DEVS 형식론을 이용한 실시간 고속 대규모 데이터 저장 시스템의 설계

(Design of a Real Time, High Speed, Large Scale Data Storage System using the DEVS formalism)

국민대학교 전자공학부 컴퓨터공학 실험실
이찬수 성영락 오하령

요 약 본 연구에서는 대용량의 데이터를 고속으로 입출력할 수 있는 데이터 저장 시스템이 가져야 할 요구사항을 분석하고, 그것을 만족하는 시스템을 설계하였다. 본 논문에서는 우선 고속 대용량, 랜덤 억세스의 조건을 만족시키기 위해 여러 대의 하드 디스크를 병렬로 연결하여 입력되는 데이터들을 나누어 저장하도록 하였다. 그러나 하드 디스크의 성능은 디스크 아암의 탐색동작에 의해 크게 영향을 받으므로 실시간 요구 조건을 만족시키기 위해선 단순히 디스크의 수를 늘이는 것 외에 디스크 아암의 탐색 동작을 효율적으로 제어할 수 있는 방법이 필요하다. 그래서 본 논문에서 설계된 시스템에서는 시스템을 MCU(Master Control Unit), DDU(Data Distribution Unit), SCU(Slave Control Unit), DSU(Data Storage Unit)의 4부분으로 나누고, 각 디스크의 디스크 아암 탐색 동작을 독립된 SCU에서 제어하도록 하였다.

설계된 내용이 주어진 요구사항들을 만족하는 것을 확인하기 위해, 본 논문에서는 이산사건 시스템을 기술하는 수학적인 언어인 DEVS 형식론을 이용하여 제안된 시스템을 기술하고 시뮬레이션하였다. 그리고 시뮬레이션되는 과정에서 생산되는 사건들의 궤적을 분석하였다. 분석결과 제안된 시스템은 앞에서 제시한 여러 요구사항들을 잘 수용함을 보았다.

1. 서 론

최근들어 멀티미디어가 각광 받으면서 대용량의 데이터를 고속으로 입출력할 수 있는 데이터 저장 시스템에 대한 요구가 늘어가고 있다. 더욱이 초고속 정보 통신망, HDTV 방송 등과 연계되어 데이터의 크기가 점차 커짐에 따라 이런 요구 또한 점차 늘어날 전망이다.

본 연구에서는 그런 시스템들이 가져야 할 요구 사항을 분석하고, 그것을 바탕으로 그 요구사항을 만족하는 시스템을 설계하였다. 실시간 고속 대용량 저장 시스템은 i) 대규모의 데이터를 고속으로 입출력할 수 있어야 한다; ii) 데이터를 연속적으로 입출력할 수 있는 것 외에 비선형적으로 랜덤하게 억세스할 수 있어야 한다; iii) 주어진 입출력 속도를 항상 만족할 수 있도록 실시간 처리 기능을 가져야 한다. 본 논문에서는 우선 고속 대용량, 랜덤 억세스의 조건을 만족시키기 위해 여러 대의 하드 디스크를 병렬로 연결하여 입력되는 데이터들을 나누어 저장하도록 하였다. 그러나 하드 디스크의 성능은 디스크 아암의 탐색동작에 의해 크게 영향을 받으므로 실시간 요구 조건을 만족시키기 위해선 단순히 디스크의 수를 늘이는 것 외에 디스크 아암의 탐색 동작을 효율적으로 제어할 수 있는 방법이 필요하다. 그래서 본 논문에서 설계된 시스템에서는 시스템을 MCU (Master Control Unit), DDU(Data Distribution Unit), SCU(Slave Control Unit), DSU(Data Storage Unit)의 4부분으로 나누고, 각 디스크의 디스크 아암 탐색 동작을 독립된 SCU에서 제어하도록 하였다.

설계된 내용이 주어진 요구사항들을 만족하는 것을 확인하기 위해, 본 논문에서는 이산사건 시스템을 기술하는 수학적인 언어인 DEVS 형식론을 이용하여 제안된 시스템을 기술하고 이산사건 시뮬레이션 언어중의 하나인 DEVSim++를 이용하여 시뮬레이션 하였다. 그리고 시뮬레이션되는 과정에서 생산되는 사건들의 궤적을 분석하였다.

본 논문의 구성은, 2장에서는 DEVS 형식론과 DEVSim++에 대하여 알아보고, 3장에서는 시스템의 구성에 앞서 고려할 점들을 알아본 뒤 시스템의 하드웨어 부분과 운영 소프트웨어를 설계하였다. 4장에서는 구성한 시스템을 DEVS 형식론에 근거하여 모델링한 것을 나타내었고, 5장에서는 모델링한 시스템을 시뮬레이션한 결과를 나타내었으며, 6장에서는 결론을 서술하였다.

2. DEVS 형식론

형식론이란 분명한 방법으로 어떤 대상을 설명하는 것을 말하며, 이는 구성요소들의 집합과 그 구성요소들

의 조작으로 나누어 설명할 수 있다.

DEVS(Discrete Event System Specification) 형식론은 계층적이고 모듈화한 방법으로 이산사건 시스템의 모델들을 기술한 것이다. 즉, DEVS 형식론은 시스템을 작은 구성요소들로 나누어 모듈화된 형태로 모델링하고 그것들을 계층적으로 구성한다.

모델의 종류로는 atomic 모델과 coupled 모델이 있다. atomic 모델은 모델의 행위를 나타내는 반면 모델의 계층적인 구조를 나타내는 coupled 모델은 몇 개의 구성요소들을 결합하여 규모가 큰 모델로 확장시키는 역할을 한다.[4, 5, 6].

3. 데이터 저장 시스템의 설계

3.1. 시스템의 기능

구성하고자 하는 데이터 저장 시스템은 다음과 같은 기능을 만족하여야 한다. 첫째, 대규모의 데이터를 고속으로 입출력할 수 있어야 한다. 둘째, 데이터를 연속적으로 입출력할 수 있는 것 외에 비선형적으로 입출력할 수 있어야 한다. 셋째, 주어진 입출력 속도를 항상 만족할 수 있도록 실시간 처리 기능을 가져야 한다. 전체적인 시스템 구성도는 그림 3.1과 같다.

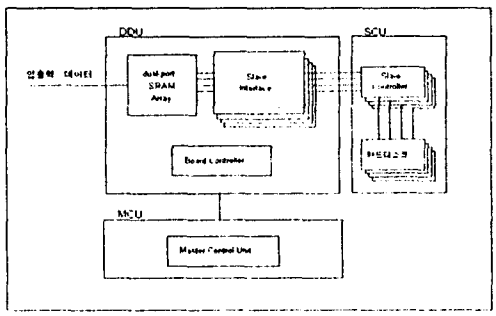


그림 3.1 시스템 구성도

3.2. 시스템 설계시 고려할 사항

본 절에서는 시스템을 구성하기에 앞서 고려해야 할 점들을 몇 가지 제시하고자 한다. 첫째로 데이터의 비선형적인 조작을 위해 저장 매체로 하드 디스크를 사용하기로 한다. 하드 디스크는 컴퓨터와의 연결 프로토콜에 따라 여러 가지로 분류할 수 있다. 일반적으로 널리 사용되는 프로토콜로는 IDE(Integrated Drive Electronics) 방식과 SCSI(Small Computer System Interface) 방식이 있다. 여기서 SCSI 방식 하드 디스크의 최대전송속도는 프로토콜에 따라 조금씩 다르나 일반적으로 5~40Mbyte/sec이며 하드 디스크의 처리 속도에 크게 영향을 미치는 파라메타 중의 하나인 평균 탐색시간은 8ms 정도로 IDE 방식의 하드 디스크 보다 빠르게 동작한다. 본 논문에서는 성능과 확장성을 고려하여 SCSI 방식의 하드 디스크로 시스템을 구성하기로 한다.[10]

둘째, 시스템은 n개 포트에서 입력이 있다고 가정하고 이들은 MCU(Master Control Unit)에서 각각을 동일한 방법으로 처리하되 독립적으로 관리하도록 한다.

셋째, 하드 디스크의 불규칙한 탐색동작 때문에 일반적으로 유효전송속도는 최대전송속도에 미치지 못한다. 이를 최대로 하기 위해서는 가능한한 데이터를 디스크의 연속적인 영역에 저장하여 디스크의 탐색동작을 최소화 하여야 한다. 이를 위해서는 기존의 파일 시스템에서와는 다른 방법으로 하드 디스크를 처리하기 위한 새로운 파일 시스템 구축이 필요하다.

넷째, 데이터를 하드 디스크의 연속적인 영역에 저장하더라도 디스크상에 불규칙하게 발생하는 배드섹터는 연속적인 영역을 깨뜨린다. 그러므로 배드섹터를 테이블로 만들어 디스크에 가지고 있다가 시스템 부팅시에 읽어와서 이를 바탕으로 배드섹터 부분을 제외한 연속적이고 논리적인 주소값을 만들어 사용한다. 또한 하나의 데이터 블록을 처리하기 위해서는 하나의 명령을 받아 처리해야 하며 이 때 하드 디스크는 탐색시간을 고려한 제한된 시간 내에서 이를 처리할 수 있어야 한다.

3.3. 시스템의 하드웨어 부분 설계

1) DSU(Data Storage Unit)

구성하고자 하는 시스템의 구성도를 그림 3.1에 나타내었다. DSU는 크게 MCU(Master Control Unit), DDU (Data Distribution Unit), SCU(Slave Control Unit)로 나눌수 있다. 여기서 SCU는 데이터가 분배되어지는 만큼 존재하며 본 논문에서는 4대의 SCU로 분배시킨다.

2) MCU(Master Control Unit)

MCU는 전체 시스템을 유지·관리하는 기능을 수행한다. 주요 기능으로는 사용자 인터페이스를 담당하고, SCU를 제어하며, 데이터 입출력을 위해 필요한 논리적인 어드레스를 한 데이터 블럭마다 발생해주며 여러 개의 프로그램의 저장과 재생을 위해 프로그램 카탈로그를 관리한다. 즉 사용자에게는 프로그램 단위의 데이터 관리 및 입출력을 제공하고 독립된 n개의 입력단을 위한 각각의 SCU에게는 블럭 단위의 관리를 제공한다.

3) DDU(Data Distribution Unit)

DDU는 dual-port SRAM 서브 시스템과 보드 콘트롤러로 구성되어 있다. dual-port SRAM 서브 시스템은 4개의 뱅크로 구성하여 입력데이터를 4대의 SCU로 각각 분배해 준다. 또한 4개의 SRAM으로 4중 미러링하여 데이터가 하드 디스크로 입출력될 때 발생될 수 있는 예외상황에도 데이터을 잃어버리지 않게한다. 또한 보드 콘트롤러는 dual-port SRAM의 어드레스 발생기능과 dual-port SRAM의 제어신호 발생기능, 그리고 MCU에

시 명령을 받아 SCU로 전달하는 기능을 수행한다.

4) SCU(Slave Control Unit)

저장시에는 DDU에서 데이터를 분배받아 하드 디스크에 저장하고, 재생시에는 저장되어 있는 데이터를 DDU로 보내는 역할을 한다. 이 때 각 블럭을 저장·재생하기 위해서 MCU로부터 명령을 전달받아 처리한다. 이 장치에서 관리되어야 할 정보들로는 논리적으로 연속적인 하드 디스크 주소값인 FBS(Field Block Set) 테이블이 있다.

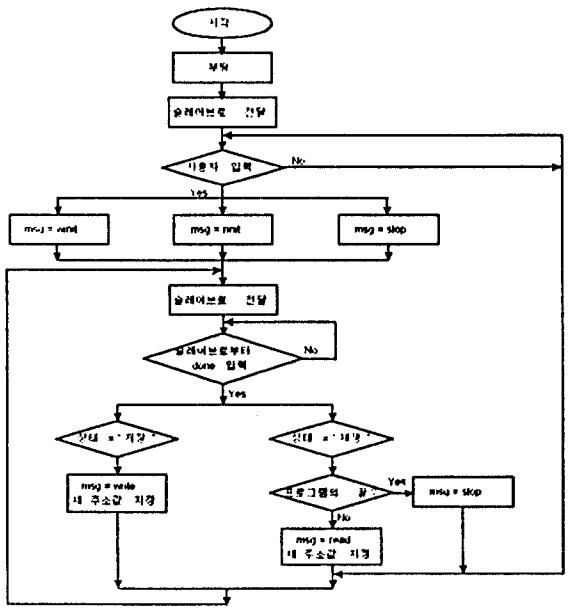
3.4. 운영 소프트웨어 시스템 설계

3.4.1 하드 디스크 정보 관리

입력되는 데이터에 비해 하드 디스크의 처리 속도가 느리므로 시간적 여유를 최대로 얻기 위해서는 한번의 탐색동작에 하나의 데이터 블럭을 처리할 수 있어야 하며 데이터들은 디스크상에 연속적으로 저장되어야 한다. 이것을 방해할 수 있는 것이 배드섹터의 발생인데 이것을 해결하기 위해서는 배드섹터 테이블(Bad Sector Table : BST)을 만들어 시스템 부팅시에 각각의 SCU에서 하드 디스크에 저장되어 있는 이 테이블을 메인 메모리로 읽어들이고 이를 바탕으로 하드 디스크의 연속적인 ID 블록의 집합인 FBS 테이블을 만들어 연속적인 논리적 주소값을 생성하게 된다.

3.4.2 소프트웨어 흐름도

1) MCU에서의 흐름도



위 흐름도를 살펴보도록 한다. 이기서 '슬레이브'는 SCU와 DDU를 MCU 나머지 부분을 뜻한다. 먼저 시스템 부팅시에 MCU는 부트 메시지를 슬레이브로 보낸다. 이후 적절하게 시스템이 부팅되면 사용자 입력으로 인한 명령이 발생하고 이에 따라 적절하게 SRAM의 외부·내부 지정 ID를 설정하기 위해 초기명령들이 발생된다. 이것이 슬레이브로 전달되고 적절하게 저장과 재생이 이루어진다. 이후 슬레이브로부터 완료메시지가 전달되면 계속하서 다음 데이터블럭을 저장하기 위한 메시지가 발생한다. 데이터의 끝 혹은 사용자의 정지메시지가 들어오면 수행을 멈춘다.

4. 시스템 모델링

4.1. 시스템 구성

본 논문에서는 구성한 시스템을 계층적이고 모듈화된 형태로 기술하는 DEVS 형식론으로 나타내기 위해 DEVSim++를 이용하였다. 이를 이용해 구성한 시스템을 시뮬레이션 하기에 앞서 그림 4.1에 나타내었듯이 3종류의 coupled 모델과 6종류의 atomic 모델로 나누었다.[9] 이 그림에서 각 모델의 왼쪽 상단에 표시가 붙어 있는 것은 atomic 모델을, 아무것도 붙어 있지 않은 것은 coupled 모델을 나타낸다

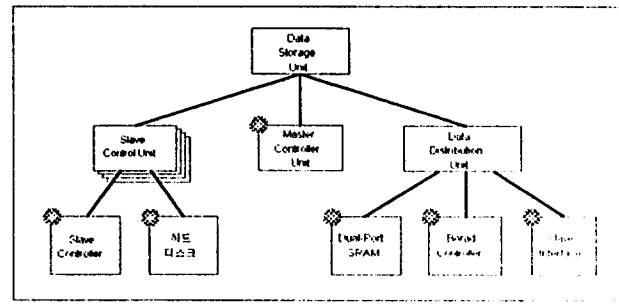


그림 4.1 시스템 모델의 계층도

4.2. 시스템 모델

시뮬레이션을 위한 각 모델들의 포트 연결들은 아래 그림 4.2과 같이 되어 있다. 여기서 atomic 모델 GENI는 데이터의 발생을 위해, GENC는 MCU에 가해지는 사용자 입력을 위해서, SCR은 저장되어 있던 데이터의 재생을 확인하기 위해 덧붙여진 모델들이다.

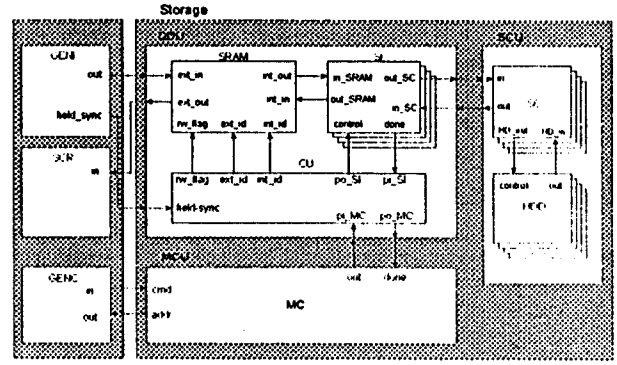


그림 4.2 모델 전체 BLOCK도

4.3. 각 부분별 모델링

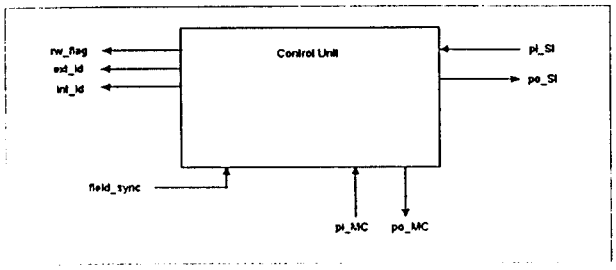


그림 4.3 콘트롤러 모델의 블록 다이어그램

데이터 분산 장치 내에 있는 보드 콘트롤러를 모델링한 것을 살펴보기로 하겠다. 보드 콘트롤러의 입력으로는 각 데이터 블록들의 싱크를 받아들이는 포트, MCU에서 명령을 받아들이는 포트, SCU에서 하드 디스크에 저장되어 있던 데이터를 받아들이는 포트가 있다. 출력으로는 dual-port SRAM의 내부 및 외부에서 사용할 뱅크를 지정해 주는 포트와 현재 동작 상태가 읽기상태인지 쓰기상태인지를 나타내 주기 위한 포트, SCU에게 명령을 보내기 위한 포트, MCU에게 완료 메시지를 보내기 위한 포트로 구성되어 있다. 개괄적인 기능은 MCU에서 명령을 받아 이에 적절한 메시지들을 SRAM과 슬레이브 인터페이스를 통해 SCU에게 전달하는 역할을 한다.

1) 외부 전이 함수

기본적으로는 보드 콘트롤러는 데이터의 입력에서 인에이블 싱크에 맞추어 외부 지정 SRAM ID를 증가시키기 위한 상태변수 eid를 갱신한다. 또한 퀀드 싱크가 들어왔다는 것을 알리기 위한 상태변수 sync_input을 하나 증가시킨 후 현재 phase가 WAIT 상태이면 단지 eid를 보내기 위한 SET_EID phase로, WAIT_SYNC이면 eid, iid, done 메시지를 보내기 위한 SEND_ALL phase로 가게 된다.

두 번째로 MCU로부터 입력이 들어왔을 경우를 살펴보자. 이 때에는 무조건 들어온 메시지를 상태변수 msg에 지장해 놓고 다음 과정을 처리한다. 지장된 msg의 명령어 부분이 winit이면 iid를 현재 상태의 eid보다 하나 적은 값으로 정한다. 이것은 데이터 지장시에 하드 디스크에서 예외상황의 발생으로 지장시간이 길어질 경우에도 이분의 SRAM에 데이터가 계속해서 적히질 수 있게 한다. 반면에 msg의 명령어 부분이 rinit이면 앞에서와는 반대로 iid를 현재 상태의 eid보다 하나 많은 값으로 정해 데이터 재생시에 하드 디스크에서 예외상황의 발생으로 읽는 시간이 길어질 경우에도 미리 지장되어 있는 이분의 SRAM에서 계속해서 데이터를 읽어 나갈 수 있게 한다. 이 과정 후에는 설정된 iid 값과 현재 읽기 상태인지 쓰기 상태인지를 나타내는 rw_flag를 보내기 위한 INIT phase로 가게 된다. 지장된 msg가 그 밖의 write, read, boot 인 경우에는 단지 슬레이브 인터페이스로 메시지를 전달해 주기 위한 SEND_MSG phase로 간다.

마지막으로 슬레이브 인터페이스로부터 입력이 들어왔을 경우를 보자. 이 입력은 보드 콘트롤러가 SCU 쪽으로 어떤 명령을 내리고 기다리다가 받는 완료 메시지일이다. 이 때 시스템 부팅 과정에서 생긴 완료 메시지를 받으면 MCU로 완료 메시지를 보내 주는 SEND_DONE phase로 바로 간다. 그 밖의 경우에서는 데이터를 각기 4대의 SCU에서 입출력하고 난 후의 완료 신호이므로 4번의 완료 신호를 받을 때까지는 기다리다 모두 받게 되면 MCU로 완료 메시지를 보내는 phase로 가게 된다.

2) 내부 전이 함수

내부 전이 함수는 외부 전이 함수에 비해 훨씬 간단하게 모델링 되었다. 각각의 phase가 SET_EID, SEND_DONE, SEND_ALL, WAIT인 경우에는 단순히 다음의 메시지를 받기까지 무한으로 기다리는 WAIT 상태로 가고, INIT, SEND_MSG 인 경우에는 슬레이브로 명령 메시지를 하나 보내게 됨으로 sync_input을 하나 줄이고 나서 WAIT phase로 가게 된다.

3) 출력 함수

각각의 출력함수는 phase에 따라 나뉜다. phase가 SET_EID인 경우에는 SRAM으로 eid값을 보내준다. INIT인 경우에는 SRAM으로는 rw_flag 값과 iid를, 슬레이브 인터페이스로는 msg를 보내준다. SEND_DONE phase에서는 SRAM으로는 iid를 MCU로는 완료를 나타내기 위한 빈 메시지를 보내준다. SEND_ALL phase에서는 SRAM으로는 eid와 iid를 보내주고, MCU에게는 완료 메시지를 보내주며, SEND_MSG phase에서는 슬레이브 인터페이스로 단지 msg만을 보내준다.

4) 시각 전진 함수

시각 전진 함수는 두 개로 나뉜다. 하나는 메시지를 보내는 시간으로 이것은 SRAM, 슬레이브 인터페이스, MCU로 메시지를 보낼 수 있는 시간중 제일 오래 걸리는 시간을 택했다. 나머지 하나는 외부에서 메시지를 받기 위해 무한대로 기다리는 시간이다.

5. 시뮬레이션

다음 그림들에서는 시뮬레이션 하는 동안에 각각의 atomic 모델에서 발생되는 내부 및 외부 전이 함수들이 발생하는 것을 표시하였다. 이기서, '↑'는 내부 전이 함수의 발생, '↓'는 외부 전이 함수의 발생을 나타내고 각각의 화살표 밑에는 그 전이 함수가 발생한 시간을 밀리초(ms) 단위로 표기하였다. 그림에서 슬레이브 콘트롤러(SC)와 하드 디스크(HDD)는 일반적으로 같은 시간

에 진이 함수가 발생하므로 4개 중 하나만 나타내었다.

1) 데이터 저장

그림 5.1은 일반적인 데이터 한 블록이 하드 디스크에 저장되는 과정을, 그림 5.2와 5.3은 데이터 저장시 하드 디스크의 예외상황 발생으로 저장이 지연되는 경우의 과정을 나타낸다.

일반적인 경우의 데이터 저장과정을 순서대로 살펴보면, a)GENI 보넷의 출력으로 데이터가 발생되고 b)이것이 DDU의 SRAM에 저장이 되며 CU의 sync단으로 입력이 된다. c)이 입력으로 CU는 SRAM의 외부 지정 ID를 증가시키기 위한 메시지를 발생한다. d)이 때 전 데이터의 처리완료 메시지가 MC에 전달되면 MC는 또다른 데이터를 위해 저장 메시지를 CU로 보내게 된다. e)그러면 CU는 MC로부터 받은 메시지를 SI로 전달한다. f)이 때 SI는 SRAM에게는 저장할 데이터의 요구 메시지를, SC로는 데이터의 저장을 준비시키는 메시지를 각각 보낸다. g)SRAM은 이에 대한 응답으로 현재 내부 지정 ID가 가르키는 곳의 SRAM의 데이터를 SI로 보낸다. h)SC는 SI로부터 메시지를 받아 하드 디스크로 디스크의 아암을 움직이기 위한 메시지를 보낸다. i)SRAM으로부터 데이터를 받은 SI는 데이터를 SC로 보낸후 j)HDD로부터 아암의 움직임이 완료되었다는 메시지를 SC가 받으면 SI로부터 받은 데이터를 HDD로 보낸다. k)HDD에서 데이터의 저장이 완료되면 SC로 저장완료 신호를 보내주고 l)SC는 SI로, SI는 CU로 완료 신호를 각각 보내준다. m)CU는 4대의 SCU로부터 완료 신호를 모두 받으면 SRAM으로는 새로운 내부 지정 ID 값을, MCU로는 완료 신호를 보낸다.

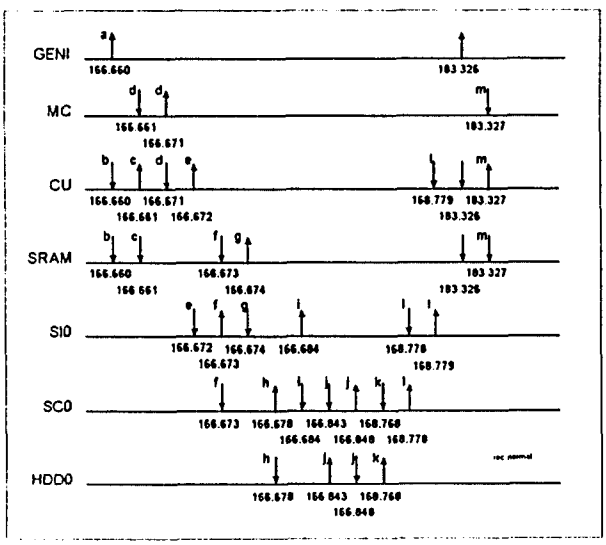


그림 5.1 시뮬레이션 결과-저장 : 일반

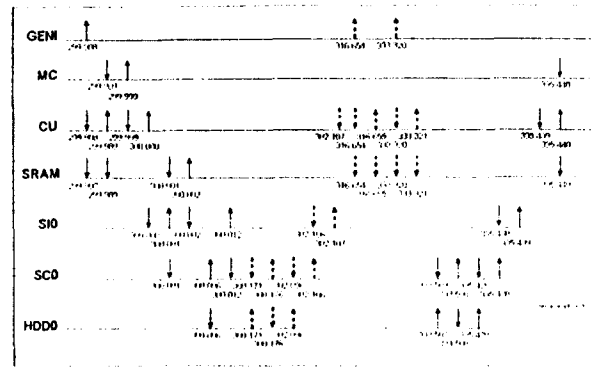


그림 5.2(a) 시뮬레이션 결과-저장 : 예외상황

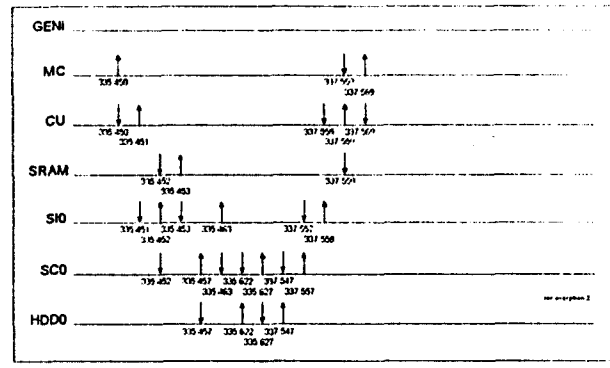


그림 5.2(b) 시뮬레이션 결과-저장 : 예외상황(계속)

2) 데이터 재생

그림 5.3은 일반적인 데이터 한 블록이 재생되는 과정을, 5.4는 저장시 하드 디스크의 예외상황 발생으로 재생이 지연되는 경우의 과정을 나타낸다.

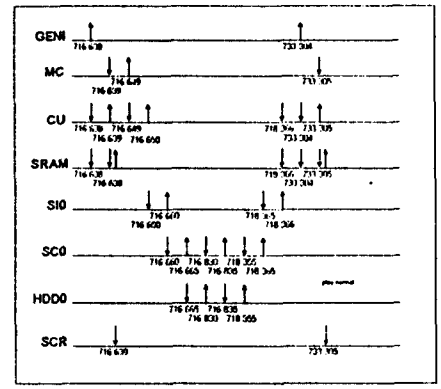


그림 5.3 시뮬레이션 결과-재생 : 일반

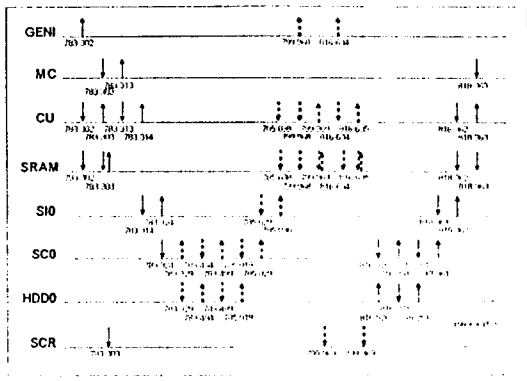


그림 5.4 시뮬레이션 결과-세션 : 예외상황

6. 결 론

본 논문에서는 대규모 데이터를 실시간으로 입출력하기 위한 저장장치를 설계하였다. 저장매체로는 하드 디스크를 이용하였는데 일반적인 방법으로는 대규모의 데이터를 실시간으로 저장할 수가 없어 입력되는 데이터를 여러 대의 하드 디스크에 나누어 저장하는 방법으로 구성하였다. 하드웨어 구성방법으로는 기본적으로는 dual-port SRAM을 이용하여 외부 디바이스 인터페이스와 하드 디스크 인터페이스가 독립적으로 데이터를 처리할 수 있도록 하고 시스템의 확장성과 신뢰도를 높이기 위해 여러 대의 PC를 이용하여 개발하였다. 또한 하드 디스크에 저장되는 데이터가 최소의 처리 시간을 갖게 하기 위해서 디스크의 연속적인 영역에 저장될 수 있게 하였고 이를 위해 BST와 FBS 테이블을 사용하여 데이터를 처리하는 방법을 사용하였다. 이렇게 시스템을 구성한 후에 시스템이 적절히 동작하는지 검증하기 위해 시스템의 각 부분을 DEVS 형식론을 이용하여 모델링하고 시뮬레이션 하였다. 그리고 시뮬레이션 되는 과정에서 발생하는 사건들의 궤적을 분석한 결과 제안한 시스템이 대규모 데이터를 위한 실시간 저장장치의 요구되는 요구조건을 잘 만족함을 보였다.

이 시스템은 입출력 포트의 수나 각 SCU의 하드 디스크의 수의 변경 등 약간의 시스템 구성 변경만으로도 새로운 데이터 저장 시스템의 요구 속도를 충분히 얻을 수 있게 확장할 수 있다.

[참고문헌]

[1] 남재일, 안치득, 정주홍, "영상부호화 기술 동향," 한국통신학회지 제11권 제8호 pp.23-36, 1994년 8월.

[2] 지원철, 박구현, 이상인, "통신 멀티미디어의 현황 및 전망," 한국통신학회지 제11권 제11호 pp.34-44, 1994년 11월.

[3] 이무호, "영상관련 국제 표준화 동향," 한국통신학회지 제11권 제8호 pp7-22, 1994년.

[4] Bernard P. Zeigler, Theory of Modelling and Simulation, John Wiley, 1976.

[5] Bernard P. Zeigler, Multifacetted Modelling and Discrete Event Simulation, Academic Press, 1984.

[6] 안명수, 박성봉, 김탁곤, "DEVSim++: 외미분에 기반한 이산사건 시스템의 객체지향 모델링 및 시뮬레이션 환경," 한국정보과학회논문지, 제 21 권, 제 9 호, pp. 1652-1664, 1994.

[7] A. I. Concepcion, Distributed Simulation on Multiprocessor: Specification, Design and Architecture, PhD thesis, Wayne State University, 1985.

[8] A. I. Concepcion and B. P. Zeigler, "DEVS Formalism: A framework for hierarchical model development," Transactions on Software Engineering, vol. 14, no. 2, pp. 228-241, Feb. 1988.

[9] Tag Gon Kim, "DEVSim++ User's Manual," KAIST, 1994.

[10] John M. Goodman, Hard Disk Secrets, IDE Books Wordwde Inc., 1993

[11] P. Venkat Rangan and H.M.Vin, "Efficient Storage Techniques for Digital Continuous Multimedia," IEEE Trans. Knowledge and Data Engineering, Vol. 5, No. 4, Aug. 1993, pp.564-573.

[12] 이석호, 화일처리론, 정익사, 1985