**Original Manuscript ID:** Access-2021-01435

**Original Article Title: “**Accelerating Spike-by-Spike Neural Networks on FPGA with Hybrid Custom Floating-Point and Logarithmic Dot-Product Approximation”

**To:** IEEE Access Editor

**Re:** Response to reviewers

Dear Editor,

Thank you for allowing a resubmission of our manuscript, with an opportunity to address the reviewers’ comments.

We are uploading (a) our point-by-point response to the comments (below) (response to reviewers), (b) an updated manuscript with yellow highlighting indicating changes, and (c) a clean updated manuscript without highlights (PDF main document).

Best regards,

Yarib Nevarez et al.

**Reviewer#1, Concern # 1:**

Section I / Paragraph 2: It mentions that “Among the family of SNNs, the SbS neural network is remarkable.” Here, several things are not clear.

* Why is the SbS network remarkable?
* What does the family of SNNs refer to and how does one family differ from each other?
* The motivation why studying and accelerating the SbS network are not clear.

**Concern:** Why is the SbS network remarkable?

**Author response:** The Spike-by-Spike (SbS) neural network is a remarkable model for its reduced complexity, which is on the less realistic side of the SNN scale of biological realism. Consequently, the hardware complexity of SbS network implementations is greatly reduced. In spite of this, SbS still uses stochastic spikes as a means of transmitting information between populations of neurons and thus retains the advantageous robustness of SNNs. These advantages range from offering the potential to achieve energy efficiency closer to that of the human brain, superior resistance against adversary attacks, and the possibility of more efficient asynchronous parallelization.

**Author action:** We updated the manuscript by clarifying this point in Section I (page 2).

**Concern:** What does the family of SNNs refer to and how does one family differ from each other?

**Author response:** SNNs emulate the real behavior of neurons in different levels of detail. The more detailed the biological part is emulated, the greater the computational complexity. Most of today’s SNNs use a very detailed model (e.g., Leaky Integrate-and-Fire (LIF)). In contrast, the SbS neural network is a model on the less realistic side of the SNN scale of biological realism.

**Author action:** We updated the manuscript by clarifying this point in Section I (page 2).

**Concern:** The motivation why studying and accelerating the SbS network are not clear.

**Author response:** Hardware accelerators that focus on SbS have only been partially investigated so far. Enhanced SbS accelerators will have a double impact. From an engineering point of view, they will contribute to the deployment of robust neural networks in small embedded systems; from a scientific point of view, they will facilitate fundamental research for neuroscience.

**Author action:** We updated the manuscript by clarifying this point in Section I (page 2).

**Reviewer#1, Concern # 2:**

Section I / Paragraph 2: It mentions that “These properties place the SbS network in between non-spiking NN and stochastically spiking NN, offering advantages from both structures.” It is not clear how the SbS network can offer advantages from both structures.

**Author response:** On one hand, the SbS model incorporates the inherent robustness of SNNs, which provide the possibility of more efficient asynchronous parallelization and superior resilience against disturbances based on the synaptic stochasticity; on the other hand, the SbS model incorporates the regular flow of information from CNNs, which are expressed on explicit vector operations.

**Author action:** We updated the manuscript by clarifying this point in Section I (page 2).

**Reviewer#1, Concern # 3:**

Section I / Paragraph 3: It mentions that “deep SbS networks are high compute and data intensive, …” It is suggested to provide the supporting data.

**Author response:** The computational cost of SbS network models is higher compared to equivalent CNN models and lower compared to regular SNN models (e.g., Leaky Integrate-and-Fire). We have added the SbS algorithm and the computational cost in the paper.

**Author action:** We updated the manuscript by providing a simplified SbS algorithm and the computational cost in multiply-accumulate (MAC) operations in Section III- C (page 4 and 5).

**Reviewer#1, Concern # 4:**

Section III-A: The explanation of SbS fundamental directly focuses on its computational aspects, without discussing the basic network overview, such as the network architecture/topology, synaptic connections, spike/information coding, etc. Therefore, this makes the following discussion difficult to follow.

**Author response / action:** We updated the manuscript by adding further explanations of SbS model in Section III – A (page 4), the basic network overview in Section III – B (page 4), a simplified algorithm in Section III – C (page 5), and detailed algorithm in supplementary materials (page 17 and 18).

**Reviewer#1, Concern # 5:**

Section III-A / Paragraph 4:

* What does the “tensor flow network” mean?
* What is the type of noise used for the observation in Fig. 2?
* It is not clear the context of the number of spikes that is discussed in this paragraph.

**Concern:** What does the “tensor flow network” mean?

**Author response:** This was refereeing to the CNN implemented in tensor flow.

**Author action:** We have corrected this in the paper, Section III – D (page 5).

**Concern:** What is the type of noise used for the observation in Fig. 2?

**Author response:** We used positive additive uniformly distributed noise.

**Author action:** We have added this information in Fig. 2 (page 5).

**Concern:** It is not clear the context of the number of spikes that is discussed in this paragraph.

**Author response:** The number of spikes is denoted as the number of iterations the SbS algorithm updates its internal representation (as a generative model).

**Author action:** We updated the manuscript by clarifying that the number of spikes represents the number of iterations for the SbS algorithm in Section III – B, C and D (page 4 and 5).

**Reviewer#1, Concern # 6:**

Section IV-A:

* It mentions that the system design from reference [46] is revisited. What are the differences between the proposed architecture in manuscript with the design from [46]? It is suggested to clarify the novelty.
* It mentions that “The hardware architecture can resize its resource utilization by changing the number of PUs instances, ...” Does it mean the proposed system supports the reconfiguration at run-time? If so, it is suggested to explain how the reconfiguration is performed.

**Concern:** It mentions that the system design from reference [46] is revisited. What are the differences between the proposed architecture in manuscript with the design from [46]? It is suggested to clarify the novelty.

**Author response:** As architectural novelty, our proposed approach in this publication is based on specialized heterogeneous processing units with approximate computing. In contrast, our previous work [46] is based on generic homogeneous accelerator units with standard floating-point, which represents elevated memory and computational costs.

**Author action:** We updated the manuscript by clarifying this point in the first two paragraphs in Section IV (page 5 and 6). In addition, we added a platform comparison table. This table compares our previous publication and the resulting platforms from the design exploration on this publication. This table contains resource utilization, power dissipation, latency, and accuracy. This table is included at the end of Section V-C (page 16).

**Concern: I**t mentions that “The hardware architecture can resize its resource utilization by changing the number of PUs instances, ...” Does it mean the proposed system supports the reconfiguration at run-time? If so, it is suggested to explain how the reconfiguration is performed.

**Author response:** The proposed system does not support the reconfiguration at run-time. The hardware architecture resizes its resource utilization by changing the number of PUs instances prior to the hardware synthesis.

**Author action:** We updated the manuscript by clarifying this point in section IV-A, second paragraph (page 6).

**Reviewer#1, Concern # 7:**

Section IV-B: It is still not clear how the bitwidth is decided for each parameter that is considered for the computation. It should be explained when discussing the custom floating-point and logarithmic representation. Otherwise, it seems like a pre-selected bitwidth without justification. For example, the partial discussion in the Section V-B.1 and Section V-B.2 can be used here, but should be supported with justification.

**Author response:**

We use either hybrid custom floating-point or logarithmic number representation on the synaptic weight matrix. We keep everything else with standard floating-point number representation.

Both the custom floating-point and logarithmic number representation have the same exponent bit width extracted from its floating-point representation. The mantissa bit width is a knob parameter to trade off resource utilization and quality-of-result. Removing mantissa results in the logarithmic number representation.

Since the synaptic weight matrix is composed of entries with normalized values, we have exponents ranging from negative values to zero.

Therefore, for each synaptic weight matrix, we look for the smallest value, which corresponds to the largest negative exponent in the values of the matrix. Hence, the bit width of the exponent is determined according to this value. Also, to further reduce the bit width, since all exponents are negative numbers, the sign bit is removed (to reduce bit width), then the exponent is handled as negative on the hardware.

**Author action:** We updated the manuscript by describing the method to determine the required bit width in Section IV-C (page 7 and 8), where we placed the formulas with their discussion and an illustrative figure.

**Reviewer#1, Concern # 8:**

Section V-A.2: What are the reasons of computing CONV layers (H2\_POOL and H3\_CONV) on processing units (PUs), and computing HY\_OUT on CPU. Isn’t it faster computing them all on the PUs?

**Author response:** Yes. Computing on hardware PUs is faster than on CPU. However, HY\_OUT is made up of a small vector of 10 neurons, which we decided to process on the CPU based on its processing latency of 4 microseconds. This latency is negligible compared to the overall performance assessment, accelerating HY\_OUT would yield a negligible gain. Furthermore, assigning a dedicated hardware PU to HY\_OUT would add data transfer and hardware interruption handling overheads, which makes this unprofitable.

**Author action:** We updated the manuscript by stating this point in Section V-A.2 (page 11).

**Reviewer#1, Concern # 9:**

Section V-B.1: It says that the detailed discussion on the number format is provided in Section IV-A, but the discussion is not there. It seems to be a wrong referencing and should be corrected.

**Author response:** Yes. Thanks for the observation.

**Author action:** We updated the manuscript by removing the content referring to Section IV-A. The paragraph containing this reference is removed as this content is now placed in Section IV-C. This content discussed the convenient properties of the synaptic weight matrix to ignore the sign bit, as well as the method for determining the bit width (page 7 and 8).

In short, we simplified the rest of the content in this section (Section V-B.1, page 13) simply by discussing the log-2 histograms of the synaptic weight matrices and using the formulas (from Section IV-C, page 7 and 8) to obtain the bit width.

**Reviewer#1, Concern # 10:**

Section V-C: Table 8 shows that the proposed architectures (hybrid custom floating-point and hybrid logarithmic) consume larger LUT and FF resources. What are the reasons? It is also suggested to provide a breakdown of the power consumption for compute and memory units.

**Author response:**

The dot-product with standard floating-point arithmetic (IEEE 754) utilizes floating-point operator cores (LogiCORE IPs) that are already implemented in other computational sections of the hardware kernel, this is reusage. However, the proposed architecture does not reuse the already instantiated LogiCORE IPs. Instead, the logic required for the hybrid custom floating-point and logarithmic approximation must be implemented.

In other words, when using standard floating-point (IEEE 754), the HLS tool implements floating-point operator cores (LogiCORE IPs) to perform floating-point arithmetic operations. These operator cores are implemented trading logic (LUT) resources for DSP usage according to the given implementation directives. These core instances can be reused over the entire hardware kernel. Our dot-product with standard floating-point computation utilizes one multiplier and one adder arithmetic core. These cores are reused from other computational blocks.

**Author action:**

We updated the manuscript by adding the discussion regarding the floating-point operator cores used in the PUs in Section V-A.2 (page 12). We added a table showing the resource utilization and power dissipation of the multiplier and adder floating-point cores (provided by the analysis resource viewer of Vivado HLS). We complement our discussion section with this topic (page 15). We support our statements with a Xilinx application note reference (Hrica, J., 2012. Floating-point design with vivado HLS. *Xilinx Application Note.*)

Regarding the breakdown of the power consumption, we added this information at the end of the discussion section (page 16). We added the breakdown of the power consumption for each platform architecture implemented: (1) our previous publication, (2) standard floating-point computation, (3) hybrid custom floating-point approximation, and (4) hybrid logarithmic approximation. This information is provided by the project summary overview, power on-chip section from Vivado.

**Reviewer#1, Concern # 11:**

Section V-A.3 / Paragraph 1: It mentions “This plot revels inherent …” Shouldn’t it be “This plot reveals inherent …”?

**Author response:** Thank you for this remark.

**Author action:** We updated the manuscript by correcting the mistake in Section V-A.3 (page 12).

**Reviewer#2, Concern # 1:**

This paper only uses MNIST as the data set for evaluation. SNN training algorithm is still in its infancy, but even so, it is not very suitable to take such a toy test as the evaluation benchmark for a hardware design. Existing SNN training studies have supported CIFAR-level test set and spiking-CNN. It is suggested that the authors improve the work in this regard.

**Author response:** We agree that MNIST is a basic data set unsuitable for evaluation of state-of-the-art machine learning algorithms. However, we use this basic machine learning task simply to provide a proof of concept to demonstrate the feasibility of our approximation technique for neural network accelerators.

**Author action:** We updated the manuscript by stating this point in Section I (page 3).

**Reviewer#2, Concern # 2:**

This paper stated that “… SbS networks provide numerous advantages over traditional ANNs and CNNs”. I don't think it is right: SNN may have the potential to surpass ANN, but it has a long way to go to prove itself in typical applications.

**Author response:** We agree that SNN have the potential to surpass ANN in typical applications; however, SNNs are still in their infancy. One of our motivations is to facilitate/support fundamental research for computational neuroscience.

**Author actions:** We updated the manuscript by stating the potential advantages of SNNs over traditional ANNs and CNNs (page 1 and 2). Moreover, we declare that one of our motivations is to facilitate fundamental research for neuroscience in Section I (page 2).

***Note:*** *References suggested by reviewers should only be added if it is relevant to the article and makes it more complete. Excessive cases of recommending non-relevant articles should be reported to ieeeaccesseic@ieee.org*