논문 2006-43SD-10-5

IP 범주화와 특성 대응을 통한 인터페이스 회로 자동 합성

(Automatic Interface Synthesis based on IP Categorization and Characteristics Matching)

윤 창 열*, 장 경 선**

(Chang-Ryul Yun and Kyoung-Son Jhang)

요 약

시스템 온 칩(SoC) 설계에서는 설계 시간을 줄이기 위해서 미리 검증된 IP(Intellectual Property) 하드웨어 블록을 사용한다. 서로 다른 통신 프로토콜을 사용하는 IP간의 연결을 위해서는 인터페이스 회로가 필요하다. SoC 설계에는 인터페이스 회로 설계가 빈번하게 발생하며, 오류발생 가능성이 높다. 따라서 인터페이스 회로의 자동생성이 필요하다. IP의 통신 프로토콜을 입력으로 인터페이스 회로를 생성하는 여러 연구가 진행되어 왔다. 그러나 IP의 통신 프로토콜만으로는 인터페이스 회로를 생성하기 어려운 경우가 있다. 주소를 사용하는 IP와 주소를 사용하지 않는 IP간 연결, 주소/데이터가 여러 포트로 전송되는 IP와 하나의 포트로 전송되는 IP간 연결, 매 전송마다 주소와 데이터가 함께 전송되는 IP와 하나의 시작주소에 여러 데이터가 전송되는 IP간의 연결 등에서는 기존 방법에 의한 인터페이스 회로의 자동 생성이 어렵다. 그것은 기존의 방법들이 IP 특성에 따른 범주 구분과, 특성 간의 대응에 따라 합성 알고리즘도 변해야 함을 고려하지 않았기 때문이다. 본 논문에서는 IP의 통신 프로토콜 특성에 따라 범주를 나누고, 어떤 특성을 갖는 IP의 연결인 지에 따라 인터페이스 합성 알고리즘이 이를 고려할 수 있도록 하였다. 실험에서는 다양한 특성을 갖는 IP 간 연결을 위한 인터페이스 회로를 생성하고 검증했음을 보인다.

Abstract

A system-on-a-chip (SoC) design uses pre-verified IP hardware blocks in order to reduce design time. We need interface circuits to connect IPs with different protocols. In SoC design we should design interface circuits frequently and these tasks are somewhat time-consuming and error-prone. So it is necessary to generate the interface circuits automatically. Several studies have been made on generating interface circuits only from the communication protocols of IPs. With existing approaches, it is not easy to generate interface circuits connecting two IPs only from communication protocols: connection between IP with address and IP without address, connection between IP with only one port to transfer address/data and IP with different ports for address and data, connection between IP that transfer address and data together and IP that transfer only one address with a number of data in a burst. No consideration of various characteristics of IPs and no changed algorithm are responsible for it. In order to solve this problem, the proposed approach categorizes communication protocols of IPs, and takes characteristics matching of IPs into account during the interface synthesis. In experiments, we show that we could correctly generate and verify interface circuits for IPs with different characteristics.

Keywords: Interface Circuits, Interface Synthesis, IP Categorization, Characteristics Matching

I. 서 론

SoC의 설계에서 다른 프로토콜을 사용하는 IP를 시스템 버스 또는 IP와 연결하여 올바르게 데이터를 주고받기 위해서는 인터페이스 회로가 필요하다. 인터페이스 회로는 올바른 데이터의 전달을 위해 프로토콜의 변환의 역할을 담당한다. 인터페이스 회로 설계는 빈번하

* 정회원, 충남대학교 BK21 차세대정보기술SW인력양성사업단
(Human Resource Development Consortium for Next Generation Software in Information Technology, Chungnam University)
** 종신회원, 충남대학교 전기정보통신공학부 컴퓨터
(Dept. of Computer Engineering Chungnam National University)
접수일자: 2006년4월12일, 수정완료일: 2006년9월15일

게 발생하며, 매우 번거롭고 오류 발생 가능성이 높기 때문에, 인터페이스 회로 생성의 자동화가 꼭 필요하다. 최근에 인터페이스 회로의 자동합성 분야의 연구에서 IP의 통신 프로토콜을 입력으로 하여 인터페이스 회로를 자동 합성하는 연구들이 많이 진행되고 있다[1, 2, 3]. 이들 연구에서는 IP의 통신 프로토콜을 추상화(FSM 형태)하고, 이에 근거하여 입력 FSM 들의 CPM(Cross Product Machine)을 생성하고, CPM에 기반 하여 인터페이스 회로를 자동생성 하는 방법이 인터페이스 자동합성에 많이 사용되고 있다.

그러나 기존의 방법으로 인터페이스 회로를 생성하기 어려운 경우도 있다. 주소를 사용하는 IP와 주소를 사용하지 않는 IP간 연결, 데이터와 주소가 여러 포트로 전송되는 IP(AHB[4], OCN[5] 등)와 하나의 포트로 전송되는 IP(PCI[6] 등)간 연결, 매 전송마다 주소와 데이터가 함께 전송되는 IP(AHB)와 하나의 시작 주소와 여러 데이터가 전송되는 IP(OCN, PCI)와 연결을 위한 인터페이스 회로의 생성에는 IP의 통신 프로토콜에 대한 정보만으로 인터페이스 회로를 자동생성하기 어렵다. 왜냐하면 기존 방법은 데이터 간 연결(matching)을 통해 CPM을 생성하는 과정이 필요하나, 앞선 설명한 예에서는 상대 IP가 해당 데이터를 구동하지 않는 경우도 있기 때문에 CPM을 생성하기 어렵다. 이런 문제를 해결하기 위해서 IP가 갖는 다양한 특성 정보와 특성들 간 대응을 위한 정보가 인터페이스 합성에 고려되어야 한다. 또한 상대 IP에서 제공되지 않는 동작을 인터페이스 회로에서 어떻게 처리해야 하는지도 결정되어야 한다. 이러한 정보는 IP간 연결을 위한 정보이므로, IP의 통신 프로토콜에는 나타나 있지 않다. 본 논문에서는 이 정보를 특성 간 대응 정보(CM-Characteristics Matching)라 부르고, 인터페이스 합성에서 연결 정보를 IP의 통신 프로토콜과 함께 고려한다.

이를 위해 본 논문에서는 IP의 통신 프로토콜의 동작 특성을 3종류로 구분하고, 어떤 동작 특성을 갖는 IP들의 연결인지에 따라 다른 형태의 인터페이스 회로가 생성될 수 있도록 하였다. 그림 1은 본 연구에서 제안된 인터페이스 회로 생성 흐름이다. 인터페이스 자동합성의 입력은 연결하려는 IP의 통신 프로토콜과, IP의 특성 간 대응 정보(CM)이다. 각 통신 프로토콜의 기술로부터 FSM이 자동 추출되고, 이 FSM에 기반 하여 인터페이스 회로 생성을 위한 FSM이 생성된다. 이때 특성 간 대응 정보에서 제공하는 정보에 따라 FSM을 만드는 알고리즘에 영향을 준다. 생성되는 인터페이스

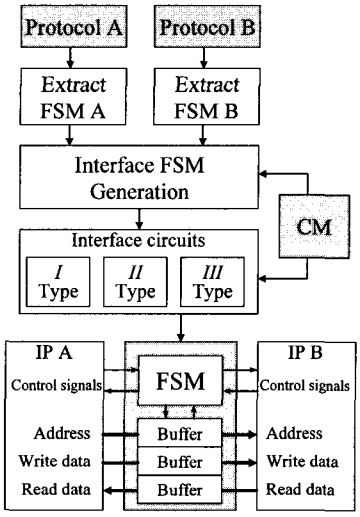


그림 1. 제안된 인터페이스 회로 생성 흐름
Fig. 1. Design flow of proposed interface synthesis.

회로(합성 가능한 RTL VHDL 코드)는 연결 IP의 특성에 따라 3가지 형태 중 한 가지의 특성을 갖게 된다.

IP 통신 프로토콜 기술을 위해 프로토콜 기술 언어를 정의하였고, 기술 언어로부터 인터페이스 FSM을 자동으로 추출하였다. 인터페이스 회로의 합성 알고리즘은 논문 [1]에서 제시된 알고리즘을 응용하였고, 합성 과정에서 IP의 특성 간 대응 정보를 고려할 수 있도록 합성 알고리즘을 변경하였다. 실험에서는 다양한 특성을 갖는 IP 간 통신을 위한 인터페이스 회로를 자동생성 하였고, 올바르게 동작함을 검증하였다.

Ⅱ장에서는 인터페이스 자동합성 관련 연구에 대해 소개 하고, Ⅲ장에서는 CM에 포함되는 내용과 인터페이스 회로의 구조에 대해 살펴본다. Ⅳ장에서는 인터페이스 자동합성을 위한 기술언어, 자동합성 알고리즘 등에 대해 소개한다. Ⅴ장에서는 다양한 프로토콜의 기술과 인터페이스 회로의 합성 결과를 보인다. Ⅵ장은 요약 및 향후 과제에 대해 설명한다.

Ⅱ. 관련 연구

인터페이스 자동 합성을 위해 IP의 통신 프로토콜을 추상화 하여 표현하는 연구가 있었다. 논문 [3]은 IP의 인터페이스 프로토콜을 정규문법으로 표현하였고, 이를 인터페이스 합성에 이용한 최초의 시도였다. PIG는 IP 간 전송되는 데이터(token)를 규정하고, IP의 입출력 포트의 동작 집합(term)을 정의한 후, 이들 간 순서로 인터페이스 프로토콜을 기술하였다. 사이클의 순서에 따라 각 신호의 이벤트 집합의 순서(Protocol Flow

Graph)를 표현하여 인터페이스 프로토콜을 기술하는 시도[7]도 있다. 이들 표현 방법은 분기가 많은 다양한 동작을 기술하기 어려웠다. 논문 [8]에서는 통신 프로토콜을 타이밍 다이어그램으로 표현하고, 각 IP의 신호들의 특성을 분류한 후, 같은 특성을 갖는 신호끼리 연결한 후, 정해진 구조(Template)를 갖는 인터페이스 회로를 생성하였다. 이런 방법은 비슷한 통신 프로토콜을 사용하는 IP 간 연결에는 효과적이나, 해당 특성을 갖는 신호가 없거나, 데이터 전송 특성이 많이 다른 IP 들 간의 연결을 처리하기 어렵다.

또한 인터페이스 프로토콜을 FSM 형태로 기술하고, 입력 FSM에 근거하여 CPM(Cross-Product-Machine)을 생성하고, 이에 기반 하여 인터페이스 회로를 자동 생성 하는 방법이 많이 연구되어 왔다. 논문 [9]은 생성된 CPM에서 불필요한 전이를 제거하는 알고리즘을 제안하였으나, 최종 인터페이스 회로의 생성에 사용자의 결정이 필요 하였다. 논문 [10]는 인터페이스 FSM으로부터 CPM을 생성하고, 전송 대역폭을 고려하기 위해, 데이터 전송에 FIFO를 사용하였으나, 인터페이스 합성에 많은 시간이 필요하다는 단점이 있다. 논문 [1]은 중재기와 브리지 등을 포함하여 모든 버스 컴포넌트들을 오토마타-기반으로 모델링 할 수 있는 프레임웍을 제시하였고, 다중 클록에서 동작하는 IP간의 인터페이스 합성 방법을 제시하였다. 그러나 데이터 전송의 특징이 많이 다른 IP 간 연결을 위한 인터페이스 생성이 어렵다. 논문 [2]은 해당 IP의 인터페이스 FSM을 이용하여 연결하려는 두 IP간 호환성을 확인하여, 부가적인 인터페이스 회로 없이 통신이 가능한지를 파악하는 방법을 제시하였다. 연구 [11]은 IP의 인터페이스 프로토콜을 상위 수준의 언어로 기술하고, 전송 데이터는 FIFO에 저장됨을 가정하였다. CPM을 생성하지 않고, FIFO를 통해서 데이터를 전달한다. IP와의 통신은 사용자의 인터페이스 기술로부터 생성된 'IPC'라는 모듈을 통해서 제어된다.

III. IP의 특성 대응 정보(CM)

다양한 특성을 갖는 IP들의 연결을 위한 인터페이스 회로를 생성하기 위해서는 IP의 동작 특성을 파악하여야 하고, 이들 특성들 간 대응 정보가 필요하다. 본 논문에서는 이를 IP 특성 대응 정보(CM)라 부른다. CM에는 IP간 연결 특성, 포트 연결, 제어 신호 정보 등이 포함될 수 있다.

표 1. 주소/데이터 전송 측면에서 IP 통신 프로토콜 구분

Table 1. The IP categorization of communication protocols on address and data transfer.

종류	특징	예
주소 분리형 (A)	주소, 쓰기, 읽기 데이터가 각각 다른 포트를 통해 전송됨	AHB, OCN, BVCI[12] 등
주소 일체형 (B)	주소, 쓰기, 읽기 데이터가 하나의 포트를 통해 전송됨	PCI 등
비주소형 (C)	주소를 사용하지 않고, 쓰기 읽기 데이터만 전송함	DES 등

본 논문에서는 IP의 동작 특성을 주소와 데이터의 전송의 측면에서, 3종류로 구분하였다. IP의 통신 프로토콜을 표 1과 같이 3종류로 분류 하였다.

서로 다른 형의 프로토콜을 사용하는 IP간 인터페이스 회로의 생성을 위해서는 기존의 방법으로는 인터페이스를 생성하기 어렵다. 주소분리형과 주소일체형 간의 연결에는 주소/데이터를 적절히 분배해야 하고, 주소를 사용하는 IP와 비주소형 IP의 연결에는 주소가 제어신호의 역할을 할 수 있기 때문에 각 IP의 주소와 데이터 간 연결뿐만 아니라 제어신호와의 연결 문제도 해결해야 한다. 이들 문제는 결국 각 IP의 어떤 포트끼리 연결할 것인지(Ports Paring)에 대한 문제, 포트를 통해 전송되는 주소/데이터를 어떻게 연결할 것인지에 대한 문제(Contents Mapping), 어떻게 서로 다른 동작 특성들을 연결할 수 있는가라는 문제(Characteristics Matching)로 나누어 생각할 수 있다. 따라서 CM에 다음과 같은 내용이 포함될 수 있다.

첫째는 포트 연결 정보이다. 각 IP의 어떤 포트들이 서로 연결되어야 하는지를 의미한다. 포트 간에는 1:1의 연결이 될 수 있고, 여러 포트가 하나의 포트에 연결될 수도 있다. 또한 데이터 전송 위해 사용될 버퍼의 크기에 대한 정보도 포함되어야 한다. 버퍼는 각각의 포트 쌍마다 크기를 다르게 설정할 수 있어야 한다. 버퍼의 크기에 따라 시스템 성능에도 영향을 미칠 수 있기 때문이다.

둘째는 주소/데이터 또는 제어정보의 올바른 전달을 위한 정보이다. 즉 해당 데이터가 보내는 IP와 받는 IP에서 동일한 의미로 해석 되도록 해야 한다. 이를 위해서는 전송 내용(Contents)에 대한 상호 연결(mapping)이 필요하다. 제어 정보의 경우, 해당 정보를 전송하기 위해서는 어떤 형태로 변경되어 전송해야 하는 지에 대한 정보가 필요하다. 예를 들어 IP의 단일전송 제어정보가 상대 IP의 단일전송을 의미하는 제어정보로 전달

```
1 //// Matching Type
2 [Matching Type] I-Type;
3 //// buffer Size, signal name, MSB, LSB, signal name, MSB,
      LSB, address or wdata or rdata
4 [Port Mapping] 1 , HADDR , 31 , 0 , FA , 31 , 0 , $address;
5 [Port Mapping] 0 , HRDATA , 31 , 0 , BD , 31 , 0 , $rdata;
6 //// signal name, MSB, LSB, values, signal name, MSB, LSB,
      values, operation, characteristic, ip_name
7 [Transactions] Burst Length : 4 , HBURST , 2 , 1 , "01" ,
  FBST , 1 , 0 , "01" , READ , NO_INCR , ocnm;
8 [Transactions] Burst Length : 16 , HBURST , 2 , 1 , "11" ,
  FBST , 1 , 0 , "11" , WRITE , INCR , ocnm;
```

그림 2. AHB 마스터 IP와 OCN MNI간 인터페이스 회로 생성을 위한 CM의 일부

Fig. 2. A part of CM for interface circuits generation between AHB master and OCN MNI.

되기 위해서는 각 신호 간 연결 방법에 대한 정보가 필요하다.

셋째는 IP의 동작 특성 간 대응에 대한 정보이다. IP의 특정 동작이 상대 IP에서 어떤 동작과 연결되는 것이고, 연결 정보가 없을 경우에 어떻게 처리할 것인지에 대한 정보가 포함되어야 한다. 예로 마스터 IP는 하나의 시작주소와 여러 데이터가 전송되는 동작을 하고 슬레이브 IP는 매 전송마다 주소가 필요하다면, 이들의 연결을 위해서는 인터페이스 회로가 시작 주소에 근거하여 매 전송마다 유효한 주소를 생성해야 한다. 또한 주소를 사용하는 IP가 주소를 사용하지 않는 IP와의 연결에서는 주소를 어떻게 처리할 것인지가 결정 되어야 한다.

설계자는 CM을 이용하여 다양한 특성을 갖는 IP간 연결을 위한 인터페이스 회로를 생성할 수 있다. 본 논문에서는 CM을 GUI 또는 스크립트 파일을 이용하여 입력하도록 구현하였다. 그림 2은 AMBA AHB 마스터 통신 프로토콜과 OCN MNI (Master Network Interface)를 연결하기 위한 CM의 일부이다. 다음은 그림 2의 각 항목에 대한 설명이다.

1. [Matching Type]: IP의 동작 특성 간 연결 (Characteristics Matching)

어떤 특성을 갖는 IP간 연결인지에 대한 정보이다. IP의 동작 특성 구분은 표 1의 기준에 의해 구분된다. 본 논문에서는 IP의 연결 형태를 3가지로 구분하였다. 주소분리형 IP간의 연결(I 형), 주소분리형 IP와 주소일체형 IP의 연결(II 형), 주소분리형 IP와 비주소형 IP간의 연결(III 형)이다. AHB와 OCN MNI는 모두 주소분리형 프로토콜이기 때문에 I 형으로 설정하였다(그림 2. 2줄).

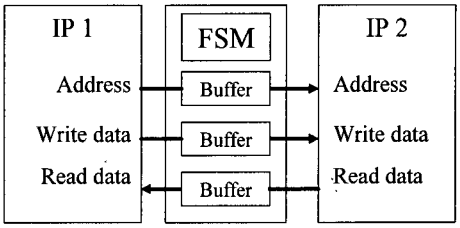


그림 3. *I* 형 연결을 위한 인터페이스 회로의 블록 다이어그램

Fig. 3. The block diagram of interface circuits for type *I* connection.

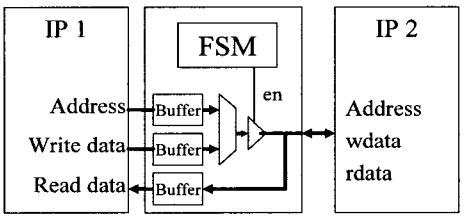


그림 4. *II* 형 연결을 위한 인터페이스 회로의 블록 다이어그램

Fig. 4. The block diagram of interface circuits for type *II* connection.

I 형 연결은 그림 3과 같이 매 포트의 쌍마다 버퍼가 생성되며(사용자 설정에 따라 생성되지 않을 수도 있음), 전달 지연된 주소/데이터는 버퍼(FIFO)에 저장된다. 데이터의 전송과 저장에 대한 제어는 두 IP의 통신 프로토콜에 근거하여 자동 생성되는 인터페이스 FSM에 의해 제어된다.

II 형 연결은 그림 4와 같이 하나의 포트와 여러 포트가 연결 되는 구조이다. PCI 프로토콜의 경우, 하나의 포트를 통해 주소가 먼저 전송되고 이어서 데이터가 전송된다. 그러나 주소분리형 IP는 주소와 데이터가 다른 포트를 통해 함께 전송되기 때문에 이들 간의 제어가 필요하다. 즉 포트에 전송되는 대상(주소, 쓰기데이터, 읽기데이터)이 무엇인지를 알 수 있어야 하며, 인터페이스 합성에 이를 위한 고려가 필요하다.

주소분리형 IP와 비주소형 IP간 연결에는 데이터 또는 동작이 1:1로 연결(matching)되지 않는 경우가 많다. 그림 5에서와 같이 AHB 버스 프로토콜에 DES 모듈을 연결을 위한 인터페이스 회로에는 AHB 여러 동작과 한번의 DES 모듈의 동작이 연결되어야 한다. 그림 5의 DES는 한 번의 암호화/복호화를 위해서 키(64bit), 평문(64bit)이 제공되어야 한다. 그러나 AHB 버스가 한번에 32bit의 데이터 전송이 가능한 경우에, 한 번의 암호화/복호화를 위해 4번의 AHB 데이터 전송이 필요하다. 따라서 본 논문에서는 설계자가 비주소형 IP의 각 포트에 주소를 할당하여 비주소형 IP를 제어하거나, 데이터

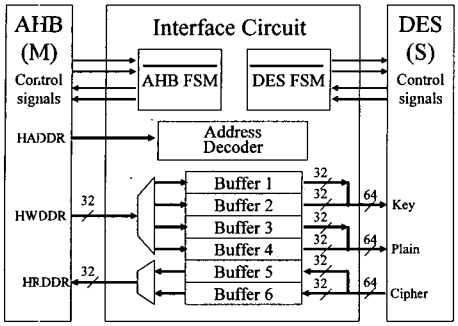


그림 5. *III* 형 연결을 위한 인터페이스 회로의 블록 다이어그램 (AHB와 DES)

Fig. 5. The block diagram of interface circuits for type *III* connection (AHB and DES).

```
//// Address/Data Buffer Mapping Information
[Port Matching] 1 , HWDATA , 31 , 0 , KEY , 63 ,32 ,
  HADDR , 7 , 0 , 0x00 , no-tick, valid-check , D;
[Port Matching] 1 , HWDATA , 31 , 0 , KEY , 31 , 0 ,
  HADDR , 7 , 0 , 0x04 , no-tick , valid-check , D;
[Port Matching] 1 , HWDATA , 0 , 0 , ENC_DEC , 0 , 0 ,
  HADDR , 7 , 0 , 0x10 , no-tick , valid-check , D;
[Port Matching] 1 , HWDATA , 0 , 0 , START , 0 , 0 ,
  HADDR , 7 , 0 , 0x14 , tick , valid-uncheck , C;
```

그림 6. AHB 마스터 IP와 DES간 인터페이스 회로 생성을 위한 CM의 일부 (III 형)

Fig. 6. A part of CM for interface circuits generation between AHB master and DES (Type III).

를 전송할 수 있도록 하였다. III 형 연결을 위한 인터페이스 생성에서는 두 IP의 동작이 1:1 연결이 아닐 수 있기 때문에 CPM을 구하지 않는다. 대신에 그림 5와 같이 각 통신 프로토콜의 역으로 동작하는 FSM을 생성하여, 각 IP의 역 FSM이 IP와의 통신을 담당하도록 하였으며, 데이터 전송은 버퍼(FIFO)를 통해 이루어지도록 하였다. 또한 주소 값에 의해 버퍼 또는 포트로 데이터 전송이 가능하도록 주소를 디코딩 하는 모듈이 추가되었다.

2. [Port Mapping]: 주소/데이터 포트 연결 정보 (Port Mapping)

두 IP간 포트 연결과 데이터 전송을 위한 버퍼 크기에 대한 정보를 포함한다. 각 IP의 어떤 포트끼리 연결이 되어야 하는지는 IP의 통신 프로토콜만으로는 알 수 없기 때문에 설계자가 해당 정보를 입력해야 한다. 이때 데이터를 임시로 저장하기 위한 저장소가 생성되는데, 이 저장소(FIFO)의 크기를 사용자가 각 포트의 쌍마다 따로 설정할 수 있다(그림 2. 4줄). 버퍼 크기, 신호 이름(마스터), MSB, LSB, 신호이름(슬레이브), MSB, LSB, 주소/쓰기데이터/읽기데이터의 순서로 기술된다. AHB의 "HADDR" 포트와 OCN MNI의 "FA" 포트가 연결되고, 두 포트의 크기는 32bit이며, 두 포트의 데이터 전송을 위해 설정된 버퍼의 크기는 '1'로 설정함을 의미한다. 버퍼의 크기를 '0'으로 하면 두 포트 간 전송을 위한 저장소는 생성되지 않는다. "$address"는 해당 포트를 통해 전송되는 대상이 주소임을 의미한다.

I 형, II 형의 연결에서는 각 포트의 연결만을 기술하나, III 형 인터페이스 회로를 위해서는 포트 또는 저장소에 주소를 할당하기 위한 정보가 추가되어야 한다.

그림 6은 AHB 마스터와 DES를 연결하기 위한 CM의 일부이다. 이때 포트의 연결은 비주소형 IP의 데이터 포트뿐만 아니라 제어를 위한 포트와도 연결될 수 있다.

그림 6의 2째 줄은 버퍼 크기 1, 주소(HADDR)의 하위 8 비트가 "0x00"일 때, HWDATA의 데이터가 KEY[63:32]로 전송됨을 의미한다. 5번째 줄의 "tick"은 한 사이클 동안만 값을 구동해야 하는 제어 신호를 의미한다. "tick"으로 표기된 신호는 한 사이클 동안만 마스터가 구동한 값을 유지하게 된다. 주로 제어 신호들을 구동하는데 사용된다. "valid-check"는 해당 버퍼에 데이터가 쓰이기 전에 값을 읽어가지 못하도록 하기 위한 부분이 인터페이스 회로의 생성에 추가된다. 줄의 마지막에 'C' 또는 'D'는 해당 포트가 제어신호(C)인지 주소/데이터인지(D)를 구분하는 정보이다.

3. [Transactions]: IP 제공 동작 정보 (Content Matching)

IP가 제공하는 버스트 동작 및 제어 신호 간의 연결 정보를 기술한다. 각 IP에서 지원하는 버스트 동작은 관련 동작의 쌍을 기술하고, 기술되지 않은 버스트 전송은 상대 IP의 단일전송과 연결된다. 그림 2의 "INCR"은 해당 버스트 전송이 매 전송마다 주소가 증가되어 전송됨을 의미하고, "NO_INCR"은 버스트 전송에 하나의 시작 주소만 전송함을 의미한다. 이 정보가 인터페이스 합성 알고리즘에 적용된다. 즉 "NO_INCR"의 특성을 갖는 IP가 마스터이고, 슬레이브 IP가 "INCR" 특성일 경우에, 마스터 IP가 주소를 구동하지 않기 때문에 인터페이스 회로에서 슬레이브 IP를 위해 주소 생성해야 하기 때문에 이를 합성 알고리즘에서 고려한다.

IV. 인터페이스 합성

1. IP 인터페이스 FSM

인터페이스 FSM은 사이클 단위의 각 포트의 동작을 나타내는 전이들과 이 전이들을 포함하는 상태로 구성된다. 인터페이스 FSM(P)은 그림 7과 같이 정의 된다. FSM의 임의 상태 q는 전이 집합(T)을 포함한다. 전이는 입출력 포트의 동작과 내부 변수의 동작으로 구성된다. 임의의 상태를 구성하는 전이들은 상태내의 다른 전이와 적어도 하나의 서로 다른 동작을 가져야 한다. 즉 모든 전이의 동작 $\{A_i, A_o, A_d, A_v\}$이 모두 같은 전이가 존재하지 않아야 한다. 인터페이스 FSM은 입력 통신 프로토콜로부터 변환이 쉽고, 인터페이스 회로의 지연 시간을 줄이기 위해 밀리(mealy) FSM으로 표현된다[13].

인터페이스 FSM $P = (Q, I, O, D, V, T \in q, q_0)$

$t = (A_i, A_o, A_d, A_v, q_N)$

Q: 제어 상태 집합 ($q \in Q$)

I, O: 입력, 출력 제어 신호 집합($i \in I, o \in O$)

D: 데이터 신호 집합(d $\in D$)

V: 사용자 정의 내부 변수($v \in V$)

T: 임의 상태의 전이 집합 ($t \in T$)

q_0: FSM의 초기 상태, q_N: 다음 상태

A_i, A_o, A_d, A_v: 입력, 출력, 데이터, 내부변수의 동작

그림 7. 인터페이스 FSM
Fig. 7. Interface FSM.

2. 인터페이스 프로토콜 기술 언어(IPDL)

IP의 통신 프로토콜의 기술을 위해 새로운 언어를 정의했고 이를 IPDL(Interface Protocol Description Language)로 부른다. IPDL의 특징은 IP의 동작(Transaction) 수준과 사이클 수준 (transfer)의 기술을 분리하여 복잡한 프로토콜의 기술을 단순화 할 수 있고, IP 내부의 상태를 기술하기 위해 의사변수(pseudo variables)를 사용하였고, 상위 수준 언어 형태로 사용자가 자유롭게 IP의 통신 프로토콜을 기술할 수 있다는 점이다.

여러 동작을 지원하는 IP들의 인터페이스 프로토콜 기술은 매우 복잡할 수 있고, 생성되는 인터페이스 회로가 비효율적이 될 수 있다. 본 논문에서는 복잡한 IP의 인터페이스 프로토콜의 기술을 동작 수준의 정보와 전이(transfer) 수준의 정보를 구분함으로써 통신 프로토콜을 간단하게 기술할 수 있다. 즉 동작(단일전송, 버스트 전송 등)에 따라 달라지는 정보를 기술하지 않고,

```
Define $new_transaction, $write;      -- ③
Interface ahbm {
      in  bit          HREADY;
      in  bit[31:0]    HADDR;
      in  bit[2:0]     HBURST;
                :
      initial   A;   -- ①
      clock   clk  rising;
      reset   rst  low;
AMBA_MASTER: BEHAVIOR   {
      int count;  -- ② // User Defined Variable
 A:    if ( $new_transaction == 1 )  --  ③ {
            HTRANS = "10";  // NONSEQ
            HADDR = $address; -- ④
            HBURST = $transaction;
            count = $burst_length;  -- ⑤
            if ($write == '1')   {
                  HWRITE = '1';
                  if ( HREADY == '1')
                        goto B;  -- ⑥
                  }
                  :
       else  {
             HTRANS = "00";
             goto A;
       }
```

그림 8. IPDL을 이용한 AMBA AHB 마스터 IP 인터페이스 프로토콜 기술의 일부
Fig. 8. A part of interface protocol description on AMBA AHB master IP with IPDL.

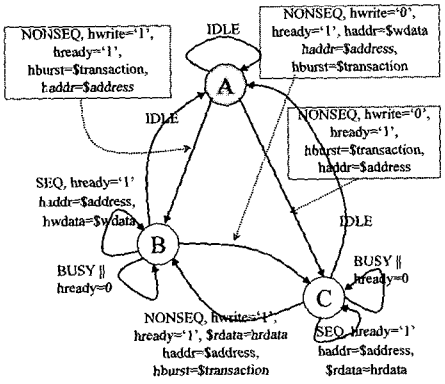
해당 신호에 대한 시간정보만을 기술하고 인터페이스 회로 생성 단계에서 신호들을 연결하여, 적은 양의 기술로 IP의 통신 프로토콜을 표현 할 수 있다.

그림 8은 온 칩 버스 프로토콜로 많이 사용되고 있는 AMBA AHB 마스터 IP의 통신 프로토콜 기술의 일부이다. 선언부분에는 IP의 포트 목록과 크기 및 FSM의 초기 상태 정보, 클록과 리셋 신호의 동작 특성을 기술한다. 사용자는 IP 내부의 상태/정보를 표현할 수 있는 의사(pseudo) 변수를 정의 할 수 있다. ①은 인터페이스 FSM의 초기 상태를 의미한다. ②는 IP의 동작기술에 필요한 사용자 정의 변수의 선언이다. "count" 변수는 버스트 동작의 수를 세기 위해 사용되는 내부 변수이다. "A"의 상태 이름 다음에 IP의 동작이 기술된다. 상태 이름 "A"부터 ⑥의 "goto" 문을 만날 때까지 한 사이클에서 동작함을 의미한다. 이들 동작 집합이 "A" 상태의 하나의 전이를 구성한다.

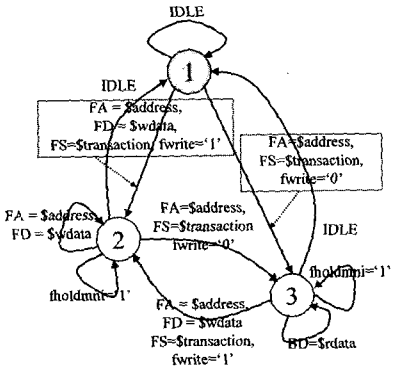
IP의 인터페이스 프로토콜을 표현하기 위해서는 IP 입출력 포트 이외에 IP 내부의 정보가 필요하다. 그러나 통신 프로토콜의 기술은 입출력 신호만을 사용해야 하므로, IP의 내부 조건을 표현하기 위해 의사(pseudo) 변수를 사용하였다. ③의 $new_transaction은 IP가 새

표 2. IPDL의 의사변수
Table 2. The pseudo variables in IPDL.

의사변수	설명
$address	IP가 전송하는 주소
$wdata	IP가 전송하는 쓰기 데이터
$rdata	IP가 전송하는 읽기 데이터
$transaction	트랜잭션 정보
$burst_length	버스트 동작 길이



(a) AHB Master FSM



(b) OCN MNI Slave FSM

그림 9. 자동 추출된 인터페이스 FSM
Fig. 9. Automatically Extracted interface FSMs.

로운 동작을 시작하는 내부 조건을 나타내기 위해서 사용한 의사 변수이다. 조건으로 표현된 의사 변수는 인터페이스 합성에 고려되지 않는다. 그러나 이러한 의사변수는 사용자의 프로토콜 기술에 대한 판독성(readability)을 향상시킬 것으로 기대한다. ④는 유효한 주소가 출력됨을 의미한다. AHB 마스터 프로토콜의 경우에 단일, 미지정(unspecified length), 4, 8, 16의 길이를 갖는 동작이 있다. 인터페이스 프로토콜의 기술에 이들을 각각 따로 기술하면 매우 복잡하게 된다. 본 논문에서는 동작에 따라서 달라지는 부분들을 의사 변수로 처리하여 인터페이스 프로토콜의 기술을 간단히 할 수 있다. 트랜잭션마다 달라지는 정보를 표현하지 않고, 해당 정보가 전송되는 시간정보만을 표현하고, 추후 CM의 정보를 이용하여 처리한다. ⑤의 "$burst_length"는 데이터 전송의 길이를 의미한다. 이 역시 CM의 정보를 근거로 처리된다. 표 2는 IPDL의 기술에 사용되는 의사변수 중 예약어로 사용되는 변수들에 대한 설명이다.

그림 9은 사용자의 기술(IPDL)에서 자동 추출된 인터페이스 FSM의 예이다. 그림 9(a)는 그림 8의 AHB 마스터 프로토콜에서 추출된 인터페이스 FSM이고, 그림 9(b)는 OCN MNI의 통신 프로토콜 기술로부터 추출한 인터페이스 FSM이다. 그림 8의 상태 "A"에서 ⑥에 이르는 동작이 그림 9 (a)의 상태 A에서 B로의 전이로 생성된다. "count = $burst_lenght" 등의 동작은 지면 관계상 생략하였다.

3. 인터페이스 합성 알고리즘

IP의 통신 프로토콜로부터 인터페이스 FSM을 생성하는 알고리즘은 [1]에서 제안된 방법을 사용하였으나, 연구 [1]과 입력이 달라 중복된 전이가 많이 생겼고, 따라서 불필요한 전이를 삭제할 수 있는 알고리즘을 추가하였으며, 또한 CM를 인터페이스 합성에 고려할 수 있도록 변경하였다. 인터페이스 합성에 사용되는 알고리즘의 근간이 되는 알고리즘의 정확성은 이미 논문[1, 13]을 통해서 검증되었다. 그것은 두 입력 FSM을 받아서 양쪽 모듈에 호환 가능한 FSM을 생성할 수 있음을 보였다. 제안된 알고리즘은 검증된 알고리즘을 데이터의 양이 상이한 경우에도 추가 정보를 이용하여 FSM을 생성할 수 있도록 변경하였다.

인터페이스 합성 알고리즘은 두 인터페이스 FSM $PA(Q_A, I_A, O_A, D_A, V_A, T_A \in q_A, q_{A0})$, $PB(Q_B, I_B, O_B, D_B, V_B, T_B \in q_B, q_{B0})$와 CM를 입력으로 하여 새로운 인터페이스(PI)를 생성한다. PI는 PA, PB의 동작을 모두 만족시킬 수 있는 CPM (Cross-Product-Machine)이다. 인터페이스 합성 알고리즘의 출력은 $PI(Q_P, I_P, O_P, D_P, V_P, T_P \in q_P, q_{A0B0})$와 각 상태의 전이 $tp(A_i, A_o, A_d, A_v, q_N)$이다.

*PI*의 입출력 포트는 두 입력 IP 포트의 합이다. 즉 $I_P = O_A \cup O_B$, $O_P = I_A \cup I_B$, $V_P = V_A \cup V_B$ 이고, D_P 는 사용자가 *CM*에서 설정한 포트 연결의 쌍이다. q_{A0B0} 는 두 입력 FSM의 초기상태의 조합으로 생성된다. Q_P, T_P 는 그림 10의 알고리즘에 의해서 생성된다. *PI*의 상태는 두 입력 FSM의 상태 정보와 버퍼의 크기 정보를 포함한다. 따라서 PI의 한 상태는 두 입력 FSM의 상태와 해당 상태에서 전송되지 않은 데이터 수[(q_A

$\in Q_A$), ($q_B \in Q_B$), B(D_P)]로 표현된다.

그림 10은 통신 프로토콜 *PA, PB, CM*을 입력으로, CPM을 생성하는 알고리즘이다. 두 입력 통신 프로토콜에 기술된 초기 상태에 근거하여 CPM의 초기 상태를 만든다. *PS*(임시 상태 집합)에 생성된 상태를 추가하고 *PS*에 더 이상 상태가 없을 때까지 다음을 반복하게 된다. T_A ={*t1*:q_A-->}는 입력 FSM의 상태 q_A의 전이집합을 의미한다. 각 전이에서 그의 역 동작 전이를 구한다(①). *PA, PB*의 출력 포트의 동작은 *PI*의 입력 포트의 동작이 되고, *PA, PB*의 입력 포트의 동작은 *PI*의 출력 포트의 동작이 되어야 하기 때문이다. *PA*와 *PB*의 역 동작 전이를 생성한 후, 이 역 동작 전이의 동작의 합이 *PI*의 새로운 전이가 된다. 생성된 전이가 유효한 전이인지 검증하는 과정을 거친다. 상대 IP에서 요구하는 주소 또는 데이터가 버퍼에 있거나, 해당 IP의 포트에서 데이터를 출력하는 경우에는 유효한 전이가 된다(②). 이때 매 전송에 주소를 요구하는 IP와 시작 주소만을 출력하는 IP간 연결의 경우에는, *CM*의 설정에 따라 요구하는 주소가 없어도 자동 생성된 주소가 있는 것으로 간주하여 유효한 전이로 처리한다. 생성된 전이가 유효하다면, 해당 전이에 의해 생성되는 상태를 계산해야 한다. 새로운 상태는 입력 FSM *PA, PB*의 관련 전이에 의해 결정된다. 상태 내의 버퍼에 저장된 데이터 수가 변경되었다면 이 정보를 새로 생성되는 상태에 고려한다. 예를 들어 해당 버퍼에 데이터가 더 쓰였다면, 새로운 데이터가 추가되었음이 고려되어야 한다(③). 새 전이에 의해 새로운 상태가 결정하고, 상태 집합($Q \cup PS$)에 새로운 상태와 동일한 상태가 없다면, 새로 생성된 상태를 *PS*에 추가한다.

새로 추가된 모든 전이의 동작은 서로 중첩되지 않아야 한다. 즉 같은 입력 조건에 서로 다른 동작을 하는 전이가 존재하지 말아야 한다. 따라서 생성된 전이의 입력 동작과 내부 변수 동작을 고려하여 중첩된 동작을 갖는지를 파악하고, 중첩된 동작이 있는 전이 중 하나만 남기고 모두 제거되어야 한다. 동일 조건을 갖는 전이를 제거 하는 기준은 데이터 전송이 많은 전이에 가중치를 두고, 가중치가 높은 하나의 전이를 남기고 나머지는 제거한다(④). 위 과정을 거치면 새로 생성된 인터페이스 상태 집합 *Q*가 생성된다. 생성된 상태에는 올바른 동작을 보장할 수 없는 상태가 생성 될 수 있다. 즉 어떤 상태로 진입한 후, 자기 자신의 상태만을 계속 반복하는 상태가 생성될 수 있다. 이런 상태들은 제거되어야 하고, 이들 상태로의 전이도 제거 되어야 한다(⑤).

```
입력: FSM(PA, PB), CM
Q = null; // Q는 PI의 상태 집합
PS = { [ PA0, PB0, B(Dp)] } // PS : 임시 상태 집합
// CurrentState = [qA, qB, B(Dp)] 처리중인 상태를 의미함
while PS != null do
 CurrentState := ps ∈ PS // PS의 임의의 상태 할당
 Determine TA ={t1:qA-->}, TB ={t2:qB-->}
 // qA, qB는 PA, PB의 상태
  for all t1 ∈ TA, t2 ∈ TB do
   t'1 := ComputeComplement(t1) -- ①
   t'2 := ComputeComplement(t2)
   if valid(t'1∪t'2, B(Dp) ) then -- ②
    B(Dp)' := ModifyCounter(B(Dp), t'1∪t'2 ) -- ③
    Transition := MakeTransition(t'1∪t'2, [q'A, q'B, B(Dp)'])
    // t'1∪t'2 이고 다음 상태가 [q'A, q'B, B(Dp)' ]인 튜플 생성
    AddTransition(Transition, CurrentState);
    // 현재 처리 상태에 생성된 튜플 추가
    if [q'A, q'B, B(Dp)']  (Q ∪ PS) then
       Add [q'A, q'B, B(Dp)'] to PS
    end if
   end if
 end for
 Prune_Transition (CurrentState) -- ④
 Add CurrentState to Q and remove from PS
end while
Prune_DeadState(Q) -- ⑤
```

그림 10. 자동 추출된 인터페이스 FSM
Fig. 10. Automatically Extracted interface FSMs.

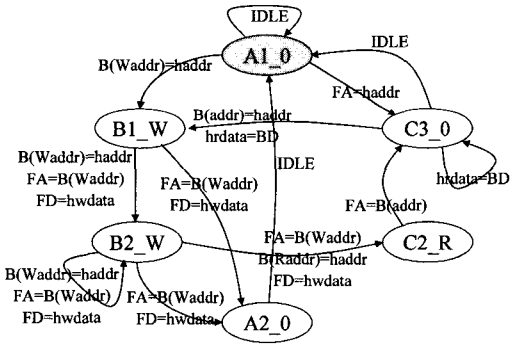


그림 11. AHB 마스터와 OCN MNI를 위해 생성된 인터페이스 FSM
Fig. 11. A generated interface FSM for AHB master and OCN MNI.

그림 11는 그림 9의 AHB의 마스터 통신 프로토콜, OCN MNI의 통신 프로토콜, CM(그림 2)를 입력으로 자동 생성된 인터페이스 FSM이다 (데이터 전송 대기 상태가 추가로 3개의 상태가 더 생성되었으나, 지면 관계상 생략함). 각 상태 이름은 입력 프로토콜의 상태 이름과 관련 있다. 상태 이름은 '마스터 IP 상태', '슬레이브 IP 상태', '_', '주소 버퍼의 상태'로 구성된다. 'W/R'은 각각 쓰기 주소와 읽기 주소를 의미한다. 따라서 "B1_W"는 AHB는 상태 'B', OCN MNI는 상태 '1', 그리고 버퍼에 쓰기 주소가 저장되어 있음을 의미한다.

V. 실 험

본 논문에서는 IP의 통신 프로토콜과 *CM*를 입력으로 인터페이스 회로를 자동 생성 툴을 구현하였다. IP의 통신 프로토콜을 위한 파서는 JavaCC[14]를 이용하여 구현하였고, 툴은 약 16,000라인의 자바로 구현하였다.

1. 다양한 프로토콜의 인터페이스 기술

SoC의 시스템 버스로 많이 사용되고 있는 AHB, APB, BVCI, OCN 등과 PCI, DES의 프로토콜을 기술하였다. 해당 IP의 IPDL 기술에는 IP 또는 버스 프로토콜에서 지원하는 모든 동작이 기술되지는 않았다. 오류 관련 프로토콜이나, 작업 취소 동작 등은 고려하지 않았다. IP의 동작 기술에는 버스트 전송, 파이프라인 전송, 동작(Transaction)간 중첩 등은 포함되어 있다. 표 3은 해당 IP 프로토콜을 기술한 IPDL 기술을 정리한 표이다. 논문 [2]에서는 AHB 마스터 쓰기 동작을 기술하는데, 18개의 상태와 40개의 전이가 필요하였으나, 본 논문에서는 동작 정보를 CM로 분리하여 기술함으로써 오류 처리를 제외한 쓰기, 읽기 동작을 기술하는데 3개의 상태와 11개의 전이로 표현할 수 있었다. 표 3은 AMBA AHB 마스터와 OCN MNI의 연결을 위한 인터페이스 합성 예이다. 여러 AHB 마스터 프로토콜 기술을 입력으로 하고, OCN MNI 프로토콜은 같은 기술을 사용하여, 그 합성 결과를 비교하였다. AHB 마스터의 프로토콜 기술을 기존의 방법과 제안된 방법을 이용하여 기술하였다.

표 3. 다양한 AHB 마스터 프로토콜 기술과 그에 따른 인터페이스 합성 결과

Table 3. Various Interface descriptions on AHB master protocol and it's synthesis results.

	AHB 마스터 프로토콜 기술			합성 결과			
	지원 Transactions	상태 수	전이 수	상태 수	전이 수	면적 (slice)	동작 속도 (MHz)
1	2 동작-쓰기 (길이: 4, 미지정)	3	13	6	44	60	201.288
2	4동작 - 쓰기/읽기 (길이: 4, 미지정)	5	23	9	58	145	158.239
3	8동작 - 쓰기 (길이: 4, 8, 16, 미지정)	17	56	116	999	579	109.716
4	16동작 - 모든 쓰기/읽기 (error, split, abort 제외)	3	11	9	53	103	180.799

표 3의 예 1, 2, 3은 각각 2개의 동작만을 지원하는 AHB 기술(4, 미지정 길이의 쓰기 동작), 4개의 동작을 지원하는 AHB의 기술(4, 미지정 길이의 쓰기/읽기 동작), 8개의 동작을 지원하는 기술(모든 쓰기 동작)을 입력으로 사용하였다. 기술하려는 동작의 수가 많아짐에 따라 상태의 수와 전이의 수도 엄청나게 증가한다. 그에 따라 자동 생성되는 인터페이스 회로의 면적과 동작속도도 많은 영향을 받는다. 그러나 제안된 방법을 이용한 4번째 예에서 알 수 있듯이, AHB 마스터 프로토콜의 모든 동작(쓰기, 읽기 포함)을 기술함에도 불구하고, 단지 3개의 상태와 11개의 전이만을 사용하였다. 또한 합성된 인터페이스 회로의 면적이나 동작속도 면에서도 기존의 기술 방법보다 훨씬 적은 면적과 빠른 속

표 4. 다양한 통신 프로토콜의 인터페이스 FSM

Table 4. Interface FSMs on various communication protocols.

IP 종류	Port 수	상태 수	전이 수	프로토콜 범주
AHB(M)	15	3	11	A
AHB(S)	15	3	9	A
APB(S)	8	3	7	A
OCN MNI	17	4	16	A
OCN SNI	18	3	13	A
BVCI Initiator	18	4	11	A
BVCI Target	18	3	8	A
PCI (M)	14	4	10	B
PCI(S)	13	4	10	B
DES	9	3	5	C

표 5. 다양한 통신 프로토콜을 사용하는 IP간 연결을 위한 인터페이스 자동 합성

Table 5. The automatic interface synthesis for connection between IPs with various communication protocols.

이름	연결형	마스터	슬레이브	포트 수	상태 수	전이 수	면적 (Slices)	동작 속도(MHz)
AHB OCN	*I*	AHB	OCN	30	9	56	103	180.799
		OCN	AHB	31	11	57	233	143.937
BVCI AHB	*I*	BVCI	AHB	31	6	34	61	355.694
		AHB	BVCI	31	20	74	130	264.277
AHB APB	*I*	AHB	APB	21	3	9	7	581.564
AHB PCI	*II*	AHB	PCI	26	6	30	72	383.362
		PCI	AHB	25	25	162	212	144.436
AHB DES	*III*	AHB	DES	13	6	18	162	297.044

(Target Device :Virtex2 xc2v6000)

도로 동작함을 알 수 있다. 표 4에서는 여러 다른 특성을 갖는 프로토콜의 기술도 3~4개의 상태로 표현할 수 있음을 보였다.

2. 인터페이스 회로 자동 합성 결과

표 5는 표 4의 인터페이스 기술을 입력으로 생성한 인터페이스 회로의 특성을 정리한 표이다. 주소를 사용하는 IP와 주소를 사용하지 않는 IP간 연결(AHB-DES), 주소/데이터가 여러 포트로 전송되는 IP와 하나의 포트로 전송되는 IP간 연결(AHB-PCI)을 위한 인터페이스 회로를 생성하였고, 매 전송마다 주소와 데이터가 함께 전송되는 IP와 하나의 시작주소에 여러 데이터가 전송되는 IP간의 연결(AHB-OCN)에 주소 생성을 위한 모듈이 추가된 인터페이스 회로가 생성된다. 실험을 통해 다양한 동작 특성을 갖는 IP간 인터페이스 회로의 생성이 가능함을 보였다. 실험 결과에 따르면 사용자가 직접 설계한 인터페이스 회로와 비교하여 평균적으로 자동 생성된 회로의 면적이 2배 정도 크고, 동작 속도는 1.5배 정도 느린 결과를 보였고, 자동 생성 코드를 실제 시스템에 직접 적용이 가능하였다. 자동 생성된 AHB-OCN 인터페이스 회로는 H.264 디코더 [15]을 통해 동작 검증하였다. 그림 12는 H.264 디코더의 일부이다. "AHB2OCN"은 AHB 마스터 모듈과 OCN MNI 간 통신을 위한 인터페이스 회로이고, "OCN2AHB"는 OCN SNI와 AHB 슬레이브와 통신을 위해 자동 생성된 인터페이스 회로이다. OCN 프로토콜은 읽기 전송에 패킷 정보(데이터를 요청한 마스터 번호)를 같이 보내야 한다. 그러나 AHB에 이런 패킷 정보를 고려하지 않기에, 인터페이스 회로가 패킷 정보를 관리하여야 한다. CM에 정보를 추가하여, 자동 생성된 인터페이스 회로가 이를 고려하도록 회로를 생성할 수 있다.

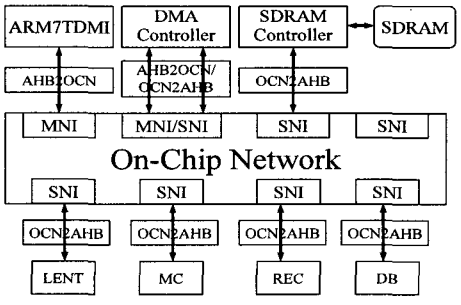


그림 12. H.264 디코더 시스템에서 자동 생성된 인터페이스 회로

Fig. 12. Automatic generated interface circuits in H.264 decoder system.

VI. 요약 및 향후 과제

본 논문에서는 IP의 통신 프로토콜 정보와 IP들의 특성 간 대응 정보(*CM*)를 이용하여, 주소를 사용하는 IP와 주소를 사용하지 않는 IP간 연결, 데이터가 여러 포트로 전송되는 IP와 하나의 포트로 전송되는 IP간 연결, 매 전송마다 주소와 데이터가 함께 전송되는 IP와 하나의 시작주소에 여러 데이터가 전송되는 IP간의 연결에도 인터페이스 회로의 자동 합성이 가능함을 보였다. 복잡한 IP의 통신 프로토콜을 의사변수를 이용하여 간단하게 기술할 수 있는 방법도 제안하였다. 또한 효과적인 인터페이스 합성을 위해 다양한 통신 프로토콜을 데이터 전송 측면에서 세 종류의 형으로 분류하고, 어떤 특성을 갖는 IP간 연결인지를 *CM*에 포함하여, 연결 IP의 특성에 따라 다른 형태의 인터페이스 회로가 생성될 수 있도록 하였다. 본 논문은 SoC 설계에 인터페이스 회로 설계의 오류를 줄일 수 있을 뿐 아니라, 빠른 인터페이스 회로 생성이 가능하여, 시스템 설계 시간을 줄이는 데도 기여할 것으로 예상된다.

향후 과제에는 각 IP가 서로 다른 클록을 사용하는 경우를 위한 인터페이스 회로 생성, 생성되는 인터페이스 회로의 최적화를 위한 알고리즘에 대한 연구도 진행중이다.

참 고 문 헌

[1] Vijay D'silva, S. Ramesh and Arcot Sowmya, "Bridge Over Troubled Wrappers: Automated Interface Synthesis," VLSI Design, 2004. Proceedings. Page(s):189 - 194, 17th International Conference on 2004.

[2] Vijay D'silva, S. Ramesh, "Synchronous Protocol Automata: A Framework for Modeling and Verification of SoC Communication Architectures," Proceedings of the Design, Automation and Test in Europe Conference and Exhibition

[3] Seawright, A., and Brewer, F., "Clairvoyant: a synthesis system for production based specification," IEEE Trans. Very Large Scale Integer. (VLSI) Syst., pp. 172-185, 1994, 2

[4] ARM. Advanced micro-controller bus architecture specification http://www.arm.com

[5] Kangmin Lee, Se-Joong Lee, Hoi-Jun Yoo, "A distributed crossbar switch scheduler for on-chip networks," Custom Integrated Circuits Conference, 2003, page 671-674, Proceedings of the IEEE 2003 21-24 Sept.

[6] Tom Shanley, Don Anderson, "PCI System Architecture", Addison Wesley, 1998.
[7] Jan Madsen and Bjame Hald, "An Approach to Interface Synthesis," in Proc. of ISSS, 1995.
[8] Yin-Tsung Hwang and Sung-Chun Lin, "Automatic Protocol Translation and Template Based Interface Synthesis for IP Reuse in SoC," the 2004 IEEE Asia-Pacific Conference on Circuits and Systems, Pages: 565-568, December 6-9, 2004.
[9] Akella, J., and McMillan, K., "Synthesizing converters between finite state protocols," Presented at the int. Conf. on Computer-Aided Design, 1991.
[10] V. Androutsopulos, D.M. Brookes and T.J.W. Clarke, "Protocol converter synthesis," IEE Proc.-Comput. Digit. Tech. Vol. 151, No. 6, November 2004.
[11] ChangRyul Yun, KyoungSon Jhang, "An Interface Protocol Component Modeling Language," 15th IEEE ASIC/SOC International Conference, 2002 Sep, NY, USA
[12] V.S.I. Alliance. http://www.vsi.org.
[13] I. De Alfaro and T. A. Henzinger. "Interface automata," Joint 8th ESEC and 9th ACM Symposium on February, 2001.
[14] "JavaCC - Java Compiler Compiler [tm]-The Java Parser Generator", http://javacc.dev.java.net/
[15] Soo Yun Hwang, Kyoung Son Jhang, June Young Chang, Han Jin Cho, "An Implementation of a Flexible OCN based SoC Platform targeting H.264 Decoder," The 13th Korean Conference on Semiconductors Pages: 917-918, 23-24 February 2006.

저 자 소 개

윤 창 열(정회원)
2000년 한남대학교 컴퓨터공학과 졸업 (학사).
2002년 한남대학교 컴퓨터공학과 졸업 (석사).
2006년 충남대학교 컴퓨터공학과 졸업 (박사).
2006년 9월~현재 충남대학교 BK21 차세대정보기술 SW인력양성사업단
<주관심분야 : 설계자동화, 인터페이스 자동합성, SoC 설계>

장 경 선(종신회원)
1986년 서울대학교 컴퓨터공학과 졸업(학사).
1988년 서울대학교 컴퓨터공학과 졸업(석사).
1995년 서울대학교 컴퓨터공학과 졸업(박사).
2001년 9월~현재 충남대학교 전기정보통신공학부 컴퓨터 전공 부교수
<주관심분야 : 컴퓨터구조, 설계자동화, SoC 설계>