논문 2011-48SD-9-6

응용 프로그램의 특성 반영이 가능한 트래픽 생성기

(Application-specific Traffic Generator)

여 필 구*, 조 걸**, 유 대 철**, 황 영 시**, 정 기 석**

(Phil-koo Yeo, Keol Cho, Dae-chul Yu, Young-si Hwang, and Ki-seok Chung)

요 약

다양한 컴포넌트의 집적과 저전력 정책에 대한 연구가 활발했던 시스템 온 칩 설계 분야에서는 최근 들어 집적되는 컴포넌트의 수가 늘어나고 특성이 다양해짐에 따라 이들의 인터커넥션 문제가 새로운 이슈로 주목받고 있다. 시스템 온 칩이 주목받기 시작한 이후로 컴포넌트들의 구성에 따른 성능을 평가하기 위한 각종 시뮬레이터의 개발이 진행되어 왔으며, 효율적인 컴포넌트간의 인터커넥션 설계를 위한 시뮬레이션 환경도 개발이 진행되어 이들을 이용한 성능 평가가 실제 설계에 반영되고 있다. 대부분의 시뮬레이션 환경은 시스템 온 칩의 성능을 테스트하는 데 있어서 수학적 확률 함수를 기반으로 한 트래픽을 사용하고 있으나, 이는 실제 칩의 동작을 테스트하기에는 한계가 있다. 따라서 실질적인 칩의 테스트를 위하여 시스템 상에서의 동작을 정확하게 모사할 수 있는 시뮬레이터의 필요성이 고조되고 있으나, 실제로 이러한 트래픽 생성 방법을 적용한 시뮬레이터는 전무한 실정이다. 이에 본 논문에서는 멀티 프로세서 시스템 온 칩 상에서 수학적 확률 모델은 물론 실제 시스템의 동작을 모사하는 시뮬레이션이 가능한 트래픽 생성 방법을 제안한다. 본 논문에서 제안된 트래픽 생성법은 실제 응용프로그램의 특성을 반영할 수 있도록 트래픽을 생성하므로 수학적 확률 함수를 이용한 트래픽 생성법보다 실제 동작에 가까운 시뮬레이션을 진행할 수 있으며 이는 인터커넥션에 따른 시스템의 성능을 실효적으로 비교할 수 있는 환경을 제공한다. 본 논문에서는 시뮬레이션을 통해 제안된 트래픽 생성법과 수학적 확률 함수를 이용한 트래픽 생성법의 차이를 비교하여 제안된 생성법의 이점에 대해 알아본다.

Abstract

Integrating massive components and low-power policies have been actively investigated for system-on-chip designs. But in recent years, finding the optimal interconnection structure among heterogeneous components has emerged as a critical system design issue. Therefore, various simulation tools to model interconnection designs are being developed and performance evaluation of simulation is reflected in the real design. But most of the simulation environments employ traffic generation based on the mathematical probability functions, and such traffic generation cannot fully cover for various situations that may be occurred in the real system. Therefore, the demand for traffic pattern generation based on real applications is increasing. However, there have been few simulators that adopt application-specific traffic generators. This paper proposes a novel traffic generation method in simulating various interconnection structures for multi-processor system-on-chip design. The proposed traffic generation method can generate traffic patterns that can reflect the actual characteristics of the application and evaluate the performance of an interconnection structure under more realistic circumstance than traffic patterns using mathematical probability functions. By comparing the differences between the proposed method and the one based on mathematical probability functions, this paper shows advantages of the proposed traffic generation method.

Keywords: traffic generator, interconnection, system on chip design, application-specific traffic, simulation

* 정회원, 삼성전자 DMC 연구소
(Design Solution Laboratory, DMC R&D Center, SAMSUNG Electronics)
** 정회원, 한양대학교 전자컴퓨터통신공학과
(Department of Electronics Computer Engineering, Hanyang University)
※ 본 연구보고서는 지식경제부 출연금으로 ETRI 시스템반도체진흥센터에서 수행한 시스템반도체 설계인력양성사업의 연구결과입니다.
접수일자: 2011년3월14일, 수정완료일: 2011년8월17일

Ⅰ. 서 론

반도체 공정 기술이 발달함에 따라 점점 더 많은 수의 프로세서 및 메모리 컴포넌트들을 하나의 칩 안에 집적하는 것이 가능해지면서 많은 임베디드 시스템들이 복합적 기능을 수행할 수 있도록 멀티 프로세서 시스템 온 칩(Multi Processor System-on-Chip, MPSoC)의 형태로 제작되고 있다. 이러한 복합적인 기능에는 기기를 제어하는 응용 프로그램부터 멀티미디어 데이터를 인코딩 및 디코딩하는 기능에 이르기까지 다양하며, 이를 처리하는 프로세서의 종류와 프로세싱 기법 또한 매우 다양하다.

과거 MPSoC 설계의 주요 이슈는 시스템 온 칩(System-on-Chip, SoC) 설계에서와 마찬가지로 집적도를 높여 크기를 줄이고 소모 전력을 낮추어 최소한의 환경에서 동작하게 하는 것이었다. 하지만 최근 MPSoC의 경우 MPSoC를 구성하는 컴포넌트들의 종류가 다양해지고 그 수가 늘어났으며, 각각의 컴포넌트가 요구하는 대역폭이 증가함에 따라 전체적인 시스템의 복잡도가 증가하였다. 따라서 MPSoC를 설계하는데 있어 시스템의 성능을 크게 좌우하는 컴포넌트 간의 인터커넥션(interconnection)[1] 이슈가 과거 MPSoC 설계의 주요 이슈였던 집적도나 저전력 기법보다 더 중요시 되고 있는 추세이다.

이렇듯 최근 MPSoC에서 그 중요성이 부각되고 있는 인터커넥션 설계에는 많은 어려움이 뒤따른다. 첫째로, 오늘날의 MPSoC 설계 방식이 주로 기존에 존재하는 IP(Intellectual Property)들을 재사용하여 구성하는 방식으로 행해짐으로써 IP들에 대해 구체적으로 알 수 없는 경우가 종종 있다는 것을 들 수 있다. 재사용되는 IP는 대부분 표준 인터페이스(interface)에 맞게 서로 다른 벤더에서 만들어 진다. 따라서 인터페이스를 제외한 내부 동작 방식 등의 IP 상세 정보를 알 수 없는 경우가 많아, 설계 시에 항상 최악의 경우(worst-case)를 고려하여야 하므로 불필요한 경우까지 고려해야 한다는 문제점이 있다. 둘째로, 인터커넥션의 설계의 목적이 대역폭 요구사항, 지연(latency) 요구사항, 면적의 최소화, 전력 소모량의 최소화와 같이 여러 가지 요구사항을 만족시켜야 한다는 점이다[1]. 뿐만 아니라 MPSoC를 구성할 수 있는 변수들이 매우 다양하기 때문에 시뮬레이션 도구를 이용하여 해당 MPSoC의 성능 및 대역폭, 지연시간 등의 요구사항을 만족하는지 검증을 하고 난 뒤 칩 제작을 진행하여야 한다는 점 역시 인터커넥션 설계가 어려운 이유 중 하나이다.

칩을 제작하기에 앞서 인터커넥션 설계를 검증하기 위해서는 고 신뢰성의 시뮬레이션 환경과 시뮬레이션에 사용할 테스트 트래픽이 필요하다. 현재 MPSoC를 위한 여러 가지 시뮬레이터들이 존재하지만[1~3, 8], [1~2]의 경우 NoC(Network-on-Chip) 시스템을 위한 시뮬레이터로 현 멀티코어 및 임베디드 시스템에는 그 결과를 반영하기 어려운 실정이다. 또다른 인터커넥트 성능 평과 관련 연구인 INTACTE[8] 시뮬레이션 툴의 경우 아키텍처를 구성하는 파라미터(인터커넥트의 길이, 폭, 동작 주파수, 지연시간)들을 설정하여 해당 아키텍처의 면적, 지연시간, 전력등을 예측하는 기능을 제공하나, 이 연구는 동작의 검증을 지원하지 않고, 구현된 인터커넥트 구조의 물리적 특성을 예측하는데 그친다는 단점이 있다. OCP-IP의 Transaction Generator[3] 연구의 경우 NoC 외에 다양한 인터커넥트 구조에 적용 가능한 시뮬레이터로 시뮬레이터들이 MPSoC 인터커넥션의 성능을 테스트하기 위해 사용하는 대부분의 트래픽은 수학적 확률 모델을 기반으로 생성되어 성능 평가에 사용된다. 하지만 이러한 트래픽으로 시뮬레이션을 진행할 경우 실제 시스템에서의 트래픽을 재현하기에 한계가 있어 시뮬레이션의 정확도가 낮다는 단점이 있다.

따라서 정확한 인터커넥션 시뮬레이션을 위해 응용 프로그램의 특성 반영이 가능한 트래픽의 필요성이 부각되고 있으나, 실제 이러한 트래픽 생성이 가능한 트래픽 생성기는 존재하지 않는 실정이다. 따라서 본 논문에서는 보다 정확한 MPSoC 인터커넥션의 성능평가를 위하여 시뮬레이션에 사용되는 트래픽 생성기를 실제 응용프로그램의 동작을 분석하여 데이터베이스화한 후 이를 기반으로 트래픽을 생성하도록 함으로써 수학적 확률 모델을 이용한 트래픽보다 더 실질적이고 정확한 트래픽 모델링 방법을 제안한다.

본 논문은 다음의 순서로 구성되어있다. Ⅱ장에서는 제안하는 트래픽 모델링을 설명하기에 앞서 두 가지 종류의 MPSoC 시뮬레이터와 본 논문에서 트래픽 모델링에 사용한 두 가지 응용 프로그램의 특성에 대해 설명한다. Ⅲ장에서는 이들 응용 프로그램의 특성을 반영한 트래픽 모델링 기법에 대한 설명과, 수학적 확률 모델링 기법에 대한 차이를 설명하고, 이를 입증하기 위해 각각의 기법으로 생성한 트래픽으로 실험한 내용

과 결과를 Ⅳ장을 통해 서술한다. 마지막으로 Ⅴ장의 결론 및 향후과제를 통해 본 논문을 마무리 한다.

Ⅱ. 관련 연구

1. Atlas NoC 시뮬레이터

Atlas NoC Simulator[2]는 GAPH(Grupo de Apoio ao Projeto de Hardware, Hardware Design Support Group) 그룹에 의해 제안된 NoC 모델을 사용하여 NoC 생성, 트래픽 생성, NoC 시뮬레이션, 트래픽 성능평가를 진행한다. 사용자 편의성을 위해 그래픽 사용자 인터페이스(Graphic user interpace; GUI)를 제공하며, NoC 환경의 설정과 NoC의 각 컴포넌트 사이에 발생하는 트래픽 설정이 가능하다. 시뮬레이션은 프로젝트 단위로 진행되며, NoC 생성 시 2D Mesh 형태의 NoC를 구성하고 라우터의 동작을 설정하여 전체적인 NoC의 동작을 시뮬레이션 한다. 라우터의 설정은 흐름제어, 가상채널의 수, 스케쥴링 방식, 2D Mesh 구조에서의 컴포넌트 수, 전송되는 플릿(flit)의 크기, 버퍼의 크기 등을 설정할 수 있으며, 이는 프로젝트 생성 시 결정한 NoC 종류에 따라 설정 내용이 조금씩 달라질 수 있다. 설정에 맞게 생성된 NoC 모델은 VHDL(Verilog Hardware Description Language)로 기술되어 있으며, 사용자가 설정한 내용을 기반으로 SystemC 언어를 사용한 테스트벤치(testbench)를 생성할 수 있다.

Atlas NoC Simulator에서 컴포넌트 별로 설정해줄 수 있는 트래픽은 수학적 확률 모델을 기반으로 하고 있다. Atlas NoC Simulator에서 사용하는 확률 모델은 균등분포(uniform distribution), 정규분포(normal distribution), 파레토분포(pareto distribution) 등이 있으며, 각 모델별로 고유의 설정 값들을 정해 시뮬레이션에 사용할 수 있다. 이렇게 설정된 값을 기반으로 일괄 생성된 트래픽은 컴포넌트 별로 할당된 입력(input) 텍스트 파일에 트래픽의 발생 시점, 크기, 목적 컴포넌트 번호 등의 정보 형태로 저장된다. 그리고 VHDL 언어로 작성된 NoC 환경과 텍스트 파일에 저장된 트래픽 정보를 ModelSim을 이용하여 시뮬레이션하고 그 결과를 각 컴포넌트에서 처리한 트래픽 량, 트래픽 발생시간과 처리 완료 시간 및 지연시간 등의 정보 형태로 출력(output) 텍스트 파일에 저장한다. 최종적으로 일련의 과정을 거쳐 생성된 입력 및 출력 텍스트 파일들의 정보를 취합하여 NoC 환경의 성능 평가를 수행하는 구조로 되어 있다.

2. OCP-IP 트랜젝션 제너레이터 패키지

OCP-IP(Open Core Protocol International Partnership)에서 개발된 Transaction Generator 2(TG2)[3]는 MPSoC에서 사용된 NoC 인터커넥션의 벤치마킹을 위한 트랜잭션 수준(transaction level)의 SystemC 언어 기반의 시뮬레이터이다. 트랜잭션 수준 모델링의 경우 RTL(Register Transfer Level) 모델링보다 그 추상화의 정도가 높기 때문에 시스템의 동작을 모델링하기 쉽고 시뮬레이션이 빠르다는 장점이 있어 MPSoC의 동작이나 대역폭 보장 여부를 판단하는 검증에 사용되고 있다. OCP-IP의 TG2는 XML(eXtensible MarkUp Language)를 이용하여 트래픽의 입력 설정, 트래픽의 패턴, 테스트할 플랫폼의 모델을 설정할 수 있으며, 이들을 시뮬레이션 할 수 있게 맵핑(mapping)하도록 설계되어 있다.

응용 프로그램 수준(application level)의 추상화에서는 태스크(task)들의 동작을 설정할 수 있으며, 맵핑 수준의 추상화에서는 위의 태스크가 어떠한 프로세싱 엘리먼트(Processing Element; PE)에서 실행될지 설정할 수 있다. 이에 따라 PE 수준의 추상화에서는 네트워크 모델에 인가할 트래픽에 대한 설정을 XML을 이용하여 적용할 수 있다. 시뮬레이션에 사용되는 네트워크 모델은 RTL이나 그보다 더 높은 수준의 모델을 적용할 수 있으며, 시뮬레이션을 위해 Atlas와 마찬가지로 ModelSim을 사용한다. OCP-IP의 TG2에서 테스트에 사용하는 트래픽은 크게 다항식(Polynomial) 모델, 균등분포, 정규분포 세 가지를 지원한다. 다항식 모델의 경우 수신한 바이트 수를 x값으로 두어 생성할 트래픽을 다항식으로 설정하여 사용한다.

3. H.264/AVC 디코더와 LDPC 디코더

H.264 혹은 MPEG-4 AVC는 ISO/IEC와 ITU가 함께 만든 비디오 압축 표준[4~5]으로 기존에 사용되는 비디오 압축 표준보다 더 높은 압축률을 가지며, 네트워크를 통한 스트리밍에 적합한 특성을 가지고 있어서 다양한 멀티미디어 응용 분야에 많이 쓰이고 있다. H.264/AVC의 비디오 시퀀스는 가장 큰 데이터 단위인 Group Of Pictures (GOP)로 구성되며, 각 GOP는 여러 개의 프레임(Frame; 또는 픽쳐(picture))으로 구성된다. 하나의 프레임은 다시 하나 또는 여러 개의 슬라이스

(slice)로, 슬라이스는 여러 개의 매크로 블록(macroblock)으로 구성된다. 매크로 블록은 응용 프로그램에서 연산의 단위가 되며, 매크로 블록은 다시 휘도 블록(luma block)과 색차 블록(chroma block)으로 나누어진다. 한 개의 매크로 블록은 여러 단계를 순차적으로 거쳐서 연산이 완료되고, 이렇게 연산이 완료된 다수의 매크로 블록이 복원된 한 프레임의 영상을 이루게 된다.

이와 같이 세분화된 데이터 단위와 명확히 구분된 순차 처리 방식을 갖는 H.264/AVC 디코더의 특성상, 다양한 병렬화 연구가 진행되어지고 있다. 대표적인 H.264/AVC 디코더 병렬화 방법으로는 태스크 수준(task level)의 병렬화 방법과 데이터 수준(data level)의 병렬화 방법이 있다. 태스크 수준의 병렬화 방법은 H.264/AVC 디코더를 각각의 처리 단계별로 나누고, 이를 스레드(thread)에 할당하여 실행함으로써 성능을 높이는 병렬화 방법이다. 이 방법의 주요 제한점은 디코딩 과정에서 각 기능별로 데이터를 처리하는 시간이 다르다는 것이다. 따라서 각 단계에서 가장 긴 처리 시간을 갖는 데이터 처리 시간이 해당 단계의 처리 시간이 되며 전체적인 처리 시간은 증가 하게 된다. 데이터 수준의 병렬화 방법은 H.264/AVC 디코더를 데이터 단위로 나누어 여러 스레드에서 동시에 처리하는 방법이다. 앞서 설명한 H.264/AVC 데이터 단위에 따라 프레임 단위, 슬라이스 단위, 매크로 블록 단위 병렬화 방법으로 나누어지며, 최근에는 매크로 블록 단위의 H.264/AVC 병렬 디코더에 대한 다양한 연구가 이루어지고 있다. 매크로 블록 단위의 데이터 수준 병렬화 방법은 동시에 처리 할 수 있는 매크로 블록을 각 스레드에 할당하여 병렬화 한다. 하지만 이 방식의 가장 큰 문제점은 매크로 블록 사이의 의존성이 존재 하게 된다는 것이다(그림 1).

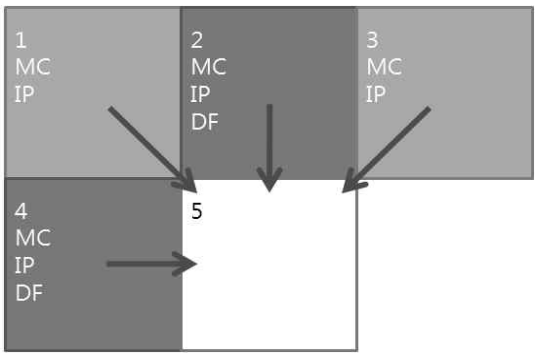


그림 1. 매크로 블록의 의존성
Fig. 1. Dependency of macro block.

따라서 매크로 블록 단위의 데이터 수준 병렬화 방법은 이와 같은 매크로 블록간 의존성의 효율적인 처리 방법에 따라 성능이 크게 좌우된다.

Low Density Parity Check(LDPC) 부호를 이용한 디코딩 방식은 1962년 Gallager에 의해 제안[6]된 방식으로, 현대의 이동통신 시스템에서 오류정정 부호로 주목받고 있다. 반복 디코딩 방식을 적용한 LDPC 디코더는 복잡도가 크지 않고, 높은 수준의 병렬 처리가 가능하여 디코딩 속도가 빠르다는 장점이 있어 하드웨어로 구현하기에 매우 적합하다. LDPC (N, M)부호는 M × N 차수의 패리티 검사 행렬 H를 이용하여 오류를 검출하고 정정한다. 수신된 신호와 패리티 검사 행렬 H를 곱한 값이 0이면 수신 신호에 오류가 없음을 의미하고, 1이면 수신 신호의 일부에 오류가 발생한 것으로 간주하여 오류 정정 프로세스를 수행한다. 패리티 검사 행렬 H에서 행이 갖는 1의 원소 개수를 행 무게(Wr; row weight), 열이 갖는 1의 원소 개수를 열 무게(Wc; column weight)라 한다. 만약 패리티 검사 행렬 H의 모든 열이 갖는 열 무게가 같고, 모든 행이 갖는 행 무게 Wr = Wc * (N / M)을 만족할 경우를 균일 LDPC 부호라 하며 그렇지 않은 경우를 비 균일 LDPC 부호라 한다. 일반적으로 균일 LDPC 부호에 비해 비 균일 LDPC 부호가 더 좋은 디코딩 성능을 보여주는 반면, 하드웨어 구현이 더 어렵다는 특징이 있다.

패리티 검사 행렬 H는 행을 체크 노드(check node)에, 열을 비트 노드(bit node)에, 그리고 행렬의 1 값을 해당 체크 노드와 비트 노드를 연결하는 에지(edge)에 대응시킨 태너 그래프(Tanner graph)로 표현이 가능하다(그림 2).

LDPC 부호는 태너 그래프 상에서 에지로 연결된 체크 노드와 비트 노드 사이에 확률 값을 가진 메시지를 반복적으로 전달하면서 신뢰성 있는 확률 값을 갖도록 하는 전달 알고리즘을 이용하여 디코딩을 수행한다. 디

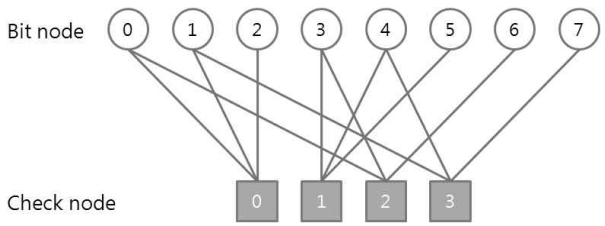


그림 2. 패리티 검사 행렬 H(8, 4)의 태너 그래프
Fig. 2. Tanner graph of H matrix (8,4).

코딩된 값을 검사하여 에러가 발견되면, 디코딩 과정을 반복하여 올바른 값이 될 때까지 디코딩을 진행하거나 미리 정의된 최대 반복횟수가 지나면 값을 버리게 된다. 이러한 메시지를 전달하는데 사용하는 알고리즘에는 심볼의 확률 값을 사용하는 방법, 우도 신뢰 전파(Log Likelihood Ratio-Belief Propagation; LLR-BP) 알고리즘, 근사화 신뢰 전파(Uniformly Most Probable-Belief Propagation; UMP-BP) 알고리즘 등이 있다. LLR-BP 알고리즘은 심볼의 확률 값을 사용하는 방법에 비해, 디코딩 과정에서 실제 확률 값의 곱셈 연산을 덧셈연산으로 대체할 수 있어 실제 확률 값을 사용하는 방법보다 효율적인 반면, 체크 노드를 계산하는데 있어 높은 복잡도를 요구하는 단점이 있다. UMP-MP 알고리즘은 LLR-BP 알고리즘의 복잡도를 줄이기 위해 제안된 알고리즘으로 약간의 성능 저하를 보이는 반면 복잡도가 상당수 줄어들고 채널 추정을 요하지 않는다는 장점이 있다.

Ⅲ. 트래픽 모델링

본 논문에서의 트래픽은 단순한 비트의 집합으로 컴포넌트에서 컴포넌트로의 이동에 필요한 정보만 가지고 있으며, 목적 컴포넌트에 도착한 트래픽이 컴포넌트 내에서 별도의 처리 과정을 거치는 것은 고려하지 않았다. 따라서 트래픽은 최소한의 정보만을 포함하며, 나머지 부분의 트래픽은 크기에 맞게 의미 없는 더미 데이터(dummy data)로 채워지게 된다. 이때의 최소 정보를 로우 데이터(raw data)라 하며, 로우 데이터로부터 트래픽을 생성하는 것을 트래픽 모델링(modeling)이라 한다. 이 장에서는 앞서 알아본 두 가지 응용 프로그램의 트래픽 모델링 방법과 그 결과를 알아본다.

1. H.264/AVC 디코더의 트래픽 모델링

H.264/AVC 디코더는 영상의 한 프레임을 여러 개의 매크로 블록으로 나누어 여러 단계를 순차적으로 거쳐 디코딩 하는 방식으로 되어 있다. 본 논문에서 사용한 H.264/AVC 디코더는 총 세 단계의 디코딩 과정을 거치며, 각 디코딩 과정에 한 개의 태스크를 할당하여 처리하는 태스크 수준의 병렬화 방식으로 구현되어 있다. 이렇게 태스크 수준의 병렬화 방식으로 처리하는 H.264/AVC 디코더는 다음의 두 가지 특성을 지닌다. 첫째, 각 태스크는 태스크 수행과 동시에 필요한 매크로 블록 데이터를 메모리로부터 읽어 들이고, 태스크 내부의 처리과정을 거쳐, 태스크 종료 직전에 연산이 완료된 매크로 블록 데이터를 메모리에 쓰는 구조로 되어 있다. 즉, 태스크에서 한 개의 매크로 블록이 처리되기 위해서는, 메모리에서 태스크로의 데이터 이동과 태스크에서 메모리로의 데이터 이동이 각각 한 번(정확히는 2개의 블록-매크로 블록과 매크로 블록을 처리하기 위한 정보를 갖는 블록-이 순차적으로 이동)씩 나타나게 되는 구조이다. 둘째, 각 태스크는 우선순위(priority) 및 의존성(dependency)이 존재한다. 다시 말해 동일 매크로 블록을 처리함에 있어 우선순위가 높은 태스크의 연산이 완료 될 때까지 다음 우선순위의 태스크는 대기 상태에 빠지게 된다는 것이다. 따라서 두 번째 태스크는 첫 번째 태스크보다 한 단계 늦게 수행되며, 세 번째 태스크 역시 두 번째 태스크보다 한 단계 늦게 수행되는 파이프라인(pipe line)식 구조를 갖게 된다.

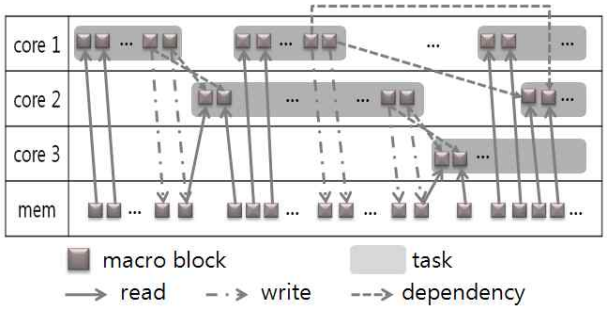


그림 3. H.264/AVC 디코더의 패턴 분석
Fig. 3. Pattern analysis of H.264/AVC decoder.

그림 3은 이러한 H.264/AVC 디코더의 트래픽을 분석한 결과를, 그림 4는 각 컴포넌트에서 처리한 매크로 블록 데이터의 단위시간당 개수를 표현한 것이다.

그림 4를 통해 알 수 있는 사실은 각 코어 컴포넌트별로 서로 다른 트래픽 처리량을 보여주는 반면 트래픽

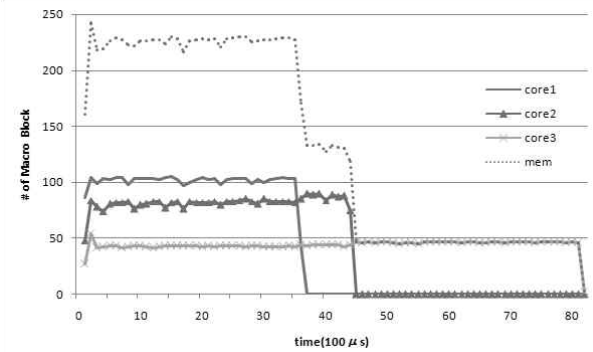


그림 4. H.264/AVC 디코더의 트래픽 처리량
Fig. 4. Traffic throughput of H.264/AVC decoder.

을 처리하는 패턴은 매우 균일하다는 것이다. 이와 같이 균일한 패턴은 후에 논할 수학적 확률 함수 모델링이 갖는 대표적인 특징 중 하나이다.

2. LDPC 디코더의 트래픽 모델링

본 논문에 사용한 LDPC 디코더는 CMMB (China Mobile Multimedia Broad-casting) 표준[7]을 기반으로 하며, 6 × 3 행렬의 패리티 검사 행렬 H와 비 균일 LDPC 부호를 사용하였다. 또한 디코더 구현을 위해 LLR-BP 알고리즘과 4의 병렬화 지수 값을 갖는 부분 병렬 구조를 사용하였다. 이러한 LDPC 디코더가 갖는 특성은 다음과 같다. 첫째, 다수의 코어 컴포넌트에서 수행하는 태스크들의 동작 방식이 모두 동일하다. LDPC의 병렬화 처리 방식은 매우 많은 양의 반복이 발생하는 루프 구문을 병렬화 지수만큼 분할하고 각 스레드에 동일하게 분배하여 처리하는 방식이다. 즉 병렬화 하지 않고 처리할 때에 발생하는 모든 트래픽을 병렬화 지수만큼의 스레드가 나눠서 처리하는 방식이다. 따라서 모든 스레드의 트래픽 패턴이 매우 비슷한 형태를 보이게 된다. 둘째, 연산의 단위가 되는 유닛의 크기가 매우 작으며, 이들 사이에는 의존성이 존재하지 않는다(그림 5). 한 개의 태스크가 처리되는 과정에서 발생하는 트래픽의 개수는 패리티 검사 행렬 H의 크기에 비례하지만, 많게는 수십에서 수백 번의 읽기 연산과 그와 비슷한 수의 쓰기 연산이 발생(본 논문에서는 각각 10회 전후)할 뿐이며, 이때 연산 유닛의 크기는 정수형 변수 한 개 또는 실수형 변수 한 개의 크기일 뿐이다. 또한 병렬 처리 구간에서 각 태스크가 공유하는 메모리 영역에 대해 읽기 연산과 쓰기 연산이 동시에 수행되는 영역이 존재하지 않으며, 서로 다른 태스크에서 같은 메모리 영역에 쓰기 연산을 수행하지 않는다. 따라서 메모리를 동시 접근 하려는 문제에 대한 의존성을 제외한 데이터 사이의 의존성은 존재하지 않을 뿐더러, 매우 작은 크기의 연산 유닛들이 매우 빠른 속도로 수행되므로, 메모리 동시 접근의 의존성 때문에 발생하는 지연 시간도 무시할 수 있는 수준의 것이다. 그림 6을 통해 알 수 있듯이, 각 코어 컴포넌트에서 처리하는 트래픽 처리량은 모든 코어 컴포넌트가 매우 유사한 반면, 각 코어 컴포넌트의 트래픽 처리 패턴은 매우 불규칙한 변화를 보인다는 것이다. 이는 H.264/AVC 디코더와는 완전히 상반된 특징이라 할 수 있다.

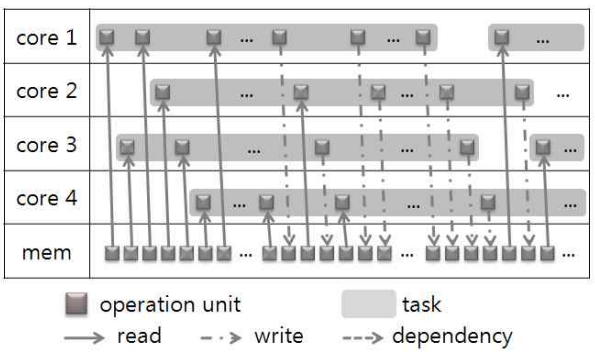


그림 5. LDPC 디코더의 패턴 분석
Fig. 5. Pattern analysis of LDPC decoder.

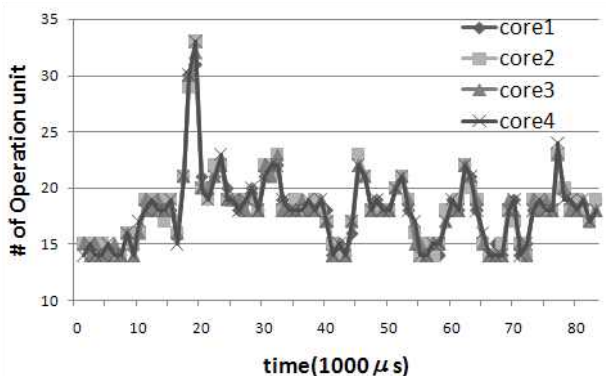


그림 6. LDPC 디코더의 트래픽 처리량
Fig. 6. Traffic throughput of LDPC decoder.

3. 기존 트래픽 모델링과의 차이

앞서 1절에서 보았던 H.264/AVC 디코더의 트래픽 분포 그래프는 수학적 확률 함수를 이용하여 어느 정도 모사가 가능하다. 예를 들어 코어 컴포넌트 1에서 처리하는 매크로 블록의 분포값은 대략 0 ~ 3700μs 구간동안 100개를 평균으로 하고 대부분이 표준편차 10개 내

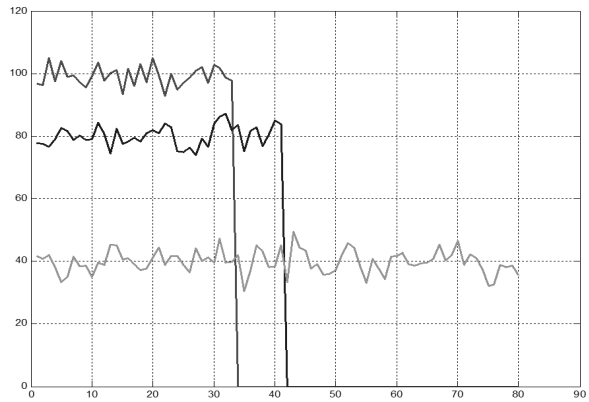


그림 7. 정규 분포 함수를 이용한 트래픽 모사
Fig. 7. Traffic simulation using normal distribution function.

에 존재한다. 코어 컴포넌트 2와 3에서 처리하는 매크로 블록의 분포값 역시 이와 같은 방식으로 분석 할 수 있다(단지 메모리 컴포넌트에서 발생하는 트래픽 패턴은 세 개의 코어 컴포넌트에서 발생한 트래픽의 합과 일치하여, 단일 수학적 확률 함수로는 모사할 수 없기에 이는 제외한다). 이를 기반으로 각 코어 컴포넌트에서 발생하는 트래픽을 정규 분포 함수로 모사한 것이 그림 7이다. 그림 4와 그림 7을 단순 비교(메모리 컴포넌트의 트래픽은 제외)하면, 결과적으로 패턴을 매우 유사하게 생성했다고 볼 수 있다. 하지만 H.264/AVC 디코더의 트래픽 패턴을 수학적 확률 함수로 모사하기 위해서 사용한 데이터는 다름 아닌 H.264/AVC 디코더를 분석한 데이터라는 사실을 상기할 필요가 있다. 즉, 수학적 확률 함수로 응용 프로그램의 특성을 모사 할 수 있다고 해도 어디까지나 응용 프로그램의 특성을 미리 분석하여 알고 있었기 때문에 가능했다는 것이다.

더욱이 H.264/AVC 디코더의 메모리 컴포넌트 트래픽 패턴처럼 특성을 알고 있다고 해도 수학적 확률 함수로는 모사할 수 없는 경우도 존재한다. 2절에서 설명한 LDPC 디코더의 경우 또한 수학적 확률 함수로는 모델링하기 어려우며, 모델링을 하더라도 그 오차가 매우 클 가능성이 있다.

Ⅳ. 실 험 결 과

1. 실험 환경

본 논문에서 제안하는 응용 프로그램의 특성 반영이 가능한 트래픽을 검증하기 위하여 시뮬레이션을 수행하였다. PC 환경에서 본 논문에 맞게 수정한 Atlas NoC 시뮬레이터를 이용하여 앞서 제시한 두 가지 응용 프로그램을 모사한 트래픽과 수학적 확률 함수 기반의 트래픽을 비교하였다.

실험은 각각의 트래픽을 생성 순서와 발생 시간을 제외한 나머지 인자(트래픽의 크기, 시작 / 목적 컴포넌트, 발생 트래픽의 개수 등)가 모두 동일한 조건에서 수행 하였다. 총 아홉 개의 컴포넌트 중 모서리에 존재하는 네 개의 컴포넌트는 사용하지 않으며, 좌측 중간 컴포넌트를 메모리 컴포넌트로, 그리고 세 개의 컴포넌트(중간 하단, 중간 상단, 우측 중간)를 코어(CPU) 컴포넌트로 사용하였다. 네 개의 미사용 컴포넌트로의 트래픽 이동은 없으므로 코어 컴포넌트와 메모리 컴포넌트는 중앙의 컴포넌트를 통해 이동하게 된다. 여기에서 중앙

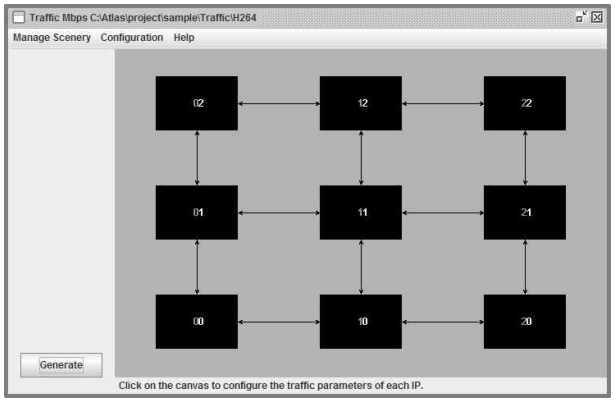


그림 8. 시뮬레이션 환경
Fig. 8. Environment of simulation.

컴포넌트는 다수의 코어 컴포넌트에서 발생하는 트래픽들을 메모리 컴포넌트에 하나씩 순서대로 전달하거나 그 반대의 역할을 수행하므로 일종의 간단한 구조의 버스라 볼 수 있다. 따라서 이와 같은 3 × 3구조는 세 개의 프로세서가 버스를 통해 단일 포트(혹은 동시 입출력이 가능한 듀얼 포트) 메모리를 공유해서 사용하는 구조(공유 버스 구조; shared Bus)로 해석할 수 있다(그림 8).

시뮬레이션의 결과는 본 논문에서 제안하는 트래픽과 수학적 확률 함수 기반 트래픽의 생성 순서 차이, 그리고 그로 인하여 생기는 트래픽 처리의 지연 시간으로 표현되며, 이들의 비교를 통해 본 논문의 타당성을 입증 하고자 한다.

2. 실험 결과

앞서 설명했던 바와 같이 H.264/AVC 디코더 각각의 코어 컴포넌트에서 보이는 트래픽의 패턴은 수학적 확률 함수인 정규 분포 함수를 통해 유사한 양상을 보이는 트래픽을 모사할 수 있다. 따라서 정확히 일치하지는 않더라도 시뮬레이션 결과가 비슷하게 나온다면, 실제 응용 프로그램의 특성을 반영하는데 있어서 기존에 사용하던 수학적 확률 모델링 기법이 유효함을 입증할 수 있다. 반면에 전혀 다른 시뮬레이션 결과가 나온다면, 기존의 수학적 확률 모델링 기법의 한계를 반증할 수 있을 것이다. 다음의 표 1은 실제 H.264/AVC 디코더 동작을 본 논문에서 제안한 모델링 기법과 수학적 확률 함수 모델링 기법을 이용하여 추출한 각각의 로우 데이터의 일부를 비교한 표이다. 또한 그림 9는 이들 로우데이터를 시뮬레이션 하였을 때 코어 컴포넌트 1에서 처리한 트래픽의 지연시간을 비교한 것이다.

표 1. H.264/AVC 디코더의 트래픽 생성
Table 1. Traffic generation of H.264/AVC decoder.

	제안 모델링		확률 모델링	
번호	컴포넌트	발생시간	컴포넌트	발생시간
1	0010	0	0010	0
2	0012	15	0012	0
3	0010	17	0021	0
4	0010	19	0010	10
5	0010	20	0012	12
6	0010	21	0010	21
7	0010	22	0012	24
8	0010	23	0021	25
9	0010	24	0010	32
10	0010	25	0012	37
...	...	...	...	...

표 1을 통해 알 수 있는 사실은 두 종류의 모델링 방식에서 생성한 트래픽의 생성 순서가 매우 다르다는 점이다. H.264/AVC 디코더의 경우 처음 10개 데이터의 태스크별 비율이 9 : 1 : 0(순서대로 태스크 1, 2, 3)으로 태스크 1의 트래픽이 대부분을 차지하는 반면 수학적 확률 함수 기반의 트래픽의 경우는 4 : 4 : 2의 비율로 비교적 고른 분포를 보여준다. 또한 10번째 트래픽의 발생 시간을 비교해 보면 H.264/AVC 디코더의 트래픽은 25μs이고 수학적 확률 함수 기반의 트래픽은 37μs로 큰 차이를 보인다. 다시 말해, H.264/AVC 디코더의 트래픽은 평균 2.5μs 마다 생성되는 반면, 수학적 확률 함수 기반의 트래픽은 평균 3.7μs마다 생성된다는 것이다. 또한 그림 9를 통해서 이렇게 서로 다른 비율로 발생한 트래픽을 처리하는데 지연시간의 결과가 얼마나 달라지는지를 알 수 있다.

위와 같은 결과는 LDPC 디코더의 경우에도 그대로 적용되었다. 다음의 표 2는 LDPC의 동작을 본 논문에서 제안한 기법을 이용하여 추출한 로우데이터의 일부

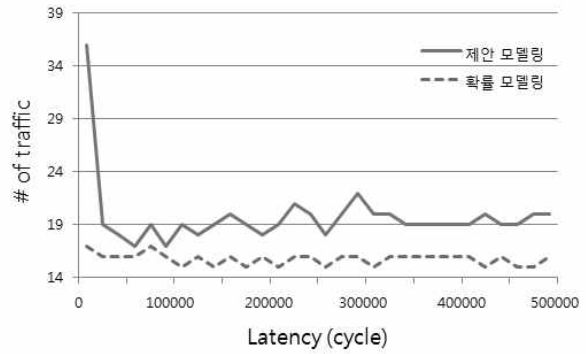


그림 9. 시뮬레이션 결과(컴포넌트 1)
Fig. 9. Simulation result(component 1).

표 2. LDPC 디코더의 트래픽 생성
Table 2. Traffic generation of LDPC decoder.

	제안 모델링	
번호	컴포넌트	발생시간
1	0010	0
2	0010	3
3	0012	6
4	0010	7
5	0021	8
6	0010	11
7	0012	12
8	0010	16
9	0012	18
10	0010	20
...	...	...

이다. 만약 LDPC 디코더를 수학적 확률 모델링 기법으로 모사할 수 있다고 하면, 각 컴포넌트에서 처리할 트래픽의 비율이 앞선 실험과 마찬가지로 비교적 고르게 나올 것이다. 하지만 실제로는 표 2에서와 같이 트래픽의 비율이 6 : 3 : 1 : 0으로 서로 비슷하지 않음을 알 수 있다.

이 결과를 통해 실제 응용 프로그램의 특성 반영 여부가 시뮬레이션 결과에 미치는 영향을 알 수 있으며, 수학적 확률 함수를 이용한 기존의 모델링 방식으로는 비록 모사가 가능할지라도 정확한 결과를 얻을 수 없다는 것을 알 수 있다.

V. 결론 및 향후과제

본 논문에서는 기존의 트래픽 모델링 기법이 갖는 단점을 보안하기 위해 응용 프로그램 특성 반영이 가능한 트래픽 모델링 기법을 제안하였다. 두 가지 실제 응용 프로그램의 특성을 분석하고, 이들의 특성을 나타낼 수 있는 로우 데이터를 추출하여 트래픽을 생성하는 방법론을 제시하였으며, 이렇게 생성된 응용 프로그램의 특성을 반영한 트래픽과 이 트래픽을 수학적 확률 함수로 모사한 트래픽을 시뮬레이션을 통해 비교하였다. 결과를 통해 일부 응용 프로그램의 특성이 반영된 트래픽은 수학적 확률 함수를 이용하여 유사 패턴을 생성할 수는 있으나, 실제 트래픽과 유사 트래픽의 생성 데이터와 시뮬레이션의 결과는 상당히 다른 양상을 보여주었고, 응용 프로그램 중에서는 수학적 확률 함수로 모사가 불가능한 경우가 있다는 것도 보였다. 이러한 과정을 통해 인터커넥션 이슈가 부각되고 있는 MPSoC 설계 시

에 시스템에 적합한 인터커넥션 방식을 정확히 선별 할 수 있는 한 가지의 방법론을 제시할 수 있었다.

본 논문에서는 위와 같은 결론을 도출하기 위해 시뮬레이션 환경에서 제안한 방법으로 생성한 트래픽을 사용하여 실험을 진행 하였다. 향후 연구 목표로는 현재 논문에 제시되어있는 H.264/AVC 디코더와 LDPC 디코더뿐만 아니라 실제 MPSoC 환경에서 사용되는 여러 가지 추가적인 응용 프로그램의 특성을 반영한 트래픽 생성기를 제작하여 시뮬레이션에 적용함으로써 MPSoC 설계 시에 적합한 인터커넥션을 선별하는 것을 목표로 한다.

참 고 문 헌

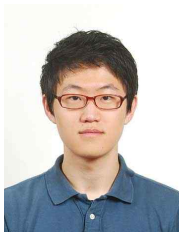
[1] NoC Simulator, http://nocsim.blogspot.com/
[2] ATLAS NoC simulator, http://www.inf.pucrs .br /~gaph/AtlasHtml/AtlasIndex_us.html
[3] OCP-IP Transaction Generator 2, http://www. ocpip.org/tg_package.php
[4] Thomas Wiegand, Gary J. Sullivan, Gisle Bjøntegaard, and Ajay Luthra, Senior Member, “Overview of the H.264/AVC Video Coding Standard”, IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 560-576, July 2003.
[5] Michael Horowitz, Anthony Joch, Faouzi Kossentini, and Antti Hallapuro, “H.264/AVC Baseline Profile Decoder Complexity Analysis,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 704-716 July 2003.
[6] R. G. Gallager, “Low density parity check codes”, IRE Trans. Inform. Theory, Vol. IT-8, Jan. 1962, pp. 21-28.
[7] Interfax China (2006-10-25). China releases mobile TV industrial standard. Retrieved on 2007-04-14
[8] Nagpal, R.Madan, A.Bhardwaj, A.Srikant, YN, INTACTE: An Interconnect Area, Delay, and Energy Estimation Tool for Microarchitectural Explorations, CASES'07, Sep. 30 - Oct. 3, 2007, Salzburg, Austria.

저 자 소 개

여 필 구(정회원)
2009년 한양대학교 컴퓨터전공 학사 졸업.
2011년 한양대학교 전자컴퓨터 공학과 석사 졸업.
현재 삼성전자 DMC 연구소 사원.
<주관심분야: 임베디드 소프트웨어, RTOS, 시스템 프로그래밍>

유 대 철(정회원)
2010년 한양대학교 컴퓨터전공 학사 졸업.
2011년 현재 한양대학교 전자컴퓨터공학과 석·박사 통합과정.
<주관심분야: 저전력 시스템 설계, 이종 멀티코어 시스템 설계>

정 기 석(정회원)-교신저자
1989년 서울대학교 컴퓨터공학과 학사 졸업.
1998년 Univ. of Illinois at Urbana-Champaign, Computer Science 박사 졸업.
1998년 Univ. of Illinois at Urbana-Champaign, 강의 전담 교수.
2000년 Synopsys, Inc. Sr. R&D Engineer
2001년 Intel Corp. Staff Engineer.
2001년 홍익대학교 컴퓨터 공학과 조교수.
2004년 한양대학교 정보통신대학 조교수.
현재, 한양대학교 융합전자공학부 부교수.
<주관심분야: 임베디드 시스템, VLSI 및 SoC 설계>



조 걸(정회원)
2009년 한양대학교 미디어통신공학과 학사 졸업.
2011년 현재 한양대학교 전자컴퓨터공학과 석·박사 통합과정.
<주관심분야: 임베디드 시스템, 멀티코어 시스템>

황 영 시(정회원)
2003년 대진대학교 컴퓨터공학 학사 졸업.
2005년 (주)한국정보통신 기술연구소 연구원.
2008년 한양대학교 전자컴퓨터통신공학과 석사 졸업.
현재 한양대학교 전자컴퓨터통신공학 박사.
<주관심분야: 임베디드 시스템, 멀티코어 시스템, 저전력 시스템, RTOS>