FTCS의 Multi-processor 방식 적용에 관한 연구

°문 봉 채*, 김 지 홍*, 김 병 국**, 변 증 남*
*한국과학기술원 전기및 전자공학과 **한국과학기술대학 전자 전산학부

A Study on the Implementation of a Multi-processor Scheme for FTCS

B.C.Moon*, J.H.Kim*, B.K.Kim**, Z.Bien*
*KAIST **KIT

ABSTRACT

To improve the reliability of boiler controller of a power plant, FTCS(Fault Tolerant Control System) is proposed. We studied to implement a Multi-processor scheme for FTCS. This paper presents the total system to experiment the performance of FTCS and the Multi-processor scheme implemented.

I. 서론

오늘날 시스템 기술의 발전은 시스템의 대규모화, 복잡화를 초래하였고, 이에 따라 고장 발생요인도 증가함으로써 신뢰도 문제가 중요시되었다. 신뢰도를 향상시키는 문제는 고장 발생 간격의 평균 시간(MTTF : Mean Time To Failures)을 증가시킴을 의미하고, 이의 방안으로써 고장 발생 가능성을 사전에 방지하고자 하는 fault avoidance 방법과 고장이 발생하더라도 올바른 결과를 내줄 수 있도록 redundancy 구조를 취하여 보다 적극적으로 대처하는 fault tolerance 방법이 있으며, 오늘날 전자 분야 발전으로 후자의 방법을 구현하기에 이르렀다.

본 연구를 통하여 발전소의 보일러 제어기중 아날로그 제어부를 대상으로 신뢰도를 개선시키기 위하여 FTCS(Fault Tolerant Control System)를 구성하여 이를 추가시킴으로써 신뢰도를 향상시키는 방법을 제안하였으며, FTCS는 고장 진단및 Back-up 제어 기능, Man-Machine Interface 기능을 갖는다. FTCS의 성능 실험을 위하여 보일러 프로세스및 그 아날로그 제어부를 모사하는 process simulator를 개발하여 이를 결합함으로써 전체 시스템을 구성하였다. 본 논문에서는 전체 시스템 구성의 근간을 이루는 Multi-processor 방식에 대하여 설명하고자 한다.

II. 전체 시스템의 구성

전체 시스템은 1대의 단말기, 1대의 graphic monitor, 4개의 프로세서 및 그 주변기기, I/O interface부, switching모듈, operator station등이 (그림 A)과 같이 구성된 시스템을 말한다. 전체 시스템은 크게 4개의 부분으로 나눌 수 있다

Man-machine interface부분은 전체 시스템을 관리하는 기능과 graphic diplay기능을 가진다. 1대의 단말기를 통하여 전체 시스템 동작 모드를 결정내리고, 1대의 모니터로 전체 시스템의 상태 및 보일러 아날로그 제어 현황을 graphic display로 나타내어 준다. system controller는 system reset, global interrupter, common bus 사용권 조정등의 기능을 갖고, 그 내부에 64 KByte의 common memory 를 내장하고 있어서, 이를 사용하여 프로세서간의 데이타 전달이 이루어 진다. GR(Graphic) - processor는 UNIX O/S(Operatin System)를 사용하고, graphic display 기능을 수행한다. System memory는 UNIX O/S가 booting되어 동작하는 영역이다. disk controller는 40MB hard disk 및 floppy diskette를 DMA 방식으로 access하는 기능을 갖는다.

고장 진단 부분은 FD(Fault Dignosis)-processor로 구성되고, 각 back-up 단위별로 주어진 입력에 대해 출력의 정격치를 계산한 후, 측정된 출력의 실제치와 비교함으로써 고장 판단을 내리고, 고장 발생시에는 고장 내용을 common memory를 통하여 타 processor에 전달한다.

백업 제어 부분은 BU(Back-up)-processor와 I/O interface 모듈로 구성된다. 주요 기능은 고장 발생 부위에 대한 backup제어기능으로 back-up제어 신호가 4개의 D/A conversion channel을 통해 출력되고, 4-to-43 배분기(distribution module)를 거친 후, switching 모듈에서 CONTRONIC-II의 고장 발생 신호를 대치하게된다. backup 제어 부분의 또다른 기능으로 아날로그 신호 및 디지탈 신호를 입력시켜 common

memory상에 전달하는 기능이 있다.

프로세스 시뮬레이터 부분은 PS(Process Simulator)-processor, D/A conversion 모듈 및 operator station으로 구성된다. 이 부분은 보일러 아날로그 제어기(CONTRON-II)및 보일러 프로세스의 동작을 모사(simulate)하는 기능을 가진다. 보일러 아날로그 제어 출력은 D/A conversion모듈을 통하여 switching 모듈에 연결하였다. 또한 설정치 조정, 배율기 조정, 자동/수동 동작 모드 등 operator 조작 기능을 주기 위하여 operator station을 제작하여 포함하였다. operator station에는 96개의 ON/OFF 스윗치및 24개의 배율기가 포함되어 있고, 고장 동작 실험을 위해 16개의 ON/OFF 스윗치로 구성한 FGP(Fault Generation Panel)를 포함시켰다. 따라서 16개의 스윗치들을 조정함으로써, 정상 동작및 여러 형태의 고장 동작을 실험할 수 있다.

이상의 4개 부분을 통합하여 전체 시스템을 구성하게 되고, 이렇게 구성된 전체 시스템은 stand-alone 형태로 발전소 보일러 프로세스 및 제어 시스템은 물론 본 연구를 통하여 개발된 back-up 시스템을 혼합한 실험을 수행할 수 있게 하였고, 실제 운용시에는 프로세스 시뮬레이터 부분을 대신하여 보일러 프로세스및 그 아날로그 제어부를 연결함으로써 보일러 제어의 신뢰도를 향상시킨다.

III. Multi-processor 구조

Multi-processor 구조는 일반적으로 common bus에 다수의 processor, common memory 및 공유 주변기기들이 (그림 B)와 같이 구성된다.

(그림 B)에서 같이 다수의 프로세서가 기억장치들을 공유함으로써 서로간의 데이타 전달을 수행하게 된다. 각 프로세서는 독립적인 작업 수행 능력을 갖고 있기 때문에, 여러 기능을 동시에 수행시키고자 할 때, 성능 개선을 기할 수 있다. 다수의 프로세서가 common bus 를 사용할 때, 어느 순간에도 두개 이상의 프로세서가 동시에 bus를 사용하는 혼란을 방지하여야 하므로, bus arbiter를 둠으로써 해결한다. 즉, bus 사용권을 가진 프로세서는 자신의 일을 완료한 후 bus사용을 풀어줌으로써 다른 프로세서가 bus를 사용할 수 있도록 하는 하드웨어적 처리를 bus arbitration이라하며, 본 시스템에서는 priority 방식 및 daisy-chain 방식을 혼합하여 사용하였다.

또한 다수의 프로세서가 작업을 분할 수행시 다른 프로세서의 작업 수행 여부에 따라, 작업 수행 또는 대기 상태의 판단을 내리게 되는데, 이 때 각 프로세서는 자신의 작업 수행 상태를 다른 프로세서에 전달할 필요가 생긴다. 이와 같은 각 프로세서의 작업 상태를 주고 받는 기능을 효율적으로 처리하기 위해 common memory상에 semaphore location을 둠으로써 해결하였다. 각 프로세서간의 통신 방식은 모든 프로세서가 수행해야 할 작업들을 순서별로 scheduling 한 후, 이를 각 step별로 구분하여, 각각의 step이 완료되면 다음 step의 작업을 수행하도록 한다. 이 때 다른 프로세서의 작업 완료 상태를 common memory상에 있는 semaphore location에 표시함으로써 다른 프로세서에 전달하게 된다. semaphore location확인 방법으로 timer에 의한 interrupt service 방식에 의해 수행함으로써 polling방식에 비해 프로세서 활용성을 높였다.

IV. 프로세스간의 통신

다수의 프로세서 각각의 고유 작업을 수행함은 물론, 프로세서간의 데이타 교환 또는 다른 프로세서의 현재의 작업 수행 상태를 확인할 필요성이 생긴다. 이를 위해서는 common memory상에 semaphore location, flag location 및 공유 data 위치를 고정시켜 두고, 사용시의 protocol 을 정함으로써 수행한다. Semaphore location은 7 byte로 구성한다. 이 location의 목적은 매 주기를 7가지 step으로 구분한 바, 각 step 진행의 상태를 나타내는 곳이다. 예를 들어, stepII를 진행하고 있는 경우, location II에 '$ff'라는 1 byte 내용이 들어 있으므로 모든 processor는 system의 현재 상태가 stepII를 진행하고 있음을 파악한다. Step II를 완료하고, step III를 진행할 경우에는 location II를 시스템의 상태가 stepIII를 진행함을 나타낸다. 또한 각 프로세서간에 전달을 필요로 하는 데이터는 그 위치를 common memory상에 둠으로서 해결하는 바, 이 때 프로세서별 access하는 시간의 배치문제는 혼란의 가능성을 배제하도록 구성하여야 한다.

(그림 C)는 발전소 보일러 제어기의 FTCS 기능을 수행하기 위하여 한 주기내의 7가지 step별 작업 내용을 나타낸 것이다.

Semaphore location은 7개의 byte로 구성하여, 한 샘플 주기내에서 7개로 구분된 step의 각각에 한 byte씩 대응된다. 각 step의 완료표시는 해당 위치에 set된 '$ff'를 지우고 다음 step위치에 '$ff'를 표시함으로서 나타낸다. 각 step별 표시자(marker)와 표시방법 (marking logic)에 대하여 알아보면 다음과 같다.

step I : 초기에는 PS프로세서가 location I에 '$ff'로 set.
초기이외에는
i) PS의 timer가 제로로 countdown 될 것.
ii) FD,BU,PS 모두 7번째 step이 완료되어 있을 것.

모두 만족할 때, location VII을 지우고 location I을 '$ff'로 set.

step II : BU프로세서가 stepI작업을 수행 완료 후 location I을 clear시키고, location II를 '$ff'로 set.

step III : i) FD 프로세서가 location II의 '$ff'를 확인하였음을 나타내기 위해 location III에 '$0f'를 표시.

ii) PS프로세서는 step II 작업을 수행 완료하고, location III에 '$0f'가 표시되었음을 확인한 후, location II를 clear시키고, location III를 '$ff'로 set.

step IV : BU 프로세서가 step III 작업을 수행 완료 후, location III를 clear 시키고, location IV를 '$ff'로 set.

step V : i) FD 프로세서가 location IV의 '$ff'를 확인하였음을 나타내기 위해 location V에 '$0f'를 표시.

ii) PS 프로세서는 step IV 작업을 수행 완료하고, location V에 '$0f'가 표시되었음을 확인한 후, location IV를 clear시키고, location V를 '$ff'로 set.

step VI : BU 프로세서가 step V작업을 완료한 후, location V를 clear 시키고, location VI를 '$ff'로 set.

step VII : i) PS프로세서가 location VI의 '$ff'를 확인하였음을 나타내기 위해 location VII에 '$0f'를 표시.

ii) FD프로세서는 step VI작업을 수행 완료하고, location VII에 '$0f'가 표시되었음을 확인한 후, location VI를 clear 시키고, location VII를 '$ff'로 set.

위에서 각 step별 표시 방법을 설명하였고, step I, step III, step V, step VII은 다른 step들에 비해서 좀더 복잡한 조건이 들어 있다. step I의 경우에는 매 샘플주기에서 시작하는 step이므로, i)조건은 일정한 샘플 주기를 얻기 위함이고, ii) 조건은 모든 프로세서가 전 step의 작업을 완료시킨 후에 새로운 주기를 시작하기 위함이다. Step III (또는 step V) 의 경우에는 i)조건을 무시하고 ii)를 수행해버리면 FD프로세서는 location II (또는 location IV) 에 '$ff' 가 된 상태를 확인하지 못함으로써 step II (또는 step IV) 작업을 skip해서 지나쳐 버린다. 이를 방지하기 위해 i)조건을 필요로 한다. step VII도 마찬가지로 PS프로세서가 step VI작업을 지나쳐 버림을 방지하기 위하여 필요하다.

Semaphore location을 확인하는 방법으로 각 프로세서는 local 하게 자신의 step 진행을 나타내는 step counting기능이 있어야 하고, 이를 SCi(step-count-flag)라 명명하면, SCi의 내용은 다음에 수행할 step이 들어 있어야 한다. 그럼으로써 동일한 step에서 중복 작업 수행을 방지할 수 있다. 다시말하면, SCi 부재시에는 step II 작업을 수행한 후, step II를 지우고 step III를 표시하는 프로세서가 아직 step II작업이 미 완료되었을 경우, step II가 확인됨으로써 또다시 step II 작업을 수행하므로써 반복수행을 하게 된다. SCi를 사용하여 semaphore location을 확인하고 작업 수행을 나타내는 한 예를 (그림 D)에서 flow chart 형태로 보여준다.

V. 결론

본 연구를 통하여 Multi-processor 방식을 적용, 실험함으로써 다음과 같은 결론을 내린다.

1. Computer control 에서 Multi-processor 방식을 적용함으로써 속도 개선을 이루고, 효과적인 Real-Time 적용이 가능함을 확인하였다.

2. 컴퓨터 분야에서의 Fault Tolerant Computing 기술을 제어 시스템의 신뢰도 향상을 위한 FTCS에 적용하는 기본 기술을 습득하였다.

3. 다수의 프로세서가 독립적인 기능을 수행할 수 있으므로 신뢰도 개선을 위한 Redundancy 구성이 용이하다.

4. 현재 구현된 Multi-processor 방식은 하나의 Common Bus를 갖는 구조를 취하기 때문에 이 Bus를 얼마나 효율적으로 사용하는가 하는 문제가 속도 개선의 관건이 된다. 또한 다수의 프로세서가 독립적인 기능을 수행하면서, 다른 프로세서의 작업 수행 여부에 따라 대기 상태에 있는 시간이 존재하므로 이 시간을 줄이는 문제에 관한 연구를 필요로 한다.

참고 문헌

[1] Maurice J. Bach, The Design of UNIX Operating Systems, Prentice-Hall, 1986.

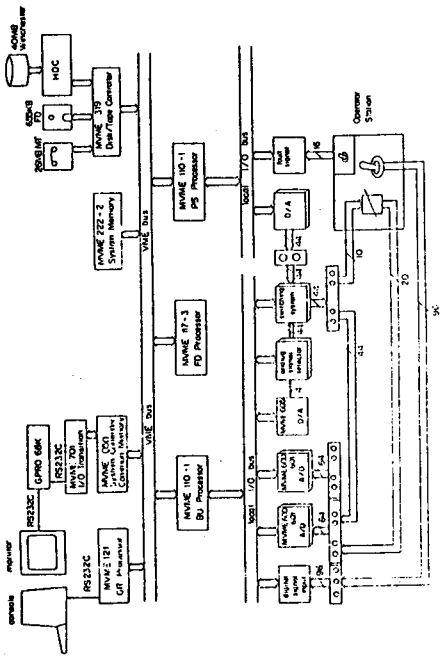
[2] Cay Weitzman, Distributed Micro/Minicomputer Systems, Prentice-Hall, 1978.

[3] G. Klefenz, Automatic Control of Steam Power Plants, Bibliographisches Institut, 1986.

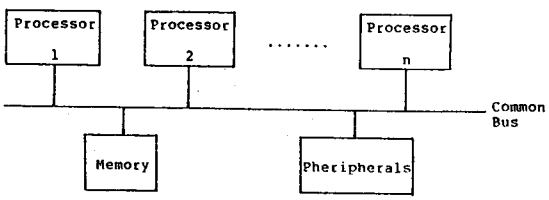
[4] 이헌, "Fault Tolerant Computing Systems," 전자교환기술 제 1 권 제 1 호, 1985.

[5] 김지홍 외 3인, "발전소 보일러 제어기에 적용한 Fault Tolerant Control System에 관한 연구," 대한 전자공학회지 제 24 권 1 호, 1987.

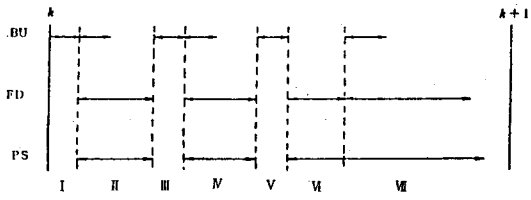
[6] VMEbus Specification Manual, Motorola.



(그림 A) 전체 시스템 구성도



(그림 B) Multiprocessor의 구조

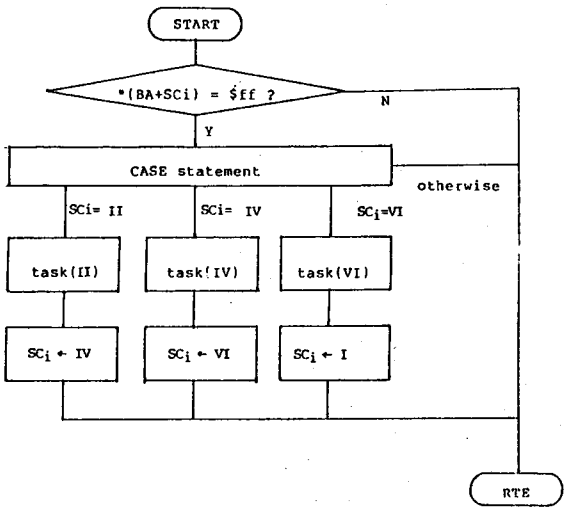


TIMING SCHEDULE IN SAMPLING PERIOD

JOB SCHEDULE IN SAMPLING PERIOD

STEP	FD	BU	PS	MARKER
I		(BU-I) data acq. for DI		BU
II	(FD-II) computation of Master nominal values	(BU-II) backup PROCESSING for master part	(PS-II) master controller	PS
III		(BU-III) data acq. for MV_k		BU
IV	(FD-IV) computation of slave nominal values	(BU-IV) backup PROCESSING for slave part	(PS-IV) slave controller	PS
V		(BU-V) data acq. for SV_k		BU
VI	(FD-III) detection		(PS-VI) boiler process	FD
VII	(FD-VII) diagnosis	(BU-VII) backup initializing and switching		BU

(그림 C) 제어 프로그램의 Time Schedule 및 Job Schedule



(그림 D) PIM interrupt service routine의 한 예

*BA = (int *) $FD0000 : base address of semaphore location
SCi = I, II, ... , VII