Design of Interface Bridge in IP-based SOC

정휘성, 양훈모, 이문기
연세대학교 전기전자공학과
Tel : 02-2123-4731 / Fax : 02-312-4584

Hwi-Sung Jung, Hun-Mo Yang, Moon-Key Lee
Electrical and Electronic Engineering, Yonsei University
E-mail : hsjung@spark.yonsei.ac.kr

Abstract

As microprocessor and SOC (System On a Chip) performance moves into the GHz speed, the high-speed asynchronous design is becoming challenge due to the disadvantageous power and speed aspects in synchronous designs. The next generation on-chip systems will consist of multiple independently synchronous modules and asynchronous modules for higher performance, so the interface module for data transfer between multiple clocked IPs is designed with Xilinx FPGA and simulated with RISC microprocessor.

I. 서론

반도체 기술이 VDSM (Very Deep Sub-Micron)으로 발전함에 따라 집적도가 높아지고 동작 클럭 주파수가 높아짐에 따라 마이크로프로세서, DSP 및 IP (Intellectual Property) 와 같은 여러 가지 기능 블록들을 하나의 칩으로 구현하고자 하는 SOC (System On a Chip)설계가 늘어가고 있다. 즉, SOC는 로직, 메모리, 입출력 장치, 통신용 기능 블록, 멀티미디어 기능 블록 등 이전에는 분리된 칩으로 구현되던 기능 블록들을 단일 칩 안에 모듈 형태로 집적함으로써 시스템을 구성하는 방법이다. 이 기술은 각종 디지털 가전기기나 휴대용 통신기기, 소형 컴퓨터에서 대형 컴퓨터에 이르기까지 모든 분야에 적용될 수 있으며 앞으로의 미래 정보기기의 형태를 크게 바꾸어 놓을 것이다.

이러한 SOC 설계 방법을 적용하기 위해서는 무엇보다도 각 블록들간을 안정적인 통신 프로토콜로 연결하는 IP 기반 솔루션이 요구된다. 그리고 SOC를 구현할 때 전체 시스템에 글로벌 클럭을 사용하면 서로 다른 동작 클럭을 사용하는 서브 블록들간의 인터페이스가 복잡해지며 클럭 분배 문제와 클럭에 따른 전력 소모와 클럭 스큐, 클럭 노이즈 문제가 발생하게 된다. 이러한 문제의 해결 방안으로 저전력에 대한 관심과 더불어 비동기식 시스템에 대한 관심이 동기식 시스템의 대안으로서 증가하게 되었다. 즉 각 서브 블록들간을 비동기식으로 인터페이스를 구현하여 클럭을 사용하지 않고 핸드셰이크방식의 프로토콜을 사용하여 데이터를 전송하는 것이다. 비동기식 회로 설계는 동기식 회로 설계에 비해 전력 소모 및 속도 측면에서 장점을 가지고 있다 [1]. 따라서 현재 시스템 수준과 회로 수준의 설계에서 비동기식 회로가 두각 되고 있다. 특히 클럭이 없는 비동기식 회로에서의 시스템 타이밍은 각각의 회로를 구성하는 요소들에 의해 결정되며, 따라서 차세대 SOC는 성능을 향상 시키기 위하여 서로 다른 클럭 주파수를 가진 여러 가지 기능 블록들과

비동기식의 블록들로 구성될 수가 있다. 이러한 블록들로 구성된 SOC내에서 고속으로 데이터를 전송하기 위해서는 신뢰성 높은 데이터 통신 구조가 필요하다.

본 연구에서는 서로 다른 클럭 주파수를 가진 Mixed-Clock 시스템에서 FIFO를 이용하여 IP 블록들을 인터페이스 할 수 있는 회로에 대해서 알아보기로 한다. 시스템에서 서로 다른 클럭을 가진 IP간의 데이터 전송은 핸드셰이크 프로토콜을 사용하여 처리하며 본 연구에서는 Xilinx FPGA를 사용하여 인터페이스 회로를 구현한 후 RISC 마이크로프로세서를 사용하여 시스템을 구성한 후 서로 다른 클럭을 가진 동기식 시스템사이에서 데이터 전송이 이루어 지는 것을 확인하였다.

II장에서는 universal bridge에 대해 설명하였고 III장에서는 이를 이용한 비동기식으로 데이터를 전달할 수 있는 시스템 구축에 대하여 설명하였다. 마지막으로 IV장의 결론으로서 본 보고서를 맺는다

II. Universal bridge

IP간의 인터페이스는 universal bridge를 통하여 이루어 지며 서로 다른 클럭을 사용하는 IP간의 데이터 통신을 가능하도록 universal bridge 역시 서로 다른 클럭 주파수로 동작하게끔 설계하였다 [2,3]. 그림 1은 인터페이스 회로와 IP를 이용한 시스템 구성의 한 예로서 블록 다이어그램을 보여주고 있다. 동작 설명은 다음과 같다. 먼저

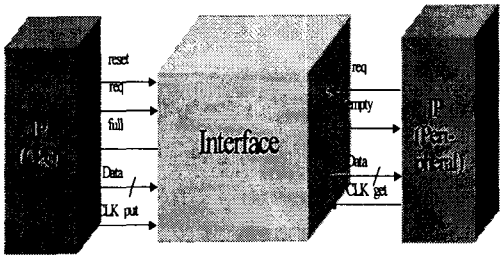

그림 1. IP-based 인터페이스 시스템 구성도

CPU쪽에서 주변 블록 (Peripheral) 방향으로 데이터를 전달하기 위하여 universal bridge 회로에게 request (=1) 신호를 보낸다. 이때 universal bridge 회로는 내부 레지스터를 체크하여 포화상태인지 아닌지를 확인한다. 만약 bridge가 포화상태가 아니면 인터페이스 회로는 CPU 방향으로 full (=0) 신호를 보내 bridge 쪽으로 데이터 전송을 허락한다. 그 후 CPU는 full 신호가 1 이 되기 전까지 데이터를 bridge쪽으로 전송할 수가 있다. 이때 bridge 내의 래치는 CPU 클럭에 동기시켜 request 신호가 low 될 때까지 CPU로부터 전송된 데이터를 저장하게 된다. 주변 회로 측면을 살펴보면 만약 주변회로가 bridge로부터 데이터를 받기 위해 bridge로 request (=1) 신호를 보내면 마찬가지로 bridge는 래치의 포화여부를 체크한다. 체크한 후 bridge가 비어있지 않으면 주변회로 쪽으로 empty (=0) 신호를 보내 데이터 전송을 허가한다. 역시 주변회로는 empty 신호가 1이 되기 전까지 데이터를 받을 수 있다. 이때에는 주변회로의 클럭에 동기시켜 bridge로부터 데이터를 가져가게 된다. 이에 대한 타이밍 다이어그램은 그림 2와 같다.

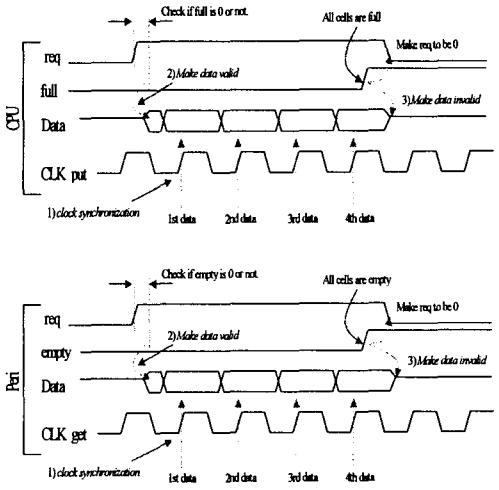

그림 2. Universal bridge 타이밍 다이어그램

그림 3은 Universal bridge 회로의 내부 구조를 보여주고 있으며 크게 데이터를 저장하는 레지스터와

핸드셰이크 컨트롤 블록, 그리고 레지스터 디텍터 부분으로 나뉘어져 있다. 데이터가 저장되는 레지스터는 그림에서 보듯이 1-2-3-4 번 방향으로 데이터가 저장되게 된다. 즉 데이터를 보내는 Sender 쪽에서 클럭에 동기되어 데이터를 보내게 되면 위의 순서로 저장이 되며 마찬가지로 데이터를 받는 Receiver 측면에서는 반대방향으로 Receiver 클럭에 동기되어 데이터가 나가게 된다.

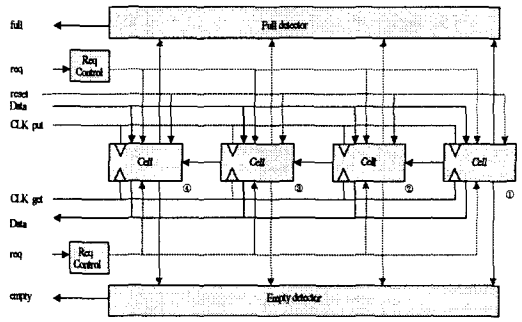

그림 3. universal bridge의 블록 다이어그램

내부구조의 동작은 토큰-링(Token-Ring) 방식을 사용하였다. 이는 write-flag 및 read-flag를 확인하여 링 형식으로 데이터가 저장되는 레지스터의 순서가 순차적으로 돌아온다는 것이다 [4]. 그림 4는 저장되는 데이터 레지스터가 옮겨져 가는 모습을 보여주는 그림으로서 동작 설명은 다음과 같다. 먼저 데이터를 보내는 CPU가 주변회로 쪽으로 데이터를 전송할 경우에는 universal bridge를 통하여 이루어진다. 따라서 CPU와 bridge간의 데이터 전송은 핸드셰이크 프로토콜 신호를 사용하게 된다. CPU에서 bridge 방향으로 데이터를 전송하려면 먼저 request 신호를 bridge쪽으로 보내 bridge로부터 허가를 받아야 한다. bridge는 request 신호를 받은 후 내부 레지스터가 데이터가 다 찾는지 아닌지 확인을 하. 만약 데이터가 포화상태일 경우 acknowledge 신호는 full 신호를 1로 하여 현재 bridge가 포화상태임을 CPU쪽으로 알리게 된다. 그러면 CPU는 데이터 전송을 hold 하게 된다. 만약 bridge내의 레지스터가 포화가 아닐 경우에는 full 신호를 0으로 하여 CPU쪽으로 주게 된다. 즉 bridge는 데이터 전송 허가를 하게 되며 CPU로부터 데이터를 받을 준비를 하게 된다. 그후 bridge내의 첫 번째 레지스터부터 CPU 클럭에 동기되어 데이터가 저장된다. 이 때 첫번째 레지스터의 write-flag는 1로 세팅되며 새로운 데이터가 들어와 있다는 것을 가리킨다. 그 후 계속하여 CPU 클럭에 동기 되어 두 번째 데이터도 두 번째 레지스터에 write 되며 마찬가지로 write-flag가 세팅된다.

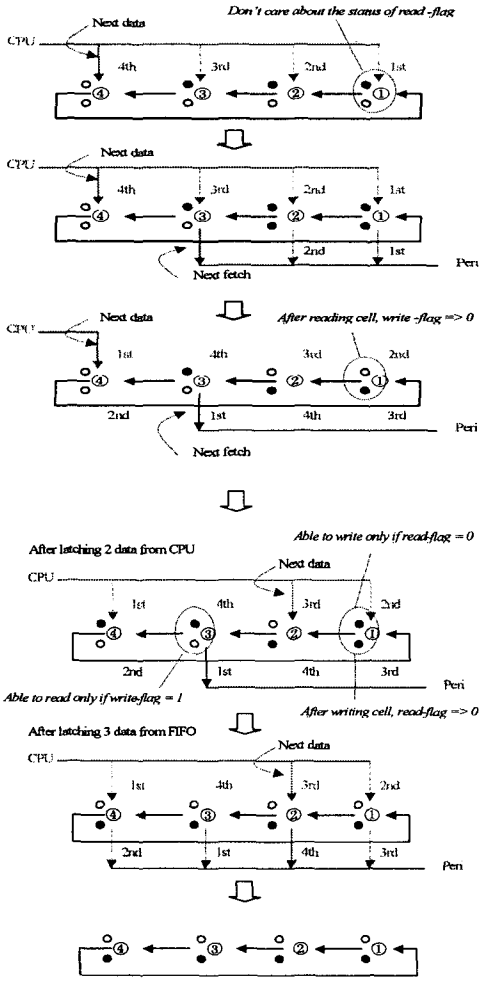

그림 4. universal bridge 내부 동작

Receiver측면에서의 데이터 read에 대해서 살펴보

면 첫번째 write-flag 를 체크하여 데이터의 유무를 파악한 후 주변회로 쪽으로 클럭에 동기시켜 데이터를 옮기게 된다.그 후 첫번째 레지스터의 read-flag는 1로 세팅되어 read가 되었다는 것을 나타내고 write-flag는 0으로 세팅되어 비어있다는 것을 나타낸다. 마찬가지로 두 번째 레지스터에 대해서도 같은 순서로 진행이 된다. universal bridge 구조에서 레지스터에 대한 시뮬레이션 결과는 그림 5와 같다.

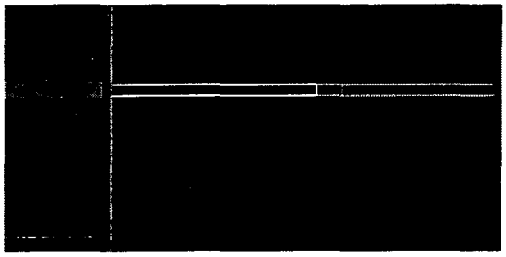

그림 5. Universal bridge 시뮬레이션 결과

III. IP-based 시스템 시뮬레이션

이상과 같이 설계한 Universal bridge회로와 RISC 마이크로 프로세서를 이용하여 그림 6과 같은 간단한 시스템을 설계할 수가 있다. 그리고 Xilinx device 를 사용하여 RISC 마이크로 프로세서를 구현하여 효과적인 시뮬레이션을 할 수 있도록 하였고 PC ISA 버스 인터페이스를 이용하여 그림 7과 같이 시스템을 구축할 수가 있다

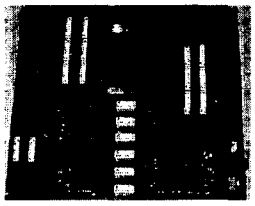

그림 6.시뮬레이션을 위한 테스트 환경 시스템

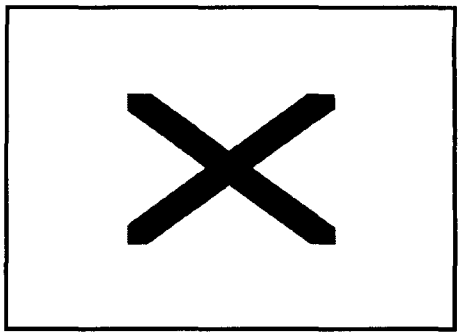

그림 7. PCISA 버스를 이용한 시뮬레이션 시스템

IV. 결론

본 연구에서는 서로 다른 클럭을 가진 블록들간의 데이터 전송을 효과적으로 할 수 있는 universal bridge에 대해 기술했으며 이를 이용하면 클럭 스피드에 구애 받지 않고 저전력으로 데이터를 전송할 수가 있다. 양방향의 universal bridge 회로를 설계하여 FPGA로 구현을 하였고 효과적인 시뮬레이션을 위하여 RISC 마이크로 프로세서를 IP로 사용하여 검증을 하였다.

Acknowledgment

본 연구는 산업자원부 System 2010 프로젝트 의 지원으로 이루어졌음.

References

[1] N.C.Paver. The design and implementation of an asynchronous microprocessor , Ph.D thesis, University of Manchester, 1994.

[2] K.Y. Yun & A.E. Dooply, Pausible Clocking-Based Heterogeneous Systems, IEEE Transactions on VLSI System, Vol.7, No.4, Dec. 1999.

[3] Chapiro Globally-Asynchronous Locally-Synchronous Systems , PhD Thesis, Stanford University, Oct. 1984

[4] Hwi-Sung Jung, Moon-Key Lee, Analysis and Implementation of Interface for Heterogeneous System, Proc. of The Second IEEE AP-ASIC, pp.147-150, Aug. 2000, Cheju, Korea.