대한산업공학회/한국경영과학회 2002 춘계공동학술대회
한국과학기술원(KAIST) 2002년 5월 3일-4일

고수준 필드버스 기반의 클러스터 툴 모듈 통신
Cluster Tool Module Communication Based on a High-level Fieldbus

이진환*, 이태억*, 박정현**
*한국과학기술원 산업공학과 **선문대학교 기계 및 제어공학부

telee@mail.kaist.ac.kr

Abstract

A cluster tool for semiconductor manufacturing is an integrated device that consists of several single wafer processing modules and a wafer transport module based on a robot. The distributed module controllers are integrated by an inter-module communication network and coordinated by a centralized controller, called a cluster tool controller (CTC). Since the CTC monitors and coordinates the distributed complex module controllers for advanced process control, complex communication messaging and services between the CTC and the module controllers are required. A SEMI standard, CTMC (Cluster Tool Module Communication), specifies application-level communication service requirements for inter-module communication. We propose the use of high-level fieldbuses, for instance, PROFIBUS-FMS, for implementing CTMC since the high-level fieldbuses are well suited for complex real-time distributed manufacturing control applications. We present a way of implementing CTMC using PROFIBUS-FMS as the communication enabler. We first propose improvements of a key object of CTMC for material transfer and the part transfer protocol to meet the functional requirements of modern advanced cluster tools. We also discuss mapping objects and services of CTMC to PROFIBUS-FMS communication objects and services. Finally, we explain how to implement the mappings.

1. 서론

클러스터 툴은 가공모듈(PM), 운송모듈(TM), 카세트모듈(CM)등으로 구성된 단일 웨이퍼 가공장비이다(그림 1 참조). TM은 웨이퍼 운반 로봇을 포함하며, 최근에는 웨이퍼 운송작업의 효율성을 위해 양팔 로봇을 장착한다. CM은 웨이퍼 보관도구인 카세트의 로딩/언로딩을 수행하는 곳이다. 또한, 가공할 웨이퍼를 TM을 사용하여 PM에 공급하며, 가공을 마친 웨이퍼를 카세트에 보관한다. PM은 가공챔버, 웨이퍼를 가공위치와 운반위치로 이동하는 척(Chuck)으로 구성된다. 클러스터 툴 내의 웨이퍼의 이동은 모두 TM을 거치며, TM과 TM에 부착되어 있는 부착모듈(AM)간에 이루어진다. 클러스터 툴은 전통적인 반도체 제조장비에 비해 다양한 요구조건을 가지는 웨이퍼를 가공 가능하고, 제조 시간이 짧아 제조 현장에 많이 적용되고 있다(Burggraaf 1995).

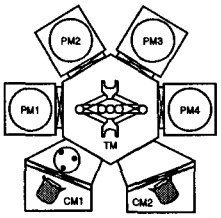

그림 1. 클러스터 툴의 구성

각 모듈별로 가공모듈컨트롤러(PMC), 운송모듈컨트롤러(TMC), 카세트모듈컨트롤러(CMC)와 같은 모듈컨트롤러(MC)가 있으며, 이들은 시스템컨트롤러인 클러스터툴컨트롤러(CTC)에 의해 통제된다(그림 2 참조)(Shin 등 2001). CTC의 주요 수행 작업은 웨이퍼 가공, 모듈간 웨이퍼 운반작업, 카세트 로딩/언로딩, 각종 상태 진단 및 복구 등이다. 최근의 클러스터 툴은 다양한 모듈 구성 및 웨이퍼 가공 순서, 상태 진단 및 복구, 시간 제약 만족, 고수준의 스케줄링 및 제어 등의 기능을 갖고 있다. 이들 기능은 CTC가 각 하위 모듈제어기와 협력하여 수행되므로 CTC와 MC간의 모듈통신(Module Communication)이 중요하다. 모듈통신의 요구사항은 고속 고신뢰성 메시지 전달, 실시간 분산 감시 및 제어, 클라이언트-서버 형식의 분산 통신, 고수준의 메시징 및 서비스, 복잡한 자료 교환, 표준적인 프로토콜 및 서비스 등을 포함한다.

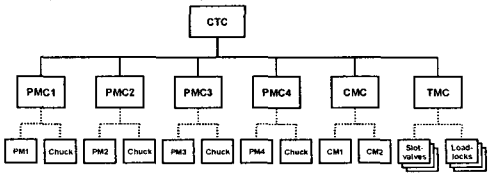

그림 2. 클러스터 툴의 제어 구조

대부분의 클러스터 툴은 모듈제어기로 PC(Personal Computer)를 사용하고 있으며 모듈 통신은 TCP/IP와 Ethernet에 기반하고 있다. 이 통신방법은 보편적이기는 하지만 클러스터 툴의 제어 및 운용을 위한 고수준 기능의 개발이 어렵고, 개발한 기능의 표준화가 어려우며, 매체 접근 방식인 CSMA/CD(Carrier Sense Multiple Access with Collision Detection)의 특성으로 인해 실시간 통신이 보장되지 않는다.

SEMI(Semiconductor Equipment and Materials International)에서는 클러스터 툴의 모듈통신을 위한 응용수준 통신 표준인 CTMC(Cluster Tool Module Communication)를 정의하였다(SEMI 1996). CTMC에는 가공관리, 레서피(Recipe)관리, 예외상황관리, 물류관리 등을 포함하는 응용수준 표준 서비스가 객체모델에 기반하여 정의되어 있다. CTMC는 메시지 전송을 위한 하위 프로토콜을 제한하지는 않지만 일반적으로 SEMI통신 표준인 SECS-I/II 및 HSMS를 사용한다(SEMI 1995). 그러나, 이들 통신방식은 통신 프로토콜 계층이 복잡하며, 실시간 통신을 위해 설계되지 않았고, 객체 지향 개념의 고수준 메세징을 지원하지 않는다. 따라서, CTMC의 구현을 위한 대안이 필요하다. 또한, CTMC 정의 자체도 종래의 물류 처리 방법에 기반하여 정의되었으므로 최근의 양팔 로봇을 장착한 클러스터 툴을 위해서는 개선, 확장이 필요하다.

최근 개방형 필드버스가 공장 자동화에 널리 사용되고 있다. 특히, PROFIBUS-FMS, WorldFIP 의 SubMMS, Foundation Fieldbus 의 FMS(Field Message Specification)와 같은 주요 필드버스는 센서/액추에이터 수준의 고속 저수준 통신서비스 이외에 분산 제어 및 자료교환을 위한 고수준 통신서비스 기능을 같이 제공하고 있다. 또한 이들은 80 년대 말 생산자동화 통신표준으로 제안되었던 MAP (Manufacturing Application Protocol)의 객체개념에 기반한 응용계층 서비스 표준인 MMS (Manufacturing Message Specification)의 통신객체 및 서비스를 단순화한 것이다. 이들 고수준 필드버스는 객체지향 개념에 기반한 표준 서비스를 제공하며, 복잡한 분산 감시 및 제어를 위한 서비스를 제공하며, 실시간 통신을 보장하므로 클러스터 툴의 모듈통신에 적합하다. 이러한 장점에도 불구하고 클러스터툴 모듈간 통신에 고수준 필드버스를 적용하기 위해서는 종래의 PLC(Programmable Logic Controller) 기반 응용과는 달리 적합한 통신객체모델 및 서비스 정의가 요구된다.

본 논문에서는 CTMC 를 구현하기 위해 고수준 필드버스의 하나인 PROFIBUS-FMS 를 통신 프로토콜 및 서비스로 사용할 것을 제안하고 그 방법을 제시한다. 먼저 모듈통신의 요구조건을 분석하고 CTMC 의 통신객체 및 서비스를 분석하고 PROFIBUS-FMS 와 비교한다. 앞에서 설명한 바와 같이 CTMC 는 양팔 로봇을 사용하는 최근 클러스터 툴의 적용시 부적절한 점이 있으므로, 웨이퍼운반을 위한 핵심 객체인 TRJob 객체와 웨이퍼운반 프로토콜을 위한 개선점을 제시한다. 또한 CTMC 의 PROFIBUS-FMS 활용을 위해 CTMC 통신객체 및 서비스를 PROFIBUS-FMS 의 통신객체 및 서비스로 대응하는 방법을 설명한다. 마지막으로 구현 방법을 설명한다.

2. 클러스터 툴 모듈 통신

CTMC 는 기존의 SEMI 표준을 사용하여 제정된 클러스터 툴의 모듈 통신을 위한 표준이다. 통신 서비스는 통신객체간의 상호작용으로 정의되었으며, 공정제어를 위한 PRJob, 물류제어를 위한 TRJob, 단위물류제어를 위한 AtomicTransfer, Recipe 관리를 위한 RecipeExecutor, 이벤트관리를 위한 EventReporting 등의 객체가 관련 서비스와 같이 정의되어 있다.

CTMC 를 사용하는 클러스터 툴 모듈통신 방법에는 CTMC/SECS 와 CTMC/HSMS 가 있다(그림 3 참조). 두 방법 모두 CTMC 서비스를 SECS-II 메시지를 사용하여 전송한다. SECS-II 는 스트림(Stream)과 펑션(Function) 개념을 사용하여 반도체 제조장비와 컨트롤러간의 통신에 사용되는 메시지를 정의한 것이다. CTMC/SECS 는 CTMC 서비스를 담고 있는 SECS-II 메시지를 RS232 를 사용하는 SECS-I 을 사용하여 전송하며 CTMC/HSMS 는 SECS-I 의 RS232 대신 Ethernet 을 사용한다. 그러나, 앞에서 설명한 바와 같이 이들 방법은 클러스터 툴의 모듈통신을 위해서는 부적절하다. 이외에 CORBA(Common Object Request Broker Architecture)와 같은 미들웨어의 사용도 가능하나 제조환경을 위한 실시간 통신을 지원하는 CORBA 의 적용은 아직 미흡하다.

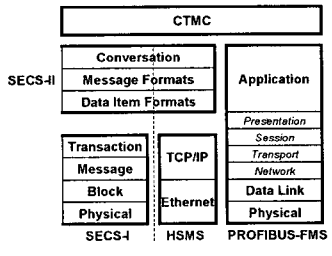

그림 3. 모듈 통신 방법의 프로토콜 구조

CTMC 는 클러스터 툴의 모듈통신을 위한 표준적인 방법이나 그 기능과 장점의 활용을

위해서는 다른 메시지 전송 방법의 모색이 필요하다.

따라서 본 연구에서는 고수준 필드버스의 하나인 PROFIBUS-FMS 를 CTMC 의 메시지 전송 프로토콜 및 서비스로 사용할 것을 제안한다. 설명된 모듈통신 방법의 프로토콜 계층은 그림 3과 같다. CTMC/PROFIBUS-FMS 방법은 OSI 통신 참조 모델에 비추어볼 때, 1 계층, 2 계층, 7 계층의 3 계층만을 포함하여, 다른 프로토콜 계층에 비해 빠르고 매체접근방식이 실시간 통신을 지원하며 객체 지향 개념을 지원한다.

PROFIBUS-FMS 는 VFD(Virtual Field Device), PI(Program Invocation), Domain, Event, Variable 과 같은 일반적인 통신 객체와 관련 서비스를 정의한다(PNO 1991). SimpleVariable 객체는 타 프로그램 언어의 변수와 같으며, Array 객체는 같은 종류의 SimpleVariable 객체를 Record 객체는 다른 종류의 SimpleVariable 객체를 포함 가능하다. VariableList 객체는 이상의 변수객체의 리스트를 포함할 수 있다. Domain 객체는 모든 종류의 통신 객체의 주소를 포함할 수 있으며, 임의의 자료 및 정보를 나타낸다. PI 객체는 일련의 Domain 객체로 구성되며, 일반적인 물리적, 논리적 프로세스를 나타낸다. 각 통신 객체는 관련 서비스를 가지고 있다. 예를 들면, PI 객체는 Start, Stop, Reset, Kill 과 같은 프로세스 제어 서비스를 제공한다. VFD 는 제조 장비를 나타내는 통신객체이며, 이들 모든 통신 객체를 포함한다(그림 4 참조)

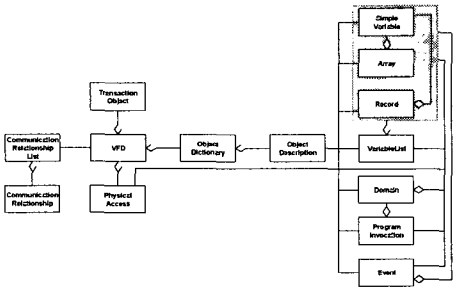

그림 4. PROFIBUS-FMS 객체간 관계

CTMC 와 PROFIBUS-FMS 가 모두 분산 제어를 위한 객체지향개념에 기반한 응용수준의 메시지 표준이지만 CTMC 는 클러스터 툴의 모듈통신을 위한 특정 응용 표준이며, PROFIBUS-FMS 는 응용분야에 제약이 없는 일반적인 통신 객체의 집합이다. 따라서, PROFIBUS-FMS 를 사용하여 CTMC 를 구현하기 위해서는 CTMC 에 정의된 통신객체를 PROFIBUS-FMS 객체로 구현해야 한다. 이 때 일반적으로 하나의 PROFIBUS-FMS 객체로 하나의 CTMC 객체를 표현하기 어려우며, CTMC 객체의 기능을 모두 표현하기 위해서는 일련의 PROFIBUS-FMS 객체가 필요하다. 이것은 MMS 와 PROFIBUS 의 응용분야 표준인 Companion Standard 또는 Profile 을 클러스터 툴 분야에 적용하는 것과 유사하다. 본 논문에서는 CTMC 객체를 PROFIBUS-FMS 객체를 사용하여 대응하는 방법을 제시한다.

3. CTMC 의 분석과 개선

최근 클러스터 툴은 효율적인 물류작업을 위해 양팔 로봇을 탑재하고 있으며, 다양한 웨이퍼 가공순서의 처리, 오류 복구 등의 다양한 요구조건을 만족해야 한다. 그러나, 모듈통신에 사용되는 CTMC 는 일반적인 반도체 제조장비를 위해 설계된 기존의 SEMI 표준에 기반하여 고성능 클러스터 툴의 적용시 부적절한 점이 있다. 본 논문에서는 물류작업관리를 위해 사용되는 TRJob 객체와 이 때 사용되는 프로토콜을 분석하고 개선점을 설명한다.

기존의 반도체 제조장비와 달리 양팔 로봇을 장착한 클러스터 툴은 스왑(Swap)이라는 웨이퍼 운반 방법을 사용한다. 스왑은 TM 이 AM 에 있는 웨이퍼를 언로딩하고 다른 팔위에 있는 웨이퍼를 즉시 해당 AM 에 로딩하는 작업이다. 스왑 작업은 기존의 웨이퍼 운반작업보다 간단하며, 작업 시간을 단축할 수 있다 (Burggraaf 1995).

만일, PM1 에 가공을 마치고 PM2 로의 운반을 기다리는 웨이퍼 001 이 있고, CM1 에 PM1 에서 가공되어야 하는 웨이퍼 002 가 있다면, 웨이퍼 운반작업의 순서는 다음과 같다. 먼저 TM 이 CM1 의 웨이퍼 002 를 집은 후 PM1 으로 이동하여 PM1 의 웨이퍼 001 을 집은 후 다른 팔에 올려져 있는 웨이퍼 002 를 내려놓는 스왑작업을 수행한다. 이상의 작업을 수행하기 위한 메시지 교환 순서와 필요한 객체 구성이 각각 그림 5, 그림 6과 같다.

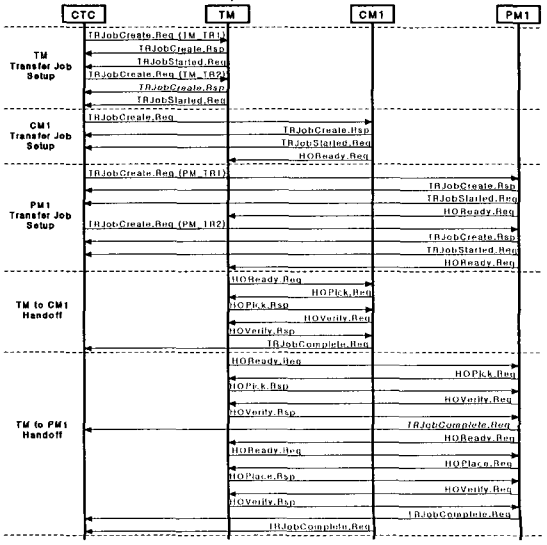

그림 5. 스왑작업을 위한 메시지 교환

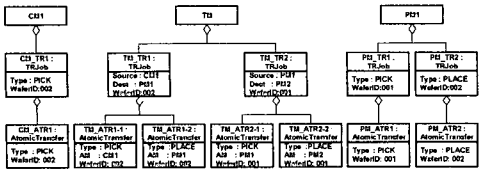

그림 6. 스왑 작업을 위한 TRJob 객체 구성

TRJob 객체를 사용하는 물류작업에는 4 개의 참가자가 필요하다. 이들은 운반작업을 시작하고, 운반작업의 종료신호를 받는 CTC, 웨이퍼가 언로드되는 AM, 웨이퍼를 로드할 AM, 그리고 TM 이다. 스왑작업의 수행을 위해 CTC 는 나머지 3 개의 모듈에 TRJobCreate 메시지를 전송하여(그림 5 참조) TRJob 객체를 생성한다. 예를 들면, PM1 에는 두개의 TRJob 객체인 PM_TR1 과 PM_TR2 를 생성한다(그림 6 참조). 이 때 각 TRJob 객체에는 AtomicTransfer 객체인 PM_ATR1 과 PM_ATR2 가 각각 생성된다. 다음으로 CM1 과 PM1 이 실제 웨이퍼 전송을 위해 TM 과 메시지를 교환한다(그림 5 참조).

AM 에서 웨이퍼의 언로딩과 로딩을 각각 Pick 과 Place 라 한다. CTC 와 AM, TM 간의 TRJob 객체의 생성 및 TRJob 완료를 위한 메시지 교환을 매크로수준(Macro-level) 메시지교환이라하며, TM 과 AM 간에 웨이퍼 전달을 위한 메시지 교환을 마이크로수준(Micro-level) 메시지교환이라 한다. AtomicTransfer 객체는 웨이퍼의 소유권의 전송을 수행한다. 예를 들면 PM1 에서 TM 으로 웨이퍼의 전송을 담당한다.

CTMC 의 TRJob 객체를 사용하면 단일 스왑작업을 위해 많은 TRJob 객체들이 생성되고 복잡한 방식으로 상호 작용한다. 또한 TM 에 생성된 TRJob 객체와 관련된 AtomicTransfer 객체들은 단순히 순차적으로 수행되지 않는다. TM 에서의 TRJob 객체와 AtomicTransfer 객체의 실행순서는 다음과 같다: TM_ATR1-1(TM_TR1 시작), TM_ATR2-1(TM_TR1 일시정지 및 TM_TR2 시작), TM_ATR1-2(TM_TR2 일시정지 및 TM_TR1 재실행 및 실행완료). 또한, PROFIBUS-FMS 를 사용한 TRJob 객체와 AtomicTransfer 객체의 구현도 복잡하다(그림 11 참조).

이러한 복잡성의 원인은 물류작업이 작업물중심 관점으로 정의되었기 때문이다. 작업물 중심 관점에서는 물류작업이 운반될 작업물, 작업물이 놓여있는 장소, 작업물이 운반될 장소, 이상의 3 개의 값으로 표현된다. 예를 들면, 다음과 같은 서비스를 사용하여 운반작업을 요구할 수 있다: TRJobCreate(MID = "Wafer1", TRSourceAmID = "CM", TRDestinationAmID = "PM1")(SEMI 1996). 이러한 작업물중심관점으로 정의된 작업지시가 대부분의 자동 물류 시스템의 운용에 적합하지만, 양팔 로봇을 가진 클러스터 툴의 물류 작업에는 적합하지 않다. 이것은 스왑 작업시 양 팔 중 한 팔이 임시 버퍼로 사용될 수 있기 때문이다. 작업 순서를 예를 들면, 위 예제에서 보통의 물류 작업 순서는 Pick, Place, Pick, Place 이겠지만, 그림 5의 스왑 작업은 Pick, Pick, Place, Pick 의 순서로 진행된다. 이 문제를 해결하기 위해 물류 작업의 정의를 작업물 중심 관점대신 작업중심관점으로 정의한다. 작업중심관점에서는 물류 작업을 수행할 작업 종류, 작업물, 작업 장소의 값, 이상의 3 개의 값을 사용하여 표현한다. 클러스터 툴에서 작업의 종류는 Pick, Place, Swap 으로 구분 가능하다(Shin 2001). 표 1은 작업중심관점을 적용하여 수정한 TRJob 객체 모델을 설명한다.

표 1. TRJob 객체의 수정 모델

	이름	설명
Attribute	ObjType	객체종류: TRJob
	TRJobID	운반작업의 ID
	TRObjType	작업물 종류
	TRObjName	작업물 ID
	TRSwapObjName (C)	스왑될 작업물 ID
	TRPartner	상대 모듈 ID
	TRPortID	운반작업 수행할 모듈의 포트
	TRCommand	작업종류
	TRState	운반작업 상태
Operation	TRJobCreate	TRJob 객체 생성
	TRJobCommand	운반작업 제어 명령
	TRJobAlert	알림: 시작, 종료

TRJob 객체의 수정은 속성의 수정과 AtomicTransfer 객체의 제거로 요약된다. 속성 중에서 TRSwapObjName 은 만일 작업종류가 스왑이라면 스왑작업을 수행할 추가의 작업물을 나타내지만, 작업종류가 스왑이 아니라면 사용할 필요가 없다. 이런 속성을 조건부(Conditional) 속성이라 하며 둥근 괄호안에 C 자를 표시하여 구분한다.

수정된 TRJob 객체를 사용한 스왑작업을 위한 메시지 교환은 그림 7과 같다. 원래의 TRJob 객체를 사용한 메시지 교환과 비교하면, 다음과 같은 차이가 있다. 첫째, AtomicTransfer 객체를 사용하지 않는다 .AtomicTransfer 객체가 운반작업의 자세한 상태를 알아내는데는 유용하나, 객체의 구성과 AtomicTransfer 간의 상호작용이 복잡하므로, 단순성과 효율성을 위해 사용하지 않는다. 둘째, 단일 물류 작업에

필요한 참가자의 수를 4 에서 3 으로 줄였다. 예를 들면, 원래의 스왑작업에서 CTC 는 세 개의 모듈 TM, CM1, PM1 에 여러 개의 TRJob 객체를 생성하였으나, 수정된 작업에서는 동시에 단지 두 개의 TRJob 객체만이 두 모듈에 생성된다. 따라서 메시지 교환 순서 및 상호작용이 간단하며, 효율적이 된다.

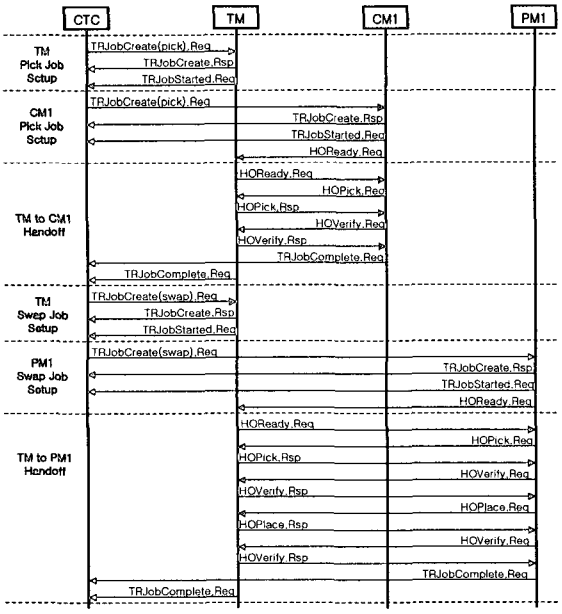

그림 7. 수정된 TRJob 객체를 사용하는 스왑 작업의 메시지 교환 순서

CTMC 를 위한 두번째 향상은 작업물 전달을 위한 메시지 교환에 대한 것이다. Gong 은 (Gong 1998) 이러한 메시지 교환 순서를 작업물 전송 프로토콜(Part Transfer Protocol – PTP)이라고 정의하였다. PTP 는 제조장비의 로딩 또는 언로딩을 위한 것이며, 메시지 교환을 시작하는 주체에 따라 제조장비 시작 제조장비 언로딩, 물류장비 시작 제조장비 언로딩, 제조장비 시작 제조장비 로딩, 물류장비 시작 제조장비 로딩의 4 가지 경우로 구분된다. CTMC 에서는 두번째와 네번째, 즉 물류장비 시작 PTP 를 주로 사용한다. 일반적인 클러스터 툴은 단지 하나의 TM 을 가지고 있으며, AM 에게는 고유한 운반작업의 상대가 결정되어 있다. 따라서, 물류장비 시작 PTP 의 사용이 합당하지만, 최근의 클러스터 툴은 복수의 TM 을 가지는 경우가 있고, 체제시간 제약과 같은 시간 제약을 만족하기 위해서는 작업시작 시간을 조절해야 할 경우가 있다. 이를 위해 물류장비가 물류작업의 시작을 결정하는 것이 아닌 CTC 가 물류작업의 시작을 결정하는 컨트롤러 시작 PTP 를 제안한다. 두 PTP 는 그림 8에 비교되어 있다.

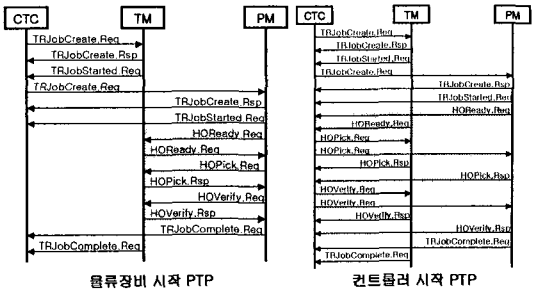

그림 8. PTP 의 비교

PTP 예제는 CTMC 메시지를 사용하였다. 각 메시지 순서에서 음영이 있는 부분이 PTP 에 해당된다. PTP 이전에 CTC 가 TRJobCreate 서비스를 사용하여 TM 과 PM 에 TRJob 객체를 생성한다. 생성된 TRJob 객체들은 PTP 를 사용하여 웨이퍼를 교환한다. 그림 8과 같이 물류장비 시작 PTP 에서는 PM 이 작업물 운반 준비가 되었다는 것을 TM 에게 알리기 위해 HOReady 메시지를 TM 에게 보낸다. PM 이 보낸 HOReady 메시지를 받은 TM 은 해당 작업물 운반 수행 시 다른 HOReady 메시지를 PM 에게 돌려보낸다. 다음으로, PM 이 TM 에게 HOPick 메시지를 보내면 실제 웨이퍼 전달이 이루어지고 웨이퍼 전달 완료 확인을 위한 HOVerify 메시지를 교환한 다음 TRJob 객체를 사용한 운반작업의 종료를 알리는 TRJobComplete 메시지를 CTC 에게 보낸다. 컨트롤러 시작 PTP 에서는 PM 과 TM 이 자신의 물류작업 상대편에게 HOReady 메시지를 보내지 않고, CTC 에게 보낸다. CTC 가 웨이퍼 운반을 결정하면, HOPick 메시지를 PM 과 TM 에게 보낸 다음, 확인 메시지를 보낸다. 이러한 PTP 에서는 CTC 가 모듈간 모든 관련 활동을 조율하게 되며, 모듈간 직접적인 상호작용은 없다. 만일 CTC 의 감시제어기능이 잘 설계되었다면, 컨트롤러 시작 PTP 는 복잡한 웨이퍼 운반 작업을 잘 수행하게 된다.

SEMI 의 물류관리표준에는 컨트롤러 시작 PTP 와 유사한 메시지 순서인 컨트롤러를 거치는 메시지 흐름(Message flow via host)이 정의되어있다. 그러나, 이것은 컨트롤러 시작 PTP 와는 다르다. 이것은 웨이퍼 운반을 수행하는 모듈간에 직접적인 통신 연결이 없을 때, 컨트롤러를 거쳐서 메시지를 보내는 것이다. 메시지 교환은 제조장비 또는 물류장비에 의해 시작된다. 컨트롤러 시작 PTP 는 메시지 교환이 CTC 에 의해 시작된다.

PROFIBUS-FMS 를 CTMC 를 위한 메시지 전송 방법으로 사용하면, 각 모듈의 제어 프로그램 교환과 같은 다른 고기능을 사용 가능하다. 현대적인 반도체 제조 장비는 다양한 웨이퍼 가공 조건을 만족해야 하며, 따라서 웨이퍼 가공 순서 및 가공 변수값의 조절 뿐 아니라

PM 제어 프로그램의 교환도 필요하다. 이 때, 가공 프로그램은 네트워크를 통해 전송되고 실행 가능해야 한다. CTMC 는 이러한 파일의 전송 및 프로세스의 제어를 지원하지 않지만, PROFIBUS-FMS 는 파일 전송 및 프로세스 제어를 Domain 및 PI 객체와 관련 서비스를 사용하여 제공한다.

4. CTMC 객체의 PROFIBUS-FMS 객체로의 대응

수정된 TRJob 객체의 정의를 사용하여 설계한 PMC 의 객체 모델은 그림 9와 같다. PM 은 TM 으로부터 웨이퍼를 받아들여 챔버라 불리우는 장치에서 가공정보를 사용하여 가공한 후, 다시 TM 에 웨이퍼를 전달한다. PMC 의 TRJob 객체는 웨이퍼의 로딩 및 언로딩을 담당한다. 또한 웨이퍼의 정보를 나타내기 위한 Wafer 객체, 웨이퍼의 가공 정보를 나타내는 Recipe 객체, 가공 장치를 나타내는 Device 객체를 포함한다. 만일 웨이퍼의 가공 정보가 바뀌면, 새로운 Recipe 를 전송받는다. 웨이퍼 가공 프로세스는 PRJob 객체에 의해 수행 및 관리된다. EventReporting 객체는 예외적 상황의 발생 또는 기타 중요한 사건의 발생을 감시하며 보고한다.

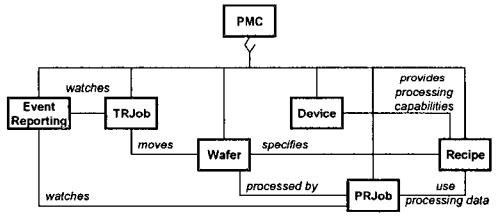

그림 9. PMC 객체 모델

CTMC 객체와 PROFIBUS-FMS 객체의 기본 개념은 매우 틀리다. CTMC 가 기반하고 있는 SEMI 소프트웨어 표준들은 물류관리, 프로세스 관리, Recipe 관리 등 각 응용기능별로 구성되어있다. 이들은 CTC 의 모든 응용 기능을 포함한다. 그러나, PROFIBUS-FMS 객체는 특정 응용을 위해 설계된 것이 아니며 일반적인 제조 시스템의 통신 및 제어를 위한 일반적인 객체들을 정의하였다.

CTMC 응용 프로그램과 PROFIBUS-FMS 통신 기능의 용이한 통합을 위해 CTMC 객체 및 그 기능을 충실히 PROFIBUS-FMS 객체에 포함시키는 것이 필요하다. 일반적으로, 하나의 PROFIBUS-FMS 객체를 사용하여 PRJob, TRJob 과 같은 복잡한 CTMC 객체를 표현하는 것은 어렵다. 그러나, CTMC 의 Recipe 객체와 EventReporting 객체는 각각 PROFIBUS-FMS 의 Domain 객체와 Event 객체로 대응 가능하다. 이 경우 원래 CTMC 객체의 모든 기능을 PROFIBUS-FMS 객체를 대응하여 사용하지는 못하지만 필수적인 기능은 적용된다.

PROFIBUS-FMS 객체들은 이미 정의된 의미를 가지며 그 정의를 확장하기 어렵다. 즉, 각 객체의 속성 및 서비스를 더하거나 수정하기 어렵다. PROFIBUS-FMS 객체의 정의를 확장하는 가장 쉬운 방법의 하나는 SimpleVariable 객체를 사용하여 원하는 속성을 나타내고 이를 원래의 객체에 연결하는 것이다. CTMC 객체의 정의는 속성과 서비스의 두 부분으로 구성된다. 단일 PROFIBUS-FMS 객체로 CTMC 객체 전체를 나타내는 것이 어렵기 때문에 CTMC 객체의 속성들은 PROFIBUS-FMS 의 변수객체로 나타내며 이들은 주요 PROFIBUS-FMS 와 연결하는 방법을 사용한다. CTMC 객체의 서비스들은 대응된 PROFIBUS-FMS 객체의 서비스들을 사용한다.

제안된 객체 대응 지침은 표 2와 같다. CTMC 객체의 단일 속성은 하나의 SimpleVariable 객체로 표현한다. 객체의 집합으로서 CTMC 객체는 Record 객체, VariableList 객체, Domain 객체와 같은 PROFIBUS-FMS 의 컨테이너 객체로 표현한다. 만일, 해당 객체가 통신 응용프로그램의 시작시 생성되어 존속하는 정적객체라면 Record 객체를, 서비스를 사용하여 수시로 생성하여 삭제하는 동적 객체라면 VariableList 객체를 사용하여 나타낸다. 변수 객체는 표현 가능한 자료의 양에 제한이 있으므로, 만일 나타내고자 하는 자료의 양이 많다면, Domain 객체를 사용한다. 이들 객체의 값을 알고 싶다면, 변수 객체를 사용할 경우 간단한 Read, Write 서비스를 이용하여 조회가 가능하나, Domain 객체를 사용하면, 복잡한 Domain 전송 서비스를 사용해야 한다. 따라서, 가능하다면 변수 객체를 사용하는 것이 효율적이다.

표 2. CTMC 와 PROFIBUS-FMS 객체 대응 지침

CTMC	PROFIBUS-FMS
단일 속성	SimpleVariable 객체
객체	Record 객체(정적객체) VariableList 객체(동적객체) Domain(크기가 클 경우)
Process	PI 객체
Recipe	Domain 객체
EventReporting	Event 객체
서비스	대응된 객체의 서비스

표 2의 대응 지침은 간단하지만 PROFIBUS-FMS 의 기능을 충분히 활용하도록 제안한다.

TRJob 또는 PRJob 과 같은 CTMC 객체들은 객체의 정의에 명확히 표현되어 있지는 않지만 각자의 프로세스를 가지고 있다. 이러한 프로세스는 PROFIBUS-FMS 의 PI 객체를 사용하여 대응하면 프로세스를 나타내는 표준적인 방법을 사용하는 것과 동시에 프로세스를 제어하는 표준적인 방법을 사용 가능하다. PI 객체와 임의의 속성을 나타내는 변수 객체를 직접 연결하는 표준적인 방법이 없으므로 그 변수 객체를 포함하는 Domain 객체를 정의하여 PI 객체와 연결한다.

CTMC 객체를 PROFIBUS-FMS 객체와 대응시킬 때, PROFIBUS-FMS 객체의 종류를 나타내기 위해 적절한 접두어를 붙인다. 접두어 SV, R, D, P 는 각각 SimpleVariable, Record, Domain, PI 를 나타낸다. 본 논문에서는 객체의 대응 중 핵심 객체인 TRJob 만을 설명한다.

TRJob 객체를 위한 PROFIBUS-FMS 모델은 그림 10과 같다. 대응 지침과 같이 TRJob 객체의 속성은 SimpleVariable 객체로 대응되고, 모든 SimpleVariable 객체는 Record 객체인 R_TRJob 에 포함된다. R_TRJob 객체는 Domain 객체인 D_TRJob 에 포함되며, 운반 작업을 수행할 PI 객체인 P_TRJob 과 연결된다.

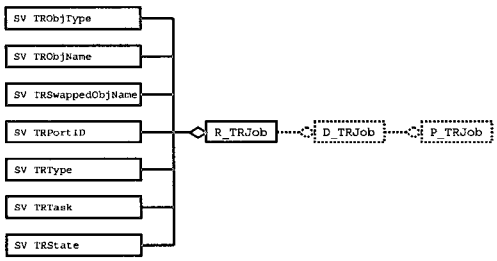

그림 10. TRJob 객체의 PROFIBUS-FMS 객체 모델

그림 10의 TRJob 객체는 수정된 TRJob 객체를 나타낸 것이다. 원래의 TRJob 객체는 최대 2 개의 AtomicTransfer 객체를 가진다. 원 TRJob 객체의 PROFIBUS-FMS 모델은 그림 11과 같다. 이 모델은 수정된 TRJob 모델에 비해 복잡하므로 TRJob 객체의 수정의 필요함을 다시 보여준다.

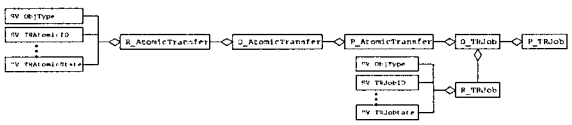

그림 11. 원 TRJob 객체의 PROFIBUS-FMS 모델

5. CTMC 서비스 대응

CTMC 객체를 PROFIBUS-FMS 객체로 대응하면, CTMC 객체의 서비스는 대응된 PROFIBUS-FMS 객체의 서비스로 수행되어야 한다. 웨이퍼 운반을 위한 메시지 서비스는 TRJob 객체의 생성을 위한 TRJobCreate, TRJob 프로세스의 취소를 위한 TRJobCommand, TRJob 객체의 상태변화를 알리기 위한 TRJobAlert 등이 있다. 이상의 CTMC 서비스를 앞에서 대응한 PROFIBUS-FMS 객체의 서비스로 다음 표 3과 같이 대응한다. 예를 들면, CTMC 의 TRJobCreate 서비스를 수행하기 위해 PROFIBUS-FMS 서비스, Write, Create, Start 를 연속으로 수행해야 한다.

표 3. TRJob 객체의 PROFIBUS-FMS 서비스 대응

CTMC 서비스	PROFIBUS-FMS 서비스
TRJobCreate	Wirte Create Start
TRJobCommand	PI 의 Start, Stop, Resume, Reset, Kill 서비스
TRJobAlert	InformationReport
TRJobComplete	InformationReport

CTMC 의 TRJobCreate 서비스를 PROFIBUS-FMS 를 사용하여 구현하기 위해서는 운반 작업의 파라메터를 Write 서비스를 사용하여 전송하고, Create 서비스를 사용하여 PI 객체를 생성한 후, Start 서비스를 사용하여 프로세스를 시작한다. 스왑 작업을 위한 CTMC 메시지를 PROFIBUS-FMS 를 사용한 구현이 그림 12와 같다. 편의상 CM1 을 생략하였으며, 모든 응답(Response)메시지는 나타내지 않았다. 이 메시지 교환은 수정된 TRJob 객체와 컨트롤러 시작 PTP 를 사용하여 원래의 CTMC 메시지와는 다른점이 있다(SEMI 1996).

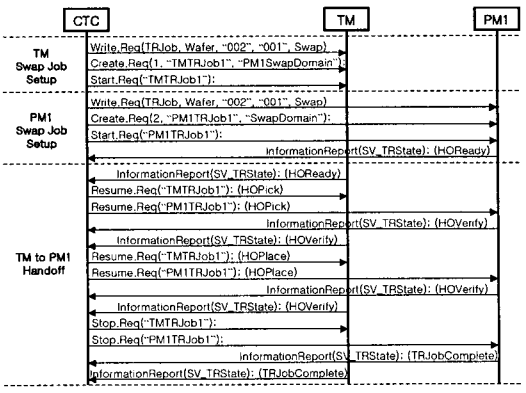

그림 12. 스왑 작업의 수행을 위한 PROFIBUS-FMS 메시지 교환

6. PROFIBUS-FMS 구현

Lee 와 Lee(2001)는 PROFIBUS-FMS 기반의 분산제어구조를 제안하고, 제어 애플리케이션에서 사용하는 메시지를 일련의 통신 서비스로 변환하는 방법인 서비스객체의 개념을 제시하였다. 이 서비스 객체의 개념은 CTMC 서비스를 PROFIBUS-FMS 서비스로 대응하는데 유용하게 적용 가능하다. PROFIBUS-FMS 와 서비스 객체를 사용하는 통신 및 응용 프로그램 구조는 그림 13과 같다.

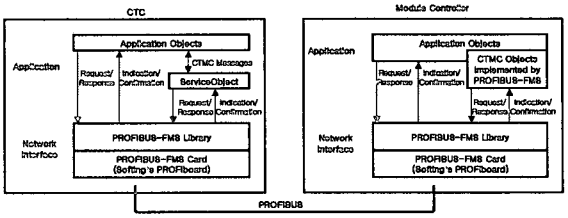

그림 13. PROFIBUS-FMS 기반의 CTMC 구현 구조

대부분의 CTC 및 MC 가 산업용 PC 기반이므로 Softing 사의 PROFIboard 와 같은 PROFIBUS-FMS 의 모든 기능을 지원하는 인터페이스 카드를 사용한다. 그러나, 비록 PROFIBUS-FMS 는 객체지향 개념상에서 설계되었지만, 대부분의 구현 제품은 전통적인 프로그램언어의 함수 수준의 구현에 머물러 있다. 이런 구현은 서비스가 통신 객체의 통합되지 않고, 분리되어 있는 함수 형태로 구현되어 있으며, (복수의) 서비스 관리가 어렵다.

이 문제를 해결하기 위해 Lee 와 Lee(2001)는 서비스 객체(Service Object)를 제시하였다. 이것은 서비스 요청 및 그 내용을 관리하기 위한 객체지향 API(Application Programming Interface)이다. 서비스 객체를 사용하여 상위의 서비스 요청을 일련의 통신 서비스로 변환 가능하다. 이 때 CTMC 서비스는 CTC 안에서 사용되는 내부 메시지가 되고, 서비스 객체는 이 내부 메시지를 일련의 PROFIBUS-FMS 서비스로 변환하여 수행한다. MC 에서는 CTMC 객체를 구현하는 PROFIBUS-FMS 객체가 서비스 요청을 받고, 처리한 후, 응답을 보낸다.

서비스 객체를 사용한 메시지 교환 예제가 그림 14와 같다. TM 이 PM1 에서 스왑작업을 수행하기를 원하는 CTC 내의 응용객체가 CTMC 의 TRJobCreate 서비스를 해당 서비스 객체에게 보내면, 서비스 객체는 이를 앞에서 설명한 대응 규칙에 의해 일련의 PROFIBUS-FMS 서비스로 변환하여 보낸다. 그림 14에서 응용객체와 서비스 객체간의 메시지 교환은 CTC 내부에서 발생하는 것이며, PROFIBUS-FMS 메시지 교환은 서비스 객체와 TM 에서 CTMC 객체를 구현되어 있는 PROFIBUS-FMS 객체간에 발생한다. 따라서, CTMC 를 지원하는 CTC 가 응용객체의 서비스 전달자를 서비스객체로 바꾸면 되어 CTC 의 응용성과 이식성이 개선된다.

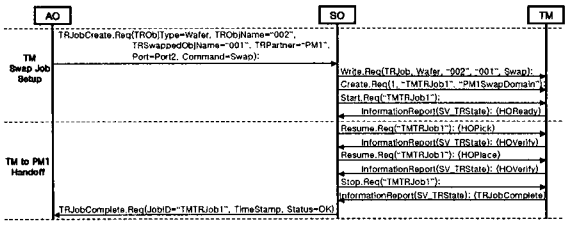

그림 14. 서비스 객체를 사용한 메시징 예

7. 결론

본 논문에서는 클러스터 툴 내의 CTC 와 MC 간의 통신인 모듈통신을 위한 표준인 CTMC 를 고수준 필드버스인 PROFIBUS-FMS 를 사용하여 구현하는 방법을 제안하였다. 최근의 양팔 로봇을 장착한 클러스터 툴을 위해 CTMC 의 TRJob 객체 및 PTP 를 개선하였다. 또한 CTMC 객체를 PROFIBUS-FMS 객체를 사용한 구현을 위해 대응 지침을 제안하였고, 해당 서비스의 대응 방법을 설명하였다. 제안된 방법은 PROFIBUS-FMS 를 클러스터 툴에 적용시 참조모델로 활용 가능하다.

참고 문헌

[1] Burggraaf, P., 1995, Coping with the high cost of wafer fabs. *Semiconductor International*, 18(3), 45-50.

[2] Gong, D. -C., 1998, An automated flow line control system design. *Computer Integrated Manufacturing Systems*, 11(3), 207-215.

[3] Lee, T. -E., and Lee, J. -H., 2001, An application design and development framework for high-level fieldbus based distributed control applications. *submitted to International Journal of Computer Integrated Manufacturing.*

[4] PNO, 1991, DIN19245 PROFIBUS Standard Part 1, 2.

[5] SEMI, 1995, SEMI E38.1-95, Communications Environment HSMS/SECS-II for Cluster Tool Module Communications.

[6] SEMI, 1996, SEMI E38-1296, Cluster Tool Module Communication (CTMC).

[7] Shin, Y. -H., Lee, T. -E., Kim, J. -H., and Lee, H. -Y., 2001, Modeling and implementation of a real-time scheduler and controller for dual-armed cluster tools. *Computers in Industry*, 45, 13-27.