論文 90-27-10-10

다중처리기 시스템의 시뮬레이션에 관한 연구

(A Study on Simulation of A Multiprocessor System)

朴 贊 政,* 申 仁 澈,** 李 相 範**

(Chan Jung Park, In Chul Shin, and Sang Burm Rhee)

要 約

본 논문은 다중 버스 상호 접속망을 갖는 다중처리기 시스템에서, 기억장치 접근 요구의 경쟁에 의하여 영향을 받는 시스템의 성능을 평가하기 위하여 이산 사건 모델을 구성하였다. 또한 시스템의 해석적 모델과 시뮬레이터 모델을 구성하여 해석적 모델의 결과와 시뮬레이터 모델의결과를 상호 검증하였다.

검증 방법으로는 프로세서의 수, 기억장치 모듈의 수, 버스의 수와 국부 기억장치 실패율을 입력인수로 하여 기억장치 밴드폭, 프로세서, 기억장치 모듈 및 버스의 이용율, 버스 상호 충돌의 정도를 결정할 수 있었다. 따라서 시스템을 설계할 때 시뮬레이션을 통하여 입력인수의 상호작용을 해석함으로써 시스템의 성능을 평가할 수 있게 된다.

Abstract

To evaluate the performance of a multiprocessor system, a discrete event model of memory interference in the system employing multiple-bus interconnection networks is proposed. An analytic model of the system is presented and then simulator models are implemented for cross-verifying the analytic results and simulation results.

The simulator model takes as input the number of processors, the number of memory modules, the number of buses and the local memory miss ratio. The model produces as output the memory bandwidth, the processor, memory module and bus utilization and the bus contention ratio. Using the model in the design of the system, it is possible to evaluate the system performance by analyzing the interaction of the input parameters.

I. 서 론

최근에 반도체 기술이 급속히 발달됨에 따라, 시스템의 성능을 향상시킬 수 있는 다중처리기 시스템에 대한 연구가 활발히 진행되고 있다.[1,2]

초기의 다중처리기 시스템들은 크로스바 스위치로 연결되도록 개발되어졌으며, 대표적인 것으로 C.mmp가 있다.[3] 크로스바 스위치는 뛰어난 정보교환 능력이 있으나 프로세서의 수가 증가하면 하드웨어가 매우 복잡하게 되어 경제성이 저하된다. 현재 다중처리기 시스템에서 많이 사용되고 있는 상호접속망(interconnection network)은 공유버스(shared bus)이다. 공유버스는 간결성 및 확장성을 가지나 버스의 이상이 생기면 전체 시스템의 수행을 중단시키고 버스 경쟁에 따른 성능저하를 초래하기 때문에 연결

*正會員, 江陵大學 電子計算學科
(Dept. of Comput. Sci., Kangnung Nat'l Univ.)
**正會員, 檀國大學校 電子工學科
(Dept. of Elec. Eng., Dankook Univ.)
接受日字 : 1990年 8月 6日

가능한 프로세서의 수가 제한된다.[4] 따라서 하드웨어의 복잡성을 감소시키고 원하는 정보교환 능력을 충분히 갖는 다중버스(multiple-bus)방식이 제시되고 있다.[5]

다중버스 방식의 다중처리기 시스템은 다른 연결방식에 비하여 하드웨어면에서 쉽게 구현될 수 있으며 경제성이 향상된다. 또한 응용 프로그램의 특성에 따라 요구되는 밴드폭(bandwidth)을 프로세서와 기억장치 모듈, 입출력 장치의 수에 적합하게 결정할 수 있는 등 시스템구성의 융통성이 높은 장점이 있어 많은 다중처리기 시스템에서 채택되고 있다.[5,6,7] 그러나 버스를 통해서만 자원의 접근이 가능하므로 버스의 전송속도에 따라 시스템의 성능이 좌우되고, 버스가 포화되면 프로세서의 수를 증가시켜도 성능의 향상이 선형적으로 증가되지는 않는다.[8,9]

본 연구에서는 각각의 프로세서가 국부 기억장치(local memory)를 가지며 다중버스 상호접속망 으로 구성된 느슨한결합(loosely coupled)다중처리기 시스템의 구조를 제안하였다. 또한 이러한 시스템의 기억장치 밴드폭, 프로세서와 버스의 이용율, 버스의 상호 충돌 정도 등에 대하여 시뮬레이션을 통하여 시스템의 성능을 평가하였다. 이때 여러가지 시스템의 설계인수(design factor) 즉, 프로세서의 수, 기억장치 모듈의 수, 버스의 수, 국부 기억장치 실패율 등을 변화함에 따라 일어나는 영향을 정량적으로 나타내어 시스템의 성능을 평가하고, 버스의 수를 최적화하였다.

Ⅱ. 다중버스 시스템 구조

1. 시스템 구성

다중처리기 시스템은 여러개의 프로세서가 기억장치 및 주변장치를 공유하면서 동일한 또는 서로 다른 데이타나 명령으로 공간적 병렬성을 갖고 연산및 정보처리를 실행하여, 대량의 데이타를 고속으로 처리함으로써 전체 시스템의 성능을 향상시키게 된다.[10]

본 연구에서 제안된 다중처리기 시스템은 국부 기억장치를 갖고 있는 P개의 프로세서와 M개의 기억장치 모듈이 B개의 다중버스로 연결되어 있는 P×M×B 시스템의 기본 블럭도는 그림 1과 같다.

P×M×B 시스템은 기억장치 모듈과 버스의 공유성에 의하여 기억장치 접근의 경쟁(memory access contention) 또는 데이타 충돌(data conflict)이 발생하게 되며, 현재 busy 상태인 자원을 요구하는 프로세서는 데이타 처리가 가능할 때까지 대기된다. 따라서 다중버스 구조를 갖는 다중처리기 시스템에서 버스수의 결정은 설계시 성능 대 가격비(performance to cost ratio)에 영향을 준다.

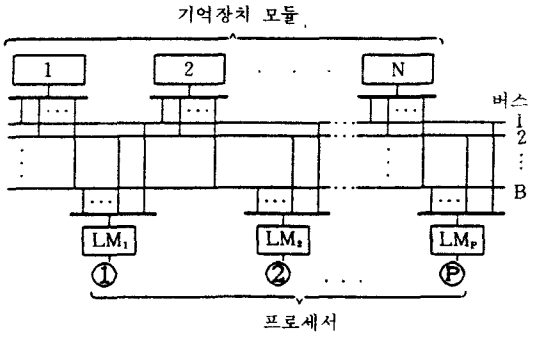

그림 1. 다중처리기 시스템
Fig. 1. A Multiprocessor system.

P×M×B 시스템은 한개의 버스 사이클을 기본 단위시간(unit time)으로 하여 동기적으로 동작하며, 각 프로세서는 임의의 기억장치 모듈에 접근할 수 있다. 이때 기억장치 모듈의 접근요구는 한 사이클의 동일 시점에서 동시에 초기화되며, 프로세서와 기억장치 모듈사이의 정보교환은 한개의 버스 사이클 동안에 수행된다. 이때, 시스템의 동작은 다음과 같은 가정을 하기로 한다.

[가정 1] 프로세서는 물리적이나 기능적으로 모두 동일하다.

[가정 2] 국부 기억장치 적중율(hit ratio)을 h, 기억장치 모듈의 접근요구율을 m이라하면 m=1−h가 된다. 이때 m를 국부 기억장치 실패율(miss ratio)이라 한다.

[가정 3] 프로세서의 기억장치 모듈 접근 요구는 매 버스 사이클에서 발생한다.

[가정 4] 프로세서의 기억장치 모듈에 대한 접근요구는 기억장치 인터리빙을 고려하여 독립적인 균등분포(uniformly distribution)를 이룬다고 본다.

[가정 5] 프로세서는 동일한 우선순위로 기억장치 모듈과 버스를 할당받는다.

각 프로세스의 기억장치 모듈 접근요구 처리는 다음 단계에 따라서 수행된다.

[단계 1] 프로세서는 버스 사이클 시점에서 기억장치 모듈 접근요구를 발생하고 수행을 중지한다. 한 모듈에 여러개의 접근요구가 동시에 발생하면, 접근요구를 발생하는 프로세서들을 라운드 로빈(round-robin) 원리로

주기적으로 스캔(scan)하는 조정기법에 의하여 적절한 서비스가 제공된다.[11,12] 접근요구가 성공되면 모듈이 해당 프로세서에 예약되며, 실패하면 다음 사이클의 시점에서 재발생된다.

[단계 2] M개의 기억장치 모듈 접근요구가 성공되면, 두번째 조정기법에 의해 B개의 버스 접근요구를 선택하여 버스가 예약되어 진다. 이러한 동작은 모듈이 예약되는 것과 동일한 방법으로 이루어진다. 버스 접근요구가 실패하면, 예약된 모듈은 해제되며 다음 사이클의 시점에서 재발생된다.

[단계 3] 성공이 된 접근요구의 자료 전달 동작은 해당 사이클의 끝점에서 이루어진다.

[단계 4] 접근요구가 성공적으로 완료된 프로세서는 일시 정지한 수행을 다시 시작하며, 예약되었던 버스와 모듈은 해제된다.

2. 해석적 모델

시스템의 성능을 평가하는 여러방법 중에서 중요한 성능평가 지표가 되는 것은 밴드폭(BW;bandwidth)이다.

제안된 시스템의 정보전달은 1사이클 타임에서 이루어져야 하므로 전체 버스의 이용율은 BW와 일치하게 되며, BW는 평균 busy 버스의 수로 정의될 수 있다.

[정리 1]

P×M×B 시스템의 각 버스 사이클에서 기억장치 모듈 접근요구율 m이 일정하고 독립적이면, 프로세서의 수행간격은
평균 $X=(1-m)/m$, 분산 $\sigma^2=(1-m)/m^2$인 기하분포이다.

[증명]

접근요구율 m이 일정하고 독립적이므로 1회 접근요구발생은 베르누이 시행(bernoulli trials)이 되며, 프로세서의 수행간격은 첫번째 접근요구가 발생될 때까지의 시행횟수이다. 따라서 프로세서의 수행간격은 베르누이 시행의 반복시행이며 첫번째 접근요구가 발생될 때까지의 시행횟수의 분포는 기하분포가 된다.[13] Q, E. D.

정리 1에서 다음을 얻을 수 있다.

[정리 2]

P×M×B 시스템에서 접근요구가 재발생되지 않는 사이클이 충분히 길게 계속되면, 임의의 기억장치 모듈에 적어도 하나의 접근요구가 존재할 확률 q는 다음과 같다.

$$q=1-(1-m/M)^P$$

정리 2로 부터 기억장치 모듈 M개 중 i개가 접근요구될 확률 f_i는 다음과 같은 이항분포가 된다.

[정리 3]

$$f_i=\binom{M}{i}\, q^i(1-q)^{M-i}$$

이때 서로 다른 모듈이 접근요구될 기대값은 q·M이다. 한 사이클에서 버스 조정기법에 q·M개의 접근요구가 주어지면, (B+1)개 이상의 접근요구는 허용되지 않으므로 한 사이클에서 허용되는 버스 접근요구의 기대값은 BW와 일치하며 다음과 같게 된다.

[정리 4]

정리1, 2, 3이 성립할 때 한 사이클에서의 밴드폭 BW는 다음과 같다.

$$BW=\sum_{i=1}^{B-1} if_i+B\sum_{i=B}^{M} f_i$$

정리 4에서 구해지는 BW는 어떠한 접근요구도 재발생되지 않는 사이클에서만 성립한다.

일반적으로 전체 사이클에 대하여 생각하면 각 프로세서의 접근요구 발생율 r은 국부 기억장치 실패율 m보다는 크게 된다. 그 이유는 초기에 발생되는 접근요구와 재발생되는 접근요구를 포함하기 때문이다. 이러한 접근요구 발생율 r을 추정하여 정리 2의 m 대신에 r을 사용함으로써 개선된 BW를 추정할 수 있다. 성공적으로 완료된 두개의 접근요구 사이에 시간간격을 살펴보자. 그림 2에서 시간간격의 평균 길이를 T라 하자. T는 수행구간, 대기구간, 전달구간으로 구분되며, 수행구간의 평균 길이는 $X=(1-m)/m$, 대기구간의 평균 길이는 b, 전달구간은 1사이클이 된다. 여기서 b는 접근요구가 실패되었다가 재발생되어 성공이 되는 사이클까지의 대기시간을 나타낸다.

따라서 각 프로세서의 접근요구 발생율 r(대기 또는 성공된 접근요구 모두 포함)은 다음과 같다.

$$r=(b+1)/T=(b+1)/(X+b+1)=[1+X/(b+1)]^{-1}$$

각 프로세서들의 접근요구 완료비율은 BW/P가되므로, 완료된 두 접근요구 사이의 평균 시간간격은

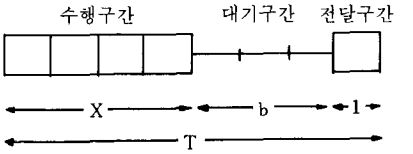

그림 2. 접근요구 시간간격
Fig. 2. Processor inter-request interval timing.

$T=P/BW$가 되어 $b+1=T\cdot r=P\cdot r/BW$가 된다. 앞 식의 $(b+1)$에 이 값을 대입하면 다음과 같은 정리를 얻게 된다.

[정리 5]

$$r=[1+X\cdot BW/(P\cdot r)]^{-1}$$

정리 5에서 r이 r의 함수로 표현되어 있으므로 BW의 추정은 그림 3과 같은 고정점 반복법 알고리듬(fixed-point iterative algorithm)으로 구할 수 있다.

단계 1. 정리2, 3, 4를 사용하여 초기 밴드폭의 추정량 BW_0를 계산하고 $r_0=p$라 정의한다.
단계 2. $r_i=[1+X\cdot W_{i-1}/(P.r_{i-1})]^{-1}$
단계 3. $q=1-(1-r_i/M)^P$
단계 4. 정리3, 4와 q를 사용 추정량 BW_i를 구한다.
단계 5. $|BW_i-BW_{i-1}|<\varepsilon$이면 종결
$|BW_i-BW_{i-1}|\geq\varepsilon$이면 단계2부터 반복
단, ε은 원하는 정밀도 또는 종결조건

그림 3. 밴드폭 추정 알고리듬
Fig. 3. Algorithm for estimated bandwidth.

지금까지 연구한 밴드폭에 대한 해석은 근사해석이다. 그 이유는 기억장치 모듈 접근요구율이 정리1에서 일정하고 독립이라고 가정했기 때문이다. 이것은 모듈 j에 대한 접근에서 대기되었던 접근요구가 재발생될 때 M개의 모듈중에서 임의로 한 모듈이 지정된다고 가정한 것을 의미한다. 실제 시스템에서는 모듈 j에 대기된 접근요구는 동일한 모듈 j에 재발생되기 때문에 n번째 사이클에서 모듈 j를 참조할 확률은 $(n-1)$번째 사이클에서 모듈 j를 참조했는가에 의하여 영향을 받으므로 독립이라고 할 수가 없다.

Ⅲ. 시뮬레이터의 구성

제안된 P×M×B 시스템의 시뮬레이션을 위하여 본 연구에서는 두개의 시뮬레이터 MODEL1, MODEL2를 구성한다.

시뮬레이터 MODEL1은 앞에서 언급한 해석적 모델의 결과가 시스템 동작을 정확하게 반영하고 있는가를 확인하기 위한 것으로, 해석적 모델을 반영한 시스템 구조를 나타내도록 시뮬레이터를 구성하고 시뮬레이션 결과를 해석적 모델의 결과와 비교함으로써 상호 검증(cross verifying)을 한다.

시뮬레이터 MODEL2는 큐잉모델(queueing model을 이용하여 시스템의 동작특성을 실제 시스템에 가깝게 나타내도록 구성하였다. 즉 MODEL2는 MODEL1과는 달리 기억장치 모듈j에 대한 접근에서 대기되었던 접근요구가 재발생될 때 동일한 모듈 j가 유지되도록 구성한다.

두개의 시뮬레이터는 그림 4와 같이 초기화부분, 사건처리부분, 결과출력부분 등세부분으로 구성된다.

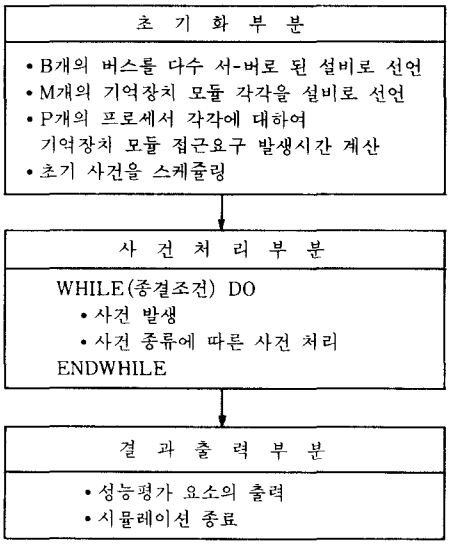

그림 4. 시뮬레이터의 기본구성 및 제어구조
Fig. 4. Block diagram of simulator and control structure.

1. 시뮬레이터 MODEL 1의 구성

시뮬레이터 MODEL1은 동기화 되어 있으며, 시간의 단위는 버스 사이클 타임이다. 초기화부분의 구성은 그림5와 같다.

초기화부분에서 제어를 넘겨 받은 사건처리부분에서는 사건을 발생시켜 사건의 종류에 따라 처리루틴을 호출하여 처리한다.

MODEL1에서 일어나는 사건은 모두 세가지로 사건1, 사건2, 사건3이 있으며, 시간처리부분의 구성은 그림6과 같다.

1) 사건 1의 처리

함수 begin-event( )이 호출되며, 그 알고리듬은 그림7과 같다.

2) 사건 2의 처리

함수 req-event(n)가 호출되며, 그 알고리듬은 그림8과 같다.

3) 사건 3의 처리

함수 end-event(n)를 호출하며, 그 알고리듬은 그

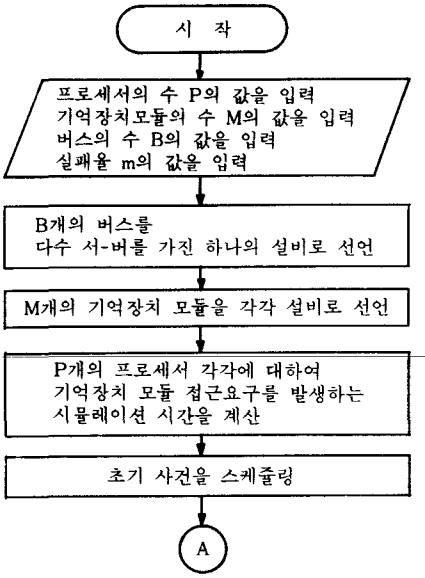

그림 5. MODEL1의 초기화부분
Fig. 5. Initialization part of MODEL1.

```
FUNCTION begin-event( );
begin
 n을 스캔의 시작점 next로 지정;
 다음과 같이 P개의 프로세서를 스캔;
   for i:=1 to P do
    if(프로세서 n에 기억모듈이 할당안 된 경우)
     then
      if(프로세서 n의 접근요구 발생시간
       =최초 접근요구 시뮬레이션 시간)
        then
         기억장치 모듈을 할당;
         다음 사건(사건2)을 스케쥴
        else
         다음 스캔의
         최초 접근요구 시뮬레이션 시간 계산
      endif;
    endif;
    n을 1;{end of for}
 다음 스캔의 시작점 next를 계산;
 return;
end;{end of begin}
```

그림 7. 함수 begin-event( )의 알고리듬
Fig. 7. Algorithm of function begin-event( ).

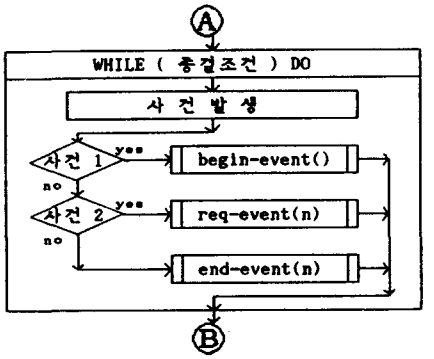

그림 6. MODEL1의 사건처리부분
Fig. 6. Event processing part of MODEL1.

```
FUNCTION req-event(n);{n:프로세서의 토큰번호}
begin
 if( 프로세서 n에 할당된 기억장치 모듈이 idle이고
   버스가 idle)
 then
   할당된 기억장치 모듈을 예약;
   버스를 예약;
   busy 버스의 수를 1증가;
   다음 사건(사건3)을 스케쥴
 else
   기억장치 모듈의 할당을 무효화;
   접근요구 발생시간을 1증가;
   새로운 최초 접근요구 시뮬레이션 시간 계산
 endif;
 return;
end;{end of begin}
```

그림 8. 함수 req-event(n)의 알고리듬
Fig. 8. Algorithm of function req-event(n).

림 9와 같다.

사건처리 부분의 종결조건이 만족되면 결과 출력 부분이 수행되어 성능 측정 자료값들을 출력하고 시뮬레이션이 종료된다.

2. 시뮬레이터 MODEL 2의 구성

기억장치와 버스에 큐를 갖는 동기 큐잉모델(synchronous queueing model)인 시뮬레이터 MODEL2는 몇가지 점을 제외하고는 시뮬레이터 MODEL1과 그 구조가 같다.

MODEL1과 MODEL2의 차이점은 MODEL2가 기억장치 모듈과 버스에 큐를 가지고 있기 때문에 기억장치 모듈의 접근요구와 버스의 접근요구를 분리하여 사건 2와 사건3에서 처리하도록 한다.

초기화부분은 그림5의 MODEL1과 같으며, 사건처리부분의 구성은 그림10과 같다.

1) 사건 1의 처리

사건 1의 처리는 그림 7과 같으며 MODEL1과 동일하다.

```
FUNCTION end-event(n);{n:프로세서의 토큰번호}
begin
  프로세서 n에 대하여
    (1) 예약된 버스를 해제;
    (2) 할당된 기억장치 모듈을 해제;
  성능 측정 자료값들을 갱신;
  프로세서 n에 대하여
    (1) 기억장치 모듈의 할당을 무효화;
    (2) 다음 접근요구 발생시간을 계산;
  최초 접근요구 시뮬레이션 시간을 계산;
  현재 사이클에서 busy 버스의 수를 1감소;
  if(busy 버스의 수=0)사건1을 스케줄
  return
end;{end of begin}
```

그림 9. 함수 end-event(n)의 알고리듬
Fig. 9. Algorithm of function end-event(n).

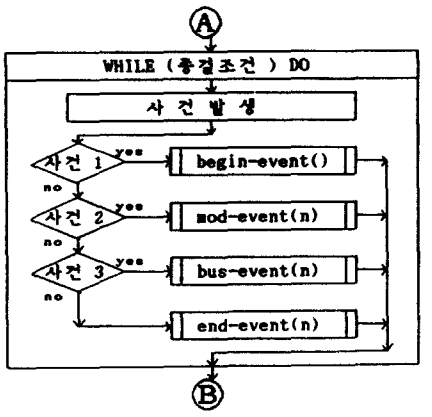

그림10. MODEL 2의 사건처리부분
Fig. 10. Event processing part of MODEL2.

2) 사건 2의 처리

함수 mod-event(n)이 호출되며 그 알고리듬은 그림11과 같다.

3) 사건 3의 처리

함수 bus-event(n)이 호출되며, 그 알고리듬은 그림12와 같다.

4) 사건 4의 처리

함수 end-event(n)이 호출되며, 그 알고리듬은 그림13과 같다.

Ⅳ. 시뮬레이션

사건중심(event - oriented) 시뮬레이션 언어인 SMPL[14]을 시뮬레이션 도구로 IBM-PC/AT(CPU: intel 80386)에서 C언어를 주언어로 하여 시뮬레이터

```
FUNCTION mod-event(n);{n:프로세서의 토큰번호}
begin
 if(프로세서 n에 할당된
    기억장치 모듈의 상태=idle)
  then
    할당된 기억장치 모듈을 예약;
    다음 사건(사건3)을 스케줄
  else
    할당된 기억장치 모듈의 큐에 대기
 endif;
 return;
end;{end of begin}
```

그림11. 함수 mod-event(n)의 알고리듬
Fig. 11. Algorithm of function mod-event(n).

```
FUNCTION bus-event(n);{n:프로세서의 토큰번호}
begin
 if(사용가능한 버스가 있으면)
  then
    버스를 예약;
    busy 버스의 수를 1증가;
    다음 사건(사건4)를 스케줄
  else
    버스 큐에 대기
 endif;
 return;
end;{end of begin}
```

그림12. 함수 bus-event(n)의 알고리듬
Fig. 12. Algorithm of function bus-event(n).

```
FUNCTION end-event(n);{n:프로세서의 토큰번호}
begin
 프로세서 n의 접근요구가 완료 되었음을 표시;
 busy 버스의 수를 1감소;
 if(busy 버스의 수=0)
  then
   for n:=1 to P do
    if(프로세서 n의 접근요구가 완료)
     then
      예약된 버스를 해제;
      버스에 대한 성능 측정자료값을 갱신;
      예약된 기억장치 모듈을 해제;
      기억장치 모듈의 성능측정 자료값을 갱신;
      기억장치 모듈의 할당을 무효화;
      다음 접근요구 발생시간을 계산;
      최초 접근요구발생 시뮬레이션시간 계산
     endif;
      사건1 스케줄{새로운 사건을 발생}
     {end of for}
    endif;
    return;
   end;{end of begin}
```

그림13. 함수 end-event(n)의 알고리듬
Fig. 13. Algorithm of function end-event(n).

를 구현하였다.

시뮬레이션 언어 SMPL은 스케쥴링, 통계수집, 확률변수생성 등에 관한 함수의 제공과 시뮬레이션 시간의 전진 및 사건발생의 순서를 조절해주는 기능을 제공한다.

밴드폭, 기억장치의 이용율, 버스 및 프로세서의 평균이용율, 버스 상호충돌 정도를 성능평가의 요소로 하였으며 그 정의는 다음과 같다.

① 밴드폭: $BW = \dfrac{\text{접근요구 완료횟수}}{\text{총실행사이클수}}$

② 기억장치 모듈이용율: $U_M = \dfrac{\text{busy 시간의 합}}{\text{총실행사이클수}}$

③ 버스 평균이용율: $U_B = \dfrac{\text{밴드폭}}{\text{버스의 수}}$

④ 프로세서 평균 이용율:

$$U_P = \frac{\text{수행구간의 총합}}{(\text{총실행사이클수} \times \text{프로세서수})}$$

⑤ 버스상호충돌정도: $C_B = \dfrac{\text{대기된 접근요구 수}}{\text{접근요구 완료횟수}}$

1. 해석적 모델과 시뮬레이터 모델의 상호검증

해석적 모델을 검증하기 위하여 연구한 인수의 범위는 프로세서의 수를 $2 \le P \le 16$, 기억장치 모듈의 수를 $P/2 \le M \le P$, 버스의 수를 $1 \le B \le M$, 국부기억장치 실패율은 $0.1 \le m \le 1$로 하였다. 이러한 인수들의 범위내에서 검증하고자 할 때 m의 값을 0.1씩 증가시킨다고 하면 P=2인 경우에 30번의 시뮬레이션이 수행되어야 한다. 만일 P=16인 경우를 생각하면 매우 많은 횟수의 수행이 요구되어진다.

이를 해결하기 위한 방법을 구하기 위하여 해석적 모델의 결과에서 인수값의 변화에 따라 밴드폭 BW의 변화를 고찰하여 보자. 예를 들어, P=M=16, m=0.5일 때 버스의 수 B에 대한 밴드폭 BW의 변화를 구하여 그림14에 나타내었다.

그림14에서 버스의 수 B가 1에서 5까지 변할 때, 밴드폭 BW의 변화는 BW≃B가 되어 선형적으로 증가한다. 이때 밴드폭 BW는 버스의 수 B에 영향을 받게 됨을 알 수 있다. 그러나 버스의 수 B가 증가되면 밴드폭 BW는 버스보다는 기억장치 모듈의 수에 크게 제한을 받게 된다. 즉 BW가 버스에 제한을 받는 영역($1 \le B \le 5$)과 기억장치 모듈에 제한을 받는 영역($10 \le B \le 16$)사이에 있는 영역($6 \le B \le 9$)에서 검증 작업이 중점적으로 이루어져야 한다.

해석적 모델과 시뮬레이터 모델에 대하여 인수값

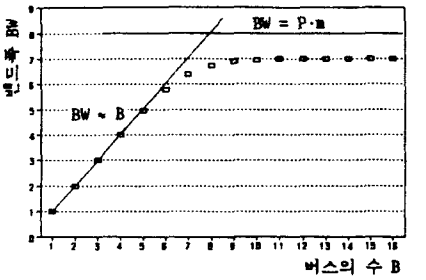

그림14. 버스 대 밴드폭(P=M=16, m= 0.5)
Fig. 14. Bus vs. bandwidth for P=M=16, m=0.4.

의 변화에 따른 시뮬레이션 수행 결과를 구하여 표1에 보였다. 표1에서 ANALTIC열은 해석적 모델의 밴드폭, MODEL1, MODEL2열은 해당되는 시뮬레이터 모델의 밴드폭을 나타내며, MD1-ANAL, MD2-ANAL, MD2-MD1열은 밴드폭의 증감을 백분율로 나타낸다. 인수들의 값은 P, M, m의 값을 먼저 선택한 후에 B의 값은 P, m으로 결정하였으며, 이는 밴드폭의 변화가 중점적으로 조사되는 부분의 값이 된다.

해석적 모델은 기억장치 모듈의 접근요구에 대하여 임의선택을 하며, 대기된 접근요구의 재발생시에도 임의선택한다. 시뮬레이터 MODEL1은 기억장치 모듈의 접근요구를 발생하는 프로세서를 라운드 로빈 스캔 기법으로 스캔하여 임의 선택에 가까운 구조로 구성되었으며 대기된 접근요구의 재발생시에도 동일하게 처리한다. 시뮬레이터 MODEL2는 FIFO (first-in first-out)큐를 이용하여 기억장치 및 버스의 접근요구를 처리하도록 구성하였다. 따라서 처음에 접근요구한 기억장치 모듈이 계속 유지되어진다. 실제 다중처리기 시스템에서의 동작은 MODEL2로 간주될 수 있다.

표1에서 해석적 모델과 시뮬레이터 MODEL1 의 밴드폭을 살펴보면 5% 이내의 편차로서 근사적으로 일치하고 있음을 알 수 있다. 한편 MODEL2의 밴드폭을 해석적 모델이나 MODEL1의 밴드폭과 비교하면 m의 값이 클 때 7%이내의 감소를 나타내고 있으나, m의 값이 작아짐에 따라 그 차이는 줄어들고 있다. 특히 버스의 수 B가 1인 경우에는 급격한 증가를 나타내고 있다. 이러한 현상은 MODEL2가 처음에 접근요구된 기억장치 모듈을 계속 유지하고 있기 때문이다.

해석적 모델에서 각 프로세서들은 동일한 우선 순위를 가지며 기억장치 모듈에 대한 접근요구는 기억장치 모듈 전체에 대하여 독립 균등분포로 가정하였

표 1. 해석적 모델과 시뮬레이터 모델의 밴드폭
Table 1. Bandwidth of analytic and simulator model.

P	M	B	m	ANALYTIC	MODEL1	MODEL2	MD1-ANAL	MD2-ANAL	MD2-MD1
16	16	16	1.000	10.3028	10.3011	9.5874	−0.0165	− 6.9437	− 6.9284
		12	0.750	9.0019	9.0576	8.6317	0.6188	− 4.1125	− 4.7021
		8	0.500	6.7137	6.8393	6.7274	1.8708	0.2041	− 1.6361
		4	0.250	3.5547	3.5977	3.6011	1.2097	1.3053	0.0945
		2	0.125	1.8019	1.7920	1.8049	−0.7147	0.0000	0.7199
16	8	8	0.500	5.8631	5.8667	5.4949	0.0614	− 6.2800	− 6.3375
		4	0.250	3.4840	3.5237	3.5097	1.1395	0.7377	− 0.3973
		2	0.125	1.7995	1.7938	1.8102	−0.3168	0.5946	0.9143
8	8	8	1.000	5.2511	5.2521	4.9458	0.0190	− 5.8140	− 5.8320
		6	0.750	4.5077	4.5875	4.4139	1.7703	− 2.0809	− 3.7842
		4	0.500	3.2731	3.3684	3.3306	2.9116	1.7567	− 1.1222
		2	0.250	1.7074	1.7108	1.7506	0.1991	2.5302	2.3264
		1	0.125	0.8631	0.8660	0.9936	0.3360	15.1199	14.7344
8	4	4	0.500	2.9789	2.9935	2.8167	0.4901	− 5.4450	− 5.9061
		2	0.250	1.6920	1.7108	1.7180	1.1111	1.5366	0.4209
		1	0.125	0.8652	0.8660	0.9936	0.0925	14.8405	14.7344
4	4	4	1.000	2.7344	2.7396	2.7396	0.1902	− 4.1691	− 4.3510
		3	0.750	2.2590	2.3429	2.2786	3.7140	0.8676	− 2.7445
		2	0.500	1.5846	1.6571	1.6606	4.5753	4.7962	0.2112
		1	0.250	0.8090	0.8276	0.9269	2.2991	14.5735	11.9986
4	2	2	0.500	1.5400	1.5335	1.4976	−0.4221	− 2.7532	− 2.3410
		1	0.250	0.8195	0.8276	0.9269	0.9884	13.1056	11.9986

다. 이를 살펴보기 위하여 P=M=16, B=6, m=0.4인 경우에 각 기억장치 모듈에 대한 이용율을 구하여 표 2에 보였다. 표 2를 보면 MODEL1과 MODEL2의 각 기억장치 모듈의 이용율은 2% 이내의 편차로서 거의 일치하므로 기억장치 모듈 전체에 대하여 접근요구는 독립 균등분포를 이루고 있음을 확인하였다.

표 2. 기억장치 모듈의 이용율
(P=M=16, m=0.4)
Table 2. Utilization of memory module for P=M=16, m=0.4.

module	MODEL1	MODEL2
1	0.3448	0.3900
2	0.3418	0.3855
3	0.3467	0.3749
4	0.3426	0.3792
5	0.3404	0.3866
6	0.3418	0.3822
7	0.3448	0.3827
8	0.3360	0.3887
9	0.3379	0.3945
10	0.3550	0.3756
11	0.3447	0.3762
12	0.3443	0.3834
13	0.3458	0.3963
14	0.3462	0.3952
15	0.3394	0.3873
16	0.3512	0.3888

2. 성능평가

제안된 P×M×B 시스템의 성능을 평가하기 위하여, 하드웨어 성능을 결정하는 인수를 추출하고 시뮬레이터 MODEL2를 이용하여 시뮬레이션하였다. 인수들은 프로세서의 수 P, 기억장치 모듈의 수 M, 버스의 수 B, 국부기억장치 실패율 m이다. 또한 시뮬레이션 결과는 밴드폭 BW, 프로세서 이용율 U_P, 버스의 이용율 U_B, 및 버스 상호충돌 정도 C_B 등을 측정하였으며, 시뮬레이션 시간은 한개의 버스 사이클 타임을 단위 주기로 하여 10,000사이클로 하였다.

1) 국부 기억장치의 실패율 변화

시스템의 설계시에 프로세서와 기억장치 모듈의 수가 결정되면 국부 기억장치 실패율은 국부 기억장

치의 크기에 따라 변화하게 된다. 이러한 국부 기억장치 실패율의 변화에 따른 시스템의 성능을 고찰하기 위하여 P=M=16, B=6, 7, 8, 9인 경우를 예로들어 m을 0.1에서 0.9까지 변화시켰을 때 BW의 변화를 구하여 그림15에 보였다.

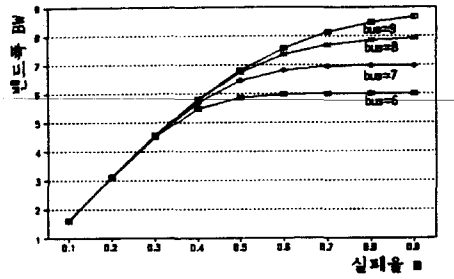

그림15. 실패율 대 밴드폭(P=M=16)
Fig. 15. Miss ratio vs. bandwidth for P=M=16.

그림15에서 국부 기억장치 실패율 m의 값을 0.1씩 증가할 때, BW는 선형적으로 증가하다가 m의 값이 일정한 값 이상이 되면 거의 변화가 없는 것을 알 수 있다. 예를 들어, P=M=16, B=6인 경우에 m의 변화에 따라 U_P, U_B, C_B를 구하면 그림16과 같다.

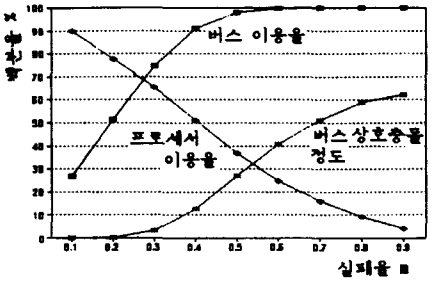

그림16. 이용율 및 버스상호충돌정도(P=M=16, B=6)
Fig. 16. Utilization and bus contention ratio for P=M=16, B=6.

그림16에서 m의 값이 0.1에서 0.4까지 변할때 U_B는 거의 선형적으로 증가하다가 m의 값이 0.4 이상이 되면 거의 변화가 없게 된다. 이러한 변화는 그림15에서 B=6일 때 m의 값이 0.4이상에서는 BW의 변화가 없는 것과 일치하고 있다. 이러한 이유는 버스 상호충돌이 m의 값이 0.1에서 0.4까지 변할때에는 거의 일어나지 않다가 m의 값이 0.4 이상이 되면 급격히 증가하게 되어 버스가 포화상태에 이르게 되기 때문이다. 이러한 버스 상호충돌 정도를 그림16에 함께 나타내었다. 또한 U_P는 m의 값이 증가할 때 거의 선형적으로 감소함을 알 수 있다. 이러한 이유는 프로세서의 기억장치 모듈 접근요구가 국부 기억장치 실패율에 의존하여 점차적으로 증가되기 때문이다.

(2) 버스 수의 변화

버스 수의 변화에 따른 시스템의 성능을 고찰하여 보자. P=M=16인 경우에 버스의 수 B를 1에서 16까지 변화시켰을 때 밴드폭 BW의 변화는 그림17과 같다.

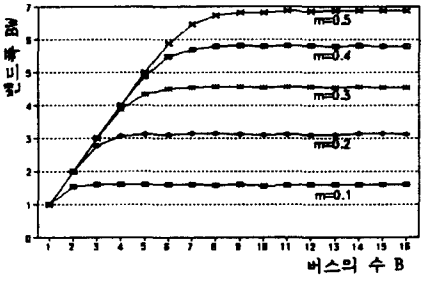

그림17. 버스 대 밴드폭(P=M=16)
Fig. 17. Bus vs. bandwidth for P=M=16.

그림17을 보면 버스의 수 B가 증가함에 따라 BW도 거의 선형적으로 증가하다가 B가 일정한 값 이상이 되면 밴드폭은 거의 변화가 없는 것을 알 수 있다. 예를 들면 P=M=16, m=0.4인 경우에 U_P, U_B, C_B를 구하면 그림18과 같다. 그림18에서 U_B의 변화를보면, B가 1에서 6까지 증가되는 동안에는 90%이상 유지되다가 B가 계속 증가되면 거의 선형적으로 감소하고 있다. 이는 그림17에서 m=0.4일 때 B가 6이상이 되면 BW의 변화가 거의 없는 것과 일치한다. 이러한 이유는 B가 적을 때에 C_B가 100%에 가깝다가 B=3에서 B=7까지는 선형적으로 감소하며, B=8 이상이 되면 0%에 가깝게 되기 때문임을 알 수 있다. U_P는 이러한 C_B에 영향을 받게되어 B가 6이 될 때까지는 거의 선형적으로 증가하나, B가 계속 증가되어도 변화가 거의 없게 된다.

지금까지 살펴본 결과에 따르면 P=M=16, m=0.4인 다중처리기 시스템에서 정보교환에 적절한 버스의 수는 6개임을 알 수 있다.

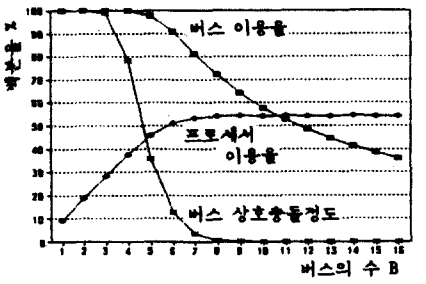

그림18. 이용율 및 버스상호충돌정도(P=M=16, m=0.4)

Fig. 18. Utilization and bus contention ratio for P=M=16, m=0.4.

본 연구는 시뮬레이션을 통하여 국부 기억장치 실패율과 버스 수의 변화에 따른 밴드폭, 프로세서의 이용율, 버스의 이용율 및 버스 상호충돌 정도를 구하여 다중처리기 시스템의 성능을 평가하였다. 이러한 시뮬레이션 결과에 따라 시스템 설계시 정보교환에 적절한 버스의 수를 결정할 수 있다. 프로세서와 기억장치 모듈의 수를 고정하고 성능평가를 하였으나 시스템 설계시 제한조건에 따라 다른 인수를 고정하고 필요한 결과인수를 최적화할 수 있다.

V. 결　　론

다중처리기 시스템은 여러개의 프로세서, 기억장치 모듈 및 주변장치를 공유하면서 고도의 병렬성을 갖고 대량의 데이타를 고속으로 처리함으로써 시스템의 성능을 향상시킬 수 있다.

본 연구에서는 각각의 프로세서가 국부 기억장치를 갖는 느슨한결합 다중버스 다중처리기 시스템의 구조를 제안하고 프로세서의 수, 기억장치 모듈의 수 버스의 수 및 국부 기억장치 실패율 등을 입력인수로 하여 밴드폭, 프로세서와 버스의 이용율 및 버스의 상호 충돌 정도 등 하드웨어 측면에서의 시스템 성능을 평가하였다. 특히 시스템의 해석적 모델과 시뮬레이션 모델을 구성하여 모델의 결과를 상호 검증하였다.

국부 기억장치 실패율의 변화에 따라 시스템 자원의 이용율은 크게 영향을 받으며, 특히 국부 기억장치 실패율이 증가함에 따라 프로세서의 이용율은 선형적으로 감소함을 알 수 있었다. 그러나 밴드폭은 국부 기억장치 실패율이 일정한 값까지 증가할 때에는 급격한 증가를 보이다가 그 이후에는 변화가 거의 없었다. 또한 버스의 수가 증가함에 따라 밴드폭과 프로세서의 이용율은 증가하지만 정보교환에 적절한 버스의 수 이상의 추가버스는 성능 향상에 도움이 되지 않는 것을 알았다.

예를 들어 프로세서와 기억장치 모듈의 수가 각각 16개이며, 국부 기억장치 실패율이 0.4인 경우에 최적 버스의 수가 6개임을 보였다. 또한 본 연구에서 구성한 시뮬레이터를 이용함으로서 다중처리기 시스템의 원시형태(prototype)을 설계할때에 성능평가 및 시스템에 적절한 버스의 수 결정에 필요한 기초 자료를 수집할 수 있게 된다. 따라서 시스템 설계에서 결정되는 입력인수에 따라 필요한 결과인수를 시뮬레이션을 통하여 결정할 수 있다.

앞으로의 연구로는 시스템의 동작이 비동기적으로 처리되는 다중처리 시스템의 연구를 통하여 실제 시스템에 더욱 접근되어져야 한다. 또한 물리적이나 기능적으로 동일하지 않은 프로세서로 구성된 다중처리 시스템에서 기억장치 모듈의 접근요구율이 서로 다른 경우와 자료의 정보 전달에 소요되는 시간이 다른 경우에 대하여 시스템의 성능을 평가하는 문제가 더욱 연구되어야 한다.

參 考 文 獻

[1] Don Towsley, "Approximate models of multiple bus multiprocessor systems," *IEEE Trans, Comput.*, vol. C-35, no. 3, pp. 220-228, Mar. 1986.

[2] B.L. Bodnar and A.C. Liu, "Modeling and performance analysis of single bus tightly coupled multiprocessors," *IEEE Trans, Comput.*, vol. 38, no. 3, pp. 464-470, Mar. 1989.

[3] W.A. Wulf and G.C. Bell, "C. mmp, a multi-miniprocessor," in *Proc. AFIPS 1972 Fall Joint Comput, Conf.*

[4] P. Markenscoff, "A deterministic model for evaluating the performance of a multiple processor system with a shared bus," *IEEE Trans. Comput.*, vol. C-33, no. 3, pp. 281-285, Mar. 1984.

[5] T.N. Mudge, et al., "Analysis of mutiple-bus interconnection networks," in *Proc. IEEE 1984 Int'l Conf, Parallel Processing*, pp. 228-232, Aug. 1984.

[6] M.A. Marson and M. Gerla, "Markov models for multiple bus multiprocessor systems," *IEEE Trans, Comput.*, vol. C-31, no. 3, pp. 239-248, Mar 1982.

[7] T.N. Mudge and D.H.B. Al-Sadoun, "A semi markov model for the performance of

multiple-bus systems," *IEEE Trans. Comput.*, vol. C-34, no. 10, pp. 934-942, Oct. 1985.

[8] D.D. Gajiki and J.K. Peir, "Essential issues in multiprocessor systems," *IEEE Trans, Comput.*, vol. 18, pp. 29-40, 1985.

[9] John Sanguinetti, "Performance of a message-based multiprocessor," *IEEE Comput.*, vol. 19, no. 9, pp. 47-55, Sep. 1986.

[10] Kai Hwang and Fate A. Briggs, *Computer Architecture and Paralllel Processing*, McGraw-Hill, 1984.

[11] T. Lang, et al., "Bandwidth of crossbar and multiple-bus connections for multiprocessors," *IEEE Trans, Comput.*, vol. C-31, pp. 1227-1233, Dec. 1982.

[12] H.S. Stone, *Introduction to Computer Architecture 2nd E*, SRA, pp. 551-555, 1980.

[13] Sheldon Ross, *A First Course in Probability*, Macmillian Publishing, 1976.

[14] M.H. MacDougall, *Simulating Computer Systems: Techniques and Tools*, MIT Press, 1987.

著 者 紹 介

朴 贊 政(正會員)

1947年 1月 14日生. 1984年 단국대학교 대학원 전자공학과 (공학석사). 단국대학교 대학원 전자공학과 박사과정 재학중. 1988年～현재 국립 강릉대학 전자계산학과 조교수. 주관심분야는 컴퓨터구조, 분산처리 시스템 등임.

●

李 相 範 (正會員) 第26卷 第11號 參照
현재 단국대학교 공과대학 전자공학과 교수

申 仁 澈(正會員)

1949年 11月 5日生. 1973年 고려대학교 전자공학과 졸업(공학사). 1986年 고려대학교 대학원 전자공학과 졸업(공학박사). 1984年～1985年 미시간 주립대학 객원교수. 1979年～현재 단국대학교 공과대학 전자공학과 교수. 주관심분야는 병렬처리 컴퓨터, VLSI 설계 등임.