking, and data-storage capability. Additionally, the data acquisition system has cellular connectivity to provide emergency-call functionalities, system health checks, and software updates. The captured data, including the radar, are uniformly sampled at a rate of 10 Hz, and all the different sources are properly synchronized.

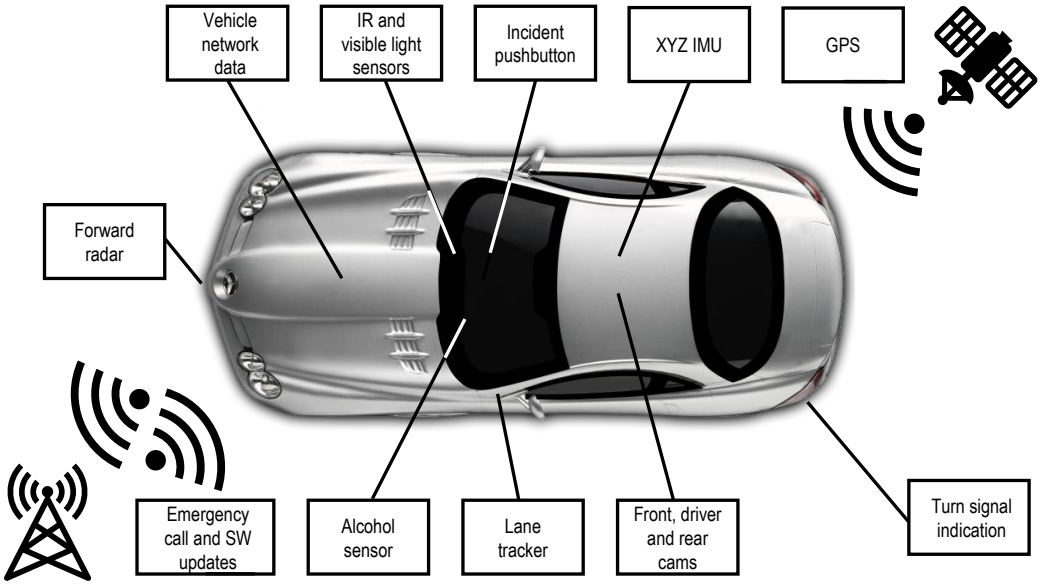

Figure 3. Data acquisition systems and sensors installed in the vehicles that participated in the SHRP2-NDS. IR: infrared; SW: software.

3.2. Car-Following Situations

Car-following describes a driver following another driver in a traffic stream. For that purpose, drivers operate throttle and brake pedals to maintain a desired range of distance from the preceding vehicle. The main objective of modeling car-following behavior is predicting the following vehicle speed and distance based on stimuli provided by the preceding vehicle for a set of road and driver characteristics. A retroactive approach can be applied in this topic, that is, based on DS and speed, predicting the desired distance between the following and the leading vehicle. This is the base of several ADAS, such as ACC, FCW, or AEB.

3.2.1. Driving Parameters and Driving-Style Characterization

The main step to correctly identify DSs is determining the adequate variables to provide a robust enough identification. The signal choice depends, however, on the desired application, which is crucial as any further processing of that data will entirely depend on that choice. Therefore, due to the plurality of applications, ranging from ADAS personalization [39] or driving correction for safety and comfort improvement [40] to fuel economy advice, there is no recommended set of parameters. Thus, for identifying aggressive drivers, high accelerations must be monitored. On the other hand, speed profiles should be monitored to analyze fuel efficiency. Additionally, the acceleration variable is generally combined with brake activation and speed measurements. In other works, pressure on the brake and throttle pedals are used as reliable indicators to identify DSs [41,42]. Furthermore, the selection of which signals are the most adequate for each application might be guided by the identification or apparition of some circumstances, or the restriction of the identification problem to some contexts or driving events, such as braking, distance-keeping, roundabouts, cornering, lane changes, or even car-following.

In this work, the DS characterization problem was restricted to steady car-following scenarios where clear distance-keeping behavior was observed. In car-following scenarios, the most relevant variables are the speed of the host vehicle (v), the relative speed of the host to the leading vehicle

(v_r), and the distance between host and leading vehicle (d). With these variables as a starting point, we derived features to parameterize car-following scenarios such as time headway (THW) and the inverse of time-to-collision (TTCi), which are commonly used.

$$THW = \frac{d}{v} \tag{1}$$

$$TTCi = \frac{v_r}{d}. \tag{2}$$

THW (Equation (1)) is the time difference between two successive vehicles when they cross a given point, whereas TTCi (Equation (2)) is the inverse of the time two vehicles would require to crash if they kept the same speed and trajectory. These parameters are often used to assess car-following styles. Nevertheless, other parameters can be used to provide more complete insight on DS in car-following scenarios. These parameters are time-exposed time headway (TETH) and time-integrated time headway (TITH) [19]. Thus, TETH (Equation (3)) represents the time exposure to a THW lower than a predefined safety threshold during a ride.

$$TETH = \sum_{t \leq T} \delta_i(t)\tau_s \qquad \delta_i = \begin{cases} 1 & \forall\, 0 \leq THW_i(t) \leq THW^* \\ 0 & \text{else,} \end{cases} \tag{3}$$

where T is the total time interval considered, $\delta_i(t)$ is a binary activation parameter, τ_s is the sampling period, $THW_i(t)$ is the instantaneous value of THW at a given moment t, and THW^* is the predefined safety threshold value (Figure 4a). On the other hand, TITH (Equation (4)), is the summation of the difference between THW^* and $THW_i(t)$ restricted to time intervals when $THW_i(t) < THW^*$ (Figure 4b).

$$TITH = \sum_{t \leq T} \left[THW^* - THW_i(t)\right] \delta_i(t)\tau_s \qquad \delta_i = \begin{cases} 1 & \forall\, 0 \leq THW_i(t) \leq THW^* \\ 0 & \text{else.} \end{cases} \tag{4}$$

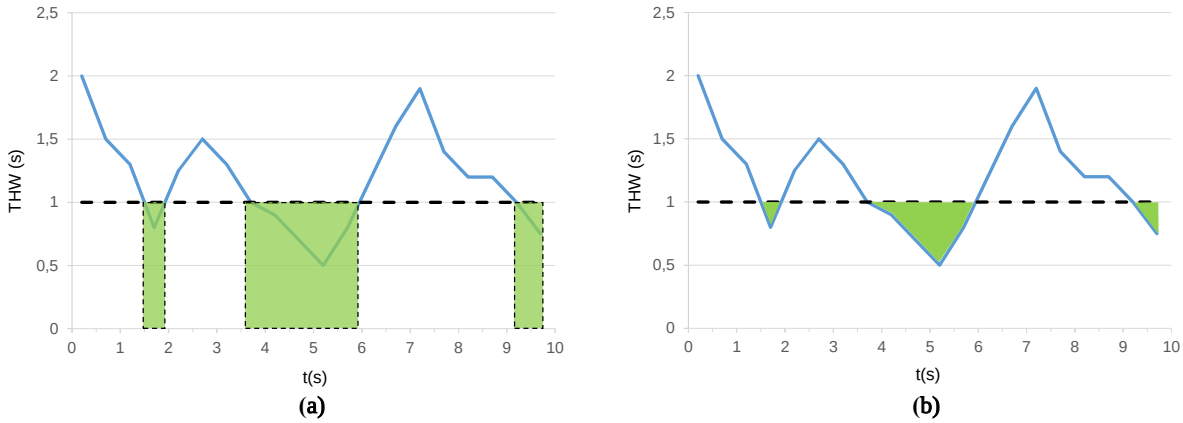

Figure 4. Representative example of car-following features: (**a**) time-exposed time headway (TETH) and (**b**) time-integrated time headway (TITH).

The above parameters have been found to characterize the following behavior in both motorists used to activate the ACC mode and non-ACC users [19]. Thus, as TETH is the time a driver rides behind a leader car below a certain THW threshold, we can determine which percentage of that trip occurs at a time distance below the recommended values. Simultaneously, TITH enables to measure how close a following vehicle has got to its leader during the TETH. Consequently, the use of these parameters jointly with the THW root-mean-square (RMS) value THW_{RMS} provides a good measure of driver behavior. As is seen later, this set of features is very helpful to identify car-following driving styles.

3.2.2. Steady Car-Following Premises

Car-following scenarios include accelerating, braking, approaching, and steady following [43]. Steady car-following circumstances occur when the relative speed between vehicles is low and $|TTCi| \leq 0.05\ s^{-1}$ [43]. The statements used in this work to segment the trips into steady car-following stretches are the same as those used in the work by the authors of [44]. Thus, it was assumed that there was a lead vehicle in front of the host vehicle and this leader traveled in the same lane. Additionally, the lead vehicle must have been at a maximum distance of 120 m and the host vehicle must have been traveling at 20 km/h at least. The maximum distance was constrained because the radar sensor could detect targets beyond that 120 m range, and following behaviors with vehicles at such distance are negligible. On the other hand, a minimum-speed constraint was applied to filter traffic jams, which cannot be considered real car-following. Additionally, the segments of interest were restricted to those lasting more than 30 s.

3.3. Car-Following Stretches in the SHRP2-NDS Trips

The selected car-following stretches were extracted from 48 different trips of 40 different drivers [45]. Each driver was identified with a numeric code that eased identification of the driving data while preserving their privacy. Most of the trips contained mixed-environment driving, ranging from parking lots and streets to motorways and highways. These trips were selected so as to involve different traffic situations as well. Different traffic situations enable researchers to better understand driver behavior and how drivers relate to each other in complex contingencies in both regular transit and safety compromising events. The segmentation of trips into car-following stretches is not trivial, and many parameters should be considered to perform it.

Steady Car-Following Segments

After applying the premises of Section 3.2.2 over the 48 trips, a total of 115 continuous car-following stretches were segmented. Nevertheless, these 115 stretches were extracted from 28 of the 48 selected trips, as the 20 remaining trips did not contain stretches that gathered the characteristics delimited in steady car-following premises. It is worth noting that the segmented stretches were not evenly distributed among the 28 trips. Thus, in order to uniformize the length of the segmented stretches and consequently reduce standard deviation to increase comparability, all stretches were split into smaller ones lasting between 30 and 59.9 s. Additionally, those with $THW_{RMS} > 4.5$ s were discarded because with this THW we could not assure significant car-following events. This new partition was composed of 176 uniformized segments with a duration of $T_{RMS} = 37.3$ s and a standard deviation of $\sigma = 8.18$ s.

4. Neuro-Fuzzy Modeling of Driving-Style Clusters

The first task involved in the design of the neuro-fuzzy sensor was the segmentation of the SHRP2-NDS trips into the set of steady car-following segments introduced in Section 3.3 and the computation of the selected features: THW_{RMS} (Equation (1)), TETH (Equation (3)), and TITH (Equation (4)) for each one of the 176 segments. These parameters are representative of a longitudinal DS in steady car-following situations; therefore, they could be used to personalize ADAS. Consequently, this task consists in grouping together similar driving styles using a clustering approach.

4.1. Driving-Style Clustering

Several clustering algorithms have been used to distinguish DSs and DS-class labeling [46]. The selection of a concrete algorithm depends on the trade-off between complexity and performance. Mean-shift clustering is based on finding high-density data areas by means of a sliding window of a specified radius, aiming to locate the centroid of each area. This algorithm has the advantage of not needing to know the number of desired clusters, as the algorithm detects them by itself, but as a

weakness; it should be pointed out that the selection of the window radius may be nontrivial. Another used clustering algorithm is density-based spatial clustering of applications with noise (DBSCAN) [47], which can filter outliers and find arbitrarily shaped and sized clusters, but does not perform well when clusters have variable density. Expectation–maximization (EM) clustering using Gaussian mixture models (GMM) is a flexible algorithm in terms of cluster covariance [48], which bridges the restriction of distance-based solutions that only work on circular-shaped clusters. Additionally, since GMMs use a probability cluster, a given datapoint can belong to several data clusters with different probability, allowing mixed membership. Agglomerative hierarchical clustering (HCA) relies on building a tree depending on the similarity/dissimilarity between data points/clusters [49], and due to this characteristic, HCA does not need to preselect a number of clusters and it is particularly suitable to recover underlying hierarchical data structures. However, when there are particularities in some of the input data, HCA tends to group all of those particular points together, causing cluster unbalance. Finally, several research pieces in driver identification [50] and road condition monitoring [51] have successfully used the k-means algorithm, as it useful for the proposed group-based DS identification application. The k-means algorithm is simple and quick since it is based on computing distances between each point and the groups' centroids. However, since the number of clusters must be previously specified, and the centroids of each cluster are randomly initialized, the repeatability of this type of clustering is not always assured. Nevertheless, it is quick enough to execute multiple runs in a reasonable period of time. Additionally, input data groups elaborated by k-means can easily be interpreted. Hence, due to prior characteristics, the k-means algorithm has been used in this work to carry out DS grouping.

K-Means Clustering Results

First, the selected features THW_{RMS}, TETH, and TITH features were computed for each of the 176 driving segments and normalized into the [0,1] range. According to the minimum following safety threshold of [16] and the selected THW in the work by the authors of [19], TETH and TITH were calculated for a critical value of $THW^* = 1.5$ s. After that, three-group k-means clustering of those segments was performed. The obtained cluster structure is depicted in Figure 5. Note that, for THW_{RMS} values higher than THW^* 1.5 s, TETH and TITH were always 0, generating the blank zone at the right $TETH - THW_{RMS}$ semiplane of the figure.

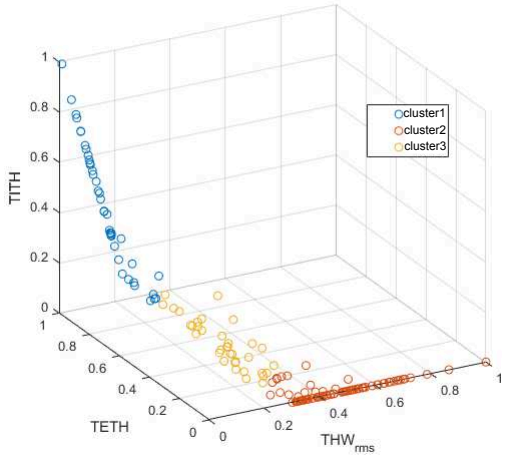

Figure 5. Clusters obtained applying the k-means algorithm to the car-following segments; THW_{RMS}, TETH, and TITH values were normalized.

DS groups were found to be stable and highly reproducible despite the randomness of the cluster centroid initialization. Therefore, within these data, a unique solid structure could be found. In Figure 6,

the distribution of THW_{RMS}, TETH, and TITH values according to this normalized cluster structure is shown. Given the distributions displayed in the figure, the clusters could be described as follows:

- Cluster 1: groups the drivers with the lowest THW_{RMS} and the highest TETH and TITH. This cluster is representative of the most aggressive car followers.
- Cluster 2: groups the drivers with high THW_{RMS} values and minimum TETH and TITH. Thus, it incorporates the least aggressive car followers.
- Cluster 3: groups the drivers with low THW_{RMS} values, medium to low TETH and the lowest TITH, representing medium aggressive car followers.

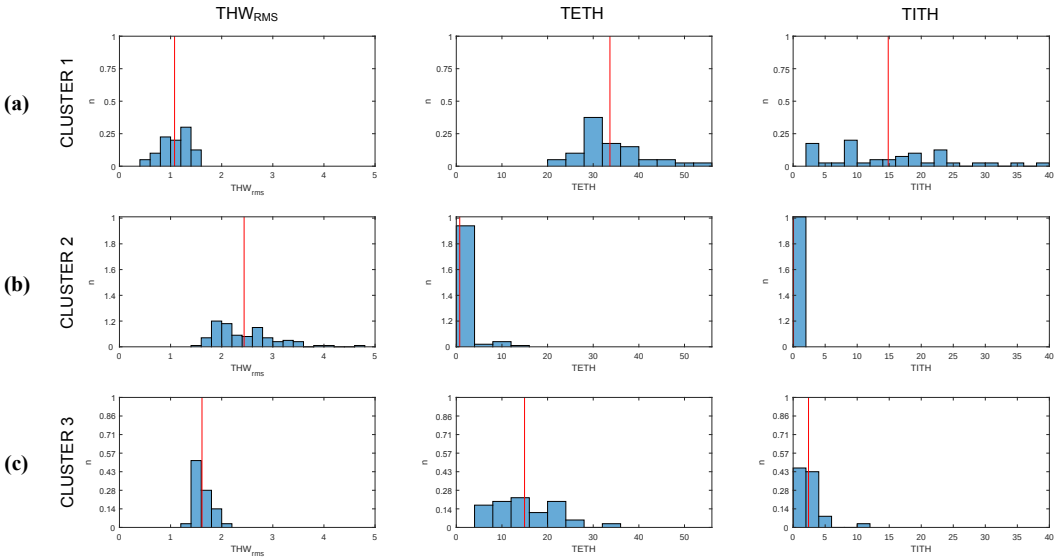

Figure 6. Histogram of THW_{RMS}, TETH, and TITH values distribution for (**a**) Cluster 1, (**b**) Cluster 2, and (**c**) Cluster 3. Red line represents average value of each distribution.

4.2. ANFIS-Based Identification

The second task in the development of the proposed intelligent sensor was the development of a high-performance model of the DS classifier obtained above. AS the proposed system was intended to accomplish online DS identification, clustering techniques could not be directly used. There is a variety of solutions suitable for efficient real-time hardware implementation. Thus, artificial neural networks (ANNs) were used in the work by the authors of [52] to score drivers depending on the safety of their DS. In the work by the authors of [53], finite-state machines (FSMs) were used to decide whether a driver belongs to a DS class depending on their driving decisions. Nevertheless, an outcoming line of work in DS identification is based on fuzzy-logic implementations [54]. Thus, in the work by the authors of [55], anomalous DS was identified applying a fuzzy inference system (FIS) on accelerometer data, and an FIS-based dangerous DS identification application was proposed in the work by the authors of [56]; in the work by the authors of [21], fuzzy logic was applied to identify DSs in an online fashion.

To accomplish this task, an adaptive neuro-FIS (ANFIS) was used since this system was suitable to model clusters with online performance. Thus, once the k-means clustering algorithm classified the steady car-following segments of Section 3.3 into three DS clusters depending on their THW_{RMS}, TETH, and TITH, an ANFIS was trained for each one of the three clusters. Each ANFIS model returned a continuous value indicating the fitting of the prior input parameters into each of the clusters. Attending to those output values, the ANFIS model with the maximum output identified the cluster to which the inputs belong.

4.2.1. Zero-Order Takagi–Sugeno Inference System

The ANFIS model used in this work was based on a zero-order Takagi–Sugeno inference system using prod-sum operators with the product for the inference process and the sum for the aggregation of the rules [57,58]. Consider set of rules:

$$R_j : \text{IF } x_1 \text{ IS } A_{1j}(x_1) \text{ AND } x_2 \text{ IS } A_{2j}(x_2) \text{ AND } \cdots x_n \text{ IS } A_{nj}(x_n) \text{ THEN } y \text{ IS } c_j, \tag{5}$$

where R_j is the jth rule $(1 \leq j \leq m)$, $x_i (1 \leq i \leq n)$ are input variables, y is the output, c_j is a constant consequent, and $A_{ij}(x_i)$ are linguistic labels; each one is associated with a membership function, $\mu_{ij}(x_i)$. In zero-order Takagi–Sugeno fuzzy models, the inference procedure used to derive the conclusion for a specific input $\mathbf{x} = \left(x_1^0, x_2^0, \cdots, x_n^0\right)$ consists of two main steps. First, the firing strength or weight w_j of each rule is calculated as

$$w_j = \prod_{i=1}^{n} \mu_{ij}\left(x_i^0\right). \tag{6}$$

Next, overall inference result y is obtained by means of the weighted average of the consequents

$$y = \frac{\sum_{j=1}^{m} w_j c_j}{\sum_{j=1}^{m} w_j} = \frac{N}{D}. \tag{7}$$

In this particular application, there are $n = 3$ inputs ($x_1 = \text{THW}_{\text{RMS}}$, $x_2 = \text{TETH}$, and $x_3 = \text{TITH}$), whereas for each of the inputs, three linguistic labels, namely, *LOW*, *MEDIUM*, and *HIGH*, were assigned and 27 rules were generated ($m = 27$). Membership functions μ_{ij} associated to these labels are bell-shaped functions, which are defined as follows,

$$\mu_{ij}(x; a, b, e) = \frac{1}{1 + \left|\frac{x-e}{a}\right|^{2b}}, \tag{8}$$

where a defines the width of the membership function, b is the steepness of the curves at each side of the center plateau, and e is the center of the function.

Finally, to train the ANFIS, a combination of a least-squares estimator (LSE) and a gradient-descent method (GDM) were used. Each epoch of this learning process was composed of a forward pass and a backward pass. In the forward pass, consequent parameters, c_j, were identified by the LSE method, and in the backward pass, antecedent parameters a, b, and e were updated by the GDM.

4.2.2. ANFIS Training

The 176 segments from Section 3.3 were labeled in accordance with the clustering described in Section 4.1 (see Figure 5). Once the segments were labeled, they were partitioned into a training set (75% of the samples) and a test set (25% of the samples). Each ANFIS cluster was trained and tested with the same set of data since they were designed to decide whether the input data belong to the cluster they represent. Consequently, membership was represented with a value of "1" if the data belonged to the cluster modeled by that ANFIS, and with "0" if not.

As can be seen in Figure 7, THW_{RMS} and TETH values were the same for all the clusters, and despite this figure being a 3D plot and TITH not being able to be displayed, it also coincided for all three ANFIS models. It could also be observed that the value of the Z-axis was "1" when data points belonged to the cluster to be modeled, and "0" when not.

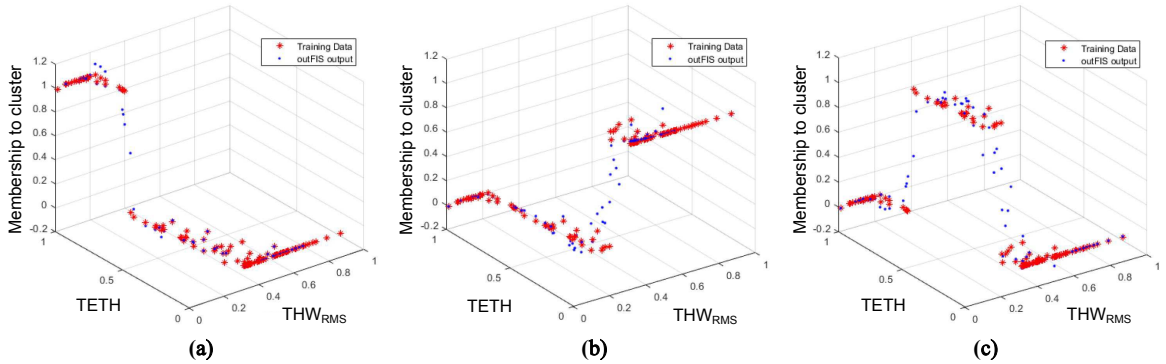

Figure 7. (**a**) Adaptive neuro-fuzzy inference system (ANFIS) 1 (Cluster 1), (**b**) ANFIS 2 (Cluster 2), and (**c**) ANFIS 3 (Cluster 3). Training data shown in red; response of corresponding trained ANFIS shown in blue.

4.2.3. ANFIS Testing and Identification Results

After the training stage, the remaining 25% of the steady car-following segments (see Section 3.3) are used to test the identification performance of the ANFIS-based DS identificator. Thus, the test segments were simultaneously input to the three ANFIS-clusters, and the neuro-fuzzy system with the highest output was considered to be the class to which the segment belonged. With this procedure, the outputs of the system were evaluated, showing an accuracy mark of 95.45%. This mark was much higher and the classification more detailed than those obtained in previous works from other authors, such as the work by the authors of [29], where, using an SVM, a given driver's DS was classified between only two clusters with an accuracy of 85%.

Additionally, the confusion matrix in Table 1 gives deeper insight into this accuracy result. By analyzing this matrix, accuracy rates of 85.71% for Cluster 1, 96.43% for Cluster 2, and 100% for Cluster 3 were reached by the final ANFIS. Nevertheless, confusions reflected in the matrix happened between contiguous clusters; hence, no erratic classification behavior happened while identifying DSs with this system.

Table 1. Confusion matrix of ANFIS-based DS identifier.

Actual/Identified	Cluster 1	Cluster 2	Cluster 3
Cluster 1	6	0	0
Cluster 2	0	27	0
Cluster 3	1	1	9

5. Implementation of FPGA-Based Intelligent Sensor

A block diagram of the proposed DS-based ADAS personalization solution is displayed in Figure 2. It is a hybrid hardware/software (HW/SW) architecture implemented on the Xilinx XC7Z045-2FFG900 PSoC [31] using the Xilinx ZC706 development board [59]. The entire HW partition of the system (deployed in the PL of the PSoC) was implemented using VHDL language and the Xilinx Vivado 2018.1 design suite. On the other hand, the remainder of the proposed system with its functionalities was to be programmed at the PS by developing a bare-metal C application that can acquire vehicle bus data; compute the THW, TETH, and TITH features; share them with the PL; retrieve the ANFIS accelerators' results; compute the personalization parameters; and send them to the ACC ECU.

5.1. Hardware Partition: ANFIS Accelerators

The building blocks on which the ANFIS HW accelerators rely were structured according to the Takagi–Sugeno Inference System (Equations (6)–(8)). After several tests, system inputs (THW^0_{RMS}, $TETH^0$ and $TITH^0$) were represented using an 8-bit fixed-point fractional data format, the bit widths

of the intermediate operations were properly propagated and trimmed for not losing precision, and output y was trimmed to a 32-bit two-complement fixed-point representation, chosen to match with the AXI4 bus width. As can be seen in Figure 8, the proposed ANFIS architecture was organized in four layers. In the first layer, the membership of the system inputs to the antecedents of the rules were evaluated. Then, in the second layer, rule activations were concurrently computed (Equation (6)). Next, in Layers 3 and 4, the weighted average of the consequents was calculated (Equation (7)). The HW partition was composed of three ANFIS accelerators, one per cluster.

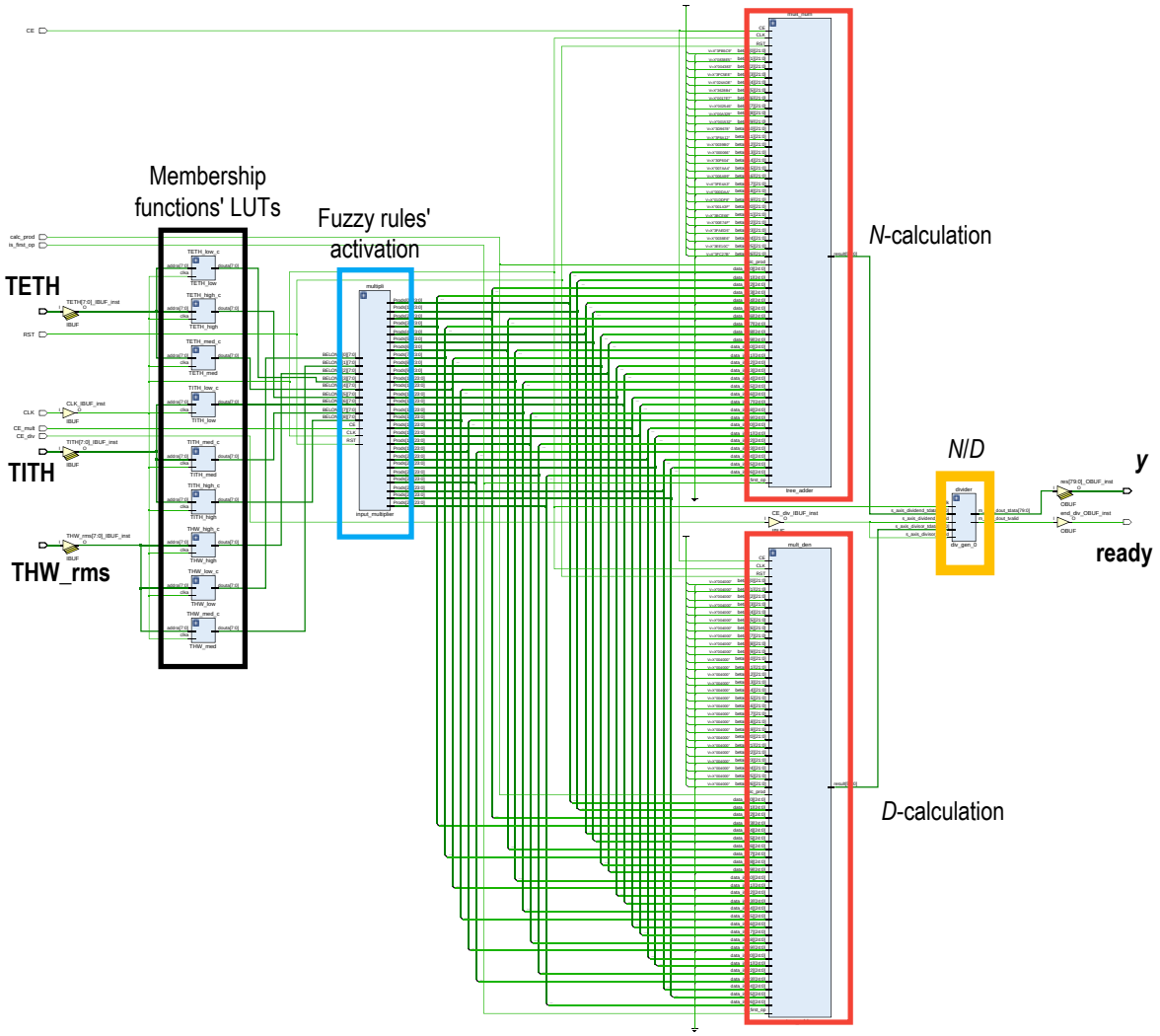

Figure 8. Block scheme of the parallel architecture of a three-input ANFIS implemented in the programmable logic (PL) of the programmable system-on-chip (PSoC). Three ANFIS cores, one per cluster, were implemented in the HW partition.

5.1.1. Membership Function Evaluation and Fuzzy-Rule Computation

The generalized bell-shaped membership functions were precalculated and stored as look-up tables (LUTs) (remarked in black in Figure 8) at the PL block RAMs (BRAMs). Therefore, evaluation of the input membership to each antecedent was straightforwardly obtained by addressing those values, lasting only one clock cycle. Once the input membership functions were evaluated, fuzzy-rule activations were calculated. As there were three membership functions for each of the three inputs, 27 weights were to be computed. These fuzzy-rule activations were three-input products, computed by a fuzzy-rule activation module remarked in blue in Figure 8. These products were efficiently computed by a full-VHDL design intended to only use Xilinx DSP resources [60], improving timing performance.

To achieve this DSP-only implementation, the three-input products were done two by two. Thus, first the product of two of the inputs was calculated and stored in an intermediate result pipeline register and second, the stored partial product was multiplied by the remaining input and saved in the output register. This product pipeline required two clock cycles.

5.1.2. Computation of Sum and Weighted Sum of Rule Activation

In the work by the authors of [61], a high-performance product–architecture sum, developed by the authors, was described. This topology, shown in Figure 9, replaced the tree-adder architecture. It was intended to minimize latency, save resources, and minimize the number of used DSPs. Thus, for a given number k of products, the proposed architecture only used k multiplier/adder blocks, while a tree adder would spend $2k-1$ of the same hardware resources. This architecture, with inputs $\mathbf{u} = (u_1, \cdots, u_k)$ and $\mathbf{v} = (v_1, \cdots, v_k)$, control signals `is_prod` and `CE`, and output p operates as follows.

1. Product signal `is_prod` set to "1" and all registers are reset.
2. Product $u_j \cdot v_j$ with $j = 1, ..., k$ computed and stored in each of the k accumulator registers.
3. Signal `is_prod` set back to "0" and first accumulation is performed. Thus, accumulator registers from 1 to $k/2$ contain the sum of $u_j \cdot v_j$ products. Registers from $k/2+1$ to k are now filled with zeros.
4. Successive $\lceil log_2 k \rceil$ accumulations are performed until valid result is present in register 0.

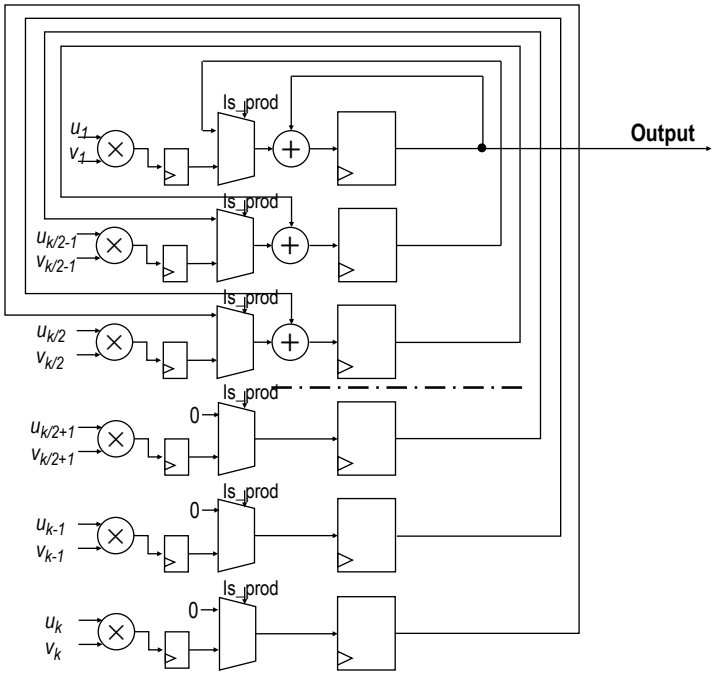

Figure 9. Scheme of proposed sum of product architecture that substitutes traditional tree-adder solution.

Two instances of this core, labeled D and N in Figure 8, are used to perform the computation of the sum and the weighted sum of rules' activation parallelly (D and N in Equation (7), respectively). In both modules, k equals the number of rules, that is, $k = 27$. For the N-module, $u_j = w_j$ and $v_j = c_j$, whereas for the D-module, $u_j = w_j$ and $v_j = 1$. A ROM storing the values of c_j is connected to the N-module. The latency of this architecture is $\lceil log_2 k \rceil + 2$; thus, with $k = 27$, the latency of both instances is seven clock cycles.

5.1.3. Divider Module

This is the last layer of the ANFIS accelerator. The divider module was elaborated by means of the Xilinx IPCore divider generator [62]. This IPCore was parameterized to match with the size of the N and D operands using the high-radix division implementation. This particular implementation could be pipelined to achieve good time performance and, as it depends on multiply–accumulate operations, it is optimally deployed in DSP blocks. With the selected word lengths and pipelined, this module requires 43 clock cycles to return valid results.

5.1.4. Parameterization and Control Signals

The complete structure of the ANFIS HW accelerator is parametric and fully customizable. LUT ROMs containing membership functions as well as consequents were simultaneously initialized. Elements such as type depths, signal bit-widths, and number of inputs, number of membership functions, or number of fuzzy rules were defined on a standalone package. The complete ANFIS was controlled by the sequence of control signals represented in the chronogram of Figure 10.

Control signals of the ANFIS HW accelerator were `rst`, `CE_mult`, `CE`, `is_prod`, and `CE_div` (see Figure 10); they worked as follows.

1. `rst` clears pipeline registers and multiplication–accumulation units.
2. `CE_mult` drives multipliers of fuzzy-rule calculation.
3. `CE` activates multiplication–accumulation units to iteratively compute N and D.
4. `is_prod`, in conjunction with the first cycle of `CE`, is used to indicate that the multiplier–accumulation unit must store the products of the fuzzy rules by their corresponding consequents instead of performing any accumulation.
5. `CE_div` triggers divider module calculating the output result of the ANFIS.

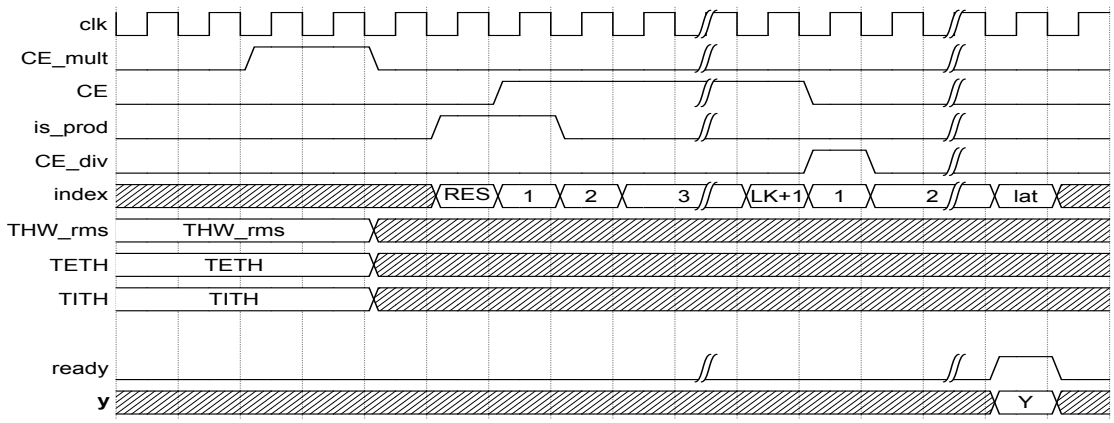

Figure 10. Chronogram of control-signal sequence of the ANFIS core.

Finishing with the ANFIS HW accelerator implementation, it is worth nothing that the three clusters of Section 4.1 must be modeled by this method. Consequently, three instances of this HW accelerator had to be deployed in the PL, each one configured with the parameters the ANFIS cluster to which it corresponds.

5.2. Experiment Results

The three ANFIS cluster HW accelerators were implemented in the selected PSoC, achieving the subsequent results.

5.2.1. Resource Usage

The full HW system was successfully implemented, with the postimplementation results displayed in Table 2. The three-ANFIS system fit into the selected PSoC's logic, leaving enough resources available for further system applications, escalations, or improvements.

Table 2. Postimplementation resources report (Xilinx XC7Z045-2FFG900).

Resource	Utilization	Available	% Used
LUT	13500	218600	6.17
Flip-flops	15759	437200	3.60
RAM blocks	15	545	2.76
DSP	294	900	32.76

5.2.2. Timing Performance

With respect to timing, the three-ANFIS' postsynthesis timing report pointed out that the minimal clock period suitable for application to the designed hardware was 7.122 ns, or a maximal clock frequency of 140.41 MHz. With this maximal clock frequency, the designed HW implementation could be used as an AXI4 peripheral dependent of an AXI4 bus clock frequency of 100 MHz. This design delayed 53 clock cycles (530 ns at $F_{CLK} = 100$ MHz) to return the computed outputs. These results outperformed the timing obtained for the full-software PC-based (20-core Intel Xeon CPU E5-2630 v4 @ 2.20 GHz with 32 GB of DD4 RAM) MATLAB model design, with top performance peaks of 1.829 ms to compute the same set of 3 ANFIS, as well as a PC-based, C-coded prototype that achieved timing marks of 12.45 µs.

The obtained timing was better than in other FPGA-based ANFIS approaches, such as the work by the authors of [63], where timings of ~12 µs were obtained in the computation of a system with the same number of inputs and outputs (three and one, respectively) as that developed in this work. On the other hand, in the work by the authors of [64], a novel ANFIS HW architecture, able to reduce the timing mark of 530 ns achieved in the present work more than 50%, was presented. Recently, several innovative architectures on other ML algorithms have been proposed with the aim of achieving extreme timing performance results. Examples of these innovations are a HW implementation of a radial basis function (RBF) network, for which operational frequencies of up to 450 MHz for high bit-width inputs were achieved [65], and an SVM implementation able to be run up to 20 times faster than other state-of-the-art techniques [66].

Consequently, the hybrid HW/SW implementation developed in this work is an innovative solution between conventional SW-based approaches and novel FPGA-based, extreme performance architectures, which, provides an adequate trade-off between complexity, performance, and development time.

5.2.3. ACC Personalization Application

The particular example of ANFIS Cluster 1 is displayed in Figure 11. In this figure, each column depicts the membership functions for each input of the ANFIS system (THW^0_{RMS}, $TETH^0$, and $TITH^0$, and output y, respectively), whereas each row corresponds to a fuzzy rule. Thus, for the selected example, with $THW^0_{RMS} = 0$, $TETH^0 = 0.5$, and $TITH^0 = 0.18$, ANFIS Cluster 1 returned an output value $y = 0.965$. Since this value was close to 1, and the input data fulfilled the description of Cluster 1 in Section 4.1, this ANFIS correctly identified this value as a member of the cluster it modeled. Additionally, this datapoint was input to the ANFIS of Clusters 2 and 3, with returning output values of 0.017 and 0.31, respectively. As a result, considering the maximum output value of the three ANFIS, the system successfully classified the inputs as Cluster 1.

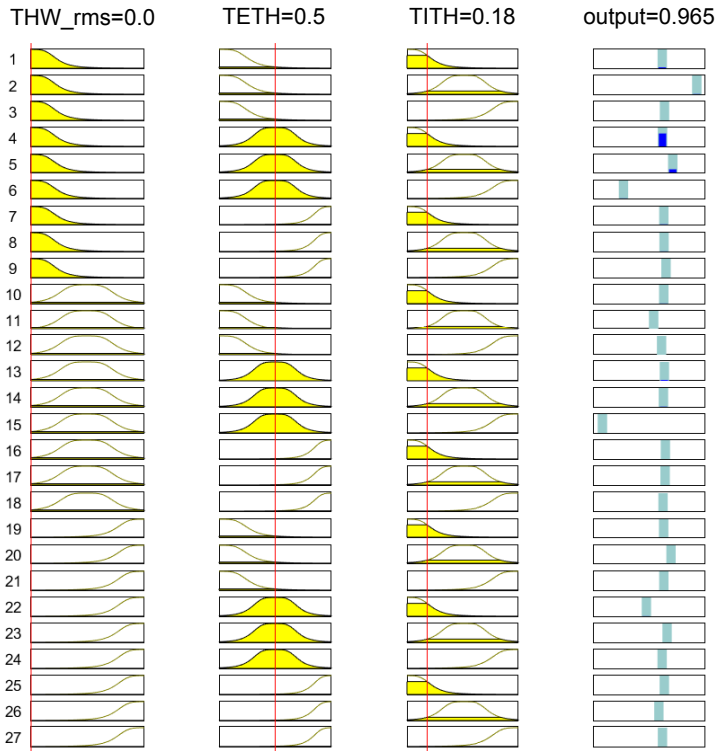

Figure 11. Rules and membership functions of ANFIS Cluster 1 for a given input.

Regarding ANFIS HW accelerator verification, in Figure 12, a simulation of the ANFIS Cluster 1 HW accelerator is depicted. As can be seen in this figure, with the same input values, the system returned output $y = 0.958$. The results obtained with the HW accelerator agree with Figure 11.

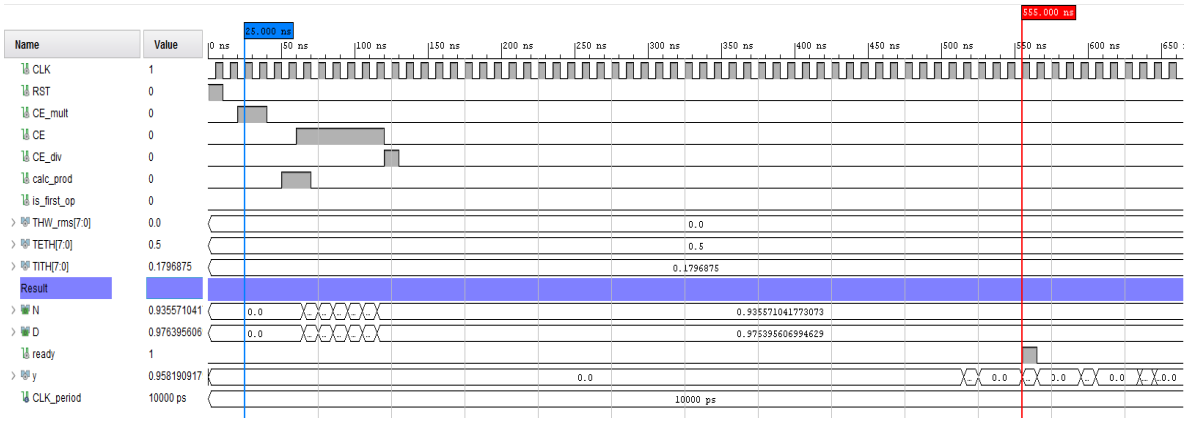

Figure 12. Simulation results of the ANFIS Cluster 1 HW accelerator obtained with the Vivado design suite.

The outputs of the three ANFIS clusters were recovered through the AXI4 bus by the software programmed in the PS partition of the PSoC. The PS determined which the highest recovered value was, hence identifying the corresponding cluster.

Individual-Based Personalization for ACC ADAS

Additionally, the software partition in the PS implemented a plane-shaped THW model for each cluster. Thus, given three clusters $1 \leq i \leq 3$, the ith plane was defined, such that

$$\widehat{THW}_i = f\left(\overline{THW}_{RMS_i}, TITH_i\right), \quad (9)$$

where $\widehat{THW}_i$ is the individualized THW adjustment, $\overline{THW}_{RMS_i}$ is the average THW_{RMS} value observed during the learning period of the system for a particular driver in a steady car-following situation, and $TITH_i$ is the normalized TITH value for the same period.

These planes were defined by the three-point method depending on the minimal, maximal, and average values for THW_{RMS} and TITH for each cluster according to Figure 6. With these distributions, the $\overline{THW}_{RMS_i}$ and σ_i for each cluster were:

- Cluster 1: $\overline{THW}_{RMS_1} = 1.08$ s, with $\sigma_1 = 0.27$ s.
- Cluster 2: $\overline{THW}_{RMS_2} = 2.44$ s, with $\sigma_2 = 0.59$ s.
- Cluster 3: $\overline{THW}_{RMS_3} = 1.61$ s, with $\sigma_3 = 0.17$ s.

For each cluster, the point of minimum THW_{RMS} and maximum TITH were assigned with a value of $\widehat{THW}_{RMS_i} = \overline{THW}_{RMS_i} - \sigma_i$, as it corresponded to drivers from that cluster who like to drive with a shorter time gap. On the other hand, drivers who would rather drive with longer time gaps (that is, those who are represented by the point of maximum THW_{RMS} and minimum TITH), have a value of $\widehat{THW}_{RMS_i} = \overline{THW}_{RMS_i} + \sigma_i$ assigned. Finally, intermediate drivers (average values of THW_{RMS} and TITH), have a value of $\widehat{THW}_{RMS_i} = \overline{THW}_{RMS_i}$. Consequently, three points $(\overline{THW}_{RMS_i}, TITH_i, \widehat{THW}_i)$ that defined each THW-modeling plane i are as follows.

- $p_{i1} = (min(THW_{RMS_i}), max(TITH_i), \overline{THW}_{RMS_i} - \sigma_i)$
- $p_{i2} = (max(THW_{RMS_i}), min(TITH_i), \overline{THW}_{RMS_i} + \sigma_i)$
- $p_{i3} = (\overline{THW}_{RMS_i}, \overline{TITH}_i, \overline{THW}_{RMS_i})$

With these considerations, the planes modeling the individualized $\widehat{THW}_i$ for each cluster according to Equation (9) were computed and shown in Figure 13.

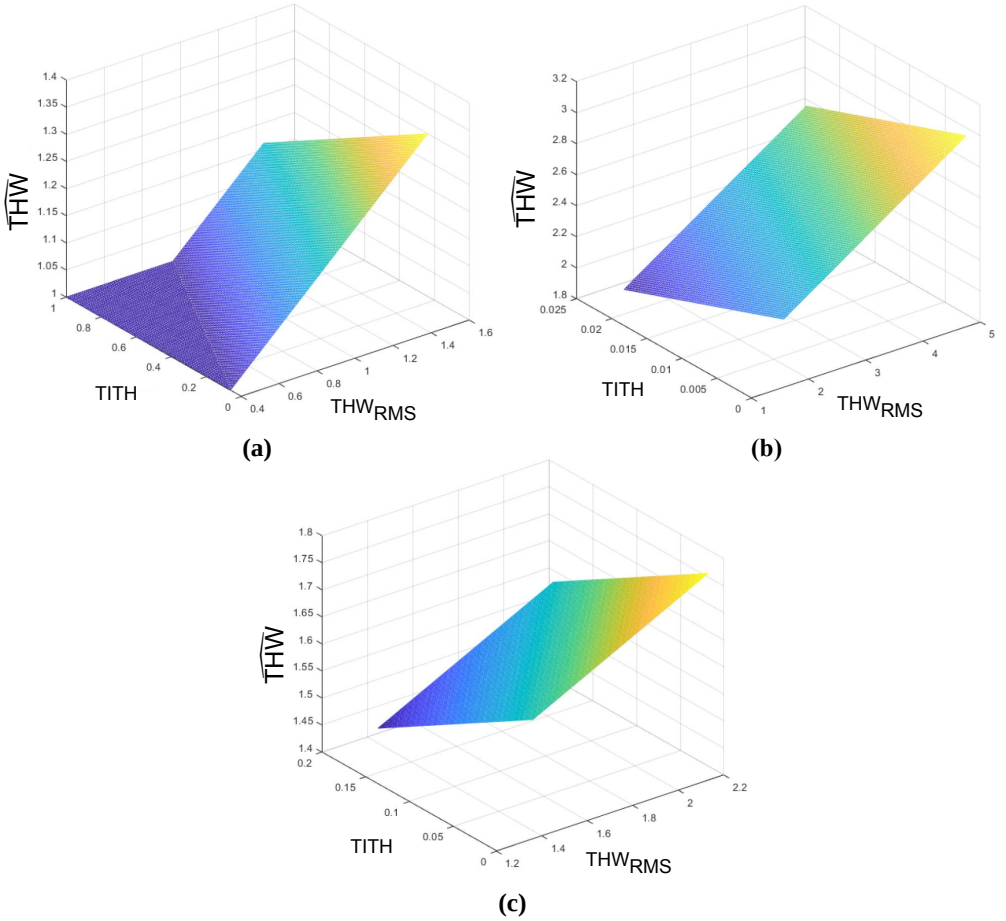

Figure 13. $\widehat{THW}_i$ model planes for (**a**) Cluster 1, (**b**) Cluster 2, and (**c**) Cluster 3.

As can be seen in Figure 13, for each of the models, the predicted $\widehat{THW}_i$ was directly proportional with $\overline{THW}_{RMS_i}$ and inversely proportional with TITH. Note that, for the Cluster 1 model, the plane was saturated to $\widehat{THW}_i = 1$ s to assure that the personalized THW value never took a value lower than the minimal safe THW values [16].

In sum, once the ANFIS accelerator identified the cluster for a given driver, one of the three models in Figure 13 was selected. Then, an individualization stage measured and computed the THW_{RMS} and TITH during a steady car-following period. Finally, with those measurements, the system evaluated the corresponding plane model and set a personalized THW value for the ACC system (see Figure 2).

6. Concluding Remarks

In this work, a machine learning approach to face the challenges of ADAS personalization was proposed. It is based on a hybrid personalization strategy for driving style modeling that uses a group-based clustering technique, namely, k-means clustering with an individual-based model that adapts the parameters of the clusters to an individual driver. This solution introduces personalization strategies that need no driver intervention with the aim of easing the use of ADAS and thus promoting their adoption in daily driving, with the ultimate goal of increasing road safety and reducing traffic accidents. The driving style clusters developed in this piece of research are representative of car-following behavior obtained with a meaningful sample of drivers in different kinds of roads, weather conditions, and lighting. Nevertheless, they can easily be extended to account for the requirements of particular groups of drivers, mainly the most vulnerable drivers (e.g., elderly or inexpert drivers). In addition, a similar approach could be used to personalize and improve current ADAS through different spotlights, such as the fuel economy or passenger comfort.

The implementation of a single-chip driving personalization system for in-car integration requires a high-speed clustering model. The solution adopted in this work relied on high-performance approximation of the clusters using an ANFIS. The universal approximation capability of ANFIS with its inherently parallelizable layered topology make this model suitable for efficient hardware implementation. The whole neuro-fuzzy sensor was successfully implemented using an FPGA device of a Xilinx Zynq-7000 PSoC providing high speed and low-power consumption for real-time ADAS implementation. In addition, due to the reconfigurable nature of FPGAs, both the hardware and the software partition of the PSoC could be updated to cope with the continuous changes that new vehicle technologies introduce.

In future works, neuro-fuzzy sensor capabilities will be enhanced by broadening the diversity of car-following scenarios. Both acceleration and braking will be analyzed. Moreover, a finer clustering approach will be investigated with the aim of categorizing driving scenarios according to, among others, weather conditions or lighting.

Author Contributions: All authors contributed equally to this work.

Funding: This work was supported in part by the Spanish AEI and European FEDER funds under Grant TEC2016-77618-R (AEI/FEDER, UE).

Conflicts of Interest: The authors declare no conflict of interest.

Disclaimer: The findings and conclusions of this paper are those of the authors and do not necessarily represent the views of VTTI, the Transportation Research Board, or the National Academies.

References

1. Coutinho, M.G.F.; Torquato, M.F.; Fernandes, M.A.C. Deep Neural Network Hardware Implementation Based on Stacked Sparse Autoencoder. *IEEE Access* **2019**, *7*, 40674–40694. [CrossRef]
2. Bengler, K.; Dietmayer, K.; Farber, B.; Maurer, M.; Stiller, C.; Winner, H. Three Decades of Driver Assistance Systems: Review and Future Perspectives. *IEEE Intell. Trans. Syst. Mag.* **2014**, *6*, 6–22. [CrossRef]
3. Leiber, H.; Czinczel, A.; Anlauf, J. Antiblockiersystem (ABS) für Personenkraftwagen. *BOSCH TECH BER* **1980**, *2*, 65–94.

4. Bleckmann, H.W.; Fennel, H.; Gräber, J.; Seibert, W.W. *Traction Control System with Teves ABS Mark II*; Technical Report, SAE Technical Paper; SAE: Warrendale, PA, USA, 1986.
5. Farmer, C.M. Effect of electronic stability control on automobile crash risk. *Traffic Inj. Prev.* **2004**, *5*, 317–325. [CrossRef] [PubMed]
6. National Highway Traffic Safety Administration (NHTSA), U.S. Department of Transportation. *49 CFR 571.126—Standard No. 126*; Electronic Stability Control Systems; SAE: Warrendale, PA, USA, 2008.
7. Council of European Union. *REGULATION (EC) No 661/2009*; Council of European Union: Brussels, Belgium, 2009.
8. Smiley, A. Behavioral adaptation, safety, and intelligent transportation systems. *Trans. Res. Rec.* **2000**, *1724*, 47–51. [CrossRef]
9. Markvollrath; Schleicher, S.; Gelau, C. The influence of Cruise Control and Adaptive Cruise Control on driving behaviour—A driving simulator study. *Accid. Anal. Prev.* **2011**, *43*, 1134–1139. [CrossRef] [PubMed]
10. Feilhauer, M.; Haering, J.; Wyatt, S. Current approaches in HiL-based ADAS testing. *SAE Int. J. Commer. Veh.* **2016**, *9*, 63–69. [CrossRef]
11. Ho, C.; Reed, N.; Spence, C. Multisensory in-car warning signals for collision avoidance. *Hum. Factors* **2007**, *49*, 1107–1114. [CrossRef]
12. Mahajan, R.N.; Patil, A. Lane departure warning system. *Int. J. Eng. Technol. Res.* **2015**, *3*, 120–123.
13. Wali, S.B.; Hannan, M.A.; Hussain, A.; Samad, S.A. Comparative survey on traffic sign detection and recognition: A review. *Przegląd Elektrotechniczny* **2015**, *91*, 38–42. [CrossRef]
14. Caber, N.; Langdon, P.; Clarkson, P.J. Designing Adaptation in Cars: An Exploratory Survey on Drivers' Usage of ADAS and Car Adaptations. In *International Conference on Applied Human Factors and Ergonomics*; Springer: Berlin, Germany, 2019; pp. 95–106.
15. Murray, L. Inside the future cars [Technology Driverless Cars]. *Eng. Tech.* **2017**, *12*, 60–62. [CrossRef]
16. Fleming, J.M.; Allison, C.K.; Yan, X.; Stanton, N.A.; Lot, R. Adaptive driver modelling in ADAS to improve user acceptance: A study using naturalistic data. *Saf. Sci.* **2018**. [CrossRef]
17. Beggiato, M.; Pereira, M.; Petzoldt, T.; Krems, J. Learning and development of trust, acceptance and the mental model of ACC. A longitudinal on-road study. *Trans. Res. Part F Traffic Psychol. Behav.* **2015**, *35*, 75–84. [CrossRef]
18. Panou, M.C. Intelligent personalized ADAS warnings. *Eur. Trans. Res. Rev.* **2018**, *10*, 59. [CrossRef]
19. Piccinini, G.F.B.; Rodrigues, C.M.; Leitão, M.; Simões, A. Driver's behavioral adaptation to Adaptive Cruise Control (ACC): The case of speed and time headway. *J. Saf. Res.* **2014**, *49*, 77.e1–84.
20. Hasenjäger, M.; Wersing, H. Personalization in advanced driver assistance systems and autonomous vehicles: A review. In Proceedings of the 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), Yokohama, Japan, 16–19 October 2017; pp. 1–7.
21. Dörr, D.; Grabengiesser, D.; Gauterin, F. Online driving style recognition using fuzzy logic. In Proceedings of the 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), Qingdao, China, 8–11 October 2014; pp. 1021–1026.
22. Bifulco, G.N.; Simonelli, F.; Di Pace, R. Experiments toward an human-like Adaptive Cruise Control. In Proceedings of the 2008 IEEE Intelligent Vehicles Symposium, Eindhoven, The Netherlands, 4–6 June 2008; pp. 919–924.
23. Bifulco, G.N.; Pariota, L.; Simonelli, F.; Di Pace, R. Development and testing of a fully adaptive cruise control system. *Trans. Res. Part C Emerg. Tech.* **2013**, *29*, 156–170. [CrossRef]
24. Wang, J.; Zhang, L.; Zhang, D.; Li, K. An adaptive longitudinal driving assistance system based on driver characteristics. *IEEE Trans. Intell. Trans. Syst.* **2012**, *14*, 1–12. [CrossRef]
25. Lefèvre, S.; Carvalho, A.; Gao, Y.; Tseng, H.E.; Borrelli, F. Driver models for personalised driving assistance. *Veh. Syst. Dyn.* **2015**, *53*, 1705–1720. [CrossRef]
26. Muehlfeld, F.; Doric, I.; Ertlmeier, R.; Brandmeier, T. Statistical behavior modeling for driver-adaptive precrash systems. *IEEE Trans. Intell. Trans. Syst.* **2013**, *14*, 1764–1772. [CrossRef]
27. Rosenfeld, A.; Bareket, Z.; Goldman, C.V.; LeBlanc, D.J.; Tsimhoni, O. Learning drivers' behavior to improve adaptive cruise control. *J. Intell. Trans. Syst.* **2015**, *19*, 18–31. [CrossRef]
28. Canale, M.; Malan, S.; Murdocco, V. Personalization of ACC Stop and Go task based on human driver behaviour analysis. *IFAC Proc. Vol.* **2002**, *35*, 357–362. [CrossRef]

29. De Gelder, E.; Cara, I.; Uittenbogaard, J.; Kroon, L.; van Iersel, S.; Hogema, J. Towards personalised automated driving: Prediction of preferred ACC behaviour based on manual driving. In Proceedings of the 2016 IEEE Intelligent Vehicles Symposium (IV), Gothenburg, Sweden, 19–22 June 2016; pp. 1211–1216, [CrossRef]
30. SHRP 2—Strategic Highway Research Program 2 (SHRP 2). Available online: http://www.trb.org/StrategicHighwayResearchProgram2SHRP2/Blank2.aspx (accessed on 29 June 2019).
31. Xilinx. *Zynq-7000 SoC Data Sheet: Overview (DS190), v1.11.1*; Xilinx: San José, CA, USA, 2018.
32. Meiring, G.; Myburgh, H. A review of intelligent driving style analysis systems and related artificial intelligence algorithms. *Sensors* **2015**, *15*, 30653–30682. [CrossRef] [PubMed]
33. Abut, H.; Erdoğan, H.; Erçil, A.; Çürüklü, B.; Koman, H.C.; Taş, F.; Argunşah, A.Ö.; Coşar, S.; Akan, B.; Karabalkan, H.; et al. Real-world data collection with "UYANIK". In *In-Vehicle Corpus and Signal Processing for Driver Behavior*; Springer: Berlin, Germany, 2009; pp. 23–43.
34. Angkititrakul, P.; Petracca, M.; Sathyanarayana, A.; Hansen, J.H. UTDrive: Driver behavior and speech interactive systems for in-vehicle environments. In Proceedings of the 2007 IEEE Intelligent Vehicles Symposium, Istanbul, Turkey, 13–15 June 2007; pp. 566–569.
35. Regan, M.; Williamson, A.; Grzebieta, R.; Tao, L. Naturalistic driving studies: Literature review and planning for the Australian naturalistic driving study. In Proceedings of the Australasian College of Road Safety Conference 2012, Sydney, Australia, 9–10 August 2012.
36. Eenink, R.; Barnard, Y.; Baumann, M.; Augros, X.; Utesch, F. UDRIVE: The European naturalistic driving study. In Proceedings of the IFSTTAR Transport Research Arena, Paris, France, 14–17 April 2014.
37. Neale, V.L.; Dingus, T.A.; Klauer, S.G.; Sudweeks, J.; Goodman, M. An overview of the 100-car naturalistic study and findings. *Natl. Highw. Traffic Saf. Adm.* **2005**, *5*, 0400.
38. Dingus, T.A.; Hankey, J.M.; Antin, J.F.; Lee, S.E.; Eichelberger, L.; Stulce, K.E.; McGraw, D.; Perez, M.; Stowe, L. *Naturalistic Driving Study: Technical Coordination and Quality control*; Number SHRP 2 Report S2-S06-RW-1; TRID: Washington, DC, USA, 2015.
39. Martínez, V.; del Campo, I.; Echanobe, J.; Basterretxea, K. Driving behavior signals and machine learning: A personalized driver assistance system. In Proceedings of the 2015 IEEE 18th International Conference on Intelligent Transportation Systems, Las Palmas, Spain, 15–18 September 2015; pp. 2933–2940.
40. del Campo, I.; Asua, E.; Martínez, V.; Mata-Carballeira, Ó.; Echanobe, J. Driving Style Recognition based on Ride Comfort Using a Hybrid Machine Learning Algorithm. In Proceedings of the IEEE 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 4–7 November 2018; pp. 3251–3258.
41. Murphey, Y.L.; Milton, R.; Kiliaris, L. Driver's style classification using jerk analysis. In Proceedings of the 2009 IEEE Workshop on Computational Intelligence in Vehicles and Vehicular Systems, Nashville, TN, USA, 30 March–2 April 2009; pp. 23–28.
42. Miyajima, C.; Nishiwaki, Y.; Ozawa, K.; Wakita, T.; Itou, K.; Takeda, K.; Itakura, F. Driver modeling based on driving behavior and its evaluation in driver identification. *Proc. IEEE* **2007**, *95*, 427–437. [CrossRef]
43. Wang, J.; Li, K.; Lu, X.Y. Chapter 6—Comparative Analysis and Modeling of Driver Behavior Characteristics. In *Advances in Intelligent Vehicles*; Chen, Y., Li, L., Eds.; Academic Press: Boston, MA, USA, 2014; pp. 159–198. [CrossRef]
44. Higgs, B.; Abbas, M. Segmentation and clustering of car-following behavior: Recognition of driving patterns. *IEEE Trans. Intell. Trans. Syst.* **2014**, *16*, 81–90. [CrossRef]
45. Custer, K.; Sudweeks, J.; Perez, M.A.; del Campo, I.; Echanobe, J.; Martinez, V.; Asua, E.; Basterretxea, K.; Bosque, G.; Martinez, U.; et al. *PSoC for Real-Time Driver Assistance Based on Machine Learning IP Cores*; Dataset, SHRP2 Naturalistic Driving Study; VTTI: Blacksburg, VA, USA, 2019;
46. Castignani, G.; Derrmann, T.; Frank, R.; Engel, T. Driver behavior profiling using smartphones: A low-cost platform for driver monitoring. *IEEE Intell. Trans. Syst. Mag.* **2015**, *7*, 91–102. [CrossRef]
47. Shen, J.; Hao, X.; Liang, Z.; Liu, Y.; Wang, W.; Shao, L. Real-time superpixel segmentation by DBSCAN clustering algorithm. *IEEE Trans. Image Process.* **2016**, *25*, 5933–5942. [CrossRef]
48. Moon, T.K. The expectation-maximization algorithm. *IEEE Signal Process. Mag.* **1996**, *13*, 47–60. [CrossRef]
49. Zhao, Y.; Karypis, G. Evaluation of hierarchical clustering algorithms for document datasets. In Proceedings of the Eleventh International Conference on Information and Knowledge Management (ACM–CIKM'02): McLean, VA, USA, 4–9 November 2002; pp. 515–524.

50. Kalsoom, R.; Halim, Z. Clustering the driving features based on data streams. In Proceedings of the IEEE INMIC, Lahore, Pakistan, 19–20 December 2013; pp. 89–94.
51. Bhoraskar, R.; Vankadhara, N.; Raman, B.; Kulkarni, P. Wolverine: Traffic and road condition estimation using smartphone sensors. In Proceedings of the IEEE 2012 Fourth International Conference on Communication Systems and Networks (COMSNETS 2012), Bangalore, India, 3–7 January 2012; pp. 1–6.
52. Brombacher, P.; Masino, J.; Frey, M.; Gauterin, F. Driving event detection and driving style classification using artificial neural networks. In Proceedings of the 2017 IEEE International Conference on Industrial Technology (ICIT), Toronto, ON, Canada, 22–25 March 2017; pp. 997–1002.
53. Kurt, A.; Özgüner, Ü. A probabilistic model of a set of driving decisions. In Proceedings of the IEEE 2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC), Washington, DC, USA, 5–7 October 2011; pp. 570–575.
54. Zadeh, L.A. Fuzzy logic. *Computer* **1988**, *21*, 83–93. [CrossRef]
55. Aljaafreh, A.; Alshabatat, N.; Al-Din, M.S.N. Driving style recognition using fuzzy logic. In Proceedings of the 2012 IEEE International Conference on Vehicular Electronics and Safety (ICVES 2012), Istanbul, Turkey, 24–27 July 2012; pp. 460–463.
56. Choudhary, A.K.; Ingole, P.K. Smart phone based approach to monitor driving behavior and sharing of statistic. In Proceedings of the IEEE 2014 Fourth International Conference on Communication Systems and Network Technologies, Bhopal, India, 7–9 April 2014; pp. 279–282.
57. Jang, J.S. ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Trans. Syst. Man Cybern.* **1993**, *23*, 665–685. [CrossRef]
58. Jang, J.S.; Sun, C.T. Neuro-fuzzy modeling and control. *Proc. IEEE* **1995**, *83*, 378–406. [CrossRef]
59. Xilinx. *ZC706 Evaluation Board for the Zynq-7000 XC7Z045 SoC (UG954), v1.7*; Xilinx: San José, CA, USA, 2018.
60. Xilinx. *7 Series DSP48E1 Slice (UG479), v1.10*; Xilinx: San José, CA, USA, 2018.
61. Mata-Carballeira, Ó.; del Campo, I.; Martínez, V.; Echanobe, J. A Hardware/Software Extreme Learning Machine Solution for Improved Ride Comfort in Automobiles. In Proceedings of the IEEE 2019 International Joint Conference on Neural Networks (IJCNN), Auckland, New Zealand, 27–30 October 2019.
62. Xilinx. *Divider Generator v5.1 (PG151)*; Xilinx: San José, CA, USA, 2016.
63. Saldaña, H.J.B.; Silva-Cárdenas, C. A digital hardware architecture for a three-input one-output zero-order ANFIS. In Proceedings of the 2012 IEEE 3rd Latin American Symposium on Circuits and Systems (LASCAS), Playa del Carmen, Mexico, 29 February–2 March 2012; pp. 1–4. [CrossRef]
64. Darvill, J.; Tisan, A.; Cirstea, M. A novel ANFIS algorithm architecture for FPGA implementation. In Proceedings of the 2017 IEEE 26th International Symposium on Industrial Electronics (ISIE), Edinburgh, UK, 19–21 June 2017; pp. 1243–1248. [CrossRef]
65. De Souza, A.; Fernandes, M. Parallel fixed point implementation of a radial basis function network in an FPGA. *Sensors* **2014**, *14*, 18223–18243. [CrossRef] [PubMed]
66. Lopes, F.F.; Ferreira, J.C.; Fernandes, M.A. Parallel Implementation on FPGA of Support Vector Machines Using Stochastic Gradient Descent. *Electronics* **2019**, *8*, 631. [CrossRef]