Journal of The Korea Society of Computer and Information
Vol. 18, No. 3, March 2013
www.ksci.re.kr
http://dx.doi.org/10.9708/jksci.2013.18.3.107

OES 센서를 이용한 반도체 식각 공정 모니터링 시스템 개발

김 상 철*

A Semiconductor Etching Process Monitoring System Development using OES Sensor

Sang-Chul Kim*

요 약

본 논문에서는 반도체 식각 공정 모니터링 시스템을 개발한다. 반도체 산업은 첨단 산업 중, 전자제품의 필수 부품을 생산하는 대표적인 고부가가치 산업으로, 세계 각국에서 치열한 개발 경쟁을 벌이고 있다. 이에 따라 반도체 제품의 품질과 특성, 그리고 생산성을 향상하기 위한 많은 연구들이 진행되고 있는데, 공정 모니터링 기술이 이에 해당한다. 실제로 반도체 회로를 형성하는 식각 공정에서의 불량은 큰 피해를 야기 시키므로, 공정을 상세히 모니터링 할 수 있는 시스템의 개발이 필요하다. 본 논문에서 기술하는 반도체 식각 공정 모니터링 시스템은 플라즈마를 이용한 건식식각 공정을 상세하게 관찰・분석하여 관리자에게 피드백하고, 설정된 시나리오에 맞게 자동으로 공정을 제어하여 공정 자동화 효율을 극대화한다. 실시간으로 모니터링을 수행하고 그 결과를 즉각적으로 시스템에 반영한다. 관리자는 시스템에서 제공하는 UI(User Interface)를 통해 공정의 현재 상태를 진단할 수 있다. 시스템은 관리자가 사전에 작성한 공정 시나리오를 따라 공정을 자동으로 제어하고, 공정중단 시점을 효율적으로 찾아내어 생산 효율을 높인다.

▸ Keywords : 반도체, 식각, 불량 검사, 플라즈마, OES 센서, 모니터링

Abstract

In this paper, we developed the semiconductor monitoring system for the etching process. Around the world, expert companies are competing fiercely since the semiconductor industry is a leading value-added industry that produces the essential components of electronic products. As a result, many researches have been conducted in order to improve the quality, productivity, and

• 제1저자, 교신저자 : 김상철
• 투고일 : 2012. 12. 29, 심사일 : 2013. 2. 11, 게재확정일 : 2013. 3. 9.
* 국민대학교 컴퓨터공학과(School of Computer Science, Kookmin University)
※본 연구 논문은 2012년도 지식경제부, 지식경제 R&D 전략기획단 미래산업선도기술개발사업(과제번호:10042421), 2013년도 국민대학교 교내연구지원금 및 중소기업청, 2011년도 산학연공동기술개발사업(No. 00045590)의 지원을 받아 수행되었습니다.

characteristics of semiconductor products. Process monitoring techniques has an important role to give an equivalent quality and productivity to produce semiconductor. In fact, since the etching process to form a semiconductor circuit causes great damage to the semiconductors, it is very necessary to develop a system for monitoring the process. The proposed monitoring system is mainly focused on the dry etching process using plasma and it provides the detailed observation, analysis and feedback to managers. It has the functionality of setting scenarios to match the process control automatically. In addition, it maximizes the efficiency of process automation. The result can be immediately reflected to the system since it performs real-time monitoring. UI (User Interface) provides managers with diagnosis of the current state in the process. The monitoring system has diverse functionalities to control the process according to the scenario written in advance, to stop the process efficiently and finally to increase production efficiency.

▸ Keywords : Semiconductor, Etching, Faulty Check, Plasma, Optical Emission Spectroscopy Sensor, Monitoring

I. 서 론

반도체 산업은 1950년대 중반부터 급격히 일어나기 시작한 우주 개발과 미사일 등 새 군용 기기의 발전을 시작으로 직결되어, 현대에 이르러서는 소형·경량의 고신뢰성 전자부품의 필요성으로부터 본격적인 마이크로일렉트로닉스(Microelectronics) 시대로 접어들게 되었고, 이에 따라 세계 각국에서 개발 경쟁을 치열하게 벌이고 있다.

특히 반도체 제조 공정에서 발생하는 불량을 사전에 방지하여, 불량으로 인한 피해를 줄이기 위해 공전중단점(End-point)을 감지하는 기술에 관심이 쏠리고 있다. 실제로 반도체 회로를 형성하는 식각 공정에서의 불량은 비용적으로 큰 피해를 야기 시킨다[1].

따라서 반도체 식각 공정을 실시간으로 관찰하고 정해진 시나리오에 의한 공정 관리·제어를 통해 불량을 사전에 방지하여 피해를 줄이고 공정 관리 자동화 효율을 극대화하는 모니터링 시스템을 기술한다[2].

본 논문의 구성은 2장에서 관련기술 및 사용 플랫폼, 3장에서 시스템 구조를 설명하고 4장에서 실험 결과를 보인다. 마지막으로 5장에서 결론과 향후 계획에 대해 기술한다.

II. 관련 연구

1. 반도체 식각 공정

반도체 식각 공정은 실리콘 원석을 가공하여 만든 웨이퍼를 식각하여 실질적인 반도체 회로를 형성하는 공정이다. 반도체 식각 공정은 습식식각(Wet Etching)과 건식식각(Dry Etching)이 있다[3].

습식식각은 반도체 산업 초기에 사용하던 방법으로, 식각하고자 하는 막과 화학적 반응을 일으키는 화학용액을 사용하여 식각하는 방법이다. 습식식각 공정은 화학반응이 수직 및 수평 방향으로 동시에 진행되기 때문에 [그림 2]와 같이 언더컷(Under Cut) 현상이 발생하게 되고, 가장 큰 단점으로 패턴의 선폭이 점점 좁아져 수직한 코어층의 형성을 어렵게 한다. 이러한 이유로 최근에는 플라즈마를 이용한 건식식각 공정을 주로 사용한다[4].

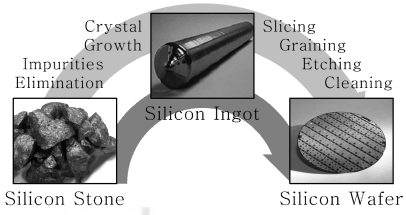


그림 1. 반도체 제조 과정
Fig. 1. Semiconductor manufacturing process

건식식각 공정은 플라즈마 내에 존재하는 원소와 표면 물질간의 화학반응, 그리고 플라즈마 내에 존재하는 활성종 입자의 표면 충돌에 기인한 반응촉진에 의해 진행된다. 먼저 웨이퍼에 회로 구성물질과 포토레지스트를 도포한다. 다음으로 회로의 패턴이 그려진 마스크를 씌워 빛에 노출시키면 빛에 노출된 포토레지스트가 제거되고 제거된 부분을 식각하여 회로를 구성하는 방법이다[5].

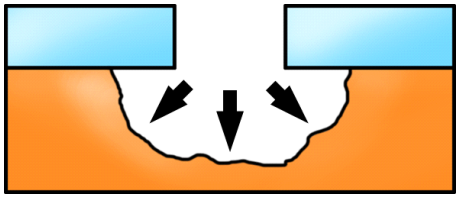


그림 2. 습식식각 모형
Fig. 2. Model of wet-etching

공정의 특성상 원자 단위로 제어를 할 수 있기 때문에 습식식각과는 달리 [그림 4]와 같이 원하는 모양으로 웨이퍼를 식각할 수 있는 장점이 있다[4].

(*포토레지스트 : 빛에 노출되면 불용성에서 가용성으로 혹은 가용성에서 불용성으로 성질이 바뀌는 물질)

2. 건식식각 공정

건식식각 공정에서 일어나는 플라즈마 반응은 상태가 전이될 때 에너지를 방출하는데, 플라즈마의 종류에 따라 방출되는 파장, 즉 빛의 색상이 다르다는 특징이 있다. [그림 5]는 건식식각 공정에서 플라즈마의 종류에 따라 염소 (Chlorine, Cl)의 발생 강도가 다른 것을 나타낸다[5].

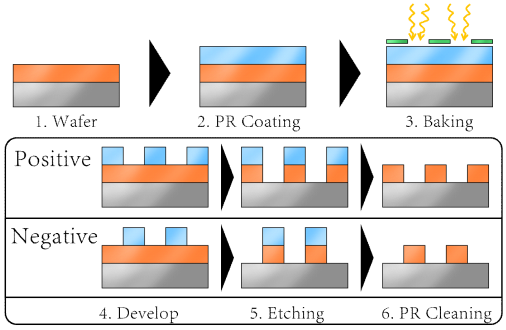


그림 3. 건식식각 과정
Fig. 3. Dray-etching process

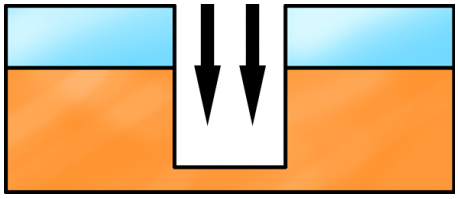


그림 4. 건식식각 모형
Fig. 4. Model of dry-etching

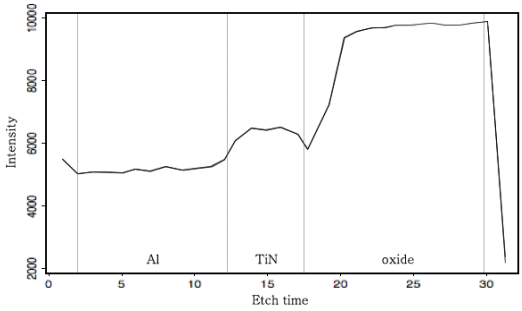


그림 5. 플라즈마 종류에 따른 Cl 발생 강도
Fig. 5. Kind of Cl intensity caused by plasma

이러한 특징을 이용하여, 공정 장비 내 센서 설치를 통해 빛의 파장을 실시간으로 관찰할 수 있다. 센서를 통해 관찰된 빛의 파장을 이용하여, 사전에 작성된 시나리오와 End-point를 확인하는 로직으로 End-point를 결정할 수 있다.

3. 기존의 모니터링 시스템

기존의 식각 공정에서는 플라즈마 속에 전극을 주입시켜 전자밀도, 전자온도, 플라즈마 전위를 측정할 수 있는 플라즈마주파수 프로브(Langmuir Probe), 물질파를 단일한 파동으로 만들어서 중성원자의 간섭무늬를 관찰하는 간섭계(Interferrometer), 그리고 진공 상태에서 시료분자를 이온화하여 생성된 이온의 질량을 측정할 수 있는 질량분석계(Mass Spectrometer)를 주로 사용하였다[6]. 하지만 이 측정기들은 플라즈마 반응이 진행되는 공정 장비 내부에 설치되어야 한다. 때문에, 공정 장비 내부의 전압, 주파수, 자기장, 가스 압력 등의 변수에 의해 공정 장비 내부에 설치된 측정기 자체에 문제가 발생할 확률이 높으며, 이로 인해 정확한 모니터링이 힘든 단점을 가지고 있다. 반면 OES 센서는 광학케이블을 이용하여 플라즈마 반응이 진행되는 공정 장비 외부에 설치할 수 있다. 그래서 OES 센서는 공정 장비 내부에 있음으로써 받는 영향을 전혀 받지 않는 장점을 가지고 있다.

4. OES Sensor

그림 [6]의 OES(Optical Emission Spectroscopy) 센서는 반도체 식각 공정에서 플라즈마 반응으로 인해 발생한 가스의 스펙트럼을 실시간으로 분석하고, 데이터로 가공하여 연결된 장치로 전송한다[5]. OES 센서는 광학 케이블을 이용하여 플라즈마 반응이 진행되는 공정 장비 외부에 설치할 수 있는 이점이 있다. 따라서 공정 장비 내부의 전압, 주파수, 자기장, 가스 압력 등의 변수에 영향을 전혀 받지 않는 장점을 가지고 있다. 내부로의 간섭을 받지 않기 때문에 좀 더 정확한 측정을 할 수 있고, 고장이 발생할 확률이 줄어들어 측정기의 수명을 연장 할 수 있다.

5. MFC 응용 프로그램

MFC(Microsoft Foundation Class)는 마이크로소프트사의 윈도우 응용 프로그램 개발용 클래스 라이브러리이다. Visual C++에 포함되어 있고, Win32 프로그래밍에 사용된다. 윈도우 기능이 복잡해짐에 따라 API를 직접 이용하는 것보다는 이러한 클래스 라이브러리를 사용하는 것이 훨씬 편리하다. MFC는 윈도우의 최신 기능을 도입함으로써 윈도우 프로그래밍을 위한 클래스 라이브러리의 사실상 표준이라 할 수 있다[7].

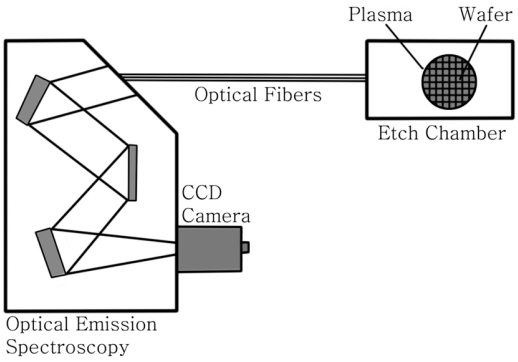


그림 6. OES 센서
Fig. 6. OES sensor

III. 본 론

1. 시스템 설계

[그림 7]의 시스템은 전체적으로 3개의 부분으로 나눠 볼 수 있다. 반도체 식각 공정을 실시간으로 관찰·분석하여 가공한 데이터를 전송하는 OES 센서, 센서로부터 전송받은 데이터를 1차적으로 가공하고 모니터링 시스템으로 전송하는 DLL Interface, 마지막으로 전송받은 데이터를 관리자에 의해 설정된 로직으로 재가공하고 설정된 시나리오에 맞게 공정이 진행되고 있는지를 감지하는 모니터링 시스템이 있다.

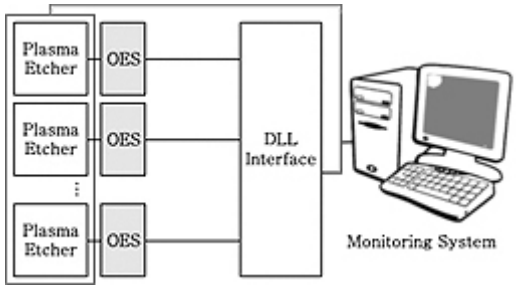


그림 7. 전체 시스템 구조
Fig. 7. System architecture

[그림 8]은 실제 공정의 모니터링 시스템을 표현한 그림이다. [그림 7]의 DLL Interface는 MFC 응용 프로그램에 포함되어 빌드되므로, 실제 공정은 [그림 8]과 같은 구조로 이루어진다[7].

1.1 OES Sensor

OES 센서는 플라즈마 식각 장비 바깥에 설치된다. [그림 8]과 같이 플라즈마 식각 장비에 구성된 투명한 렌즈와 OES 센서 사이를 광섬유 케이블로 연결하면 내부를 관찰할 수 있다. 관찰된 결과는 [그림 9]와 같이 일정 범위의 파장을 구간별로 나눈 CCD (Charge-coupled Device) Array에 맺히는데, 이를 USB Interface를 통해 DLL Interface로 전송한다[7].

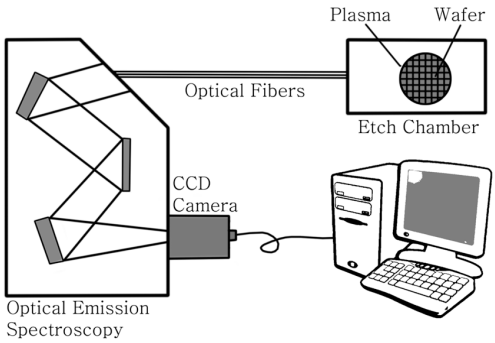


그림 8. 공정 상의 모니터링 시스템 구조
Fig. 8. System architecture in etching process

본 논문에서 기술하는 모니터링 시스템에서는 200nm~

900nm의 범위의 파장을 검출해낼 수 있는 OES 센서를 사용한다. 200nm~900nm는 자외선, 가시광선, 적외선에 해당하는 범위이다. OES 센서는 200nm~900nm의 범위를 3648개의 구간으로 나누어 각 구간별 강도를 측정한다. 즉, CCD Array가 3648의 길이로 구성된 것을 사용한다.

1.2 DLL Interface

DLL Interface는 OES 센서와 PC 사이에서 매우 중요한 역할을 하는데, 크게 3가지의 일을 수행한다.

첫 번째로 OES 센서로부터 전송받은 데이터를 저장할 메모리 공간을 미리 예약하고 데이터를 전송받을 준비작업을 수행한다. 메모리 공간은 연결된 OES 센서의 개수에 맞는 공간이 필요하다. 준비작업에는 여러 개의 OES 센서를 구분하기 위한 구조적 정보를 구성한다. 구조의 포맷은 c언어 구조체를 사용한다. 두 번째로 주기적으로 데이터를 전송받는다. 데이터 전송은 OES 센서의 CCD Array의 값을 예약한 메모리 공간에 저장하는 것으로 이루어진다. 저장 형태는 MFC에서 제공하는 long 자료형의 배열로, 하나의 OES 센서 당 3648 길이의 배열을 사용한다. 세 번째로 데이터를 1차 가공하는 일을 수행한다. MFC에서 사용할 수 있는 형태의 데이터 가공함수들을 제공한다.

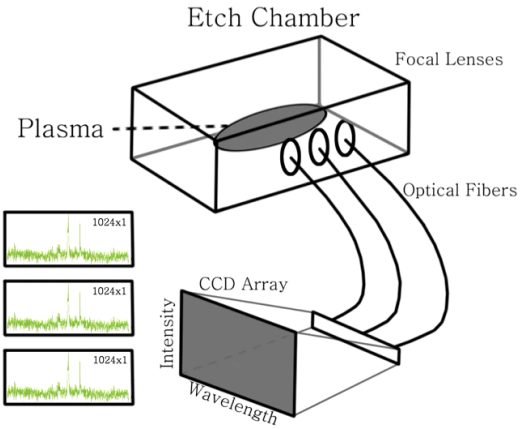


그림 9. 공정에서의 OES 센서 구성
Fig. 9. Installation of OES sensor in etching process

1.3 Monitoring System (End-point Detecting System)

모니터링 시스템의 기본적인 역할은 전송받은 데이터를 분석하는 것이다. 부가적으로, 분석된 데이터를 재가공 및 수집하는 작업을 통해 자동화 효율을 높이는 기능들을 제공한다.

〔그림 10〕은 모니터링 시스템에서 데이터를 전송받는 구조를 나타낸 그림이다. 모니터링 시스템에는 데이터를 전송받는 하나의 thread가 존재한다. 이 thread에서는 DLL Interface에서 예약한 메모리 공간에 접근하여 데이터를 얻는다. 이를 Data gathering thread라 하는데, ms 단위로 갱신되는 데이터를 실시간으로 처리하기위해 모니터링 시스템의 메인흐름에 포함되지 않고 별도의 thread로 구성된다.

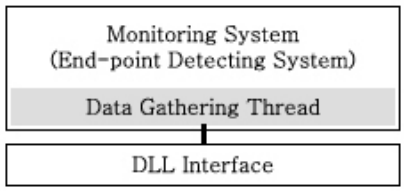


그림 10. 모니터링 시스템으로의 데이터 전송 구조
Fig. 10. Data-transfer in the monitoring system

모니터링 시스템은 관리자에게 현재 공정의 상태를 스펙트럼 차트를 기반으로 보여준다. 〔그림 11〕과 같이, 차트는 전체 파장 범위의 실시간 intensity를 보여주는 것과 원하는 파장 범위의 intensity를 시간 흐름에 근거하여 보여주는 것이 있다.

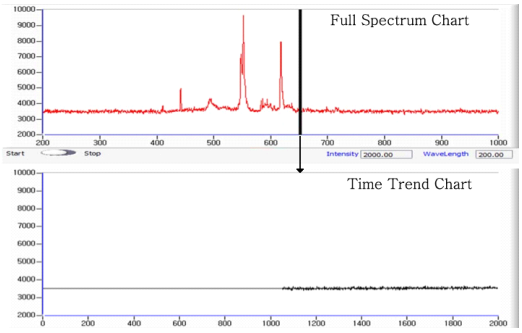


그림 11. 모니터링 시스템의 차트 UI
Fig. 11. Chart UI in the monitoring system

모니터링 시스템의 가장 큰 역할은 End-point를 찾는 작업이다. 관리자는 〔그림 12〕과 같은 화면을 통해 현재 공정의 시나리오를 작성한다. 모니터링 시스템은 작성된 시나리오에 따라 공정이 진행되고 있는지를 확인하고, 시나리오의 End-point 항목의 상황에 도달하면 공정을 중단하게 된다.

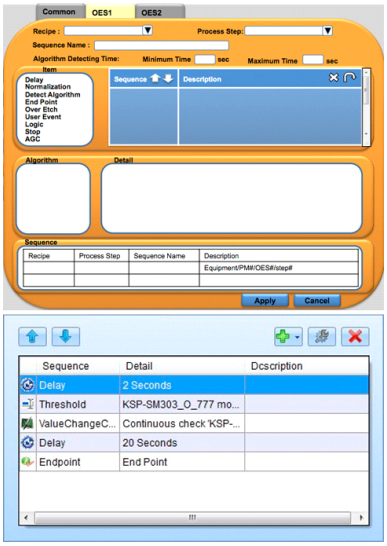


그림 12. 모니터링 시스템에서의 시나리오 작성 화면
Fig. 12. Scenario setting UI in the monitoring system

〔그림 13〕은 모니터링 시스템의 전체 흐름을 중요 요소만 추려서 간단히 나타낸 것이다.

2. 시스템 구현

2.1 OES Sensor information

OES 센서는 DLL Interface에 의해 제어된다. OES 센서는 센서를 사용하기 위한 초기화 단계에서 DLL Interface로 센서 정보를 제공한다. 센서 정보는 sensor id, reserved name, wave length, wave length table로 구성된다. 다수의 OES 센서가 연결 되어있는 경우, 전송받은 센서 정보를 통해 각 센서를 구별할 수 있다. 〔그림 14〕는 DLL Interface에서 센서 정보를 저장하기 위한 구조체를 정의한 것이다.

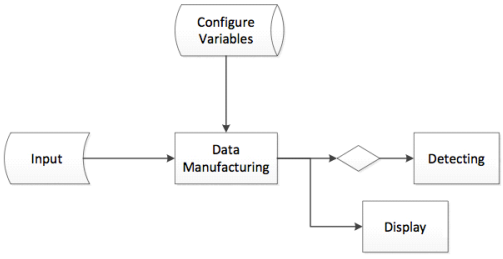


그림 13. 모니터링 시스템의 전체 흐름
Fig. 13. System flow of the monitoring system

```
typedef struct OesUnitInformation {
    s32 id;
    char model_name[256];
    char serial[256];
    s32 wave_length;
    double wave_length_table[4096];
} OesUnitInformation;
```

그림 14. OES 센서 정보 구조체
Fig. 14. Structure of OES sensor information

2.2 DLL Interface

〔그림 15〕는 DLL Interface의 OES 센서를 제어하는 API 리스트이다.

```
OESDLL_API s32 OesQueryDevice(s32 _index, struct OesUnitInformation *
    _oes_information_pointer);
OESDLL_API s32 OesSetIntegrationTime (s32 _id, s32 _integration_time);
OESDLL_API s32 OesGetIntegrationTime(s32 _id);
typedef void (*oes_callback)(s32 _id, SYSTEMTIME * _time, TCHAR * _filename ,
    long * _array, s32 _length);
OESDLL_API s32 OesPrepare(TCHAR * _data_path,s32 _integration_time);
OESDLL_API s32 OesProbe(void);
OESDLL_API s32 OesRegister(oes_callback _callback);
OESDLL_API s32 OesStart(void);
OESDLL_API s32 OesStop(void);
OESDLL_API s32 OesTerminate(void);
```

그림 15. OES 센서 제어 API
Fig. 15. API to control OES sensor

DLL Interface의 OES Register API는 CCD Array의 값을 주기적으로 전송받아 예약된 메모리 공간에 저장하는 하나의 callback 함수를 thread로 구성한다. 이 역할을 시작으로 모니터링 시스템의 흐름이 시작된다. CCD Array의 값을 메모리에 저장한 뒤, Data gathering thread에 데이터가 갱신된 것을 알린다.

〔그림 16〕은 OES Register API를 이용하여 thread로 구성하는 callback 함수를 구현한 것이다.

```
CEDSApp* pApp = (CEDSApp*)AfxGetApp();

if(pApp->GetOesDevCount() <= 0) return;

if(pApp->m_OesThread) {
    OESDataType* pData = new OESDataType;
    pData->_id      = _id;
    pData->_time    = _time;
    pData->_file    = _filename;
    pData->_array   = _array;
    pData->_length  = _length;

    pApp->m_OesThread->PostThreadMessage(THREAD_MSG_OES, 0, LPARAM(pData));

    pData = NULL;
}
```

그림 16. OES callback 함수
Fig. 16. OES callback function

모니터링 시스템이 시작되면 OES 센서 정보를 얻기 위해 각 센서에 질의하는 작업을 거친다. 센서 정보 구조체 변수를 OES Query Device API의 파라미터로 하여 API를 호출하면 구조체 변수에 OES 센서의 정보가 저장된다. OES 센서의 개수만큼 질의를 반복하여 각 센서의 정보를 얻는다. 〔그림 17〕은 OES 센서 정보를 얻는 루틴을 구현한 것이다.

〔그림 18〕은 DLL Interface에서 CCD Array의 값을 1

차적으로 가공할 수 있는 함수 리스트의 일부이다. OES 센서의 output에 맞게 설계되어, 복잡한 가공 로직도 지연이 거의 없는 높은 성능을 보여준다.

```
s_OesDeviceCount = OesProbe();
if(s_OesDeviceCount <= 0) {
    return FALSE;
}

OesRegister(OesDeviceHandler);
OesPrepare(_T("C:\\Oes"), 10);

s_OesModule = new SOesModule[s_OesDeviceCount];
for(int idx = 0; idx < s_OesDeviceCount; idx++) {
    s_OesModule[idx].pid = pm;
}
```

그림 17. OES 센서 정보 질의 과정
Fig. 17. Getting routine of OES sensor information

2.3 Monitoring System

모니터링 시스템의 메인 class는 App class이다. App class는 시스템 내의 모든 객체에 접근할 수 있고, 반대로 모든 객체의 App class로의 접근 또한 허용한다. App class에서 시스템 내 객체들의 포인터를 리스트로 관리하는 방법으로 이를 구현한다. [그림 19]는 모니터링 시스템 내 객체 구조를 나타낸 것이다.

Shift	Takes all data points that comprise a spectral graph (*a*) and shifts them *n* units to the right (greater). A negative *n* will shift left (lower). Units are dependent upon the resolution of the spectrograph being used. Common is ½ nm per unit, but high-resolution instruments move ¼ nm per unit.	Shift(a,n)
Max	The resulting value is the largest value for either *a* or *b*. During execution, at any given time, this function may return *a* or *b*.	Max(a,b)
Min	The resulting value is the smallest value for either *a* or *b*. At any time during execution, this function may return *a* or *b*.	Min(a,b)
Abs Val	If *a* is *0* or higher, then the result is *a*. If *a* is negative, then the result is made positive.	Abs(a)
EXP	Performs the exponent function on the value for *a*.	Exp(a)
LOG10	Performs the log10 function to the value for *a*.	Log10(a)
Sqrt	Results in the square root of the value for *a*.	Sqrt(a)
TAN	Performs the trigonometry function tangent to the value *a*.	Tan(a)

그림 18. DLL Interface에서 제공하는 가공 함수
Fig. 18. Manufacturing function set of DLL interface

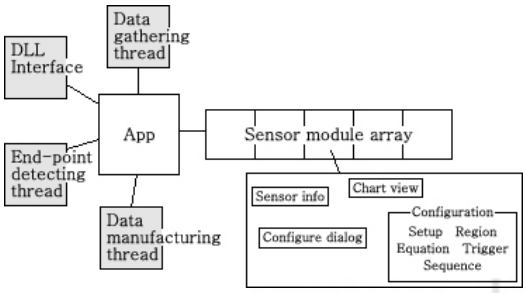


그림 19. 구조화된 객체 맵
Fig. 19. Map of structured objects

모니터링 시스템에서 OES 센서와 관련된 모든 정보는 Sensor module 객체 내에 구성된다. 구성 상세에 대한 내용은 하위 절에서 설명한다.

모니터링 시스템의 흐름을 살펴보면 [그림 20]에서 보듯이 3개의 thread가 있다. 각 thread는 동일한 데이터를 이용하여 작업을 수행한다. 즉, 여러 thread의 데이터 공유가 일정 주기로 반복된다. 각 thread에서 데이터 사용 시, thread간 충돌을 방지하기 위해 raw data(DLL Interface가 CCD Array 값을 메모리에 저장한 가공 전 데이터) 크기의 2배에 해당하는 메모리 공간을 추가로 사용한다.

[그림 20]에서처럼 모니터링 시스템은 먼저 OES 센서들을 준비하는 작업으로 시작된다. OES 센서로부터 어떤 수치의 주기로 데이터를 전송받을 것인지 설정하고, 각 센서들의 정보를 얻는다. 이 과정에서 오류가 발생하면 시스템은 시작될 수 없다. 이 후의 흐름은, OES callback thread → Data gathering thread → Data manufacturing thread → Data displaying 순서의 event pushing으로 이루어진다. 위 과정의 반복으로 시스템이 진행된다. Event pushing은 일반적으로 데이터가 갱신되었음을 알리는 것을 의미한다. 위 과정의 각 항목에 대한 내용은 하위 절에서 설명한다.

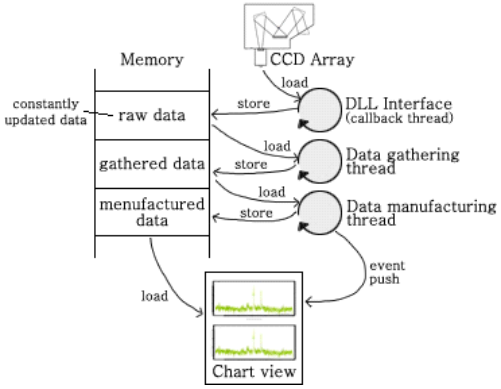


그림 20. 모니터링 시스템의 전체 흐름 구조
Fig. 20. Structured system flow of the monitoring system

2.3.1 User Interface

모니터링 시스템의 주요 User Interface는 Configuration dialog와 Chart view, 그리고 Recipe view로 세 부분으로 나눌 수 있다. Configuration dialog는 [그림 21]과 같이 구현된다.

Chart view는 [그림 22]에서와 같이 Full spectrum chart와 Time trend chart로 구성된다. Full spectrum chart는 어느 범위의 파장이 어떤 원소의 것인지를 알려주는 Data source

리스트가 함께 구성되고, Time trend chart는 관리자에 의해 사전에 설정된 Equation 리스트가 함께 구성된다.

Data source 리스트에서 원하는 항목을 선택하면 Full spectrum chart에 해당 범위가 highlight되어 관리자에게 편의를 제공한다. Equation 리스트에서 원하는 항목을 선택하면 관리자가 사전에 설정한 범위의 데이터를 설정된 방법으로 가공한 뒤 시간 흐름에 근거하여 Time trend chart에 보여준다.

〔그림 23〕과 같이 Recipe view는 Process의 현재 상태를 나타내는 view이다. 현재 공정의 상태가 관리자가 설정한 시나리오의 어느 항목에 해당되는지를 보여준다.

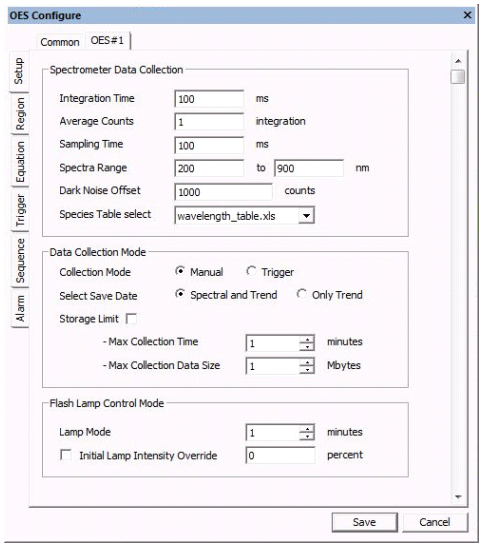


그림 21. Configuration 대화상자
Fig. 21. Configuration dialog

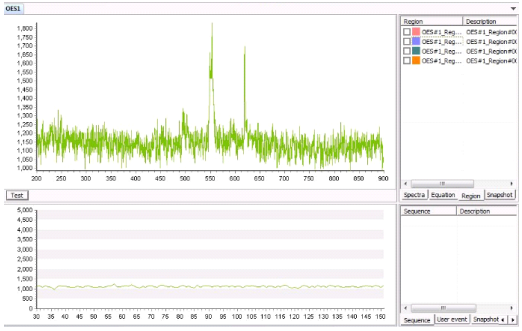


그림 22. Chart 화면
Fig. 22. Chart view

그림 23. Recipe 화면
Fig. 23. Recipe view

2.3.2 Configuration

관리자는 공정에 맞게 시스템 전체를 사전에 설정할 수 있다. OES 센서로부터 전송받는 데이터는 공정 장비의 설비나 주변 환경 등의 요소에 의해 변형되는 것이 다반사이다. 때문에 모니터링 시스템에서는 이를 사용가능한 데이터로 가공한 뒤 사용하는데, 관리자는 어떤 방법으로 데이터를 가공할 것인가를 사전에 설정한다.

이 후, 데이터를 어떻게 추가 가공하여 사용할 것인지를 설정하고, 공정 시나리오를 작성하는 것을 마지막으로 사전 설정 단계가 마무리된다.

〔그림 24〕는 Configuration의 항목을 상세 설명과 함께 나타낸 것이다.

Item	Function
Setup	set sensor's features and storage options
Region	set regions of each sensor
Equation	set up a series of displaying options using regions
Trigger	set options for the save data
Sequence	The goal of this setting is EPD(End-point detecting). Make a series of steps making the EPD signal after app start gathering from sensors.

그림 24. Configuration 항목
Fig. 24. Item of Configuration

Configuration에서 정보는 〔그림 19〕의 Sensor Module 객체의 하위 객체로 구성되며, App class를 통해 접근할 수 있다. Configuration에서는 End-point를 찾기 위한 과정을 어떤 방법으로 수행해나갈 것인가를 전체적으로 설정한다고 할 수 있다. 이를 설정하는 일련의 과정의 예를 도식화하면 다음과 같다. Configuration 과정은 다음과 같다. Setup에서 변형된 데이터를 사용가능하도록 복원하는 상세 설정을 한 뒤, Region에서 현재 공정에서 관찰하고자하는 파장 범위를 설정한다. 그리고 Equation에서 각 Region을 어떻게 가공하고 어떤 산출물을 만들어낼지 설정하고, Trigger에서 데이터 수집 조건을 설정한다. 마지막으로 Sequence에서 앞서 설정한 산출물을 이용하여 공정의 시나리오를 작성한다.

2.3.3 Data gathering

Data gathering은 〔그림 20〕의 Data gathering thread가 하는 일을 의미한다. Data gathering thread는 DLL Interface에 등록된 OES callback thread가 저장한 데이터를 복사하여 gathered data를 만들고 Data manufacturing thread에 가공할 수 있는 데이터가 준비되었음을 알린다. 〔그림 25〕는 Data gathering thread의 구현 일부이다.

2.3.4 Data manufacturing

Data manufacturing은 gathered data를 사용가능한 데이터로 1차 가공하는 일을 의미한다. Configuration의 setup에 설정된 데이터 가공 정보를 토대로 manufactured data를 만들어내고, Chart view에 데이터가 갱신되었음을 알린다. Data manufacturing은 Data manufacturing thread에 의해 일어나는데, 이 때 가공된 manufactured data는 Chart view의 Full spectrum chart에 display 되는 데이터이다. 〔그림 26〕은 Data manufacturing thread의 구현 일부이다.

```
if(msg.message == THREAD_MSG_OES && msg.lParam != NULL) {
    OESDataType* tData = (OESDataType*)msg.lParam;
    OESDataType* cData = new OESDataType;
    memcpy(cData, tData, sizeof(OESDataType));
    delete tData;
    tData = NULL;

    POSITION pos = pApp->m_listSM_Ct_Wnd.GetHeadPosition();
    while(pos != NULL) {
        SMOesInfo_doc* pCurrent = NULL;
        pCurrent = (SMOesInfo_doc*)pApp->m_listSM_Ct_Wnd.GetNext
            (pos);
        if(pCurrent->_pSMInfo->Info.id == cData->_id) {
            pCurrent->SetSMOesInfo(cData);
            break;
        }
    }

    delete cData;
    cData = NULL;
}
```

그림 25. Data gathering thread의 구현
Fig. 25. Implementation of data gathering thread

```
_dLength    = _doc->_length;
if(_avgData == NULL) {      // At once, to initializing.
    _avgData = new long[_dLength];
    memset(_avgData, 0, sizeof(long) * _dLength);
    _tmpData = new long[_dLength];
    memset(_avgData, 0, sizeof(long) * _dLength);
}
if(_pDMaxACnt <= 0) {       // At once, to initializing.
    _pDMaxACnt  = _pSMInfo->Configure.Setup.AverageCounts;
    _pDAvgCnt   = 0;
    memset(_avgData, 0, sizeof(long) * _dLength);
}
if(_pDAvgCnt == 0) {
    memset(_avgData, 0, sizeof(long) * _dLength);
}

long* _pAvg = _avgData;
long* _pCol = _doc->_array;
int loopCnt = _dLength;
while(loopCnt--) {
    *_pAvg += *_pCol;
    _pAvg++;
    _pCol++;
}
_pDAvgCnt++;

if(_pDSValid == SMDATA_INVALID && _pDTValid == SMDATA_INVALID)
    _pData = NULL;

if(_pDAvgCnt == _pDMaxACnt) {
    long* _pTmp = _tmpData;
    _pAvg = _avgData;
    loopCnt = _dLength;
    while(loopCnt--) {
        *_pTmp = *_pAvg / _pDMaxACnt;
        _pTmp++;
        _pAvg++;
    }
    _pData = _tmpData;
    _pDSValid = SMDATA_VALID;
    _pDTValid = SMDATA_VALID;
    _pDAvgCnt = 0;
}
```

그림 26. Data manufacturing thread의 구현
Fig. 26. Implementation of data manufacturing thread

2.3.5 Data displaying

Chart view는 Full spectrum chart와 Time trend chart로 구성된다. Full spectrum chart는 Data manufacturing thread로부터 데이터가 갱신되었음을 알리는 이벤트가 푸싱 되면 manufactured data를 chart data로 설정하고 chart를 갱신한다. 〔그림 27〕은 Full spectrum chart를 갱신하는 부분의 구현 일부이다.

```
if(pCFrame->m_SMOesInfo->_pDSValid == SMDATA_INVALID) return 0;
if(pCFrame->m_SMOesInfo->_pData == NULL) return 0;
m_chartCtrl[0].Add(
                pCFrame->m_SMOesInfo->_pSMInfo->Info.wave_length,
                pCFrame->m_SMOesInfo->_pSMInfo->Info.wave_length_table,
                pCFrame->m_SMOesInfo->_pData
                );
if(m_DNOSIdx != -1)
    m_chartCtrl[m_DNOSIdx].CheckDataSource();
pCFrame->m_SMOesInfo->_pDSValid = SMDATA_INVALID;
return 0;
```

그림 27. Full spectrum chart displaying의 구현
Fig. 27. Implementation of full spectrum chart displaying

Time trend chart는 Equation 리스트에서 관리자가 선택한 Equation의 time trend를, 선택한 시점에서부터 보여준다. Equation data의 가공은 관리자가 선택한 시점에서부터 실시간으로 이루어진다. 즉, 〔그림 28〕은 지속적으로 갱신되는 manufactured data를, 설정된 로직으로 재가공하여, 그 값을 시간 흐름에 근거하여 보여준다.

```
if(_cState == STATUS_ON && _tIndex == _idx)
    return;

_tIndex = _idx;
_cState = STATUS_ON;
m_chartCtrl[0].Clear();
m_chartCtrl.GetAxis().GetBottom().SetMinimum(0);
m_chartCtrl.GetAxis().GetBottom().SetMaximum(MAX_TREND_ELE);
_cIsBtmVisible = CBTM_STATIC;
```

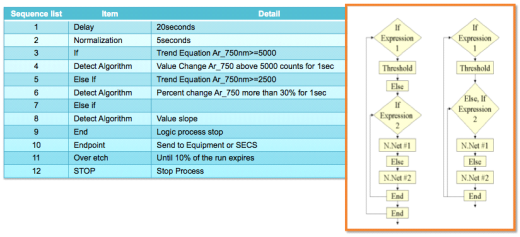
그림 28. Time trend element 선택 구현
Fig. 28. Implementation of selection of time trend element

2.3.6 End-point Detection

반도체 식각 공정 모니터링 시스템의 최종 목표는 EPD(End-point detection)이다. EPD는 관리자가 작성한 시나리오에 의해 이루어진다. 관리자는 Configuration의 많은 설정 기능들을 이용하여, 원본 데이터와 함께 쓰임새 있게 가공된 다양한 형태의 데이터를 얻을 수 있다. 이 데이터들을 토대로 작성된 시나리오는 해당 공정의 레시피가 된다. 〔그림 29〕는 관리자가 Configuration의 sequence 항목에서 작성한 시나리오의 형태와 모니터링 시스템의 내부에서 해당 시나리오가 적용되는 로직을 나타낸 것이다.

모니터링 시스템은 manufactured data가 갱신될 때마다 현재 sequence item에 기술된 상태에 도달하였거나 또는 기술된 조건에 부합하는지를 판단한다. 판단 지표는 현재 공정

상태를 가장 잘 나타내는 Chart data이다. EPD가 완료되면 EPD Signal을 실제 공정 장비에 전송하여 공정을 중단한다. [그림 30]은 sequence의 item 종류와 그 기능을 나열한 것이다.



Sequence list	Item	Detail
1	Delay	20seconds
2	Normalization	5seconds
3	If	Trend Equation Ar_750nm>=5000
4	Detect Algorithm	Value Change Ar_750 above 5000 counts for 1sec
5	Else If	Trend Equation Ar_750nm>=2500
6	Detect Algorithm	Percent change Ar_750 more than 30% for 1sec
7	Else if	
8	Detect Algorithm	Value slope
9	End	Logic process stop
10	Endpoint	Send to Equipment or SECS
11	Over etch	Until 10% of the run expires
12	STOP	Stop Process

그림 29. EPD sequence의 예
Fig. 29. Example of EPD sequence

Item	Function
Delay	set a time that does not monitoring after step starting
Normalization	set a time to calculate the average value for the input data's intensity
Detect Algorithm	set algorithm to perform EPD
Endpoint	send a EPD signal to process module or system UI
Over Etch	set a over-etch time after EPD
User event	set user event
Logic	set logic to perform EPD
Stop	stop data gathering

그림 30. Sequence 항목
Fig. 30. Item of sequence

2.3.7 System Flow

시스템은 센서로 부터 데이터를 수신하는 것을 시작으로 진행된다. 센서는 데이터를 메모리의 일정 영역에 주기적으로 갱신하고, 이를 데이터 가공 프로세스에 알린다. 데이터 가공 프로세스는 알림을 수신하면 사전에 설정된 Setup 항목에 맞게 데이터 Sampling을 진행한다. Setup에서 설정된 수치로 원본 데이터에서 Noise를 제거한 값을 설정된 Count 만큼, 그리고 설정된 주기로 수신한 뒤 평균값을 계산한다. 100ms 주기로 10 Count의 Sampling을 진행하면 초당 10개의 데이터의 평균값을 시스템에서 활용하게 되는데, 이는 각기 다른 공정 환경에 따른 Normalization을 위한 과정이다. 가공이 완료되어 데이터의 갱신이 일어날 때마다 Chart의 UI를 갱신하는 프로세스, Trigger 프로세스, Sequence 프로세스, 그리고 Trigger 및 Sequence 프로세스에서 참조하는 Equation 객체의 최신 값을 시스템 내에 설계된 Equation Cache에 반영하는 프로세스에 알려진다. Trigger 및 Sequence 프로세스는 Equation Cache를 참조하여 조건부 기능을 실행한다. Chart의 UI를 갱신하는 프로세스는 최신 데이터 및 Chart 오른편의 Source check 상태에 맞게 UI를 갱신한다. Trigger 프로세스는 조건 만족 시, 데이터를 수집·저장하는 새로운 프로세스를 생성한다. Sequence 프로세스는 내부에서 순차적인 처리를 하는 자료구조를 이용하여, 조건 만족 시 새로운 프로세스를 생성하여 수행하도록 진행한다. 이에 의한 시나리오 진행 상태는 Recipe view에서 확인할 수 있고, Sequence 프로세스에서 EPD 이벤트가 실행되면 공정이 종료된다.

IV. 실험 및 고찰

본 시스템에서 사용된 OES H/W는 최소 10ms 주기로 데이터를 전송하는 것이 허용된다. 하지만 모니터링 시스템에서 구현한 Interface에서는 최소 100ms 주기로 데이터를 수신 받을 수 있도록 설계되었다. 이는 시스템 내 데이터 처리 시, 충돌 없이 원활한 메모리 공유를 위한 수치이다. 따라서 Interface의 최대 성능인 100ms 주기의 환경에서 앞의 시스템 구현에서 예를 든 10 Count 당 1 Sampling으로 실험을 진행한다. 또한, 공정의 특성상 실제 공정과 비슷한 환경을 만들어 낼 수 없으므로, Data gathering의 정상 수행 여부와 Configuration에 의한 가공 결과가 Chart view에 명확히 적용되는지에 대한 여부, 그리고 Log를 통해 실험 수행 시간을 확인한다. [그림 32]는 Data gathering 및 가공이 설정된 환경에 따라 기대되는 시간 내에 성공적으로 수행되었음을 보여준다[8].

실험은 일정 시간동안 몇 번의 Data Sampling이 진행되는지 확인하고, 기대되는 수치에 부합하는지를 확인하는 것으로 진행한다. [그림 31]은 실험을 위해 모니터링 시스템의 빌드 전 코드에 삽입한 소스코드이다[9].

[그림 31]의 소스코드는 모니터링 시스템에서 Data gathering을 시작하는 시간과 종료하는 시간을 Log에 기록하고, 이 동안 진행된 Data Sampling 횟수를 종료와 동시에 Log에 기록하기 위한 것이다. [그림 32]의 실험 결과, 28.5분 동안 1712번의 Data Sampling이 진행되었다. 이는 1분 당 60번의 Data Sampling이 진행되었다고 볼 수 있는데, 사전에 설정한 Data Sampling 주기와 일치하는 결과이다.

```
void CEDSConfigureDiag::OnBnClickedOesStartBtn()
{
    CEDSApp* pApp = (CEDSApp*)AfxGetApp();
    pApp->m_pForegroundPModule->SetOesStatusStart();
    pApp->m_testCount = 0; // Sampling 횟수 초기화
}

void CEDSConfigureDiag::OnBnClickedOesStopBtn()
{   CEDSApp* pApp = (CEDSApp*)AfxGetApp();
    pApp->m_pForegroundPModule->SetOesStatusStop();
    // Sampling 횟수 Log에 출력
    gLog(LOG, (double)pApp->m_testCount);
}

void COESSensorModule::SetWorkedData(long* pData) {
    cs_wData.Lock();
    memset(worked, 0, sizeof(long) * info.wave_length);
    memcpy(worked, pData, sizeof(long) * info.wave_length);
    cs_wData.Unlock();
    CEDSApp* pApp = (CEDSApp*)AfxGetApp();
    pApp->m_testCount++; // Sampling 횟수 1 증가
}
```

그림 31. 실험을 위해 삽입된 소스코드
Fig. 31. Source code for test

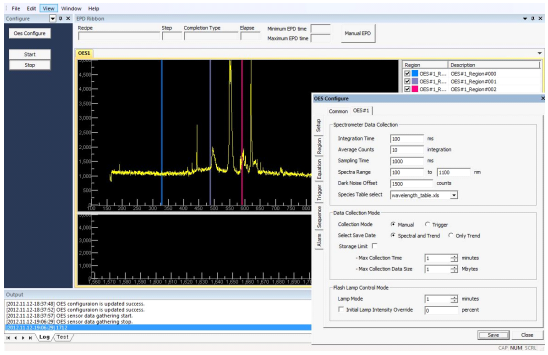


그림 32 실험 진행 후 모니터링 시스템
Fig. 32 The monitoring system after test

본 실험을 통해, 공정 환경에 맞게 실시간으로 데이터 가공 방법의 제어가 가능하고, Normalization 기능으로 공정에서 요구하는 최소 단위의 프로세스 수행 주기에서 데이터의 신뢰도를 높임으로써, 각기 다른 공정 환경에서 제어 효율을 높이는 데 큰 역할을 한다는 것을 검증해낼 수 있었다.

V. 결론

본 논문에서 개발한 모니터링 시스템은 다음과 같은 우수성을 가진다. 첫째로, 공정장비 외부에서 장비 내부를 정확하게 관찰할 수 있는 광학 센서의 사용으로, 공정장비 내부에 센서를 설치하는데 필요한 부담을 줄일 수 있다. 둘째로, Equation을 이용하여 1~N차에 걸친 로직을 만들 수 있어서, 무한에 가까운 수의 데이터 가공 방법을 사용할 수 있다. 마지막으로, 자동화된 검사 시스템은 예측되지 않은 상황에서 1차적인 대처를 수행하여 피해를 사전에 방지한다.

반도체 식각 공정 모니터링 시스템은 센서 데이터 분석을 통해 공정의 가시적인 상태를 관찰하는 동시에, 현재 공정 단계를 진단하는 로직을 실시간으로 실행하여 EPD(End-point Detecting) 기능을 수행해낸다.

모니터링 시스템이 없는 반도체 식각공정에서는 불량을 사전에 예측할 수 없는 경우가 다반사이다. 컴퓨터가 공정을 제어함과 동시에 모니터링 기능을 함께 수행하면, 불량 발생으로 인한 피해를 없애고, 공정 자동화 효율을 높일 수 있는 이점이 생긴다.

본 논문에서는 반도체 식각공정 모니터링 시스템이라는 하나의 제한된 주제로서 설명하였지만, 실시간 데이터 처리를 통한 모니터링 시스템은 얼마든지 다른 사업이나 산업에 응용되어 적용될 수 있겠다.

감사의 글

본 연구 논문은 2012년도 지식경제부, 지식경제 R&D 전략기획단 미래산업선도기술개발사업(과제번호:10042421), 2013년도 국민대학교 교내연구지원금 및 중소기업청, 2011년도 산학연공동기술개발사업(No. 00045590)의 지원을 받아 수행되었습니다.

참고문헌

[1] S. Limanond, J. Si, and K. Tsakalis, "Monitoring and control of semiconductor manufacturing processes," IEEE Control Systems, vol. 18, issue 6, pp. 46 - 58, Dec 1998.

[2] M. Mozetic, U. Cvelbar, A. Vesel, N. Krstulovic, S. Milosevic, "Interaction of Oxygen Plasma with Aluminum Substrates," IEEE Transactions on Plasma Science, vol. 36, no. 4, pp. 868 - 869, Aug. 2008.

[3] Z. Navratil, D. Trunec, R. Smid, and L. Lazar, "A software for optical emission spectroscopy - problem formulation and application to plasma diagnostics," Czech. J. Physics, vol. 56, 2006.

[4] Z. Navratil, D. Trunec, R. Smid, and L. Lazar, "A software for optical emission spectroscopy - problem formulation and application to plasma

diagnostics," Czech. J. Physics, vol. 56, 2006.
〔5〕 R. Chen, "OES-based Sensing for Plasma Processing in IC Manufacturing," Doctor of Philosophy in Engineering - Electrical Engineering and Computer Sciences - University of California at Berkeley.
〔6〕 H. H. Yue, S. J. Qin, R. J. Markle, C. Nauert, and M. Gatto, "Fault Detection of Plasma Etchers Using Optical Emission Spectra," IEEE Trans. Semicond. Manuf., vol. 13, no. 3, pp. 374 - 385, Aug. 2000.
〔7〕 R. N. Calheiros, R. Ranjan, A. Beloglazov, A. F. De Rose, R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Software: Practice and experience, vol. 41, issue 1, pp. 23 - 50, Jan. 2011.
〔8〕 S. C. Kim, "An Efficient Data Transmission Strategy using Adaptive-Tier Low Transmission Power Schedule in a Steady-state of BMA," Journal of KSCI, Vol. 15, No. 5, pp. 103-111, May. 2010.
〔9〕 I. H. Kim, "A Point of Production System for Semiconductor Wafer Dicing Process," Journal of KSCI, Vol. 14, No. 10, pp. 55-61, Oct. 2009.

저 자 소 개

김 상 철
1994-2000: 삼성 SDS, 삼성 테크윈 시스템 엔지니어
2005: 미국 오클라호마 주립대학교, Electrical & Computer Eng., 공학박사
현재: 국민대학교 컴퓨터공학부 부교수
관심분야: 정보통신, IT 융합 S/W
Email: sckim7@kookmin.ac.kr