논문 2008-45SD-4-16

규칙적인 NoC 구조에서의 네트워크 지연 시간 최소화를 위한 어플리케이션 코어 매핑 방법 연구

(Application Core Mapping to Minimize the Network Latency on Regular NoC Architectures)

안 진 호*, 김 홍 식**, 김 현 진**, 박 영 호***, 강 성 호**

(Jin-Ho Ahn, Hong-Sik Kim, Hyunjin Kim, Youngho Park, and Sungho Kang)

요 약

본 논문에서는 규칙적인 형태의 NoC 중 mesh 구조를 기반으로 한 어플리케이션 코어 매핑 알고리즘 연구 내용을 소개한다. 제안된 알고리즘은 ant colony optimization(ACO) 기법을 이용하여 주어진 SoC 내장 코어 및 NoC 특성 정보를 대상으로 가장 효과적인 코어 배치 결과를 도출한다. 설계 목적으로 사용된 네트워크 지연 시간 측정을 위해 평균 홉 수 계산 결과를 이용하였으며 제한 조건으로는 NoC 대역폭을 기준으로 하였다. 12개의 코어로 구성되는 실제 기능 블럭을 대상으로 실험한 결과 계산 시간이나 매핑 결과 모두 우수함을 확인할 수 있었다.

Abstract

In this paper, we propose a novel ant colony optimization(ACO)-based application core mapping method for implementing network-on-chip(NoC)-based systems-on-chip(SoCs). The proposed method efficiently put application cores to a mesh-type NoC satisfying a given design objective, the network latency. Experimental results using a functional circuit including 12 cores show that the proposed algorithm can produce near optimal mapping results within a second.

Keywords: NoC, Application Core Mapping, Ant Colony Optimization

I. 서 론

Network-on-Chip(NoC)는 지금까지의 온칩연결구조와 달리 칩의 적용 분야 및 기능에 따라 매우 다양한 형태로 구현이 가능하기 때문에 전체 시스템 개발 시간 및 용도 별로 최적화된 구조를 위해서는 플랫폼 기반 설계 방식과 자동화가 매우 중요하다. 특히 멀티프로세서 시스템과 같이 범용 동작을 목표로 하는 시스템이 아니고 특정 목적과 기능을 가지는 시스템을 NoC로 구현할 경우 더욱 중요하다고 할 수 있다. 따라서 표준화되고 효율적인 NoC 기반 시스템 설계 과정 구축과 이를 지원하는 각종 툴의 개발은 NoC 기반 시스템의 성공 유무를 결정하는 핵심 연구 분야이다. NoC는 설계 고려 사항이 굉장히 많기 때문에 필요한 기능 및 구조를 제한된 개발 시간 하에서 효과적으로 도출하기 위해서는 상위 수준의 NoC 모델링을 얼마나 정확하게 그리고 빠르게 할 수 있는가가 중요하다[1~3]. 상위 수준의 NoC 개발은 C, C++, XML 혹은 SystemC와 같은 상위 수준의 언어를 이용한 NoC의 생성 및 검증 단계를 의미한다. 상위 수준 NoC 개발의 최초 과정은 NoC와 내장 코어 기술(description) 수준이나 방법, 벤치마크, 입출력 데이타의 형태나 내용 등에 대한 정의이다. NoC를 어떤 수준에서 어떻게 표현해야 하는지 그리고 어떤 내용이 포함되어야 하는지에 대한 사전 정의가 선행되

* 정회원, 호서대학교 전자공학과
(Hoseo Univ., Electronic Engineering)
** 정회원, 연세대학교 전기전자공학과
(Yonsei Univ., Electrical&Electronic Engineering)
*** 정회원, 한국전자통신연구원 NoC 기술팀
(ETRI, Network Tech. Lab., NoC Tech. Team)
접수일자: 2008년2월14일, 수정완료일: 2008년3월27일

고 이후 NoC의 입출력 파라미터, 예를 들면 NoC상의 패킷 형태 및 트래픽 특성에 대한 세부적인 결정이 필요하다. 이러한 선행 단계가 완료되면 두 번째 과정으로 NoC 모델링 플랫폼을 구축해야 한다. 모델링 플랫폼은 mapper, scheduler, compiler 그리고 simulator를 통칭하는 것으로 NoC 자동 생성에 필요한 관련 툴 또는 방법을 의미한다. Mapper란 SoC에 내장되는 여러 블럭들을 NoC 상에 할당하는 것으로 최적화를 위해서는 NoC 생성과 동시에 이루어져야 한다. 하지만 문제의 단순화를 위해서 두 문제를 분리하여 처리할 수 있다. Scheduler는 현재 구성에서 최소의 리소스만으로 전체 패킷 처리량(throughput)을 최대화시키고자 하는 것이다. 이 결과에 따라 기존의 매핑 과정을 다시 취할 수 있다. Compiler란 NoC 기술 수준을 변경하는 것으로 상위 수준의 NoC 모델을 합성 가능한 RTL 또는 SystemC수준의 형태로 변형하는 것을 포함한다. 마지막으로 simulator는 결정된 NoC 모델을 가상적으로 시뮬레이션하는 것으로 성능 평가 및 예측에 사용한다. 그림 1에서 상위 수준 NoC 개발의 전체적인 과정을 나타내었다.

일단 생성된 NoC는 실제 코어 모델을 NoC에 할당하고 task 단위로 스케줄링하여 해당 결과가 설계 목적에 부합되는지를 확인한다. 설계 목적과 부합되지 않으면 다시 매핑 및 스케줄링 과정을 반복하는데 일정 횟수까지 목표를 만족시키지 못하면 다시 상위로 올라가서

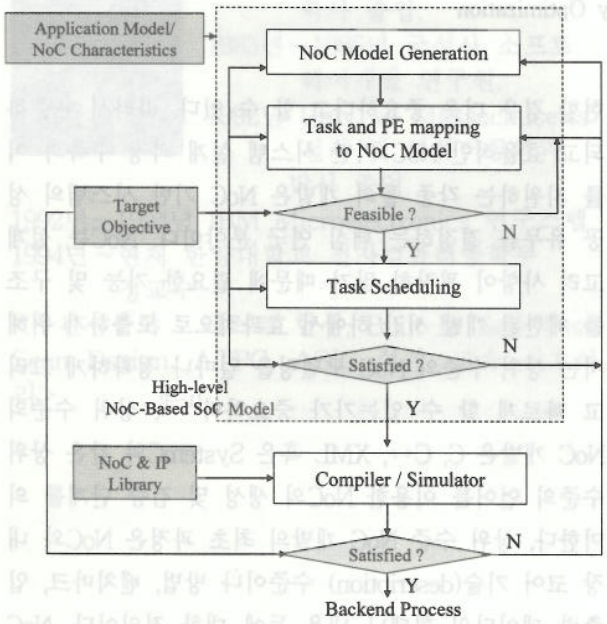

그림 1. 상위 수준 NoC 개발 과정
Fig. 1. High-Level NoC Design Flow.

NoC 파라미터들을 수정하여 새로운 NoC 모델을 만들어 다시 그림 1의 과정을 수행한다.

본 연구에서는 이러한 NoC 설계 과정 중 가장 포괄적이며 NoC의 장점을 최대화할 수 있는 NoC 구조에 어플리케이션 코어들과 여러 작업 등을 매핑하는 방법을 개발한다. NoC 매핑 및 스케줄링은 NP-complete 문제 수준의 복잡도를 가지고 있기 때문에 많은 계산 시간이 요구되므로 효과적인 근사화 알고리즘이 필요하다[4~5].

II. 기존 연구의 분석

NoC 개발을 위해 제일 먼저 선행되어야 할 것은 구현 과정에서 고려되어야 할 여러 요구 조건들이다. 이러한 요구 조건에는 설계자가 판단하고 결정할 설계 목적(design or user objective)들과 설계 제한조건(design constraint), 그리고 내장되는 코어들의 특성(core characteristic) 등이 포함된다. 상기 조건들은 상호 의존적으로 하나의 조건이 다른 조건들에 미치는 영향이 적용 범위 및 정도에 따라 다양하기 때문에 많은 가정 및 선험적인 접근이 필요하다. 예를 들면 어떤 코어가 초당 100Mbyte정도의 대역폭을 필요로 한다면 이를 만족시키기 위해서는 NoC의 동작 속도와 링크 폭 등의 제어 조건이 있다. 그 중 동작 속도를 조절하면 NoC의 전력 소모량, 타이밍 조건 등이 변화하고 이에 따라 NoC의 레이아웃이나 네트워크 연결부의 구조 등이 가변될 수 있다. 만약 NoC의 레이아웃이 변경되면 데이터 패킷의 평균 라우팅 거리가 바뀌기 때문에 NoC의 전체 throughput이 변경된다. 이러한 throughput에 대한 제어 조건으로 다시 NoC의 동작 속도나 링크 폭 등이 사용될 수 있으므로 결국 무한 반복 과정이 발생한다. 따라서 문제의 단순화를 위해 일부 조건은 고정하는 방식으로 접근해야 한다[1].

최근 수년간 NoC 개발 과정에서 발생하는 다양한 문제들을 제기하며 이에 대한 효과적인 개선 방안들이 꾸준히 발표되고 있다[1~5]. 그러나 아직까지는 많은 제한 요소 및 가정을 전제로 하는 적용 분야별 혹은 NoC 토폴로지 단위의 성능 해석 및 관련 개발 방법 이외에는 일반적으로 통용될 수 있는 방법론이 아직은 전무한 상황이다.

NoC 매핑을 위해 사용되는 설계 목적으로는 평균 홉 수, 전송 에너지 등을 주로 사용하며 최근에는 복수개의 평가 기준을 적용시킨 경우나 실제 사용 시 발생

하는 다양한 시스템의 기능에 따라 매핑 결과를 변화시키는 아이디어도 발표되었다[6~7]. 그러나 코어의 크기나 동작이 확연히 다른 경우 이웃한 두 노드의 라우터는 전송 지연 시간과 전력 소모량이 크게 달라진다. 따라서 정확한 지연 시간 및 전력 소모량을 계산하기 위해서는 매핑 과정에서 레이아웃에 대한 정보도 추가하는 것이 보다 정확한 결과를 도출하기 위하여 필요하다[5]. 그러나 이에 따른 알고리즘의 복잡도는 추가되는 정보의 양에 비례해 증가할 것이다. 또한 열적 평형 문제도 함께 고려하는 것이 필요한데 이것은 고밀도 구조에서 발생하는 hotspot으로 인한 열적 불평등이 초래하는 하드웨어의 수명 단축 및 오동작 가능성을 줄이기 위함이다.

최적화된 NoC 구성을 위해서는 코어 사이의 데이타 전송 스케쥴링과 코어 매핑을 묶어서 처리하는 것이 바람직하다. 그러나 태스크 단위의 스케쥴링 자체가 너무 복잡하기 때문에 먼저 코어간 데이타 전송양 및 시간을 고정시킨 이후에 매핑을 실행하고 그 결과에 따라 미리 고정한 스케쥴링을 수정하고 이후 같은 과정을 반복하면서 목표로 하는 결과를 유도하는 점진적인 방법을 사용하고 있다.

Ⅲ. ACO 기반 어플리케이션 코어 매핑

1. 문제의 정의(Problem Formulation)

매핑 문제를 어플리케이션 코어 그래프, $G(V,E)$로 정의하면 $v_i \in V$는 코어, $e_{ij} \in E$는 코어 v_i와 v_j 사이를 연결하는 링크로서 연결된 코어 사이의 communication이 있음을 나타낸다. 그리고 e_{ij}의 값은 v_i와 v_j사이의 communication 양을 의미한다. 또한 mesh 구조의 NoC는 $M(N,L)$로 나타내며 $n_i \in N$는 NoC상의 노드, $l_{ij} \in L$은 노드 n_i와 n_j 사이를 연결하는 물리적인 선이 있음을 나타낸다. 그리고 l_{ij}의 값은 해당 링크의 대역폭을 의미한다. 상기 내용을 기반으로 어플리케이션 코어 매핑을 정의하면 식 (1)과 같다.

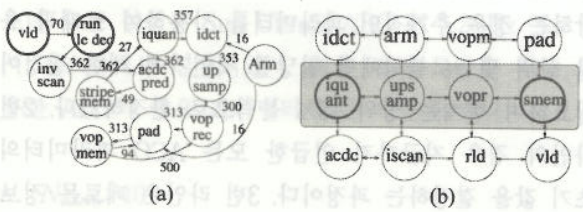

그림 2. 어플리케이션 코어의 매핑 예 [8]
Fig. 2. Application Core Mapping Example.

$$map: G \rightarrow M, s.t. map(v_i) = n_j, \quad \forall v_i \in V, \exists n_j \in N \tag{1}$$

단 매핑은 $|V| \leq |M|$일 때 성립하며 NoC 대역폭 제한 조건을 정의하면 다음과 같다.

$$\sum_{k=1}^{|E|} C_{ij}^k \leq |l_{ij}|, \forall i,j \in N \tag{2}$$

$$C_{ij}^k = \begin{cases} |e^k|, & if\ C_{ij} \in Path(e^k) \\ 0, & otherwise \end{cases}$$

식 (2)에서 C_{ij}는 어플리케이션 코어 그래프 중 임의 두 코어 사이의 communication e^k $(1 \leq k \leq E)$가 NoC의 노드 n_i와 n_j 사이를 경유함(=$Path(e^k)$)을 가리키고 대역폭 제한 조건을 만족하기 위해서는 NoC상의 임의 링크 l_{ij} $(1 \leq i,j \leq N)$를 경유하는 모든 communication의 총 합은 해당 링크의 최대 대역폭을 초과해서는 안된다는 의미이다.

그림 2에서 어플리케이션 코어 매핑의 예를 들었다. 그림 2 (a)는 시스템 실제 구성도에 대한 어플리케이션 코어 그래프를 나타낸다. 그림에서 화살표 및 숫자는 각각 두 코어 사이에 전송되는 데이타의 방향 및 양(MB/s)을 가리킨다. 그림 2 (b)는 (a)를 mesh 구조의 NoC에 매핑한 결과이다.

2. ACO 파라미터의 정의

ACO는 개미로 모델링된 agent들의 해집합 탐색 결과가 페로몬 테이블로 누적되고 이러한 페로몬은 다음 agent들의 탐색 시 기초 정보가 되어 보다 우수한 해를 선택하는 agent들을 점차로 증가시켜 어느 정도의 시간이 흐르면 결국 대부분의 agent들이 하나의 해로 수렴하는 형태로 최적 값을 찾는 방식이다. 보다 자세한 내용은 참고문헌 [9-10]을 참조바란다. 먼저 코어 매핑 문제를 ACO 기반으로 접근하기 전에 다음과 같이 몇 가지 전제를 하였다.

- NoC의 구조: Mesh
- 라우팅 알고리즘: XY Routing
- 서비스: Guaranteed Service
- 코어 크기: 모든 코어의 크기는 동일함
- 제한 조건: 링크 대역폭(Link Bandwidth)
- 설계 목적: 네트워크 지연 시간, 즉 평균 홉 수

이외에도 트래픽 패턴은 균일하다고 가정하였으며

데이타의 패킷화 및 충돌로 인한 지연 시간은 별도로 고려하지 않았다. 매핑 문제를 위하여 ACO 주요 파라미터들을 다음과 같이 정의하였다.

- 페로몬의 정의 ($\tau_i(k)$)

페로몬, $\tau_i(k)$는 코어 i가 노드 k에 위치하는 것에 대한 선호도이며 다음과 같이 계산된다.

$$\tau_i(k) = \Sigma_{g_j \in S} \tau(k_i, g_j), \quad 1 \le i,j \le V,\ 1 \le k,g \le N \tag{3}$$

식 (3)에서 S는 현재 ant에 의하여 이미 다른 노드에 매핑된 코어들의 집합이며 g_j는 코어 j가 g 노드에 위치하고 있다는 것을 나타낸다. 결국 상기 식에서 (k_i, g_j)가 의미하는 것은 노드 g에 코어 j가 위치하고 있을 때 노드 k에 코어 i가 매핑되는 것에 대한 선호도를 가리킨다.

- 선험 값의 선정 (η_i)

선험 값 η_i는 다양한 동작 조건 하에서 최적 값을 좀 더 수월하게 구하기 위해 사전 결정되거나 이미 알고 있는 주요 정보를 매핑 위치 선택에 참조할 수 있도록 삽입하는 과정이다. 본 연구에서 사용되는 η_i는 어플리케이션 코어 중 다른 코어와의 데이타 송수신 양이 많은 것을 골라 이웃 노드와의 연결 링크가 많은 노드에 우선적으로 배정하는 방식을 사용한다. 즉 그림 2의 예제 중 vop reconstruction처럼 트래픽을 많이 유발하는 블럭을 가능하면 mesh의 내부 노드에 할당되도록 가중치를 부여하는 것이다. 물론 실제 코어 매핑 시 페로몬 정보와 선험 값의 반영 비율은 임의로 조절이 가능하다.

- 페로몬과 선험 값을 이용한 확률적인 결정

Ant가 코어 i를 노드 k에 할당하는 확률 값, $p_i(k)$는 다음과 같이 계산된다.

$$p_i(k) = \frac{\tau_i(k) \cdot \beta}{\Sigma_{\epsilon=1}^{N} (\tau_i(\epsilon) \cdot \beta)}, \quad \begin{matrix} if\ \eta(i) = k, \beta \ge 1 \\ else \qquad \beta = 1 \end{matrix} \tag{4}$$

식 (4)를 간단히 정리하면 $p_i(k)$는 현재 비어있는 노드 중 하나를 선택할 때 이미 할당된 다른 코어와 그 위치에 대한 선호도의 총 합을 기준으로 가장 높은 선호도를 부여받은 노드에 대한 할당 가능성을 증가시키는 것이다. 식에서 β는 할당될 노드 중 선험 값으로 결정된 노드에 대해서는 다른 노드에 비해 상대적인 선호도를 증가시키기 위해 가중치를 부여하는 파라미터로 이 값은 프로그램 상에서 임의 변경 가능하다. 단 최초로 매핑되는 코어와 그 위치는 완전 임의 방식으로 선정한다.

- 페로몬 정보의 수정

모든 코어를 매핑한 이후 네트워크 지연 시간을 측정하여 그 결과에 따라 해당 매핑의 우수성을 평가하고 그 내용을 페로몬 정보 수정을 통하여 반영하는 과정이다.

$$\tau(t+1) = \rho \cdot \tau(t) + \Delta\tau(t) \tag{5}$$

상기 식 (5)에서 ρ는 페로몬의 평균 감쇠 파라미터이며 $\Delta\tau$는 페로몬 강화 파라미터이다. 알고리즘 성능에 큰 영향을 미치는 강화 파라미터는 현재 상수 값으로 고정시키는 방식을 사용한다.

- Exploration과 Exploitation의 균형을 위한 방법

해집합의 충분한 검색 및 결과 값의 신뢰성을 위하여 iteration-best solution(S^{ib})와 global-best solution(S^{gb}), 그리고 페로몬의 최소값(τ^{min}) 설정 방식을 적용한다. 또한 군집체(colony) 당 ant의 수는 하나로 고정한다.

3. ACO 기반 어플리케이션 코어 매핑 과정

전체적인 ACO 기반 어플리케이션 코어 매핑 과정은 그림 3에서 나타내었다. 매핑 과정에 대한 세부 설명에 앞서 사용된 ACO 파라미터 중 앞에서 언급하지 않은 것을 정의하면 다음과 같다.

- N_{ib}: Iteration-best ant의 수
- N_{colony}: 하나의 군집체 내 ant의 수
- γ: S^{gb}를 다시 사용하기 위해 대기하는 횟수

그림 3의 1번 라인은 모든 코어들에 대한 선험 값을 구하는 과정이다. 전체 코어 중 일부에만 선험 값을 할당하는 경우 추가적인 파라미터를 사용하여 트래픽 유발 상위 몇 % 블럭에만 할당할 것인가에 대한 결정이 필요하며 현재는 상위 15% 블럭으로 한정하였다. 2번 라인의 경우 지금까지 언급한 모든 ACO 파라미터의 초기 값을 결정하는 과정이다. 3번 라인은 페로몬 정보를 초기화시키는 과정으로 초기 값은 일반적으로 ρ의 크기에 따라 결정되며 현재는 $100/(100-\rho)$로 정의하였

```
1. Compute heuristic variable, η_i ;
2. Set ACO parameters;
3. Initialize pheromone trails;
4. While (result is unsatisfactory) {
5.   While (N_ib < γ) {
6.     While (i < N_colony) {
7.       Application Core Mapping;
         Check BW constraint;
         Evaluate network latency;
       }
8.   Find S^ib; pheromone_update(S^ib);
     }
9. Find S^gb; pheromone_update(S^gb);
   }
```

그림 3. ACO 기반 어플리케이션 코어 매핑 과정
Fig. 3. ACO-Based Application Core Mapping Flow.

다. 그리고 4번 라인부터 ACO를 이용한 반복적인 매핑 과정이 시작된다. 전체 매핑 과정은 크게 2개의 루프로 구성되는데 첫 번째 루프(5번 라인부터 시작)는 iteration-best solution을 구하는 과정이며 두 번째 루프(6번 라인에서 시작)는 군집체 단위의 코어 매핑 결과를 구하는 과정이다. 7번 라인에서 구한 매핑 결과는 대역폭 제한 조건을 검사하고 이를 만족할 경우에만 네트워크 지연 시간 계산을 통하여 최종 매핑 결과 값으로 확정하게 된다. 만약 대역폭 조건을 위반하는 경우 별도의 네트워크 지연 시간 계산 없이 사전에 결정한 최대 지연 시간으로 할당한다. 군집체 단위의 매핑 결과는 iteration-best solution 및 페로몬 정보 수정에 사용한다. 그리고 일정 횟수의 iteration-best solution이 반영되면 그 중 가장 우수한 매핑 결과를 global-best solution과 비교, 기존 solution보다 우수할 경우 이를 대체하며 global-best solution 결과는 γ 번마다 한 번씩 페로몬 수정에 사용한다.

IV. 실험 결과

제안한 알고리즘은 C 시뮬레이션을 통하여 검증하였으며 모든 실험은 Sun UltraSPARC III 1.2GHz 프로세서 시스템에서 진행되었다. 최종 결과 도출을 위해 사용한 ACO의 주요 파라미터 값은 다음과 같다.

- N_{colony}: 10
- γ : 10
- ρ : 0.95
- τ^{min}: 2.0, $\tau(0)$: 20.0 (페로몬의 초기 값)
- $\Delta\tau$: 0.2

상기 값들은 ACO 관련 이전 연구와 계산 시간 등을 고려하여 실험적으로 결정된 것이다. 그리고 실험에서 사용한 어플리케이션 코어 정보는 [8]에서 제시한 video object plane decoder를 사용하였다. 그러나 코어들의 트래픽 크기를 제외한 다른 세부 정보, 즉 크기나 전력 소모량에 관련된 정보가 없기 때문에 전력 소모량 최소화를 설계 목적으로 제시한 기존 결과와의 직접적인 비교는 불가능하다. 참고로 그림 2 (b)의 매핑 결과를 평균 홉 수만 계산하면 1.92이며 β의 값을 10, 그리고 B/W limit를 1000으로 고정하고 본 논문에서 제안한 알고리즘으로 매핑하였을 때 평균 홉 수는 1.75이며 매핑 결과를 그림 4에서 나타내었다. 그러나 설계 목적이 변화하거나 추가되더라도 그림 3의 코어 매핑 과정 대부분은 그대로 유지되며 매핑 후 제한조건 만족 여부를 확인하는 부분과 일부 ACO 파라미터들의 조절만이 요구된다.

또한 주어진 어플리케이션 코어 매핑 최적화를 위한 β값 및 합당한 대역폭을 구하기 위하여 β 및 대역폭의 범위를 가변시키며 평균 네트워크 지연 시간을 구하였다. β의 범위는 선험 값 η_i를 사용하지 않는 1부터 선험 값에 의해 코어 위치가 거의 고정되는 20까지 고려하였고 링크 대역폭의 제한 영역은 입력된 코어 정보를 기준으로 유효 매핑 결과를 얻을 수 있는 최소 대역폭부터 임의 매핑이 가능한 대역폭 영역까지 실험하였다. 실험 결과는 표 1에서 나타내었다.

표 1의 1-2번 열은 대역폭을 1000(MB/s)으로 고정하고 β를 변화시키면서 평균 네트워크 지연(홉 수)을 측정한 결과이다. 결과를 보면 β가 커지면서 트래픽이 많은 일부 코어들의 위치가 거의 일정하게 되고 이에 따라 최종 결과의 변화가 없어짐을 확인할 수 있다. 즉 β

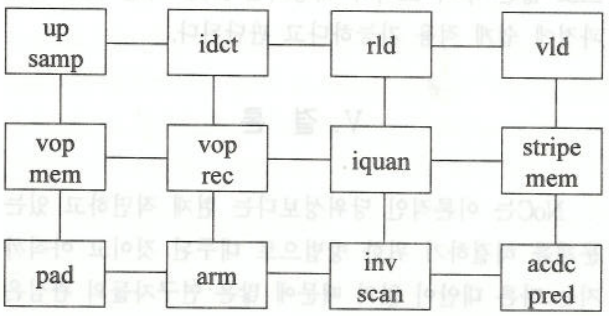

그림 4. 어플리케이션 코어 매핑 결과
Fig. 4. Application Core Mapping Result.

표 1. 실험 결과
Table 1. Experimental Results.

β (B/W: 1000)	Average Latency	B/W Limit (β : 10)	Average Latency
1	1.75	700	1.83
3	1.83	900	1.92
5	1.5	1100	1.75
7	1.83	1300	1.67
9	1.83	1500	1.67
11	1.67	1700	1.67
13	1.83	1900	1.75
15	1.75	2100	1.67
17	1.75	2300	1.5
19	1.75	2500	1.5

가 커지면 ant의 해집합 검색 범위가 줄어들어 알고리즘 계산 시간이 감소하지만 전체 최적 값에 이르지 못하고 국부 최적(local optimal) 값으로 수렴될 확률이 증가한다. 그러나 β가 작아지면 해집합 범위가 증가하여 최종 결과의 파동(fluctuation)이 발생할 확률이 증가하므로 이를 방지하기 위한 ACO 파라미터, 특히 강화 파라미터($\Delta\tau$),의 일부 변경이 필요할 수 있다. 표 1의 3-4번 열은 β를 10으로 고정하고 대역폭의 크기를 변화시키면서 평균 네트워크 지연을 측정한 결과이다. 대역폭이 증가하면서 네트워크 지연 정도가 감소하는 것을 확인할 수 있으며 또한 계산 시간도 크게 감소하였다. 지금까지의 결과를 분석하면 대부분의 경우 근사화된 최적 값으로의 수렴은 보장되나 최종 결과에 이르는 시간 및 이에 요구되는 NoC 특성, 예를 들어 동작속도나 라우팅 알고리즘,의 최적화도 동시에 이루기 위해서는 효과적인 ACO 파라미터의 결정이 선행되어야 함을 알 수 있다.

12개의 코어로 구성된 본 실험에서 고정된 수행 조건별로 제안된 알고리즘을 이용하여 최종 결과를 도출하는데 소요된 시간은 평균 1초 내외였다. 따라서 일반적으로 많은 수의 코어가 내장되는 NoC 기반 SoC 설계 과정에 쉽게 적용 가능하다고 판단된다.

V. 결 론

NoC는 이론적인 당위성보다는 현재 직면하고 있는 문제를 해결하기 위한 방법으로 대두된 것이고 아직까지는 다른 대안이 없기 때문에 많은 연구자들의 관심은 이제 NoC의 상용화와 시스템 구축 방법으로 수렴되고 있다. 본 연구에서 수행된 규칙적인 NoC 구조 하에서의 어플리케이션 코어 매핑 방법은 NoC 기초 연구의 일환일 뿐만 아니라 NoC 기반 SoC 설계 플랫폼 개발의 시작으로 향후 본 연구 성과는 실용적인 설계 플랫폼 구축과 이와 관련한 많은 아이디어를 도출하고 검증할 수 있는 초석이 될 수 있을 것이다.

참 고 문 헌

[1] K. Goossens, J. Dielissen, O. P. Gangwal, S. G. Pestana, A. Radulescu, and E. Rijpkema, "A Design Flow for Application-Specific Networks on Chip with Guaranteed Performance to Accelerate SOC Design and Verification," Proc. DATE, pp. 1182-1187, 2005.

[2] S. G. Pestana, E. Rijpkema, A. Radulescu, K. Goossens, and O. P. Gangwal, "Cost-Performance Trade-offs in Networks on Chip: A Simulation- Based Approach," Proc. DATE, pp. 764-769, Feb. 2004.

[3] L. Benini, "Application Specific NoC Design," Proc. DATE, pp. 250-256, Mar. 2006.

[4] R. Pop and S. Kumar, "Mapping Applications to NoC Platforms with Multithreaded Processor Resources," NORCHIP Conf., pp. 36-39, Nov. 2005.

[5] K. Srinivasan and K. S. Chatha, "A Methodology for Layout Aware Design and Optimization of Custom Network-on-Chip Architectures," Proc. ISQED, Mar. 2006.

[6] S. Murali and G. D. Micheli, "Bandwidth -Constrained Mapping of Cores onto NoC Architectures," Proc. DATE, 2004.

[7] S. Murali, M. Coenen, A. Radulescu, and K. Goossens, "A Methodology for Mapping Multiple Use-Cases onto Networks on Chips," Proc. DATE, 2006.

[8] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini, and G. D. Micheli, "NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-Chip," IEEE Trans. on Parallel and Distributed Systems, Vol.16, No. 2, pp. 113-129, 2005.

[9] M. Dorigo and T. Stuetzle, "Ant Colony Optimization," MIT Press, 2004.

[10] F. Glover and G. Kochenberger, "Handbook of Metaheuristics," Kluwer Academic Publishers, 2003.

저 자 소 개

안 진 호(정회원)
1995년 연세대학교 전기공학과 학사 졸업.
1997년 연세대학교 전기공학과 석사 졸업.
2002년 LG전자 DTV연구소 주임연구원.
2006년 연세대학교 전기전자공학과 박사 졸업.
2008년 현재 호서대학교 전자공학과 전임강사.
<주관심분야: SoC 설계 및 응용, 테스트>

김 현 진(학생회원)
1997년 연세대학교 전기공학과 학사 졸업.
1999년 연세대학교 전기 및 퓨 컴퓨터공학과 석사 졸업.
2005년 삼성전기 중앙연구소 선임연구원
2008년 현재 연세대학교 전기전자공학과 박사과정
<주관심분야: SoC 설계 및 응용, CAD>

김 홍 식(정회원)
1997년 연세대학교 전기공학과 학사 졸업.
1999년 연세대학교 전기 및 컴퓨터공학과 석사 졸업.
2004년 연세대학교 전기전자공학과 박사 졸업.
2005년 Virginia 공대 박사후 연구원.
2006년 삼성전자 시스템 LSI 사업부 책임연구원
2008년 현재 연세대학교 TMS 사업단 연구교수.
<주관심분야: SoC 설계 및 테스트>

박 영 호(정회원)
1985년 대전산업대학교 전자계산학과 졸업.
2008년 현재 한국전자통신연구원 NoC 기술팀 책임연구원.
<주관심분야: SoC 설계, 컴퓨터네트워크>

강 성 호(평생회원)
1986년 서울대학교 제어계측공학과 학사 졸업.
1988년 The University of Texas, Austin 전기 및 컴퓨터 공학과 석사 졸업.
1992년 The University of Texas, Austin 전기 및 컴퓨터 공학과 박사 졸업.
1992년 미국 Schlumberger 연구원.
1994년 Motorola 선임 연구원.
2008년 현재 연세대학교 전기전자공학과 교수.
<주관심분야: SoC 설계 및 SoC 테스트>