34th Annual International Conference of the IEEE EMBS
San Diego, California USA, 28 August - 1 September, 2012

Promise of a Low Power Mobile CPU based Embedded System in Artificial Leg Control

Robert Hernandez, *Student member, IEEE*, Fan Zhang, *Student member*, *IEEE,*
Xiaorong Zhang, *Student member*, *IEEE*, He Huang, *Member*, *IEEE,* Qing Yang, *Fellow, IEEE*

***Abstract*—This paper presents the design and implementation of a low power embedded system using mobile processor technology (Intel Atom™ Z530 Processor) specifically tailored for a neural-machine interface (NMI) for artificial limbs. This embedded system effectively performs our previously developed NMI algorithm based on neuromuscular-mechanical fusion and phase-dependent pattern classification. The analysis shows that NMI embedded system can meet real-time constraints with high accuracies for recognizing the user's locomotion mode. Our implementation utilizes the mobile processor efficiently to allow a power consumption of 2.2 watts and low CPU utilization (less than 4.3%) while executing the complex NMI algorithm. Our experiments have shown that the highly optimized C program implementation on the embedded system has superb advantages over existing PC implementations on MATLAB. The study results suggest that mobile-CPU-based embedded system is promising for implementing advanced control for powered lower limb prostheses.**

***Index Terms*—Neural-machine interface, real-time system, embedded system, support vector machine, control of artificial limbs.**

I. Introduction

A neural-machine interface (NMI) based on neuromuscular-mechanical fusion [1] and phase-dependent pattern recognition (PR) strategy [2] has been successfully developed in our research group to identify user intent for volitional control of powered lower limb prostheses. Embedded implementation of this complex NMI algorithm for real-time operation is essential for lower limb prostheses, but is challenging due to the rigorous system requirements. First, the prosthesis control must be accurate and responsive to enable lower limb amputees to perform different tasks safely and intuitively. In addition, the prosthesis control system must perform continuously for 6-8 hours daily without interruption. Finally, the system must be easily integrated into the prosthetic limb. These requirements demand the embedded system to be computational powerful, low power, and small in size.

In our previous study, Field Programmable Gate Arrays (FPGAs) have been used as the embedded system to implement our designed NMI with Linear Discriminant Analysis (LDA)-based classifiers [3]. The prototype demonstrated promising performance for real-time NMI implementation. Although extremely effective, FPGAs pose many challenges during the design stage, such as language syntax, design environment, and toolsets [4]. Another concern with the use of FPGAs is its requirement of special purpose hardware design and fabrication giving rise to high cost. For example, a Support Vector Machine (SVM)-based classifier improved the accuracy of NMI for intent recognition compared to LDA [1]. However, hardware programming the complex SVM algorithm on a FPGA is challenging and time consuming. These difficulties limit our capability to further optimize and develop the NMI for neural control of powered lower limb prostheses.

With the wide availability of commodity off-the-shelf hardware such as Personal Computers (PCs), an efficient and cost-effective way of implementing our NMI is to develop an NMI program specifically tailored to such Commercial of the Shelf (COTS) hardware. Existing PC implementations of our SVM-based NMI algorithms, however, are mainly based on MATLAB giving rise to high overheads and poor real-time performance. Our objective here is to develop a C program realizing our NMI algorithm on a commodity PC that is portable and fast enough.

One alternative to FPGA and regular CPU is a mobile CPU. Mobile CPUs are low cost, low power, and much smaller devices than regular CPUs (as shown in Fig. 1 [5]). In addition, they have the capability to provide the flexible design environment as a PC/CPU combination. However, the computational power of mobile CPUs, such as the Intel Atom™ Z530, is relatively low [6,7]. Therefore, in this study, we are interested to investigate whether or not a mobile CPU can execute a highly computational intensive algorithm, such as our phase-dependent, SVM-based NMI for powered lower limb prostheses.

This paper makes the following contributions:

- Design and implementation of a NMI for artificial legs based on mobile processors;
- Design and implementation of a highly optimized, C-based, embedded application tailored to execute a phase-dependent NMI with SVM classifiers;
- A performance analysis that evaluates the potential of mobile processors for embedded implementation of a NMI for neural control of powered lower limb prosthesis.

II. System Design

A. Hardware Architecture

To provide viable use capability of a NMI, the NMI must be small, dissipate low power, and be fast enough to execute the classification algorithm in real-time. To meet these requirements, the AxiomTek eBOX530-820-FL fanless

*This research was supported in part by the Department of the Navy (Naval Undersea Warfare Center, Newport, Rhode Island) and NSF/CPS #0931820, NSF#1149385, NIH #RHD064968A, NSF/CCF #0811333 and NSF/CCF #1017177.

The authors are with the Department of Electrical, Computer and Biomedical Engineering, University of Rhode Island, Kingston, RI 02881, USA

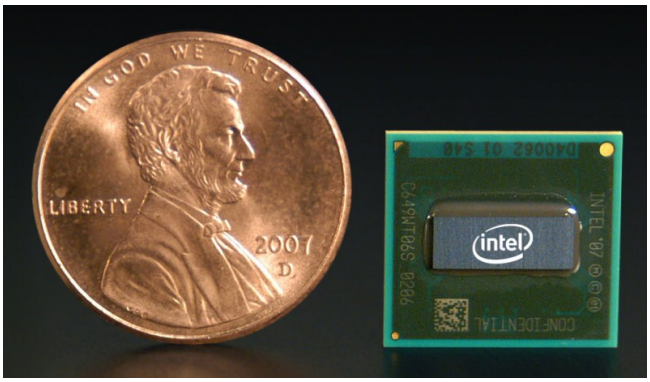

Figure 1. Intel Atom™ mobile CPU size compared to a United States penny (a United States penny is approximately 19.05 millimeters in diameter)

embedded hardware with the Intel Atom™ Processor Z530 (512K cache, 1.6 GHz) was chosen [8]. The Intel Atom™ Processor Z530 provided the highest performance and lowest power dissipation of Hyper-Threading capable mobile CPUs, which is ideal for thermally constrained and fanless embedded applications [9,10]. The Hyper-Threading technology allows the operating system and the NMI application to execute simultaneously on two Hyper-Threads as they would on two physical processors [11]. This minimizes the impacts of the OS execution on the real time embedded NMI application.

B. Software Architecture

C was chosen as the software language in our study because of its superior performance for real-time embedded applications [12-15]. To enhance the system performance, several programming techniques were used in the design and implementation of the application. First, dynamic memory management is one of the most expensive operations in C applications [16], which may cost 30% of the total execution time for the heap intensive C applications [16]. To avoid this problem, the various data structures within the software were defined statically with pre-defined maximum sizes. Secondly, to increase the reliability of the application, the data structures were placed in the application's data segment, not in the application's stack [17], to help avoid stack overflows. Other performance enhancements implemented included loop unwinding [18] and inline function expansion [19]. Loop unwinding is an efficient means to increase the utilization of pipelines and helps eliminate loop overhead [18]. Inline function expansion replaces a function call with the body of the function, which reduces the overhead associated with a function call during program execution [19].

The designed Neuromuscular-Mechanical fusion PR algorithm, utilizes SVM classification. The open source library LIBSVM [20] was used and specifically tailored to our embedded NMI application for real-time SVM classification. LIBSVM was also utilized in our previous MATLAB implementation, which served as a baseline for accuracy determination of the embedded application.

III. Pattern Recognition Algorithm

The previously developed NMI identifies the user's locomotion mode based on electromyographic (EMG) signals recorded from the residual thigh muscles and mechanical forces/moments signals recorded from prosthetic pylon. These EMG and mechanical data are segmented by the sliding analysis windows. Features are extracted from the raw EMG and mechanical data in each analysis window and fused into one feature vector. This feature vector is sent to a phase-dependant pattern classifier for determination of user intent. The phase-dependant pattern classifier consists of multiple sub-classifiers for individual defined gait phases and a gait phase detector that identifies current gait phase and switches the corresponding sub-classifier on. Detailed description of this previously designed NMI can be found in [1] and [2].

A. Feature Extraction

In this study, four time-domain (TD) features (the mean absolute value, the number of zero crossings, the waveform length, and the number of slope sign changes) were extracted from EMG signals in each analysis window. For mechanical measurements, the mean, minimum, and maximum values in each analysis window were extracted as the features. More detailed information can be found in [1]. The length of sliding analysis window and window increment were 150 ms and 50ms, respectively.

The features and increments were chosen to match our previous MATLAB implementations [21], thereby providing a baseline for an accuracy comparison with the newly designed embedded application.

B. Phase Dependant Pattern Recognition

To accurately determine user intent, SVM utilizing a Radial Basis Function (RBF) kernel [21] was utilized. The SVM gamma parameter of 0.015 was used.

In the designed phase-dependant classifier, four sub-classifiers were defined corresponding to the following four gait phases: initial double limb stance (phase 1), single limb stance (phase 2), terminal double limb stance (phase 3), and swing (phase 4) [21]. The gait phase detector detects these gait phases based on the vertical Ground Reaction Force (GRF). In order to build the parameters in the classifiers, training procedure must be conducted on a training data set. During training, the output of phase detector is used to label the training data with the corresponding gait phase. Each classifier is trained only with the data pertinent for its gait phase. When testing the classification, the gait phase detector determines which classifier is responsible for the determination of user intent. The algorithmic data flow of the phase-dependant pattern recognition is shown in Fig. 2.

C. Software Implementation

To implement the Neuromuscular-Mechanical Fusion PR, three applications were developed. The first application accepts offline raw training data, performs the EMG and mechanical feature extraction, fuses and then normalizes the features into vectors. The feature vectors are then separated into their corresponding gait phases and provided to the training application. The first application is also responsible for generating the normalization parameters required by the PR to normalize the testing data, when determining user intent. The second application accepts the four sets of training vectors and generates four SVM models, one model for each gait phase. The third application accepts raw offline testing data, the four gait phase SVM models, and the

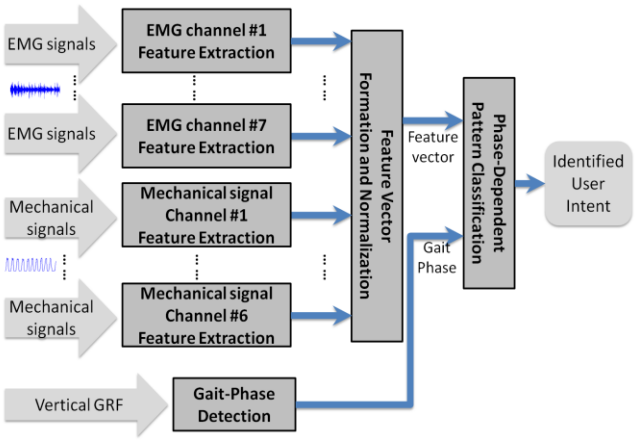

Figure 2. Phase-dependant PR algorithmic data flow

normalization parameters. The application extracts EMG and mechanical features from the raw testing data. The features are then fused and normalized, with the provided normalization parameters, into a vector. Finally, the application determines the current gait phase, and forwards the test vector to the respective phase based classifier for determination of user intent. The software implementation data flow is shown in Fig. 3.

IV. PEFORMANCE EVALUATION

This study was conducted with approval of Institutional Review Board (IRB) at the University of Rhode Island and informed consent of the subject. The evaluation was performed offline on the data collected from a male subject with a transfemoral amputation. The collected data included the EMG signals from the subject's residual thigh muscles and mechanical forces/moments measured by a 6 degree-of-freedom load cell mounted on the prosthetic pylon. The monitored residual muscles included the *rectus femoris* (RF), *vastus lateralis* (VL), *vastus medialis* (VM), *biceps femoris long head* (BFL), *semitendinosus* (SEM), *biceps femoris short head* (BFS), and *adductor magnus* (ADM). The recognition accuracy of NMI by using the designed embedded system was compared with the results of existing PC implementations on MATLAB. In addition, the timing and processor loading of the application's execution on the embedded hardware were evaluated. A power consumption comparison between similar proposed NMI embedded systems and this embedded system was provided.

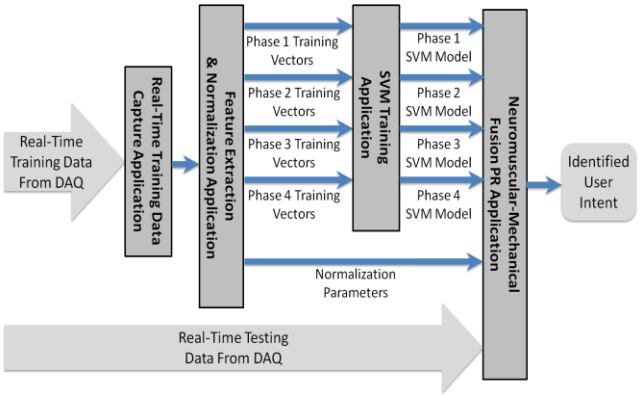

Figure 3. Software implementation data flow

A. Recognition Accuracy of NMI

The offline data was composed of seven different classes: level-ground walking, ramp ascent, ramp descent, stair ascent, stair descent, sitting, and standing. The comparison of recognition accuracies of the NMI by using the designed embedded system and existing PC implementations on MATLAB are provided in Table I. This study utilized a slightly different value for the gamma parameter required by the SVM classifiers. The different gamma value was shown to provide a slightly higher accuracy during testing. This is noticeable in the comparison results, whereby the embedded application slightly outperformed the MATLAB model in PR accuracies.

Both the MATLAB results and the embedded application had lower Phase 4 (swing) accuracies. Two explanations for this result are provided in [22]. The first is that there is little force/moment data present during the swing phase from the prosthetic pylon [22]. The second explanation is related to the swing phase being longer than any of the other three phases, leading to larger variations in the EMG features [22].

B. Execution Timing and Processor Loading on the Embedded Hardware

This previously designed NMI algorithm was executed on the Intel Atom™ based embedded hardware and the performance results were evaluated. A total of 3555 predictions were produced by the Intel Atom™ based embedded hardware. For the purpose of this evaluation, the prediction time will be defined as the total time to execute feature extraction, normalization, gait phase detection and classification for a single analysis window. The mean prediction time was 0.8455 milliseconds with a standard deviation of 0.1044 milliseconds. The worst case prediction executed in 2.1265 milliseconds. These results clearly show that the embedded system is capable of real-time implementation at 50ms and 20ms window increments. If the embedded system is combined with a highly responsive Data Acquisition (DAQ) system to provide the EMG and mechanical data, even a window increment of 10ms may be feasible. At the 10ms window increment, the interface to the DAQ and the DAQ system drivers will become of the utmost importance.

Because there is additional loading on the CPU to execute the data logging for post analysis, the CPU loading provided by the operating system may be inaccurate. Therefore the mean and maximum value of CPU loading was calculated by (1) which were 1.691% and 4.253% respectively.

$$CPU\ Loading = \frac{Prediction\ Time}{Window\ Increment\ (50ms)} * 100 \qquad (1)$$

TABLE I. MATLAB and embedded software classification accuracies

CLASSIFIER	MATLAB MODEL	EMBEDDED SW
PHASE 1	97.74%	98.33%
PHASE 2	96.72%	98.82%
PHASE 3	98.63%	98.67%
PHASE 4	95.18%	95.66%

C. Power Consumption Comparison

Previous studies have utilized Field Programmable Gate Arrays (FPGA) and PCs for similar NMI applications [23]. The reported power consumption for the FPGA was 3.499 watts and the AMD Turion 64x2 CPU within [23] can utilize up to 35 watts [23]. The Intel Atom™ Z530 Processor utilized in this embedded system design dissipates 2.2 watts [9]. The Intel Atom™ CPU's power dissipation is less than one-fifteenth that of the CPU and less than two third that of the FPGA.

V. CONCLUSIONS

This paper presented the design and implementation of a mobile CPU based embedded system for a NMI for artificial leg control. The performance evaluation showed that the highly optimized C-based embedded application combined with the mobile-CPU-based embedded hardware, can easily meet real-time constraints. The performance evaluation also shows that there is no loss in classification accuracy, when compared with the MATLAB model [21]. In fact, there is a slight increase due to the use of a different SVM gamma parameter. Lastly, the CPU utilized for this embedded system dissipated less power than other systems designed for similar applications. Future work to be performed includes interfacing the embedded system to a DAQ to create a real-time capable system and testing the system on lower limb amputees.

REFERENCES

[1] 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. 2867-75, 2011.

[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] X. Zhang, Q. Yang and H. Huang, "A Neural-Controlled Cyber Physical System for Intent Recognition for Artificial Legs," presented at Design Automation Conference, San Francisco, 2012 (Accepted).

[4] I. Gonzalez, E. El-Araby, P. Saha, T. El-Ghazawi, H. Simmler, S. Merchant, B. Holland, C. Reardon, A. George, H. Lam, G. Stitt, N. Alam, M. Smith, "Classification of application development for FPGA-based systems," *Conf Proc National Aerospace Electronics Conference,* 2008

[5] J. Mahoney. (2008, July) "Intel CEO: Atom Platform Something 'Most of Us Wouldn't Use'" [online] Available: http://gizmodo.com/5026401/intel-ceo-atom-platform-something-most-of-us-wouldnt-use [March 22, 2012]

[6] PassMark Software. (2012, March) "PassMark CPU Benchmarks - Low End CPU's," [online] Avaliable: http://www.cpubenchmark.net/low_end_cpus.html [March 22, 2012]

[7] B. Crothers. (2008, April) "New Intel design may spur (more) tiny PCs," [online] Avaliable: http://news.cnet.com/8301-13924_3-9928126-64.html [March 22, 2012

[8] AxiomTek Corporation. (2012). "*Fanless Embedded System with Intel® Atom™ Processor*" [online]. Available: http://axiomtek.com/Download/Spec/ebox530-820-fl.pdf [March 19, 2012]

[9] Intel Corporation. (2010, June). "*Intel® Atom™ Processor Z5xx Series Datasheet*" [online]. Available: http://www.intel.com/content/www/us/en/processors/atom/atom-z540-z530-z520-z510-z500-45-nm-technology-datasheet.html [March 19, 2012]

[10] Intel Corporation. (2011, April). "*Intel® Atom™ Processor Z6xx Series Datasheet*" [online]. Available: http://www.intel.com/content/www/us/en/processors/atom/atom-z6xx-datasheet.html [March 19, 2012]

[11] D. Marr, F. Binns, D. Hill, G. Hinton, D. Kaoufaty, J. Moller, M. Upton "Hyper-Threading technology architecture and microarchitecture," *Intel Technology Journal*, vol. 6 no. 1, pp 4-15, 2002

[12] M. Weiskirchne. (2003, September). *Comparison of execution times of Ada, C and Java*. EADS Deutschland GmbH Military Aircraft. Muenchen, Germany. [online] Available: http://www.aicas.com/info/EADS_benchmark_languare_comparison.pdf [March 19, 2012]

[13] P. Sestoft. (2010, Feburary). *Numeric performance in C, C# and Java*. University of Copenhagen. Copenhagen, Denmark. [online] Available: http://www.itu.dk/~sestoft/papers/numericperformance.pdf [March 19, 2012]

[14] Cherrystone Software Labs. (2010, August). *Algorithmic performance comparison between C, C++, Java and C# programming languages*. Cherrystone Software Labs. Boston, MA [online] Available: http://www.cherrystonesoftware.com/doc/AlgorithmicPerformance.pdf [March 19, 2012]

[15] P. Ambika, H. Ahmed. (2001). *A performance analysis of Java and C*, University of Columbia, New York, NY, Avaliable: http://www.cs.columbia.edu/~sedwards/classes/2001/w4995-02/litsurveys/haseeb-ambika.pdf [March 19, 2012]

[16] D. Tiwari, S. Lee, J. Tuck, Y. Solihin. "Exploiting fine-grained parallelism in dynamic memory management," *IPDPS,* 2010

[17] L. Ferres. (2010). *Memory management in C: The heap and the stack.* Universidad de Concepcion, Concepcion, Chile. [online] Avaliable: http://www.inf.udec.cl/~leo/teoX.pdf [March 19, 2012]

[18] A. Nicolau, "Loop quantization: unwinding for fine-grain parallelism exploitation," Cornell Univerisy, 1985, Avaliable: http://ecommons.library.cornell.edu/bitstream/1813/6549/1/85-709.pdf [March 19, 2012]

[19] W. W. Hwu, P. P. Chang, "Inline function expansion for compiling C programs," *ACM SIGPLAN '89 Conference on Programming Language Design and Implentation*, Portland, Oregon, June 1989

[20] C. C. Chang, C. J. Lin, "LIBSVM: a library for support vector machnies," *ACM Transactions on Intelligent Systems and Technology*, vol. 2 issue 3, pp. 27:1-27:27, 2011

[21] F. Zhang, H. Huang, "Real-Time Recognition Of User Intent For Neural Control Of Artificial Legs,"*MEC'11*, New Brunswick, Fredericton, NB Canada, August 2011.

[22] H. Huang, F. Zhang, L. J. Hargrove, Z. Dou, D. Rogers, and K. B. Englehart, "Continuous locomotion-mode identification for prostethic legs based on neuromuscular-fusion," *IEEE Trans Biomed Eng*, vol. 58 issue 10, pp. 2867-2875, October 2011

[23] X. Zhang, H. Huang, Q. Yang, "Design and implentation of a special purpose embedded system for neural machine interface," *Conf Proc IEEE International Conference on Computer Design,* 2010, pp. 166-72.