Complementary Detection for Hardware Efficient On-site Monitoring of Parkinsonian Progress

Ameer Mohammed, *Member, IEEE* and Andreas Demosthenous, *Fellow, IEEE*

***Abstract*—The progress of Parkinson's disease (PD) in patients is conventionally monitored through follow-up visits. These may be insufficient for clinicians to obtain a good understanding of the occurrence and severity of symptoms in order to adjust therapy to the patients' needs. Portable platforms for PD diagnostics can provide in-depth information, thus reducing the frequency of face-to-face visits. This paper describes the first known on-site PD detection and monitoring processor. This is achieved by employing complementary detection which uses a combination of weak k-NN classifiers to produce a classifier with a higher consistency and confidence level than the individual classifiers. Various implementations of the classifier are investigated for trade-offs in terms of area, power and detection performance. Detection performances are validated on an FPGA platform. Achieved accuracy measures were: Matthews correlation coefficient of 0.6162, mean F1-score of 91.38%, and mean classification accuracy of 91.91%. By mapping the implemented designs on a 45 nm CMOS process, the optimal configuration achieved a dynamic power per channel of 2.26 µW and an area per channel of 0.24 mm^2.**

***Index Terms*—Biomedical signal processor, classifier, deep brain stimulation (DBS), event detection, feature extraction, Parkinson's disease (PD).**

I. Introduction

PARKINSON'S disease (PD) has complex mechanisms [1], and to optimize therapy, a better understanding of its dynamics is required. Currently, the standard for diagnosing and monitoring parkinsonian progress in patients is observation of visual feedbacks from them [2]. These may be insufficient since it is only monitored during follow-up visits. Research in disease monitoring has ranged from using mobile devices that have short message service, web-based applications and Bluetooth capability to measure the frequency of symptom onset so that medical interventions could be delivered or better diagnosis can be made [3]. These systems can be implemented on software applications running on the patient's commercial smartphone and connected to the clinician's information systems [4], for example, the WebBioBank, a web-based system for collecting clinical and neurophysiological data [5]. It is specifically created for deep brain stimulation (DBS) management, and can also be connected to the patient's mobile applications so that it can safely be used for web-based tele-monitoring and caregiver support [3]. Such disease monitoring can be used to provide a more refined therapy and for biomarker selection based on patient data collected.

The power required to transmit data in neural signal processing systems dominates that for recording and data conversion [6] and offline processing based on transmitting raw time series data as suggested in [3], [5], is an inefficient approach. Based on the power and bandwidth constraints involved in continuously sending neural signals, it will be more resource efficient to periodically send patient progress as state estimates after on-site and online analysis. Such an integrated platform for on-site and online analysis and monitoring of PD signals is still unavailable. For on-site and online analysis, there is a need to develop miniaturized real-time platforms that could monitor disease progress. These specialized hardware platforms would facilitate mobile diagnostics for better disease management. Portable platforms for PD diagnostics could provide more in-depth information and reduce the number of face-to-face visits required to optimize therapy. In PD monitoring, the aim is to provide long-term monitoring of the patient's condition for clinicians to better understand the symptoms so that therapy could be more accurately tailored to patients' needs.

This work presents an interface processor that can process local field potentials (LFP) [1] at the point of recording so that Parkinsonian states are communicated and logged onto an external platform. The processing chain is shown in Fig. 1. The analog-front-end (AFE) consisting of the low noise amplifier (LNA) and band-pass filter (BPF) are interfaced to an analog-to-digital converter (ADC). Digitized neural signals are sent to the PD detection processor. After PD events are detected they are sent over a communications link to a PD event log for monitoring. The system consisting of the LNA, BPF, ADC and PD detection processor is intended to be fully online and on-chip.

Manuscript received January 06, 2018; revised March 31, 2018. This work was partially supported by the Presidential Special Scholarship Scheme for Innovation and Development (PRESSID), Nigeria.

A. Mohammed and A. Demosthenous are with the Department of Electronic and Electrical Engineering, University College London, Torrington Place, WC1E 7JE London, U.K. (e-mail: ameer.mohammed.13@ucl.ac.uk; a.demosthenous@ucl.ac.uk).

[1] LFP is the electric potential recorded in the extracellular space in brain tissue, typically recorded using microelectrodes. LFP have temporal structure mainly in the frequency range of 0-100 Hz.

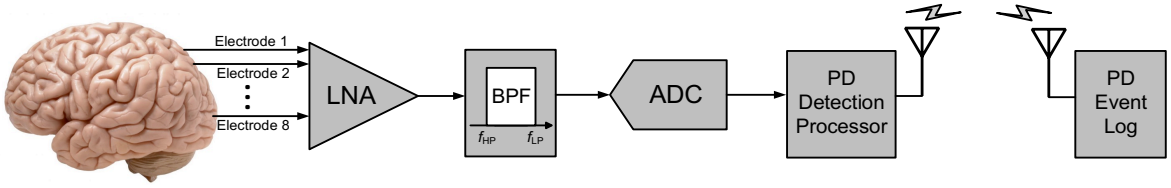

Fig. 1. The functional block diagram of the PD monitoring platform.

The PD detection processor detects PD events and sends them over a communication interface (instead of sending raw LFP data) to a PD event log. This reduces the amount of data sent over the communication link, which in turn reduces communication energy as well as bandwidth requirements of the system. Detection performance is assessed by validating various k-nearest neighbors (k-NN) classifier configurations on a field programmable gate array (FPGA) platform. In addition, on-chip power and area estimates for each of the PD classifiers is obtained by mapping the implemented designs to a 45 nm CMOS process. The main motivation for using 45 nm CMOS process is that it provides an optimal balance in terms of fabrication cost and on-chip power. The long-term plan is to implement fully implantable systems that manage DBS therapy.

The rest of paper is organized as follows. Section II describes the FPGA prototyping platform for the processor. Section III details the major functional units of the PD detection processor. The measured results are described in Section IV. Discussion and concluding remarks are presented in Section V and Section VI respectively.

II. Hardware Implementation

The objective is to provide a hardware platform for real-time processing of acquired neural data so that PD events can be distinguished from non-PD events. A fully online implementation performs on-site and real-time PD detection so that only PD events are transmitted to caregivers or stimulation devices to trigger actionable outputs. Currently, the mechanisms of PD are still under debate and the PD detection algorithms may need to be updated as a deeper understanding of the mechanisms of PD are gained. The most suitable candidate is an FPGA-based platform since it provides more flexibility for investigating various implementations of the PD detector compared with an application specific integrated circuit (ASIC) implementation. An FPGA based platform offers the best compromise between adaptability and portability. Moreover, register transfer level (RTL) implementations used on FPGAs are easy to translate to ASICs and microcontrollers. The PD detection processor was programmed on an Artix-7 FPGA as shown in Fig. 2 (a). The FPGA based PD detection tool was implemented in three major layers: the MATLAB layer, universal asynchronous receiver/transmitter (UART) layer and the FPGA layer. Synthesised LFP test data are transferred from MATLAB to the FPGA board through the serial communication link [USB-JTAG cable in Fig. 2 (b)] to the Artix-7 FPGA for processing.

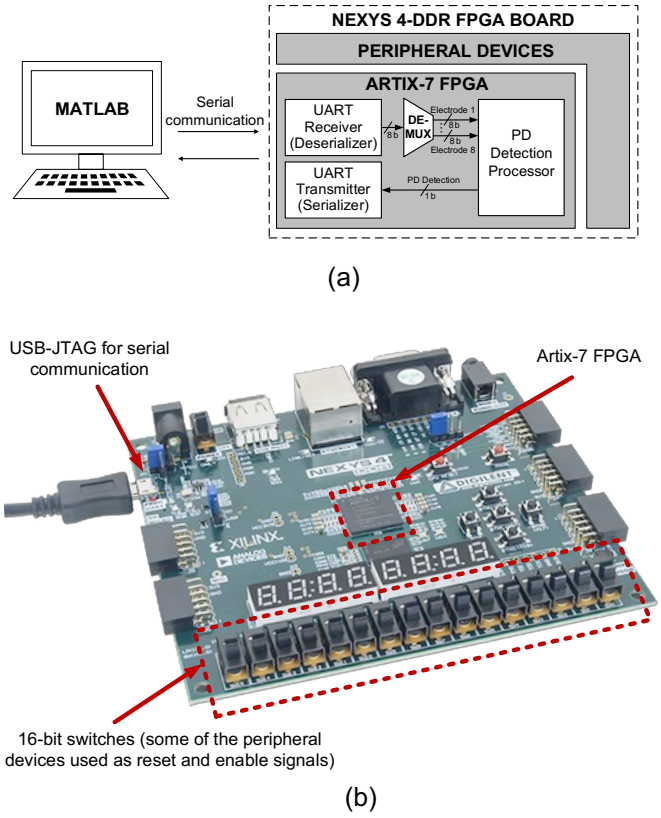

Fig. 2. Physical implementation of the PD detection processor. (a) Architecture; (b) Nexys 4 DDR FPGA board.

Processed data is sent back to MATLAB for performance evaluation of the PD detector (using signal ground truth stored in MATLAB).

A. Test Datasets

The original test datasets used were LFP recordings from the subthalamic nucleus (STN) of subjects exhibiting a combination of bradykinesia and/or rigidity during the onset of PD, with less noticeable tremor. Recordings were made from nine patients with PD who had bilaterally implanted DBS electrodes in their STN and are referred to as dataset A–I. The datasets contained separate ON and OFF levodopa (L-dopa) data between 5 to 10 minutes long. The ON and OFF L-dopa LFP data are used as representative non-PD and PD data respectively. The data was obtained from the Department of Clinical Neurology, University of Oxford. Recordings were made prior to the connection of a subcutaneous DBS pacemaker and stimulation was stopped during recording. Details on the daily drug dosage, on and off UPDRS score and

dominant symptoms for eight of the nine patients are summarized in [7]. The permanent quadri-polar macro-electrode used was model 3389 (Medtronic Neurologic Division, Minneapolis, MN) consisting of 4 platinum-iridium cylindrical contacts. Its contacts are numbered 0, 1, 2 and 3, with 0 being the most caudal and 3 being the most cranial for both right and left electrodes – making a total of eight monopolar channels for each patient.

The semi-synthetic LFP signals consist of PD and non-PD semi-synthetic templates created from the original LFP recordings. The LFP synthesis, involved fitting autoregressive moving average (ARMA) models to the real LFP recordings to produce semi-synthetic LFP templates. The semi-synthetic templates are concatenated to create PD and non-PD episodes of long duration. The duration of the PD and non-PD episodes are defined in a pseudorandom manner using a Poisson distribution. Fitting an ARMA model provides the flexibility to manipulate the signal characteristics so that all underlying conditions can be represented. The complete LFP data synthesis process and a detailed description of the LFP recordings are provided in [8].

B. FPGA Implementation

FPGA is a hardware platform that is configured using hardware description language (HDL). Compared to microcontrollers and ASICs, FPGA provides the best trade-off between speed and flexibility. FPGAs have a mix of the fixed architecture in ASIC and the structured programme execution style synonymous with microcontrollers [9]; this makes them in between the two extremes regarding speed and flexibility and serves as a good compromise. It is for this reason that the hardware-efficient implementation of the PD detector is validated on an FPGA platform. Also, since it is structured to perform complex operations in parallel as in ASICs and the long-term goal is to implement fully online and real-time implantable ASICs that can be deployed for PD monitoring and DBS modulation, it serves as a design step towards ASIC implementation. A further important reason for the choice of an FPGA based PD detector, is the need to implement and validate on a platform that provides flexibility to investigate the performance of various functional units and update the PD detection algorithm.

The hardware used is the Nexys4 DDR processing board from Xilinx with Artix-7 FPGA shown in Fig. 2(b). The Nexys4 DDR uses its own expansion system and has 60 I/O pins that can be interfaced to external devices. The board uses 3.3V I/O. It has on board peripheral devices that are accessible to the Artix-7 chip as peripheral I/O devices. Its FPGA is the XC7A100T-1CSG324C [10]. In addition to the FPGA chip, the Nexys4 DDR board has a number of peripheral devices such as LEDs, switches, temperature sensor, accelerometer, a speaker amplifier, microelectromechanical systems (MEMS) digital microphone, and a number of input and output devices for a variety of interfaces [11]. The USB-JTAG port is used for FPGA programming and data streaming through UART. For communication with a host PC and programming the Artix-7 FPGA, it uses a USB mini-B connector.

C. Input/Output Interface

Semi-synthetic LFP signals (Section II.A) are fed to the FPGA from MATLAB. LFP samples are quantized to 8-bit fixed point representation. Serial communication is implemented using UART protocol. From Fig. 2(a), the input data from MATLAB is divided into packets of 8 bits that are sent in serial format to the FPGA platform. The UART receiver implemented on FPGA receives serial input data from MATLAB, buffers them until a complete word (8-bits) is obtained before it sends it as input to the PD detection processor. The UART data packets are sent with a channel identifier packet to determine the recording channel from which the LFP signal originates. The packet and channel identifier are used to demultiplex the input signal to the appropriate channel of the PD detection processor, where LFP epochs are classified as '1' (PD) or '0' (non-PD) binary events via the UART transmitter to MATLAB.

III. System Overview

The PD detection processor performs feature extraction, feature selection and classification. Its top-level diagram is shown in Fig. 3. It consists of five major functional units: feature extraction for training, feature extraction for detection, PD detection processor finite state machine (FSM), memory banks and k-NN unit. The processor has two operating regimes: concurrent training and detection, and detection only. During concurrent detection and training, time multiplexing is used to acquire training data via the feature extraction for training unit from each of the recording channels. The training data is stored in the memory banks. It is used to train the feature selection (for feature and channel selection) as well as the k-NN classifier. To avoid interrupting PD detection, two feature extraction engines are used: the first is active only during training, and the second is active all the time and is used for detection. The PD processor FSM coordinates all these units.

The PD detection processor has four input clocks. *clk1* operates at 128 Hz and controls real-time acquisition of input LFP data. *clk2* operates at 1 Hz and controls feature extraction (and PD event log). Features are computed from 256 samples/channel, with 50% overlap in samples. Five features are computed from the buffered 256 samples. Feature extraction is necessary because for direct use of time-series data as input to classifiers for detection, the dynamic power of the classifier may be impractical for real-time detection. There are preliminary processing stages before classification; feature extraction and feature selection. The maximum ratio method (MRM) is used for feature and channel selection [8]. It has two operating clocks: *clk1* and *clk3*. MRM training is conducted sequentially per channel and is controlled by *clk3*, whose frequency varies for various configurations of the PD detection processor. It varies between 1/81 Hz to 1/756 Hz depending on the configuration (the various configurations are discussed in later sections). The final clock is *clk4*, which is only used by the 'PD detection processor FSM' to trigger a new training cycle; *clk4* is 200 times slower than *clk3*. All

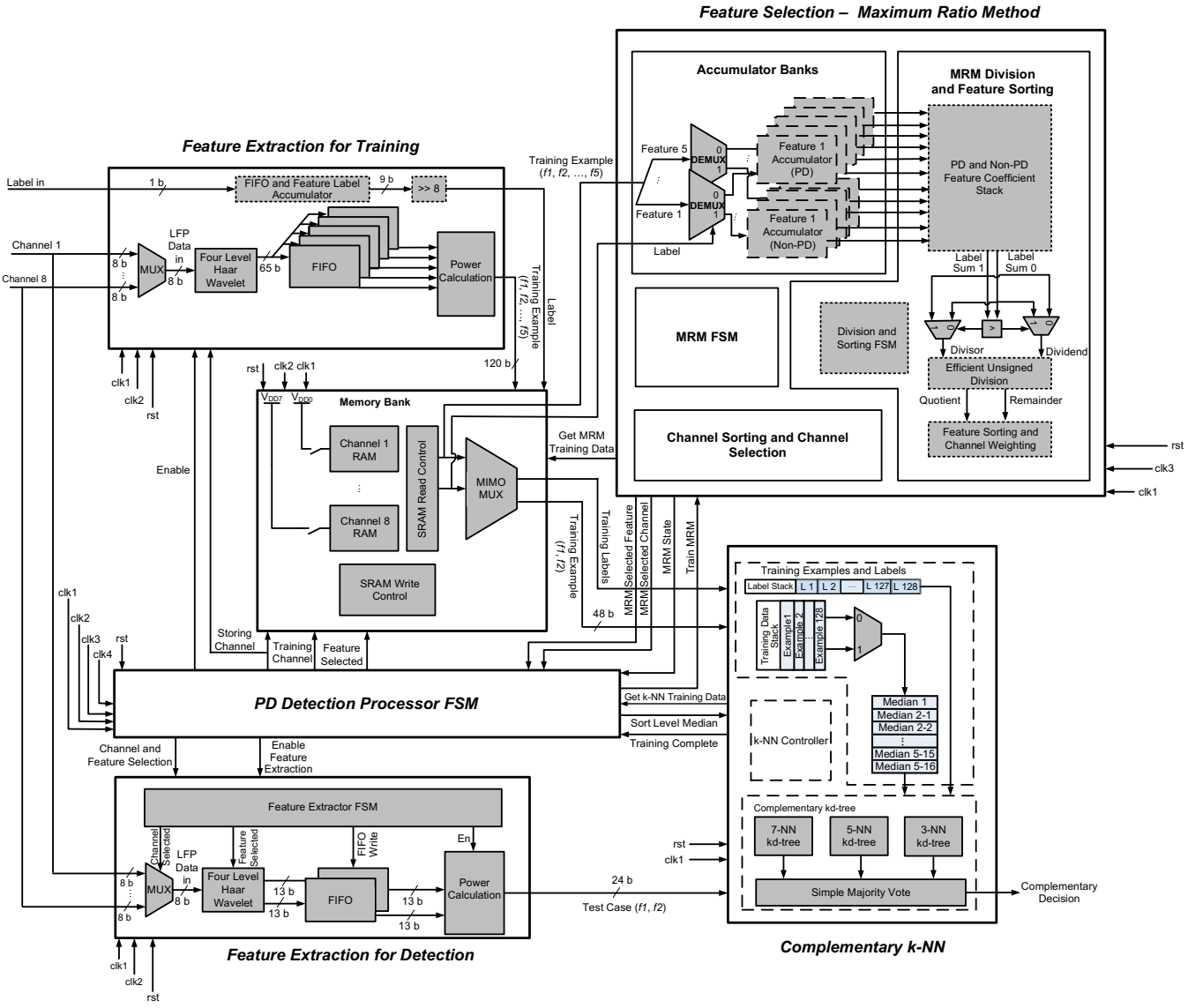

Fig. 3. The functional units of the PD detection processor.

clocks are controlled from a 1024 Hz master for a maximum clock frequency for *clk1* of 128 Hz.

Since fully implantable hardware is expected to operate under high-reliability requirements and strict power-density regimes, algorithmic and hardware optimizations that strike the optimal balance between efficacy and complexity need to be investigated. It is for this reason various configurations of the PD detection processor are investigated. The following sections describe the functional units in more detail.

A. Feature Extraction

A common algorithm that is widely used for time-frequency analysis of neural signals is the discrete wavelet transform (DWT). To extract useful information, DWT decomposes a signal into different levels based on frequency content. DWT is suitable for feature because the decomposition into different levels enhances the signal to noise ratio of the neural signals, which facilitates the identification of PD and non-PD events. The Haar wavelet has been commonly used due to its favourable balance between complexity of hardware implementation and detection performance. The Haar wavelet is ideal for capturing non-continuous frequencies [12]. Also, Haar-wavelets have been shown to be suitable in hardware-aware implementations for time-frequency analysis [13].

An approach using four-level Harr wavelet decomposition was used here because it separated features into the desired brain wave bands as seen in Fig. 4:

- Gamma band activity is greater than 30 Hz. Level 1 detail coefficients produce LFP characteristics between 32–49.5 Hz. The input LFP signal is band-pass filtered between 0.5–49.5 Hz, and then down-sampled to fs = 128 Hz and the maximum frequency is 64 Hz.
- Beta band activity is between 13–30 Hz. Level 2 detail coefficients characterise LFP activity into frequencies between 16–32 Hz.
- Alpha band activity is 8–12 Hz. Level 3 detail

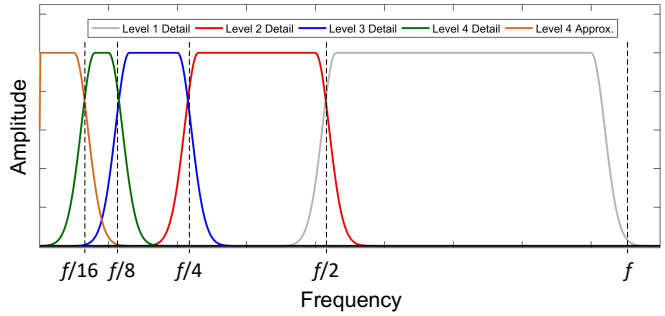

Fig. 4. A frequency domain illustration of four-level wavelet decomposition. ($f = f_s/2$, where f_s = sampling frequency).

coefficients are representative of LFP activity of frequencies between 8–16 Hz.

- Theta band activity is between 3–8 Hz. Level 4 detail coefficients of the LFP activity are between 4–8 Hz.
- Delta band is between 0.5–3 Hz. The level 4 approximation coefficients obtained represent LFP activity between 0.5–4 Hz.

Each level of decomposition is down-sampled by two at each successive level. More decomposition levels may not be useful because it results in reduced frequency bands which may contain little or no relevant information.

1) Hardware implementation

The two operating phases of the PD processor FSM are concurrent detection and training and detection only. Two feature extraction units are used because during concurrent detection and training, a separate feature extraction stage is used to store training data from each of the channels. To extract the five features (sub-band LFP power) in hardware, three major blocks are required:

- A four-level Haar DWT block that computes the wavelet coefficients at each decomposition level.
- A power calculation block that computes the features at each level based on their corresponding coefficients.
- A first-in, first-out (FIFO) memory block to synchronise the four-level Haar wavelet block with the power calculation block.

Fig. 3 shows the structure of the two implementations of the feature extraction units. The four-level Haar wavelet block is synchronised to operate at the same sampling rate as the input data (*clk1* = 128 Hz) because the system is designed to adopt real-time PD detection. However, features are updated every second and the power calculation block is synchronised to operate at *clk2* = 1 Hz. The power features are obtained using a 2-second window (consisting of 256 samples) of LFP signals with 50% overlap between windows so that features are updated every second. The average power of the coefficients in each level is:

$$P_i = \frac{1}{N_i} \sum_{n=0}^{N_i-1} |x[n]|^2 \qquad (1)$$

where N_i is the number of samples $x[n]$. N_i is 128 for level 1 detail coefficients, 64 for level 2 detail coefficients, 32 for level 3 detail coefficients and 16 for level 4 coefficients. The four-level Haar wavelet block is synchronised to the power calculation block using a FIFO memory. The FIFO block is a dual-port RAM consisting of a memory and controller block. It has separate read and write pointers that are used for controlling reading and writing operations in the feature extraction unit. Below is a brief description of the two implementations of the feature extraction units.

a) Feature extraction for training: All coefficients from the four-level Haar wavelet are transferred to the power calculation block through a five-level FIFO corresponding to coefficients from each level. The FIFO for level 1 detail coefficients is made up of 128 memory locations, level 2 detail uses 64 memory locations and so on; with level 4 approximation coefficients consisting of 16 memory locations. The 'feature label accumulator and FIFO' block collects ground truth information from training data. Each epoch of training data consists of a stream of labels for input samples which are used as ground truth information for training purposes as well as to validate performance.

b) Feature extraction for detection: Since 50% of buffered coefficients are reused by the feature extractor, the PD processor FSM controls reading samples from the FIFO. However, the number of samples buffered varies depending on the decomposition level of the DWT. The feature extractor controls the reading and writing based on the feature selected. The channel with the most pronounced variation is adopted. The best two of the five features from the adopted channel are used for classification.

2) Four-level Haar wavelet

To obtain the detail and approximation coefficients at the i-th level, approximation coefficients from the previous level a_{i-1} serve as input as in (2). These are convolved with a half-band low-pass filter (LPF) h_0 generating the approximation coefficients a_i, and with a half-band high-pass filter (HPF) g_0 to generate the detail coefficients d_i. This can be represented mathematically as,

$$a_i(k) = \sum_n h_0(n)\, a_{i-1}(2k-n) \qquad (2)$$

$$d_i(k) = \sum_n g_0(n)\, a_{i-1}(2k-n) \qquad (3)$$

where n represents the index of the filter coefficients (low and high pass filters), k is the index of the input signal (approximation coefficients, a_{i-1} and detail coefficients, d_{i-1}). In the first part of the equations for both approximation and detail coefficients there is a down-sampling by two at each level before filtering. Computationally efficient Haar wavelet adopts the polyphase implementation in [13]. The filters for the Haar wavelet are 2-tap FIR filters given by $g_0 = {}^1/_{\sqrt{2}}$ [1, -1] and $h_0 = {}^1/_{\sqrt{2}}$ [1, 1].

B. Memory Bank

To facilitate concurrent training and detection, and efficient sharing of other computational resources, the PD detection processor has a memory bank for storing training data. The major functional units of the memory bank are shown in Fig. 3, which include an eight-channel random access memory (RAM), a static RAM (SRAM) write control and an SRAM read control. Each channel RAM is used for storing training data from its corresponding channel. The memory locations of the channel RAMs are scalable depending on the number of training examples required. The number of training examples stored depends on the configuration of the PD detection processor (mainly determined by the k-NN configuration adopted). The training data for each channel consists of half PD examples and of non-PD examples. This is to ensure generalizability for both PD and non-PD events. Training data for feature and channel selection (five features/training example) as well as k-NN training data (two features/training example) after channel selection, are obtained from the memory bank. The memory bank uses power and clock gating to reduce power consumption.

C. Feature Selection

Feature selection chooses the most relevant features to reduce the memory and computational resources. It also reduces data over-fitting at the classification stage, since some features are noisy and can lead to degradation in classifier performance. Feature selection results in significant reductions in the area and power of the PD detection hardware. For multichannel application, the classification phase has to be trained channel by channel either in sequence or parallel, and the best performing channel is selected for use. This can be computationally intensive. Alternatively, the MRM estimates the most informative channel and uses the features from this channel (as against using all channels [8]). The complete procedure for feature selection using the MRM is described in [8]. Fig. 3 shows the function blocks of the MRM unit: accumulator bank, MRM division and feature sorting, channel sorting and selection, and the MRM FSM. The latter controls the MRM training for each channel and the ranking of channels based on the separability of their PD and non-PD classes. Feature and channel ranking are time-multiplexed to enable logic reuse.

The MRM process starts by streaming the training data of each channel sequentially. For each channel, training examples for PD and non-PD events are accumulated and stored in the PD and non-PD coefficient stack in Fig. 3. The MRM dividend and divisor are determined for each of the five features (of each channel) for PD and non-PD training examples. The 'division and sorting FSM' controls this. Both PD and non-PD events have five features. The separability of each feature is determined using the ratio of the mean for features of PD and non-PD examples. Subsequently, the ratio of all the features for each channel is obtained, and the channel weight is obtained using the 'feature sorting and channel weighting' unit. The channel weights and feature ranks are sent to the 'channel sorting and selection' unit, where the channel weights and feature ranks of each channel are stored. The process is repeated for all the other channels. This is used to assess the most separable channel and features for use in PD detection. The channel and features with the most pronounced variation are communicated to the PD processor FSM.

D. PD Classifier

Machine-learning algorithms offer the benefit of understanding disease progression in patients. However, their computations are not well supported by conventional DSP platforms; particularly when high order models are used. Pre-processing stages like feature extraction and feature selection are necessary to reduce the computational demands of detection algorithms. In PD monitoring, the classifier is required to select PD events from non-PD events using acquired neural signals. The PD classifier produces a binary output; high is a PD event and low a non-PD event. The detection results are intended for use either by caregivers or are interfaced to stimulation devices. PD monitoring and event detection can be implemented either by using online or offline classification. This work has implemented fully online classification to facilitate real-time PD detection.

The PD classifier employs a k-NN classifier. Its overall architecture is shown in Fig. 3. The k-NN classifier consists of four functional units: training examples and label block, level median stack, k-NN controller and kd-tree block. Training data from the memory bank is stored in the 'training examples and labels' block. This data is used to obtain the various medians for different levels of the kd-tree search (nearest neighbors are obtained using kd-tree search). The medians in the level median stack are used for kd-tree implementation of the k-NN classifier. The k-NN classifier uses a kd-tree approach. The kd-tree was chosen because it uses a hardware efficient implementation of k-NN [14]. The kd-tree distance metric uses the minimum number of computations, but it has reduced accuracy. It is for this reason this work investigates its use in different k-NN configurations, which are investigated using different levels of the kd-tree search as well as nearest neighbors. The PD classifier in Fig. 3 uses kd-tree implementation in a complementary k-NN configuration where consensus is established using majority voting between different implementations to produce a classifier with a confidence level stronger than that of the three disparate and weak classifiers in the configuration. The following section provides more detail on kd-tree implementation.

1) k-dimensional tree implementation

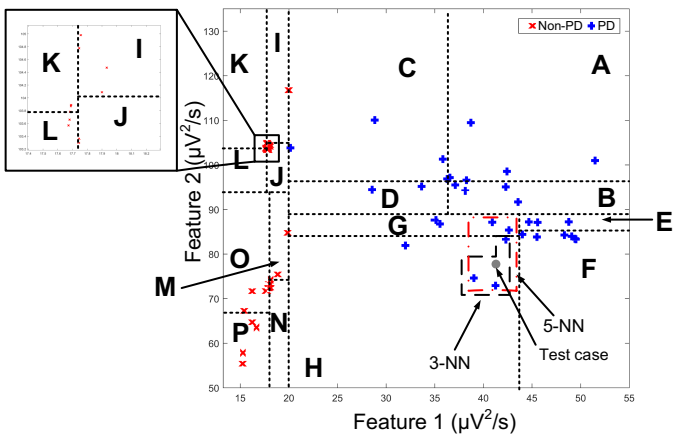

Fig. 5. Feature space depicting kd-tree compartmentalization.

PD classification using k-NN usually compares the input feature vector to the k-closest training examples. This distance is computed using different metrics, namely; Euclidean [15], L^P distances [16], Mahalonobis distance [17], and approximate distance metrics like kd-tree implementations. Among these, the kd-tree implementation is the least computationally intensive [14]. It uses a binary decision tree to drill down n-levels, with each level generating a splitting hyperplane that divides the space into two parts, known as half-spaces. The splitting hyperplane is chosen such that every node in the tree is related to one of the k-dimensions (in this case two-dimensions, since there are two dimensional features) and its direction is perpendicular to the axis it splits as indicated in Fig. 5 which shows a four-level kd-tree search. The feature space is divided into 16 compartments by the kd-tree search. For an N level kd-tree, there are 2^N nodes.

In Fig. 5, for level 1, a splitting hyperplane is chosen at the median of the x-axis (feature 1) values, all points with a value below the median are categorized in the left sub-tree (consisting of compartments labelled I–P) and those greater than the median are categorised to the right tree (with compartments labelled A–H). In this situation, the hyperplane is determined by the x-axis. However, for level 2 splitting, two new hyperplanes are used to further split the left and right compartments of the feature space into four compartments. The process of binary splitting returns to the x-axis in level 3, which further splits each of the four compartments into eight compartments. Level 4 splitting then splits the eight compartments into 16 compartments. These 16 compartments are shown for the feature space of dataset C, with each compartment having four training examples making a total of 64 training examples. Fig. 5 shows how the 3-NN can be used to classify a test case, as the three closest training examples from compartment 'H' are used. However, to use 5-NN or 7-NN classification, only three level kd-tree is required. For 5-NN or 7-NN classification, the nearest neighbours from compartments 'G' and 'H' in Fig. 5 are used in classifying the test data. This work uses a maximum of a five level kd-tree. Various implementations for the 3-NN, 5-NN, 7-NN and the complementary k-NN (which will be indicated by X-NN from now onwards) are discussed in the following section.

2) k-NN configurations

Different levels of kd-tree search, as well as nearest neighbors, are implemented and tested for accuracy and hardware resource trade-offs. For k-NN, the best value of k is very dependent on the dataset. For k-value selection, a larger k-value suppresses the effects of outliers. However, it creates less distinct decision boundaries [18]. It is for this reason various orders and configuration of nearest neighbors algorithms are investigated.

The classifiers studied include k-NN with three, five and seven neighbors; and an ensemble of the classifiers which is the complementary configuration. The 'k-NN controller' block in Fig. 3 controls the sorting of features in the training examples and the storing of the hyperplane (median) points for each level of the kd-tree. In the complementary configuration, a multi-classifier vote is adopted based on the outputs from each of the three classifiers and a simple majority vote is used to generate a consensus. Below is a brief description of the various implementations.

a) 3-NN kd-tree: These implementations are designed to terminate the search at nodes with three training examples such that each test case is classified according to the class dominated by the node it falls into, using simple majority vote. Configurations using three, four and five levels kd-tree having 24, 48 and 96 training examples respectively, are investigated for accuracy and hardware resource trade-offs.

b) 5-NN kd-tree: These implementations are designed to terminate the search at nodes with five training examples. Configurations using three, four and five levels kd-tree using 40, 80 and 120 training examples respectively, are investigated for accuracy and hardware resource trade-offs.

c) 7-NN kd-tree: These implementations are designed to terminate the search at nodes with seven training examples. Configurations using three, four and five levels kd-tree using 56, 112 and 224 training examples respectively, are investigated for accuracy and hardware resource trade-offs.

d) Complementary kd-tree: In this configuration, a multi-classifier vote is adopted based on the outputs from each of the three classifiers (3-NN, 5-NN and 7-NN). A simple majority vote is used to generate a consensus between the three classifiers. The kd-tree configuration consists of nodes with four training examples such that 3-NN could be obtained from the final nodes. 5-NN and 7-NN are obtained from the second-to-last level, since it is a node with eight training examples. A typical example of this implementation is shown in Fig. 5, which uses 3/4 kd-tree levels. 5-NN and 7-NN can be obtained at level 3, while 3-NN is obtained at level-4. Implementations using 3/4 kd-tree levels, which have 64 training examples and implementations using 4/5 kd-tree levels, which have 128 training examples are investigated for accuracy and hardware resource trade-offs. Table I summarises the various kd-tree

TABLE I
SUMMARIZING THE VARIOUS *kd*-TREE IMPLEMENTATIONS INVESTIGATED

k-NN Implementation	kd-tree Levels	Number of Training Examples	Relative Training Frequency*
3-NN	3	24	5.33
	4	48	2.67
	5	96	1.33
5-NN	3	40	3.20
	4	80	1.60
	5	160	0.80
7-NN	3	56	2.29
	4	112	1.14
	5	224	0.57
X-NN	3/4	64	2
	4/5	128	1

* Training frequency measured relative to X-NN (4/5), which is trained once a day.

implementations and their training data requirements.

E. Operating Modes

The PD detector has two modes of operation: concurrent detection and training, and detection only. During both modes, real-time PD detection continues and both operating modes use fully on-site computation. During concurrent detection and training, the PD detector concurrently detects PD events and trains to determine the appropriate input channel and features for PD monitoring as well as the kd-tree hyperplanes for each level. This mode requires the most computing and memory requirement. Nevertheless, it only lasts for 5.5% of the time. It involves three sub-modes: training example storage, MRM training and k-NN training. These sub-modes take 4%, 1% and 0.5% of the time respectively. Table II summarizes the active functional units during each of the sub-modes of concurrent detection and training. The second mode is PD detection only, which happens most of the time. During this mode, the PD detector transmits PD events at one-second intervals. Only three functional units are active for majority (94.5%) of the time as highlighted in Table II.

TABLE II
OPERATING MODES OF THE PD DETECTION PROCESSOR

Operating Mode	Sub-Mode	Duration (×1/*clk3*)†	Active Functional Units (aside from processor FSM)
Concurrent Detection and Training	Training example storage	8	Feature extraction for training, memory bank, feature extraction for detection and k-NN.
	MRM training	2	Memory bank, MRM, feature extraction for detection and k-NN.
	k-NN training	1	Memory bank, feature extraction for detection and k-NN.
Detection only	–	189	Feature extraction for detection and k-NN.

†*clk3* varies for each implementation. For X-NN (4/5) it is 1/432 Hz; this frequency has to be multiplied by the relative training frequency in Table I to obtain the equivalent value of *clk4* for other implementations of the PD detection processor.

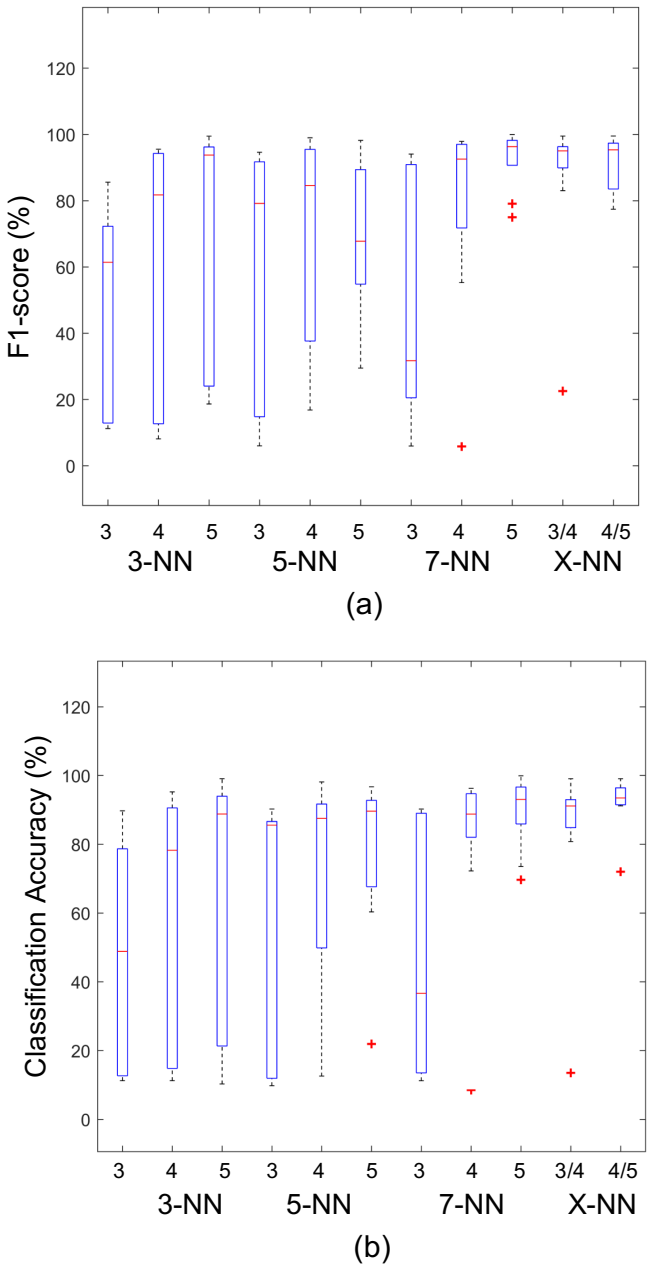

Fig. 6. Detection performance for various implementations of the PD detection processor (number of training examples for each configuration summarized in Table I).

IV. PERFORMANCE EVALUATION

To evaluate the performance of the proposed PD detection system, semi-synthetic neural signals constructed from real LFP recordings were used (Section II.A). A major bottleneck in validating implantable electronics is access to patients due to the necessary detailed clinical regulatory oversight. Semi-synthetic neural signals were preferred to experimentally-recorded neural signals because all underlying conditions in the original LFP signal can be modeled alongside the ground truth information. In addition, semi-synthetic signals offer the opportunity to synthesise longer LFP recordings from real LFP recordings with short duration. Detection performance

TABLE III
SUMMARIZING COMPLEXITY AND MEAN ACCURACY FOR VARIOUS IMPLEMENTATIONS

S/No	Implementation		Complexity Measures (per channel)		Accuracy Measures		
			(μW)	(mm^2)	Mean MCC	Mean F1 (%)	Mean class. acc. (%)
1	3-NN	3	2.28	0.0433	0.0012	51.29	49.74
2		4	2.37	0.1175	0.2479	56.69	55.70
3		5	2.31	0.1915	0.3513	68.20	64.76
4	5-NN	3	2.23	0.0465	0.1608	56.12	60.00
5		4	2.34	0.1651	0.3275	68.43	69.48
6		5	2.53	0.2873	0.5081	68.11	77.96
7	7-NN	3	1.80	0.0496	0.1315	50.20	51.18
8		4	2.49	0.2126	0.4820	78.37	79.79
9		5	2.73	0.3815	0.6422	92.68	89.66
10	X-NN	3/4	2.22	0.1414	0.3542	86.18	82.14
11		4/5	2.26	0.2385	0.6162	91.38	91.93

and hardware resource utilisation were observed. Detection performance results were obtained from the FPGA and estimate of hardware resources were obtained both from post-synthesis power and area estimates when mapped onto a 45 nm CMOS process. Design of ASICs for PD detection and monitoring has not been considered largely due to insufficient empirical evidence on the behavior of the DBS mechanism. It may not be cost-effective to fabricate the processor on a silicon chip. At this stage of development an adaptable platform such as a FPGA is required.

A. Detection Performance

For the hardware test, performance measures were obtained over a complete training and test period *clk4*. It has a frequency of $1/86400$ Hz for the X-NN (4/5) implementation. For other implementations the frequencies are multiples of X-NN (4/5) *clk4* as summarised in the relative training frequency in Table I. Before performance evaluation, training is conducted which lasts 5.5% of the time. Then for the other 94.5% of the time, test cases are detected. The box plots in Fig. 6 (a) are the F1-score for the various configurations. It can be seen that the F1-score increases with an increase in the number of training examples, except in the case of 5-NN (5), 7-NN (3), X-NN (3/4) and X-NN (4/5). In the case of 5-NN (5), this could be attributed to outliers in the training examples due to an increased number of training examples. Outliers tend to bias the classifier model. In the case of 7-NN (3), the use of seven nearest neighbours in a training set consisting of 56 training examples can increase susceptibility to outliers i.e. if outliers are a large part of the training examples.

With a median F1-score of 95.02%, the X-NN (3/4) has a better performance than the 5-NN (5) and 7-NN (4) which have 67.75% and 92.54% respectively. However, they use more training examples than the X-NN (3/4). The improved performance in X-NN (3/4) can be attributed to the complementary detection it uses. For the X-NN (4/5), its median F1-score trails that of the best, which is the 7-NN (5)

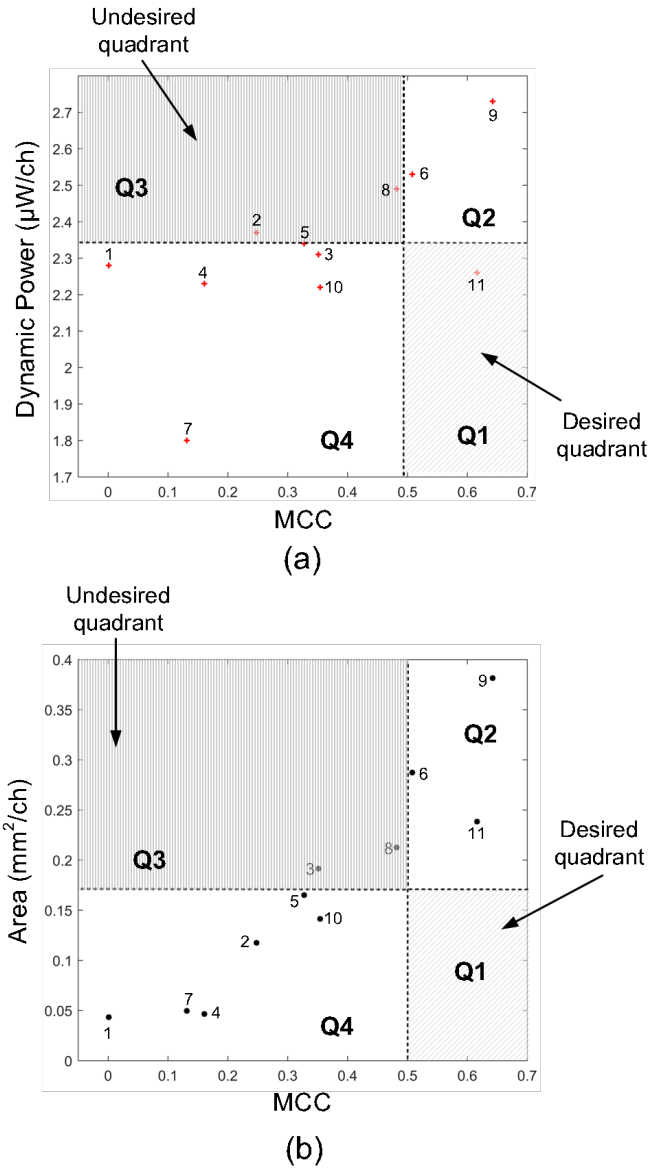

Fig. 7. On-chip power and area. (a) MCC vs dynamic power; (b) MCC vs area.

by less than 1% even though the 7-NN (5) uses 224 training examples compared to 128 training examples used by the X-NN(4/5). The top three implementations in terms of F1-score are: 7-NN (5) with a median of 96.29%, X-NN (4/5) with a median of 95.34% and X-NN (3/4) with a median of 95.02%.

The classification accuracy measures in Fig. 6 (b) follow almost the same pattern as the F1-score in Fig. 6 (a). However, the major difference is that the X-NN (4/5) has a better classification accuracy performance than the 7-NN (5). The top three in terms of classification accuracy are: X-NN (4/5) with a median of 91.93%, 7-NN (5) with a median of 89.66% and X-NN (3/4) with a median of 82.14%. The effect of outliers was reduced using complementary detection to establish consensus between weak and disparate classifiers. Periodical updating the training examples also reduces the effects of outliers. These results clearly show that the complementary k-NN configuration improves detection

TABLE IV
SUMMARIZING FPGA RESOURCES

Configuration		LUT		Registers		Mux / Demux		DSP	
		Slices	(%)	Slices	(%)	Slices	(%)	Slices	(%)
3-NN	3	8377	13.2	6139	4.8	236	0.5	6	2.5
	4	11071	17.5	7492	5.9	619	1.3	6	2.5
	5	17135	27	10140	8.0	1078	2.3	6	2.5
5-NN	3	10284	16.2	6953	5.5	331	0.7	6	2.5
	4	15351	24.2	9033	7.1	987	2.1	6	2.5
	5	23855	37.6	13017	10.3	2512	5.3	6	2.5
7-NN	3	11514	18.2	7656	6.0	696	1.5	6	2.5
	4	17675	27.9	10646	8.3	1674	3.5	6	2.5
	5	29839	47.1	15997	12.6	3636	7.7	6	2.5
X-NN	3/4	14194	22.4	8284	6.5	885	1.9	6	2.5
	4/5	20993	33.1	11560	9.1	1904	4	6	2.5

performance.

Fewer training examples may result in limited complexity; however, this occurs at reduced accuracy. To obtain the configuration with the best trade-off in terms of accuracy and complexity, configurations using fewer training examples are made to train more frequently. Thus, training frequency is inversely proportional to number of training examples. Configurations having a higher number of training examples have a lower training frequency. The training frequency and training examples are staggered so that the average number of training examples used by each configuration per day is the same. In pattern recognition, a high number of training examples results in high variance which could cause data overfitting. While a low number of training examples could lead to poor generalizability [18]. Periodically updating the training examples can be used to overcome issues such as poor generalizability and data overfit.

B. On-chip Power and Area

For ASIC resource requirements, various implementations of the architecture were synthesised and mapped into the 45 nm NanGate digital cell library [19]. The power is analysed based on a core voltage of 1.1 V. The 7-NN (3) configuration has the lowest dynamic power requirements, while the 3-NN (3) has the lowest area requirements. The resulting estimates of dynamic power and area per channel for different configurations of the PD detection processor are summarized in Table III. Generally dynamic power is expected to increase with an increase in the number of training examples. However, this is not the case for the 3-NN (4), which is an outlier. Dynamic power is mainly due to clock signals, which makes it dependent on the architecture of the controller. It can then be deduced that the control architecture adopted for the 3-NN (4) uses more control and gating signals than its counterparts. The area consumption is commensurate with FPGA resource utilisation. However, this is not the case for the dynamic power consumption, where there is an increase in clock activity for each implementation with an increase in the number of kd-tree levels used.

To put the complexity measures in perspective, Fig. 7 (a) shows the Matthews correlation coefficient (MCC) vs dynamic power and Fig. 7 (b) shows the MCC vs area. The various implementations of the PD processor are represented by the serial numbers in Table III. The plots are divided into four regions: Q1, Q2, Q3 and Q4. Q1 represents the implementations that have an average MCC greater than 0.5 and also have dynamic power lower than the average dynamic power for all implementations. In Fig. 7 (a), only X-NN (4/5) implementation is in this category. This is the most desirable outcome in terms of dynamic power, i.e. a classifier to producing a model that accurately classifies test cases as well as having a dynamic power lower than the class average. The other quadrants in Fig. 7 are: Q2 which represents an average MCC greater than 0.5 and a dynamic power greater than the average which is 2.32 μW/channel. This represents a good detector but has a higher power consumption than Q1. Q3 represents the unwanted condition, in which the detector consumes more power than the class average and produces an MCC that is less than 0.5, which signifies weak positive correlation. In the fourth region, Q4 the implementation results in a weak positive correlation (MCC < 0.5) and a power consumption less than the average for all the implementations. This may be more desirable than Q2 depending on what is more important between power consumption and detector performance measure in MCC. The X-NN (3/4) produces the best performance for the implementations in Q4. For the area consumption in Fig. 7(b), there is an almost linear increase in area with an increase in MCC. Fig. 7(b), divides the quadrants in a similar way to Fig. 7 (a). However, the quadrants on the y-axis are divided based on the average on-chip area of the configurations. None of the configurations fall into Q1. In Q2, the 5-NN (5), 7-NN (5) and X-NN (4/5) fall into Q2 where the X-NN (4/5) presents a better on-chip area trade-off than the others since it utilizes relatively less power. Q3 also represents the undesirable quadrant, with large area consumption and an average MCC below 0.5. In Q4, the best for MCC is the X-NN (3/4). These results demonstrate that the complementary configuration offers a good trade-off between complexity and accuracy.

C. FPGA Resources

The major building blocks for the FPGA implementation are: LUT slices, register slices, multiplexers and DSP units. The Artix-7 FPGA has 63400 LUT slices, 126800 register slices, 240 DSP slices and 47550 multiplexer slices [11]. The resource utilisation for the various implementations are summarised in Table IV. Only six DSP slices were used. These were used in the feature extraction unit. Other resources increase with an increase in nearest neighbours as well as kd-tree levels. The k-NN and memory bank dominate in terms of resources as the kd-tree level grows. The feature extraction units and the MRM stay approximately the same throughout. DSP slices are only used in the feature extraction block for multiplication to compute the power of the coefficients from the four-level Haar wavelet.

D. Comparison with Other Neurological Event Detectors

TABLE V
PERFORMANCE COMPARISON WITH STATE-OF-THE-ART NEUROLOGICAL EVENT DETECTION PROCESSORS

Reference	This work	[33]	[28]	[34]	[35]	[36]	[37]	[29]	[38]	[39]
Year	2017	2017	2017	2016	2015	2015	2014	2014	2013	2013
Application	PD onset detection	Epileptic seizure detection	Epileptic seizure detection	Epileptic seizure detection	Epileptic seizure detection	Epileptic seizure detection	Epileptic seizure detection	Epileptic seizure detection	Epileptic seizure detection	Epileptic seizure detection
CMOS tech. (nm)	45	350	130	180	180	180	130	180	180	180
No of channels	8	32	24	8	16	16	18	8	8	7
Neural signals	LFP	EEG	EEG, ECoG	EEG	EEG	iEEG	EEG	EEG	EEG	ECoG
Area/ch. (mm^2)	0.2384	0.0491	12 [a]	3.125	3.125	0.0625	0.0833	13.47 [a]	25 [a]	0.7436
Core voltage (V)	1.1	1.25	1.2	1.8	1.8	0.8	1.2	1.8	1.8	1.0
Dynamic power/ch. (µW)	2.26	0.95	1300 [b]	FE = 7, SVM = 12.56	2.45	0.85	5.94	2800 [c]	8.25	4.09
Efficacy (%)	F1 = 91.38, Class. acc. = 91.93, MCC = 0.62 [d]	Sensitivity = 87	Sensitivity = 88 – 96	Sensitivity = 95.1	Sensitivity = 95.7, Specificity = 98	Sensitivity = 100	Sensitivity = 91 – 96	Sensitivity = 92	Class. acc. = 84.4	Class. acc. = 93.1

[a] Gives the total area of the chip not area per channel.
[b] Total for 24 channels and other stages.
[c] Total for 8 channels and other stages.
[d] Not measured in %.

Table V summarises the characteristics of state of the art neurological event detection processors implemented since 2013. The implementation with the best efficacy, area and dynamic power trade-offs in this work, which uses the X-NN (4/5) classifier is adopted and compared with other neurological event detection processors. Published neurological event detection processors have focused on epileptic seizure detection. Comparison of PD detection with other neurological event detectors presented in Table V is confined to epilepsy detection.

This work performs on a comparable level in terms of area and power to other neurological event processors. The classifier used by this processor adopts k-NN classification, which is a population dependent algorithm that extrapolates properties more accurately when larger training sets are used. Hence, memory banks were adopted for storing training examples. This increased the effective area per channel of the processor. The detection processor developed in this work is the first processor for PD detection. So far, neurological event detection processors have focussed on epileptic seizure detection. For seizure detection, the major signals of interest are electroencephalography (EEG) and electro-corticography (ECoG), which mostly lie below 30 Hz, which are obtained from the scalp and cortex respectively.

In seizure detection, most studies use phase synchronisation between signals recorded from different channels as the biomarker of interest. Seizures are detected when phase synchronisation increases above a specific threshold. An increase in EEG or ECoG synchronisation between channels can be indicative of an epileptic episode. The main challenge of the phase synchronisation approach is that the baseband synchronisation varies across subjects and between channels. Nevertheless, neurological signals for epilepsy (EEG and ECoG) are more distinguishable than those of PD (LFP signals). A closed-loop system for epileptic seizure suppression has already gained FDA approval [20]. The unpredictable nature of LFP signals for PD and their higher frequency content compared to EEG and ECoG makes their processing more computationally intensive. LFP signals are used as biomarkers in PD detection and monitoring due to their stability and rich spatiotemporal content [21], which is necessary to overcome the unpredictable nature of PD. The fact that they can be obtained from existing stimulation electrodes means that they are minimally invasive. EEG signals are usually obtained from electrodes attached to the scalp. This makes them impractical for ambulatory disease monitoring as they are not implanted, which means they could limit patients' quality of life. Another issue is that correlation between PD and EEG signals are yet to be established or reported.

Regarding efficacy, most studies in epileptic seizure detection have focussed on sensitivity, which only measures true-positive rate. This means a randomly guessing processor that always returns a positive can achieve a 100% sensitivity since it will get all the actual positives (with 0% specificity). Therefore, more balanced measures may be required for more comprehensive assessment of processor efficacy. This is particularly necessary for situations where event detection informs therapy. Administering therapy when it may not be needed can result in side effects [22]. This is why more balanced measures like the MCC and F1-score are adopted in this work.

V. DISCUSSION

A. Personalized Health Monitoring in PD

In PD, proper health monitoring can lead to better PD mitigation as well as reduce the number of face-to-face visits by patients. Further understanding on how disease progresses in patient population can be achieved through remote health monitoring. This can involve the analysis of data from multiple patients so that adaptive DBS strategies that incorporate more universal features or biomarkers could be incorporated in tackling varying PD disorders. This will make

therapy more proactive and less retroactive. As more data is collected a better understanding of disease progression can be obtained, which could be used to refine therapy. Moreover, since diverse population of PD patients are afflicted with various dominant symptoms, implementations of personalised health monitoring systems can be tailored to cater for individual patient needs. Early detection of patient deterioration could help caregivers to immediately take action in modifying stimulation parameters or therapeutic paradigms [23]. This can be achieved by remote and continuous monitoring of the disease states. The information obtained from health monitoring systems is intended to be logged to monitor disease progression or relayed to caregivers in the event of an emergency. This can make therapeutic interventions more sustainable by early recognition of unresponsive symptoms so that alternative therapeutic regimes are provided [24].

Conventional DBS leads usually consists of 4 platinum-iridium cylindrical contacts. They are implanted either unilaterally or bilaterally. This means they consist of 4 or 8 electrodes. Recordings can be obtained from the same electrodes that are used for stimulation, making them minimally invasive [25]. An eight-channel design was adopted so that it is compatible with conventional DBS leads. DBS leads produced by all major manufacturers have a similar operating principle, with slight differences in technical features.

B. Supporting Technology

Advances in bio-sensing, signal processing, data communication and nanotechnology offer the possibility of fully implantable systems for disease monitoring. The major requirement is to devise systems that detect patient-specific physiological states in real-time using minimally invasive and low power devices. The development in machine-learning algorithms that are capable of exploiting statistical properties in the data to model specific correlations will facilitate more accurate decision making by healthcare experts or devices [26]. For disease monitoring using machine learning models, prior pre-processing is as important, or even more important than the use of the machine learning models themselves [27]. This is because the models are only as good as the input signals that are fed to them. The use of advanced LNA and ADC is essential to achieve high-performance detection. The state of the art systems in [28] and [29] consisting of sensing (LNA, BPF and ADC) and detection stages, utilize 1.3 mW and 2.8 mW respectively. Thus, total power consumption is below 3 mW.

C. Challenges

Healthcare monitoring systems are required to be operational for long periods of time. Monitoring systems like the LiveNet system [30], which uses an accelerometer, electrocardiography, electromyography and skin conductance to obtain information on PD and epileptic seizure by transmitting raw samples of data to external devices. Adopting this approach imposes high power and throughput requirements. This motivated the adoption of an implementation that can be deployed for on-site real-time processing. Due to the need for detection algorithms to be made implantable, there will be a demand for increased storage and computing capability to handle large and dynamically changing training data. Large training data may be required as classification accuracy improves as more knowledge on the disorder is incorporated into the processor. Other challenges confronted by personalised health monitoring systems are security and privacy issues [31]. Security and privacy is mainly dependent on the communication method used for transmitting PD events. A serious effort is already invested in research on encryption to protect from data interception and tampering [32].

VI. Conclusion

In this paper, complementary PD detection was presented as a hardware-efficient method for on-site PD detection. The PD detection processor is the first known implementation for on-site and real-time monitoring. The design leveraged on the flexibility of FPGA to test the performance of various configurations of the PD detection processor. Estimates for power and area were obtained for 45 nm CMOS technology. The PD detection processor presented a comparable level of performance to state-of-the-art neurological event detectors. Since spectral features are useful as biomarkers in other neurological applications, the proposed approach may find a range of different applications and implementations. For on-site PD monitoring, the challenge is the realization of an efficient processor that can handle the complexity of physiological signals and still meet the requirement for implantation in chronic applications.

References

[1] S. J. Schiff, "Towards model-based control of Parkinson's disease.," *Philos. Trans. A. Math. Phys. Eng. Sci.*, vol. 368, no. 1918, pp. 2269–308, May 2010.

[2] J. Volkmann, J. Herzog, F. Kopper, and G. Deuschl, "Introduction to the programming of deep brain stimulators.," *Mov. Disord.*, vol. 17 Suppl 3, pp. S181-7, Jan. 2002.

[3] S. Marceglia, E. Rossi, M. Rosa, F. Cogiamanian, L. Rossi, L. Bertolasi, A. Vogrig, F. Pinciroli, S. Barbieri, and A. Priori, "Web-based telemonitoring and delivery of caregiver support for patients with Parkinson disease after deep brain stimulation: protocol.," *JMIR Res. Protoc.*, vol. 4, no. 1, p. e30, Mar. 2015.

[4] M. H. Myers, M. Threatt, K. M. Solies, B. M. McFerrin, L. B. Hopf, J. D. Birdwell, and K. A. Sillay, "Ambulatory Seizure Monitoring: From Concept to Prototype Device.," *Ann. Neurosci.*, vol. 23, no. 2, pp. 100–11, Jul. 2016.

[5] E. Rossi, M. Rosa, L. Rossi, A. Priori, and S. Marceglia, "WebBioBank: A new platform for integrating clinical forms and shared neurosignal analyses to support multi-centre studies in Parkinson's Disease," *J. Biomed. Inform.*, vol. 52, pp. 92–104, 2014.

[6] N. Verma, A. Shoeb, J. Bohorquez, J. Dawson, J. Guttag, and A. P. Chandrakasan, "A Micro-Power EEG Acquisition SoC With Integrated Feature Extraction Processor for a Chronic Seizure Detection System," *IEEE J. Solid-State Circuits*, vol. 45, no. 4, pp. 804–816, Apr. 2010.

[7] S. Little, A. Pogosyan, S. Neal, B. Zavala, L. Zrinzo, M. Hariz, T. Foltynie, P. Limousin, K. Ashkan, J. Fitzgerald, A. L. Green, T. Z. Aziz, and P. Brown, "Adaptive deep brain stimulation in advanced Parkinson disease," *Ann. Neurol.*, vol. 4, no. 3, pp. 449–457, Sep. 2013.

[8] A. Mohammed, M. Zamani, R. Bayford, and A. Demosthenous, "Toward on-demand deep brain stimulation using Online Parkinson's

disease prediction driven by dynamic detection," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 25, no. 12, pp. 2441–2452, Dec. 2017.

[9] M. Ciletti, *Advanced Digital Design with Verilog HDL*, Second. New Jersey: Pearson Higher Education, 2010.

[10] Xilinx, "7 Series FPGA Data Sheet: Overview," *Product Specification*, 2017. [Online]. Available: https://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf. [Accessed: 20-Oct-2017].

[11] Digilent, "Nexys4 DDR FPGA Board Refrence Manual," 2016. [Online]. Available: https://reference.digilentinc.com/_media/reference/programmable-logic/nexys-4-ddr/nexys4ddr_rm.pdf. [Accessed: 20-Oct-2017].

[12] D. Sundararajan, *Discrete Wavelet Transform: A Signal Processing Approach*. New York, United States: John Wiley and Sons Ltd, 2015.

[13] D. Marković and R. Brodersen, *DSP Architecture Design Essentials*. New York, NY: Springer, 2012.

[14] K. Zhou, H. Qiming, and B. Guo, "Real-time kd-tree construction on graphics hardware," *ACM Trans. Graph.*, vol. 27, no. 5, 2008.

[15] A. Page, C. Sagedy, E. Smith, N. Attaran, T. Oates, and T. Mohsenin, "A Flexible Multichannel EEG Feature Extractor and Classifier for Seizure Detection," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 62, no. 2, pp. 109–113, Feb. 2015.

[16] M. Hasanlou and F. Samadzadegan, "Comparative Study of Intrinsic Dimensionality Estimation and Dimension Reduction Techniques on Hyperspectral Images Using K-NN Classifier," *IEEE Geosci. Remote Sens. Lett.*, vol. 9, no. 6, pp. 1046–1050, Nov. 2012.

[17] K. Q. Weinberger and L. K. Saul, "Distance Metric Learning for Large Margin Nearest Neighbor Classification," *J. Mach. Learn. Res.*, vol. 10, pp. 207–244, 2009.

[18] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. John Wiley & Sons, Inc., 2001.

[19] Nangate Inc, "NanGate FreePDK45 Open Cell Library," 2017. [Online]. Available: http://www.nangate.com/?page_id=2325. [Accessed: 25-Oct-2017].

[20] R. E. Elliott, A. Morsi, O. Tanweer, B. Grobelny, E. Geller, C. Carlson, O. Devinsky, and W. K. Doyle, "Efficacy of vagus nerve stimulation over time: Review of 65 consecutive patients with treatment-resistant epilepsy treated with VNS greater than 10years," *Epilepsy Behav.*, vol. 20, no. 3, pp. 478–483, Mar. 2011.

[21] S. Little and P. Brown, "What brain signals are suitable for feedback control of deep brain stimulation in Parkinson's disease?," *Ann. N. Y. Acad. Sci.*, vol. 1265, no. 1, pp. 9–24, 2012.

[22] J. F. Baizabal-Carvallo and J. Jankovic, "Movement disorders induced by deep brain stimulation.," *Parkinsonism Relat. Disord.*, Jan. 2016.

[23] M. Parastarfeizabadi, A. Z. Kouzani, M. Moffitt, K. Otto, D. Kipke, and C. McIntyre, "Advances in closed-loop deep brain stimulation devices," *J. Neuroeng. Rehabil.*, vol. 14, no. 1, p. 79, Dec. 2017.

[24] A. Priori, "Technology for deep brain stimulation at a gallop," *Mov. Disord.*, vol. 30, no. 9, pp. 1206–1212, Aug. 2015.

[25] S. Stanslaski, P. Afshar, P. Cong, J. Giftakis, P. Stypulkowski, D. Carlson, D. Linde, D. Ullestad, A.-T. Avestruz, and T. Denison, "Design and validation of a fully implantable, chronic, closed-loop neuromodulation device with concurrent sensing and stimulation.," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 20, no. 4, pp. 410–21, Jul. 2012.

[26] G. Meyfroidt, F. Güiza, J. Ramon, and M. Bruynooghe, "Machine learning techniques to examine large patient databases.," *Best Pract. Res. Clin. Anaesthesiol.*, vol. 23, no. 1, pp. 127–43, Mar. 2009.

[27] A. E. W. Johnson, M. M. Ghassemi, S. Nemati, K. E. Niehaus, D. Clifton, and G. D. Clifford, "Machine learning and decision support in critical care," *Proc. IEEE*, vol. 104, no. 2, pp. 444–466, Feb. 2016.

[28] H. Kassiri, S. Tonekaboni, M. T. Salam, N. Soltani, K. Abdelhalim, J. L. P. Velazquez, and R. Genov, "Closed-loop neurostimulators: A survey and a seizure-predicting design example for intractable epilepsy treatment," *IEEE Trans. Biomed. Circuits Syst.*, vol. 11, no. 5, pp. 1–15, Oct. 2017.

[29] W.-M. Chen, H. Chiueh, T.-J. Chen, C.-L. Ho, C. Jeng, M.-D. Ker, C.-Y. Lin, Y.-C. Huang, C.-W. Chou, T.-Y. Fan, M.-S. Cheng, Y.-L. Hsin, S.-F. Liang, Y.-L. Wang, F.-Z. Shaw, Y.-H. Huang, C.-H. Yang, and C.-Y. Wu, "A fully integrated 8-channel closed-loop neural-prosthetic CMOS SoC for real-time epileptic seizure control," *IEEE J. Solid-State Circuits*, vol. 49, no. 1, pp. 232–247, Jan. 2014.

[30] M. Sung, C. Marci, and A. Pentland, "Wearable feedback systems for rehabilitation," *J. Neuroeng. Rehabil.*, vol. 2, no. 1, p. 17, Jun. 2005.

[31] P. J. Soh, G. A. E. Vandenbosch, M. Mercuri, and D. M. M.-P. Schreurs, "Wearable wireless health monitoring: Current developments, challenges, and future trends," *IEEE Microw. Mag.*, vol. 16, no. 4, pp. 55–70, May 2015.

[32] A. Zhang, L. Wang, X. Ye, and X. Lin, "Light-weight and robust security-aware D2D-assist data transmission protocol for mobile-health systems," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 3, pp. 662–675, Mar. 2017.

[33] S. Iranmanesh and E. Rodriguez-Villegas, "A 950 nW analog-based data reduction chip for wearable EEG systems in Epilepsy," *IEEE J. Solid-State Circuits*, vol. 52, no. 9, pp. 2362–2373, Sep. 2017.

[34] M. A. Bin Altaf and J. Yoo, "A 1.83 J/classification, 8-channel, patient-specific epileptic seizure classification SoC using a non-linear support vector machine," *IEEE Trans. Biomed. Circuits Syst.*, vol. 10, no. 1, pp. 49–60, Feb. 2016.

[35] M. A. Bin Altaf, C. Zhang, and J. Yoo, "A 16-channel patient-specific seizure onset and termination detection SoC with impedance-adaptive transcranial electrical stimulator," *IEEE J. Solid-State Circuits*, vol. 50, no. 11, pp. 2728–2740, Nov. 2015.

[36] M. Shoaran, C. Pollo, K. Schindler, and A. Schmid, "A fully integrated IC with 0.85-uW/channel consumption for epileptic iEEG detection," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 62, no. 2, pp. 114–118, Feb. 2015.

[37] M. Shoaib, K. H. Lee, N. K. Jha, and N. Verma, "A 0.6–107 μW energy-scalable processor for directly analyzing compressively-sensed EEG," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 61, no. 4, pp. 1105–1118, Apr. 2014.

[38] J. Yoo, L. Yan, D. El-Damak, M. A. Bin Altaf, A. H. Shoeb, and A. P. Chandrakasan, "An 8-channel scalable EEG acquisition SoC with patient-specific seizure classification and recording processor," *IEEE J. Solid-State Circuits*, vol. 48, no. 1, pp. 214–228, Jan. 2013.

[39] T.-J. Chen, S.-C. Lee, C.-H. Yang, C.-F. Chiu, and H. Chiueh, "A 28.6 μW mixed-signal processor for epileptic seizure detection," in *Proc. Symp. VLSI Circuits (VLSIC)*, 2013, Kyoto, Japan, pp. C52–C53.