Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2018) 26: 1820 – 1832

doi:10.3906/elk-1708-117

Hardware implementation and comparison of displacement retrieval algorithms for a laser diode-based optical feedback interferometric sensor

Ghazi REHMAN[1], **Usman ZABIT**[2], **Muhammad Obaid ULLAH**[1,*]
[1]Department of Electrical Engineering, Faculty of Electronics and Electrical Engineering, University of Engineering and Technology, Taxila, Pakistan
[2]Department of Electrical Engineering, National University of Sciences and Technology, Islamabad, Pakistan

Received: 10.08.2017 • **Accepted/Published Online:** 03.04.2018 • **Final Version:** 27.07.2018

Abstract: Optical feedback interferometer (OFI) lasers, also called self-mixing (SM) lasers, have been widely explored over the last couple of decades due to their low cost, compactness, and self-aligned nature and they provide a very good solution for measurements of displacement, vibration, distance, velocity, etc. The SM effect takes place when a part of the laser beam is fed back to the active laser cavity after reflecting from the target. The reflected beam interferes with the emitted beam and hence the optical and spectral characteristics of the laser get changed. To retrieve the vibration or displacement signal of the target from the SM signal, different postprocessing algorithms have been proposed, such as the phase unwrapping method (PUM). The first step of the PUM leads to the coarse estimation of the laser phase and the final step is an iterative joint estimation of 2 parameters, namely laser coupling coefficient C and linewidth enhancement factor α. To make this algorithm applicable for real-time measurements, parallel joint estimation for a wide range of C and α values needs to be done. In this research, 3 algorithms, namely PUM, direct fringe unwrapping (DFU), and improved DFU (IDFU) were tested for FPGA implementation by using Verilog HDL (hardware description language) so that more precise and real-time vibration and displacement signals of targets could be extracted from the SM sensor in an embedded systems environment. These algorithms were developed using Verilog HDL for implementation on the Xilinx Spartan-3 Xcs400-FG320 development board. Our designed IDFU algorithm performed 0.492 times better than the parallel PUM algorithm in maximum clock frequency and 1.53 and 1.21 times better than the PUM in slice registers and LUT utilization of hardware resources, respectively. The designed DFU algorithm can operate 1.355 times better than IDFU in maximum clock frequency and 25.34 and 14.25 times better than IDFU in slice registers and LUT utilization of hardware resources, respectively.

Key words: Self-mixing, laser sensors, real time, phase unwrapping method, direct fringe unwrapping, improved direct fringe unwrapping, field-programmable gate array

1. Introduction

The optical feedback interferometer (OFI) or self-mixing (SM) [1,2] technique has been frequently used during last 2 decades for measurement of displacement [3], velocity [4], vibration [5,6], acceleration [7], distance [8,9], and flow [10] as the subsequent sensor is of low cost, simple, self-aligned, and compact [11]. The SM phenomenon arises when a laser beam enters the active laser cavity after partially bouncing from a target. The reflected beam interferes with the emitted beam and hence the optical and the spectral characteristics of the laser are

*Correspondence: obaid.ullah@uettaxila.edu.pk

changed. The characteristics or shape of the resultant SM signal rely on the laser linewidth enhancement factor α (also known as Henry's factor [12]) as well as on the intensity of optical feedback coupling factor C (also designated as Acket's parameter [13]). A simple SM sensor of $\lambda/2$ precision can be developed by counting the transition or fringes contained by the SM signal [8][5-7]. To get precision better than $\lambda/2$, the shape of fringes contained within the SM signal are studied to extract more information. The SM fringe shape is a good indicator of actual target movement[2, 4, 5]. By exploiting this idea, various displacement measurement SM techniques with higher precision have been proposed.

There are 2 principal steps involved in the phase unwrapping method (PUM) [14]. The first step leads to the coarse estimation of the phase of laser output under feedback once all SM interferometric fringes are correctly detected [8,11,15]. The second step of the PUM involves a joint estimation of parameters C and α. This is a signal processing step and comprising an optimization routine. The precision of measurement of experimental target movement provided by the PUM is of the order of $\lambda/20$ [14]. A detailed study of the PUM indicates that the first step of the PUM is relatively easy to implement and yields a staircase-shaped phase signal by adding/subtracting 2π [16]. The PUM's second step, however, is comparatively more complex to implement, mainly because of the involvement of an iterative minimization routine .[3]. Thus, this iterative minimization routine step makes the iterative PUM very slow with very high latency and thus not attractive for real-time applications. To further improve the measurement precision of the PUM approach, a local phase inversion needs to be detected and corrected by identifying the location of the inverted phase based on peak and valley location within every SM interferometric fringe [3,17,18].

On the other hand, a real-time algorithm was proposed that is based on direct fringe unwrapping (DFU) [19]. In the DFU technique, an interesting point to note is that only 2 consecutive samples of the SM signal are required to be processed at a given time to extract the displacement of the target. Therefore, output can be updated with only one sample delay. A successive derivative of the SM signal is taken that is based on subtracting the previous sample from the current sample, i.e. $P(t) - P(t-1)$, and then comparing the result with the threshold value in order to detect the fringes contained by the SM signal [20]. Simple addition of all these fringes results in a staircase signal, then by adding normalized $P(t)$ it presents sample results in the unwrapped displacement signal $D(t)$. Consequently, every arriving sample of SM signal results in a subsequent output sample of displacement, with a very small latency .[8]. Thus, the DFU algorithm is faster than other algorithms to extract vibration and displacement signals from SM signals [21]. That being said, it is important to state that input experimental SM signals usually contain amplitude variations as a function of optical feedback level. Thus, normalization of input SM signals is essential for correct working of DFU. A normalization block has therefore been implemented in the present work even if it results in an increase in latency of the overall algorithm. Note that the fast nature of DFU is enabled due to removal of the analytical solution (through conjoint parameter estimation) carried out in the PUM. As a result, its measurement performance is strongly dependent on the operating C value. Thus, simulations have indicated how precision of DFU varies as a function of C from $\lambda/5$ for $C = 3.75$ to $\lambda/18$ for $C = 1.75$.

Further studies of DFU suggest that the precision of the DFU technique can be further improved, so we propose a new technique called improved direct fringe unwrapping (IDFU). In IDFU, we first globally normalize the SM signal. Then discontinuities of SM signal fringes are detected. Based on these detected discontinuities, the SM signal is segmented into sections where each section is delimited from other sections by its surrounding discontinuities. To normalize the SM signal, peak and valley locations are found in each piece of

the segment. After the normalization of the SM signal, the phase signal is then unwrapped. As per the results presented later in this paper, the measurement precision of target movement provided by IDFU is of the order of $\lambda/9$.

Rapid growth and improvement in density and the performance per watt of FPGAs has been witnessed in the last decade. This development subsequently made the design of high-performance and high-precision sensors possible, which can perform real-time tasks on a single chip. An FPGA-based design for an optical feedback self-mixing interferometry system (OFSMI) displacement sensing system was proposed in [22]. The proposed design of the FPGA unit performs noise reduction, signal peak detection, and impulse magnitude tracking. It uses median filters to remove sparkle-like noise and bandpass ?lters to reduce slow time-varying fluctuations and high frequency noise. Results of the OFSMI architecture show that the FPGA-based signal processing unit can achieve fast and reliable displacement sensing. Another FPGA-based ?ltering and normalization process for SM signals was proposed in [23] to achieve real-time and better quality sensing. The FPGA preprocessing stage comprises a denoising unit and a normalization unit. Depending upon the features of noise contained in the SM signal, a combination of a wavelet transform-based filter and a median ?lter is used in the denoising part. The normalization stage retains the SM signal in the range [–1, 1]. Hardware cosimulation carried out on a Xilinx Spartan-3E board verified the performance of this FPGA-based preprocessing method. A phase estimation circuit was developed in [24], which was implemented on FPGA hardware and experimentally tested using a new state-of-the-art data acquisition system developed for the NASA SWOT project. This circuit is capable of detecting very small phase differences in time-varying analog signals. The proposed FPGA design was tested using a 3 gigasamples per second data acquisition system. Phase was calculated adaptively with an error of less than 0.021° (0.006% of 360°). A high-speed and very precise phasemeter prototype was proposed in [25] with an ability to be used for differential wave-front sensing. The proposed phase measurement algorithm was implemented on an Altera DE2-115 FPGA board and can be used in heterodyne interferometry to measure displacement.

The main goal of this research article is to implement and compare said algorithms by using Verilog HDL in order to test these using 2 different FPGA emulation devices. To make the PUM algorithm fast and applicable for real-time retrieval of vibration and displacement signals from SM signals in an embedded systems environment, in this paper, we propose HDL implementation of parallelized estimation of C and α parameters, presented as the parallelized PUM (PPUM). This is achieved by creating memory registers for a range of C and α values, and then by exploiting the parallel operation characteristic of HDL blocks, we unwrap vibration and displacement signals from SM signals. Then HDL implementation of the DFU algorithm (by incorporating the normalization block, essential for real-world experimental SM signals showing variation in SM signal amplitude) is carried out. As the DFU technique is faster but has less precision than the PUM, to improve the precision of DFU we propose HDL implementation of a new algorithm known as IDFU that results in better precision than DFU at the cost of some extra processing steps (required for identification of peak and valley locations followed by local fringe normalization). At the end, a comparison of HDL-based resource utilization and maximum clock frequency of these algorithms is carried out.

The rest of this article is structured as follows. Section 2 explains the theory of self-mixing. Sections 3, 4, and 5 describe the PUM, DFU, and IDFU algorithms and their FPGA HDL-based implementations, respectively. In Section 6 the results of these algorithms are discussed, and, finally, Section 7 provides the conclusion of this research.

2. Self-mixing theory

In the SM phenomenon, when a laser wave reflects from a target and couples with the active cavity of the laser, the wavelength of the laser is no longer constant, i.e. λ_0, but it varies and becomes $\lambda_F(t)$ when the target moves around. The variations caused in the laser diode's optical output power $P(t)$ because of optical feedback can be formulated as [16]:

$$P(t) = P_0 + mP_0\cos(x_F(t)) \tag{1}$$

where P_0 is the optical power emitted under no-feedback conditions and m represents the index of modulation, while $x_F(t)$ represents the phase signal under feedback and is given by:

$$x_F(t) = \frac{2\pi D(t)}{\lambda_F(t)/2}, \tag{2}$$

where $D(t)$ represents displacement of the moving target and $\lambda_F(t)$ represents the emitted wavelength of LD under feedback conditions. These fluctuations in the wavelength of the laser senor can be found by [16]:

$$x_0(t) = x_F(t) + C\,sin[x_F(t) + \arctan(\alpha)], \tag{3}$$

where $x_0(t)$ represents phase signals under no feedback as a function of λ_0, which is given as below:

$$x_0(t) = \frac{2\pi D(t)}{\lambda_0/2} \tag{4}$$

where $D(t)$ represents displacement of the moving target and λ_0 represents wavelength under no feedback. In SM interferometry, the coupling parameter C has an important role. Variations in C value occur primarily due to changes in laser-to-target distance and the remote target's surface reflectivity [13] and cause fluctuations in SM operating regimes, which may vary from weak to moderate regimes and then to strong feedback regime displacement [26]. On the other hand, the α parameter is a laser diode device parameter with typical values of 4 to 5 for Fabry–Perot type laser diodes [12]. Regarding the impact of α on the SM signal, it introduces asymmetry to the SM signal's shape [18]. That being said, its influence is far less significant as compared to the C parameter, variations of which can easily occur during experimental sensing. Note that all the simulated SM signals presented in this paper are based on the behavioral model of SM sensors, which allows modeling SM signals as a function of $D(t)$, C, and α [27].

3. HDL Implementation of PUM

The PUM is a 2-step algorithm. The first step of the PUM results in a coarse estimation of the phase $x_F(t)$, also called rough PUM [16]. PUM's second step is a signal processing step in which a joint estimation of C and α is achieved by using an optimization routine. This iterative estimation of C and α makes this technique slow and time-consuming [28]. To make this technique fast and applicable for real-time applications, estimation of parameters C and α can be parallelized.

Figure 1 shows our proposed parallelized PUM, which is based on joint parallel estimation of C and α parameters. In Figure 1, the arc-cosine block takes the inverse cosine of the FPGA input SM signal with the help of the CORDIC lookup table. We detect fringes by taking the difference of 2 consecutive arc-cos samples and then compare the difference result with a threshold value. In the staircase block we continuously add or

subtract fringes that are detected in the fringe detection block, which results in a staircase signal. Then we add a staircase signal with arc-cos signal that results in rough recovery of phase signal $x_F(t)$. To make parallel estimations of C and α, we define memory registers for a range of C values in a weak feedback regime, i.e $C \leq 1$ and α values, and then by exploiting the parallel operation characteristic of HDL blocks, we implement characteristic Eq. (1) for unwrapping vibration signals from SM signals.

C and α estimation processes used in Parallel-PUM are depicted in Figure 2. While implementing Parallel-PUM (cf. Figure 1), we take 4 values of C and 4 values of α. For each value of C, we take all values of α and thus we get 16 output target displacements. We then take the difference of 2 consecutive values of each output and pass each difference output through the high-pass FIR filter. Next, global minimization is applied on output displacement signals and we observe that at $C = 0.8$ and $\alpha = 5$ (the values used to simulate the input SM signal) we get the best possible target displacement signal, as shown in Figure 3.

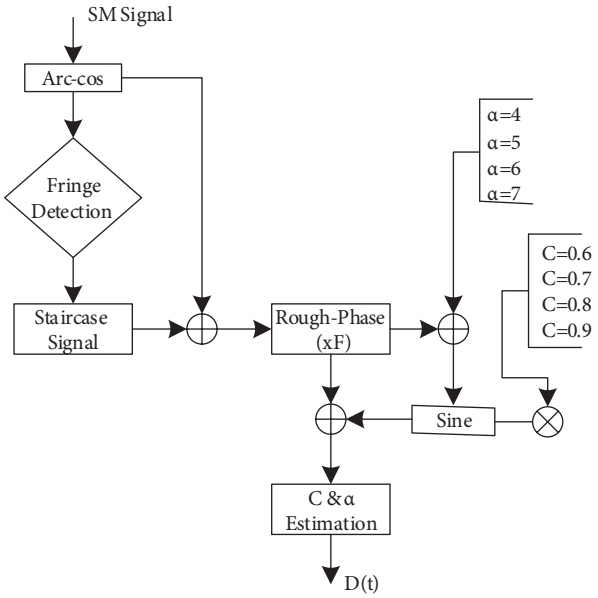

Figure 1. Schematic flow chart diagram of Parallel-PUM.

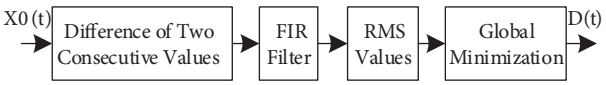

Figure 2. Block diagram of C and α estimation process used in Parallel-PUM.

We have used a parallelized estimation of C and α, where 16 estimations are conducted in parallel to speed up the estimation process. However, the range of C and α can be expanded by using other C and α values in the successive iterations. This, however, will increase the latency of the Parallel-PUM, which is the cost to pay for enlarging the working range of the currently designed algorithm in terms of processing SM signals with a larger C range.

4. HDL implementation of DFU

The DFU algorithm recovers displacement covered by the target from the SM signal using a phase unwrapping technique. The sawtooth-shaped fringes of the SM signal belong to a regime of moderate feedback and have an apparent linear relation with the motion of the target, so precision can be achieved more than $\lambda/2$ by using this linearity concept of fringes.

The SM fringe shape (see, for example, Figure 3a) represents the movement of the target except when

sharp discontinuities occur [21]. Each SM fringe normally represents a $\lambda/2$ displacement of the target. The unwrapping of the target displacement signal is thus achieved through detection of fringes followed by counting of fringes to increase precision more than $\lambda/2$. As DFU is of a real-time nature and a simple technique, this has been previously implemented on a microcontroller-based hardware prototype along with a combination of an adaptive optics-based displacement sensor model that enables vigorous stabilization of the SM optical-feedback regime [29].

The steps that are involved in the DFU method are listed in a flowchart shown in Figure 4. In the first step, we take the derivative of the SM signal shown in Figure 5a. In the second step, we detect the fringes/transitions shown in Figure 5b (which may have either positive amplitude or negative amplitude) by comparing the derivative signal with a threshold value. In the third step, we continuously add or subtract fringes that were detected in the previous step, which results in the staircase signal shown in Figure 5c. In the fourth and final step, we add the normalized SM signal to this staircase signal, which results in the unwrapped target vibration signal shown in Figure 5d. The normalization of the SM signal is needed because even slight mismatching in amplitudes results in additional measurement errors.

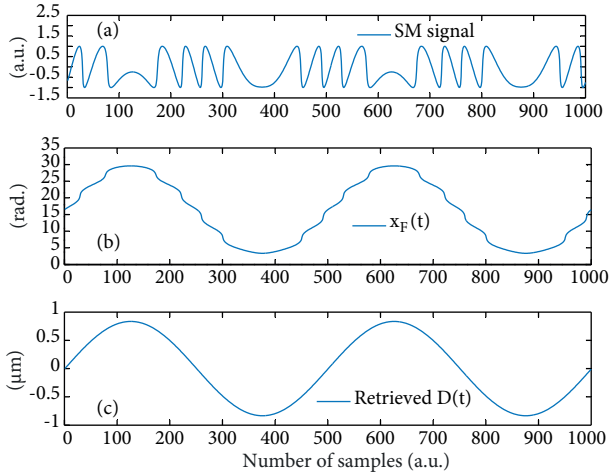

Figure 3. (a) SM signal with $C = 0.8$, $\alpha = 5$, and $\lambda = 785$ nm; (b) unwrapped phase signal; (c) optimum retrieved displacement signal.

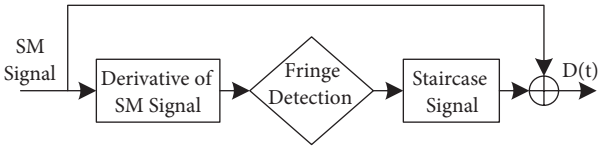

Figure 4. Schematic flow chart of DFU.

5. HDL implementation of IDFU

The DFU technique is faster but has less precision than the PUM. However, the precision of the DFU technique can be improved at the cost of some extra processing steps. We propose a new algorithm known as IDFU, which has target movement measurement precision of order of $\lambda/9$. In IDFU first we globally normalize the SM signal. Then discontinuities of SM signal fringes are detected. On the basis of these detected discontinuities, the SM signal is segmented into sections where each section is delimited from other sections by its surrounding discontinuities [3]. To normalize the SM signal, peak and valley locations are found in each piece of the segment. After the normalization of the SM signal, the phase signal is then unwrapped.

Figure 6 shows the block diagram of HDL implementation of IDFU. In the global normalization block we first find the single maximum and single minimum value in the SM signal and store the values in registers. We then globally normalize the FPGA input SM signal shown in Figure 7 by using the following relation:

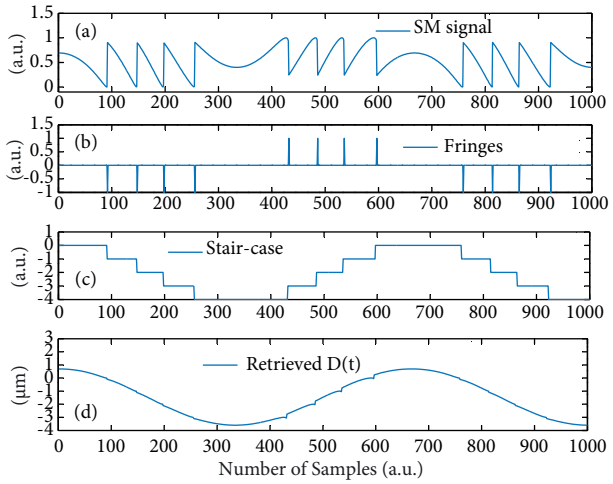

Figure 5. (a) SM signal with $C = 1.75$, $\alpha = 5$, and $\lambda = 785$ nm; (b) detected fringes; (c) staircase signal; (d) output displacement signal.

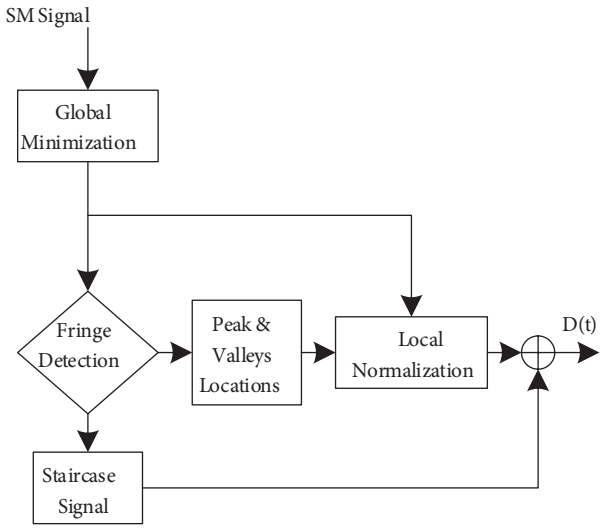

Figure 6. Schematic flow chart of IDFU.

$$D_{in\_Norm} = (d_{in} - \min)/(\max - \min), \tag{5}$$

where d_{in} is the input SM signal to the FPGA and max and min are register-stored maximum and minimum values of the SM signal, respectively. In the fringe detection block we then detect fringe discontinuities and their locations in the globally normalized SM signal and store these values and locations. Based on these detected fringe locations, in the local normalization block, we then find maximum and minimum values from one fringe location to another fringe location for all detected fringes in the globally normalize SM signal and store the values in registers. We then locally normalize the globally normalized SM signal shown in Figures 7a–7e with the help of Eq. (4) by using all fringes' max and min values to correctly extract the phase signal from the SM signal.

6. Results and discussion

The Spartan-3 FPGA series by Xilinx has been used for implementation and testing of these 3 algorithms. This section is further divided into 3 subsections, i.e. PUM results, DFU results, and IDFU results.

6.1. PUM HDL results

In HDL implementation of the PUM, we used 1000 samples of the SM signal. Each sample of the SM signal has a fixed 32-bit width. Table 1 shows the timing report and hardware resource utilization of the PUM algorithm. The FPGA-based PUM algorithm operates at 55.190 MHz maximum frequency.

6.2. DFU HDL results

In HDL implementation of DFU, we used 1000 samples of the SM signal. Each sample of the SM signal has a fixed 32-bit width. Table 2 shows the timing report and hardware resource utilization of the DFU algorithm. The FPGA-based DFU algorithm operates at 82.604 MHz maximum frequency.

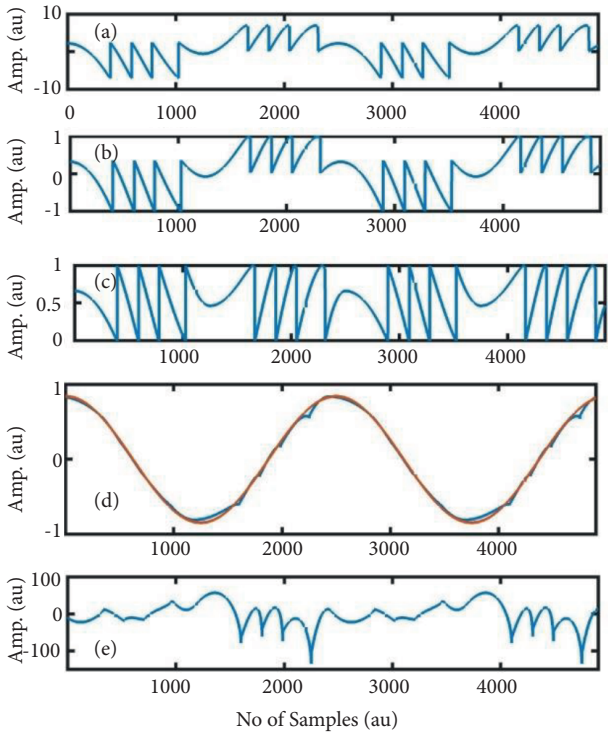

Figure 7. (a) SM signal with $C = 3.75$, $\alpha = 5$, and $\lambda = 785$ nm; (b) globally normalized SM signal; (c) locally normalized SM signal; (d) output displacement signal provided by IDFU (blue line) as compared with reference simulated motion (red line); (e) error between reference motion and IDFU-based retrieved motion.

Table 1. Timing and device resource utilization summary of PUM design.

Maximum frequency	55.190 MHz
Latency	5500 clock cycles
No. of slice registers	3138
No. of slice LUTs	1666
No. of occupied slices	276
No. of LUT flip flop pairs used	476
No. of bounded IOBs	274

6.3. IDFU HDL results

In HDL implementation of IDFU, we used 1000 samples of the SM signal. Each sample of the SM signal has a fixed 32-bit width. Table 3 shows the timing report and hardware resource utilization of the IDFU algorithm. The FPGA-based IDFU algorithm operates at 111.995 MHz maximum frequency.

6.4. Comparisons of HDL implementations of PUM, DFU, and IDFU

The Spartan-3 xcs320-FG420 Xilinx FPGA board has been used for implementation and testing of the PUM, DFU, and IDFU algorithms. Table 4 shows the comparisons of timing reports and hardware resource utilizations of the PUM, DFU, and IDFU algorithms. The FPGA-based PUM algorithm operates at 55.190 MHz maximum

Table 2. Timing and device resource utilization summary of DFU design.

Maximum frequency	82.604 MHz
Latency	1000 clock cycles
No. of slice registers	81
No. of slice LUTs	97
No. of occupied slices	35
No. of LUT flip flop pairs used	74
No. of bounded IOBs	34

Table 3. Timing and device resource utilization summary of IDFU design.

Maximum frequency	111.995 MHz
Latency	2500 clock cycles
No. of slice registers	2053
No. of slice LUTs	1382
No. of occupied slices	202
No. of LUT flip flop pairs used	3649
No. of bounded IOBs	202

clock frequency while the DFU and IDFU algorithms operate at 111.995 MHz and 82.604 MHz, respectively. Though the DFU algorithm has higher frequency than IDFU, IDFU has higher target movement measurement precision as compared to DFU. The results show that our designed IDFU algorithm performed 0.492 times better than the PUM algorithm in maximum clock frequency and 1.53 and 1.21 times better than the PUM in slice registers and LUT utilization of hardware resources, respectively. The designed DFU algorithm can operate 1.355 times better than IDFU in maximum clock frequency and 25.34 and 14.25 times better than IDFU in slice registers and LUT utilization of hardware resources, respectively.

Table 4. Device resource utilization summary of PUM, DFU, and IDFU designs.

Logic	Used				
	PUM	IDFU	DFU	Improvement factor of IDFU w.r.t PUM	Improvement factor of DFU w.r.t IDFU
Maximum frequency	55.190 MHz	111.995 MHz	82.604 MHz	0.492 MHz	1.355 MHz
Latency (clock cycles)	5500	2500	1000	2.2	2.5
No. of slice registers	3138	2053	81	1.53	25.34
No. of slice LUTs	1666	1382	97	1.21	14.25
No. of occupied slices	276	202	35	1.37	5.77
No. of LUT flip flop pairs used	476	3649	74	0.13	49.31
No. of bounded IOBs	274	202	34	1.53	5.94

6.5. Comparisons of HDL implementations of PUM, DFU, and IDFU using the Virtex7 FPGA Device

The Virtex7 XC7VX330T-FFG1157 Xilinx FPGA board has been used for implementation and testing of the PUM, DFU, and IDFU algorithms. Table 5 shows the comparisons of timing reports and hardware resource utilizations of the PUM, DFU, and IDFU algorithms by using the Virtex7 Xilinx FPGA board. The FPGA-based PUM algorithm operates at 185.042 MHz maximum clock frequency while the DFU and IDFU algorithms operate at 239.051 MHz and 205.731 MHz, respectively.

Table 5. Device resource utilization summary of PUM, DFU, and IDFU using Virtex7 FPGA board.

Logic	Used				
	PUM	IDFU	DFU	Improvement factor of IDFU w.r.t PUM	Improvement factor of DFU w.r.t IDFU
Maximum frequency	185.042 MHz	239.051 MHz	205.731 MHz	0.774 MHz	1.162 MHz
Latency (clock cycles)	5500	2500	1000	2.2	2.5
No. of slice registers	2984	1843	62	1.619	29.725
No. of slice LUTs	1703	1421	136	1.1985	10.448
No. of occupied slices	274	204	34	1.343	6.00
No. of LUT flip flop pairs used	581	3867	179	0.150	21.6033
No. of bounded IOBs	274	202	34	1.356	5.941

6.6. Comparisons of measurement performance of PUM, DFU, and IDFU algorithms

Table 6 presents the measurement performance in terms of root mean square (RMS) error and absolute maximum (max.) error of the considered SM displacement retrieval algorithms for the same input SM signal corresponding to $C = 3.75$, $\alpha = 5$, and $\lambda = 785$ nm, while comparisons of error in displacement retrieval are shown in Figures 8a–8d. Specifically, the graphical presentation of error in displacement retrieval can be seen in Figures 7e, 8b, and 8d for IDFU, DFU, and PUM, respectively. For the considered simulated SM signal, Table 6 presents the results of precision = wavelength/max. error for each of these algorithms for operating wavelength $\lambda = 785$ nm, which turns out to be $\lambda/28$, $\lambda/5$, and $\lambda/9$ for PUM, DFU, and IDFU, respectively.

Table 6. Measurement performance in terms of RMS error of different SM displacement retrieval algorithms for the same input SM signal corresponding to $C = 3.75$, $\alpha = 5$, and $\lambda = 785$ nm.

Algorithm	RMS error (nm)	Max. error (nm)	Precision	Time taken to extract target displacement signal from 1000 samples of SM signal
PUM	13.9	27.5	$\lambda/28$	1.8 ms
DFU	56.7	150.5	$\lambda/5$	0.8 ms
IDFU	27.8	89.4	$\lambda/9$	1.2 ms

It is thus seen that the PUM provides the best measurement precision performance. Likewise, it is also seen that the use of local normalization of SM fringes in IDFU (see Figure 6) allows achieving better precision

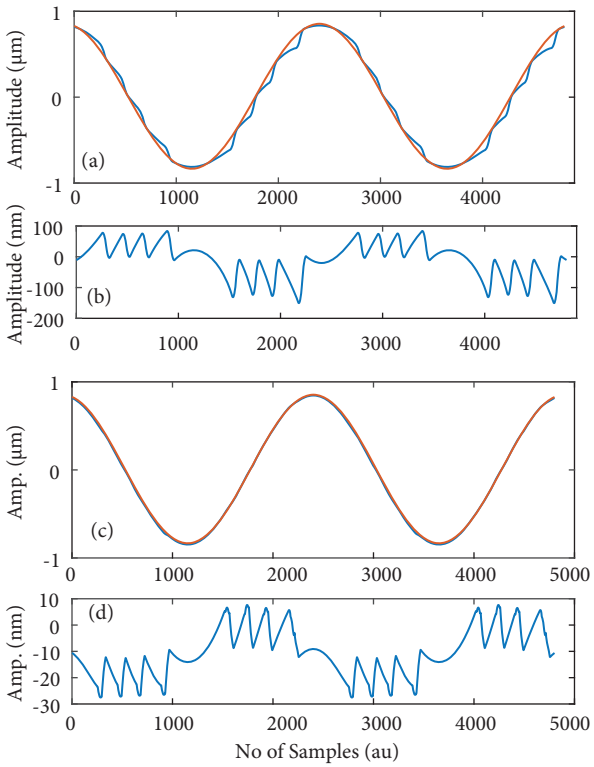

Figure 8. (a) Output displacement signal provided by DFU (blue line) as compared with reference simulated motion corresponding to $C = 3.75$, $\alpha = 5$, and $\lambda = 785$ nm (red line); (b) error between reference motion and DFU-based retrieved motion; (c) output displacement signal provided by PUM (blue line) as compared with reference simulated motion (red line); (d) error between reference motion and PUM-based retrieved motion.

performance as compared to DFU, which does not use local normalization. Use of local normalization of SM fringes becomes important for higher values of C because SM fringes begin to reduce in amplitude as a result of increase in C beyond the value of 1 (the same can be seen by comparing Figure 5a based on $C = 1.75$ with Figures 7a and 7b based on $C = 3.75$). As a consequence, DFU gives poor performance in the case of an increase of C. For example, simulations have shown that the precision of DFU decreases from $\lambda/18$ for $C = 1.75$ to $\lambda/5$ for $C = 3.75$. As reduction in SM fringe amplitude is a major cause of poorer DFU precision for higher C signals, local normalization of individual SM fringes is carried out in IDFU so that individual fringe amplitudes are appropriately adjusted (as seen in Figure 7b and Figure 7c). This then provides comparatively better displacement measurement precision results for IDFU as compared with DFU, as also graphically presented in Figures 7e and 8b.

Table 6 also provides a comparison of computation time for the PUM, DFU, and IDFU algorithms. For the PUM, the execution of 1000 samples of the SM target signal is completed in 1.8 ms to extract target displacement, while the same 1000 samples of the target are executed using DFU in 0.8 ms to extract target displacement. Similarly, for IDFU, the execution of 1000 target samples took 1.2 ms. Computation time comparison shows that the DFU algorithm is about 55.5% faster in extracting target displacement than the PUM for the said case. Equivalently, DFU only takes 44.4% of the time consumed by the PUM to do the same job. Similarly, IDFU is about 33.3% faster than the PUM while it is 50% slower than DFU.

7. Conclusion

The FPGA-based HDL implementation of 3 SM phase unwrapping algorithms is a significant step towards the full integration of a digital SM sensor into a chip, which would be accomplished to deliver nanometric precision in real-time applications for sensing displacement and vibration signals of targets on embedded systems. In this research work, by using HDL, we simulated 3 phase unwrapping algorithms of SM laser sensors, namely PUM, DFU, and IDFU, for 2 different FPGA hardware devices. The final step of the PUM involves an iterative joint estimation of C and α parameters, which makes this algorithm slow and time-consuming and not applicable to real-time applications. We proposed parallel joint estimation of C and α parameters so it becomes fast and can be used in real-time applications. We tested these 3 algorithms on a Xilinx Spartan-3 Xcs400-FG320 development board using Verilog HDL. When we compare the HDL-based results of the PUM and IDFU, we find that our designed IDFU algorithm performs 0.492 times better than Parallel-PUM in maximum clock frequency and 1.53 and 1.21 times better than the PUM in slice registers and LUT utilization of hardware resources, respectively. The designed DFU algorithm can operate 1.355 times better than IDFU in maximum clock frequency and 25.34 and 14.25 times better than IDFU in slice registers and LUT utilization of hardware resources, respectively.

References

[1] Taimre T, Nikolić M, Bertling K, Lim YL, Bosch T, Rakić AD. Laser feedback interferometry: a tutorial on the self-mixing effect for coherent sensing. Adv Opt Photonics 2015; 7: 570-631.

[2] Donati S. Developing self-mixing interferometry for instrumentation and measurements. Laser Photonics Rev 2012; 6: 393-417.

[3] Siddiqui AA, Zabit U, Bernal OD, Raja G, Bosch T. All analog processing of speckle affected self-mixing interferometric signals. IEEE Sens J 2017; 17: 5892-5899.

[4] Magnani A, Norgia M. Spectral analysis for velocity measurement through self-mixing interferometry. IEEE J Quantum Elect 2013; 49: 765-769.

[5] Khan ZA, Zabit U, Bernal OD, Ullah MO, Bosch T. Adaptive cancellation of parasitic vibrations affecting a self-mixing interferometric laser sensor. IEEE T Instrum Meas 2017; 66: 332-339.

[6] Lu L, Zhang W, Yang B, Zhou J, Gui H, Yu B. Dual-channel self-mixing vibration measurement system in a linear cavity fiber laser. IEEE Sens J 2013; 13: 4387-4392.

[7] Yang Y, Li X, Kou K, Zhang L. Optical accelerometer design based on laser self-mixing interference. In: SPIE 2015 Photonic Instrumentation Engineering II; 11–12 February 2015; San Francisco, CA, USA. pp. 93690R-93690R-6.

[8] Bernal OD, Seat HC, Zabit U, Surre F, Bosch T. Robust detection of non-regular interferometric fringes from a self-mixing displacement sensor using bi-wavelet transform. IEEE Sens J 2016; 16: 7903-7910.

[9] Norgia M, Giuliani G, Donati S. Absolute distance measurement with improved accuracy using laser diode self-mixing interferometry in a closed loop. IEEE T Instrum Meas 2007; 56: 1894-1900.

[10] Norgia M, Pesatori A, Rovati L. Self-mixing laser doppler spectra of extracorporeal blood flow: a theoretical and experimental study. IEEE Sens J 2012; 12: 552-557.

[11] Arriaga AL, Bony F, Bosch T. Real-time algorithm for versatile displacement sensors based on self-mixing interferometry. IEEE Sens J 2016; 16: 195-202.

[12] Henry CH. Theory of the linewidth of semiconductor lasers. IEEE J Quantum Electron 1985; 18: 259-264.

[13] Taimre T, Rakic AD. On the nature of Acket's characteristic parameter C in semiconductor lasers. Appl Opt 2014; 53: 1001-1006.

[14] Bes C, Plantier G, Bosch T. Displacement measurements using a self-mixing laser diode under moderate feedback. IEEE T Instrum Meas 2006; 55: 1101-1105.

[15] Zabit U, Bosch T, Bony F. Adaptive transition detection algorithm for a self-mixing displacement sensor. IEEE Sens J 2009; 9: 1879-1886.

[16] Zabit U, Bernal O, Bosch T. Time-frequency signal processing for a self-mixing laser sensor for vibration measurement. In: IEEE Sensors 2012; 28–31 October 2012; Taipei, Taiwan. New York, NY, USA: IEEE. pp. 1-4.

[17] Bernal OD, Zabit U, Bosch T. Study of laser feedback phase under self-mixing leading to improved phase unwrapping for vibration sensing. IEEE Sens J 2013; 13: 4962-4971.

[18] Fan Y, Yu Y, Xi J, Chicharo JF. Improving the measurement performance for a self-mixing interferometry-based displacement sensing system. Appl Optics 2011; 50: 5064-5072.

[19] Norgia M, Pesatori A. Fully analog self-mixing laser vibrometer. In: 2011 IEEE International Instrumentation and Measurement Technology Conference; 10–12 May 2011; Binjiang, China. New York, NY, USA: IEEE. pp. 1-4.

[20] Magnani A, Pesatori A, Norgia M. Self-mixing vibrometer with real-time digital signal elaboration. Appl Optics 2012; 51: 5318-5325.

[21] Zabit U, Bernal O, Bosch T. Self-mixing sensor for real-time measurement of harmonic and arbitrary displacements. In: 2012 IEEE International Instrumentation and Measurement Technology Conference; 13–16 May 2012; Graz, Austria. New York, NY, USA: IEEE. pp. 754-758.

[22] Li Z, Yu Y, Xi J, Ye H. FPGA-based signal processing in an optical feedback self-mixing interferometry system. In: Photonics Asia 2010; 18–20 October 2010; Beijing, China. pp. 78550M-1-78550M-7.

[23] Sun Y, Yu Y, Fan W, Xi J. FPGA based filter design for self-mixing interferometry signals. In: International Conference on Optical Instruments and Technology; 6–9 November 2011; Beijing, China. pp. 819909-1-819909-7.

[24] Lu SJ, Siqueira P, Vijayendra V, Chandrikakutty H, Tessier R. Real-time differential signal phase estimation for space-based systems using FPGAs. IEEE T Aero Elec Sys 2013; 49: 1192-1209.

[25] Wang C. FPGA-based, 4-channel, high-speed phasemeter for heterodyne interferometry. MSc, University of Rochester, Rochester, NY, USA, 2013.

[26] Tartwijk GV, Lenstra D. Semiconductor lasers with optical injection and feedback. Quantum Semicl Opt 1995; 7: 87.

[27] Plantier G, Bes C, Bosch T. Behavioral model of a self-mixing laser diode sensor. IEEE J Quantum Elect 2005; 41: 1157-1167.

[28] Xi J, Yu Y, Chicharo JF, Bosch T. Estimating the parameters of semiconductor lasers based on weak optical feedback self-mixing interferometry. IEEE J Quantum Elect 2005; 41: 1058-1064.

[29] Bernal OD, Zabit U, Bosch TM. Robust method of stabilization of optical feedback regime by using adaptive optics for a self-mixing micro-interferometer laser displacement sensor. IEEE J Sel Top Quant 2015; 21: 336-343.