THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

IEICE ICDV 2014

An FPGA-based Variable-length 4-Channel Separation FastICA Implementation

*Dinh-Thien Vu, †Trong-Thuc Hoang, ‡Ngoc-Hung Nguyen, and §Trong-Tu Bui
Digital Signal Processing and Embedded System Laboratory (DESLab)
Faculty of Electronics and Telecommunications (FETEL)
The University of Science, Ho Chi Minh City (VNU-HCMUS)
227 Nguyen Van Cu St., Ho Chi Minh City, Vietnam
Email: *dinhthien.fetel@gmail.com, †‡§{htthuc, nnhung, bttu}@fetel.hcmus.edu.vn

***Abstract*—Independent Component Analysis (ICA) is one of the most popular and powerful tool that has been used widely in the field of signal processing. Due to its complexity, implementing ICA became a challenge for designers. In this paper, authors proposed an FPGA-based ICA implementation using FastICA algorithm. The design can process 4 audio channels with variable length from 2^9 to 2^{25} samples. The proposed implementation achieves the speed of 11.27 Mbps and can process over 1.4 million samples per second.**

I. Introduction

In the field of signal processing research area, the Independent Component Analysis (ICA) is one of the most popular and powerful technique. ICA algorithm and its implementations have been developed for over half of a century, and yet it still drawn attention from many researchers. ICA is the common method to solve the problem of Blind Source Separation (BSS) [1]. The principle of the ICA algorithm is the decorrelation the signals that are of second-order statistics using a minimum of a priori information. Furthermore, ICA can reduce higher order statistical dependencies between reconstructed signals. Because of this, ICA becomes very effective for other applications beside of BSS problem, such as speech [2], image [3], and biomedical [4]. To conclude, ICA algorithm is best suited for unsupervised sources separation while has only the observation mixed signals.

According to [5], ICA algorithm has many modification models. The original model is called Standard ICA (sICA). There is the Convolutive ICA (fICA) model which is a sICA with FIR filters. The fICA approaches were applied for biomedical blind sources separation [6]. However, both sICA and fICA have the same issue that they cannot use the a prior information regarding the shape of the signals. The temporally constrained ICA (cICA) [7] gives the solution to overcome such an issue. The cICA method constrains the temporal shape of the desired and useful components. Therefore, it can bring the prior information into the extracting process. JADE (Joint Approximate Diagonalization of Eigen-matrices) ICA [8] is another popular ICA method. The primary advantage of the JADE approach is that it can perform very effectively on a small number of observations input signals. Informax (information maximization) ICA has been developed by A. J. Bell and T. J. Sejnowski [9] and its extended modification is given by T-W. Lee et al. [10]. Infomax approach based on the maximization of entropy in a single-layer feed-forward neural network, it can be classified as an unsupervised learning algorithm. The infomax algorithm is best for separating the super-gaussian distributions sources: "sharply peak probability density functions with heavy tails" [10]. However, the drawback of infomax is that it cannot separate negative kurtosis, uniform distribution, sources. Generally, the infomax ICA has the small range of sources separations. The extend version of it [10] has been developed for wider range of applications while maintaining the simplicity. Among presented ICA methods, FastICA approach is the hardware-friendly algorithm which first introduced by A. Hyvärinen and E. Oja [11]. It is an approximation algorithm of standard ICA with fixed-point iterations to minimize the error. FastICA method achieves 10 to 100 times faster than conventional methods of ICA. Therefore, it becomes the most successful linear ICA algorithm due to its strong advantages of easy to implement and fast convergence.

Although the effectiveness of ICA has been verified by many researchers, the software solutions cannot satisfy the real-time requirement due to the complexity of the algorithm. However, the hardware approaches have to use approximation models leads to less accuracy in the comparison with the standard model. As a result, ICA implementations became a challenge for hardware designers throughout decades. There are many VLSI implementations have been done [12]–[14]. According to the comparative study of Hongtao Du et al. [15], the VLSI solutions of ICA algorithms require extremely efficient hardware design and sufficient IC resources. There are many techniques and technologies have been used such as analog CMOS, Analog-Digital mixed signal, ASICs, and FPGAs. "Each technology has its own characteristics, and none of them can balance between a high-density low-cost design and a shorter turnaround development period" [15]. However, new development in FPGAs design methodology is a promising approach as claimed in [4] and [15].

In this paper, the authors present an FPGA-based implementation using FastICA algorithm. The proposed system can separate 4 audio channels with variable length from 2^9 to 2^{25} samples. The implementation can perform at maximum frequency of 62 MHz. It can process over 1.4 million samples per second. With 8-bit audio data, the design achieves the speed of 11.27 Mbps.

The remainder of this paper is organized as follows. Section II briefly reviews the FastICA algorithm. Section III proposes the variable-length 4-channel FPGA implementation. Section IV presents the experimental results. And finally, Section V gives the conclusion of the research.

ISBN: 978-4-88552-294-9

II. Background Algorithms

A. Independent Component Analysis

ICA algorithm can be defied by the statistical variables model. There are n random resources variables $s_1, ..., s_n$ that made n random observations variables $x_1, ..., x_n$. Then, it is a linear combination in the form as can be seen in Eq. 1.

$$x_i = a_{i1}s_1 + a_{i2}s_2 + ... + a_{in}s_n \quad (i = 1, 2, ..., n) \tag{1}$$

where, $a_{ij}(i, j = 1...n)$ are real coefficients. By definition, the s_i is statistically independent of each other. Thus, the goal of ICA algorithm is solving the equation of $\boldsymbol{x = As}$. The ICA method can be done only when the following constraints are satisfied:

- The original sources signals are statistically independent with each other.
- Mixing matrix A is a square matrix (source signal and mixed signal equal) and able to inverse.
- Maximum of only one original source signal has Gaussian distribution.

ICA algorithm performs a linear transformation $\boldsymbol{y = Wx}$. As a consequence, the components $\boldsymbol{y_i}$, with $\boldsymbol{i = 1, N}$, are possible mutual independence by maximizing the functions measuring mutual independence $\boldsymbol{Fy_1....y_N}$ ($\boldsymbol{y_N}$ is the recovered signal).

B. FastICA

FastICA method was developed and first introduced by A.Hyvärinen and E. Oja [11] in 1997. The algorithm aims to reduce the complexity of the origin method by using fixed-point approach and iteration equations. FastICA has been proved that it is a hardware-friendly algorithm. FastICA is used for calculating the non-Gaussian measure of mutual independence. There are three main steps in FastICA method: Centering, Whitening, and ICA estimation.

1) Centering: The most basic and necessary preprocessing is to center data $\boldsymbol{x}$. It can be done by subtracting mean vector $\boldsymbol{E\{x\}}$ in order to make data $\boldsymbol{x}$ a zero-mean variable, as shown in Eq. 2.

$$x_{new} = x - E\{x\} \tag{2}$$

Vector $\boldsymbol{x}$ is called centering when it has zero-mean. The reason for centering is that the real signal always has noises, and the most common noise it the white noise with Gaussian distribution. Centering is how to eliminate white noise as well as help the separation process becomes simpler in general.

2) Whitening: Whitening $\boldsymbol{x}$-vector is based on uncorrelated and covariance $\boldsymbol{x}$ matrix which is the identity matrix of centered $\boldsymbol{x}$-vector with zero-mean. Whitening is a process that transforming the mixing matrix $\boldsymbol{A}$ to orthogonal by multiplying $\boldsymbol{V}$ matrix with $\boldsymbol{x}$-vector data as seen in Eq. 3.

$$z = Vx \tag{3}$$

where $\boldsymbol{V}$ whitening matrix is calculated by Eigen Value Decomposition (EVD) of the covariance matrix. Eq. 4 gives the EVD computation.

$$E\{xx^T\} = EDE^T \tag{4}$$

where, $\boldsymbol{E}$ is the orthogonal matrix of eigenvectors of $\boldsymbol{E\{xx^T\}}$, and $\boldsymbol{D}$ is the diagonal matrix of its eigenvalues, $\boldsymbol{D = diag(d1, .., dn)}$. Because of $\boldsymbol{A}$ matrix is orthogonal, $\boldsymbol{A^{-1} = VA}$ is also orthogonal. Then, whitening is considered to solve half of the ICA computation based on $\boldsymbol{W}$ matrix approximation on the orthogonal space

3) ICA Estimation: To use non-gaussianity in ICA estimation, we must have a quantitative measure of non-gaussianity of a random variable, $\boldsymbol{y}$. There are two well-known aprroximation methods, negentropy and kurtosis.

Approximating Negentropy: To measure of non-gaussianity that is zero for a gaussian variable and always non-negative, one often uses a definition of differential entropy, called negentropy. Negentropy $\boldsymbol{J}$ is defined as in Eq. 5.

$$J(y) = H(y_{gauss}) - H(y) \tag{5}$$

where, $\boldsymbol{y_{gauss}}$ is a Gaussian random variable of the same covariance matrix as $\boldsymbol{y}$. As mentioned above, negentropy has properties that alway non-negative, and it is zero if and only if $\boldsymbol{y}$ has a Gaussian distribution. The negentropy estimation is hard to compute. Therefore, it can be approximated by the contrast function $\boldsymbol{G_i}$ as shown in Eq. 6.

$$J(y) = \sum_{i=1}^{p}[E\{G_i(y)\} - E\{G_i(v)\}]^2 \tag{6}$$

The $\boldsymbol{G}$ value must be chosen to not growing too fast. The following guide as shown in Eq. 7 helps for choosing $\boldsymbol{G}$.

$$\begin{aligned} G_1 &= \tfrac{1}{a_1} log(cosh(a_1 u)) \\ G_2 &= -exp(-\tfrac{u^2}{2}) \\ G_3 &= \tfrac{y^4}{4} \end{aligned} \tag{7}$$

where, $1 \leq a_1 \leq 2$ are suitable constants.

Approximating Kurtosis The original measurement of non-gaussianity is kurtosis or the fourth-order accumulation. The kurtosis of $\boldsymbol{y}$ is defined by Eq. 8.

$$Kurt(y) = E\{y^4\} - 3(E\{y^2\})^2 \tag{8}$$

The $\boldsymbol{y}$ compoent in Eq. 8 is assumed the unit variance, then the right-hand side is simplified to $\boldsymbol{E\{y^4\} - 3}$. As a result, the kurtosis is simply a normalized version of the fourth-moment $\boldsymbol{E\{y^4\}}$. For a gaussian distribution of $\boldsymbol{y}$, the fourth-moment equals to $\boldsymbol{3(E\{y^2\})^2}$. Thus, kurtosis is zero for a gaussian random variables. For most (but not all) non-gaussian random variables, kurtosis is non-zero.

ISBN: 978-4-88552-294-9

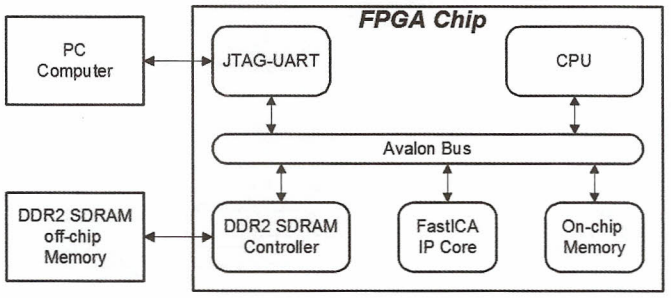

Fig. 1: FPGA system overview.

III. Proposed Implementation

A. FPGA System

The implementation is built by Verilog HDL code. The ModelSim is used for verify the funtionality, then Quartus is deployed to synthesize the circuit. The system is built on Altera Stratix IV with EP4SGX230KF40C2 FPGA chip.

Fig. 1 gives the overview of the system. The system is made for testing the FastICA IP Core. Before the start of the IP Core, four different data of observation sources must be presented first. To do that, the CPU in FPGA co-operates with the PC Computer to transfer the data to DDR2 SDRAM by using the JTAG-UART communication. The on-chip memory is used for storing the program code of CPU. After the transfer of sources data is completed, the CPU starts the process of the FastICA IP Core. Then, the IP Core aceeses to the DDR2 SDRAM to read the observation sources data. It computes the FastICA algorithm, then write back the result data to the SDRAM when the process is completed. After that, the CPU uses the JTAG-UART to write the result data back to the PC Computer.

B. FastICA IP Core

The block diagram of the FastICA IP Core is shown in Fig. 2. As can be seen in the figure, the process of the IP Core can be divided into six major steps as follows: Centering, Covariance, EVD, Whitening, ICA Estimation, and Compute Result. There are two DMAs, i.e. Master Read and Master Write, that help to communicate with the Avalon Bus, and two fifos are used for transferring the data in and out of the Core.

First of all, the Centering module reads the data by the Master Read to compute the mean value and centers the whole data. The output centered-data are both writen back to RAM by the Master Write for later use and go directly to the Covariance module in order to compute the covariance matrix. The EVD module receives the covariance matrix and calculates the eigen vectors and eigen values. Then, the eigen vectors and eigen values are transferred to the Whitening module. The Whitening module uses the information from EVD module to whiten the centered-data, then writes the whiten-data to the RAM by the Master Write. The Whitening module also gives the whiten matrix to the ICA Estimation module. the ICA Estimation module needs the whiten matrix along with whiten data to compute the W matrix. Finally, the Compute Result module reads the centered-data from the RAM to multiply with the W matrix from the ICA Estimation module. The result of that multiplication is also the result data that are writen back to RAM.

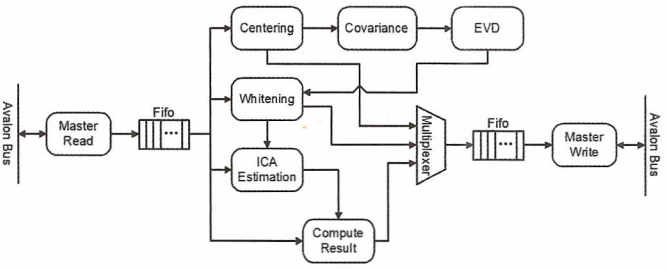

Fig. 2: FastICA IP core block diagram.

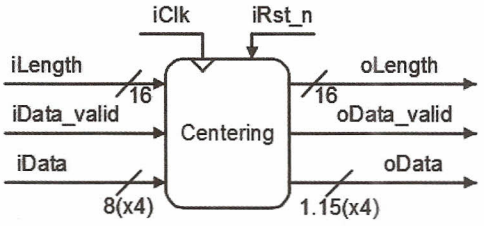

Fig. 3: Top-view of Centering module.

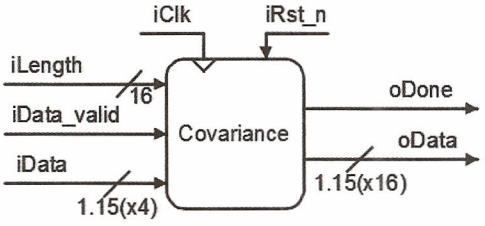

Fig. 4: Top-view of Covariance module.

1) Centering: The top-view of the Centering module is shown in Fig. 3. The inputs are 4-channel audio data as can be seen in the figure as the width 8(x4) of $iData$. The 8-bit audio are unsigned numbers. After centering, they become 8-bit singed numbers within ranged of -127 to $+128$. By not effect on the final result, the data are assumed to fall in the ranged of -1 to $+1$. Then, 8-bit signed numbers become 1.7-bit fixed-point signed numbers. Then, the mean value is a 1.7-bit fixed-point signed number, too. However, in order to increase the accuracy, the mean value and centered-data take more 8 bits behind the dot. Then, they become 1.15-bit fixed-point signed numbers as can be seen in Fig. 3. The $Length$ signals are used for giving the total number of input samples.

2) Covariance: Fig. 4 gives the top-view of the Covariance module. The module receives the centered-data directly from the Centering module. It needs the $iLength$ signal to known the total number of samples in order to compute the covariance matrix. With 4-channel, the covariance matrix is a $4*4$ matrix. Then, it has 16 numbers in total which are transferred to the next module by the $oData$ signals. The $oDone$ signal triggers the process of the next module.

3) EVD: The top-view of the EVD module is given by Fig. 5. The process of EVD module is to compute the eigen vectors and eigen values. The module receives covariance matrix from the iEn_data and $iData$ signals. When the process is completed, $oDone$ signal is activated. With 4-channle, there are 4 eigen vectors which make the E matrix, and 4 eigen values which make the D matrix. The $oMatrix\_E$ and $oMatrix\_D$ give the E and D matrices, respectively.

ISBN: 978-4-88552-294-9

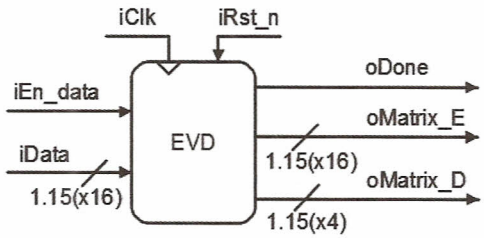

Fig. 5: Top-view of EVD module.

The block diagram of the EVD module is shown in Fig. 6. The EVD module is based on the Jacobi eigen value algorithm. It requires a iteration equations to achieve the goal. Then, the Controller is used for control the iteration process. CS Eigen module computes the values for each iteration. During the process, a Matrix Multiplication Unit (MMU) is needed to multiply matrices. MMU is controlled by the CS Eigen module, and it is also used for store the result for each step.

4) Whitening: Fig. 7 shows the top-view of the Whitening module. It uses the avalon bus signals to communicate with DMAs in order to read the centered-data and write the whiten-data. The whiten-data are stored in the SDRAM at a different offset with the centered-data. The oEn_matrix signal is asserted when the process is done. The oWh_matrix gives the whiten matrix to the next module. The whiten matrix is a $4 * 4$ matrix with 1.15-bit fixed-point signed numbers.

5) ICA Estimation: As mentioned above, there are two mainstreams approaches for the estimation process: negentropy and kurtosis. However, kurtosis has been proved that it is suitabled for hardware designs in the comparison with negentropy approaches. In kurtosis method, there are many iteration equations that could satisfy the requirement of the algorithm. They are pow3, tanh, gauss, and skew as can be seen in Eq. 9, Eq. 10, Eq. 11, and Eq. 12, respectively.

$$w = \frac{1}{N} X (X^T w)^3 - 3B \tag{9}$$

$$\begin{aligned} hypTan = &\ tanh(a_1 X^T w) \\ w = &\ \tfrac{1}{N}(X hypTan - a_1 \textstyle\sum (1 - hypTan^2)^T w) \end{aligned} \tag{10}$$

$$w = \frac{1}{N}(X gauss - \sum (dGauss)^T w) \tag{11}$$

$$w = \frac{1}{N} X (X^T w)^2 \tag{12}$$

In the aboved equations of kurtosis method, N is the number of samples, w is the decomposition matrix which also is the goal of the algorithm, X is the data after centering and whitening, B and a_1 and $gauss$ are the parameters.

The $tanh$, Eq. 10, and $gauss$, Eq. 11, equations have the high complexity that leads to sufficient resouces cost. The $skew$ equation, Eq/ 12 is the simplest of all. However, it has been verified that cannot achieve the requirement of accuracy. As a result, the $pow3$ equation, Eq. 9 is chosen to be implemented.

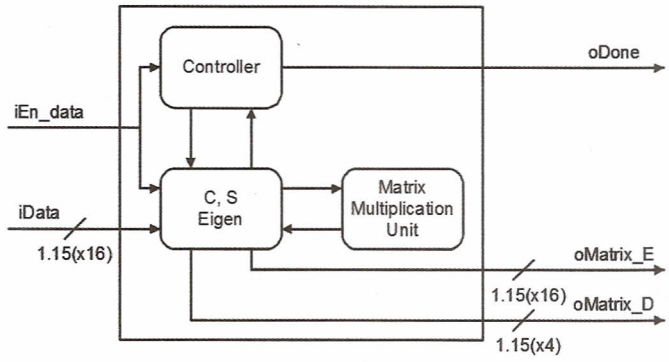

Fig. 6: EVD Block Diagram.

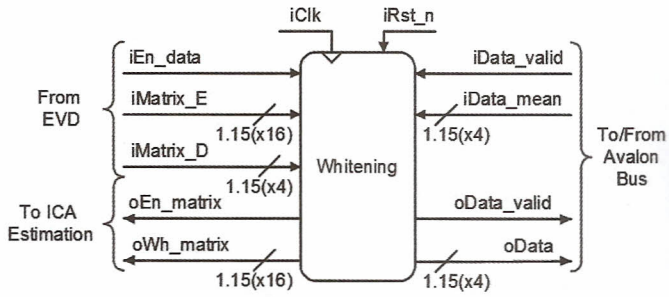

Fig. 7: Top-view of Whitening module.

Fig. 8 shows the top-view of the ICA Estimation module. The module receives the whiten-data from the avalon bus through DMAs. And with the whiten matrix transferred by the iEn_data and iWh_matrix signals, the ICA Estimation module computes the decomposition matrix W. The W matrix is given to the Compute Result module by the $oDone$ and $oMatrix\_W$ signals. The W matrix is a $4*4$ matrix upon the 4-channel separation application.

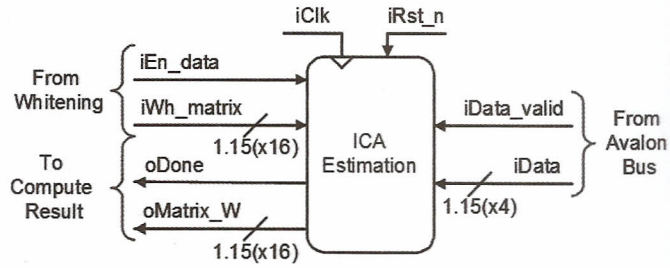

Fig. 8: Top-view of ICA Estimation module.

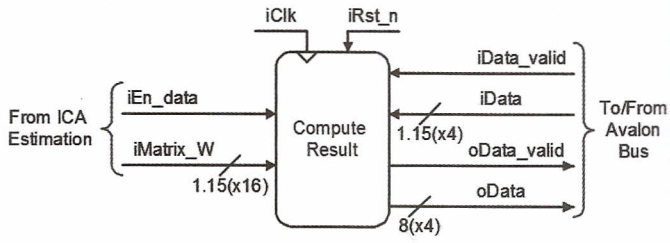

Fig. 9: Top-view of Compute Result module.

ISBN: 978-4-88552-294-9

TABLE I: Resources experimental results compared with other implementations.

	Proposed Design	[16]	[17]	[18]	[19]
Algorithm	FastICA	Parallel ICA	ICNN	Optimized ICA	Infomax
FPGA Chip	Altera Stratix IV SGX230	Xilinx Virtex V1000E	Xilinx Virtex XCV 812 E	Xilinx Virtex II XC2V 8000	Altera Cyclone II C35F
No. of channel	4	N/A	N/A	N/A	4
Length of samples	2^9 to 2^{25}	6,000	500	10,000	N/A
Data width	8	N/A	N/A	16	8
Sample rates (kHz)	1,408	N/A	N/A	57.53	64
Slices	N/A	11,318	12,271	5,500	N/A
Combinational logic	16,099	19,114	N/A	N/A	16,605
Registers	10,934	6,061	N/A	N/A	N/A
Memory bits	9,216	N/A	N/A	N/A	24,576
Other resources	645 DSP block 18-bit elements	N/A	N/A	N/A	N/A
F_{Max} **(MHz)**	62	20.161	50	185.58	68

6) Compute Result: The top-view of the Compute Result module is shown in the Fig. 9. After receiving the decomposition W matrix by iEn_data nad $iMatrix\_W$ signals, the Compute Result module reads the centered-data by avalon bus signals through DMAs. The centered-data are multiplied with the decomposition W matrix. And the result of that multiplication is the final result of the ICA algorithm.

Naturally, the result data are 1.15-bit fixed-point signed numbers due to the width of both W matrix and centered-data. However, owing to the original audio data are 8-bit unsigned numbers, the result data that writen back to PC Computer must be 8-bit unsigned numbers, too. It can done by removing 8 *least significant bits* of the multiplication result which has the width of 1.15-bit fixed-point signed. Then, the result data become 1.7-bit fixed-point signed numbers. By considering the dot of the fixed-point doesnot exists, equals to multiply with 2^7, then they are 8-bit signed numbers. Finally, the *most significant bit* is reversed (i.e. by NOT logic gate), equals to add the results with $+128$ value, then the data from 8-bit signed numbers become 8-bit unsigned numbers. After that, the results data are transferred to the PC Computer via $oData$ and $oData\_valid$ signals in order to complete the whole ICA computation.

IV. Experimental Results

The proposed implementation is designed and verified by Verilog HDL code and Altera Stratix IV SGX230 FPGA chip. Tab. I gives the resources results and compared with the other implementations in [16]–[19]. The design claimed to has the maximum frequency of 62 MHz as shown in the table. The design consumes over 9,000 memory bits, and most of the memory resources are used for fifos. In the design, all fixed-point multiplications and divisions along with all super mathematical computations such as square root are built based on the CORDIC (COordinate Rotation DIgital Computer) algorithm. By using CORDIC, the less memory resources are needed, and the timing performance is improved.

With the sample rates at 1,408 samples per second, 8-bit data width, and F_{Max} equals 62 MHz, the proposed design achieves the speed of 11.27 mega-bit per second (MBps). In the comparison with other designs, it is clear that the proposed implementation has better timing performances. The strong advantages of the design is the variable sample-length from 2^9 to 2^{25} samples for each processing. With the common audio sampling rate of 44.1 kHz, the proposed design can separate an audio wave that has the length about 11.61 milisecond to 12 minutes 40.87 second.

The results data are compared with the ideal results extracted from the MATLab software tool. The MSE (Mean Square Error) is deployed to quantitive the comparison. The implementation results have been tested under various length and audio samples. After all, the average MSE value approximately equals to $1e-3$.

V. Conclusion

A variable-length 4-channel FPGA-based implementation has been presented in this paper. The system is built on Altera Stratix IV SGX230 FPGA chip for the verification. FastICA algorithm is chosen for the implementation along with the *pow3 kurtosis* equation. The proposed design can separate 4-channel with the length vary from 2^9 to 2^{25} samples for each time of processing. The experimental results show that the implementation achieves maximum frequency of 62 MHz with the speed of 11.27 Mbps. The design claimed to process over 1.4 million samples per second with 8-bit resolution input audio wave. The proposed implementation uses CORDIC algorithm to compute the fixed-point multiplications and divisions along with super mathematical computation. By deploying CORDIC method, system reduces the memory resources also with improves the timing performances. The results are compared with the ideal results from MATLab in order to quantitive the accuracy of the implementation. And the MSE value it achieves is approximated to $1e-3$.

Acknowledgment

This work was granted under project C2014-18-04 by the Vietnam National University.

References

[1] T. W. Lee, M. S. Lewicke, and T. J. Sejnowski, "ICA mixture models for unsupervised classification of non-gaussian classes and automatic context switching in blind signal separation," *in IEEE Trans. on Pattern Anal. Mach. Intell.*, Vol. 22, No. 10, pp. 1078-1089, Oct. 2000.

[2] T. W. Lee, "Independent Component Analysis: Theory and Applications," Boston, MA: Kluwer, 1998.

[3] M. Lennon, G. Mercier, M. C. Mouchot, and L. Hubert-Moy, "Independent component analysis as a tool for the dimensionality reduction and the representation of hyperspectral images," *in Proc. SPIE Remote Sens.*, Vol. 4541, pp. 2893-2895, Toulouse, France, Sep. 2001.

TABLE I: Resources experimental results compared with other implementations.

	Proposed Design	[16]	[17]	[18]	[19]
Algorithm	FastICA	Parallel ICA	ICNN	Optimized ICA	Infomax
FPGA Chip	Altera Stratix IV SGX230	Xilinx Virtex V1000E	Xilinx Virtex XCV 812 E	Xilinx Virtex II XC2V 8000	Altera Cyclone II C35F
No. of channel	4	N/A	N/A	N/A	4
Length of samples	2^9 to 2^{25}	6,000	500	10,000	N/A
Data width	8	N/A	N/A	16	8
Sample rates (kHz)	1,408	N/A	N/A	57.53	64
Slices	N/A	11,318	12,271	5,500	N/A
Combinational logic	16,099	19,114	N/A	N/A	16,605
Registers	10,934	6,061	N/A	N/A	N/A
Memory bits	9,216	N/A	N/A	N/A	24,576
Other resources	645 DSP block 18-bit elements	N/A	N/A	N/A	N/A
F_{Max} **(MHz)**	62	20.161	50	185.58	68

6) Compute Result: The top-view of the Compute Result module is shown in the Fig. 9. After receiving the decomposition W matrix by iEn_data nad $iMatrix\_W$ signals, the Compute Result module reads the centered-data by avalon bus signals through DMAs. The centered-data are multiplied with the decomposition W matrix. And the result of that multiplication is the final result of the ICA algorithm.

Naturally, the result data are 1.15-bit fixed-point signed numbers due to the width of both W matrix and centered-data. However, owing to the original audio data are 8-bit unsigned numbers, the result data that writen back to PC Computer must be 8-bit unsigned numbers, too. It can done by removing 8 *least significant bits* of the multiplication result which has the width of 1.15-bit fixed-point signed. Then, the result data become 1.7-bit fixed-point signed numbers. By considering the dot of the fixed-point doesnot exists, equals to multiply with 2^7, then they are 8-bit signed numbers. Finally, the *most significant bit* is reversed (i.e. by NOT logic gate), equals to add the results with $+128$ value, then the data from 8-bit signed numbers become 8-bit unsigned numbers. After that, the results data are transferred to the PC Computer via $oData$ and $oData\_valid$ signals in order to complete the whole ICA computation.

IV. Experimental Results

The proposed implementation is designed and verified by Verilog HDL code and Altera Stratix IV SGX230 FPGA chip. Tab. I gives the resources results and compared with the other implementations in [16]–[19]. The design claimed to has the maximum frequency of 62 MHz as shown in the table. The design consumes over 9,000 memory bits, and most of the memory resources are used for fifos. In the design, all fixed-point multiplications and divisions along with all super mathematical computations such as square root are built based on the CORDIC (COordinate Rotation DIgital Computer) algorithm. By using CORDIC, the less memory resources are needed, and the timing performance is improved.

With the sample rates at 1,408 samples per second, 8-bit data width, and F_{Max} equals 62 MHz, the proposed design achieves the speed of 11.27 mega-bit per second (MBps). In the comparison with other designs, it is clear that the proposed implementation has better timing performances. The strong advantages of the design is the variable sample-length from 2^9 to 2^{25} samples for each processing. With the common audio sampling rate of 44.1 kHz, the proposed design can separate an audio wave that has the length about 11.61 milisecond to 12 minutes 40.87 second.

The results data are compared with the ideal results extracted from the MATLab software tool. The MSE (Mean Square Error) is deployed to quantitive the comparison. The implementation results have been tested under various length and audio samples. After all, the average MSE value approximately equals to $1e-3$.

V. Conclusion

A variable-length 4-channel FPGA-based implementation has been presented in this paper. The system is built on Altera Stratix IV SGX230 FPGA chip for the verification. FastICA algorithm is chosen for the implementation along with the *pow3 kurtosis* equation. The proposed design can separate 4-channel with the length vary from 2^9 to 2^{25} samples for each time of processing. The experimental results show that the implementation achieves maximum frequency of 62 MHz with the speed of 11.27 Mbps. The design claimed to process over 1.4 million samples per second with 8-bit resolution input audio wave. The proposed implementation uses CORDIC algorithm to compute the fixed-point multiplications and divisions along with super mathematical computation. By deploying CORDIC method, system reduces the memory resources also with improves the timing performances. The results are compared with the ideal results from MATLab in order to quantitive the accuracy of the implementation. And the MSE value it achieves is approximated to $1e-3$.

Acknowledgment

This work was granted under project C2014-18-04 by the Vietnam National University.

References

[1] T. W. Lee, M. S. Lewicke, and T. J. Sejnowski, "ICA mixture models for unsupervised classification of non-gaussian classes and automatic context switching in blind signal separation," *in IEEE Trans. on Pattern Anal. Mach. Intell.*, Vol. 22, No. 10, pp. 1078-1089, Oct. 2000.

[2] T. W. Lee, "Independent Component Analysis: Theory and Applications," Boston, MA: Kluwer, 1998.

[3] M. Lennon, G. Mercier, M. C. Mouchot, and L. Hubert-Moy, "Independent component analysis as a tool for the dimensionality reduction and the representation of hyperspectral images," *in Proc. SPIE Remote Sens.*, Vol. 4541, pp. 2893-2895, Toulouse, France, Sep. 2001.

[4] Lan-Da Van, Di-You Wu, and Chien-Shiun Chen, "Energy-Efficient FastICA Implementation for Biomedical Signal Separation," *in IEEE Trans. on Neural Networks,* Vol. 22, No. 11, pp. 1809-1822, Nov. 2011.

[5] M. Phegade, and P. Mukherji, "ICA based ECG signal denoising," *in Proc. of Int. Conf. on Advances in Computing, Communications and Informatics (ICACCI),* pp. 1675-1680, Aug. 2013.

[6] M. Milanesi, N. Vanello, V. Positano, MF Santarelli, R. Paradiso, D. De Rossi, and L. Landini, "Comparative evaluation of decomposition algorithms based on frequency domain blind source separation of biomedical signals," *in Proc. of Int. Conf. on Mathematical Methods and Computational Techniques in Electrical Engineering WSEAS,* pp. 324-329.

[7] Wei Lu and J. C. Rajapakse, "Approach and applications of constrained ICA," *in IEEE Trans. on Neural Networks,* Vol. 16, No. 1, pp. 203-212, Jan. 2005.

[8] L. De Lathauwer, B. De Moor, and J. Vandewalle, "Independent component analysis and (simultaneous) third-order tensor diagonalization," *in IEEE Trans. on Signal Processing,* Vol. 49, No. 10, pp. 2262-2271, Oct. 2001.

[9] A. J. Bell and T. J. Sejnowski, "An information maximization approach to blind separation and blind deconvolution," *Neural Computation,* Vol. 7, No. 6, pp. 1129-1159, Nov. 1995.

[10] T-W. Lee, M. Girolami, and T. J. Sejnowski, "Independent Component Analysis using an Extended Infomax Algorithm for Mixed Sub-Gaussian and Super-Gaussian Sources," *Neural Computation,* Vol. 11, pp. 417-441, 1999.

[11] A. Hyvärinen and E. Oja, "A fast fixed-point algorithm for independent component analysis," *Neural Computation,* Vol. 9, No. 7, pp. 1483-1492, 1997.

[12] Chiu-Kuo Chen, E. Chua, C. Fu, Shao-Yen Tseng, and Wai-Chi Fang, "A hardware-efficient VLSI implementation of a 4-channel ICA processor for biomedical signal measurement," *in Proc. of IEEE Int. Conf. on Consumer Electronics (ICCE),* pp. 607-608, Jan. 2011.

[13] Wei-Yeh Shih, Jui-Chieh Liao, Kuan-Ju Huang, Wai-Chi Fang, G. Cauwenberghs, and Tzyy-Ping Jung, "An efficient VLSI implementation of on-line recursive ICA processor for real-time multi-channel EEG signal separation," *in Proc. of IEEE Int. Conf. on Engineering in Medicine and Biology Society (EMBC),* pp. 6808-6811, July 2013.

[14] Kuan-Ju Huang, Wei-Yeh Shih, Jui Chung Chang, Chih Wei Feng, and Wai-Chi Fang, "A pipeline VLSI design of fast singular value decomposition processor for real-time EEG system based on on-line recursive independent component analysis," *in Proc. of IEEE Int. Conf. on Engineering in Medicine and Biology Society (EMBC),* pp. 1944-1947, July 2013.

[15] Hongtao Du, Hairong Qi, and Xiaoling Wang, "Comparative Study of VLSI Solutions to Independent Component Analysis," *in IEEE Trans. on Industrial Electronics,* Vol. 54, No. 1, pp. 548-558, Feb. 2007.

[16] Hongtao Du and Hairong Qi, "An FPGA implementation of parallel ICA for dimensionality reduction in hyperspectral images," *in Proc. IEEE Int. of Geoscience and Remote Sensing Symposium (IGARSS '04),* Vol. 5, pp. 3257-3260, Sept. 2004.

[17] AB. Lim, J. C. Rajapakse, and AR. Omondi, "Comparative study of implementing ICNNs on FPGAs," *in Proc. of Int. Joint Conf. on Neural Networks (IJCNN '01),* Vol. 1, pp. 177-182, 2001.

[18] M. Ounas, S. Chitroub, R. Touhami, M. Yagoub, and S. Gaoua, "Digital circuit design for FPGA based implementation of ICA for real time Blind Signal Separation," *in Proc. of Int. Conf. on Microelectronics (ICM 2008),* pp. 60-63, Dec. 2008.

[19] Wei-Chung Huang, Shao-Hang Hung, Jen-Feng Chung, Meng-Hsiu Chang, Lan-Da Van, and Chin-Teng Lin, "FPGA implementation of 4-channel ICA for on-line EEG signal separation," *in Proc. of IEEE Int. Conf. on Biomedical Circuits and Systems (BioCAS 2008),* pp. 65-68, Nov. 2008.

ISBN: 978-4-88552-294-9