Minimizing Classification Energy of Binarized Neural Network Inference for Wearable Devices

Morteza Hosseini, Hirenkumar Paneliya, Utteja Kallakuri, Mohit Khatwani, and Tinoosh Mohsenin
University of Maryland Baltimore County

Abstract— In this paper, we propose a low-power hardware for efficient deployment of binarized neural networks (BNNs) that have been trained for physiological datasets. BNNs constrain weights and feature-map to 1 bit, can pack in as many 1-bit weights as the width of a memory entry provides, and can execute multiple multiply-accumulate (MAC) operations with one fused bit-wise xnor and population-count instruction over aligned packed entries. Our proposed hardware is scalable with the number of processing engines (PEs) and the memory width, both of which adjustable for the most energy efficient configuration given an application. We implement two real case studies including Physical Activity Monitoring and Stress Detection on our platform, and for each case study on the target platform, we seek the optimal PE and memory configurations. Our implementation results indicate that a configuration with a good choice of memory width and number of PEs can be optimized up to 4× and 2.5× in energy consumption respectively on Artix-7 FPGA and on 65nm CMOS ASIC implementation. We also show that, generally, wider memories make more efficient BNN processing hardware. To further reduce the energy, we introduce Pool-Skipping technique that can skip at least 50% of the operations that are accompanied by a Max-Pool layer in BNNs, leading to 37% operation reduction in the Stress Detection case study. Compared to the related works using the same case studies on the same target platform and with the same classification accuracy, our hardware is respectively 4.5× and 250× more energy efficient for the Stress Detection on FPGA and Physical Activity Monitoring on ASIC, respectively.

Keywords— Binarized Neural Networks, Low Energy Wearable Devices, Machine Learning, ASIC, FPGA

I. Introduction

In recent years, the demand for using wearable technology has increased dramatically with the advancement of technology and the growth of the Internet of Things (IoT). Wearable devices are extending their applications in many diverse domains from consumer electronics, such as smartwatches and activity trackers for monitoring one's fitness and wellness, to medical devices that track a patient's vital physiological signs such as heart rate and brain activity. Such devices usually process real-time data, read from multimodal sensors continuously, and suffer from resource-bound and limited battery budget due to their small size, online monitoring, and portability. Therefore, minimizing the power dissipation of these devices while meeting real-time requirements is a subject of interest [1–5].

Deep Neural Networks (DNNs) are among the effective machine learning (ML) approaches that are employed majorly in the processing units of such devices. DNNs can extract featured information from raw multimodal time-series signals without much prior knowledge of the signal [2]. However, DNNs with high-precision weights consume high-energy operations and require large blocks of memories for storing and processing the DNN model. To tackle the problem of memory and power consumption, weight pruning and weight quantization techniques [6–8] as well as low precision weight networks [9–11] have been proposed in recent years as effective approaches to compressing DNN models. [9], [10], and [11] proposed respectively BinaryConnect (BC), Binarized Neural Network (BNN), and Binary-Weight-Networks (BWN), all of which constrain every weight of the neural network to either -1 or $+1$, that can be stored in the smallest possible unit of memory, i.e. 1 bit (0 for -1 and 1 for $+1$), thereby allowing to pack as many weights in as a memory entry can accommodate. In BNNs, not only the weights but the feature-map is also constrained to -1 or $+1$, thus resulting in bit-wise operations between each neuron output and every synapse weight. Due to the minimal aspects of binary weight DNNs, they have gained significant attention recently. BNNs work well on small datasets [9], and will be shown in this work that they can be employed on physiological time-series data as well.

Time-series data classification in ML applications is conventionally processed with techniques such as Dynamic Time Warping (DWT), Recurrent Neural Networks (RNN), and Long Short-Term Memory (LSTM) [12]. These techniques have complicated algorithms and hardware implementation, and mainly deal with complex input data such as text or speech. DNNs, nevertheless, have also been used in the classification of time-series data [2, 13]. In this approach, the real-time data is cut and buffered into frames of samples that may or may not contain specific patterns, e.g. seizure pattern, that annotate the frame with specific labels. The frame of temporal data can be multi-channel and can be used as raw data or processed in the frequency domain to feed a DNN, or represented as an image-like frame whose sequence of image rows follow the sequence of sensors that pick data, e.g. EEG sensors on the skull can represent the brain activity spatially and temporally [14].

In this paper, a low-power scalable hardware is proposed for multimodal physiological data classification that uses BNNs. In its Verilog HDL, the number of PEs and memory width can be adjusted to fit the minimal energy consumption given an application. The hardware is designed to be minimal, and use resource sharing and on-chip memories at its finest.

The main contributions of this paper include:

- Train two physiological case studies including Physical Activity Monitoring, and Stress Detection for BNNs
- Improve BNN accuracy by increasing the number of parameters to achieve the same accuracy as full-precision
- Propose a scalable parallel hardware for BNN inference that can be configured for different number of PEs, and memory width, both of which adjustable for minimal classification energy given a case study.
- Propose an operation skipping technique, referred to as *Pool-Skipping*, for Max-Pool layer in BNNs that can skip more than 50% of effectless operations accompanied with MP and 37% of the BNN operations in total.
- Analyze hardware parameters, including energy vs memory width, and energy vs number of PEs and find the optimal configuration per case study on FPGA and on ASIC.
- Synthesis and Implementation of the platform on FPGA and post place-and-route ASIC layout in 65nm CMOS technology, and provide the energy and time reports of the two case studies.

II. Related Work

Several multimodal data classification methods have been proposed recently. [2] proposed a complexity-reduced Deep Convolutional Neural Network (DCNN) for physiological case studies and a 16-bit hardware platform for their inference. [13] proposed an architecture that uses a CNN per variable and tested their CNN on Congestive Heart Failure Detection and on Physical Activity Monitoring datasets with detection accuracies of respectively 94% and 93%. [15] proposed a scalable system that addresses concurrent activity recognition. Their model extracts temporal and spatial features from a multimodal sensory data by means of a 7 layer CNN followed by an LSTM and is tested on three activity datasets. [16] introduced DeepSense which is a deep learning framework that integrates DNNs and RNNs for addressing feature customization challenges in sensory datasets. [17] investigated the opportunity to use deep learning for identifying nonintuitive features from cross-sensor correlations by means of an RBM. In [1] a kernel decomposing scheme in binary-weight networks is proposed that skips redundant computations and achieves 22% energy reduction on image classification. BiNMAC is proposed in [18] which is a programmable manycore accelerator for BNNs designed for physiological and Image processing case studies.

III. Binarized Neural Networks

BNNs constrain both weights and the feature-map to either -1 or $+1$. Similar to full-precision DNNs, BNNs consist of layers equivalent to those of the standard DNNs, albeit implemented rather differently; for the first layer of BNN, data from each input channel are in full-precision values, but model weights of the BNN are trained and constrained to either -1 or $+1$. Therefore, Multiply-Accumulation (MAC) operations are replaced by a series of ADDs/SUBs. For the next layers, all input features are binarized using binary activation functions (AF) and packed inside registers. Therefore, a MAC operation between two vectors that hold such packed weights can be performed by summing (considering 0 as -1) over the result of the bit-wise *xnor* of the two vectors. The summation over the bits of a register is referred to as $population - count$ and denoted by *pcnt* in this work.

In order to apply the AF to the real-valued variables and transform them to either -1 or $+1$, deterministic binarization function is proposed for AF by [10] which is essentially a *sign* function as shown in equation (1), where x is the real-valued variable and x_b is its binarized value.

$$x_b = \text{sign}(x) = \begin{cases} +1 & if x \geq 0 \\ -1 & otherwise, \end{cases} \tag{1}$$

Thus, the feed-forward path of a layer, that is traditionally denoted with $Y = AF(W.X + B)$ in standard DNNs, can be formulated for BNNs by equation (2):

$$Y = sign(pcnt(xnor(W, X))) \tag{2}$$

Where X and W are respectively tensors of input (feature-map) and weights with bias (B) implicitly included, *xnor* performs bit-wise *xnor* operations on rows from W and columns from X, *pcnt* sums over the result of this bit-wise operation, and *sign* is the AF applied to the generated scalar value from *pcnt*. As explained earlier, from a hardware point of view it is more efficient for both binarized weights and feature-map to be packed in memory entries. For example, one row of a binarized $W_{128\times128}$ can be stored in 4 memory entries of a 32-bit machine, and the whole matrix can be packed inside 512 entries (2KB). Therefore, practically in an M-bit computing machine, the population-count part in equation (2) should be performed as per every chunk of aligned bit-wise *xnor*. Hence, before applying the activation function the population-count needs to be carried out several times until the completion of fetching all patch elements that contribute to generating one output value. This accumulation, before applying the *sign* AF, can be denoted by the following equation [9]:

$$temp+ = pcnt(xnor(W_M, X_M)) \tag{3}$$

Where W_M and X_M are chunks of W and X that are sequentially read from packed memory entries of an M-bit machine until they are fetched completely. Only then one can infer $Y = sign(temp)$ from equations (2) and (3).

For the last layer, weights can either be constrained to -1 or $+1$, that would be treated similarly as previous layers, or trained with full-precision values that, with respect to the binarized activation values coming from the previous layer would convert the MAC operation into ADD/SUB over the weight values, analogous to the first layer. Binarizing the last layer of BNNs can sometimes result in failure and therefore different techniques have been proposed to deal with the last layer [19, 20]. We use high-precision (16-bit) weights for the last layer of the first case study in this paper because it resulted in a stabilized BNN accuracy during trainaing over 100 epochs.

Max-Pool (MP) is a frequently-employed layer in DNNs that sub-samples the feature-map, and is traditionally

placed before AF. It can be shown that if AF is a non-descending function of its input, then placing MP after AF results in equivalent outcome. In BNNs therefore:

$$MP(sign(Y)) = sign(MP(Y)) \quad (4)$$

Placing MP after AF layer in BNNs has two advantages: 1) a bulky comparator that would have to operate on the real-valued results from the previous Conv/FC layer in order to execute MP before AF, can be converted to an *or* gate if MP follows AF with binarized output. Because, if $y_i \in \{0, 1\}$, then $max(y_1, y_2, ..., y_n) = or(y_1, y_2, ...,y_n)$, where y_i and n are elements and the pool size of the MP 2) as soon as one of the elements of the pool, at which the MP seeks the maximum, is equal to the maximum possible value (1 in BNNs), the pool exploring can be asserted as 'done', and the rest of the exploration in the pool can be skipped, that is conducive to skipping of large portion of computation imposed from the preceding Conv/FC layer. We refer to this property as *Pool-Skipping* and in Section VI show that it can skip up to 37% of the total operations for the first case study that has 3 MP layers.

Since in Neural Networks one operation is usually considered as either one multiplication or one addition, therefore each time either a bit-wise *xnor* or a *pcnt* is executed, the number of carried out operations is equal to the number of vector bits to be executed. Consequently in a BNN process, by fusing and well pipelining the *pcnt* and bit-wise *xnor* operations in an M-bit machine, one can achieve 2×M operations per execution of equation 3. This implies that the performance of an M-bit BNN processing machine can be proportional to M, which decides its registers, data bus, memory width, etc. We will evaluate this feature in Section VI.

IV. BNN for Physiological Case Studies

BNN models can compress the model size by 8 to 64 times, at the expense of some accuracy loss. The accuracy loss can usually be compensated by augmenting the BNN model size by 2-11× [20]. In order to present a fair comparison between our hardware and related works, in this section, we introduce two case studies that were used in [2] and [18]. We use Torch framework from [9] to train the BNN models and to grab the weights for inference on hardware.

A. Case study 1: Stress Detection

The Stress Detection dataset [21] includes physiological non-EEG signals that are labeled with four neurological stress status. The DCNN in [2] utilizes a cascade of 5 Conv layers of size 16, 16, 8 and 8 filter banks and 2 FC layers of size 64 and number-of-classes respectively. Input is cut into frames of 64 samples. All filters have a shape of 1×5. We conducted a set of experiments on binarizing this baseline configuration, abiding by the same number of layers, but increasing the number of layer parameters in each experiment. When binarizing the original 16-bit model, the BNN model is compressed by 16 times, but the accuracy drops from the reported 93.8% to 81.8%. In order to compensate the accuracy loss to the baseline level, we increased the number of parameters by approximately 9.5×, and still, the final binarized model is 1.7× smaller than the 16-bit baseline. Fig. 1 shows the incremental augmentation and the impact of binarizing over the baseline DCNN, and the compensation in accuracy with the increase of binarized model size. In total, the binarized DCNN for Stress Detection has 98K parameters and requires 13KB of memory to store the model. For every classification of this dataset, 7.32 Million operations are required. Fig. 2-A shows the BNN configuration and table I summarizes the details for each case study after augmentation.

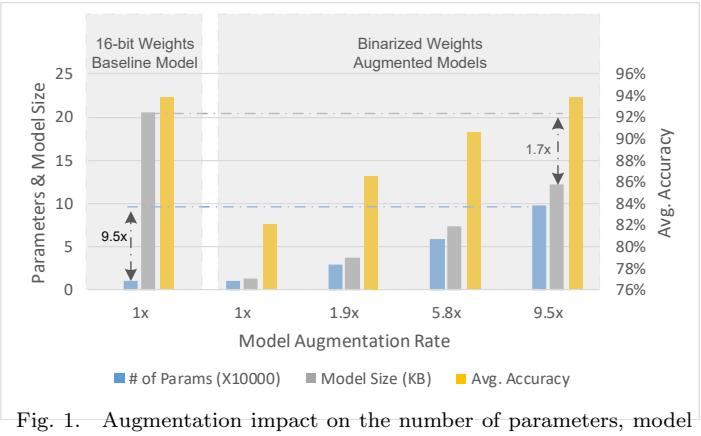

Fig. 1. Augmentation impact on the number of parameters, model size, and accuracy from a 16-bit baseline model for Stress Detection to BNN models

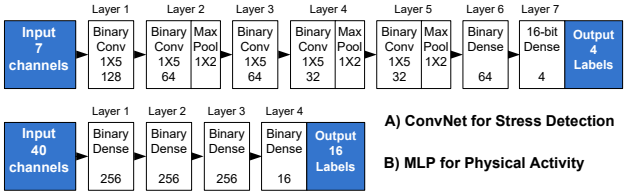

Fig. 2. BNN architectures of the two case studies used in this paper

TABLE I

DETAIL SUMMARY OF THE TWO CASE STUDIES AND THE CHARACTERISTICS OF THEIR BNNS USED IN THIS PAPER

Dataset	Input Shape	# of Labels	# of Params	Model Size	Total Ops	Avg Accuracy
Stress	1×64×7	4	98K	13KB	7.32M	94.1%
PAMAP2	40×1×1	16	145K	18KB	0.29M	97.8%

B. Case study 2: Physical Activity Monitoring

Physical Activity Monitoring dataset (PAMAP2) [22] contains data from 9 subjects recorded from sensors such as IMUs, heart rate monitor that totally has 40 valid channels labeled with 12 physical activities. We use a 3-layer binarized Multi-Layer Perceptron (MLP) as proposed in [18] for classification of the PAMAP2 dataset with an average accuracy of 97.8%. In total, the BNN MLP has 145K parameters, storable in 18KB, and requires 290K operations. Fig. 2-B shows the configuration and Table I summarizes the details.

V. Scalable Hardware Platform

In this section, we introduce a minimal hardware architecture that is scalable with the number of PEs and

memory width that can be adjusted to match the time and energy requirements given an application. For the parallelization scheme, we use output channel tiling as in [23] in which every PE contributes to generating an independent channel of the feature-map. This scheme is shown to result in the best computation to memory communication ratio [24]. It also facilitates implementing the Pool-Skipping method with a simple control logic.

The high-level abstraction of the hardware is depicted in Fig.3. The main components of the hardware include Filter Memory to store the model weights, an Input-Memory and a Feature-Map Memory that are used for the intermediate data and alternately switch their tasks, i.e. buffering the input data/feature-map to be exported to PEs in one memory and importing the output feature-map from the PEs in another memory. All the three mentioned memories are shared between the PEs and addressed by a global controller. The global controller is the only unit of the hardware that uses a multiplier for address calculations.

Each PE comprises a Filter cache to temporarily store an individual filter for the process, and an Output cache to concatenate the single output bits and prepare them to be sent to the Feature-Map Memory along with other PEs. Once this cache is full, all the PEs concatenate their first bit of output-cache and send the packed packet out to the feature-map memory. Then, next bits are sent out accordingly. Resource sharing is carefully taken into consideration such that the major logic of every PE is only a pipeline of bit-wise *xnor*, *pcnt*, and an accumulator to satisfy the equation (2). The accumulator ADDs/SUBs over either the input data w.r.t binarized packed filter for the first BNN layer, or on the output of *pcnt* for mid-layers, or over the filter data w.r.t binarized packed data for the last BNN layer. Two shifters, not depicted in Fig. 3 to avoid obfuscation, shift the packed filter/data for the first/last layers. A local controller, by means of Multiplexers to allocate the existing resources, handles the data-flow such that FC and Conv layers with different strides are executed with respect to the BNN layer.

A bit-wise *or* gate is implemented after the Output cache to perform MP and is by-passed if there is no MP. Pool-Skipping is implemented in such a way that, given MP, whenever there is a +1 in the pool-under-explore, +1 is written quickly to its relevant location in the Output cache, the pipeline is flushed, and the operation is skipped to a stride as wide as the pool size. The would-be adjacent bits in the Output cache are left as 'don't care' as their old values will be masked by the effect of the written +1.

VI. Evaluation

A. Parallelization

As mentioned in Section III, BNN inference performance in our M-bit datawidth machine is proportional to M, omitting the first/last layers that may use full-precision values. Meanwhile, with the split and tiled threads dispatched to N concurrent PEs, the computation can be expedited by N times if the memory communication is disregarded. Clock frequency, f, is another factor that decides the computation rate. Therefore in our hardware, the performance is proportional to these three metrics:

$$Performance \propto f \times M \times N \tag{5}$$

The performance of the first and the last layer, similar to traditional accelerators, is proportional to $f \times N$ only.

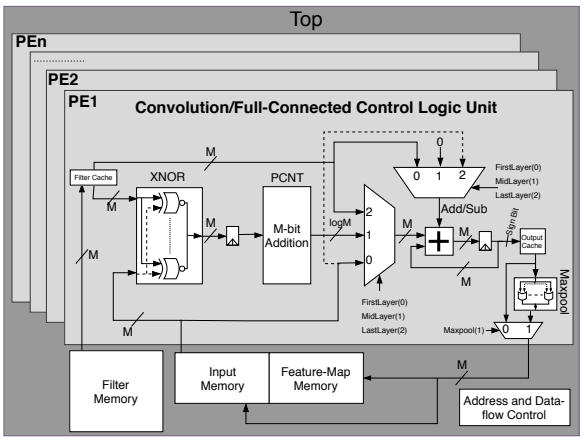

Fig. 3. Block diagram of hardware architecture which comprises Filter, Input, and Feature-Map memories that are addressed by a global controller, and are shared between PEs. Each PE includes a filter and output cache, a pipeline of bit-wise *xnor*, *pcnt* and accumulator, and by means of MUX and a local controller performs first/mid/last BNN layer with respect to Conv/FC operations. A bit-wise *or* gate is also embedded to perform MP.

B. Memory Energy Evaluation

We will seek a trade-off between M and N in the next section for minimizing the energy consumption, but beforehand, we analyze a set of ARM memories generated with Artisan Memory Compiler in 65nm and at frequency of 128MHz with various datawidth. For each generated memory we registered the energy per entry read/write, and

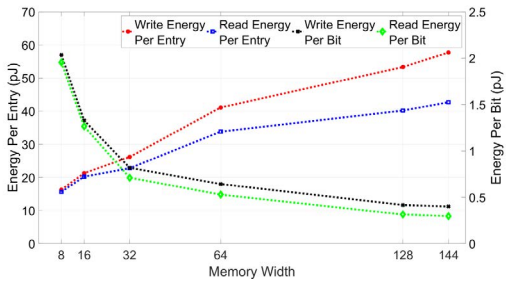

Fig. 4. Energy curves inferred from the technical data sheets of ARM memories showing the trend for read/write per entry and per bit on increasing ARM Memory Width

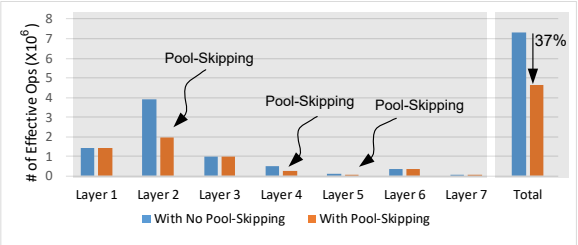

Fig. 5. Impact of Pool-Skipping on the number of effective operations for the Stress Detection case study

TABLE II

IMPLEMENTATION RESULTS OF THE PROPOSED BNN PROCESSOR ON XILINX FPGA (ARTIX-7). THE RESULTS OBTAINED AT THE CLOCK FREQUENCY OF 100 MHZ AND WITH MEMORY WIDTH OF 128 BITS

Case Studies / Implementation Characteristics	Stress Detection			Physical Activity Monitoring		
	Serial	Semi Parallel	Fully Parallel (Opt.)	Serial	Semi Parallel (Opt.)	Fully Parallel
# of PEs	1	4	8	1	4	8
BRAM	6	6	6	3	3	3
DSP	1	1	1	1	1	1
# of slices	341	1076	2047	385	1329	2601
Latency (ms)	1.14	0.31	0.19	0.06	0.03	0.03
Throughput(k label/s)	0.87	3.23	5.22	17.17	36.15	38.12
Total Power (mW)	78	98	115	76	79	91
Energy (uJ)	89.3	30.3	22.0	4.4	2.2	2.4

calculated the energy per bit read/write. Fig. 4 plots the results of this analysis and indicates that it is more energy efficient to read 1 bit of information, as a BNN weight, if wider memories are used.

C. Pool-Skipping

As explained in Section III, Pool-Skipping can skip a large portion of operations that are preceding a MP. For the Stress Detection case study, 3 MPs with pool size of 1×2 follow 3 Conv layers that contribute to 76.5 % of the total operations. Fig. 5 shows the impact of Pool-Skipping on the number of operations per layer and in total for the BNN of Stress Detection. Applying Pool-Skipping, the average number of effective operations is reduced by 37%.

VII. Implementation Results and Analysis

A. FPGA Implementation

In order to emulate the case studies on the proposed hardware, FPGA is used for the evaluation setup. We choose the low-power Artix-7 family and select the smallest package whose Block RAMs (BRAMs) can suffice the model of the case studies. Each case study was synthesized, placed and routed and implemented using Xilinx Vivado. with different numbers of PEs and different width for BRAMs. Fig. 6-(left) depicts the classification energy for the Stress Detection vs number of PEs w.r.t varying memory widths, and Fig. 6-(right) depicts classification energy vs Memory width with varying number of PEs for Physical Activity. Table II provides the implementation results for the two cases with memory width of 128 bits, at 100MHz and with more implementation details. In Table II, for each case study the optimal (Opt.) configuration in terms of energy is given for a specific number of PEs. For the Stress Detection, the optimal config. (8PEs) requires 4× less energy, and for the Physical Activity, the optimal config.(4PEs) consumes 2× less energy as compared to their serial configs (1PE).

B. ASIC Implementation

Various configurations per case study were synthesized using RC Compiler and post place-and-routed using the Encounter SOC from Cadence using 65nm standard cell library CMOS technology. For the two case studies we obtained slightly different optimal configs per case study on ASIC. Fig. 7-(left) and Fig. 7-(right) respectively depict the classification energy for the first and second case studies

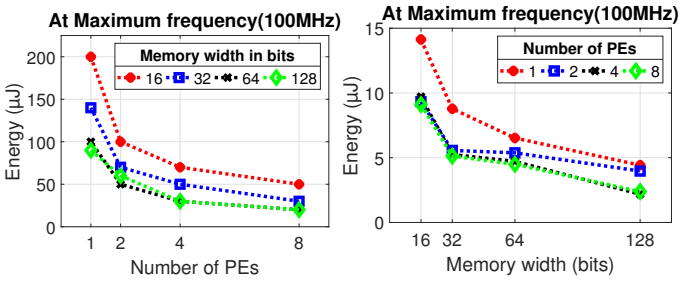

Fig. 6. (left) impact on the classification energy with the increase of the number of PEs for four different Memory widths for the Stress Detection (right) impact on classification energy with increasing the number of Memory width for 4 different PEs for the Physical Activity Monitoring on the FPGA

TABLE III

POST PLACE AND ROUTE RESULTS OF THE PROPOSED BNN PROCESSOR ON ASIC CMOS 65NM 1.1V. THE RESULTS OBTAINED AT CLOCK FREQUENCY OF 128 MHZ AND WITH A MEMORY WIDTH OF 128 BITS

Case Studies / Implementation Characteristics	Stress Detection			Physical Activity Monitoring		
	Serial	Semi Parallel (Opt.)	Fully Parallel	Serial	Semi Parallel	Fully Parallel (Opt.)
# of PEs	1	4	8	1	4	8
Area Util. (%)	94	91	90	94	93	92
Clock Freq. (MHz)	128					
Max Freq. (MHz)	1000					
Core Area (mm2)	0.49	0.56	0.66	0.18	0.19	0.20
Latency (ms)	2.13	0.55	0.29	0.04	0.018	0.014
Throughput(k label/s)	0.47	1.80	3.42	22.22	52.81	68.52
Leakage power (mW)	5.77	6.42	7.27	3.61	3.66	3.71
Dynamic power (mW)	5.37	13.83	32.85	0.80	1.23	1.62
Total power (mW)	11.14	20.25	40.13	4.41	4.90	5.32
Energy (uJ)	23.81	11.25	11.73	0.20	0.09	0.077

w.r.t. the varying PEs and varying memory widths, and Table III provides the implementation results with memory width 128 bits, at 128MHz in more details. Both the Figure and the table confirm energy per bit analysis inferred from Fig. 4. From table III it is concluded that for the Stress Detection and Physical Activity, the optimal configs are respectively 2× and 2.5× more energy efficient than their serial config counterpart. The ASIC post-layout views for the optimal configs of both cases are shown in Fig. 8. The reason for the Physical Activity case to have smaller PE logic is that in its HDL the MP and FC logics are automatically removed and less number of SRAM cells have been dedicated as the cache memories.

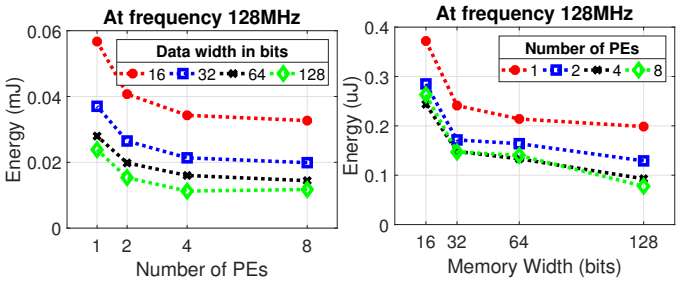

Fig. 7. (left) impact on classification energy with increase of the number of PEs for four different Memory widths for Stress Detection (right) impact on classification energy with increasing the Memory widths for 4 different PEs for Physical Activity on a 65nm ASIC

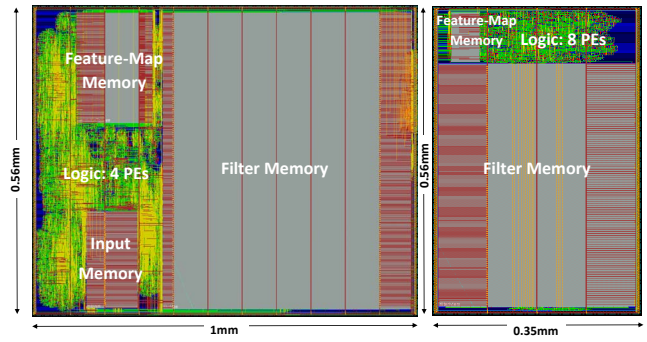

Fig. 8. Post-layout views of BNN for Stress Detection (left) and for Physical Activity (right), in 65nm TSMC CMOS technology.

VIII. Comparison

We compare our proposed hardware on the two case studies with two relevant works, [2] and [18], from which we obtained our BNN configurations. The comparison is summarized in Table IV. In summary, despite 9.5× augmentation in the number of parameters, the optimal config. for the Stress Detection is 4.5× and 6.8× more energy efficient as compared to [2] implemented respectively on Artix-7 FPGA and on ASIC 65nm. For the Physical Activity, the minimal hardware with no MP and Conv logics, is 250× more energy efficient as compared to a programmable, yet low-power manycore platform in [18].

TABLE IV
COMPARISON OF THE PROPOSED HARDWARE WITH OPTIMIZED CONFIGS WITH RELATED WORKS

Metrics	[2]		**This Work**		[18]	**This Work**
Application	Stress Detection				Physical Activity	
Technique	16-bit DCNN		BNN		BNN	BNN
Accuracy(%)	94		94.1		97.8	97.8
Platform	FPGA Artix-7	ASIC 65nm	FPGA Artix-7	ASIC 65nm	ASIC 65nm	ASIC 65nm
Freq. (MHz)	100	100	100	128	145	128
Latency (us)	1000	800	190	550	77	14
Power (mW)	154	60	115	22	270	5.3
Energy (uJ)	150	50	22	11	20	0.08
Energy Imp.	-	-	**6.8×**	**4.5×**	-	**250×**

IX. Conclusion

We proposed a scalable hardware for inference of BNNs that have been trained for physiological datasets. The proposed hardware has flexibility in adjusting the memory width and number of PEs, both adjustable for the most energy efficient configuration while meeting time requirements given an application. Two case studies including Physical Activity Monitoring and Stress Detection are trained and deployed on the hardware, and for each, the optimal number of PEs and memory-width is sought. Our implementation results with the two case studies on Artix-7 FPGA and on 65nm ASIC CMOS standard cell library indicate that the configuration parameters of the hardware, if adjusted well, can optimize the energy consumption up to 4× and 2.5× respectively on FPGA and on ASIC. An operation skipping method, named Pool-Skipping, is also proposed for BNNs that skips effectless operations that precede a Max-Pool layer, and can skip up to 37% of operations in the Stress Detection case study. When compared to the related works that use the same case studies on the same target platforms, FPGA and ASIC, and with the same classification accuracy, our hardware is respectively 4.5× and 250× more energy efficient for the Stress Detection on FPGA and Physical Activity Monitoring on ASIC.

References

[1] H. Kim *et al.*, "A kernel decomposition architecture for binary-weight convolutional neural networks."

[2] A. Jafari *et al.*, "Sensornet: A scalable and low-power deep convolutional neural network for multimodal data classification," *IEEE TCAS-I: Regular Papers*, 2018.

[3] A. Page, A. Kulkarni, and T. Mohsenin, "Utilizing deep neural nets for an embedded ecg-based biometric authentication system," in *BioCAS*. IEEE, 2015.

[4] T. Abtahi *et al.*, "Accelerating convolutional neural network with fft on tiny cores," in *ISCAS*. IEEE, 2017.

[5] T. Abtahi, C. Shea, A. Kulkarni, and T. Mohsenin, "Accelerating convolutional neural network with fft on embedded hardware," *IEEE Transactions on VLSI Systems*, 2018.

[6] W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in *International Conference on Machine Learning*, 2015.

[7] J. Shen and S. Mousavi, "Least sparsity of p-norm based optimization problems with p¿1," *SIAM Journal on Optimization*, 2018.

[8] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[9] M. Courbariaux *et al.*, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv preprint arXiv:1602.02830*, 2016.

[10] ——, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in Neural Information Processing Systems*, 2015, pp. 3123–3131.

[11] M. Rastegari *et al.*, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 525–542.

[12] F. Karim *et al.*, "Lstm fully convolutional networks for time series classification," *IEEE Access*, 2018.

[13] Y. Zheng *et al.*, "Time series classification using multi-channels deep convolutional neural networks," in *International Conference on Web-Age Information Management*. Springer, 2014, pp. 298–310.

[14] M. Soleymani *et al.*, "Analysis of eeg signals and facial expressions for continuous emotion detection," *IEEE Transactions on Affective Computing*, no. 1, 2016.

[15] X. Li *et al.*, "Concurrent activity recognition with multimodal cnn-lstm structure," *arXiv preprint arXiv:1702.01638*, 2017.

[16] S. Yao *et al.*, "Deepsense: A unified deep learning framework for time-series mobile sensing data processing," in *Proceedings of the 26th International Conference on World Wide Web*, 2017.

[17] V. Radu *et al.*, "Towards multimodal deep learning for activity recognition on mobile devices," in *UbiComp*. ACM, 2016.

[18] A. Jafari *et al.*, "Binmac: Binarized neural network manycore accelerato," in *ACM Proceedings of the 28th Edition of the Great Lakes Symposium on VLSI (GLSVLSI)*. ACM, 2018.

[19] W. Tang, G. Hua, and L. Wang, "How to train a compact binary neural network with high accuracy?" in *AAAI*.

[20] Y. Umuroglu *et al.*, "Finn: A framework for fast, scalable binarized neural network inference," in *Proceedings of the SIGDA*. ACM, 2017.

[21] J. Birjandtalab *et al.*, "A non-eeg biosignals dataset for assessment and visualization of neurological status," in *Signal Processing Systems (SiPS)*. IEEE, 2016.

[22] A. Reiss and D. Stricker, "Introducing a new benchmarked dataset for activity monitoring," in *Wearable Computers (ISWC), 2012 16th International Symposium on*. IEEE, 2012.

[23] A. Page *et al.*, "Sparcnet: A hardware accelerator for efficient deployment of sparse convolutional networks," *Journal of Emerging Technologies in Computing Systems*, 2017.

[24] C. Zhang *et al.*, "Optimizing fpga-based accelerator design for deep convolutional neural networks," in *Proceedings of SIGDA*. ACM, 2015.