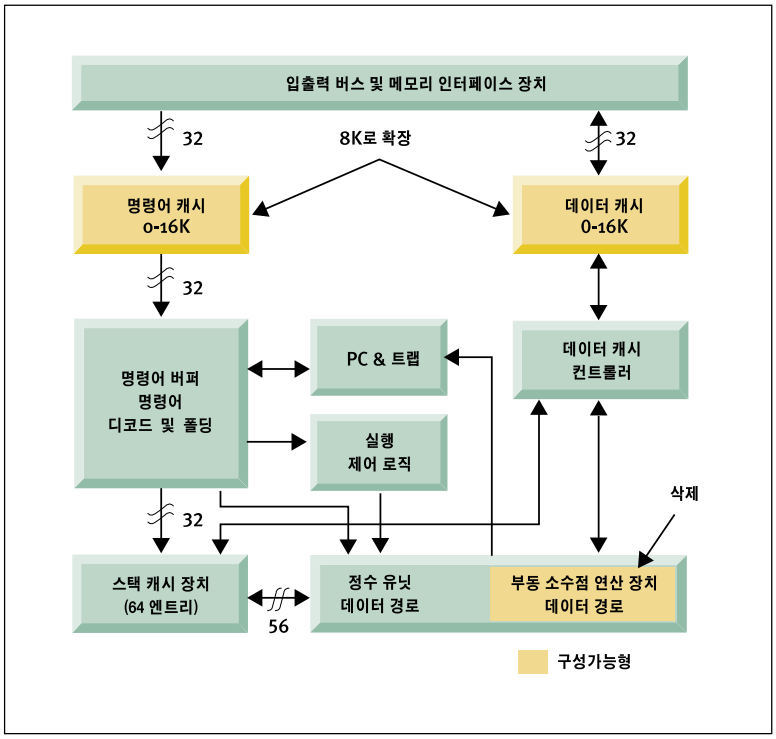
전자엔지니어
Electronic Engineering Times – Korea

디자인 코너

EDA 플랫폼 벤치마크: 데스크탑 FPGA 설계 흐름



PicoJava 코어는 하이엔드 FPGA 설계 흐름의 벤치마킹에 훌륭한 작업 대상이다.

By James Lee
Principal Consulting Engineer
Intrinsix Corp.

Bob Peterson
Technical Writer
Monterey Design Systems

Eric Shieh
Principle Staff
Shieh Resources Inc.

E-mail: jpollak@intrisix.com

ASIC 설계를 위한 다양한 하드웨어 플랫폼에 대해 진행중인 평가는 잠시 접어두고 이번 차례의 EDA 플랫폼 벤치마크는 FPGA 설계를 위한 플랫폼에 대해 살펴보기로 한다. 우리는 소자의 용량을 백만 게이트 범위까지 확장함으로써 FPGA 합성 및 레이아웃 작업이 데스크탑 플랫폼에 대한 문제를 제기하길 기대했었다. 그러나 이러한 기대는 잘못된 것이었다. 오늘날의 ASIC 표준에서 볼 때 프로세서와 메모리가 비교적 적절한 편인 윈도우즈 NT PC에서 거대한 FPGA 설계를 충분하고도 원활하게 실행할 수 있다는 사실이 우리의 벤치마크 결과 나타났다.

이 벤치마크 회로에는 우리가 (어리석게도) 합성에 Synplicity社의 Synplify를 이용하는 Xilinx Virtex 1000 FPGA와 Xilinx Alliance 배치 및 결선 툴을 어떻게든 일치시키려고 했던 피코자바(picoJava) 프로세서가 포함된다. 우리는 또한 ASIC 설계 테스트에 오랫동안 사용해 왔던 서로 다른 두 개의 Talisman hod 설계를 벤치마킹한다. ASIC을 이용해 정상적으로 작업을 진행하면 FPGA 벤치마크에 대한 짧은 전환 시간에 강한 인상을 받게 될 것이다.

평소와 마찬가지로 우리는 위와 같은 벤치마크를 실행하는 동안에 여러 가지 재미있는 심화 현상에 부딪쳤다. 하드웨어와 소프트웨어 모두에 압력을 가하려는 우리의 노력은 일상적으로 설계 공정의 한계를 분명히 나타내는 고장에서 성과를 올리게 된다. 게다가 우리는 예기치 못한 귀중한 발견을 하게 되었는데, 이번 벤치마크의 후반부에서 얻은 EDA 테스팅에 대한 중요한 이익은 바로 SCSL(Sun Community Source Licensing) 프로그램이다.

Verilog 코드의 소스

EDA 플랫폼을 벤치마킹하는데 있어서 해결 과제 중 한 가지는 적절한 설계를 찾아야 할 필요성이다. 다른 팀들이 우리의 테스트를 복사할 수 있도록 설계는 반드시 자유롭게 이용할 수 있어야 한다. 이는 우리가 설계를 소유하여 이를 사용할 수 있게 만들던지, 아니면 모두에게 개방되어 있는 공공 도메인 소스에서 설계를 얻어내야 함을 뜻한다.

또한 설계는 벤치마킹 중인 플랫폼에 스트레스를 주기에 적합한 크기와 복잡도를 가져야만 한다. 우리가 이 벤치마킹 시리즈를 시작한지 불과 얼마 안되는 짧은 시간 동안에 프로세서 속도와 메모리 용량이 향상되어 과거에 난제였던 설계들이 최근의 데스크탑 시스템에서는 무리 없이 진행되었다. 따라서 우리는 아주 정확한 수준의 복잡도를 이루어 내기 위해 256 개나 되는 많은 사례로 비교적 작은 규모의 설계를 늘리는 일에 의존해왔다. 그러나 "실제" 설계는 그 같은 구조의 규칙성이 없기 때문에 그 같은 벤치마크 회로들이 완전히 만족스럽지는 못하다.

그렇다면 SCSL의 세계를 탐구하는 일은 얼마나 기쁜 일인가! SCSL은 Sun社의 웹 사이트(www.sun.com)에 들어가면 이용할 수 있으며, 전에 Sun社의 디자이너들이 자사의 하드웨어와 소프트웨어 제품을 제작했던 것과 동일한 소스 코드가 마련되어 있다. 라이선싱 조건에 동의함으로써 당신은 Sparc 프로세서의 소스 코드와 Jini 연결 기술에서 HPC 클러스터 툴(Sun社의 메시지 통과 인터페이스 라이브러리 및 병렬 파일 시스템과 같은)과 자바 플랫폼까지 광범위하고 흥미있는 제품들을 다운로드받을 수 있다.

위의 라이선싱 조건은 IP의 한 부분에서 또 다른 부분으로 변화하지만, SCSL은 항상 초기 평가 및 개발 단계가 진행되는 동안에는 코드를 마음대로 사용할 수 있게 해준다. EDA 테스팅은 평가 단계를 뛰어 넘지 못하기 때문에 우리는 실제 설계를 위한 무료 소스를 보유하고 있다.

벤치마크 전 주지 사항

Sun社의 피코자바 코어는 본래 Java Virtual Machine이 규정한 대로 자바 바이트코드 명령어를 직접 실행하는 소형 마이크로프로세서로 구성되어 있다. 이러한 직무 설명은 "한 번 작성으로 어디서나 실행할 수 있다"는 자바의 철학에 위배되는 것처럼 보일 수도 있을 것이다. 그러나 이 개념은 플랫폼과 위치 독립성의 이점은 유지하면서 자바 프

로그램의 실행 효율성을 극대화 시킴으로써 오버헤드를 최소화하자는 것이다. 다시 말해 다양한 플랫폼에서 자바 코드를 실행시킬 수 있으며 피코자바 플랫폼에서도 자바코드를 높은 수준의 효율성으로 실행시킬 수 있다는 얘기이다. 피코자바가 본래 갖고 있는 바이트코드 실행 기능은 이러한 효율성을 가능하게 하지만, 그 밖의 프로세서들은 저스트 인 타임(just-in-time) 컴파일러를 이용하는 바이트코드를 해석하거나 역학적으로 컴파일링해야 한다.

우리는 EDA 플랫폼을 테스트 하기 위해 picoJava-II 버전을 선택했다. Sun社의 picoJava-II 다운로드 판에는 picoJava-II 프로그래머 참고 매뉴얼, 소프트웨어 개발 환경, 시뮬레이션 환경, RTL 디자인 파일, 검증 테스트 스위트, 샘플 스크립트 및 RTL 참고문서(177 MB라는 다소 큰 용량의 소스 코드와 참고 문서)가 들어 있다. 이러한 참고 문서 중 일부는 웹 브라우저를 이용하여 디자인의 Verilog 계층 구조를 살펴볼 수 있도록 해주는 여러 가지 편리한 HTML 파일들로 구성되어 있다.

피코자바 코어는 확실히 정교하고 작은 프로세서이다. Sun社의 문헌에 따르면, 피코자바 코어에는 동일한 클럭 주파수에서 동작하는 그 밖의 단일 스칼라 고성능 RISC 프로세서와 호환되는 가격 및 성능으로 레거시 C/C++을 실행할 수 있도록 순방향 명령어 폴딩이 있는 6 단계의 RSIC 파이프라인이 있다. 또한 피코자바 코어는 방법의 발동 및 국부 변수들로부터의 부하(load) 은폐 뿐만 아니라 연속 동기화 및 다양한 가베지(garbage) 수집 방법들을 제공하므로 객체 지향형 프로그래밍을 능률적으로 해준다. 가장 일반적으로 사용되는 명령어들은 하드웨어로 실행되지만, 복잡한 명령어들은 마이크로코딩되며, 가장 복잡한 명령어들은 소프트웨어로 트랩 및 에뮬레이트된다.

피코자바 코어는 Sun社의 제품이지만 가장 큰 규모의 FPGA라 할지라도 일치시키기엔 너무 크다. 따라서 부동 소수점 연산 장치를 삭제하고 코어의 명령어와 데이터 캐시를 각각 8 KB까지 확장 시켰는데, 이는 지정된 범위인 0 KB ~ 16 KB의 중간치를 나타낸다. 우리는 또한 주사의 입력/출력/허가 모듈에 연결되지 않은 입출력 소자에 대해서도 설명했다. 모든 피코자바 모듈에는 연결 및 구동되지 않는 주사 출력 핀이 있다. ASIC에서는 이러한 입출력 소자가 연결되겠지만 FPGA는 다른 방법으로 고착 고장을 테스트하기 때문에 입출력 소자가 필요 없다. 우리는 Synplify가 연결되지 않은 입출력 소자들에 대해 문제를 일으키므로 이를 제거하면 일이 쉬워질 수 있음을 알게 되었다.

Sun社는 구성 가능형 블록으로 FPU와 캐시를 지명하고, 이 블록들 역시 구성하기가 쉬웠다는 사실에 유념한다. 우리의 최종 피코자바 벤치마크에는 64만 9,770 개의 게이트가 들어 있으며 81 퍼센트의 목표 FPGA 이용도를 자랑한다.

SCSL 연구 라이선스에 대한 조건 중 하나는 Sun社의 허가 없이는 기술을 배포할 수 없으며 이와 같은 경우에는 변호사를 선임할 아무런 근거가 없다는 내용이다. 벤치마크 테스트를 복사하고자 한다면 Sun社의 웹 사이트인 www.sun.com/microelectronics/communitysource/picojava/techinfo.html에 들어가 직접 다운로드 받을 수 있다. 주사 체인과 FPU를 삭제하고 캐시를 확장하는 것을 잊지 않도록 하자. 우리는 Synplify(Synopsys의 translate_off 및 translate_on 의사 주석을 이용하는)에서 예를 든 RAM을 "블랙박스화"하여 Alliance가 Virtex RAM 블록을 이용하는 RAM을 구현하도록 만들었다. 당신의 피코자바 버전이 우리의 버전과 동일한지의 여부는 대규모 FPGA 설계 흐름을 테스팅하기 위해 동등한 설계를 얻는 것보다는 덜 중요한 문제이다.

우리는 수정된 피코자바 코드에 Sun社의 테스트 벡터를 실행시키지 않았으며, 혹은 다른 어떤 방법으로도 설계를 검증하지 않았다는 점에 유의하자. EDA 벤치마크의 특성이 설계 품질의 문제점을 하찮게 만들기 때문에 검증 툴이 테스트(예를 들면 우리의 Verilog-XL 벤치마크)에 고유한 것이 아닌 이상 우리의 벤치마크 설계를 검증하지 않는다.

Talisman 벤치마크

과거의 벤치마크는 EDA 툴과 하드웨어 플랫폼의 성능이 설계마다 크게 달라질 수 있는지의 여부를 반복해서 실험해 왔다. 따라서 피코자바가 오늘날의 FPGA 흐름을 문제 삼기에 이상적인 벤치마크인 것처럼 보이지만, 우리는 또한 ASIC 벤치마크의 비축고인 Talisman hod로부터의 설계를 포함시키고 있다. Talisman은 Microsoft社와 Cirrus Logic社가 공동설계한 그래픽 엔진이다. Microsoft社는 우리의 벤치마킹 프로젝트를 위해 소스 코드(그중에도 hod라고 불리는 섹션)의 상당 부분을 자유롭게 이용할 수 있도록 해주었다.

우리의 기존 Talisman hod는 FPGA에 맞춰 넣기에는 너무 크기 때문에 94만 6,400 개의 게이트 버전이 될 때까지 삭감한 다음 39만 6,800 개의 게이트 버전을 얻기 위해 이를 더 삭감했다. 이러한 벤치마크 회로들의 Verilog 소스는 www.isdmag.com/edabenchmark에서 얻을 수 있다.

94만 6,400 개의 게이트 버전 Talisman hod는 우리의 목표 FPGA를 거의 백 퍼센트 이용해야 하기 때문에 우리는 이를 Talisman_100으로 더빙했다. 우리는 동일하게 제로의 확률을 갖는 것은 없다는 사실을 물리학에서 배웠으므로 이 설계가 실제로 FPGA에 적응될 확률은 제로가 아니라는 점을 알고 있었다. 그러나 우리는 이러한 노력으로 제로 가까이에서 감지하게 될 것이라고 생각했다. 단지 데스크탑 PC에 어느 정도의 해결 과제를 제공해 주는 벤치마크를 원한 것뿐이었다. 이와 반대로 39만 6,800 개의 게이트 버전에는 단지 60 퍼센트의 이용도(따라서 Talisman_60)만이 필요하므로 이는 보다 현실적인 설계를 나타낸다. 우리의 목표 FPGA는 Xilinx Vertex 계열 제품의 하나인 XCV1000이다. 이 거대한 제품에는 112만 4,022 개의 시스템 게이트(2만 7,648개의 로직 셀), 512개의 사용자 입출력 핀 그리고 클럭 분산을 깔끔하게 제공하기 위한 네 개의 디지털 지연 잠금 루프가 있다. Xilinx社는 출력 지연에 대한 클럭이 3 나노초 이하라고 주장한다.

FPGA의 구성 가능형 로직 블록(CLB)은 RAM 블록, 입출력 상호연결 영역(VersaRing), 구성 로직 및 입출력 블록으로 둘러싸여 칩의 중앙에 위치하고 있다. 따라서 조사 테이블, 멀티플렉서, 플립플롭 및 그밖의 요소들은 각 CLB에서 네 번씩 복사된다. 각각의 CLB에는 라우팅을 목적으로 하는 슬라이스가 두 개씩 있는데, 이들 각 슬라이스에는 두 세트의 자원이 들어있다.

범용 회로를 구현하기 위해 이러한 코스 그레인 FPGA 자원을 이용하는 데에는 특수 합성 및 레이아웃 지원이 필요하다. 앞에서 언급했듯이 우리는 합성에 Xilinx社의 Alliance 배치 및 결선 툴에 의해 잇따르는 Synplicity社의 Synplify를 이용했다. Alliance의 독특한 기능에는 변환, 매핑, 타이밍 분석, 배치 및 결선 그리고 최종 타이밍 분석 등이 있다.

GUI 측면에서 보기

만일 ASIC 설계에 익숙해져 있다면 FPGA 툴이 약간 다르다는 것을 알게 될 것이다. 더 정확히 얘기하자면 FPGA 툴은 그래픽 사용자 인터페이스를 이용하는 데 특히 중점을 두고 있다. 그 결과 Synplify와 Alliance는 모두 대부분의 ASIC 툴보다 사용하기가 더 쉽지만, 배치 모드가 항상 완벽하게 동작하지는 않는다.

예를 들어 배치 모드에서 이 툴을 처음 실험했을 때 Synplify는 우리의 네트워크 서버로부터 설계 파일을 이용하는 데 헤메는 양상을 보였다. 우리가 Synplify의 그래픽 사용자 인터페이스를 통해 상호 작용하는 방식으로 Synplify를 실행시키자 파일을 복구하는 데 아무런 문제가 없었으며, 로컬 디스크에서 설계 파일을 이용하는 것은 결코 문제가 되지 않았다. 우리는 네트워크 트래픽으로 인한 변동을 피하기 위해 항상 각 기계의 로컬 디스크에서 벤치마크를 실행하므로 네트워크 액세스 상의 블록은 문제점이 아니었다. 또한 우리가 키보드에서 Synplify를 중단하자 NT의 작업 모니터 공정 목록은 분명히 마지못해 넘겨주는 또 다른 몇 초 동안 지속되는 작업을 보여준다는 사실을 알아냈다. 그러나 Synplify가 실행을 정상적으로 완료하면 헤메지 않게 된다. 따라서 정확한 실행 시간을 측정하는 데는 아무런 문제가 없었다.

배치 명령으로 Alliance의 기능을 모두 제어할 수는 없으므로 Alliance를 자동으로 실행하는 데는 다소 문제가 있었다. 우리는 이러한 기능들을 실행하기 위해 매크로를 기록할 수 있었지만 스크립트로 이 매크로를 시작했다. 퇴장 명령을 내릴 때 툴은 확실히 퇴장하길 원하는지를 묻는 대화 상자를 팝업 시키기 때문에 Alliance의 타이밍 분석기 기능을 우리의 벤치마크에 포함시킬 수 없었다. 배치 파일은 "예"에 클릭하는 방법을 알지 못한다.

스크립트는 일관된 방법으로 복잡한 툴의 기능을 반복할 수 있게 해주므로 벤치마킹 공정에 필수적이다. 우리가 신중하게 끼워 넣은 스크립트의 세트는 또한 툴의 실행 시간을 일관되게 측정하기 위한 기준을 제공한다. 만일 툴이 말끔히 퇴장하지 않는다면 실행 시간을 정확하게 잴 수 없으므로 우리는 벤치마크에서 타이밍 분석기를 그냥 제외시켰다.

만일 당신이 벤치마킹 업무에 종사하고 있지 않다면 설계 공정이 짧고 반복이 거의 없는 한 그래픽 사용자 인터페이스에 보다 많이 의존하는 것이 당연하다. 어째서 겨우 몇 초 동안만 사용하려는 스크립트를 개발하는 것일까? 그것은 당신이 원하는 방식으로 FPGA 툴을 설치하고 몇 초 동안 툴을 실행시키면 작업이 모두 완료되기 때문이다.

전통적으로 FPGA는 이처럼 빠른 설계 흐름을 고려하기에 충분할 만큼 작은 형태를 유지해 왔다. 마침 FPGA의 용량이 증가했을 때 설계 툴의 개선은 전환 시간을 다루기 쉽게 유지시켰다. 그러면 이제 시장에 진입하고 있는 수백만 게이트급의 FPGA들이 이같은 상황을 변화시킬 수 있을까? 또한 배치 모드 처리에 의존하는 연산 팜의 사용이 늘어나는 문제는 어떻게 될까?

우리는 Synplicity社의 FPGA 제품 마케팅 이사인 Jeff Garrison씨에게 배치 모드 대 GUI의 운용에 관한 회사의 관점에 대해 질문했다. 그는 Synplicity社는 GUI와 함께 TCL과 배치 모드를 모두 지원한다는 사실을 언급하면서 "스크립트 덕택에 효율적으로 라이선싱하고 일관성 또한 유지하게 되지만, 그같은 문제점은 사실 개인의 선호도에 관한 문제"라고 말했다. 우리는 사용하기 편리하고, 제어하기 쉽도록 만드는 데 중점을 두고 있으므로 버튼을 클릭하는 일이 적다. 우리의 원칙은 툴보다는 오히려 설계를 제어하도록 하는 것이다." 실제로 Synplify는 설치에 거의 제약이 없다.

Garrison씨가 말한 모든 내용은 진행중인 변화에 주목한다는 것이다. "보다 규모가 큰 FPGA를 도입한 덕택에 우리는 규모가 보다 큰 업체에서 배치 모드를 더 많이 사용하기 시작하는 것을 보고 있다. 보다 많은 ASIC 디자이너들이 FPGA 설계 쪽으로 이동함에 따라 우리는 배치 모드가 더 많이 사용되는 것을 보게 될 것으로 기대한다"고 그는 말했다.

끊임 없는 개선

ASIC과 FPGA 툴 간의 배치 대 GUI의 차이점 외에도 우리는 FPGA 툴이 훨씬 더 자주 업그레이드를 경험하는 것처럼 보인다는 사실을 깨달았다. Synplicity와 Xilinx社의 툴은 모두 우리가 이 툴에 대해 연구하는 동안 수정을 경험하였고, 이 툴들의 개선 내력은 오래 지속되는 많은 ASIC 툴과 비교하면 거의 광분에 가까운 정도임을 알 수 있었다. 우리는 ASIC 툴이 사용되어 온 지가 더 오래 됐으므로 보다 성숙할 것이라 생각하긴 했지만 차이점을 완전히 밝힐 수는 없었다.

Synplicity社의 Garrison씨는 미세 입자 ASIC 구조와는 반대로 FPGA는 한 벤더에서 다른 벤더로 폭넓게 변화하는 코스 구조를 가지고 있다고 설명한다. 최고의 결과를 위해 합성은 각 FPGA 아키텍처에 대해 주문화 되어야 한다. "사람들은 FPGA 합성이 ASIC 합성보다 사실상 더 어렵다는 사실을 깨닫지 못하고 있다"고 그는 말했다.

오늘날의 FPGA 시장에서 수많은 벤더들에게 뒤쳐지지 않으려 하다 일이 더욱 어려워진다. 최신 소자들에 대한 즉각적인 지원이 갖는 중요성을 생각할 때는 특히 그렇다. 그 결과 Synplicity社에서는 일반적으로 1사분기에 한 번 정도씩 새로이 갱신된 소프트웨어를 출시하고 있다고 Garrison씨는 말했다.

우리의 벤치마크 노력은 Synplify의 첫 번째 버전(5.1.5a)이 전체 피코자바 설계를 처리할 수 없기 때문에 이처럼 열정적인 계획에서 이익을 얻었다. 우리는 나머지 설계에서 피코자바의 정수 단위를 분리하여 서둘러 진행했지만 결과는 Alliance 소프트웨어(2.1i 버전)와 충돌했다. 대략 그 시기에 우리는 새로운 버전의 Synplify(5.2.2a)와 Xilinx(서비스 팩 21i_sp1_alliance_pc.exe 및 21i_sp1_data_pc.exe)를 모두 수령했다. 이 갱신 버전들은 문제를 해결해 주었으며 새로운 버전의 Synplify는 이전 버전보다 약 두 배 정도 실행 속도가 빨랐다. □

[Integrated System Design]