Efficient implementation of LMS adaptive filter-based FECG extraction on an FPGA

Bhavya Vasudeva ✉, *Puneesh Deora, Pradhan Mohan Pradhan, Sudeb Dasgupta*

Department of Electronics and Communication Engineering, Indian Institute of Technology Roorkee, Uttarakhand, India
✉ *E-mail: bvasudeva@ec.iitr.ac.in*

Published in Healthcare Technology Letters; Received on 19th February 2020; Revised on 28th May 2020; Accepted on 4th June 2020

In this Letter, the field programmable gate array (FPGA) implementation of a foetal heart rate (FHR) monitoring system is presented. The system comprises a preprocessing unit to remove various types of noise, followed by a foetal electrocardiogram (FECG) extraction unit and an FHR detection unit. To improve the precision and accuracy of the arithmetic operations, a floating-point unit is developed. A least mean squares algorithm-based adaptive filter (LMS-AF) is used for FECG extraction. Two different architectures, namely series and parallel, are proposed for the LMS-AF, with the series architecture targeting lower utilisation of hardware resources, and the parallel architecture enabling less convergence time and lower power consumption. The results show that it effectively detects the R peaks in the extracted FECG with a sensitivity of 95.74–100% and a specificity of 100%. The parallel architecture shows up to an 85.88% reduction in the convergence time for non-invasive FECG databases while the series architecture shows a 27.41% reduction in the number of flip flops used when compared with the existing FPGA implementations of various FECG extraction methods. It also shows an increase of 2–7.51% in accuracy when compared to previous works.

1. Introduction: Over the past few decades, analysis of foetal electrocardiogram (FECG) has proven to be a tool of great importance when it comes to monitoring the well-being of the foetus during pregnancy and labour, unearthing vital information such as foetal heart rate (FHR), heart rate variability etc. Any abnormalities in these parameters indicate that the foetus is in distress, possibly due to asphyxia, which is a major cause of neonatal deaths. Regular FHR monitoring can enable a clinician to intervene in due time to prevent such cases.

Various methods [1] used to obtain FHR are auscultation (Doppler ultrasound, fetoscope), foetal phonocardiography, foetal magnetocardiography, and other invasive methods where electrodes have direct contact with foetal skin. These methods are not suitable for mobile, regular, low-cost, real-time monitoring of the foetus. An alternative method to obtain FHR is to calculate it from FECG, which can be extracted from the non-invasive abdominal electrocardiogram (ECG) recordings acquired from a pregnant subject. This ECG signal contains FECG contaminated with maternal ECG (MECG), power line interference, motion artefacts etc. Various techniques [1] involving statistical and time-domain analysis have been exploited to extract the FECG. Adaptive filtering [2], non-linear decomposition [3], blind source separation [4] (independent component analysis (ICA)-based methods), wavelet transform (WT)-based techniques [5–7], neural network-based approaches [8] are widely used.

Although the methods based on ICA perform better than those which use single-channel recordings, they require the acquisition of multi-channel abdominal signals which may be uncomfortable for the subject. Such methods also require visual inspection of the signals, and an appropriate number of data segments have to be selected manually for a representative template [1]. The real-time implementation of such methods is not suitable unless block delayed analysis is considered [9]. Among the methods using single-channel recordings, Kalman filter-based approaches offer high performance, but their implementation complexity is quite high and they require the R peaks of the signal to have a consistent morphological shape [9]. WT-based methods can also prove to be computationally intensive, can have drawbacks such as increased hardware area, complex routing etc., and require the selection of an appropriate wavelet, to achieve the required performance level. As foetal signals and other interferences are not always linearly separable, non-linear decomposition-based methods can be used for extracting FECG in such cases. However, such methods require some prior information about the desired and undesired parts of the signal and have a high-computational complexity [9]. As the adaptive filter is an accurate method for FECG extraction and its computational complexity is relatively low [1], a least mean squares algorithm-based adaptive filter (LMS-AF) is chosen for this study.

The signal strength of the FECG is low as compared to the MECG [9]. In rural areas, where cellular connectivity is low, the transmission of these signals for processing in the cloud is not suitable. The FPGA can eliminate the need for an extra standby computing device that would be required for computational purposes. Also, it is a better prototyping platform for hardware implementation compared to traditional digital signal processors (DSPs). The FPGA implementation can also serve as a step towards the development of a low-cost FHR monitoring system as a system on chip.

Previous hardware implementations of LMS-AF focusing on FECG extraction are discussed below. The work presented by Hatai *et al.* [10] was tested on synthetic data, with up to 88% accuracy. As synthetic data has significantly better morphology of the PQRST complex and lower noise than real signals, no pre-processing was performed. Dynamic thresholding was used for peak detection. Morales *et al.* [11] used a field-programmable analogue array for analogue signal preprocessing and an FPGA for FECG extraction with accuracy in the range of 87–93%. LabVIEW FPGA module was used to generate the hardware design. Arias-Ortega *et al.* [12] implemented the LMS-AF on a digital signal controller, used a low-noise analogue front end to achieve 93.1% accuracy and 87.1% sensitivity. In [13], the authors implemented an LMS-AF-based FECG extraction system on an FPGA. However, FHR calculation is done manually and an algorithm to automate this process is not presented. Some other methods for FECG extraction [14–18] have also been implemented on hardware. Some of the aforementioned works [11, 15, 16] reportedly use fixed-point arithmetic, which leads to lower precision than floating-point (FP) arithmetic. Furthermore, FP addition utilises much fewer resources than the logarithmic number system (LNS). As the main advantage of the LNS is its efficient division operation, which is not required in LMS-AF, we have opted for LMS with FP operations. It has also been reported that FP operations are difficult to implement on FPGA as the

algorithm is very complex and leads to excessive consumption of logic elements [19].

The main contributions of this Letter are as follows:

- For the foetal R peak detection, a norm for the determination of the threshold is proposed to avoid the detection of false positives.
- A FP unit (FPU) is developed for the FPGA implementation to support FP calculations, and hence improve the precision and accuracy of the system. Although Xilinx has a core for FP operations, it is not an open-source internet protocol, and thus cannot be used for application-specific integrated circuit designing, which is why the FPU is developed. The system is tested on both real and synthetic ECG signals, and the results validate the robustness of the system.
- For the implementation of the LMS-AF module, two different architectures, namely series and parallel, are proposed. While the former is developed for lower hardware utilisation, the latter is better in terms of lower latency and power consumption. FPGA implementation and simulation results validate the same.

2. Methodology

2.1. Preprocessing: The information that needs to be extracted from the thoracic and abdominal signals is masked by various types of noise [9], such as power line interference at 50 Hz, low-frequency baseline wander, broadband muscle noise, motion artefact etc. To retain the MECG and FECG components [9] and attenuate the sources of noise, the signals are preprocessed.

To remove the high frequencies, a fourth-order low-pass Butterworth filter is used. The cut-off of the filter is kept at 45 Hz so that the ECG components in the signal are retained [9]. For the low-pass filter, the Bessel, Butterworth, Chebyshev, and RC filters [20] were considered. Among these, the Bessel filter offers a slower transition from pass-band to stop-band as compared to the other filters of the same order. The Chebyshev filters have ripple in the pass-band, while Butterworth and Bessel filters do not. Moreover, Butterworth filters have a significantly better frequency response (flat in the pass-band) than a simple RC filter of the same order. Therefore, a Butterworth filter is used in this work. As the cut-off of the filter is not sharp, the frequencies above 35 Hz and below 55 Hz lie in the transition band of the filter.

The peak at 50 Hz due to the power line interference is not sufficiently attenuated by the low-pass filter. Therefore, a notch filter [20] centred at 50 Hz (quality factor 25) is used.

Another source of noise is the baseline wander, which is a low-frequency noise is resulting from the respiration or movement of the subject or electrodes during recording. Since only the components corresponding to the baseline wander need to be removed, a high-pass filter is not used for this application. Three techniques, namely polynomial fitting using polynomial regression, two-stage median filtering, and two-stage moving average filtering [21], are considered to obtain an approximation of the baseline wander present in the signal. The complexity of these methods is $O(mN^2)$, $O(Nn\log(n))$, and $O(N)$, respectively, where N is the total number of samples, m is the order of the polynomial, and n is the window size. A two-stage moving average filter is used in this work as it is the most efficient of all and gives a smooth approximation of the baseline wander. The operations performed are summarised in the following equations:

$$M_1[n] = \frac{1}{N_1}\sum_{i=0}^{N_1-1} x[n+i-N_1+1] \tag{1}$$

$$M_2[n] = \frac{1}{N_2}\sum_{j=0}^{N_2-1} M_1[n+j-N_2+1] \tag{2}$$

where x is the input signal, M_1 is the first stage mean with window size N_1, M_2 is the second stage mean with window size N_2, and n is the sample index. N_1 and N_2 are kept as 200 in this work as a larger value yields an extremely smooth estimate, due to which some of the changes in the baseline are not captured properly, whereas a smaller value yields an erratic estimate. To remove the baseline wander, the output of the two-stage moving average filter is subsequently subtracted from the input signal.

2.2. LMS algorithm: To separate the FECG from the preprocessed thoracic and abdominal ECG signals, LMS-AF [22] is used. Let $\boldsymbol{x}[n] = [x[n], x[n-1], \ldots, x[n-m+1]]^{\mathrm{T}}$, represent the input to the filter, where $x[n]$ is the sample value at instant n, m is the order of the filter, and $(.)^{\mathrm{T}}$ denotes the transpose operator. $\boldsymbol{w}[n] = [w_{m-1}[n], w_{m-2}[n], \ldots, w_0[n]]$, is the weight vector, where $w_{m-k}[n]$ is the k^{th} weight at sample instant n. The output of the filter at the n^{th} sampling instant is given by (3). The error signal is calculated using (4), where $d[n]$ is the desired signal. The weight updation is carried out using (5), where μ is the step size, ∇ is the gradient operator, and $k = 0, 1, \ldots, m-1$.

$$y[n] = \boldsymbol{x}^{\mathrm{T}}[n]\boldsymbol{w}[n] \tag{3}$$

$$e[n] = d[n] - y[n] \tag{4}$$

$$\begin{aligned} w_k[n+1] &= w_k[n] - \mu\nabla(e^2[n]) \\ &= w_k[n] + 2\mu e[n]x[n-m+k+1] \end{aligned} \tag{5}$$

For this work, the thoracic signal is considered as the desired signal $d[n]$, and the abdominal signal is the input $\boldsymbol{x}[n]$. The criteria for convergence of the filter weights are satisfied in around 12,000 samples. m and μ are set to 19 and 7×10^{-5}, respectively.

The MECG components in thoracic and abdominal signals are not exactly the same [1], which leads to some residual maternal R peaks in the resulting error signal $e[n]$ [22]. After the convergence of weights of the filter, the FECG is enhanced and the MECG is attenuated in $e[n]$ (the output of LMS-AF).

2.3. FHR detection: The Pan and Tompkins algorithm (PTA) [23] is well-known for detecting R peaks in the ECG signals of a single subject. A modified version of PTA is used to detect the foetal R peaks from a mixture of foetal R peaks and residual maternal R peaks. The output of the LMS-AF is differentiated, squared, and then passed through a mean filter of length 40. Since the extracted FECG contains residual maternal R peaks as well as sharper foetal R peaks, these operations, which are also a part of PTA, enhance the foetal R peaks. Hence, the resultant signal sdm has higher amplitude for foetal R peaks as compared to the maternal R peaks. To determine the threshold value *th* which can be used to distinguish between the foetal and maternal R peaks, a new norm is proposed. Unlike PTA where adaptive thresholding is used, the proposed method uses a single value of '*th*' for a data set. m_1 denotes the mean of the signal '*sdm*'. The procedure for calculation of '*th*', which is repeated N times, is summarised below:

(1) If ($in > m_1$ and $in > R_1$) then $R_3 = in$, $R_4 = R_2$
(2) else if ($in < m_1$) then $pv = R_3$, $pl = R_4$, $m_2 = m_2 + \frac{pv}{N}$
(3) end if
(4) $R_1 = in$
(5) $R_2 = R_2 + 1$
(6) If ($R_2 = N - 1$) then $th = (m_1 + m_2)/2$
(7) end if

Here, *in* denotes the current input, N is the number of inputs, $R_1(R_2)$ is used to store the input value (location) for the next cycle, and $R_3(R_4)$ is used to conditionally store the input value (location). The locations and values of local maxima are denoted by *pl* and *pv*, respectively. m_1, m_2, R_1, and R_2 are initialised to zero.

The detection of foetal R peaks is based on two criteria: their amplitude should be above '*th*', and if two local maxima occur within 200 samples of each other, the one with the larger amplitude denotes the foetal R peak. The maximum FHR can be 200 beats per minute (bpm) [24] (300 samples at 1 kHz sampling frequency). Therefore, maxima separated by at least 200 samples (300 bpm) are considered for the detection of foetal R peaks. The procedure for the above is listed below:

(1) if ($pv > th$) then
(2) if ($pl - R_1 > 200$) then $out = R_1$, $R_1 = pl$, $R_2 = pv$
(3) else
(4) if ($pv > R_2$) then $out = pl$, $R_1 = pl$, $R_2 = pv$
(5) else $out = R_1$
(6) end if
(7) end if
(8) end if

Here, out denotes the locations of the foetal R peaks detected. R_1 and R_2 are initialised with *pl* and *pv*, respectively.

The difference between the consecutive R peaks, as detected in the previous stage, is the RR interval. As the weights of the LMS-AF converge around 12,000 samples, only the RR intervals for peaks occurring after 12,000 samples are considered. The average of these RR intervals is taken and divided by the sampling frequency to get the average RR interval length in seconds. The FHR is calculated as follows:

$$\text{FHR(bpm)} = \frac{60}{\text{RR interval length (s)}} \tag{6}$$

3. Implementation of FPGA: For the FPGA implementation, the proposed system is divided into four units as shown in Fig. 1.

3.1. FP unit: Since the input values to the system are FP numbers, logic cannot be defined directly on such numbers in Verilog. Therefore, an FPU is developed for performing basic arithmetic operations (addition, subtraction, and multiplication) and comparison. The FP numbers are converted to their 32-bit binary representation as per the IEEE 754 standard [25], in which the first bit is the sign bit *s*, followed by 8 bits for exponent *e*, and 23 bits for fraction *f*. The value of the number is given by $(-1)^s 1 \cdot f 2^{(e-\text{bias})}$, where $1 \cdot f$ is the mantissa *m* and *bias* is $2^7 - 1$. The inputs to the FPU module include a 2-bit sequence to select one of the four available operations and two 32-bit FP numbers (*A* and *B*). When the numbers enter the module, they are split into three parts, i.e. sign, exponent and mantissa denoted by s_a, e_a, m_a and s_b, e_b, m_b for *A* and *B*, respectively. *m* is stored in the form of 24 bits with 1 concatenated to 23 bits of *f*. s_{out}, e_{out}, and m_{out} denote the sign, exponent and mantissa of the output.

The procedure followed for the FP adder is listed below:

(1) if ($e_a = e_b$) then $e_{\text{out}} = e_a$
(2) else if ($e_a > e_b$) then $e_{\text{out}} = e_a$, $d = e_a - e_b$, $m_b = m_b \gg d$
(3) else $e_{\text{out}} = e_b$, $d = e_b - e_a$, $m_a = m_a \gg d$
(4) end if
(5) if ($s_a = s_b$) then $m_{\text{out}} = m_a + m_b$, $s_{\text{out}} = s_a$
(6) else
(7) if ($m_a > m_b$) then $s_{\text{out}} = s_a$, $m_{\text{out}} = m_a - m_b$
(8) else $s_{\text{out}} = s_b$, $m_{\text{out}} = m_b - m_a$
(9) end if
(10) end if

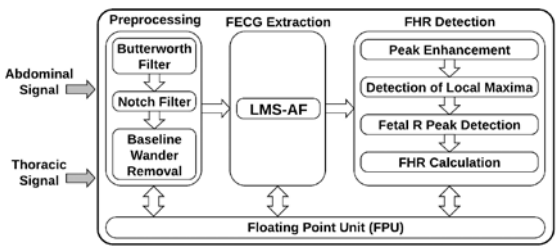

Fig. 1 *Block diagram of the FPGA implementation of the system*

Here, $\gg$ denotes the right shift operation. A similar procedure is followed for the FP subtractor, except that when the sign bits are same, subtraction is performed after comparing the mantissas and when they are opposite, addition is performed. The outputs of the FP multiplier are given by $s_{\text{out}} = s_a \oplus s_b$, $e_{\text{out}} = e_a + e_b - bias$, and $m_{\text{out}} = m_a \times m_b$, where $\oplus$ denotes the bit-wise XOR operation. For all the three operations, the next stage is to normalise the output. When m_{out} is not of the form $1 \cdot f_{\text{out}}$, a repetitive process of shifting m_{out} left by one place and subtracting 1 from e_{out} is followed till the first bit of m_{out} becomes 1. s_{out}, e_{out}, and f_{out} are concatenated to get the 32-bit output of the operation.

For FP comparison, let c_{out} denote a 2-bit sequence to denote the three cases, i.e. $A > B$ ($c_{\text{out}} = 01$), $A = B$ ($c_{\text{out}} = 00$), and $A < B$ ($c_{\text{out}} = 10$). The procedure is listed below:

(1) If ($s_a > s_b$) then $c_{\text{out}} = [10]$
(2) else if ($s_b > s_a$) then $c_{\text{out}} = [01]$
(3) else
(4) if ($e_a > e_b$) then $c_{\text{out}} = [01]$
(5) else if ($e_b > e_a$) then $c_{\text{out}} = [10]$
(6) else
(7) if ($f_a > f_b$) then $c_{\text{out}} = [01]$
(8) else if ($f_b > f_a$) then $c_{\text{out}} = [10]$
(9) else $c_{\text{out}} = [00]$
(10) end if
(11) end if
(12) end if

As the output should be a 32-bit number for uniformity, 30 0*s* are concatenated to c_{out} to get the output.

3.2. Preprocessing: The following three modules are implemented
3.2.1 Butterworth filter: In this module, one input value is used in every clock cycle to get the output as follows [20]:

$$O[k] = \alpha I[k] + \beta O[k-1] + \gamma O[k-2] + \delta O[k-3] + \epsilon O[k-4]$$

where $I[k]$ is the sample value at instant k and $O[k]$ is the output value. The values of the constants are obtained from the transfer function of the filter, as per the cut-off and order of the filter. In this work, $\alpha = 0.00308$, $\beta = 3.28391$, $\gamma = -4.08689$, $\delta = 2.28117$, and $\epsilon = -0.48140$.

3.2.2 Notch filter: This module works similarly to the previous module, following the equation [20]:

$$O[k] = \alpha I[k] + \beta I[k-1] + \gamma I[k-2] + \delta O[k-1] + \epsilon O[k-2]$$

In this case, $\alpha = 0.99405$, $\beta = -1.31278$, $\gamma = 0.99405$, $\delta = 1.31272$, and $\epsilon = -0.98804$.

3.2.3 Baseline wander removal: Fig. 2 shows the structure of the two-stage moving average filter. As in (1), the first stage mean M_1 is the average of N_1 values. In every clock cycle, the input is added to M_1 and $x[N_1 - 1]$ is subtracted from M_1, both after getting multiplied by $(1/N_1)$. For the moving average operation, all the values in memory 1 are shifted by one position, so that

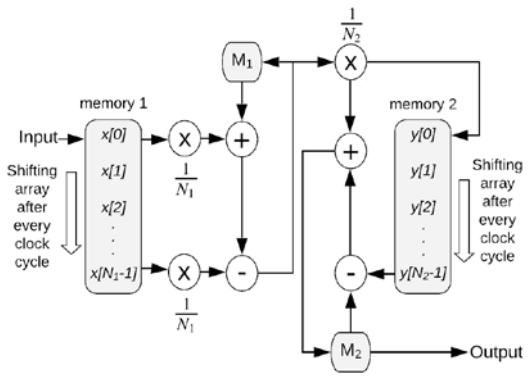

Fig. 2 *Structure of the two-stage moving average filter*

$x[N_1 - 1]$ is discarded and a new value is stored in $x[0]$. A similar procedure is followed for calculating the second stage mean M_2 as per (2). M_1 is multiplied by $(1/N_1)$, stored in memory 2 and also added to M_2. The last value of memory 2 can then be directly subtracted from M_2 to obtain the second stage mean. The values of M_2 represent the baseline wander approximation. The output of the two-stage moving average filter is used to remove the baseline wander from the input by performing one subtraction operation in every clock cycle. The latency of each of these modules is 1 clock cycle.

3.3. FECG extraction: This stage comprises the LMS-AF, for which two different architectures are proposed. Figs. 3*a* and *b* illustrate the proposed series and parallel architectures of the LMS-AF module, respectively. In both the figures, $\beta = 2\mu$. As the magnitude of abdominal and thoracic signals may not be of the same order, they have to be scaled appropriately (denoted by scaling in the figures) before being used in the algorithm.

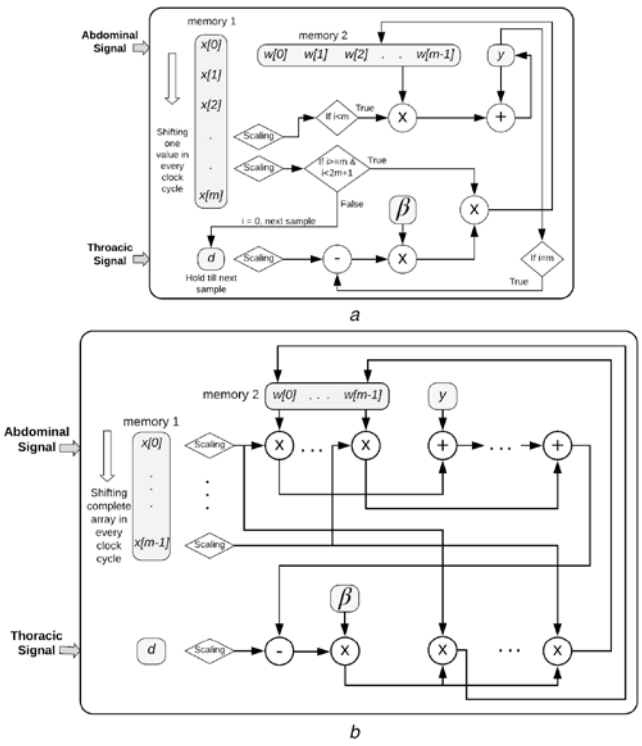

Fig. 3 *Illustration of proposed series and parallel architecture of LMS-AF*
a Series architecture
b Parallel architecture

3.3.1 Series architecture: In Fig. 3*a*, memory 1 stores the vector $\boldsymbol{x}^{\mathrm{T}}[n]$ and an extra element, and memory 2 contains the weights of the filter. In every clock cycle, one element of $x^{\mathrm{T}}[n]$ is scaled, multiplied by one element of $\boldsymbol{w}[n]$ and added to $y[n]$. The same element of $\boldsymbol{x}^{\mathrm{T}}[n]$ is also copied to the immediate next position in $\boldsymbol{x}^{\mathrm{T}}[n]$. Thus, after m clock cycles, $y[n]$ has been obtained as in (3), and $\boldsymbol{x}^{\mathrm{T}}[n]$ has shifted by one index. In the following clock cycle, error is calculated using (4), and the updated value of the first weight of the filter is also obtained. This updated weight value is stored in its position in the next clock cycle. This sequential process is repeated until all the weights are updated, which corresponds to $m + 1$ clock cycles. After a total of $2m + 1$ clock cycles, a new input value is stored in $x[0]$ so that $\boldsymbol{x}^{\mathrm{T}}[n]$ is updated. The register containing $d[n]$ also gets updated. As the required output for a particular pair of $\boldsymbol{x}^{\mathrm{T}}[n]$ and $d[n]$ is obtained after $2m + 1$ clock cycles, the latency of this module is $2m + 1$ clock cycles.

3.3.2 Parallel architecture: In Fig. 3*b*, the memory 1 (vector $\boldsymbol{x}^{\mathrm{T}}[n]$) gets updated with the next input value in every clock cycle. Each element of $\boldsymbol{x}^{\mathrm{T}}[n]$ is scaled, and then multiplied with the elements from the memory 2 (vector $\boldsymbol{w}[n]$). These are added to obtain $y[n]$, as in (3). The register containing $d[n]$ is updated in every clock cycle and is used to calculate the error, using (4). Since $2\mu e[n]$ is used in every weight updation, it is calculated first, and subsequently multiplied with the values from memory 1 to update the weights, using (5). The updated weights are stored in memory 2. Thus, all the operations involving the error calculation and weight updation are performed in a single clock cycle and the latency of this module is 1 clock cycle.

3.4. FHR detection: The four modules of this stage are
3.4.1 Peak enhancement: In this module, the input is differentiated, squared, and passed through a mean filter of length P. The mean m_1 of *sdm*, which is required in the next module, is also determined in this module. The operations executed in every clock cycle are summarised below:

1. $pval = cval$
2. $cval = input$
3. $sdiff = (cval - pval) \times (cval - pval)$
4. $M[0] = sdiff \times \dfrac{1}{P}$
5. $sdm = sdm + M[0] - M[P-1]$
6. $m_1 = m_1 + sdm \times \dfrac{1}{N}$
7. Shift the elements of M by one position

Here, *cval* and *pval* denote the current and previous input values, respectively. *sdiff* denotes the differentiated and squared signal, which is stored in the memory M (size P) after multiplication by $1/P$. N denotes the number of input samples.

3.4.2 Detection of local maxima: In this module, the local maxima are determined, using m_1 as the threshold. The operations executed in every clock cycle are listed in Section 2.3.

3.4.3 Foetal R peak detection: The operations executed in every clock cycle for this module are also listed in Section 2.3.

3.4.4 FHR calculation: The RR intervals are estimated using the differences between consecutive peak locations (*out*). Two registers are used for storing the current input and the previous input. The estimated RR intervals are accumulated and averaged out, after which FHR is obtained using (6).

4. Results and discussion: To test the system for real signals, the non-invasive FECG (NiFECG) database [26] and database for

doi: 10.1049/htl.2020.0016

identification of systems (DaISy) [27] are used. In the NiFECG database, the signals have been sampled at 1 kHz with 16-bit resolution. From the data set to be tested, one thoracic and one abdominal signal are chosen as inputs to the system. These are shown in Figs. 4*a* and *b*, respectively. DaISy consists of eight channels, 10 s recordings sampled at 250 Hz, where three channels are thoracic signals and the rest are abdominal signals. The synthetic signals were simulated using the FECGSYN toolbox [28] in MATLAB, at a sampling rate of 1 kHz. Figs. 4*c* and *d* show the thoracic and abdominal input signals. It is observed that the synthetic signals are less noisy than the real signals. In the thoracic signals, all the peaks are maternal R peaks. In the abdominal signals, the higher peaks are maternal, and those annotated as *fpk* are foetal R peaks.

Figs. 5*a* and *d* show the time series for real and synthetic signals after preprocessing. The output of the LMS-AF stage, where FECG is enhanced and MECG is attenuated, is shown in Figs. 5*b* and *e*. The signal obtained after peak enhancement (labelled *sdm*) and the detected foetal R peaks (denoted by *fpk*), are shown in Figs. 5*c* and *f*. Table 1 lists the results obtained for sensitivity, specificity, accuracy, and FHR for the tested datasets. It is observed that the proposed norm for the determination of the threshold, which is used for foetal R peak detection, results in no false positives. The application of PTA [23] on ecga444 [26] data set gives a sensitivity of 78.72%, specificity of 48.28%, and accuracy of 67.11%, as this method was developed for detecting R peaks in ECG signals of a single subject, whereas the output of LMS-AF has enhanced FECG as well as residual MECG.

In Table 2, the performance of the proposed work is compared with various FECG extraction methods. The proposed work shows an increase of 1.34% in the sensitivity and 2% in the accuracy for DaISy. The proposed method also shows an increase of 1.02% in the sensitivity and 7.51% in accuracy when compared to works that have tested their systems on both NiFECG and DaISy.

The system is implemented on the Xilinx Artix-7 FPGA (XC7A100TCSG324-1), equipped with 63,400 look-up tables (LUTs), 126,800 flip flops (FFs), 240 DSPs, and 210 input/output ports. Except for the baseline wander removal, the detection of local maxima, and the LMS-AF module, all the modules have minimal resource (~0 LUTs and FFs) and power utilisation (0.068 W). The baseline wander removal module consumes 2.691 W power, and utilises 820 LUTs and 94 FFs. The power per cycle is 89.683 μW. The detection of the local maxima module consumes 0.167 W power, and utilises 45 LUTs and 34 FFs. The power per cycle is 9.278 μW.

For the LMS-AF module, the resource utilisation and power consumption depend on the architecture and the order of the filter. There is a trade-off between the series and parallel architectures in terms of convergence time and resource utilisation. For the parallel design, the number of operations in every clock cycle is more than the series design, and hence the resource utilisation is greater. On the other hand, the series architecture distributes

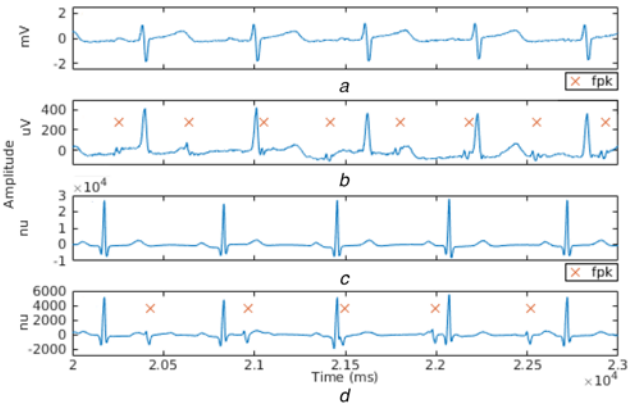

Fig. 4 *Waveforms representing various ECG signals*
a Real thoracic signal
b Real abdominal signal
c Synthetic thoracic signal
d Synthetic abdominal signal. Synthetic data set has no units (nu)

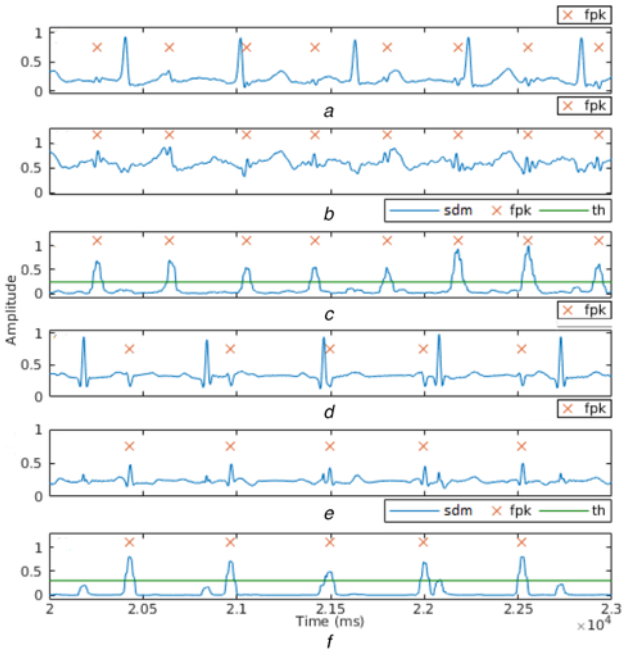

Fig. 5 *Results of*
a Preprocessing
b LMS-AF
c Peak detection for real signals
d Preprocessing
e LMS-AF
f Peak detection for synthetic signals. All values are normalised between 0 and 1

Table 1 Results obtained for different datasets using the proposed approach

Dataset	FHR, bpm	Sensitivity, %	Specificity, %	Accuracy, %
ecgca444 [26]	152	95.74	100	97.37
ecgca840 [26]	161	96	100	97.37
ecgca746 [26]	147	97.78	100	98.53
ecgca771 [26]	153	100	100	100
DaISy Channel 2 [27]	143	100	100	100
DaISy Channel 3 [27]	143	100	100	100
synthetic [28]	115	100	100	100

Table 2 Comparison of performance of the proposed method with various FECG extraction methods

Method	Dataset	Sensitivity, %	Accuracy, %
Le *et al.* [29]	DaISy	98.68%	98.04
Gini *et al.* [30]	DaISy	91%	87.30
Lima-Herrera *et al.* [31]	DaISy and NiFECG	97.50%	92.10
Morales *et al.* [11]	DaISy and NiFECG	—	89
proposed method	DaISy	100%	100
proposed method	DaISy and NiFECG	98.5%	99.04

–, Not reported.

Table 3 Comparison of hardware implementations of the proposed method and various FECG extraction methods

Method	Device	Convergence time, ms	Consumption of power, W	LUTs	FFs
LMS [10]	XC6SLX45-3-CSG394	—	—	1042	440
LMS [11]	Spartan3E XC3S500E	600	—	—	—
LMS [12]	dsPIC30F6014A	0.33	1.67[a]	—	—
OL-JADE [14]	OMAP L137	948	—	—	—
infomax [15]	Stratix-V 5SGXEA7N2F45C2	3.4-54	0.55	—	—
neural network [16]	Stratix-II EP2S15F484C3	—	—	9726	4324
BSS [17]	Spartan-3	—	—	3002	405
proposed series	Artix-7	18.72	6.478	2368	294
proposed parallel	XC7A100TCSG324-1	0.48	1.954	22 407	640

–, Not reported.
[a]The system proposed by Ortega *et al.* [12] consumes 1 W, for the current absorption of 200 mA and supply of 5 V, at 30 MHz operating frequency.

the same number of operations across more clock cycles, and hence needs more time for convergence, and consumes more power. Also, an increase in the filter order results in an increase in the number of operations as well as resource utilisation.

In Table 3, the existing implementations of various FECG extraction methods on different hardware platforms are compared with the proposed architectures of LMS-AF after mapping the power consumption and convergence time to 50 MHz operating frequency. As the number of cycles is different for the two architectures, there is a large difference in the values of power consumption. The power per cycle is 7.823 μW for series and 65.133 μW for parallel architecture (30,000 input samples). The series architecture uses nine instances of the FPU module, and shows 27.41% reduction in the number of FFs, whereas the number of LUTs is comparable to the other methods. The parallel architecture uses 98 instances of the FPU module, and shows up to 85.88% reduction in the convergence time when compared with the methods [11, 15, 17] using NiFECG database. The convergence time for series is 39 times ($m = 19$) more than that of parallel architecture, as the latency is $2m + 1$ and 1 clock cycles, respectively.

The use of FP operations in the implementation of the proposed architectures greatly enhanced the precision and accuracy of the system. Many of the methods listed in Table 3 have reportedly used fixed-point numbers. The use of fixed-point numbers would have resulted in a lower resource utilisation and power consumption as the operations involving FP numbers are computationally intensive [10, 17, 19]. However, the use of fixed-point numbers compromises with the accuracy of the system.

5. Conclusion: In this Letter, the FPGA implementation of a complete system for preprocessing ECG signals, extracting FECG, and subsequently calculating FHR is presented. For the removal of high-frequency components, power line interference, and baseline wander, Butterworth, Notch, and two-stage moving average filters are used, respectively. For FECG extraction, an LMS-AF is used, and series and parallel architectures are designed for its implementation. The precision and accuracy of the complete system are significantly enhanced by the use of FP arithmetic, for which an FPU is developed. Comparisons with previous work show that the proposed parallel architecture requires the least time for convergence of filter weights, while the proposed series architecture has low resource utilisation.

6 References

[1] Hasan M., Reaz M., Ibrahimy M., *ET AL.*: 'Detection and processing techniques of FECG signal for fetal monitoring', *Biol. Proced. Online*, 2009, **11**, pp. 263–295

[2] Martens S.M.M., Rabotti C., Mischi M., *ET AL.*: 'A robust fetal ECG detection method for abdominal recordings', *Physiol. Meas.*, 2007, **28**, (7), pp. 373–388

[3] Richter M., Schreiber T., Kaplan D.T.: 'Fetal ECG extraction with nonlinear state-space projections', *IEEE Trans. Biomed. Eng.*, 1998, **45**, (1), pp. 133–137

[4] Ping G., Ee-Chien C., Wyse L.: 'Blind separation of fetal ECG from single mixture using SVD and ICA'. 4th Int. Conf. on Information, Communications & Signal Processing and 4th Pacific-Rim Conf. on Multimedia, Singapore, 2003, vol. 3, pp. 1418–1422

[5] Wu S., Shen Y., Zhou Z., *ET AL.*: 'Research of fetal ECG extraction using wavelet analysis and adaptive filtering', *Comput. Biol. Med.*, 2013, **43**, (10), pp. 1622–1627

[6] Sutha P., Jayanthi V.: 'Fetal electrocardiogram extraction and analysis using adaptive noise cancellation and wavelet transformation techniques', *J. Med. Syst.*, 2018, **42**, (21)

[7] Hassanpour H., Parsaei A.: 'Fetal ECG extraction using wavelet transform'. 2006 Int. Conf. on Computational Intelligence for Modelling Control and Automation (CIMCA'06), Sydney, 2006, pp. 179–179

[8] Reaz M.B.I., Wei L.S.: 'Adaptive linear neural network filter for fetal ECG extraction'. Proc. Int. Conf. on intelligent sensing and information processing, Chennai, India, 2004, pp. 321–324

[9] Sameni R., Clifford G.D.: 'A review of fetal ECG signal processing issues and promising directions', *Open Pacing Electrophysiol. Ther. J.*, 2010, **3**, pp. 4–20

[10] Hatai I., Chakrabarti I., Banerjee S.: 'FPGA implementation of a fetal heart rate measuring system'. 2nd Int. Conf. on Advances in Electrical Engineering (ICAEE 2013), 2013, pp. 160–164

[11] Morales D.P., Garcia A., Castillo E., *ET AL.*: 'An application of reconfigurable technologies for non-invasive fetal heart rate extraction', *Med. Eng. Phys.*, 2013, **35**, (7), pp. 1005–1014

[12] Arias-Ortega R., Gaitan-Gonzalez M.J., Yanez-Suarez O.: 'Implementation of a real-time algorithm for maternal and fetal heart rate monitoring in a digital signal controller platform'. Annual Int. Conf. of the IEEE Engineering in Medicine and Biology, Buenos Aires, 2010, pp. 2354–2357

[13] Subha T.D., Reshma V.D.: 'A study of non-invasive heart rate monitoring system by using FPGA', *Mater. Today, Proc.*, 2017, **4**, (2), Part B, pp. 4228–4238

[14] Pani D., Dessi A., Cabras B., *ET AL.*: 'A real-time algorithm for tracking of foetal ECG sources obtained by block-on-line BSS techniques'. Computing in Cardiology, Krakow, 2012, **39**, pp. 65–68

[15] Tortia E., Koliopoulos D., Matraxia M., *ET AL.*: 'Custom FPGA processing for real-time fetal ECG extraction and identification', *Comput. Biol. Med.*, 2017, **80**, pp. 30–38

[16] Hasan M.A., Ibrahimy M.I., Reaz M., *ET AL.*: 'VHDL modeling of FECG extraction from the composite abdominal ECG using artificial intelligence'. Proc. IEEE Int. Conf. on Industrial Technology, ICIT, Gippsland, VIC, Australia, 2009, pp. 1–5

[17] Chareonsak C., Sana F., Wei Y., *ET AL.*: 'Design of FPGA hardware for a real-time blind source separation of fetal ECG signals', *IEEE Int. Workshop on Biomed. Circuits Syst.*, 2004, **S2**, (4), pp. 13–16

[18] Ibrahimy M.I., Reaz M.B.I., Yasin F.M., *ET AL.*: 'Fetal QRS complex detection algorithm for FPGA implementation'. IEEE Int. Conf. on Computational Intelligence for Modelling, Control and Automation, and Int. Conf. on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06), Vienna, 2005, pp. 846–850

[19] Sahin S., Kavak A., Becerikli Y., *ET AL.*: 'Implementation of floating point arithmetics using an FPGA', In: 'Mathematical methods in engineering' (Springer, Dordrecht, The Netherlands, 2007), pp. 445–453
[20] Chua L.O., Desoer C.A., Kuh E.S.: 'Linear and nonlinear circuits' (McGraw-Hill, New York, 1987)
[21] Awodeyi A.E., Alty S.R., Ghavami M.: 'Median based method for baseline wander removal in photoplethysmogram signals'. IEEE Int. Conf. on Bioinformatics and Bioengineering, Boca Raton Florida, 2014, pp. 311–314
[22] Widrow B., Stearns S.D.: 'Adaptive signal processing' (Prentice-Hall, Englewood Cliffs, NJ, 1985)
[23] Pan J., Tompkins W.J.: 'A real-time QRS detection algorithm', *IEEE Trans. Biomed. Eng.*, 1985, **32**, pp. 230–236
[24] von Steinburg S.P., Boulesteix A.L., Lederer C., *ET AL.*: 'What is the 'normal' fetal heart rate?', *PeerJ*, 2013, **1**, (82)
[25] 'IEEE standard for binary floating-point arithmetic', 1985 (ANSI/IEEE Std 754–1985)
[26] Goldberger A.L., Amaral L.A.N., Glass L., *ET AL.*: 'Physiobank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals', *Circ. Electron. Pages*, 2000, **101**, (23), pp. e215–e220
[27] De Moor B.L.R. (ed.): 'DaISy: Database for the Identification of Systems', Department of Electrical Engineering, ESAT/SISTA, K.U. Leuven, Belgium, URL: http://www.esat.kuleuven.ac.be/sista/daisy/, May 2019. [Used dataset: Cutaneous potential recordings of a pregnant woman, section Biomedical Systems, 96–012.]
[28] Behar J., Andreotti F., Zaunseder S., *ET AL.*: 'An ECG model for simulating maternal-foetal activity mixtures on abdominal ECG recordings', *Physiol. Meas.*, 2014, **35**, (8), pp. 1537–1550
[29] Le T., Moravec A., Huerta M., *ET AL.*: 'Unobtrusive continuous monitoring of fetal cardiac electrophysiology in the home setting'. IEEE Sensors, New Delhi, 2018, pp. 1–4
[30] Gini J.R., Ramachandran K.I., Nair R.H., *ET AL.*: 'Portable fetal ECG extractor from abdECG'. Int. Conf. on Communication and Signal Processing (ICCSP), Melmaruvathur, 2016, pp. 845–848
[31] Lima-Herrera S.L., Alvarado-Serrano C., Hernandez-Rodriguez P.R.: 'Fetal ECG extraction based on adaptive filters and wavelet transform: validation and application in fetal heart rate variability analysis'. 13th Int. Conf. on Electrical Engineering, Computing Science and Automatic Control (CCE), Mexico City, Mexico, 2016, pp. 1–6