Intuitive Real-Times Platform for Audio Signal Processing and Musical Instrument Response Emulation

Patrick Gaydecki and Sheheera Ismail
School of Electrical and Electronic Engineering
University of Manchester
Manchester M13 9PL, United Kingdom
patrick.gaydecki@manchester.ac.uk

***Abstract*— In recent years, the DSP group at the University of Manchester has developed a range of DSP platforms for real-time filtering and processing of acoustic signals. These include *Signal Wizard 2.5*, *Signal Wizard 3* and *Vsound*. These incorporate processors operating at 100 million multiplication-accumulations per second (MMACs) for SW 2.5 and 600 MMACS for SW 3 and *Vsound*. SW 3 features six input and eight output analogue channels, digital input/output in the form of S/PDIF and a USB interface. For all devices, The software allows the user, with no knowledge of filter theory or programming, to design and run standard or completely arbitrary FIR, IIR and adaptive filters. Processing tasks are specified using the graphical icon based interface. In addition, the system has the capability to emulate in real-time linear system behavior such as sensors, instrument bodies, string vibrations, resonant spaces and electrical networks. Tests have confirmed a high degree of fidelity between the behavior of the physical system and its digitally emulated counterpart. In addition to the supplied software, the user may also program the system using a variety of commercial packages via the JTAG interface.**

Keywords—arbitrary filter; intuitive DSP; real-time linear systems DSP

I. INTRODUCTION

In previous publications, the design was described of real-time DSP systems for the automatic design of FIR, IIR and adaptive filters, and for emulating in real time the behavior of arbitrary linear systems, such as analogue circuits, sensor responses or acoustic instrument bodies [1, 2]. In this paper, several key advances are detailed that relate to all aspects of the system, including the hardware module, the high-level user interface and the real-time firmware. Of particular importance is the ability of the new system to accommodate analogue network responses, phase shift blocks and impulse or frequency responses of physical acoustic systems. The high level interface allows the user to specify signal processing operations by clicking the appropriate icon and entering the desired parameters. *Signal Wizard 2.5* is a dual channel, 100 MMAC device, *Signal Wizard 3* operates at 600 MMACS and features six input and eight output analogue channels, digital audio input/output in the form of S/PDIF and a USB interface. *Vsound* also operates at 600 MMACS and is a system intended solely for use with electric violins. In each case, the software allows the user, with no knowledge of filter theory or programming, to design and run standard or completely arbitrary finite impulse response (FIR), infinite impulse response (IIR) and adaptive filters. The systems have the capability to emulate in real-time linear system behavior such as sensor systems, instrument bodies, vibrations, resonant spaces and electrical networks. They are particularly suited to the processing of multichannel audio and music signals.

II. SIGNAL PROCESSING REPERTOIRE

A signal processing operation is specified by locating an appropriate icon within the graphical design area. Broadly, the operations fall into two categories: simple processing functions involving a single scalar quantity applied to a signal, and more complex vector processing operations including filtering and transforms. The simple signal processing repertoire includes multiplication by a constant (gain adjustment), addition of signals, multiplication of signals (modulation) and time delaying. Vector operations include FIR and IIR filtering, adaptive filtering, real-time Fourier and Hilbert transforms, quadrature signal processing and IIR to FIR translation.

A. Basic linear filter theory and algorithmic implementation

The (linear) process of filtering in time t is encapsulated in the convolution integral

$$y(t) = \int_{-\infty}^{\infty} h(\tau)x(t-\tau)d\tau \tag{1}$$

where $y(t)$ is the output (filtered) signal, $x(t)$ is the incoming signal, τ is the time-shift operator and $h(\tau)$ is the impulse response of the filter [3]. In discrete space, this equation may be implemented using either an FIR or IIR solution. In the case of the FIR filter, the infinite response is truncated, which yields an expression of the form

$$y[n] = \sum_{k=0}^{M-1} h[k]x[n-k] \tag{2}$$

with the z-transform of the impulse response, i.e. the transfer function $H(z)$, being given by

$$H(z) = \frac{Y(z)}{X(z)} = \sum_{k=0}^{\infty} h[k]z^{-k} \tag{3}$$

The basic design problem is therefore reduced to finding the appropriate impulse response $h[k]$ which yields the desired frequency response. The system here exploits the *frequency sampling method*, which proceeds as follows:

- The frequency response of the filter is specified in the Fourier-domain. We stress that this may, if desired, be an ideal brick-wall type, but it need not be so. It could equally be any arbitrary shape. For a linear phase filter, the frequency response is determined by setting the real terms to their intended values, and leaving the imaginary terms as zero.
- The inverse Fourier transform is taken of the designated frequency response, which generates a time-domain function. This is usually done with an inverse FFT. Because of the mechanics of its operation, this will result initially in a time domain signal which is not centered, i.e. the right-hand part of the signal will start at $t = 0$, and the left-hand part will extend backwards from the final value. Hence manipulation is necessary to centre the impulse response.
- The ends of the impulse response must now be tapered to zero using a suitable window function, such as a Hanning type. Application of the window minimizes ripples in the pass and stop bands but it also increases the width of the transition zone.

For a simple linear phase pass-band filter, the frequency sampling method is encapsulated by the expression

$$h[n] = f_w[n]F^{-1}\{H[k]\} \qquad \begin{cases} H_r[k] = 1, & f_l < k < f_h \\ H_r[k] = 0, & \text{elsewhere} \\ H_i[k] = 0 \end{cases} \tag{4}$$

where $f_w[n]$ denotes the window function. For a filter with an arbitrary frequency response, the method is adapted so that

$$h[n] = f_w[n]F^{-1}\{H[k]\} \qquad \begin{cases} H_r[k] = a_k \\ H_i[k] = b_k \end{cases} \tag{5}$$

B. IIR design using the bilinear z-transform and pole-zero placement.

In contrast, IIR filters rely on recurrence formulae, where the output signal is given by

$$y[n] = \sum_{k=0}^{N} a[k]x[n-k] - \sum_{k=1}^{M} b[k]y[n-k] \tag{6}$$

In this case therefore, the transfer function is

$$H(z) = \frac{a[0] + a[1]z^{-1} + \ldots + a[m]z^{-m}}{1 + b[1]z^{-1} + \ldots + b[n]z^{-n}} = \frac{\sum_{m=0}^{M} a[m]z^{-m}}{1 + \sum_{n=1}^{N} b[n]z^{-n}} \tag{7}$$

The system allows the user to design filters using either the bilinear z-transform (BZT) method or the pole-zero placement method. The former technique is implemented by a module of the software containing a library of pre-defined passive analogue arrangements, based on capacitors, resistors and inductors, and whose Laplace descriptions are transposed by the software into the z-plane. The system allows the user to express digital filters based on passive equivalents, or on active equivalents such as Butterworth and Chebyshev low pass, high pass, band pass or band stop designs.

The pole zero placement method is, in contrast, an essentially interactive method, whereby the designer positions poles and zeros on the unit circle to affect a desired response. The software allows the user to locate as many as one hundred complex conjugate pairs of poles and zeros, i.e. in essence facilitating the realization of arbitrary IIR frequency response.

C. Inverse filtering

The output $y(t)$ of any linear system may be modeled by considering the impulse response $h(t)$, together with the input signal $x(t)$. In such a system, the output is convolution operation between the input signal and the impulse response, i.e.

$$y(t) = x(t) * h(t) \tag{8}$$

The presence of noise is a significant factor in determining the efficacy of the inverse filtering procedure; if, as is often the case, the noise is independent of the signal, then Equation (8) becomes

$$y(t) = x(t) * h(t) + s(t) \tag{9}$$

Inspection of Equations (8) and (9) shows that to conduct deconvolution in the time domain, the impulse response of the inverse filter must be pre-computed, and then applied to the output signal using the normal time convolution equation. Typically,

$$\tilde{h}(t) = F^{-1}\left[\frac{1}{H(\omega) + s}\right] \tag{10}$$

Hence

$$x(t) = y(t) * \tilde{h}(t) \tag{11}$$

This approach is adopted by the present system and has proven very useful for both audio reconstruction and loudspeaker equalization.

D. Adaptive filtering

The real-time systems incorporate an adaptive filter that is expressed as a least mean square (LMS). This is a successive-approximation technique that obtains the optimal filter coefficients required for the minimization of the error, E. The theoretical validation of the adaptive filter is lengthy and for the purpose of brevity it is not included here. It is described fully by Gaydecki in [4]. It is implemented as follows. Initially, the coefficients of the filter will be any arbitrary value; by convention, they are usually set to zero. Each new input signal value $x[n]$ generates an output signal value $y[n]$, from which is generated the error signal value, $e[n]$, by subtracting it from a desired signal value, $d[n]$. The value of $h[k]$ is then modified by the expression

$$h_{new}[k] = h_{old}[k] + \Delta e[n]x[n-k] \tag{12}$$

In other words, the new value of $h[k]$ is calculated by adding to its present value a fraction of the error signal $\Delta e[n]$, multiplied by the input signal value $x[n-k]$, where $x[n-k]$ is the input signal located at the k^{th} tap of the filter at time n. For reasons of stability, it is important that only a fraction of $e[n]$ is added. If the rate of convergence is too rapid, the algorithm will over-shoot and become unstable. The LMS algorithm is

not the only method employed in the implementation of adaptive filters; neither is it the fastest to converge to the optimum value for a given filter coefficient. However, the LMS method is readily programmed for real-time environments and very stable.

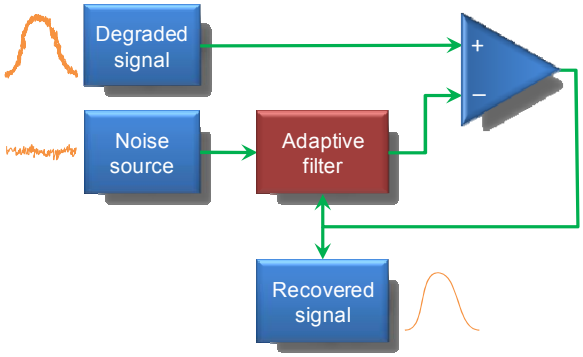

Fig. 1. The dual input (true) adaptive filter.

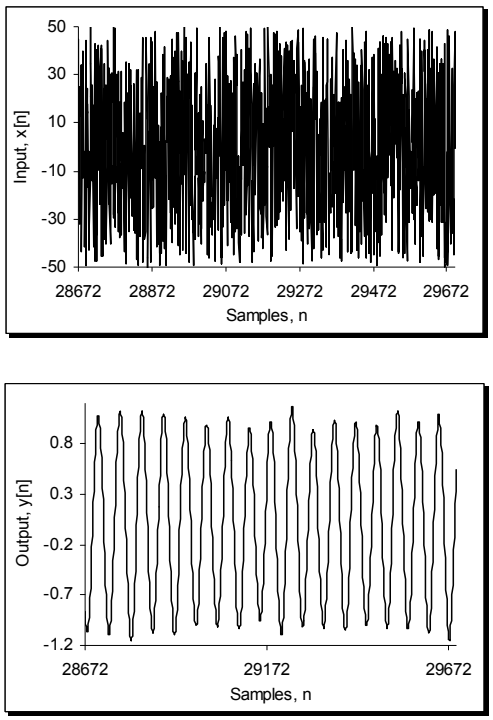

Fig. 2. Input (upper) and output (lower) signals from a true adaptive filter operated in real time by the system.

The true adaptive filter, employed here and shown in Figure 1, assumes that the source of the noise is available as an input to the system. Figure 2 shows typical input and output signal traces (after convergence) with a peak-to-peak input signal-to-noise ratio of 1:100. In this instance, the original SNR was of the order of -40 dB; after filtering, the SNR was approximately 29 dB, representing an improvement of 69 dB. Such a large change is not, however, always possible, depending strongly on the degree of correlation between the pure noise input and the noise present in the narrow band signal. With these DSP systems the principal application of the adaptive filter is for broadband noise cancellation, in which the bandwidth of the noise encroaches on that of the signal.

E. Sine wave synthesis

Signal Wizard 2.5 exploits interpolation to generate individual sine waves, and for complex multi-frequency signals it employs a bank of parallel interpolators, the outputs of which are summed to produce the final composite waveform. This method is far more accurate than the look-up table approach and provides much greater flexibility and frequency resolution. Compared with the look-up table method, it uses minimal memory resource; specifically, for each sine wave, it requires five 24-bit words to store the polynomial coefficients and a single word to store the time increment. In comparison, a single look-up table may comprise several hundred or thousand values. Since the interpolation system calculates the output in real-time rather than using stored values, it is ideally suited to digital-function generators, in which the frequency may be changed at any point by the user with no perceptible delay.

Its performance was evaluated by making Total Harmonic Distortion (THD) measurements for several spot frequencies, both for the digital interpolation system and, for comparison, a conventional bench-top analogue function generator (Thurlby Thandar model TG230). In each case, the sine waves were digitized and analyzed by a high-precision waveform analyzer with 24-bit precision. Table 1 confirms that in every case, the performance of the digital interpolation system was superior to that of the analogue function generator.

TABLE I. THD FIGURES FOR DIGITAL VERSUS ANALOGUE SINE WAVE GENERATION

System	**% THD @ 2 kHz**	**% THD @ 3 kHz**	***% THD @ 5 kHz***
Digital	0.057	0.084	*0.334*
Analogue	*0.663*	*0.861*	*17.55*

F. Real-time violin emulation

A variant of the Signal Wizard platform, called *Vsound*, has been developed to process the raw electrical output from an electric violin to produce an output signal which, when fed to an amplifier and loudspeaker, approximates closely the timbre of an acoustic instrument. *Vsound* acts like the body of an acoustic violin. It takes the saw tooth waveform produced by the electric instrument, and modifies it using an in-built algorithm, to produce an output that is strikingly similar to that of a wooden instrument. The modification of the input is performed by a very powerful processor which is fast enough to respond the incoming signal in real time. The device holds in its memory up to fifty far-field impulse responses of wooden instruments, any one of which may be convolved in real-time with the input signal to synthesize the modified output signal. This first stage convolution is realized as an FIR structure. In addition, the device incorporates a parametric equalizer, blender, reverberation unit and a gain adjustment function. The user interface includes a set of navigator-style buttons and a simple display screen. The device may be connected to a computer or tablet and controlled from a separate software package. Preliminary trials with a professional violinist have

confirmed that the system improves significantly the tonal quality of the raw string force signal.

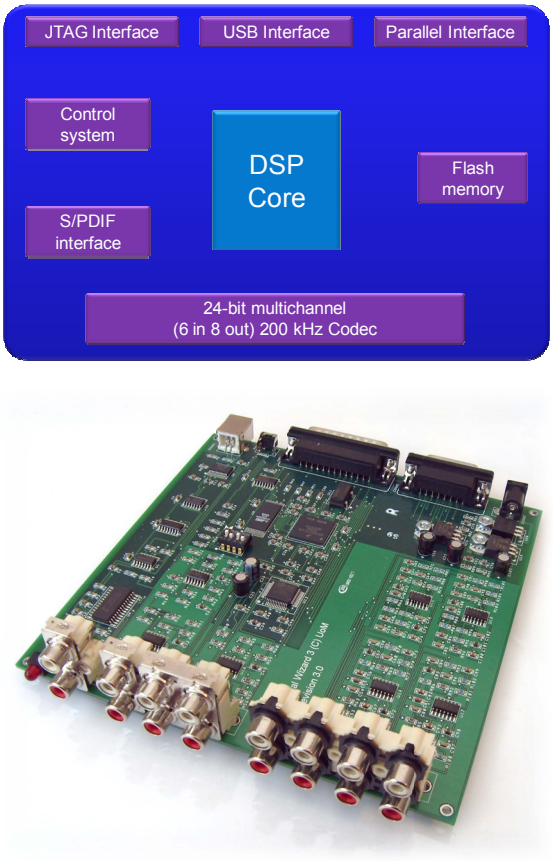

Fig. 3. System model (upper) and main circuit board (lower).

G. *Other functions*

In addition to the filtering operations described above, the various Signal Wizard platforms incorporate many other useful audio signal processing functions, including delay, arbitrary mixing, chorusing, beam forming, microphone array processing, echo, reverberation, musical effects and arbitrary waveform synthesis.

III. System Hardware and Software Architecture

The various hardware and software platforms described here, i.e. *Signal Wizard 2.5, Signal Wizard 3* and *Vsound*, have been developed within the School of Electrical and Electronic Engineering at the University of Manchester.

A. *Signal Wizard 2.5*

Signal Wizard 2.5 is a dual channel 24-bit codec device and employs a Freescale DSP56309 operating at 100 MMACS as its core DSP. It communicates with the host computer using a standard serial link operating at 115.2 k bits / second. The integrated software allows the user to design, download and execute real time-filters with a few mouse clicks.

B. *Signal Wizard 3*

Signal Wizard 3 features a DSP56321 operating at 600 MMACS. Audio-bandwidth processing, involving in particular real-time sensor signal processing, sensor response emulation, musical instrument emulation and ambient sound field characterization, often requires inputs from multiple sources. *Signal Wizard 3*, with its multiple inputs and outputs, is ideally suited to these tasks. The improvement in speed has been made possible primarily by the incorporation into the device of an enhanced filter coprocessor (EFCOP). The system also incorporates flash memory to hold both the operating system and the coefficients of filters designed using the high level software interface.

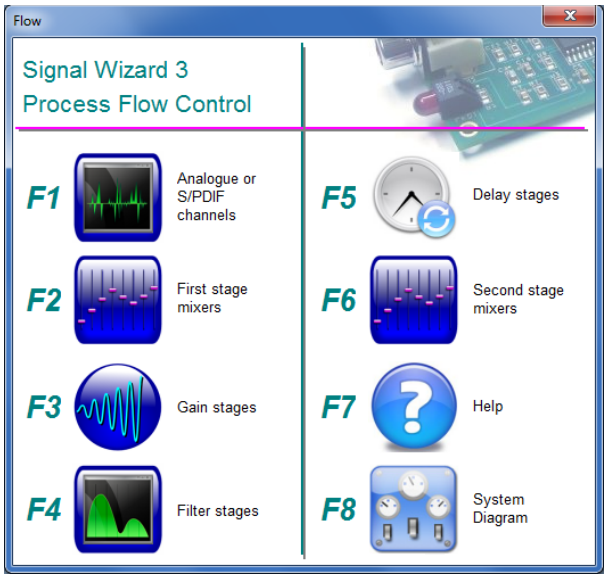

Fig. 4. Main operations window of *Signal Wizard 3*.

It includes a CS42438 codec that supports six input channels, eight output channels with a maximum sample frequency of 192 kHz. Additionally, digital audio inputs and outputs are provided in the form of an SPDIF interface unit. This obviates the need for analogue to digital and digital to analogue conversion, thereby precluding the introduction of noise in the signal transmission stages. The system communicates with the host computer via a high-speed USB link, improving data transmission rates for real-time spectrum analysis and signal capture. A further critical refinement of this system is the incorporation of a JTAG [5] interface; this allows users to develop their own software, using 3rd party development tools such as the Freescale assembler and hardware debugger [6], or the Tasking C compiler [7]. A schematic of the new hardware is shown in Figure 3, together with a photograph of the board.

The new hardware is accompanied by a radically different user interface, which allows for much greater flexibility with regard to the sequence of processing operations that may be conducted in real time. As shown by Figure 4, a sequence of algorithms is built up using the graphical design window, into which the operator places icons that represent the functions that are to be executed. These icons are linked by lines that represent the signal paths, and may be connected to the input channels, other icons and eventually to the output channels in any desired sequence.

C. *Vsound*

For *Vsound*, the output from the electric violin string pickup (normally a piezo-electric transducer) is fed to a high-impedance (5 MΩ) gain-switchable preamplifier and then to the analogue-to-digital converter section of a 24-bit codec sampling at 50 kHz. The output from the codec is then fed to the DSP device that performs all of the main-stage processing, i.e. body impulse response convolution, parametric equalization, blending (dry/wet mixing), reverberation and final volume control. The processing chain is depicted in Figure 5. After processing, the output signal is fed back to the digital-to-analogue section of the codec and from there to appropriate buffers to drive both high power audio amplifiers and headphones. Other sub-systems include a display, keypad and interface to allow the unit to be connected to and programmed by a computer. The final device is shown in Figure 6. Convolution of signal with a long-duration impulse response is a compute-intensive operation. In the case of signals sampled within the audio band, it places a severe constraint on the specifications of the DSP device which is selected to perform the task. In this case, for example, the impulse response comprises 4096 coefficients. With a sample rate of 50 kHz, this alone represents a processing burden of 204.8 MMACs. In addition, the processing core must also accommodate the twenty IIR filters in the equalizer, the seven comb / low-pass filters and phase scrambler of the reverb unit, a master gain and the blender function. Modern DSP devices achieve the speeds required with the use of Harvard architecture, hardware multipliers and enhanced filter coprocessors, which are optimized to perform seamless multiplication and accumulation. Typically, such systems perform convolution at close to or equal 100% efficiency (i.e. two MMACs per clock cycle when using both the DSP core and the coprocessor).

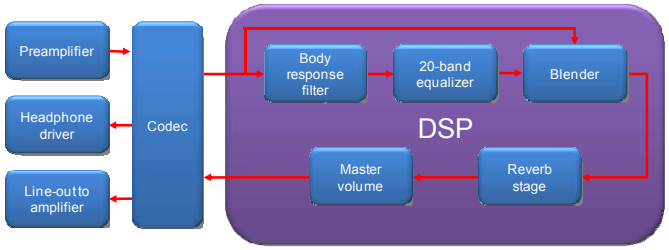

Fig. 5. Main system components and signal processing chain.

IV. Discussion

Over the last twenty years, DSP has evolved from an obscure discipline into a technology that is pivotal to the infrastructure of society, our personal lives and scientific endeavors. It lies at the heart of almost every modern electronic digital device, so the question naturally arises: is it true that digital implies high quality, better than could be obtained from traditional analogue techniques? The answer to the last question is yes, but why this is so may not always be obvious. Strictly speaking, DSP involves manipulation of signals that have their origins in the analogue world. Such signals may be produced for example, by video, audio, radio telemetry, radar, thermal, magnetic or ultrasonic sensor systems, to name but a few from a truly enormous range of devices and instruments.

Fig. 6. Complete *Vsound* device showing display screen and navigator buttons

Application areas are expanding rapidly [8], but the one constant and perhaps most significant barrier to learning is the investment in terms of time and intellectual effort required to understand the functions and instruction set of a particular device, construct the system, and write the algorithms. This cycle can take many months. Contrast this with designing and fabricating a 2nd order analogue filter based on two resistors, two capacitors and one op-amp, a process that might take fifteen minutes. Perhaps for this reason, scientists and engineers who wish to use a particular filter will first attempt an analogue solution. In contrast, DSP algorithms, such as filters, tend to be used by individuals who are both familiar and comfortable with the art of DSP, in terms of the electronics, coding and mathematics.

Signal Wizard Systems were developed to address these obstacles. They are complete, integrated devices for real-time and offline digital signal processing (DSP) without the need for specialist know-how, mathematics or theory. They comprise advanced, purpose-designed hardware modules that implement the DSP operations in real time, and software that controls the hardware according to the user's requirements. Most important, the systems require minimal knowledge of digital signal processing (DSP) theory on the part of the user and none of the mathematics associated with digital filter design. Filters and other algorithms [9, 10] can be designed in seconds, downloaded and executed with just a few mouse clicks. Given these attributes, they are ideally suited not only as research tools but as teaching aids at both undergraduate and postgraduate level. For this specific purpose, a DSP laboratory teaching package has been developed using the Signal Wizard range of devices, and has proven an invaluable tool for training our student cohort in the practical aspects of DSP engineering design and programming.

V. Concluding Remarks

A range of versatile, simple to use and powerful real-time DSP systems have been developed that incorporate purpose designed, high-speed processing boards and graphical Windows-based software that interfaces seamlessly with the

hardware. This software allows a designer to specify an arbitrary series of algorithms with minimum effort. Once designed, the sequence of operations may be downloaded as instructions to the hardware and executed on demand.

REFERENCES

[1] P. Gaydecki, J Woodhouse and I Cross, "Advances in audio signal enhancement and stringed instrument emulation using real-time DSP". Proc. Forum Acusticum Sevilla 2002 (on CD; ISBN: 84-87985-06-8).

[2] P. Gaydecki and B Fernandes , "An advanced real-time DSP system for linear systems emulation, with special emphasis on network and acoustic response characterisation" Measurement Science and Technology vol. 14, pp. 1944-1954, 2003.

[3] E. C. Ifeachor and B. W. Jervis, Digital Signal Processing: a Practical Approach, ISBN-10: 0201596199, Prentice Hall, 2001.

[4] P. Gaydecki, Foundations of Digital Signal Processing: theory, algorithms and hardware design, ISBN: 0852964315, IET Publishing, 2004.

[5] DSP56321 Reference manual Freescale Semiconductor Literature Distribution Centre, P.O. Box 5405, Denver, Colorado 80217, 2005.

[6] DSP56321EVM user's manual Freescale Semiconductor Literature Distribution Centre, P.O. Box 5405, Denver, Colorado 80217, 2005.

[7] Tasking Data Sheet: Motorola DSP56xxx Software Development Toolset, 2012

[8] R. G. Lyons, Streamlining digital signal processing, ISBN: 0470131578, Wiley-Interscience, 2007.

[9] P. Gaydecki, "New real-time algorithms for arbitrary, high precision function generation with applications to acoustic transducer excitation.", J. Phys.: Conf. Ser. 178 012015, 2009.

[10] Dai Xiang-ming, "Implementation of arbitrary function generator (AFG) with linear interpolation", Journal of Electron Devices, vol. 31, pp. 1397-1400, 2008.