Turkish Journal of Electrical Engineering & Computer Sciences

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

Research Article

Turk J Elec Eng & Comp Sci
(2013) 21: 699 – 713

doi:10.3906/elk-1104-22

Real-time implementation of an amplitude-locked loop: a validation on the dSPACE DS1006-based platform

Jora Mangala GONDA,[1,*] **Sumam DAVID**[2]

[1]Department of Electrical and Electronics Engineering, National Institute of Technology Karnataka, Surathkal, Srinivasnagar, Mangalore 575025, India

[2]Department of Electronics and Communication Engineering, National Institute of Technology Karnataka, Surathkal, Srinivasnagar, Mangalore 575025, India

Received: 19.04.2011 • **Accepted:** 28.02.2012 • **Published Online:** 03.05.2013 • **Printed:** 27.05.2013

Abstract: The extraction of harmonics and/or the fundamental from a distorted waveform is an important process in the implementation of custom-power devices. Several schemes towards this have been proposed in the past. Among these, the algorithms based on synchronous (with respect to the supply voltages) extraction (both in phase and amplitude) have certain established advantages over the others. Amplitude-locked loops (ALLs) have been in use in signal-communication systems but are limited to sinusoidal inputs. There is a need for fast and rugged algorithms to synchronously extract harmonics and/or the fundamental from a distorted waveform in many power system applications. In this paper a real-time implementation of a novel scheme, which is based on an adaptation of an ALL, is presented for synchronous extraction of harmonics and/or the fundamental from a distorted periodic waveform. The operation of the algorithm, its performance, and its design aspects are briefly discussed. The main features of this ALL are simplicity, speed of operation, noise rejection, availability of both fundamental and harmonics without much additional processing, and excellent insensitivity to distortion (robustness). Furthermore, it is applicable to single-phase or 3-phase systems. This paper reports a real-time hardware implementation of the algorithm, thereby validating it. The algorithm is implemented on a real-time hardware-emulation platform, a dSPACE modular system (configured around the DS1006 processor board). It is tested for various cases of interest and the results are presented.

Key words: Amplitude locking, phase locking, harmonics extraction, amplitude demodulation, noise rejection

1. Introduction

The extraction of harmonics or the fundamental component of voltages and/or currents is one of the main tasks in the implementation of custom-power devices. The techniques available for this purpose broadly fall into single-phase and 3-phase categories. They can also be classified into open-loop techniques and closed-loop techniques. A comparative evaluation of some of these methods can be found in [1]. The importance of efficient extraction and the applications thereof are covered extensively in [2]. The closed-loop techniques have the distinct advantage of being capable of staying synchronized (i.e. zero-deviation, amplitude- and phase-locked) to the input, which is very important in all applications connected to the utility grid. Extracting the fundamental and/or harmonics from a single-phase signal in a closed-loop scheme is the primary focus of this paper. The proposed scheme falls into the class of closed-loop techniques [3, 4, 5, 6, 7].

*Correspondence: joragondam@gmail.com

Phase- and amplitude locking have been in use in the field of signal communications for the purpose of amplitude modulation or demodulation. In 1991, a vector-locked loop (VLL) was presented by DaSilva in a patent [6], wherein peak detection is used for magnitude determination. Pettigrew presented a scheme for amplitude locking in another patent [8] in 1994. Both of the aforementioned schemes assume the input to be sinusoid and hence cannot directly be used for extracting harmonics from distorted inputs. Moir [3] presented an analysis of the scheme given by Pettigrew.

In the area of synchronized filtering of a nonsinusoidal (distorted) waveform, the scheme presented in 1995 by Luo and Hou [4] has the desirable feature of staying locked to the phase-locked loop (PLL) reference while being simple to understand and to implement. However, its settling time is a function of the integrator gain, and an increase in the gain to improve the dynamic performance leads to an increased distortion in the filtered output. A VLL for synchronous extraction of harmonics was presented for the first time in 2002 by Karimi-Ghartemani and Iravani [7]. An improved scheme was presented by Karimi et al. [5] in 2003, wherein a low-pass filter is introduced in cascade with the integrator, which improves the steady-state response and allows for a higher gain, thereby reducing the settling time. Recently Bhat et al. [9] proposed an algorithm with features comparable to the ones mentioned above, while additionally having a provision of self-tuning the resonant frequency of the filter.

Pettigrew's scheme was adapted by Gonda et al. [10] for the extraction of harmonics, using the demodulation property of the scheme. The above adaptation involves developing the mathematical framework to substantiate the extension and suggesting modifications to improve the performance parameters. A flat second-order-low-pass filter (SOLF) pretuned to a cut-off frequency (ω_c) has been used for improving the performance.

This paper focuses on the implementability of the above algorithm in real time. A modular system [11] based on the DS1006 processor board, a real-time emulation platform, is used for the purpose. The algorithm is implemented on this system and the results obtained from multiple test-runs are compiled and presented.

The paper is organized as follows: in Section 2, a brief review of the analysis presented by Moir is given in order to understand the operation of the scheme and to place the ensuing discussion in the proper perspective. The adaptation of the scheme to extract the fundamental is explained in Section 3 with the necessary mathematical proof. The design parameters influencing the performance parameters are identified. Design considerations are discussed in Section 4 and empirical rules for the design are presented. The real-time hardware-emulation platform is explained in Section 5. The process of validation is presented in Section 6. The experimental setup is detailed in Section 7 and the real-time experimental results are presented in Section 8.

2. Brief description of Pettigrew's ALL

The amplitude-locked loop (ALL) proposed by Pettigrew [8] is shown in Figure 1. Moir [3] presented an analysis of the circuit. The principle of the scheme is captured in the following short description. In the circuit shown in Figure 1, the output of the multiplier-M1 is

$$y(t) = u(t)x_I(t). \tag{1}$$

Consider an input,

$$x_I(t) = c(t)\cos(\omega_o t + \phi_i), \tag{2}$$

with $c(t)$ as a modulating signal and $\cos(\omega_o t + \phi_i)$ as the carrier. For the purpose of analysis $c(t)$ and ϕ_i are initially considered constants, but in reality they may be time-varying. It is assumed that $c(t) \neq 0$. The signal

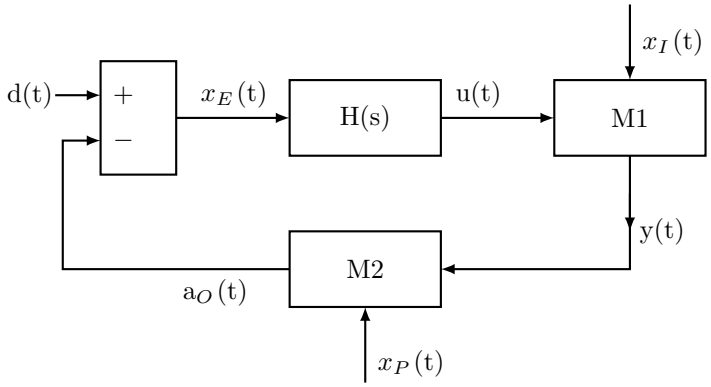

Figure 1. Block diagram of Pettigrew's ALL.

$x_P(t)$, given by

$$x_P(t) = B_P \cos(\omega_o t + \phi_P), \tag{3}$$

is obtained as the output of a PLL with center frequency, equal to the carrier frequency ω_o of $x_I(t)$ and also having $x_I(t)$ as the input. Let the output of the multiplier M2 be

$$a_O(t) = x_P(t)y(t). \tag{4}$$

When the ALL is in the locked position, the error signal is

$$\begin{aligned} X_E(s) &= D(s) - A_O(s) &(5)\\ &= D(s) - X_P(s) * Y(s), &(6)\end{aligned}$$

where the above equations are in the Laplace domain and the operator $*$ denotes convolution.

In the normalized output of the PLL, $B_P = 1$. Neglecting the $2\omega_o$ terms in $x_E(t)$, and considering

$$H(s) = \frac{G}{s}, \tag{7}$$

an integrator with a gain G, it is shown in [3] that

$$U(s) = \frac{G}{s + c\dfrac{G}{2}\cos(\phi_P - \phi_i)} D(s). \tag{8}$$

With a step input of magnitude D for $d(t)$ and with $x_E(t)$ converging to zero in the steady state, the response is

$$u(t) = \frac{2D}{c\cos(\phi_P - \phi_i)}. \tag{9}$$

Under locked conditions and when $\phi_i = \phi_P$ (which is usually the case),

$$u(t) = \frac{2D}{c}. \tag{10}$$

Thus, $u(t)$ becomes the scaled inverse of amplitude c of input $x_I(t)$. Eqs. (1) and (2) with Eq. (10) then lead to

$$y(t) = 2D\cos(\omega_o t + \phi_i), \tag{11}$$

which is a scaled version of the carrier $\cos(\omega_o t + \phi_i)$, where the scaler $2D$ is decided by the amplitude reference $d(t) = D$. Note that $y(t)$ is devoid of the modulating signal c and hence is the amplitude-demodulated version of $x_I(t)$. Eq. (11) shows that the system has the ability to remove the modulating signal, leaving behind only the carrier. Then $u(t)$ becomes the scaled inverse of the amplitude of the modulating signal in the input. That is, the ALL tracks the inverse of the modulus of the modulating signal.

Although this analysis considered only the case of $c(t) = c$ (a constant), it is intuitive that for the case of time-varying modulating signal $c(t)$,

$$u(t) = \frac{2D}{c(t)}, \tag{12}$$

and $y(t)$ is still given by Eq. (11). However, this is true only under the condition that the product of the frequency and the depth of modulation remains within a certain range (tracking range).

3. Adaptation of Pettigrew's ALL for fundamental and/or harmonics extraction

3.1. Basic operation

Consider a signal input $x_I(t)$, a periodic waveform with an average component x_{DC},

$$x_I(t) \quad = \quad x_{DC} + \sum_{k=1}^{\infty} C_k \cos(k\omega_o t + \phi_k), \tag{13}$$

where k is the harmonic order (with $k = 1$ corresponding to the fundamental), C_k and ϕ_k refer to the Fourier series coefficients, and ω_o is the fundamental frequency in rad/s. The output of the PLL (pretuned to a free-running frequency of ω_o), which is purely sinusoidal and in phase with the fundamental component of $x_I(t)$, is

$$x_P(t) \quad = \quad \cos(\omega_o t + \phi_1), \tag{14}$$

where $B_P = 1$ is considered. Now each term in Eq. (13) can be considered as the product of a modulated signal determined by the coefficient C_k and a carrier determined by $(k\omega_o)$. If $x_I(t)$ were to contain only the fundamental component at ω_o and if the PLL is tuned to ω_o, it is clear from Eqs. (9) and (10) that C_1 can be obtained as the reciprocal of $u(t)$ and appropriately scaled in proportion to the commanded amplitude factor D. It is also clear from Eq. (10) that if D is chosen as 0.5, then u will be reciprocal of C_1 – the fundamental amplitude of $x_I(t)$. It is also true that a_O, which is obtained as a multiplication of 2 in-phase sine waves of like frequency, contains a constant equal to half the product of each of the amplitudes. Thus, the choice of $D = 0.5$ is further justified. It should be noted that a_O also contains $2\omega_o$ components, attenuated by the integrator in the $H(s)$. Additional attenuation can be achieved, as will be explained shortly. The product of C_1 and $x_P(t)$ recovers the input signal for this case of a single-frequency input. Thus,

$$x_O(t) \quad = \quad c(t)x_P(t) \tag{15}$$

$$\quad = \quad C_1\cos(\omega_o t + \phi_1). \tag{16}$$

This adaptation is shown in Figure 2. It should be noted that, under the steady-state, the fundamental amplitude of $y(t)$ shall be unity; until then the corrections will continue in the feedback loop.

3.2. Principle of working with input containing a constant and the harmonics

The transient response of $u(t)$, and hence that of $c(t)$, is decided by the rate of charging the integrator in $H(s)$, which is determined by the constant value component in $x_E(t)$. It should be noted according to Eq. (4) that any component in $x_I(t)$ other than the ω_o component will not produce a constant value in $a_O(t)$. Thus, the presence of such components in $x_I(t)$ will not affect the transient response of $u(t)$ and hence that of $c(t)$. Therefore, all such terms in Eq. (13), other than that associated with ω_o, will not affect the transient response. However, they have a direct bearing on the steady-state response of $c(t)$ and show up as ripple in it. However, the steady-state and transient performances can be improved by inserting a low-pass filter $F(s)$, as shown in Figure 2.

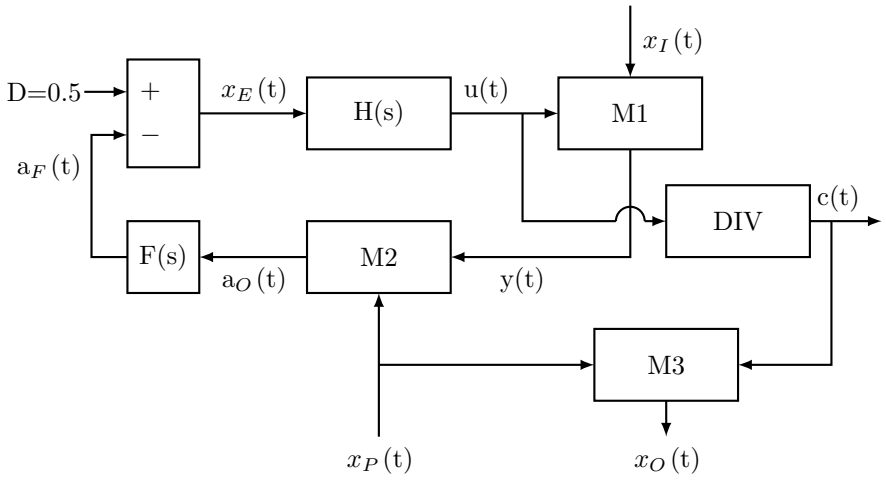

Figure 2. Block diagram showing the adaptation.

4. Design considerations

Ideally $F(s)$ should pass only the average component, blocking the fundamental (which may creep-in due to the presence of a constant part in x_I) and all the higher-order harmonics. Hence, the cut-off frequency should be considerably less than ω_o (i.e. $\omega_c << \omega_o$). Then there will be better attenuation of the harmonics in the loop and hence improvements in the steady-state performance parameters.

It is to be noted that the design parameters are the gain G and the parameters of the filter $F(s)$. On the other hand, the performance parameters are settling time, harmonic rejection, steady-state error, amplitude modulation, and noise rejection. The system consists of linear components $H(s)$ and $F(s)$. However, the presence of multipliers makes the system nonlinear. This makes it difficult to apply small-signal-analysis, normally applicable on systems with nonlinear elements. Hence, here it is not attempted to obtain analytical expressions relating the performance parameters to the design parameters. Instead, an extensive simulation study is carried out and experiments are conducted to arrive at a good set of values for the design parameters. It should also be noted that there is a need to limit the lower level for $u(t)$, especially during the start-up.

Otherwise, division by zero can create a problem, and also the amplitude $C_1(t)$ of the fundamental in x_O will start building from an unrealistically large value, making the settling time unnecessarily longer. This limit can be chosen with the knowledge of the practical range of the amplitude for the fundamental under any given circumstances.

4.1. PLL design

This scheme works in association with a PLL taking $x_I(t)$ as the reference input. Note that the proposed ALL output will be of the same frequency as that produced by the PLL. Therefore, it is necessary to design the PLL with a free-running frequency close to the fundamental frequency component in the input. The PLL performance can be optimized by well-established techniques [12].

4.2. Design of the circuit

There are only 2 components to be designed in the system, $H(s)$ and $F(s)$. $H(s)$ is used to drive the steady-state error to zero as quickly as possible, by appropriate choice of the gain G. The function of the low-pass filter, $F(s)$, is to attenuate the harmonics generated in $a_O(t)$ due to the presence of higher-order harmonics in $x_I(t)$, as well as the fundamental and second-order harmonic produced due to the interaction of the fundamental in $x_I(t)$ with $x_P(t)$. It is also necessary that it does this as fast as possible. Hence, it is found through experiment that a flat SOLF is a good choice and a suitable cut-off frequency of ω_c is selected. The transfer function of the low-pass filter is as given below:

$$F(s) = \frac{1}{\frac{s^2}{\omega_c^2} + \frac{s}{Q\omega_c} + 1}, \tag{17}$$

where ω_c is the cut-off frequency in rad/s and Q is the quality factor. Since the SOLF introduces undesirable delay, a larger value of 35 Hz is chosen for ω_c, as a compromise between the conflicting requirements of attenuating the ripple (seeks a smaller ω_c) and improving the transient response (seeks a larger bandwidth and hence ω_c). It is observed that an increase in the gain G (for a fixed ω_c) results in an increase in the steady-state error and a decrease in the settling time. A similar relation holds for a change in ω_c (for a fixed G). This makes the tuning of the algorithm very easy; however, with the relations being nonlinear, it is achieved through extensive simulation study.

5. Real-time emulation platform

The algorithm is to be tested on a real-time hardware-emulation platform, a dSPACE modular system configured around the DS1006 processor board. While such a system can be useful in different applications, here it is used for the purpose of development, testing, and validation of an algorithm for real-time implementation. This process becomes important in cases like the algorithm under consideration where the nonlinearity does not permit an analytical solution and numerical techniques are the only alternatives. After testing and fine-tuning the parameters, in actual practice the algorithm may be implemented on any other hardware, like a microprocessor (μP), a microcontroller (μC), a digital signal processor (DSP), digital signal controllers (DSCs), or field programmable gate arrays (FPGAs). Any system to be implemented in real-time has to be first modeled and simulated in a higher-level language, like C/C^{++} / Simulink. This code or program is then cross-compiled into the object code of the processor housed in the hardware-emulation platform. It consists of 3 main components: hardware, a code-development assistant, and a graphical user interface (GUI). The hardware,

in general, consists of a high-speed processor, analog and digital input-output (I/O) devices or interfaces, and a communication arrangement between the host computer and the platform. The code-development assistant is basically a cross-compiler. It provides facility for developing the model in a higher-level language and cross-compiling it to the machine code corresponding to the processor in the hardware unit. The GUI helps in downloading the code thus developed onto the processor, controlling the execution of the program, and also serves as a virtual instrumentation (VI) unit. Each of these units are explained briefly in the following text with reference to the system used in this case.

5.1. Hardware section – the DS1006-based modular system

The dSPACE modular system [11], suitable for real-time computations, consists of a DS1006 processor board for calculating processing-intensive real-time models; a set of peripherals including a DS2002 analog-to-digital converter (ADC) board, a DS2101 digital-to-analog converter (DAC) board, and a DS4003 digital I/O board; and other accessories for interconnections. All of these components are housed in the PX10 expansion box with PHS bus, containing 10 PCI slots with the provision for interconnection between the cards (using the PHS cable). The communication between the host computer and the modular system is either through Ethernet cable (viz. RJ45) or optical link. The DS814 housed in PX10 and the DS817 housed in the personal computer (PC) slot have provisions for this. The system is shown schematically in Figure 3. Each of these components is briefly described below with its specifications.

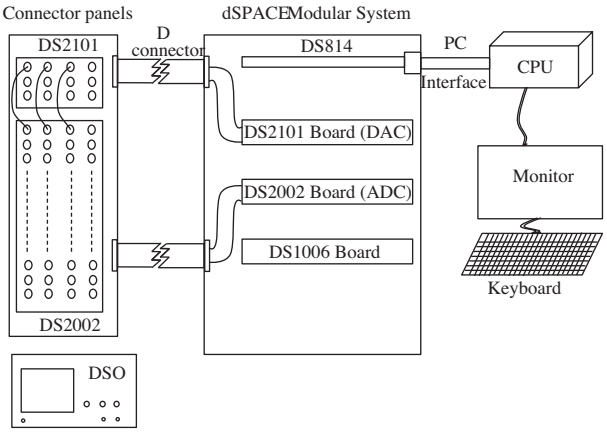

Figure 3. Schematic of DS1006-based modular system.

DS1006, a processor board: The board is built around the AMD Opteron®, x86-compatible 64-bit server, multicore processor, running at 2.6 GHz. It provides 512 kB L2 cache per core and 6.0 MB shared L3 cache. The DS1006 also has 1.0 GB local memory for executing real-time models, 128 MB global memory per core for exchanging data with the host-PC, and 2.0 MB on-board boot flash memory for automatic, host-independent booting of real-time applications.

DS2002: A 32-channel ADC with a conversion time of 3.8 μs for 12-bit conversion and input voltage range of ± 5 V or ± 10 V. There is also a Connector Panel CP2002 to facilitate connection of signals to the ADC board.

DS2101: A 5-channel DAC board with 12-bit resolution, output range ± 5 V, ± 10 V, or 0 to +10 V programmable, 3 μs settling time. There is also a Connector Panel CP2101 to facilitate connection of signals to the ADC board.

DS4003: A 92-line programmable digital I/O board. The inputs and the outputs are TTL-compatible.

When the dSPACE modular system is installed, a separate component will be added to the Simulink's Real Time Interface (RTI) Library. This library will contain ports for all the peripherals available on the modular system. These ports are used for connecting the peripherals with the processor executing the program.

5.2. Code-development assistant

In the modular system considered, a code-development assistant system is achieved with collaboration between dSPACE and MathWorks. With this arrangement, the system to be studied is modeled in Simulink and tested. It is then compiled to the machine code, using the *build* feature in the simulation profile of Simulink, by the target language compiler (TLC). If the emulation platform is installed, the TLC is installed and it adds necessary features to the simulation profile in MATLAB. This provides for targeting the code to be developed, specific to AMD Opteron®. The code will be available in *filename*.sdf, a system description file. This code shall be downloaded to the processor for running in real time.

5.3. Graphical user interface and virtual instrumentation

The installation of the dSPACE modular system also installs the Control-Desk, the GUI and the VI for the system. It provides for setting the emulation profile, downloading the *.sdf* file onto the processor, and controlling the execution of the program. It also has a virtual instrumentation, through which it is possible to control real-time inputs to the model, observe (in virtual meters (analog or digital) or in virtual scopes) the quantities of interest, and store the results of the experiments.

6. Process of validation

6.1. Simulation

The algorithm to be tested is first simulated in Simulink of MATLAB. The schematic is shown in Figure 4. The edited algorithm is initialized with the parameters and the signals are assigned proper values. Appropriate simulation step size (a fixed step size is chosen, which is necessary for real-time simulation) and an appropriate numerical integration algorithm are selected (like the Range–Kutta fourth-order). The algorithm is tested and checked for desirable performance parameters. Several repeated trials of the experiments are conducted and the parameters of the system are adjusted until satisfactory results are obtained. The results are reported in [10].

6.2. Real-time emulation

Now the model is set up for real-time testing on the real-time hardware-emulation platform, the dSPACE modular system based on DS1006. The signals connected to the model simulated above are now removed. The signal sources used above now work as signal generators in reality and are directed to the DACs on the DS2101 board. These signals are connected externally to the ADC inputs available on the DS2002 board. This is achieved by utilizing the ports provided in the RTI Library of the Simulink toolbox. The ADC outputs are picked up by the processor through the ADC output ports. Thus, the emulation setup is made ready.

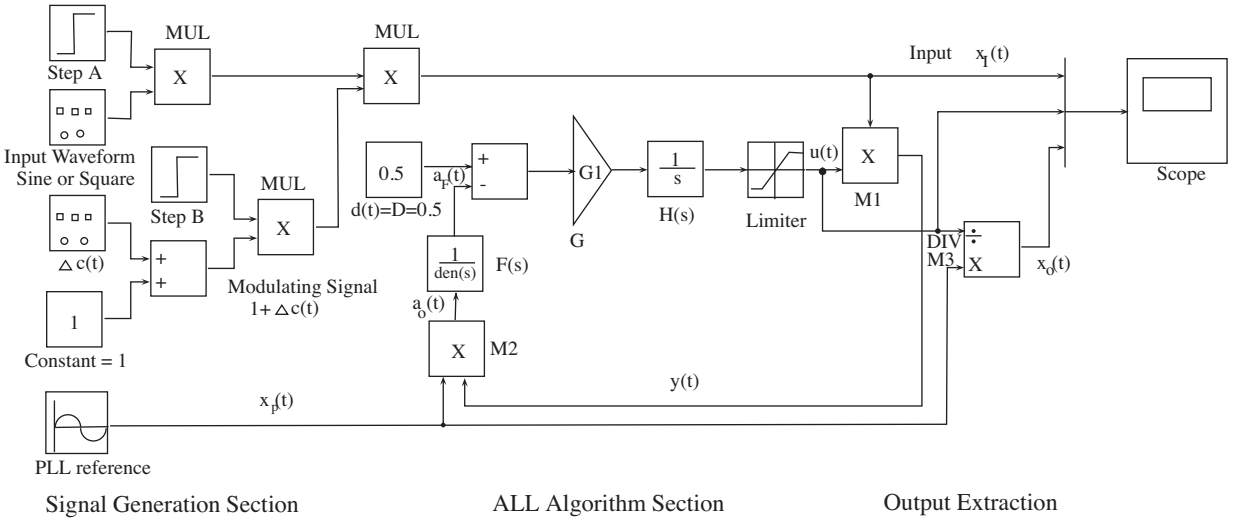

Figure 4. Simulink schematic of the algorithm.

Now the same simulation profile used for the off-line simulation is used also for the emulation of the algorithm. However, the duration of simulation is set to infinity so that the real-time emulation results can be observed in a continuous time. The results are observed and any fine-tuning of the system parameters is done if found necessary. The final values of the system parameters arrived at are reported, which can be used for implementation in a μC, μP, DSP, DSC, or FPGA. The whole process is depicted in Figure 5.

7. Experimental setup

The experimental setup is shown schematically in Figure 6. The algorithm to be verified in real-time is first simulated in MATLAB and Simulink. Input and output ports are appropriately selected from the RTI Library for the DS1006 modular system. The ADC and DAC cards provide the necessary interface to the external system. MATLAB generates the code for the system, which is downloaded into the processor in the DS1006 board.

In Figure 7, a snapshot of the hardware setup used for real-time experiments is shown. It consists of the dSPACE modular system, the connector panels, the computer system, and the digital storage oscilloscope.

8. Real-time experiment

Extensive emulation studies were conducted on the scheme under consideration using MATLAB and Simulink. Various features of the system with $\omega_c = 70\pi$ rad/s, $Q = 0.85$, and $G = 118$ were explored for different inputs. A lower limit of 0.2 for u ($\equiv 5$ for C_1) was set. The model of the algorithm is run on DS1006 with fixed step sizes of 20 μs, 50 μs, and 100 μs. With minor tuning of the controller parameters for each case, all cases gave satisfactory results, comparable with the off-line simulation results presented in [10]. The results for 50 μs are presented in Figures 8 through 13. All results presented (except for Figure 11, which was computed off-line) are screenshots directly obtained from the Control-Desk of the modular system.

8.1. Response to step input: amplitude C_1

A square-wave input of unit amplitude with a frequency of $\omega_o = 100\pi$ rad/s is applied to the circuit at $t = 0.1$ s. The settling time indicated by the signals u and c is captured in Figure 8. It can be observed that the

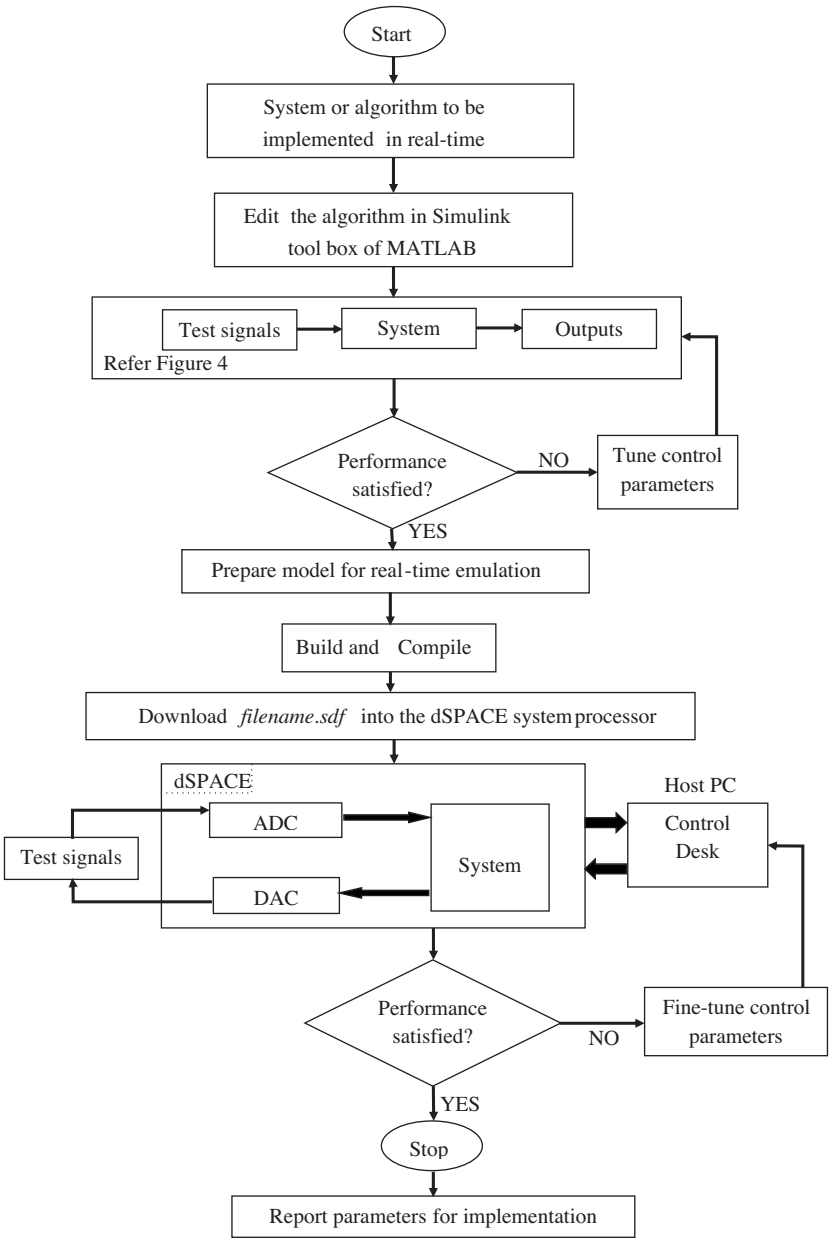

Figure 5. Schematic for the process of validating the algorithm on a real-time hardware-emulation platform.

system settles to a final value within around 1.5 cycles, which is much better than the existing schemes. The expected fundamental amplitude ($=\frac{4}{\pi}$) is also plotted to show the settling time.

8.2. Response to step input: fundamental $x_1(t)$

For the square-wave input, the fundamental extracted signal along with the input is shown in Figure 9. It can be clearly seen that the fundamental settles within 1.5 cycles. This is as good as the best algorithms in this class. It is also to be noted that the extracted fundamental is in synchronization with respect to the input square-wave and the steady-state error is less than 1.5%.

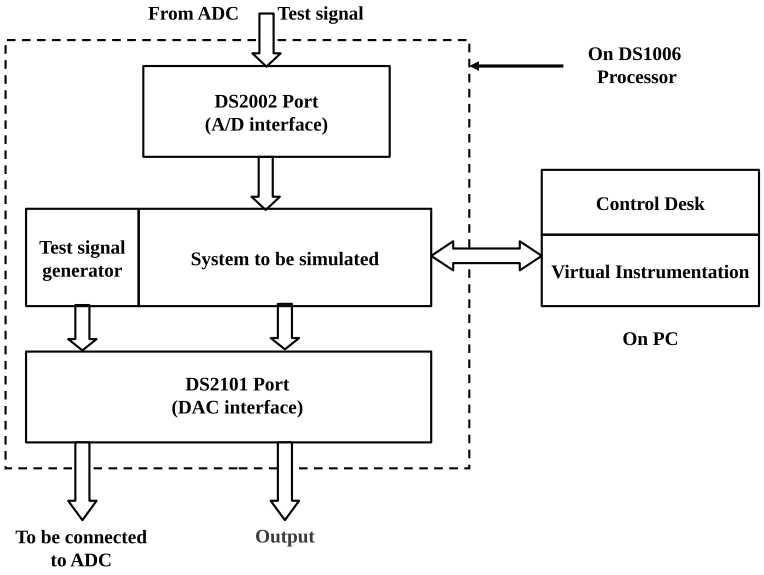

Figure 6. Real-time experimental setup schematic.

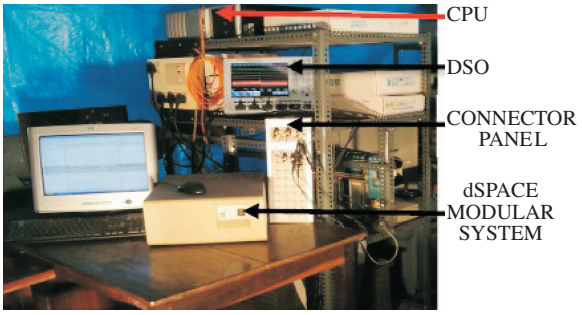

Figure 7. Experimental setup.

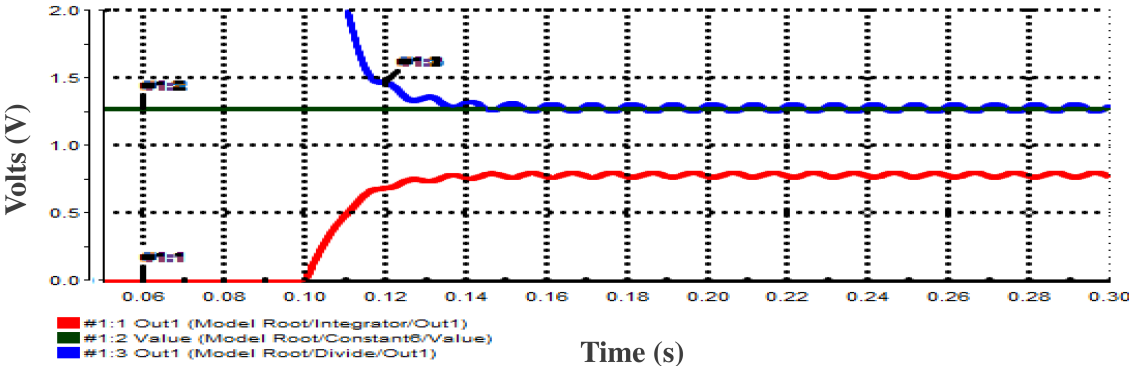

Figure 8. Step response showing settling time for a square-wave input: 1:1 $\leftrightarrow$ u, 1:2 $\leftrightarrow$ $\frac{4}{\pi}$, 1:3 $\leftrightarrow$ c.

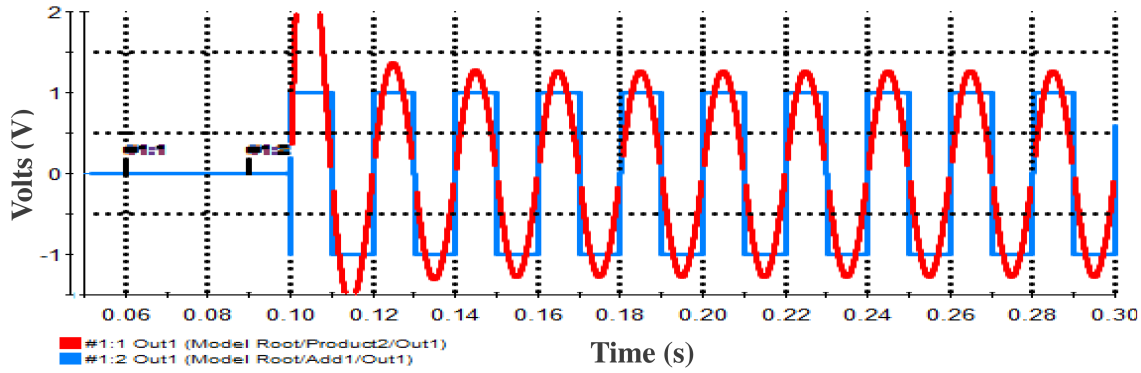

Figure 9. Output for a square-wave input: blue – input; red – output.

8.3. Response to step input: harmonics in $x_I(t)$

The harmonic component extracted from the square-wave is shown in Figure 10. The discrete Fourier series (DFS) of the input square-wave ($x_I(t)$) and the output ($x_O(t)$) are obtained to compare the accuracy of filtering. The DFS is computed using the fast Fourier transform (FFT) algorithm available in MATLAB. It is obtained for one full cycle of the input and output. The magnitude of the input and output are plotted in Figure 11. It is observed that the 2 plots are exactly overlapping or have the same magnitude at the fundamental frequency and the output is completely clean from the harmonics. This is a measure of the good quality of filtering by the scheme.

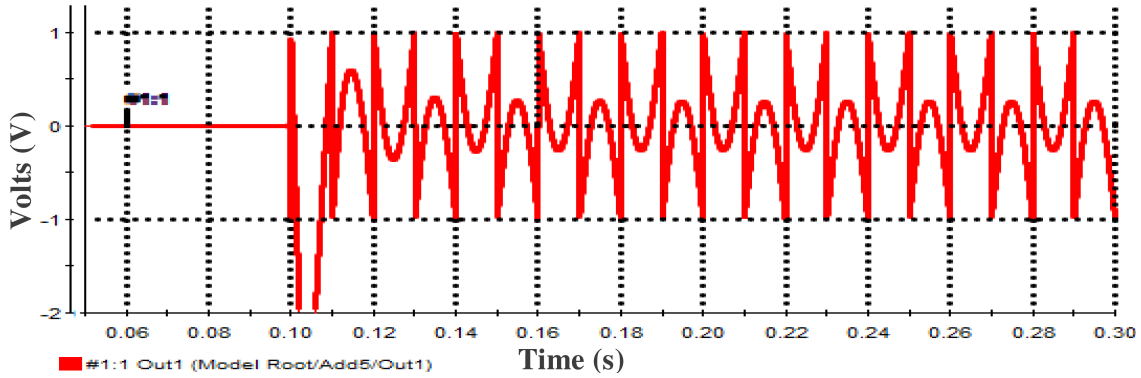

Figure 10. Harmonics extracted from a square-wave input.

8.4. Response to input amplitude modulation

The performance of the scheme is verified for tracking a slow-varying amplitude, as might happen in a power system. This feature is also important for amplitude demodulation in other applications, like in signal-communication systems. The input considered is:

$$x_I(t) \quad = \quad [1 + \Delta c(t)]\sin(100\pi t) = [1 + 0.2\sin(4\pi t)]\sin(100\pi t). \tag{18}$$

The modulating signal $\Delta c(t)$ $(=0.2\sin(4\pi t)$ having a relative depth of modulation of 0.2 units and a frequency of modulation of 2 Hz) is modulating the 50 Hz signal with an amplitude of 1.0 unit. Figure 12 displays an area

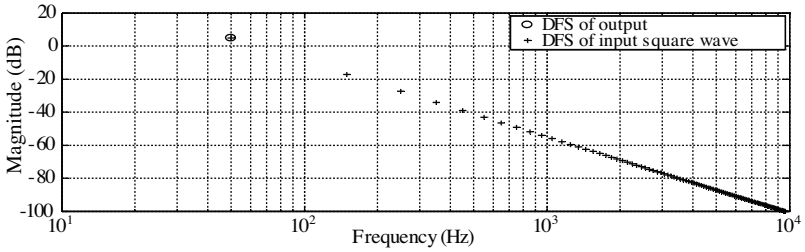

Figure 11. Frequency spectrum of the input and output depicting the total absence of harmonics (**O**) in the output and exact matching of amplitudes of the input (+) with that of the output (**O**) at the fundamental frequency (50 Hz).

zoomed to around one cycle of the modulating signal in order to clearly show the negligible phase lag between the output and the input modulating signal. It should be noted that the phase angle must be measured between the input and the average of the output, where the ripples due to 50 ± 2 Hz should vanish. This shows the good tracking ability of the algorithm and is on a par with the competing algorithms in this category.

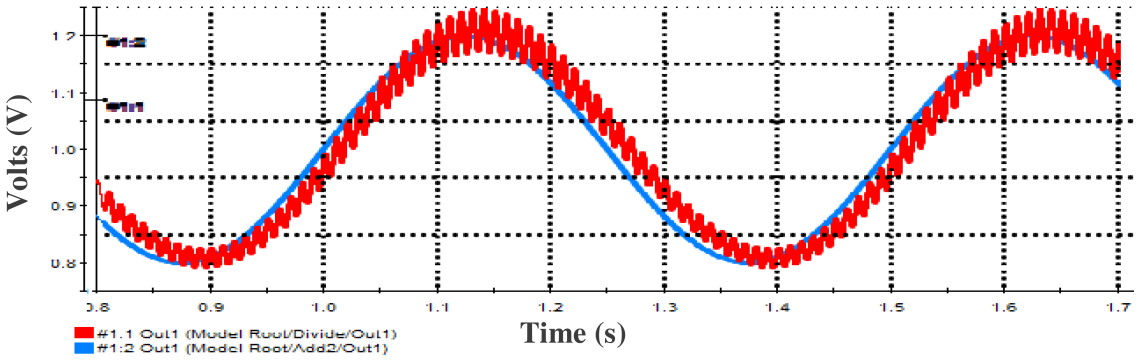

Figure 12. Output amplitude to an input whose amplitude is modulated: blue – input; red – output.

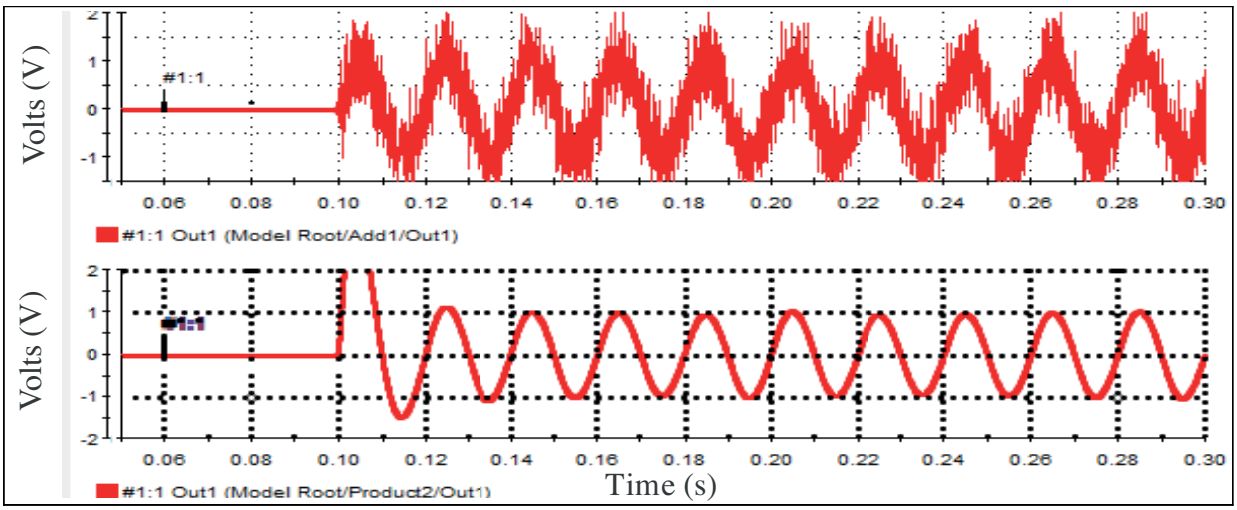

Figure 13. Performance under noisy conditions: top – noisy input; bottom – filtered output.

8.5. Noise rejection

Experiments are conducted to assess the noise rejection capabilities of the ALL. Figure 13 shows the input, which is a sine wave of 100π rad/s with band-limited white noise, and the corresponding output. For an input with signal-to-noise-and-distortion ratio (SNDR) of 6 dB, the SNDR of the output was observed to be 36.3 dB. It can be clearly seen that the noise is severely snubbed in the output. It works very well under noisy environments.

9. Conclusions

It was proposed to validate the real-time implementation of a novel ALL for extracting the harmonics and/or the fundamental from a distorted periodic signal. The scheme is an adaptation of Pettigrew's ALL. The mathematical framework to substantiate the extension and a modification in the form of adding a SOLF for improving the performance were presented. An intuitive explanation for the working of the scheme was presented. The design parameters G, ω_c, and Q were identified. A procedure for selection of these parameters was explained. The features of the proposed scheme are as follows: it has features similar to a PLL; its performance can be optimized by appropriate choice of G, ω_c, and Q; its transient response is good; and it has excellent insensitivity to harmonics and noise. It can find applications in synchronous separation of fundamental and harmonics in a distorted periodic waveform (single-phase or 3-phase), amplitude and frequency tracking, noise rejection, and amplitude demodulation or peak detection.

The algorithm was implemented on a real-time emulation platform, the dSPACE modular system based on DS1006, and the results of the tests were presented. It was found that the output settles to within 1.5% error as early as 1.5 cycles and it remains in synchronism with the input. The DFS of the output showed very little trace of harmonics. The noise-rejection property of the algorithm was tested with a noisy input signal having an SNDR of 6 dB. It was found that the SNDR of the output is 36.3 dB, proving its applicability in a noisy environment. The modulation performance was tested using a modulating signal with a modulation frequency of 2 Hz and a relative depth of modulation of 0.2 units. It was found that the output very closely follows the input, both in magnitude and phase.

Its real-time implementation responses compare very favorably with the results of its off-line simulation. Hence, the possibility of its application to active power filtering of harmonics in power systems stands confirmed. This scheme has all the good qualities of the schemes presented so far and has an almost ideal harmonic filtering capability, as can be seen in Figure 11, which is an additional advantage. It has a slightly higher computational burden because of the extra multipliers required. However, this is not of serious concern in comparison with the improved desirable performance parameters that the scheme offers (in particular, an attenuation of harmonics to as low as –100 dB over the frequency range of interest), especially when the implementation is on a digital platform, as is the current trend.

Acknowledgments

The first author would like to thank Mr I. R. Rao, a colleague in the Department of Electrical Engineering at NITK Surathkal, for rendering assistance in manuscript editing. He also acknowledges Mr Ritesh A. Bhat, who was an undergraduate student in the department during the academic year 2009–2010. The authors gratefully acknowledge the painstaking review and the constructive suggestions for improving the quality of this paper.

References

[1] J.M. Gonda, V.A. Adithya, S. David, "Performance analysis of compensation current extraction techniques for 3ϕ, $3-$wire shunt active power filter under unbalanced supply", Proceedings of IEEE International Conference on Power Systems, 2009.

[2] E. Watanabe, H. Akagi, M. Aredes, Instantaneous Power Theory and Applications to Power Conditioning, New Jersey, John Wiley and Sons, 2007.

[3] T.J. Moir, "Analysis of an amplitude-locked loop", Electronics Letters, Vol. 31, pp. 694–695, 1995.

[4] S. Luo, Z. Hou, "An adaptive detecting method for harmonic and reactive currents", IEEE Transactions on Industrial Electronics, Vol. 42, pp. 85–89, 1995.

[5] H. Karimi, M. Karimi-Ghartemani, M.R. Iravani, "An adaptive filter for synchronous extraction of harmonics and distortions", IEEE Transactions on Power Delivery, Vol. 18, pp. 1350–1356, 2003.

[6] M.K. DaSilva, "Vector locked loop", US Patent No. 5,105,168, 1991.

[7] M. Karimi-Ghartemani, M.R. Iravani, "A nonlinear adaptive filter for online signal analysis in power systems: applications", IEEE Transactions on Power Delivery, Vol. 17, pp. 617–622, 2002.

[8] A.M. Pettigrew, "Amplitude-locked loop circuits", UK Patent No. 5,341,106, 1994.

[9] R.A. Bhat, J.M. Gonda, S. David, "A novel vector-locked loop scheme for synchronized extraction of harmonics", Proceedings of the Fifth International Conference on Information and Automation for Sustainability, pp. 447–452, 2010.

[10] J. M. Gonda, R. A. Bhat, S. David, "Adapting Pettigrew's amplitude-locked loop for fast and synchronized extraction of fundamental and harmonics", Proceedings of the Third International Conference on Power Electronics and Intelligent Transportation System,Vol. 4, pp. 341–344, 2010.

[11] dSPACE GmbH, dSPACE-Manual for Modular System Based on D1006, Germany, dSPACE GmbH, 2005.

[12] W.F. Egan, Phase-Lock Basics, New Jersey, John Wiley and Sons, 2007.