

SNN eXpress: Streamlining Low-Power AI-SoC Development With Unsigned Weight Accumulation Spiking Neural Network

Hyeonguk Jang¹ | Kyuseung Han¹  | Kwang-Il Oh¹  | Sukho Lee¹ |
Jae-Jin Lee¹ | Woojoo Lee² 

¹AI Edge SoC Research Section,
Electronics and Telecommunications
Research Institute, Daejeon, Republic of
Korea

²School of Electrical and Electronics
Engineering, Chung-Ang University,
Seoul, Republic of Korea

Correspondence

Woojoo Lee, School of Electrical and
Electronics Engineering, Chung-Ang
University, Seoul, Republic of Korea.
Email: space@cau.ac.kr

Funding information

This work was supported by the
Electronics and Telecommunications
Research Institute (ETRI) grant funded by
the Korea government (24ZS1230,
memory-computation convergence
neuromorphic computing technology).

Abstract

SoCs with analog-circuit-based unsigned weight-accumulating spiking neural networks (UWA-SNNs) are a highly promising solution for achieving low-power AI-SoCs. This paper addresses the challenges that must be overcome to realize the potential of UWA-SNNs in low-power AI-SoCs: (i) the absence of UWA-SNN learning methods and the lack of an environment for developing applications based on trained SNN models and (ii) the inherent issue of testing and validating applications on the system being nearly impractical until the final chip is fabricated owing to the mixed-signal circuit implementation of UWA-SNN-based SoCs. This paper argues that, by integrating the proposed solutions, the development of an EDA tool that enables the easy and rapid development of UWA-SNN-based SoCs is feasible, and demonstrates this through the development of the SNN eXpress (SNX) tool. The developed SNX automates the generation of RTL code, FPGA prototypes, and a software development kit tailored for UWA-SNN-based application development. Comprehensive details of SNX development and the performance evaluation and verification results of two AI-SoCs developed using SNX are also presented.

KEYWORDS

low-power AI-SoCs, SNN eXpress, spiking neural networks, UWA-SNN

1 | INTRODUCTION

Various methods for implementing a spiking neural network (SNN) in hardware and incorporating it into a system-on-chip (SoC) to create low-power AI-SoCs have been actively studied, including digital [1, 2], mixed-signal [3], and purely analog circuit-based approaches [4, 5]. Among these, analog SNNs have superior low-power characteristics, making them suitable for

low-power AI-SoCs. This is because analog SNNs perform computations asynchronously only when neurons generate spikes, resulting in reduced dynamic power consumption. In addition, the frequent sampling and conversion steps required in digital and mixed-signal SNNs are unnecessary in analog SNNs, allowing for more energy-efficient signal processing.

However, significant challenges remain to be overcome in developing SoCs equipped with analog SNNs:

(i) establishing an environment for training the analog SNN and developing applications using the trained SNN model are extremely challenging and (ii) an AI-SoC is a mixed-signal SoC that consists of various digital-logic-based IPs and analog SNN hardware, making it nearly impossible to test and validate applications on the system until the SoC is ultimately fabricated. The first challenge arises from the fact that constructing customized training and development environments suitable for various analog SNNs often requires considerable time and effort. For instance, with the most recent SNNs best suited for low-power AI-SoCs, such as an analog SNN that only accumulates unsigned weight values (unsigned weight-accumulating SNN or UWA-SNN), neither a training environment for the SNN model nor a training method has been established, making the development of low-power AI-SoCs using this technology extremely difficult. The second challenge originates from the fact that FPGA prototyping, typically used to quickly simulate applications before SoC development in the SoC development flow, is unfeasible for SoCs that include analog circuits. In other words, there is no practical method to validate SNN applications in hardware prior to chip fabrication.

To address the challenges of low-power AI-SoC development, we propose an AI-SoC architecture, as illustrated in Figure 1, and a corresponding electronic design automation (EDA) solution called *SNNexpress* (SNX). SNX provides SoC developers with a list of IP components that constitute the SoC, including the SNN hardware (i.e., the SNN accelerator and spike generator), CPU core, memory, and peripheral IP. Once developers select these components to compose the AI-SoC architecture, SNX automatically generates the RTL code, FPGA prototype, and software development kit (SDK) for the given architecture.

To develop the SNX, we first selected the UWA-SNN, which is most suitable for low-power AI-SoCs. As no learning method has been proposed for UWA-SNNs to date, we developed a method for this purpose.

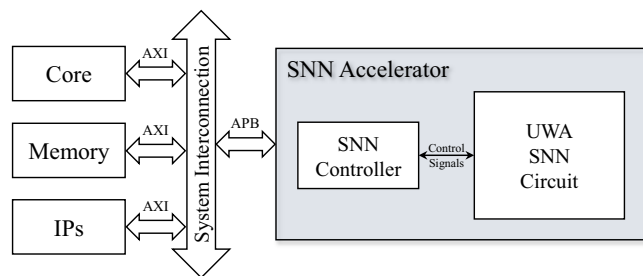


FIGURE 1 Architecture of a low-power AI-SoC with an integrated UWA-SNN accelerator.

Accordingly, we developed SNX to automatically build an environment in which developers can train UWA-SNN models and develop applications using the trained SNN model. In addition, to overcome the inability to perform FPGA prototyping owing to the mixed-signal SoC, we developed an SNN accelerator composed of digital logic that functions equivalently to the analog-circuit-based UWA-SNN and included this IP in the list of IPs provided by SNX. Consequently, even before the mixed-signal SoC is fabricated, developers can validate and test the developed applications on the FPGA prototype of the target SoC.

In this study, we provide detailed information on the proposed AI-SoC architecture with UWA-SNN and SNX solutions. We present a comprehensive description of the experiments conducted to validate and evaluate the proposed technology.

2 | UWA-SNN

Various circuit design techniques have been reported for implementing analog SNNs, including the use of capacitors [5], memristors [4], and digital-to-analog converters [3]. Among these, the use of capacitors is currently considered the most practical method for implementing SNNs owing to their energy efficiency, compatibility with standard CMOS processes, and robustness against process variations.

The analog SNN circuit employing capacitors can be classified into three types based on their weight representations. The first type employs separate synaptic memories for positive and negative weight values [7]. Depending on the input presynaptic spike, the spikes in positive and negative neurons are triggered by positive and negative synaptic memories, respectively. The final emitted spike count was determined by the difference between the spikes from positive and negative neurons. This type of analog SNN requires separate synaptic memories and neurons for the positive and negative weight values, respectively. This inevitably results in an increased circuit area. Moreover, because of the need for additional computations to determine the difference between positive and negative neuron spikes, there can be an accompanying increase in latency and power consumption.

The second type utilizes the most significant bit of synapse memory as a sign bit to store signed weight values [8, 9]. This approach only requires the addition of one sign bit, circumventing the problem of increased area. However, a potential issue arises if a mismatch exists between the magnitudes of the implemented positive and negative current sources in the synapse, leading

to computational errors. Another challenge is the discontinuity in computation before and after a spike is emitted. Specifically, if the threshold voltage of a neuron is exceeded and a spike is fired, the membrane potential is reset to zero. Consequently, any subsequent negative post-synaptic charges do not affect the membrane potential formed before the neuron fires.

Finally, the UWA-SNN streamlines only the amassed unsigned weight values. This approach can lead to a more compact circuit design, fewer additional computations, reduced latency, and lower power consumption. Hence, a UWA-SNN is potentially suitable for low-power AI-SoCs.

To validate and evaluate the UWA-SNN, we designed an actual UWA-SNN circuit and fabricated it on a chip, as illustrated in Figure 2a,b based on our previous work [6]. The circuit has a crossbar structure and utilizes an address-event representation (AER) interface. This increases the membrane potential by accumulating charge in a connected capacitor, and the neuron fires when this potential exceeds its threshold. The synapse of the circuit has a MOSFET switch that is activated upon a spike input and a current source controlled by the synapse memory value symbolizing the weight. Thus, with an input spike, the switch is activated, supplying a charge equal to the weight of the synaptic memory to the neuronal membrane capacitor. The connected output neuron fires a spike when the accumulated charge in the capacitor surpasses a threshold.

Through experiments with the developed UWA-SNN circuit, we obtained conclusive results that the UWA-SNN excels in terms of low-power consumption, as expected. Hence, in this study, we developed SNX to facilitate the automatic design of low-power AI-SoCs that incorporate UWA-SNNs.

3 | TRAINING METHODOLOGY FOR UWA-SNN

Several widely known training algorithms for SNNs include spike-timing-dependent plasticity, which adjusts the synaptic connection strength between pre- and post-synaptic neurons based on the relative timing of spikes [10–12], and deep neural network (DNN)-to-SNN conversion methods, which transform pretrained DNN weights into connection delays and strengths between neurons in SNNs [13, 14]. Although the latter approach is favored as a more practical method for high-performance, efficient multilayer SNNs, a DNN-to-SNN conversion method specifically for UWA-SNNs is yet to be explored. Existing DNN-to-SNN conversion methods support data types, such as 8-bit or 16-bit signed integers (int8 or int16, respectively), yielding quantized weights as signed integers, whereas the UWA-SNN only supports the unsigned 8-bit unsigned integer (uint8) data type for weights. Therefore, in this study, we developed a DNN-to-SNN conversion-based training method for UWA-SNNs.

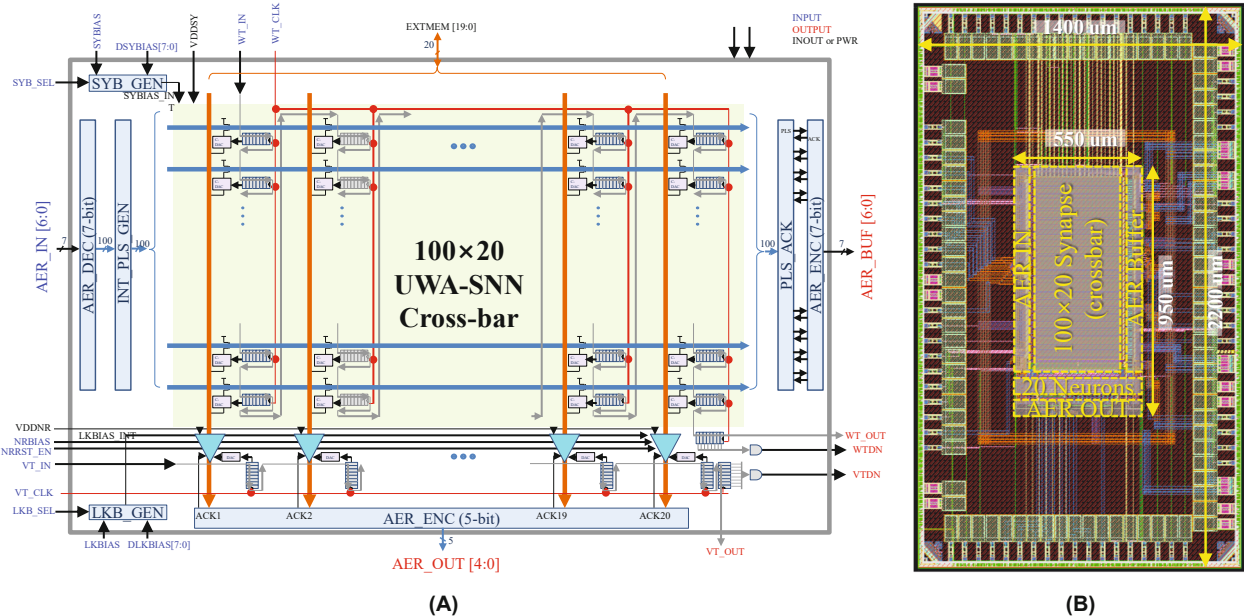


FIGURE 2 Block diagram and top layout of a UWA-SNN core: (a) block diagram and (b) top layout. Reproduced from K.-I. Oh et al., *Design of 0.9V-Operating Low-power AI SNN Analog Core* (Autumn Annual Conference of IEIE, Gwangju, Republic of Korea), 2020 [6].

We initially considered the characteristics of the hardware for SNN inference and the features of both the DNN and SNN, setting various hyperparameters, such as neuron models, activation functions, neuron coding methods, data types, and network structures, to minimize accuracy degradation owing to the conversion. More specifically, we made the following decisions:

1. For the neuron model of both the SNN and DNN, we selected a neuron model that does not consider bias for the DNN, as the UWA-SNN circuit cannot compute bias.
The integrated-and-fire (I&F) neuron model was chosen for the SNN, as it is similar to the DNN neuron model and has no need for leakage compensation.
2. Regarding activation functions, we used a spiking function in the SNN, which needs to generate continuous values into output spikes, whereas we refrained from using an activation function in the DNN owing to compatibility issues.
3. We used rate coding [15] as the neuron coding method for the SNN because it can represent the intensity as the frequency (rate) of input spikes, similar to how each node in the DNN encodes continuous intensity information.
4. Concerning data types, we followed the data types of weight values and thresholds supported by the UWA-SNN circuit, setting both the DNN and SNN weight data types as uint8, and the SNN threshold data type as uint16.
5. As the UWA-SNN circuit can perform SNN operations for fully-connected (FC) layers because of its crossbar structure, we only considered the network structure of the FC layer.

Subsequently, we focused on the deep learning and conversion parts of the DNN-to-SNN process, as illustrated in Figure 3. The key point in deep learning is that the DNN model must learn the weights as uint8. However, while current deep-learning frameworks, such as PyTorch and TensorFlow, support uint8 as a data type for inference inputs and outputs, they do not support quantizing weights to uint8. In the case of MobileNet, the weights can be quantized to uint8, whereas the biases can be quantized to int32. As the neuron model targeted

by the UWA-SNN core does not use biases, this type of quantization is unsuitable for the DNN-to-SNN conversion required by the UWA-SNN.

To address this issue, we devised a new technique for quantizing weights to uint8 during deep learning. This involves clipping weights smaller than zero during training to ensure that they are learned as unsigned values, followed by applying quantization-aware training (QAT). There are two main quantization techniques: post-training quantization [16, 17] and QAT [18, 19]. The primary distinction between the two is the timing of their quantization process. Post-training quantization is performed after the model has been trained, whereas QAT integrates quantization during the training phase. We adopted QAT because of its inherent advantages. In the case of post-training quantization, there was an initial decline in accuracy owing to clipping during training, followed by further accuracy degradation originating from the quantization process. Conversely, with QAT, both clipping and quantization occurred during training. This indicates that the effects of clipping and quantization can be simulated in real-time, allowing the DNN model to learn while considering these factors. Therefore, whereas post-training quantization induces accuracy drops at two separate stages, QAT results in a single consolidated accuracy decline.

In the conversion part, we deliberated on how to determine the weights, input firing rate, and threshold of the SNN model to maximize its accuracy from the DNN model. We concluded that, while the weights of the SNN model should directly use the uint8 weights from the DNN model without any transformation, we needed to devise a method to determine the optimal input firing rate and threshold. We accomplished this by implementing a grid search and measuring the accuracy of various combinations of thresholds and input firing rates. The accuracy was calculated for all nodes using the same threshold and was obtained by counting all the results from the test dataset.

To elucidate the method for determining the accuracy based on the threshold and input firing rate values, the accuracy was calculated by counting the true and false inferences across all the test datasets. The procedure for executing the inference and counting the results for the single-input data is as follows: First, the input data were transformed into input spikes based on the rate coding method, which is contingent on the input firing rate. Subsequently, using these spikes as inputs, an SNN inference was performed, and the number of spikes generated at each node in the output layer was counted. Once the SNN inference is completed for all the created input spikes, the node that generates the highest number of spikes is deemed the inference outcome. Finally, this

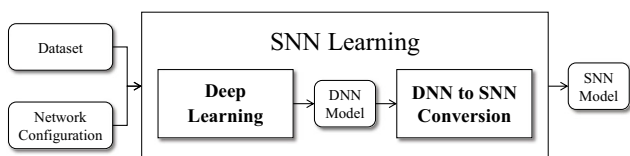


FIGURE 3 Proposed training process for the UWA-SNN.

inference result is compared with the true label and counted as either correct or incorrect based on the comparison. Thus, we propose a DNN-to-SNN conversion method that directly utilizes the weight values of the DNN model and employs a grid search to determine the threshold and input firing rate values that maximize the accuracy.

4 | DIGITAL SNN ACCELERATOR

The architecture of the AI-SoC includes a mixed-signal SNN accelerator as shown in Figure ref fig:mixedsocarch. This accelerator comprises the UWA-SNN circuit in Figure 2 and a digital *SNN Controller*, which is designed to receive instructions from the core and control the UWA-SNN circuit. In this study, we designed an SNN controller to enable the completion of the SNN accelerator and digital SNN processor, which are functionally equivalent to the UWA-SNN circuit, to enable the FPGA prototyping of the AI-SoC.

The SNN controller was designed with two interfaces: a communication interface with digital IP and an interface with a UWA-SNN circuit. The former interface was designed as an APB slave interface that can communicate with the core by connecting to the system interconnect, whereas the latter interface was designed with input/output AER signals and setting signals to communicate with the UWA-SNN, which exchanges spike event information based on the existing AER and sets the weight and threshold using the setting signals. For inference, the SNN controller follows a series of operations: the core writes the weight and threshold values into the SNN controller, which then writes these values into the UWA-SNN circuit by setting the signals for the initial setup. After the core writes AER data for the input spikes, the SNN controller generates an AER signal to deliver the input spikes to the SNN circuit. The SNN controller then receives the inference results from the UWA-SNN via AER signals, decodes this information, and counts the number of output spikes generated at each node in the SNN output layer, thereby storing the results in a register. Finally, the core reads these values and determines the class of the node with the highest number of output spikes based on the inference.

Subsequently, we design the key IP, the digital SNN processor, as illustrated in Figure 4, which enables the FPGA prototyping of the entire SoC with the UWA-SNN. The external interface of this processor was designed to exchange input/output spikes with AER signals, similar to the UWA-SNN circuit, and set the weight and threshold using setting signals. The digital SNN processor consists of an *SNN processing unit*, which performs SNN

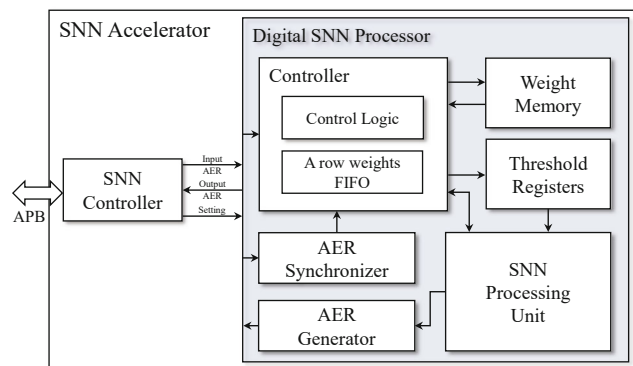


FIGURE 4 Architecture of the SNN accelerator including the digital SNN processor.

computations for input spikes, *weight memory*, which stores synapse weight values, *threshold registers*, which stores neuron thresholds, an *AER synchronizer*, which synchronizes asynchronous AER signals, an *AER generator*, which encodes output spikes produced by the SNN processing unit into AER signals, and a *controller*, which decodes synchronized AER signals to obtain input spikes and controls the SNN processing unit and weight memory to perform SNN computations for input spikes.

The digital SNN processor performs a series of operations designed to complete SNN inference. Initially, the SNN controller transmits the weight and threshold values of the SNN model to the digital SNN processor by setting the signals. The internal controller of the digital SNN processor then stores these weight and threshold values in the weight memory and threshold registers, respectively, thus completing the initial setup for SNN inference. Second, the AER synchronizer synchronizes the asynchronous AER signals entering the input AER port. The AER signals are asynchronous because the target UWA-SNN circuit uses asynchronous input and output signals. In the third step, the controller decodes the synchronized AER signals to obtain the input spike information, retrieves the weight values of the synapses connected to the input spikes of each layer from the weight memory, and delivers them to the SNN processing unit. The input spikes for each layer are obtained by decoding the AER signals for the first layer and the output spikes generated from the previous layer by the SNN processing unit for the subsequent layers. Fourth, the SNN processing unit accumulates the weight values received from the controller in the membrane potential registers and generates output spikes when the membrane potential exceeds a threshold value. If these output spikes are not generated from the nodes of the last layer, they become input spikes for the next layer, and the SNN processing unit requests the controller for the weights of the next layer. However,

if the spikes originate from the last layer, they are delivered to the AER generator. Finally, the AER generator encodes the output spikes from the nodes of the last layer into the AER and transmits them to the SNN controller via the output AER signals.

5 | DEVELOPMENT OF SNX

We developed SNX to facilitate the development of the AI-SoC with the UWA-SNN. To develop SNX, we built it upon RISC-V eXpress (RVX) [20] to augment its functionalities. For reference, the RVX is an EDA tool enabling developers to construct an SoC platform rapidly and conveniently using various RISC-V cores. RVX provides users with a selection of various IPs for the SoC configuration, accepts user-selected IPs as input, and generates the RTL code and development environment for the SoC constructed from these IPs. It has been successfully applied in the research and development of low-power SoCs intended for edge devices [21–24].

We embedded our developed UWA-SNN training methodology and SNN accelerator into the RVX to develop SNX. Additionally, we extended SNX to optionally support a spike generator hardware that can be used for UWA-SNN-based AI-SoC development. This spike generator utilizes rate coding, a common spike-coding technique, and is designed using the architecture illustrated in Figure 5. This spike generator reads the input values from the input memory, compares the 8-bit input values with 8-bit pseudo-random numbers generated by a linear feedback shift register (LFSR), generates spikes based on these comparisons, and stores the spike information in the spike memory.

SNX uses an architectural description of the target AI-SoC, including the SNN accelerator, from the developer as the input. By leveraging the existing functionalities supported by RVX, SNX automatically generates RTL code for the complete SoC along with FPGA prototypes and an SDK. SNX facilitates the entire design flow of the AI-SoC by supporting both the hardware and software

development processes, as highlighted in green and blue, respectively, in Figure 6.

To streamline the software development process, we initially developed and subsequently ported a program to SNX that trained the UWA-SNN model based on the methodology outlined in Section 3. To further aid in AI application development using the trained UWA-SNN model and the SNN accelerator, we formulated an API necessary for preprocessing during inference and SNN accelerator control and integrated this API into SNX.

6 | EXPERIMENTAL WORK

Furthermore, this study evaluated the proposed training methodology through experiments, along with the validation of SNX. In this process, we utilized the MNIST dataset [25] and the face/non-face image dataset (FACE) [26]. As illustrated in Figure 7a, the MNIST dataset consists of 8-bit grayscale images of handwritten numbers, whereas, as shown in Figure 7b, the FACE dataset comprises 8-bit RGB color images of human faces and non-faces, intended for binary image classification to discern whether an image is a human face.

Initially, the proposed SNN training methodology was evaluated using the MNIST and FACE datasets. We configured the UWA-SNN model trained on the MNIST

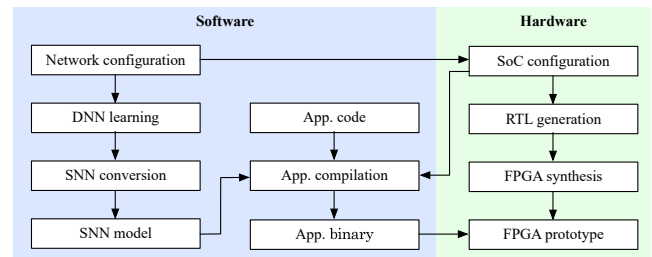


FIGURE 6 SNX-supported AI-SoC design flow.

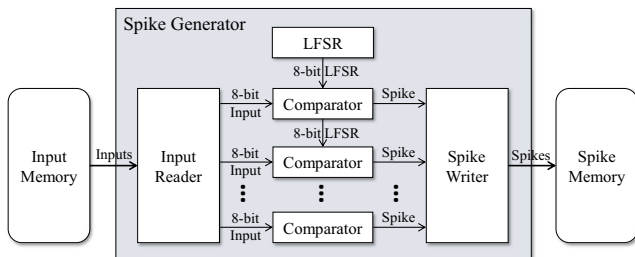


FIGURE 5 Architecture of the spike generator supported by SNX.



FIGURE 7 (a) MNIST and (b) FACE dataset images.

dataset with a threshold of 800 and an input firing rate of 500, whereas the model trained on the FACE dataset had a threshold of 3500 and an input firing rate of 300. The network structure of the model trained using the MNIST/FACE dataset comprised 784/2352 nodes in the input layer, 255/255 nodes in hidden layer 1, 84/84 nodes in hidden layer 2, and 10/2 nodes in the output layer. This configuration resulted in the accuracies of 92.50% for MNIST and 96.82% for FACE. A lower accuracy was observed for MNIST, potentially owing to the loss of information from clipping applications in MNIST, which require more complex differentiation than FACE. Despite the limitation that our proposed methodology is most effective for lower-complexity datasets, it remains valuable as the only known training method for UWA-SNNs.

Subsequently, we developed an AI-SoC and applied it to both the MNIST and FACE datasets using SNX. First, the MNIST-specific AI-SoC was configured with an SNN accelerator and a RISC-V ORCA [27] core, alongside a 128 kB SRAM, SDRAM, controller, IROM, platform controller, and UART, SPI as shown in Figure 8. The RTL simulation and FPGA prototyping were performed using the RTL code automatically generated by SNX. Using the SDK provided by SNX, we developed an MNIST application and confirmed that it functioned correctly in the FPGA prototype.

Subsequently, for FACE, we developed a facial recognition system for edge cloud systems composed of a FACE-specific AI-SoC and a server, leveraging SNX to prove that even more complex AI-SoC architectures could be developed. In this system, the FACE-specific AI-SoC, serving as an edge device, determines whether the input image contains a face, and if so, transmits the image to the server via Bluetooth. The server then runs a DNN-based facial distinction application and sends the results to the AI-SoC. As illustrated in Figure 9, we used two high-performance RISC-V rockets [28] as cores in the FACE-specific AI-SoC architecture. *Core 0* handles image resizing and image transmission to the

server via Bluetooth. *Core 1* is responsible for generating spikes for the resized images, controlling the UWA-SNN, and determining recognition results. Our validation confirmed that the system operated normally, demonstrating that SNX effectively provides an integrated development environment for AI systems.

Finally, we conducted an experiment to demonstrate the advantages of evaluating the application performance and estimating SoC power even before the AI-SoC is finalized into a chip. First, Table 1 and Table 2 present detailed power estimation results for each layer within the analog circuits designed with 28 nm FDSOI technology for the MNIST and FACE datasets, respectively. From these tables, it can be observed that the power consumption is proportional to the number of synapses and neurons. We then synthesized the RTL code obtained from SNX using the same 28 nm FDSOI technology and obtained the estimated power consumption for each application using Synopsys Primetime. We substituted

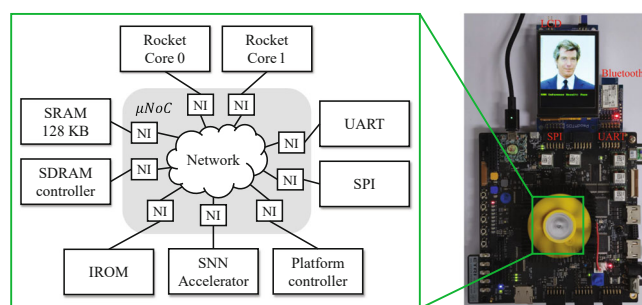


FIGURE 9 SoC architecture and its FPGA prototype for FACE.

TABLE 1 Power consumption estimation of the analog UWA-SNN circuit for MNIST (W).

	Neuron	Synapse	Overall
Hidden layer 1	1.61E-04	2.44E-03	2.60E-03
Hidden layer 2	5.27E-05	2.83E-04	3.35E-04
Output layer	6.27E-06	2.00E-05	2.63E-05
All layers	2.19E-04	2.75E-03	2.97E-03

TABLE 2 Power consumption estimation of the analog UWA-SNN circuit for FACE (W).

	Neuron	Synapse	Overall
Hidden layer 1	1.61E-04	7.33E-03	7.49E-03
Hidden layer 2	5.27E-05	2.83E-04	3.35E-04
Output layer	1.25E-06	1.22E-05	1.34E-05
All layers	2.14E-04	7.62E-03	7.84E-03

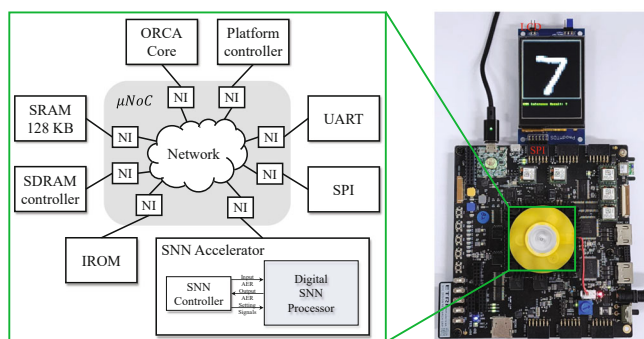


FIGURE 8 SoC architecture and its FPGA prototype for MNIST.

TABLE 3 Estimated power/performance for the mixed-signal UWA-SNN accelerator and entire AI-SoC based on SNX predictions for fabrication using 28 nm FDSOI technology.

	Mixed signal SNN Acc.	Entire SoC	Exec. time
MNIST	7.84 mW	49.38 mW	1.033 ms
FACE	2.97 mW	60.85 mW	1.732 ms

the power value of the digital SNN processor from the power report with the power value of an analog circuit designed using the same process to provide a power estimation of the entire SoC and mixed-signal SNN accelerator, as listed in Table 3. Additionally, we derived the results for the execution time of the application on each SoC platform. The average execution time for SNN inference for 10,000 MNIST test images is 1.033 ms, and for 1732 FACE test images, it is 1.825 ms.

7 | CONCLUSION

We developed SNX to facilitate the quick and easy development of SoCs with a UWA-SNN. To realize this, we first proposed a unique training methodology for the UWA-SNN and built an automatic application development environment based on it.

Furthermore, we designed a digital SNN accelerator equivalent to an analog UWA-SNN circuit for the functional and performance evaluation of target applications on FPGA prototypes before final chip fabrication. Finally, we developed SNX, which automatically generates the RTL code for the SoC, FPGA prototypes, and SDK based on the architecture set up by the developer. We demonstrated the ability of SNX to expedite AI-SoC development, as originally intended, through AI-SoCs specifically designed for the MNIST and FACE datasets using SNX. This establishes the value and effectiveness of the proposed tool for the rapid and seamless development of low-power AI-SoCs.

ORCID

Kyuseung Han  <https://orcid.org/0000-0002-9151-3447>

Kwang-Il Oh  <https://orcid.org/0000-0002-8715-7929>

Woojoo Lee  <https://orcid.org/0000-0001-5736-4583>

REFERENCES

- S. Davidson and S. B. Furber, *Comparison of Artificial and Spiking Neural Networks on Digital Hardware*, *Front. Neurosci.* **15** (2021), 651141.
- M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, and Y. Liao, *Loihi: A Neuromorphic Manycore Processor With On-Chip Learning*, *IEEE Micro* **38** (2018), no. 1, 82–99.
- S. Uenohara and K. Aihara, *CMOS Mixed-Signal Spiking Neural Network Circuit Using a Time-Domain Digital-To-Analog Converter*, (IEEE Int. Symp. Circuits and Systems, Daegu, Rep. of Korea), 2021, pp. 1–5.
- J. K. Eshraghian, X. Wang, and W. D. Lu, *Memristor-Based Binarized Spiking Neural Networks: Challenges and Applications*, *IEEE Nanotechnol. Mag.* **16** (2022), no. 2, 14–23.
- S. Hwang, J. Yu, G. H. Lee, M. S. Song, J. Chang, K. K. Min, T. Jang, J.-H. Lee, B.-G. Park, and H. Kim, *Capacitor-Based Synaptic Devices for Hardware Spiking Neural Networks*, *IEEE Electron Device Lett.* **43** (2022), no. 4, 549–552.
- K.-I. Oh, S.-E. Kim, T.-W. Kang, H. Kim, and J.-J. Lee, *Design of 0.9v-Operating Low-Power AI SNN Analog Core*, (Autumn Annual Conference of IEIE, Gwangju, Rep. of Korea), 2020, pp. 932–933.
- S. Hwang, H. Kim, J. Park, M.-W. Kwon, M.-H. Baek, J.-J. Lee, and B.-G. Park, *System-Level Simulation of Hardware Spiking Neural Network Based on Synaptic Transistors and I&F Neuron Circuits*, *IEEE Electron Device Lett.* **39** (2018), no. 9, 1441–1444.
- A. Joubert, B. Belhadj, O. Temam, and R. Hélot, *Hardware Spiking Neurons Design: Analog or Digital?* (Int. Joint Conf. on Neural Networks, Brisbane, Australia), 2012, pp. 1–5.
- S. Kim, S. Kim, S. Um, S. Kim, K. Kim, and H.-J. Yoo, *Neuro-CIM: ADC-Less Neuromorphic Computing-In-Memory Processor With Operation Gating/Stopping and Digital–Analog Networks*, *IEEE J. Solid-State Circ.* **2023** (2023), 2931–2945.
- G. Bi and M. Poo, *Synaptic Modifications in Cultured Hippocampal Neurons: Dependence on Spike Timing, Synaptic Strength, and Postsynaptic Cell Type*, *J. Neurosci.* **18** (1998), no. 24, 10464–10472.
- P. U. Diehl and M. Cook, *Unsupervised Learning Of Digit Recognition Using Spike-Timing-Dependent Plasticity*, *Front. computat. Neurosci.* **9** (2015), 99.
- S. Song, K. D. Miller, and L. F. Abbott, *Competitive Hebbian Learning Through Spike-Timing-Dependent Synaptic Plasticity*, *Nature Neurosci.* **3** (2000), no. 9, 919–926.
- P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu, and M. Pfeiffer, *Fast-Classifying, High-Accuracy Spiking Deep Networks Through Weight and Threshold Balancing*, (Int. Joint Conf. Neural Networks, Killarney, Ireland), 2015, pp. 1–8.
- B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, and S.-C. Liu, *Conversion of Continuous-Valued Deep Networks to Efficient Event-Driven Networks for Image Classification*, *Frontiers Neurosci.* **11** (2017), 682.
- W. Guo, M. E. Fouda, A. M. Eltawil, and K. N. Salama, *Neural Coding in Spiking Neural Networks: A Comparative Study for Robust Neuromorphic Systems*, *Front. Neurosci.* **15** (2021), 638474.
- R. Banner, Y. Nahshan, and D. Soudry, *Post Training 4-Bit Quantization of Convolutional Networks for Rapid-Deployment*, *Adv. Neural Inform. Process. Sys.* **32** (2019), 7950–7958.
- Y. Choukroun, E. Kravchik, F. Yang, and P. Kisilev, *Low-Bit Quantization of Neural Networks for Efficient Inference*, (IEEE/CVF Int. Conf. Computer Vision Workshop, Seoul, Rep. Korea), 2019, pp. 3009–3018.
- B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, *Quantization and Training of*

Neural Networks for Efficient Integer-Arithmetic-Only Inference, (Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA), 2018, pp. 2704–2713.

19. R. Krishnamoorthi, Quantizing Deep Convolutional Networks for Efficient Inference: A Whitepaper, arXiv preprint, 2018. <https://doi.org/10.48550/arXiv.1806.08342>
20. K. Han, S. Lee, K.-I. Oh, Y. Bae, H. Jang, J.-J. Lee, W. Lee, and M. Pedram, *Developing TEI-Aware Ultralow-Power SoC Platforms for IoT End Nodes*, IEEE Int. Things J. **8** (2021), no. 6, 4642–4656.
21. E. Choi, J. Park, K. Lee, J.-J. Lee, K. Han, and W. Lee, *Day-Night Architecture: Development of an Ultra-Low Power RISC-V Processor for Wearable Anomaly Detection*, J. Syst. Architect. **152** (2024), 103161. <https://doi.org/10.1016/j.sysarc.2024.103161>
22. H. Jang, K. Han, S. Lee, J.-J. Lee, S.-Y. Lee, J.-H. Lee, and W. Lee, *Developing a Multicore Platform Utilizing Open RISC-V Cores*, IEEE Access **9** (2021), 120010–120023.
23. J. Park, E. Choi, K. Lee, J.-J. Lee, K. Han, and W. Lee, *Developing an Ultra-Low Power RISC-V Processor for Anomaly Detection*, (Design, Automation & Test in Europe Conf. & Exhibition, Antwerp, Belgium), 2023, pp. 1–2.
24. J. Park, K. Han, E. Choi, S. Lee, J.-J. Lee, W. Lee, and M. Pedram, *Florian: Developing a Low-Power RISC-V Multicore Processor With a Shared Lightweight FPU*, (IEEE/ACM Int. Symp. Low Power Electronics and Design, Vienna, Austria), 2023, pp. 1–6.
25. Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-Based Learning Applied to Document Recognition*, Proc. IEEE **86** (1998), no. 11, 2278–2324.
26. J. Rodge, Binary-Image-Classifer-pytorch, 2024. <https://github.com/jayrodge/Binary-Image-Classifer-PyTorch/tree/master/face>
27. ORCA, Orca-RISC-V, 2024. <https://github.com/kammoh/ORCA-risc-v>
28. K. Asanović, R. Avizienis, J. Bachrach, S. Beamer, D. Biancolin, C. Celio, H. Cook, D. Dabbelt, J. Hauser, A. Izraelevitz, S. Karandikar, B. Keller, D. Kim, J. Koenig, Y. Lee, E. Love, M. Maas, A. Magyar, H. Mao, M. Moreto, A. Ou, D. A. Patterson, B. Richards, C. Schmidt, S. Twigg, H. Vo, and A. Waterman, *The Rocket Chip Generator*. UCB/EECS-2016-17. EECS Department, University of California, Berkeley, 2016. <http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.html>

AUTHOR BIOGRAPHIES



Hyeonguk Jang received his B.S. and M.S. degrees in electrical engineering from Gyeongsang National University, Jinju, Republic of Korea, in 2013 and 2015, respectively, and his Ph.D. degree in ICT from the University of Science and Technology, Daejeon, Republic of Korea, in 2021. He is currently working as a postdoctoral researcher at the

Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea. His interests include spiking and deep neural networks, and the design of RISC-V-based low-power embedded systems.



Kyuseung Han received his B.S. and Ph.D. degrees in electrical engineering and computer science from Seoul National University, Seoul, Republic of Korea, in 2008 and 2013, respectively. Since 2014, he has been working at the Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea, where he is currently a principal researcher. His research focuses on system-on-chip architectures and their design automation techniques for low-power embedded systems.



Kwang-Il Oh received his B.S. degree in electrical engineering from Kyungpook National University, Daegu, Republic of Korea, in 2002, and his M.S. and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology, Daejeon, Republic of Korea, in 2004 and 2009, respectively. From March 2009 to February 2015, he was a senior engineer at LX Semicon Co., Ltd., Daejeon, Republic of Korea, where he was involved in the high-bandwidth interface circuit design for LCD modules. Since March 2015, he has been with the Electronics and Telecommunications Research Institute in Daejeon, Republic of Korea, where he is involved in neuromorphic chip design. His research interests include low-power analog circuits, signal integrity, and neuromorphic circuits.



Sukho Lee received his Ph.D. degree in information communications engineering from Chungnam National University, Daejeon, Republic of Korea, in 2010. He is currently a principal researcher in the AI Edge SoC Research Section of the Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea. His current research interests include ultralow power system-on-chip design, embedded system design, video codec design, and video image processing.



Jae-Jin Lee received his B.S., M. S., and Ph.D. degrees in computer engineering from Chungbuk National University, Cheongju, Republic of Korea, in 2000, 2003, and 2007, respectively. He is currently a Project Leader with the Low-Power AI System-on-Chip

Design Research Division, Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea. His research interests include ultralow-power embedded RISC-V processor designs and event driven neuromorphic computing architectures for brain-inspired spiking deep neural networks.



Woojoo Lee received the B.S. degree in electrical engineering from Seoul National University, Seoul, South Korea, in 2007, and the M.S. and Ph.D. degrees in electrical engineering from the University of Southern California, Los Angeles, CA, USA, in 2010 and

2015, respectively. He was a Senior Researcher with the system on chip (SoC) Design Research Group,

Electronics and Telecommunications Research Institute, Daejeon, Korea, from 2015 to 2016, and an Assistant Professor with the Department of Electrical Engineering, Myongji University, Seoul, Korea, from 2017 to 2018. He is currently an Associate Professor with the School of Electrical and Electronics Engineering and also holds a joint appointment with the Department of Intelligent Semiconductor Engineering at Chung-Ang University, Seoul, Korea. His research interests include low-power VLSI, SoC, Processing-in-Memory, neuromorphic processor designs, and system-level power and thermal management.

How to cite this article: H. Jang, K. Han, K.-I. Oh, S. Lee, J.-J. Lee, and W. Lee, *SNN eXpress: Streamlining Low-Power AI-SoC Development With Unsigned Weight Accumulation Spiking Neural Network*, ETRI Journal **46** (2024), 829–838, DOI [10.4218/etrij.2024-0114](https://doi.org/10.4218/etrij.2024-0114)