Journal of The Korea Society of Computer and Information
Vol. 19, No. 9, September 2014
www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2014.19.9.021

특이치 분해를 위한 최적의 2차원 멀티코어 시스템 탐색

박 용 훈*, 김 철 홍**, 김 종 면*

Exploration of an Optimal Two-Dimensional Multi-Core System for Singular Value Decomposition

Yong-Hun Park*, Cheol-Hong Kim**, Jong-Myon Kim*

요 약

특이치 분해는 다양한 분야의 데이터 집단에서 고유한 특성을 찾는 특징 추출 분야에 많이 활용되고 있다. 하지만 특이치 분해의 복잡 행렬 연산은 많은 연산 시간을 요구한다. 본 논문에서는 특이치 분해의 대표적인 알고리즘인 one-sided block Jacobi를 고속 처리하기 위해 2차원 멀티코어 시스템을 이용하여 효율적으로 병렬 구현하고 성능을 향상시킨다. 또한, one-sided block Jacobi 알고리즘의 다양한 행렬 (128x128, 64x64, 32x32, 16x16)을 서로 다른 2차원 PE 구조에 구현하고 성능 및 에너지를 분석함으로써 각 행렬에 대한 최적의 멀티코어 구조를 탐색한다. 더불어 동일한 행렬의 one-sided block Jacobi 알고리즘에 대해 선택된 멀티코어 구조와 상용 고성능 그래픽스 프로세싱 유닛 (GPU)과의 성능 비교를 통해 제안한 2차원 멀티코어 방법의 잠재 가능성을 확인한다.

▸ Keywords : 디자인 공간 탐색, 특이치 분해, 단면 블록 자코비, 2차원 멀티코어 시스템

Abstract

Singular value decomposition (SVD) has been widely used to identify unique features from a data set in various fields. However, a complex matrix calculation of SVD requires tremendous computation time. This paper improves the performance of a representative one-sided block Jacoby algorithm using a two-dimensional (2D) multi-core system. In addition, this paper explores an optimal multi-core system by varying the number of processing elements in the 2D multi-core system with the same 400MHz clock frequency and TSMC 28nm technology for each matrix-based one-sided block Jacoby algorithm (128x128, 64x64, 32x32, 16x16). Moreover, this paper

•제1저자 : 박용훈 •교신저자 : 김종면
•투고일 : 2014. 6. 18, 심사일 : 2014. 7. 15, 게재확정일 : 2014. 8. 28.
* 울산대학교 전기전자컴퓨터공학과(School of Electrical, Electronics, and Computer Engineering, University of Ulsan)
** 전남대학교 전자컴퓨터공학부(School of Electronics and Computer Engineering, Chonnam National University)
※ 이 논문은 2013년도 정부(미래창조과학부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. NRF-2013R1A2A2A05004566).

demonstrates the potential of the 2D multi-core system for the one-sided block Jacoby algorithm by comparing the performance of the multi-core system with a commercial high-performance graphics processing unit (GPU).

▸ Keywords : Design space exploration, singular value decomposition, one-sided block Jacobi, two-dimensional multi-core system.

I. 서 론

특이치 분해는 특이치 값의 특징인 유일성, 행렬의 일부분이 변해도 특이치는 크게 변하지 않는 시스템의 안정성, 그리고 행렬의 계수가 작은 근사 행렬로 쉽게 변환 할 수 있는 근사화의 특징을 가지고 있어[1-9] 다양한 분야의 특징 추출 연구에 많이 활용 되고 있다. 특이치 분해가 많이 활용되고 있는 분야로는 전력 전달, 영상 처리, 오디오 및 음성 처리, 패턴 인식, 머신 지능, 의학 영상 등이 있다[10]-[20]. 하지만 특이치 분해는 행렬식이나 고차 방정식 해법과 같은 복잡 행렬 연산을 요구하기 때문에 상당한 처리 시간이 요구된다.

이를 해결하기 위해 선형 프로세싱 엘리먼트 (processing element, PE) 구조를 이용하여 one-sided Jacobi 알고리즘의 2개 열 단위를 각각의 PE에 맵핑해 병렬 처리가 가능하게 함으로써 성능을 향상시켰다[21]. 하지만 one-sided block Jacobi 알고리즘의 행렬 크기에 비해 사용가능한 PE 개수가 제한적이라는 단점이 있다. 1990년도 후반 이후에는 슈퍼컴퓨터를 이용해 one-sided block Jacobi 알고리즘의 성능을 개선하려는 시도가 있었다[22-26]. 슈퍼컴퓨터를 이용한 특이치 분해 가속화 연구는 특이치 분해 알고리즘의 성능은 개선하지만 시스템 면적, 에너지 소비 및 가격 측면을 고려할 때 응용분야에서의 사용은 매우 제한적이다.

이를 대체하기 위한 시스템으로 단일 명령어, 다중 데이터 (SIMD) 기반 멀티코어 프로세서가 유망하다[27-30]. SIMD 기반 멀티코어 프로세서는 많은 수의 저비용 프로세싱 엘리먼트(processing element, PE)를 구성함으로써 고성능을 추구하는 동시에 이웃하는 PE들 사이에 짧은 와이어를 이용하여 데이터 통신을 수행함으로써 저전력을 만족시킨다. 특히 지역성과 규칙성이 있는 행렬 연산은 SIMD 기반 멀티코어 프로세서에 적합하다.

본 논문에서는 SIMD 기반 멀티코어 프로세서를 이용하여 one-sided block Jacobi 알고리즘의 성능을 가속화한다. 또한 one-sided block Jacobi 알고리즘의 다양한 행렬 구조 (128x128, 64x64, 32x32, 16x16)를 서로 다른 형태의 2차원 PE 구조에 구현하고 성능 분석을 통해 최적의 PE구조를 선택한다. 또한 동일한 one-sided block Jacobi 알고리즘에 대해 선택한 멀티코어의 PE구조와 상용 고성능 그래픽스 프로세싱 유닛 (GPU)와 성능 비교를 통해 제안하는 멀티코어 프로세서에 대한 잠재가능성을 입증한다.

본 논문의 구성은 다음과 같다. 2장에서는 특이치 분해에 대한 배경 지식과 본 논문의 관련 연구에 대해 소개하고, 3장에서는 SIMD 기반 멀티코어 시스템을 이용한 one-sided block Jacobi 알고리즘의 병렬 구현에 대해 설명한다. 4장에서는 SIMD 기반 멀티코어 시스템에 대한 성능 분석을, 5장에서는 본 논문을 결론을 맺는다.

II. 배경 지식 및 관련 연구

1. 특이치 분해

특이치 분해 (singular value decomposition, SVD)는 크기가 크고 다루기 힘든 행렬을 보다 작은 가역 정사각 행렬로 근사하게 분해하는데 유용하게 사용되는 행렬 분해 방법이다. $n \times n$ 크기 행렬의 계수가 r인 행렬 A의 특이치 분해 결과는 식(1)과 같다. 직교 행렬(orthogonal matrix) U는 AA^T의 고유벡터(eigenvector)들의 집합이고, 직교 행렬 V^T는 A^TA의 고유벡터들의 집합이며 u_k와 v_k는 벡터이다. 대각 행렬(diagonal matrix) S는 특이치 행렬을 의미하며, A^TA의 결과로부터 행렬식 연산을 통해 특성방정식을 만들어 내고 특성 방정식의 해들에 루트(square root)를 취하고 난 후 크기가 큰 순서대로 나열한다.

$$A = USV^T = [u_1, u_2, \ldots u_N] \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_N \end{bmatrix} [v_1, v_2, \ldots v_N]^T \quad (1)$$

2. One-sided Jacobi 알고리즘

특이치 분해는 행렬식을 연산을 통해 특성 방정식을 만들어내는 과정과 특성방정식의 해를 구하는 복잡하고 높은 연산량을 요구한다. One-sided Jacobi는 2개의 열에 대한 연산을 반복적으로 수행하는 알고리즘으로 특이치 분해를 비교적 규칙적이고 간단하게 만들어 준다.

알고리즘 1은 one-sided Jacobi의 슈도코드를 보여주며, 행렬 A는 입력행렬, 행렬 V는 단위행렬이다. p와 q는 중복되지 않는 서로 다른 수로 열의 선택에 사용된다. 또한 입력 행렬은 a, b, c를 만드는 과정에서 부분 대칭화를 수행하기 때문에 입력 행렬로 A^TA나 AA^T와 같은 대칭 행렬을 만들지 않는다. 또한 tolerance를 지정해 역치값과 근사한 값이 되면 더 이상 반복하지 않도록 sweep의 횟수를 제한한다. 완성된 행렬 A의 각 열 성분을 정규화 함으로써 좌측 특이 벡터 U를 만들어 내고, 완성된 행렬 A의 각 열에 대해 놈(norm)을 구함으로써 특이치 행렬 S를 만든다. 마지막으로 행렬 V는 우측 특이 벡터가 된다. 그림 1(a)는 하나의 PE에서 one-sided Jacobi를 처리하는 과정에서 2개의 열이 선택되는 순서를 보여준다. 열을 선택하는 과정 하나하나가 1 step이며, 모든 열에 대해 연산이 전부 처리된 상태가 1 sweep이다. One-sided Jacobi에서 2개의 열을 선택하여 처리해야 할 step의 수는 행렬의 크기가 n x n일 경우 1에서 n-1까지의 합이다.

알고리즘1. One-sided Jacobi

repeat
for all pairs p 〉 q

$$a = \sum_{k=1}^{n} A_{kp}^2$$

$$b = \sum_{k=1}^{n} A_{kq}^2$$

$$c = \sum_{k=1}^{n} A_{kp} * A_{kq}$$

$$\theta = (b-a)/(2*c)$$
$$t = sign(\theta)/(|\theta| + \sqrt{1+\theta^2})$$
$$cs = 1/\sqrt{1+t^2}$$
$$sn = cs*t$$

for k = 1 to n
$$tmp = A_{kp}$$
$$A_{kp} = cs*tmp - sn*A_{kq}$$
$$A_{kq} = sn*tmp + cs*A_{kq}$$
endfor

for k = 1 to n
$$tmp = V_{kp}$$
$$V_{kp} = cs*tmp - sn*V_{kq}$$
$$V_{kq} = sn*tmp + cs*V_{kq}$$
endfor
endfor

until convergence(all $|c|/\sqrt{ab} \leq$ tolerance)

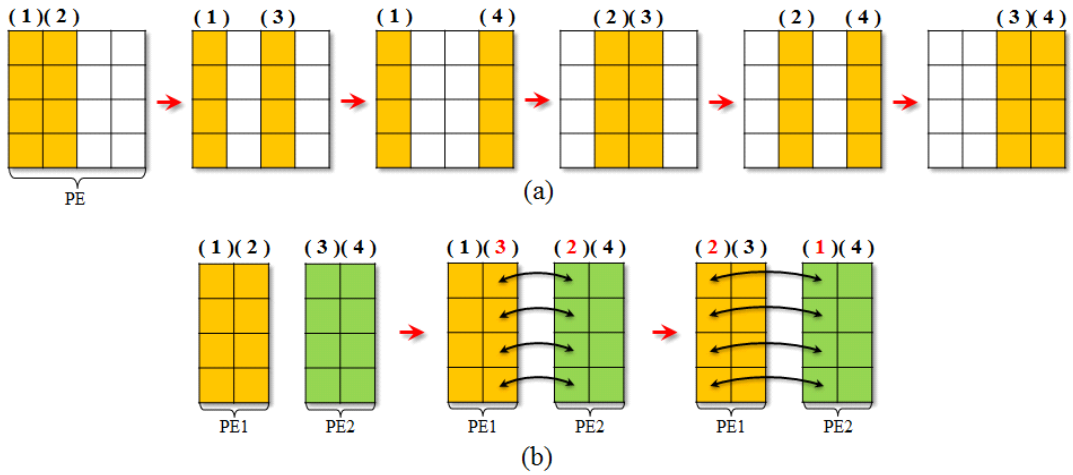

그림. 1. (a) One-sided Jacobi, (b) one-sided block Jacobi의 실행 과정
Fig. 1. Execution processes of (a)one-sided Jacobi and (b)one-sided block Jacobi

3. 병렬 구현 관련 연구

One-sided block Jacobi는 병렬 프로세서 시스템에서 각 PE 당 2개의 열을 담당하여 동시에 처리함으로써 one-sided Jacobi알고리즘을 병렬 처리하는 방법이다[21]. 그림 1(b)는 one-sided block Jacobi의 열 선택 순서를 보여준다. One-sided block Jacobi 처리를 위해서는 PE 간의 열 교환이 요구된다. 또한 PE 간의 열 교환은 전송 목적지가 다양할수록 순차적인 흐름이 많이 생겨나 성능 저하의 요인이 된다. 따라서 전송 목적지를 최소화하고, PE 간의 데이터 전송을 최소화하는 것이 중요하다. Round robin ordering은 이웃한 PE 간의 데이터를 전송하는 방법으로 데이터 전송의 일정한 흐름을 통해 전송 목적지를 최소화하고, 양 끝 PE 간의 통신이 필요 없어 메쉬 (mesh) 형태로 연결된 PE 구조에 적합하다.

그림 2는 8x8 행렬의 round robin ordering의 연산 순서를 보여준다. 각 PE에 맵핑된 숫자는 열의 번호를 의미한다. 행렬의 크기가 n x n 일 때 1부터 n-1까지 합의 처리량을 가지는 one-sided Jacobi와는 달리 처리해야 할 step의 수는 n-1번이 되기 때문에 빠른 처리가 가능하다.

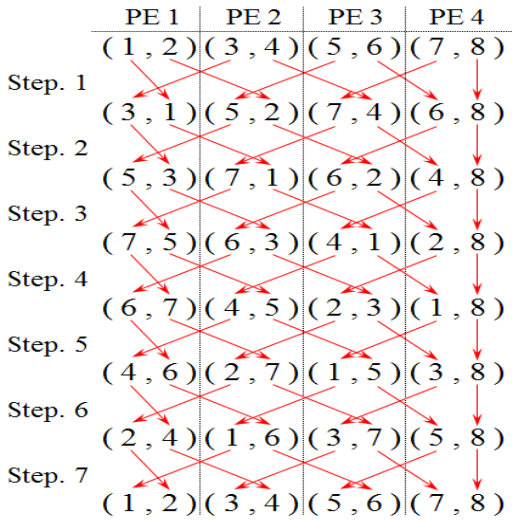

그림. 2. 라운드 로빈 순서
Fig. 2. Round robin ordering

위 방법 이외에도 특이치 분해를 가속화시키기 위해서 선형의 PE구조를 이용한 연구들이 소개되었다[22-30]. 하지만 one-sided block Jacobi의 경우 선형의 PE 구조뿐만 아니라 직사각형 혹은 정사각형 형태의 PE 구조를 통해서 더욱 더 성능을 향상시킬 수 있다. 따라서 본 논문에서는 one-sided block Jacobi 알고리즘의 다양한 행렬 (128x128, 64x64, 32x32, 16x16)을 서로 다른 2차원 PE 아키텍처의 멀티코어 시스템에 병렬구현하고 그에 대한 성능 및 에너지 효율을 분석함으로써 각 행렬의 특이치 분해 알고리즘에 대한 최적의 멀티코어 구조를 탐색하고자 한다.

III. 2차원 멀티코어 시스템을 이용한 one-sided block Jacobi 알고리즘의 병렬 구현

1. 2차원 멀티코어 시스템 아키텍처

그림 3은 본 논문에서 사용한 2차원 멀티코어 시스템의 아키텍처를 보여준다. 다수의 PE들과 이를 제어하는 어레이 제어 유닛(Array Control Unit, ACU)으로 구성된다. 또한 각 PE의 로컬 메모리에 데이터가 균등하게 분배되고, 격자 (mesh) 구조를 통해 인접한 PE 간의 데이터 전송이 수행되며, 동일한 명령어가 각 PE의 데이터를 동시에 처리한다. 각 PE는 다음과 같은 특징을 갖는다.

- 32비트 폭으로 구성된 로컬메모리
- 32비트 폭의 16개 3포트 범용 레지스터
- 기본적인 산술/논리 연산을 수행하는 ALU(arithmetic logic unit)
- 64비트 곱셈 및 누산기(multiplier and accumulator)
- 멀티 비트 산술/논리 시프트 연산을 수행하는 배럴 시프터
- 지역 정보를 이용해 각 PE들을 활성 및 비활성 시키는

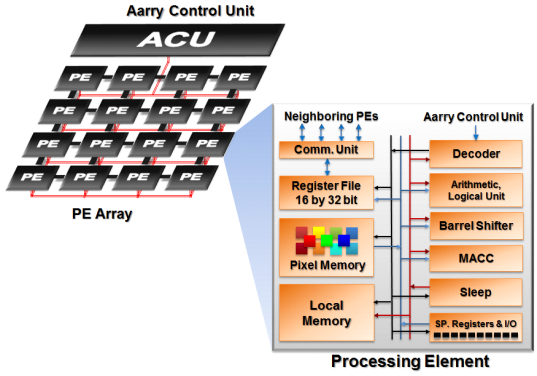

그림. 3. 2차원 멀티코어 시스템 구조
Fig. 3. 2-dimensional multi-core system architecture

Sleep 유닛

■ 이웃하는 PE들과 데이터 통신을 위한 NEWS (north-east-west-south) 네트워크 및 직렬 입·출력

2. One-sided block Jacobi 알고리즘의 병렬 구현

8 x 8 행렬의 경우, 그림 4(a)는 선형 PE 구조에 2개의 열씩 각 PE에 저장하는 것을 보여준다. 그림 4(b)는 직사각형의 PE 구조로 행에 대하여 한 행의 PE를 추가하였을 때 열의 데이터를 반으로 분할하여 저장하는 것을 보여준다. 유사한 방법으로 그림 4(c)는 정사각 형태의 PE 구조를 이용하여 행렬 데이터를 저장할 때 행의 수만큼 열을 나누어 저장하는 것을 보여준다.

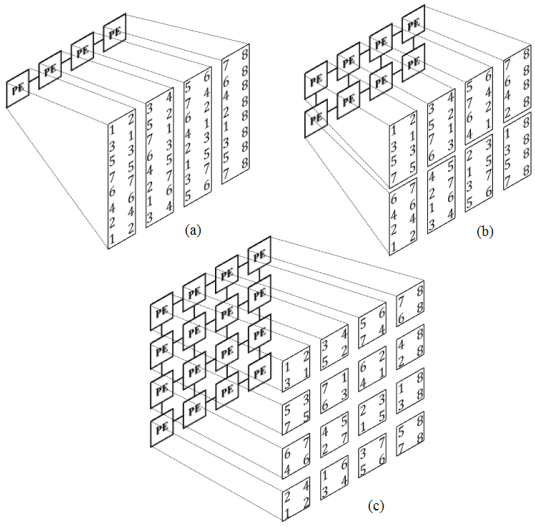

그림. 4. (a) 선형 PE 구조, (b) 직사각형 PE 구조, (c) 정사각형 PE 구조를 이용한 데이터 맵핑
Fig. 4. Data mapping using (a)a linear PE array, (b)a rectangular PE array, and (c)a square PE array

알고리즘 2는 one-sided block Jacobi의 각 연산 부분을 함수의 형태로 보여준다. n은 행렬의 열의 크기를 나타내며, n-1은 round robin ordering의 횟수를 의미한다. One-sided block Jacobi 알고리즘에서는 PE 간의 데이터 전송을 포함해 7개의 함수로 구성된다. 〔make a, b, c〕와 〔transfer a, b, c〕는 one-sided block Jacobi 과정 중에서 a, b, c를 연산하는 과정을 나타낸다.

알고리즘2. One-sided block Jacobi
repeat
for 1 to n-1
〔make a, b, c〕
〔transfer a, b, c〕
〔make cs, sn〕
〔make min〕
〔transfer cs, sn〕
〔col-update〕
〔col exchange〕
endfor
until (Min ≤ tolerance)

〔make a, b, c〕는 각 PE에 분산된 데이터를 이용해서 각각의 PE가 저장하고 있는 행렬의 데이터들에 대해서 a, b, c를 연산하는 과정을 동시에 수행하여 각 PE가 가지고 있는 열 데이터의 부분별 a, b, c 결과들을 만들어 낸다. 그리고

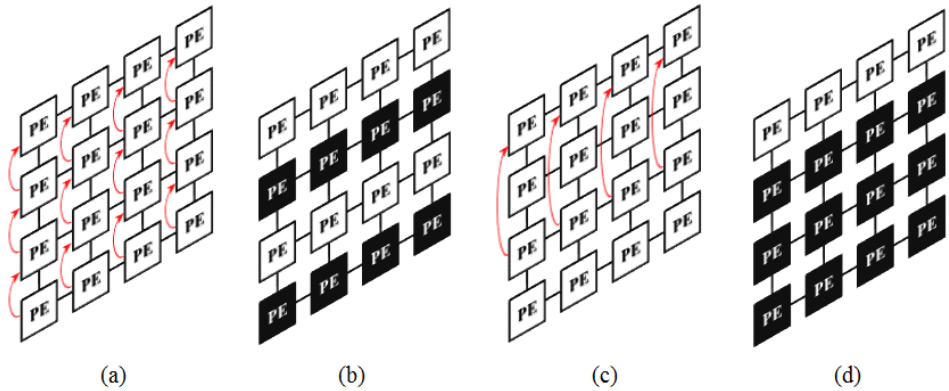

그림. 5. 〔transfer a, b, c〕 에서의 데이터 전송 과정
Fig. 5. Data transfer process in 〔transfer a, b, c〕

〔transfer a, b, c〕를 통해서 분산된 결과를 하나로 합함으로써 최종 a, b, c 결과를 만들어 낸다. 그림 5는 각각의 PE에 분산된 연산 결과를 첫 번째 행의 PE들에게 전달하는 과정을 보여준다. 본 논문에서 사용한 2차원 멀티코어 시스템에서는 모든 명령어가 PE에 동시에 수행되며, 그림 5(a)에서는 이웃하는 PE에 동시에 연산 결과를 전송한다. 그림 5(b)에서는 홀수 행의 PE는 짝수 행의 PE들로부터 받은 데이터를 이용하여 더하기 연산을 수행한다. 짝수 행의 PE는 이때 비활성화 상태가 되어 연산을 수행하지 않는다. 그림 5(c)는 첫 번째 행의 PE 데이터와 세 번째 행의 PE 데이터를 더하기 위해서 연속해서 2번의 데이터를 전송하여 첫 번째 PE로 보낸다. 마지막으로 그림 5(d)에서는 전송된 데이터에 대해 첫 번째 행의 PE들이 더하기 연산을 수행하여 a, b, c 결과를 만든다. 이때 나머지 PE들은 비활성화 상태가 된다.

〔make cs, sn〕단계는 첫 번째 행의 PE들이 a, b, c를 이용해서 cs와 sn을 만드는 과정이다. 〔make min〕단계에서는 a, b, c를 이용하여 각 PE가 min 값을 만들고, 이 값들 중에서 가장 작은 값을 min 값으로 선택한다. 또한 tolerance 값보다 작을 경우, 알고리즘은 종료된다. 병렬 구조에서 가장 작은 값을 찾기 위해서는 PE 간의 데이터 전송을 이용해서 각 PE에 분산된 min 값을 모아야 한다. 본 논문에서 사용한 2차원 멀티코어 시스템에서는 비활성화 PE가 있는지를 확인하여 각 PE가 가지고 있는 min 값을 tolerance 값과 비교하여 더 작은 값이 있으면 알고리즘이 종료되도록 구현하였다.

〔transfer cs, sn〕은 cs와 sn 값을 분산된 열의 데이터를 가지고 있는 나머지 PE들에게 전달하는 기능을 한다. 그리고 〔col-update〕를 통해서 전달된 cs와 sn 값을 이용하여 각 PE에서는 자신이 가지고 있는 열의 데이터에 연산을 수행한다. 마지막으로 〔col-exchange〕를 통하여 PE 간의 round robin ordering을 수행하여 데이터를 교환한다.

IV. 실험 환경 및 성능 분석

1. 실험 환경 및 성능 평가 지표

표 1은 다양한 2차원 멀티코어 모델들의 성능을 평가하기 위한 세 가지 지표를 보여준다. 실행시간 C는 알고리즘이 수행되는데 필요한 총 cycle 수를 의미하며, f_{clk}는 시스템의 클럭 주파수를 의미한다. 에너지는 알고리즘이 수행되는데 필요한 총 에너지를 나타내며, 면적은 시스템의 단위 면적을 나타낸다.

표 1. 성능 평가 지표
Table 1. Performance evaluation metrics

실행 시간	$t_s = \frac{C}{f_{dk}}$
에너지 효율	$\eta_E = \frac{1}{t_s \times Energy_{(Joule)}}$
면적 효율	$\eta_A = \frac{1}{t_s \times Area_{(mm^2)}}$

표 2. 본 논문에서 사용한 2차원 멀티코어 모델의 파라미터
Table 2. Parameters of 2D multi-core models used in this paper

Parameter	Value						
Matrix Size	128 x 128						
number of PEs	64 x 1	64 x 2	64 x 4	64 x 8	64 x 16	64 x 32	64 x 64
Data per PE (DPPE)	256	128	64	32	16	8	4
Local Memory per PE (word)	1024	512	256	128	64	32	16
Matrix Size	64 x 64						
number of PEs	32 x 1	32 x 2	32 x 4	32 x 8	32 x 16	32 x 32	
Data per PE (DPPE)	128	64	32	16	8	4	
Local Memory per PE (word)	512	256	128	64	32	16	
Matrix Size	32 x 32						
number of PEs	16 x 1	16 x 2	16 x 4	16 x 8	16 x 16		
Data per PE (DPPE)	64	32	16	8	4		
Local Memory per PE (word)	256	128	64	32	16		
Matrix Size	16 x 16						
number of PEs	8 x 1	8 x 2	8 x 4	8 x 8			
Data per PE (DPPE)	32	16	8	4			
Local Memory per PE (word)	128	64	32	16			
Clork Frequency	400MHz						
Interconnet Network	Mesh						
Technology	TSMC 28nm						

표 3. 각 PE 구조 별 실행시간, 시스템 면적, 에너지 소비
Table 3. Execution time, system area and energy consumption for each PE architecture

Parameter	Value						
Matrix Size	128 x 128						
number of PEs	64 x 1	64 x 2	64 x 4	64 x 8	64 x 16	64 x 32	64 x 64
Execution Time (ms)	520.95	270.17	145.57	84.68	55.31	42.07	37.36
System Area (mm^2)	26.77	34.29	49.63	80.58	142.80	267.60	517.68
Energy ($Joule$)	0.0572	0.0538	0.0551	0.0617	0.0776	0.1129	0.1688
Matrix Size	64 x 64						
number of PEs	32 x 1	32 x 2	32 x 4	32 x 8	32 x 16	32 x 32	
Execution Time (ms)	118.26	63.39	36.40	23.37	17.39	15.03	
System Area (mm^2)	8.56	12.39	20.11	35.64	66.79	129.21	
Energy ($Joule$)	0.0059	0.0060	0.0067	0.0084	0.0120	0.0198	
Matrix Size	32 x 32						
number of PEs	16 x 1	16 x 2	16 x 4	16 x 8	16 x 16		
Execution Time (ms)	26.93	15.30	9.67	7.06	5.98		
System Area (mm^2)	3.09	5.02	8.89	16.67	32.24		
Energy ($Joule$)	0.00065	0.00072	0.00089	0.00126	0.00204		
Matrix Size	16 x 16						
number of PEs	8 x 1	8 x 2	8 x 4	8 x 8			
Execution Time (ms)	6.20	3.86	2.77	2.31			
System Area (mm^2)	1.25	2.22	4.16	8.04			
Energy ($Joule$)	0.00007	0.00009	0.00012	0.00020			

표 2는 본 논문에서 사용한 2차원 멀티코어 모델의 파라미터를 보여준다. 특징 추출 분야에서 많이 사용되는 특이치 분해의 4가지 행렬(128x128, 64x64, 32x32 16x16)을 여러 가지 형태의 2차원 멀티코어 구조에 병렬 구현하여 성능 및 효율을 분석한다.

2. 실행 시간

표 3은 one-sided block Jacobi 알고리즘의 4가지 행렬(128x128, 64x64, 32x32 16x16)에 대한 PE 구조별 실행시간을 보여준다. PE 수가 2배씩 증가할수록 실행시간은 감소하지만, 실행시간의 감소폭은 현저히 줄어드는 것을 알 수 있다. 이는 암달의 법칙 (Amdhal's Law)에서와 같이 PE의 증가 수에 따라 성능이 비례적으로 증가하지 않기 때문이다. 그림 6은 128x128 행렬에 대해 하나의 PE에서 각 함수를 처리하는데 요구되는 총 사이클 수의 예를 보여준다. 〔make-min〕과 〔make cs, sn〕 함수는 PE 구조가 변하더라도 요구되는 연산량은 동일하기 때문에 소요되는 사이클 수는 변하지 않는다. 반면 〔make a, b, c〕, 〔col-update〕 및 〔col-exchange〕함수는 PE 수가 2배씩 증가함에 따라 2배씩 감소한다. 〔transfer a, b, c〕 함수의 경우는 PE 수가 2배씩 증가 할 때 일정한 수준으로 증가한다. 마지막으로 〔transfer cs, sn〕 함수는 cs와 sn 값을 분산된 열의 데이터를 가지고 있는 나머지 PE들에게 전달하는 기능으로써 PE 수가 2배씩 증가하면 사이클 수도 2배씩 증가하지만, 소요되는 연산량이 다른 함수보다 상대적으로 작기 때문에 전체 사이클 수에는 거의 영향을 미치지 않는다. 함수 중에서 〔make a, b, c〕와 〔col-update〕 함수가 가장 큰 사이클 수를 요구한다. PEs=64x1에서 PEs=64x16까지는 〔make a, b, c〕와 〔col-update〕 두 개의 함수가 필요로 하는 사이클 수가 다른 함수에 비해 크기 때문에 PE 수가 2배 증가하면 2배에 가까운 사이클 수가 감소함을 알 수 있다. 하지만 PEs=64x16 이후에서는 나머지 함수들과의 사이클 수 차이가 점점 비슷해지면서 사이클 수의 감소 비율은 계속 낮아진다.

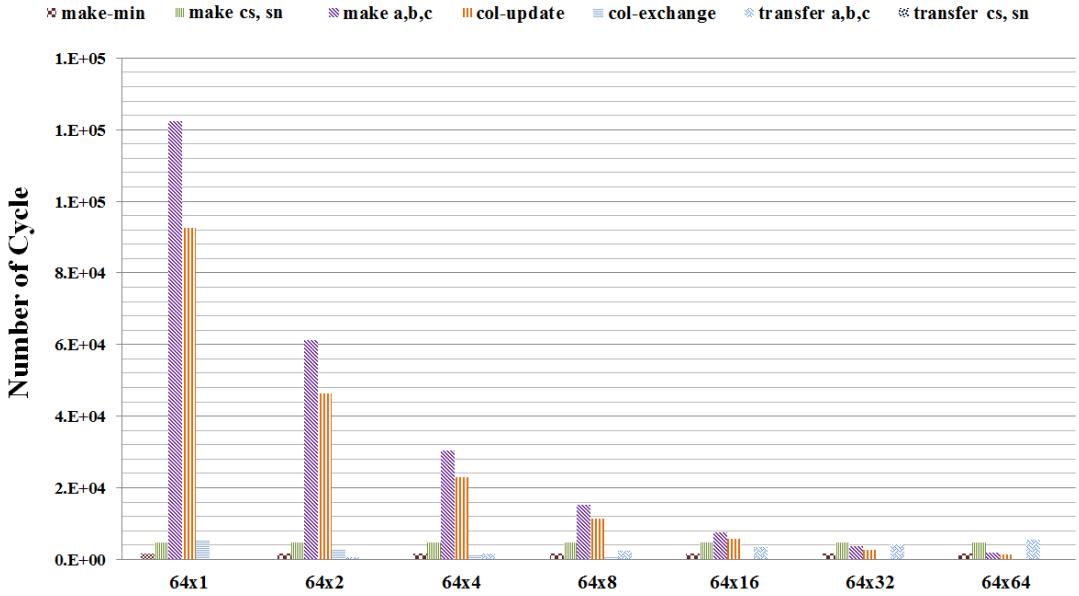

그림. 6. PE 구조별 사이클 수
Fig. 6. Number of cycles for each PE architecture

3. 시스템 면적 효율 및 에너지 효율

One-sided block Jacobi 알고리즘의 4가지 행렬(128x128, 64x64, 32x32 16x16)에 대한 PE 구조별 시스템 면적 및 에너지 소비는 표 3에 기술되어 있다. 시스템 면적과 에너지를 구하기 위해 Synopsis사의 design compiler (DC)를 이용하였으며 TSMC 28nm 공정을 사용하였다. 시스템 면적은 PE 수의 증가에 따른 면적이외의 PE간의 와이어 등으로 인해 PE 개수가 2배씩 증가할 때 시스템 면적은 2배 이상 증가한다. 하지만 에너지 소비량은 2배씩 증가하지 않는다. 이는 동일한 행렬 구조에서 서로 다른 PE 구조를 적용하더라도 특이치 분해 알고리즘을 수행하는데 요구되는 총 명령어 수는 거의 동일하기 때문이다.

그림 7과 그림 8은 각각 PE 구조별 에너지 효율과 시스템 면적 효율을 보여준다. 에너지 효율은 소비된 단위 에너지 당 처리량을 의미하며, 시스템 면적 효율은 단위 시스템 면적당 처리량을 의미한다. 세로축은 에너지 효율 및 시스템 면적 효율의 평균값으로 정규화 되어있기 때문에 세로축의 수치보다는 에너지 및 시스템 면적 효율이 나타내는 모양이 중요하다. 그림 7에서 보는바와 같이 각 행렬에 대한 최적의 에너지 효율을 보여주는 PE구조는 다음과 같다. 128x128 행렬은 PEs=64x16 구조에서, 64x64 행렬은 PEs=32x8 구조에서, 32x32 행렬은 PEs=16x4 구조에서, 16x16 행렬은 PEs=8x4에서 최적의 에너지 효율을 보인다. 에너지 효율은 특이치 분해 알고리즘의 수행에 따른 에너지 소비와 실행 시간에 영향을 받는다. 따라서 PE 수가 증가함에 따른 에너지 소비 증가에 비해 실행 시간의 감소가 더 크기 때문에 에너지 효율은 PE 수가 증가할수록 계속적으로 증가한다. 하지만 어떤 일정 개수의 PE 증가 이후로는 잦은 PE 간 내부 통신과 PE 활성화/비활성화 명령어의 증가로 인해 에너지 소비의 증가폭이 커지고 이로 인해 에너지 효율은 감소하는 경향을 보인다.

그림 8에서 보는바와 같이 각 행렬에 대한 최적의 시스템 면적 효율을 보여주는 PE구조는 다음과 같다. 128x128 행렬의 경우 PEs=64x8 구조에서, 64x64 행렬의 경우

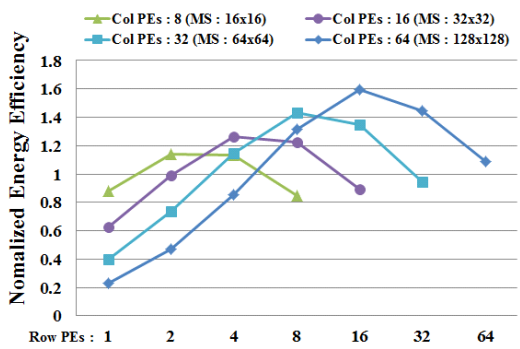

그림. 7. 정규화 된 에너지 효율
Fig. 7. Normalized energy efficiency

PEs=32x4 구조에서, 32x32 행렬의 경우 PEs=16x2 구조에서, 16x16 행렬의 경우 PEs=8x1 구조에서 최적의 시스템 면적 효율을 보인다. 에너지 효율과 유사하게 시스템 면적 효율 또한 특이치 분해 알고리즘의 수행에 따른 실행 시간과 시스템 면적에 영향을 받는다. 따라서 PE 수가 증가함에 따른 시스템 면적 증가에 비해 실행 시간의 감소가 더 크기 때문에 시스템 면적 효율은 PE 수가 증가할수록 계속적으로 증가한다. 하지만 어떤 일정 PE 개수의 증가 이후로는 잦은 PE 간 내부 통신과 PE 활성화/비활성화 명령어의 증가로 인해 실행시간의 증가폭이 커지고 이로 인해 시스템 면적 효율은 감소하는 경향을 보인다.

4. 상용 GPU와의 성능 비교

본 절에서는 선택된 멀티코어 시스템과 상용 고성능 GPU와의 성능 비교를 보여준다. 본 실험에서 사용된 GPU는 NVIDIA GTX 560으로 336개의 코어, 1GB 글로벌 메모리 및 48KB 공유 메모리를 가지고 있으며, 1,620MHz 동작주파수로 동작한다. 반면 멀티코어 시스템은 400MHz의 동작주파수로 동작한다. 상용 GPU와의 정확한 비교는 적정하지 않지만 본 절의 목적은 상용 GPU와의 비교를 통해 선택한 최적의 멀티코어 시스템의 잠재가능성을 알아보기 위함이다. 표 4는 서로 다른 행렬에 대해 상용 GPU와 멀티코어 시스템의 실행시간을 보여준다. 행렬의 크기가 커질수록 멀티코어 시스템이 더 높은 성능을 보여준다. GPU의 멀티스트리밍 프로세서 사이에서는 PE 간의 직접적인 통신이 불가능하기 때문에 one-sided block Jacobi 알고리즘 처리를 위해서 많은 수의 PE를 구성하게 되면 많은 오버헤드가 발생하기 때문이다.

표 4. 멀티코어 시스템과 상용 GPU와의 성능 비교
Table 4. Performance comparison between the selected multi-core system and a commercial GPU

Execution Time	GPU(GTX 560)	Selected multi-core
128x128	272.479 ms	37.36 ms
64x64	40.06 ms	15.03 ms
32x32	7.324 ms	5.98 ms
16x16	1.518 ms	2.31 ms

V. 결 론

본 논문에서는 다양한 애플리케이션의 특징 추출에서 널리 사용되고 있는 특이치 분해 알고리즘인 one-sided block Jacobi를 가속화하기 위하여 2차원 멀티코어 시스템을 이용하여 병렬 구현하였다. 또한 one-sided block Jacobi 알고리즘의 다양한 행렬 구조를 서로 다른 2차원 PE 구조에 구현하고 성능 분석한 결과를 바탕으로 각 행렬에 대해 최적인 2차원 PE구조를 탐색하였다. 또한 선택된 멀티코어 시스템을 상용 고성능 GPU와의 성능 비교를 통해 선택된 멀티코어 시스템이 one-sided block Jacobi 알고리즘 처리에 대해 잠재가능성이 있음을 보였다. 향후 본 논문에서의 멀티코어 모델을 FPGA 프로토타입 시스템에 구현하여 기능 및 성능을 재검증할 예정이다.

참고문헌

[1] E. Beltrami, "On bilinear functions," Journal of Mathematics, Vol. 11, pp. 98-106, 1873.

[2] C. Jordan, "Memory on bilinear forms," Journal of Pure and Applied Mathematics, Vol. 19, pp. 35-54, 1874.

[3] J. J. Sylvester, "A new proof that a general quadric may be reduced to its canonical form (that is, a linear function of squares) by means of a real orthogonal substitution," Messenger of Mathematics, Vol. 19, pp. 1-5, 1889.

[4] E. Schmidt, "On the theory of linear and nonlinear integral equations," Journal of Mathematische Annalen, Vol. 65, pp. 370-399, 1907.

[5] H. Weyl, "The asymptotic law granting the eigenvalues of linear partial differential equations with an application of the theory of black body radiation," Journal of Mathematische Annalen, Vol. 71, pp. 441-479, 1912.

[6] K. Fernando, H. Nicholson, "Identification of linear systems with input and output noise: the Koopmans-Levin method," IEE Proceedings. Control Theory and Applications, Vol. 132, pp. 30-36, 1985.

[7] Ake Björck, "A bidiagonalization algorithm for solving large and sparse ill-posed systems of linear equations," Journal of BIT Numerical Mathematics, Vol. 28, pp. 659-670, 1988.

[8] M. Darouach, M. Zasadzinski, S. J. Xu,

"Full-order observers for linear systems with unknown inputs," IEEE Transaction on Automatic Control, Vol. 39, No. 3, pp. 606-609, March 1994.

[9] R. G. King, M. W. Watson, "System reduction and solution algorithm for singular linear difference systems under rational expectations," Journal of Computational Economics, Vol. 20, pp. 57-86, 2002.

[10] A. Samui, S. R. Samantaray, "Wavelet singular entropy-based islanding detection in distributed generation," IEEE Transaction on Power Delivery, Vol. 28, No. 1, pp. 411-418, January 2013.

[11] W. Dong, G. Shi, and X. Li, "Nonlocal image restoration with bilateral variance estimation : a low-rank approach," IEEE Transactions on Image Processing, Vol. 22, No. 2, pp. 700-711, 2012.

[12] F. G. Yan, M. Jin, X. Qiao, "Low-complexity DOA estimation based on compressed MUSIC and its performance analysis," IEEE Transactions on Signal Processing, Vol. 61, No. 8, pp 1915-1930, 2013.

[13] S. C. Chan, Y. J. Chu, Z. G. Zhang, K. M. Tsui, "A NEW variable regularized QR decomposition-based recursive least M-estimate algorithm—performance analysis and acoustic applications," IEEE Transactions on Audio, Speech, and Language Processing, Vol. 21, No. 5, pp. 907-922, May. 2013.

[14] A Rajwade, A Rangarajan, A Banerjee, "Image denoising using the higher order singular value decomposition," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 35, No. 4, pp. 849-862, April 2013.

[15] Shutao Li, Leyuan Fang, Haitao Yin, "An efficient dictionary learning algorithm and its application to 3-D medical image denoising," IEEE Transactions on Biomedical Engineering, Vol. 59, No. 2, pp. 417-427, February 2012.

[16] A. Jindal, Mingyan Liu, "Networked computing in wireless sensor networks for structural health monitoring," IEEE Transactions on Networking, Vol. 20, No. 4, pp. 1203-1216, August 2012.

[17] G. Tang, A. Nehorai, "Stability of low-rank matrix reconstruction: A constrained singular value view," IEEE Transactions on Information Theory Society, Vol. 58, No. 9, 2012.

[18] G. H. Golub and C. Reinsch, "Singular value decomposition and least square solutions," Journal of Numerische Mathematik, Vol. 14, No. 5, pp. 403-420, Apr. 1970.

[19] J. Demmel, K. Veselic, "Jacobi's method is more accurate than QR," SIAM Journal on Matrix Analysis and Applications, Vol. 13, No. 4, pp. 1204-1245, 1992.

[20] B. A. Chartres, "Adaptation of the Jacobi method for a computer with magnetic-tape backing store," The Computer Journal, Vol. 5, No. 1, pp. 51-60, 1962.

[21] V. L. Charles, "The block Jacobi method for computing the singular value decomposition," Cornell University, 1985.

[22] B. B. Zhou, R. P. Brent, M. Kahn, "A one-sided Jacobi algorithm for the symmetric eigenvalue problem," in Proc. of 3rd Parallel Computing Workshop, 1994.

[23] B. B. Zhou, R. P. Brent, "A parallel ring ordering algorithm for efficient one-sided Jacobi SVD computations," Journal of Parallel and Distributed Computing, Vol. 42, No. 1, pp. 1-10, 1997.

[24] B. B. Zhou, R. P. Brent, "On parallel implementation of the one-sided Jacobi algorithm for singular value decompositions," in Proceedings of Euromicro Workshop on Parallel and Distributed Processing, pp. 401-408, 1995.

[25] Y. Takahashi, Y. Hirota, Y. Yamamoto, "Performance of the block Jacobi method for the symmetric eigenvalue problem on a modern massively parallel computer," in Proceedings of Algoritmy, pp. 151-160, 2012.

[26] I, Bethune, J. M. Bull, N. J. Dingle, N. J.

Higham, " Performance analysis of asynchronous Jacobi's method implemented in MPI, SHMEM and OpenMP," Manchster Institute for Mathematical Sciences School of Mathematics, 2012.

〔27〕 A. Gentile, D. S. Wills, "Portable video supercomputing," IEEE Transactions on Computers, Vol. 53, No. 8, pp. 960-973, 2004.

〔28〕 S. M. Kang, J. M. Kim, "Multimedia extension instructions and optimal many-core processor architecture exploration for portable ultrasonic image processing," Journal of Korea Society Computer Institute, Vol. 17, No. 8, pp. 1-10, 2012.

〔29〕 J. Y. Kim. D. K. Shon, J. M. Kim, H. S. Jun "Parallel implementation and performance evaluation of the SIFT algorithm using a many-core processor," Journal of Korea Society Computer Institute, Vol. 18, No. 9, pp. 1-10, 2013.

〔30〕 J. S. Seo, M. S. Kang, C. H. Kim, J. M. Kim, "Design space exploration of embedded many-core processors for real-time fire feature extraction," Journal of Korea Society Computer Institute, Vol. 18, No. 10, pp. 1-12, 2013.

저 자 소 개

박 용 훈

2012: 울산대학교
전기공학부 공학사.
현 재: 울산대학교
전기전자컴퓨터공학과 석사과정.
관심분야: 임베디드시스템,
컴퓨터 구조, 병렬 처리.
Email : ase018@naver.com

김 철 홍

1998 : 서울대학교 컴퓨터공학사.
2000 : 서울대학교 컴퓨터공학부 석사.
2006 : 서울대학교
전기컴퓨터공학부 박사
2005-2007년 :삼성전자 반도체총괄
책임연구원
2007-현재 : 전남대학교
전자컴퓨터공학부 교수
관심분야 : 임베디드시스템,
컴퓨터구조, SoC설계,
저전력 설계
Email: cheolhong@gmail.com

김 종 면

1995: 명지대학교
전기공학과 공학사.
2000: University of Florida
전기컴퓨터공학과 공학석사.
2005: Georgia Tech
전기컴퓨터공학과 공학박사
2005~2007: 삼성종합기술원
전문연구원
2007~현 재: 울산대학교
전기공학부 교수
관심분야: 임베디드 SoC, 병렬처리.
Email : jmkim07@ulsan.ac.kr