**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.

**Author response:**

**Author action:** We updated the manuscript by ….

To answer the questions, we will address first what are the advantages of SNN and how one family or alternative differ from the other one; then we will answer why SbS are remarkable, and finally the motivation to study and accelerate SbS.

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

Spiking neural n etworks (SNN) offer an alternative to standard CNNs with a high potential \cite{abderrahmane2020design} . One of the most interesting aspects of this approach is its inherent robustness: Like the brain, also spiking artificial neuronal networks can operate reliably using mechanisms that are inherently non-reliable – a prime examples in the brain are stochastic synapses \cite{smetters1996synaptic}. Even when using such unreliable and stochastic elements, spiking networks are not only remarkably reliable \cite{mcdonnell2011benefits} but can also achieve superior robustness against perturbations from noise in the input and adversary attacks \cite{ernst2007efficient, Dapello2020.06.16.154542}. Beside robustness, SNN have further advantages like the possibility of higher energy efficiency and more efficient asynchronous parallelization. For example, Loihi \cite{davies2018loihi}, a SNN developed by Intel, can solve LASSO optimization problems with an over three orders of magnitude better energy-delay product than conventional approaches. These advantages are motivating large research programs by major companies (e.g. Intel \cite{davies2018loihi} and IBM \cite{TrueNorth\_Trans15}) as well as pan-european projects in the domain of spiking networks \cite{Spinnaker\_TransSolid\_13}, and attempts to transfer features from SNNs to standard DNNs \cite{pfeiffer2018deep}.

The interest in SNN by industry and academia has produced a plethora of SNN architectures. As a reference, Tab.\ref{tab:comp\_spike\_hw} summarizes some of the more important ones (further details can be found on \cite{Spiking\_HW\_Review\_Trans19} for details).

… copy the rest...

**Author action:** We have udpated the introduction with the text…. (COPY PICTURE for consistency)

**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:**

**Author action:** We updated the manuscript by ….

**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:**

**Author action:** We updated the manuscript by ….

**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:**

**Author action:** We updated the manuscript by ….

**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.

**Author response:**

**Author action:** We updated the manuscript by ….

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

Section IV-A:

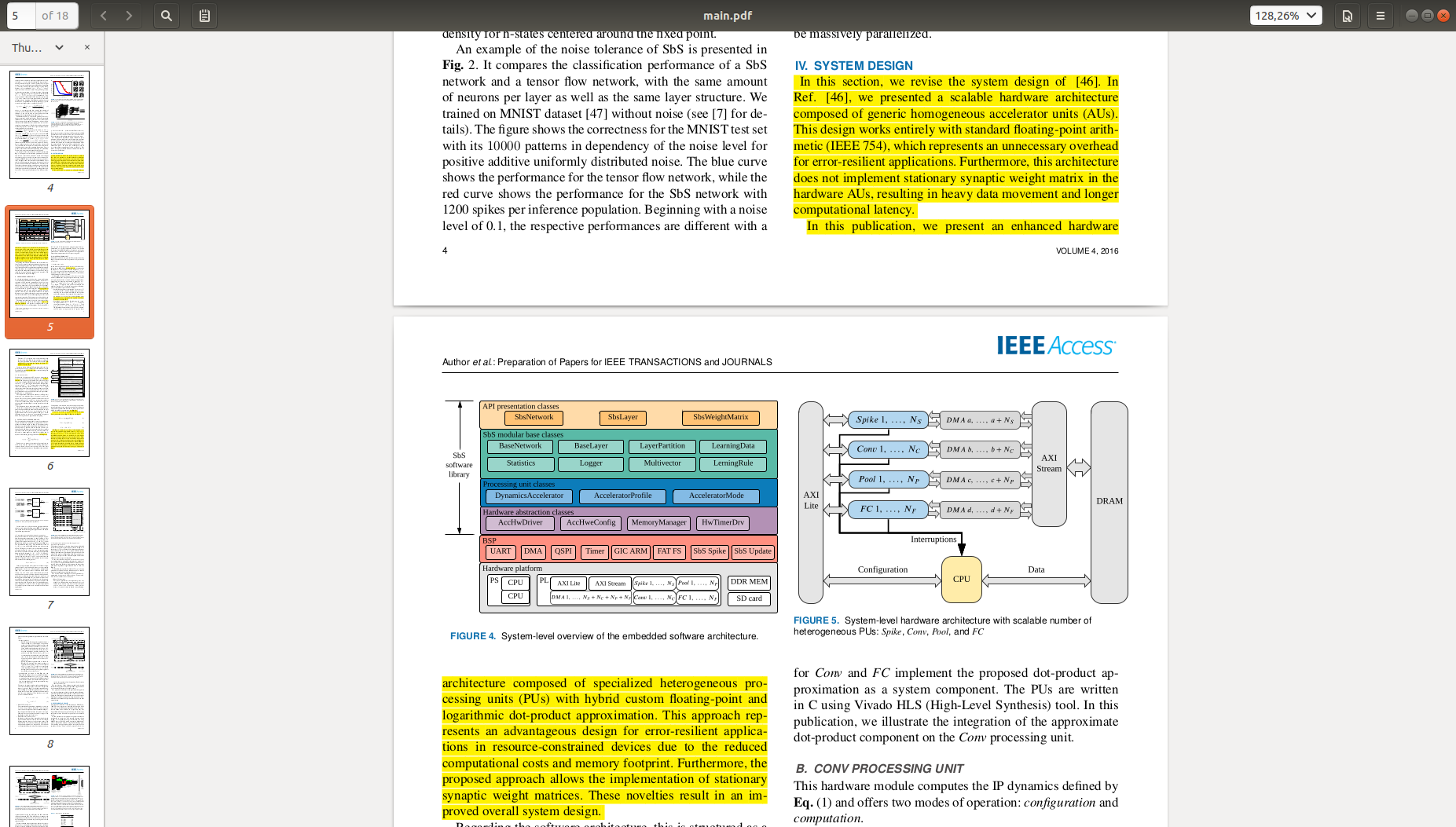
* It mentions that the system design from reference [46] is revisited. What are the differences between the proposed architecture in manuscript with the design from [46]? It is suggested to clarify the novelty.
* It mentions that “The hardware architecture can resize its resource utilization by changing 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.

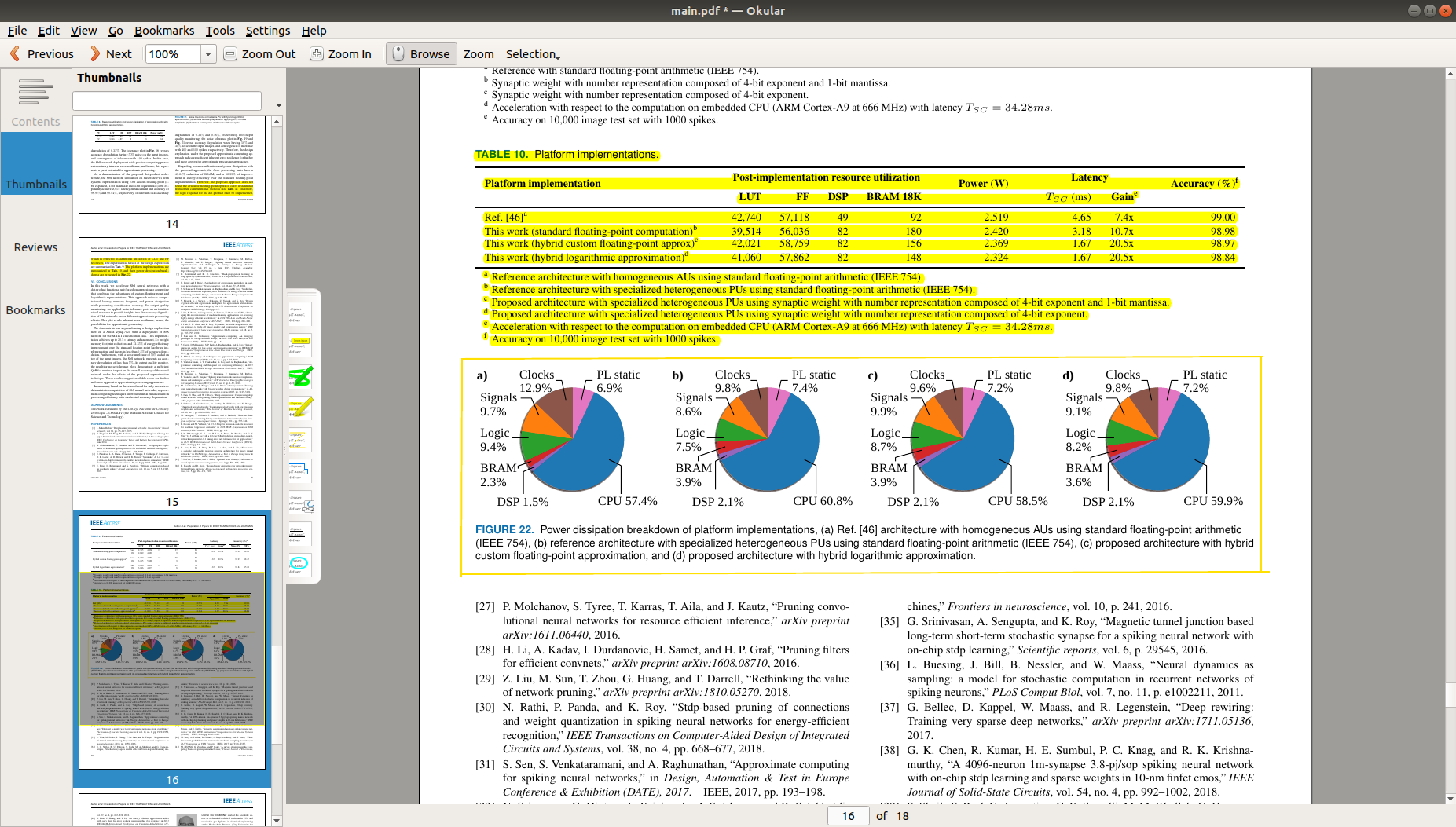
**Author response:**

* 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.
* 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 the first two paragraphs in Section IV.
* In addition, we added a platform comparison table. This table compares our previous publication in Ref. [46] 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 (results and discussion).





* We updated the manuscript by clarifying that the hardware architecture resizes its resource utilization by changing the number of PUs instances prior to the hardware synthesis. This is done in section IV-A, second paragraph.

|  |
| --- |
| The hardware architecture can resize its resource utilization by changing the number of PUs instances prior to the hardware synthesis, this provides scalability with a good trade-off between area and throughput. |

**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:**

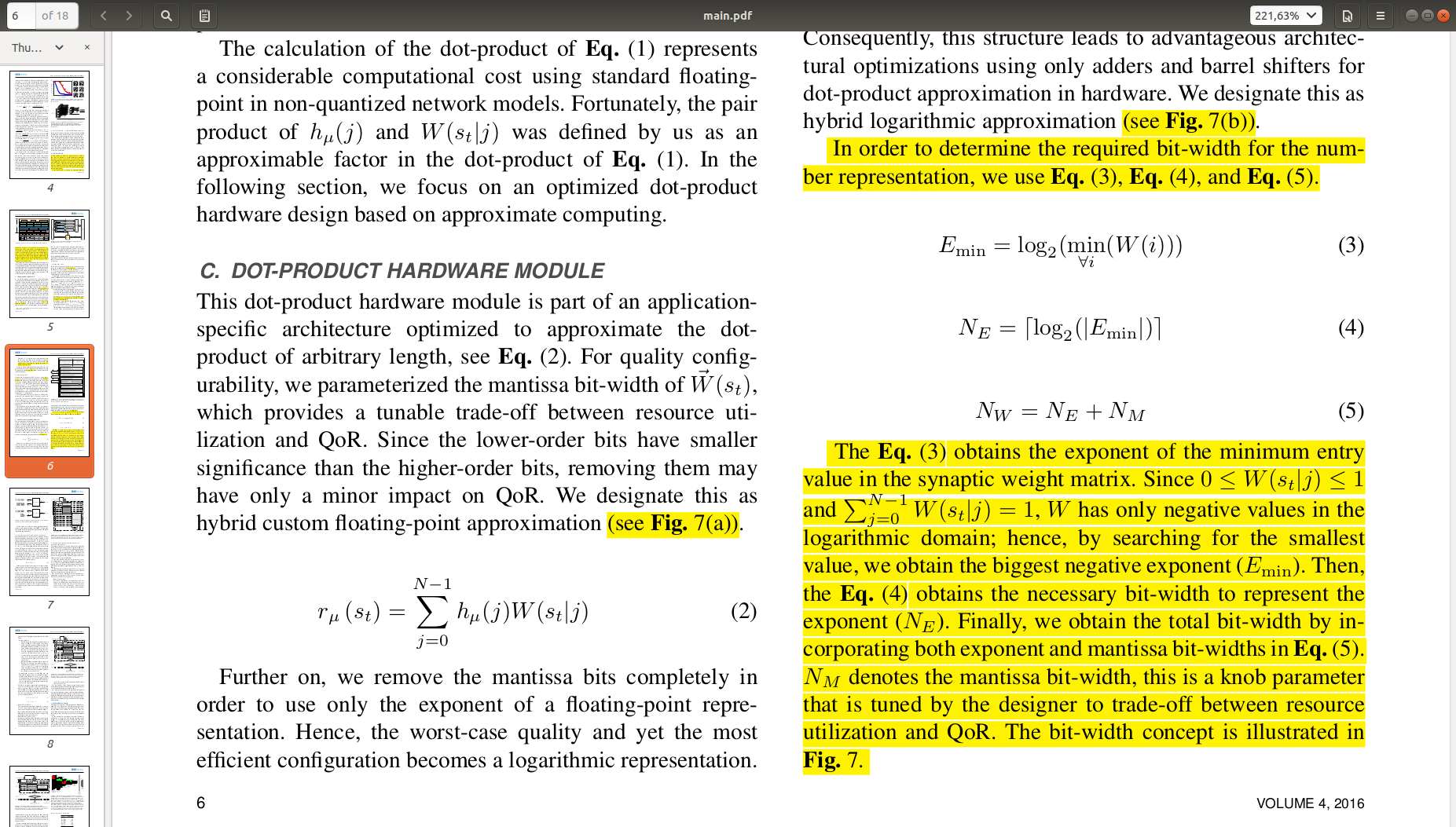
We use either hybrid custom floating-point or logarithmic number representation on the synaptic weight matrix. We keep everything else with standard floating-point number representation.

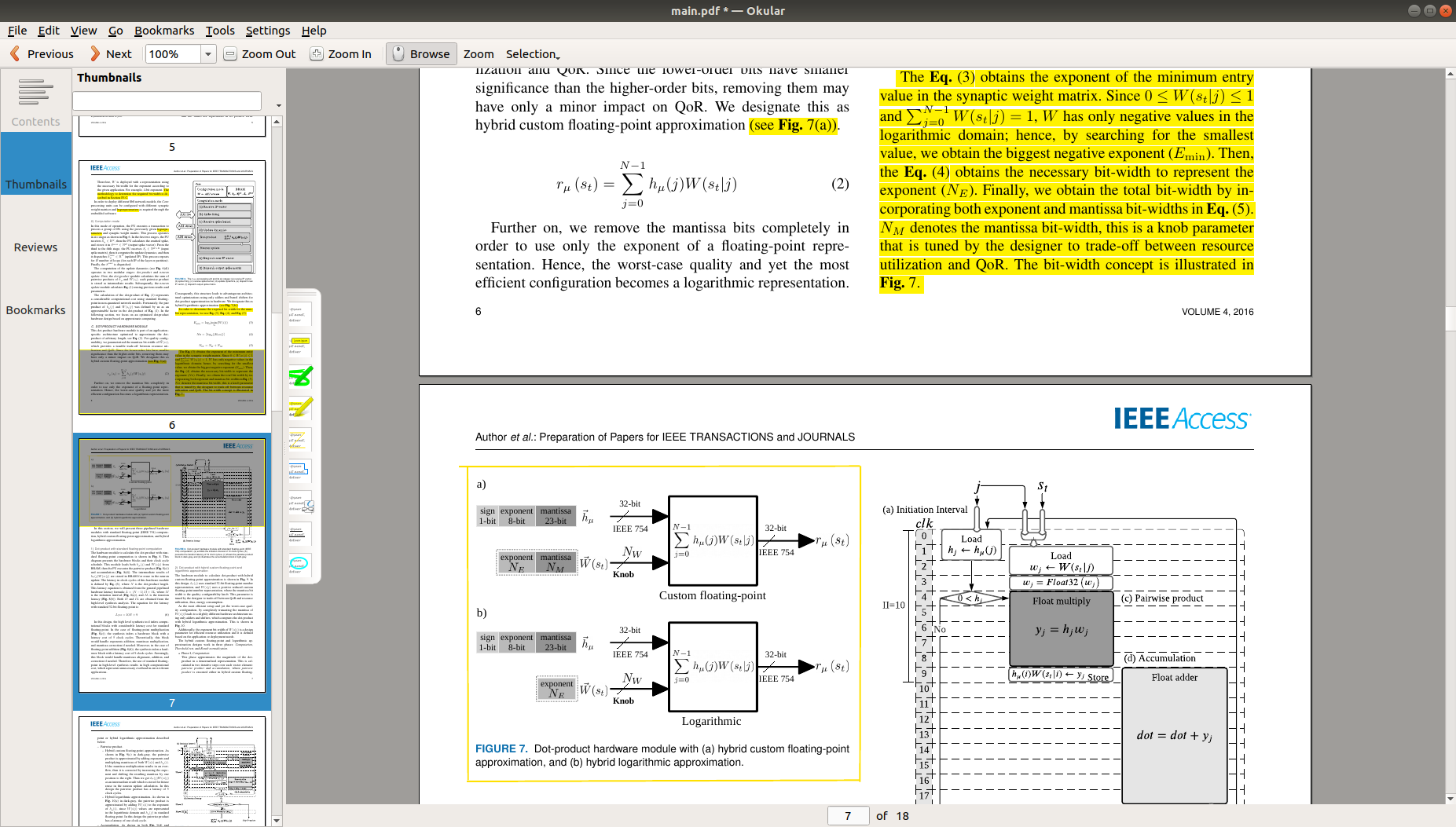
Both the custom floating-point and logarithmic number representation have the same exponent bit width extracted from its floating-point representation. The mantissa bit width is a knob parameter to trade off resource utilization and quality-of-result. A null mantissa bit-width results in the logarithmic 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. Therefore, 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 ignored (set/handled as negative in hardware).

**Author action:** We updated the manuscript by describing the method to determine the required bit width in Section IV-C, where we placed the formulas with their discussion and an illustrative figure. In Section IV-B, we added a reference to Section IV-C.





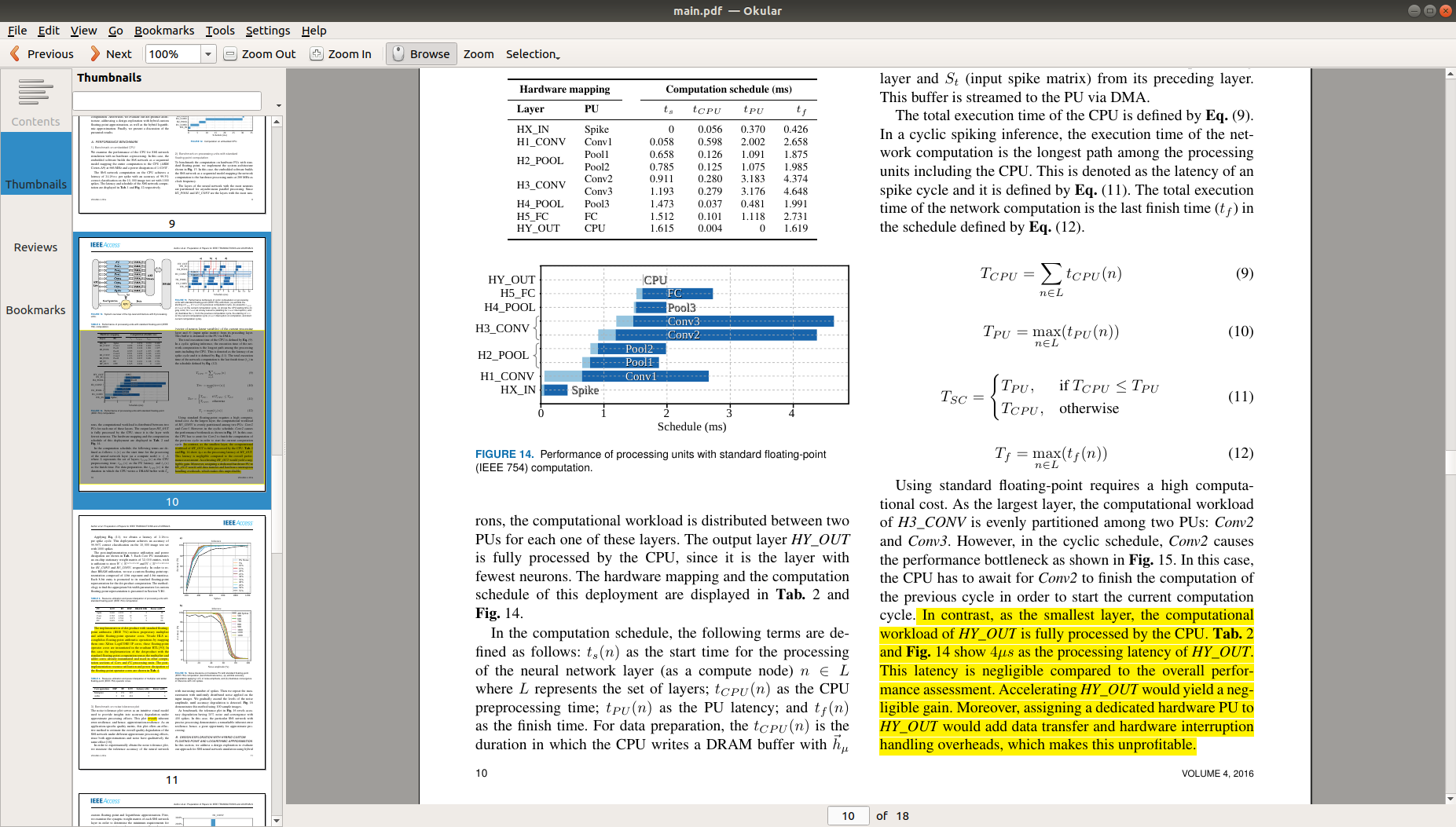
**Reviewer#1, Concern # 8:**

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 clarifying this point in Section V-A.2



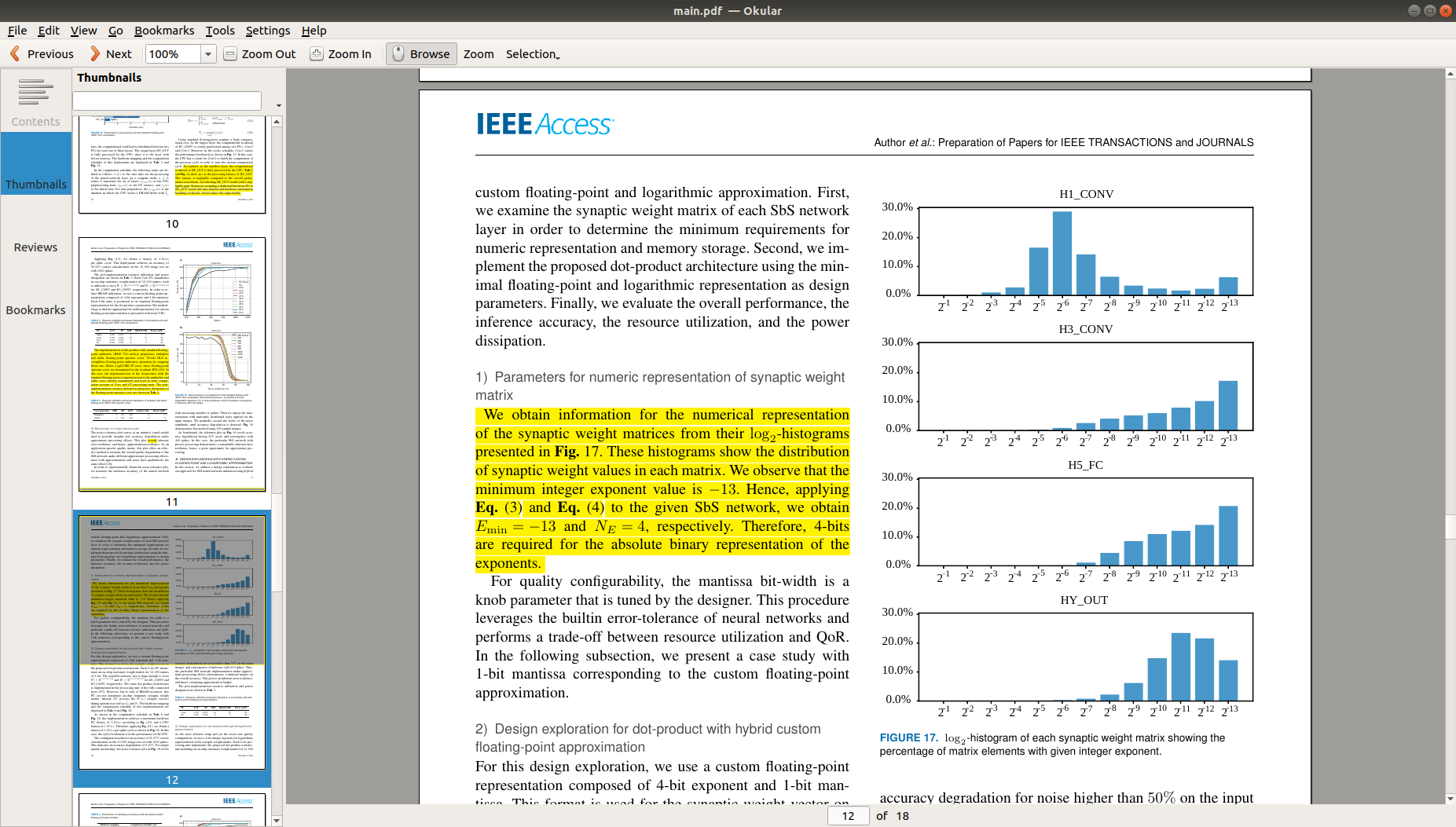
**Reviewer#1, Concern # 9:**

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.

We simplify the rest of the content in this section (Section V-B.1) simply by discussing the log-2 histograms of the synaptic weight matrices and using the formulas (from Section IV-C) 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.

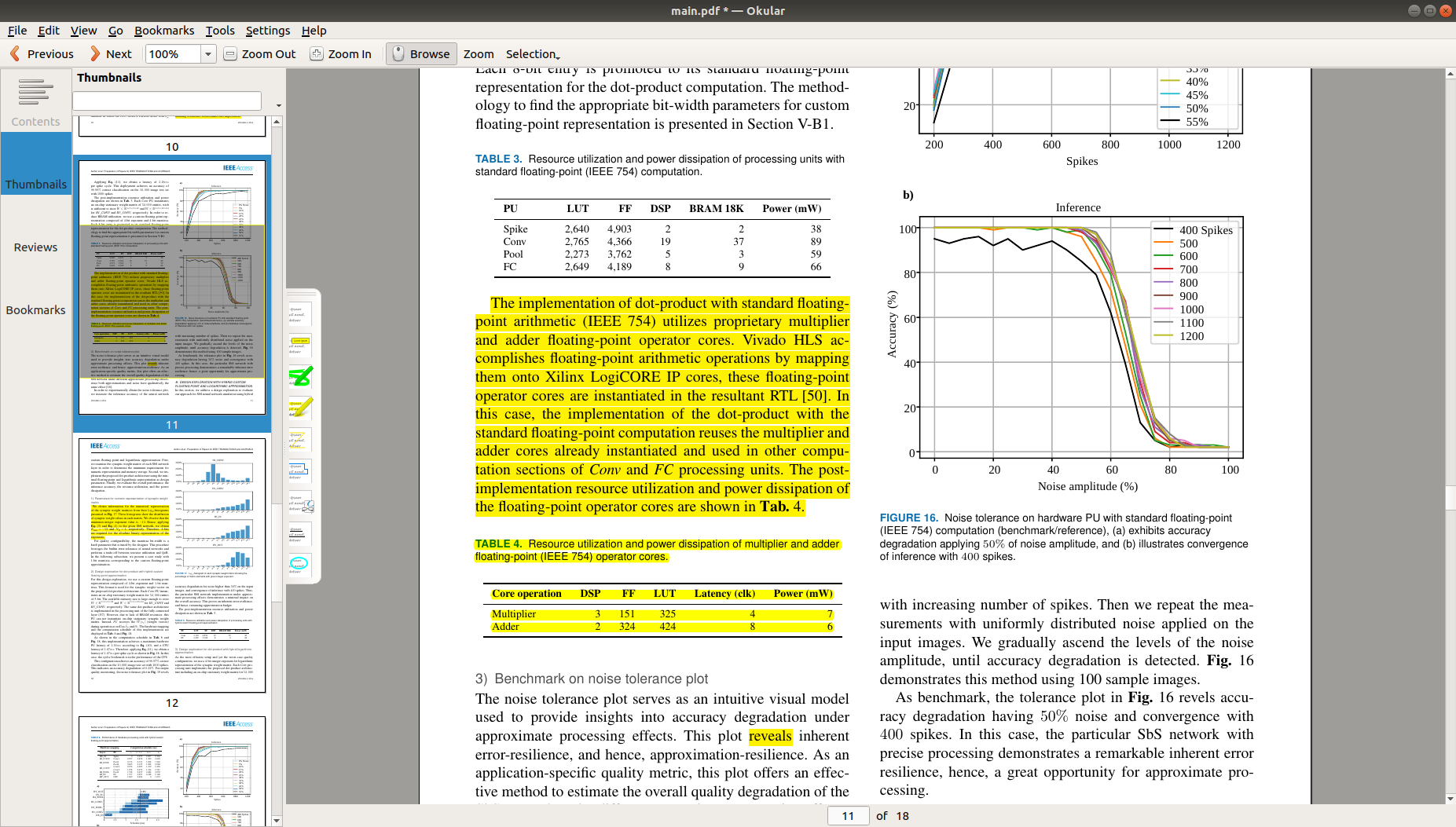
**Author response:**

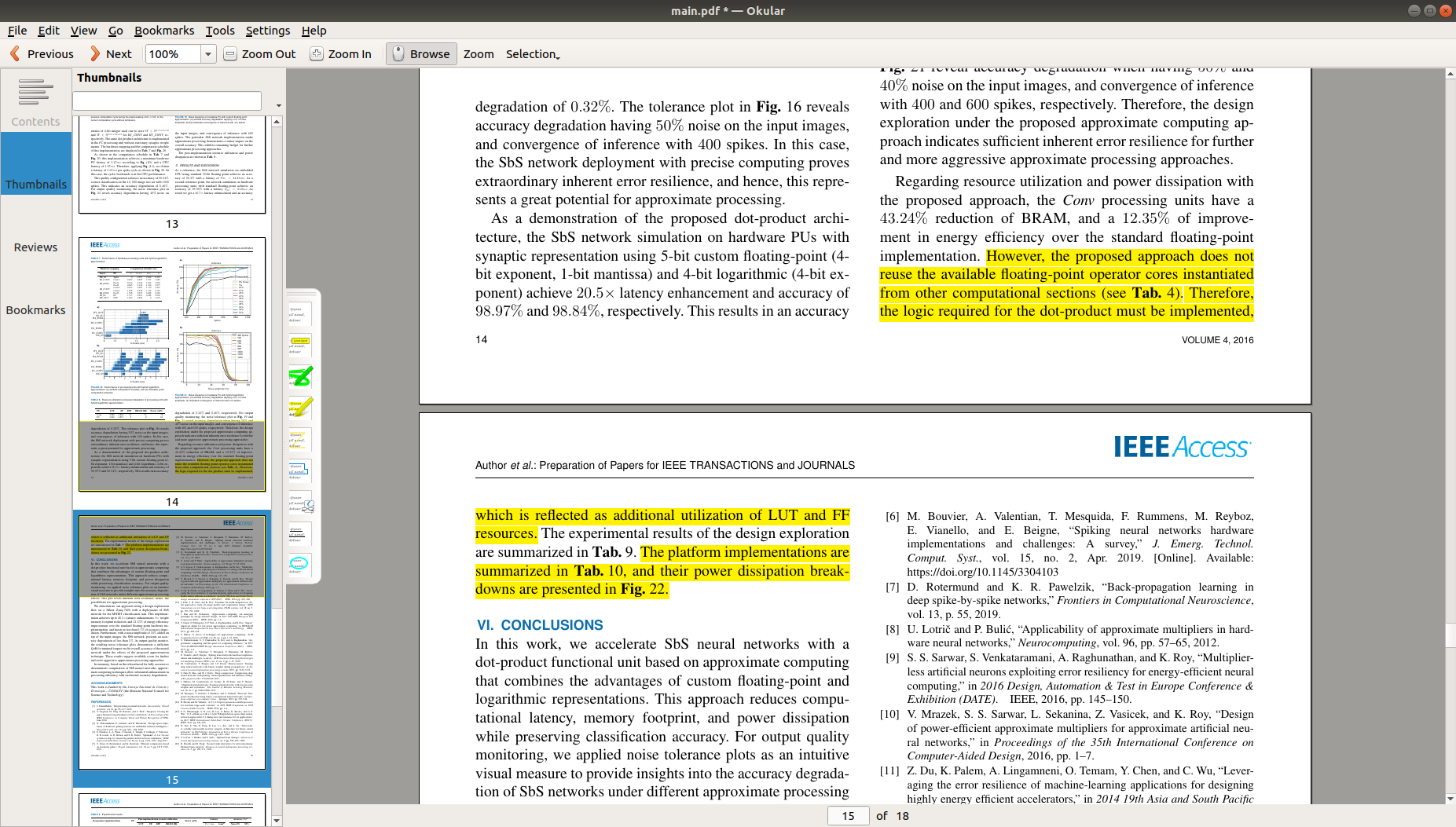
The dot-product with standard floating-point arithmetic (IEEE 754) reuses floating-point operator cores (LogiCORE IPs) already implemented in other computational sections of the hardware kernel (Conv and FC). However, the proposed architecture does not reuse the already instantiated LogiCORE IPs. Instead,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

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.

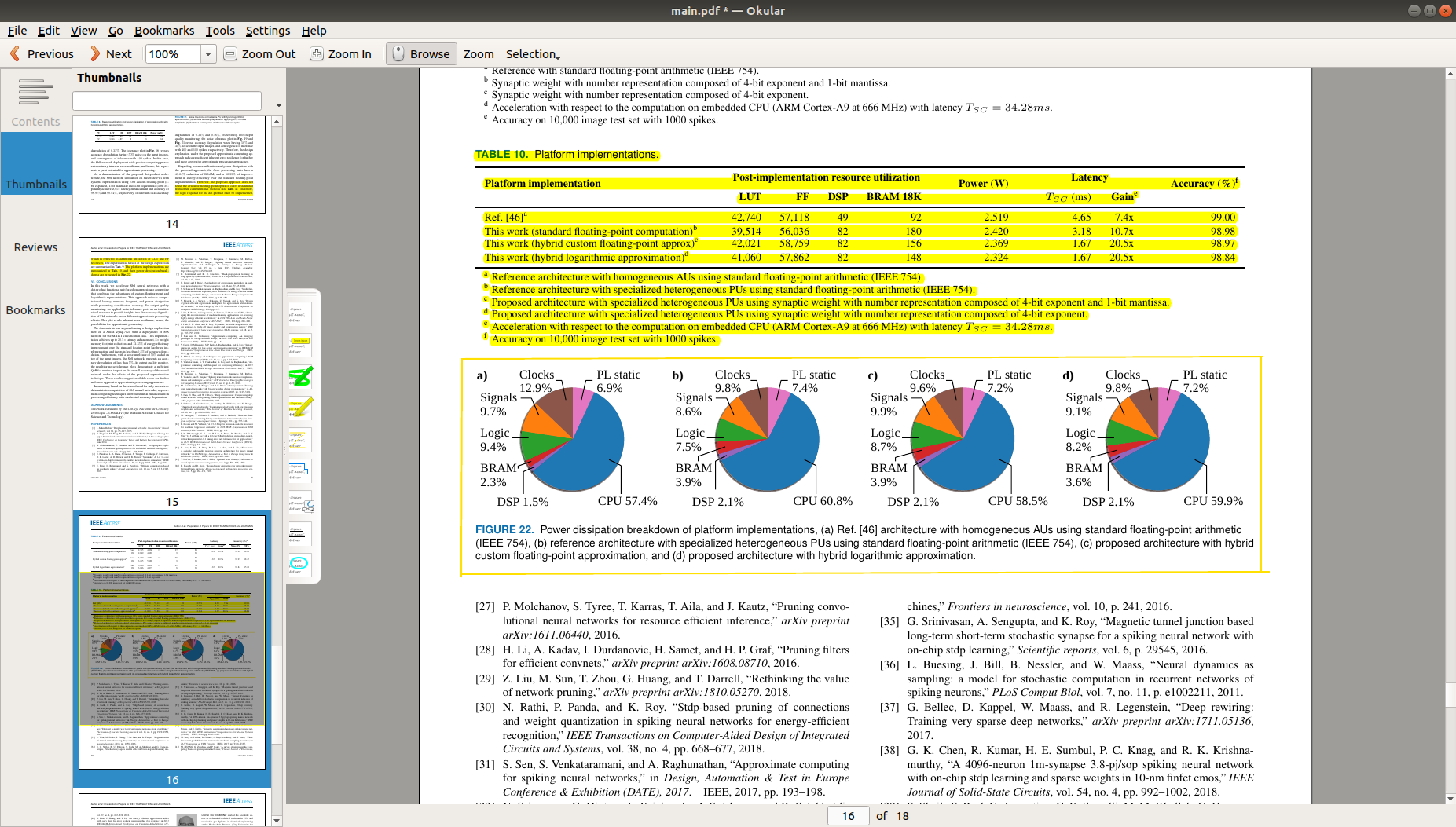
**Author action:**

* We updated the manuscript by adding the discussion regarding the floating-point operator cores used in the PUs. We add 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 seccion with this topic. We support our statements with a Xilinx application note (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. We added the breakdown of the power consumption for each platform architecture implemented: (1) our previous publication Ref. [46], (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:** Thanks for this observation.

**Author action:** We updated the manuscript by correcting the mistake.

**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:**

**Author action:** We updated the manuscript by ….

**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:**

**Author action:** We updated the manuscript by ….

***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*