An Embedded FPGA Accelerator for a Stand-alone Dual-Mode Assistive Device

Ali Jafari[1], Maysam Ghovanloo[2], and Tinoosh Mohsenin[1]

[1]Department of Computer Science & Electrical Engineering, University of Maryland, Baltimore County
[2]School of Electrical & Computer Engineering, Georgia Institute of Technology

***Abstract*—This paper presents a stand-alone Dual-mode Tongue Drive System (sdTDS) which is designed for people with severe disabilities to control their environment using their tongue motion and speech. The sdTDS detects user's tongue motion using a magnetic tracer placed on tongue and an array of magnetic sensors embedded in a wireless headset and at the same time it can capture the user's voice using a small microphone embedded in the same headset. A real-time FPGA-based local processor is proposed which can perform all required signal processing to convert raw data generated by magnetic sensors and microphone to user commands at sensor side, rather than sending all raw data out to a PC or smartphone. The proposed sdTDS significantly reduces the transmitter power consumption and subsequently increases the battery life. Assuming the sdTDS user issues one command every 20 ms, implementing the proposed local processor reduces the data volume that needs to be wirelessly transmitted from 25.6 kb/s to 0.3 kb/s. To evaluate the functionality and performance of the sdTDS processor, it has been implemented on a Xilinx Zynq SoC device (ARM+Artix FPGA) and at frequency of 100 MHz, it consumes 2 mJ energy. The detection accuracy is 96.6% for tongue motion, and 97.5% for speech recognition.**

I. Introduction

According to a 2008 survey [1], approximately 5.6 million people reported suffering from some form of paralysis, 36% of whom claimed to have a lot of difficulty in moving and 16% claimed they were completely unable to move. Although much has been done to increase the quality of life for people inflicted with paralysis and to help them lead productive and independent lives, there are particularly difficult challenges that arise when dealing with those suffering from tetraplegia, in which all four limbs are affected by the paralysis [2].

Assistive Technologies (ATs) can help people with disabilities to perform many daily activities with minimal or no assistance and increase their independence. ATs can potentially take advantage of any abilities of these individuals such as eye movements, head motion, muscle contractions, brain signals and even tongue movements to provide this population with alternative means to interact with computers and electronic devices [3]–[5]. Multi-modal assistive devices can increase the number of alternatives available to users to perform different tasks simultaneously while giving users the ability to switch among different input sensor modalities, based on their needs, convenience, and environmental conditions [5], [6].

To convert output of each sensor modality to a meaningful command, different complex signal processing such as filtering, feature extraction and machine learning classification are needed to be performed. Transmitting raw data generated from different sensors to a PC/smartphone for further signal processing results in high power consumption, which is not desired in battery-powered wearable assistive devices.

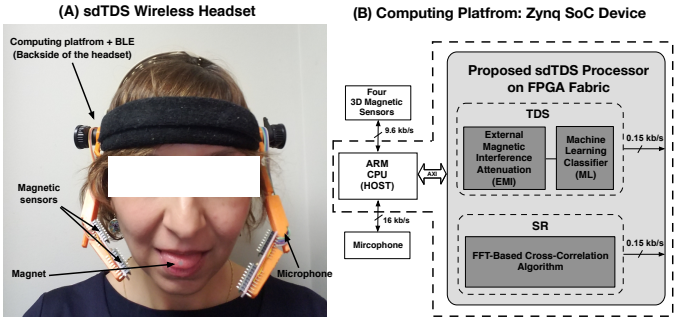

Fig. 1. (A) sdTDS wireless headset including 4 magnetometers, a microphone, processor and BLE. (B) Top block diagram of the processor which is used in sdTDS and contains Tongue Drive and Speech Recognition kernels implemented on Zynq SoC device.

In this work, sdTDS shown in Fig. 1-A is proposed which can perform the entire signal processing at the sensor node and only sends the final decision through a BLE (Bluetooth Low Energy) to a smartphone/PC which significantly reduces the transmission power consumption by reducing the transmission rate from 25.6 kb/s to 0.3 kb/s, assuming the user issues one command every 20 ms and without transmission overhead. sdTDS is placed on a wireless headset which integrates analog front end (Four magnetometers and a microphone), an FPGA-based local processor and a BLE. The FPGA-based proposed processor shown in Fig. 1-B is able to receive the data coming form two different sensor modalities: 4 magnetometers for tracing a magnet placed on user's tongue and a microphone for recording user's voice and then it performs the entire required signal processing for each modality in parallel. The detected commands can be used for example for controlling a mouse and a wheelchair.

II. ALGORITHM DESIGN AND ANALYSIS

An overview of the proposed algorithms design which is implemented on Zynq SoC (ARM+Artix FPGA) device is discussed in this section.

A. Tongue Drive

Tongue is an ideal candidate for developing a wearable AT device [5]–[7], because of fast movement with many degrees of freedom and high flexibility. TDS detects the tongue motion by measuring the magnetic field variation generated by a magnetic tracer attached to the tongue using an array of magnetic sensors placed on a wireless headset. As shown in Fig. 1-B, TDS mainly contains an External Magnetic Interference (EMI)

attenuation and a Machine Learning (ML) classifier module. EMI attenuation module receives the data coming from 4 magnetometers and generates some features that can be used for classifying the user-defined commands.

1) External Magnetic Interference Cancellation: Magnetic sensors are highly sensitive and inevitably affected by external magnetic interference (EMI), such as the Earth's magnetic field (EMF). This results in a poor signal-to-noise ratio (SNR) at the sensor outputs, which degrades the performance of the machine learning algorithm. Therefore, eliminating the EMI is necessary to enhance the TDS performance, and reduce the probability of errors in command interpretation. The raw data from each sensor includes the user data generated by the magnetic tracer and EMI. Since the relative position and orientation of each two sensor modules are fixed, the sensor outputs in response to EMI are linearly related. This enables us to remove the EMI interference by subtracting one sensor measurement from another and then classifying the command using the remaining difference between the two sensors measurements of the actual user data. Since the sensors provide 3-D measurements, subtracting one measurement from another will provide a 3-D feature for classification. Applying equations (1-5) will eliminate EMI for left-side sensors and they can be repeated for the right-side sensors, producing a total of two 3-D features that can be used for classifying the user-defined commands. The following abbreviations will be used for the equations (1-5): Raw = R, Data = D, Feature = F, Mapped = M, Front Left = FL, Back Left = BL, Front Right = FR, Back Right = BR, Mapped EMI = MEMI.

$$R_{FL} = D_{FL} + EMI_{FL} \tag{1}$$

$$R_{BL} = D_{BL} + EMI_{BL} \tag{2}$$

$$MR_{BL} = R_{BL} * Mapping_{coefficients} = MD_{BL} + MEMI_{BL} \tag{3}$$

$$EMI_{FL} = MEMI_{BL} = EMI \tag{4}$$

$$F_L = R_{FL} - MD_{BL} = D_{FL} - MD_{BL} \tag{5}$$

Replacing (1-4) into (5) results EMI to be ideally removed from the signal. In order to generate the linear equations used in (5), first it is needed to record a set of data where there is known to be a linear relationship between the back sensors measurements and the front sensors measurements. In order to generate this data set, the magnetic tracer is removed from the system and only the remaining EMI is recorded, which is assumed to be equal across the four sensors. This data is input to a linear regression function in order to generate the coefficients which will map the back senors data to the front sensors coordinate system.

2) Machine Learning Classifier: To evaluate the proposed TDS accuracy, different machine learning classifiers such as K-Nearest Neighbor (KNN), Support Vector Machine (SVM) and Logistic Regression (LR) have been employed and evaluated with respect to their complexity, power consumption and accuracy. Fig. 2 shows a comparison among these ML algorithms in terms of detection accuracy, number of computations and memory requirements. As it can be seen from the results, LR achieves similar accuracy compared to another algorithms, it needs 762× and 12× less computation and also it requires 400× and 10× less memory (for saving the train data) compared to KNN and SVM algorithms. Hence, in this paper LR is chosen as the ML classifier and implemented on Zynq device.

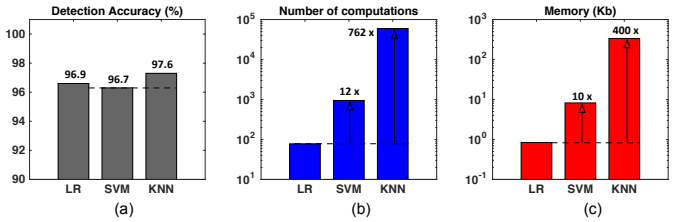

Fig. 2. Machine Learning classifiers comparison in terms of detection accuracy, number of computations and memory requirements.

Logistic Regression is a discriminative probabilistic model that utilizes the logistic function to map real-valued inputs between 0 and 1 which can be interpreted as probabilities. Training is an iterative process that finds maximum likelihood regression coefficients (weight vectors). Samples are typically classified by selecting the label with the greater probability between $P(Y = K|X)$ and $P(Y = 0|X)$ given in equation (6), where $w_{k,i}$ are the weights for each of the N features for the label, k, and x is the input vector:

$$P(Y = k|X) = \frac{1}{1 + exp\left(w_{k,0} + \sum_{i=1}^{N} w_{k,i}x_i\right)} \tag{6}$$

Also, probability ratios are used which significantly reduce computational complexity as shown in equation (7).

$$\frac{P(Y = 1 : K - 1|X)}{P(Y = K|X)} \Rightarrow exp\left(w_{0,1:K} + \sum_{i=1}^{N} w_{i,1:k}x_i\right) \Rightarrow \sum_{i=0}^{N} w_{i,1:k}x_i \tag{7}$$

B. Speech Recognition

Speech Recognition (SR) technology can produce literally unlimited number of available commands. The people with severe disabilities can take advantage of this technology as long as they have the ability to speak. However, the task completion times for most of the existing SR software packages are long because of the inherent delay of SR [8], [9]. Also, they need a PC/smartphone for operating. Therefore, a fast embedded SR design is proposed which can recognize different voice commands right at the microphone side, rather than sending all voice signals to a PC and using a SR software packages to process them. Also, the proposed SR is stand-alone as it does not require a PC/Smartphone or a software package.

One way of recognizing a voice signal is to compare similarity of a signal to other pre-labeled voice signals. In this work, cross-correlation algorithm is proposed to find the similarity of voice signals. The cross-correlation function shown in (8) is used to measure similarity of two signals as a function of the displacement of one relative to the other. Cross-correlation has been employed in different applications such as pattern recognition, single particle analysis, electron tomography, averaging, cryptanalysis, and neurophysiology.

$$(f \star g)[n] = \sum_{m=-\infty}^{\infty} f^*[m] \times g[m + n] \tag{8}$$

Any vector of input test voice signal is cross correlated by all training data. The training data include pre-labled voice signals. Then the results of cross correlation of the input test signal with all the training data are compared to find a maximum. The label of that particular cross correlation which leads to a maximum, will be the label for the input test signal. The cross-correlation function is computationally

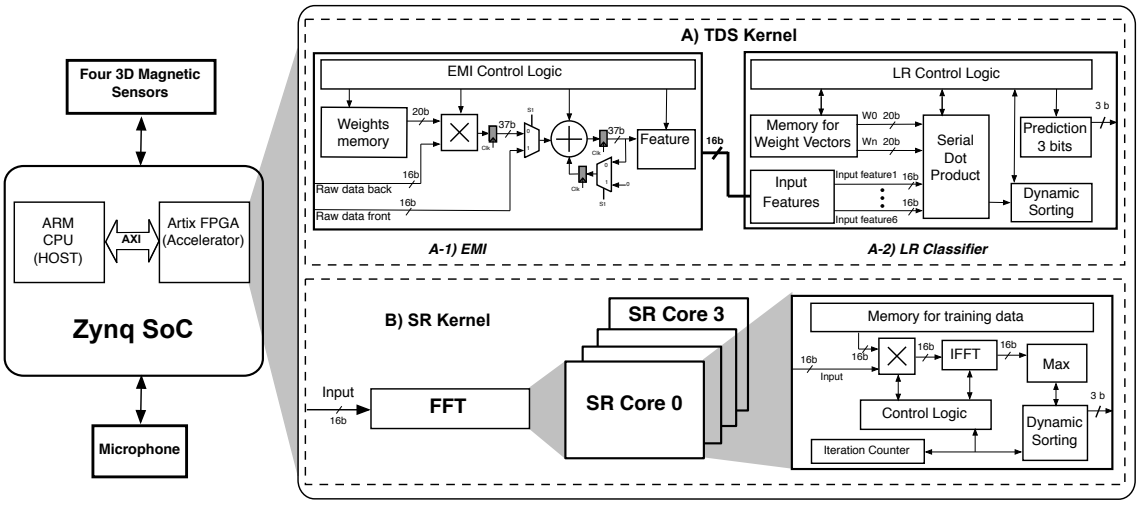

Fig. 3. Hardware architecture used in the proposed processor including TDS and SR kernels. TDS (A) contains EMF and LR modules and SR (B) contains 4 computation cores for each voice command in parallel.

intensive, therefore an FFT-based cross-correlation algorithm is implemented in this work to reduce the computations and consequently decreases the latency.

1) FFT-Based Cross-Correlation: FFT cross-correlation uses the principle that multiplication in the frequency domain corresponds to cross-correlation in the time domain as it is shown in (9). The input signals are transformed into the frequency domain using the FFT, multiplied, and then transformed back into the time domain using the inverse FFT. By using the FFT algorithm, cross-correlating via the frequency domain can be performed faster than directly cross-correlating the time domain signals. The final result is same; however, only the number of calculations will be reduced.

$$(f \star g)[n] = IFFT\left\{ FFT^{*}(f[n]) \times FFT(g[n]) \right\} \quad (9)$$

2) Dataset and performance measurement: Several different data sets are captured using a microphone and Matlab software, for training and testing purpose. The proposed SR is designed to recognize 4 commands (Left, Right, Forward, Backward), although it can potentially be designed to recognize more number of commands. To estimate SR algorithm accuracy, it is trained on 10 reference voice records from a user for each of 4 commands. The detector is then tasked with detecting 10 new test voice records. Based on the results, the detection accuracy is 97.5%.

III. Hardware Architecture

Implementing hardware architecture for TDS and SR algorithms faces several challenges such as pre-processing the features, computational model implementation, managing memory transfers. Fig. 3 depicts sdTDS hardware architecture with implementation details. Fig. 3-A shows TDS kernel. As it can be seen from Fig. 3-A-1, EMI block contains one multiplier, one adder/subtractor, one Block RAM for saving EMI calibration data, 2 multiplexers, a few registers, and a state machine block. The input data coming from serial interface is 16-bit two's complement. Also, the EMI calibration data are represented by 20 bits. After performing EMI cancellation, the data is truncated to 16 bits and saved in feature memory.

Based on the results provided in Section II-A-2, LR is chosen to be implemented as the ML classifier. Usually, LR is trained offline [10]. An offline training is performed to obtain weight vector using Matlab. The weight vectors are converted to fixed point format and are represented by 16 bits. The LR architecture shown in Fig. 3-A-2 consists of four main blocks: a memory block for saving weight vectors, a serial dot product engine with bias vector addition, a dynamic sorting module to find maximum values and related indexes, and a state machine for controlling all sub-modules. Verilog HDL is used to describe architecture and hardware of the TDS.

Fig. 3-B shows SR hardware architecture implemented using Xilinx Vivado High Level Synthesis (HLS). At the front end of the SR kernel, FFT function converts time-domain voice signal to frequency-domain signal. Then the data is broad-casted to 4 parallel SR cores. Each core performs the processing for each command simultaneously. Also, Fig. 3-B shows the internal architecture of each core. As it can be seen, the input data coming from FFT function are multiplied with training data which are saved in the memory. The training data are FFT of the training voice signals and generated using Matlab and saved on the Block RAMs of FPGA. After performing the multiplication operation the data is transferred back to the time-domain using IFFT function. Then, Maximum module finds maximum of the resulting time domain signal. This process repeats for the number of training data, for each command. Dynamic Sorting module finds the global maximum among all others through the Maximum module.

A. Data Format Exploration

A key consideration in a hardware architecture design is whether to utilize fixed-point or floating-point format. Floating-point format has the advantage of supporting dynamic range, underflow/overflow conditions, and a nonuniform scale. However, the downside is that floating-point requires significantly more logic than an equivalent fixed-point implementation. We model a custom fixed-point FFT algorithm to find optimum datapath word length. To

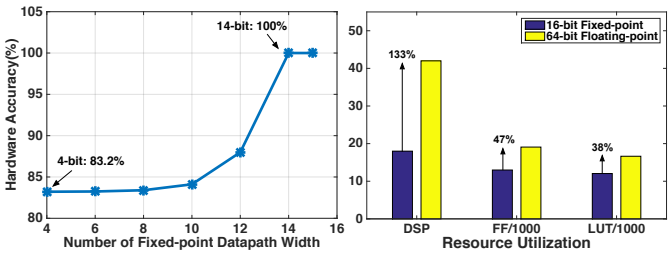

Fig. 4. Comparison of 64-bit Floating-point and 16-bit Fixed-point representations in terms of required resources for the FFT core.

quantify performance gap between floating point Matlab implementations and proposed fixed-point architectures, we calculate average accuracy from the estimated signal obtained from Matlab software solution and hardware implementations. Fig. 4 shows accuracy of FFT computation signal is dependent on number of fixed-point bits used in datapath. Based on the results, 2.14 fixed-point format gives hardware accuracy of 100%. Also, Fig. 4 compares the required device utilization when using either fixed-point or floating-point for FFT core. Floating-point requires nearly 133% more DSP than fixed-point, 47% more flip flops and 38% more LUT. These results are based on the implemented design using HLS. The required level of resource utilization decreases drastically for the fixed-point representation. Therefore, the proposed SR design is implemented in 16-bit (2.14) fixed-point format.

IV. FPGA Implementation Results

The complete proposed sdTDS solution which includes TDS and SR kernels is implemented on a Xilinx Zynq SoC device (ARM+Artix FPGA), xc7z020clg484 package.

Fig. 5 shows power breakdown of post-place and route implementation. Power consumption is obtained using Vivado power estimation tool. As it can be seen from the figure, the static power consumption of FPGA fabric is around 173 mW and the total power consumption of the design is around 677 mW. The ARM processor power consumption is excluded as it is not used for performing processing.

Table I shows the device utilization breakdown of the proposed sdTDS on the Zynq SoC device. The table shows that the proposed processor uses 96 DSP Slices which is 44% of the available DSPs on the device. Also, 65% of the slices on FPGA are used and only 12% of Block RAMs are utilized for the proposed deign. The latency for TDS and SR to finish all computations for one window of input data is 6.6 μs and 3.1 ms, respectively.

V. Conclusion

This paper presents sdTDS, a dual-mode assistive device for people with tetraplegia, who have severe disabilities in their daily life. sdTDS processor detects user's tongue motion generated by a magnet and four magnetometers and in parallel recognizes different voice commands generated by the user through microphone. The proposed embedded processor performs all signal processing including EMI cancellation and ML classification for the TDS kernel and FFT-based cross-correlation for the SR kernel, locally and in parallel at sensor side and sends out only the commands. This approach significantly reduces the transmission power consumption by reducing the transmission rate from 25.6 kb/s to 0.3 kb/s. By

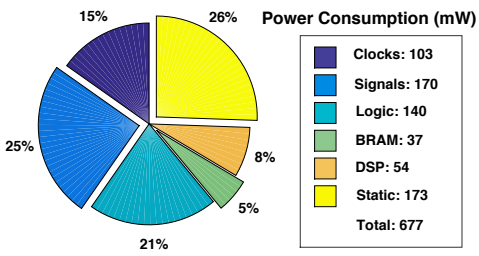

Fig. 5. Power consumption breakdown of sdTDS processor on Zynq SoC device.

TABLE I. sdTDS Processor resource utilization on Zynq SoC device

Resource Utilization Summary			
Resource	Used (#)	Available (#)	Utilization (%)
DSP Slices	96	200	44
BRAM	16.5	140	12
Slice	8685	13300	65
Slice LUT	27544	53200	52
Slice Registers	27544	106400	31

employing the proposed processor, the BLE doesn't need to be active all the time and it can be in advertisement mode which consumes very low power. The processor is implemented on a Zynq SoC device and at 100 MHz frequency, it consumes 2 mJ energy. The processor can detect the tongue motion and input voice signals with accuracy of 96.6% and 97.5% within 6.6 μs and 3.1 ms, respectively. The detected commands can be used to control different devices such as mouse and wheelchair.

References

[1] A. Cahill, H. Fredine, and L. Zilberman, "Initial briefing: Prevalence of paralysis including spinal cord injuries in the united states," 2009.

[2] A. Jafari, M. Ghovanloo, and T. Mohsenin, "A real-time embedded fpga processor for a stand-alone dual-mode assistive device," in *Field-Programmable Custom Computing Machines (FCCM), IEEE 25th International Symposium*. IEEE, 2017, pp. 199–199.

[3] A. Jafari *et al.*, "An eeg artifact identification embedded system using ica and multi-instance learning," in *International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2017.

[4] D. J. McFarland *et al.*, "Emulation of computer mouse control with a noninvasive brain? computer interface," *Journal of neural engineering*, vol. 5, no. 2, p. 101, 2008.

[5] X. Huo, H. Park, J. Kim, and M. Ghovanloo, "A dual-mode human computer interface combining speech and tongue motion for people with severe disabilities," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 21, no. 6, pp. 979–991, 2013.

[6] M. Sahadat *et al.*, "A multimodal human computer interface combining head movement, speech and tongue motion for people with severe disabilities," in *Biomedical Circuits and Systems Conference (BioCAS)*. IEEE, 2015, pp. 1–4.

[7] A. Jafari *et al.*, "A low power wearable stand-alone tongue drive system for people with severe disabilities," *IEEE Transactions on Biomedical Circuits and Systems*, 2017.

[8] E. I. Abbas and A. A. Refeis, "Isolated uttered words recognition based on gmm/hmm algorithms using sopc/nios ii processor build on altera cyclone ii fpga chip," in *Engineering Sciences (FNCES), 2012 First National Conference for*. IEEE, 2012, pp. 1–8.

[9] T. Sledevič and D. Navakauskas, "Fpga based fast lithuanian isolated word recognition system," in *EUROCON*. IEEE, 2013, pp. 1630–1636.

[10] A. Jafari *et al.*, "A low power seizure detection processor based on direct use of compressively-sensed data and employing a deterministic random matrix," in *Biomedical Circuits and Systems Conference (BioCAS)*. IEEE, 2015, pp. 1–4.