논문 2005-42CI-5-4

매엽식 세정장비의 동작순서 시뮬레이션 및 웨이퍼 처리량 측정에 관한 연구

(Study on Measurement of Wafer Processing Throughput and Sequence Simulation of SWP(Single Wafer Process) Cleaning Equipment)

선 복 근*, 한 광 록**

(Bok Keun Sun and Kwang Rok Han)

요 약

본 연구에서는 웨이퍼의 식각, 세정, 연마 공정에 사용되는 매엽식 세정장비의 동작순서의 시뮬레이션과 단위 시간당 처리량 측정 방법에 대해 연구한다. 유한상태기계를 바탕으로 스케쥴링 알고리즘에 따른 로봇의 상태를 정의하여 시뮬레이션 모델을 구축하였으며, 이에 따른 시뮬레이션 수행을 통해 세정장비의 시간당 처리량을 측정하였다. 본 연구에서 제시한 시뮬레이션 기법을 통해 레시피와 로봇의 동작속도에 따라 세정장비의 단위시간당 처리량을 측정하고, 처리량을 극대화 할 수 있는 레시피와 로봇의 동작순서를 찾아낼 수 있다.

Abstract

In this study, we study measurement of wafer processing throughput and sequence simulation of single wafer type for wafer cleaning equipments that were used for etching, cleaning and polishing of wafer. Based on finite state machine, simulation model was built with identification of robot's status according to scheduling algorithm. Moreover, through performance of simulation as above, throughput per hour of cleaning equipment was measured. By the simulation method that are proposed in this paper, we could measure the wafer throughput per hour according to recipe and robot motion speed, and find optimal recipe and moving sequence of robot that maximize the throughput.

Keywords: 시뮬레이션, 처리량 측정, 매엽식, 웨이퍼 세정 장비

I. 서 론

웨이퍼 세정기술의 목적은 표면의 오염제거이다. 반도체의 미세화와 고집적화가 급격하게 진행되면서 웨이퍼 표면의 오염이 반도체의 신뢰성이나 수율에 직접적인 영향을 미치게 되므로 미세한 파티클이나 금속오염의 제거가 제조공정의 중요한 문제로 작용한다. 반도체의 제조공정 자체가 오염발생 요인을 많이 내포하고 이러한 오염발생을 방지하기 위해 개별적인 세정공정이 필요하다. 개별적으로 다양한 세정공정이 요구되면서 세정장치도 다양한 방식으로 만들어지고 있으며 크게 웨이퍼처리 매수에 따라 웨이퍼를 배치로 처리하는 배치식(batch process)과 한 매씩 처리하는 매엽식(single wafer process)으로 구분한다[1][3][4].

배치식 세정장비는 웨이퍼를 이동시켜 여러 개의 약품에서 처리하는 다조식, 웨이퍼는 정지해 있으면서 여러 개의 약품으로 처리하는 단조식과 두 가지 방식을

* 정회원, ** 종신회원, 호서대학교 컴퓨터공학과
(Dept. of Computer Engineering, Hoseo University)
접수일자: 2005년5월31일, 수정완료일: 2005년9월6일

혼합한 하이브리드 방식으로 나눌 수 있으며 웨이퍼를 회전시키면서 약품을 분사하는 스프레이식도 있다. 웨이퍼를 한 장씩 처리하는 매엽식 세정장치의 경우 배치식 장비에 비해서 여러 가지 장점을 가지고 있다. 공정마다 새로운 약품의 공급이 가능하고 다른 웨이퍼의 영향을 받지 않으며 약품처리 후 웨이퍼가 이동하지 않고 회전하면서 세정을 하므로 약액의 치환성이 좋다[1][7].

300mm 이상의 차세대 반도체와 LCD, 유기 EL등의 차세대 디스플레이의 제조를 위한 매엽식 공정이 확산됨에 따라 최근의 제조장비는 서로 연관된 복수의 공정 모듈과 장비를 서로 연계시키고 로봇으로 이동을 자동화하는 복합제조장비가 확산되고 있다. 이와 같은 추세로 볼 때 향후 반도체공정의 90% 이상을 이러한 공정장비가 담당할 것으로 전망되고 있다[6][12]. 이러한 복합제조장비는 운영 및 제어가 복잡하여 고도의 스케쥴링 및 제어 알고리즘과 각종 운용 소프트웨어를 요구한다[5].

최근의 국내 장비업체들이 식각이나 박막증착공정 등에 대한 복합장비를 개발하여 출시하고 있으나 스케쥴링, 제어 및 운용소프트웨어 등의 기술개발은 취약한 상황이다. 장비별 또는 업체별로 다양한 장비가 제시되고, 웨이퍼의 공정흐름이 계속 바뀜에 따라 위와 같은 기술은 점점 더 복잡해지고 있으며, 장비업체에게는 제품의 완성도와 생산성 향상 및 상용화를 위한 필수기술이 되고 있다[2][3].

따라서 본 연구에서는 반도체 장비 제조회사와의 공동연구를 통해 매엽식 세정장비의 동작순서 시뮬레이션을 통한 웨이퍼의 처리량 측정 방법을 연구하며, 로봇의 동작속도와 웨이퍼 및 챔버의 레시피(recipe)를 외부입력 값으로 하여 입력 값에 따른 시간당 처리량 계산 및 통계정보 제공, 결과의 XML 문서화가 가능한 시뮬레이션 시스템(OSIMS, Optimized SIMulation System)을 개발한다.

본 논문의 구성은 Ⅱ장에서 관련연구로 매엽식 세정장비의 일반적인 구성과 동작방식에 대해 소개하고, Ⅲ장에서는 장비의 스케쥴링 기법과 자료구조에 대해 고찰한다. Ⅳ장에서는 스케쥴링 기법에 따르는 시뮬레이션 소프트웨어의 설계와 구현에 대해 논하며 Ⅴ장에서 단위시간당 웨이퍼 처리량 등 시스템의 평가와 스케쥴링 알고리즘의 개선점을 찾고자 한다. 마지막으로 Ⅵ장에서 결론과 향후과제에 대해 논한다.

Ⅱ. 관련연구

1. 매엽식 세정장비의 일반적 구성

반도체의 공정이 300mm 대구경 웨이퍼로 발전하가면서 수율의 증대와 품질의 향상을 위해서 이전의 카세트 단위의 배치식 공정에서 개별 웨이퍼 한 장씩 공정을 진행하는 매엽식 공정으로 기술이 확산되고 있다[6]. 이러한 매엽식 장비는 구성 모듈간 이송 거리를 줄이기 위해서 복수의 모듈을 원형으로 구성하고 복수 개의 웨이퍼 운송로봇을 통해 웨이퍼를 이동한다. 그림 1은 본 논문의 연구에 사용되는 세정장비의 개요도를 나타낸다.

장비는 크게 4개의 모듈로 구성되어 있다. 풉(FOUP : Front Opening Unified Pod)은 웨이퍼의 이동시 사용하는 용기라고 할 수 있다. 웨이퍼는 공정별 장비를 거치면서 처리되므로 장비 간 이동시 수평유지와 오염방지를 위해 풉과 같은 용기를 통해 이송되어진다. 하나의 풉에 20~25장의 웨이퍼가 담기게 되고 이러한 풉이 1~4개까지 세정장비에 연결되어 처리될 수 있다. 인덱서(indexer)는 웨이퍼를 운송하는 로봇이며, 처리해야 할 웨이퍼를 풉에서 버퍼(buffer)로 이송하거나 챔버(chamber)에서 처리를 완료한 웨이퍼를 버퍼에서 풉으로 이송한다. 버퍼는 인덱서와 이송로봇(TR : Transfer Robot)이 웨이퍼를 주고받는 인터페이스 역할을 한다. 이송로봇은 처리해야 할 웨이퍼를 버퍼에서 챔버로 이송하거나 챔버에서 처리를 완료한 웨이퍼를 버퍼로 이송하는 역할을 수행한다. 챔버는 정해진 레시피에 따라서 세정작업을 수행한다.

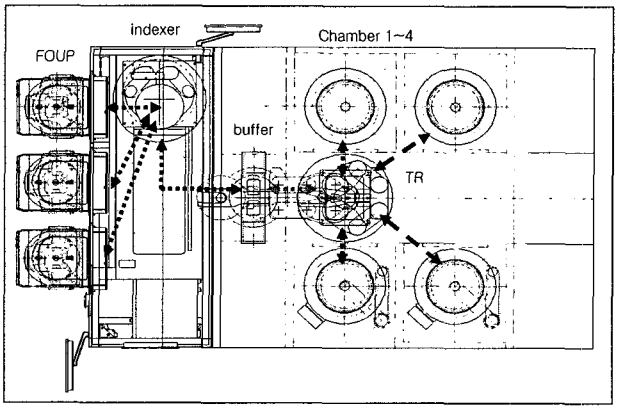

그림 1. 매엽식 세정장비 개요도
Fig. 1. Outline of SWP Cleaning Equipment.

2. 매엽식 세정장비의 동작방식

세정장비는 스케쥴링 및 제어를 담당하는 CTC(Cluster Tool Controller)에 의해 동작을 진행한다. 웨이퍼 진행 흐름의 관리, 웨이퍼 전송, 풉의 관리, 인덱서와 이송로봇의 제어, 챔버의 화학약품 제어 및 공유메모리와 프로세스간의 신호(signal) 관리 등 CTC는 세정장비의 두뇌에 해당한다고 할 수 있다.

장비를 제조하는 입장에서는 이송로봇의 동작속도를 변경하거나 레시피를 변경할 경우 등 장비에 다양한 변화를 주면서 장비의 생산성을 높여야 한다. 이때 생산성을 측정하기 위해서는 장비에 변화를 준 후에 실제로 테스트를 해보는 수밖에 없으며, 이는 비용의 상승과 비효율의 원인이 된다. 이러한 관점에서 스케쥴링 알고리즘과 동일하게 동작하는 시뮬레이션 프로그램의 개발이 요구되므로, 본 논문에서는 단위시간당 웨이퍼 생산량 측정이 가능한 시뮬레이션 방법을 연구한다.

Ⅲ. 스케쥴링 기법과 자료구조

3.1 스케쥴링 기법

본 연구에서 사용된 장비는 제 Ⅱ장의 그림 1과 같이 풉, 인덱서 등의 다양한 모듈과 로봇으로 구성되어 있다. 각각의 모듈과 로봇은 상태에 따라 신호를 발생시키고, 신호를 받아 상태를 변화하거나 동작을 수행한다. 이렇게 각 모듈과 로봇에서 만들어진 신호는 스케쥴러에 의해 구조체로 변환되어 처리된다.

스케쥴러는 제어해야 할 모듈과 로봇의 구성에 따라 테스크(task)를 생성하고 각 테스크에 신호를 전달하면서 장비를 제어한다. 테스크는 쓰레드(thread)로 생성되고 동작한다[8]. 그림 2는 스케쥴링 시 생성되는 테스크와 테스크간의 신호 및 데이터 통신에 대한 예를 나타낸다.

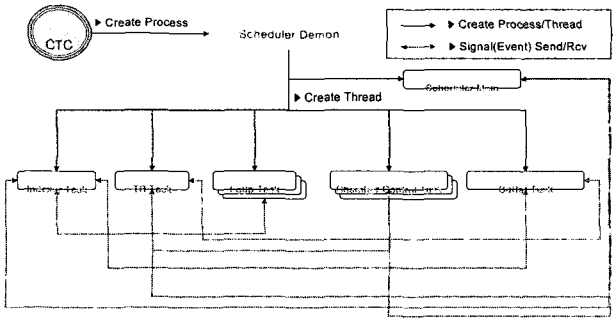

그림 2. 스케쥴러의 테스크 생성과 신호 전달
Fig. 2. Task creation and signal processing of scheduler.

그림 2는 스케쥴러의 핵심 테스크만을 개조식으로 표현한 것이며, 실제 스케쥴러는 로그 데이터 처리를 위한 DB 테스크 등 보조 테스크를 여러 개 더 생성한다. 데몬(Scheduler Demon)은 시스템의 구성 및 각종 테스크를 생성하며, 메인 테스크(Scheduler Main)는 웨이퍼의 흐름 관리를 수행하고 주요 테스크와의 신호 통신을 통해 명령을 전달한다. 인덱서 테스크는 인덱서 로봇의 웨이퍼 흐름관리를 수행하고 풉 및 버퍼와의 신호 통신을 통해 웨이퍼 투입을 관리한다. 이송로봇 테스크는 이송로봇 모듈의 제어와 웨이퍼 흐름 관리를 수행하며, 버퍼 및 챔버와의 신호 통신을 통해 챔버로의 웨이퍼 투입을 관리한다. 풉 테스크, 챔버 테스크, 버퍼 테스크 역시 각 모듈의 제어를 위해 모듈의 개수만큼 테스크가 생성되며 인덱서 로봇과 이송로봇의 사이에서 웨이퍼 흐름을 제어하면서 각 모듈의 고유기능을 수행한다.

스케쥴러는 장비에서 발생하는 제어신호를 구조체로 만들어 시그널 큐에 삽입하고, 각 모듈별 테스크는 큐에서 자신에게 해당하는 구조체를 추출하여 해당 작업을 수행한다. 또한 모듈간 데이터의 관리와 전달을 위해 공유메모리 기법을 사용하여 메모리를 관리한다.

3.2 자료구조

가. 공유메모리

스케쥴러는 각 장비에서 사용하는 데이터를 관리하기 위해서 공유메모리 기법을 사용하며, 시그널 큐를 사용하여 신호를 전달한다. 그림 3은 스케쥴러에서 사용하는 공유메모리 구조를 단순화 하여 표현한 것이다.

그림 3에서 나타낸 것과 같이 스케쥴러는 장비신호에 해당하는 Signal, 동작 시 사용하는 Run, 정적데

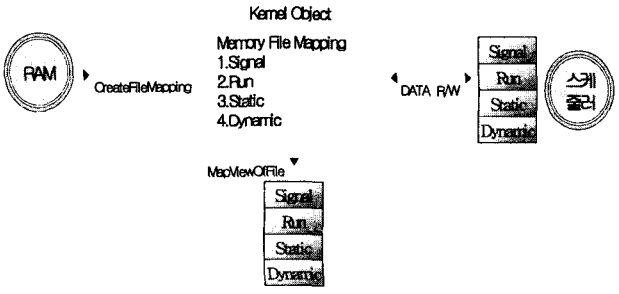

그림 3. 공유메모리의 구조
Fig. 3. Shared memory structure.

이터 관리를 위한 Static, 동적 메모리 관리를 위한 Dynamic 자료를 처리하며, 이는 CreateFileMapping, MapViewOfFile의 프로그래밍 과정을 거쳐 메모리맵파일로 만들어져 관리된다. 스케쥴러는 이러한 메모리맵파일을 통해 여러 모듈과 로봇의 신호 및 데이터 처리를 수행한다[13].

나. 순환형 시그널 큐

각 모듈 및 로봇에서 발생한 시그널은 Signal 구조체 변환되어 시그널 큐에 저장된다. 이때 전달되는 데이터는 신호의 수신자와 발신자 정보, 해당 신호 이름, 신호에 해당하는 각종 정보가 담긴 구조체이다. 큐의 구조는 그림 4와 같다.

모든 신호는 그림 4의 시그널 원형 큐에서 관리하며, 256개의 신호를 저장할 수 있다. 장비에서 신호가 발생하게 되면 SendSignalCode 함수를 통해서 Signal 구조체가 큐에 푸쉬 된다. 스케쥴러의 모든 테스크는 GetSignalCode 함수를 통해서 큐의 Front에 있는 Signal 구조체를 검사한다, Signal 구조체의 수신자에 해당하는 테스크는 RecvSignal 함수를 통해서 시그널을 가져오고 구조체는 큐에서 팝 된다.

본 논문에서 연구한 장비의 스케쥴러는 그림 3,그림 4와 같은 공유메모리와 큐를 사용하여 웨이퍼의 흐름을 관리하고 로봇 및 여러 모듈을 제어한다. 다음 절에서 웨이퍼의 흐름제어와 로봇 및 각 모듈의 신호에 따른 동작 방식을 살펴보기로 하자.

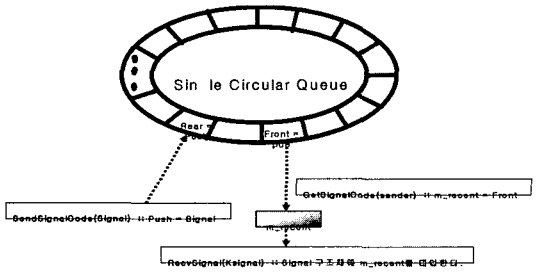

그림 4. 시그널 큐의 구조
Fig. 4. Signal queue structure.

3.3 웨이퍼 흐름 제어

장비가 처리하기 위한 웨이퍼는 풉에 위치한다. 본 논문에서 연구한 장비는 매엽식 세정장비이므로 챔버에서 웨이퍼를 한 장씩 처리한다. 장비를 통해 웨이퍼가 처리되는 과정은 그림 5와 같다.

풉에 위치한 웨이퍼는 인덱서, 버퍼, 이송로봇, 챔버

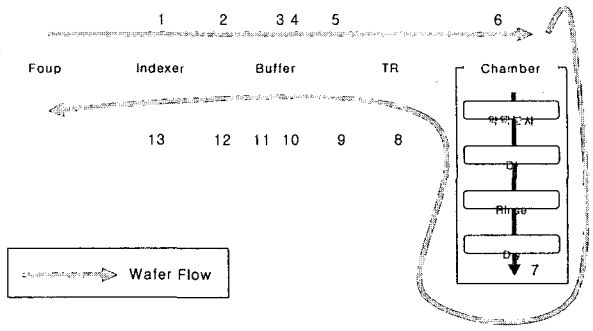

그림 5. 장비에서 처리하는 웨이퍼의 흐름
Fig. 5. Flow of wafers processed by equipment.

표 1. 장비의 동작순서
Table 1. Motion sequence of equipment.

번호	장비의 동작
1	인덱서가 웨이퍼를 풉에서 선택
2	인덱서가 웨이퍼를 버퍼로 이송
3	인덱서가 웨이퍼를 버퍼에 놓음
4	이송로봇이 웨이퍼를 버퍼에서 선택
5	이송로봇이 웨이퍼를 챔버로 이송
6	이송로봇이 웨이퍼를 챔버에 놓음
7	웨이퍼의 화학처리
8	이송로봇이 처리완료 웨이퍼를 선택
9	이송로봇이 처리완료 웨이퍼를 버퍼로 이송
10	이송로봇이 처리완료 웨이퍼를 버퍼에 놓음
11	인덱서가 처리완료 웨이퍼를 선택
12	인덱서가 처리완료 웨이퍼를 이송
13	인덱서가 처리완료 웨이퍼를 풉에 놓음

에서의 화학처리를 거쳐 다시 역순으로 이송되어지고 원래 위치해 있던 풉으로 돌아오는 것으로써 흐름을 마친다. 그림 5의 숫자에 따른 장비의 동작을 표 1에 나타낸다.

그림 5와 표 1을 참조하면 장비에서 웨이퍼의 흐름과 이에 따른 장비의 동작을 큰 흐름에서 관찰할 수 있다. 장비의 각 모듈 및 로봇의 동작은 자신과 관계있는 다른 장치와의 시그널 교환을 통해서 이루어지며, 이러한 시그널은 스케쥴러에 의해 관리된다. 예를 들어 위 표에서 1번 과정을 수행하기 전에 인덱서의 동작을 제어하는 인덱서 테스크는 스케쥴러에게 버퍼의 상태를 알려 달라는 시그널을 보내게 된다. 만일 인덱서가 버퍼에 웨이퍼를 내려놓을 수 없는 상황일 경우, 인덱서가 웨이퍼를 집어 오면 안 되기 때문이다. 마찬가지로 이송로봇이 4번 과정을 수행하기 위해서는 이송로봇 테스크가 버퍼에 웨이퍼가 존재하는지, 화학처리를 위한 챔버가 유휴상태에 있는지등의 상황을 알아야 하므로 시

그널을 보내고 그 결과를 알아낸다.

위에서 예로 설명한 상황외의 필요한 모든 사항을 고려하여 스케줄러가 설계되었으며, 본 논문에서는 이러한 상황을 고려한 시뮬레이터 개발에 대해 연구한다.

Ⅳ. OSIMS의 설계와 구현

본 논문에서는 장비의 스케쥴링 시뮬레이션 소프트웨어를 만들기 위해서 먼저 각 모듈과 로봇에서 발생하는 신호에 따르는 장비의 동작과 상태를 정의하고 이를 모델링 한다. 또한 레시피에 따라 틀려지는 웨이퍼의 흐름과 로봇의 동작시간에 따라 달라지는 처리량의 측정을 위해서 이러한 값들을 외부변수로서 편집이 가능하게 한다[9][10].

각 모듈과 로봇의 동작과 상태정의를 위하여 유한 상태 오토마타를 바탕으로 장비를 모델링하고 카운터를 사용하여 카운터의 1 클럭마다 장비의 상태를 체크하고 3600 클럭을 1시간으로 하여 처리량을 측정하는 방법을 사용한다.

4.1 시뮬레이션 시스템 설계

시스템의 설계를 위해 장비의 모듈과 로봇을 객체화 하였으며, 이에 따르는 시스템의 UML 다이어그램을 그림 6에 나타낸다.

시스템을 살펴보면, 어플리케이션(Application Class)이 풉(Foup Class), 인덱서(Indexer Class), 버퍼(Buffer Class), 이송로봇(TransferRobot Class), 챔버(Chamber Class) 객체를 생성하고 시뮬레이션을 시작한다.

한 번의 클럭에 모든 장비의 상태를 체크하고 상태에 따라 정해진 시뮬레이션 동작을 수행하기 위해 TimerTask를 상속한 카운터(TickCounter Class)를 어플리케이션이 생성하며, 카운터 인터페이스(TickListenerInterface)를 통해 각 모듈에 클럭 이벤트가 전달된다.

각 모듈은 이벤트에 해당하는 자신의 정해진 동작을 수행하고 스케쥴 인터페이스(ScheduleInterface)를 통해서 어플리케이션과 통신을 수행한다. 또한 어플리케이션은 레시피 메니저(RecipeManager Class)를 생성하여 레시피를 생성, 편집, 수정하며, 이를 시뮬레이션에 적용한다[7].

각 모듈의 간단한 동작을 살펴보면, 먼저 풉은 시뮬

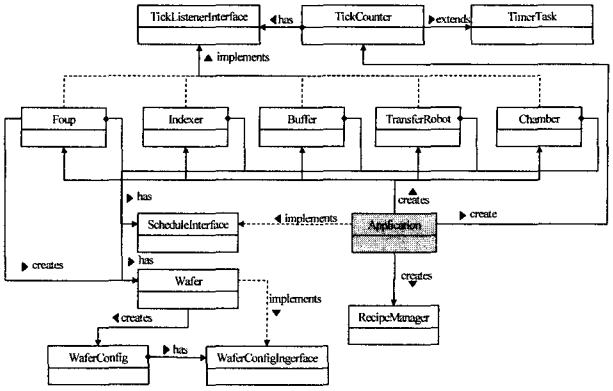

그림 6. 시뮬레이터 시스템의 클래스 다이어그램
Fig. 6. UML of simulation system.

레이션을 위한 웨이퍼(Wafer Class)를 생성하고 이때 생성된 웨이퍼는 인덱서, 버퍼, 이송로봇, 챔버를 통해 이동하면서 처리된다. 생성된 웨이퍼는 웨이퍼 설정(WaferConfig class) 모듈을 생성하여 웨이퍼의 각종 정보를 설정하며, 웨이퍼 설정 인터페이스(WaferConfigInterface Interface)를 통해 상호 정보교환을 수행한다.

다음절부터 상태 전이도(STD:Status Transition Diagram)를 사용하여 인덱서 로봇과 이송 로봇의 상태 정의와 설계를 위한 모델링에 대하여 기술한다. 상태 전이도를 통해 주요 모듈에서 감지하는 이벤트에 의해 변화하는 시스템의 동작을 나타내기 위하여 시스템의 제어를 나타내는 유한상태 오토마타를 확장하여 도식화 한다[11].

가. 인덱서 로봇

인덱서 로봇은 풉과 버퍼 사이에서 웨이퍼를 이송시키는 로봇이다. 그림 7과 같은 상태변화도에 따라 모델링한다.

인덱서 로봇은 그림 7과 같이 6개의 상태와 4개의 이벤트를 가진다. 초기화 후 대기상태에서 감지되는 이벤트에 따라 상태가 변화하면서 정해진 동작을 수행한다. 예를 들면, 풉에서 버퍼로 웨이퍼를 이송하는 상태인 FTB 상태에서는 이벤트를 통해 로봇의 이송시간(RTT)을 전달받고 이송시간이 0이 될 때까지 상태를 유지하며, 이송시간이 완료되면 역시 정해진 동작을 수행하고 상태를 CT상태로 변화시킨다. 이와 같은 동작을 통해 인덱서 로봇이 수행하는 웨이퍼의 이송에 대한 시뮬레이션 모듈을 설계한다. * 이벤트는 무조건 상태를 변화시키는 이벤트로써 현재상태에서 처리해야 할

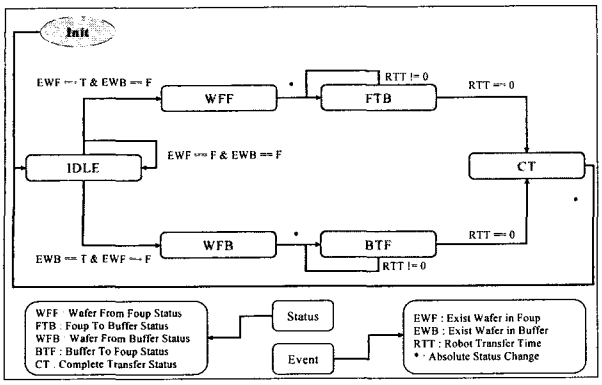

그림 7. 인덱서 로봇의 상태변화도
Fig. 7. STD of indexer robot.

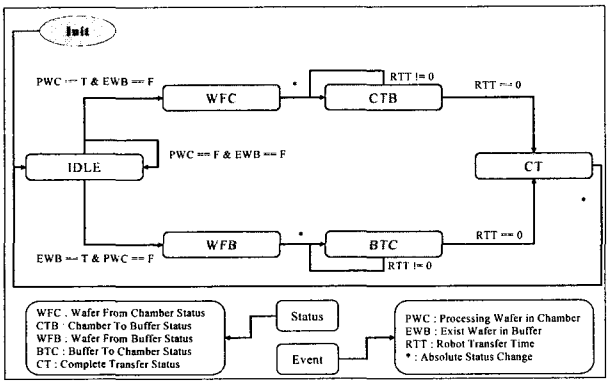

그림 8. 이송로봇의 상태변화도
Fig. 8. STD of transfer robot.

작업을 수행한 후 로봇의 상태를 바로 다음상태로 변경시킨다.

나. 이송로봇

이송로봇은 버퍼와 챔버 사이에서 웨이퍼를 이송시키는 로봇이며, 그림 8과 같은 상태변화도에 따라 모델링한다.

4.2 OSIMS 구현

OSIMS의 실행초기 메인화면은 그림 9와 같으며, 웨이퍼의 상태트리 부분, 로봇 및 화학시간 설정부분, 인덱서와 풉, 챔버와 이송로봇의 상황표시부분, 레시피 설정부분, 컨트롤 패널 부분, 현재 진행상황을 나타내는 부분으로 나뉜다.

웨이퍼 상태 트리는 로봇시간 및 화학시간 설정 시 사용자 선택화면이며, 시뮬레이션 수행시 웨이퍼의 정보를 볼 수 있는 화면이다. 로봇 및 화학시간 설정 버튼은 로봇의 시간과 챔버의 화학처리시간을 설정하는 부분이다. 인덱서 등의 상황표시 부분은 각 장비 및 로봇

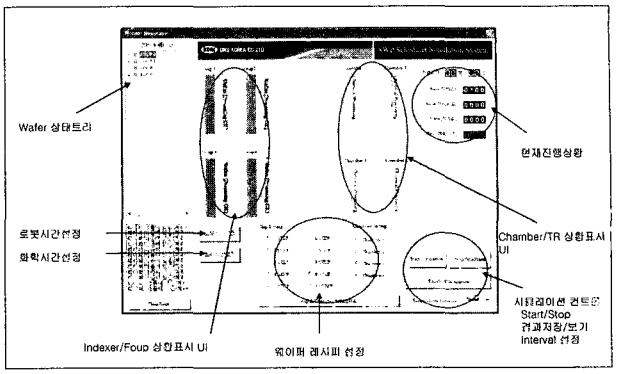

그림 9. OSIMS의 실행 화면
Fig. 9. Main screen of OSIMS.

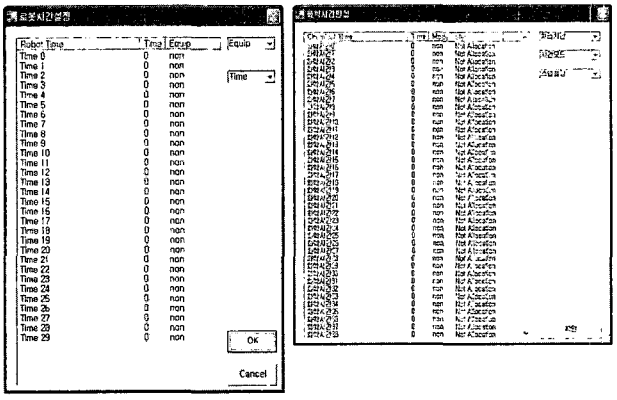

그림 10. 로봇시간 및 화학처리시간 설정화면
Fig. 10. Config screens of robot and chemical processing time.

의 현재 상황을 실시간으로 보여주는 부분이며, 현재진행상황 부분은 웨이퍼처리상황, 버퍼대기상황을 실시간으로 보여준다. 레시피설정 부분은 웨이퍼가 처리될 레시피를 설정하는 부분이며, 컨트롤 패널부분은 시뮬레이션의 시작과 종료, 시뮬레이션 속도설정 등의 시뮬레이션 제어를 담당하는 부분이다.

가. 외부 파라미터 설정

그림 10은 로봇처리시간과 화학처리시간을 설정하는 화면을 나타낸다. 로봇처리시간을 위한 파라미터는 30개이며, 운용자가 선택하여 편집이 가능하다.

이송로봇이 웨이퍼를 집어 올리는 동작은 여러 단계로 나뉠 수 있으므로 각 단계의 시간을 따로 설정하여 시뮬레이션이 가능하며, 화학처리시간을 위한 파라미터는 40개, 10개의 레시피로 웨이퍼에 설정되도록 구현하였다. 화학처리를 위한 파라미터를 입력한 후 메인화면의 웨이퍼 레시피 설정부분에서 각 웨이퍼의 레시피를 설정하고, 처리될 챔버를 설정하게 되면 설정된 레시피대로 웨이퍼가 이동하면서 시뮬레이션을 수행한다.

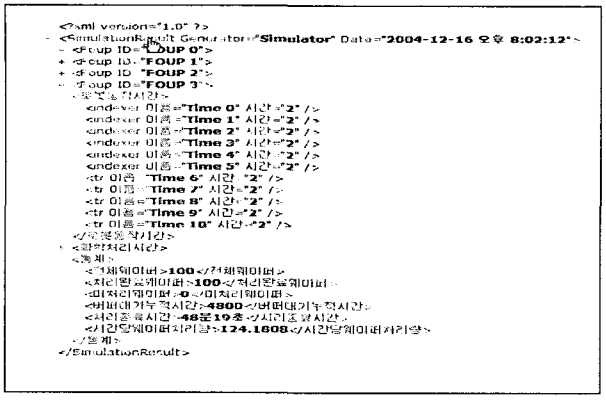

그림 11. 시뮬레이션 결과의 통계화면
Fig. 11. Simulation result screen example.

나. 통계 및 XML 문서화

OSIMS의 모든 설정을 마치고 시뮬레이션을 수행한 후, 통계 및 결과데이터는 XML문서로 저장된다. 그림 11은 시뮬레이션 결과 및 통계화면을 나타낸다.

그림 11을 살펴보면, 로봇동작시간으로 인덱서 로봇이 Time0~Time5라는 이름으로, 이송로봇이 Time6~Time10이라는 이름으로 각 2초씩 세부동작이 설정되어 있다. 또한 통계부분을 살펴보면, 전체 웨이퍼와 버퍼대기 누적시간, 처리종료시간 및 시간당 웨이퍼의 처리량이 계산되어 나타난다.

시뮬레이션 결과를 나타내는 XML 문서에는 시뮬레이션 일자, 웨이퍼 정보, 로봇의 동작시간, 챔버의 화학처리시간, 처리량 및 통계등의 정보가 들어간다. 장비의 운용자는 이 XML 문서를 통해 시뮬레이션 결과를 고찰하고, 외부 파라미터를 바꿔서 반복 테스트 해 봄으로써 장비의 성능을 개선할 수 있다.

4.3 OSIMS의 실행

그림 12는 실행되고 있는 OSIMS의 화면을 나타낸다. 그림 12의 진행상황 부분을 살펴보면 14분 8초의 시뮬레이션을 수행하여, 21장이 처리완료 되었고, 7장이 처리중 이며 처리중인 웨이퍼 중 3장이 버퍼에 대기 중임을 알 수 있다.

1번 풉의 웨이퍼 21장은 처리완료이며, 1번 풉의 웨이퍼 4장과 2번 풉의 웨이퍼3장이 현재 처리중이다. 처리중인 웨이퍼 7장중 3장의 웨이퍼는 챔버 1~3에서 화학처리중이며, 이송로봇이 이송하기위한 웨이퍼 한 장과 인덱서 로봇이 이송하기위한 웨이퍼 두장이 버퍼에 대기 중이다. 레시피 설정부분을 살펴보면, 현재 시뮬레이션 되고 있는 모든 웨이퍼는 1스텝의 화학처리를 챔

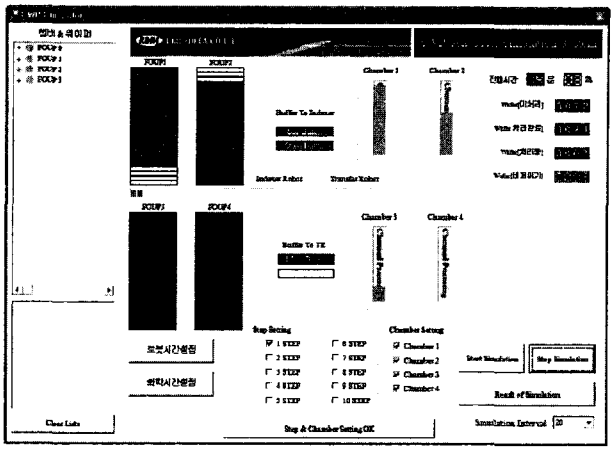

그림 12. OSIMS의 실행화면
Fig. 12. Run-time screen example of OSIMS.

버1~챔버4에서 수행되고 있는 것을 알 수 있다. [그림 12]에서 나타내는 시뮬레이션은 한 클럭이 20밀리초 마다 발생되며, 1시간의 시뮬레이션 수행을 위해서는 72초의 시간이 걸린다. OSIMS는 5밀리초 ~ 2초 마다 클럭을 발생시킬 수 있다.

V. 실험 및 고찰

OSIMS의 성능평가 및 고찰은 정확도, 파라미터 및 레시피 변경을 통한 시간당 웨이퍼 처리량 측정을 통해 수행한다. 성능평가 환경은 윈도우즈 XP Professional, 펜티엄 4 2.4GHz, 1GB RAM이며, OSIMS는 C#을 통해 개발되었으며, 닷넷 플랫폼에서 동작한다.

먼저 현재 장비의 로봇이송속도 및 화학처리시간과 동일하게 데이터를 입력하여 시뮬레이션을 수행함으로써 장비 운용 시 출력 데이터와의 일치성 검사를 통해 시뮬레이션 정확도 검사를 수행한다. 장비 및 OSIMS의 설정값은 표 2에, 처리량의 측정값과 오차율은 표 3에 나타낸다.

로봇의 이송시간 및 화학처리시간은 세부적으로 나

표 2. 장비와 OSIMS 설정값
Table 2. Configuration value of equipment and OSIMS.

설정항목		처리시간 (초)
로봇이송시간	인덱서 로봇	7
	이송로봇	8
화학처리시간	시작시간	6
	종료시간	6
	처리시간	81
웨이퍼 레시피	모든 웨이퍼는 챔버1~4에서 처리	

표 3. 처리량 및 오차율
Talbe 3. Throughput and error rate of simulation.

시간당 웨이퍼 처리량	
장비 처리량	120장
OSIMS 처리량	124장
오차율	3.3%

표 4. 파라미터 설정 및 웨이퍼 레시피 설정 (단위:초)
Table 4. Parameter and wafer recipe configuration. (unit:sec)

Round	로봇이송시간		화학처리시간			웨이퍼 레시피
	I	T	S	E	P	
1	6	7	6	6	81	Recipe1
2	7	7	6	6	81	Recipe2
3	7	8	6	6	81	Recipe3
4	7	8	5	5	80	Recipe4
5	5	5	5	5	75	Recipe5
6	7	8	6	6	80,85	Recipe6
7	7	8	6	6	80,85	Recipe7
8	7	8	6	6	75,90	Recipe8
9	7	8	6	6	75,80, 85,95	Recipe9
10	5	5	5	5	75,75	Recipe10

타낼 경우 많은 항목이 존재하며, 표 2에 나타나는 설정값은 세부 설정값을 통합하여 나타낸 것이다. 표 3에서 나타나는 웨이퍼 처리량의 오차는 장비 스케쥴러의 시간 복잡도(time complexity)가 OSIMS보다 상대적으로 크기 때문에 발생되는 소프트웨어적 오차로 판단되며, OSIMS의 출력 값에 이 오차율을 적용시켜야 한다.

두 번째로 파라미터 및 레시피의 변경을 통한 시간당 웨이퍼 처리량의 측정을 수행한다. 본 연구의 목적이 장비의 처리량을 소프트웨어적으로 예측하는 것이므로, 이 부분의 시뮬레이션 결과가 장비를 만들어 내고 운용하는 업체 측면에서는 매우 중요하다고 할 수 있다.

표 4는 실험을 위한 파라미터 및 레시피의 설정값을 나타내며, I는 Indexer, T는 TR의 이동속도를 나타내며, S는 시작시간, E는 종료시간, P는 화학처리 시간을 나타낸다.

그림 13은 표 4의 설정값에 따라 OSIMS의 시뮬레이션 후 측정된 웨이퍼의 처리량을 나타내며, 그림 14는 버퍼에서 대기하는 웨이퍼의 시간을 누적하여 나타낸다. 처리도중 버퍼에서 병목현상이 발생할 경우 웨이퍼가 버퍼에서 대기하며, 이 대기시간을 감소시킬 경우 웨이퍼의 처리량은 늘어나게 된다.

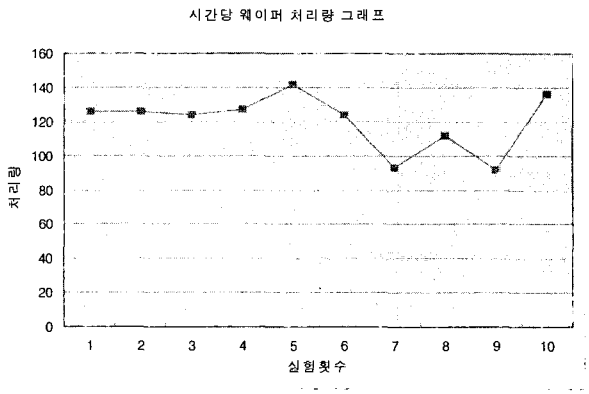

그림 13. 시간당 웨이퍼 처리량
Fig. 13. Simulation result of wafer throughput per hour.

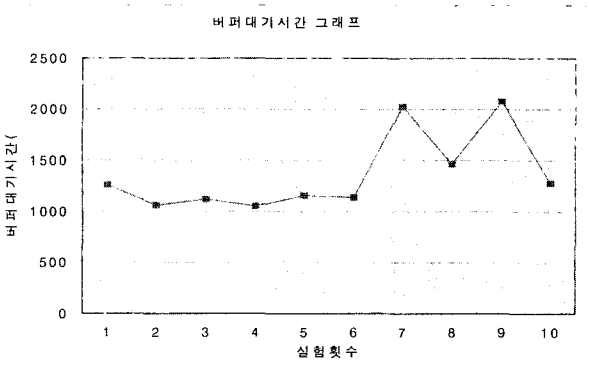

그림 14. 웨이퍼의 버퍼대기시간
Fig. 14. Buffer wait time of wafer.

인덱서 및 이송로봇의 움직임 속도 향상에는 한계가 있으며, 그 한계 내에서 로봇의 이송속도를 단축시켜도 처리량에는 크게 영향을 미치지 않는다. 그러나 화학처리시간을 단축시키거나, 웨이퍼 처리 레시피를 변경시킬 경우, 시간당 웨이퍼 처리량에 큰 변화가 일어나는 것이 표 4와 그림 13을 통해 나타난다. 또한 그림 13과 그림 14에서 나타난바 와 같이 시간당 웨이퍼 처리량과 버퍼대기시간은 정확하게 반비례한다. 로봇의 이송속도 변화가 웨이퍼 처리량에 영향을 크게 못 미치는 이유는 로봇의 이송속도에 비하여 웨이퍼의 화학처리시간이 상대적으로 매우 크기 때문이다.

따라서 장비의 처리량 증가를 위해서는 버퍼대기시간의 감소와 웨이퍼 처리를 위한 레시피의 최적화가 필요하며, 버퍼의 대기시간을 감소시키기 위하여 버퍼를 2단 또는 3단으로 제작하는 것을 고려해 볼 필요성이 있다.

VI. 결 론

300mm 이상의 차세대 반도체, LCD, 유기 EL등의 차세대 디스플레이의 제조를 위한 매엽식 공정이 확산됨에 따라 최근의 제조장비는 서로 연관된 복수의 공정 모듈과 장비를 서로 연계시키고 로봇으로 이동을 자동화하는 복합제조장비가 확산되고 있으며, 향후 반도체 공정의 90% 이상을 이러한 공정장비가 담당할 것으로 전망되고 있다. 이러한 복합제조장비는 운영 및 제어가 복잡하여 고도의 스케쥴링 및 제어 알고리즘과 각종 운용 소프트웨어를 요구한다.

본 연구에서 매엽식 세정장비의 동작순서 시뮬레이션을 통한 웨이퍼의 처리량 측정 방법을 연구하였으며, 로봇의 동작속도와 웨이퍼 및 챔버의 레시피를 외부 입력 값으로 하여 입력 값에 따른 시간당 처리량 계산 및 통계정보 제공, 결과의 XML 문서화가 가능한 시뮬레이션 시스템인 OSIMS를 개발하였다.

실험결과 웨이퍼의 레시피를 최적화 하고, 버퍼의 대기시간을 감소시키기 위하여 2단 또는 3단으로 버퍼를 구성하는 방안을 제시하였다. 이를 해결하려면 스케쥴링 알고리즘의 변화와 함께 최적화된 레시피를 찾기 위해 반복되는 시뮬레이션이 필요하며, OSIMS 또한 변화하는 스케쥴링 알고리즘에 맞게 개선해야 할 것이다.

참 고 문 헌

[1] 박영춘, "반도체 세정공정 및 장치기술 동향", 반도체산업, pp. 51-60, 2003.

[2] 박영균, "테스트 핸들러의 기술적 동향", 한국반도체산업협회 기술동향 분석보고서, 2001.

[3] 박헌휘, 이춘수, 최승우, 함승주, "반도체 공정에서의 기상 세정장비 개발에 관한 연구", 한국산학기술학회 춘계학술대회 발표논문집, pp. 79-81, 2001.

[4] 이치웅, "가스 용해기능수에 의한 반도체 웨이퍼 세정기술", 고압가스, pp. 292-296, 2004.

[5] 전자자료사, "시뮬레이션 기반의 반도체 제조공정", 반도체, 1227-741X,(5), pp. 93-98, 2004.

[6] 최장섭, "Track 장비 산업의 발전과 동향", 73-81, 반도체산업, pp. 73-81, 2003.

[7] E. Gamma, R. Helm, R. Johnson, J. Vlissides, "Design Patterns", Addison Wesley, pp. 345-367, 2003.

[8] G. Shepherd, Scot Wingo, "MFC Internals", Addison Wesley, pp. 178-190, 2004.

[9] Lim, Low, Gan, Cai, "Implementation Lessons of Performance Prediction Tool for Parrallel Conservative Simulation(Research Note)", LNCS, vol. 1900, pp. 189-201, 2001.

[10] LU S. H, P.R. Kumar, "Distributed scheduling based on due dates and buffer priorities", IEEE Transactions on Automatic Control, 36, 1406-1416, 1991.

[11] Peter Linz, "An Introduction to Formal Languages and Automata", Heath, pp. 37-71, 2002.

[12] 石井宏明, "モジュール 構成に 依る 枚葉式 洗浄装置", クリーンテクノロジー, vol. 11, no. 8, pp. 32-33, 2003.

[13] "http://msdn.microsoft.com/library" Microsoft

저 자 소 개

선 복 근(정회원)
1999년 호서대학교 컴퓨터공학과 졸업(학사)
2001년 호서대학교 벤처전문대학원 컴퓨터응용기술학과 졸업(공학석사)
2003년～현재 호서대학교 반도체제조장비국산화연구센터 전임연구원
2005년 현재 호서대학교 대학원 컴퓨터공학과 박사과정 재학
<주관심분야 : 정보검색, 에이전트시스템, 멀티미디어 동기화, HCI>

한 광 록(종신회원)
1984년 인하대학교 전자공학과 졸업(공학사)
1986년 인하대학교 대학원 정보공학 전공(공학석사)
1989년 인하대학교 대학원 정보공학 전공(공학박사)
1989년～1991년 한국체육과학원 선임연구원
1991년～현재 호서대학교 컴퓨터공학부 교수
2001년～2002년 ISI University of South California 방문연구원
<주관심분야 : 멀티미디어, 정보검색, 자연어처리, 기계번역, HCI, 지능형에이전트>