# 재구성형 프로세서 성능과 레지스터와의 상관 관계 탐구

김용주, 허인구, 양승준, 이종원, 최영규, 백윤흥
서울대학교 전기컴퓨터공학부
e-mail : yjkim@optimizer.snu.ac.kr

# Performance exploration on the number of register for Coarse grained reconfigurable array processor

Yongjoo Kim, Ingoo Heo, Seungjun Yang, Jongwon Lee, Youngkyu Choi, Yunheung Paek
Dept. of Electrical Engineering, Seoul National University

**요 약**

재구성형 프로세서는 파워를 적게 사용하면서도 높은 성능을 낼 수 있는 프로세서이다. 재구성형 프로세서는 하드웨어에 최대한 많은 계산 자원을 넣으면서도 구조를 최대한 간단하게 하여 저전력 소모와 고성능을 동시에 추구하였다. 하지만 구조를 최대한 간단히 하는 과정에서 프로그램의 수행을 관리하는 많은 하드웨어 로직이 빠지게 되었는데, 이 부분은 컴파일러에서 코드를 생성할 때 스케쥴링과 수행 순서까지 정해지도록 소프트웨어적 관점에서 처리하기로 하였다. 이를 사용하기 위해 컴파일러는 입력된 프로그램을 분석하고 재구성형 프로세서에서 수행될 수 있는 형태로 코드를 각 계산자원에 매핑하는 작업을 수행해 주어야 한다. 재구성형 프로세서의 레지스터는 이 매핑 과정에서 데이터의 전달을 위해서 주로 사용되게 된다. 이 논문에서는 다양한 멀티미디어 응용 프로그램을 사용하여 멀티미디어 환경에서 재구성형 프로세서가 사용될 때 레지스터 개수가 성능에 미치는 영향을 제시한다.

## 1. 서론

모바일 매체를 통한 멀티미디어 활용이 폭발적으로 증가함에 따라 고성능, 저전력의 시스템에 대한 요구가 날이 갈수록 커지고 있다. 모바일 기기에서 수준 높은 멀티미디어 서비스에 대한 요구가 점점 커짐에 따라 모바일 기기 안에 탑제되는 비디오, 오디오 및 3D display 응용기술도 계속 발전하고 있다. 하지만 이러한 응용 프로그램의 경우에는 대부분 많은 계산 자원을 필요로 하므로, 제한된 전력과 계산자원이 제공되는 모바일 기기에서는 이에 맞는 최적화된 소프트웨어와 하드웨어를 사용하여야 할 필요가 있다.

이전까지는 이런 복잡한 요구를 맞추기 위해서 주문형 직접회로(ASIC)를 많이 사용하였다. 주문형 직접회로는 특정응용에 최적화되어 있는 하드웨어이므로 강력한 성능을 발휘하면서도 전력소모를 최소화할 수 있다. 하지만 주문형 직접회로의 경우 하드웨어를 특정 응용에 최적화 시켜서 제작하기 때문에 수행내용이 조금만 달라지더라도 재사용을 할 수 없고 새로 제작을 하여야 한다. 즉 응용이 달라질 때마다 다시 만들어야 하므로 제작시간이 오래걸리고 개발자에 많은 부담이 된다. 다양한 코덱이 사용되고, 점점 더 요구가 다양해지고 있는, 게다가 사용자들의 요구가 빠르게 바뀌는 멀티미디어 환경에서 이러한 점은 커다란 약점이 될 수 밖에 없다.

모바일 환경에서 프로세서와 주문형 직접회로 방식의 대안으로 제시된 것이 바로 재구성형 프로세(Coarse Grained Reconfigurable Array processor, CGRA)[1-3]이다. 재구성형 프로세서는 일반 프로세서에 비해 많은 수의 계산 자원 (Processing Element)을 갖고 있기 때문에 높은 수준의 병렬 연산을 수행할 수 있다는 특징을 가지고 있으며 설정 메모리(configuration memory)가 존재하여, 각 계산 자원의 수행 방식을 매 cycle 마다 바꿔줄 수 있다. 이런 점 때문에 높은 성능을 유지하면서도 유연한 응용프로그램의 수행이 가능하여 주문형 직접회로의 대안으로 꼽히고 있다.

하지만 재구성형 프로세서는 성능 강화와 파워소모 절약을 위해서 일반적인 프로세서들이 가지고 있는 다양한 하드웨어 회로를 포기하였다. 대신 그 생략된 하드웨어에서 처리해야 할 부분(데이터 관리, 코드 스케쥴링 등)을 컴파일러에게 분담을 하였다. 이에 컴파일러에서는 재구성형 프로세서에서 수행되는 응용을 작은 작업으로 나누어 각 계산자원에게 분배하고 수행순서를 정하는 작업을 처리해야 한다. 이 과정에서 데이터의 의존성(dependency)을 어기지 않으면서 데이터를 효율적으로 전달하기 위해서 계산자원 일부를 데이터 전달에 사용하게 된다. 이때 각 계산자원에 레지스터가 있을 경우 데이터 전달을 레지스터를 통해서 함으로써 계산자원 낭비를 줄일 수 있다. 하지만 레지스터의 크기를 늘리는 것은 하드웨어 제작비

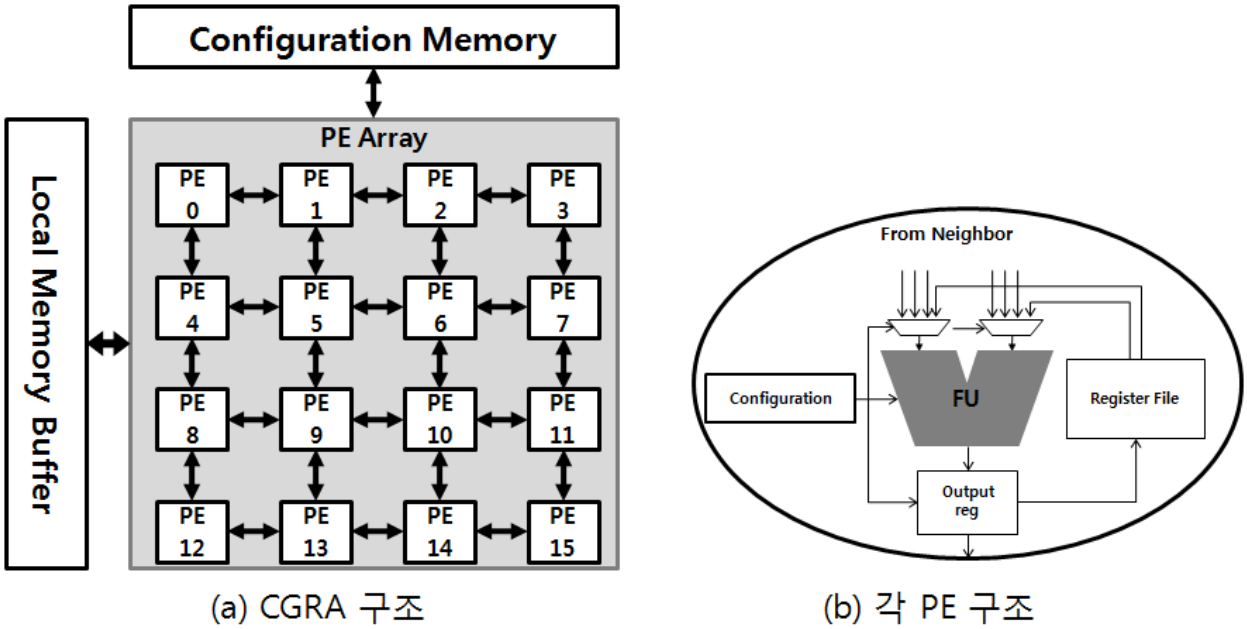


**그림 1 CGRA 의 구조와 각 PE 의 구조**

용을 증가시키기 때문에 최적화된 레지스터의 갯수를 찾아내는 것이 중요하다. 이 논문에서는 임베디드 환경에서 많이 쓰이는 멀티미디어 응용에서 최적화된 레지스터 갯수를 제안한다.

2 장에서는 재구성형 프로세서의 구조에 대해서 좀 더 자세히 설명한다. 3 장에서는 재구성형 프로세서를 위한 최신 스케쥴링 알고리즘과 레지스터 갯수의 상관관계에 대해서 설명하고 4 장과 5 장에서는 실험결과와 결론을 제시한다.

## 2. 재구성형 프로세서의 구조

재구성형 프로세서(Coarse Grain Reconfigurable Array architecture)는 PE(Processing Element)가 어레이 형태로 모여서 그물망 네트워크 형태를 띠고 있다. PE 는 기본적으로 산술이나 논리 연산을 수행할 수 있고, 곱셈회로나 메모리 접근 회로가 달려있는 경우에는 곱셈이나 load/store 등의 다른 연산도 수행할 수 있다. 기능이 다른 PE 가 2 가지 이상 있는 경우 이종의 재구성형 프로세서라고 부른다. PE 는 주변의 다른 PE 와 연결되어 있으며, 그 연결을 사용해서 서로 데이터를 주고 받을 수 있다. 일반적으로 PE 는 자신의 상하좌우에 있는 4 군데의 PE 와 연결된다. 그리고 응용의 복잡도에 따라 대각선에 있는 PE 와도 연결되는 등의 연결이 늘어나는 경우도 있다. 재구성형 프로세서의 경우 PC(program counter)가 없고 그와 관련된 제어 연산자도 없다. 대신 PE 의 동작을 제어해 주는 configuration 이 있다. Configuration 은 재구성형 프로세어용 instruction 으로 표현할 수 있는데, PE 어레이의 동작을 지정해 놓은 설정을 뜻한다. Configuration 을 바꾸게 되면 PE 의 동작 전체가 바뀌게 된다. 이것은 컴파일러에서 만들어지며, 만들어진 configuration 은 configuration 메모리에 저장되어 필요할 때마다 PE 의 설정을 바꾸는데 사용된다. Configuration 메모리 이외에도 재구성형 프로세서에는 data 를 저장하는 지역 메모리(local memory)가 존재한다. 그림 1 은 재구성형 프로세서의 구조를 보여준다. 그림 1. (a)는 전체적인 재구성형 프로세서의 구조를 보여주고 (b)는 각 PE 의 구조를 보여준다.

## 3. 응용프로그램의 스케쥴링과 레지스터

재구성형 프로세서는 바로 전 section 에서 설명한 것처럼 일반적인 프로세서와 구조가 많이 다르다. 그래서 응용의 컴파일도 다른 식으로 적용 해야 한다. 일반적으로 재구성형 프로세서는 loop 의 반복되는 작업을 최대한 빠르게 처리하기 위해서 사용한다. 그래서 loop 의 내용 부분을 DFG(data flow graph)로 변환해서 재구성형 프로세서에 작업을 할당하게 된다. DFG 는 계산 내용을 표현하는 node 와 의존성을 표현하는 edge 로 구성되며, 각 node 에서 계산된 결과값이 다음에 어떤 node 로 가서 사용되는지 데이터가 흐르는 것처럼 표현이 된다. DFG 는 node 들과 그 사이의 의존성이 명확히 들어나기 때문에, 의존성이 없는 node 들을 쉽게 찾을 수 있다. 스케쥴러는 이렇게 찾은 병렬성을 재구성형 프로세서에서 최대한 동시에 수행시키도록 operation 을 스케쥴한다

병렬성을 최대한 늘리기 위해서는 한 cycle 에 최대한 많은 계산이 동시에 수행되어야 한다. Software pipelining 은 loop 의 반복을 pipeline 시켜서 최대한 많은 계산이 동시에 수행되도록 만드는 것이며 모듈로 스케쥴링은 software pipeline 알고리즘중 가장 잘 알려진 스케쥴링 알고리즘이다.[4] 모듈로 스케쥴링의 목적은 loop 의 하나의 반복이 수행될 수 있는 시간인 II(initiation interval)를 최대한 줄이면서 스케쥴링하는 것이다. 모듈로 스케쥴링은 먼저 MII(minimum II)를 구한 후 구해진 최소 II 부터 스케쥴링을 수행한다. MII 는 resMII 와 recMII 의 최대값을 이용하여 구한다.

DFG 의 node 의 개수를 계산하여 DFG 스케쥴링에 필요한 최소한의 자원를 확보할 수 있는 최소 II 를 ResMII(resource constrained lower bound) 이라고 정의한다. RecMII(recurrence constrained lower bound)는 loop 의 반복 간의 데이터 의존성을 분석하여 의존성이 지켜질 수 있도록 하는 최소한의 II 를 구할 수 있는데 이 값이 recMII 가 된다. 모듈로 스케쥴링은 이렇게 구한 MII 부터 스케쥴링을 수행하게 된다. 만약 스케쥴링이 실패할 경우에는 II 의 값을 1 증가시키고 다시 스케쥴링을 시도하는데, 이 시도는 스케쥴링이 성공할 때까지 반복된다.

재구성형 프로세서에서의 모듈로 스케쥴링은 재구성형 프로세서의 특성상 다른 프로세서에 적용할 때와 약간 다르게 적용되어야 한다. VLIW 등의 다른 병렬 프로세서들은 데이터의 전달에 제한이 없지만 재구성형 프로세서는 각 PE 마다 연결이 한정되어 있기 때문에 데이터를 전달하고자 하는 목적 PE 로 연결되어 있지 않을 수가 있다. 이 경우에는 데이터를 몇 개의 PE 를 거쳐서 목표 PE 까지 전달할 수 있는데 이것을 라우팅 이라고 한다. 하지만 라우팅은 data 를 전달하는데 사용되는 PE 들이 다른 작업을 수행할 수 없기 때문에 PE 자원이 많이 소모된다. 또한 목표 PE 로 전달 하는 데에는 물리적인 거리로 나타나는 공간적인 제약 이외에도 목표하는 시간에 도착해야만 하는 시간적인 제약도 있다. 또한 이종의 PE 도 있기 때문에 자원적인 제약도 있다. 재구성형 프로세서에서는 데이터의 전달을 좀 더 쉽게 하기 위하여 레지스터를 데이터를 전달하는 용도로 사용한다. 일반적인 프로세서용 컴파일러의 경우 코드를 생성한 다음에 레지스터 할당을 나중에 수행하게 된다. 하지만 최근에 나온 재구성형 프로세서용 스케쥴링 알고리즘[5] 대부분이 각 계산을 PE 에 할당하면서 데이터 이동경로를 바로바로 계산을 하게 된다. 즉 코드 스케쥴 단계에서 PE 의 소모를 줄이기 위해서 레지스터를 사용하게 되는데, 레지스터가 많을 경우 PE 를 다른 계산을 위해서 아껴놓을 수가 있다. 하지만 레지스터를 너무 많이 늘릴 경우 하드웨어 비용이 많이 증가하기 때문에 이 레지스터의 갯수를 최적화 할 필요가 있다.

## 4. 실험

재구성형 프로세서는 그림 1 에서 나온 것과 같은 4x4 이종 어레이 프로세서를 사용하였다. 각 PE 는 모두 multiplier unit 을 가지고 있고, 메모리 접근은 4 개의 PE 만 가능하다고 가정하였다. 실험에 사용한 모듈로 스케쥴링 알고리즘은 가장 최근에 나온 휴리스틱 알고리즘을 사용하였다.[5] 응용프로그램은 임베디드 시스템에서 많이 쓰이는 멀티미디어 응용 프로그램의 핵심 반복문을 사용하였다. 각 응용프로그램에 대해서 레지스터의 숫자를 변화시키면서 실험을 측정하였는데, 각 PE 마다 각각 레지스터 파일이 있는 것을 가정하였고 레지스터 파일의 크기는 0, 2, 4 로 측정하였다.

표 1 은 레지스터 파일 크기 변화에 따른 성능 변화를 보여준다. 각 수치는 모듈로 스케쥴링을 하였을 때의 계산된 II 의 값을 나타낸다. 각 결과값은 10 번 측정한 후 결과값을 평균값을 구했다. x 는 데이터의 routing 이 PE 를 과도하게 소모함으로 인해 PE 의 부족이 발생해 결과값이 제대로 구해지지 않는 상황을 나타낸다.

| | 레지스터 사이즈 | | | |
|---|---|---|---|---|
| | 0 | 2 | 4 | 6 |
| Wavelet | x | 4 | 4 | 4 |
| Compress | 3 | 3 | 3 | 3 |
| GSR | 4 | 3.7 | 3.4 | 3.4 |
| Swim1 | x | 8 | 7.5 | 7.5 |
| Swim2 | x | 9 | 8.6 | 8.5 |
| Swim3 | 8.6 | 5 | 5 | 5 |

**표 1 레지스터 숫자 변화에 따른 II 값의 변화**

실험결과 레지스터 사이즈가 늘어나면서 성능이 점점 좋아지다가 어느 숫자 이후로는 성능의 증가가 없을을 발견할 수 있었다. 전체적으로 보았을 때 레지스터 4 개까지는 성능이 좋아졌으며 6 개부터는 성능 증가가 없음을 확인할 수 있었다.

## 5. 결론

재구성형 프로세서는 파워를 적게 사용하면서도 높은 성능을 낼 수 있는 프로세서이다. 하지만 이를 사용하기 위해서는 컴파일러에게 과도한 스케쥴링 부담이 주어지는 문제가 있었다. 이는 각 PE 에 레지스터를 만듬으로써 완화시킬 수 있는데 이 갯수를 너무 늘릴 경우 하드웨어 비용이 늘어나는 부담이 있었다. 이 논문에서는 여러가지 응용에 대해서 레지스터 사이즈를 조절하며 최적값을 탐구함으로써 멀티미디어 환경에서 재구성형 프로세서를 제작할 경우에 도움이 되는 정보를 제공하였다.

**Acknowledgement**

본 연구는 교육과학기술부/한국과학재단 우수연구센터 육성사업(과제번호 2009-0063249), 서울시 산학연 협력사업(10560), 2009 년도 정부(교육과학기술부)의 재원으로 한국과학재단의 국가지정연구실사업(No.2009-0083190)의 지원을 받아 수행되었습니다.

## 참고문헌

[1] Singh, H., Lee, M.-H., Lu, G., Bagherzadeh, N., Kurdahi, F., Filho, E.: Morphosys: An integrated reconfigurable system for data-parallel and computation-intensive applications. IEEE Trans. Comput. 49(5), 465–481 (2000)

[2] B. Bougard, B. De Sutter, D. Verkest, L. Van der Perre, and R. Lauwereins. A coarse-grained array accelerator for softwaredefined radio baseband processing. IEEE Micro, 28(4):41–50, 2008.

[3] Y. Kim, M. Kiemb, C. Park, J. Jung, and K. Choi. Resource sharing and pipelining in coarse-grained reconfigurable architecture for domainspecific optimization. In *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*, pages 12–17, Washington,

DC, USA, 2005. IEEE Computer Society.
[4] B. R. Rau. Iterative modulo scheduling: An algorithm for software pipelining loops. In Proc. of the 27th Annual International Symposium on Microarchitecture, pages 63–74, Nov. 1994.
[5] Park, H., Fan, K., Mahlke, S., Oh, T., Kim, H., Kim, H.: Edge-centric modulo scheduling for coarse-grained reconfigurable architectures. In: PACT 2008, pp. 166–176. ACM, New York (2008)