Design Methodology

장준영 선임연구원 / **한진호** 연구원
배영환 책임연구원 / **조한진** 팀장

한국전자통신연구원 시스템IC설계팀
E-mail : jychang@etri.re.kr

SoC Platform 기반 Design Methodology

실리콘 처리 기술의 고속화 요구와 유무선 환경에서 동영상 통신이 가능한 비디오 폰, 영상 회의 시스템, 이동 통신용 단말기 등의 전자 제품 사용자의 급증은 시스템을 하나의 칩에 집적화하는 SoC(System-On-a-Chip) 설계 기술을 요구하고 있다. 칩의 복잡도와 SoC 제품의 생산성 차이가 계속적으로 증가함에 따라 현재의 IC 설계 방법으로는 SoC 제품의 성능과 요구의 변화를 만족시킬 수 없다. 칩의 면적을 최소화하고 성능을 최대화하며 게이트 수준의 최적화를 통한 기존의 셀 기반 설계 방법으로는 설계의 생산성 문제를 해결할 수 없다. 이러한 문제를 해결 위한 새로운 설계 방법인 IP 재사용을 기반으로 한 플랫폼 기반 설계가 제시되었다. 플랫폼 기반 설계는 SoC 제품을 빠르게 개발하기 위한 응용 기반 통합 플랫폼과 재사용이 가능한 IP(Intellectual Property) 이용한 플랫폼 기반 설계(Platform-Based Design) 방법이다. 새로운 설계 방법은 90% 이상의 IP 재사용을 통해서 설계 시간을 단축하며, 시스템 수준에서의 최적화를 통해서 제품의 시장 경쟁력(Time-to-Market)의 문제를 해결하기 위한 방법이다.

1. 서론

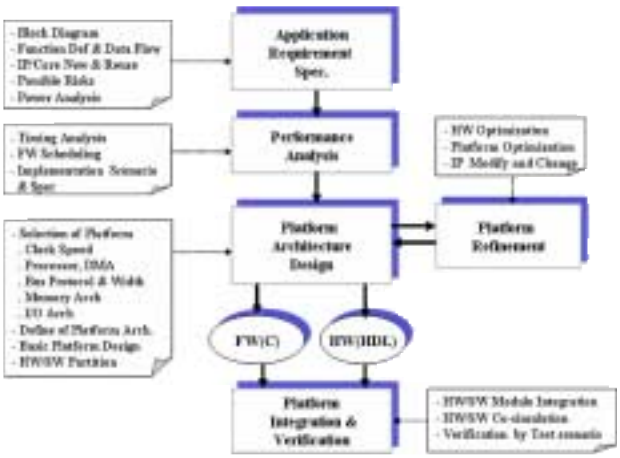
SoC Platform 기반 설계 요구

실리콘 처리 기술의 고속화 요구와 유무선 환경에서 동영상 통신이 가능한 비디오 폰, 영상 회의 시스템, 이동 통신용 단말기 등의 전자 제품 사용자의 급증은 시스템을 하나의 칩에 집적화하는 SoC(System-On-a-Chip) 설계 기술을 요구하고 있다. 칩의 복잡도와 SoC 제품의 생산성 차이가 계속적으로 증가함에 따라 현재의 IC 설계 방법으로는 SoC 제품의 성능과 요구의 변화를 만족시킬 수 없다. 칩의 면적을 최소화하고 성능을 최대화하며 게이트 수준의 최적화를 통한 기존의 셀 기반 설계 방법으로는 설계의 생산성 문제를 해결할 수 없다. 이러한 문제를 해결 위한 새로운 설계 방법인 IP 재사용을 기반으로 한 플랫폼 기반 설계가 제시되었다. 플랫폼 기반 설계는 SoC 제품을 빠르게 개발하기 위한 응용 기반 통합 플랫폼과 재사용이 가능한 IP(Intellectual Property) 이용한 플랫폼 기반 설계(Platform-Based Design) 방법이다. 새로운 설계 방법은 90% 이상의 IP 재사용을 통해서 설계 시간을 단축하며, 시스템 수준에서의 최적화를 통해서 제품의 시장 경쟁력(Time-to-Market)의 문제를 해결하기 위한 방법이다.

플랫폼 기반 설계는 기술-기반 플랫폼(Technology-Driven Platform)과 응용-기반 플랫폼(Application-Driven Platform)으로 구분된다. 기술-기반 플랫폼은 모듈 설계를 한 후에 성능 분석을 통해서 최적화 및 검증을 실행 한 후에 IP 라이브러리를 사용하여 칩을 설계하는 상향식 설계(Bottom-up Design) 방법이다. 응용-기반 플랫폼은 응용 분야의 요구 사항에 따라서 다양한 계열로 나누고, 이를 분석하여 적합한 플랫폼의 구조를 결정한 다음, 실제 IP 라이브러리를 적용하여 칩을 설계하는 시스템 수준 설계의 하향식 설계(Top-Down Design) 방법이다. 응용-기반 플랫폼은 상위 수준에서 응용 분야 따라서 다양한 성능 분석이 가능하므로 개발 실패 부담을 최소화할 수 있다. 미리 정의된 블록이나 모델들을 재사용하므로 개발 시간을 단축할 수 있는 장점이 있다. 플랫폼을 이용한 설계 방법은 높은 초기 플랫폼 설계 비용과 미리 정의된 플랫폼으로 인해서 설계의 유연성이 제한되는 어려움은 있으나 개발된 플랫폼을 이용하여 다양한 응용 분야를 개발하는데 시간과 비용의 감소를 통해서 보완될 수 있다.

SoC Platform 기반 설계 방법

플랫폼 기반 설계는 플랫폼을 설계하는 과정과 플랫폼을 사용하여 응용 분야 제품을 적용하는 과정으로 구분된다. 플랫폼을 설계하는 과정에서는 응용 분야에 따른 IP나 하드웨어 및 소프트웨어 모듈을 특성화하여 선택하며, 시스템 구조를 결정하고 성능과 전력 소모를 최적화하기 위한 성능 분석 과정을 통해서 플랫폼을 결정하고 선택한다. 플랫폼을 사용하는 과정에서는 플랫폼의 응용 분야에 필요한 IP 및 하드웨어 및 소프트웨어 모듈을 최적화하여 통합하고 검증을 실행한다.



[그림 1] SoC Platform 기반 설계 흐름도

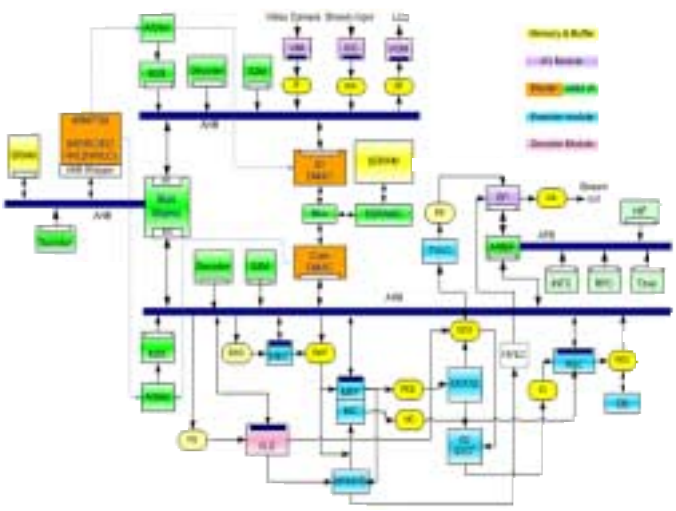
[그림 1] 은 SoC 설계를 위한 플랫폼 기반 설계 방법의 흐름도이다. 플랫폼을 구성하는 과정에서는 먼저 적용하고자 하는 응용 분야의 영역과 제품의 발전 방향을 조사하고 정의한다. 초기 플랫폼을 구성하기 위해서 초기 플랫폼 구조를 구성하는 프로세서와 DSP를 선정하고 데이터 전송 및 처리를 위한 온칩 버스와 내부 및 외부 메모리를 선택한다. 응용 분야에 적합한 하드웨어 및 소프트웨어 IP를 결정하고 IP에 필요한 Wrapper를 설계한다. 성능 분석 및 플랫폼 선택 및 구성 과정에서 메모리 Bandwidth와 전력 소모 및 성능들을 분석 및 평가하여 응용 분야의 요구 사항을 만족하는 플랫폼을 선택하고 구성한다. 플랫폼 Refinement 단계에서는 응용 분야를 개발하기 위한 플랫폼을 조정 및 최적화하는 과정으로 버스 크기, 클럭 속도 데이터 전송 프로토콜에 따라서 각 모듈들을 최적화한다. 하드웨어와 소프트웨어 모듈을 분할하고 소프트웨어의 싸이클을 측정하여 태스크 스케쥴링을 생성한다. 플랫폼 구현과 검증 단계에서는 분할된 소프트웨어 모듈은 플랫폼의 프로세서에서 실행되며 하드웨어 모듈은 플랫폼에 연결된다. 소프트웨어와 하드웨어 사이의 혼합 검증을 통해서 구현된 플랫폼을 검증한다.

2. Platform Selection

플랫폼을 구성하는 프로세서 코어, DSP와, 버스 구조와 소프트웨어 구조를 통해서 응용 분야에 맞는 플랫폼을 선택한다. 최근 SoC 플랫폼 기반 설계에서 버스 구조를 이용한 통신 기반 플랫폼을 많이 사용하고 있다. 통신 기반 플랫폼으로 많이 사용되는 버스 구조는 ARM사의 AMBA, Sonic's SiliconBackplane과 온칩 버스의 버스 Bandwidth를 향상시키기 위한 온칩 네트워크 구조 등이 있다. 응용분야에 따라서 적합한 버스 구조를 선택하여 플랫폼을 구성할 수 있다.

멀티레이어 온칩 버스 구조를 갖는 AMBA 버스

ARM사의 AMBA 버스는 단일 버스 구조를 갖는 ASB/APB 버스와 버스트 데이터 전송 , 분할 데이터 전송이 가능한 멀티레이어 버스 구조를 갖는 AHB/APB 버스가 있다. 기존의 ASB 버스에서는 하나의 물리적인 버스를 하나의 마스터가 점유하고 있으면, 다른 마스터들은 통신을 할 수가 없었다. 이를 해결하기 위해 멀티레이어 AHB 버스는 멀티레이어로 구성된 여러 개의 물리적인 버스를 사용하는 방법이다. 이러한 물리적인 버스간의 통신은 버스 브리지를 이용하여 통신을 하게 하는데 이 버스 브리지는 동시에 서로 다른 버스를 충돌 없이 연결할 수 있는 Interconnection matrix 구조를 갖는다. [그림 2]는 멀티레이어 AHB 버스로 구성된 플랫폼을 사용하여 MPEG-4 영상을 코덱으로 초당 25~30프레임을 처리할 수 있는 시스템을 구성하였다.



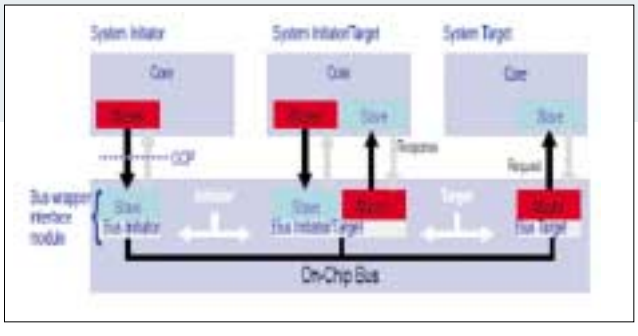
[그림 2] 멀티레이어 AHB 버스 구조를 이용한 MPEG-4 영상 코덱

ARM 코어와 프로그램 메모리가 연결된 AHB 시스템 버스와 Encoder와 Decoder 모듈로 구성된 AHB 코어 버스와 영상 입출력을 전담하는 AHB 버스와 주변장치 모듈을 연결하는 APB 모듈로 구성되어 있다. 1개의 마스터 입력 포트와 2개의 슬레이브 출력 포트로 구성된 Bus Matrix에 의해 3개의 AHB 버스가 연결되어 있다. 마스터 입력 포트에서 AHB 코어 버스로 데이터를 전송하는 동안 영상 입출력 모듈인 VIM과 ISC는 IODMAC를 통해서 입출력 AHB 버스를 이용하여 SDRAM과 데이터 전송을 할 수 있다. 버스 레이어 수에 따라서 동시에 처리할 수 있는 통신량이 증가하게 된다. Bus Matrix를 이용하여 최대 마스터 입력 포트 8개와 슬레이브 출력 포트 8개 갖는 멀티레이어 구조를 구성할 수 있다. 이러한 구조는 각각의 독립된 버스 레이어마다 중재기와 디코더를 각각 갖고 있어야 하기 때문에 많은 마스터가 동작하는 시스템에서는 버스 레이어를 추가할 때마다 비용이 증가하고 버스 레이어를 변경하는 데이터를 전송하는 경우에 한 싸이클이 소모되므로 데이터 처리량이 감소하는 문제점이 있다.

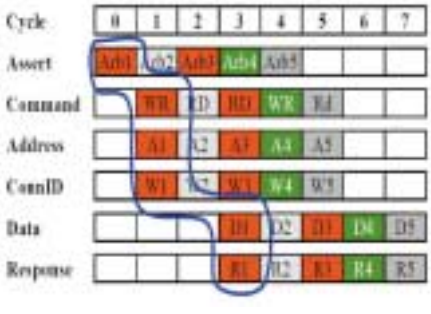
이를 해결하기 위한 방안으로 ARM 사에서 최근 AMBA AXI (Advanced eXtensible Interface) 프로토콜을 발표하였다. AXI 인터페이스는 다양한 버스를 연결하기 위해 버스 브리지를 사용하지 않고 구성 요소들을 쉽게 연결 확장할 수 있는 채널 기반의 상호 연결 구조를 가진 확장된 인터페이스 구조를 갖는다.

시분할 온칩 버스 구조를 갖는 SiliconBackplane

Sonic사의 SiliconBackplane은 동일한 버스를 사용하면서 버스를 시분할하여 점유하도록 하는 방식도 있다. 버스에 연결된 마스터들에게 사용할 수 있는 시간 영역을 부여를 하고, 버스 중재기는 정해진 시간 동안만 마스터에게 버스 사용권을 부여하게 된다. 이렇게 되면 온칩 버스에 연결된 마스터들은 하나의 마스터가 많은 시간을 점유를 해서 많은 시간동안 버스를 기다릴 필요가 없어진다. 자신의 차례가 되어 필요한 통신이 있으면 버스를 사용하게 된다. 마스터들은 Bandwidth가 낮아진 버스를 마스터 자신이 계속해서 사용하고 있는 것으로 보이게 된다. Bandwidth가 낮아지는 것을 해결하기 위해 버스의 동작을 연결된 마스터의 개수만큼 Pipelining시킨다. 그러면 버스에 연결된 마스터들은 시분할 하지 않은 버스와의 동일한 Bandwidth를 갖는 버스를 자신만이 점유해서 사용하는 효과를 낼 수 있다. [그림 3]은 시분할 온칩 버스 중의 하나인 Sonics사의 SiliconBackplane의 동작을 보여주고 있다.



[그림 3] 시분할 온칩버스인 SiliconBackplane

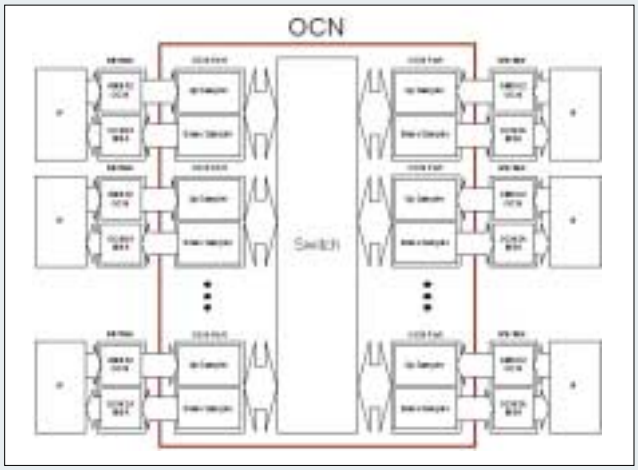


[그림 4] Pipeline Diagram

코어는 마스터나 슬레이브가 될 수 있다. 코어와 Siliconbackplane의 Bus wrapper interface module은 OCP(Open Core Protocol)을 따르고, Bus Initiator와 Bus Target을 Configurable Agent라고 하고 이를 통해서 온칩 버스와 연결시키고 있다. 분산된 제어 구조를 갖고 있어 각각의 코어마다 연결되어 있는 Configurable Agent가 모든 동작을 관장한다. [그림 4]의 Pipeline Diagram을 보면 자신의 차례가 되면 Configurable Agent는 Core에 Assert를 주게 되고, 다음 싸이클에 코어로부터 제어 경로를 통해 명령, 주소, 목적지 주소(ConnID)를 내보내고, 데이터 경로를 통해 다시 자신의 차례에 데이터(Dx)를 보내고, 그에 대한 응답 신호(Rx)를 받게 된다. 파란색 영역안의 빨간색이 동일한 마스터에서의 요청을 처리하는데 2 싸이클의 Latency를 갖고 있다.

온칩 네트워크

온칩 버스를 여러 개의 레이어로 나누어서 다른 버스 레이어 내부에서는 동시에 사용하도록 하는 멀티레이어 온칩 버스나, 버스를 시분할을 하여 여러 마스터가 동시에 버스를 사용하도록 보이게 하는 시분할 온칩 버스가 있다.
궁극적인 통신 구조의 사용도를 높이고, 버스를 사용하려는 마스터의 기다림을 없애기 위한 구조로 온칩 네크워크가 있다. 이는 버스에서 물리적으로 Wire를 공유해서 사용하기 때문에 생기는 IP들간의 데이터를 전송시 Bandwidth 제약을 없애준다. 온칩 네트워크의 구조는 [그림 5]와 같다.

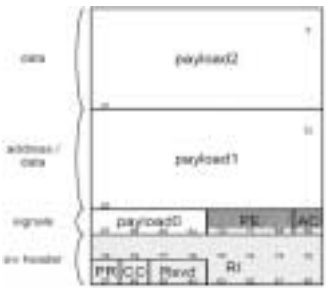


[그림 5] 온칩 네트워크 구조

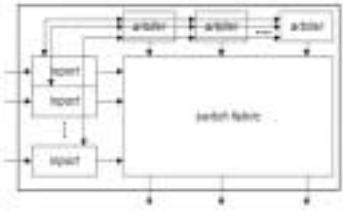
이는 네트워크의 특징을 그대로 온칩에 적용한 것이다. 온칩 네트워크에서는 하나의 IP가 네트워크를 쓸 동안 다른 IP가 다음에 네트워크를 사용하겠다는 요청을 하고 기다리는 것이 아니라 동시에 네트워크를 사용할 수 있다.

이러한 것이 가능한 이유는 다음과 같다.

1. 패킷이라는 데이터를 보내는 단위를 사용.
2. 패킷을 수집하고 원하는 목적지까지는 보내는 역할을 하는 스위치를 사용.
3. IP들과 스위치까지 서로 다른 패스로 연결이 되어 동시에 여러 패킷을 네트워크에 전송가능.



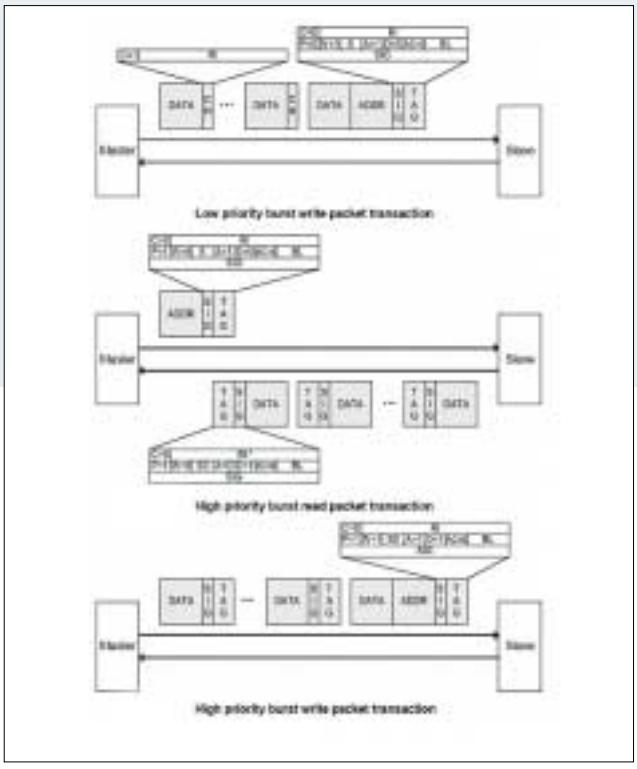
[그림 6] 패킷 구성



[그림 7] 스위치 구조

패킷은 [그림 6]과 같이 하나하나가 목적지와 출발지 그리고 패킷의 특징을 담고 있는 Tag를 갖고 있어 서로 다른 IP에서 만들어진 패킷이 서로 섞여 있어도 Tag를 디코딩하고 원하는 목적지에 순차적으로 보낼 수가 있다. 이러한 Tag를 디코딩을 하여 원하는 목적지에 순차적으로 보내는 기능을 스위치(Switch Fabric)가 담당하고 있다. 스위치의 구조는 [그림 7]과 같다.

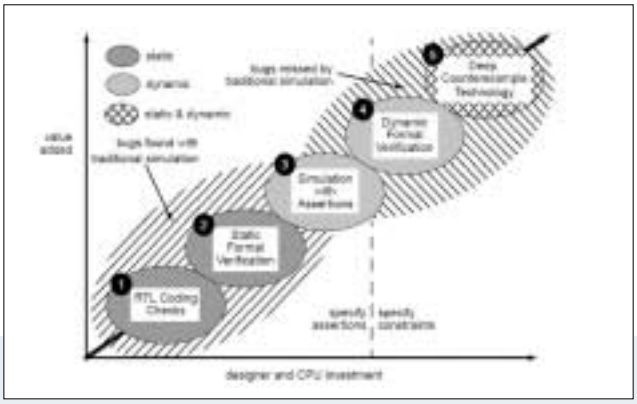
스위치는 Inport를 통해 여러 목적지를 갖고 있는 패킷들이 들어오게 된다. 이는 Switch Fabric을 통해 원하는 목적지로 보내지게 된다. 각각의 Inport는 Switch Fabric을 통해 모든 목적지로 연결이 되어 있다. Inport에서는 Tag를 디코딩을 해서 중재기(Arbiter)에 Outport 요청 신호를 보내게 된다. 각각의 Outport는 중재기를 갖고 있고, 중재기는 Outport가 비어 있다면 요청신호를 받아들이고 Inport에 있는 패킷을 Outport로 보내게 된다. 동시에 Outport의 개수만큼의 패킷을 목적지에 보낼 수 있다. Outport가 사용 중이라면 패킷은 Inport에서 대기를 하게 되어 Queuing이 일어나는데 버스에서 하나의 마스터가 원하는 모든 작업을 마치는 동안 버스의 사용 승인을 기다리는 것보다는 작은 수 패킷을 기다리게 될 것이다. 온칩 네트워크에서는 컴퓨터 네트워크와는 달리 [그림 8]과 같은 Burst read, Burst write 전송을 지원하고 있다. 데이터 전송은 대부분이 순차적으로 증가하는 주소를 갖고 있는 데이터의 read나 write 동작이다. 매 번 요청하는 데이터의 주소를 보내는 것은 통신 구조 Bandwidth의 낭비이다. 이를 지원하기 위해 데이터 크기와 연속 전송 횟수에 따라 Burst 전송을 지원한다. 이를 위해서는 스위치에서 Burst 전송으로 되어 있는 패킷은 마스터나 슬레이브로 전송을 할 때 중간에 또 다른 마스터나 슬레이브의 패킷이 섞이지 않도록 해야 하기 위함이다.



[그림 8] Burst Read/Write 전송

3. Platform Verification

Assertion 기반 검증



[그림 9] Assertion 기반 검증 단계

SoC 플랫폼을 이용하여 설계하는 동안 발생되는 설계 에러와 버그를 찾아서 정정하기 위한 검증 방법과 과정에 대해서 설명한다. AMBA 기반 플랫폼은 프로세서 코어나 DSP와 주변 모듈들이 AHB/APB 버스에 연결되어 있다. 설계된 하드웨어 모듈을 플랫폼에 통합 위해서 각 하드웨어 모듈의 기능적 검증(Function Verification)이 필요하다. Assertion을 이용하여 빠른 시간에 설계의 오류와 버그를 찾아서 정정할 수 있으며 통합되는 하드웨어 모듈과 인터페이스 모듈이 플랫폼의 버스 프로토콜에 적합한가를 검사할 수 있다. Assertion 기반 검증 방법은 기존의 시뮬레이션 방법에 비해서 검증 과정 동안 설계의 관찰성과 제어성을 향상시킬 수 있다. 이를 위해서 Assertion을 이용한 검증 방법과 Formal 검증 방법이 사용된다. Assertion 검증 방법은 설계된 하드웨어 모듈 내부에 Assertion을 추가하거나 Assertion 모니터 모듈을 연결하여 시뮬레이션을 통해서 설계 의도에 맞게 동작하는가를 검증하므로 검증 과정의 관찰성을 높일 수 있다. Formal 검증 방법은 설계 모듈의 속성을 검사하는 방법으로 Assertion을 만족하는가를 검증하는 수학적 방법으로 정적(Static) 과 동적(Dynamic) 검증 방법이 있다. 정적 검증 방법은 Assertion에 관련된 모든 가능한 동작을 광범위하게 탐색하고 Assertion의 위반 여부를 결정하여 주는 검증 방법이다. 이 방법은 블록 레벨의 검증에 사용되며, 테스트벤치 없이 RTL 검증을 수행하여 deep bug, corner case bug등을 찾아 줌으로 수백만 시뮬레이션 싸이클을 단축하는 장점을 가진다. 동적 검증 방법은 테스트 벡터에 의해 생성되는 내부 상태를 관찰하여 모듈 Assertion의 잠재적 위반을 찾아내는 방법이다. 테스트 벡터를 이용하여 생성된 내부 상태 조건들을 이용하여 빠르게 잠재된 버그를 찾아낸다.
Assertion 기반 검증 단계는 다음과 같다.

Assertion 기반 시뮬레이션과 설계 규칙 검사

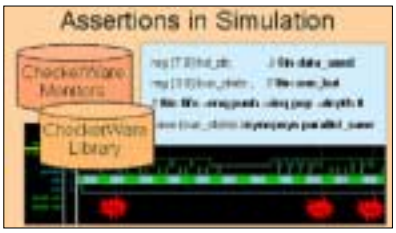
RTL 설계의 내부 블록이나 Worry case부분이나 복잡한 RTL 구조 내에 시뮬레이션이 가능한 Assertion을 추가한다. 특히, 많은 오류나 버그가 발생될 수 있는 인터페이스 프로토콜 부분에 Assertion을 추가한다. 시뮬레이션이 가능한 Assertion을 추가한 모듈을 시뮬레이션하여 설계 규칙 에러 검사하여 에러나 버그를 찾아낸 후 수정한다.

제약사항이 없는 정적 Formal 검증

위반 사항이 발생하지 않는 Assertion을 결정하고 제거한다.

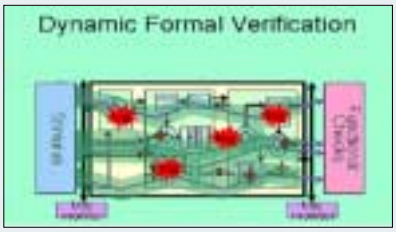
Assertion에 의한 시뮬레이션

테스트벤치를 이용하여 Assertion 시뮬레이션을 통해서 버그를 발견하는 과정이다.



부분적인 제약조건을 가지고 동적 Formal 검증

테스트벤치에 의해서 생성된 내부 조건을 검사하기 위한 제약조건을 Assertion에 부분적으로 추가하여 시뮬레이션을 통해서 빠르게 버그를 찾아낸다.



완전한 제약조건을 가진 정적 Formal 검증

Assertion에 관련된 모든 가능한 동작을 광범위하게 탐색하여 버그를 찾아내는 정적 검증 방법으로 위 단계에서 찾아내지 못한 버그를 찾는데 사용되며, 많은 시간이 소요된다.

Assertion 기반 온칩 네트워크 플랫폼

ARM의 AMBA 버스와 온칩 네트워크로 구성된 플랫폼에 Assertion 검증 방법을 도입한 플랫폼의 구조는 [그림 10]과 같다. ARM 코어와 데이터 전송을 담당하는 DMAC는 마스터 모듈로 AHB 버스 인터페이스를 갖는다. 하드웨어 모듈은 슬레이브 모듈로 응용분야에 필요한 하드웨어 및 주변 모듈로 구성된다. 이 모듈들이 고속의 병렬 처리가 가능한 네트워크 개념의 스위치를 이용하여 온칩 네트워크 구조로 연결된 온칩 네트워크 플랫폼이다.

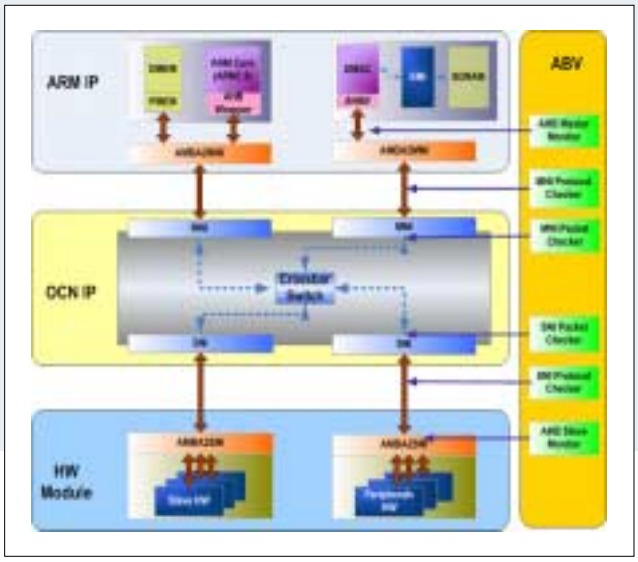
AMBA 버스에 연결된 마스터와 슬레이브 IP들을 온칩 네트워크에 연결하기 위해서 AMBA2OCN 인터페이스 모듈을 사용한다. AMBA2OCN 인터페이스에는 마스터 모듈인 ARM 코어와 DMAC를 마스터 네트워크 인터페이스(MNI)를 연결하기 위한 AMBA2MNI와 슬레이브 모듈과 슬레이브 네트워크 인터페이스(SNI)를 연결하기 위한 AMBA2SNI 모듈로 구성된다. Assertion 기반의 온칩 네트워크 플랫폼을 구성하기 위해서 각 RTL로 구성된 하드웨어 모듈에 Assertion를 추가한다. 데이터 전송을 담당하는 DMAC의 기능 검증을 위해서 Assertion을 추가한다.

DMAC의 AMBA 버스 인터페이스 부분에 Assertion으로 구성된 AMBA 모니터를 추가하여 AMBA 버스 프로토콜을 검증한다. AMBA 버스와 네트워크 인터페이스 상의 온칩 네트워크 프로토콜 검증을 위해서 프로토콜 검증기를 사용한다. Assertion 기반의 온칩 네트워크 플랫폼을 이용하여 SoC 설계를 진행함으로 온칩 네트워크 기반의 설계 및 검증 환경을 제공할 수 있다. SoC 설계에서 검증에 걸리는 시간을 단축할 수 있으며 설계된 IP들이 통합된 온칩 네트워크 플랫폼의 검증을 용이하게 해준다.

4. 결론

최근에 무선 멀티미디어 통신 장치를 개발하는 데 많이 사용되는 통신 기반 플랫폼을 설계하고 검증하는 방법에 대해서 설명하였다. SoC 플랫폼 기반 설계에서 중요한 요소는 응용분야에 적합한 플랫폼을 선택하여 최적화하고 검증하는 과정이다. 고성능 통신 구조를 갖는 플랫폼을 선택하기 위해 멀티레이어 AMBA 버스, SiliconBackPlane 및 온칩 네트워크 구조와 기능에 대해서 기술하였다. 온칩 네트워크와 같은 고성능 버스 구조를 갖는 플랫폼을 선택함으로 데이터 처리량이 많은 멀티미디어 응용 설계의 성능을 향상 시킬 수 있다.

플랫폼 설계에서 많은 시간이 소요되는 검증 문제를 해결하기 위해서 Assertion 기반 Formal 검증 방법을 적용하였다. Assertion 기반 SoC 플랫폼을 이용한 설계 및 검증을 통해서 설계 및 검증 시간의 단축이 가능하게 되어 제품의 시장 경쟁력을 향상 시킬 수 있고 검증된 IP의 재사용을 향상 시킬 수 있다.



[그림 10] Assertion 기반 온칩 네트워크 플랫폼

〈참고문헌〉

[1] ARM Ltd, "AMBA Specification Rev 2.0", Document Number ARM IHI 0011A, 1999.

[2] D. I. August, K. Keutzer, S. Malik and A. R. Newton, A Disciplined Approach to the Development of Platform Architectures. SASIMI, 2001.

[3] Jim Tobias (editor), VSI Alliance, Platform-Based Design DWG, List of Terms and Definitions, Revision 4, 4 March 2002.

[4] Assertion-based Verification V2.1 User Guides, 0-In Design Automation, 2003