Executive Summary of the Thesis

Design and optimization of an FPGA-based system for real-time human stress level assessment

Laurea Magistrale in Electronics Engineering - Ingegneria Elettronica

Author: Nicolò Campanini, Salvatore Torsello

Advisor: Prof. Christian Pilato

Academic year: 2022-2023

1. Introduction

In recent decades, everyday human life is surrounded by a connected and dynamic world, so the individual is exposed to mental pressure situations which could lead to stress conditions. Several researches have been conducted on stress evaluation and different engineering fields are involved when monitoring is needed.
This thesis propose an FPGA-based solution for the assessment of human stress level in real-time, giving particular attention on the development of a modular system, which could be part of a more complex and portable device. Starting by the analysis of the possible signals related to the stress level, our work first addresses the problem with an high-level approach by developing a Python-based processing chain and training a classifier, then it moves to the design of a low-level system composed by an hardware accelerator and a Linux application. A complete description of the validation methods is reported at the end, comparing the simulated model with the one implemented on the target board.

2. Problem assessment

Since stress produces changes in the human physiology, different parameters of the body are influenced by it. To create a stress estimation system we had to choose some of those parameters to be used as input. We selected Electrocardiogram (ECG) and the electrodermal activity (EDA) for our work and we developed a processing chain for them. Those two signals were specifically chosen due to being easy to measure in a non-invasive fashion, requiring cheap sensors and having been intensively studied in the medical community.

2.1. Datasets and signals

To develop a signal processing algorithm for stress detection, experimental data were needed. Due to the constraints imposed by this work of thesis, no ad-hoc experiment could be created but publicly available datasets have to be used. For our work we choose to use two of those datasets: WESAD[6] and DRIVEDB[4].
Those are based on different experiments and different stressors (office work and driving task), but they provide similar stress-level rankings (Relaxed, Normal, Stressed) as well as the input signals we need. We assumed that the induced stress-levels were equal between the two dataset and therefore merged their data for the development of our algorithm.
The two signals used are ECG and EDA.

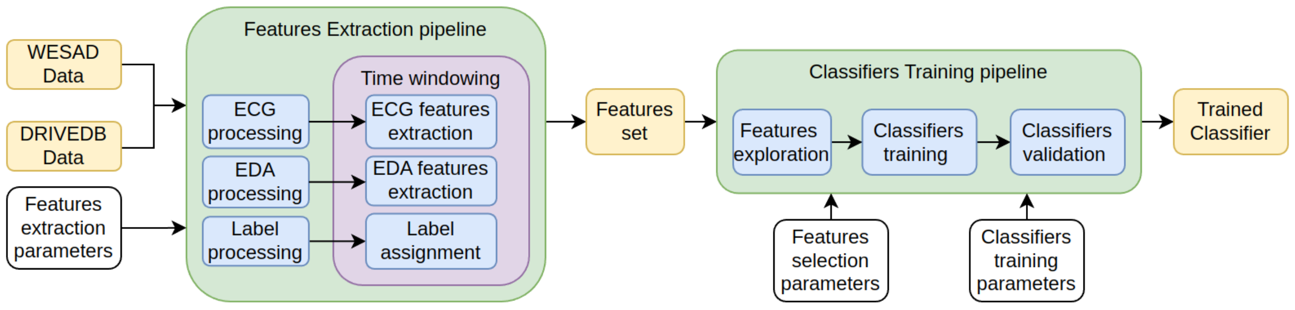

Figure 1: Python algorithm Overall structure

ECG The electrocardiogram signal is describing the electrical activity of the heart over a period of time. It is as voltage signal obtained using multiple electrodes placed on the patient skin to detect the small electrical changes that are a consequence of cardiac muscle depolarization and re-polarization.

EDA The ElectroDermal Activity is the capacity of the human skin to conduct an electrical current in response to different types of stimuli. The common measuring technique consists into the application of a weak electrical current or a low voltage source (which could be DC or AC) across two electrodes places next to each other on the skin, and measuring respectively the voltage or the current.

2.2. Algorithm development

Our Python algorithm, developed in Jupyter, is composed of two separate pipelines which can be individually run and tuned, one for features extraction and the other for classifier training. As it is shown in Figure 1, the *Feature Extraction pipeline* is comprised of the processing of the physiological signals and the time windowing of the resulting characteristics. Different time window sizes were tried during features extraction, ranging from $20s$ to $60s$, with step size of $5s$. The *Classifier Training pipeline* is used to explore and clean the feature set, train different classifiers and validate them with 10-fold cross-validation method.
In the following two paragraphs, the processing of the two physiological signals is explained in detail.

ECG The ECG processing chain is mainly focused on extracting the necessary meaningful signals for Heart Rate Variability (**HRV**) analysis [1]. HRV is the physiological phenomenon of variation in the time interval between heartbeats and it has been shown to be linked to stress level and mental status. The processing chain is centered around a QRS-detection algorithm used to extract heart-beat position from the raw signal. In our work we choose to use the Pan-Tompkins algorithm [5] due to its ease of use and high detection accuracy. From the heart-beat positions two meaningful signal are computed from: **NN** and **Cardiotach**. NN signal describes the evolution of the beat-to-beat interval between different beats while Cardiotach signal describes the same change in time. Features are extracted from those signals in three domains: time, non-linear and frequency. The features of the first two domain are computed directly from the NN and Cardiotach signal while a Lomb-Scargle periodogram is used to perform the spectral analysis.

EDA The main objective of the EDA processing chain is to compute and extract some characteristics that are strictly correlated with the applied stress stimuli [2]. In order to remove individual-to-individual variations, we chose to apply a **percentile normalization**, centering the signal at zero (subtracting the median) and scaling the samples by the Inter-Quantile Range (IQR). Then we decomposed the the resulting signal into tonic and phasic. The tonic component, also called Skin Conductance Level (SCL), was extracted by low-pass filtering the normalized EDA signal with a cutoff frequency of $0.05Hz$. It describes long term changes and slow spontaneous electrical fluctuations, on which statistical features were computed. The phasic component, also called Skin Conductance Response (SCR), was obtained by band-pass filtering with cutoff frequencies of $0.05Hz$ and

$1.8Hz$. It refers to the faster changing elements of the EDA signal and from it critical points [3] were detected consisting in peaks, points of onset (point at which a change in the slope of the curve occurs) and half recovery points (point at which the signal decreases to half of the peak value). At the end, from phasic and the detected critical points we computed the syntactical features which give a geometric description of the signal.

Classifier After a cleaning step which removes features with correlation coefficient higher than 0.95, the classifier pipeline applies different feature selection algorithms and train different classifier model through an iterative approach. The training process was performed for the EDA part on both normalized and non-normalized data. Since applying the normalization step in a real-time setup requires the entire signal to be known in advance, we chose to use the non-normalized data, even if this led to a lower accuracy (-9%). In the end, the best result was obtained with a **XGBoost** [7] model, trained on 25 features, chosen with the **CHI-2** selector, on a $60s$ time window. The resulting accuracy we achieved was of 61%.

3. Proposed solution

We choose to implement the Python algorithm as a heterogeneous system based on the Zynq™ 7010 SoC [8] to better satisfy the modularity requirements. The chosen system architecture is shown in Figure 2. It is composed of a *Biosignal Coprocessor*, implemented in the FPGA part of the SoC, and a *Linux application* running on the SoC CPU.

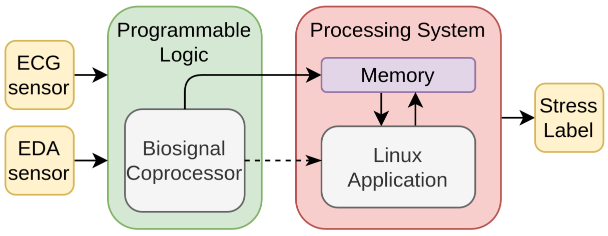

Figure 2: System architecture idea

3.1. Hardware accelerator

The accelerator, designed in Vitis™ HLS, accepts as inputs the two data streams at $500Hz$ (ECG and EDA), it performs all filtering and processing operations required for the extraction of meaningful signals from the two sensors and it produces as output the meaningful signals for the current time window at $1Hz$.

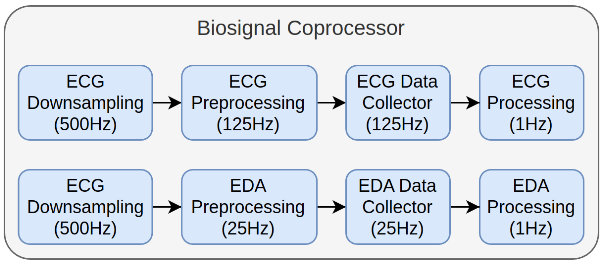

Figure 3: Biosignal Coprocessor structure

Downsampling Here the input data, received from the sensors through AXI-4-Stream interfaces, are downsampled from $500Hz$ to $125Hz$ (ECG) and $25Hz$ (EDA). To perform this operation we implemented and tuned two different downsampling chains based on FIR filters.

Preprocessing After the downsampling step, here ECG and EDA raw data are treated differently. For the ECG signal we performed the QRS detection using a variant of the Pan-Tompkins algorithm [5] to find the position of the heartbeats. The EDA signal instead is firstly separated into its tonic and phasic components using a specific filtering chain. Then the zero-crossing of the phasic first derivative and the sign of the phasic second derivative are computed here to be later used for EDA critical points detection.

Data collector The output data produced by the two preprocessing block are aggregated into $1s$ data blocks, this requires storing 125 samples for ECG and 25 samples for EDA. The aggregated blocks are provided at $1Hz$ at the processing stage.

Processing The processing stage is keeping track of the last 60 blocks received by the Data collector stage and performing different operations on them. For the ECG signal, we extracted the NN and Cardiotach and we filtered the Cardiotach signal with a zero-phase filtering approach. On the EDA signal, instead, we applied the critical points detection algorithm, to

flag the corresponding position of those points. At the end the output produced for both signals are transferred to the main system memory.
After the processing is completed, data are ready to be read from the Linux application. In order to signal this to the CPU, we added an interrupt source which is raised only in those cycles when the system has completed both (ECG and EDA) output data transfers to memory.

Implementation Once all the stages were developed, tested and connected together, we used Vitis™ HLS to synthesize the required Biosignal Coprocessor IP core to be later used in Vivado™. The most important configurations used during the implementation are the fabric clock, at $100MHz$, and the use of *hls::stream* object to create FIFO communication channels between the stages. We also used arbitrary precision data types (*ap_int* and *ap_fixed*) to reduce resource usage, improve performance and avoid floating point arithmetic. We chose to use a Memory-Mapped AXI-4 interface to write data outputs to system memory without CPU intervention, optimizing it to perform burst write by automatically widening the port bit-width greatly enhancing performance. The designed IP core provides also an AXI-4 Lite interface which we used to configure memory addresses for the Memory-Mapped interface and to control the core itself.

3.2. Linux application

Once we exported the IP core from Vitis™ HLS we moved to Vivado™ to generate the platform bit-stream using it to compile a custom Petalinux [9] distribution. On top of it, we developed a multi-thread application using Vitis™ IDE as sketched in Figure 4. We designed the application using the producer-consumer paradigm, with a mutex and a conditional variable to ensure reliable and stable synchronization between the threads. We validated all the described stages with unit-testing, comparing each block output with a golden one obtained by its equivalent in Python. The different threads are now explained in details.

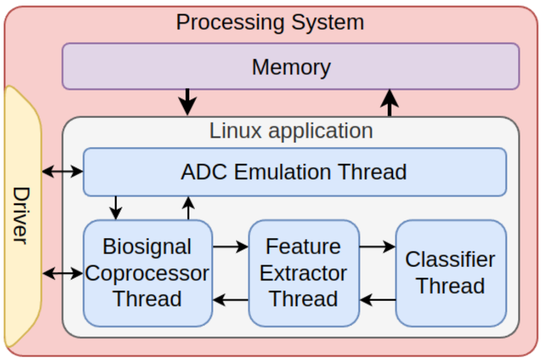

Figure 4: Linux application structure

ADC emulation This thread runs at the ADC target frequency of $500Hz$ and focuses on the emulation of real signal coming from sensors. This is done by firstly loading data on physical memory reading from stimulus files and transferring them to AXI4-stream channels through DMAs. This stage will be not needed in the final system since it was designed only for validation purposes.

Biosignal Coprocessor This thread interfaces directly to the Biosignal Coprocessor IP core through the AXI4-Lite slave interface and the driver generated by Vitis™ HLS, which works on top of the UIO driver. After data have been transferred at the input of the Biosignal Coprocessor, the thread waits for an interrupt from the hardware core doing a blocking read on the corresponding character device. After the function returns, thread operation is restored, and the feature extractor thread is notified, signalling that data are ready to be read at the predefined memory addresses.

Feature extractor The scope of this thread is to perform features extraction on the signals received from the Biosignal Coprocessor. The ECG and EDA signals are again processed in a separate fashion and the resulting features are merged in a single data structure composed of 41 double precision floating point values. The Lomb-Scargle periodogram is calculated from the ECG NN signal to allow for spectral features computation. After the computation of the features, the thread notifies the classifier signalling that data are ready to be consumed.

Classifier The last stage of the chain is the classifier thread, which waits for a new set of computed features and then applies the stress classification on them, obtaining as output the stress label.

3.3. Results

We tested the developed IP-core and the overall system to validate their performance in terms of classification accuracy, resource occupation, and timings.

Timing results The logic timings we obtained from Vitis™ HLS and Vivado™ are reported in Table 1. As shown by the obtained clock periods and Worst-Negative-Slack (WNS) for both Vitis™ HLS and Vivado™, the Biosignal Coprocessor as well as the entire system are able to run at the target clock frequency of $100MHz$ with a consistent margin.

Parameter	Value
CP Vitis™ HLS	$8.397ns$ ($119.1MHz$)
CP Vivado™	$9.580ns$ ($104.4MHz$)
WNS Vitis™ HLS	$1.603ns$
WNS Vivado™	$0.420ns$

Table 1: Biosignal Coprocessor logic timings

We extracted the execution timings of the different blocks of the system using different approaches. For the hardware part we used the cosimulation from Vitis™ HLS to extract the latency if the Biosignal Coprocessor. For the different software threads, instead, we used a hardware timer to measure the time each thread requires to execute. The timings we obtained are reported in Table 2. The obtained results are well below the requirements for both the hardware part as well as the software part.

Classification results We extracted the classification results using the ADC emulation approach previously explained. The results are shown in Figure 5 normalized over the rows. It

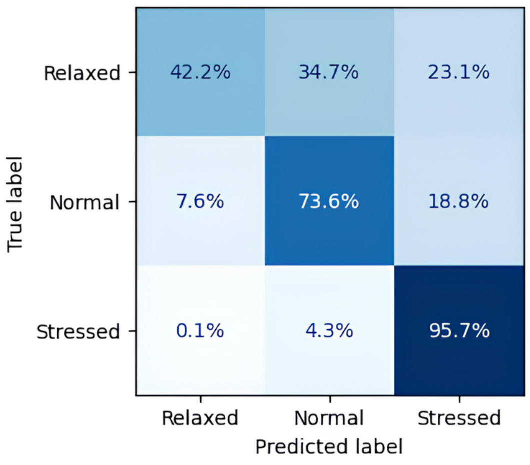

Figure 5: Classification results

can be easily seen that misclassification is occurring a lot between the *Relaxed* and *Normal* levels while the system is extremely accurate at detecting the *Stressed* condition. In our opinion this classification error is mainly caused by inter-individual variability which plays a more significant role at those specific stress levels. Moreover the datasets we used for training are based on different stressors and this could affect the obtained classification accuracy. A more specific set of experiment is needed to train the classification algorithm and obtain a more reliable

	Min	Avg	Max
Biosignal Coprocessor		$2ms$ ($500Hz$)	
Result [s]	$2.40\mu s$	$9.97\mu s$	$1.39ms$
Result [Hz]	$416.7KHz$	$100.3KHz$	$714.9Hz$
Feature Extractor		$1s$ ($1Hz$)	
Result [s]	$2.23ms$	$29.7ms$	$48.9ms$
Result [Hz]	$448.5Hz$	$33.7Hz$	$20.5Hz$
Classifier		$1s$ ($1Hz$)	
Result [s]	$255.5\mu s$	$438.1\mu s$	$839.3\mu s$
Result [Hz]	$3.91KHz$	$2.28KHz$	$1.12KHz$

Table 2: System timing results

estimation.

Resource occupation Table 3 shows the resource occupation of the entire system when composed in Vivado™. This includes the IP core as well as DMA blocks used for ADC emulation and all interconnection structures.
The entire system is able to fit inside the Zynq™ 7010 SoC without exceeding the available resources. Additional resources are still available and can be easily used for further improvements of the Biosignal Coprocessor IP core.

Resource	Used	Available	Used %
LUT	12054	17600	68.49%
FF	20240	35200	57.50%
DSP	27	80	33.75%
BRAM	23	60	–

Table 3: Resource occupation from Vivado™

4. Conclusions

The estimation of human stress level is an hot topic of the affective-computing world and is an important step in the creation of healthier and safer work environment as well as preventing accidents on high-risk jobs.
With the system we developed in this work of thesis we were able to estimate the stress level in real-time using ECG and EDA, two signals easy to collect with simple and cheap sensors in a non-invasive fashion. The obtained stress estimation performances were satisfactory knowing that only data from publicly available datasets have been used.
The proposed heterogeneous architecture processes data in hardware at high frequency directly from sensor, thus simplifying the extraction of the meaningful signals and avoiding the use of CPU computational power for those tasks. The software application also allows for an easy scalability of the system and simplifies the development of additional software part of the system. The obtained system, with its modular and expandable architecture, can therefore be considered a platform which can be improved on and act as a base for the creation of more complex wearable stress detection systems.

References

[1] George E Billman, Heikki V Huikuri, Jerzy Sacha, and Karin Trimmel. An introduction to heart rate variability: methodological considerations and clinical applications, 2015.

[2] Jason J Braithwaite, Derrick G Watson, Robert Jones, and Mickey Rowe. A guide for analysing electrodermal activity (eda) & skin conductance responses (scrs) for psychological experiments. *Psychophysiology*, 49(1):1017–1034, 2013.

[3] Michael E Dawson, Anne M Schell, and Diane L Filion. The electrodermal system. *Handbook of psychophysiology*, 2:200–223, 2007.

[4] Jennifer A Healey and Rosalind W Picard. Detecting stress during real-world driving tasks using physiological sensors. *IEEE Transactions on intelligent transportation systems*, 6(2):156–166, 2005.

[5] Jiapu Pan and Willis J Tompkins. A real-time qrs detection algorithm. *IEEE transactions on biomedical engineering*, pages 230–236, 1985.

[6] Philip Schmidt, Attila Reiss, Robert Duerichen, Claus Marberger, and Kristof Van Laerhoven. Introducing wesad, a multimodal dataset for wearable stress and affect detection. In *Proceedings of the 20th ACM international conference on multimodal interaction*, pages 400–408, 2018.

[7] XGBoost Python Package 2014; xgboost 1.7.6 documentation — xgboost.readthedocs.io. `https://xgboost.readthedocs.io/en/stable/python/index.html`, n.d. [Accessed 24-08-2023].

[8] Xilinx. AMD Adaptive Computing Documentation Portal — docs.xilinx.com. `https://docs.xilinx.com/v/u/en-US/zynq-7000-product-selection-guide`, n.d. [Accessed 25-08-2023].

[9] AMD Adaptive Computing Documentation Portal — docs.xilinx.com. `https://docs.xilinx.com/r/en-US/ug1144-petalinux-tools-reference-guide/Introduction`, n.d. [Accessed 31-08-2023].