ISSN 2383-6318(Print) / ISSN 2383-6326(Online)
KIISE Transactions on Computing Practices, Vol. 21, No. 10, pp. 645-650, 2015. 10
http://dx.doi.org/10.5626/KTCP.2015.21.10.645

메모리 반도체 검사 장비 인터페이스를 위한 크로스플랫폼 소프트웨어 기술

(CTIS: Cross-platform Tester Interface Software for Memory Semiconductor)

김 동 수 † 강 동 현 † 이 은 석 †† 이 규 성 ††† 엄 영 익 ††††
(Dong Su Kim) (Dong Hyun Kang) (Eun Seok Lee) (Kyu Sung Lee) (Young Ik Eom)

요 약 메모리 반도체 패키지 검사 공정에서 TIS(Tester Interface Software)는 디바이스가 검사 장비에 투입 될 때부터 배출될 때까지 검사 장비가 디바이스 검사를 진행하는데 필요한 모든 소프트웨어 기능을 제공한다. 하지만, 공정에서 사용되는 장비와 장비를 제어하기 위한 컴퓨터 및 운영체제의 종류가 다양하여 동일한 기능을 수행해야 하는 TIS가 테스터 장비마다 독립적으로 개발 및 운영되고 있다. 이는 많은 시간과 비용을 요구할 뿐만 아니라 소프트웨어의 품질에도 많은 영향을 미치고 있으며, 이러한 문제는 추가되는 장비의 종류가 증가할수록 심화될 것이다. 본 논문에서는 이러한 문제를 해결하기 위해 이종 장비와 운영체제에 적용 가능한 CTIS(Cross-platform Tester Interface Software)을 제안한다.

키워드: 메모리 반도체, 패키지 테스트, 크로스플랫폼, 래퍼 기법, Qt 라이브러리

Abstract Tester Interface Software (TIS) provides all software functions that are necessary for a testing device to perform the test process on a memory semiconductor package from the time the device is put into the test equipment until the device is discharged from the equipment. TIS should perform the same work over all types of equipment regardless of their tester models. However, TIS has been developed and managed independently of the tester models because there are various equipment and computer models that are used in the test process. Therefore, more maintenance, time and cost are required for development, which adversely affects the quality of the software, and the problem becomes more serious when the new tester model is introduced. In this paper, we propose the Cross-platform Tester Interface Software (CTIS) framework, which can be integrated and operated on heterogeneous equipment and OSs.

Keywords: memory semiconductor, package test, cross-platform, wrapper method, Qt library

· 이 논문은 2015년도 정부(교육부)의 재원으로 한국연구재단의 기초연구사업 지원을 받아 수행된 것임(NRF-2013R1A 1A2012790)
· 이 논문은 제41회 동계학술발표회에서 '메모리 반도체 검사 장비용 크로스 플랫폼 소프트웨어 개발'의 제목으로 발표된 논문을 확장한 것임

† 학생회원 : 성균관대학교 정보통신대학
dungking82@naver.com
kkangsu@skku.edu
†† 비 회 원 : 삼성전자 설비혁신팀
manido25.lee@samsung.com
††† 비 회 원 : 삼성전자 TEST기술팀
dubig.lee@samsung.com
†††† 종신회원 : 성균관대학교 정보통신대학 교수(Sungkyunkwan Univ.)
yieom@skku.edu
(Corresponding author임)

논문접수 : 2015년 5월 21일
(Received 21 May 2015)
논문수정 : 2015년 7월 2일
(Revised 2 July 2015)
심사완료 : 2015년 7월 8일
(Accepted 8 July 2015)

정보과학회 컴퓨팅의 실제 논문지 제21권 제10호(2015. 10)

1. 서 론

반도체 공정은 크게 세가지 공정으로 구성되어 있다. 첫 번째로 웨이퍼를 제조하는 FAB(Fabrication) 공정, 완성된 웨이퍼를 개별 디바이스로 자르고 리드프레임을 붙여서 패키지를 만드는 조립공정, 마지막으로 제품이 정상적으로 동작하는지 여부를 검사하는 테스트 공정이다. 테스트 공정은 다시 웨이퍼를 검사하는 EDS 테스트 공정과 패키지 칩을 검사하는 패키지 테스트 공정으로 나뉜다.

그림 1과 같이 패키지 테스트에서 사용되는 장비는 테스터(Tester)와 핸들러(Handler)[1] 그리고 테스터를 제어하는 EWS(Engineering Work Station)로 구성된다.

테스터는 디바이스를 검사할 때 전자신호를 발생시켜 패키지에 전달하고, 핸들러는 패키지를 테스트 보드에 삽입하고 검사 완료 후에 패키지를 배출시킨다.

반도체 테스트 공정에서 사용되는 물류 단위는 'LOT'으로서, 1개의 LOT은 수천 개의 디바이스로 이루어져 있다. 테스트 공정은 LOT이 설비에 투입되는 'LOTIN', 디바이스를 검사하는 'TESTING', 검사 완료된 LOT을 배출하는 'LOTEND'로 구성되어 있다. EWS에서 동작하는 TIS는 LOTIN부터 LOTEND 까지 테스터가 디바이스 검사를 진행하는데 필요한 모든 소프트웨어 기능을 제공한다. TIS는 테스터 모델과 상관없이 전 설비에 동일한 기능을 제공하여야 하지만, 현재 공정에서 사용중인 테스터와 EWS의 종류가 다양하여 모델별로 독립적인 개발 및 관리가 이루어지고 있다. 이는 많은 시간과 비용을 필요로 할 뿐 아니라 소프트웨어의 품질에도 많은 영향을 미치고 있으며, 새로운 기종의 테스터가 도입되고 있는 시점에서 더 큰 문제로 부각될 것이다. 본 논문에서는 테스터 모델별로 이원화 되어 있는 TIS를 통합하고, 이종 장비와 운영체제에 적용 가능한 CTIS를 제안한다.

본 논문의 구성은 다음과 같다. 1장의 서론에 이어 2장에서는 현재 운영되고 있는 TIS의 기능과 현황을 살펴본다. 3장에서는 제안한 CTIS의 설계 및 구현 방법에 대하여 설명하고, 4장에서 CTIS의 성능을 테스터 장비의 시간 손실 관점에서 TIS와 비교 및 분석한다. 마지막으로 5장에서 결론을 서술한다.

2. 관련연구

본 장에서는 TIS의 기능을 설명하고 현재 테스터 모델별로 운영되고 있는 현황을 살펴본다.

2.1 TIS의 기능

그림 2와 같이 TIS는 공정기능 라이브러리(Process Function Library), 테스터 시스템 제어 라이브러리(Tester System Control Library), 서버 인터페이스(Server Interface), 사용자 인터페이스(User Interface)로 구성되어 있다. 공정기능 라이브러리는 LOTIN부터 LOTEND까지 진행되는 물류처리, 기준정보 설정, 테스트 프로그램 다운로딩, 데이터 처리, 이상처리, 연속 테스트(Continuous Testing)[2] 지원 등을 수행하는 라이브러리를 제공한다. 테스터 시스템 제어 라이브러리는 테스터 프로세서와 핸들러 파라미터를 설정하고, 테스터와 핸들러 간 GPIB(General Purpose Instruction Bus)[3] 통신을 제어한다. 서버 인터페이스는 장비제어 서버(Equipment Control Server)와 TCP/IP, FTP 통신을 진행하며 물류처리, 기준정보, 이상처리에 필요한 정보를 주고받으며, 테스트 결과 데이터를 전송한다. 마지막으로 유저 인터페이스는 작업자가 물량을 투입, 배출 시에 정보를 입력하고 공정 진행 현황에 대한 정보를 제공한다.

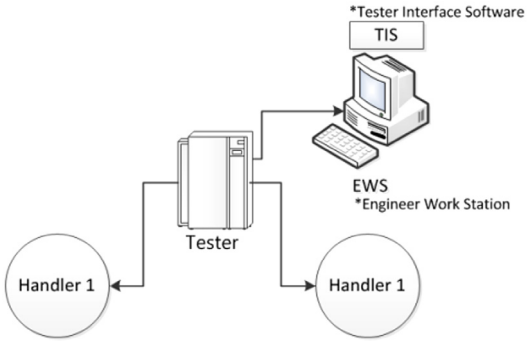


그림 1 패키지 디바이스 테스트 장비의 구성
Fig. 1 Structure of the package device test equipment

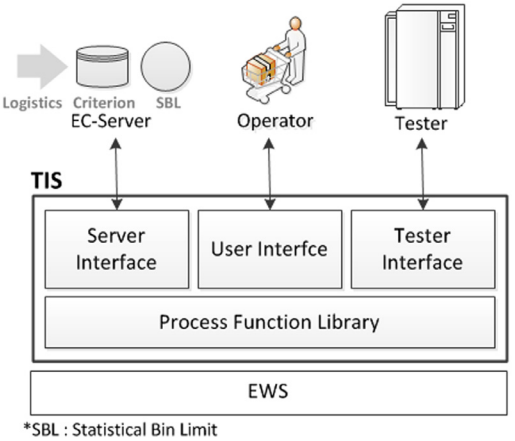


그림 2 TIS 구조
Fig. 2 TIS architecture

2.2 TIS의 현황

표 1과 같이 검사 장비 환경은 EWS의 운영체제에 따라 유닉스, 리눅스, 윈도우즈 계열로 나눌 수 있다. 유닉스 계열 장비는 6종류의 테스터 모델이 있으며, 솔라리스 5.8 외에 2종의 운영체제가 존재한다. 유닉스 계열의 TIS는 특히 장비 업체에서 제공하는 언어인 ATL(Advan Test Language)를 사용하여 테스터 설정, GPIB 통신 제어 및 사용자 인터페이스를 제공한다. 리눅스 계열의 장비는 수세 리눅스 9.2 외에 1종의 운영체제가 있으며, 윈도우즈 계열의 장비는 Windows 2003 외에 3종의 운영체제가 있다. 총 12종의 테스터 모델이 있으며, 각각의 테스터는 모델 별로 다른 종류의 EWS로 제어되기 때문에 TIS 역시 테스터 모델별로 12종의 소프트웨어가 독립적으로 개발, 운영되고 있다.

표 1 테스터, EWS, TIS 현황
Table 1 Status of tester, EWS, and TIS (Unit: Type)

Category		Unix	Linux	Windows	Sum
Tester	Model	6	2	4	12
EWS	OS	3	2	3	8
	Model	6	2	3	11
TIS	Language/Library	C, ATL	C	C++ /MFC	-
	UI	TUI	TUI	GUI	-
	Network	GPIB, TCP/IP, RS232	GPIB, TCP/IP	GPIB	-
	Count	6	2	4	12

*ATL : Advan Test Language
*TUI/GUI : Text User Interface/Graphical User Interface

3. CTIS

CTIS는 크게 공정 기능 라이브러리(Function Library), 시스템 제어 라이브러리(System Control Library), EWS와 핸들러 간 통신 모듈(Handler Communication Module), 사용자 인터페이스, 심볼로 구성되어 있다. 이종 테스터 적용을 위해 시스템제어 라이브러리에 래퍼(Wrapper)[4] 기법을 적용하였고, 이종 운영체제 적용을 위해 Qt 라이브러리[5]를 사용하였다.

3.1 공정기능 라이브러리

그림 3과 같이 공정기능 라이브러리는 테스트 공정의 투입, 1차 검사(Prime Test), 2차 검사(Retest)[6], 배출 과정에서 테스터 모델과 상관없이 동일하게 수행해야 하는 기능들을 동적 라이브러리(Dynamic Library)로 제공함으로써, 사용자 인터페이스와 핸들러 통신 모듈에서 동시에 라이브러리 호출이 가능하도록 한다. 검사 과정과 라이브러리 기능을 기준으로 분류하여 LOTIN, TESTING, LOTEND, COMMON, UTIL의 5개 부문으로 모듈화 되었으며 총 194개의 라이브러리를 제공한다.

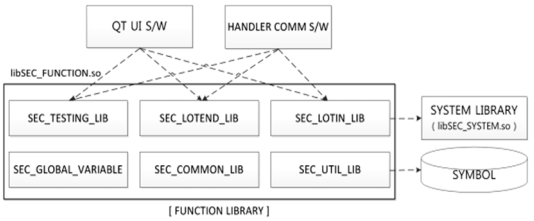


그림 3 공정 기능 라이브러리 구조
Fig. 3 Architecture of the process function library

LOTIN 부문은 멀티 LOT 테스트가 가능한 장비내에서 LOT의 단위인 SIDE를 결정하고, 물류 시스템에 투입된 LOT을 등록한다. 또한 투입된 제품에 해당하는 테스트 기준정보에 맞게 테스터와 핸들러를 설정하고, 테스트 프로그램을 다운로드하여 디바이스 테스트에 필요한 준비를 진행한다. TESTING 부문은 핸들러 이벤트에 해당하는 기능을 수행하며, 주로 테스트 중 발생되는 결과 데이터를 추출하여 처리하는 작업을 진행한다. LOTEND 부문은 테스트가 완료되고 핸들러가 LOT을 배출하는 시점에 수행되며, 물류 시스템에 배출된 LOT 정보를 등록하고 수율, 효율, 품질 데이터를 취합하여 테스트 결과에 대한 이상 여부를 판정한다.

3.2 시스템 제어 라이브러리

그림 4와 같이 시스템 제어 라이브러리는 테스터 프로세서(Tester Processor) 설정, 테스터에 테스트 프로그램 로딩 등의 기능과 테스터 모델별로 요구되는 기능을 제공한다. 테스터 모델별로 시스템을 제어하기 위해 제공되는 라이브러리와 방식이 다르기 때문에, 모델별로 필요한 라이브러리를 개발하고 CTIS가 적용되어 있는 테스터 모델에 맞는 라이브러리를 호출해 주는 래퍼함수[7]를 제공하여 공정기능 라이브러리에서 사용하도록 있도록 하였고, 다른 모듈에서도 호출 가능할 수 있게 동적 라이브러리로 제공된다. 새로운 테스터 모델이 도입 되었을 경우, 신규 모델의 시스템 제어 라이브러리를 추가하면 CTIS가 적용 가능하도록 확장성을 고려하였다.

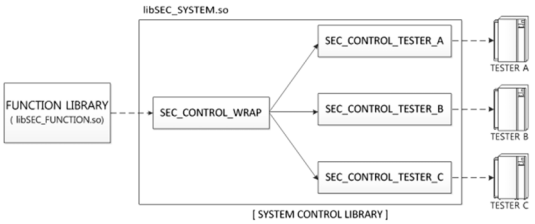


그림 4 시스템 제어 라이브러리 구조
Fig. 4 Architecture of the system control library

3.3 핸들러 통신 모듈

테스터와 핸들러는 GPIB 통신으로 반도체 디바이스 검사를 위한 정보를 주고받는다. TIS는 테스터를 제어함으로써 간접적으로 GPIB 통신에 관여하고 있으며, GPIB 통신 제어를 위해 테스터에서 제공하는 라이브러리는 테스터 모델별로 다르다. 특히 2장에서 언급한 바와 같이 유닉스 계열의 장비는 GPIB 제어부를 ATL로 구현해야 하므로, 크로스플랫폼을 위해 상용 언어를 사용해야 하는 CTIS에서는 적절하지 못하다. 이런 제약을 해소하기 위하여 그림 5와 같이 EWS와 핸들러 간에 허브를 활용한 로컬 네트워크를 구축하였고, TCP/IP 통신을 이용하여, CTIS와 핸들러 간의 직접 통신이 가능하게 되었다. 반도체 테스트 공정에서는 효율적인 테스터의 활용을 위하여, 한 대의 테스터에 2대의 핸들러를 연동시킨다. 그림 6에서 알 수 있듯이 핸들러 통신 모듈은 2대의 핸들러에서 발생하는 메시지를 순차적으로 처리해야 하기 때문에, 각각의 핸들러 메시지를 수신하는 리시브 스레드(Receive Thread) 2개와 메시지를 발생 순서대로 저장하기 위한 메시지 큐(Message Queue), 메시지 큐에서 순차적으로 메시지를 처리하는 워크 스레드(Work Thread)로 구성하였다. 워크 스레드는 메시지를 파싱하고, 해당 이벤트에 대한 공정기능 라이브러리를 호출하여 각각의 이벤트가 잘 처리될 수 있도록 한다.

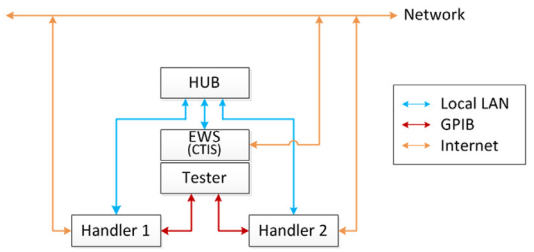


그림 5 네트워크 구성도
Fig. 5 Network diagram

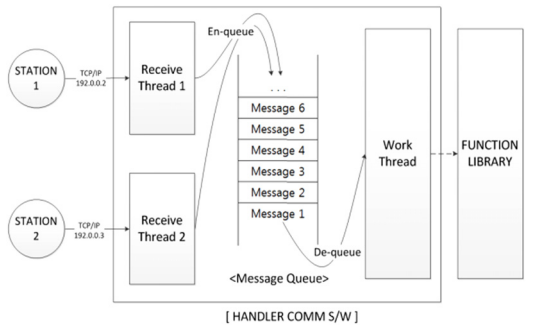


그림 6 핸들러 통신 모듈 구조
Fig. 6 Architecture of the handler communication module

3.4 Qt 라이브러리 및 사용자 인터페이스

Qt는 C++기반의 크로스플랫폼 프레임워크[8]로서, GUI 뿐만 아니라 네트워크, 멀티스레스, SQL 등 광범위한 라이브러리를 제공하여 크로스플랫폼 어플리케이션을 개발하는데 유용하게 사용된다. 그림 7과 같이 CTIS에서는 GUI, 핸들러 통신 모듈의 멀티스레드와 TCP/IP 통신 등에 Qt Library를 활용하여 여러 운영체제에 CTIS를 적용 가능하게 하였다. 특히, 유닉스와 리눅스 계열 장비에서는 기존에 제공되던 TUI와 커맨드 명령어 대신 GUI를 제공하였고, 전체 장비에서 동일한 사용자 인터페이스를 제공함으로써 작업자에게 편의를 제공한다.

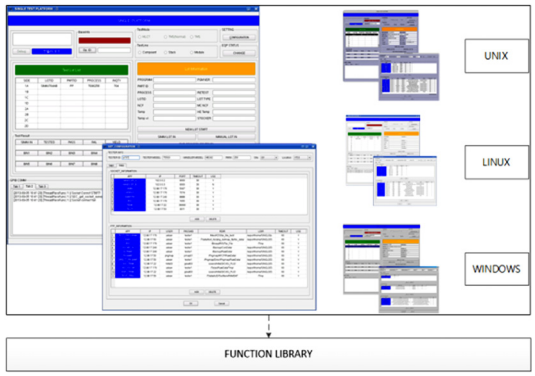


그림 7 Qt 사용자 인터페이스
Fig. 7 Qt user interface

3.5 심볼

CTIS는 테스트 정보 관리를 위해 파일 형태의 심볼을 사용하고, 각 심볼 파일 안에는 XML(Extensible Markup Language)[9] 형태로 단위 정보들을 관리한다. 표 2와 같이 테스트 정보들은 LOT, 장비, 테스트 프로그램, 핸들러 통신 메시지의 4 종류로 구분하고 12개의 파일로 나누어져 있다. 파일 형태이므로, 프로그램 비정상 종료 시에도 심볼 정보는 소멸되지 않고 프로그램

표 2 심볼의 종류
Table 2 Type of Symbols

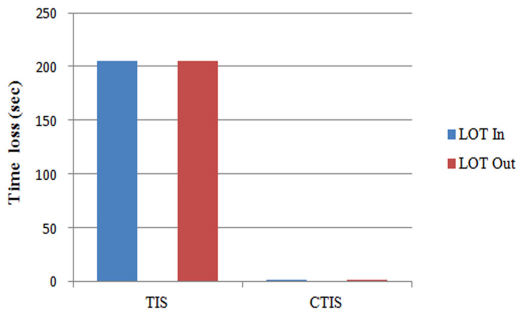
Symbol	I-T	F-N	I-N	Explain
LOT_SIDE	65	8	520	LOT information
SYSTEM	20	1	20	Equipment information
PROGRAM	6	1	6	Test program information
HANDLER_COM_STN	30	2	60	Message of handler communication
Sum	121	12	606	-

*I-T : Number of information types
*F-N : Number of symbol files
*I-N : Number of total informations

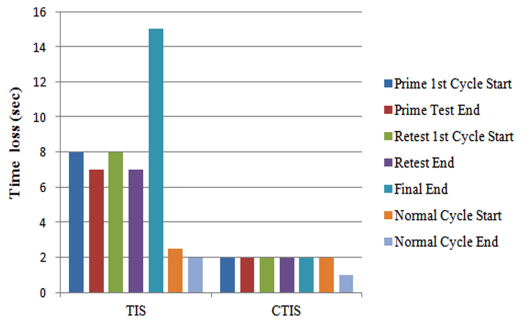
재기동시에 활용 가능하며, 파일 접근 권한 제한으로 외부 접근을 방지한다.

4. 성능평가

성능평가를 위하여 테스트 장비에 기존의 TIS와 CTIS를 각각 적용했을 때, LOT 투입부터 배출까지의 시간 손실(Time Loss)[10]을 비교하였다. 이 때, 시간 손실이란 LOT이 테스터에 투입 될 때부터 배출 될 때까지 시간 중에 테스터가 디바이스를 테스트 하는 시간(Under Test)를 제외한 나머지 시간을 의미한다. 시간 손실은 LOT의 투입과 배출 시에 발생하는 LOT 교체 손실과 테스트 진행 중 발생하는 인덱스 손실이 있다. 시간 손실 관점에서 TIS와 CTIS의 가장 큰 차이점은 핸들러 통신 방식이다. TIS에서 GPIB 통신의 단점은 통신 속도가 느리고, 특히 2 핸들러 구조에서 발생하는 언밸런스로 인한 손실이 크다는 점이다. 이에 반해 CTIS의 TCP/IP 통신은 속도가 빠르고 언밸런스 문제를 피할 수 있다는 장점이 있다. 그림 8에서 알 수 있듯이 CTIS 적용 후 Lot 교체 손실에서 0.1%, 인덱스 손실에서 0.9%의 효율 향상이 있는 것을 확인할 수 있다.



(a) Lot change time loss



(b) Index time loss

그림 8 CTIS와 TIS의 시간 손실 비교

Fig. 8 Comparison of time loss between CTIS and TIS

5. 결 론

본 논문에서는 다양한 테스터 모델과 운영체제로 인해 독립적으로 개발, 운영되었던 TIS를 통합하기 위해 래퍼 기술, Qt 라이브러리, TCP/IP 통신을 활용한 크로스플랫폼 테스터 인터페이스 소프트웨어를 제안하였다. CTIS를 메모리 반도체 테스트 공정에 적용함으로써, 개발, 유지보수를 위한 비용 및 기간을 줄일 수 있을 뿐만 아니라 설비 효율 향상에도 기여하는 것을 확인할 수 있다. 향후, 장치 기반 산업인 반도체의 다른 공정에도 CTIS에서 적용했던 기법을 활용한다면 유사한 문제들을 해결할 수 있을 것이다.

References

[1] K. Toshiba, "Test Handler for Semiconductor Devices," Patent No. US5805472, United States, 1997.

[2] Samsung Electronics Co., Ltd., "Method for Continuous Electrical Testing Throughout Identification of Lot and Test Tray," Patent No. US7633288, United States, 2006.

[3] Samsung Electronics Co., Ltd., "Test System of Semiconductor Device Having a Handler Remote Control and Method of Operating the Same," Patent No. US7230417, United States, 2005.

[4] M. Leeke and A. Jhumka, "An Automated Wrapper-based Approach to the Design of Dependable Software," *Proc. of the 4th International Conference on Dependability*, pp. 43-50, 2011.

[5] S. McIntosh and Y. Kamei, "The Impact of Code Review Coverage and Code Review Participation on Software Quality: A Case Study of the Qt, VTK, and ITK Projects," *Proc. of the 11th Working Conference on Mining Software Repositories*, pp. 192-201, 2014.

[6] C. M. Lee and J. H. Lee, "A Study on Retest Strategy Considering DUT Reliability for Memory Test," *Proc. of the World Congress on Engineering and Computer Science*, Vol. 2, pp. 1068-1073, 2011.

[7] Mark Feldman, "Enterprise Wrappers for Information Assurance," *Proc. of the DARPA Information Survivability Conference and Exposition*, Vol. 2, pp. 120-122, 2003.

[8] A. Ezust and P. Ezust, An Introduction to Design Patterns in C++ with Qt, 2nd Ed., pp. 116. Prentice Hall, USA, 2012.

[9] M. Necasky, I. Mlynkova, J. Klimek, and J. Maly, "When Conceptual Model Meets Grammar: A Dual Approach to XML Data Modeling," *Jounal of Elsevier : Data & Knowledge Engineering*, Vol. 72, pp. 1-30, Feb. 2012.

[10] K. Lee, A. Chung, and S. Kim, "System and Method for Automatically Analyzing and Managing

Loss Factors in Test Process of Semiconductor Integrated Circuit Devices," Patent No. US2003000 5376, United States, 2001.

김 동 수

2008년 한국항공대학교 전자공학과 학사 2008년~현재 삼성전자 Test & Package 센터 선임 연구원. 2014년~현재 성균관대학교 전자전기컴퓨터공학과 석사과정 관심분야는 운영체제, 크로스플랫폼, 시스템 소프트웨어

강 동 현

정보과학회 컴퓨팅의 실제 논문지 제 21 권 제 1 호 참조

이 은 석

2001년 광운대학교 전기공학과 학사. 2001년~현재 삼성전자 Test & Package 센터 책임 연구원. 관심분야는 미래 반도체 라인, 자동화 시스템

이 규 성

1996년 숭실대학교 전자계산학과 학사 1996년~현재 삼성전자 Test & Package 센터 수석 연구원. 관심분야는 자동화 시스템, 운영체제, 컴퓨터 통신

엄 영 익

정보과학회 컴퓨팅의 실제 논문지 제 21 권 제 1 호 참조