AMBA 버스와 IP간의 통신을 위한 인터페이스 자동생성에 관한 연구

준회원 서 형 선*, 이 서 훈**, 황 선 영***

A Study on Automatic Interface Generation for Communication between AMBA Bus and IPs

Hyung-sun Seo* *Associate Member,* Sir-hun Lee**, Sun-young Hwang***

요 약

본 논문은 SoC 설계시 AMBA 버스와 다른 프로토콜을 갖는 IP간의 통신을 위한 인터페이스 설계를 위한 확장 STG 표현을 제안하며, 이를 적용하여 다양한 IP간의 통신을 위한 프로토콜 탐색 알고리즘과 인터페이스를 자동 생성하는 시스템의 구축을 제시한다. 시스템은 동기/비동기 전송타입, 데이터 사이즈 등이 서로 다른 프로토콜을 갖는 IP 간의 데이터 전송이 가능한 인터페이스 모듈을 생성한다. AMBA AHB 버스와 타겟 IP로써 비디오 디코더간의 매뉴얼한 인터페이스 설계와 자동생성된 모듈간의 성능을 비교한 결과 burst 통신의 성능은 거의 차이를 보이지 않았다. Single 통신의 경우 매뉴얼한 설계에 비해 다소 떨어지는 성능을 보여줬으나 전체 IP의 면적을 고려할 때 극히 미미한 면적 증가만을 보였다.

Key Words : AMBA, SoC, Interface

ABSTRACT

This paper describes a study on the automatic generation system of the interface for communication among AMBA bus and IPs with different protocols. Employing an extended STG, the proposed system generates the interface modules required for the communication among IPs with different protocols. For an example system, the interface module for communication between AMBA AHB bus and a video decoder has been generated and verified in its functionality. The area and latency have been compared with the manually designed interface. For burst-mode communication, the generated interface module shows the comparable performance with the manually designed module. For single-mode communication, the generated interface module shows a slightly worse performance than the manually designed module. However, the increased area is negligible considering the size of the IP.

I. 서 론

최근 VLSI 설계 공정 및 제조 기술의 발달로 칩의 집적도는 매년 50% 이상 증가하고 있으나, 설계 생산성은 이에 미치지 못하고 있다[1]. 이에 설계 시간과 비용을 줄이고 생산성을 향상시키기 위해 IP(Intellectual Property) 블록을 사용한 설계의 재사용 기법이 연구 개발되어, 이미 검증되고 설계에 사용된 제품이나 지적 재산권, 특허로 보호 받는 IP 블록의 사용은 계속 증가하고 있다. 특히 설계의 재

*서강대학교 전자공학과 CAD & Computer System 연구실 (hwang@ccs.sogang.ac.kr)
논문번호 : 040054-0203, 접수일자 : 2004년 2월 9일
※본 연구는 대학 IT연구센터 (연세대학교 ITRC) 육성 지원사업 지원으로 수행되었습니다.

사용은 시스템 레벨에서는 이미 보편화 된 것으로 시스템을 구성하는 모든 모듈들을 설계하여 제작하지 않고, 기존에 사용 중인 프로세서나 ASIC, FPGA, 메모리 등의 다양한 모듈 중에서 사용자의 사양에 맞는 제품을 선택하고 이들을 구성한다[2].

기존에 설계된 하드웨어 모듈과 소프트웨어 모듈을 이용하여 새로운 시스템을 디자인 할 경우, 모듈간의 데이터 전달 방식, 프로세서의 동작 주파수, 신호 동기 방식 등이 일반적으로 일치하지 않아 모듈간의 데이터 교환을 위해 프로토콜 변환을 지원하는 인터페이스 블록을 설계해야 한다. 시스템 통합과정에서 모듈간의 인터페이스 설계가 전체 설계 시간 및 비용의 대부분을 차지하므로 이를 상위 수준에서 자동적으로 합성하는 자동화 설계가 요구된다. 인터페이스 자동 설계에 대한 연구는 대부분 인터페이스의 프로토콜을 기술하기 위한 독자적인 기술 방법을 가정하고 있으며, 이는 HDL을 사용한 인터페이스의 특성 기술이 어렵기 때문이다. 하드웨어 모듈들 간의 데이터 전송을 지원하기 위하여 전용 인터페이스가 요구되며, 자동생성을 위해 동기식 인터페이스 방식과 비동기식 인터페이스의 설계 구현이 요구된다[2][3]. 상이한 프로토콜을 사용하는 IP 간의 통신을 위해서는 모듈간에 정보가 전달될 수 있는 물리적인 데이터패스와 입출력포트나 데이터패스를 제어하는 프로토콜이 요구된다. 데이터패스의 상태를 각 모듈에게 전달하기 위한 제어 신호간의 동기화가 필요하다. 인터페이스 회로의 설계를 위하여 글로벌 클럭을 사용한 동기식 방법과 입력 신호와 피드백 신호의 변화에 따라 상태가 변화하는 비동기식 방법이 제안되었다[2][4]. 비동기 회로는 모든 모듈에 글로벌 클럭을 사용하지 않으므로 모듈간 같은 클럭에 의한 동작이 요구되지 않아 클럭 스큐에 의한 문제점을 고려하지 않고 회로를 설계할 수 있다. 또한 가장 느린 속도로 동작하는 모듈을 고려하여 클럭을 조절하지 않고 회로를 설계할 수 있어 평균적으로 빠른 동작 속도를 보이며 적은 전력소모의 설계가 가능하다[5]. 동기식 인터페이스 설계는 플립플롭을 사용하여 시스템의 동작 시간을 클럭에 맞추어서 조절하므로 설계가 간단하고 테스트가 용이하다는 장점이 있다. 반면에 칩의 집적도가 증가하고 다양한 모듈을 사용하여 시스템을 설계할 경우, 모든 모듈에 대하여 동일한 클럭을 사용하면 동기를 맞추기가 어렵고 버스나 콘트롤 신호의 전송 길이가 늘어남에 따라 클럭 스큐 문제가 발생한다는 단점이 있다[6]. 상이한 프로토콜을 사용하는 모듈들 간의 통신을 위한 연구로서 SoC 버스 표준이 제안 되고있다. 본 연구에서는 ARM사에서 제안한 AMBA 버스에 기반을 두고 다양한 프로토콜을 갖는 IP와의 인터페이스를 위한 기법을 제시한다.

그림 1은 전형적인 AMBA 시스템의 구조로서, AMBA는 SoC 설계를 위한 on-chip 통신의 표준안이 되고 있다. AMBA는 AHB(Advanced High-performance Bus), APB(Advanced Peripheral Bus)로 구성되며, 여기에는 ARM과 같은 CPU나 DMA 등이 연결될 수 있으며 peripheral macrocell 통신을 위해서 APB가 연결될 수 있는데 이것은 AHB의 로컬 버스처럼 여겨지며 브리지를 사용하여 연결된다[7].

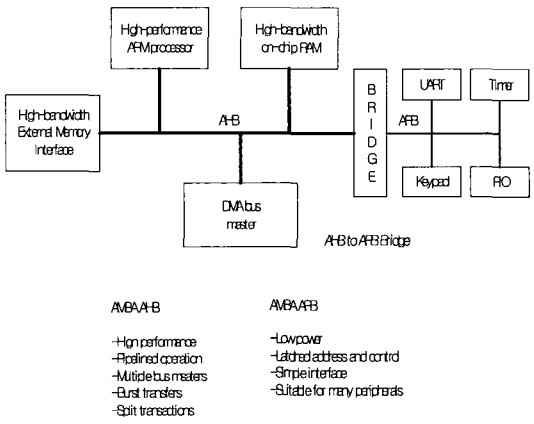


그림 1. 전형적인 AMBA 시스템의 구조

인터페이스 합성을 위해 모듈의 semantics에 따라서 다양한 합성 방식이 제안되었다. 하드웨어 모듈간의 인터페이스와 하드웨어 모듈과 소프트웨어 모듈, 소프트웨어 모듈간의 인터페이스 대한 연구가 보고 되고 있다[4][8][9]. 인터페이스 합성에 기본이 되는 하드웨어 모듈간의 인터페이스 합성에 관한 다양한 기법들로써 동기 회로는 모듈의 수행이 global 클럭과 클럭에 의한 플립 플롭에 의해서 제어되며 동작한다. 동기 회로는 모듈의 동작 속도와 신호의 전달 속도 차이에 따라 발생하는 레이스와 해자드에 대한 고려가 필요치 않아 시스템 설계와 분석이 용이한 반면 클럭 속도의 증가에 따른 스큐 문제와 버스의 길이에 제한을 가지며, 클럭을 구동하기 위하여 더 많은 전력을 사용하고 모듈간 클럭 주파수가 일치하지 않은 경우 별도의 클럭 변환 모듈을 사용해야 시스템에 합성할 수 있어 모듈의 연결이 제한적이다. 동기식 모듈간의 인터페이스 합성

은 모듈간의 인터페이스 입력기술을 위해 합성이 가능한 HDL 이나 로직 다이어그램을 이용한 방법이 이용되고 있으나 인터페이스 합성에 필요한 concurrency, conflicts 등의 semantics를 충분히 반영하지 못하므로 동기화 PN(Petri Net), 동기화 언어, RG(Regular Grammar)등을 이용한 연구가 보고되었다[4][10]. 타이밍 다이어그램은 프로토콜을 웨이브 형태로 표현하며 디바이스나 프로세서의 데이터 북에서 프로토콜 기술에 보편적으로 사용된다. 신호의 변화를 나타내기 쉽고 인터페이스의 동작을 직관적으로 알아볼 수 있으며 동기 회로와 비동기 회로를 모두 나타낼 수 있기 때문에 인터페이스를 나타내는 일반적인 방법으로 폭 넓게 사용되고 있다. Borriello에 의하여 제안된 Suture 알고리즘은 타이밍 다이어그램으로부터 추출된 정보로 이벤트를 구성하여 인터페이스를 합성한다. 이벤트 그래프는 STG의 종류로 사이클이 존재하지 않으며 인과관계와 타이밍 제약 조건 등을 포함한다[4][8][11]. 타이밍 다이어그램으로 인터페이스를 합성할 경우에는 회로를 합성하고 최적화 하는데 필요한 모든 정보를 나타낼 수가 없으므로 결과 회로에 대하여 별도의 최적화 과정을 필요로 한다는 단점이 있다. 인터페이스 합성은 모듈간의 프로토콜이 다른 경우에는 handshake 방식으로 쉽게 동기를 맞출 수 있어 데이터 전송의 신뢰도가 높기 때문에 예전부터 비동기식 설계 기법이 적용되었다. 비동기 인터페이스 합성을 위한 입력은 프로토콜의 동작을 기술하는데 적합한 상위 레벨의 모델인 STG(Signal Transition Graph)가 보편적으로 사용된다[4]. STG는 각 신호의 변화를 하나의 버텍스로 나타내고, 신호변화 사이의 인과 관계를 아크로 연결한 방향성이 있는 그래프로 전체적인 프로토콜의 동작을 기술한다[12]. 본 논문에서는 신호의 concurrency나 conflicts 등의 프로토콜 semantics를 충실히 반영하고 신호들의 인과 관계를 정확히 나타낼 수 있는 STG를 적용하여 인터페이스 기술을 하도록 하였으며 프로토콜의 상위레벨 기술로 프로토콜을 탐색, 변환과정을 거친다. 인터페이스 모듈의 생성을 위해 상위레벨 기술시 지나친 복잡성과 프로토콜 기술의 한계점을 보완하는 확장 STG를 제안하여 적용하였다.

본 논문은 다음과 같은 구성을 갖는다. 2절에서는 인터페이스 자동생성기에 대한 전체적인 개관을 설명하며, 3절에서는 프로토콜 기술에 적합한 STG를 소개하고 프로토콜 탐색을 용이하도록 변형된 확장 STG의 표현과 예제를 다룬다. 그리고 이를 적용한 프로토콜 탐색 알고리즘을 설명한다. 4절에서는 생성된 인터페이스 모듈의 구조와 실험 결과를 보이며, 5절에서 결론을 맺는다.

Ⅱ. 시스템의 개관

상이한 프로토콜을 사용하는 모듈들 간의 통신을 위한 연구로서 본 연구에서는 On-chip 버스의 표준으로 자리잡은 AMBA 버스에 기반을 두고 다양한 프로토콜을 갖는 IP와의 인터페이스를 위한 기법을 제시한다. 제안된 시스템은 서로 다른 프로토콜을 사용한 모듈간 통신용 인터페이스 생성을 위하여 프로토콜을 확장 STG를 이용하여 상위레벨에서 기술한다. 그림 2는 확장 STG로 기술된 프로토콜 정보와 라이브러리에 의해서 최종적으로 인터페이스 모듈이 생성되는 흐름을 보인다. AMBA와 타겟 IP에 대해 기술된 인터페이스 프로토콜 정보는 인터페이스 모듈을 합성하기 위한 입력 기술로서 동기 인터페이스 정보, Datapath size 정보, 전송 타입 정보, 시간 제약 정보를 포함한다. 인터페이스 생성기는 기술된 입력 정보와 탐색된 프로토콜 정보에 의해서 생성된 로직 게이트와 해당 라이브러리를 포함하는 인터페이스 모듈을 생성한다.

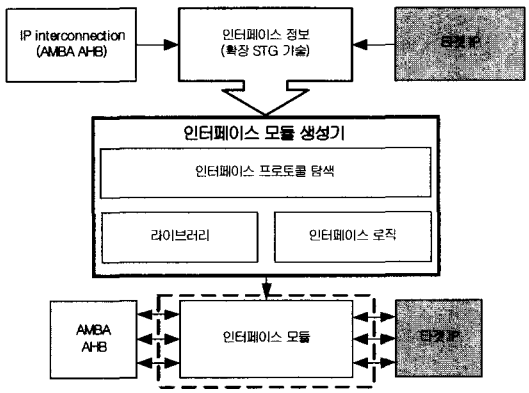


그림 2. 인터페이스 자동생성 흐름도

시스템을 구성하는 각 모듈은 데이터 전송을 위한 고유의 프로토콜을 가진다. 예를 들어, 프로세서는 글로벌 클럭을 이용하여 프로토콜의 제어 신호를 조절하고, 메모리 소자와 기타 다른 모듈들은 제조 회사의 고유의 데이터 교환 방식을 사용한다.

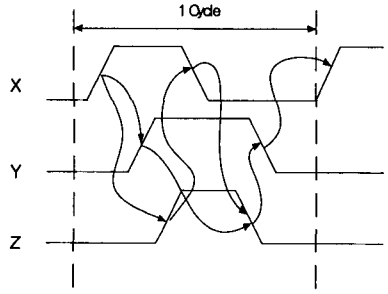
모듈간의 원활한 데이터 교환을 위해서는 인터페이스 로직을 통하여 각 모듈간 데이터 패스를 구성하고 프로토콜을 변환하여 데이터를 전송한다[7].

Ⅲ. 제안된 확장 STG를 이용한 프로토콜 기술

본 절에서는 프로토콜 기술에 적합한 상위레벨 모델로써 사용되는 STG에 대해서 소개하며, 비동기 인터페이스 프로토콜 기술에 적합한 기존의 STG가 가지는 기술의 한계점을 보완하여 동기식 및 시간 제약 조건에 대한 기술이 가능한 확장된 형태의 STG 기술을 제시한다. 확장 STG를 적용한 서로 다른 프로토콜을 사용하는 모듈간의 통신을 위한 모델링 기법을 제시한다.

1. STG(Signal Transition Graph)

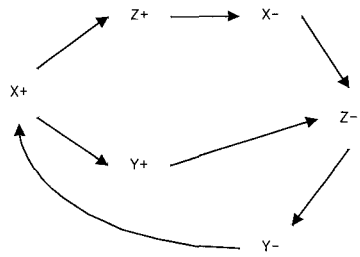
비동기 회로의 합성은 STG 레벨에서 직접 수행되지 않고, 회로를 구성하는 논리식의 유도가 가능한 하위 레벨의 표현 방식인 SG(State Graph)로 변환한 후 수행된다. 그림 3에 비동기 회로의 동작을 나타내는 간단한 타이밍 다이어그램과 이를 표현하는 STG, SG를 보인다. 그림 3 (a)에서 X, Y, Z 세 신호로 이루어진 프로토콜을 기술하고, 그림 3 (b)는 이로부터 구한 STG의 텍스트 형태의 기술, 그림 3 (c)는 STG에 버텍스로 나타내지며, 신호의 변화에 따른 인과 관계는 아크로 표현된다. 그림 3 (c)의 Y-에서 X+로 향하는 아크의 검은 원은 초기값을 나타내어 신호의 변화가 X+에서 시작됨을 나타낸다. X+는 신호 X의 값이 0에서 1로 변화함을 나타내고 X-는 X의 값이 1에서 0으로 변화함을 의미한다. 그림 3 (d)의 버텍스는 회로의 상태를 바이너리 코드로 표시하고, 아크는 신호의 변화를 나타낸다. 초기 상태의 '000'은 신호 X,Y,Z가 모두 low 상태에 있음을 나타내고, 신호 X가 low에서 high로 상태가 변화하여 회로의 상태는 '100'으로 바뀐다.



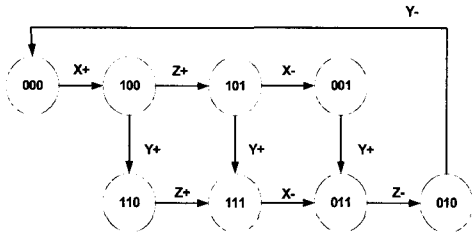
(a)

```
.name sender
.input X
.output Y Z
.graph
X+ Y+ Z+
Z+ X-
Y+ Z-
Z- Y-
Y- X+
.end
```

(b)



(c)



(d)

그림 3. 비동기 회로에 대한 STG 기술 예. (a) 타이밍 다이어그램, (b) STG의 텍스트 표현, (c) STG, (d) SG.

2. 제안된 확장 STG 표현과 예제

프로토콜의 그래프 기술은 상위 레벨에서의 프로토콜 탐색을 용이하게 할 수 있으나 복잡한 통신 구조를 갖는 IP의 경우에 그래프를 이용한 기술시 지나치게 복잡해지는 단점을 갖는다. 제안된 인터페이스 생성 알고리즘에서는 인터페이스 프로토콜로서의 많은 기능들 즉 데이터 사이즈, 전송 타입, 시간 제약정보 등은 라이브러리 화하여 그래프 기술에 대한 복잡성을 줄이도록 하였다.

비동기식 모듈의 상이한 프로토콜을 갖는 모듈간

의 프로토콜 변환은 송수신 모듈의 데이터 패스를 제어하는 신호들을 분석하여 동일한 타입의 신호 사이에 방향성을 고려한 아크를 연결함으로써 양쪽 모듈이 handshake 방식으로 신뢰성 있게 데이터를 교환할 수 있도록 한다. 이러한 프로토콜 변환 알고리즘을 사용하여 상이한 프로토콜을 사용하는 모듈간의 인터페이스를 상위수준에서 합성하는 것이 제안되었다[2]. 동기적인 방식으로 데이터를 전송하는 모듈의 기술을 위하여 IMEC Laboratory에서는 Generalized-STG 모델을 제안하였다[12]. Generalized-STG는 시스템 레벨에서의 복잡한 동기/비동기 인터페이스의 추상적인 동작 모델의 기술을 가능하게 한다. 주요한 추가된 기술에는 signal level 뿐만 아니라 signal transition에 있어서의 기술이 가능한 level semantics와 don't care, undefined behavior, boolean guards에 관한 정보를 포함한다. 또한 이 기술방식은 비동기 모델을 사용한 동기 FSM 모델링이 가능하여 동기/비동기 혼합 모델에 대한 기술이 가능하다. 또한 SG(State Graph) 레벨로도 정의될 수 있다.

본 논문에서는 각 클럭 rising edge나 falling edge에서의 트랜지션에 따른 신호 변환만을 기술한다. 이렇게 기술된 그래프에 의해서 신호변화와 데이터 전송에 대한 아크를 탐색하도록 했다. 본 논문에서는 이것을 확장 STG라 부른다. 그림 4는 AMBA AHB 버스의 기본 전송 타이밍도를 보인다. 마스터는 항상 HCLK이 rising edge에서 어드레스와 제어 신호를 전송하게 되고, 그 다음의 HCLK rising edge에서 슬레이브는 어드레스와 제어 신호를 샘플링하게 된다. 그 다음의 rising edge에서 해당 데이터를 받는다. 여기서 쓰기 전송일 때 데이터 phase의 전체 사이클에 걸쳐서 HWDATA에 데이터를 실으며 읽기 모드일 때에는 HREADY가 HIGH가 되는 부분에서만 HRDATA가 유효하다[7].

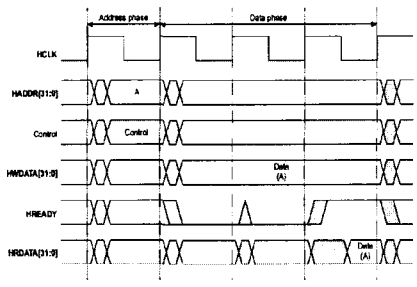


그림 4. AMBA AHB 버스의 기본 전송.

그림 5은 그림 4의 AMBA AHB 버스의 기본 전송에 대한 확장 STG를 적용한 그래프를 보인다. 각 신호는 클럭의 rising edge에 동기 되어 전송된다고 가정하며, HREADY가 low일때는 이것이 high가 될 때까지 대기한다.

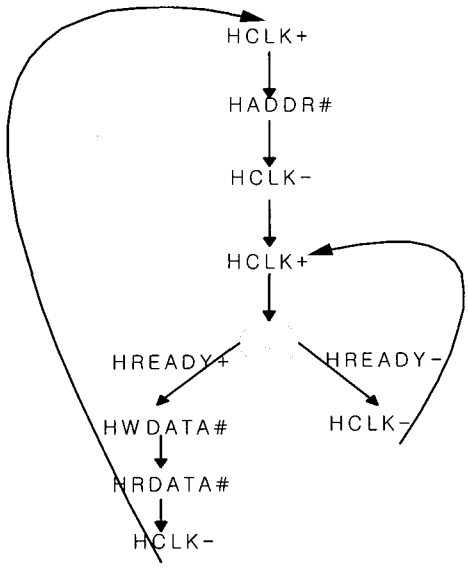


그림 5. 그림 4에 대한 확장 STG 기술.

그림 6은 MPEG-2 비디오 디코더의 configuration 레지스터의 호스트 쓰기와 읽기를 위한 타이밍도를 보인다. 타겟 IP로서 AMPHION사에서 제공하는 MPEG-2 비디오 디코더는 호스트 CPU와 함께 동작할 때에 호스트 비디오 디코더간의 데이터 송수신은 비디오 디코더의 호스트 인터페이스 유닛을 통해 이루어진다. 이 호스트 인터페이스는 configuration과 상태 레지스터의 읽고 쓰기와 호스트에서 메모리의 읽고 쓰기를 지원한다. 호스트 쓰기시에는 그림 7의 A 라인부분에서 보이는바와 같이 클럭의 rising edge에서 발생한다. H_notRegCS가 low가 되고 H_notWrite 신호가 high를 유지할 때, H_DataOut이 유효하다면 다음 사이클에서 H_notDatDrv는 low가 되면서 데이터가 전송된다[13].

그림 7은 그림 6에 대한 확장 STG를 적용한 그래프를 보인다. 프로토콜 탐색을 용이하게 하기 위해서 읽기와 쓰기에 대해서 따로 그래프를 표현하였다.

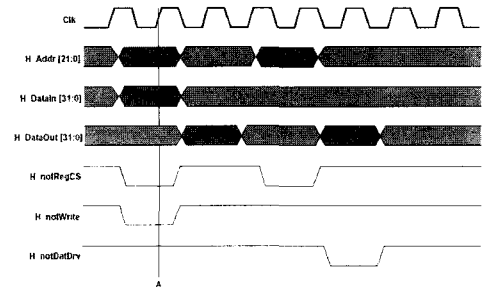
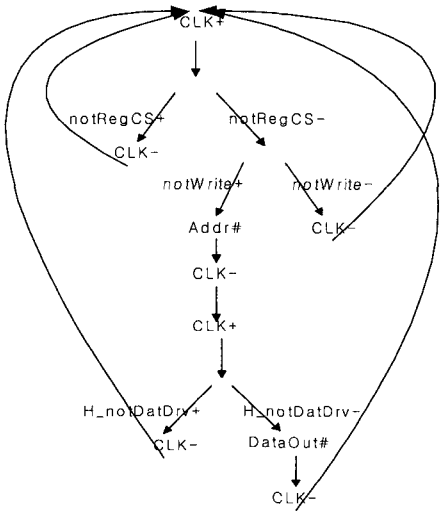
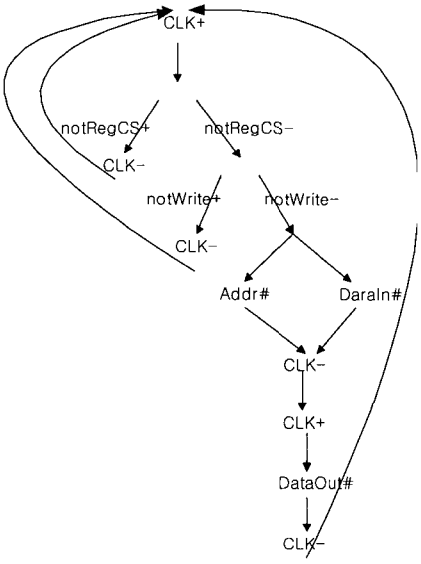


그림 6. Configuration 레지스터 호스트 쓰기와 호스트 읽기 타이밍도



(a)



(b)

그림 7. 그림 7에 대한 확장 STG 기술. (a) 호스트 읽기, (b) 호스트 쓰기.

3. 제안된 확장 STG을 적용한 통신모델

그림 8과 그림 9는 지금까지 제시한 확장 STG 표현을 적용한 AMBA AHB 버스와 타겟 IP에 대한 읽기, 쓰기시의 통신모델을 보인다. 그림 8은 AHB와 비디오 디코더간의 호스트 쓰기시의 확장 STG를 이용한 통신 모델로써 AHB와 비디오 디코더간의 굵은 선은 서로간의 통신을 위하여 주어야 할 이벤트 시점을 나타낸다. AHB의 클럭 신호에 동기되어 어드레스 정보는 우선 미리 생성된 버퍼에 저장되며, HREADY+에 의해서 해당 데이터 신호도 미리 버퍼에 저장하게 된다. 이렇게 먼저 버퍼에 저장된 어드레스와 데이터 정보는 비디오 디코더의 notRegCS와 notWrite 포트에 신호를 보내어 데이터의 송신이 이루어진다.

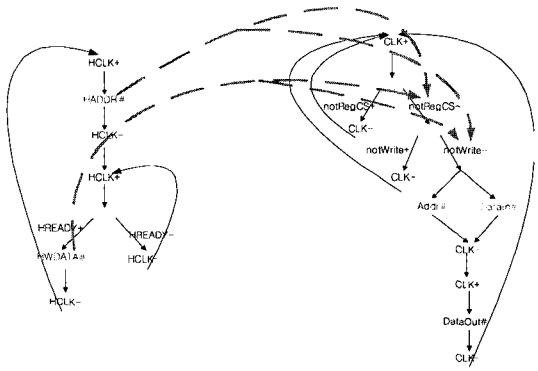


그림 8. AHB와 디비오 디코더간의 호스트 쓰기시의 확장 STG를 이용한 통신 모델.

그림 9는 AHB와 비디오 디코더간의 호스트 읽기시의 확장 STG를 이용한 통신 모델로써 AHB와 비디오 디코더간의 굵은 선으로 연결된 선은 서로간의 통신을 위한 이벤트를 주어야할 시점을 나타낸다. AHB의 첫 번째 클럭 사이클 HCLK+에 동기되어 어드레스정보는 우선 미리 생성된 버퍼에 저장되면서 해당 어드레스에 대한 데이터를 받기위해 비디오 디코더의 notRegCS와 notWrite를 활성화시키는 신호로써 사용된다. 다음 클럭 사이클 HCLK+에서 HREADY-에 의해서 데이터 전송을 받기위한 대기 상태가 유지된다. 비디오 디코더의 데이터 포트 즉 DataOut신호가 유효해 지면 데이터는 버퍼에 저장되면서 AHB의 HREADY신호를 high로 만들며 데이터가 전송된다.

기타 다른 IP의 경우에도 읽기와 쓰기에 대한 제

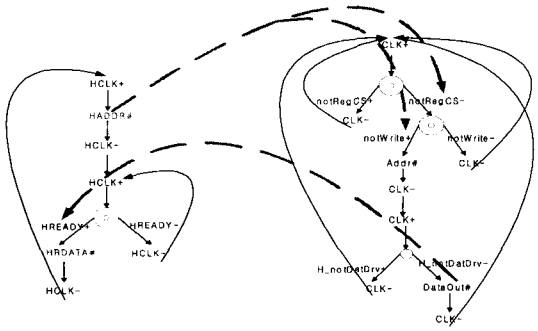


그림 9. AHB와 비디오 디코더간의 호스트 읽기시의 확장 STG를 이용한 통신 모델.

어 신호를 탐색과 미리 생성된 버퍼를 이용하여 모듈들 간의 통신이 가능하며, 각 IP의 특성에 따르는 카테고라이즈된 라이브러리의 구축이 가능하다.

Ⅳ. 인터페이스 모듈의 구조와 실험결과

지금까지 AMBA AHB와 비디오 디코더간의 통신을 위한 그래프를 이용한 통신모델 알고리즘을 제시하였다. 제안된 알고리즘은 읽기/쓰기 전송시 타겟 모듈에 대한 파라미터 정보 즉 전송타입이나 데이터 어드레스 사이즈 정보에 의해 미리 생성된 버퍼와 그래프 모델로 기술된 프로토콜을 탐색하여 생성된 게이트를 포함하는 인터페이스 모듈을 생성한다.

그림 10은 위에서 예시한 AMBA AHB와 비디오 디코더의 호스트 인터페이스와의 통신을 위한 인터페이스 모듈을 블록 다이어그램으로 나타낸 것이다. 타겟 IP의 전송타입과 데이터 어드레스 크기 정보

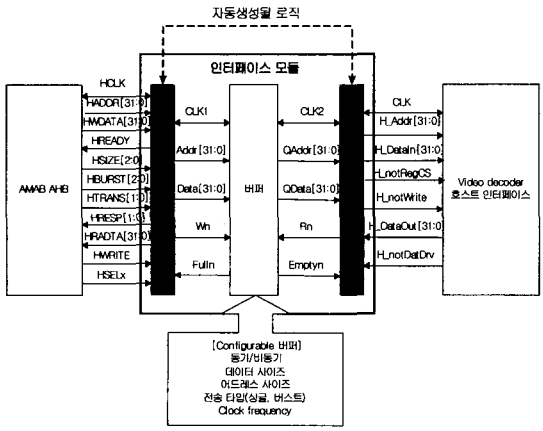


그림 10. 생성된 인터페이스 모듈 구조.

에 의해 결정된 버퍼와 서로간의 통신을 위한 프로토콜 탐색에 의해 생성된 로직 유닛에 의해서 서로간의 통신이 가능하다. 텍스트 형태로 기술된 프로토콜 정보는 인터페이스 자동생성기의 입력 정보로 활용되어 VHDL형태의 코드를 생성한다. 생성된 코드는 Design Analyser를 이용하여 합성하였고, 수행 면적과 파워, 인터페이스 모듈에서의 데이터 지연시간을 측정하였으며 Synopsys의 Vhdl Simulator를 이용하여 결과를 검증하였다. 하이닉스의 0.35 ㎛ 공정 라이브러리를 사용하였으며 면적 측정단위는 two input nand 크기를 1로 하는 게이트의 수를 뜻한다.

AMBA AHB 버스와 타겟 IP로써 사용된 비디오 디코더의 프로토콜 정보로부터의 매뉴얼한 설계와 자동 생성된 인터페이스 모듈을 비교 하였다. 매뉴얼한 설계시 설계자에 따라서 다른 성능을 보일 수 있으므로 인터페이스 자동생성기의 버퍼를 사용한 설계를 하였다. 표 1은 자동으로 생성된 모듈과 매뉴얼로 설계된 모듈과의 성능 비교를 보인다. 첫 번째 열은 전송모드에 따라서 single, burst 전송 모두를 지원하는 모듈과 single 전송만 지원하는 모듈로 구분하였고, 두 번째 열은 각 모듈에 대한 면적정보, 세 번째 열은 파워, 네 번째 열은 latency에 대한 정보를 보인다. Single, burst 전송 모두 지원의 경우 같은 버퍼의 사용으로 자동생성된 모듈과 매뉴얼한 설계에 의한 모듈간의 면적, 파워 및 latency에 대한 성능은 비슷한 차이를 보였으나 latency에 있어서는 30%의 성능 향상을 보인다. Single 전송만 지원하는 경우 면적과 파워, latency에 있어서 매뉴얼한 설계에 비하여 면적과 파워 소모가 각각 31.2%, 51.4% 증가 하였으나 실험에서 사용된 IP가 약 8만 게이트로 구성되어, IP의 면적을 고려할 경우 인터페이스에 의한 면적 증가율은 약 0.04%에 불과하다.

	single, burst 전송 모두 지원			single 전송만 지원		
	매뉴얼 설계	자동 생성	비교	매뉴얼 설계	자동 생성	비교
면적*	1046.5	1114.1	+6.5%	173.0	227.0	+31.2%
파워	19.8 mW	20.5 mW	+3.5%	3.5 mW	5.3 mW	+51.4%
Latency	3.0 ns	2.1 ns	-30.0%	0.47 ns	0.49 ns	+4.3%

표 1. 인터페이스 모듈 성능 비교 표

*Unit gate : Two-input NAND (size : 1)

V. 결론 및 추후과제

기존의 STG가 가지는 문제점과 이를 해결하기 위한 방안으로 확장 STG를 제안하여 제시하였으며, 제안된 모델링 기법을 이용하여 프로토콜을 기술하고 서로 다른 프로토콜간의 통신이 가능한 인터페이스 모듈의 생성 알고리즘을 제시하였다. 확장 STG의 기술은 시간 제약조건과 동기식 모델에 대한 기술 및 Arbiter Behavior에 관한 기술을 정의하며, 전송방식과 데이터 사이즈 등에 관한 정보는 라이브러리화하여 인터페이스 프로토콜 기술을 단순화하였다. 구현된 라이브러리와 확장 STG를 이용하여 기술된 프로토콜 정보로부터 추출된 게이트 수준의 로직을 생성하여 서로 다른 프로토콜을 사용하는 모듈간의 통신이 가능하도록 하였다. 동작을 검증하기 위하여 on-chip 버스의 표준으로 자리잡은 AMBA AHB 버스를 이용하였으며, AMPHION의 MPEG-2 비디오 디코더와의 인터페이스 회로를 합성하여 데이터 전송이 가능함을 확인하였다. 제안된 시스템 상에서의 인터페이스 구현을 위한 모듈간의 인터페이스 설계에 많은 시간을 줄일 수 있다.

그러나, 다양한 전송 타입에 대해 보다 효율적인 전송 모델 기법 보완과 하드웨어 모듈뿐만 아니라 소프트웨어 모듈들과의 통신을 위한 인터페이스 자동생성에 관한 연구가 필요하다.

감사의 글

본 연구는 대학 IT 연구센터 (연세대학교 ITRC) 육성 지원 사업의 연구결과로 수행되었습니다.

참 고 문 헌

[1] Semiconductor Industry Association, "National Technology Roadmap for Semiconductor", 2002.

[2] B. Lin and S. Vercautern, "Synthesis of Concurrent System Interface Modules with Automatic Protocol Conversion Generation", in Proc. IEEE International Conference on Computer-Aided Design, Nov. 1994, pp. 101~108.

[3] R. Passersome, J. Rowson, and A. Sangiovanni-Vincentelli, "Automatic Synthesis of Interface between Incompatible Protocols", in Proc. Design Automation Conference, June 1998, pp. 8~13.

[4] R. Ortega, L. Lavagno, and G. Borriello, "Models and Methods for HW/SW Intellectual Property Interfacing", in Proc. System Synthesis, 1999, pp. 397~432.

[5] J. Cortadella, M. Kishinevsky, L. Lavagno, and A. Yakolev, "Automatic Synthesis and Optimization of Partially Specified Asynchronous System", in Proc. Design Automation Conference, June 1999, pp. 110~115.

[6] C. Ykman-Couvreur, B. Lin, and H. De. Man, "ASSASSIN : A Synthesis System for Asynchronous Control Circuits", IMEC Release Documentation, 1995.

[7] AMBATM Specification(AHB) (Rev 2.0), ARM Ltd, May 1999.

[8] G. De Jong and B. Lin, "A Communication Petri Net Model for the Design of Concurrent Asynchronous Modules", in Proc. Design Automation Conference, June 1994, pp. 49~55.

[9] K. Bilinski and E. Dagless, "High Level Synthesis of Synchronous Parallel Controllers", in Proc. Application and Theory Petri Net, June 1996, pp. 93~112.

[10] A. Baganne, J. Philippe, and E. Martin, "A Formal Technique for Hardware Interface Design", in Proc. IEEE International Symposium on Circuits and Systems, June 1997, pp. 1592~1595.

[11] P. Knudsen and J. Madsen, "Graph Based Communication Analysis for Hardware /Software Codesign", in Proc. Workshop on Hardware/Software Codesign, May 1999, pp. 131~135.

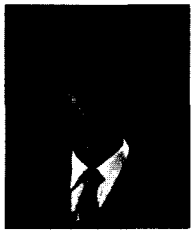
[12] P. Vanbekbergen, Ch. Ykman-Couvreur, B Lin, and H. De Man, "A Generalized Signal Transition Graph Model for Specification", in Proc. European Design and Test Conference, Feb 1994, pp. 378~384.

[13] CS6651 MPEG-2 Video Decoder for FPGA Datasheet, AMPHION Ltd, Aug 2002.

황 선 영 (Sun-Young Hwang)

1976년 2월 : 서울 대학교 전자공학과 졸업
1978년 2월 : 한국 과학원 전기 및 전자공학과 공학석사 취득
1986년 10월 : 미국 Stanford 대학 전자공학 박사학위 취득
1976~1981 : 삼성 반도체 주식회사 연구원, 팀장
1986~1989 : Stanford 대학 Center for Integrated System 연구소 책임 연구원
Fairchild Semiconductor Palo Alto Reaserch Center 기술 자문
1989~1992 : 삼성전자(주) 반도체 기술 자문
1989년 3월~현재 : 서강대학교 전자공학과 교수
2002년 4월~현재 : 서강대학교 정보통신대학원장
<관심분야> SoC 설계 및 framework 구성, CAD시스템, Computer Architecture 및 DSP System Design 등

서 형 선 (Hyung-sun Seo) 준회원



2001년 2월 : 서강대학교 전자공학과 졸업
2004년 2월 : 서강대학교 전자공학과 석사학위 취득
2004년 4월~현재 : 엠택비젼(주) SoC 연구소 주임 연구원

<관심분야> SoC 설계, System verification, RTOS

이 서 훈 (Ser-Hun Lee)

2003년 2월 : 서강대학교 전자공학과 졸업
2003년 4월~현재 : 서강대학교 전자공학과 대학원 CAD & Computer Systems 연구실 석사과정

<관심분야> SoC 설계, IP Interface 자동설계기법