Journal of The Institute of Electronics and Information Engineers Vol.53, NO.8, August 2016 http://dx.doi.org/10.5573/ieie.2016.53.8.049
ISSN 2287-5026(Print) / ISSN 2288-159X(Online)

논문 2016-53-8-6

링크 도선 길이를 고려한 고성능 비동기식 NoC 토폴로지 생성 기법

(Link-wirelength-aware Topology Generation for High Performance Asynchronous NoC Design)

김 상 헌*, 이 재 성*, 이 재 훈*, 한 태 희**

(Sang Heon Kim, Jae Sung Lee, Jae Hoon Lee, and Tae Hee Han©)

요 약

어플리케이션 특성에 따라 링크 대역폭 요구량이 다양하게 분포하는 이종 (heterogeneous) 아키텍처 기반 네트워크-온-칩 (Network-on-Chip, NoC) 설계에 있어 링크 지연 시간이 독립적으로 설정될 수 있는 비동기식 프로토콜을 적용할 경우 동기식 설계에 비해 성능 향상의 기회가 확대될 수 있다. 본 논문에서는 비동기식 NoC에서 각 링크의 대역폭 요구량과 도선 길이에 따른 지연 시간 모델을 제시하고 이를 최적화하는 simulated annealing (SA) 기법을 이용한 플로어플랜 기반 토폴로지 생성 알고리즘을 제안하였다. 생성된 토폴로지와 각 링크의 도선 길이를 기반으로 대응하는 도선 지연시간을 계산하고 로직 합성 단계를 거쳐 생성된 gate-level netlist와 표준지연시간 모델을 이용한 시뮬레이션을 통해 성능을 측정하였다. 링크 도선 길이를 고려하지 않은 일반적인 토폴로지 생성 알고리즘인 TopGen과 비교하여, 제안된 알고리즘이 다양한 어플리케이션 실험에서 평균 13.7% 지연 시간 단축 효과 및 처리량 측면 지표인 실행 시간에서 평균 11.8% 감소 효과가 있음을 확인할 수 있었다.

Abstract

In designing heterogeneous architecture based application-specific network-on-chips (NoCs), the opportunities of performance improvement would be expanded when applying asynchronous on-chip communication protocol. This is because the wire latency can be configured independently considering the wirelength of each link. In this paper, we develop the delay model of link-wire-length in asynchronous NoC and propose simulated annealing (SA) based floorplan-aware topology generation algorithm to optimize link-wirelengths. Incorporating the generated topology and the associated latency values across all links, we evaluate the performance using the floorplan-annotated sdf (standard delay format) file and RTL-synthesized gate-level netlist. Compared to TopGen, one of general topology generation algorithms, the experimental results show the reduction in latency by 13.7% and in execution time by 11.8% in average with regards to four applications.

Keywords : Asynchronous Network-on-Chip (NoC), Topology generation, High performance design

* 학생회원, ** 평생회원, 성균관대학교 정보통신대학
(College of Information & Communication Engineering, Sungkyunkwan University)
© Corresponding Author (E-mail : than@skku.edu)
※ 본 연구는 미래창조과학부 및 정보통신기술진흥센터의 대학ICT연구센터 육성지원사업 (IITP-2016-H8501-16-1005) 및 2015년도 정부(교육부)의 재원으로 한국연구재단의 기초연구사업 연구결과로 수행되었음 (NRF-2015R1D1A1A01057278)

Received ; May 17, 2016 Revised ; July 22, 2016
Accepted ; July 29, 2016

Ⅰ. 서 론

클럭 스큐, 클럭 분배 트리의 복잡도 증가와 같은 문제들로 인해 고성능, 저전력 달성 한계에 당면하고 있는 동기식 회로의 대안으로 핸드셰이크 (handshake) 프로토콜을 이용하는 비동기식 설계 방식이 다시 주목을 끌고 있다[1]. 특히 가장 느린 신호 경로 지연시간을 기준으로 전역 클럭 주파수가 결정되는 동기식 회로에 비해 블록 간의 다양한 종료시점을 활용해 성능 향상이

가능한 비동기식 설계 방식은 평균 지연 시간을 최적화함으로써 클럭 주파수에 의존하지 않고 고성능화를 도모할 수 있다[2]. 또한 비동기식 설계는 다수의 클럭 도메인을 사용하는 동기식 시스템에서 발생하는 metastability 문제도 해결할 수 있다[3]. 이러한 장점들에도 불구하고 비동기식 설계는 해저드 (hazard) 문제 해결의 어려움, CAD tool의 부재와 검증의 어려움 등의 이유로 주류가 되지 못해 왔다. 그러나 ITRS 2012년도 자료에 의하면 2022년경에는 면적 기준으로 칩 내 회로의 45%가 비동기식 설계 방식을 취할 것으로 예측하고 있으며[4], 2014년 발표된 IBM의 신경망 프로세서 SyNAPSE[5] 사례에서 볼 수 있듯이, 동기식 설계 기법을 보완하는 형태로 점차 활용 범위가 확대될 것으로 전망된다.

특히 오늘날 다양한 코어(core)가 다수 집적되고 블록 간 데이터 전송량이 폭증함으로써 등장한 온칩 커뮤니케이션 아키텍처인 네트워크-온-칩(Network-on-Chip, NoC) 설계에서 인터커넥션 지연 시간과 다양한 링크 대역폭을 수용하기 위해 비동기식 설계방식을 적용하려는 시도가 활발히 진행되고 있다[6~7]. 비동기식 NoC에서는 동작 종료 시점이 다양하게 설정되므로, 라우터 간 또는 라우터-코어 간 링크들이 링크 도선 길이와 같은 물리적인 특성에 의하여 각각 독립적인 가용 대역폭을 갖는다[8]. 따라서 비동기식 NoC에서는 링크 도선 길이가 전체 시스템의 성능에 중대한 영향을 미치며 이를 최적화함으로써 성능 향상이 가능하다.

이러한 특성은 특정용도 (application-specific) 시스템 설계 시 더욱 효과적으로 적용될 수 있다. 특정용도 NoC 설계의 경우 어플리케이션 특성에 의하여 각 링크에서의 대역폭 요구량이 상이하며 이는 설계 초기 단계에서 예측 가능하다. 또한 특정용도 NoC는 보통 다양한 면적 특성을 가진 이종 (heterogeneous) 코어들의 집합으로 구성된다[9]. 이로 인해 플로어플랜 (floorplan) 시 코어들의 위치에 따라 각 링크의 도선 길이 또한 균일하지 않다. 만약 플로어플랜 결과 대역폭 요구량이 높은 링크의 도선 길이가 길게 설정되고 대역폭 요구량이 낮은 링크의 도선 길이가 상대적으로 짧게 설정된다면 전체적인 성능 저하를 유발하게 된다. 따라서 각 링크의 대역폭 요구량에 따라 도선 길이를 최적화시킴으로써 보다 고성능을 달성할 수 있다[10].

NoC 설계 흐름에서 링크 도선 길이의 최적화는 보통 토폴로지를 생성한 후 플로어플랜 단계에서 수행되므로 토폴로지 생성 단계에서는 링크 도선 길이와 같은 물리적 정보를 고려하지 않아 왔다[11~12]. 플로어플랜은 토폴로지에 상당히 의존적이므로 토폴로지 생성 단계에서 플로어플랜을 고려하는 것이 성능 최적화와 timing closure에 유리하다는 것은 잘 알려져 있다. 가장 긴 도선 길이를 갖는 링크에 의해 최대 동작 주파수 성능이 결정되는 동기식 NoC와는 달리 각 링크 간 핸드셰이크 주기가 독립적으로 결정되는 비동기식 NoC에서는 전체적으로 평균 도선 길이를 최적화해야한다는 측면에서 토폴로지 생성 단계에서 플로어플랜 결과를 반영하는 효과가 더 크다.

본 논문에서는 특정용도 시스템의 고성능 비동기식 NoC 설계를 위해 토폴로지 생성 단계에서 가상 (virtual) 플로어플랜 정보를 반영한 토폴로지 생성 알고리즘을 제안하고자 한다. 이를 위해 링크 도선 길이와 도선 지연 시간 모델로부터 각 링크의 지연 시간을 구하고, 홉 수, 링크 경쟁과 같은 토폴로지 특성과 링크 지연 시간을 함께 고려한 비동기식 NoC에서의 지연 시간 모델을 먼저 도출한다. 도출된 지연 시간 모델을 기반으로 메타 휴리스틱 알고리즘인 simulated annealing (SA)를 적용해 그림 1에 보인 바와 같이 가상 플로어플랜 → 지연 시간 모델 계산 → 토폴로지 평가 → 토폴로지 변형 과정을 반복 수행함으로써 최종의 토폴로지를 생성한다.

본 논문의 Ⅱ장에서는 비동기식 NoC 및 링크 도선 길이를 고려한 NoC 설계 기법에 대한 관련 연구들을 소개하고, Ⅲ장에서는 고성능 비동기식 NoC를 위한 링크 도선 길이 최적화의 구체적인 목표에 대해 설명한다. Ⅳ장에서는 링크 도선 길이를 최적화하는 SA 기반

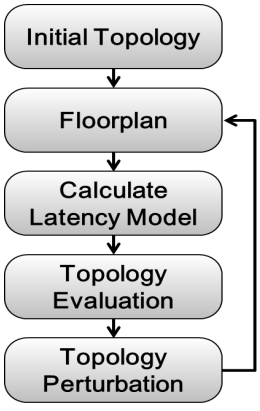


그림 1. 제안하는 가상 플로어플랜-토폴로지 생성 알고리즘의 흐름도

Fig. 1. Flowchart of virtual floorplan-topology generation algorithm

토폴로지 생성 알고리즘을 제안하고, Ⅴ장에서는 제안하는 알고리즘으로부터 얻은 토폴로지, 가상 플로어플랜에 대하여 RTL 수준에서의 성능 평가 실험 결과를 보인다. 마지막으로 Ⅵ장에서 결론을 정리한다.

Ⅱ. 관련 연구

2000년대 중반부터 비동기식 라우터 기반 NoC 아키텍처 연구가 이루어져 왔다. 대표적으로 QoS (Quality of Service) 및 가상 채널 (virtual channel) 을 지원하는 QNoC[13]와 적응적 라우팅 알고리즘을 적용한 Hermes-AA[14]가 있다. 하지만 플로어플랜의 결과로 결정되는 링크 지연 시간이 해당 링크의 가용 대역폭에 직접적인 영향을 주는 비동기식 NoC에서는, 라우터의 성능 향상만으로 전체적인 성능 향상을 기대하기 어렵다. 따라서 보다 고성능을 달성하기 위해서는 시스템 설계 단계에서 각 링크의 지연 시간을 최적화하는 접근이 필요하다.

비동기식 NoC 구조 연구는 라우터 구조에 초점을 맞춘 경우가 많고 시스템 수준에서의 연구는 부족한 편이다. D. Gebhardt et al.은 특정용도 비동기식 NoC에서 대역폭 요구량이 높은 링크에 파이프라인을 적용하였으며[15], J. You et al.은 코어를 먼저 가상 플로어플랜한 후 라우터를 휴리스틱한 방법으로 삽입하여 평균 링크 도선 길이를 최소화하고 위의 링크 파이프라이닝을 함께 적용시키는 최적화 기법을 제안하였다[8].

이와 같이 기존의 시스템 수준 비동기식 NoC 연구들에서는 주로 토폴로지 생성 후 가상 플로어플랜 단계에서 링크 도선 길이의 평균값을 최소화하거나 대역폭 요구량이 많은 링크를 우선적으로 배치하여 도선 길이를 최적화하였다. 이러한 방법들은 국부적 최적화에 그치고 있어, 전체 평균 지연 시간 또는 전체 평균 처리량과 같은 합리적인 성능 모델로 전체 시스템 성능 평가에 기반 한 최적화에는 적합하지 못하다. 또한 기존 비동기식 NoC 연구들에서는 토폴로지 생성 단계에서 링크 도선 길이를 고려한 사례가 없었다. 본 논문에서는 비동기식 NoC에서의 성능 평가 모델을 제안하고 이를 최적화함으로써 전체 시스템의 성능 향상을 달성하고자 한다. 토폴로지 생성 단계에서 토폴로지의 성능을 정확하게 평가할 수 있는 성능 평가 모델을 제안하기 위해 가상 플로어플랜의 결과를 사용한다.

Ⅲ. 링크 도선 길이 최적화의 문제

본 장에서는 링크 도선 길이 최적화의 문제에 대해 사례 연구로 분석한다. 이를 통해 비동기식 NoC에서의 분석적 성능 평가 모델을 제안한다. 최종적으로 제안한 성능 평가 모델을 이용하여 링크 도선 길이 최적화의 문제를 정의한다.

1. 링크 도선 길이 최적화의 방향성

앞서 언급하였듯이 비동기식 회로는 핸드셰이크 프로토콜에 의해 링크 데이터 전송이 동기화 된다. 그림 2는 4-위상 프로토콜에서의 핸드셰이크 과정을 보여주고 있다. 그림 2에서 알 수 있듯이 한 번의 데이터 전송을 위해 두 번의 핸드셰이크 신호 왕복이 일어난다.

도선 길이가 긴 라우터-라우터 간 또는 라우터-코어 간 링크에서 두 번의 핸드셰이크 신호 왕복은 링크의 성능 측면에서 불리하게 작용한다. 이러한 이유로 비동기식 NoC의 링크에서는 보통 한 번의 핸드셰이크 신호 왕복으로 데이터를 전송시키는 2-위상 프로토콜 (혹은 LEDR (Level-Encoded Dual-Rail)[16])을 사용한다[6]. 하지만 여전히 핸드셰이크 신호의 왕복은 링크 성능 향상에 병목점이 된다. 따라서 비동기식 NoC의 링크 도선 길이를 최소화 시키는 것은 성능 향상 측면에서 필수적인 요소이다. 기존 연구들은 링크 도선 길이의 평균값을 최소화하는 방법으로 접근하였지만 그림 3의 사례 연구를 통해 알 수 있듯이, 링크 도선 길이의 평균값만을 최소화하는 것이 성능 향상과 항상 연결되는 것은 아니다.

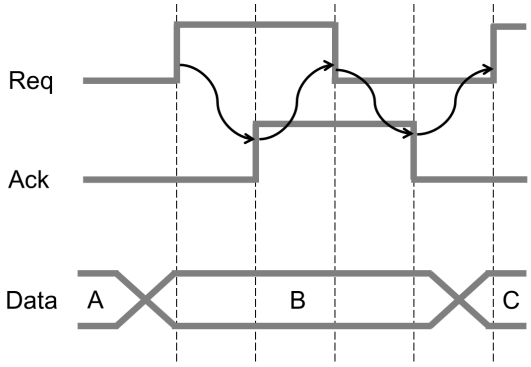
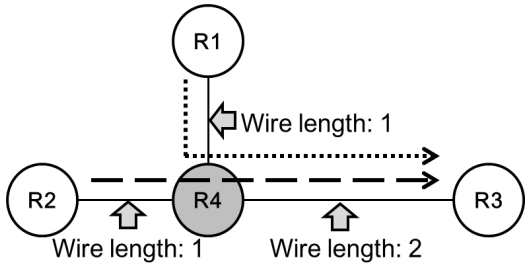
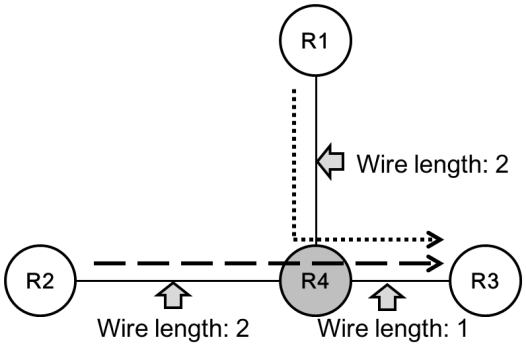


그림 2. 4-위상 핸드셰이크 프로토콜의 동작
Fig. 2. 4-phase handshake protocol.



(a) 사례 1: 대역폭 요구량이 높은 링크가 긴 도선 길이를 갖는 경우



(b) 사례 2: 대역폭 요구량이 높은 링크가 짧은 도선 길이를 갖는 경우

그림 3. 사례 연구: 대역폭 요구량과 도선 길이와의 관계
Fig. 3. Case study: Relationship between required bandwidth and link wire length.

그림 3의 두 사례는 코어들의 면적이 다양하게 분포되는 어플리케이션의 플로어플랜 결과로 발생할 수 있는 상황이다. 두 사례는 각각 Router1 → Router4 → Router3, Router2 → Router4 → Router3의 경로를 갖는다. 두 경로가 겹치는 Router4 → Router3의 링크는 대역폭 요구량이 다른 링크보다 크게 설정 된다. 이러한 상황에서 링크 도선 길이의 평균값은 사례 1(그림 3-(a))의 경우가 유리하게 보인다. 하지만 실제로는 사례 1의 경우 Router4에 패킷들이 빠르게 도착하여 계속 쌓이는 반면 사례 2(그림 3-(b))의 경우 Router3으로 빠르게 패킷을 내보낼 수 있어서 결과적으로 사례 1보다 더 좋은 성능을 내게 된다. 이러한 경우를 반영하기 위해 단순히 평균 링크 도선 길이를 최소화하는 것보다는 경로 충돌, 링크 당 통신량과 같이 토폴로지에 의해 결정되는 요소들을 추가적으로 고려해야 한다.

2. 비동기식 NoC에서의 지연 시간 모델

동기식 NoC에서와는 달리 비동기식 NoC에서의 성능 평가 모델은 좀 더 복잡하게 계산된다. 플로어플랜의 결과인 링크 지연 시간과 경로 충돌, 홉 수와 같이 토폴로지로부터 결정되는 부분을 모두 포함하는 성능 평가 모델을 설정하고 최적화한다면 보다 고성능 설계가 가능하다. 하나의 플릿 (flit)에 대하여 출발지에서 목적지로 전송하는데 걸리는 지연 시간은 아래의 식 (1)과 같이 링크에서의 지연 시간과 라우터에서의 대기 시간으로 나눌 수 있다.

$$Latency_{asynch} = \sum_{i=1}^{N_{hop}} LD_i + \sum_{i=1}^{N_{hop}-1} WT_i \quad (1)$$

식 (1)에서의 LD_i는 i번째 링크에서의 지연시간 (Link Delay)이며 WT_i은 i번째 라우터에서의 대기 시간 (Waiting Time)이다. 또한 N_{hop}는 해당 경로에서의 홉 수이다. 링크 지연 시간은 도선 길이에 따라 결정됨으로 플로어플랜의 결과로부터 얻을 수 있는데, 이 때 RC chain 도선 모델의 지연 시간 식 (2)를 이용한다.

$$\tau = \frac{RC}{2} = \frac{r_w c_w L^2}{2} \quad (2)$$

식 (2)에서 r_w, c_w는 각각 도선의 단위 길이 당 저항, 단위 길이 당 커패시턴스이며, L은 도선 길이이다. 식 (2)에서 볼 수 있듯이 도선의 지연 시간은 도선 길이의 제곱에 비례한다.

라우터에서의 대기 시간은 라우터의 중재 방식에 따라 달라지는데 최악의 상황 (worst-case)을 고려하여 계산한다. 여기서 worst-case는 출력 포트에서 경로 충돌 시 최하위의 우선순위를 가졌을 때이다. Worst-case를 고려함으로써 링크 경쟁, 링크 당 대역폭 요구량과 같은 부분을 반영할 수 있다.

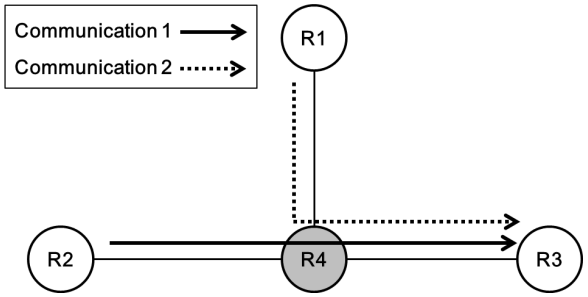


그림 4. 사례 연구: 경로 충돌 시 라우터 내 대기 시간
Fig. 4. Case study: Waiting time at router buffer when link contention occurs.

그림 4는 Router1 → Router3, Router2 → Router3의 두 경로가 Router4에서 충돌하는 것을 보여준다. 따라서 Router4 → Router3 링크에는 두 경로가 중첩된

다. 이 때 Communication 1의 플릿이 worst-case에서 라우터에 대기하는 시간은 Communication 2의 플릿이 먼저 선택되어 라우터로부터 전송되는 시간이다. 즉, Router4 → Router3 링크에서 플릿 1개가 전송되는 데 걸리는 지연시간으로 계산할 수 있다. 이를 일반화하면 라우터 대기 시간을 다음의 식 (3)과 같이 모델링 할 수 있다.

$$WT_i = (NP_{i+1} - 1) \times LD_{i+1} \quad (3)$$

식 (3)에서 WT_i은 경로 중 i번째 라우터에서의 대기 시간이고, NP_{i+1}은 i+1번째 링크에서 충돌하는 경로의 수(Number of Paths)이다. 마지막으로 LD_{i+1}는 i+1번째 링크의 지연 시간이다. 그림 4의 상황을 대입하면 NP_{i+1} = 2 이며, LD_{i+1}는 Router4 → Router3 링크에서의 지연 시간일 것이다. (NP_{i+1}-1)은 해당 경로와 i+1번째 링크를 함께 경쟁 중인 다른 경로의 수를 나타낸다. 결과적으로 식 (1)은 다음의 식 (4)와 같이 풀어 쓸 수 있다.

$$Latency_{asynch} = \sum_{i=1}^{N_{hop}} LD_i + \sum_{i=1}^{N_{hop}-1} \{(NP_{i+1} - 1) \times LD_{i+1}\} \quad (4)$$

식 (4)에서 NP과 N_{hop}은 토폴로지로부터, LD는 플로어플랜으로부터 결정되는 변수들이다. 식 (4)를 통해 지연 시간을 최소화하기 위해서는 홉 수(N_{hop})와 경로 충돌의 수(NP_{i+1}) 그리고 링크 도선 길이(LD)를 최소화해야 한다는 것을 알 수 있다.

3. 링크 도선 길이 최적화의 문제 정의

주어진 CTG (Communication Task Graph)에서 고성능 NoC를 달성하기 위해 링크 도선 길이를 최적화하는 플로어플랜, 토폴로지를 탐색하는 것이 본 논문에서 다루고자 하는 문제이다. 이를 위해 앞서 제안했던 식 (4)의 지연 시간 모델을 사용한다.

CTG는 태스크 간 통신 패턴을 나타내는 태스크 특성 그래프이며 $G=<V,E>$로 표현된다. 여기서 $v_i \in V$ 는 태스크를 나타내고, $e_{i,j} \in E$ 는 태스크가 v_i가 v_j로 데이터를 보내는 통신을 나타낸다. 이를 통해 비동기식 NoC에서의 지연 시간 최소화 문제를 다음과 같이 정의하였다.

Asynchronous NoC Latency Optimization Problem

Given G = <V, E> and the number of routers,
find topology and floorplan **such that**
$\sum_{\forall e_{i,j}} Latency_{asynch}(e_{i,j})$ **is minimized**

즉, 주어진 CTG와 라우터 수로부터 모든 커뮤니케이션들의 평균 지연 시간을 최소화하는 토폴로지와 플로어플랜을 검색하는 문제이다. 다음 장에서는 위의 문제를 풀기 위해 SA 기반의 최적화 알고리즘을 제안한다.

Ⅳ. 제안하는 토폴로지 생성 알고리즘

본 장에서는 고성능의 비동기식 NoC를 달성하기 위한 SA기반 토폴로지 생성 알고리즘을 제안한다. SA 기법을 간략히 설명한 후 이를 풀고자 하는 문제에 적용시킨다.

1. Simulated annealing 기법

SA 기법은 전역 최적화 문제에 대한 일반적인 확률적 메타 휴리스틱 알고리즘으로 NP-hard 문제의 최적해를 찾는데 쓰일 수 있다. 특히 본 논문에서 다루고자 하는 비정형 토폴로지 생성 문제와 같이 다수의 후보해 집단을 생성하기 어려운 문제의 경우 하나의 해를 변형시키며 최적의 해를 찾는 SA 기법이 적절하다. SA 기법의 간략한 의사 코드는 그림 5와 같다.

SA 기법에서는 전역 인자 T (Temperature)가 사용되는데 임의의 값으로 초기화된다. 주어진 탐색 공간 내에서 현재의 해 근방에 있는 해를 임의로 찾고, 만약 현재의 해보다 우수하면 현재의 해를 버리고 임의의 해로 교체한다. 반대로 현재의 해보다 우수하지 못하면 그 차이(ΔCost)를 계산한 후 그 차이와 T 값에 의해 결정되는 확률로 임의의 해를 받아들일지 결정한다. 이 때 받아들이는 확률은 T가 클수록, ΔCost가 작을수록 커지게 된다. 마지막으로 전역 인자 T 값을 감소시키며 위의 동작을 T가 미리 정한 값보다 작아질 때까지 반복한다. T가 큰 알고리즘 초기에는 임의의 해가 현재의 해보다 비록 좋지 못하더라도 탐색 공간을 탐색하기 위해 열등한 해를 높은 확률로 수용시킨다. 반면 T가 0에 가까워짐에 따라 탐색 공간을 충분히 탐색했다고 판단하여 현재의 해를 유지하거나 최대한 전역 최적해에 가까운 해로 이동한다. 이러한 방법은 지역 최적

Simulated Annealing(SA) Algorithm

Input : Search space

```
1:  set initial temperature(T)
2:  set initial solution(S)
3:  repeat
4:      generate a perturbation
5:      calculate new solution's cost(newCost)
6:      if newCost < oldCost
7:          accept new solution
8:      else
9:          accept new solution with probability p(ΔCost,T )
10:     decrease the temperature
11: until the temperature is frozen (T < predetermined number)
```

Output : The solution at T = predetermined number

그림 5. SA 기법의 의사코드
Fig. 5. Pseudo-code of SA algorithm.

점에 빠졌을 때의 대책이 될 수 있으며 전역 최적해에 근사한 해를 구할 수 있다.

2. SA 기반 토폴로지 생성 알고리즘

본 논문에서 풀고자하는 문제를 SA에 적용하면 그림 6과 같은 알고리즘 흐름이 될 것이다. 토폴로지 생성 알고리즘은 CTG와 라우터 수를 입력으로 하여 초기 토폴로지를 생성함으로써 시작된다 (Line 2). 초기 토폴로지는 TopGen[17]이라는 비정형 토폴로지 생성 알고리즘으로부터 얻는다. TopGen은 클러스터링 기반 토폴로지 생성 알고리즘으로 홉 수와 링크 당 통신량을 최적화하는 토폴로지를 생성한다. 전역 변수 T의 값을 100으로 초기화하여 (Line 1) 0이 될 때까지 Line 4-16의 동작을 반복하며, 반복 할 때마다 T를 1씩 감소시킨다 (Line 17-18). 마지막으로 T가 0일 때의 토폴로지와 가상 플로어플랜 결과를 알고리즘의 결과물로 얻는다. 알고리즘을 자세히 설명하기 위해 반복문 내 주요 함수들에 대해서 살펴보겠다.

가. 가상 플로어플랜과 경로 설정

식 (4)의 지연 시간 모델을 계산하기 위해서는 각 링크의 지연 시간을 알아야 한다. 이는 플로어플랜의 결과로 결정되며 따라서 토폴로지 생성 알고리즘에 플로어플랜을 포함해야한다 (Line 5). 본 논문에서는 가상 플로어플랜을 위해 Parquet이라는 툴을 사용하였다[11]. Parquet은 입력으로 받은 토폴로지 내에서 도선 길이

SA-based Topology Generation Algorithm

Input : CTG, # of Routers

```
1:  set initial temperature(T = 100)
2:  set initial topology( old_Topology )
3:  repeat
4:      Perturb( old_Topology, new_Topology )
5:      Floorplan( new_Topology )
6:      PathAllocation( new_Topology )
7:      new_Latency = Estimation( new_Topology )
8:      if new_Latency < old_Latency
9:          old_Topology = new_Topology
10:         old_Latency = new_Latency
11:     else
12:         ΔCost = | old_Latency – new_Latency |
13:         accept_prob = CalculateProb( ΔCost, T )
14:         if random(0,1) < accept_prob
15:             old_Topology = new_Topology
16:             old_Latency = new_Latency
17:     T = T - 1
18: until the temperature is frozen (T < 0)
```

Output : *old_Topology* at $T = 0$

그림 6. SA 기반 토폴로지 생성 알고리즘의 의사코드
Fig. 6. Pseudo-code of SA-based Topology Generation algorithm.

최소화와 전체 면적 최소화를 진행하며 결과물로 최적화된 플로어플랜 그래프, 각 링크의 도선 길이를 출력한다.

지연 시간 모델을 계산하기 위해 각 링크에서의 지연 시간뿐만 아니라 각 링크의 예상 통신량이 필요하다. 이를 위해 CTG상에 존재하는 모든 커뮤니케이션마다 토폴로지에서의 최단 경로를 설정하고 각 링크마다 예상되는 통신량을 계산한다 (Line 6).

나. 토폴로지 평가

토폴로지의 평가를 위해 식 (4)의 지연 시간 모델을 이용한다 (Line 7). 가상 플로어플랜과 경로 설정 함수로부터 얻은 정보들을 바탕으로 각 커뮤니케이션의 지연 시간을 계산하고 이를 합산한다. 평가 항목이 지연 시간이므로 작은 값을 가질수록 성능이 더 높은 토폴로지라고 할 수 있다. 따라서 만약 이전의 해($old\_Latency$)보다 지연 시간 모델의 값이 더 작게 나타나면 더 좋은 결과라 판단하여 이를 새로운 해로 받아들인다 (Line 8-10). 그 반대의 경우에는 이전의 토폴로지에서의 지연 시간 값($old\_Latency$)과 현재 토폴로지의 지연 시간 값($new\_Latency$)의 차이(ΔCost)를 구

하고, 전역 변수 T와 지연 시간 값 차이(ΔCost)에 따라 달라지는 확률 값($accept\_prob$)을 구한다 (Line 11-13). 이 확률로 현재의 해를 새로운 해로 받아들인다 (Line 14-16).

다. 확률적 수용을 위한 확률 설정 함수

수용 확률($accept\_prob$)은 ΔCost와 전역 변수 T에 의해 결정되며 (Line 13), T는 클수록 그리고 ΔCost는 작을수록 값이 커지게 된다. 이를 위한 확률 설정 함수 (CalculateProb)를 다음의 식과 같이 설정하였다.

$$accept_{prob} = e^{-\frac{k \cdot \Delta Cost}{T}} \quad (5)$$

식 (5)는 SA 기반 알고리즘의 확률적 수용 함수에서 전형적으로 쓰이는 식의 형태이다[18]. 식 (5)에서 k는 어플리케이션마다 ΔCost의 값이 다양하게 분포되는 것에 의해 확률이 비이상적으로 설정되는 것을 방지하기 위한 상수이다. 식 (5)를 통해 전역 변수 T가 크고 ΔCost의 값이 작은 경우에는 1에 가까운 값으로, 반대의 경우에는 0에 가까운 값으로 확률 값($accept\_prob$)을 설정할 수 있다.

라. 토폴로지 변형

마지막으로 탐색 공간 내에서 현재의 해 근방에 있는 임의의 해를 찾기 위해 이전의 토폴로지를 변형시켜 새로운 토폴로지를 생성한다 (Line 4). 본 논문에서는 이를 위해 이전의 토폴로지에서 가장 긴 도선 길이를 갖는 라우터-라우터 간 링크를 검색한다. 그 다음 그림 7에서와 같이 해당 링크와 연결된 두 라우터 각각에서 임의로 선택된 포트를 서로 교환하여 토폴로지를 변형시킨다.

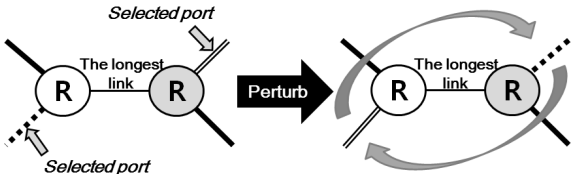


그림 7. 토폴로지 변형의 예
Fig. 7. An example of topology perturbation.

이러한 방법을 쓰는 이유는 전체 토폴로지 중 단지 인접한 두 라우터의 포트를 임의로 교환함으로써 각 경로의 홉 수에 최대한 영향을 주지 않기 위해서이며, 또한 가장 긴 도선 주변의 토폴로지를 변경함으로써 플로어플랜 결과에 영향을 주기 위해서이다.

V. 실 험

제안한 SA 기반 알고리즘으로부터 얻은 토폴로지 및 플로어플랜의 성능을 평가하기 위해 대표적인 하드웨어 기술 언어인 Verilog를 이용한 RTL (register-transfer-level) 시뮬레이션 환경을 구성하였으며 Nangate의 45nm 공정 오픈 셀 라이브러리[19]를 이용해 Synopsys 사의 DesignCompiler로 합성하였다. 합성 후 timing 파일인 표준지연시간(standard delay format, sdf) 파일을 생성하였다. 비동기식 NoC에서 링크 지연 시간의 분포가 성능에 미치는 영향을 보이기 위해 가상 플로어플랜의 결과인 각 링크의 도선 길이를 이용하여 RC chain 도선 모델에서의 링크 지연 시간을 계산하고 이를 sdf 파일의 해당 영역에 반영한다. 반영된 sdf 파일을 테스트벤치에 포함하여 시뮬레이션 함으로써 토폴로지와 더불어 가상 플로어플랜 결과까지 실험에 반영할 수 있다. 이러한 방법은 플로어플랜의 결과 중 링크 지연 시간만을 반영하는 것이므로 실제 물리 설계를 진행했을 때 추가적으로 생성되는 로직들(clock tree buffer, hold buffer 등)에 의하여 발생하는 오차요인이 있을 수 있다. 실제 칩 제작을 통한 검증은 많은 시간과 비용이 소모되어 본 연구에서는 기존의 알고리즘 수준 연구에서와 비슷한 가정을 통해 수행하였고 향후 정밀도를 보다 향상시키는 방안에 대해 후속 연구를 진행하고자 한다. 실험에 사용한 비동기식 라우터의 아키텍처는 앞서 언급하였던 QNoC[12]와 동일하게 구현하였다. QNoC는 기본적인 비동기식 NoC 라우터로 4-위상 bundled 프로토콜을 사용하며 4-입력 뮤텍스(MUTEX)를 사용하여 최대 5개의 포트를 지원할 수 있다. 또한 패킷을 생성하고 받는 코어 모델은 클럭 기반으로 동작하며 1GHz로 설정하였다.

성능 비교를 위해 제안한 알고리즘에서 초기 토폴로지로 설정했던 TopGen을 실험의 비교군으로 사용하였다. TopGen은 일반적인 비정형 토폴로지 생성 알고리즘으로 플로어플랜 결과와 같은 물리적 정보를 고려하지 않은 알고리즘이다. 따라서 제안한 알고리즘으로부터 얻은 토폴로지와 비교 실험을 함으로써 토폴로지 생

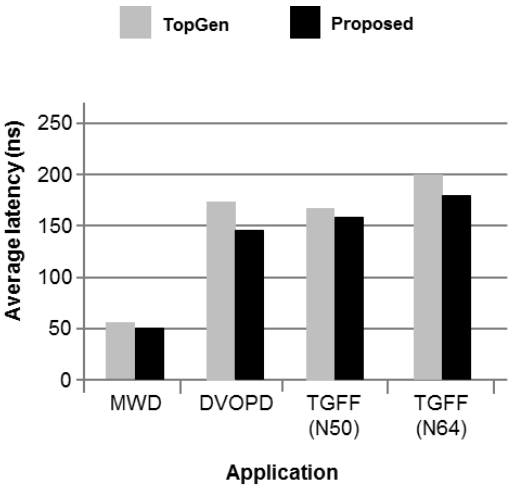
표 1. 각 어플리케이션의 CTG 정보
Table1. Information of CTG.

Benchmark	Number of nodes	Number of edges
MWD	12	12
DVOPD	32	44
TGFF(N50)	50	119
TGFF(N64)	64	187

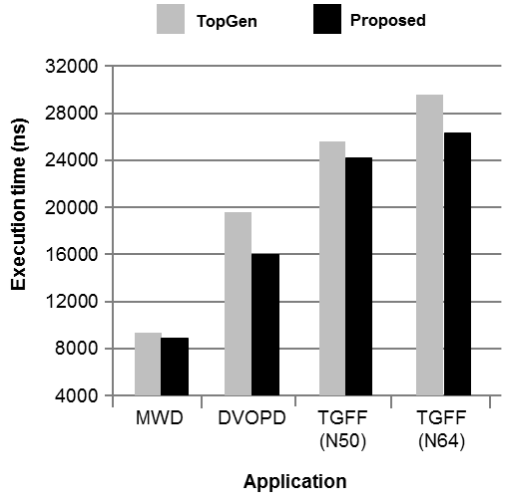
성 시 플로어플랜 결과를 반영하는 것의 효과를 보일 수 있다.

이러한 환경 하에서 벤치마크 어플리케이션인 MWD, DVOPD와 TGFF[20]로 생성한 2개의 어플리케이션들에 대해 실험을 진행하였다. TGFF로 생성한 2개의 어플리케이션은 각각 50개, 64개의 노드를 갖는다. 태스크 간의 트래픽은 각 어플리케이션의 CTG에 기반하여 출력 빈도 및 메시지의 크기를 조절 및 생성할 수 있게 하였다. 표 1은 실험에 사용한 각 어플리케이션의 노드 수와 커뮤니케이션의 수를 나타낸다.

그림 8은 앞서 언급하였던 4개 어플리케이션에 대하여 TopGen과 제안한 알고리즘으로부터 얻은 토폴로지들의 성능 측정 결과를 보여준다. 우선 그림 8-(a)의 평균 지연 시간 결과를 보면 제안한 SA 기반 알고리즘의 토폴로지가 TopGen으로부터 얻은 토폴로지들에 비해 평균 13.7% 가량 감소한 것을 볼 수 있다. 이러한 평균 지연 시간 감소의 결과는 링크 도선 길이를 고려한 비동기식 NoC에서의 지연 시간 모델을 최소화하는 것이 실제 성능에 직접적이고 효과적인 영향을 미쳤다고 해석할 수 있다. 다음으로 그림 8-(b)는 한 번의 입력 패턴을 종료하는데 걸린 실행 시간을 보여준다. 제안한 알고리즘의 토폴로지가 TopGen의 토폴로지보다 평균 11.84% 적게 걸린 것을 볼 수 있다. 이는 제안한 알고리즘이 링크 도선 길이를 최적화함으로써 지연 시간 측면뿐만 아니라 처리량 측면에서도 우수한 토폴로지를 생성하는 것을 나타낸다.



(a) 평균 지연 시간 비교 평가



(b) 실행 시간 비교 평가

그림 8. 어플리케이션에 따른 성능 측정
Fig. 8. Performance evaluation under application.

VI. 결 론

본 논문에서는 비동기식 NoC 설계에서 링크의 도선 길이가 성능에 직접적인 영향을 미치는 점을 이용하여 이를 최적화하기 위한 토폴로지 생성 단계의 지연 시간 모델을 제안하고, SA 기반 알고리즘을 통해 제안한 지연 시간 모델을 최소화하는 토폴로지를 생성하였다.

제안된 지연 시간 모델을 최소화하는 것의 효과를 검증하고자 RTL 수준의 시뮬레이션 환경을 구축하였고, 표준지연시간(sdf) 파일에 가상 플로어플랜 결과를 반영하여 시뮬레이션 하였다. 4개의 어플리케이션에 대하여 일반적인 토폴로지 생성 알고리즘 중 하나인 TopGen과 비교 실험을 한 결과 평균 13.7%의 지연 시간 감소와 평균 11.84% 실행 시간 감소 효과를 볼 수 있었다. 실험 결과를 통해 링크 도선 길이를 고려한 비동기식 NoC에서의 지연 시간 모델이 시스템 설계 단계에서 효과적으로 사용될 수 있음을 확인하였다.

REFERENCES

[1] M. Krstic, E. Grass, F. K. Gurkaynak, and P. Vivet, “Globally Asynchronous, Locally Synchronous Circuits: Overview and Outlook,” IEEE Design & Test of Computers, vol. 24, no. 5, pp. 430-441, Sept.-Oct. 2007.

[2] S. Hauck, “Asynchronous Design Methodologies : An Overview,” Proceedings of the IEEE, vol. 83, no. 1, pp. 69-93, Jan. 1995.

[3] A. Lines, “Asynchronous interconnect for synchronous SoC design,” IEEE Micro, vol. 24, no. 1, pp. 32-41, Jan.-Feb. 2004.

[4] “International Technology Roadmap for Semi-

conductors," Semiconductor Industry Association, 2012.
[5] IBM Research, Introducing a Brain-Inspired Computer. (2014) [Online]. Available: http://www.research.ibm.com/articles/brain-chip.shtml.
[6] M. Imai and T. Yoneda, "Improving Dependability and Performance of Fully Asynchronous On-chip Networks," in Proceeding of IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC), pp. 65-76, Apr. 2011.
[7] O. Naoya, A. Matsumoto, T. Funazaki and T.Hanyu, "High-Throughput Compact Delay-Insensitive Asynchronous NoC Router," IEEE Transactions on Computers, vol. 63, no. 3, pp. 637-649, Mar. 2014.
[8] J. You, D. Gebhardt, and K. S. Stevens, "Bandwidth Optimization in Asynchronous NoCs by Customizing Link Wire Length," in Proceeding of IEEE International Conferfence on Computer Design (ICCD), pp. 455-461, Oct. 2010.
[9] Y. Bei, C. Song, S. Dong, S. Chen, and S. Goto, "Floorplanning and Topology Generation for Application-Specific Network-on-Chip," in Proceedings of Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 535-540, Jan. 2010.
[10] D. Gebhardt, J. You, and K. S. Stevens, "Design of Energy-Efficient Asynchronous NoC and Its Optimization Tools for Heterogeneous SoCs," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 30, no. 9, pp. 1387-1399, Sept. 2011.
[11] S. N. Adya and I. L. Markov, "Fixed-outline Floorplanning : Enabling Hierarchical Design," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 11, no. 6 pp. 1120-1135, Dec. 2003.
[12] Z. Wang, J. Xu, X. Wu, Y. Ye, W. Zhang, M. Nikdast, X. Wang. and Z. Wang, "Floorplan Optimization of Fat-Tree-Based Networks-on-Chip for Chip Multiprocessors," IEEE Transactions on Computers, vol. 63, no. 6, pp. 1446-1459, Jun. 2014.
[13] D. Rostislav, V. Vishnyakov, E. Friedman and R. Ginosar, "An Asynchronous Router for Multiple Service Levels Networks on Chip," in Proceedings of IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC), pp. 44-53, Mar. 2005.
[14] J. J. H. Pontes, M. T. Moreira, F. G. Moraes, and N. L. V. Calazans, "Hermes-AA: A 65nm asynchronous NoC router with adaptive routing," in Proceedings of IEEE International SOC Conference (SOCC), pp. 493-498, Sept. 2010.
[15] D. Gebhardt, J. You, and K. S. Stevens, "Link pipelining strategies for an application-specific asynchronous NoC," in Proceedings of IEEE/ACM International Symposium on Networks on Chip (NoCS), pp. 185-192, May. 2011.
[16] M. E. Dean, T. E. Williams, and D. L. Dill, "Efficient Selftiming with Level Encoded 2-phase Dual-rail (LEDR)," in Proceedings of University of California/Santa Cruz Conference on Advanced research in VLSI, pp. 55-70, Apr. 1991.
[17] Y. Ar, S. Tosun, and H. Kaplan, "TopGen: A new algorithm for automatic topology generation for Network on Chip architectures to reduce power consumption," in Proceedings of International Conference on Application of Information and Communication Technologies (AICT), pp. 1-5, Oct. 2009.
[18] D. Bertsimas and J. Tsitsiklis, "Simulated annealing," Statistical Science, vol. 8, no. 1, pp. 10-15, Feb. 1993.
[19] Silicon Integration Initiative. Nangate open cell library. (2008) [Online], Avaialable: http://www.si2. org/openeda. si2. org/projects/nangatelib, Accessed on: May 2016.
[20] R. P. Dick, D. L. Rhodes, and W. Wolf, "TGFF: Task Graphs for Free," in Proceedings of International Workshop on Hardware/Software Codesign (CODES/CASHE), pp. 97-101, Mar. 1998.

저 자 소 개

김 상 헌(학생회원)

2015년 성균관대학교 전자전기공학과 학사 졸업.

2015년 3월～현재 성균관대학교 IT융합학과 석사과정.

<주관심분야 : SoC 설계, NoC>

이 재 훈(학생회원)

2011년 성균관대학교 반도체시스템공학과 학사 졸업.

2014년 성균관대학교 전자전기컴퓨터공학과 석사 졸업.

2014년 3월～현재 성균관대학교 전자전기컴퓨터공학과 박사과정.

<주관심분야 : SoC 설계, NoC>

이 재 성(학생회원)

2015년 고려대학교 전자 및 정보 공학과 학사 졸업.

2015년 3월～현재 성균관대학교 전자전기컴퓨터공학과 석박사과정.

<주관심분야 : SoC 설계, NoC>

한 태 희(평생회원)

1992년 KAIST 전기 및 전자공학과 학사 졸업.

1994년 KAIST 전기 및 전자공학과 석사 졸업.

1999년 KAIST 전기 및 전자공학과 박사 졸업.

1999년 3월～2006년 8월 삼성 전자 통신 연구소 책임 연구원.

2006년 9월～2008년 2월 한국산업기술대학교 전자공학과 조교수.

2008년 3월～현재 성균관대학교 정보통신대학 반도체시스템공학과 부교수.

2011년 5월～2013년 4월 지식경제부 시스템반도체 PD.

<주관심분야 : SoC 아키텍처 및 설계 방법론, DIC, 메모리/스토리지 시스템 구조, 임베디드 SW, IT융합기술>