FPGA Based Rapid Prototyping of a Crosstalk-resistant Adaptive Decorrelator

Tom. J. Moir
School of Engineering
AUT University, St Pauls Street
Auckland, NewZealand
tmoir@aut.ac.nz

ABSTRACT: *This paper describes a method of rapid prototyping of a Field-Programmable Gate Array (FPGA) based adaptive decorrelator. The decorrelator has the ability to separate two acoustically mixed sound sources in real-time. The separation method used is known as a Crosstalk Resistant Adaptive Noise Canceller (CRANC) or often a Symmetric Adaptive Decorrelator (SAD). The implementation is performed on a National Instruments Compact Rio and programed in a subset language of LabView (or G code). The algorithm runs two concurrent least-mean squares (LMS) which are cross-coupled to each other. The nature of the implementation means that several CRANC filters can be cascaded and pipelined to speed up the throughput.*

Keywords:FPGA, Adaptivve filtering, Decorrelation, LabView, CRANC, SAD, LMS

Received: 22 May 2013, Revised 29 June 2013, Accepted 5 July 2013

1. Introduction

The problem of separating random unknown noise and a desired random signal (e.g a speech signal) has been one of interest since the pioneering work of Wiener [1] and Kalman [2]. Such approaches require accurate models of the time-varying characteristics of signal and noise in order to be successful. In a real environment rarely do we have the opportunity to have such information at hand, and a noise or signal source can arrive in various forms from any direction. So with no such apriori information available, it was not until the early work of Widrow and co-workers [3] that algorithms that "self-learned" characteristics obtainable from more than one sensor became available. The approach made use of an earlier invention based on the mathematical method of steepest descent and is known as least-mean squares (LMS) [4]. The Widrow method had applications in environments where the geometry was fixed (ie the layout of the room and position of the signal and noise sources) and used a second (or more) reference sensor to pick up the noise on its own. (often a distance away from the signal) This noise, although correlated with the additive noise to be removed, needs to be aligned in frequency and amplitude by means of the LMS algorithm and subtracted from the signal plus noise.

The above approach can be made to work under certain restricted environments, but it soon became apparent that having the two (or more) sensors far apart was not as practical as having them close together. Many approaches have been used to try and solve the problem where the sensors are close together. These include switching techniques which require a voice-activity detector (VAD) to distinguish signal from noise[5]. The problem with this is that the cancellation is only as good as the speed

and accuracy of the VAD which often fails at low signal to noise ratios (SNRs) or is unable (or fast enough) to distinguish between two competing talkers. Ordinary LMS noise cancelling and its variants have already been implemented elsewhere on FPGA devices[6-12].

A more modern approach is known as blind-source separation (BSS)[13] where a VAD is not required and the sensors can be close together. The BSS approach is far more computationally difficult to implement on an FPGA, though some attempts have been made already eg [14] for a constant mixing matrix and [15] for convolutive mixtures.

A compromise between the more complex BSS method and the more traditional LMS approaches is found in adaptive decorrelation methods[16-21]. These methods do not attempt to do any filtering as such, but instead remove the interlinking cross-coupling transfer functions between the sources. In essence, these methods are a form of diagonalization of the cross-correlation matrix. The simplest form of these decorrelators is often known as a crosstalk-resistant adaptive noise canceller (CRANC) or even a symmetric adaptive decorrelator (SAD)[22]. The method uses two cross-coupled LMS algorithms as shown in the next section. It is this technique which is implemented herein using an FPGA programmed in LabView "*G*" code. This is not the first time that LabView has been used to implement an adaptive filter on a CompactRio (eg [23]), but this is the first such implementation of the two-input decorrelator. A much earlier implementation was performed on a Masscomp mini-computer obtaining up to 11dB SNR improvement for a sampling frequency of 8kHz with 24 taps/LMS[19]. The advantages of using FPGAs are well known. These include the inherent parallelism leading to faster run-times and lower power consumption than DSP processors.

.

2. Theory

We have two random signals t_k^1, t_k^2which are mixed via the natural acoustics occurring in a room. Figure 1 illustrates this. The object of the exercise is to separate these two signals using so-called blind-source separation. In other words no apriori information is known other than the fact that there are only two random signals. If the two received signals (after having passed through some acoustic mixing transfer functions) are known as S_k^1, S_k^2then we have a 2×2 coupling matrix of FIR transfer functions H_{11}(provided it is non-singular),and

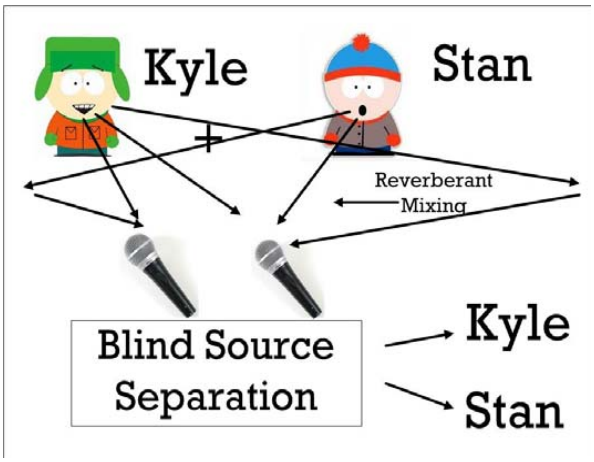

Figure 1. Showing acoustic mixing and the FPGA separator. Microphones are 6cm apart

$$\hat{s}_k = H_{11}(q^{-1})\, t_k = \begin{bmatrix} h_{11}(q^{-1}) & h_{12}(q^{-1}) \\ h_{21}(q^{-1}) & h_{22}(q^{-1}) \end{bmatrix} \begin{bmatrix} t_k^1 \\ t_k^2 \end{bmatrix} \qquad (1)$$

If H_{11} is singular then the two sources cannot be separated. This would be the case when two signals originated in the same geometric position.

In equation (1), (q^{-1}) is defined as the backwards shift operator such that for a discrete signal y_k we have $q^{-1} y_k = y_{k-1}$.
A block-diagram of the mixing process is shown in Figure 2.

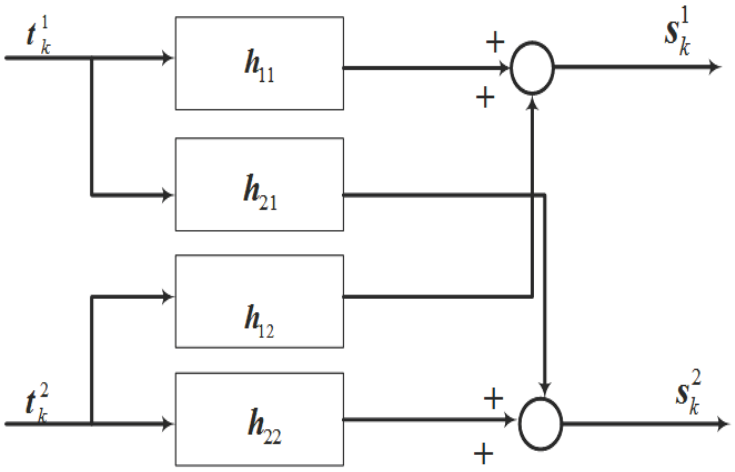

Figure 2. Block diagram of the acoustic mixing process

It is the cross-coupling terms which are to be removed. However, they are unknown along with the forward path terms. To proceed, we must further assume that h_{22} and h_{11} are both minimum-phase and write in matrix format

$$\begin{bmatrix} s_k^1 \\ s_k^2 \end{bmatrix} = \begin{bmatrix} 1 & g_{21}(q^{-1}) \\ g_{12}(q^{-1}) & 1 \end{bmatrix} \begin{bmatrix} r_k^1 \\ r_k^2 \end{bmatrix} \qquad (2)$$

where $r_k^1 = h_{11}\, t_k^1$ and $r_k^2 = h_{22}\, t_k^2$, $g_{12} = h_{12} / h_{22}$, $g_{21} = h_{21} / h_{11}$ and we estimate the r_k terms instead of the t_k terms. The r_k signals will differ from the t_k signals only by FIR transfer functions h_{11} and h_{22}. Hence we are only interested in separation and not necessarily obtaining ideal estimates of the signals themselves (which if required need further single-channel deconvolution). We find that in practice, convolution by these (unknown) FIR acoustic transfer functions make no difference to the quality of the decorrelated signal. This is illustrated below in Figure 3.

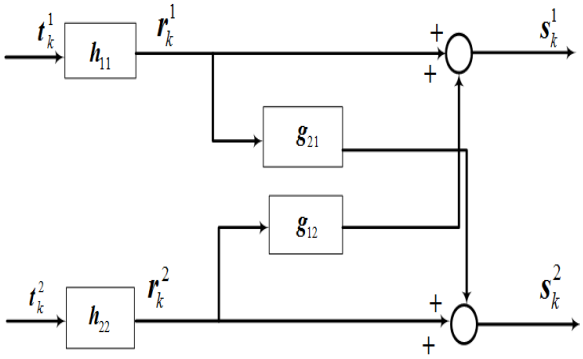

Figure 3: Simplified mixing process where there are only two unknown cross-coupling terms and the forward paths are unity

Separation is then found via the weight-vector updates of the two LMS algorithms. For n weights/LMS we have

$$w_{k+1}^1 = w_k^1 + \mu_1 e_k^1 (X_k^1)^T$$
$$w_{k+1}^2 = w_k^2 + \mu_2 e_k^2 (X_k^2)^T \qquad (3\text{ a, b})$$

Where $e_k^i = s_k^i - X_k^T w_k^i$, $i = 1, 2$, $X_k^1 = [e_{k-1}^2, e_{k-2}^2 \ldots, e_{k-n}^2]^T$ and $X_k^2 = [e_{k-1}^1, e_{k-2}^1 \ldots, e_{k-n}^1]^T$. The two de-correlated random signals are $\hat{r}_k = e_k^i$, $i = 1, 2$. We make the assumption for strict causality that the two polynomials $g_{12}(0) = g_{21}(0) = 0$. The μ_1, μ_2 coefficients are the step sizes of the individual LMS algorithms. Since the algorithm is symmetric, we usually make $\mu = \mu_1 = \mu_2$ The two-input decorrelator is illustrated in Figure 4 and its adaptive counterpart in Figure 5.

1 T

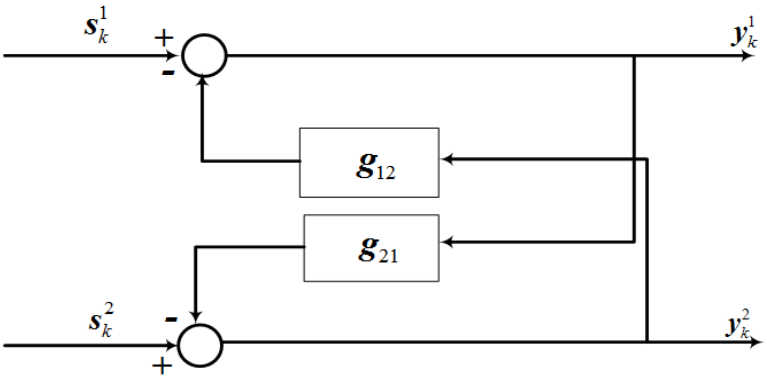

Figure 4: Backwards eparation method[17]

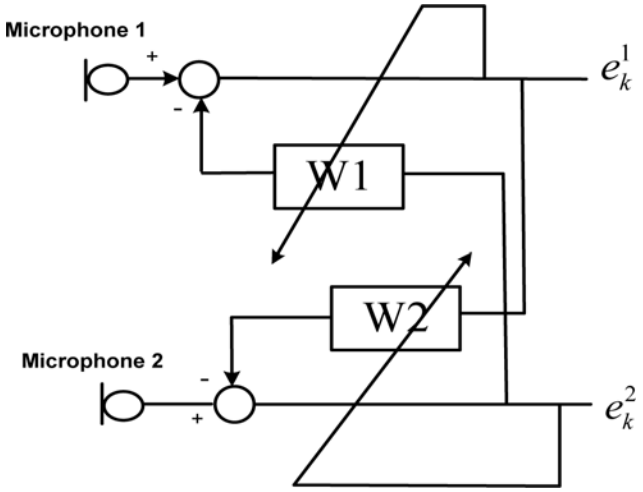

Figure 5: Two input decorrelator showing cross-coupled LMS.

It is known from several theoretical studies[18, 24] that the above algorithm gives rise to biased estimates and as such does not give rise to "*optimal*" estimates in the sense of least-squares. However, the algorithm is a good compromise when limited resources are available for implementation in a real-time environment and still give excellent performance.

3. Architecture of the Real-time Target

An FPGA rapid-prototyping architecture was chosen as the target for implementing the real-time adaptive decorrelator -the National Instruments CompactRio (Trademark of National Instruments). The two LMS algorithms can easily be run in parallel and an FPGA is ideal for this. Ordinary sequential programming is limited by the so-called Van Neumann bottleneck of fetch and decode for each instruction. Even though threaded programming is possible on a conventional processor, the effect is purely an illusion, as the processor merely time-division multiplexes between different threads and is not truly concurrent. Of course multiple processors can be used as an alternative approach to this problem provided one can communicate fast enough between them.

An FPGA array on the other hand has no processor at all and runs code as a digital circuit often at much lower clock speeds. However, since it is a digital circuit, any code must run concurrently unlike threads in a conventional processor. This enables applications for low power consumption with fast speed.

The Rio runs as a stand-alone unit and is programmed in the high-level data-flow language LabView. The Rio used in the first set of experiments is based on the Virtex 5 integrated circuit from Xilinx and has an eight-slot reconfigurable embedded chassis that

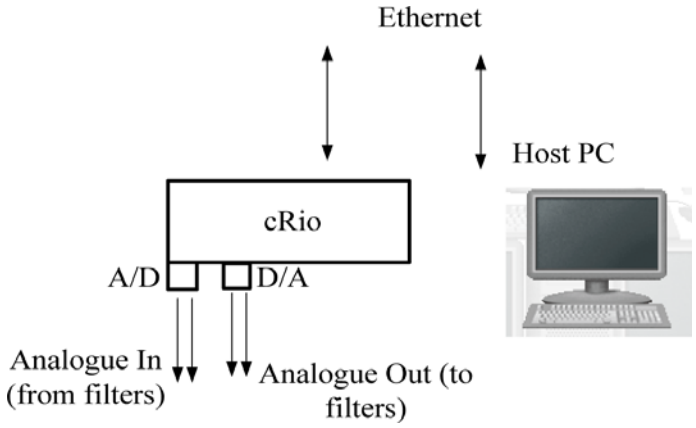

Figure 6: Compact Rio structure.

accepts any CompactRIO I/O module. In fact little knowledge of hardware descriptive languages such as VHDL are required at all and we can implement the complete algorithm using this rapid prototyping approach. The Rio is linked to a host PC via the Ethernet for programming purposes and initial testing (Figure 6) . A 4-input sigma-delta A/D convertor was used and a 4-ouput D-A convertor. We only use two of these inputs and outputs and the remainder are left for future algorithm expansion when using more sensors. The sampling rate is easily changed on the host computer. The Rio has its own processor onboard, but for this application only the FPGA was used in the end module. Early versions of the program used the PC as an oscilloscope to view the estimated signals. This is done by a FIFO and using the onboard co-processor on the Rio. This overhead is not required on the final working module however.

For *n* weights, the algorithm in equations (3a,b) requires approximately 4n multiplications and 4n additions, twice as many as ordinary LMS. Two cross-coupled LMS algorithm has many advantages however. One in particular which has not been exploited until now, is the ability to split the overall filter of order n into two cascaded filters of order *n*/2. This is because there are two outputs (errors) unlike the usual one output for conventional LMS and these feed naturally into the two inputs of the next stage. Normally there would be no advantage of doing so, but on an FPGA we can use pipelining to execute the two stages in parallel offering further computational efficiency at the expensive of a time-delay of *n*/2 samples whilst the first stage "*fills up*".

Pipelining is the use of feedback nodes or shift-registers in order to allow items that would normally execute serially to execute in parallel. Within a negative feedback loop it is not possible to add pipelining, since this adds a time-delay which in turn de-stabilizes the loop. This is why the decorrelator is split into two separate decorrelators as shown in Figure 7. This of course can be extended to more stages, but this uses more space on the FPGA. Likewise several parts of the individual LMS algorithms themselves were sped up considerably by exploiting parallelisms. A computation $X_k^T w_k$ such as used in ordinary LMS (a dot product of two vectors) can be split into two parallel loops by the following.Let $X_k = [\tilde{x}_k^1, \tilde{x}_k^2]$ and $w_k = [\tilde{w}_k^1, \tilde{w}_k^2]$.Then the dot product becomes $X_k^T w_k = (\tilde{x}_k^1)^T \tilde{w}_k^1 + (\tilde{x}_k^2)^T \tilde{w}_k^2$ which will run twice as fast due to the parallel computation. This does not affect the stability of the loops (unlike pipelining), because the two computations which run in parallel have to wait for each other to finish before they can add the result for the dot product.

The update equation of the weight vectors (3a,b) were also split into two parallel updates to give further improvement. Splitting up dot products in this manner however uses more space (gates) on the FPGA as the cost of the improvement in speed. This is because the LabView code will not "*unwrap*" a FOR loop, and creating two of them must double the silicon space used. There are two methods used to store weight vectors on the FPGA. The first method is by using arrays, which is very expensive on slice registers.

As far as possible the length of any arrays are usually quite limited in size. It was possible to obtain arrays of length 50 terms (X4 for two LMS algorithms). In order to get more weights, a second ANC (ANC2 in Figure 7) of equal size was used in cascade using pipelining and block Ram. It was not possible to use the block ram approach on both ANC's for a good sampling rate so a mixture of the array method and block Ram method were used. In this way, a sampling frequency of 33.3kHz was obtained. The ADC was 24 bits and the whole algorithm used 24 bits with 4 bits integer and 20 bits fraction. A simulation was carried out using the fixed-

point algorithm in ordinary LabView first in order to verify that these values were sufficient for the dynamic range of the ADC. This also gave a value for the step size of $\mu = \mu_1 = \mu_2$ around 0.008. This in turn is implemented for speed using integer arithmetic using a power of 2, in this case $\mu = 2^{-7} =$ 20.0078125.

A small part of the array-based cross-coupled LMS graphical code is shown in Figure 8 below.

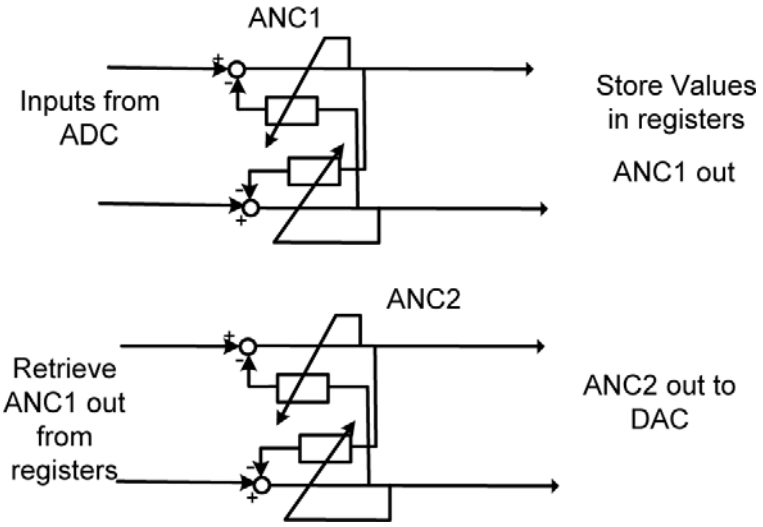

Figure 7. Pipelining two stages of decorrelation (adaptive noise cancelling ANC)

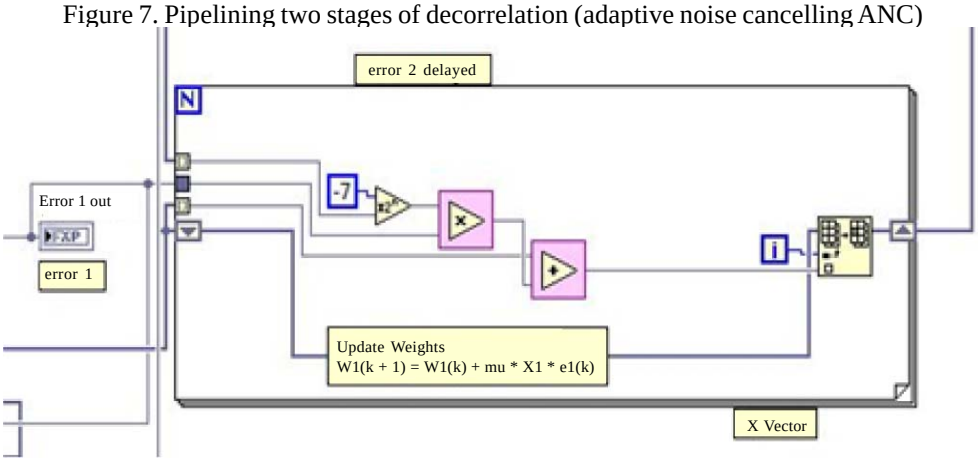

Figure 8. Implementation of equation (3a) using arrays.

This is a FOR loop which is auto-indexed (that is the loop runs for the length of the input array) and runs as many times as the number of weights. At the two ends of the graphical FOR loop are registers which store the updated weights sample by sample. This is standard "*G*" based programming except it is in Fixed-Point arithmetic. The same bit of code but run using the memory method instead of arrays is shown in Figure 9 below.

Also in this code is a timed while-loop. This is included as an extra finesse. Using a timed-loop executes the code within it in one clock cycle (40MHz base clock) instead of one clock cycle per operation. It is not always possible to do this and the compiler will indicate an error where there are timing problems. In this case however it satisfies the hardware requirements.

Figure 10 shows how $X_k w_k$ is split into two summations which exploit the parallelism of an FPGA.

$$X_k^T w_k = (\tilde{x}_k^1)^T \tilde{w}_k^1 + (\tilde{x}_k^2)^T \tilde{w}_k^2$$

Finally, Figure 11 shows how simple it is to implement pipelining by introducing registers (here shown as a delay for both inputs)

between the two cross-coupled LMS algorithms.

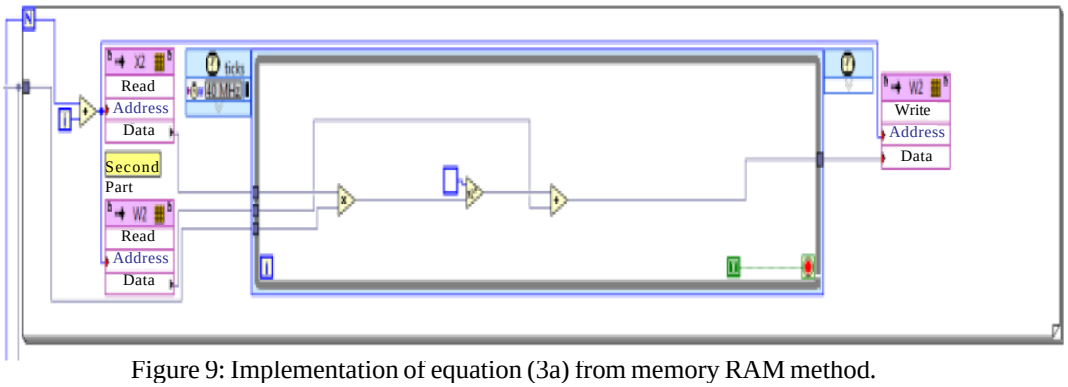

Figure 9: Implementation of equation (3a) from memory RAM method.

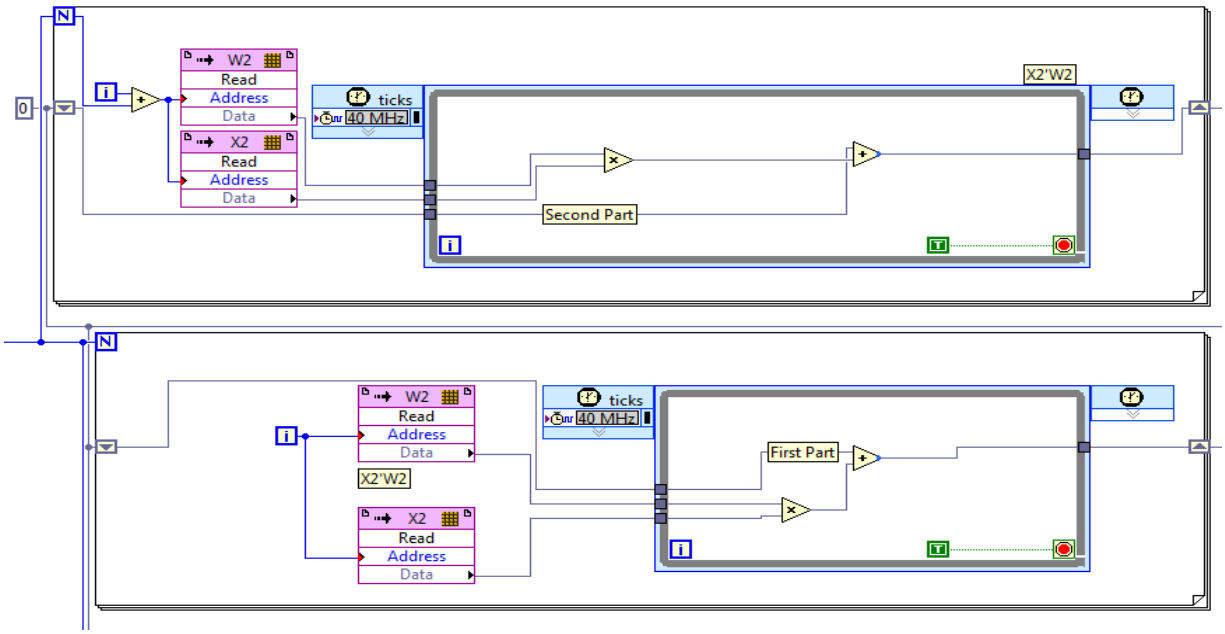

Figure 10: Dot product of two vectors split into two parallel pieces of code. (RAM method).

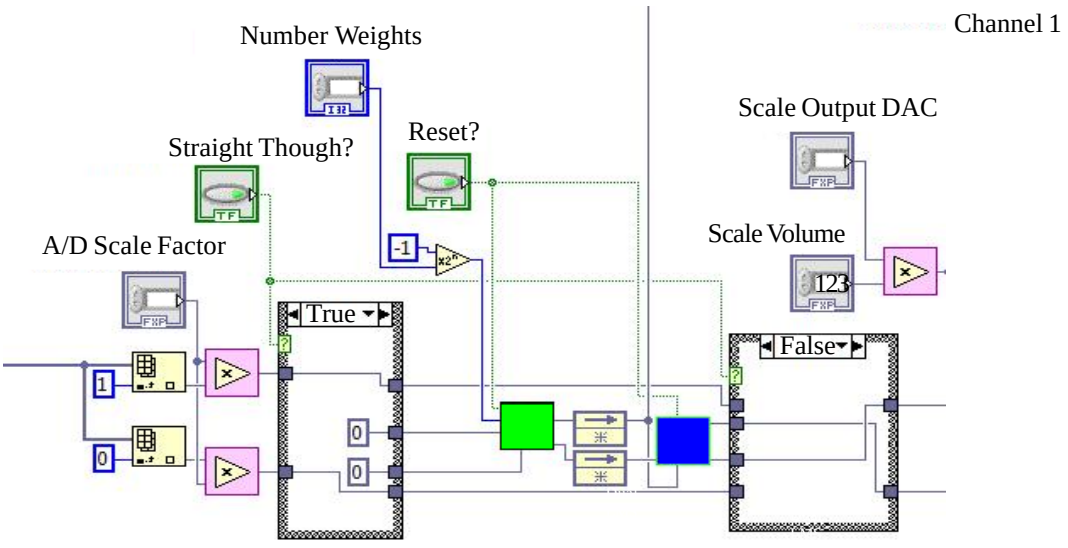

Figure 11: Pipelining of the two CRANC algorithms.

Optional FPGA software automatic gain controls (AGCs) were put on both A/D inputs as a further precaution against saturation of the input signals. These are discussed in the Appendix .

4. Performance of the Real-time Adaptive Decorrelator.

The decorrelator (in fact two decorrelators in pipelined cascade), used a total of 92 weights at a sampling frequency of 33.33kHz. This was not an equal split, but a split of 50 weights on the array based decorrelator and 42 on the block Ram based correlator. For a sampling freq of 33.33kHz, and the speed of sound taken to be 340m/s, we can easily calculate that an acoustic transfer function with maximum delay of 92Ts secs can be accounted for, where Ts is the sampling interval of 30¼s. The distance travelled in this time for 92 weights is 0.94m. We therefore assume that path lengths (the length of signal source to microphone) can be no longer than this distance. This is quite adequate for some applications but unsuitable for rooms with large reverberation times.

In order to compare with similar algorithms, we used a pre-recorded stereo file obtained from Lee at the Salk institute. This was recorded in a room of a person speaking numbers from one to ten with a radio acting as interference. In his work, he used an infomax approach in a feedforward neural network implemented in the frequency domain using a polynomial filter matrix algebra technique.

This file was repeatedly played and re-recorded by a digital audio recorder and used as the source. It was then played into the compact Rio and the output was recorded digitally with the decorrelator periodically switched on and off for comparison. A block diagram is shown in Figure 12.

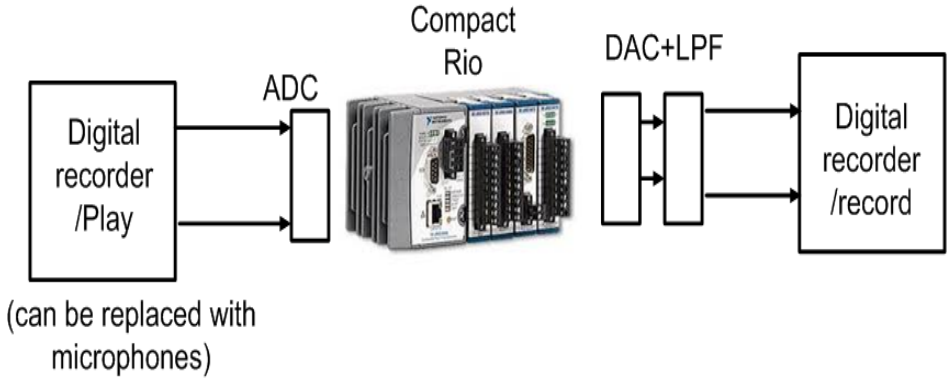

Figure 12: Block diagram of experimental setup.

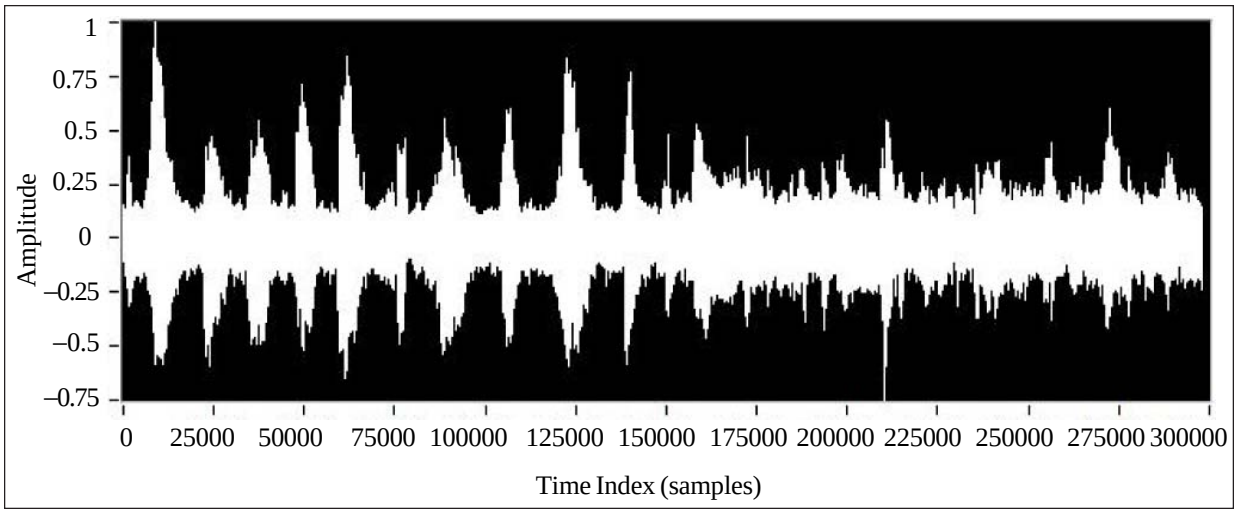

Figure 13: Channel 1 recording of FPGA error output 1. The decorrelator is switched off half way through and shows the re-emergence of the background music.

This method was chosen rather than a "*live*" method with two microphones, because of repeatability of the experiment. The same section of mixed speech (or speech plus noise) can then be tried on any algorithm for comparison purposes. The SNR was measured before and after the FPGA decorrelator was used and an improvement of around 11dB was measured. (see Figure 13) The results obtained were clear of any audio distortion but not as good as the Infomax solution. However, the Infomax solution was not obtained in real-time and is far more complex to implement than this approach.

We see the range of the decorrelator for different bandwidth (ie sampling frequencies) in Table 1.So at the expensive of less available bandwidth and poorer quality of speech, more weights are available for use and hence more distance from the microphone at which the signals can be separated.

Bandwidth(kHz)	Total No of weights	Distance from Microphones(m)
25	70	0.48
16.65	92	0.94
6.25	111	3
5	130	4.4
4.16	150	6.1

Table 1. Maximum number of weights for a given bandwidth.

5. Further Improvements

With more modern improvements to FPGA logic gates (in this case larger arrays), it will always be able to improve such algorithms. At the time of writing, a Rio unit based on the Spartan 6 LX 150 which has 147K logic cells was used for a second experiment. Further simplification was used by reducing the word length used. 16 bit words were used as is more common, with 13 bit fraction and 3 bit integer. The dynamic range of the analogue data was then 3.999 to -4 volts. (Often written as Q3.13 or <16,3> format) It was found that despite the amount of resources used by arrays, that the trade-off with using memory instead did not offer any improvements of significance, since the ROM memory fetch cycle and write slowed down the overall speed and hence the sampling rate. With the new architecture it was possible to implement a decorrelator with 300 weights in total pipelined across three cascaded decorrelators each of length 100. (shown in Figure 14 below). The ability to handle larger-order decorrelators is important in that for rooms with large reverberation times (normally measured and defined as the T60 time for sound to diminish to 60dB of its original value), the filter orders must also be larger. This is aside from the sampling rate which determines the bandwidth (Nyquist rate) and the available maximum distance from the microphones to the sound sources.

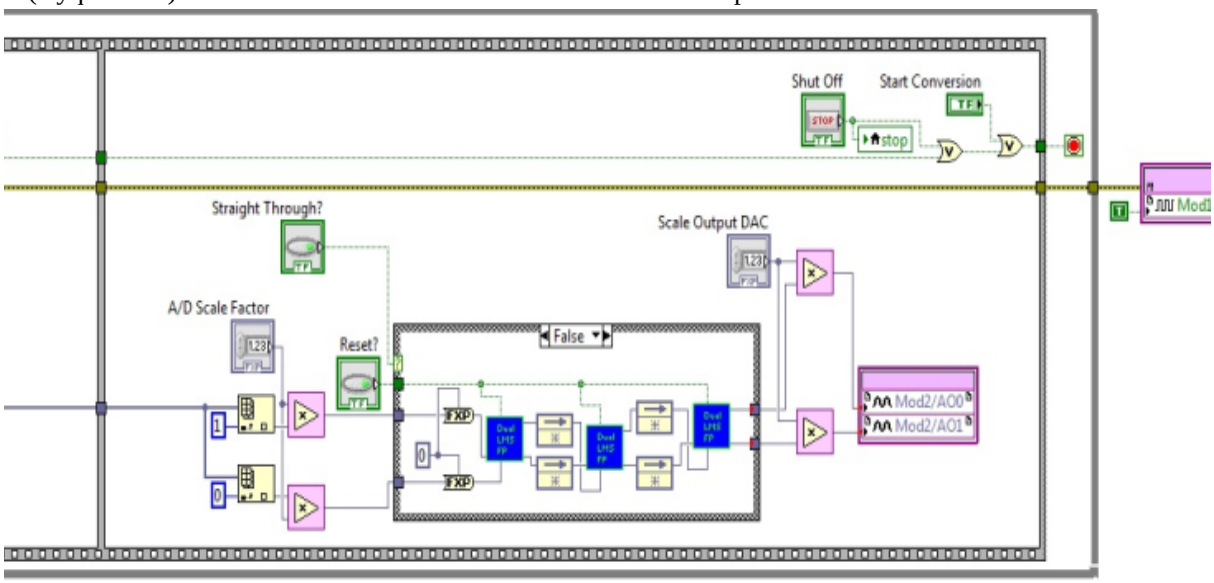

Figure 14: Showing three pipelined CRANC algorithms

This architecture was able to have a sampling time of 33.3kHz. The Speech-Speech mixture wav files of Girolami were processed using this method [25]. The files were played through the digital recorder directly into the ADC of the FPFA decorrelator. A good level of separation was achieved for each channel. Cross talk from the respective other channel was detectable by the human ear for both channels, but the level was quite insignificant and barely audible. The speech signals are shown in Figure 15 below.

6.Conclusions

The implementation of a real-time decorrelator for separating two random (but correlated) signals has been shown. The FPGA approach offers many advantages over straight DSP implementations in that parallelism is inherent in the design together with lower power consumption. The decorrelator has been successfully applied to the separation of real-time speech signals and should prove useful in many areas of speech processing. The work also highlights the fact that in such specialized areas of signal processing it is not always possible to have small filter sizes and the hardware is only just coming to a stage where it is becoming of practical use and could facilitate the development of many new products.

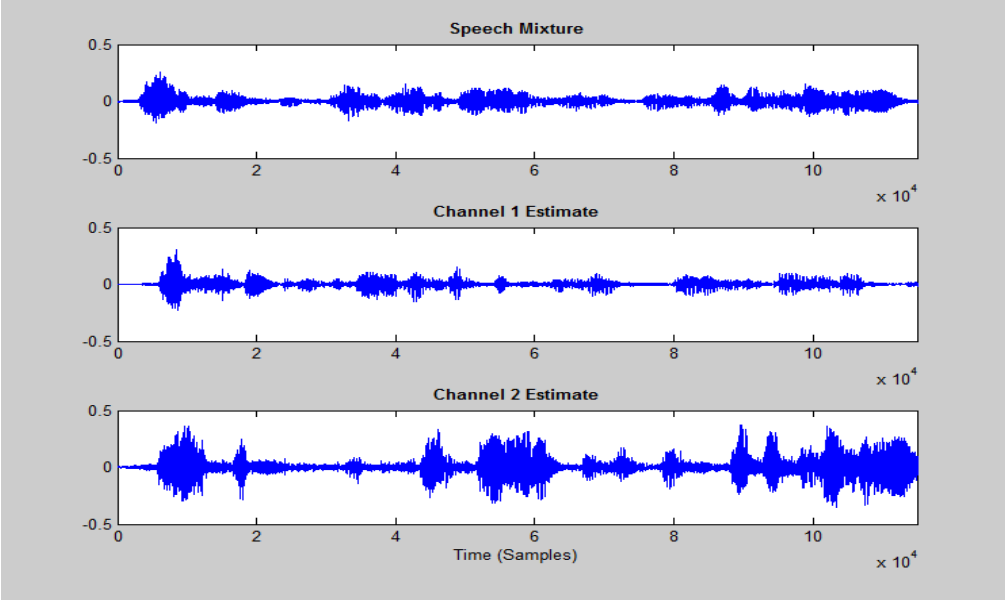

Figure 15: Shows original acoustic speech mixture and the two separated channels.

Appendix Automatic Gain Control

The automatic gain control used (AGC) is best explained from Figure 16.

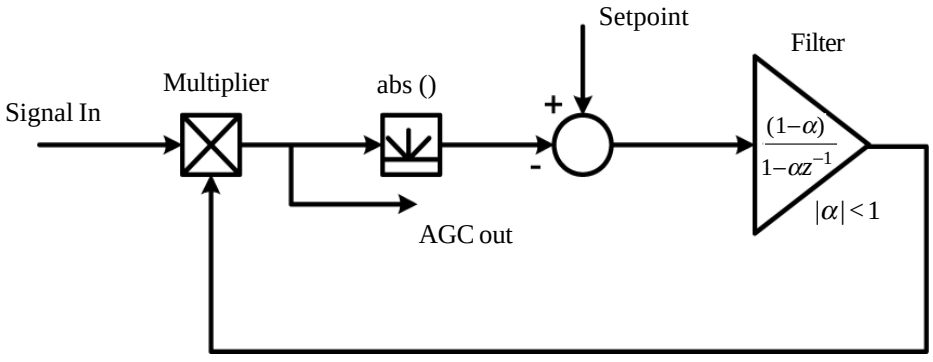

Figure 16. Automatic Gain-Control (AGC)

The AGC consists of a voltage-controlled amplifier (here a multiplier is used), a detector for amplitude (here an absolute value is used but a squarer could be used instead at the expense of greater computational load), a setpoint and a filter. Typically the

setpoint is set to unity and defines the size of the envelope of the AGC ouput. There is no advantage here in using a square-law instead of the absolute value as both give similar results. The filter output will be slowly varying and in the case of a constant envelope at the signal input, will be dc,with a small amount of ac feed-through added to this depending on the bandwidth of the filter. This dc will, by the principle of negative feedback either get bigger or smaller in inverse proportion to the envelope of the input signal. As the signal gets bigger the filter output will get smaller and hence reduce the AGC output back to stable level.

A pure integrator could be used instead of the lowpass filter, but this results in an overflow condition (the integrator tries to ramp to infinity) when the signal input is theoretically zero or not connected. So instead a "*leaky*" integrator (or lowpass filter) is used. This gives a finite steady-state error to step changed in signal envelope, but this is not critical for this application. The bandwidth of the loop is only a few Hz, since making the bandwidth too high, results in amplification of additive noise and even distortion and flattening of the signal envelope. Typically the α term must be less than unity (for stability) and a value of $\alpha = 0.5 = 2^{-1}$ was used for the real-time application which is conveniently a negative power of 2, as is $1 - \alpha$.

Acknowledgement

I would like to thank Sayuj Nath and Evan Fu of National Instruments Auckland for helpful assistance on this work.

References

[1] Wiener, N. (1949). *Extrapolation, Interpolation and Smoothing of Stationary Time Series*: New York,Wiley.

[2] Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems, *Trans ASME, Journal of Basic Engineering*, 82 Series D, p. 35-45.

[3] Widrow, B., Glover JR, J. R., Mc Cool, J. M., Kaunitz, J., Williams, C. S., Hearn, R. H., Zeidler, J. R., Dong JR, E., Goodlin, R. C. (1975). Adaptive noise cancelling:Principles and applications, *IEEE Proceedings*, 63, p. 1692-1716, 1975.

[4] Widrow, B., Hoff, M. E. (1960). Adaptive switching circuits, *IRE Wescon Convention Record*, p. 96-104.

[5] Van Compernolle, D., Leuven, K. U. (1990). Switching adaptive filters for enhancing noisy and reverberant speech from microphone array recordings, in *Proceedings of the IEEE International conference on acoustics,speech and signal processing*, Albuquerque, p. 833-836.

[6] De-Sheng, C., Po-Yu, C., Yi-Wen, W. (2011). Hardware/software Co-design of NLMS adaptive filters on FPGA, *in Consumer Electronics (ISCE)*, 2011 *IEEE* 15*th International Symposium on*, p. 442-445.

[7] Di Stefano, A., Scaglione, A., Giaconia, C. (2005). Efficient FPGA implementation of an adaptive noise canceller, in *Computer Architecture for Machine Perception, CAMP* 2005, *In: Proceedings. Seventh International Workshop*, p. 87-89.

[8] Fohl, W., Matthies, J. (2009). A FPGA-based adaptive noise cancelling system, in 12*th Int Conference on Digital Audio Effects (DAFx-09)* Como,Italy, 2009.

[9] Frech, A., Limmer, S., Russer, P. (2011). Noise cancelling algorithms for FPGA-based time-domain EMI measurements in real-time, in *Electromagnetic Compatibility (EMC)*, 2011 *IEEE International Symposium on*, p. 484-488.

[10] Ramos, R., Manuel, A., Olivar, G., Trullols, E., Del Rio, J. (2001). Application by means of FPGA of an adaptive canceller of 50 Hz interference in electrocardiography, in *Instrumentation and Measurement Technology Conference*, 2001. *IMTC* 2001. *In*: *Proceedings of the 18th IEEE*, p. 32-37 (1).

[11] Salah, M., Zekry, A. H., Kamel, M. (2011). FPGA implementation of LMS adaptive filter, in *National Radio Science Conference (NRSC)*, 2011 28*th*, p. 1-8.

[12] Mustafa, R., Mohd Ali, M. A., Umat, C., Al-Asady, D. A. (2009). Design and implementation of least mean square adaptive filter on Altera Cyclone II Field Programmable Gate Array for active noise control," in *Industrial Electronics & Applications, 2009. ISIEA 2009. IEEE Symposium on*, p. 479-484.

[13] Choi, S., Cichocki, A., Park, H., Lee, S. (2005). Blind Source Separation and Independent Component Analysis: A Review., *Neural Information Processing (Letters and Reviews)*, (6), p. 1-57.

.[14] Kuo-Kai, S., Ming-Huan, L., Yu-Te, W., Po-Lei, L. (2008). Implementation of Pipelined FastICA on FPGA for Real-Time BlindSource Separation, *Neural Networks, IEEE Transactions on*, 9, p. 958-970.

[15] Sattar, F., Charayaphan, C. (2002). Low-cost design and implementation of an ICA-based blind source separation algorithm, in *ASIC/SOC Conference*, 2002. 15*th* AC. *Annual IEEE International*, 2002, p. 15-19.

[16] Mirchandani, G., Zinser, R. L., Jr., Evans, J. B. (1992). A new adaptive noise cancellation scheme in the presence of crosstalk [speech signals], *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*,39, p. 681-694.

[17] Van, D., Compernolle., Van Gerven, S. . (1992). Feedforward and feedback in a symmetric adaptive noise canceler., in *Eusipco*-92, *Sixth European Signal Processing Conference Aug* 24-27, Brussels Belgium, 1992, p. 1081-1084.

[18] Lepauloux, L., Scalart, P., Marro, C. (2009). A Efficient Low-Complexity Algorithm for Crosstalk-Resistant Adaptive Noise-Canceller," in 17*th European Signal Processing Conference* (*EUSIPCO* 2009), Glasgow,Scotland, August 24-28, p. 204-208.

[19] Mirchandani, G., Gaus, Jr., R., Bechtel, L. K., (1986). Performance characteristics of a hardware implementation of the cross-talk resistant adaptive noise canceller, in *ICASSP 86*, Tokyo, p. 93-96.

[20] Parsa, V., Parker, P. A., Scott, R. N. (1996). Performance analysis of a crosstalk resistant Adaptive Noise Canceller, *Ieee Transactions on Circuits and Systems Ii-Analog and Digital Signal Processing*, 43, p. 473-482, July.

[21] Kou, S. M., Peng, W. M. (2000). Principle and applications of asymmetric crosstalk-resistant adaptive noise canceler, *Journal of the Franklin Institute*, 337, p. 57-71

[22] Compernolle, D. V., Gerven, S.V. (1992). Signal separation in a symmetric adaptive noise canceler by output decorrelation, in *Acoustics, Speech, and Signal Processing, . ICASSP*-92, 1992 *IEEE International Conference on*, p. 221 - 224.

[23] Szopos, E., Hedesiu, H. (2009). LabVIEW FPGA based noise cancelling using the LMS adaptive algorithm, *Acta Technica Napocensis*, (50), p. 5-8.

[24] Van, S., Gerven., Van Compernolle, D. (1995). Signal separation by symmetric adaptive decorrelation: stability, convergence, and uniqueness, *Signal Processing, IEEE Transactions on*, 43, p. 1602-1612.

[25] Ikeda, S., Some Trials of Blind Source Separation" (Speech-Speech) Retrieved 20th September 2013 from http://www.ism.ac.jp/~shiro/research/blindsep.html.