EURASIP Journal on Applied Signal Processing 2005:18, 3076–3086
© 2005 C. Charoensak and F. Sattar

Design of Low-Cost FPGA Hardware for Real-time ICA-Based Blind Source Separation Algorithm

Charayaphan Charoensak
School of Electrical and Electronic Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798
Email: ecchara@ntu.edu.sg

Farook Sattar
School of Electrical and Electronic Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798
Email: efsattar@ntu.edu.sg

Received 29 April 2004; Revised 7 January 2005

Blind source separation (BSS) of independent sources from their convolutive mixtures is a problem in many real-world multi-sensor applications. In this paper, we propose and implement an efficient FPGA hardware architecture for the realization of a real-time BSS. The architecture can be implemented using a low-cost FPGA (field programmable gate array). The architecture offers a good balance between hardware requirement (gate count and minimal clock speed) and separation performance. The FPGA design implements the modified Torkkola's BSS algorithm for audio signals based on ICA (independent component analysis) technique. Here, the separation is performed by implementing noncausal filters, instead of the typical causal filters, within the feedback network. This reduces the required length of the unmixing filters as well as provides better separation and faster convergence. Description of the hardware as well as discussion of some issues regarding the practical hardware realization are presented. Results of various FPGA simulations as well as real-time testing of the final hardware design in real environment are given.

Keywords and phrases: ICA, BSS, codesign, FPGA.

1. INTRODUCTION

Blind signal separation, or BSS, refers to performing inverse channel estimation despite having no knowledge about the true channel (or mixing filter) [1, 2, 3, 4, 5]. BSS technique has been found to be very useful in many real-world multisensor applications such as blind equalization, fetal ECG detection, and hearing aid. BSS method based on ICA technique has been found effective and thus commonly used [6, 7]. A limitation using ICA technique is the need for long unmixing filters in order to estimate inverse channels [1]. Here, we propose the use of noncausal filters [6] to shorten the filter length. In addition to that, using noncausal filters in the feedback network allows a good separation even in the case where the direct channels filters do not have stable inverses. A variable step-size parameter for adaptation of the learning process is introduced here to provide a fast and stable convergence.

FPGA architecture allows optimal parallelism needed to handle the high computation load of BSS algorithm in real time. Being fully custom-programmable, FPGA offers rapid hardware prototyping of DSP algorithms. The recent advances in IC processing technology and innovations in the architecture have made FPGA a suitable alternative to using powerful but expensive computing platform.

In spite of its potential for real-world applications, there have been very few published papers on real-time hardware implementation of the BSS algorithm. Many of the works such as [8] focus on the VLSI implementation and do not provide a detailed set of specifications of the BSS implementation that offer a good balance between hardware requirement (gate count and minimal clock speed) and separation performance. Here, we propose an efficient hardware architecture that can be implemented using a low-cost FPGA and yet offers a good blind source separation performance. Extensive set of experimentations, discussion on separation performance, and the proposal for future improvement are presented.

The FPGA design process requires familiarity with the associated signal processing. Furthermore, the developed FPGA prototype needed to be verified, through both functional simulation and real-time testing, in order to fully understand the advantages and pitfalls of the architecture under investigation. Thus, an integrated system-level environment for software-hardware co-design and verification is needed. Here, we carried out FPGA design of a real-time implementation of the ICA-based BSS using a new system-level design tool called System Generator from Xilinx.

The rest of the paper is organized as follows. Section 2 provides an introduction to BSS algorithm and the application of noncausal filters within the feedback network. Section 3 describes the hardware architecture of our FPGA design for the ICA-based BSS algorithm. Some critical issues regarding the implementation of the BSS algorithm using the limited hardware resources in the FPGA are discussed. Section 4 presents the system level design of the FPGA followed by detailed FPGA simulation results, synthesis result, and the real-time experimentation of the final hardware using real environment setup. The summary and the motivations for future improvement are given in Section 5.

2. BACKGROUND OF BSS ALGORITHM

2.1. *Infomax or entropy maximization criterion*

BSS is the main application of ICA, which aims at reducing the redundancy between source signals. Bell and Sejnowski [9] proposed an information-theoretic approach for BSS, which is referred to as the Infomax algorithm.

2.2. *Separation of convolutive mixture*

The Infomax algorithm proposed by Bell and Sejnowski works well only with instantaneous mixture and was further extended by Torkkola for the convolutive mixture problem [10]. As shown in Figure 1, minimizing the mutual information between outputs u_1 and u_2 can be achieved by maximizing the total entropy at the output.

This architecture can be simplified by forcing W_{11} and W_{22} to be mere scaling coefficients to achieve the relationships shown below [6, 11]:

$$\begin{aligned} u_1(t) &= x_1(t) + \sum_{k=0}^{L_{12}} w_{12}(k)u_2(t-k), \\ u_2(t) &= x_2(t) + \sum_{k=0}^{L_{21}} w_{21}(k)u_1(t-k), \end{aligned} \tag{1}$$

and the learning rules for the separation matrix:

$$\Delta w^{ij} \propto (1-2y_i)u_j(t-k). \tag{2}$$

2.3. *Modified ICA-based BSS method using modified Torkkola's feedback network*

Torkkola's algorithm works only when the stable inverse of the direct channel filters exists. This is not always guaranteed in real-world systems. Considering the acoustical condition when the direct channel can be assumed a nonminimum phase FIR filter, the impulse response of its stable inverse will become noncausal infinite double-sided converging sequence, or a noncausal IIR filter. By truncating this noncausal IIR filter, the stable noncausal FIR filter (W_{12} or W_{21}) can be realized [12]. It was also shown in [6] that the algorithm can be easily modified to use the noncausal unmixing FIR filters. The relationships between the signals are now changed to

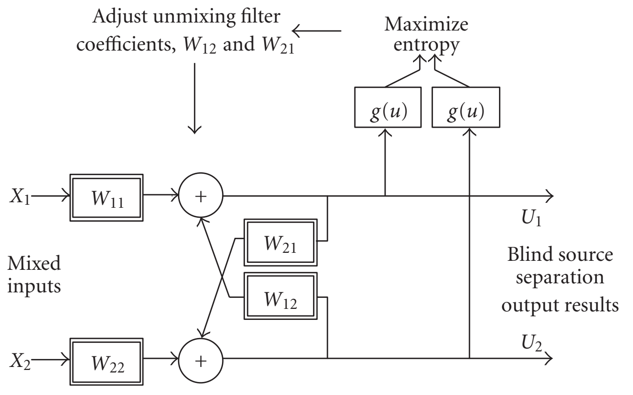
Adjust unmixing filter coefficients, W_{12} and W_{21}
Maximize entropy
$g(u)$
$g(u)$
X_1
W_{11}
W_{21}
W_{12}
Mixed inputs
X_2
W_{22}
U_1
Blind source separation output results
U_2

FIGURE 1: Torkkola's feedback network for BSS.

$$u_1(t) = x_1(t+M) + \sum_{k=-M}^{M-1} w_{12}(k)u_2(t-k), \tag{3}$$

$$u_2(t) = x_2(t+M) + \sum_{k=-M}^{M-1} w_{21}(k)u_1(t-k), \tag{4}$$

where M is half of the filter length, L, that is, $L = 2M + 1$ and the learning rule

$$\Delta w^{ij}_{(t_1-p_1+M)} = \Delta w^{ij}_{(t_0-p_0+M)} + K(u_i(t_0)u_j(p_0)), \tag{5}$$

where

$$K(u_i(t_0)) = \lambda(1-2y_i(t_0)), \tag{6}$$

$$\begin{aligned} y_i(t_0) &= \frac{1}{1+e^{-u_i(t_0)}}, \\ t_1 &= t_0 + 1, \\ p_0 &= t_0 - k \quad \text{for } k = -M, -M+1, \ldots, M, \\ p_1 &= t_1 - k \quad \text{for } k = -M, -M+1, \ldots, M. \end{aligned} \tag{7}$$

The term λ in (6) represents the variable learning step size which is explained in more detail in Section 3.5.

3. ARCHITECTURE OF HARDWARE FOR BSS ALGORITHM

The top-level block diagram of our hardware architecture for the BSS algorithm based on Torkkola's network is shown in Figure 2. The descriptions for the subsystems in the figure will be discussed in the following subsections. Some critical issues regarding the practical realization of the algorithm while minimizing the hardware resources are discussed.

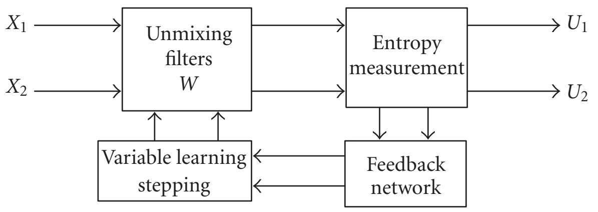

Figure 2: Top-level block diagram of hardware architecture for BSS algorithm based on Torkkola's network.

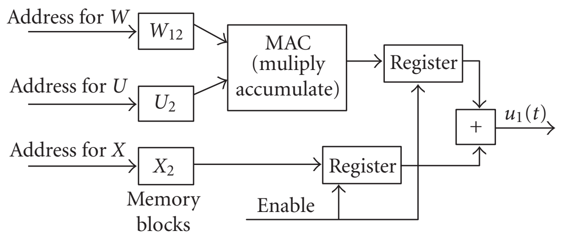

Figure 3: Implementation of (8) for the modified Tokkola's feedback network.

3.1. Practical implementation of the modified Torkkola's network for FPGA realization

In order to understand the effect of each filter parameter such as filter length (L) and learning step size on the separation performance, a number of simulation programs written in MATLAB were tested. A set of compromised specification is then proposed for practical hardware realization [11].

As a result of our MATLAB simulations, we propose that in order to reduce the FPGA resource needed, as well as to ensure real-time BSS separation given the limited maximum FPGA clock speed, the specifications shown below are to be used. Sections 3.2 and 3.6 explain the impact of some parameters on hardware requirement in more details.

(i) Filter length, $L = 161$ taps.
(ii) Buffer size for iterative convolution, $N = 2500$.
(iii) Maximum number of iterations, $I = 50$.
(iv) Approximation of the exponential learning step size using linear piecewise approximation.

The linear piecewise approximation is used to avoid complex circuitry needed to implement the exponential function (see Section 3.5 for more explanation of implementation, and Section 4.2.1 for comparison of simulation results using exponential function and linear piecewise function).

There are many papers discussing the effect of filter length on the separation performance [13]. We have chosen a smaller filter length considering the maximum operating speed of the FPGA and considered only the case of echo of a small room. (See Section 3.6 for calculation of required FPGA clock speed and Section 4.2.3 for the FPGA simulation result.)

3.2. Three-buffer technique

In real-time hardware implementation, to achieve an uninterrupted processing, the hardware must process the input and output as streams of continuous sample. However, this is in contrast with the need of batch processing of BSS algorithm [14]. To perform the separation, a block of data buffer has to be filtered iteratively. Here, we implement a buffering mechanism using three 2500-sample ($N = 2500$) buffers per one input source. While one buffer is being filled with the input data, a second buffer is being filtered, and the third buffer is being streamed out.

A side effect of this three-buffer technique is that the system produces a processing delay equivalent to twice the time needed to fill up a buffer. For example, if the signal sampling frequency is 8000 Hz, the time to fill up one buffer is $2500/8000 = 0.31$ second. The system will then need another 0.31 second to process before the result being ready for output. The total delay is then $0.31 + 0.31 = 0.62$ s. This processing delay may be too long for many real-time applications. A suggestion on applying overlapped processing windows, together with polyphase filter, in order to shorten this delay is given in Section 5.

3.3. Implementation of feedback network mechanism

According to feedback network in (3), there is a need to address negative addresses for the values of $w_{12}(i)$ when $i < 0$. In practice, the equation is modified slightly to include only positive addresses:

$$u_1(t) = x_1(t+M) + \sum_{i=-M}^{M} w_{12}(i+M)u_2(t-i). \tag{8}$$

Equation (8) performs the same noncausal filtering on u_2 as in (3) without the need for negative addressing of w_{12}. Equation (4) is also modified accordingly.

The block diagram shown in Figure 3 depicts the hardware implementation of (8). Note that the implementation of the FIR filtering of w_{12} is done through multiply-accumulate unit (MAC) which significantly reduces the numbers of multipliers and adders needed when compared to direct parallel implementation. The tradeoff is that the FPGA has to operate at oversampling frequency (see Section 3.6).

3.4. Mechanism for learning the filter coefficients

The mechanism for learning of the filter coefficients was implemented according to (5). The implementation of the variable learning step size, λ, is explained in the next subsection.

3.5. Implementation of variable learning step size

In order to speed up the learning of the filter coefficients shown in (5), we implement a simplified variable step-size technique.

In our application, the variable learning step size in (6), that is, the learning step size λ, may be implemented using

(9) below where n is the iteration level, λ_0 is the initial step size, and I is the maximum number of iterations, that is, 50:

$$\lambda = \exp\left(- u_0 - \frac{n}{I}\right), \tag{9}$$

where

$$u_0 = - \log_2 (\lambda_0) - \frac{1}{I}. \tag{10}$$

The exponential term is difficult to implement in digital hardware. Lookup table could be used but will require a large block of ROM (read only memory). Alternative to using lookup ROM is the CORDIC algorithm (COrdinate Rotation DIgital Computer) [15]. However, circuitry for CORDIC algorithm will impose a very long latency (if not heavily pipelined) which will result in the need for even higher FPGA clock speed. Instead, we used a linearly decreasing variable step size as shown:

$$\lambda = 0.0006 - 0.000012n. \tag{11}$$

We had carried out MATLAB simulations to compare the separation performance using exponential equation and linear piecewise term equation. It was found out that, using the specifications given in Section 3.1, there is no significant degradation in separation performance (see also simulation results in Section 4.2.1). A multipoint piecewise approximation will be implemented in future improvement.

3.6. *Calculation of the required FPGA clock speed*

As mentioned earlier that in order to save hardware resource, multiply-accumulate (MAC) technique is used. This implies that MAC operation has to be done at a much higher rate than that of the input sampling frequency. This MAC operating frequency is also the frequency of the FPGA clock. Thus, a detailed analysis of the FPGA system clock needed for real-time blind source separation is required.

The FPGA clock frequency can be calculated as shown in (12) (see also [11]) where Fs is the sampling frequency of the input signals, L is the tap length of the FIR filter, and I is the number of iterations:

$$\text{FPGA clock frequency} = L \cdot I \cdot Fs. \tag{12}$$

In our FPGA design, filter tap $L = 161$, iterations $I = 50$, and input sampling frequency $Fs = 8000$ Hz; the FPGA clock frequency is thus

$$161 \cdot 50 \cdot 8000 = 64.4\,\text{MHz}. \tag{13}$$

This means that the final FPGA design must operate properly at 64.4 MHz. In practice, the maximum operating speed of a hardware circuit can be optimized by analyzing the "critical path" in the design. A more detailed analysis of the critical path in the FPGA design is given in Section 4.1.2.

Note that the frequency 64.4 MHz also represents the number of multiplications per second needed to perform the blind source separation (per channel). This represents a very large computation load if a general processor, or DSP (digital signal processor), is to be used for real-time applications. Using fully hardware implementation, the performance gain is easily obtained by bypassing the fetch-decode-execute overhead, as well as by exploiting the inherent parallelism. FPGAs allow the above-mentioned advantage plus the reprogrammability and cost effectiveness.

Some discussion on applying polyphase filter to increase the filter length while maintaining the FPGA clock speed is given in section 5.

4. SYSTEM-LEVEL DESIGN AND TESTING OF THE FPGA FOR BSS ALGORITHM

4.1. *System-level design of FPGA for BSS algorithm*

System generator provides a bit-true and cycle-true FPGA blocksets for functional simulation under MATLAB Simulink environment thus offering a very convenient and realistic system-level FPGA co-design and cosimulation. Some preliminary theoretical results from MATLAB m-file can directly be used as reference for verifying the FPGA results (refer to [16] and http://www.xilinx.com/system_generator for more detailed description).

Note that the FPGA design using system generator is different from the more typical approach using HDL (hardware description language) or schematics. Using system generator, the FPGA is designed by means of Simulink models. Thus, the FPGA functional simulation can be carried out easily right inside Simulink environment. After the successful simulation, the synthesizable VHDL (VHSIC HDL where VHSIC is very-high-speed integrated circuit) code is automatically generated from the models. As a result, one can define an abstract representation of a system-level design and easily transform it into a gate-level representation in FPGA.

The top level design of the ICA-based BSS algorithm based on the described architecture is shown in Figure 4. As can be seen from the figure, the FPGA reads in the data from each wave file, and outputs the separated signals, as streams of 16-bit data sample. The whole design is made up of many subsystems. Here, a more detailed circuit design of the subsystem which implements the computation for updating the filter coefficients is shown in Figure 5.

The process of FPGA design is simplified. The Simulink design environment also enhances the system-level software-hardware co-design and verification. The upper right-hand side of Figure 3 shows how to use Simulink "scope" to display the waveform generated by the FPGA during simulation. This technique can be used to display the waveform at any point in the FPGA circuitry. This offers a very practical way to implement a software-hardware co-design and verification. Note also the use of the blockset "to wave file" to transfer the simulation outputs from the FPGA back to a wave file for further analysis.

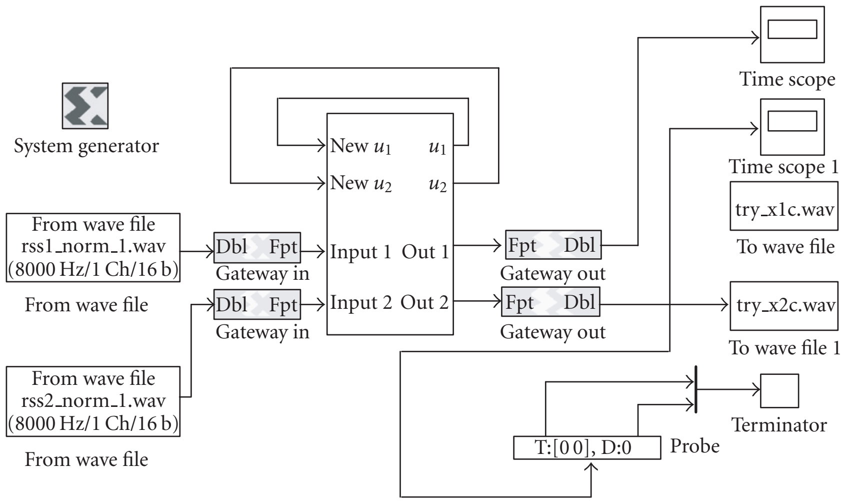

Figure 4: Top-level design of BSS using system generator under MATLAB Simulink.

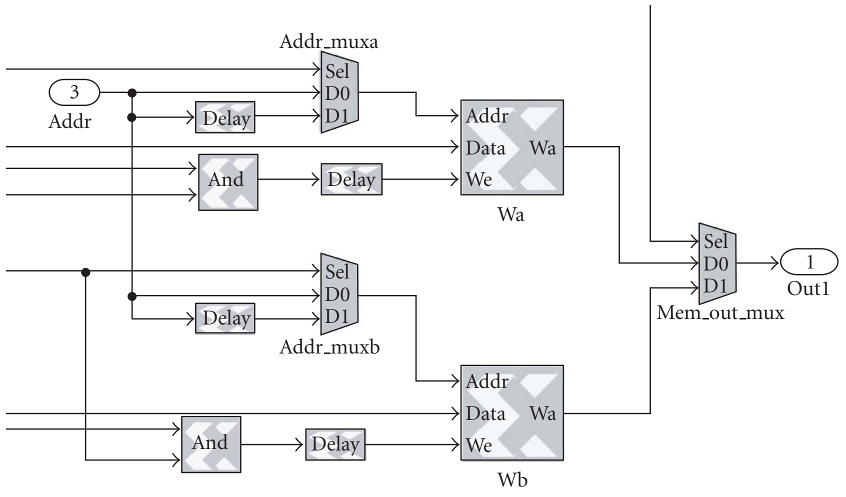

Figure 5: Detailed circuit for updating the filter coefficients.

4.1.1. Using fixed-point arithmetic in FPGA implementation

When simulating a DSP algorithm using programming languages such as C and MATLAB, double precision floating point numeric system is commonly used. In hardware implementation, fixed-point format numeric is more practical. Although several groups have implemented floating-point adders and multipliers using FPGA devices, very few practical systems have been reported [17]. The main disadvantages of using floating point in FPGA hardware are higher resource requirements, higher clock frequency, and longer design time than an equivalent fixed-point system.

For fixed filters, the analysis of the effect of fixed-point arithmetic on the filter performance has been well presented in other publications [18, 19]. An analysis of word length effect on the adaptation of LMS algorithm is given in [18].

We use only fixed-point arithmetic in our design. As mentioned earlier, we use 16-bit fixed-point at the inputs and outputs of our FPGA design. In practice, depending on the applications, the size of data width should be selected taking into account the desired system performance and FPGA gate count. In terms of separation performance, it can be shown that word length of the accumulator unit in the MAC unit is most critical. Using system generator, it is easy to study the effect of round-off error on the overall system performance and that helps in selecting the optimal bit size in the final FPGA design.

4.1.2. Analysis of critical path in FPGA implementation of BSS algorithm

It was shown in Section 3.6 that in order for the FPGA to perform the blind source separation in real time, the required

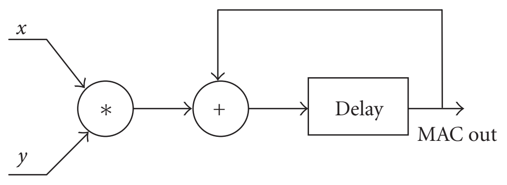

Figure 6: Simplified block diagram of MAC operation.

clock frequency is 64.4 MHz. We will now analyze the critical paths in the ICA-based BSS design in order to verify that the clock frequency can be met. Referring to Figures 1 and 2, one can see that the maximum operating speed of the FPGA is determined by the critical path in the multiply-accumulate blocks. The propagation delay in this critical path is the summation of the combinational delay made up of one multiplier and one adder as shown in the simplified block diagram in Figure 6. Based on Xilinx data sheet of Virtex-E FPGA, delays of the dedicated pipeline multiplier and adder units are

(i) delay of 16-bits adder: 4.3 nanoseconds,
(ii) delay of pipelined 16-bit multiplier: 6.3 nanoseconds.

Thus, we can approximate the total delays in the critical path to be 4.3 + 6.3 = 10.6 nanoseconds, which converts to 94 MHz maximum FPGA clock frequency. This is much higher than the required 64.4 MHz needed. Note that this is only an approximation and we mention it here for the purpose of discussion about our BSS architecture. The accurate maximum clock speed can easily be extracted from the FPGA synthesis result which is given in Section 4.3. Note also that the available dedicated multipliers depend very much on the FPGA family and the size of device used.

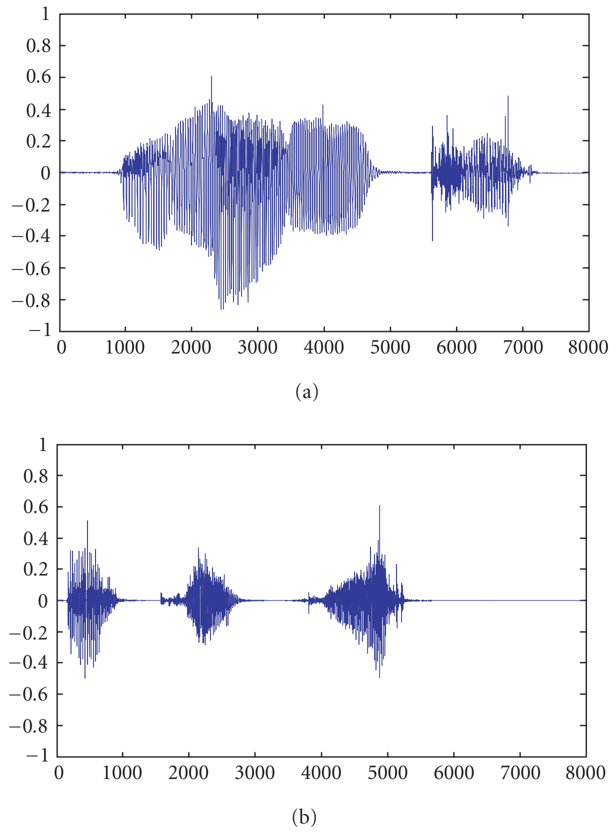

Figure 7: Two original female voices used for FPGA simulations: (a) first female voice and (b) second female voice.

4.2. Simulation results of the FPGA design for ICA-based BSS

We carried out experimentations using different wave files, all sampled at 8000 Hz, and with various types of mixing conditions. The output separation performance results were measured.

4.2.1. FPGA simulation using two female voices

In this simulation, we use wave files of two female voices. Each file is approximately one-second long, with sampling frequency of 8000 samples per second, and 16 bits per sample. The files were preprocessed to remove dc component and amplitude normalized. The two original input voices are shown in Figures 7a and 7b, respectively.

To simulate the mixing process, the two inputs were processed using the instantaneous mixing program called "*instamix.m*" downloaded from the website http://www.ele.tue.nl/ica99 given in [20]. The mixing matrix used is [0.6 1.0, 1.0 0.6]. The two mixed voices are shown in Figures 8a and 8b.

The separation results from the FPGA simulation are shown in Figures 9 and 10. The separated outputs are shown in Figures 9a and 9b. By comparing the separated output voice in Figure 9a to the corresponding mixed voice in Figure 8a, the separation is clearly visible. By listening to the output, the voice is much more intelligible. Similar conclusion can be drawn for the other voice (the measurements to show the separation performance are given in the next subsection).

The error of the separated first female voice, measured by subtracting the output in Figure 9a from the original input in Figure 7a, is shown in Figure 10a. Figure 10b shows error of the second voice.

Figures 11a and 11b show the FPGA simulation results using linear piecewise function, compared to exponential function shown in (7). Figure 11a shows the plot of learning step sizes, using exponential function and linear piecewise function, against number of iterations. Figure 11b compares the averaged changes of filter coefficients Δw_{12} and Δw_{21}, for all the 161 taps, plotted against number of iterations. It can be seen that, for the maximum number of iterations used (= 50), both the learning step size using exponential function and the learning step size using linear piecewise function converge to zeros properly.

4.2.2. FPGA simulation using one female voice mixed with Gaussian noise

A second experiment was carried out to measure the separation performance of the FPGA under noisy environment [21]. The first female voice used in the last experimentation

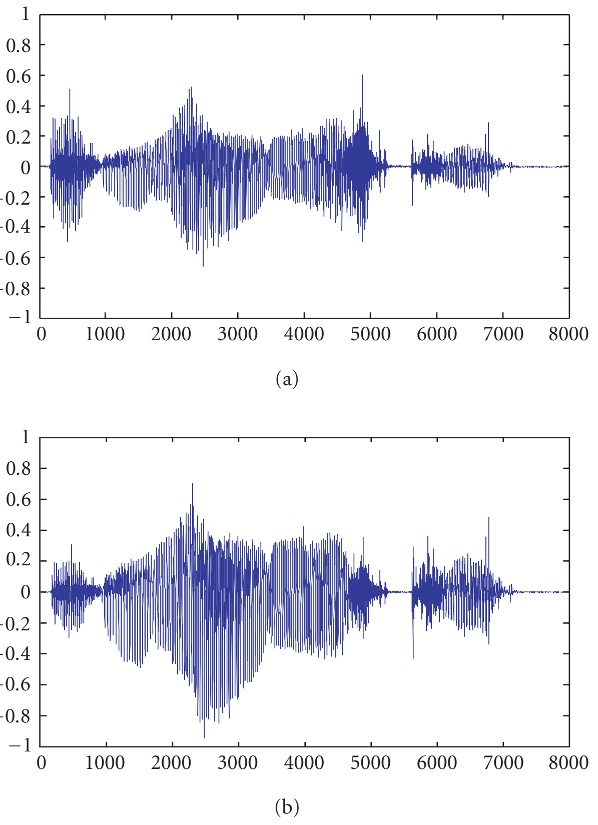
FIGURE 8: Two mixed female voices used for FPGA simulation. The program "*instamix.m*" downloaded from http://www.ele.tue.nl/ica99 is used; (a) mixed first signal and (b) mixed second input.

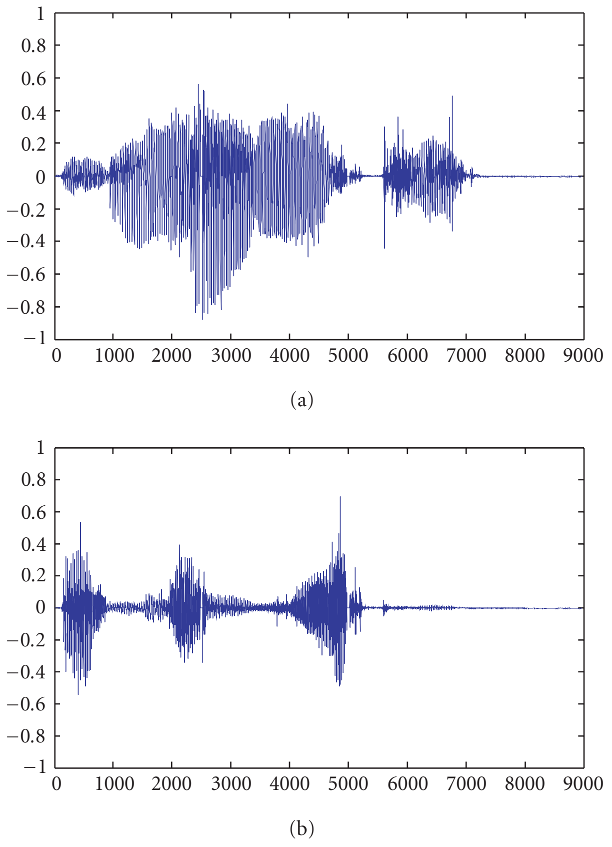
FIGURE 9: Two separated voices from FPGA simulation; (a) separated first voice and (b) separated second voice.

was mixed with white Gaussian noise (see Figure 12) and the signal-to-noise ratios (SNRs), before and after the BSS operation by the designed FPGA, were measured. The same mixing matrix [0.6 1.0, 1.0 0.6] was used. By adjusting the variance of the white Gaussian noise source, σ^2, the input SNR varies in the range −9 dB to 10 dB.

The input signal-to-noise ratio, SNRi, is defined in this experimentation as

$$\text{SNRi (dB)} = 10\log\left[\frac{\sum_{i=1}^{T} x_1^2(i)}{\sum_{i=1}^{T} x_2^2(i)}\right]. \tag{14}$$

Here, T is the total number of samples in the time period of measurement and $x_1(i)$ and $x_2(i)$ are the original female voice and the white Gaussian noise respectively.

Similarly, the output signal-to-noise ratio, SNRo, is defined in (15). $e_1(i)$ represents the overall noise left in the first separated output u_1 and defined as $e_1(i) = u_1(i) - x_1(i)$ as shown in Figure 12:

$$\text{SNRo (dB)} = 10\log\left[\frac{\sum_{i=1}^{T} x_1^2(i)}{\sum_{i=1}^{T} e_1^2(i)}\right]. \tag{15}$$

The improvement of SNR after BSS processing, SNRimp (dB), is defined as

$$\text{SNRimp (dB)} = 10\log\left[\frac{\sum_{i=1}^{T} x_2^2(i)}{\sum_{i=1}^{T} e_1^2(i)}\right]. \tag{16}$$

The result in Figure 13 shows the averaged output SNRs and the average improvement of SNRs plotted against input SNRs. It can be seen that as the input SNRs varies, the output SNRs is almost the constant at approximately 35 dB. The maximum achievable out SNRs is limited by the width of the datapath implemented inside the FPGA. Due to this reason, the amount of improvement of SNRs decreases with the increasing input SNRs.

4.2.3. *FPGA simulation using two female voices mixed using simulated room environment*

Next experimentation was carried out to measure separation performance of the designed FPGA under realistic room environment using the same two female voices. The room environment was simulated using the program "*simroommix.m*" downloaded from the same website mentioned earlier. The coordinates (in meter) of the first and second signal sources

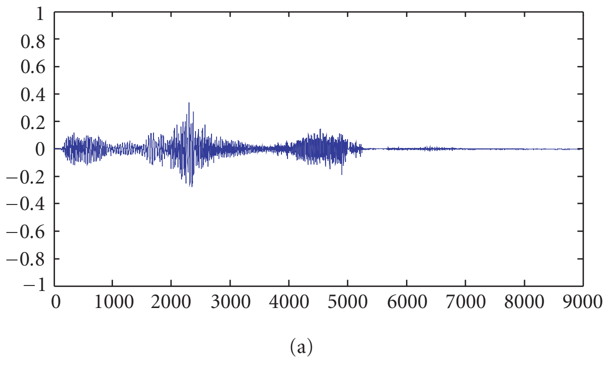
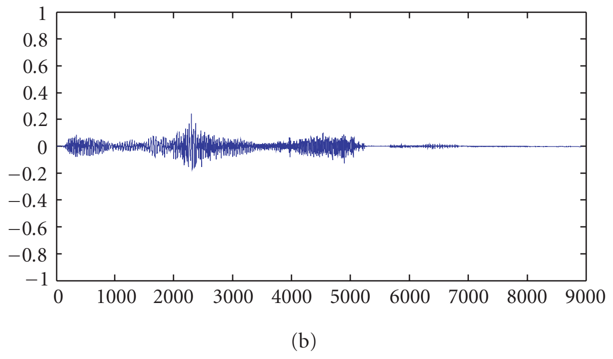

Figure 10: Error of the two separated voices; (a) error of separated first voice and (b) error of separated second voice.

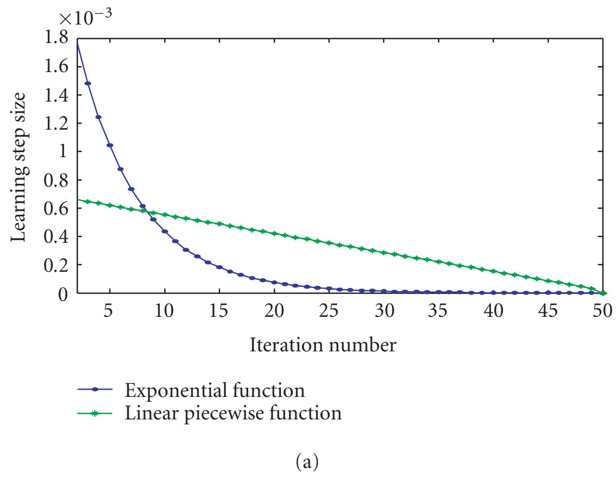
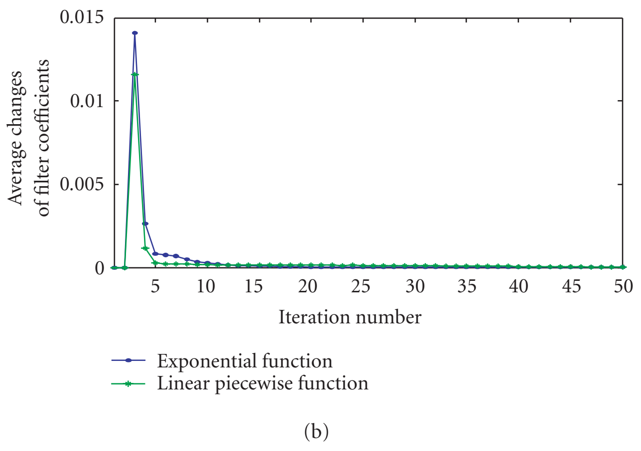

Figure 11: (a) Learning step sizes using exponential and linear piecewise functions and (b) averaged changes of filter coefficients Δw_{12} and Δw_{21} using the two functions.

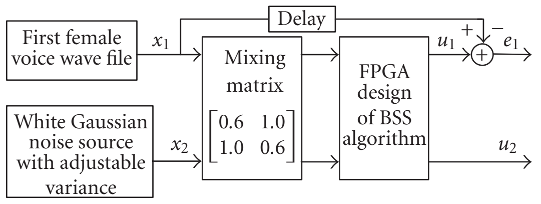
First female voice wave file
White Gaussian noise source with adjustable variance
x_1
x_2
Delay
Mixing matrix
0.6 1.0
1.0 0.6
FPGA design of BSS algorithm
u_1
u_2
+
−
e_1

Figure 12: FPGA simulation to test BSS using one female voice mixed with white Gaussian noise with adjustable variance σ^2.

are $(2, 2, 2)$ and $(3, 4, 2)$ respectively. The locations of the first and second microphone are $(3.5, 2, 2)$ and $(4, 3, 2)$ respectively. The room size is $5 \times 5 \times 5$. The simulated room arrangement is shown in Figure 14.

The measurement of separation performance is based on using (17) (see [20]). Here, S_j is the separation performance of the jth separated output where $u_{j,xj}$ is the jth output of the cascaded mixing/unmixing system when only input x_j is active; $E[u]$ represents the expected value of u:

$$S_j = 10 \log \left(\frac{E\left\{ \left(u_{j,x_j}\right)^2 \right\}}{E\left\{ \left(\sum_{i \neq j} u_{j,x_i} \right)^2 \right\}} \right). \tag{17}$$

The program "*bsep.m*," which performs the computation as shown in (17), was downloaded from the website and used to test our FPGA. It was found that before the BSS,

$$S_1 = 17.29\,\text{dB}, \qquad S_2 = 10.63\,\text{dB}, \tag{18}$$

and after the BSS operation,

$$S_1 = 20.29\,\text{dB}, \qquad S_2 = 16.53\,\text{dB}. \tag{19}$$

Thus, the BSS hardware improves the separation by 3 dB and 5.9 dB for channels 1 and 2, respectively. The figures may not appear very high. However, by listening to the separated output signals, the improvement was obvious and further improvement can be made by increasing the filter lengths. Note also that the measurements of separation performance and the improvement after the BSS depend very much on the room arrangement.

4.2.4. FPGA simulation using convolutive mixture of voices

In order to test the FPGA using convolutive mixtures recorded in a real cocktail party effect, two samples voices were downloaded from a website (T-W Lee's website

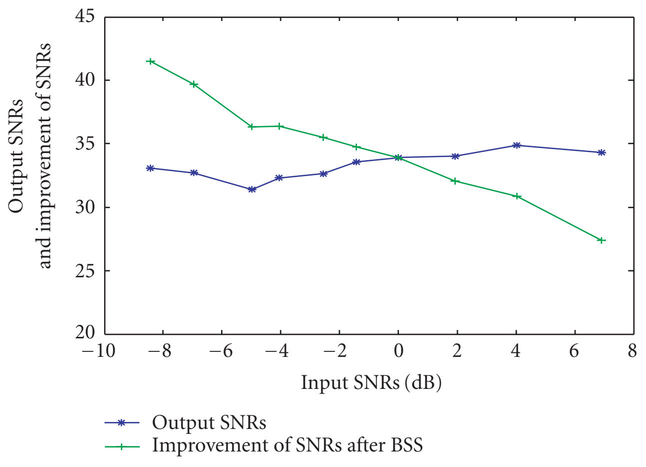

FIGURE 13: Results of SNR measurements using one female voice mixed with white Gaussian noise. The figure shows the output SNRs and the improvement of SNRs against the input SNRs.

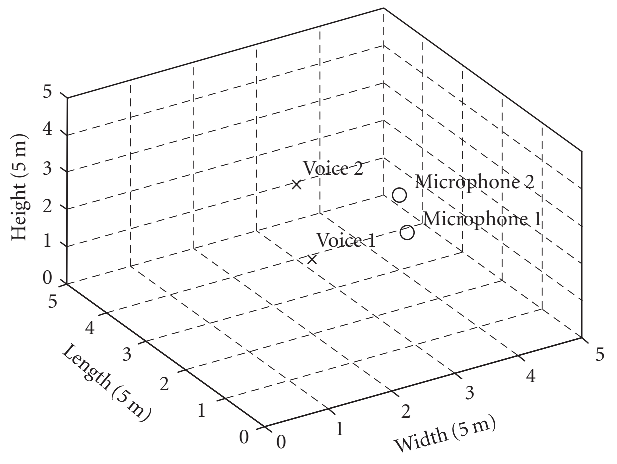

FIGURE 14: Arrangement of female voice sources and microphones in simulated room environment using program "*simroommix.m*."

at http://inc2.ucsd.edu/~tewon/) and used in the FPGA simulation. The two wave files used were from two speakers recorded speaking simultaneously. Speaker 1 counts the digits from one to ten in English and speaker 2 counts in Spanish. The recording was done in a normal office room. The distance between the speakers and the microphones was about 60 cm in a square ordering. A commercial program was used to resample the original sampling rate of 16 kHz down to 8000 Hz. The separation results from the developed FPGA can be found at our website (http://www.ntu.edu.sg/home/ecchara/—click project on the left frame). Note that we present a real-time experimentation using these convolutive mixtures in Section 4.4. We have also posted the separation results of the same convolutive mixtures using MATLAB program at the same website for comparison.

Because the original wave files (before the convolutive mixing) were not available, we could not perform the measurement of the separation performance. However, by listening to the separated outputs, we notice approximately the same level of separation as in our earlier experimentations.

TABLE 1: Detailed gate requirement of the BSS FPGA design.

Number of slices for logic	550
Number of slices for flip flops	405
Number of 4 input LUTs	3002
used as LUTs	2030
used as a route-thru	450
used as shift registers	522
Total equivalent gate count for the design	100 213

TABLE 2: Maximum combinational path delay and operating frequency of the FPGA design for BSS.

Maximum path delay from/to any node	15.8 ns
Maximum operating frequency	71.2 MHz

4.3. FPGA synthesis result

After the successful simulation, the VHDL codes were automatically generated from the design using system generator. The VHDL codes were then synthesized using Xilinx ISE 5.2i and targeted for Virtex-E, 600 000 gates. The optimization setting for the ISE is for maximum clock speed. Table 1 details the gate requirement of the FPGA design. The total gate requirement reported by the ISE is approximately 100 kgates. The price of the FPGA device in this range of gate size is very low. Note that all of the buffers are implemented using external memory.

Table 2 shows the reported maximum path delay and the highest FPGA clock frequency.

4.4. Real-time testing of FPGA design

The real-time testing of the FPGA was done using a prototype board equipped with a 600 000-gate Virtex-E FPGA device. The system setup is shown in Figure 15. The wave files of the convolutive mixtures used in Section 4.2.4 were encoded into the left and right channels of a stereo MP3 file which is then played back repeatedly using a portable MP3 player. The prototype board is equipped with 20-bit A/D and D/A converters and the sampling frequency was set to 8000 samples per second. Only the highest 16 bits of the sampled signals were used by the FPGA.

The FPGA streamed out the separated outputs which were then converted into analog signals, amplified, and played back on the speaker. By listening to the playback sound, it was concluded that we had achieved the same level of separation as found in earlier simulations.

5. SUMMARY

In this paper, the hardware implementation of the modified BSS algorithm was realized using FPGA. A set of compromised specification for the BSS algorithm was proposed

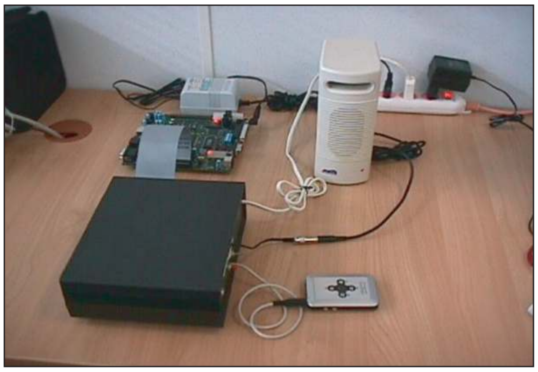
Figure 15: System setup for real-time testing of the blind source separation in real time.

taking into consideration the separation performance and hardware resource requirements. The design implements the modified Torkkola's BSS algorithm using noncausal unmixing filters which reduce filter length and provides faster convergence. There is no whitening process of the separated sources involved except that there is a scaling applied in the mixed inputs proportional to their variances.

The FPGA design was carried out using a system level approach and the final hardware achieves the real-time operation using minimal FPGA resource. Many FPGA functional simulations were carried out to verify and measure the separation performance of the proposed simplified architecture. The test input signals used include additive mixtures, convolutive mixtures, and simulated room environment. The results show that the method is robust against noise; that is, producing a small variation of the output SNR with respect to a large variation in the input SNR. Note that here we have considered the sensors to be almost perfect, or high quality with negligible sensor error.

A relatively short filter length of 161 tap is first attempted here due to the limitation of the maximum clock speed of the FPGA. The FPGA can perform the separation successfully when the delay is small, that is, less than $161/8000 = 20$ ms. In the environment where the delay is longer, a much longer filter tap is needed and this can be realized. In this case, we propose that, in order to keep the FPGA clock frequency the same, multiple MAC engines should be used together with the implementation of polyphase decomposition. For example, if the tap length of $161 * 16 = 2576$ is needed, 16 MACs engines are to be implemented with the 16-band polyphase decomposition. Since one MAC engine is made up of only one multiplier and one accumulator, the additional MAC engines will not lead to much additional gates.

The application of block mode separation leads to the side effect of a long processing delay. It was shown in Section 3.2 that our current design poses a long delay of 0.62 s. A practical solution to this long delay is to apply overlapped processing windows. For example, if the processing delay is to be reduced to $0.62/32 = 20$ ms, the 2500-sample windows will have to be overlapped by $31/32 = 97\%$, that is, the BSS has to be performed for every 78 input samples.

In this paper, we consider the case of 2 sources (or 2 voices) and 2 sensors (or 2 microphones). In the future, we will carry out further improvements in our FPGA architecture to tackle the situation when the number of sources is higher, or lower, than the number of sensors. Using our existing design, we have done some FPGA simulation using 3 voices with 2 microphones. The separation results are good (please visit the website http://www.ntu.edu.sg/home/ecchara/ for listening test). In this situation, two of the three voices are successfully separated (from each other) while the third suppressed voice is still present at both of the outputs. We will improve on our current FPGA design to handle this situation. Considering the fact that redesigning the FPGA takes much time, we will carry out the above improvements in our future works.

REFERENCES

[1] T.-W. Lee, *Independent Component Analysis: Theory and Applications*, Kluwer Academic, Hingham, Mass, USA, 1998.

[2] R. M. Gray, *Entropy and Information Theory*, Springer, New York, NY, USA, 1990.

[3] P. Comon, "Independent component analysis, a new concept?" *Signal Processing*, vol. 36, no. 3, pp. 287–314, 1994.

[4] K. Torkkola, "Blind source separation for audio signals—are we there yet?" in *Proc. IEEE International Workshop on Independent Component Analysis and Blind Signal Separation (ICA '99)*, pp. 239–244, Aussois, France, January 1999.

[5] T.-W. Lee, A. J. Bell, and R. Orglmeister, "Blind source separation of real world signals," in *Proc. IEEE International Conference on Neural Networks (ICNN '97)*, vol. 4, pp. 2129–2134, Houston, Tex, USA, June 1997.

[6] F. Sattar, M. Y. Siyal, L. C. Wee, and L. C. Yen, "Blind source separation of audio signals using improved ICA method," in *Proc. 11th IEEE Signal Processing Workshop on Statistical Signal Processing (SSP '01)*, pp. 452–455, Singapore, August 2001.

[7] H. H. Szu, I. Kopriva, and A. Persin, "Independent component analysis approach to resolve the multi-source limitation of the nutating rising-sun reticle based optical trackers," *Optics Communications*, vol. 176, no. 1-3, pp. 77–89, 2000.

[8] C. Abdullah, S. Milutin, and C. Gert, "Mixed-signal real-time adaptive blind source separation," in *Proc. International Symposium on Circuits and Systems (ISCAS '04)*, pp. 760–763, Vancouver, Canada, May 2004.

[9] A. J. Bell and T. J. Sejnowski, "An information-maximisation approach to blind separation and blind deconvolution," *Neural Computation*, vol. 7, no. 6, pp. 1129–1159, 1995.

[10] K. Torkkola, "Blind separation of convolved sources based on information maximization," in *Proc. IEEE Signal Processing Society Workshop on Neural Networks for Signal Processing VI*, pp. 423–432, Kyoto, Japan, September 1996.

[11] F. Sattar and C. Charayaphan, "Low-cost design and implementation of an ICA-based blind source separation algorithm," in *Proc. 15th Annual IEEE International ASIC/SOC Conference (ASIC/SOC '02)*, pp. 15–19, Rochester, NY, USA, September 2002.

[12] B. Yin, P. C. W. Sommen, and P. He, "Exploiting acoustic similarity of propagating paths for audio signal separation," *EURASIP Journal on Applied Signal Processing*, vol. 2003, no. 11, pp. 1091–1109, 2003.

[13] H. Sawada, S. Araki, R. Mukai, and S. Makino, "Blind source separation with different sensor spacing and filter length for each frequency range," in *Proc. 12th IEEE Workshop on Neural

Networks for Signal Processing (NNSP '02), pp. 465–474, Martigny, Valais, Switzerland, September 2002.

[14] A. Cichocki and A. K. Barros, "Robust batch algorithm for sequential blind extraction of noisy biomedical signals," in *Proc. 5th International Symposium on Signal Processing and Its Applications (ISSPA '99)*, vol. 1, pp. 363–366, Brisbane, Queensland, Australia, August 1999.

[15] R. Andraka, "A survey of CORDIC algorithms for FPGAs," in *Proc. ACM/SIGDA 6th International Symposium on Field Programmable Gate Arrays (FPGA '98)*, pp. 191–200, Monterey, Calif, USA, February 1998.

[16] Xilinx, Xilinx System Generator v2.3 for the MathWorks Simulink: Quick Start Guide, February 2002.

[17] J. Allan and W. Luk, "Parameterised floating-point arithmetic on FPGAs," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '01)*, vol. 2, pp. 897–900, Salt Lake City, Utah, USA, 2001.

[18] J. R. Treichler, C. R. Johnson Jr., and M. G. Larimore, *Theory and Design of Adaptive Filters*, Prentice-Hall, Upper Saddle River, NJ, USA, 2001.

[19] S. Haykin, *Adaptive Filter Theory*, Prentice-Hall, Upper Saddle River, NJ, USA, 1996.

[20] D. Schobben, K. Torkkola, and P. Smaragdis, "Evaluation of blind signal separation methods," in *Proc. IEEE International Workshop on Independent Component Analysis and Blind Signal Separation (ICA '99)*, pp. 261–266, Aussois, France, January 1999.

[21] A. K. Barros and N. Ohnishi, "Heart instantaneous frequency (HIF): an alternative approach to extract heart rate variability," *IEEE Trans. Biomed. Eng.*, vol. 48, no. 8, pp. 850–855, 2001.

Charayaphan Charoensak received the M.A.S. and Ph.D. degrees in electrical engineering from the Technical University of Nova Scotia, Halifax, NS, Canada, in 1989 and 1993, respectively. He is currently an Assistant Professor with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, in the Division of Information Engineering. His research interests include applications of FPGAs in digital signal processing, sigma-delta signal processing, pattern recognition, face recognition, adaptive signals, speech/audio segmentation, watermarking, and image processing.

Farook Sattar is an Assistant Professor at the Information Engineering Division, Nanyang Technological University, Singapore. He has received his Technical Licentiate and Ph.D. degrees in signal and image processing from Lund University, Sweden, and M.Eng. degree from Bangladesh University of Technology, Bangladesh. His current research interests include blind signal separation, watermarking, speech/audio segmentation, speech enhancement, 3D audio, image feature extraction, image enhancement, wavelets, filter banks, and adaptive beamforming. He has training in both signal and image processing, and has been involved in a number of signal and image processing-related projects sponsored by the Swedish National Science and Technology Board (NUTEK) and Singapore Academic Research Funding (AcRF) Scheme. His research has been published in a number of leading journals and conferences.