A Real-Time Machine Learning Module for Motion Artifact Detection in fNIRS

Renas Ercan
HUB of Intelligent Neuro-Engineering (HUBIN)(UCL)&Department of Physics(University of Cambridge)
University of Cambridge
Cambridge, UK
re378@cam.ac.uk

Yunjia Xia
HUB of Intelligent Neuro-engineering (HUBIN), DSIS
University College London(UCL)
London, UK
yunjia.xia.18@ucl.ac.uk

Yunyi Zhao
HUB of Intelligent Neuro-engineering (HUBIN), DSIS
University College London(UCL)
London, UK
yunyi.zhao.21@ucl.ac.uk

Rui Loureiro
Division of Surgery & Interventional Science(DSIS)
University College London(UCL)
London, UK
r.loureiro@ucl.ac.uk

Shufan Yang
Institute of Medical & Biological Engineering, School of Mechanical Engineering
University of Leeds
Leeds, UK
s.f.yang@leeds.ac.uk

Hubin Zhao
HUB of Intelligent Neuro-engineering (HUBIN), DSIS
University College London(UCL)
London, UK
hubin.zhao@ucl.ac.uk

***Abstract*— Functional Near-Infrared Spectroscopy (fNIRS) is a neuroimaging method which can be implemented with a wearable form factor. However, the data of fNIRS can be affected by motion artifact, which is conventionally processed offline using MATLAB-based software package via a bulky PC. This study trains a Support Vector Machine (SVM) algorithm and proposes a hardware design approach based on an FPGA to achieve the first real-time fNIRS motion artifact detection. The SVM hardware architecture proposed here utilizes a partially sequential–partially parallel implementation of the classification algorithm where Support Vector channels are consolidated into a single oversampled channel. A high classification accuracy of 97.42%, low FPGA resource utilization of 38,354 look-up tables and 6024 flip-flops with 10.92 us latency is achieved, outperforming conventional CPU SVM methods. These results show that an FPGA-based fNIRS motion artifact detector can be exploited whilst meeting real-time and resource constraints that are crucial in high-performance reconfigurable hardware systems.**

Keywords*— *fNIRS, machine learning, motion artifact detection, real-time, support vector machines (SVM), Field-programmable gate array (FPGA)

I. INTRODUCTION

Functional Near-Infrared Spectroscopy (fNIRS) serves as a non-invasive optical neuroimaging modality, quantifying the concentrations of oxygenated and deoxygenated hemoglobin proximate to the brain's surface, thereby enabling the inference of relative alterations in neural activation [1]. The advantages of fNIRS, epitomized by its relatively superior spatial resolution, cost-effectiveness, and portability, contribute to its broad spectrum of applications, encompassing cognitive neuroscience, clinical neurology, and personalized healthcare [1]. However, fNIRS data is prone to contamination by motion artifacts, which significantly deteriorates the quality of the recorded optical signals. These artifacts emerge from changes in the orientation or distance of the sources and detectors relative to the skull, induced by movements, thereby altering the impedance and inducing perturbations in the fNIRS signal. Consequently, these perturbations may engender misidentification of functional cortical activity [2]-[3].

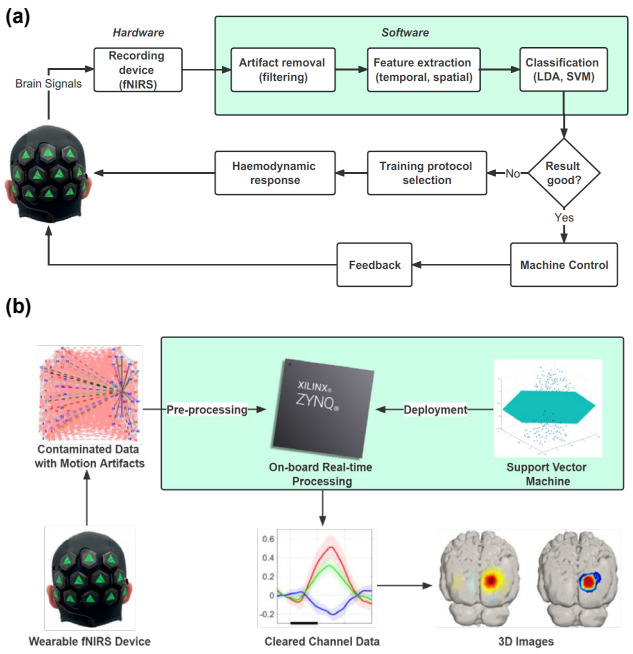

Fig. 1. (a) Conventional neuroimaging scheme for control applications. The green area specifically highlights the segment of imaging development that is addressed by this study. **(b)** Whole system design for FPGA motion artifact detector. The green area indicates our novelty motion artifact detection module.

Fig. 1(a) delineates a conventional neuroimaging system, wherein the green area represents the signal processing stream with standard fNIRS toolkits. Fig. 1(b) illustrates the comprehensive system design for an FPGA-based motion artifact detector in this study. A Support Vector Machine (SVM) has been seamlessly integrated into an FPGA chip, functioning as a real-time motion artifact detection module.

Prevailing methodologies for processing and mitigating motion artifacts in fNIRS data predominantly rely on robust, stationary hardware such as desktop PCs or workstations,

This research is supported by The Royal Society Research Grant (RGS/R2/222333), Engineering and Physical Sciences Research Council Grant (13171178 R00287), European Research Council (ERC) under the European Union's Horizon Europe Research and Innovation Program (No. 101099093), Department of Orthopaedics and Musculoskeletal Science/the Wellcome Trust/EPSRC through the WEISS Centre of UCL(203145Z/16/Z), SHED project Royal Academy of Engineering (IF2223-172) and Innovate UK KTP (013191). (*Renas Ercan and Yunjia Xia are co-first authors.*)

operating in an entire offline mode. These methodologies, although derived from various methods proposed in preceding studies, harbor inherent limitations as they are predicated on specific assumptions to characterize motion artifacts, coupled with a distinctive selection of parameters [4]. Such constraints curtail the broader applicability of fNIRS technology, relegating its utilization to controlled settings like laboratories or hospitals. Additionally, real-time fNIRS signal processing is imperative for expeditious decision-making in clinical scenarios, fluid interaction in Brain-Computer Interfaces (BCIs) and ensuring efficacy and data integrity in neuroscientific/clinical research. Herein, we introduce a real-time motion artifact detection module based on FPGA, meticulously crafted for wearable fNIRS systems, marking a pioneering attempt to furnish online, real-time hardware classification of fNIRS motion artifacts. This endeavor leverages a Machine Learning (ML) approach, employing SVM, which constitutes a cadre of rapid and reliable supervised machine learning classification algorithms, acclaimed for their superior classification velocities and memory efficiency. Their adeptness in swiftly learning and generalizing from training data renders them particularly propitious for real-time classification in embedded hardware applications. While SVMs have found applications in various fNIRS studies [5]-[6], it merits noting that none have addressed real-time classification of motion artifacts hitherto.

The seminal contributions of this paper encompass the formulation of a novel SVM module, realized through a software/hardware co-development paradigm, transcending the limitations inherent in software scheduling by proffering an online processing modality. Moreover, our Register Transfer Level (RTL) implementation exudes efficiency, obviating the necessity for instructions or shared memories.

Although FPGAs provide flexible digital circuit design and expansive parallel computational prowess, previous endeavors to harness these attributes have encountered noteworthy limitations [7]-[8]. These attempts frequently resorted to an array of simplification methods to ameliorate hardware complexity, albeit at the expense of classification accuracy. Additionally, the architectures devised in these attempts were devoid of both flexibility and scalability. Hence, this study aspires to provide a significantly more optimized solution for FPGA-based, real-time MA detection.

II. System Implementation

The SVM algorithm creates a decision boundary that can segregate n-dimensional space into classes so that new data points are easily classified into the correct category using soft margin or hard margin. The implementation process commenced with an architectural design phase conducted within the MATLAB R2020b and Simulink environment, laying down the foundational framework for the SVM aimed at motion artifact classification. Subsequent model development was rigorously validated using a Simulink testbench with the fNIRS motion artifact dataset, which had undergone Principal Component Analysis (PCA) for denoising and acceleration of model training. Insights derived from the validation phase informed iterative refinements, leading to a robust SVM implementation proficient in motion artifact classification, thereby achieving a high degree of accuracy and reliability in performance. In the evaluation of SVM model, all mis-classification errors are calculated to provide a value for the classification rate of each architecture.

Raw fNIRS data was obtained from a study wherein subjects wearing the fNIRS device were given tasks such as walking. The data was passed through an fNIRS specific data processing toolbox called Homer3 and a function called 'hmrMotionArtifact' to determine periods of motion artifact [9]. The purpose of finding these periods of motion artifacts was to provide labelled data for the training of the SVM model. Training datasets were created through the down sampling and balancing of a larger fNIRS dataset. The dataset includes 4 features as input and binary output to indicate motion artifacts and normal signal.

A. Training the SVM model

High-level language, Python, was used to program the SVM ML model algorithm as an easy way to tune parameters. Upon constructing the finished model, the support vectors were extracted with 55 support vectors generated, and the associated Lagrange multiplier coefficients and bias value were obtained.

B. System Design

The combined fNIRS data pre-processing and RBF kernel SVM algorithm architecture was simulated within the Simulink environment. This project utilized MATLAB and Simulink's automatic HDL code generation to convert the digital system architecture to HDL code, such HDL code creation methods generally produce Verilog code that is highly optimized and efficient whilst requiring minimal changes; these overall allows for a fast development time.

The RBF kernel in the SVM algorithm assumes that incoming data has been centered and scaled [8]. Therefore, the data is pre-processed so that incoming fNIRS signals need to be normalized. Data preprocessing is designed to use an exponentially weighted running mean and standard deviation blocks for fixed sample windows. An exponentially weighted running mean in the frequency domain can be represented as a real pole of which the implementation in the time domain is straightforward. Therefore, a single-pole IIR filter circuit can be created. Then, taking the z transform, we find the transfer function:

$$H(z) = \frac{a}{1-(1-a)z^{-1}} \qquad (1)$$

Here, $0 < a < 1$ is a constant that determines the effective length of the running average. To go to the continuous domain, we make the substitution $z = e^{sT}$, where T is the sample time and solve to find a pole at $s = \frac{1}{T}\log(1-a)$. Where we choose a as $a = 1 - \exp(\frac{2\cdot\pi\cdot T}{\tau})$, where τ is given to be the averaging time constant. The optimum value of τ and subsequently a was found through a comprehensive brute force search that evaluates classification accuracy as a result;

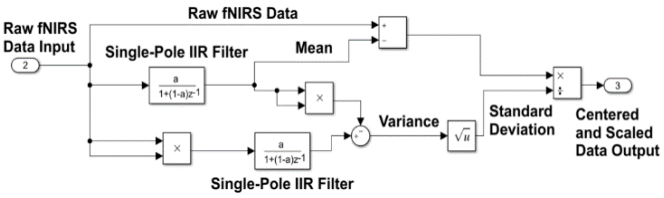

Fig. 2. Data preprocessing module (submodule in Fig.3 below).

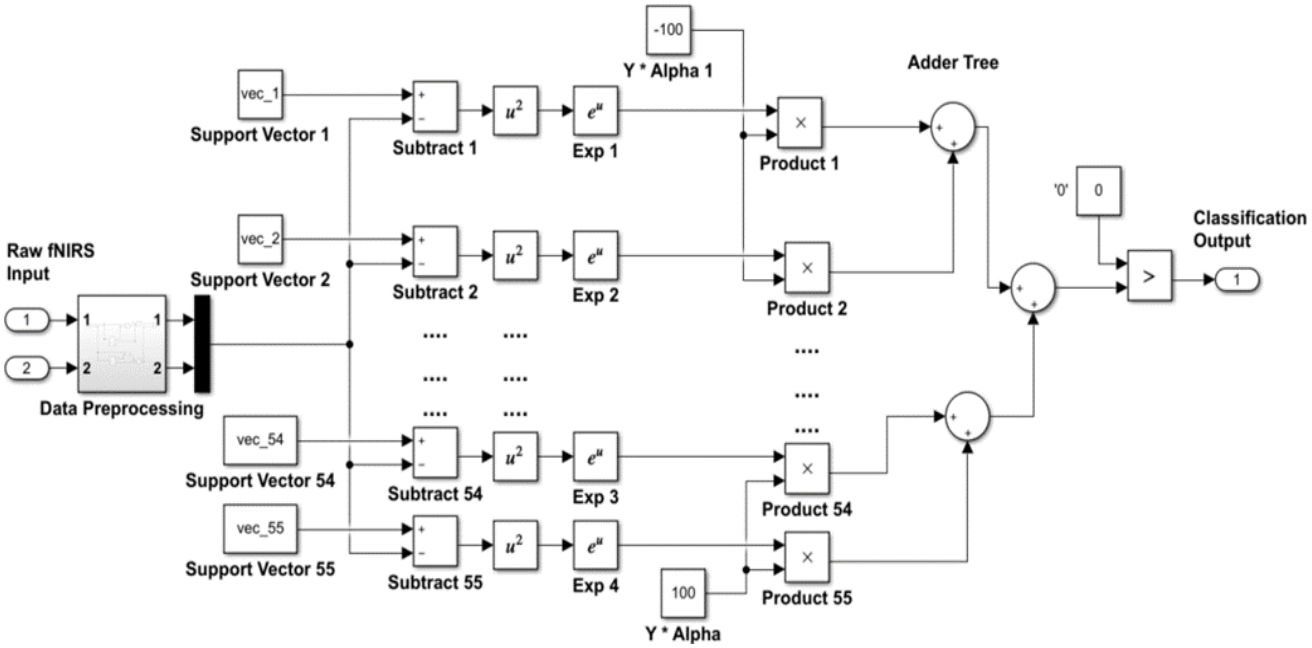

Fig. 3. Simulink architecture of the SVM algorithm in its three key units: kernel realization (A), inner-product addition with an adder tree (B), and a '0' threshold comparison (C).

the final value taken forward was $a = 0.01$. Given that the transfer function (1) calculates the exponentially weighted running mean, it can be used s to efficiently compute the variance and the standard deviation. The variance and standard deviation outputs of these computations are the pre-processed fNIRS data. The Simulink architecture used to process a single feature of the input fNIRS signal is shown in Fig. 2. Each feature of the fNIRS data (of which this work uses two) has a pre-processing RTL channel following the architecture given in Fig. 2 and explained above. It is noted that two pre-processing channels are operating in parallel.

We design the streaming SVM architecture based on the functional decomposition of the SVM kernel. Here the fundamental arithmetic operations of the gaussian radial basis function kernel (2) are directly mapped to Simulink arithmetic blocks. The proposed SVM hardware design, which is segmented into three principal blocks (as shown in Fig. 3): a kernel realization (A), inner-product addition with an adder tree (B), and a threshold comparison (C) [8]. The support vector values, and Lagrange multiplier coefficients were pre-defined using software SVM. Fig. 3 shows the data pre-processing feeding processed fNIRS inputs into the SVM algorithm architecture. The pre-processed fNIRS data is streamed into square difference units with the fifty-five support vectors. These calculate the square difference between the fNIRS signal and the support vectors before being passed to exponential function units, which perform the calculations

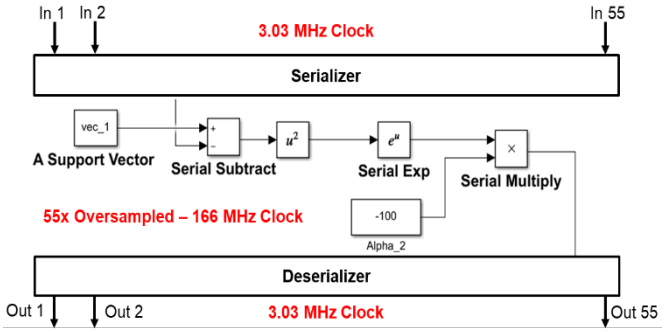

Fig. 4. The new and final architecture of the SVM RTL demonstrating the serial singular and oversampled channel.

required to achieve the RBF kernel function. The adder tree and multipliers construct the classification function (1). Finally, the classification results are forecasted using the output of the adder tree and a relational operator compared to '0' - the classification that indicated the presence of a motion artifact.

III. Design and Simulation

The RTL digital design is evaluated using Xilinx Vivado and then generated bitstream to download into Xilinx Zynq Ultrascale+ MPSoC AXU3EG development board [10]. Simulink tools generate arithmetic modules and capture the digital design in Verilog code. In this work a zero-latency strategy was pursued, and all HDL was written in the IEEE754 32-bit single-precision floating-point format.

The overarching philosophy utilizing a RTL design was to gradually replace critical blocks designed and tested in the Simulink architecture with synthesizable Verilog blocks, ultimately creating a streaming architecture. The entire gate-level design employs this streaming architecture where the output of a subsystem is fed directly to the input of the next subsystem. Hence, we can attain a low latency, as the results of each subsystem are not stored in off-chip memory but instead can be immediately used.

The underlying principle of the SVM classifier architecture

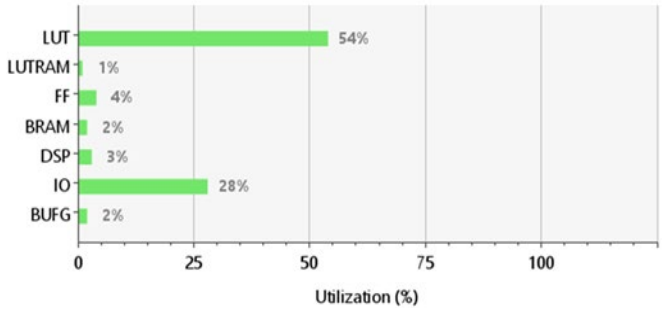

Fig. 5. Visual depiction of the key FPGA resource of SVM and the utilization of these resources within the FPGA.

TABLE I. CLASSIFICATION ACCURACY RESULTS FOR THE SIMULINK SVM MODEL AND RTL DESIGNS

Single Channel Oversampled SVM Model		
Dataset Used in Testing	***Simulink Classification Accuracy***	***RTL Classification Accuracy***
Balanced Dataset 1	91.94%	100.00%
Balanced Dataset 2	92.41%	97.80%
Unbalanced Dataset 1	91.14%	95.87%
Unbalanced Dataset 2	89.82%	96.00%
Average Classification Accuracy	91.33%	97.42%

was to exploit the FPGA's parallel computational power and resources to execute the decision function most efficiently; computation of this function involves highly parallelizable vector operations. Consequently, the data pre-processing and RBF Simulink designs were partitioned into core floating-point arithmetic operations that form the real pole IIR filter and parallel Support Vector channels. The proposed gate-level HDL design for the SVM classifier mirrors the digital Simulink architecture illustrated in Fig. 3. It employs the parallelism inherent in FPGAs, enabling all fifty-five Support Vector channels to operate concurrently and thus facilitating a parallelized classification system. The raw fNIRS input signal is streamed into the FPGA and processed through the pre-processing units. Subsequently, the kernel calculation, which forms the foundation of the SVM algorithm's RTL, is undertaken.

This computation employs basic floating-point processing units such as adders and multipliers. These units utilize the FPGA's native parallelism to significantly expedite the SVM decision function computation. The system design is conscious of resource utilization, integrating a scheme that transforms the fully parallel design into a hybrid model that operates partially in parallel and partially serially. The fifty-five Support Vector channels are then integrated into a singular stream of time-multiplexed samples on a single channel, optimizing the resource-intensive kernel and inner-product accumulation RTL hardware, shown in Fig. 4.

To maintain effective timing for each channel in this streaming design, the single shared channel's RTL is oversampled at 55 times the base clock rate of the overall model. As a result, the model's latency is only extended by one cycle of the base rate. However, this necessitates a reduced base rate of 45.45 kHz to accommodate the 200 MHz clock speed for the oversampled channel, leading to a decrease in the model's running speed. Consequently, this design choice optimizes resource consumption at the cost of increased complexity, more intricate scheduling, and elevated model latency. After processing the data from the fifty-five Support Vector channels at the oversampled rate, the data are momentarily stored in CLB flip-flops. At the end of each base clock cycle, the samples are deserialized back into a parallel format for propagation through the adder tree, preparing them for classification.

IV. RESULTS AND DISCUSSION

The overarching metric used for model evaluation was classification accuracy. When each of the datasets was applied to the model, we used the number of motion artifacts as determined by the digital architectural or gate-level error counter. The fewer motion artifacts incorrectly identified, the better the model performance. In addition, the metrics of FPGA resource allocation is used as complementary values to evaluate each model, particularly when looking at the work's real-time objectives.

Classification accuracy and FPGA resource utilization form the key performance metrics in this study. Derived from Simulink architecture simulations and synthesis, the average classification accuracy for digital architectural and gate-level simulations stand at 91.33% and 97.42% respectively, as outlined in Table I.

The criticality of FPGA resource utilization comes into play when considering area requirements. If these requirements are not met, the RTL design cannot be transferred onto the hardware device. Different digital designs and work objectives dictate the value of various resources. Given the real-time objectives of our neuroimaging FPGA, the priority is to minimize the utilization of specific resources. Consequently, memory logic utilization remains intentionally low. Fig. 5 illustrates overall FPGA resource utilization.

Our results highlight a substantial underutilization of a crucial resource, the Look-Up Tables (LUTs), with a consumption rate of only 54% in the final RTL design. This underutilization is a consequence of adopting a single-channel oversampling architecture. The benefits of this approach extend beyond a good fit for our RTL design on the FPGA used, making the SVM RTL highly transferable to numerous other FPGAs in the market.

Applying different optimization directives led to an increase in latency (10.92 us) and a decrease in throughput, effectively trading off with low resource utilization. Our system can handle 45450 samples per second, allowing for a throughput of 45.45 Kilosamples (or Kilobits) per second. Given a frame rate of 15 frames per second in an fNIRS system, this equates to 3030 real-time processing channels, thus enabling the system to accommodate most commercially available fNIRS systems for real-time motion artifact detection.

V. CONCLUSION

In conclusion, this work presents a notable advancement in the machine-learning based real-time processing of fNIRS data, crucial in real-world environments and seamless BCIs. The designed FPGA-based SVM machine learning classification algorithm offers a robust solution for the real-time identification and detection of motion artifacts. This study brings forward a specifically designed and systematically tested FPGA machine learning platform for fNIRS modality, marking its novelty. Demonstrating high processing speed and accuracy in motion detection, the developed architecture successfully maintains low area and latency. Hence, the presented work here opens new possibilities for effective real-time motion artifact detection in commercially available fNIRS systems.

REFERENCES

[1] Zhao Y, Luo H, Chen J, et al. Learning based motion artefacts processing in fNIRS: A mini review. Frontiers in Neuroscience, 17: 1280590.

[2] D. Perpetuini, D. Cardone, C. Filippini, A. M. Chiarelli, and A. Merla, "A Motion Artifact Correction Procedure for fNIRS Signals

Based on Wavelet Transform and Infrared Thermography Video Tracking," *Sensors*, vol. 21, no. 15, p. 5117, Jul. 2021, doi: 10.3390/s21155117.

[3] R. J. Cooper *et al.*, "A Systematic Comparison of Motion Artifact Correction Techniques for Functional Near-Infrared Spectroscopy," *Front Neurosci*, vol. 6, 2012, doi: 10.3389/fnins.2012.00147.

[4] Y. Gao *et al.*, "Deep learning-based motion artifact removal in functional near-infrared spectroscopy," *Neurophotonics*, vol. 9, no. 04, Apr. 2022, doi: 10.1117/1.NPh.9.4.041406.

[5] B. Koo *et al.*, "A hybrid NIRS-EEG system for self-paced brain computer interface with online motor imagery," *J Neurosci Methods*, vol. 244, pp. 26–32, Apr. 2015, doi: 10.1016/j.jneumeth.2014.04.016.

[6] F. Putze *et al.*, "Hybrid fNIRS-EEG based classification of auditory and visual perception processes," *Front Neurosci*, vol. 8, Nov. 2014, doi: 10.3389/fnins.2014.00373.

[7] X. Song, H. Wang, and L. Wang, "FPGA Implementation of a Support Vector Machine Based Classification System and Its Potential Application in Smart Grid," in *2014 11th International Conference on Information Technology: New Generations*, IEEE, Apr. 2014, pp. 397–402. doi: 10.1109/ITNG.2014.45.

[8] S. Afifi, H. GholamHosseini, and R. Sinha, "FPGA Implementations of SVM Classifiers: A Review," *SN Comput Sci*, vol. 1, no. 3, p. 133, May 2020, doi: 10.1007/s42979-020-00128-9.

[9] T. J. Huppert, S. G. Diamond, M. A. Franceschini, and D. A. Boas, "HomER: A review of time-series analysis methods for near-infrared spectroscopy of the brain," *Appl Opt*, vol. 48, no. 10, Apr. 2009, doi: 10.1364/AO.48.00D280.

[10] "ALINX AXU3EG or AXU3EGB: Xilinx Zynq UltraScale+ MPSOC ZU3EG Ethernet FPGA development board." https://www.xilinx.com/products/boards-and-kits/1-1cm64x4.html (accessed Jun. 14, 2023).