Implementing an FPGA System for Real-Time Intent Recognition for Prosthetic Legs

Xiaorong Zhang, He Huang, and Qing Yang
University of Rhode Island
{zxiaorong, huang, qyang}@ele.uri.edu

ABSTRACT
This paper presents the design and implementation of a cyber physical system (CPS) for neural-machine interface (NMI) that continuously senses signals from a human neuromuscular control system and recognizes the user's intended locomotion modes in real-time. The CPS contains two major parts: a microcontroller unit (MCU) for sensing and buffering input signals and an FPGA device as the computing engine for fast decoding and recognition of neural signals. The real-time experiments on a human subject demonstrated its real-time, self-contained, and high accuracy in identifying three major lower limb movement tasks (level-ground walking, stair ascent, and standing), paving the way for truly neural-controlled prosthetic legs.

Categories and Subject Descriptors
C.3 [**Special-purpose and Application-based Systems**]: Real-time and embedded systems.

General Terms
Algorithms, Performance, Design, Experimentation

Keywords
Neural-machine interface, embedded system, prosthetic leg, field-programmable gate array (FPGA)

1. INTRODUCTION
Neural-machine interface (NMI) is a typical example of biomedical cyber physical system (CPS) which utilizes neural activities to control machines. The neural signals collected from nerves, central neurons, and muscles contain a lot of important information that can represent human states such as emotion, intention, and motion. In such a CPS, a computer senses bioelectric signals from a physical system (i.e. human neural control system), interprets these signals, and then controls an external device, such as a power-assisted wheelchair [1], a telepresence robot [2], or a prosthesis [3-5], which is also a physical system.

The neural signals captured from muscles are called electromyographic (EMG) signals. The EMG signals can be picked up with electrodes on the body surface and are effective bioelectric signals for expressing movement intent. In recent years, EMG-based NMI has been widely studied for control of artificial limbs in order to improve the quality of life of people with limb loss. Researchers have aimed at utilizing neural information to develop multifunctional, computerized prosthetic limbs that perform like natural-controlled limbs. The NMI needs to interface with multiple sensors for collecting neural signals, decipher user intent, and drive the prosthetic joints simultaneously. EMG pattern recognition (PR) is a sophisticated technique for characterizing EMG signals and classifying user's intended movements. It usually contains a training phase for constructing the parameters of a classifier from a large amount of EMG signals, and a testing phase for recognizing user intent using the trained classifier. While the PR algorithm for artificial arm control has been successfully developed and neural-controlled prosthetic arms have already been clinically tested [4, 6-7], there has been no EMG-based NMI commercially available for control of powered prosthetic legs. Challenges in the management of both physical and computational resources have limited the success of a CPS for neural control of artificial legs.

One of the challenges on physical resources is due to the muscle loss of leg amputees. Patients with leg amputations may not have enough EMG recording sites available for neuromuscular information extraction [8]. The non-stationary of EMG signals during dynamic leg movement further increases the difficulty of user intent recognition (UIR). To address this challenge, Huang et al. proposed a phase-dependent PR strategy for classifying user's locomotion modes [8]. This PR algorithm extracted neural information from limited signal sources and showed accurate classification (90% or higher accuracy) of seven locomotion modes when 7-9 channels of EMG signals were collected from able-bodied subjects and leg amputees. The performance of the phase-dependent PR strategy was further improved by incorporating EMG signals with mechanical signals resulting from forces/moments acting on prosthetic legs [9]. The experimental results showed that the classification accuracies of the neuromuscular-mechanical fusion based PR algorithm were 2%-27% higher than the accuracies derived from the strategies using EMG signals alone, or mechanical signals alone [9].

The challenges on computational resources include tight integration of software and hardware on an embedded computer system that is specifically tailored to this environment. It requires high speed classifier training, fast response, real-time decision making, high reliability, and low power consumption. Embedded systems are usually resource constrained and typically have processors with slower system clock, limited memory, small or no hard drives. To make the idea of neural-controlled artificial leg a reality, we need efficiently manage the constrained computational resources to meet all the requirements for smooth control and safe use of prosthetic legs. Our previous study proposed an NMI implemented on a commodity 32-bit microcontroller unit (MCU) for recognizing two non-locomotion tasks of sitting and standing in real-time [10]. It was reported that there was a noticeable delay of 400 ms for producing classifying decisions, implying inadequate computational power of the MCU for real-time control of artificial legs. Furthermore, this NMI implementation realized only the testing phase of the PR algorithm on the MCU. The training algorithm which involved intensive computations was implemented on graphic processing units (GPUs) and showed good speedups

DAC 2012, June3–7, 2012, San Francisco, California, USA.

over CPU-based implementation [10]. However, currently most of the GPU cards only have PCI Express interfaces and are not portable. Relative high power consumption further makes it more difficult to use GPU as an embedded wearable device.

To tackle these technical challenges, we present a new design of an embedded system that is specifically tailored to the new NMI. A unique integration of hardware and software of the embedded system is proposed that is suitable to this real-time CPS with adequate computational capability, high energy efficiency, flexibility, reliability, and robustness. The NMI on an embedded platform continuously monitors EMG activities from leg muscles as well as mechanical forces/moments acting on prosthetic legs. Information fusion technique is then used to decode and decipher the collected signals to recognize users' intended locomotion modes in real-time. The embedded system contains two major parts: a data collection module for sensing and buffering input signals and an intelligent processing unit for executing the UIR algorithm. The data collection module was implemented on a microcontroller unit (MCU) with multiple on-board analog-to-digital converters (ADCs) for signal sampling. A reconfigurable FPGA device was designed as the main computing engine for this system. There are several reasons for choosing FPGAs for the designed NMI. First, the parallelism of FPGAs allows for high computational throughput even at low clock rates. Secondly, FPGAs are not constrained by a specific instruction set, thus are more flexible and more power efficient than processors. Furthermore, FPGAs can easily generate customized IO interfaces with existing IP cores, and appear to be good choices for real-time embedded solutions. In our design, a high-level synthesis tool was used to help reducing the implementation difficulty of coding with hardware design language (HDL). A special parallel processing algorithm for UIR was designed, realizing the neuromuscular-mechanical fusion based PR algorithm coupled with the real-time controlling algorithm in hardware. A serial peripheral interface (SPI) was built between the MCU and the FPGA to transfer digitized input data from the MCU to the FPGA device. The decision stream of user's intended movements can be output to either control a powered prosthetic leg or drive a virtual reality (VR) system with the purpose of evaluating the NMI. Although our previous research has made the attempt to use FPGA in EMG pattern recognition and has shown high processing speed in the offline analyses [11], the embedded system presented here is the first complete CPS for the NMI that implements both training and testing modules on one single chip. The New CPS integrates all the necessary interfaces and control algorithms for interacting with the physical system in real-time.

The newly designed NMI was completely built and tested as a working prototype. The prototype was then used to carry out real-time testing experiments on an able-bodied subject for classifying three movement tasks (level-ground walking, stair ascent, and standing) in real-time. The system performance was evaluated to demonstrate the feasibility of a self-contained and high performance real-time NMI for artificial legs. Videos of our experiments on the human subject can be found at http://www.youtube.com/watch?v=KNhihjXProU.

This paper is organized as follows. Next section presents the overall system design. Section 3 describes the detailed implementation of the UIR algorithm. The experimental results are demonstrated in Section 4. We conclude our paper in Section 5.

2. SYSTEM DESIGN

The architecture of designed CPS is shown in Figure 1. The embedded NMI samples input signals from two physical systems--a human neuromuscular system and a mechanical prosthetic leg. The sampled signals are then processed to decipher user's intent to control the prosthesis. The NMI consists of two modules: a data collection module built on an MCU with multiple on-chip ADCs for sensing and buffering input signals, and an FPGA device as the computing engine for fast data decoding and pattern recognition. A serial peripheral interface (SPI) is located between the two devices for transferring digitized input data from the MCU to the FPGA device.

1) Input signals: Multi-channel EMG signals are collected from multiple surface electrodes mounted on patient's residual muscles. Mechanical forces and moments are recorded from a 6 degrees-of-freedom (DOF) load cell mounted on the prosthetic pylon. The EMG signals and the mechanical signals are preprocessed by filters and amplifiers and then simultaneously streamed into the NMI.

2) MCU module: The MCU device does not do any compute-intensive task. It provides multi-channel on-chip ADCs to sample the input signals and convert the analog signals to digital data. The digitized data is then stored in the user-defined result queues allocated in the RAM buffer. In the system RAM, two equal sized result queues are defined. With direct memory access (DMA) support, the MCU core can be insulated from the data acquisition process. Thus these two result queues forms a circular buffer that can continuously receive new data while transmitting old data to the FPGA module for further processing.

3) FPGA module: The FPGA device receives digitized data from the MCU module continuously. In order to fully utilize the computing capacity of the FPGA system and produce dense decisions, the input signals are segmented by overlapped analysis windows with a fixed window length and window increment [4]. The designed FPGA module contains six components: an SPI module that serially receives input signals from the MCU module, a user defined module implementing the UIR algorithm, a high-speed on-chip memory for fast online pattern recognition, an SDRAM controller that interfaces with a large-capacity external SDRAM, parallel IOs for outputting UIR decisions, and a soft processor for managing hardware components and directing data flows. The FPGA module works in two modes: offline training and online pattern recognition. Offline training needs to be performed before using the artificial leg and also whenever a complete re-training is required. During the training procedure, users are instructed to do different movement tasks, and a large amount of data is collected by the NMI to train the classifier. The external SDRAM is only used in the offline training phase to store the training data because FPGAs usually have limited on-chip memory. For online pattern recognition, the input streams are stored in the on-chip memory for fast processing and provided to the classifier for decisions to continuously identify the user's intended movements.

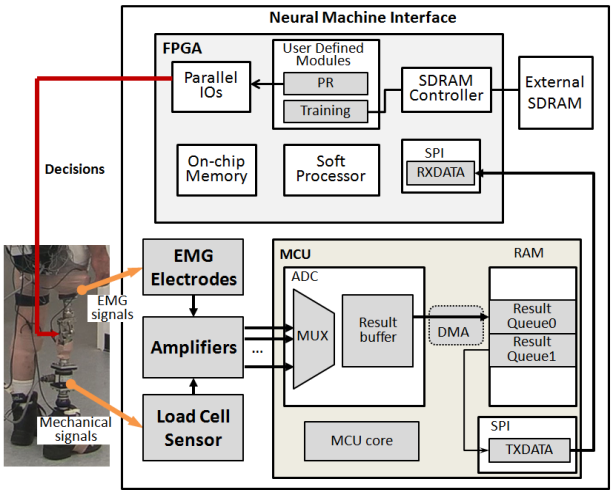

Figure 1. System architecture of the embedded NMI for artificial legs.

3. IMPLEMENTATION OF THE UIR ALGORITHM ON FPGA

3.1 Architecture of the UIR Strategy

The architecture of the UIR strategy based on neuromuscular-mechanical information fusion and phase-dependent pattern recognition (PR) is shown in Figure 2. It is a self-contained architecture that integrates the functions of training and phase-dependent pattern recognition in one embedded system. For every analysis window, features of EMG signals and mechanical signals are extracted from each input channel. A feature vector is formed and normalized by fusing the features from all the input channels. The feature vector is then fed to the classifier for pattern recognition. The phase-dependent classifier consists of a gait phase detector and multiple classifiers. Each classifier is associated with a specific gait phase. During the process of pattern recognition, the gait phase for current analysis window is first determined by the phase detector, and then the corresponding classifier is adopted to do the classification. In this study, four gait phases are defined: initial double limb stance (phase 1), single limb stance (phase 2), terminal double limb stance (phase 3), and swing (phase 4) [9]. The real-time gait phase detection is based on the measurements of the vertical ground reaction force (GRF) sampled from the 6-DOF load cell.

In the real-time embedded system design, to ensure a smooth control of artificial legs, precise timing control is necessary. Figure 3 shows the timing diagram of the control algorithm during the real-time UIR process. In the designed system, the MCU and the FPGA device collaborates to produce a decision at every window increment. While the MCU is sampling data for window $i+1$, the user intent recognition for window i, including the tasks of SPI data transfer, feature extraction, gait phase detection, feature vector formation and normalization, and pattern recognition must be done within the window increment. In other words, the execution time of the UIR algorithm determines the minimum window increment. Larger window increments will introduce longer delay to the NMI decision, which may not be safe to control the prosthesis in real-time. Therefore fast processing speed is very critical to the embedded system design.

3.2 Parallel Implemetations on FPGA

The implementation of the CPS was based on the Altera DE3 education board with a Stratix III 3S150 FPGA device, coupled with the Freescale MPC5566 132 MHz 32 bits MCU evaluation board (EVB) with 40-channel 12-bit on-chip ADCs. The MPC5566 module and the DE3 module are connected with each other via

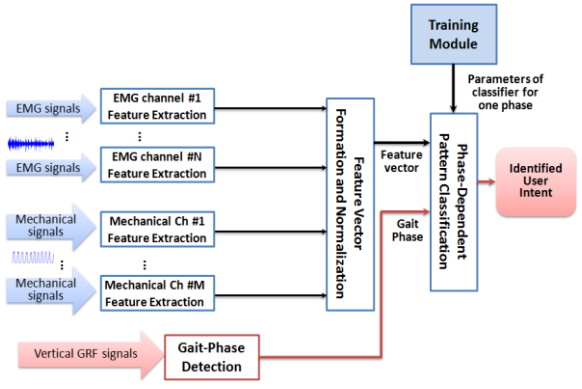

Figure 2. Architecture of UIR strategy based on neuromuscular-mechanical fusion-based phase-dependent pattern recognition.

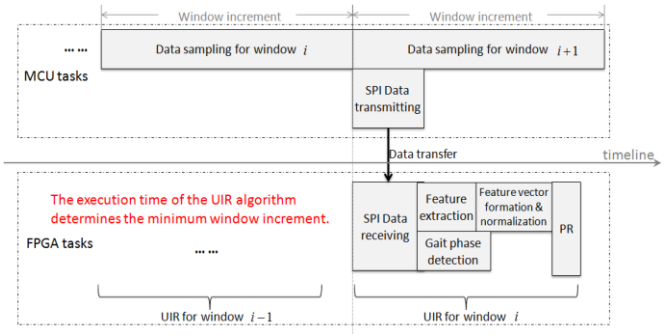

Figure 3. Timing diagram of the control algorithm during online UIR process.

serial peripheral interface (SPI). In this design, DE3 was configured as the SPI master and MPC5566 was the slave. A parallel UIR algorithm tailored to FPGA was designed and implemented on DE3. Fixed-point operations were adopted in this implementation because of their less resource cost and lower latency than floating-point operations. In addition, because the input signals were sampled by ADCs with 12-bit resolution, all the arithmetic operation types in the PR algorithm could be handled by 32-bit fixed-point data formats with careful management.

The UIR algorithm was implemented on the FPGA with the help of a high-level synthesis tool--CoDeveloper from Impulse Accelerated Technologies. The PR algorithm was first developed using C programming language, and then CoDeveloper was used to generate VHDL (VHSIC hardware description language) modules from the C program. The VHDL modules were integrated into the FPGA system as the user defined modules as shown in Figure 1, and worked with other hardware components as a complete NMI. To utilize the parallelism of FPGAs, CoDeveloper provides a multiple process, parallel programming model. In our design, the algorithm was partitioned into a set of processes. These processes can run on the FPGA in parallel if there are no data dependencies. The communications between processes can be done using communication objects, such as streams, signals, and shared memories. Streams are implemented in hardware as dual-port FIFO RAM buffers. A stream connects two concurrent processes (a producer and a consumer), where the producer stores data into and the consumer accesses data from the stream buffer. A single process can be associated with multiple input and output streams. Signals are useful objects to communicate status information among processes. Shared memories are used to store and access large blocks of data from specific external memory locations using block read and block write functions.

1) Feature Extraction: Before offline training or online pattern recognition is performed, features need to be extracted from raw input signals. In every analysis window, four time-domain (TD) features (mean absolute value, number of zero crossings, waveform length, and number of slope sign changes) are extracted from each EMG channel. For the mechanical forces/moments recorded from the 6-DOF load cell, the mean value is calculated as the feature from each individual DOF. The procedure of feature extraction is independent for individual input channel and identical for homogeneous sensors. This property can be utilized to greatly reduce the computation time for feature extraction because all the channels can be processed in parallel. Figure 4 shows the partitioned processes and the data flows of the FPGA implementation of feature extraction. Each white box in the figure represents a small process. The black arrows located between processes are one-way data streams. In this design, $N+6$ parallel threads are generated, where N denotes the number of EMG channels, and the other six threads are assigned for extracting

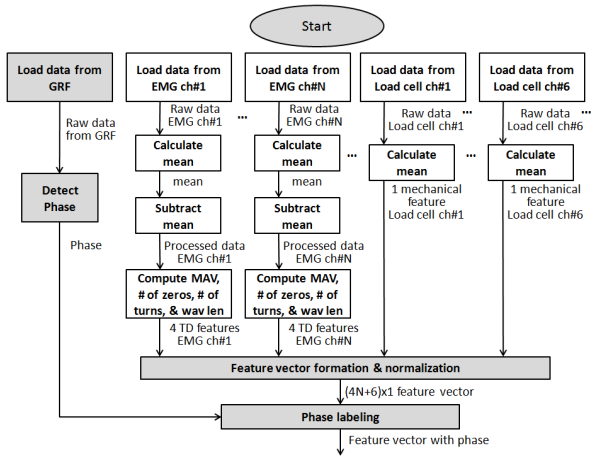

Figure 4. Partitioned processes and data flows of the FPGA implementation of feature extraction.

features from mechanical forces/moments. For each EMG channel, the thread contains four processes: loading raw input data from memory, calculating mean, subtracting mean from the raw data, and extracting four TD features from the processed data. For mechanical forces/moments, each thread fetches raw data from memory and calculates mean as the mechanical feature. After all the features are extracted, the feature streams are sent to the process of feature vector formation and normalization and then fused into a $(4N+6)\times 1$ feature vector. To implement the phase-dependent PR strategy, a thread of gait phase detection loads the vertical GRF measured from the load cell in each analysis window, and then determines current gait phase. This thread is also independent from the threads for feature extraction so that it can run simultaneously with other threads. The detected gait phase is streamed to the phase labeling process, and the feature vector generated in current window is labeled with a specific gait phase. During online pattern recognition, the feature vector with a labeled phase is the input data for pattern classification. In the training procedure, signals are recorded for a period of time under each movement task. Same procedure of feature extraction is performed for every training window. A $(4N+6)\times M_p (p\in[1,4])$ feature matrix is generated as the training data for each gait phase, where M_p is the number of training windows in the p_{th} phase.

2) Pattern Recognition: In this study, linear discriminant analysis (LDA) is adopted for user intent classification because of its computational efficiency for real-time prosthesis control and the comparable accuracy to more complex classifiers [7]. Four gait phases are defined for recognizing user's locomotion mode, giving rise to four LDA-based classifiers. Each classifier is trained for a specific phase. The details of the LDA algorithm can be found in the supplemental material.

Most of the computations involved in the training algorithm are matrix operations. Because a large amount of data need to be processed in the training procedure, the dimensions of the matrices can be very large. Only using on-chip memory is not enough to handle all the computations. External memory with large capacity is required to store the processing data. In our implementation, several external memory buffers are defined to store either large matrices during the training computations or data that might be reused in the online PR phase or the re-training phase. A process is designed to perform a simple task with a small block of data, such as a matrix row/column, and store partial results in the external memory. In this way, the operations of subsequent matrix rows/columns can be efficiently pipelined.

During online pattern recognition, based on the gait phase of current analysis window, the parameters of the corresponding classifier are loaded from memory. The observed feature vector derived from each analysis window is provided to the classifier for intent recognition.

4. PROTOTYPING & EXPERIMENTAL RESULTS

This study was conducted with Institutional Review Board (IRB) approval at our university and informed consent of subjects. To evaluate the performance of the designed NMI, two experiments with different purposes were conducted. First, to evaluate the classification accuracy and the computation speed of the FPGA-based PR algorithm, the performance of the FPGA implementation was compared with our previous software implementation by processing the same dataset offline. Secondly, to evaluate the performance of the entire CPS, a real-time test was carried out on a male able-bodied subject for identifying three movement tasks (level-ground walking, stair ascent, and standing).

4.1 Performance of FPGA vs. CPU

In order to verify the correctness of the FPGA-based PR algorithm and compare the performance of the FPGA design with our previous Matlab implementation, we processed the same dataset on both platforms. The testing dataset was previously collected from a male patient with transfemoral amputation (TF). Seven EMG channels recording signals from the gluteal and thigh muscles and six channels of mechanical forces/moments measured by a 6-DOF load cell were collected in this dataset for identifying three locomotion modes including level-ground walking, stairs ascent, and stairs descent. The dataset was segmented by overlapped analysis windows. The window length and the window increment were set to 160 data points and 20 data points, respectively. The dataset contained 936 analysis windows totally, where 596 of them were used as the training data and the rest 340 windows were testing data. The Matlab implementation was based on a PC with Intel Core i3 3.2 GHz CPU and 6 GB DDR3 SDRAM at 1333 MHz. For the FPGA implementation, a 1GB DDR2-SDRAM SO-DIMM module was plugged into the DDR2 SO-DIMM socket on the DE3 board as the system external memory. The Altera high performance DDR2 SDRAM IP generated one 200 MHz clock as SDRAM's data clock and one half-rate system clock 100 MHz for all other hardware components in the system. The dataset was preloaded into the SDRAM, and the output decisions were printed to the Nios II console [12] for performance evaluation.

It was observed that the classification results of the FPGA system matched very well with the Matlab implementation. Both platforms provided a training accuracy of 98.99% and a testing accuracy of 98.00%. The missed classification points of the two implementations appeared in the same locations. These results clearly demonstrated that the FPGA-based PR algorithm did not lose any computation accuracy as compared to the software implementation.

Table 1 compares the execution time of the LDA-based PR algorithm between the software implementation and the FPGA design. Two configurations with different number of input channels were considered, one with 7 EMG channels and 6 mechanical channels, the other with 12 EMGs and 6 mechanical channels. For the training algorithm that processed 600 analysis windows, the FPGA provided a speedup of around 7X over the software implementation. In the testing phase, the FPGA system took less than 0.3 ms to classify one analysis window. Compared with the Matlab implementation, the FPGA-based PR testing algorithm demonstrated a speedup of 30 times for the configuration of 7

Table 1. Comparison of the execution time of the PR algorithm

	Configuration	FPGA	Matlab	Speedup
Training Algorithm (600 analysis windows)	7 EMGs 6 Mech.	0.46 s	3.2 s	6.96 x
	12 EMGs 6 Mech.	0.64 s	4.7 s	7.34 x
Testing algorithm (classify one analysis window)	7 EMGs 6 Mech.	0.23ms	6.8 ms	29.56 x
	12 EMGs 6 Mech.	0.25 ms	9.5 ms	38.00 x

EMGs and 6 mechanical signals. If more input channels were used (i.e. 12 EMG channels and 6 mechanical channels), a more significant speedup of 38X was observed, which further demonstrated the advantages of FPGA parallelism. From Table 1 we can see that the FPGA implementation of the testing algorithm shows better performance than the training algorithm. This is because the testing algorithm only used fast on-chip memory while the computation complexity of the training algorithm required the FPGA to interact with the external memory. In our experiments it was observed that loading training data from external memory to the FPGA took more than half of the total execution time of the training algorithm. The summary of FPGA resource utilization is listed in Table 2.

4.2 System performance in real-time

The designed NMI prototype was tested on one male able-bodied subject (Figure 5) in real-time. A plastic adaptor was made so that the subject could wear a hydraulic passive knee on the left side. Seven surface EMG electrodes (MA-420-002, Motion Lab System Inc., Baton Rouge, LA) were used to record signals from the gluteal and thigh (or residual thigh) muscles on the subject's left leg. An MA-300 system (Motion Lab System Inc., Baton Rouge, LA) collected seven channels of EMG signals. A ground electrode was placed near the anterior iliac spine of the subject. The mechanical ground reaction forces and moments were measured by a 6-DOF load cell mounted on the prosthetic pylon. The analog EMG signals and mechanical signals were digitally sampled at the rate of 1.1 KHz by the MPC5566 EVB. The intent decisions made by the FPGA device were sent out to 4-bit parallel IO pins on the DE3 board, and displayed by a software GUI. The window length and the window increment were still set to 160 data points and 20 data points, respectively.

Three movement tasks (level-ground walking (W), stair ascent (SA) and standing (ST)) and four mode transitions (ST→W, W→ST, ST→SA and SA→ST) were investigated in this experiment. For the subject's safety, he was allowed to use hand railings and a walking stick. A training session was conducted first to collect the training data for the pattern classification. The

Table 2. Stratix III 3S150 Resource Utilization

Resources	Available	Training 12 EMG 6 Mech.	Training 7 EMG 6 Mech.	Online PR 12 EMG 6 Mech.	Online PR 7 EMG 6 Mech.
Combinational ALUTs	113,600	46%	33%	32%	25%
Memory ALUTs	56,800	3%	2%	3%	2%
Registers	113,600	43%	30%	27%	24%
Block memory bits	5,630,976	16%	12%	16%	12%
DSP blocks	384	72%	44%	27%	24%

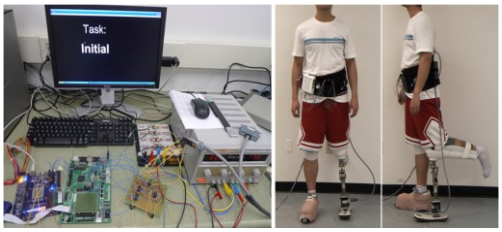

Figure 5. The NMI prototype based on MPC5566 EVB and DE3 education board (left figure) and the experimental setup of the real-time test on a male able-bodied subject (right figure).

subject was instructed to do each movement task for about 10 seconds in one trial. Three trials were collected as the training data. In the real time testing sessions, 10 real-time testing trials were conducted. To evaluate the system performance of real-time intent recognition, we adopted the evaluation criteria as described in our previous study [13]. The testing data were separated into static states and transitional periods. The static state was defined as the state of the subject continuously walking on the same type of terrain (level ground and stair) or performing the same task (standing). A transitional period was the period when subjects switched locomotion modes. The purpose of the UIR system is to predict mode transitions before a critical gait event for safe and smooth switch of prosthesis control mode. In this study the critical timing was defined for each type of transition. For the transitions from standing to locomotion modes (level-ground walking and stair ascent), the critical timing was defined at the beginning of the swing phase (i.e. toe-off). For the transitions from locomotion modes to standing, the critical timing was the beginning of the double stance phase (i.e. heel contact). The real time performance of our embedded system was evaluated by the following parameters.

Classification Accuracy in the Static States: The classification accuracy in the static state is the percentage of correctly classified observations over the total number of observations in the static states.

The Number of Missed Mode Transitions: For the transitions from standing to locomotion modes, the transition period starts one second before the critical timing, and terminates at the end of the single stance phase after the critical timing; for the transitions from locomotion modes to standing, the transition period includes the full stride cycle prior to the critical timing and the period of one second after the critical timing. A transition is missed if no correct transition decision is made within the defined transition period.

Prediction Time of Mode Transitions: The prediction time of a transition in this experiment is defined as the elapsed time from the moment when the decisions of the classifier changes movement mode to the critical timing for the investigated task transitions.

The overall classification accuracy in the static states across 10 testing trials for classifying level-ground walking, stair ascent and standing was 99.31%. For all the 10 trials, no missed mode transitions were observed within the defined transition period. Table 3 lists the average and the standard deviation of the prediction time for four types of transitions. The results show that there was around 104 ms decision delay for the transitions from stair ascent to standing (SA→ST). This is because the subject

Table 3. Prediction Time of Mode Transitions Before Critical Timing

Transition	ST→W	W→ST	ST→SA	SA→ST
Prediction Time (ms)	412.8±76.7	124.39±114.2	549.83±139.2	-104.67±54.1

could not perform foot-over-foot alternating stair climbing with a passive knee joint. In our experiments, the subject climbed stairs by lifting the sound leg on one step and then pulled up the prosthetic leg on the same step, which produced the same pattern as the mode transition from stair ascent to standing. Therefore the transition SA→ST was only able to be recognized after the subject was standing still. This problem will be eliminated by replacing the passive device with a powered knee in the near future. Wearing the powered knee, the prosthesis user is able to climb stairs foot-over-foot, which provides a very different pattern from the transition SA→ST. For the other three types of transitions (ST→W, W→ST, and ST→SA), the user intent for mode transitions can be accurately predicted 104-549 ms before the critical timing for switching the control of prosthesis. Figure 6 shows the real-time system performance for one representative testing trial. The white area in Figure 6 denotes the periods of static states (level-ground walking, stair ascent, and standing), the gray area represents the transitional period, and the black vertical dash line indicates the critical timing for each transition. We can see in this trial all the transitions were correctly recognized within the transitional period. No missed classifications occurred in the static states in this trial. The video of our real-time experiments can be found at http://www.youtube.com/watch?v=KNhihjXProU.

5. CONCLUSIONS

This paper presented the design and implementation of the first complete cyber physical system of neural machine interface for artificial legs. The new CPS implemented both training and testing modules on one single chip, and integrated all the necessary interfaces and control algorithms for identifying the user's intended locomotion modes in real-time. The designed NMI incorporated an MCU for sensing and buffering input EMG signals and mechanical signals, and an FPGA device as the computing engine for fast decoding and pattern recognition. A special parallel processing algorithm for UIR was designed and implemented that realized the neuromuscular-mechanical fusion based PR algorithm coupled with the real-time controlling algorithm on the FPGA. The FPGA implementation of the PR algorithm achieved a speedup of 7X over the Matlab implementation for the training phase, and a speedup of more than 30X for the testing phase with no sacrifice of computation accuracy. The designed NMI prototype was tested on an able-bodied subject for accurately classifying multiple movement tasks (level-ground walking, stair ascent, and standing) in real-time. The results demonstrated the feasibility of a self-contained and high performance real-time NMI for artificial legs. Our future work includes real-time testing of the designed NMI system on amputee subjects, using the NMI system to control powered prosthetic legs, studying management of power consumption, and increasing the system reliability.

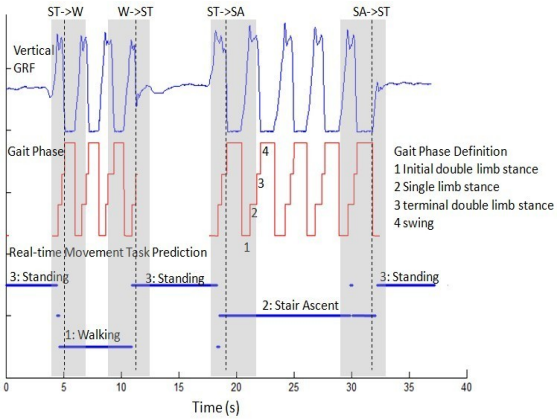

Figure 6. Real-time system performance for one representative testing trial. The white area denotes the periods of static states (level-ground walking, stair ascent, and standing); the gray area represents the transitional period; the black vertical dash line indicates the critical timing for each transition.

6. ACKNOWLEDGMENTS

This work is partly supported by National Science Foundation NSF/CPS #0931820, NIH #RHD064968A, NSF #1149385, NSF/CCF #1017177, and NSF/CCF #0811333. The authors thank Fan Zhang, Quan Ding, Ding Wang, Lin Du, and Ming Liu at the University of Rhode Island, for their suggestion and assistance in this study.

7. REFERENCES

[1] Oonishi, Y., Oh, S., and Hori, Y., "A New Control Method for Power-Assisted Wheelchair Based on the Surface Myoelectric Signal," Industrial Electronics, IEEE Transactions on, vol. 57, pp. 3191-3196, 2010.

[2] Tonin, L., Carlson, T., Leeb, R., and Millán, J. R., "Brain-Controlled Telepresence Robot by Motor-Disabled People," in 33rd Annual International Conference of the IEEE EMBS, Boston, MA, 2011.

[3] Parker, P. and Scott, R., "Myoelectric control of prostheses," Critical reviews in biomedical engineering, vol. 13, p. 283, 1986.

[4] Englehart, K. and Hudgins, B., "A robust, real-time control scheme for multifunction myoelectric control," IEEE Trans Biomed Eng, vol. 50, pp. 848-54, Jul 2003.

[5] Lin, C. T., Ko, L. W., Chiou, J. C., Duann, J. R., Huang, R. S., Liang, S. F., Chiu, T. W., and Jung, T. P., "Noninvasive neural prostheses using mobile and wireless EEG," Proceedings of the IEEE, vol. 96, pp. 1167-1183, 2008.

[6] Kuiken, T., "Targeted reinnervation for improved prosthetic function," Phys Med Rehabil Clin N Am, vol. 17, pp. 1-13, Feb 2006.

[7] Huang, H., Zhou, P., Li, G., and Kuiken, T. A., "An analysis of EMG electrode configuration for targeted muscle reinnervation based neural machine interface," IEEE Trans Neural Syst Rehabil Eng, vol. 16, pp. 37-45, Feb 2008.

[8] Huang, H., Kuiken, T. A., and Lipschutz, R. D., "A strategy for identifying locomotion modes using surface electromyography," IEEE Trans Biomed Eng, vol. 56, pp. 65-73, Jan 2009.

[9] Zhang, F., DiSanto, W., Ren, J., Dou, Z., Yang, Q., and Huang, H., "A Novel CPS System for Evaluating a Neural-Machine Interface for Artificial Legs," 2011, pp. 67-76.

[10] Zhang, X., Liu, Y., Zhang, F., Ren, J., Sun, Y. L., Yang, Q., and Huang, H., "On Design and Implementation of Neural-Machine Interface for Artificial Legs," IEEE Transactions on Industrial Informatics, 2011.

[11] Zhang, X., Huang, H., and Yang, Q., "Design and Implementation of A Special Purpose Embedded System for Neural Machine Interface," in ICCD'2010, Amsterdam, the Netherlands, October 2010.

[12] Altera. Nios II Processor: The World's Most Versatile Embedded Processor. Available: http://www.altera.com/products/ip/processors/nios2/ni2-index.html

[13] Huang, H., Zhang, F., Hargrove, L., Dou, Z., Rogers, D., and Englehart, K., "Continuous Locomotion Mode Identification for Prosthetic Legs based on Neuromuscular-Mechanical Fusion," Biomedical Engineering, IEEE Transactions on, pp. 1-1, 2011.

Supplemental Material

S1. PATTERN RECOGNITION USING LINEAR DISCRIMINANT ANALYSIS

The principle of the LDA-based PR strategy is to find a linear combination of features which separates multiple locomotion classes $C_g (g \in [1, G])$. G denotes the total number of classes. Suppose μ_g is the mean vector of class C_g and every class shares a common covariance matrix $\sum$, the linear discriminant function is defined as

$$d_{C_g} = \bar{f}^T \Sigma^{-1} \mu_g - \frac{1}{2} \mu_g{}^T \Sigma^{-1} \mu_g . \quad (1)$$

During the training procedure, $\sum$ and μ_g are estimated based on the feature matrix calculated from the training data. The estimations of $\sum$ and μ_g are expressed as

$$\widetilde{\Sigma} = \frac{1}{G} \sum_{g=1}^{G} \frac{1}{K_g - 1} (F_g - Mi_g)(F_g - Mi_g)^T$$

and

$$\widetilde{\mu}_g = \frac{1}{K_g} \sum_{k=1}^{K_g} \bar{f}_{C_g,k}$$

where K_g is the number of analysis windows in class C_g; $\bar{f}_{C_g,k}$ is the k_{th} observed feature vector in class C_g; $F_g = [\bar{f}_{C_g,1}, \bar{f}_{C_g,2}, ..., \bar{f}_{C_g,k}, ..., \bar{f}_{C_g,K_g}]$ is the feature matrix of class C_g; $Mi_g = [\widetilde{\mu}_g, \widetilde{\mu}_g, ..., \widetilde{\mu}_g]$ is the mean matrix that has the same number of columns as in F_g. The results of the LDA training procedure can be represented by a weight matrix as $W = [\overline{w}_1, \overline{w}_2, ..., \overline{w}_g, ..., \overline{w}_G]$ and a weight vector as $\bar{c} = [c_1, c_2, ..., c_g, ..., c_G]$. Here

$$\overline{w}_g = \widetilde{\Sigma}^{-1} \widetilde{\mu}_g \quad (2)$$

and

$$c_g = -\frac{1}{2} \widetilde{\mu}_g{}^T \widetilde{\Sigma}^{-1} \widetilde{\mu}_g . \quad (3)$$

Therefore (1) can be estimated as

$$\widetilde{d}_{C_g} = \bar{f}^T \overline{w}_g + c_g . \quad (4)$$

The major task of the training procedure is to calculate the mean vector $\widetilde{\mu}_g$ for each class, the common covariance matrix $\widetilde{\Sigma}$, and its inverse matrix $\widetilde{\Sigma}^{-1}$. In practice, matrix inversion is a compute-intensive and time consuming task, which should be avoided if possible. From (2) and (3), it can be found that $\widetilde{\Sigma}^{-1}$ does not appear alone. If $\widetilde{\Sigma}^{-1} \widetilde{\mu}_g$ can be calculated in an efficient way, W and $\bar{c}$ can be achieved easily. In our implementation, a more efficient algorithm was adopted to solve this problem. First, a Cholesky decomposition is performed as $\widetilde{\Sigma} = R^T \times R$, where R is upper triangular. Then $\widetilde{\Sigma}^{-1} \widetilde{\mu}_g$ can be quickly computed with a forward substitution algorithm for a lower triangular matrix R^T, followed by a back substitution algorithm for an upper triangular matrix R. The condition of a successful Cholesky decomposition is that $\widetilde{\Sigma}$ must be symmetric and has real positive diagonal elements, which can be perfectly satisfied by a covariance matrix. In this way, (2) and (3) can be reformulated as $\overline{w}_g = R \setminus (R^T \setminus \widetilde{\mu}_g)$ and $c_g = -\frac{1}{2} \widetilde{\mu}_g{}^T \overline{w}_g$.

During the testing phase, the observed feature vector $\bar{f}$ derived from each analysis window is applied to calculate $\widetilde{d}_{C_g}$ in (4) for each movement class and is classified into a class $\widetilde{C}_g$ that satisfies

$$\widetilde{C}_g = \arg\max C_g \{\widetilde{d}_{C_g}\}, C_g \in \{C_1, C_2, ..., C_G\} .$$