Ó. Mata-Carballeira, I. del Campo, V. Martínez and J. Echanobe, "*A Hardware/Software Extreme Learning Machine Solution for Improved Ride Comfort in Automobiles,"* **2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 2019**, paper N-20134 https://doi.org/10.1109/IJCNN.2019.8852435 © 2019 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

A Hardware/Software Extreme Learning Machine Solution for Improved Ride Comfort in Automobiles

Óscar Mata-Carballeira, Inés del Campo, Victoria Martínez, Javier Echanobe
Department of Electricity and Electronics
University of the Basque Country (UPV/EHU)
Leioa, Basque Country, Spain
oscar.mata@ehu.eus

***Abstract*—Automotive ride comfort has become an important research topic in recent years due to the increasing level of automation in currently produced cars. These premises also apply to manned cars. In this work, a hybrid hardware/software extreme learning machine for improved ride comfort in automobiles is proposed. This system is based on a single-chip implementation able to provide real-time information about the level of ride comfort by classifying driving data into several comfort classes. To develop this system, unsupervised hierarchical clustering analysis (HCA) and supervised extreme learning machine (ELM) have been used jointly, to enhance the overall performance of the entire system, reaching classification success rates of up to 95%. This approach has been implemented on a Xilinx Zynq-7000 programmable system-on-chip. This chip is able to process data in real time and to identify the comfort class, achieving low latency marks and high operational frequencies due to its DSP-based implementation. These performance and accuracy marks, together with its low power consumption make this development suitable for novel practical implementations in current production cars.**

***Index Terms*—ride comfort, hardware, software, PSoC, ELM, HCA, DSP.**

I. Introduction

The classification and identification of how motorists drive (driving style) is gaining attention, particularly in the field of customizable advanced driving assistance systems (ADAS) and, in general, in highly automated vehicles. This growing interest has been motivated by the interest in classify driving behavior with the aim of improving the style of human drivers. Personalizing the attitude of the already existing ADAS to make them more comfortable and attractive to encourage users for their extensive use is a field of interest as well. To carry out driving style characterization, several approaches can be undertaken, such as rule-based solutions, model-based approaches and machine-learning-based techniques [1]. Most of these solutions are focused on road safety and energy efficiency [2]. Throughout this article, the spotlight is put on machine-learning-based implementations, combining unsupervised and supervised algorithms to classify the driving style of drivers depending on the global ride comfort they provide. This work proposes a single-chip implementation, suitable for in-car integration, focused on an innovative viewpoint: the ride comfort of the driver and the passengers.

There exist many parameters that contribute to the global ride comfort in the same manner that the driver does. These parameters are derived from both the environment within the vehicle (e.g. smell, temperature, humidity, etc.), as well as from the characteristics of the passenger (e.g. gender, age, health conditions, etc.) along with his/her behavior (keeping or not his/her gaze on the road). Nevertheless, the parameter that compromises the overall comfort perception the most is vibrations. Several pieces of research link perceived distress, motion sickness and the frequency components of the vibrations with the resonance frequencies of the organs of the human body [3], [4]. Although some vibrations are caused by the constructive characteristics of either the roadway or the vehicle itself including the handling of the automobile, the most comfort compromising ones are those derived from the driving behavior and handling skills of the driver. In that sense, if the driving style (DS) of the motorist could be correctly identified, various actions could be executed in order to influence the driver's behavior resulting in the minimization of the vibrations originated from the handling itself, improving the passengers' comfort.

In ride comfort terms, there are three possible landscapes pertaining to the forecasted evolution of car mobility introduced by the Society of Automotive Engineers (SAE) [5]: Firstly, in the short term (and current one), on which cars are still manned (SAE automation levels 0-1), the main objective of ride-comfort intended driving style characterization is to recommend the driver to modify certain welfare-compromising behaviors; secondly, in the medium and near future term (SAE automation levels 2-4), with semi-automated cars, the aim is setting comfortable automated behaviors or even personalizing ADAS to mimic the style of the human driver; finally, on the long-term horizon of fully autonomous cars (SAE automation level 5) in which drivers will switch their role to turn into mere passengers, the importance of achieving a fairly high global ride comfort level is even more important than the standard in the current paradigm of manned automobiles.

In this work, a single-chip implementation of a driving style (DS) hardware/software (HW/SW) identification system is proposed. This system is able to provide real-time information about the level of ride comfort during a road trip with the aim

This work was supported in part by the Spanish AEI and European FEDER funds under Grant TEC2016-77618-R (AEI/FEDER, UE) and by the University of the Basque Country (UPV/EHU) under Grant PPG17/20.

Accepted Manuscript

of warning the driver every time the passengers' comfort is compromised. Automotive applications must be power efficient and secure enough for safety-critical functions, fields in which FPGAs stand out. These devices provide higher power consumption/speed ratio than other computational alternatives, and the speeds that can be reached when an adequate design is implemented within them allow their integration in higher autonomy-level systems. Additionally, their reconfigurability enables the possibility of further hardware updates with no modifications in the logic boards of the vehicle, with the cost efficiency this implies in such a constantly evolving field. A step further from FPGAs are programmable system-on-chips (PSoCs), which, in addition to an FPGA, rely on a multi-core microprocessor to execute sequential tasks (i.e. HW/SW solution).

The architecture of the above chip is based on a DS characterization model previously developed by the authors [6], where unsupervised and supervised machine learning methods have been used jointly to enhance the overall performance of the system. Notably, the unsupervised hierarchical clustering analysis (HCA) algorithm has been used to discover patterns of driving data (DS classes or styles) according to the ride comfort perspective. Afterwards, a supervised extreme learning machine (ELM) has been trained to model the DS system. The ELM scheme has been selected owing to its suitability for true real-time learning and adaptation. Regarding training, ELMs present better learning performance and speed due to the absence of local minima and reduced number of parameters. This approach has been proven to be successful, and, with an accuracy of up to the 95% [6], it is suitable to be ported to a real world automotive application.

The remainder of this work will be structured as follows: Section II will present the main concepts on ride comfort assessment while the ride-comfort Related Driving Styles will be explained in section III; section IV will focus on describing the proposed DS recognition system and implementation results, while the conclusions and future works will be clarified in section V.

II. Ride Comfort

Vibration-originated discomfort happens is caused by different features [7]: magnitude, frequency, direction and duration of accelerations. Hence, the quality of the ride must be assessed taking into account both discrete events (such as abrupt lane changes) and the average motion of the vehicle (which, if sustained during a long period of time, can cause discomfort and, eventually, motion sickness).

This discomfort can be determined by the maximum values of acceleration and jerk (time derivative of acceleration), that usually arise at aggressive lane changes, while entering and leaving corners or while braking [8]. High values of acceleration and jerk can cause discomfort even when they occur during very short periods of time. Besides transient discomfort produced by punctual acceleration and jerk, sustained, low-frequency motion is the main contributor to motion sickness, while high-frequency motion causes stress. High frequencies rarely cause motion sickness.

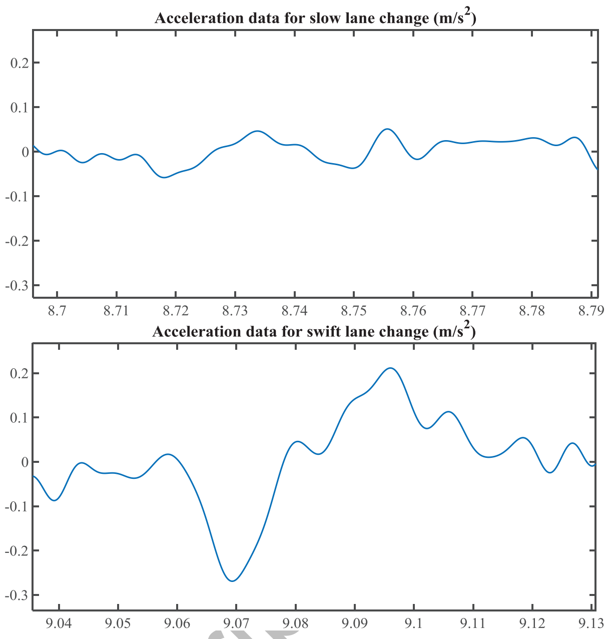

Fig. 1. Lateral accelerations in slow (top) and swift (bottom) lane changes.

To illustrate these facts, a good example could be driving behavior in lane changes. From a variety of trajectories, slow and swift lane changes are chosen as extreme cases of total comfort and total discomfort. As can be seen in Fig. 1, in contrast with the slow lane change, in the swift lane change high amplitude transient lateral accelerations happen, resulting in high jerk values likely to rapidly cause motion sickness in occupants.

Nevertheless, besides this qualitative analysis, these comfort conditionants must be quantified so as to limit them to acceptable levels. Perception of vibrations in the human body depends on the direction and the spectral components of each signal, therefore, the power spectral density (PSD) of each signal is a powerful tool to assess ride quality [3], [4]. The PSD of a signal provides the power of a signal by unity of frequency. In most applications, a good approximation is the squared magnitude of the Fourier transform,

$$PSD_{sum} = 2\sum_{i=1}^{M/2}(re(X(i))^2 + im(X(i))^2), \qquad (1)$$

where M is the number of samples of the data subset which the Fourier transform was applied to, and X the output coefficients of the Fourier transform.

To unify the quantification criteria, the International Organization for Standardization (ISO) 2631-1 [9] defines several methods to measure vibrations as well as to process data to standardized performance measures concerning health, perception, comfort and motion sickness. The quantified performance measures of ISO 2631 are based on frequency-weighted root

Accepted Manuscript

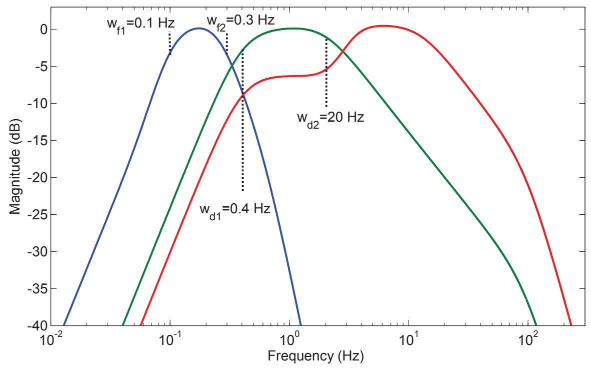

Fig. 2. Amplitude responses of different weighting filters in ISO 2631, w_f: motion sickness (blue), w_d: global comfort horizontal-component (green), and w_k: global comfort vertical-component (red).

mean square (RMS) computations of acceleration data in each axis.

The ISO norm defines several filters that delimit the frequency bands where different components of discomfort are present: filters w_f, w_d, and w_k, where filter w_f is representative of motion sickness discomfort, while the filters w_d and w_k model the horizontal and vertical components of global discomfort, respectively. As can be see in Fig. 2 the most critical frequencies for motion sickness are those occurring in the range from 0.1 to 0.3 Hz, while the ones for global discomfort in the horizontal component can be found between 0.4 and 20 Hz.

III. Ride-Comfort Related Driving Styles

Several approaches can be developed taking into account the PSD and the RMS components considered in ISO 2631 so as to identify how comfortable the DS of a given motorist is. Unsupervised machine learning algorithms are very useful for carrying out this identification task by performing statistical analysis on the input signals. The most remarkable algorithms able to perform identification tasks are Gaussian mixture models (GMMs) [10] in car-following tasks and in pedal operation-based identification. On the other hand, principal component analysis (PCA) is an interesting method for reducing the dimensionality of the input data, as well as for carrying out extensive analyses of the driving-related features extracted by models such as GMMs [11]. Additionally, dynamic time wrapping and hierarchical Bayesian regression models are used to detect aggressive events and to characterize driving style at roundabouts [12], respectively, while hierarchical clustering analysis (HCA) can be used to identify groups of drivers based on similarities in their driving statistics. In the following sections, both the unsupervised HCA and the supervised ELM used in the DS identification chip will be summarized.

A. Unsupervised HCA-based Driving Style Identification

HCA is a bottom-up approach that iteratively measures the distance between any two clusters (initially single drivers) merging the two closest ones in each step. The measurement

TABLE I
Driving style recognition: sensors, signals and features.

Sensors and units	Signals (time series: 32 Hz sample rate)	Time (RMS)	Frequency (PSD sum)
CAN bus	SWA: Steering Wheel Angle	1	8
	SWS: Steering Wheel Speed	2	9
	VS: Vehicle Speed	3	10
	PGP: Percent gas pedal	4	11
	ERPM: Engine RPM	5	12
Pressure sensors	BP: Brake pedal pressure	6	13
	GP: Gas pedal pressure	7	14

NOTE: High level features are derived from analysis frames of 128 sec with an overlapping of 127 sec.

method in this case considers the Euclidean distance and follows a single-linkage, nearest neighbor algorithm [6]. This unsupervised clustering procedure generates a hierarchical structure in the form of a binary tree.

To carry out the development of the HCA-based identification algorithm, a set of real-world driving data has been used, concretely, the Sabançi University's UYANIK instrumented car dataset [13], [14]. This dataset contains several driving signals obtained from both the CAN-bus of the vehicle and an aftermarket inertial measurement unit (IMU), as well as pressure sensors on the brake and throttle pedals of the vehicle.

In order to identify driving styles taking into account the statements in ISO-2631, the HCA algorithm is applied on the PSDs of the XYZ axes' accelerations (since they summarize the spectral contents of the signals) obtained from frames of 128 seconds from several trips from the UYANIK's IMU with a sample rate of 32 Hz, resulting in data subsets of 4096 samples. After the application of the HCA algorithm on these data, three driver classes can be distinguished: *fairly comfortable*, *slightly uncomfortable* and *uncomfortable*.

B. Supervised ELM-based Driver Classification

In the previous section, driving data frames are grouped into three categories depending on how comfortably their motorists drive and these frames are labeled according to these categories. With these labeled data, the supervised training and testing of an ELM can be performed. Let us introduce the basics of ELM.

The ELM method assumes that the weights and biases of the hidden nodes of a single hidden-layer feedforward neural network (SLFN) are randomly generated [15]. The scheme of an ELM is displayed in Fig. 3. Thus, with L being the number of hidden nodes, the output of an ELM is calculated as follows:

$$f_L(\mathbf{x}) = \sum_{i=1}^{L} \beta_i h_i(\mathbf{x}) = \mathbf{h}(\mathbf{x})\beta, \tag{2}$$

Accepted Manuscript

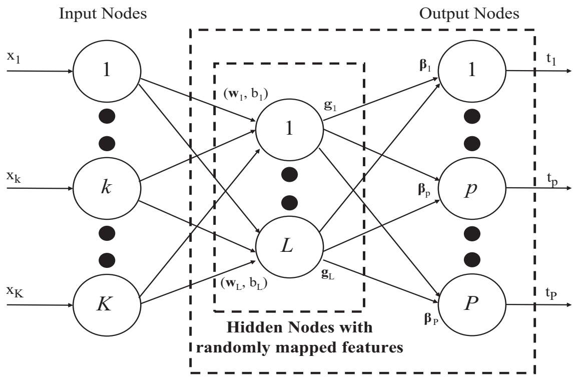

Fig. 3. ELMs are SLFNs with the particularity of having random weights and biases for the hidden layer.

with $\beta = [\beta_1, \ldots, \beta_L]^T$ the matrix of weights from the hidden layer to the output nodes; $\mathbf{h}(\mathbf{x}) = [g_1(\mathbf{x}), \ldots, g_L(\mathbf{x})]$ are the hidden node outputs (with random features) for the input $\mathbf{x}$; and given a number K of training samples, ELM is the solution of the following problem:

$$\mathbf{H}\beta = \mathbf{T}, \tag{3}$$

Defined a set of input and training arrays $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_K]^T$ and $\mathbf{T} = [\mathbf{t}_1, \ldots, \mathbf{t}_K]^T$ (where $\mathbf{t}_i$ are the target labels), the matrix of hidden nodes' outputs $\mathbf{H} = [\mathbf{h}^T(\mathbf{x}_1), \ldots, \mathbf{h}^T(\mathbf{x}_K)]^T$ is computed. Thus, since (3) is a System of Linear Equations, it can be solved in the subsequent manner:

$$\beta = \mathbf{H}^{\dagger}\mathbf{T}, \tag{4}$$

where $\mathbf{H}^{\dagger}$ is the Moore-Penrose generalized inverse of matrix $\mathbf{H}$.

The ELM trained in this work has the 14 driving features of Table I as inputs and the 3 driving classes as outputs. Several numbers of nodes for the hidden layer have been tested, while finally 100 hidden neurons were allocated, returning accuracy marks of up to 95% [6]. Consequently, the ELM topology to best deploy in the design is a 14-100-3 single hidden-layer feedforward neural network (SLFN) ELM.

IV. Development of the Driving Style Recognition System

A block diagram of the proposed DS recognition system is displayed in Fig. 4. It is a hybrid HW/SW implementation based on a PSoC. Thus, while the software part (allocated in the microprocessor) carries out the extraction of the RMS (time domain) and PSD sum (frequency domain), as well as the management of the data exchanges between the system and the CAN-bus of the vehicle, the hardware part accomplishes the ELM-based identification of driving styles based on those characteristics.

A. HW/SW architecture

To successfully implement the mixed HW/SW design of Fig. 4, the specific architecture of the selected PSoC must be taken into account in order to partition adequately the hardware and software sections and to communicate them correctly. In this case, a Xilinx Zynq-7000 device was selected. This Xilinx family relies on a dual-core ARM microprocessor with its peripherals (known as Processor System or PS) where the software will be run, and a full-size FPGA (known as Programmable Logic or PL) where the SLFN ELM will be deployed [16]. The PS is connected to the PL as well as its peripherals through a variant of the AMBA bus, known as AXI4.

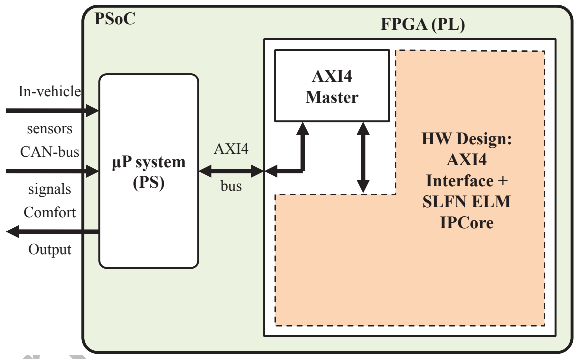

Fig. 4. Block diagram: development process of the hybrid DS characterization system for ride comfort classification.

B. Hardware Partition: SLFN ELM implementation

The hardware partition's top-bottom hierarchy, deployed within the PSoC's FPGA, can be described as follows:

The building blocks on which the SLFN ELM HW implementation relies are structured mainly according to the ELM diagram in Fig. 3 with the only difference that the sums of products in the hidden neurons have been swapped by multiplication-accumulation operations for hardware economy reasons. Thus, the HW development relies on as many hidden neuron and output neuron blocks as hidden nodes and output nodes that the 14-100-3 architecture has (100 hidden and 3 output nodes). This configuration choice allows us to achieve minimum latency marks, improving the overall performance. This SLFN ELM HW implementation has been fitted with input/output registers and an AXI4 interface to allow it to communicate with the PS. It has also been correctly wrapped to form a ready-to-use SLFN ELM IPcore. This SLFN ELM IPcore is depicted as an orange block in Fig 5 along with the PS and the AXI4 master. Comparing blocks of Fig. 5 and 4, we can see that the orange blocks in both figures match each other.

1) Hidden Neuron top-level module: The artificial, fixed-point-arithmetic-based neuron connecting (according to the indicated top-bottom hierarchy) a multiplier-accumulator (MAC) module and an Activation Function module is showed in Fig. 6. This architecture has been chosen as the most adequate

Accepted Manuscript

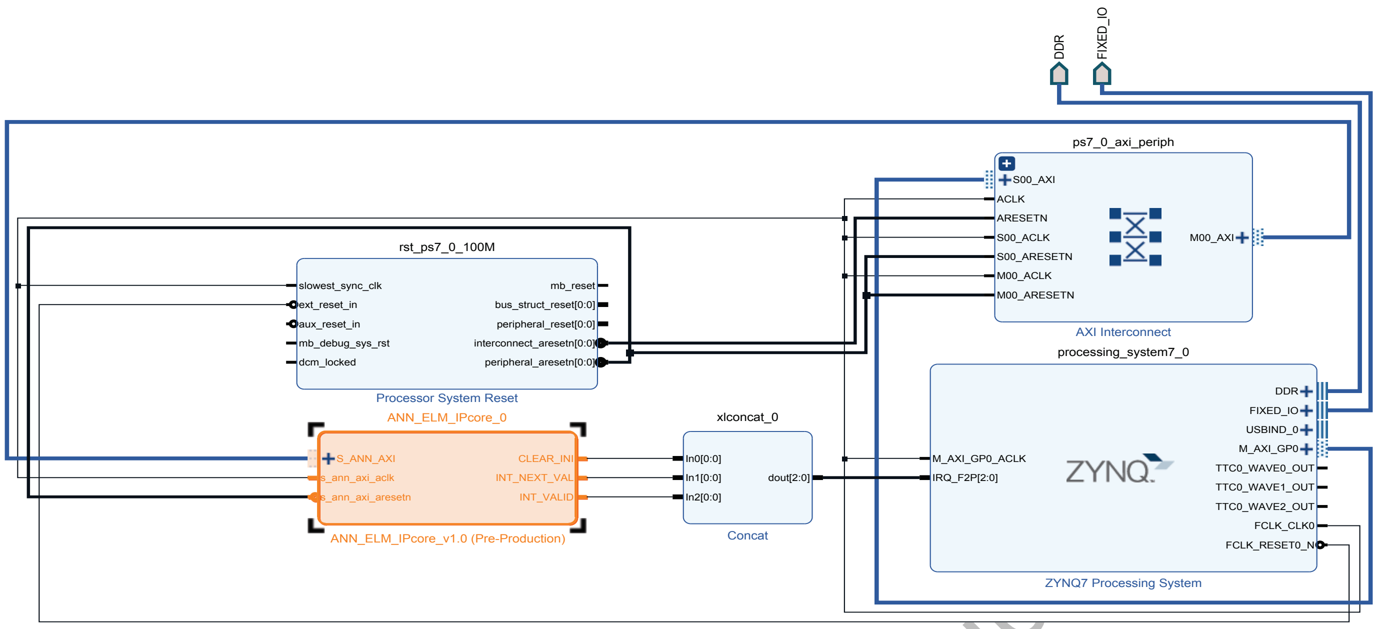

Fig. 5. AXI4 peripheral made from the SLFN ELM IP Core. The I/O flow is connected through the AXI port, while the outputs correspond to the interrupts.

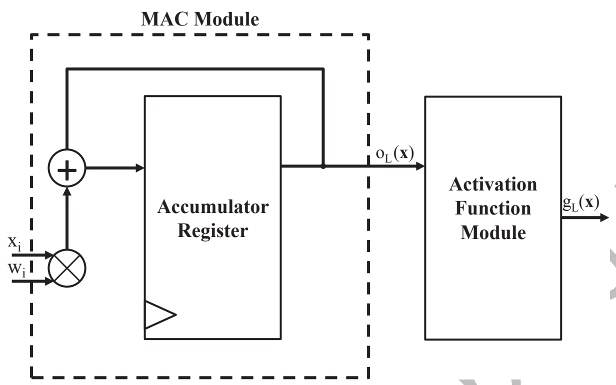

Fig. 6. RTL schematic of the complete, VHDL-parametrized fixed point hidden artificial neuron.

Accepted Manuscript

option since, if the input data, as well as the depth of the accumulator register are correctly sized, the MAC implementation only consumes 1 DSP block per hidden neuron, allowing extreme performance rates while keeping the count of utilized resources sufficiently low (digital signal processing (DSP) blocks are high-performance FPGA resources able to compute products, sums and accumulations in a very efficient way).

As can be seen in Fig. 6, the MAC module is placed on the left side of the hidden neuron while the activation function assembly rests on the right-hand part. The MAC module operates as described in (5). Thus, given a set of L hidden neurons and K inputs, the output o^L of each hidden neuron's MAC is the sum of products of the values on each input x_i by its associated weight w_i^L for each hidden neuron:

$$o_L(\mathbf{x}) = \sum_{i=1}^{K} x_i w_i^L \tag{5}$$

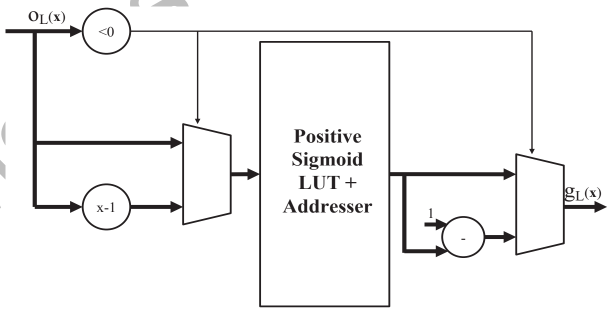

Fig. 7. Architecture of the sigmoid module.

Finally, the output g^L of each hidden neuron is calculated through the evaluation of the logistic function as described in (6)

$$g_L(\mathbf{x}) = g\left(o_L(\mathbf{x})\right) = \frac{1}{1+e^{-o_L(\mathbf{x})}} \tag{6}$$

This expression is evaluated by addressing the values of o^L from a pre-calculated logistic function look-up table (LUT). Nevertheless, in order to save resources, and taking into account that the logistic function has uneven symmetry, only the values of the positive domain of this function are stored, obtaining the outputs for the negative domain by calculating a symmetry in terms of (7),

$$g(x) = 1 - g(-x); x < 0 \tag{7}$$

The architecture to carry out the described logistic function evaluation is showed in Fig. 7.

From left to right, this module (in Fig. 7) operates as follows: The MAC output o_L is input to the module and

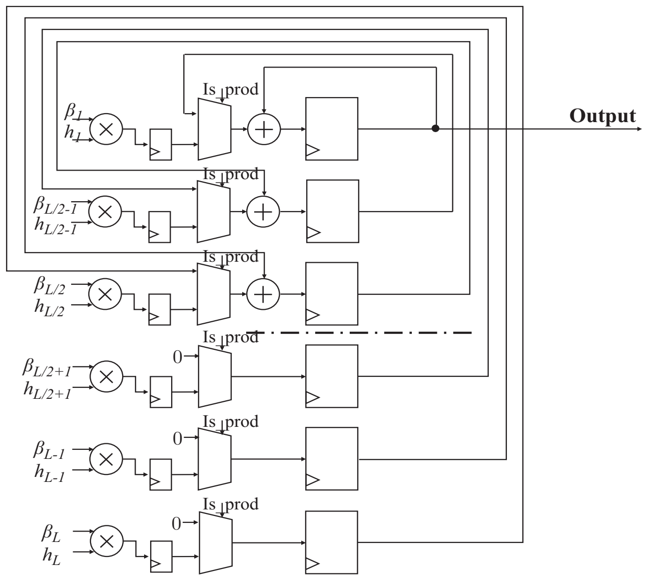

Fig. 8. Proposed alternative for the tree adder.

its sign is evaluated. Simultaneously, this o_L is two's complemented, obtaining $-o_{\tilde{N}}$ and both complemented and non-complemented values are input to a multiplexer actuated by the sign evaluation. This operation is performed in order to obtain at the output of the multiplexer an always-positive data since it is intended to be used to address the positive sigmoid LUT. Subsequently, with the always-positive MUX output, the contents of the LUT are properly addressed and its output is conducted both directly and additively inverted (performing the symmetry for the negative domain case) to the sigmoid output MUX, on which, depending on the sign of the input data, the LUT's output or its symmetric value are selected, achieving at the sigmoid output the desired value of $g_L = g(o_L)$.

2) Output Neurons: The output neurons of this SLFN implementation have been designed with the objectives of achieving the minimal possible latency while reaching top operational frequencies. Firstly, since output neurons are nothing but sums of products, we considered implementing a pipelined tree adder [17] due to its low latency (its output delays $L/2$ clock cycles to be calculated). However, this design uses $2L-1$ multiplier/adder blocks to perform at its minimum latency and it is not directly implementable by only using Xilinx DSP resources, so its performance cannot reach maximum levels. Hence, a new architecture (see Fig. 8) was developed with the aim of only using DSP resources while keeping low latency rates comparable to those achieved with the tree adder architecture.

This proposed architecture operates as follows:

- The product signal `is_prod` is set to "1" and all the registers are reset.
- The product $h_i \cdot \beta_i$ with $i = 1, \ldots, K$ is stored in each of the L accumulator registers.
- Signal `is_prod` is set back to "0" and the first accumulation is performed. Thus, the accumulating registers from 1 to $L/2$ contain the first sum of the equivalent tree adder. Registers from $L/2+1$ to L are now filled with zeros.
- Successive $\lceil log_2 L \rceil$ accumulations are performed until the valid result is present in register 1.

With this solution, the performance achieved is good and no additional resources other than those from DSP blocks are used to achieve high operational frequencies.

3) Parametrization and control signals of the SLFN ELM: The complete structure is built by means of generic instantiation of the hidden and output neuron blocks. LUT ROMs containing the hidden weights, bias and output neuron weights (β) are initialized simultaneously. Elements such as type depths and bit-widths of signals as well as the number of instances of each component depend on parameters defined on a standalone package. The complete SLFN is controlled by the sequence of control signals represented in chronogram of Fig. 9

According to Fig. 9, the control signals of the SLFN ELM IPCore are `RST`, `load`, `CE`, `sum`, `odd` and `calc_prod` and they work as follows:

- `RST` clears the output neurons.
- `load` initializes the hidden neurons to their bias values.
- `CE` is used to activate the accumulators' cascade of the output neurons to iteratively compute the response of the SLFN.
- `sum` is used to accumulate the inputs that are fed up successively into the SLFN according to signal `input_data`.
- `odd` and `calc_prod` in conjunction with the first cycle of `CE` are used to indicate that the output neurons must store the products of the outputs of the hidden neurons by their corresponding β values instead of performing any accumulation.

It is worth noting that with this control scheme, and due to the architecture of the hidden nodes, the features are introduced in bursts, so only 1 feature is available at a time. This characteristic, however, allows the overlapping of the signals `CE` and `sum`, shrinking the total computation time, with the output neurons computing the response corresponding to the last input burst, while the hidden nodes for the current burst are being recalculated simultaneously. A control unit implementing this control scheme is elaborated and both the SLFN and control unit blocks are instantiated together to conform the final SLFN ELM IP Core.

C. Software Partition

The software application has been deployed on the ARM of the PSoC. It operates the full system correctly and performs the data processing operations: dataflow operations, windowing and fast Fourier transform (FFT).

1) Windowing: Finite data frames of 128 seconds of input signals are used to extract RMSs and PSD sums. These finite data frames have a sample rate of 32 Hz, which makes a total of 32 $samples/s \cdot 128\ s = 4096\ samples$ for each signal.

Accepted Manuscript

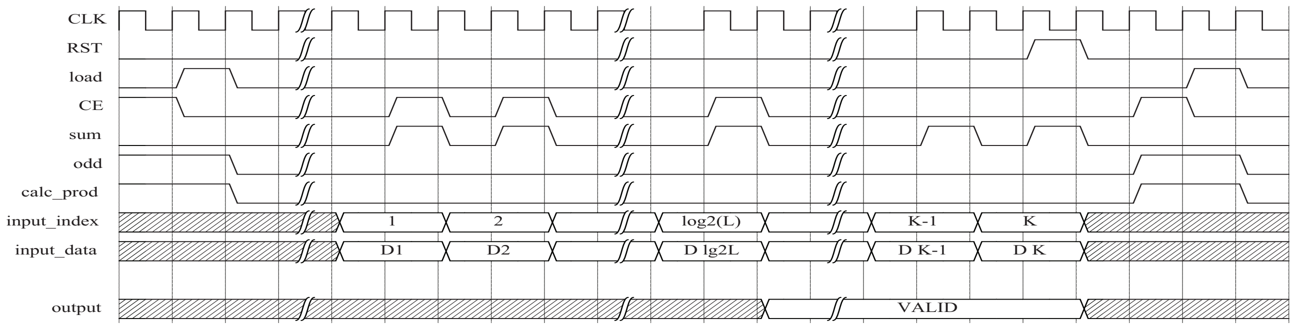

Fig. 9. Full SLFN chronogram.

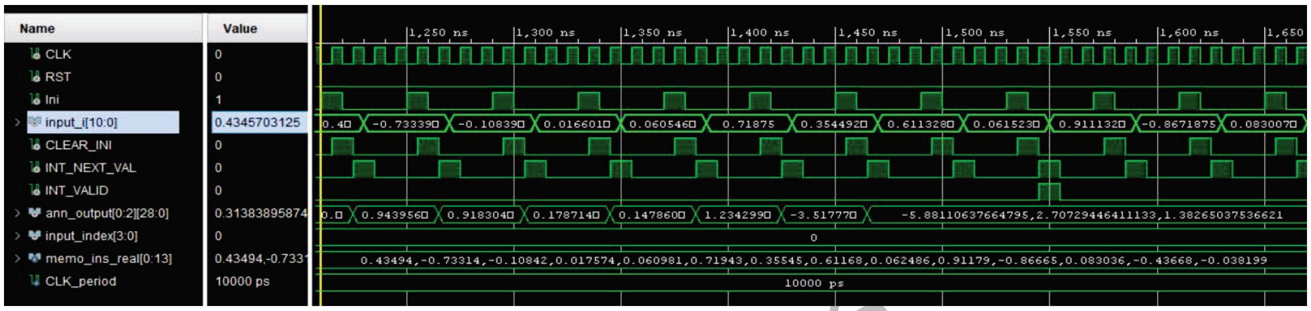

Fig. 10. Response of the VHDL implemented SLFN ELM IP Core.

Accepted Manus

However, taking a piece of data as it is, since that would be like applying a rectangular window, causes several problems, of which spectral leakage is one [18]. In order to minimize this problem, a window function is applied, concretely a Hamming window function. Thus, according to Table I, 7 Hamming windows are calculated for each signal of interest (SWA, SWS, VS, PGP, ERPM, BP and GP). The value of the window $w(n)$ that we must apply on each sample is carried out following the subsequent equation [19]:

$$w(n) = 0.54 - 0.46cos\left(\frac{2\pi n}{M}\right) \tag{8}$$

with n being the positional index of the sample for which the window is being calculated into the data piece, and M the total number of samples of the data piece. Thus, given a piece $\mathbf{C}_k = [c(1), \cdots, c(n)]$ with $n = 1, ..., M$, its corresponding windowing $\mathbf{H}_k = [h(1), \cdots, h(n)]$ is:

$$h(n) = c(n) \cdot w(n) \tag{9}$$

2) Computation of RMS and PSD sum: Once the Hamming windows for SWA, SWS, VS, PGP, ERPM, BP and GP have been calculated, it is necessary to calculate the RMS and PSD sum values for each of the seven windows. The most efficient manner to do this is through the calculation of the FFT [19], since once the FFT is computed, operations on the harmonics can be used to obtain the PSD sum for each windowed signal. The RMS is also calculated.

3) The complete C program: All the individual operations of the preceding sections have been successfully integrated in a C program. The pseudo-code of this program is:

a) Read the first 128 seconds of data (first 4096 samples window).
b) Perform windowing.
c) Compute FFT for each signal's window.
d) Compute PSD and RMS (14 features)
e) Send the burst of 14 features from the PS to the PL though AXI4 bus.
f) Read the output neurons from the AXI4 bus and de-scale the data.
g) The output neuron containing the maximum value is selected as the current comfort class.
h) Wait until the next second.
i) Shift the window 32 samples right (1 second overlap at 32 Hz sample rate).
j) Jump to b.

D. Experimental Results

The full HW/SW system was successfully implemented, returning the post-implementation report summary shown in Table II. The full system fits into the selected PSoC, leaving a significant percentage of the total resources free for further applications.

Regarding timing, the SLFN ELM IPCore post-synthesis timing report states that the minimal clock period that can be achieved with the designed hardware is 9.278 ns, or a maximal clock frequency of 107.78 MHz. Thus, the designed IPCore can be used with an AXI4 Bus Clock frequency of

TABLE II
POST-IMPLEMENTATION RESOURCES REPORT. XILINX XC7Z045-2FFG900

Resource	Utilization	Available	% Used
LUT	82000	218600	37.51
LUTRAM	60	70400	0.09
FF	849	437200	0.19
DSP	550	900	61.11

100 MHz, delaying 26 clock cycles (260 ns at $F_{CLK} = 100$ MHz) to return the computed outputs. This timing outperforms the results previously obtained by using a full-software SLFN ELM MATLAB implementation (20-core Intel Xeon CPU E5-2630 v4 @ 2.20 GHz), with top performance peaks of up to 174 μs to compute the same 14-100-3 topology.

Fig. 10 displays a practical example of driver classification by means of the HW implemented SLFN ELM. Thus, the 14 input features extracted by the SW partition from the windowed input data are sequentially fed to the SLFN ELM through `input_i`. Subsequently, when signal `INT_VALID` goes high-level, the 3 outputs of the SLFN ELM are available in output `ann_output` and ready to be read. The classification procedure finishes by sending these outputs to the SW partition to identify the output with the greatest value. The comfort class that the input features fit with, in this case, is the second cluster: *slightly uncomfortable*.

V. CONCLUSIONS AND FINAL REMARKS

In this work, a hybrid HW/SW extreme learning machine solution has been developed and implemented with the aim of helping to improve ride comfort in automobiles. This system is based on a single-chip implementation of a DS identification system. The selected approach has proven its efficiency with a success rate of up to 95%, while classifying driving data into comfort categories. The single-chip, PSoC-based implementation described in this paper has shown low latency and high operational frequency marks, with the SLFN ELM implementation being able to solve a classification example in 260 ns, making the system more than suitable for real-time processing. Another remarkable fact about this implementation is its low power consumption, with a power input of 2.1 W, making the system suitable to be boarded in real automobiles.

On the other hand, the developed SLFN ELM IPCore is parametrically defined, allowing us to rebuild the entire SLFN ELM by only re-initializing the ROMs and varying the number of inputs, number of outputs and hidden nodes parameters on a VHDL package. This makes this HW development flexible enough to implement a wide variety of SLFN ELM topologies effortlessly. The timing performance of this development makes it suitable for application to a variety of ADAS on which execution times are a critical constraint.

Besides overall ride comfort, the Driving Style (DS) identification can be applied to a variety of different but tightly related fields: correction of the behavior of human drivers in order to improve road safety or fuel economy; enhancement of the previously existing ADAS with the experience of the human driver; and increment of the perceived quality of the automated driving systems based on developing new control strategies.

Finally, it should be noted that there are no ready-to-market, driver advising-based comfort-improving solutions. The proposed single-chip implementation could easily be mounted in current production cars with only minor modifications, resulting in an increase of comfort on manned automobiles.

REFERENCES

[1] C. M. Martinez, M. Heucke, F. Wang, B. Gao, and D. Cao, "Driving style recognition for intelligent vehicle control and advanced driver assistance: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 666–676, March 2018.

[2] R. McAllister, Y. Gal, A. Kendall, M. Van Der Wilk, A. Shah, R. Cipolla, and A. V. Weller, "Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning." International Joint Conferences on Artificial Intelligence, Inc., 2017.

[3] N. Karlsson and H. Tjrnbro, "Motion sickness in cars, department of product and production development," 2012.

[4] J. Eriksson and L. Svensson, "Tuning for ride quality in autonomous vehicle," 2015.

[5] S. M. Casner, E. L. Hutchins, and D. Norman, "The challenges of partially automated driving," *Communications of the ACM*, vol. 59, no. 5, pp. 70–77, 2016.

[6] I. del Campo, E. Asua, V. Martínez, Ó. Mata-Carballeira, and J. Echanobe, "Driving style recognition based on ride comfort using a hybrid machine learning algorithm," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 3251–3258.

[7] J. Arnold, M. Morioka, M. Griffin *et al.*, "Rate of growth of vibration discomfort with increasing magnitude of fore-and-aft, lateral, and vertical whole-body vibration in the frequency range 1 to 10 hz," 2016.

[8] J. Lee and S. Choi, "Braking control for improving ride comfort," in *MATEC Web of Conferences*, vol. 166. EDP Sciences, 2018, p. 02002.

[9] ISO-2631-1, "Iso, mechanical vibration and shock-evaluation of human exposure to whole-body vibration-part 1: General requirements," 1997.

[10] C. Miyajima, Y. Nishiwaki, K. Ozawa, T. Wakita, K. Itou, K. Takeda, and F. Itakura, "Driver modeling based on driving behavior and its evaluation in driver identification," *Proceedings of the IEEE*, vol. 95, no. 2, pp. 427–437, 2007.

[11] M. Ishibashi, M. Okuwa, S. Doi, and M. Akamatsu, "Indices for characterizing driving style and their relevance to car following behavior," in *SICE, 2007 Annual Conference*. IEEE, 2007, pp. 1132–1137.

[12] A. Mudgal, S. Hallmark, A. Carriquiry, and K. Gkritza, "Driving behavior at a roundabout: A hierarchical bayesian regression analysis," *Transportation research part D: transport and environment*, vol. 26, pp. 20–26, 2014.

[13] H. Abut, H. Erdoğan, A. Erçil, B. Çürüklü, H. C. Koman, F. Taş, A. Ö. Argunşah, S. Coşar, B. Akan, H. Karabalkan *et al.*, "Real-world data collection with uyanik," in *In-Vehicle Corpus and Signal Processing for Driver Behavior*. Springer, 2009, pp. 23–43.

[14] E. Öztürk and E. Erzin, "Driver status identification from driving behavior signals," in *Digital Signal Processing for In-vehicle Systems and Safety*, J. Hansen, P. Boyraz, K. Takeda, and H. Abut, Eds. Springer Business-Science, 2012, pp. 31–55.

[15] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489 – 501, 2006.

[16] L. H. Crockett, R. A. Elliot, M. A. Enderwitz, and R. W. Stewart, *The Zynq Book: Embedded Processing with the Arm Cortex-A9 on the Xilinx Zynq-7000 All Programmable Soc*. Strathclyde Academic Media, 2014.

[17] M. Horauer and D. Loy, "Adder synthesis," *Institute of Computer Technology., University of Technology Vienna, Emails: horauer, loy@ ict. tuwien. ac. at, Supported by the Austrian Science Foundation (FWF)*, 1995.

[18] E. Jacobsen and R. Lyons, "The sliding dft," *IEEE Signal Processing Magazine*, vol. 20, no. 2, pp. 74–80, 2003.

[19] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

Accepted Manuscript