FPGA-based Acceleration of High Density Myoelectric Signal Processing

Alexander Boschmann, Andreas Agne, Linus Witschen, Georg Thombansen, Florian Kraus, Marco Platzner
Department of Computer Science, University of Paderborn, 33098 Paderborn, Germany
Email: { alexander.boschmann — andreas.agne — witschen — thombang — fkraus — platzner }@uni-paderborn.de

***Abstract*—In recent years, advances in electromyographic (EMG) sensor technology and machine learning algorithms have led to an increased research effort into high density EMG based pattern recognition methods for prosthesis control. With the goal set on an autonomous multi-movement prosthesis that is capable of performing training and classification of an amputee's EMG signals, the focus of this paper lies in the acceleration of the embedded signal processing chain. Using the Xilinx Zynq as a low-cost off-the-shelf reconfigurable processing platform, we present a solution that is able to compute prosthesis control signals from multi-channel EMG input with up to 256 channels with a maximum processing delay of less than a single millisecond. While the presented system is able to perform training as well as classification, most of our efforts were focused on the acceleration of the feature extraction units, achieving a speed-up of 6.7 for feature extraction alone, and 4.8 for the total processing time as compared to a software only solution.**

I. Introduction

Modern myoelectric upper-limb prostheses enable the user to perform activities of daily living, restore independence and increase quality of life. The use of electromyography (EMG) has been studied as a control source for active prostheses for decades. EMG signals are physiological signals generated during muscular contraction and can be measured as electrical potentials on the skin surface above the remaining muscles. Common myoelectric prostheses use measures like mean absolute value of the EMG amplitude [1] or the signal's rate of change [2] at each electrode site and map them to proportionally control one degree of freedom (DOF). Selecting a different prosthesis function is often achieved by co-contraction of muscles or utilizing hardware switches [3] and is often cited as cumbersome and counter-intuitive [4].

An active area of research are pattern recognition-based control schemes, potentially enabling the user to control multiple DOF intuitively. They operate on the assumption that a set of features extracted from the EMG signal is repeatable for a specific muscle contraction and distinguishable from a set of features taken from different contractions. In literature, the pattern recognition processing chain is often broken down to four components: data preprocessing, data windowing, feature extraction and classification [5]. Common preprocessing steps are removal of power-line harmonics or electrode movement artefacts. Due to its stochastic nature, raw EMG signals are not suitable as input to pattern recognition-based classifiers. Instead, descriptive features are extracted from overlapping data window segments to increase information density. For this reason, the usage of linear discriminant analysis (LDA) in combination with time-domain (TD) features is a popular choice in recent literature. Due to its algorithmic simplicity and low consumption of resources, this combination is predestined for implementation on embedded hardware.

Despite many successfully conducted studies under laboratory conditions, pattern recognition-based control still lacks robustness in a real-world setup [5]. Effects like electrode shifts following donning and doffing [6] or variations in limb position [7] decrease accuracy and usability significantly. These effects have been studied but remain an open problem. User acceptance strongly depends on the pattern recognition method's robustness against non-stationary conditions.

Research on myoelectric control has focused on a small number of EMG channels for low computational delay and better clinical applicability. Due to advances in EMG sensor technology, high density EMG (HD EMG) sensor arrays are available today, offering hundreds of signal sources. Current desktop HD EMG amplifiers can already acquire up to 256 channels, an updated version with up to 384 channels is announced [8]. It has been shown that increasing the number of input channels can improve robustness in non-stationary conditions for traditional EMG signal classification [9] or combinations of HD EMG-based features and classifiers like experimental variogram [10] or structural similarity [11]. However, processing hundreds of EMG channels poses a challenge for an embedded system. Microprocessors used in current prostheses are not suitable for HD EMG data processing in real-time.

The main objective of our work is implementing a controller for autonomous multi-movement prostheses capable of classifying HD EMG signals. Our application scenario dictates that a prosthesis controller must be an embedded system able to perform the computation autonomously inside the prosthesis. Since classification performance strongly depends on a small controller delay, off-loading the computation to external devices like smartphones over wireless communication channels is unsuitable. Furthermore, embedded prosthesis controllers must also be small in size and show a moderate energy consumption because they are operated by batteries which are typically recharged once a day. These characteristics make HD EMG prosthesis controllers an appealing application domain for FPGAs. Since the classification algorithm's runtime is dominated by a feature extraction process applied to indepen-

978-1-4673-9406-2/15/$31.00

dent signal sources, it can potentially be efficiently performed in parallel on an FPGA. Another argument for an FPGA-based approach over a CPU-based one is the potentially better energy-efficiency for the signal processing tasks. Compared to an ASIC solution, the FPGA approach features the required reprogrammability. In particular, the inherently different working modes of the controller (training the classifier and classification of unknown data) could even be a use case for partial reconfiguration.

The novel contribution of this work is an architecture for accelerating the classification chain for HD EMG data. While the classification chain consists of a well-studied combination of TD feature extraction and LDA classification, we present a Xilinx Zynq-based system capable of processing up to 256 channels with a maximum processing delay of less than one millisecond, which to our knowledge is the fastest embedded HD EMG implementation to date.

The remainder of this work is structured as follows. In Section II we give background information about EMG signal processing, ReconOS and the Zynq platform. In Section III we discuss related work, in particular other approaches to real-time EMG processing. In Section IV we describe the architecture of our proposed system. In Section V we validate our system in a real world online experiment with an amputee. Section VI contains experimental results and finally we draw a conclusion in Section VII.

II. Background

In this section, we first give relevant background information on the widely-used windowed approach to real-time EMG signal processing and its implications on the controller delay. Then, we describe the features we compute and the subsequent pattern learning and classification process. Afterwards, we give a short outline on the ReconOS multi-threaded approach to hardware acceleration and the Zynq system on chip.

A. Real-time EMG signal processing

The EMG signal's stochastic nature makes it necessary to extract a set of descriptive features from an analysis window. In order to maximize the amount of classifier decisions over time, an overlapping window approach proposed by Englehart [12] is often implemented. Figure 1 illustrates the approach.

Features are extracted from the analysis window T_a. Subsequently, the system uses the signal processing time T_d to perform the pattern recognition algorithm. In order to remove misclassifications in the decision stream, the classifier decision is added to a decision queue with the last n decisions. The majority vote winner is then passed to the movement controller. To make the decision stream as dense as possible, a new analysis window is calculated each T_{inc} period. Evidently, T_d has to be smaller or equal to T_{inc}.

Farrell and Weir [13] define the delay between the user's change of movement and the first correct controller output as controller delay D, as shown in Figure 1 (grey area). D is influenced by T_a, T_{inc}, T_d and n:

$$D = \frac{1}{2} \cdot T_a + \left(\frac{n+1}{2}\right) \cdot T_{inc} + T_d \tag{1}$$

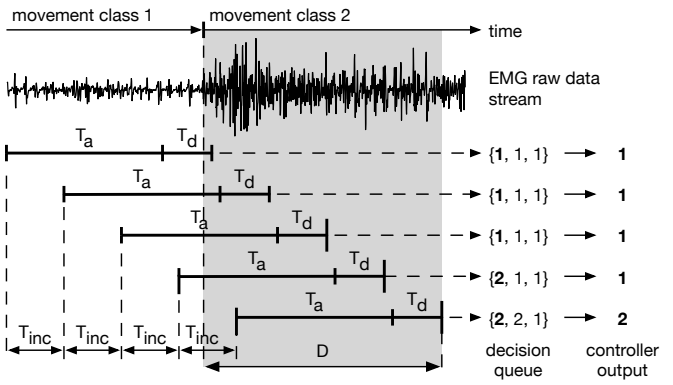

Fig. 1. Windowed approach to EMG classification. One channel of EMG data from two movement classes is shown on top. Below, five analysis windows T_a with subsequent signal processing time T_d and window increments of T_{inc} are illustrated. The decision queue ($n = 3$) and controller output are displayed on the right side. The controller delay D extends between the user's change in EMG signals and the first correct controller output, depicted as grey area.

In order to minimize D, one has to minimize T_a, T_{inc}, T_d and n. Minimizing T_a would be beneficial for controller delay but comes at the expense of classification accuracy. According to Smith et al. [14], the optimal value for T_a is between 150 ms and 250 ms. Another tradeoff is minimizing n: a smaller value improves controller delay while a larger value eliminates more misclassifications. In this work, we use T_a = 150 ms and n = 5, hence the optimal controller delay for our implementation is 79 ms (based on sampling rate of 1000 Hz, T_{inc} = T_d = 1 ms).

B. Time-domain features

In our EMG signal processing, we extract the well-established time-domain (TD) [15] feature set, as described by Hudgins et al. For each channel, one of the following features is calculated from the EMG signal: mean average value (MAV), length of the waveform (WFL), the amount of zero-crossings (ZC) and the amount of slope-sign-changes (SSC).

$$MAV = \frac{1}{N} \sum_{n=1}^{N} x_n \tag{2}$$

$$WFL = \sum_{n=1}^{N-1} |x_{n+1} - x_n| \tag{3}$$

$$ZC = \sum_{n=1}^{N-1} \big(sgn\left(x_n \cdot x_{n+1}\right) \cap |x_n| \geq threshold\big) \tag{4}$$

$$SSC = \sum_{n=2}^{N-1} \Big(sgn\big((x_n - x_{n-1}) \cdot (x_n - x_{n+1})\big)\Big) \tag{5}$$

$$sgn(x) = \left\{ \begin{array}{ll} 1, & \text{if } x \geq threshold \\ 0, & \text{otherwise} \end{array} \right\}$$

C. Classification

For classification of movements, we make use of a multi-class LDA as described by [16]. In the training phase (Algorithm 1), the algorithm is provided with a set of extracted feature vectors from training data and class labels for each class incrementally. A covariance matrix $\mathbf{\Sigma_k}$ and

mean vector $\boldsymbol{\mu_k}$ are calculated and the samples are discarded before proceeding with the next class to save memory space. Finally, all covariance matrices are averaged to $\boldsymbol{\Sigma}$ and inverted using Cholesky decomposition. Multiplied with the matrix $\boldsymbol{\mu}$ containing all mean vectors, we obtain the projection matrix $\mathbf{W}$. Additionally, we compute an offset vector C using the projection matrix $\mathbf{W}$, the mean matrix $\boldsymbol{\mu}$ and the number of classes. In the classification phase (Algorithm 2), the feature vector is projected to a k-dimensional space (k being the number of classes) using $\mathbf{W}$ and added to the offset vector C. The index of the element with the maximum value represents the predicted class.

Algorithm 1 LDA training

procedure TRAINING($feature\_vectors, class\_labels$)
  for all classes k **do**
    $\boldsymbol{\Sigma_k} \leftarrow covariance(feature\_vectors_k)$
    $\mu_k \leftarrow mean(feature\_vectors_k)$
  end for
  $\boldsymbol{\mu} \leftarrow concatenation(\mu_k)$
  $\boldsymbol{\Sigma} \leftarrow mean(\boldsymbol{\Sigma_k})$
  $\mathbf{W} \leftarrow inverse(\boldsymbol{\Sigma}) \cdot \boldsymbol{\mu}$
  $C \leftarrow \frac{1}{2} \cdot dot(\boldsymbol{\mu}, \mathbf{W}) + log(\frac{1}{\#classes})$
end procedure

Algorithm 2 LDA prediction

procedure PREDICTION($feature\_vector$)
  $Y \leftarrow feature\_vector \cdot \mathbf{W} + C$
  $class \leftarrow argmax_k(Y_k)$
  return $class$
end procedure

D. ReconOS

The proposed architecture follows the ReconOS multi-threaded approach to hardware acceleration, as detailed in [17]. In this approach, ReconOS acts as a run-time environment on top of an existing POSIX compliant operating system. By utilizing well known operating system abstractions, the application is first partitioned into multiple software threads that communicate and synchronize using standard methods like mutexes and message boxes. Performance critical threads are then moved to hardware where the ReconOS run-time environment provides a seamless integration into the existing multi-threaded application. A key aspect of this integration is handled by ReconOS by providing the same programming model abstractions to both, hardware and software threads. The reason we chose this approach is that (a) the ReconOS design methodology allows for easy design space exploration by making hardware and software threads interchangeable, and (b) that ReconOS lends itself well to a shared memory approach as it is used in the proposed application architecture.

E. Zynq

The Zynq-7000 systems on chip comprise a processing system (PS) based on a dual-core ARM Cortex-A9 processor and a Xilinx 7 series FPGA programmable logic (PL). The major functional blocks of the PS are the application processor unit, on-chip memory and external memory interfaces, various I/O interfaces, and several interconnect types to allow fast communication between the different blocks. The PL provides configurable logic blocks, block RAM, digital signal processors and analog-to-digital converters. Most of the communication between the PS and PL take place on the AXI interconnect which is also capable of connecting to user-designed IP blocks in the PL [18].

III. Related Work

An extensive range of feature sets, including time-domain [15], frequency-domain [12] or autoregressive coefficients [19] has been investigated for use in EMG pattern recognition-based control schemes. The computed features are fed as input into a classifier in order to determine the motion class. Various statistical and learning classifiers like linear discriminant analysis (LDA) or support vector machines (SVM) have been investigated, a vast comparison is given in [20]. Hargrove et al. [21] demonstrate that if a suitable feature set is used, most modern classifiers will have comparable performance. In this work, we use well-studied TD features in combination with LDA as pattern recognition algorithm to classify EMG data.

Borbely et al. [22] introduced an LDA-based classification design for a Zynq-7000 platform. In this work, an ANSI C implementation was developed to process offline EMG data on a PC as well as on the ARM cores of a Zynq-7020. Further, a hardware implementation for an FPGA, including a preprocessor unit for feature extraction and a vector processor unit for vector and matrix operations was designed. While the proposed software system was tested processing up to 8 channels, the performance of their hardware design, considered only the synthesis results of the preprocessor unit for feature extraction. Based on these results, the authors state that their hardware system is able to process HD EMG data, which was not confirmed by an existing system implementation. Our approach is to process HD EMG data both, on software and accelerated on hardware, and to present results for tests of the implemented system.

While time-domain (TD) features are commonly used in upper-limb pattern recognition, they are unsuitable for lower-limb application due to the non-stationary leg EMG signals. Time-frequency-domain (TFD) features are outperforming TD features in this context. However, current embedded prosthetic controllers are not sufficient for TFD feature extraction in real-time. Xiao et al. [23] used an NVIDIA 8800 GTX GPU to accelerate this processing step and achieved impressive speedups compared to a P4 CPU. While their accelerator implementation is an interesting preparatory work, we believe that acceleration must take place locally on the prosthesis. Also, the authors only utilize one channel of EMG, while it was our goal to process hundreds of channels in parallel.

IV. Architecture

As illustrated by Figure 2, the proposed application receives multi-channel HD EMG data from a sensor array and periodically computes prosthesis movement commands. The embedded prosthetics controller is responsible for pattern learning, pattern recognition, and prosthesis movement. In this section we give a short overview on the hardware/software architecture of the prosthetics controller.

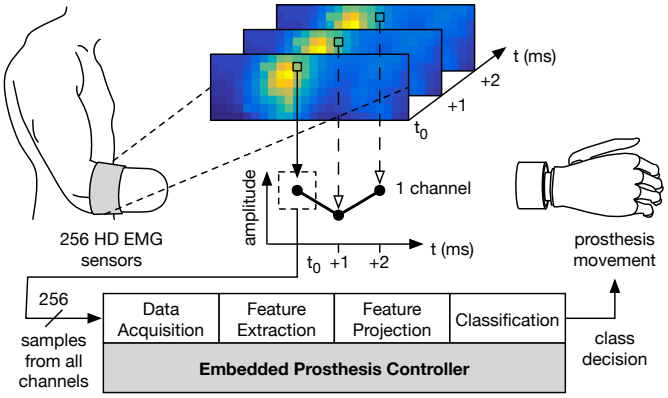

Fig. 2. Top level view on the proposed application. HD EMG data from up to 256 channels are acquired periodically each millisecond, illustrated as a heat-map for the current (time t_0) and two following samplings. One channel is displayed as raw data in the center. The current samples of all channels are fed into the signal processing chain. The output of the classification is used for controlling the prosthesis.

We use a ReconOS based architecture under a Linux operating system on the Xilinx Zynq processing platform. The application is divided into hardware and software threads, with the software threads executed on the ARM CPUs and the hardware threads running on the programmable logic (PL) - the FPGA part of the Zynq. Threads communicate using shared memory and messages boxes - a service provided by the ReconOS runtime environment. Message boxes can be seen as 32 bit wide FIFO buffers with synchronous (blocking) access semantics, managed by the operating system. Threads interact with message boxes by posting or receiving single word messages.

Figure 3 shows a conceptual thread-level view of the system architecture. The partitioning into threads follows the EMG processing chain as described in Section II with data acquisition, feature projection, and classification running as software threads and feature extraction running as either hardware or software thread, or both. During training, an additional software thread for LDA is run on the CPU. While it is possible to pipeline data through the processing chain and run the threads concurrently, at present the processing of the individual stages is done sequentially, so that only a single thread is active at any time. Data flow and synchronization between threads is facilitated by shared memory buffers and pointers to these buffers exchanged via message boxes.

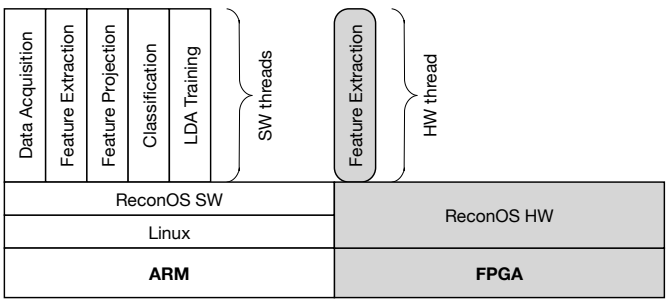

Fig. 3. Partitioning of the application into hardware and software threads.

Figure 4 gives a more detailed view on system architecture with the focus on the feature extraction hardware thread (FE-HWT). The thread possesses two interfaces provided by the ReconOS runtime environment. One interface, the operating system interface (OSIF), allows the thread access operating system services. In the FE-HWT this interface is used for message passing. The second interface, the memory interface (MEMIF) handles accesses to main memory. It is fully burst capable and provides address translation into the virtual address space of the main application process.

The thread executes in three sequential phases. First, a pointer to the location of the sample window is received from a message box and the sample window is read from main memory to a thread-local memory. In the second phase, feature extraction runs on all features in parallel. In the third phase, the extracted feature vector is written back to main memory and a message is sent to the next thread in the processing chain - the feature projection thread. The operation of the FE-HWT is controlled by two state-machines. One state machine - the operating system finite state machine (OS-FSM) - is an integral part to any ReconOS hardware thread. It handles the communication with the software system as well as all memory accesses. In the feature extraction thread, the OS-FSM handles sending and receiving of messages, reading of the sample data into a local memory, moving the sample data to the individual feature extraction units and writing back the resulting feature vectors to shared memory. A second state machine (AUX-FSM) is responsible for storing the computed feature values into a local memory buffer. Sample data is provided for two channels at a time (two samples per 32 bit word). Computed feature values are 32 bit integers.

The feature extraction units are implemented as separate modules that are instanced from the top-level module of the FE-HWT and connected via one incoming and one outgoing 32 bit wide FIFO. Due to this design, the feature extraction units are streaming processors that lend themselves well to a high level synthesis approach.

V. Functional Validation

To validate our system, all results were checked for numeric correction against a reference implementation in MATLAB. Additionally, we conducted two real-time experiments with an 18 years old male trans-radial amputated test person. The test setup is depicted in Figure 5. A MindMedia Nexus 16 [24] was used to acquire 8 channel EMG data which were sent to the ZedBoard via TCP/IP. Since there are currently no prosthetic arms available capable of executing 8 movements, we implemented a virtual prosthesis that can be seen in a 3d Augmented Reality (AR) scene by the test person wearing an Oculus Rift DK2 [25].

We implemented an application based on the Unity 5 game engine to display the video stream from the stereoscopic camera on the Oculus Rift display. A conical marker worn by the test person on his amputation stump was used to place the virtual prosthesis in the video stream. The virtual prosthesis was controlled by control signals from the ZedBoard.

Two experiments were conducted to validate our embedded prosthesis controller under real-world conditions. In the first experiment, the LDA classifier was trained with EMG data of 8 movements (hand open, lateral grip, wrist extension and

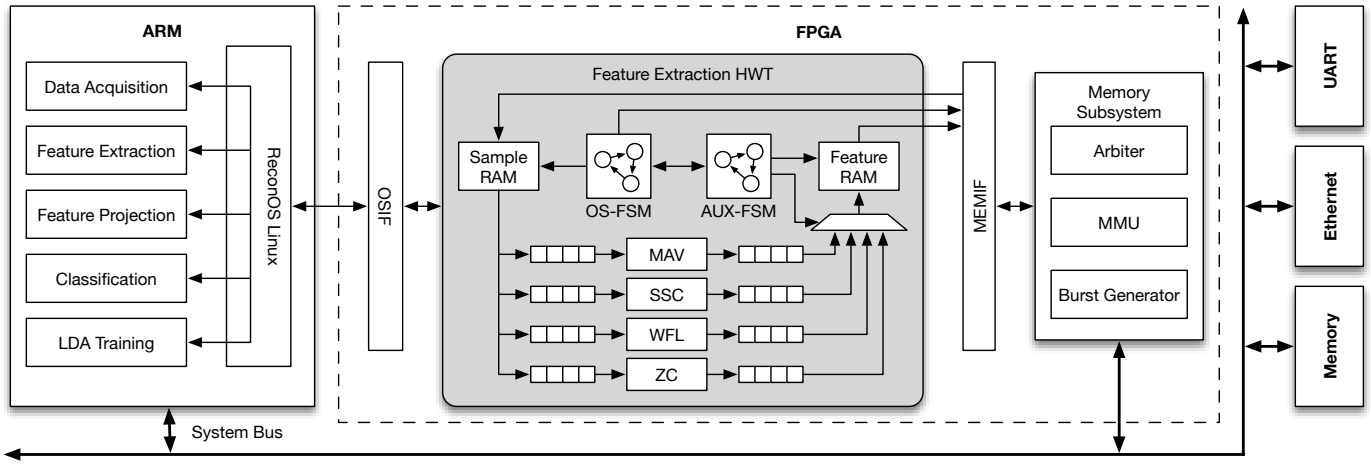

Fig. 4. Application architecture with the focus on the feature extraction hardware thread. The ReconOS runtime enables communication between hardware and software through OS and memory interfaces. The feature extraction units are streaming processors internal to the hardware thread.

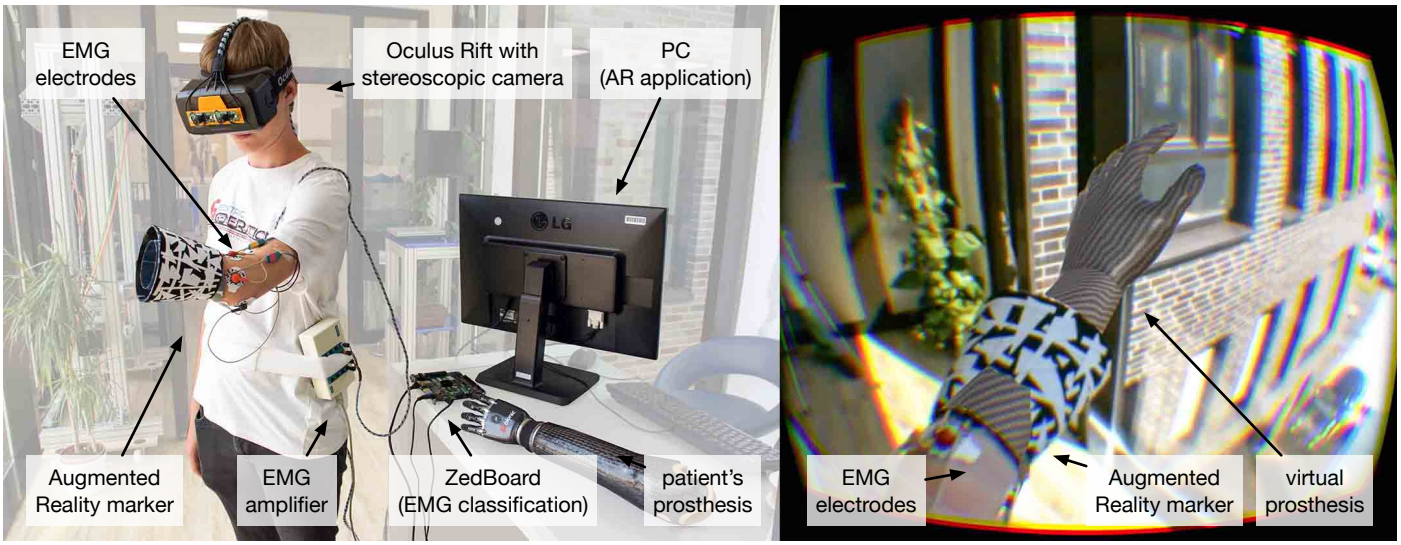

Fig. 5. The experimental setup is shown left. The test person's field of view is depicted right. Only left eye view is shown.

flexion, pronation and supination, index finger extension and rest.) Each movement was repeated 3 times for 5 seconds. The amputee could then move and control the virtual prosthesis freely. In the second experiment, the LDA classifier was trained similar to Experiment 1. Then, the target achievement control (TAC) test [26] was conducted, as implemented in [27]. Instead of moving and controlling the virtual prosthesis freely, the user was prompted to move the virtual arm to predefined target postures. We measured real-time performance in terms of trials the user successfully completed. The TAC test is a well-established tool for evaluating real-time myoelectric pattern recognition control of upper-limb prosthesis. The experiments showed that our system performs well under real-time conditions.

VI. Experimental Results

We implemented the architecture described in Section IV on the ZedBoard [18] development platform. The ZedBoard features a Xilinx Zynq XC7Z020 device, 512 MB DDR system memory and various IO interfaces, such as USB (for UART and JTAG) and Ethernet. While our setup supports both, training and classification, the classification of the EMG signal is more time critical because data must be processed in real time and large controller delays (above 100 ms) are deemed inadequate for sustained prosthesis use.

Generally, there is no data dependence in the computational path of our application. For the evaluation of the application recorded real-world HD EMG data was used for tests using up to 192 channels. When more channels were needed, synthetic data was used.

Starting from a pure software implementation, we observe that the majority of time spent during classification is used for feature extraction. This is shown in Figure 6 where we measured the processing time over the classification chain. Calculation times for feature extraction, feature projection, classification and communication were measured for up to 256 channels. As suggested by these results, our acceleration efforts focused on the feature extraction component.

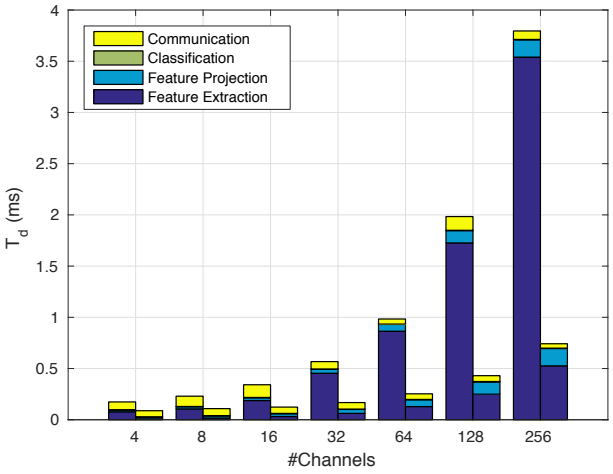

Fig. 6. Time spent in different parts of the classification tool-chain: feature extraction, projection, classification, and communication overhead. T_a = 150 ms was used and values were averaged over 1600 individual measurements. For each channel count, the left bar is software-only and the right bar is with hardware acceleration.

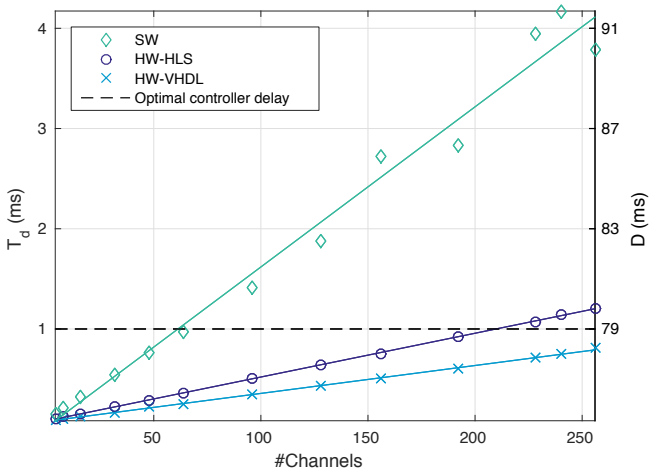

Fig. 7. Total processing delay for a complete sample window and three different feature extraction implementations: software, hardware HLS, hardware VHDL. Values were acquired with T_a = 150 ms, and averaged over 1600 individual measurements. Decision queue depth for controller delay calculation was $n = 5$.

For classification we have evaluated three different setups: First, we tested a software only design, running exclusively on the ARM processor, without hardware acceleration. Then, we tested a hardware accelerated system with feature extraction units implemented by Vivado high level synthesis (HLS). In the third setup, we replaced the HLS generated cores with modules designed by hand in VHDL. Each setup was tested with a varying number of input channels ranging from 4 to a maximum of 256. Figure 7 shows our results in terms of total processing delay (full processing chain) for a complete sample window. All data points were acquired using a window size of 150 samples and averaged over 1600 individual measurements. A processing delay of 1 ms is highlighted because this corresponds to the lowest achievable controller delay of 79 ms with a window shift of 1 ms given our setup with 1000 Hz sampling rate. Our results also show the expected linear increase of processing delay with respect to the number of input channels. While the maximum controller delay for the software-only solution is still acceptable (smaller than 100ms) over the complete tested range of up to 256 channels, only the accelerated design achieves the minimum delay of 79 ms. It is generally accepted that a smaller controller delay (with all other relevant factors being equal) correlates with better usability of the prosthesis. An extrapolation of the results suggests, that the unaccelerated application will surpass a controller delay of 100 ms when more than 380 channels are used. In comparison, the accelerated design is expected to handle over 2200 input channels with less than 100 ms controller delay.

We also see that there is a noticeable difference between the HLS generated accelerators and the VHDL implementations. This is magnified when we take a look at the execution times for feature extraction alone. Figure 8 shows processing times for the individual feature extractors, MAV, SSC, WFL, ZC, and the combined time for extracting all four features. Measurements were taken for the three different implementations: software, hardware generated by HLS, and hardware specified in VHDL. The results reveal two major observations: One is that the bulk of savings in processing time comes not so much from the acceleration of the individual features, but for the most part from the features being computed in parallel in hardware, rather than sequentially in software. The other observation is that, again, the hand-written accelerators perform much better than the HLS generated ones. The reason for that is that, with the exception of MAV, the HLS tool was not able to pack all computations needed to process a sample into a single cycle, in contrast to the VHDL specified feature extractors. Still, even the HLS generated cores deliver good speed-ups, and for more complex feature extractors, and depending on the time and effort one is willing to spend on the implementation, HLS might even be a better option.

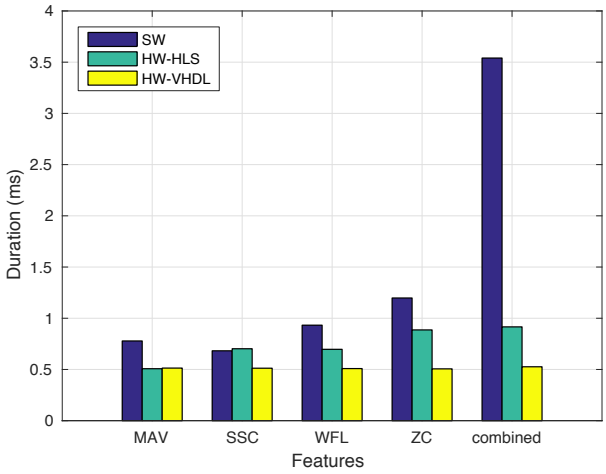

Fig. 8. Processing times for three different implementations (software, hardware HLS, hardware VHDL) of the feature extractors MAV, SSC, WFL, ZC, and all features combined. Values have been acquired with 256 channels, T_a = 150 ms, and averaged over 9852 individual measurements.

Although the speed-up achieved so far is significant, a look at the utilization of the Zynq PL shows that we are only using a small fraction of the available area. Table I shows the utilization of the Zynq FPGA for the individual HLS and

VHDL feature extraction units, as well as for the complete hardware thread. The small utilization numbers indicate that on the one hand we could use a much smaller FPGA for an embedded prosthetics controller, thereby reducing costs and power consumption, and on the other hand we could theoretically achieve better speed-ups on the existing platform by utilizing the complete FPGA.

TABLE I. ZYNQ PL UTILIZATION

		MAV	ZC	SSC	WFL	Total	Utilization
VHDL	**LUTs**	207	251	383	216	2376	1.58 %
	FFs	76	80	139	107	1138	0.38 %
	BRAMs	0	0	0	0	33	7.93 %
	DSPs	0	0	0	0	4	0.52 %
HLS	**LUTs**	218	351	337	286	2377	1.58 %
	FFs	74	130	214	179	1351	0.45 %
	BRAMs	0	0	0	0	33	7.93 %
	DSPs	0	0	2	0	6	0.78 %

Figure 6 shows that even after speeding up feature extraction by a factor of 6.7, still most of the time for classification is spent there, even though feature projection and communication begin to play a bigger part. This suggests that further improvements of the feature extraction hardware thread may be worth looking at. Figure 9 shows how the time is currently spent in the hardware thread. While feature write-back is negligible, reading in the sample window and feature vector computation are dominating. The current implementation reads in the complete sample window in every cycle, even for a window shift of 1 sample. This stage could be much improved by reading in only the new samples at every time step. Also, currently only two channels are processed at a time in the feature extraction units. As the utilization shown in Table I indicates, there should be enough room to process over 100 channels in parallel.

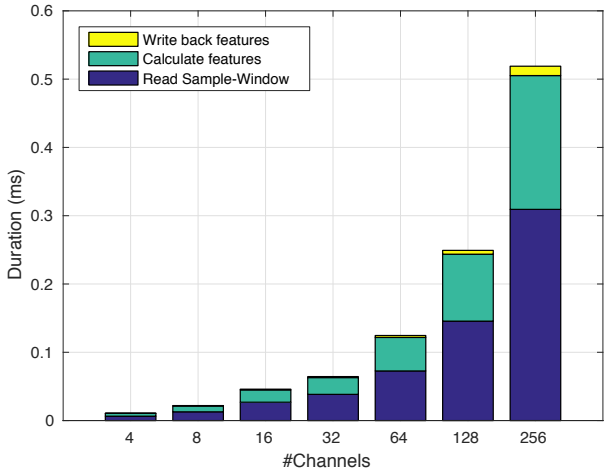

Fig. 9. Duration of the execution phases of the feature extraction hardware thread. The phases are separated into: reading in sample window from memory to local RAM, calculating features and writing features back from local RAM to memory. Values were acquired with T_a = 150 ms and averaged over 9852 individual measurements.

Figure 10 examines LDA training times and the share of prominent parts during the training phase. Displayed is the accumulated value of all 1600 extracted windows with hardware feature extractors implemented in VHDL. Processing time of

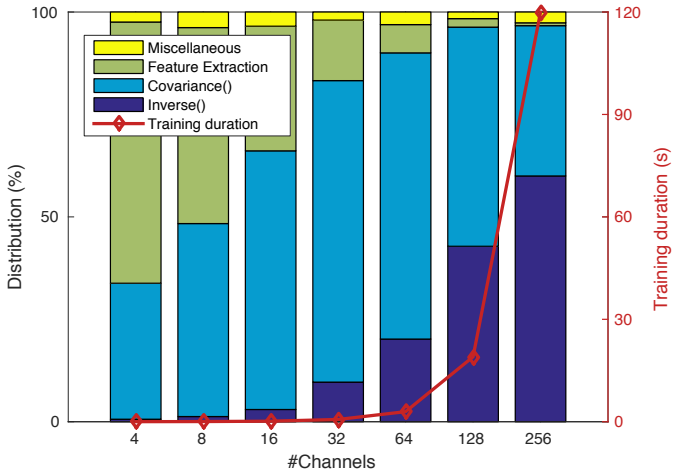

Fig. 10. Time spent for training over the number of channels (red curve) and the relative duration of the individual phases. Given T_a = 150 ms with T_{inc} = 50 ms and 80000 samples, features of 1600 windows are extracted. Training times were measured with different numbers of channels.

covariance (see line 3 of Algorithm 1) is the accumulated time of 8 covariance calculations (one per class), the duration for inverse calculation (see line 8 of Algorithm 1) is measured at the end of the training phase. The entry "Miscellaneous" comprises times for communication and additional calculations during training. Looking at the partial execution times, both, covariance and inverse matrix calculations increase substantially with increasing number of channels. To decrease overall training time acceleration of both matrix operations is desirable and can be achieved by parallelization in hardware as shown for inverse matrix calculation in [28] and covariance in [29].

VII. CONCLUSION

We have presented an architecture for an embedded HD EMG prosthetics controller based on the ReconOS multithreading approach to hardware acceleration. The system was implemented on a Xilinx Zynq platform and validated in a real-life situation by an amputated test person who was able to use a virtual prosthesis controlled by our device. The system was then evaluated in terms of performance and reconfigurable resource utilization. The hardware accelerated design achieved speed-ups of up to 4.8 over the software-only solution, allowing for a processing delay lower than the sample period of 1 ms, effectively reducing the total controller delay to 79 ms.

In future work, we plan to further utilize the available area on the Zynq PL for the computation of more complex features. Also, we are working on a design that reconfigures between training and classification, allowing to use the complete FPGA for each phase.

REFERENCES

[1] D. S. Dorcas and R. N. Scott, "A three-state myo-electric control," *Med. Biol. Eng. Comp.*, vol. 4, pp. 367–370, 1966.

[2] D. Childress, "A myoelectric three state controller using rate sensitvity," in *Proc. Int. Conf. Math. Biol. and Ecology (ICMBE)*, 1969.

[3] C. Lake and J. M. Miguelez, "Evolution of microprocessor based control systems in upper extremity prosthetics," *Technology and Disability*, vol. 15, pp. 63–71, Aug. 2003.

[4] D. Atkins, D. Heard, and W. Donavan, "Epidemiologic Overview of lndividuals with Upper-Limb Loss and Their Reported Research Priorities," *J. Prosthet. Orthot.*, vol. 8, pp. 2–11, 1996.

[5] E. J. Scheme and K. B. Englehart, "Electromyogram pattern recognition for control of powered upper-limb prostheses: State of the art and challenges for clinical use," *J. Rehabilitation Research and Development*, vol. 48, no. 6, pp. 643–18, 2011.

[6] A. J. Young, L. J. Hargrove, and P. J. Kyberd, "The Effects of Electrode Size and Orientation on the Sensitivity of Myoelectric Pattern Recognition Systems to Electrode Shift," *IEEE Trans. Biomed. Eng.*, vol. 58, no. 9, pp. 2537–2544, Aug. 2011.

[7] A. Fougner, E. Scheme, A. D. C. Chan, K. B. Englehart, and O. Stavdahl, "Resolving the Limb Position Effect in Myoelectric Pattern Recognition," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 19, no. 6, pp. 644–651, Nov. 2011.

[8] *Quattrocento Multichannel Data Sheet*, OT Bioelettronica, 2015. [Online]. Available: http://www.otbioelettronica.it/attachments/article/72/Quattrocento.pdf

[9] A. Boschmann and M. Platzner, "Reducing the limb position effect in pattern recognition based myoelectric control using a high density electrode array," in *IEEE ISSNIP Biosignals and Biorobotics Conference (BRC)*, 2013.

[10] A. Stango, F. Negro, and D. Farina, "Spatial Correlation of High Density EMG Signals Provides Features Robust to Electrode Number and Shift in Pattern Recognition for Myocontrol," *IEEE Trans. Neural. Syst Rehabil. Eng.*, vol. 23, no. 2, pp. 189–198, Mar. 2015.

[11] A. Boschmann and M. Platzner, "Towards Robust HD EMG Pattern Recognition: Reducing Electrode Displacement Effect Using Structural Similarity," in *IEEE Int. Conf. Eng. Med. Biolog. (EMBC)*, Jun. 2014, pp. 4547–4550.

[12] K. Englehart and B. Hudgins, "A robust, real-time control scheme for multifunction myoelectric control," *IEEE Trans. Biomed. Eng.*, vol. 50, no. 7, pp. 848–854, Jul. 2003.

[13] T. R. Farrell and R. F. Weir, "Analysis Window Induced Controller Delay for Multifunctional Prostheses," in *Proc. MyoElectric Controls Symposium (MEC)*, 2008.

[14] L. H. Smith, L. J. Hargrove, B. A. Lock, and P. J. Kyberd, "Determining the Optimal Window Length for Pattern Recognition-Based Myoelectric Control: Balancing the Competing Effects of Classification Error and Controller Delay," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 19, no. 2, pp. 186–192, Apr. 2011.

[15] B. S. Hudgins, P. A. Parker, and R. N. Scott, "A New Strategy for Multifunction Myoelectric Control," *IEEE Trans. Biomed. Eng.*, vol. 40, no. 1, pp. 82–94, 1993.

[16] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (Second Edition)*. Wiley-Interscience, 2001.

[17] A. Agne, M. Happe, A. Keller, E. Lubbers, B. Plattner, M. Platzner, and C. Plessl, "ReconOS: An Operating System Approach for Reconfigurable Computing," *IEEE Micro*, vol. 34, no. 1, pp. 60–71, 2014.

[18] *Zynq-7000 Technical Reference Manual*, Xilinx, February 2015. [Online]. Available: http://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf

[19] D. Gaupe, J. Salahi, and D. Zhang, "Stochastic analysis of myoelectric temporal signatures for multifunctional single-site activation of prostheses and orthoses," *J. Biomed. Eng.*, vol. 7, 1985.

[20] E. J. Scheme, K. B. Englehart, and B. S. Hudgins, "A One-Versus-One Classifier for Improved Robustness of Myoelectric Control," in *Congress of the Intl. Soc. of Electrophysiology and Kinesiology*, 2010.

[21] L. J. Hargrove, K. B. Englehart, and B. S. Hudgins, "A Comparison of Surface and Intramuscular Myoelectric Signal Classification," *IEEE Trans. Biomed. Eng.*, vol. 54, no. 5, pp. 847–853, Apr. 2007.

[22] B. J. Borbely, Z. Kineses, Z. Vorohazi, Z. Nagy, and P. Szolgay, "Analysis of myoelectric signals using a Field Programmable SoC," in *Eur. Conf. Circuit Theory and Design (ECCTD)*, 2013.

[23] W. Xiao, H. Huang, Y. Sun, and Q. Yang, "Promise of embedded system with GPU in artificial leg control: enabling time-frequency feature extraction from electromyography." in *IEEE Int. Conf. Eng. Med. Biolog. (EMBC)*, 2009, pp. 6926–6929.

[24] *Nexus 16*, Mind Media, February 2014. [Online]. Available: http://www.mindmedia.info/CMS2014/en/products/systems/nexus-16

[25] *DK2 Development Kit 2 Instruction Manual*, Oculus VR, July 2014. [Online]. Available: http://static.oculus.com/sdk-downloads/documents/Oculus_Rift_DK2_Instruction_Manual.pdf

[26] A. M. Simon, L. J. Hargrove, B. A. Lock, and T. A. Kuiken, "Target Achievement Control Test: evaluating real-time myoelectric pattern-recognition control of multifunctional upper-limb prostheses." *Journal of rehabilitation research and development*, vol. 48, no. 6, pp. 619–627, 2011.

[27] M. Ortiz-Catalan, R. Brånemark, and B. Håkansson, "BioPatRec: A modular research platform for the control of artificial limbs based on pattern recognition algorithms." *Source code for biology and medicine*, vol. 8, no. 1, p. 11, 2013.

[28] M. Karkooti, J. R. Cavallaro, and C. Dick, "Fpga implementation of matrix inversion using qrd-rls algorithm," in *Proc. Asilomar Conf. on Signals, Systems and Computers*, 2005, pp. 1625–1629.

[29] J. Chilson, R. Ng, A. Wagner, and R. Zamar, "Parallel computation of high dimensional robust correlation and covariance matrices," in *ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2004, pp. 533–538.