Hindawi
International Journal of Reconfigurable Computing
Volume 2018, Article ID 3721756, 9 pages
https://doi.org/10.1155/2018/3721756

Research Article

On a Real-Time Blind Signal Separation Noise Reduction System

Ka Fai Cedric Yiu[1] **and Siow Yong Low**[2]

[1]*Department of Applied Mathematics, The Hong Kong Polytechnic University, Hung Hom, Hong Kong*
[2]*School of Electronics and Computer Science, University of Southampton, Malaysia Campus, Iskandar Puteri, Johor, Malaysia*

Correspondence should be addressed to Ka Fai Cedric Yiu; macyiu@polyu.edu.hk

Received 30 May 2018; Revised 22 October 2018; Accepted 13 November 2018; Published 4 December 2018

Academic Editor: John Kalomiros

Blind signal separation has been studied extensively in order to tackle the cocktail party problem. It explores spatial diversity of the received mixtures of sources by different sensors. By using the kurtosis measure, it is possible to select the source of interest out of a number of separated BSS outputs. Further noise cancellation can be achieved by adding an adaptive noise canceller (ANC) as postprocessing. However, the computation is rather intensive and an online implementation of the overall system is not straightforward. This paper intends to fill the gap by developing an FPGA hardware architecture to implement the system. Subband processing is explored and detailed functional operations are profiled carefully. The final proposed FPGA system is able to handle signals with sample rate over 20000 samples per second.

1. Introduction

Speech enhancement has found numerous applications in human machine interfaces, hearing aids, and even hearing protection devices [1, 2]. However, research in this field is still very much ongoing as enhancing speech in an online fashion is not an easy task. The objective of speech enhancement is to estimate the desired speech signal from the noisy observation, which consists of both speech and noise signals. The challenge lies in designing an optimal filter that can suppress noise while maintaining the perceptual and spectral features of the speech signal [3]. Moreover, it is essential for the speech enhancement process to be seamless and transparent. In most implementations, both the noise estimation and its suppression are typically performed in the frequency domain for computational simplicity. The main issue with speech enhancement is when the background noise is nonstationary, e.g., babble or cafeteria noise. As speech is also nonstationary, analyzing the overlapped spectral of speech and noise can be challenging. If the noise is erroneously estimated, it will result in the infamous "musical noise" effect [4, 5].

To resolve the nonstationarity issue, spatial filtering or beamforming can be used to spatially filter out the speech signal from the noisy signal [6–8]. In this case, localization information and the array geometry will need to be carefully incorporated into the beamforming design for a good performance. This is because beamforming is sensitive to the steering vector and localization errors [9, 10]. Any mismatch between the observed signals and the model signal will result in performance degradation. An interesting approach to bypass the need of a priori information and potential model mismatch is blind signal separation (BSS) [11–13]. BSS as it is widely known requires only statistical independence among its inputs to separate the mixed observed signals [14]. A closer analysis into BSS reveals that BSS primarily exploits spatial information to perform the separation of mixed signals just like a set of beamformers [15]. However, BSS in its original formulation remains a separation algorithm, which yields a number of separated outputs. In speech enhancement applications, typically there is only one source of interest in a noisy background.

Low et. al [16, 17] proposed the use of kurtosis measure to select the source of interest out of a number of separated BSS outputs. The other BSS outputs are then used for further noise cancellation in the form of references for an adaptive noise canceller (ANC). However, an online implementation of the BSS-ANC system is not straightforward. This paper extends the BSS-ANC in [16] by making it online and optimizing the

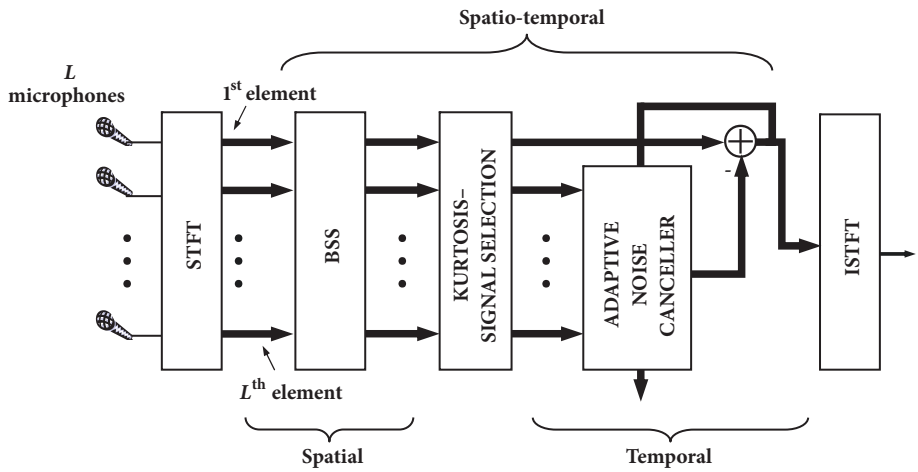


Figure 1: The spatiotemporal processor with L microphones. The figure shows how the kurtosis unifies the spatial and temporal decorrelators as one spatiotemporal processor. Each bold arrow represents the M frequency bins.

implementation by using multilevel parallelism [18, 19]. For the design of the resulting online system, since the calculations include the independent component analysis (ICA) which involves intensive matrix algebra, it is advantageous to implement it on a machine which allows massive parallelism. In general, microprocessor is not fast enough and ASIC is too inflexible when the separation is carried out adaptively. In view of this, we investigate the implementation of the proposed beamformer on an FPGA system.

In the literature, the implementation of a time-delay sonar beamformer on reconfigured devices has been reported [20]. The beamformer achieved six times speedup over dedicated DSP systems. Another beamformer implementation involves delta-sigma modulation, and the beamformer is applied to medical ultrasonic application [21]. There is also implementation for antenna signals [22] or for audio applications [23] in the time domain [24]. However, these studies do not consider subband processing and have not considered a parallel filter structure. It is therefore the main contribution of the paper to build the first FPGA hardware architecture with parallel beamforming filters for embedded systems. The complete architecture is implemented to simulate a real-time operation for the final signal separation system. The main contributions of the paper can be summarized as follows:

(i) In the FPGA hardware architecture, fixed point arithmetics are applied with a careful bitwidth analysis to explore suitable bitwidth of the system. The optimized integer and fraction size using fixed point arithmetic can reduce the overall circuit size significantly compared with a basic implementation of the algorithm in FPGA.

(ii) The hardware accelerator is used to perform the most time consuming part of the algorithm. We implement the algorithm and evaluate on a Virtex-4 platform. By calculating the number of samples handled per second, the proposed FPGA-based architecture can process a maximum of 22758 samples per second, which realizes the real-time capability.

Section 2 gives a description of the offline BSS-ANC speech enhancement system and Section 3 details the proposed hardware architecture and design. Specifically, the dataflow of the main operation and algortihmic profiling are presented. The experimental results are then included in Section 4 with Section 5 concluding the findings.

2. A Revisit to the BSS-ANC System

Figure 1 illustrates the BSS-ANC system. The BSS acts as a set of beamformers, which separates the target signal from the background noise by using the L observations. The signal is postprocessed by an ANC in order to enhance the target signal further and to carry out echo cancellation at the same time. To differentiate the target signal from the other outputs, a statistical measure is applied to guide the BSS.

2.1. Second-Order Based Blind Signal Separation (BSS). Second-order decorrelation is incapable of performing BSS as decorrelation does not imply independence. However, additional assumptions about the system can be incorporated to achieve separation. For instance, if the sources are nonstationary, then their respective covariances at different time intervals are linearly independent. This is consistent with the observation that speech is highly nonstationary. A typical speech signal consists of approximately ten to fifteen phonemes per second and each of these phonemes has varying spectral characteristics [25]. As such, additional information can be obtained to perform the separation. Intuitively, exploiting nonstationarity can be viewed as having the same number of equations as the number of unknowns to be solved. Needless to say, under stationary condition, there is only one equation to solve more than one unknown. This explains the reason why second-order based BSS alone is insufficient to perform the separation. Nonstationarity has been reported to successfully solve the instantaneous mixing model [12, 26, 27].

Consider a convolutive mixture of N sources with L sensors (where $L \geq N$), the observed signal vector $\mathbf{x}(t) = [x_1(t), \ldots, x_L(t)]^T$, at each of the sensors is

$$\mathbf{x}(t) = \sum_{p=0}^{P-1} \mathbf{H}(p)\, \mathbf{s}(t-p) \tag{1}$$

where $\mathbf{s}(t) = [s_1(t), \ldots, s_N(t)]^T$ is the N-source vector, $\mathbf{H}(p)$ is a $L \times N$ mixing matrix, P is the length of the impulse response from the nth source to the lth sensor, and $(\cdot)^T$ denotes the transpose. Mathematically, BSS blindly finds an unmixing matrix, $\mathbf{W}(p)$, which is of dimension $N \times L \times P$ to recover the sources from the observed L mixtures, up to an arbitrary scaling and permutation.

Following the approach in [12], the solution to the joint diagonalization of M covariance matrices can be estimated as

$$\widehat{\mathbf{W}}(\omega) = \arg\min_{\mathbf{W}(\omega)} \sum_{m=0}^{M-1} \|\mathbf{E}(\omega, m)\|_F^2, \tag{2}$$

where $\|\cdot\|_F^2$ is the squared Frobenius norm and the error function is $\mathbf{E}(\omega, m) = \mathbf{W}(\omega)[\mathbf{R}_x(\omega, m)]\mathbf{W}^H(\omega) - \mathbf{\Lambda}_s(\omega, m)$. Here, $\mathbf{\Lambda}_s(\omega, m)$ is the covariance matrix of the sources $\mathbf{s}(t)$ at frequency ω and $(\cdot)^H$ denotes Hermitian transposition.

As explained previously, the estimation of the frequency domain unmixing weights $\mathbf{W}(\omega)$ leads to arbitrary permutation of each frequency bin. This is because successful separation in each frequency bin does not mean that the separated sources are properly aligned and scaled for the reconstruction process. One simple method to solve this problem is to set a constraint on the time-domain filter size of the unmixing weights, F, such that $\mathbf{W}(\tau) = 0, \tau > F \ll \Omega$, where Ω is the number of frequency bins. As shown in [12], the constraint gathers independent frequencies to form a continuity of the spectra. In this way, the permutation problem can be tackled. A side benefit to the above is that the constraints smooth out fluctuations in the weighting because of possible nonconverging bands, which ensures a smooth transition from one band to the other. This reduces or prevents artifacts during the reconstruction process.

2.2. Postprocessor: Adaptive Noise Canceller. BSS as it is algorithmically defined attempts to recover L number of sources given L observations. In the case thus far, we have only one target signal (e.g., in hands-free mobile applications with multiple noise sources). By appropriately choosing the speech dominant BSS output, further decorrelation can be performed via an adaptive noise canceller (ANC). In this case, the kurtosis is used as the outputs discriminator. Thus, BSS output with the highest kurtosis value will be the speech dominant output and the remaining $L-1$ outputs will serve as reference signals for the ANC. Similar to the generalized sidelobe canceller (GSC), the spatiotemporal decorrelator [17] benefits from the addition of more elements since each element provides an additional degree of freedom for the ANC to adapt on, provided that the references are uncorrelated.

The following modified frequency domain leaky LMS algorithm for the frequency ω is used:

$$\begin{aligned}\mathbf{h}(\omega, k+1) &= (1-\beta)\,\mathbf{h}(\omega, k) \\ &\quad + z^*(\omega, k)\,\mathbf{y}_{\text{ref}}(\omega, k)\, f(\omega, k),\end{aligned} \tag{3}$$

where the $(L-1)K \times 1$ stacked reference weights are

$$\mathbf{h}(\omega, k) = \left[\mathbf{h}_1^T(\omega, k), \ldots, \mathbf{h}_{L-1}^T(\omega, k)\right]^T, \tag{4}$$

and

$$\begin{aligned}\mathbf{h}_l(\omega, k) &= \left[h_l(\omega, k), \ldots, h_l(\omega, k-K+2),\right. \\ &\quad \left. h_l(\omega, k-K+1)\right]^T.\end{aligned} \tag{5}$$

Similarly, the $(L-1)K \times 1$ stacked reference signals are

$$\mathbf{y}_{\text{ref}}(\omega, k) = \left[\mathbf{y}_{1,\text{ref}}^T(\omega, k), \ldots, \mathbf{y}_{L-1,\text{ref}}^T(\omega, k)\right]^T, \tag{6}$$

where

$$\begin{aligned}\mathbf{y}_{l,\text{ref}}(\omega, k) &= \left[y_{l,\text{ref}}(\omega, k), \ldots, y_{l,\text{ref}}(\omega, k-K+2),\right. \\ &\quad \left. y_{l,\text{ref}}(\omega, k-K+1)\right]^T.\end{aligned} \tag{7}$$

The nonlinear function $f(\omega, k)$ is given as

$$f(\omega, k) = \frac{\gamma}{K\widehat{\sigma}_z^2(\omega, k) + \gamma \mathbf{y}_{\text{ref}}^H(\omega, k)\,\mathbf{y}_{\text{ref}}(\omega, k)}, \tag{8}$$

and the constants β and γ are the leaky factor and the step size, respectively. The role of the leaky factor is to prevent the adaptive filters from having one or more modes that are undriven and undamped. This usually happens when there is no energy in the subband, i.e., the autocorrelation has zero eigenvalues. Thus, in such an event, the leaky factor will stabilize the filter by forcing those modes to zero. The order of the filter is K and $\widehat{\sigma}_z^2(\omega, k)$ is a time-varying estimate of the output signal power $z(\omega, k)$ that adjusts the step size according to the target signal level. It is built upon the fact that excess MSE increases with both the step size and the target signal [16]. When this happens, the function in (8) will effectively reduce the step size. The output signal power is estimated using the square of vector norm of length K and then exponentially averaged as

$$\widehat{\sigma}_z^2(\omega, k) = (1-\lambda)\,\widehat{\sigma}_z^2(\omega, k-1) + \lambda \|\mathbf{z}(\omega, k)\|^2, \tag{9}$$

where

$$\begin{aligned}&\mathbf{z}(\omega, k) \\ &\quad = \left[z(\omega, k), \ldots, z(\omega, k-Q+2), z(\omega, k-Q+1)\right]^T,\end{aligned} \tag{10}$$

λ is the smoothing parameter, $\|\cdot\|$ denotes the Euclidean norm, and the output of the overall system is given as

$$z(\omega, k) = y_{\text{target}}(\omega, k) - \mathbf{h}^H(\omega, k)\,\mathbf{y}_{\text{ref}}(\omega, k). \tag{11}$$

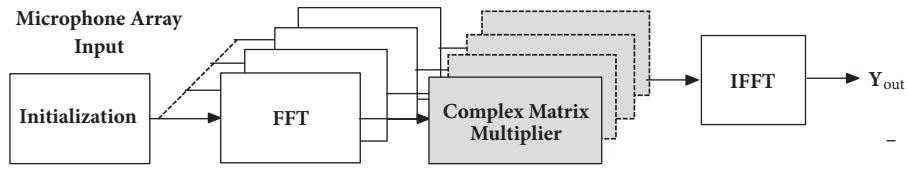


Figure 2: Dataflow of the main operations.

3. Hardware Architecture and Design

As explained in Section 2, the entire algorithm is implemented in the frequency domain. The main dataflow of the proposed system is shown in Figure 2, which performs the following processes:

(1) Transform the input signal to their frequency domain representations via short time FFT;

(2) Filter the frequency transformed signals by the weight estimates from the Complex Matrix Multiplier;

(3) Reconstruct the signal estimates back to the time domain via short time IFFT (inverse FFT).

The architecture makes use of parallelism property of the algorithm via frequency domain. In summary, one can explore implementing parallelism at several levels, including [28]

(i) loop level parallelism, where consecutive loop iterations can be run in parallel,

(ii) task level parallelism, where entire procedures inside the program can be run in parallel,

(iii) data parallelism.

Since the algorithm is made up of a control part and a computation part, the first stage consists in locating the computational kernels of the algorithms. The algorithmic profiling is performed to determine the time consumption of the computation kernels. The profiling exercise can be summarised in Tables 1 and 2. The results show that both the FFT/IFFT and Complex Matrix Multiplier operations are the most computationally expensive, which takes up 98.9% of the CPU time. Thus, FPGA modules will be developed and optimized for these operations. The remaining parts of the code can then be run by software powered by the PowerPC processor supported by Auxiliary Processor Unit (APU). Note that Virtex-4 family FPGAs are suitable to carry out this implementation. It has an Auxiliary Processor Unit (APU) controller which can simplify the integration of hardware accelerators and coprocessors. These hardware accelerator functions behave as extensions to the PowerPC, thereby offloading the CPU from heavy computational tasks.

Figure 3 shows the block diagram of this architecture, which includes two APU channels linked to the FFT/IFFT module and the Complex Matrix Multiplier module simultaneously. In an effort to better synchronize the execution times between the FFT/IFFT operation and the Complex Matrix Multiplier operation, multiple instances of Complex Matrix Multiplier module can be created and aligned with the

Table 1: Profiling results of overall operations.

Function	Time (s)	%Overall Time
Perform BSS	152.1	86.9%
Calculate BSS Output	14.2	8.1%
Post Processing ANC	4.9	2.8%
OTHERS	3.9	2.2%

Table 2: Profiling results of detailed operations.

Operation	%Overall Time
24-bit FFT/IFFT (256 pt)	43.8%
Complex Matrix Multiplier	55.1%
OTHERS	1.1%

computational time. The proposed system makes use of the hardware accelerator to provide a high level of parallelism. These functional units can be operated in parallel across the frequency bins. Indeed, the parallel nature of the frequency domain allows a high degree of parallelism to be implemented. Moreover, it is possible to explore both performance and area optimization to find a trade-off point in the implementation. Furthermore, these functional units are highly scalable and adaptable and are dedicated to implementing elementary arithmetic operations. These units can be inserted or removed from the architecture in an immediate way, just by setting the value of dedicated VHDL generics. This feature allows direct tuning of many key parameters of the architecture, e.g., width of the bus, latency of the functional units, and throughput.

The block diagram of the hardware accelerator is given in Figure 4. The diagram illustrates the two FCB channels connecting to the two accelerators, namely, a FFT/IFFT module responsible for short time frequency transformation and time domain reconstruction and a Complex Matrix Multiplier module used to calculate the gradient of the error function and updating the unmixing frequency domain weights. The details of the APU instruction execution and data flow in the FPGA architecture are described in [28]. Briefly, once the APU passes an instruction to the hardware accelerator, the instruction will be decoded by the decoder logic. The instruction will then be executed on the data from the memory. Following the decoding of the instruction, the state machines handle data transfers between the APU and the operations module by using registers to store data for the subsequent operations. The data from memory is transferred between the APU and the interfacing logic via the *load* and *store* instructions.

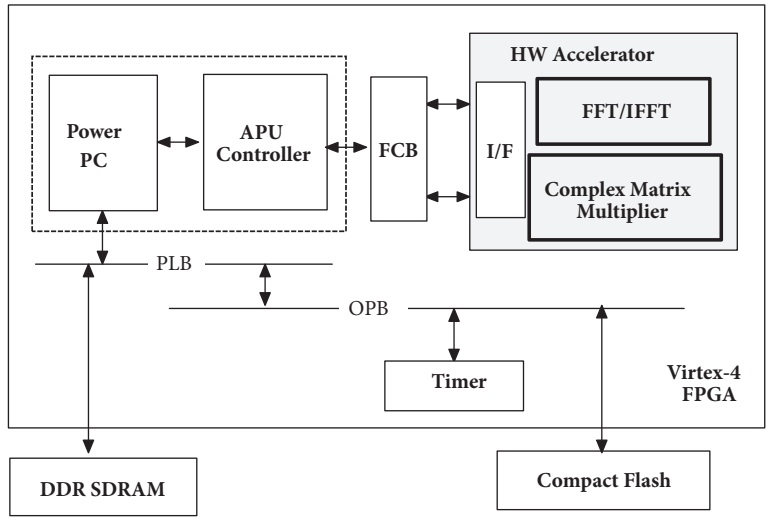


Figure 3: Block diagram of the proposed FPGA architecture for BSS.

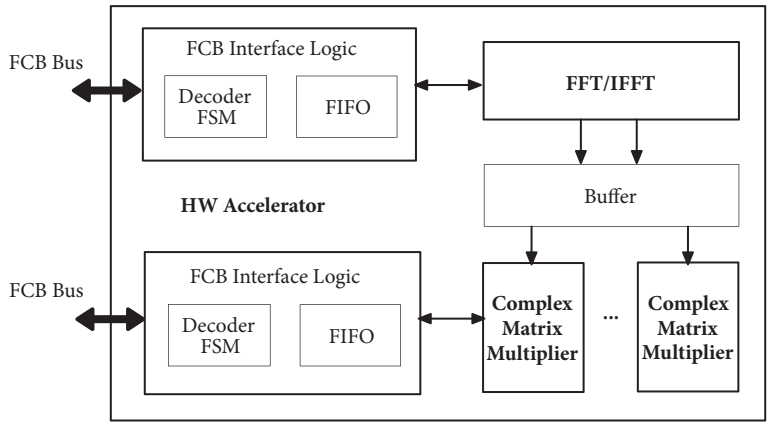


Figure 4: Block diagram of the hardware accelerator.

While the FFT/IFFT can be implemented by using the core generator LogiCORE IP FFT provided by the vendor tools [29], the Complex Matrix Multiplier module must be designed to maximize the system performance. The block diagram of the implementation is depicted in Figure 5. As the adaptation is a data-oriented application, the dependency and movement of the data are important in order to ensure that the update function can be implemented in a time-multiplexed fashion, and the scheduling of the operations can be done sequentially to achieve the desired trade-off between performance and circuit complexity.

The data flow in the proposed architecture can be explained by the following [28]:

(1) In the first instance, the filtering process starts with the processor forwarding a load instruction for filter input data to the APU;

(2) The instruction set is passed to the interface logic via the APU, which decodes the instruction and waits for data from memory to arrive;

(3) The input data is sent to the interface logic;

(4) The processor forwards the store instruction to the APU in anticipation of the filter output once the load instructions are completed;

(5) The interface logic decodes the store instruction and waits for data from the filter module;

(6) Once processing is performed, the operation module then returns results to the interface logic;

(7) The interface logic returns the output data to the processor via the APU and they are written to memory.

Figure 6 presents the main state machine that is responsible for the *load* and *store* operations. The hardware state machine manages the data transfer by sending and receiving the data from software to hardware or vice versa. This state machine communicates with the processor via the APU.

When the hardware state machine states a ready flag, it means that it is ready to accept the data. The role of PowerPC is to provide the data and address it. It will also issue a valid signal, which provides the indication to write. Once the hardware obtains the valid signal, it then writes the data at

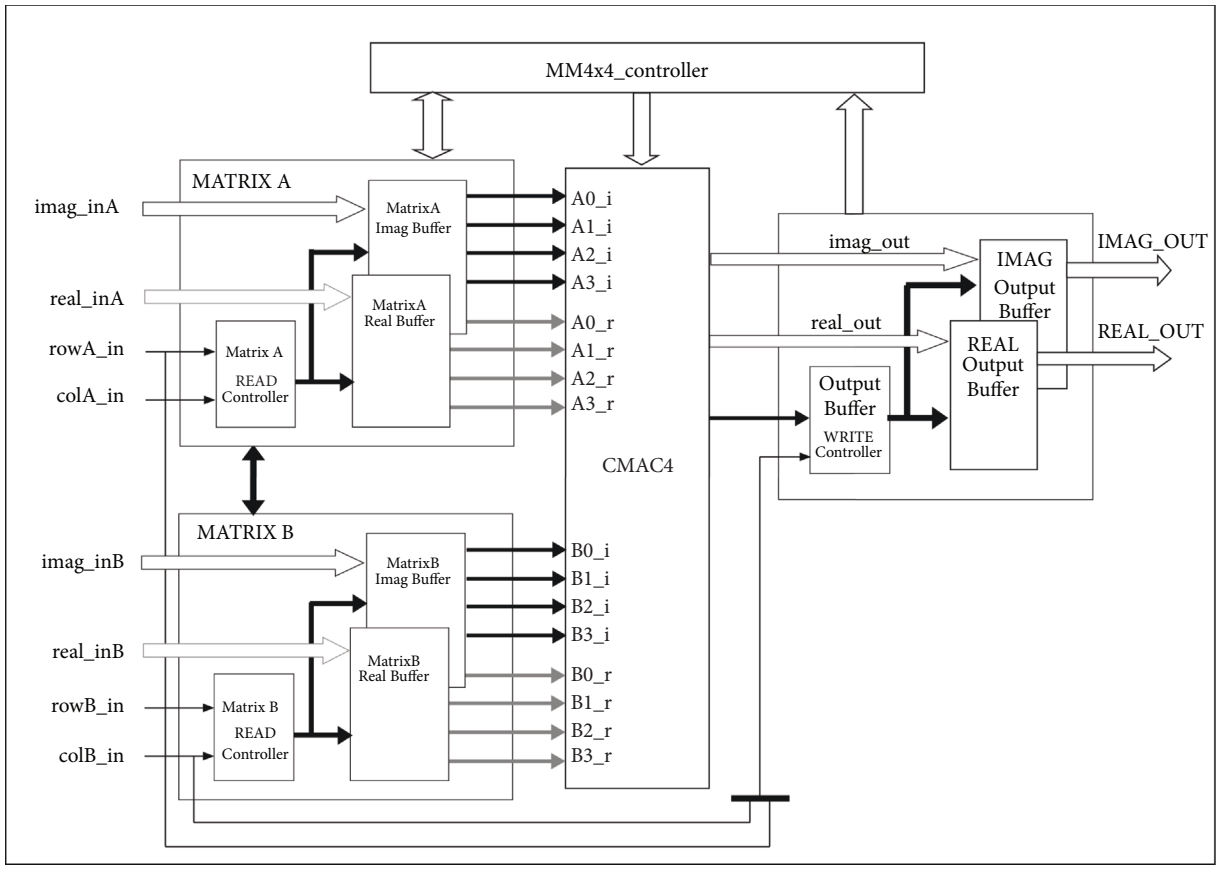


Figure 5: Block diagram of the Complex Matrix Multiplier module.

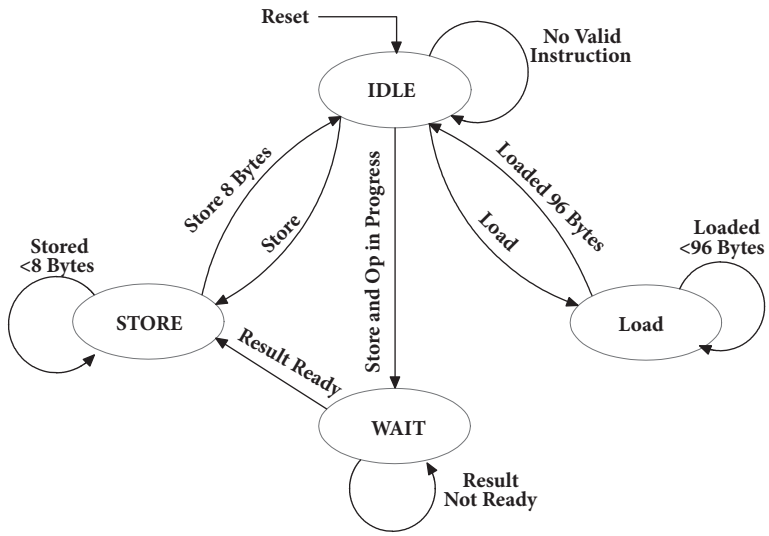
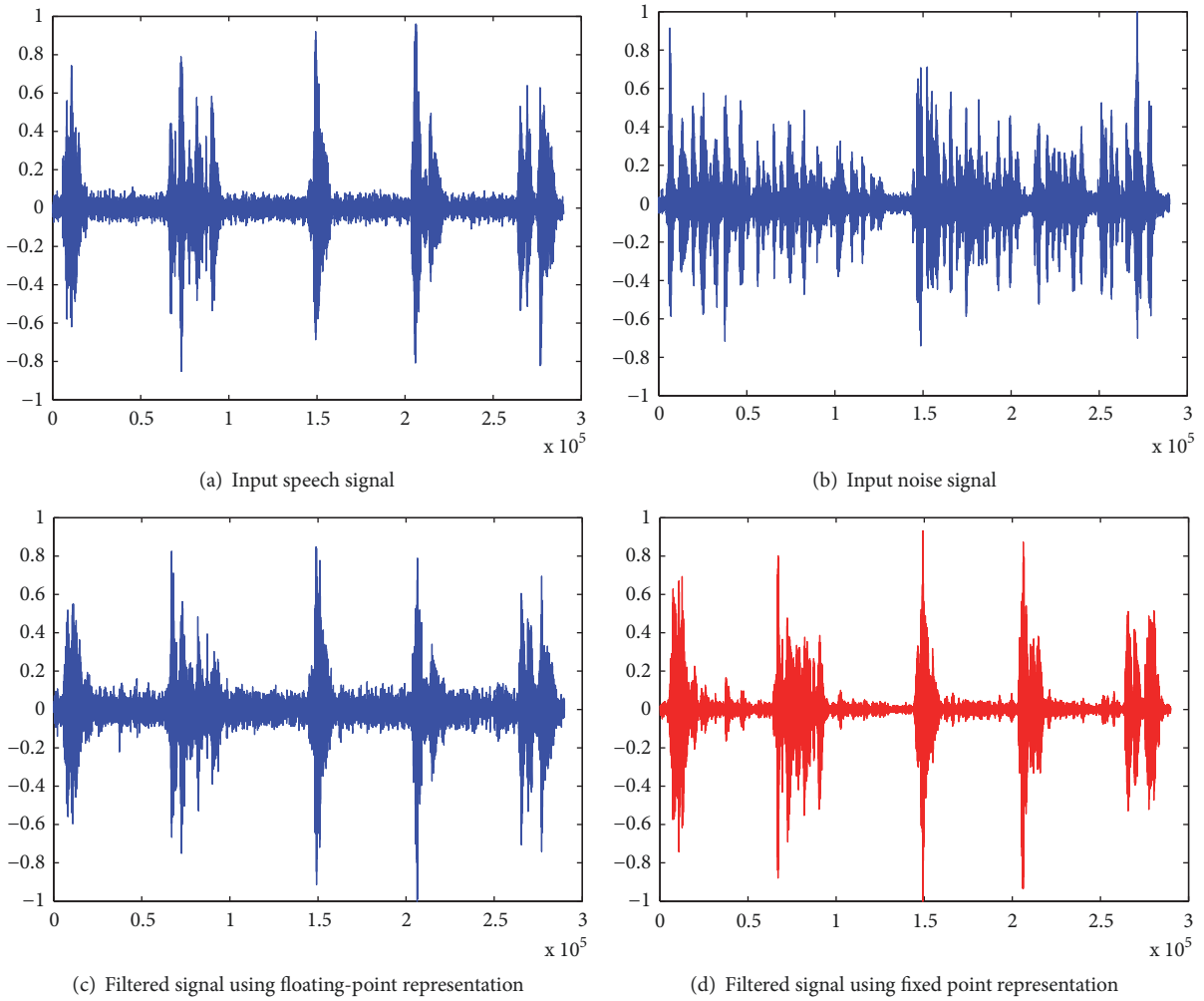


Figure 6: The data flow of the main state machine.

the address provided by the PowerPC. Once completed, the hardware asserts a flag which informs the PowerPC that data has been written. This triggers a wait state whereby the system awaits for the result. Once the result is ready, it then asserts a result ready flag. Interestingly, the PowerPC can detect the completion in two ways [28]. In the default state, the processor can continuously poll a bit in order to detect when the calculation has completed. The second approach works by enabling an interrupt. Upon the completion of the process, the interrupt output signal will be asserted. The interrupt approach helps to save valuable time by avoiding the polling loops to perform other calculations. This is especially the case while the hardware accelerator is updating the weights of BSS.



(a) Input speech signal

(b) Input noise signal

(c) Filtered signal using floating-point representation

(d) Filtered signal using fixed point representation

Figure 7: Input signal and results of BSS.

4. Results

The performance evaluation of the proposed hardware architecture was simulated in the Xilinx XC4VSX55-12-FF1148 chip. The settings for the experiment were as follows:

(i) Input sequences were English speech from male/female with four microphones;

(ii) Sampling frequency was 16 kHz;

(iii) Prototype filter length in the analysis and synthesis filter banks was 128;

(iv) Number of taps in the adaptive filters was 5;

(v) Parameter α was 0.15, regulator for BSS was 1, and order for the temporal weights (ANC) was 3;

(vi) Number of subbands was chosen as $M = 256$ unless otherwise stated;

(vii) Number of iterations for the BSS was 2000.

The first task is to figure out a suitable bitwidth for the fixed point arithmetic. The input speech and noise signals are displayed in Figures 7(a) and 7(b), respectively. By adjusting the bitwidths, it turns out that the appropriate integer size is 12 (fraction size is 20). Further increase in the size does not improve the results significantly. The BSS filtered output using a 32-bit fixed-floating point arithmetic is compared with the pure floating point implemenation as shown in Figures 7(c) and 7(d). Using 32-bit fixed-float architecture, very similar results are achieved, showing that partial fixed point operations have not affected the accuracy of the calculations much. In practice it may be possible to overflow occasionally even if the integer size is 12, so saturation arithmetic has been employed in the hardware design to minimise the impact.

Table 3: Implementation results of BSS.

FPGA device	XC4VSX55-12	XC2VP30-7
Slices used	5937 (12%)	8916(32%)
DSP48/MULT used	72 (14%)	72 (52%)
Block RAM used	8 (2%)	8 (5%)
Frequency (MHz)	184.8	165.9

Table 3 represents the implementation results of the proposed hardware design for BSS on both Xilinx XC4VSX55-12-FF1148 and Xilinx XC2VP30-7-FF896 FPGA devices. Note

Table 4: Maximum speedup with multiple instances in the FPGA device.

Samples/s	Number of Instances		Slices	DSP
	FFT / IFFT	Complex Matrix Multiplier	Used	Used
3057.1	1	1	24%	14%
5434.8	1	2	37%	24%
8152.3	1	3	50%	33%
11039.5	1	4	63%	42%
14096.6	1	5	73%	52%
17323.6	1	6	86%	61%
5265.0	2	1	35%	18%
8661.8	2	2	48%	27%
12907.8	2	3	61%	35%
17663.2	2	4	74%	46%
22758.4	2	5	87%	53%
5434.8	3	1	46%	22%
9341.1	3	2	59%	31%
14436.3	3	3	72%	40%
20380.7	3	4	85%	49%
5604.7	4	1	57%	26%
10020.5	4	2	70%	34%
15795.0	4	3	83%	43%

that DSP48 is a coarse-grained DSP embedded block in Virtex-4 Series FPGA for multiplication while MULT is a block multiplier in Virtex-2 Pro Series FPGA.

In order to estimate the performance of the proposed FPGA-based BSS system, we first incorporate one instance of FFT/IFFT and one instance of Complex Matrix Multiplier hardware accelerators. Taking one data block with 256 samples, assuming the sampling rate is 16kHz, the number of clock cycles required for processing the block of data in the frequency domain is measured as 15,421,718. Therefore, given that the period of one clock cycle is $1/(184MHz) = 5.43ns$ on a Virtex-4 FPGA, the FPGA-based BSS can perform one step of speech enhancement in $0.0837s$ or equivalently 3057.1 samples per second.

Table 4 summarises the implementation results when adding more instances of the filter in an XC4VSX55-12-FF1148 FPGA chip. It shows how the number of instances affects the speedup. With different slices and DSPs being deployed, maximum frequency and speedup vary when multiple instances are implemented on an XC4VSX55-12-FF1148 FPGA device. A XC4VSX55-12-FF1148 chip can accommodate at most two FFT/IFFT and five Complex Matrix Multiplier hardware accelerators, so the maximum sampling rate will be 22758.4 samples per second. Consequently, it can indeed achieve real-time performance.

5. Conclusions

In this paper, an online blind signal separation system has been proposed, which involves designing the separation matrix and a postfiltering noise canceller. A hardware implementation of the algorithms on an FPGA virtex-4 system has been described. In the algorithm, in order to achieve computational efficiency, a frequency domain implementation is employed to speed up the convergence of the beamformers. The complete architecture is simulated in hardware and results show that real-time performance can be achieved when an FPGA-based hardware accelerator performs the critical parts of the algorithm. The resulting embedded system will find applications in modern multimedia systems. As a future extension, it would be of interest to investigate power consumption of the final design based on the technique in [30].

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This paper is supported by RGC Grant PolyU 152200/14E and PolyU Grant 4-ZZGS and G-YBVQ. The authors would like to thank Mr. Xiaoxiang Shi for carrying out the implementation on FPGA.

References

[1] "Microphone Arrays: Signal Processing Techniques and Applications," in *Digital Signal Processing*, Springer-Verlag, Berlin, 2001.

[2] J. Benesty, S. Makino, and J. Chen, "Speech Enhancement," in *Signals and Communication Technology*, Springer-Verlag, Berlin, March 2005.

[3] P. Loizou, *Speech Enhancement: Theory and Practice*, CRC Press, Taylor and Francis, Boca Raton, Florida, USA, 2007.

[4] R. Martin, "Noise power spectral density estimation based on optimal smoothing and minimum statistics," *IEEE Transactions on Speech and Audio Processing*, vol. 9, no. 5, pp. 504–512, 2001.

[5] R. Miyazaki, H. Saruwatari, T. Inoue, Y. Takahashi, K. Shikano, and K. Kondo, "Musical-noise-free speech enhancement based on optimized iterative spectral subtraction," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 20, no. 7, pp. 2080–2094, 2012.

[6] L. J. Griffiths and C. W. Jim, "An alternative approach to linearly constrained adaptive beamforming," *IEEE Transactions on Antennas and Propagation*, vol. 30, no. 1, pp. 27–34, 1982.

[7] I. Claesson and S. Nordholm, "A Spatial Filtering Approach to Robust Adaptive beamforming," *IEEE Transactions on Antennas and Propagation*, vol. 40, no. 9, pp. 1093–1096, 1992.

[8] S. Y. Low, N. Grbić, and S. Nordholm, "Speech enhancement using multiple soft constrained subband beamformers and non-coherent technique," in *Proceedings of the 2003 IEEE International Conference on Accoustics, Speech, and Signal Processing*, vol. 5, pp. 489–492, Hong Kong, April 2003.

[9] H. Q. Dam, S. Y. Low, S. Nordholm, and H. H. Dam, "Adaptive microphone array with noise statistics updates," in *Proceedings of the 2004 IEEE International Symposium on Cirquits and Systems*, vol. 3, pp. 433–436, Canada, 2004.

[10] S. Y. Low, N. Grbic, and S. Nordholm, "Robust microphone array using subband adaptive beamformer and spectral subtraction," in *Proceedings of the 8th IEEE International Conference on Communications Systems, ICCS 2002*, pp. 1020–1024, Singapore, 2002.

[11] J.-F. Cardoso, "Blind signal separation: statistical principles," *Proceedings of the IEEE*, vol. 86, no. 10, pp. 2009–2025, 1998.

[12] L. Parra and C. Spence, "Convolutive blind separation of non-stationary sources," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 8, no. 3, pp. 320–327, 2000.

[13] Z. Lv, B.-B. Zhang, X.-P. Wu, C. Zhang, and B.-Y. Zhou, "A permutation algorithm based on dynamic time warping in speech frequency-domain blind source separation," *Speech Communication*, vol. 92, pp. 132–141, 2017.

[14] S. Gannot, E. Vincent, S. Markovich-Golan, and A. Ozerov, "A Consolidated Perspective on Multimicrophone Speech Enhancement and Source Separation," *IEEE/ACM Transactions on Audio Speech and Language Processing*, vol. 25, no. 4, pp. 692–730, 2017.

[15] S. Araki, R. Mukai, S. Makino, T. Nishikawa, and H. Saruwatari, "The fundamental limitation of frequency domain blind source separation for convolutive mixtures of speech," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 11, no. 2, pp. 109–116, 2003.

[16] S. Y. Low, S. Nordholm, and R. Togneri, "Convolutive blind signal separation with post-processing," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 12, no. 5, pp. 539–548, 2004.

[17] S. Y. Low and S. Nordholm, "A Blind Approach to Joint Noise and Acoustic Echo Cancellation," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, (ICASSP '05)*, vol. 3, pp. 69–72, Philadelphia, Pennsylvania, USA, 2005.

[18] J. Y. Mori and M. Hubner, "Multi-level parallelism analysis and system-level simulation for many-core Vision processor design," in *Proceedings of the 2016 5th Mediterranean Conference on Embedded Computing (MECO)*, pp. 90–95, Bar, Montenegro, June 2016.

[19] G. Zhong, A. Prakash, S. Wang, Y. Liang, T. Mitra, and S. Niar, "Design Space exploration of FPGA-based accelerators with multi-level parallelism," in *Proceedings of the 20th Design, Automation and Test in Europe Conference and Exhibition, DATE 2017*, pp. 1141–1146, Switzerland, March 2017.

[20] P. Graham and B. Nelson, "FPGA-based sonar processing," in *Proceedings of the 1998 ACM/SIGDA 6th International Symposium on Field Programmable Gate Arrays, FPGA*, pp. 201–208, February 1998.

[21] B. G. Tomov and J. A. Jensen, "A new architecture for a single-chip multi-channel beamformer based on a standard FPGA," in *Proceedings of the 2001 Ultrasonics Symposium*, pp. 1529–1533, USA, October 2001.

[22] M. D. Van De Burgwal, K. C. Rovers, K. C. H. Blom, A. B. J. Kokkeler, and G. J. M. Smit, "Adaptive beamforming using the reconfigurable MONTIUM TP," in *Proceedings of the 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools, DSD 2010*, pp. 301–308, France, September 2010.

[23] D. Theodoropoulos and G. Kuzmanov, "A reconfigurable beamformer for audio applications," in *Proceedings of the 2009 IEEE 7th Symposium on Application Specific Processors, SASP 2009*, pp. 80–87, USA, July 2009.

[24] N. Dubey and R. Mehra, "Blind Audio Source Separation in Time Domain using ICA Decomposition," *International Journal of Computer Applications*, vol. 132, no. 6, pp. 48–53, 2015.

[25] P. K. Ghosh, A. Tsiartas, and S. Narayanan, "Robust voice activity detection using long-term signal variability," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 19, no. 3, pp. 600–613, 2011.

[26] J. Li, H. Zhang, M. Fan, and J. Zhang, "Non-stationary sources separation based on maximum likelihood criterion using source temporal–spatial model," *Neurocomputing*, vol. 275, pp. 341–349, 2018.

[27] J. Yin, Z. Liu, Y. Jin, D. Peng, and J. Kang, "Blind Source Separation and Identification for Speech Signals," in *Proceedings of the 2017 International Conference on Sensing, Diagnostics, Prognostics and Control (SDPC)*, pp. 398–402, Shanghai, August 2017.

[28] K. F. C. Yiu, Z. Li, S. Y. Low, and S. Nordholm, "FPGA multi-filter system for speech enhancement via multi-criteria optimization," *Applied Soft Computing*, vol. 21, pp. 533–541, 2014.

[29] Inc. Xilinx, "Fast Fourier Transform product specification," LogiCORE IP Product Guide, DS260, 2011.

[30] W. Tang, C. H. Ho, C. Sham, and K. F. C. Yiu, "Low-power reconfigurable acceleration of robust frequency-domain echo cancellation on FPGA," in *Proceedings of the 2010 International Conference on Green Circuits and Systems (ICGCS)*, pp. 361–364, Shanghai, China, June 2010.