JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 33, 51-62 (2017)
DOI: 10.6688/JISE.2017.33.1.4

Hardware Implementation of Local Mean Decomposition*

PEI-YIN CHEN, YEN-CHEN LAI AND PING-HSUAN LAI
Department of Computer Science and Information Engineering
National Cheng Kung University
Tainan, 701 Taiwan
E-mail: {pychen; P78001324; P76014101}@mail.ncku.edu.tw

Local mean decomposition (LMD) is an effective signal analysis method for analyzing nonlinear and nonstationary signals. LMD has been usefully applied in a wide variety of applications. However, achieving real-time LMD calculations in software is difficult. In this paper, a flexible, low-cost, and high-performance hardware architecture for LMD is proposed that satisfies the real-time requirements of various LMD applications. All proposed circuits were developed using Verilog and then synthesized using the Synopsys Design Compiler with the Taiwan Semiconductor Manufacturing Company 0.18-μm cell library. With the help of parameterization, the proposed LMD circuit can easily be used for various applications and hardware architectures.

Keywords: hardware, local mean decomposition (LMD), real-time, signal processing, demodulation

1. INTRODUCTION

Most real-world data are nonlinear and nonstationary [1], but conventional data analysis methods, such as the Fourier transform, are ineffective for nonlinear and nonstationary data. The Hilbert-Huang transform (HHT) [1] is an effective data analysis method for analyzing nonlinear and nonstationary signals. Empirical mode decomposition (EMD), the major component of HHT, can decompose a signal into a set of scaled data sequences and a final residue. EMD has been widely used for data analysis [2-5]. However, the EMD process loses frequency and amplitude information [6]. To solve these problems, local mean decomposition (LMD) [6], an adaptive signal decomposition method, has been proposed.

LMD has been widely used for analyzing nonlinear and nonstationary signals, such as radar image processing [7], biosignal processing [8], blind source separation [9, 10], and rolling bearing fault diagnosis [11, 12]. Although the analysis results of LMD are more stable and precise than those of EMD [7], the sifting process of LMD requires considerable time to generate frequency-modulated (FM) signals. Writing software that achieves real-time LMD calculations is difficult. However, real-time analysis and reaction are necessary in many applications, particularly safety-related applications such as biosignal detection and mechanical fault diagnosis. Hardware implementations can operate in parallel to reduce the overall computation time [13-15]. Therefore, a high-performance hardware architecture for accelerating LMD calculations must be developed. In practice, different LMD applications [6-12] have different timing requirements and require various data lengths, data widths, and precisions. To solve these problems, a flexi-

Received November 27, 2015; revised March 7, 2016; accepted March 13, 2016.
Communicated by Jiann-Liang Chen.
* This work was supported in part by the National Science Council of Taiwan under Grant NSC 101-2221-E-006-151-MY3.

ble, low-cost, and high-speed design for LMD is proposed in this paper.

According to our review of the literature, this is the first complete LMD hardware design. All modules of the proposed LMD circuit described in this paper were developed in the Verilog hardware description language (HDL). Variables in the proposed LMD circuit are as parameterized as possible. Users can choose the most suitable design and architecture for different LMD applications. With the optimization and pipeline scheme, the proposed LMD circuit is very suitable for high-speed LMD applications, and can easily be embedded into other hardware systems.

The remainder of this paper is organized as follows: The LMD algorithm is described in Section 2. The hardware architecture is presented in Section 3. Section 4 illustrates the experimental results and the statistical data. The conclusions are provided in Section 5.

2. LMD ALGORITHM

LMD decomposes a signal into a finite set of product functions (PFs) and a residue. Each PF is a product of an amplitude envelope and an FM signal. Analysis of the PF easily yields signal information such as instantaneous amplitude and instantaneous frequency.

In LMD, an original signal $x(t)$ can be decomposed as

$$x(t) = \sum_{i=1}^{k} pf_i(t) + u(t), \tag{1}$$

where k is the total number of PFs, $pf_i(t)$ is the ith PF, and $u(t)$ is the final residue. The LMD procedure is summarized in the following steps:

Step 1: Initialize the settings as follows: i=1, j=1, $u_{i-1}(t)=x(t)$, and $s_i^{j-1}(t) = x(t)$, where i is the current PF count, j is the current iteration count, $u_{i-1}(t)$ is the residue after i–1 PFs have been extracted, and $s_i^{j-1}(t)$ is the candidate FM signal generated in the previous iteration.

Step 2: Identify the local extrema of $s_i^{j-1}(t)$. Calculate the local mean and local magnitude from each two successive extrema.

Step 3: Generate the successive local mean function (SLMF) and successive envelope estimate (SEE) by extending the values of local means and local magnitudes between successive extrema, respectively.

Step 4: Use the moving average (MA) to obtain the smoothly varying SLMF (SVSLMF) $m_i^j(t)$ and the smoothly varying SEE (SVSEE) $a_i^j(t)$ by smoothing the SLMF and the SEE, respectively.

Step 5: Sift mean and demodulate amplitude: $s_i^j(t)=(s_i^{j-1}(t)-m_i^j(t))/a_i^j(t)$, where $s_i^j(t)$ is the new candidate FM signal generated in the current decomposition iteration.

Step 6: Review the stopping criterion. If the stopping criterion is not satisfied, then proceed to the next iteration by letting $j=j+1$ and repeating Steps 2-6. If the stopping criterion is satisfied, then compute the ith PF $pf_i(t)$ and proceed to Step 7.

Step 7: Update the residue $u_i(t)$ by subtracting the obtained PF function $pf_i(t)$ from the previous step: $u_i(t)=u_{i-1}(t)-pf_i(t)$, where $u_i(t)$ is the residue after the previous i PFs have been extracted.

Step 8: Determine whether $u_i(t)$ becomes a constant or contains no more oscillations. If $u_i(t)$ is not a constant and contains oscillations, then let $i=i+1$, $j=1$, $s_i^{j-1}(t)= u_{i-1}(t)$, and repeat Steps 2-8 to identify the remaining PFs. If $u_i(t)$ becomes a constant or contains no more oscillations, then let the final residue be $u(t)=u_i(t)$ and stop the LMD procedure because no more PFs can be obtained.

3. PROPOSED DESIGN FOR LMD

A low-cost and high-performance LMD architecture was developed in Verilog HDL, and each module was synthesized using the Synopsys Design Compiler with the Taiwan Semiconductor Manufacturing Company (TSMC) 0.18-μm cell library. The synthesized circuit was then laid out and verified using the Synopsys IC Compiler for design rule check (DRC), layout versus schematic (LVS) check, and electrical rule check. Users can set parameters and obtain functional LMD circuits that satisfy their parameter settings. This flexible design enables users to combine the proposed LMD circuit with other practical circuits easily for various real-time LMD applications.

In this section, the hardware architecture of the proposed design is discussed in detail. Fig. 1 shows a state diagram of the proposed LMD; Table 1 lists the required cycle counts to finish the operations of each state, where T is the data length of signals and *iter* is the iteration count of S2. The iteration count *iter* depends on the signal qualities. Fig. 2 shows the system architecture, which is composed of data memories (FM signal and residue), a control unit, a feature extractor, an interpolating module, a smoothing module, and a stopping controller. The control unit monitors the data flow and sends control signals to all other modules. It sends signals to control timing statuses and schedules the reading and writing statuses of the data. The four main blocks, namely the feature extractor, the interpolating module, the smoothing module, and the stopping controller, which correspond to Steps 2-4 and 6 of the LMD procedure, are described in detail in the following subsections.

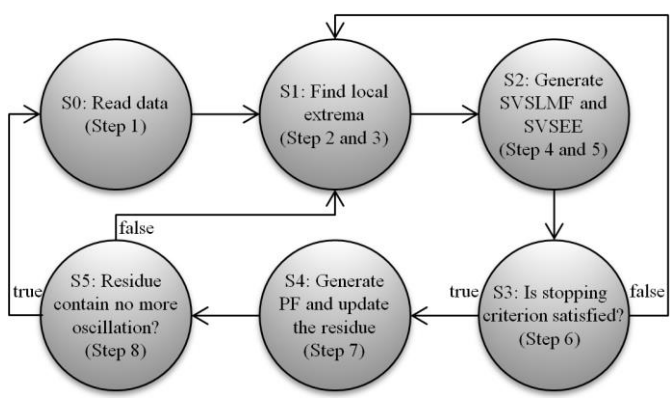

Fig. 1. State diagram of the LMD hardware.

Table 1. Required clock cycle numbers of each state.

State	S0, S1, S3, S4, S5	S2
Required Clock Cycles	*T*	*T*∗*iter*

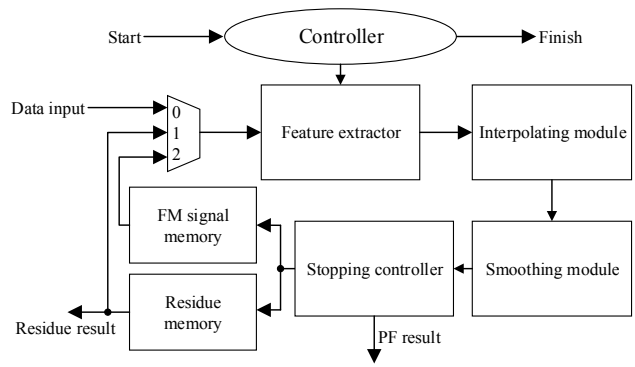

Fig. 2. System architecture.

3.1 Feature Extractor

In LMD, the first step is to extract the local means and local magnitudes. The feature extractor contains two subblocks, namely the extrema extractor and the feature generator. The extrema extractor uses a moving search window to identify the local extrema, then the feature generator computes local means and local magnitudes.

The evaluated sample is treated as a local extremum if the sample is the minimum or maximum within the search window. The equation that extracts local extrema is given as

$$\begin{cases} s_i^{j-1}[x] \text{ is a minimum}, & s_i^{j-1}[x] < s_i^{j-1}[x+k] \ \forall\ 1 \le |k| \le w \\ s_i^{j-1}[x] \text{ is a maximum}, & s_i^{j-1}[x] > s_i^{j-1}[x+k] \ \forall\ 1 \le |k| \le w, \\ s_i^{j-1}[x] \text{ is a not an extremum}, & \text{otherwise} \end{cases} \tag{2}$$

where x is the location index of the current sample to be evaluated and $2w+1$ is the search window size. Fig. 3 shows the hardware architecture of the extrema extractor, where the search window size is five.

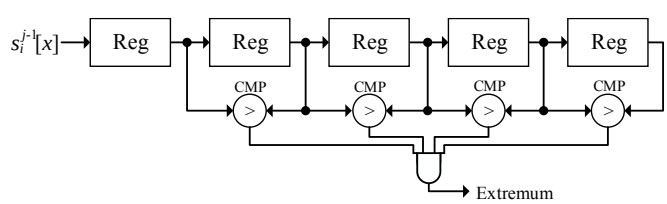

Fig. 3. Hardware architecture of the extrema extractor.

The lth local mean value m_l of each two successive extrema values e_l and e_{l+1} is given as

$$m_l = \frac{e_l + e_{l+1}}{2}, \tag{3}$$

and the lth local magnitude value a_l is given as

$$a_l = \left| \frac{e_l - e_{l+1}}{2} \right|. \tag{4}$$

Observing the similarity between Eqs. (3) and (4), the computation of m_l and a_l can be combined for resource sharing. Fig. 4 shows the hardware architecture of the feature generator.

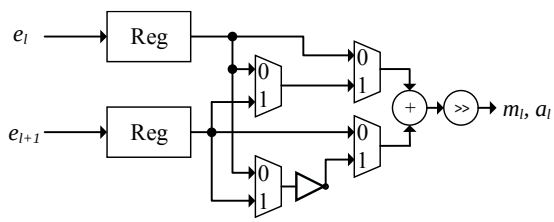

Fig. 4. Hardware architecture of the feature generator.

3.2 Interpolating Module

After the local means and local magnitudes have been obtained, the interpolating module is used to generate the SLMF and SEE. In the interpolating module, all local means and local magnitudes are interpolated as straight lines extending between each two successive extrema to calculate the SLMF and SEE, respectively. As with the feature generator, the computations of the SLMF and SEE can be combined for resource sharing. Fig. 5 shows the hardware architecture of the interpolating module.

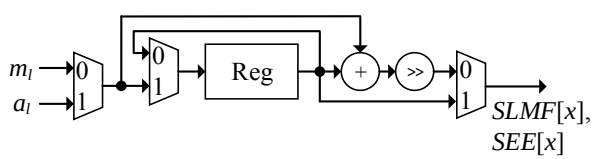

Fig. 5. Hardware architecture of the interpolating module.

3.3 Smoothing Module

The smoothing module is the performance bottleneck in LMD. In the smoothing module, the SLMF and SEE are smoothed using the MA equation to form the SVSLMF and SVSEE, respectively. The MA equation is defined as

$$MAV[x] = MAV[x-1] + \frac{P_M}{N} - \frac{P_{M-N}}{N}, \tag{5}$$

where $MAV[x]$ is the moving average value to be obtained, $MAV[x-1]$ is the previous moving average value, N is the moving window size, P_M is the newest value in the MA queue, and P_{M-N} is the oldest value in the previous MA queue. Eq. (5) shows that dividers are necessary for the MA. However, conventional dividers require long computation times and may considerably reduce the clock rate of an LMD circuit. Therefore, the smoothing module was optimized to prevent these problems in the present research.

First, the calculations of the dividers are eliminated. When the division is rewritten as multiplication of the dividend with the reciprocal of the divisor, Eq. (5) can be simplified as

$$\begin{cases} SUM[x] = SUM[x-1] + P_M - P_{M-N} \\ MAV[x] = SUM[x] * \dfrac{1}{N} \end{cases}, \tag{6}$$

where $SUM[x]$ is the summation of incoming values and $1/N$ is a predefined constant value.

Second, a four-stage pipeline architecture is applied to accelerate the clock rate of the smoothing module. At the first stage, the smoothing module stores the feedback values and the incoming values. The second stage computes $SUM[x]$. The third stage computes $MAV[x]$. At the final stage, the smoothing module normalizes $MAV[x]$ and obtains the smoothed value.

Third, the smoothing processes for the SLMF and the SEE are designed in parallel to reduce execution time. Fig. 6 shows the hardware architecture of the smoothing module.

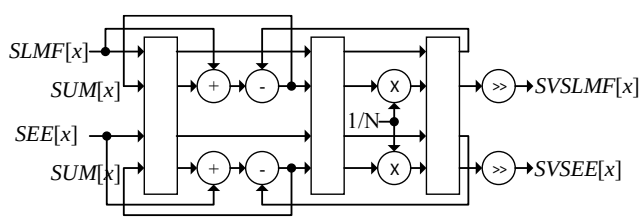

Fig. 6. Hardware architecture of the smoothing module.

3.4 Stopping Controller

The stopping controller reviews the SVSEE, stops the FM iteration, and computes the PFs. It contains three subblocks: the SVSEE checker, the FM normalizer, and the PF generator.

The SVSEE checker determines whether the stopping criterion has been satisfied. The candidate FM signal $s_i^j(t)$ is considered to be a pure FM signal $s_i^n(t)$ if

$$\lim_{n\to\infty} a_i^n(t) = 1. \tag{7}$$

The stopping criterion is satisfied when the candidate FM signal becomes a pure FM signal. Obviously, the computation process of the limit is complicated and unsuitable for hardware implementation. Moreover, the iterative operation can consume considerable computation time. Hence, an approximation method replaces Eq. (7) in the SVSEE checker. The equation used in the SVSEE checker is written as

$$1-\delta < a_i^n[x] < 1+\delta, \tag{8}$$

where δ is a user-defined error tolerance value. Fig. 7 shows the hardware architecture of the SVSEE checker.

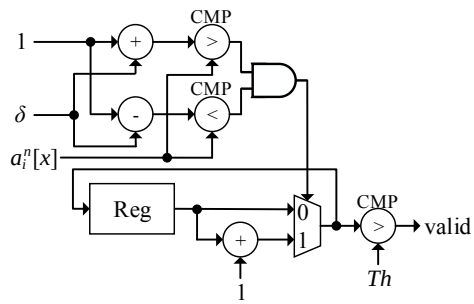

Fig. 7. Hardware architecture of the SVSEE checker.

The FM normalizer normalizes the FM signal. When the stopping criterion is satisfied, the resulting FM signal from the SVSEE checker may still be an approximation of a pure FM signal because Eq. (8) is an approximate equation. The FM normalizer ensures that the resulting FM signal is a pure FM signal, which can be written as

$$-1 \le s_i^n[x] \le 1, \tag{9}$$

Fig. 8 shows the hardware architecture of the FM normalizer.

The PF generator computes the PFs by multiplying the pure FM signal with an envelope function, $a_i[x]$. The equations for calculating a PF and the corresponding envelope function are given by

$$pf_i[x] = a_i[x]s_i^n[x], \tag{10}$$

where

$$a_i[x] = a_i^1[x]a_i^2[x]\cdots a_i^n[x] = \prod_{q=1}^{n} a_i^q[x]. \tag{11}$$

Fig. 9 shows the hardware architecture of the PF generator.

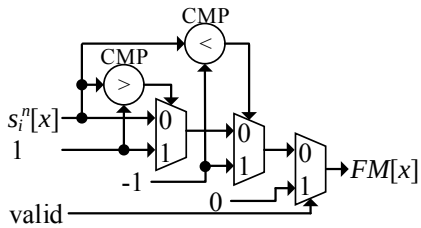

Fig. 8. Hardware architecture of the FM normalizer.

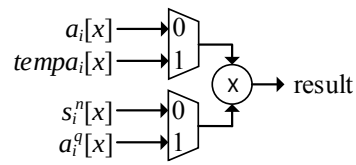

Fig. 9. Hardware architecture of the PF generator.

4. IMPLEMENTATION AND COMPARISON

All modules of the proposed LMD circuit described in this paper were developed in Verilog and then synthesized using the Synopsys Design Compiler with the TSMC 0.18-μm cell library. The design layout was then generated using the Synopsys IC Compiler. The DRC/LVS error of the layout result was verified using Mentor Calibre. Static timing analysis was conducted using Synopsys PrimeTime. The simulation results were verified using Cadence Incisive Enterprise Simulator, Synopsys VCS, and Mentor ModelSim.

In practice, different LMD applications have different timing requirements and require various data lengths, data widths, and precisions. The variables in the proposed LMD circuit are parameterized as much as possible so that the circuit can achieve the highest possible flexibility. Seven designs were chosen for precision testing, namely 8-, 12-, 16-, 20-, 24-, 32-, and 48-bit designs. Table 2 lists the clock rates and gate counts of the seven proposed designs with a data length of 200. The layout result of the 48-bit design is shown in Fig. 10.

Table 2. Clock rates and gate counts of seven proposed designs.

Design	8-bit	12-bit	16-bit	20-bit	24-bit	32-bit	48-bit
Clock Rate (MHz)	88.41	78.74	70.97	64.55	59.24	50.83	39.26
Gate Count	23758	28519	33359	38471	44142	55415	82413

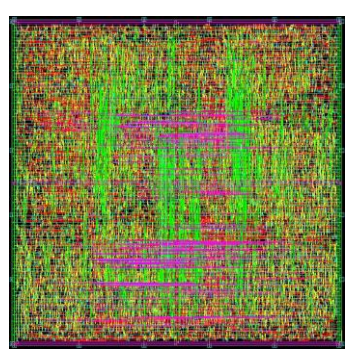

Fig. 10. Layout of the proposed 48-bit design developed using the TSMC 0.18-μm cell library.

For comparison, a software solution was implemented using C programming language in the Visual Studio 2010 integrated development environment and run on a Microsoft Windows 7 computer with Intel® Core™ i7-3770 CPU @ 3.4GHz. Table 3 shows the execution time comparison with a data length of 200. Figs. 11 and 12 show the comparison results of decomposing an electrocardiogram (ECG) signal when using 48- and 24-bit precision, respectively. The difference between the software solution and the proposed hardware solution is defined as

$$ERROR = \frac{\left| pf_i^{software}(t) - pf_i^{hardware}(t) \right|}{\left| pf_i^{software}(t) \right|}. \tag{12}$$

Fig. 13 shows the error rates of the seven designs compared with that of the software solution.

Table 3. Execution time comparison between software solution and proposed hardware solution.

Design	I7-3770 (Software)	48-bit (Hardware)	24-bit (Hardware)
Execution time (second)	2.48	0.015	0.010

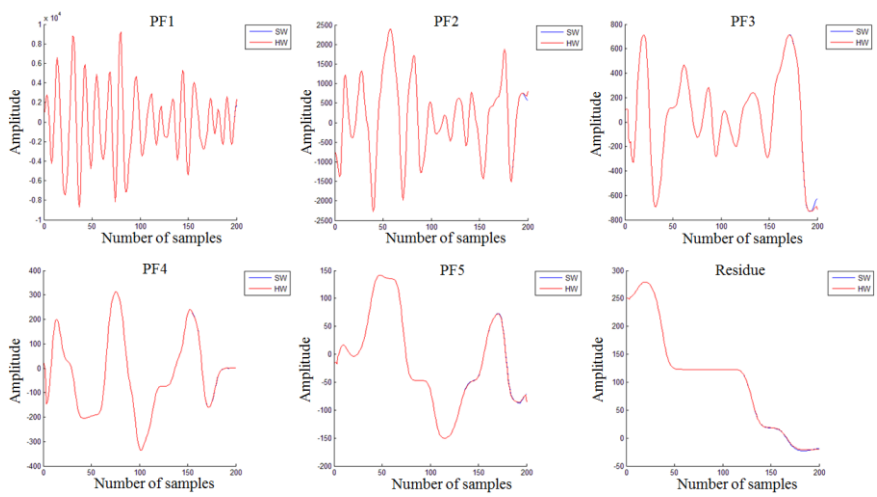

Fig. 11. Comparison of applying LMD to an ECG signal when using a software solution and the proposed hardware solution with 48-bit precision.

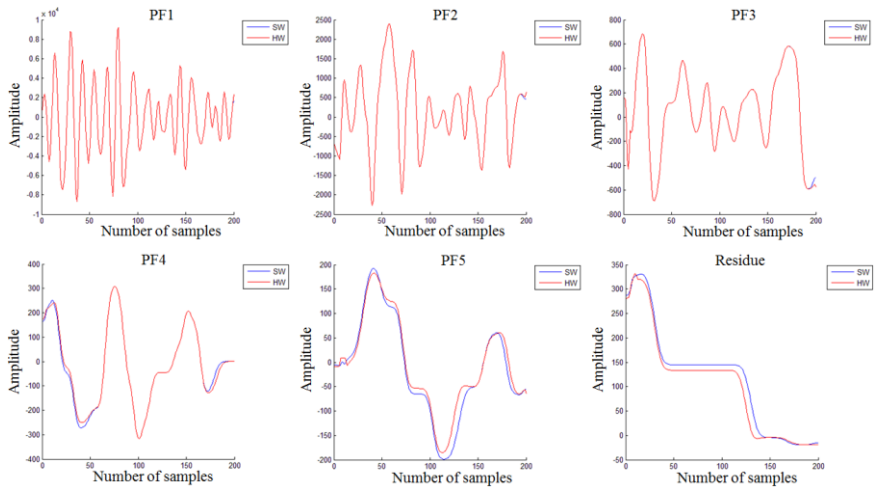

Fig. 12. Comparison of applying LMD to an ECG signal when using a software solution and the proposed hardware solution with 24-bit precision.

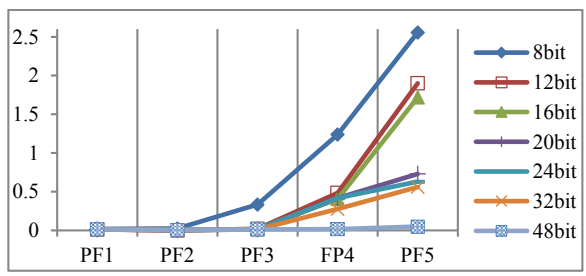

Fig. 13. Error rates of seven designs compared with that of a software solution.

5. CONCLUSIONS

In this paper, a high-speed hardware implementation of an LMD algorithm is proposed. Variables in the proposed LMD circuit are parameterized as much as possible. Because of the parameterized design, the proposed LMD circuit can be easily embedded into other hardware systems. Users can choose the most closely coincident result for various LMD applications. A comparison shows that the proposed hardware LMD implementation is 165-248 times faster than a software solution. Moreover, the decomposition results show that the proposed designs are not only fast, flexible, and low-cost but also have satisfactory precision. With these features, the proposed designs are extremely suitable for various LMD applications.

REFERENCES

1. N. E. Huang, Z. Shen, S. R. Long, M. C. Wu, H. H. Shih, Q. Zheng, N. C. Yen, C. C. Tung, and H. H. Liu, “The empirical mode decomposition and the Hilbert spectrum

for nonlinear and non-stationary time series analysis," in *Proceedings of the Royal Society of London*, *Series A: Mathematical*, *Physical and Engineering Sciences*, Vol. 454, 1998, pp. 903-995.

2. W. C. Shen, H. I. Jen, and A. Y. Wu, "New ping-pong scheduling for low-latency EMD engine design in Hilbert-Huang transform," *IEEE Transactions on Circuits and Systems-II: Express Briefs*, Vol. 60, 2013, pp. 532-536.
3. B. Biswal, M. Biswal, S. Mishra, and R. Jalaja, "Automatic classification of power quality events using balanced neural tree," *IEEE Transactions on Industrial Electronics*, Vol. 61, 2014, pp. 521-530.
4. G. Georgoulas, I. P. Tsoumas, J. A. Antonino-Daviu, V. Climente-Alarcon, C. D. Stylios, E. D. Mitronikas, and A. N. Safacas, "Automatic pattern identification based on the complex empirical mode decomposition of the startup current for the diagnosis of rotor asymmetries in asynchronous machines," *IEEE Transactions on Industrial Electronics*, Vol. 61, 2014, pp. 4937-4946.
5. S. Shukla, S. Mishra, and B. Singh, "Power quality event classification under noisy conditions using EMD-based de-noising techniques," *IEEE Transactions on Industrial Informatics*, Vol. 10, 2014, pp. 1044-1054.
6. J. S. Smith, "The local mean decomposition and its application to EEG perception data," *Journal of the Royal Society Interface*, 2005, pp. 443-454.
7. B. Yuan, Z. P. Chen, and S. Y. Xu, "Micro-doppler analysis and separation based on complex local mean decomposition for aircraft with fast-rotating parts in ISAR imaging," *IEEE Transections on Geoscience and Remote Sensing*, Vol. 52, 2014, pp. 1285-1298.
8. H. C. Hsueh and S. Y. Chien, "On-line local mean decomposition and its application to ECG signal denoising," in *Proceeding of IEEE Biomedical Circuits and Systems Conference*, 2014, pp. 17-20.
9. W. Li and H. Z. Yang, "Blind source separation in underdetermined model based on local mean decomposition and AMUSE algorithm," in *Proceeding of the Chinese Control Conference*, 2014, pp. 7206-7211.
10. Y. Guo, G. R. Naik, and N. Y. Hung, "Single channel blind source separation based local mean decomposition for biomedical applications," in *Proceeding of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2013, pp. 6812-6815.
11. J. Y. Wang and Z. X. Liu, "A self-adaptive analysis method of fault diagnosis in roller bearing based on local mean decomposition," in *Proceeding of the Chinese Control and Decision Conference*, 2014, pp. 218-222.
12. Y. X. Bu, J. D. Wu, J. Ma, X. D. Wang, and Y. G. Fan, "The rolling bearing fault diagnosis based on LMD and LS-SVM," in *Proceeding of the Chinese Control and Decision Conference*, 2014, pp. 3797-3801.
13. R. D. Chen, P. Y. Chen, and C. H. Yeh, "A low-power architecture for the design of a one-dimensional median filter," *IEEE Transactions on Circuits and Systems-II: Express Briefs*, Vol. 62, 2015, pp. 266-270.
14. S. Chen and H. Chang, "Fully pipelined low-cost and high-quality color demosaicking VLSI design for real-time video applications," *IEEE Transactions on Circuits and Systems-II: Express Briefs*, Vol. 62, 2015, pp. 588-592.

15. A. Balatsoukas-Stimming, A. J. Raymond, W. J. Gross, and A. Burg, "Hardware architecture for list successive cancellation decoding of polar codes," *IEEE Transactions on Circuits and Systems-II: Express Briefs*, Vol. 61, 2014, pp. 609-613.

Pei-Yin Chen (陳培殷) received the B.S. degree from National Cheng Kung University, Tainan, Taiwan, in 1986, the M.S. degree from Pennsylvania State University, University Park, in 1990, and the Ph.D. degree from National Cheng Kung University, in 1999, all in Electrical Engineering. He is currently a Distinguished Professor with the Department of Computer Science and Information Engineering. His research interests include very large scale integration chip design, video compression, fuzzy logic control, and gray prediction.

Yen-Chen Lai (賴彥澄) received the B.S. degree in Computer Science and Information Engineering from National Cheng Kung University, Tainan, Taiwan, in 2010. He is currently working toward the Ph.D. degree in Computer Science and Information Engineering with the Department of Computer Science and Information Engineering, National Cheng Kung University. His research interests include very large scale integration chip design, signal processing, and vehicular technology.

Ping-Hsuan Lai (賴秉暄) received the B.S. and M.S. degrees in Computer Science and Information Engineering in 2012 and 2014, respectively, from National Cheng Kung University, Tainan, Taiwan. His research interests include image processing, very large scale integration chip design, and data compression.