energies

MDPI

Article

AHEAD: Automatic Holistic Energy-Aware Design Methodology for MLP Neural Network Hardware Generation in Proactive BMI Edge Devices

Nan-Sheng Huang [1,*], Yi-Chung Chen [2], Jørgen Christian Larsen [1] and Poramate Manoonpong [1]

1 Embodied AI and Neurorobotics Laboratory, SDU Biorobotics, Mærsk Mc-Kinney Møller Institute, University of Southern Denmark, 5230 Odense, Denmark; jcla@mmmi.sdu.dk (J.C.L.); poma@mmmi.sdu.dk (P.M.)

2 Department of Electrical and Computer Engineering, Tennessee State University, Nashville, TN 37209, USA; ychen@tnstate.edu

* Correspondence: nan@mmmi.sdu.dk

Received: 23 March 2020; Accepted: 22 April 2020; Published: 1 May 2020

Abstract: The prediction of a high-level cognitive function based on a proactive brain–machine interface (BMI) control edge device is an emerging technology for improving the quality of life for disabled people. However, maintaining the stability of multiunit neural recordings is made difficult by the nonstationary nature of neurons and can affect the overall performance of proactive BMI control. Thus, it requires regular recalibration to retrain a neural network decoder for proactive control. However, retraining may lead to changes in the network parameters, such as the network topology. In terms of the hardware implementation of the neural decoder for real-time and low-power processing, it takes time to modify or redesign the hardware accelerator. Consequently, handling the engineering change of the low-power hardware design requires substantial human resources and time. To address this design challenge, this work proposes AHEAD: an automatic holistic energy-aware design methodology for multilayer perceptron (MLP) neural network hardware generation in proactive BMI edge devices. By taking a holistic analysis of the proactive BMI design flow, the approach makes judicious use of the intelligent bit-width identification (BWID) and configurable hardware generation, which autonomously integrate to generate the low-power hardware decoder. The proposed AHEAD methodology begins with the trained MLP parameters and golden datasets and produces an efficient hardware design in terms of performance, power, and area (PPA) with the least loss of accuracy. The results show that the proposed methodology is up to a 4X faster in performance, 3X lower in terms of power consumption, and achieves a 5X reduction in area resources, with exact accuracy, compared to floating-point and half-floating-point design on a field-programmable gate array (FPGA), which makes it a promising design methodology for proactive BMI edge devices.

Keywords: neural network; edge device; field-programmable gate array; hardware acceleration; high-level synthesis; energy-aware design; brain–machine interface

1. Introduction

A brain–machine interface (BMI) is a direct and unconventional communication link between the brain and a physical device [1,2]. It enables the study of neuronal activity, representing cognitive processes of planning and mental simulation of action sequences. BMI is classified into reactive and proactive (cognitive) types. The reactive mode denotes actions generated by flexibly responding to the environmental stimuli from sensory inputs in unpredictable situations. The proactive mode, on the

other hand, denotes actions that are generated to accomplish intended goals based on active intentions in predictable situations. In the field of cognitive neuroscience, BMI introduces a new opportunity for smart environmental control by human–environment interaction.

Multifunctional Brain–Computer Interface (BCI) [3] and electroencephalogram (EEG) technology have been adopted as the BMI for applications of environmental control [3], accelerated information seeking [4], and neuro-controlled upper limb prosthesis [5]. However, the biomedical signals of these applications do not provide cognitive properties for proactive control. The SI-CODE [6] introduced a bidirectional BMI for decoding multiunit neural activity and transmitting information back to the brain by electrical stimulation. Although the multiunit activities do provide rich cognitive properties of the brain, the project does not take advantage of the cognitive properties. In the industry, big tech companies such as Microsoft, Facebook, and Neuralink are also researching next-generation BMI products [7–9]. However, these research projects are reactive, such that reflexive and cumbersome behaviors dominate the outcomes, instead of primary cognitive (proactive) human traits—e.g., planning and mental simulation of actions—where a large amount of interest lies. Plan4Act (In Plan4Act, the main goal is to exploit the neuronal signatures of upcoming, planned actions of an agent to proactively support it during the execution of these forthcoming activities. P.M. is the PI of the work package 3 in Plan4Act. J.C.L and N.S.H are responsible for the research development of the energy-efficient neural decoder hardware accelerator in Plan4Act.) [10] is a unique project for studying proactive predictions of future planned actions with online decoding. Its uniqueness lies in the fact that it has a proactive BMI composed of wireless data acquisition for receiving neural brain activity [11] and a neural network decoder for processing the recorded sequence-predicting neural activity and inferring predicted action sequences for proactive control in real time.

The development of a neural network decoder for proactive BMI control has two major challenges: adaptivity and power consumption. As the implanted floating microwire arrays (FMAs) move relative to the recorded cells and the information represented by a specific neuron's activity may change due to neuroplasticity, the decoder from day one cannot be leveraged to work in different experimental sessions on the day n later [12,13]. As a result, the neural network decoder needs to adapt to the above situations on day n with a modification of the topology and a retrain procedure for recalibration. The modification related to neural network implementation is a time-consuming process leading to a potential delay for the whole project. On the other hand, the neural decoder of the BMI is a battery-powered real-time embedded system, which requires a highly energy-efficient computing unit [14] for irregular parallelism and custom data types [15]. To address the above challenges, it is believed that the neural network decoder on a field-programmable gate array (FPGA) with a concurrent optimization of performance, power, and area (PPA) [14] is a promising solution. In [16], two high-performance hardware coprocessors for the probabilistic neural network and neural ensemble decoding based on Kalman filter were implemented on FPGA for real-time BMI applications. Apart from that, it is well known that fixed-point implementation consumes fewer resources and less power than floating-point implementations in FPGA literature [15,17].

To address the aforementioned challenges, an automatic holistic energy-aware design (AHEAD) methodology is introduced for the design automation of an energy-efficient neural network decoder for the proactive BMI. The core processing component of the neural network decoder is a multilayer perceptron [18] (MLP) inference that has features of uniformity, innate parallelism, scalability, dynamically adaptivity, and fault tolerance [19–21]. The MLP will be trained by the golden datasets for the initial weights. AHEAD extracts the network feature from MLP and creates the corresponding hardware model and test bench. AHEAD adopts fixed-point number representation for the hardware implementation of the MLP because of the comparable accuracy in finite-precision performance [22], shorter latency, reduction of logic, memory area, and power consumption on an FPGA [17,23] compared to its floating-point implementation. Since the floating-point to fixed-point conversion is a nonconvex NP-hard optimization problem [24] and requires language-specific programs [25–28], AHEAD develops a bit-width identification (BWID) loop method for the tedious conversion because

of the close synergy between fixed-point parameter estimation and configurable hardware generation. The BWID automatically estimates the required fixed-point bit-width parameters with the least loss of accuracy and bit-width through the reconstruction of the given MLP neural network from the MLP parameters and golden datasets without the user's program code. To simplify and accelerate the system implementation on an FPGA, AHEAD encompasses a high-level synthesis (HLS) [29] design flow that automatically generates a register-transfer level (RTL) for the PPA-optimized system with fixed-point bit-width MLP, pipelines, and parallel low-power microarchitectures.

The AHEAD methodology can implement an energy-efficient MLP hardware accelerator, including integration with the embedded processor as a full system, within an hour. Not only is the development effort minimized, but also the development time is significantly reduced from several days to an hour. Furthermore, the generated design, without a loss of accuracy, is about 4X faster in execution time, 5.97X better in energy efficiency, 3X lower in slice look-up tables (LUTs), 8X lower in slice registers, 243X lower in DSP48Es, and 5X lower in block rams (BRAMs) compared to the floating-point implementation in the experiment. Thus, the AHEAD methodology can deliver a rapid, low-power design and implementation of the MLP neural decoder to meet the power requirements [30] for the FPGA implementation for BMI applications.

The contributions of this paper to the problem of the design of the low-power MLP hardware accelerator for proactive BMI control edge devices are as follows:

1. A novel holistic design methodology, for the first time, bridges the gap between the BMI developers and the hardware developers for automatic energy-aware MLP hardware generation with trained MLP parameters and golden datasets.
2. An energy-aware MLP hardware generation for proactive BMI control with automatic nonuniform fixed-point bit-width identification capabilities.
3. Fully automatic methodology frees the resources of domain experts across the developers to do the iterative, tedious, labor-intensive, error-prone floating-to-fixed point conversion and low-power hardware design task.
4. The design methodology is independent of machine learning tools and programming languages.

The rest of this paper is organized as follows. Section 2 describes the background of the system architecture of the proactive BMI control and the MLP neural network decoder. Section 3 presents the new holistic design methodology for the low-power MLP hardware design in the proactive BMI control, which includes a description of the solution methodology and lists the main architecture of the framework. Section 4 elaborates on the implementation of the methodology, which comprises the automatic energy-aware MLP hardware generation. Section 5 presents the results of the benchmarking cases with comparisons in terms of accuracy, power, performance, and area. Section 6 provides discussions regarding future work.

2. Background of Proactive BMI Control

This section presents an overview of the proactive BMI system architecture, system requirements, the hardware design challenge, the role of the neural decoder, and the MLP neural network.

2.1. Plan4Act System Architecture

Plan4Act [10] is a European project to proactively predict the future planned actions of an agent by extracting this planning knowledge from the agent's neuronal activities. The basis for this is recent experimental results that show that complex planning and sequencing information is represented by neural activity in the (monkey) brain [31,32]. These investigations are further extended and transferred to a BMI setup in terms of a wearable or Medical Internet of Things (IoT) edge device for controlling devices with more foresight than in currently existing systems. Therefore, through the development of such a proactive BMI edge device, a future path for people with disabilities to interact with their (smart) environment in a more robust way is opened up for the first time. Figure 1 shows a scenario

considered in the Plan4Act: a motor-impaired patient initiates a thought to go to the toilet from the bedroom; A denotes the action to open the door of the bedroom; B is the action to turn on the light of the living room; C_1 denotes the action to open the door and turn on the light in the toilet; and C_2 represents the action to open the door and turn on the light on the terrace. Action sequences AB may lead to the different following actions C_1 or C_2, since the common path is AB.

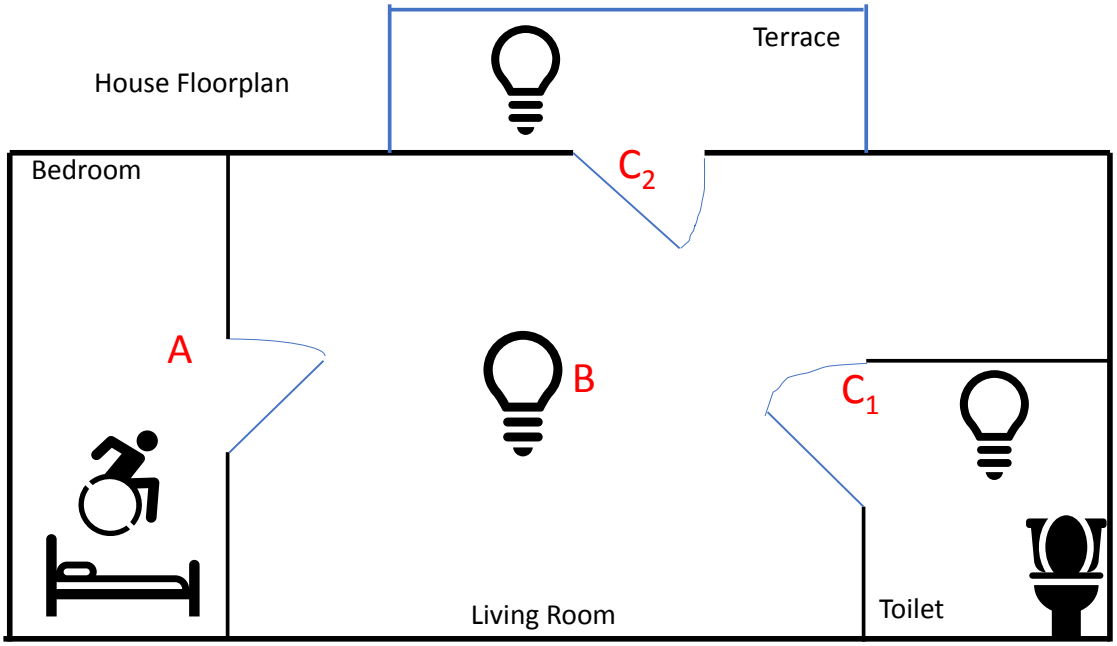


Figure 1. Proactive brain–machine interface (BMI) control scenario.

The Plan4Act project aims to develop the novel proactive BMI control to predict the desired action sequences ABC_1 while acquiring the neuronal data of either action A or actions AB, and to avoid the false case ABC_2. It is different from reactive control whereby the ABC_1 actions are executed in sequence; herein, the actions are translated into a complex proactive BMI control problem in sequence: the BMI needs to acquire, analyze, and identify sequence-predicting neuronal activity in the brain while executing the tasks; the mathematical models based on the interaction of neuronal activity and plasticity mechanisms are developed to understand this sequence-predicting neural activity and to provide the algorithmic basis of neural signal decoder design.

Figure 2 shows a block diagram of the complete system architecture. The Plan4Act system architecture of the proactive BMI control features two principal subsystems, namely, a wireless transmit-and-receive neural recording subsystem for data acquisition of neuronal activity in the brain, and a neural decoding-and-control subsystem for feature extraction, classification of the recorded sequence-predicting neural activity, and further proactive control of smart devices in the home environment.

As illustrated at the top of Figure 2, the neuronal data are measured and collected through wireless neural recordings during animal experiments [11]. The neuronal activities are from 192-channel FMAs implanted in three different brain areas of the rhesus macaque: the parietal reach region (PRR), the dorsal premotor cortex (PMd), and the primary motor cortex (M1). The processing flow of the proactive BMI control is shown at the bottom of Figure 2. The neuronal data are transmitted wirelessly to the processing unit of the proactive BMI. The data are first filtered by a front-end signal processing unit for data acquisition and synchronization and then further decoded by a neural signal decoder for predicting proactive behaviors.

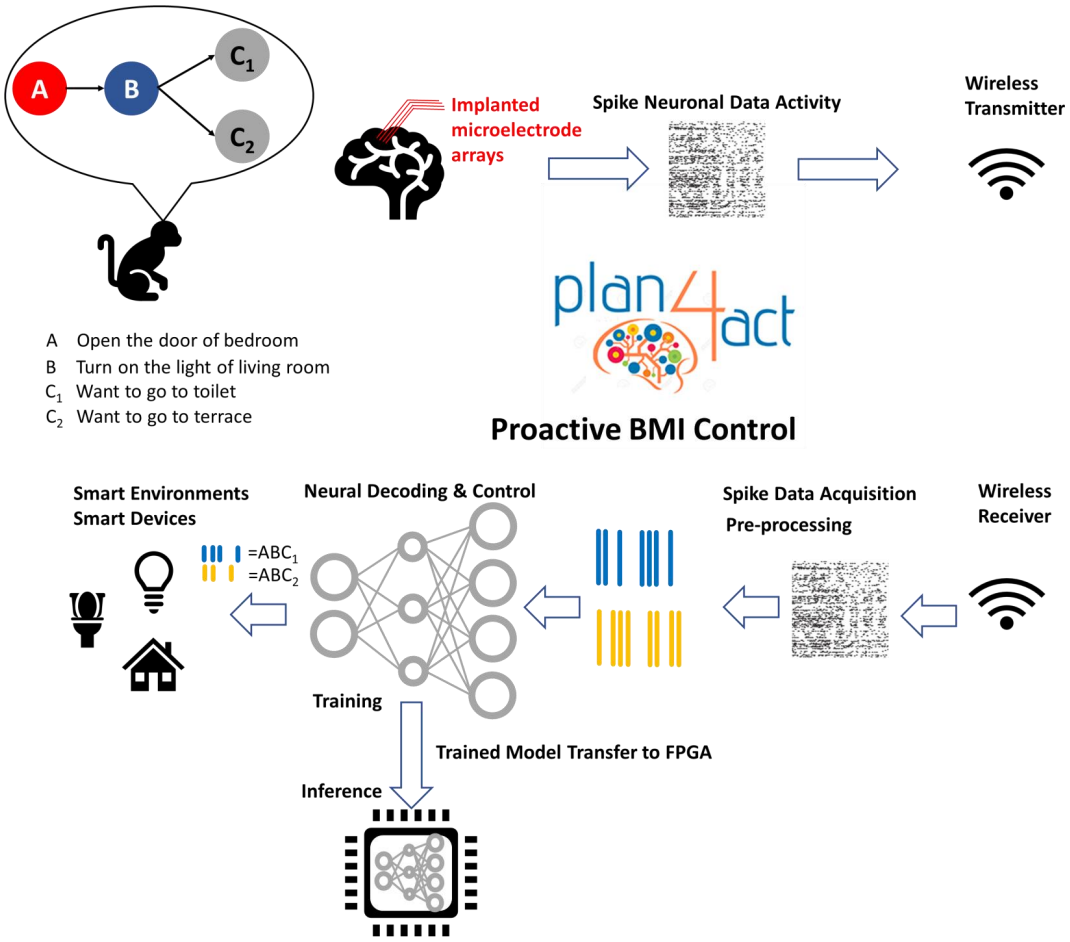


Figure 2. Proactive brain–machine interface (BMI) system architecture. FPGA—field-programmable gate array.

Various architectures have been studied for the design of the neural signal decoder during the phases of hardware development in the Plan4Act system. MLP neural network based architecture is inspired by biological neurons [20] and has features of uniformity, innate parallelism, scalability, dynamic adaptivity, and fault tolerance, which specifically fit the requirements of the proactive BMI edge devices. The training of the MLP neural network is offline and in the form of supervised learning with the dataset collected from experiments [11]. The trained model is used for the development of the neural decoder, which is a real-time, low-power inference on an edge device. This neural decoder is the main processing unit of the proactive BMI edge device and can robustly extract and predict sequence-predicting neural activity for real-time proactive control. However, the decoder requires regular recalibration through a retrain procedure since the implanted electrodes embedded in the brain may move relative to the neurons. In addition, neurons exhibit the characteristic of neuroplasticity, which changes the dimensionality of neuronal dynamics in the latent space [12,13]. Figure 3 shows the hardware design flow and challenges of the neural decoder design. The recalibration and model retraining resulting from the abovementioned issues means the neural decoder must be constantly redesigned and reimplemented, which is time-consuming and demands a lot of effort. In this work, the proposed AHEAD methodology addresses the problem by introducing fully automatic design automation.

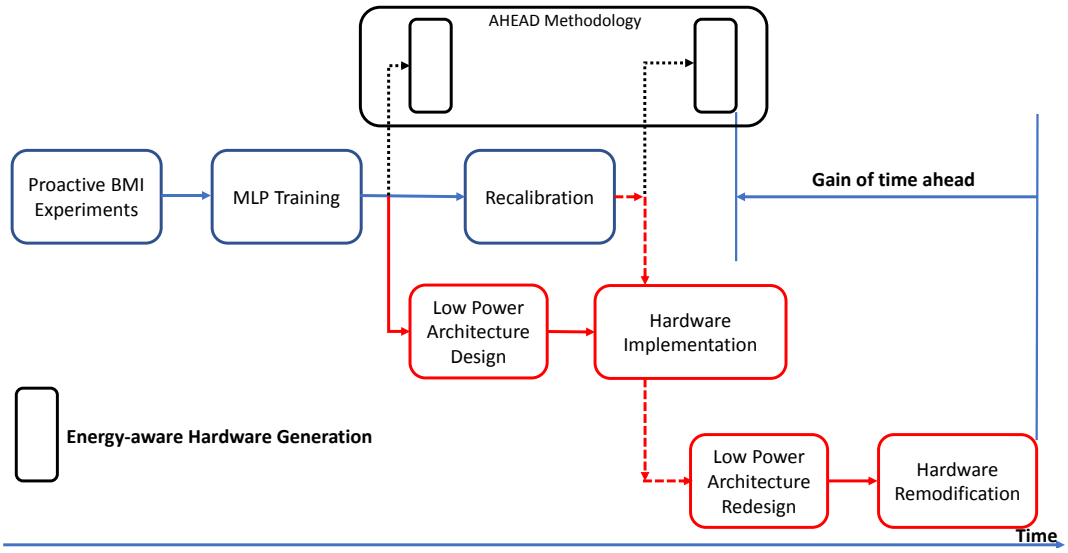


Figure 3. Hardware design challenge in proactive BMI control.

2.2. MLP Network of the Neural Decoder

Figure 4 demonstrates a four-layer, fully connected MLP with 768 neurons in the input layer, 48 neurons in the 1st hidden layer, 20 neurons in the 2nd hidden layer, and seven neurons in the output layer, denoted as 768-48-20-7. The linear, hyperbolic tangent, and sigmoid activation functions are applied in hidden layer one, hidden layer two, and the output layer, respectively.

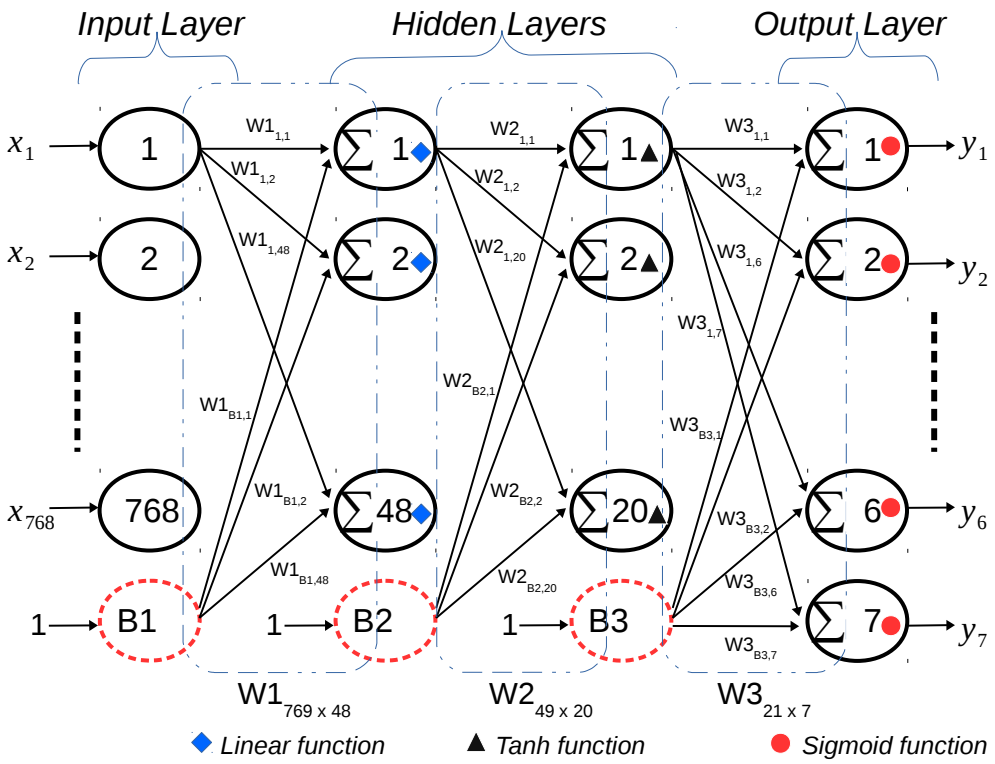


Figure 4. Multilayer perceptron (MLP) structure.

By default, each layer equips itself with one bias neuron as default, receiving the input value '1' except in the output layer. The input neurons use a linear activation function, which linearly transmits the external inputs to the subsequent layer without any further operation. Furthermore, the essential operation of each neuron is to multiply and accumulate all the weighted inputs with weight values and then perform the activation function for further output. As an example of the input to the first hidden layer, the input vector X has 769 elements with one bias neuron $[x_1 x_2 x_3 \cdots x_{767} x_{768} 1]$, which are multiplied by $W1$ to form vector $Y1$, as shown in Equation (1):

$$Y1 = X \cdot W1, \tag{1}$$

$$W1_{769,48} = \begin{pmatrix} w1_{1,1} & w1_{1,2} & \cdots & w1_{1,48} \\ w1_{2,1} & w1_{2,2} & \cdots & w1_{2,48} \\ \vdots & \vdots & \ddots & \vdots \\ w1_{768,1} & w1_{768,2} & \cdots & w1_{768,48} \\ w1_{B1,1} & w1_{B1,2} & \cdots & w1_{B1,48} \end{pmatrix}. \tag{2}$$

The output vector of the first hidden layer $Z1$ is the result of applying the activation function f on $Y1$, as illustrated in Equation (3):

$$Z1 = f(Y1). \tag{3}$$

The same procedures are also applied to all the other layers. Moreover, to reduce the number of memory access operations for lower power, Equation (1) can be further split as follows:

$$Y1 = X \cdot W1 = X' \cdot W1' + WB1', \tag{4}$$

$$W1'_{768,48} = \begin{pmatrix} w1_{1,1} & w1_{1,2} & \cdots & w1_{1,48} \\ w1_{2,1} & w1_{2,2} & \cdots & w1_{2,48} \\ \vdots & \vdots & \ddots & \vdots \\ w1_{768,1} & w1_{768,2} & \cdots & w1_{768,48} \end{pmatrix}, \tag{5}$$

where X' is $[x_1 x_2 x_3 \cdots x_{767} x_{768}]$ and $WB1'$ is $[w1_{B1,1} w1_{B1,2} \cdots w1_{B1,48}]$ due to the fact that the bias neuron has a constant input value of 1.

In addition, the MLP parameters include the number of layers, the number of neurons per layer, the type of activation function per layer, and trained weight matrix files. They are capable of reconstructing the MLP neural network, which is like the eigenvalue and eigenvector of a matrix.

3. The AHEAD Methodology

This section depicts the detailed methodology of AHEAD in the Plan4Act architecture from the system point of view. By employing the attribute of fault tolerance in the MLP neural decoder along with the bit-width identification method, the low-power fixed-point parameters are automatically estimated. Then, the energy-efficient hardware is generated from the configurable hardware generator.

3.1. AHEAD—System Overview

The proposed AHEAD methodology illustrated in Figure 5 takes holistic thinking from the reconfigurable requirement of MLP hardware configuration that arises from the recalibration-based BMI neural decoder development to the final automatic delivery of low-power MLP neural decoder. The architecture is defined by the leverage of concurrent design reuse for the hardware generation and energy-aware design automation framework for microarchitectures with cross-layer optimization.

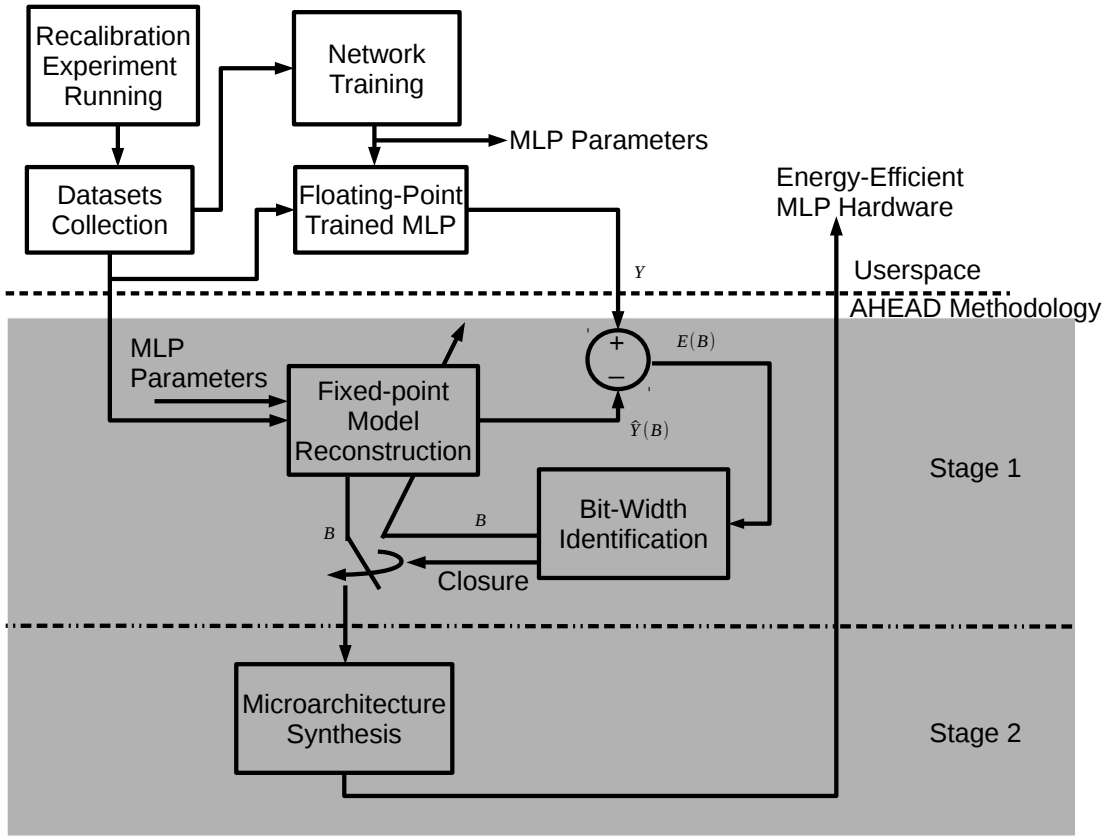


Figure 5. The automatic holistic energy-aware design (AHEAD) methodology.

As demonstrated at the top of Figure 5, userspace is the algorithm development environment for neural decoding and control. The userspace contains activities of a series of recalibration experiments, collections of datasets, and training of MLP models. A BMI developer has to perform these activities for further hardware development. Since AHEAD only requires the parameters of the trained MLP model and the golden datasets, users can opt for their favorite development platform and environment without limitation.

A two-stage design flow is then performed for autotuned hardware implementation. Stages are conducted in sequence as depicted in the middle and at the bottom of Figure 5. The first stage performs the automatic bit-width identification (BWID) to estimate the low-power fixed-point bit-width parameters of a given trained MLP model by exploiting the inherent error resilience of the neural network. The resultant fixed-point bit-width parameters are nonuniform, which push to the bit-width limit of error-resilience for each signal node in the network. In hardware implementation, a fixed-point model is more energy-efficient than the floating-point one because fewer bit-widths implies less logic-gate and wire connections. The second stage, as illustrated in Figure 5, is the implementation stage of AHEAD, which automatically generates the core component of the proactive BMI, i.e., a configurable MLP neural signal decoder. It facilitates microarchitecture synthesis based on the fixed-point MLP model and implements the synthesized low-power MLP core on the FPGA.

3.2. Stage 1: Automatic Bit-Width Identification

The trained MLP model provided by users is firstly reconstructed by a fixed-point hardware model, as illustrated in the middle of Figure 5, but the bit-width information is still not determined. The BWID with the closed-loop feedback path is employed in the following step for the estimation

of the bit-width parameters with the least approximation error against the results of the trained MLP model.

The reconstructed system is composed of N signal nodes, which are defined by N different fixed-point data types in terms of the bit-width. The bit-width is an integer value, and a bit-width vector denoted by B consists of a set of N bit-widths, namely, $B = \{b_1, b_2, ..., b_N\} \in I^N$. The objective function f can be modeled by the summation of the individual bit-width hardware implementation cost function c as follows:

$$f(B) = \sum_{k=1}^{N} c_k(b_k). \tag{6}$$

The quantized performance loss function E is expressed and constrained as follows:

$$E(B) = Y - \hat{Y}(B) \leq AxE_{min}, \tag{7}$$

where Y is the target output, $\hat{Y}$ is the estimated output by the reconstructed fixed-point model, and AxE_{min} is the minimum approximate error.

The lower bound bit-width is denoted by lb and the upper bound bit-width is denoted by ub. The constraints with lb and ub are also considered for each bit-width variable as follows:

$$b_{k\_lb} \leq b_k \leq b_{k\_ub}, \forall k = 1, ..., n. \tag{8}$$

Finally, the complete bit-width identification problem can then be stated as

$$\begin{aligned} \min_{B \in I^n} \quad & f(B) \\ \text{s.t.} \quad & E(k, B) \leq AxE_{min} \\ & B_{lb} \leq B \leq B_{ub}. \end{aligned} \tag{9}$$

The fixed-point bit-width parameters estimated from Stage 1 are forwarded to Stage 2 while the BWID achieves closure in terms of minimum approximation error.

3.3. Stage 2: Configurable High-Performance Low-Power MLP Microarchitecture

The generic microarchitecture of the AHEAD methodology for N-layer MLP is shown in Figure 6. Each layer aims to perform the computation of Equations (3) and (4), which is composed of two main coarse-grained processing elements (CG-PE). The first processing element is for the computation of the vector-weight matrix multiplication $X' \cdot W1'$, denoted as CG-PE-1. The buffer is cleared to zero initially, then the vector-weight matrix multiplication is performed on the multiplier and accumulator (ACC) blocks. The intermediate results are stored in buffer-1. The second processing element, denoted as CG-PE-2, includes two computations. The first computation is to accumulate the results of CG-PE1 with biased weights $B1$ and save to buffer-2, which fulfills the final computation of Equation (4). The second computation is to perform the associated activation function, as shown in Equation (3). The final outputs are stored in buffer-3 and serve as the input vector to the next layer. The computation of each layer in the MLP neural network employs the same microarchitecture as CG-PE1 and CG-PE2, including the configurable activation function. Thus, the implementation of the whole MLP neural network forms a modular and scalable structure in layers. Moreover, six signal nodes have been defined in the datapath of each layer, as shown in Figure 6. Each signal node has a distinct definition of fixed-point data type in terms of integer and fractional bit-width. For example, each weight matrix is assigned a fixed-point data type. Thus, each weight in the same weight matrix has the same bit-width.

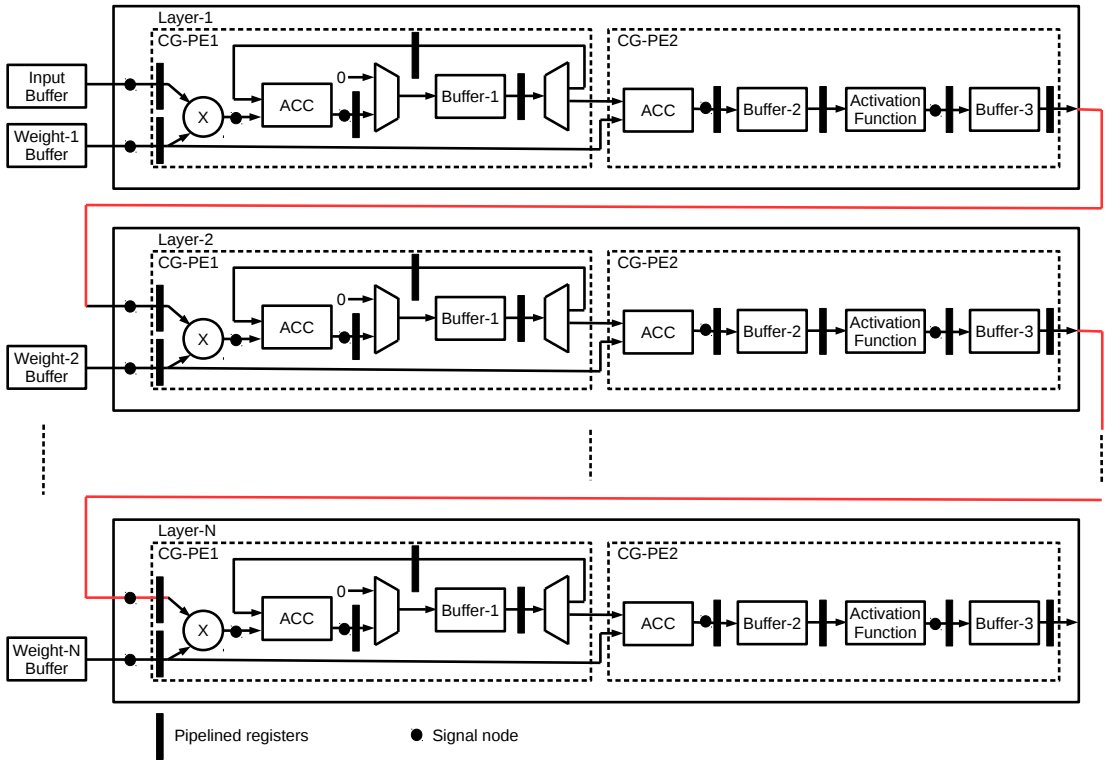
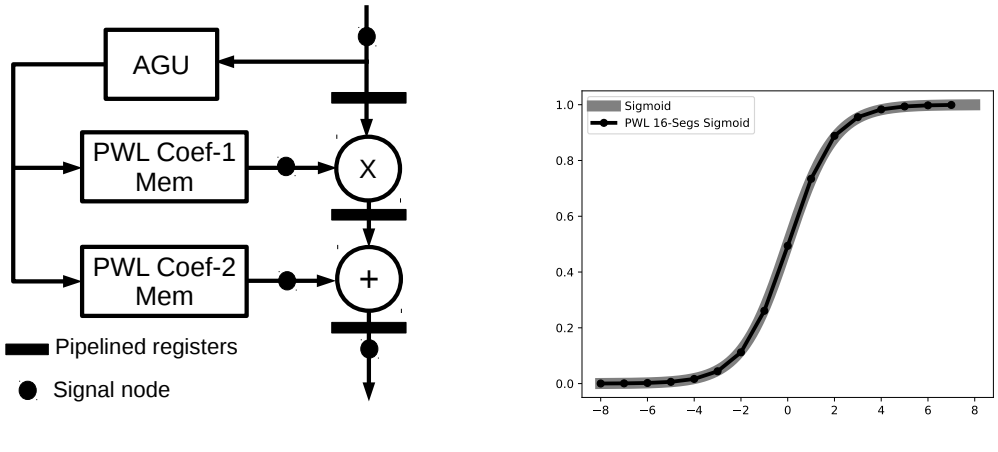


Figure 6. MLP microarchitecture.

Parallel architecture is adopted for both CG-PE1 and CG-PE2 to increase the throughput. Furthermore, the highly parallel architecture takes advantage of the higher throughput to trade-off a reduction in the voltage and clock frequency for lower power at the cost of an extra amount of area [33]. To achieve the goal, the number of multiplier and accumulation units are parameterizable, which gives run-time flexibility for configuration. In addition, the pipeline architecture is introduced to help reduce computation latency among different operation units by boosting the hardware utilization rate. The input data and weight matrix are placed at the input buffer and weight memory, respectively. They are connected to the on-chip AXI bus and can be programmed by any master on the bus. All storage elements, such as memory or buffer, are implemented by either registers or partitioned on-chip memories with configurable parameters to increase the memory bandwidth.

Approximations of sigmoid and hyperbolic tangent functions have been presented in many studies, as shown in [34–36]. In this work, the piecewise-linear (PWL) approximation method is adopted to implement the hyperbolic tangent and sigmoid activation function because of high approximation accuracy with the adjustable number of lines and simplicity of hardware implementation. The primitive first-order function is line segment $y = ax + b$, and the whole approximation is composed of a series of L-segments to represent the nonlinear activation function. The a and b are PWL coefficients for each line segment. The output of the activation function is generated by addressing the corresponding line segment for computation of the given input value x. Thus, it simplifies the microarchitecture by utilizing primitive logic elements such as adder and multiplier. The microarchitecture of the configurable PWL-approximated activation function is demonstrated in Figure 7a. All the PWL coefficients of line segments are stored in the memory, and the address generation unit (AGU) is designed to generate the associated address of PWL coefficients for computation of the activation function in terms of basic multiply and add operations. Furthermore, Figure 7b illustrates the PWL 16-segments sigmoid function, which is the approximation of the sigmoid.



(**a**) (**b**)

Figure 7. Hardware implementation of PWL approximated function for hyperbolic tangent and sigmoid. (**a**) Microarchitecture of the piecewise-linear (PWL) approximation function. AGU—address generation unit. (**b**) Output comparison of the PWL 16-segments sigmoid and original sigmoid function.

As the neuronal signals are real numbers, the mainstream solution for implementation is the use of floating-point arithmetic with a large dynamic range. However, half-precision and fixed-point arithmetics are alternatives with a lower area and power consumption at the price of lower precision, as compared to the floating-point format. Figure 8 shows the formats of IEEE-754 single-precision floating-point (FP32), IEEE 754-2008 half-precision floating-point (FP16), and fixed-point Q-format. Note that the fixed-point Q-format is a signed representation. The total bit-widths are composed of integer bit-widths (IBW) and fractional bit-widths (FBW). However, the use of the fixed-point Q-format requires the bit-width parameters for IBW and FBW to be determined, which are identified by the BWID automatically in AHEAD.

Thanks to the inherent characteristics of biological neural networks in terms of error resilience, the MLP can tolerate a fair degree of inaccuracy. Given this trade-off, the fixed-point Q-format is used for the low-power architecture design and implementation to simplify the complicated arithmetic operations for the improvement of the hardware performance metrics in terms of PPA.

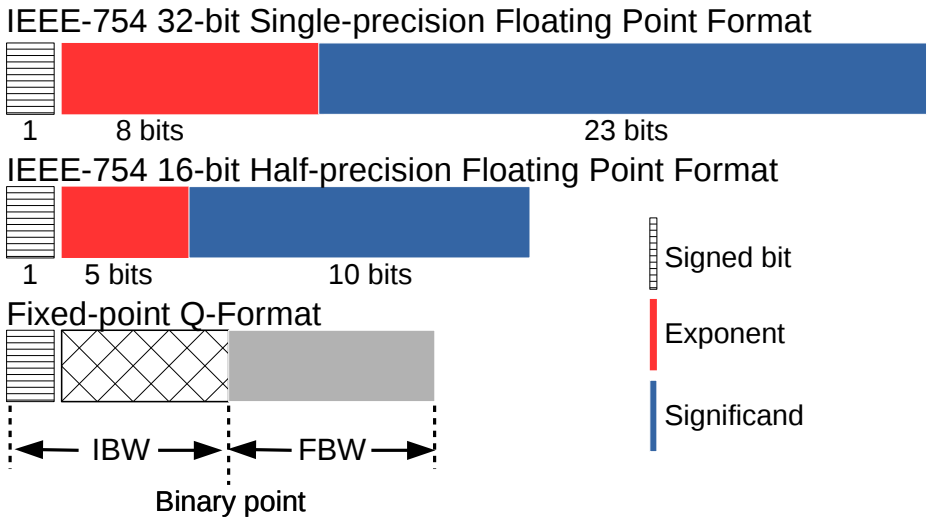


Figure 8. Comparison of number systems. IBW—integer bit-widths; FBW—fractional bit-widths.

4. Detailed Realization of the AHEAD Methodology

This section describes key implementation details of the AHEAD methodology, including three major components: MLP hardware generation, automatic test bench generation (ATBG), and bit-width identification (BWID). The complete implementation of the data and control flow with the building blocks for the AHEAD methodology is detailed in Figure 9.

Figure 9 demonstrates the implementation of the AHEAD methodology shown earlier in Figure 5. The userspace layer is the working environment of the proactive BMI developer for network model training and performance validation after dataset collection from the BMI recalibration experiments. For the generation of the energy-efficient MLP hardware accelerator, the metadata of the trained MLP parameters were required, including weight files and golden datasets. For the three major components, Stage 1 in the MLP hardware generation provides the reconstructed MLP hardware model for the BWID loop, and Stage 2 generates the energy-efficient MLP hardware accelerator. The ATBG creates the test bench with automatic bit-true simulation environments for BWID. The BWID controls the whole bit-true simulation.

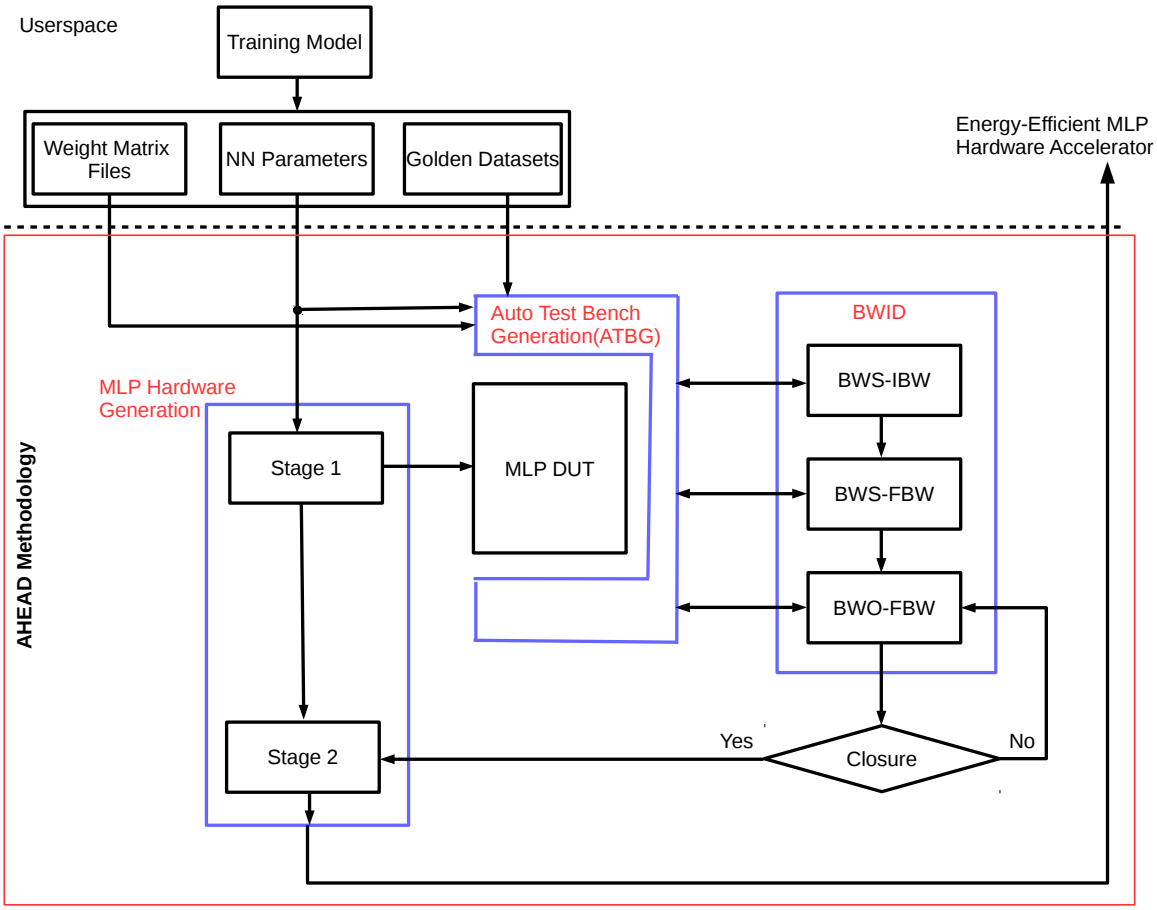


Figure 9. Implementation of the AHEAD methodology.

4.1. MLP Hardware Generation

The MLP hardware generation plays a dual role in the AHEAD methodology. In Stage 1, it generates the reconstructed fixed-point hardware model denoted as the MLP design under test (DUT), but the values of the fixed-point bit-width parameters are undetermined. The generated output is used for further identification of the fixed-point parameters from the given MLP model parameters. Hence, the requirement to reconstruct the model is only the correctness of the MLP function. In Stage 2, after the fixed-point parameters are identified, it is used by the microarchitecture synthesis to generate

the final low-power hardware accelerator. Furthermore, it performs the fully parallel and pipelined microarchitecture synthesis for all neural network layers, as illustrated in Figure 6. The final output is the fixed-point hardware design in terms of either Verilog or VHDL code that users can integrate into the top design of the proactive BMI control subsystem in the edge device.

The concept of design reuse is taken into account at the beginning of the implementation stage to maximize the benefit of the hardware implementation in minimizing the design effort with risk, while also improving design productivity. Therefore, the realization of the MLP hardware generation opts to exploit the high-level synthesis methodology over low-level HDLs, which facilitates the achievement of design reuse. It decouples the design for functionality from the design for the hardware architecture to some extent compared to traditional HDLs techniques. In HLS, the function behavior of the design specification is captured by the HLS C/C++ design file; though the specific details about underlying hardware architecture and implementation choices are described by separate compiler directives files. For example, the parallel and nonparallel versions of a given design requirement require two different RTL implementation codes. However, when it comes to HLS, the HLS C/C++ implementation code is the same and only the design constraint files are different. Thus, the MLP microarchitecture is implemented in HLS by configurable PPA options per network layer, which encapsulates the HLS compiler directives into easy-to-use commands, including parallel, pipeline, and memory partition.

4.2. Automatic Test Bench Generation (ATBG)

The purpose of the ATBG is to generate a bit-true simulation platform to iteratively compute and track the output with the golden reference value in the golden datasets, which is a vital infrastructure in the realization of the BWID loop. The block receives the following inputs. First, the MLP model parameters are utilized to configure the HLS C/C++ test bench template for the generation of the test bench with new signal nodes. Second, the dynamic range measurement functions are added to profile the dynamic range of each signal node in order to pave the road for the following IBW determination in BWID. Third, the weight matrix files and golden datasets are configured to create the stimulus generator of the reference and scoreboard-checking models, respectively. Each simulation result is recorded and maintained via the scoreboard for automatic comparison. Then, a sequencer is generated to serve as a coordinator between these verification components, as mentioned above. Finally, the MLP DUT is instantiated in the generated HLS C/C++ test bench for bit-true simulation.

4.3. Bit-Width Identification (BWID)

BWID aims to estimate the optimal IBW and FBW values subject to the minimum constraints of AxE_{min} and bit-width values. The implementation is to realize Equation (9) by initiating the fixed-point bit-true Simulation-in-the-Loop (SIL) on top of the generated test bench. It is assumed that the delivered solution is a nonuniform bit-width in some sense. A nonuniform bit-width is used to employ different bit-width values for each variable in the MLP algorithm instead of the same bit-width value. It can further reduce the area resources and power consumption of the hardware implementation.

For each signal node in the design, IBW is determined by dynamic range analysis, and FBW is explored by sensitivity-based precision analysis. The whole flow of the BWID is shown in the right of Figure 9. It consists of three steps: bit-width selection (BWS)-IBW, BWS-FBW, and bit-width optimization (BWO)-FBW, as elaborated in the following.

BWS-IBW: The bit-width selection (BWS) is carried out to determine the value of IBW for each signal node in the reconstructed MLP. The dynamic range analysis is conducted directly by running the reconstructed bit-true fixed-point simulation in which the environment has been constructed from the ATBG. The profiled IBW values are forwarded to the next step once the BWID is finished.

BWS-FBW: Sensitivity analysis is adopted for the determination of the FBW candidates for each FBW in the BWS-FBW step. It is an iterative and sweeping process. There is only one FBW variable that can be replaced with the target value under exploration, and all other variables are initialized

with the perfect IBW and FBW value in each simulation run. The perfect xBW results from allocating as large a fixed-point bit-width as possible to reduce the finite bit-width effect while dynamic range profiling, the default of which is 20 bits. The minimum FBW value with the best decoding performance against the golden datasets is the output of BWS-FBW for this FBW variable. Then, it iterates to the next candidate in the FBW variable list. For example, if an MLP has five layers and five signal nodes for each layer, there are a total of 25 FBW variables under analysis. For the sweeping experiments of all FBW variables, if it ranges from 2 bits to 20 bits with a step size of 1 bit, 25*19 = 475 simulation runs are required to perform the whole sensitivity analysis. In the end, the searched values serve as the FBW candidates of the MLP as an initial solution of fixed-point parameters. Instead, the work employs the binary search algorithm to reduce the number of simulation runs dramatically for speed-up.

BWO-FBW: The FBW values determined in the previous section are done so under the assumption that other noncontrol signal nodes are perfect FBW, which is too optimistic. Hence, the strategy of in-place bit-width optimization (BWO) for FBW is executed for the final adjustment. It utilizes these FBW values explored in the BWS-FBW step as a set of coarse-grained initial values for further minimization optimization via a stochastic local search method. In this work, the stochastic hill-climbing algorithm is used. An objective function is used to indicate the quality of the parameters under minimization optimization. The objective function is leveraged from the study [37] and modified as follows:

$$f_{obj}(X) = \alpha(1 - P) + (1 - \alpha)\frac{Avg_{FBW}}{Perf_{FBW}}, \tag{10}$$

where α is a hyperparameter, X is the list of FBW parameters, P is decoding performance, Avg_{FBW} is the average FBW, and $Perf_{FBW}$ is the perfect FBW. The term $(1 - P)$ is defined as the loss of accuracy as compared with the golden outputs in the datasets. The idea for Equation (10) is to be able to find the appropriate FBW parameters that make a trade-off balance between the loss of accuracy and minimum average FBW.

Finally, the BWID ends when it reaches the closure, and the list of final IBW and FBW values are forwarded to the microarchitecture synthesis in Stage 2 as the best fixed-point bit-width parameters for the ultimate energy-efficient MLP hardware generation.

5. Experimental Results

In this section, the effectiveness of the AHEAD methodology is validated with two cases from proactive BMI recalibration experiments that demonstrate the full range of the methodology's capabilities. The two case studies of the trained MLP decoders were generated after the BMI recalibration. Table 1 shows the MLP parameters of the two cases. These MLP parameters, along with the associated golden datasets, were used by the AHEAD methodology to produce the associated fixed-point hardware accelerator. In addition, to evaluate the quantitative effectiveness, corresponding MLP hardware accelerators with FP32 and FP16, which are based on the same MLP microarchitecture, were implemented as benchmarks, respectively.

Table 1. Parameters of two MLP retrained models.

Model Parameters	Case 1	Case 2
Number of Layer	3	4
Number of Neurons in Input Layer	800	768
Number of Neurons in Hidden Layer 1	20 (Sigmoid)	48 (Sigmoid)
Number of Neurons in Hidden Layer 2	-	20 (Sigmoid)
Number of Neurons in Output Layer	2 (Sigmoid)	2 (Sigmoid)
Total signal nodes for BWID	18	24

BWID—bit-width identification.

For the BMI recalibration, the network training was conducted on a laptop with a simulation environment created on mlpack [38], which is an open-source machine learning software library for C++, built on top of the Armadillo C++ linear algebra library [39].

The proposed AHEAD methodology was implemented in Python and shell script on a laptop. The laptop ran on Ubuntu 16.04 LTS OS, which was installed on an HP EliteBook 820 G3 machine equipped with an Intel Core i5-6200U processor (two cores running at 2.3 GHz, 3 MB cache) and 8 GB of RAM. The hardware generations were targeted at 100 MHz on Xilinx FPGAs (Xilinx Instrument, San Jose, CA) using Xilinx Vivado HLS 2019.2 for high-level synthesis [40] and Vivado 2019.2 for synthesis and physical implementation [41].

5.1. BMI Recalibration Case 1

For the parameters of Case 1, as shown in Table 1, the network structure consisted of three layers with 800 neurons in the input layer, 20 neurons in the hidden layer, and two neurons in the output layer. In addition, there are 6*3=18 signal nodes for BWID due to the the 3-layer MLP neural network, and each layer has six signal nodes. The activation function of the hidden and output layer was the sigmoid function. The target FPGA technology was xc7z020clg400-1 on the Xilinx PYNQ-Z1 development board with a 100-MHz clock constraint. Table 2 illustrates the total execution time of Case 1. The total generation time of the AHEAD methodology was less than 25 minutes, with the most time-consuming steps being the BWS-FWL and BWO-FWL.

Table 2. Execution time of AHEAD methodology for Case 1.

Stage	Steps	Execution Time [1]
BWID	BWS-IBW	00:15
	BWS-FBW	11:45
	BWO-FBW	10:10
Microarchitecture synthesis	energy-efficient hardware generation	1:20

[1] Times are in mm:ss format. BWS—bit-width selection; BWO—bit-width optimization.

As shown in Table 3, the evaluation metrics include accuracy, performance, power, and area (APPA). For the accuracy, the delivered design (results in nonuniform bit-widths with the average bit-width of 7.47 bits) achieved a 0% loss of accuracy. The loss of accuracy is compared to the results of the golden datasets provided by floating-point MLP model training.

Table 3. Accuracy, performance, power, and area (APPA) comparison of Case 1.

Metrics	Type	FP32	FP16	Fixed-Point
Accuracy	Loss of Accuracy	0%	0%	0% [1]
Performance	Max Frequency	103.7 MHz	105.6 MHz	106.2 MHz
	Max Throughput	30K	31K	125K
	Max Latency	32.86 us	32.28 us	8.01 us
Power	Dynamic Power	408 mW	246 mW	88 mW
Area	Slice LUTs (Utilization %)	13,702 (25.76%)	6955 (13.07%)	2759 (5.19%)
	Slice Registers (Utilization %)	15,543 (14.61%)	9059 (8.51%)	1610 (1.51%)
	DSP48E1s (Utilization %)	103 (46.82%)	82 (37.27%)	21 (9.55%)
	BRAMs (Utilization %)	42 (30%)	21.5 (15.36%)	8.5 (6.07%)

[1] The resultant average bit-width is 7.47 bits. LUTs—look-up tables; BRAMs—block rams.

For performance benchmarking, the maximum frequency of the generated design was 106.2 MHz, which is slightly higher than the FP32 and FP16 implementation. However, the latency archived 8.01 us and delivered up to approximately 4X the speed in the inference engine execution in comparison with

the FP32 and FP16 implementations. The throughput, which is defined as decoding times per second, demonstrates the same trend. Furthermore, the power consumption was reduced by approximately 4.63X and 2.3X with respect to FP32 and FP16 implementations, respectively. The proposed work achieves more significant results in performance and power than the other two benchmarks because of the reduction of the bit-widths identified automatically by the BWID loop.

In the area usage comparison, the generated design also had the highest area efficiency among these cases. The breakdown of the area consumption is as follows: 5.19% in slice LUTs, 1.51% in slice registers, 9.55% in DSP48E1s, and 6.07% in on-chip block rams (BRAMs) in terms of utilization on the target FPGA. Our design exhibits the following reduction in area utilization on FPGA: 9.29% in BRAMs, 27.72% in DSP48E1s, 7% in slice registers, and 7.88% in slice LUTs, with the same loss of accuracy compared to the FP16 implementation due to the lower bit representation. It is worth noting that FP32 and FP16 implementations consume a significant number of DSP48E1s due to the use of floating-point addition and multiplication arithmetic operations. Furthermore, the low hardware resource utilization on FPGA implies that the proactive BMI edge device could accommodate a lot more hardware functions, further raising its value.

5.2. BMI Recalibration Case 2

In order to evaluate the capability of scalability in terms of different network topologies and the growth of neural networks sizes, the four-layer MLP network was used in Case 2, with 768 neurons in the input layer, 48 neurons in hidden layer 1, 20 neurons in hidden layer 2, and 2 neurons in the output layer, as indicated in Table 1. Apart from that, the MLP topology results in 6*4 = 24 signal nodes for BWID. It targeted Xilinx xc7z030sbg485-1 FPGA on the PicoZed development board with the same clock constraint since the FP32 implementation of Case 2 does not fit on the xc7z020clg400-1.

As can be seen in Table 4, the total hardware generation time took less than 35 minutes, with the BWO-FWS occupying approximately 60% of the total execution time. The increase in execution time is due to Case 2 having much more signal nodes for BWID, which increases the bit-width exploration space.

Table 4. Execution time of AHEAD methodology for Case 2.

Stage	Steps	Execution Time [1]
BWID	BWS-IBW	00:17
	BWS-FBW	09:57
	BWO-FBW	20:40
Microarchitecture synthesis	energy-efficient hardware generation	1:23

[1] Times are in mm:ss format.

Table 5 presents the comparison of the APPA results of the generated implementation against the FP32 and FP16 implementations under the same MLP microarchitecture. The resultant fixed-point implementation had the average bit-width of 6.95 bits without loss of accuracy.

The latency and throughput of the resultant fixed-point implementation were about 4X faster than the FP32 and FP16 implementations when compared with the performance. Besides, regarding the total energy consumption, the resultant fixed-point design was approximately 5.97X more energy-efficient than the FP32 implementation and used 2.73X less power than the FP16 implementation. This reveals that the proactive BMI edge device can have a faster decoding time with lower power consumption, which is crucial for portable proactive BMI edge devices.

Finally, the experimental results show that the resultant fixed-point implementation had approximately a 9.86% reduction in slice LUTs, 10.68% less slice registers, a 48.25% reduction in DSP48E1s, and 14.72% less BRAMs compared to the FP16 implementation. It is interesting to note that the resultant fixed-point implementation used 15.33% in slice LUTs and 0.25% in DSP48E1s due to the

resultant average bit-width being lower and causing the Xilinx synthesis engine to adjust the synthesis strategy to use slice LUTs for logic function synthesis.

Table 5. APPA comparison of Case 2.

Metrics	Type	FP32	FP16	Fixed-Point
Accuracy	Loss of Accuracy	0%	0%	0% [1]
Performance	Max Frequency	104.5 MHz	108.8 MHz	114.6 MHz
	Max Throughput	29K	30K	124K
	Max Latency	34.69 us	33.36 us	8.05 us
Power	Dynamic Power	1,319 mW	604 mW	221 mW
Area	Slice LUTs (Utilization %)	37,375 (47.55%)	19,801 (25.19%)	12,050 (15.33%)
	Slice Registers (Utilization %)	41,118 (26.16%)	22,134 (14.08%)	5347 (3.4%)
	DSP48E1s (Utilization %)	243 (60.75%)	194 (48.5%)	1 (0.25%)
	BRAMs (Utilization %)	119 (44.91%)	60 (22.64%)	21 (7.92%)

[1] The resultant average bit-width is 6.95 bits.

6. Discussion

The hurdles to the development of proactive BMI control edge devices on FPGA are achieving low power consumption and meeting the reconfigurable requirement of the MLP hardware configuration due to the need for portable edge devices and BMI recalibration, respectively. As indicated in the literature [29], it takes much effort and resources from the hardware team to modify or even redesign the low-power MLP hardware accelerator because of the induced specification change. Moreover, the analysis of fixed-point bit-widths is a tedious and labor-intensive task [24,27] which must be executed by either the software team or hardware team. Instead, the AHEAD methodology tackles the issue by automating the complete fixed-point hardware analysis to digital design flow in order to reduce the development efforts and time of redesign, reverification, and reimplementation. Thus, it can have a rapid hardware update for the BMI edge devices on FPGA due to BMI recalibration.

This work aims to address the aforementioned design gap to create an autonomous design methodology that analyzes the problems from a holistic view, including the recalibration needs of proactive BMI experiments, the run change after network retraining, the low-power design, and the hardware design flow concurrently. The experimental results in Section 5 indicate that high-performance, low-power, fixed-point hardware accelerators can be generated automatically. Moreover, the resultant fixed-point hardware consumes fewer area resources and less power while retaining comparable results in terms of decoding performance, as compared with golden datasets. This is achieved by taking advantage of the synergy between the BWID loop and design reuse in an autonomous way. The design reuse is realized by configurable MLP hardware generation. In addition, the configurable HLS template-based hardware accelerator serves as a platform for not only the BWID loop but also low-power hardware generation. Thus, the significant advantages of the work include boosting the design productivity and facilitating the generation process of the low-cost and low-power hardware design for proactive BMI control edge devices on FPGA.

The energy-aware hardware generation was devised using the holistic cross-layer low-power design methodology, which spans from the architecture to microarchitecture level. The use of the fixed-point arithmetic also improves performance, as demonstrated. From the perspective of the bit-width selection, the quantization of large-scale neural networks has been intensively studied [42–44]. However, previous works normally quantize all the layers uniformly. Moreover, prior methods require domain knowledge of both machine learning and hardware architecture to explore where to retain more bits to extract the low-level features in a specific layer. Compared with prior works, the proposed AHEAD methodology employs the characteristic of a neural network in which different layers have different redundancy to results in nonuniform bit-widths for different layers in terms of mixed precision. Additionally, the BWID method, which is inspired by system identification, facilitates

to automatically reconstruct the fixed-point hardware model to explore the appropriate bit-widths configurations from given metadata in terms of MLP parameters instead of providing any C/C++ implementation code.

From the viewpoint of the software and FPGA hardware developers, our methodology acts as an autonomous agent and frees up the resources in fixed-point precision analysis and low-power hardware design. Furthermore, the proposed methodology does not pose any restriction in the choice of neural network training software tools or programming language.

Future research will be dedicated to the inclusion of automatically efficient piecewise linear approximation of an arbitrary nonlinear activation function, including the number of linear segments and associated fixed-point coefficients in the AHEAD framework. Then, future work can extend to the energy-efficient hardware generation of radial basis function (RBF) neural network and echo state network (ESN) in AHEAD. In RBF, each neuron in the hidden layer employs different Gaussian activation functions. RBF and ESN are vital methods in temporal nonlinear neural signal processing, such as in biorobotics and biomedical engineering. Finally, to fully extend the AHEAD methodology to other applications, the standard format of metadata could be designed to support different neural networks.

Author Contributions: Research and development, N.-S.H. and Y.-C.C.; experiments, N.-S.H.; writing—original draft preparation, N.-S.H.; writing—review and editing, Y.-C.C., J.C.L., and P.M.; supervision, J.C.L.; project work package PI, P.M.; funding acquisition, P.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Horizon 2020 Framework Programme (FETPROACT-01-2016FET Proactive: emerging themes and communities) grant number 732266 (Plan4Act).

Acknowledgments: We thank Jan-Matthias Braun, and Ricardo Rodrigues do Carmo for supporting the work in the dataset generation of neuronal signals for BMI proactive control experiments. We also thank Alexander Gail and his team from the Sensorimotor Group of Deutsches Primatenzentrum GmbH (DPZ) for providing neuronal signals and the image of Spike Neuronal Data Activity in Figure 2 and discussion on the Proactive BMI control scenario in Figure 1, Sergio Guillen Barrionuevo and his team from MYSPHERA S.L. for providing the image of Neural Decoding & Control in Figure 2, Florentin Wörgötter, Christian Tetzlaff and their team for their fruitful discussions of the Plan4Act control scenario and the Plan4Act system architecture, and María Teresa Arredondo Waldmeyer and her team from Life Supporting Technologies (LifeSTech) at Technical University of Madrid (UPM) for technical discussion on the FPGA interface to smart home devices. Finally, we thank the Plan4Act project's team members for the overall project concept development and general technical discussion.

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

References

1. Hirata, M. Brain machine-interfaces for motor and communication control. In *Cognitive Neuroscience Robotics B*; Springer: Berlin, Germany, 2016; pp. 227–251.
2. Bablani, A.; Edla, D.R.; Tripathi, D.; Cheruku, R. Survey on Brain-Computer Interface: An Emerging Computational Intelligence Paradigm. *ACM Comput. Surv. (CSUR)* **2019**, *52*, 1–32. [CrossRef]
3. Miralles, F.; Vargiu, E.; Dauwalder, S.; Solà, M.; Müller-Putz, G.; Wriessnegger, S.C.; Pinegger, A.; Kübler, A.; Halder, S.; Käthner, I.; et al. Brain Computer Interface on Track to Home. Available online: https://www.hindawi.com/journals/tswj/2015/623896/ (accessed on 5 March 2020).
4. MindSee Project. 2020. Available online: http://mindsee.eu/ (accessed on 5 March 2020).
5. Micera, S.; Carpaneto, J.; Raspopovic, S.; Granata, G.; Mazzoni, A.; Oddo, C.M.; Cipriani, C.; Stieglitz, T.; Mueller, M.; Navarro, X.; et al. Toward the Development of a Neuro-Controlled Bidirectional Hand Prosthesis. In *International Workshop on Symbiotic Interaction*; Springer: Berlin, Germany, 2015; pp. 105–110.
6. SI-CODE Project. 2020. Available online: https://www.sicode.eu/ (accessed on 5 March 2020).
7. Weston, P. Battle for Control of Your Brain: Microsoft Takes on Facebook with Plans for a Mind-Reading HEADBAND That Will Let You Use Devices with the Power of Thought. 2018. Available online: http://www.dailymail.co.uk/sciencetech/article-5274823/Microsoft-takes-Facebook-mind-reading-technology.html (accessed on 5 March 2020).

8. Brown, K.V. Here Are the First Hints of How Facebook Plans to Read Your Thoughts. 2018. Available online: https://gizmodo.com/here-are-the-first-hints-of-how-facebook-plans-to-read-1818624773 (accessed on 5 March 2020).
9. Musk, E. An integrated brain-machine interface platform with thousands of channels. *J. Med. Internet Res.* **2019**, *21*, e16194. [CrossRef]
10. Plan4Act Project. 2017. Available online: http://plan4act-project.eu/index.php/about/ (accessed on 5 March 2020).
11. Berger, M.; Gail, A. The Reach Cage environment for wireless neural recordings during structured goal-directed behavior of unrestrained monkeys. *bioRxiv* **2018**, 305334. [CrossRef]
12. Gallego, J.A.; Perich, M.G.; Naufel, S.N.; Ethier, C.; Solla, S.A.; Miller, L.E. Cortical population activity within a preserved neural manifold underlies multiple motor behaviors. *Nat. Commun.* **2018**, *9*, 1–13. [CrossRef] [PubMed]
13. Gallego, J.A.; Perich, M.G.; Chowdhury, R.H.; Solla, S.A.; Miller, L.E. A stable, long-term cortical signature underlying consistent behavior. *BioRxiv* **2018**, 447441. . [CrossRef]
14. Che, S.; Li, J.; Sheaffer, J.W.; Skadron, K.; Lach, J. Accelerating compute-intensive applications with GPUs and FPGAs. In Proceedings of the IEEE Symposium on Application Specific Processors, Anaheim, CA, USA, 8–9 June 2008; pp. 101–107.
15. Nurvitadhi, E.; Venkatesh, G.; Sim, J.; Marr, D.; Huang, R.; Ong Gee Hock, J.; Liew, Y.T.; Srivatsan, K.; Moss, D.; Subhaschandra, S.; et al. Can FPGAs beat GPUs in accelerating next-generation deep neural networks? In Proceedings of the ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA), Monterey, CA, USA, 22–24 February 2017; pp. 5–14.
16. Wang, D.; Hao, Y.; Zhu, X.; Zhao, T.; Wang, Y.; Chen, Y.; Chen, W.; Zheng, X. FPGA implementation of hardware processing modules as coprocessors in brain-machine interfaces. In Proceedings of the 2011 IEEE Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Boston, MA, USA, 30 August–3 September 2011; pp. 4613–4616.
17. Savich, A.W.; Moussa, M.; Areibi, S. The impact of arithmetic representation on implementing MLP-BP on FPGAs: A study. *IEEE Trans. Neural Netw.* **2007**, *18*, 240–252. [CrossRef]
18. Lippmann, R. An introduction to computing with neural nets. *IEEE Assp Mag.* **1987**, *4*, 4–22. [CrossRef]
19. Schuman, C.D.; Potok, T.E.; Patton, R.M.; Birdwell, J.D.; Dean, M.E.; Rose, G.S.; Plank, J.S. A survey of neuromorphic computing and neural networks in hardware. *arXiv* **2017**, arXiv:1705.06963.
20. Misra, J.; Saha, I. Artificial neural networks in hardware: A survey of two decades of progress. *Elsevier Neurocomput.* **2010**, *74*, 239–255. [CrossRef]
21. Chippa, V.K.; Mohapatra, D.; Raghunathan, A.; Roy, K.; Chakradhar, S.T. Scalable effort hardware design: Exploiting algorithmic resilience for energy efficiency. In Proceedings of the IEEE Design Automation Conference, Anaheim, CA, USA, 13–18 June 2010; pp. 555–560.
22. Nikolić, Z.; Nguyen, H.T.; Frantz, G. Design and implementation of numerical linear algebra algorithms on fixed point DSPs. *EURASIP J. Adv. Signal Process.* **2007**, *2007*, 087046. [CrossRef]
23. Xilinx. Reduce Power and Cost by Converting from Floating Point to Fixed Point. 2017. Available online: https://www.xilinx.com/support/documentation/white_papers/wp491-floating-to-fixed-point.pdf (accessed on 5 March 2020).
24. Fixed-Point Refinement of Digital Signal Processing Systems. 2019. Available online: https://hal.inria.fr/hal-01941898/file/FixedPointRefinement.pdf (accessed on 5 March 2020).
25. Sung, W.; Kum, K.I. Simulation-based word-length optimization method for fixed-point digital signal processing systems. *IEEE Trans. Signal Process.* **1995**, *43*, 3087–3090. [CrossRef]
26. Cantin, M.A.; Savaria, Y.; Lavoie, P. A comparison of automatic word length optimization procedures. In Proceedings of the IEEE International Symposium on Circuits and Systems, Proceedings (Cat. No. 02CH37353), Phoenix-Scottsdale, AZ, USA, 26–29 May 2002; Volume 2, p. II.
27. Roy, S.; Banerjee, P. An algorithm for trading off quantization error with hardware resources for MATLAB-based FPGA design. *IEEE Trans. Comput.* **2005**, *54*, 886–896. [CrossRef]
28. Han, K. Automating tRansformations from Floating-Point to Fixed-Point for Implementing Digital Signal Processing Algorithms. Ph.D. Thesis, The University of Texas at Austin, Austin, TX, USA, 2006.

29. Cong, J.; Liu, B.; Neuendorffer, S.; Noguera, J.; Vissers, K.; Zhang, Z. High-level synthesis for FPGAs: From prototyping to deployment. *IEEE Trans. Comput.-Aided Des. Integrated Circ. Syst.* **2011**, *30*, 473–491. [CrossRef]
30. Heelan, C.; Komar, J.; Vargas-Irwin, C.E.; Simeral, J.D.; Nurmikko, A.V. A mobile embedded platform for high performance neural signal computation and communication. In Proceedings of the 2015 IEEE Biomedical Circuits and Systems Conference (BioCAS), Atlanta , GA, USA, 22–24 October 2015; pp. 1–4.
31. Ohbayashi, M.; Picard, N.; Strick, P.L. Inactivation of the dorsal premotor area disrupts internally generated, but not visually guided, sequential movements. *J. Neurosci.* **2016**, *36*, 1971–1976. [CrossRef] [PubMed]
32. Tanji, J. Sequential organization of multiple movements: involvement of cortical motor areas. *Annu. Rev. Neurosci.* **2001**, *24*, 631–651. [CrossRef] [PubMed]
33. Parhi, K.K. *VLSI Digital Signal Processing Systems: Design and Implementation*; John Wiley & Sons: Hoboken, NJ, USA, 2007.
34. Basterretxea, K.; Tarela, J.M.; Del Campo, I. Approximation of sigmoid function and the derivative for hardware implementation of artificial neurons. *IEE Proc.-Circ. Dev. Syst.* **2004**, *151*, 18–24. [CrossRef]
35. Armato, A.; Fanucci, L.; Pioggia, G.; De Rossi, D. Low-error approximation of artificial neuron sigmoid function and its derivative. *Electron. Lett.* **2009**, *45*, 1082–1084. [CrossRef]
36. Gomar, S.; Mirhassani, M.; Ahmadi, M. Precise digital implementations of hyperbolic tanh and sigmoid function. In Proceedings of the 2016 IEEE 50th Asilomar Conference on Signals, Systems and Computes, Pacific Grove, CA, USA, 6–9 November 2016; pp. 1586–1589.
37. Vieira, S.M.; Mendonça, L.F.; Farinha, G.J.; Sousa, J.M. Modified binary PSO for feature selection using SVM applied to mortality prediction of septic patients. *Appl. Soft Comput.* **2013**, *13*, 3494–3504. [CrossRef]
38. Curtin, R.; Edel, M.; Lozhnikov, M.; Mentekidis, Y.; Ghaisas, S.; Zhang, S. mlpack 3: A fast, flexible machine learning library. *J. Open Source Softw.* **2018**, *3*, 726. [CrossRef]
39. Sanderson, C.; Curtin, R. Armadillo: a template-based C++ library for linear algebra. *J. Open Source Softw.* **2016**, *1*, 26. [CrossRef]
40. Xilinx. Vivado Design Suite User Guide: High-Level Synthesis (UG902). 2020. Available online: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2019_2/ug902-vivado-high-level-synthesis.pdf (accessed on 5 March 2020).
41. Xilinx. Vivado 2019.2—Design Flows Overview. 2019. Available online: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2019_2/ug892-vivado-design-flows-overview.pdf (accessed on 5 March 2020).
42. Choi, J.; Wang, Z.; Venkataramani, S.; Chuang, P.I.J.; Srinivasan, V.; Gopalakrishnan, K. Pact: Parameterized clipping activation for quantized neural networks. *arXiv* **2018**, arXiv:1805.06085.
43. Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A.; Adam, H.; Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 2704–2713.
44. Han, S.; Mao, H.; Dally, W.J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv* **2015**, arXiv:1510.00149.