Received 1 August 2022, accepted 10 November 2022, date of publication 18 November 2022, date of current version 8 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3223350

RESEARCH ARTICLE

Optimized Continuous Wavelet Transform Algorithm Architecture and Implementation on FPGA for Motion Artifact Rejection in Radar-Based Vital Signs Monitoring

AMEEN BIN OBADI[1,2], **MEDIEN ZEGHID**[3,4], **PHAK LEN EH KAN**[5], **(Member, IEEE), PING JACK SOH**[6], **(Senior Member, IEEE), MARCO MERCURI**[7], **(Senior Member, IEEE), AND OMAR ALDAYEL**[8,9,10], **(Member, IEEE)**

[1]Advanced Communication Engineering (ACE) CoE, Faculty of Electronic Engineering Technology, Universiti Malaysia Perlis, Arau, Perlis 02600, Malaysia
[2]Department of Electronics and Communication Engineering, Faculty of Engineering and Petroleum, Hadhramout University, Al-Mukalla, Hadhramaut, Yemen
[3]Department of Computer Engineering and Networks, College of Engineering in Wadi Alddawasir, Prince Sattam Bin Abdulaziz University, Wadi Alddawasir 11991, Saudi Arabia
[4]Electronics and Micro-Electronics Laboratory, University of Monastir, Monastir 5000, Tunisia
[5]Faculty of Electronic Engineering Technology, Universiti Malaysia Perlis, Arau, Perlis 02600, Malaysia
[6]Centre for Wireless Communications (CWC), University of Oulu, 90014 Oulu, Finland
[7]Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica (DIMES), University of Calabria, 87036 Arcavacata di Rende (CS), Italy
[8]Department of Electrical Engineering, King Saud University, Riyadh 11421, Saudi Arabia
[9]Prince Sultan Advanced Technology Research Institute (PSATRI), KSU, Riyadh 11451, Saudi Arabia
[10]KACST-TIC in RF and Photonics for the e-Society (RFTONICS), KSU, Riyadh 11451, Saudi Arabia

Corresponding authors: Ameen Bin Obadi (ameen.binobadi@gmail.com) and Ping Jack Soh (pingjack.soh@oulu.fi)

The work of Ping Jack Soh was supported by the Academy of Finland 6G Flagship Program under Grant 346949. The work of Marco Mercuri was supported in part by the Programma Operazione Nazionale e Innovazione 2014–2020, and in part by the Fondo Sociale Europeo, Azione 1.2 ‘‘Attrazione e Mobilità Internazionale dei Ricercatori’’ under Grant CUP H24I19000410005.

ABSTRACT The continuous wavelet transform (CWT) has been used in radar-based vital signs detection to identify and to remove the motion artifacts from the received radar signals. Since the CWT algorithm is computationally heavy, the processing of this algorithm typically results in long processing time and complex hardware implementation. The algorithm in its standard form typically uses software processing tools and is unable to support high-performance data processing. The aim of this research is to design an optimized CWT algorithm architecture to implement it on Field Programmable Gate Array (FPGA) in order to identify the unwanted movement introduced in the retrieved vital signs signals. The optimization approaches in the new implementation structure are based on utilizing the frequency domain processing, optimizing the required number of operations and implementing parallel processing of independent operations. Our design achieves significant processing speed and logic utilization optimization. It is found that processing the algorithm using our proposed hardware architecture is 48 times faster than processing it using MATLAB. It also achieves an improvement of 58% in speed performance compared to alternative solutions reported in literature. Moreover, efficient resources utilization is achieved and reported. This advanced performance of the proposed design is due to consciously implementing comprehensive approaches of multiple optimization techniques that results in multidimensional improvements. As a result, our achieved design is suitable for utilization in high-performance data processing applications.

INDEX TERMS Continuous wavelet transform, FPGA implementation, radar remote sensing, motion artifact rejection, random body movements, FFT-based CWT, parallel processing.

The associate editor coordinating the review of this manuscript and approving it for publication was Christian Pilato.

I. INTRODUCTION

Random body movement rejection in vital-sign detection application is one of the main challenges faced by the researchers in effectively implementing indoor radars [1], [2], [3], [4]. Moreover, there is more research directed recently towards real-time detection applications, as such feature is highly desirable. Real-time detection requires powerful processing capability, which is not always affordable for everyday use [5], [6], [7], [8], [9]. This opens opportunities of inventing novel solutions, which are capable of processing complex random body movements detection algorithms to meet the real-time requirements [10].

Different approaches were used in the literature to overcome the challenge of random body movements in vital signs. Researchers in [11] tackled this issue by using the continuous wavelet transform (CWT) algorithm to identify the contribution of the motion artifacts in the phase signal and then to smooth it by applying a moving average filter. Meanwhile, in [12] the features of the frequency spectrum of vital signs while undergoing random body motions are analyzed. This work utilized the motion modulation effect and extracted the direction of the body motion with the new position of the respiration peaks. Since body movements introduce frequency shifts in the spectrum, the direction and amount of this frequency shift depends on the direction and the speed of the body motion. Thus, this feature was used to account for the body motions in the spectrum to detect the breathing rate accordingly. Meanwhile, the work in [13] effectively reduced the random movement using two methods: the complex signal demodulation (CSD) method and the arctangent demodulation (AD) method, which were implemented on Doppler radar detection of vital signs. It was targeted for sleep monitoring and baby monitoring to eliminate false alarm caused by random movements. The CSD resulted to be more immune against the effects of the DC offset, whereas the AD reduces the effect of harmonics and inter-modulation interference and high carrier frequencies. Finally, an adaptive phase compensation method was used for random body movements cancellation in [14]. To measure the random body movements of a subject, a camera was integrated in the radar system. The camera measurement was fed back into the system as the phase information. The presence of large body movements may result in receiver saturation. However, the use of phase compensation avoids such a problem. A simple video processing was also performed to extract the random body information without using any markers.

The reported algorithm in [11] uses targets with random body motion that affect the detection of vital signs. It uses CWT to identify the locations of the artifacts and then applies the moving average filter to smooth these identified artifacts. It also uses the discrete wavelet transform (DWT) to separate the heartbeat signal from the respiration signal, which results in accurate detection. However, this innovative algorithm is computationally complex and thus does not overcome implementation requirements such as high speed, low resource utilization, and low-power consumption for high-performance and potential real-time processing scenarios. To overcome these challenges, this algorithm needs to be employed in innovative architectures in the processing platforms for high-performance processing metrics.

As will be outlined in the next section, the CWT algorithm has been used in the literature for different applications, in different structures, and designed and implemented in different processing platforms, see [15], [16], [17], [18], [19], and [20]. The use of CWT for the specific application of detection unwanted body movement was outlined in [11]. In the article, a standard desktop computer was used as the main processing unit for the algorithm. That work focused on proving the viability of the CWT algorithm when applied to more practical scenario using radar for detection and monitoring of heart rate (HR) and respiration rate (RR), so accuracy was one of the most important parameters to be reported. However, to investigate the high-performance data processing viability of the algorithm, other parameters (processing speed, resources utilization) are yet to be investigated. Since the algorithm was implemented in the software loaded into the central processing unit (CPU) of a desktop for execution, it implies that the implementation of the algorithm is performed in a sequential manner. In this case, execution speed, processing time, and the potential to apply such algorithm in real-time applications are questionable. In summary, the main gap in the literature is that the CWT algorithm is very complex and in order to process this algorithm especially when very large data samples are involved, it is important to consider the processing speed and resources needed. Consequently, it is apparent that there is a need to provide a CWT design and implementation that can be used for vital sign detection scenario with high-performance processing speed.

Since CWT in [11] has been validated to be successful in identifying unwanted movements, our proposed work here investigates the CWT use for high-performance detection of unwanted movement. This is achieved by designing an FPGA implementation architecture with high speed, low processing time and optimized hardware resources. To our best knowledge, the FPGA implementation of the CWT algorithm proposed in [11] has not been explored yet. The proposed solution provides a processing structure by adopting several optimization techniques. The design utilizes the FPGA reconfigurability and parallelism features to implement the optimization objectives.

A. OUR CONTRIBUTION

- A new architecture implementation of CWT on FPGA for unwanted body movement detection: we developed a new CWT architecture implementation on FPGA to overcome the gaps described earlier in the introduction. This was done through several optimizations implemented in the design to improve processing speed and logic utilization:
  - CWT processing speed optimization: we have been able to implement several speed optimizations on (i) pre-processing and modifications of the algorithm input data (ii) the FPGA architecture design. Examples of such optimizations are wavelet function optimization,

wavelet scales optimization, Fast Fourier Transform (FFT) output optimization and multiplication optimization. The CWT processing speed was improved due to adopting and implementing these optimizations.
- CWT resources utilization optimization: we have been able to implement several resources optimizations methods on (i) the algorithm input data pre-processing and modifications (ii) the FPGA architecture design. Examples of such optimizations are wavelet function optimization, wavelet scales optimization, FFT output optimization, multiplication optimization and random access memory (RAM) requirements optimizations. The CWT logic utilization was improved due to adopting and implementing these optimizations.

- Applicability to high-performance data processing for unwanted body movement detection application: we developed the design so that it can be applied for high-performance detection applications. This is due to the significant improvement achieved in the processing speed.

The hardware design optimization done in this work is not referred to the design optimization of the primitive blocks of functions such as the multiplier or the RAM block. The work is done in the following:

- How the multipliers and RAMs are used.
- The connection and the control of how and when each block is connected to other blocks to maximize speed and minimize resources.
- The flow of data from one block to another and how that is controlled to maximize speed and minimize resources.
- The input signal and wavelet samples

In summary, the above points allowed the use of the FPGA primitive blocks in the most efficient way in the design.

The rest of this paper is organized as follows: Section II presents related works in the literature, Section III provides brief background on CWT, Section IV provides details on the CWT processor design features and implementation, Section V outlines the results and comparison with the state of the art works, Section VI provides the conclusion.

II. RELATED WORKS

The CWT has been attracting researchers' interest in the last decades therefore multiple implementation approaches of the algorithm are presented in the literature. In [21], a CWT based approach was proposed to measure the voltage flicker resulting in power systems from fast load variations. It uses a Gaussian modulated wavelet function as the basis wavelet in the algorithm. Based on the resulting CWT coefficients, the flicker frequency response and the amount of system frequency deviation can be evaluated. This approach calculates the CWT coefficients in time domain and the algorithm was implemented using LabView software. It might be accurate in detecting flicker voltages in comparison to typical FFT approaches. However, performing it in time domain involves a very complex computational steps within the convolution, increasing the time requirements for processing. Besides that, [22] proposed a unified architectural framework for DWT and CWT based on a reconfigurable lifting scheme to be used in image processing application. The proposed architecture supports the use different wavelets based on the reconfigurability of the lifting scheme. The unified scheme consists of a reconfigurable lifting scheme array (RLSA), a reconfigurable address generator (RAG), two dual port static random access memory (SRAM) and the main controller unit. The design was implemented using very small number of scales and very small decomposition levels (3 levels), limiting the access and identification of various frequencies of interest. The processing of 512×512 image using the three-level decomposition was reported to be completed in 12.6 ms. The combination of DWT and CWT in one scheme might result in better resource utilization, especially when considering large number of scales and decomposition levels. Nonetheless, parameters of every new wavelet function to be used in this scheme need to be defined and then embedded in the design to start calculating the wavelets. This step can be replaced by precalculating the wavelet coefficients themselves, storing them in a memory and then calling them when needed.

Another algorithm that combines the use of DWT and CWT is presented in [23]. In this work, a hybrid method utilizing DWT and CWT was implemented on FPGA for underwater target motion estimation in sonar systems. The design parts consist of the DWT bank filtering for signal de-noising and the CWT convolver for target motion estimation. In the proposed structure of the CWT, only one multiplier with one adder were used to map the convolution process. This work was then improved by the same authors and presented in [24]. Scale optimization block was added to this design to estimate the sets of filters coefficients to be used by the CWT by determining the optimal scales. Despite being focused on saving of area, this comes at the cost of its speed of computation. In this version of the design, the CWT is implemented using 5 multipliers and 4 adders following the same principle in the previous design flow. Despite Improving the computation time, the convolution itself in time domain is more complex compared to doing only multiplication in frequency domain.

Next, the researchers in [25] developed an algorithm to detect and classify six types of electrocardiogram (ECG) signal beats using neural network classifier. Prior to inserting the signal samples to the classifier, they were pre-processed using CWT and principal component analysis (PCA) algorithm. CWT was mainly used to extract the ECG signal features while the use of PCA was to reduce the size of the data before being fed as input to the classifier. The combination of CWT and PCA featured a more effective input data to the neural network leading to better classification results. The HAAR mother wavelet was used in the CWT estimation using ten scales via MATLAB. The performance accuracy of the classifier in this method is very high. The speed feature of this method was not investigated, and no results were reported. However, it should be apparent that the use of CWT increases the complexity and time consumption of the processing.

The works presented in [26], [27], and [28] are multiple improvements built on each other by the same researchers. They presented the design and the implementation of CWT algorithm using FPGA to detect and extract the event related potential (ERP) signal part of the electroencephalogram (EEG) signals. The basic idea of the implemented algorithm is to conduct the CWT in frequency domain rather than time domain to speed up the computation. The algorithm processing steps are designed and implemented on FPGA. In addition, optimization techniques were used to further reduce the time processing requirements as well as the logic utilization. In this implementation, the zeroes in the Morlet wavelet were removed from calculations. In addition, the scales used were reduced to focus only on the scales supporting the targeted ERP feature for extraction. The initial design computes the CWT in 1 ms, which was improved by the optimization to around 0.57 ms. The achieved run-time is good in this work partially due to the moderate number of samples in the signal (1024 points). Further investigations should be carried out for longer lengths of the signal and how that affects the run time and logic utilization.

The work in [29] proposed a fringe pattern recognition method using CWT in Fourier space. This work attempted to reduce the algorithm execution time by designing and implementing it on FPGA. Its design consists of the wavelet core, input and output buffer memory sized at 1 KB each and a NIOSII processor. The design was tested on a 512×512 fringe pattern image, which was downloaded on the external SRAM to the FPGA. The FPGA requires around 100 ms to process the image while using C language it needs around 1 s, and around 650 ms if a high-end station with higher processing capability was used. The resources utilization when using Altera cyclone IV was 61% of logic elements, 49% of on chip memory and 100% of embedded multipliers.

Another fringe pattern recognition and fringe phase extraction application using CWT was presented in [30]. The CWT algorithm is implemented on an FPGA using the multiplication between the two-dimensional pattern spectrum and the two-dimensional wavelet kernel spectrum to avoid the complex convolution operations and reduce processing time. The Morlet function was used as the mother wavelet. The design consists mainly of digital data acquisition module, data buffer module, configuration module, CWT operation module and an output module. The heart of the design is the CWT operation. When the input signal matrix samples are of 256 × 256 in size, the FPGA execution time is less than 10 ms (using 200 MHz clock) whereas when using MATLAB, it was 1 s.

The work proposed in [31] was aimed to design a hardware description language (VHDL) module that detects the R wave and interval in an ECG signal to obtain the HR in real-time using the CWT with splines. The CWT was designed to function in time domain with the wavelet function selected as the first derivative of a second order spline function. Among all scales of the wavelet function, only one scale was used in the CWT calculation, which is scale 8, and it was selected because it contains the frequencies of interest in the passband. The processing time requirement in this design was 20 ms with an accuracy of 90%. The designed module was then implemented in an FPGA prototype presented in [32]. In this prototype, four modules were implemented: the data receiver module, the module for obtaining the HR, module to manage the storage of data in micro-SD card and a module to manage the processed data visualization. The prototype was tested with 8 files of input data with a reported accuracy of 99.5%. The prototype design bandwidth (BW) is 200 Hz and power consumption of 625 mW/h.

III. BACKGROUND ON CWT

CWT is a windowing technique with variable-sized regions that allows to have different frequency and time resolutions depending on what is needed. For example, at high frequencies usually a high time resolution is necessary, but not at low frequencies where a high time resolution causes redundancy. With CWT, a time frequency representation of the signal is obtained and therefore the artifacts, that typically have also higher frequencies than the normal vital sign signals, can be clearly seen and located in time. Using CWT, the vital signs phase signal disturbed by the artifacts and its corresponding CWT can be extracted. The artifacts can be identified in the time domain exploiting the frequency information [33], [34], [35], [36].

The CWT of an input signal $x(t)$ for a selected mother wavelet function $\Psi(t)$ at a given time b and scale a is defined as [37], [38], [39], [40], and [41]:

$$(a, b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} x(t)\Psi(\frac{t-b}{a})dt. \tag{1}$$

The definition of the convolution between the input signal $x\,(t)$ and a linear time invariant filter $h\,(t)$ is:

$$(x * h)\,(b) = \int_{-\infty}^{\infty} x(t)h(b-t)dt, \tag{2}$$

if $h\,(t)$ is defined in terms of the mother wavelet function as:

$$h(t) = \frac{1}{\sqrt{a}}\Psi(\frac{-t}{a})dt. \tag{3}$$

The CWT can be seen as the convolution process between the input signal and the mother wavelet in time domain. This convolution produces the CWT coefficients. This convolution equation in a simple compact form is [28]:

$$C(a, b) = \int_{-\infty}^{\infty} x(t)\Psi^{*}_{a,b}(t)dt, \tag{4}$$

where the sign $*$ represent complex conjugate.

If the total number of convolutions is small (this is true in the case of small signal samples) this time domain method can be used in practical situations. Alternatively, the CWT algorithm is calculated in frequency domain in which the convolution operation is replaced by multiplication operation as follows:

$$g1(t) * g2(t) \;<=>\; G1(\omega) \times G2(\omega), \tag{5}$$

TABLE 1. Equations of commonly used mother wavelets.

Wavelet	Domain	Equation
Morlet (ω_0 is the frequency)	Time	$\pi^{-1/4}e^{i\omega_0 t}e^{-t^2/2}$
Morlet (ω_0 is the frequency)	Frequency	$\pi^{-1/4}H(\omega)e^{-(s\omega-\omega_0)^2/2}$
Mexican Hat ($m = order$)	Time	$\frac{2^m i^m m!}{\sqrt{\pi(2m)!}}(1-it)^{-(m+1)}$
Mexican Hat ($m = order$)	Frequency	$\frac{2^m}{\sqrt{m(2m-1)!}}H(\omega)(s\omega)^m e^{-s\omega}$
Paul ($m = derivative$)	Time	$\frac{(-1)^{m+1}}{\sqrt{\Gamma(m+1/2)}}\frac{d^m}{dt^m}(1-it)^{-(m+1)}$
Paul ($m = derivative$)	Frequency	$\frac{i^m}{\sqrt{\Gamma(m+1/2)}}(s\omega)^m e^{-(s\omega)^2/2}$

$H(\omega)$is the Heaviside step function, $H(\omega) = 1\ if\ \omega > 0$ and zero otherwise.
ω_0 is the non-dimensional frequency.
s is the scale.
$\mathbf{\Gamma}$ stands for the gamma function.

where the lower-case g is the input signal and wavelet signal in time domain while the upper-case G is the input signal and wavelet signal in frequency domain. In addition, the sign $*$ represents the convolution.

By definition, processing using CWT requires the selection and use of a wavelet function either in time domain or a frequency domain. If the process is based on time domain, then the most computationally complex step is the convolution between the wavelet function and the input signal. If transformation is based on frequency domain, then the process is in the form of multiplication between the wavelet function and the input signal.

In every CWT, the mother wavelet signal needs to be selected. There are certain criteria for the selection of the mother wavelet such as the finite energy and the admissibility factor, which are elaborated in [31]. The most commonly selected mother wavelets in the CWT algorithm are the analytic Morlet, the Mexican hat and the Paul wavelet functions whose formulas are shown in Table 1 [28]. It summarizes the three functions equations in time domain and their corresponding frequency domain equations.

One of the most widely used mother wavelets in the biomedical applications is the Morlet function. The Morlet wavelet consists of complex sinusoidal waves modulated by Gaussian envelop. The Morelt function is typically selected for biomedical applications such the vital sign detection due to its simplicity and suitability for spectral analysis.

The Morlet wavelet used by MATLAB is defined by:

$$\Psi(\omega) = 2e^{-(\omega-6)^2/2}U(\omega), \tag{6}$$

where $U(\omega)$ is the unit step in frequency domain. if the unit step is ignored for a while, the time domain Morlet used in MATLAB is:

$$\Psi(t) = \sqrt{\frac{2}{\pi}}e^{-t^2/2}e^{i6t}. \tag{7}$$

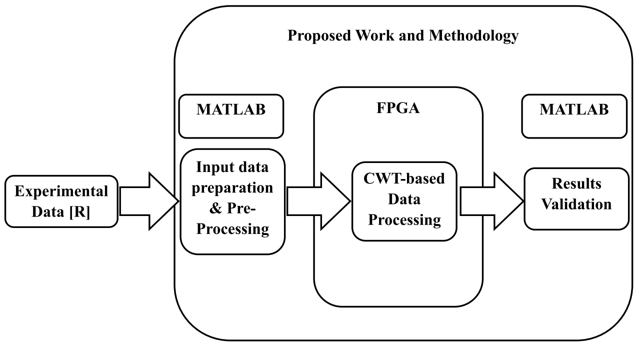

Proposed Work and Methodology
MATLAB
FPGA
MATLAB
Experimental Data [R]
Input data preparation & Pre-Processing
CWT-based Data Processing
Results Validation

FIGURE 1. Block diagram of the CWT processor design methodology.

IV. CWT PROCESSOR DESIGN FEATURES AND IMPLEMENTATION

Different methods and approaches are planned to optimize an architecture for the CWT algorithm on FPGA to be used for vital sign detection. The following approaches are among the ones used to achieve the optimized design:

- Capitalizing on the parallelism features of FPGA.
- Reducing unnecessary calculation steps in the algorithm.
- Optimal selection of the scales fit for the application.
- Using the available IP cores.

A. CWT PROCESSOR DESIGN METHODOLOGY

The overall block diagram of the proposed CWT processor design methodology is shown in Fig. 1. It contains three main parts of this work, which are:

- Input data preparation and preprocessing;
- CWT-based data processing;
- Results validation.

The first and the third ones are considered the software parts and are processed using MATLAB while, the second one, is the hardware part, which includes the FPGA.

Detailed description of all components of Fig. 1 is presented below.

1) EXPERIMENTAL DATA UNIT

This unit represents the experimental measurement data matrix resulting from the experiment conducted in [11]. Each element in the Matrix **R** is a complex element of a real part and an imaginary part ($I + Q$) of base-band digitized signal samples. **R** contains 45,572 rows representing the slow time samples and 50 columns representing the fast time samples or (range). The important parameters used in the experiment which are utilized in this work are the following:

- Sampling time in slow time (across rows) t_{s0} = 3.072 ms.
- Sampling frequency in slow time (across rows) f_s = 325.5208 Hz.
- Range resolution $\Delta_R = 0.02$ m.

The above parameter values of the sampling time, sampling frequency, and range resolution are based on the system designed in [11].

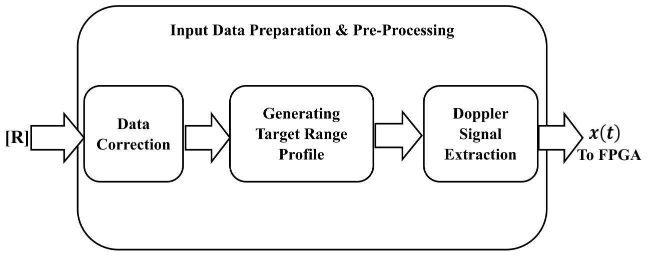

Input Data Preparation & Pre-Processing
[R]
Data Correction
Generating Target Range Profile
Doppler Signal Extraction
$x(t)$
To FPGA

FIGURE 2. Block diagram of the input data preparation and pre-processing.

2) INPUT DATA PREPARATION AND PRE-PROCESSING UNIT

Internally, this unit consists of three sequential steps as shown in the Fig 2. The purpose of this step is to correct the data from the offset caused by the cables in the experiment and focus on the time and range of interest. This step is also important to insert the correct data to the CWT processor, which contains the target information where oscillations due to the presence of the RR and HR of the volunteers.

a: DATA CORRECTION

To properly process the data matrix **R**, some correction steps are applied to extract the part of the matrix on which subsequent algorithms in the FPGA is applied, these steps are:

- Starting the data processing of **R** at the 21st second of the measurement, as the first 20 seconds were preparation to actual measurement.
- Starting the data processing of **R** at the 7th sample in the columns to account for around 1.4 m offset due to the cables used in the experiment. This reduces the range of interest to less than 10 m.
- Resulting matrix is **G** with 39,062 rows representing slow time samples and 44 columns representing fast time samples (range).

b: GENERATING TARGET RANGE PROFILE

- Generating slow time axis (x axis) vector using the sampling time data in slow time.
- Generating range axis (y axis) vector using the range resolution data.
- Finding magnitude of each element in the complex matrix **G**.
- Mapping magnitudes of **G** data to the mesh grid having slow time as x axis and ranges as y axis with color map indicating the data values. The target range profile is shown in Fig 3.

c: DOPPLER SIGNAL EXTRACTION

- From the range profile, identifying the range bin with oscillation, which is in this case $(range_bin) = 13$ indicating target presence at 2.6 m from the radar.
- Creating the Doppler signal by keeping all rows of **G** and only one column corresponding to $(range_bin) = \mathbf{13}$. This result is the Doppler signal complex vector of size 39,062. The magnitude of this Doppler signal is shown in Fig. 4.

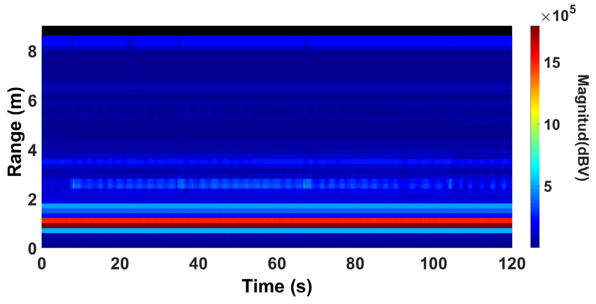


FIGURE 3. Target range profile.

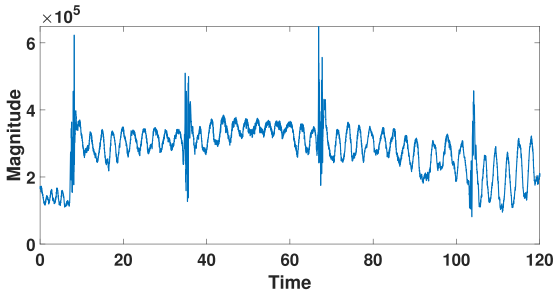


FIGURE 4. Doppler signal magnitude.

The Doppler signal is used as input of the CWT processing unit. The length of the complex test signal $x(t)$ selected for the processing unit is of $N = 4096$ samples.

3) CWT BASED DATA PROCESSING UNIT

Before starting the FPGA design, all steps of CWT in the context of vital sign detection and using the experimental data collected are simulated in MATLAB to ensure correctness of the steps and to establish a comparison point later. The algorithm steps are configured on the FPGA again to ensure correctness of these steps and to establish another point of reference.

There are two main CWT basic algorithm structures found in the literature: the time domain and the frequency domain. Because the first one involves complex convolution process, the second one is adopted in this work. The most commonly used methods to move from time domain to frequency domain are the short Fourier transform (STFT) and the FFT. Because the latter involves lower computational complexity, it is adopted in this work. The structure of the FFT-based CWT algorithm is shown in Fig 5. It shows the flow chart of the CWT algorithm, which contains the following critical functions/steps:

- FFT;
- Multiplication;
- Inverse FFT (IFFT).

Those are the most complex and time-consuming operations of the CWT algorithm.

To determine the size and the number of scales of the mother wavelet, the sampling frequency and length of the input signal needs to be known. Once the scales are

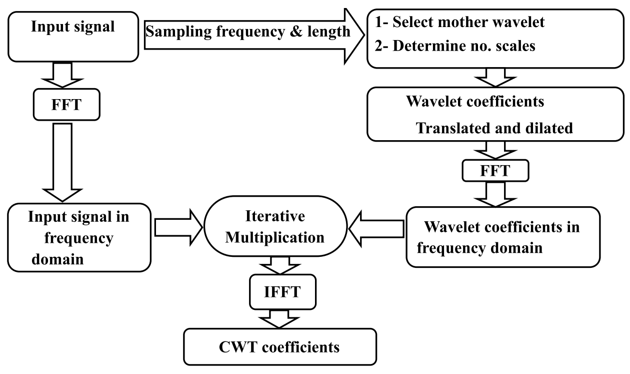


FIGURE 5. **Flowchart of the FFT-based CWT algorithm.**

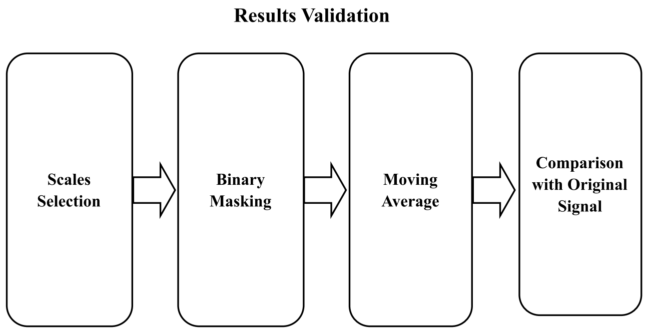


FIGURE 6. **Block diagram of result validation process.**

determined, the wavelet coefficient matrix is generated in time domain $\Psi(t)$, which has all dilated and translated versions of the mother wavelets based on the scales. To shift from time domain to frequency space, the FFT should be applied both on the input signal $x(t)$ and on the wavelet coefficients $\Psi(t)$ to obtain $X(\omega)$ and $\Psi(\omega)$, respectively. After that, the frequency domain signal is multiplied by the wavelet coefficient in frequency domain at each scale. The resulting multiplications at each scale are converted back to time domain using the IFFT process to produce the CWT coefficients in time domain. The algorithm is applied for the proposed design as follows:

- The sampling frequency of the input signal is $f_s = 325.5$ Hz and length is $N = 4096$.
- The mother wavelet function is selected (Morlet wavelet) with N samples in time domain.
- The total number of scales $S = 89$. This is determined based on the input signal considering two factors: 1) the sampling frequency of the input signal and 2) the number of samples of the input signal. These two factors are used as input to MATLAB to determine the number of scales.
- Generating the wavelet coefficient matrix in time domain $\Psi(t)$ of size $N \times S$ where each column represents the wavelet function in time domain at each scale.
- Applying the FFT on $\Psi(t)$ at each scale to get the frequency domain wavelet coefficients $\Psi(\omega)$ of size $N \times S$, where $\text{FFT}(\Psi(t)) = \Psi(\omega)$. The FFT requires $S\,N\,log\,N$ operations.
- Applying the FFT on $x(t)$ to get the frequency domain vector signal $x(\omega)$ of size N samples. $FFT(x(t)) = x(\omega)$, where $x(\omega)$ is a complex vector of data. The FFT requires $N\,log\,N$ operations.
- Applying point by point multiplication between each point in the column vector $x(\omega)$ and the corresponding points in the first column of the wavelet coefficient matrix $\Psi(\omega)$ already calculated, then repeat the multiplication operation between the points of $x(\omega)$ with the corresponding points of the second column of the wavelet coefficients matrix $\Psi(\omega)$, and then the third column until reaching to the last column no. S. The output of this step is a matrix let us call it $\mathbf{M}_x(\omega) = x(\omega) \times \Psi(\omega)$. The number of multiplication operations here is $2 \times N \times S$ operations (since there is real part and imaginary part in each sample of $x(\omega)$).
- Applying the IFFT on $\mathbf{M}_x(\omega)$ column wise. The result of this operation is the matrix $\mathbf{W}_{x(t)}$ containing the CWT coefficients results, which has N rows and S columns. $IFFT(\mathbf{M}_x(\omega)) = \mathbf{W}_{x(t)}$.

To better optimize the algorithm, several observations on the CWT algorithm steps are as follow:

- From the input signal sampling frequency and the number of samples, the number of scales of the selected mother wavelet is determined prior to algorithm computation.
- The algorithm implementation is optimized by directly calculating the wavelet coefficients matrix in the frequency domain $\Psi(\omega)$. This is done instead of calculating it in time domain and then applying the FFT to obtain the frequency domain coefficients. This optimization significantly reduces computation time of applying the FFT on $\Psi(t)$, which is of complexity around $N\,SlogN$.
- The algorithm implementation is optimized by pre-calculating the wavelet coefficients and storing them instead of synthesizing the wavelet equation in frequency domain. This approach reduces resources when implementing the wavelet equation in FPGA as well as the computational time. This optimization approach makes the algorithm design and FPGA architecture implementation more modular and reusable for other wavelets without drastically modifying the design.
- The IFFT can be optimized by having it start once enough input is available from the output of the multiplication. It does not have to wait until full completion of the multiplication. This makes these steps overlapping in time, which improves the speed of the calculation.

4) RESULTS VALIDATION UNIT

The purpose of this unit is to validate the resulting CWT coefficients to ensure that design accurately identifies the artifacts' locations, which are then suppressed. The block diagram of this step is shown in Fig. 6.

a: SCALES SELECTION (APPLICATION DEPENDENT)

The resulting CWT coefficient matrix contain profiling of the signal components on which CWT was applied.

The lower scale numbers such as 1,2,3, etc. contain the highest frequency components of the signal while the highest scales such as S, S-1, S-2, etc. contain the lowest frequency components of the signal. In the vital signs signal, the maximum mechanical displacements of lungs and heart have typical amplitudes of 1 mm and 0.1 mm, respectively. Depending on the subject activity and health condition, the vital signs frequencies range between 0.1 Hz to 3 Hz. Besides that, there exists certain frequency ranges in the target signal which need investigation to identify the unwanted movement corrupting the vital signal. These artifacts typically have higher frequencies and amplitude than the typical vital signs. The range of frequencies of 4 Hz to 20 Hz is selected, which corresponds to scales between 26 and 50. Thus, these scales are used to represent the range of frequencies of interest on which binary masking is applied.

b: BINARY MASKING

- The binary masking is applied on the CWT coefficients from scales 26 up to scale 50, therefore it is applied on $\hat{S} = 25$ scales only and not all the S scales.
- This is done by creating new vectors containing the maximum magnitudes of the CWT coefficients at the frequency range of interest (to locate artifacts of unwanted movements). To be specific, it is achieved by finding the maximum magnitude of the columns from $\mathbf{W}_{x(t)}$ at the $\hat{S}$ scales resulting in $s1$ of $\hat{S}$ samples size.
- Then, setting up threshold TRS_1 for binary masking. All values below TRS_1 are zero and all values above TRS_1 are one, so that:
  If $s1 \leq TRS_1$ then $s1 = 0$ else $s1 = 1$.
- The resulting vector where values are 1's are the artifacts locations where moving average is applied.

c: MOVING AVERAGE

- This is applied on Doppler signal at the artifacts locations.
- This is done by first setting up the number of moving average points: $pts = 101$. This value is selected due to the unsatisfactory level of artifact reductions produced by other values (e.g. 21 and 51 points). Instead, the choice of 101 points performed well in attenuating the artifacts.
- Find indices of the artifacts in the binary masking where to apply moving average ind_1.
- Apply moving average on $x(t)$ at the indices where the artifacts are found, as:
  $x(ind_1(i)) = 1/pts(v_{(i-50)} + v_{(i-49)} + v_{(i-48)} \ldots\ldots + v_i + v_{(i+1)} + \ldots + v_{(i+50)})$.
  where $x(ind_1(i))$ is the value of x at index ind_1 of (i); (i) is any integer value starting from 1 until $length(ind_1)$ and v is the sample value at certain location.

d: COMPARISON

After the moving average is applied at the artifact's locations, the resulting signal is compared with the original $x(t)$ to observe the level of improvement.

B. CWT PROCESSOR ARCHITECTURE AND FPGA IMPLEMENTATION

1) BASIC IMPLEMENTATION ARCHITECTURE

The input signal $x(t)$ is separated into two parts, one containing the real part $I(t)$ and the other containing the imaginary $Q(t)$, with each part consisting of N samples. $I(t)$ is used as the input test signal in the FPGA design. The CWT steps identified are mapped from the hardware point of view, with Fig. 7 presenting the basic block diagram of the operations and architecture of the FFT-based CWT algorithm. It shows the hardware implementation of the FFT-based CWT algorithm of Fig. 5. Therefore, it represents the second part of Fig. 1 (i.e., CWT-based data processing). The main functions/steps, such as FFT, multiplication, IFFT, are designed and implemented. In addition, the "control module" is used to control the data flow, to synchronize the processes, and to establish/define the relation between all the design blocks and modules. Moreover, since some data needs to be stored during the process, different RAMs are introduced to the design. In this work, VHDL was used for FPGA design coding and implementation. Quartus prime 15.1 software was used in all the stages of the FPGA design. The processing steps of this design are as follow:

- Storing $I(t)$ in RAM_1 at one sample per clock cycle. The memory size is of N locations each of 20 bits width.
- Reading $I(t)$, which is of 20 bits width from RAM_1 to the N points FFT. This FFT is of fixed-point representation of the data.
- Store the output of the real part of the FFT operation at RAM_2, while the imaginary part at RAM_3. These two memories are of size N locations each of 20 bits width.
- Calculating the Morlet wavelet values via MATLAB using the input signal length N and the sampling period t_{s0}. This results in a huge matrix of size $N \times S$ containing all the frequency domain values of the wavelet function for S scales. Each value is represented with 8 bits.
- Writing the Morlet wavelet coefficients form the wavelet input interface to RAM_0, which is of size $N \times S$ locations each of 8 bits width.
- Synchronized reading from RAM_0 and RAM_2 to the inputs of the multiplier module at a rate of one sample per clock cycle. This reading from RAM_2 starts from 0 to $N-1$ and then the process is repeated S times. While the reading from RAM_0 starts from 0 to $N-1$ and then continues from N to $2N-1$ and so on to go over all the values of RAM_0.
- Multiplying RAM_2 and RAM_0 samples at latency of one clock cycle and storing the output in RAM_4 of size $N \times S$ locations each of 28 bits width,
- Synchronized reading from RAM_0 and RAM_3 to the inputs of the multiplier module at a rate of one sample per clock cycle for N cycles. Then, repeated reading of RAM_3 values for S times while the reading from RAM_0 continues to go over all the values corresponding to all S scales.

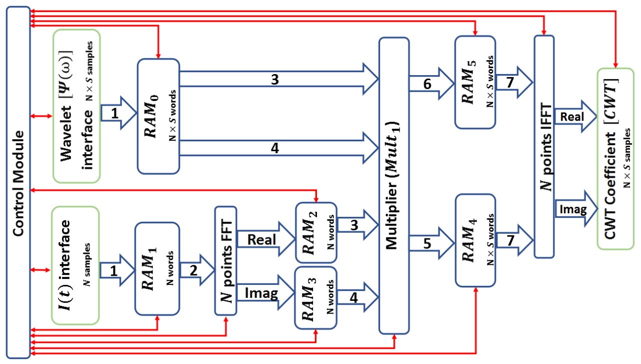

Control Module
Wavelet $[\Psi(\omega)]$ interface $N \times S$ samples
$I(t)$ interface N samples
RAM_0 $N \times S$ words
RAM_1 N words
N points FFT
Real
Imag
RAM_2 N words
RAM_3 N words
Multiplier $(Mult_1)$
RAM_5 $N \times S$ words
RAM_4 $N \times S$ words
N points IFFT
CWT Coefficient $[CWT]$ $N \times S$ samples
1
2
3
4
5
6
7

FIGURE 7. Basic FFT-based CWT processor architecture.

- Using the same multiplier, multiplying RAM_3 and RAM_0 samples at latency of one clock cycle and storing the output in RAM_5 of size $N \times S$ locations each of 28 bits width.
- Reading from RAM_4 and RAM_5 to the N points IFFT. This IFFT is of fixed point representation of the data.
- Obtaining the IFFT results, which represent the $\mathbf{W}_I$ values.
- Using the control module as the brain sending and receiving of control signals to the different modules of the design. It generates all memory addresses, $read_enable$ and $write_enable$ signals, in addition to controlling and synchronizing the operation of FFT, Multiplier and the IFFT.

The green outlined boxes in Fig. 7 are the input and output modules of the system while the numbers inside the arrows represent the sequence of the process. In addition, the red lines connecting the control module to all other modules in the system represent all the control interface and signals.

2) OPTIMIZED IMPLEMENTATION ARCHITECTURE

Several approaches are proposed in this part of the paper to advance the performance of the algorithm. Based on the previously presented block diagram in Fig. 7, optimizations in the following areas are proposed:

a: WAVELET FUNCTION OPTIMIZATION

Since the Morlet values in frequency domain are needed in this work, precalculating the Morlet frequency domain values and storing them in a memory to be used in the design is the approach adopted in this work for the following reasons. First, Previous approaches synthesized the Morlet time domain function (then applied the FFT) or synthesized the Morlet equation in frequency domain to obtain the required values in the frequency domain. However, this process is complex and consumes excessive resources given the complex nature of the Morlet equation. Second, this time domain to frequency domain conversion adds more computational steps that are avoidable. Third, the adopted approach in this work makes the design more modular. If the wavelet selection is changed then only the memory needs to be reloaded with the new values of the new wavelet. On the contrary, if the function is synthesized, it requires redesigning the whole wavelet circuitry. Using the proposed method, a large amount of the FPGA resources are saved for use in other processing steps, resulting in a simpler design for possible implementation on inexpensive FPGAs. Moreover, a memory initialized with.mif file containing the precalculated wavelet values is used instead of writing the samples one by one to the memory.

b: WAVELET SCALES OPTIMIZATION

The optimization in the wavelet scales is conducted as follows:

- Eliminating unneeded scales: the required wavelet values consist of $N \times S \times 8 = 2,916,352$ bits. If the required number of scales is reduced, then there can be an excellent reduction in the number of samples as well. Each wavelet scale represents a bandpass filter (BPF) with specific center frequency. Based on knowledge of the input signal and the expected range of frequency components, only selected scales of interest with the possible frequency components of the input signal are kept, while all other scales can be removed. It is expected that the unwanted movements causing artifacts are of higher frequency than the typical vital signs. Due to the nature of these artifacts originating from unwanted movements (moving limbs, crossing legs, waving hands etc.), the range of frequencies of interest falls between around 4 Hz to 20 Hz. This corresponds to the scales between 26 and 50, whereas the remaining scales are not needed. Eliminating these unneeded scales is performed specifically for this application to reduce the required memory size and the number of multiplication cycles. Reduction of $S - \hat{S} = 64$ scales is obtained in this step. The reduction obtained in this step by only selecting the needed scales is:
  - Total bits in S scales: $S \times N \times 8 = 2,916,352$ bits.
  - Total bits in selected $\hat{S}$ scales: $\hat{S} \times N \times 8 = 819,200$ bits.
  - Total Reduced bits from S scales: $64 \times N \times 8 = 2,097,152$ bit.
- Eliminating unneeded samples: in addition, another optimization level on scales is done by removing unneeded samples. By closely looking at the structure of each wavelet scale, it is noticeable the BPF have many leading zeros and trailing zeros, the non-zero values are concentrated in the middle of each scale. Moreover, the wavelet scale shape gets narrower as we move from scale 1 to scale S, resulting in the increase of zero values in each scale samples. The idea here is to keep track of the index of nonzero values rather than storing, reading, and multiplying the whole scale samples (zeros and non-zeros). This is because storing the zero values is not useful and its result in multiplication is already known. Table 2 presents all the wavelet at each scale with the indices of non-zero values. The $\hat{S}$ selected scales of interest are also presented in the table starting from scale

26 to scale 50. Since the number of zeros constitute large number of samples, this reduces the memory needed to store the precalculated wavelet values and reduce the number of cycles needed for the multiplication process. This also reduces the memory requirements for the RAM needed to store the multiplication results.

Based on the proposed reduction of scales and zeros, total number of non-zero points needed to be stored is 6220 points (locations). To standarize the RAM size needed, these points are reduced to 6144 points by removing 76 points from either scale 26 or 50 to minimize the effect on the results. The exact memory size needed is $6144 \times 8 = 6$ KB. As a result, only 6144 wavelet samples are read from the RAM to perform 6144 multiplications.

The reduction obtained in this step by removing zeros from selected scales is:

- Total bits in $\hat{S}$ scales: $\hat{S} \times N \times 8 = 819,200$ bits.
- Non-zero bits in $\hat{S}$ scales: $6144 \times 8 = 49,152$ bits.
- Reduced bits from $\hat{S}$ scales: $819,200 - 49,152 = 770,048$ bits.

This results in huge improvements in three dimensions:

- RAM needed to store the wavelet values.
- Number of multiplication cycles needed (one cycle per sample).
- RAM needed to store the multiplication outputs.

c: FFT OUTPUT OPTIMIZATION

This FFT module is adopted from OpenCores in Quartus Prime. The input signal to the system $I(t)$ is read by the FFT as N samples each sample is 20 bits, the output of the FFT is again N samples while each sample is 20 bits. This means a RAM of $N \times 20 = 81,920$ bits is needed to store the real part of the FFT output (RAM_1). A similar RAM is also needed to store the imaginary part of the FFT output (RAM_2). However, RAM_1 and RAM_2 locations are reduced based on the following:

- Since zero reduction is applied to the wavelet scales, the corresponding index containing non-zero values at the output samples from the FFT are of no use.
- As a result, these corresponding values in the FFT output do not need to be stored in RAM_1 and RAM_2, hence further memory reduction as well as reduction in the number of multiplication cycles is achieved.
- According to Table 2, the lowest index used from the wavelet values is index 40 while the highest index is 709, therefore only 670 samples are needed from the FFT output rather than N samples.

The reduction obtained in RAM_1 and RAM_2 by considering the FFT output optimization is:

- Total bits size of RAM_1 and RAM_2 prior to considering the FFT output optimization: $2 \times N \times 20 = 163,840$ bits.
- Total bits size of RAM_1 and RAM_2 post considering the FFT output optimization: $2 \times 670 \times 20 = 26,800$ bits.
- Reduced bits size in RAM_1 and RAM_2: 137,040 bits.

TABLE 2. Wavelet scales of non-zero components and their corresponding frequencies.

Scale	Corresponding center frequencies of wavelet (Hz)	Start Index of Non-Zero Points	End Index of Non-Zero Points	Non-Zero Points Length
1	119.88	1149	3500	2352
2	111.85	1072	3500	2429
3	104.36	1000	3490	2491
4	97.38	933	3257	2325
5	90.85	871	3039	2169
6	84.77	813	2835	2023
7	79.09	758	2645	1888
8	73.80	708	2468	1761
9	68.85	660	2303	1644
10	64.24	616	2149	1534
11	59.94	575	2005	1431
12	55.93	537	1871	1335
13	52.18	501	1745	1245
14	48.69	467	1629	1163
15	45.43	436	1520	1085
...	...	...	...	...
...	...	...	...	...
26	21.19	204	709	506
27	19.77	191	662	472
28	18.45	178	617	440
29	17.21	166	576	411
30	16.06	155	538	384
31	14.99	145	502	358
32	13.98	135	468	334
33	13.05	126	437	312
34	12.17	118	408	291
35	11.36	110	380	271
36	10.60	103	355	253
37	9.89	96	331	236
38	9.22	90	309	220
39	8.61	84	288	205
40	8.03	78	269	192
41	7.49	73	251	179
42	6.99	68	234	167
43	6.52	64	219	156
44	6.09	60	204	145
45	5.68	56	190	135
46	5.30	52	178	127
47	4.94	49	166	118
48	4.61	46	155	110
49	4.30	43	144	102
50	4.02	40	135	96
...	...	...	...	...
...	...	...	...	...
70	1.00	11	34	24
71	0.94	10	32	23
72	0.87	10	30	21
73	0.82	9	28	20
74	0.76	9	26	18
75	0.71	8	24	17
76	0.66	8	23	16
77	0.62	7	21	15
78	0.58	7	20	14
79	0.54	7	18	12
80	0.50	6	17	12
81	0.47	6	16	11
82	0.44	6	15	10
83	0.41	5	14	10
84	0.38	5	13	9
85	0.35	5	12	8
86	0.33	5	12	8
87	0.31	4	11	8
88	0.29	4	10	7
89	0.27	4	9	6

Total Nonzero points = 41,896.
Closest standard memory size is 41,984 (41k).

d: MULTIPLICATION OPTIMIZATION

The multiplication module is adopted from OpenCores in Quartus Prime.The optimization in the multiplication process is conducted as follows:

- Starting Time of the Multiplication Process: typically, once the FFT completes its operation and produces all the values to RAM_1 and RAM_2, then the multiplication process commence. However, when closely examining the design, the multiplication process related to values of scale 26 needs the sample number 204 from the FFT output, and the index is incremented until it arrives to the last value (709) corresponding to scale 26 as shown in Table 2. After that, the multiplication of the values from scale 27 onwards starts with index 191 and so on. As shown in Table 2, it is observed that the starting indexes decrease from one scale to another. This is due to the fact that each scale moves across the frequency axis towards the lower center frequencies. As a result, parallel reading and writing operations are constructed, i.e. the writing operation is activated at the output sample number 40 from the FFT while the reading is activated once sample number 204 is produced from the FFT to RAM_1 and RAM_2 for 6144 cycles. There is no need to wait until all the 670 samples from the FFT are produced.
- Parallel Multipliers: this is considered since there are real and imaginary output samples from the FFT. Since the multiplication is done one sample per cycle, utilizing the parallelism feature via synthesizing two parallel multiplier (one each for handling the real and imaginary samples, respectively) reduces the number of cycles by 50%. Increasing the number of multipliers will have positive impact on reducing the required multiplication cycles. However, increasing the number of multipliers above 2 does not reduce the required number of multiplications in a linear manner. The optimal impact occurs when using 2 parallel multipliers (which is adopted in our design). Nonetheless, adopting higher number of parallel multipliers is possible at the expense of extra logic utilization while not improving the speed significantly.
- Number of Required Multiplications: originally, the number of multiplications is based on the original number of scales and number of samples in each scale. As a result, $2 \times N \times S = 729,088$ cycles are needed. Since only selected scales are used, only non-zero values per scale are used and parallel multipliers are used. As a result, the number of cycles needed for multiplication are also reduced to only 6144 clock cycles.

e: RAM OPTIMIZATION

The optimization in the RAM usage is performed as follows:

- Storing Multiplication Outputs: the needed size to store the multiplication results is reduced due to several reasons. Firstly, the optimization reduced the number of scales. Secondly, zero values from the selected scales are now eliminated. Finally, the unnecessary FFT outputs are now eliminated. The reduction obtained in RAM_3 and RAM_4 to store the multiplication outputs is:
  - Total bits size of RAM_3 and RAM_4 prior to considering the RAM optimization: $2 \times N \times S \times 28 = 20,414,464$ bits.
  - Total bits size of RAM_3 and RAM_4 post considering the RAM optimization: $2 \times 6144 \times 28 = 344,064$ bits.
  - Reduced bits size in RAM_3 and RAM_4: 20,070,400 bits.
- Storing the Wavelet Scales Values: the need for a dedicated memory initialized with the wavelet values in RAM_0 is completely eliminated. Instead, RAM_4 is initialized with the wavelet values and is also used to store the output of the multiplication. This is possible as RAM_4 is utilized only at the beginning of multiplication process. Therefore, a write during read operation is incepted at RAM_4. This is conducted to read the wavelet values initialized at RAM_4 to the input of the multiplier. At the same time, the output of the multiplier is saved again in RAM_4 in the addresses from which values have already been read. A careful control process is designed to separate the read and write operation from the same address by at least 3 cycles. In addition to that, a careful address generation of $read_{enable}$ and $write_{enable}$ signals is performed.

f: IFFT OPTIMIZATION

The IFFT module is adopted from OpenCores in Quartus Prime.Given all previous optimizations conducted over the FFT process, multiplication process, zeros and scales optimizations, inserting the correct values in the IFFT module is vital to ensure a correct IFFT process. A precise control module is designed to insert the leading and trailing zeros to the input IFFT signal at the correct locations. This requires careful design of the reading operation from RAM_3 and RAM_4 where the multiplication outputs are stored.

The block diagram in Fig. 8 depicts the optimized algorithm implementation on the FPGA. In this optimized design, the input $I(t)$ is fed to the FFT directly upon reading from an external input interface. Besides that, the wavelet coefficients are precalculated directly in the frequency domain and initialized at RAM_4, which is also used to store the multiplication outputs. In this design, only non-zero values from scale 26 to scale 50 are saved and used. Only 670 samples of the FFT outputs are needed and stored in RAM_1 and RAM_2. Furthermore, it is seen in Fig. 8 that two parallel multipliers are introduced ($Mult_1$ and $Mult_2$). The outputs of the two multipliers are stored in parallel into the optimized RAM_3 and RAM_4, which are then read into the IFFT.

The green outlined boxes in Fig. 8 are the input and output modules of the system while the numbers inside the arrows represent the sequence of the process. Besides that, the red lines connecting the control module to all other modules in the system represent all the control interface and signals.

The timeline of the different operations of the proposed optimized FFT-based CWT processor of Fig. 8 is shown in Fig. 9. This timeline shows the three major operations in the process each with a different color: 1) FFT operation is shown in cream color 2) multiplication operation is shown in green

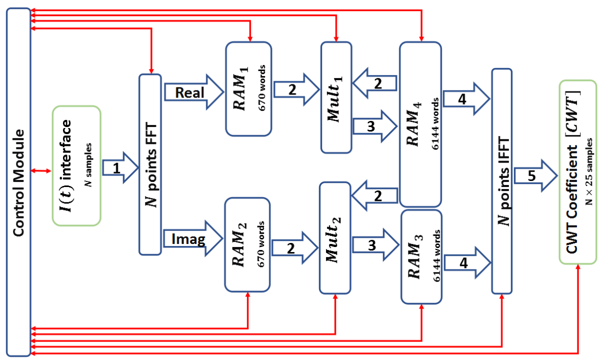


FIGURE 8. Proposed optimized FFT-based CWT processor architecture.

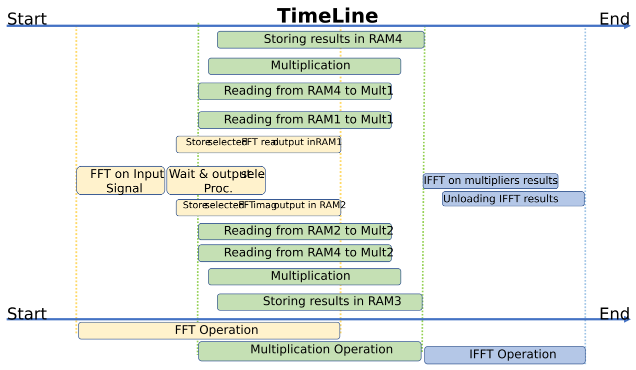

TimeLine
Start
End
Storing results in RAM4
Multiplication
Reading from RAM4 to Mult1
Reading from RAM1 to Mult1
Store selected FFT real output in RAM1
FFT on Input Signal
Wait & output sele Proc.
Store selected FFT imag output in RAM2
Reading from RAM2 to Mult2
Reading from RAM4 to Mult2
Multiplication
Storing results in RAM3
IFFT on multipliers results
Unloading IFFT results
FFT Operation
Multiplication Operation
IFFT Operation

FIGURE 9. Operation timeline of the optimized FPGA design.

color 3) IFFT operation is shown in pale-blue color. Besides that, it shows the related steps with each major operation in the same color. It can be observed that there are several parallel steps running on the same time. Moreover, the IFFT operation is launched once the complete multiplication steps are performed.

3) CONTROL MODULE

The control module functions as the brain of the proposed optimized design, generating and synchronizing all control signals from one central module, as follows:

- **FFT operation:**
  Generates and controls the timing of the $sink_{valid}$, $sink_{start}$, $sink_{end}$ and *reset* signals.
- **Control of the clock cycle counter:**
  Starts from the first activation of the FFT representing the point at which the design starts actual processing. This counter values are used at different places in the design to trigger specific signals and deactivate others. It is also used at later stage to calculate the number of cycles needed from the start of processing until the end.
- **Multiplication:**
  Generates and manages the control signals needed for the multiplication. This includes the multiplication $clock_{enable}$ signal, and the selection and synchronization of the two input signals to the multiplier. This is critical in terms of timing, so synchronization needs to be performed correctly.
- **IFFT:**
  Controls the operation of the IFFT by generating and controlling the timing of the $sink_{valid}$, $sink_{start}$, $sink_{end}$ and *reset* signals of the IFFT.
- **Memory operations:**
  Generates and controls the timing and address counting signals. These are used to generate and control the timing of the read-write operation to RAM_1, RAM_2, RAM_3 and RAM_4. These operations are also controlled in conjunction with the $read_{enable}$ and $write_{enable}$ signals of the these RAMs.

4) TIMING AND LATENCY ANALYSIS

The block diagram of Fig. 10 shows the time analysis and clock cycles requirements of the design illustrated in Fig. 7. The blue filled boxes represent the number of cycles needed for the specific operation. The letters (L) inside the arrows represent data loading process, whereas the red dashed arrows represent a wait/no operation until the specified location of the other arrow end is reached. It is apparent that the most time-consuming steps in the design are the multiplication stage and the IFFT. This is because these stages need to be repeated for the different scales of the wavelet. It is expected that the reduction in the number of scales required and the reduction of the samples in each scale strongly impact the calculation speed improvement as well as the logic utilization reduction.

In addition to that, the block diagram of Fig. 11 shows the time analysis and clock cycles requirements of the design in Fig. 8. Similar to the previous figure, the blue filled boxes represent the number of cycles required. The arrows filled with (L) represent loading process, whereas the red dashed arrows represent a wait/no operation until the specified location of the other arrow end is reached. It is also apparent that the most time-consuming steps in the design are the multiplication stage and the IFFT. This is because these stages need to be repeated for the different scales of the wavelet. The following equation shows how the total number of cycles needed to complete the algorithm processing is calculated:

$$\begin{aligned} TC &= N + (N + 84) + 39 + 203 + 6144 + N \\ &\quad + (N + 84) + (N \times \hat{S}), \end{aligned} \tag{8}$$

where TC is the total required cycles.

5) DATA SOURCES AND DATA COLLECTION TECHNIQUES

The main sources of data are obtained from recently conducted radar-based human vital sign detection experiments by a group of researchers from IMEC - Netherlands, Maastricht University - Netherlands, and IMEC - Belgium [11]. This data is comprised of reflected signals from target objects in practical experimental setups. The experiment was conducted in a 'brainstorming' office area that mimics a typical room setting. These settings contain furniture, metal shelves and objects, metal walls, personal computers (PCs), instruments, tables, sofas, a big screen, and chairs. Wi-Fi repeater stations were also active in the environments where the measurements

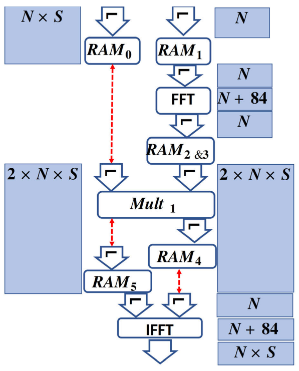


FIGURE 10. Processing clock cycles of the basic FPGA design.

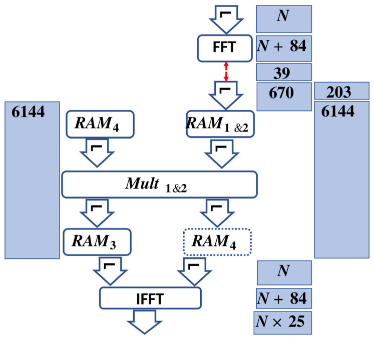


FIGURE 11. Processing clock cycles of the optimized FPGA design.

were performed. The two radar antennas, which were horizontally separated by 10 cm, were placed at 1.25 m height above a reinforced concrete floor [11].

The experimental radar set-up includes a radar module, a digital signal processor/field-programmable gate array (DSP/FPGA) board, an analog-to-digital converter (ADC) and a laptop, see Fig. 12. The radar block consists of a waveform generator based on a programmable phase-locked loop (PLL), a power divider, a low noise amplifier (LNA), a gain block, a radiofrequency (RF) mixer, a base-band filter and an amplifier. The radar waveform is generated by a PLL that is configured by the DSP/FPGA board. This signal feeds a power divider that splits it into two branches. The first output is connected to the transmitter antenna. The signal reflected from the target is received, amplified, and then mixed with a copy of the transmitted signal. On the receiving path, the signal is amplified by the LNA and gain block and then fed into the RF input of the mixer. The local oscillator (LO) input of the mixer is connected to the second output of the power divider. The baseband signal produced by the mixer is amplified, filtered, and digitized by the ADC. The DSP/FPGA manages both waveform generation and acquisition. The radar sensor is based on a linear frequency-modulated continuous-wave (FMCW) architecture. It transmits a series of chirps, separated from each other by an off interval, where no signal is transmitted. The radar sensor was designed using commercial off-the-shelf components [11].

Using described set-up, eight experiments each of two minutes were conducted. In each measurement, a volunteer is invited to breathe normally and avoid any other movements when seated on an 'acoustic sofa', hidden behind the high sound-absorbing back panels. Two different absolute distances of 2.6 m and 5.4 m were evaluated. These measurements were then repeated at the same distance, but now with the addition of moderate random body movements. The volunteers were instructed to perform four or five moderate random body movements (moderate limb movements, crossing the legs, and so on) per measurement at 11, 50, 69 and 98 s [11].

The process of taking the input data from the MATLAB and storing it in the RAMs, and of taking the output data from FPGA to MATLAB is performed via co-simulation between MATLAB and FPGA. The data transfer synchronization is controlled using a process built specifically for this purpose in our control module.

V. RESULTS AND COMPARISON

A. CWT MATLAB SIMULATION

An artificially generated input test signal with imposed noise in the signal is used as the input signal to observe the effectivness of the algorithm in detecting the unwanted movements (noise) in the signal. This signal was generated with 700 samples and a sampling period of 0.1 s. As shown in Fig 13, the signal contains four artifacts at different times. The Morlet function was used as the mother wavelet in this simulation. The filter banks were calculated using a signal length of 700 points with a sampling period of 0.1 s. The Morlet wavelet coefficients were then calculated and 66 scales was identified as the needed number of scales. Fig. 14 shows the used Morlet wavelets in frequency domain at different scales with normalized amplitude at 2. As observed from the graph, the lower scale number represents a wavelet at higher center frequency. Besides that, the shape of the wavelet becomes narrower when moving from a lower scale to a higher scale.

Utilizing the Morlet wavelet coefficients, CWT algorithm was applied on the input signal. The CWT complex coefficient matrix was generated for all the 66 scales. Using the CWT coefficients at selected scales, the locations of the unwanted artifacts in the input signal can be identified. In Fig 15, maximum magnitudes of the coefficients were plotted over time. These coefficients clearly contain information related to the noise in the signal. The figure shows the four distinct locations of the artifacts in the input signal. To locate these locations accurately, a binary mask was generated and shown in Fig 16. These identified locations in the input signal corrupted with unwanted artifacts were applied with the moving average filter. Fig 17 shows the input signal before and after the moving average was implemented on the identified artifacts locations using CWT. Clear improvement in the input signal can be seen after removing the high frequency components artifacts in the signal. This simulation case shows that the CWT can identify the locations of random

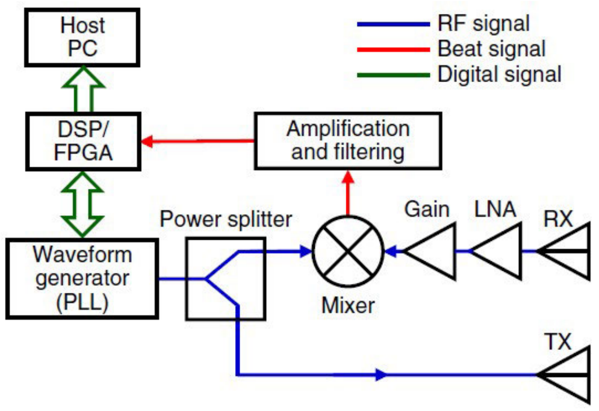


FIGURE 12. Radar components set up in [11].

movements in the signal, and hence applying moving average on these location rejects these artifacts from the signal.

After successfully using the CWT algorithm on the artificially generated signal (with noise), it is then applied on experimental radar data to further verify the algorithm results using MATLAB. The graph in Fig. 18 shows the Doppler signal of a person with unwanted random movement introduced in the signal at four locations. The experimental input signal spans time of 120 seconds with 39062 samples. This signal was segmented into individual segments with a time span of around 11 seconds each. The part of the Doppler signal with the unwanted random movement utilized in this work is shown in Fig 19. This segmentation is performed as a quick test of smaller parts of the long original signal. Fig. 20 shows the Morlet wavelet at different scales in frequency domain. These wavelets are used for implementing the CWT on the experimental test signals. As can be seen, each wavelet scale is a BPF with different center frequency. The first scales are very wide, but as the wavelet scale number increase, the BPF moves to lower frequencies and becomes narrower.

After applying CWT algorithm, the complex matrix coefficients of the CWT were generated. The maximum magnitudes of these coefficientswith scales of between 26 and 50 were calculated. These values reveal information of the location of the unwanted random movement at each location in the signal as shown in Fig. 21. It also shows the maximum magnitudes of the coefficients of the selected scales, which shows some areas with much higher values than the rest. A binary mask is then generated to identify the unwanted random movements in the Doppler signal as seen in Fig. 22. The moving average filter was then applied on the signal at the identified locations resulting in the signal presented in Fig. 23. The unwanted random movement and with high frequency components were successfully removed from the identified artifacts locations.

To further validate these results, the CWT was applied on another segment of the experimental signal where unwanted artifacts were not present. The new input signal is shown in Fig. 24 with no presence of artifacts. The algorithm was successful in building an all-zero binary mask based on the CWT

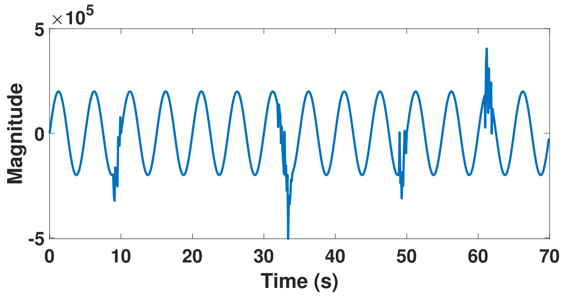


FIGURE 13. Input signal with unwanted random movements.

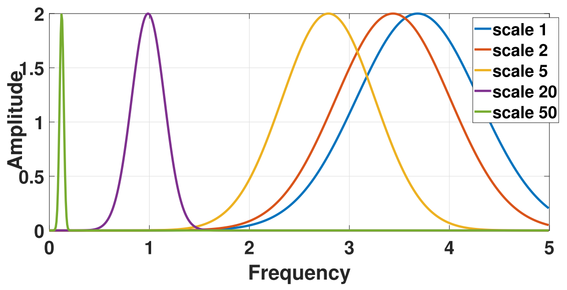


FIGURE 14. Morlet wavelet at different scales.

output coefficients. As a result, unwanted random movement was not detected, as shown in Fig. 25 and hence the moving average filter was not applied on any part of the input signal. This demonstrates that the algorithm is capable of detecting the presence of the artifacts while not generating erroneous behavior in their absence.

Furthermore, another segment of the Doppler test signal with unwanted movement at a different location was injected in the system to test its validity. This test input signal is shown in Fig. 26. The system successfully identified the location of these artifacts and applied the moving average to improve it. Fig. 27 shows very clear identification of the artifacts in the signal through the binary mask. It also shows the improvement in the signal after applying the moving average at the identified location.

The execution time of the MATLAB algorithm is around 82 ms when using a computer running on Windows10 with Intel(R) Core (TM) i7 CPU @ 1.80 GHz 1.99 GHz and 16 GB RAM.

B. CWT FPGA IMPLEMENTATION RESULTS

From the previous MATLAB simulation results, it is confirmed that the proposed algorithm is capable of locating the unwanted movements and artifacts in the Doppler signal used in the system. In this section, the results from the algorithm implementation on the FPGA are examined and validated. Fig. 28 shows the Doppler signal used in the FPGA as the input signal containing unwanted artifacts. Note that the Morlet wavelet function used in the FPGA is the one used in the MATLAB simulation from Fig.20.

Utilizing the Morlet wavelet to conduct the CWT process on the input signal, the output of the FPGA is expected to be

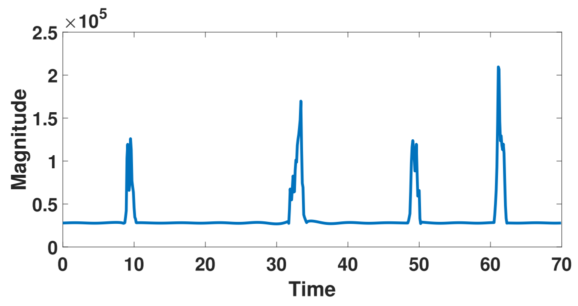


FIGURE 15. CWT coefficients maximum magnitude.

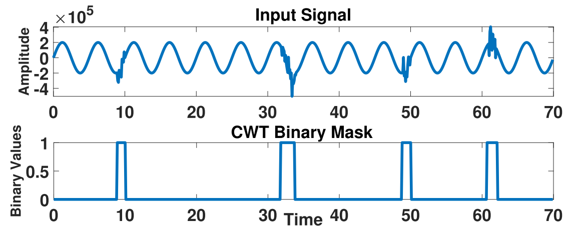


FIGURE 16. Binary masking.

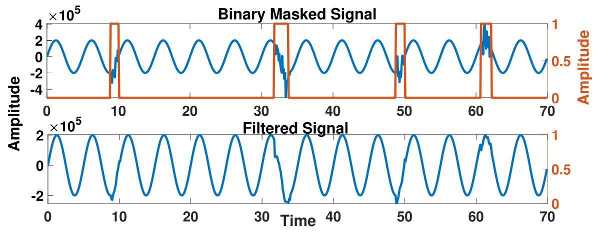


FIGURE 17. Binary masked signal and filtered signal after applying moving average.

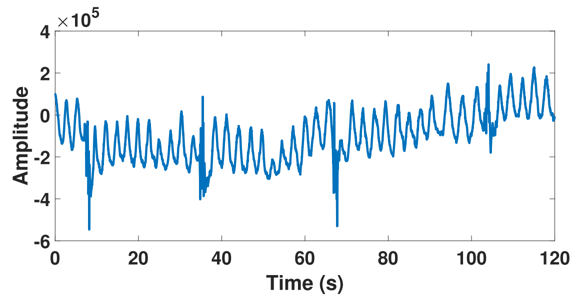


FIGURE 18. Complete Doppler signal with unwanted random movement.

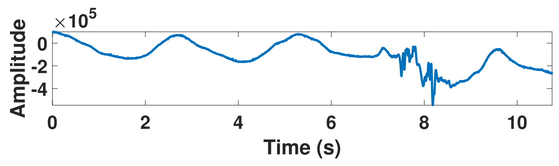


FIGURE 19. Input Doppler signal with unwanted random movement.

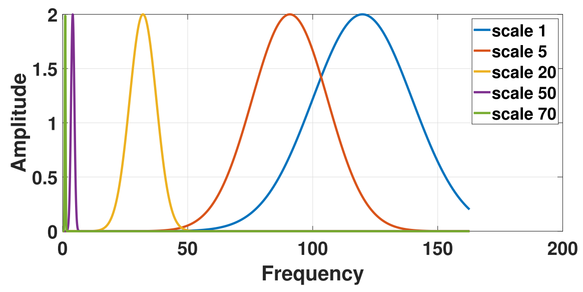


FIGURE 20. Morlet wavelet as different scales.

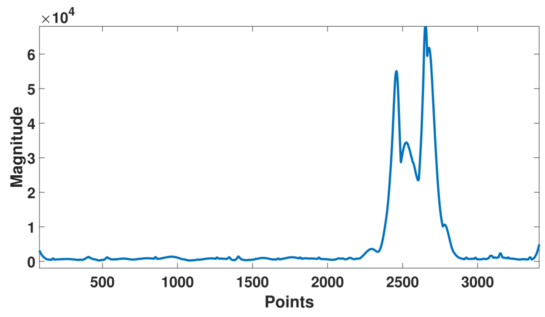


FIGURE 21. Maximum values of CWT at selected scales.

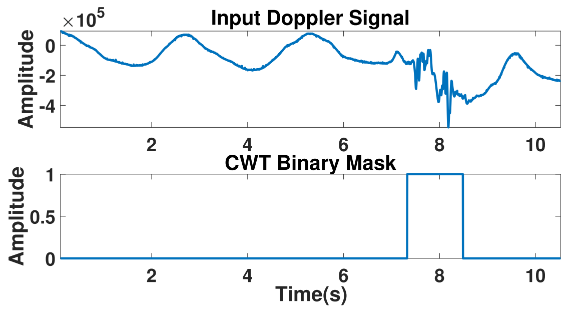


FIGURE 22. Binary masking.

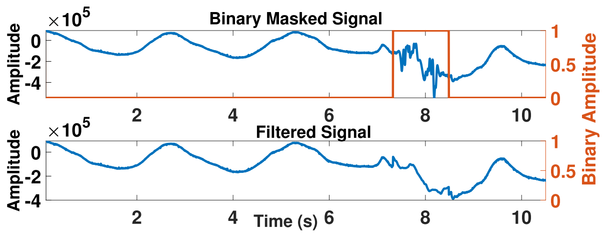


FIGURE 23. Binary masking and filtered signal after applying CWT and moving average.

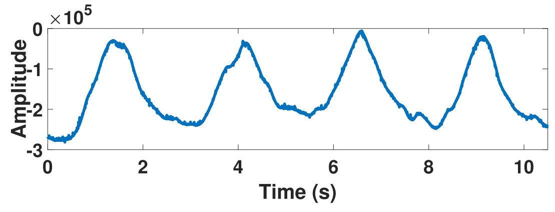


FIGURE 24. Input Doppler signal with no artifacts.

the **C**WT complex coefficients matrix of size $\hat{S} \times N$. Since our design used the scales from 26 to 50, the related CWT coefficients are presented in Fig. 29. It can be seen that these scale samples do contain information about the locations of the artifacts in the input signal. However, the information of the overall unwanted movement is scattered across the different scales. Thus the maximum values of this CWT coefficients matrix were calculated, this clustering the artifacts

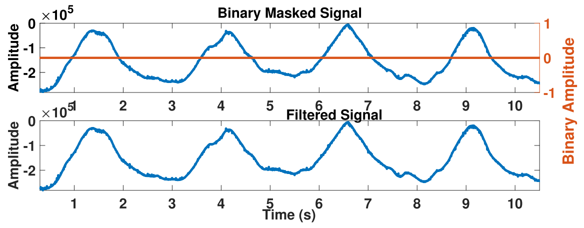


FIGURE 25. Result of binary masking and filtered signal after applying CWT and moving average.

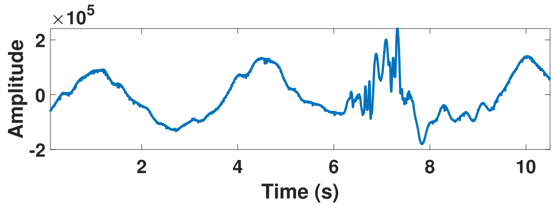


FIGURE 26. Input Doppler signal with unwanted movement.

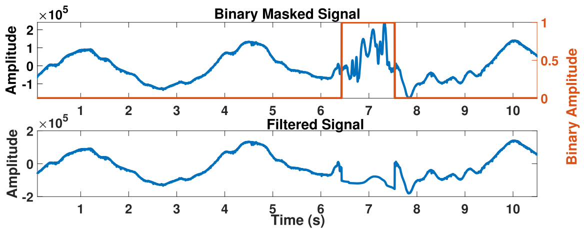


FIGURE 27. Result of binary masking and filtered signal after applying CWT and moving average.

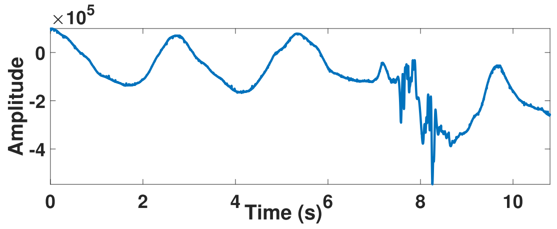


FIGURE 28. Input Doppler signal to the FPGA CWT processor.

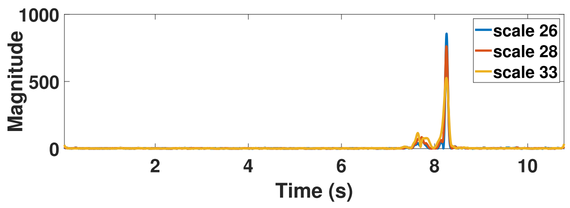


FIGURE 29. Samples of CWT output coefficients for selected scales.

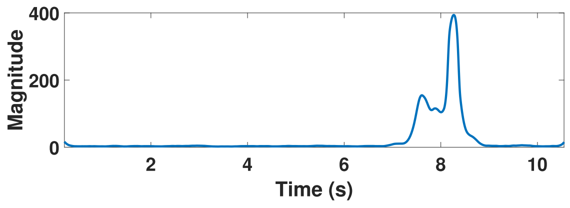


FIGURE 30. Maximum values of CWT matrix.

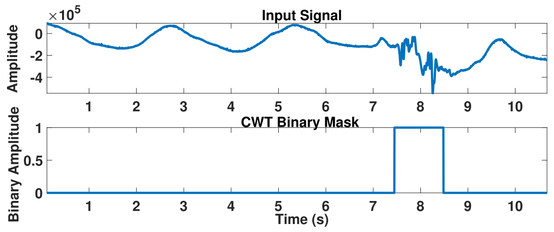


FIGURE 31. Binary mask generated from the CWT complex matrix.

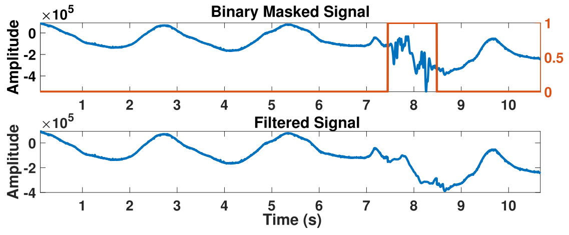


FIGURE 32. Binary masking and filtered signal after applying CWT and moving average.

information in one graph as shown in Fig. 30. In order to exactly extract the artifacts location in the signal, the binary mask shown in Fig. 31 was generated. It shows clearly that it locates the places where unwanted random movement is present in the signal. Utilizing this binary mask to apply the moving average filter on the identified locations, results in the improved signal at the unwanted random movement locations in Fig. 32.

It is clear that the output from the FPGA is capable of identifying the artifacts in the signal indicating successful implementation of the proposed optimized algorithm. Table 3 outlines the different attributes in the proposed CWT processor FPGA architecture pre and post optimization. One of the main attributes is the number of scales, which was reduced from S to $\hat{S}$. Moreover, the memory bit requirement was reduced from 23,576,576 bits to 3,708,646 bits indicating major speed and resources improvement. For example, the required multiplication cycles were reduced from 729,088 to N and the total processing cycles needed were reduced from $1,478,824$ to $125,307$.

Results of the optimized FPGA implementation are tabulated in Table 4. It shows that the maximum frequency the design can achieve based on the critical path is 74.1 MHz and therefore a processing time of 1.69 ms is achieved. Besides that, the proposed design utilizes 42% of the FPGA logic elements and 72% of RAM bit locations on an Altera Max10 FPGA, specifically the 10M50DAF484C7G device. The power consumption of the optimized FPGA design is measured using the power analyzer tool in Quartus prime and is found to be 363 mW.

The details of the processing cycles performance related to each operation in the design pre optimization are presented

TABLE 3. Summary of design attributes pre and post optimization.

Attribute/Results	Pre Optimization	Post Optimization
Input signal samples	N	N
Wavelet samples per scale	N	N
Number of scales	S	$\hat{S}$
Number of bits per signal sample	20	20
Number of bits per wavelet sample	8	8
Memory requirements	$N \times 20$ bits for RAM_1 $N \times 20$ bits for RAM_2 $N \times 20$ bits for RAM_3 $N \times S \times 8$ bits for RAM_0 $N \times S \times 28$ bits for RAM_4 $N \times S \times 28$ bits for RAM_5 Total is 23,576,576 bits	670×20 bits for RAM_1 670×20 bits for RAM_2 6144×28 bits for RAM_3 6144×28 bits for RAM_4 Total is 3,708,646 bits
Memory requirements improvement	98.4% reduction in memory bits requirements	
Required multiplication cycles	$2 \times S \times N = 729,088$	N
Multiplication cycles improvement	99.2% reduction in multiplication cycles	
Total processing cycles required	$1,478,824$ cycles	$125,307$ cycles
Cycles Improvement	91.53% reduction in the required cycles	

TABLE 4. FPGA results of optimized CWT processor architecture.

Result Item	Value
Maximum Frequency F_{Max}	74.1 MHz
Total Usage of RAM bits locations in the FPGA	72%
Total Usage of logic elements in the FPGA	42%
Total computation time at F_{Max} post optimizations	$125307 cycles * 1/F_{Max} = 1.69 ms$
Total power consumption	363 mW

TABLE 5. Details of design timing performance pre optimizations.

Activity	Clock Cycles
Loading RAM_0	$N \times S$
FFT Loading	N
FFT Processing	$N + 84$
FFT output to RAM_2 and RAM_3	N
Multiplications	$2 \times N \times S$
IFFT Loading	N
FFT Processing	$N + 84$
FFT outputs	$N \times S$
Total	1,478,824

TABLE 6. Details of design timing performance post optimizations.

Activity	Clock Cycles
FFT Loading	N
FFT Processing	$N + 84$
Waiting Specific FFT output samples	39
Loading to RAM_1 and RAM_2	203
Multiplications	6144
IFFT Loading	N
FFT Processing	$N + 84$
FFT outputs	$N \times \hat{S}$
Separation and Miscellaneous	8
Total Clock Cycles	125,307

in Table 5, whereas Table 6 summarizes the details of the processing cycles performance related to each operation in the optimized design. From Table 3, 4, 5 and 6, it is clear that there is significant (91.53%) reduction in the number of clock cycles required for processing the design. This is due to the collective implementation of the multiple optimization approaches outlined in this paper. In addition to that, the significant reduction in the memory resources in the optimized design is also due to the same reason. A total of 98.4% reduction in the memory bits requirements is achieved when comparing the basic architecture to the optimized one.

Table 7 compares the proposed optimized work in this article and other state-of-the-art works found in the literature. It is very difficult to establish fair comparison as each design uses different processing platform, CWT algorithm structure, signal length and number of scales in the wavelet function. Nonetheless, performance comparison of state-of-the-art designs with our proposed work is done considering these different factors. Firstly, the work in [32] implements CWT algorithm on FPGA to detect the R signal from the ECG signal. The focus of this work was to prove the concept of utilizing the CWT for such application and to achieve higher accuracy. Therefore, the processing time and speed, and the resources utilized to implement the algorithm on FPGA were not explicitly reported. Next, the work in [28] is the most similar design with the proposed work. This work was developed for feature extraction of ERP from EEG signal. When comparing the number of cycles required for [28] with the proposed design, we find that it needs 76,012 cycles while our design needs 125,307 cycles. However, it is important to note that the application presented in [28] is different form this work, and therefore different signal length and scales are required. The input signal length used in [28] is $N/4$ and our input signal length is N, which is four times longer than the input signal of [28]. This indicates that if N length was used in [28] it would require more or less 4 times the current number of cycles required while using a signal with $N/4$ length. Yet, the proposed design required less than double the cycles needed in [28]. To be more specific, if the proposed design used a signal length of $N/4$, it would require around 31,326 cycles, which is 58.8% faster than [28]. As a result, the proposed design would have a processing time

TABLE 7. Comparison with other works.

Comparison	[28]-2016	[32]-2019	[30]-2019	Our work
CWT Algorithm Method	FFT-Based	CWT with Spline	2D FFT based	FFT-Based
Application	ERP feature extraction	R signal detection from ECG signal	phase extraction of fringe pattern	unwanted random movement detection in vital signal detection
Signal Length	1024	NA	256×256	4096
Processing Technology Used	FPGA Spartan 3AN	Xilinx Artix-7 XC7A35T-ICPG236C	Altera Cyclone IV EP4CE115F29	FPGA Altera Max10
Mother Wavelet	Morlet	B-Splines	Morlet	Morlet
Original Number of Scales	37	Not Reported	1	89
Final Number of Scales	21	Not Reported	1	25
Processing Clock Cycles	76,012	Not Reported	NA	125,307
Maximum Frequency	80 MHz (with SRAM)	Not Reported	200 MHz	74.1 MHz
Processing Time	0.57 ms (no SRAM, 133 MHz)	Not Reported	8 ms	1.69 ms on FPGA 82 ms using MATLAB in laptop with Intel, i7-8550U CPU @ 1.80 GHz 1.99 GHz and 16 GB RAM

of 0.42 ms compared to 0.57ms achieved by [28]. This is largely due to the various optimizations implemented in the design to eliminate unnecessary calculation cycles, conduct parallel processing and to focus on optimizing the most computationally intensive parts of the algorithm. In addition, the design in [30] implemented 2D FFT based CWT to analyze fringe patterns. It achieved very good processing time of 8 ms compared to the length of the signal. The reason of this is due to the use of only one scale in the wavelet, compared to $\hat{S}$ scales in the proposed design. As aforementioned, the factor determining the selection of scales to be used is application dependent. In our case, 25 scales were required to be used to cover wider range of expected unwanted frequencies in the Doppler signal. However, when the application requires narrow range of frequencies, it is suitable to use few scales that cover the specific frequency range.

It is not easy to compare the utilization of resources in each FPGA design unless the same FPGA is used. However, this work performed advanced optimizations on the resources to ensure it is implemented on low-end, cost-effective FPGA such as the Max 10 from Altera. Among these optimizations, eliminating the need of the dedicated memory to store the wavelet values. This was done by re-utilizing RAM_4 for that purpose and also for storing the multiplication results. Besides that, resource optimization was performed by reducing the memory required to store the outputs of the FFT from N to only 670 locations. In addition, reduction of the memory requirements to store the multiplication results was also part of the resources optimization. It is believed that these techniques position the proposed work to be competitive with other works from the prospective of resource utilization.

VI. CONCLUSION

This work realizes CWT processor on FPGA architecture implemented with several optimization approaches to make it suitable for high-performance data processing applications. It is demonstrated in the results that the proposed FPGA design achieves high calculation speed performance compared to state of the art designs as well as reduced resource utilization. In addition, there is significant improvement of the hardware execution time over MATLAB execution time (48 times faster). This signifies the importance of and benefit of utilizing optimized hardware processing implementations on platforms such as FPGA to process complex algorithms.

REFERENCES

[1] K. Peng, J. Jau, J. Li, S. Member, and C. Chen, "Single-antenna Doppler radars using self and mutual injection locking for vital sign detection," *IEEE Trans. Microw. Theory Techn.*, vol. 59, no. 12, pp. 3577–3587, Nov. 2011.

[2] F. Khan and S. Cho, "A detailed algorithm for vital sign monitoring of a stationary/non-stationary human through IR-UWB radar," *Sensors*, vol. 17, no. 2, p. 290, Feb. 2017.

[3] P. Li and N. Huo, "A portable 24 GHz Doppler radar system for distant human vital sign monitoring," in *Proc. 5th Int. Conf. Inf. Sci. Control Eng. (ICISCE)*, Jul. 2018, pp. 1050–1052.

[4] M.-C. Tang, F.-K. Wang, and T.-S. Horng, "Single self-injection-locked radar with two antennas for monitoring vital signs with large body movement cancellation," *IEEE Trans. Microw. Theory Techn.*, vol. 65, no. 12, pp. 5324–5333, Dec. 2017.

[5] C. H. Hsieh, Y. F. Chiu, Y. H. Shen, T. S. Chu, and Y. H. Huang, "A UWB radar signal processing platform for real-time human respiratory feature extraction based on four-segment linear waveform model," *IEEE Trans. Biomed. Circuits Syst.*, vol. 10, no. 1, pp. 219–230, Feb. 2016.

[6] H. Suarez and Y. R. Zhang, "System-on-chip architecture and validation for real-time transceiver optimization: APC implementation on FPGA," in *Proc. SPIE, Int. Soc. Opt. Eng.*, vol. 9461, 2015, pp. 22–36.

[7] V. L. Petrović, M. M. Janković, A. V. Lupšić, V. R. Mihajlović, and J. S. Popović-Bošović, "High-accuracy real-time monitoring of heart rate variability using 24 GHz continuous-wave Doppler radar," *IEEE Access*, vol. 7, pp. 74721–74733, 2019.

[8] Y. Quan, Y. Li, X. Gao, and M. Xing, "FPGA implementation of real-time compressive sensing with partial Fourier dictionary," *Int. J. Antennas Propag.*, vol. 2016, pp. 1–12, Dec. 2016.

[9] Y. Yang and A. E. Fathy, "Development and implementation of a real-time see-through-wall radar system based on FPGA," *IEEE Trans. Geosci. Remote Sens.*, vol. 47, no. 5, pp. 1270–1280, May 2009.

[10] A. Bin Obadi, P. J. Soh, O. Aldayel, M. H. Al-Doori, M. Mercuri, and D. Schreurs, "A survey on vital signs detection using radar techniques and processing with FPGA implementation," *IEEE Circuits Syst. Mag.*, vol. 21, no. 1, pp. 41–74, Sep. 2021.

[11] M. Mercuri, I. R. Lorato, Y.-H. Liu, F. Wieringa, C. Van Hoof, and T. Torfs, ''Vital-sign monitoring and spatial tracking of multiple people using a contactless radar-based sensor,'' *Nature Electron.*, vol. 2, pp. 252–262, Jun. 2019.

[12] J. Tu, T. Hwang, and J. Lin, ''Respiration rate measurement under 1-D body motion using single continuous-wave Doppler radar vital sign detection system,'' *IEEE Trans. Microw. Theory Techn.*, vol. 64, no. 6, pp. 1937–1946, Jun. 2016.

[13] C. Li and J. Lin, ''Random body movement cancellation in Doppler radar vital sign detection,'' *IEEE Trans. Microw. Theory Techn.*, vol. 56, no. 12, pp. 3143–3152, Dec. 2008.

[14] C. Gu, G. Wang, Y. Li, T. Inoue, and C. Li, ''A hybrid radar-camera sensing system with phase compensation for random body movement cancellation in Doppler vital sign detection,'' *IEEE Trans. Microw. Theory Techn.*, vol. 61, no. 12, pp. 4678–4688, Dec. 2013.

[15] X. Hu and T. Jin, ''Short-range vital signs sensing based on EEMD and CWT using IR-UWB radar,'' *Sensors*, vol. 16, no. 12, p. 2025, Nov. 2016.

[16] P. Y. Sevilla-Camacho, J. B. Robles-Ocampo, J. C. Jauregui-Correa, and D. Jimenez-Villalobos, ''FPGA-based reconfigurable system for tool condition monitoring in high-speed machining process,'' *Measurement*, vol. 64, pp. 81–88, Mar. 2015.

[17] D. Baccar and D. Söffker, ''Wear detection by means of wavelet-based acoustic emission analysis,'' *Mech. Syst. Signal Process.*, vols. 60–61, pp. 198–207, Aug. 2015.

[18] V. M. Valenzuela-De La Cruz, M. A. Gurrola-Navarro, C. A. Bonilla-Barragan, and R. Carrasco-Alvarez, ''Delay in QRS complex detection using the wavelet transform for real-time applications,'' in *Proc. 14th Int. Conf. Electr. Eng., Comput. Sci. Autom. Control (CCE)*, no. 2, Oct. 2017, pp. 1–6.

[19] D. Baccar and D. Söffker, ''Identification and classification of failure modes in laminated composites by using a multivariate statistical analysis of wavelet coefficients,'' *Mech. Syst. Signal Process.*, vol. 96, pp. 77–87, Nov. 2017.

[20] R. Kumar and H. O. Bansal, ''Hardware in the loop implementation of wavelet based strategy in shunt active power filter to mitigate power quality issues,'' *Electr. Power Syst. Res.*, vol. 169, pp. 92–104, Apr. 2019.

[21] S.-J. Huang and C.-W. Lu, ''Enhancement of digital equivalent voltage flicker measurement via continuous wavelet transform,'' *IEEE Trans. Power Del.*, vol. 19, no. 2, pp. 663–670, Apr. 2004.

[22] K. Sun, X. Pan, and L. Ping, ''A reconfigurable computing engine for wavelet transforms,'' in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, Jun. 2007, pp. 1–4.

[23] S. Cheng, C.-H. Tseng, and M. Cole, ''Efficient and effective VLSI architecture for a wavelet-based broadband sonar signal detection system,'' in *Proc. 14th IEEE Int. Conf. Electron., Circuits Syst.*, Dec. 2007, pp. 593–596.

[24] S. Cheng, C. H. Tseng, and M. Cole, ''A novel FPGA implementation of a wideband sonar system for target motion estimation,'' in *Proc. Int. Conf. Reconfigurable Comput. (FPGAs ReConFig)*, 2008, pp. 349–354.

[25] P. Ghorbanian, A. Ghaffari, A. Jalali, and C. Nataraj, ''Heart arrhythmia detection using continuous wavelet transform and principal component analysis with neural network classifier,'' *Comput. Cardiol.*, vol. 37, pp. 669–672, Dec. 2010.

[26] Y. T. Qassim, T. Cutmore, D. James, and D. Rowlands, ''FPGA implementation of Morlet continuous wavelet transform for EEG analysis,'' in *Proc. Int. Conf. Comput. Commun. Eng. (ICCCE)*, Jul. 2012, pp. 59–64.

[27] Y. T. Qassim, T. Cutmore, and D. Rowlands, ''Multiplier truncation in FPGA based CWT,'' in *Proc. Int. Symp. Commun. Inf. Technol. (ISCIT)*, Oct. 2012, pp. 947–951.

[28] Y. T. Qassim, T. R. Cutmore, and D. D. Rowlands, ''Optimized FPGA based continuous wavelet transform,'' *Comput. Elect. Eng.*, vol. 49, pp. 84–94, Jan. 2016.

[29] A. Abid, ''Fringe pattern demodulation using the one-dimensional continuous wavelet transform: Field-programmable gate array implementation,'' *Appl. Opt.*, vol. 52, no. 7, pp. 1468–1471, 2013.

[30] L. Mao, J. Ma, H. Chen, Y. Ma, S. Chen, and C. Xu, ''2D-CWT IP core design and implementation for fringe pattern analysis,'' *IOP Conf. Series, Mater. Sci. Eng.*, vol. 569, no. 3, Jul. 2019, Art. no. 032071.

[31] F. M. Nez-Suarez and C. Alvarado-Serrano, ''VHDL module for the r wave detection in real time using continuous wavelet transform,'' in *Proc. 16th Int. Conf. Electr. Eng., Comput. Sci. Autom. Control (CCE)*, Sep. 2019, pp. 1–5.

[32] F. Martinez-Suarez and C. Alvarado-Serrano, ''Prototype of an ambulatory ECG monitoring system with R wave detection in real time based on FPGA,'' in *Proc. 16th Int. Conf. Electr. Eng., Comput. Sci. Autom. Control (CCE)*, Sep. 2019, pp. 1–5.

[33] A. H. Gharekhan, S. Arora, P. K. Panigrahi, and A. Pradhan, ''Distinguishing cancer and normal breast tissue autofluorescence using continuous wavelet transform,'' *IEEE J. Sel. Topics Quantum Electron.*, vol. 16, no. 4, pp. 893–899, Aug. 2010.

[34] C. Gu, ''VFTO spectrum analysis method based on continuous wavelet transform,'' in *Proc. 10th Int. Conf. Commun., Circuits Syst. (ICCCAS)*, Dec. 2018, pp. 369–372.

[35] S. M. K. Zaman, H. U. M. Marma, and X. Liang, ''Broken rotor bar fault diagnosis for induction motors using power spectral density and complex continuous wavelet transform methods,'' in *Proc. IEEE Can. Conf. Electr. Comput. Eng. (CCECE)*, May 2019, pp. 1–4.

[36] P. Shegokar and P. Sircar, ''Continuous wavelet transform based speech emotion recognition,'' in *Proc. 10th Int. Conf. Signal Process. Commun. Syst. (ICSPCS)*, Dec. 2016, pp. 1–8.

[37] M. S. Diab and S. A. Mahmoud, ''Continuous wavelet transform OTA-C band pass filter on field programmable analog arrays,'' in *Proc. Adv. Sci. Eng. Technol. Int. Conf. (ASET)*, Feb. 2020, pp. 1–5.

[38] Z. Jianghong, ''Continuous Marr wavelet transform based on ladder structure log-domain filter,'' in *Proc. 3rd Int. Conf. Intell. Syst. Design Eng. Appl.*, Jan. 2013, pp. 470–472.

[39] A. B. Stepanov, ''Neural network model of wavelets for the continuous wavelet transform,'' in *Proc. Int. Conf. Comput. Technol. Phys. Eng. Appl. (ICCTPEA)*, Jun. 2014, pp. 177–178.

[40] A. Belkhou, A. Achmamad, and A. Jbari, ''Classification and diagnosis of myopathy EMG signals using the continuous wavelet transform,'' in *Proc. Sci. Meeting Elect.-Electron. Biomed. Eng. Comput. Sci. (EBBT)*, Apr. 2019, pp. 1–4.

[41] X. Wang, B. Wang, and W. Chen, ''The second-order synchrosqueezing continuous wavelet transform and its application in the high-speed-train induced seismic signal,'' *IEEE Geosci. Remote Sens. Lett.*, vol. 18, no. 6, pp. 1109–1113, Jun. 2021.

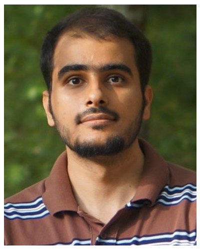
AMEEN BIN OBADI received the B.Sc. degree (Hons.) in electrical engineering and the M.Sc. degree in electrical engineering (analog and digital electronics) from the King Fahd University of Petroleum and Minerals (KFUPM), Saudi Arabia, in 2010 and 2014, respectively. He is currently pursuing the Ph.D. degree with Universiti Malaysia Perlis (UniMAP), Malaysia. He worked as an IP Commercialization Executive at Dhahran Techno-Valley Company and as a Manager of technology venture development at RPDC. His experience in this area ranges across managing a portfolio of technologies including evaluation, marketing, technology advancements, licensing and/or spin-outformation. He backed up his experience in technology commercialization with training sessions in Europe and the U.S. provided by ASTP-PROTON and AUTM. He started his career at the Japanese company JGC Gulf in the oil and gas EPC industry as an Electrical Engineer in Khobar. He then worked at Schneider Electric in the energy division handling Saudi Electricity Company contractors' portfolio in the main office in Riyadh.

MEDIEN ZEGHID received the Ph.D. degree in information and communication, sciences and technologies from the University of South Brittany, Lorient-France, in 2011. He is currently an Assistant Professor with the Department of Computer Engineering and Networks, Prince Sattam Bin Abdulaziz University. His research interests include information security, architectural synthesis for the crypto-systems, image and video coding, and optical communication.

PHAK LEN EH KAN (Member, IEEE) received the Bachelor of Electrical Engineering degree in electronic from UTM, the M.Sc. degree in information technology from UUM, and the Ph.D. degree in computer engineering from the UniMAP–University of Birmingham, U.K. He worked as an Engineer/a Senior Engineer at Multinational Companies–Electronic Industries for six years before joined the University Malaysia Perlis (UniMAP) as a Lecturer, in January 2003. He is currently an Associate Professor at the Faculty of Electronic Engineering Technology, UniMAP. He is also the Head of the High Performance Computing Group under CoE Advanced Computing, UniMAP. He has published over 110 articles in international journals and proceedings Scopus indexed. His research interests include reconfigurable computing and FPGA, digital design and embedded systems, digital and image processing, system on chip (SoC), smart systems, and the IoT. He is currently a Chartered Engineer (U.K.), a Professional Technologist (MBOT), a Graduate Member of BEM, and a member of BCS and IACSIT.

PING JACK SOH (Senior Member, IEEE) was born in Sabah, Malaysia. He received the bachelor's and master's degrees from Universiti Teknologi Malaysia, and the Ph.D. degree from KU Leuven, Belgium. He is an Associate Professor at the Centre for Wireless Communications (CWC), University of Oulu, Finland. He started his career as a Test Engineer and a Research and Development Engineer, from 2002 to 2004 and from 2005 to 2006, respectively. He was then a Lecturer with Universiti Malaysia Perlis (UniMAP), from 2006 to 2009, before moving to KU Leuven, as a Research Assistant from 2009 to 2013 and a Postdoctoral Research Fellow from 2013 to 2014. Since 2014, he was a Research Affiliate with the ESAT-WAVECORE Research Division. Before moving to Finland, he was a Senior Lecturer (2014–2017) and an Associate Professor until 2021 at UniMAP. Within UniMAP, he was formerly the Deputy Director at the Centre for Industrial Collaboration (2007–2009), the Deputy Dean of the University's Research Management and Innovation Center (RMIC) (2014–2017), and the Head of the Advanced Communication Engineering (ACE) Research Centre (2020). He is currently the Coordinator for the ''Devices and Circuit Technology'' strategic research area within the 6G flaghip program. He has published more than 300 articles in journals and international conference proceedings, besides four book chapters. His research interests include antennas and their applications in wearables/body area communication, metasurfaces and reflectors, 5G/6G communications, compact satellites, EM safety and absorption, and wireless techniques for healthcare. He was a recipient of the URSI Young Scientist Award in 2015, the IEEE MTT-S Graduate Fellowship for Medical Applications in 2013, and the IEEE AP-S Doctoral Research Award in 2012. He was also the Second Place Winner of the IEEE Presidents' Change the World Competition in 2013. Two of his (co)authored journals were awarded the CST University Publication Award in 2012 and 2011. He is a Chartered Engineer registered with the U.K. Engineering Council and a member of the IET and URSI. He also volunteers in the IEEE MTT-S Education Committee. He is also an Associate Editor of the *International Journal of Numerical Modeling: Electronic Networks, Devices and Fields* (Wiley).

MARCO MERCURI (Senior Member, IEEE) received the B.S. and M.S. degrees in electronic engineering from the University of Calabria, Italy, in 2006 and 2009, respectively, and the Ph.D. degree in electronic engineering from KU Leuven, Belgium, in 2015. He was a Senior Researcher with the Holst Centre/imec–The Netherlands, The Netherlands, from 2015 to 2021, working on biomedical applications of microwave/RF. He is currently a Researcher at the University of Calabria. He is the author and the coauthor of over 50 publications in peer-reviewed journals, conferences, and books. He holds several patents on radars for biomedical applications. His research interests include microwave/RF radar systems and algorithms, contactless health monitoring, biomedical applications, embedded systems, and wireless sensor networks. He was a recipient of the 2013 IEEE MTT-S Graduate Fellowship Award and the ISSCC 2019 Demonstration Session Certificate of Recognition. He received the second place both to the 2013 IEEE President's Change the World Competition (PCW) and to the 2013 IEEE MTT-S YouTube/YouKu Video Competition.



OMAR ALDAYEL (Member, IEEE) received the B.S. degree from King Saud University, Riyadh, Saudi Arabia, the M.S. degree from KTH– the Royal Institute of Technology, Stockholm, Sweden, and the Ph.D. degree in electrical engineering from The Pennsylvania State University, University Park, PA, USA, in 2007, 2011, and 2017, respectively. He is currently an Assistant Professor at the Department of Electrical Engineering, King Saud University. He worked at King Saud University, as a Lecturer, from 2011 to 2013. His research interests include radar signal processing, detection and estimation, and convex optimization.