System on Chip (SoC) for Invisible Electrocardiography (ECG) Biometrics

Francisco Melo*,†, Horácio Neto*, Hugo Silva*†

*Instituto Superior Técnico, Lisboa, Portugal
†Instituto de Telecomunicações, Lisboa, Portugal

Abstract—Recent approaches to Electrocardiography (ECG) focus on the sensor integration in everyday objects, leading to the concept of "invisible" ECG (aka "off-the-person"). This allows the creation of a highly granular ECG medical history, useful for prevention and monitoring of cardiovascular diseases. In many use cases, everyday use objects are shared between multiple people (e.g. ECG monitoring in sanitary facilities), reason for which an ECG identification is required to separate the users’ ECG record history. This work proposes an Artificial Intelligence (AI)-enabled ECG-based identification embedded system for toilet seats, composed of a RISC-V based System-on-a-Chip (SoC) and an ECG sensor. The sensor was especially dimensioned to enable invisible electrocardiography by capturing ECG signals on the thighs’ surface, which are then processed in the SoC. A novel Binary Neural Network (BNN) designed in the scope of this work, the ECG-ID-BNet, achieves state-of-the-art accuracies of up to 100% (for a subset of users compatible with the target application), while being the most lightweight Neural Network (NN) in the literature that performs ECG classification. This network was implemented in our SoC’s hardware accelerator that, when compared to a similar software implementation of a conventional, non-binarized, Convolutional Neural Network (CNN) version of our network, achieves a $176270 \times$ speedup, arguably outperforming all the current state-of-the-art CNN-based ECG identification methods.

Index Terms—Electrocardiography, Identification, Binary Neural Networks, System-on-a-Chip, RISC-V, Hardware Accelerator

I. INTRODUCTION

Invisible ECG enables signal acquisition without the user having to go through complex procedures like in conventional ECG hospital equipment. There has been an increasing number of applications where invisible ECG allows the creation of a highly granular ECG medical history, useful for prevention and monitoring of cardiovascular diseases. One such use case is ECG monitoring in sanitary facilities, in which the toilet seat is shared between multiple people, requiring an identification to separate each users’ medical record. These ECG-enabled devices often need to meet small size, low throughput, and power constrains (e.g. battery-powered), thus needing to be both resource and energy-efficient. As further described in Section III-A, this system requires 3 heartbeats to perform an identification, reason for which a latency requirement of $3 \times \frac{1}{60} \approx 1.2$ s for the whole identification process was imposed, in order for the system to achieve real-time identification.

Due to the lack of off-the-shelf alternatives, an ECG sensor was especially designed to capture ECG signals on the thighs’ surfaces, aiming at enabling invisible electrocardiography in toilet seats, and conditioning the acquired signal to be fitted for digital conversion.

The rest of the paper is organized as follows. Section II summarizes the current state-of-the-art techniques for ECG identification, new BNN approaches, an overview of current RISC-V processors and current CNN acceleration techniques. Section III focuses on the development of the novel ECG-ID-BNet and analyses its accuracy in different configurations. Section IV introduces the SoC and describes the hardware platform; additionally, it also presents a hardware resource utilization report and a performance profiling of the implementation. Section V outlines the hardware accelerator of ECG-ID-BNet, and presents an analysis of its performance and hardware resource usage; moreover, it compares its performance with the software-only RISC-V implementation. Section VI summarizes the experimental results obtained. Finally, Section VII concludes this work and presents its main outcomes and future work.

II. STATE-OF-THE-ART

A. ECG-based Identification

Due to the lack of off-the-shelf alternatives, an ECG sensor was especially designed to capture ECG signals on the thighs’ surfaces, aiming at enabling invisible electrocardiography in toilet seats, and conditioning the acquired signal to be fitted for digital conversion.

Previous work in ECG-based biometric recognition uses two main feature extraction strategies: fiducial and non-fiducial [1]. The fiducial points are six waves (P, Q, R, S, T and U) present in a typical ECG signal [2], [3] (see Figure 1), and the feature extraction methods based entirely on these points entail the computing of certain metrics. These include amplitude, slope values, or peak-to-peak interval, just to name
a few examples [4], [5]. Recent work has found that, since this strategy depends on a reduced number of features from the ECG signal, it does not benefit from all the information that the signal has to offer. Using the whole heartbeat waveform morphology is currently considered the best approach to promote a richer feature extraction [6]–[8]. In what concerns identification algorithms, the best-performing methods found in the state-of-the-art, use classifiers trained on the extracted features, which can be categorized into:

- **Classical Machine Learning:** Assigning the unknown input sample to the class in which the features are the most similar [9]. Classifiers such as SVM, LDA and KNN are popular in previous work [6], [7], [10]–[12].

- **Neural Networks based classifiers:** In contrast with distance based methods, NN based classifiers learn the complex relations between different input feature vectors.

In [13], the authors propose a PCA-based algorithm applied to a feature space of 16 fiducial features. Afterwards, a quadratic polynomial model was used as a classifier. This approach managed to achieve 95% accuracy, when predicting the identity in a universe of 20 subjects.

In [14] then authors proposed a NN based ECG identification. A PCA algorithm was implemented for feature extraction and a back propagation NN was used as a classifier. The dataset used was composed of 24 hour records from 33 different patients. Using a sequence of 10 heart beats per prediction, the accuracy obtained was 97%.

In [15] the authors propose a cascaded CNN (2 CNNs) for ECG identification. The first CNN, the F-CNN, acts as a feature extractor, which works with pre-processed ECG records that are filtered, to remove unwanted noise, and heartbeats segmented into templates (based on the R peak). The second CNN, the M-CNN, uses as input not only the features generated by the F-CNN, but also the original template that was fed into F-CNN, and then it performs binary classification, i.e. match or no match. The system managed to achieve 99.3% accuracy, using a dataset composed of ECG recordings from 40 different subjects.

In summary, the most recent studies about ECG identification use CNN based NNs to achieve state-of-the-art accuracy [15], [16] and, to our best knowledge, no previous work that implements BNNs for this application were found.

### B. Binary Neural Networks

Quantization of the network’s parameters is a popular practice to save memory usage and increase computational performance. The parameters are typically represented by 32-bit floating-point values, however, the hardware required to process floating-point data is much more complex and slower, comparatively to what is needed to handle integer data [17]. An approach commonly found in literature is to quantize floating point values to 8- or 16-bit integer values, but a more extreme quantization can be performed.

Binarized Neural Networks, first proposed by Courbariaux et al. in [17], introduced the concept of constraining the values of the network’s parameters to either +1 or -1 [17], allowing a 1-bit representation. This new quantization paradigm helps minimize the memory footprint taken by a NN.

| TABLE I |
| DIFFERENCE BETWEEN DOT PRODUCT AND XNOR. |
| A | B | A · B |
| -1 | -1 | 1 |
| -1 | 1 | -1 |
| 1 | 1 | 1 |

In Table I, a comparison between a multiplication and a XNOR logic operation is illustrated. If the values ",1" and ",1" are, respectively, encoded with the binary values "0" and "1", a perfect correlation exists between the two operations. Moreover, in the result of \(A \oplus B\), if the number of unset bits (bits with the value "0") is subtracted from the number of set bits (bits with value "1"), the result is the same to a Multiply and Accumulate (MAC) operation. This binary equivalent of the integer MAC is called XNOR Dot Product (XNP) [17], where popcount refers to the process of counting the number of set bits.

These XNPs are very simple because of their bitwise nature. Comparatively to floating-point and even fixed-point multiplications, a XNOR is much simpler to compute, it requires less hardware, and it leads to faster execution times. Furthermore, XNP operations have been found to be highly energy efficient [18], in comparison with MAC operations. As such, this concept is especially appealing for limited resources and low-power devices.

**Standardization** of the neurons/filters’ input data, making it follow a standard normal distribution, is a common process in order to increase stability and training speed. Such operation is known as Batch Normalization (BN) [19], and acts as a form of regularization, being a type of optional layer commonly found in network topologies such as CNN. The expression that defines BN is shown in Equation 1, where \(x\) is the input, \(\mu\) is the mean of the batch, \(\sigma^2\) is the variance of the batch, \(\gamma\) and \(\beta\) are learned parameters, respectively, a learned gain and bias, and \(\epsilon\) is added for numerical stability.

\[
BN(x) = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \gamma + \beta \tag{1}
\]

Implementing a BN in the forward pass is equivalent to an integer threshold in BNNs. The network computes \(sign(BN(x))\), hence, this computation can be reorganized into a single threshold calculation [20],

\[
\text{sign}(x) = \begin{cases} 
+1 & x \geq \tau \\
-1 & x < \tau 
\end{cases} \tag{2}
\]

where

\[
\tau = -\beta \frac{\sqrt{\sigma^2 + \epsilon}}{\gamma} + \mu, \quad \gamma > 0 \tag{3}
\]

If \(\gamma\) is negative, the input \(x\) would need to be negated, and the same threshold function can be used.
C. FPGA based CNN

Previous authors have explored hardware acceleration of the inference stage of 2D CNNs. Field-Programmable Gate Array (FPGA)-based CNN accelerators aim to improve inference performance by parallelizing the CNN forward propagation. There are 6 main ways to parallelize a CNN’s inference [21]: intra convolution parallelism, inter convolution parallelism, inter feature map parallelism, intra feature map parallelism, inter layer parallelism and batch parallelism.

Previous FPGA-based CNN accelerators [22]–[24] follow a standard architecture (depicted in Figure 2) that entails: an external memory, used to hold the CNN’s parameters and input/output feature maps; an input and output buffer to cache the input and output feature maps; a Processing Element (PE) that processes the operations required to compute the outputs using the cached inputs and the weights that are maintained in a PE’s internal buffer; and a controller which controls the overall execution.

In works such as [22], [23], a PE (illustrated in Figure 3) performs concurrent MAC operations to solve a single 2D convolution. This is an example of intra convolution parallelism.

The authors of [22] take advantage of inter convolution parallelism. Their accelerator implements a PE that is capable of processing multiple input feature maps, by summing the result of multiple concurrent 2D convolutions with the objective of computing a single 3D convolution necessary to produce an element of an output feature map.

In [26], the authors implemented all layers in a pipelined structure that enables the execution of all layers concurrently, while requiring a substantial amount of FPGA resources. This is an example of inter layer parallelism. In the same work, the authors propose a divide and conquer strategy in the computation of fully connected layers that, if executed all at once, requires a substantial amount of memory to hold all the operands. They divide the operation into multiple simple sub-convolutions, whose results can be accumulated to get the final result; this is an example of batch parallelism.

In [27], the authors note that in modern CNN architectures, such as [28], in deeper convolutional layers, the number of input/output channels surpasses the actual input/output feature map size. In such CNNs, the authors argue that intra feature map parallelism is preferred over inter feature map parallelism.

III. BCNN ECG IDENTIFICATION

In this work a non-fiducial Binary Convolutional Neural Network (BCNN) based strategy for ECG biometric recognition was developed. A CNN was chosen given the superiority of this kind of method, as described in Section II-A. The proposed ECG-ID-BNet architecture uses a fully connected layer as a classifier after the convolutional layers of the feature extractor. This is particularly advantageous as it offers size-optimization, low-power, and high accuracy in ECG-based biometric identification on the edge. The PyTorch [29] Python-based deep learning library was used to assist in the NN specification.

A. Preprocessing

Similarly to state-of-the-art works such as [15], the proposed ECG-ID-BNet was designed to take filtered, segmented ECG recordings (templates) as inputs. A digital Finite Impulse Response (FIR) bandpass filter of order 300 was used to preserve the signal only in the 3 to 45 Hz range, while the Hamilton method was used for segmentation; this follows the approach proposed in [30].

After segmentation, the BCNN input data is represented as 32-bit floating point numbers. To minimize the memory footprint, the data is quantized to 8-bit unsigned integers. The specific 8-bit quantization is chosen from the discussion of the accuracy degradation caused by the quantization, evaluated in Section VI-A. It is common practice to perform standardization on the NN’s input data. However, a preliminary study was made to characterize the impact of this technique and it showed that it didn’t improve the NN’s performance.

In addition, template averaging, which has been shown in previous work [30] to work as a filter in the template space and attenuate the effect of outlier templates, was experimented to evaluate the impact on the accuracy. The averaging uses 3 temporally consecutive templates.
B. Datasets

Two datasets were used in this work, namely a publicly available dataset [31] and our own dataset, collected using the proposed ECG sensor.

**AliveCor:** The AliveCor dataset was chosen due to the fact that it is one of the biggest reference datasets with ECG signals acquired using dry electrodes, on the extremities of the body. To our best knowledge, within publicly available datasets, these conditions are the closest to our application’s (in which data is collected at the thighs using dry electrodes). The ECG recordings were collected using the AliveCor device [32] in a single-lead configuration, and 300 Hz sample rate. The dataset contains 8,528 ECG recordings lasting from 9 seconds (s) to just over 60 s. Among the available records, 60 s long ECG recordings, each belonging to a different subject, were randomly selected to constitute the dataset for this work. The 60 s duration was chosen in order to maximize the dataset size and to avoid over representation of a single class. Within the selected records, we chose fifty subjects since it is a common upper bound for the number of subjects in previous work in ECG biometrics [33].

**Ours:** Due to the anatomical placement of the sensors in our approach, the ECG signal has lower signal to noise ratio, which makes it more prone to noise artifacts [34] consequently reducing its overall quality. This results in an increased number of outlier templates [34], i.e. templates which do not follow the typical P-Q-RST complex morphology (see Figure 1). Although the AliveCor dataset also entails some outlier templates, it is still considered superior and it was thus used as the baseline for the ECG-ID-BNet development. The ECG recordings were collected at the thighs at 100 Hz sample rate. This dataset contains three minute long recordings belonging to 10 different subjects.

C. Evaluation Methodology

To evaluate the NN performance, a test set was used. The conventional way to test Machine Learning (ML) models is to perform a simple hold-out split. This technique consists of splitting the original dataset into a training and test set, usually in a 80/20 ratio. When the dataset is small and some of its classes have a small population, using this strategy may not guarantee data representativeness in the evaluation.

In order to ensure a more robust and precise evaluation method, the k-fold cross validation was used. This strategy splits the dataset K times, making the model train and evaluate in K different instances. Specifically, the split was made with \( K = \frac{L}{C} \), with \( L \) being total number of samples in the whole dataset and \( C \) the number of classes. This split ensures that the minimum amount of samples are present in the test set while maintaining data representativeness, by having at least one sample of each class. For simplicity, in the sections below, the mentioned “accuracy” refers to the average accuracy of each evaluation instance.

D. Architecture Selection

Due to the novelty of the problem, a NN had to be developed from ground up. A 1D CNN was picked as a baseline because, as already discussed in Section II-A, it is the current state-of-the-art for ECG-based identification. In order to minimize the NN’s computational resources, the network is a BCNN, a CNN-based BNN.

Three constrains to the architecture selection were introduced, in order to enable the deployment of the developed BCNN to the targeted hardware platform described in Section IV, without the need to increase the available memory through an external memory: (1) overall reduced number of layers; (2) special care was taken when deciding the number of fully connected layers to minimize memory usage thus, only one fully connected layer to act as a classifier in the last stage of the network was allowed; (3) At least one max pooling layer should be present, mostly to attenuate the fully connected layer’s memory footprint in the overall NN memory usage.

To select the NN architecture, a hyperparameter tuning analysis was made using the Tune toolbox [35]. This solution enables an automatic search of traditional hyperparameters, such as number of layers, amount of filters in each layer, kernel size in each filter, batch size, number of max pooling layers and number of training epochs. In addition, a custom hyperparameter was defined, which determined if the template averaging, discussed in Section III-A, was applied to the dataset. Furthermore, the hyperparameter tuning implemented [35] takes advantage of ASHA [36], a robust hyperparameter optimization algorithm that further improves the hyperparameter search.

The last layer’s activation function chosen was the softmax, and thus its pre-activations cannot be binarized. The training loss function selected was Cross-Entropy (CE) and the gradient descent optimizer used to train the BCNN is Adam [37], which is based on Stochastic Gradient Descent (SGD).

IV. SOFTWARE IMPLEMENTATION

The RISC-V Instruction Set Architecture (ISA) was first developed in 2010, at UC Berkeley, and has been gaining popularity in both academia and industry [38]. Compared with other architectures, the RISC-V provides a number of advantages that makes it especially attractive for development [39]:

- The open-source nature allows any interested entity access to the source Intellectual Property (IP) of the cores without any licensing issue.
- Even though the development of the ISA is open-source, its major features are already well defined and stable, which also attracts software development.
- Additional functionalities are available through a set of extensions, which are also well defined after stabilization.
- Due to its modular nature, the ISA is suitable for both high performance and low power chip applications. Additionally, specialized application processors that feature dedicated accelerators are also supported by the ISA.
In a first attempt to deploy the ECG-ID-BNet on the edge, the IOb-SoC [40], a RISC-V based SoC implemented in a FPGA was chosen as the target hardware platform, as it provides a convenient and efficient infrastructure to create FPGA SoC. In this section, the software implementation is described.

A. Hardware Platform

IOb-SoC [40] is an open-source RISC-V based SoC platform written in Verilog. This platform is composed of a RISC-V soft-core, the PicoRV32 [41], an internal SRAM subsystem, an optional external memory interface and peripherals. It implements the RISC-V RV32IMC instruction set [38] and it can be programmed with the RISC-V GNU Compiler Toolchain [42]. The hardware platform selected to implement the IOb-SoC was the FPGA development board Basys3 [43], which features a Xilinx Artix-7 XC7A35T FPGA whose main characteristics are shown in Table II.

| TABLE II | XC7A35T FPGA MAIN SPECIFICATIONS. |
|-----------|----------------------------------|
| LUTs      | FFs     | 36kb BRAMs | DSP Slices |
| 33280     | 41600   | 50         | 90         |

B. ECG-ID-BNet Memory Usage

Since the target hardware platform to deploy the NN is a resource constrained embedded system, the memory usage of the NN must be carefully monitored. A memory usage study of ECG-ID-BNet, whose architecture is detailed in Table IX, was conducted in order to define the memory necessary to:

(1) hold the network’s parameters; 
(2) hold the input/output feature maps of each layer.

C. Memory Reorganization

The traditional storing order of a filter’s weights is \((x, c)\), where \(x\) indicates the kernel width dimension and \(c\) the channel dimension. However, this work proposes the weight arrangement \((c, x)\), illustrated in Figure 4.

Doing this rearrangement enables inter convolution parallelism, allowing the fast computation of a single output feature map element. Furthermore, every ECG-ID-BNet layer (see Table IX) has a 64 multiple number of input filters, with exception of the first layer. With this ECG-ID-BNet feature, it is possible to group 64 1-bit weights in 64-bit blocks. 64-bit was chosen as the size of the blocks, over other valid bit widths such as 32-bit, in order to ensure compatibility between the software application and the IP developed in Section V. By having weights memory arrangement, it is possible to enable operand parallelism, taking advantage of the 32-bit computing of the targeted processor if the input feature maps are also in the \((c, x)\) arrangement.

D. Execution Flow

Each unit’s output becomes the following unit’s input. This enables the adoption of two memory buffers, used in a ping-pong fashion, where their place alternate every unit iteration.

In Table III each unit’s memory usage is summarized. The convolutional units have a considerably lower number of parameters when compared to the fully connected one.

As a requirement for the hardware platform, it is concluded that ECG-ID-BNet requires 57200 Bytes of memory to hold the parameters and two buffers of at least 2784 Bytes each to hold the units’ input/output feature maps.

| TABLE III | ECG-ID-BNET MEMORY USAGE SUMMARY. |
|-----------|----------------------------------|
| Input Fmaps (Bytes) | Output Fmaps (Bytes) | Parameters (Bytes) |
| Convolutional Unit 1 | 180 | 2784 | 2048 (3.6%) |
| Convolutional Unit 2 | 2784 | 1344 | 7680 (13.4%) |
| Convolutional Unit 3 | 1344 | 2592 | 8192 (14.3%) |
| Convolutional Unit 4 | 2592 | 624 | 7680 (13.4%) |
| Fully connected Unit | 624 | 200 | 31600 (55.2%) |
| Max. 2784 | Total: 57200 (100%) |

Figure 5 depicts the high-level flowchart of the software application. A pre-compiled binary file with all necessary ECG-ID-BNet parameters is used for inference. The file contains: global parameters, such as number of units and convolutional kernel size; unit parameters, including the type of unit, the presence of max pooling, the unit’s relative position.
TABLE IV
IOb-SoC ECG-ID-BNet SRAM Section Memory Size.

| Section       | Memory Size (Bytes) |
|---------------|---------------------|
| Firmware      | 28024               |
| Stack         | Unknown             |
| Model Parameters | 57200             |

TABLE V
IOb-SoC ECG-ID-BNet SRAM Address Mapping.

| Section       | Address Range          |
|---------------|------------------------|
| Firmware      | 0x000000 - 0x0FFFFF    |
| Stack         | 0x000000 - 0x0FFFFF    |
| Model Parameters | 0x10000 - 0x1FFFFF    |

in the network, number of input and output feature maps, output feature map size, and the convolution/fully connected layer’s parameters (weights and binarization thresholds for first and hidden units). A Pseudo SoftMax, instead of SoftMax, is performed on the batch normalized outputs of the last unit, where the predicted class is assigned by determining the max output value, skipping the probability distribution transformation carried out by SoftMax.

E. IOB-Soc Configuration

The memory requirements of ECG-ID-BNet are presented in Section IV-B. Considering the resources of the hardware platform (briefed in Table II), it was concluded that the memory resources of the FPGA are sufficient to hold the firmware and to satisfy the BCNN’s memory needs. Hence, the external memory is not necessary.

Special care was taken to the dimensioning of the SRAM since, beyond the parameters and the firmware, the program’s stack, where the input and output feature map buffers reside, also takes a considerable amount of memory and thus needs to be taken into account.

IOB-SoC’s SRAM is composed of cascaded 36kb Block RAMs and the BRAMs are configured to be byte-addressable, by having the configuration $4k \times 9$ (9-bit $4k$ words) and ignoring one bit per word. Considering the sections memory sizes in Table IV, a preliminary test was conducted using a 128kB SRAM implementation; the program ran successfully, thus validating the SRAM’s size. The final SRAM memory address mapping is described in Table V.

Since each BRAM has 4k 9-bit sized words, the total SRAM’s expected usage of BRAMs is given by $\frac{128k \times 9}{4k} = 32$.

F. IOB-SoC Optimization

Contrary to the a BCNN software implementation, which is able to have a speedup of 32 in all units (except the first) over a CNN implementation (see Section IV-C) due its inter convolution parallelism, a conventional CNN cannot achieve any kind of parallelism during inference in a software implementation (using only one core). Thus, a PicoRV32 software implementation of the inference of a conventional CNN version of ECG-ID-BNet would take around $0.24 + 32(0.37 + 0.4 + 0.34 + 0.01) = 36.1s$. This implies a speedup of $\frac{36.1}{1.36} = 26.5$ of a BCNN version over a conventional CNN one.

The performance results show that a software only approach is not able to achieve real-time ECG identification, since the overall identification process (preprocessing, feature extraction and classification) needs to be under 1.2s (see Section I) and only the ECG-ID-BNet takes 1.36s > 1.2s. As such, a more efficient solution for the ECG-ID-BNet inference is needed.

V. IP-Core

An IP core was designed to execute ECG-ID-BNet in the same FPGA development board used in Section IV. The hardware architecture (illustrated in Figure 6), selected based on the state-of-the-art review (Section II-C), is similar to the ones developed in the works [22]–[24].

![Fig. 6. ECG-ID-BNet accelerator high-level architecture.](image)

Two methods of CNN parallelism were used: (1) inter convolution, by having a PE processing $N_{ops}$ inputs, each belonging to a different feature map; and (2) inter feature map, by having a group of $N_{PEs}$ PEs concurrently computing multiple output feature maps (i.e. each PE executes one filter). These allow all PEs to process the same input feature map elements, thus reducing the required input memory bandwidth. The possible number of concurrently processed input feature maps is limited due to BRAM bandwidth, which is explored in Section V-B.

A. Execution Flow and Control

![Fig. 7. ECG-ID-BNet IP core execution flow.](image)
The control unit of the proposed IP is subdivided into two units that work in parallel: (1) the parameters control unit, responsible to download the parameters of each filter from the parameters RAM into each corresponding PE internal buffer; and (2) the execution control unit that oversees the inference process by controlling the PEs and input/output feature map RAMs via control signals. Both control units are accomplished with finite state machines.

In Figure 7 a depiction of the execution flow is presented. In the scope of this work “session” is defined as the computation of $N_{PEs}$ output feature maps concurrently by the PEs, thus multiple sessions are needed in order to fully execute a layer, e.g. with $N_{PEs} = 32$ and $N_{out,c} = 64$, $N_{sessions} = \frac{N_{out,c}}{N_{PEs}} = 2$ are needed. Moreover, an “even” and “odd” session refer to an even numbered session (e.g. session 0) and to an odd number session (e.g. session 1). The ECG-ID-BNet inference starts with a setup phase, which involves downloading the first batch of $N_{PEs}$ filters, to enable the execution of the first session. After that, the execution parameters control units work concurrently in a ping-pong fashion: when one is working in an “even” session, the other is working in an “odd” one.

This promotes execution performance, but puts a memory constrain in the internal PE memories, since they need to be large enough to hold two sets of filters’ weights at a time. The control unit employs counters to iterate through the network’s inference iterators. The input/output memory module contains an address calculator, which uses a counter to keep track of the current output address and current output channel to compute the PEs’ outputs memory destinations.

B. Memory Specification

The target FPGA contains 50 36kb BRAMs (detailed in Table II), and each BRAM can be configured as single-port RAM, simple dual-port RAM, true dual-port RAM, simple-port RAM or dual-port RAM [44]. The simple dual-port RAM is the configuration that enables maximum port width; due to the FPGA constraints, the BRAM’s $depth \times width$ ratio must be one of the following: $32k \times 1$; $16k \times 2$; $8k \times 4$; $4k \times 9$; $2k \times 18$; $1k \times 36$; or $512 \times 72$ [45].

| TABLE VI |
| ECG-ID-BNet MEMORY USAGE PER FILTER/NEURON IN EACH UNIT SUMMARY. |
| # Filters/Neurons | Parameters (Bytes) | Filter/Neuron Memory Size (Bytes) |
|-------------------|--------------------|----------------------------------|
| Convolutional Unit 1 | 128 | 2048 (3.6%) | 16 |
| Convolutional Unit 2 | 64 | 7680 (13.4%) | 120 |
| Convolutional Unit 3 | 128 | 8192 (14.3%) | 64 |
| Convolutional Unit 4 | 64 | 7680 (13.4%) | 120 |
| Fully connected Unit | 50 | 31600 (55.2%) | 632 |
| Total: | 57200 (100%) | Max: 632 |

This IP uses the internal FPGA BRAMs to hold three main data components: (1) NN parameters; (2) weights for each filter; and (3) input and output feature maps. In Table VI a detailed ECG-ID-BNet memory usage is shown, where the memory necessary to hold the weights of each filter/neuron are indicated.

The memory holding the parameters is configured as a multiple cascaded single-port RAM. The amount of BRAMs needed to support the parameters is defined by $\frac{57200 \times 8}{36k} = 12.41$ (ECG-ID-BNet’s parameters size divided by BRAM size), which gives a total of 13 BRAMs. As shown in Table III, the two buffers must have a minimum size of 2784 bytes; this can be accomplished with a single BRAM for each of them. Similarly, in Table VI, it is possible to check that the maximum filter or neuron size is 632 bytes, and since each PE’s weights buffer must be able to hold two filters/neurons, it’s minimum size must be $632 \times 2 = 1264$ bytes, which can also be accomplished with a single BRAM.

Due to the fact that the weight and input/output RAMs are composed of a single BRAM, the maximum possible width, due to the FGPA constraints, is 72 bits, unless more BRAMs are cascaded. However, that would deplete rapidly the number of BRAMs available and consequently the possible number of PEs. Using the same memory reorganization described in Section IV-C, each PE processes multi-bit blocks of inputs and weights, enabling inter convolution parallelism. Additionally, by analysing the topology of ECG-ID-BNet (Table IX), it is possible to conclude that the number of input feature maps is always a multiple of 64, with the exception of the first unit. This enables a fast and simple computing of output feature map elements by each PE, if the number of concurrently processed inputs and weights $N_{ops}$ is 64, which is possible to achieve by configuring the weights and input feature map BRAMs as $512 \times 72$ and using 64 out of the 72 bits of the configured BRAM width.

The memory generator [44] was used to generate the parameters, input feature maps, output feature maps, and each of the weights RAMs. Table VII summarizes the configurations of each RAM generated.

| TABLE VII |
| RAM CONFIGURATIONS SUMMARY. |

| Parameters | # BRAMs | RAM Configuration (depth x width) |
|------------|---------|----------------------------------|
| Input Feature Map | 1 | $512 \times 64$ |
| Output Feature Map | 1 | $512 \times 64$ |
| Weights | 1/PE | $512 \times 64$ |
| Total: $15 + N_{PEs}$ | | |

The maximum possible number of PEs is thus limited by the amount of available BRAMs. Since the target FPGA features 50 BRAMs (detailed in Table II), the maximum number of PEs is given by $N_{PEs_{max}} = 50 - 15 = 35$. However, by inspecting the number of executing sessions it takes to compute each layer, it is possible to reduce the number of PEs to 32, without penalizing performance, as shown in Table VIII.
VI. RESULTS

A. ECG-ID-BNet

This section presents a detailed ECG-ID-BNet architecture hyperparameter search and an accuracy study on two different datasets. Additionally, it explores the binarization, the input quantization and how the presence of BN layers impacts on the accuracy.

Hyperparameter Search: With the results of the hyperparameter search, it was possible to determine the network architecture that produces the best accuracy of 94.4 ± 2.3(%), while complying with the constraints established in Section III-D. Table IX presents the selected architecture for ECG-ID-BNet.

| Layer Type       | # Filters/Neurons | # Sessions Required |
|------------------|-------------------|---------------------|
| Unit 1           | 128               | 2^{N_{PEs}=32} × 2^{N_{PEs}=35} |
| Unit 2           | 64                | 2^{N_{PEs}=32} × 2^{N_{PEs}=35} |
| Unit 3           | 128               | 2^{N_{PEs}=32} × 2^{N_{PEs}=35} |
| Unit 4           | 64                | 2^{N_{PEs}=32} × 2^{N_{PEs}=35} |
| Fully connected  | 50                | 2^{N_{PEs}=32} × 2^{N_{PEs}=35} |

Total: 14^{N_{PEs}=32} × 14^{N_{PEs}=35}

VI. RESULTS

B. Software Implementation

With the objective of implementing the IOb-SoC in the Basys3 development board, a synthesis and implementation were performed, where the SoC’s internal SRAM was pre-loaded with the firmware and the ECG-ID-BNet’s parameters.

The IOb-SoC hardware design was completed and FPGA implementation was conducted. The FPGA resource usage are presented in Table XIII and the clock frequency for the PicoRV32 was finalized at 100MHz. Among the BRAMs used by IOb-SoC, 1 is used for its internal bootloader and the other 32 are used for the SRAM.

| LUTs           | FFs       | 36k BRAMs | DSP Slices |
|----------------|-----------|-----------|------------|
| 2233 (67.6%)   | 1212 (2.9%) | 34 (66%)  | 4 (1.4%)   |

The ECG-ID-BNet was executed in the IOb-SoC and every unit’s execution times were monitored; the results are presented in Table XIV. A compiler optimization was tested (O3)
and it resulted in a 40% execution speedup, when compared to a non-optimized implementation. The highest execution time is observed in the convolutional units, where the program spends 99% of the time.

C. ECG-ID-BNet IP-Core

With the purpose of evaluating the developed IP core, a synthesis was performed. A resource utilization report was made to the four main modules of the IP. The report is represented in Table XV, where the PE cluster is the module containing all $N_{PEs} = 32$ PEs.

It shows that, relatively to the FPGA resources, there is 19% LUT, 10% FF, 94% BRAM and 0% DSP usage. The PE cluster uses on average 103 LUTs and 109 FFs per PE. The input/output memory module, beyond the 2 BRAMs, uses logic resources for the output address calculator. No DSP slice is used because the proposed IP does not require complex computations, such as multiplications or divisions. The control unit uses 543 LUTs for its finite states machines and various counters, of which 258 are dedicated to the parameters control unit and 285 for the execution control unit.

Along the synthesis, it was imposed a clock frequency constraint of 100 MHz as a first proposition and a timing report revealed a Worse Negative Slack (WNS) of 1.1 ns. WNS is the difference between the clock period and the longest delay between a pair of registers. This means the timing constraint imposed is met and the longest path between register has a path delay 1.1 ns shorter than the clock period of the circuit.

A simulation of the IP core execution was made for performance profiling purposes. Table XVI shows the results, where the execution times are presented, per unit, alongside the software only implementation (O3 level optimized) already shown in Section VI-B. Additionally, the speedup that the IP core manages to achieve over the software only execution is also represented. In comparison with other units, the first one achieves half the speed up. This is due to the fact that each PE is only capable to process a single input and weight at a time. Another notable observation is the increased inter-convolution parallelism that the IP core has over the software implementation, since each PE processes 64 inputs and weights at the time, while the other processes 32 inputs and weights at a time. This fact is especially visible on the speedup difference between the first and the other convolutional units, where the first unit’s speedup is half of that of the others.

The last unit is not able to accomplish nearly as much speedup as the rest of the units. This is due to the fact that, contrary to the convolution layers, the fully connected layer’s parameters are numerous (see Table III) and downloading them onto each PE’s internal memory takes about 5× longer than the other layers. The execution of this unit is faster than its parameters downloading process, thus, most of the time spent in this unit the execution is idle.

As stated in Section IV-F, a conventional CNN version of ECG-ID-BNet would take 36.1 s to execute in a software-only paradigm. Thus, we argue that this BCNN accelerator achieves at least a $\frac{36.1}{204.8} = 176270$ speedup over the CNN PicoRV32 software implementation.

VII. CONCLUSIONS

The novel high performance ECG-ID-BNet, a BNN which implements ECG classification was designed, together with an embedded system for ECG identification, which implements ECG-ID-BNet. The main components of such system were developed, namely the ECG sensor, the RISC-V based SoC architecture, and the NN accelerator. The system managed to achieve state-of-the-art accuracies, 100% and 99.3% in the AliveCor and our datasets, respectively. The system speeds up the ECG classification process by a factor of $176270 \times$, over a similar software implementation but with a CNN, non-binarized, version of ECG-ID-BNet, arguably outperforming the current state-of-the-art CNN-based ECG identification methods in terms of execution time.
