論 文

대형 스파스 메트릭스 선형방정식을 효율적으로 해석하는 씨스톨릭 어레이

正會員 李 秉 洪* 正會員 蔡 洙 煥** 正會員 金 正 善***

A Systolic Array to Effectively Solve Large Sparce Matrix Linear System of Equations

Byung Hong Lee*, Soo Hoan Chae*, Jung Sun Kim*** *Regular Members*

要 約

스트라이프 구조를 이용하여 대형 스파스 선형 방정식을 해석하는 CGM 반복 씨스톨릭 알고리즘을 제시하고, 이를 씨스톨릭 어레이로 구현했다. 메트릭스 A를 상삼각 행렬과 대각선 행렬, 하삼각 행렬로 나누어서 이들이 별개의 선형 어레이에 의해서 병행적으로 실행되도록 했다. 따라서 1개의 선형 어레이를 사용했을 때보다도 실행시간이 대략 1/2로 단축되며, 스트라이프 구조를 이용하므로서 불규칙하게 분포된 스파스 메트릭스의 연산을 효율적으로 할 수 있다.

ABSTRACT

A CGM iterative systolic algorithm to solve large sparse linear systems of equations is presented. For implementation of the algorithm, a systolic array using the stripe structure is proposed. The matrix A is decomposed into a strictly lower triangular matrix, a diagonal matrix, and a strictly upper triangular matrix, and the two formers and the latter are concurrently computed by different linear arrays. Hence, the execution time of this approach is reduced to half of the execution time of the case that a linear array is used. computation of the irregularly distributed sparse matrix can be executed effectively by using the stripe structure.

I. 서 론

선형 방정식을 해석하는 일은 많은 응용 분야에서 특히 디지탈 신호처리에 있어서 매우 중요한 것이다. 많은 문제들을 효율적으로 고속처리하기 위해서는 병렬로 동작하는 많은 프로세서들이 요구된다. 제한된 칩 면적에 멀티-프로세서 회로를 구성하기 위해서는 하나 또는 소수의 타스크(task)만을 처리하는 기본적인 프로세서들로 구성된 특수 목적의 프로세서

*烏山專門大學
**韓國航空大學 電子計算學科
***韓國航空大學 電子工學科
論文番號 : 92-74(接受1992. 3. 14)

어레이가 필요하다. 만약에 이들 프로세서 어레이들이 지역적(locally), 규칙적(regularly)으로 연결되고 몇 가지 서로 다른 형태의 기본적인 프로세서 셀(cell)들로 구성된다면 이들 프로세서 어레이들을 씨스톨릭/웨이브프론트 어레이(systolic/wavefront array)라고 한다. 이러한 구조는 특히 VLSI 구현에 적합하다.

선형 방정식을 해석하는 알고리즘으로서는 크게 나누면 직접법과 반복법이 있다. 직접법의 경우에는 소거과정에서 스파스 메트릭스(sparse matrix)가 필인(fill-in)으로 말미암아 덴스 메트릭스(dense matrix)로 되어 씨스톨릭 어레이의 면적 복잡도(area complexity)가 증가한다. 예를 들어 방정식의 수를 n이라고 하면 면적 복잡도는 $O(n^2)$으로 된다.[1] 따라서 제한된 칩 면적을 고려해서 대형의 문제는 적절한 분할 방법을 사용해야 한다. 이에 반해서 반복법은 전체적인 시간 복잡도(time complexity)는 증가하지만 nonzero 요소의 수를 e라고 하면 O(e)로서 면적 복잡도는 적으므로 대형 스파스 메트릭스의 경우에는 직접법 보다 반복법을 사용하는 것이 좋다.[15,16]

반복법 중에는 M.R.Hestens & E.Stiefel이 제시한 CGM(Method of Conjugate Gradient) 알고리즘이 있는 데, 이는 n개의 선형 방정식을 n스텝으로 해석하는 수렴 속도가 가장 빠른 반복 알고리즘으로서, Gauss 소거법을 포함한 일반적인 방법의 특수한 경우이며, J.Gôtze & U.Schwiegelshohn은 2차원의 씨스톨릭 쇼팅 프로시저(sorting procedure)에 의하여 이 알고리즘을 씨스톨릭하게 실행하는 2차원의 씨스톨릭어레이를 구성하여 면적 복잡도를 O(e)로, 시간 복잡도를 $O(n\sqrt{e})$로 되게 하였다.[1,2]

알고리즘을 씨스톨릭 어레이로 사상(mapping)하는 일반적인 단계는 우선 DG(Dependence Graph)를 생성하고 이 DG를 SFG(Signal Flow Graph)로 사상한다. 그리고 이 SFG로부터 씨스톨릭 어레이를 구성한다. 이러한 일반적인 단계에 있어서 M.T. O'keefe & J.A.B.Fortes는 Li & Wah의 패러미터 방법과 Moldovan & Forters의 데이타 종속(data dependency) 방법 사이에 관계를 검토하고 각 방법의 모델과 패러미터, 그들사이의 수학적인 관계를 유도했다.[3] 그리고 Y.T.Hwang & Y.H.Hu는 새로운 씨스톨릭 프로시저로서 PDG(Parameterized Depedence Graph)를 제시하고, 이것을 이용해서 멀티-스테이지(multi-stage) 알고리즘을 개발했다.[4],[9]

일반적으로 병렬알고리즘을 표현하기 위해서 그래프를 이용하면 편리하며, 병렬성은 문제를 병렬로 실행할 수 있는 독립된 서브타스크로 나누든가 파이프라인 방식으로 실행될 수 있는 종속적인 서브타스크로 분할해서 얻어질수 있다.[12],[13],[14] J.J. Navarro & J.M.Llaberia등은 메트릭스의 크기에 제한을 받지않고 효율적으로 밴드 메트릭스를 비롯하여 여러가지 메트릭스를 실행할 수 있도록 분할하는 자료구조와 변형방식을 제안했으며,[5] W.Luk은 4가지 종류의 씨스톨릭 밴드 메트릭스 곱셈기를 설명하고 파이프라인을 하는 데 따라 포함되는 트레이드-오프(trade-offs)를 제시했다.[6],[7],[8]

앞에서 기술한 바와 같이 스파스 메트릭스는 소거법을 사용하는 경우에는 소거과정에서 덴스 메트릭스로 되기 때문에 스파스 특성을 효율적으로 이용할 수 없다. 그래서 스파스 메트릭스의 경우에는 반복법을 사용하게 되는 데 R.Melhm은 스파스 특성을 보유하는 스트라이프 구조(stripe structure)를 제안하여, 씨스톨릭 어레이의 응용이 규칙적인 계산에만 한정되지 않고 스파스 메트릭스와 같은 불규칙한 문제들이 씨스톨릭 네트워크를 통해서 해석될 수 있음을 보여 주었다.[10],[11] R.Melhem이 제안한 스트라이프 구조는 불규칙하게 분포된 스파스 메트릭스를 그것의 본래의 형태대로 처리할 수 있을 뿐만아니라 프로세서 셀의 수도 줄일 수 있으므로 스파스 메트릭스의 VLSI 구현에 적합하다.

본 논문에서는 스트라이프 구조(stripe structure)의 메트릭스를 해석하며, CGM(Method of Conjugate Gradient) 알고리즘으로 부터 유도된 씨스톨릭 알고리즘과 2개의 선형 어레이를 사용하여 시간 복잡도를 개선하는 씨스톨릭 어레이를 제시한다.

R.Melhem은 스트라이프 구조의 매트릭스를 처리하기 위하여 하나의 선형 어레이를 사용하였으나 본 논문에서는 두개의 선형 어레이를 사용하여 병행적으로 실행되도록할 것이며, 스트라이프 구조의 메트릭스를 해석하기 위하여 CGM 알고리즘으로 부터 씨스톨릭 알고리즘을 유도할 것이다.

Ⅱ. 메트릭스 A의 스트라이프 구조

메트릭스 $A=\{a_{ij} : i, j=1,\ldots,n\}$의 스트라이프 S는 메트릭스 A의 각각의 행 i에서 하나의 원소 위치(i,

j)를 포함하는 원소 위치들의 한 집합으로서 다음과 같이 정의되며 모든 행이나 열에 대한 원소의 위치를 필연적으로 포함하는 것은 아니다. 1, m=1,2,...,n이라고하면,

$S=\{i, \sigma(i) ; i\in i_dom(s)$이고, $i<1$에 대해서 $\sigma(i)<\sigma(1)$ 이다.$\}$ (2-1)

또는

$S=\{(\mu(j),j ; j\in j_dom(s)$이고, $j<m$에 대해서 $\mu(j)<\mu(m)$ 이다.$\}$ (2-2)

여기서 i_dom(s)와 j_dom(s)는 {1,2,.....,n}이고, σ는 열인덱스(column index) $\sigma(i)$를 행 i에, μ는 행인덱스(row index) $\mu(j)$를 열 j에 각각 결합하는 증가함수이다. 식 2-1과 식 2-2로 부터 정의되는 스트라이프에는 모든 행이나 열의 원소 위치를 필연적으로 포함하는 것이 아니므로 모든 행이나 열의 원소 위치를 포함하도록 스트라이프를 확대할 수 있다. 이와 같이 확대된 스트라이프 구조를 S의 행 보수 $\bar{S}^r$(row complement) 또는 열 보수 $\bar{S}^c$(column complement)라고 하며 다음과 같이 정의한다.

$$\bar{S}^r=\{(i, \bar{\sigma}(i)) : i=1,.....,n\} \quad (2\text{-}3)$$

여기서 $i\in i\_dom(s)$ 이면 $\bar{\sigma}(i)=\sigma(i)$ 이고, i=2, 3,...,n에 대해서 $\bar{\sigma}(i)\geq\bar{\sigma}(i-1)$이다. 또

$$\bar{S}^c=\{(\bar{\mu}(j), j) ; j=1,.....,n\} \quad (2\text{-}4)$$

여기서 $j\in j_dom(S)$이면 $\bar{\mu}(j)=\mu(j)$이고 j=2, 3,...,n에 대해서 $\bar{\mu}(j)\geq\bar{\mu}(j-1)$이다.

지금 2개의 스트라이프 $S_1=\{i, \sigma_1(i)$와 $S_2=(i, \sigma_2(i)\}$가 주어졌을 때 $\sigma_1(i)<\sigma_2(i)$이면 $S_1<S_2$이다. A의 스트라이프 구조는 스트라이프들의 집합 $\sum_A=\{S_1,...,S_m\}$이다. 여기서 $S_1<S_2,...<S_m$이고, m은 $\sum_A$의 스트라이프 카운트이다. 그리고 $S_1\cup....\cup S_m$은 A에 있어서 nonzero 원소들의 모든 위치를 포함한다. 즉, $(i, j)\in S_1\cup....\cup S_m$이면 $a_{ij}=0$이다. 스트라이프 구조 $\sum_A$가 주어지면 $\sum_A$의 행과 열의 보수는 각각 $\bar{\sum}^r_A=\{\bar{S}^r_1,...,\bar{S}^r_m\}$과 $\bar{\sum}^c_A=\{\bar{S}^c_1,...,\bar{S}^c_m\}$으로 정의 된다.

그림 1은 메트릭스 A에 대한 스트라이프 구조의 한 예를 나타내며, 그림에서 "0"은 zero이고, "a_{ij}"는 nonzero를 표시한다. 그리고 스트라이프 S에 포함되는 원소 위치들은 원 내에 있다.

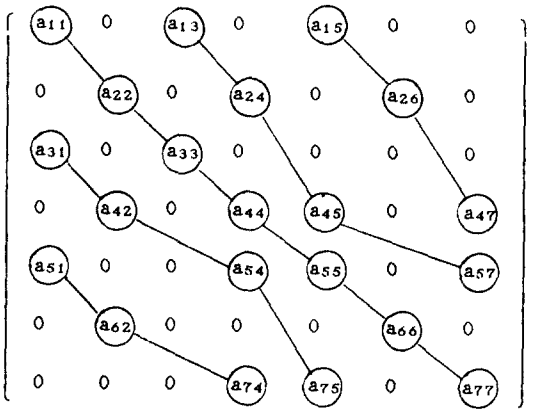

그림 1. 스트라이프 구조의 예
Fig. 1. an example of the stripe structure

Ⅲ. CGM 알고리즘

CGM은 직접법이나 반복법 어느 것으로도 사용할 수 있는 방법으로서 반복법 중에서는 가장 수렴 속도가 빠르며,[2] A x =b 형태의 대형 스파스 메트릭스 선형 방정식을 해석하는 가장 잘 알려진 반복법중에 하나이다. 여기서 A는 n x n 대칭, 정치정부호행렬(symmetric, positive definite matrix)이며, x와 b는 n x 1 벡타로서 x는 미지수, b는 기지수이다.

CGM은 n 스텝 정도로 종료되며, 알고리즘 실행중에 메트릭스의 형태가 변하지 않으므로 특히 스파스 메트릭스에 적합하다.[2] 직접법으로 사용될 경우에는 메트릭스 A가 행마다 e개의 nonzero 원소를 가지고 있다면 이론적으로 대략 $(e+5)n^2$의 곱셈이 필요하므로 Gauss 소거법에 비해서 많은 산술연산이 필요하다.[2],[17] 따라서 직접법보다는 주로 반복법이 사용되며, 이 알고리즘의 실행을 위해서는 메트릭스 A는 대칭이 되어야 하는 데 비대칭인 경우에는 $A=A^T$ (여기서 A^T는 A의 전치 메트릭스이다.)이면 A x =b를 A^TA x $=A^Tb$로 고쳐 쓸 수 있으므로 새로운 대칭 행렬 A^TA에 대해서 해석할 수 있다.

CGM 알고리즘은 Cholesky decompostion과 비교해보면 GCM은 안정조건 α_i/α_{i-1}의 비가 크면 클수

록 round-off 에러가 더 빨리 누적되며, $\rho=\lambda_{max}/\lambda_{min}$이 1에 가까우면 가까울수록 비교적 안정하다. 여기서 α는 스텝의 크기를 표시하고, λ_{max}, λ_{min}은 각각 메트릭스 A의 eigen-value로서 최대값과 최소값을 표시한다.[1] 한편 Cholesky decomposition은 항상 불안정하며, 수치적인 안정조건은 $g_{ij}^2 \le \sum_{p=1}^{j} g_{ip}^2 = a_{ii}$로 주어진다. g는 Cholesky triangle의 원소이다.[15] 그리고 Cholesky decomposition은 n개의 평방근의 값을 결정하는 데 필요한 연산의 수에 크게 좌우된다. 일반적으로 $n^3/6$ flops의 산술연산이 필요하다.[15] 이에 비하여 CGM은 하나의 메트릭스-벡타 곱셈과 매 반복마다 5 flops의 산술 연산으로 n 스텝 정도로 연산이 끝난다.[1]

다음은 CGM 알고리즘이다.[1,2]

〈알고리즘 1〉

$X_1=0$, $r_1=b$, $P_1=r_1$

For k=1 to n do

$W_k=A\,P_k$

$$\alpha_k=\frac{P_k^T\,r_k}{P_k^T\,W_k}$$

$X_{k+1}=X_K+\alpha_K\,P_K$

$r_{k+1}=r_k-\alpha_k\,W_k$

$$\beta_k=\frac{W_k^T\,r_{k+1}}{P_k^T\,W_k}$$

$P_{k+1}=r_{k+1}-\beta_k\,P_k$

여기서 P_k는 갱신(update)의 방향을 표시하는 방향 벡타이고, α_k는 스텝크기(stepsize)를 표시하는 스칼라량이다. 그리고 r_k는 잔여 벡타이다. 이 알고리즘은 메트릭스-벡타 곱셈 $W_k=A\,P_k$가 주 프로시저가 되며, W_k는 다음과 같은 내적(inner product)으로 계산된다.

$$W_k^i = a_i^T\,P_k = \sum_{j=1}^{n} a_{ij}\,P_k^j \qquad (2\text{-}5)$$

Ⅳ. 씨스톨릭 어레이와 알고리즘

3절에서 설명한 CGM 알고리즘으로 부터 씨스톨릭 알고리즘을 유도하고 씨스톨릭 어레이를 구성하기 위하여 다음과 같이 정의한다.

〈정의 1〉메트릭스 $A=\{a_{ij}; i,j=1,2,\ldots,n\}$의 스트라이프 S는 식 2-1과 2-2와 같이 정의하고, 스트라이프 구조 $\sum_A$는 $\{S_1,\ldots,S_{2m-1}\}$으로 한다.

〈정의 2〉메트릭스 A는 A_L+D+A_U로 분해한다. 여기서 A_L은 하삼각 행렬이고, A_U는 상삼각 행렬이다. 그리고 D는 대각선 행렬이다.

〈정의 3〉메트릭스 A는 $A=A^T$(여기서 A^T는 A의 전치행렬이다.)이면 대칭행렬이다. 이때 $A_U=A_L^T$ 여기서 A_L^T는 A_L의 전치행렬이다.)이다.

정의 1과 2로 부터 메트릭스 A는 다음과 같이 된다.

$$A=A_L+D+A_U$$
$$=(A_L+D)+A_L^T$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{bmatrix} = \begin{bmatrix} 0 & & & & \\ a_{21} & 0 & & & \\ a_{31} & a_{32} & 0 & & \\ a_{41} & a_{42} & a_{43} & 0 & \\ a_{51} & a_{52} & a_{53} & a_{54} & 0 \end{bmatrix} +$$

$$\begin{bmatrix} a_{11} & & & & \\ & a_{22} & & & \\ & & a_{33} & & \\ & & & a_{44} & \\ & & & & a_{55} \end{bmatrix} + \begin{bmatrix} 0 & a_{12} & a_{13} & a_{14} & a_{15} \\ & 0 & a_{23} & a_{24} & a_{25} \\ & & 0 & a_{34} & a_{35} \\ & & & 0 & a_{45} \\ & & & & 0 \end{bmatrix}$$

알고리즘 1과 정의 1,2,3으로 부터 시스톨릭 알고리즘은 다음과 같이 도출된다.

〈알고리즘 2〉

$X_0=0$, $r_0=b$, $P_0=r_0$

Repeat k=0 until $r_k=\varepsilon$(여기서 ε은 허용 오차)

1) 1.1) $W1_k=(A_L+D)*P_k$

1.2) $W2_k=A_U*P_k$

1.3) $W_k=W1_K+W2_k$

1.4) $\lambda_k=(r_k, r_k)$

1.5) $\alpha_k=(P_k, W_k)$

2) 2.1) $\alpha_k=\lambda_k/\alpha_k$

3)3.1) $X_{k+1}=X_k+\alpha_k * P_k$

3.2) $r_{k+1}=r_k-\alpha_k * W_k$

4)4.1) $\beta_k=\lambda_{k+1}/\lambda_k$

5)5.1) $P_{k+1}=r_{k+1}+\beta_k * P_k$

여기서 $(r_k, r_k)=r_1^2+....+r_n^2$이고, $(P_k, W_k)=\sum_{i,j=1}^{n} P_i * a_{ij} * P_i$ 이다.

그림 2은 〈알고리즘 2〉을 구현한 씨스톨릭 어레이다. 그리고 그림 3은 씨스톨릭 셀(cell)의 구조를 표시한다. 씨스톨릭 셀은 그림과 같이 2가지 형태가 있다. (a)는 3개의 입력 포트와 2개의 출력 포트가 있으며, 곱셈과 덧셈을 하는 기능이 있다. 그리고 (b)는 2개의 입력 포트와 1개의 출력 포트를 가지고 있으며, 나눗셈과 곱셈을하는 기능이 있다. 이들 씨스톨릭 셀들은 모두 로칼 메모리(local memory ; LM)를 내장하고 있다. 그림 2의 씨스톨릭 어레이에서 ARRAY 유니트의 윗쪽 선형 어레이는 상삼각 행렬의 원소와 벡타의 내적과 그의 합이 계산된다. 그리고 아랫쪽선형 어레이는 하삼각 행렬과 대각선 행렬의 원소와 벡타의 내적이 계산된다. 이들의 각 어레이에서 계산된 결과는 누적되면서 각 셀로 전파된 다음 그림에서 MD/A 유니트의 가산기 SA로 입력된다. 가산기 SA에서는 이들 상하 어레이의 출력이 더해져서 $W_k=A P_k$가 계산된다.

지금 스트라이프 메트릭스 A를 생각해보면 메트릭스 A는 앞에서 기술한 바와 같이 A_L+D+A_U로 분해할 수 있으며, 이들의 스트라이프 구조는 하삼각 행렬의 스트라이프 구조 $\sum_{AL}=\{S_1,...,S_{m-1}\}$, 대각선 행렬의 스트라이프 구조 $\sum_D=\{S_m\}$이다. 여기서 S_m은 모든 대각선 원소의 위치 (i,j), i=1,2,...,n을 포함한다. 그리고 상삼각행렬의 스트라이프의 구조 $\sum_{AU}=\{S_{m+1},....,S_{2m-1}\}$이다. 여기서 메트릭스 A가 대칭이므로

$\sum_{AL}=\sum_{AU}$이다. 따라서 그림 2와 같이 각 스트라이프 구조의 원소마다 하나의 셀을 할당하면 그림에서 ARRAY 유니트의 윗쪽 어레이에 m−1개의 셀이 필요하고, 아랫쪽 어레이에 m개의 셀이 필요하다. 결국은 메트릭스-벡타 곱셈을 하는 데 2m−1개의 셀로 구성된 어레이가 필요하게 된다.

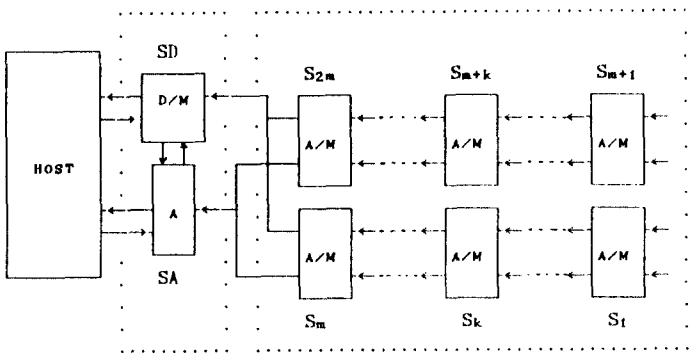

그림 2. 씨스톨릭 어레이
Fig. 2. systolic array

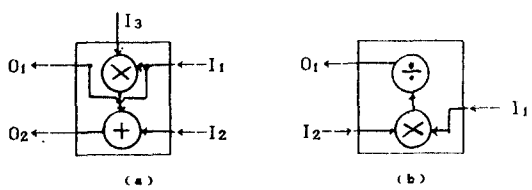

그림 3. 프로세서 셀
Fig. 1. processor cell

지금 알고리즘을 설명하기위해서 T를 전역 클럭(global clock)에 의해서 정의되는 시간이라고 하고, t=T-k+1를 k번째 셀에서 정의되는 로칼타임(local time)이라고 하자. 그리고 베타의 원소가 셀의 입력 포트에 입력되고나서 한 사이클이 경과한후 셀에서 처리된 결과가 셀의 출력 포트에 나타난다고 하자.

k번째 셀의 어떤 주어진 시간 t에서 ARRAY 유니트의 상하 어레이 입력 포트에 베타 P_t가 입력되면, k번째 로칼 메모리 LM_{LK}와 LM_{UK}에서 스트라이프 S_k와 S_{m+k}의 원소 $a_{t,\sigma k}(t)$와 $a_{\mu k}(t),t$가 각각 선택되어 $a_{t,\sigma k}(t)$* P_t와 $a_{\mu k}(t),t$ * P_t의 곱셈 연산이 동일 사이클에서 실행되고, 이들의 연산 결과는 각각의 로칼 메모리 $LM_{LK}(t)$와 $LM_{UK}(t)$에 저장된다. 여기서 만약에 t∈j_dom(S_k)이면 스트라이프의 정의에 의해서 $\bar{\mu}k(t)=\mu k(t)$이다. 그리고 $i=\mu k(t)$이므로 $a_{\mu k}(t),t$ * $P_t=a_{i,\sigma(i)}$ * $P\sigma(i)$이다. 그다음 사이클에서 k번째 셀의 입력포트에 W 데이타 스트림의 원소 W_i가 도착하면 $LM_{UK}(i)$ 및 $LM_{LK}(i)$에 저장된 대응하는 원소가 선택된다. 이때 스트라이프 S_k가 행 i에 대응하는 원소 위치를 포함하고 있으며 $LM_{LK}(i)=a_{i,\sigma(i)}$ * $P_{\sigma(i)}$, $LM_{UK}(i)=a_{\bar{\mu}(j),j}$ * P_j이고, 그렇지 않으면 $LM_{LK}(i)=0$, $LM_{UK}(i)=0$가 된다. W 데이타 스트림(stream)의 원소 $W_1, W_2, \ldots, W_n$은 모두 0으로 초기화 되고, P 데이타 스트림의 원소 $P_1, P_2, \ldots, P_n$과 같은 방향으로 전파(propagate)된다. 따라서 어떤 특정한 i에 대해서 W_i는 k에 대해서 (i,j)∈S_k이면 a_{ij} * P_j 항이 ARRAY 유니트의 상하 어레이에서 각각 누적되어 결과는 $W_k=AP_k$의 i번째 원소 $W_i=\sum_{j=1}^{n} a_{ij}$ * P_j가 가산기 SA 출력단자에서 얻어진다. 1≤k≤m에 대해서 k번째 셀의 동작을 기술하면 다음 알고리즘과 같다.

〈알고리즘 3〉

For t=1 to n do

윗쪽 어레이

(1)Read P_t from I_1 into α and $W_{t-\delta k}$ from I_2 into φ

(2)Read $a_{\bar{\mu}k(t),t}$ and $\bar{\mu}k(t)$ from $LM_{UK}(t)$ into β and λ, respectively

(3)$LM_{UK}(\lambda)\leftarrow\beta$ * α

(4)$\varphi\leftarrow\varphi+LM_{UK}(t-\delta k)$

(5)Write α and φ on O_1 and O_2, respectively

아래쪽 어레이

(1)Read P_t from I_1 into α' and $W_{t-\delta k}$ from I_2 into φ'

(2)Read $a_{t,\sigma k(t)}$ and $\sigma k(t)$ from $LM_{LK}(t)$ into β' and λ', respectively

(3)$LM_{LK}(\lambda')'\leftarrow\beta'$ * α'

(4)$\varphi'\leftarrow\varphi'+LM_{LK}(t-\delta k)$

(5)Write α' and φ' on O_1 and O_2, respectively

이 알고리즘에서 δk는 k번째 셀에서의 곱셈과 k+1번째 셀에서의 곱셈사이에 인터리브(interleave)되는 덧셈의 곱셈에 대한 지연시간이다. 다시 말하면 베타 P_k가 입력되고나서 W_k가 입력되는 시간을 결정해주는 것이다. 지금 P^*(곱셈하는 데 걸리는 시간)와 P^+(덧셈하는 데 걸리는 시간)가 2-레벨 파이프라인으로 실행된다고 가정하면 알고리즘에서 스탭 (3)의 결과들은 사이클 t+P^*에서 로칼메모리에 저장된다. 이렇게 되려면 (t+P^*)-$\delta k\leq\mu k(t)$의 조건이 만족되어야 하며, 이 조건은 j∈j dom(S_k)에 대해서

$$\delta k \geq j-\mu k(j)+P^* \tag{4-1}$$

이 만족되어야 한다. 이 식은 셀 k에서 요구되는 δk의 최소 값이다.

만약에 W_1과 W_m이 동일한 전역 사이클 T 동안 셀 k와 k+1에 각각 있다면

$$\delta k+1=\delta k-P^++1 \tag{4-2}$$

이고, $\delta 1=Ts-1$로 된다. 여기서 Ts는 W_k 스트림의 최소 시간으로

$$Ts=B1+P^*+1 \tag{4-3}$$

이다. 따라서 전체 ARRAY 유니트를 실행하는 시간은 $n+B1+m(P^++1)+P^*$로 된다. 여기서 B1은 메트릭스의 하측 대역폭이다.[1]

그림 2에서 ARRAY 유니트의 출력은 벡터 P_k와 (A_L+D) * P_k 및 A_U * P_k가 된다. 이들중 P_k는 그림 2에서 MD/A 유니트의 SD 셀로 입력되고, 나머지는 SA 가산기로 들어가 서로 더해진후 SD 셀에 입력된다. 이들 사이에 동작 제어는 HOST에 의해서 행해진다. 여기서 HOST는 씨스톨릭 셀을 초기화하며, 매 반복 스탭을 초기화하고 제어한다. 또 매 반복 때마다 MD/A에 P_i, r_i를 공급하고 MD/A로 부터 벡터 P_{i+1}, r_{i+1}과 스칼라 α를 받는다. 그리고 X_{i+1}를

계산하고, r_{i+1}를 체크하여 수렴여부를 확인한다. 시스템의 각 유니트에 할당된 여러가지 계산과 유니트 사이에 데이타 이동을 표 1에 요약해 놓았다.

표에서 각 열은 〈알고리즘 2〉의 각 스탭에 대응한다. 데이타의 이동은 처음 6개 행에 표시되고, 산술연산의 실행은 다음 3개의 행에 표시된다. 여기서 간단히 하기위해서 덧셈, 곱셈, 나눗셈을 모두 1개의 사이클로 연산이 되는 것으로 가정하면 GCM 반복 알고리즘은 대략 $n+B_l+2m+12$ 씨스톨릭 사이클마다 1회씩 반복 수행된다. 시스템에서 모든 씨스톨릭 셀이 거의 모든 사이클에서 이용이 되므로 어레이의 이용율이 거의 100%에 가깝다.

표 2와 표 3은 다음과 같은 〈예 1〉과 〈예 2〉에 대하여 수치 실험한 결과이다. 〈예 1〉은 sparsity가 적은 경우이고, 〈예 2〉는 sparsity가 큰 경우이다. 표에서 보는 바와 같이 〈예 2〉가 〈예 1〉 보다 빨리 수렴하며, 어느 경우나 기껏해야 N(방정식의 차수)스텝으로 수렴이 된다.

〈예 1〉

$$A=\begin{bmatrix}1 & 2 & -1 & 1\\ 2 & 5 & 0 & 2\\ -1 & 0 & 6 & 0\\ 1 & 2 & 0 & 3\end{bmatrix}$$

$$b=[0\ 2\ -1\ 1]^t$$

〈예 2〉

$$A=\begin{bmatrix}1 & 0 & 0 & 1\\ 0 & 1 & 0 & 1\\ 0 & 0 & 1 & 0\\ 1 & 0 & 0 & 1\end{bmatrix}$$

$$b=[0\ 2\ -1\ 1]^t$$

표 1. 〈알고리즘 2〉의 실행과정

Table 1. Execution processes of 〈algorith 2〉

		step 1	step 2	step 3	step 4	step 5
데이타이동	MD/A에 LM의 초기 내용	W_{k-1}	W_k	W_k	W_k	W_k
	HOST→MD/A	P_k		r_k		
	MD/A→ARRAY	P_k				
	ARRAY→MD/A	W_k				
	MD/A→HOST		α_k	r_{k+1}		P_{k+1}
	MD/A에 LM의 최후 내용	W_k	W_k	W_k	W_k	W_k
실행	ARRAY에서 실행	1.1 1.2				
	MD/A에서 실행	1.3 1.4 1.5	2.1	3.2	4.1	5.1
	HOST에서 실행			3.1		

표 2. 〈예 1〉에 대한 실행 결과

		components of the vector				
step	vector	1	2	3	4	α_i
0	x_0	1	0	0	0	
	r_0	−1	0	0	0	
	p_0	−1	0	0	0	
	w_0	−1	−2	1	−1	1
1	x_1	0	0	0	0	
	r_1	0	2	−1	1	

	p_1	−6	2	−1	1	
	w_1	0	0	0	1	6
2	x_2	−36	12	−6	6	
	r_2	0	2	−1	−5	
	p_2	−30	12	−6	0	
	w_2	0	0	−6	−6	.833
3	x_3	−61	22	−11	6	
	r_3	0	2	4	0	
p_3	−20	10	0	0		
	w_3	0	10	20	0	.2
4	x_4	−65	24	11	6	
	r_4	0	0	0	0	
	p_4	0	0	0	0	
	w_4	0	0	0	0	

표 2. 〈예 2〉에 대한 실행 결과

		components of the vector				
step	vector	1	2	3	4	α_i
0	x_0	0	0	0	0	
	r_0	0	2	−1	1	
	p_0	0	2	−1	1	
	w_0	1	2	−1	1	1
1	x_1	0	2	−1	1	
	r_1	−1	0	0	0	
	p_1	−1	.3333334	−.1666667	.1666667	
	w_1	−.83333333	.3333334	−.1666667	−.833333	1.2
2	x_2	−1.2	2.4	−1.2	1.2	
	r_2	0	−.4	.2	1	
	p_2	−1.2	0	0	1.2	
	w_2	0	0	0	0	

V. 결 론

메트릭스 A를 $A=(A_L+D)+A_U$로 분해하여 대형 스파스 메트릭스 선형 방정식 $Ax=b$를 해석하는 CGM 시스톨릭 알고리즘을 제시하였다. CGM 반복 알고리즘은 대략 $n+B_1+2m+12$ 씨스톨릭 사이클마다 1회씩 반복 수행된다. 이를 구현하기위해서 메트릭스를 스트라이프 구조로 변환하고, 스트라이프 구조의 메트릭스를 실행하는 씨스톨릭 어레이를 제안했다. 따라서 제안된 씨스톨릭 어레이는 CGM 알고리즘의 주 프로시저인 $W_k = A P_k$의 메트릭스-베타 곱셈을 2개의 선형 어레이가 병행적으로 실행하므로서 실행시간이 1개의 선형 어레이를 사용한 것에 비해 절반에 가깝게 단축한다. 그리고 스트라이프 구조를 이용하므로서 불규칙하게 분포된 스파스 메트릭스를 효율적으로 처리할 수 있다.

앞으로 본 연구 과제는 스파스 메트릭스 A가 비대칭인 경우 CGM 씨스톨릭 알고리즘의 작성과 씨스톨릭 어레이를 구성는 것으로 확장될 것이다. 또한 스트라이프 S의 행과 열의 함수 관계로 부터 스파스 메스릭스 분포를 정렬하는 알고리즘의 연구도 고려될 것이다.

참 고 문 헌

1. J.Götze and U.Schwiegelshohn, "Sparse matrix-vector multiplication on a systolic array," IEEE 1988 Int. Conf. on Acoustic, Speech and signal Processing, New York, Vol. 9, pp.2061-2064, 1988.
2. M.R. Hestenes and E. Stiefel, "Method of Conjugate Gradients for Solving Linear Systems," J. of Research of the National Bureau of Standards, Vol. 49, No.6, Dec. 1952.
3. M.T.O' Keefe and J.A.Forts, "A comparative study of two systematic design methodologies for systolic arrays," Proc. of Int. Conf. Parallel Processing, pp.672-675, Aug. 1986.
4. Y.T.Hwang and Y.H.Hu, "Parameterized dependence graph and its applications to multi-stage systolic mapping procedure," IEEE 1988 Int. Cof. on Acoustics, Speech and Signal Processing, New York, Vol.2, pp.1029-1032, 1990.
5. J.J.Navarro, J.M.LLaberia, and Mateo Valereo, "Solving matrix problems with no size restriction on a systolic array processor," Proc. of Int. Conf. Parallel Processing, pp. 676-683, Aug. 1986.
6. W.Luk, "Systolic band-matrix multipliers," Electronics letters, Vol.26, No.6, pp.403-405, March 1990.
7. M.H.LEE and Y.YASUDA, "New 2D systolic array algorithm for DCT/DST," Electronics letters, Vol.25, No.25, pp.1702-1704, Dec. 1989.
8. S.PS.Ram, "Matrix and vector multiplications using a $2_{1/2}$ dimensional systolic array," Electronics letters, Vol.26, No. 18, pp. 1453-1454, Aug. 1990.
9. R.W.Stewart, "Mapping signal processing algorithms to fixed architectures," IEEE 1988 Int. Conf. on Acoustics, Speech and Signal Procssing, New York, Vol.9, pp.2037-2040, 1988.
10. RmMelhem, "A systolic accelerator for the iterative solution of sparse linear systems," IEEE Tras. on computers, Vol.38, No.11, pp. 1591-1595, 1980.
11. R.Melhem, "Determination of stripe structures for finte element matrices," SIAM J. NUMER.ANAL., Vol.24, No6, pp.1433, 1987.
12. S.Pawagi, P.S.Gopalarkrishnan, and I.V. Ramakrishnan, "A parallel algorithm for dominators," Proc. of Int. Conf. Parallel Processing, pp.877-879, Aug. 1986.
13. D.Helmbold and E.Mayr, "Perfect graphs and parallel algorithms," Proc. of Int. Conf. Parallel Processing, pp.853-860, Aug. 1986.
14. Po RONG CHANG and C.S.George LEE, "A decomposition approach for balacing large-scale acyclic data flow graphs," IEEE Trans. on computers, Vol.29, No.1, Jan. 1990.
15. G.H.Golb & C.F.Van Lona, Matrix computations, The Johns Hopkins University press, pp. 372-377, 1987.
16. D.A.Arpin and Yongmin Kim, "PARSOR:A parallel processor for sparse matrix solution by SOR iterration," Department of electronical engineering univers-ity of Washington, Seattle, Washington 98195.
17. A.Jennings, Matrix computation for engineers ad scientists, John Wiley & Sons, pp.212-221, 1977.

李 秉 洪(Byeong Hong Lee) 正會員
1944年 12月 6日生
1970年 2月：韓國航空大學通信工學科 卒業(學士)
1982年 2月：東國大學校 大學院 電子工學科卒業(碩士)
1988年 9月～現在：韓國航空大學 大學院 電子工學科 博士課程
1974年 11月～1979年 2月：麗水水産專門大學 專任講師
1979年 3月～現在：烏山專門大學 教授

蔡 洙 煥(Soo Hoan Chae) 正會員
1950年 10月 28日生
1973年 2月：韓國航空大學 電子工學科 卒業(工學士)
1973年 7月～1977年 8月：空軍技術將校
1977年 8月～1983年 7月：金星通信研究所
1983年 8月～1985年 5月：美國알라바마大學 電算學科(理學碩士)
1985年 5月～1988年 12月：美國알라바마大學 電氣工學科(工學博士)(電子計算機構造專攻)
1989年 3月～現在：韓國航空大學 電子計算學科 助教授

金 正 善(Jeong Sun Kim) 正會員
1941年 5月 5日生
1965年～1990年 現在：航空大學 教授
1988年～1990年：電子工學科 學科長
1990年 2月：電子計算所長
1990年 1月～：韓國通信學會 理事