濟州大學校 産業技術硏究所 論文集, 第9卷, 1號, pp.146~155, 1998. 6
Cheju National Univ. R.I.I.T. Jour., Vol.9, No.1, pp.146~155, 1998. 6

# 양방향 연상 기억장치의 ASIC 설계

고 경 희* · 강 민 제**

# Bidirectional Associative Memory Design using an ASIC Design Techniques

Gyung-Hee Ko* and Min-Je Kang**

## ABSTRACT

In this paper, Bidirectional Associative Memory is designed by using an ASIC design technology. This system consists of five blocks that were described with VHDL through top-down design methodology. SYNOPSIS CAD tool is used to simulate and synthesize logically. To implement the system by using FPGA, the FLEX8000 library of ALTERA company is used.

**Key words** : Bidirectional associative memory, ASIC design, VHDL, FPGA

## Ⅰ. 서론

컴퓨터의 처리방식을 인간의 특성과 유사하게 만들고자 하는 노력은 오래 전부터 계속되어 왔는데 그 중의 하나가 신경회로망이다. 신경회로망은 1943년 McCulloch와 Pitts의 논문에서 시작되어 발전해 오다가 60년대 말 기대했던 퍼셉트론(Perceptron)이 간단한 선형분리 문제도 풀어내지 못함을 알게되면서 침체기를 걷게 되었다. 그러던 것이 80년대가 되면서 Hopfield, Hinton, Grossberg, Rumelhart, Kohonen 등이 신경회로망에 대한 새로운 관심을 불러일으키면서 다시 활발한 연구가 시작되었다.[1]

신경회로망에 대한 다각적이고 광범위한 연구가 진행됨에 따라 실제 회로에서의 응용 가능성이 입증되고 있으나 신경회로망에 대한 연구의 대부분은 모델 단계의 연구와 모의 실험에 그치고 있다.

ASIC 기술의 발전은 이러한 이론적인 시스템을 실제 하드웨어로 구현하려는 노력을 가속시켰는데, 특히 하드웨어 기술언어(HDL), 캐드 툴(CAD tool), FPGA의 발전은 사용자가 쉽게 하드웨어로 구현할 수 있는 환경을 만들어주었다.[2]

본 논문에서는 신경회로망의 연상 기억장치의 일종으로 오류가 있는 패턴을 입력했을 때 기억된 패턴쌍을 연상해내는 양방향 연상 기억장치를 ASIC 기술을 이용하여 하드웨어로 설계하였다.

## Ⅱ. 양방향 연상 기억장치

양방향 연상 기억장치는 두개의 층으로 구성된 내

* 제주대학교 대학원
Graduate School, Cheju Nat'l Univ.
** 제주대학교 전자공학과
Dept. of Electronic Eng. Cheju Nat'l Univ.

용-주소 기억장치이면서 이질연상 기억장치이다. Fig. 1은 디지털 양방향 연상 기억장치의 구조를 나타내는데 화살표는 정보의 흐름을 표시한 것이다. 두 개의 층 $X$와 $Y$는 $X=(x_1, x_2, \cdots, x_n)$와 $Y=(y_1, y_2, \cdots, y_m)$의 행벡터이고, 연결강도 $W$로 상호연결 되어 있다. 입력패턴을 이용하여 저장된 패턴을 연상하기 위해 순방향과 역방향의 정보흐름을 사용한다.

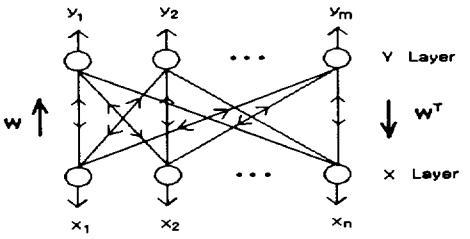


Fig. 1 Topology of BAM.

초기행렬 $X$가 연결강도 $W$를 통해서 뉴론 층 $Y$의 입력으로 들어간다면 $Y'$는 다음과 같이 구할 수 있다.

$$Y' = a(WX) \quad \text{또는,}$$
$$y_j' = a\left(\sum_{i=1}^{n} w_{ij} x_i\right) \quad j=1,2,\ldots,m \tag{1}$$

식 (1)을 가지고 입력패턴 $X$는 $Y$층의 출력패턴으로 처리되고 변환된다. 여기서, $a(\cdot)$는 문턱함수이다. 지금 벡터 $Y'$는 연결강도 $W^T$를 통해서 역방향인 $X$층으로 보내지고 다음의 출력을 만든다.

$$X' = a(W^T Y') \quad \text{또는,}$$
$$x_i' = a\left(\sum_{j=1}^{m} w_{ij}^T y_j\right) \quad i=1,2,\ldots,n \tag{2}$$

연상된 $X'$가 입력 X와 다르면 $X'$는 다시 $Y$층의 입력으로 들어가고 식 (1)에 따라 $Y''$를 만든다. 이 처리과정은 $X$와 $Y$의 변화가 없을 때까지 계속된다.

$Y' = a(WX)$ 첫 번째 순방향과정
$X' = a(W^T Y')$ 첫 번째 역방향과정

$$Y'' = a(WX') \quad \text{두 번째 순방향과정} \tag{3}$$
$$\vdots$$
$$X'^k = a(W^T Y'^k) \quad \text{k 번째 역방향과정}$$

적당한 메모리 동작을 위해서 각층의 변화는 같은 시간에 발생하지 않는다고 가정한다.

## 2.1 양방향 신경회로망의 부호화

양방향 연상 기억장치는 뉴론들 사이의 연결강도들을 수정함에 의해서 학습한다. X층의 뉴론의 수가 n이고, Y층의 뉴론의 수가 m일 때 연결강도는 n열 m행의 행렬이고, X와 Y층 사이의 연결강도는 모든 입력들이 에너지가 감소하는 안정한 방향으로 빠르게 수렴해 간다.

저장될 벡터쌍 $\{(X^{(1)}, Y^{(1)}), (X^{(2)}, Y^{(2)}) \cdots (X^{(p)}, Y^{(p)})\}$들이 p개 있다면 저장 가능한 벡터의 용량 p는 X층에 n개의 뉴론이 존재하고 Y층에 m개의 뉴론이 존재한다면 저장될 패턴의 수는 각 층에 존재하는 뉴론 수 보다 작다. 따라서, $p < \min(n, m)$이고, 좀더 안정적인 패턴인식을 위한 저장용량은 패턴의 수를 $p < \sqrt{\min(n, m)}$로 정해주는 것이 좋다고 알려져 있다.[3]

Hebb의 가설에서 연결강도의 값이 입력패턴과 출력패턴 간의 상관(correlation)값에 따라 변한다고 가정하였는데, 입력이 이진형태 일 때 쌍극형태로 만들어주어 연결강도를 구하는 것이 상관부호화가 개선된다.

$$W = \sum_{s=1}^{p} X^{(s)T} Y^{s} = X^{(1)T} Y^{(1)} + X^{(2)T} Y^{(2)} + \cdots + X^{(p)T} Y^{(p)} \tag{4}$$

여기서, 열벡터 $X^{(i)T}$는 행벡터 $X^{(i)}$의 벡터변환이다.

$$W = \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ & \vdots & & \\ w_{n1} & w_{n2} & \cdots & w_{nm} \end{pmatrix} \tag{5}$$

행렬성분 $w_{ij}$는 뉴런 $x_i$와 뉴런 $y_j$사이에서 대칭적 연결강도를 나타낸다.

양방향 연상 기억장치의 특별한 경우는 X층과 Y층이 같고 모든 $X^{(i)} = Y^{(i)}$일 때 발생한다. 그러면 $W = W^T$이고, 에너지가 최소일 때 단일 패턴 $X^i$에 저장하는 대칭적인 자동연상 기억장치가 된다.

## 2.2 양방향 신경회로망의 복호화

복호화는 오류가 있는 패턴이 입력되었을 때 연결강도를 통하여 저장된 패턴을 연상해내는 과정이다. 입력패턴 X가 X층에 입력되었다면 n개의 뉴론들은 연결강도를 가로질러 Y층의 m개의 뉴론의 값을 만들어내는데 복호화를 위한 입력패턴은 이진형태를 가지며 뉴론 값이 0이면 동작하고, 1이면 동작하지 않는다.

X층에서 입력패턴 $X$가 $X^{(j)}$와 가장 유사한 입력이라면 $Y'$는 다음 식과 같다.

$$\begin{aligned} Y'^{(j)} &= X^{(j)}W = X^{(j)}\sum_{s=1}^{p} X^{(s)T}Y^{(s)} \\ &= X^{(j)}X^{(j)T}Y^{(j)} + X^{(j)}\sum_{s\neq j}^{p} X^{(s)T}Y^{(s)} \end{aligned} \tag{6a}$$

식 (6a)에서 입력된 X가 $X^{(j)}$와 가장 유사한 입력이므로 $X^{(j)}X^{(j)T}Y^{(j)}$가 가장 큰 값을 가지면서 ($X^{(j)}, Y^{(j)}$)에 수렴해가고 만약 저장된 입력벡터들이 서로 직교(orthogonal)한다면 서로 다른 $X^{(s)}$와 $Y^{(j)}$에서 $X^{(s)T}X^{(j)} = 0$이므로 $Y'^{(j)}$는 $Y^{(j)}$가 되어 더욱 정확한 연상이 가능하다. 출력 이진신호를 만들기 위해서 $Y'^{(j)}$을 문턱값으로 자르는데 입력 합이 $Y'^{(j)}$의 문턱값을 초과하면 $Y'^{(j)}$의 출력은 1이고, 문턱값보다 작으면 $Y'^{(j)}$의 출력은 0, 문턱값과 같으면 $Y'^{(j)}$는 현재의 상태를 그대로 유지한다. 구해진 패턴 $Y'^{(j)}$는 연결강도를 통해서 X층의 $X'^{(j)}$로 출력신호가 나간다. X층에서 Y패턴을 구하기 위해서는 W를 사용하고 Y층에서 X패턴을 구하기 위해서는 $W^T$를 사용한다.

$$\begin{aligned} X'^{(j)} &= Y'^{(j)}w_{j1}^T + Y'^{(j)}w_{j2}^T + \cdots + Y'^{(j)}w_{jm}^T \\ &= Y'^{(j)}w_{jj}^T + Y'^{(j)}w_{j1}^T + \cdots + Y'^{(j)}w_{jm}^T \end{aligned} \tag{6b}$$

이렇게 구해진 각 $X'^{(j)}$는 합해진 입력들로부터 문턱함수를 이용하여 이진신호를 만들고, 그것을 다시 Y층으로 돌려보내는 방식으로 반복하여 에너지가 최소화되는 점으로 빠르게 수렴한다.

# III. 양방향 연상 기억장치의 설계

시스템을 ASIC 설계기술을 이용하여 하드웨어로 구현하는 방법에는 여러 가지가 있는데 VHDL을 이용하여 모델링하고 FPGA로 칩을 구현하는 방식으로 양방향 신경회로망을 설계하였다.[2),4),5),6)] 먼저 양방향 연상 기억장치의 구조와 동작특성을 살펴보고 그에따른 설계사양을 결정한다. FPGA로 구현할 수 있는 칩의 크기를 고려하여 X층과 Y층의 뉴론의 개수는 8개와 4개로 제한하여 설계하였다.

X층과 Y층의 패턴입력을 위한 신호(x, y)와 시스템의 동작을 제어하기 위한 신호(clr), 내부신호들을 발생시키기 위한 일정한 신호(clk)를 인가할 수 있게 모두 14개의 입력포트를 구성하고, 연상을 끝낸 패턴을 출력하기 위하여 출력포트를 12개로 구성하였다.

양방향 연상 기억장치의 구현을 위하여 다섯 개의 블록으로 나누어 설계하였는데 각 블록은 VHDL로 코딩하였고, 블록마다 Synopsys 캐드툴을 이용하여 구조적 시뮬레이션을 통하여 기능을 검증하고, 검증이 끝나면 합성을 하였다. 합성된 회로를 가지고 게이트레벨 시뮬레이션을 하여 구조적 시뮬레이션과 같은 결과가 나타나는지를 검증하고, 검증된 블록들을 전체 합성하여 FPGA로 구현하기 위한 네트리스트를 생성하였다.[6),7)]

Fig. 2는 양방향 연상 기억장치의 전체 회로도이다. 양방향 연상 기억장치의 다섯 개의 블록은 상태발생기블록, 입력블록, 기억블록, 연산블록, 출력블록이다.

상태발생기블록은 상태신호를 발생하여 정해진 규칙에 따라 일정하게 동작할 수 있도록 내부에 신호를 공급하는 블록이고, 입력블록은 연상해낼 패턴을 입력하기 위한 블록으로 in_reg 블록과 b_input 블록으로 구성되어 있다. 기억블록은 미리 계산된 대표패턴을 저장하고 연산과정 동안 저장된 값을 신호에 맞게 출력하는 블록으로 b_rom 블록과 b_ad 블록이

있으며. 연산블록은 대표패턴과 입력패턴간의 백터곱에 대한 연산을 수행하는 블록이고. 출력블록은 연상을 통하여 오류가 제거된 패턴을 출력하는 블록으로 b_comp 블록과 out_reg 블록으로 이루어져 있다.

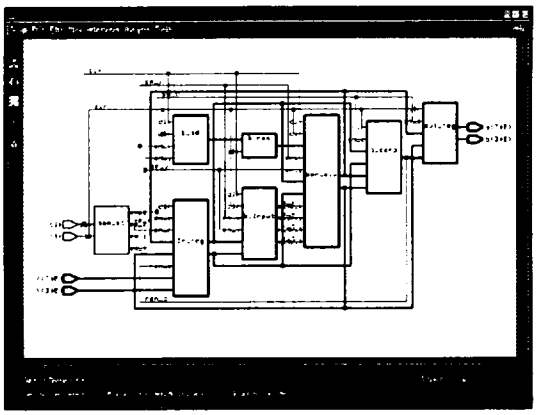

Fig. 2 Total schematic diagram for BAM.

### 3.1 상태발생기블록

상태발생기는 외부에서 입력되는 클럭(clk)을 가지고 두 개의 층(8개의 뉴론과 4개의 뉴론)을 가진 연상 기억장치를 구동시키기 위한 en_x. en_y 신호를 발생시키고. 정확한 연상을 위한 반복을 제어하는 en_c 신호와 외부 입력패턴과 연산을 통해 구해진 입력패턴의 제어를 위한 en_r 신호를 발생시킨다.

Fig. 3은 상태발생기블록이 구조적 시뮬레이션으로 검증된 후 게이트레벨로 합성된 모습을 보여주고. Fig. 4는 게이트레벨 시뮬레이션한 결과이다. 출력신호 en_x는 입력패턴 $X$를 입력 가능하게 하는 신호로 clr이 '0'일 때. clk가 32번 동작할 동안 '1'의 상태를 유지한다. en_y는 입력패턴 $Y$를 입력 가능하게 하는 신호로 clr이 '0'이고. en_x가 '0'일 때 clk가 32번 동작할 동안 '1'이 되는 상태를 반복한다. en_c는 en_x와 en_y가 한번씩 동작(clk가 64번 동작했을 때)하면 한 주기라고 하는데 한 주기가 지날 때 마다 발생하는 신호로 비교기의 동작을 제어한다. en_r은 외부 입력패턴과 연산을 통해 들어온 입력패턴을 제어하는 신호이다.

VHDL을 이용하여 Fig. 4와 같은 출력을 얻는 회로를 설계할 경우 언어의 입장에서 보면 여러 가지

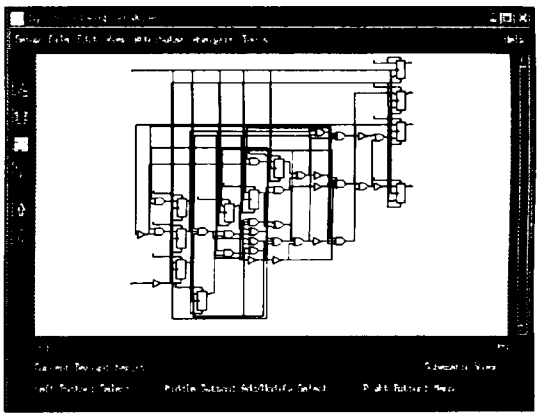

Fig. 3 Schematic diagram of State generator.

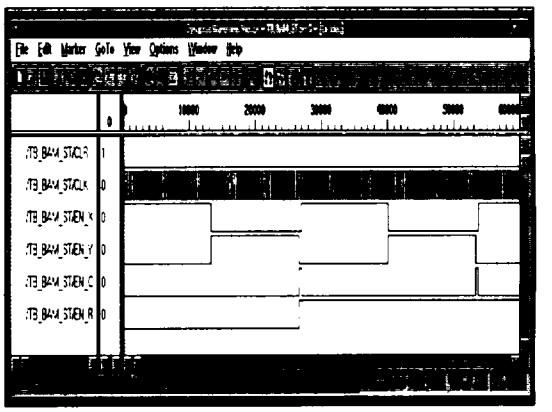

Fig. 4 Output of State generator.

방법이 있다. if 구문만을 이용하여 설계할 수도 있고. if 구문과 elsif 구문을 섞어서 설계할 수도 있고. case구문을 이용하여 설계하는 방법으로도 구현할 수 있다. 하지만 하드웨어의 측면에서 보면 크기와 속도 등을 고려하여 신중한 선택이 필요하다. case구문은 모든 신호의 상태를 기술해주는 ROM을 설계하는 경우에 사용하고 상태발생기의 경우에는 부적당하다. if구문만을 이용하여도 elsif구문으로 연결해 사용한 것과 같은 결과가 나타난다면 if구문으로만 잘라서 설계하는 것이 좋다. elsif구문은 프로그램 상의 이해를 돕기는 좋으나 하드웨어 구현 시 Flip- Flop 소자가 더 많이 사용된다. 합성한 결과 if구문만을 사용했을 경우 게이트 수가 12개 줄어들었다. en_c와 en_r 신호를 발생시키기 위해서는 회로내부에서 카운터를

해 주는 변수가 있어야 한다. 이때 손쉽게 정수형 변수를 사용하기 쉬우나 최소한의 논리벡터(std_logic_vector) 변수를 사용하는 것이 필요 없는 셀 블록을 이용하여 합성하게 되는 것을 막고 하드웨어의 크기를 줄일 수 있다. 상태발생기블록을 합성할 경우 변수를 정수로 사용했을 때 라이브러리에 있는 add(b_st_DW01_inc_32) 셀 블록을 가져다 합성하고 전체 사용 게이트의 수가 512개인 반면, std_logic_vector(6 downto 0)를 사용해서 설계했을 경우 기존의 셀은 이용되지 않고, 전체 게이트의 수는 170개만을 사용하여 상태발생기 블록을 구현하였다.

## 3.2 입력블록

입력블록에는 in_reg 블록과 b_input 블록이 있다.

in_reg는 외부에서 주어진 입력패턴(x, y)을 출

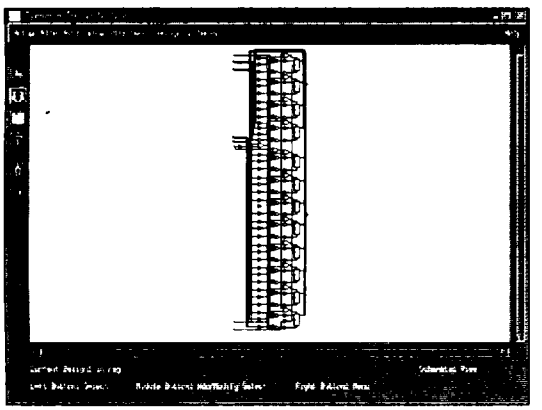

Fig. 5 Schematic diagram of Input register.

력하거나 기대되는 완전한 출력패턴을 얻기 전에 연산한 패턴(out_x, out_y)을 다시 입력으로 받아들여 출력하기 위한 블록이다. Fig. 5는 게이트레벨로 합성된 회로이다.

출력신호 ren_r은 비교기의 출력신호로 신호가 '0'이면 더 이상의 연상할 필요가 없으므로 버퍼가 출력을 내보내지 않고, '1'이면 out_x와 out_y가 in_x, in_y로 출력된다.

Fig. 6은 in_reg 블록을 게이트레벨 시뮬레이션한 결과이다.

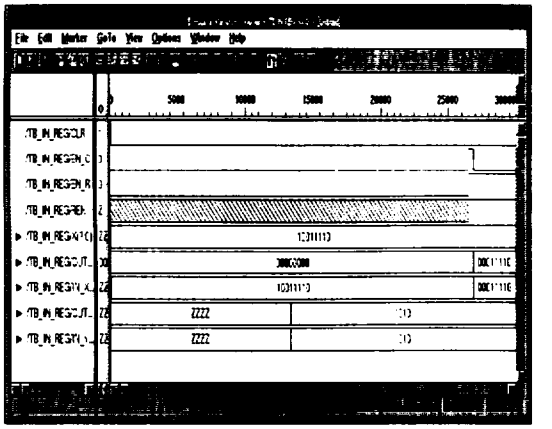

Fig. 6 Output of In_reg.

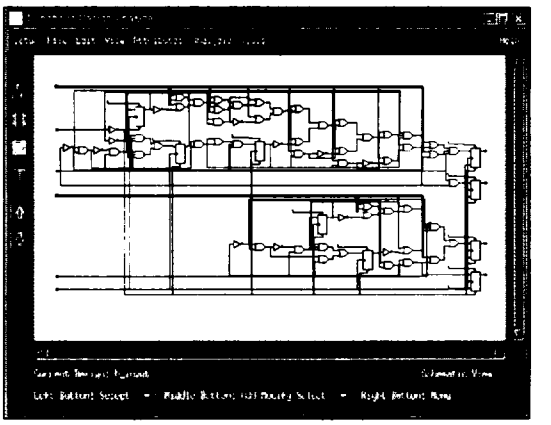

Fig. 7 Schematic diagram of B_input.

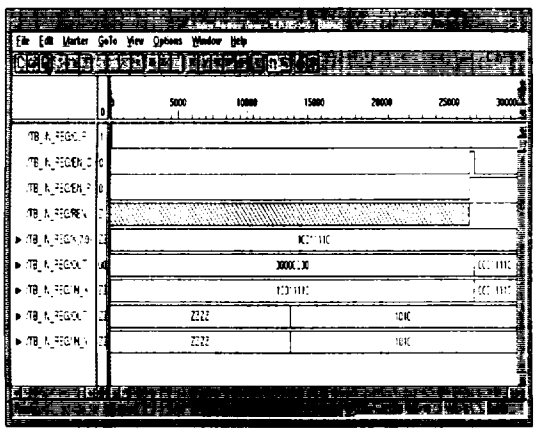

Fig. 8 Output of b_input.

B_INPUT은 입력패턴과 기억된 연결강도의 백터곱을 위하여 입력패턴을 뉴론(1비트)단위로 출력(o_x, o_y)하고, 연상작용으로 출력패턴을 구하기 위한 입력패턴과 연결강도의 곱에 따른 행과 열의 구분을 위한 신호(x_t, y_t)를 내보내는 블록이다.

Fig. 7과 Fig. 8은 B_input 블록의 게이트레벨 회로도와 시뮬레이션 결과를 나타낸 그림이다.

### 3.3 기억블록

계산된 연결강도의 값을 저장한 b_rom과 연결강도와 입력패턴의 곱을 위하여 b_rom에 저장된 내용을 불러오게 하는 주소를 출력(addr)하는 b_ad가 있다.

Fig. 9는 b_rom 블록의 게이트레벨 회로도이다.

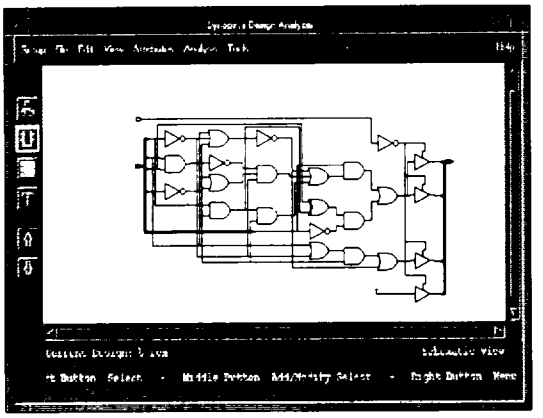

Fig. 9 Schematic diagram of b_rom.

b_rom은 계산된 연결강도의 값을 저장하였다가 b_ad블록의 주소값에 따라 저장된 값을 출력(data)한다.

세 개의 대표패턴을 가지고 연결강도를 구하는 식(6)을 이용해서 저장될 패턴 p의 개수가 3인 연결강도W를 구하면 다음과 같다.

X패턴의 요소의 개수 n=8, Y패턴의 요소의 개수 m=4이므로 연결강도는 크기가 8행 4열인 행렬인데 ROM으로 저장할 때는 열을 따라 각 요소들을 저장한다.

ROM의 크기는 32×4bit이다.

$$W=\begin{bmatrix} 3 & 1 & -1 & -3 \\ 1 & 3 & -3 & -1 \\ -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ 3 & 1 & -1 & -3 \\ -1 & -3 & 3 & 1 \\ -1 & -3 & 3 & 1 \\ -1 & 1 & -1 & 1 \end{bmatrix} \Rightarrow$$

| 주소 | 내용 |
|---|---|
| 00000 | 0011 |
| 00001 | 0001 |
| 00010 | 1111 |
| ⋮ | ⋮ |
| 11111 | 0001 |

Fig. 10은 롬블록의 게이트레벨 시뮬레이션이다.

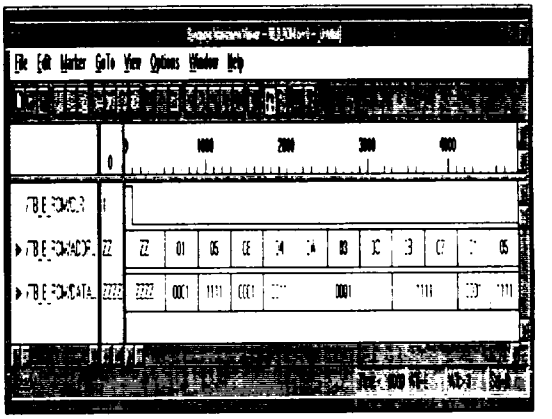

Fig. 10 Output of bam_rom.

b_ad는 ROM에 저장된 연결강도를 입력패턴과의 벡터곱이 가능하도록 내용을 출력하기 위한 블록이다.

Fig. 11은 b_ad 블록의 게이트레벨 회로도이다.

출력신호 addr은 en_x가 '1'일 때 X패턴을 이용하

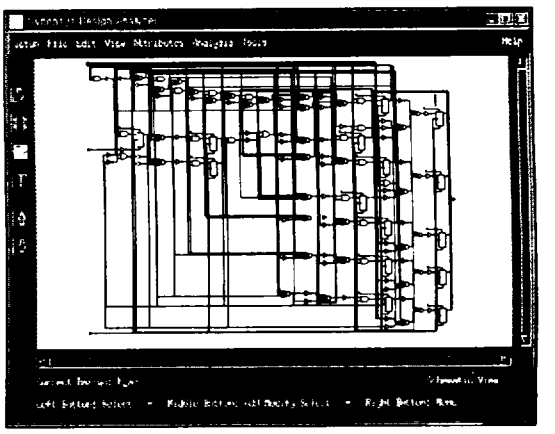

Fig. 11 Schematic diagram of b_ad.

여 출력패턴 Y를 구하는 것( $Y = WX$)으로 연결강도의 열을 이용하여 행렬 곱을 하므로 출력은 clk가 동작할 때마다 1씩 증가하고, en_y가 '1'일 때 출력패턴 X를 구하는 것( $X = W^T Y$)으로 연결강도의 행을 이용하여 행렬 곱을 하므로 출력 addr은 행의 곱을 할 때는 clk가 동작할 때마다 8씩 증가하고 열의 이동을 위해 4씩 증가한다.

Fig. 12는 Bam_add 블록의 게이트레벨 시뮬레이션 결과이다.

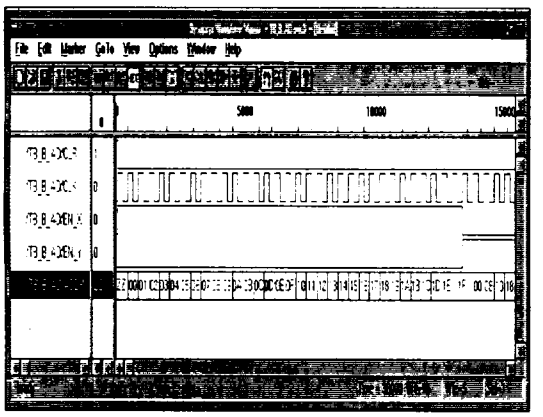

Fig. 12 Output of Bam_add.

## 3.4 연산블록

저장된 패턴을 연상해 내기 위하여 입력패턴(in_x, in_y)과 저장된 연결강도와의 벡터곱을 하는

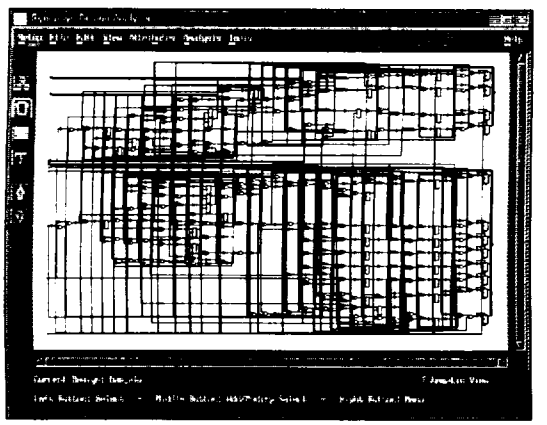

Fig. 13 Schematic diagram of bam_alu.

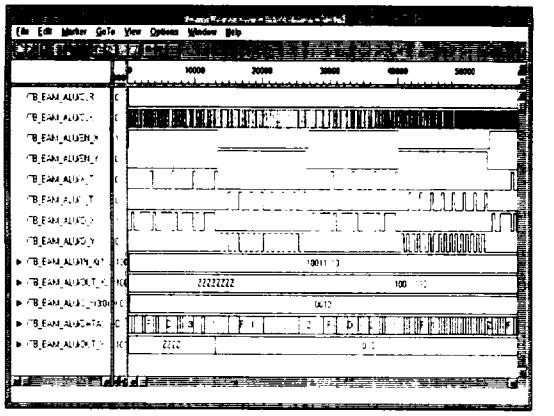

Fig. 14 Output of bam_alu.

블록으로 벡터곱을 하는데는 요소간의 곱셈과 덧셈이 필요하지만 입력패턴이 {0, 1}로 표현되므로 입력이 '1'이면 ROM에 기억된 데이터를 읽어와 더하면 벡터곱이 가능하다. 식 (27)을 이용하여 연상작용을 한다.

Fig. 13과 Fig. 14는 연산블록의 게이트레벨 회로도와 시뮬레이션 결과를 보여준다.

## 3.5 출력블록

출력블록에는 비교기블록과 출력기블록이 있다.

Fig. 15는 b_comp 블록의 게이트레벨 회로도이다.

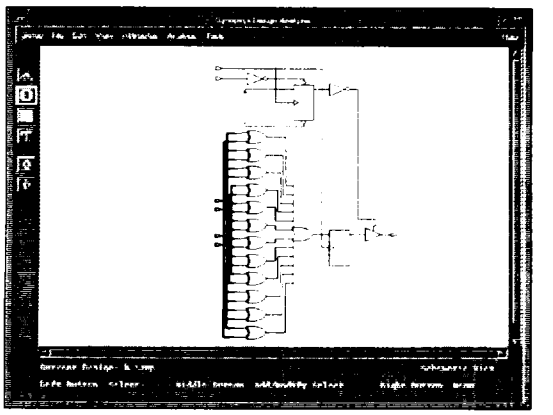

Fig. 15 Schematic diagram of b_comp.

위의 Fig. 15에서 b_comp(비교기)는 연산블록의 입력패턴(한 주기 전의 출력패턴 in_x, in_y)과 연산과정을 거친 출력패턴(out_x, out_y)을 비교하는

블록으로 출력패턴의 변화가 없으면 완전한 연상을 한 것이므로 출력기가 동작할 수 있는 신호(ren_r)를 출력한다.

Fig. 16은 b_comp 블록의 게이트레벨 시뮬레이션 결과이다.

출력기(out_reg)는 비교기의 출력(ren_r)이 '0'이면 입력패턴 out_x와 out_y를 출력(A, B)한다.

Fig. 17과 Fig. 18은 출력기블록의 게이트레벨 회로도와 시뮬레이션 결과이다.

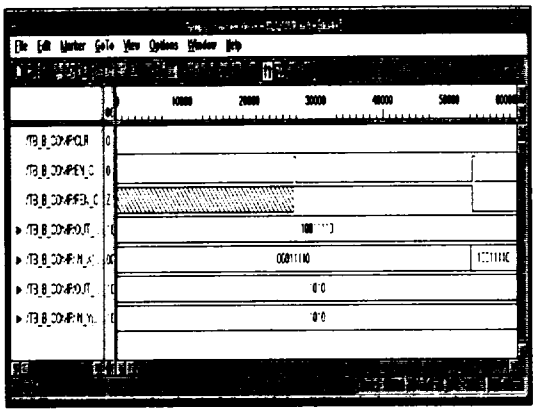

Fig. 16 Output of b_comp.

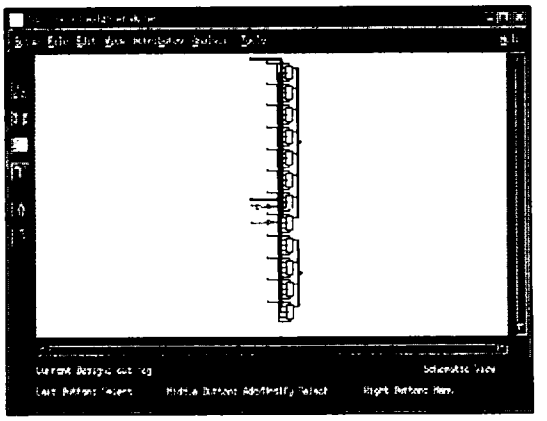

Fig. 17 Schematic diagram of out_reg.

## Ⅳ. 시뮬레이션 결과

양방향 연상 기억장치는 두 개의 층으로 이루어져

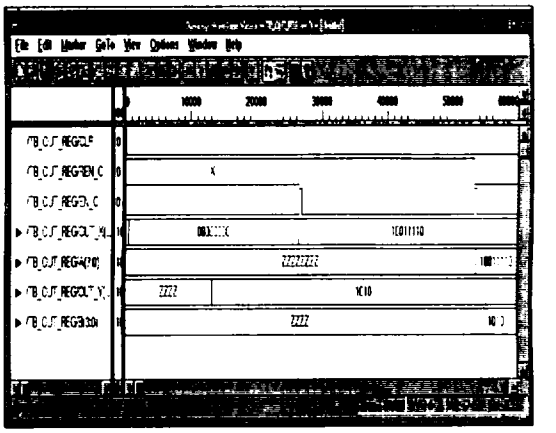

Fig. 18 Output of out_reg.

있는데 비슷한 환경에서는 각 층의 뉴론의 개수가 많을수록 더 정확한 연상을 할 수 있다. 그러나 하드웨어로 구현하는 데는 구현할 소자의 크기에 따라 한계가 있고, 확장성이 있는 블록설계를 FPGA로 간단히 제작하기 위해서 8개의 뉴론을 가진 X층과 4개의 뉴론을 가진 Y층으로 이루어진 시스템을 구성하였다.

앞 절에서 저장될 패턴의 수는 두 개의 층의 뉴론 중에서 작은 수의 뉴론에 영향을 받는다 ($p < \min(n, m)$)는 것을 알았다. Y층의 뉴론의 개수가 4개이므로 안정된 연상을 위해서는 4개의 패턴을 저장하고, 좀더 안정된 패턴의 연상을 위해서는 2개의 패턴을 저장하는 것이 좋은데, 3개의 패턴을 이용하여 양방향 연상 기억장치를 설계했다. X층과 Y층의 뉴론의 개수를 증가시키면 저장할 수 있는 대표패턴의 수는 더욱 커진다.

연결강도를 구하기 위한 X층과 Y층의 세 개의 대표패턴은 아래와 같고, Fig. 19는 대표패턴의 뉴론이 '0'일 때 흰 사각형, '1'일 때 검은 사각형인 그림으로 표현한 것이다.

$$X^{(1)} = \{1\ 0\ 0\ 1\ 1\ 1\ 1\ 0\} \quad Y^{(1)} = \{1\ 0\ 1\ 0\}$$
$$X^{(2)} = \{0\ 0\ 1\ 0\ 0\ 1\ 1\ 1\} \quad Y^{(2)} = \{0\ 0\ 1\ 1\}$$
$$X^{(3)} = \{1\ 1\ 1\ 0\ 1\ 0\ 0\ 1\} \quad Y^{(3)} = \{1\ 1\ 0\ 0\}$$

먼저 시스템이 제대로 동작하고 있는지를 살펴보기 위하여 왜곡이 없는 세 개의 대표패턴을 입력패턴으로 하여 양방향 신경회로망을 시뮬레이션 한 결과

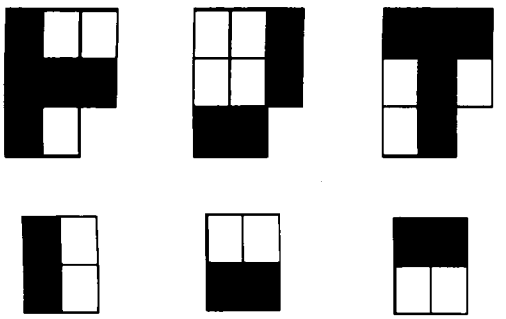

Fig. 19 Input patterns.

출력은 정확한 값을 가짐을 확인하였다.

다음은 왜곡이 있는 경우의 출력을 보여준다.

Fig. 20은 X 입력패턴에 하나의 왜곡이 있을 때 양방향 신경회로망의 출력이다.

$X^{(1)}$(1 0 0 1 1 1 1 0 ) ->X = (1 1 0 1 1 1 1 0)
Y = (0 0 0 0)

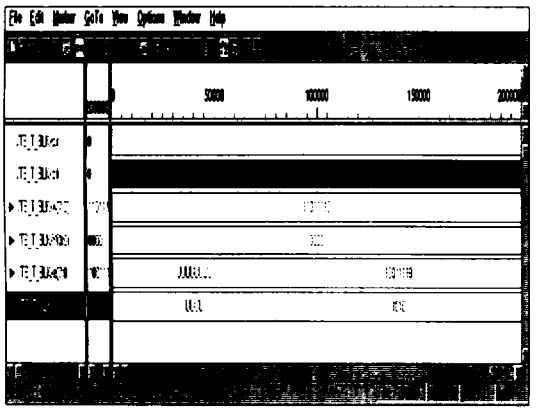

Fig. 20 Output of total block using a $x^{(1)}$ input pattern.

Fig. 21과 Fig. 22는 X 입력패턴에 두개의 왜곡이 있을 때 양방향 신경회로망의 출력은 다음과 같다.

$X^{(1)}$(1 0 0 1 1 1 1 0 ) ->X = (1 1 1 1 1 1 1 0 )
$X^{(2)}$ =(0 0 1 0 0 1 1 1)->X = (1 1 1 0 0 1 1 1)
Y = ( 0 0 0 0 )

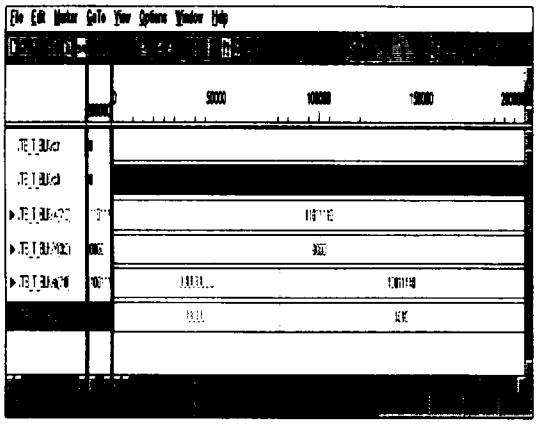

Fig. 21 Output of total block using a $x^{(1)}$ pattern with noise.

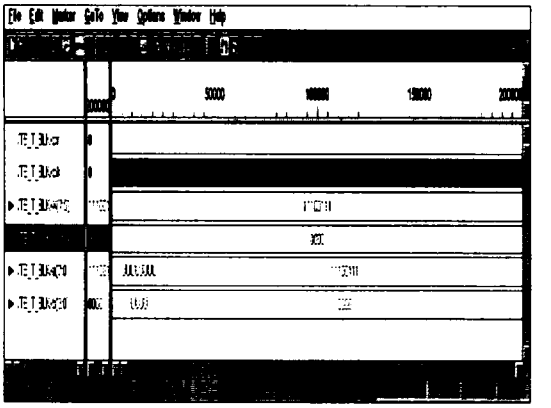

Fig. 22 Output of total block using a $x^{(2)}$ pattern with noise.

$X^{(2)}$ = (0 0 1 0 0 1 1 1)-> X=(1 1 1 0 0 1 1 1)
Y = ( 0 0 1 1 )

Fig. 22의 결과를 살펴보면 오류가 있는 입력패턴이 저장된 대표패턴을 찾아내지 못함을 보여주고 있다. 이것은 연상해내는 복호화 과정에서 구하려는 뉴론의 벡터합의 값이 0이 되었을 경우 연상되기 전의 값을 갖기 때문이다. 예를 들어 설명하면 오류가 있는 X패턴이 입력되었을 때 구해진 값이 ( 0 4 −4 0 )이면 Y′값은( 0 1 0 0 )이 아니라 ( $y_1$ 1 0 $y_4$ )이 된다. Fig. 23의 패턴의 경우 Y′값은

{ $y_1\ y_2\ y_3\ y_4$ }인 경우가 되고, Y패턴을 { 0 0 0 0 }로 입력했기 때문에 Y'값은 { 0 0 0 0 }이 되어 정확히 연상하지 못한다. 이러한 경우 Y패턴을 오류가 없는 값으로 입력하면 Fig. 23과 같이 저장된 대표패턴을 정확히 연상해 내는 것을 알 수 있다.

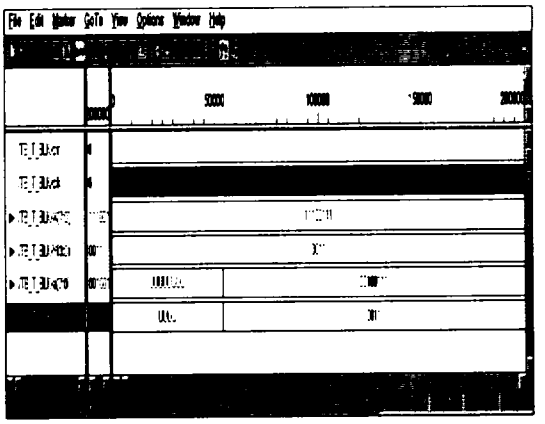

Fig. 23 Output of total block using a $x^{(2)}$ pattern with noise.

## V. 결론

본 논문에서는 디지털 신경회로망인 양방향 연상 기억장치를 하드웨어로 구현하기 위하여 ASIC 설계하였다. 양방향 연상 기억장치는 VHDL을 이용하여 다섯 개의 블록으로 설계하였는데 각각의 블록들은 구조적 시뮬레이션을 통하여 검증하였다. 검증된 블록들을 전체 합성하고 게이트레벨로 시뮬레이션하여 재검증하였다. 이러한 과정을 통하여 각 블록들은 개별적으로 유사한 동작을 하는 다른 시스템에서 이용될 수 있고, 더 큰 시스템으로의 확장도 쉽게 이루어 질 수 있다.

앞으로 연구해야 할 점은 성능개선된 Hopfield 아날로그 신경회로망을 ASIC 설계기술을 이용하여 구현하고, 병렬성이 크며 부호화가 가능한 양방향 신경회로망을 구현하고, 다양한 디지털 시스템들을 ASIC 설계기술로 구현하는 것 등이다.

## 참고문헌


1) Jacek M. Zurada. 1992. "Introduction to Artificial Neural Systems." West Publishing Company.

2) 최 명 렬. 1996. "주문형 반도체 설계 ASIC DESIGN" 하이테크정보.

3) Bart Kosko. 1988. "Bidirectional Associative Memories" IEEE. pp. 49-60.

4) 조 중 휘, 홍 윤 식, 정 연 모, 김 영 철. 1996. "VHDL 기초와 응용." 반도체설계교육센터.

5) Roger Lipsett, Carl F. Schaefer, Cary Ussery. 1993. "VHDL : Hardware Description and Design." Intermetrics Inc.

6) ALTERA. 1992. "MAX+PLUS II Getting started." ALTERA corporation.

7) SYNOPSYS. "Design Analyzer Reference" Synopsys Inc.