論文96-33B-9-11

신경회로망을 위한 BiCMOS 난수발생기

(BiCMOS Random Pulse Generator for Neural Networks)

金 糾 泰 *, 崔 奎 烈 **, 鄭 德 鎭 **

(Kuy-Tae Kim, Kyu-Yul Choi, and Duck-Jin Chung)

요 약

확률론적인 방법의 정확한 계산을 위해서 입력값은 난수발생기에서 발생한 난수를 이용해서 정확한 펄스열로 변환되어야 한다. 난수발생기는 초기값에 의하여 그 능력이 많이 결정되므로 본 논문에서는 자유롭게 초기값을 사용할 수 있도록 계수기를 사용한 셀 방식을 제안한다. 확률적인 계산에서는 곱셈수행을 위해서 AND 게이트를 통과하므로 계산된 출력값을 위하여 기존에 나와있는 난수발생기를 이용한 방법과 본 논문에서 제안된 MCA 난수발생기를 이용한 방법에 대해서 비교했다. 확률론적인 방법을 병렬처리하기 위해서 하나의 큰 난수발생기가 여러 개의 작은 난수들을 발생시켜 주는 방법에 대해서 연구하였다. 병렬방법에서는 난수발생기가 많은 입력 비교기를 구동 해주기 위해서 큰 구동능력을 가져야되므로, SPICE를 사용해서 BiCMOS와 CMOS를 이용하한 구동 능력을 비교했다.

Abstract

In the stochastic structure for doing exact calculation, an input number must be changed to a pulse stream. Because the performance of random number generator(RNG) is controlled by its initial condition, we suggested newly modified cellular automata(MCA) which is uses a counter for boundary condition. We compared newly suggested MCA RNG to previously reported RNGs using the AND gate passing outputs which have the same meaning of multiplication in the stochastic calculation. In order to use stochastic we studied about the method, one large RNG can generate many small random numbers. In this method, RNG must have large drive capabilities for many input comparator. So we studied about drive capabilities using BiCMOS circuit and CMOS circuit by SPICE.

I. 서 론

본 논문에서는 디지털 회로에서 문제점인 곱셈기와 비선형 전달함수에 복잡도를 줄이는 방법으로 웨이트가 없는 펄스열의 확률론적인 계산으로 곱셈기와 덧셈기가 간단한 AND게이트와 OR게이트로 대체될 수 있는 확률론적인 방법을 신경회로망에 대해서 사용하였다[1].

확률론적인 방법은 기본적으로 계산을 확률적으로 수행하므로 기계적인 계산 수행을 하는 기존의 방법과 다른 방법이다. 피연산자를 웨이트가 없는 이진 값의 연속으로 재표현함으로써 곱셈이나 덧셈과 같은 기계적인 연산이 매우 간단한 디지털 하드웨어를 사용함으로써 실현할 수 있다[2].

이러한 확률론적인 방법은 대규모의 곱셈과 덧셈이 필요로 되어지는 신경회로망과 같은 구조에 잘 적용이 될 수 있다.

Von Neumann이 주장한 확률론적인 방법의 이론은

* 正會員, LG 半導體
(LG Semiconductor Co.)
** 正會員, 仁河大學校 電子材料工學科
(Inha University Electronic Materials & Devices Engineering
接受日字:1995年10月20日, 수정완료일:1996年8月20日

다음과 같이 간단하게 요약된다: 만일 두 개의 피연산자 A와 B가 [0, 1]의 범위 안에서 두개의 웨이트를 가지고 있지 않은 임의의 연속체인 ^A와 ^B로 다음과 같이 나타내어졌을 때[1]

$$\text{and} \quad \begin{matrix} P(\hat{}A==1)=A \\ P(\hat{}B==1)=B \end{matrix} \tag{1}$$

^A와 ^B가 서로 독립적이라면 곱셈은 다음과 같이 묘사될 수 있으며,

$$P((\hat{}A==1)\ AND\ (\hat{}B==1))=AB \tag{2}$$

^A와 ^B가 서로 소라면 덧셈은 다음과 같이 계산될 수 있다.

$$P((\hat{}A==1)\ OR\ (\hat{}B==1))=A+B \tag{3}$$

다시 말해서 하드웨어적인 관점에서, 기본적인 두 가지 원리는 피연산자들이 확률론적으로 재 표현되었을 때 곱셈은 AND 게이트로 덧셈은 OR 게이트로 수행될 수 있음을 보여준다.

위와 같이 확률론적인 방법에 의해서 연산을 할 때, 피연산자를 서로 독립적이며 웨이트를 가지고 있지 않은 이진 값의 연속으로 만들어 주기 위해서 아날로그 난수와 디지털적인 난수를 사용하게 된다. 아날로그 방법의 경우에는 아날로그적인 난수를 사용하기 때문에 적은 면적으로도 계산을 수행할 수 있지만 주위 잡음에 계산이 많은 영향을 받게 되므로 정밀도에 한계가 있다. 디지털적인 방법에서는 피연산자를 재표현하기 위해서 피연산자와 난수발생기를 이용한 임의의 난수를 비교하여서 펄스를 만들어 준다. 따라서 디지털적인 방법은 예측이 가능하고 주위 잡음에 대해서 면역성이 강하게 나타난다. 그러나, 디지털적인 방법을 통해서 만들어지는 난수의 경우 일정한 규칙을 통해서 난수를 발생시키기 때문에 완전한 난수가 만들어지지 못하고 의사 난수(Pseudo Random Number)가 제작되기 때문에 난수발생기에서 생성되는 난수에 따라서 계산 결과가 변경되게 된다.

확률론적인 방법에서 더욱 정확한 결과값을 얻기 위해서는 서로 독립적인 펄스로 재 표현되어야 하므로 피연산자들은 서로 독립적인 난수들을 필요로 한다.

확률론적인 연산을 하기 위해서 사용하는 의사 난수 발생기의 종류로는 대표적으로 Linear Feedback Shift Register(LFSR)와 Cellular Automata(CA)방법을 사용되는데, LFSR방법은 간단한 구조를 가지고 있으나 난수의 평균이 초기값에 의해서 일정하지 않기 때문에 정확한 연산을 기대하기가 어렵고, 비트가 늘어나도 늘어난 비트 표현에 비해 작은 양의 주기가 늘어나는 문제점을 가지고 있다. CA 방법은 다양한 표현 방법을 가지고 있으나 사용된 값에 비해 주기가 적으며, 초기값이 잘못 선택되었을 경우에 난수가 발생하지 않는 경우도 발생하게 된다.

본 논문에서는 기존에 사용되는 난수발생기를 개선하여 수정된 난수발생기의 방법에 대해서 기존의 난수발생기와 비교, 논의하고, 많은 병렬 연산이 필요로 하는 신경회로망에서 사용하기 위해서 BiCMOS와 CMOS로 회로를 구성한 결과를 SPICE를 사용해서 비교했다.

Ⅱ. Pseudo Random Pulse Generator

1. Stochastic Processing을 위한 Pseudo Random Number Generator의 정의

디지털적인 방법으로 난수를 발생시켰을 때 발생하는 Pseudo random number는 완전한 난수가 아니기 때문에 확률론적인 연산을 하기 위해서 각각의 난수들은 다음과 같은 특징을 가져야 한다.

1) 모든 난수들이 확률론적으로 독립적이어야 한다
2) 피연산자의 정확한 표현을 위해서 난수의 평균이 피연산자 범위의 반이 되어야 한다
3) 모든 범위의 피연산자가 정확히 표현되기 위해서 모든 난수의 발생 빈도는 같아야 한다
4) 피연산자의 재 표현한 결과가 주기적인 영향을 없애기 위해서 모든 난수들의 주기의 최소공배수가 연산하는 주기보다 길어야 한다

2. 난수발생기의 방법

Linear Feedback Shift Register(LFSR)

LFSR은 shift register와 XOR 게이트의 조합으로 구성되는 간단한 구조 때문에 신경회로망에서 사용되는 구조이다. n 비트 shift register를 사용하였을 때에 2^n-1의 주기를 가지게 된다.[2]

주기를 늘이기 위해서는 비트 수를 많게 해주어야 하지만, 비트 수를 늘이게 되면 shift register에 의해서 크기가 이동하는 주기가 나타나게 되어서 난수의 형태가 톱니 형태가 나오게 된다.

Cellular Automata(CA)

CA 방법에서는 자신의 전 값에 의해서 이웃한 다음 값을 결정하는 방법이다. 이 이웃한 위치에 영향을 주는 방식에 따라서 CA의 형태가 결정이 되는데, CA 방법은 전의 값과 그 결과로 만들어지는 값의 관계에 의해서 결정되게 된다. CA rule 90은 다음과 같은 규칙을 가지고 있다[3] [4].

t-1 :	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0
	1 0 1	1 1 0	1 1 1		
t :	0	1	0	1	1
	0	1	0		

위와 같이 만들어진 방식은 논리곱과 합의 형태로 변환이 가능하며, 본 논문에서는 다음과 같은 CA 방법들을 사용하였다.

$$\text{Rule 30} \quad a_i(t+1) = a_{i-1}(t) \oplus \{a_i(t) \cup a_{i+1}(t)\}$$

$$\text{Rule 45} \quad a_i(t+1) = a_{i-1}(t) \oplus \{a_i(t) \cup \overline{a_{i+1}(t)}\} \qquad (4)$$

$$\text{Rule 90} \quad a_i(t+1) = a_{i-1}(t) \oplus a_{i+1}(t)$$

$$\text{Rule 150} \quad a_i(t+1) = a_{i-1}(t) \oplus a_i(t) \oplus a_{i+1}(t)$$

CA 90에서의 형태를 보면 그림 1과 같다.

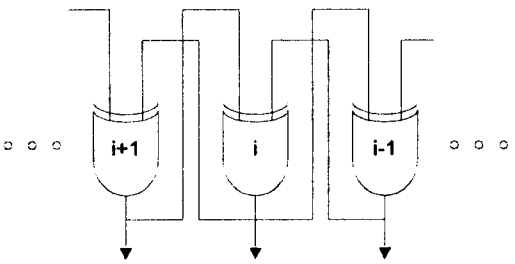


그림 1. CA 90의 형태
Fig. 1. A CA rule 90.

CA 방법을 구현하기 위해서는 경계 조건을 만들어 주어야 하는데, 지금까지 알려진 경계 조건은 그림 2와 같다.

CA 방법의 경우 다양한 변화 방법을 제공하지만 사용된 비트에 비해서 주기가 작은 단점을 가지고 있으며, 초기값에 따라서 주기가 달라지게 된다. 그림 3은 CA rule 30에 대한 순환 구조와 경로를 보여준다.

그림 3과 같이 난수의 주기가 여러 가지가 있으므로 길이가 짧은 주기가 되지 않기 위해서 난수발생기를 제작할 때 초기값을 고려해 주어야 하는 단점을 가지고 있다.

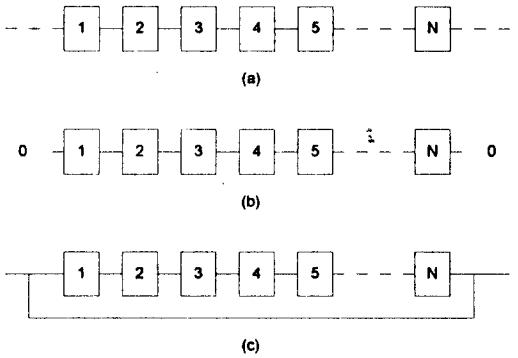


그림 2. (a) 간단한 1차원 CA (b) 0 경계조건 (c) 순환적인 경계조건
Fig. 2. (a) A simple one-dimensional CA. (b) Null boundary conditions, (c) Cyclic boundary conditions.

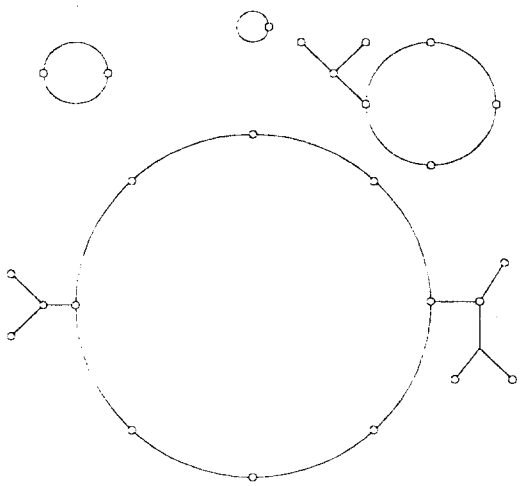


그림 3. CA 30에 대한 전형적인 순환 구조와 경로
Fig. 3. Typical cycles and paths to the cycles for CA rule 30.

Hybrid Cellular Automata(HCA)

HCA 방법은 앞서 설명한 CA 방법 두 가지를 병행하여 사용하는 방법으로 본 논문에서는 그 구조가 상대적으로 간단한 CA 90과 CA 150을 같이 사용하는 방법에 대해서 조사하였다.

Modified Cellular Automata(MCA)

본 논문에서는 CA 방법의 주기와 초기치 문제를 개선하기 위해서, counter를 이용해 CA의 경계조건을 만들어 주는 방법을 제안한다.

그림 4는 counter를 이용해서 개선한 CA의 경계를 보여준다.

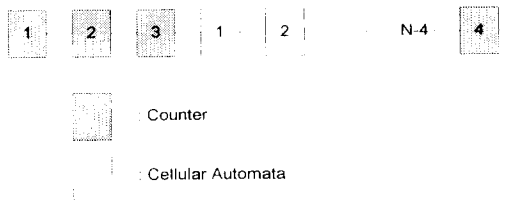


그림 4. 변형된 CA 방법의 형태
Fig. 4. A modified cellular automata(MCA).

그림 4에서와 같이 4비트 counter의 3번 비트와 4번 비트가 CA의 경계가 되면서 독립적인 방식으로 0과 1을 주기적으로 CA로 전달을 하여 주고 따라서 어떠한 초기값에도 상관없이 MCA는 난수를 발생시켜줄 수 있게 된다. 본 논문에서는 CA에서 사용한 네가지 방법에 대해서 MCA방식을 도입하여서 테스트를 한다.

3. 난수의 추출 방법

$1/2^N$의 정확도를 가진 확률론적인 연산을 하기 위해서는 2^N비트 이상의 주기를 가진 난수발생기가 필요로 하게 되는데 N비트의 난수발생기를 가지고는 2^N비트 이상이 되는 난수를 발생할 수 없다. 따라서 N보다 많은 비트, 즉 N+α비트의 난수발생기를 사용해 주어야 하는데, 만일 M개의 입력을 사용한다면 M(N+α)비트를 발생해 주어야 한다. 난수발생기의 크기를 줄이며 상호 독립적인 난수를 발생시켜 주기 위해서 그림과 같이 2차원적으로 저장하여서 각 시간대의 난수를 사용하는 방법이 있다.

그림 5의 경우를 볼 때 M개의 난수를 발생시켜 주기 위해서 많은 메모리가 사용되지만 실제 주기는 첫 번째 난수발생기에 의해서 결정이 되기 때문에, 많은 양의 메모리 소자가 사용되어도 난수 주기 상의 변화는 없게 된다.

따라서 본 논문에서는 주기를 늘이기 위해서 크기가 큰 난수발생기를 사용하며, 각각의 난수를 위해서 난수발생기 중에서 임의 위치의 비트를 선택하여 난수로 사용하는 방법을 제안한다. 이 경우 많은 난수를 필요로 하여도 임의의 위치에서 선택을 할 수 있으므로 난수발생기의 크기를 변경시켜 줄 필요가 없으며, 큰 비트 수의 난수발생기를 사용하기 때문에 난수의 주기도 충분히 크게 할 수 있다.

본 논문에서는 32비트 난수발생기 하나를 사용하여서 피연산자와 임의의 12비트 난수 2개를 비교하여 펄스열로 변환한 후 곱셈에 해당하는 AND 게이트에 대한 결과를 알아보았다.

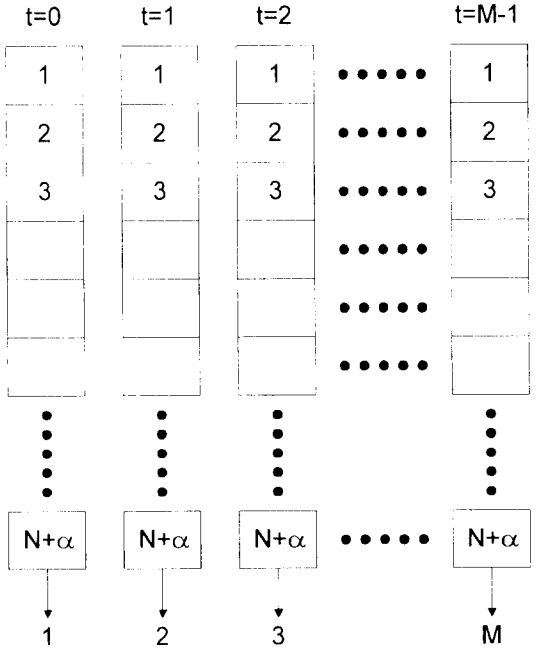


그림 5. M개의 난수를 발생시키기 위한 2차원 구조
Fig. 5. Two dimensional structure for generating M random numbers.

Ⅲ. 실험 결과

1. 난수의 주기와 평균 비교

Ⅱ장 1절에서 설명한 난수에 얼마나 합당한지를 검사하기 위해서 2절에서 사용한 난수발생기 각각에 대해서 주기와 그 평균값을 연산하여 보았다.

32비트의 난수를 발생시켰을 때, 좌우 비대칭 구조인 초기값 1을 주었을 때의 주기와 평균은 다음과 같다.

대칭 구조인 초기값 81818181h(10000001100000010000001100000001$_2$)를 주었을 때의 평균은 다음과 같다.

표 1과 2에서 주기의 크기와 평균을 비교하여 보았을 때 CA 방법의 경우 초기값에 의해서 주기와 평균이 많은 차이를 보이고 있다. 따라서 CA 방법을 사용할 때에는 각 RNG 비트의 길이에 따라서 길이가 긴 주기의 초기값을 따로 조사하여 주어야 한다. 그러나, 표 1의 CA 45에서와 같이 1,515,025개의 난수를 발생한 다음에 2가지 값만 나오는 경우로 수렴하는 것과

같이 그 주기를 찾아보기 힘든 경우도 있게 된다.

표 1. 초기 조건을 1로 하였을 때의 주기
Table 1. The period when initial value is 1.

난수발생기의 종류		주기	평균
LFSR		1023	0.236559
CA rule 30	0 경계 조건	2	0.583333
	순환 경계조건	1842428	0.499391
CA rule 45	0 경계조건	36	0.463125
	순환 경계조건*	2	2
CA rule 90	0 경계조건	62	0.0902
	순환 경계조건	1	0
CA rule1 50	0 경계조건	62	0.249202
	순환 경계조건	16	0.320484
HCA rule 90 rule 150	0 경계조건	1023	0.473633
	순환 경계조건	8	0.816406
MCA rule 30		3360	0.498179
MCA rule 45		352	0.498969
MCA rule 90		524256	0.499999
MCA rule 150		524256	0.500001

* 단, CA rule 45의 경우에는 1,515,025개의 난수를 발생한 다음에 FFFFh와 0값이 번갈아 나왔음.

표 2. 초기 조건을 81818181h로 하였을 때의 주기
Table 2. The period when initial value is 81818181h.

난수발생기의 종류		주기	평균
LFSR		1023	0.236559
CA rule 30	0 경계조건	2	0.583333
	순환 경계조건	8	0.5
CA rule 45	0 경계조건	1	0.285714
	순환 경계조건	45	0.5
CA rule 90	0 경계조건	31	0.300245
	순환 경계조건	1	0
CA rule 150	0 경계조건	31	0.394034
	순환 경계조건	4	0.632353
HCA rule 90 rule 150	0 경계조건	1023	0.494379
	순환 경계조건	2	0.558824
MCA rule 30		3360	0.498179
MCA rule 45		352	0.498969
MCA rule 90		524256	0.499999
MCA rule 150		524256	0.500008

이에 비하여 표 1과 2에서 MCA rule 90과 rule 150의 경우, 순환/비순환에 대해 신경회로망을 위해 필요한 정밀도인 2^{12}=4096에 비해서 큰 주기와 평균값을 가지고 있음을 보여준다[5].

표 3에서는 다양한 대칭/비대칭 초기값에 대한 CA 30과 CA 45와 MCA rule 90과 rule 150의 주기와 평균을 보여준다.

표 3. 다양한 초기값에 대한 CA 30과 CA 45와 MCA 90과 MCA 150의 주기와 평균
Table 3. The periods and average values of CA 30, CA 45, MCA 90 and MCA 150 about several initial values.

초기값	RNG rule	주기	평균
00000001h	CA 30	1842428	0.499391
	CA 45	2	0.5
	MCA 90	524256	0.499998
	MCA 150	524256	0.500010
00081000h	CA 30	5842428	0.499999
	CA 45	2	0.5
	MCA 90	524256	0.499999
	MCA 150	524256	0.500008
11111111h	CA 30	8	0.5
	CA 45	2	0.5
	MCA 90	524256	0.499999
	MCA 150	524256	0.500010
12121212h	CA 30	40	0.5
	CA 45	24	0.416667
	MCA 90	524256	0.499999
	MCA 150	524256	0.500008
12344321h	CA 30	1842428	0.499406
	CA 45	2	0.5
	MCA 90	524256	0.499999
	MCA 150	524256	0.500008
80000001h	CA 30	1842428	0.499529
	CA 45	31636700	0.499985
	MCA 90	524256	0.499999
	MCA 150	524256	0.500008
81818181h	CA 30	8	0.5
	CA 45	2	0.5
	MCA 90	524256	0.499999
	MCA 150	524256	0.500008
87654321h	CA 30	1842428	0.499465
	CA 45	2	0.5
	MCA 90	524256	0.499999
	MCA 150	524256	0.500008

단, CA 30과 CA 45의 경우 순환 경계 조건을 사용하여 계산하였다.

CA 45의 경우를 보면 초기값 80000001h에서 제일 긴 주기를 보이지만 초기값 00000001h, 00081000h, 11111111h, 12344321h, 81818181h, 87654321h에서는 2밖에 되지 않는 주기를 보이며, 특히 00081000h에 대해서는 1,360,161개의 난수를 발생한 다음에 2인 주기로 수렴되고, 87654321h에 대해서는 2,435,892개의 난수를 발생한 후 수렴하고, 12344321h인 주기에 대해서는 1,471,501개의 난수를 발생한 후에 수렴을 하게 된다. 이와 같이 2인 주기로 수렴하는 경우에 대해서 별도의 모의 실험을 통해서 수렴을 하는지 안하는지 조사를 해 주어야 한다.

2. 확률론적인 방법을 이용한 곱셈 결과의 비교

컴퓨터 모의실험을 통하여 확률론적인 방법에 의한 곱셈을 연산하여 보았다. 연산을 위한 구성도은 그림 6과 같다.

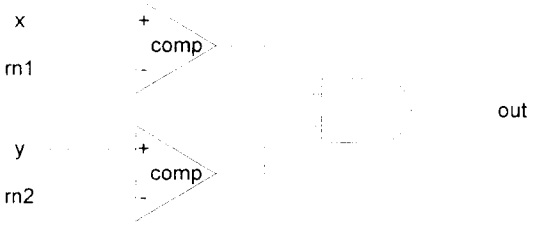


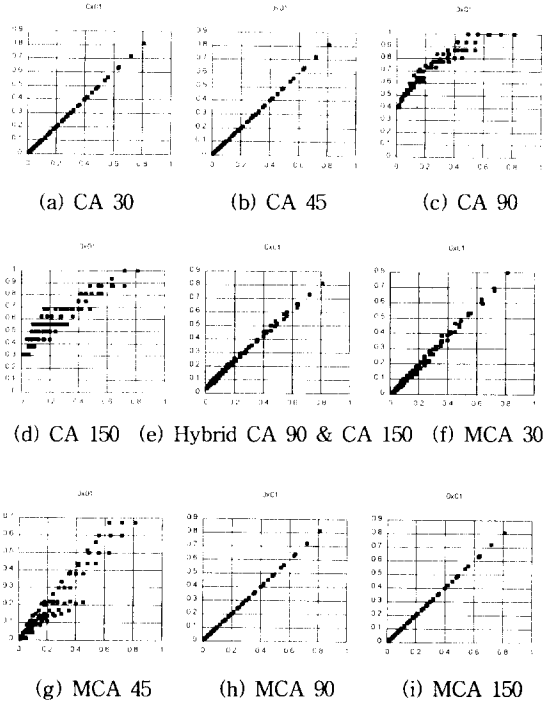
그림 6. 곱셈 결과를 위한 구성도
Fig. 6. Schematic for multiplication.

x의 범위는 [0, 1]이고 y의 범위가 [0, 1]일 때 그림 6에서 난수 rn1과 rn2가 서로 독립적이라면 x와 y는 비교기에 의해서 웨이트 없는 펄스열인 ^x와 ^y로 변하게 되고 AND 게이트에 의해서 논리곱이 수행되어서 out에는 x×y의 결과가 연산되게 된다.

입력 x와 y를 각각 0.1부터 0.9까지 0.1씩 증가를 하였고, 펄스열의 길이를 4096(=2^{12})으로 하였을 때의 연산 결과의 결과는 그림 7과 같다.

그림에서 x축은 x×y의 산술 연산 값을 나타내며, y축은 확률론적인 방법에 의해서 연산된 연산 결과이다. 만일 난수발생기가 서로 독립적인 난수를 발생시킨다면 그래프는 y = x의 그래프의 결과가 나온다.

그림 7에서 보면 CA 30과 CA 45와 MCA 90과 MCA 150에서 좋은 결과를 보여준다. 이 결과는 표 1의 주기와 평균을 보면 쉽게 예측을 할 수 있는데, 표 1에서 CA 30, MCA 90, MCA 150 각각의 주기와 평균을 보면 (1842428, 0.499391), (524256, 0.499999), (524256, 0.500001)과 같이 긴 주기와 거의 0.5에 평균을 가지고 있음을 알 수 있다. CA 45는 정확하게 나오는 것과 같이 보이나 1,515,025개의 난수를 발생한 다음에 FFFFh와 0값이 번갈아 나온다. CA 150의 경우 평균적으로 낮은 값을 가진 난수가 발생하게 되므로 연산 결과가 전체적으로 크게 나옴을 알 수 있다. MCA 30과 MCA 45의 경우에는 주기가 연산 주기 4096보다 작기 때문에 반복적인 형태가 되어서 그 결과가 퍼져 보인다.



(a) CA 30 (b) CA 45 (c) CA 90

(d) CA 150 (e) Hybrid CA 90 & CA 150 (f) MCA 30

(g) MCA 45 (h) MCA 90 (i) MCA 150

그림 7. 초기값 1에 대한 확률론적인 곱셈의 결과. x축; 산술적인 연산값, y축; 확률론적인 방법에 의한 연산값
Fig. 7. The multiplied results when the initial value is 1. x axis; arithmetic results, y axis; stochastic results.

연산 결과가 상대적으로 정확하게 나온 CA 30, CA 45, MCA 90, MCA 150의 네 가지 방식에 대해서 초기값들을 변경했을 때의 연산 결과는 다음과 같다.

표 4의 내용 중에 오차에 관한 내용만을 이용해서 그림으로 표시하면 그림 8과 같이 된다.

그림 8에서 보듯이 CA 30과 CA 45의 경우에는 초기값에 의해서 그 연산 오차가 크게 변함을 알 수 있다. 이 결과와 표 3의 주기를 비교하여 보면, CA 30의 경우 초기값 11111111h에서 주기는 8이고, 초기값

표 4. 다양한 초기값에 대한 오차의 평균과 분산
Table 4. average of errors and standard deviation about several initial value.

Initial Condition	CA 30	CA 45	MCA 90	MCA 150
00000001h	0.000607±2.33e-07	0.000845±4.15e-05	0.000531±1.60e-07	0.000507±2.08e-07
00081000h	0.000526±5.23e-05	0.225353±0.0166	0.000785±5.70e-05	0.000394±3.15e-05
11111111h	0.120617±0.004479	0.283333±0.02	0.000446±2.08e-07	0.000555±1.33e-07
12121212h	0.125±0.007345	0.189114±0.012058	0.000843±8.22e-05	0.000669±4.91e-05
12344321h	0.000572±5.52e-05	0.220603±0.016718	0.000462±3.83e-05	0.000545±4.32e-05
80000001h	0.000571±5.43e-05	0.000925±7.52e-05	0.000437±3.99e-05	0.000744±5.53e-05
81818181h	0.120617±0.007436	0.283333±0.015713	0.000826±6.07e-05	0.000742±5.55e-05
87654321h	0.000632±5.84e-05	0.185652±0.016819	0.000721±6.46e-05	0.000596±5.54e-05
오차평균	0.046143	0.173645	0.000631	0.000594

12121212h에서 주기는 40이고, 초기값 81818181h에서 주기는 2로 주기가 작아지면 연산 오차가 늘어남을 보여준다. CA 45의 경우 긴 주기값을 갖는 80000001h와 00000001h에서 어느 정도 정확한 연산이 나왔는데, 00000001h에서 정확한 연산이 나온 이유는 주기 2로 가는 수렴과정에서 난수가 발생한 것으로 추측된다.

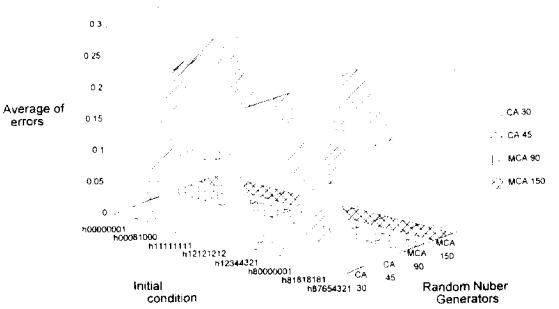
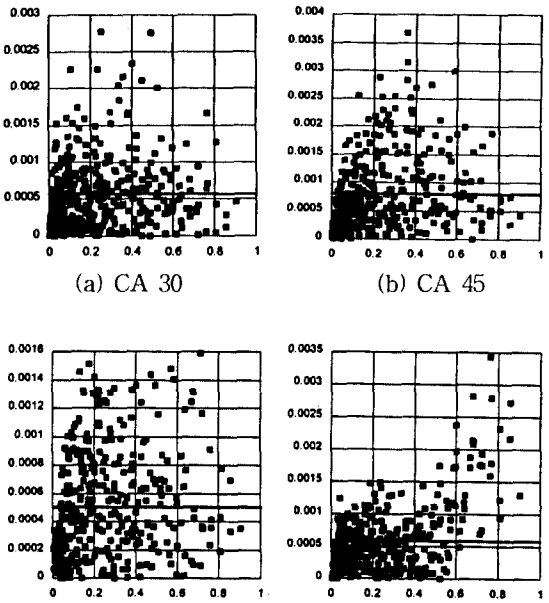


그림 8. 초기값에 따른 오차의 평균
Fig. 8. Average errors about several initial values.

즉, 난수발생기의 주기가 연산 주기(4096)보다 충분히 커졌을 경우에 정확한 연산이 됨을 알 수 있다.

보다 자세한 연산 정도를 알아보기 위하여 입력 x와 y를 각각 0.05부터 0.95까지 0.05씩 증가를 하였고, 초기값을 1로 하였을 때 펄스열의 길이를 4096(=2^{12})으로 하였을 때의 연산 오차의 평균과 분포는 다음 그림과 같다.

그림을 살펴보면 CA 방법의 경우 오차가 큰 폭으로 퍼져있음을 알 수 있으며, CA 방법들에 비해 MCA 90의 방법의 오차 정도가 적음을 알 수 있으며, MCA 150의 경우를 보면 0.5가 넘는 값에서 오차가 늘어나는 경향을 볼 수 있다. 따라서 MCA 90이 제일 적당한 방법으로 기대된다.



(a) CA 30 (b) CA 45

(c) MCA 90 (d) MCA 150

그림 9. 초기값 1에 대한 오차의 정도
Fig. 9. Erros about 1's initial value.

3. 부하 용량의 비교

신경회로망의 특징은 동일한 구조의 연산을 많이 한다는 점이다. 하나의 뉴론의 일부인 확률론적인 연산의 모습은 그림 10과 같다.

그림 10과 같은 연산에서 하나의 x_iw_i의 연산을 하기 위해서는 2개의 독립적인 난수가 필요하게되고 따라서 N개의 입력을 가지고 있는 신경회로망을 위한 난

수의 개수는 다음과 같다.

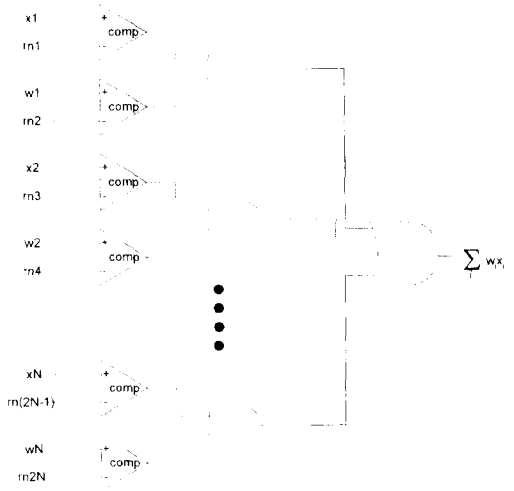


그림 10. Stochastic 구조를 이용한 뉴론의 일부
Fig. 10. The part of neuron's schematic using stochastic processing.

$$\text{Needed Random Numbers} = 2N$$

신경회로망의 M개의 뉴론이 병렬로 사용된다면 다음과 같게 된다.

$$\text{Needed Random Numbers} = 2N \times M$$

그러나, M개의 뉴론의 경우 서로 독립적으로 연산이 되어지기 때문에 상호 독립적인 난수가 필요 없게 되므로 실제로 필요한 독립적인 난수의 개수는 2N이 된다.

필요한 수의 난수를 만들기 위해서 본 논문에서는 난수보다 큰 크기의 난수발생기에서 임의로 필요한 비트의 난수를 선택하는 방법을 채택하였다. K비트의 난수발생기에서 L비트의 난수를 선택할 경우 x를 위한 난수와 w를 위한 난수는 독립적이어야 하므로 난수 둘을 선택하였고, OR 게이트를 위해서 선택된 난수의 경우 x와 w의 곱의 형태로 나타남으로 다시 K비트 중에 임의의 L 비트씩 2개의 난수를 선택하였다. 따라서 이론적으로 선택될 수 있는 난수의 개수는 다음과 같다.

$$ {}_{K}C_{L} \cdot {}_{K-L}C_{L} = \frac{K!}{(K-L)! \cdot L!} \cdot \frac{(K-L)!}{(K-2L)! \cdot L!} = \frac{K!}{(K-2L)! \cdot L! \cdot L!} \quad (6)$$

이 방법을 이용하여 하나의 난수발생기를 이용하여서 많은 수의 난수를 발생할 수 있게 되는데, 이와 같이 L비트로 된 2N개의 난수를 발생하였을 경우 난수발생기 한 개의 비트에 연결된 입력은 평균적으로 다음과 같이 연산될 수 있다.

$$\frac{2N \times M \times L}{K} \quad (7)$$

예를 들어 8개의 입력에 8개의 출력으로 12비트 연산을 할 때 32비트 난수발생기 하나에 연결된 연결선은 평균적으로 48개가된다.

MOS 회로에서는 출력 부하를 구동 해주기 충분하도록 출력 트랜지스터가 커야 한다. 그러나, 큰 출력 트랜지스터는 입력 부하가 늘어나기 때문에 전 단계의 속도를 감소시키게 된다. MOS 버퍼 회로의 등가회로 모습은 그림 11과 같다.

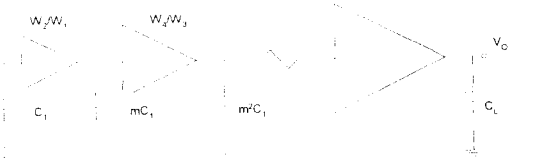


그림 11. MOS 버퍼 회로의 등가회로
Fig. 11. The equivalent circuit for MOS buffer.

또한 단계의 수는 다음과 같이 표현된다[6].

$$N = \ln \frac{C_L}{C_1} \quad (8)$$

입력 용량은 다음과 같다.

$$C_1 = (C_{OX}L + 3C_{GDO})(W_1 + W_2) \quad (9)$$

CMOS의 입력을 두 단계로 하였을 때, 동일 면적 조건에서 half-swing BiCMOS 버퍼와 CMOS 버퍼의 비교를 위해서 표 5의 값을 사용하였다.[6]

표 5. BiCMOS와 CMOS 버퍼의 비교
Table 5. Compare between BiCMOS and CMOS buffers.

Circuit	CMOS	BiCMOS
Buffer stages	2	1
$m = \sqrt[N]{C_L/C_1}$	8.51	-
$A_N = A_1 \frac{m^N - 1}{m - 1}$	$9.51A_1$	-
Emitter area	-	$2 \times 40\,\mu m^2$
Silicon area	$7600\,\mu m^2$	$7600\,\mu m^2$

본 논문에서는 출력 부하가 1p, 3p, 5p일 때를 비교하여 보았으며, 그림 12는 출력 부하에 따른 지연시간의 모습을 보여준다.

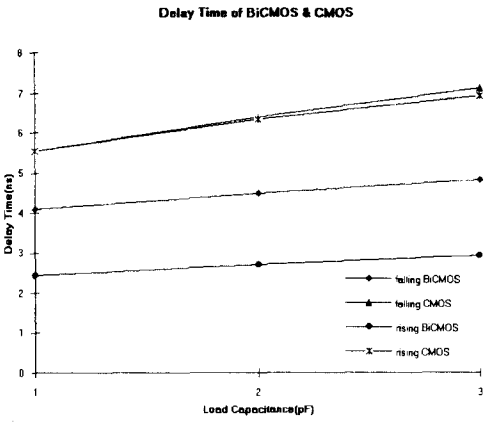


그림 12. CMOS와 BiCMOS의 timing 비교
Fig. 12. Compare the timing of CMOS and that of BiCMOS.

Ⅳ. 결 론

본 논문에서는 많은 병렬 연산이 필요한 신경회로망에서 확률론적인 연산의 수행을 위해서 네 가지의 난수발생기 필요조건을 추론하여 보았으며, 이를 기존의 난수발생기인 LFSR과 CA와 HCA와 본 논문에서 제시된 개선된 난수발생기인 MCA에 적용하였다.

추론된 필요조건과 연산의 정확성을 알아보기 위해서 C 프로그래밍 언어로 난수발생기의 주기와 평균을 구한 후 이 결과를 각각의 난수발생기를 이용한 확률론적인 연산의 정확성을 비교한 결과, 연산의 정확성이 주기와 난수의 평균에 의해서 예측할 수 있었으며, 본 논문에서 제시한 MCA 90과 MCA 150방법의 경우 초기치에 관계없이 규칙적인 주기와 평균을 가지고 있으므로 연산 결과의 정확성을 보여준다.

병렬처리를 필요로 하는 신경회로망에서 확률론적인 방법에 의한 연산을 할 때 병렬적인 특성에 의해서 난수발생기에 많은 출력 부하가 예상이 되므로, Bi-CMOS와 CMOS로 구현한 난수발생기를 SPICE로 모의실험하였다. Bipolar의 빠른 구동능력에 의해서 BiCMOS로 제작된 난수발생기를 사용한 시스템에서는 더 빠른 clock을 사용할 수 있으므로 더욱 향상된 performance를 가질 것으로 예상된다.

※ 본 논문은 '94년 교육부 반도체분야 학술연구조성비에 의해서 지원되었음.

참 고 문 헌

[1] S. T. Ribeiro, "Random Pulse Machine," *IEEE Trans. Electronic Computer*, vol. EC-16, pp. 261-276, 1967.

[2] Dziem Nguyen and Fred Holt, "Stochastic Processing in a Neural Network Application," *International Joint Conference on Neural Networks*, vol. 3, pp. 281-291, 1987.

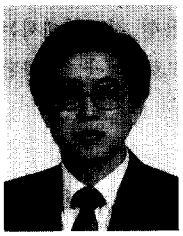
[3] P. D. Dortensis, et al., "Cellular Automata-based pseudorandom number generator for built-in self-test," *IEEE Trans. Computer-Aided Design*, vol. 8, pp. 842-859, 1989.

[4] P. D Dortensis, R. D. Mcleod, and H.C Card, "Parallel random number generator for VLSI systems using cellular automata," *IEEE Trans. Computers*, vol. 38., pp. 1466-1472, 1989.

[5] JeungWon Suh and Soo-Ik Chae, "A Backpropagation Algorithm for Neural Networks Using Random Pulse Stream," *Artificial Neural Networks*, vol. 2, pp. 1035-1038.

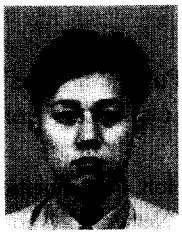
[6] Wen Fang and Arthur Brunnschweiler, "An Accurate Analytical BiCMOS Delay Expression and its Application to Optimizing High-Speed BiCMOS Circuits," *IEEE J. Solid-State Circuits*, vol. 27, No. 2, pp. 191-202, 1992.

저 자 소 개



鄭 德 鎭(正會員)

1948년 2월 8일생, 전자공학회 정회원, 1970년 2월 서울대학교 전기공학과 학사, 1984년 미국 Utah State University 전기공학과 석사, 1988년 미국 University of Utah 전기공학과 박사, 1989년 ~ 인하대학교 전자재료공학과 부교수



崔 奎 烈(正會員)

1996년 인하대학교 전자재료공학과 졸, 현 인하대학교 전자재료공학과 석사과정

金 紐 泰(正會員)

1994년 인하대학교 전자재료공학과 졸, 1996년 인하대학교 전자재료공학과 석사, 1996 ~ 현 LG반도체 연구원