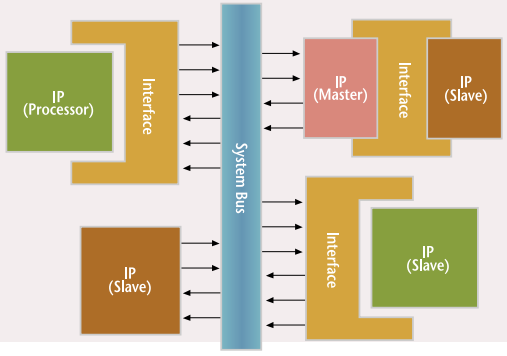
윤창열 충남대 컴퓨터공학과 박사과정 E-Mail : yun@ce.cnu.ac.kr
장경선 | 교수 |
충남대 전기정보통신공학부 E-Mail : ksjhang@ce.cnu.ac.kr

인터페이스 합성 자동화 기술과 동향

서론

VLSI 집적 기술의 빠른 발전에 따라 성능, 전력소비, 소형화 등의 장점을 갖는 SoC 설계와 구현이 가능하게 되었다. 수백만 게이트의 SoC를 처음부터 설계하는 것은 비실용적이고 개발 시간도 많이 필요하기 때문에, IP를 재사용 하는 것이 일반화 되고 있다. 이는 새로운 상품을 빠르게 만들고, TTM(Time to Market)을 지키는 데에도 도움이 된다. IP를 SoC 설계에 사용하기 위해서는 IP간의 데이터 전송을 위한 인터페이스 설계와 그에 대한 검증 작업이 필요하다. 이는 SoC의 설계에서 아주 빈번한 작업이다. [그림 1]은 IP를 시스템에 연결하기 위한 인터페이스 회로의 역할이다. 인터페이스는 SoC의 시스템 버스와 호환되지 않는 프로토콜을 갖는 IP를 시스템버스와 연결하기 위해 필요하고, 서로 다른 프로토콜을 갖는 IP 사이의 통신에서도 인터페이스가 필요하다. 인터페이스 설계를 위해서는 설계자가 IP의 자세한 신호와 프로토콜에 대해 이해해야 한다. 인터페이스 설계는 오류가 발생하기 쉽고, 지루한 작업이 될 수 있다. 따라서 인터페이스 설계의 자동화가 필요하다. 이런 인터페이스 설계의 자동화는 설계자의 노력을 줄이고, 시스템의 개발 시간도 단축시킬 수 있을 것이다. 본 고에서는 IP 재사용 기반의 SoC 설계에 필요한 인터페이스 회로를 자동적으로 생성하는 방법에 대해 살펴보고자 한다.



[그림 1] IP 연결을 위한 인터페이스의 역할

인터페이스 설계를 자동화하기 위해서는, 해당 IP의 인터페이스 프로토콜에 대한 정보가 필요하다. 설계자는 IP의 프로토콜에 대한 이해뿐 아니라, 인터페이스의 자동생성을 위해 IP의 인터페이스 프로토콜을 형식적인 형태로 표현해야 한다. 또한 각 IP의 입출력 신호의 이름이 다를 수 있기 때문에, IP간의 제어신호/데이터 연결 정보도 필요하다. 따라서 인터페이스 자동생성의 입력물은 IP의 인터페이스 프로토콜에 대한 정보와 IP간의 데이터/제어신호 간의 연결정보이고, 출력물은 RTL 형태의 코드로 합성 가능한 인터페이스 회로이다.

인터페이스 합성

(1) 고려사항

인터페이스는 유효한 데이터의 전송과 시스템의 안정된 동작을 위해 다음과 같은 점을 만족해야 한다.

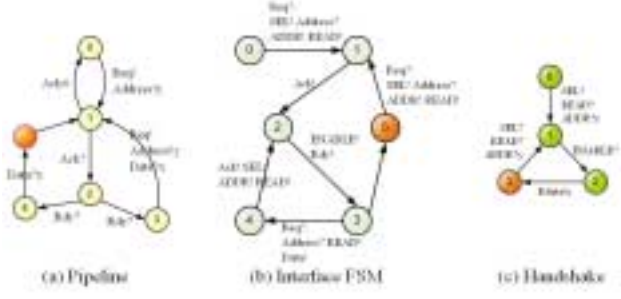
· 각 데이터 전송의 타이밍 요구조건 충족 · 일관된 데이터 전송 보장
· 데이터 전송 타입 보장 · 전송 타입 보장 · 통신 프로토콜 준수 (오류 상태 등)

인터페이스 회로는 각 IP에서 요구하는 타이밍 요건을 충족해야한다. 예를 들어 메모리 데이터 입출력 시에 일정 시간동안 데이터를 유지한다거나, 150 ns 이내에 데이터 전송을 마쳐야 하는 등의 요구조건이 있다. 인터페이스의 동작도 이런 조건을 반드시 충족시켜야 데이터 전송에 문제가 없을 것이다. 또한 데이터 전송의 일관성을 유지하는 것이 필요하다. 이는 인터페이스회로를 통해 전달된 데이터가 원본 데이터와의 일치해야 함을 의미한다. 데이터 전송 타입은 각 IP의 전송 동작이 유지되어야 함을 의미한다. 인터페이스회로의 중요한 역할 중 하나는 연결되는 IP의 통신 프로토콜을 준수해야 하는 것이다.

(2) 인터페이스 프로토콜의 표현

인터페이스 생성을 자동화하기 위해서, 설계자는 IP의 인터페이스 프로토콜을 이해하고, 인터페이스 프로토콜을 형식을 갖는 형태로 기술해야 한다. 인터페이스 프로토콜을 RTL 코드에서 직접 추출하려 하였으나, RTL 코드만으로는 정확한 인터페이스 프로토콜을 얻을 수 없었다. 따라서 설계자가 직접 인터페이스 프로토콜을 기술하는 것이 필요하다. IP의 프로토콜은 Soft IP의 경우 RTL 코드에서 프로토콜을 파악하게 되고, Hard IP나 Firm IP의 경우에는 타이밍다이어그램에 의존하게 된다. 이런 정보들을 근거로 기술하는 IP의 인터페이스 프로토콜의 기술 방식은 크게 둘로 나눌 수 있다. IP의 동작을 FSM(Finite State Machine) 형태로 표현하는 방법[1,2,3,8]과 각 신호들의 순서관계를 고려하여 STG(Signal Transition Graph)와 유사한 형태로 인터페이스 프로토콜을 표현하는 방법[4,5,7,9]이 있다. FSM 형태의 기술은 IP의 인터페이스 프로토콜의 다양성(Alternative)을 표현할 수 있는 장점을 갖으나, 시간제약조건 등을 표현하기가 어렵다. STG를 이용한 방식은 주로 타이밍다이어그램을 근거로 생성하게 되는데, 신호의 순서관계나 시간제약 조건을 표시하기에 적합하다.

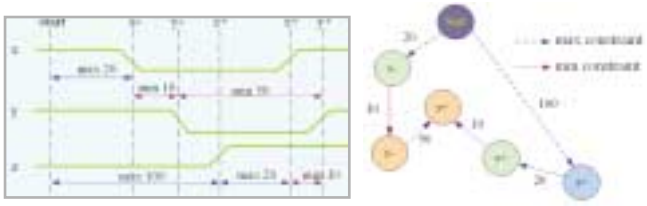
[그림 2]는 간단한 인터페이스를 FSM 형태로 기술한 예이다. [그림 2(a)]는 파이프라인을 허용하는 인터페이스의 표현이고, [그림 2(c)]는 핸드쉐이크 프로토콜을 기술한 것이다. [그림 2(b)]는 두 프로토콜 사이의 인터페이스 FSM이다. 각 상태의 전이는 클럭과 동기화 된다. 각 전이에는 IP의 입출력 포트에 나타나는 신호의 동작 표시되어 있다. 신호의 이름 뒤에 "?"로 표시된 것은 입력(조건)임을 의미하고, "!"로 표시된 신호는 출력(구동)신호 임을 의미한다. "#"는 해당 신호에 이벤트가 발생하지 말아야 함을 의미한다.



[그림 2] 인터페이스 FSM[2]

[그림 3]은 타이밍 다이어그램에 근거하여, 신호에서 나타나는 이벤트의 순서관계를 표시하는 예이다. IP의 전달물에 타이밍다이어그램이 포함되어 있으므로, [그림 3(a)]에서 [그림 3(b)]를 추출할 수 있다. [그림 3(b)]의 상태의 "x−"는 신호 x의 값이 high에서 low로의 전이가 발생함을 의미한다. STG와 비슷한 방법으로

신호의 순서관계를 Petri net으로 표현하려는 시도도 있다[5].



(a) 타이밍 타이어 그램 (b) Signal Transition Graph

[그림 3] 인터페이스 STG의 예[4]

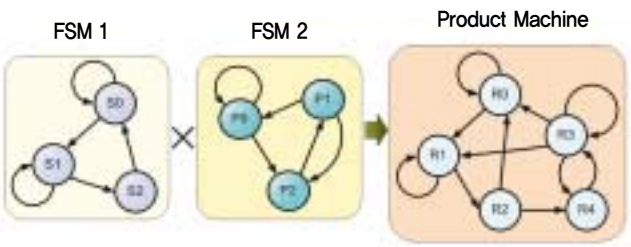
앞서 제시된 기술 방식이외에도 많은 방법이 시도되었다. 연구 [7]에서는 입출력 포트에서 발생하는 이벤트를 순서대로 표시하였다. 노드는 이벤트, 에지는 변화의 순서를 나타내었다. 연구 [8]은 VHDL 기술의 일부를 이용하여 인터페이스를 기술하려하였다. Wait문, 신호/변수 할당문을 사용하여 인터페이스 프로토콜을 표시 하였으나, 여러 분기를 갖는 프로토콜을 표현하기 어려웠다. 연구[9]는 PFG(Protocol Flow Graph)를 이용하여 사이클의 경과와 이벤트를 동시에 기술 가능하였으나, 트랜잭션 수준의 정보를 표현하기 어려웠다. 이렇듯 IP의 인터페이스를 표현하기 위한 많은 연구가 있었지만, 아직 IP의 인터페이스를 표현하기 위한 표준은 없다. 인터페이스 프로토콜의 표현 방식은 주로 해당 IP의 특성에 따라 선택하게 된다. 클럭에 동기화 되어 동작하는 IP는 FSM 형태의 기술이 인터페이스 프로토콜을 표현하는데 용이하고, 비동기적 특성을 갖는 IP의 인터페이스 프로토콜의 기술은 클럭을 사용하지 않기 때문에, 각 신호의 순서관계를 표시하는 STG 방식의 표현이 적합하다.

(3) 인터페이스 합성 방법

인터페이스 프로토콜에서 인터페이스 회로를 자동 생성하는 방법에 대해 알아보자. 합성 알고리즘도 앞서 기술한 표현방식에 많은 영향을 받아, 크게 FSM 기반 인터페이스 합성 방법과, STG 기반 인터페이스 합성 방법으로 구분할 수 있다.

가. FSM 기반 인터페이스 합성

FSM 기반 인터페이스 합성은 연결 IP의 인터페이스 프로토콜을 기술한 FSM과 IP 연결 정보를 입력 받는다. 해당 FSM의 통합은 [그림 4]와 같이 Product Machine을 생성하고, 불필요한 상태를 삭제한다. 또한 IP의 연결 정보를 이용하여 최적의 FSM을 생성한다. FSM 통합 방식으로 인터페이스를 사용하는 방법을 살펴보자.



[그림 4] FSM 통합

A. PIG(Protocol Interface Generation)[1]

PIG는 IP의 인터페이스 프로토콜을 정규수식의 형태로 표현하였다. 각 사이클마다 IP의 포트에 나타나는 신호의 값을 노드로 정하고, 인터페이스 프로토콜은 이러한 노드의 순서관계로 표현하였다. 이러한 정규수식에서 FSM을 생성하고, [그림 4]의 방법을 이용하여 인터페이스 FSM을 자동생성 하였다. 이 연구에서는 인터페이스의 합성뿐만 아니라, IP의 테스트에도 응용 가능한 테스트 벤치의 생성이나, IP의 동작을 모니터링 할 수 있는 모니터 모듈의 생성에도 응용하였다. IP의 포트 값을 모니터링 하여, IP의 동작이나 전달 데이터에 대한 정보를 얻을 수 있다. 또한 IP를 테스트하기 위한 "Driver" 모듈도 생성할 수 있다. 그러나 단일 트랜잭션의 기술만을 허용하였고, 상태가 복잡하면 많은 상태가 생성되는 문제가 발생하고 데이터를 연속적으로 전달하는데 문제가 발생하였다. 연구[2]에서는 각 데이터 신호마다 버퍼를 삽입하여 이런 단점을 극복하였다.

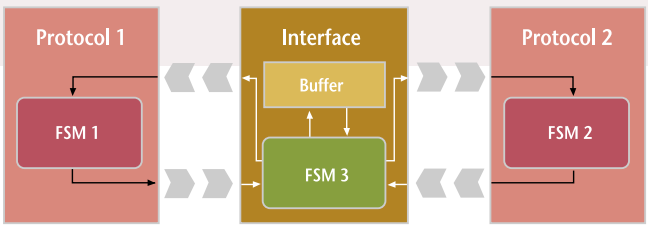
B. Synchronous Protocol Automata[2]

Synchronous Protocol Automata 방법은 IP의 동작을 사이클 수준으로 기술한다. 기술 방법은 [그림 2]와 같은 방법을 사용한다. 연구 [2]의 특징은 프로토콜의 호환성을 검사할 수 있다는 점이다. 연결하려는 IP들의 프로토콜을 입력 받아서, IP 간의 프로토콜의 호환성을 검사한다. 즉 호환 가능한 IP는 인터페이스의 추가 없이 직접 연결 가능한 것이다. 이 기능을 버스 시스템에 적용하면, 해당 IP가 버스 프로토콜에 호환 가능한지를 파악하는 기능으로 사용할 수 있다. [2]에서 제안한 인터페이스의 구조는 [그림 5]와 같다. 한 가지 주목할 점은 제어 신호를 제외한 데이터 신호마다 버퍼를 생성한다는 점이다. 각 프로토콜에서 구동하는 데이터를 버퍼에 저장하여, 필요시에 데이터를 구동하는 것이다. 또한 버퍼에 저장된 데이터의 관리를 위해 각각 카운터가 있다. 버퍼의 데이터는 데이터가 사용되기 전까지 값을 유지한다. 다음은 인터페이스 FSM을 생성하는 간략한 알고리즘의 설명이다.

① 입력 FSM에 근거하여 초기상태 생성
② 각 전이에 의한 새로운 상태 생성
③ 생성된 상태로의 전이에 필요한 각 동작의 Complement(action)를 찾음
 – S = Complement(action) : S : 해당 action의 반대 동작
④ 생성된 상태가 유효한 상태인지 검사(기준 – 전달 데이터의 존재 유무)
⑤ 유효한 상태이면 상태 목록에 추가한다.
⑥ 처리되지 않은 상태가 있다면② 번으로 돌아감
⑦ 생성된 상태에서 중복된 상태 제거

위의 과정을 거치면, [그림 2(b)]의 인터페이스 FSM이 생성된다. 이런 FSM을 포함하여 [그림 5]의 인터페이스가 생성된다. 연구 [2]에서는 AMBA AHB[6]의 각 모듈에 대한 인터페이스 프로토콜을 기술하여, 다른 시스템 버스와의 연결에 해당 인터페이스를 적용하였다.

이 인터페이스 생성 방법의 단점은 비동기적인 동작에 대처할 수 없다는 점이다.

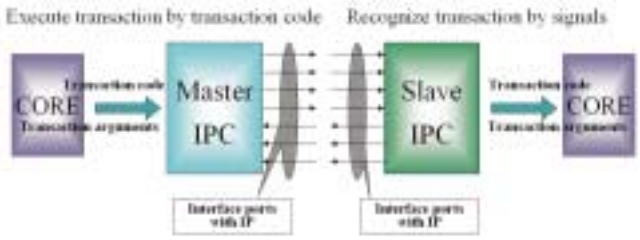


[그림 5] 인터페이스 구조[2]

C. IPC(Interface Protocol Component)[3]

이 연구의 가장 큰 특징은 IP의 인터페이스 프로토콜을 트랜잭션 수준으로 추상화했다는 점이다. IP의 동작 기술에 트랜잭션 수준의 정보를 추가하여, 설계자가 IP의 프로토콜에 대한 자세한 이해 없이도, 트랜잭션 수준의 연결만으로 인터페이스를 만들 수 있는 장점이 있다. 즉 IPC가 IP의 동작을 트랜잭션 수준으로 추상화 한다. 이런 IPC는 마스터 IPC, 슬레이브 IPC의 두 종류가 있다. 마스터 IPC는 IP를 구동하여 트랜잭션을 시작시킬 수 있는 모듈이고, 슬레이브 IPC는 IP의 동작을 모니터링 하여, 전달 데이터나 트랜잭션을 인식할 수 있는 모듈이다. [그림 6]은 IPC의 동작 특성을 나타낸 것이다. [그림 6]에서 Core 모듈이 IPC에 트랜잭션 정보를 전달한다. 트랜잭션 코드는 마스터 IPC를 구동하기 위한 트랜잭션 정보이고, 트랜잭션 전달인자는 트랜잭션 수행에 필요한 데이터를 의미한다.

이런 IPC의 연결로 올바른 데이터 전송이 가능한 인터페이스를 생성 할 수 있다. 그러나 IPC를 이용한 인터페이스 합성 방법은, IPC가 오버헤드로 작용할 여지가 있다.



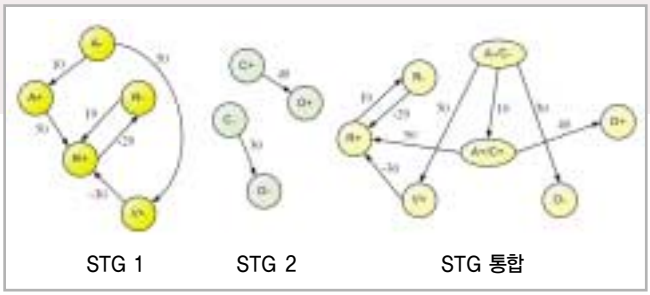
[그림 6] 마스터/슬레이브 IPC[3]

나. STG 기반 인터페이스 합성

FSM 기반의 인터페이스 합성 방법에 이어, STG 기반의 인터페이스 합성 방법에 대해 알아보자.

A. 조합회로 인터페이스 합성[4]

연구 [4]에서는 프로세서와 메모리의 연결에 필요한 조합회로의 특성을 갖는 인터페이스를 자동생성하기 위한 방법을 제시한다. 인터페이스 프로토콜의 변환에 전력소모를 최소화하기 위한 개념을 도입한 것이 주목할만하다. 즉 각 신호의 switching을 줄이는 인터페이스 회로가 생성되도록 한다. 이렇게 하기 위해서 연구 [4]에서는 입출력 신호를 세 가지 범주로 분류한다. 첫째는 VDD나 GND에 직접 연결 가능한 신호, 둘째 IP간 직접 연결이 가능한 신호, 셋째 IP간 직접 연결은 불가하지만, 사이에 조합회로를 추가하여 연결 가능한 신호이다. 이렇게 신호를 세 그룹으로 분류하고 각 그룹에 따라 다른 해결책을 제시한다. 첫 번째와 두 번째 신호의 그룹은 STG(그림 3 참조)의 통합 후, 통합된 STG에 근거하여 각 신호의 연관성 및 신호의 전이를 최소화하는 최적의 해를 찾는다. [그림 7]은 STG의 통합 과정을 표현한 그림이다.

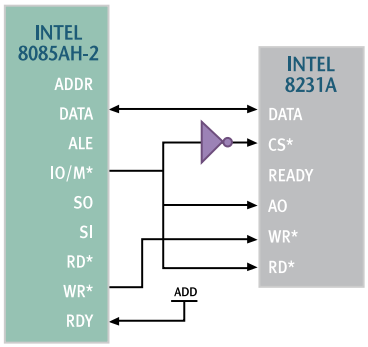


[그림 7] STG 통합(Merge(STG1,{A+,A−},STG2,{C+,C−})[4]

세 번째 신호의 그룹은 타이밍 다이어그램을 이용한다. 타이밍 다이어그램을 아주 작은 타이밍 슬롯으로 나누어, 각 슬롯 별로 신호의 연관성을 찾아낸다. 예로 신호 CS*는 모든 슬롯에서 not(IO/M*)의 특성을 갖는다면, [그림 8]과 같이 IO/M 신호와 CS* 신호에 인버터를 연결하게 된다. 위 과정을 거쳐 [그림 8]의 인터페이스를 생성하였다.

B. 비동기회로 인터페이스 합성[5]

비동기 인터페이스 합성에 대한 연구 [5]는 비동기로 동작하는 IP의 인터페이스를 PN(Petri Net) 형태로 기술하고, 그로부터 인터페이스 FSM을 생성하는 방법을 제시하였다. 타이밍다이어그램에서 STG를 추출하며, STG는 PN을 이용하여 표현한다. 이렇게 생성된 PN을 기반으로 상태 그래프를 생성하고, 새로운 상태를 할당한다. 이 방법은 비동기회로의 합성에 많이 사용되는 방법이다. 이렇게 생성된 상태 그래프에서 FSM을 만들게 된다. 이 FSM을 근거로 인터페이스 회로를 생성한다.



[그림 8] 프로세서와 메모리 간의 인터페이스 회로[4]

결론

지금까지 인터페이스 합성의 필요성과 인터페이스 표현 방법, 여러 가지 인터페이스 합성 방법에 대해 알아보았다. 인터페이스 자동 합성 방법이 효과적으로 활용된다면 설계자의 노력을 줄이고, 빠른 시간에 SoC의 설계와 검증을 마칠 수 있을 것으로 기대된다. 지금까지 많은 연구가 있었음에도 불구하고, 인터페이스 회로의 자동생성 방법은 보편화되지 못하고 있다. 각 IP의 인터페이스의 특징이 너무 다양하기 때문에 모든 IP의 특성과 요구사항을 모두 만족시키는 보편적인 인터페이스 합성 규칙을 찾기 어렵기 때문일 것이다. 또한 인터페이스의 자동 생성과정에 설계자의 의도나 개입이 필요한 경우가 많다. 인터페이스 합성을 위해서는 누군가가 IP의 인터페이스 프로토콜을 기술해 주어야 하고, 연결하려는 IP간의 데이터 전송을 위해, “Transaction”, “byte enable”, “error condition”등의 연결 정보간의 Matching도 제공해야 한다. 어떤 경우에는, IP의 모든 기능을 제공하지 못하고 일부만 사용할 수 있는 경우도 발생할 것이다.

자동 생성된 인터페이스의 결과가 사람이 직접 작성한 인터페이스보다 효율이 떨어질 수 있다. 인터페이스 합성 방법은 최대한 빠른 검증을 위해 인터페이스 회로가 필요한 경우에 활용 가능할 것이다.

현재 인터페이스 연구는 기존의 합성 방법에 전력 소모를 적게 하기 위한 방법을 찾는 연구[10]가 이루어지고 있다. 또한 시스템 기반의 다른 시스템 버스와의 연결을 위한 라이브러리 제공 방식 등에 의한 인터페이스 합성 방법도 많이 연구[11]되고 있다.

[참고문헌]

[1] R. Passerone, “Automatic Synthesis of Interfaces between Incompatible Protocols,” M.S. Thesis, University of California at Berkeley, 1997.
[2] D' silva, V.; Ramesh, S.; Sowmya, A.; “Synchronous Protocol Automata: A Framework for Modeling and Verification of SOC Communication Architectures,” Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings ,Volume: 1 , 16-20 Feb. 2004 Pages:390 - 395
[3] ChangRyul Yun, KyoungSon Jhang, “An Interface Protocol Component Modeling Language”, 15th IEEE ASIC/SOC International Conference, 2002 Sep, NY, USA
[4] Ki-Seoc Chung, Rajesh K. Gupta Taewhan Kim, C.L. Liu, “Synthesis and Optimization of Combinational Interface Circuits,” Journal of VLSI Signal Processing 31, 243~261, 2002 Kluwer Academic Publishers. Manufactured in The Netherlands.
[5] Kishinevsky, M.; Cortadella, J.; Kondratyev, A.;”Asynchronous interface specification, analysis and synthesis,” Design Automation Conference, 1998. Proceedings , 15-19 June 1998 Pages:2 - 7
[6] ARM Ltd,”AMBA Specification Rev 2.0”, Document Number ARM IHI 0011A, 1999
[7] Gaetano Borriello, and Randy H. Katz, “Synthesis and Optimization of Interface Transducer Logic,” Proc. ICCAD '87, pp.274-277. (Event Graph)
[8] Jan Madsen and Bjarne Hald, “An Approach to Interface Synthesis,” in Proc. of ISSS, 1995.
[9] Sanjiv Narayan and D. D. Gajski, “Interfacing incompatible protocols using interface process generation,” in Proc. of DAC, 1995, pp.468-473.
[10] Liveris, N.D.; Banerjee, P.;“Power aware interface synthesis for bus-based SoC designs,” Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings , Volume: 2 , 16-20 Feb. 2004 Pages:864 - 869 Vol.2
[11] “ Platform Express component Integrator' s Guide version 2.1g” , June 2004 http://www.mentor.com/