IntechOpen

Digital Systems

Edited by Vahid Asadpour

DIGITAL SYSTEMS

Edited by **Vahid Asadpour**

Digital Systems
http://dx.doi.org/10.5772/intechopen.74915
Edited by Vahid Asadpour

Contributors

Andres G Marrugo, Jesus Pineda, Lenny A Romero, Raul Vargas, Jaime Meneses, Shunsuke Koshita, Shiming Ge, Amer Zayegh, Nizar Albassam, Steven Voldman, Suneeta Harlapur, Alfonso Martinez Cruz, Kelsey Alejandra Ramírez Gutiérrez, Ignacio Algredo Badillo, Alejandro Medina Santiago, Kwang-Ting Cheng, René Cumplido Parra, Ricardo Barrón-Fernández, Prometeo Cortés-Antonio, Sreenatha Anavatti

Notice

First published in London, United Kingdom, 2018 by IntechOpen
eBook (PDF) Published by IntechOpen, 2019
IntechOpen is the global imprint of INTECHOPEN LIMITED, registered in England and Wales, registration number: 11086078, The Shard, 25th floor, 32 London Bridge Street
London, SE19SG – United Kingdom
Printed in Croatia

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library

Additional hard and PDF copies can be obtained from orders@intechopen.com

Digital Systems
Edited by Vahid Asadpour

p. cm.

Print ISBN 978-1-78984-540-2

Online ISBN 978-1-78984-541-9

eBook (PDF) ISBN 978-1-83881-811-1

Meet the editor

Vahid Asadpour worked at the Photonic Laboratory of the University of California Los Angeles (UCLA). He graduated from the Polytechnic University of Tehran, achieving an M.S. and PH.D. degree in biomedical engineering. He was a visiting professor and researcher at the University of North Dakota (UND). He is currently working in the field of artificial intelligence and machine learning and their applications in digital signal processing. In addition, he is using digital signal processing in medical imaging and speech processing. He developed brain computer interfacing algorithms and published one book chapter with IntechOpen and several journal and conference papers in this field and other areas of intelligent signal processing. Adding to his professional experimental skills, he also designed medical devices including a laser Doppler monitoring system and therapeutic drug estimation algorithm. He affiliates with the Sadjad University of Technology.

Contents

Preface

Digital system design requires unembellished simulation that eliminates potential risks and harm to users and manufacturers. Most of modern design and analysis tools are targeted at custom integrated circuits that are costly and time prohibitive to implement at high level designs. The purpose of this book is to provide a review on advanced digital system design and simulation through computer aided design (CAD) and machine learning tools. We present the practical applications of CAD and machine learning modeling and synthesis in digital system design to construct a basis for effective design and provide a tutorial of digital systems functionality. We review theoretical principles, discrete mathematical models, computer simulations and machine learning methods in related areas. In this book, implementation of frequency analysis methods is presented at software and hardware levels. Variable Digital Filter (VDF) is presented as one of the vastly used hardware processing tools in the field of digital systems. A detailed description is provided on the advanced design methods of VDFs. Practical application of field-programmable gate array (FPGA) in a control system is presented in this book and an evolutionary algorithm is presented for functional verification of FPGA design. Deep learning has been used as an efficient tool for compression of digital networks and increasing the processing speed. Some of the useful deep learning methods have been introduced and the applications of them is presented in digital system design and verification. Several architectures are introduced and evaluated including general regression neural networks and convolutional neural networks.

As the editor of the book, I would like to acknowledge the contribution of the authors. These efforts allowed the updated materials in the field of digital systems to be available for the researchers and users in this field.

Dr. Vahid Asadpour
Sadjad University of Technology
Research Scientist at University of California Los Angeles (UCLA)

Section 1

Digital Systems and Firmware Algorithms

Chapter 1

Fourier Transform Profilometry in LabVIEW

Andrés G. Marrugo, Jesús Pineda, Lenny A. Romero,
Raúl Vargas and Jaime Meneses

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.78548

Abstract

Fourier transform profilometry (FTP) is an established non-contact method for 3D sensing in many scientific and industrial applications, such as quality control and biomedical imaging. This phase-based technique has the advantages of high resolution and noise robustness compared to intensity-based approaches. In FTP, a sinusoidal grating is projected onto the surface of an object, the shape information is encoded into a deformed fringe pattern recorded by a camera. The object shape is decoded by calculating the Fourier transform, filtering in the spatial frequency domain, and calculating the inverse Fourier transform; afterward, a conversion of the measured phase to object height is carried out. FTP has been extensively studied and extended for achieving better slope measurement, better separation of height information from noise, and robustness to discontinuities in the fringe pattern. Most of the literature on FTP disregards the software implementation aspects. In this chapter, we return to the basics of FTP and explain in detail the software implementation in LabVIEW, one of the most used data acquisition platforms in engineering. We show results on three applications for FTP in 3D metrology.

Keywords: 3D reconstruction, Fourier transform profilometry, FTP, LabVIEW

1. Introduction

Three-dimensional (3D) shape measurement techniques are widely used in many different fields such as mechanical engineering, industry monitoring, robotics, biomedicine, dressmaking, among others [1]. These techniques can be classified as passive, like in stereo vision in which two or more cameras are used to obtain the 3D reconstruction of a scene, or as active, like in fringe projection profilometry (FPP) in which a projection device is used to project a pattern onto the object to be reconstructed. When compared with other 3D measurement techniques, FPP has the advantages of high measurement accuracy and high density. There are two types of FPP

methods: phase shifting and Fourier-transform profilometry (FTP). Phase-shifting methods offer high-resolution measurement at the expense of projecting several patterns onto the object [2–4], whereas FTP is popular because only one deformed fringe pattern image is needed [5]. For this reason, FTP has been used in many dynamic applications [6] such as vibration measurement of micromechanical devices [7] and measurement of real-time deformation fields [8].

FTP was proposed by Takeda et al. [5, 9] in 1982 and has since become one of the most used methods [3, 10]. Its main advantages are full-field analysis, high precision, noise-robustness [11], among others. In FTP, a Ronchi grating, or a sinusoidal grating, or a fringe pattern from a digital projector is projected onto an object, and the depth information of the object is encoded into the deformed fringe pattern recorded by an image acquisition device as shown in **Figure 1**. The surface shape can be decoded by calculating the Fourier transform, filtering in the spatial frequency domain, and calculating the inverse Fourier transform. Compared with other fringe analysis methods, FTP can accomplish a fully automatic distinction between a depression and an elevation of the object shape. It requires no fringe order assignments or fringe center determination, and it needs no interpolation between fringes because it gives height distribution at each pixel over the entire field. Since FTP requires only one or two images of the deformed fringe pattern, it has become one of the most popular methods for real-time 3D reconstruction of dynamic scenes.

Although FTP has been extensively studied and used in many applications, to the best of our knowledge a complete reference in which the implementation details are fully described is nonexistent. In this chapter, we describe the FTP fundamentals and the implementation of an FTP system in LabVIEW one of the most used engineering development platforms for data acquisition and laboratory automation. The chapter is organized as follows. In Section 2 we describe the FTP fundamentals and a general calibration method, in Section 3 we describe how FTP is implemented in LabVIEW, and finally in Section 4 we show three applications of FTP for 3D reconstruction.

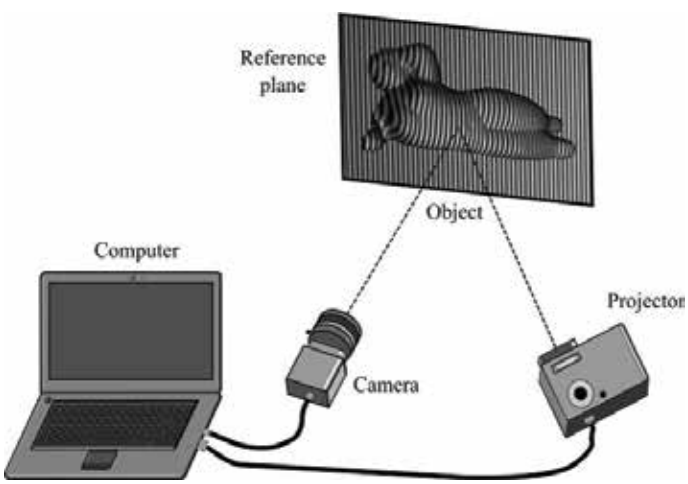

Figure 1. Fringe projection system.

2. FTP fundamentals

There are many implementations of FPP. However, all share the same underlying principle. A typical FPP setup consists of a projection device and a camera as shown in **Figure 1**. A fringe pattern is projected onto a test object, and the resulting image is acquired by the camera from a different direction. The acquired fringe pattern image is distorted according to the object shape. In terms of information theory, it is said that the object shape is encoded into a deformed fringe pattern acquired by the camera. The object shape is recovered/decoded by comparison to the original (undeformed) fringe pattern image. Therefore, the phase shift between the reference and the deformed image contains the information of the object shape.

By projecting a fringe pattern onto the reference plane, the fringe pattern (with period $p_0 = 1/f_0$) on the reference plane observed through the camera can be modeled as

$$g_0(x,y) = a_0(x,y) + b_0(x,y)\cos\left[2\pi f_0 x + \phi_0(x,y)\right]. \tag{1}$$

Likewise, when the object is placed on the reference plane, the deformed fringe pattern observed through the camera is given by

$$g(x,y) = a(x,y) + b(x,y)\cos\left[2\pi f_0 x + \phi(x,y)\right], \tag{2}$$

where $a_0(x,y)$ and $a(x,y)$ represent the non-uniform background illumination, $b_0(x,y)$ and $b(x,y)$ the contrast of the fringe pattern. f_0 is the fundamental frequency of the observed fringe pattern (also called carrier frequency). $\phi_0(x,y)$ and $\phi(x,y)$ are the original phase modulation on the reference plane R where $z(x,y) = 0$ and the phase modulations resulting from the object height distribution, respectively. $a(x,y)$, $b(x,y)$ and $\phi(x,y)$ are assumed to vary much slower than the spatial carrier frequency f_0. The principle of FTP is shown schematically in **Figure 2**. The input fringe pattern from Eqs. (1) and (2) can be rewritten using Euler's formula in the following form

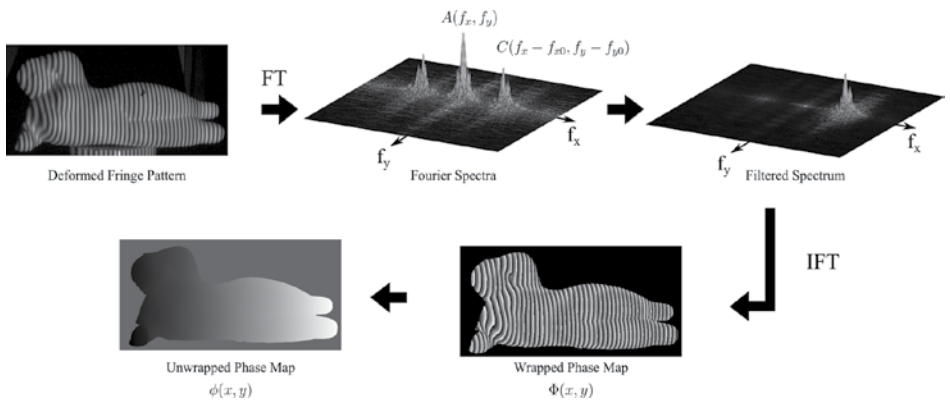

Figure 2. Principle of the filtering via Fourier transform (FT) method. IFT, inverse FT.

$$g(x,y) = a(x,y) + c(x,y)\exp\{2\pi i f_0 x\} + c^*(x,y)\exp\{-2\pi i f_0 x\}, \tag{3}$$

with

$$c(x,y) = \frac{1}{2}b(x,y)\exp\{i\phi(x,y)\}, \tag{4}$$

where $*$ denotes a complex conjugate.

Next, the phase of the fringe patterns is recovered using the Fourier Transform method. Using one-dimensional notation for simplicity, when we compute the Fourier transform of Eqs. (1) and (2) the Fourier spectrum of the fringe signals splits intro three spectrum components separated from each other, which gives

$$G(f_x, y) = A(f_x, y) + C(f_x - f_0, y) + C^*(f_x + f_0, y), \tag{5}$$

as shown in two dimensions in **Figure 2**. With an appropriate filter function, for instance, a Hanning filter, the spectra are filtered to let only the fundamental component $C(f_x - f_0, y)$. A Hanning window is given by [11],

$$H(f_x) = 0.50\left[1 + \cos\left(\beta\pi\frac{f_x - f_0}{f_c}\right)\right], \tag{6}$$

where f_c is the cutoff frequency at a 50% attenuation ratio, $\beta = 1/2$ and f_x varies from $f_0 - f_c/\beta$ to $f_0 + f_c/\beta$. The inverse Fourier Transform is applied to the filtered component, and a complex signal is obtained

$$\widehat{g}_0(x,y) = \frac{1}{2}b(x,y)\exp\{i(2\pi f_0 x + \phi_0(x,y))\}, \tag{7}$$

$$\widehat{g}(x,y) = \frac{1}{2}b(x,y)\exp\{i(2\pi f_0 x + \phi(x,y))\}. \tag{8}$$

The variable related to height distribution is the phase change $\Delta\phi(x,y)$ [9]:

$$\Delta\phi(x,y) = \Phi(x,y) - \Phi_0(x,y) = \phi(x,y) - \phi_0(x,y), \tag{9}$$

with

$$\Phi_0(x,y) = \tan^{-1}\left(\frac{\mathfrak{I}[\widehat{g}_0(x,y)]}{\mathfrak{R}[\widehat{g}_0(x,y)]}\right), \tag{10}$$

$$\Phi(x,y) = \tan^{-1}\left(\frac{\mathfrak{I}[\widehat{g}(x,y)]}{\mathfrak{R}[\widehat{g}(x,y)]}\right), \tag{11}$$

where $\mathfrak{I}[.]$ and $\mathfrak{R}[.]$ denote the imaginary and the real part, respectively. The phases obtained from Eqs. (10) and (11) are wrapped into the principal value $[-\pi, \pi]$. The wrapped phase is

unwrapped by using a suitable phase unwrapping algorithm [12] that gives the desired phase map as shown in **Figure 2**. The phase map $\Delta\phi(x, y)$ is proportional to the height of the object surface.

2.1. System calibration

The calibration of FPP systems plays an essential role in the accuracy of the 3D reconstructions. Here we describe a simple yet extensively used calibration called the reference-plane-based technique, i.e., to convert the unwrapped phase map $\Delta\phi(x, y)$ to height z.

The optical axis geometry of the FTP measurement system is depicted in **Figure 3**. The optical axis $E_p^{'} - E_p$ of a projector lens crosses the optical axis $E_c^{'} - E_c$ of a camera lens at a point O on a reference plane R. This reference plane is normal to the optical axis $E_c^{'} - E_c$ and serves as a reference to measure the height of the object $z(x, y)$. d is the distance between the projector and the camera, l_0 is the distance between the camera and the reference plane. The fringe pattern image (with period p) is formed by the projector lens on plane I through point O. p is related to the carrier frequency by $f_0 = 1/p_0 = \cos\theta/p$. The height of the object surface is measured relative to R. From the point of view of the projector, point A on the object surface has the same phase value as point C on the reference plane R, $\Phi_A = \Phi_C^R$, where the superindex R denotes a point on the reference plane. On the camera sensor, point A on the object surface and point D on the reference plane are imaged on the same pixel. By subtracting the reference phase map from the object phase map, we obtain the phase difference at this specific pixel

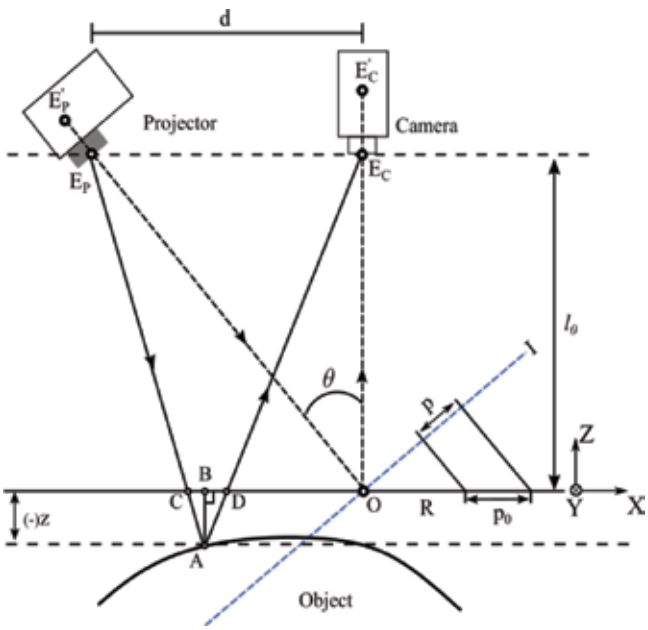

Figure 3. Fringe projection system.

$$\Delta\Phi_{AD} = \Phi_A - \Phi_D^R = \Phi_C^R - \Phi_D^R = \Phi_{CD}^R. \tag{12}$$

The triangles $\Delta E_p E_c A$ and ΔCDA are similar, and the height $\overline{AB}$ of point A on the object surface relative to the reference plane can be related to the distance between points C and D

$$\Delta z(x,y) = \overline{AB} \approx \frac{l_0}{d}\overline{CD} \propto \Delta\Phi_{CD}^R = \Phi_A - \Phi_D^R. \tag{13}$$

Combining Eqs. (12) and (13) a proportional relation between the phase map and the surface height can be obtained for any point (x, y)

$$\Delta z(x,y) \propto \Delta\phi(x,y) = \Phi(x,y) - \Phi_0(x,y), \tag{14}$$

where $\Phi(x, y)$ is the object phase map and $\Phi_0(x, y)$ is the reference plane phase map. Assuming the reference plane has a depth of z_0, the depth value for each measured point can be represented as

$$z(x,y) = z_0 + k_0 \times [\Phi(x,y) - \Phi_0], \tag{15}$$

where k_0 is a constant determined through calibration and z_0 is usually set to 0.

We have shown how the object surface height is related to the recovered phase through FTP. The model described by Eq. (15) has many underlying assumptions and is often extended to cover more degrees of freedom. Moreover, a general calibration process in FPP can be carried out employing the methodology shown in **Figure 4**. First, we propose a model that best describes the system, while also considering metrological requirements such as speed, robustness, accuracy, flexibility and reconstruction scale. Some authors have proposed to use several calibration models based on polynomial or fractional fitting functions [13, 14], bilinear interpolation by look-up table (LUT) [15] and stereo triangulation [16–18]. These calibration models require different strategies or techniques that allow relating metric coordinates with phase values. In step II, we select or design a strategy that fits the proposed calibration model and characteristics of the elements to a given experimental setup, such as the type of projector (i.e., analog or digital projection) and camera (i.e., monochrome or color). These strategies consist in projecting and capturing fringe patterns onto 3D-objects [19] or 2D-targets [16, 20] with highly accurate known measurements. In some cases, the calibration consists in displacing the targets along the z axis using a linear translation stage [19]. The purpose is to obtain a correspondence between a metric coordinate system and the phase images captured with the camera. In step III, the correspondences are used to calculate the parameters that are part of the proposed

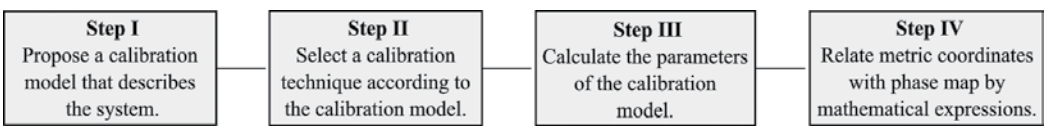

Figure 4. General calibration methodology.

model, and the best data obtained in step II. Finally in step IV, with the complete model, we can find mathematical expressions that convert phase maps to XYZ-coordinates.

3. LabVIEW implementation

In this section, we explain the details of the FTP software implementation in LabVIEW. LabVIEW stands for Laboratory Virtual Instrument Engineering Workbench and is a system-design platform and development environment for a visual programming language from National Instruments [21]. It allows integrating hardware, acquiring and analyzing data, and sharing results. Because it is a visual programming language based on function blocks, it is a highly intuitive integrated development environment (IDE) for engineers and scientists familiar with block diagrams and flowcharts. Every LabVIEW block diagram also has an associated front panel, which is the user interface of the application.

The acquisition and processing strategies described in this section require the installation of the following software components:

- **NI vision acquisition software**, which installs **NI-IMAQdx**. This software driver allows the integration of cameras with different control protocols such as USB3 Vision, GigE Vision devices, IEEE 1394 cameras compatible with IIDC, IP (Ethernet) and DirectShow compatible USB devices (e.g., cameras, webcams, microscopes, scanners). **NI vision acquisition software** also includes the driver **NI-IMAQ** for acquiring from analog cameras, digital parallel and Camera Link, as well as NI Smart Cameras. This hardware compatibility is the main advantage of using LabVIEW for vision systems. This compatibility greatly facilitates the development of applications for different types of cameras and busses.
- **NI vision development module (VDM)**. This package provides machine vision and image processing functions. It includes IMAQ Vision, a library of powerful functions for vision processing. In this library, there is a group of VIs that analyze and process images in the frequency domain. We will make use of these functions throughout the entire chapter.

NI VDM and Vision Acquisition Software are supported on the following operating systems:

• Windows 10; Windows 8.1; Windows 7 (SP1) 32-bit; Windows 7 (SP1) 64-bit; Windows Embedded Standard 7 (SP1); Windows Server 2012 R2 64-bit; Windows Server 2008 R2 (SP1) 64-bit.

3.1. Image acquisition

There are two primary ways to obtain images in LabVIEW: loading an image file or acquiring directly from a camera. The wiring diagram in **Figure 5(a)** illustrates how to perform a continuous (grab) acquisition in LabVIEW using Vision Acquisition Software. A Grab acquisition begins by initializing the camera specified by the Camera Name Control and configuring the driver for acquiring images continuously. Using IMAQ Create, we create a temporary

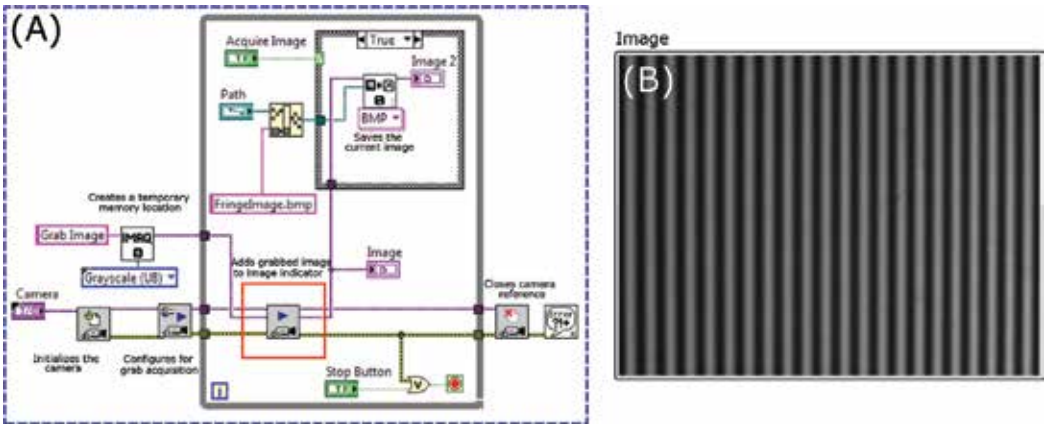

Figure 5. Grab acquisition in LabVIEW. (a) Block diagram. (b) Image indicator in front panel.

memory location for the acquired image. This function returns an IMAQ image reference to the buffer in memory where the image is stored. The reference is the input to the IMAQ Grab VI for starting the acquisition. The grabbed image is displayed on the LabVIEW front panel using an Image Indicator (see **Figure 5(b)**), which points to the location in memory referenced by the IMAQ image reference. A while loop statement allows adding each grabbed image to the image indicator as a single frame. Finally, the image acquisition is finished by calling the IMAQ close VI that releases resources associated with the camera and the interface.

The acquired image is written to a file in a specified format by using the IMAQ Write File 2 VI. The graphics file formats supported by this function are BMP (windows bitmap), JPEG, PNG (portable network graphics), and TIFF (tagged image file format). However, note that lossy compression formats, such as JPEG, introduce image artifacts and should be avoided to ensure accurate image-based measurements. The saved image can be displayed in a secondary image indicator by enabling the Snapshot option. When enabling the Snapshot Mode, the Image Display control continues to display the image as it was when the image was saved during the Case Structure execution, even when the inspection image has changed. To configure the Image Display control for working in Snapshot Mode, right-click on the control on the front panel and enable the Snapshot option.

Another way to acquire an image using a camera is presented in the **Figure 6**. This example uses the NI Vision Acquisition Express to perform the acquisition stage. The Vision Acquisition Express VI is located in the Vision Express palette in LabVIEW, and it is commonly used to quickly develop image acquisition applications due to its versatility and intuitive development environment. Double-clicking on the Vision Acquisition Express VI makes a configuration window appear which allows choosing a device from the list of available acquisition sources, selecting an acquisition type, and configuring the acquisition settings. Concerning the acquisition types, there are four main modes: single acquisition with processing, continuous acquisition with inline processing, finite acquisition with inline processing and finite acquisition with post-processing. The last two acquisition types are similar, except that for a finite acquisition with post-processing the images are only available after they are all acquired. The configuration of the

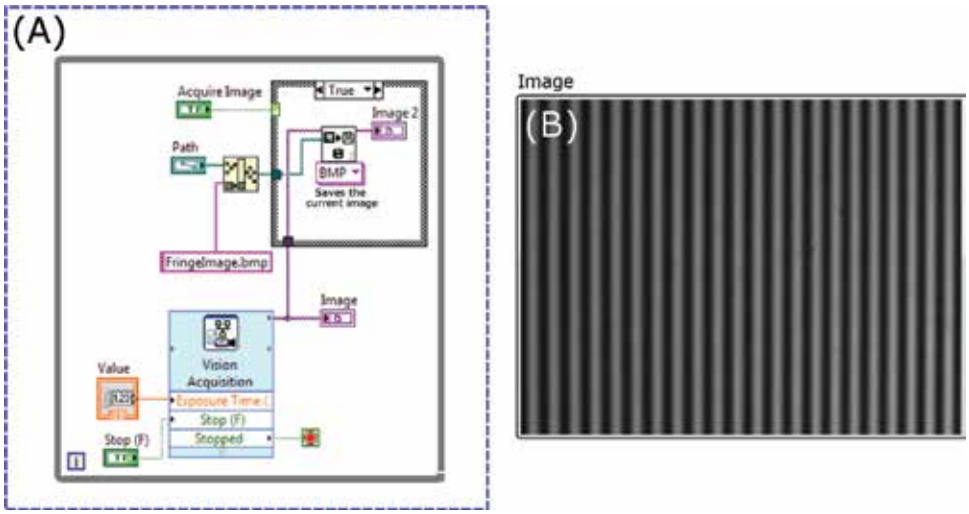

Figure 6. Continuous acquisition using IMAQ vision acquisition express. (a) Block diagram. (b) Image indicator in front panel.

acquisition settings is one of the most relevant processes during configuration and allows the simultaneous manipulation of camera attributes like Exposure Time, Trigger Mode, Gain, Gamma Factor, among others. For this example, we configured the acquisition for working in a continuous acquisition with inline processing mode, which continuously acquires images until an event stops the acquisition. Additionally, the Exposure Time attribute can be modified during the acquisition process by using a Numeric Control. As with the example in **Figure 5**, the captured image is displayed in a secondary image indicator during the Case Structure execution.

In Fringe Projection systems, the manipulation of certain camera attributes (e.g., the Exposure Time attribute) is required to capture high-quality images and to enable to work under different lighting environments with different constraints. In the example above, we introduced the possibility of manipulating camera attributes during acquisition using the Vision Acquisition Express. This manipulation of attributes is also possible by programming a simple snap, grab, or sequence operation based on low-level VIs (as in the example in **Figure 5**) using IMAQdx property nodes. The attribute manipulation requires providing the property node with the name of the attribute we want to modify and identifying the attribute representation, which can be an integer, float, Boolean, enumeration, string or command. In general, cameras share many attributes; however, they often have specific attributes depending on the manufacturer. These attributes should be known beforehand to ensure good acquisition control. At the development stage, LabVIEW does not know or display the name of the attributes or representations. Furthermore, if the documentation is not available, we suggest using the Measurement and Automation Explorer (MAX). MAX is a tool that allows the configuration of different acquisition parameters and is useful when it is required to manipulate attributes of a device with a specific interface within the LabVIEW programming environment. For example, suppose we want to modify the exposure time of our camera (Basler Aca 1600-60gm), but we do not have information about supported attributes. Here is where MAX becomes a

powerful tool for vision system developers. This attribute verification is done by selecting the desired attribute from the Camera Attributes tab in the Measurement and Automation Explorer and identifying its name (i.e., ExposureTimeAbs) and representation (i.e., floating-point format). Therefore, the section of the block diagram inside a red box in **Figure 5** can be modified in order to allow setting the ExposureTimeAbs attribute value using a Property Node as shown in **Figure 7**.

Both acquisition methods have their advantages and disadvantages concerning their implementation in vision systems. On the one hand, the use of the NI Vision Acquisition Express allows to quickly and easily develop acquisition applications, even without having a high knowledge of the tools for image acquisition offered by LabVIEW. However, this could be a disadvantage if our purpose is to have complete control over the acquisition. On the other hand, the low-level VIs provide greater control and versatility over the application development, but the implementation of vision systems based on low-level VIs can be a complicated task for novice users of NI Vision Acquisition Software and LabVIEW.

Once the acquired fringe image file has been written to disk, it is loaded for processing. The block diagram in **Figure 8** illustrates how to perform this procedure in LabVIEW. The IMAQ ReadFile VI opens and reads an image from a file stored on the computer into an image reference. The loaded pixels are converted automatically into the image type supplied by IMAQ Create VI. From now on we refer to the Fringe Image to the loaded fringe image.

3.2. Fringe pattern projection

In the previous section, we described several acquisition methods for capturing images from a camera in LabVIEW. However, in fringe projection systems there are many different fringe pattern projection technologies and choosing the correct one becomes extremely important for an accurate three-dimensional reconstruction. A fringe pattern projector can be considered as an analog device (e.g., LED pattern projector) or as a digital device (e.g., DLP, LCoS, and LCD digital display technologies). LED pattern projectors are ideal for high-resolution three-dimensional reconstruction applications. If equipped with an objective lens and a stripe pattern reticle, these projectors offer great versatility for manipulating the optics of the system and obtaining results according to the metrological requirements. The main disadvantage of this type of projection system is the impossibility of manipulating the projected fringe pattern. Therefore, its use is often restricted to techniques in which only a single fringe image is necessary to obtain the 3D information, such as in the case of FTP.

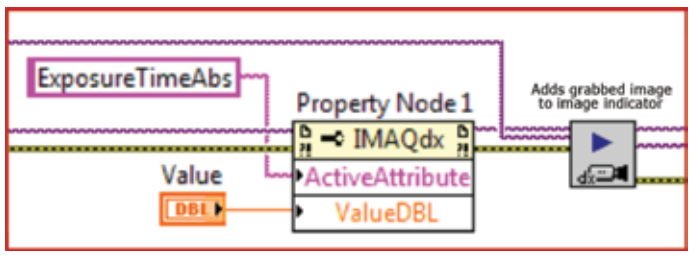

Figure 7. Setting the ExposureTimeAbs attribute value using a property node.

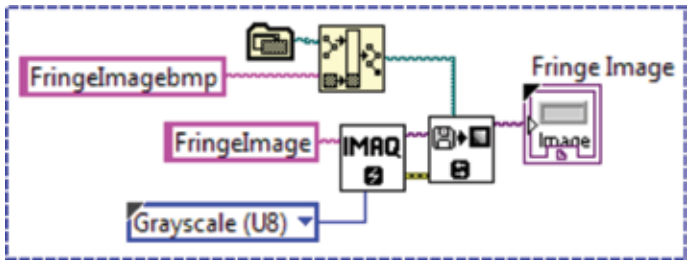

Figure 8. Reading an image file in LabVIEW.

Fringe Projection systems can also take advantage of a computer to generate sinusoidal fringe patterns that are projected using a digital projector. The key to a successful 3D reconstruction system based on digital fringe projection focuses on generating high-quality fringes to meet the metrological requirements. Ideally, assuming the projector is linear in that it projects grayscale values ranging from 0 to 255 (0 black, and 255 white), the computer-generated fringe patterns can be described as follows,

$$I(i,j) = \frac{255}{2}\left[1 + cos\left(\frac{2\pi i}{p_d} + \varphi\right)\right], \tag{16}$$

where p_d represents the number of pixels per fringe period, φ refers to the phase shift, and (i,j) are the pixel indices. Eq. (16) is implemented using the numeric functions provided by the NI LabVIEW Base Package. An example of a pattern generator block diagram is shown in **Figure 9**. In this program the Numeric Indicators enable the modification of the fringe pitch and the phase shift according to the application requirements.

An alternative to a block diagram implementation of Eq. (16) LabVIEW provides a MathScript RT Module as a scripting language. The module allows the combination of textual and graphical approaches for algorithm development. In **Figure 10** we provide an example on how to use the MathScript RT Module for fringe generation in LabVIEW.

Once the fringe images have been generated, they are sent to a digital video projector for projection. A video projector is essentially a second monitor. Therefore the fringe image is displayed by

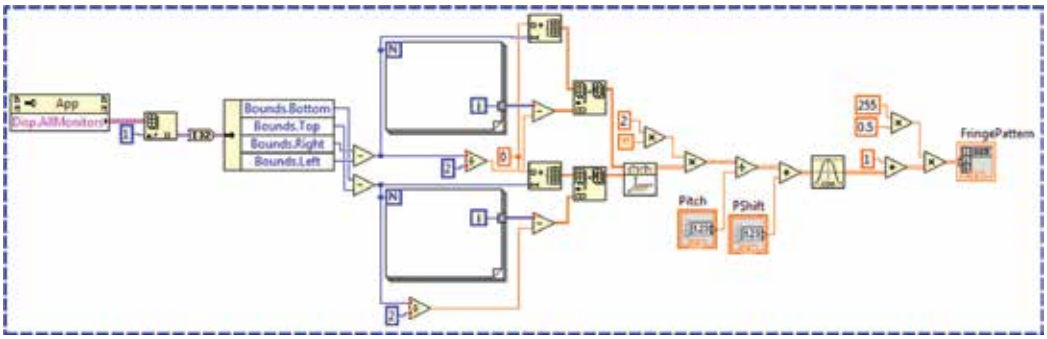

Figure 9. Block diagram for fringe pattern generation.

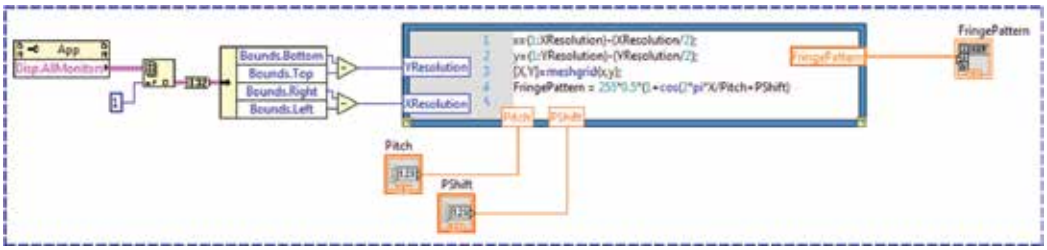

Figure 10. Fringe pattern generation example using the LabVIEW MathScript RT module.

using the External Display VIs provided by the NI Vision Development Module. Here, we use IMAQ WindDraw VI to display the image in an external image window. The image window appears automatically when the VI is executed. Having beforehand the information from all the available displays on the computer, including their resolution and bounding rectangles, we set the position of the image window to be displayed on the desired monitor. This setting is done with IMAQ WindMove VI. Additionally, using IMAQ WindSetup VI the appearance and attributes of the window can be modified to hide the title bar. Note that the default value for this attribute is TRUE which shows the title bar. The block diagram in **Figure 11** illustrates a projection stage in LabVIEW. Here, we use a Property Node for obtaining the information about all the monitors on the computer. The Disp.AllMonitors property Returns information about their bounding rectangles and bit depths.

3.3. Phase retrieval

Phase retrieval is carried out by Fourier transform profilometry. In LabVIEW, the IMAQ FFT VI computes the discrete Fourier transform of the fringe image. This function creates a complex image in which low frequencies are located at the edges, and high frequencies are grouped at the center of the image. Note that for the IMAQ FFT VI a reference to the destination image must be specified and configured as a Complex(CSG) image. Once the deformed fringe pattern is 2-D Fourier transformed, the resulting spectra are converted into a complex

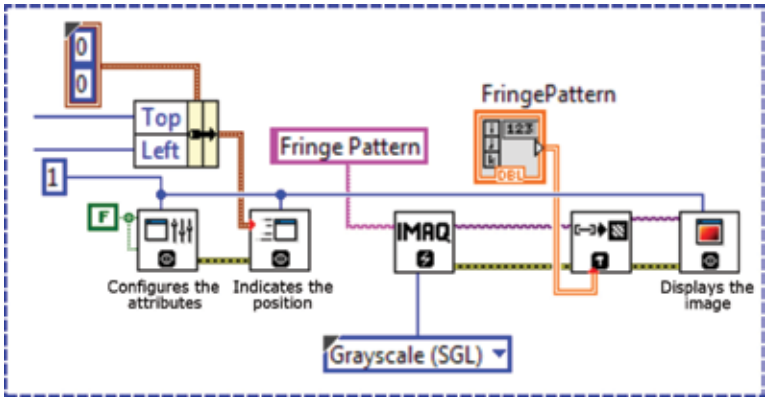

Figure 11. Second monitor configuration in LabVIEW.

2D array to perform the filtering procedure, thus obtaining the fundamental frequency spectrum in the frequency domain. The following step is to compute the inverse Fourier transform of the fundamental component. The Inverse FFT VI is for computing the inverse discrete Fourier transform (IDFT) of a complex 2D array. By using this function, we calculate the inverse FFT of the fundamental component which contains the 3D information. Finally, we obtain the phase by applying Eq. (11). Here, we use Complex To Re/Im Function to break the complex 2D array into its rectangular components and Inverse Tangent(2 Input) Function for performing the arctangent operation. With the example in **Figure 12(a)** we illustrate the phase retrieval process in LabVIEW. In this figure, the Fringe Image and Hanning W refer to the fringe pattern image shown in **Figure 12(b)** and the Hanning window filter array, respectively. The resultant wrapped phase map is shown in **Figure 12(c)**.

3.4. Hanning filter design

In Section 2 we showed that in FTP a filtering procedure is performed to obtain the fundamental frequency spectrum in the frequency domain. Once the Fourier transform is computed, the resultant spectrum is filtered by a 2-D Hanning window defined by Eq. (6). In LabVIEW, the IMAQ Select Rectangle VI is commonly used to specify a rectangular region of interest (ROI) in an image. We use the IMAQ Select Rectangle VI for manually selecting the region in the Fourier spectrum corresponding to the fundamental frequency component. Here, the image is displayed in an external display window and through the use of the rectangle tools, provided by the IMAQ Select Rectangle VI, we estimate the optimal size and location of the filtering

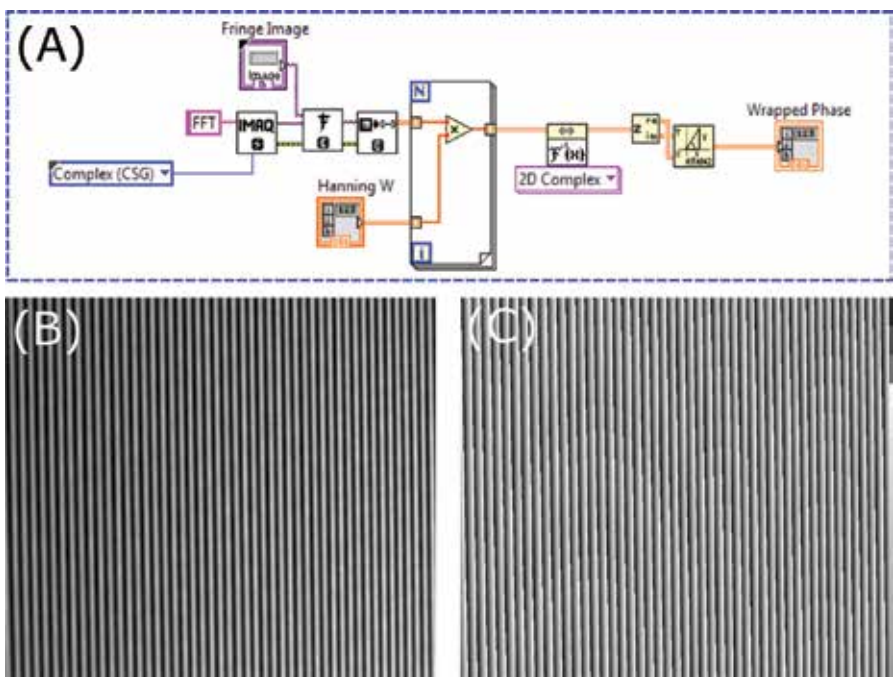

Figure 12. Phase retrieval process in LabVIEW. (a) Block diagram. (b) Fringe pattern image. (c) Wrapped phase map.

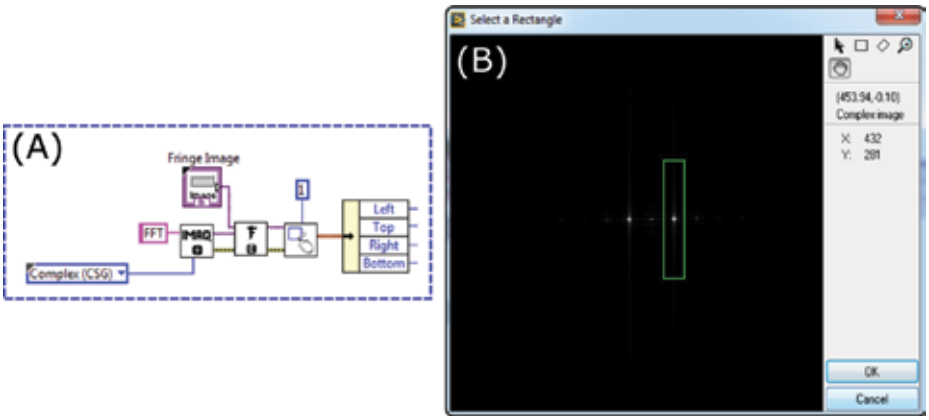

Figure 13. Manual selection of the filtering window. (a) Block diagram. (b) External display window and rectangle tools.

window that guarantees the separation between the fundamental frequency component and other unwanted contributions. The block diagram shown in **Figure 13(a)** indicates the IMAQ Select Rectangle VI to manually select the region corresponding to the first order spectrum. The Fringe Image is the fringe pattern image in **Figure 12(b)**. The IMAQ FFT VI computes the

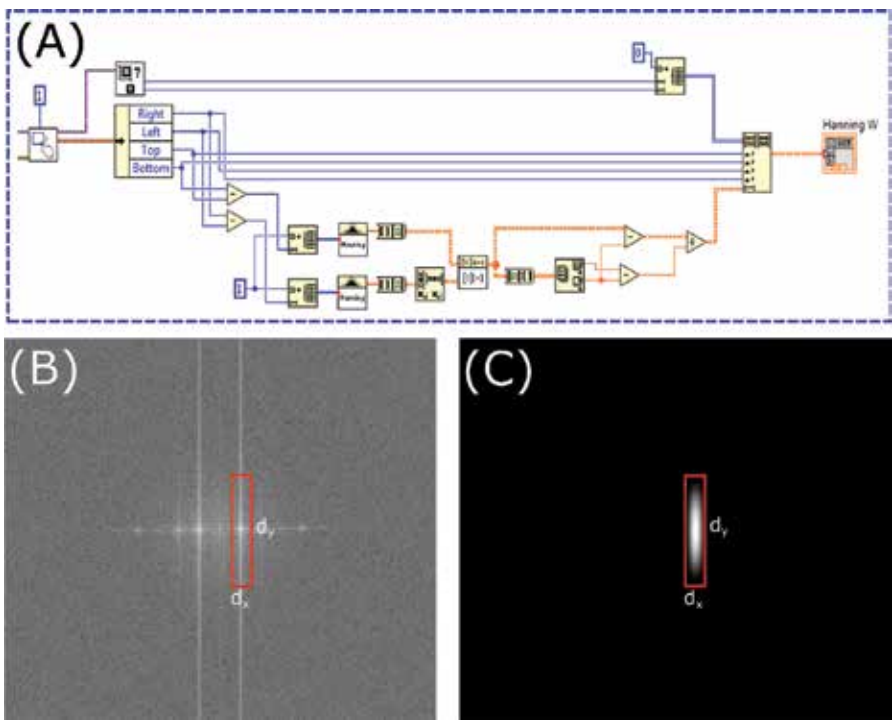

Figure 14. Hanning filter design in LabVIEW. (a) Continuation of the block diagram in **Figure 13(a)**. (b) Fourier transform magnitude spectra displayed by the external window in **Figure 13(b)**. d_x and d_y relate to the size in x and y of the filtering window, respectively. (c) 2D-hanning window.

discrete Fourier transform of the Fringe Image. The resultant complex spectrum is displayed using an external display window as shown in **Figure 13(b)**. By using the selection tools located on the right side of the window, we can manually select the rectangular area of interest.

The IMAQ Select Rectangle VI returns the coordinates (i.e., left, top, right and button) of the chosen rectangle as a cluster. Therefore, it is necessary to access each element from the cluster to extract the window information. For this reason, we add the Unbundle By Name function to the block diagram which unbundles a cluster element by name. Based on this information, we calculate the size and location of the Hanning window filter. Finally, using the Hanning Window VI two 1-D Hanning windows are created whose lengths correspond to the size of x and y of the filtering window, respectively. The two-dimensional Hanning window is obtained by the separable product of these two 1-D Hanning windows [22]. The block diagram in **Figure 14(a)** illustrates the filtering design stage in LabVIEW. d_x and d_y, in **Figure 14(b)**, relate to the size in x and y of the selected filtering window, respectively. Finally, the obtained 2D Hanning window is shown in **Figure 14(c)**.

3.5. Phase unwrapping

The phase unwrapping process is carried out comparing the wrapped phase at neighborhoods and adding, or subtracting, an integer number of 2π, thus obtaining a continuous phase. This definition is for the one-dimensional phase unwrapping process. However, for two-dimensional (2-D) phase unwrapping this is not readily applicable, and additional steps must be taken to

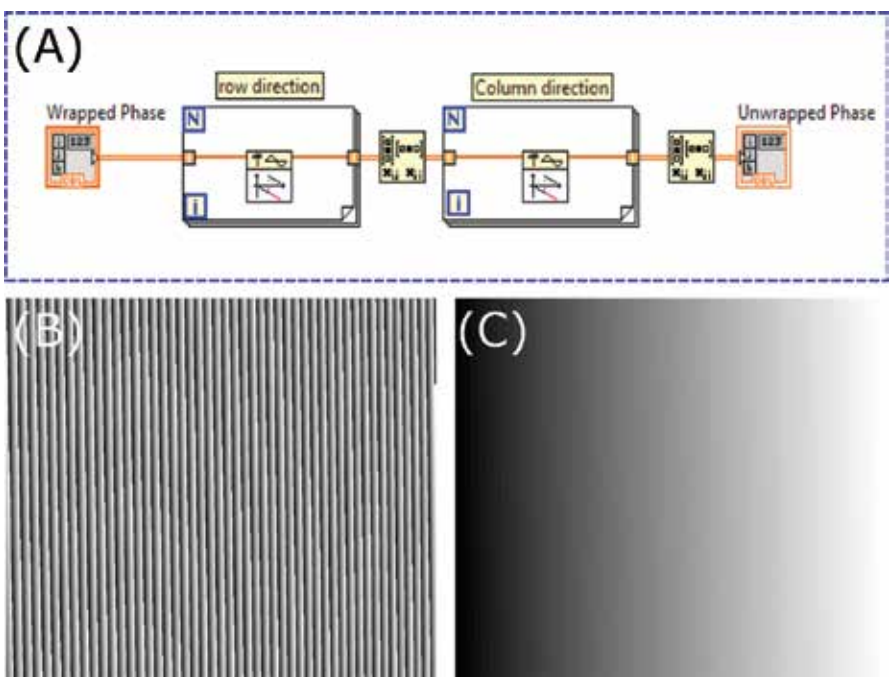

Figure 15. Bidimensional phase unwrapping in LabVIEW. (a) Wrapped phase map. (b) Unwrapped phase map.

obtain the unwrapped solution. The conventional approach for 2-D phase unwrapping can be accomplished by applying 1-D phase unwrapping first row-wise followed by 1-D phase unwrapping column-wise in two steps. The block diagram in **Figure 15(a)** illustrates this process. Here, the Unwrap Phase VI unwraps a 1D-phase array by eliminating discontinuities whose absolute values exceed π. Thus, a for loop is required to compute the continuous phase for each row of the 2-D wrapped phase array. For 1-D phase unwrapping column-wise, we use the Transpose Matrix Function to calculate the conjugate transpose of the resultant array before executing the for loop statement. **Figure 15(b)** and **(c)** show a wrapped phase map and its unwrapped counterpart, respectively. In addition to this approach, many 2D phase-unwrapping algorithms have been proposed, especially to address discontinuities and noise [12]. These other methods can also be implemented in LabVIEW either with block diagrams, using math scripts, with precompiled C++ code in .dll files, or via integration of external functions with other environments such as MATLAB. However, an explanation of the details of these other approaches is beyond the scope of this chapter.

4. Applications

FPP is often used as a non-contact surface analysis technique in industry inspection. In this section, we show the 3D surface reconstruction of a dented steel pipe. A dent is a permanent plastic deformation of the cross-section of the pipe. In the example shown in **Figure 16** the dent was produced penetrating the pipe with a diamond cone indenter. In **Figure 16(a)** and **(b)** we show the tested object, and the deformed fringe pattern image, respectively. The goal is to measure the depth of the dent with high accuracy and to obtain the surface shape of the pipe for subsequent deformation analysis. In **Figure 16(c)** and **(d)**, we show the wrapped, and unwrapped phases obtained by FTP, respectively. The unwrapped phase map is converted to metric coordinates using a calibration model. In **Figure 17(a)**, we show the reconstructed pipe shape with the texture map. A profile across the reconstructed pipe, thought the dent, is shown in **Figure 17(b)**. Analyzing this profile, we can measure the depth of the dent to approximately 4 mm.

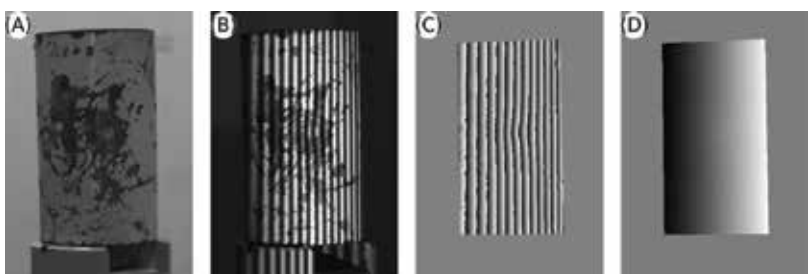

Figure 16. FTP analysis of a indented pipe. (a) Texture image. (b) Deformed fringe pattern. (c) Wrapped phase. (d) Unwrapped phase.

http://dx.doi.org/10.5772/intechopen.78548

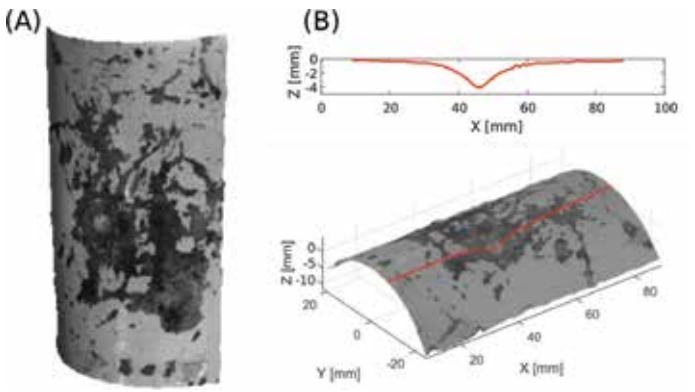

Figure 17. (a) 3D reconstructed shape. (b) Cross section of the 3D reconstruction.

Another application of FPP is in facial metrology, where several patterns are projected onto the face to obtain a 3D digital model. 3D shape measurement of faces plays an important role in several fields like in the biomedical sciences, biometrics, security, and entertainment. Human face models are widely used in medical applications for 3D facial expression recognition [24] and measurement of stretch marks [25]. Usually, the main challenge is the movement of the patient. The movement can produce errors or noise in the 3D reconstruction affecting its accuracy. Hence, 3D scanning techniques that require few images in the reconstruction process, like FTP, are commonly used. In **Figure 18** we show an experimental result of reconstructing a live human face. The captured image with the deformed fringe pattern is shown in **Figure 18(a)**. In **Figure 18(b)** and **(c)** we show the 3D geometry acquired rendered in shaded mode and with texture mapping, respectively. Note that several facial regions with hairs, like the eyebrows, are reconstructed with high detail. While other areas, under shadows, like the right side of the nose, are not correctly reconstructed.

Finally, another area where FPP has frequently been used is in cultural heritage preservation. The preservation of cultural heritage works requires accurately scanning sculptures, archeological remains, paintings, etc. In **Figure 19** we show the 3D reconstruction of a sculpture replica.

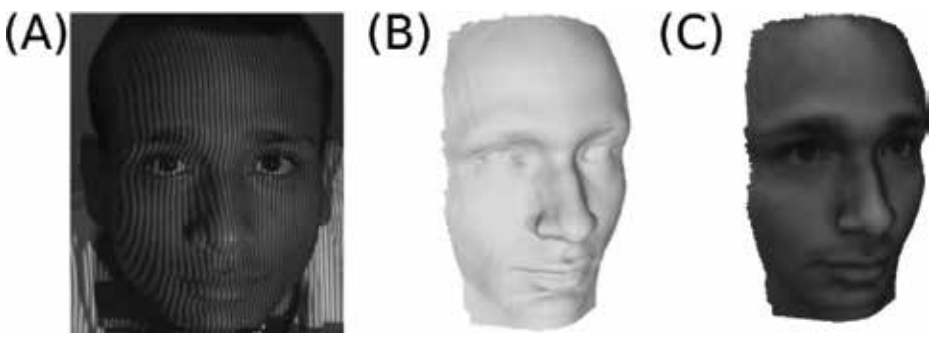

Figure 18. (a) Fringe pattern onto face. (b) 3D rendered model in shaded mode. (c) 3D rendered model with color texture mapping.

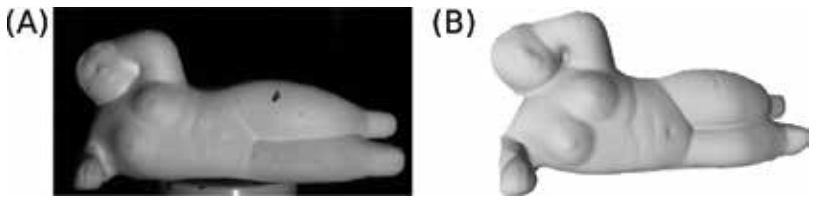

Figure 19. FTP 3D reconstruction of a sculpture replica of "Figura reclinada 92 - Gertrudis" by Fernando Botero [23]. (a) Texture image. (b) 3D reconstruction.

Acknowledgements

This work has been partly funded by Colciencias (Fondo Nacional de Financiamiento para la Ciencia, la Tecnología y la Innovación Francisco José de Caldas) project (538871552485) and by the Universidad Tecnológica de Bolívar (Dirección de Investigación, Emprendimiento e Innovación). J. Pineda and R. Vargas thank Universidad Tecnológica de Bolívar for a Master's degree scholarship

Author details

Andrés G. Marrugo[1]*, Jesús Pineda[1], Lenny A. Romero[2], Raúl Vargas[1] and Jaime Meneses[3]

*Address all correspondence to: agmarrugo@utb.edu.co

1 Facultad de Ingeniería, Universidad Tecnológica de Bolívar, Cartagena, Colombia

2 Facultad de Ciencias Básicas, Universidad Tecnológica de Bolívar, Cartagena, Colombia

3 Grupo de Óptica y Tratamiento de Señales, Universidad Industrial de Santander, Bucaramanga, Colombia

References

[1] Zhang S. Handbook of 3D Machine Vision: Optical Metrology and Imaging. CRC Press; 2013. pp. 1

[2] Gorthi SS, Rastogi P. Fringe projection techniques: Whither we are? Optics and Lasers in Engineering. 2010;**48**(2):133-140

[3] Zappa E, Busca G. Static and dynamic features of Fourier transform profilometry: A review. Optics and Lasers in Engineering. 2012;**50**(8):1140-1151

[4] Hariharan P, Oreb BF, Eiju T. Digital phase-shifting interferometry: A simple error-compensating phase calculation algorithm. Applied Optics. 1987;**26**(13):2504-2506

[5] Takeda M, Ina H, Kobayashi S. Fourier-transform method of fringe-pattern analysis for computer-based topography and interferometry. Journal of the Optical Society of America. 1982;**72**(1):156

[6] Su X, Zhang Q. Dynamic 3-D shape measurement method: A review. Optics and Lasers in Engineering. 2010;**48**(2):191-204

[7] Petitgrand S, Yahiaoui R, Danaie K, Bosseboeuf A, Gilles J. 3D measurement of micromechanical devices vibration mode shapes with a stroboscopic interferometric microscope. Optics and Lasers in Engineering. 2001;**36**(2):77-101

[8] Felipe-Sesé L, Siegmann P, Díaz FA, Patterson EA. Integrating fringe projection and digital image correlation for high-quality measurements of shape changes. Optical Engineering. 2014;**53**(4):044106

[9] Takeda M, Mutoh K. Fourier transform profilometry for the automatic measurement of 3-D object shapes. Applied Optics. 1983;**22**(24):3977

[10] Takeda M. Fourier fringe analysis and its application to metrology of extreme physical phenomena: A review [invited]. Applied Optics. 2013;**52**(1):20-29

[11] Lin JF, Su X. Two-dimensional Fourier transform profilometry for the automatic measurement of three-dimensional object shapes. Optical Engineering. 1995;**34**(11):3297-3302

[12] Ghiglia DC, Pritt MD. Two-Dimensional Phase Unwrapping: Theory, Algorithms, and Software. Vol. 4. New York: Wiley; 1998

[13] Huntley JM, Saldner H. Temporal phase-unwrapping algorithm for automated interferogram analysis. Applied Optics. 1993;**32**(17):3047-3052

[14] Liu H, Su W-H, Reichard K, Yin S. Calibration-based phase-shifting projected fringe profilometry for accurate absolute 3D surface profile measurement. Optics Communications. 2003;**216**(1):65-80

[15] Merner L, Wang Y, Zhang S. Accurate calibration for 3D shape measurement system using a binary defocusing technique. Optics and Lasers in Engineering. 2013;**51**(5):514-519

[16] Zhang S, Huang PS. Novel method for structured light system calibration. Optical Engineering. 2006;**45**(8):083601

[17] Li K, Bu J, Zhang D. Lens distortion elimination for improving measurement accuracy of fringe projection profilometry. Optics and Lasers in Engineering. 2016;**85**:53-64

[18] Arciniegas J, González AL, Quintero LA, Contreras CR, Meneses JE. Sistema de reconstrucción tridimensional aplicado a la exploración superficial de fallas y defectos en tuberías con refuerzo no metálico para el transporte de hidrocarburos. Revista Investigaciones Aplicadas. 2015;**9**(1):12-18

[19] Hu Q, Huang PS, Fu Q, Chiang F-P. Calibration of a three-dimensional shape measurement system. Optical Engineering. 2003;**42**(2):487-493

[20] Huang Z, Xi J, Yu Y, Guo Q. Accurate projector calibration based on a new point-to-point mapping relationship between the camera and projector images. Applied Optics. 2015; **54**(3):347-356

[21] Travis J, Kring J. LabVIEW for Everyone: Graphical Programming Made Easy and Fun. Prentice-Hall; 2007. pp. 10

[22] Easton RL Jr. Fourier Methods in Imaging. John Wiley & Sons; 2010. pp. 567

[23] IPCC. Monumento Artistico "Figura Reclinada de la Gorda Gertrudis" [Online]. 2016. Available from: http://www.ipcc.gov.co/index.php/noticias/item/260-botero

[24] Zhang S. Recent progresses on real-time 3d shape measurement using digital fringe projection techniques. Optics and Lasers in Engineering. 2010;**48**(2):149-158

[25] Gómez ALG, Fonseca JEM, Téllez JL. Proyección de franjas en metrología óptica facial. INGE CUC. 2012;**8**(1):191-206

Chapter 2

Recent Advances in Variable Digital Filters

Shunsuke Koshita, Masahide Abe and
Masayuki Kawamata

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.79198

Abstract

Variable digital filters are widely used in a number of applications of signal processing because of their capability of self-tuning frequency characteristics such as the cutoff frequency and the bandwidth. This chapter introduces recent advances on variable digital filters, focusing on the problems of design and realization, and application to adaptive filtering. In the topic on design and realization, we address two major approaches: one is the frequency transformation and the other is the multi-dimensional polynomial approximation of filter coefficients. In the topic on adaptive filtering, we introduce the details of adaptive band-pass/band-stop filtering that include the well-known adaptive notch filtering.

Keywords: variable digital filter, frequency transformation, polynomial approximation, adaptive notch filtering, adaptive band-pass/band-stop filtering

1. Introduction

Digital filter is well known as one of the essential and fundamental components in signal processing devices. In addition, many signal processing applications such as digital audio equipment and telecommunication systems sometimes require simultaneous realization of digital filtering and real-time control of filter characteristics. Such requirements can be fulfilled by means of variable digital filters (VDFs). Research on VDFs emerged in the 1970s and since then, many results have been reported. Among them, details of the results until the 1990s are widely reviewed in [1].

The problems that should be solved in development of VDFs are essentially the same as those in digital filters of fixed characteristics. Hence, research topics on VDFs as well as fixed characteristic filters are broadly classified into three categories [2]: the approximation problem, the realization

problem, and the implementation problem. Moreover, in the field of VDFs, application-oriented results have also been actively reported. One of the famous applications is adaptive notch filters, which have been studied since the 1980s and the details will be reviewed in this chapter.

In the sequel, fundamentals of VDFs are first reviewed. Then, recent results on VDFs are introduced and discussed with focus on the approximation problem, the realization problem, and the applications. Such topics include some results proposed by the authors of this chapter.

2. Fundamentals of VDFs

2.1. Definition

VDFs are defined as the frequency selective digital filters (e.g., low-pass filters and band-pass filters) of which frequency characteristics can be changed in real time by means of controlling some parameters. A popular example of such VDFs is shown in **Figure 1**, which is the variable low-pass filter (VLPF) of which cutoff frequency can be changed by controlling the single parameter η. Another example shown in **Figure 2** is the variable band-pass filter (VBPF), where the bandwidth is fixed and the pass-band center frequency can be changed by the single parameter ξ.

It should be noted that VDFs are different from "filters with variable (adjustable) coefficients" which are used in adaptive filtering. Details of the differences are as follows:

- In the case of general adaptive filtering, all filter coefficients are changed by an adaptive algorithm. On the other hand, most of the coefficients of a VDF are fixed or given as some functions of a few variable parameters. For example, in the VLPF of **Figure 1**, only the single parameter η can be changed, and the other coefficients are fixed or given as functions of η.
- VDFs are different from general adaptive filters with respect to the mechanism of changing the frequency characteristics. In VDFs, the characteristics are changed but the frequency selectivity such as the low-pass and the band-pass shape is preserved. In other words, VDFs control the frequency characteristics under the constraint of preservation of frequency selectivity. On the other hand, general adaptive filters do not require this constraint. This means that such adaptive filters converge to optimal ones of which characteristics do not necessarily possess frequency selectivity.

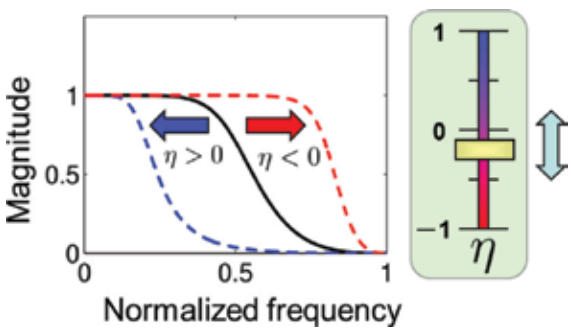

Figure 1. Example of VLPF.

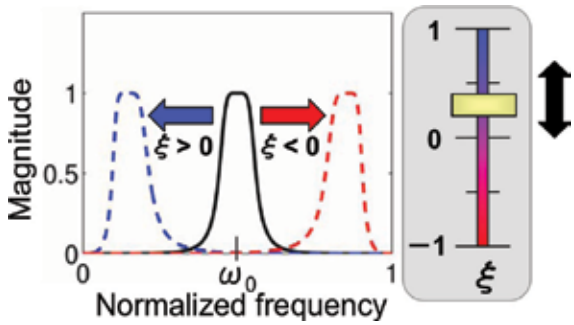

Figure 2. Example of VBPF.

Figure 3. Procedure to obtain VDF.

2.2. How to obtain VDFs

This subsection reviews the procedure to obtain VDFs. The required procedure is basically the same as that in the case of fixed characteristic filters, where three important problems must be considered as shown in **Figure 3**: *approximation*, *realization*, and *implementation* [2]. In this chapter, we pay special attention to the approximation problem and the realization problem. The approximation problem is to obtain an input-output characterization such as transfer function from a prescribed specification of a VDF. The realization problem is to determine a structure (i.e., an appropriate set of adders and multipliers or an appropriate list of primitive operations for filtering) corresponding to the input-output characterization.

In the approximation problem for VDFs, the required task is to describe an input-output relationship (e.g., transfer function) of the VDF in such a manner that the description includes variable parameters. For example, consider the approximation problem for the VLPF shown in **Figure 1**. If one wishes to obtain this VLPF as an FIR filter, the approximation problem is to describe the transfer function in the form of

$$H(z,\eta) = \sum_{k=0}^{N} h_k(\eta) z^{-k} \tag{1}$$

and it is also necessary to describe each coefficient $h_k(\eta)$ as a function of η. Therefore, the approximation problem for this VLPF is to determine a set of functions $\{h_k(\eta)\}$ $(0 \le k \le N)$. Similarly, if one wishes to obtain IIR-type VLPF, it is necessary to describe the transfer function in the form of

$$H(z,\eta)=\frac{\sum_{k=0}^{M}b_k(\eta)z^{-k}}{1+\sum_{m=1}^{N}a_m(\eta)z^{-m}} \tag{2}$$

and to determine the filter coefficients as the functions $\{a_m(\eta)\}$ $(1\le m\le N)$ and $\{b_k(\eta)\}$ $(1\le k\le M)$.

3. Research topics on VDFs

This section introduces research topics on VDFs from the viewpoints of the approximation problem and the realization problem. Two methods have been widely used for approximation and realization of VDFs: one is based on the variable transformation of transfer functions and the other is based on the multi-dimensional (M-D) polynomial approximation of filter coefficients. In the sequel details of these two methods are reviewed and some recent results on these two methods are introduced.

3.1. VDFs based on variable transformation of transfer functions

In this method, we first need to design the transfer function of "prototype filter," which is usually low pass, and its coefficients are fixed (i.e., variable parameters are not included in this transfer function). Next, we apply a variable transformation to this prototype transfer function and obtain a desired VDF, where the variable transformation makes use of a function which includes variable parameters that are associated with the components to be changed in frequency characteristics. Many approaches exist for variable transformations, and the most famous approach is the frequency transformation [3]. The frequency transformation makes use of all-pass functions for the variable transformation. Although details of the frequency transformation are well reviewed in [1], this chapter will also review this topic with some additional discussions. This is because many results using the frequency transformation have been still reported in recent years and some of such results include the authors' works.

Now, consider again the VLPF shown in **Figure 1**. If frequency transformation is used to obtain this VLPF, the first step is to prepare the transfer function of a prototype low-pass filter. Such a transfer function is denoted by $H_{\mathrm{p}}(z)$. Then, applying the following frequency transformation to $H_{\mathrm{p}}(z)$, we can obtain the desired VLPF with the transfer function $H(z,\eta)$:

$$\begin{aligned} H(z,\eta)&=H_{\mathrm{p}}(z)\big|_{z^{-1}\leftarrow T(z,\eta)}\\ T(z,\eta)&=\frac{z^{-1}-\eta}{1-\eta z^{-1}} \end{aligned} \tag{3}$$

where $T(z,\eta)$ is the first-order all-pass function. By changing the value of η in $H(z,\eta)$, we can control the cutoff frequency of the VLPF. If $\eta>0$, the cutoff frequency becomes lower than that of the prototype filter. The converse holds if $\eta<0$. Stability of this VLPF is guaranteed if the

prototype filter is stable and $|\eta| < 1$ is satisfied. Also, note that $|T(e^{j\omega}, \eta)| = 1$ holds for any η and ω because $T(z, \eta)$ is all-pass.

We next discuss the realization problem for this VLPF. From the realization point of view, Eq. (3) means that a block diagram of this VLPF can be obtained by replacing each delay element z^{-1} in the prototype filter with the all-pass filter $T(z, \eta)$. However, in most cases, such replacement causes delay-free loops and results in $H(z, \eta)$ with unrealizable block diagram. To explain this problem, consider a second-order IIR prototype filter with the transfer function given by

$$H_{\mathrm{p}}(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \tag{4}$$

and the block diagram given by the direct form as in **Figure 4(a)**. Applying the aforementioned replacement of delay elements with $T(z, \eta)$ yields the VLPF of which the block diagram corresponds to **Figure 4(b)**. It is now clear that **Figure 4(b)** includes delay-free loops, and hence it is impossible to implement this block diagram. It is well known that delay-free loops can be avoided by means of mathematical manipulations of transfer function or difference equation. However, such manipulations are not good solutions in the case of VDF realization. For example, applying $z^{-1} \leftarrow T(z, \eta)$ to $H_{\mathrm{p}}(z)$ given by Eq. (4) and then performing mathematical manipulations, we obtain the transfer function of the second-order VLPF as follows:

$$\begin{aligned}
H(z, \eta) &= \frac{b_0'(\eta) + b_1'(\eta) + b_2'(\eta)}{1 + a_1'(\eta) + a_2'(\eta)} \\
a_1'(\eta) &= \frac{-2\eta + a_1(1 + \eta^2) - 2a_2\eta}{1 - a_1\eta + a_2\eta^2} \\
a_2'(\eta) &= \frac{\eta^2 - a_1\eta + a_2}{1 - a_1\eta + a_2\eta^2} \\
b_0'(\eta) &= \frac{b_0 - b_1\eta + b_2\eta^2}{1 - a_1\eta + a_2\eta^2} \\
b_1'(\eta) &= \frac{-2b_0\eta + b_1(1 + \eta^2) - 2b_2\eta}{1 - a_1\eta + a_2\eta^2} \\
b_2'(\eta) &= \frac{b_0\eta^2 - b_1\eta + b_2}{1 - a_1\eta + a_2\eta^2}.
\end{aligned} \tag{5}$$

If we implement the VLPF using this description, the computational cost significantly increases because the filter coefficients $a_1'(\eta)$, $a_2'(\eta)$, $b_0'(\eta)$, $b_1'(\eta)$ and $b_2'(\eta)$ must be recalculated according to the change of η. In particular, the filter coefficients in Eq. (5) are rational polynomials that require divisions for recalculation of filter coefficients, causing very high implementation cost.

One of the popular methods to overcome this problem is the Taylor approximation-based description [4]. This method applies the first-order Taylor series approximation to all of the rational polynomials of filter coefficients in VDFs, under the assumption that the absolute

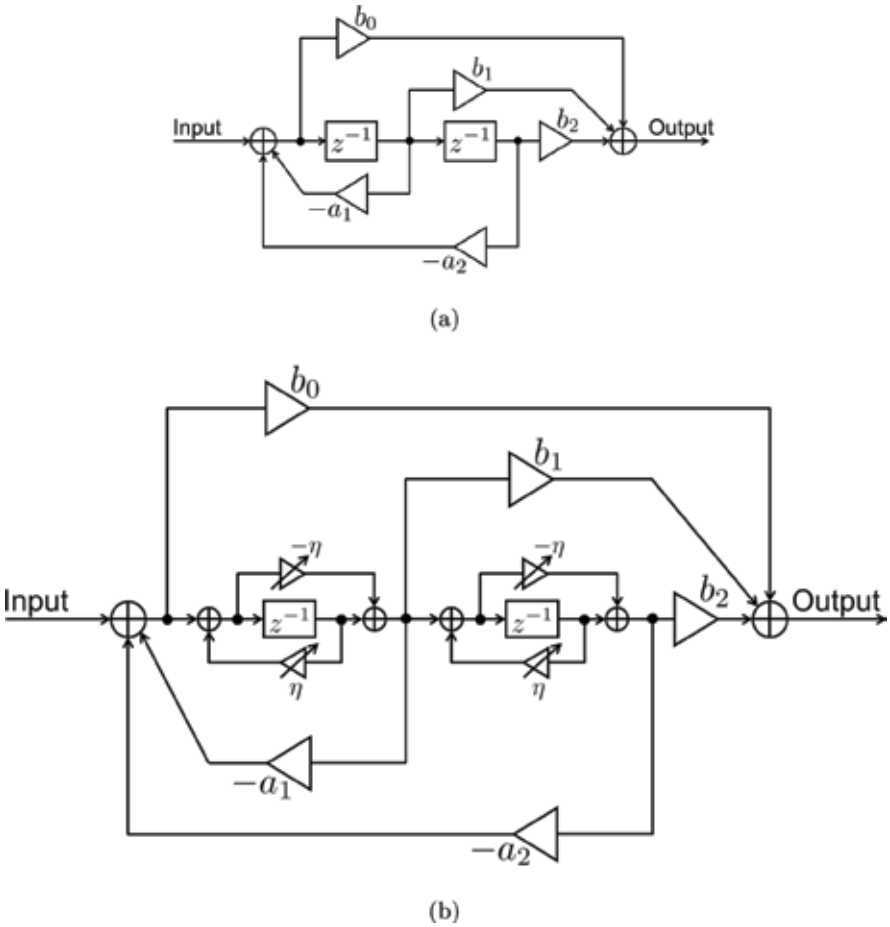

Figure 4. Problem in realization of VLPF based on the frequency transformation: (a) second-order prototype filter, and (b) VLPF given by applying $z^{-1} \leftarrow T(z, \eta)$ to the prototype filter.

values of all variable parameters are small. For example, in the case of Eq. (5), it is assumed that $|\eta| \ll 1$ and the filter coefficients are approximated to

$$\begin{aligned} a_1'(\eta) &\approx a_1 + \left(a_1^2 - 2 - 2a_2\right)\eta \\ a_2'(\eta) &\approx a_2 + (a_1 a_2 - a_1)\eta \\ b_0'(\eta) &\approx b_0 + (a_1 b_0 - b_1)\eta \\ b_1'(\eta) &\approx b_1 + (a_1 b_1 - 2b_0 - 2b_2)\eta \\ b_2'(\eta) &\approx b_2 + (a_1 b_2 - b_1)\eta. \end{aligned} \tag{6}$$

These new coefficients do not require divisions, and hence the VLPF can be realized in terms of additions and multiplications, as shown in **Figure 5**. In addition, this realization does not

http://dx.doi.org/10.5772/intechopen.79198

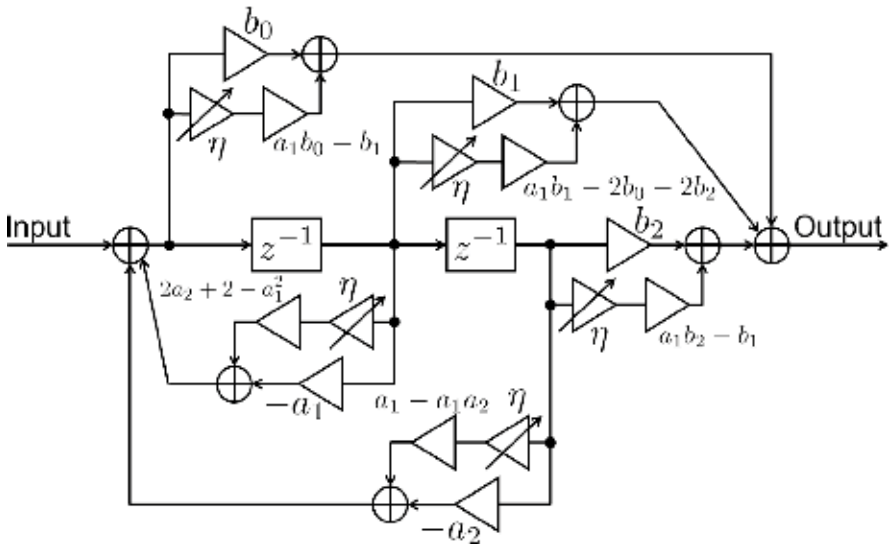

Figure 5. Second-order VLPF based on the frequency transformation and first-order Taylor series approximation.

require recalculation of filter coefficients even if the value of η is changed. This is because all of the multipliers except for η in this block diagram are realized as fixed coefficients.

Although the VLPFs based on the Taylor approximation provide an effective realization method, they have a serious drawback that the range of variable cutoff frequency is quite limited. This limitation is due to the assumption of $|\eta| \ll 1$, which means that the approximation error becomes larger as the cutoff frequency of the VLPFs goes far from that of the prototype filter. In addition, the VLPFs may become unstable if the value of $|\eta|$ is inappropriately large. In order to overcome these problems, some alternative methods are proposed [4–6]. All of these methods make use of low sensitivity structures for realization of block diagrams for the prototype filter. Then the replacement $z^{-1} \leftarrow T(z, \eta)$ and the Taylor approximation are applied to such block diagrams, leading to the desired VDFs. Although the methods given by [4–6] can be applied to the limited classes of transfer functions, the Taylor approximation error becomes smaller than the standard VDFs based on the direct form. This approach is also extended to the 2-D VDFs [7].

There are some other approaches for the reduction of the Taylor approximation error. In [8], the approach based on wave digital filters is presented. Although this approach requires the knowledge of analog filter theory, very high precision is attained in the resultant VDFs, and hence the variable cutoff frequency can be controlled in relatively wide range. In [9], state-space representation is used for construction of the block diagram of the prototype filter, and series approximations are applied to avoid the significant increase of the implementation cost of frequency transformation-based VDFs. This approach does not need any restriction that appeared in the conventional methods, and hence the method of [9] can be applied to arbitrary transfer functions and arbitrary state-space structures. Furthermore, in [10], the VDFs based on the combination of frequency transformation and coefficient decimation are proposed, and it is

shown through FPGA implementation and performance evaluation that the proposed method attains very low cost for hardware implementation.

As discussed above, the problem of delay-free loops is an important issue in the approximation/realization of frequency transformation-based VDFs. It should be noted that, however, this problem does not always happen. In general, this problem happens if the all-pass function in the frequency transformation has a nonzero constant term in the numerator. This case corresponds to the VDFs with variable bandwidth. In other words, the problem of delay-free loops does not happen when the VDFs have fixed bandwidth, as shown in **Figure 2**.

We conclude this subsection with a summary of the merits and the drawbacks of the frequency transformation-based VDFs. The merits are as follows:

- Variable characteristics can be easily obtained because the theory of controlling cutoff frequency is based on the simple variable transformations.
- If Taylor approximation is not carried out, the frequency transformation preserves many useful properties on the shape of magnitude responses. For example, when a prototype low-pass filter is the Butterworth filter that possesses the monotonic and maximally flat magnitude response, the VDFs given by applying frequency transformations to this prototype filter also possess the monotonic and maximally flat magnitude responses.
- The aforementioned merit facilitates the design of adaptive band-pass or band-stop filters because the cost function for adaptive filtering becomes unimodal, leading to an adaptive algorithm that converges to the globally optimal solution. Details will be discussed in the next section.
- Compared with the VDFs based on the M-D polynomial approximation, the frequency transformation-based VDFs require much less computational cost in the filtering.

Next, the drawbacks are summarized as follows:

- As stated earlier, if the bandwidth needs to be variable in VDFs, the frequency transformation causes delay-free loops and this problem must be appropriately solved.
- If one wishes to obtain VDFs with multiple passbands or stopbands such as VBPFs, VBSFs, and variable multi-band filters, it is necessary to use high-order all-pass functions for the frequency transformation. As a result, the order of VDFs becomes higher than that of the prototype filter. For example, the order of the frequency transformation-based VBPFs and VBSFs becomes doubled as compared with the order of the prototype filter.
- Linear-phase VDFs cannot be obtained because the all-pass functions to be used in the frequency transformation are IIR filters. Even if a prototype filter is FIR, applying the frequency transformations simply results in IIR-type VDFs.
- Realization of variable characteristics is quite limited. To be specific, the frequency transformation can provide only the VDFs with variable cutoff frequencies. In other words, other components such as the transition bandwidth and the stopband attenuation cannot be controlled.

3.2. VDFs based on M-D polynomial approximation of filter coefficients

To the authors' best knowledge, the VDFs based on the M-D polynomial approximation of filter coefficients have been most actively studied [11–23] in the field of VDFs. One of the significant benefits of this approach over the frequency transformation-based VDFs is that many kinds of variable characteristics as well as variable cutoff frequencies can be attained. For example, this approach can provide VLPFs with variable transition bandwidth and variable stopband attenuation, as shown in **Figure 6**. In addition, since this approach is applicable to FIR filters as well as IIR filters, linear-phase characteristics and variable group delay can be attained in VDFs.

The first step to obtain this type of VDFs is to determine a set of K variable parameters $(\psi_1, \psi_2, \cdots, \psi_K)$ which correspond to the desired variable components of frequency characteristics such as cutoff frequency, transition bandwidth, and stopband attenuation. Such variable parameters are referred to as spectral parameters. After this step, filter coefficients of the desired VDFs are described as M-D polynomials with respect to these variable parameters. For example, the transfer function of an N-th order VDF with K variable parameters is described by

$$H(z, \psi_1, \psi_2, \cdots, \psi_K) = \sum_{n=0}^{N} h_n(\psi_1, \psi_2, \cdots, \psi_K) z^{-n} \tag{7}$$

and each filter coefficient $h_n(\psi_1, \psi_2, \cdots, \psi_K)$ is described in terms of the following M-D polynomial:

$$h_n(\psi_1, \psi_2, \cdots, \psi_K) = \sum_{m_{\psi_1}=0}^{M_{\psi_1}} \sum_{m_{\psi_2}=0}^{M_{\psi_2}} \cdots \sum_{m_{\psi_K}=0}^{M_{\psi_K}} c_n(m_{\psi_1}, m_{\psi_2}, \cdots m_{\psi_K}) \psi_1^{m_{\psi_1}} \psi_2^{m_{\psi_2}} \cdots \psi_K^{m_{\psi_K}}. \tag{8}$$

The approximation problem for this kind of VDFs is to determine the set of coefficients $\{c_n(m_{\psi_1}, m_{\psi_2}, \cdots m_{\psi_K})\}$ for $0 \le n \le N$. Here, it should be noted that $M_{\psi_1}, M_{\psi_2}, \cdots, M_{\psi_K}$ denote the orders of the M-D polynomials that, respectively, correspond to the variables $\psi_1, \psi_2, \cdots, \psi_K$. In order to obtain the set $\{c_n(m_{\psi_1}, m_{\psi_2}, \cdots m_{\psi_K})\}$, the standard approach is based on the

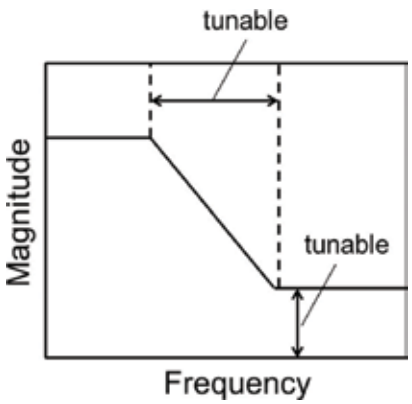

Figure 6. Example of VLPF based on the M-D polynomial approximation of filter coefficients.

minimization of an error function with respect to approximation of a prescribed ideal characteristic of the desired VDF and a curve fitting method to describe the desired M-D polynomials.

In realization of the VDFs given as above, Farrow structure [24] is widely used. To explain this, consider a simple VDF with a single variable parameter ψ_1. The transfer function of this VDF is given by

$$\begin{aligned} H(z,\psi_1) &= \sum_{n=0}^{N} h_n(\psi_1) z^{-n} \\ &= \sum_{n=0}^{N} \sum_{m_{\psi_1}=0}^{M_{\psi_1}} c_n(m_{\psi_1}) \psi_1^{m_{\psi_1}} z^{-n} \end{aligned} \tag{9}$$

which can be rewritten as

$$H(z,\psi_1) = \sum_{m_{\psi_1}=0}^{M_{\psi_1}} \left(\sum_{n=0}^{N} c_n(m_{\psi_1}) z^{-n} \right) \psi_1^{m_{\psi_1}}. \tag{10}$$

Now, by using the following definition

$$H_{m_{\psi_1}}(z) = \sum_{n=0}^{N} c_n(m_{\psi_1}) z^{-n}, \quad 0 \le m_{\psi_1} \le M_{\psi_1}, \tag{11}$$

the description of the VDF $H(z,\psi_1)$ becomes

$$H(z,\psi_1) = \sum_{m_{\psi_1}=0}^{M_{\psi_1}} H_{m_{\psi_1}}(z) \psi_1^{m_{\psi_1}}. \tag{12}$$

Using this description, we can realize $H(z,\psi_1)$ by means of the Farrow structure as shown in **Figure 7**. The block diagram of **Figure 7** is interpreted as the parallel combination of the set of N-th order FIR filters with fixed coefficients and the weights ψ_1. Since these N-th order FIR

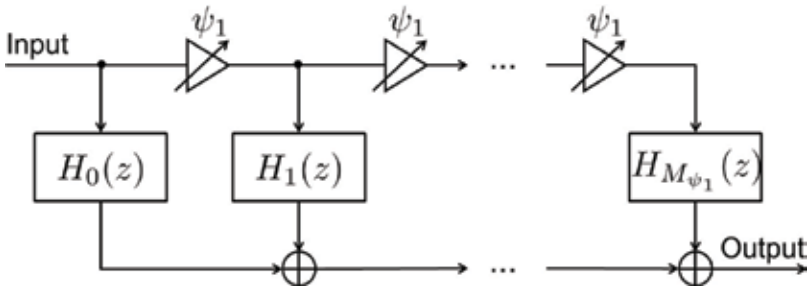

Figure 7. Realization of M-D polynomial approximation-based VDF based on the Farrow structure.

filters do not include ψ_1, recalculation of their coefficients according to the change of ψ_1 is not required. In this sense, the Farrow structure is suitable for the implementation of M-D polynomial approximation-based VDFs.

A drawback of the M-D polynomial approximation-based VDFs is the high computational cost in the filtering because the filter coefficients are described by M-D polynomials. In addition, this approach limits the range of variable characteristics. As in the case of frequency transformation with Taylor approximation, this limitation comes from the M-D polynomial approximation. Furthermore, since this approach requires a number of filters with fixed coefficients, their hardware implementation may cause an increase of characteristic degradations that comes from finite wordlength effects such as coefficient sensitivity and roundoff noise. However, such degradations can be suppressed by using high accuracy filter structures, and this approach has been recently proposed by the authors [23].

3.3. VDFs based on other approaches

In addition to the aforementioned two approaches, many other methods have also been presented in the literature. In [25], VDFs with variable bandwidth without delay-free loops can be achieved at low cost by means of cascade connection of a single subfilter. In [26–28], by applying the frequency response masking and the fast filterbank to design of VDFs, significant reduction of implementation cost over the VDFs with the Farrow structure is attained.

Also, VDFs for adaptive filtering have been widely studied. One of the famous methods in such VDFs is the variable notch filters with second-order IIR transfer functions. All of these variable notch filters successfully provide the variable characteristics by simple mechanism without delay-free loops or increase of computational cost. Other adaptive-filter-oriented VDFs include notch filters with variable attenuation at the notch frequency, comb filters with variable bandwidth, and variable attenuation. Details of these topics will be addressed in the next section.

4. Research topics on VDFs for adaptive filtering

In this section, we first pay attention to adaptive notch filters (ANFs) that are the special case of adaptive band-stop or band-pass filters. The ANFs are the most famous application of VDFs to adaptive signal processing, and many results on the ANFs have been reported since the 1980s. In addition to the ANFs, this section also introduces some other types of VDFs that are applied to adaptive filtering.

4.1. ANF based on all-pass filter

As shown in **Figure 8**, an ANF plays a central role in automatic detection and suppression of an unknown sinusoid immersed in a wide-band signal such as white noise. In order to detect and suppress the sinusoid, the ANF is controlled by an adaptive algorithm in such a manner that the notch frequency ω_0 of the ANF converges to the unknown frequency ω_s of the

sinusoid. Hence, the ANF can be considered as the VDF with variable notch frequency, and the value of ω_0 at the steady state becomes the estimate of the frequency ω_s of the sinusoid. Therefore, ANFs are used not only for the detection/suppression of a sinusoid, but also for the frequency estimation.

Although the ANF shown in **Figure 8** is intended to suppress a sinusoid, the ANF is also capable of enhancement of the sinusoid and suppression of the white noise. This can be achieved by using a peaking filter, which is also called a resonator or an inverse notch filter, as an adaptive filter instead of using a notch filter. Alternatively, the notch filter can also be used: in this case, the sinusoid can be enhanced by subtracting the output of the notch filter from the input signal.[1] Such systems together with the ones shown in **Figure 8** are widely used in many practical applications such as radar, sonar, telecommunication system with the suppression of narrowband interference and howling suppressor in speech processing.

In the sequel, we explain the fundamentals of ANFs, that is, their problem statement and the mechanism of control of the notch frequency. As shown in **Figure 8**, the problem statement of ANFs usually describes the input signal as the sum of a sinusoid and a white noise. Hence, the input signal, denoted by $u(n)$, is given by

$$u(n) = A\sin\left(\omega_s n + \phi\right) + w(n) \tag{13}$$

where A and ω_s are, respectively, the amplitude and frequency of the unknown sinusoid, and ϕ is the random initial phase uniformly distributed in $[0, 2\pi)$. The signal $w(n)$ is a zero-mean white noise, and it is uncorrerated to ϕ. Based on this setup, let $y(n)$ be the output signal of the ANF.

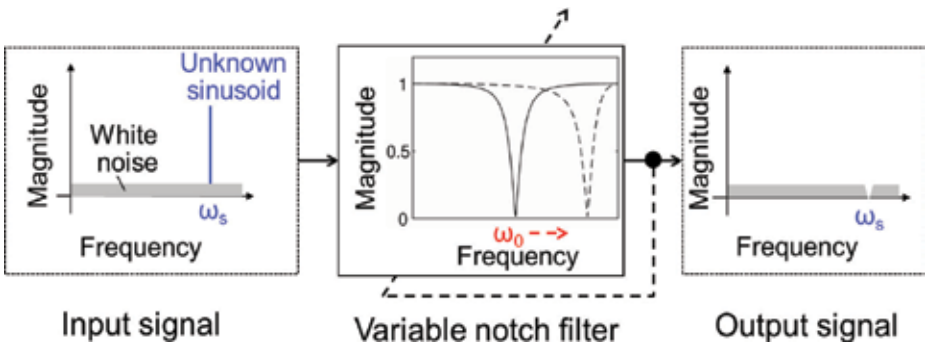

Figure 8. Detection and suppression of sinusoid using ANF.

[1]Note that this approach depends on the characteristic of a notch filter, and hence the use of an inappropriate notch filter may result in failure of enhancement of a sinusoid. The reason of this lies in the fact that the signal which is obtained by subtracting the output of a band-stop filter from the input is not necessarily equivalent to the output of a band-pass filter. However, in the case of the ANF based on a second-order all-pass filter, the frequency characteristic of the notch filter satisfies complementary properties that allow us to successfully obtain a signal equivalent to the band-pass-filtered signal by subtracting the notch-filtered signal from the input.

There are some methods to describe the transfer function of the notch filter for adaptive filtering. In this chapter we focus on the one based on the second-order all-pass filter [29]. This notch filter is described by the following transfer function

$$H(z,\eta,\xi) = \frac{1}{2}(1 + T(z,\eta,\xi)) \tag{14}$$

where $T(z,\eta,\xi)$ is the second-order all-pass filter of the form

$$T(z,\eta,\xi) = \frac{\eta - (1+\eta)\xi z^{-1} + z^{-2}}{1 - (1+\eta)\xi z^{-1} + \eta z^{-2}}. \tag{15}$$

Hence Eq. (14) is described as

$$H(z,\eta,\xi) = \frac{1+\eta}{2}\frac{1 - 2\xi z^{-1} + z^{-2}}{1 - (1+\eta)\xi z^{-1} + \eta z^{-2}}. \tag{16}$$

In this notch filter, the parameter η determines the 3-dB notch width, and the parameter ξ determines the notch frequency ω_0. This means that the notch filter given in this way can control the notch width and the notch frequency independently. Also, it is interesting to note that this notch filter can be interpreted as a VDF given by the frequency transformation [30]: it is clear that this notch filter is obtained by applying the frequency transformation $z^{-1} \leftarrow T(z,\eta,\xi)$ to the prototype filter of the form

$$H_{\mathrm{p}}(z) = \frac{1}{2}\left(1 + z^{-1}\right). \tag{17}$$

To be more precise, this notch filter has the same transfer function as that of the second-order Butterworth band-stop filter [31]. Therefore this notch filter has unity gain at $\omega = 0$ and $\omega = \pi$, and zero gain at ω_0. In addition, the magnitude response of this notch filter is monotonically decreasing in $0 < \omega < \omega_0$ and monotonically increasing in $\omega_0 < \omega < \pi$.

Figure 9 shows the block diagram of ANF based on this notch filter. As stated earlier, when the ANF attains steady state, the component of the sinusoid in the input $u(n)$ is suppressed at

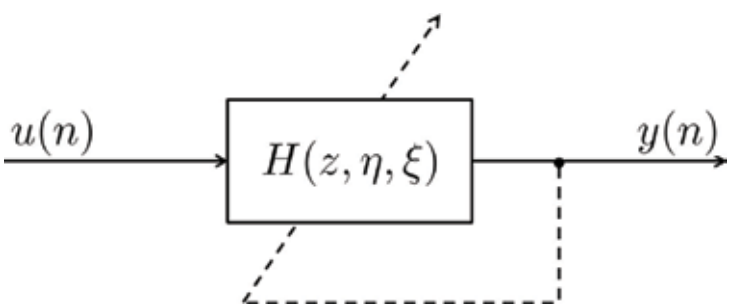

Figure 9. ANF based on the second-order all-pass filter.

the output signal $y(n)$. Here, it should be noted that many adaptive algorithms assume that the notch width is fixed, and that only the notch frequency ω_0 is controlled to estimate the frequency of the sinusoid. For this reason, we focus on how to control ω_0.

The most standard method to control ω_0 is based on the minimization of a cost function by means of the gradient descent method. Although this is similar to general adaptive filters, ANFs differ from the general adaptive filters in that the cost function to be used in ANFs is the mean square output, that is, $E\left[y^2(n)\right]$. In other words, ANFs do not usually deal with the error signal between a reference signal and the filter output.[2] Since ANFs control ω_0, the cost function $E\left[y^2(n)\right]$ must be formulated as a function of ω_0. This can be successfully achieved and, in addition, $E\left[y^2(n)\right]$ becomes unimodal if the input signal is given as in Eq. (13) and the ANF has monotonic magnitude response. Therefore, in such a case, the optimal notch frequency that minimizes $E\left[y^2(n)\right]$ can be successfully found by the gradient descent method. In fact, the optimal value of ω_0 coincides with ω_s if the all-pass-based ANF is used [32–34]. Hence, using the gradient of $E\left[y^2(n)\right]$ with respect to ω_0 in an adaptive algorithm allows ω_0 to converge to ω_s, leading to detection/suppression of the sinusoid.

Remark 1 *If the transfer function of the ANF is not based on the all-pass function, the optimal value of ω_0 may slightly deviate from ω_s. In other words, the frequency estimation is biased. This topic will be addressed in the next subsection.*

However, the gradient descent method has a serious drawback that the convergence speed becomes very slow when the initial value of ω_0 is distant from ω_s. To overcome this problem, many strategies have been proposed. In [32–34], the normalized lattice structure is applied to construct the notch filter, and the adaptive algorithm makes use of the state variable of the normalized lattice structure instead of the information of the gradient. This approach is called the Simplified Lattice Algorithm (SLA) and successfully accelerates the convergence speed at low computational cost. Furthermore, in [35], the authors have extended the SLA and proposed a new algorithm called the Affine Combination Lattice Algorithm (ACLA), and it has been proved that the ACLA achieves faster convergence than the SLA. Other approaches to improve the convergence speed include the methods based on the least square algorithm with forgetting factor [36], parallel combination of multiple notch filters with different notch width [37, 38], and construction of additional monotonically increasing function for the gradient [39, 40].

There are many other important research topics on the ANFs. One of them is the theoretical analysis of the behavior of ANFs at steady state. In [41], a steady-state analysis is presented for ANFs based on the one-multiplier lattice structure. This analysis enables us to evaluate the performance of ANFs such as the accuracy of frequency estimation. Also, in [42], the authors propose a unified method on the steady-state analysis of frequency estimation MSE (mean square error) for the SLA and the ACLA. As another research topic, in [43] fundamental frequency estimation using inverse notch filter is proposed.

---

[2] Although some literature refers to $y(n)$ as the error signal, in the authors' opinion this terminology is incorrect. This is because the error in ANFs should be defined as the difference between the frequency of the sinusoid and its estimate, i.e. $\omega_s - \omega_0$. This quantity clearly differs from $y(n)$.

4.2. ANFs based on other approaches

Other types of ANFs have also been well studied. For example, the following second-order notch filter [44] is very well known:

$$H(z, r, a) = \frac{1 + az^{-1} + z^{-2}}{1 + arz^{-1} + r^2 z^{-2}} \tag{18}$$

where a and r correspond to the parameters that, respectively, control the notch frequency and the notch width. Hence, in this case, the parameter a is controlled by an adaptive algorithm to estimate ω_s. This notch filter is designed by the famous method called the constrained poles and zeros (CPZ), and this notch filter has been most widely used for ANFs [44–51].

Since the transfer function of this notch filter is different from that of the all-pass-based notch filter, the properties of these notch filters are also somewhat different. For example, the all-pass-based notch filter has the unity peak gain, whereas the peak gain of the CPZ-based notch filter depends on the notch width. This also makes the difference with respect to the value of $E\left[y^2(n)\right]$, see [52] for the details. Another difference between these two notch filters is that the all-pass-based ANFs provide unbiased frequency estimation, whereas the CPZ-based ANFs do not. Although this fact shows a drawback of the CPZ-based ANFs, many adaptive algorithms to reduce the bias have been proposed for the CPZ-based ANFs.

In addition to the CPZ-based notch filters, there exist many other types of notch filters. In [53, 54], the specific second-order transfer function is constructed in such a manner that it corresponds to a lattice structure. In [55–58], the bilinear transformation to a second-order analog filter is applied to the notch filter design. In [59], the frequency transformation is used to design a notch filter, but the prototype filter used here is different from Eq. (17).

4.3. Adaptive filtering based on high-order VBPFs/VBSFs

All of the adaptive filters that were addressed in previous subsections are based on second-order VDFs. On the other hand, there exist some results on high-order VDFs in adaptive signal processing. Needless to say, second-order ANFs have a drawback that it is difficult to realize sharp cutoff characteristics, causing insufficient frequency selectivity and relatively poor signal-to-noise ratio (SNR) at the output signal. On the other hand, in [31], the authors improve the output SNR by means of higher-order VBPFs or VBSFs instead of using second-order notch filters in the adaptive filtering. As shown in **Figure 10**, high-order filters can realize sharper cutoff characteristics than second-order filters and provide higher output SNR.

Compared with ANFs, little has been studied on the adaptive filtering based on the high-order VDFs. To the authors' best knowledge, the most significant work is found in [60–63], where fourth-order Butterworth VBPF and VBSF are applied to adaptive filtering, and their center frequencies are controlled by adaptive algorithms. Furthermore, the convergence characteristics are also theoretically analyzed. In this work, it is also claimed that the use of much higher-order VDFs for adaptive filtering is almost impossible because higher-order transfer functions involve mathematically more complicated descriptions, and hence it is conjectured that formulations of filter coefficients with variable characteristics and adaptive

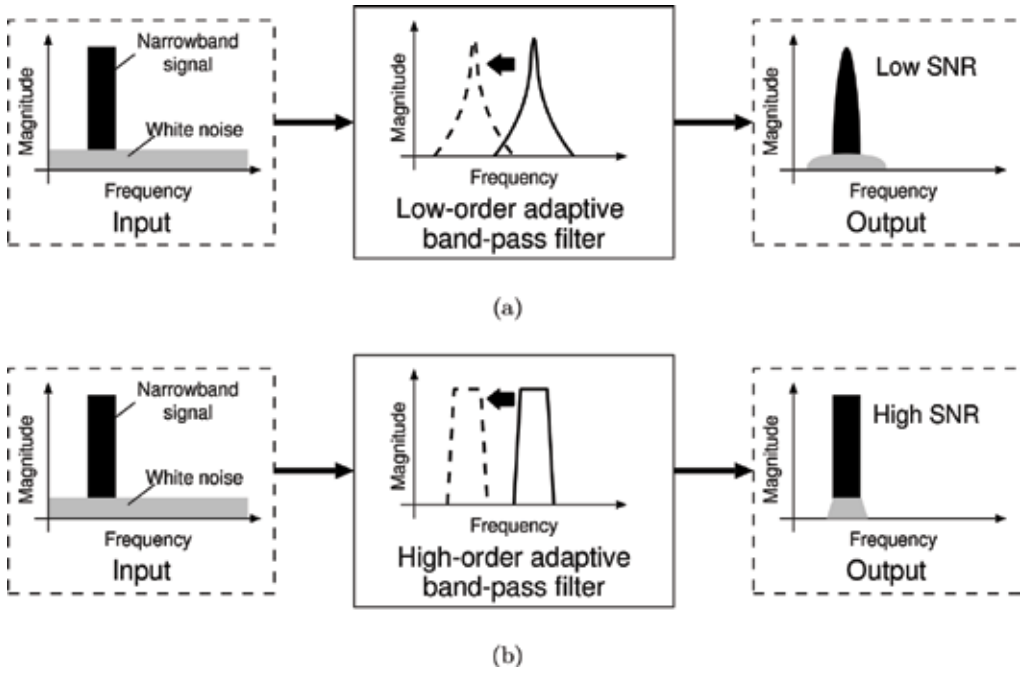

Figure 10. Example of adaptive band-pass filtering: (a) using second-order VBPF, and (b) using high-order VBPF.

control of them become very complicated. However, in the authors' recent work [31], we have successfully realized adaptive filtering based on higher-order VBPFs/VBSFs, where we have derived a gradient descent method-based adaptive algorithm for arbitrary-order VBPFs/VBSFs in a simple form by means of frequency transformation in terms of the block diagram as well as the mathematical description. As a result, it is demonstrated in [31] that the use of higher-order VBPFs/VBSFs for adaptive filtering leads to higher output SNR than the use of ANFs.

As stated above, adaptive band-pass/band-stop filtering based on high-order VBPFs/VSFs can be realized in a simple manner. However, there are still many open problems such as mathematical discussion of convergence of the adaptive algorithm, improvement of convergence speed, and suppression of large quantization errors that are generated due to the nature of high-order narrowband filters. Although the problem of quantization errors can be solved by means of the state-space-based VBPFs/VBSFs [64], further investigations are need to cope with the other problems.

4.4. Other VDFs for adaptive filtering

In addition to the ANFs and higher-order adaptive band-pass/band-stop filtering, many applications of other VDFs to adaptive filtering have been presented. In [65], adaptive filtering based on the cascade connection of second-order all-pass filters is proposed. This method is shown to be superior to the standard ANF-based methods for the detection of multiple sinusoids. Another

approach for the detection of multiple sinusoids is also proposed in [66–68], where comb filters with variable bandwidth and variable notch gain are applied to adaptive filtering.

Furthermore, adaptive filtering based on VLPFs can be found in the literature [69]. It should be noted that, in general, realization of adaptive low-pass filtering is much more difficult than adaptive notch filtering or adaptive band-pass/band-stop filtering. The reason of this lies in the difficulty in the problem setup that can describe a unimodal cost function. However in the work of [69], a unimodal cost function is successfully obtained by considering the detection of passband-edge frequency of a low-pass filtered signal and using the approach of weighted cost function.

5. Conclusion

This chapter has reviewed recent research activities on VDFs with focus on the approximation problem, the realization problem, and the applications to adaptive filtering. Since this chapter has paid attention to 1-D VDFs with variable magnitude responses, the introduction of other types of VDFs such as M-D VDFs and variable fractional-delay filters has been omitted. For a similar reason, VDF applications other than adaptive filtering have also been omitted. Although VDFs have been studied for a long time, many elegant results are still being proposed, and hence the research on VDFs will continue to be an active area of investigation.

Author details

Shunsuke Koshita*, Masahide Abe and Masayuki Kawamata

*Address all correspondence to: kosita@mk.ecei.tohoku.ac.jp

Department of Electronic Engineering, Graduate School of Engineering, Tohoku University, Sendai, Japan

References

[1] Stoyanov G, Kawamata M. Variable digital filters. Journal of Signal Processing. July 1997; **1**(4):275-289

[2] Roberts RA, Mullis CT. Digital Signal Processing. Boston, USA: Addison-Wesley; 1987

[3] Constantinides AG. Spectral transformations for digital filters. IEE Proceedings. Aug. 1970;**117**(8):1585-1590

[4] Mitra SK, Neuvo Y, Roivainen H. Design of recursive digital filters with variable characteristics. International Journal of Circuit Theory and Applications. 1990;**18**:107-119

[5] Murakoshi N, Watanabe E, Nishihara A. A synthesis of variable IIR digital filters. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences. Mar. 1992;**E75-A**(3):362-368

[6] Matsukawa H, Kawamata M. Design of variable digital filters based on state-space realizations. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences. Aug. 2001;**E84-A**(8):1822-1830

[7] Jang H-J, Kawamata M. Realization of high accuracy 2-D variable IIR digital filters. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences. Oct. 2002;**E85-A**(10):2293-2301

[8] Watanabe E, Ito M, Murakoshi N, Nishihara A. A synthesis of variable wave digital filters. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences. Jan. 1994;**E77-A**(1):263-271

[9] Koshita S, Abe M, Kawamata M. Variable state-space digital filters using series approximations. Digital Signal Processing. Jan. 2017;**60**:338-349

[10] Darak SJ, Prasad VA, Lai EM-K. Efficient implementation of reconfigurable warped digital filters with variable low-pass, high-pass, bandpass, and bandstop responses. IEEE Transactions on Very Large Scale Integration (VLSI) Systems. June 2013;**21**(6):1165-1169

[11] Zarour R, Fahmy MM. A design technique for variable digital filters. IEEE Transactions on Circuits and Systems. Nov. 1989;**36**(11):1473-1478

[12] Deng T-B. Design of recursive 1-D variable filters with guaranteed stability. IEEE Transactions on Circuits and Systems II. Sept. 1997;**44**(9):689-695

[13] Pun CKS, Chan SC, Yeung KS, Ho KL. On the design and implementation of FIR and IIR digital filters with variable frequency characteristics. IEEE Transactions on Circuits and Systems II. Nov. 2002;**49**(11):689-703

[14] Chan SC, Pun CKS, Ho KL. A new method for designing FIR filters with variable characteristics. IEEE Signal Processing Letters. Feb. 2004;**11**(2):274-277

[15] Johansson H, Löwenborg P. On linear-phase FIR filters with variable bandwidth. IEEE Transactions on Circuits and Systems II. Apr. 2004;**51**(4):181-184

[16] Deng T-B. Closed-form design and efficient implementation of variable digital filters with simultaneously tunable magnitude and fractional delay. IEEE Transactions on Signal Processing. June 2004;**52**(6):1668-1681

[17] Tsui KM, Chan SC. Design of complex-valued variable FIR digital filters and its application to the realization of arbitrary sampling rate conversion for complex signals. IEEE Transactions on Circuits and Systems II. July 2005;**52**(7):424-428

[18] Dam HH, Cantoni A, Teo KL, Nordholm S. Variable digital filter with least-square criterion and peak gain constraints. IEEE Transactions on Circuits and Systems II. Jan. 2007; **54**(1):24-28

[19] Dam HH, Cantoni A, Nordholm S, Teo KL. Variable digital filter with group delay flatness specification or phase constraints. IEEE Transactions on Circuits and Systems II. May 2008;**55**(5):442-446

[20] Miyata T, Aikawa N. A design method for variable linear-phase FIR filters with changing multifactors for checkweighers. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences. Aug. 2010;**E93-A**(8):1400-1407

[21] Miyata T, Aikawa N. A design of FIR filters with variable notches considering reduction method of polynomial coefficients for real-time signal processing. International Journal of Innovative Computing, Information and Control. Sept. 2013;**9**(9):3527-3536

[22] Darak SJ, Vinod AP, Lai EM-K, Palicot J, Zhang H. Linear-phase VDF design with unabridged bandwidth control over the Nyquist band. IEEE Transactions on Circuits and Systems II. June 2014;**61**(6):428-432

[23] Koshita S, Abe M, Kawamata M. Minimum roundoff noise realization for variable IIR digital filters based on M-D polynomial approximation. In: The IEEE International Symposium on Circuits and Systems (ISCAS); May 2018

[24] Farrow CW. A continuously variable digital delay element. In: The IEEE International Symposium on Circuits and Systems (ISCAS); June 1988. pp. 2641-2645

[25] Stoyanov G, Uzunov I, Kawamata M. Design and realization of variable IIR digital filters as a cascade of identical subfilters. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences. Aug. 2001;**E84-A**(8):1831-1839

[26] Yu YJ, Lim YC, Shi D. Low-complexity design of variable bandedge linear phase FIR filters with sharp transition band. IEEE Transactions on Signal Processing. Apr. 2009; **57**(4):1328-1338

[27] Yu YJ, Xu WJ. Mixed-radix fast filter bank approach for the design of variable digital filters with simultaneously tunable bandedge and fractional delay. IEEE Transactions on Signal Processing. Jan. 2012;**60**(1):100-111

[28] Xu WJ, Yu YJ, Johansson H. Improved filter bank approach for the design of variable bandedge and fractional delay filters. IEEE Transactions on Circuits and Systems I. Mar. 2014;**61**(3):764-777

[29] Regalia PA, Mitra SK, Vaidyanathan PP. The digital all-pass filter: A versatile signal processing building block. Proceedings of the IEEE. Jan. 1988;**76**(1):19-37

[30] Chambers JA, Constantinides AG. Frequency tracking using constrained adaptive notch filters synthesised from allpass sections. IEE Proceedings, Part F. Dec. 1990;**137**(6):475-481

[31] Koshita S, Kumamoto Y, Abe M, Kawamata M. Adaptive IIR band-pass/band-stop filtering using high-order transfer function and frequency transformation. Interdisciplinary Information Sciences. Nov. 2013;**19**(2):163-172

[32] Regalia PA. Adaptive IIR Filtering in Signal Processing and Control. New York, USA: Marcel Dekker; 1995

[33] Regalia PA. An improved lattice-based adaptive IIR notch filter. IEEE Transactions on Signal Processing. Sept. 1991;**39**(9):2124-2128

[34] Regalia PA. A complex adaptive notch filter. IEEE Signal Processing Letters. Nov. 2010; **17**(11):937-940

[35] Nakamura S, Koshita S, Abe M, Kawamata M. A new adaptive notch filtering algorithm based on normalized lattice structure with improved mean update term. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences. July 2015;**E98-A** (7):1482-1493

[36] Matsuura K, Watanabe E, Nishihara A. Adaptive line enhancers on the basis of least-squares algorithm for a single sinusoid detection. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences. Aug. 1999;**E82-A**(8):1536-1543

[37] Kawamura A, Itoh Y, Okello J, Kobayashi M, Fukui Y. Parallel composition based adaptive notch filter: Performance and analysis. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences. July 2004;**E87-A**(7):1747-1755

[38] Kawamura A, Iiguni Y, Itoh Y. An adaptive algorithm with variable step-size for parallel notch filter. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences. Feb. 2006;**E89-A**(2):511-519

[39] Sugiura Y. A fast and accurate adaptive notch filter using a monotonically increasing gradient. In: Proceedings of the European Signal Processing Conference (EUSIPCO); Sept. 2014. pp. 1756-1760

[40] Koshita S, Munakata H, Abe M, Kawamata M. Correct formulation of gradient characteristics for adaptive notch filters based on monotonically increasing gradient algorithm. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences. July 2017;**E100-A**(7):1557-1561

[41] Mvuma A, Nishimura S. Steady-state analysis of a simplified lattice-based adaptive IIR notch filter. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences. June 2000;**E83-A**(6):965-972

[42] Koshita S, Noguchi Y, Abe M, Kawamata M. Analysis of frequency estimation MSE for all-passbased adaptive IIR notch filters with normalized lattice structure. Signal Processing. Mar. 2017;**132**:85-95

[43] Sugiura Y, Kawamura A, Iiguni Y. Performance analysis of an inverse notch filter and its application to F_0 estimation. Circuits and Systems. Jan. 2013;**4**(1):117-122

[44] Nehorai A. A minimal parameter adaptive notch filter with constrained poles and zeros. IEEE Transactions on Acoustics, Speech, and Signal Processing. Aug. 1985;**33**(4):983-996

[45] Pei S-C, Tseng C-C. Adaptive IIR notch filter based on least mean *p*-power error criterion. IEEE Transactions on Circuits and Systems II. Aug. 1993;**40**(8):525-529

[46] Xiao Y, Takeshita Y, Shida K. Steady-state analysis of a plain gradient algorithm for a second-order adaptive IIR notch filter with constrained poles and zeros. IEEE Transactions on Circuits and Systems II. July 2001;**48**(7):733-740

[47] Xiao Y, Takeshita Y, Shida K. Tracking properties of a gradient-based second-order adaptive IIR notch filter with constrained poles and zeros. IEEE Transactions on Signal Processing. Apr. 2002;**50**(4):878-888

[48] Xiao Y, Ma L, Khorasani K, Ikuta A. Statistical performance of the memoryless nonlinear gradient algorithm for the constrained adaptive IIR notch filter. IEEE Transactions on Circuits and Systems I. Aug. 2005;**52**(8):1691-1702

[49] Lim YC, Zou YX, Zheng N. A piloted adaptive notch filter. IEEE Transactions on Signal Processing. Apr. 2005;**53**(4):1310-1323

[50] Punchalard R. Mean square error analysis of unbiased modified plain gradient algorithm for second-order adaptive IIR notch filter. Signal Processing. Nov. 2012;**92**(11):2815-2820

[51] Punchalard R. A modified inverse tangent based adaptive algorithm for a second-order constrained adaptive IIR notch filter. Signal Processing. Jan. 2014;**94**:350-358

[52] Munakata H, Koshita S, Abe M, Kawamata M. Performance comparison of adaptive notch filters with respect to the output power due to the white noise input. In: Proceedings of the International Workshop on Smart Info-Media Systems in Asia (SISA); Aug. 2015. pp. 30-34

[53] Cho NI, Choi C-H, Lee SU. Adaptive line enhancement by using an IIR lattice notch filter. IEEE Transactions on Signal Processing. Apr. 1989;**37**(4):585-589

[54] Cho NI, Lee SU. On the adaptive lattice notch filter for the detection of sinusoids. IEEE Transactions on Circuits and Systems II. July 1993;**40**(7):405-416

[55] Martin KW, Sun MT. Adaptive filters suitable for real-time spectral analysis. IEEE Transactions on Circuits and Systems. Feb. 1986;**33**(2):218-229

[56] Kwan T, Martin K. Adaptive detection and enhancement of multiple sinusoids using a cascade IIR filter. IEEE Transactions on Circuits and Systems. July 1989;**36**(7):937-947

[57] Petraglia MR, Shynk JJ, Mitra SK. Stability bounds and steady-state coefficient variance for a second-order adaptive IIR notch filter. IEEE Transactions on Signal Processing. July 1994;**42**(7):1841-1845

[58] Xiao Y, Tadokoro Y, Kobayashi Y. A new memoryless nonlinear gradient algorithm for a second-order adaptive IIR notch filter and its performance analysis. IEEE Transactions on Circuits and Systems II. Apr. 1998;**45**(4):462-472

[59] DeBrunner V, Torres S. Multiple fully adaptive notch filter design based on allpass sections. IEEE Signal Processing Letters. Feb. 2000;**48**(2):550-552

[60] Raja Kumar RV, Pal RN. A gradient algorithm for the center-frequency adaptive filters. Proceedings of the IEEE. Feb. 1985;**73**(2):371-372

[61] Raja Kumar RV, Pal RN. Recursive center-frequency adaptive filters for the enhancement of bandpass signals. IEEE Transactions on Acoustics, Speech, and Signal Processing. June 1986;**34**(3):633-637

[62] Raja Kumar RV, Pal RN. Performance analysis of the recursive center-frequency adaptive bandpass filters. Signal Processing. June 1989;**17**(2):105-118

[63] Raja Kumar RV, Pal RN. Tracking of bandpass signals using center-frequency adaptive filters. IEEE Transactions on Acoustics, Speech, and Signal Processing. Oct. 1990;**38**(10): 1710-1721

[64] Koshita S, Miyoshi K, Abe M, Kawamata M. High-performance variable band-pass/band-stop state-space digital filters using Gramian-preserving frequency transformation. Digital Signal Processing. Apr. 2014;**27**:175-184

[65] Cousseau JE, Werner S, Donate PD. Factorized all-pass based IIR adaptive notch filters. IEEE Transactions on Signal Processing. Nov. 2007;**55**(11):5225-5236

[66] Sugiura Y, Kawamura A, Iiguni Y. A comb filter design method using linear phase FIR filter. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences. Aug. 2012;**E95-A**(8):1310-1316

[67] Sugiura Y, Kawamura A, Iiguni Y. An adaptive comb filter with flexible notch gain. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences. Nov. 2012;**E95-A**(11):2046-2048

[68] Sugiura Y, Kawamura A, Iiguni Y. A comb filter with adaptive notch gain and bandwidth. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences. Apr. 2013;**E96-A**(4):790-795

[69] Yamaguchi K, Watanabe E, Nishihara A. Adaptive lowpass filters. In: Proceedings of IEEE Asia-Pacific Conference on Circuits and Systems (APCCAS); Dec. 2004. pp. 510-513

Section 2

Digital Systems Hardware and Protection

Chapter 3

Electrostatic Discharge Protection and Latch-Up Design and Methodologies for ASIC Development

Steven H. Voldman

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.81033

Abstract

Electrostatic discharge (ESD) has been an issue in devices, circuits, and systems for electronics for many decades, as early as the 1970s, and continued to be an issue until today. In this chapter, the issue of ESD protection design and methods for Application-Specific Integrated Circuits (ASICs) will be discussed. The chapter will discuss ESD design in an ASIC environment. The discussion will address ESD design layout, design rules and practices, and the method of integration of ESD protection into the ASIC design practice. Part of the methodology is the floor planning of an ASIC design, I/O library, integration of ESD into I/O cells, power distribution, and placement of power pads, in both array and peripheral design methodologies. As part of the ASIC I/O design, guard rings and latch-up interactions will be highlighted.

Keywords: electrostatic discharge, latch-up, ASICs, ASIC I/O integration, ASIC power distribution

1. Introduction

Electrostatic discharge (ESD) design, practices, and methods are a fundamental to the implementation of an ASIC design environment [1–28]. The integration of ESD and latch-up in an ASIC environment is typically a top-down design flow. In the ASIC environment, the chip size, the number of I/O circuits, the bus location, placement of the I/O cells, and integration of the ESD elements and power are all synthesized. The top-down design flow is as follows:

- Functional definition
- Technology decision

- I/O definition with ESD network and guard ring definition
- Power pad definition with ESD power clamp and guard ring definition
- Core function placement
- Number of I/O
- I/O placement
- Power pads and number of ESD power clamps required
- Core to core ESD networks
- Core to core guard rings
- I/O to core guard rings

2. Electrostatic discharge

Electrostatic discharge (ESD) is a common form of component-level failure from manufacturing, shipping, and handling in an ASIC environment [1–8]. Two of the tests used in the qualification and release process of an ASIC design system are the human body model (HBM) and the charged-device model (CDM) standards [1–8].

2.1. Human body model

The human body model (HBM) is the most widely established standard for the qualification and release of semiconductor components in the semiconductor industry [1–4, 6–8]. The HBM test is integrated into the qualification and release process of the quality and reliability teams for components in ASIC organizations, corporations, and foundries. The model was intended to represent the interaction of the electrical discharge of a human being, who is charged, with a component or an object. The charged source then touches a component or an object using a finger. The physical contact between the charged human being and the component or object allows for current transfer between the human being and the object.

HBM failure mechanisms typically are associated with failures on the ASIC peripheral circuitry of a semiconductor chip that are connected to signal pins. HBM ESD networks are used to establish an alternative current path to avoid failure of the ASIC peripheral circuitry. HBM failures can also occur on the power rails due to inadequate bus widths and ESD power clamps between the power rails. HBM failures can occur in both passive and active semiconductor components. The failure signature is typically isolated to a single device or a few elements in a given current path where the current exceeded the power to failure of the circuit elements. ESD circuits are designed to be responsive to HBM pulse widths; specifically, the RC-triggered ESD power clamp is a vulnerable ESD circuit.

An example of an ESD protection network is known as a dual-diode network [1–4, 7, 8]. The dual-diode ESD network is a commonly used network for complementary metal-oxide

http://dx.doi.org/10.5772/intechopen.81033

semiconductor (CMOS) technology. A first p-n diode element is formed in an n-well region where the p-anode is the p-diffusion implant of the p-channel MOSFET device and the n-cathode is the n-well region connected to the power supply V_{DD}. This is sometimes referred to as the "up diode." A second p-n diode element is formed in a p-well or p-substrate region where the n-cathode is the n-diffusion implant of the n-channel MOSFET device or the n+/n-well implant, and the p-anode is the p-well region or p-substrate region connected to the power supply V_{SS}. This is sometimes referred to as the "down diode." This circuit provides a "forward bias" ESD protection solution for positive and negative ESD pulse events to the two power rails V_{DD} and V_{SS}. An advantage of the dual-diode ESD network for ASIC environments is that it is easy to migrate from technology generation to technology generation and is scalable.

In an ASIC design methodology, the ESD network is integrated within the I/O library element. The I/O cell can contain a bond pad, guard rings, ESD network, receiver, and off-chip driver (OCD) elements [2, 3, 7, 8].

2.2. Charged-device model

The charged-device model (CDM) is an electrostatic discharge (ESD) test method that is part of the qualification of semiconductor components in an ASIC design system [6]. The CDM event is associated with the charging of the semiconductor component through different charging processes. Charging of the package can be achieved through direct contact charging or field-induced charging processes. The field-induced charging method is called the field-induced charged-device model (FI-CDM).

Charged-device ESD solutions utilize an additional circuit element local to the receiver network. For CDM protection, an additional resistor and second dual diode are added, where the second stage element is adjacent to the MOSFET receiver. The purpose of the second stage element is to divert the electric charge in the substrate adjacent to the MOSFET receiver to the bond pad without destruction of the receiver dielectric and circuitry.

2.2.1. CDM and long-narrow ASIC I/O

In an ASIC environment, each generation attempts to squeeze in as many I/O circuits on the periphery, by reducing the width of the I/O cell, and compensate by increasing the height of the ASIC I/O cell [11–15]. With the long-narrow ASIC I/O cell, the receiver network is moved farther away from the bond pad and the first stage of the ESD network. As a result, the second dual-diode stage is even more necessary to achieve excellent CDM ESD results.

3. ASIC requirements

In an ASIC environment, there are rules and requirements that are established in the design methodology. These rules and requirements for ESD design, latch-up design layout, to application issues of placement of power, placement of grounds. Additionally, the power sequencing for power down and power up is specified in the methodology. These fundamental ASIC rules have

a significant influence on the ESD circuits, ESD design methodology, and ESD circuit placement. Additionally, the ASIC system must achieve latch-up specification objectives.

3.1. ESD protection-level requirements

In a release of an ASIC system, there are qualification expectations of ESD protection levels. Each ASIC I/O cell is tested for ESD and CDM test processes, and the entire I/O library is to achieve above the desired protection levels for qualification. In the past, the desired protection levels for HBM and CDM were >4000 V HBM and > 500 V CDM; with technology scaling, these objectives have been changed to lower levels.

3.1.1. ESD design rules

ESD design rules of the physical dimensions of the ESD networks are typically contained within the technology design manual. The ASIC library is required to fulfill the technology ESD design manual rule set. The ESD design rules provide the circuit, layout, and physical dimensions.

3.2. Latch-up requirements

Latch-up requirements are also needed for the qualification of an ASIC design system [5]. The latch-up requirements used in all corporations and foundries are in the JEDEC latch-up specification and test method.

3.2.1. Latch-up design rules

As ASIC systems became more complex with integration of system on chips (SOC), the number of latch-up design rules has increased. Historically, latch-up rules consisted of four rules—(1) the distance between a PFET and its corresponding n-well contact, (2) the distance between an NFET and its closest substrate contact, (3) spacing of PFET to n-well edge, and lastly (4) spacing of NFET to n-well edge [5]. With scaling, there were many additional rules established between ESD and I/O, I/O to I/O, PFET to core logic, and NFET to core logic. With complexity, more guard rings were added to isolate the different regions of an ASIC implementation.

3.3. ASIC application requirements

In the definition of an ASIC system, there are many application rules and requirements that are established. These include area requirements, power distribution, and power sequencing.

3.3.1. Area requirements

In an ASIC system, there is a given chip area specified for the I/O circuitry [9–15]. This is planned as a certain percentage of the total chip area. Additionally, the ESD networks also can only be a certain defined percentage of the I/O cell. Typically, the ESD area desired is less than 20–25% of the total I/O cell area.

http://dx.doi.org/10.5772/intechopen.81033

3.3.2. Power-up: sequence dependent

ASIC system can define the sequencing requirements for the power and the signal pins. Some ASIC systems can have a sequence-dependent power up and shutdown. In these systems, the ESD networks are not to be "on" during power up or power down.

3.3.3. Power-up: sequence independent and hot plugging

ASIC system can define the sequencing requirements for the power and the signal pins. Some ASIC systems require sequence-independent power up and shutdown. In these systems, the ESD networks are not to be "on" during power-up or power down. In this case, the ESD networks must not be forward biased in power up or power down. As a result, new sequence-independent ESD networks that do not forward bias were required. A sequence-independent ESD network was implemented into a 0.5-um ASIC system with significant success with a floating well control network [2, 3, 9].

3.3.4. Power distribution and placement requirements

ASIC methodologies establish requirements for the frequency of placement of "power cells" and "ground cells" to support the I/O and core circuitry. For example, some corporations stated that their ASIC system must be a power cell for every fourth or fifth I/O location. This provided a significant opportunity for ESD protection, since an "ESD power clamp" can be placed in the area allocating for the VDD and VSS power pins. As the frequency of placement of the ESD power clamps increases, the series resistance loss of the power bus or ground bus decreases; this allows for a lower resistance path for the current to flow through the complete network, providing improved ESD robustness [2, 3, 7, 8].

3.3.5. Frequency bandwidth

As ASIC technology transitions to advanced technology nodes, the application frequency is increased. From an ESD perspective, the expectation from ASICs is that the capacitance loading of the ESD network must be reduced to not impact the frequency bandwidth of the I/O networks. This can be achieved through semiconductor process modification, layout and design, and reduction of the size of the ESD networks. Through ESD novelty and innovation, the frequency bandwidth of signal inputs can be realized without reduction of ESD reliability concern.

3.3.6. Input leakage and IDD limitations

An additional concern is the input leakage requirements and IDD limitations. Input leakage can be minimized through proper design of ESD networks through process, circuit topology, and layout innovations. A larger concern is the ESD power clamps on the VDD power supply. It is critical to limit the number of ESD power clamps to not impact the IDD leakage limit for the application.

4. ASIC I/O

In ASIC I/O design, ESD protection is integral in the definition and methodology. In the following section, this will be discussed.

4.1. I/O and ESD integration

In the definition of an ASIC I/O, the off-chip driver (OCD), the receiver, and ESD circuitry are co-designed to integrate the networks into a common physical space [7]. This requires planning in the methodology to allocate the right percentage of area for each of these elements. Different methodologies are used depending on the foundry or corporation.

4.2. I/O and ESD design integration and synthesis

In one methodology, the ASIC system supported different off-chip driver (OCD) sizes by adjusting the number of MOSFET fingers that were connected. The number of MOSFET fingers used was the maximum driver strength for the I/O library providing different impedance as well. There are many options on what can be done in this methodology.

For example, there can be a 20, 30, and 50 Ω impedance OCD circuit offering, by attaching a different number of MOSFET fingers in a given I/O cell. In one method, the residual MOSFET OCD fingers were grounded and used as a "grounded gate NMOS" ESD network. In this case, the residual MOSFET fingers act as a natural ESD protection and utilize the "unused" section of the OCD. The advantage of this method is that the unused portion of the I/O is utilized. In a second embodiment, instead of grounding the residual MOSFET fingers, a dummy inverter load was attached to keep the residual fingers "off" or in a low logic state. Using a dummy inverter, the MOSFET gates that are not grounded, and "turn-on" from MOSFET snapback at the same impedance state as the MOSFET OCD. In this fashion, both the active and residual elements work together for ESD protection [7, 8].

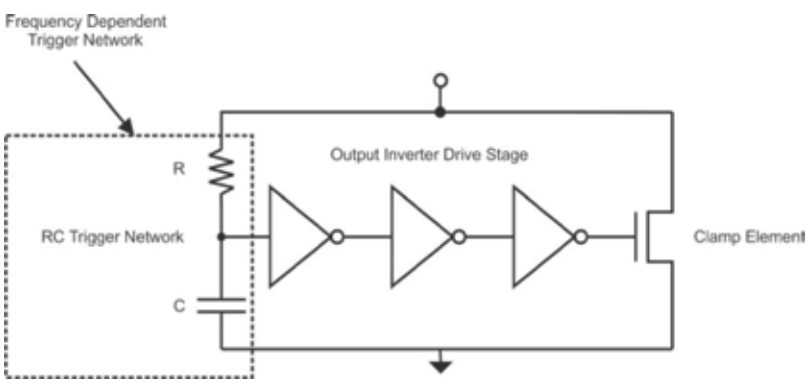

Figure 1. RC-triggered ESD power clamp.

http://dx.doi.org/10.5772/intechopen.81033

4.3. I/O and ESD power clamp integration

ASIC methodologies establish requirements for the frequency of placement of "power cells" and "ground cells" to support the I/O and core circuitry [2–4, 7, 8, 11–15]. This provided a perfect opportunity for ESD protection, since an "ESD power clamp" can be placed in the area allocating for the power pins, the VDD pin location (**Figure 1**).

At one time, prior to the invention of ESD power clamps, it was just as empty areas; it was realized that the ESD power clamps can be placed in these regions. As the frequency of placement of the ESD power clamps increases, the series resistance loss of the power bus or ground bus decreases; this allows for a lower resistance path for the current to flow through the complete network, providing improved ESD robustness.

4.4. Ground-to-ground ESD networks

ESD networks are required between ground power rails for every independent domain. In an ASIC system, analog and digital circuits are in separate power domains [7, 8, 11–18]. These domains must be interconnected through ground-to-ground ESD networks. These networks must be bidirectional to allow current to flow in both directions. These networks do not have to be symmetric (e.g., m diodes in one direction and n diodes in the opposite direction). These ground-to-ground ESD networks can be placed in the VSS pin location.

4.5. Master/slave ESD systems

In some ASIC embodiments, the ESD power clamps are integrated across the entire system. In this type of system, there is one "master," which triggers the set of ESD power clamps on all in parallel (**Figure 2**) [2, 3, 7, 8]. A "master ESD power clamp" contains the trigger network that then sends a signal to turn on all the "slave ESD power clamps." This system has the advantage of turning on all ESD power clamps in parallel across the entire ASIC system, significantly lowering the "on-resistance" of the single ESD power clamp. The disadvantage of this system is an ESD signal bus (from the master clamp to the slave clamps) must be distributed with the ASIC power busses around the complete chip (**Figure 3**).

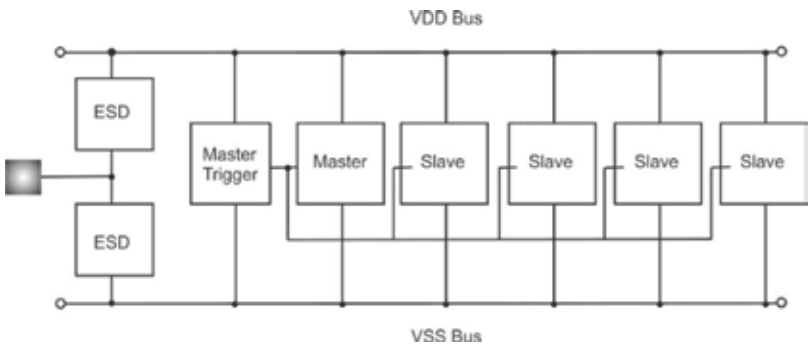

Figure 2. Master/slave ESD system.

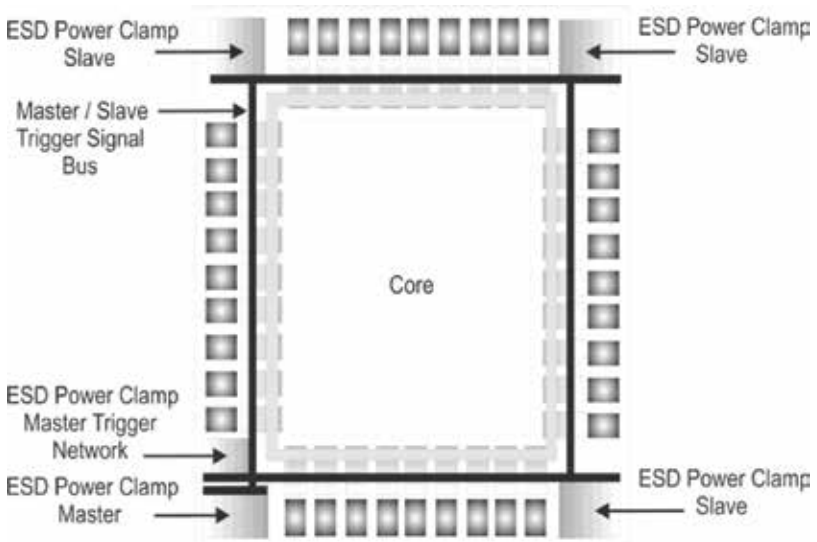

Figure 3. Master/slave ESD system floor plan.

5. ASIC I/O and latch-up: guard ring integration

In ASIC design methodologies, the I/O must address both ESD and latch-up [11–15]. Latch-up of the ASIC I/O can occur with improper design of the outer guard rings and internal guard rings in a given ASIC I/O cell.

5.1. I/O outer guard ring

In some ASIC methodologies, a guard ring is placed around the entire I/O cell region. This leads to a natural "frame" for the circuit. It serves as a source to collect internal current injected from inside the I/O circuit or external current being injected from outside the I/O circuit. In other methodologies, this outer guard ring is overlapped or integrated with the adjacent I/O cell; this is not detrimental and saves space [7, 8].

Additionally, even further methodologies, there is no "ring" around the entire circuit but only on the top and bottom. This can lead to I/O to I/O interaction, which is not desirable.

5.2. I/O circuit to adjacent I/O circuit

In ASIC I/O design, ESD and latch-up problems can occur when the design is "fully populated" versus "partially populated." In digital applications, typically the design is fully populated and is "I/O limited." But, in some analog applications, in core-dominated designs, the core establishes the chip size (e.g., core-dominated design), leading to a partially populated periphery.

http://dx.doi.org/10.5772/intechopen.81033

5.2.1. Fully populated I/O periphery

In the case of a chip perimeter that is fully populated by I/O cells, concerns about latch-up events between adjacent I/O can be evaluated and tested (**Figure 4**). In this case, guard rings can be placed between the I/O cells, and latch-up interaction can be quantified [7, 8].

5.2.2. Partially populated I/O periphery

In the case of a chip perimeter that is not fully populated, concerns about latch-up events between I/O and adjacent cells can occur. In many ASIC systems, decoupling capacitors can be placed adjacent to an I/O cell. An n-well decoupling capacitor has a large n-well region that can serve as a cathode to a lateral pnpn formed by an adjacent PFET to form a pnpn. In this low populated system, new latch-up rules are needed to avoid CMOS latch-up in ASIC environments without full populated I/O cells.

5.2.3. Adjacency latch-up rules

In some design methodologies, it is necessary to verify what the adjacent circuit is [11–15]. In a fully populated I/O ring, the adjacent circuit will be another I/O cell. In this case, it is well understood. But, in cases where it is not populated, additional rules may be required to avoid latch-up between adjacent circuits. The most common failure was latch-up between an I/O cell and a decoupling capacitor [7–8].

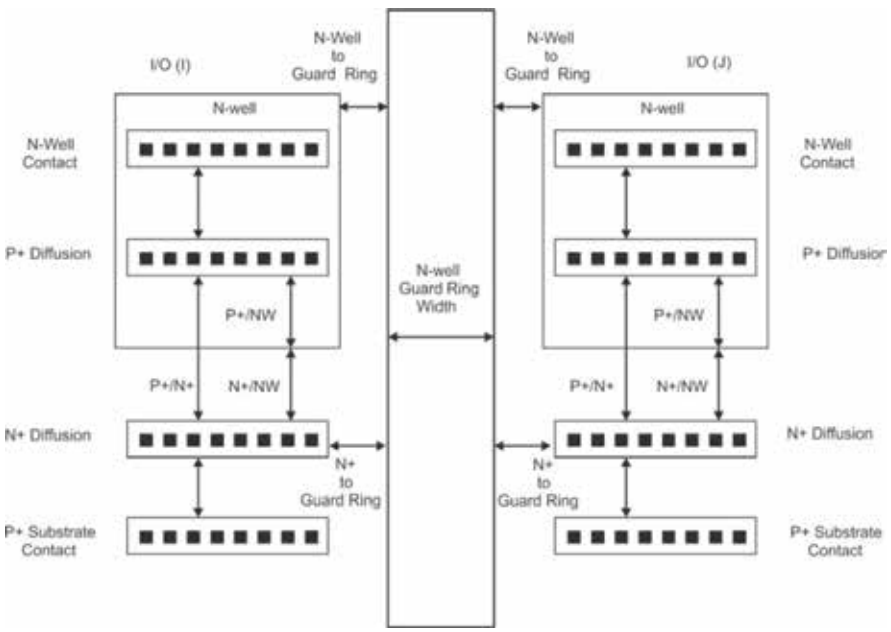

Figure 4. I/O to I/O latch-up test structure.

5.3. ESD to I/O circuit

Proper isolation between the ESD circuit contained within the I/O cell and the I/O circuit itself is necessary to avoid CMOS latch-up. The I/O circuit may contain an NFET pull-down, a PFET pull-up, and ballasting resistor bank. This is achievable by proper guard rings around the ESD network and the choice of what elements are placed adjacent to the ESD device. Note that there are multiple elements in an I/O circuit, and the choice of what is placed adjacent to the ESD network has to be co-synthesized with the power bus placement for the I/O cell (e.g., bond pad, VDD, VSS, AVDD, and AVSS). Since the ESD network typically has an element for positive and negative polarity pulse events, both the VDD and the VSS must be local to the ESD network and likewise for the PFET and NFET OCD elements.

For negative polarity ESD events, the n-diffusion or n-well resistor banks will be in parallel with the ESD elements, and the adjacency to their respective guard rings is key to provide "current sharing" and avoid "current robbing" of the ESD event. It was found by matching the space between n-type elements, and the guard rings provided maximum ESD results.

5.4. I/O to core circuitry

To avoid interaction between the I/O circuitry and the core circuitry, additional guard rings have been placed to isolate the I/O from the core circuits. The core circuits are very sensitive to CMOS latch-up since they have no guard rings surrounding the MOSFETs. To avoid latch-up issues between the I/O and core circuitry, additional requirements are established [5]:

- Latch-up space design rule between PFET OCD circuit and core circuits
- Latch-up space design rule between NFET OCD circuit and core circuits
- N+ guard ring of specified width between I/O region and core circuitry
- P+ substrate guard ring of specified width between I/O region and core circuitry

5.5. Core-to-core circuitry

Latch-up can occur between different core regions. This occurs when cores are placed adjacent to each other, where there is no design rule check, and when there is no history of the cores being adjacent in prior designs. An example is between a PFET-dominated core and a bank of decoupling capacitors with an n-well plate (**Figure 5**) [7, 8].

To avoid latch-up interaction between the circuitry of cores, additional guard rings, moats, substrate contacts, and space can be added in the design (**Figure 6**) [7, 8].

5.6. Digital, analog, and RF core circuitry

Latch-up and noise can occur between digital, analog, and radio frequency (RF) cores placed on the same substrate in an ASIC design. This occurs when cores are placed adjacent to each other, where there is no design rule check, and when there is no history of the cores being

http://dx.doi.org/10.5772/intechopen.81033

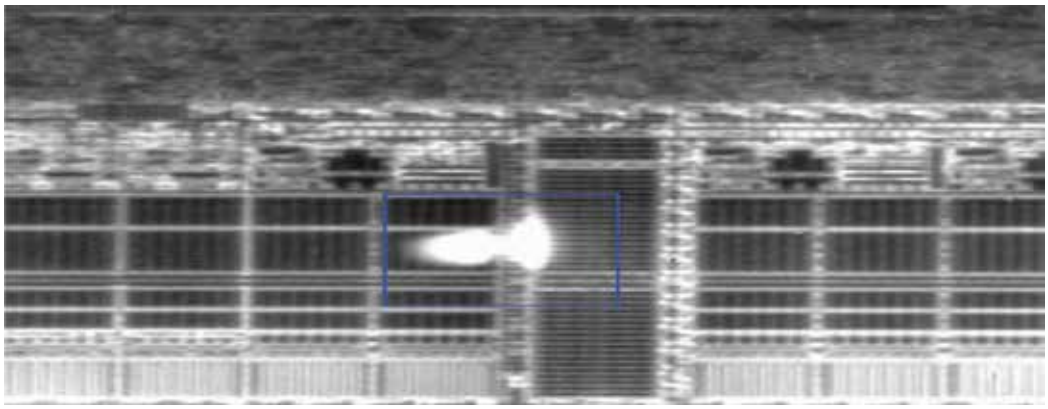

Figure 5. Latch-up between peripheral I/O and adjacent decoupling capacitors.

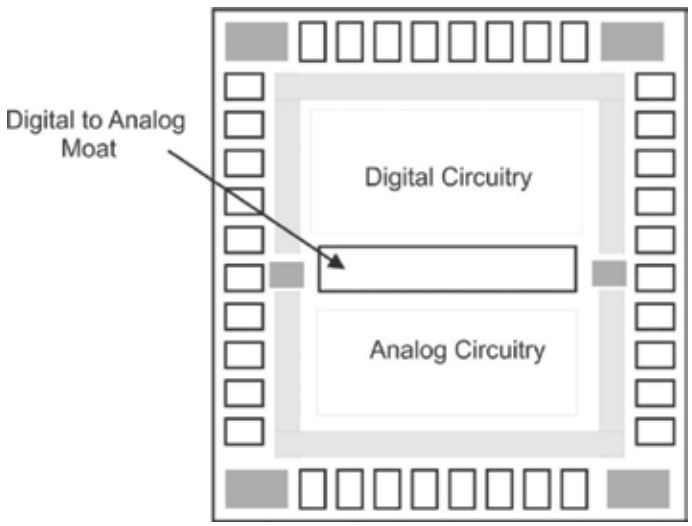

Figure 6. Digital to analog integration with moa.

placed close together. To avoid latch-up and noise interaction between digital and analog circuits, large guard rings, moats, substrate contacts, and spaces are added. The space between the digital and analog cores can be significant and as large as 40 to 100 um. In these designs, the power grids and grounds are also separated and spatially isolated [7, 8].

5.7. Internal ESD networks: digital to analog signals

In mixed signal (MS) and system-on-chip (SOC) ASIC designs, there are signal lines that transfer from the digital core to the analog core. The power grids and ground planes are physically isolated to improve noise isolation. Digital core driver circuits transmit signals to analog core receiver networks in the analog section of the semiconductor chip. With the

separation of the digital and analog grounds, and physical space separation, overvoltage of the analog receiver can occur due to the voltage drops in the signal line and the ground connections (**Figure 7**).

One of the solutions is to place an "internal ESD network" on the internal signal line between the digital driver circuit and the analog receiver. This can be achieved by adding a resistor to the signal line and a grounded gate NMOS prior to the analog receiver (**Figure 8**).

A second solution is to place "third part" inverter stages and ground-to-ground connections between the digital and analog cores. This methodology has less performance or signal impact and is a more migratable solution (**Figure 9**) [7–8].

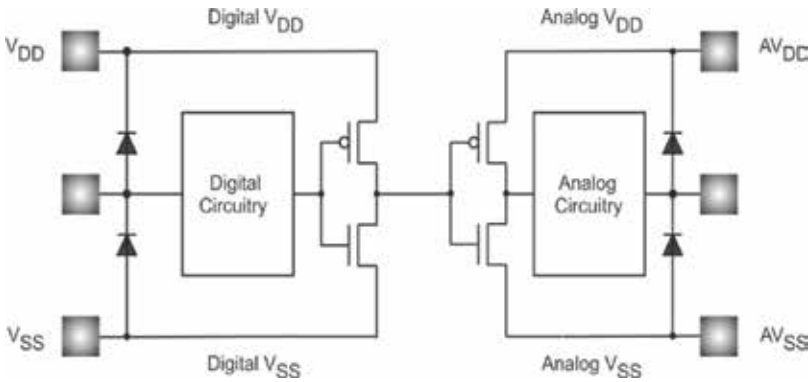

Figure 7. Digital to analog core internal signal lines requiring internal ESD.

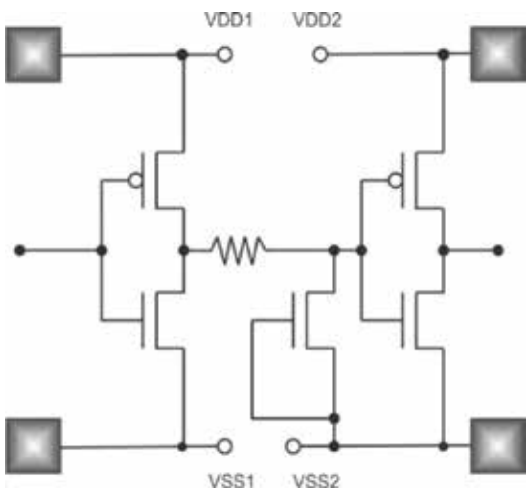

Figure 8. Internal ESD networks between cores.

http://dx.doi.org/10.5772/intechopen.81033

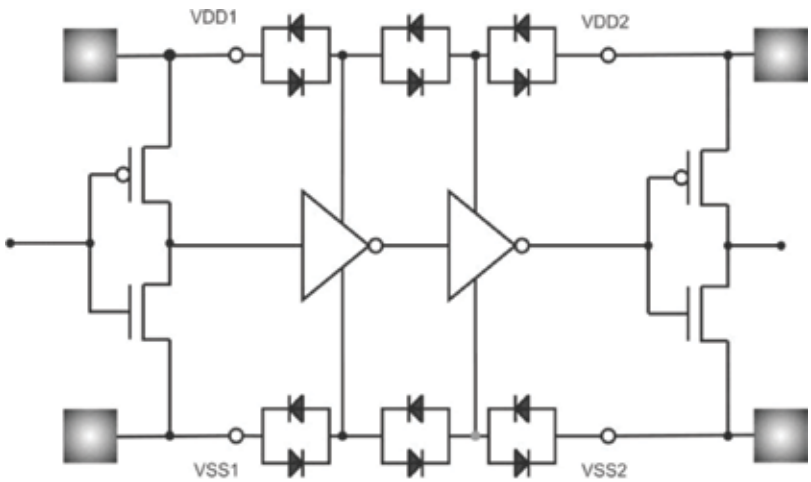

Figure 9. Third-party circuits between digital and analog cores.

6. Array I/O versus peripheral I/O architectures

In high pin count environments, the I/O networks can be distributed with the core of a semiconductor chip instead of the periphery. This is referred to as "array I/O" where the I/O cells are placed in an array fashion throughout the core. This has a large advantage for wiring, and performance, but alters the ESD and latch-up methods and needs.

6.1. Array I/O and ESD

In an array I/O environment, the I/O and ESD are co-integrated into the same I/O cell within the semiconductor chip and ASIC core. A significant change in the ESD results is that the ESD failure distribution is dependent on the wire width from the bond pad to the I/O cell. In this fashion, a "transfer wire" extends from the bond pad to the I/O cell. The ESD failure mechanism can be the wiring itself and may limit the robustness of the system. The wiring choices on how to get from the bond pad to the ESD network are also key to the robustness of the ASIC system. The ESD robustness of the system can be "wiring limited."

6.2. Array I/O and latch-up

A key issue in an array I/O environment is the onset of latch-up [7, 8]. All the circuitry surrounding the I/O cell is core circuitry with no guard rings placed around them. Latch-up can occur from the injection of carriers into the substrate from the ESD network into the surrounding circuitry (**Figure 10**). Solutions to avoid this issue are as follows:

- Wider guard rings around the I/O cell to capture carriers

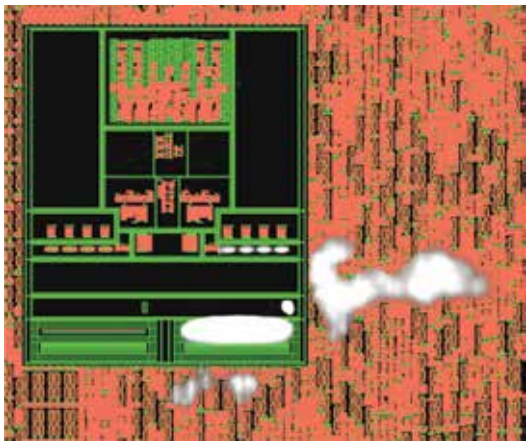

Figure 10. An example of array I/O and latch-up propagation in the core circuitry.

- Different guard ring structures with higher efficiency of carrier capture (e.g., trench, moats, etc.)
- Decreasing the n-well and p-well contact placement in the core circuitry adjacent to the I/O cell

7. Advanced technology nodes

Electrostatic discharge (ESD) will be an issue in ASIC technology as we evolve to new technologies, new devices, and 2.5D and 3D systems. These transitions will have a significant effect on ESD protection in the future.

7.1. 2.5D and 3D ASIC systems

In today's applications, migration to 2.5D and 3D systems has begun. In 2.5D applications, there are a significant number of wire bonds that interconnect the stacked chips. This has implications to electrical overstress (EOS) and wire-bond reliability. In 3D applications, the introduction of through-silicon via (TSV) technology changes the interrelation of how current flows through the multi-chip design; this has an influence on power grid design, placement of the TSV structure, and ESD devices. ESD design may require co-design with the power and ground placement of the multi-chip ASIC system.

7.2. Silicon on insulator (SOI)

Silicon on insulator (SOI) has been a mainstream technology since 2000 [1–3]. Microprocessors have been designed with excellent ESD protection levels in partially depleted SOI technology. From an ESD perspective, new SOI ESD structures were integrated, as well as addressing new SOI failure mechanisms, since these structures behaved differently than bulk ESD elements [29, 30].

7.3. FinFET

Presently, the FinFET device is being integrated into advanced sub-25 nm technologies [31–33]. With the FinFET structure, the layout and guard ring strategy will be influenced leading to different ESD layouts and designs. In the future, the direction may include both bulk and SOI FinFETs, which will respond differently for ESD and for latch-up. With the scaling of the FinFET structure, the shallow trench isolation (STI) will be scaled leading to higher parasitic bipolar current gain in the FinFET technology [33].

8. Closing comments and summary

Electrostatic discharge (ESD) has been a crucial issue in ASIC design flow and release and will continue to be an issue as semiconductor devices are scaled below 20 nm in both future and present-day nanotechnology era. As technologies migrate to sub-25 nm technology, ESD, latch-up, and EOS will be an issue for both bulk and SOI FinFET technologies.

Acknowledgements

I would like to thank the IBM ASIC I/O development team for the many years of collaboration and integration of ESD and latch-up solution into four generations of ASIC design methods. Special thanks to team lead Douglas Stout, James Pequignot, and Jeffrey Sloan. I would also like to thank the IBM Cadence™ software development of Susan Strang, Don Jordan, and C.N. Perez for implementing the ESD hierarchical parametrized cell system.

Author details

Steven H. Voldman

Address all correspondence to: voldman@ieee.org

IEEE Fellow, United States of America

References

[1] Voldman S. ESD: Physics and Devices. Chichester: Wiley; 2004

[2] Voldman S. ESD: Circuits and Devices. Chichester: Wiley; 2005

[3] Voldman S. ESD: Circuits and Devices. 2nd ed. Wiley; 2015

[4] Dabral S, Maloney TJ. Basic ESD and I/O Design. West Sussex: Wiley; 1998

[5] Voldman S. Latchup. Chichester: Wiley; 2006

[6] Voldman S. ESD Testing: From Components to Systems. Chichester: Wiley; 2017

[7] Voldman S. ESD: Design and Synthesis. Chichester: Wiley; 2011

[8] Voldman S. ESD: Analog Design. Chichester: Wiley; 2014

[9] Voldman S. Power sequence independent electrostatic discharge protection circuits. U.S. Patent No. 5,610,791, March 11th, 1997

[10] Panner J, Bednar T, Buffet P, Kemerer D, Stout D, Zuchowski P. The first copper ASICs: A 12M-gate technology. Proceedings of the IEEE Custom Integrated Circuits Conference (CICC). 1999:347-350

[11] Kaeslin H. Digital Integrated Circuit Design. England: Cambridge University Press; 2012

[12] Wakerly JF. Digital Design Principles and Practices. New York: Prentice-Hall; 1990

[13] Weste N, Harris D. CMOS VLSI Design. New York: Pearson; 2013

[14] Allen PE, D R H. CMOS Analog Circuit Design. OUP; 2012

[15] Eshraghian K, Weste N. Principles of VLSI Design. New York: Addison-Wesley; 1988

[16] Pequignot J, Rahman T, Sloan J, Stout D, Voldman S. Method and apparatus for providing ESD protection. U.S. Patent No. 6,157,530, December 5th, 2000

[17] Pequignot J, Rahman T, Sloan J, Stout D, Voldman S. Method for providing ESD protection for an integrated circuits. U.S. Patent No. 6,262,873, July 17th, 2001

[18] Pequignot J, Rahman T, Sloan J, Stout D, Voldman S. ASIC book to provide ESD protection on an integrated circuit. U.S. Patent No. 6,292,343, September 18th, 2001

[19] Brennan C, Sloan J, Picozzi D. CDM failure modes in 130 nm ASIC technology. In: Proceedings of the Electrical Overstress/Electrostatic Discharge (EOS/ESD) Symposium, 2004; pp. 182-186

[20] Pequignot J, Sloan J, Stout D, Voldman S. Electrostatic discharge protection networks for triple well semiconductor devices. U.S. Patent Application, July 15, 2004

[21] Voldman S, Perez C, Watson A. Guard rings: Theory, experimental quantification, and design. In: Proceedings of the Electrical Overstress/Electrostatic Discharge (EOS/ESD) Symposium, October 2005; pp. 131-140

[22] Voldman S, Perez C, Watson A. Guard rings: Structures, design methodology, integration, experimental results, and analysis for RF CMOS and RF mixed signal silicon germanium technology. Journal of Electrostatics. 2006;**64**:730-743

[23] Voldman S, Strang S, Jordan, D. A design system for auto-generation of ESD circuits. In: Proceedings of the International Cadence Users Group (ICUG), September 2002

[24] Voldman S, Strang S, Jordan D. An automated electrostatic discharge computer-aided design (CAD) system with the incorporation of hierarchical parameterized cells in BiCMOS analog and RF technology for mixed signal applications. In: Proceedings of the Electrical Overstress/Electrostatic Discharge (EOS/ESD) Symposium, October 2002; pp. 296-305

[25] Perez C, Voldman S. Method of forming a guard ring parameterized cell structure in a hierarchical parameterized cell design, checking and verification system. U.S. Patent Application 20040268284, December 30, 2004

[26] Chapman P, Collins D, Voldman S. Design methodology of guard ring design resistance optimization for latchup prevention. U.S. Patent No. 7,549,135, June 16th, 2009

[27] Watson A, Voldman S. Methodology of quantification of transmission probability for minority carrier collection in a semiconductor chip. U.S. Patent No. 7,200,825, April 3rd, 2007

[28] Perez C, Voldman S. Method of displaying a guard ring within an integrated circuit. U.S. Patent No. 7,350,160, March 25th, 2008

[29] Yuan F. CMOS Current-Mode Circuits for Data Communication. New York: Springer; 2007

[30] Semenov O, Sarbishaei H, Sachdev M. ESD Protection Device and Circuit Design for Advanced CMOS Technologies. New York: Springer; 2008

[31] Russ C. ESD issues in advanced CMOS bulk and FinFET technologies: Processing, protection devices and circuit strategies. Microelectronics and Reliability;**208**:1403-1411

[32] Galy P, Bourgeat J, Guitard N, Lise JD. Ultracompact ESD protection with BIMOS-merged dual back-to-back SCR in hybrid bulk 28-nm FD-SOI advanced CMOS technology. IEEE Transactions on Electron Devices. 2017

[33] Dai CT, Chen SH, Linten D, et al. Latchup in bulk FinFET technology. IEEE Transactions of Electron Devices. 2017

Chapter 4

Design of Controller for Brushless Direct Current Motors Using FPGA

Suneeta Harlapur

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.79873

Abstract

Brushless direct current (BLDC) motors are the pillar of advanced controllers. This chapter presents a portion of the central thoughts hidden plan of FPGA based BLDC motor controller. It covers a considerable amount of ground, yet at a genuinely essential level to make central ideas clear. This chapter gives a great strategy which is useful to aid the outline and control of financially savvy, productive brushless direct current (BLDC) motors. Speed Control of BLDC motor utilizing PIC microcontrollers requires more equipment, and with the accessibility of FPGA adaptable highlights inspired to build up a financially savvy and dependable control with variable speed go. In this chapter, utilizing an algorithm which utilizes the Resolver signals caught from the motor is created with the assistance of Resolver to Digital converters. The VHDL program produces the terminating beats required to drive the MOSFETs of three stage completely controlled scaffold converter driven by drivers. The provided outline procedure is observed to be great and proficient.

Keywords: BLDC, FPGA, MOSFET, RDC, VHDL

1. Introduction

Brushless direct current (BLDC) motor controllers have received considerable attention in the past few years. The desirable features of brushed DC torque motors like torque-speed characteristics, accurate speed control are maintained in the BLDC motor approach, the problems posed by brush DC motors like arcing, which cause high EMI and frequent changes of brushes and commutators have been eliminated or minimized.

1.1. History of brushless DC motor

Most punctual confirmation of brushless DC motor was in 1962, directly after Wilson and Trickey shaped a "DC Machine with Solid State Commutation". It was in this way created owing to summit torque, summit reaction drive for claim to fame dedications, for example, tape and circle drives for PCs, mechanical autonomy and situating frameworks and in flying machine where brush wear was grievous because of low stickiness. In conjunction with approach denoting equivalence capable and changeless magnet stuffs with high power, high voltage transistors in the ahead of schedule to mid-1980s the capacity to create such a motor reasonable turned into a realism.

Impressive primary substantial brushless DC motors of 50 hp. were composed at POWERTEC Industrial Corporation in the late 1980s by Robert E. Lordo. Today, the greater part of the significant motor makers makes brushless DC motors. Brushless DC drives take a shot at the same standard as all DC motors yet the motor is worked "back to front" along with the fields on top of pole of the motor and its "armature" all things considered. The fields turn and effective "armature" stays stationary.

Keeping in mind the end goal to copy the activity of the commutator, an encoder was mounted in contact with the pole of the motor to intellect the position of the fields on the pole. The controller "meets" the attractive position data and decides through the basic rationale of which motor lead ought to have current setting off to a winding and which motor lead ought to give back the current from the winding.

1.2. Construction and operation of the BLDC motor

A BLDC motor comprises of a stator made out of covered steel stacked up to convey the windings. The brushless motors are for the most part controlled by utilizing a three stage power semiconductor span. In numerous motors, the essential quantities of loops are imitated to have littler introduction steps and littler torque swells.

A BLDC motor configured in a star pattern with three coils is considered here. The rotor in a typical BLDC motor is made out of permanent magnets. Increasing the number of poles does give better torque at the cost of reduced maximum possible speed [1, 2]. The motor requires a rotor position sensor for beginning and giving legitimate substitution succession to turn on the force gadgets in the inverter span. In light of the rotor position, the force gadgets are commutated successively for each 60°.

The replacement succession for BLDC motors has three windings. The first is empowered to positive force (current goes into the winding), the second twisting is for negative force (current ways out from the winding) and the third one is in a non-invigorated condition. The cooperation between the attractive field produced by the stator loops and the lasting magnets makes the required torque.

The BLDC motor drive framework comprises of a DC power supply changed on to the stator stage windings of the motor through an inverter by force exchanging gadgets. The discovery of rotor position decides the exchanging arrangement of the inverter. Three-stage inverters are for the most part used to control these motors, requiring a rotor position sensor for beginning

and giving the correct recompense grouping to stator windings. These position sensors can be Hall sensors, Resolvers, or Absolute position optical encoders. Though sensorless BLDC motor control is feasible using back-EMFs, they have some disadvantages. But still, sensorless control of BLDC motor has been receiving great interest.

1.3. Electronic commutation

With a specific end goal to make the motor pivot, the curls are invigorated in a pre-characterized succession, making the motor turn in one course. Running the grouping in the converse request makes the motor keep running the other way. The course of the current decides the introduction of the attractive field created by the loop.

The attractive field pulls in and repulses the changeless magnet rotor. By changing the present stream in the curls and in this way the extremity of the attractive fields at the right minute and in the right succession, the motor turns. Rotation of the current through the stator curls is alluded to as 'commutation'.

A three-phase BLDC motor has six steps of commutation. In six-step commutation, only two out of the three BLDC motor windings are used at a time, as shown in **Figure 1** using a three-phase half-bridge inverter arrangement.

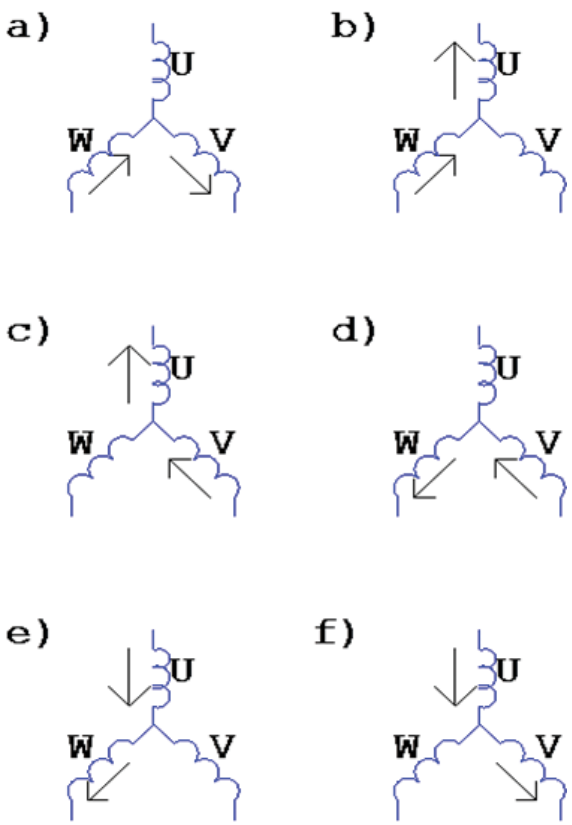

Figure 1. Six stages of commutation.

Steps are equivalent to 60 electrical degrees, and so, six steps make a full, 360° rotation. When each of the six states in the recompense arrangement has been executed, the grouping is rehashed to proceed with the revolution of the motor. This succession speaks to a full electrical turn. For motors with numerous post matches, the electrical revolution does not relate to mechanical turn.

In a BLDC motor, the commutation is achieved using feedback sensors. Hall Effect Sensors, Resolvers and Optical encoders are commonly used feedback sensors. In this research work, a resolver fitted to the motor shaft has been used as the feedback device, whose two signals are converted to a precise shaft position, using a resolver to digital converter (AD2S83) with a resolution of 12-bits.

1.3.1. Three phase inverter

The BLDC motor control comprises of creating DC streams in the motor stages. This control is subdivided into two free operations: in the first place, stator and rotor flux synchronization, and after that control of the present worth. Both operations are acknowledged through the three-stage inverter portrayed in the accompanying plan. The flux synchronization has been gotten from the position data originating from resolver. From the position, the controller characterizes the proper pair of MOSFET, which must be driven. The direction of the current to a settled 60° reference can be figured it out as shown in **Table 1** and circuit shown in **Figure 2** respectively.

1.3.2. Resolvers

Resolvers are transducers that convert the angular position and/or angular velocity of a rotating shaft to an electrical signal. They deliver signals proportional to the sine and cosine of the shaft angle. When the rotor is excited with a reference voltage of the form A sin(ωt), the voltages induced across the two stator windings are of the form:

$$S1 - S2 = A \sin(\omega t) \sin(\theta) \tag{1}$$

$$S2 - S4 = A \sin(\omega t) \cos(\theta) \tag{2}$$

where 'θ' is the shaft angle of the rotor. The two resolver signal outputs form the input to a Resolver to Digital Converter (RDC), which digitizes the shaft angle information into a digital format, for further processing by FPGA for the electronic commutation.

1.3.3. Resolver to digital converter

The resolver mounted on the motor shaft takes a shot at the transformer standard. The essential twisting is on the resolver's rotor and relying upon its pole edge, the prompted voltage in the two auxiliary windings are moved by 90°. The position information is obtained in a digital format using an Analog Devices Resolver to Digital Converter (RDC) [3]. The RDC also provides velocity signal in analog form with a 32.5 rps/V dc.

Sectors degree	Coil excitation	MOSFET ON
0–60°	W-V	T5, T4
60–120°	W-U	T5, T2
120–180°	V-U	T3, T2
180–240°	V-W	T3, T6
240–300°	U-W	T1, T6
300–360°	U-V	T1, T4

Table 1. Sector degree versus coil excitation.

1.3.4. Field programmable gate array (FPGA)

The Spartan-3 FPGA [4] with advanced process technology delivers more functionality in BLDC motor controller. The Spartan-3 family is a superior alternative to mask programmed ASICs and avoids the high initial cost, lengthy development cycles, and the inherent inflexibility of conventional ASICs. FPGA programmability permits modifications in the field without disturbing the hardware setup.

The Spartan-3 XC3S400 gadget comprises of 896 Configurable Logic Blocks (CLBs) contains RAM-based Look-Up Tables (LUTs) to actualize rationale and capacity components so that there will be no need of outer memory. Info/yield Blocks (IOBs) control the stream of information between the 116 I/O sets. Computerized Clock Manager (DCM) squares give self-aligning, completely advanced answers for conveying, deferring, duplicating, partitioning, and stage moving clock signals.

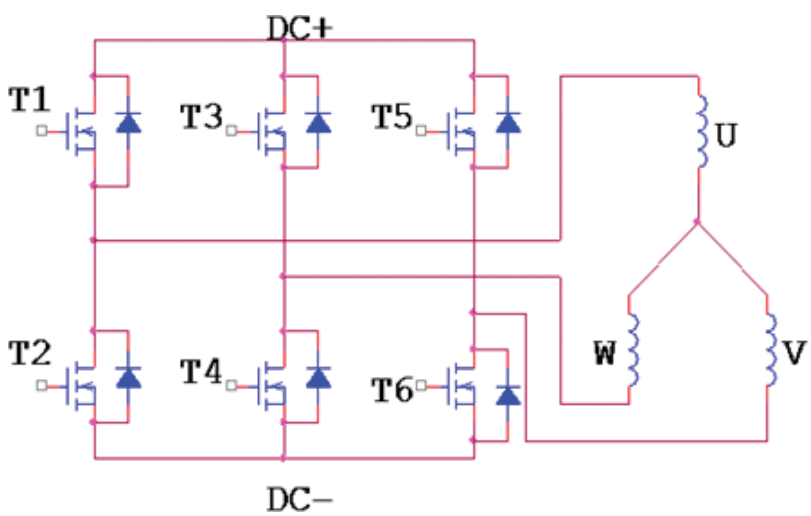

Figure 2. Three phase inverter and stator coil excitation.

1.4. Implementation of BLDC motor controller on FPGA

The Spartan-3 XC3S400 FPGA has a very good alternative to mask programmed ASICs and avoids the high cost and lengthy development process of BLDC motor controller.

1.4.1. Implementation of open loop BLDC motor controller on FPGA

The FPGA works like a controller to read the information from resolver to digital converter and to perform suitable electronic commutation. The controller is implemented for the constant speed by controlling the width of the PWM signal. The scheme of FPGA role in open loop BLDC motor controller is shown in **Figure 3**.

1.4.2. Implementation of closed loop BLDC motor controller on FPGA

The FPGA forms a controller to read-in resolver to digital converter, perform electronic commutation by reading the servo error from the analog to digital converter.

The speed control function is implemented by controlling the width of the PWM gated pulses. This plan of FPGA part in BLDC motor speed controller is appeared in **Figure 4**.

1.4.2.1. Speed controller of BLDC motor

The variable velocity control of a BLDC motor is acquired by utilizing inverter yield which has a variable recurrence and variable voltage source. The speed of the motor is related to the number of poles and frequency of the supplied voltage as below:

$$N = 120f/P \tag{3}$$

where N—speed in rpm, P—number of poles and f—frequency of the supply.

The selected BLDC motor for this work has six numbers of poles and tested for 1000 rpm speed with a 50 Hz power supply. The period of this supply is 20 ms and the duration of each

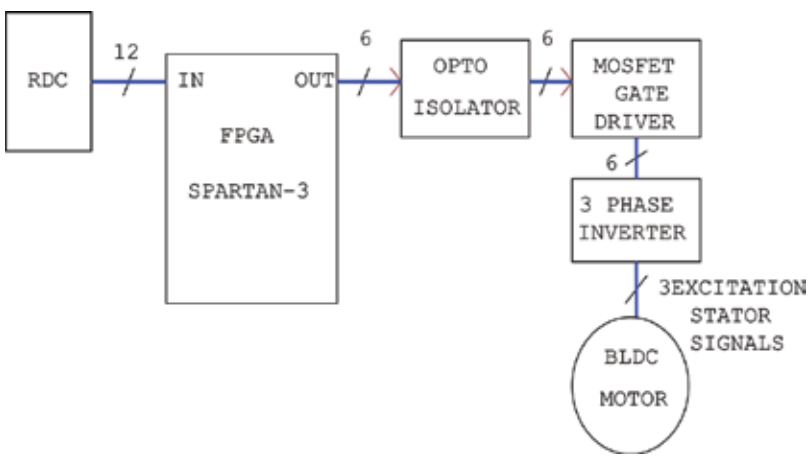

Figure 3. FPGA as open loop BLDC motor controller.

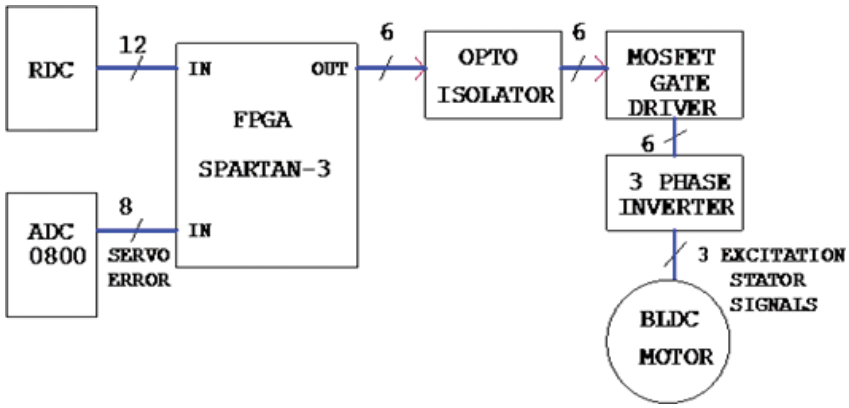

Figure 4. FPGA as closed loop BLDC motor controller.

stage of the six step commutation is 3.33 ms. Thus, the set frequency is configured according to the rpm required.

The variable voltage is obtained, using PWM technique by modifying the width of the pulses. This variable voltage sends variable current to the stator coils based on the required torque of the load.

In this work, a hybrid approach has been selected for the BLDC motor speed controller. The speed and the current loops have been implemented by using an operational amplifier. The digitized error is read by using the FPGA to compute the pulse width of the waveform to be sent to the gate control of MOSFETs. A closed loop speed controller requires a reference speed to follow. The motor speed is fed back to determine the error between the reference speed and motor speed. This error in speed is amplified and fed to a current loop where the actual motor current measured with LEM sensor is compared for the determination of the torque error. This error is amplified and fed to a 8-bit Analog to Digital Converter (Model ADC.0800). This digitized error is fed to the FPGA to determine the PWM width so as to control the stator voltage and current to the stator coils. This speed-controller scheme is shown in **Figure 5**.

The generated gated signals are passed on through opto-isolators to the MOSFET gate drivers. The three-phase full bridge circuit drives the motor. This BLDC motor has a resolver mounted on its rotor shaft to provide its angular position. This motor delivers a rated torque of 0.41 Nm at the rated speed of 7000 rpm.

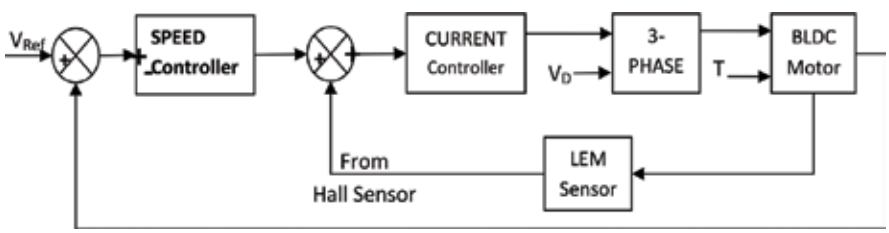

Figure 5. Block diagram of speed control of a BLDC motor.

The gate pulses generated from the FPGA are given to the driver circuit which consists of MOSFET- based inverter bridge. When the motor starts rotating, the coils are energized correspondent with the sequence.

The three phase currents are controlled to incorporate a quasi-square waveform in order to synchronize with back EMF to produce the constant torque. The resolver provides the motor shaft position in terms of sine and cosine waveforms.

The resolver feedback signals are in analog form, which is converted in to digital form with the help of resolver- to- digital converter (RDC). The RDC outputs are fed to the FPGA for further processing. The controller provides two error signals Velocity feedback and Current feedback.

2. Summary

FPGA has been interfaced to a RDC for position feedback information of the motor shaft. The electronic commutation sequence is generated and loaded into the output port to drive the three-phase inverter. The speed control is implemented with suitable analog electronics in conjunction with PWM determination, both for duty cycle and frequency by the FPGA. Mathematical modeling of the BLDC motors has been implemented and the MATLAB Simulink simulation is carried out to determine the static and dynamic response of the drive system.

Because of their elite brushless DC motors are increasing wide acknowledgment in telescope drive framework. The velocity control for a brushless DC engine has been outlined and incorporated into a FPGA. Keeping in mind the end goal to control motor torque, current controller is composed and executed in a FPGA SPARTAN-3.

Author details

Suneeta Harlapur

Address all correspondence to: sunitahaveri@gmail.com

Vemana Institute of Technology, Bangalore, India

References

[1] Gambhir R, Jha AK. Brushless DC motor: Construction and applications. International Journal of Engineering Science. 2013;**2**(5):72-77

[2] Jahns TM, Kliman GB, Neumann TW. Interior permanent magnet synchronous motors for adjustable-speed drives. IEEE Transaction on Industrial Application. 1986;**35**:738-746

[3] Analog Devices AD2S83 Data Sheets

[4] Xilinx Spartan 3 Family Data Sheets

Chapter 5

Functional Verification of Digital Systems Using Meta-Heuristic Algorithms

Alfonso Martínez-Cruz, Ignacio Algredo-Badillo,
Alejandro Medina-Santiago,
Kelsey Ramírez-Gutiérrez,
Prometeo Cortés-Antonio,
Ricardo Barrón-Fernández,
René Cumplido-Parra and Kwang-Ting Cheng

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.80048

Abstract

Trends in technological developments, such as autonomous vehicles, home automation, connected cars, IoT, etc., are based on integrated systems or application-specific integrated circuits with high capacities, where these systems require even more complex devices. Thus, new techniques to design more secure systems in a short time in the market are needed. At this point, verification is one of the highest costs in the manufacturing stage and most expensive in the design process. To reduce the time and cost of the verification process, artificial intelligence techniques based on the optimization of the coverage of behavioral areas have been proposed. In this chapter, we will describe the main techniques used in the functional verification of digital systems of medium complexity, focusing especially on meta-heuristic algorithms such as particle swarm optimization, genetic algorithms, and so on. Several results are presented and compared, where the opportunity areas will be described.

Keywords: digital systems, meta-heuristics, FPGAs, PSO, genetic algorithms, functional coverage, verification, automation

1. Proposed techniques for functional verification of digital systems

New applications in different areas, such as the automotive industry, robotics, IoT, and smartphones, among others, require increasingly complex digital devices. This implies the use of new techniques to reduce the design time of the devices, ensuring useful functionality according to the specification. It is important to know that manual simulation and functional verification require much time and expertise, so it has been necessary to develop software tools that improve performance, reduce manufacturing times, decrease verification costs, and increase the confidence level of the RTL implementations. In addition, new systems use a large amount of computational resources and new algorithms that increase the complexity of digital systems and require new methods to analyze and evaluate the device under verification (DUV). Several works of researchers on functional coverage methods have been made. Most studies use the following philosophies: static (methods based on logical or mathematical techniques), dynamic (methods based on simulation), and hybrid methods (combining static and dynamic). Next, works based on meta-heuristic and data mining algorithms report different methods for verification.

To perform verification of digital systems, different approaches have applied heuristic algorithms, for example, genetic algorithms (GA) that apply the evolution theory, where individuals within a population adapt to the conditions to the environment, compete for resources, and generate the evolution of the population through operators such as selection, crossing and mutation. Most of the time, the generation of pseudorandom tests produces worse results than this generation of test sequences. For example, authors in [1] perform a PowerPC architecture verification using genetic algorithms by generating pseudorandom custom instructions and encoding a sequence of instructions with a fixed length. The population size is small to reduce system simulation time. In the same way, in [2] the authors presented an implemented method to generate directed tests through a genetic algorithm, and a cell represents the chromosome in a uniform random distribution in two limits; the different parameters of the method were not fully automated; therefore, extensive knowledge of the evolutionary framework by the user is needed. In addition, in [3], the authors configure a genetic algorithm, which is included in a software platform to improve the functional coverage in a device. In this latter, chromosome coding is based on established instructions, and the proposed method helps to achieve uncovered tasks and increases the hit rate in the test of hard cases, improving the results of the pseudorandom test generation.

Some works have implemented ant colony optimization (ACO) and particle swarm optimization (PSO). On the one hand, the ACO uses the imitation of the behavior of ants seeking better paths from the initial place to the food place; ants get their food by means of pheromones and, in this way, other ants walk the paths and can provide positive feedback. In [4], a method based on the ACO that combines the pseudorandom test generation with a software platform that generates the digital system states is presented. The results show a reduction in computational complexity compared to random generation and other heuristics based on GA. On the other hand, the PSO algorithm is based on the interaction between the particles in the swarm. For instance, the authors in [5] present a verification method by using branches as a coverage metric and a PSO algorithm to perform the validation of RTL implementations.

http://dx.doi.org/10.5772/intechopen.80048

Other algorithms have been applied; for example, in [6], the authors used Bayesian networks in a functional verification method, and this type of networks is a model based on probabilistic graphs that are composed of random variables or nodes and edges that represent dependencies between them. The verification has feedback and the ability to cover hard cases and increase the coverage rate of progress, even though a manual configuration for the process was required. Other techniques were proposed for hardware verification based on meta-heuristics [7], where a differential evolution (DE) algorithm is applied; the verification is based on a coverage model using *coverpoints* and the algorithm is used to generate test vector sequences.

Works have improved the functional coverage using data mining. In [8], the authors proposed a learning methodology where knowledge from test is extracted. The extracted data is reused to generate tests with similar values to other important ones and cover new assertions. The method is applied to perform a constrained random verification of a processor and reports improvements in assertions coverage through the information extracted in the verification. The authors in [9] proposed an automatic learning method of rules regarding micro-architectural behavior of the instructions, and these rules were embedded in a stimuli generator tool. The method is applied in a microprocessor, improves the quality of the test cases generated and reaches interesting coverage events. In addition, [10] describes a method based on decision trees. In this method, before activating the sentences, they go through an engineering of formal verification to filter the candidate alterations in the output, generating automating RTL sentences. The proposed method was divided into two spaces: static and dynamic techniques. Static analysis techniques were used to direct the data mining process. In addition, Hidden Markov Models (HMM) are statistical methods that use probability measurements for sequential models of the data represented by sequences.

Other techniques used in functional verification are based on mutations that are changes of the RTL implementation, and such coverage metrics are used to drive the verification progress during simulation. For example, in [11] the authors proposed a methodology to verify a microprocessor using mutations. To test the vector sequences, the design simulation is performed first, then a set of mutations is added, and the verification is executed. Finally, a comparison of the results is made. One of the problems occurs when a large number of mutations are added because the verification time is increased.

In this chapter, an alternative hybrid method that uses coverage models is presented. This method represents the device behavior through CoverPoints, and fitness functions focused on sets of specific behavioral regions. In particular, a PSO algorithm with a re-initialization mechanism (BPSOr) is described. The method represents a hybrid technique that uses a simulation tool and meta-heuristic algorithms through a proposed verification interface.

2. Functional verification elements

In large-scale electronic integration design, functional verification is the verification process of a design logic that complies with specific rules design for its operation and manufacturing in an integrated circuit. The functional verification answers the question: “Does the proposed

electronics design meet the desired design and functionality requirements?" A complex task with times and high computational efforts is presented mainly in VLSI design. The functional verification is adjacent to a deeper design verification that, in addition to functional verification, adopts nonfunctional aspects such as time, design and power, implemented in the design of mixed circuits for signal processing.

There are different elements which work during the functional verification process. A verification system usually consists of several types of components:

1. Test generators are used in the stages of the functional verification where the test vectors are used to detect a fault presented in the specifications and the generation of the code. These generators use a full SAT type of NP resolution that is computationally expensive. In other types of generators, the vectors are created manually, for instance, the patented graphics-based generator (GBM). In short, modern generators create random vectors that are applied statistically on the design verification. Therefore, the users of the generators do not clearly specify the requirements to the test generation.
2. The supervisors interpret the stimuli produced by the vector generator for the DUV inputs. Generators create entries with a high level of abstraction, for example, transactions or instructions in assembly language. Supervisors convert this entry into inputs for the DUV as defined in the design interface specification.
3. The simulators (software tool) excite the circuits under verification to obtain their outputs, depending on the current state of the design and the input vectors injected (verification vectors). In this case, the software tool has a description of the design network list.
4. The monitor converts the state of design and its outputs into an abstraction transaction level that will be stored in a score-board database for later verification.
5. The verifier validates the score-board data. In some cases, the generator produces the expected results, in addition to the inputs. For those cases, the verifier must validate actual results that match the expected results.
6. The supervisor is included in the verification environment and manages all the above components together.

Figure 1 shows a pseudorandom test generation scheme where the functional coverage is used as coverage metric. The verification is done using constraints for the stimuli during the device simulation. After a specific number of iterations, the coverage information is reviewed by

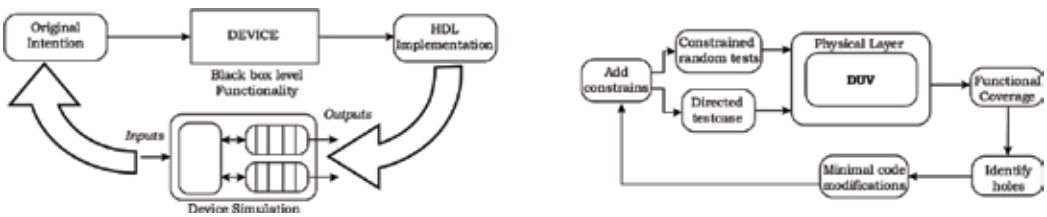

Figure 1. HDL verification through pseudorandom test generation.

http://dx.doi.org/10.5772/intechopen.80048

identifying the holes produced and, then adding more constraints. Finally, the process is executed until a stop criterion is met.

Verification is a very difficult task due to a large volume of possible test cases that exist even in a simple design. The verification can be attacked by many methods:

1. The logical simulation executes the logic of a circuit before building it to obtain its approximate behavior.
2. Simulation acceleration applies special-purpose hardware to the logic simulation problem.
3. Programmable logic creates a version of a system; this is expensive and even much slower than real hardware and orders of magnitude faster than simulation. For example, they can be used to start the operating system in a processor.
4. Formal verification attempts to prove mathematically that certain requirements are met or that certain undesired behaviors cannot occur.
5. Automated verification uses automation to adapt the test bench to changes in the register transfer level code.
6. Specific HDL versions and other heuristics are used to find common problems.

Different methodologies have been proposed in order to perform the functional verification. Three different philosophies have been suggested in order to perform the functional verification: static methods (formal methods), dynamic methods (which are based on simulation) and hybrid methods (which does not fall in formal and informal methods). Every philosophy contains different strategies in order to test the digital system functionality. For example, formal methods perform the verification using mathematical expressions to give a formal description of the device's behavior. Examples are model checking, theorem proving, etc.

During the verification based on dynamic methods, the stimuli are used to exercise the functionality, and test benches are also implemented and added to the verification environment. These methods are very scalable and practical. Due to the greater constant complexity of the devices, the use of these methods in the industry is very common. On the other hand, even if the designs are completely verified, it is not easy to guarantee that there are no errors.

Hybrid methods make up the third category, combining the formal and dynamic techniques. This type of methods is focused on increasing the coverage obtained from the bottleneck guiding the search through the full coverage space. A disadvantage is that its design requires broad background about verification techniques.

2.1. Problems solution through meta-heuristic algorithms

Searching directly for test vectors sets that appropriately evaluate and examine the functionality of the developed devices is not trivial. For example, for the deterministic methods, the consumption of resources is generally growing exponentially, which depends on the size and architecture of the circuit. Consequently, other solutions have been proposed, i.e., methods that use meta-heuristic are mainly applied to decrease the computational complexity when verifying the device.

Meta-heuristics methods are algorithms to find a global solution using local approximations and heuristics. A meta-heuristic represents a top-level strategy which guides the heuristics to solve a problem. Frequently, not all search details are specified and can be adjusted according to a specific problem. Alternatively, there are general techniques to handle the directed search where optimal local solutions will be avoided; they are employed in the verification context. For instance, in genetic algorithms, a population of individuals is used as an initial set of solutions. The fitness value of a test sequence represents how good an individual is. In addition, the search for solutions is directed by an individual's combination that uses a set of operators.

There are different definitions of meta-heuristics; commonly, a meta-heuristic can be defined as a process that drives other heuristics through a combination of elements to explore and exploit the search spaces. Besides that, it uses learning strategies to manage the information obtained and achieve optimal solutions. Some examples of meta-heuristics are ant colony, artificial bee colony algorithm, genetic algorithms, etc. Many works have used this type of algorithms to find solutions to different problems. Its applicability is suitable in optimization problems where the computation of cost functions is so expensive and influenced by a type of noise. Consequently, meta-heuristics are techniques that find good solutions in large search spaces.

2.2. Automated functional verification in digital systems

In this work, the functional verification of the devices is designed and executed automatically. Moreover, when the functional verification uses the coverage data that is produced from each simulation, it is named as "directed functional verification." A fundamental aspect is the coverage information (integrity measurement) for the test sets and represents the data where the revision is made in the verification. In addition, the analysis of this process allows the generation of new test sequences to evaluate other coverage regions.

The verification by simulation of the device is carried out when the expected functionality is translated into the implementation of RTL according to the specification and the criteria of the designer. Then, the device is reviewed through a series of steps, for example, checkers, monitors, test-benches, etc. In the end, the verification platform gives the coverage results that express the percentage of functionality verified. When reviewing the functional verification definition in [12], the RTL implementation of a device based on a set of features and operational requirements should be provided, to execute the verification, which is composed of the process that guarantees the device implementation that complies with each feature given in the specification.

Automated functional verification involves different elements such as coverage models, control flow graphs, test sequences, and cost functions, among other elements. When these elements interact, a system of test vector generation is formed. Some verification methods use this type of scheme to perform verification of digital devices.

An important case occurs when the test generation uses feedback information to explore new behavior regions, when this happens it is named as coverage-directed test generation. There are different definitions, for instance, according to [12], this generation allows to produce different test sequences to exercise different functionalities (characteristics of the coverage

models) of the device. Therefore, this process occurs when a test sequence is injected into the input of a device and then a new function value is exercised. Then, the value obtained is stored. Finally, all device states are reviewed and a new test is generated.

In other words, first, a test vector sequence is injected into the input of the device; then, if a new feature from the intended behavior is covered, the test sequence and the value of the feature exercised are stored. Later, the device states are reviewed and another test vector is produced to verify the "DUV." After this, all the states are verified and the values of coverage metrics are analyzed.

Figure 2 shows the main steps in the automation of the directed test generation. In this scheme, a verification plan based on based on a functional specification is needed, which describes what characteristics of the device will be verified and how it will be done.

2.3. Functional coverage models and coverage metrics

A functional coverage model can be described as a functional coverage space where the device behavior is captured. This means that it represents a coverage space that contains the interrelationships that exist between inputs, outputs, tasks, events, conditions and characteristics, which could show the correct functionality of a device with a confidence degree of a device. The coverage model is designed based on the implementation or device specification and a coverage metric or coverage structure.

A coverage metric consists of a heuristic to measure what part of the device behavior has been verified correctly. The main objective of this measure is to reflect which parts of the functionality have been met with correct execution during the processing of the information by the device, i.e., functional coverage (verify that all characteristics meet the specification), statement coverage (verify if the lines of code in the HDL implementation are exercised), branch coverage (analyze if the paths are traveled through the branches during the simulation), and finite-state machine (check how many states have been covered correctly).

The models are fundamental components of the verification process. A coverage model using stimuli, events, constraints, and CoverPoints is generated. It means that the coverage models are representations that map the intended behavior through characteristics, inputs, outputs, and its interrelations. A coverage model can be based on coverage points (CoverPoints). CoverPoints represent the values of each variable in a coverage model.

A coverage model can be defined as the different characteristics to represent the device behavior according to a functional specification that has different constraints. In particular,

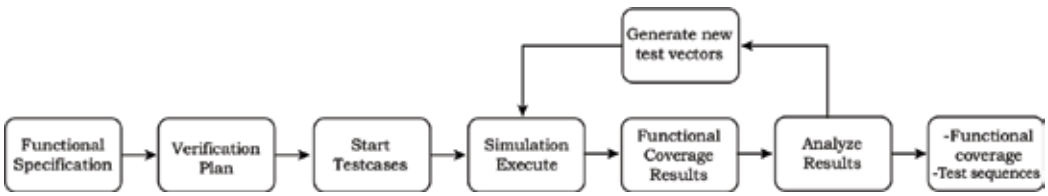

Figure 2. Automation of directed test generation.

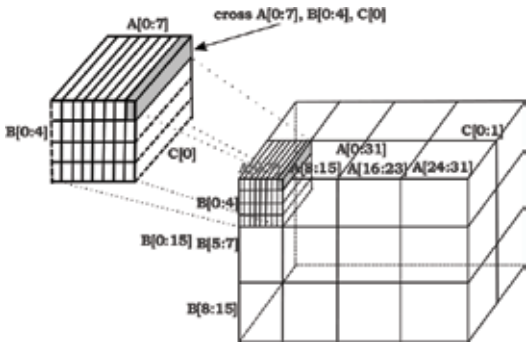

Figure 3. Functional coverage model.

the way of representing the behavior affects the granularity of the model, that is, a model with more characteristics can represent the original intention more efficiently, and as a consequence, it has a higher level of granularity. The accuracy of that model describes the implementation. The coverage model may contain explicit and implicit device behavior features. Moreover, the models are designed according to the device specification and implementation. **Figure 3** shows a coverage model where fidelity of a model determines how closely the model defines the actual requirements of a device behavior.

3. Verification method using BPSOr algorithm

The proposed verification method uses the BPSOr algorithm, which is based on several psychological aspects and social elements. In this social-cognitive context, individuals must interact among them, where the best performance occurs within the particle group and previous behaviors. Each individual is a particle, each particle group is a neighborhood, and each cognitive and social particle behavior is influenced by an improved performance from the groups.

At this point, two proposals are presented: *lbest* (local-best) and *gbest* (global-best). In the first proposal, the particle with the best performance in its groups affects to the remaining parts. In the second proposal, the swarm is important, because particles are connected among them, where the best performance of a particle from the swarm affects it and the results are improved.

In the swarm, each dimension is analyzed, and there are two main computational problems: memory and velocities; the first one establishes the best particle location, comparing the actual position and other better ones by means of the search. In addition, a key metric is the rate of change, which is computed for the particle based on the velocities to obtain *gbest* (the best global) and *lbest* (the best local solution). Incremental changes in both learning and attitude are simulated, providing the granularity of the search in the problem space. On the one hand, speed represents changes in probabilities, which may have the value "1" or "0." On the other

http://dx.doi.org/10.5772/intechopen.80048

hand and considering the particle dimension, the attitude of the changes represents the probability, which can be "1" or "0." For these reasons, the sigmoid function $S(V_{id})$ [13] transforms velocities to probability values and obtains a zero state for each particle, see Eq. 1. If v_{id} is high, the particle bit will probably be 1, and if the v_{id} is low, the particle bit will probably be 0, where v_{id} is a value in the range $[V_{min}, V_{max}] = [0.0, 1.0]$, ensuring that two possible values take the dimension bit (for the sequences): "1" or "0."

$$S(V_{id}) = \frac{1}{1 + exp\,(-v_{id})} \tag{1}$$

It is important to control the influence (from paths by each particle and other particles in the population), because the particles can move to the regions where the fitness variables have the best values. In this case, the pseudorandom values have produced better results when they are the mutation operators in the genetic algorithms. If the PSO algorithm is expressed in real numbers, a great number of problems are presented in binary domains, requiring extra operations for converting real values to binary values.

In binary versions, the PSO algorithm uses binary data directly with a re-initialization process, see **Algorithm 1**. The latter is composed of instructions or rules, where a particle is represented by a set and its elements are binary sequences. In this algorithm, in the first step, the position $\vec{x_i}$ and velocity $G\left(\vec{xi}\right)$ are initialized and computed for each particle. In the second step, $G\left(\vec{xi}\right)$ and its best previous position p_{id} are compared. If $G\left(\vec{xi}\right)$ is better, then its best position p_{id} is equal to x_{id}. In this case, the velocities v_{id} are compared. For every particle dimension, x_{id} has a value "0" when ρ_{id} position fitness is less than $s(v_{id}(t))$ (from sigmoidal speed function), but it has a value "1." These steps are executed until stop condition is reached.

Algorithm 1. Pseudocode of Binary PSO with a re-initialization process (BPSOr).

```
Swarm Initialization.
while No termination condition is reached do
    while No termination condition for coverage percentage is reached do
        for i=1 → Particle_Number do
            Compute fitness values: G(x⃗_i) and G(p⃗_i)
            If G(p⃗_i) is less or equal than G(x⃗_i), then the new best position p_id is x_id.
            for Each neighbour do
                Compare the best performance of population G(p⃗_g) with the performance of
                each particle in the neighborhood G(p⃗_i). If there is a better performance than
                the best global then replace it.
                Compute the current velocity v_i(t) and constrain it according to V_max and
                V_min.
                For every particle dimension x_id and the Sigmoid function, the assigned
                output is '1' or '0'.
            end
        end
    end
end
If the global best performance G(globalBest) is greater or equal than the best current performance
G(current⃗Best), then the velocities v_id(t) for each dimension particle are re-initialized according
to the probability P_re.
Best particle is registered.
Swarm is re-initialized.
```

In this pseudocode, V_{max} and V_{min} are constraints of each probability of change, where each position of the particle is considered, and the re-initialization process avoids local solutions and covers new behavior regions. This process is based on population-based measures, and if the best global performance is greater than the best current performance of the swarm, then the swarm of particles is initialized again. Consequently, the best particle position and the best particle of the population are stored. In addition, both the current positions and particle velocities are re-initialized. To re-initialize the velocities, a probability value is computed, whose aim is to avoid a convergence in an optimal local solution.

The main aspect of this algorithm is the decision when the bit string has a value of 1 or 0, which is based on the probability and is defined as a function of personal and social factors, see Eq. 2, where: (a) $v_{id}(t-1)$ is a measure of the current probability (individual predisposition) for the decision of 1 or 0; (b) φ_1 and φ_2 are positive random numbers, which are obtained form an uniform distribution, and they represent predefined upper limits; (c) $r1$ and $r2$ are positive random numbers, which can take some value from 0 to 1; (d) $x_{id}(t)$ describes the current state, when a bit-string d is analyzed for the individual i; (e) t represents the current discrete time, and $t-1$ represents the previous discrete time; (f) p_{id} is the variable that represents the best state and has a value of 1 if the individuals with the best success are located when x_{id} is 1 and 0 in otherwise; (g) p_{gd} is the best neighbor and has a value of 1 if the best success is reached by some number at the moment of examining the neighborhood with state 1 and has a value of 0 in the other case; and (h) ρ_{id} describes a vector or data structure of random numbers, which are obtained by using an uniform distribution among 0.0 and 1.0, and P_{re} represents the re-initialization factor with real value in the unit interval 0.0 to 1.0.

$$v_t(t) = v_{id}(t-1) + r1 \times \varphi_1\left(p_{id} - x_{id}(t-1)\right) + r2 \times \varphi_2\left(p_{gd} - x_{id}(t-1)\right) \tag{2}$$

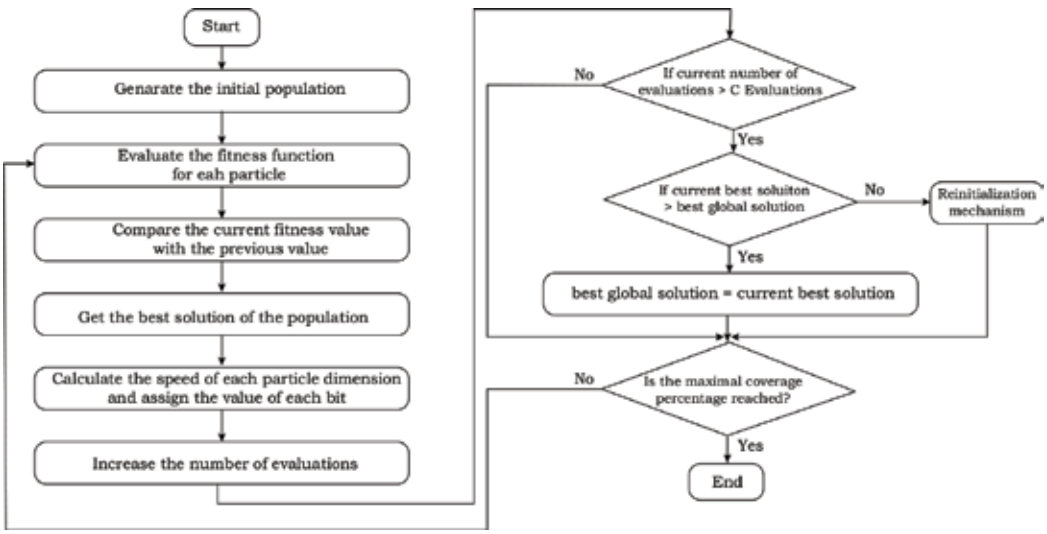

Figure 4. Flow diagram of BPSOr algorithm.

http://dx.doi.org/10.5772/intechopen.80048

Figure 4 shows the flow diagram of BPSOr algorithm. The different advantages of the binary PSO algorithm with re-initialization (BPSOr) enable to produce test sequences, operating in the verification context and analyzing the devices, which are being verified.

4. Test vector generation method

A proposed interface based on heuristic algorithms and a software tool is used. Moreover, some steps to verify the digital systems are performed. The description of the test generation method implemented in this work is shown in **Algorithm 2**. Firstly, the device parameters must be configured and initiated. In the same way, for the meta-heuristic process, several parameters are initiated and assigned based on the operational requirements (specifications) and implementation. Then, the set of device parameters are initialized.

Algorithm 2. General method of generation of test vector sequences.

```
Initialization of the device under verification.
Initialization of the parameters of the verification method.
Configuration and initialization of the functional verification modules.
Test generation algorithm initialization.
while Stop condition is not met do
    while The coverage criteria reached are not met do
        Generate new test sequences based on the coverage values achieved.
        Evaluate the digital device using a software tool.
        Analyze the information of the values obtained from functional coverage and save the
          statistics.
        Evaluate the fitness function based on the set of specified coverage points.
        Save the best current solution.
    end
end
```

In this case, BPSOr algorithm generates the test sequences; then, a simulation tool to evaluate them is used. The coverage information from device simulation is reviewed and saved. Then, the fitness variables are computed and the best values are stored, which are used in the new iteration.

Local-best topology was implemented in the verification method to perform different experiments. The scheme of this topology is shown in **Figure 5** where test vector sequences are clustering in some sets representing groups of particles; in this case, the particles or test sequences are affected by its fitness value and the best in its neighborhood. The best particle consists of the test sequence with the best fitness value in the group. Additionally, each test sequence or particle can communicate with others in its group. Later, in every iterations, the set of particles is directed toward the best particle in the swarm.

Global-best configuration is represented in **Figure 6**; in this topology, each particle is affected by the best solution in the swarm. All particles are included in the same group and they move toward the best solution. After every algorithm iteration, the test sequence with the best performance guides to the others through the search space.

On the other hand, the fitness function used in the algorithms is shown in Eq. 3. This function is focused on the percentage of holes produced in specific CoverPoints (P_{eh}). Therefore, the

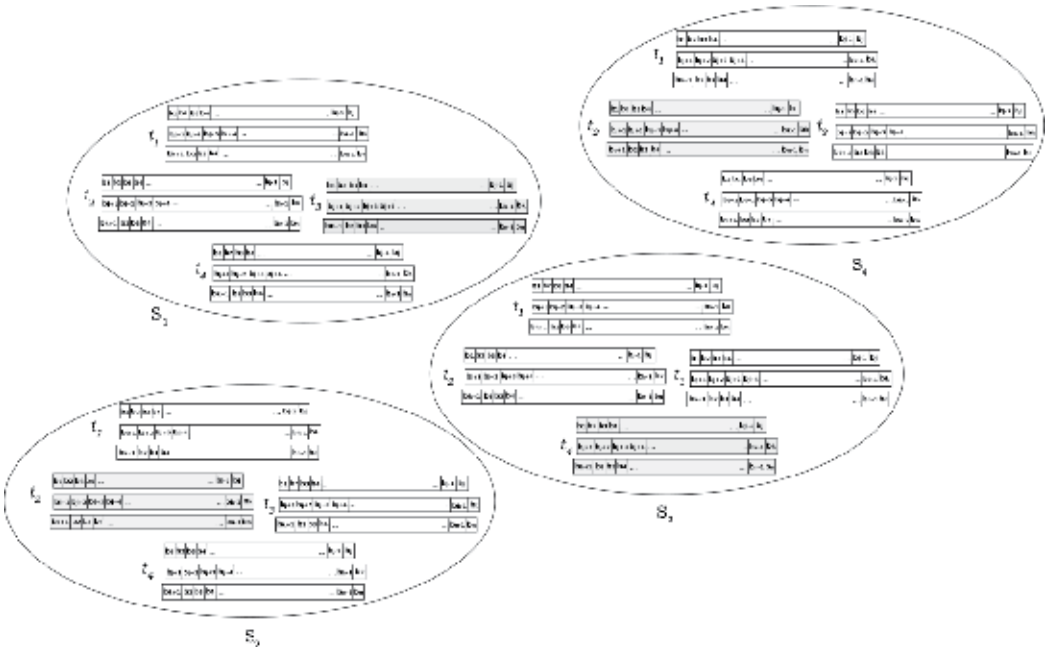

Figure 5. Groups of test vector sequences using the local-best topology.

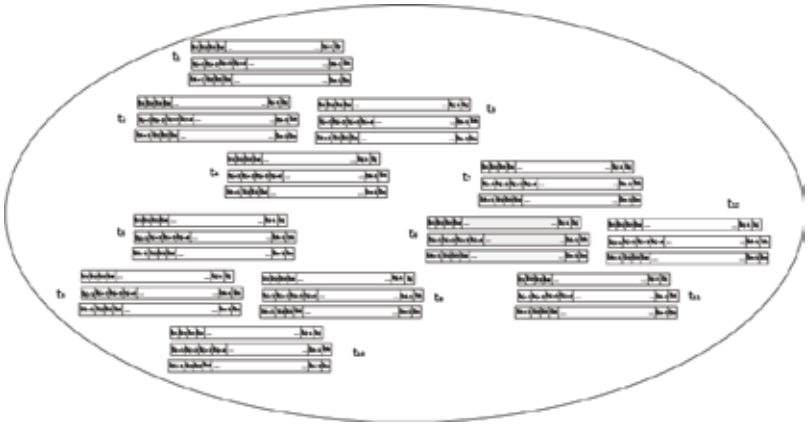

Figure 6. Groups of test vector sequences using the global-best topology.

problem is translated to maximize the number of points covered and, at the same time, minimize the percentage of holes in specific behavior regions.

$$f_1 = MAX\left(\frac{1}{P_{eh}}\right) \tag{3}$$

Test generation sequences are produced in the verification environment to verify the devices. In the beginning, a new binary sequence is tested and analyzed in the device, running the

http://dx.doi.org/10.5772/intechopen.80048

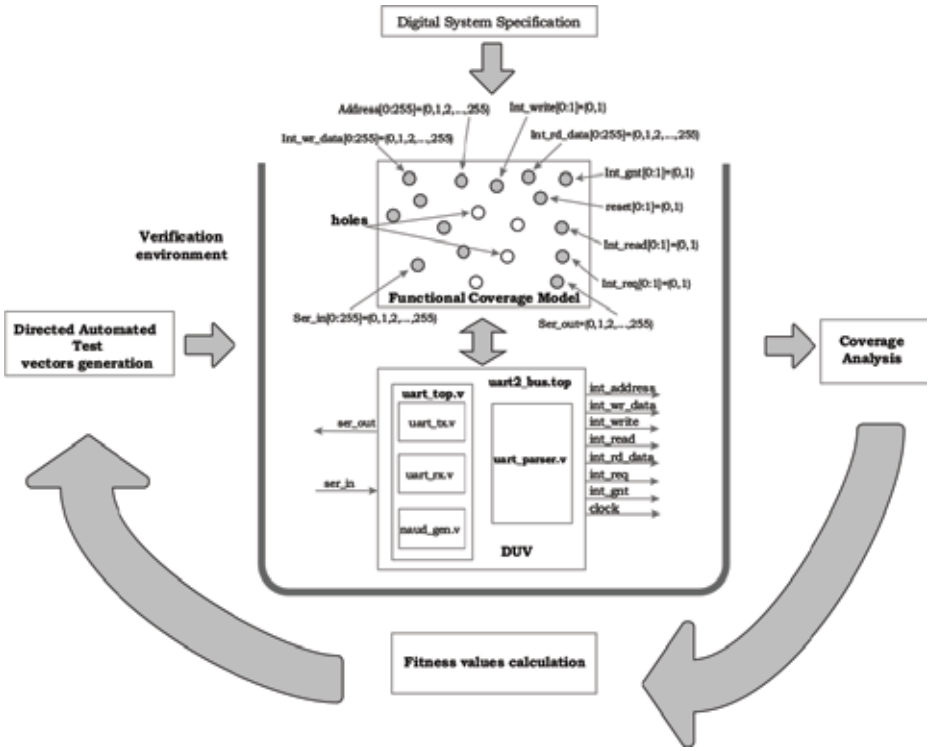

Figure 7. Proposed scheme.

respective simulation. Then, after the last sequence is completed, the cost values are quantified. Their calculation dependents of the points and holes are determined during the respective simulation. The information obtained is delivered to the generator module of test sequences. Therefore, a new sequence is generated and the process is repeated while the stop condition is not reached.

The proposed verification system is composed of several modules that are connected through an interface between C and SystemVerilog languages. **Figure 7** shows a scheme where the system couples the device under verification and the verification process is performed at the RTL level.

5. Case study

The proposed verification method is validated through different experiments using two digital systems. Additionally, the performance of the BPSOr, genetic algorithm, PSO, and random test generation is compared. RTL implementation of the devices was employed as benchmarks in the verification platform. The applicability of this type of method focuses on the block-level verification of IP cores because the automatic verification depends on the controllability degree of events generated from the stimulus during the device simulation. PSO algorithm with a re-initialization mechanism can be more complex computable, however, because this algorithm

achieves fine solutions very quickly, the verification time could be reduced. The best scenarios with different features of BPSOr algorithm will be presented.

Devices such as a UART-IP core were employed in order to perform its verification. The UART-IP can be used as a transmitter and receiver. A 16-bit address bus and an 8-bit data bus are included in the IP core. Its verification was based on the functional specification and the RTL code implementation. The coverage model was implemented using 785 bins in 12 CoverPoints. The initialization and configuration were performed based on the specification. Besides, the verification of a FIFO memory was performed using a coverage model with 784 bins. The memory is often contained in devices such as processors, UARTs, interfaces, and so on. Its implementation was designed in Verilog language and the configuration of the signals was controlled according to the features described in the functional specification.

To develop the proposed experiments, different values of parameters were used, which were included in several scenarios. Therefore, a scenario consists of a set of parameters that are used for the meta-heuristic algorithm. In the case of BPSOr algorithm, the parameters such as topology (global or local), velocity values, number of particles, and ϕ value were modified. Additionally, running a scenario of a defined number of times with a specific parameter configuration is defined as an experiment. The size of the swarm used was among 3 and 16 particles. Also, "global-best" and "local-best" topologies were implemented in the algorithm. The ϕ variable was modified with values from 2.0 to 4.0 for the scenarios. On the other hand, the evaluation of the test sequences was performed using two fitness functions, which are based on the coverage obtained. Basically, these functions get the CoverPoints and the holes generated at the run-time. When the simulation of a device ends, the coverage produced is sent to the test generator module and, finally, a new test is generated.

The results obtained are expressed in the best scenarios where information such as the best, the average, the total iterations, etc. are included. In addition, the binary test sequences were evaluated by modifying their number of elements or length. For example, if a particle is composed of two sequences, then its height is equal to 2.

The obtained results from the best scenarios will be presented to analyze the BPSOr performance. Furthermore, a genetic algorithm (GA) with elitism feature was implemented. Some algorithm parameters such as crossover percentage, mutation percentage, maximal number of evaluations, and population size were modified. A stop criterion was defined using the total number of evaluations. Besides, all CoverPoints were clustered focusing in the points that required to be exercised.

The experiments were performed using a computer with Linux Fedora Core 23. The features of the computer are as follows: Processor model: Intel Core i7-4790 K CPU-4 GHz., RAM: 8 GB, CPU: 4298.5 MHz, and Cache: 8192 KB. Additionally, all experiments were performed over a Linux Fedora Operating System, where the verification platform was successfully installed. After this, the obtained data were saved and reviewed. The obtained results from simulations were handled as statistical information to obtain the best fitness values.

When the verification process is performed, some characteristics could not be exercised due to different factors; for instance, if the behavior regions have the same cost values, then the fitness

functions could not give a difference regarding to other regions. Even, if more algorithm iterations are used, then the test sequences generated will cover the same behavior regions and the holes will not be covered. It is important to design strategies by focusing on the regions that are not easily covered. One strategy consists of a group the CoverPoints in sets with different weights to produce higher behavior areas. In addition, efficient search algorithms are required. Therefore, meta-heuristic algorithms can guide the search usefully and exercise all functionality of the device.

5.1. Experiments

The functional verification method based on meta-heuristic algorithms can test the functionality regions by focusing on specific behavior parts that can required more exploration. In these experiments, the verification method is used to verify two different digital systems. First, to show the performance of the GA a set of experiments will be developed. The genetic algorithm used was a binary version where the best individual remained in the next epoch (elitism). **Table 1** contains the parameters used for three different scenarios. Each experiment was run 30 times and then the coverage percentages and average time were stored. For instance, in the first case, a population of 100 individuals was configured with a crossover of 0.5 percentage and a mutation of 0.001 percentage.

Table 2 shows the obtained results for four best scenarios. Reviewing the results, in the third scenario, a few number of iterations was required in order to reach 100 coverage percentage. Besides, the average time used was 160.46 minutes.

Parameters	**GA scenarios**			
	1	**2**	**3**	**4**
Crossover percentage	0.5	0.5	0.45	0.45
Mutation percentage	0.001	0.001	0.0005	0.003
Population size	150	120	100	100
f	f_1	f_1	f_1	f_1

Table 1. GA algorithm settings for four different scenarios.

Final values	**1**	**2**	**3**	**4**
Best value	100	100	100	100
Worst value	95.44	97.13	99.08	98.30
Average value	98.23	99.12	99.87	99.62
Average evaluations	8090	7944	7353	7623
Average Time (min)	178.09	173.05	160.46	165.36

Table 2. Results obtained using a genetic algorithm in the platform to verify a UART-IP core.

Table 3 shows the four best scenarios using the BPSOr algorithm to perform the verification of a IP-UART core. One of the parameters that was changed is the swarm size. In this case, 3, 6, 9, and 12 particles were used in the proposed method. For example, in the first scenario, the parameters used were: 9 particles, 3 neighborhoods, ϕ = 4.0, global-best topology, and the f_1 cost function.

After, the experiments were performed, the obtained information was reviewed and the best results for the four scenarios are presented in **Table 4**. According to these results, using the fourth scenario, the average number of iterations was 1065 in 23.085 minutes to achieve 100 coverage percentage.

Table 5 contains the obtained results for four algorithms: GA, pseudorandom, BPSO, BPSOr, etc. In these experiments, different parameters over the verification platform were changed. In addition, four of the best scenarios are presented showing the best, worst, and average coverage. Also, the average number of iterations and the average time are added.

Commonly, the pseudorandom test generation is used to exercise the device functionality during the functional verification. Reviewing the results, at the start, the coverage percentage was increased very quickly. However, after achieving a coverage threshold percentage, more iterations to increase the coverage were needed. For instance, in the case of the UART-IP core, percentages over 95% were obtained.

According to the results the use of meta-heuristic algorithms to guide the search during the functional verification of digital systems is a good alternative because the behavior areas can

Parameters	**BPSOr scenarios**			
	1	**2**	**3**	**4**
Number of particles	9	6	12	3
Number of neighborhoods	2	2	4	1
ϕ max	4	4	4	4
Topology	G-best	G-best	G-best	G-best
f	f1	f1	f1	f1

Table 3. Configuration parameters of the BPSOr algorithm for four scenarios using the UART-IP core for two sequence solutions.

Scenario	**Number of evaluations**	**Best value**	**Worst value**	**Average**	**Time (min)**
1	2137	100	100	100	46.53
2	1608	100	100	100	37.12
3	2719	100	100	100	60.954
4	1065	100	100	100	23.085

Table 4. Results obtained for four different scenarios using the BPSOr algorithm in the proposed platform with a UART-IP core.

Final values	Binary GA	pseudorandom	BPSOr	BPSO
Best value	100	96.09	100	100
Worst value	99.08	94.53	100	100
Average value	99.87	95.27	100	100
Average evaluations	7353	8000	2137	2194
Average Time (min)	160.46	174.73	46.538	48.755

Table 5. Functional coverage results obtained using genetic algorithms, pseudorandom generation algorithms, PSO and BPSOr to verify a UART-IP core.

be covered very quickly. In the case of genetic algorithms, the population of individuals can evolve by modifying the test sequences to exercise new features of the device. One of the problems is that the guide is based on the evaluations of all population which evolves by means of operators such as mutation, crossover, etc.; when the population size increases, most number of evaluations in each epoch is required; thus, the simulation time is increased. During the functional verification, percentages over 99% were reached using the UART-IP core and the FIFO memory using less time than pseudorandom generation.

On the other hand, when the BPSOr algorithm was used in the verification platform, more functionality was exercised requiring less number of evaluations. Different from the original version of PSO, the BPSOr algorithm can re-initialize the particle swarm based on the current best coverage percentage and the number of iterations performed on run time. It means, if the coverage percentage is not increased, then the best solution, the best particle positions, and the positions and velocities of the particles are reinitialized. This mechanism is used to avoid to fall in local optima solutions and guide the search to behavior regions not explored. In addition, in most of the experiments, the coverage results obtained with BPSOr algorithm were higher than PSO algorithm. It is important to mention that meta-heuristics can be useful techniques to guide the test generation during the verification of devices.

6. Conclusions

Complexity of digital systems is constantly increasing; therefore, the implementation of new methods to improve the confidence and reduce the time of design is required. In this chapter, a verification method based on the use of meta-heuristic algorithms is described. Techniques such as genetic algorithms and particle swarm optimization algorithms were used to verify the digital systems, and a comparison was presented. Also, elements such as coverage models, fitness functions, and software tools are included. According to the results, the use of meta-heuristic algorithms such as the BPSOr algorithm and fitness functions can be useful to exercise the device functionality by focusing on behavior regions that have not been covered. In the case of GA, the coverage results obtained show that a lower number of iterations than pseudorandom test generation is required. Although in the best coverage scenarios a coverage percentage of 100 was obtained, it was observed that when increasing the number of

individuals, the number of iterations used was increased; thus, more time was used in each iteration. The PSO algorithm obtained higher coverage percentages than GA and pseudorandom generation. A main characteristic is that a fewer number of individuals or particles than GA are required. In the case of the BPSOr algorithm, the number of iterations required was less than PSO and GA in most of experiments; therefore, the verification time was reduced. Consequently, hybrid verification methods can improve the performance during the functional verification at block level of digital systems.

Author details

Alfonso Martínez-Cruz[1]*, Ignacio Algredo-Badillo[1], Alejandro Medina-Santiago[1], Kelsey Ramírez-Gutiérrez[1], Prometeo Cortés-Antonio[2], Ricardo Barrón-Fernández[3], René Cumplido-Parra[1] and Kwang-Ting Cheng[4]

*Address all correspondence to: amartinezc@inaoep.mx

1 National Institute of Astrophysics, Optics and Electronics, San Andres Cholula, Mexico

2 Tijuana Institute of Technology, Tijuana, Mexico

3 Computing Research Center, National Polytechnic Institute, Mexico City, Mexico

4 College of Engineering, University of California, Santa Barbara, California, USA

References

[1] Bose M, Shin J, Rudnick EM, Dukes T, Abadir M. A genetic approach to automatic bias generation for biased random instruction generation. In: Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546). Vol. 1. 2001. pp. 442-448

[2] Samarah A, Habibi A, Tahar S, Kharma N. Automated coverage directed test generation using a cell-based genetic algorithm. In: 2006 IEEE International High Level Design Validation and Test Workshop. Nov 2006. pp. 19-26

[3] Shen H, Wei W, Chen Y, Chen B, Guo Q. Coverage directed test generation: Godson experience. In: 2008 17th Asian Test Symposium. Nov 2008. pp. 321-326

[4] Li M, Hsiao MS. An ant colony optimization technique for abstraction-guided state justification. In: 2009 International Test Conference. Nov 2009. pp. 1-10

[5] Puri P, Hsiao MS. Fast stimuli generation for design validation of rtl circuits using binary particle swarm optimization. In: 2015 IEEE Computer Society Annual Symposium on VLSI. July 2015. pp. 573-578

[6] Fine S, Ziv A. Coverage directed test generation for functional verification using bayesian networks. In: Proceedings of 2003 Design Automation Conference (IEEE Cat. No.03CH37451). June 2003. pp. 286-291

[7] Cruz AM, FernÃ¡ndez RB, Lozano HM. Automated functional coverage for a digital system based on a binary differential evolution algorithm. In: 2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence. Sept 2013. pp. 92-97

[8] Chen W, Wang LC, Bhadra J, Abadir M. Simulation knowledge extraction and reuse in constrained random processor verification. In: 2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC). May 2013. pp. 1-6

[9] Katz Y, Rimon M, Ziv A, Shaked G. Learning microarchitectural behaviors to improve stimuli generation quality. In: 2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC). June 2011. pp. 848-853

[10] Vasudevan S, Sheridan D, Patel S, Tcheng D, Tuohy B, Johnson D. Goldmine: Automatic assertion generation using data mining and static analysis. In: 2010 Design, Automation Test in Europe Conference Exhibition (DATE 2010). March 2010. pp. 626-629

[11] Xie T, Mueller W, Letombe F. Mutation-analysis driven functional verification of a soft microprocessor. In: 2012 IEEE International SOC Conference. Sept 2012. pp. 283-288

[12] Cruz AM, Fernández RB, Lozano HM, Ramírez Salinas MA, Villa Vargas LA. Automated functional test generation for digital systems through a compact binary differential evolution algorithm. Journal of Electronic Testing. Aug 2015;**31**(4):361-380

[13] Kennedy J, Eberhart RC. A discrete binary version of the particle swarm algorithm. In: 1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation. Vol. 5. Oct 1997. pp. 4104-4108

Section 3

Machine Learning in Digital Systems

Chapter 6

Efficient Deep Learning in Network Compression and Acceleration

Shiming Ge

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.79562

Abstract

While deep learning delivers state-of-the-art accuracy on many artificial intelligence tasks, it comes at the cost of high computational complexity due to large parameters. It is important to design or develop efficient methods to support deep learning toward enabling its scalable deployment, particularly for embedded devices such as mobile, Internet of things (IOT), and drones. In this chapter, I will present a comprehensive survey of several advanced approaches for efficient deep learning in network compression and acceleration. I will describe the central ideas behind each approach and explore the similarities and differences between different methods. Finally, I will present some future directions in this field.

Keywords: deep learning, deep neural networks, network compression, network acceleration, artificial intelligence

1. Introduction

With the rapid development of modern computing power and large data collection technique, deep neural networks (DNNs) have pushed artificial intelligence limits in a wide range of inference tasks, including but not limited to visual recognition [1], face recognition [2], speech recognition [3], and Go game [4]. For example, visual recognition method proposed in [5] achieves 3.57% top-5 test error on the ImageNet LSVRL-2012 classification dataset, while face recognition system [6] achieves over 99.5% accuracy on the public face benchmark LFW [7], which both have surpassed human-level performance (5.1% on ImageNet [8] and 97.53% on LFW [9], respectively).

These powerful methods usually rely on DNNs containing millions or even billions of parameters. For example, the "very deep" VGG-16 [10], which achieves very impressive performance on ImageNet LSVRC 2014, uses a 16-layer deep network containing 138 million parameters and takes more than 500 MB in storing the model. Beyond the remarkable performance, there is increasing concern that the larger number of parameters consumes considerable resources (e.g., storage, memory, and energy), which hinders their practical deployment. First, for a deep neural network (DNN) usage on mobile, the storage bandwidth is very critical both for model size and data computation. For example, the mobile-first companies (such as Facebook and Baidu) are very care about the sizes of the uploaded file, while mobile sensor data companies (such as Google and Microsoft) usually build largely cloud powered systems with limited mobile computation. Second, for a DNN usage in cloud, memory bandwidth demand is very important to save transmission and power. Therefore, smaller models via DNN compression at least mean that they (1) are easier to download from App Store, (2) need less bandwidth to update to an autonomous car, (3) are easier to deploy on embedded hardware with limited memory, (4) need less communication across servers during distributed training, and (5) need less energy cost to perform face recognition.

The objective of efficient methods is to improve the efficiency of deep learning through smaller model size, higher prediction accuracy, faster prediction speed, and lower power consumption. Toward this end, a feasible solution is performing model compression and acceleration to optimized well-trained networks. In this chapter, I will first introduce some background of deep neural networks in Section 2, which provides us the motivation toward efficient algorithms. Then, I will present a comprehensive survey of recent advanced approaches for efficient deep learning in network compression and acceleration, which are mainly grouped into five categories, including network pruning category in Section 3, network quantization category in Section 4, network parameter structuring category in Section 5, network distillation category in Section 6, and compact network design category in Section 7. After that, I will discuss some future directions in this field in Section 8. Finally, Section 9 gives the conclusion.

2. Background

In this section, a brief introduction and analysis are given with some classic networks as examples on the structure of deep networks, computation and storage complexity, weight distribution, and memory bandwidth. This analysis inspires the behind motivation of model compression and acceleration approaches.

Recently, deep convolutional neural networks (CNNs) have become very popular due to their powerful representational capacity. A deep convolutional neural network (CNN) usually has a hierarchical structure of a number of layers, containing multiple blocks of convolutional layers, activation layers, and pooling layers, followed by multiple fully connected layers. **Figure 1** gives the structures of two classic CNNs, where (a) AlexNet [1] and (c) VGG-16 [10] consist of eight and sixteen layers, respectively. The two networks are larger than 200 MB and 500 MB, which makes them difficult to deploy on mobile devices. The convolutional layers dominate most of the computational complexity since they need a lot of multiplication-and-addition

http://dx.doi.org/10.5772/intechopen.79562

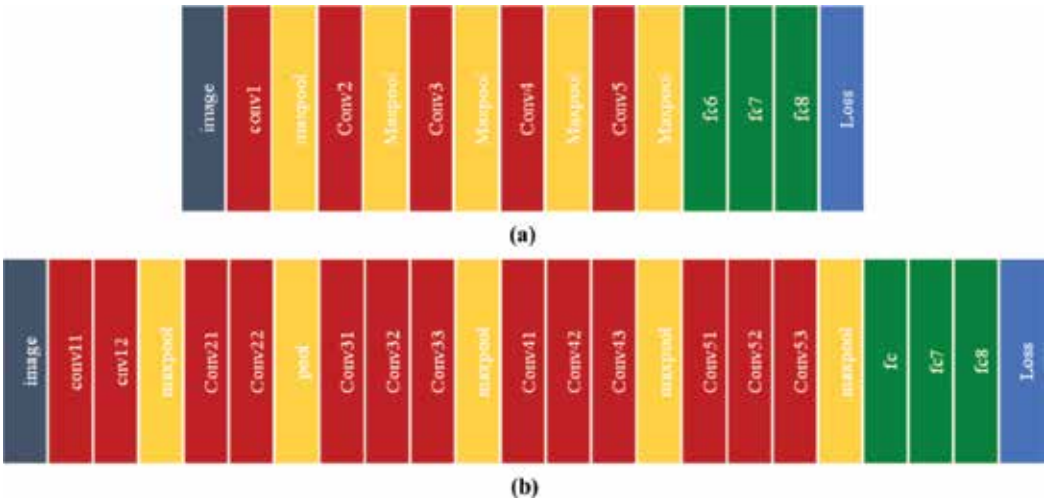

Figure 1. The structures of two classic deep networks. AlexNet (a) and VGGNet (b) all contain multiple convolutional layers (red), activation layers, and pooling layers (yellow), followed by multiple fully connected layers (green). The input and loss layer are marked in mazarine and blue, respectively.

(MAC) operations to extract local pattern, while they contain less weights due to weight sharing and local connectivity. By contrast, fully connected layers contain most of the weights since dense matrix-vector multiplications are very resource-intense. In addition, an activation layer (such as ReLU) contains a nonlinear function to activate or suppress some neurons. It can make the network more sparse and robust again to over-fitting while reducing the number of connections. A pooling layer is followed by a convolutional layer and aims to merge semantically similar features to reduce the memory.

As shown in **Table 1**, the complexity of CNNs could be spitted into two parts: (1) the computational complexity of a CNN is dominated by the convolutional layers and (2) the number of parameters is mainly related to the fully connected layers. Therefore, most model acceleration approaches focus on decreasing the computational complexity of the convolutional layers, while the model compression approaches mainly try to compress the parameters of the fully connected layers.

Network	Computational complexity			Parameter complexity		
	MACs	Conv (%)	FC (%)	Size	Conv (%)	FC (%)
AlexNet	724 M	91.9	8.1	61 M	3.8	96.2
VGG-16	15.5G	99.2	0.8	138 M	10.6	89.4
GoogleNet	1.6G	99.9	0.1	6.9 M	85.1	14.9
ResNet-50	3.9G	100	0	25.5 M	100	0

Table 1. The computational and parameter complexities and distributions for deep CNNs.

DNNs are known to be over-parameterized for facilitating convergence to good local minima of the loss function during model training [11]. Therefore, such redundancy can be removed from the trained networks in the test or inference time. Moreover, each layer contains lots of weights near zero value. **Figure 2** shows the probability distribution of the weights in two layers of AlexNet and VGG-16, respectively, where the weights are scaled and quantized into [−1, 1] with 32 levels to convenient visual display. It can be seen that the distribution is biased: most of the (quantized) weights on each layer are distributed around zero-value peak. This observation demonstrates that the weights can be reduced through weight coding, such as Huffman coding.

The memory bandwidth of a CNN model refers to the inference processing and greatly impacts the energy consumption, especially when running on embedded or mobile devices. To analyze the memory bandwidth of a trained CNN model, a simple but effective way is applied here by performing forward testing on multiple images and then analyzing the range of each layer output. The memory of each layer is dependent on bit width of each feature and

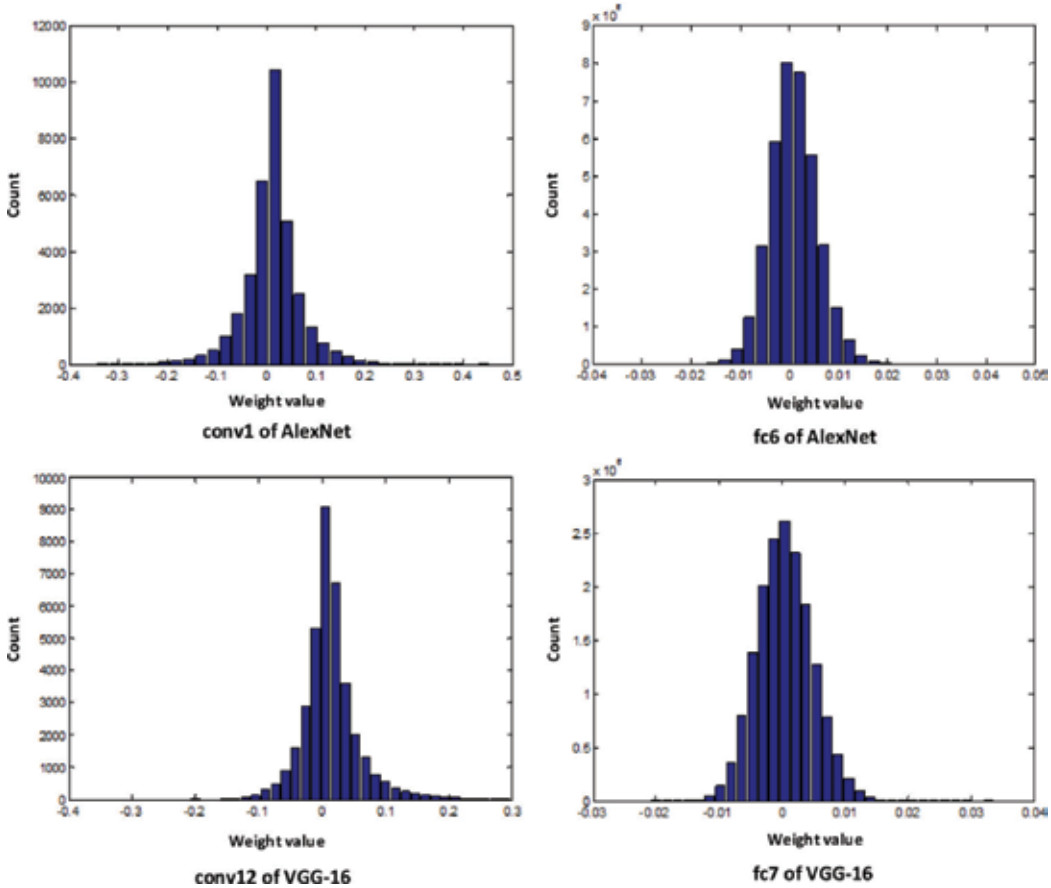

Figure 2. The probability distribution of weights in two layers of AlexNet and VGG-16. It is shown that the weight distribution is around a zero-value peak.

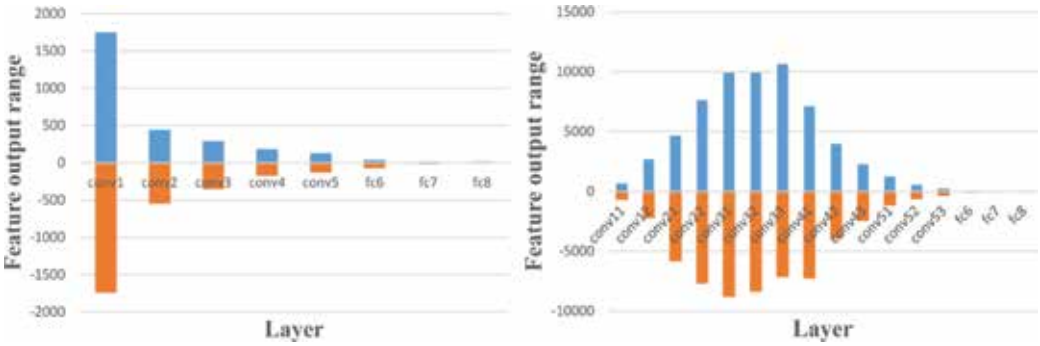

Figure 3. The memory bit-width range of each layer of AlexNet (left) and VGG-16 (right).

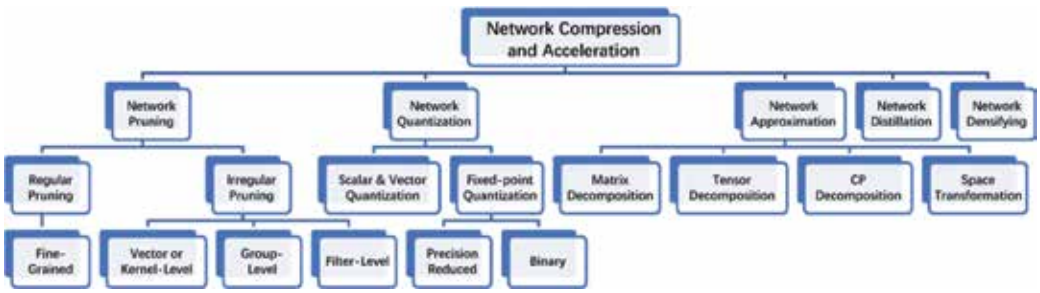

Figure 4. The main categories of network compression and acceleration approaches.

the number of output features. 1000 images from ImageNet dataset are randomly selected to perform inference with AlexNet and VGG-16, respectively; the mean range of output features on each layer are shown in **Figure 3**. It shows that the ranges of memory bandwidths in each layer are different and variable. Inspired by that, network compression and acceleration approaches can be designed to dynamically control the memory allocation in network layers by evaluating the ranges of each layer. Following these observations, many efficient methods for network compression and acceleration have been proposed, and several survey papers could be found in [12–14]. As shown in **Figure 4**, these approaches are grouped into five main categories according to their scheme for processing deep networks: pruning, quantization, approximation, distillation, and densification. In the following sections, I will introduce the advanced approaches in these categories.

3. Network pruning

DNNs are known to be over-parameterized for facilitating convergence to good local minima of the loss function during network training [11]. Therefore, the optimally trained deep networks usually contain redundancy on parameters. Inspired by that, network pruning category aims to remove such redundancy from the pre-trained networks in the inference time. In this

way, pruning approaches are applied to prune the unimportant or unnecessary parameters to significantly increase the sparsity of the parameters. Recently, many approaches are proposed, which consist of regular pruning approaches and irregular pruning approaches. As stated in [13], regular pruning refers to fine-gained pruning, while irregular pruning approaches are further categorized into four classes according to the pruning levels: vector level, kernel level, group level, and filter level. **Figure 5** shows different pruning methods. The core of network pruning is measuring the importance of weights or parameters.

Fine-grained pruning is the most popular approaches used in network pruning. It removes any unimportant parameters in convolutional kernels by using an irregular manner. In early work, LeCun et al. proposed optimal brain damage, a fine-grained pruning technique that estimates the saliency of the parameters by using the approximate second-order derivatives of the loss function w.r.t the parameters and then removes the parameters at a low saliency. This technique shows to work better than the naive approach. Later, Hassibi and Stork [15] came up with optimal brain surgeon, which performed much better than optimal brain damage although costing much more computational consumption. Recently, Chaber and Lawrynczuk [16] applied optimal brain damage for pruning recurrent neural models. Han et al. developed a method to prune unimportant connection and then retrain the weights to reduce storage and computation [17]. Later, they proposed a hybrid method, called Deep Compression [18], to compress deep neural networks with pruning, quantization, and Huffman coding. On the ImageNet dataset, the method reduced the storage required by AlexNet by 35× from 240 MB to 6.9 MB and VGG-16 by 49× from 552 MB to 11.3 MB both without loss of accuracy. Recently, Guo et al. [19] improved Deep Compression via dynamic network surgery which incorporated connection splicing into the whole process to avoid incorrect pruning. For face recognition, Sun et al. [20] proposed to iteratively learn sparse ConvNets. Instead of removing individual weights, Srinivas et al. [21] proposed to remove one neuron at a time. They presented a systematic way to remove the redundancy by wiring similar neurons together. In general, these irregular pruning approaches could achieve efficient compression of model sizes, but the memory footprint still has not been saved.

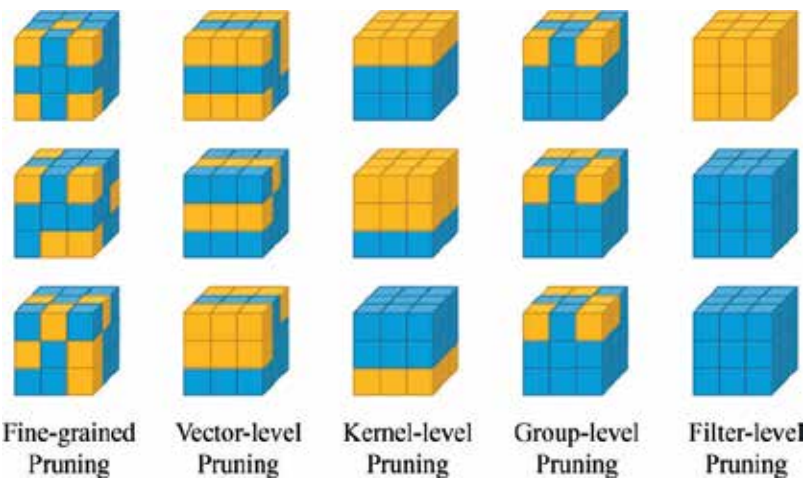

Figure 5. Different pruning methods for a convolutional layer that has three convolutional filters of size 3 × 3 × 3 [13].

Different from fine-grained pruning, vector-level and kernel-level pruning remove vectors in the convolutional kernels and 2D-convolutional kernels in the filters in a regular manner, respectively. Anwar et al. [22] proposed pruning a vector in a fixed stride via intra-kernel strided pruning. Mao et al. [23] explored different granularity levels in pruning. Group-level pruning aims to remove network parameters according to the same sparse pattern on the filters. In this way, convolutional computation can be efficiently implemented with reduced matrix multiplication. Lebedev and Lempitsky [24] revised brain damage-based pruning approach in a group-wise manner. Their approach added group-wise pruning to the training process to speed up the convolutional operations by using group-sparsity regularization. Similarly, Wen et al. [25] pruned groups of parameters by using group Lasso. Filter-level pruning aims to remove the convolutional filters or channels to thin the deep networks. Since the number of input channels is reduced after a filter layer is pruned, such pruning is more efficient for accelerating network inference. Polyak and Wolf proposed two compression strategies [26]: one based on eliminating lowly active channels and the other on coupling pruning with the repeated use of already computed elements. Luo et al. [27] proposed ThiNet to perform filter-level pruning. The pruning is guided by feature map, and the channels are selected by minimizing the construction error between two successive layers. Similarly, He et al. [28] applied an iterative two-step algorithm to prune filters by minimizing the feature maps. Generally speaking, these regular pruning (vector-level, kernel-level, group-level, and filter-level) approaches are more suitable for hardware implementations.

4. Network quantization

Typically, DNNs apply floating-point (such as 32-bit) precision for training and inference, which may lead to a large cost in memory, storage, and computation. To save the cost, network quantization category uses reduced precision to approximate network parameters. These approaches consist of scalar or vector quantization and fixed-point quantization (see **Figure 4**).

Scalar or vector quantization techniques are originally designed for data compression, where a codebook and a set of quantization codes are used to represent the original data. Considering that the size of codebook is much smaller than the original data, the original data could be efficiently compressed via quantization. Inspired by that, scalar or vector quantization approaches are applied to represent the parameters or weights of a deep network for compressing. In [29], Gong et al. applied k-means clustering to the weights or conducting product quantization and achieved a very good balance between model size and recognition accuracy. They achieved 16–24× compression of the network with only 1% loss of accuracy on ImageNet classification task. Wu et al. [30] proposed quantized CNN to simultaneously speedup the computation and reduce the storage and memory overhead of CNN models. This method obtains 4–6× speedup and 15–20× compression with 1% loss of accuracy on ImageNet. With the quantized CNN model, even mobile devices can accurately classify images within 1 second. Soulié et al. [31] proposed compressing deep network during the learning phase by adding an extra regularization term and combining product quantization of the network parameters.

Different from scalar and vector quantization approaches, fixed-point quantization approaches directly reduce the precision of parameters without codebooks. In [32], Dettmers proposed 8-bit approximation algorithms to make better use of the available bandwidth by compressing 32-bit gradients and nonlinear activations to 8-bit approximations, which obtains a speedup of 50×. In [33], Gupta et al. used only 16-bit wide fixed-point number representation when using stochastic rounding and incur little to no degradation in the classification accuracy. Lin et al. [34] proposed a fixed-point quantizer design by formulating an optimization problem to identify optimal fixed-point bit-width allocation across network layers. The approach offered larger than 20% reduction in the model size without performance loss, and the performance continued to improve after fine-tuning. Beyond fixed-point quantization with reduced precision, an alternative is using binary or ternary precision to lower the parameter representation. Soudry et al. [35] proposed Expectation Propagation (EP) algorithm to train multilayer neural networks. The algorithm has the advantages of parameter-free training and discrete weights, which are useful for large-scale parameter tuning and efficient training implementation on precision limited hardware, respectively. Courbariaux et al. [36] introduced BinaryConnect to provide deep neural network learning with binary weights. BinaryConnect acts as regularizer like other dropout schemes. The approach obtained near state-of-the-art results on permutation-invariant MNIST, CIFAR-10, and SVHN. Esser et al. [37] proposed training with standard backpropagation in binary precision by treating spikes and discrete synapses as continuous probabilities. They trained a sparse connected network running on the TrueNorth chip, which achieved a high accuracy of 99.42% on MNIST dataset with ensemble of 64 and 92.7% accuracy with ensemble of 1. Hubara et al. [38] introduced a method to train Binarized Neural Networks (BNNs) at run-time. In trained neural networks, both the weights and activations are binary. BNNs achieved near state-of-the-art results on MNIST, CIFAR-10, and SVHN. Moreover, BNNs achieved competitive results on the challenging ImageNet dataset (36.1%, top 1 using AlexNet) while drastically reducing memory consumption (size and number of accesses) and improving the speed of matrix multiplication at seven times. Later, Rastegari et al. [39] proposed XNOR-Net for ImageNet classification. They proposed two approximations to standard CNNs with binary-weight-networks and XNOR-Networks. The first approximation achieved 32× memory saving by replacing 32-bit floating-point weights with binary values. The second approximation enabled both the filters and the activations being binary. Moreover, it approximated convolutions using primarily binary operations. In this way, it achieved 58× faster convolutional operations and 32× memory savings while a much higher classification accuracy (53.8%, top 1 using AlexNet) than BNNs. Beyond the great reductions on network sizes and convolutional operations, these binarization schemes are based on simple matrix approximations and ignore the effect of binarization on the loss. To address this problem, Hou et al. [40] recently proposed a loss-aware binarization method by directly minimizing the loss w.r.t. the binarized weights with a proximal Newton algorithm with diagonal Hessian approximation. This method achieved good binarization performance and was robust for wide and deep networks. Motivated by local binary patterns (LBP), Xu et al. [41] proposed an efficient alternative to convolutional layers called local binary convolution (LBC) for facilitate binary network training. Compared to a standard convolutional layer, the LBC layer affords significant parameter savings, 9×–169× in the parameter numbers, as well as 9×–169× savings in model size. Moreover, the resulting CNNs with LBC layers

achieved comparable performance on a range of visual classification tasks, such as MNIST, SVHN, CIFAR-10, and ImageNet. Targeting at more faithful inference and better trade-off for practical applications, Guo et al. [42] introduced network sketching for pursuing binary-weight CNNs. They applied a coarse-to-fine model approximation by directly exploiting the binary structure in pre-trained filters and generated binary-weight models via tensor expansion. Moreover, an associative implementation of binary tensor convolutions was proposed to further speedup the generated models. After that, the resulting models outperformed the other binary-weight models on ImageNet large-scale classification task (55.2%, top 1 by using AlexNet). In order to reduce the accuracy loss or even improve accuracy, Zhu et al. [43] proposed Trained Ternary Quantization (TTQ) to reduce the precision of weights in neural networks to ternary values. TTQ trained the models from scratch with both ternary values and ternary assignment, while network inference only needed ternary values (2-bit weights) and scaling factors. The resulting models achieved an improved accuracy of 57.5%, top 1, using AlexNet on ImageNet large-scale classification task against full-precision model (57.2%, top 1, using AlexNet). TTQ was argued to be viewed as sparse binary-weight networks, which can potentially be accelerated with custom circuit. Generally speaking, the binary or ternary quantization approaches can greatly save the costs on model sizes, memory footprint, and computation, which make them friendly for hardware implementations. However, the accuracy needs to be improved especially in large-scale classification problems.

5. Network approximation

As stated in Section 2, the most computational cost of network inference comes from the convolution operators. In general, the convolutional kernel of a convolutional layer is represented with a 4D tensor, such as $K \in \mathbf{R}^{w \times h \times c \times s}$, where w and h are the width and height of the kernel filter, c is the number of input channels, and s indicates the target number of feature maps. The convolutional operation is performed by first transforming the kernel into a t-D ($t = 1,2,3,4$) tensor and then computed with efficient mathematical algorithm, such as by using Basic Linear Algebra Subprograms (BLAS). Inspired by that, network approximation aims to approximate the operation with low-rank decomposition.

Some approaches approximate 2D tensor by using singular value decomposition (SVD). Jaderberg et al. [44] decomposed the spatial dimension $w \times h$ into $w \times 1$ and $1 \times h$ filters, which achieved a 4.5× speedup for a CNN trained on a text character recognition dataset, with only an accuracy drop of 1%. Observing that the computation is dominated by the convolution operations in the lower layers of the network, Denton et al. [45] exploited the redundancy present within the convolutional filters to derive approximations that significantly reduce the required computation. The approach delivered 2× speedup on both CPU and GPU while keeping the accuracy within 1% of the original network on object recognition tasks. In [46], the authors proposed using a sparse decomposition to reduce the redundancy in model parameters. They obtained maximum sparsity by exploiting both interchannel and intrachannel redundancy and performing fine-tuning to minimize the recognition loss, which zeros out more than 90% of parameters and with a less than 1% loss of accuracy on the ImageNet.

Inspired by that the convolutional layer can be calculated with matrix-matrix multiplication; Figurnov et al. [47] used loop perforation technique to eliminate redundant multiplication, which allows to reduce the inference time by 50%.

By successive 2D tensor decompositions, 3D tensor decompositions can be obtained directly. Zhang et al. [48] applied the strategy that conducts a 2D decomposition on the first weight tensor after SVD. Their approach had been used to accelerate very deep networks for object classification and detection tasks. Another 3D tensor decomposition, Tucker decomposition [49], was proposed to compress deep CNNs for mobile applications by performing SVD along the input channel dimension for the first tensor after 2D decomposition. To further reduce complexity, a block-term decomposition [50] method based on low-rank and group sparse decomposition was proposed by approximating the original weight tensor by the sum of some smaller subtensors. By rearranging these subtensors, the block-term decomposition can be seen as a Tucker decomposition where the second decomposed tensor is a block diagonal tensor.

4D tensor decomposition can be obtained by exploring the low-rank property along the channel dimension and the spatial dimension. This is used in [51], and the decomposition is CP decomposition. The CP decomposition can achieve a very high speedup, for example, as 4.5× speedup for the second layer of AlexNet at only 1% accuracy drop.

Beyond low-rank tensor decomposition approaches which are performed in original space domain, there are some network approximation approaches by processing parameter approximation in transformation domain. In [52], Wang et al. proposed CNNPack to compress the deep networks in frequency domain. CNNPack treated convolutional filters as images and then decomposed their representations in the frequency domain as common parts shared by other similar filters and their individual private parts (i.e., individual residuals). In this way, a large number of low-energy frequency coefficients in both parts can be discarded to produce high compression without significantly compromising accuracy. Moreover, the computational burden of convolution operations in CNNs was relaxed by linearly combining the convolution responses of discrete cosine transform (DCT) bases. Later, Wang et al. [53] extended frequency domain method to the compression of feature maps. They proposed to extract intrinsic representation of the feature maps and preserve the discriminability of the features. The core is employing circulant matrix to formulate the feature map transformation. In this way, both online memory and processing time were reduced. Another transformation domain scheme is hashing, such as HashedNets [54] and FunHashNN [55].

In general, network approximation category focuses on accelerating network inference and reducing network sizes at a minimal performance drop. However, the memory footprint usually cannot be reduced.

6. Network distillation

Different from the above approaches which compress a pretrained deep network, network distillation category aims to train a smaller network to simulate the behaviors of a more complex

pre-trained deep network or the ensemble of multiple pre-trained deep networks. The training applies a teacher-student learning manner. Early work proposed by [56] proposed model compression, where the main idea was to use a fast and compact model to approximate the function learned by a slower, larger but better-performing model. Later, Hinton et al. proposed knowledge distillation [57] that trained a smaller neural network (called student network) by taking the output of a large, capable, but slow pre-trained one (called teacher network). The main strength of this idea comes from using the vast network to take care of the regularization process facilitating subsequent training operations. However, this method requires a large pre-trained network to begin with which is not always feasible. In [58], the authors extended this method to allow the training of a student that is deeper and thinner than the teacher, using not only the outputs but also the intermediate representations learned by the teacher as hints to improve the training process and final performance of the student. Inspired by these methods, Luo et al. [59] proposed to utilize the learned knowledge of a large teacher network or its ensemble as supervision to train a compact student network. The knowledge is represented by using the neurons at the higher hidden layer, which preserve as much information as the label probabilities but are more compact. When using an ensemble of DeepID2+ as teacher, a mimicked student is able to outperform it and achieves 51.6× compression ratio and 90× speedup in inference, making this model applicable on mobile devices. Lu et al. [60] investigated the teacher-student training for small-footprint acoustic models. Shi et al. [61] proposed a task-specified knowledge distillation algorithm to derive a simplified model with preset computation cost and minimized accuracy loss, which suits the resource-constraint front-end systems well. The knowledge distillation method relied on transferring the learned discriminative information from a teacher model to a student model. The method first analyzed the redundancy of the neural network related to a priori complexity of the given task and then trains a student model by redefining the loss function from a subset of the relaxed target knowledge according to the task information. Recently, Yom et al. [62] defined the distilled knowledge in terms of flow between layers and computed it with the inner product between features from two layers. The knowledge distillation idea was used to compress networks for object detection tasks [63–65].

Generally speaking, the network distillation approaches achieve a very high compression ratio. Another advantage of these approaches is making the resulting deep networks more interpretable. One issue needed to be addressed is reducing the accuracy drop ever improving the accuracy.

7. Network densifying

Another direct category for obtaining network compression and acceleration is to design more efficient but low-cost network architecture. I call this category as “network densifying,“ which aims to design compact deep networks to provide high accurate inference. In recent years, several approaches have been proposed following this line. The general ideas to achieve this goal include the usage of small filter kernels, grouping convolution, and advanced regularization.

Lin et al. [66] proposed Network-In-Network (NIN) architecture, where the main idea is using 1 × 1 convolution to increase the network capacity while making the computational complexity small. NIN also removed the fully connected layers instead of a global average pooling to reduce the storage requirement. The idea of 1 × 1 convolution is spread wide used in many advanced networks such as GoogleNet [67], ResNet [68], and DenseNet [69]. In [70], Iandola et al. designed a small DNN architecture termed SqueezeNet that achieves AlexNet-level accuracy on ImageNet but with 50× fewer parameters. In addition, with model compression techniques, SqueezeNet can be compressed to less than 1 MB (461× smaller than AlexNet). By using multiple group convolution, ResNeXt [71] achieved much higher accuracy than ResNet when costing the same computation. MobileNet [72] applied depth-wise convolution to reduce the computation cost, which achieved a 32× smaller model size and a 27× faster speed than VGG-16 model with comparable accuracy on ImageNet. ShuffleNet [73] introduced the channel shuffle operation to increase the information change within the multiple groups. It achieved about 13× speedup over AlexNet with comparable accuracy. DarkNet [74, 75] was proposed to facilitate object detection tasks, which applied most of the small convolutional kernels.

Moreover, some advanced regularization techniques are used to enhance the sparsity and robustness of deep networks. Dropout [76] and DropConnect [77] are widely exploited in many networks to increase the sparsity of activations for memory saving and weights for model size reduction, respectively. The activations neurons, including rectified Liner Unit (ReLU) [1], and its extends such as P-ReLU [78] are used to increase the sparsity of activations for memory saving while provide a speedup for model training, therefore they can facilitate the design of more compact networks.

8. Conclusions and future directions

It is necessary to develop efficient methods for deep learning via network compression and acceleration for facilitating the real-world deployment of advanced deep networks. In this chapter, I give a survey of recent network compression and acceleration approaches in five categories. In the following, I further introduce a few directions in the future in this literature including hybrid scheme for network compression, network acceleration for other visual tasks, hardware-software codesign for on-device applications, and more efficient distillation methods.

- **Hybrid scheme for network compression.** Current network compression approaches mainly focus on one single scheme, such as by using network quantization and network approximation. This processing leads to insufficient compression or large accuracy loss. It is necessary to exploit a hybrid scheme to combine the advantages from each network compression category. Some attempts can be found in [18, 79], which have demonstrated good performance.
- **Network acceleration for other visual tasks.** Most current approaches aim to compress and accelerate deep networks for image classification tasks, such as ImageNet large-scale

object classification, MNIST handwriting recognition, CIFAR object recognition, and so on. Very little effort have been attempted for other visual tasks, such as object detection, object tracking, semantic segmentation, and human pose estimation. Generally, direct using network acceleration approaches for image classification in these visual tasks may encounter a sharp drop on performance. The reason may come from that these visual tasks requires more complex feature representation or richer knowledge than image classification. The work has provided an attempt in facial landmark localization [80]. Therefore, this challenging problem on network acceleration for other visual tasks is one of the future directions.

- **Hardware-software codesign for on-device applications.** To realize the practical deployment on resource-limited devices, the network compression and acceleration algorithms should take the hardware design into consideration besides software algorithm modeling. The requirements from recent on-device applications such as autopiloting, video surveillance, and on-device AI enable tht it is highly desirable to design hardware-efficient deep learning algorithm according to the specific hardware platforms. This co-design scheme will be one future direction.
- **More effective distillation methods.** Network distillation methods have proven efficient for model compression in widespread fields beyond image classification, for example, machine translation [81]. However, these methods usually suffer from accuracy drop in inference, especially for complex inference tasks. Considering its efficacy, it is necessary to develop more effective distillation methods to extend their applications. Recent works [82–84] have given some attempts. Therefore, developing more effective distillation methods is one of the future directions in this field.

Acknowledgements

This work was partially supported by grants from National Key Research and Development Plan (2016YFC0801005), National Natural Science Foundation of China (61772513), and the International Cooperation Project of Institute of Information Engineering at Chinese Academy of Sciences (Y7Z0511101). Shiming Ge is also supported by Youth Innovation Promotion Association, Chinese Academy of Sciences.

Author details

Shiming Ge

Address all correspondence to: geshiming@iie.ac.cn

Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China

References

[1] Krizhevsky A, Sutskever I, Hinton G. Imagenet classification with deep convolutional neural networks. In: Neural Information Processing Systems (NIPS '12); 3-6 December 2012; Lake Tahoe; 2012. pp. 1097-1105

[2] Taigman Y, Yang M, Ranzato M, et al. DeepFace: Closing the gap to human-level performance in face verification. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '14); 24-27 June 2014; Columbus. New York: IEEE; 2014. pp. 1701-1708

[3] Amodei D, Ananthanarayanan S, Anubhai R, et al. Deep speech 2: End-to-end speech recognition in English and Mandarin. In: International Conference on Machine Learning (ICML '16); 19-24 June 2016; New York: IEEE; 2016. pp. 173-182

[4] Silver D, Huang A, Maddison CJ, et al. Mastering the game of Go with deep neural networks and tree search. Nature. 2016;**529**:484-489. DOI: 10.1038/nature16961

[5] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '16); 27-30 June 2016; Las Vegas. Nevada State: IEEE; 2016. pp. 770-778

[6] Schroff F, Kalenichenko D, Philbin J. Facenet: A unified embedding for face recognition and clustering. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '15); 8-10 June 2015; Boston. New York: IEEE; 2015. pp. 815-823

[7] Learned-Miller E, Huang GB, RoyChowdhury A, et al. Labeled faces in the wild: A survey. In: Kawulok M, Celebi M, Smolka B, editors. Advances in Face Detection and Facial Image Analysis. Cham: Springer; 2016. pp. 189-248. DOI: 10.1007/978-3-319-25958-1_8

[8] Russakovsky O, Deng J, Su H, et al. ImageNet large scale visual recognition challenge. International Journal of Computer Vision (IJCV). 2015;**115**(3):211-252. DOI: 10.1007/s11263-015-0816-y

[9] Kumar N, Berg AC, Belhumeur PN, et al. Attribute and simile classifiers for face verification. In: IEEE International Conference on Computer Vision (ICCV '09); 29 Sepetember–2 October 2009; Kyoto. New York: IEEE; 2009. pp. 365-372

[10] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations (ICLR '15); 7-9 May, 2015; San Diego; 2015

[11] Denil M, Shakibi B, Dinh L, et al. Predicting parameters in deep learning. In: Neural Information Processing Systems (NIPS '13); 5-10 December 2013; Lake Tahoe; 2013. pp. 2148-2156

[12] Sze V, Chen YH, Yang TJ, et al. Efficient processing of deep neural networks: A tutorial and survey. Proceedings of the IEEE. 2017;**105**(12):2295-2329. DOI: 10.1109/JPROC.2017.2761740

[13] Cheng J, Wang P, Li G, et al. Recent advances in efficient computation of deep convolutional neural networks. Frontiers of Information Technology & Electronic Engineering. 2018;**19**(1):64-77. DOI: 10.1631/FITEE.1700789

[14] Cheng Y, Wang D, Zhou P, et al. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. IEEE Signal Processing Magazine. 2018;**35**(1):126-136. DOI: 10.1109/MSP.2017.2765695

[15] Babak Hassibi, David G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In: Neural Information Processing Systems (NIPS '93); 1993; Denvor. San Francisco: Morgan Kaufmann; 1994. pp. 164-171

[16] Chaber P, Lawrynczuk M. Pruning of recurrent neural models: An optimal brain damage approach. Nonlinear Dynamics. 2018;**92**(2):763-780. DOI: 10.1007/s11071-018-4089-1

[17] Han S, Pool J, Tran J, Dally W. Learning both weights and connections for efficient neural networks. In: Neural Information Processing Systems (NIPS '14); 11-12 December 2014; Montreal; 2014. pp. 1135-1143

[18] Han S, Mao H, Dally W. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. In: International Conference on Learning Representations (ICLR '16); 2-4 May 2016; San Juan; 2016

[19] Guo Y, Yao A, Chen Y. Dynamic network surgery for efficient DNNs. In: Neural Information Processing Systems (NIPS '16); 5-10 December 2016; Barcelona; 2016. pp. 1379-1387

[20] Sun Y, Wang X, Tang X. Sparsifying neural network connections for face recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '16); 27-30 June 2016; Las Vegas. Nevada State: IEEE; 2016. pp. 4856-4864

[21] Srinivas S, Babu RV. Data-free parameter pruning for deep neural networks. In Proceedings of the British Machine Vision Conference (BMVC '15); 7-10 September 2015; Swansea: BMVA Press; 2015. pp. 31.1-31.12

[22] Anwar S, Hwang K, Sung W. Structured pruning of deep convolutional neural networks. ACM Journal on Emerging Technologies in Computing Systems (JETC). 2017;**13**(3):32. DOI: 10.1145/3005348

[23] Mao H, Han S, Pool J, et al. Exploring the Regularity of Sparse Structure in Convolutional Neural Networks [Internet]. Available from: https://arxiv.org/pdf/1707.06342 [Accessed: May 12, 2018]

[24] Lebedev V, Lempitsky V. Fast convnets using group-wise brain damage. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '16); 27-30 June 2016; Las Vegas. Nevada State: IEEE; 2016. pp. 2554-2564

[25] Wen W, Wu C, Wang Y, et al. Learning structured sparsity in deep neural networks. In: Neural Information Processing Systems (NIPS '16); 5-10 December 2016; Barcelona; 2016. pp. 2074-2082

[26] Polyak A, Wolf L. Channel-level acceleration of deep face representations. IEEE Access. 2015;**3**:2163-2175. DOI: 10.1109/access.2015.2494536

[27] Luo JH, Wu J, Lin W. ThiNet: A filter level pruning method for deep neural network compression. In: IEEE International Conference on Computer Vision (ICCV '17); 22-29 October 2017; Venice Italy; 2017. pp. 5058-5066

[28] He Y, Zhang X, Sun J. Channel pruning for accelerating very deep neural networks. In: IEEE International Conference on Computer Vision (ICCV '17); 22-29 October 2017; Venice. New York: IEEE; 2017. pp. 1389-1397

[29] Gong Y, Liu L, Yang M, et al. Compressing deep convolutional networks using vector quantization. In: International Conference on Learning Representations (ICLR '15); 7-9 May 2015; San Diego; 2015

[30] Wu J, Leng C, Wang Y, et al. Quantized convolutional neural networks for mobile devices. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '16); 27-30 June 2016; Las Vegas. Nevada State: IEEE; 2016. pp. 4820-4828

[31] Soulié G, Gripon V, Robert M. Compression of deep neural networks on the fly. In: International Conference on Artificial Neural Networks; 6-9 September 2016; Barcelona. Cham: Springer; 2016. pp. 153-160

[32] Dettmers T. 8-bit approximations for parallelism in deep learning. In: International Conference on Learning Representations (ICLR '16); 2-4 May 2016; Caribe Hilton; 2016

[33] Gupta S, Agrawal A, Gopalakrishnan K, et al. Deep learning with limited numerical precision. In: International Conference on Machine Learning (ICML '15); 6-11 July 2015; Lille. JMLR; 2015. pp. 1737-1746

[34] Lin DD, Talathi SS, Annapureddy VS. Fixed point quantization of deep convolutional networks. In: Proceedings of The International Conference on Machine Learning (ICML '16); 19-24 June 2016; New York. JMLR; 2016. pp. 2849-2858

[35] Soudry D, Hubara I, Meir R. Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights. In: Neural Information Processing Systems (NIPS'14); 8-13 December 2014; Montreal; 2014. pp. 963-971

[36] Courbariaux M, Bengio Y, David JP. Binaryconnect: Training deep neural networks with binary weights during propagations. In: Neural Information Processing Systems (NIPS '14); 11-12 December 2014; Montreal; 2014. pp. 3123-3131

[37] Esser SK, Appuswamy R, Merolla P, et al. Backpropagation for energy-efficient neuromorphic computing. In: Neural Information Processing Systems (NIPS '14); 11-12 December 2014; Montreal; 2014. pp. 1117-1125

[38] Hubara I, Courbariaux M, Soudry D, et al. Binarized neural networks. In: Neural Information Processing Systems (NIPS '16); 5-10 December 2016; Barcelona; 2016. pp. 4107-4115

[39] Rastegari M, Ordonez V, Redmon J, et al. XNOR-Net: Imagenet classification using binary convolutional neural networks. In: European Conference on Computer Vision (ECCV '16); 8-16 October 2016; Amsterdam. Cham: Springer; 2016. pp. 525-542

[40] Hou L, Yao Q, Kwok JT. Loss-aware binarization of deep networks. In: International Conference on Learning Representations (ICLR '17); 24-26 April 2017; Palais des Congrès Neptune, Toulon, France; 2017

[41] Juefei-Xu F, Boddeti VN, Savvides M. Local binary convolutional neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '17); 21-26 July 2017; Honolulu, Hawaii. New York: IEEE; 2017. pp. 19-28

[42] Guo Y, Yao A, Zhao H, Chen Y. Network sketching: Exploiting binary structure in deep CNNs. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '17); 21-26 July 2017; Honolulu, Hawaii. New York: IEEE; 2017. pp. 5955-5963

[43] Zhu C, Han S, Mao H, Dally WJ. Trained ternary quantization. In: International Conference on Learning Representations (ICLR '17); 24-26 April 2017; Palais des Congrès Neptune, Toulon, France; 2017

[44] Jaderberg M, Vedaldi A, Zisserman A. Speeding up convolutional neural networks with low rank expansions. In: Proceedings of the British Machine Vision Conference (BMVC '15); 7-10 September 2015; Swansea: BMVA Press; 2015

[45] Denton EL, Zaremba W, Bruna J, et al. Exploiting linear structure within convolutional networks for efficient evaluation. In: Neural Information Processing Systems (NIPS '14); 11-12 December 2014; Montreal; 2014. pp. 1269-1277

[46] Liu B, Wang M, Foroosh H, et al. Sparse convolutional neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '15); 8-10 June 2015; Boston. New York: IEEE; 2015. pp. 806-814

[47] Figurnov M, Ibraimova A, Vetrov D P, et al. Perforated CNNs: Acceleration through elimination of redundant convolutions. In: Neural Information Processing Systems (NIPS '16); 5-10 December 2016; Barcelona; 2016. pp. 947-955

[48] Zhang X, Zou J, He K, et al. Accelerating very deep convolutional networks for classification and detection. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2016;**38**(10):1943-1955. DOI: 10.1109/tpami.2015.2502579

[49] Kim YD, Park E, Yoo S, et al. Compression of Deep Convolutional Neural Networks for Fast and Low Power Mobile Applications [Internet]. Available from: https://arxiv.org/pdf/1511.06530 [Accessed: May 12, 2018]

[50] Wang P, Cheng J. Accelerating convolutional neural networks for mobile applications. In: Proceedings of the 2016 ACM on Multimedia Conference; 15-19 Oct. 2016; Amsterdam, The Netherlands; 2016. p. 541-545

[51] Lebedev V, Ganin Y, Rakhuba M, et al. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. In: International Conference on Learning Representations (ICLR '15); 7-10 May 2015; San Diego; 2015

[52] Wang Y, Xu C, You S, et al. Cnnpack: Packing convolutional neural networks in the frequency domain. In: Neural Information Processing Systems (NIPS '16); 5-10 December 2016; Barcelona; 2016. pp. 253-261

[53] Wang Y, Xu C, Xu C, Tao D. Beyond filters: Compact feature map for portable deep model. In: Proceedings of The International Conference on Machine Learning (ICML '17); 8-11 August 2017; Sydney. JMLR; 2017. pp. 3703-3711

[54] Chen W, Wilson J, Tyree S, et al. Compressing neural networks with the hashing trick. In: Proceedings of the International Conference on Machine Learning (ICML '15); 6-11 July 2015; Lille. JMLR; 2015. pp. 2285-2294

[55] Shi L, Feng S. Functional Hashing for Compressing Neural Networks [Internet]. Available from: https://arxiv.org/pdf/1605.06560 [Accessed: May 12, 2018]

[56] Bucilu C, Caruana R, Niculescu-Mizil A. Model compression. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 20-23 August 2006; Philadelphia. ACM; 2006. pp. 535-541

[57] Hinton G, Vinyals O, Dean J. Distilling the Knowledge in a Neural Network [Internet]. Available from: https://arxiv.org/pdf/1503.02531 [Accessed: May 12, 2018]

[58] Romero A, Ballas N, Kahou SE, et al. FitNets: Hints for thin deep nets. In: International Conference on Learning Representations (ICLR '15); 7-10 May 2015; San Diego; 2015

[59] Luo P, Zhu Z, Liu Z, et al. Face model compression by distilling knowledge from neurons. In: Thirtieth AAAI Conference on Artificial Intelligence (AAAI '16); 12-17 February. 2016; Phoenix, Arizona, USA; 2016. pp. 3560-3566

[60] Lu L, Guo M, Renals S. Knowledge distillation for small-footprint highway networks. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '17); 5-9 March 2017; New Orleans. New York: IEEE; 2017. pp. 4820-4824

[61] Shi M, Qin F, Ye Q, et al. A scalable convolutional neural network for task-specified scenarios via knowledge distillation. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP); 5-9 March 2017; New Orleans. New York: IEEE; 2017. pp. 2467-2471

[62] Yim J, Joo D, Bae J, et al. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '17); 21-26 July 2017; Honolulu, Hawaii. New York: IEEE; 2017. pp. 7130-7138

[63] Shen J, Vesdapunt N, Boddeti VN, et al. In Teacher We Trust: Learning Compressed Models for Pedestrian Detection [Internet]. Available from: https://arxiv.org/pdf/1602.00478 [Accessed: May 12, 2018]

[64] Li Q, Jin S, Yan J. Mimicking very efficient network for object detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '17); 21-26 July 2017; Honolulu, Hawaii. New York: IEEE; 2017. pp. 7341-7349

[65] Chen G, Choi W, Yu X, et al. Learning efficient object detection models with knowledge distillation. In: Neural Information Processing Systems (NIPS '17); 4-9 December 2017; Long Beach, CA; 2017. pp. 742-751

[66] Lin M, Chen Q, Yan S. Network in Network [Internet]. Available from: https://arxiv.org/pdf/1312.4400 [Accessed: May 12, 2018]

[67] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '17); 21-26 July 2017; Honolulu, Hawaii. New York: IEEE; 2017. pp. 1-9

[68] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '17); 21-26 July 2017; Honolulu, Hawaii. New York: IEEE; 2017. pp. 770-778

[69] Huang G, Liu Z, Weinberger K Q, et al. Densely connected convolutional networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '17); 21-26 July 2017; Honolulu, Hawaii. New York: IEEE; 2017. pp. 4700-4708

[70] Iandola FN, Han S, Moskewicz MW, et al. SqueezeNet: AlexNet-Level Accuracy with 50× Fewer Parameters and < 0.5 MB Model Size [Internet]. Available from: https://arxiv.org/pdf/1602.07360 [Accessed: May 12, 2018]

[71] Xie S, Girshick R, Dollár P, et al. Aggregated residual transformations for deep neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '17); 21-26 July 2017; Honolulu, Hawaii. New York: IEEE; 2017. pp. 5987-5995

[72] Howard AG, Zhu M, Chen B, et al. Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications [Internet]. Available from: https://arxiv.org/pdf/1704.04861 [Accessed: May 12, 2018]

[73] Zhang X, Zhou X, Lin M, et al. Shufflenet: An Extremely Efficient Convolutional Neural Network for Mobile Devices [Internet]. Available from: https://arxiv.org/pdf/1707.01083 [Accessed: May 12, 2018]

[74] Redmon J, Divvala S, Girshick R, Farhadi A. You only look once: Unified, real-time object detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '16); 27-30 June 2016; Las Vegas. Nevada State: IEEE; 2016. pp. 779-788

[75] Redmon J, Farhadi A. YOLO9000: Better, faster, stronger. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR '17); 21-26 July 2017; Honolulu, Hawaii. New York: IEEE; 2017. pp. 7263-7271

[76] Srivastava N, Hinton GE, Krizhevsky A, et al. Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research. 2014;**15**(1):1929-1958

[77] Wan L, Zeiler MD, Zhang S, et al. Regularization of neural networks using DropConnect. In: Proceedings of the 30th International Conference on Machine Learning (ICML '13); 16-21 June 2013; Atlanta, GA; 2013. pp. 1058-1066

[78] He K, Zhang X, Ren S, Sun J. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In: IEEE International Conference on Computer Vision (ICCV '15). 7-13 December 2015; Santiago, Chile. IEEE; 2015. pp. 1-9

[79] Ge S, Luo Z, Zhao S, et al. Compressing deep neural networks for efficient visual inference. In: IEEE International Conference on Multimedia and Expo (ICME '17). 10-14 July 2017; Hong Kong. IEEE: 2017. pp. 667-672

[80] Zeng D, Zhao F, Shen W, Ge S. Compressing and accelerating neural network for facial point localization. Cognitive Computation. 2018;**10**(2):359-367. DOI: 10.1007/s12559-017-9506-0

[81] Kim Y, Rush AM. Sequence-level knowledge distillation. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '16). 1-4 November 2016; Austin, Texas; 2016. pp. 1317-1327

[82] Lopez-Paz D, Bottou L, Schölkopf B, Vapnik V. Unifying distillation and privileged information. In: International Conference on Learning Representations (ICLR '16). 2-4 May 2016; San Juan; 2016. pp. 1-10

[83] Hu Z, Ma X, Liu Z, et al. Harnessing deep neural networks with logic rules. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL '16). 7-12 August 2016; Berlin, Germany; 2016. pp.1-11

[84] Luo Z, Jiang L, Hsieh JT, et al. Graph Distillation for Action Detection with Privileged Information [Internet]. Available from: https://arxiv.org/abs/1712.00108 [Accessed: December 30, 2017]

Chapter 7

Neural Network Principles and Applications

Amer Zayegh and Nizar Al Bassam

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.80416

Abstract

Due to the recent trend of intelligent systems and their ability to adapt with varying conditions, deep learning becomes very attractive for many researchers. In general, neural network is used to implement different stages of processing systems based on learning algorithms by controlling their weights and biases. This chapter introduces the neural network concepts, with a description of major elements consisting of the network. It also describes different types of learning algorithms and activation functions with the examples. These concepts are detailed in standard applications. The chapter will be useful for undergraduate students and even for postgraduate students who have simple background on neural networks.

Keywords: neural network, neuron, digital signal processing, training, supervised learning, unsupervised learning, classification, time series

1. Introduction

The artificial neural network is a computing technique designed to simulate the human brain's method in problem-solving. In 1943, McCulloch, a neurobiologist, and Pitts, a statistician, published a seminal paper titled "A logical calculus of ideas immanent in nervous activity" in Bulletin of Mathematical Biophysics [1], where they explained the way how brain works and how simple processing units—neurons—work together in parallel to make a decision based on the input signals.

The similarity between artificial neural networks and the human brain is that both acquire the skills in processing data and finding solutions through training [1].

2. Neural network's architecture

To illustrate the structure of the artificial neural network, an anatomical and functional look must be taken on the human brain first.

The human brain consists of about 10^{11} computing units "neurons" working in parallel and exchanging information through their connectors "synapses"; these neurons sum up all information coming into them, and if the result is higher than the given potential called action potential, they send a pulse via axon to the next stage. Human neuron anatomy is shown in **Figure 1** [2].

In the same way, artificial neural network consists of simple computing units "artificial neurons," and each unit is connected to the other units via weight connectors; then, these units calculate the weighted sum of the coming inputs and find out the output using squashing function or activation function. **Figure 2** shows the block diagram of artificial neuron.

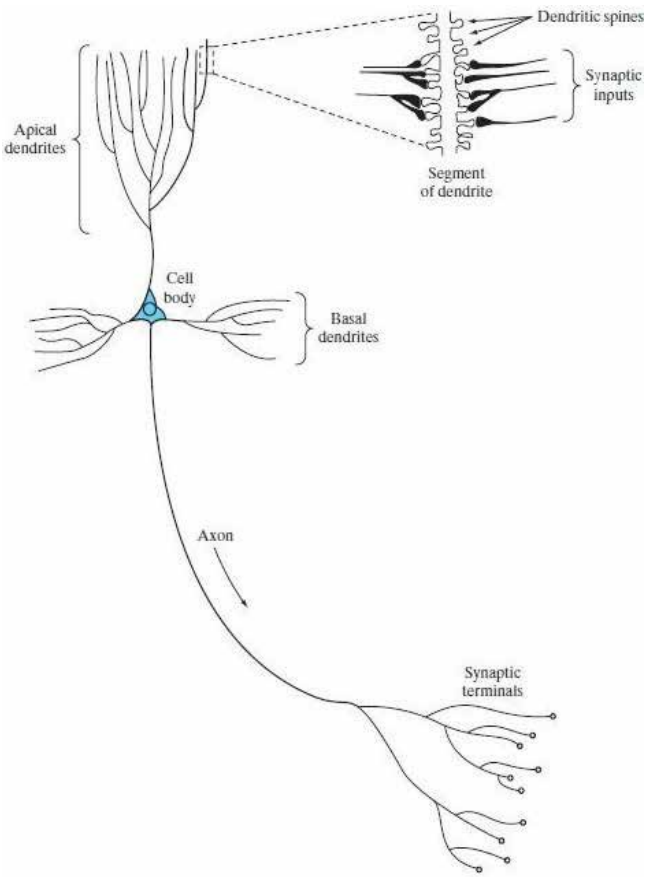

Figure 1. Human neuron anatomy.

http://dx.doi.org/10.5772/intechopen.80416

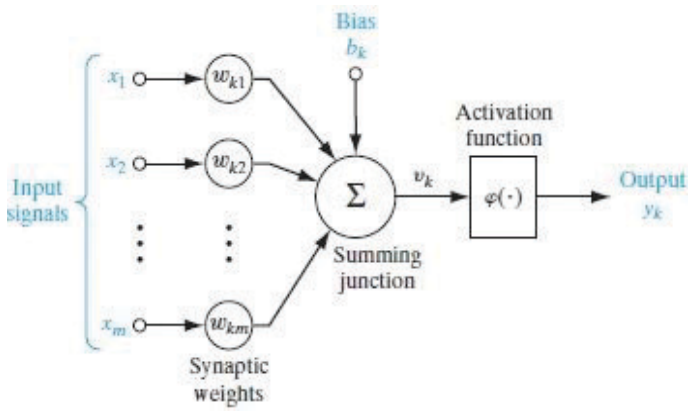

Figure 2. Block diagram of artificial neuron.

Based on the block diagram and function of the neural network, three basic elements of neural model can be identified:

1. Synapses, or connecting links, have a weight or strength where the input signal x_i connected to neuron k is multiplied by synaptic weight ww_{ki}.
2. An adder for summing the weighted inputs.
3. An activation function to produce the output of a neuron. It is also referred to as a squashing function, in that it squashes (limits) the amplitude range of the output signal to a finite value.

The bias b_k has the effect of increasing or decreasing the net input of the activation function, depending on whether it is positive or negative, respectively.

Mathematically, the output on the neuron k can be described as

$$y_k = \varphi\left(\sum_{i=1}^{m} x_i.w_{ki} + b_k\right) \tag{1}$$

where

$x_1, x_2, x_3, \ldots\ldots, x_m$ are the input's signals.

$w_{k1}, w_{k2}, w_{k3}, \ldots, w_{km}$ are the respective weights of neuron.

b_k is the bias.

φ is the activation function.

To clarify the effect of the bias on the performance of the neuron, the output given in Eq. (1) is processed in two stages, where the first stage includes the weighted inputs and the sum which is donated as S_k:

$$S_k = \sum_{i=1}^{m} x_i.w_{ki} \tag{2}$$

Then, the output of adder will be given in Eq. (3):

$$v_k = S_k + b_k \tag{3}$$

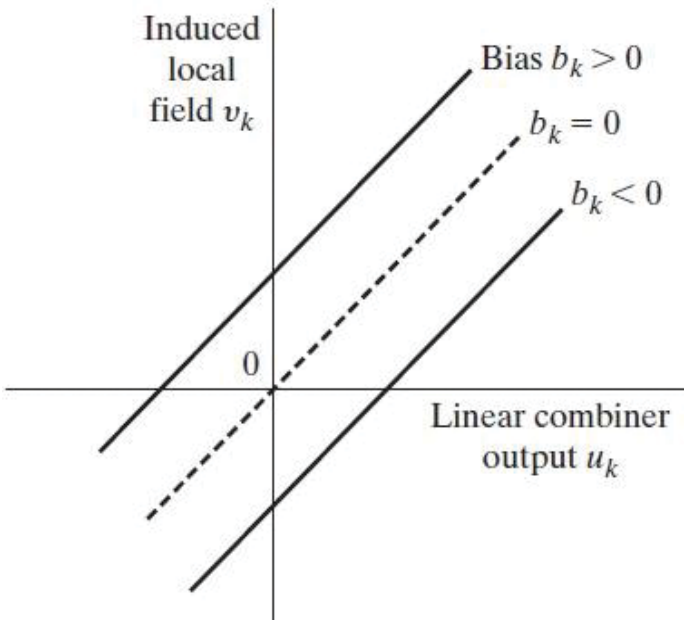

Figure 3. Effect of bias.

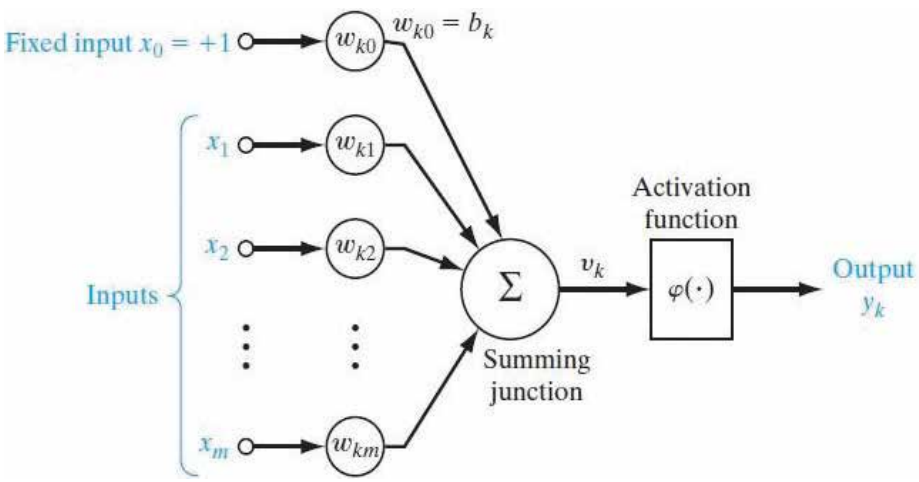

Figure 4. Neuron structure with considering bias as input [1].

http://dx.doi.org/10.5772/intechopen.80416

where the output of neuron will be

$$y_k = \varphi(v_k) \tag{4}$$

Depending on the value of the bias, the relationship between the weighted input and adder output will be modified [3] as shown in **Figure 3**.

Bias could be considered as an input signal x_0 fixed at +1 with synaptic weight equal to the bias b_k as shown in **Figure 4** [3].

3. Types of activation function

Activation function defines the output of neuron as the function to the adder's output v_k. The following sections describe the different activation functions:

3.1. Linear function

Where neuron output is proportional to the input as shown in **Figure 5**.

And, it can be described by

$$y_k = v_k \tag{5}$$

3.2. Threshold (step) function

This activation function is described in **Figure 6** where the output of neuron is given by

$$y_k = \begin{cases} 1 \ \ if \ v_k \geq 0 \\ 0 \ \ if \ v_k < 0 \end{cases} \tag{6}$$

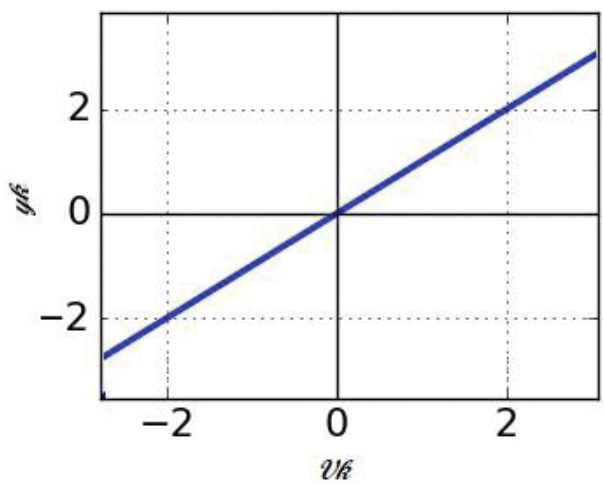

Figure 5. Linear activation function.

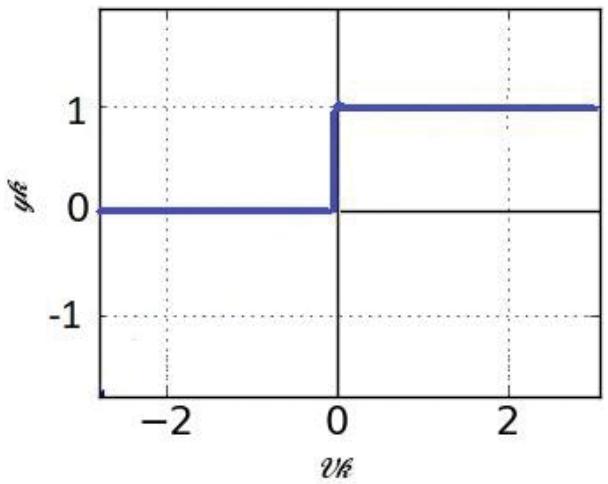

Figure 6. Threshold activation function.

In neural computation, such a neuron is referred to as the *McCulloch-Pitts model* in recognition of the pioneering work done by McCulloch and Pitts (1943); the output of the neuron takes on the value of 1 if the induced local field of that neuron is nonnegative and 0 otherwise. This statement describes the *all-or-none property* of the McCulloch-Pitts model [4].

3.3. Sigmoid function

The most common type of activation functions in neural network is described by

$$y_k = \frac{1}{1 + e^{v_k}} \tag{7}$$

Figure 7 shows the sigmoid activation function, it is clearly observed that this function has nonlinear nature and it can produce analogue output unlike threshold functions which produce output in discrete range [0, 1].

Also, we can note that sigmoid activation function is limited between 0 and 1 and gives an advantage over linear activation function which produces output form $-\infty$ to $+\infty$ [5].

3.4. Tanh activation function

This activation function has the advantages of sigmoid function, while it is characterized by output range between -1 and 1 as shown in **Figure 8**.

The output is described by

$$y_k = \frac{2}{1 + e^{-2v_k}} - 1 \tag{8}$$

http://dx.doi.org/10.5772/intechopen.80416

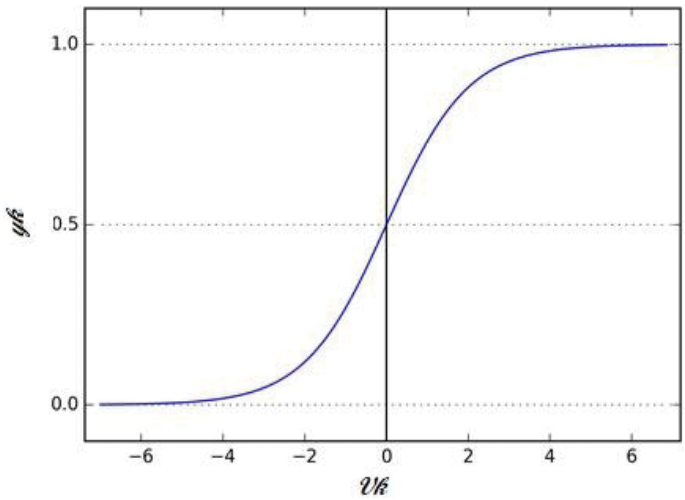

Figure 7. Sigmoid activation function.

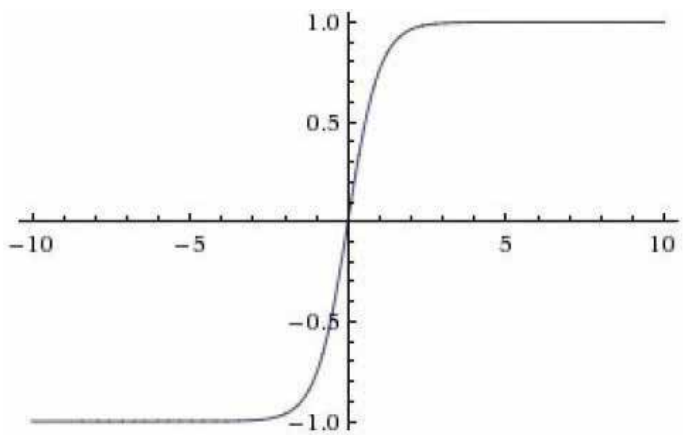

Figure 8. Tanh activation function.

4. Neural network models

The manner in which the neurons of a neural network are structured is intimately linked with the learning algorithm used to train the network [1]. Three main models can be identified for the neural network.

4.1. Single-layer feedforward neural network

In a layered neural network, the neurons are organized in the form of layers [1]. The simplest structure is the single-layer feedforward network that consists of input nodes connected

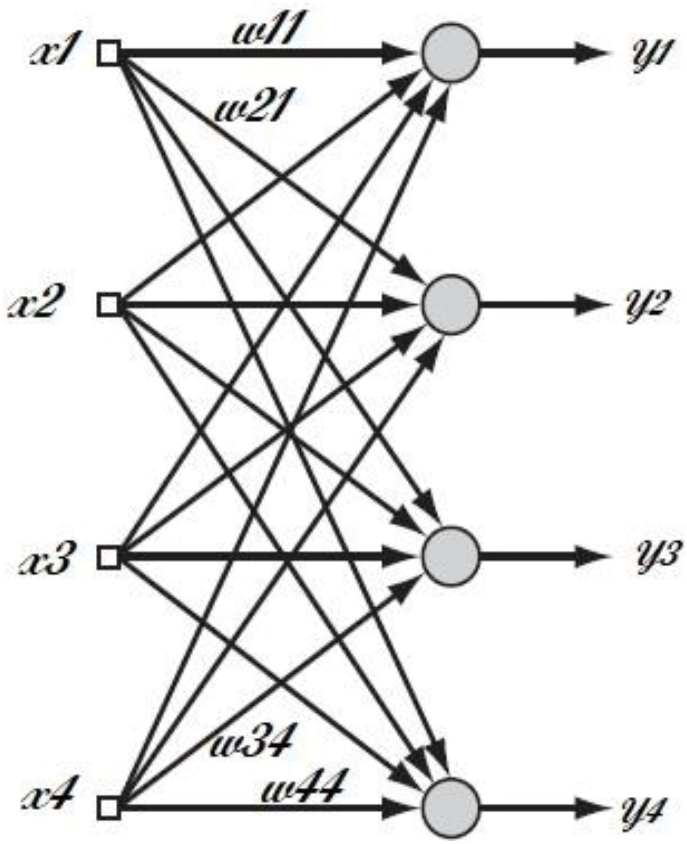

Figure 9. Single-layer neural network.

directly to the single layer of neurons. The node outputs are based on the activation function as shown if **Figure 9**.

Mathematically, the inputs will be presented as vectors with dimensions of $1 \times i$, while the weights will be presented as a matrix with dimensions of $i \times k$, and outputs will be presented as a vector with dimensions of $1 \times k$ as given in Eq. (9):

$$\left[y_1,\ y_2, \ldots,\ y_k\right] = \left[x_1,\ x_2, \ldots, x_i\right] \left[\begin{bmatrix} w_{11} & w_{21} & \cdots & w_{k1} \\ \vdots & & \ddots & \vdots \\ w_{1k} & w_{2k} & \cdots & w_{ik} \end{bmatrix}\right] \tag{9}$$

4.2. Multilayer feedforward neural network

The second class of a feedforward neural network distinguishes itself by the presence of one or more hidden layers, whose computation nodes are correspondingly called hidden neurons as shown in **Figure 10**.

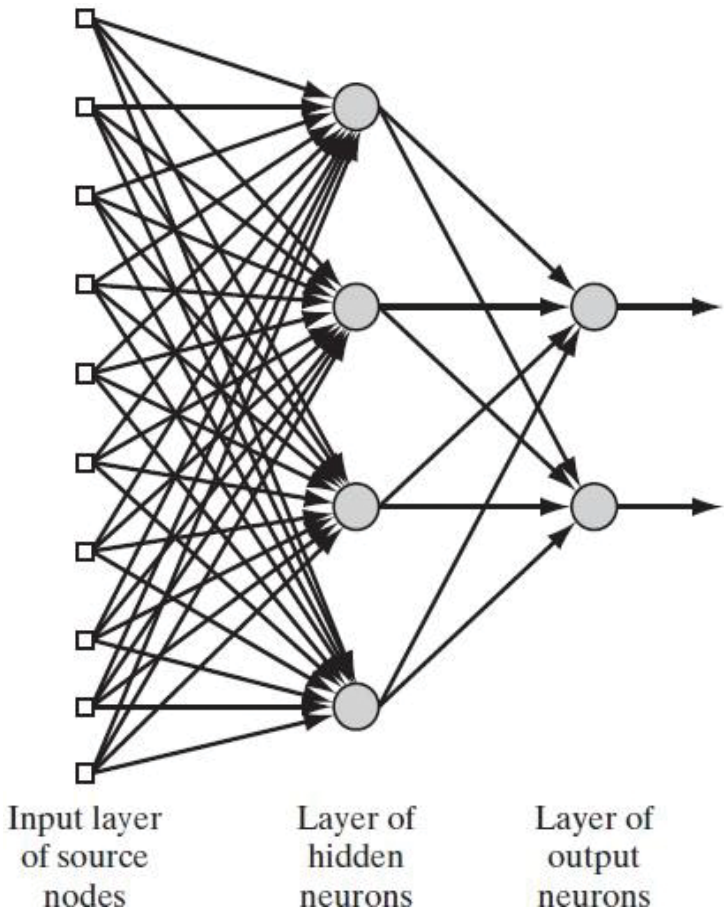

Figure 10. Multilayer feedforward neural network.

By adding one or more hidden layers, the network is enabled to extract higher-order statistics from its input [1].

5. Neural network training

The process of calibrating the values of weights and biases of the network is called training of neural network to perform the desired function correctly [2].

Learning methods or algorithms can be classified into:

5.1. Supervised learning

In supervised learning, the data will be presented in a form of couples (input, desired output), and then the learning algorithm will adapt the weights and biases depending on the error signal between the real output of network and the desired output as shown in **Figure 11**.

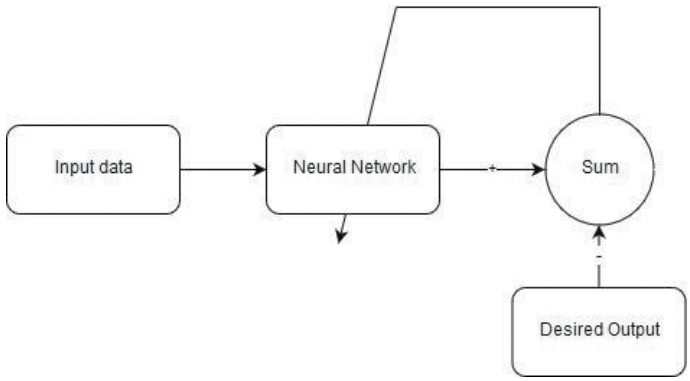

Figure 11. Supervised learning.

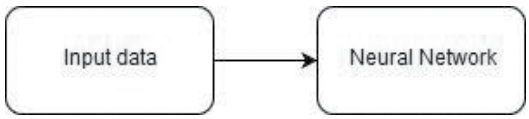

Figure 12. Unsupervised learning.

As a performance measure for the system, we may think in terms of the mean squared error or the sum of squared errors over the training sample defined as a function of the free parameters (i.e., synaptic weights) of the system [1].

5.2. Unsupervised learning

To perform unsupervised learning, a competitive learning rule is used. For example, we may use a neural network that consists of two layers—an input layer and a competitive layer. The input layer receives the available data. The competitive layer consists of neurons that compete with each other (in accordance with a learning rule) for the "opportunity" to respond to features contained in the input data (**Figure 12**) [1].

6. Neural networks' applications in digital signal processing

Digital signal processing could be defined using field of interest statement of the IEEE Signal Processing Society as follows:

Signal processing is the enabling technology for the generation, transformation, extraction, and interpretation of information. It comprises the theory, algorithms with associated architectures and implementations, and applications related to processing information contained in many different formats broadly designated as signals. Signal processing uses mathematical, statistical,

computational, heuristic, and/or linguistic representations, formalisms, modelling techniques and algorithms for generating, transforming, transmitting, and learning from signals. [6].

Based on this definition, many neural network structures could be developed to achieve the different processes mentioned in the definition.

6.1. Classification

One of the most important applications of an artificial neural network is classification, which can be used in different digital signal processing applications such as speech recognition, signal separation, and handwriting recognition and detection [7].

The objects of interest can be classified according to their features, and classification process could be considered as probability process, since the classification of any object under a given class depends on the likelihood that the object belongs to the class more than the probability of belonging to the other classes [8].

Assume that $\boldsymbol{X}$ is the vector of features for the objects of interest which could be classified into classes $c \in \psi$ where ψ is the pool of classes. Then, classification will be applied as follows:

$$\boldsymbol{X} \text{ belongs to the class } c_i \text{ if } P(c_i|\boldsymbol{X}) > P\left(C_j|\boldsymbol{X}\right) \text{ when } i \neq j \tag{10}$$

To decrease the difficulty of solving probability equations in Eq. (10), discriminant function is used, and then Eq. (10) will be.

$$Q_i(\boldsymbol{X}) > Q_j(\boldsymbol{X}) \quad \textit{if } (c_i|\boldsymbol{X}) > P\left(C_j|\boldsymbol{X}\right) \text{when } i \neq j \tag{11}$$

Classification process will be described using Eq. (12)

$$\boldsymbol{X} \text{ belongs to the class } c_i \text{ if } \; Q_i(\boldsymbol{X}) > Q_j(\boldsymbol{X}) \tag{12}$$

One of the examples of classification is QPSK modulator output detection, where detection is considered as a special case of classification.

Assume that the received signal is X:

$$X = s + n \tag{13}$$

where n is normally the distributed noise signal and s is the transmitted signal.

The output of QPSK modulator is shown in **Figure 13**, where the samples are arranged in four classes.

By adding white Gaussian noise, the received signal will be as shown in **Figure 14**.

The neural network shown in **Figure 15** is used to detect and demodulate the received signal, where the network consists of one hidden layer with five neurons and an output layer with two neurons.

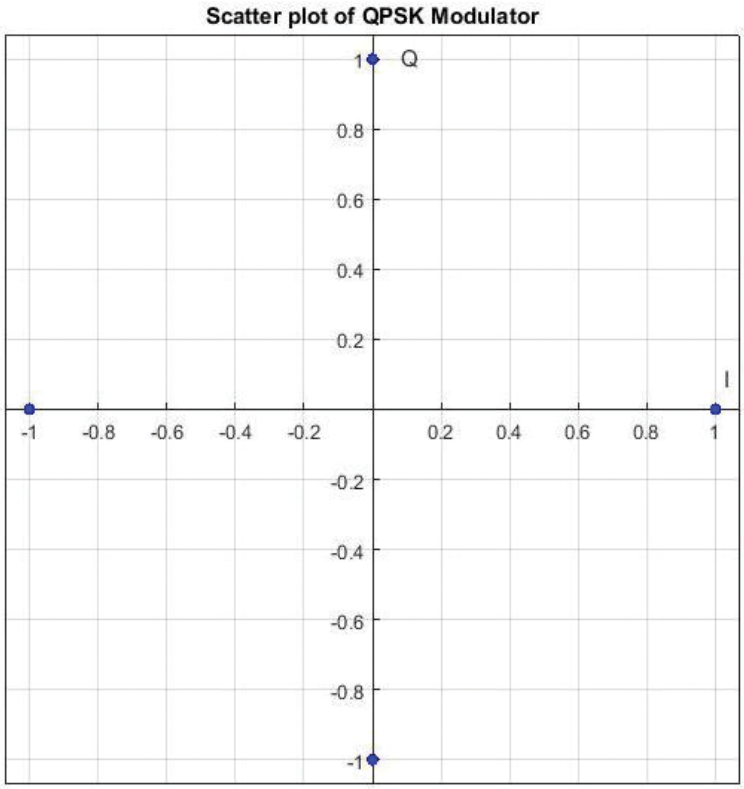

Figure 13. QPSK modulator output.

Figures 16 and **17** show the performance of neural network evaluated using mean squared error (MSE) criteria.

6.2. Time series prediction

A series is a sequence of values as a function of parameter; in the case of time series, the values will be as a function of the time. So, many applications use time series to express their data, for example, metrology, where the temperature is described as time series [7].

The interesting problem in time series is the future prediction of the series values; neural networks can be used to predict the future results in series in three ways [9]:

- Predict the future values based on the past values of the same series; this way can be described by

$$\hat{y}(t) = E\{y(t)|y(t-1), y(t-2), \ldots\} \tag{14}$$

- Predict the future values based on the values of relevant time series, where

http://dx.doi.org/10.5772/intechopen.80416

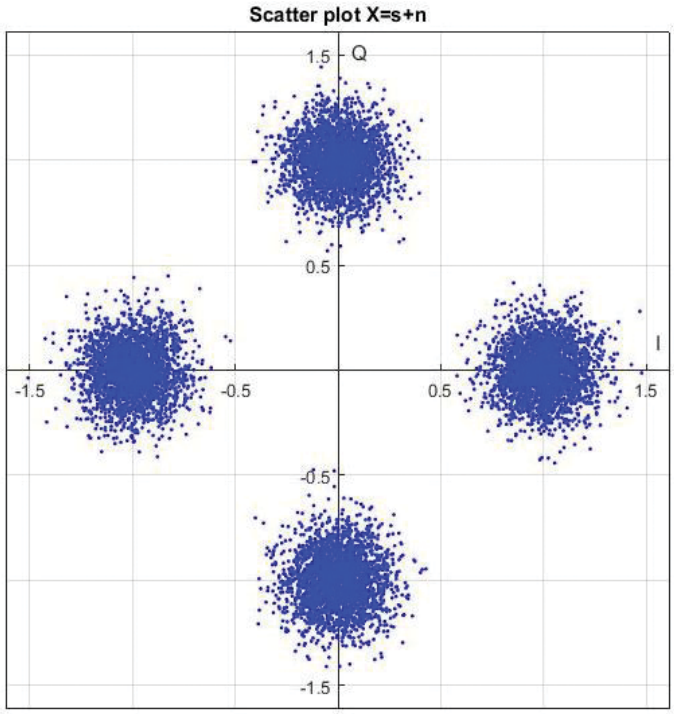

Figure 14. QPSK output with noise.

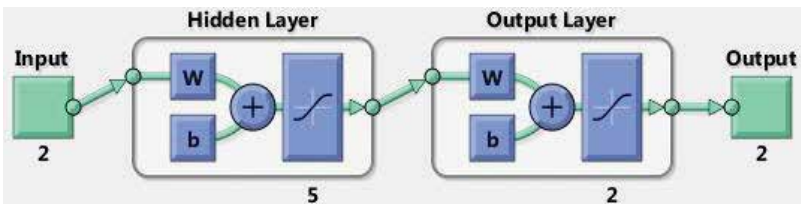

Figure 15. Structure of neural network.

$$\hat{y}(t) = E\{y(t)|x(t), x(t-1), x(t-2), \ldots\} \tag{15}$$

- Predict the future values based on both previous cases, where

$$\hat{y}(t) = E\{y(t)|x(t), x(t-1), x(T-2), \ldots, y(t-1), y(t-2), \ldots\} \tag{16}$$

Figure 18 shows the predicted series by neural network based on the first way, where the samples of original series are given of determined period, and then the neural network predicts the future values of the series based on series behavior.

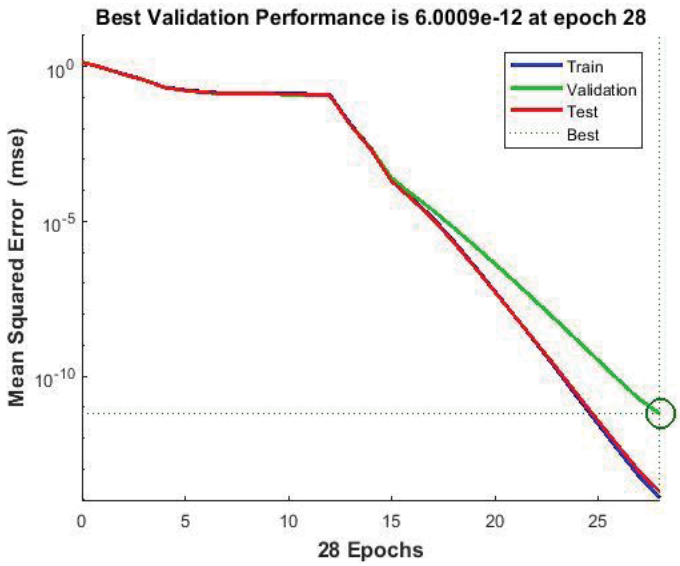

Figure 16. MSE of training, validation, and test vs no. of epochs.

6.3. Independent component analysis

The goal of the independent component analysis (ICA) is to separate the linearly mixed signals. ICA is a type of blind source separation when the separation is performed without the pre-information about the source of signals or the signal-mixing coefficients. Although the problem of separating the blind source, in general, is not specified, the solution of use can be obtained under some assumptions [10].

ICA model assumes that n independent signals $s_i(t)$ where $i = 1, 2, 3, \ldots, n$ are mixed using matrix:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \tag{17}$$

Then, mixed signal $x_i(t)$ could be expressed as

$$x_i(t) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} s_j(t) \tag{18}$$

As the separation process is blind, that is, both a_{ij} and $s_j(t)$ are unknown; thus, ICA assumes that the mixed signals are statistically independent and have non-Gaussian distribution [11].

Neural network shown in **Figure 19** is used to estimate the unmixing matrix W.

http://dx.doi.org/10.5772/intechopen.80416

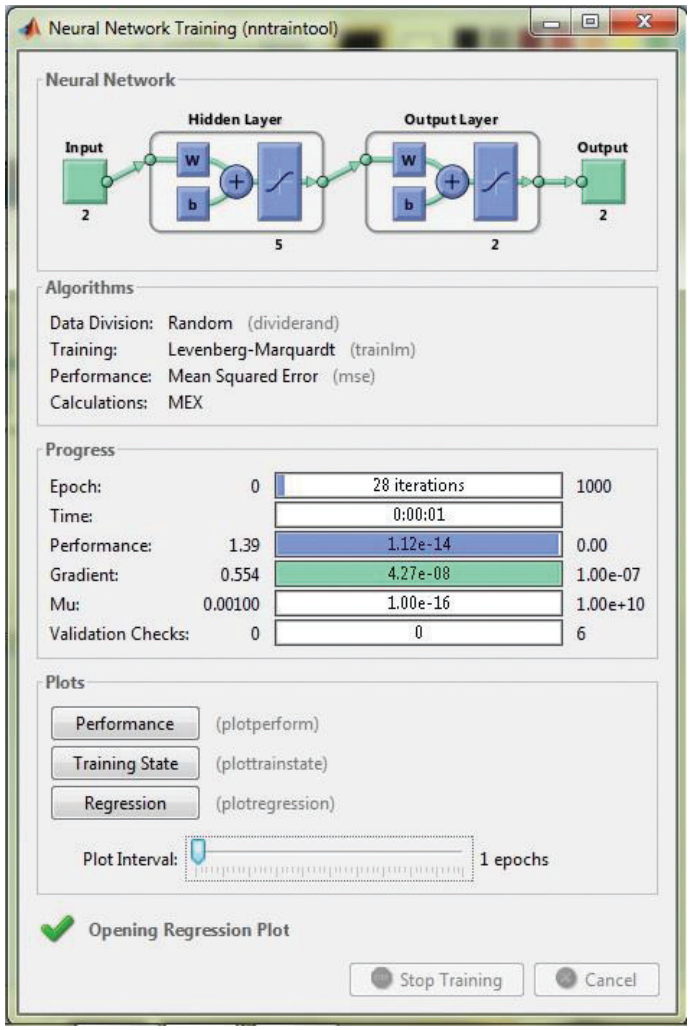

Figure 17. Training parameters and results.

The separated signals $y_i(t)$ are given as

$$y = Wx \tag{19}$$

$$W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \cdots & \ddots \end{bmatrix} \tag{20}$$

Different methods could be applied to find W, for example, natural gradient based define W as

$$\frac{dW}{dt} = \eta(t)\left[1 - f(y(t))g^T(y(t))\right]W \tag{21}$$

where $\eta(t)$ is the training factor and both f and g are the odd functions.

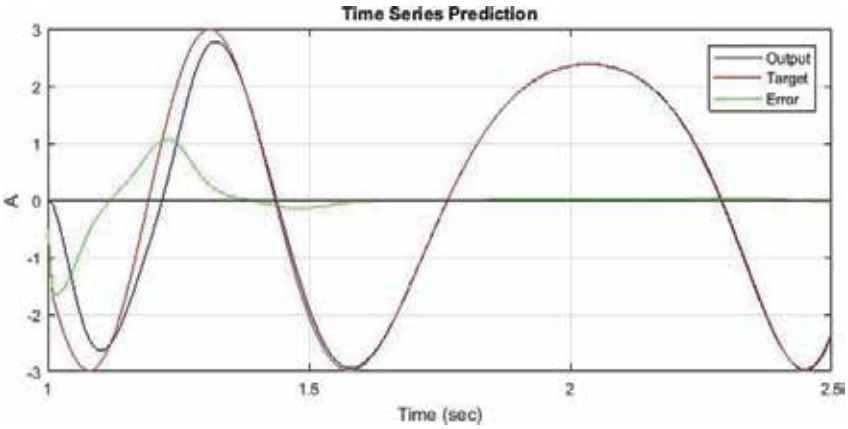

Figure 18. Time series prediction.

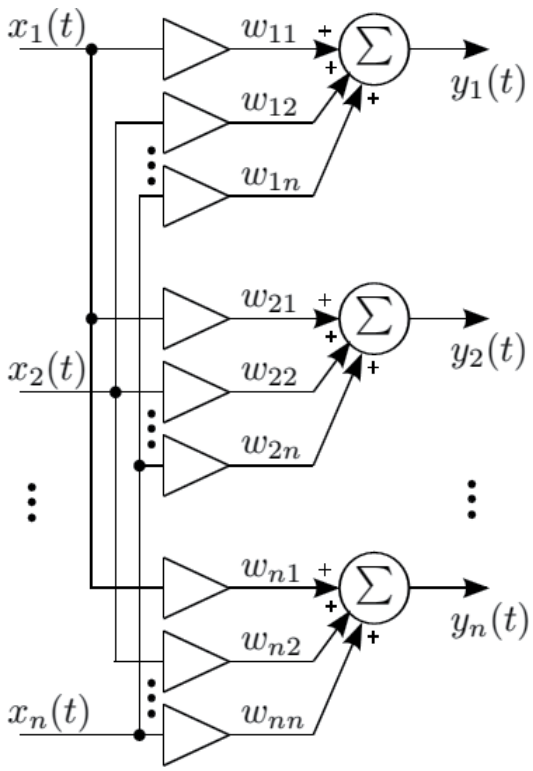

Figure 19. ICA neural network [12].

Author details

Amer Zayegh and Nizar Al Bassam*

*Address all correspondence to: nizar@mec.edu.om

Middle East College, Muscat, Oman

References

[1] Haykin S. Neural Networks and Learning Machines. 3rd ed. Hamilton, Ontario, Canada: Pearson Education, Inc.; 2009. 938 p

[2] Smith S. The Scientist and Engineer's Guide to Digital Signal Processing. 2011

[3] Hu Y, Hwang J. Handbook of Neural Network Signal Processing. Boca Raton: CRC Press; 2002

[4] Milad MAMRAN. Neural Network Demodulator For. International Journal of Advanced Studies. 2016;**5**(7):10-14

[5] Jan Michalík. Applied Neural Networks for Digital Signal Processingwith DSC TMS320 F28335. Technical Univerzity of Ostrava; 2009

[6] Constitution. IEEE Signal Processing Society [Online]. 2018. Available from: https://signalprocessingsociety.org/volunteers/constitution [Accessed: March 03, 2018]

[7] Kriesel D. A Brief Introduction to Neural Network. Bonn: University of Bonn in Germany; 2005

[8] Gurney K. An Introduction to Neural Networks. London: CRC Press; 1997

[9] Maier H, Dandy G. Neural networks for the prediction and forecasting of water resources variables: A review of modelling issues and applications. Environmental Modelling & Software. 2000;**15**(1):101-124

[10] Hansen LK, Larsen J, Kolenda T. On Independent Component Analysis for Multimedia Signals, Multimedia Image and Video Processing. CRC Press; 2000. pp. 175-199

[11] Mørup M, Schmidt MN. Transformation invariant sparse coding, Machine Learning for Signal Processing, IEEE International Workshop on (MLSP), Informatics and Mathematical Modelling. Technical University of Denmark, DTU; 2011

[12] Pedersen MS, Wang D, Larsen J, Kjems U. Separating Underdetermined Convolutive Speech Mixtures, ICA2006. 2006

Chapter 8

Applications of General Regression Neural Networks in Dynamic Systems

Ahmad Jobran Al-Mahasneh, Sreenatha Anavatti, Matthew Garratt and Mahardhika Pratama

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.80258

Abstract

Nowadays, computational intelligence (CI) receives much attention in academic and industry due to a plethora of possible applications. CI includes fuzzy logic (FL), evolutionary algorithms (EA), expert systems (ES) and artificial neural networks (ANN). Many CI components have applications in modeling and control of dynamic systems. FL mimics the human reasoning by converting linguistic variables into a set of rules. EA are metaheuristic population-based algorithms which use evolutionary operations such as mutation, crossover, and selection to find an optimal solution for a given problem. ES are programmed based on an expert knowledge to make informed decisions in complex tasks. ANN models how the neurons are connected in animal nervous systems. ANN have learning abilities and they are trained using data to make intelligent decisions. Since ANN have universal approximation abilities, they can be used to solve regression, classification, and forecasting problems. ANNs are made of interconnected layers where every layer is made of neurons and these neurons have connections with other neurons. These layers consist of an input layer, hidden layer/layers, and an output layer.

Keywords: applications, general regression, neural networks, dynamic systems

1. Introduction

Nowadays, computational intelligence (CI) receives much attention in academic and industry due to a plethora of possible applications. CI includes fuzzy logic (FL), evolutionary algorithms (EA), expert systems (ES), and artificial neural networks (ANN). Many CI components have applications in modeling and control of dynamic systems. FL mimics the human reasoning by converting linguistic variables into a set of rules. EA are metaheuristic population-based

algorithms which use evolutionary operations such as mutation, crossover, and selection to find an optimal solution for a given problem. ES are programmed based on an expert knowledge to make informed decisions in complex tasks. ANN model how the neurons are connected in animal nervous systems. ANN have learning abilities and they are trained using data to make intelligent decisions. Since ANN have universal approximation abilities [1], they can be used to solve regression, classification, and forecasting problems. ANNs are made of interconnected layers where every layer is made of neurons, and these neurons have connections with other neurons. These layers consist of an input layer, hidden layer/layers, and an output layer. ANN have two major types as shown in **Figure 1**: feed-forward neural network (FFNN) and recurrent neural network (RNN). In FFNN, the data can only flow from the input to hidden layer, while in RNN, the data can flow in any direction. The output of a single-hidden-layer FFNN can be written as

$$Y = (W_{HO}\, h(\mathbf{x}\, W_{IH} + \mathbf{b_I})) + \mathbf{b_O} \tag{1}$$

where Y is the network output, W_{HO} is the hidden-output layers weights matrix, h is the hidden layer activation function, $\mathbf{x}$ is the input vector, W_{IH} is the input-hidden layers weights matrix, $\mathbf{b_I}$ is the input layer bias vector, and $\mathbf{b_O}$ is the hidden layer bias vector.

The output of a single-hidden-layer RNN with a recurrent hidden layer can be written as

$$Y = (W_{HO}\, h(\mathbf{x}\, W_{IH} + \mathbf{h_{t-1}}\, W_{HH} + \mathbf{b_I})) + \mathbf{b_O} \tag{2}$$

The training of neural networks involves modifying the neural network parameters to reduce a given error function. Gradient descent (GD) [2, 3] is the most common ANN training method:

$$\theta_{new} = \theta_{old} - \lambda \frac{\partial E}{\partial \theta} \tag{3}$$

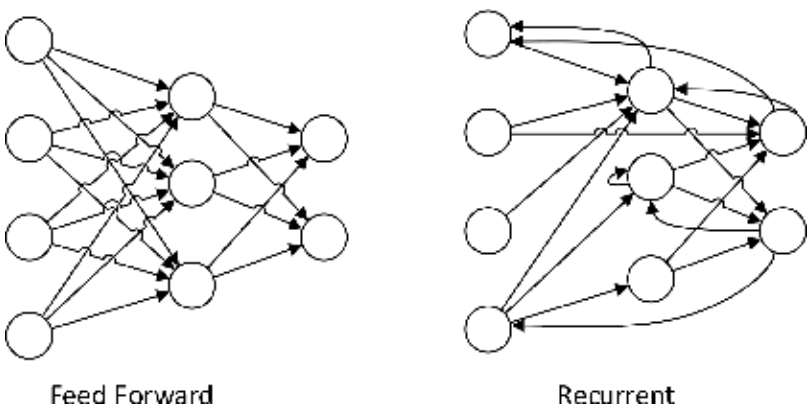

Figure 1. Feed-forward and recurrent networks.

where θ are the network parameters, λ is the learning rate, and E is the error function:

$$E = \frac{1}{N}\sum_{I=1}^{N}(y-t)^2 \tag{4}$$

where N is the number of samples, y is the network output, and t is the network target.

2. General regression neural network (GRNN)

The general regression neural network (GRNN) is a single-pass neural network which uses a Gaussian activation function in the hidden layer [4]. GRNN consists of input, hidden, summation, and division layers.

The regression of the random variable y on the observed values X of random variable x can be found using

$$E[y|X] = \frac{\int_{-\infty}^{\infty} yf(X,y)dy}{\int_{-\infty}^{\infty} f(X,y)dy} \tag{5}$$

where $f(X,y)$ is a known joint continuous probability density function.

When $f(X,y)$ is unknown, it should be estimated from a set of observations of x and y. $f(X,y)$ can be estimated using the nonparametric consistent estimator suggested by Parzen as follows:

$$\hat{f}(X,Y) = \frac{1}{2\pi^{(p+1)/2}\,\sigma^{(p+1)}}\frac{1}{n}\sum_{i=1}^{n} e^{-\frac{\left(X-X^i\right)^T\left(X-X^i\right)}{2\sigma^2}} e^{-\frac{\left(Y-Y^i\right)^2}{2\sigma^2}} \tag{6}$$

where n is the number of observations, p is the dimension of the vector variable, and x and σ are the smoothing factors.

Substituting (6) into (5) leads to

$$\hat{Y}(X) = \frac{\sum_{i=1}^{n} e^{\frac{-\left(X-X^i\right)^T\left(X-X^i\right)}{2\sigma^2}} \frac{\int_{-\infty}^{\infty} ye^{-\frac{\left(Y-Y^i\right)^2}{2\sigma^2}}dy}{\int_{-\infty}^{\infty} e^{-\frac{\left(Y-Y^i\right)^2}{2\sigma^2}}dy}}{\sum_{i=1}^{n} e^{\frac{-\left(X-X^i\right)^T\left(X-X^i\right)}{2\sigma^2}}} \tag{7}$$

After solving the integration, the following will result:

$$\hat{Y}(X) = \frac{\sum_{i=1}^{n} ye^{\frac{-\left(X-X^i\right)^T\left(X-X^i\right)}{2\sigma^2}}}{\sum_{i=1}^{n} e^{\frac{-\left(X-X^i\right)^T\left(X-X^i\right)}{2\sigma^2}}} \tag{8}$$

2.1. Previous studies

GRNN was used in different applications related to modeling, system identification, prediction, and control of dynamic systems including: feedback linearization controller [5], HVAC

process identification and control [6], modeling and monitoring of batch processes [7], cooling load prediction for buildings [8], fault diagnosis of a building's air handling unit [9], intelligent control [10], optimal control for variable-speed wind generation systems [11], annual power load forecasting model [12], vehicle sideslip angle estimation [13], fault diagnosis for methane sensors [14], fault detection of excavator's hydraulic system [15], detection of time-varying inter-turn short circuit in a squirrel cage induction machine [16], system identification of nonlinear rotorcraft heave mode [17], and modeling of traveling wave ultrasonic motors [18].

Some significant modifications of GRNN include using fuzzy c-means clustering to cluster the input data of GRNN [19], modified GRNN which uses different types of Parzen estimators to estimate the density function of the regression [20], density-driven GRNN combining GRNN, density-dependent kernels and regularization for function approximation [21], GRNN to model time-varying systems [22], adapting GRNN for modeling of dynamic plants [23] using different adaptation approaches including modifying the training targets, and adding a new pattern and dynamic initialization of σ.

2.2. GRNN training algorithm

GRNN training is rather simple. The input weights are the training inputs transposed, and the output weights are the training targets. Since GRNN is an associative memory, after training, the number of the hidden neurons is equal to the number of the training samples. However, this training procedure is not efficient if there are many training samples, so one of the suggested solutions is using a data dimensionality reduction technique such as clustering or principal component analysis (PCA). One of the novel solutions to data dimensionality reduction is using an error-based algorithm to grow GRNN [24] as explained in **Algorithm 1**. The algorithm will check whether an input is required to be included in the training, based on prediction error before training GRNN with that input. If the prediction error without including that input is more than the certain level, then GRNN should be trained with it.

Algorithm 1 GRNN growing algorithm

1: Train GRNN with 10% of the training data
2: **for** $i \le i_{final}$ **do**
3: Find the output of GRNN y_i of input$_i$
4: Find the MSE: $\frac{1}{2}(y_i - Target_i)^2$
5: **if** MSE>Threshold **then**
6: Train GRNN with the input$_i$ Do not train GRNN with the input$_i$
7: **end if**
8: **end if**

Dataset	Training error after/before k-means MSE	Testing error after/before k-means MSE	Size reduction %
Abalone	0.0177/0.002	0.0141/0.006	99.76
Building energy	0.047/3.44e-05	0.0165/0.023	99.76
Chemical sensor	0.241/0.016	0.328/0.034	97.99
Cholesterol	0.050/4.605e-05	0.030/0.009	92

Table 1. Using GRNN with k-means clustering.

2.2.1. Reducing data dimensionality using clustering

Clustering techniques can be used to reduce the data dimensionality before feeding it to the GRNN. k-means clustering is one of the popular clustering techniques. The k-means clustering algorithm is explained in **Algorithm 2**. Also, results of comparing GRNN performance before and after applying k-means algorithm are shown in **Table 1**. Although the training and testing errors will increase, there are large reductions in the network size.

The aim of the algorithm is to minimize the distance objective function:

$$J = \sum_{i=1}^{N} \sum_{j=1}^{M} \left\| x_i - c_j \right\|^2 \tag{9}$$

Algorithm 2 : K-means clustering algorithm

1: Randomly select the number of clusters k and their centers c_i

2: **for** $i \le i_{final}$ **do**

3: Calculate the distance between the data point and the clusters centers c_i.

4: Assign every data sample to its closest cluster.

5: Calculate the clusters centers c_i again.

6: **end for**

2.2.2. Reducing data dimensionality using PCA

PCA can be used to reduce a large dataset into a smaller dataset which still carries most of the important information from the large dataset. In a mathematical sense, PCA converts a number of correlated variables into a number of uncorrelated variables. PCA algorithm is explained in **Algorithm 3**.

2.3. GRNN output algorithm

After GRNN is trained, the output of GRNN can be calculated using

Algorithm 3 : PCA algorithm

1: Find the dot product matrix $X^T X = \sum_{i=1}^{N} (X_i - \mu)^T (X_i - \mu)$

2: Find the eigenvalues of $X^T X = XV\Lambda^T$

3: Find the eigenvectors $U = XV\Lambda^{-1/2}$

4: Find d number of principal components $U_d = [u_1, u_2, .. u_d]$

5: Compute $Y = U_d^T X$

6: Find the covariance matrix of Y as $YY^T = U^T XX^T U = \Lambda$

7: Find $W = U\Lambda^{-1/2}$

Dataset	Training error after/before PCA MSE	Testing error after/before PCA MSE	Size reduction %
Abalone	0.197/0.002	0.188/0.006	99.8
Building energy	0.061/3.44e-05	0.049/0.023	99.6
Chemical sensor	0.241/0.016	0.328/0.034	98.3
Cholesterol	0.026/4.605e-05	0.028/0.009	92

Table 2. Using GRNN with PCA.

$$D = (X - W_i)^T (X - W_i) \tag{10}$$

$$\hat{Y} = \frac{\sum_{i=1}^{N} W_o e^{(D/2\sigma^2)}}{\sum_{i=1}^{N} e^{(D/2\sigma^2)}} \tag{11}$$

where D is the Euclidean distance between the input X and the input weights W_i, W_o is the output weight, and σ is the smoothing factor of the radial basis function.

GRNN output calculation is explained in **Algorithm 4**.

Algorithm 4 GRNN output

1: **procedure** $\hat{Y}$ X

2: Find the Eucldian distance D using (10)

3: Find the numerator of (11)

4: Find the denominator (11)

5: Divide the numerator over denominator

6: **return** $\hat{Y}$

7: **end procedure**

Other distance measures can be also used such as Manhattan (city block), so (10) will become

$$D = X - W_i \tag{12}$$

3. Estimation of GRNN smoothing parameter (σ)

Since σ is the only free parameter in GRNN and suitable values of it will improve GRNN accuracy, it should be estimated. Since there is no optimal analytical solution for finding σ, numerical approaches can be used to estimate it. The holdout method is one of the suggested methods. In this method, samples are randomly removed from the training dataset; then using the GRNN with a fixed σ, the output is calculated using the removed samples; then the error is calculated between the network outputs and the sample targets. This procedure is repeated for different σ values. The smoothing parameter (σ) with the lowest sum of errors is selected as the best σ. The holdout algorithm is explained in **Algorithm 5**.

Algorithm 5 Holdout method to find GRNN σ

1: Initialize GRNN σ vector $[\sigma_{low}:\Delta\sigma:\sigma_{up}]$
2: **for** $i \le i_{final}$ **do**
3: **for** $j \le j_{final}$ **do**
4: Randomly select datasample (data (j)) and remove it from the training data
5: Train GRNN with the reduced dataset
6: Find GRNN output $y(j)$ using the reduced sample (data (j))
7: Find GRNN $MSE(j)$ between $y(j)$ and the reduced sample data target
8: **end for**
9: Find the summation of the MSE
10: Find σ with the lowest summation of the MSE
11: **end for**

Other search and optimization methods might be also used to find σ. For instance, genetic algorithms (GA) and differential evolution (DE) are suitable options. **Algorithm 6** explains how to find σ using DE or GA. Also, the results of using DE and GA are depicted in **Figure 2**.

Algorithm 6 DE/GA find GRNN σ

1: Divide the data into training dataset and testing dataset
2: Train GRNN with the training data
3: **for** $i \le maxiterations$ **do**
4: Generate a random population of σs
5: Evaluate the cost of each *sigma*
6: Perform mutation, crossover and selection
7: Find the best σ with the lowest MSE
8: **end for**

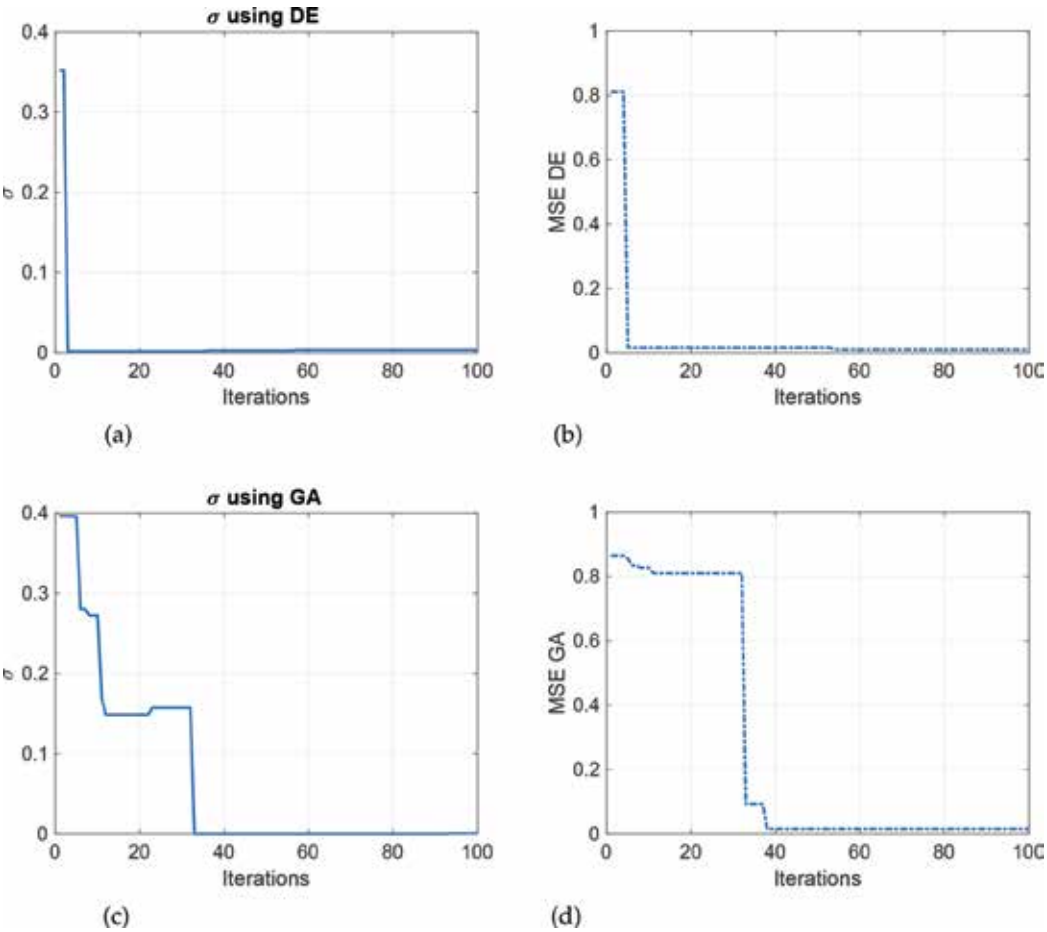

Figure 2. DE and GA used to estimate GRNN σ. (a) Estimation of σ using DE (b) MSE evolution when using DE to estimate s (c) Estimation of σ using GA (d) MSE evolution when using GA to estimate σ.

Both of GA and DE can find a good approximation of σ within 100 iterations only; however, DE converges faster since it is a vectorized algorithm.

4. GRNN vs. back-propagation neural networks (BPNN)

There are many differences between GRNN and BPNN. Firstly, GRNN is single-pass learning algorithm, while BPNN needs two passes: forward and backward pass. This means that GRNN consumes significantly less training time. Secondly, the only free parameter in GRNN is the smoothing parameter σ, while in BPNN more parameters are required such as weights, biases, and learning rates. This also indicates that GRNN quick learning abilities and its suitability for online systems or for system where minimal computations are required. Also, another difference is that since GRNN is an autoassociative memory network, it will store all the distinct input/output samples while BPNN has a limited predefined size. This size growth

Type	Dataset	Training time (sec)	Training error (MSE)	Testing error (MSE)
GRNN	Abalone	0.621	0.342	0.384
BPNN	Abalone	1.323	0.436	0.395
GRNN	Building energy	0.630	0.0731	0.628
BPNN	Building energy	1.880	0.1152	0.631
GRNN	Chemical sensor	0.701	0.888	1.316
BPNN	Chemical sensor	1.473	0.228	1.584
GRNN	Cholesterol	0.801	0.037	0.172
BPNN	Cholesterol	2.099	0.061	0.215

Table 3. GRNN vs. BPNN training and testing performance.

issue is resolved by either using clustering or PCA (read Sections 2.21 and 2.2.2). Finally, GRNN is based on the general regression theory, while BPNN is based on gradient-descent iterative optimization method.

To show the advantages of GRNN over BPNN, a comparison is held using standard regression datasets built inside MATLAB software [25]. For all the datasets, they are divided 70% for training and 30% for testing. After training the network with the 70% training data, the output of the neural network is found using the remaining testing data. The most notable advantage of GRNN over BPNN is the shorter training time which confirms its selection for dynamic systems modeling and control. Also, GRNN has less testing error which means it has better generalization abilities than BPNN. The comparison results are summarized in **Table 3**.

5. GRNN in identification of dynamic systems

System identification is the process of building a model of unknown/partially known dynamic system based on observed input/output data. Gray-box and black-box identification are two common approaches of system identification. In the gray-box approach, a nominal model of a dynamic system is known, but its exact parameters are unknown, so an identifier is used to find these parameters. In the black-box approach, the identification is based only on the data. Examples of black-box identification include fuzzy logic (FL) and neural networks (NN). GRNN can be used to identify dynamic systems quickly and accurately. There are two methods to use GRNN for system identification: the batch mode (off-line training) and sequential mode (online training). In the batch mode, all the observed data is available before the system identification, so GRNN can be trained with a big chunk of the data, while in the sequential mode only a few data samples are available for identification.

5.1. GRNN identification in batch training mode

In the batch mode, the observed data should be divided into training, validation, and testing. GRNN will be fed with all the training data to identify the system. Then in the validation stage, the network should be tested with different data, usually randomly selected, and the error is

recorded for every validation test. Then the validation process is repeated several times. Usually 10 times is standard. And then the average validation error is found based on all the validation tests. This validation procedure is called k-fold cross validation a standard technique in machine learning (ML) applications. To test the generalization ability of an identified model, a new dataset is used called testing dataset. Based on the model performance in the testing stage, one can decide whether the model is suitable or not.

5.1.1. Batch training GRNN to identify hexacopter attitude dynamics

In this example, GRNN is used to identify the attitude (pitch/roll/yaw) of a hexacopter drone based on real flight test data in the free flight mode. The data consist of three inputs: rolling, pitching, and yawing control values and three outputs: rolling, pitching, and yawing rates. The dataset contains 6691 data samples with a sample rate of 0.01 seconds. A total of 4683 samples are used to train GRNN in the batch mode, and the remaining data samples (2008) are used for testing. The results of hexacopter attitude identification are shown in **Figure 3(a–c)**. The results are accurate with very low error. MSE in training stage is 0.001139 and 0.00258 in the testing stage. Also, the training time was only 0.720 seconds.

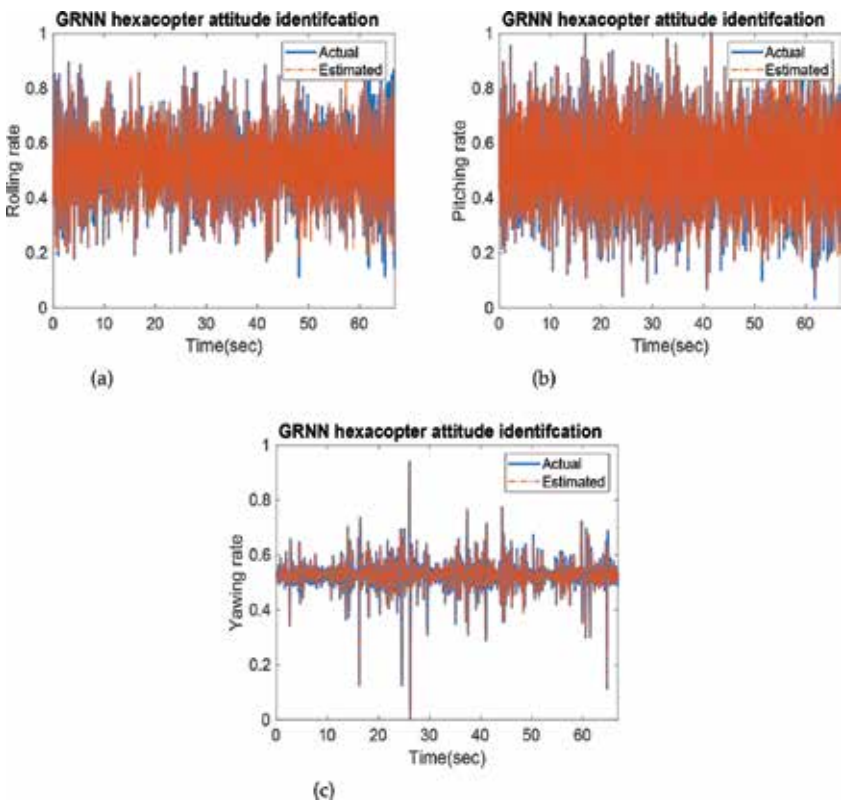

Figure 3. Attitude identification of hexacopter in batch training: (a) rolling rate identification, (b) pitching rate identification, and (c) yawing rate identification.

http://dx.doi.org/10.5772/intechopen.80258

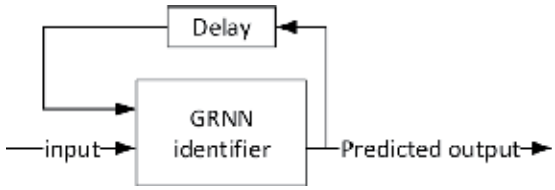

Figure 4. Sequential training GRNN.

5.2. GRNN identification in sequential training mode

In sequential training, the data flow once at a time which makes using the batch training procedures impossible. So GRNN should be able to find the system model from only the current and past measurements. So it is a prediction problem. Since GRNN converges to a regression surface even with a few data samples and since it is accurate and quick, it can be used in the online dynamic systems identification.

5.2.1. Sequential training GRNN to identify hexacopter attitude dynamics

To use GRNN in sequential mode, it is preferred to use the delayed output of the plant as an input in addition to the current input as shown in **Figure 4**. The same data which was used for batch mode is used in the sequential training. The inputs to GRNN are the control values of

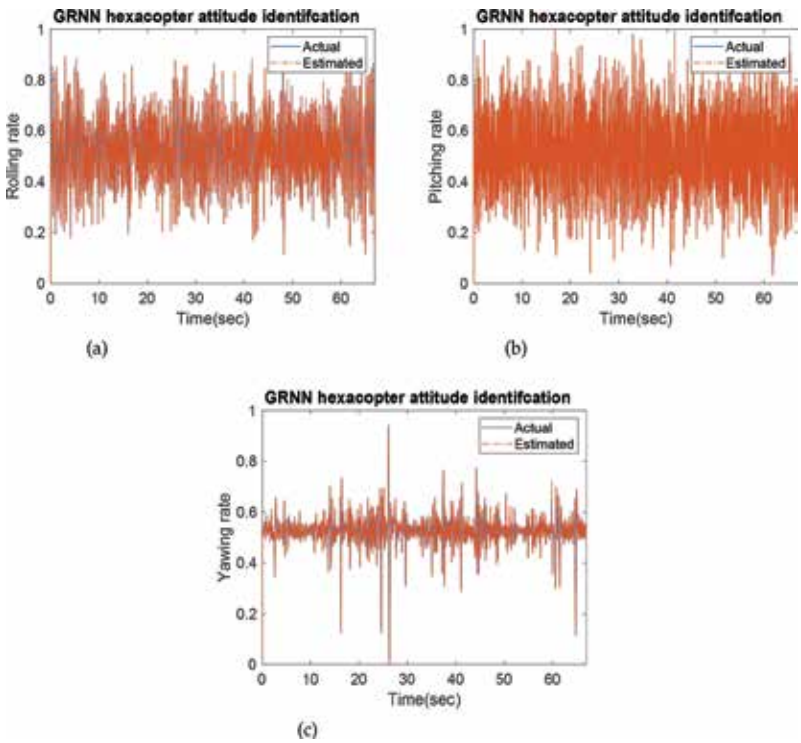

Figure 5. Attitude identification of hexacopter in sequential training: (a) rolling rate identification, (b) pitching rate identification, and (c) yawing rate identification.

rolling, pitching, and yawing and the delayed rolling, pitching, and yawing rates. The results of using GRNN in the sequential training mode are shown in **Figure 5(a–c)**. The results of sequential training are more accurate than the results in batch training.

6. GRNN in control of dynamic systems

The aim of adding a closed-loop controller to the dynamic systems is either to reach the desired performance or stabilize the unstable system. GRNN can be used in controlling dynamic systems as a predictive or feedback controller. GRNN in control systems can be used as either supervised or unsupervised. When GRNN is trained as a predictive then the controller input and output data are known, so this is a supervised problem. On the other hand, if GRNN is utilized as a feedback controller (see **Figure 6**) without being pretrained, only the controller input data is known so GRNN have to find the suitable control signal u.

6.1. GRNN as predictive controller

To utilize GRNN as a predictive controller, it should be trained with input-output data from another controller. For example, training a GRNN with a proportional integral derivative (PID) controller input/output data as shown in **Figure 7**. Then the trained GRNN can be used as a controller.

6.1.1. Example 1: GRNN as predictive controller

If we have a discrete time system Liu [26] described as

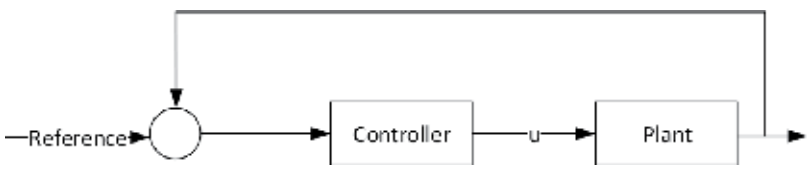

Figure 6. Unsupervised learning problem in control.

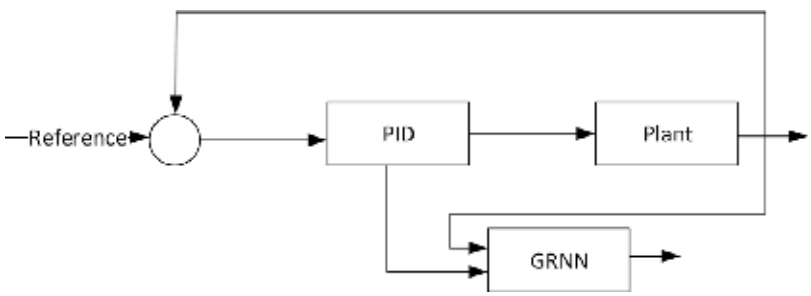

Figure 7. Training GRNN as predictive controller.

$$y(k+1) = 0.8 * sin\,(y(k)) + 15 * u(k) \tag{13}$$

The desired reference is $y_d(k) = 2^* sin\,(0.1\pi t)$.

The perfect control law can be written as

$$u(k) = \frac{y_d(k+1)}{15} - \frac{0.8 * sin\,(y(k))}{15} \tag{14}$$

To train GRNN as a predictive controller, the system described in (13) and (14) is simulated for 50 seconds. Then the controller output u and the plant output y were stored. GRNN is trained with the plant output as input and the controller output as output. For any time step the plant output is fed to GRNN, and the controller output u is estimated. The estimated controller output by GRNN and the perfect controller output are almost identical as shown in **Figure 8**. Also, the tracking performance after using GRNN as a predictive controller is very accurate as shown in **Figure 9**.

6.2. GRNN as an adaptive estimator controller

Since GRNN has robust approximation abilities, it can be used to approximate the dynamics of a given system to find the control law especially if the system is partially known or unknown.

Assume there is a nonlinear dynamic system written as

$$\dot{x} = f(x,t) + bu + d \tag{15}$$

where $\dot{x}$ is the derivative of the states, $f(x,t)$ is a known function of the states, b is the input gain, and d is the external disturbance.

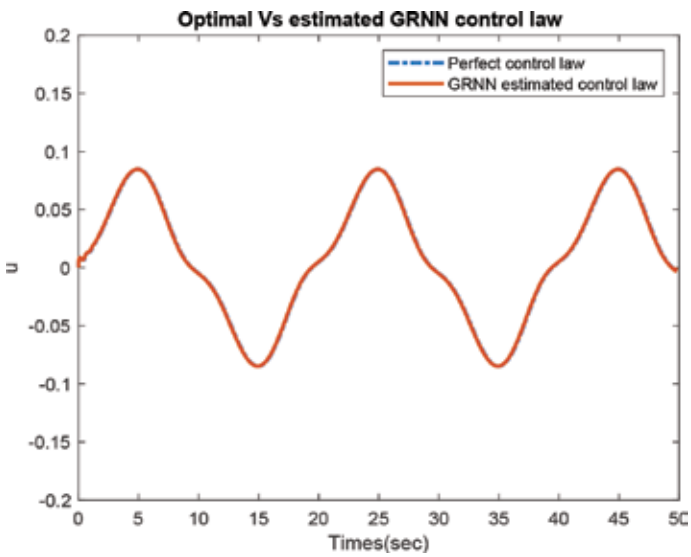

Figure 8. Perfect vs. estimated GRNN controller output.

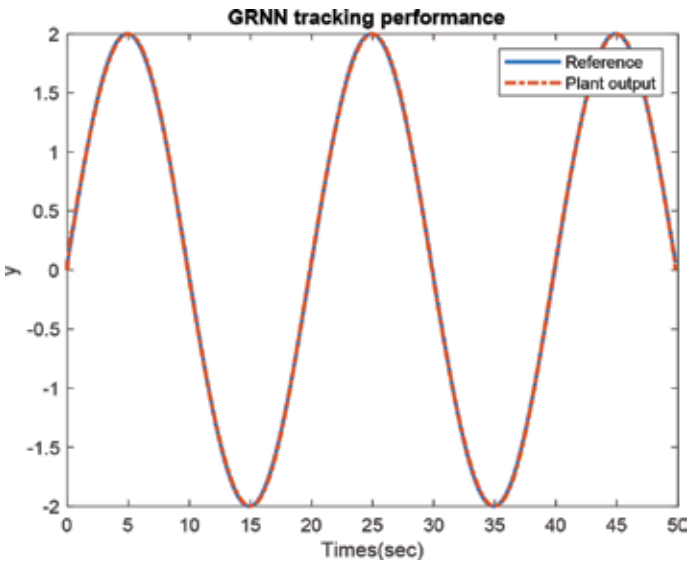

Figure 9. GRNN tracking performance.

The perfect control law can be written as

$$u = \frac{1}{b}(\dot{x} - f(x,t) - d) \tag{16}$$

If $f(x,t)$ is unknown, then the control law in (16) cannot be found; hence, the alternative is using GRNN to estimate the unknown function $f(x,t)$. To derive the update law of GRNN weights, let us define the objective function as MSE error function as follows:

$$E = \frac{1}{2}(\hat{y} - y)^2 \tag{17}$$

where $\hat{y}$ is the estimation of GRNN and y is the optimal value of $f(x,t)$. To derive the update law of the GRNN weights, the error should be minimized with respect to GRNN weights W:

$$\frac{\partial E}{\partial W} = \left(\hat{W}H - y\right) * H \tag{18}$$

where $\hat{W}$ is the GRNN current hidden-output layers weights and H is the hidden layer output, so the update law of GRNN weights will be

$$W_{i+1} = W_i + H\left(\hat{W}H - y\right) \tag{19}$$

6.3. Example 2: using GRNN to approximate the unknown dynamics

Let us consider the same discrete as in example 1:

$$y(k+1) = f(k) + 15*u(k) \tag{20}$$

The desired reference is $y_d(k) = 2^* sin\,(0.1\pi t)$

where $f(k)$ is unknown nonlinear function.

The perfect control law can be written as

$$u(k) = \frac{-f(k)}{15} + \frac{y_d(k)}{15} \tag{21}$$

GRNN is used to estimate the unknown function $f(k)$. With applying the update law in (19), $f(k)$ is estimated with an acceptable accuracy as shown in **Figure 10**. MSE between the ideal and the estimated $f(k)$ is 0.0033. The accurate controller tracking performance is also shown **Figure 11**.

6.4. GRNN as an adaptive optimal controller

GRNN has learning abilities which means it is suitable to be an adaptive intelligent controller. Rather than approximating the unknown function in the control law (16), one can use GRNN to approximate the whole controller output as shown in **Figure 12**. The same update law as in (19) can be used to update GRNN weights to approximate the controller output u.

6.4.1. Example 3: using GRNN as an adaptive controller

Let us consider the same discrete system as in (13):

$$y(k+1) = 0.8*sin\,(y(k)) + 15*u(k)$$

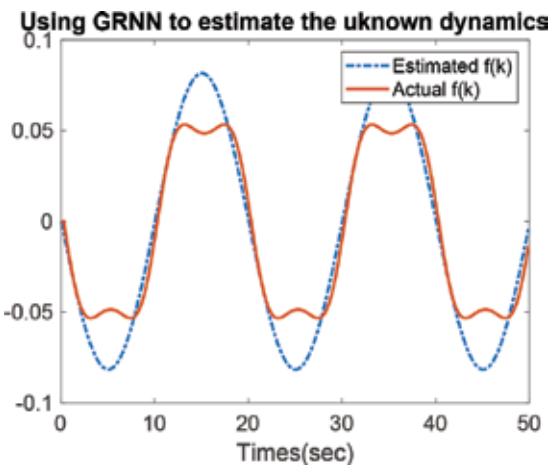

Figure 10. Using GRNN to estimate the unknown dynamics.

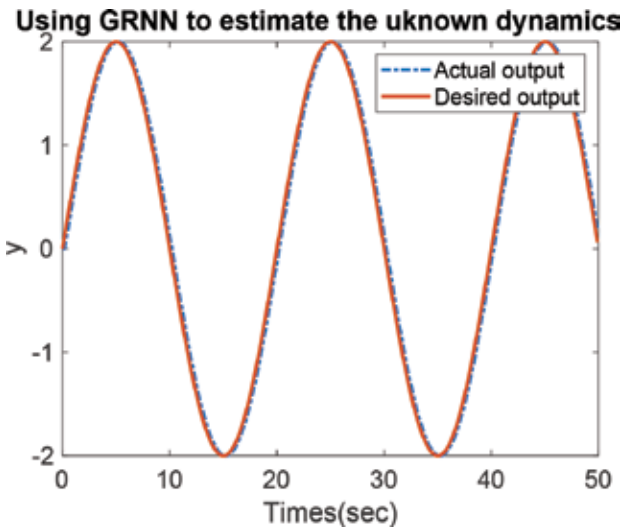

Figure 11. GRNN tracking performance for example 2.

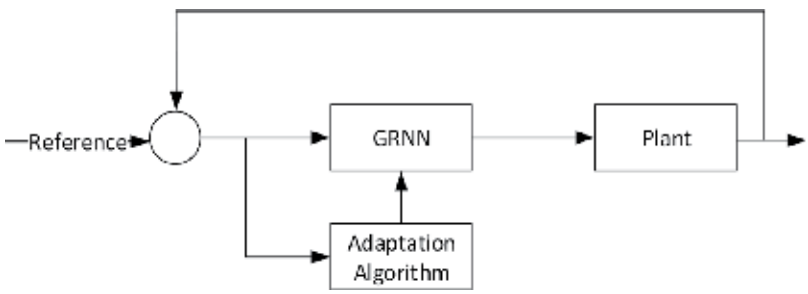

Figure 12. Training GRNN as an adaptive controller.

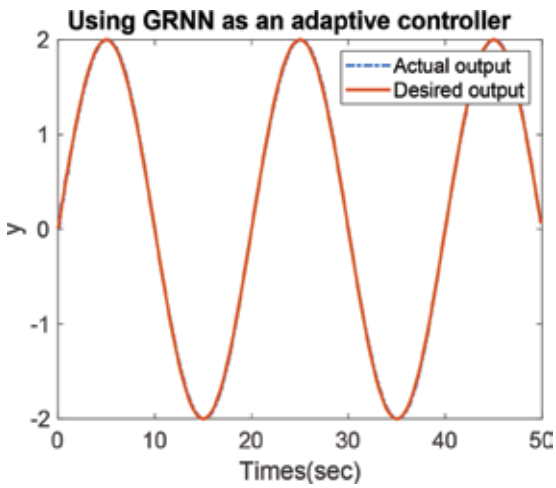

Figure 13. GRNN tracking performance for example 3.

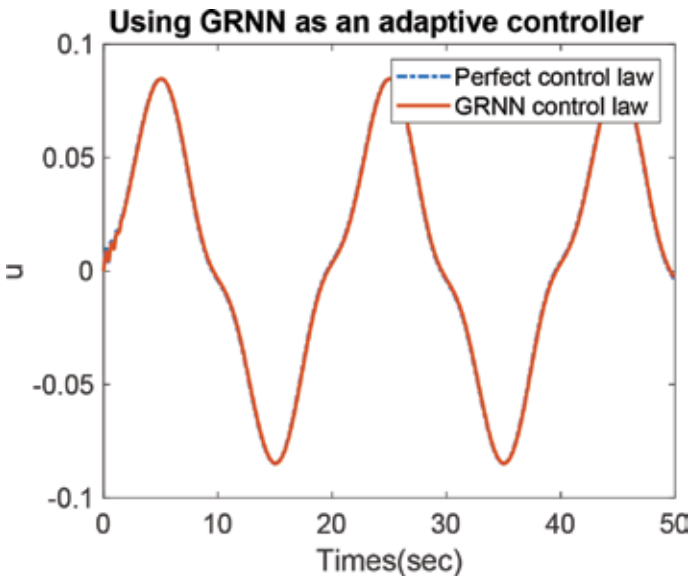

Figure 14. GRNN Estimated control law for example 3.

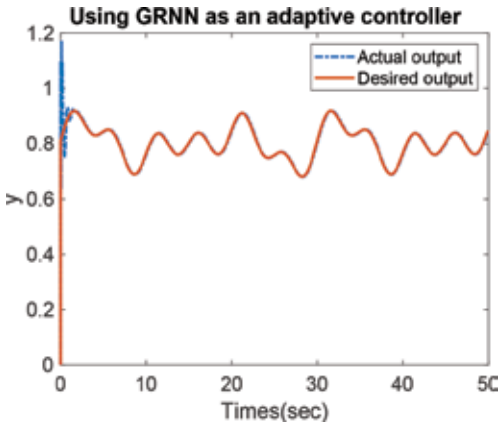

Figure 15. GRNN as an adaptive controller in Example 4.

with the same desired reference $y_d(k) = 2 * sin\,(0.1\pi t)$, but in this case GRNN is used to estimate the full controller output u as shown in **Figure 14** and the tracking performance is shown in **Figure 13**.

6.4.2. Example 4: using GRNN as an adaptive controller

Let us use GRNN to control a more complex discrete plant [27] described as

$$y(k+1) = 0.2\,cos\,(0.8(y(k) + y(k-1))) + 0.4\,sin\,(0.8(y(k-1) + y(k) + 2u(k) + u(k-1))) + 0.1(9 + y(k) + y(k-1)) + \frac{2(u(k) + u(k-1))}{(1 + cos\,(y(k))} \quad (22)$$

The desired reference in this case is

$$y_d(k) = 0.8 + 0.05(\sin(\pi k/50) + \sin(\pi k/100) + \sin(\sin(\pi k/150)))$$

The tracking performance of adaptive GRNN is shown in **Figure 15**.

7. MATLAB examples

In this section, GRNN MATLAB code examples are provided.

7.1. Basic GRNN Commands in MATLAB

In this example, GRNN is trained to find the square of a given number.

To design a GRNN in MATLAB:

Firstly, create the inputs and the targets and specify the spread parameter:

```
clc
clear all
x=[1 3 5 7 9 11];
y=[1 9 25 49 81 121];
spread=0.1;
```

Secondly, create GRNN:

```
net1=newgrnn(x,y,spread);
```

To view GRNN after creating it:

```
view(net1)
```

The results are shown in **Figure 16**.

To find GRNN output based on a given input:

```
y2=net1(4);
```

The result is 17.

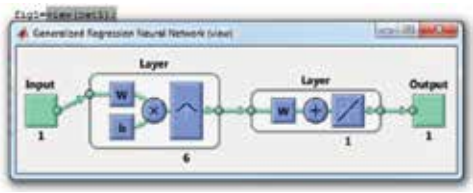

Figure 16. View GRNN in MATLAB.

7.2. The holdout method to find σ

```
      clc
      clear all
      close all
      x=[1 3 5 7 9 11];
      y=[1 9 25 49 81 121];
      data=[x;y];
%keep the original dataset
      data_or=data;

% %spread vecotr . it can be more than zero and less or equal one
      sp=0.0001:0.01:1;
      emin=100;
      tic
for j=1:length(sp)

      for i=1:10 %10 folds

%s e l e c t a random data sample from training data set
      randxi=randi(6);

      randx(:,i)=data(:,randxi);
%remove the selected training sample from the training data set
      data(:,randxi)=[];
%train GRNN with the reduced training dataset
      net1=newgrnn(data(1,:),data(2,:),sp(j));
%find GRNN output of the removed sample
      y1(i)=net1(randx(1,i));
% find the meaan sugre error between GRNN output and the removed
sample
% target
      me(i)=mse(y1(i),randx(2,i));
%rest the training data as it was before remving the training sample
      data=data_or;
%
      end
% find the summation of the errors for the current sigma
m_sum(j)=sum(me);
end

%find sigma which have the least errors summation
      [idx,ii]=min(m_sum);
      best_signa=sp(ii);
      time_holdout=toc;
      disp('best sigma is..');
      disp(best_sigma);
```

Author details

Ahmad Jobran Al-Mahasneh[1], Sreenatha Anavatti[1]*, Matthew Garratt[1] and Mahardhika Pratama[2]

*Address all correspondence to: s.anavatti@adfa.edu.au

1 School of Engineering and Information Technology, The University of New South Wales Canberra, ACT, Australia

2 School of Computer Science and Engineering, Nanyang Technological University, Singapore

References

[1] Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators. Neural Networks. 1989;**2**(5):359-366

[2] Hagan MT, Menhaj MB. Training feedforward networks with the Marquardt algorithm. IEEE Transactions on Neural Networks. 1994;**5**(6):989-993

[3] Hecht-Nielsen R. Theory of the backpropagation neural network. In: Neural Networks for Perception. Netherlands: Elsevier; 1992. pp. 65-93

[4] Specht DF. A general regression neural network. IEEE Transactions on Neural Networks. 1991;**2**(6):568-576

[5] Schaffner C, Schroder D. An application of general regression neural network to nonlinear adaptive control. In: Fifth European Conference on Power Electronics and Applications, IET; 1993. pp. 219-224

[6] Ahmed O, Mitchell JW, Klein SA. Application of general regression neural network in hvac process identification and control, Technical report. Atlanta, GA (United States): American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc; 1996

[7] Kulkarni SG, Chaudhary AK, Nandi S, Tambe SS, Kulkarni BD. Modeling and monitoring of batch processes using principal component analysis assisted generalized regression neural networks. Biochemical Engineering Journal. 2004;**18**(3):193-210

[8] Ben-Nakhi AE, Mahmoud MA. Cooling load prediction for buildings using general regression neural networks. Energy Conversion and Management. 2004;**45**(13–14):2127-2141

[9] Lee WY, House JM, Kyong NH. Subsystem level fault diagnosis of a building's air-handling unit using general regression neural networks. Applied Energy. 2004;**77**(2): 153-170

[10] Sahroni A. Design of intelligent control system based on general regression neural network algorithm. GSTF Journal on Computing (JoC). 2018;**2**(4):103-110

[11] Hong CM, Cheng FS, Chen CH. Optimal control for variable-speed wind generation systems using general regression neural network. International Journal of Electrical Power and Energy Systems. 2014;**60**:14-23

[12] Li HZ, Guo S, Li CJ, Sun JQ. A hybrid annual power load forecasting model based on generalized regression neural network with fruit fly optimization algorithm. Knowledge-Based Systems. 2013;**37**:378-387

[13] Wei W, Shaoyi B, Lanchun Z, Kai Z, Yongzhi W, Weixing H. Vehicle sideslip angle estimation based on general regression neural network. Mathematical Problems in Engineering. 2016;**2016**:1-7

[14] Huang K, Liu Z, Huang D. Fault diagnosis for methane sensors using generalized regression neural network. International Journal of Online Engineering (iJOE). 2016;**12**(03):42-47

[15] He XY, He SH. Fault detection of excavators hydraulic system using dynamic general regression neural network. Applied Mechanics and Materials, Trans Tech Publications. 2011;**48**:511-514

[16] Pietrowski W. Detection of time-varying inter-turn short-circuit in a squirrel cage induction machine by means of generalized regression neural network. COMPEL-The International Journal for Computation and Mathematics in Electrical and Electronic Engineering. 2017;**36**(1):289-297

[17] Stenhouse T. Integration of helicopter autonomy payload for non-linear system identication of rotorcraft heave mode. The UNSW Canberra at ADFA Journal of Undergraduate Engineering Research. 2018;**9**(2):1-15

[18] Chen TC, Yu CH. Generalized regression neural-network-based modeling approach for traveling-wave ultrasonic motors. Electric Power Components and Systems. 2009;**37**(6): 645-657

[19] Husain H, Khalid M, Yusof R. Automatic clustering of generalized regression neural network by similarity index based fuzzy c-means clustering. In: 2004 IEEE Region 10 Conference, TENCON 2004, IEEE; 2004. pp. 302-305

[20] Tomandl D, Schober A. A modified general regression neural network with new, efficient training algorithms as a robust black box-tool for data analysis. Neural Networks. 2001; **14**(8):1023-1034

[21] Goulermas JY, Liatsis P, Zeng XJ, Cook P. Density-driven generalized regression neural networks for function approximation. IEEE Transactions on Neural Networks. 2007;**18**(6): 1683-1696

[22] Rutkowski L. Generalized regression neural networks in time-varying environment. IEEE Transactions on Neural Networks. 2004;**15**(3):576-596

[23] Seng T, Khalid M, Yusof R. Adaptive general regression neural network for modelling of dynamic plants. Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering. 1999;**213**(4):275-287

[24] Al-Mahasneh AJ, Anavatti SG, Garratt MA. Altitude identification and intelligent control of a flapping wing micro aerial vehicle using modified generalized regression neural networks. In: 2017 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE; 2017. pp. 2302-2307

[25] Mathworks 2018. Neural Network Toolbox Sample Data Sets for Shallow Networks. [Accessed: June 11, 2018]

[26] Liu J. Radial basis function neural network control based on gradient descent algorithm. In: Radial Basis Function Neural Network Control for Mechanical Systems. Germany: Springer; 2013. pp. 55-69

[27] Adetona O, Garcia E, Keel LH. A new method for the control of discrete nonlinear dynamic systems using neural networks. IEEE Transactions on Neural Networks. 2000; **11**(1):102-112

Edited by Vahid Asadpour

This book provides an approach toward the applications and principle theory of digital signal processing in modern intelligent systems, biological engineering, telecommunication, and information technology. Assuming the reader already has prior knowledge of signal processing theory, this book will be useful for finding novel methods that fit special needs in digital signal processing (DSP). The combination of signal processing and intelligent systems in hybrid structures rather than serial or parallel processing provide the best mechanism that is a better fit with the comprehensive nature of human.

This book is a practical reference that places the emphasis on principles and applications of DSP in digital systems. It covers a broad area of digital systems and applications of machine learning methods including convolutional neural networks, evolutionary algorithms, adaptive filters, spectral estimation, data compression and functional verification.

The level of the book is ideal for professional DSP users and useful for graduate students who are looking for solutions to their design problems. The theoretical principles provide the required base for comprehension of the methods and application of modifications for the special needs of practical projects.

Published in London, UK

IntechOpen

ISBN 978-1-83881-811-1
9 781838 818111