論文98-35C-4-5

병렬 DSP 시스템을 이용한 화력발전소 고속 시뮬레이션

(High-Speed Simulation for Fossil Power Plants Using a Parallel DSP System)

朴 喜 晙 *, 金 炳 國 *

(Hee-Jun Park and Byung-Kook Kim)

요 약

화력발전소는 많은 수식과 미분방정식으로 모델링될 수 있다. 대규모의 복잡한 화력발전소를 컴퓨터를 이용하여 시뮬레이션 하고자 할 때, 수식계산에 많은 시간이 소요되므로 실시간 또는 실시간 이상의 속도로 시뮬레이션 하려면 빠른 수행능력을 가진 계산기 개발이 필요하다. 이 논문에서는 여러 개의 DSP 프로세서로 구성된 병렬 시스템을 제작하여, 병렬처리를 통해 시뮬레이션의 수행시간을 단축시키는 방법을 제안한다. 이를 위해서 병렬 DSP 시스템을 확장 가능한 구조로 설계하였고, 범용 DSP 모듈과 VME 인터페이스 모듈을 제작하였다. 그리고 병렬 연산 및 통신 기능을 고려한 모델을 정립하고 작업할당의 현실적인 비용함수로서 simulation period를 제안하여, 화력발전소 모델에 대한 작업할당을 수행하였다. 병렬 DSP 시스템개발과 현실적인 작업할당 모델을 통해 병렬처리의 효율을 높일 수 있었으며 실시간 이상의 속도로 시뮬레이션이 가능하였다.

Abstract

A fossil power plant can be modeled by a lot of algebraic equations and differential equations. When we simulate a large, complicated fossil power plant by a computer such as workstation or PC, it takes much time until overall equations are completely calculated. Therefore, new processing systems which have high computing speed is ultimately needed for real-time or high-speed(faster than real-time) simulators. This paper presents an enhanced strategy in which high computing power can be provided by parallel processing of DSP processors with communication links. DSP system is designed for general purpose. Parallel DSP system can be easily expanded by just connecting new DSP modules to the system. General purpose DSP modules and a VME interface module was developed. New model and techniques for the task allocation are also presented which take into account the special characteristics of parallel I/O and computation. As a realistic cost function of task allocation, we suggested 'simulation period' which represents the period of simulation output intervals. Based on the development of parallel DSP system and realistic task allocation techniques, we could achieve good efficiency of parallel processing and faster simulation speed than real-time.

Ⅰ. 서 론

시스템 해석과 시뮬레이션을 위하여 다양한 패키지가 많이 개발되고 있으나, 현재 주로 사용되고 있는 패키지들(Ctrl-C, EASY5, Simnon, SIMULINK, MATRIXx)은 제어알고리즘의 최적해를 구하거나 제어모델의 응답특성을 해석하는 목적으로 사용되기 때문에 시뮬레이션 속도에는 큰 비중을 두지 않았다. 하지만 실제 제어기와 연결되어 on-line으로 제어기의

* 正會員, 韓國科學技術院 電氣·電子工學科
(Dept. of Elec. Engineering, KAIST)

接受日字:1998年2月9日, 수정완료일:1998年3月30日

작동을 검증하기 위해서는 실제발전소의 샘플링 시간마다 정해진 시뮬레이션 계산을 모두 완료하여 결과를 제어기로 전송할 수 있어야 한다. 즉 시뮬레이터는 실시간이라는 요건이 필요하므로 빠른 계산 능력을 가지고 있어야 한다. 또 일반적으로 발전소의 시동에 걸리는 시간은 1~2시간정도로서 상당히 장시간이 걸리므로, 신속한 결과검증을 위해서 실시간 보다 빠른 시뮬레이터가 필요하게 된다.

기존의 연구는 화력발전소 모델의 많은 계산량을 실시간으로 시뮬레이션 하기 위하여, 모델을 근사화, 선형화하거나 고속 전용 계산기를 개발하는 등의 두 가지 방향으로 진행되었다.

첫째, 모델의 근사화, 선형화에 대한 연구는 다음과 같다. Kwan은 200MW급의 석탄연소 자연순환형 보일러에 대하여 109개의 방정식과 31개의 상태변수로 모델링하였고 안정적인 해를 얻기 위하여 5 ms 정도의 적분간격이 필요함을 지적하였다. 선형화 및 근사화에 의해 5%의 제한된 변화에 대해서만 시뮬레이션이 가능하였다[1]. Dolezal의 경우 방정식을 선형화하거나 계수를 일정하게 취급할 경우, 상태변화가 큰 시뮬레이션을 수행할 때, 오차가 증가함을 지적하고 각기기 별로 얻어진 비선형 미분방정식을 풀이하는 방법으로 시뮬레이션 하였다. 계산능력의 부족으로 비선형 방정식의 일부에 해석해를 도입하는 방법을 취하였다[2].

둘째, 전용계산기 개발로 실시간 시뮬레이션을 하려 하였던 연구는 다음과 같다. Rafian은 decomposition technique을 이용하여 power system모델의 상태 방정식을 1~3개의 area로 분리하여 병렬처리하여 실시간의 속도로 시뮬레이션할 수 있었으나, 작업할당이 수동적으로 정해졌고 효율적인 작업할당에 대한 고려가 부족하였다[3]. Yoshida는 발전기가 100개정도인 power system을 대상으로 다중 프로세서 구조의 컴퓨터에서 운전원 훈련용 시뮬레이터를 개발하였다. 하지만 이 연구에서도 task간의 통신량을 무시한 작업할당방법을 사용하는 등 작업할당에 대한 고려가 부족하였다[4].

모델의 선형화, 단순화에서 계산량을 감소시킬 경우 오차가 누적되어, 시뮬레이션의 정확도가 떨어진다. 그러므로 이 연구에서는 화력발전소의 모델을 여러 개의 DSP 프로세서를 이용하여 병렬처리 하므로서 실시간 이상의 고속으로 시뮬레이션 하는 방법을 제시하고, 화력발전소 시뮬레이션에 적합한 작업할당 알고리즘을 개발함으로서 병렬처리의 효율을 높였다. 이 작업할당 알고리즘은 task의 동기화, 통신선의 contention, 프로세서의 병렬 연산 및 통신 기능 등 현실적인 환경을 고려한 것이다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 제작된 병렬 DSP 시스템의 개발목적, 구조, 성능을 설명하고, Ⅲ장에서는 새로운 비용함수인 simulation period를 제안하고 branch and bound 알고리즘을 이용하여 simulation period를 최소화하는 작업할당 방법을 제시한다. Ⅳ장에서는 [5]에서 인용한 시뮬레이션 대상 화력발전소의 블럭도, task graph, 계산량 등을 구체적으로 정리한다. Ⅴ장에서 작업할당결과, 시뮬레이션 속도와 병렬처리 효율, 다른 계산기와의 속도비교 등을 분석하고 Ⅵ장에서 결론을 맺는다.

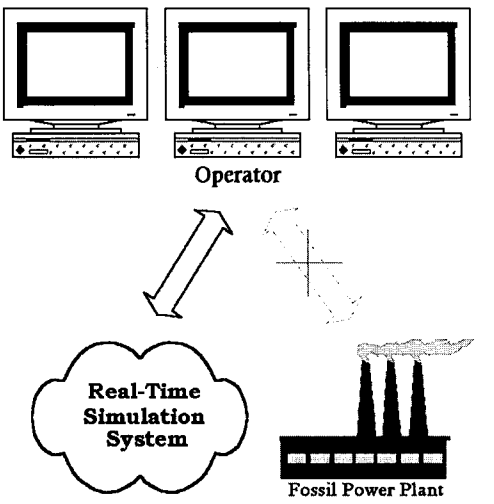

그림 1. 화력발전소 시뮬레이터의 역할
Fig. 1. Function of Fossil Power Plant Simulation.

Ⅱ. 병렬 DSP 시스템 개발

개발된 병렬 DSP 시스템은 플랜트의 고속 시뮬레이션 목적으로 사용되므로 다음과 같은 요건들을 고려하여 설계하였다. 첫째, 프로세서 자체만으로도 빠른 계산 능력을 가지고 있어야 하므로 병렬처리 컴퓨터 제작목적으로 설계된 Texas Instrument사의 TMS 320C40 프로세서를 사용하였다. 이 프로세서는 6개의 고속통신포트, 6개의 DMA coprocessor, 25 MIPS의 floating point 계산능력, on-chip program

cache 등 병렬처리 설계에 용이하고 응용대상인 시뮬레이션의 방대한 계산량을 처리하기에 적합한 프로세서이다[6] [7]. 둘째, 병렬컴퓨터의 확장성을 높이고 임의의 network 구성이 용이하도록 distributed-memory구조로 설계하였고, 각 기능을 하는 module을 제작하여 연결함으로서 개발 및 debugging을 쉽게 하고 시스템의 안정성을 높였다. 셋째, 일반적으로 플랜트용 제어시스템은 VME bus를 기반으로 제어보드와 host computer가 연결되는 구조로 되어있으므로 병렬처리 컴퓨터 부분과 VME bus의 인터페이스를 위한 회로가 필요하다. 이 회로를 통해서 시뮬레이터와 제어기가 정보를 주고받으며 시뮬레이션을 하게 되는 것이다. VME interface는 한 개의 module로 제작되었고 나머지 module을 고정시키는 mother board 역할을 하게 된다. 이와 같이 구성된 병렬 DSP 시스템은 앞의 분류에 의하면 MIMD, distributed-memory 구조이며 7개이하의 프로세서를 사용할 때 fully-connected network을 구성할 수 있다.

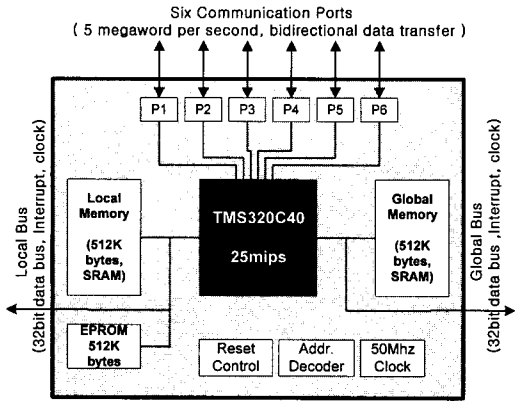

그림 2. DSP 모듈의 블럭도
Fig. 2. Block diagram of DSP module.

DSP module과 VME interface module를 이용하여 병렬 DSP 시스템을 구성하였다. 전체 시스템의 구성은 그림 4와 같다. workstation이 호스트 컴퓨터로 사용되며 제어기로서 CPU30 board가 사용된다. 발전소 시뮬레이터용으로 병렬 DSP 시스템이 사용되는데, 현재 한 개의 VME interface module과 네 개의 DSP module로 구성되어 있고 fully-connected network으로 연결되어 있다. 호스트 컴퓨터에서는 제어기와 시뮬레이터에서 수행될 프로그램 개발이 이루어진다. 프로그램은 어셈블리와 C언어로 coding되며 이 프로그램이 호스트 컴퓨터의 하드디스크에 저장되어 있어야한다. CPU30 board와 호스트 컴퓨터는 Ethernet으로 연결되어 있고 CPU30 board와 병렬 DSP 시스템은 VME bus로 연결된다. Ethernet을 통해 CPU30 board와 병렬 DSP 시스템에서 수행될 제어 프로그램과 시뮬레이션 프로그램이 다운로드 되며 시스템 start-up이 모두 종료되면 CPU30 board와 병렬 DSP 시스템은 VME bus를 통해서 입출력값을 주고받으며 시뮬레이션을 수행하게 된다.

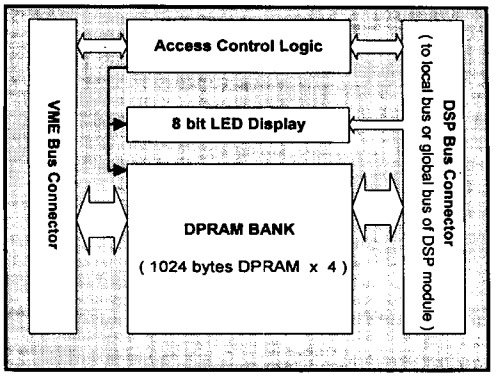

그림 3. VME 인터페이스 모듈의 블록도
Fig. 3. Block diagram of VME interface module.

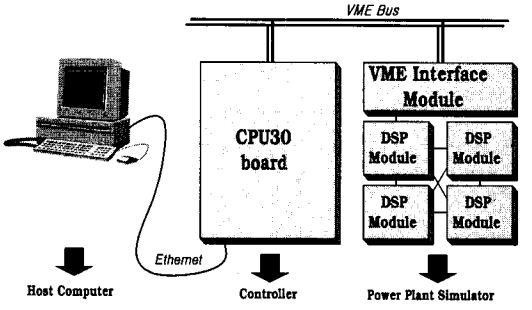

그림 4. 전체 시스템의 구성
Fig. 4. The structure of Overall Systems.

Ⅲ. 통신환경을 고려한 작업의 최적할당방법

1. 기존의 비용함수와 문제점

병렬컴퓨터는 각 프로세서가 network으로 구성되어 있기 때문에 통신부담이 발생하고 통신량이 많아질수록 통신에 걸리는 시간이 많아져서 전체 시스템의 성능이 저하되게 된다. 이와 같은 시스템의 성능저하는 각 프로세서에 할당된 부하의 불균형 및 과도한

IPC(Interprocessor Communication)에 기인한다. 이런 데이터 통신에 따른 통신부담은 병렬컴퓨터의 processing element간 통신량과 통신선의 속도에 따라 정해지므로 task가 어느 프로세서에서 수행되는가에 따라서 수행시간이 달라진다. 각각의 프로세서에 대해 수행되는 task의 집합을 정해주는 것을 작업할당(Task Assignment, Task Allocation)이라고 한다. 병렬컴퓨터와 같은 분산시스템에 대한 작업할당은 가능한 각각의 할당에 대해서 비용함수(cost function)를 정의하고 이 비용함수가 최소가 되도록 하는 방향으로 연구되어 왔다[8]~[12].

기존의 많은 연구에서 수행비용(execution cost)과 통신비용(communication cost)의 합을 비용함수로 정의하여 작업할당문제를 연구하고 있다. 즉 task i가 processor k에서 실행되고 task j가 processor l에서 실행될 때, execution cost를 $R_{i,k}$라고 하고, task i와 task j사이의 통신량을 $W_{i,j}$, processor k와 processor l사이의 거리(minimum number of communication links)를 $d_{k,l}$, X를 M개의 task를 N개의 프로세서에 할당하는 할당함수(assignment function)라고 하면 할당 X에 대한 비용은 식 1과 같이 표현된다[8].

$$\begin{aligned} COST(X) &= exec.\ cost + comm.\ cost \\ &= EXEC(X) + COMM(X) \\ &= \sum_{i=0}^{M} R_{i,X(i)} + \sum_{i,j} W_{i,j} \cdot d_{X(i),X(j)} \end{aligned} \quad (1)$$

하지만 기존의 연구에 사용된 모델은 계산작업과 통신작업을 순차적(sequential)이라고 가정하였고 통신선에 대한 간섭비용은 고려하지 않았다. 하지만 최근에 개발된 프로세서, 특히 TMS320C40은 통신과 계산을 병렬적(parallel)으로 수행할 수 있는 기능을 제공하고 있으므로 수행비용과 통신비용의 합으로 할당을 평가하는 것은 시뮬레이션 수행환경에서 비현실적이다. 또 수행될 플랜트 시뮬레이션에 대한 task들은 동기를 맞추며 수행되어야 하고 통신의 방향성이 있다는 특징을 가지고 있어 이러한 환경을 고려하지 않은 기존의 할당방법으로는 최적의 할당을 얻을 수 없다. 또 실제로 제작된 병렬 DSP 시스템은 동일한 성능의 동일한 프로세서로 구성되어 있으므로 수행시간의 합을 비용함수에 넣는 것은 불필요하다.

2. Simulation Period 제안

이 연구에서 사용될 하드웨어 환경과 task에 대한 가정은 다음과 같다.

- 병렬 연산 및 통신 : TMS320C40 프로세서의 DMA기능을 이용하여 계산작업과 데이터통신이 동시에 병렬로 이루어진다.
- 병렬처리 시스템은 fully-connected network 구조이다. (그림 5)
- synchronization : 각 task에 대한 계산은 동기화를 맞추어야 한다.
- no precedency : 각 task에 대한 계산이 종료된 후 task간의 입출력 값을 통신을 통해 갱신한다.

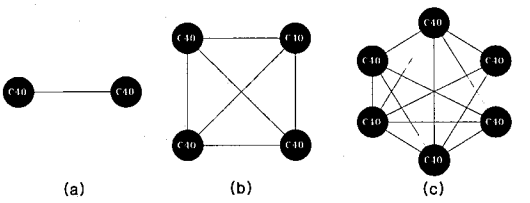

그림 5. Fully-connected Network (a) 2 DSPs (b) 4 DSPs (c) 6 DSPs

Fig. 5. Fully-connected Network. (a) 2 DSPs (b) 4 DSPs (c) 6 DSPs

시뮬레이션은 각 프로세서에 할당된 발전소의 서브플랜트 즉 보일러, 터빈, 응축기 등에 대한 계산을 수행하여 그 결과를 제어기에 샘플링 주기마다 전송하는 작업인데 고속 시뮬레이션시에는 시뮬레이션 결과가 출력되는 주기를 최소로 감소시켜야 한다. 시뮬레이션 결과값의 출력 주기를 시뮬레이션 주기(simulation period)라고 정의한다. 작업할당 알고리즘은 시뮬레이션 주기를 비용함수로 하여, 이 값을 최소화하는 할당을 구하는 것을 목적으로 한다. 다음은 사용될 symbol에 대한 정의이다.

n : computational iteration

N_T : number of tasks

N_P : number of processors

T_i : execution time of task i

S_{ij} : communication time from task i to task j

A_p = { i : *task i* 는 *processor p* 에 할당된 *task* }

E_p = *execution time on processor p*

$$= \sum_{i \in A_p} T_i$$

C_{pr} = *communication time from processor p to processor r*
$= \sum_{i\in A_p,\ j\in A_r} S_{ij}$

$F_p(n)$: execution start time on processor p in n-th iteration

$D_{pr}(n)$: communication start time from processor p to processor r in n-th iteration

M_p : communication precedency of processor p

z_{pr} : period of local iterations between processor p and processor r

z를 period of computational iteration 즉 simulation period라고 정의하면 다음이 성립한다.

$$F_p(n+1) = F_{p(n)} + z \tag{2}$$

$$D_{pr}(n+1) = D_{pr(n)} + z \tag{3}$$

1) 프로세서가 2개인 경우

2개의 프로세서가 그림 5(a)와 같이 연결되어 있을 때 z를 구해본다. $E_1 \le E_2$ 라고 하면 $M_1 \ge M_2$ 와 $M_1 \le M_2$ 두가지경우에 대해서 z를 구할 수 있다.

- processor 1이 communication predecessor일 경우 ($M_1 \ge M_2$)

processor 1의 communication precedency가 processor 2의 그것보다 크므로 processor 1은 execution후 곧바로 결과값을 전송한다. processor 2는 processor 1의 전송이 종료된 후 출력값을 전송하며 각 프로세서는 이전 execution이 종료되었고 입력값이 유효하면 다음 iteration의 execution을 시작한다.

$$D_{12}(n) = F_1(n) + E_1 \tag{4}$$

$$D_{21}(n) = \max[D_{12}(n)+C_{12},\ F_2(n)+E_2] \tag{5}$$

$$F_1(n+1) = \max[F_1(n)+E_1,\ D_{21}(n)+C_{21}] \tag{6}$$

$$F_2(n+1) = \max[F_2(n)+E_2,\ D_{12}(n)+C_{12}] \tag{7}$$

위의 4조건에 의해 execution과 communication에 대한 sequence가 정해지며 simulation period는 다음과 같다.

$$z^* = \max[E_1+C_{12}+C_{21},\ E_2] \tag{8}$$

- processor 2가 communication predecessor일 경우 ($M_1 \le M_2$)

processor 2의 communication precedency가 processor 1의 그것보다 크므로 processor 2는 execution후 곧바로 결과값을 전송한다. processor 1은 processor 2의 전송이 종료된 후 출력값을 전송하며 각 프로세서는 이전 execution이 종료되었고 입력값이 유효하면 다음 iteration의 execution을 시작한다.

$$D_{21}(n) = F_2(n) + E_2 \tag{9}$$

$$D_{12}(n) = \max[D_{21}(n)+C_{21},\ F_1(n)+E_1] \tag{10}$$

$$F_2(n+1) = \max[F_2(n)+E_2,\ D_{12}(n)+C_{12}] \tag{11}$$

$$F_1(n+1) = \max[F_1(n)+E_1,\ D_{21}(n)+C_{21}] \tag{12}$$

위의 4조건에 의해 execution과 communication에 대한 sequence가 정해지며 simulation period는 다음과 같다.

$$z^{**} = \max[E_2+C_{21}+C_{12},\ E_1] \tag{13}$$

z^*와 z^{**}를 비교해 보면 가정에서 $E_1 \le E_2$ 라고 했으므로 z^*의 경우 $E_1+C_{12}+C_{21}$또는 E_2가 되고 z^{**}의 경우 $E_2+C_{12}+C_{21} > E_1$이므로 $E_2+C_{12}+C_{21}$이 된다.

$$\begin{aligned} E_1+C_{12}+C_{21} &\le E_2+C_{12}+C_{21} \\ E_2 &\le E_2+C_{12}+C_{21} \end{aligned} \tag{14}$$

$$\therefore\ z^* \le z^{**} \tag{15}$$

- 병렬 연산 및 통신 기능을 사용하지 않을 경우

두 프로세서의 execution이 모두 종료된 후 communication이 시작되므로 이때의 simulation periodb는 다음과 같다.

$$\begin{aligned} z` &= \max[E_1,\ E_2] + C_{12} + C_{21} \\ &= E_2 + C_{12} + C_{21} \end{aligned} \tag{16}$$

그러므로 $z^*, z^{**}, z`$ 의 관계는 다음과 같다.

$$z^* \le z^{**} = z` \tag{17}$$

두 프로세서 중 execution time이 적은 프로세서가 communication predecessor가 되어야 병렬 연산 및 통신에 의한 속도향상을 기대할 수 있다. 그러므로 $E_p < E_r$ 이면 $M_p > M_r$ 이 되도록 communication precedency를 정한다.

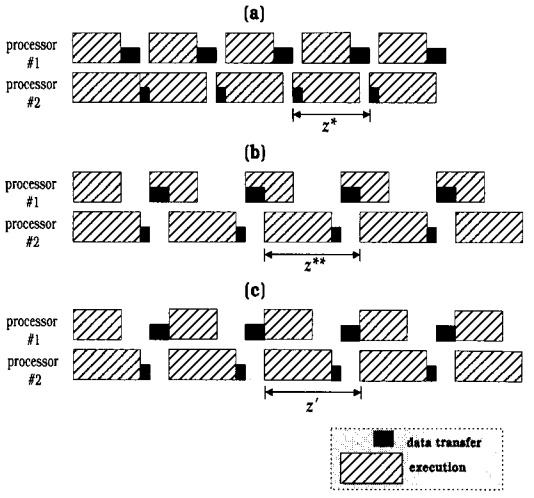

그림 6. Processor 2개의 병렬처리 수행 절차 (a) $M_1 > M_2$ (b) $M_1 < M_2$ (c) 계산과 통신이 순차적일 경우

Fig. 6. Parallel processing squence of two processors. (a) $M_1 > M_2$ (b) $M_1 < M_2$ (c) Squential computation and I/O

2) 프로세서가 n개인 경우

앞에서 고려했던 프로세서가 2개인 경우를 확장하면 프로세서가 2개 이상일 때의 z를 구할 수 있다. 그림 7은 m+1 개의 프로세서로 구성된 fully-connected network에서 processor p와 연결돼 있는 부분만을 그린 것이다. 이 경우를 살펴보면 fully-connected network은 프로세서 2개 경우의 집합이라고 볼 수 있다. 그러므로 processor p의 execution start time과 communication start time은 processor r부터 processor r+m-1까지의 조건이 모두 성립해야 하며 다음과 같이 나타낼 수 있다.

$$\begin{aligned} F_p(n+1) = \max[\ & F_p(n)+E_p,\ D_{r,p}(n)+C_{r,p} \\ & ,\ D_{r+1,p}(n)+C_{r+1,p} \\ & ,\ D_{r+2,p}(n)+C_{r+2,p} \\ & ,\ \cdots \\ & ,\ D_{r+m-1,p}(n)+C_{r+m-1,p}] \end{aligned} \tag{18}$$

$M_p \geq M_{r+i}$ 일때,

$$D_{p,r+i}(n) = F_p(n)+E_p \tag{19}$$

$M_p \leq M_{r+i}$ 일때,

$$D_{p,r+i}(n) = \max[F_p(n)+E_p,\ D_{r+i,p}(n)+C_{r+i,p}] \tag{20}$$

processor p와 연결된 m개의 통신선에 대하여 local simulation period, z_{pr} 을 구할 수 있다.

$M_p \geq M_r$ 이면

$$z_{pr} = \max[E_p+C_{pr}+C_{rp},\ E_r] \tag{21}$$

$M_p \leq M_r$ 이면

$$z_{pr} = \max[E_p,\ E_r+C_{pr}+C_{rp}] \tag{22}$$

각 프로세서가 동기를 맞추며 작동하기 때문에 전체 network의 simulation period를 z라고 하면 z는 local simulation period 중 가장 큰 값이 된다.

$$z = \text{ } 1 \leq p \leq N_P,\ 1 \leq r \leq N_P \text{ 일때,} \quad p \neq r\text{을 만족하는 모든 } z_{pr} \text{ 중 최대값} \tag{23}$$

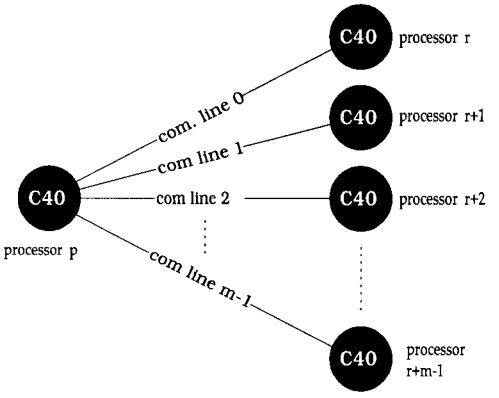

그림 7. Fully-connected network의 일부분

Fig. 7. A part of fully-connected network.

3. Branch and Bound Algorithm

제안된 simulation period를 최소화하는 작업할당은 branch and bound (BB) 알고리즘을 이용하여 구할 수 있다. Branch and bound 알고리즘은 최적해를 구할 수 있는 방법으로서 목적함수의 lower bound 또는 upper bound를 계산하고 불필요한 solution을 제거해 가면서 최적해를 찾게 된다. 그러므로 계산과정을 단축시킬 수 있다[13][14]. 최악의 경우 branch and bound 알고리즘은 가능한 모든 해에 대해서 목적함수를 계산하게 되므로 최적해가 존재한다면 최적해를 찾는 것이 보장된다. 작업할당의 문제정의는 다음과 같다.

- A가 임의의 할당일 때, simulation period, $z = f(A)$를 목적함수로 하고 이 값을 최소화하는 최적해 A_{op}를 구한다. 이 때 어떠한 A에 대해서도 A_{op}는 다음을 만족시킨다.

$$z_{op} = f(A_{op}) \leq f(A)$$

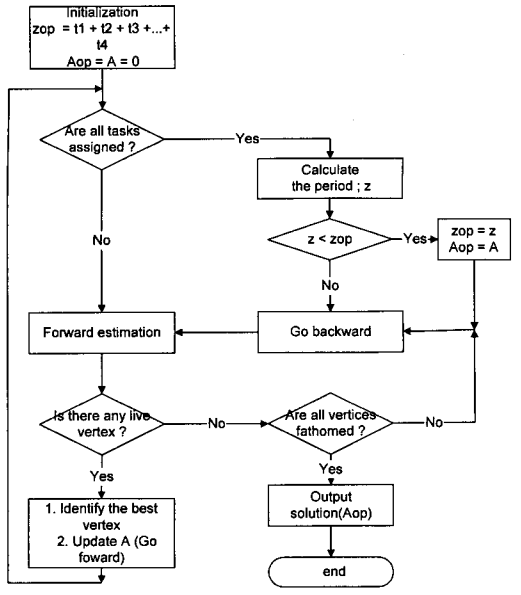

그림 8. Branch and bound 알고리즘의 순서도
Fig. 8. Flow chart of branch and bound algorithm.

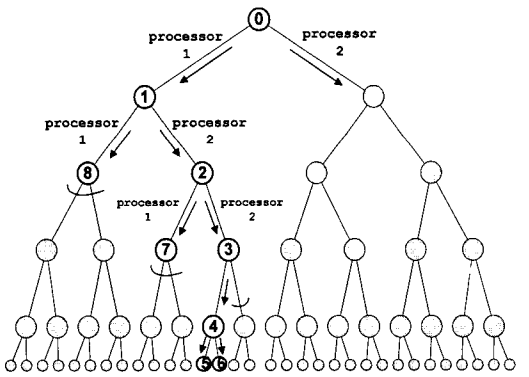

그림 9. Branch and bound 알고리즘으로 최적할당을 구하는 과정
Fig. 9. Process of searching the optimal solution using branch and bound algorithm.

그림 8은 branch and bound 알고리즘의 flow chart이다. 현재까지의 최소 simulation period, z_{op}와 그때의 assignment, A_{op}를 초기화하는 것으로 시작된다. 초기에 z_{op}는 충분히 큰 값으로 설정하면 된다. 모든 task가 할당되지 않았다면 forward search를 하여 branch중에 live vertex가 있는 지 검사하고 z가 가장 작은 branch로 이동한다. 이때 현재까지의 최소 simulation period, z_{op}보다 큰 z를 가지는 branch가 있으면 삭제(fathomed) 시킨다. 이 branch에 대해서는 더 이상의 계산이 필요 없다. 이동하면서 할당 A를 갱신하게 되고 tree의 leaf에 도달하게 되면 모든 task가 할당되었음을 의미하므로 z를 계산해서 만약 z_{op}보다 작다면 z_{op}와 A_{op}를 갱신하게 된다. forward search를 할 때 하부 branch가 모두 삭제되어 있다면 backward search에 의해 상위 vertex로 이동하여 search를 계속하게 된다. 모든 vertex가 삭제되었다면 더 이상의 search는 불필요하며 z_{op}와 A_{op}를 최적해로 출력하게 된다.

Ⅳ. 시뮬레이션 대상 화력발전소

기존의 시뮬레이터에 사용된 발전소 모델은 플랜트가 정상상태에서 동작할 때의 상태에 대해 선형화한 형태가 대부분이었으므로 시뮬레이션 할 수 있는 범위가 제한되는 단점이 있었다. 이 연구에서는 대상 발전소 모델을 위해 [5]의 내용을 기본으로 하고 정확한 시뮬레이션이 요구되는 부분은 [15]의 내용을 부분적으로 적용하여 모델을 구성하였다. 이 모델은 드럼형 보일러 발전소에 대한 모델이며, 보일러에서 터빈에 이르는 각 구성요소들을 나누어, 각각에 대해 모델식을 구했다. 따라서 계산량은 많아지지만, 각 서브 플랜트에 대한 입출력 값을 모두 얻을 수 있으며 멀티루프제어기의 적용 검증에도 적합한 형태이다. 또한 서브 플랜트에 대한 모델링이 달라지거나 다른 플랜트로 대체 될 때도 해당부분만 간단히 수정할 수 있는 장점이 있다. [5]는 비선형상미분방정식을 바탕으로 모델을 기술하였고, [15]는 비선형상미분방정식과 편미분방정식을 이용하여 모델을 기술하였다. 그림 10은 [5]에서 인용된 화력발전소의 전체적인 block diagram이며 각 서브 플랜트에 대한 수식의 통계는 표 1과 같다.

각 서브플랜트 모델을 C 언어 또는 assemble 언어로 프로그램한 후 DSP Module에 다운로드하여 execution cost를 측정할 수 있다. 이 작업은 DSP 프로세서내부에 있는 timer를 읽어서 정확하게 측정할 수 있고, 각 서브플랜트간의 통신량은 입출력 데이터량과 통신속도에 의해서 구할 수 있다. 다음은 이런 방법으로 구한 화력발전소의 task graph와 각 task의 수행비용과 통신비용이다.

화력발전소의 A/D 변환 샘플링 주기는 일반적으로 최소 0.25 sec이다. 하지만 Yoshida는 적분 시간간격이 10 msec에서 100 msec로 커짐에 따라 오차가 증

가하며, 장시간 시뮬레이션 할 경우 이러한 오차가 누적됨을 지적하였다[4].

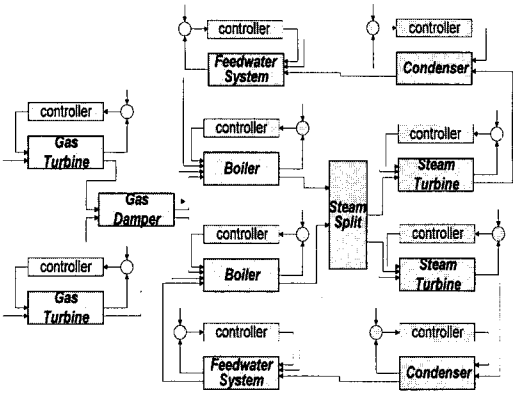

그림 10. 화력발전소의 블럭도
Fig. 10. Block Diagram of fossil power plants.

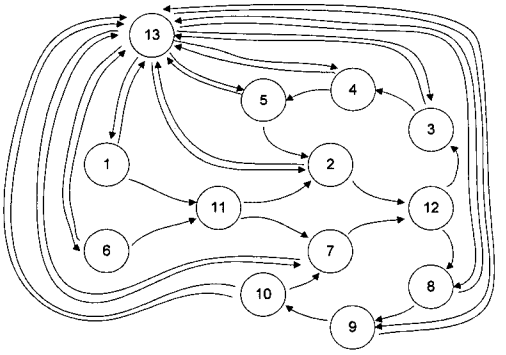

그림 11. 화력발전소의 Task Graph
Fig. 11. Task Graph of fossil power plants.

표 1. 각 서브 플랜트에 대한 수식의 구성
Table 1. The compositiion of equations for each subplant.

수식 / 플랜트	대수적 방정식	상미분 방정식	편미분 방정식
boiler	42	15	2
gas turbine	22	4	0
steam turbine	21	9	0
condenser	18	4	0
feedwater system	26	6	0
electrical generator	14	3	0
gas merge and split	5	0	0
total	296	82	4

표 2. Processor 개수에 따른 작업할당 소요시간
Table 2. Number of processors vs. task assignment time.

프로세서의 개수	number of total leaves	number of calculated leaves	할당에 소요된 시간 (sec)
1	1	1	0
2	4,096	108	0.33
3	531,441	57	0.57
4	16,777,216	1536	1.81
5	244,140,625	6	15.98

표 3. 각 Task간의 communication time
Table 3. Communication time between tasks.

From	To	Communication Time (usec)
13	1	1.6
13	2	2.0
13	3	1.4
13	4	1.2
13	5	1.4
13	6	1.6
13	7	2.0
13	8	1.4
13	9	1.2
13	10	1.4
1	13	2.0
2	13	2.0
3	13	1.6
4	13	1.2
5	13	1.4
6	13	2.0
7	13	2.0
8	13	1.6
9	13	1.2
10	13	1.4
1	11	3.8
6	11	3.8
11	2	4.2
11	7	4.2
2	12	4.0
7	12	4.0
12	3	2.6
12	8	2.6
3	4	3.6
8	9	3.6
4	5	3.2
5	2	3.4
9	10	3.2
10	7	3.4

또한 Kwan은 시뮬레이션 모델에서 주어진 미분방정식을 Runge-Kutta 방법으로 계산할 때, 안정된 해를 얻기 위해서 5 msec의 시간간격으로 적분할 필요가

있음을 지적하였다[1]. 그러므로 정확한 시뮬레이션 결과를 얻기 위해 Kwan이 제시한 시간간격으로 계산한다면 계산량이 많아지게 되고 실시간 시뮬레이션 조건을 만족시키기 힘들게 된다.

V. 결과 및 고찰

병렬 연산 및 통신을 고려한 최적할당 알고리즘을 프로그램으로 구현한 뒤 앞에서 작성된 화력발전소의 task graph의 할당결과를 살펴본다. 작업할당은 DSP 프로세서를 2~7개 사용하였을 경우에 대해 수행하여 보았다. 그림 12와 그림 13은 각각 DSP 프로세서를 2개 사용하였을 경우와 4개 사용하였을 경우에 대한 할당결과이다.

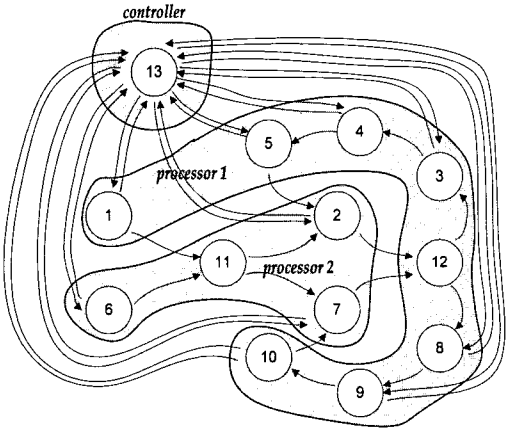

그림 12. 화력발전소의 작업에 대한 할당결과 (2개의 processor를 사용할 경우)
Fig. 12. Task assignment of fossil power plants. (using 2 processors)

표 4는 화력발전소의 task, 12개를 4개의 프로세서에 할당하는 데 소요된 시간과 계산된 할당의 개수이다. 모든 할당에 대한 simulation period를 모두 계산한 뒤 최소값을 구하는 linear search와 tree의 root에서 branch하면서 최소값을 가지는 할당을 찾는 BB algorithm을 비교하였다. linear search의 경우, 할당의 경우의 수는 16,777,216이고 이 중에서 최적해를 찾기 위해서는 6543 sec가 소요된다. 하지만 BB algorithm을 사용할 경우, 1536개의 할당만을 계산하였고 최적해를 찾는 데 1.81 sec라는 적은 시간이 소요되었다. 표 5는 프로세서 개수에 따른 작업할당 소요 시간이다. 프로세서의 개수가 증가함에 따라 작업할당에 소요되는 시간도 증가하는 경향이 있으나, BB algorithm을 통해 계산하는 branch의 개수를 많이 감소시켰음을 알 수 있다.

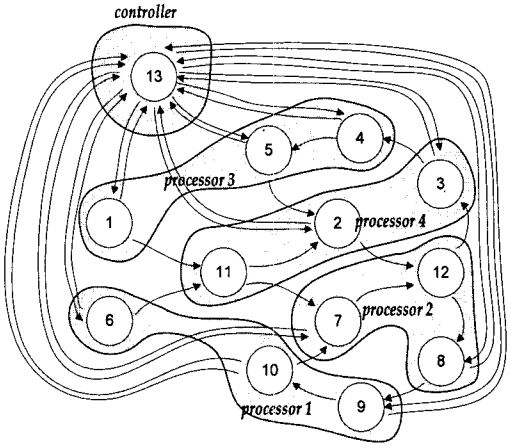

그림 13. 화력발전소의 작업에 대한 할당결과 (4개의 processor를 사용할 경우)
Fig. 13. Task assignment of fossil power plants. (using 4 processors)

표 4. 4개의 Processor에 작업할당 소요시간
Table 4. Task assignment time for four processors.

	Linear Search	BB algorithm
계산한 할당의 개수	4^{12} = 16,777,216	1536
할당에 소요된 시간	6543 sec	1.81 sec

표 5. Processor 개수에 따른 작업할당 소요시간
Table 5. Number of processors vs. task assignment time.

프로세서의 개수	number of total leaves	number of calculated leaves	할당에 소요된 시간 (sec)
1	1	1	0
2	4,096	108	0.33
3	531,441	57	0.57
4	16,777,216	1536	1.81
5	244,140,625	6	15.98

그림 14는 프로세서의 개수가 증가함에 따라 simulation period의 변화를 보여준다. 4개의 프로세서를 사용할 때까지는 simulation period가 계속 감소하지만, 5개이상의 프로세서를 사용 때 더 이상의

simulation period 감소는 없다. 이것은 task 중 계산량이 가장 많은 boiler 부분에 의한 것이다. 그림 14를 살펴봄으로서 실제 시뮬레이터를 구현할 때 5개 이상의 프로세서를 사용할 필요가 없다는 것을 알 수 있고, 요구되는 시뮬레이션 속도에 따라서 프로세서의 개수를 정할 수 있다.

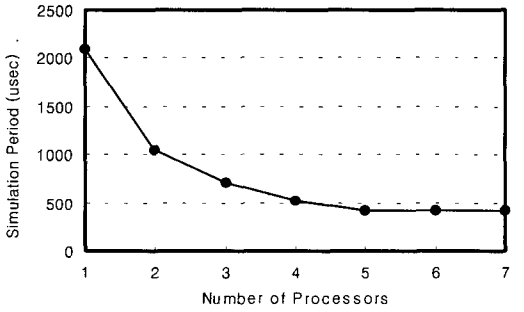

그림 14. Processor 개수 대 simulation period
Fig. 14. Number of processor vs. simulation period.

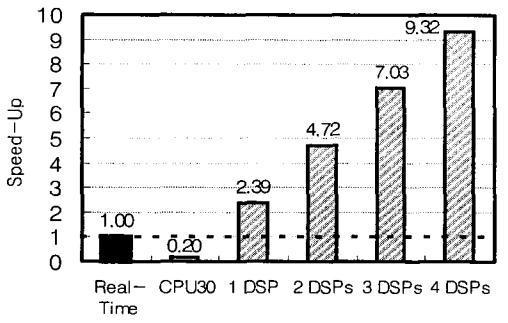

그림 15. 시뮬레이션 환경에 따른 속도 비교
Fig. 15. Simulation environment vs. speedup.

그림 15는 CPU30 board와 병렬 DSP 시스템에서 시뮬레이션을 수행할 경우, 어느 정도 속도로 시뮬레이션 할 수 있는가를 나타내는 그래프이다. CPU30 board는 VME bus상에 연결되어 사용되는 일반적인 컴퓨터 보드이며, MC68030 프로세서와 floating point coprocessor가 장착되어 있다. 시뮬레이션은 Runge-Kutta method를 이용하여 적분의 시간간격을 5 msec로 하였을 경우이다. 그러므로 5 msec마다 시뮬레이션 결과를 출력하는 속도를 real-time, 즉 1이라고 하였을 경우 CPU30과 병렬 DSP 시스템의 speed-up을 살펴보면 다음과 같다. CPU30 board는 speed-up이 0.20으로 real-time 요건을 만족시킬 수 없다. DSP module을 1개 사용할 경우는 speed-up이 2.39으로서 real-time 요건을 만족시키며, DSP module의 개수를 증가시킴으로서 시뮬레이션 속도를 향상시킬 수 있음을 알 수 있다. 특히 4개의 DSP module을 사용할 경우는 speed-up이 9.32으로서 120초 동안의 화력발전소에 대한 시뮬레이션을 12.9초만에 수행할 수 있는 속도이다. 반면에 CPU30 board를 이용할 경우 606.1초가 소요된다. 그러므로 DSP 프로세서를 이용한 병렬처리 시스템 구현으로 화력발전소를 실시간 또는 실시간이상의 고속 시뮬레이션이 가능하였다. 개발된 화력발전소 시뮬레이터는 발전소를 구성하고 있는 각 서브 플랜트 단위로 프로그래밍 하므로 관심 있는 부분에 대한 값들(온도, 질량, 부피) 등을 쉽게 알 수 있다.

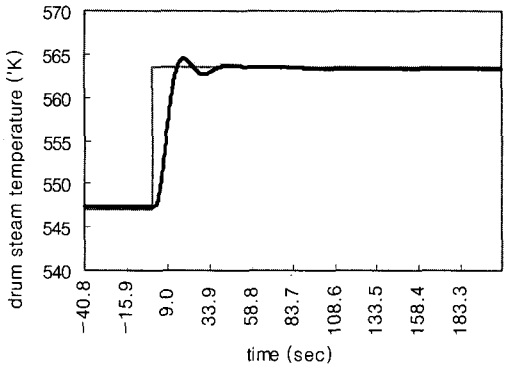

그림 16. 드럼내부 증기의 온도
Fig. 16. drum steam temperature (K).

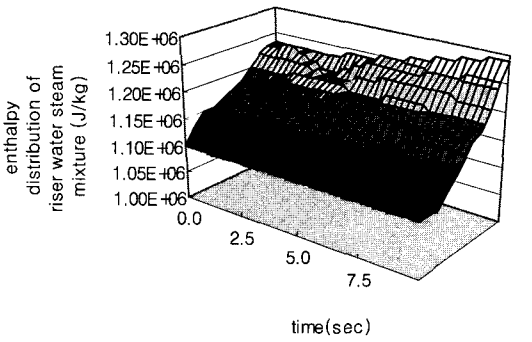

그림 17. 승수관 내부의 물과 수증기 혼합물 엔탈피 분포
Fig. 17. Enthalpy distribution of riser water steam mixture (J/Kg).

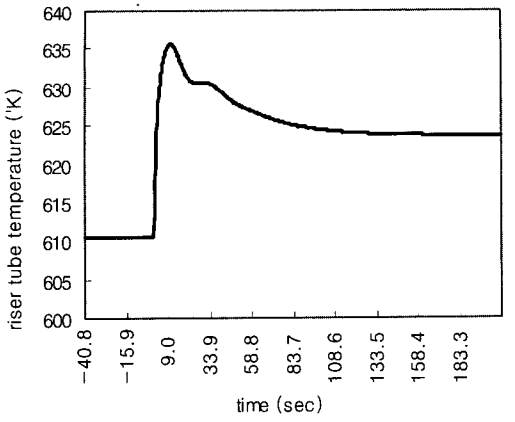

그림 18. 승수관 외벽의 온도
Fig. 18. Riser tube temperature (K).

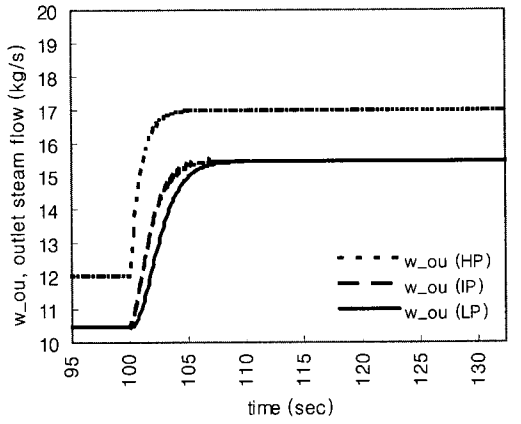

그림 19. 스팀 터빈의 유량
Fig. 19. Outlet steam flow (kg/s).

그림 16, 17, 18, 19는 화력발전소 시뮬레이션 결과 중 보일러와 스팀 터빈에 관련된 값들이다.

VI. 결 론

본 논문에서는 병렬처리를 이용하여 많은 계산량이 요구되는 화력발전소 시뮬레이션을 실시간 이상의 고속으로 수행하는 방법을 제시하였다. 병렬처리 시뮬레이터를 개발하였으며 병렬처리 환경에서의 작업할당 문제에 대한 해결방법을 제시하였다.

고속 시뮬레이터 개발에서 가장 기본적인 요건은 빠른 계산능력을 발휘할 수 있는 우수한 병렬처리 시스템의 제작이다. 병렬처리 목적으로 개발되었고 floating-point 계산에서 뛰어난 계산능력을 가지고 있는 DSP 프로세서, TMS320C40을 사용하여 DSP module과 VME interface module을 제작하여 병렬 DSP 시스템을 구성하였다. 그리고 병렬 DSP 시스템에 대한 운영환경 즉 각 프로세서에 대한 program download와 communication protocol 등을 개발하였다. 이 병렬처리 시스템은 계산량이 많고 고속의 수행시간이 요구되는 분야에 사용될 수 있다. 이 연구에서도 CPU30 Board에서 실시간 계산이 어려운 화력발전소 시뮬레이션에 대한 계산을 한 개의 DSP Module에서 수행할 수 있었으며, 여러 개의 DSP Module로 병렬처리할 경우 계산속도를 더욱 향상시켜 고속의 시뮬레이션이 가능함을 제시하였다.

병렬처리 시스템의 작업할당에 대한 연구도 수행되었다. 기존의 작업할당 연구는 각 프로세서의 수행비용과 통신비용의 합을 비용함수로 정의하여 이를 최소화하는 할당을 구하는 방향으로 진행되어 왔으나, 현재 개발된 병렬 DSP 시스템을 비롯한 많은 컴퓨터에서 제공하고 있는 병렬 연산 및 통신 기능을 고려하지 않았고, 화력발전소 task는 동기화라는 특징을 가지고 있기 때문에 현실적인 환경과는 거리가 있는 할당방법이었다고 볼 수 있다. 이 연구에서는 이러한 현실적인 조건을 고려한 모델을 정의하고 최적할당을 구하는 알고리즘을 개발하였다. 이 방법으로 얻은 할당은 병렬 연산 및 통신 기능을 충분히 고려한 할당이라고 할 수 있다. 마지막으로 화력발전소의 모델에 대한 계산을 병렬 DSP 시스템에서 수행하여 실시간 또는 그 이상의 속도로 시뮬레이션이 가능함을 보였다.

참 고 문 헌

[1] H. W. Kwan, J. H. Anderson, "A Mathematical Model of 200MW Boiler", *Int. J. Control*, vol. 12, no. 6, pp. 977-998, 1972.

[2] R. Dolezal, *Simulation of Large State Variations in Steam Power Plants-Dynamics of Large Scale Systems*, Springer-Verlag, 1987.

[3] M. Rafian, M. J. H. Sterling, "Parallel Processor Algorithm for Power System Simulation", *IEE proc*, vol. 135, no. 4, pp. 285-290, 1988.

[4] T. Yoshida, S. Kyuwa, "Operator Train-

ing Simulator with Real-time Stability Analysis", *IEEE trans. on Power Systems*, vol. 7, no. 2, pp. 721-729, 1994.

[5] A. W. Ordys, A. W. Pike, *Modeling and Simulation of Power Geration Plants*, Springer-Verlag, 1994.

[6] *TMS320C4x Parallel Processing Development System Technical Reference*, Texas Instruments, 1993.

[7] *Parallel Processing With the TMS320 C4x*, Texas Instruments, 1994.

[8] Min-You Wu, "On Runtime Parallel Scheduling for Processor Load Balancing", *IEEE trans. on Parallel and Distributed Systems*, vol. 8, no. 2, pp. 173-186, 1997.

[9] Harold S. Stone, "Multiprocessor Scheduling with the Aid of Network Flow Algorithm", *IEEE trans. Software Eng.*, vol. 3, no. 1, pp. 87-93, 1977.

[10] Paul Bay and Alexander Thomasian, "Data Allocation Heuristics for Distributed Systems", *IEEE INFOCON*, 1985.

[11] C. Murray Woodside and Gerald G. Monforton, "Fast Allocation of Processes in Distributed and Parallel Systems", *IEEE trans. of Parallel and Distributed Systems*, vol. 4, no. 2, pp. 164-174, 1993.

[12] Konstantinos Konstantinides, "Task Allocation and Scheduling Models for Multiprocessor Digital Signal Processing", *IEEE trans. on Acoustics, Speech and Signal Processing*, vol. 38, no. 12, pp. 2151-2161, 1990.

[13] Robert S. Garfinkel and George L. Nemhauser, *Integer Programming*, New York Wiley Interscience, pp. 108-153, 1972.

[14] H. M. Salkin, *Integer Programming*, Reading, MA:Addison-Wesley, 1975.

[15] 김형래, "보일러 시스템의 열수력학적 동특성 시뮬레이션", 한국과학기술원(KAIST) 기계공학과, 1995

저 자 소 개

朴 喜 晙(正會員)

1996년 2월 건국대학교 전자공학과 공학사. 1998년 2월 한국과학기술원 전기 및 전자공학과 공학석사. 1998년 3월 ~ 현재 한국과학기술원 전기 및 전자공학과 박사과정. 주관심분야는 병렬처리, 실시간 시스템, 컴퓨터 구조

金 炳 國(正會員)

1975년 2월 서울대학교 전자공학과 공학사. 1977년 2월 한국과학기술원 전기 및 전자공학과 공학석사. 1981년 2월 한국과학기술원 전기 및 전자공학과 공학박사. 1981년 ~ 1986년 우진계기(주) 연구실장. 1982년 ~ 1984년 University of Michigan, Postdoctoral research. 1986년 ~ 현재 한국과학기술원 전기 및 전자공학과 교수. 주관심분야는 실시간 시스템, 이동로봇제어, 로봇 비전, 공정 제어, 고신뢰도 제어, 지능제어