Sub-100ns On-Probe Spike Sorting via Resource-Constrained FPGA

Aarav Gaur
Merrill F. West High School
aaravgaurx@gmail.com

Abstract

On-probe spike sorting aims to discriminate between neurons using electrical signals recorded directly at the probe, enabling real-time applications such as brain-computer interfaces. To generate ground-truth training data, we follow SpikeForest/MEArec-style simulation methodology to construct a synthetic multi-neuron tetrode recording with physiologically-motivated waveforms, controlled SNR, and realistic spike overlap. Unlike SpikeForest's currently-available real paired recordings, which provide ground truth for only a single neuron per recording, our synthetic approach provides ground truth for all 8 simulated neurons simultaneously, enabling supervised multi-class discrimination training, while also avoiding a compiled NEURON binary as a dependency—at the cost of using parametric rather than biophysically-simulated waveforms. We train a compact MLP classifier on this dataset and deploy it via hls4ml, using quantization-aware HLS to target FPGA hardware. The resulting classification stage achieves a macro-averaged F1 of 0.83 across all classes (8 neurons + noise). After quantization using hls4ml, the model keeps 98.46% exact agreement with the unquantized classifier—achieving inference latency of 0.095 μs—all while remaining within the resource budget of a small 220 DSP Block FPGA (Xilinx Zynq-7020), demonstrating the possible computational feasibility of real-time, multi-neuron, on-probe spike sorting.

1 Introduction and related works

Spike sorting is defined as attributing extracellular action potentials to its source neuron (1). Tetrodes are used to record spikes from 4 channels (2). This in general causes overlapping spikes coming from different neurons, which we explicitly model in this paper (3).

Currently, SOTA spike sorters (KiloSort, MountainSort, and IronClust) run offline rather than on closed-loop applications (4). This limits their usage in real-time applications, creating a critical need for online spike sorting methods optimized for closed-loop brain-computer interfaces (BCIs).

The main contribution is demonstrating future brain-computer interface feasibility by implementing accurate spike sorting on a Xilinx Zynq-7020 – a small and cheap ($\sim$ \$200) FPGA, containing only 220 DSP blocks – at low latency. This was done using hls4ml (5) and Vitis HLS. To our knowledge, this is the first demonstration of sub-100ns multi-neuron spike sorting on a resource-constrained FPGA, as prior FPGA spike sorting systems report classification latencies in the sub-millisecond range (6).

2 Methods

Code is located at https://github.com/AaravGaurdev/sub100ns-spike-sorting-fpga.

Submitted to 7th Fast Machine Learning for Science Conference (FastML 2026). Do not distribute.

2.1 Data generation

We generate 1000 seconds of recording sampled at 30 kHz with 8 neurons. Each neuron, n, is assigned waveform parameters amplitude (A), width factor (w), asymmetry factor (α), and channel weighting (4 channels) chosen to produce diverse, distinguishable templates across the population (see Table 1). The resulting per-neuron SNR range (8.7–21.4 dB) falls within the difficulty range used in SpikeForest's synthetic tetrode study sets (4; 7).

Table 1: Neuron template parameters

n	**Amp.** μV (A)	**Width** w	**Asymmetry** α	**Channel Weights** ($\gamma_0, \gamma_1, \gamma_2, \gamma_3$)	**Description**
0	80	1.0	1.2	$[1.0, 0.5, 0.2, 0.1]$	Strong on γ_0
1	50	1.3	0.8	$[0.1, 1.0, 0.6, 0.2]$	Strong on γ_1
2	120	0.8	1.5	$[0.3, 0.4, 1.0, 0.4]$	Strong on γ_2, large
3	35	1.5	1.0	$[0.2, 0.1, 0.3, 1.0]$	Strong on γ_3, small
4	60	1.1	0.9	$[0.7, 0.7, 0.1, 0.3]$	Spreads $\gamma_0 + \gamma_1$
5	90	0.9	1.3	$[0.2, 0.5, 0.5, 0.8]$	Spreads γ_2 + γ_3
6	45	1.4	1.1	$[0.5, 0.2, 0.7, 0.5]$	Mixed profile
7	70	1.0	1.0	$[0.4, 0.8, 0.3, 0.6]$	Mixed profile

An extracellular action potential waveform of a particular neuron is modeled as a function of time parameter. The wave is active in $t \in [0, 2\pi]$, and zero otherwise. A waveform is defined as a mixture of 2 gaussian curves as $W_n(t) = \text{trough}_n(t) + \text{peak}_n(t)$ (7; 8; 9; 10). We derive A, w, and α from neuron n.

$$W_n(t) = \begin{cases} -A \exp\left(-\frac{1}{2}\left(\frac{t-1.8}{0.5w}\right)^2\right) + 0.4A\alpha \exp\left(-\frac{1}{2}\left(\frac{t-3.5}{0.8w}\right)^2\right) & t \in [0, 2\pi] \\ 0 & \text{otherwise} \end{cases}$$

Then templates are generated per neuron as a function of tetrode channel (c) and time (t) for each channel. Let $c \in \{0, 1, 2, 3\}$ and $t \in [0, 2\pi]$. We derive γ_c from neuron n.

$$W_n(c, t) = \gamma_c \cdot W_n(t)$$

Neuron firing is modeled as a Poisson process with a refractory period of 2ms. Neurons $n = 0, ..., 7$ fire randomly while maintaining average firing rates of 5, 8, 3, 12, 7, 15, 4, 10 in Hz respectively in order to maintain difficulty seen in literature.

Formally, for each neuron n, inter-spike intervals between spike $j-1$ and j are drawn as:

$$\Delta t_{n,j} \sim \text{Exponential}(\lambda_n), \quad \text{where } \lambda_n = \text{firing rate of neuron } n$$

A refractory period is enforced as:

$$\Delta t_{n,j} \leftarrow \max(\Delta t_{n,j}, \tau_{\text{ref}}), \quad \tau_{\text{ref}} = 2\,\text{ms}.$$

Spike times are then constructed via cumulative summation:

$$t_{n,j} = \sum_{i=1}^{j} \Delta t_{n,i},$$

and converted to discrete sample indices where $s_{n,j}$ denotes the discrete spike time (sample index) corresponding to $t_{n,j}$.

$$s_{n,j} = \lfloor f_s\, t_{n,j} \rfloor \text{, where } f_s = 30\,\text{kHz is the sampling rate.}$$

Now we construct the recording, $r_c(t)$, for each tetrode channel ($c \in \{0, 1, 2, 3\}$). We add bandpassed Gaussian noise $\eta_c(t)$ (300-3000 Hz).

$$r_c(t) = \left(\sum_{n=0}^{7} \sum_{j} W_n(c, t - s_{n,j}) \right) \; + \; \eta_c(t)$$

We end with a Spatio-temporal spike (ψ) overlap of 10.3%. This is calculated as $\psi = \frac{\text{Cross-Neuron overlapping spike windows of 2ms}}{\text{total spikes}}$.

Table 2: Neuron data generation metrics with summary statistics

Neuron ID	Peak Amplitude (µV)	SNR (dB)	Spike Count
0	75.9	17.6	5,055
1	45.9	13.2	7,921
2	117.8	21.4	3,138
3	30.1	9.6	12,056
4	39.6	12.0	6,955
5	69.6	16.9	15,050
6	27.4	8.7	4,026
7	53.6	14.6	9,956
Average	57.5	14.2	8,019.6
Std. Dev.	29.9	4.3	4,126.2

2.2 Spike detection and labeling

Now we simulate blind (without ground truth) detection of spikes.

The noise floor of a channel, ϕ_c is estimated to be median of $|r_c(t)|$ divided by 0.6745. We detect a spike when $|r_c(t)| > 4\phi_c$ and there hasn't been a spike detected within 1ms before this sample.

For each blindly found spike, we sample 30 samples before and after the spike (totaling 60 samples). After this, we label each spike by temporal proximity to the nearest spike from ground truth (in rare cases, $\sim 0.8\%$ of detections, more than one neuron's spike falls within the match window; we assign the globally nearest one regardless of neuron identity). A detected spike is classified as noise if the nearest ground-truth spike is further than 0.5 ms (15 samples) away.

After this, we perform an 80/20 train-test split. Then we flatten each spike to be structured from $(4, 60) \rightarrow (240)$. Finally we apply PCA to ensure that each spike is represented by 16 dimensions (the input layer size of the MLP).

In our final dataset we have 78% Neurons matched to ground truth (rest set to noise group). Our average Signal to Noise ratio is 14.2 dB ($\sigma = 4.3$ dB), with Neuron 6 (8.7 dB) and Neuron 3 (9.6 dB) lowest.

2.3 Training and quantization

PyTorch was used to train the MLP model. We apply inverse-frequency class weighting to the cross-entropy loss. Quantization agreement was verified by comparing argmax predictions between the float32 PyTorch model and the HLS C-simulation on the test set. The learning rate decayed by a factor of 0.5 every 100 epochs.

Table 3: Configuration parameters for the MLP Classifier and hls4ml (5) synthesis

Framework / Domain	Configuration Parameter	Value / Setting
MLP Architecture	Input Layer Size	16 (PCA components)
	Hidden Layer 1	16 units (ReLU)
	Hidden Layer 2	12 units (ReLU)
	Output Layer Size	9 units (8 neurons + 1 noise)
	Loss Function	Categorical Cross-Entropy
	Optimizer	Adam ($\beta_1 = 0.9, \beta_2 = 0.999$)
	Initial Learning Rate	3×10^{-3}
	Batch Size	64
	Epochs	500
hls4ml Settings	Target FPGA Board	Xilinx Zynq 7020
	Clock Period	5.0 ns (200 MHz)
	IO Resolution (‘IOType‘)	‘io_parallel‘
	Reuse Factor	3
	Default Precision (‘Precision‘)	‘ap_fixed<16,9>‘
	Activation Implementation	Table-based (‘lut‘)
	Strategy	Latency / Resource Optimized

3 Results and discussion

Our model itself achieves an accuracy of 85.29% with a 0.83 macro F1, comparable to the per-unit accuracy range reported for leading sorters (MountainSort4, KiloSort2, IronClust) on synthetic 4-channel tetrode recordings of similar SNR difficulty (4). This is largely being weighed down by the high-difficulty classes (i.e. neuron 3, 6 which have low SNR; noise which is structurally ambiguous).

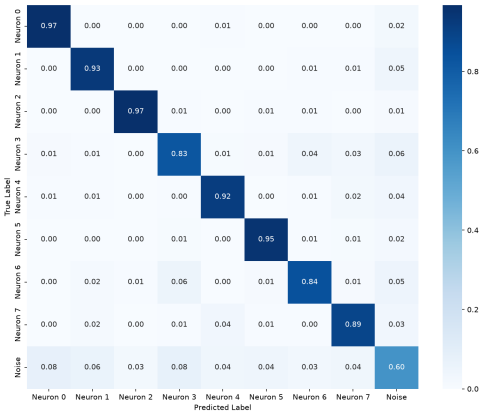

Figure 1: Confusion matrix of PyTorch float32 classifier

Table 4: Vitis HLS Synthesis and quantization performance summary

Evaluation Domain	Metric / Resource Element	Synthesized Value
Timing & Latency	Target Clock Period	5.00 ns
	Estimated Clock Period	4.362 ns
	Clock Uncertainty	0.62 ns
	Latency	**95.000 ns** (19 cycles)
	Initiation Interval (II)	3 cycles
Resource Utilization	DSP Blocks	186 / 220 (84%)
	Flip-Flops (FF)	26,122 / 106,400 (24%)
	Look-Up Tables (LUT)	19,938 / 53,200 (37%)
	Block RAM (BRAM_18K)	0 / 280 (0%)
	UltraRAM (URAM)	0 / 0 (0%)
Quantization Verification	PyTorch vs. HLS Bit Agreement	98.46%
	Bit-Accuracy Deviations	157 / 10,207 samples

Our deployment achieves 95 ns latency (classification stage only; see limitations) while remaining within the resource budget of this low-cost FPGA, with 98.46% exact agreement between the quantized hardware implementation and the original float32 model. This required a Reuse Factor of 3, as lower Reuse Factors exceeded DSP budget on this device. However on a larger FPGA, a lower reuse factor (and correspondingly lower latency) would likely be achievable, given the additional DSP headroom; we instead target this constrained, low-cost device to better reflect the cost and power budget realistic for on-probe BCI deployment.

3.1 Limitations

Our results are based on synthetic data using parametric (rather than biophysically-simulated) waveforms and a single random seed (42). While the classification stage meets the resource budget of a low-cost FPGA, on-chip synthesis of the upstream PCA feature-extraction stage exceeded this budget even after precision and reuse-factor tuning; reported latency and resource figures therefore apply to the classification stage alone.

References

[1] Rey, H. G., Pedreira, C., & Quian Quiroga, R. (2015). Past, present and future of spike sorting techniques. Brain research bulletin, 119(Pt B), 106–117. https://doi.org/10.1016/j.brainresbull.2015.04.007

[2] Wilson, M., & McNaughton, B. (1993). Dynamics of the hippocampal ensemble code for space. Science, 261(5124), 1055–1058. https://doi.org/10.1126/science.8351520

[3] Kim, M. K., Kim, S. P., & Sohn, J. W. (2024). Synthetic data-driven overlapped neural spikes sorting: decomposing hidden spikes from overlapping spikes. Molecular brain, 17(1), 89. https://doi.org/10.1186/s13041-024-01161-y

[4] Magland, J., Jun, J. J., Lovero, E., Morley, A. J., Hurwitz, C. L., Buccino, A. P., Garcia, S., & Barnett, A. H. (2020). SpikeForest, reproducible web-facing ground-truth validation of automated neural spike sorters. eLife, 9, e55167. https://doi.org/10.7554/eLife.55167

[5] Schulte, J.-F., Ramhorst, B., Sun, C., Mitrevski, J., Ghielmetti, N., Lupi, E., … Hauck, S. (2026). hls4ml: A Flexible, Open Source Platform for Deep Learning Acceleration on Reconfigurable Hardware. ACM Transactions on Reconfigurable Technology and Systems, 19(2), 1–35. https://doi.org/10.1145/3801979

[6] Wang, P. K., Pun, S. H., Chen, C. H., McCullagh, E. A., Klug, A., Li, A., … Lei, T. C. (2019). Low-latency single channel real-time neural spike sorting system based on template matching. PLOS ONE, 14(11), e0225138. https://doi.org/10.1371/journal.pone.0225138

[7] Buccino, A. P., & Einevoll, G. T. (2021). MEArec: A Fast and Customizable Testbench Simulator for Ground-truth Extracellular Spiking Activity. Neuroinformatics, 19(1), 185–204. https://doi.org/10.1007/s12021-020-09467-7

[8] Hill, D. N., Mehta, S. B., & Kleinfeld, D. (2011). Quality metrics to accompany spike sorting of extracellular signals. The Journal of neuroscience : the official journal of the Society for Neuroscience, 31(24), 8699–8705. https://doi.org/10.1523/JNEUROSCI.0971-11.2011

[9] Gold, C., Henze, D. A., Koch, C., & Buzsáki, G. (2006). On the origin of the extracellular action potential waveform: A modeling study. Journal of neurophysiology, 95(5), 3113–3128. https://doi.org/10.1152/jn.00979.2005

[10] Souza, B. C., Lopes-dos-Santos, V., Bacelo, J., & Tort, A. B. L. (2019). Spike sorting with Gaussian mixture models. Scientific Reports, 9(1). https://doi.org/10.1038/s41598-019-39986-6