Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

FPGA-based implementation of an active vibration controller

Alberto Leva *, **Luigi Piroddi** *

* *Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy*
{leva,piroddi}@elet.polimi.it

Abstract: This manuscript deals with the implementation of active noise and vibration control systems on very high-speed computational architectures, more specifically on Field Programmable Gate Arrays (FPGA). Starting from a particular application, namely the vibration control of a turbomolecular vacuum pump, general design considerations are devised. A feedforward controller is designed based on multiple adaptive notch filters and the filtered-x LMS algorithm. An implementation of the presented control technique is illustrated, and some tests are reported to show its efficiency.

Keywords: FPGA, Active Noise Control, Active Vibration Control.

1. INTRODUCTION

The active control of noise (ANC) and vibrations (AVC) is a well-established research area (Fuller et al., 1996; Kuo and Morgan, 1996), but few real-world applications are documented. One reason for this is the lack of suitable control system electronics capable of the required computational power and numerical precision and - at the same time - convenient for industrial production standards (Antila, 2004). The newest achievements in electronic devices, such as powerful FPGA-based control systems, may however provide the adequate and cost effective solution to this problem.

Nowadays, FPGAs are typically employed as co-processors for DSP or microcontroller units, for additional processing and logic tasks, while the core signal processing functions are typically demanded to a specialized DSP chip (Antila, 2004). For example, a hybrid DSP-FPGA architecture for ANC is reported in Hashemian (1995). However, the most recent FPGA models are powerful and flexible enough to be used for signal processing as well. In Di Stefano et al. (2005) a dedicated hardware FPGA-based implementation of an ANC system using a modified LMS algorithm is discussed.

This manuscript describes the FPGA-based implementation of a narrowband active noise and vibration controller adopting the filtered-x LMS (fxLMS) adaptive feedforward approach. The implementation was carried out using the National Instruments^{TM} compactRIO® architecture, programmed with the LabVIEW^{TM} development system. The designed system was originally targeted for the vibration control of a turbomolecular vacuum pump, using accelerometers, piezoelectric actuators and a custom damper unit, as discussed in Piroddi et al. (2006a,b, 2007), where some experimental results are also provided. However, the underlying methodology and engineering are definitely more general, hence the scope of the manuscript. Starting from the peculiarities of the class of addressed problems, some general design ideas are introduced and discussed regarding the FPGA implementation of ANC/ANV systems. Heuristic guidelines are provided to deal with the tradeoffs that typically arise when implementing similar systems in general. Also, some considerations on the architectural requirements of controllers of the type presented are reported (the implementation of the controller for the application mentioned above, for three channels and 3-harmonic disturbances, was carried out on a 3 Mgates FPGA). Finally, the validation of the proposed controller (in simulation, so as to guarantee reproducible conditions) allows for some interesting remarks on the interplay between near frequencies to be eliminated, due to the finite precision of the fxLMS weights' adaptation.

2. THE ADDRESSED TYPE OF APPLICATIONS

The particular application that triggered the development of the presented controller was the AVC of a turbomolecular vacuum pump (Piroddi et al., 2006a,b, 2007). Turbomolecular vacuum pumps are used in high vacuum applications, such as electronic microscopes, and operate at high speeds, typically over 40 kRPM and sometimes up to 75 kRPM. Though pump rotors are very carefully balanced (Rava et al., 1987), residual asymmetries cause vibrations (Casaro, 1989). While of small entity, such vibrations can hamper the performance of the high precision equipment to which the pump is attached, and even damage the pump in the long run.

In the quoted reference (Piroddi et al., 2007) an AVC is presented, realised by inserting between the pump and the vacuum system an active damper, in turn composed of a metallic bellow with three axial accelerometers and three axial piezoelectric force actuators, forming three colocated couples with 120° spacing. The work (Piroddi et al., 2007) discusses the efficacy of that controller at reducing vibrations, despite the simple, decentralised structure, and the presence of axial action only.

In this work, the focus is conversely set on the engineering problems pertaining to the implementation of that controller on an FPGA. This allows to address, by some generalisation, the typical issues not only of that particular application, but also of analogous vibration control ones, characterised by

- Essentially (multi-)harmonic nature of the disturbance acting on the system;
- Quite precise knowledge (or measurements) of the spectral contents of the disturbance (in the presented application, the rotor and bearings' rotation frequencies dominate

the disturbance spectrum, and are known with good precision);
- High variability of the disturbed system dynamics or - more in general - severe stability problems, highly advising open-loop control versus, e.g., closed-loop solutions based on anti-notch filters;
- High bandwidth requirements, calling for very high sampling rates;
- Implementation based on FPGAs or similar devices.

3. REVIEW OF THE ADAPTIVE NOTCH FILTER TUNED WITH THE FXLMS ALGORITHM

A single-frequency adaptive notch filter based on the filtered-x LMS (fxLMS) algorithm (Morgan and Thi, 1993; Kuo and Morgan, 1996), can be used to cancel single harmonic disturbances. A sinusoidal reference signal is first generated at the same frequency as the disturbance, which is then suitably modulated in amplitude and phase, so as to follow (and cancel out) the disturbance. More precisely, the generated control signal is given by

$$u(k) = w_0 x_0(k) + w_1 x_1(k), \tag{1}$$

where k is the discrete time, $x_0(k) = sin(k\theta)$, $x_1(k) = cos(k\theta)$, and $\theta = \omega T_s$, ω being the continuous-time frequency of the harmonic disturbance, and T_s the sampling time. To iteratively adapt the weights w_0 and w_1, the fxLMS algorithm employs the relationships

$$\begin{aligned} w_0(k) &= w_0(k-1) + \mu e(k-1) x'_0(k-1) \\ w_1(k) &= w_1(k-1) + \mu e(k-1) x'_1(k-1) \end{aligned} \tag{2}$$

where $x'_0(k)$ and $x'_1(k)$ are, respectively, the responses of the disturbed system to $x_0(k)$ and $x_1(k)$, $e(k)$ is the 'error', i.e., the output of the system in the absence of the disturbance to be cancelled, and μ is a parameter usually termed the 'adaptation gain'. The adaptive notch filter thus requires some knowledge on the frequency response $G(j\omega)$ of that system, since apparently in (2)

$$\begin{aligned} x'_0(k) &= |G(j\omega)| sin(k\theta + \arg(G(j\omega))) \\ x'_1(k) &= |G(j\omega)| cos(k\theta + \arg(G(j\omega))) \end{aligned} \tag{3}$$

However, at a cost of a potentially slower adaptation and of possibly oscillatory behaviors of the weights, the mentioned process knowledge can be quite coarse. In particular, the adaptive algorithm can in principle tolerate up to $\pm 90^\circ$ phase uncertainty, since - roughly - it suffices that the phase uncertainty is not large enough to make the weights' adaptation go in the wrong direction.

If the disturbance to be cancelled is multi-harmonic, several adaptive notch filters may be combined in direct, parallel, direct/parallel, or cascade forms (Kuo and Morgan, 1996). In the following the parallel form has been adopted, where the necessary number of cancellers are connected in parallel, each one devoted to manage a single disturbance frequency (although all such blocks are adapted using the same, multi-harmonic error signal). For full details on the algorithm, including the necessary proofs of convergence, stability and so forth, the interested reader can refer to Morgan and Thi (1993); Kuo and Morgan (1996).

4. THE FPGA IMPLEMENTATION

In the literature, the adaptive notch filter is typically analysed under two assumptions, namely (a) the spectral purity of the generated sine and cosine signals in (1), and (b) infinite-precision computations in the weight updating mechanism (2). If one or both hypotheses are violated, the overall system becomes far more difficult to analyse, since potentially significant nonlinearities arise. While these hypotheses are realistic for DSP implementations, they turn out to be critical to meet when architectures like an FPGA are used for the signal processing tasks.

In an FPGA environment the sinusoid signal is not a built-in primitive and signal generation mechanisms like lookup tables are frequently adopted (Vankka and Halonen, 2001). Unfortunately, such mechanisms do not provide an adequate sine/cosine accuracy for precise harmonic disturbance rejection. Actually, it appears that the spurious frequencies generated by a lookup table sinusoidal generator may significantly limit the control performance and even hamper its correct behavior.

In principle, signal accuracy can be enhanced through the use of *very* high wordlenghts, but then a tradeoff with FPGA occupancy should be considered. For example, a lookup table sine generator with 14-bit output and a 16-bit register for the phase accumulator needs a ROM space of approximately $2 \cdot 10^5$ bits to achieve a 'spurious performance' (i.e., a noise-to-signal ratio) of about -90 dB; similar cases are discussed in works like Nicholas and Samueli (1987); Nicholas et al. (1988), or again in Vankka and Halonen (2001). The performance of the solution proposed here is better, and the silicon space used is comparable—let alone the problem of shifting the frequency. Notice also that the disturbed system itself may be nonlinear, thus impairing the significance of $G(j\omega)$ in (3). However, in the single-frequency case, what the fxLMS adaptive scheme needs to know is just how that system responds to a sinusoid at a given frequency with a sinusoid *at the same frequency*: there may be response at *other* frequencies in the case of a nonlinear system, but the algorithm goal is not to cancel such additional response, and so can be attained anyway. Things get more complex in the multi-harmonic case, where inputting a sinusoid at one of the frequencies to cancel may evoke some response at another frequency to cancel, therefore introducing some undesired (and hard to model) couplings. Of course, that problem is more likely to arise if the frequencies to cancel are close—a case where selecting individual components by filtering is particularly difficult, by the way. Similarly, floating-point computation is not a built-in feature of FPGA devices and has to be implemented on purpose, if required, at great silicon area cost. It is therefore important to select which computations should be really performed in emulated floating-point computation (and with what accuracy) and which should not necessarily be implemented in this way. Both of the issues just mentioned will be evidenced in the tests of section 5.

The controller was implemented on a National Instruments CompactRIO system. The CompactRIO is composed of a chassis holding a PC host and up to eight input/output modules of different types (analogue, digital, and so on). The chassis backplane holds a FPGA (1 or 3 Mgates devices are at present available). The typical CompactRIO application is composed of a 'host' part, running on the chassis PC, and of a 'FPGA' part, executed by the backplane device. Both parts can access the input/output modules, and communicate with one another. In addition, both parts can be developed in the LabVIEW graphical programming environment. The LabVIEW development system, completed with the suitable toolkits, takes care of compiling the host and the FPGA part of an application (in the

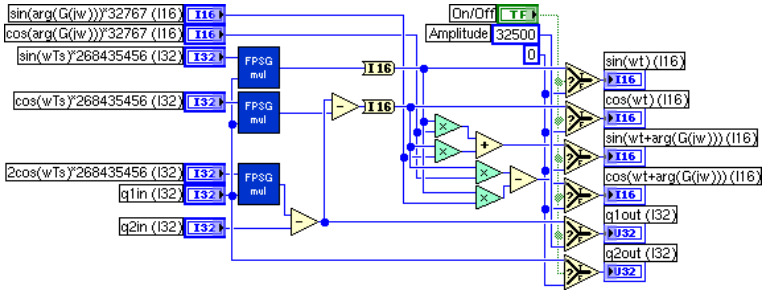

Fig. 1. Block diagram of the sine/cosine generator and phase shifter.

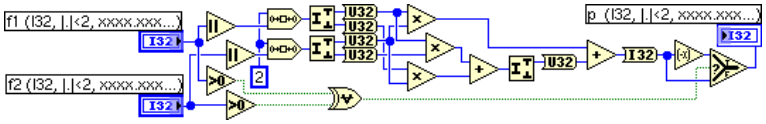

Fig. 2. Block diagram of the floating point multiplier.

latter case through an automatic translation into the VHDL language), and of deploying both parts to the CompactRIO system. However, the organisation of the application, with particular reference to the PC-FPGA communication and to timing, is left under the designer's control, the development system providing of course the necessary elementary primitives. For more information, the interested reader can consult the CompactRIO documentation (`http://www.ni.com/compactrio`).

4.1 Design considerations

Despite the simplicity of the fxLMS algorithm, many problems may arise in its implementation, mostly related to computational precision. The employed FPGA architecture only offers integer computations, as most similar devices, and any other calculus machinery is entirely up to the designer. The latter has then to face a space/accuracy tradeoff, deciding where high accuracy is essential and deserves generous space allocation on the FPGA.

As a result of the design experience gathered with the work presented herein, the following can be stated. First, since accuracy costs a lot, the best idea is to spend accuracy to achieve high-purity sine generators, given that lookup tables are not accurate enough, or to realise closed-loop filters—not necessary herein, see Piroddi et al. (2006a,b, 2007). If this is done, with conveniently designed *ad hoc* floating point arithmetics, it is possible to have all the required signal processing implemented in the FPGA (an important peculiarity of the presented application). Also, in the presence of pure generators, the (low-frequency) fxLMS weight adaptation can be coarse, and even employ integer numbers (thus fitting too on the FPGA, a second important peculiarity). Actually, thanks to the generators purity, the weights typically enter some 'limit cycle-like' behavior that has practically no effect on the controlled variables.

4.2 Code organisation

The presented controller is composed of three parallel, independent noise cancellers, each one relative to an accelerometer/piezo actuator colocated couple (decentralized control). Given the scope of this work, we concentrate on one of the three identical 'channels' (briefly, the 'controller'). The controller is in turn composed of three parallel blocks (each of which will be termed the 'canceller' from now on for brevity), devoted to eliminate the three main disturbance frequencies, namely the rotor rotational frequency (about 700 Hz to give a figure) and the two frequencies associated to the bearings (about 260 and 270 Hz, thus quite close to one another). Each canceller is composed of four main functional blocks (some of which can be shared between different channels): the multi-frequency sine/cosine generator, the phase shifter, the fxLMS weights' updater, and the computation of the control signal.

Multi-frequency sine/cosine generator This block, like the phase shifter of section 4.2.2, is shared among the channels for apparent efficiency and silicon area reasons. It employs the well known (Proakis and Manolakis, 1995) discrete-time dynamic system

$$\begin{aligned} q(k) &= 2\cos(\omega T_s)q(k-1) - q(k-2) \\ x_0(k) &= \sin(\omega T_s)q(k-1) \\ x_1(k) &= q(k) - \cos(\omega T_s)q(k-1) \end{aligned} \tag{4}$$

initialised with the value $q(0)$ of the internal variable q set to the desired wave amplitude, and $q(-1)$ set to zero. An advisable value for $q(0)$ is 'a bit less' of the full bit scale devoted to the sine/cosine waves, to avoid possible overflow problems: in the vacuum pump application that value was 32500, the (signed) sine/cosine waves being represented with 16 bit integers.

For precision reasons, *despite the limited number of bits employed*, floating point computations were emulated on the FPGA by means of a custom number representation, reducing the exponent bits and increasing the mantissa bits with respect to the standard IEEE754 format. The overall LabVIEW block diagram of the sine/cosine generator, including also the phase shifter described in section 4.2.2, is shown in figure 1, while figure 2 shows the block diagram of the FP multiplier - the 'FPSG MUL' block in figure 1 - optimised for sine/cosine generation (i.e., for signals of magnitude less or equal to the unity) with (4).

The signal generator test of section 5 will evidence how pure the obtained signals are, while the overall test in the same section will show that having very pure signals allows for coarser precision computations in the rest of the fxLMS algorithm.

Phase shifter The adaptive notch filter, see (3), conceptually requires to pass the generated sine and cosine signals through a model of the disturbed system, here in the form of a transfer function. To avoid having that transfer function implemented in

the FPGA for silicon space reasons, it is however advisable to simply shift those signals by the transfer function's frequency response at the frequency ω of interest, therefore computing the shifted signals as

$$\begin{aligned} \sin(\omega t + \arg(G(j\omega))) &= \sin(\omega t)\cos(\arg(G(j\omega))) \\ &\quad + \cos(\omega t)\sin(\arg(G(j\omega))) \\ \cos(\omega t + \arg(G(j\omega))) &= \cos(\omega t)\cos(\arg(G(j\omega))) \\ &\quad - \sin(\omega t)\sin(\arg(G(j\omega))) \end{aligned} \tag{5}$$

As will be shown by the tests of section 5, integer computations are here sufficient for a successful performance of the fxLMS algorithm, due to its relative tolerance of phase uncertainty. In addition, the frequency response phase need not be computed on the FPGA, but can be calculated by the cRIO processor and then passed to the FPGA. Figure 3 shows the block diagram of the used multiplier.

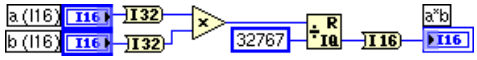

Fig. 3. Block diagram of the integer multiplier.

Notice that the magnitude effect of $G(j\omega)$ is not implemented, the phase shift only being applied. This simply means leaving the amplitude adjustment of the canceling control signal entirely to the adaptation of the filter weights, see (2). This is apparently possible in principle, and proved successful in practice—see again the tests of section 5. As a consequence, a significant amount of silicon space could be saved.

Update of the fxLMS weights and computation of the control signal The block diagram of figure 4 shows the canceller functional blocks devoted to update the filter weights and to compute the control signal.

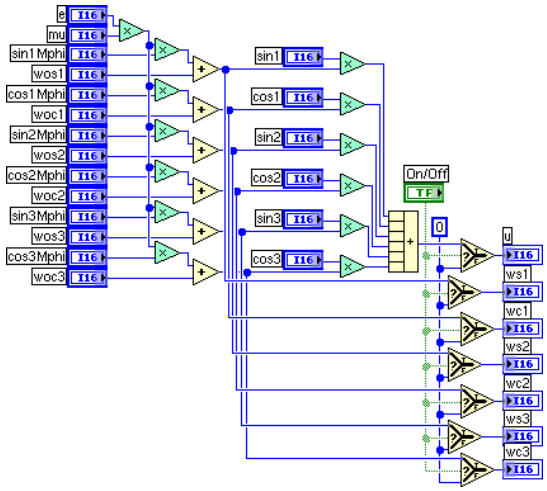

Fig. 4. Block diagram for the filter weights update and the control signal computation.

4.3 Overall application

The block diagram of the overall FPGA part of the application (3 channels, 3 cancelled frequencies per channel) is shown in figure 5. The application has also a host part, running off-line (i.e., not subject to the real-time sampling) on the CompactRIO PC, and devoted to computing the integer parameters for the FPGA part. The FPGA part is shown in figure 5, fits in a 3 Mgates device, and can run safely at the frequencies required for the addressed type of applications (20 kHz for the pump vibration control).

5. CONTROLLER VALIDATION

Referring to the pump application, a single channel with three frequencies was tested in simulation, closing the loop on an accurate frequency domain model of the pump, reported in Piroddi et al. (2007). For computational simplicity, the frequency response function used for weight adaptation was done with a simplified version of the above model. This also demonstrates the model error tolerance of the fxLMS algorithm for this application.

A first test requires the elimination of a single frequency, namely the bearings' higher one (274 Hz). The resulting signals (in converter units, represented as 32 bit signed integers) are shown in figure 6, while figure 7 shows the behavior of the only two filter weights involved. As can be seen, the cancellation is good and the weights' convergence rapid (in all the tests the sampling frequency is 20 kHz, the time axis is graduated in samples and the frequency axis in Hz).

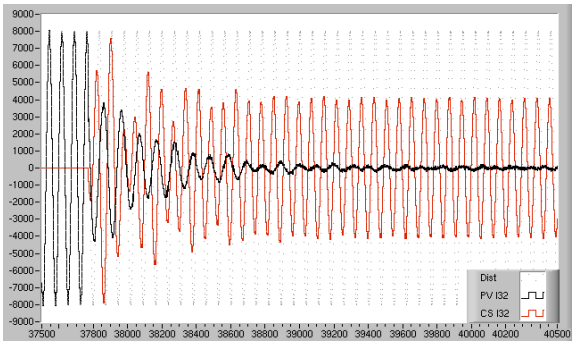

Fig. 6. Cancellation of a single frequency: controlled variable (PV I32, black), control signal (CS I32, red) and disturbance (Dist, gray).

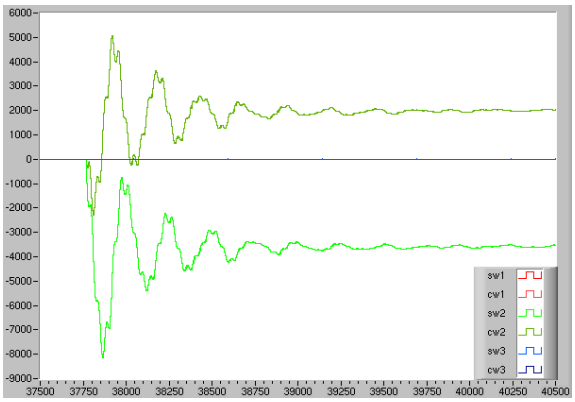

Fig. 7. Cancellation of a single frequency: filter weights.

Figures 8, 9 and 10 show the relevant signals, spectra and weights when two 'far' frequencies need canceling, namely the rotation frequency (697 Hz) and the bearings' lower one (264 Hz). As can be seen, everything is satisfactory like in the previous test. Finally, figures 11, 12 and 13 show what happens when the three frequencies of 264, 274 and 697 are all to be cancelled. Results are still good, but the two near frequencies apparently 'interact' due (also) to the finite precision implementation, causing a low-frequency modulation of the weights.

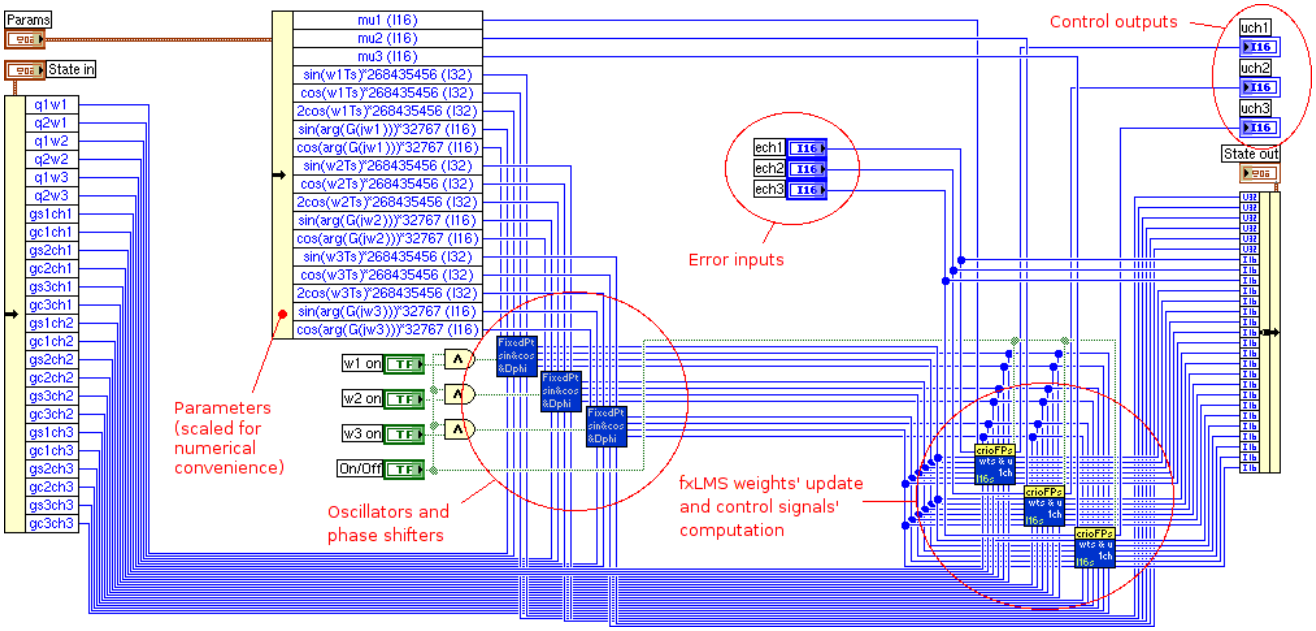

Fig. 5. Block diagram of the overall FPGA part of the application.

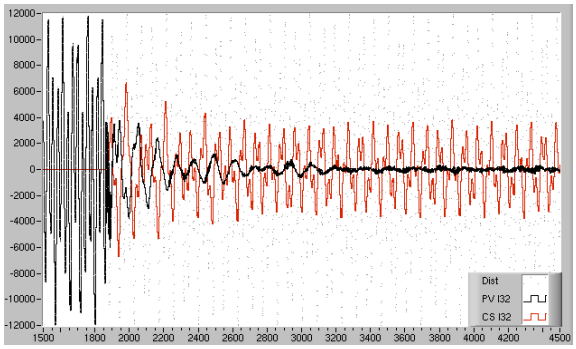

Fig. 8. Cancellation of two 'far' frequencies: controlled variable (PV I32, black), control signal (CS I32, red) and disturbance (Dist, gray).

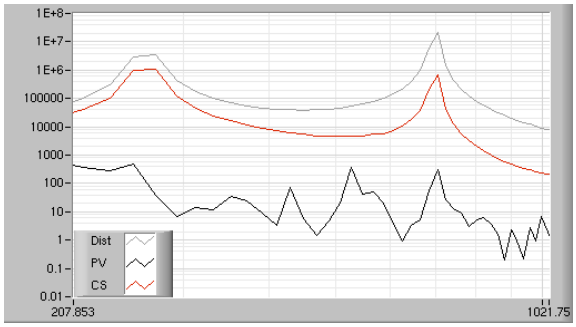

Fig. 9. Cancellation of two 'far' frequencies: spectra of the controlled variable (PV I32, black), control signal (CS I32, red) and disturbance (Dist, gray).

Further tests were carried out to ensure the capability of the implemented controller to fulfill timing requirements.

As a result of the performed validation, and generalising from the particular (pump control) application, we can state the following. First, the adopted design choice is correct: the tests confirm that having a high-purity signal generation allows for coarser computations in the following phases, with particular

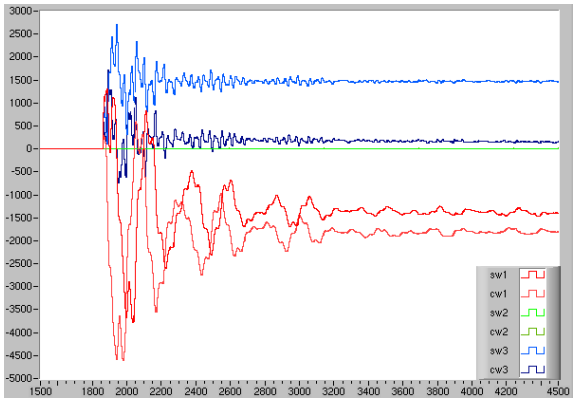

Fig. 10. Cancellation of two 'far' frequencies: filter weights.

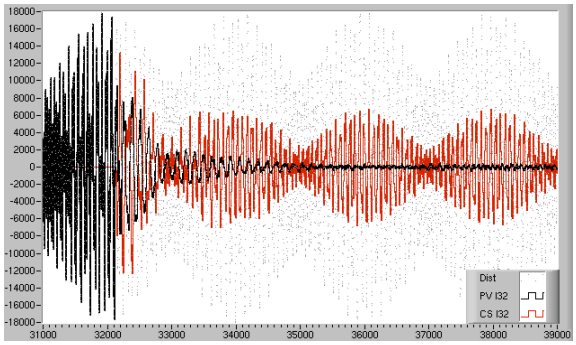

Fig. 11. Cancellation of three frequencies, two of which 'near': controlled variable (PV I32, black), control signal (CS I32, red) and disturbance (Dist, gray).

reference to the filter weights adaptation. Following that design guideline allows to save silicon space, and is practically mandatory if the cancelled frequencies need to be moved. Then, as expected, also with finite precision the effects of the adaptation gains are extremely limited; the fxLMS performance is hardly affected by them, if not beyond very wide limits, and essentially with respect to the cancellation settling time. Finally, and

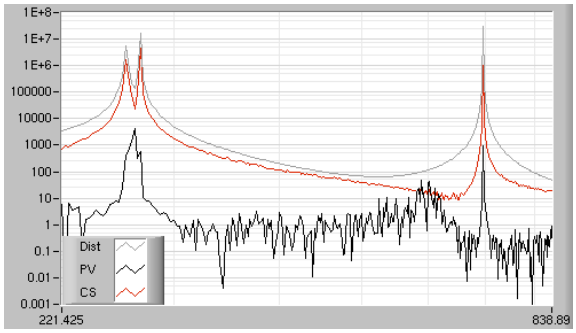

Fig. 12. Cancellation of three frequencies, two of which 'near': spectra of the controlled variable (PV I32, black), control signal (CS I32, red) and disturbance (Dist, gray).

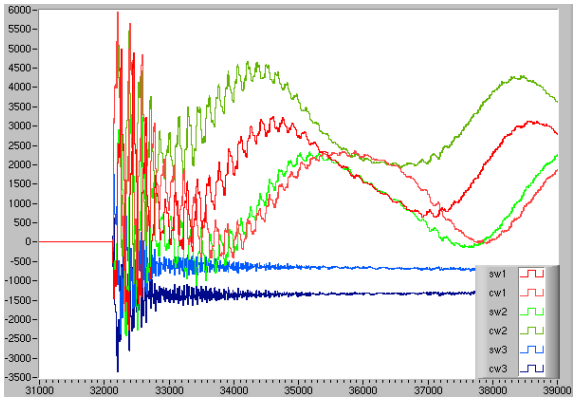

Fig. 13. Cancellation of three frequencies, two of which 'near': filter weights.

also in this case as expected, computational issues appear to provoke some interplay between 'near' frequencies, although satisfactory performance can be obtained anyway. The weights' oscillations of figure 13 disappear if the update is done with high precision floating point arithmetics, *even if the weights are then quantised to be integers*. This leads to search the reason for the observed effects in the inherently nonlinear character of finite-precision arithmetics—a character that is enhanced by reducing the precision. More details on the effects of finite precision arithmetic in this particular context, that it would be lengthy and inessential to report here, can be found in the companion paper Maggio et al. (2007).

Experimental validation of the FPGA-based controller has also been carried out on the turbomolecular vacuum pump application with encouraging results, see Piroddi et al. (2007).

6. CONCLUSIONS AND FUTURE WORK

The implementation of AVC systems on very high-speed computational architectures, taking the FPGA case as example, was discussed. Starting from the particular case of vibration control in a turbomolecular vacuum pump, some general considerations for the design of similar applications were introduced. Peculiar to applications designed along the proposed path is that all of the required signal processing can fit on the FPGA.

The implemented controller is a general purpose multiharmonic disturbance rejection system, based on the adaptive notch filter concept, and the fxLMS algorithm; the controller can be employed for various narrowband noise or vibration rejection problems. Some simulations were provided to show the validity of the proposed design choices for the FPGA coding. Future work will include further code optimisations, inclusion of the capability of varying the cancelled frequencies over time (which should not pose particular problems with the proposed implementation scheme) and testing on further physical control systems.

REFERENCES

M. Antila. Contemporary electronics solutions for active noise control. In *2004 International Symposium on Active Control of Noise and Vibration, ACTIVE 04*, Williamsburg (Virginia), USA, 2004.

F. Casaro. Vibrations in turbomolecular pumps: spectra and source location. *J. Vac. Sci. Technol.*, A7(3):2377–2380, 1989.

A. Di Stefano, A. Scaglione, and C. Giaconia. Efficient FPGA implementation of an adaptive noise canceller. In *7th International Workshop on Computer Architecture for Machine Perception, CAMP05*, pages 87–89, Palermo, Italy, 2005.

C. R. Fuller, S. J. Elliott, and P. A. Nelson. *Active control of vibration*. Academic Press, London, United Kingdom, 1996.

R. Hashemian. Design of an active noise control system using combinations of DSP and FPGAs. In *PLD Conference Proceedings*, 1995.

S. M. Kuo and D. R. Morgan. *Active Noise Control Systems: Algorithms and DSP Implementations*. John Wiley & Sons, New York (NY), USA, 1996.

M. Maggio, A. Leva, and L. Piroddi. Finite-precision implementation issues in narrowband active control. In *Proc. CDC07 (to appear)*, New Orleans, LA (USA), 2007.

D.R. Morgan and J. Thi. A multitone pseudocascade filtered-x LMS adaptive notch filter. *IEEE Trans. on Signal Processing*, 41(2):946–956, 1993.

H. T. Nicholas and H. Samueli. An analysis of the output spectrum of direct digital frequency synthesizers in the presence of phase accumulator truncation. In *Proc. 41st Annual Frequency Control Symposium*, pages 495–502, Philadelphia, PA (USA), 1987.

H. T. Nicholas, H. Samueli, and B. Kim. The optimization of direct digital frequency synthesizer performance in the presence of finite word length effects. In *Proc. 42nd Annual Frequency Control Symposium*, pages 357–363, Baltimore, MD (USA), 1988.

L. Piroddi, A. Leva, and F. Casaro. Modeling and active vibration control of a turbomolecular vacuum pump. In *Proc. IEEE International Conference on Control Applications*, Munich, Germany, 2006a.

L. Piroddi, A. Leva, and F. Casaro. Vibration control of a turbomolecular vacuum pump using piezoelectric actuators. In 45^{th} *IEEE Conference on Decision and Control*, San Diego (CA), USA, 2006b.

L. Piroddi, A. Leva, and F. Casaro. Feedback and feedforward active vibration control schemes for a turbomolecular vacuum pump. In *Proc. ECC 2007*, Kos, Greece, 2007.

J. G. Proakis and D. K. Manolakis. *Digital Signal Processing: Principles, Algorithms and Applications*. Prentice-Hall, Englewood Cliffs (NJ), USA, 1995.

E. Rava, E. Rava, and M. Marzio. Recent developments in turbomolecular pump rotor balancing techniques. *J. Vac. Sci. Technol.*, A5(4):2530–2533, 1987.

J. Vankka and K. Halonen. *Direct digital synthesizers: theory, design and applications*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2001.