객체지향적 생산 시뮬레이터의 개발

김 종 수*

Development of Object Oriented Manufacturing Simulator

Jong-Soo Kim*

Abstract

This paper concerns the issues involved in development and use of manufacturing simulation software using the Object Oriented Programming System(OOPS) in a computer integrated manufacturing environment, with particular emphasis on large scale integrated circuit fabrication. We introduce OOPS and compare it with existing simulation packages as well as general purpose languages. Our implementation of OOPS shows numerous advantages over the other programming systems. Based on our experience, guidelines for developing manufacturing simulation systems in OOPS are discussed.

1. 서 론

여러해 동안 버클리대학(U.C. Berkeley)의 공학시스템연구센터(Engineering Systems Research Center)의 연구원들은 캘리포니아주의 산타클라라(Santa Clara)에 있는 인텔(Intel)사의 반도체 생산공정의 생산계획 및 관리를 위한 새로운 방법을 연구하였다.

마이크로(MICRO-Microelectronics Innovation and Computer Research Opportunities)로 명명된 이 연구 프로젝트의 내용은 크게 두가지로 나누어져서 첫째는 Dynamic RAM의 중기 생산계획(Corporate-Level Production Plan)의 수립을 위한 선형계획 모델을 동적 생산함수(Dynamic Production Function)를 이용하여 만들고 이를 인텔사의 공장에 적용하는 것이다[5, 8].

두번째는 Bi-Polar, NMOS와 CMOS를 만드는(Fabricate) 공정에 있어서 투입에서 제품이 완성될 때까지의 시간(Cycle Time, Throughput Time)이 길어질수록 수율(Yield)이 줄어든다는 점을 고려한 새로운 투입 제어방법(Release Control Policy)의 개발에 그 목적이 있었다[3, 13].

프로젝트 수행중 C 언어를 이용하여, Bi-Polar, NMOS와 CMOS를 만드는 공정을 컴퓨터 모델화

* 한양대학교 산업공학과

하여 시뮬레이션(Simulation)하는 과정에서, 사용된 시뮬레이터가 가지는 아래와 같은 문제점들로 인하여 그 효율성에 의문을 가지게 되었다(본 연구의 대상이 된 시스템들은 워크스테이션의 수는 18-41개이며, 가공이 필요한 작업의 수는 24-182스텝(Step)으로 구성되어 있다).

즉, 주어진 시스템의 모형화는 C와 같은 기존의 언어로 프로그램된 시뮬레이터(Simulator)가 적합하지만 변형된 시스템의 시뮬레이션에는 많은 어려움이 따른다는 점이다. 모형화된 시스템에 새로운 작업배정방법(Dispatching Rule)이나 투입제어방법을 추가하거나, 또는 워크스테이션(Workstation)의 특성을 변형하거나(예로써 동일 병렬시스템(Identical Parallel Machine System)의 일양 병렬시스템(Uniform Parallel Machine System)으로의 변형을 들 수 있다.) 제품의 흐름경로를 변경시킨 시스템을 시뮬레이션 할 때는 이미 만들어진 시뮬레이터에 많은 변경이 필요하게 되며, 이를 위해 많은 노력과 시간이 소요되면서도 변형된 부분과 기존의 모형과의 사이에 많은 문제점이 발생하게 된다는 것이다. 이러한 문제점들을 해결하기 위하여 프로젝트에 참가한 연구진들은 다양한 형태의 반도체 생산 시스템을 쉽게 시뮬레이션 할 수 있는 시뮬레이터(BLOCS/M(Berkeley Library of Software Objects for Simulation and Control/Manufacturing)이라 명명됨)를 객체지향적 프로그램 언어(Objective Oriented Programming(OOP) Language)를 사용하여 개발하였으며, 본 논문에서는 이러한 시뮬레이터를 개발하는 과정에서 얻어진 경험을 바탕으로 객체지향적 시뮬레이터가 가지는 특징들을 설명하고 이러한 특징들을 새로운 시뮬레이터의 개발 및 현장적용에 응용하는 방법에 대해 논하고자 한다. 시뮬레이터의 모형화 대상이 된 반도체 생산시스템이 주위에서 흔히 볼 수 있는 다단계 자원제약적 생산 시스템이라는 점에서 본 논문의 결과는 이러한 특징을 지닌 다른 품목의 생산시스템에도 적용될 수 있을 것이다[8].

2. 시뮬레이터의 개발

컴퓨터 소프트웨어 분야 및 컴퓨터 언어의 다양성과 사용언어가 시뮬레이터의 기능(Functionality)에 미치는 영향을 고려할 때 시뮬레이터에 사용될 언어는 신중히 선택하여야 한다. 개발되어지는 시뮬레이터에 사용할 언어의 선택을 위해 가용한 컴퓨터 언어 및 시뮬레이션 패키지들을 분석한 결과 다음과 같이 구분되어졌다.

2-1. 생산시스템용 시뮬레이션 패키지

이러한 시뮬레이션 패키지들은 생산 시스템을 모형화하기 용이하도록 개발되었다. MAP/1[16], Simfactory[7], Xcell[6] 등이 그 좋은 예이다. 이러한 패키지들은 프로그래머가 아닌 사람들도 쉽게 사용할 수 있도록 고안되어 있고, 적은 노력으로도 실제 적용이 가능한 모델을 만들 수 있는 장점이 있다. 그러나, 일반적으로 새로운 통제방법(Control Policy)을 추가하는 등의 수정이 불가능하며 새로운 시스템에의 유연성(Flexibility)이 부족하다는 단점이 있다.

2-2. 범용 시뮬레이션 언어/패키지

GPSS[14], SLAM[10], SIMSCRIPT[2], SIMAN[11] 등과 같은 것들이 이 범주에 속하며, 이들은 다양한 시스템을 쉽게 모형화 할 수 있다는 장점을 가지고 있다. 이러한 시뮬레이션 언어들은 포트란(FORTRAN)을 사용한 서브루틴(Subroutine)과 제어문들로 구성되어 있으며 이로 인해 모형화에 필요한 노력을 절감할 수 있다. 그러나 이러한 시뮬레이션 언어들은 전형적이고 간단한

시스템의 모형화를 주대상으로 하므로 큰 규모의 시스템에는 부적합하다는 단점이 있다.

또한 공정과 물류의 이동(Process and Material Flow)의 모형화와 분석을 위해 만들어졌기 때문에 컴퓨터 통합 생산시스템과 같은 복잡한 시스템을 모형화 하는데는 어려운 점들이 있다[17]. 이러한 언어들을 바탕으로 한 시뮬레이션 모형의 단점에 관해서는 Shannon 등[15]의 연구를 참조할 수 있다.

2-3. 범용 언어

FORTRAN은 효율적인 수치계산능력으로 인하여 1970년대에 시뮬레이션을 수행하는데 가장 보편적으로 많이 이용되었다. 또한 근래에는 PASCAL과 C를 이용한 모형화도 늘어나고 있다. 이러한 범용 언어들은 어떠한 형태의 시스템도 모형화가 가능하다는 장점이 있는 반면에 이와 연관된 여러가지 제약요소도 함께 가지고 있다. 예를 들어 서론에서 소개한 C로 프로그램된 시뮬레이터는 6000 줄 이상의 방대한 크기이지만 대상으로 하는 시스템의 극히 일부만을 모형화 할 수 있었으며, 이후 모형의 부분적 확장을 위하여 다시 2000 줄 이상의 프로그램을 추가하게 되었으며 이를 위해서 원프로그램을 이해하기 위한 많은 노력이 필요하게 되었다[13]. 이러한 경험으로 인하여 범용언어로 만들어진 프로그램의 확장성과 유지에 관련된 문제의 심각성을 인식하게 되었다.

2-4. 객체 지향적 프로그래밍 (OOP) 언어

오늘날 사용, 연구되고 있는 객체지향적 프로그래밍 언어는 Smalltalk[4]에서 그 유래를 찾을 수 있다. 일반적으로 객체지향적 프로그래밍 시스템의 두가지 중요한 요소는 객체(Object)와 메세지(Message)이다[9]. 객체는 데이타와 객체가 할 수 있는 동작의 집합이며, 각 객체는 다른 객체가 직접적으로 변형할 수 없는 고유의 경우변수(Instance Variable)에 필요한 데이터를 저장하고 있다. 객체들의 동작들은 메소드(Method)라고 하는 알려진 절차(Procedure)에 의해 정의된다. 결과적으로 객체란 개념은 데이타와 절차들의 속성들로 구성되는 것이다. 위에서 설명된 것처럼 관계된 모든 데이타를 한 모듈(객체)내의 경우변수에 국한시키고 그 접근을 일련의 미리 결정된 절차에 의해서만 변형되게 하는 형태를 캡슐화(Encapsulation)라고 부른다. 이러한 각 객체들은 상호간에 메세지의 전송에 의해 다른 객체와 서로 연락하는, 즉 한 객체가 메세지를 받으면, 그 객체는 그에 상응하는 메소드로 반응하게 되며 이 메소드에 의해 정해진대로 다른 객체들에게 메세지를 전달하게 되며 이러한 메세지의 전송들에 의해 프로그램의 수행이 이루어진다. 이 경우 메세지를 교환하는 것은 한가지 중요한 차이점을 제외하고는 포트란이나 C와 같은 언어들에서 사용되는 함수(Function) 또는 서브루틴과 흡사하다. 즉, 이러한 언어들은 미리 결정된 절차를 불러내지만 객체지향적 언어에서는 불려질 절차(즉 메소드)에 관한 결정은 각 객체들에게 맡겨지게 된다.

지금까지 소개된 다양한 소프트웨어 언어 및 패키지들의 특징을 검토한 결과, 연구진들은 개발될 시뮬레이터의 유연성(Flexibility)과 재사용성(Reusability)을 고려할 때 객체지향적 언어가 가장 적합하다는 결론에 이르렀다.

3. 모형의 설명

시뮬레이터를 개발하기 위하여 우선 다양한 형태의 반도체 생산시스템이 공통적으로 가지는 특징들과 개개의 시스템의 특이성에 대한 분석작업이 선행되어졌다. 그 과정은 시뮬레이터의 전체적인 메카니즘 뿐만 아니라, 객체와 정보의 흐름, 그리고

각 객체들에 의하여 수행될 작업들의 확인을 포함한다. 이러한 분석결과, 개발되는 시뮬레이터가 실제 현장의 시스템을 모형화하기 위해서는 다음과 같은 요건을 갖추어야 하는 것으로 결론지어졌다.

- 다수의 제품을 생산하며
- 동적이며, 기지의 확률분포에 따라 작업이 도착하며
- 제품의 대체경로가 존재하고
- 동일한 능력을 갖는 대체 워크스테이션(Alternative Workstation)이 있으며
- 각 기계고장은 기지의 분포에 따르며
- 다수의 동일한 또는 일양 병렬기계(Parallel Identical or Uniform Machine)들로 하나의 워크스테이션이 구성되며
- 기계와 노동력이 고려대상이 되는 제한된 자원이며
- 워크스테이션간의 운반시간에 대한 분포는 알려져 있고
- 투입제어(Release Control) 및 작업배정(Dispatching), 그리고 경로선정(Route Selection)에 대한 의사결정이 되어야 한다는 점이다.

시스템이 위와 같이 분석된 후 이러한 시스템을 모형화하기 위한 객체들을 확인하고, 각 객체들 내부에서 일어날 수 있는 가능한 모든 사건들을 파악하여 메소드로 전환(Mapping) 하게 된다. 이렇게 개발되어진 객체들은 모형화할 시스템이 주어지면 필요한 객체들만 뽑아서 시뮬레이터를 구성하는 일종의 소프트웨어 라이브러리(Software or Object Library)의 역할을 하게 된다. 이 단계에서 본 연구를 위한 프로그래밍 언어가 객체지향적 언어 중에서 선택되었다.

후보로 나타난 Smalltalk, C++, Objective-C 중에서 메뉴얼의 구성, 개발될 객체들의 종류, 대형으로부터 개인용에 이르는 다양한 컴퓨터 시스템에 대한 지원 가능성을 고려하여 Objective-C가 프로그램을 위한 언어로 선택되었다. 결국, 객체들의 프로그래밍은 Unix(BSD 4.3) 환경하의 Vax Station의 Micro-Vax 시스템에서 이루어졌다.

4. 시뮬레이터의 실행

시뮬레이션을 할 시스템이 주어지면 주어진 실제 생산시스템을 분석하여 필요한 객체들을 파악하게 되며, 각 객체들에 필요한 특성을 분석하게 된다. 이 단계에서 소프트웨어 라이브러리에 있는 객체들 중에서 필요한 객체들만을 선택하고 필요시 기존의 각 객체에 추가로 요구되는 특이성을 부여하기 위하여 메소드를 첨가하는 수정작업을 하게 되며, 이후 대상의 시스템과 준비된 객체들로 구성된 컴퓨터 모형이 동일한 것인지를 확인하게 된다. 이와 같은 작업이 완료된 후 객체들의 실행을 위해서 이들을 연결시키는 주프로그램(Main Program)을 준비한다. 경험을 통해서 볼 때 본 연구의 대상 시스템을 위한 주 프로그램들은 그 크기가 100 줄 이하로 구성되어 있다.

컴파일을 통하여 주프로그램에서 지정한 모든 객체들이 포함되었는지를 Objective-C 컴파일러가 메세지 테이블을 통하여 검토하게 되며, 성공적인 컴파일 후에 시뮬레이터는 정해진 객체에 메소드에 의한 메세지를 흘려 보내기 시작함으로써 실행될 수 있는데 메세지를 받은 객체(들은)는 이번에는 메소드로 정해진 객체(들)에게 다른 메세지들을 보내게 되며, 이 메세지의 흐름은 시뮬레이션이 종료될 때까지 계속된다. 고로 객체들 사이의 메세지 흐름과 객체 내부의 메소드들이 객체지향적 시뮬레이터의 두가지 주 요소이고, 다른 언어들과 구분되어지는 특성이다.

본 시뮬레이션 시스템에 대한 투입자료로서 투입제어, 작업배정, 경로선정 규칙 뿐만 아니라 워크스테이션에서의 기계의 수, 매개변수의 분포, 시뮬레이션 수행기간과 같은 매개변수의 시방도

요구된다. 수행의 결과로써 평균 대기시간, 생산시간(Throughput Time), 워크스테이션의 효율, 각 워크스테이션에서의 시간에 따른 재공품의 양을 포함한 중요한 시스템 정보를 터미널 또는 사용자의 주문에 따라 보고서로 만들 수도 있다. 시뮬레이션이 실행되는 동안 시스템의 현황을 에니메이션을 이용한 그래픽 화면으로도 얻을 수 있다(예로서 워크스테이션의 재공품 양의 시간에 따른 변화를 들 수 있다).

그래픽 출력을 위해서는 TEK4207 또는 MicroVAX 워크스테이션 터미널과 같은 그래픽 터미널을 사용해야 한다. 개발된 시뮬레이터의 실행을 통하여 객체지향적 언어를 이용한 시뮬레이터가 많은 매력적인 요소를 가지고 있음을 알 수 있었으며, 주요한 장점들을 열거하면 다음과 같다.

(1) 실세계 사건들의 표현

객체에 직접 영향을 주는 데이타와 절차를 한 장소에 묶음으로써, 객체지향적 언어시스템은 실제 객체들과 자연스런 일대일 대응을 가능하게 한다. 예를 들어, 제조 시스템에서는 기계, 작업자, 경로 등과 같은 실제 객체에 직접 대응되는 소프트웨어 객체들이 존재하게 되며 객체들의 이러한 특징을 이용함으로써 시뮬레이터를 보다 손쉽게 디자인하고 프로그램 할 수 있다.

(2) 재사용성(Reusability)

한번 개발된 모듈들은 다른 객체들에 독립적이기 때문에 다른 변형된 시스템에서도 이용이 가능하게 된다. 그 외에, 객체의 특징중 하나인 상속성(Inheritance)을 이용하여 세부적 특징이 다른 객체들도 기존의 객체를 근거하여 만들 수 있다. 더우기 시뮬레이션 수행중의 메세지의 흐름은 그것이 불러내는 객체의 메소드들과는 독립적이므로 일반 컴퓨터 언어의 서브루틴 라이브러리의 재사용에 발생하는 제약조건없이 재사용성을 가질 수 있으며, 결과적으로 객체지향적 언어시스템의 객체들은 독립적으로 개발 또는 변형되어져서 새로운 시스템의 시뮬레이션에 재사용될 수 있는 것이다.

(3) 단순성

객체들이 독립적이므로 이를 이용하는 시뮬레이터가 보다 단순해질 수 있다. 시스템의 복잡도는 시스템을 개발하는데 대한 노력을 지수분포적으로 증가시키게 되므로 단순성은 설계, 프로그래밍, 테스트, 유지 등을 포함하는 개발의 모든 단계를 보다 용이하게 한다.

(4) 적응성(Adaptability)

이것은 기술한 다른 장점들과 밀접한 관련이 있다. 메세지를 통한 객체들과 관계들의 집합은 실제의 물리적 시스템의 움직임을 모방(Emulation)한다. 어떤 시스템에 대해서 어떤 모듈이 필요하고 그것들이 어떤 속성들을 가지고 있어야 하는지를 쉽게 개념화 할 수 있으며, 각 모듈은 모델화하려는 다양한 실제 시스템을 위하여 독립적으로 수정되고 완전한 시뮬레이터로 결합될 수 있다.

(5) 프로그래머 생산성

객체지향적 언어로 프로그래밍하는 동안에 모델화되어야 하는 시스템의 객체들의 계층(Class)에 요구되는 가장 공통적인 특성/동작들만 프로그램화함으로써 소프트웨어 객체를 만든다. 이것을 흔히 계층 객체(Class Object)라고 한다. 이 계층 객체(지금은 상위계층(Superclass))의 하위계층으로 선언되는 어떤 다른 객체는 그 상위계층으로부터 모든 데이터와 메소드들을 물려받는다.(이를 상속성이라 부른다.) 예를 들어서, 전기모터와 같은 것은 전기에너지를 기계적 에너지로 바꾸는 객체를 나타내는 객체들의 계층이라 말할 수 있다. 다음으로 '직류 모터'를 '전기 모터'의 하위계층(Subclass)이라고 선언할 수 있다. 직류 모터를 선언할 때 일반적인 전기 모터에 의해 나타내어지는 속성들을 묘사할 필요는 없다. 단지 전기 모터에 존재하지 않는 직류 모터만의 특수성이나 부가적인 기

능만을 묘사하면 된다. 이 상속성의 개념은 일반적 특성을 위하여 어떤 객체를 다시 프로그램해야 하는 필요성을 없게 만들며 기존의 객체에 특이성만을 첨가함으로써 원하는 객체를 만들 수 있게 한다.

위와 같은 장점들이 있는 반면 객체지향적 시뮬레이터의 약점도 발견되었다. 그 중 가장 부정적 영향을 미치는 것이 수행시간의 효율성이다. 시뮬레이션되는 시스템의 복잡도가 증가함에 따라 시뮬레이터를 실행하는데 요구되는 시간은 C 언어로 쓰여진 동등한 시뮬레이터의 수행시간이 선형으로 증가하는데 비하여 지수함수적으로 증가하는 것으로 관찰되었다.

5. 결 론

본 논문에서는 객체지향적 언어로 개발된 시뮬레이터가 소프트웨어 라이브러리로 사용됨으로써 기존 시뮬레이터들에 비해 단순성, 재사용성, 적응성 등의 측면에서 많은 장점을 가지고 있음을 살펴보았다. 이러한 장점들은 컴퓨터 통합 생산시스템(Computer Integrated Manufacturing System)과 인공지능(Artificial Intelligent) 영역을 포함하는 여러 연구분야에 응용될 수 있을 것이다[1, 12, 15, 17]. 앞으로의 활발한 응용을 위해서는 다음과 같은 점들에 대한 심도깊은 연구개발이 필요할 것이다.

5-1. 재사용성에 대한 심화된 검토

우리의 라이브러리에 있는 객체들은 서로 다른 형태(Configuration)를 가지고 있는 시스템들을 대상으로 한 시뮬레이션에 다양하게 쓰여질 수 있음을 보여주었다. 이 말은 개발된 시뮬레이터가 대부분의 반도체 제조환경을 시뮬레이션하는데 사용될 수 있음을 의미한다. 반면에 어떤 시스템의 경우는 상당한 수정이 필요하기 때문에 이 시뮬레이터가 사용되지 못할 수도 있다. 따라서 객체지향적 언어로 개발된 시뮬레이터의 재사용성의 범위 및 이를 증대시킬 수 있는 모형화 방법에 관한 연구가 요구되고 있다.

5-2. 객체 지향적 언어로 구성된 시뮬레이터의 수행효율성에 관한 연구

대상이 된 시스템들을 시뮬레이션하는데 본 시뮬레이터는 C로 쓰여진 동등한 시뮬레이터보다 최대 6배의 시간이 걸린다는 것을 실험을 통해 알 수 있었다. 따라서 복잡한 실세계 문제들에 대한 객체지향적 언어로 구성된 시뮬레이터의 수행도를 알아내는 것과 보다 효율성있는 시뮬레이터의 개발에 관한 연구가 필요하다.

참고문헌

[1] Ben-Arieh, D., "Manufacturing System Application of a Knowledge Based Simulation," *Proceedings of the 8th Annual Conference on Computers and Industrial Engineering*, pp. 459-463, 1986.

[2] CACI, *SIMSCRIPT II.5 Programming Languages*, CACI Inc., Los Angeles, CA, 1983.

[3] Glassey, C.R., and Resende, G.C., "Closed-Loop Release Control for VLSI Circuit Manufacturing," *IEEE Transactions on Semiconductor Manufacturing*, Vol. 1, No. 1, pp. 36-46, 1988.

[4] Goldberg, A., and Robson, D., *Smalltalk-80 : The Language and its Implementation*, Addison-Wesley, Reading, MA, 1983.

[5] Hackman, S.T., and Leachman, R.C., "A General Framework for Modeling Production," *Management Science*, Vol. 35, No. 4, pp. 478-495, 1989.

[6] Jones, C.V., and Maxwell, W.L., "A System for Manufacturing Scheduling with Interactive Computer Graphics," *IIE Transactions*, Vol. 18, No. 3, pp. 298-303, 1986.

[7] Klein, B., "SIMFACTORY Tutorial," *Proceedings of the 1986 Winter Simulation Conference*, pp. 530-542, 1986.

[8] Leachman, R.C., "Preliminary Design and Development of a Corporate-Level Production Planning System for the Semiconductor Industry," ESRC 86-11, Engineering Systems Research Center. University of California at Berkeley, November 1986.

[9] LeClaire, B., "Object Oriented," *OR/MS Today*, Vol. 18, No. 7, pp. 20-24, February 1991.

[10] Pegden, C., *Introduction to SLAM*, Systems Modeling Corporation, PA, 1985.

[11] Pritsker, A.A.P., and Pegden, C., *Introduction to Simulation and SIMAN*, Halstead Press, John Wiley & Sons, New York, 1987.

[12] Ramamoorthy, C.V., Shekhar, S., and Garg, V., "Software Development Support for AI Programs," *IEEE Computer*, pp. 30-40, January 1987.

[13] Resende, M.G.C., "Shop Floor Scheduling of Semiconductor Wafer Manufacturing," ESRC 87-1, Engineering Systems Research Center. University of California at Berkeley, September 1987.

[14] Schriber, T.J., *Simulation Using GPSS*, Wiley, New York, 1974.

[15] Shannon, R.E., Mayor, R., and Adelsberger, H.H., "Expert Systems and Simulation," *Simulation*, pp. 306-310, June 1985.

[16] Wortman, D.B.N and Minor, R.J., "Designing Flexible Manfuacturing Systems Using Simulation," *Proceedings of AUTOFACT 6*, pp. 24-29, October 1986.

[17] Young, R.E., Vesterager, J., Wichman, K.E., and Heide, J., "Simulation Uses in CIM Development," *International Journal of Computer Integrated Manufacturing*, Vol. 1, No. 1, pp. 50-54, 1988.