# BOTDA Fiber Sensor System Based on FPGA Accelerated Support Vector Regression

Huan Wu, Hongda Wang, Chester Shu, *Senior Member, IEEE*, Chiu-Sing Choy, *Senior Member, IEEE*, and Chao Lu

***Abstract*— Brillouin optical time domain analyzer (BOTDA) fiber sensors have shown strong capability in static long haul distributed temperature/strain sensing. However, in applications such as structural health monitoring and leakage detection, real-time measurement is quite necessary. The measurement time of temperature/strain in a BOTDA system includes data acquisition time and post-processing time. In this article, we propose to use hardware accelerated support vector regression (SVR) for the post-processing of the collected BOTDA data. Ideal Lorentzian curves under different temperatures with different linewidths are used to train the SVR model to determine the linear SVR decision function. The performances of SVR are evaluated under different signal-to-noise ratios (SNRs) experimentally. After the model coefficients are determined, algorithm-specific hardware accelerators based on field-programmable gate arrays (FPGAs) are used to realize SVR decision function. During the implementation, hardware optimization techniques based on loop dependence analysis and batch processing are proposed to reduce the execution latency. Our FPGA implementations can achieve up to 42× speedup compared with software implementation on an i7-5960*x* computer. The post-processing time for 96 100 Brillouin gain spectrums (BGSs) along with 38.44-km fiber under test (FUT) is only 0.46 s with FPGA board ZCU104, making the post-processing time no longer a limiting factor for dynamic sensing. Moreover, the energy efficiency of our FPGA implementation can reach up to 226.1× higher than the software implementation based on CPU.**

***Index Terms*— Brillouin optical time-domain analyzer (BOTDA), digital signal processing, distributed optical fiber sensing, field-programmable gate arrays (FPGAs), hardware implementation, support vector machine (SVM).**

## I. Introduction

DISTRIBUTED optical fiber sensors allow many points to be measured simultaneously, and it is compatible to ubiquitously deployed underground fiber system for telecommunication purpose; therefore, they attract interest from both industry and academia [1]–[4]. Distributed optical fiber sensors mainly depend on scattering effects in optical fiber, Brillouin optical time-domain analyzer (BOTDA) based on stimulated Brillouin scattering (SBS) was invented in 1990 [5], it can measure strain as well as the temperature. BOTDA sensors rely on SBS of two counter-propagating light waves, a continuous-wave (CW) signal, and a pulsed pump. The frequency offset between the pump and probe is scanned around the Brillouin frequency shift (BFS) of the fiber to reconstruct the Brillouin gain spectrum (BGS). Since the change of BFS has a linear relationship with the change of temperature and strain on the fiber, an important operation in a BOTDA system is to find the BFS from the measured BGS to determine temperature or strain information along the fiber under test (FUT). In an ideal BGS, BFS is the shift in peak gain frequency. However, acquired BGSs are always contaminated by noises. Therefore, post-processing algorithms are needed to accurately determine BFS from the measured BGSs. The conventional wisdom to predict the BFS information from the BGS is Levenberg–Marquardt algorithm (LMA) curve fitting [6], [7]. However, its complexity is often a limiting factor in the sensing speed of a BOTDA system, especially for long sensing distance.

In recent years, the performances of BOTDA are improved significantly due to the rapid developments of the technology. The sensing distance of BOTDA can achieve hundreds of kilometers [8], and the spatial resolution can be reduced to the millimeter level [9]. Longer sensing distance brings a larger amount of sensing data and finer resolution requires a higher sampling rate and smaller frequency scanning step which result in denser sensing points. The sensing data volume keeps increasing, which adds the computational load for post-processing. In a real scenario, to extract temperature/strain information from the measured BGSs with low latency is quite necessary. Several works have mentioned the challenges of post-processing in real applications. In [10], a non-curve fitting technique called cross correlation method (XCM) was proposed based on the calculation of cross correlation (XC) between an ideal Lorentzian curve and the measured BGS to determine BFS. In [11], a modified version of XCM is implemented on field-programmable gate array (FPGA) to speed up the processing time at the cost of sensing accuracy. Artificial neural network (ANN) is proposed for BOTDA system to improve the sensing accuracy and processing speed [12]. However, the training of ANN is difficult due to numerous hyperparameters. Recently, we reported a machine learning method called support vector classification (SVC) to extract temperature information from measured BGSs with simple training strategy and fast processing speed [13]–[15].

Manuscript received May 27, 2019; revised July 17, 2019; accepted July 30, 2019. Date of publication October 10, 2019; date of current version May 12, 2020. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 61435006, in part by The Chinese University of Hong Kong (CUHK) Group Research Scheme, CUHK Direct, under Grant 4055090, and in part by the Research Grants Council (RGC) of Hong Kong Project under Grant RGC GRF CUHK 14204918 and Grant PolyU 152658/16. The Associate Editor coordinating the review process was Pekka Keränen. *(Huan Wu and Hongda Wang contributed equally to this work.) (Corresponding author: Huan Wu.)*

H. Wu and C. Lu are with the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong (e-mail: hkpolyu.wu@poly.edu.hk; chao.lu@polyu.edu.hk).

H. Wang, C. Shu, and C.-S. Choy are with the Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: 1155039965@link.cuhk.edu.hk; ctshu@cuhk.edu.hk; cschoy@ee.cuhk.edu.hk).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIM.2019.2936775

By far, most post-processing methods are implemented on X86-based computers like windows desktop and Linux servers. These methods normally rely on separate stages including experimental data measurement, data storage, data transmission, and data post-processing. Due to the discontinuous workflow, real-time post-processing is challenging because additional cost by the data storage and transmission cannot be ignored, not to mention the running time of the computation-intensive algorithms. Thus, it is challenging to realize real-time surveillance and monitoring based on software solutions. For the applications which need extremely low latency, the hardware solutions are alternatives and can alleviate the difficulties, as mentioned above [16]. Among the current hardware platforms, FPGA and ASIC are two representatives which can provide sufficient computation capability, moreover, FPGA and ASIC are more compact which means they can be easily integrated into a portable BOTDA system. Compared with ASIC, FPGA has the advantages of reconfigurability and fast deployment time especially with the help of high-level synthesis [17]. The computation capability of FPGA can outperform CPU by one order of magnitude through massively parallelizing the algorithm in an efficient manner [18]. However, not all the algorithms can be easily implemented on FPGA because of the inadaptability to fixed hardware structures. LMA curve fitting is an iterative optimization method based on the error gradient estimation, which requires matrix inversion during error correction. From the hardware perspective, the iterative process and matrix inversion are expensive to realize. For XC, the total multiply-accumulation (MAC) operations required is proportional to the square of frequency number in the BGS, thus pose a huge computation burden, therefore, in [11], moving average and referential triangular pulse are adopted to reduce the computational complexity at the expense of BFS accuracy. The ANN proposed in [12] is also not very suitable for efficient hardware implementation since the sigmoid nonlinear activation function in each neuron can only be approximated with a big lookup table (LUT) or low-order polynomial expansion. As for SVC [13]–[15], $n$-class SVC is built upon $n(n-1)/2$ binary classifiers and each classifier has a unique number of support vectors, the irregular computation pattern does not fit a fixed hardware structure. What is more, $n$ is normally larger than 100 which generates more than 5000 independent binary classifiers, causing heavy storage requirements and computation loads. Considering the convenience and efficiency of the algorithm to hardware migration, we propose a novel method called support vector regression (SVR) in our recent conference paper [19]. In this article, an in-depth comparison with other methods regarding the algorithm performances is included. Moreover, multiple optimization strategies with high-level synthesis are proposed and proved experimentally. The main contributions of this article are as follows.

1) A temperature prediction method based on the SVR is proposed. The experimental results prove that SVR can achieve comparable performance with existing BFS extraction methods like SVC, XC, LMA Lorentzian curve fitting (LCF), and ANN.
2) Hardware implementations of SVR decision function are realized on two FPGA boards. Optimizations to linear SVR decision function through loop analysis and batch processing are proposed to take advantages of high flexibility and scalability of modern FPGA devices. These optimization methods transform the decision function into the matrix–matrix multiplication and matrix–vector multiplication and parallelize these operations by tiling the large matrix into smaller ones.
3) Post-processing time for 96 100 BGSs along 38.44-km FUT can be completed in 0.46 s with Xilinx ZCU104 using the proposed hardware optimization techniques. It achieves 42× speedup compared with the software implementation running on an i7-5960$x$ computer. Meanwhile, the 26.5-W power consumption of ZCU104 is also much lower than the conventional CPU, making the energy efficiency of our FPGA implementation 221.6× higher than the software implementation based on LIBSVM [20].

This article is organized as follows. Section II describes the principle of SVR and its training process for temperature extraction in a BOTDA system. Section III introduces the experimental setup of BOTDA and evaluates the performance of SVR under different signal-to-noise ratios (SNRs) experimentally. FPGA optimizations and implementations of linear SVR decision function are given in Section IV. Section V concludes this article.

## II. Principle of SVR and Training Process for Temperature Extraction in a BOTDA System

Suppose we have training data $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_l, y_l)\}$, where $\boldsymbol{x}_i \in R^n$ is the training sample and $y_i \in R$ is label. In linear case, we construct a linear decision function to fit the training data

$$f(\boldsymbol{x}) = \langle \boldsymbol{w}, \boldsymbol{x} \rangle + b \tag{2.1}$$

where $\langle \cdot, \cdot \rangle$ denotes the dot product, $\boldsymbol{w}$ is the norm vector of the linear function and $b$ is the intercept. Traditional linear least-square error regression derives a decision function by minimizing the deviation between predicted value $f(\boldsymbol{x}_i)$ and given value $y_i$ for all training data. Unlike linear least-square error fitting, SVR allows a tolerance degree to errors not greater than $\varepsilon$, as shown in Fig. 1(a). Only the data points outside the shaded region contribute to the error and the deviations are penalized in a linear fashion, as shown in Fig. 1(b). The goal of SVR is to find a function that fits current training data with a deviation no larger than $\varepsilon$, and at the same time as flat as possible. One way to ensure this is to minimize the norm, i.e., $\|\boldsymbol{w}\|^2 = \langle \boldsymbol{w}, \boldsymbol{w} \rangle$. We can write this problem as a convex optimization problem as follows:

$$\min \frac{1}{2}\|\boldsymbol{w}\|^2 \quad \text{s.t.} \begin{cases} y_i - \langle \boldsymbol{w}, x \rangle - b \le \varepsilon \\ \langle \boldsymbol{w}, x \rangle + b - y_i \le \varepsilon. \end{cases} \tag{2.2}$$

The above convex optimization problem is feasible in cases where $f(\boldsymbol{x})$ actually exists and all pairs $(\boldsymbol{x}_i, y_i)$ are within $\varepsilon$ precision. However, in most cases, not all $(\boldsymbol{x}_i, y_i)$ are within $\varepsilon$

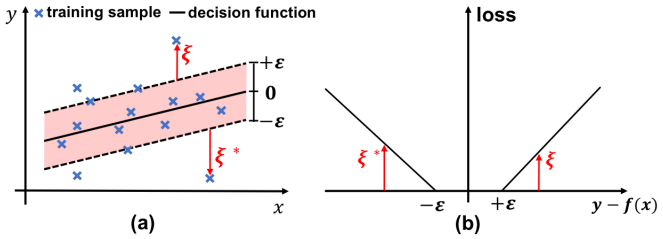



Fig. 1. (a) 1-D linear SVR for illustration. (b) Linear loss function.

precision, then we can introduce slack variables $\xi_i$, $\xi_i^*$ to deal with this problem. Hence, we get the following formulation:

$$\min \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*)$$
$$\text{s.t.} \begin{cases} y_i - \langle \boldsymbol{w}, \boldsymbol{x}\rangle - b \le \varepsilon + \xi_i \\ \langle \boldsymbol{w}, \boldsymbol{x}\rangle + b - y_i \le \varepsilon + \xi_i^* \end{cases} \tag{2.3}$$

where $\xi_i$, $\xi_i^* \ge 0$, the constant $C > 0$ determines the trade-off between the flatness of $f(x)$ and the amount up to which deviations larger than $\varepsilon$ are tolerated. Equation (2.3) is known as the primal problem of SVR algorithm and it can be transformed to dual problem and solved by quadratic programming [21]. The solution is as follows:

$$\mathbf{w} = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)(\boldsymbol{x}_i) \tag{2.4}$$

$$f(\boldsymbol{x}) = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)\langle \boldsymbol{x}_i, \boldsymbol{x}\rangle + b \tag{2.5}$$

where $\alpha_i$ and $\alpha_i^*$ are the dual variables, $\langle \boldsymbol{x}_i, \boldsymbol{x}\rangle$ represents the inner product between the training sample $\boldsymbol{x}_i$ and test sample $\boldsymbol{x}$. From (2.5), we can see that once the model parameters are identified, SVR only depends on $\boldsymbol{x}_i$ with corresponding $(\alpha_i - \alpha_i^*)$ which are non-zero, these $\boldsymbol{x}_i$ are called support vectors and they are the subsets of training data.

In our case, to process the measured BGSs collected from a BOTDA system, a high dimensional linear SVR is used, the normalized gain value at every frequency on the BGS forms feature vector $\boldsymbol{x}_i$, and the corresponding temperature of the BGS is label $y_i$. The use of SVR includes two phases, the training phase and testing phase, as shown in Fig. 2. During the training phase, the simulated ideal BGSs together with the corresponding temperature labels serving as the training samples are used to get linear decision function for temperature prediction. We design the simulated ideal BGSs by using ideal Lorentzian curve as the gain profile for the training of SVR

$$\mathrm{g}(\nu) = \frac{g_B}{1 + \left[\frac{(\nu - \nu_B)}{\Delta\nu_B/2}\right]^2} \tag{2.6}$$

where $g_B$, $\nu_B$, and $\Delta\nu_B$ are the peak gain, BFS, and bandwidth of the BGS, respectively. Peak gain is set as 1, BFSs of the ideal BGSs from a temperature range of 0 °C to 70 °C with 0.5 °C step are determined using the temperature coefficient (0.975MHz/°C) of the FUT. It should be mentioned that the

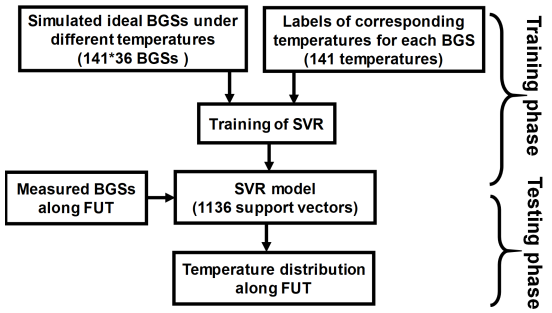



Fig. 2. Training and testing phase of SVR.

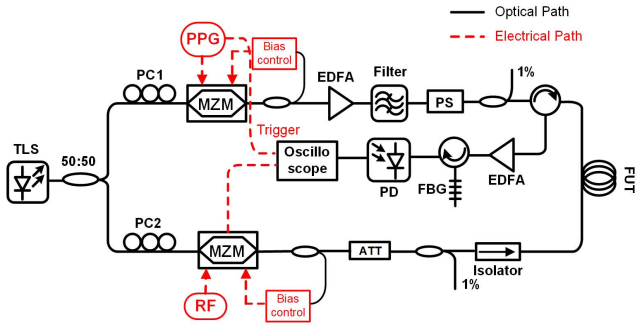



Fig. 3. BOTDA experimental setup. TLS, PC, PPG, RF, PS, MZM, ATT, FUT, FBG, and PD.

temperature range is determined by the application, in our case, the room temperature is around 20 °C and part of the fiber is heated to 50 °C; therefore, we set the temperature range in [0, 70]. The bandwidth of the BGS is determined by the pulsewidth/spatial resolution, to accommodate pulsewidth from minimum 10 ns to infinity (CW), we set the bandwidth of the ideal BGSs from 30 to 100 MHz at a step of 2 MHz. Finally, we have 141 × 36 ideal BGSs to train the SVR. The frequency range of $\nu$ is from 10.78 to 11.0 GHz with 1-MHz step, therefore, we have 220 frequencies. After training, we get 1136 support vectors in the SVR model. In the testing phase, the fixed model predicts a continuous temperature value for each normalized measured BGS collected from a BOTDA system.

## III. BOTDA Setup and Experimental Results

### A. BOTDA Experimental Setup

The experimental setup of the BOTDA system is shown in Fig. 3. The output of a tunable laser source is set around 1550 nm and is split into two branches using a coupler. The CW light in the upper branch is modulated by a Mach–Zehnder modulator (MZM) driven by a pulse pattern generator (PPG) to generate optical pump pulses. The bias controller after MZM is to stabilize the applied voltage. The pump is then amplified by an erbium-doped fiber amplifier (EDFA) and passes through a polarization scrambler (PS) to eliminate polarization dependent noise. In the lower branch, another high extinction ratio MZM is driven by a radio frequency (RF) generator. The bias controller is biased at the Null point to generate a carrier suppressed double-sideband probe signal.

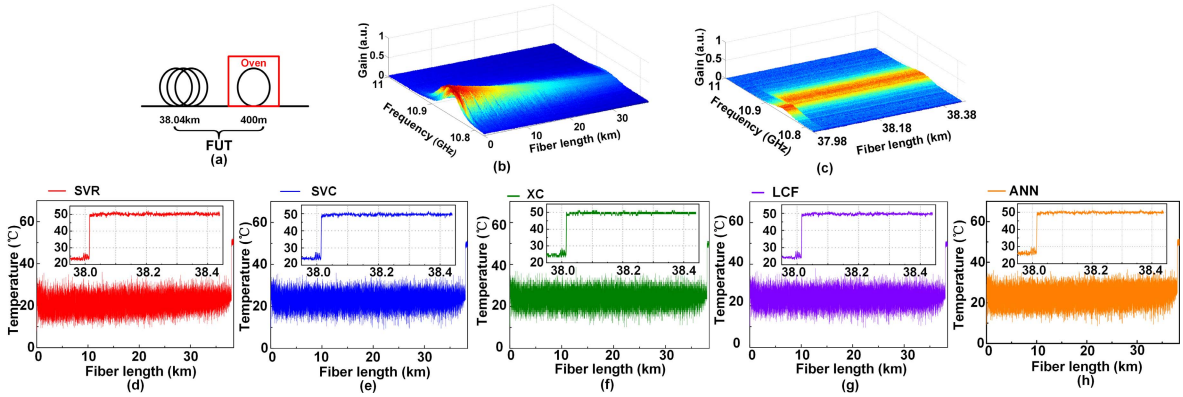



Fig. 4. (a) 38.44-km FUT with last 400 m heated to 50 °C. (b) Measured BGS distribution along FUT. (c) Zoomed-in view of the last heated section. Temperature distribution along FUT determined by (d) SVR, (e) SVC, (f) XC, (g) LCF, and (h) ANN.

TABLE I
PERFORMANCES OF SVR, SVC, XC, LCF, AND ANN UNDER DIFFERENT SNRS

| SNR (dB) | MT (s) | SVR | | | SVC | | | XC | | | LCF | | | ANN | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | STD | RMSE | MAE | STD | RMSE | MAE | STD | RMSE | MAE | STD | RMSE | MAE | STD | RMSE | MAE |
| **4.5** | 2.7 | 2.087 | 2.343 | 1.89 | 1.852 | 1.861 | 1.479 | 2.021 | 2.339 | 1.852 | 1.964 | 1.985 | 1.605 | 2.12 | 2.133 | 1.701 |
| **6** | 5.4 | 1.578 | 1.584 | 1.264 | 1.556 | 1.562 | 1.241 | 1.655 | 1.783 | 1.432 | 1.563 | 1.61 | 1.276 | 1.783 | 1.789 | 1.439 |
| **8** | 17 | 1.262 | 1.295 | 1.037 | 1.127 | 1.161 | 0.919 | 1.225 | 1.29 | 0.996 | 1.122 | 1.159 | 0.928 | 1.321 | 1.357 | 1.078 |
| **10** | 33.8 | 0.785 | 0.803 | 0.634 | 0.73 | 0.86 | 0.661 | 0.783 | 0.878 | 0.694 | 0.732 | 0.864 | 0.701 | 0.858 | 0.917 | 0.74 |
| **12** | 88 | 0.582 | 0.588 | 0.469 | 0.537 | 0.749 | 0.596 | 0.619 | 0.645 | 0.472 | 0.527 | 0.639 | 0.514 | 0.568 | 0.717 | 0.583 |

An optical attenuator (ATT) is used to control the probe power followed by an isolator to block the signal from the pump branch. The probe signal is detected by a photodetector (PD) after the lower-frequency probe sideband is selected by using a fiber Bragg grating (FBG) filter. Local BGSs are reconstructed with RF scanned around the BFS of FUT.

Ensemble average is commonly used to increase SNR since the signal is at least partially reproducible while noise is random from one measurement to the next. Conventional BOTDA systems widely adopt the ensemble average at the cost of longer data acquisition time. Even though it greatly improves the signal quality, it limits the BOTDA systems for dynamic sensing. Both experimental and algorithmic methods have been proposed to reduce the number of ensemble average [22], [23].

Another limit for BOTDA dynamic sensing is post-processing of collected BOTDA data. Total temperature/strain sensing time of the BOTDA system can be expressed as follows:

$$T = T_{\text{acq}} + T_{\text{pp}} = (T_c \cdot N_{\text{avg}} + T_s)N_{\text{freq}} + T_{\text{pp}} \qquad (3.1)$$

where $T_{\text{acq}}$ is the data acquisition time, $T_{\text{pp}}$ is the post-processing time, $T_c = 2nL/c$ is the time of flight, $L$ is the length of FUT, $n$ is the refractive index of the fiber, and $c$ is the light speed in the vacuum. $N_{\text{avg}}$ is the number of ensemble average, $T_s$ is the frequency switching time of RF which is around hundreds of milliseconds, and $N_{\text{freq}}$ is the number of scanned frequencies.

### *B. Experimental Results*

To evaluate the performance of SVR, we use the BOTDA setup shown in Fig. 3 to measure the BGS distribution along 38.44-km FUT. The last 400-m section of FUT is free from strain and put in a temperature oven heated to 50 °C, as shown in Fig. 4(a). The sampling rate is 250 MSample/s, corresponding to 96 100 sampling points for 38.44-km FUT. Fig. 4(b) shows the BGSs distribution measured with 20-ns pump pulse, 1024 times averaging, and the sweeping frequency is from 10.78 to 11.0 GHz with 1-MHz frequency step. Fig. 4(c) is the zoomed-in view of the last heated section.

Next, the measured BGSs are processed by SVR. For comparison, we also process the BGSs by SVC, XC, LMA LCF, and ANN, respectively, as shown in Fig. 4(d)–(h). The insets in Fig. 4(d)–(h) show the zoomed-in view at the heated section. We can see that the temperature information along FUT have been successfully extracted by these algorithms. Then, we investigate the performances of these algorithms under different SNRs, the pump pulse is fixed at 20 ns and frequency scanning step at 1 MHz. SNR is defined as the ratio between the mean amplitude of Brillouin peak and its standard deviation (STD). We collect the BGSs from 4.5 to 12 dB by using 32 to 1024 times of averaging. According to (3.1), theoretical measurement time (MT) varies from 2.7 to 88 s when averaging number increases from 32 to 1024. The performances of these algorithms are evaluated by three merits, STD, root mean square error (RMSE) and mean absolute error (MAE). The performances of the five algorithms under different SNRs are shown in Table I, we can see that the lower STD, RMSE, and MAE can be achieved with higher SNR for all these methods at the cost of longer MT. However, there is not a single method exhibiting overwhelming performances over others for all SNRs. For example, when SNR is 8 dB, LCF achieves the lowest uncertainty, however, at 4.5 dB, SVC shows the lowest uncertainty. Another phenomenon is that

STD, RMSE, and MAE may not be consistent. For instance, at 12 dB, LCF has the lowest uncertainty while SVR has the lowest RMSE and MAE, it is because the true temperature is involved in calculating RMSE and MAE. However, the true temperature is not strictly 50 °C at any fiber location in the oven at any time, the uniformity of the oven we use is about 0.5 °C, which may introduce systematic error in this process. Considering the systematic error and random error in data acquisition, the performances of the five algorithms are comparable. Overall, to enhance the performances of BOTDA fiber sensors, the key is to improve the signal quality since the improvement margin by different BFS extraction methods is quite limited.

Even though various methods can be used to extract BFS information, as analyzed previously, the adaptability of these algorithms to hardware structures are different. SVR predicts the result by regular matrix-vector multiplication and inner-product as shown in (2.5), which is very suitable to be parallelized and pipelined from the hardware perspective. With a dedicated FPGA accelerator, the processing speed of linear SVR can be significantly improved.

## IV. FPGA Optimizations and Implementations of SVR

In this section, a hardware architecture for the linear SVR decision function is presented. In the following, Section IV-A introduces the direct implementation of linear SVR decision function and discusses its drawbacks. In Section IV-B, optimizations to the direct implementation by loop analysis are proposed to reduce the latency. In Section IV-C, the batch processing method is proposed to further speed up the running time. In Section IV-D, 96 100 measured BGSs from 38.44-km FUT are processed by two FPGA boards, experimental results and comparison with software implementation and other works are described. In Section IV-E, we give an in-depth theoretical analysis and discussion for FPGA acceleration with the proposed optimization techniques.

### A. Direct Implementation of Linear SVR Decision Function

If we simplify $(\alpha_i - \alpha_i^*)$ in decision function (2.5) as $\beta_i$ and expand the inner product to a sum-of-product term, then we can have the reformulated decision function as follows:

$$f(\boldsymbol{x}) = \sum_{i=1}^{N_s} \beta_i \sum_{j=1}^{M} SV_{ij} \cdot x_j + b \tag{4.1}$$

where SV represents support vectors obtained from the training process, $N_s$ is the number of support vectors and is 1136, as given in Section II, $M$ is the dimension of the input feature vector and is equal to 220. The data path of (4.1) can be illustrated in Fig. 5 and the corresponding pseudocode is shown in Algorithm 1. In Fig. 5, MAC 1 corresponds to the inner summation of (4.1) and is denoted as partial sum, while MAC 2 corresponds to the outer summation and is denoted as the final sum. The total MAC operations in MAC 1 and MAC 2 are $(N_s \cdot M + N_s)N_{\mathrm{BGS}}$, where $N_{\mathrm{BGS}}$ is the number of BGSs. To process 96 100 measured BGSs from 38.44-km FUT, about

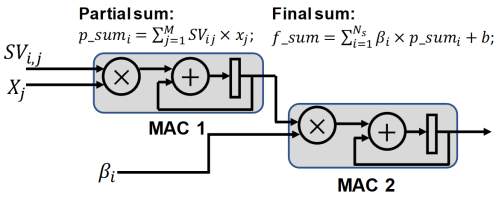



Fig. 5. Data path of linear SVR, note that the bias term $b$ in (4.1) is omitted since it is used to initialize the register in MAC 2.

$2.4 \times 10^{10}$ multiplications and $2.4 \times 10^{10}$ summations are needed, resulting in a heavy computation burden for real-time processing.

**Algorithm 1** Original Linear SVR Without Optimization

**Input**: feature vector $x[M]$
**Require**: support vectors $SV[N_s][M]$, support vector corresponding multipliers $\beta[N_s]$, $bias$
**Output**: regression result $f(x)$
**Initialize**: $p\_sum[N_s] \leftarrow 0$, $f\_sum \leftarrow bias$
**L1: for** $i = 0$ to $N_s - 1$ **do**
  **L2: for** $j = 0$ to $M - 1$ **do**
    $square \leftarrow SV[i][j] * x[j]$;
    $p\_sum[i] \leftarrow p\_sum[i] + square$;
  **end for**
  $temp[i] \leftarrow \beta[i] * p\_sum[i]$;
  $f\_sum \leftarrow f\_sum + temp[i]$;
**end for**
$f(x) \leftarrow f\_sum$;

In hardware design, parallel and pipeline are the two common techniques to improve the performance. However, the loop-carried dependence in the inner loop L2 in Algorithm 1 causes long pipeline initiation interval and inefficient hardware utilization efficiency. Moreover, due to the existed dependence, the parallelism of this direct implementation cannot be achieved without restructuring the code, thus, the total latency is heavily restricted. To accelerate the decision function and enable real-time processing, optimizations must be performed to overcome the limitations.

### B. Loop Dependence Analysis and Optimizations

To remove the loop-carried dependence and parallelize the partial sum computation, first, we need to perform loop dependence analysis [24]. In Algorithm 1, the statements inside L2 exhibit inter-dependence with respect to the iterator $j$ but show no inter-dependence on iterator $i$. Thus, we seek to change the execution order of L1 and L2 to remove the inter-dependence. However, the nested loop is imperfect (perfect nested loops mean the statements only exist inside the innermost loop), we need to take a two-step optimization.

1) *Loop Distribution:* We find that the statements inside L2 do not depend on the statements between L1 and L2, this means we can safely break loop L1 and distribute the statements between L1 and L2 outside. After the loop distribution, a new loop L3 is formed which is

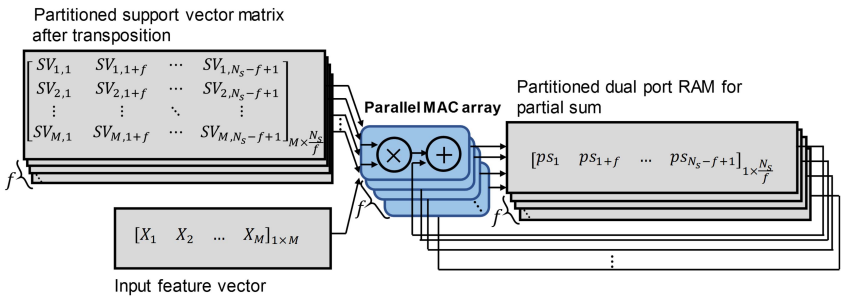



Fig. 6. Hardware structure for calculating the partial sum. The memory for support vector matrix and partial sum is partitioned to $f$ individual blocks, respectively. Meanwhile, $f$ parallel MAC units keep updating the values of the partial sum iteratively.

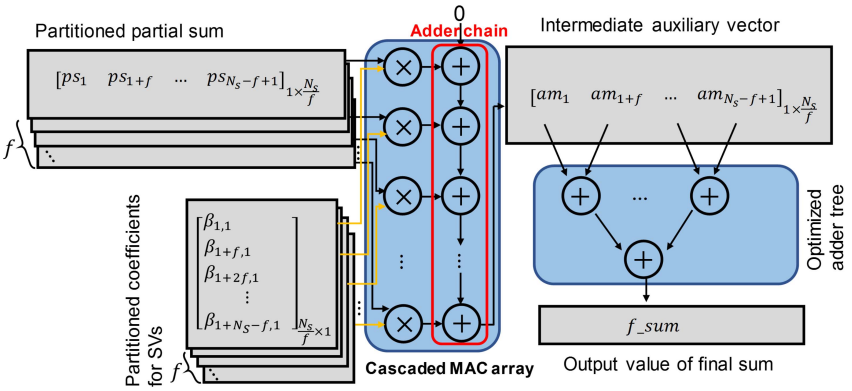



Fig. 7. Hardware structure for calculating the final sum. The coefficients for SVs are also partitioned to $f$ blocks as partial sum to accommodate the parallel access of the cascaded MAC array. The output of the adder chain is written to the intermediate auxiliary vector and accumulated through an optimized adder tree to generate the final sum.

only responsible for the accumulation of final sum, while L1 and L2 become a perfect nested loop and calculates the partial sum.

2) *Loop Interchange:* In the perfect nested loop L1-L2, loop-carried dependence prevents efficient pipeline strategy to be applied because of the long execution latency of the accumulator. The pipeline initiation interval is restricted by the propagation delay of the adder, which is normally larger than one clock cycle for floating-point numbers. When working in higher frequency, the propagation delay could further consume more clock cycles, resulting in a longer pipeline initiation interval. In Algorithm 1, no inter-dependence is observed between the statements inside L2 and the iterator $i$, therefore, we can interchange L1 and L2 to remove the dependence and make the nested loop executed consecutively in each clock cycle. After loop interchange, the partial sum is read and write simultaneously with no conflict on the access addresses, which indicates that the partial sum should be mapped to the dual-port RAM on FPGA.

The pseudocode after the loop distribution and loop interchange is in shown Algorithm 2. Since the execution order of L1 and L2 is changed, the support vector matrix also needs to be transposed accordingly. The total execution latency in clock cycles can be expressed as follows:

$$\text{Latency} = N_s \cdot M + N_s \cdot T_a \tag{4.2}$$

**Algorithm 2** Optimized Linear SVR With Loop Distribution and Interchange

**Input**: feature vector $x[M]$
**Require**: support vectors $SV[M][N_s]$, support vector corresponding multipliers $\beta[N_s]$, $bias$
**Output**: regression result $f(x)$
**Initialize**: $p\_sum[N_s] \leftarrow 0$, $f\_sum \leftarrow bias$
**L1: for** $i = 0$ to $M - 1$ **do**
  **L2: for** $j = 0$ to $N_s - 1$ **do** ◁ **loop unroll**
    $square \leftarrow SV[i][j] * x[i]$;
    $p\_sum[j] \leftarrow p\_sum[j] + square$;
  **end for**
**end for**
**L3: for** $i = 0$ to $N_s - 1$ **do** ◁ **loop unroll**
  $temp[i] \leftarrow \beta[i] * p\_sum[i]$;
  $f\_sum \leftarrow f\_sum + temp[i]$;
**end for**
$f(x) \leftarrow f\_sum$;

where $T_a$ is the propagation delay of the floating point adder in clock cycles. Since the loop-carried dependence is removed by loop optimizations, the operations including memory read, compute and memory write are fully pipelined with initiation interval of 1 clock cycle. Thus, no additional memory access cost is involved in (4.2). Only the total number of loop iterations contributes to the latency of the nested loop. For

loop L3, the pipeline initiation interval is limited by the loop-carried dependence of the accumulator, so the latency $N_s \cdot T_a$ depends on both the number of loop iterations and the propagation delay of the accumulator in cycles. Compared with the original implementation in Algorithm 1, our proposed loop optimization method in Algorithm 2 has achieved a speedup of 7.75× at the same DSP utilization rate, demonstrating a great improvement of the pipeline efficiency.

Parallelization is another advantage after eliminating loop-carried dependence by loop distribution and interchange. In Algorithm 2, we know that $p\_sum[j]$ and $p\_sum[j+1]$ are calculated independently, thus we can unroll the loop L2 directly to increase the parallelism without changing the code structure. After unrolling, massive parallelized MAC units can be mapped to DSP slices on FPGA easily. Meanwhile, the same level of parallelism can also be applied to L3 with a cascaded MAC structure to accelerate the accumulation. Assume, we unroll L2 and L3 with a factor of $f$ simultaneously, as indicated in Algorithm 2, the total latency can be calculated as follows:

$$\text{Latency} = \underbrace{\frac{N_S \cdot M}{f}}_{\text{Partial sum}} + \underbrace{f \cdot T_a + \frac{N_S}{f} + L_{\text{tree}}(f)}_{\text{Final sum}}$$

$$L_{\text{tree}}(f) \approx \begin{cases} T_a \left\lceil \log_2 \dfrac{N_S}{f} \right\rceil, & f > \dfrac{N_S}{2f} \\ T_a \left\lceil \log_2 \dfrac{N_S}{f} \right\rceil + 2 \left\lceil \dfrac{N_s}{2f^2} \right\rceil - 2, & 1 < f < \dfrac{N_S}{2f} \end{cases} \tag{4.3}$$

where $L_{\text{tree}}(f)$ is the latency of the adder tree inside L3 after unrolling. $L_{\text{tree}}(f)$ has different expressions with small and large unroll factors, but in both cases, it has little effect on the total latency; therefore, it can be dropped safely in later analysis. The corresponding hardware structure for calculating the partial sum and final sum are shown in Figs. 6 and 7. According to Fig. 6, the latency of partial sum after unrolling is inversely proportional to the unroll factor *f,* since the parallel MAC array processes the multiply-accumulate operations concurrently. In Fig. 7, the latency of the final sum after unrolling consists of three terms, i.e., the latency of the cascaded MAC array, latency of feeding each partial sum to the MAC array and latency of the optimized adder tree, which are equal to $f \cdot T_a$, $N_s/f$ and $L_{\text{tree}}(f)$, respectively. The hardware structures are further discussed in detail in the next part. Note that the latency for $f = 1$ is calculated separately as (4.2). To study the effect of parallelization, we apply different unroll factors to loop L2 and L3 in Algorithm 2. The target platform is Xilinx ZCU104 and the working frequency is set to 200 MHz. All the input signals and intermediate values use single-precision floating point data type. The execution latency and speedup factor are collected from the Vivado HLS synthesis report, shown in Fig. 8(a). We can see that the latency for one regression decreases fast as the unroll factor increases, and the speedup almost scales linearly when the unroll factors are relatively small ($\leq$ 36). However, if we further increase the unroll factor, the linear scaling does not hold and the acceleration effect is weakened. When the unroll

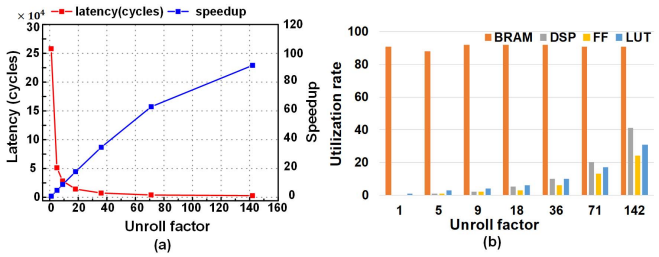



Fig. 8. (a) Speedup and latency versus unroll factor. (b) Hardware utilization rate on ZCU104 versus unroll factor.

factor increases to 142, the real speedup is about 92×. This can be explained by the following equation:

$$\text{Latency} \approx \begin{cases} \dfrac{N_s \cdot M}{f}, & \text{for small } f \\ \dfrac{N_s \cdot M}{f} + f \cdot T_a + \dfrac{N_s}{f}, & \text{for large } f. \end{cases} \tag{4.4}$$

For small unroll factor, the latency for final sum calculation in (4.3) is negligible compared with the partial sum, therefore, the latency is approximately inversely proportional to $f$. For large unroll factor, the latency of the adder chain within L3 is comparable to that of the partial sum, so the linear scaling does not hold anymore. The hardware consumption is shown in Fig. 8(b), we can see the DSP consumption scales linearly with the unroll factor, while the block-RAM (BRAM) consumption does not change much because the support vectors dominate most of the BRAM usage. The LUT and flip-flop (FF) consumptions are also proportional to the unroll factor. The results prove that area-performance trade-off can be easily achieved with the proposed optimization method.

### C. Batch Processing Method

From the experimental results in Fig. 8(a), we know that the latency can be greatly reduced with the proposed two-step optimization method and loop unroll, thus a notable speedup can be achieved. However, the linear scaling relationship is not valid for large unroll factors. If we want to achieve high parallelism with a large unroll factor, the latency of $f \cdot T_a$ of the long adder chain becomes prominent, since it is proportional to the unroll factor $f$. Under this circumstance, very deep pipeline stages of the adder chain in L3 will cause the MAC units under-utilized.

Batch processing method is common for GPU based acceleration, especially when used in the training of deep neural networks (DNNs) [25]. It can take advantage of the available on-chip memory to store multiple inputs and process them simultaneously in the pipeline. The hardware utilization efficiency can be significantly improved since the arithmetic units are no longer waiting for new inputs in idle state. Posewsky and Ziener [26] further extended this idea to FPGA-based DNN accelerator and achieved an order of magnitude improvement compared to existing methods.

To further improve the hardware utilization efficiency of L3 with large unroll factor, for the first time, we propose a batch processing method for SVR to process a batch of input vectors at a time. With batch processing, the nested

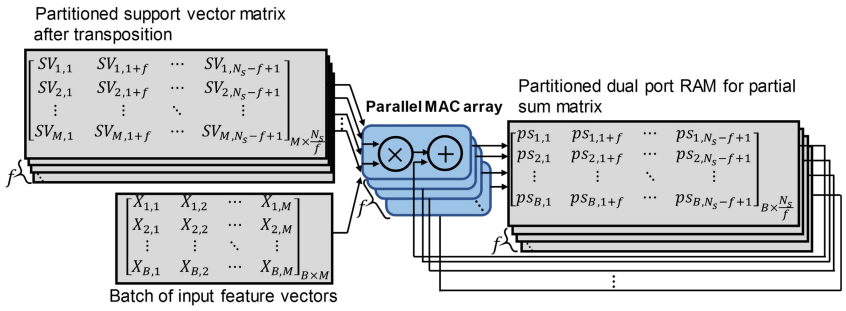



Fig. 9. Hardware structure for calculating the partial sum matrix with batch processing. The memory for support vector matrix and partial sum matrix are partitioned to $f$ individual blocks, respectively. Meanwhile, $f$ parallel MAC units keep updating the values of the partial sum iteratively.

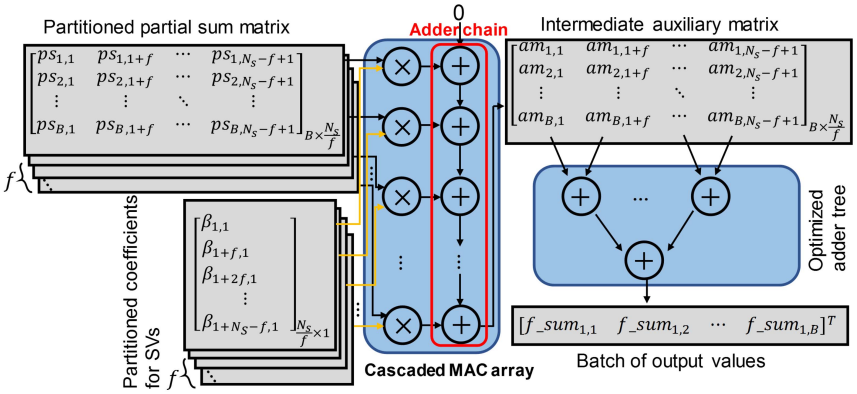



Fig. 10. Hardware structure for calculating the final outputs with batch processing. The coefficients for SVs are also partitioned to $f$ blocks as partial sum matrix to accommodate the parallel access of the cascaded MAC array. The output of the adder chain is written to the intermediate auxiliary matrix and accumulated through an optimized adder tree to generate the final output values.

loop L1 and L2 in Algorithm 2 become a three-level nested loop L1, L2, and L3 while the original L3 loop turns into a nested loop L4 and L5. The pseudocode for batch processing is shown in Algorithm 3. Note that no change to the existing computation structure is required for batch processing, more on-chip memory for storing batch of intermediate signals is the additional overhead. The total latency of Algorithm 3 with a large unroll factor $f$ can be calculated as follows:

$$\text{Total latency} \approx \underbrace{\frac{B \cdot N_S \cdot M}{f}}_{\text{Partial sum}} + \underbrace{f \cdot T_a + \frac{B \cdot N_S}{f}}_{\text{Final sum}} \tag{4.5}$$

where $B$ is the batch size. If we divide the total latency by $B$, the average latency of the adder chain inside L3 is now shared by $B$ inputs

$$\text{Average Latency} \approx \frac{N_s \cdot M}{f} + \frac{f \cdot T_a}{B} + \frac{N_s}{f}. \tag{4.6}$$

When $B$ increases, the average latency of the adder chain will decrease and finally we can have the approximate average latency as follows when $B$ is large enough:

$$\text{Average Latency} \approx \frac{N_s(M+1)}{f}. \tag{4.7}$$

In (4.7), we can see the latency is only dependent on the unroll factor $f$, which exhibits an inversely proportional relationship and the linear scaling of speedup holds.

The hardware structure for calculating the three-level nested loop L1-L2-L3 in Algorithm 3 is shown in Fig. 9. Compared with Fig. 6, a larger memory is required to store the batch of input feature vectors and partial sum matrix, while the parallel MAC array remains the same. To enable multiple access to the support vector matrix, array partition is performed to increase the memory bandwidth and the partition factor is equal to the unroll factor $f$. Moreover, the partitioned partial sum matrix is mapped to the dual-port RAM to enable simultaneous read and write operations. In every clock cycle, $f$ support vectors and one element from input vectors are read to the parallel MAC array, the accumulation results are written to the dual-port RAM concurrently. It takes totally $B \cdot N_s \cdot M/f$ cycles to finish updating the partial sum matrix. After this, the partial sum matrix will be used to calculate the final sum.

The hardware structure of the nested loop L4-L5 is presented in Fig. 10, which increases the memory consumption for the intermediate auxiliary matrix on the basis of Fig. 7. Different from the parallel MAC array shown in Fig. 9, the massive MAC units are reconstructed to a cascaded MAC array. In every clock cycle, $f$ elements from partial sum matrix and coefficients vector are fetched to the MAC array, while only one output is generated to the intermediate auxiliary matrix at a time. The long adder chain inside the MAC array is heavily pipelined to ensure the initiation interval of 1 clock cycle. It takes $B \cdot N_s/f$ cycles to feed all the inputs to the MAC array, however, the latency of the adder chain is

**Algorithm 3** Optimized Linear SVR With Loop Distribution, Loop Interchange and Batch Processing

**Input**: multiple feature vectors $x[B][M]$
**Require**: support vectors $SV[M][N_s]$, support vector corresponding multipliers $\beta[N_s]$, $bias$
**Output**: classification results $f(x[B])$
**Initialize**: $p\_sum[N_s] \leftarrow 0, f\_sum[B] \leftarrow bias$
**L1: for** $k = 0$ to $B-1$ **do**
  **L2: for** $i = 0$ to $M-1$ **do**
    **L3: for** $j = 0$ to $N_s - 1$ **do** ◁ **loop unroll**
      $square\ [k] \leftarrow SV[i][j] * x[k][i]$;
      $p\_sum[k][j] \leftarrow p\_sum[k][j] + square[k]$;
    **end for**
  **end for**
**end for**
**L4: for** $k = 0$ to $B-1$ **do**
  **L5: for** $i = 0$ to $N_s - 1$ **do** ◁ **loop unroll**
    $temp[k][i] \leftarrow \beta[i] * p\_sum[k][i]$;
    $f\_sum\ [k] \leftarrow f\_sum[k] + temp[k][i]$;
  **end for**
**end for**
$f(x[B]) \leftarrow f\_sum[B]$;

not negligible since it is directly proportional to the unroll factor $f$. After the intermediate auxiliary matrix is completely updated, an optimized adder tree will generate the final outputs in serial, the time consumption $B \cdot L_{\text{tree}}(f)$ of this adder tree is trivial since $N_S/f$ is normally very small for large unroll factors.

To verify the effectiveness of the proposed batch processing method, we apply different batch sizes on Algorithm 3. The unroll factor of 284 is chosen to maximize the use of the available DSP resources on ZCU104. The latency and speedup versus batch size are shown in Fig. 11(a). We can see that the latency decreases rapidly along with the increase in batch size, and finally converges to about 900 clock cycles. Meanwhile, the speedup increases along with the batch size, and the maximum speedup achieved is 275× with the batch size of 40. The hardware utilization is shown in Fig. 11(b). We can see that the DSP, BRAM and FF usage does not change much when the batch size increases. Only the LUT consumption slightly increases, since the storage requirement for intermediate values like partial sum matrix and intermediate auxiliary matrix is proportional to the batch size. The overall hardware utilization for large batch size does not impose a heavy burden on the resources, which proves our proposed batch processing method is also area-efficient for hardware implementation.

### *D. Implementation Results on ZC706 and ZCU104*

Next, we implement linear SVR decision function on two different FPGA platforms based on the proposed optimization methods. Two FPGA boards are Xilinx ZC706 and ZCU104, as shown in Fig. 12, the corresponding chips are Xilinx Zynq XC7Z045 and Zynq UltraScale+ ZU7EV, respectively. The post-implementation resource utilization is shown in Table II. It can be observed that the resources are used adequately for

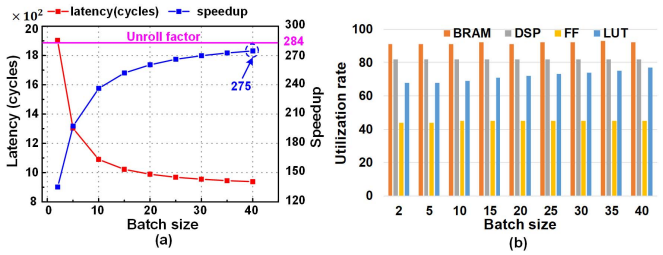



Fig. 11. (a) Speedup and latency versus batch size. (b) Hardware utilization rate on ZCU104 versus batch size.


Fig. 12. FPGA boards of (a) Xilinx ZC706 and (b) ZCU104.

TABLE II
POST-IMPLEMENTATION RESOURCE UTILIZATION OF ZC706 AND ZCU104

| | Xilinx ZC706 | | | Xilinx ZCU104 | | |
|---|---|---|---|---|---|---|
| | Used | Available | Utilization rate (%) | Used | Available | Utilization rate (%) |
| BRAM | 290.5 | 545 | 53.30 | 286 | 312 | 91.67 |
| DSP | 710 | 900 | 78.89 | 1421 | 1728 | 82.23 |
| LUT | 111415 | 218600 | 50.97 | 149623 | 230400 | 64.94 |
| FF | 73213 | 437200 | 16.75 | 199529 | 460800 | 43.30 |

both platforms, while DSP and BRAM resources are the main constraints since they determine the maximum parallelism degree. The performances of two FPGA boards are shown in Table III, which also includes a software implementation based on widely used LIBSVM running on a windows desktop with i7-5960$x$ [27] CPU and 32-GB RAM. From Table III, we can see that the software implementation with LIBSVM needs 19.41 s for the post-processing of 96 100 BGSs from 38.44-km FUT when it works at 3 GHz, taking up 18% ∼ 87.8% of total MT. On the contrast, our implementation with ZC706 can complete the post-processing in 1.98 s, while the power consumption of the FPGA development board is only 14.43 W when it works at 100 MHz, taking up 2.2% ∼ 42.3% of MT. Furthermore, the implementation with ZCU104 completes the post-processing in 0.46 s when it works at 200 MHz, taking up 0.52% ∼ 14.5% of MT. The power consumption is 26.5 W. The working frequency difference between ZC706 and ZCU104 is due to the different manufacturing technology by the two FPGAs, and advanced technology can enable higher working frequency. The equivalent performance of the three platforms is 2.48, 24.3, and 104 GFLOPS, respectively. Since ZCU104 provides 1.92× DSP resources over ZC706, the unroll factor for ZCU104 can be doubled compared with ZC706. The results prove that the hardware accelerators can

TABLE III

PERFORMANCE COMPARISON BETWEEN SOFTWARE IMPLEMENTATION AND TWO FPGA PLATFORMS

| Platform | Intel i7-5960x | Xilinx ZC706 | Xilinx ZCU104 |
|---|---|---|---|
| Technology | 22nm | 28nm | 16nm |
| Frequency | 3.0 GHz | 100 MHz | 200 MHz |
| Power | 140 W | 14.43 W | 26.50 W |
| Running time(sec) | 19.41 | 1.98 | 0.46 |
| $T_{pp}/T$ (%) | 18~87.8% | 2.2~42.3% | 0.52~14.5% |
| Performance (GFLOPS) | 2.48 | 24.3 | 104 |
| Energy efficiency | 1x | 95.1x | 221.6x |

achieve real-time post-processing for the BOTDA data, which are 9.8× and 42× faster than the software implementation. Meanwhile, we also evaluate the energy efficiency as (4.8), where energy = power × running time. The two FPGA implementations achieve 95.1× and 226.1× energy efficiency compared with i7-5960$x$, which could save plenty of energy in all-day monitoring environments

$$\text{Energy efficiency} = \frac{\text{Energy consumed by CPU}}{\text{Energy consumed by the target accelerator}}. \quad (4.8)$$

We also compare the performances of our FPGA accelerator with a recent study [11]. Abbasnejad and Alizadeh [11] adopt an XC-based method to extract the BFS information. Since the computation complexity of XC is proportional to the square of frequency number of the input BGS, the authors simplify the original algorithm through a moving average filter to narrow the search region at the cost of reduced estimation accuracy. The time consumption of processing 96100 BGSs in [11] is 0.33 s, corresponding to 14.7 GOPS equivalent performance. Although our accelerator costs 0.13 s longer than [11], considering the absolute performance of our accelerator is 104 GFLOPS which is about 7× higher than [11], the increased time consumption is trivial and will not pose any burden in real applications. Besides, the floating-point data type in our design can provide much higher precision and dynamic range than a fixed point in [11], and the quantization process is also eliminated. Moreover, our method does not involve any pre-processing like interpolation and moving average as in [11] to reduce the computation complexity, hence, the overall workflow is more concise. In summary, the performance of our BOTDA fiber sensor accelerator is competitive regarding both the time consumption and the absolute performance.

Besides, we list our accelerator and several recent FPGA-based support vector machine (SVM) implementations in Table IV. Although our SVR model is 10.4× and 12.2× larger than [28] and [29], the average time consumption for a single classification or regression of our accelerator is only 9% and 62% of [28] and [29]. Reference [30] has a similar model size with us since it contains five independent classes, but its average time consumption for a single classification is 52.2× longer than ours.

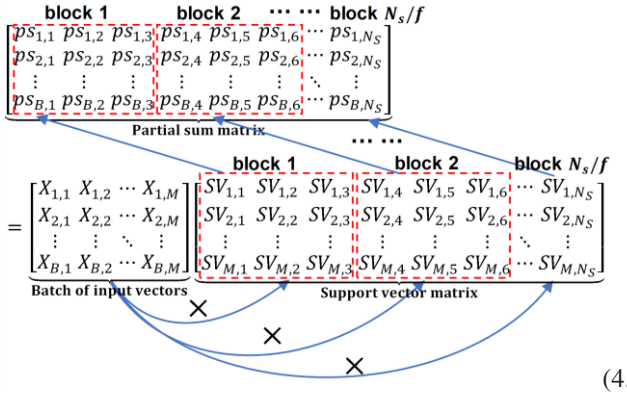
TABLE IV

COMPARISON WITH OTHER FPGA BASED SVM IMPLEMENTATIONS

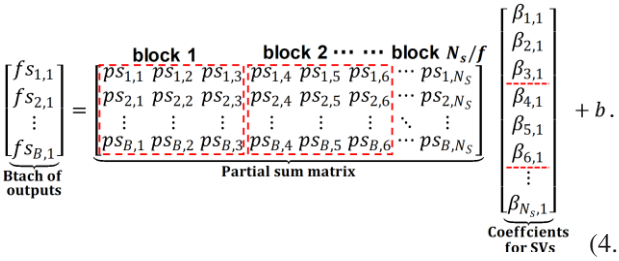
| | EMBC'13 [29] | TCI'15 [30] | JSPS'17 [28] | Proposed |
|---|---|---|---|---|
| Device model | Xilinx XC4VSX35 | Xilinx XC5VLX110T | Xilinx XC7Z020 | Xilinx XCZU7EV |
| Task | Microarray classification | Image classification | Arrhythmia detection | Temperature extraction |
| Number of support vectors | 20 | 100 | 1274 | 1136 |
| Feature dimension | 1024 | 500 | 18 | 220 |
| Frequency (MHz) | 137.7 | 50 | 25 | 200 |
| Time consumption (sec) | $7.64 \times 10^{-6}$ | $2.5 \times 10^{-4}$ | $5.12 \times 10^{-5}$ | $4.79 \times 10^{-6}$ |

### *E. Theoretical Analysis and Discussion*

In Sections IV-B and IV-C, we have systematically optimized the original linear SVR decision function for hardware implementation. Loop distribution and loop interchange enable efficient pipeline strategy to be used for partial sum calculation, loop unroll further greatly reduces the latency through parallelizing the MAC operations. Furthermore, the batch processing method makes the latency of the long adder chain shared by multiple inputs, which makes the linear scaling of speedup holds approximately. These optimization techniques make the SVR decision function very suitable to be mapped to FPGA, which are also reflected in the hardware structures in Figs. 9 and 10. If we further analyze Algorithm 3, we can find that we have actually transformed the partial sum matrix calculation and final sum vector calculation to matrix-matrix multiplication and matrix-vector multiplication as follows:




(4.9)




(4.10)

For matrix–matrix multiplication in (4.9), we tile the support vector matrix into small blocks and the input vectors multiply each block in serial. The partial sum matrix is also tiled accordingly. For the matrix-vector multiplication in (4.10),

the coefficients vector for support vectors also needs to be partitioned to maintain the same level of parallelism. As a result, the two operations are both heavily parallelized, which could take advantage of massive DSP resources and dual-port RAMs on FPGA. To be more specific, the parallel MAC array for matrix–matrix multiplication and cascaded MAC array for matrix–vector multiplication are based on the same amount of DSP resources, making our implementation achieve very high hardware utilization efficiency since almost no DSP resources are idle during the computation.

## V. Conclusion

In this article, a new temperature prediction method for BOTDA fiber sensor systems based on SVR is proposed. We experimentally verify that SVR can achieve comparable performances as SVC, XC, LCF, and ANN under different SNRs. From the hardware perspective, SVR is more hardware friendly than the other four methods without modifications and complicated pre-processing. To accelerate the processing speed of SVR, the linear SVR decision function is optimized systematically. The loop-carried dependence in loop iterations is eliminated by loop distribution and loop interchange. Therefore, the pipeline efficiency of the nested loop is greatly improved. We also propose a batch processing method to further decrease the latency. Using the proposed optimization methods, linear SVR decision function is implemented on two FPGA boards Xilinx ZC706 and ZCU104 to process 96 100 BGSs from 38.44-km FUT acquired from a BOTDA system. Our hardware accelerator can achieve up to 42× speedup compared with the software implementation with i7-5960*x* CPU. The post-processing time for 96 100 BGSs along 38.44-km FUT is only 0.46 s with ZCU104, which makes our implementation capable of real-time prediction. Meanwhile, the power consumption of FPGA is also much lower than a high-end CPU, making the energy efficiency of our FPGA implementation up to 226.1× higher than the software implementation based on LIBSVM.

## References


[1] A. Barrias, J. R. Casas, and S. Villalba, "A review of distributed optical fiber sensors for civil engineering applications," *Sensors*, vol. 16, no. 5, p. 748, 2016.

[2] E. Buchoud *et al.*, "Quantification of submillimeter displacements by distributed optical fiber sensors," *IEEE Trans. Instrum. Meas.*, vol. 65, no. 2, pp. 413–422, Feb. 2016.

[3] M. J. Garcia, J. A. Ortega, J. A. Chavez, J. Salazar, and A. Turo, "A novel distributed fiber-optic strain sensor," *IEEE Trans. Instrum. Meas.*, vol. 51, no. 4, pp. 685–690, Aug. 2002.

[4] C. Wang and K. Shida, "A low-cost double-fiber model distributed optical fiber sensor," *IEEE Trans. Instrum. Meas.*, vol. 56, no. 4, pp. 1481–1487, Aug. 2007.

[5] T. Kurashima, T. Horiguchi, and M. Tateda, "Distributed-temperature sensing using stimulated Brillouin scattering in optical silica fibers," *Opt. Lett.*, vol. 15, no. 18, pp. 1038–1040, Sep. 1990.

[6] C. Li and Y. Li, "Fitting of Brillouin spectrum based on LabVIEW," in *Proc. 5th Int. Conf. Wireless Commun., Netw. Mobile Comput.*, Sep. 2009, pp. 1–4.

[7] C. Zhang, Y. Yang, and A. Li, "Application of Levenberg-Marquardt algorithm in the Brillouin spectrum fitting," *Proc. SPIE*, vol. 7129, Oct. 2008, Art. no. 71291Y.

[8] M. A. Soto, S. Le Floch, and L. Thévenaz, "Bipolar optical pulse coding for performance enhancement in BOTDA sensors," *Opt. Express*, vol. 21, no. 14, pp. 16390–16397, Jul. 2013.

[9] A. Denisov, M. A. Soto, and L. Thévenaz, "Going beyond 1000000 resolved points in a Brillouin distributed fiber sensor: Theoretical analysis and experimental demonstration," *Light, Sci. Appl.*, vol. 5, no. 6, 2016, Art. no. e16074.

[10] M. A. Farahani, E. Castillo-Guerra, and B. G. Colpitts, "Accurate estimation of Brillouin frequency shift in Brillouin optical time domain analysis sensors using cross correlation," *Opt. Lett.*, vol. 36, no. 21, pp. 4275–4277, 2011.

[11] M. Abbasnejad and B. Alizadeh, "FPGA-based implementation of a novel method for estimating the Brillouin frequency shift in BOTDA and BOTDR sensors," *IEEE Sensors J.*, vol. 18, no. 5, pp. 2015–2022, Mar. 2018.

[12] A. K. Azad, L. Wang, N. Guo, H.-Y. Tam, and C. Lu, "Signal processing using artificial neural network for BOTDA sensor system," *Opt. Express*, vol. 24, no. 6, pp. 6769–6782, 2016.

[13] H. Wu, L. Wang, N. Guo, C. Shu, and C. Lu, "Brillouin optical time-domain analyzer assisted by support vector machine for ultrafast temperature extraction," *J. Lightw. Technol*, vol. 35, no. 19, pp. 4159–4167, Oct. 1, 2017.

[14] H. Wu, L. Wang, N. Guo, C. Shu, and C. Lu, "Support vector machine assisted BOTDA utilizing combined Brillouin gain and phase information for enhanced sensing accuracy," *Opt. Express*, vol. 25, no. 25, pp. 31210–31220, 2017.

[15] H. Wu, L. Wang, Z, Zhao, C. Shu, and C. Lu, "Support vector machine based differential pulse-width pair Brillouin optical time domain analyzer," *IEEE Photon. J.*, vol. 10, no. 4, Aug. 2018, Art. no. 6802911.

[16] G. Gabriel, C. Jara, J. Pomares, A. Alabdo, L. Poggi, and F. Torres, "A survey on FPGA-based sensor systems: Towards intelligent and reconfigurable low-power sensors for computer vision, control and signal processing," *Sensors*, vol. 14, no. 4, pp. 6247–6278, 2014.

[17] J. Cong, B. Liu, S. Neuendorffer, J. Noguera, K. Vissers, and Z. Zhang, "High-level synthesis for FPGAs: From prototyping to deployment," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 4, pp. 473–491, Apr. 2011.

[18] B. A. Draper, J. R. Beveridge, A. P. W. Bohm, C. Ross, and M. Chawathe, "Accelerated image processing on FPGAs," *IEEE Trans. Image Process.*, vol. 12, no. 12, pp. 1543–1551, Dec. 2003.

[19] H. Wu, H. Wang, C. Shu, C.-S. Choy, and C. Lu, "Brillouin optical time domain analyzer fiber sensor based on FPGA accelerated support vector regression," in *Proc. Opt. Fiber Commun. Conf. Exhib. (OFC)*, Mar. 2019, pp. 1–3.

[20] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, 2011.

[21] L. Bottou and C.-J. Lin, "Support vector machine solvers," *Large Scale Kernel Mach.*, vol. 3, no. 1, pp. 301–320, 2007.

[22] N. Guo, L. Wang, H. Wu, C. Jin, H.-Y. Tam, and C. Lu, "Enhanced coherent BOTDA system without trace averaging," *J. Lightw. Technol.*, vol. 36, no. 4, pp. 871–878, Feb. 15, 2018.

[23] M. A. Soto, J. A. Ramírez, and L. Thévenaz, "Intensifying the response of distributed optical fibre sensors using 2D and 3D image restoration," *Nature Commun.*, vol. 7, Mar. 2016, Art. no. 10870.

[24] H. Wang, W. Shi, and C.-S. Choy, "Hardware design of real time epileptic seizure detection based on STFT and SVM," *IEEE Access*, vol. 6, pp. 67277–67290, 2018.

[25] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. ACMMM*, vol. 2014, pp. 675–678.

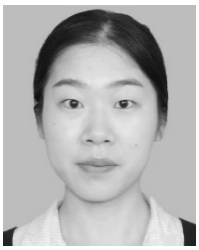
[26] T. Posewsky and D. Ziener, "Efficient deep neural network acceleration through FPGA-based batch processing," in *Proc. Int. Conf. ReConFigurable Comput. FPGAs (ReConFig)*, Cancun, Mexico, 2016, pp. 1–8.

[27] *Intel Core i7-5960X Processor Extreme Edition*. Accessed: Aug. 14, 2019. [Online]. Available: https://ark.intel.com/content/www/us/en/ark/products/82930/intel-core-i7-5960x-processor-extreme-edition-20m-cache-up-to-3-50-ghz.html

[28] V. Tsoutsouras, K. Koliogeorgi, S. Xydis, and D. Soudris, "An exploration framework for efficient high-level synthesis of support vector machines: Case study on ECG arrhythmia detection for Xilinx Zynq SoC," *J. Signal Process. Syst.*, vol. 88, no. 2, pp. 127–147, 2017.

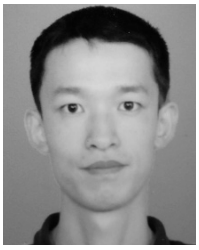
[29] H. M. Hussain, K. Benkrid, and H. Seker, "Reconfiguration-based implementation of SVM classifier on FPGA for Classifying Microarray data," in *Proc. 35th Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Osaka, Japan, Jul. 2013, pp. 3058–3061.

[30] M. Qasaimeh, A. Sagahyroon, and T. Shanableh, "FPGA-based parallel hardware architecture for real-time image classification," *IEEE Trans. Comput. Imag.*, vol. 1, no. 1, pp. 56–70, Mar. 2015.



**Huan Wu** received the Ph.D. degree in electronic engineering from The Chinese University of Hong Kong, Hong Kong, in 2018.

She is currently a Post-Doctoral Fellow with the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong. Her current research interests include distributed optical fiber sensing and digital signal processing for sensing systems.



**Hongda Wang** received the B.S. degree in electronic engineering from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2013, and the M.S. degree in electronic engineering from The Chinese University of Hong Kong, Hong Kong, in 2014, where he is currently pursuing the Ph.D. degree.

His current research interests include neural signal processing, field-programmable gate array, and deep learning.

**Chester Shu** (M'91–SM'02) received the Ph.D. degree in applied physics from Columbia University, New York, NY, USA, in 1991, where he performed graduate research on ultrafast optoelectronics.

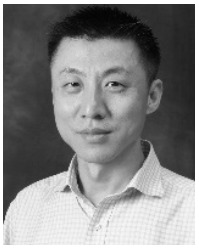
He was involved in the design and characterization of multiquantum well lasers during his internship at Siemens Corporate Research Labs, and in integrated polymeric electrooptic modulators at a start-up in Bothell, WA, USA. He is currently a Professor with the Center for Advanced Research in Photonics, Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong. He has made original contributions in nonlinear fiber optics and ultrafast processing of optical communication signals. In particular, he developed new approaches of Raman-enhanced and Brillouin-enhanced optical processing that offered exceptional benefits in the processing bandwidth, efficiency, adaptability, and signal quality. He conducted research in applying specialty fibers and new fiber and semiconductor-based devices to process single and multiwavelength signals with diverse bit rates, modulation formats, propagation modes, and polarizations. He has authored more than 400 articles and has delivered a number of talks in major technical conferences. His current research interests include the generation, processing, detection, and characterization of optical signals for advanced communication and fiber sensing systems.

Dr. Shu has been an Organizer/Technical Program Committee Member of many conferences, including the Optical Fiber Communication Conference and Exposition (OFC), the IEEE Lasers and Electro-Optics Society (LEOS), the Asia Communications and Photonics Conference (ACP), the OptoElectronics and Communications Conference (OECC), the Photonics Global Conference (PGC), the International Conference on Photonics in Switching and Computing (PS), and the Australian Conference on Optical Fiber Technology (ACOFT). He was the Chairman of the IEEE Hong Kong LEOS (now Photonics Society) Chapter in 2006 and 2007. He is currently a Subject Editor (Optical Communications) of *Electronics Letters* and an Associate Editor of the IEEE PHOTONICS JOURNAL. He also served as a Topical Editor of the *Optics Letters* in optical fiber communications from 2005 to 2011.

**Chiu-Sing Choy** (SM'96) received the B.Sc., M.Sc., and Ph.D. degrees in electrical and electronics engineering from The University of Manchester, Manchester, U.K., in 1983, 1984, and 1987, respectively.

In 1985, he spent a year with Ferranti Microelectronics, Oldham, U.K., participating in ASIC technology research. In 1986, he joined the Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong, where he is currently a Professor. He is also an External Examiner for Multimedia University, Cyberjaya, Malaysia, and an External Reviewer for The Chinese University of Hong Kong, Shenzhen, China.

Dr. Choy is currently a fellow of The Hong Kong Institution of Engineers (HKIE), Hong Kong. He was the Chairperson of the Electronics Division in 2006/2007 and a Council Member of HKIE. He serves in many international conferences and is currently the Co-Chair of the International Forum on Mobile Technology and Applications (IFMTA). He is active in supporting the local electronics industry. He led a number of ITF supported industry projects in optical pick-up designs, RFID technology, and structured ASIC. He has published widely with over hundreds of articles to his credit.



**Chao Lu** received the B.Eng. degree in electronic engineering from Tsinghua University, Beijing, China, in 1985, and the M.Sc. and Ph.D. degrees from The University of Manchester, Manchester, U.K., in 1987 and 1990, respectively.

He joined the School of Electrical and Electronic Engineering, Nanyang Technological University (NTU), Singapore, in 1991, where he has been a Lecturer, a Senior Lecturer, and an Associate Professor until 2006. From June 2002 to December 2005, he was with the Institute for Infocomm Research, Agency for Science, Technology and Research (A∗STAR), Singapore, as the Program Director and Department Manager leading a Research Group in the area of optical communication and fiber devices. He joined the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong, in 2006, where he is currently a Chair Professor of fiber optics. Over the years, he has published more than 300 articles in major international journals. In addition to academic research work, he has had many industrial collaborative research projects and has a number of awarded patents. His current research interests include high-capacity transmission techniques for long haul and short reach systems and distributed optical sensing systems.

Dr. Lu is a fellow of the Optical Society (OSA). He has been an organizer or a technical program committee member of many international conferences.