저전력 네트워크-온-칩을 위한 통신 최적화 기법

(Communication Optimization for Energy-Efficient Networks-on-Chips)

신 동 군 † 김 지 홍 ††
(Dongkun Shin) (Jihong Kim)

요 약 네트워크-온-칩은 미래 시스템-온-칩 제품을 위한 실용적인 개발 플랫폼으로서 부각되고 있다. 우리는 전압 조절이 가능한 회선을 가진 네트워크-온-칩에서 태스크간 통신으로 인한 전력 소모를 최소화하기 위한 정적 알고리즘을 제시한다. 최적의 통신 속도를 찾기 위해 제시된 (유전자 알고리즘에 기반한) 기법은 네트워크 망 구조, 태스크 할당, 타일 매핑, 라우팅 경로 할당, 태스크 스케줄링과 회선 속도 할당을 포함한다. 제시된 설계 기법은 기존의 기법과 비교하여 평균 28%까지 전력 소비를 감소시킬 수 있다는 것을 실험 결과는 보여 준다.

키워드 : 네트워크-온-칩, 저전력, 설계자동화

Abstract Networks-on-Chip (NoC) is emerging as a practical development platform for future systems-on-chip products. We propose an energy-efficient static algorithm which optimizes the energy consumption of task communications in NoCs with voltage scalable links. In order to find optimal link speeds, the proposed algorithm (based on a genetic formulation) globally explores the design space of NoC-based systems, including network topology, task assignment, tile mapping, routing path allocation, task scheduling and link speed assignment. Experimental results show that the proposed design technique can reduce energy consumption by 28% on average compared with existing techniques.

Key words : network-on-chip, low-power, design automation

1. 서 론

네트워크-온-칩(NoC)은 시스템-온-칩(SoC) 제품의 실용적인 개발 플랫폼으로서 최근에 제시되고 있다 [1,2]. 네트워크-온-칩은 구조화되고 모듈화된 네트워크 인터페이스를 제공하여 복잡한 온-칩(on-chip) 네트워크 문제를 극복하는 데 특히 효과적이다. 네트워크가 구조화되고 사전에 배선되기 때문에 그들의 전자적 파라메터를 잘 제어하여 최적화할 수 있고 그 결과 전력 소모와 전송 시간을 상당히 줄일 수 있는 적극적 기법을 사용할 수 있다. 또한, 모듈간의 표준화된 인터페이스는 재사용성과 호환성을 증대시킨다.

그림 1(a)에서 보듯이, NoC 기반의 시스템은 일반적으로 정형의 타일(tile)들로 구성되며 각각의 타일은 프로그램 가능한 마이크로 프로세서나 ASIC, 또는 FPGA 등이 될 수 있다. 전용 배선을 사용하여 연결하는 대신에, 각 타일들은 타일간의 패킷을 전송하는 상호 연결망(interconnection)에 연결되어 있다. 그림 1(b)에 보듯이, NoC 내의 라우터(router)는 입력/출력 회선(link)과 버퍼, 그리고 크로스바(crossbar) 스위치로 구성되어 있다.

NoC 기반의 시스템에서 온-칩 네트워크는 시스템 전

· 이 연구를 위해 연구 장비를 지원하고 공간을 제공한 서울대학교 컴퓨터 연구소에 감사드립니다. 이 논문은 2007 년도 두뇌한국21 사업과 2007 년도 정부(과학기술부)의 재원으로 한국과학재단의지원을 받아 수행된 연구입니다(No. R0A-2007-000-20116-0).

† 정 회 원 : 성균관대학교 정보통신공학부 교수
dongkun@skku.edu
†† 종신회원 : 서울대학교 컴퓨터공학부 교수
jihong@davinci.snu.ac.kr

논문접수 : 2005년 9월 27일
심사완료 : 2007년 12월 22일

력 소모의 상당한 부분을 차지한다. 예를 들면, 16개의 처리 소자(processing element) 타일로 구성된 MIT Raw 온-칩 네트워크는 전체 칩 전력의 36%를 소비하며[3], Alpha 21364 프로세서의 경우에는 전체 칩 전력의 20%가 라우터와 회선에서 소모된다[4].

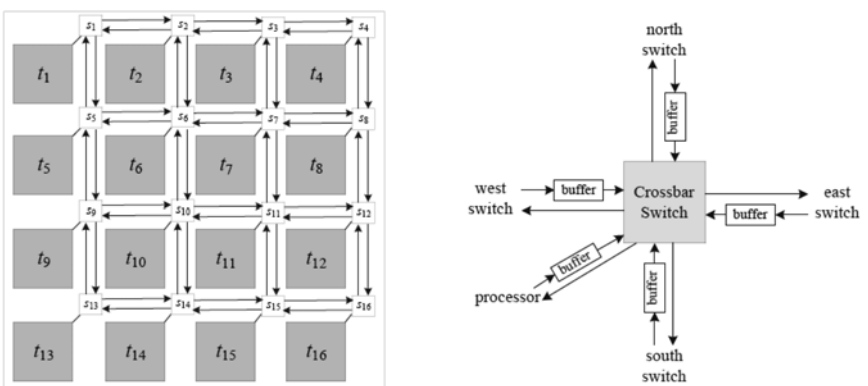
NoC를 위한 저전력 기법 중에 기대할 만한 것으로 통신 회선의 속도와 동작 전압을 조절하는 것이 있다[5]. 두 종류의 속도 조절 기법이 있는데, 하나는 온라인 기법으로 실행시간의 통신량 변동을 기준으로 통신 속도를 동적으로 조절하는 것이며 다른 하나는 오프라인 기법으로 목표 응용프로그램의 통신 경향을 바탕으로 각 회선에 적절한 고정 통신 속도를 할당하는 것이다. 오프라인 기법은 설계 시간에 시스템 설계자가 통신 지연시간을 예측할 수 있다는 점에서 실시간 응용에 더 적합하다. 게다가 온라인 기법이 실행시간에 통신량을 관찰해야 하는 반면에 오프라인 기법은 실행시간의 오버헤드를 요구하지 않는다.

본 논문은 전압 조절이 가능한 회선을 가진 저전력 NoC를 위해 오프라인의 회선 속도 할당 알고리즘을 제안한다. 제안되는 알고리즘은 주기적 실시간 응용의 태스크 그래프를 가지고 실시간 응용의 시간 제약조건을 보장하면서도 NoC의 전력 소모를 최소화하는 각 회선의 적당한 통신 속도를 결정한다. 또한, 상호연결 네트워크의 누설 전력(leakage power)을 무시할 수 없기 때문에 (예를 들어, 0.07μm 공정에서 누설전력과 스위칭 전력을 합한 전체 전력 소모의 21%를 차지한다[6]), 제안된 알고리즘은 통신이 할당 되지 않은 회선은 정적으로 전원을 차단한다.

일반적인 멀티프로세서 시스템에서 설계 흐름은 2가지 핵심 과정으로 구성되는 데, 태스크 할당과 태스크 스케줄링이다. 설계 제약조건(예, 실행시간과 전력소모)과 처리 소자(processing element)들이 주어졌을 때, 각 태스크는 우선 적절한 처리 소자에 할당되고(태스크 할당), 다음에 처리 소자 내에서 태스크들의 실행 순서가 결정된다(태스크 스케줄링). 그러나, NoC 시스템에서는 2개의 과정이 추가로 필요한 데, 타일 매핑(mapping)과 라우팅(routing) 경로 할당이다. 타일 매핑은 각 처리 소자를 NoC의 하나의 타일위치에 배치하는 것이며 라우팅 경로 할당은 타일간의 통신 경로를 결정하는 과정이다. 예를 들어, 그림 1(a)와 같이 NoC가 16개의 처리 소자를 가졌다면 각 처리 소자가 어떤 타일에 배치될지 결정해야 하며 데이타가 타일 t_1에서 타일 t_{16}으로 전송되어야 할 때 s_1부터 s_{16}의 스위치 중에 어떤 것이 데이타를 전달할 지를 결정해야 한다. (본 논문에서는 타일 매핑과 라우팅 경로 할당의 2가지 과정을 통틀어 네트워크 할당이라는 용어를 사용한다.)

NoC 시스템에서는 태스크 할당과 태스크 스케줄링 과정뿐만 아니라 네트워크 할당 과정에서 정해진 설계 사항이 각 회선의 통신 속도에 상당한 영향을 미칠 수 있는데, 그것은 설계 과정의 결과에 따라 통신량의 패턴이 변하기 때문이다. 그러므로, 저전력의 NoC 시스템을 설계하기 위해서는 모든 설계 과정에서 회선의 전력 소모에 대한 영향을 고려한 최적화가 이루어져야 한다.

예를 들어, 태스크 τ_1, τ_2, τ_3, τ_4가 p_1, p_2, p_3, p_4에 각각 할당된 그림 2(a)의 태스크 그래프 g를 생각해 보자. 타일 매핑 알고리즘은 그림 2(b)에 보이는 것과 같이 p_1, p_2, p_3, p_4 를 각각 $tile_3$, $tile_4$, $tile_2$, $tile_1$에 배치한 네트워크 할당을 만들어 낼 수 있다. 그림에서 회선상의 숫자는 해당 회선의 통신량을 나타내고 있다. 라우팅 경로가 XY-라우팅 알고리즘[7]에 의해 할당된다고 가정하면 패킷은 먼저 X축을 따라 라우팅되며 일단 패킷이 목표 타일과 같은 Y축 행에 도달하면 Y축을 따라서 라우팅된다. 그래서, 네트워크 할당 NA_1의 통신 비용은 90이 된다. 그러나 타일 매핑이 네트워크 할당 NA_2 (그



(a) An NoC with 16 tiles (b) Stricture of a router

그림 1 NoC 기반 시스템의 구조

림 2(c))로 바뀌면 더 작은 70의 통신 비용이 든다. 그러므로, 네트워크 할당 NA_1보다 네트워크 할당 NA_2가 전력소모가 적다고 할 수 있다.

이제 NA_2에서 각 회선의 통신 속도를 어떻게 할당할지 생각해 보자. 만약에 태스크 τ_3가 경성 마감시간을 가졌다면, $tile_1$으로부터 $tile_2$로의 회선이 임계 경로(critical path)에 있으며 τ_2와 τ_3사이뿐만 아니라 τ_2와 τ_4사이의 데이타도 전송해야 하기 때문에 이 회선은 빠른 속도를 가져야 한다. 즉 임계 경로가 비임계 경로와 회선을 공유하지 않도록 네트워크 할당 NA_2를 개선시킬 필요가 있다는 것을 알 수 있다. 그러므로, 라우팅 경로 할당은 각 회선의 속도 결정에 대한 영향을 고려하여 진행되어야 한다.

만약 NA_2에서 에지 (τ_2, τ_3)의 라우팅 경로를 $tile_1 \rightarrow tile_2 \rightarrow tile_4$로부터 $tile_1 \rightarrow tile_3 \rightarrow tile_4$로 바꾸면, 그림 2(d)의 네트워크 할당 NA_3이 된다. NA_3는 NA_2와 같은 양의 전체 통신량을 가졌지만 $tile_1$에서 $tile_2$까지의 회선에 더 낮은 속도를 할당할 수 있다.

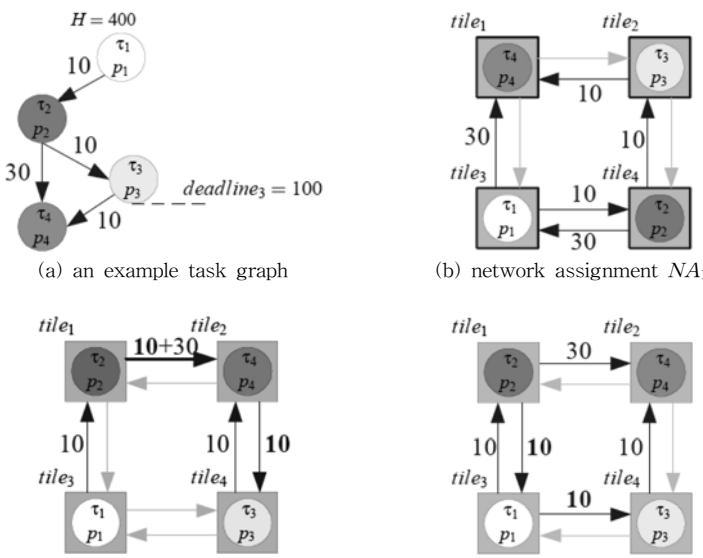
본 논문에서는 기존의 NoC 설계 기법이 전압 및 속도 조절이 가능한 회선을 가진 NoC에는 효과적이지 못함을 보이고 전체 설계 공간을 효과적으로 탐색하는 (유전자 알고리즘에 기반한) 새로운 최적화 기법을 제시한다. 실험 결과, 제안된 설계 알고리즘은 기존의 알고리즘에 비해 평균 28%의 전력 소모 감소를 가져왔다.

본 논문의 구성은 다음과 같다. 2장에서는 NoC 설계 기법에 대한 관련 연구를 간략히 정리하며 3장에서는 전체 설계 흐름과 문제 정의를 다룬다. 4장에서 상세 설계 기법들이 서술되며 5장에서는 실험 결과를 제시한다. 6장에서는 요약과 향후 연구방향을 제시하며 논문을 마무리한다.

2. 관련 연구

여러 연구 그룹들이 NoC 기반 시스템에서의 전력 소모를 줄이기 위한 설계 기법을 연구해 왔다. 예를 들면, Simunic과 Boyd[8]는 NoC 시스템을 위한 전력 관리 기법을 제안했다. 이 기법은 네트워크 기반의 전력 관리를 통해 노드 기반의 전력 관리 기법보다도 미래의 작업량을 더욱 효과적으로 예측하고 있다. 이 기법이 처리 소자에 중심을 둔 것에 비해 다른 기법들[4,5,9-13]은 통신 회선이 많은 전력을 소모하므로 통신 회선에서의 전력 소모를 줄이려는 시도를 했다. Kim과 Horowitz[5]는 가변 주파수 회선을 제안했는데, 이것은 회선의 동작 주파수가 변할 때 회선이 사용하는 전압을 최소값으로 조정하여 전력 소모를 줄인다. 가변 주파수 회선은 3.5Gb/s에서는 197mW를 소모하지만 1Gb/s에서는 21mW를 소모하여 10배의 전력 감소를 가져올 수 있다.

가변 주파수 회선의 특징을 이용하여 Sang[4]은 회선과 입력 버퍼의 사용효율에 따라 회선의 클럭 주파수와 동작 전압을 조정하는 히스토리 기반의 동적 전압 조절 기법을 개발했다. Soterious[9]는 회선의 누설 전력 소모를 줄이기 위해 통신량의 변화를 바탕으로 일부 회선의 전력을 차단하는 동적 전력 관리 기법을 제시했다. Worm[10]은 온-칩 네트워크를 위한 적응적 저전력 전송 기법을 제안했다. 이 기법은 회선의 전압을 동적으로



(a) an example task graph (b) network assignment NA_1

(c) network assignment NA_2 (d) network assignment NA_3

그림 2 동기 예제

조절함으로써 서비스 품질(QoS)에 대한 요구 조건을 만족시키면서도 신뢰 가능한 통신을 제공하도록 전력을 최소화한다.

이상의 기법들이 온라인 기법인 데 반해, 본 논문에서는 각 회선에 적절한 고정 속도를 할당하는 오프라인 기법을 제시한다. 본 기법은 응용 프로그램의 태스크 그래프로부터 통신 패턴 정보를 활용한다. 비슷한 문제를 다룬 몇몇 연구가 있었는데, Hu[11]는 통신 트래픽을 감소시켜 전력 소모를 줄이는 네트워크 할당 알고리즘을 제안했다. Marcon[12]은 통신간의 의존성을 고려하여 Hu의 알고리즘을 개선하였다. Ogras[13]는 통신 패턴을 바탕으로 NoC의 망 구조에 대한 최적화 기법을 제안했다.

하지만 이들은 회선의 속도를 조절하는 문제는 고려하지 않고 태스크 할당과 스케줄링이 완료되었다는 가정하에 단지 네트워크 할당만을 고려했다. Lei[14]도 타일 매핑에 있어서 태스크 그래프의 통신 패턴을 사용하였지만, 이 기법의 목적은 전체 실행 시간을 최소화하는 타일 매핑을 찾는 것이 목적이었다.

3. NoC 시스템의 전체 설계 과정

3.1 명세와 아키텍쳐 모델

본 논문에서는 주기적 실시간 응용 프로그램을 태스크 그래프 G=<V, E>로 나타내는 데, G는 방향성 비순환 그래프이며 V는 태스크의 집합, E는 태스크간의 방향성 에지의 집합이다. G에서 각각의 방향성 에지 $e(\tau_i, \tau_j)$는 τ_i와 τ_j 사이의 선행관계를 나타내는 데, 태스크 τ_j가 시작되기 전에 반드시 태스크 τ_i가 완료되어야 함을 의미한다. (간편한 서술을 위해 $e(\tau_i, \tau_j)$는 $e_{i,j}$로 표기한다.) 태스크 그래프 G의 주기는 period(G)로 표기한다. G에 있는 태스크 τ_i는 응용 프로그램이 정확히 동작하기 위해서 지켜져야 할 마감시간 d_i를 가질 수 있다. 각 에지 $e_{i,j}$는 τ_i와 τ_j가 서로 다른 처리 소자에 할당되었을 때, 두 태스크 사이에 전송되어야 할 통신 데이타량을 의미하는 $w(e_{i,j})$를 가지고 있다. 그림 2(a)는 태스크 그래프의 예를 보여 주고 있다.

본 논문에서는 NoC의 망 형태로 그림 1(a)와 같은 정규적 타일 기반의 그물(mesh) 형 NoC 구조를 가정한다. $m \times m$ 개의 타일을 가진 NoC 기반의 시스템은 <T, L>로 나타낼 수 있는 데, T={t_1, ..., t_m, ..., t_{m2}}는 타일들의 집합이며 L={ℓ_1, ..., $\ell_{4m(m-1)}$}은 타일간의 회선의 집합이다. 모든 타일들은 같은 공간인 A를 차지한다고 가정한다. 타일 t_i와 t_j간의 회선은 $\ell_{i\to j}$로 나타내며 $\ell_{i\to j}$의 송신 타일과 수신 타일을 각각 src($\ell_{i\to j}$)와 dst($\ell_{i\to j}$)로 표시한다. W(ℓ_i)는 회선 ℓ_i를 거쳐가는 모든 데이타의 전체량을 의미한다. 처리 소자들의 집합은 R={r_1,...,r_n}로 표시되며 r_i는 i번째 처리 소자를 의미한다. 처리 소자의 개수와 타일의 개수는 동일하다고 가정한다. 즉, |R|=|T|. 각 타일은 행과 열 좌표를 의미하는 t_i.x와 t_i.y를 가지고 있다. (t_i.x와 t_i.y는 각각 (i-1)/m의 몫 및 나머지와 동일하다.)

3.2 문제 정의

주어진 태스크 그래프 G=<V, E>에 대해서, NoC 시스템 설계의 첫 번째 단계는 태스크 할당 과정으로 함수 Φ: V→R로 표시한다. 서로 다른 처리 소자에 할당된 태스크들 사이만 통신 부하를 발생시키므로 태스크 할당은 전체 통신 부하에 영향을 미친다. 다음 과정인 타일 매핑은 함수 Ψ: R→T로 나타낸다. 타일 매핑도 또한 전체 통신 부하에 영향을 미치는 데 그 이유는 타일간의 거리가 달라지기 때문이다. 그 다음은 타일간의 라우팅 경로의 할당으로 함수 Ω: E→P를 사용하는 데, P는 회선순서들의 집합이다. 라우팅 경로 할당 후, 각 회선 ℓ_i에 대해서 W(ℓ_i)가 $\sum_{\forall e_j, \ell_i \in \Omega(e_j)} w(e_j)$ 로 결정된다.

라우팅 경로 할당 과정은 최단 경로 라우팅을 쓰기 때문에 전체 통신 부하에 영향을 미치지 않는다. 하지만 각 회선의 W(ℓ_i)에는 영향을 미치기 때문에 회선의 속도에 영향을 준다. 태스크 스케줄링 과정은 동일한 처리 소자에 할당된 태스크들의 실행 순서를 결정하며 태스크 그래프 g에 추가의 에지를 삽입하여 g′으로 변형시키기 때문에, 이 과정을 O: G→G로 표현할 수 있다. 타일 매핑과 라우팅 경로 할당 과정은 태스크 스케줄링 전에 먼저 실시되는데, 그 이유는 태스크 스케줄링을 위해서는 노드 간의 통신 시간을 알아야 하기 때문이다. 회선 속도 할당 과정은 여유 시간을 활용하여 전력 소모를 줄일 수 있도록 각 회선의 클럭 속도를 결정한다. 회선 속도 할당 과정을 나타내기 위해서 S: L→C 함수를 사용하는 데, 여기서 C는 회선의 사용한 가능한 클럭 속도의 집합이다. 결론적으로 저전력 NoC를 위한 회선 전력 최적화 문제는 다음과 같이 정의할 수 있다.

Link Energy Optimization Problem

Given $G=<V,E>, R$ and $N=<T,L>$,

find the functions Φ, Ψ, Ω, O and S such that

$E = \sum_{\ell_i \in L} (C_L \cdot W(\ell_i) \cdot f(\ell_i)^2 + H \cdot P_{leakage}(\ell_i))$ is minimized

subject to $\forall \tau_i \in V, \ \theta(\tau_i) \le d_i$.

여기서 C_L는 회선의 평균 스위칭 커퍼시턴스(capacitance)를 의미하며, $f(\ell_i)$와 $P_{\text{leakage}}(\ell_i)$는 회선 ℓ_i의 클럭 속도와 누설 전력을 나타낸다. 통신이 할당되지 않은 회선은 전원을 차단한다고 가정하기 때문에 W(ℓ_i)=0인 ℓ_i에 대해서는 $P_{\text{leakage}}(\ell_i)$도 0이 된다. Θ($\tau_i$)는 τ_i의 종

료시간이다. 회선의 전력소모는 온-칩 네트워크의 전력 모델[15]이나 온-칩 네트워크 시뮬레이터[16]를 통해서 좀 더 정확하게 측정할 수 있다.

본 논문에서는 네트워크 할당 문제에 초점을 맞추기 위해 동일한 처리 소자들로 시스템이 구성되어 있다고 가정하며 통신에 의한 전력 소모만을 고려한다.

4. 저전력 NoC 설계

그림 3은 5단계의 최적화 과정으로 구성된 NoC 기반 시스템의 설계 과정을 보여 주고 있다. 주어진 태스크 그래프와 NoC 구조에 대해서, 이 설계 과정은 최적화된 NoC 시스템을 생성해 낸다. NoC의 망 구조가 주어지지 않은 경우에는 망 구조 선택(topology selection) 과정이 먼저 진행되어야 한다. 본 논문은 설계 공간을 효과적으로 탐색하기 위해서 3개의 유전자 알고리즘(GA)인 GA 기반 태스크 할당 알고리즘 (GA-TA), GA 기반 타일 매핑 알고리즘 (GA-TM), 그리고 GA 기반 라우팅 경로 할당 알고리즘(GA-RPA)을 사용한다. 3개의 알고리즘은 중첩 반복 구조로 되어 있다. 예를 들어, 각 GA-TM 솔루션에 대해서 적합도(fitness) 평가를 위한 라우팅 경로 할당, 태스크 스케줄링 그리고 회선 속도 할당 과정이 진행된다. 네트워크 망 구조가 주어지지 않은 경우에는 GA 기반 망 구조 선택 알고리즘(GA-TS)이 사용된다.

최적화를 위해 유전자 알고리즘이 채택된 이유는 다음의 2가지 이유 때문이다. 본 논문의 통신 최적화 문제는 NoC 시스템의 타일 개수에 따라 차례곱(factorial)으로 늘어나는 매우 큰 문제 탐색 공간을 가지고 있다. 단지 16개의 타일에 대해서도 16! (≈20조)개의 타일 매핑이 가능하다. 더구나 모든 설계 과정들이 회선 속도 할당 과정과 밀접하게 연관되어 있기 때문에 각 과정을 독립적으로 최적화하여 좋은 속도 할당 결과를 찾을 수는 없다. 즉, 설계 문제의 전체 복잡도는 각 단계별 복잡도의 합이 아닌 곱이 되는 것이다.

그림 4는 전형적인 GA의 구조를 보여주고 있다. GA는 초기 개체군(population)에서 시작하여 교차(cross-over)와 변이(mutation) 연산을 사용하여 개체군을 진화시켜간다. GA의 성능에 영향을 미치는 다양한 설계 요소들이 그림 4의 우측에 나타나 있다. 이러한 요소들

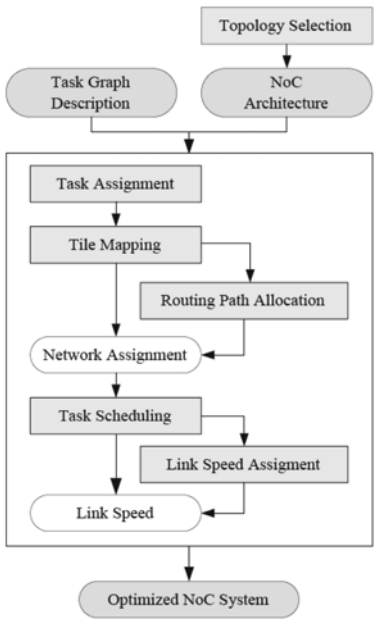


그림 3 저전력 NoC 시스템을 위한 설계 흐름

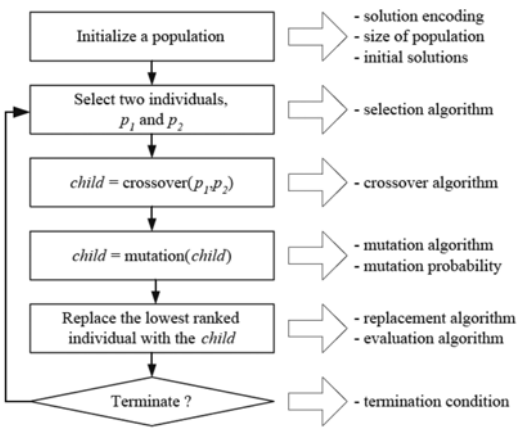


그림 4 유전자 알고리즘의 전형적인 구조와 설계 고려사항

은 주의 깊게 선택되어야 한다.

표 1은 회선 전력 최적화 문제를 위해서 본 논문에서 사용된 GA 연산을 종합적으로 보여주고 있다.

표 1 회선 전력 최적화를 위한 GA 연산

설계 단계	유전자 인코딩	교차연산	변이연산
GA-TA	1 차원 정수 배열	2-point crossover	무작위 선택된 유전자의 값 교체
GA-TM	1 차원 정수 배열	cycle crossover [17]	무작위 선택된 2개의 유전자의 값 교환
GA-RPA	1 차원 정수 배열	coordinate crossover	임의의 2개 타일간의 라우팅 경로 교체
GA-TS	2 차원 이진수 배열	2-D geographic crossover [18]	무작위 선택된 유전자의 값 교체

4.1 GA 기반의 태스크 할당

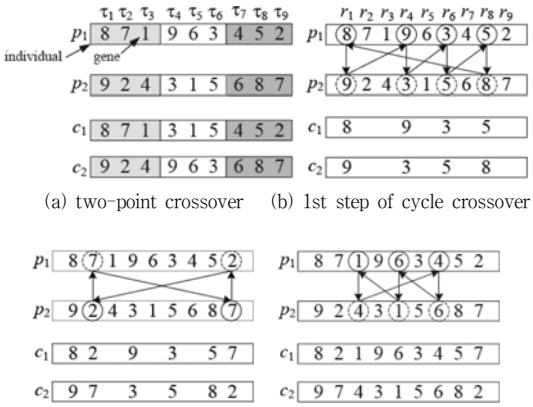
하나의 처리 소자에 많은 태스크가 할당될수록 필요한 메모리나 게이트가 많아짐으로 해당 처리 소자가 차지하는 면적도 커지게 된다. 각 타일은 같은 크기를 가지므로 처리 소자의 면적에 대한 제약은 $A_\phi(r_i) \le A$로 표현할 수 있는데, $A_\phi(r_i)$는 태스크 할당 함수 Φ에 대해서 처리 소자 r_i의 면적을 의미한다. 만약에 동일한 처리 소자에 할당된 2개의 태스크 사이에 에지 e가 있다면, w(e)는 0으로 설정된다. 이렇게 태스크 할당 과정은 전체 통신량에 영향을 미친다. 하나의 태스크 할당 솔루션은 1차원 정수 배열로 표현된다. 예를 들어, 그림 5(a)에서 개체 p_1의 태스크 τ_1은 r_8에 매핑되어 있다. 여기서는 GA에서 널리 쓰이는 2-지점 교차연산 (2-point crossover)를 사용하고 있다. 부모 개체인 p_1과 p_2는 같은 두 개의 지점에서 분리되며 자식 개체인 c_1은 p_1의 첫 번째 부분과 p_2의 두 번째 부분, 그리고 p_1의 세 번째 부분으로 만들어진다. 변이 연산은 어느 개체의 임의로 선택된 유전자들을 다른 값으로 변경하여 새로운 개체를 생성해 낸다.

4.2 GA 기반의 타일 매핑

타일 매핑의 솔루션도 1차원 정수 배열로 표현된다. 예를 들어, 그림 5(b)의 개체 p_1에서 처리 소자 r_1은 타일 t_8에 매핑되어 있다. GA 기반의 타일 매핑에 있어서는 주의 깊게 교차 연산을 설계해야 한다. 타일 매핑을 위해서 2-지점 교차연산을 사용한다면 하나의 타일에 다른 2개의 처리 소자들이 할당되는 비정상의 솔루션을 얻을 수도 있다. 그래서, 본 논문에서는 개체의 인코딩 이 순서를 의미할 때 적합한 회전식 교차연산(cycle crossover)[17]을 사용한다. 그림 5(b)-(d)는 회전식 교차연산을 사용하여 어떻게 자식 개체를 만들어 내는 지 보여주고 있다. 변이 연산은 임의로 선택된 2개의 유전자를 서로 교환하여 새로운 자식 개체를 만든다.

각 개체에 대해서 라우팅 경로 할당, 태스크 스케줄링, 회선 속도 할당 과정이 진행된다. 타일 매핑에서는 연산 시간을 줄이기 위해서 망 구조상으로 동일한 개체들을 확인하여 이들에 대해서는 하나의 개체에 대해서만 평가 과정을 거치게 된다. 동일한 형태의 개체인지를 확인하기 위해서는 각 개체는 '정렬된 형태'로 변형하여 비교한다. 정렬된 형태란 다음과 같은 속성을 가지고 있다: (1) 처리 소자 $\Psi^{-1}(t_1)$은 $\Psi^{-1}(t_m)$, $\Psi^{-1}(t_{m^2-m+1})$, $\Psi^{-1}(t_{m^2})$보다 작은 인덱스를 가진다. 여기서 t_1, t_m, t_{m^2-m+1}, t_{m^2}는 코너에 있는 4개의 타일을 의미한다; (2) 처리 소자 $\Psi^{-1}(t_m)$는 $\Psi^{-1}(t_{m^2-m+1})$보다 작은 인덱스를 가진다. 임의의 타일 매핑은 회전이나 미러링(mirroring)을 통해서 정렬된 형태로 바꿀 수 있다.

타일 매핑 후에는 통신 부하량의 집합인 CL을 구성



(a) two-point crossover (b) 1st step of cycle crossover

(c) 2nd step of cycle crossover (d) 3rd step of cycle crossover

그림 5 GA-TA와 GA-TM의 교차 연산

한다. $w(e_{i,j})>0$인 에지 $e_{i,j}$에 대해서 다음과 같은 3가지의 속성을 가진 통신 부하량 υ를 생성한다: $\upsilon_{src} = \Psi(\Phi(\tau_i))$, $\upsilon_{dst} = \Psi(\Phi(\tau_j))$, $\upsilon_{data} = w(e_{i,j})$.

4.3 GA 기반의 라우팅 경로 할당

타일들간의 정보 전달을 위해서 NoC 기반 시스템은 네트워크를 통해서 데이타 패킷을 라우팅하는 기법이 필요하다. 본 논문에서는 최소경로 정적 라우팅만을 고려하는데, 그 이유는 온-칩 네트워크에 더욱 적합하며 통신량도 적게 발생시키기 때문이다.

라우팅 경로 할당에서 거쳐갈 타일 번호들의 열로 표현되는 라우팅 정보를 출발 지점의 처리 소자가 데이타 패킷과 함께 보낸다고 가정한다. 출발 지점과 종착 지점 사이의 중간 라우터들은 라우팅 정보의 첫 번째 값을 가지고 전송 방향을 결정하며 나머지 라우팅 정보를 데이타와 함께 전달한다. 그림 1(a)와 같은 정규 2-차원 그물(mesh) 구조에서는 라우팅 경로 할당 과정은 두 개 타일간의 맨하탄(manhattan) 거리가 n일 때 n개의 방향을 결정하는 것이다.

GA 기반 라우팅 경로 할당에서는 개체를 1차원 정수 배열로 표현한다. 각 배열은 통신 부하량 $\upsilon \in CL$의 라우팅 경로를 나타낸다. 예를 들어, 그림 6은 통신 부하량 υ ($\upsilon_{src} = t_1$, $\upsilon_{dst} = t_{16}$)에 대한 2 가지 서로 다른 라우팅 경로를 보여주고 있다. 라우팅 경로 p_1과 p_2는 다음과 같이 표현될 수 있다:

$$p_1 = \{\ell_{1\to2}, \ell_{2\to3}, \ell_{3\to7}, \ell_{7\to11}, \ell_{11\to15}, \ell_{15\to16}\}$$
$$p_2 = \{\ell_{1\to5}, \ell_{5\to9}, \ell_{9\to10}, \ell_{10\to11}, \ell_{11\to12}, \ell_{12\to16}\}.$$

라우팅 경로의 솔루션을 생성하는 데는 주의를 기울여야 하는데, 그 이유는 라우팅 경로 솔루션에서 인접한 타일간의 통신 회선이 주어진 네트워크 망 구조에서 제공되지 않는다면 이 솔루션은 사용할 수 없기 때문이다. 그래서 라우팅 경로 할당은 특별한 교차 연산이 필요하

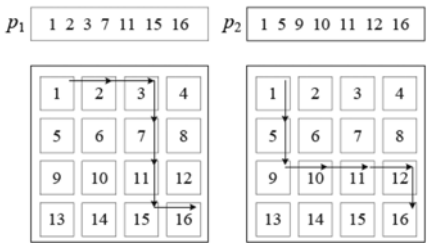
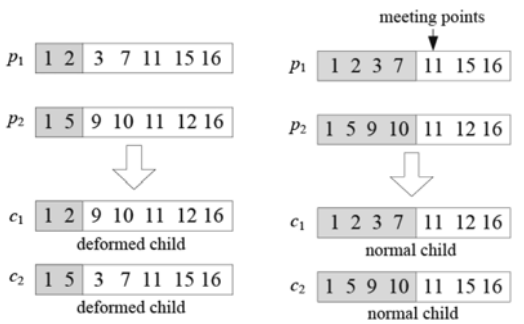


그림 6 라우팅 경로 인코딩



(a) one-point crossover (b) coordinate crossover

그림 7 라우팅을 위한 교차 연산

다. 그림 7(a)에서 볼 수 있듯이, 경로 라우팅을 위해서 1-지점 교차연산을 사용한다면 통신 회선 $\ell_{2\to9}$와 $\ell_{5\to3}$가 네트워크 망 구조에서 제공되지 않기 때문에 사용 불가능한 솔루션을 얻게 된다. 교차 연산은 사용 가능한 솔루션만을 생성해 내며 부모 개체의 특성을 자식 개체가 물려받을 수 있도록 보장해야 한다. 이러한 요구조건을 만족시키기 위해서 본 논문에서는 경로 라우팅을 위해 특별히 고안된 '좌표 교차연산(coordinate crossover)'을 제안한다. 그림 7(b)는 좌표 교차연산을 보여주고 있다. 먼저, 2개의 부모 개체로부터 미팅 지점(meeting point)들을 찾는다. 미팅 지점은 2개의 부모 개체로 표현된 서로 다른 라우팅 경로가 만나게 되는 지점이다. 예를 들어, 그림 6에서 p_1과 p_2로 표현된 2개의 라우팅 경로는 타일 t_{11}에서 만난다. (출발 지점과 종착 지점인 t_1과 t_{16}은 제외한다.) 두 번째로 2개의 부모 개체는 미팅 지점 바로 이전에서 분리되어 다중 지점 교차 연산처럼 부모들의 분리된 부분을 혼합하여 자식 개체를 만들게 된다.

만약에 2개의 미팅 지점 mp_α와 mp_β가 있다면, 부모 개체들은 미팅 지점에 의해 p_1={sp_i, sp_j, sp_k}와 p_2={sp_a, sp_b, sp_c}로 분리된다. 여기서 부분 경로 sp_j와 sp_b의 첫 번째 타일은 mp_α이며 부분 경로 sp_k와 sp_c의 첫 번째 타일은 mp_β이다. 이로부터 자식 개체 c_1={sp_i, sp_b, sp_k}와 c_2={sp_a, sp_j, sp_c}가 생성된다. 만약에 2개의 부모 솔루션이 정상이라면 sp_i의 마지막 타일과 sp_b의 첫 번째 타일 사이 및 sp_b의 마지막 타일과 sp_k의 첫 번째 타일 사이에 통신 회선이 있는 것이 보장되므로 자식 솔루션도 정상이다. 그러므로, 좌표 교차연산은 사용 가능한 자식 개체만을 생성한다고 말할 수 있다. 또한 자식 개체들은 부모 개체들이 공유하는 회선들을 물려받는 특징을 가진다.

변이 연산을 위해서는 임의로 선택된 2개의 타일간의 라우팅 경로를 변경한다. 이와 같이 특별히 고안된 교차연산과 변이연산을 이용하여 사용 가능한 솔루션 공간만을 탐색하는 것이 가능해진다. 정규형의 그물 구조 망에서는 [19]에서 소개된 것과 같이 좀 더 단순한 GA 인코딩, 교차연산, 변이연산을 사용할 수도 있다.

4.4 태스크 스케줄링

태스크 스케줄링을 위해서는 태스크들의 이동성(mobility)을 우선 순위로 사용하는 리스트(list) 스케줄링 알고리즘을 사용하였다. 태스크의 이동성은 가능한 빠른 시작 시간과 가능한 늦은 종료 시간의 차이를 의미한다. 이러한 시간을 알기 위해서는 태스크 그래프의 연결 에지가 가지는 통신 지연 시간을 알아야만 한다.

Marcon 등[12]은 에지의 통신 지연시간을 계산하기 위해서 통신 의존 및 계산 그래프(communication dependence and computation graph: CDCG)를 사용했다. 여기서는 CDCG를 사용하여 각 통신 부하의 패킷 지연 시간, 라우팅 지연시간 그리고 회선 쟁탈 지연시간을 측정했다. 회선 쟁탈 지연시간은 하나의 통신 회선을 2개 이상의 통신 부하가 공유할 때 발생한다.

본 논문의 기법에서는 통신 의존성을 태스크 그래프로부터 알 수 있다. 2개의 에지 $e_{i,j}$와 $e_{n,m}$에 대해서 태스크 그래프에 τ_j로부터 τ_n까지의 경로가 존재하면 에지 $e_{n,m}$는 $e_{i,j}$에 의존성을 가진다. 2개의 독립적인 통신 에지 $e_{i,j}$와 $e_{n,m}$가 하나의 통신 회선 ℓ_k를 공유하면 회선 쟁탈 지연 시간이 생길 수 있기 때문에 $e_{n,m}$를 $e_{i,j}$의 ℓ_k에 대한 후보 충돌 에지(candidate conflicting edge)라고 한다 $(e_{i,j} \overset{\ell_k}{\leftrightarrow} e_{n,m})$. 회선 쟁탈 지연 시간은 2개의 통신 에지 $e_{i,j}$와 $e_{n,m}$가 통신 회선 ℓ_k를 동시에 사용하고자 할 때만 발생하며 이 경우에 $e_{n,m}$를 $e_{i,j}$의 ℓ_k에 대한 충돌 에지(conflicting edge)라고 한다 $(e_{i,j} \overset{\ell_k}{\Leftrightarrow} e_{n,m})$.

본 논문에서는 목표 NoC가 타일간의 메시지를 플릿(flit) 단위로 전송하는 웜홀(wormhole) 스위칭 알고리즘을 사용한다고 가정한다. 에지 $e_{i,j}$의 통신 지연시간을 측정하기 전에 먼저 2가지 기본 요소인 σ_r과 $\sigma_l(e_{i,j})$를 측정해야 한다. σ_r은 라우터내에서 라우팅을 결정하는 데 걸리는 시간을 의미하며 $\sigma_l(e_{i,j})$는 회선을 통해서 하나의 플릿을 보내는 데 걸리는 최대 시간이다.

$\sigma_l(e_{i,j})$는 통신 에지 $e_{i,j}$를 위해서 사용되는 통신 회선의 동작 속도에 따라 다른 값을 가진다. $\sigma_l(e_{i,j})$는 다음과 같이 측정할 수 있다:

$$\sigma_l(e_{i,j}) = \left\lceil \frac{F}{B} \right\rceil \cdot \frac{1}{MIN_{\ell_k \in \Omega(e_{i,j})}(f(\ell_k))}$$

F와 B는 플릿 크기와 통신 회선의 비트 넓이를 나타낸다. $MIN_{\ell_k \in \Omega(e_{i,j})}(f(\ell_k))$는 $\Omega(e_{i,j})$내의 회선 중에 가장 낮은 속도를 가지는 통신 회선의 클럭 속도이다.

만약에 메시지 헤더 크기가 1 플릿이라면, 회선 쟁탈 지연시간이 없는 경우의 통신 시간은 다음과 같이 구할 수 있다[7]:

$$\sigma'(e_{i,j}) = \eta(e_{i,j}) \cdot (\sigma_r + \sigma_l(e_{i,j})) + \sigma_l(e_{i,j}) \cdot \left\lceil \frac{w(e_{i,j})}{F} \right\rceil$$

여기서 $\eta(e_{i,j})$는 $e_{i,j}$의 패킷이 출발 타일로부터 종착 타일까지 갈 때 거치는 라우터의 개수다. 통신 회선의 클럭 속도인 $f(\ell_k)$에 따라서 에지 $e_{i,j}$의 통신 지연 시간 $\sigma'(e_{i,j})$는 바뀐다.

만약에 $e_{i,j}$와 충돌하는 에지가 있다면, 회선 쟁탈 지연시간을 고려해야 한다. 그러나, 회선 쟁탈 시간은 스위칭 알고리즘, 가상 채널, 통신 시작 시간 등의 다양한 요소에 따라 달라지기 때문에 정확한 값을 계산하기는 힘들다. 정확한 통신 시작 시간을 알고 있다면 가상 채널을 사용하지 않는 웜홀 스위칭에 대해서는 회선 쟁탈 지연시간을 그림 8과 같이 측정할 수 있다. 공유되는 회선 ℓ_k에 메시지 C가 도착할 때, 이 메시지는 회선 ℓ_k에 C보다 먼저 도착한 메시지 A와 B가 회선 ℓ_k를 놓을 때까지 기다려야 한다. 그러므로, 에지 $e_{i,j}$에 대한 회선 쟁탈 지연시간을 다음과 같이 표현할 수 있다:

$$\sigma_c(e_{i,j}) = \sum_{\ell_k \in \Omega(e_{i,j})} \sigma_c(e_{i,j}, \ell_k) = \sum_{\ell_k \in \Omega(e_{i,j})} \{MAX_{\substack{\ell_k \\ e_c \Leftrightarrow e_{i,j}}} (\delta_r(e_c, \ell_k)) - \delta_a(e_{i,j}, \ell_k)\}$$

$\delta_r(e_c, \ell_k)$는 e_c의 메시지가 회선 ℓ_k를 놓아주는 시간이며 $\delta_a(e_{i,j}, \ell_k)$는 $e_{i,j}$의 메시지가 회선 ℓ_k에 도착하는 시간이다.

그러나, 메시지의 도착시간은 해당 태스크의 실행 시간에 따라 변할 수 있다. 즉, 회선 쟁탈 지연시간은 태스크의 실행시간에 의해 결정된다. 하지만, 설계 시간에 태스크의 정확한 실행 시간을 알 수는 없으므로 통신 시간을 구하기 위해서 다음과 같이 모든 후보 충돌 에지들이 고려되어야 한다:

$$\sigma_c(e_{i,j}) = \sum_{\substack{\ell_k \\ e_c \leftrightarrow e_{i,j}, \ell_k \in \Omega(e_{i,j})}} \sigma'(e_c)$$

전체 통신 지연시간은 $\sigma(e_{i,j}) = \sigma'(e_{i,j}) + \sigma_c(e_{i,j})$가 된다.

4.5 링크 속도 할당

회선의 속도 할당을 위해서는 Schmitz와 Al-Hashimi [20]가 제안한 전압과 클럭 속도 할당 알고리즘과 비슷한 방법을 사용했다. 그들의 알고리즘은 먼저 마감시간과 태스크간의 선행 관계를 바탕으로 각 태스크의 여유시간(slack time)을 측정한다. 다음은 여유시간을 가진 태스크 τ_i의 $\Delta E(\tau_i)$를 계산한다. $\Delta E(\tau_i)$는 τ_i에게 할당된 시간이 (낮은 클럭 속도로) Δt만큼 증가되었을 때의 전력 감소를 의미한다. 가장 큰 $\Delta E(\tau_i)$를 가진 태스크 τ_i에게 할당된 시간을 Δt만큼 증가시킨 후 여유시간을 가진 태스크가 없을 때까지 같은 과정을 반복한다.

이 알고리즘은 동적 전압 조절이 가능한 처리 소자에 할당된 태스크의 동작 속도를 결정할 수 있지만, 본 논문의 회선 속도 할당 알고리즘은 실행 시간에 변하지 않는 각 회선의 동작 속도를 결정한다. 각 회선의 여유시간을 측정하기 위해서는 그 회선을 공유하는 통신 부하를 가진 에지를 고려해야 한다. 회선의 여유시간은 이

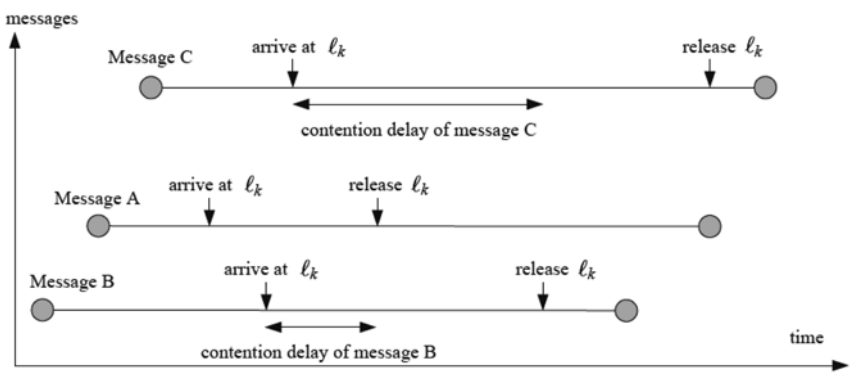
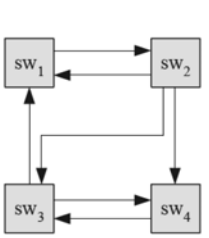


그림 8 충돌하는 통신들의 회선 쟁탈 지연시간

들 에지들의 여유시간 중의 최소값이다. 즉, $\xi(\ell_i) = MIN_{\ell_i \in \Omega(e_j)}(\xi(e_j))$. $\xi(\ell_i)$와 $\xi(e_j)$는 ℓ_i와 e_j의 여유시간이다.

4.6 GA 기반의 네트워크 구조 선택

더욱 효과적으로 저전력 NoC 시스템을 구축하기 위해서는 통신 회선과 스위치를 목표 응용의 통신 패턴을 바탕으로 최적화하는 응용에 특화된 NoC 망구조를 사용하는 것이 좋다[21,13]. 이전의 연구가 NoC 라이브러리내의 미리 정의된 네트워크 망 구조 중에서 가장 효과적인 하나를 찾는 것에 반해, 본 논문에서는 모든 가능한 NoC 망구조를 탐색하기 위해 GA 기반의 망구조 선택(GA-TS) 알고리즘을 사용한다. 그림 9는 응용에 특화된 NoC 망구조를 위한 GA 인코딩을 보여준다. 그림 9(a)의 망구조에 대해서 그림 9(b)에 보듯이 이진수의 2차원 배열을 가지고 스위치들 사이의 연결을 표현할 수 있다. 좌표 (i,j)에 있는 값은 스위치 sw_i로부터 sw_j로의 회선이 있는 지를 보여준다. 예를 들어, sw_2로부터 sw_3로의 회선이 있기 때문에 좌표 (2,3)의 값은 1이다.



src \ dst	sw_1	sw_2	sw_3	sw_4
sw_1	0	1	0	0
sw_2	1	0	1	1
sw_3	1	0	0	1
sw_4	0	0	1	0

(a) an example of NoC topology (b) switch connection encoding

그림 9 NoC 망 구조의 GA 인코딩

서로 다른 NoC 망구조는 다른 하드웨어 비용을 가진다. 예를 들어, 그림 9에 나타난 NoC 망구조는 3×1 스위치(sw_2) 1개, 2×2 스위치(sw_3) 1개, 그리고 2개의 1×2 스위치(sw_1와 sw_4)를 필요로 한다. 4개의 2×2 스위치가 필요한 그물망 구조에 비해서 전력소모와 면적 비용에서 더욱 효율적이다.

응용에 특화된 NoC 망구조에서도 표 1에 있는 GA-TA, GA-TM 그리고 GA-RPA의 GA 알고리즘을 사용할 수 있다. 그러나, 몇 가지 수정이 필요하다. 예를 들어, 그물망 구조의 NoC와는 다르게 회전식 교차연산을 통해 비정상적 솔루션이 만들어 질 수 있는데, 이것은 어떤 타일 매핑에서는 통신 부하를 위해 가능한 라우팅 경로가 존재하지 않을 수 있기 때문이다. 이 경우에 모든 통신 부하에 대해서 하나 이상의 가능한 경로가 생기도록 변이 연산을 통해 타일 매핑을 변경한다.

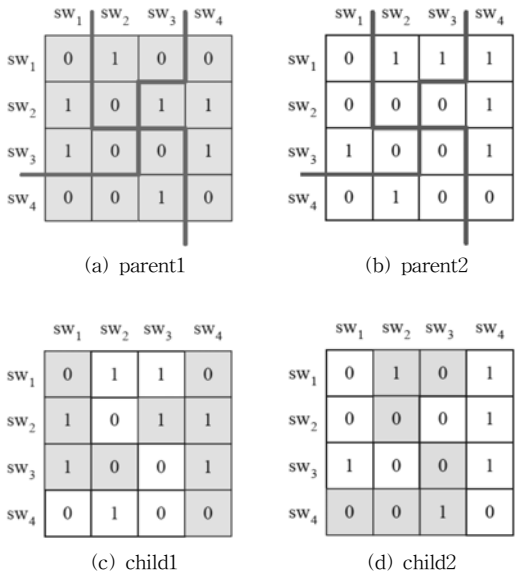


그림 10 NoC 망 구조를 위한 2차원 지리적 교차 연산

GA-TS의 교차연산을 위해서는 2차원 지리적(geographic) 교차 연산 알고리즘을 사용한다[18]. 이 방법에서는 2개의 부모 솔루션이 같은 방법으로 여러 개의 조각들로 나뉘며 자식 솔루션들은 그림 10과 같이 각 조각들을 혼합하여 만든다.

5. 실험 결과

실험에서는 태스크 할당, 타일 매핑, 라우팅 경로 할당, 회선 속도 할당의 각 단계에서의 최적화 기법의 효과를 측정해 보았다. 실험에서는 크게 2가지 형태의 태스크 그래프를 사용하였는데, 하나는 무작위로 생성된 가상의 태스크 그래프들이며 다른 하나는 [11]에서 소개된 실제 응용(H.263 부호기/복호기와 MP3 부호기/복호기를 가진 멀티미디어 시스템)의 태스크 그래프이다. 무작위로 생성된 가상의 그래프는 다양한 형태의 그래프에 대해서 실험해 보기가 유용하다. 그림 11은 무작위로 생성된 태스크 그래프 g1~g16에 대해서 다양한 최적화 설정에서의 통신 회선의 전력 소모를 보여주고 있다. 결과는 무작위 태스크 할당, 무작위 타일 매핑, XY-라우팅을 사용하고 회선 속도 조정을 사용하지 않는 설계 기법에 의해 구해진 전력 소모에 정규화되어 있다. 입력으로 사용한 그래프가 무작위로 생성한 가상의 그래프이기 때문에 실제 시스템 상에서 실행시킬 수가 없으므로 전력 소모값은 실제 전력 소모를 측정한 것이 아니고 3장에서 모델링한 전력 소모 값을 사용했다.

각 태스크 그래프의 첫 번째 막대는 단지 회선 속도 할당만을 적용시켰을 때의 결과이다(opt_{LS}). 두 번째 막

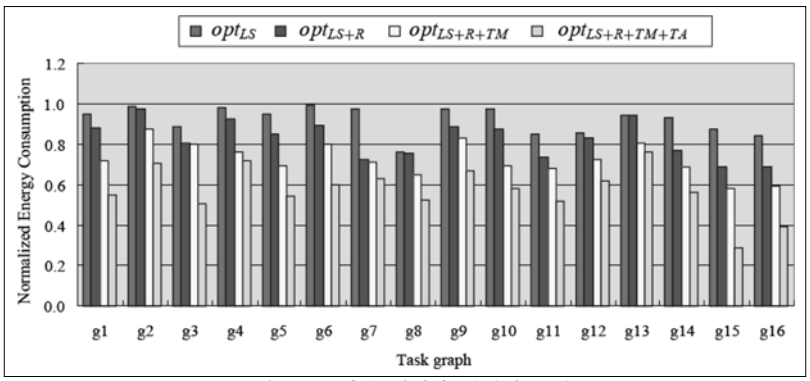


그림 11 중첩된 최적화 기법의 효과

대는 회선 속도 할당뿐만 아니라 라우팅 경로 할당을 위해 GA-RPA 알고리즘을 사용했을 때의 결과이다 (opt_{LS+R}). 세 번째 막대는 타일 매핑을 위해서 GA-TM이 추가로 사용되었을 때의 결과이다($opt_{LS+R+TM}$). 네 번째 막대는 모든 최적화 알고리즘들이 적용되었을 때의 전력 소모를 나타낸다($opt_{LS+R+TM+TA}$). 전력 소모는 opt_{LS}, opt_{LS+R}, $opt_{LS+R+TM}$, $opt_{LS+R+TM+TA}$에 의해서 각각 8%, 17%, 27%, 43%가 줄어들었다. 전력 소모 감소는 태스크 그래프의 특성(예, 여유 시간과 통신 부하량)과 무작위 설정의 성능에 의해 좌우된다. 예를 들면, 회선 속도 할당(opt_{LS})은 적은 전력 감소를 가져왔는데, 그것은 무작위 태스크 할당, 무작위 타일 매핑 그리고 XY-라우팅이 적은 여유 시간을 생성했기 때문이다. 이 실험 결과로부터 모든 설계 단계가 통신 전력에 큰 영향을 미치며 전력 소모는 모든 단계에서 최적화되어야 한다는 것을 알 수 있다. (리스트 스케줄링은 가장 널리 쓰이는 기법이므로 태스크 스케줄링 기법은 다른 기법과 비교하지 않았다.)

실험에서는 제안된 GA 기반의 NoC 설계 기법을 이전의 기법들과도 비교했다. Hu와 Marculescu는 분기와 한정(branch-and-bound: B&B) 알고리즘을 사용한 타일 매핑 기법 및 회선간의 통신 부하량을 균등화하는 라우팅 경로 할당 알고리즘을 제안했다[11]. 이 기법에 추가로 본 논문에서는 제안된 태스크 할당, 태스크 스케줄링 그리고 회선 속도 할당 알고리즘을 통합하여 실험에 사용하였다. 이 새로운 알고리즘을 본 논문에서는 BB로 표기한다.

B&B 알고리즘은 탐색 트리를 따라가며 최적의 솔루션을 찾는 기법이다. 이 알고리즘은 분기(branch)와 한정(bound)의 2단계로 구성된다. 매핑이 전혀 이루어지지 않은 루트(root) 노드에서 출발하여 분기 단계에서는 이전의 부분적으로 매핑된 솔루션으로부터 하나의 매핑되지 않은 처리 소자를 추가로 매핑시킴으로서 새로운 부분적으로 매핑된 솔루션(중간 노드)을 모든 처리 소자들이 매핑될 때(리프(leaf) 노드)까지 생성한다. 탐색 공간을 줄이기 위해서 한정 단계는 중간 노드로부터 추가의 분기를 진행할지를 결정한다. 한정 단계는 현 시점까지 생성된 리프 노드의 최소 비용인 UBC와 현재의 중간 노드로부터 생성될 리프 노드의 최소 비용인 LBC를 비교하여 만약 UBC<LBC이면 중간 노드를 분기시키는 것을 멈추고 다른 중간 노드를 탐색한다. [11]에서 라우팅 경로 할당은 휴리스틱을 사용하고 있다. 이 휴리스틱은 B&B 탐색 알고리즘의 분기 단계에서 매핑되는 타일들간의 라우팅 경로를 결정하는 데 사용되며 회선의 통신 부하량을 균등화시키도록 노력한다.

BB에서 솔루션의 비용은 통신회선의 동적 전력 소모에 비례하는 통신량으로부터 측정된다. 본 논문에서는 통신이 할당되지 않았을 때 회선의 전원을 차단할 수 있다는 가정으로 BB 알고리즘을 개선했다. 본 논문에서는 BB 알고리즘과 구별하기 위해서 개선된 알고리즘을 BB^+로 부른다. BB^+ 기법은 솔루션의 비용 측정 시에 누설 전류를 고려할 수 있다.

BB^+ 기법에서는 라우팅 경로 할당을 위한 휴리스틱도 개선하여 통신량이 이미 할당된 회선에 우선순위를 두어 추가로 통신량을 할당한다. 이러한 알고리즘으로 통신량이 할당되지 않은 회선의 숫자를 늘릴 수 있다.[1)]

그림 12는 라우팅 경로 할당을 위한 개선된 휴리스틱을 보여주고 있다. 먼저, 통신부하 리스트(CL)를 v의 가능한 라우팅 경로의 숫자를 기준으로 정렬한다. 다음은 순서대로 각 v의 라우팅 경로를 할당한다. 함수 *choose_link*는 회선의 통신부하량을 고려하여 x-방향 출력 회선과 y-방향 출력회선 중에 하나의 회선을 선택한다. 함수 *get_xlink*(*cur_tile,tar_tile*)와 *get_ylink*(*cur_*

1) 라우팅 경로 할당에서 회선의 대역폭(bandwidth)은 고려되지 않았다. 이것은 회선의 버퍼 크기와 관련이 있다. 그러나, 본 논문에서는 회선이 충분한 버퍼를 가지고 있어 대역폭에 대한 제약이 없다고 가정한다.

```
1:void path_allocation(){
2:   Sort CL by the number of possible paths of each υ;
3:   for each υ in CL {
4:     cur_tile := υ_src; tar_tile := υ_dst;
5:     while (cur_tile ≠ tar_tile) {
6:       ℓ := choose_link(cur_tile, tar_tile);
7:       cur_tile := dst(ℓ);
8:       W(ℓ) += υ_data;
9:}}}
10:struct link choose_link(cur_tile, tar_tile) {
11:  ℓ_X := get_xlink(cur_tile, tar_tile);
12:  ℓ_Y := get_ylink(cur_tile, tar_tile);
13:  if (cur_tile.x = tar_tile.x) return ℓ_Y;
14:  else if (cur_tile.y = tar_tile.y) return ℓ_X;
15:  else {
16:    if (W(ℓ_X) = 0 ) return ℓ_Y;
17:    else if (W(ℓ_Y) = 0 ) return ℓ_X;
18:    else if (W(ℓ_X) > W(ℓ_Y)) return ℓ_Y;
19:    else return ℓ_X;
20:}}
```

그림 12 BB$^+$의 라우팅 경로 할당 알고리즘

tile, tar_tile)는 *cur_tile*의 출력 회선 중에 x-방향 출력 회선과 y-방향 출력회선을 알려준다. BB의 라우팅 경로 알고리즘은 16, 17번째 줄을 제외하고는 그림 12와 동일하다.

[11]에서는 B&B 알고리즘이 짧은 시간에 최적의 솔루션을 찾게 해 준다고 주장한다. 그러나, 태스크 스케줄링과 회선 속도 할당을 고려하면 BB와 BB$^+$는 최적의 솔루션을 찾을 수 없는 데, 그 이유는 B&B에서 중간 노드의 정확한 비용을 태스크 스케줄링과 회선 속도 할당 전에는 알 수 없기 때문이다.

그림 13은 본 논문에서 제안된 GA 기반 알고리즘을 BB와 BB$^+$와 비교한 실험 결과이다. 네트워크 할당 과정만을 비교하기 위해서 모든 알고리즘은 미리 정해진 동일한 태스크 할당을 가지고 실험되었다. 결과는 무작위 타일 매핑과 XY 라우팅을 사용하는 기법에 대해 정규화한 값이다. 실험에서 보듯이, BB는 전력 소모를 평균 66% 정도 줄였지만 때로는 (태스크 그래프 g15) 무작위 매핑보다도 나쁜 결과를 낳았다. 이것은 BB가 누설 전류와 회선 속도 할당을 고려하지 않기 때문이다. BB$^+$와 GA 기반 기법은 전력 소모를 평균 22%와 39% 줄였다. GA 기반 기법은 BB 기법과 비교해 평균 28% 전력 소모를 감소시킨다.

멀티미디어 시스템의 태스크 그래프에 대한 실험에서는 태스크 그래프상에서 각 태스크가 처리 소자에 할당된 상태였기 때문에 타일 매핑과 라우팅 경로 할당 단계만을 평가해 보았다. 본 논문의 GA 기반 알고리즘은 무작위 타일 매핑과 XY 라우팅 기법에 비교해 35%의 전력 소모를 가져왔다.

표 2는 태스크 그래프의 특징과 $opt_{LS+R+TM}$ 알고리즘이 Pentium-III 500 MHz Linux 장비에서 실행될 때 걸린 시간을 보여주고 있다. 유전자 알고리즘의 실행시간은 개체군의 크기, 변이 확률 그리고 종료 조건 등의 다양한 요소에 의해 좌우된다. 실험에서 GA-TM과 GA-RPA의 개체군 크기는 각각 $|T|^2/2$와 $|T|$로 설정했다. 타일의 개수 $|T|$는 9(3×3)와 16(4×4)가 사용되었다. 표 2로부터 GA 기반 알고리즘은 태스크 그래프가 많은 에지를 가졌을 때, 즉 많은 개수의 통신 부하가 있을 때 긴 시간이 걸린다는 것을 알 수 있다. 태스크 그래프 g3와 g14와 같은 예외는 회선 속도 알고리즘의 특징 때문이다. 회선 속도 할당 알고리즘은 회선의 지연 시간을 여유시간이 없을 때까지 Δt 만큼 증가시켜가며 반복한다. 그래서 태스크 그래프가 여유시간이 많을 때는 시간이 오래 걸린다. 그러므로 빠른 회선 속도 할당 알고리즘을 사용하면 전체 알고리즘의 속도를 개선할 수 있다. BB와 BB$^+$는 $|T|$=9일 때는 잘 작동했지만 $|T|$=16일 때는 매우 느리게 동작했다. 특별한 속도 향상 기법 없이 BB는 $|T|$=16일 때 태스크 그래프 g2에 대해서 솔루션을 찾는 데 1시간 이상이 걸렸다. 그러나, GA 기반의 알고리즘은 비교적 복잡한 그래프에 대해서도 1분 이하의 시간이 걸려 설계 방법론으로서 사용이 가능하다는

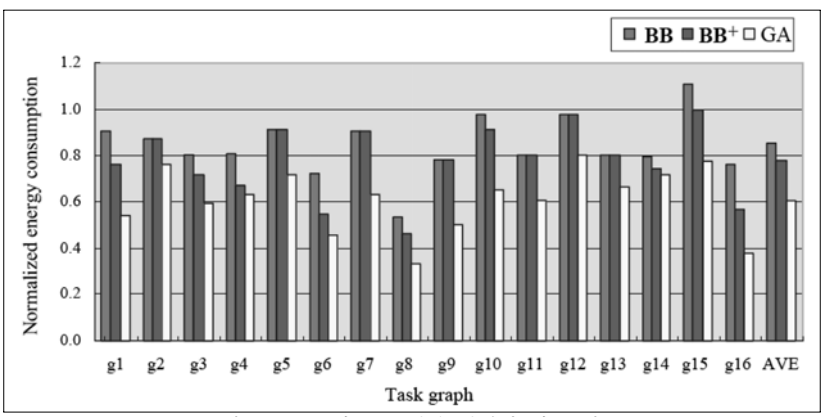


그림 13 B&B와 GA 기반 기법의 성능 비교

표 2 $opt_{LS+R+TM}$ 알고리즘의 실행시간 (time (sec))

TG	N_{node}/N_{edge}	3×3	4×4	TG	N_{node}/N_{edge}	3×3	4×4
g1	26/43	5.2	8.4	g9	30/29	6.5	43.9
g2	40/77	9.3	35.5	g10	36/50	12.5	57.9
g3	20/33	9.6	38.8	g11	37/36	9.0	26.3
g4	40/77	11.6	38.6	g12	24/33	5.0	9.6
g5	20/26	5.7	20.1	g13	31/56	9.9	58.6
g6	20/27	3.6	13.8	g14	29/56	6.2	19.3
g7	18/26	5.1	15.8	g15	12/15	3.2	9.4
g8	16/15	3.2	12.2	g16	14/19	2.8	4.2

* N_{node} = # of nodes and N_{edge} = # of edges

것을 보여주었다.

6. 결 론

본 논문에서는 가변 전압 회선을 가진 NoC 기반 시스템에서 통신으로 인한 전력 소모를 최적화하는 알고리즘을 제안했다. 제안된 알고리즘은 통신 망 구조 선택, 태스크 할당, 타일 매핑, 라우팅 경로 할당 그리고 회선 속도 할당 등 다양한 설계 단계에서 통신 전력을 최적화한다. 저전력의 NoC 시스템의 매우 큰 탐색 공간을 효과적으로 검사하기 위해서 유전자 알고리즘이 사용되었다. 제안된 알고리즘은 무작위 타일 매핑과 XY 라우팅을 사용하는 알고리즘에 비해 온-칩 네트워크의 전력 소모를 평균 35% 감소시켰다.

본 논문의 작업은 다양한 방향으로 확장될 수 있다. NoC의 비동기식(asynchronous) 통신 프로토콜 때문에 본 논문에서는 최악 통신 지연시간을 사용하였다. 그러나, 통신 에지간의 정확한 충돌관계를 파악하여 통신 지연시간에 대한 좀 더 정확한 값을 얻을 수 있을 것이다. 또 다른 쟁점은 각 라우터의 버퍼 크기이다. 타일 매핑과 라우팅 경로 할당에 따라서 요구되는 버퍼 크기가 달라지기 때문에 버퍼 크기에 대한 제약 조건이 있는 NoC 기반 시스템을 설계할 필요가 있다.

참 고 문 헌

[1] L. Benini and G. De Micheli. Networks on Chip: A New SoC Paradigm. *IEEE Computer*, 35(1): 70-78, 2002.

[2] W. J. Dally and B. Towles. Route Packets, Not Wires: On-chip Interconnection Networks. In *Proc. Design Automation Conference*, pages 684-689, 2001.

[3] H.-S. Wang, L.-S. Peh, and S. Malik. Power-Driven Design of Router Microarchitectures in On-Chip Networks. In *Proc.International Symposium on Microarchitecture*, pages 105-116, 2003.

[4] L. Shang, L.-S. Peh, and N. K. Jha. Dynamic Voltage Scaling with Links for Power Optimization of Interconnection Networks. In *Proc. International Symposium on High-Performance Computer Architecture*, 2003.

[5] J. Kim and M. Horowitz. Adaptive Supply Serial Links with Sub-1V Operation and Per-pin Clock Recovery. In *Proc. International Solid-State Circuits Conference*, 2002.

[6] X. Chen and L.-S. Peh. Leakage Power Modeling and Optimization in Interconnection Networks. In *Proc. International Symposium on Low Power Electronics and Design*, pages 90-95, 2003.

[7] J. Duato, S. Yalamanchili, and L. Ni. *Interconnection Networks*. Morgan Kaufmann, 1997.

[8] T. Simunic and S. Boyd. Managing Power Consumption in Networks on Chips. In *Proc. Design, Automation, and Test In Europe*, pages 110-116, 2002.

[9] V. Soteriou and L.-S. Peh. Dynamic Power Management for Power Optimization of Interconnection Networks Using On/Off Links. In *Proc. Symposium on High Performance Interconnects*, pages 15-20, 2003.

[10] F. Worm, P. Ienne, P. Thiran, and G. De Micheli. An Adaptive Low Power Transmission Scheme for On-chip Networks. In *Proc. International System Synthesis Symposium*, pages 92-100, 2002.

[11] J. Hu and R. Marculescu. Exploiting the Routing Flexibility for Energy/Performance Aware Mapping of Regular NoC Architectures. In *Proc. Design, Automation and Test in Europe Conference*, pages 10688-10693, 2003.

[12] C. Marcon, N. Calazans, F. Moraes, A. Susin, I. Reis, and F. Hessel. Exploring NoC Mapping Strategies: An Energy and Timing Aware Technique. In *Proc. Design, Automation and Test in Europe Conference*, pages 502-507, 2005.

[13] Umit Y. Ogras and R. Marculescu. Energy- and Performance-Driven NoC Communication Architecture Synthesis Using a Decomposition Approach. In *Proc. Design, Automation and Test in Europe Conference*, pages 352-357, 2005.

[14] T. Lei and S. Kumar. A Two-step Genetic Algorithm for Mapping Task Graphs to a Network on Chip Architecture. In *Proc. Euromicro Symposium on Digital Systems Design*, pages 180-187, 2003.

[15] N. Eisley and L.-S. Peh. High-Level Power Analysis for On-Chip Networks. In *Proc. Design, Automation and Test in Europe Conference*, pages 10688-10693, 2003.

[16] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik. Orion: A Power-Performance Simulator for Interconnection Networks. In *Proc. International Symposium on Microarchitecture*, pages 294-305, 2002.

[17] I. Oliver, D. Smith, and J. Holland. A Study of Permutation Crossover Operations on the Traveling Salesman Problem. In *Proc. International Con-*

ference on Genetic Algorithm, pages 224-230, 1987.

[18] A. B. Kahng and B. R. Moon. Toward More Powerful Recombinations. In *Proc. International Conference on Genetic Algorithms*, pages 96-103, 1995.

[19] D. Shin and J. Kim. Power-Aware Communication Optimization for Networks-on-Chips with Voltage Scalable Links. In *Proc. International Conference on Hardware/Software Codesign and System Synthesis*, pages 170-175, 2004.

[20] M. T. Schmitz and B. M. Al-Hashimi. Considering Power Variations of DVS Processing Elements for Energy Minimisation in Distributed Systems. In *Proc. International Symposium on System Synthesis*, pages 250-255, 2001.

[21] A. Jalabert, S. Murali, L. Benini, and G. De Micheli. xPipesComiler: A Tool for Instantiating Application-Specicific NoCs. In *Proc. Design, Automation, and Test in Europe*, pages 20884-20889, 2004.

신 동 군

1994년 서울대학교 계산통계학과 학사. 2000년 서울대학교 전산과학과 석사. 2004년 서울대학교 전기컴퓨터공학부 박사. 2004년~2007년 삼성전자 책임연구원. 2007년~현재 성균관대학교 정보통신공학부 전임강사. 관심분야는 임베디드 시스템, 저전력 시스템, 실시간 시스템, 컴퓨터 구조

김 지 홍

1986년 서울대학교 계산통계학과 학사 1988년 University of Washington 컴퓨터과학과 석사. 1995년 University of Washington 컴퓨터과학 및 공학과 박사. 1995년~1997년 미국 Texas Instruments사 선임연구원. 1997년~현재 서울대학교 전기 컴퓨터공학부 교수. 관심분야는 임베디드시스템, 컴퓨터구조, 저전력시스템