Implementing Gated Recurrent Units On Field-Programmable Gate Array For Brain-Computer Interface Applications

1st Aleksander Berezowski
Schulich School of Engineering
University of Calgary
Calgary, Canada
aleksander.berezowsk@ucalgary.ca

2nd Dr. Eli Kinney-Lang
Schulich School of Engineering
University of Calgary
Calgary, Canada
eli.kinneylang@ucalgary.ca

2nd Dr. Denis Onen
Schulich School of Engineering
University of Calgary
Calgary, Canada
donen@ucalgary.ca

***Abstract*—**
***Index Terms*—**

I. Introduction

BCI systems are predominantly deployed on Central Processing Units (CPUs) and Graphics Processing Units (GPUs) []. While these platforms offer programming flexibility and ease of development, they present significant limitations for practical BCI applications. CPUs execute instructions sequentially with limited parallelism, resulting in higher latency for real-time neural signal processing. GPUs, while providing substantial parallel processing capabilities, consume considerable power, making them unsuitable for portable, wearable, or implantable BCI devices. Both platforms face fundamental trade-offs between computational performance and power efficiency that constrain their viability for BCI systems requiring continuous operation, immediate responsiveness, and extended battery life.

Field-Programmable Gate Arrays (FPGAs) offer a compelling alternative deployment platform that addresses these limitations through fundamentally different architectural characteristics. FPGAs provide reconfigurable hardware that enables flexible deployment and updating of different BCI machine learning models while maintaining the performance benefits of custom hardware implementations. Their architecture supports massive parallelism through spatially distributed logic resources, enabling simultaneous execution of multiple operations that would require sequential processing on CPUs. This parallelism translates directly to superior power efficiency, with FPGA implementations often consuming orders of magnitude less power than equivalent GPU solutions while maintaining comparable or superior throughput. The low-latency characteristics of FPGA implementations are particularly valuable for closed-loop BCI applications where milliseconds of delay can significantly impact system usability and user experience. Cai et al. demonstrated these advantages in practice by implementing an epilepsy detection algorithm on an FPGA, showing that FPGA-based BCI systems achieve higher accuracy than software-based implementations while simultaneously reducing power consumption.

The potential of GRU implementations on FPGAs for BCI applications has begun to be explored, though fundamental questions about their design remain unresolved. Zaghloul et al. presented the first hardware implementation of a GRU in 2021, demonstrating substantial resource efficiency advantages over LSTM implementations. Their GRU hardware implementation used 50% fewer Buffers, 42% fewer Digital Signal Processors (DSPs), 39% less Block RAM (BRAM), and 35% fewer Slice Look-Up Tables (LUTs) compared to an LSTM implementation, while also achieving higher inference performance per Watt and lower execution time. This work established that GRUs are more resource-efficient than LSTMs when implemented in hardware, confirming the computational advantages suggested by algorithmic analysis. Rizwan et al. subsequently refined this implementation in 2025, advancing the hardware architecture. However, while these studies demonstrated that hardware GRU implementations are viable and efficient compared to LSTMs, they did not systematically investigate how specific design parameters within GRU implementations affect the complex trade-offs between resource utilization, timing performance, power consumption, and computational accuracy.

A critical gap remains in understanding the relationships between fundamental design parameters and system performance in FPGA-based GRU implementations. When designing hardware for BCI applications, engineers must make decisions about input dimensionality (corresponding to the number of neural signal channels), hidden state size (determining the network's representational capacity), and numerical precision (affecting both accuracy and hardware cost). Each of these parameters influences multiple performance metrics simultaneously, creating a complex design space with non-obvious trade-offs. For instance, increasing numerical precision improves computational accuracy but may degrade timing performance and increase power consumption. Similarly, larger hidden states enhance model capacity but consume more hardware resources and may impact inference latency. Without

systematic characterization of these relationships, designers must rely on intuition or exhaustive trial-and-error approaches, potentially resulting in suboptimal implementations that either over-provision resources or fail to meet performance requirements.

This research addresses this gap by systematically investigating the question: "How does input size, hidden state size, and word size affect resource usage, inference time, power consumption, and accuracy of brain-computer interface gated recurrent unit machine learning models deployed on field-programmable gate arrays?" The primary objective is to quantitatively characterize how different design parameters affect the trade-offs between competing performance metrics in FPGA-based GRU implementations. By establishing these relationships through comprehensive design space exploration, this work provides actionable guidance for designing optimal FPGA-based GRUs for specific BCI application constraints. Understanding these trade-offs is crucial for determining the best GRU design for scenarios with different priorities—whether minimizing power consumption for wearable devices, maximizing accuracy for clinical applications, or balancing multiple objectives for general-purpose BCI systems.

The hardware implementation architecture investigated in this work builds upon the foundation established by Zaghloul et al. The architecture consists of two primary modules: the gate module and the output module. The gate module implements the reset gate, update gate, and activation functions using two Multiply-and-Accumulate (MAC) units that are summed and passed through sigmoid or tanh activation functions. The output module performs Hadamard products and summation to generate the final hidden state. In modern implementations, GRUs are calculated using the following equations:

By providing the first systematic quantification of how input dimensionality, hidden state size, and numerical precision affect the multidimensional performance space of FPGA-based GRU implementations, this research establishes a foundation for principled hardware design decisions in BCI applications. The insights derived from this characterization enable engineers to navigate the complex trade-offs inherent in hardware neural network implementations, ultimately advancing the development of efficient, accurate, and practical FPGA-based BCI systems suitable for clinical deployment and assistive technology applications.

II. Methodology

A. Experimental Design

The experimental system comprises several interconnected Python modules and a Tcl script that automate the entire design, implementation, and analysis pipeline. This automated framework enables systematic exploration of the GRU design space by programmatically generating hardware descriptions, orchestrating FPGA toolchain execution, and extracting performance metrics.

Python scripts automatically generate SystemVerilog code for the GRU module, top-level wrapper, and a testbench based on specified parameters. A Tcl script orchestrates the Xilinx Vivado tool to perform simulation, synthesis, optimization, placement, and routing while generating detailed reports. Additional Python scripts parse Vivado-generated reports and simulation outputs to extract hardware metrics and calculate the accuracy measurements Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE).

B. Design Space and Variables

Independent Variables

- INT_WIDTH: Number of integer bits in fixed-point representation, determining the range of representable values. The test data ranges from -3.28 to 2.96, requiring a minimum INT_WIDTH of 3 bits based on two's complement representation range of $-2^{\text{INT\_WIDTH}-1}$ to $2^{\text{INT\_WIDTH}-1} - 1$. This study uses INT_WIDTH = 6 to provide sufficient headroom and prevent overflow during intermediate calculations.
- FRAC_WIDTH: Number of fractional bits in fixed-point representation, determining numerical precision. Lower values reduce hardware size but increase quantization error. The tested range is $FRAC\_WIDTH \in \{4, 9, 14, 19, 24\}$, representing a spectrum from coarse precision to floating-point precision [1].
- d (Input Dimension): Input feature dimension corresponding to the number of EEG channels in BCI applications. The dataset used contains 64 channels, thus 64 will be the upper range for d [2]. Özkahraman et al. demonstrated an 83% accuracy using Motor Imagery with just 6 channels, thus 6 will be the lower range for d [3]. The tested range is $d \in \{4, 8, 16, 32, 64\}$.
- h (Hidden State Size): Hidden state dimension, a hyperparameter typically determined during GRU training. During GRU training it was found that the lower range for h was 4, and the upper range was 16. The tested range is $h \in \{4, 6, 8, 12, 16\}$.

Dependent Variables

- Resource Utilization Metrics: Quantity of Lookup Tables (LUTs), flip-flops (registers), Block RAMs (BRAMs), and DSPs (Digital Signal Processors) used.These metrics are extracted from detailed reports output by Vivado.
- Timing Metrics: Worst Negative Slack (WNS) measures timing margin relative to the 100 MHz clock constraint (10ns period).
- Power Metrics: Total power (W) represents complete FPGA power consumption. Dynamic power (W) measures power from switching activity. Static power (W) quantifies leakage current. All power metrics are extracted from Vivado's power analysis reports.
- Accuracy Metrics: MAE and RMSE quantifies prediction accuracy relative to the ground truth. MAE provides a robust error metric less sensitive to outliers than RMSE, however RMSE does a better job quantifying sensitivity to outliers [4].

Controlled Variables

- Target FPGA: Fixed to a Xilinx Artix-7, ensuring consistent LUT architecture, slice organization, and resource availability across all trials, as this can affect the outputted design [5].
- Clock Frequency: Fixed to 100 MHz, providing a consistent performance target.
- Vivado Version: Fixed to 2024.1, as CAD tool versions can affect synthesis and implementation results [5].
- Synthesis Settings: Identical optimization flags and strategies applied across all designs.
- Test Vectors: The same data is used for each testbench, providing 100 reproducible test cases without random variation.
- Weight Values: Identical pre-initialized weight matrices used across all configurations, ensuring that performance differences result from architectural parameters rather than weight variation.
- Generation Scripts: SystemVerilog generation logic remains constant across all parameter combinations, varying only the parametric inputs.

C. Trial Execution Flow

Each experimental trial follows a structured sequence to ensure consistency:

1) RTL Generation: An untested combination of INT_WIDTH, FRAC_WIDTH, d, and h is selected from the design space and used as input parameters for RTL generation. Three SystemVerilog modules are automatically generated: the GRU module implementing Equations (**??**)–(**??**), a wrapper module that instantiates the GRU with synthesis preservation directives to prevent logic optimization, and a testbench with 100 deterministic test vectors using a BCI motor imagery dataset for reproducible accuracy evaluation.
2) Vivado Execution: A Vivado project is created and all generated SystemVerilog files are added to it. Synthesis is executed with resource and timing optimizations, placement and routing is performed, and detailed reports on resource utilization, timing analysis, and power consumption are generated. Finally, a simulation is run using the generated testbench to produce a simulated output.
3) Data Capture: The detailed reports are parsed to extract hardware metrics including LUTs, registers, BRAMs, DSPs, WNS, and power consumption.
4) Accuracy Calculation: Each simulated output is compared to a ground truth calculated using floating point numbers using MAE and MSE to determine GRU implementation accuracy.

This structured flow ensures that each trial is executed identically and carryover effects are eliminated through complete regeneration of all files.

III. Results and Analysis

GRU implementations were synthesized for the XCU250-FIGD2104-2L-E FPGA. This particular FPGA was selected solely to ensure sufficient resources for all design variations under investigation, thereby preventing resource constraints from limiting the scope of experimental results. The choice of this specific device does not constrain the applicability of findings; the architectural optimizations and performance characteristics demonstrated herein are device-agnostic and can be applied to any FPGA platform. In practical deployments, the target FPGA should be selected based on the resource requirements of the optimized design rather than constraining the design to fit a predetermined device.

A. Design Space Exploration Results

Of the 125 design parameter sets, 11 could not be generated due to insufficient resource availability on the FPGA. This limitation could not be addressed by selecting a larger FPGA, as the largest device available in Xilinx Vivado had already been employed. The failed design parameter sets are presented in Table 1.

d	h	INT_WIDTH	FRAC_WIDTH
64	16	6	24
64	16	6	19
32	16	6	24
32	16	6	19
16	16	6	24
64	12	6	24
64	12	6	19
32	12	6	24
8	12	6	24
64	8	6	24
64	6	6	24

Fig. 1. Table illustrating failed design parameter sets.

B. Parameter Correlation Analysis

The correlation matrix presented in Figure 2 reveals several significant relationships between design parameters and performance metrics, providing insights into the fundamental trade-offs inherent in FPGA-based GRU implementations.

1) Resource Utilization Patterns: The hidden state size (h) exhibits the strongest positive correlation with hardware resource consumption, demonstrating correlations of 0.41 with LUTs, 0.50 with registers, and 0.27 with DSPs. This relationship reflects the quadratic scaling of hidden-to-hidden weight matrices ($W_{hr}, W_{hz}, W_{hn} \in \mathbb{R}^{h \times h}$) with respect to h, as defined in Equations (**??**)–(**??**). The input dimension (d) demonstrates a moderate positive correlation with LUT utilization (0.36) and DSP usage (0.18), attributable to the linear scaling of input-to-hidden weight matrices ($W_{ir}, W_{iz}, W_{in} \in \mathbb{R}^{h \times d}$).

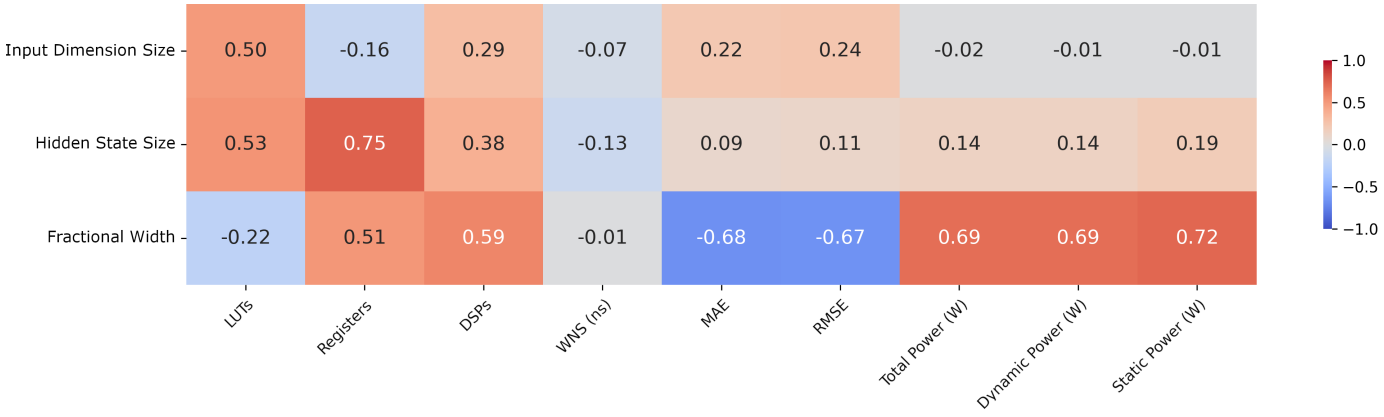

Fig. 2. Correlation matrix illustrating the relationships between independent variables and dependent variables.

Conversely, FRAC_WIDTH shows a weak negative correlation with LUT consumption (-0.29), suggesting that increased precision does not proportionally increase combinational logic requirements, likely due to the fixed-width arithmetic units in the implementation.

2) Timing Performance: WNS demonstrates an inverse relationships with architectural complexity. The hidden state size exhibits a negative correlation with WNS (-0.27), indicating that larger hidden states impose greater timing constraints due to increased computational depth and interconnect complexity. Similarly, the input dimension shows a negative correlation with WNS (-0.30), reflecting the additional logic delays introduced by wider input vectors. FRAC_WIDTH presents a more substantial negative correlation with WNS (-0.38), demonstrating that higher precision arithmetic operations introduce longer critical paths through the combinational logic.

3) Accuracy Characteristics: FRAC_WIDTH exhibits the strongest influence on implementation accuracy, with correlation coefficients of -0.70 for MAE and -0.69 for RMSE. These strong negative correlations confirm that increased fractional precision directly reduces quantization error, as expected from fixed-point number theory. The architectural parameters h and d show negligible correlations with accuracy metrics (MAE: 0.02 and 0.13; RMSE: 0.03 and 0.14, respectively), indicating that network topology does not significantly affect numerical precision in hardware implementations when precision is held constant.

4) Power Consumption: Total power consumption demonstrates modest negative correlations with d (-0.30), h (-0.25), and FRAC_WIDTH (-0.32), contrary to the expected positive relationship between circuit complexity and power dissipation. This counterintuitive finding warrants further investigation and may result from Vivado's power estimation methodology or interactions between resource utilization and routing efficiency. Dynamic power exhibits a strong positive correlation with FRAC_WIDTH (0.56), suggesting that higher precision arithmetic operations generate increased switching activity, consistent with the larger number of bits transitioning during computations. The architectural parameters d and h show weak negative correlations with dynamic power (-0.07 and 0.08, respectively), indicating minimal impact of network topology on switching activity. Static power demonstrates negative correlations with all design parameters (d: -0.30, h: -0.27, FRAC_WIDTH: -0.38), mirroring the pattern observed in total power consumption. This relationship suggests that static power may be influenced by factors beyond simple resource count, potentially including variations in routing congestion, placement density, or device-level power optimization strategies employed by the synthesis toolchain. The correlation patterns between static and total power are nearly identical, indicating that static power constitutes a significant fraction of total power dissipation in these implementations.

5) Design Trade-off Implications: The correlation analysis reveals a fundamental design trade-off between accuracy and hardware efficiency. FRAC_WIDTH simultaneously improves accuracy (strong negative correlations with MAE and RMSE) while degrading timing performance (negative correlation with WNS) and increasing dynamic power consumption (positive correlation of 0.56). The hidden state size primarily affects resource utilization with minimal impact on accuracy, suggesting that h can be optimized for computational requirements without significantly compromising numerical precision. The relatively weak correlations between structural parameters (d, h) and timing metrics indicate that the implemented architecture maintains acceptable timing margins across the explored design space, with FRAC_WIDTH serving as the primary determinant of critical path delay.

IV. Discussion

Figure 3 shows a description of the output variables, which is used in this discussion.

A. Static Power Dominance and Idle Operation

The descriptive statistics reveal that static power consumption (mean: 2.95 W, standard deviation: 0.00 W) constitutes the dominant component of total power dissipation (mean: 3.00 W, standard deviation: 0.09 W), while dynamic power contributes minimally (mean: 0.05 W, standard deviation: 0.09 W). This distribution directly correlates with the observed timing characteristics, where the mean WNS of 9.29

	LUTs	Registers	BRAMs	DSPs	WNS (ns)	Total Power (W)	Dynamic Power (W)	Static Power (W)	MAE	RMSE
mean	65294.98	332.63	0.00	1072.84	9.29	3.00	0.05	2.95	0.02	0.03
std	70919.31	194.77	0.00	1143.91	0.15	0.09	0.09	0.00	0.03	0.04
min	3916.00	80.00	0.00	0.00	8.21	2.95	0.00	2.94	0.01	0.01
25%	20467.25	180.00	0.00	166.50	9.24	2.95	0.00	2.94	0.01	0.01
50%	41605.00	300.00	0.00	729.00	9.31	2.96	0.01	2.94	0.01	0.01
75%	80159.75	480.00	0.00	1620.00	9.38	3.00	0.06	2.94	0.01	0.01
max	418852.00	960.00	0.00	4800.00	9.49	3.38	0.42	2.95	0.10	0.15

Fig. 3. Description of the dependent variables.

ns indicates substantial timing margin relative to the 10 ns clock period constraint. The critical path delay averages only 0.71 ns (7% of the clock period), meaning the FPGA logic remains idle for approximately 93% of each clock cycle, with logic transitions occurring only during the brief computation window. This idle time explains the negligible dynamic power consumption, as switching activity, the primary driver of dynamic power dissipation, occurs infrequently. The near-constant static power across all design configurations (range: 2.94-2.95 W) reflects continuous leakage current independent of computational activity, confirming that power optimization in these implementations must primarily address static rather than dynamic power consumption.

B. Temporal Multiplexing and Bit-Serial Optimization Opportunities

The vast disparity between FPGA clock frequencies (100 MHz in this study) and typical BCI sampling rates (on the order of Hz) [6] presents a substantial optimization opportunity through temporal multiplexing across multiple clock cycles. The FPGA executes approximately one million clock cycles between successive BCI data samples, yet the current implementations complete their computations within a single clock cycle and remain idle while awaiting new input data. The exceptionally short critical path delays (mean: 0.71 ns, maximum: 1.79 ns) represent only 7% to 18% of the available 10 ns clock period, quantifying the underutilization within each clock cycle. However, the more significant inefficiency lies in the extended idle periods between BCI samples, during which the hardware remains powered but performs no useful computation. By distributing GRU computations across clock cycles through time-multiplexed architectures, hardware resource requirements could be dramatically reduced without compromising real-time processing requirements. One particularly effective implementation of temporal multiplexing is through bit-serial architectures, which compute results sequentially one bit position at a time rather than processing all bits in parallel, trading temporal resources for spatial efficiency [7], [8]. While parallel addition of two 32-bit integers requires 32 adders operating in a single clock cycle, an equivalent bit-serial implementation utilizes only one adder over 32 cycles. This architectural approach could significantly reduce the LUT, register, and DSP resource requirements observed in the current implementations. The observed timing margins (mean WNS: 9.29 ns) indicate that extending individual arithmetic operations across multiple clock cycles would not violate timing constraints. Given the million-fold difference between FPGA clock rates and BCI sampling rates, even fully bit-serial implementations of all arithmetic operations would complete well within the inter-sample interval, making this aggressive hardware minimization strategy viable for BCI applications while fundamentally shifting the design paradigm from spatial parallelism to temporal efficiency. This and other temporal reuse strategies could achieve order-of-magnitude reductions in hardware consumption while maintaining sufficient throughput to process BCI signals.

C. Diminishing Returns of Fractional Precision

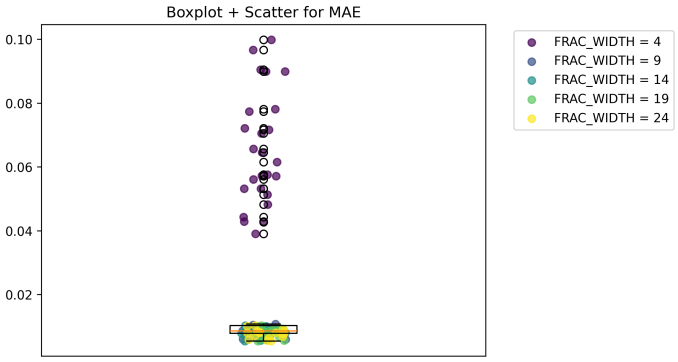

Fig. 4. Boxplot and scatter visualization for MAE.

The boxplot and scatter visualizations for MAE (Figure 4) and RMSE (Figure 5) reveal a critical threshold effect in the relationship between fractional precision and accuracy. While FRAC_WIDTH = 4 exhibits substantially degraded accuracy with high error dispersion (MAE range: approximately 0.04-0.10, RMSE range: approximately 0.06-0.15), the accuracy improvements diminish rapidly beyond moderate precision levels. The implementations with FRAC_WIDTH $\in \{9, 14, 19, 24\}$ form tightly clustered distributions in the

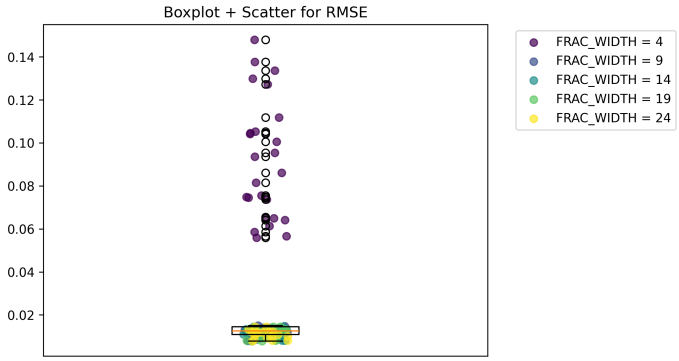

Fig. 5. Boxplot and scatter visualization for RMSE.

lower portion of both plots, with both boxplots indicating statistically similar performance. The marginal reduction in error between FRAC_WIDTH = 14 and FRAC_WIDTH = 24 is negligible relative to the substantial hardware and timing costs associated with higher precision arithmetic, as evidenced by the correlation analysis showing FRAC_WIDTH's strong negative impact on WNS (-0.38) and positive impact on dynamic power (0.56). This plateau effect suggests that FRAC_WIDTH values beyond approximately 9-14 bits provide minimal accuracy benefits while incurring disproportionate implementation costs, establishing an optimal precision threshold for resource-constrained FPGA deployments. The clustering pattern indicates that practitioners can confidently select moderate precision configurations (FRAC_WIDTH $\approx$ 9-14) to achieve near-optimal accuracy while minimizing hardware overhead, critical path delays, and power consumption.

V. Conclusion and Future Work

This research systematically investigated the fundamental design trade-offs in FPGA-based GRU implementations for BCI applications, addressing the critical question of how input size, hidden state size, and numerical precision affect resource utilization, inference time, power consumption, and accuracy. Through comprehensive exploration of 114 successfully synthesized design configurations, this work provides the first quantitative characterization of the complex relationships between architectural parameters and performance metrics in hardware-implemented GRUs for neural signal processing.

The primary contributions of this work includes establishment of a reproducible, automated framework for systematic GRU hardware design space exploration, quantitative analysis revealing that fractional precision (FRAC_WIDTH) serves as the dominant factor affecting both accuracy and timing performance, while hidden state size (h) primarily influences resource consumption, and identification of critical optimization opportunities through temporal multiplexing that could achieve order-of-magnitude hardware reductions without compromising BCI processing requirements. The correlation analysis revealed strong negative relationships between FRAC_WIDTH and accuracy metrics (MAE: -0.70, RMSE: -0.69), while demonstrating the accuracy plateau beyond moderate precision levels (FRAC_WIDTH $\approx$ 9-14 bits), establishing practical guidelines for precision selection in resource-constrained deployments.

A key finding with profound implications for future implementations is the identification of massive underutilization stemming from the temporal mismatch between FPGA clock frequencies (100 MHz) and BCI sampling rates (typically on the order of Hz). The observed timing margins (mean WNS: 9.29 ns, representing 93% idle time per clock cycle) combined with the million-fold difference between processing and sampling rates reveals that current implementations complete computations within microseconds yet remain idle for milliseconds awaiting new samples. This fundamental inefficiency presents an extraordinary optimization opportunity: temporal multiplexing strategies, particularly bit-serial architectures, could dramatically reduce hardware resource requirements while maintaining sufficient throughput for real-time BCI processing. Such temporal reuse approaches could fundamentally shift the design paradigm from spatial parallelism to temporal efficiency, potentially enabling GRU implementations on substantially smaller, lower-cost FPGAs suitable for wearable and implantable BCI devices.

Future theoretical work should investigate hardware-specific arithmetic optimizations to further reduce resource consumption. Shift-and-add multiplication techniques, which replace costly multipliers with sequences of shift and addition operations, could significantly reduce DSP and LUT utilization, particularly given the timing margins available for distributing operations across multiple clock cycles. Exploration of approximate computing techniques, custom activation function implementations, and coefficient quantization strategies could yield additional resource savings. Most critically, detailed investigation of bit-serial and time-multiplexed architectures should quantify the achievable hardware reductions and their impact on power consumption, potentially demonstrating feasibility of ultra-low-resource implementations that process operations sequentially rather than in parallel.

On the practical side, validation of FPGA-based BCI GRU implementations with real human subjects represents essential future work that has not previously been demonstrated. While this study utilized a motor imagery dataset for accuracy evaluation, real-time deployment with live neural signal acquisition would validate the complete BCI pipeline including signal preprocessing, real-time inference, and closed-loop device control. Such validation would assess system robustness to signal variability, noise characteristics, and artifacts inherent in real-world BCI applications, while evaluating user experience metrics including system responsiveness, control accuracy, and usability. Successful human validation would establish the clinical and practical viability of FPGA-based GRU systems for assistive technology applications, potentially demonstrating superior performance compared to conventional CPU and GPU implementations in terms of latency, power efficiency, and portability.

By providing quantitative characterization of design trade-offs and identifying critical optimization pathways, this

work establishes a foundation for developing next-generation FPGA-based BCI systems that balance computational accuracy, hardware efficiency, and power consumption to enable practical deployment in clinical and assistive technology applications.

VI. References

[1] D. Goldberg, "What every computer scientist should know about floating-point arithmetic," *ACM Computing Surveys*, vol. 23, no. 1, pp. 5–48, Mar 1991.

[2] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "Physiobank, physiotoolkit, and physionet," *Circulation*, vol. 101, no. 23, Jun 2000.

[3] A. Özkahraman, T. Ölmez, and Z. Dokur, "Performance improvement with reduced number of channels in motor imagery bci system," *Sensors*, vol. 25, no. 1, p. 120, Dec 2024.

[4] C. Willmott and K. Matsuura, "Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance," *Climate Research*, vol. 30, pp. 79–82, 2005.

[5] A. Yan, R. Cheng, and S. J. Wilton, "On the sensitivity of fpga architectural conclusions to experimental assumptions, tools, and techniques," *Proceedings of the 2002 ACM/SIGDA tenth international symposium on Field-programmable gate arrays - FPGA '02*, 2002.

[6] S. Ajami, A. Mahnam, and V. Abootalebi, "Development of a practical high frequency brain-computer interface based on steady-state visual evoked potentials using a single channel of eeg," *Biocybernetics and Biomedical Engineering*, vol. 38, no. 1, pp. 106–114, 2018.

[7] T. Thongkham and Y. Jewajinda, "A low-cost bit-serial hardware architecture and fpga implementation of spiking neural network," *2024 28th International Computer Science and Engineering Conference (ICSEC)*, pp. 1–4, Nov 2024.

[8] S. TAKAMAEDA-YAMAZAKI, H. NAKATSUKA, Y. TANAKA, and K. KISE, "Ultrasmall: A tiny soft processor architecture with multi-bit serial datapaths for fpgas," *IEICE Transactions on Information and Systems*, vol. E98.D, no. 12, pp. 2150–2158, 2015.