초직접회로를 위한 신경망 연구

이윤구, 정홍 (포항공과대학교 미디어랩)

1. 연구 목적 및 필요성

일반적으로 많이 활용되는 신경망에서 계산량이 많은 부분을 하드웨어로 구현함으로써 전체적인 계산 시간을 단축시키려 한다. 신경망에서 각 node들이 다음 layer로의 connection들을 곱해서 더해주고 시그모이드 값을 발생시키는 작업은 반복적이고 많은 계산량을 요구하므로 이 부분을 하드웨어로 만들 경우 속도를 증가시킬 수 있다고 판단된다.

사실 여러 가지의 신경망이나 패턴 분류를 위한 신호처리를 위해서 공통적으로 필요로 하는 연산이 바로 이 부분이다. 전체적인 계산 상 속도 결정단계인 이 부분을 하드웨어로 구현한다면 상당한 도움이 될 것이다. 그리고 현재 계속해서 발전하고 있는 FPGA 기술은 현재까지는 상당한 하드웨어를 필요로 해서 실용성이 그리 크지 않는 이런 분야에서의 칩 계발을 부추기고 있다.

현재 계발중인 칩은 반응시간이 빠른데다 고속 실시간 병렬처리에 적합한 구조로 되어 있다. 사실 계속 이어 붙이기만 하면 되는 구조로 되어 있기 때문에 필요에 따라서 계속 확장이 가능하다. 곧 앞으로 사용빈도가 증가될 신경망이나 패턴 인식 신호처리에서 필수적으로 사용하게 될 계산에 적합한 특성을 지니고 있다.

2. 연구내용 및 방법

기본적으로 신경망에서 곱과 합과 LUT를 하드웨어로 구현한다. 곧 A*B(n) (A: node의 값으로 이뤄진 행렬, B(n): connection value로 이뤄진 행렬, *: 내적)을 구하고 LUT를 읽는 것이다. 새로운 구조로서 비트분리구조의 누산기를 제안한다. 이에따라 A*B(n)의 값이 행렬의 크기에 상관없이 단지 신호의 비트크기 만큼에 해당하는 클럭 수 만에 구해진다. 곧 노드의 수에 의해서 생기는 계산량의 증가가 없게 된다. 이는 비트분리구조의 누산방식이 전체를 자리수 비트별로 곱해서 더해주는 특성을 가지고 있기 때문이다.

먼저 비트분리구조의 누산기를 기존의 연구들을 바탕으로 해서 VHDL을 이용한 하드웨어로 구현하고 이를 컨트롤하는 부분을 만들게 된다. 그렇게 해서 만들게 되는 하드웨어의 기본적인 동작 내용은 다음과 같다. 누산기에 먼저 계수로서 노드의 값들을 집어 넣는다. 그리고 shift register에 이 노드들에 곱하게 될 값들을 로드한다. 로드가 끝나게 되면 자리수의 개수만큼의 클럭동안 이 shift register를 움직이면서 누산을 시작해서 다음 노드의 처음 값을 얻게 된다. 이를 G function을 거치고 나서 하위의 shift register에 저장한다. 그리고 나서 다시 처음에 말한 shift register에 다음 connection을 저장하고 계산해서 마찬가지로 G function을 거친 후에 하위의 register에 저장한다.

이런 방식으로 계속 진행해서 하위의 register를 다음 노드의 값들로 모두 채운 후에 그 값을 컴퓨터로 로드하게 된다.

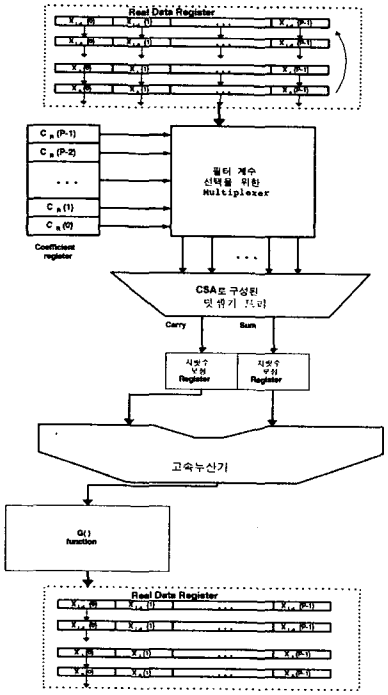
그런데 이 때 마지막으로 얻은 하위의 shift register를 다음에 계수들로 사용하고 다음에 다시 connection값들을 받아들인 후에 계산을 진행할 수도 있다. 곧 하드웨어에 꼭 필요한 최소의 정보를 로드해서 원하는 마지막 노드에서의 값을 얻을 수도 있다. forward process의 결과를 최소의 data로드로서 구할 수 있는 것이다.

3. 구체적인 연구 수행내용

이 시스템은 신경망 에뮬레이터 소프트웨어, PCI 드라이버, 신경망칩으로 구성된다. 신경망칩은 신경의 기본계산을 하며 PCI드라이버는 신호를 이 칩에 공급하고 결과를 넘겨 받는다. 에뮬레이터는 신호를 신경망칩이 처리할 수 있도록 가공해 공급하고 결과를 받아 처리한다. 아울러 신경망칩을 이용해 MLP, recurrent net, Hopfield net 등 여러 종류의 신경망의 기능을 갖도록 한다.

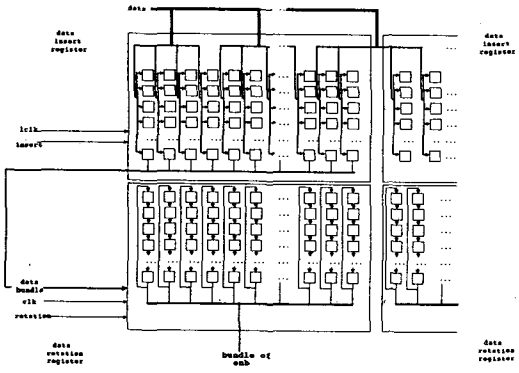
신경망 칩은 시프트레지스터, 연산부, 누산기, LUT, 제어부, PCI 통신부로 구성되어 있다. 시프트레지스터는 입력신호와 신경망 연결강도를 저장하고 있고 연산부는 CSA를 이용한 고속 덧셈기이며 누산기는 이 결과를 바복적으로 더하거 LUT에서는 시그모이드값을 읽어낸다. PCI부는 이 칩을 PCI Bus와 연결하며 제어부는 이 모든 시스템을 통솔하고 제어한다.

먼저 지금 까지 만들어진 전체 하드웨어를 도시하고 이에 대한 개략적 설명을 한 후 중요한 부분에 대해 설명하도록 하겠다.



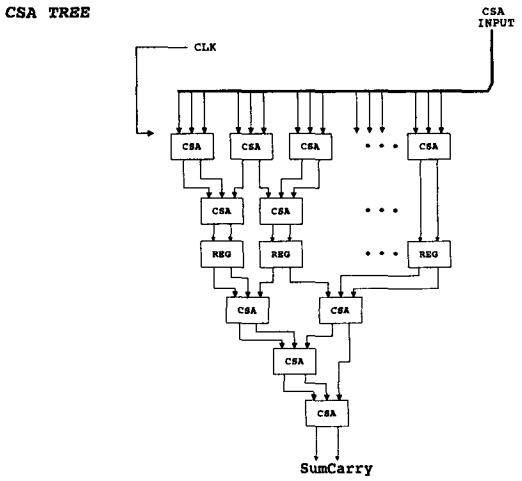
<전체적인 하드웨어 구조>

위에서 coefficient_register는 계수를 채우는 곳으로 먼저 control 부분에서 PCI에서 오는 정보를 이 부분에 채우게 된다. 그리고 나서 상단의 real_registerd에 값을 채우게 되는데 채우고 나서 계산이 끝나면 하단의 registerd에 sigmmoid를 통과한 값을 채우게 된다. 일단 채우고 나면 다시 상단의 register를 채우게 되고 다시 계산을 한후에 sigmoid를 거치고 하단의 reister를 채우게 된다. 이런 식으로 하단의 register를 전부 채우게 되면 이를 PIC에 보내게 되고 컴에서 이를 읽게 된다. 이런 일련의 순차적인 과정들을 제어하는 control 부분이 있다.control 부분은 계수 register를 제어하는 부분, 상단의 register를 제어하는 부분, 연산부분을 제어하는 부분, 하위의 register를 제어하는 부분, PCI에 data를 싣는 것을 제어하는 부분이 있고 이 전체를 제어하는 Core부분이 있다. 이들의 동작은 상기에 설명한 것과 같다.



<상단의 register>

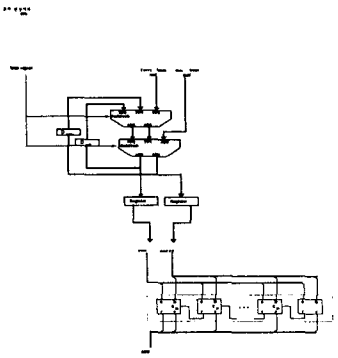
계수 부분과는 달리 두개의 레지스터를 사용하였다. 먼저 상단의 레지스터는 PCI에서 오는 값을 한 PCI 클럭에 4개씩 채운다. 이렇게 해서 전체를 채우면 하단의 레지스터에 이 값들을 그대로 로드한다. 이렇게 로드된 data는 계산부분에 적용되는 클럭에 동기되어 한 클럭에 하나씩 회전하게 된다. 굳이 많은 용량을 들인 것은 상이한 클럭에 동기된 두 시스템에서 발생할 수 있는 시간적 지연을 방지하기 위함이다.



<CSA TREE>

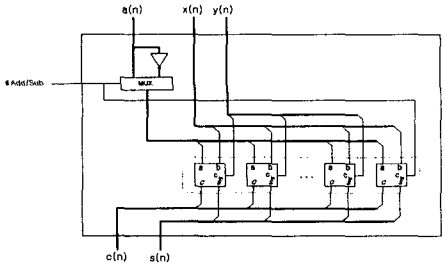
상기는 CSA tree로써 이를 사용해서 tree를 구성하여 sum과 carry를 내보낸다. 그런데 비트별로 값을 합산하기 때문에 누산기에서 누산하기 전에 자리수를 보정해 주어야 한다. 곧 이 sum과 carry는 위의 전체 그림에서 볼 수 있듯이 자리수 보정 래지스터를 거치게 된다.

자리수 보정 래지스터에서는 자리수 별로 (counter를 이용) 값을 보정한다. 이렇게 해서 보정된 값은 다음의 누산기에 들어가게 된다.



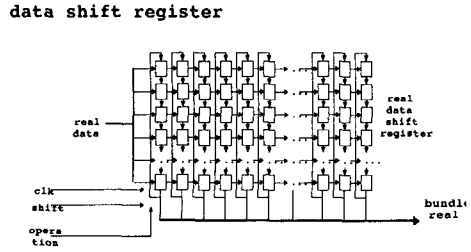
<고속 누산기와 CPA>

먼저 덧셈기 Treed에서 오는 값들을 위의 누산기에서 누산시킨다. 처음 값이 들어오면 0과 함께 들어오는 값을 합산해서 내보내는데 이렇게 합산된 값은 다음 값이 들어오면 그때 그 값과 함께 합산되게 된다. 그러다가 마지막에 최상위 비트들의 합이 들어오면 그때는 각 비트들을 inverse 시켜서 더해주는데 이는 2의 보수의 특성으로 인한 것이다.



<고속 연산기 내부>

앞에서 설면한 고속 연산기의 내부 모습이다. 위에서 볼 수 있듯이 ADD/SUB명령을 8번째 합산 시에 줌으로써 2의 보수의 합을 구하게 된다. 이렇게 해서 나온 값은 바로 CPA에 들어가서 값을 내보낸다.

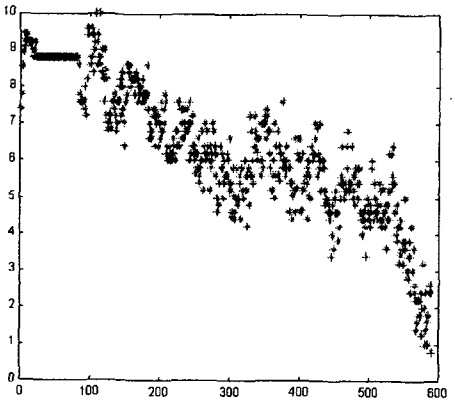


<하단의 data_shift_register>

이것이 data shift register로써 위에서 합산된 결과가 이쪽으로 하나씩 들어오게 된다. 필요한 만큼 연산이 끝나면 컴퓨터에서 PCI를 통해 이쪽의 결과를 읽게 된다.

4. 구현된 칩에 의한 실험결과

간단한 문자인식을 칩을 사용해서 구현해 보았다. 10개의 알파벳을 인식하는 다층 펄셉트론에서 입력에 대해서 각 뉴론의 값들을 구하는 계산과정을 칩을 사용해서 구했는데 계수로는 입력값을 넣고 위에 보이는 레지스터에는 connection weight를 넣었다. 한번 계수레지스터를 채운 다음에 상단에 보이는 레지스터를 다음 layer로 넘어 가기 전까지 계속해서 채움으로써 output 값들을 구해 내었다. 이렇게 구해진 output들은 칩 내부의 shift 레지스터에 저장되었다가 PCI를 통해서 읽어 들인다.



10개의 알파벳 각각은 8*8로 이뤄진 -1과 1의 조합으로 표현되었으며 곧 input은 64개의 노드로 구성된다. 은닉 layer는 하나고 44개의 펄셉트론으로 구성된다. output은 8개의 펄셉트론을 가지고 있다. 시그모이드 함수는 하드웨어 내부에 있는 LUT로써 대체되었는데 −1.27과 1.27 사이를 26개의 구간으로 나눈 값들을 사용하였다.

결과적으로 옆에 볼 수 있는 학습 그래프를 얻었다. 이 학습 graph는 iteration에 따른 error의 감소를 나타내고 있다.

5. 연구결과에 대한 기대효과 및 활용방안

신경망칩을 구현함에 있어 비트 레벨의 누산기 라는 개념을 사용함으로써 보다 빠른 동작을 기대할 수 있다. 또한 이런 누산기를 여러 개 사용할 경우 손쉽게 빠른 병렬처리를 구현할 수 있는데 그것은 누산기가 노드가 몇 개 이던지 간에 항상 같은 클럭 만에 다음 노드의 값을 도출하므로, 여러 개의 누산기에서 일정한 시간에 다음 값을 내보내고 이렇게 나온 값들이 connection값들과 함께 마찬가지로 계산되기 때문이다.

곧 비전시스템이나 음성인식과 같이 실시간 처리가 요구되는 분야에서의 활용이 기대된다.

6. references

[1]Hong Jeong. Modeling and transformation of systolic network. Master's thesis, Massachusetts Institute of Technology, 1984

[2]Chales E. Leiserson, Flavio M. Rose, and Jaimes B. Saxe. Digital cicuit optimization. VLSI Memo No. 82-100, Apr. 1982

[3]T. J. Robnett and B. Charny. A high-speed systolic apeture radar processor. In IEEE International Conference on Acoustics, Speech, and Signal Processing, volume 4 , page 357-360, Feb. 1992.