의미론에 근거한 멀티프로세서 시스템 모델링 기법

°홍 경표, 김탁곤
한국과학기술원 전기 및 전자공학

A Semantic Based Approach to Modeling Multiprocessor Systems

°Gyung Pyo Hong, Tag Gon Kim
Dept. of Electrical Engineering
Korea Advanced Institute of Science and Technology

Abstract

This paper shows an approach to modeling multiprocessor systems based on a mathematical semantics that is compatible with the underlining view of multiprocessor systems. Zeigler's DEVS formalism provides one such semantics. The formalism provides a framework for hierarchical, modular construction of disrete event models of multiprocessor systems. DEVSIM++ is used to develop hierarchical, modular models of multi-bus multiprocessor systems based on the DEVS semantics. Simulation results for such models are also provided.

1. 서 론

최근들어 많은 응용 분야에서 고성능 멀티프로세서 시스템의 수요가 증가함에 따라 멀티프로세서 시스템의 효율적인 설계를 위한 연구가 활발히 진행되고 있다.[12] 복잡한 구조의 시스템을 개발하는 과정은 매우 많은 시간과 경비를 요하는 작업으로서 다양한 형태의 시스템 구조를 효율적으로 분석할 수 있는 방법이 요구된다. 이러한 방법으로서 시뮬레이션 모델링 기법이 사용된다.

멀티프로세서 시스템은 여러개의 독립적인 객체들로 구성된 모듈라 시스템으로서 이러한 시스템의 설계는 독립적인 모듈들의 계층적 결합/분해 과정을 통해서 이루어진다. 따라서 멀티프로세서 시스템의 모델링을 위한 가장 적합한 방법은 모듈라하고 계층적인 접근방법이 될 것이다. 멀티프로세서 시스템을 이산사건 시스템 레벨에서 모델링하고 시뮬레이션 하기 위한 언어(예: SIMSCRIPT II.5)들은 많이 존재하지만, 이들은 언어 자체의 제한때문에 모듈라하고 계층적인 모델링을 위한 의미론을 지원하지 못한다. [3],[8]

본 논문에서는 수학적 의미론에 근거한 멀티프로세서 시스템 모델링 기법에 대하여 기술한다. 의미론은 이산사건 모델을 실세계에 존재하는 객체와 동일하게 기술할 수 있는 표현력을 제공하며, 또한 이들 모델을 통해서 새로운 모델을 계층적으로 구성할 수 있는 방법론을 제공한다. Zeigler의 DEVS(Descrete Event Systems Specification) 형식론은 이산 사건 시스템을 모듈라하고 계층적으로 specification 할 수 있는 표현력을 가진다.[4] 따라서 DEVS 형식론은 상기 기술한 의미론을 지원하며, 이는 이산 사건 모델의 객체지향 specification과 상당히 유사한 측면을 가진다.[8]

DEVS 형식론의 구현에 의한 시뮬레이션 환경의 구축을 위해 많은 연구가 있었으며 [5],[6],[7], 구현상의 차이점은 [1]에 잘 나타나 있다. 본 논문에서는 이산사건 시스템 모델링 및 시뮬레이션을 위해서 DEVSIM++을 사용하였다.[7] DEVSIM++은 DEVS 형식론을 C++을 이용하여 구현한 시뮬레이션 환경으로서 의미론이 지원되며, C++로 구현하였기 때문에 강력한 객체 지향 표현력이 보장되고 빠른 수행 속도를 가진다.

2. 멀티프로세서 시스템 모델링

멀티프로세서 시스템은 구조적인 측면에서 크게 loosely-coupled 시스템 및 tightly-coupled 시스템으로 구분 가능하며, 이들의 자세한 특성상의 차이점은 [9]에 기술되어있다. tightly-coupled 멀티프로세서 시스템의 경우 프로세서와 이의 지역 기억장치, 공통 기억장치 및 상호결합망들이 결합된 형태로 모델링 가능하며, 상호결합망을 통해서 프로세서와 공통 기억장치 사이의 데이타 전송이 이루어진다. 상호결합망에는 crossbar switch, shared bus 시스템 및 multistage 망 등이 있으며, 이중 가장 간단한 형태가 다중버스 시스템(multiple bus system)이다. 다중버스 시스템 구조에서 데이타는 한개의 버스를 통해서 전달되며, 이의 성능은 프로세서, 버스 및 공통기억장치의 갯수 및 버스 중재방식에 따라 달라지게 된다.

멀티프로세서 시스템은 기억장치 및 프로세서 등의 독립적인 객체들이 모여서 하나의 단위 시스템을 구성하고 단위 시스템, 상

호결합망 및 공통 기억장치가 결합되는 형태로 모델링 가능하다. 이러한 구조는 멀티프로세서 시스템이 모듈라하고 계층적인 구조를 가지고 있음을 나타내며, 상기 기술한 바와 같이 이러한 형태의 구조는 DEVS 형식론으로 잘 표현될 수 있음을 알 수 있다. 따라서 DEVS 형식론에 의거한 멀티프로세서 모델과 실제 시스템은 개념적으로 일치함을 알수 있다.

3. DEVSIM++ 환경에서의 계층적 모델링

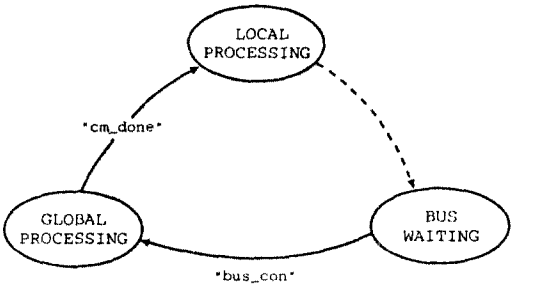
이 장에서는 DEVSIM++ 환경하에서 멀티프로세서 시스템의 모델링 방법에 대하여 기술한다. 다중버스 멀티프로세서 시스템의 경우 기억장치 및 프로세서의 갯수가 증가하거나, 처리속도가 빨라지는 경우 버스의 갯수가 성능저하의 주된 요인이 되기 때문에 버스의 수가 프로세서의 수에 비해 상대적으로 매우 작은 시스템을 모델링 하였다.[2]

DEVSIM++은 DEVS 형식론을 C++로 구현한 시뮬레이션 환경으로서 DEVS 의미론을 지원하고 C++이 제공하는 객체지향 표현력 및 빠른 수행 속도를 가진다.[11] DEVSIM++ 환경하에서의 모델링은 시스템을 분해하여 가장 기본적인 객체를 atomic 모델로 개발하고, 이들을 결합하여 더 큰 시스템 요소를 표현하는 coupled 모델을 개발하는 과정으로 이루어 진다.[7] 다중버스 멀티프로세서 시스템에서의 atomic 모델은 프로세서, 버퍼 및 버스 등이 있으며, coupled 모델은 프로세서들로 구성된 PROCS, 기억장치들로 구성된 CMS 등이 있다.

3.1 Atomic 모델 개발

본 절에서는 atomoc 모델 PROC의 개발에 대하여 기술한다. atomic 모델의 개발은 대상 객체의 기능 분석후 이를 DEVS 형식론에 입각하여 기술하고, 이를 DEVSIM++ 환경에서 구현하는 절차를 거친다. PROC은 프로세서와 기억장치와 구성되며, 이의 입출력 포트 및 상태변수는 그림 1a 와 같다. atomoc 모델 PROC은 local 연산을 수행하는 'local_processing phase', 버스 서비스를 기다리는 'bus_waiting' 및 공통 기억장치를 사용하는 'global_processing' phase를 가진다. local processing을 마치면 PROC은 CM을 사용하기 위해서 버스 사용을 요구하게 되며, 이때 버스를 사용할 수 없으면 bus waiting phase에서 버스의 할당을 기다리게 된다. 만약 버스의 사용이 가능한 경우 global_processing phase에서 공통 기억장치를 이용한 작업을 수행한 후에 local_processing phase로 돌아온다. 그림 1b는 상기 기술한 과정을 phase 천이도로 표시한 것이다. phase 간의 천이는 외부 입력에 의해서 발생되는 외부 천이(external transition) 및 외부의 입력없이 시간전진 함수(time_advanced function)에 의해서 발생하는 내부 천이(internal transition)가 있으며, 그림 1b에서 실선은 외부 천이, 점선은 내부 천이를 표시한다.

(a) I/O ports and state variables



(b) pahse transition diagram

그 림 1. atomic 모델 PROC
Fig. 1 Atomic Model PROC

그림 2는 atomic 모델 PROC의 DEVS specification중 내부 천이 및 출력함수의 Pseudo-code를 보여주며, 그림 3은 specification을 DEVSIM++로 구현한 코드이다. 버스 및 공통 기억장치의 atomic 모델 GBUS 및 CM도 PROC과 동일한 방법으로 개발 가능하며, GBUS는 PROC과 CM을 연결하는 버퍼로 모델링하고 CM은 프로세서가 공통기억장치를 사용하는 기간만큼 시간을 지연시키는 지연회로로 모델링 가능하다. 버스 중재 방식은FIFS(First-In-First-Service) 방식을 사용하였다.

3.2 Coupled 모델 개발

coupled 모델은 atomic 모델의 결합을 통해서 개발되는데, atomic 모델에는 다른 모델들과의 결합에 관계된 정보가 들어있지 않다. 따라서 설계자는 coupled 모델 specification시에 모델들 간의 상호 결합정보를 기술해 주어야한다. 시뮬레이션 모델 MULTIPROC은 2개의 coupled 모델 PROCS, CMS와 atomic 모델 GBUS로 구성되며, MULTIPROC의 동작을 위해서는 PROC과 GBUS 사이에는 버스사용 요구신호 및 승인신호를 주고받을 수 있어야 하며, GBUS와 CMS 사이에는 버스사용 승인신호 및 CM 사용완료 정보를 주고 받을 수 있어야 한다. 이러한 정보를 기술한 DEVS specification은 그림 4에 나타나 있으며, 그림 5는 specification을 DEVSIM++로 구현한 코드이다. 또한 시뮬레이션의 수행을 위해서는 데이타를 생성하는 atomic 모델 GEN 및 데이타를 수집 분석하는 atomic 모델 TRANS로 구성된 experimental frame이 필요하게 되며, 이 experimental frame과 coupled 모델 MULTIPROC이 결합된 전체 시뮬레이션 모델 MULTISIM은 그림 6에 나타나 있다.

```
// Internal Transition //
case phase of :
   "LOCAL_PROCESSING": phase = WAITING
   "COMMON_PROCESSING":
         phase = "LOCAL_PROCESSING",
         job = buffer(first)

// Output Function //
case phase of :
   "LOCAL_PROCESSING" : bus_req = job
```

그 림 2. PROC 모델의 DEVS 사양
Fig. 2 DEVS Specification for the PROC model(Pseudo-code)

```
// Internal Transition //
void PROC_int_transfn(State_vars& s)
{
   if (s.get_vlaue("phase") == LOCAL_PROCESSING)
      s.set_value("phase", BUS_WAITING);
}

// Output Function //
void PROC_outputfn(const State_vars& s)
{
   if (s.get_value("phase") == LOCAL_PROCESSING)
      message.set("bus_req", TRUE);
}
```

그 림 3. atomic 모델 PROC의 DEVSIM++ 코드
Fig. 3 DEVSIM++ code for atomic model PROC

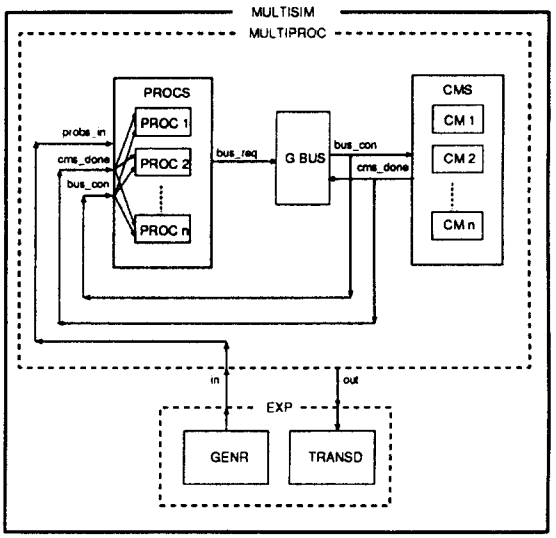
$MULTIPROC = \langle D, \{M_i\}, \{I_i\}, \{Z_{ij}\}, SELECT \rangle$

D = {PROCS, CMS, GBUS}
M_{PROCS}, M_{CMS}, M_{GBUS}
I_{PROCS} = {GBUS}
I_{CMS} = {GBUS}
I_{GBUS} = {PROCS,CMS}
$Z_{PROCS,GBUS}$ = bus_req → gbus_in
$Z_{CMS,GBUS}$ = cms_out → cm_done
$Z_{GBUS,PROCS}$ = gb_out → bus_con
$Z_{GBUS,CMS}$ = gb_out → cms_in
$SELECT(\{PROCS, CMS, GBUS\})$ = PROC
$SELECT(\{CMS, GBUS\})$ = GBUS

그 림 4. couple 모델 MULTIPROC의 DEVS 사양
Fig. 4 DEVSIM++ specification

4. DEVSIM++ 환경에서의 시뮬레이션 수행

시뮬레이션은 프로세서,기억장치 및 버스의 조합에 따른 processing power의 측정을 목적으로 하였다. processing power는 연산을 수행하고 있는 평균적인 프로세서의 수로 정의되며, load factor는 평균적인 global 연산시간과 local 연산시간의 비로 정의된다. 입력데이타의 생성 및 출력데이타의 수집/분석을 위해서 SIMFIT++ [11]을 사용하였으며, 상세한 입력 해석 방법은 [10]에 기술되어 있다. 테이블 1은 load factor=0.1이고 프로세서의 수가 12개인 경우 각각 1000, 2000 및 5000개의 데이타를 가지고 5회의 반복적인 시뮬레이션을 수행한 결과를 보여준다. 그림 7은 MULTIPROC 모델의 시뮬레이션 결과를 보여준다.



그 림 6. 시뮬레이션 모델 MULTISIM
Fig. 6 Overall Simulation Model MULTISIM

```
void make_MULTIPROC(Coupled_models& MULTIPROC, Coupled_models& PROCS,
   Coupled_models& CMS, Atomic_models& GBUS)
{
   make_GBUS(GBUS);
   MULTIPROC.add_outports(1, "out");
   MULTIPROC.add_children(3, &PROCS, &CMS, &GBUS);
   MULTIPROC.add_coupling(&PROCS, "bus_req", &GBUS, "gbus_in");
   MULTIPROC.add_coupling(&GBUS, "gbus_out", &PROCS, "bus_con");
   MULTIPROC.add_coupling(&GBUS, "gbus_out", &CMS, "cms_in");
   MULTIPROC.add_coupling(&CMS, "cms_out", &GBUS, "cm_done");
   MULTIPROC.add_coupling(&CMS, "cms_out", &MULTIPROC, "out");
   MULTIPROC.add_priority(3, &GBUS, &PROCS, &CMS);
}
```

그 림 5. coupled 모델 MULTIPROC의 DEVSIM++ 코드
Fig. 5 DEVSIM++ code for coupled model MULTIPROC

5. 결 론

본 논문에서는 멀티프로세서 시스템의 모듈라하고 계층적인 구조와 동등한 수학적 의미론을 제공하는 Zeigler의 DEVS 형식론에 근거한 멀티프로세서 시스템의 모델링 및 시뮬레이션 방법에 대하여 기술하였다. 기존의 시뮬레이션 언어는 언어 자체의 표현력의 한계로 인해서 본 고에서 기술한 모델링 방법을 지원하지 못하나, DEVSIM++은 DEVS 형식론이 제공하는 의미론을 C++을 이용하여 구현하였기 때문에 모듈라하고 계층적인 구조적 특성을 갖는 시스템의 모델링시 실세계에 존재하는 시스템과 잘 대응되는 모델을 구할 수 있는 장점을 가진다. 또한 C++이 제공하는 객체 지향형 표현력 및 빠른 수행속도를 가진다. 또한 본 고에서 기술한 모델링 방법은 다른 환경에서 구축된 모델들을 의미론에 입각해서 쉽게 변환이 가능하며, 이미 개발된 모델들을 결합하여 새로운 모델들을 개발 가능하기 때문에 지식베이스 모델링이 가능하다.

참고문헌

[1] Myung S. Ahn and Tag G. Kim, "Simulating Message-routing Algorithms for Hypercubes Using DEVSIM++", *Proc. of the '93 SCS Western Multiconference (Object-Oriented Simulation Conference)*, 1993.

[2] M.A.Marsan, G. Balbo and G. Conte, *Performance Models of Multiprocessor Systems*, The MIT Press, 1986.

[3] "Implementation of the model base concept in SIMSCRIPT II.5", Master Thesis University of Arizona, Tucson, AZ, USA(1987)

[4] A.I. Concepcion and B.P. Zeigler, "DEVS formalism: A framework for hierarchical model development", *IEEE Trans. on Software Eng.*, Vol. 14, No. 2, pp 228-241, Feb., 1988.

[5] J.M. Farrow and S. Sevinc, "Modelling Tools for A Common LISP Object-oriented System Environment", *Proc. of the Second Annual Conf. on AI, Simulation and Planning in High Autonomy Systems*, 1991.

[6] Tag G. Kim and B.P. Zeigler, "The DEVS-Scheme Simulation and Modelling Environment", Chapter 2 in *Knowledge Based Simulation: Methodology and Application* (eds: Paul A. Fishwick and Richard B. Modjeski) Springer Verlag., Inc., pp. 20-35, 1990.

[7] Tag G. Kim and Sung B. Park, "The DEVS Formalism: Hierarchical Modular Systems Specification in C++", *Proc. of the 1992 European Simulation Multiconference*, pp. 152-156, 1992.

[8] Tag G. Kim, "The DEVS Formalism: Reusable Model Specification in an Object-Oriented Framework", *International Journal in Computer Simulation*, accepted for publication.

[9] Kai Hwang, Faye A. Briggs *Computer Architecture and Parallel Processing*, McGraw-Hill, Inc., 1984.

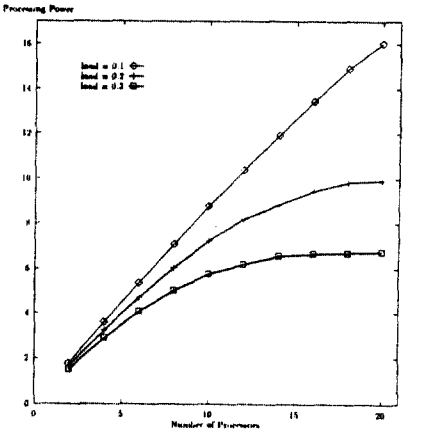
[10] Averill M.Law, W.David Kelton, *Simulation Modeling and Analysis*, McGraw-Hill, Inc., 1991.

[11] Sung B. Park "DEVSIM++: A Semantic Based Tool for Object-Oriented Modeling of Discrete Event Systems", Master thesis Dept. of Electrical Engineering, KAIST, 1993.

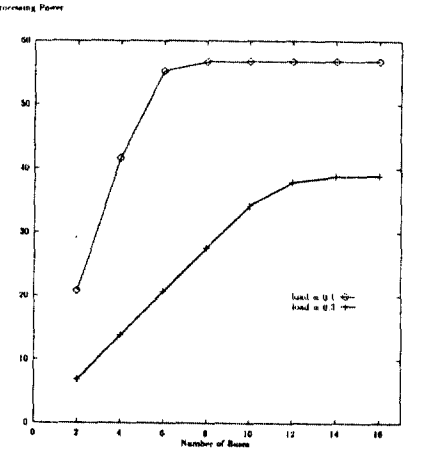
[12] L.W. Tucker and G.G. Robertson, "Architecture and Applications of the Connection Machine", *IEEE Comput. Mag.*, pp.26-38, Aug. 1988.

시험수	Run-Len=1000	Run-Len=2000	Run-Len=5000
1	10.39	10.40	10.41
2	10.45	10.49	10.43
3	10.44	10.46	10.42
4	10.37	10.40	10.41
5	10.41	10.39	10.42

도 표 1. 평균 Processing Power



Processing power vs. Num of Buses
Num of processors=64, memories=20



Processing power vs. Num of Processors
Num of Buses=2, memories=3

Fig. 7 시뮬레이션 결과