Intelligent Information Management, 2010, 2, 46-52
doi:10.4236/iim.2010.21006 Published Online January 2010 (http://www.scirp.org/journal/iim)

Scientific Research

Adaptive Parallel Computation for Blind Source Separation with Systolic Architecture

H. JEONG[1], Y. KIM[2], H. J. JANG[3]
[1]*Department of EEE, POSTECH, Pohang, South Korea*
[2]*Digital media & Communication R&D Center, Samsung Electronics, Suwon, South Korea*
[3]*Department of Information Technology POSTECH, Pohang, South Korea*
Email: {*hjeong, ddda, hjj*0501}*@postech.ac.kr*

Abstract

The purpose of Blind Source Separation (BSS) is to obtain separated sources from convolutive mixture inputs. Among the various available BSS methods, Independent Component Analysis (ICA) is one of the representative methods. Its key idea is to repetitively update and calculate the measures. However, dealing with the measures obtained from multi-array sensors causes obstacles for real-time use. In order to solve this problem, it is necessary to convert the software implementation of BSS algorithm into the hardware architecture. Through the use of hardware architecture, the BSS algorithm can efficiently work within a relatively short time. In this study, we investigate a practical method using a parallel algorithm and architecture for hardware use in a blind source separation. We design a feedback network for real-time speech signal processing. The network is composed of forward and updates algorithms. The architecture of the network is systolic and therefore it is suitable for parallel processing. We only have to add and connect modules for scaling. This paper covers the process from the systolic design of BSS to the hardware implementation using Xilinx FPGAs. The simulation results of our proposed implementation are also represented in the experimental section. In that section, our architecture returns satisfying results with robust qualities.

Keywords: BSS, convolutive mixtures, VLSI, field programmable gate array (FPGA)

1. Introduction

In the signal processing area, BSS is considered to be one of the fundamental problems. BSS assumes a Multi-Input-Multi-Output (MIMO) model. It is primarily based upon the principle that we can recover independent sources from mixture inputs. Among various forms of BSS, the simplest form is when mixtures are assumed to be linear instantaneous mixtures of sources [1]. In the 1980's, Jutten and Herault [2] formalized the problem and there have been many models proposed to solve this problem [3–6].

From the many available models, this paper suggests the use of K. Torkkola's feedback network [7–9] algorithm because it has the ability to deal with convolutive mixtures. For the learning algorithm, we propose T. Nomura's extended Herault-Jutten method [10] algorithm. By using these algorithms, the linear systolic architecture of an efficient BSS method can be designed and implemented in this paper. This architecture is composed of forward and updates processors. In the chip, we connected each processing element in a systolic manner. Therefore, we can easily scale up the architecture by adding more identical chips. Fabricated FPGA enables us to reduce the development period and verify the algorithms using hardware at a low cost, even though we cannot optimize the hardware implementation with FPGA when compared to Application-Specific-Integrated-Circuit (ASIC). Accordingly, we use a Very-High-speed-integrated-circuit-Hardware-Description-Language (VHDL) and fabricated FPGA in order to design the BSS chip.

Our paper mainly consists of three parts: Section 2 derives an algorithm for a feedback network, and Section 3 shows detailed architecture of the feedback network. Se-

ctions 4 and 5 show the simulation results and conclusion.

2. Backgrounds

This section provides an introduction to convolutive mixing model. Then, it focuses on K. Torkkola's feedback network algorithm and T. Nomura's extended Herault-Jutten method, which is implemented in software by Choi and Cichocki's method [6,11].

2.1. Convolved Mixture Model

In a convolved mixture model in which a real speech signal is assumed, the acoustic environment imposes a different impulse response between each source and array sensor. Given N statistically independent speech sources $\mathbf{s}(t)=[s_1(t), s_2(t), \ldots, s_N(t)]^T$ and M signals measured at the array sensors $\mathbf{x}(t)=[x_1(t), x_2(t), \ldots, x_M(t)]^T$, the mixing model is represented as

$$x_i(t) = \sum_p \sum_{j=0}^{n} h_{ij,p}(t) s_j(t-p), \text{ for i=1,2,...,m,.} \quad (1)$$

Here, $\{h_{ij,p}\}$ is the room impulse response between the j^{th} source and the i^{th} microphone and $x_i(t)$ denotes the signal present at the i^{th} microphone at time instant t. If we simplify the model to two mixture inputs of two independent sources (M = 2, N = 2) in the Z domain, the model can be shown as

$$\begin{cases} X_1(z) = S_1(z) + H_{12}(z) S_2(z), \\ X_2(z) = H_{21}(z) S_1(z) + S_2(z). \end{cases} \quad (2)$$

2.2. Feedback Network Algorithm

[7,12] suggests a feedback network algorithm as follows

$$y_i(t) = x_i(t) + \sum_{p=0}^{L} \sum_{j \neq i}^{n} w_{i,j,p}(t) y_j(t-p), \text{ for i, j=1,2,...,n,} \quad (3)$$

where $w_{ij,p}$ is the weight between $y_i(t)$ and $y_j(t-p)$. (4) is the abbreviated form of the output vector y(t).

$$\begin{aligned} y(t) &= x(t) + \sum_{p=0}^{L} W_P(t) y(t-p), \\ &= \left[I - W_0(t)\right]^{-1} \{x(t) + \sum_{P=1}^{L} W_P(t) y(t-p)\}. \end{aligned} \quad (4)$$

$$\begin{cases} Y_1(z) = X_1(z) + W_{21}(z) Y_2(z), \\ Y_2(z) = X_2(z) + W_{12}(z) Y_1(z). \end{cases} \quad (5)$$

Figure 1 and (5) represent (4) in the Z-domain. If the feedback network performs perfect separation (Y (z) = S(z)), then we can easily derive the weight from (2) and (5):

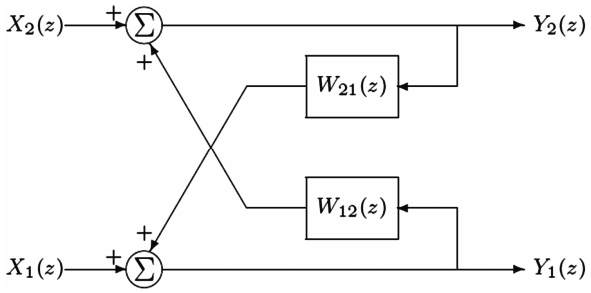

Figure 1. Feedback network for the separation of convolutively mixing sources

$$\begin{cases} W_{12}(z) = -H_{12}(z), \\ W_{21}(z) = -H_{21}(z). \end{cases} \quad (6)$$

The existence of this solution requires $H_{12}(z)$ and $H_{21}(z)$ to have stable inverses. If these are not fulfilled, the network is unable to achieve separation.

In [2], the Jutten-Herault algorithm suggested a learning algorithm of weight W for instantaneous mixtures. [10] extended the Jutten-Herault algorithm and proposed the blind separation model, based upon the assumption that observable signals are convolutively mixed. The learning algorithm updates W using the following form.

$$W_P(t) = W_P(t-1) - \eta_t f(y(t)) g(y^T(t-p)), \quad (7)$$

In (7), η_t is the learning rate, and $f(.)$ and $g(.)$ are odd symmetric functions. The update rule is based on the gradient descent method. It is easy to see that the correlation between $f(y_i(t))$ and $g(y_j(t-p))$ is removed when the learning algorithm achieves convergence. In this paper, we use the signum function as $f(.)$ and the 1st order linear function as $g(.)$ for easy hardware implementation.

2.3. The Modified Algorithm

By assuming that the speech signals are quasi-stationary, we can use the r^{th} frame input $\boldsymbol{x}^r(t)$ and the $(r-1)^{th}$ frame weight $\boldsymbol{w}^{r-1}(t)$ in order to obtain the r^{th} frame output $\boldsymbol{y}^r(t)$. The r^{th} frame weight $\boldsymbol{w}^r(t)$ is also obtained from $\boldsymbol{y}^r(t)$. With this logic, we can modify the forward and update process Equations (4) and (7) as follows

$$\begin{aligned} y^{(r)}(t) &= x^{(r)}(t) + \sum_{P=0}^{L} W_p^{(r-1)}(t) y^{(r)}(t-p) \\ &= \left[I - W_0^{(r-1)}(t)\right]^{-1} \{x^{(r)}(t) + \sum_{P=1}^{L} W_P^{(r-1)(t)} y^{(r)}(t-p)\} \end{aligned} \quad (8)$$

$$W_P^{(r)}(t) = W_P^{(r-1)}(t-1) - \eta_t f(y^{(r)}(t)) g(y^{(r)}(t-p))^T \quad (9)$$

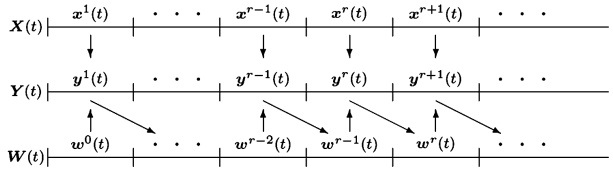

Figure 2. Timing diagram of modified algorithm

Figure 2 represents a schematic diagram of the modified algorithm.

3. Systolic Architecture for Feedback Network

In this section, we will establish systolic architecture for the forward and update process. Each sub part contains overall architecture and internal architecture. The computational complexity is also analyzed in this section.

3.1. Systolic Architecture for Forward Processor

First, we will introduce the overall architecture of the forward process. This architecture is spatially efficient because it can accommodate more time delays with a given limited space. The below is the cost of the forward processor $f_{i,p}(t)$ at time index t.

$$f_{i,0}(t) = 0,$$
$$f_{i,p}(t) = f_{i,p-1}(t) + (w_{ij,p}y_j(t-p) + w_{ij,0}w_{ji,p}y_i(t-p)) \quad \text{for p=1,2,...,L.} \tag{10}$$

We can combine (4) and (10) as follows.

$$\begin{cases} y_1(t) = (1-w_{12,0}w_{21,0})^{-1}\{x_1(t) + w_{12,0x2}(t) + f_{1,L}(t)\}, \\ y_2(t) = (1-w_{21,0}w_{12,0})^{-1}\{x_2(t) + w_{21,0x1}(t) + f_{2,L}(t)\}. \end{cases} \tag{11}$$

Figure 3 is the systolic array architecture of the forward processor. This architecture calculates the PE cost with $L + 1$ processing elements. When $p = 1, 2, ..., L$, the p^{th} PE uses inputs $y_1(t-p)$, $y_2(t-p)$, and $f_{i,p-1}(t)$ from $(p-1)^{th}$ PE, and weights $w_{12,p}$ and $w_{21,p}$ in order to updates PE cost $f_{i,p}(t)$ by (10). At the end of the array $p = L+1$, $y_1(t)$ and $y_2(t)$ are calculated by using (11). This array is very scalable and can be easily implemented. This array allows us to considerably reduce computational complexity. Each PE consists of 2 adders, 2 registers, and 2 multipliers. The internal structure of the PE is represented in Figure 4.

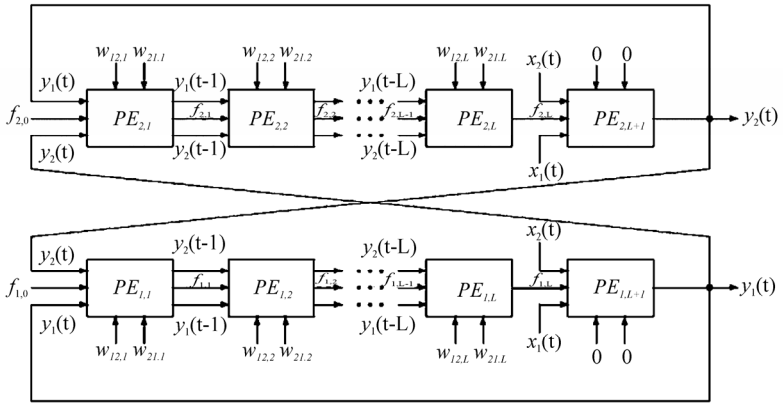

Figure 3. Linear systolic array for forward processor

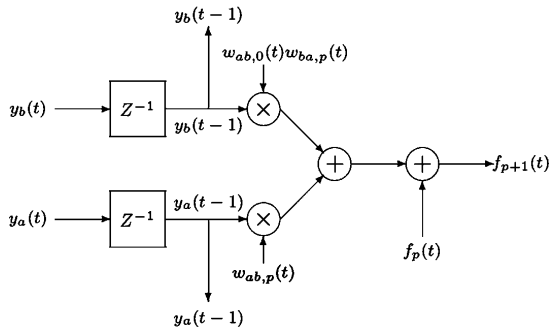

Figure 4. The internal structure of the processing element

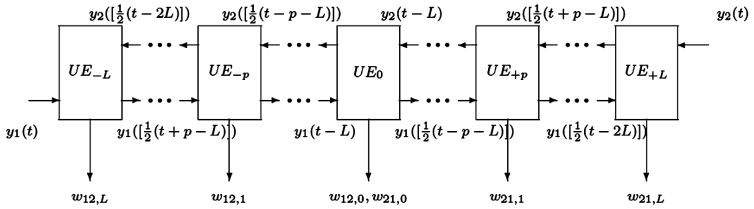

Figure 5. Linear systolic array for the update processor

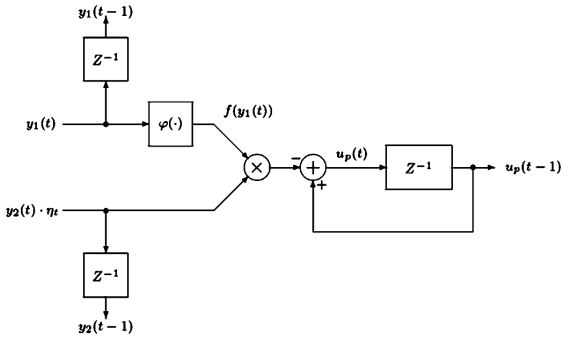

Figure 6. The internal structure of update element

The forward process of the feedback network algorithm consists of two linear arrays of *(L + 1)* PEs. 4L buffers are required for output ***y(t)*** and the cost of PE needs *2L* buffers. Since each output ***y(t)*** is By bits long, a memory with $O(LB_y)$ bits is needed. In this manner, each PE must store the partial cost of B_p bit and thus additional $O(LB_p)$ bits are needed. Therefore, we need the total $O(LB_y + LB_p)$ bits.

3.2. Systolic Architecture for the Update Processor

This part shows the architecture used for weighted updates. This architecture also consists of processing elements with a similar operation. In the update processor, we call the processing element Update Element (UE). If the number of outputs is two (n = 2), (7) can be written as

$$\begin{cases} w_{12,P}(t) = w_{12,P}(t-1) - \eta_t f(y_1(t)) y_2(t-p), \\ w_{21,P}(t) = w_{21,P}(t-1) - \eta_t f(y_2(t)) y_1(t-p). \end{cases} \tag{12}$$

Figure 5 shows the systolic array for the update processor. In the figure, *2L + 1* UEs are connected in series when the number of PE is L. We also define the forward process clock and the update process clock as CLK_f and $CLK_u = 2*CLK_f$ respectively. The even UEs are only active at even times. Similarly, the odd UEs are only active at odd times. The UE equation can be represented as

$$\begin{cases} \mu_P(t) = \mu_P(t-1) - \eta_t \varphi(y_1[1/2(t-p-L)])) y_2[1/2(t+p-L)]), \\ \mu_P(t) = \mu_P(t-1) - \eta_t \varphi(y_2[1/2(t-p-L)])) y_1[1/2(t+p-L)]). \end{cases} \tag{13}$$

Here, [x] denotes the maximum integer which is not over x.

If the number of PE is *L*, the architecture of the update consists of *2L + 1* UEs shown in Figure 5. We denote the forward process clock as CLK_f , and the update clock is $CLK_u = 2*CLK_f$. At even times, only even UEs are active and the odd UEs are inactive. At odd times, the UE play the roles in a reversed way. The UE equation has the form

$$\begin{cases} \mu_P(t) = \mu_P(t-1) - \eta_t \varphi(y_1[1/2(t-p-L)])) y_2[1/2(t+p-L)]), \\ \mu_P(t) = \mu_P(t-1) - \eta_t \varphi(y_2[1/2(t-p-L)])) y_1[1/2(t+p-L)]). \end{cases} \tag{14}$$

where *[x]* is the maximum integer which is not over *x*. Figure 6 is the internal structure for the UE. The P^{th} UE receives two inputs $y_1(t)$ and $y_2(t)$, and returns the updated UE cost $u_p(t)$ as a final result. In the figure, *f()* is the signum function, $f(y_1(t)) = sign(y_1(t))$.

Figure 6 shows the internal structure of the UE of the feedback network. The p^{th} UE receives two inputs $y_1(t)$ and $y_2(t)$, then one input becomes $f(y_1(t))$. Finally, it updates the UE cost $u_p(t)$. In this architecture, *f()* is the signum function, $f(y_1(t)) = sign(y_1(t))$. The update of the

feedback network algorithm uses a linear array of (*2L + 1*) UEs. The output *y(t)* and the cost of UE need *4L + 2* and *2L + 1* buffers respectively. If UE stores the partial cost of Bu bits, the total $O(LB_y + LB_w)$ bits are sufficient.

3.3. Overall Architecture

Figure 7 and Table 1 represent the whole feedback network architecture. We assume that there are two mixture inputs from two independent sources. The goal of the architecture is to reduce the correlation between the inputs as much as possible. In 3.3, we represent the overall feedback network algorithm. The system consists of Initialization, Forward process, Update process, and Weights update. The initialization step is for setting the state suitable for the beginning of the system. At this step, all of the weights and costs of processing elements are set to zero. Through the forward process step and the update process step, all of the weights are calculated and reloaded recursively for the separation.

4. Experiments

Using VHDL, we designed the system for implementation with an FPGA(Xilinx Virtex-II XC2V8000). The device utilization can be seen in Table 3. We fully tested the chip with and without noise. In this section, we tested our method based upon a VHDL simulation. We also tested the chip by extensively using ModelSim simulation

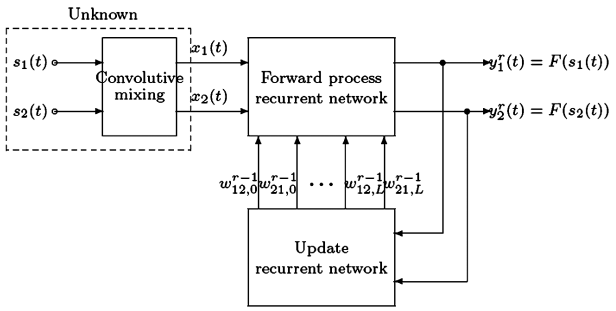

Figure 7. Overall block diagram of feedback network architecture

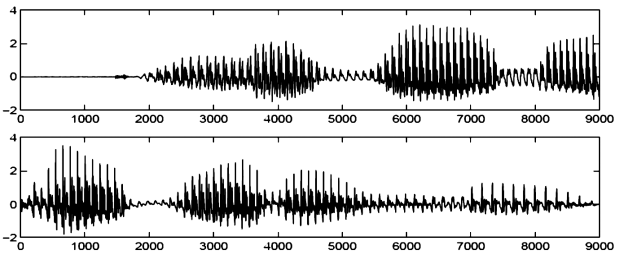

Figure 8. Two original speech signals

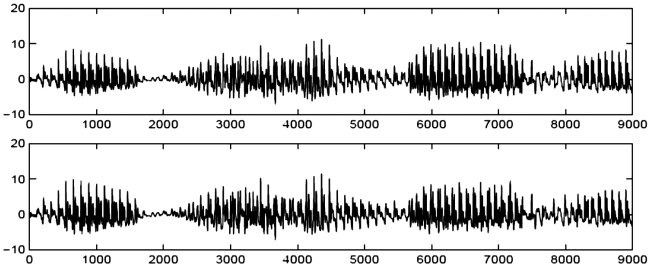

Figure 9. Two mixtures of speech signals

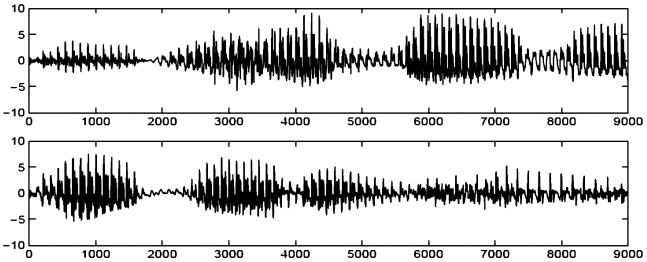

Figure 10. Two recovered signals using the feedback network

Table 1. Algorithm overall of recurrent network architecture

	(1) Initialization
for t=0	All the weights are initialized as zeros : $\mathbf{W}_p$=0 The cost of processing elements are initialized as zeros : $f_{1,p}$=0, $f_{2,p}$=0
	(2) Forward process
for T=1, 2.,,,,N-1	For p=1,2,…,L: $f_{1,p}= f_{1,p-1}+(w_{12,p}y_2(t-p)+ w_{12,0}\ w_{21,p}y_1(t-p))$ $f_{2,p}= f_{2,p-1}+(w_{21,p}y_1(t-p)+ w_{21,0}\ w_{12,p}y_2(t-p))$ For p=L: $y_1(t)=(1- w_{12,0}\ w_{21,0})^{-1}\ \{x_1(t)+ w_{12,0}\ x_2(t)+f_{1,L}\ \}$ $y_2(t)=(1- w_{21,0}\ w_{12,0})^{-1}\ \{x_2(t)+ w_{21,0}\ x_1(t)+f_{2,L}\ \}$
	(3) Update process
for T=N, N+1,…, 3N-1	If t=even, then for p=even : $u_p(t)= u_p(t-1)-\eta_t\varphi(y_1([1/2(t-p-L)]))\ y_2([1/2(t+p-L)])$ else t=odd, then for p=odd : $u_p(t)= u_p(t-1)-\eta_t\varphi(y_1([1/2(t-p-L)]))\ y_2([1/2(t+p-L)])$
	(4) Weights update
for t=3N	For p=-L,…,-2,-1 : $w_{12,p}=u_{-p}(t)$ For p=0 : $w_{12,0}= w_{21,0}=u_0(t)$ For p=1,2,…,L : $w_{21,p}=u_p(t)$

Table 2. The experimental results of SNRI with noisy mixtures

		SNRI1(dB)	SNRI2(dB)
SNR (signal to ratio) = $10\log_{10}$(s/n)	Clean	2.7061	4.9678
	10dB	2.6225	4.9605
	5dB	2.4396	4.7541
	0dB	2.3694	4.6506
	-5dB	1.9594	3.9757
	-10dB	0.1486	3.2470

Table 3. Results from two-inputs two-output implementation (L=40)

	Forward Processor	Update Processor	Overall
Number of Slices	32,692	20,629	44,027
Number of Slice FF	46,184	5,801	52,391
Number of 3 inputs LUTs	49,883	36,490	87,394
Max. frequency	63.529MHz	86,760MHz	63.529MHz

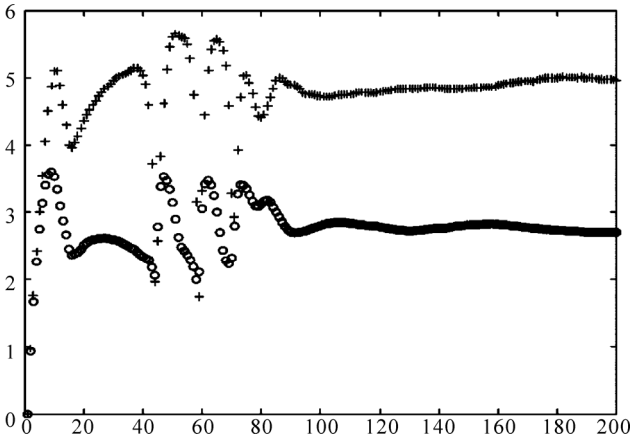

Figure 11. The convergence of SNRI ('o': SNRI1, '+': SNRI2)

tools. It is designed to interface with the PLX9656 PCI chip. For the parametric setup, we fixed $L = 100$ and $\eta_t = 10^{-7}$. As a performance measure, we used a signal to noise ratio improvement by Choi and Cichocki [11], as

$$SNRI_i = 10\log_{10}\frac{E\{(x_i(k)-s_i(k))^2\}}{E\{(y_i(k)-s_i(k))^2\}} \quad (15)$$

Figure 8 shows two different sampled speech signals, which were used for this simulation. The received signals ***x(t)*** are the mixture records of ***s(t)*** and the recovered signals ***y(t)*** and are shown in Figure 9 and 10 respectively. In this test, the experiment results are $SNRI_1 = 2.7061$, $SNRI_2 = 4.9678$.

We have also tested the performance of our method in noisy environments. In Table 2, the system shows a robust performance even in high SNR (above 0dB only). Figure 11 is the graph showing the convergence of learning for the feedback network. In the figure, our system seems to converge at r $\geq$ 100.

5. Conclusions

In this paper, the systolic algorithm and architecture of a feedback network have been derived and tested with VHDL code simulation. This scheme is fast and reliable since the architectures are highly regular. In addition, the processing can be done in real time. The full scale system can be easily obtained by the number of PEs, and UEs. Our system has two inputs but we will extend it for N inputs.

Because the algorithms used for hardware and software implementation differ significantly it will be difficult, if not impossible, to migrate software implementations directly to hardware implementations. The hardware needs different algorithms for the same application in terms of performance and quality. We have presented a fast and efficient VLSI architecture and implementation of BSS. The architecture has the form of a linear systolic array using simple PEs that are connected with only neighboring PEs and thus can be easily scalable with more identical chips.

6. Acknowledgement

This work has been supported by Brain Korea 21 project and Ministry of Knowledge Economy under Human Resources Development Program for Convergence Robot Specialists.

7. References

[1] T. Lee, A. Bell, and R. Orglmeister, "Blind source separation of real world signals," in ICNN, 1997.

[2] C. Jutten and J. Herault, "Blind separation of source, part I: An adaptive algorithm based on neuromimetic architecture," Signal Processing, Vol. 24, pp. 1–10, 1991.

[3] F. Asano, S. Ikeda, M. Ogawa, and H. Asoh, and N. Kitawaki, "Combined Approach of Array Processing and Independent Component Analysis for Blind Separation of Acoustic Signals," IEEE Transactions Speech and Audio Processing, Vol. 11, No. 3, pp. 204–215, 2003.

[4] S. C. Douglas, Malay Gupta, Hiroshi Sawada, and Shoji Makino, "Spatio-Temporal FastICA Algorithms for the Blind Separation of Convolutive Mixtures," IEEE Transactions Audio, Speech and Language Processing, Vol. 15, No. 5, pp. 204–215, 2007.

[5] R. Aichner, H. Buchner, F. Yan and W. Kellermann, "A real-time blind source separation scheme and its application to reverberant and noisy acoustic environments," EURASIP Journal on Applied Signal Processing, pp. 1260–1277, 2007.

[6] M. Ounas, S. Chitroub, R. Touhami, M. C. E. Yagoub, "Digital circuit design of ICA based implementation of FPGA for real time Blind Signal Separation", MLSP 2008. IEEE Workshop on, October 2008.

[7] K. Torkkola, "Blind separation of convolved sources based on information maximization," Proceedings IEEE Workshop Neural Networks for Signal Processing, pp. 423–432, 1996.

[8] K. Torkkola, "Blind separation of delayed source based on information maximization," Proceedings ICASSP, Atlanta, GA, pp. 7–10, May 1996.

[9] K. Torkkola, "Blind Source Separation for Audio Signal-Are we there yet?" IEEE Workshop on Independent Component Analysis and Blind Signal Separation, Aussois, France, January 1999.

[10] T. Nomura, M. Eguchi, H. Niwamoto, H. Kokubo and M. Miyamoto, "An Extension of The Herault-Jutten Network to Signals Including Delays for Blind Separation," IEEE Neurals Networks for Signal Processing, VI, pp. 443–452, 1996.

[11] S. Choi and A. Cichocki, "Adaptive blind separation of speech signals:Cocktail party problem," in Proeeding International Conference Speech Processing (ICSP'97), pp. 617–622, August 1997.

[12] N. Charkani and Y. Deville, "Self-adaptive separation of convolutively mixed signals with a recursive structure - part I: Stability analysis and optimization of asymptotic behaviour." Signal Processing, Vol.73, No. 3, pp. 255–266, 1999.

