sensors

Article

Hardware/Software Co-Design of Fractal Features Based Fall Detection System

Ahsen Tahir [1,2,*], Gordon Morison [1], Dawn A. Skelton [3] and Ryan M. Gibson [1]

[1] School of Computing, Engineering and Built Environment, Glasgow Caledonian University, Glasgow G4 0BA, UK; Gordon.Morison@gcu.ac.uk (G.M.); Ryan.Gibson@gcu.ac.uk (R.M.G.)
[2] Department of Electrical Engineering, University of Engineering and Technology, Lahore, Punjab 54890, Pakistan
[3] School of Health and Life Sciences, Glasgow Caledonian University, Glasgow G4 0BA, UK; Dawn.Skelton@gcu.ac.uk
* Correspondence: ahsen.tahir@gcu.ac.uk

Received: 5 March 2020; Accepted: 16 April 2020; Published: 18 April 2020

Abstract: Falls are a leading cause of death in older adults and result in high levels of mortality, morbidity and immobility. Fall Detection Systems (FDS) are imperative for timely medical aid and have been known to reduce death rate by 80%. We propose a novel wearable sensor FDS which exploits fractal dynamics of fall accelerometer signals. Fractal dynamics can be used as an irregularity measure of signals and our work shows that it is a key discriminant for classification of falls from other activities of life. We design, implement and evaluate a hardware feature accelerator for computation of fractal features through multi-level wavelet transform on a reconfigurable embedded System on Chip, Zynq device for evaluating wearable accelerometer sensors. The proposed FDS utilises a hardware/software co-design approach with hardware accelerator for fractal features and software implementation of Linear Discriminant Analysis on an embedded ARM core for high accuracy and energy efficiency. The proposed system achieves 99.38% fall detection accuracy, 7.3× speed-up and 6.53× improvements in power consumption, compared to the software only execution with an overall performance per Watt advantage of 47.6×, while consuming low reconfigurable resources at 28.67%.

Keywords: fall detection; wearable sensors; classification; machine learning; fractal features; hardware software co-design; FPGA; reconfigurable design; embedded system on chip

1. Introduction

Falls are the highest cause of death from injury in individuals over the age of 65, resulting in high mortality, morbidity and immobility [1]. Falls produce a high cost for the National Health Service (NHS), with associated costs over £2 billion and 4 million bed days per year [1]. Fall Detection Systems (FDS) provide aid and support for individuals who live alone and may not be able to call for prompt medical aid due to injury or unconsciousness. Additionally, fall detection systems have been reported to obtain 26% and 80% improvements for hospitalization and death rates by providing immediate medical aid on fall event detection [2]. FDS detect and classify falls from other movements of a human body caused by Activities of Daily Life (ADL).

The movements of a human body are a result of complex non-linear interactions between feed-forward spinal circuitry and feed-back mechanisms from muscles, skin and various senses [3]. It is a non-linear dynamic system, which can be modelled as a chaotic system and analysed with chaos theory and fractal dynamics. While, the present work in non-linear dynamic systems understands the human body as a chaotic system and is limited to walking activity and gait analyses. There is a significant gap in understanding of the non-linear dynamics of falls through fractal analysis of accelerometer signals for falls and ADL.

Fractals are self-similar structures where the whole is similar to its parts. Self-similarity in structures or patterns can be approximate and limited to one or more parts. Self-similarity can also be statistical in nature. Many real world objects show statistical self-similarity where statistical properties are similar at different scales. Statistical self-similarity may manifest in time varying signals. However, sensor signals do not need to be truly fractal in nature. The fractal dimension has a positive correlation with irregularity of a signal, according to Mandelbrot [4] and can be used as a measure of signal irregularity. Our work first determines if such an irregularity measure of the signal is a discriminant feature for classification of falls from ADL. While many fractal analysis methods have been proposed in literature, our work utilises Autoregressive Fractionally Integrated Moving Average (ARFIMA) for fractal analysis, since the technique has the advantage of solving the problem of biased overestimation and higher errors of fractal parameters for complex processes [5,6], such as human movements and activities. ARFIMA works with stationary signals and non-stationary signals can be analysed with ARFIMA by first conversion to stationary signals [7]. Once the signal is converted to a stationary signal, ARFIMA is applied and fractal parameters are calculated. The fractal parameters are then determined for the original non-stationary signal through conversion from the stationary signal parameters [7]. Therefore, stationarity testing is imperative for such an approach. For stationarity testing fall and ADL signals are analysed with Augmented Dickey-Fuller (ADF) and Kwiatkowski-Phillips-Schmidt-Shin (KPSS) tests in Section 4.2 for a rigorous treatment to understand the stationarity characteristics and provide empirical evidence of non-stationarity. For non-stationary signals it is important to find the order of difference for conversion to a stationary signal. ARFIMA modelling is performed in Section 4.3.1 using a public fall dataset by Kwolek et al. [8] discussed in Section 3. ARFIMA modelling in Section 4.3.1 shows that fractal parameters are useful discriminant features to classify falls, to the best of our knowledge, fractal parameters have not been used to classify falls from ADL. While, ARFIMA provides robust analysis, it requires development of ARFIMA models and careful inspection of goodness of fit functions to determine the fractal dynamics, which is not appropriate for real-time classification events. The relationship of Discrete Wavelet Transform (DWT) with fractal computations [9,10] is leveraged for real-time implementation on an embedded wearable device for fall detection.

Hardware reconfigurable design of a fall detection system can satisfy the design constraints of high performance per Watt to enable real-time implementation with computationally intensive algorithms for higher accuracy requirements. Embedded multicore FPGAs are a suitable platform for hardware implementations and acceleration of intelligent systems/machine learning models on embedded devices and provide good performance to power ratio [11,12]. A real-time sustainable operation requires low power consumption and high throughput with high performance per Watt for a multi-level wavelet transform and fractal features extraction process for FDS. The high throughput and low latency achieved with reconfigurable design can provide system scalability for multiple multichannel sensors at a low power budget. One of the major contributions of our work is a reconfigurable accelerator design which leverages multi-level DWT for real-time fractal computations performed directly on non-stationary signals, since ARFIMA method is not feasible for an embedded low latency implementation. We utilise a hardware/software co-design of a fall detection system to satisfy the design constraints of high performance per Watt to enable real-time implementation of computationally intensive parts of the algorithm. The hardware accelerator for multi-level DWT and fractal features leverages various design innovations and optimizations. Pipelined arithmetic trees are utilised for convolution operations and variance computations. Memory system optimizations are performed to provide required throughput to the arithmetic trees through cyclic two-port memory blocks. DWT convolution and subsequent downsampling operations are optimised into a single operation by skipping alternative computations and storing filter coefficients in flipped form. The number of clever design optimizations and their results are discussed in Sections 7.1 and 8, respectively. The fractal dimensions along with the DWT low pass coefficients obtained as a byproduct of fractal computations from the hardware accelerator are passed on as features to the embedded ARM core for machine learning classification. A Linear Discriminant Analysis (LDA) classifier is then applied to the feature set for classification of falls on an embedded ARM core in the SoC device. We show that with fractal features and DWT coefficients, classification of falls from ADL provide high accuracy of 99.38%, a high throughput, low power consumption and high performance per Watt for a hardware/software co-designed system. The main contributions of our work are as follows:

- Fractal analyses undertaken to explore the irregularity of accelerometers for stationary and non-stationary fall/ADL signals.
- Fractal features utilised as irregularity metric of the signal for fall detection and classification for the first time, to the best of our knowledge.
- Energy efficient hardware accelerator for high throughput and maximal re-use of computation blocks for the feature extraction process.
- Multi-level DWT and fractal features clever design innovations and optimizations, including pipelined arithmetic trees for fractal computations, cyclic two-port memory optimizations, convolution and subsequent downsampling optimization through a single operation etc.
- Hardware/software co-design of a portable FDS system for sustainable operation and real-time classification, evaluating wearable accelerometer sensor.
- Higher performance and accuracy of 99.38% than existing hybrid vision and accelerometer fall detection systems.
- Low power and latency optimised design with high performance per Watt of 46.7$\times$.

2. Related Work

Current fall detection systems have a significant focus on using machine learning algorithms and can be classified into sensor-based [13–17] and vision-based [18–23] systems. The data obtained from sensors or camera is processed to extract features for classification of human movements as falls. Most sensor-based FDS have employed acceleration data [8,16,24,25]. Gibson et al. [25] evaluated accelerometer data with wavelet transforms and principal component analysis to detect falls and utilise compressive sensing based techniques to reduce transmission information. They further utilise multiple classifiers with a majority voting system to improve performance over a single classifier for robust classification [24]. Sukor et al. [16] used accelerometer data from the MobiFall dataset [26] for signal processing and features selection. A number of time-domain and frequency domain features were used including Spectral Density and Spectral Energy. Kwolek et al. [8] applied support vector machines with accelerometer signals and image evaluation for fall detection. Hsieh et al. [13] presented the hierarchical fall event algorithm that uses a dual threshold and machine learning based approach for detection of falls from triaxial accelerometer with sensitivity, specificity and accuracy values above 98%. Zhong et al. [17] presented a real-time algorithm system based on the thresholds of velocity and displacement to classify falls. A second order filter was applied to the signal to minimize the impact of drift on vertical velocity.

Current work on chaotic and fractal analysis of human movements is limited to gait analysis and human walk. Human gait models have been determined to possess elements of chaotic systems in [27]. A local dynamic stability analysis of walking activity and human gait has determined a positive Lyapunov Exponent (LE) [28–31]. A positive LE is a signature characteristic of chaotic systems and a measure of sensitivity to small perturbations. Recently, [28,32] have associated local dynamic stability with the risk of falls. Morbidoni et al. [33] utilise electromyography signals for deep learning classification of stance, swing phases during natural walking. Pairot et al. [34] determined the validity of commercially available, wearable sensors by monitoring gait during running and concluded that only few metrics measured by commercially available sensors are valid. Park et al. [35] developed a real-time healthcare monitoring system for monitoring gait and vital signs, and utilised machine learning classification for health conditions, such as disordered gait and onset of stroke. Margiotta et al. [36] utilised a wearable device wireless system to analyse time gait variability, for early diagnosis of health conditions. Nguyen et al. [37] captured gait characteristics with multiple wearable body sensors and performed gait classification to determine subject groups with abnormalities. Schneider et al. [38] proposed a gait analysis system to determine gait parameters and speed. The authors utilised a camera and an accelerometer sensor for discriminating various gait speeds through frequency domain gait features. Coviello et al. [39] utilised embedded devices with inertial sensors to propose a large multi-sensor platform for measurement of activities through reduced single node complexity and guaranteeing time synchronization for acquired samples. Sahoo et al. [40] proposed an early detection technique for inertial sensor based system to detect gait events early and reduce the effect of delay in powered prosthetic and assistive devices. Apart from stability analysis, fractal dynamics of walk and human gait has also been analysed in [41–43]. Fractal dimensions have been used in biomedical systems for detection of anomalies [44,45]. Koutsiana et al. [44] evaluated fractal dimensions on wavelet transformed data for detection of fetal heart sounds, while Zhang et al. [45] used fractal dimensions to detect brain anomalies. current work has not investigated fractal features for training machine learning algorithms for fall and activities, let alone an embedded reconfigurable device implementation.

Recent published work with FPGAs and SoCs with programmable logic resources such as the Xilinx Zynq system either do not provide power consumption analysis or suffer from lower accuracies with accelerometer sensors for fall detection. Vision-based approaches have limitations, while higher accuracy is also achieved at the cost of higher power consumption. Senouci et al. [46] utilised spatio-temporal information from camera images including wavelet transform, object bounding box height, width, aspect ratio and motion variation with hardware acceleration on a Zynq device to classify falls in real-time with Support Vector Machine (SVM) and Adaboost algorithms. However, their implementation did not give any power consumption values. Ali et al. [47] implemented a sensor-based FDS on a Zynq System on Chip (SoC) device which applies DWT and PCA for feature extraction and a binary decision tree for classification. The system suffered from lower accuracy and no power consumption analysis has been performed. Ong et al. in [48,49] presented an FPGA-based architecture for visual fall detection with a CMOS camera. They achieved a frame rate of 60 fps for VGA resolutions of 640 $\times$ 480 with a pixel processing pipeline which implements feature extraction. Furthermore, the design is optimised in [49] for power giving a reduction of up to 33% in power consumption from their initial design. The technique not only suffered from the limitations of a vision-based systems, but resulted in a high power consumption of 5.2 W which is not feasible for an embedded wearable device. Abdelhedi et al. [50] proposed an FPGA based solution for fall detection on a Zybo board with a single accelerometer. Their work used threshold method with the sum-vector of three accelerometer axes and the body tilt angle for fall detection. All the processing is performed in software on the ARM core, while only the detection decision is sent to the programmable logic for fall detection indication (blinking LEDs). The work was further expanded in [51] by implementing a hardware core for the above mentioned operations in programmable logic on a Zybo FPGA board. To the best of our knowledge, this is the first implementation of a fractal feature accelerator using multi-level DWT for classification of falls from ADL.

3. Fall Detection System Overview

The proposed fall detection system concept is illustrated in Figure 1 and is designed to evaluate wearable accelerometer sensors with efficient machine learning algorithms implemented on an embedded reconfigurable Zynq device.

While the proposed Zynq SoC design is potentially a prime candidate for a wearable/body mounted configuration, in this work, the proposed system is designed as a Zynq device base station for portability, sustainability and scalability, and performs ADL/fall classification and detection derived from dataset signals.The fall classification and detection decision is transmitted from the Zynq SoC device to the wireless router, from where a medical aid centre is notified for an immediate medical response.

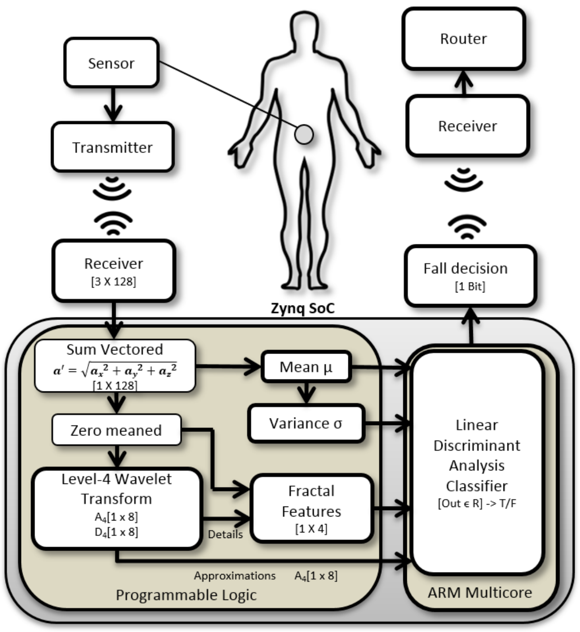


Figure 1. Wearable fall detection system overview.

The Zynq SoC device is a development board and consists of dual-core ARM processor and a Programmable Logic (PL) core. The Zynq device has 6.3 in $\times$ 6.3 in dimensions with a SoC measuring 19 mm $\times$ 19 mm. The Zynq board includes 256 KB on chip RAM, 512 MB board RAM, Serial Peripheral Interface (SPI) and Inter-Integrated Circuit (I2C) interfaces and 12-bit Analog to Digital Converters (ADC). The system has on chip voltage and temperature sensors. External accelerometer sensors can be connected wirelessly to the board or directly through the ADC and communicate directly with the PL. The device works with 12 V battery operated or external power converter. The movement signals can be obtained from any wearable accelerometer sensor in units of G-force (g), with similar specifications to the dataset by Kwolek et al. [8], against which the design is validated. The proposed Zynq SoC design is compatible with accelerometer sensors, such as x-IMU [52] and ADXL345-EP [53] which have a programmable sampling rate of 32 Hz and a full scale range of -8 to +8 g at 12-bit resolution. Window segments of 128 samples from each of the tri axis acceleration signals $\mathbf{a_x}$, $\mathbf{a_y}$ and $\mathbf{a_z}$ along three axes of motion x, y and z are processed to obtain the sum vectored signal $\mathbf{a}' = \sqrt{\mathbf{a}_\mathbf{x}^2 + \mathbf{a_y}^2 + \mathbf{a_z}^2}$. The process captures the signal variations along the three axis of motion into a single signal and reduces the processing time. The mean $\mu_{a'}$ of the sum vectored signal $\mathbf{a}' = \{a'(n)\}$ where $n = \{1, \ldots, 128\}$, is computed and the sum vectored signal is zero meaned, $\mathbf{a} = \mathbf{a}' - \mu_{a'}$. The zero mean signal $\mathbf{a} = \{a(n)\}$ for $n = \{1, \ldots, 128\}$ is then wavelet transformed with a Daubechies-4 DWT function to obtain the DWT approximation $\mathcal{A}_1$ and detail coefficients $\mathcal{D}_1$ at level 1. The process is repeated 4 times to achieve a 4-level wavelet transform with a DWT output signal of size [1 $\times$ 8] for each of the approximation and detail coefficients. The variance of the DWT detail coefficients $\mathcal{D}_i$ and the variance $\sigma_a^2 = \sum_{n=1}^{N}(a(n) - \mu_a)^2/(N-1)$ of the zero meaned, sum vectored accelerometer signal $\mathbf{a}$ with mean $\mu_a = 0$ and $N = 128$ are used for fractal analysis, since the variance of detail coefficients can be utilised to determine fractal dimensions according to Equations (25)–(27). The fractal dimensions along with the mean $\mu_{a'}$ and variance $\sigma_{a'}^2$ of the sum vectored accelerometer signal $\mathbf{a}' = \{a'(n)\}$ for $n = \{1, \ldots, 128\}$ and DWT approximation coefficients $\mathcal{A}_4$ at level 4 are used as a feature set for classification and detection of falls. The classification is carried out on the feature set with machine learning classifiers on the ARM core of a reconfigurable embedded system, as shown in Figure 1. Multiple DWT level analysis is a computationally intensive process and requires hardware implementation to investigate resource usage with associated DWT levels and fall detection accuracy. The fractal dimensions are calculated from the wavelet coefficients on the reconfigurable logic hardware accelerator.

Dataset: The proposed DWT based fractal feature and machine learning algorithms for FDS were trained, tested and validated with Matlab and publicly available fall accelerometer data by Kwolek et al. [8]. The data consists of falls and various activities of daily life including walking, kneeling, picking up objects, standing up, sitting down and lying. The data is obtained from an Inertial Measurement Unit with 12 bit three axis accelerometer and 16 bit gyroscope. The device was worn with the pelvis by 5 individuals who performed different kind of falls, including backward, forward and lateral falls. The data obtained from sensors is transmitted wirelessly from the device through Bluetooth. The sampling rate of the accelerometer is 32 Hz (overall sampling rate is 256 Hz for multichannel sensors). In our work, we utilise the accelerometer data for testing proposed features and algorithm. The accelerometer sensor values vary from -8 to +8 g. Three axes of accelerometer values are shown in Figure 2 with 128 sample segments for fall activity. Samples from each of the activities were used for training and validation of the proposed architecture. The input accelerometer data and the target outputs are stored as C arrays in the Xilinx Software Development Kit (SDK). The arrays are loaded into the Zynq memory and the accelerometer values are read in a loop. The data values are used as inputs into the hardware accelerator and the output features obtained from the hardware accelerator are used for LDA classification algorithm executed on the ARM core. The classification results obtained are then compared with the target outputs in the array and classification accuracy is calculated.

While the classification performance values have been computed for the given dataset. The fractal dimension, according to Mandelbrot [4] has a positive correlation with irregularity of a signal, and can be used as a measure of signal irregularity. The different irregularity characteristics of fall signals is a generic observation, which distinguishes all fall signals composed of a spike with ADL. The ADL signals are mostly composed of irregular episodic variations and hence can be distinguished from falls based on their irregularity characteristics.

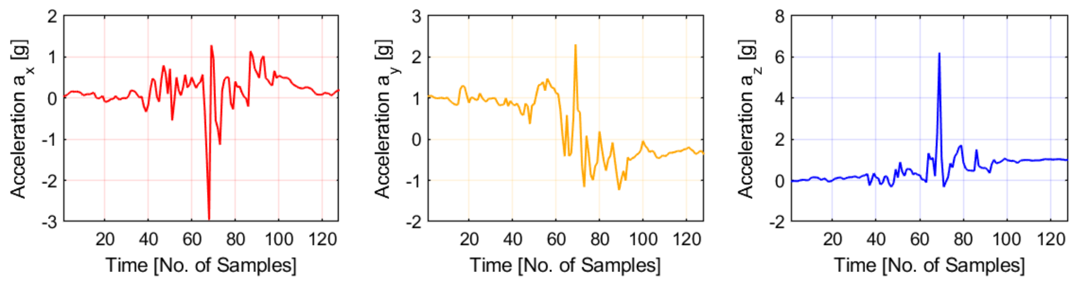


Figure 2. Three axes accelerometer signals $\mathbf{a_x}$, $\mathbf{a_y}$ and $\mathbf{a_z}$ for falls.

4. Signal Processing and Fractal Analysis

4.1. Discrete Wavelet Transform

The Discrete Wavelet Transform is a projection of the sum vectored and zero mean accelerometer signal $\mathbf{a} = \{a(n)\}$ where $n = \{1, \ldots, N\}$ and $N = 128$ samples, on a family of basis functions $\phi_{i,k}(n)$ and $\psi_{i,k}(n)$. The family of basis functions is obtained from the translations and dilations of the scaling function $\phi(n)$ and mother wavelet $\psi(n)$, which are defined as:

$$\phi_{i,k}(n) = 2^{-\frac{i}{2}}\phi\left(2^{-i}n - k\right) \tag{1}$$

$$\psi_{i,k}(n) = 2^{-\frac{i}{2}}\psi\left(2^{-i}n - k\right) \tag{2}$$

where k represents discrete translations and 2^i are dyadic dilations. The DWT on signal $a(n)$ can be given as:

$$\mathcal{A}_i(k) = \sum_n a(n)\phi_{i,k}(n) \tag{3}$$

$$\mathcal{D}_i(k) = \sum_n a(n)\psi_{i,k}(n) \tag{4}$$

where $\mathcal{A}_i$ and $\mathcal{D}_i$ are wavelet approximation and detail coefficients for the sum vectored and zero mean accelerometer signal $a(n)$, respectively at level i. Index k of wavelet coefficients corresponds to the shift index of the scaling and mother wavelet functions. The approximations also known as low pass wavelet coefficients $\mathcal{A}_i$ are used as an input signal in Equations (3) and (4) multiple times to produce 4-level DWT approximations $\mathcal{A}_4$ and details $\mathcal{D}_4$. The final approximations $\mathcal{A}_4$ and details $\mathcal{D}_4$ are a vector of size [1 $\times$ 8] each, as shown in FDS overview in Figure 1. The 4-level DWT approximations $\mathcal{A}_4$ are directly used as features for classification in the FDS system, while the $\mathcal{D}_4$ details are used to compute fractal dimension features for the signal as illustrated in Figure 1. The use of fractal features in the fall classification model is based on a fractal analysis of falls and activities, which show that fractal dimension can be used as an irregularity metric of the signal to discriminate between falls and activities. Sections 4.2 and 4.3 discuss stationarity tests and the fractal analysis of falls and activities, respectively.

4.2. Stationarity Tests

While there are many fractal analysis methods, such as Detrending Fluctuation Analysis (DFA) and Power Spectral Density (PSD), the ARFIMA method has been demonstrated to be superior to other methods for complex processes [6]. AFRIMA, however requires a process to be stationary or converted to a stationary process for analysis. Therefore stationarity testing of falls and ADLs signals or conversion to a stationary process in case of non-stationarity is imperative for computation of fractal parameters. The stationarity and non-stationarity test are discussed below, and a fractal analysis with ARFIMA is presented in next section. The ADF and KPSS tests are used for stationarity testing. ADF assumes the signal is non-stationary and tests for the presence of unit root. The test for unit root [54] determines if the process is non-stationary with stationary increments, i.e., first order difference stationary. The sum vectored, zero mean signal $\mathbf{a} = \{a(n)\}$ can be viewed as a unit root process with stationary increments in an ADF test. While KPSS test assumes that the signal is level stationary and tests whether the assumption is correct or not, which results in a double validation strategy.

4.2.1. Augmented Dickey-Fuller Test

The Augmented Dickey-Fuller (ADF) test [54] determines the stationarity of the process, which represents the human body movements responsible for signal variations. Different types of ADF tests are available. The form evaluated here is the ADF constant stationary with no trend, since signal values in our case do not exhibit long term time trends. However, the non-stationarity of signals against trend stationarity are also confirmed with the KPSS test in the next Section 4.2.2 for rigorous treatment. ADF test provides a rejection decision for a unit root process hypothesis H_0 of the form given in (5) against an alternative hypothesis H_1 in (6):

$$H_0: \; a(n) = a(n-1) + \sigma_1 \Delta a(n-1) + \cdots + e(n) \tag{5}$$

$$H_1: \; a(n) = \gamma a(n-1) + \sigma_1 \Delta a(n-1) + \cdots + e(n) \tag{6}$$

where,

- Δ is the difference operator with $\Delta a(n) = a(n) - a(n-1)$.
- $e(n)$ is an innovation process.
- γ is an autoregressive coefficient with value < 1.

The test gives a value of 0 or 1. A value of 0 means failure to reject the null hypothesis, while 1 means rejection of the null hypothesis. A p-value of less than 0.05 (5%) is used for the rejection decision and represents 95% confidence level for accepting the alternative hypothesis of stationarity. The hypothesis are as follows:

Null Hypothesis H_0: $\gamma = 1$ indicated by a test output of 0 means failure to reject the null hypothesis of a unit root process. The accelerometer signal generating process is considered to be non-stationary.

Alternative Hypothesis H_1: $\gamma < 1$ indicated by a test output of 1 means rejection of a unit root process in favor of alternative model. The accelerometer signal generating process is considered to be stationary. The Shwert rule for determining maximum lags for the ADF test [55] is given as:

$$p_{max,ADF} = \left[12 \times \left(\frac{N}{100}\right)^{1/4}\right] \tag{7}$$

where N is the total number of samples. The tests were performed with lag values according to two criteria. Firstly, the ADF test was performed with the maximum lag value p_{max}. Ng et al. [56] suggested a lag length selection procedure based on first selecting the maximum lag value p_{max} for p. If the t-statistic representing the significance is greater than 1.6 then p_{max} is used as the final lag value for the test, otherwise the lag was reduced by one and the process repeated. Secondly, the lag values were obtained from Akaike Information Criterion (AIC) [57] and used for ADF tests. Different criteria may provide different lag values for the test, since there is no single criteria which may apply to all cases, we tested stationarity against both lag selection criterias, which confirmed same results for the stationarity/non-stationarity tests.

4.2.2. Kwiatkowski-Phillips-Schmidt-Shin Test

Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test assumes a signal is trend stationary and tests for non-stationarity. The null hypothesis in KPSS test that the accelerometer signal generating process is stationary is opposite to the ADF test. The null hypothesis H_0 is given in Equation (8) representing a stationary process. While, the alternative hypothesis of unit root H_1 is given in Equation (10).

$$H_0: \; a(n) = c(n) + \lambda n + v_1(n), \text{ where} \tag{8}$$

$$c(n) = c(n-1) + v_2(n) \tag{9}$$

$$H_1: \; a(n) = a(n-1) + d_c + \lambda n + e(n) \tag{10}$$

where,

- λn represents a deterministic trend with coefficient λ and the number of samples n.
- d_c represents drift constant.
- $c(0)$ for $n = 0$ is fixed and represents an intercept. The subsequent values are calculated from (9).
- $e(n)$ is an innovation process.
- $v_1(n)$ represents a stationary process.
- $v_2(n)$ is a distributed process which is identically distributed and independent with 0 mean.

The test gives a value of 0 or 1. A value of 0 means failure to reject the null hypothesis, while 1 means its rejection. A p-value of less than 0.05 (5%) is used for the rejection decision and represents 95% confidence level for accepting the alternative hypothesis of non-stationarity. The hypothesis are as follows:

Null Hypothesis H_0: indicated by a test output of 0 means failure to reject the null hypothesis of a stationary process. The signal generating process is therefore considered to be stationary.

Alternative Hypothesis H_1: indicated by a test output of 1 means rejection of the null hypothesis in favor of the alternative model. The signal generating process is considered to be non-stationary. The rule for determining maximum lags is given by Kwiatkowski et al. [58] as:

$$p_{max,KPSS} = \sqrt{N} \tag{11}$$

The tests were performed with fewer lags and the sensitivity was measured by adding more lags.

4.2.3. Stationarity Results

The results of both the ADF and KPSS tests show that both the fall and ADL signals are non-stationary, which is indicated by positive results for the ADF tests and inability to reject the null hypothesis. While, the KPSS tests were negative and rejected the null hypothesis with a high confidence value of more than 95%, corresponding to a p-value of less than 0.05. Furthermore, applying first order difference filter results in stationary signals, as supported by the results of ADF test which rejects the null hypothesis with confidence of greater than 95% and a p-value of less than 0.05. For ARFIMA based fractal analysis in Section 4.3.1, which requires conversion to a stationary signal, the first difference filter operation is considered sufficient for conversion to stationary signal and is used for computing fractional difference parameter d, related to the Hurst exponent and fractal dimensions.

4.3. Fractal Analysis: ARFIMA

The fractal parameter analysis ARFIMA is based on the modelling for better accuracy of results. In [59], Box et al. presented a family of Autoregressive Integrated Moving Average (ARIMA) models to introduce short-term relationships in time varying processes. ARFIMA models are a generalization of ARIMA models based on fractional calculus. ARFIMA models can be used to find fractal dimensions and provide a fractional value of the differencing parameter d, which is directly related to the Hurst exponent H. The fractional value of d is bounded by $[-0.5, 0.5]$ and applies only to a stationary process. ARFIMA can be applied to a non-stationary signal $a(n)$ by initially converting to a stationary process through the difference operator, $s(n) = a(n) - a(n-1)$. In contrast, the fractional parameter $\tilde{d}$ for the non-stationary process $a(n)$ can then be obtained by adding 1 to the fractional parameter d for the stationary process $s(n)$ [7]. ARIMA models have three components: Autoregressive AR(r), Moving Average MA(q) and the Integrated part I(d). The autoregressive AR(r) term describes the value $s(n)$ by a weighted-sum of previous r values and a random variable $\epsilon(n)$:

$$s(n) = \alpha + \sum_{l=1}^{r} \zeta_l s(n-l) + \epsilon(n) \tag{12}$$

where $\zeta_1, \zeta_2, \cdots, \zeta_r$ are autoregressive coefficients and α is a constant. The autoregressive terms decay over time for stationary processes. The MA(q) term describes the current value $s(n)$ by a weighted-sum of previous q random perturbations $\epsilon(n-1), \cdots, \epsilon(n-l)$.

$$s(n) = \mu_s + \sum_{l=1}^{q} \theta_l \epsilon(n-l) + \epsilon(n) \tag{13}$$

where $\theta_1, \theta_2, \cdots, \theta_p$ are moving average coefficients and μ_s is the mean of $s(n)$. The integrated I(d) part represents the order of difference d required for the ARIMA Model. It specifies whether the observed values are directly modelled; $d = 0$ or their differences $d = 1, 2, \cdots$ are modelled. Given a lag operator $\mathbb{L}s(n) = s(n-1)$ for all $n > 1$, where $\mathbb{L}^l s(n) = s(n-l)$.

$$\begin{aligned}\Delta s(n) &= s(n) - s(n-1)\\ &= (1-\mathbb{L})s(n) \qquad (14)\\ \Delta^d s(n) &= (1-\mathbb{L})^d s(n) \qquad (15)\end{aligned}$$

The ARIMA model can be described as:

$$(1 - \sum_{l=1}^{r} \zeta_l \mathbb{L}^l)(1-\mathbb{L})^d(s(n) - \mu_s) = (1 + \sum_{l=1}^{q} \theta_l \mathbb{L}^l)\epsilon(n) \qquad (16)$$

In ARIMA models d is an integer, while in fractional ARFIMA(r, d, q) models capture fractal dynamics with real values for d. An ARFIMA(r, d, q) model for the accelerometer difference signal $s(n)$ can be described by Equation (16), where the fractional difference operator $\Delta^d = (1-\mathbb{L})^d$ can be represented by a binomial expansion for real number d with Gamma function as:

$$\begin{aligned}(1-\mathbb{L})^d &= \sum_{l=0}^{\infty} \binom{d}{l} (-\mathbb{L})^l\\ &= \sum_{l=0}^{\infty} \frac{\Gamma(d+1)}{\Gamma(l+1)\Gamma(d+1-l)}(-\mathbb{L})^l \qquad (17)\end{aligned}$$

with general form of ARFIMA(r, d, q) process defined as:

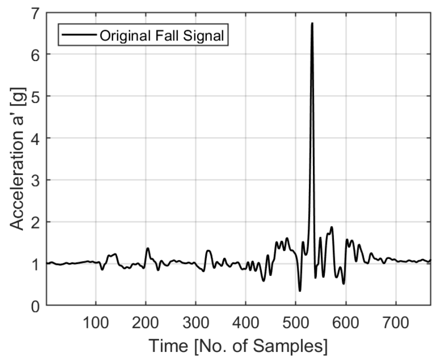
$$\Theta(\mathbb{L})(1-\mathbb{L})^d s(n) = \Theta(\mathbb{L})\epsilon(n) \qquad (18)$$

where d is between -0.5 to 0.5 and represents the self-similarity of the ARFIMA process. For non-stationary process $a(n)$, the fractional difference value $\tilde{d}$, signal spectral exponent β, Hurst exponent H and fractal dimension fd can then be calculated from d as:

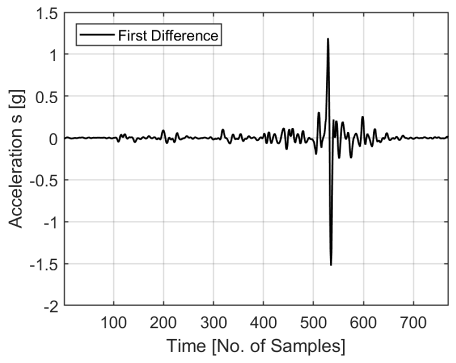
$$\begin{aligned}\tilde{d} &= d + 1 \qquad (19)\\ \beta &= 2\tilde{d} \qquad (20)\\ H &= \frac{\beta - 1}{2} \qquad (21)\\ fd &= 2 - H \qquad (22)\end{aligned}$$

4.3.1. Fractal Dimensions of Falls with ARFIMA

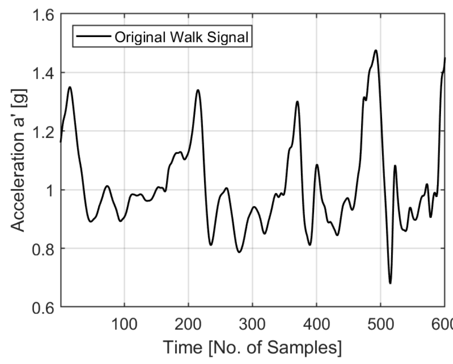
Estimation of fractal dimensions was performed after application of the first order difference filter to the activity signals to convert the non-stationary process into a stationary process for ARFIMA modelling. We fitted 20 ARFIMA (r, d, q) models with $r \in [0, 4]$ and $q \in [0, 4]$ to the output of the first difference filter and retained fractional difference d with the lowest log likelihood and AIC values for accuracy and parsimony along with the model fitness values. For illustration purposes, the original signals for fall, walking and picking up objects along with their first difference signals are shown in Figure 3. While, the selected ARFIMA models for falls, walking and picking up objects with the lowest log likelihood and AIC values are illustrated in Table 1. The fractional difference d values of the selected ARFIMA models with best goodness of fit statistics are then utilised for computing fractional integration coefficient $\tilde{d}$ for the original non-stationary process and the corresponding fractal dimension using Equations (19)–(22). The mean values for fractional integration coefficient $\tilde{d}$, Hurst parameter H and fractal dimensions fd are given in Table 2 for all the falls and ADLs. The value of Hurst exponent $H = 0.49$ and fractal dimension 1.01 for falls demonstrates a clear distinction in irregularity characteristics of falls compared to ADLs. ADLs show higher irregularity w.r.t. the dimension parameter. It is clear from analysis that the value of fractal dimension for falls at 1.01 can act as a good discriminant for fall classification, since all other activities have relatively higher values. The fractal dimension was not calculated for "lying" in Table 2 because of an unreliably high Standard Deviation (SD) greater then the mean value, which results in a fractal dimension of 2.



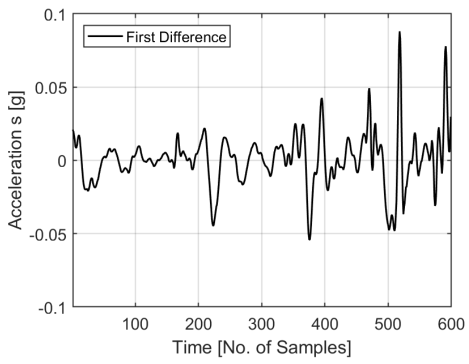
(**a**) Accelerometer fall signal



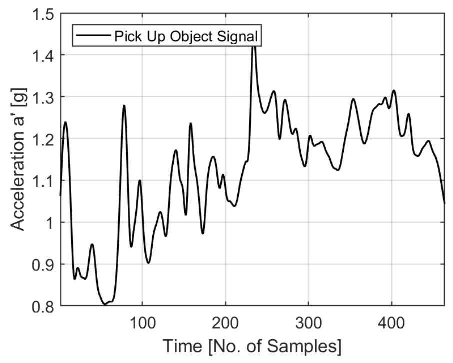
(**b**) First difference fall signal



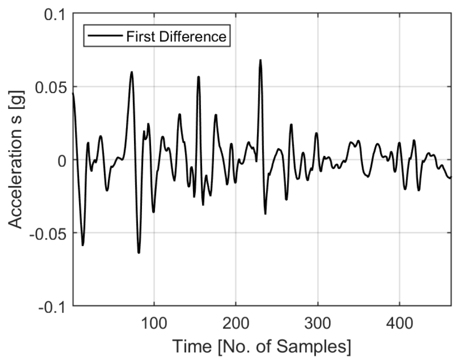
(**c**) Accelerometer walk signal



(**d**) First difference walk signal



(**e**) Accelerometer picking up signal



(**f**) First difference picking up object

Figure 3. Original and first difference signals.

Table 1. ARFIMA model for falls, walking and picking up objects.

Activity	ARFIMA	AR(1)	MA(1)/MA(4)	Likelihood	AIC	d	$\tilde{d}$	H
Falling	(1,d,1)	0.64	0.46	1144.5	−3.94	0.49	1.49	0.99
Walking	(1,d,4)	0.86	2.1, 2.3, 1.9, 0.8	2785	−9.27	−0.48	0.52	0.02
Picking up	(1,d,4)	0.9	1.5, 1.6, 1.2, 0.4	2353	−10.13	−0.32	0.68	0.18

Table 2. Hurst exponent from ARFIMA analysis.

Activity	d		$\tilde{d} = d + 1$	$H = d + 0.5$	$fd = 2 - H$
	Mean	**SD**	**Mean**	**Mean**	**Mean**
Falls	1.490	0.003	1.49	0.99	1.01
Walking	0.518	0.002	0.518	0.018	1.982
Kneeling down	0.554	0.258	0.554	0.054	1.946
Sitting down	0.999	0.000	0.999	0.499	1.501
Standing up	0.99	0.007	0.990	0.490	1.51
Picking up objects	0.699	0.091	0.699	0.199	1.801
Lying	0.225	0.347	-	-	-

While, ARFIMA is more rigorous and robust for computation of fractal parameters for complex processes [5,6]. The use of stationary signals, higher computations and selection of ARFIMA models are not feasible for an embedded approach, where signals are non-stationary or stationarity characteristics of the process are not known. DWT based computation of fractal dimensions does not suffer from the limitations of ARFIMA and can be directly applied to non-stationary signals for real-time computation of fractal dimensions.

5. Proposed Fall Classification with Fractal Features

This section discusses the proposed fall classification algorithm based on fractal features for an embedded wearable device. The previous Section 4.3.1 utilises ARFIMA analysis to establish that fractal dimension for falls is distinct with a mean value of 1.01, as compared to other activities which have a mean fractal dimension of at least 1.5, as illustrated in Table 2. While, ARFIMA analysis is robust and less prone to error for complex processes [5,6]. ARFIMA is computationally expensive and requires computing various ARFIMA models for a number of different parameters before a selection of the best-fit could be made based on goodness of fit statistic functions. Hence fractal dimension computations based on ARFIMA analysis are not suitable for an embedded system to compute and utilise fractal features for detection of falls in a low latency wearable device with real time constraints. We show that such a system can utilise DWT based method for computation of fractal dimensions for non-stationary falls and activities signals in an embedded system with low latency requirements. The proposed reconfigurable embedded FDS therefore utilises DWT for computation of fractal features. The mathematical basis for the DWT based method are given in next Section 5.1 and the classification algorithm used in the FDS is discussed in Sections 5.2. While, classification accuracy improvements with fractal features are presented in Section 5.3.

5.1. DWT Based Fractal Features

The fractal dimensions can be computed from the Hurst exponent H, which can be calculated for non-stationary accelerometer signals by use of DWT detail coefficients [9,10]. In the DWT method, high pass wavelet decomposition filter has a frequency band of $f_s/2^{i+1} < \omega < f_s/2^i$ for the ith level wavelet detail coefficients $\mathcal{D}_i$ for a sampling frequency of f_s. The variance of the detail coefficients can therefore be used as the power spectrum density $S(\omega)$ of the original signal. The power spectrum density of the signal can be given in terms of variance σ_a^2 of the non-stationary, sum vectored and zero mean signal $a(n)$ and the power spectrum exponent β, as:

$$S(\omega) = \frac{\sigma_a^2}{\|\omega\|^\beta} \tag{23}$$

Replacing the above with variance of the detail coefficients results in the following equation:

$$var(\mathcal{D}_i) = \frac{\sigma_a^2}{(2^i)^\beta} \tag{24}$$

The power spectrum exponent β can be calculated from the above equation as:

$$\beta = \frac{log_2\left[\sigma_a^2/var(\mathcal{D}_i)\right]}{log_2[2^i]} \tag{25}$$

The Hurst exponent H and fractal dimension fd can be calculated as:

$$H = \frac{\beta - 1}{2} \tag{26}$$

$$fd = 2 - H \tag{27}$$

5.2. Proposed Algorithm

The proposed algorithm performs fall detection and classification from ADLs based on features obtained from multi-wavelet transform, mean of the signal, variance of the signal and the subsequent computation of fractal dimensions at each level of the multi-level wavelet transform. The feature extraction part is performed in hardware, where the classification is done on the ARM core in software. The steps of the proposed algorithm are given below and the flow chart is illustrated in Figure 4 with hardware and software implemented components. The following steps are performed on 128 sample windows of accelerometer readings with 50% overlapping between the windows for each of the three accelerometer axis.

1. Compute the sum vectored signal $\mathbf{a}'$ using, $\mathbf{a}' = \sqrt{\mathbf{a_x}^2 + \mathbf{a_y}^2 + \mathbf{a_z}^2}$ of the tri axis accelerometer signals, $\mathbf{a_x}$, $\mathbf{a_y}$ and $\mathbf{a_z}$.
2. Compute the mean $\mu_{a'}$ of the sum vectored signal $\mathbf{a}'$ and convert to a zero mean signal, $\mathbf{a} = \mathbf{a}' - \mu_{a'}$.
3. Compute variance $\sigma_{a'}^2$ of the sum vectored signal $\mathbf{a}'$ to use as a feature.
4. Compute variance σ_a^2 of the sum vectored and zero mean signal $\mathbf{a}$ for computation of fractal dimensions.
5. Perform Periodic padding of the zero mean signal $\mathbf{a}$ and compute first-level wavelet transform approximations $\mathcal{A}_1$ and details $\mathcal{D}_1$.
6. Compute the mean μ_{D_1} and variance $\sigma_{D_1}^2$ of the detail coefficients and use the variance σ_a^2 of the signal $\mathbf{a}$ in step 4 to compute the fractal dimension $fd1$ at level 1.
7. Perform Periodic padding of the first-level wavelet detail coefficients and compute second-level wavelet transform approximations $\mathcal{A}_2$ and details $\mathcal{D}_2$.
8. Compute the mean μ_{D_2} and variance $\sigma_{D_2}^2$ of the second level detail coefficients and use the variance σ_a^2 of the signal $\mathbf{a}$ in step 4 to compute the fractal dimension $fd2$ at level 2.
9. Perform Periodic padding of the second-level wavelet detail coefficients and compute third-level wavelet transform approximations $\mathcal{A}_3$ and details $\mathcal{D}_3$.
10. Compute the mean μ_{D_3} and variance $\sigma_{D_3}^2$ of the third level detail coefficients and use the variance σ_a^2 of the signal $\mathbf{a}$ in step 4 to compute the fractal dimension $fd3$ at level 3.
11. Perform Periodic padding of the third-level wavelet detail coefficients and compute fourth-level wavelet transform approximations $\mathcal{A}_4$ and details $\mathcal{D}_4$.
12. Compute the mean μ_{D_4} and variance $\sigma_{D_4}^2$ of the fourth level detail coefficients and use the variance σ_a^2 of the signal $\mathbf{a}$ in step 4 to compute the fractal dimension $fd4$ at level 4.
13. Assemble a feature vector of wavelet approximations at level 4, $\mathcal{A}_4$ [1×8], mean of the sum vectored signal, $\mu_{a'}$ [1×1], variance of the sum vectored signal, $\sigma_{a'}^2$ [1×1] and instantaneous fractal dimensions, $\{fd1, fd2, fd3, fd4\}$ of dimensions [1×4] at all four levels of wavelet transform.
14. Perform classification with LDA machine learning algorithm between falls and no falls.
15. In case of a fall, transmit fall event occurrence for medical aid response.
16. Repeat from step 1 for the next 128 sample window with 50% overlap.

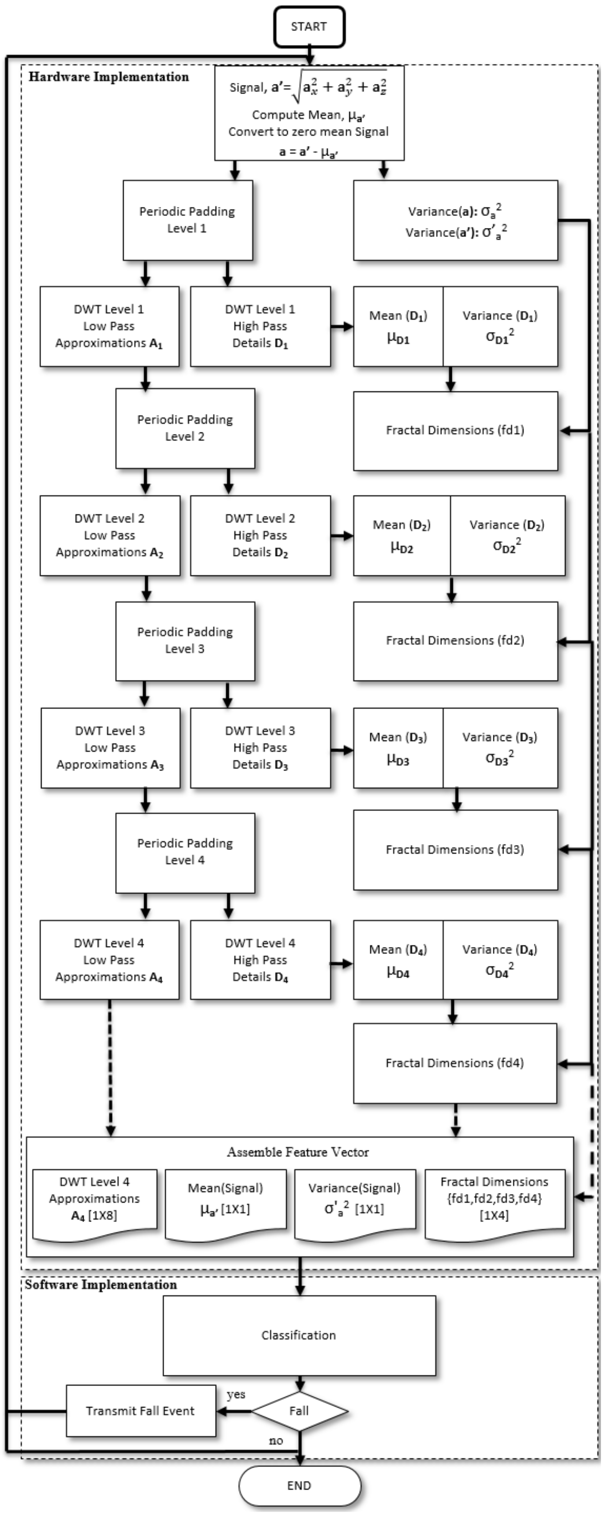

START
Hardware Implementation
Signal, $a' = \sqrt{a_x^2 + a_y^2 + a_z^2}$
Compute Mean, $\mu_{a'}$
Convert to zero mean Signal
$a = a' - \mu_{a'}$
Periodic Padding Level 1
Variance(a): σ_a^2
Variance(a'): σ'^2_a
DWT Level 1 Low Pass Approximations A_1
DWT Level 1 High Pass Details D_1
Mean (D_1) μ_{D1}
Variance (D_1) σ_{D1}^2
Periodic Padding Level 2
Fractal Dimensions (fd1)
DWT Level 2 Low Pass Approximations A_2
DWT Level 2 High Pass Details D_2
Mean (D_2) μ_{D2}
Variance (D_2) σ_{D2}^2
Periodic Padding Level 3
Fractal Dimensions (fd2)
DWT Level 3 Low Pass Approximations A_3
DWT Level 3 High Pass Details D_3
Mean (D_3) μ_{D3}
Variance (D_3) σ_{D3}^2
Periodic Padding Level 4
Fractal Dimensions (fd3)
DWT Level 4 Low Pass Approximations A_4
DWT Level 4 High Pass Details D_4
Mean (D_4) μ_{D4}
Variance (D_4) σ_{D4}^2
Fractal Dimensions (fd4)
Assemble Feature Vector
DWT Level 4 Approximations A_4 [1X8]
Mean(Signal) $\mu_{a'}$ [1X1]
Variance(Signal) σ'^2_a [1X1]
Fractal Dimensions {fd1,fd2,fd3,fd4} [1X4]
Software Implementation
Classification
Transmit Fall Event
yes
Fall
no
END

Figure 4. Flow chart of the proposed scheme.

5.3. Classification Performance of Fractal Features

The classification performance of the DWT based fractal features was determined by implementing the feature extraction process and the LDA machine learning algorithm in Matlab. The algorithm was tested and evaluated on a public fall dataset by Kwolek et al. [8]. The fall dataset [8] is discussed previously in Section 3. The classification performance for different set of features and the proposed scheme are evaluated in Table 3. LDA classification algorithm finds the maximum separation between falls and other activities by maximising between the class variance and minimising the variance within the classes. The fractal features alone can provide high performance with significant classification accuracy of 91.84% with LDA classifier, as compared to level 2, 3 and 4 wavelet coefficients as illustrated in Table 3. In addition, combining fractal and wavelet coefficients results in higher accuracy of 95.61%. Classification accuracy further increases to 99.38% for the proposed scheme, when mean and SD of the signals are added to the feature set, as illustrated in Table 3.

Table 3. Classification accuracy for features.

Features	**LDA Classifier**
	Accuracy (%)
Wavelet Transform Lvl 2	79.77 ($\pm$0.08)
Wavelet Transform Lvl 3	83.09 ($\pm$0.08)
Wavelet Transform Lvl 4	85.76 ($\pm$0.09)
Fractal Features	91.84 ($\pm$0.21)
Fractal, Wavelet Lvl 4	95.61 ($\pm$0.09)
Fractal, Wavelet Lvl 4, Mean, SD	99.38 ($\pm$0.19)

Table 4 compares the performance of the proposed system features and classifier with state-of-the-art work. The latest works on FDS have utilised a number of machine and deep learning classifiers including SVM, Quadrature SVM (QSVM), Random Forest (RF), K-Nearest Neighbour (KNN), Decision Tree (DT), Ensemble Bagged Tree (EBT), Artificial Neural Network (ANN) and deep hybrid Random Neural Network (RNN) as illustrated in Table 4. The systems presented in [60–63] use a large number of features as compared to our system, which utilises only four features namely, fractal dimensions, DWT level 4 coefficients, mean and SD. The works in [60,62,63] leverage additional gyroscopic sensors apart from the tri-axes accelerometer, while [61] uses multiple accelerometer sensors attached to waist, chest and thigh. Our proposed system utilises only one accelerometer sensor attached to the pelvis to achieve the best performance results. The systems in [60,62,63] utilise ensemble learning techniques to achieve their best results along with other classifiers, such as SVM, QSVM and KNN. The work in [64] utilises raw accelerometer values with a deep hybrid RNN classifier, however suffers from the complexity of a deep neural network and provides 7.1% less accuracy than our proposed scheme. The proposed system utilises the LDA classifier and fractal features to give the best performance results in terms of accuracy, sensitivity and specificity of 99.10%, 99.90% and 99.38%, respectively, while utilising low number of features.

Table 4. Classification Performance Comparison.

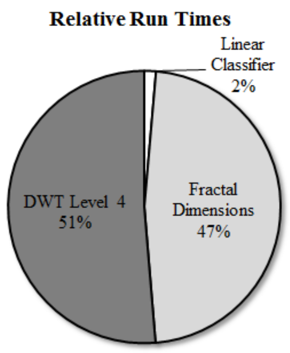
Authors	**KSE'18 [60]**	**IEEE Sensors'18 [61]**	**PEIS'19 [64]**	**IEEE Access'19 [62]**	**IEEE Sensors'19 [63]**	**Proposed FDS**
Dataset	Self-simulated	SisFall Data [65]	Fall Data [8]	Public Datasets	SisFall Data [65]	Fall Data [8]
Sensor	Tri-axes Acc., Gyro.	Tri-axes Acc.	Tri-axes Acc.	Tri-axes Acc., Gyro.	Tri-axes Acc., Gyro.	Tri-axes Acc.
Sensor Location	Hip	Waist, Chest,	Waist	Thigh, Chest	Waist	Pelvis
Features	Mean, SD, Energy, Entropy, Hjorth Mobility and Complexity, Sum Vector etc.	Mean, SD, Maxima, Minima, Kurtosis, Skewness, Corr. Coefficients etc.	x, y, z Axes Acceleration	Mean, Maxima, Minima, Auto Cross Correlation Peak PSD etc.	Mean, Maxima, Minima, SD, Sum Vector, Kurtosis, Skewness etc.	Fractal and Wavelet Lvl 4 Features, Mean, SD
Classifier	SVM, RF	SVM, KNN, DT, Naive Bayes	Deep Hybrid RNN	ANN, KNN, EBT, QSVM	KNN, SVM, RF	LDA
Sensitivity	94.37%	98.30%	-	-	80.07%	99.10%
Specificity	-	-	-	-	98.27%	99.90%
Accuracy	-	97.60%	92.23%	97.70%	96.82%	99.38%

6. Proposed Algorithm Performance Analysis

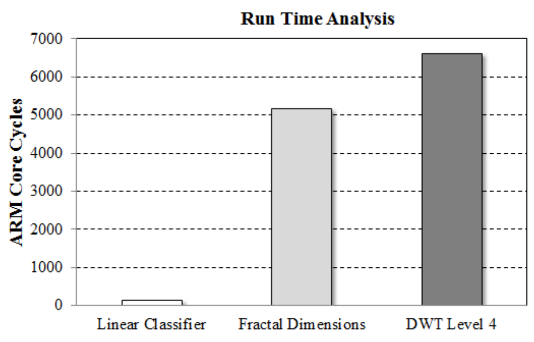
The software performance of the proposed algorithm with DWT based fractal feature extraction and LDA classification algorithm was determined on an embedded ARM cortex A9 at 666 MHz with C/C++ implementation. Xilinx SDK was used for running and evaluating embedded code.

The multi-level wavelet transform and fractal dimension algorithms were observed to be more computationally intensive than the LDA classification algorithm. The DWT algorithm consumed the highest percentage of the total runtime at 51%, slightly higher than the fractal computations which were also computationally expensive at 47% of the total runtime cost. The higher computational load of DWT algorithm resulted from four convolution operations performed for each of the four levels of wavelet transform. Overall, the four convolutions consume more cycles compared to the entire algorithm with approximately 6700 cycles of the embedded ARM core, where the first convolution is computed for 128 samples, second for 64, third for 32 and fourth for 16. The fractal algorithm computes means and variances from the obtained DWT coefficients at each level and utilises the signal variances to compute fractal dimensions, which is also computationally expensive. The fractal algorithm consumes approximately 5000 cycles of the embedded ARM core. The computationally intensive part of the fractal algorithm is the computation of variances of the wavelet detail coefficients obtained from the four convolutions, where variance computations involve subtraction of each accelerometer sample with the mean of the signal and calculation of the square of each term, corresponding to product of each term with itself. The overall computational complexity of the algorithm is $O(n^2)$, which is due to the variance computations performed in the algorithm. Moreover, the machine learning classification performed with the LDA algorithm consumed only 2% of the total runtime cost, which was not considered significant enough for hardware acceleration. The relative execution times are illustrated in Figure 5a, while the run times of the proposed algorithm are shown in Figure 5b.

Due to significantly higher computational requirements, fractal computations and multi-level wavelet transforms were accelerated with custom designed reconfigurable hardware architecture to achieve higher dividends for power, latency and performance per Watt metrics. While LDA classification algorithm was executed as embedded software program on the ARM processor.



(**a**) Relative run times



(**b**) Run times

Figure 5. Run time analysis of the proposed algorithm.

7. FDS Hardware/Software Co-Design

The hardware/software co-design along with the hardware architecture, design and optimizations are discussed in this section. The design flow consists of synthesizing the high level C++ code with the Xilinx Vivado high level synthesizer to an equivalent Register Transfer Language (RTL). The RTL is then exported in the form of a Vivado Intellectual Property (IP). The IP along with other peripherals and the ARM core IP block are imported in the Vivado block diagram environment. The system is synthesised and exported to the Xilinx SDK for running embedded code. The SoC consists of the Processing System (PS) and the PL part. The PS with the ARM cores, caches, ports and controllers is connected to the PL with the IPs provided by Xilinx as illustrated in Figure 6. The IPs required for SoC design are Xilinx proprietary AXI protocol interconnects for AXI buses, AXI timer, the AXI Direct Memory Access (DMA) and a PS reset module. The AXI stream interface allows the hardware accelerator to connect to the PS through the AXI DMA IP. The DDR memory has higher throughput than the L2 cache and L2 cache provides lower latency. Due to low latency requirements associated with a real-time classification task, our SoC design connects the DMA to the L2 cache through the Accelerator Coherency Port (ACP). The ACP connects to the L2 cache via the Snoop Control Unit (SCU) which keeps all reads and writes coherent between the accelerator and the ARM cores. A number of synthesis optimizations were utilised for design optimisations.

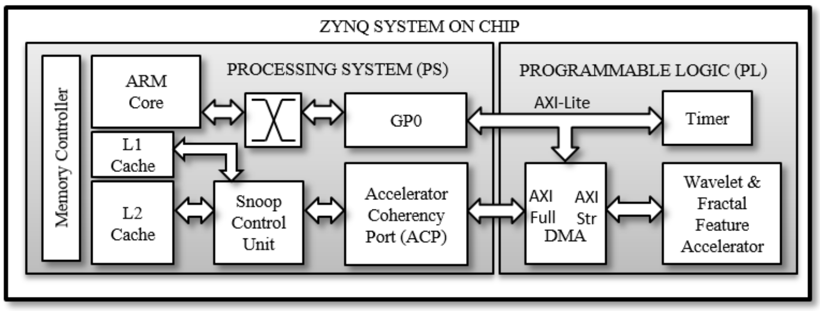


Figure 6. Zynq SoC design for a fall detection system.

The major parts of the design were implemented in the C/C++ high level programming language. The algorithm implementation in C/C++ is synthesized in Xilinx Vivado HLS design suite to verify the timing and design parameters, such as operating clock frequency and design latency. The high level code is then further optimised by manual code restructuring/rewriting and introduction of HLS optimization directives such as PIPELINE, LOOP UNROLL etc. The synthesis parameters are again verified for timing and design parameters, until satisfactory results are achieved.

7.1. Hardware Acceleration

Multi-Level Wavelet Transform Hardware: The low pass filter and high pass filter wavelet coefficients are calculated from the original signal with periodic padding and the process is repeated for 4-level wavelet transform. Next level wavelet transform is computed from the previous low pass wavelet coefficients. The final level-4 low pass wavelet coefficients are used as features in the classification, while the detail coefficients from each stage are used to determine the fractal dimensions. The hardware DWT implementation is based on signal convolution with filter coefficients and subsequent down sampling of signal values. Our implementation stores the filter coefficients in the reverse order in block RAM and does not require flipping the filter coefficients for convolution. It allows us to access 8 simultaneous signal samples as well as filter coefficients for Daubechies 4 wavelet from the block RAM partitioned into 8 blocks with two reading ports each. A single loop computes both set of wavelet coefficients low pass approximations and high pass details. The computation for every alternate coefficient is skipped instead of down sampling later to save computation time and power. The unoptimised and optimised DWT algorithms are illustrated in Algorithms 1 and 2, respectively. The computation of a single coefficient is done with an eight sum of products pipelined operation for low latency processing. The arithmetic tree for computation of wavelet transform is illustrated in Figure 7.

Fractal Feature Hardware: The fractal dynamics is calculated for each level of wavelet detail coefficients. The fractal dimension is based on an an efficient log to the base 2 operation of the ratio of variance of wavelet detail coefficients to the original zero mean signal. The computation of mean and variance is performed through arithmetic trees and the fractal dimension is calculated with the synthesized circuit operations given in Figure 8. The algorithm is illustrated in Algorithm 3.

LDA Classifier: The coefficients obtained from LDA training are used in the classifier to evaluate a discriminant function between falls and no falls. The classifier is implemented in software on the processing system ARM core. The set of feature values are received from the hardware accelerator through the ACP port connected to the AXI bus. The features are read by the ARM core and the discriminant function is computed and the output is fed to a simple decision tree for a fall or a no fall decision, which determines if the value is ≥ 0 for a fall decision or not. The hardware accelerator is based on five design iterations. The final design is proposed as an efficient feature extraction accelerator for classification. The latency of the proposed arithmetic circuits varied from 8.6 to 9.2 nsec during the synthesis phase, hence a clock period of 10nsec, corresponding to the frequency of 100 MHz was utilised for the entire design. The designs with their optimizations and code restructuring are explained below:

- **Design I**: The design I consists of embedding the wavelet filter coefficients in local memory, to allow hardware to perform fast operations with low latency access to filter coefficients and reduced main memory access operations. The intermediate result arrays used in the algorithm are also embedded in local memory for fast read and write access. Furthermore, the functions are inlined to take advantage of synthesis optimizations with their surrounding code.
- **Design II**: The design II is further optimised by adding pipelines to the padding implementation, wavelet transform and variance calculation for computation of fractal dimensions.
- **Design III**: The design III consists of unrolling the loops over and above design II. The loops representing convolution operations in wavelet transform and variance computations are unrolled by a factor of 8.
- **Design IV**: The design IV is implemented with arithmetic trees. It only assumes elements of design I for its implementation. The computations of wavelet transform and variance for fractal dimensions are resolved into tree structures. It requires code restructuring and rewriting. The loops are manual unrolled to accommodate computational arithmetic tree structures.
- **Final Design**: The final design is based on pipelining the arithmetic trees implemented in design IV and manual unroll. Along with the optimizations of design I and IV, here we propose the technique of skipping the alternative computations of the wavelet filter convolutions instead of downsampling after the convolution operation is performed. The downsampling is embedded in the convolution computation rather than implemented separately. The integration reduces latency and number of stages within the algorithm, resulting in savings to execution time, logic and memory resources.

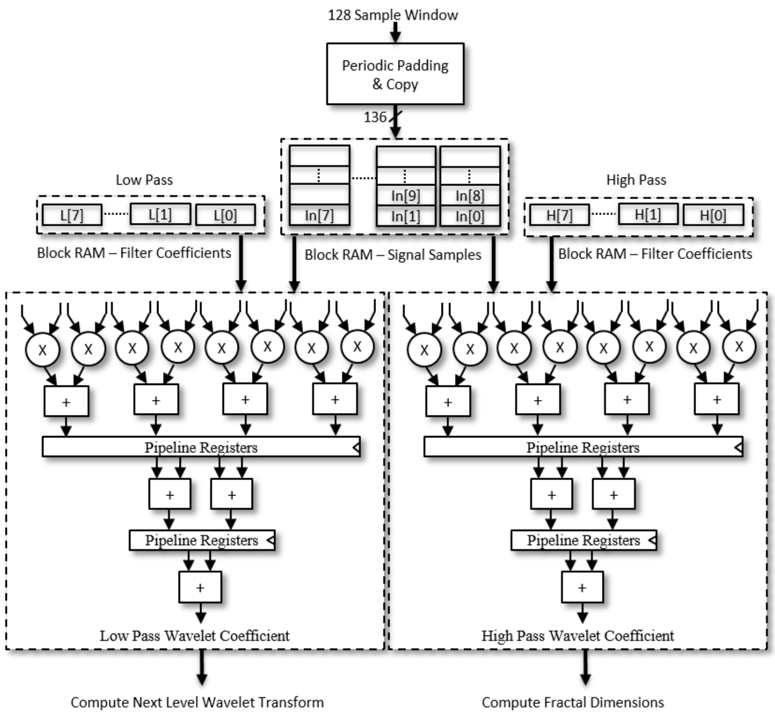

128 Sample Window
Periodic Padding & Copy
136
Low Pass
L[7]
L[1]
L[0]
Block RAM – Filter Coefficients
In[7]
In[9]
In[1]
In[8]
In[0]
Block RAM – Signal Samples
High Pass
H[7]
H[1]
H[0]
Block RAM – Filter Coefficients
X
+
Pipeline Registers
Low Pass Wavelet Coefficient
High Pass Wavelet Coefficient
Compute Next Level Wavelet Transform
Compute Fractal Dimensions

Figure 7. Hardware arithmetic pipeline of wavelet transform.

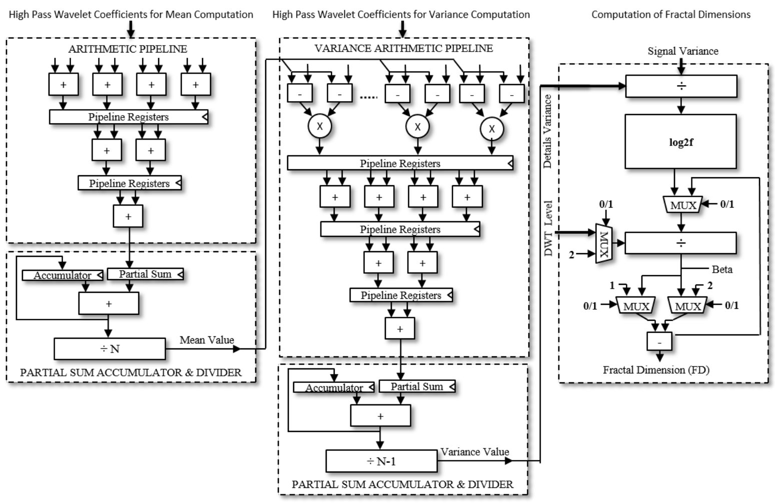


Figure 8. Logic circuit for mean, variance and fractal dimension computations.

Algorithm 1: Discrete Wavelet Transform Algorithm.

```
Input: $i \leftarrow$ DWT level;
$a[n] \leftarrow$ sum vectored, zero mean signal;
$N \leftarrow$ sum vectored, zero mean signal length;
$\phi_f[n_f] \leftarrow$ flipped low pass filter coefficients;
$\psi_f[n_f] \leftarrow$ flipped high pass filter coefficients;
$N_f \leftarrow$ wavelet filter coefficients length;
$a_p[n_p] \leftarrow$ padded signal a[n];
$N_p \leftarrow N + N_f - 1$, padded signal length;
$out_{\mathcal{A}_i}[k] \leftarrow$ low pass filter output;
$out_{\mathcal{D}_i}[k] \leftarrow$ high pass filter output;
$\mathcal{A}_i[k] \leftarrow$ final low pass output: Approximations;
$\mathcal{D}_i[k] \leftarrow$ final high pass output: Details;
Output: $\mathcal{A}_i[k]$, $\mathcal{D}_i[k]$ wavelet coefficients
1  Function DWT($a[n]$, $\phi_f[n_f]$, $\psi_f[n_f]$, $\mathcal{A}_i[k]$, $\mathcal{D}_i[k]$):
2      $out_{\mathcal{A}_i}[k], out_{\mathcal{D}_i}[k] \leftarrow$ WAVELET_CONVOLUTION($a[n], \phi_f[n_f], \psi_f[n_f]$);
3      $\mathcal{A}_i[k], \mathcal{D}_i[k] \leftarrow$ DOWNSAMPLING($out_{\mathcal{A}_i}[k], out_{\mathcal{D}_i}[k]$);
4      return $\mathcal{A}_i[k], \mathcal{D}_i[k]$;
5  Function WAVELET_CONVOLUTION($a_p[n_p], \phi_f[n_f], \psi_f[n_f]$):
6      $k \leftarrow 0$;
7      for $n_p \in [0, 1, \ldots, N_p - N_f]$ do
8          $mac_1 \leftarrow 0, mac_2 \leftarrow 0$ ;
9          for $n_f \in [0, \ldots, N_f - 1]$ do
10             $mac_1 \leftarrow mac_1 + a_p[n_f + n_p] \times \phi_f[n_f]$;
11             $mac_2 \leftarrow mac_2 + a_p[n_f + n_p] \times \psi_f[n_f]$;
12         $out_{\mathcal{A}_i}[k] \leftarrow mac_1$;
13         $out_{\mathcal{D}_i}[k] \leftarrow mac_2$;
14         $k \leftarrow k + 1$;
15     return $out_{\mathcal{A}_i}[k], out_{\mathcal{D}_i}[k]$;
16 Function DOWNSAMPLING($out_{\mathcal{A}_i}[k]$, $out_{\mathcal{D}_i}[k]$):
17     for $k \in [0, 1, \ldots, \lceil \frac{N}{2} \rceil - 1]$ do
18         $\mathcal{A}_i[k] \leftarrow out_{\mathcal{A}_i}[k \times 2]$;
19         $\mathcal{D}_i[k] \leftarrow out_{\mathcal{D}_i}[k \times 2]$;
20     return $\mathcal{A}_i[k], \mathcal{D}_i[k]$;
```

Algorithm 2: Proposed Optimised Discrete Wavelet Transform.

```
Input: $i \leftarrow$ DWT level
$a[n] \leftarrow$ sum vectored, zero mean signal
$N \leftarrow$ sum vectored, zero mean signal length
$\phi_f[n_f] \leftarrow$ flipped low pass filter coefficients
$\psi_f[n_f] \leftarrow$ flipped high pass filter coefficients
$N_f \leftarrow$ wavelet filter coefficients length
$a_p[n_p] \leftarrow$ padded signal a[n]
$N_p \leftarrow N + N_f - 1$, padded signal length
$out_{\mathcal{A}_i}[k] \leftarrow$ low pass filter output
$out_{\mathcal{D}_i}[k] \leftarrow$ high pass filter output
$\mathcal{A}_i[k] \leftarrow$ final low pass output: Approximations
$\mathcal{D}_i[k] \leftarrow$ final high pass output: Details
Output: $\mathcal{A}_i[k], \mathcal{D}_i[k]$ wavelet coefficients
Function DWT($a[n], \phi_f[n_f], \psi_f[n_f]$):
    #pragma HLS array_partition $a_p[n_p]$ cyclic 8
    #pragma HLS array_partition $\phi_f[n_f]$ cyclic 8
    #pragma HLS array_partition $\psi_f[n_f]$ cyclic 8
    #pragma HLS array_partition $\mathcal{A}_i[k]$ cyclic 8
    #pragma HLS array_partition $\mathcal{D}_i[k]$ cyclic 8
    $k \leftarrow 0$
    $prod_\phi[] \leftarrow 0, prod_\psi[] \leftarrow 0, sum^1_\phi[] \leftarrow 0, sum^1_\psi[] \leftarrow 0$
    $sum^2_\phi[] \leftarrow 0, sum^2_\psi[] \leftarrow 0, sum^3_\phi[] \leftarrow 0, sum^3_\psi[] \leftarrow 0$
    while $n_p \in [0, \ldots, N_p - N_f]$ do
        #pragma HLS pipeline
        /* Independent arithmetic trees for low and high pass filter convolutions */
        $prod_\phi[0] \leftarrow a_p[0 + n_p] \times \phi_f[0],\ prod_\psi[0] \leftarrow a_p[0 + n_p] \times \psi_f[0]$
        $prod_\phi[1] \leftarrow a_p[1 + n_p] \times \phi_f[1],\ prod_\psi[1] \leftarrow a_p[1 + n_p] \times \psi_f[1]$
        $prod_\phi[2] \leftarrow a_p[2 + n_p] \times \phi_f[2],\ prod_\psi[2] \leftarrow a_p[2 + n_p] \times \psi_f[2]$
        $prod_\phi[3] \leftarrow a_p[3 + n_p] \times \phi_f[3],\ prod_\psi[3] \leftarrow a_p[3 + n_p] \times \psi_f[3]$
        $prod_\phi[4] \leftarrow a_p[4 + n_p] \times \phi_f[4],\ prod_\psi[4] \leftarrow a_p[4 + n_p] \times \psi_f[4]$
        $prod_\phi[5] \leftarrow a_p[5 + n_p] \times \phi_f[5],\ prod_\psi[5] \leftarrow a_p[5 + n_p] \times \psi_f[5]$
        $prod_\phi[6] \leftarrow a_p[6 + n_p] \times \phi_f[6],\ prod_\psi[6] \leftarrow a_p[6 + n_p] \times \psi_f[6]$
        $prod_\phi[7] \leftarrow a_p[7 + n_p] \times \phi_f[7],\ prod_\psi[7] \leftarrow a_p[7 + n_p] \times \psi_f[7]$

        $sum^1_\phi[0] \leftarrow prod_\phi[0] + prod_\phi[1],\ sum^1_\psi[0] \leftarrow prod_\psi[0] + prod_\psi[1]$
        $sum^1_\phi[1] \leftarrow prod_\phi[2] + prod_\phi[3],\ sum^1_\psi[1] \leftarrow prod_\psi[2] + prod_\psi[3]$
        $sum^1_\phi[2] \leftarrow prod_\phi[4] + prod_\phi[5],\ sum^1_\psi[2] \leftarrow prod_\psi[4] + prod_\psi[5]$
        $sum^1_\phi[3] \leftarrow prod_\phi[6] + prod_\phi[7],\ sum^1_\psi[3] \leftarrow prod_\psi[6] + prod_\psi[7]$

        $sum^2_\phi[0] \leftarrow sum^1_\phi[0] + sum^1_\phi[1],\ sum^2_\psi[0] \leftarrow sum^1_\psi[0] + sum^1_\psi[1]$
        $sum^2_\phi[1] \leftarrow sum^1_\phi[2] + sum^1_\phi[3],\ sum^2_\psi[1] \leftarrow sum^1_\psi[2] + sum^1_\psi[3]$

        $sum^3_\phi[0] \leftarrow sum^2_\phi[0] + sum^2_\phi[1],\ sum^3_\psi[0] \leftarrow sum^2_\psi[0] + sum^2_\psi[1]$

        $\mathcal{A}_i[k] \leftarrow sum^3_\phi[0]$
        $\mathcal{D}_i[k] \leftarrow sum^3_\psi[0]$
        $k \leftarrow k + 1$
        $n_p \leftarrow n_p + 2$ // Downsampling embedded in loop
    return $\mathcal{A}_i[k], \mathcal{D}_i[k]$
```

Algorithm 3: Proposed FRACTAL_DIM.

```
Input: $i \leftarrow$ DWT level
$N \leftarrow$ sum vectored, zero mean signal length
$n_D \leftarrow$ wavelet detail coefficients length
$a[n] \leftarrow$ sum vectored, zero mean signal
$\mathcal{D}_i[k] \leftarrow$ wavelet detail coefficients
Output: fd fractal dimensions
Function FRACTAL_DIM($a[n]$, $\mathcal{D}_i[k]$, $i$):
    #pragma HLS array_partition $a[n]$ cyclic 8
    #pragma HLS array_partition $\mathcal{D}_i[k]$ cyclic 8
    #pragma HLS inline
    $\mu_D \leftarrow \text{MEAN}(\mathcal{D}_i[k], n_D)$
    $\sigma_D^2 \leftarrow \text{VARIANCE}(\mathcal{D}_i[k], n_D, \mu_D)$
    $\sigma_a^2 \leftarrow \text{VARIANCE}(a[n], N, \mu_a = 0)$
    $\beta \leftarrow \frac{1}{i} \times \log_2(\sigma_s^2 / \sigma_D^2)$
    $\mathcal{H} \leftarrow (\beta - 1)/2$
    $fd \leftarrow 2 - \mathcal{H}$
    return $fd$
Function MEAN($\mathcal{D}_i[k]$, $n_D$):
    $sum_1[] \leftarrow \emptyset, sum_2[] \leftarrow \emptyset, sum_3[] \leftarrow \emptyset$
    for $k \in [0, 8, \ldots, n_D - 8]$ do
        #pragma HLS pipeline
        /* Arithmetic tree with width 8 */
        $sum_1[0] \leftarrow \mathcal{D}_i[k] + \mathcal{D}_i[k+1]$
        $sum_1[1] \leftarrow \mathcal{D}_i[k+2] + \mathcal{D}_i[k+3]$
        $sum_1[2] \leftarrow \mathcal{D}_i[k+4] + \mathcal{D}_i[k+5]$
        $sum_1[3] \leftarrow \mathcal{D}_i[k+6] + \mathcal{D}_i[k+7]$

        $sum_2[0] \leftarrow sum_1[0] + sum_1[1]$
        $sum_2[1] \leftarrow sum_1[2] + sum_1[3]$

        $sum_3[0] \leftarrow sum_2[0] + sum_2[1]$
    $\mu_D \leftarrow sum_3[1] / n_D$
    return $\mu_D$
Function VARIANCE($\mathcal{D}_i[k]$, $n_D$, $\mu_D$):
    $sub_1[] \leftarrow \emptyset, prod_1[] \leftarrow \emptyset$
    $sum_1[] \leftarrow \emptyset, sum_2[] \leftarrow \emptyset, sum_3[] \leftarrow \emptyset$
    for $k \in [0, 8, \ldots, n_D - 8]$ do
        #pragma HLS pipeline
        /* Arithmetic tree with width 8 */
        $sub_1[0] \leftarrow \mathcal{D}_i[k] - \mu_D$
              ..
        $sub_1[7] \leftarrow \mathcal{D}_i[k+7] - \mu_D$

        $prod_1[0] \leftarrow sub_1[0] \times sub_1 01]$
              ..
        $prod_1[7] \leftarrow sub_1[7] \times sub_1[7]$

        $sum_1[0] \leftarrow prod_1[0] + prod_1[1]$
              ..
        $sum_1[3] \leftarrow prod_1[6] + prod_1[7]$

        $sum_2[0] \leftarrow sum_1[0] + sum_1[1]$
        $sum_2[1] \leftarrow sum_1[2] + sum_1[3]$

        $sum_3[0] \leftarrow sum_2[0] + sum_2[1]$
    $\sigma_D^2 \leftarrow sum_3[0] / (n_D - 1)$
    return $\sigma_D^2$
```

Synthesis Results: The resources required for the proposed hardware accelerators are given in Table 5. The final design uses around 3× more resources than the design I. However, the overall resource utilization remains relatively low at 28.67%, which is a significantly low value keeping in view a decrease of 10× in latency compared to design I and keeps logic area low for power efficient design.

Table 5. Hardware designs resource utilization and latency.

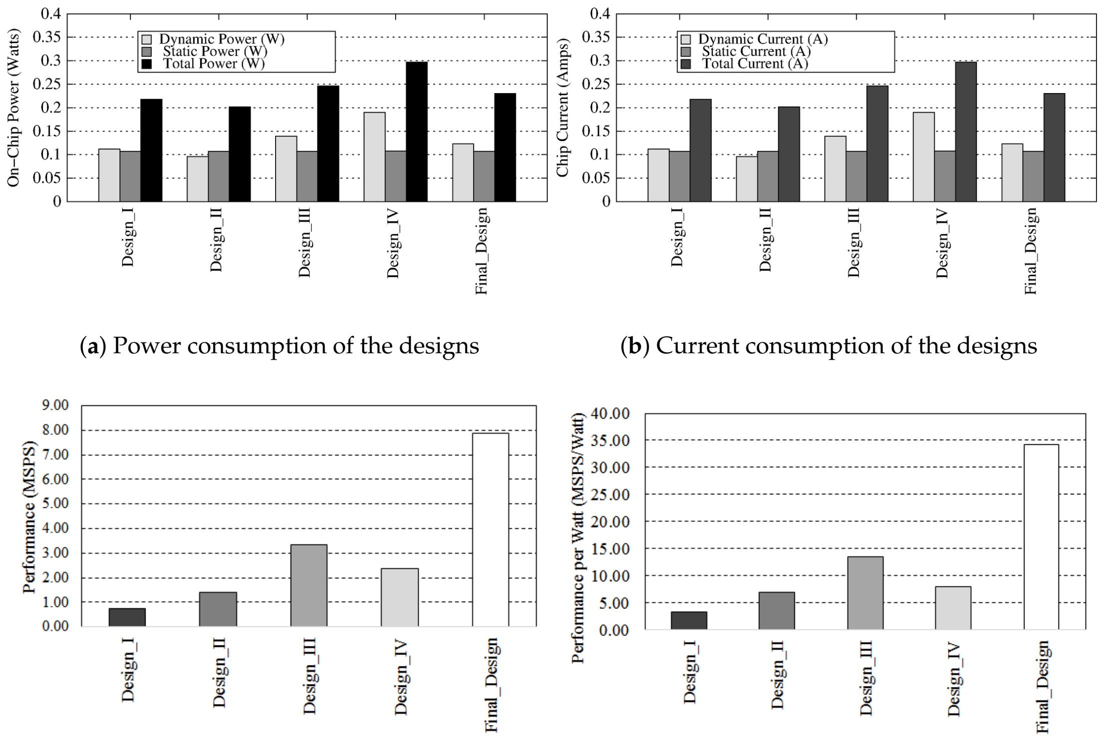
Hardware Resources	**Design I**		**Design II**		**Design III**		**Design IV**		**Final Design**	
	Used	**%**	**Used**	**%**	**Used**	**%**	**Used**	**%**	**Used**	**%**
LUT Logic	2516	4.73	2775	5.22	4394	8.26	5403	10.16	7222	13.58
CARRY4	141	1.06	141	1.06	139	1.05	139	1.05	139	1.05
Register	3328	3.13	3637	3.42	7370	6.93	10,161	9.55	12,853	12.08
LUT Shift Reg.	112	0.64	121	0.7	1419	8.16	132	0.76	147	0.84
LUT Dist. RAM	160	0.92	160	0.92	224	1.29	288	1.66	192	1.1
Muxes	10	0.02	10	0.02	15	0.03	10	0.02	10	0.02
Total	6902	10.5	7459	11.34	14,265	25.72	17,353	23.2	21,649	28.67
Block RAM	7	5	7	5	8.5	6.07	10	7.14	12	7.14
DSP48E	10	4.55	10	4.55	10	4.55	10	4.55	10	4.55
I/O	119	59.5	119	59.5	119	59.5	119	59.5	119	59.5
Latency cycles at 100 MHz	17381		9070		3841		5423		1629	

8. Hardware System Results

Power Consumption: The power consumption of the design iterations and the final design is given in Table 6. The final optimised design consumes 0.23 W power, an increase of 4.8% over design I with 10× improvement in latency. Furthermore, it shows an improvement of 6.52× over the power consumption of the ARM core at 666 MHz. The dynamic, static and total power of all designs is illustrated in Figure 9a, while the current intake of the hardware designs is shown in Figure 9b. The dynamic power consumption of the final design remains almost the same, however there is a small increase in static power consumption. The current consumption verifies this since there is an increase in static current.

Speed-Up: The final design shows a speed-up of 7.3× over software execution as illustrated in Figure 10. The design III with pipelining and unrolling of loops in wavelet transform and variance computations gives around 3× improvement over software execution, however further improvements to design does not result in any considerable advantage but incurs high area overhead costs. The arithmetic tree design decision with clever optimizations to code structure and flow were deemed necessary for good results.

Performance & Performance per Watt: The performance of the design in Million Samples Processed per Second (MSPS) is shown in Figure 9c and performance per Watt in MSPS per Watt is illustrated in Figure 9d. The final design has the highest performance at 7.86 MSPS, an improvement of 10.68× over design I and a performance per Watt of 34.16 MSPS per Watt, an improvement of 10.11× over design I. The improvements in performance per Watt over software execution are 47.6×.



(**a**) Power consumption of the designs (**b**) Current consumption of the designs

(**c**) Performance in processed samples/sec (**d**) Performance per Watt of the designs

Figure 9. Power and performance results of hardware designs.

The proposed FDS is compared with current and comparable work in Table 7. The proposed FDS demonstrated the highest accuracy, sensitivity and precision with 99.38%, 99.10% and 99.96%, respectively. The proposed system has the lowest latency of 0.1 µs/sample compared to all the given systems. Our system achieves the lowest latency of 0.1 µs/sample by utilising pipelined arithmetic tree structures, which can process an entire window segment of 128 samples in 1629 cycles at 100 MHz operating frequency. The processing of entire window segment with arithmetic trees results in a higher throughput and an effective latency of 0.1 µs/sample with 128 sample bursts of data streaming into the proposed hardware design. The latest works in [66,67] have a higher latency of 1 s, while [47,50,51] have latencies in microseconds, however they suffer form lower classification performance. Furthermore, the proposed system also consumes relatively low to moderate resources over all hardware implementations from design I to final design, which allows flexibility in performance and area trade-off requirements for processing. The final design consumes lower LUT resources than [67,68], and final optimised design in [47] with 17.63%, 18.43% and 26.29% of the LUT resource consumption of the designs, respectively. While our final design consumes higher LUT resources at 167% and 292% of [50,51] respectively, but has 10% higher sensitivity and 2.9% higher specificity than both the designs. The first design iteration design I of our proposed system can be utilised for low area/resource consumption and consumes the second lowest LUT resources of all the designs in Table 7 with only [51] consuming 2% less resources. Design II also consumes low resources in Table 7 with [51] consuming 10.9% less resources. Design III and IV are placed in the middle of Table 7, in terms of LUT resource consumption, lower than [67,68], but higher than [50,51].

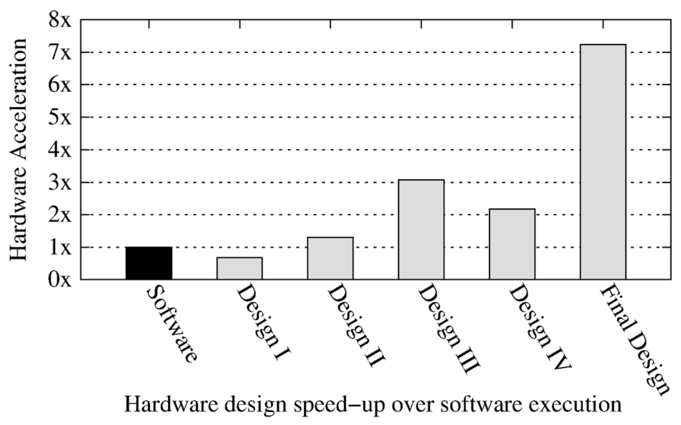


Figure 10. Hardware speed-up relative to software execution.

Table 6. Hardware designs power consumption.

Hardware Resources	**Design I (Watts)**	**Design II (Watts)**	**Design III (Watts)**	**Design IV (Watts)**	**Final Design (Watts)**
Clocks	0.029	0.029	0.044	0.056	0.047
LUT Logic	0.013	0.014	0.019	0.025	0.015
CARRY4	0.001	0.001	0.001	0.001	<0.001
Register	0.001	0.001	0.002	0.003	0.001
LUT Shift Reg.	<0.001	<0.001	<0.001	<0.001	<0.001
LUT Dist. RAM	<0.001	<0.001	<0.001	<0.001	<0.001
Muxes	<0.001	<0.001	<0.001	<0.001	<0.001
Total	0.016	0.016	0.022	0.03	0.017
Signals	0.022	0.022	0.037	0.052	0.029
Block RAM	0.02	0.013	0.018	0.026	0.016
DSP48E	0.006	0.006	0.006	0.005	0.002
I/O	0.019	0.011	0.012	0.02	0.012
Static Power	0.107	0.107	0.107	0.108	0.107
Total Power	0.218	0.202	0.246	0.297	0.23

Similarly, the proposed design consumes lower number of signal processing blocks DSP48E at 10, while the other designs in [51,68] and the final optimised design in [47] consume 12, 28 and 70 blocks, respectively. The unoptimised initial design in [47], however consumes half the number of signal processing blocks but suffers from lower fall detection performance. The proposed design consumes higher RAM blocks in order to simultaneous provide data samples and filter coefficients to pipelined arithmetic trees, which results in the lowest latency of all designs.

All of the current and comparable works in Table 7 except the proposed system suffer from lack of reproducible classification results. The classification performance and accuracies of the proposed system in Table 7 are reproducible due to validation through a public dataset, while all the remaining works lack reproducibility of results due to the use of self-simulated experiments for which data is not available publicly. While the system in [67] also utilises the Mobifall dataset [26], however the dataset is combined with self-simulated experiments and provides combined results which are not reproducible.

Table 7. Proposed algorithm system comparison with current comparable work.

	AICCSA'14 [47]	**SAI'16** [68]	**AICCSA'16** [50]	**IDT'16** [51]	**BHI'17** [66]	**TBioCAS'19** [67]	**Proposed System**
Sensor	Tri-axes Acc.	ECG + Acc.	Tri-axes Acc.	Tri-axes Acc.	Tri-axes Acc.	Tri-axes Acc.	Tri-axes Acc.
Method	PCA + DT	K-NN	Thresholding and SVM	Thresholding and SVM	Thresholding	Thresholding	Fractal Features
Processor	Zynq	Zynq Z-7000	Zynq Z-7010	Zynq Z-7010	FPGA	Virtex5	Zynq Z-7020
Frequency	121–298 MHz	100 MHz	-	-	-	1 KHz	100 MHz
Latency	97.86–2 μs	-	0.86 μs	6.34 μs	1 s	1 s	0.1 μs/sample
LUT	3381–27465	40955	4314	2470	-	39190	2516–7222
Flip Flops	1352–10129	24015	3815	2539	-	21750	3328–12853
BRAM	4–0	2	1	2	-	-	12
DSP48E	5–70	28	-	12	-	-	10
Validation	Self-simulated	Self-simulated	Self-simulated	Self-simulated	Self-simulated	Self-simulated	Fall Dataset [8]
Reproducibility	x	x	x	x	x	x	✓
Sensitivity	-	-	89.50%	89.50%	98.10%	98.60%	99.10%
Specificity	-	-	97.00%	97.00%	99.20%	99.10%	99.90%
Accuracy	88.40%	82.14%	-	-	-	-	99.38%

9. Conclusions

This work presents the hardware/software co-design of a novel fall detection system based on fractal features for learning and classification. Fractal dimensions capture irregularity characteristics of the signal and provide good discriminant capability for falls against ADLs with a high classification accuracy of 99.38%. While, wavelet transforms can be used to compute fractal dimensions of non-stationary signals, the process is computationally intensive. An embedded wearable SoC for our proposed fall detection based on fractal features can provide higher accuracy at low performance per Watt through reconfigurable design innovations. The computationally intensive wavelet transform and fractal dimension computations are accelerated on reconfigurable hardware through clever design optimizations, while LDA based machine learning algorithm is implemented in parallel on the ARM Cortex A9 chip. The 100 MHz design frequency provided high throughput and low latency with a good trade-off between performance and power consumption, which was sufficient for our case. The final hardware design gives 7.3× speed-up and 6.53× improvements in power consumption, compared to the software only execution. The final design also provides the lowest latency of 0.1 μs/sample processed as compared to all the current FPGA based implementations of FDS. Overall the performance per Watt yields advantage of 47.6×. The proposed FDS provides high throughput with a high classification accuracy, sensitivity and precision of 99.38%, 99.10% and 99.96% respectively. Additionally, the classification performance of the proposed system is reproducible due to validation through public dataset, unlike current embedded FPGA implementations of FDS which utilise self-simulated experiments. Moreover, the proposed FDS consumes relatively low programmable logic resources of the Zynq device at 28.67%. Furthermore, the achieved performance provides the sustainability and scalability for multiple multichannel sensors for future work.

Author Contributions: Conceptualization, A.T.; methodology, A.T.; software, A.T.; validation, A.T.; formal analysis, A.T.; investigation, A.T.; writing–original draft preparation, A.T.; visualization, A.T.; writing–review and editing, A.T., R.M.G., G.M.; supervision, R.M.G., G.M., D.A.S.; project administration, R.M.G., G.M., D.A.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research receive no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Mathematical Notation

$\mathbf{a}$	Sum vectored, zero mean acceleration signal vector
$\mathbf{a}'$	Sum vectored, acceleration signal vector
$\mathbf{a_x}$	Acceleration signal vector along x-axis
$\mathbf{a_y}$	Acceleration signal vector along y-axis
$\mathbf{a_z}$	Acceleration signal vector along z-axis
$a(n)$	Acceleration signal at sample n
$\mathcal{A}_i$	Wavelet approximation coefficients at level i
β	Signal spectral exponent

d	Fractional difference parameter
$\tilde{d}$	Fractional integration coefficient
d_c	Drift constant in stationary test
$\mathcal{D}_i$	Wavelet detail coefficients at level i
$e(n)$	Innovation process in stationary test
$\epsilon(n)$	Random variable
f_s	Sampling frequency
fd	Fractal dimension
γ	Autoregressive coefficient with value < 1
Γ	Gamma Function
H	Hurst exponent
H_0	Null hypothesis in stationary test
H_1	Alternative hypothesis in stationary test
i	Wavelet transform level
$I(d)$	Integrated part of ARIMA/ARFIMA model
k	Shift/translation index
$\mathbb{L}$	Lag operator
λ	Deterministic trend with coefficient
μ_a	Mean of sum vectored, zero mean acceleration signal vector
$\mu_{a'}$	Mean of sum vectored acceleration signal vector
μ_{D_i}	Mean of wavelet detail coefficients at level i
μ_s	Mean of accelerometer first order difference signal
n	Signal sample number
N	Total number of samples
ω	Angular frequency
$p - value$	Confidence value for unit root tests
$p_{max,ADF}$	Maximum delay value for ADF test
$p_{max,KPSS}$	Maximum delay value for KPSS test
$\phi_{i,k}(n)$	Scaling wavelet function with shift index k, at level i
$\psi_{i,k}(n)$	Mother wavelet function with shift index k, at level i
q	Number of random perturbations
r	Number of autoregressive terms
$s(n)$	First order difference, accelerometer signal
$S(\omega)$	Power spectrum density
σ_a^2	Variance of sum vectored, zero mean acceleration signal vector
$\sigma_{a'}^2$	Variance of sum vectored acceleration signal vector
$\sigma_{D_i}^2$	Variance of wavelet detail coefficients at level i
θ_p	General moving average coefficients
$v_1(n)$	Stationary process
$v_2(n)$	Independent and identically distributed process with 0 mean
ζ_r	General autoregressive coefficients

References

1. Tian, Y.; Thompson, J.; Buck, D.; Sonola, L. *Exploring the System-Wide Costs of Falls in Older People in Torbay*; King's Fund: London, UK, 2013.
2. Noury, N.; Rumeau, P.; Bourke, A.K.; ÓLaighin, G.; Lundy, J.E. A proposal for the classification and evaluation of fall detectors. *Innov. Res. Biomed. Eng.* **2008**, *29*, 340–349. [CrossRef]
3. Rossignol, S.; Dubuc, R.; Gossard, J.P. Dynamic Sensorimotor Interactions in Locomotion. *Physiol. Rev.* **2006**, *86*, 89–154. [CrossRef]
4. Mandelbrot, B. How long is the coast of Britain? Statistical self-similarity and fractional dimension. *Science* **1967**, *156*, 636–638. [CrossRef] [PubMed]
5. Sowell, F. Modeling long-run behavior with the fractional ARIMA model. *J. Monet. Econ.* **1992**, *29*, 277–302. [CrossRef]
6. Stadnitski, T. Measuring fractality. *Front. Physiol.* **2012**, *3*, 127. [CrossRef] [PubMed]

7. Diebolt, C.; Guiraud, V. A note on long memory time series. *Qual. Quant.* **2005**, *39*, 827–836. [CrossRef]
8. Kwolek, B.; Kepski, M. Human fall detection on embedded platform using depth maps and wireless accelerometer. *Comput. Methods Progr. Biomed.* **2014**, *117*, 489–501. [CrossRef] [PubMed]
9. Flandrin, P. Wavelet Analysis and Synthesis of Fractional Brownian Motion. *IEEE Trans. Inf. Theory* **1992**, *38*, 910–917. [CrossRef]
10. Wornell, G.W.; Oppenheim, A.V. Estimation of Fractal Signals from Noisy Measurements Using Wavelets. *IEEE Trans. Signal Process.* **1992**, *40*, 611–623. [CrossRef]
11. Struharik, R.; Vukobratovic, B. AIScale—A coarse grained reconfigurable CNN hardware accelerator. In Proceedings of the 2017 IEEE East—West Design and Test Symposium, EWDTS 2017, Novi Sad, Serbia, 29 September–2 October 2017.
12. Wang, C.; Gong, L.; Yu, Q.; Li, X.; Xie, Y.; Zhou, X. DLAU: A scalable deep learning accelerator unit on FPGA. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2017**, *36*, 513–517. [CrossRef]
13. Hsieh, C.Y.; Shi, W.T.; Huang, H.Y.; Liu, K.C.; Hsu, S.J.; Chan, C.T. Machine learning-based fall characteristics monitoring system for strategic plan of falls prevention. In Proceedings of the 4th IEEE International Conference on Applied System Innovation 2018, Chiba, Japan, 13–17 April 2018; pp. 818–821.
14. Nguyen Gia, T.; Sarker, V.K.; Tcarenko, I.; Rahmani, A.M.; Westerlund, T.; Liljeberg, P.; Tenhunen, H. Energy efficient wearable sensor node for IoT-based fall detection systems. *Microprocess. Microsyst.* **2018**, *56*, 34–46. [CrossRef]
15. Pang, I.; Okubo, Y.; Sturnieks, D.; Lord, S.R.; Brodie, M.A. Detection of Near Falls Using Wearable Devices. *J. Geriatr. Phys. Ther.* **2019**, *42*, 48–56. [CrossRef] [PubMed]
16. Sukor, A.S.A.; Zakaria, A.; Rahim, N.A. Activity recognition using accelerometer sensor and machine learning classifiers. In Proceedings of the 2018 IEEE 14th International Colloquium on Signal Processing & Its Applications (CSPA), Batu Feringghi, Malaysia, 9–10 March 2018; pp. 233–238.
17. Zhong, Z.; Chen, F.; Zhai, Q.; Fu, Z.; Ferreira, J.P.; Liu, Y.; Yi, J.; Liu, T. A Real-time Pre-impact Fall Detection and Protection System. In Proceedings of the 2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), Auckland, New Zealand, 9–12 July 2018; pp. 1039–1044.
18. Ali, S.F.; Khan, R.; Mahmood, A.; Hassan, M.T.; Jeon, M. Using Temporal Covariance of Motion and Geometric Features via Boosting for Human Fall Detection. *Sensors* **2018**, *18*, 1918. [CrossRef]
19. Doulamis, A.; Doulamis, N. Adaptive Deep Learning for a Vision-based Fall Detection. In Proceedings of the 11th PErvasive Technologies Related to Assistive Environments Conference on—PETRA'18, Athens, Greece, 16–18 July 2018; ACM Press: New York, NY, USA, 2018; pp. 558–565.
20. Hsieh, Y.Z.; Jeng, Y.L. Development of Home Intelligent Fall Detection IoT System Based on Feedback Optical Flow Convolutional Neural Network. *IEEE Access* **2018**, *6*, 6048–6057. [CrossRef]
21. Kim, S.; Ko, M.; Lee, K.; Kim, M.; Kim, K. 3D fall detection for single camera surveillance systems on the street. In Proceedings of the 2018 IEEE Sensors Applications Symposium (SAS), Seoul, Korea, 12–14 March 2018; pp. 1–6.
22. Lu, N.; Wu, Y.; Feng, L.; Song, J. Deep Learning for Fall Detection: 3D-CNN Combined with LSTM on Video Kinematic Data. *IEEE J. Biomed. Health Inform.* **2018**, *23*, 314–323. [CrossRef] [PubMed]
23. Xu, T.; Zhou, Y. Elders' fall detection based on biomechanical features using depth camera. *Int. J. Wavelets Multiresolut. Inf. Process.* **2018**, *16*, 1840005. [CrossRef]
24. Gibson, R.M.; Amira, A.; Ramzan, N.; Casaseca-de-la-Higuera, P.; Pervez, Z. Multiple comparator classifier framework for accelerometer-based fall detection and diagnostic. *Appl. Soft Comput.* **2016**, *39*, 94–103. [CrossRef]
25. Gibson, R.M.; Amira, A.; Ramzan, N.; Casaseca-de-la Higuera, P.; Pervez, Z. Matching pursuit-based compressive sensing in a wearable biomedical accelerometer fall diagnosis device. *Biomed. Signal Process. Control.* **2017**, *33*, 96–108. [CrossRef]
26. Vavoulas, G.; Pediaditis, M.; Spanakis, E.G.; Tsiknakis, M. The mobifall dataset: An initial evaluation of fall detection algorithms using smartphones. In Proceedings of the 13th IEEE International Conference on BioInformatics and BioEngineering, Chania, Greece, 10–13 November 2013; pp. 1–4.
27. Perc, M. The dynamics of human gait. *Eur. J. Phys.* **2005**, *26*, 525–534. [CrossRef]
28. Bizovska, L.; Svoboda, Z.; Janura, M.; Bisi, M.C.; Vuillerme, N. Local dynamic stability during gait for predicting falls in elderly people: A one-year prospective study. *PLoS ONE* **2018**, *13*, e0197091. [CrossRef]

29. Rivera, D.G.; Merino, A.D.P.; Treviño, M.A.D.V.; Etcheverry, G. Chaotic Analysis on Human Gait Time-Series Signals. *Int. J. Inf. Electron. Eng.* **2016**, *6*, 313.
30. Iqbal, S.; Zang, X.; Zhu, Y.; Saad, H.M.A.A.; Zhao, J. Nonlinear time-series analysis of different human walking gaits. In Proceedings of the 2015 IEEE International Conference on Electro/Information Technology (EIT), Dekalb, IL, USA, 21–23 May 2015; pp. 25–30.
31. Josiński, H.; Michalczuk, A.; Świtoński, A.; Mucha, R.; Wojciechowski, K. Quantifying chaotic behavior in treadmill walking. In *Asian Conference on Intelligent Information and Database Systems*; Springer: Cham, Switzerland, 2015; pp. 317–326.
32. Reynard, F.; Vuadens, P.; Deriaz, O.; Terrier, P. Could Local Dynamic Stability Serve as an Early Predictor of Falls in Patients with Moderate Neurological Gait Disorders? A Reliability and Comparison Study in Healthy Individuals and in Patients with Paresis of the Lower Extremities. *PLoS ONE* **2014**, *9*, e100550. [CrossRef]
33. Morbidoni, C.; Cucchiarelli, A.; Fioretti, S.; Di Nardo, F. A deep learning approach to EMG-based classification of gait phases during level ground walking. *Electronics* **2019**, *8*, 894. [CrossRef]
34. Pairot de Fontenay, B.; Roy, J.; Dubois, B.; Bouyer, L.; Esculier, J. Validating commercial wearable sensors for running gait parameters estimation. *IEEE Sens. J.* **2020**. [CrossRef]
35. Park, S.J.; Hussain, I.; Hong, S.; Kim, D.; Park, H.; Benjamin, H.C.M. Real-time Gait Monitoring System for Consumer Stroke Prediction Service. In Proceedings of the 2020 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 4–6 January 2020; pp. 1–4.
36. Margiotta, N.; Avitabile, G.; Coviello, G. A wearable wireless system for gait analysis for early diagnosis of Alzheimer and Parkinson disease. In Proceedings of the 2016 5th International Conference on Electronic Devices, Systems and Applications (ICEDSA), Ras Al Khaimah, UAE, 6–8 December 2016; pp. 1–4.
37. Nguyen, M.D.; Mun, K.R.; Jung, D.; Han, J.; Park, M.; Kim, J.; Kim, J. IMU-based Spectrogram Approach with Deep Convolutional Neural Networks for Gait Classification. In Proceedings of the 2020 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 4–6 January 2020; pp. 1–6.
38. Schneider, B.; Banerjee, T.; Grover, F.; Riley, M. Comparison of gait speeds from wearable camera and accelerometer in structured and semi-structured environments. *Healthc. Technol. Lett.* **2020**, *7*, 25–28. [CrossRef] [PubMed]
39. Coviello, G.; Avitabile, G. Multiple Synchronized Inertial Measurement Unit Sensor Boards Platform for Activity Monitoring. *IEEE Sens. J.* **2020**. [CrossRef]
40. Sahoo, S.; Saboo, M.; Pratihar, D.K.; Mukhopadhyay, S. Real-Time Detection of Actual and Early Gait Events During Level-Ground and Ramp Walking. *IEEE Sens. J.* **2020**. [CrossRef]
41. Sekine, M.; Tamura, T.; Akay, M.; Fujimoto, T.; Togawa, T.; Fukui, Y. Discrimination of walking patterns using wavelet-based fractal analysis. *IEEE Trans. Neural Syst. Rehabil. Eng.* **2002**, *10*, 188–196. [CrossRef]
42. Terrier, P.; Dériaz, O. Kinematic variability, fractal dynamics and local dynamic stability of treadmill walking. *J. Neuroeng. Rehabil.* **2011**, *8*, 12. [CrossRef]
43. Hausdorff, J.M. Gait dynamics, fractals and falls: finding meaning in the stride-to-stride fluctuations of human walking. *Hum. Mov. Sci.* **2007**, *26*, 555–589. [CrossRef]
44. Koutsiana, E.; Hadjileontiadis, L.J.; Chouvarda, I.; Khandoker, A.H. Fetal Heart Sounds Detection Using Wavelet Transform and Fractal Dimension. *Front. Bioeng. Biotechnol.* **2017**, *5*, 1–9. [CrossRef] [PubMed]
45. Zhang, Y.D.; Chen, X.Q.; Zhan, T.M.; Jiao, Z.Q.; Sun, Y.; Chen, Z.M.; Yao, Y.; Fang, L.T.; Lv, Y.D.; Wang, S.H. Fractal Dimension Estimation for Developing Pathological Brain Detection System Based on Minkowski-Bouligand Method. *IEEE Access* **2016**, *4*, 5937–5947. [CrossRef]
46. Senouci, B.; Charfi, I.; Heyrman, B.; Dubois, J.; Miteran, J. Fast prototyping of a SoC-based smart-camera: A real-time fall detection case study. *J. Real Time Image Process.* **2016**, *12*, 649–662. [CrossRef]
47. Ali, A.A.S.; Siupik, M.; Amira, A.; Bensaali, F.; Casaseca-de-la Higuera, P. HLS based hardware acceleration on the zynq SoC: A case study for fall detection system. In Proceedings of the 2014 IEEE/ACS 11th International Conference on Computer Systems and Applications (AICCSA), Doha, Qatar, 10–13 November 2014; pp. 685–690. [CrossRef]
48. Ong, P.S.; Chang, Y.C.; Ooi, C.P.; Karuppiah, E.K.; Tahir, S.M. An FPGA Implementation of Intelligent Visual Based Fall Detection. *Int. J. Comput. Electr. Autom. Control. Inf. Eng.* **2013**, *7*, 184–189.

49. Ong, P.S.; Ooi, C.P.; Chang, Y.C.; Karuppiah, E.K.; Tahir, S.M. An FPGA-based hardware implementation of visual based fall detection. In Proceedings of the 2014 IEEE Region 10 Symposium, Kuala Lumpur, Malaysia, 14–16 April 2014; pp. 397–402.
50. Abdelhedi, S.; Baklouti, M.; Bourguiba, R.; Mouine, J. Design and implementation of a fall detection system on a Zynq board. In Proceedings of the IEEE/ACS International Conference on Computer Systems and Applications, Agadir, Morocco, 29 November–2 December 2017; pp. 1–7.
51. Abdelhedi, S.; Baklouti, M.; Bourguiba, R.; Mouine, J. Vivado HLS-based implementation of a fall detection decision core on an FPGA platform. In Proceedings of the 2016 11th International Design and Test Workshop, Hammamet, Tunisia, 18–20 December 2017; pp. 115–120.
52. x-IMU. Available online: https://x-io.co.uk/x-imu/ (accessed on 11 April 2020).
53. Analog Devices. *Digital Accelerometer ADXL345-EP*; Rev. B.; Analog Devices: Norwood, MA, USA, 2013.
54. Dickey, D.A.; Fuller, W.A. Distribution of the estimators for autoregressive time series with a unit root. *J. Am. Stat. Assoc.* **1979**, *74*, 427–431.
55. Schwert, G.W. Tests for unit roots: A Monte Carlo investigation. *J. Bus. Econ. Stat.* **2002**, *20*, 5–17. [CrossRef]
56. Ng, S.; Perron, P. Unit root tests in ARMA models with data-dependent methods for the selection of the truncation lag. *J. Am. Stat. Assoc.* **1995**, *90*, 268–281. [CrossRef]
57. Akaike, H. Information theory and an extension of the maximum likelihood principle. In *Selected Papers of Hirotugu Akaike*; Springer: New York, NY, USA, 1998; pp. 199–213.
58. Kwiatkowski, D.; Phillips, P.C.; Schmidt, P.; Shin, Y. Testing the null hypothesis of stationarity against the alternative of a unit root. *J. Econom.* **1992**, *54*, 159–178. [CrossRef]
59. Box, G.E.P.; Jenkins, G.M. *Time Series Analysis: Forecasting and Control*; Holden-Day: San Francisco, CA, USA,1976; p. 575.
60. Nguyen, T.L.; Le, T.A.; Pham, C. The internet-of-things based fall detection using fusion feature. In Proceedings of the 2018 10th International Conference on Knowledge and Systems Engineering (KSE), Ho Chi Minh City, Vietnam, 1–3 November 2018; pp. 129–134.
61. Liu, K.C.; Hsieh, C.Y.; Hsu, S.J.P.; Chan, C.T. Impact of sampling rate on wearable-based fall detection systems based on machine learning models. *IEEE Sens. J.* **2018**, *18*, 9882–9890. [CrossRef]
62. Chelli, A.; Pätzold, M. A machine learning approach for fall detection and daily living activity recognition. *IEEE Access* **2019**, *7*, 38670–38687. [CrossRef]
63. Hussain, F.; Hussain, F.; Ehatisham-ul Haq, M.; Azam, M.A. Activity-aware fall detection and recognition based on wearable sensors. *IEEE Sens. J.* **2019**, *19*, 4528–4536. [CrossRef]
64. Tahir, A.; Ahmad, J.; Morison, G.; Larijani, H.; Gibson, R.M.; Skelton, D.A. Hrnn4f: Hybrid deep random neural network for multi-channel fall activity detection. *Probab. Eng. Inf. Sci.* **2019**, 1–14. [CrossRef]
65. Sucerquia, A.; López, J.D.; Vargas-Bonilla, J.F. SisFall: A fall and movement dataset. *Sensors* **2017**, *17*, 198. [CrossRef]
66. Saadeh, W.; Altaf, M.A.B.; Altaf, M.S.B. A high accuracy and low latency patient-specific wearable fall detection system. In Proceedings of the 2017 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI), Orlando, FL, USA, 16–19 February 2017; pp. 441–444.
67. Saadeh, W.; Butt, S.A.; Altaf, M.A.B. A patient-specific single sensor IoT-based wearable fall prediction and detection system. *IEEE Trans. Neural Syst. Rehabil. Eng.* **2019**, *27*, 995–1003. [CrossRef] [PubMed]
68. Abunahia, D.G.; Ismail, T.A.; Al Ola, H.R.A.; Amira, A.; Ali, A.A.S.; Bensaali, F. A Reconfigurable Connected Health Platform Using ZYNQ System on Chip. In Proceedings of the SAI Intelligent Systems Conference, London, UK, 21–22 September 2016; pp. 857–867.