韓國 컴퓨터情報學會 論文誌
第8卷 第3號, 2003. 9.

2003-8-3-2-1

상미분 방정식을 위한 시스토릭어레이

박 덕 원*

A Systolic Array for Ordinary Differential Equations

Durk Won Park*

요 약

상미분 방정식은 물리학, 기계학, 전기학, 열역학 등에 많이 이용되는 방정식이나 수식이 복잡하고 처리 속도가 늦어서 실시간 처리에 어려움이 많다. 그래서 이 논문에서는 소프트웨어적인 방법으로는 많은 계산량으로 인하여 처리 속도가 떨어지므로 시스토릭어레이를 이용하여 Runge-Kutta 방법으로 상미분 값을 구하는 새로운 하드웨어를 제안하였다. 이 제안한 하드웨어는 처음 셀에서의 입력이 연속적으로 각 셀을 거치면서 처리되어 마지막셀에서는 상미분 값을 얻을 수 있다. 이렇게 처리함으로서 기존의 소프트웨어적인 방식에 비하여 수렴 속도도 빠르고 정확한 근사 값을 구할 수 있으므로 실시간 처리에 많이 이용될 수 있을 것이며, 기존의 다른 수치처리를 하는 하드웨어와 통합하여 사용될 수 있다. 이 논문에서는 제안한 하드웨어를 시뮬레이션하여 정확한 결과가 나오는 것을 확인하였다.

Abstract

An ordinary differential equation in analytical numerics is utilized to some applications, for example, physics, mechanical engineering, electrical engineering, thermodynamics and etc. But this equation has problems a lots to process in the real time processing by software method. This paper is proposed a systolic Arrays architecture for solving the Runge-Kutta method. it is one of method for solving an ordinary differential equation. the proposed its architecture is very high speed and regular. this hardware proposed in this paper may be part of the mathematical problem solver's tool kit in the future and may be available to many applications in the engineering .

▸ Keywords : Systolic Array, Ordinary Differential Equation, Runge-Kutta Method

* 세명대학교 컴퓨터과학과 부교수

Ⅰ. 서 론

수치해석에서 상미분 방정식은 물리학, 기계학, 전기학, 열역학 등에 많이 이용되는 방정식이나 수식이 복잡하고 처리 속도가 늦어서 어려움이 많았으나 최근 들어 가격이 저렴하면서도 계산속도가 빠른 소형 반도체 소자들을 생산할 수 있는 VLSI 기술이 급격하게 발전하고 있으며 이런 반도체 기술에 힘입어 소프트웨어적인 방법보다도 하드웨어적인 방법 등이 많이 이용되어 점차적으로 처리 속도가 빨라지고 있다. 그럼에도 불구하고 영상처리 등의 응용분야와 이들 분야의 기본처리 작업이 아주 중요하고 시간이 많이 걸리는 수치처리에서도 실시간 처리를 위한 요구가 많아지고 있다. 이런 요구에 부합하기 위해서 병렬처리에 대해서 많은 관심과 연구가 진행되어 왔고, 이 분야 또한 여러 가지 분야에서 괄목한 만한 성과를 가져왔다. 주로 영상처리, 음성인식, 로버트와 그 이외의 다양한 인공지능의 분야에서 많은 성과를 가져 왔다. 이렇게 병렬처리는 컴퓨터의 성능을 한 단계 높이고 실시간에 제약을 받던 작업들을 실시간 처리가 가능하도록 했다. 컴퓨터에 의한 수치처리는 아직도 하드웨어적인 기법보다는 소프트웨어적인 알고리즘을 의해서 처리되고 있다. 그러나 이러한 추세는 새로운 하드웨어의 기법과 많은 응용분야에서의 실시간 처리 요구에 의해서 하드웨어적인 방법으로 바뀌어 가고 있으며, 이에 따라서 국내외에서 많은 연구가 진행되고 있다. 그리고 이에 가장 적합한 하드웨어 기법 중에 하나가 시스토릭어레이(Systolic Array)라 할 수 있다. 어떠한 하나의 일을 처리하는데 있어서 전체 처리시간은 크게 입출력에서 걸리는 시간과 연산에 필요한 시간으로 볼 수 있는데 이들 중에서 입출력에서 걸리는 시간이 연산에 걸리는 시간보다 많은 그러한 일에 이 시스토릭어레이가 가장 적합하기 때문이다[1][4].

이 하드웨어 기법은 한번 메모리로부터 가져온 데이터는 필요한 요소 요소에서 모두 사용된 후에 그 결과 또는 본래의 데이터를 다시 메모리에 저장하므로 보통의 대역폭을 가지고 높은 처리 효율을 가져올 수 있다.

결국 Compute-bound computation에 적합하다. 컴퓨터를 이용한 수치처리에는 각각의 처리에 따라서 많은 알고리즘이 개발되어 사용되고 있으나 이러한 알고리즘을 실시간 처리를 위해서는 여러 가지 제약이 따른다. 특히 미분 방정식을 컴퓨터에 의해서 처리하는 경우에는 많은 연산횟수로 인하여 입출력의 문제도 있지만 계산과정이 복잡하고 처리시간이 많이 걸려서 다른 응용분야에 적용을 하기 위해서는 하드웨어적인 기법을 도입하는 것이 필수적이라 하겠다. 그래서 이 논문에서는 상미분 방정식을 수치처리 하는 방법 중 결과가 가장 정확하여 많이 이용되는 Runge-Kutta 방법이 수식이 복잡하고 많은 계산 량으로 인해 소프트웨어적인 알고리즘에 의해서는 실시간 처리를 하는데 있어서 처리속도가 떨어지므로 시스토릭어레이로 구현을 하였다[5][10].

Ⅱ. Runge-Kutta 상미분법

미분 방정식을 푸는 수치해법으로는 Taylor급수에 의한 방법, Euler의 방법, 수정 Euler의 방법, Runge-Kutta 방법 등이 간단하면서도 정확한 근사 값을 구하는 Runge-Kutta 방법이 가장 널리 사용되고 있다. 이 방법은 Taylor 급수를 몇 번째 항까지 취했느냐에 따라서 그 취해진 항의 최대 미계수에 따라 2차, 3차, 4차, 5차로 분류할 수 있으며 이것이 바로 Runge-Kutta 방법이라 할 수 있다. 이 방법은 고차 도함수의 계산을 요구하지 않으면서 Taylor 급수 접근의 정확도를 나타낼 수 있으므로 다른 방법에 비해서 많이 이용되고 있다.

Taylor의 급수는 식 (1)과 같이 y값의 1차, ... , n차 도함수 값 $y''(x), y'''(x), \ldots\ldots, y^{(n)}(x)$을 구해야 함으로 좋은 방법은 아니다. 그래서 식 (1)로부터 도함수를 포함하지 않은 식으로 유도할 수 있다.

$$y_{(i+1)} = y_i + h(y_i' + \frac{h}{2!} + \ldots. + \frac{h^{(n-1)}}{n!} y_i^{(n)}) \quad \cdots\cdots (1)$$

또한 Taylor의 급수에서 h값을 충분히 작게 하면 그 해는 오차가 아주 작아진다. 그래서 1차 미분 방정식의 해로서 Taylor의 급수의 제1차 항까지만 구하고 2차 항 이상을 절단하여 변형하면 식 (2)와 같은 Euler의 공식을 구할 수 있다.

$$y_{(i+1)} = y_i + hy_i \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (2)$$

이 공식은 실제 미분 방정식으로는 많이 사용되지는 않지만 이 식 (2)에서 y 값의 증분을 계산하는데 이것을 기울기를 사용하고 있다. 그런데 이 기울기로 구간의 시점과 종점에서의 기울기의 평균값을 얻을 수 있다. 이렇게 해서 얻어진 식 (3)이 수정된 Euler의 공식이다.

$$y_{(n+1)} = y_n + h\frac{(y'_n + y'_{n+1})}{2} \quad \cdots\cdots (3)$$

식 (3)은 $x_{(n+1)}$ 점에서의 y 값은 더 좋은 해가 된다. 그러나 이런 방식은 모두 도함수를 구해야 하는 어려움이 있으나 Runge-Kutta에 의해서 제안된 방법은 도함수를 구하지 않아도 된다.

Runge-Kutta 방법은 적은 횟수의 반복적인 계산으로도 즉, 소구간의 크기를 아주 작게 하지 않아도 정확한 근사 값을 구할 수 있다. 가장 일반적으로 이용되는 4계의 Runge-Kutta 방법은 식 (4)와 같다.

$$y_{(n+1)} = y_n + \frac{1}{6}(m_1 + 2m_2 + 2m_3 + m_4) \quad \cdots\cdots (4)$$

단,

$$\begin{pmatrix} m_1 = hf(x_n, y_n) \\ m_2 = hf(x_n + \frac{h}{2}, y_n + \frac{m_1}{2}) \\ m_3 = hf(x_n + \frac{h}{2}, y_n + \frac{m_2}{2}) \\ m_4 = hf(x_n + h, y_n + m_3) \end{pmatrix}$$

III. Systolic Array로 적용

1. 수치해석을 위한 시스토릭어레이

시스토릭어레이의 구성을 보면 간단한 셀들의 집합으로 구성이 되며, 이들 셀들은 간단한 몇 가지의 연산 기능만을 갖고 있다. 그래서 동일한 연산을 반복 수행하는 곳이나 입력이 여러 곳의 프로세서에서 이용되는 문제에 적합한 하드웨어라 할 수 있다. 특히 반복처리가 많고 수식이 복잡한 수치처리 등에 아주 적합하다고 할 수 있으며, 많이 이용되고 있다. 여기서 사용되는 셀을 내부 연산 스텝 프로세서라고 하며 이들 셀들은 이용되는 분야에 따라서 다양하게 구성된다. 많이 사용되는 구성은 그림1과 같으며 이런 구성은 메트릭스×벡터, L-U분해, Romberg 적분법 등에 이용된다[1][3].

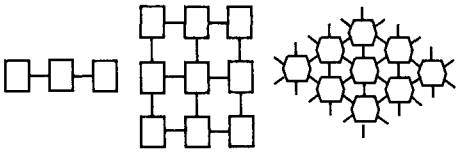


그림 1. 시스토릭어레이의 일반적인 구성
Fig.1. Various systolic array configurations

2. 4차의 Runge-Kutta 방법을 시스토릭 어레이로 적용

4차의 Runge-Kutta 방법을 시스토릭어레이로 적용을 하면 〈그림 2〉와 같이 나타낼 수 있다. 여기서 첫 번째 셀과 마지막 셀은 웨이트가 하나씩이며, 중간 셀 들은 2개의 웨이트를 갖는다. 이들 웨이트에는 상수 값이 들어가며 $w1$은 1/6 이 그리고 $w2$에는 2가 들어간다.

입력으로는 x_0, h_0, y_0 를 받아들여 마지막 셀에서는 최종 결과인 y값을 출력하게 된다. 셀의 구조를 보면 초기 셀, 중간 셀, l 마지막 셀의 구조가 다른데 이것은 〈그림 3〉, 〈그림 4〉, 〈그림 5〉와 같다.

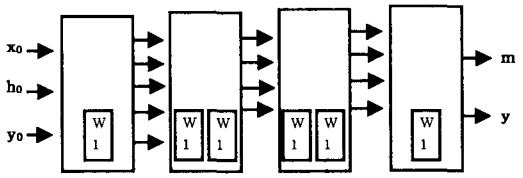


그림 2. Runge-Kutta 방법으로 상미분 방정식을 푸는 시스토릭어레이
Fig. 2. Systolic Array for solving Ordinary Differential Equations by Runge-Kutta methods

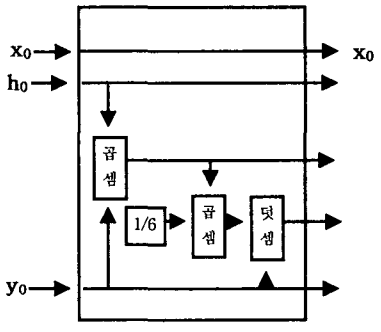


그림 3. 첫 번째 셀의 구조
Fig. 3. Design of a first cell

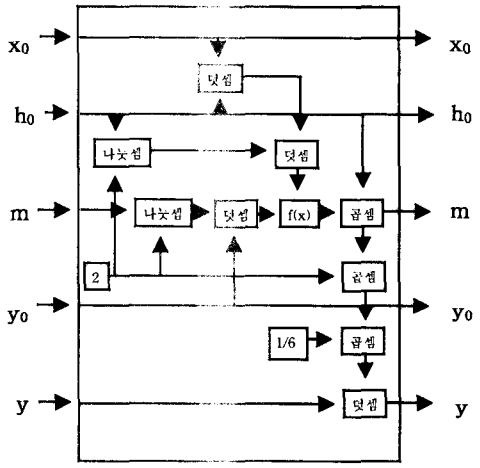
첫 번째 셀에서는 함수 값을 구하기 위한 x_0 와 h_0 그리고 초기 함수 $f(x)$값을 가지고 각 셀에서 계산된 결과를 누적시켜나가는 y_0값을 입력으로 받아들인다. 첫 번째 셀에서의 결과 x_0, h_0, m, y_0, y 는 두 번째 셀의 입력으로 들어가 두번 째 셀에서 필요로 하는 값을 계산하게 된다.

그 이외의 셀에서도 바로 전의 셀의 출력이 그 다음 셀의 입력으로 들어가게 된다.

〈그림 1〉에서 입력 스트림 x_0, $h_{n-1}, \ldots$ 는 계산하려고 하는 상미분 방정식이 n 개 인 경우

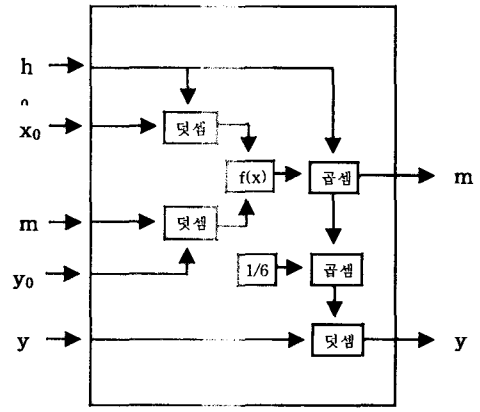
$$x_{n-1}, \ldots\ldots, x_2, x_1, x_0$$
$$h_{n-1}, \ldots\ldots, h_2, h_1, h_0$$
$$y_{y-1}, \ldots\ldots, y_2, y_1, y_0$$

와 같다. 이들 입력은 연속적으로 각각의 셀을 거치면서 처리되어 마지막 셀을 지나면 하나씩의 결과를 얻게 된다.



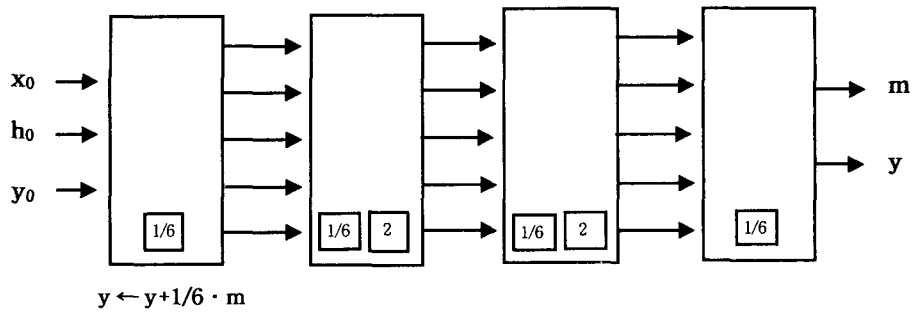
x0 ← x0　　h0←h0
m ← h0 · f(x0+h/2, y0+m/2)
y0 ← y0　　y←y+1/6 · 2m

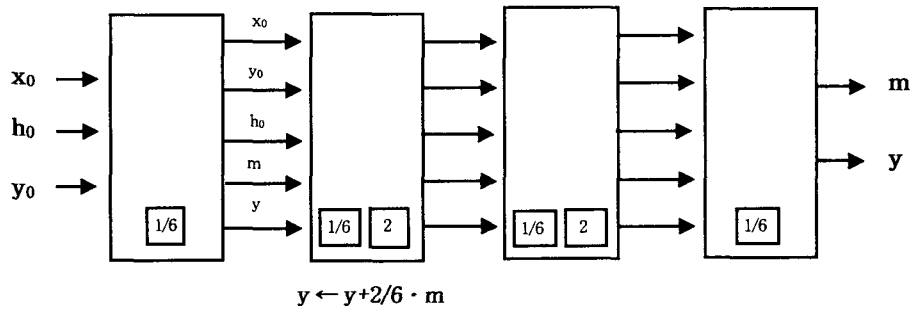
그림 4. 일반적인 셀의 구조
Fig. 4. Design of general cell

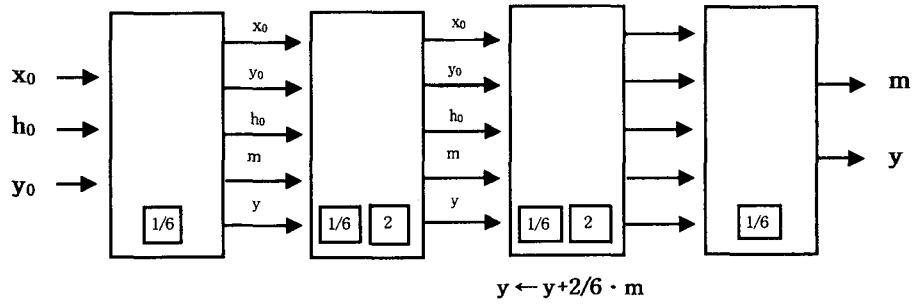


m ← h0 · f(x0+h, y0+m)
y = y+1/6 · m

그림 5. 마지막 셀의 구조
Fig. 5. Design of a final cell







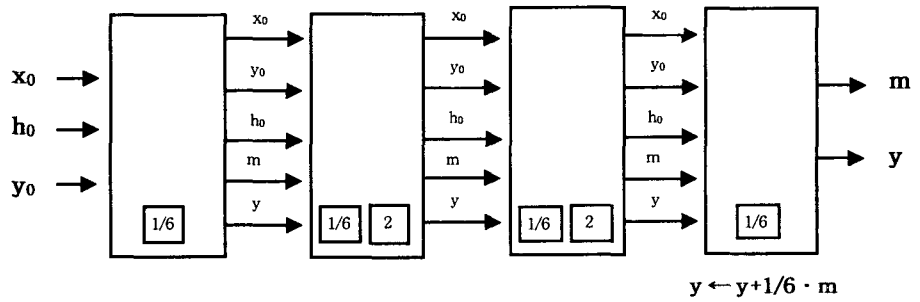


그림 6. 상미분 방정식을 푸는 시스토릭 어레이의 Snapshot
Fig. 6. Snapshot of systolic arrays for ordinary differential equations

이 시스토릭어레이에서 상미분 방정식의 값을 구하는 전과정의 Snapshot을 보면 그림6과 같으며, 마지막 셀에서 구해진 y값이 Runge-Kutta 방법으로 구한 최종 결과이다.

〈그림 6〉의 Snapshot에서 처음 셀 부터 마지막 셀까지 4단계를 거치면서 식 (5), 식 (6)이 계산이 된다. 첫 번째 셀에서는 식 (5)가 두 번째와 세 번째 셀에서는 식6이 그리고 마지막 셀에서는 식 (5)와 같이 계산되어진다.

$$y = y + \frac{1}{6} \cdot m \quad \cdots\cdots (5)$$

$$y = y + \frac{2}{6} \cdot m \quad \cdots\cdots (6)$$

결국은 식 (8)에서 계산된 값은 앞의 식 (4)에서의 결과와 같아진다.

Ⅳ. 시뮬레이션

이 논문에서 제안한 시스토릭어레이를 C언어를 이용하여 시뮬레이션하였다. 그 결과 상미분 방정식의 결과가 소프트웨어적인 방식보다 원하는 근사값을 빠르고 정확하게 구할 수 있었다. 여기서는 소수점 이하 5 자리까지 구하였으며, 시뮬레이션에서 시스토릭 부분의 중요 알고리즘은 〈표 1〉과 같다.

표 1. 상미분 방정식을 위한 알고리즘
Table 1. Algorithm for ordinary differential equations

```
INPUT  x0, h0, y0   REAL  x0, h0, y0
OUTPUT y

  Step1  x0 ← x0,  y0 ← y0
         h0 ← h0,  m ← h0*y0
         y ← y0+1/6 *m

  Step2 for I=1, 2  do step 3-4
     Step3  x0 ← x0,  y0 ← y0
     Step4  h0 ← h0,  m ← h0*f(x0+h/2, y0+m/2)
     Step5  y ← y0+1/6 *2m

  Step6  m ← h0*f(x0+h, y0+m)
         y ← y0+1/6 *m

  Step7 stop
```

Ⅴ. 결 론

이 논문에서는 수치해석에서 상미분 방정식을 푸는 방법은 많이 이용되는 것 중의 하나이나 계산량에 비해서 소프트웨어적인 방법으로 해결하면 처리 속도가 떨어져서 실시간 처리하는데 어려움이 많다. 또한 연속처리를 하거나 다른 복합된 문제를 해결하는 데는 이런 방식은 적합하지가 않다.

그래서 이 논문에서는 상미분 방정식을 푸는 방법 중에서 가장 널리 사용되는 4차 Runge-Kutta 방법을 시스토릭어레이로 해결하는 하드웨어를 제안하였다. 이 제안한 하드웨어는 소프트웨어적인 방법으로 시뮬레이션하여 정확하게 동작되는 것을 확인하였다.

이 논문에서 제안한 하드웨어는 소프트웨어적으로 상미분 방정식을 푸는데 걸리는 시간에 비해서 빠르므로 실시간 처리에 필요한 각종 응용분야에 적용이 가능하며, 수치해석의 문제를 해결하는 툴 키트의 한 부분으로서 이용이 가능하다. 그리고 이 하드웨어는 계속해서 이러한 방정식을 풀기 위해서 시스토릭어레이의 입력을 연속적으로 변화시키면서 입력하면 계속해서 새로운 상미분 방정식을 풀 수가 있다. 여기서 제안한 이 하드웨어는 수치해석의 다른 문제를 해결하는 다른 하드웨어와 함께 이용 가능하다[2].

이 논문에서의 보완해야 하는 부분으로서는 각 셀에서의 $f(x)$함수 값을 처리하는 부분을 좀더 보완하여야 하며, 각 셀에서 걸리는 사이클시간이 일정하지 않아서 전체 처리 시간이 길어질 수 있다는 것이다. 그리고 이 하드웨어가 잘 이용될 수 있도록 하기 위해서는 수치해석의 다른 문제를 해결하는 부분들이 시스토릭어레이로 제안되어 함께 이용되도록 해야 할 것이다.

참고문헌

[1] 박덕원, " Systolic Arrays를 이용한 경계선 검출에 대한 연구", 숭실대학교 대학원 석사학위 논문. 1987.

[2] 박덕원, "Romberg 적분법을 위한 Systolic Arrays", 한국OA학회, 제3권 제4호, pp55-6 1998.12.

[3] 박덕원, "영상처리의 국부적 연산을 위한 Systolic Arrays", 세명대학교, 세명논총, 제6집, 175-193. 1997.

[4] Mckeon, G. P., "Iterated Interpolation Using a Systolic Array", ACM Trans. on Mathematical Software, vol.12, No.2, 162-170. 1986.

[5] Kung, H. T., "Why Systolic Architecture", Computer Magazine 15(1), 37-46. 1982.

[6] Kai.Hwang and Faye A.Briggs, "Computer Architecture and Parallel Processing", Mcgraw. Hill Series in Computer Organization and Architecture, 769-807. 1984.

[7] Fisher, A. L. and Kung, H. T., "Synchronizing Large VLSI Processor Arrays", IEEE Trans. on Computers, vol. c-34, no.8. 1985.

[8] Kung, H. T and Picard, R. L., "One-dimensional Systolic Arrays for Multidimensional Convolution and Resampling", VLSI for Pattern Recognition and Image Processing, Spring-verlag, 9-24. 1984.

[9] Fisher, A. L., Kung, H. T. and Monier, L.M., "Architecture of the PSC : A Programmable Systolic Chip", Proceeding of the 10th Annual Symposium on Computer Architecture. 1983.

[10] Bridges, G. E., "Dual Systolic Architecture for VLSI Digital Signal Processing System", IEEE Trans. on Computers, vol. c-35, no.10, 916-923. 1986.

저 자 소 개

박 덕 원

1986 숭실대학교 전자계산학과 졸업(학사)

1988 숭실대학교 대학원 전자계산학과 졸업(석사)

1997 충남대학교 대학원 계산통계학과 졸업(박사)

1988 ~ 1991 대덕대학 조교수

1991~ 현재 세명대학교 컴퓨터과학과 부교수

〈관심분야〉 컴퓨터아키텍처, 영상처리, 병렬처리