A Neural Machine Interface Architecture for Real-Time Artificial Lower Limb Control

Jason Kane, Qing Yang, Robert Hernandez, Willard Simoneau, and Matthew Seaton
Dept. of Electrical, Computer, and Biomedical Engineering
University of Rhode Island
Kingston, RI, 02881, USA

Abstract— **This paper presents a novel architecture of a lower limb neural machine interface (NMI) for determination of user intent. Our new design and implementation paves the way for future bionic legs that require high speed real-time deterministic response, high accuracy, easy portability, and low power consumption. A working FPGA-based prototype has been built, and experiments have shown that it achieves average performance gains of around 8x that of the equivalent software algorithm running on an Intel Core i7 2670QM, or 24x that of an Intel Atom Z530 with no perceivable loss in accuracy. Furthermore, our fully pipelined and parallel non-linear support vector machine-based FPGA implementation led to a 6.4x speedup over an equivalent GPU-based design. In this paper, we also characterize our achieved timing margin to show that our design is capable of supporting real-time wireless communications. With additional refinement, such a wireless personal area network (PAN) system will provide improved flexibility on an individual basis for electromyography (EMG) sensor placement.**

Keywords—Field Programmable Gate Arrays (FPGA); Support Vector Machines (SVM); Neural Machine Interface (NMI); Artificial Leg Control; Parallel Architectures

I. INTRODUCTION

Until recently, commercially available lower limb prosthetics have mainly focused on the use of simple passive devices. Surface electromyography (EMG) is one sensory interface that has been used in the past to successfully predict active user intent to control upper limb devices [1]. The main issue with applying this pattern recognition (PR) technology to patients with transfemoral (TF) amputations is that the detected lower limb EMG signals are typically non-stationary during locomotion. These EMG signals are however quasi-cyclic with respect to locomotion gait and somewhat stationary within short windows. In [2], it was shown that a phase-dependent EMG pattern classifier could be constructed to accurately predict intended user movements. When the EMG data is fused with measured moments and ground reaction force from the artificial limb, a more accurate algorithm was developed, achieving over 95% accuracy for locomotion detection [3] [4] [5]. The algorithm in [3] and [4] used linear discriminate analysis (LDA) for classification, and was later realized in a field programmable gate array (FPGA) to obtain real-time performance [6]. To achieve higher accuracy, a support vector machine [7] (SVM) based classifier has been studied [5] and shown to provide greater accuracy than the LDA approach. Because the ultimate goal is to control an artificial limb in a real-time and wearable environment, a portable embedded system is desirable with low form factor and low power consumption.

In this paper, we present a new lightweight design of the more accurate SVM-based algorithm in an FPGA. Our FPGA implementation differs greatly from that presented by the LDA approach in [6], as it utilizes a non-linear SVM-based detection algorithm. While this increases the design complexity, it has been shown to provide higher accuracy in determining user intent over the LDA approach [5]. Furthermore, in an effort to increase performance via pipelining and parallelism, our FPGA implementation was written and optimized in VHDL. In contrast, the previously created LDA-based FPGA was auto-generated using Impulse C C-to-HDL CoDeveloper [8].

This paper makes the following contributions:

(C1) Development of an architecture providing a well-defined, deterministic response using a previously published highly accurate SVM-based algorithm. An FPGA implementation should execute faster than a general purpose CPU due to the exploitable parallelism of the algorithm. Dedicated pipelined hardware driven by state machines will guarantee lower and more consistent response times than a traditional CPU or GPU, both of which are reliant on operating systems. This becomes important when one's stability and safety are dependent on a functional system.

(C2) Introduction of a power and area efficient architecture for artificial lower limb control. Most importantly, this work should serve as the first step in creating a physically smaller and lower power ASIC implementation of the SVM-based algorithm. This is of the utmost importance, since our computational engine needs to fit within the volume constraints of the artificial leg and provide a minimum of 6 to 8 hours of continuous use.

(C3) Demonstration of a system capable of real-time wireless communications. Making comparisons with an existing real-time wireless protocol, we will show that sufficient prediction time slack exists such that our design can communicate with sensory devices in a time critical manner. Wireless sensors are desired for EMG readings because, depending on a TF amputee subject's residual nerves, some muscle locations may be preferred over others for placement. Our eventual wireless system should provide flexibility, allowing us to target those regions on an individual basis.

The remainder of this paper is organized as follows. In the next section we briefly introduce our proposed real-time system and describe our existing NMI algorithm. Section 3

This research was sponsored in part by the Department of the Navy (Naval Undersea Warfare Center, Newport, Rhode Island), the Office of Naval Research (ONR) Independent Applied Research (IAR) initiative, and NSF grants CCF-1017177 and CCF-1421823.

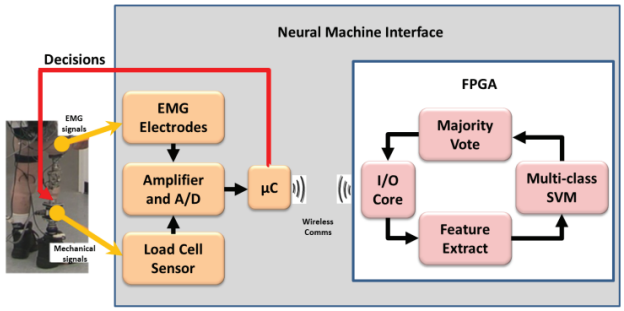

Fig. 1. Proposed Target Control System

details our prototype hardware implementation. In Sections 4 and 5 we introduce our testing methodology and discuss our results. We then present our conclusions and discuss future work in this area.

II. System and Algorithm Design

The proposed final state of our prosthetic system can be seen in Fig. 1. A detailed description of data flow in the system is summarized in [9]. Capabilities and limitations of the implemented prototype design are discussed in Section 3.

To obtain force and moment measurements, the artificial lower limb contains an integrated six degrees of freedom (6-DOF) load cell manufactured by Bertec Corporation, model AM6514. Intended user movements are captured through the use of seven EMG sensors. These sensors are placed at several strategic muscle locations that are dependent on the subset of classifications being performed. Analog readings from all thirteen channels are amplified and digitized as 16-bit samples at a 1kHz sampling rate and then transmitted over a wireless interface. The amplification, A/D conversion, and wireless transmission occur at each sensor individually. In the final design a real-time protocol, such as RT-WiFi [10], will be used to transfer the data within a guaranteed time window. This creates a personal area network (PAN) and allows for the sensors to be placed with much greater flexibility.

The wireless data is received by an input/output (I/O) core which forwards the raw ADC data to the FPGA's feature extractor. Once the feature extractor has obtained enough sensor data to occupy one window, a set of features are computed and forwarded to the multiclass SVM module. The SVM module takes these features and performs the current classification of user intent, which is then forwarded to a ten-point majority vote algorithm. The result of the majority vote is the final predicted user intention, which is sent through the wireless link back to a device on the artificial limb that will use the information to adjust limb impedance and position accordingly via its own separate state machine. Fig. 2 depicts a timeline of the major events, as seen from the FPGA, that take place during operation. Details of notable algorithm events are described in [9].

III. Prototype System Architecture Implementation

For our concept prototype, we implement the FPGA portion of the design in Fig 1. We later use offline EMG/Load Cell data gathered from our existing lower limb prosthesis hardware for validation testing in Sections 4 and 5. A high

Fig. 2. System Event Timeline

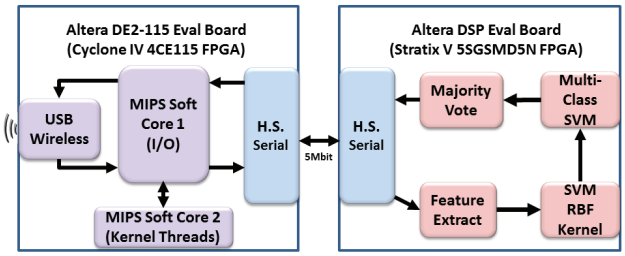

Fig. 3. System Architecture Implementation

level block diagram of the hardware implementation can be seen in Fig. 3. In the subsections that follow, the major three components of the prototype design are discussed.

A. Wireless and Inter-Module Communications

A dual core implementation of a superscalar MIPS-like CPU was synthesized to act as a transmitter/receiver between incoming raw EMG and load cell data and outgoing predictions. Due to its small footprint, USB device compatibility, and built-in support for several standard wireless communication protocols, a real-time embedded build of Linux was chosen for an operating system. One core on the system is dedicated to handling kernel tasking. The other core is responsible for receiving wireless data and transmitting it to the Feature Extractor/SVM module over a fast 5Mbit serial link as well as receiving predictions from the serial link and relaying them wirelessly back to the artificial limb.

B. Feature Extraction and Phase Detection

The feature extraction unit is depicted in detail in Fig. 4. At the front-end of the unit, incoming digital samples are tagged by channel and sample number. To minimize computational noise, our hardware implementation performs as many lossless operations as possible in the integer domain before converting to floating point for use with the existing SVM classifier models. The use of less complex integer operations also saves clock cycles and reduces hardware complexity. Fig. 4 uses a dashed line to delineate between the use of integer and floating point operations in the hardware.

Data initially enters the feature extraction unit in unsigned 16-bit format, but is converted to signed 32-bit format for performing feature calculations. Transferring the data in its original 16-bit format saves valuable wireless transmission time. Information regarding incoming frames is kept, and each functional unit automatically begins calculating its features upon receiving the last sample it needs.

Six load cell units operate in parallel, one for each channel. Integer operations are performed to determine the minimum, maximum, and mean of the last eight data windows. A conversion to floating point then takes place and the resulting 18 features are normalized. Like the load cell units, one EMG unit exists for each EMG channel, resulting in a total of 7 parallel units. Three features (the mean of the absolute value of all samples, number of zero crossings, and number of turns) are calculated using only integer operations, then converted to floating point and normalized. On the other hand, the length

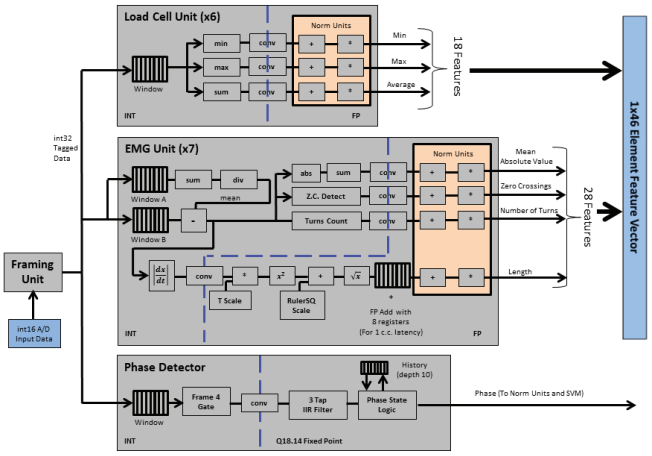

Fig. 4. Feature Extraction Unit

feature is mostly calculated using floating point arithmetic. This is necessary due to the presence of a square root operation, which would result in loss of precision if equivalent integer operations were performed. To reduce latency in the length calculations, a seven-stage pipelined floating point addition unit at the final calculation stage is used to compute eight partial sums that are continually re-injected into the pipeline, rather than stall the pipeline for each addition operation. When all partial sums are computed, they are combined together, and the result is normalized and presented as a feature to the SVM unit.

The phase detection unit performs a 3-tap IIR filter operation on the fifth oldest window of vertical ground reaction force data. The integer data is converted to fixed point Q18.14 format to reduce the amount of time taken for calculations. The output of the filter is used as input to a simple state machine to predict the current gait phase. The prediction is stored in a ten entry history buffer to aid in future predictions.

C. *Multiclass SVM and Majority Vote*

The Multiclass SVM unit is designed to handle three classes. The SVM unit's block RAM is primed with model data prior to run-time, including support vectors, combined coefficient and class labels, and bias values. The RAM is designed to accept and store up to four different models, corresponding to the four gait phases. When the SVM unit is signaled to begin a classification, the stored SVM model corresponding to the current gait phase at the time of feature latching is used for all subsequent operations. A multi-stage pipelined floating point Gaussian kernel computation [11] then begins. Once the first valid sample exits the pipeline, subsequent valid samples are output on each clock cycle until all remaining calculations are complete. The scalar outputs from the kernel operation are stored in RAM and later retrieved as needed by the SVM weighting operation that follows.

After kernel function execution, the SVM decision function [7] is evaluated. The hardware achieves high throughput by attempting to maximize the total number of parallel weighting operations. The final value for each SVM comparison is delivered in a pipelined fashion to a vote counting unit, which determines the winner for each binary classification and tallies votes. When it has detected that a target number of votes have been placed, the final classification is sent to the majority vote unit.

The majority vote hardware maintains a sliding window of the last 10 predictions. Prior to receiving a new prediction, it adjusts its window and tallies the votes for each stored classification. When the new classification is available only a few clock cycles are required to update the vote tally and output the final result.

IV. Experimental Evaluation Methodology

This study was conducted with Institutional Review Board (IRB) approval at the University of Rhode Island and informed consent of subjects. Testing occurred in two phases: feature extraction and SVM performance evaluation, and real-time wireless system performance evaluation. For all testing, 3-Class offline data from [9] was utilized to benchmark the performance of the new system.

Two sets of 3-Class data were obtained from a male able-bodied subject. The first set of data was used to predict stair ascent and consisted of 6905 individual classification tests. The second set of data was used to predict walking motion and consisted of 7391 classification tests.

A software program was created to transmit required input data wirelessly to the FPGA module under test and receive responses. Cycle-accurate counters embedded within the FPGA hardware determined the amount of time to independently perform both feature extraction and multiclass SVM. This timing information was reported back to the software program along with the resulting prediction. Accuracy and timing benchmarks were achieved by comparing against a current generation Intel i7 CPU running a previously published C software-based implementation [9]. The Intel i7 system under test consisted of the following major components: an Intel Core i7 2670QM CPU, 6 GB of RAM, and Windows 7 64-bit operating system.

V. Performance Evaluation

After design compilation, the Altera Quartus II tool predicted that our total thermal design power (TDP) is 2.3 Watts, roughly matching that of the Intel Atom Z530 mobile CPU that ran the comparable NMI algorithm in [9]. Our complete system should require less power than this mobile device, as all memory and most communications resources are contained within the Stratix V.

During the first phase of testing, the offline A/D sample data from the 20ms 3-Class trial was sent to the feature extractor in real-time. The three classes predicted by this model include stair ascent, standing, and level-ground walking. The system reported back the class prediction and timing taken to perform feature extraction and multiclass SVM prediction. The mean timing and overall speed-up achieved over the Intel i7 by the FPGA to complete both feature extraction and multiclass SVM predictions can be seen in Table I.

In [9], the Intel Z530 Atom achieved a mean prediction time of 0.721ms over the same set of trials and in [12], a highly parallelized GPU-based version of our algorithm was run on a 35W GeForce GT 540m with 96 CUDA Cores, yielding an average prediction time of 0.192ms. Comparing with our

TABLE I. Feature Extraction and SVM Prediction Timing

Trial	Average Time Taken (ms)		FPGA Speed Increase		
	Intel i7	*FPGA*	*Min*	*Max*	*Average*
3 Class Stair Up	0.25	0.03	5.51x	171.46x	8.40x
3 Class Walking	0.24	0.03	5.54x	25.14x	8.03x

results in the table, we have achieved a speedup of 24x and 6.4x respectively over these platforms. In the case of the GPU, a 15.2x power advantage is also achieved. Another interesting comparison can be seen between the SVM and LDA-based FPGAs. In an equivalent 3-Class trial, [6] reports the average prediction time of its FPGA as being 0.23ms. As seen in Table I, the SVM-based FPGA only takes 0.03ms on average – a 7.6x performance increase. While LDA generally requires fewer computations than SVM, the hardware optimizations present in our design, combined with the fact that the LDA FPGA was created with a software to HDL converter both contribute to this unexpected difference in performance.

Unlike the CPU implementation, our feature extraction consistently occurs within a fixed number of clock cycles, and since the amount of time taken for a classification decision is proportional to the selected gait phase, predictions can be expected to occur within a precise, guaranteed timeframe. Further analysis showed that with the exception of one prediction, results were identical to those produced by the Intel i7, and met the high level of accuracy reported in [5] and [9]. The one mismatch was determined to be a near-boundary case.

In the second phase of testing, the software program used previously for data injection was modified to record the round trip timing of the system. Timing was recorded for both wired Ethernet and 802.11g WiFi using TCP/IP with the Nagle algorithm disabled for speed. The wired Ethernet implementation was chosen as a control to ensure that the soft-core CPU and SVM-based limb algorithm hardware were capable of handling the latency and throughput required for real-time communications. The desired processing slack time was determined using data from [10]. In their paper, the developers of RT-WiFi found that in a noisy environment, a payload of 460 bytes will incur a transmission delay of at most 4.2ms. Extrapolating linearly to the 520 bytes required for each 20ms window prediction, it can be expected that our system should achieve a guaranteed slack time of at least 4.75ms per window.

The same trial subsets of 3-Class data used during the first phase of testing were again used to determine performance. In the wired implementation, all prediction responses were received within the desired 20ms prediction timeframe. Most arrived within 4ms, with a few taking as long as 14ms. This yields a minimum slack time of 6ms, which satisfies our derived timing requirement. As expected, during 802.11g WiFi testing, the system was incapable of meeting real-time demand, with responses of up to 60 ms. Clearly, a solution similar to RT-WiFi is needed to achieve the desired response.

VI. Conclusions

This paper explored using FPGAs to improve the real-time performance of an existing NMI algorithm for lower limb control. It was found that by implementing the real-time limb control algorithm in an FPGA we are able to meet required accuracy, while completing all computations within a much smaller and bounded timeframe when compared to the previous software-based implementations. In addition, the FPGA provides a wearable footprint and power rating required by limb control. Analyzing the published data from the authors of [10] along with our wired Ethernet results, we found that the prototype system should be capable of real-time communications if a proper custom data link layer is substituted. Future work may be performed on the topic of further power reduction. ASIC implementations have been known to yield considerably better power ratings than FPGAs and require significantly less volume. A study will need to be undertaken to quantify the advantages an ASIC design would have over an FPGA.

References

[1] K. Englehart and B. Hudgins, “A robust, real-time control scheme for multifunction myoelectric control,” IEEE Transactions on Biomedical Engineering, vol. 50, pp. 848-854, Jul 2003.

[2] H. Huang, T. A. Kuiken, and R. D. Lipshutz, “A strategy for identifying locomotion mode using surface electromyography,” IEEE Trans Biomed Eng, vol 56, pp. 67-73, 2009.

[3] H. Huang, Y. Sun, Q. Yang, F. Zhang, X. Zhang, Y. Liu, J. Ren and F. Sierra, “Integrating Neuromuscular and Cyber Systems for Neural Control of Artificial Legs” in The ACM/IEEE First International Conference on Cyber-Physical Systems, Sweden, 2010.

[4] Fan Zhang; DiSanto, W.; Jin Ren; Zhi Dou; Qing Yang; He Huang, “A Novel CPS System for Evaluating a Neural-Machine Interface for Artificial Legs,” Cyber-Physical Systems (ICCPS), 2011 IEEE/ACM International Conference on , pp.67-76, 12-14 April 2011.

[5] H. Huang, F. Zhang, L. J. Hargrove, Z. Dou, D. R. Rogers, and K. B. Englehart, "Continous locomotion-mode indentification for prosthetic legs based on neuromuscular-mechanical fusion," IEEE Trans Biomed Eng, vol 58, pp. 2867-2875, 2011.

[6] Xiaorong Zhang; He Huang; Qing Yang, “Implementing an FPGA system for real-time intent recognition for prosthetic legs,” Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE, pp.169-175, 3-7 June 2012.

[7] C. Cortes and V. Vapnik, “Support-Vector Networks,” Machine Learning, vol 20, no 3, pp. 273-297, Sept. 1995.

[8] Impluse Accelerated Technologies. “Impulse CoDeveloper C-to-FPGA Tools.” Internet: www.impulseaccelerated.com/products_universal.htm, 2013 [October 10, 2013].

[9] Hernandez, R.; Qing Yang; He Huang; Fan Zhang; Xiaorong Zhang, “Design and implementation of a low power mobile CPU based embedded system for artificial leg control,” Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE , pp. 5769-5772, 3-7 July 2013.

[10] Yi-Hung Wei; Quan Leng; Song Han; Mok, A.K.; Wenlong Zhang; Tomizuka, M., “RT-WiFi: Real-Time High-Speed Communication Protocol for Wireless Cyber-Physical Control Applications,” Real-Time Systems Symposium (RTSS), 2013 IEEE 34th, pp.140-149, 3-6 Dec. 2013.

[11] J. H. Friedman, “Another Approach to Polychotomous Classification,” Stanford University, Stanford, CA, 1996.

[12] Hernandez, R.; Faella, J., “Towards policy and guidelines for the selection of computational engines,” Systems Conference (SysCon), 2013 IEEE International , pp.88-95, 15-18 April 2013.