논문 2008-45SD-5-10

다중처리가 가능한 새로운 Globally Asynchronous, Locally Dynamic System 버스 구조

(A Novel Globally Asynchronous, Locally Dynamic System Bus Architecture Based on Multitasking Bus)

최 창 원*, 신 현 출**, 위 재 경***

(Chang-Won Choi, Hyeon-Chul Shin, and Jae-Kyung Wee)

요 약

본 논문에서는 새로운 On-Chip 버스로 다중처리 기반의 GALDS 버스 구조를 제안하였고 성능을 검증하였다. 제안된 GALDS 버스 구조는 멀티 마스터 멀티 슬레이브의 다중 처리를 지원하는 세그먼트(segment) 기반의 고성능의 양방향 다중처리 버스 구조(bi-direction multitasking bus architecture)이다. 또한, 시스템의 태스크(task) 분석에 의해서. 버스는 버스 동작 주파수의 배수 값을 갖는 주파수 사이에서 각각의 IP에 최적화된 동작 주파수를 선택하기 때문에 전체 전력 소모를 줄일 수 있다. 서로 다른 동작 주파수를 입력받은 IP들 간의 효율적인 데이터 통신을 위하여, 본 구조에서는 비동기 양방향 FIFO를 기반으로 하는 비동기 Wrapper 설계하였다. 또한, 버스 세그먼트의 추가만으로 시스템의 쉬운 확장이 가능하기 때문에, 제안된 구조는 IP 재사용 및 구조적 변경이 용이한 장점을 갖는다. 제안된 버스의 검증을 위해 4-마스터/4-슬레이브를 가지는 4-세그먼트의 버스와 비동기 Wrapper를 Verilog HDL을 이용하여 구현하였다. 버스의 다중처리동작 검증은 버스와 IP의 동작 주파수 비가 1:1, 1:2, 1:4, 1:8인 경우를 기준으로 시뮬레이션을 통해 마스터 IP에서 슬레이브 IP 사이의 데이터 읽기 및 쓰기 전송 동작을 확인하였다. 데이터 전송은 Advanced Microcontroller Bus Architecture (AMBA)과 호환 가능한 16 Burst Increment 모드로 하였다. 제한된 GALDS 버스의 최대 동작 지연시간은 쓰기 동작 시 22 클럭, 읽기 동작 시 44 클럭으로 확인되었다.

Abstract

In this paper, we propose a novel Globally Asynchronous, Locally Dynamic System (GALDS) bus and demonstrate its performance. The proposed GALDS bus is the bidirectional multitasking bus with the segmented bus architecture supporting the concurrent operation of multi-masters and multi-slaves. By analyzing system tasks, the bus architecture chooses the optimal frequency for each IP among multiples of bus frequency and thus we can reduce the overall power consumption. For efficient data communications between IPs operating in different frequencies, we designed an asynchronous and bidirectional FIFO based on an asynchronous wrapper with hand-shaking interface. In addition, since systems can be easily expandable by inserting bus segments, the proposed architecture has advantages in IP reusability and structural flexibility. As a test example, a four-segment bus having four masters and four slaves were designed by using Verilog HDL. We demonstrate multitasking operations with read/write data transfers by simulation when the ratios of operation frequency are 1:1, 1:2, 1:4 and 1:8. The data transfer mode is a 16 burst increment mode compatible with Advanced Microcontroller Bus Architecture (AMBA). The maximum operation latency of the proposed GALDS bus is 22 clock cycles for the bus write operation, and 44 clock cycles for read.

Keywords : Segmented Bus, AMBA, GALS, DVFS, GALDS

Ⅰ. 서 론

디지털 융합화(Digital Convergence)된 기기의 수요가 증가하면서 하나의 기기에 다양한 기능을 포함하기 위한 핵심요소인 고성능의 SoC 설계가 요구되고 있다.

* 학생회원, ** 정회원, ** 정회원-교신저자, 숭실대학교 전자공학과
(School of Electronic Engineering Soongsil University)
※ 본 연구는 숭실대학교 교내연구비 지원으로 이루어졌음
접수일자: 2008년1월15일, 수정완료일: 2008년4월25일

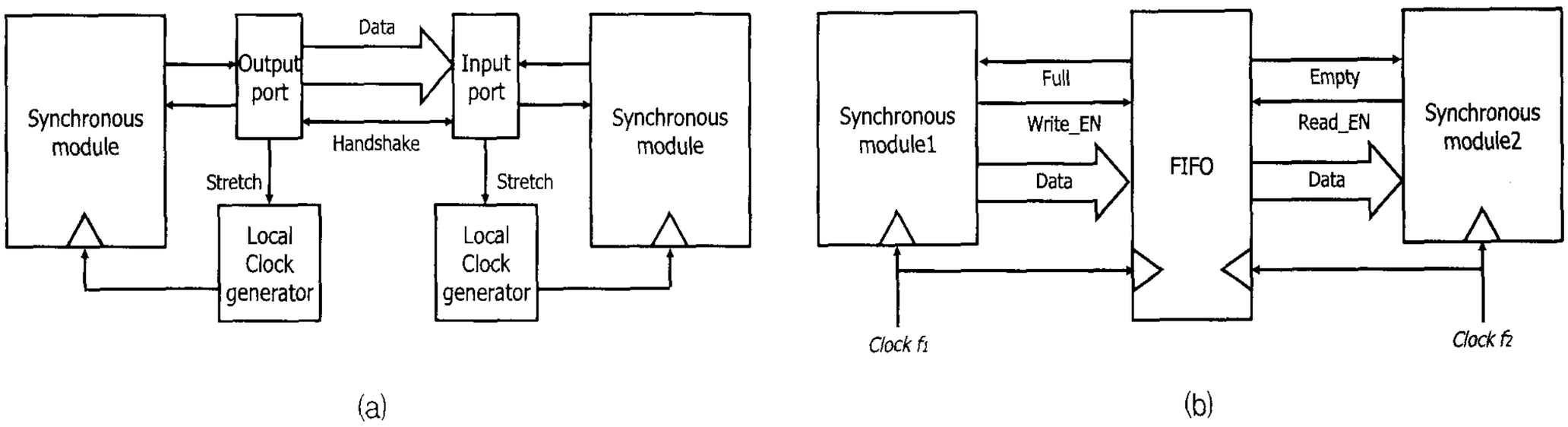


그림 1. (a)Pausible Clocking 기반의 GALS 시스템 (b)FIFO 기반의 GALS 시스템
Fig. 1. (a)GALS system with Pausible Clocking (b)GALS system with FIFO-Based.

따라서 하나의 칩 내에 다양한 IP들이 집적되고 IP 간의 신속하고 정확한 통신으로 시스템의 성능을 향상시킬 수 있는 고성능의 버스가 필요하다. 특히 멀티미디어용 SoC는 고속 임베디드 프로세서 및 DSP, 고화질 동영상을 처리하기 위한 높은 버스 대역폭을 요구하는 IP들을 필요로 한다. 하지만 현재 사용되고 있는 공유 버스 구조는 한 번에 하나의 IP만이 버스에 접근이 가능하다. 이 때문에 각 IP는 버스 사용권 허가를 기다리는 시간이 길어지고 이는 버스 시스템 전체의 병목현상으로 이어진다. 이러한 병목현상은 각각의 IP들의 동작 레이턴시를 증가시켜 전체 시스템 성능의 심각한 저하를 유발한다. 또한 하나의 클럭 소스에서 모든 IP에 클럭을 공급하는 동기방식의 사용으로 높은 성능을 위하여 점점 고속으로 동작하는 IP들 간의 클럭 분배 문제가 발생한다. 즉, 전체 시스템의 동기화가 어렵다.[1~2]

본 논문에서는 기존의 공유 버스에서의 낮은 버스 대역폭을 향상시키기 위한 버스 다중 처리 방법으로 분할된 버스(Segmented Bus) 구조를 사용하여 가능한 많은 IP들 사이의 통신을 동시에 처리할 수 있게 하여 버스 전체 대역폭을 증가시켰다.[3] 또한, 효과적인 저전력 시스템의 구현을 위한 DVFS 기법의 적용을 위해 시스템의 태스크(task)에 따라 버스 동작 주파수의 정수배에 해당하는 서로 다른 동작 주파수가 각 IP에 공급되도록 하였다.[4] 여기서 서로 다른 주파수로 동작하는 IP와 버스, 버스와 IP 사이의 통신 문제를 해결하기 위해 GALS (Globally Asynchronous, Locally Synchronous) 기법을 적용하였다. 이를 위해 Asynchronous FIFO를 기반으로 하는 Asynchronous Wrapper를 이용하여 IP와 버스 사이의 데이터 전송 시 정확한 데이터 통신이 가능하도록 설계하였다. 그리하여 각 IP와 버스사이에서는 비동기방식(Globally Asynchronous)으로 통신이 가능하게 하였다.[5~6]

본 논문은 다음과 같이 구성된다. Ⅱ-1에서는 GALS 시스템의 개요와 기존의 GALS기법의 솔루션을 비교해 본다. Ⅱ-2에서 본 논문에서 제안하는 GALDS 방식의 버스 구조를 설명하고, Ⅱ-3에서 전체 버스동작을 설명한다. Ⅲ에서는 전체 버스 구현 시뮬레이션 결과를 보이고 Ⅳ에서 결론을 맺는다.

Ⅱ. 본 론

1. GALS 개요 및 솔루션 비교

GALS 시스템은 기본적으로 전역적(Globally) 단일 클럭을 사용하지 않으며, 각 모듈이 서로 독립적인 클럭을 사용하는 지역적 동기(Locally Synchronous) 모듈로 구성된다. 각 모듈간의 데이터 전송은 특화된 접속장치(Wrapper)를 통해 비동기 핸드셰이크 프로토콜(Handshake Protocol)에 의해 수행된다. 그러므로 전역 클럭을 사용하지 않음으로써 클럭 스큐, 지터와 소비전력 문제 해결이 가능하다. 이러한 GALS 시스템을 구성하기 위해 가장 중요한 부분은 서로 다른 클럭 도메인 사이의 인터페이스 역할을 하는 모듈이다. 사용되는 방법은 크게 Pausible Clocking과 FIFO를 이용한 방법이 있다.[1]

가. Pausible Clocking을 사용한 GALS 시스템

그림 1(a)는 Pausible Clocking 방식의 전체적인 구성이다.[7] 이 방식은 데이터 전송 시 데이터를 보내는 모듈과 받는 모듈 마다 각각의 자체적인 Local Clock Generator를 갖고 있다. 데이터 전송은 각각의 모듈에

연결된 입출력 포트를 이용한다. 출력 포트에서 상대편의 입력 포트로 데이터를 보내는 동안에는 Local Clock Generator의 출력이 멈춰 지역 동기 모듈(Locally Synchronous module)의 동작이 정지하게 된다. 이 때 출력 포트와 입력 포트 사이에서 핸드셰이크를 이용하여 데이터를 전송한 후에 다시 Local Clock Generator의 동작이 시작된다. 그리하여 데이터를 전송 시 이종 클럭 도메인 간의 클럭 문제를 해결 할 수 있다. 이 방법은 면적과 동작 레이턴시(Latency)가 짧고 Meta Stablility 문제를 해결 할 수 있다.[1] 하지만 클럭의 공급 없이 핸드셰이크만으로 데이터를 전송하기 때문에 한 번에 보낼 수 있는 데이터의 양이 극히 제한되고 많은 양의 데이터를 연속으로 전송 시 핸드셰이크로 인한 동작 레이턴시가 증가하는 문제가 있다. 또한 각 모듈마다 Local Clock Generator를 필요로 하는 문제점이 있다.[8]

나. FIFO 기반의 GALS 시스템

그림 1(b)와 같이 FIFO-based 방식은 이종 클럭 도메인을 갖는 모듈 사이의 데이터 전송을 위해 FIFO를 사용한다. 이 방식은 FIFO을 사용하여 디자인이 간단하다. 또한 FIFO는 연속적으로 읽기/쓰기가 가능하므로 한 번에 많은 데이터를 연속적으로 보낼 수 있다. 반면에 FIFO로 인한 면적의 증가와 데이터가 FIFO를 통과해야 하기 때문에 Pausible Clocking 보다 큰 동작 지연시간을 갖는 문제가 있다. 하지만 FIFO-based 방식은 많은 데이터를 연속으로 전송 시 데이터를 FIFO에 연속 적으로 저장 후 연속적으로 읽기가 가능하여 첫 데이터를 전송할 때의 지연시간이 지난 후에는 데이터가 연속적으로 FIFO를 지나갈 수 있으므로 Pausible Clocking 방식보다 데이터 전송의 측면과 전체 동작 레이턴시 측면에서 유리하다. 따라서 본 논문은 GALS 방식을 구현하기 위해 이종 클럭 도메인 사이의 인터페이스 회로로 구현이 비교적 간단하고 연속적인 데이터 전송(Burst Transfer)에 유리한 FIFO-Based 방식을 적용하여 시스템을 구성하였다.

2. GALS 개요 및 솔루션 비교

GALDS 버스 구조는 그림 2와 같다. 마스터 IP들이 동시에 버스에 접근이 가능한 Segmented 버스 구조를 사용하였다. 그리고 IP 클럭과 버스 클럭이 다르더라도 안정된 데이터 전송을 위하여 GALS기법을 도입하였다. 이를 위해 FIFO-Based 방식을 적용한 Asynchronous

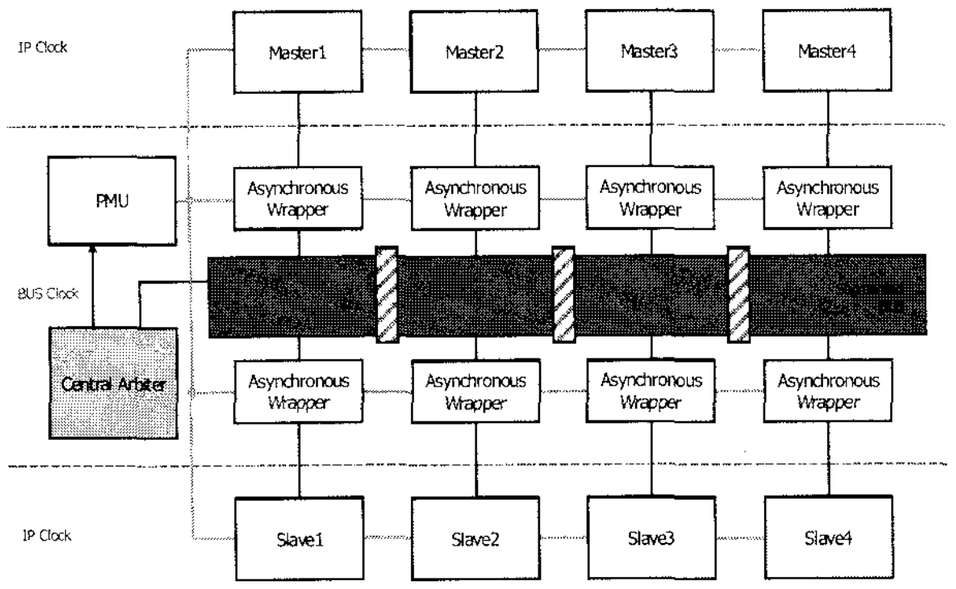


그림 2. 제안한 GALDS BUS의 전체 구조
Fig. 2. The proposed GALDS BUS Architecture.

Wrapper(AW)를 IP와 버스 사이에 연결하였다. 마지막으로 각 IP들의 동작 상태에 따라 IP에 공급되는 전력 및 주파수 스케일링 가능하도록 하는 DVFS 방식을 위한 PMU을 사용한다. 이 방식은 PMU에 의하여 시스템 동작 상태에 따라 몇 가지의 특정 동작주파수를 선택하여 각 IP에 공급하게 된다. 여기서 각 IP에 공급되는 동작 주파수는 버스 클럭의 정수배 1:1, 1:2, 1:4, 1:8로 고정되어 있다. 이와 같은 이유로 AW에는 버스 클럭의 정수배 주파수가 입력되므로 비동기 방식의 구현이 더욱 쉬워졌다. 기존의 비동기 시스템은 내부 Local Clock Generator를 사용하여 각각의 지역 동기 모듈이 어떤 클럭을 입력 받을지 예측을 할 수 없는 문제가 있어 이것은 시스템을 전체 설계하는데 걸림돌이 되어 왔기 때문이다. 하지만 본 논문에서 제안하는 버스 시스템은 시스템이 예측 가능한 클럭만을 모듈에 공급하므로 AW의 회로가 간단해지고 클럭의 변화에 따른 동작 레이턴시 문제를 줄일 수 있다. 따라서 본 논문에서는

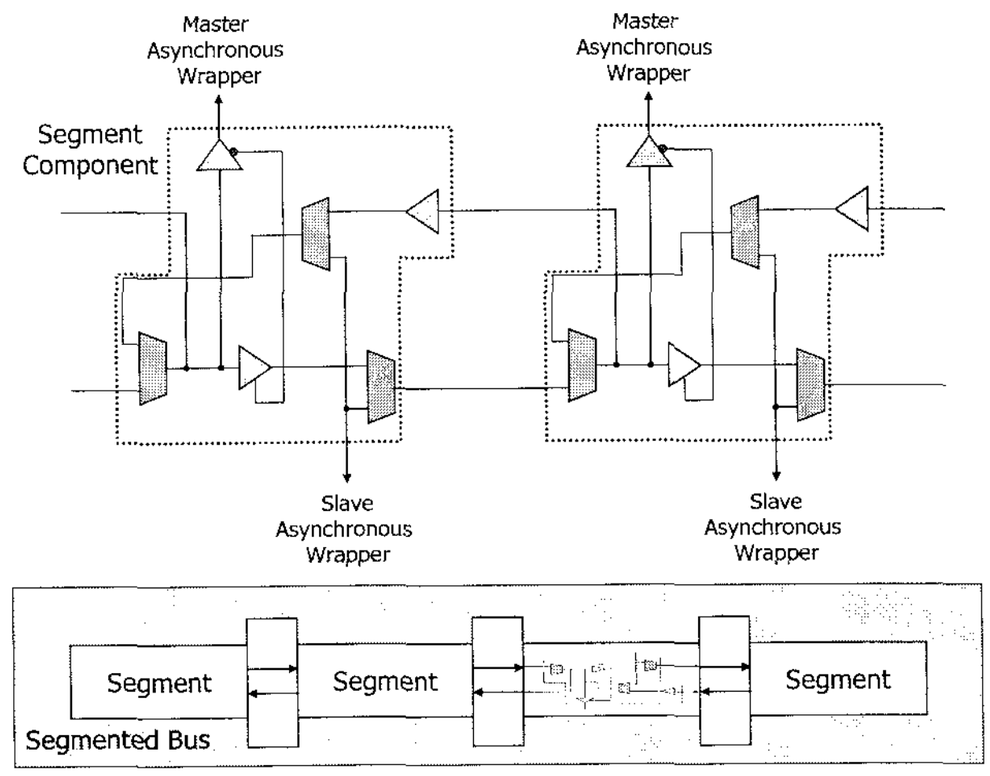
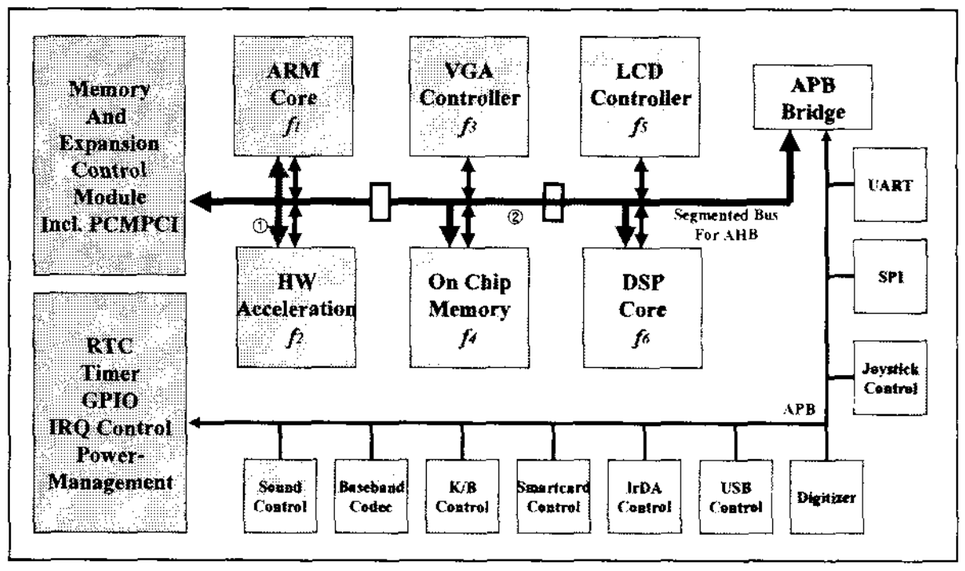


그림 3. 제안한 GALDS 버스의 분할 버스 구조
Fig. 3. A Segmented Bus channel in the proposed GALDS Bus.



Source : ARM Solutions Smart Phones and Communication

그림 4. 제안된 분할 버스의 다중 처리 동작의 예
Fig. 4. An example of Multi-Accessible Operation in the proposed GALDS Bus.

GALS 방식과 DVFS 방식을 결합한 GALDS 버스 구조를 설계하였다.[4]

또한 GALDS 버스는 Segment를 추가함으로써 버스 시스템의 다중처리가 가능함과 동시에 시스템의 확장이 용이하고 AMBA Specification Revision 2.0을 만족하여 기존의 ARM 기반의 플랫폼에 바로 적용이 가능하다.

위와 같이 제안하는 GALDS 버스는 DVFS 방식의 사용으로 각 IP에 시스템의 상황에 따라 서로 다른 클럭 주파수가 입력되어도, 각 IP마다 연결된 AW를 이용하여 비동기 방식으로 버스와 통신이 가능하다. 또한 버스 통신은 세그멘티드 버스 방식의 사용으로 다중처리가 가능한 장점이 있다. 즉, 제안하는 GALDS 버스는 비동기 방식으로 다중처리가 가능한 새로운 버스이다.

가. 제안한 GALDS 버스의 세그멘티드 구조

버스는 세그먼트 하나와 마스터 IP, 슬레이브 IP가 하나의 단위로 구성되어 그림 3과 같이 수평적으로 확장이 가능하도록 설계되었다.[5] 마스터 IP에서 나온 데이터는 서로 연결된 세그먼트 채널을 통과하여 목적지에 도착하게 된다. 이 구조는 IP들의 버스 채널 이용 요구에 따라 Central Arbiter에 의해 세그먼트 사이가 연결되거나 끊어지게 되어 동시에 통신을 하는 다른 마스터 IP의 데이터와 충돌을 막는다. 이와 같은 이유로 가능한 많은 IP가 동시에 버스에 접근하여 각자의 목적지로 데이터 전송이 가능하다. 4개의 세그먼트로 구성한 시스템의 경우 통신을 하는 IP들의 수에 따라 최대 4개의 마스터 IP가 동시에 통신이 가능한 구조로 설계하였다. 즉, 첫 번째 세그먼트와 연결된 마스터 IP의 데이터가 수평적으로 연결된 세그먼트들 지나가야 한다면 그 데이터가 통과하는 세그먼트들에 연결된 마스터 IP는 버스에 접근을 할 수가 없다. 그림 4은 SoC에 적용시 세그먼트 버스를 구성하는 세그먼트 채널 및 IP간의 통신이 이루어지는 것을 나타낸다. 기존의 공유 버스의 경우 ①과 ②의 과정이 동시에 일어날 수 없으나 세그먼트 구조의 경우 동시에 처리가 가능하므로 버스의 효율이 증가하게 된다.

나. 제안한 GALDS 버스의 전원 관리 유닛(PMU)

PMU는 저전력 시스템을 구현하기 위하여 저전력 시스템 기법 중의 하나인 DVFS를 적용하기 위해 설계된

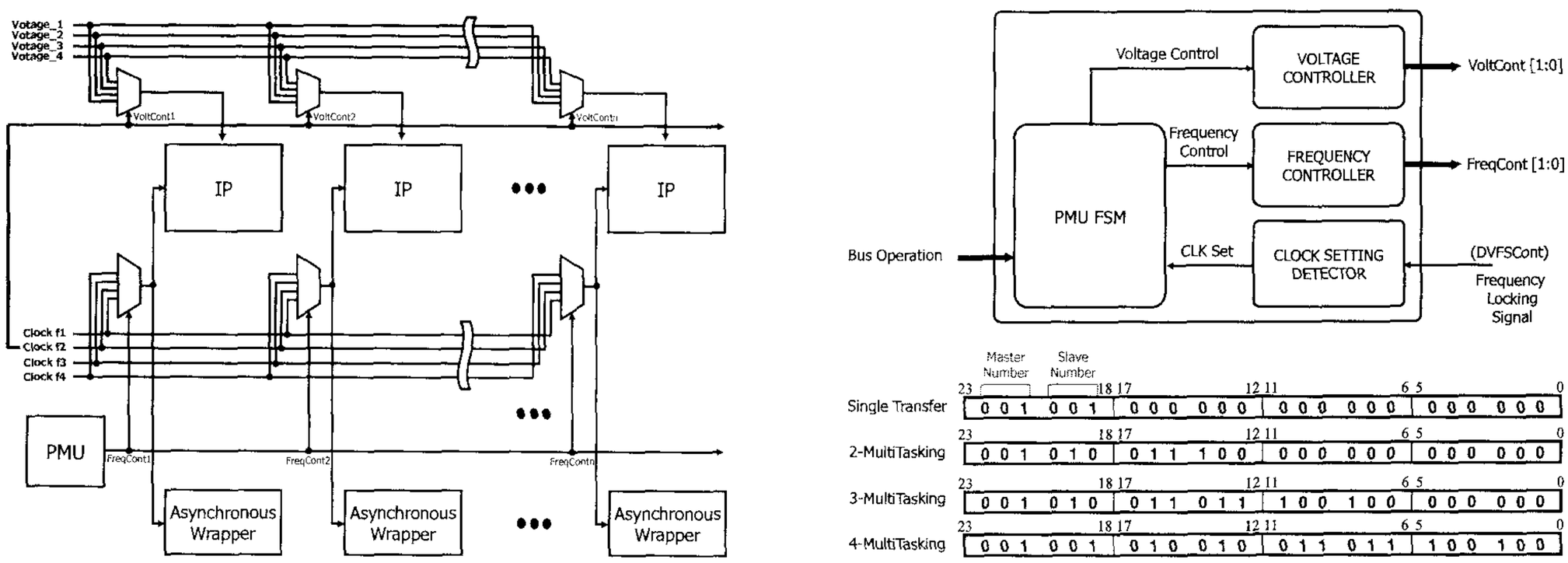


그림 5. 제안한 PMU의 구조 (a) 제안한 PMU의 주파수 선택 (b) 제안한 PMU의 구조 및 입력 신호
Fig. 5. The proposed PMU Structure (a) Clock frequency select using the proposed PMU (b) The proposed PMU Structure and Input Signal.

모듈이다. 저전력 시스템의 구현을 위하여 PMU는 시스템에서 각 IP들의 동작 정도에 따라 각 IP에 공급되는 전압과 클럭 주파수를 제어한다. 그림 5(b)에서와 같이 PMU는 시스템 내부의 IP 동작 정보(Bus Operation)를 Central Arbiter로부터 전달받아 시스템 동작에 따른 전압과 주파수 변경을 수행한다. 여기서 입력되는 클럭은 DLL을 통하여 입력되는 모든 클럭의 페이즈를 맞춰서 입력되므로 서로 다른 클럭 소스의 사용으로 인한 클럭의 매칭 문제를 해결할 수 있다.

Central Arbiter로부터 입력되는 그림 5(b)에서의 버스동작 신호는 현재 버스를 사용하고 있는 마스터 IP 및 슬레이브 IP의 정보가 들어있다. 따라서 이 신호를 이용하여 PMU의 FSM은 그림 5(a)와 같이 전체 시스템의 전압과 클럭 주파수를 선택 제어한다. 구조적으로 PMU는 다른 디바이스를 스위칭하여 외부의 파워를 컨트롤하는 것이 가능하다. 이와 더불어 동작하지 않는 IP는 일정 시간 후에 자동으로 끄는 logic enable 기능을 가진다. 또한 동시에 IP 클럭에 다른 클럭 공급원을 스위칭하거나 IP 클럭과 PLL 또는 외부 입력 주파수를 끄는 역할을 한다.

다. 제안한 GALDS 버스의 Asynchronous Wrapper

Asynchronous Wrapper (AW)의 구조는 그림 5와 같다. 앞서 언급한 것과 같이 AW는 GALS 시스템의 구성에 가장 핵심이 되는 요소이다.[9] IP와 버스 사이의 서로 다른 이종 클럭 도메인 간에서 데이터를 안정되게 전송할 수 있게 한다. 이를 위하여 AW 내부는 Asynchronous FIFO 및 FIFO의 읽기/쓰기 제어 신호를 출력하는 로직(IP to Wrapper FSM, Wrapper to IP FSM)으로 구성된다. 또한 DVFS 방식을 적용한 본 시스템의 특성상 IP의 동작 주파수가 변하더라도 데이터 전송이 가능해야한다. 그렇게 때문에 AW는 GALDS 버스 시스템에서 동작 주파수가 변하여도 정상적으로 데이터 전송이 가능하도록 설계하였다. 물론 동작 주파수의 변화는 버스 클럭의 정수배로 변화되어야 한다. 이러한 기능을 구현하기 위하여 AW는 읽기 클럭과 쓰기 클럭 비율의 변화에도 동작이 가능하도록 설계된 Asynchronous FIFO를 사용한다. 이 FIFO는 읽기 클럭와 쓰기 클럭의 비가 1:1, 1:2, 1:3, 1:4과 같은 정수배의 클럭 변화에 올바르게 동작하여, 시스템의 동작 상황에 따른 클럭 변화시 데이터가 올바르게 전송될 수 있게 한다. 또한 본 버스 시스템은 SoC에서 가장 많이 사용

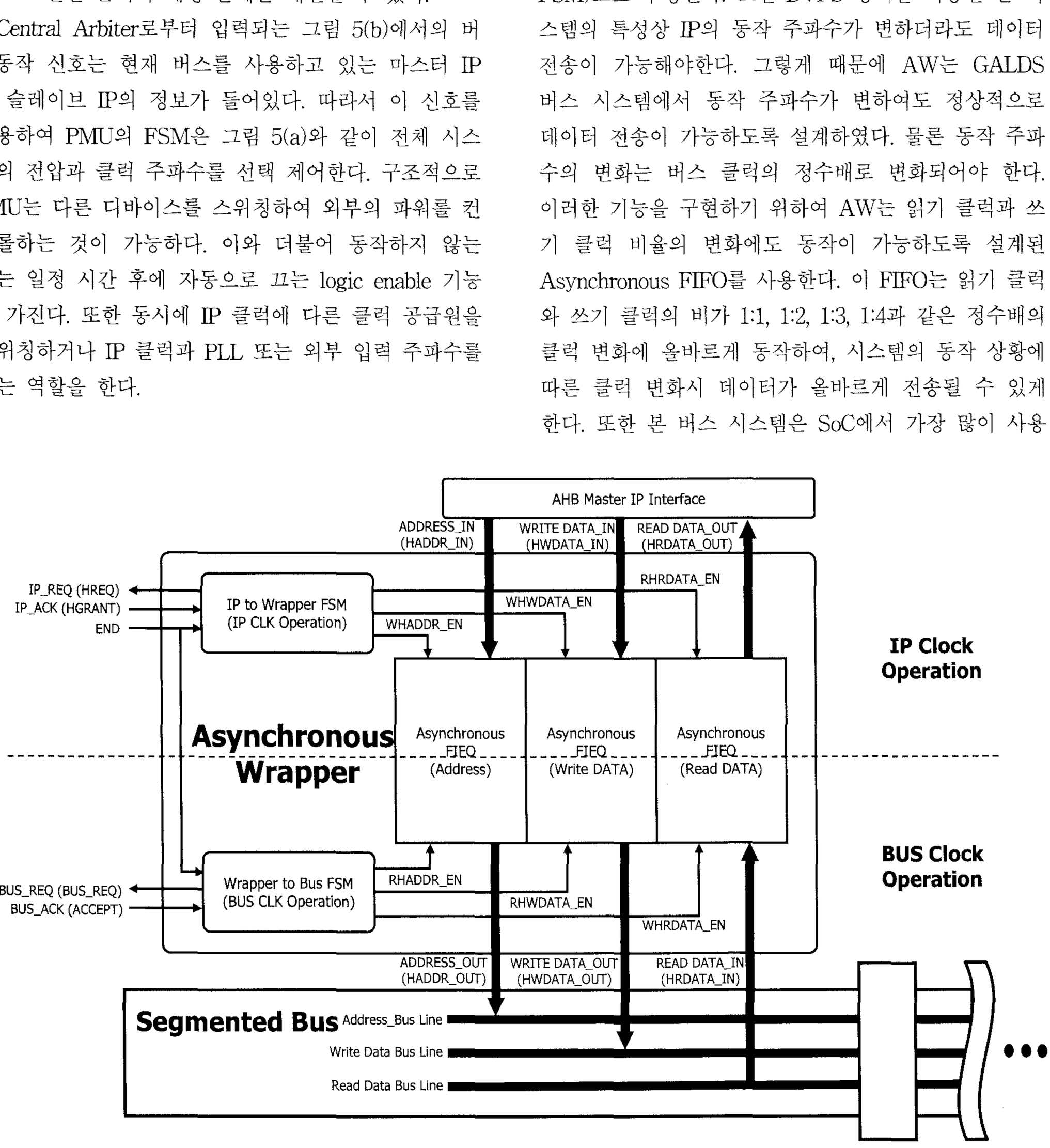
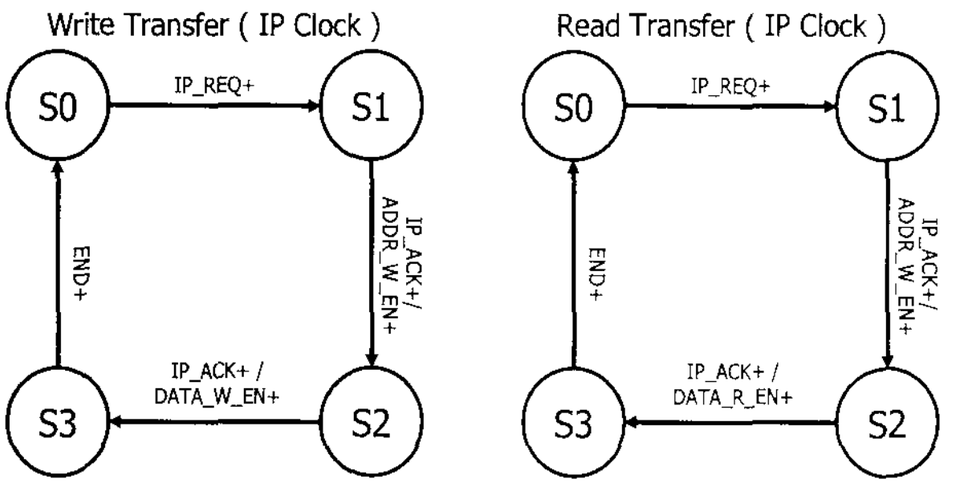
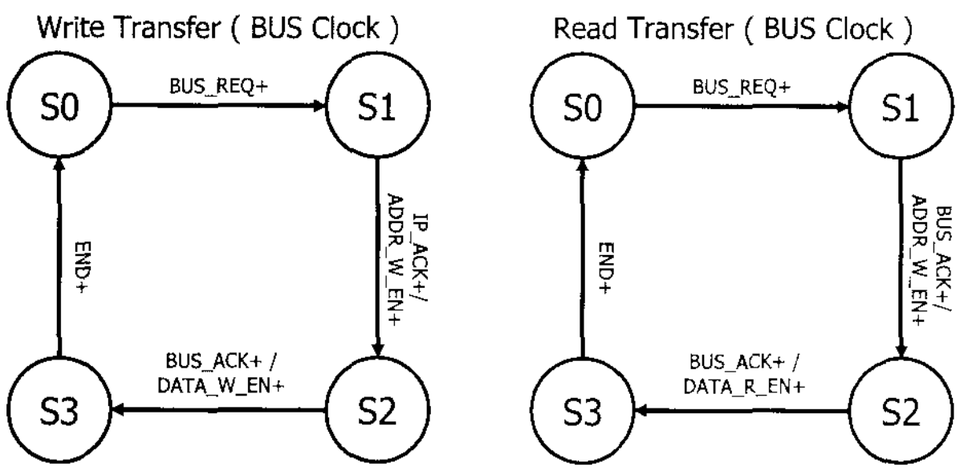


그림 6. 제안한 GALDS 버스의 Asynchronous Wrapper(AW) 구조
Fig. 6. Asynchronous Wrapper(AW) for the proposed GALDS Bus.



(a)



(b)

그림 7. Asynchronous FIFO를 제어하기 위한 FSM
Fig. 7. FSM for Asynchronous FIFO control.

에서 널리 사용 가능하다.[6] 이와 같은 이유로 Asynchronous FIFO는 버스에 주소를 보내기 위한 FIFO 1개, 데이터의 읽기/쓰기 동작을 위한 FIFO 2개, 총 3개로 구성되어있다. 마지막으로 그림 6과 같이 IP 클럭으로 동작하는 FIFO 제어신호 FSM과 버스 클럭으로 동작하는 FIFO 제어신호 FSM으로 정확히 분리하여 FIFO 제어신호의 올바른 출력이 가능하게 하였다.

3. GALDS 버스 시스템의 전체 동작

그림 8와 같이 IP에서 발생된 요청(①REQ)에 의해 Central Arbiter는 버스의 채널 상태 및 마스터 IP의 우선순위에 따라 통신을 허가한다(ⓑACCEPT). 시스템의 통신 이전에 각 IP의 전압과 주파수는 이미 버스에서 PMU로 들어오는 컨트롤 신호에 의해 설정되어 있다. 만약 외부로부터 공급되는 전압과 주파수가 안정화되지 않았다면 PMU는 Central Arbiter에 통신 대기 신호를 보내 데이터의 손실을 막는다. 전압과 주파수의 스케일링에 따라 전송되는 데이터는 AW와 세그멘티드 버스를 거쳐 목적지에 데이터를 전송하여 통신을 완료한다.

그림 8에서와 같이 우선 마스터 IP와 마스터 AW 사이에서의 데이터 전송을 위하여 IP클럭을 이용하여 마스터 IP의 데이터를 AW로 보내게된다. 마스터 IP는 REQ신호를 마스터 WA로 보내고 마스터 AW로부터 GRANT신호를 받아 데이터를 보내어 FIFO에 저장한다. 여기서의 과정은 AMBA Specification 2.0을 정확히 준수한다. 그 다음 마스터 AW는 버스 채널을 통해 버스 클럭을 이용하여 상대편의 슬레이브 AW로 데이터를 전송하게 된다. 여기서 마스터 AW는 버스의 중재

되고 있는 AMBA Specification 2.0을 적용하여 기존의 SoC 플랫폼에 쉽게 적용을 할 수 있다. 따라서 Wrapper와 IP들간의 데이터 전송은 AMBA의 규약에 맞춰서 하게 된다. 여기서 주의 할 점은 마스터 IP와 AW사이, 슬레이브 IP와 AW 사이에서 AMBA 규격을 이용한다는 것 이다. 그리고 Master AW와 Slave AW 사이의 통신은 본 논문에서 자체적으로 도입한 세그멘티드 버스를 통해 통신을 한다. 이와 같이 IP와의 연결에 AMBA 규격을 사용함으로서 ARM 기반의 시스템

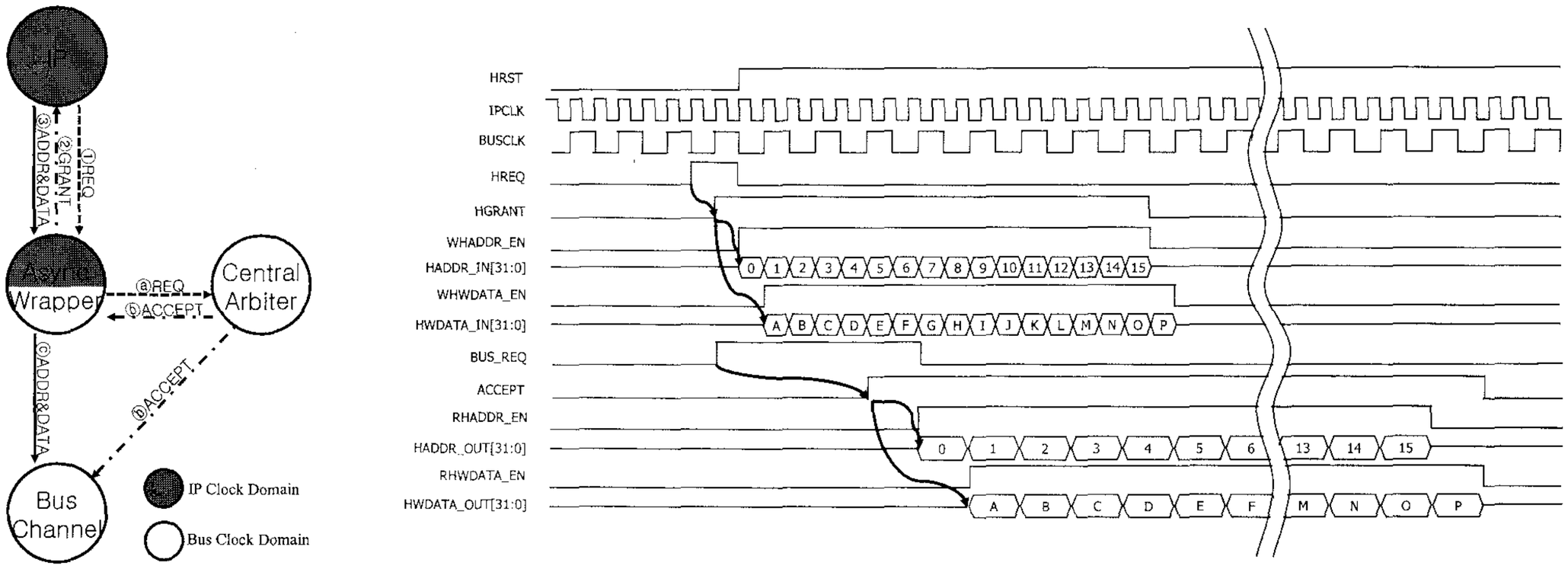
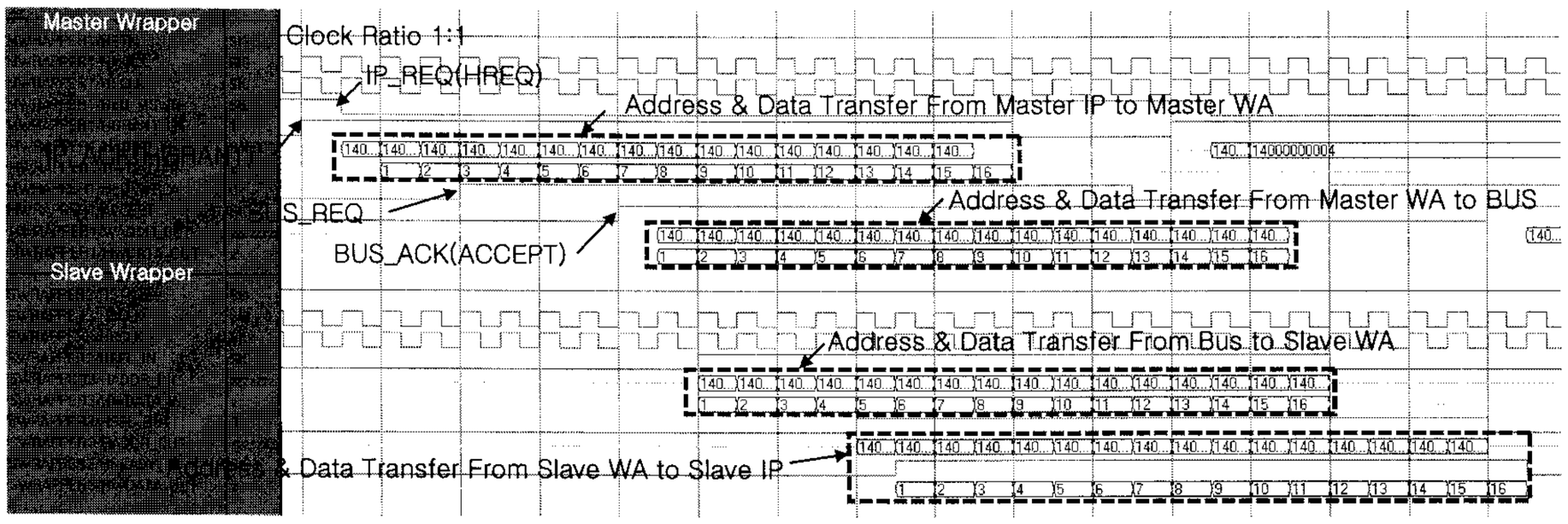
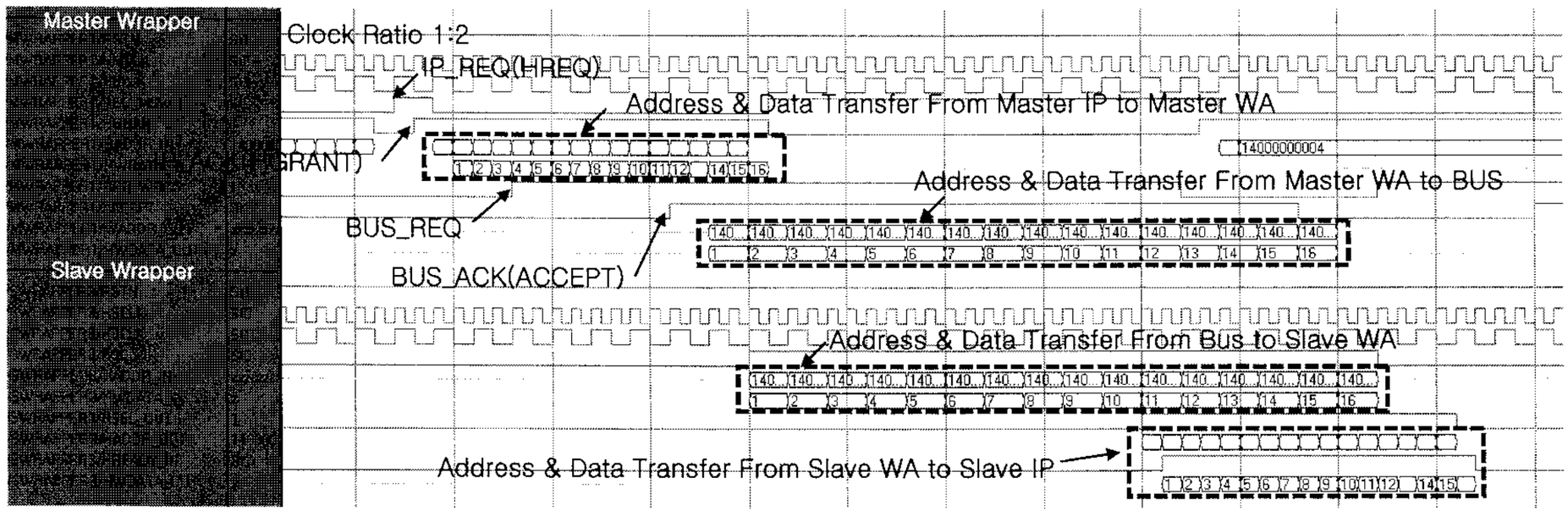


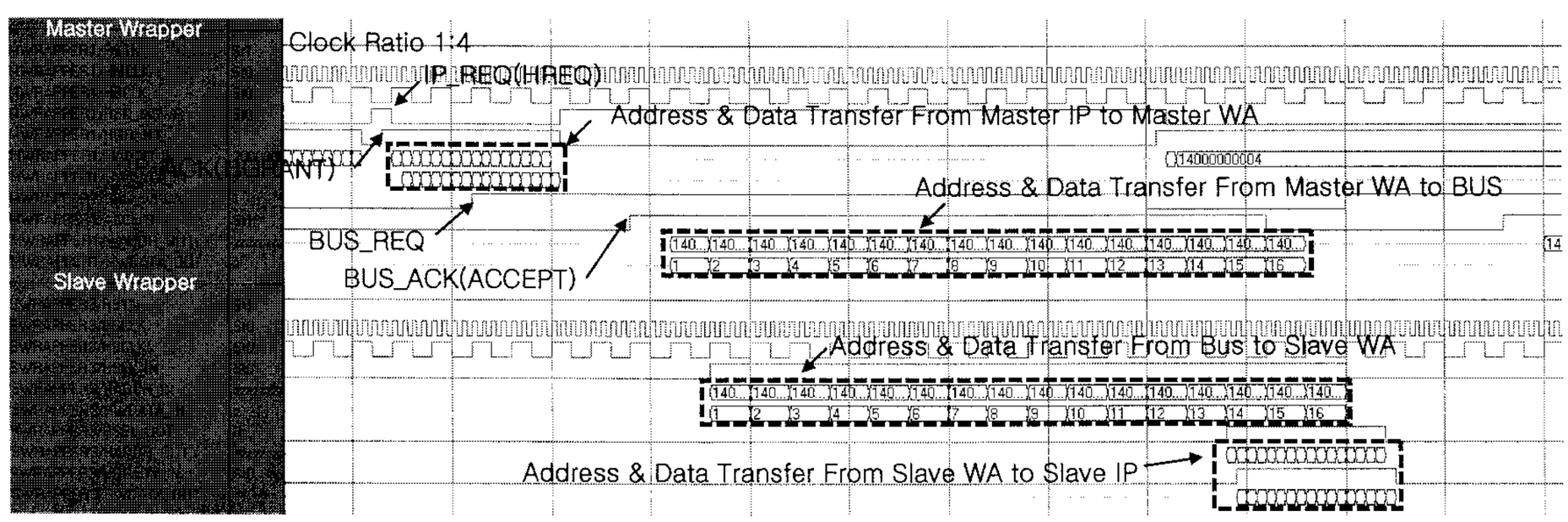
그림 8. 제안한 GALDS 버스의 동작 및 타이밍 그림
Fig. 8. An example of the proposed GALDS BUS Operation and Timing Diagram.



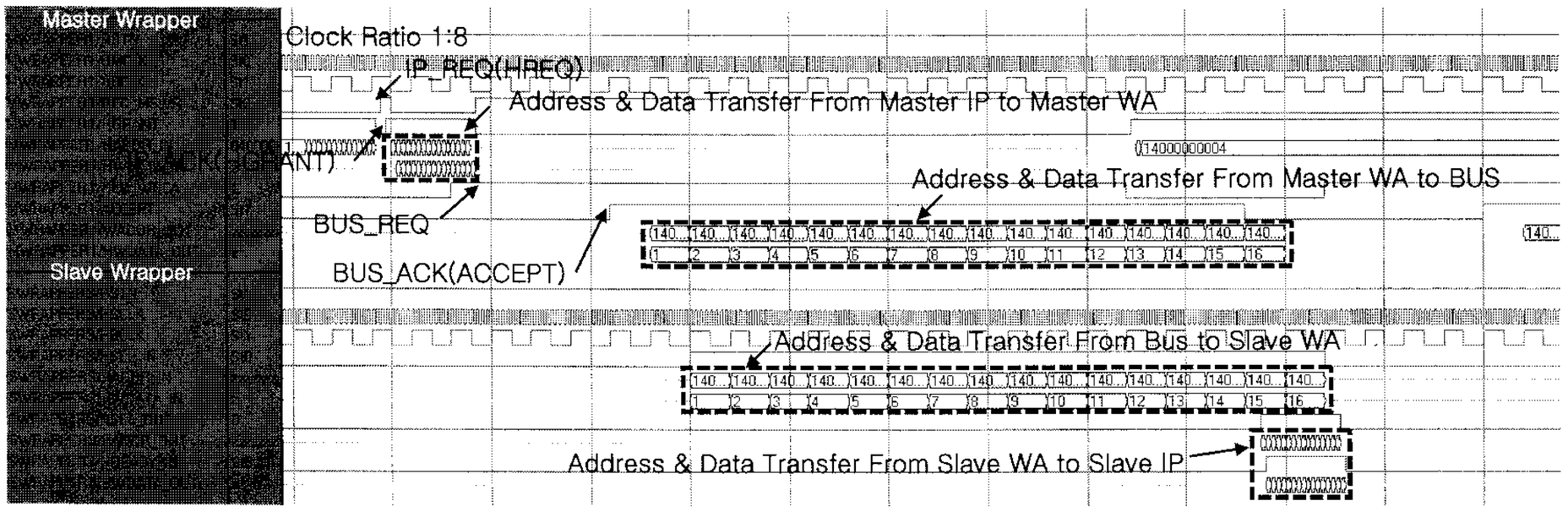
(a) 1:1 Frequency Ratio (Bus Write Operation in Master and Slave Wrapper)



(b) 1:2 Frequency Ratio (Bus Write Operation in Master and Slave Wrapper)



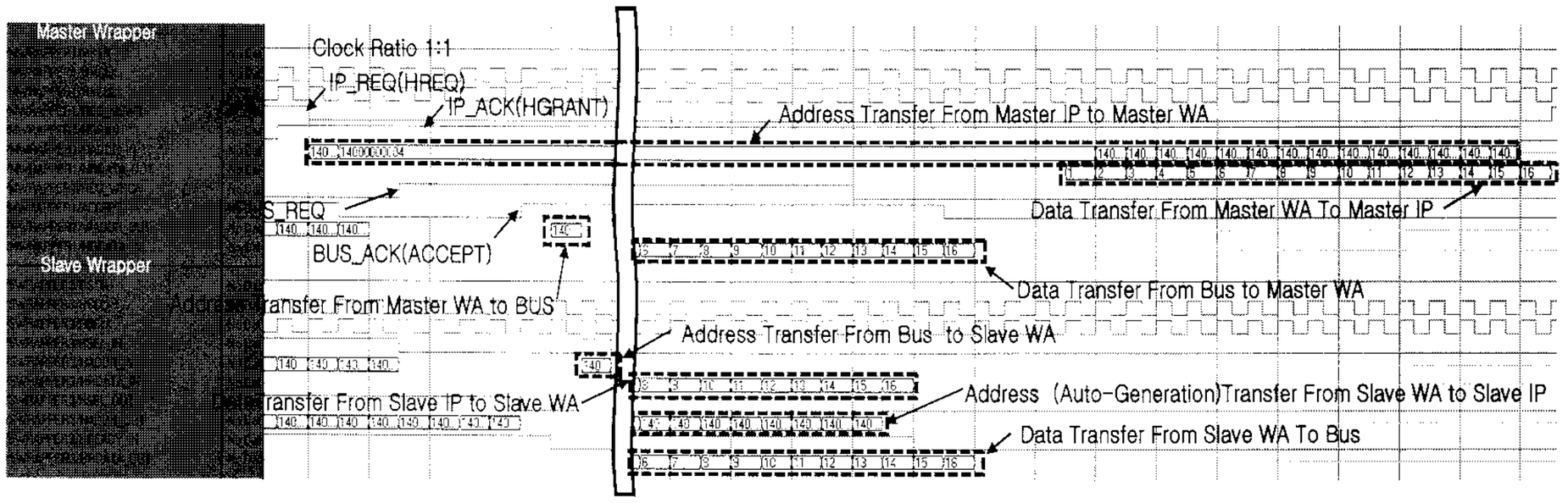
(c) 1:4 Frequency Ratio (Bus Write Operation in Master and Slave Wrapper)



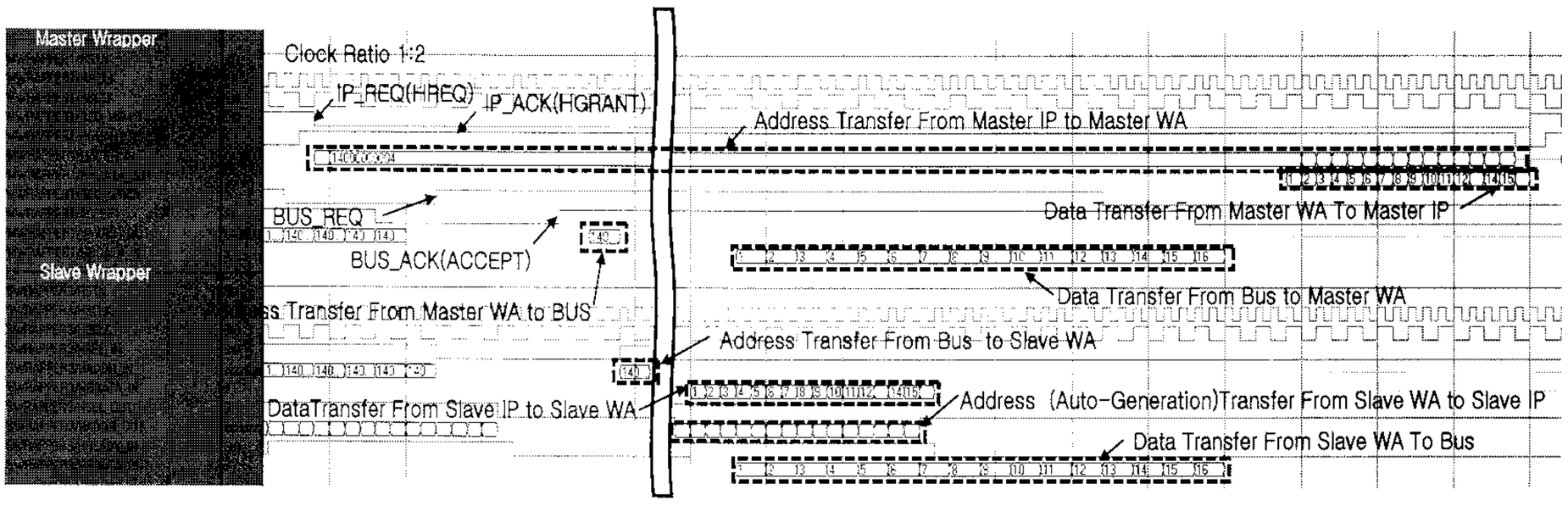
(d) 1:8 Frequency Ratio (Bus Write Operation in Master and Slave Wrapper)

그림 9. 제안한 GALDS 버스 쓰기 동작 시의 IP와 AW사이의 시뮬레이션 파형

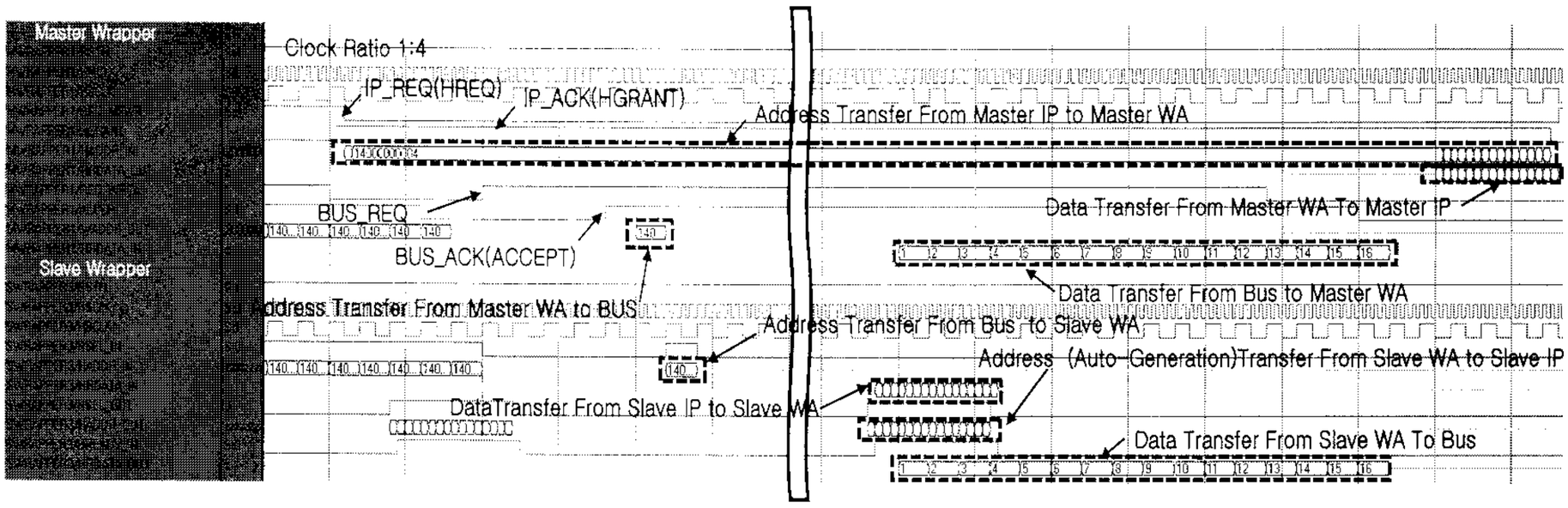
Fig. 9. Simultaion Waveform between IP and AW for the proposed GALDS Bus Write Operation.



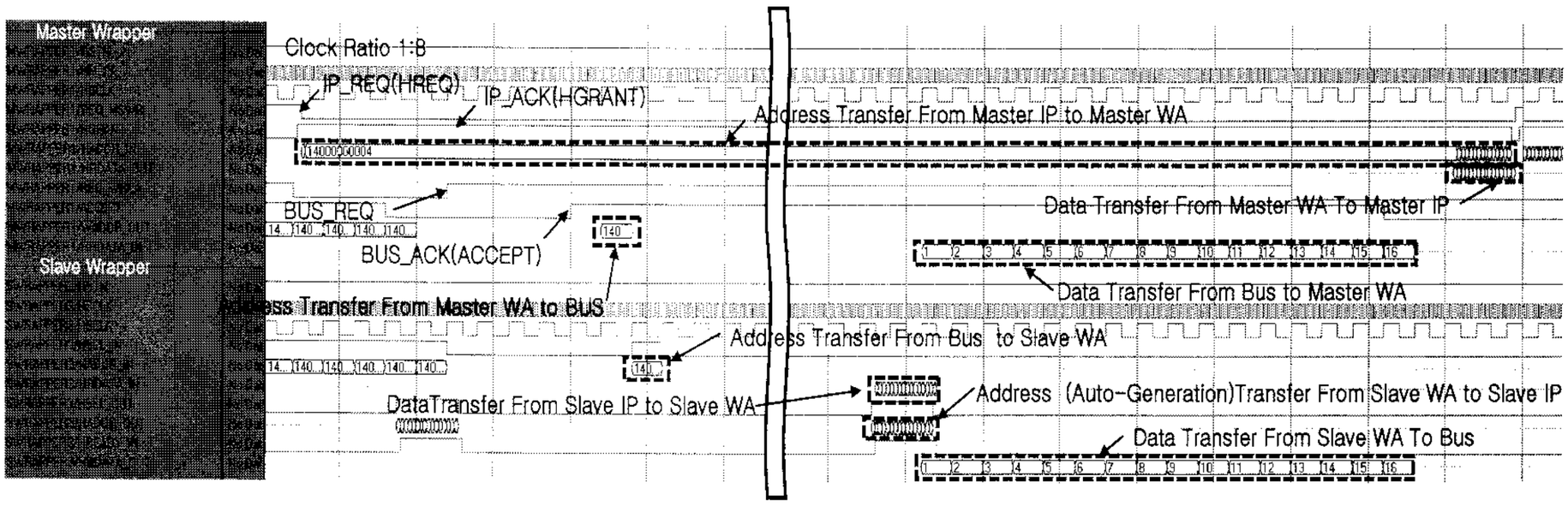
(a) 1:1 Frequency Ratio (Bus Read Operation in Master and Slave Wrapper)



(b) 1:2 Frequency Ratio (Bus Read Operation in Master and Slave Wrapper)



(c) 1:4 Frequency Ratio (Bus Read Operation in Master and Slave Wrapper)



(d) 1:8 Frequency Ratio (Bus Read Operation in Master and Slave Wrapper)

그림 10. 제안한 GALDS 버스 쓰기 동작시 IP와 AW사이의 시뮬레이션 파형
Fig 10. Simultaion Waveform between IP and AW for the proposed GALDS Bus Read Operation.

역할 및 버스채널을 관리하는 Central Aribiter로 REQ신호를 보내 버스 사용을 요청하고 버스 ACCEPT신호를 받은 후 버스 채널로 데이터를 보내게 된다. 위와 마찬가지로 버스 채널로부터 데이터를 받은 슬레이브 AW는 슬레이브 IP와 AMBA Specification 2.0 프로토콜을 이용하여 통신을 하여 데이터 전송을 끝내게 된다. 여기서 마스터 IP에서 보내는 데이터가 모두 마스터 AW로 전송 된 후에 마스터 AW에서 버스 채널로 데이터를 보낸다면 데이터전송 레이턴시가 커진다. 따라서 본 버스에서는 FIFO에 모든 데이터가 들어오기 전에 버스채널로 전송을 시작하도록 설계하여 레이턴시를 줄일 수 있도록 하였다.

Ⅲ. 실 험

제안된 버스 구조는 AMBA AHB Interface에 데이터 신호들을 입력하여 시뮬레이션을 하고 검증하였다. 각각의 마스터 IP가 버스에 대한 읽기 쓰기 동작을 반복하여 버스에 요청하여 IP와 버스 사이에서 데이터가 올바르게 전송되는 가를 확인하고 IP 입력 주파수의 변화에도 데이터가 손실 없이 전송되는 가를 확인하였다.

(a) 1:1 Frequency Ratio (Bus Read Operation in Master and Slave Wrapper)전체 시스템은 마스터 IP 4개, 슬레이브 IP 4개로 구성하였다. Central Arbiter의 중재방식은 Weighted Round Robin 방식을 사용하여 시스템에서 사용 빈도가 높은 IP가 버스 접근이 쉽도록 하였다. Central Arbiter는 4개의 마스터 IP의 버스 접근 요구에 따라 앞서 설명한 중재방식과 버스와의 다중 접속이 가능한가에 따라 버스에 접근 가능한 마스터 IP가 선택된다.

IP들의 동작 상태에 따라 PMU에서 선택이 가능한 IP클럭 주파수는 버스 클럭의 1 배, 2 배, 4 배 및 8 배에 해당하는 동작주파수가 입력되도록 하였다. 또한 AMBA Specification에 따라 마스터 IP의 데이터 전송 모드는 32bit로 하였고 16 Increments Burst 동작을 수행하도록 하였다. 그림 9와 그림 10의 시뮬레이션은 그림 4의 첫 번째 그림과 같은 Master Wrapper1과 Slave Wrapper3사이에서의 전송 예이다. IP 클럭이 버스 클럭의 정수배로 변화시 IP와 버스 사이의 통신이 올바르게 되는 것을 보여준다. 그림 9는 마스터 IP에서 슬레이브 IP로의 데이터 쓰기 동작을 시뮬레이션 한 그림이다. 버스 동작주파수의 배수로 IP 클럭이 변화해도 동작이 올바르게 되는 것을 알 수 있다. 그림 10은 마스터 IP가 슬레이브 IP의 데이터를 읽어오는 데이터 읽기 동작을 시뮬레이션 한 그림이다. 쓰기 동작과는 다르게 읽기 동작은 슬레이브에서 데이터를 받아와야 하므로 쓰기 동작 보다 많은 시간이 필요하다. 마지막으로 표 2는 버스의 전송 동작이 시작 될 때부터 쓰기 동작시 마스터 IP에서 슬레이브 IP까지 쓰기 데이터가 도착하는데 걸린 시간이고 읽기 동작시 마스터 IP로 읽기 데이터가 도착하는데 걸리는 시간을 나타낸 것 이다.

표 1. 제안한 GALDS 버스의 동작 스펙
Table 1. The proposed GALDS BUS Specification.

Description	Specification
Tasking Numbers	1 ≤ Current Tasking ≤ 4 (Max 16)
Master Numbers	4 (Max. 16)
Slave Numbers	4 (Max. 16)
Frequency Ratio (BUS/IP)	1/1, 1/2, 1/4, 1/8

표 2. 버스클럭과 IP클럭비에 따른 데이터 읽기/쓰기에 필요한 클럭 지연시간
Table 2. Data Write and Read Clock Latency in accordance with a clock ratio between Bus and IP.

Clock Ratio (Bus Clock : IP Clock)	Data Write (Clock)	Data Read (Clock)
1:1	13	35
1:2	18	39
1:4	20	42
1:8	22	44

Ⅳ. 결 론

본 논문에서는 SoC시스템에서 널리 쓰이고 있는 공유(Shared) 버스에 다중 접속 방식, GALS와 DVFS기법을 적용한 새로운 시스템 버스인 GALDS 버스를 제안하였다. 제안된 버스 구조는 여러 개의 서로 다른 동작 주파수를 갖는 IP들이 동시에 버스 다중 접속이 가능하여 기존의 공유 버스 방식에 비해 효율적으로 버스를 사용할 수 있다. 시스템의 사용량에 따른 IP와 버스 사이의 클럭 비 변화에도 안정된 데이터 전송이 가능하여 저전력 시스템을 위해 적합한 버스 구조이다. 이는 GALS기법을 FIFO-based 방식으로 구현하여 이종 클

럭 도메인 간의 데이터 전송 문제를 해결하였기 때문이다. 이 버스 구조는 기존의 AMBA 버스 기반의 SoC 플랫폼에 바로 적용 할 수 있어 기존의 시스템으로의 이식이 가능하다. 이와 같이 본 논문에서는 새로운 저전력 SoC 시스템을 위한 GALDS 버스를 제안한다.

참 고 문 헌

[1] Krstic, M.; Grass, E.; Gurkaynak, F.K.; Vivet, P, "Globally Asynchronous, Locally Synchronous Circuits: Overview and Outlook", IEEE Design & Test of Computers, vol 24, no 5, pp430 - 441, Sept.-Oct. 2007.

[2] P. Teehan, M. Greenstreet, and G. Lemieux, "A Survey and Taxonomy of GALS Design Styles", IEEE Design & Test of Computers, Volume 24, Issue 5, pp418 - 428, Sept.-Oct. 2007.

[3] T. Seceleanu, J. Plosila, P. Liljeberg, "On-Chip Segmented Bus: A Self-Timed Approach", 15th Annual IEEE International ASIC/SOC Conference, pp216 - 220, 25-28 Sept. 2002.

[4] G. Magkilis, G. Semeraro, D. Albonesi, S. Dropsho, S. Dwarkadas, and M. Scott, "Dynamic Frequency and Voltage Scaling for a Multiple-clock-domain Microprocessor," IEEE Micro, vol. 23, no. 6, pp. 62-68, Nov. 2003.

[5] Atanu Chattopadhyay and Zeljko Zilic, "GALDS:A Complete Framework for Designing Multiclock ASICs and SoCs," IEEE Trans. on VLSI Syst., vol. 13, no 6, pp641 - 654, June 2005.

[6] Kulmala, A., Hamalainen, T.D., Hannikainen, M., "Comparison of GALS and Synchronous Architectures with MPEG-4 Video Encoder on Multiprocessor System-on-Chip FPGA", Digital System Design: Architectures, Methods and Tools, 2006. DSD 2006. 9th EUROMICRO Conference, pp83 - 88, 2006.

[7] Kenneth Y. Yun, Ryan P. Donohue, "Pausible Clocking: A First Step Toward Heterogeneous Systems" Computer Design: VLSI in Computers and Processors, 1996. ICCD '96, pp118 - 123, 7-9 Oct. 1996.

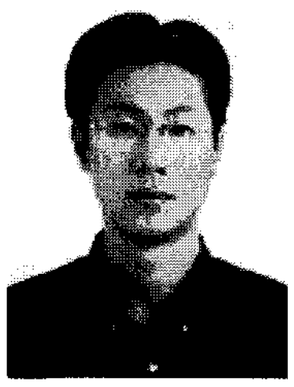
[8] M. Krstic, E. Grass, C. Stabl and M.Piz, "System integration by request-driven GALS design", IEE Proc.-comput. digit. Tech., Vol. 153, No.5, pp362 - 372, Sep 2006.

[9] A. R. Ravi, "Globally-Asynchronous, Locally-Synchronous Wrapper Configurations for Point-to-Point and Multi-Point Dara Communication," B.E. Banfalore University, 2001.

[10] ARM Ltd. AMBATM Specification, Rev 2.0. Reference Nr ARM IHI0011A, May 13, 1999.

저 자 소 개

최 창 원(학생회원)
2006년 숭실대학교 정보통신전자공학부 학사 졸업.
2007년 숭실대학교 전자공학과 석사 과정.
<주관심분야 : Asynchronous System, On-chip Bus, Mixed Mode Design>



신 현 출(정회원)
1997년 포항공과대학교 학사 졸업.
1999년 포항공과대학교 석사 졸업.
2004년 포항공과대학교 박사 졸업.
2004년~2007년 Johns Hopkins 의대 박사 후 연구원
2007년~현재 숭실대학교 정보통신전자공학부 전임강사
<주관심분야 : 뇌-기계 접속 시스템, 생체신호처리, 뇌 혈관 영상 시스템>

위 재 경(정회원)-교신저자
1988년 연세대학교 물리학과 학사 졸업.
1990년 서울대학교 물리학과 석사 졸업.
1998년 서울대학교 전자공학과 박사 졸업.
1990년~2002년 하이닉스 메모리 연구소 근무
2002년~2004년 한림대학교 정보통신공학부 조교수
2004년~현재 숭실대학교 정보통신전자공학부 부교수
<주관심분야 : System-in-Package 설계 및 고속 SoC, high speed I/O interface, DLL/PLL, Mixed Mode design>