논문 2008-45SD-12-3

플라잉 브릿지 버스 아키텍처

(Flying Bridge Bus Architecture)

이 국 표*, 윤 영 섭*

(Kookpyo Lee and Yungsup Yoon)

요 약

SoC와 같이 많은 컴포넌트로 구성된 버스 토폴로지(topology)에서는 여러 버스가 계층적으로 나누어져 있으며, 버스 간에는 브릿지로 연결되어 있다. 브릿지 토폴로지는 버스 내에서 컴포넌트의 동시 통신이 가능하기 때문에 버스의 성능을 획기적으로 향상시킬 수 있다. 그러나 버스 간의 데이터 전송이 발생할 때, 브릿지 블록에서 레이턴시가 증가할 수 있다. 본 연구에서는 다양한 종류의 브릿지 토폴로지에 대해 살펴보고, 각각의 장단점을 분석해 보았으며, 성능, IP의 재사용, 타이밍 마진, 게이트 수, 설계 마진 등의 측면에서 우수한 성능을 보여주고 있는 플라잉 브릿지 토폴로지에 대해 제안하고 있다. 기존 버스 브릿지는 단지 버스 간의 데이터를 교환하는 역할을 하지만, 플라잉 브릿지는 버스와 슬레이브 간에 직접 통신을 통해 데이터 전송하는 특징을 갖는다. 위와 같은 직접 통신방법은 공용버스의 트래픽 부담을 줄일 수 있으며 고성능의 브릿지 통신을 가능하게 할 수 있다.

Abstract

Several shared buses are divided hierarchically and connected with a bridge in the bus topology that consists of many components such as SoCs. Because the bridge topology is capable of the simultaneous communication of components in the several buses, the bus performance has improved definitely. However, when the inter-bus data transaction happens, the latency increases seriously in the bridge block. In this paper, a variety of bridge architectures are analyzed in the point of view of merit and demerit. Superior frying bridge topology is proposed in the aspects of performance, IP reusablility, timing margin, gate count and circuit complexity. In contrast with the conventional bridge that has only a role to switch the inter-bus data, the frying bridge can communicate directly between the bus and the slave, which decreases the traffic overhead of a shared bus and improves the performance of a bridge communication.

Keywords : bus topology, SoC, AMBA

I. 서 론

여러 가지 기능을 갖는 컴포넌트를 하나의 칩에 집적화시킨 SoC칩은 IP 간의 호환성(flexibility), 여러 컴포넌트를 추가할 수 있는 확장성 (scalability), 속도, 저전력 등에 대해 주로 연구되고 있다.[1~3] 세계적인 회사에서는 독자적인 SoC용 버스 아키텍처를 개발하고 있는데, AMBA[4], Core Connect[5], Silicon MicroNetworks[6] 등의 버스 아키텍처가 현재 상용화되어 있다.

컴포넌트가 많은 SoC의 대부분 버스 토폴로지(topology)에서는 여러 버스를 계층적으로 구분하고 버스 사이에 브릿지를 연결하여 사용하고 있다. 이 방법은 컴포넌트의 증가분만큼 계층적인 버스를 추가함으로써, 버스에 대한 컴포넌트의 집중현상을 막고, 공용버스의 성능을 향상시킬 수 있다. 그러나 버스 간의 데이터 전송이 발생할 때, 브릿지 블록에서 레이턴시가 증가하는 단점을 가지고 있다.

본 연구에서는 현재까지 연구되고 있는 다양한 종류의 브릿지 토폴로지에 대해 살펴보고, 각각의 장단점을 분석해 보았다. 그리고 성능, IP의 재사용, 타이밍 마진, 게이트 수, 설계 마진 등의 측면에서 좀더 우수한 플라잉 브릿지 토폴로지에 대해 제안하고 있다.

기존 버스 브릿지는 버스 간의 데이터를 중개하는 역

* 정회원, 인하대학교 전자공학과
(Div. of Electronic Engineering, Inha University)
접수일자: 2008년5월7일, 수정완료일: 2008년12월1일

할을 하지만, 플라잉 브릿지는 버스와 슬레이브 간에 직접 통신을 통해 데이터 전송하는 특징을 갖는다. 위와 같은 직접 통신방법은 공용버스와 무관하게 동작됨에 따라, 공용버스의 트래픽 부담을 줄일 수 있으며 고성능의 브릿지 통신을 가능하게 할 수 있다.

Ⅱ. 본 론

1. 브릿지를 통한 데이터 전송방법

그림 1에는 다중 버스구조의 버스통신이 나타나 있는데 ①의 실선 화살표로 나타나 있는 것처럼 공용버스1과 공용버스2는 개별적으로 데이터 전송하는 것이 가능하다. 그러므로 싱글 버스구조와 달리 여러 데이터를 동시에 전송할 수 있는 장점을 가지고 있다.

그런데, ②의 점선 화살표로 나타나 있는 것처럼 공용버스1의 마스터와 공용버스2의 슬레이브 사이에서 데이터 전송을 할 경우, 브릿지 블록을 통과해야 한다. 만약 마스터 M2와 슬레이브 S4 사이에 데이터 전송을 한다면, 아비터 AB1에 버스 사용 허가를 받아서 브릿지 블록에 마스터 M2의 데이터를 전송해야 하며, 그 이후 브릿지가 아비터 AB2에 버스 사용 허가를 받아서 최종적으로 슬레이브 S4에 M2의 데이터를 전송해야 한다. 이때 브릿지는 데이터를 받아들이는 슬레이브 기능과 데이터를 전송하는 마스터 기능을 동시에 해야 하며, 공용버스1과 공용버스2의 버스 프로토콜, 클록 주파수 등을 고려하여 데이터를 중개시켜야 한다. 결국 버스

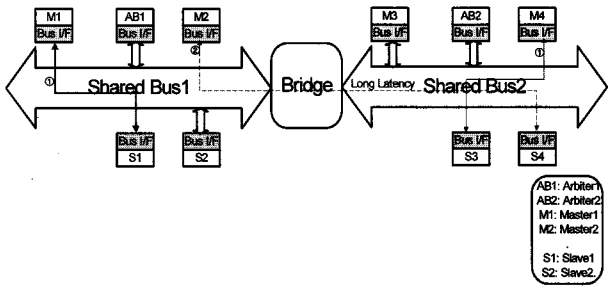

그림 1. 다중 버스구조의 버스 통신
Fig. 1. Bus communications of multiple bus architecture.

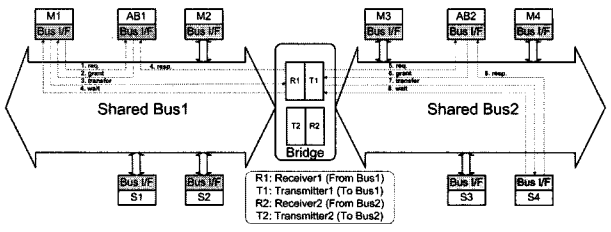

그림 2. 다중 버스에서의 데이터전송 흐름도
Fig. 2. Data transaction flow in the multiple shared bus.

사이의 데이터 전송은 복잡하게 진행되고 레이턴시(latency)가 매우 크기 때문에, 다중버스 아키텍처의 성능하락에 가장 큰 원인이 된다.

AMBA[4], Core Connect[5], Silicon MicroNetworks[6] 등의 대부분 버스 아키텍처는 브릿지 등을 이용한 다중 버스통신 방법을 채택하고 있다. 그림 2에서 다중버스 데이터 통신은 브릿지 블록에 의해 제어된다. 브릿지 블록에서는 공용버스에서 전송되는 버스 신호를 받는 수신부(receiver) R1, R2와 받은 버스 신호를 다른 공용버스로 전송하기 위한 송신부(transmitter) T1, T2가 방향에 따라서 2쌍 존재하는데, 동작 순서는 표 1과 같다.

다중버스에서 브릿지를 통과하는 그림 2의 데이터 전송에 대한 타이밍 도가 그림 3에 나타나 있다. 데이터 전송을 위해 공용버스1(shared bus1)의 마스터에서 버스 요청신호(req.1)가 "0"에서 "1"로 활성화되어 아비터에 전달된다. 아비터의 버스허가 신호(grant1)에 따라, 브릿지가 공용버스2(shared bus2)의 아비터에게 요청신호(req.2)를 보내 주고, 버스허가 신호(grant2)를 받아서 최종적으로 데이터를 전송하게 된다.

브릿지의 총 데이터 전송을 위한 레이턴시(latency)는 식(1)과 같다.

$$L_{bt} = L_{r1} + L_{r2} + L_b + L_d \qquad \text{식(1)}$$

여기서, L_{bt}는 브릿지통신에 소요되는 총 레이턴시,

표 1. 다중 버스구조의 데이터전송
Table 1. Data transaction of multiple bus architecture.

순서	내용
1	공용버스 1(shared bus1) 의 마스터가 아비터 AB1에게 버스 사용 요청(req.) 을 하고, 아비터 AB1이 우선순위 알고리즘을 통해 마스터에게 버스사용권을 부여(grant) 한다.
2	버스 사용권을 부여받은 마스터는 슬레이브에 주소, 데이터, 컨트롤 신호 등을 브릿지(bridge) 의 수신부 1(receiver1) 에 전송(transfer) 한다.
3	브릿지의 수신부1은 데이터 처리상황을 파악하여 마스터와 아비터 AB1에게 각각 지연신호(wait)와 응답신호(resp.)를 보내준다.
4	브릿지의 송신부1(transmitter1) 은 아비터 AB2에게 공용버스2의 버스 사용 요청을 하고, 아비터 AB2가 우선순위 알고리즘을 통해 브릿지에게 버스사용권을 부여(grant) 한다.
5	버스 사용권을 부여받은 브릿지는 슬레이브에게 주소, 데이터, 컨트롤 신호 등을 브릿지(bridge) 의 송신부1에 전송(transfer) 한다.
6	브릿지의 발신부1은 데이터 처리상황을 파악하여 마스터와 아비터 AB2에게 각각 지연신호(wait)와 응답신호(resp.)를 보내준다.

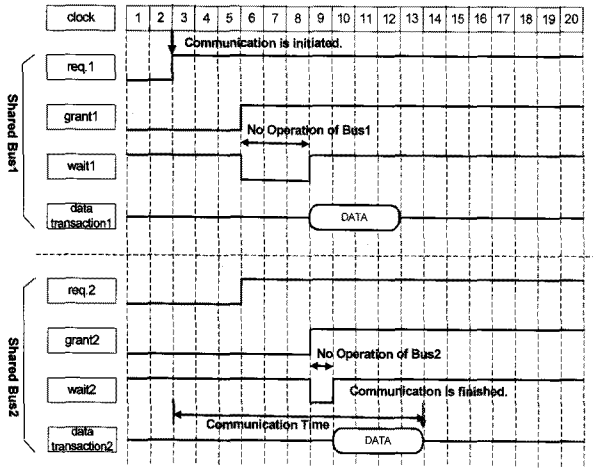

그림 3. 브릿지 통신에 대한 타이밍 도
Fig. 3. Timing diagram about bridge communication.

L_{r1}은 공용버스1에서 버스요청 레이턴시, L_{r2}는 공용버스2에서 버스요청 레이턴시, L_b는 브릿지를 통과할 때 발생하는 레이턴시, L_d는 실제 데이터 전송에 소요되는 레이턴시이다. 결국 그림 3에서 4개의 데이터를 버스트(burst)로 전송할 경우에 총 11cycle의 레이턴시가 필요하다. 그러나 이는 버스 요청이 3cycle만에 이루어질 경우이며, 실제 버스 요청에 필요한 시간은 우선순위에 따라 크게 늘어날 수 있다. 그림3에서 버스1의 사용을 위해 기다리는 시간(wait1)이 공용버스2에서 버스사용권을 받는 시간(grant2)과 동일하게 3cycle 소요되는데, 이때 버스1의 사용이 중지되므로 버스의 성능이 떨어지게 된다. 그리고 버스2도 브릿지를 통과할 때 발생하는 레이턴시 L_b만큼 버스동작이 중지된다.

결론적으로 브릿지를 이용한 다중버스 아키텍처는 버스 내의 데이터 통신이 별개로 이루어져서 버스 별로 병렬처리가 가능한 장점이 있지만, 버스 사이의 데이터 통신은 시간 지연이 크기 때문에 성능이 크게 악화될 수 있다.

2. 브릿지와 관련된 최근 연구

브릿지를 통과할 경우 성능이 심각하게 하락하는 현상을 극복하기 위해, 여러 가지 연구가 이루어지고 있다. AMBA 시스템에서 사용하는 방식은 브릿지에서 분리 응답신호(resp.1=SPLIT)를 발생시켜서 버스점유권(grant1)을 철회하는 방식을 이용한다.[4]

그림 3에서 공용버스1의 wait1신호가 3cycle동안 "0"이었지만, 버스점유권을 철회하는 방식을 사용하는 그림 4의 경우는 공용버스1의 wait1신호가 항상 "1"을 유지한다. 이럴 경우 wait1신호가 "1"이므로 새로운 버스 통신이 가능하여, 버스1의 사용률이 높아진다.

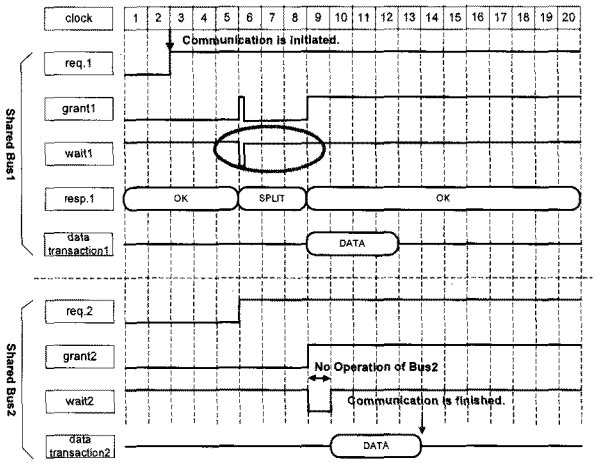

그림 4. 버스 split에 대한 타이밍 도
Fig. 4. Timing diagram about bus split.

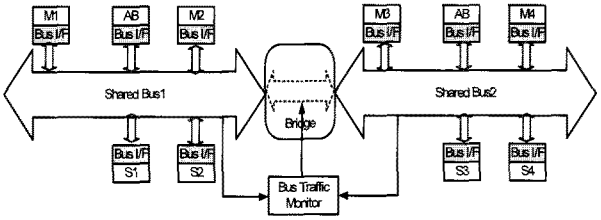

그림 5. 버스 트래핑 모니터를 이용한 브릿지 통신
Fig. 5. Bridge communication using bus traffic monitor.

그러나 split 방식은 아비터, 마스터, 슬레이브에 대하여 split을 지원하도록 수정·설계해야 하며, 설계 또한 까다롭다. 경우에 따라 버스철회 후, 다시 버스요청을 받으려고 할 경우 많은 시간지연이 발생할 수도 있다.

버스 전송량을 판단하는 모니터 블록을 이용하여 브릿지 통신의 성능을 향상시킨 FLEXBUS 아키텍처가 최근에 연구되고 있다.[7] FLEXBUS 아키텍처구조를 간단하게 재구성하여 그림 5에 표현하였다. 그림 5는 버스 트래픽 모니터(bus traffic monitor) 블록에서 버스전송량을 판단하여, 브릿지를 bypass하거나, 브릿지를 사용하거나 하는 판단을 한다. 예를 들어 마스터 M1에서 슬레이브 S3으로 데이터를 전송할 경우에는 브릿지를 bypass시켜서 브릿지 레이턴시를 줄일 수 있다. 그리고 마스터 M1에서 슬레이브 S2으로 데이터를 전송하고 동시에 마스터 M3에서 슬레이브 S3으로 데이터를 전송할 경우, 브릿지를 연결시켜서 성능을 향상시킬 수 있다. 버스 전송을 판단하여 싱글버스 구조와 다중버스 구조로 변형되어 성능을 극대화시킨 방법이다.

그림 5의 방식이 적용되기 위해서는 우수한 성능의 버스 트래픽 모니터를 구현해야 하고, 타이밍 지연을 최소화해야 한다. 버스 트래픽 모니터를 우수하게 하기 위해서는 설계로직이 복잡해지므로, 타이밍 지연 또한

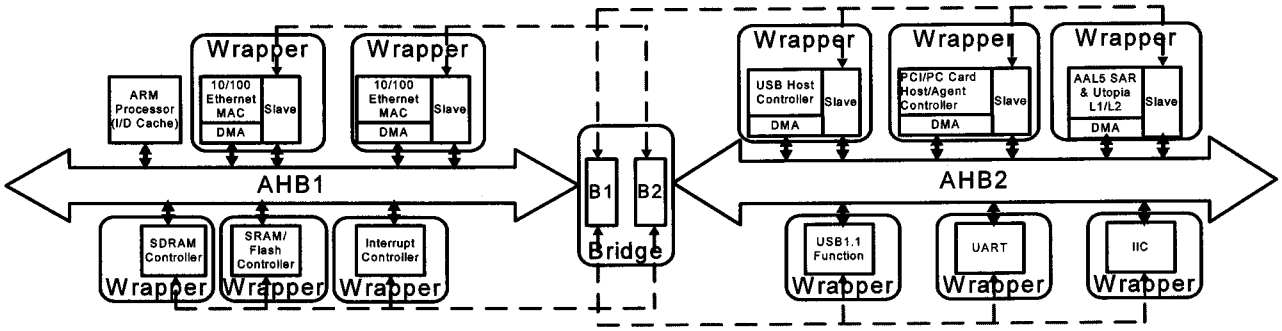

그림 6. 플라잉 브릿지의 블록도
Fig. 6. Block diagram of a flying bridge.

길어질 수밖에 없다.

결국 split 방식과 버스 트래픽 모니터 방식은 브릿지 성능을 향상시킬 수 있지만, 설계의 복잡도와 타이밍 지연의 단점을 가지고 있어서 실제적인 적용에 많은 한계를 가지고 있다.

3. 플라잉 브릿지 구조

그림 6은 플라잉 브릿지(flying bridge)를 이용한 버스통신을 보여주고 있다. ARM 프로세서와 이더넷 두 채널에 대한 마스터는 공용버스1에 연결되어 있으며, USB 호스트, PCI 컨트롤러와 SAR 컨트롤러는 공용버스2에 연결되어 있는 IEEE 802.11 네트워크 SoC[8]를 플라잉 브릿지에 적용하였다. 공용버스1의 마스터에서 공용버스2의 슬레이브로 버스통신이 있을 경우, 브릿지 B1 블록을 거친 후 공용버스2를 통과하지 않고 직접 슬레이브와 버스통신을 하게 된다. 반대로 공용버스2의 마스터에서 공용버스1의 슬레이브로 버스통신이 있을 경우, 브릿지 B2 블록을 거친 후 공용버스1을 통과하지 않고 직접 슬레이브와 버스통신을 하게 된다.

플라잉 브릿지를 통한 버스들 간의 데이터 통신은 슬레이브 단에 연결되어 있는 공용버스를 통과하지 않으므로 버스 중재에 따른 대기시간을 절약할 수 있고, 싱글버스 데이터 통신과 유사한 성능을 나타낼 수 있다. 그러나 모든 슬레이브는 공용버스와 브릿지 간의 인터페이스를 할 수 있는 래퍼(wrapper)가 있어야 하며, 새로운 기능을 갖는 브릿지를 설계해야 한다.

플라잉 브릿지를 실제적으로 적용하기 위해서는 표 2의 체크리스트를 우선적으로 파악해야 한다. 표 2에서 1번 성능부분은 다음 5절의 성능분석 부분에서 자세히 파악할 것이지만, 기본적으로 버스 중재의 감소에 따른 플라잉 브릿지의 성능향상을 예상할 수 있다. 표 2에서 2-5번은 다음 절에서 직접 설계해 보고 특성들에 대해 분석할 것이다.

표 2. 플라잉 브릿지 적용을 위해 고려해야 할 사항
Table 2. A point to be considered in oder to apply flying bridge.

No.	체크리스트
1	버스 성능을 실제적으로 향상시킬 수 있는가?
2	기존 IP의 재사용이 가능한가?
3	회로 추가 시 타이밍 복잡도가 낮아서 타이밍 마진이 충분하고, 원하는 클록 주파수를 나타낼 수 있는가?
4	전체에서 차지하는 게이트 수(gate count)가 충분해서 칩 면적에 많은 영향을 미치지 않는가?
5	개념적으로 설계자가 이해하기 쉽고, 설계의 난이도가 높지 않은가?

4. 플라잉 브릿지 설계

플라잉 브릿지는 표 2의 체크리스트를 충실히 반영하여 설계하였다. 기존 IP에 대해서 변형없이 사용가능하도록 슬레이브 래퍼 블록을 도입하였으며, 표2의 3-5항을 충실하게 이행하기 위해 브릿지 블록도 최대한 단순화하였다. 전 세계의 70%이상을 차지하고 있는 AMBA 시스템 중에 마스터를 지원하는 AHB 버스에 플라잉 브릿지를 적용하였는데, 그림 7에 브릿지 B1 블록도가 나타나 있다. 브릿지 B1 블록과 B2 블록은 설계적으로 동일하며 단지 공용버스와 슬레이브 연결만 다르다. 공용버스로부터 나오는 AHB 신호는 모두 슬레이브로 직접 연결되며, 슬레이브들로부터 나오는 AHB 신호는 먹스 로직을 거쳐서 공용버스로 전달된다. 브릿지 블록은 마스터 관련 AHB 신호는 모두 bypass되고, 슬레이브로부터 받은 3신호는 먹스 로직을 거쳐서 공용버스로 전달된다.

그림 8에 슬레이브 래퍼에 대한 블록도가 나타나 있는데, 공용버스의 신호와 브릿지 신호를 선택하는 먹스 로직과 "HREADY"를 만들어 주는 로직으로 구성된다.

그림 2의 일반적인 브릿지 블록과 비교해서 설계 구조가 아주 간단하며 단순히 신호연결을 목적으로 구성

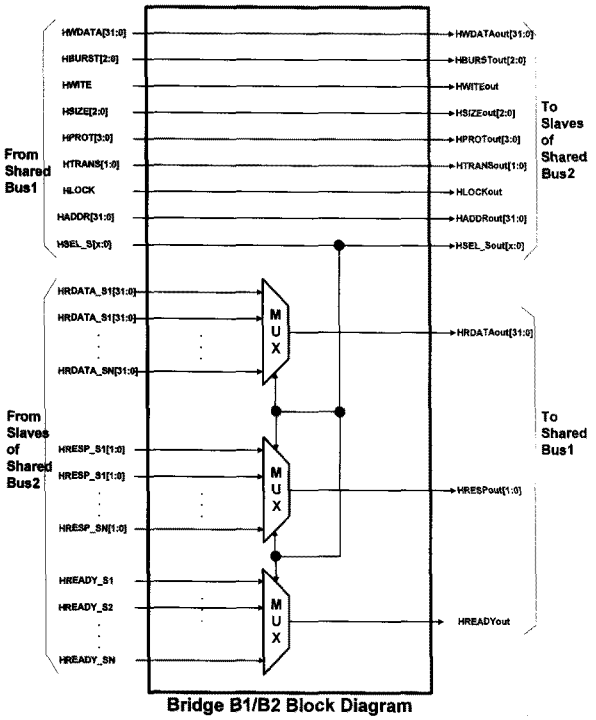

그림 7. 브릿지 B1 블록도
Fig. 7. Block diagram of bridge B1.

되어 있다. 상대적으로 설계상의 타이밍 마진도 충분하고, 칩면적을 감소시킬 수 있다. 슬레이브 래퍼는 브릿지로부터 받은 마스터 신호와 공용버스로부터 받은 마스터 신호를 선택하는 먹스로직과 슬레이브의 "HREADY"신호를 발생하는 블록으로 구성되어 있다. 그러나 브릿지 블록과 마찬가지로 실제 신호 지연시간은 미미한 수준이며, 전체 칩 사이즈에도 큰 영향을 주지 않는다.

5. 플라잉 브릿지의 성능분석

표 3에 브릿지 종류별 특성을 비교하였다. 그림 2의 일반 브릿지는 수신부와 송신부를 포함하고 있으며, 버스의 성능은 낮지만 설계의 난이도, 게이트 수, 타이밍 마진 등은 다른 브릿지들과 비교하여 중간정도로 볼 수 있다. 그

표 3. 브릿지 종류별 특성비교
Table 3. Property comparison with various bridge types.

	일반 브릿지 (그림2)	버스 모니터 브릿지 (그림4)	Split 브릿지 (그림5)	플라잉 브릿지 (그림6)
버스 성능	낮다	높다	중간	높다
IP의 재사용	높다	높다	낮다	높다
타이밍 마진	중간	적다	중간	많다
게이트 수	중간	많다	많다	적다
설계의 난의도	중간	높다	높다	낮다

림 4의 버스 모니터 방식은 데이터 트래픽을 판단하여 버스 토폴로지를 바꾸는 형태이므로 설계 난이도가 상대적으로 높으며, 그림 5의 split 방식은 버스점유를 철회하고 필요에 따라 버스를 재요청하므로 구현하기 어렵고, 마스터, 슬레이브, 아비터를 수정해야 한다. 그에 반해 플라잉 브릿지는 간단한 래퍼만 슬레이브에 추가해 주고, bypass 형태의 브릿지 블록을 설계하여 손쉽게 구현할 수 있다. 설계의 오버헤드와 타이밍 마진 측면에서 다른 브릿지 토폴로지에 비해 우수하다.

다음으로 플라잉 브릿지의 성능에 대해 파악해 보았는데, 브릿지 통신의 성능은 식(1)의 레이턴시에 의해 결정된다.

식(1) 관점에 split 브릿지는 일반 브릿지와 동일한 레이턴시를 나타내거나, 버스 재요청 과정에서 일반 브릿지보다 레이턴시가 길어질 수 있다. 버스 모니터 브릿지는 공용버스를 재선택하거나, 싱글버스와 다중버스를 버스 트래픽에 따라 선택하는 방법이므로, 버스 트래픽의 복잡도, 버스 모니터의 성능에 의해 좌우된다.

우선적으로 split 브릿지보다 레이턴시가 짧은 일반 브릿지를 비교해 보았다. 식(1)에서 공용버스1의 버스요청 레이턴시 L_{r1}과 데이터 전송 레이턴시 L_d는 일반 브릿지와 플라잉 브릿지가 동일하며, 플라잉 브릿지에서 브릿지를 통과할 때 발생하는 레이턴시 L_b는 필요없다.

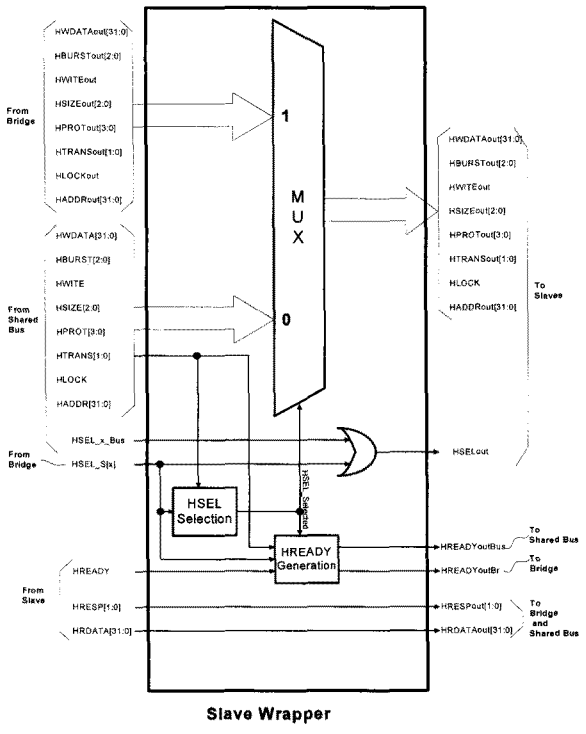

그림 8. 슬레이브 래퍼 블록도
Fig. 8. Block diagram of slave wrapper.

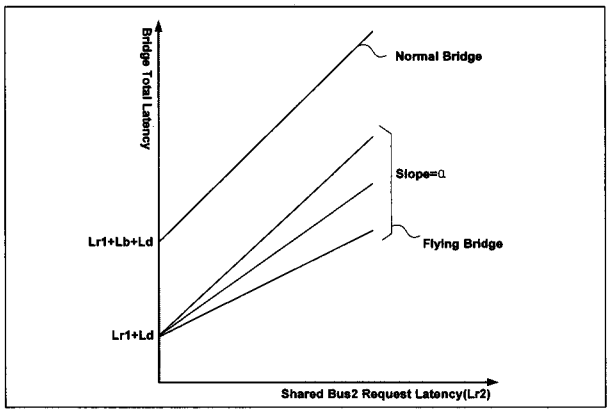

그림 9. 공용버스2의 요청 레이턴시에 따른 전체 브릿지 레이턴시

Fig. 9. Total bridge latency due to shared bus2 request latency.

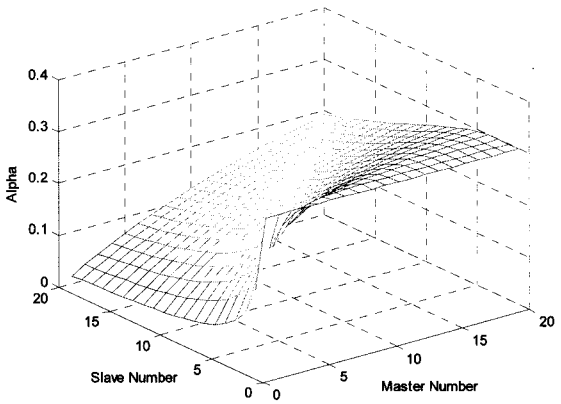

그림 10. 마스터와 슬레이브의 개수에 따른 α값

Fig. 10. α value due to the number of masters and slaves.

식(2)와 그림 9에 플라잉 브릿지의 레이턴시에 대하여 자세히 표현되어 있다.

$$L_{fbt} = L_{r1} + \alpha L_{r2} + L_d \qquad \text{식 (2)}$$

$$\alpha = \sum_{n=1}^{n} \sum_{p=1}^{p} P_{BrSp} \bullet P_{MnSp} \qquad \text{식 (3)}$$

식(2)에서 L_{fbt}와는 플라잉 브릿지의 전체 레이턴시, α는 슬레이브에서 브릿지 요청과 공용버스 요청이 동시에 발생할 확률, P_{BrSp}는 브릿지에서 슬레이브 Sp에 접근할 확률, P_{MnSp}는 마스터 Mn이 슬레이브 Sp에 접근할 확률이다. 식(3)에서 마스터 Mn이 슬레이브 Sp에 접근할 확률이 마스터와 슬레이브에 관계없이 동일하고, 모든 마스터가 슬레이브에 접근할 확률이 30%라고 가정하면 그림 10과 같은 α값을 얻을 수 있다.

그림 10에 마스터가 3개, 슬레이브가 6개 존재하는 그림 6의 IEEE 802.11 네트워크 SoC의 경우를 대입하면, α가 0.12이다. 결국 일반 브릿지와 비교했을 때 플라잉 브릿지의 레이턴시가 크게 감소됨을 알 수 있다.

Ⅲ. 결 론

다양한 종류의 브릿지 토폴로지에 대해 성능, IP의 재사용, 타이밍 마진, 게이트 수, 설계 마진 등에 대해 살펴보았다. 그리고 플라잉 브릿지 토폴로지 방식을 제안하고, 기존의 브릿지 방식과 비교해 보았다.

플라잉 브릿지는 버스 간의 데이터 전송을 중개하는 역할을 하는 브릿지의 고전적인 방식을 탈피하고, 데이터를 직접 전달하는 방식으로 설계의 복잡도를 크게 낮출 뿐만 아니라 우수한 성능을 나타내었다.

다음 논문에서는 다중버스의 성능분석 시뮬레이터를 개발하여 브릿지 종류별 동적 성능분석을 구현할 계획이다. 그리고 RTL 코드 설계 및 합성을 통해 정량적인 게이트 수와 타이밍 마진을 파악해 보고, 버스 모니터 브릿지와 플라잉 브릿지의 동적 시뮬레이션을 통해 성능을 비교 분석할 계획이다.

참 고 문 헌

[1] R. Lu and C.-K. Koh, "SAMBA-Bus: A High Performance Bus Architecture for System-on-Chips", IEEE Trans. on VLSI Systems, vol. 15, no. 1, pp.69-79, 2007.

[2] K. Lahiri, A. Raghunathan, and G. Lakshminarayana, "The lotterybus on-chip communication architecture", IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 14, no. 6, pp.596-608, 2006.

[3] R. Lu and C.-K. Koh, "A high performance bus communication architecture through bus splitting", in Proc. Asia-South Pacific Design Autom. Conf., pp.751-755, 2004.

[4] ARM, Limited, Cambridge, U.K., "AMBA Specification", 1999.

[5] IBM, Armonk, NY, "CoreConnect bus architecture", 1999.

[6] Sonics, Inc., Mountain View, CA, "Silicon micronetworks technical overview", 2002.

[7] K. Sekar, K. Lahiri, A. Raghunathan, and S. Dey, "FLEXBUS: A high performance system-on-chip communication architecture with a dynamically configurable topology", in Proc.

Design Autom. Conf., pp.571-574, 2005.
[8] http://www.samsung.com/global/business/semiconductor/productInfo.do?fmly_id=234&partnum=S3C2510A

저 자 소 개

이 국 표(정회원)
대한전자공학회 논문지
제45권 SD편 제4호 참조

윤 영 섭(정회원)
대한전자공학회 논문지
제45권 SD편 제4호 참조