제33회 한국정보처리학회 춘계학술발표대회 논문집 제17권 제1호 (2010. 4)

반도체 생산 공정 디바이스 관리 및 모니터링에 관한 연구

송경수*, 이리나*, 니 게오르기*, 김수희*
*호서대학교 컴퓨터공학과
e-mail : shkim@hoseo.edu

A Study on Semiconductor Process Device Managing and Monitoring

Kyung-Soo Song*, Li-Na Lee*, Georgy Ni*, Su-Hee Kim*
*Dept. of Computer Engineering, Hoseo University

요 약

반도체 공정 과정에서 설비의 효율의 높이고 생산성 향상을 도모하며 라인의 품질 사고 방지를 위하여 공정 진행 과정을 실시간으로 모니터링하는 것은 매우 중요하다. 이 연구에서는 반도체 공정의 전반적인 작업현황을 실시간으로 파악하고 비정상적인 상황이 발생하는 경우 즉각적으로 대응하기 위해 생산현장의 임베디드 PC들과 통신하여 공정 처리 데이터를 저장하고 이상 발생과 이력 정보 등을 제공하는 웹기반 모니터링시스템을 설계하고 구현하고자 한다. 이 논문의 궁극적인 목표는 작업라인의 현황을 실시간으로 파악하는 온라인 시스템을 구축하여 인력의 고도화를 도모하고 기업의 생산성 증대와 제품의 품질 향상에 기여하는 것이다.

1. 서론

현재 우리나라 반도체 공정기술은 세계 최고 수준의 경쟁력을 갖고 있다. 반면 반도체 공정을 실시간으로 모니터링하고 분석하는 진단기술은 많은 연구가 진행되고 있음에도 공정기술에 부합하는 수준에 도달하지 못하고 있다는 평을 받고 있다. 미래 반도체 산업에서 경쟁력을 확보하려면 반도체 제조에 대한 실시간 모니터링 및 진단을 통한 제어기술개발 등을 통해 생산시간을 단축하고, 수율 향상을 이루어내는 진단기술이 매우 중요하다. 다시 말하면 공정과정이나 설비에 대한 관리부분에서 이벤트 상황 등을 자동으로 감지하는 기술의 개발이 시급한 상황이다.

이러한 기술들에 의해 얻어지는 파급효과는 다음과 같다.

첫째, 설비효율의 증가에 따른 생산성 향상

그 이유는 반도체 공정 작업과정은 일의 대부분을 작업자가 진행하므로, 작업자가 설비에 액션을 취하지 않으면 그 설비는 가동되지 않으므로, 실시간 모니터링 및 제어기술개발을 통한 설비 자동화를 구축했을 경우 설비 운용을 극대화 할 수 있다.

둘째, 인력의 고도화

협소한 작업 공간에서 적은 인원으로 많은 양의 일을 처리할 수 있다.

셋째, 라인의 품질사고 방지

반도체 설비의 자동화를 통해 사람이 실수 할 수 있는 부분을 기계적으로 처리함으로써 에러율을 줄이고 라인의 작업 효율을 극대화 시킬 수 있다.

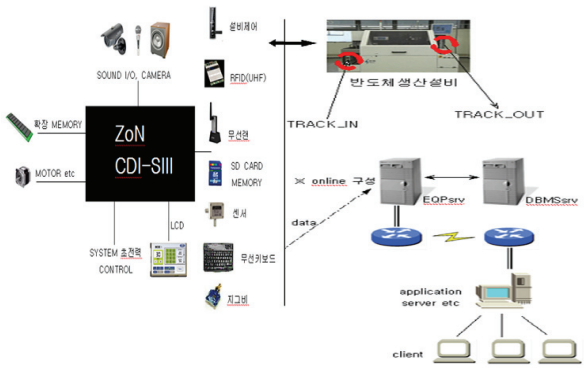
넷째, 작업 라인의 온라인 구축 및 누적 데이터 활용을 통한 사고 예방을 들 수 있다.

이 연구에서는 반도체 공정의 전반적인 작업현황을 실시간으로 파악하고 비정상적인 상황이 발생하는 경우 즉각적으로 대응하기 위해 생산현장의 임베디드 PC들과 통신하여 공정 처리 데이터를 저장하고 이상 발생과 이력 정보 등을 제공하는 웹기반 모니터링시스템을 설계하고 구현하고자 한다.

2. 시스템 개요

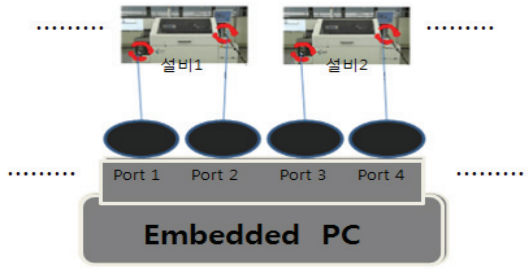
먼저 반도체 생산 공정에 대해 알아보면 일반적으로 자재를 담는 박스와 같은 역할을 하는 Magazine이 라인위에서 설비에 투입되고 설비 작업이 끝나면 Magazine이 설비 밖으로 나오게 되는 방식으로 진행된다.

다음의 <그림 1>을 살펴보면 가운데 실선을 중심으로 좌측 부분은 임베디드 PC 구현을 위한 보드에 관련된 부분이고 우측은 상단의 반도체 생산설비와 하단의 소프트웨어에 해당하는 부분이다. 보드에는 UHF RFID를 연결하여, 각 생산설비의 자동 TRACK IN, TRACK OUT 및 설비상태를 모니터링 한다. 또한 각종 센서 등을 이용하여 보다 손쉽고 정확하게 관련 데이터를 얻을 수 있을 것으로 기대된다.



<그림 1> 임베디드 기반 반도체설비 온라인 시스템 전체 아키텍쳐

<그림 1> 좌측 상단에 보이는 생산설비와 설비에서 진행중인 작업현황에 대한 정보를 얻어서 서버로 전송할 임베디드 PC와의 연결구성은 <그림 2> 와 같이 2대의 설비당 임베디드 PC 한 대로 구성된다.

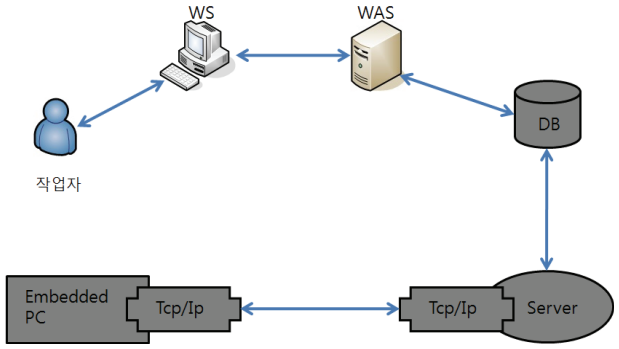


<그림 2> 반도체 생산 설비와 임베디드 PC 구성

이러한 구성을 바탕으로 전체 시스템이 동작한다. 다음 장에서 전체진행사항은 언급이 되겠지만 하드웨어 단에서 이루어지는 과정을 간략하게 살펴보면 Magazine이 설비에 투입될 때 자동으로 RFID 태그를 통해 Track In을 인지하여, 온라인으로 실시간 데이터가 인터페이스되며, 설비 작업을 모두 마치면 Magazine이 설비 밖으로 나올 때 Track Out을 인지하여 온라인으로 데이터가 전송이 된다. 이때 설비1 에 대한 데이터는 임베디드 PC의 1번 포트와 2번 포트를 통해 , 마찬가지로 설비2 에 대한 데이터는 3번 포트와 4번 포트를 통해 액션을 취하고 임베디드 PC는 이벤트가 전송된 포트를 인지하여 그 정보를 서버로 전송할 데이터에 포함시킨다. 이 과정에서 공장의 온라인 시스템과 연계하여, 실시간으로 설비의 작동유무를 체크할 수 있다. 설비쪽에서 임베디드 PC의 LCD를 통해 한눈에 보여지는 데이터를 통해 작업자의 욕구와 작업 능률을 높이며, 웹 어플리케이션을 통해 원격으로도 설비를 관리, 관측 할 수 있다.

3. 디바이스 관리 및 모니터링 시스템 설계

디바이스(생산설비) 모니터링을 위한 데이터 처리 및 사용에 대한 전반적인 구성은 다음의 <그림 3>과 같다.

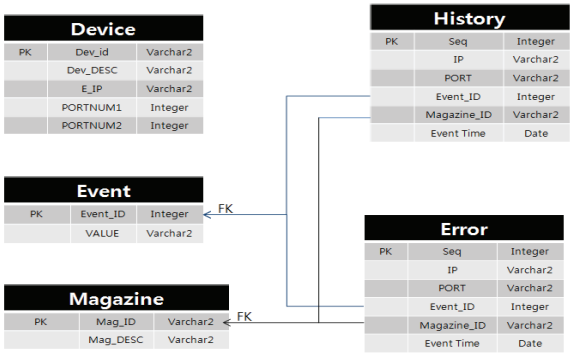


<그림 3> 디바이스 관리 및 모니터링 시스템 구성도

앞서 언급한 것과 같이 임베디드 PC에서는 반도체 작업 공정 중 일어나는 이벤트 상황에 대한 데이터를 서버로 전송한다. 서버에서는 전송받은 데이터를 분석하여 필요한 내용들을 추출해내고 이를 데이터베이스로 저장한다. 작업자(또는 관리자)는 웹 서비스를 통해 웹 어플리케이션 서버로 접속하여 실시간으로 데이터베이스에 저장된 내용들에 대한 검색 및 기준 정보 등록 등의 작업을 수행할 수 있어야 한다. 여기서 기준 정보란 디바이스나 Magazine의 일련번호와 같이 공장에서 취급중인 자원에 대한 정보들을 말한다. 추가적으로 공정과정 중 예외상황이 발생하게 되면 관리자에게 휴대용 단말기 등을 이용해 디바이스 혹은 임베디드 PC에 문제가 있음을 신속히 알려 해당 문제를 조속히 처리할 수 있어야 한다.

3.1 데이터베이스 설계

테이블의 구성은 <그림 4>와 같이 총 5개로 구성된다



<그림 4> 데이터베이스 스키마

Device 테이블은 디바이스의 일련번호와 디바이스와 관련된 정보, 모니터하고 있는 임베디드 PC의 IP 정보, Port 정보 등을 관리하고 디바이스의 일련번호를 기본키로 사용한다. Magazine 테이블은 Magazine의 일련번호와 Magazine에 대한 정보를 관리하고 디바이스 테이블과 마찬가지로 일련번호를 기본키로 사용한다. Event 테이블은 Event ID와 이벤트 상태에 대한 정보를 관리하며 예상 가능한 이벤트 상황의 수가 한정적이기 때문에 시퀀스의 개

넘으로 Event ID를 사용하도록 한다. 이 세 개의 테이블은 기준정보를 담고 있으며 History 테이블은 IP, Port, Event, Magazine에 대한 데이터와 이벤트 발생시간, 시퀀스 넘버를 관리한다. 기본적으로 History는 전체공정에서 발생하는 모든 이벤트 상황에 대한 데이터를 누적/저장하고 정상적인 이벤트가 아닌 예외상황 발생 시에는 History 테이블과 Error 테이블에 동시에 저장한다[1].

3.2 프로토콜 구조 설계

공정 현장에서 일어나는 일련의 처리 현황들이 임베디드 PC를 통해 서버로 넘어간다. 이때 임베디드 PC와 서버는 TCP/IP 소켓 통신을 사용하여 데이터를 주고 받는다. 이 연구에서는 다음의 <그림 5>와 같은 구조로 데이터 통신이 이루어진다.

1byte	1byte	1byte	n byte			2byte
Start Code	명령어 종류	BODY SIZE	BODY			Check SUM
			RES_UID (5 Byte)	EVENT_ID (1Byte)	MGZ_ID (n Byte)	

<그림 5> 패킷 구조

위의 <그림 5> 에 대한 설명을 덧붙이자면 패킷의 총 크기는 가변이며 Magazine ID에 대한 부분을 제외하고 나머지 데이터는 고정적인 값을 갖는다. Start Code는 F0를 디폴트로 하며 패킷의 시작을 의미한다. 명령어 종류는 공정과정에서 발생한 예외상황에 대해 알려주고 BODY SIZE는 RES_UID부터 MGZ_ID까지의 크기를 의미한다. Check Sum은 BODY의 모든 데이터를 합한 값으로 패킷의 이상 유무를 판별하기 위한 데이터이다. 다음의 <그림 6>은 위에서 설명한 패킷의 예를 표현한 것이다.

F0	00	0C	80 7D DF 01 03	01	4D 47 5A 30 30 31	03 60
①	②	③	④	⑤	⑥	⑦

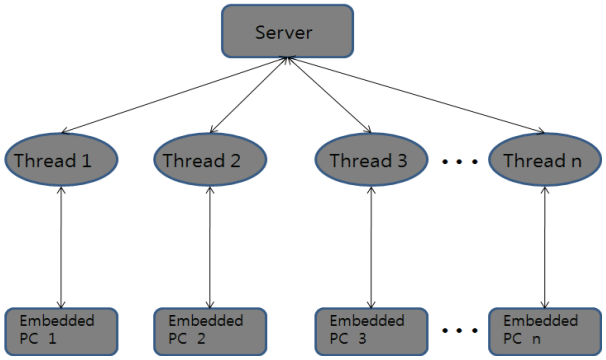
① START CODE [F0] - Default
② 명령어 종류 [00] – Normal [01] – Error
③ BODY SIZE [13]
④ 장비 Unique ID [(IP)168.126.63.1+(PORT)3
⑤ Event ID [1] – Track In
⑥ Magazine ID [MGZ001]
⑦ Check SUM [360] - ④|⑤|⑥ 1Byte Check Sum
80+7D+DF+01+03+01+4D+47+5A+30+30+31 = 360

<그림 6> 패킷의 예

다음 4장에서 패킷분석에 대한 부분을 언급하겠지만 패킷처리과정에 대해 요약하면 위와 같은 구조로 전송된 패킷을 서버에서 각 영역별로 분리하고 Start Code와 Check Sum을 통해 패킷의 이상 유무를 검사한다.

3.3 서버 설계

일반적으로 서버와 클라이언트 간 통신이 이루어질 때는 클라이언트가 서버에 접속요청을 하게 되고 서버에서 승인을 하게 되면 데이터를 주고 받을 수 있도록 설계된다. 하지만 임베디드 PC에서 서버로 접속요청을 할 경우 각각의 임베디드 PC에서 자신의 주소와 서버의 주소를 모두 기억하고 있어야 하는 등의 번거로움이 있다. 이러한 문제점을 해결하기 위해 <그림 7>과 같이 서버에서 멀티스레드 개념을 사용하여 각각의 스레드가 클라이언트가 되어 임베디드 PC에 접속을 요청하고 데이터 통신을 할 수 있도록 한다[4,5].



<그림 7> 서버와 임베디드 PC 간의 통신 개념도

4. 프로토 타입 시스템 개발

본 연구에서 사용될 서버의 기능은 이클립스 환경에서 자바를 사용하여 시스템 개발을 했고, Microsoft 사의 Visual C++ 6.0 환경에서 가상의 임베디드 PC 환경을 구현하여 서버와의 통신 기능 및 데이터 처리과정을 시험해 보았다.

■ 개발환경
- 서버 : Java(이클립스)
- 임베디드 PC : Microsoft Visual C++
- 웹 어플리케이션 : JSP
- 데이터베이스 : Oracle 10g Standard

다음의 <그림 8>은 프로토 타입 통신 테스트를 위한 샘플 데이터이다.

F0	00	0C	80	7D	DF	01	03	01	4D	47	5A	30	30	31	03	60
F0	01	0C	80	7D	DF	02	01	03	00	00	00	00	00	00	01	E2
F0	00	0B	80	7D	DF	03	01	01	4D	47	5A	30	33		03	00
F0	00	0B	80	7D	DF	03	01	01	4D	47	5A	30	33		03	32
F0	00	0C	80	7D	DF	01	04	02	4D	47	5A	30	30	32	03	63
F0	00	0B	80	7D	DF	03	02	02	4D	47	5A	30	34		03	35

<그림 8> 통신테스트를 위한 샘플데이터

일반적으로 패킷을 분석한 결과 Start Code와 Check SUM 데이터에 문제가 없다면 해당 데이터는 명령어 코드에서 확인한 일반 혹은 예외상황에 대한 데이터를 데이터베이스에 저장하도록 처리된다. 예를 들어 <그림 8>의 샘플데이터 2번째 행의 경우 Magazine 테이블에 해당하

는 데이터가 포트오류로 인해 제대로 확인되고 있지 않는 상황이다. 이 경우 데이터베이스 내에 History 테이블과 Error 테이블에 해당 데이터를 저장하도록 처리한다. 3번째 행의 데이터에서는 마지막 부분의 Check Sum 값이 잘못 전송 된 경우이다. 이런 경우 서버에서는 패킷분석을 통해 패킷에 문제가 있음을 감지하고 응답데이터로 NAK를 전송하여 재전송을 요구한다. 또한 공정과정에서 문제가 발생한 것이 아니고 단순히 통신과정에서 문제가 발생했기 때문에 해당 데이터는 데이터베이스에 저장하지 않는다.

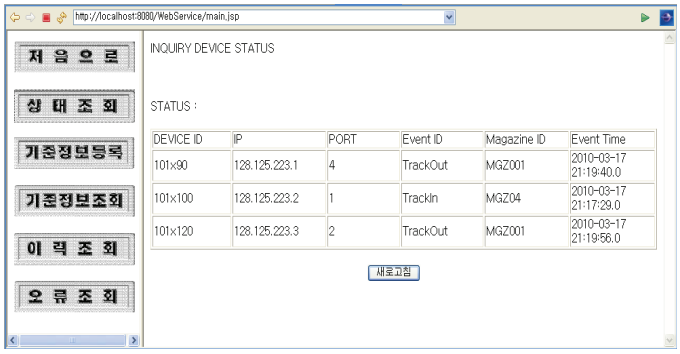
이와 같은 방식으로 통신이 이루어지고 다음의 <그림 9>는 전송한 데이터와 응답 메시지를 콘솔창에서 출력한 화면이다.

<그림 9>가상 임베디드 PC - 서버간 데이터 통신

샘플데이터 부분에서 설명한 것과 같이 3번째 열에서는 NAK로 약속된 '03'을 수신하였고 다시 올바른 데이터를 전송했을 때 ACK로 약속된 '02'를 수신했다[2,3].

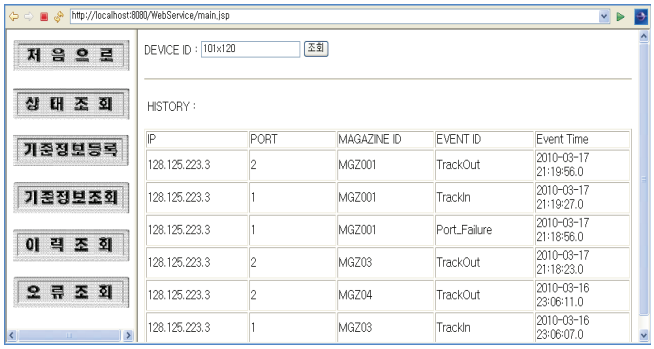
이런 과정을 통해 데이터 통신이 이루어지고 서버에서는 패킷을 영역별로 분리하여 데이터베이스로 저장한다. 저장된 정보는 다음의 <그림 11>,<그림 12>와 같이 웹에서 확인할 수 있다.

<그림 11>은 디바이스 공정 상태를 조회한 결과이다. History 테이블에 누적된 데이터 중 각 디바이스별로 가장 최근정보를 검색하여 디바이스에서 작업 중인 내용과 라인에 투입되어 있는 Magazine에 대한 내용을 확인할 수 있다. Event ID에 표시된 내용 중 Port Failure는 패킷 부분에서 설명한 것과 같이 포트가 오동작을 일으켰음을 의미한다[6].



<그림 11> 디바이스 공정 조회

<그림 12>는 디바이스 일련번호를 사용해 데이터를 검색한 결과화면이다. History 테이블에는 기본적으로 아이피와 포트정보만 저장하고 디바이스 일련번호는 저장하지 않는다. 그렇기 때문에 입력받은 디바이스 일련번호를 사용하여 Device 테이블에서 해당 디바이스의 IP와 Port 정보를 얻게 된다. 그렇게 얻은 IP와 Port 정보에 해당하는 History 테이블내의 데이터를 검색하도록 구현하였다. 이력조회 화면에서는 일반적인 공정데이터 뿐만 아니라 <그림 12>의 세 번째 결과행과 같이 포트 오동작을 일으킨 이력도 함께 확인 할 수 있고 Error 테이블에 저장된 데이터는 메뉴 중 오류조회를 통해 확인 할 수 있다[6].



<그림 12> 공정 이력 조회

5. 결론

본 논문에서는 반도체 생산 공정 장비 관리 및 모니터링 시스템을 설계하고 프로토 타입 시스템을 개발하였다. 공정 과정 중 작업자가 일일이 체크하고 관리해야 하는 부분을 기계적으로 처리함으로써 오류를 줄이고 웹 어플리케이션을 이용하여 처리/누적된 데이터를 실시간으로 확인하여 처리함으로써 생산성을 향상시키며 적은 인원으로 작업이 처리될 수 있도록 하는 유도하는 것이 본 연구개발의 취지이다.

향후에는 통신방식을 HSMS 프로토콜로 대체하여 데이터 통신의 신뢰성 향상을 도모하고 이러한 개발경험 및 기술력을 바탕으로 공정자동화 뿐만 아니라 물류자동화에 이르기까지 효율적인 데이터처리와 DB구축 기술을 적용할 수 있는 시스템을 개발하고자 한다. 이런 과정을 통해 완성된 시스템이 현업에 실제 적용되도록 하는 것이 이 연구의 최종목표이다.

참고문헌

[1] Raghu Ramakrishnan, Johannes Gehrke, Database Management Systems, 3rd Edition, McGraw-Hill Korea, 2008.

[2] 윤성우, TCP/IP 소켓 프로그래밍, FREELEC, 2007.

[3] 백창우, TCP/IP 소켓 프로그래밍, 한빛미디어, 2005.

[4] 김은옥, 박상욱 , JAVA프로그래밍, 삼양미디어, 2007.

[5] Steven Haines, Stephen Potts, Java^{TM}2 기초 플러스, 성안당, 2004.

[6] 황희정, 자바 웹 프로그래밍, 한빛미디어, 2004.