실시간 음성 분리 시스템 구현을 위한 고속 병렬구조의 하드웨어 아키텍쳐

정 홍, 김 용, 성주희[O]
포항 공과 대학교
{hjeong, ddda, match2[O]}@postech.ac.kr

Parallel Hardware Architecture for Real-time Blind Source Separation

Hong Jeong, Yong Kim, Juhee Seong[O]
Dept. of Electrical Engineering, POSTECH

요 약

독립적인 여러 개의 음원의 convoultive mixture로부터 blind source separation(BSS)을 수행하는 것은 수년간 활발히 연구되어 오고있다. 그러나 많은 BSS 알고리즘이 존재함에도 불구하고, 직접적으로 하드웨어를 구현할 수 있는 알고리즘은 실제로 매우 드물다. 이 논문의 목표는 FPGA를 이용하여 실시간으로 효과적인 구현이 가능한 BSS 구조를 소개하는 것이다.

1. 서 론

요즘 개발되고 있는 음성인식 시스템들은 잡음이 없는 음성의 경우에는 매우 높은 인식률을 보이고 있지만, 잡음이 포함되는 경우에는 그러하지 못한 경우가 많다. 그러나 음성인식 시스템이 사용되는 환경은 여러 가지 잡음을 포함하고 있는 경우가 대부분이기 때문에 실생활에 응용하기란 한계가 있다.

본 논문에서는 convolutive mixing되어 있는 다수의 독립된 신호를 분리하는 방법으로 K. Torkkola가 제안한 feedback network[1]과 학습 알고리즘으로 T. Nomura에 의해 제안된 gradient descent method를 사용하여 Herault-Jutten method를 확장한 extend the Herault-Jutten method[4]를 바탕으로 고속 병렬 아키텍쳐를 설계하였다.

본 아키텍쳐는 크게 forward process와 update process부분으로 구성되어 있다. 두 부분 모두 간단한 구조를 가지는 프로세싱 엘리먼트를 병렬로 배열하는 방식으로 설계가 되어 그 효율이 높고, 실시간 분리가 가능한 시스템으로 설계되었다.

2. 이론적 배경

BSS을 하기 위해 우선 서로 독립된 음성신호가 convolutive mixture되어 마이크로 들어오는 mixing model을 가정한다. 음성 데이터 벡터를 $\vec{s}(t) = [s_1(t), s_2(t), \ldots, s_n(t)]^T$ 라고 하고 마이크를 통해 들어오는 신호 벡터를 $\vec{x}(t) = [x_1(t), x_2(t), \ldots, x_m(t)]^T$라고 하면 x(t)는 다음과 같다.

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

$\{h_{ij,p}\}$는 j번째 음성과 i번째 마이크 사이의 room impulse response이다.

Convolution mixture되어 있는 신호를 feedback network을 이용하여 BSS 하는 알고리즘은 이미 [1, 2]등에 소개되었다. 우선 이 알고리즘에 대해 간단히 설명하겠다. 시간 지연(delay)을 L번째까지 제한한 경우 feedback network의 i번째 output을 $y_i(t)$라고 하면 다음과 같다.

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

이때 $w_{ij,p}$는 $y_i(t)$와 $y_j(t-p)$의 synaptic weight이다. 이를 매트릭스(matrix) 형태로 하면 다음과 같다.

$$\begin{aligned} \vec{y}(t) &= \vec{x}(t) + \sum_{p=0}^{L} \vec{W}_p(t) \vec{y}(t-p), \\ &= [\vec{I} - \vec{W}_0]^{-1} \{\vec{x}(t) + \sum_{p=1}^{L} \vec{W}_p \vec{y}(t-p)\}. \end{aligned} \quad (3)$$

여기서 n=2 라고 하고 z-domain 에서 보면 다음과 같다. 그림 1 은 feedback network 를 그린 그림이다.

$$\begin{cases} Y_1(z) = X_1(z) + W_{12}(z) Y_2(z), \\ Y_2(z) = X_2(z) + W_{21}(z) Y_1(z). \end{cases} \quad (4)$$

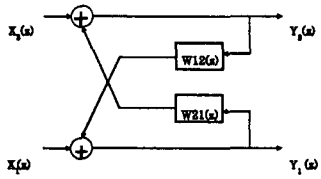

그림 1: The feedback network for the separation of convolutive mixing sources

Extension Jutten-Herault algorithm[4,5]이 synaptic weight $\vec{W}$ 의 학습 알고리즘으로 사용되었고 다음과 같이 정의된다.

$$\vec{W}_p(t) = \vec{W}_p(t-1) + \eta_t f(\vec{y}(t))\vec{y}^T(t-p) \qquad (5)$$

여기서 η_t 는 learning rate로 $f(\vec{y}(t))$ 와 $\vec{y}^T(t-p)$ 의 correlation이 없어질 때 이 학습 알고리즘은 수렴한다.

비선형 함수f(·)는 signum function, $f(y_i(t))=sign(y_i(t))$ 나 hyperbolic tangent function, $f(y_i(t))=\tanh(y_i(t))$ 가 많이 사용되는데 이 논문에서는 하드웨어로 구현이 목적이므로 구현이 간단한 signum function을 사용하였다.

3. 고속 병렬 아키텍처

3.1 Forward Processing을 위한 고속 병렬 구조

고속 병렬 아키텍처는 프로세싱 엘리먼트 (PE, processing element)들로 구성되어 있고, 각각의 PE는 동일한 구조를 가지고 있어 하드웨어로 구현함에 있어 최적의 구조를 가지고 있다.

PE의 코스트를 다음과 같이 정의한다.

$$f_{i,0}(t) = 0$$

$$f_{i,p}(t) \equiv s_{i,p-1}(t) + (w_{ij,p}y_j(t-p) + w_{ij,0}w_{ji,p}y_i(t-p)) \quad \text{for } p=1, 2, \ldots, L \qquad (6)$$

(3)을 n=2로 하고, (6)를 이용해 다시 풀어 쓰면 다음과 같다.

$$\begin{aligned} y_1(t) &= -(1-w_{12,0}w_{21,0})^{-1}\{(x_1(t)+w_{12,0}x_2(t)+f_{1,L}(t)\} \\ y_2(t) &= -(1-w_{21,0}w_{12,0})^{-1}\{(x_2(t)+w_{21,0}x_1(t)+f_{2,L}(t)\} \end{aligned} \qquad (7)$$

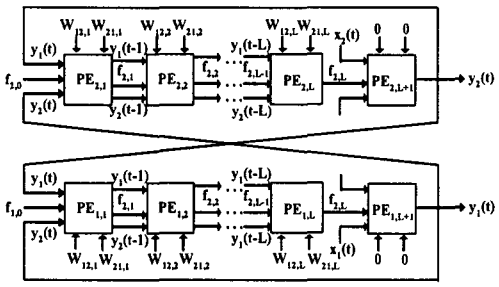

그림 2: Linear systolic array for dynamic feedback network

그림 2는 systolic array구조를 가지는 forward process의 블록 다이어그램이다. 각 단계마다 필요한 코스트를 구하기 위한 PE는 L개씩 둘 줄이 있는데, 이 2L개의 PE를 거쳐서 구해진 코스트를 (7)에 넣어 마지막 단에서 약간 변형된 PE를 거침으로써, 각각 $y_1(t)$와 $y_2(t)$을 출력한다.

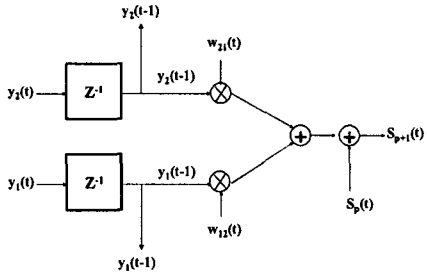

그림 3: The internal structure of processing element

그림 3은 dynamic feedback network 의 1번째부터 L번 째 까지 PE의 구조이다. PE는 각각 2개의 레지스터, 가산기, 곱셈기로 구성 되어 있다. 전 단계의 PE에서 $y_1(t)$, $y_2(t)$이 입력으로 들어오면 레지스터를 통과한 후 각각 weight를 곱하여 더하게 된다. 이 값은 전 PE에서 들어온 코스트 $f_{i,p-1}(t)$에 누적이 되어 $f_{i,p}(t)$값이 된다.

3.2 Update를 위한 고속 병렬 구조

여기서는 앞 단원에서 사용되는 weight를 학습하는 부분의 하드웨어 아키텍쳐에 대해서 알아보겠다. 이 아키텍쳐 또한 동일한 동작을 하는 프로세싱 엘리먼트가 병렬로 연결되어 있는 구조를 가지고 있다. 앞의 forward process의 PE와 구별하기 위해 update에서 사용되는 프로세싱 엘리먼트는 UE(Updating Element)라고 하겠다.

(5)를 n=2로 하고, 식을 풀어 쓰면 다음과 같다.

$$\begin{cases} w_{12,p}(t) = w_{12,p}(t-1) - \eta_t f(y_1(t))y_2(t-p), \\ w_{21,p}(t) = w_{21,p}(t-1) - \eta_t f(y_2(t))y_1(t-p). \end{cases} \qquad (8)$$

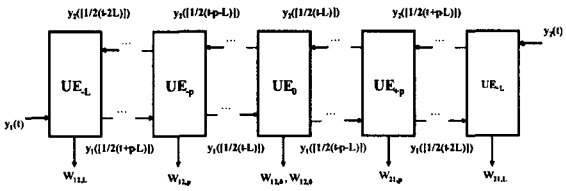

그림 4: Linear systolic array for update

그림 4에서 볼 수 있듯이 weight를 입력으로 갖는 PE의 개수가 L개라고 하면 UE는 개수는 2L+1개로 이루어져 있다. 따라서 forward process와 update부분이 서로 동시에 작동을 하기 위해서 update의 클럭은 forward process의 클럭보다 2배가 빠르게 설계를 하였다. 즉 forward process의 클럭 스피드를 CLK_f라고하면 update 클럭 스피드는 $CLK_u = 2*CLK_f$가 된다.

Forward process 의 output 으로 $y_1(t)$, $y_2(t)$가 CLK_f 의 속도

로 나오면 update 부분에서는 그림 4에서 볼 수 있듯이 각각 양쪽 끝에서 같은 값을 두 번씩 CLK_u으로 들어가게 한다. 이는 속도를 2배 빨리하는 대신 같은 값을 두 번씩 입력으로 들어가게 되므로 forward process의 output과 update의 input 이 동기가 이루어진다. 또한 UE는 홀수 번째 시간에서는 홀수 번째만, 짝수 번째 시간에서는 짝수 번째만 작동을 하게 된다.

그림 5는 UE의 내부 구조를 나타내는 그림이다. 입력으로 각각 $y_1([1/2(t-p-L)])$와 $y_2([1/2(t+p-L)])$이 들어오면 그 중 하나를 $f(y_1([1/2(t-p)]))$와 같이 함수를 취하는데 본 아키텍처에서는 $f((y_1([1/2(t-p)])) = sign((y_1([1/2(t-p)]))$으로 signum function을 사용하기로 한 바 있다. (여기서 [x]는 x를 넘지 않는 최대 정수를 의미한다.)

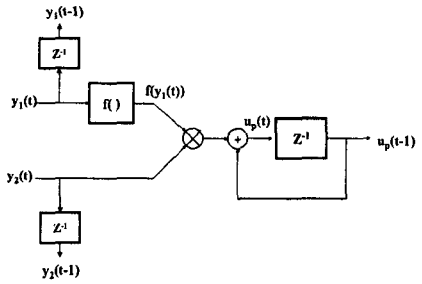

그림 5: The internal structure of update element

Feedback network의 전체 블록 다이어그램은 다음과 같다. 우리가 분리하고자 하는 음성($s_1(t)$, $s_2(t)$)은 마이크($x_1(t)$, $x_2(t)$)로 들어오기까지 convolutive mixing이 된다. 이 과정을 모르기 때문에 음성 신호가 spatially independent하다는 가정을 바탕으로 spatially dependence을 줄이는 방향으로 feedback network의 weight를 학습시킨다.

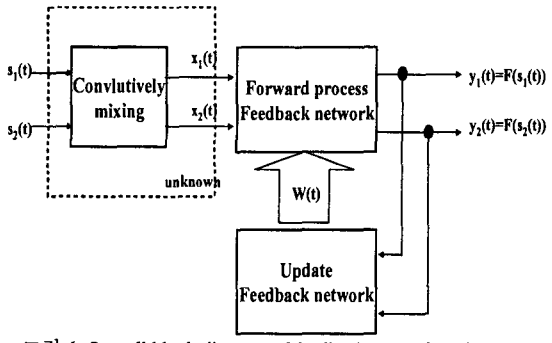

그림 6: Overall block diagram of feedback network architecture

3.3 복잡도

Forward process는 시간지연이 L일때 (L+1)개씩 2쌍의 PE가 선형 배열로 이루어져 있다. 본 시스템은 출력 $\tilde{y}(t)$를 위한 4L개의 버퍼와 PE의 코스트를 위한 2L개의 버퍼가 필요하다. 따라서 출력 $\tilde{y}(t)$가 B_y비트로 이루어져 있고, PE의 코스트가 B_p로 이루어져 있다고 하면 forward process의 경우는 $O(LB_y+LB_p)$비트의 메모리가 필요하다. Update는 2L+1의 UE가 선형배열로 이루어져 있다. 이 부분에서는 $\tilde{y}(t)$을 위한 2(2L+1)의 버퍼가 필요하고, weight를 위한 2L+1의 버퍼가 필요하다. 따라서 weight가 B_w 비트로 이루어져 있다면 update의 경우는 $O(LB_y+LB_w)$비트의 메모리가 필요하다.

3.4 FPGA 하드웨어 시스템 구현

이 시스템은 FPGA(Xilnx Virtex-II XC2V8000)에 맞게 VHDL을 이용하여 설계되었다. 컴퓨터와 설계된 칩과의 인터페이스를 위해서는 PLX9656 PCI chip을 이용하였다.

4. 결론

본 논문에서는 음성의 BSS 를 위해 feedback network 알고리즘을 하드웨어 구현에 적합하게 수정하여 고속 병렬 구조를 가지는 하드웨어 아키텍처를 설계하였다.

현재 시스템은 2 개의 convolutive mixing 된 음성을 분리하는 경우를 설계하였지만, 앞으로 이를 n 개의 입력의 경우로 확장하면서 성능을 향상시킬 계획이다.

감사의 글

이 연구는 BK사업단 및 반도체 설계 교육 센터의 지원을 받음.

참고 문헌

[1] K. Torkkola, "Blind separation of convolvoed source based on information maximization.," *Proc. IEEE Workshop Neural Networks for Signal Processing*, pp.423-432, 1996.

[2] N. Charkani and Y. Deville, "Self-adaptive separation of convolutively mixed signals with a recursive structure - part I: Stability analysis and optimization of asymptotic behavior.," *Signal Processing*, 73(3) pp.255-266, 1999.

[3] 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.

[4] 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 Neural Networks for Signal Processing*, VI pp.443-452, 1996.

[5] K. Torkkola, "Blind separation of delayed source based on information maximaization.," *Proc. ICASSP*, Atlanta, GA, May pp.7-10, 1996.

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

[7] Aapo Hyvarinen, Juha Karhunen and Erkki Oja, "Independent Component Analysis," *John Wiley & Sons, Inc.*, 2001.

[8] K.C. Yen and Y. Zhao, "Adaptive co-channel speech separation and recognition," *IEEE Tran. Speech and Audio Processing*, 7(2):138-151, 1999.

[9] A. Cichocki, S. Amari, and J. Cao, "Blind separation of delayed and convolved signal with self-adaptive learning rate," *in NOLTA*, tochi, Japan, pp. 229-232, 1996.