NoC 시스템을 위한 하드웨어 컴포넌트를 위한 인터페이스 에이전트의 자동 생성

박정태*, 장경선*, 프랑코 삐리**
*충남대학교 컴퓨터공학과
**이태리 피렌체대학교 전자공학과
e-mail : sun@cnu.ac.kr

An Interface Agent Generation for Hardware Components in a NoC System

Jung-Tae Park*, Kyoung-Son Jhang*, Franco-Pirri**
*Dept. of Computer Engineering, Chung-Nam National University
**Dept. of Electronics, University of Florence, Italy.

요 약

NoC 시스템은 기본적으로 서로 다른 클럭 도메인에서 동작하는 여러 버스 시스템들이 NoC 를 통해서 연결되는 것으로 간주할 수 있다. NoC 에 다른 버스 인터페이스 IP 를 부착하려면 별도의 래퍼를 사용해야 하며, 면적과 지연시간이 추가되는 것이 일반적이다. 본 논문에서는, 추가적인 래퍼의 필요성을 제거하기 위하여, 주어진 버스 인터페이스에 맞는 인터페이스 에이전트 또는 네트워크 인터페이스를 자동 생성하는 방법을 제안한다. 이를 위하여, 한가지 NoC 시스템을 위해 표준적인 패킷 포맷을 정의하였으며, 거기에는 패킷에 대한 라우팅 정보 뿐 아니라, 여러 종류의 버스 프로토콜의 데이터, 주소, 제어 정보도 포함될 수 있도록 정의되었다. 그리고, 인터페이스 에이전트는 표준 패킷 포맷과 특정 버스 인터페이스 프로토콜 간의 변환 작업을 수행한다. 실험을 통해서, PVCI, WISHBONE, AHB, OCP 와 같은 몇 가지 버스 인터페이스에 대해 자동생성된 네트워크 인터페이스들 간에, 표준 패킷 포맷을 이용한 데이터 통신이 중요 정보의 손실 없이 잘 이루어짐을 보인다.

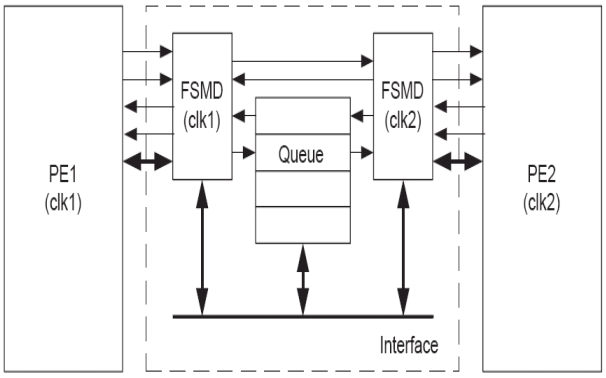
1. 서론

SoC (시스템 온 칩) 시스템의 복잡도의 증가로, NoC(Network On Chip) 기반 SoC 설계방법이 버스 기반 설계 방법의 대안으로 제안되었다 [1]. NoC 기반 시스템과 버스 기반 시스템 간에 몇 가지 일반적인 차이점을 기술하면 다음과 같다. 1) 먼저 데이터 전송이 네트워크 인터페이스를 통해서 패킷 형태로 전달된다. 2) 한 순간에 여러 마스터들이 동시에 NoC 를 통해 데이터 전송을 할 수 있다. 3) 요구 전송과 응답 전송이 분리되어서 수행됨으로써 대역폭 낭비를 줄일 수 있다. 4) 일반적으로 NoC 시스템은 여러 가지 버스 서브시스템들이 NoC 를 통해 통신하는 형태라고 간주할 수 있다. 5) NoC 의 네트워크 인터페이스는 나름대로의 고정된 인터페이스 프로토콜을 갖거나, AMBA [2], VCI(Virtual Component Interface) [3], OCP [4] 와 같은 많이 사용되는 실질적인 표준 인터페이스 프로토콜을 고정된 인터페이스로 취하기도 한다.

일반적으로 인터페이스 에이전트가 고정된 한가지 인터페이스 프로토콜 A 만 허용하기 때문에, 새로운 IP 나 버스 프로토콜 B 를 통합하려고 할 경우에는 A 와 B 간의 래퍼를 추가적으로 설계해야 한다. 이렇게 할 경우에 추가적인 래퍼로 인해서 면적과 지연 시간의 추가를 가져올 수 있다. 본 논문에서는 이를 개선하기 위한 방법을 제시하려고 한다. 즉, 바로 원하는 B 프로토콜을 수용하는 인터페이스 에이전트를 자동 생성하도록 하는 것이다.

네트워크 인터페이스 회로를 자동적으로 구성하는 방법으로 Passerone [8]에 의한 product FSM 및 pruning 기법을 사용할 수도 있으나, 이 방법은 추후에 pruning 방법에 따라 생성되는 회로의 크기가 지나치게 커질 수 있는 한계를 가지고 있다. 본 논문에서는 그런 한계를 극복하기 위해, 수작업으로 설계된 인터페이스 모듈들을 효과적으로 사용할 수 있는 방식을 사용하였다. 즉, FSMD (FSM with data path)와 함께 큐나 FIFO 를 사용하는 방법이다 [19, 20]. 그림 1 은 FSMD 와 큐를 사용한 인터페이스 회로 구조를 보여준다. FSMD 는 FSM + data path 를 의미하며, 논문 [9]에서는 FSMD 를 해당하는 인터페이스 프로토콜 기술로부터 생성하였다. 또한 논문 [10]의 경우에도 FSMD 와 유사한 IPC(interface protocol component)를 인터페이스 프로토콜로부터 생성하였다. 별도의 기술 언어로부터 FSMD 나 IPC 를 생성하는 것은 나름대로 장점을 가질 수도 있지만, 한편으로는 설계자가 만든 인터페이스 회로보다 특히 면적 면에서 불리한 설계 결과를 가져오는 원인이 된다. 논문 [9]의 FSMD 방법은 FSMD 간의 통신을 필요로 하나, 논문 [10]의 IPC 방법은 IPC 간의 통신이 필요하지 않으며, IPC 는 해당

인터페이스 신호들과 FIFO 사이에 필요한 데이터 전송이나 제어 동작만을 수행한다. 본 논문에서는 면적 면에서 불리한 인터페이스 기술 언어로부터 FSM 을 생성하는 방식을 지양하고, 설계자가 직접 설계한 인터페이스 모듈을 기반으로 NoC 용 인터페이스 에이전트를 생성하는 방법을 제안한다. 인터페이스 에이전트의 구조는 FSMD 나 IPC 의 경우와 유사하게 두 개의 인터페이스 모듈과 FIFO 들로 구성된다. 하지만, 기존의 FSMD 나 IPC 와는 달리, 각 인터페이스 모듈은 VHDL 이나 Verilog 모듈로 설계자가 직접 설계하거나, 이미 설계된 것을 사용하기 때문에, 두 개 이상의 FSM 도 포함할 수 있다. 예를 들면, OCP 의 경우에는 요구 전송과 응답 전송이 분리되어 있어서, 각각 FSM 으로 기술되어야 한다.



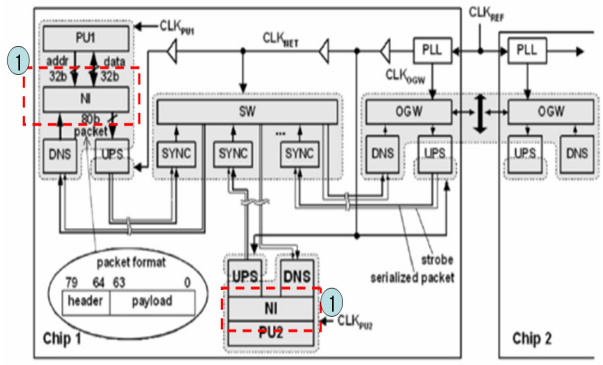
(그림 1) FSMD 기반 인터페이스 회로 구조 [9].

2 절에서는 NoC 와 인터페이스 에이전트에 관해 설명하며, 3 절에서는 본 논문에서 제안하는 표준 패킷 포맷의 정의와 인터페이스 에이전트 구조에 관해 설명한다. 4 절에서는 실험과 결과가 제시되어 있고, 마지막 절에서는 논문의 요약과 결론의 제시한다.

2. NoC 와 인터페이스 에이전트

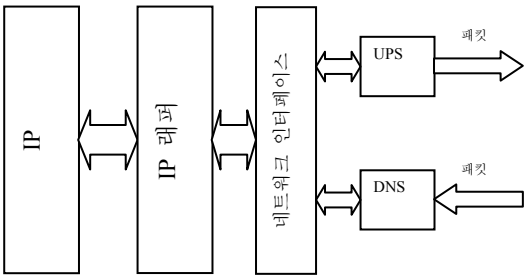
(그림 2)는 BONE 2.0 [6] 이라고 하는 NoC 의 구조를 보여 준다. BONE 은 네트워크인터페이스인 NI (Network Interface)들과 UPS (Up Sampler), DNS (Down Sampler), FIFO,클럭동기화 모듈인 SYNC(synchronizers), 네트워크 스위치인 SW(network switches)와 오프칩 게이트웨이인 OGW (Off-chip gateway) 등으로 구성된다. PU1 과 PU2 는 PU(Processing Unit)에 해당하며, NoC 에 부착된 CPU 나 IP 와 같은 계산 노드를 의미한다. UPS 는 업 샘플러라는 의미이며, 80 비트 패킷을 4 개의 20 비트 플릿(flit)으로 나누어서 스위치로 보내는 기능을 수행한다. DNS 는 다운 샘플러라는 의미이며, 스위치로부터 플릿을 받아서, 패킷을 조립하는 역할을 수행한다. BONE 은 대개 스타 구조를 가지며, 스위치와 패킷에 포함된 라우팅 정보를 기반으로한 동적인 패킷 스위칭 기법을 사용한다. 본 논문에서 대상으로 하는 BONE 구조는 BONE 3.0 으로 88-비트 패킷을 사용하는 점만 차이가 있다.

인터페이스 에이전트는 바로 (그림 2)에 나온 NI 부분에 해당하며, 해당 부분을 자동 생성하는 것이다. 따라서, 인터페이스 에이전트는 한쪽으로는 UPS 및 DNS 와 정보를 교환하고, 다른 쪽으로는 특정 인터페이스 프로토콜 또는 버스 프로토콜과 정보를 교환한다. 즉, UPS 쪽으로는 특정 인터페이스 프로토콜의 데이터를 받아서 88 비트 패킷으로 만들어서 보내는 역할을 하고, DNS 쪽에서는 88 비트 패킷을 받아서 특정 인터페이스 프로토콜의 데이터로 보내는 역할을 수행한다. 또한, 그림 2 의 네트워크 인터페이스는 스위치인 SW 에 의해 라우팅에 사용될 수 있도록, 특정 인터페이스 프로토콜에서 오는 주소 정보를 기반으로 라우팅 정보를 패킷에 넣을 수 있도록 주소 맵 정보를 포함하고 있다.

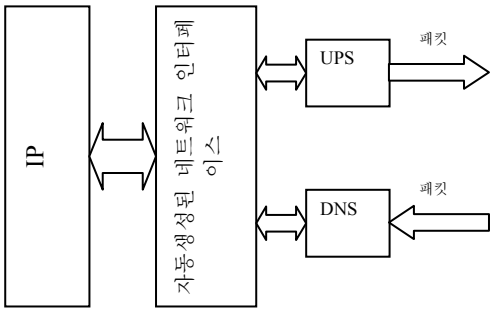


(그림 2) NoC 의 구조 [6].

NoC 시스템이 한 종류의 IP 에 대한 인터페이스만 제공하는 경우에는 다른 IP 를 접속해야 하는 경우에는 그림 3(a)와 같이 네트워크 인터페이스 외에 해당 IP 를 위한 래퍼를 별도로 설계해서 붙여야 한다. 본 논문에서 이와 같은 구조를 개선하여 IP 가 UPS/DNS 에 하나의 인터페이스를 통해 직접 연결될 수 있도록 각 IP 별 네트워크 인터페이스를 생성하는 방법을 제시하고자 한다. 그렇게 함으로써, 불필요한 지연이나 이중적인 버퍼 사용을 없앨 수 있게 된다. 그림 3(b)에서 보여 주는 자동 생성된 네트워크 인터페이스가 래퍼(wrapper)의 기능까지 수행하게 된다.



(a) 고정된 네트워크 인터페이스



(b) 네트워크 인터페이스와 주변 환경

(그림 3) 고정된 네트워크 인터페이스와 자동생성된 네트워크 인터페이스 간의 차이

3. 표준 패킷 포맷의 정의 및 네트워크 인터페이스의 구조

표준 패킷 데이터 포맷은 아래 <표 1>와 같이 구성할 수 있다. 88 비트의 패킷이 일반화된 필드 이름들($trans, $size, $be, $burstlen, $address, $wdata 등)에 의해 사용되는 한가지 방식을 보여주고 있다. 패킷 데이터 포맷은 2 가지로 나누어져 있다. 즉, 마스터 패킷 포맷과 슬레이브 패킷 포맷이 있다. 마스터 (슬레이브) 패킷 포맷은 마스터 (슬레이브) IP 가 슬레이브 (마스터) IP 에게 보내는 데이터를 표현하기 위해 사용된다. <표 1>은 마스터패킷포맷의 예이며, 슬레이브패킷포맷은 지면상 생략되었다. <표 1>에서 보는 바와 같이 마스터 패킷은 몇가지 버스 프로토콜를 포함할 수 있도록 패킷 엔코딩 포맷을 갖는다. 그래서, 2 가지 형태의 엔코딩 형식을 갖게 되었다. AHB 는 87, 86 번 비트 값이 00 이고, 나머지 WISHBONE[5], PVCI[3], OCP[4]는 87, 86 번 비트 값이 01 이다. 두 부류 간의 차이는 85, 84, 71, 70 번째 비트들을 채우는 정보에 차이가 있을 뿐이다. <표 1>에서, $froute 는 마스터 패킷의 라우팅 정보 필드를 의미한다.

여러 가지 다른 인터페이스를 지원하기 위해서는 해당 인터페이스에 나오는 포트나 비트와 표준 패킷 포맷에 나오는 일반화된 필드 이름 간의 변환을 정의해야 한다. <표 1>의 표는 그 예를 보여주고 있다. 4 개의 마스터 신호를 마스터 패킷 포맷으로 변환하는 방식이 <표 1>에 나타나 있다. ← 표시는 해당 인터페이스 정보로부터 마스터/슬레이브 표준 패킷 포맷의 해당 비트들을 채우는 방식을 표현한 것이다. 예를 들어 AHB 의 경우에, 87, 86 번 비트는 00 이 되며, HTRANS 정보가 85, 84 번 비트에 매핑되고, HSize 의 하위 2 비트가 71,70 번째 비트에 매핑됨을 의미한다.

일반적으로 버스 개념에서는 마스터에서 슬레이브에게 요구 전송을 보낼 때 많은 정보를 보내게 된다. 그러나, 대상으로 하고 있는 NoC 에서는 88 비트의 패킷을 사용하기 때문에, 모든 정보를 보낼 수 없을 경우도 있다. 예를 들면, <표 1>에서 볼 수 있듯이 AHB 의 HPROT 정보는 잘 사용되지 않는 정보이므로 디폴트 값으로 처리되어 마스터 패킷에는 표시되지 않는다. 하지만, AHB 슬레이브로 전달될 경우에는, 슬레이브 네트워크 인터페이스에서 디폴트 값인 '0000'이 자동으로 생성되도록 해야 한다. 또한 HSIZE 도 3 비트이지만, 본 논문에서는 데이터 버스인 HWDATA, HRDATA 가 모두 32 비트이므로, 그 기본 값은 '010' (32 비트 데이터 너비를 의미함)이다. 하지만, 모두 전송하기에는 패킷에 공간이 없으므로 하위 두 비트만 전송하고, 받은 쪽에서 상위 비트 '0'을 기본 값으로 복원하도록 네트워크 인터페이스를 구성해야 한다.

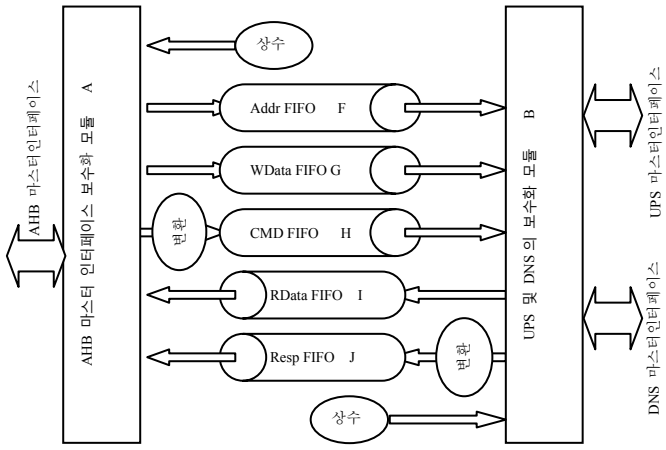
<표 1>의 패킷 포맷과 매핑 정보는 전혀 새로운 종류의 IP 를 도입할 경우에 새로운 종류의 정보를 표시하기 위해 변경이나 추가가 필요할 수도 있다. 또한, 현재는 요구 채널과 응답 채널이 분리되어 있는 프로토콜로 OCP 만을 포함시켰으나, 추후 연구에서 AXI [7], OCP 의 기본 확장/버스트 확장 신호 등도 포함시킬 예정이다. 따라서, 사후에 필요한 프로토콜의 추가를 위해서, 현재 마스터 패킷의 87 번 비트는 항상 0 으로 지정하도록 되어 있다

<표 1> 표준 마스터 패킷 포맷 정의

Bit	Master Packet FPacket[87:0]	← AHB	← WISHBN	← PVCI	← OCP basic
[87]	0	0	0	0	0
[86]	0 (1)	0	1	1	1
[85]	$trans[1] ($be[3])	HTrans[1]	SEL[3]	BE[3]	1
[84]	$trans[0] ($be[2])	HTrans[0]	SEL[2]	BE[2]	1
[83:72]	$froute[11:0]				
[71]	$size[1] ($be[1])	HSize[1]	SEL[1]	BE[1]	1
[70]	$size[0] ($be[0])	HSize[0]	SEL[0]	BE[0]	1
[69]	$eop	1	1	EOP	1
[68]	$burstlen[2]	HBurst[2]	0	0	0
[67]	$burstlen[1]	HBurst[1]	0	0	0
[66]	$burstlen[0]	HBurst[0]	0	0	0
[65]	$lock	HLOCK	LOCK	0	MCmd
[64]	$writeread	HWRITE	WE	not RD	MCmd 대응
[63:32]	$wdata[31:0]	HWDATA	DOUT	WDATA	MData
[31:0]	$address[31:0]	HADDR	ADR	ADDRESS	MAddr

(그림 4) AHB 마스터를 NoC 에 붙이기 위해서

필요한 AHB 마스터 네트워크 인터페이스 모듈의 개략적인 구조를 보여준다. AHB 마스터로부터 오는 데이터를 받아들일 수 있는 AHB 마스터에 대한 보수화 (complement) 동작을 수행하는 보수화 모듈과 정방향 FIFO 3 개(주소 Addr FIFO, 쓰기 데이터 WData FIFO, 명령 CMD FIFO)와 역방향 FIFO 2 개(읽기 데이터 RData FIFO, 응답 Resp FIFO), 그리고 UPS 와 DNS 에 대한 보수화 모듈로 구성되어 있다. 일부 필드의 값들은 상수 값들로 채워지기 때문에, ((그림 4))에도 상수로 채워지는 신호들이 있음을 보여주고 있다.



(그림 4) 개괄적인 AHB 마스터 네트워크 인터페이스의 구조

4. 실험

<표 2>는 수작업에 의해 설계된 네트워크 인터페이스 모듈 (이 경우에도 보수화 모듈은 이미 수작업으로 설계된 것이므로 재사용됨)과 자동 생성 프로그램으로 생성된 네트워크 인터페이스 모듈 간의 면적 및 동작 속도를 비교하고 있다. 본 실험의 목적은 자동 생성된 네트워크 인터페이스가 면적이나 동작 속도 면에서 매우 불리한 방식이 아님을 보이기 위한 것이다. 그에 따라서, 한가지 전제는 수작업 설계도 이미 설계된 보수화 모듈을 사용하고, FIFO 를 중간에 사용하는 구조로, 자동 설계의 경우와 동일한 구조를 사용한다. 다만, FIFO 의 사용 방식, FIFO 주변에 있는 변환 모듈의 기술 방법이나 위치에 있어서 차이를 보일 수 있다. 따라서, 전반적으로 동작 속도나 면적 면에서 큰 차를 보이고 있지 않음을 알 수 있다. 면적(#LUTs- LUT 의 수)의 대부분이 RAM 부분(즉, FIFO 부분)임을 알 수 있으며, Logic 부분 중에서도 해당 FIFO 구현을 위해 필요한 Logic 부분이 포함되어 있다. <표 3>은 FIFO 의 depth 가 4 인 경우와 2 인 경우에 32 비트 또는 64 비트 FIFO 의 면적을 LUT 수로 보여주고 있고, LUT 개수 중에서도 Logic 부분과 RAM 부분을 구분해 놓았다. 표 2 의 실험을 위해서는 모두 FIFO 의 depth 를 모두 2 로 설정해 두었다. 그리고, 각 네트워크 인터페이스 모듈은 최소한 3 개의 32 비트 FIFO 3 개 (주소, 읽기데이터, 쓰기 데이터)를 포함한다. 그러므로, 최소한 FIFO 에 의해서 사용되는 면적은 255 (= 85 x 3) 임을 알 수 있다. 따라서 자동 생성된 네트워크 인터페이스 모듈은 수작업에 의한 설계와 면적이나 동작 속도에서 큰 차이를 보이지 않으며, 전체 면적 중에서 FIFO 에 의해 사용되는 면적이 전체면적의 60~70%를 차지함을 알 수 있다.

<표 2> 자동 설계된 네트워크 인터페이스와 수작업에 의한 설계의 비교

	수작업 설계				자동 설계			
네트워크 인터페이스	#LUTs	Logic	RAM	MHz	#LUTs	Logic	RAM	MHz
AHB 마스터	359	145	214	114.3	340	126	214	111.9
AHB 슬레이브	330	114	216	146.2	306	92	214	130.8
OCP 마스터	315	115	200	136.5	314	114	200	137.4
OCP 슬레이브	291	91	200	139.9	303	103	200	127.0
PVCI 마스터	314	110	204	129.6	314	110	204	129.6
PVCI 슬레이브	319	115	204	131.3	315	107	208	134.4
WB 마스터	322	114	208	138.0	329	123	206	133.5
WB 슬레이브	284	76	208	136.1	312	104	208	133.9

<표 3> 인자에 따른 FIFO 의 면적

	4x2 FIFO 32/64 비트	2x1 FIFO 32/64 비트
# 4LUTs	94/158	85/149
Logic	30/30	21/21
RAM	64/128	64/128

4. 결론 및 향후 연구

기존의 NoC 에서는 네트워크 인터페이스가 자체적인 표준 인터페이스를 사용함으로써 다른 버스 인터페이스를 갖는 IP 들을 통합할 때 NoC 에 직접 붙일 수 없고 별도의 래퍼를 사용해야 하는 오버헤드가 발생한다. 이러한 오버헤드를 없애기 위해서, 본 논문에서는 다른 버스 프로토콜의 IP 를 직접 NoC 에 부착할 수 있도록 해당 버스 IP 와 호환되는 네트워크 인터페이스를 자동 생성하는 방법을 제안하였다. 이를 위하여, NoC 시스템을 위해 표준적인 패킷 포맷을 정의하였으며, 거기에는 패킷에 대한 라우팅 정보 뿐 아니라, 여러 종류의 버스 IP 의 데이터, 주소, 제어 정보, 응답 정보 등도 포함될 수 있도록 정의하였다. 네트워크인터페이스는 표준패킷포맷과 특정 버스 인터페이스간의 변환 작

업을 수행한다. 실험을 통해서, 자동 생성된, PVCI, OCP, Wishbone, AHB 버스 IP 를 위한 네트워크인터페이스 간에 표준 패킷 포맷을 이용한 데이터 통신이 중요한 정보의 손실이 없이 잘 이루어질 수 있음을 보였다.

참고문헌

[1] L. Benini and G. De Micheli. “Networks on chips: A new SoC paradigm”, IEEE computer, 35(1), 2002
[2] ARM, AMBA Specification, http://www.arm.com
[3] VSI Alliance, http://www.vsi.org
[4] O.C.P.I.P.A. Inc. Http://www.ocpip.org
[5] “WISHBONE System-on-Chip(SoC) Interconnection Architecture for Portable IP Cores,” http://opencores.org
[6] Se-Joong Lee, et al., “Packet-Switched On-Chip Interconnection Network for System-On-Chip Applications”, IEEE Transaction on Circuits and Systems, II: EXPRESS BRIEFS, VOL. 52, NO. 6, JUNE 2005, pp.308-312.
[7] AMBA AXI v1.0 specification, http://www.arm.com
[8] Passerone R., et al., “Automatic synthesis of interfaces between incompatible protocols,” Proceedings of Design Automation Conference, 15-19 Jun 1998 Page(s):8 - 13
[9] D. Shin, et al., “Interface Synthesis from Protocol Specification,” Technical Report (CECS-02-13), April, 12 2002, Center for Embedded Computer Systems University of California, Irvine.
[10] C. Yun, et al., "Automatic Interface Synthesis based on the Classification of Interface Protocols of IPs", IEEE/ACM Asia and South Pacific Deisgn Automation Conference 2008 Proceeding, pp.589-594, Jan. 2008.