하드웨어 설계 데이터 관리에 관한 고찰

A Consideration for Management of Hardware Design Data

이재철(J. C. Lee) 컴퓨터구조연구실 선임연구원
김용연(Y. Y. Kim) 컴퓨터구조연구실 선임연구원, 실장

대규모 시스템의 개발에 있어서 하드웨어 설계 데이터가 방대해짐에 따라 데이터의 형상 관리가 필요하게 되었고, 보다 효율적인 설계 관리하에서 신뢰성 있는 하드웨어 설계용 라이브러리를 설계하기 위해서는 데이터 관리 도구가 요구된다. 고속병렬컴퓨터 시스템 개발을 위한 하드웨어 설계 환경에서는 설계 데이터의 효율적인 형상 관리를 위하여 TDM(Team Design Manager) 설계 관리 도구를 적용하였다. 본 고에서는 여러 워크스테이션(머신)들로 구성되어 클라이언트/서버 컴퓨팅을 지원하는 분산 하드웨어 환경에서의 설계 데이터 형상 관리환경 및 하드웨어 설계 데이터의 관리기법에 관하여 고찰하였다.

I. 서 론

시스템의 하드웨어 설계 및 개발에 있어서 설계의 규모가 점차 커짐에 따라 설계의 복잡도가 증가하게 되면서 하드웨어 설계 데이터의 관리가 절실히 요구되고 있다.

대규모 시스템 설계상에서 여러가지 버전의 설계 데이터 유지 및 보호 조치가 어려웠고, 오래된 설계 데이터들의 상황을 추적할 수가 없었다. CAD 시스템에서 개발된 모든 CAD 데이터의 설계 관리에 관련된 정보 제공을 위해서 항시 온라인으로 정보 접근이 가능해야 한다. 설계 데이터 관리에 있어서 데이터를 수정한 한번의 처리가 일어나면 이전의 변경된 데이터 값은 잃어버리게 되었다. 보다 발전된 관리기법에서는 설계 과정의 객체 버전을 관리해 주는 것을 데이터베이스가 지원한다. 이러한 것은 일반적으로 소프트웨어 설계 및 CAD 분야에서 적용된다[1-5].

하드웨어 설계 데이터들의 연관성 확인 및 안정된 설계 데이터들을 효율적으로 공유하기 위해서는 설계 데이터 관리 도구가 요구된다. 고속병렬컴퓨터, SPAX (Scalable Parallel Architecture computer based on X-bar network) 시스템 개발을 위한 CADPIA (CAD for Personal and Intelligent Agency) 환경은 여러 워크스테이션(머신)들로 구성되어 클라이언트/서버 컴퓨팅을 지원하는 분산 하드웨어 환경이다. 이 하드웨어 설계환경에서는 모든 설계자들의 데이터가 효율적으로 관리될 수 있도록 TDM[6, 7] 도구

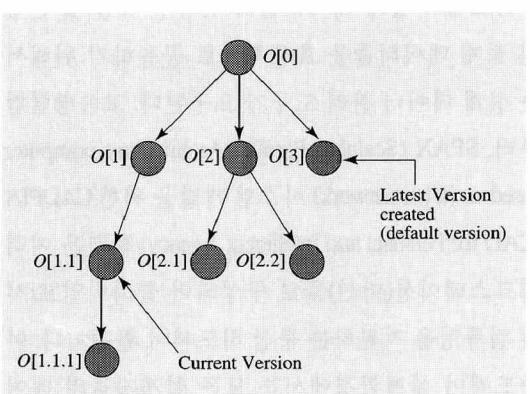
를 적용하였다.

본 고에서는 고속병렬컴퓨터 하드웨어 개발에서 적용한 하드웨어 설계 형상관리 및 라이브러리 개발을 위한 환경 중심으로 살펴본다. II장에는 설계 데이터 관리의 개념을 이해하기 위한 데이터 관리의 기본 개념을 언급하고, III장에서는 하드웨어 설계환경인 CADPIA의 분산 하드웨어 환경을 소개한다. 그리고 IV장에서는 고속병렬컴퓨터의 하드웨어 설계를 위한 CAD용 라이브러리 설계 관리의 적용 사례에 대하여 기술하며, V장에서 결론을 맺는다.

II. 데이터 관리의 기본 개념

설계 데이터 관리에 있어서 보다 발전된 관리 기법에서는 설계 과정의 객체 버전을 관리해 주는 것을 데이터베이스가 지원한다. 이러한 것은 일반적으로 소프트웨어 설계 및 CAD 분야에서 적용된다.

설계 관리 도구는 데이터 관리를 위해 버전 및 형상 관리, 변경 사항 통보, 설계 변경의 용이성, 설계 데이터의 보호 등의 기본적인 요구 사항들을 제공한다.

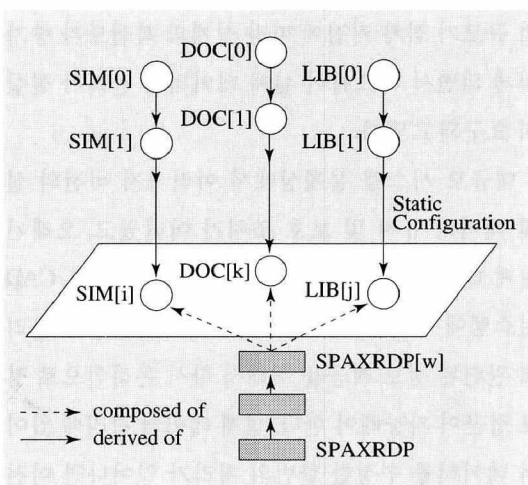


(그림 1) Version history

버전 관리에 있어서, 버전은 순간적인 시점에서 생성된 설계 객체의 의미 함축의 속성을 가지며, 주어진 한 객체의 버전은 그 이전 버전의 시점 이후에 수정되어 생성되고, 초기 버전과 함께 시작한다. 이러한 버전의 파생은 버전의 이력에 의해 제공되며, 이러한 개념을 그래픽으로 나타낸 것이 (그림 1)이다. (그림 1)에서 트리의 노드들은 버전을 나타내고 화살표는 파생 관계를 보여주는 것이다. 노드들의 버전 *O*[1], *O*[2], 그리고 *O*[3]는 *O*[0]에서 파생된 갱신 버전들이다.

형상 관리에 있어서 형상은 정적의 형상과 동적의 형상으로 나누어 질 수 있다. 정적 형상은 부품 객체의 형상과 규격 버전 사이에서 명확하고 영구히 그 링크를 정의할 수 있는 것이며, 반면에, 동적 형상은 하나 이상의 규격 버전들이 동작 시간에서 형상을 이룰 수 있는 것을 말한다.

(그림 2)는 SPAX 개발을 위한 설계 데이터의 형상 개념을 나타낸 것이다. (그림 2)에서 나타낸 SPAXRDP[*w*]는 출하된 SPAX 시스템의 설계 데이



(그림 2) SPAX 설계 데이터의 형상 개념

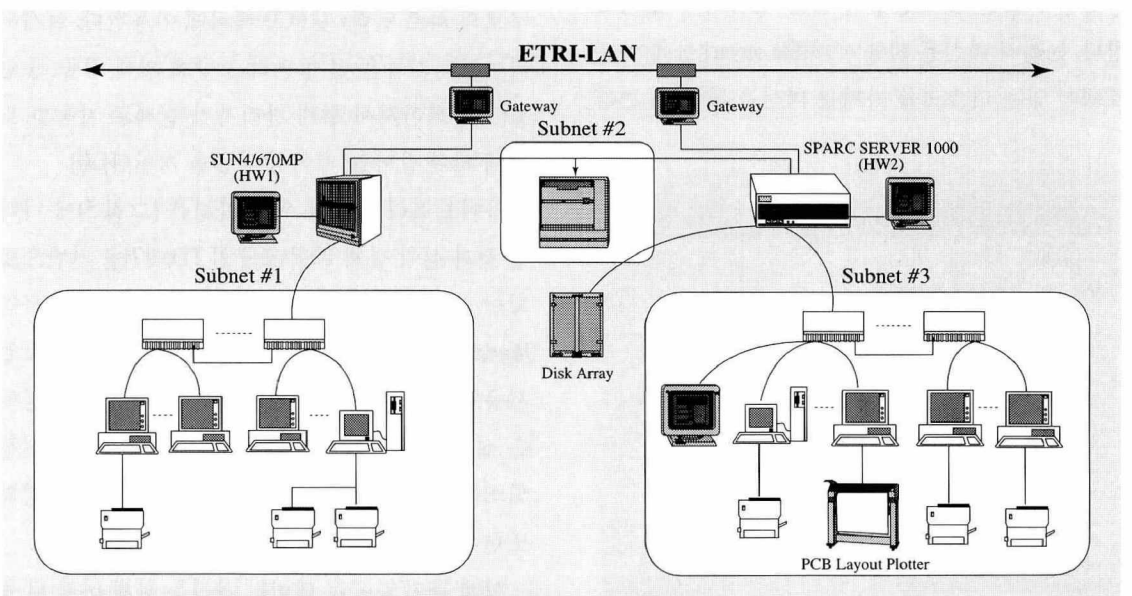
터 패키지 즉 SPAXRDP의 버전을 의미하며, SIM[*i*], LIB[*j*], DOC[*k*]는 각각 SPAX의 시뮬레이션(SIMulation) 데이터, SPAX 하드웨어 설계용 라이브러리(LIBrary), 설계용 지침 문서(DOCumentation)를 의미하는 것들에 대한 버전들이다.

하드웨어 설계 데이터의 버전 변경이 자주 일어나고, 설계자간의 안정된 데이터들을 효율적으로 공유하기 위해서는 버전 변경의 통보 기법이 필요하다. 설계자가 개발한 설계 데이터의 현재 버전을 체크아웃 연산을 통해 그 내용을 변경할 수 있으며, 개정된 버전을 체크인 연산을 하고 설계 관리자에게 변경 요구를 하여 설계 관리자로부터 검증을 거쳐 변경 승인이 이루어진다. 이 때 설계자들에게 메시지를 전달하게 된다. 관리 상태에 있는 설계 데이터들은 설계자들 임의대로 변경이 불가능하며, 반드시 체크아웃, 체크인 연산을 통해야만 버전 갱신을 할 수 있도록 보호되고 있다[1, 8].

III. CADPIA의 하드웨어 환경

CADPIA[9]의 하드웨어 구성은 네트워크 과부하 및 서버 시스템으로의 부하 집중에 의한 성능 저하 등의 문제점들을 해결하기 위해, 양 방향 서버 시스템 및 팀별 서브서버(subserver) 개념을 도입하여 (그림 3)과 같이 클라이언트/서버 컴퓨팅을 지원하는 분산 하드웨어 환경으로 구성하였다.

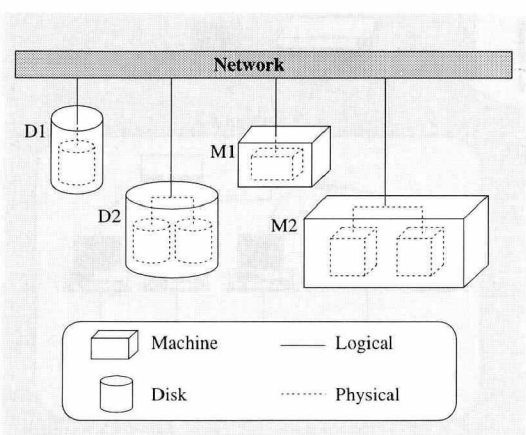
CADPIA 하드웨어 구성의 특징은 대용량 멀티프로세서인 두 대의 서버 시스템을 양 방향으로 설치하여 동일한 소프트웨어를 분리, 탑재함으로써 한 대의 서버 시스템으로 집중되었던 과부하를 분산시키며, 한 대의 서버 시스템 고장 시 전체 시스템으로의 파급 영향을 방지할 수 있도록 하였다. 뿐만 아니라 대용량의 CAD 소프트웨어 및 설계 데이터를 탑재시킬 수 있도록 디스크 어레이를 장착하였으며, 각 서브 네트워크 내에 여러 클라이언트 시



(그림 3) CADPIA 하드웨어 네트워크의 모습

스템들이 연결되는 또 다른 서브 네트를 설치, 운용함으로써 가능한한 메인 서버 시스템의 과부하가 분산되도록 시스템을 구성하였다. 이러한 하드웨어 환경은 대규모의 CAD 작업을 여러 명이 동시에 수행할 때 발생하는 성능 저하를 최대한 방지할 수 있으며, 시스템 고장에도 대비할 수 있는 분산 하드웨어 환경이다.

(그림 3)과 같은 분산 하드웨어 환경은 네트워크 파일 시스템(NFS)으로 운용되며, 여러 논리적 디스크를 걸치고 있다. 대개 한 논리적 디스크는 한 파일 시스템에 의해 만들어진다. 이를 사용자(혹은 CAD 도구 사용자)의 관점에서 분산 하드웨어 설계 환경의 모델로 나타낸 것은 (그림 4)와 같다. (그림 4)를 살펴보면, 3개의 물리적 머신 및 디스크 그리고, 2개의 논리적 머신 및 디스크로 구성되어 있다. 이 모델에서 논리적 머신(혹은 프로세서), 논리적 디스크, 논리적 네트워크를 구별할 수 있다. (그림 3)에서의 프린터, 주변장치 등은 생각하지 않기로 한다. 논리적 머신은 파일 서버(file server) 상에 마운트되어 있는 디스크를 가지는 머신을 말한다. 논리적 머신이나 논리적 디스크는 한 개 이상의 머신이나 디스크로 구성된다. CAD 설계 작업에서의 예를 들면, (그림 4)의 논리적 머신 M2는 물리적 머신의 사용자가 또다른 사용자와 어느 한 파일 서버에 마운트하여 동일한 파일 시스템을 사용하며, 물리적으로 디스크 D1을 가진 머신이라 할 지라도 실제로, 데이터의 저장 및 접근이 이루어지는 것은 D2에 속하는 물리적 디스크이다. 이 모델은 하드웨어 설계 데이터의 저장이나 CAD 데이터베이스를 위한 배치(allocation) 메커니즘에 이용될 수 있다[8].



(그림 4) 분산 하드웨어 설계 환경 모델

IV. 라이브러리 설계 관리의 적용 사례

1. 라이브러리 설계 데이터 관리 환경

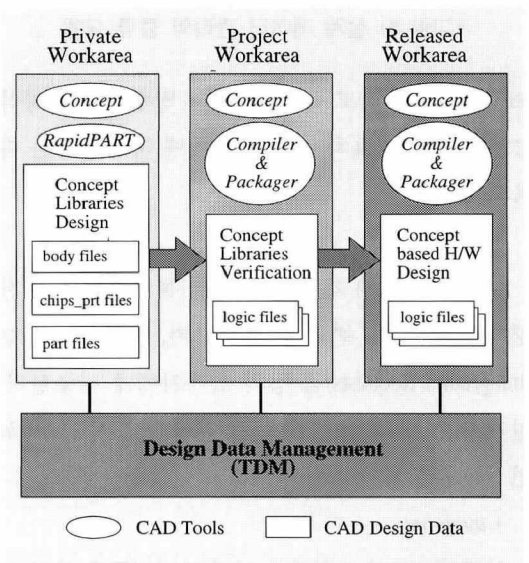
라이브러리 설계는 SPAX 시스템의 하드웨어 설계 및 PCB(Printed Circuit Board) 제작에 필요한 것들을 대상으로 하며, 하드웨어 회로도 편집(CAD 도구로의 회로 입력) 및 PCB 제작에 이용된다. 설계된 라이브러리들은 설계 관리 도구에 의해 검증, 승인을 요청되어져서 설계 관리자가 실제로 자동화 도구에 의한 컴파일, 패키징 검증을 거친다[10].

라이브러리 설계를 위한 환경은 (그림 5)에 나타낸 것과 같이 설계 관리 도구인 TDM[7]을 기반으로 되어 있다. 그리고 *Concept* 라이브러리 설계 도구인 *RapidPART*, *Packager* 등의 유틸리티를 병행적으로 사용한다. 라이브러리 설계 관리를 위한 작업 영역은 설계 관리 도구에서 설정하는 환경하에서 설정되어져야 하며, 상위 설계 관리자를 위한 작업영역 또한 설계 관리 도구에 의해 설정되어져야 한다.

설계 관리 도구는 데이터 관리를 위해 버전 및 형상 관리, 변경 사항 통보, 설계 변경의 용이성, 설계

데이터의 보호 등의 기본적인 요구 사항들을 제공한다[10, 11].

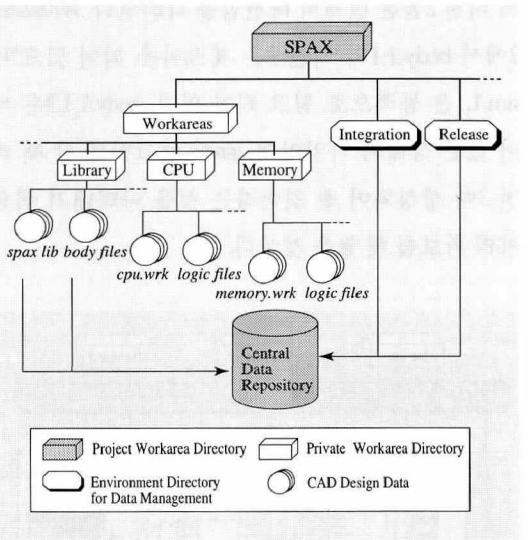
설계자가 개발한 설계 데이터의 현재 버전을 체크아웃 연산을 통해 그 내용을 변경할 수 있으며, 개정된 버전을 체크인 연산을 하고 설계 관리자에게 변경 요구를 하여 설계 관리자로부터 검증을 거쳐 변경 승인이 이루어진다. 이 때 설계자들에게 메시지를 전달하게 된다[1-5].



(그림 5) TDM 기반의 라이브러리 설계 환경

관리 상태에 있는 설계 데이터들은 설계자들 임의대로 변경이 불가능하며, 반드시 체크아웃, 체크인 연산을 통해야만 버전 갱신을 할 수 있도록 보호되고 있다[2].

(그림 6)은 SPAX 시스템의 하드웨어 설계 데이터 관리를 위한 프로젝트 디렉토리, 작업 영역 디렉토리, 설계 데이터 관리를 위한 환경 즉 *integration*, *release* 디렉토리 등으로 조직 구성된 모습을 보여



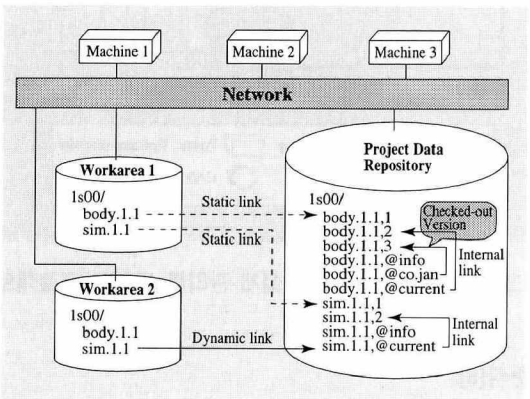
(그림 6) SPAX 시스템의 설계 관리를 고려한 프로젝트 조직

준다[8].

(그림 7)은 분산 디스크 환경에서의 라이브러리 설계 데이터의 저장 및 관리 구조를 나타낸 것이다. (그림 4)와 같은 환경에서의 3개의 물리적 머신과 디스크 상태에서 라이브러리 설계 작업을 하지만, 네트워크 파일 시스템 운용상에서의 한개의 논리적 머신과 디스크 구조의 환경과 같은 상태이다.

라이브러리의 설계 및 관리에 있어서 버전 제어 및 형상 관리의 개념을 정의한다는 것은 매우 복잡하지만 연산 메커니즘을 통하여 살펴보기로 한다[10]. (그림 7)에 나타낸 각 Workarea에서의 ls00/과 같은 디렉토리들은 형상으로서 사용되며, 정적 참조는 정확한 파일 버전에서 가리키는 링크를 의미하며, 동적 참조는 현재의 파일 즉, 저장 디렉토리에서 ",@current" 링크로 가리키는 것이다. *Workarea* 1에서의 body.1.1과 sim.1.1은 버전 1에 정적으로 링크되어 있는 것을 나타낸다. 그리고 body.1.1과 sim.1.1

의 버전 2들은 현재의 버전임을 나타낸다. *Workarea* 2에서 body.1.1의 버전 3은 체크아웃 되어 있으며, sim.1.1은 동적으로 링크 되어 있다. body.1.1,3은 비어 있는 상태의 파일이며, jan이 체크인을 할 때 버전 3이 생성되어 질 것이라는 것을 나타내기 위한 자리 확보를 해 놓은 것이다.

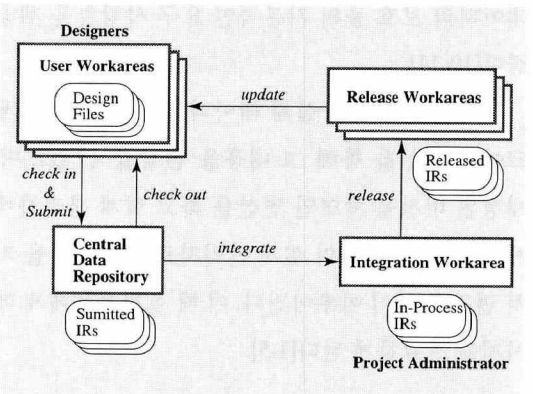


(그림 7) 분산 디스크 환경에서의 라이브러리 설계 데어터 저장 및 관리 구조

2. 설계 데이터 관리 모델

설계 데이터의 관리 모델은 한 *project* (설계관리를 위한 톱레벨의 디렉토리 계층을 의미) 내의 설계 팀원의 규모 및 설계자와 다른 설계자 간의 데이터 공유 혹은 보호 기준에 따라 설계 데이터 관리 모델은 정규 모델(formal model) 및 공유 모델(sharing model)로 분류할 수 있다. 그리고 정규 모델에서 공유 모델을 도입한 혼합 공유모델(mixed sharing model)이 있다.

정규 모델은 (그림 8)과 같으며, 설계자의 개인 작업영역에서 설계된 데이터가 상위 설계자(project administrator)에 의해 검증 및 릴리즈되어 다른 설계자가 그 데이터를 액세스하기까지 *submit, integrate,*



(그림 8) 설계 데이터 관리의 정규 모델

release, update 라고 하는 네 개의 명령어를 수행하는 사이클이 필요하다. 이 명령어에 대한 기능은 다음과 같다.

• *submit*

설계자가 설계 데이터를 상위 설계자에게 버전 생성 혹은 설계 변경을 요청하며, 이를 위해 IRs (Integration Requests)를 상위 설계자에게 전송한다. 설계자는 submit을 수행하기 전에 반드시 *check in* 연산을 수행해야 한다.

• *integrate*

설계자로부터 요청받은 설계 데이터들을 상위 설계자가 검증하여 설계 데이터의 형상에 포함시키기 위한 통합을 승인한다. 그러나 검증 결과가 불량이거나 다른 설계 데이터와 충돌이 되었다면 *reject* 명령어를 사용하여 통합 요청을 거부할 수 있다.

• *release*

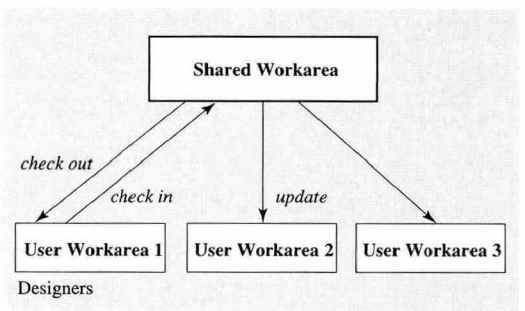
통합이 승인된 설계 데이터들은 검증이 완료되어 릴리즈 작업영역에서 IRs를 릴리즈함으로써 다른 설계자들의 데이터들과 함께 형상을 이루게 한다.

• *update*

한 *project* 디렉토리 내에서의 여러 설계자들의 개

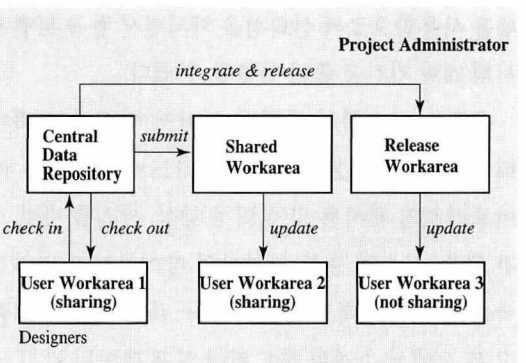
인 작업영역의 old 버전의 설계 데이터를 릴리즈된 최신 버전의 설계 데이터로 자동적으로 갱신한다.

공유 모델은 (그림 9)와 같이 설계 팀원 모두 데이터를 공유하는 모델이며, *check in*된 데이터들은 공유작업영역에 저장된다. 공유 모델의 모든 사용자 작업영역에서는 공유 작업영역의 데이터 내용을 볼 수 있다. 그리고 통합 요청이나 릴리즈를 위한 *submit*의 수행이 필요없다. 각각의 설계자들은 *release* 한가지만을 수행할 수 있다. 설계자가 버전 생성을 원하고, 다른 설계자의 변경되는 데이터들을 쉽게 접근하면서 설계자들이 독립적으로 책임과 권한을 가지고 설계를 하는데 유용하다. 또한, 설계 데이터들의 추적 및 탐사를 요하는 결합을 요구하지 않는 경우에 적합하다.



(그림 9) 설계 데이터 관리의 공유모델

혼합 공유 모델은 (그림 10)에 나타낸 것과 같이 정규 모델의 메커니즘을 가지며, 공유 모델을 혼합한 설계 관리 모델이다. 설계자들이 수정된 설계 데이터들을 차기의 *integrate*이나 *release* 과정을 거치고 싶지 않을 경우에 서로간 공유 작업영역에 연결하여 변경된 설계 데이터를 액세스할 수 있도록 한다. 설계자가 공유 작업영역에 연결할 때 공유 작업영역은 IR을 *submit* 하며, 공유 작업영역은 *submit*된 데이터들로 *update* 된다. 모든 설계자들은 공유 작업영역 내의 데이터들을 보고 싶을 때 *update* 연산을 사용할 수 있다. 만일 어느 사용자가 공유 작업영역 내에서 수정된 데이터들을 보고 싶지 않겠다고 하면 그 작업 영역을 공유 작업영역에서 분리시키고, 차기의 *release* 과정을 거칠 때까지 기다려야만 한다.



(그림 10) 설계 데이터 관리의 혼합 공유 모델

V. 결 론

본 고에서는 고속병렬컴퓨터 하드웨어 개발에서 적용한 하드웨어 설계 형상관리 및 라이브러리 개발을 위한 환경 중심으로 살펴보았다. 설계 데이터 관리의 기본 개념 및 CADPIA의 분산 하드웨어 설계환경, 고속병렬컴퓨터 하드웨어 설계를 위한 라이브러리 설계 개발에서의 하드웨어 설계 형상관리 적용 사례, 그리고 설계 데이터 관리 모델을 통해 설계 팀원들간의 라이브러리 공유 및 설계 관리의 기법을 알아보았다.

다수의 인원으로 구성되는 복잡한 설계의 검증이 요구되는 경우 정규 모델의 설계 관리가 행해진다. 이 모델은 상위 설계자의 검증을 통한 신뢰성을 고

려한 설계를 위해 적합하지만, 설계 기간의 단축효과는 좀 떨어질 수도 있다. 설계 팀원의 수가 아주 적은 경우 설계자들의 개인 작업영역에서 시험 및 검증을 거치는 경우에는 공유 모델이 유리할 수 있음을 알 수 있다. 그리고 설계 팀원의 수는 적지만 정규 모델의 메커니즘을 사용하면서 공유모델로의 전환이 용이한 동적 모델(dynamic model)로는 혼합 공유 모델이 적합함을 알 수 있었다. 그리고 혼합 공유 모델을 사용함으로써 신뢰성을 가지면서 정규 모델에서의 개발 기간을 좀더 단축할 수 있다.

대규모 시스템의 개발에 있어서 체계적인 데이터 관리 기법을 정립함으로써 하드웨어 설계용 라이브러리의 재사용 및 신뢰성 향상, 시스템 개발 기간 단축, 시스템 통합 시험시의 라이브러리 디버깅 극소화 등의 기대효과를 가질 수 있을 것이다. 그리고 본 고에서 소개된 하드웨어 설계 데이터 관리 모델은 하드웨어 설계 데이터의 저장이나 CAD 데이터베이스를 위한 배치 메커니즘에 이용될 수 있으리라 본다.

참고문헌

[1] Elisa Bertino and Lorenzo Martino, *Object-Oriented Database Systems, Concepts and Architectures*, Reading MA: Addison Wesley, 1993.

[2] Marina Zanella and Paolo Gubian, "A conceptual model for design management," *Computer Aided Design*, vol. 28, no. 1, pp. 33 - 49, 1996.

[3] Won Kim *et al.*, "Features of the ORION Object Oriented Database System," in *Object Oriented Databases, and Applications*, Kim, W. and Lochovsky, F., Eds., Reading MA: Addison Wesley, 1989. pp. 251 - 282.

[4] Chou, H. and Kim, W., "A unifying framework for version control in a CAD environment," in *Proc. Intl. Conf. on Very Large Data Bases*, Kyoto, Japan, Aug. 1986, pp. 336 - 346.

[5] Batory, D. and Kim, W., "Modeling concepts for VLSI CAD objects," *ACM Trans. on Databases Systems*, vol. 10, no. 2, pp. 322 - 346, Sep. 1985.

[6] Team Design Manager User Guide Version 4.0, Cadence Design Systems, Inc., Jun. 1995.

[7] Team Design Manager Training Manual Release 4.1, Educational Services Group (ESG) of Cadence Design Systems, Inc., Dec. 1995.

[8] G. W. Sloof *et al.*, "Design data management in a distributed hardware environment," *Proc. European Design Automation Conference*, IEEE Computer Society Press, pp. 34-38, 1990.

[9] 이재철 외 2인, "CADPIA: 고속병렬컴퓨터 개발을 위한 하드웨어 설계 기법 및 환경," 1995년도 하계 종합학술대회 논문집, 대한전자공학회, 제18권 제1호, pp. 364 - 367, 1995. 6.

[10] 이재철 외 2인, "CADPIA: 고속병렬컴퓨터 개발에서의 하드웨어 설계데이터 관리: 사례연구," 1995년도 추계 종합학술대회 논문집, 대한전자공학회, 제18권 제2호, pp. 1109 - 1112, 1995. 12.

[11] 이해진 외 5인, "CAD 데이터베이스를 위한 버전 관리 시스템의 설계 및 구현," 정보과학회 논문지, 정보과학회, 제21권 제3호, pp. 555 - 564, 1994. 3.