# IBM SP2와 SGI Origin 2000에서의 병렬 VHDL 시뮬레이션*

## Parallel VHDL Simulation on IBM SP2 and SGI Origin 2000

정영식**, 정문정***
Young-Sik Jeong, Moon-Jung Chung

**Abstract**

In this paper, we present the results of simulation by running parallel VHDL simulation on typical MPP(Massively Parallel Processor) systems such as IBM SP2 and SGI Origin 2000. Parallel simulation uses the synchronous protocol and parallel program is implemented using MPI(Message Passing Interface) based on message passing model, so that it can run on any parallel programming environment which supports MPI, a standard communication library. And then GVT(Global Virtual Time) computation for parallel simulation is based on the global broadcasting with MPI_Bcast(), which ia a standard function in MPI and piggybacking. Our benchmark exhibits that as the size of VHDL grows, the parallel simulation has a better performance compared with the sequential simulation. In addition, we also show the results of comparison between IBM SP2 and SGI Origin 2000 by applying the same application to those indirectly.

* 이 연구는 한국과학재단의 해외 Post Doc.지원사업에 의하여 수행되었음.
** 원광대학교 컴퓨터공학과
*** 미시간 주립대학교 전산학과

## 1. 서론

VHDL(Very high speed integrated circuit Hardware Description Language)은 시스템 설계 시 하향식(top-down) 설계 방법론으로 사용되는 가장 일반적인 언어이다. VHDL 시뮬레이터는 많은 회사에서 제공되고 있고 많은 사용자들이 실제 목적에 가장 적합한 언어를 선택하여야 한다. 이중에서 가장 중요한 요소는 시뮬레이션 속도로서 이를 중심으로 시뮬레이터를 선택한다. 또한, 디지털 회로 시뮬레이션은 VLSI(Very Large Scale Integrated) 설계(design)와 검증(verification)에 아주 중요한 역할을 한다[3,12]. 즉, 디지털 회로 시뮬레이션은 회로 조립에 많은 시간과 비용이 소비되기 전에 회로의 타이밍(timing) 정보를 획득하고 회로의 기능(functionality) 검증을 위해 자주 사용된다[12]. 디지털 회로 시뮬레이션은 추상적 수준 (abstraction level), 구조적 게이트(architectural gate) 수준 및 구조적 스위치(switch) 수준 등으로 세분화되지만[12] 특히, 게이트 수준의 시뮬레이션은 시뮬레이션 방법과 알고리즘을 통해 효율적으로 구현이 가능하고 게이트 모델이 디지털 설계 표현으로 부울 방정식(boolean equation)에 직접적으로 사상이 되기 때문에 가장 널리 사용된다[2,12]. 본 연구에서도 게이트 수준을 고려하여 시뮬레이션을 수행한다.

디지털 회로 시뮬레이션을 일반적인 컴퓨터를 이용하여 순차적인(sequential) 시뮬레이션을 수행하면 많은 계산시간이 소요되는 단점이 있다. 특히, 시뮬레이션 하고자 하는 모델이 확률변수에 의해 표현되는 시스템일 경우는 그 정도가 더욱 심하다. 이러한 문제를 해결하기 위하여 병렬 시뮬레이션을 위한 연구가 많이 진행되어 왔다[2,12,13,14]. 최근에 대규모 병렬 프로세서(massively parallel processors)에서 병렬 시뮬레이션(parallel simulation)은 VLSI 설계와 같은 높은 성능의 컴퓨팅 문제(high performance computing problems)를 위해 가장 널리 사용, 연구되는 도구로써 보편화되고 있다[4,5,12]. 기존에 병렬 시뮬레이션을 위해 개발된 병렬 프로그램들은 이식성(portability)이 낮다는 단점이 있었다. 최근 이식성을 높이기 위해 MPI (Message Passing Interface) 라는 메시지 전송 프로토콜의 표준이 개발되었다[7].

기존의 디지털 회로에 대한 병렬 시뮬레이션의 접근 방법은 대칭형 다중 프로세서 시스템(symmetric multiprocessor system)에서 작은 수의 프로세서를 사용하였고 큰 회로의 실제적인 시뮬레이션 결과가 구체적으로 제시된 경우가 아주 드물었다[1]. 병렬 분산 시스템에서 비동기화 방식의 프로토콜(protocol)에 대한 논문[12]에서는 여러 가지 디지털 회로들에 대한 결과를 제시하였고 이때, 적용된 병렬 시스템은 다중 프로세서(multi-processor)상에서 수행되었고 비동기적 방식을 사용하였다. 또한, 국내에서는 DEVS(Discrete Event System Specification) [14] 형식론(formalism)을 C++환경에서 구현된 DEVS 추상화 시뮬레이터를 구축한 연구결과가 발표된 적이 있다[13]. 또한, Petri Net을 이용한 병렬 프로그래밍 환경을 구축하여 보다 효과적인 병렬 프로그램 개발을 위한 지원도구를 개발한 적이 있다[15].

본 연구에서는 기존의 연구들과는 달리 실제 큰 디지털 회로를 나타내는 VHDL에 대해 병렬 시뮬레이션을 IBM SP2와 SGI Origin 2000 시스템에서 수행하여 얻은 결과들을 제시하고자 한다. 이를 위해 먼저, VHDL로 표현된 디지털 회로에 대해 perl를 이용하여 일반 언어인 C++에서 이용할 수 있도록 원래의 VHDL과 구조적, 행위적으로 동일한 기능을 갖는 디지털 회로로 변환한다. 또한, 본 논문에서는 표준 MPI를 이용하여 병렬 프로그램화시키고 시뮬레이션 프로토콜은 동기화 방식을 사용하여 그 결과들을 제시한다. 이때, 병렬 시뮬레이션을 위해 국제 표준인 ISCAS '89 회로 중 최대의 게이트 수를 갖는 s38584 디지털 회로를 기본적으로 사용한다. 또한, 병렬 시뮬레이션의 병렬성 정도 검사를 위해 디지털 회로의 크기(size)에 따른 속도 향상의 정도를 살펴보기 위해 회로의 크기를 두 배, 네 배의 크기를 가진 회로로 증대시켜서 병렬 시뮬레이션을 수행시킨다. 본 연구의 주 목적은 실제 큰 디지털 회로 시뮬레이션을 병렬 컴퓨터에서 실행한 결과들을 제시하고 VHDL전용 언어를 사용하지 않고, 병렬 프로그래밍에 의한 시뮬레이션을 할 수 있는 기반을 마련하는 것이다. 또한, 비동기적 시뮬레이션 프로토콜 적용으로의 발판을 마련하며 GVT 계산과 시뮬레이

션 수행에 대해 보다 나은 성능을 가진 알고리즘을 개발할 수 있는 토대를 마련하는 것이다.

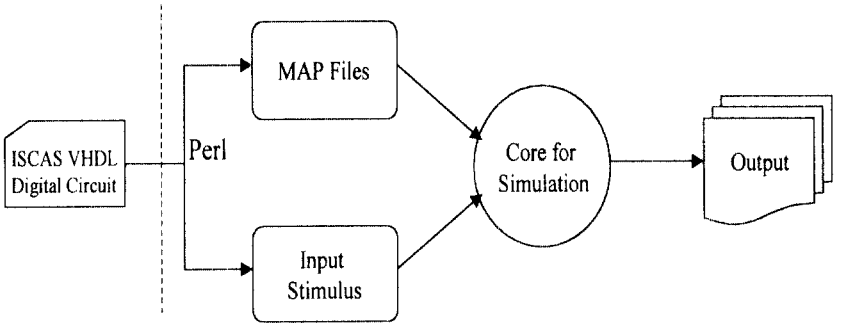
본 논문의 구성은 제 2장에서 병렬 시뮬레이션을 위한 전반적인 환경을 소개하고 제 3장에서는 병렬 시뮬레이션을 수행할 대규모 병렬 시스템에 대한 분류와 본 연구에서 실제 사용할 IBM SP2와 SGI Origin 2000시스템의 구성과 특성에 대해서 설명한다. 제 4장에서는 논리적인 프로세스 구성, 병렬 시뮬레이션을 위한 동기화 프로토콜과 GVT 계산 알고리즘 및 병렬 프로그래밍 모델을 제시한다. 마지막장에서는 병렬 시뮬레이션을 수행한 결과들을 두 가지 GVT 계산 방법에 대한 성능평가 결과, 디지털 회로에 대한 기본 척도들에 대한 성능평가 결과 및 확장성 분석 결과 등으로 나누어서 제시한다.

## 2. 병렬 시뮬레이션을 위한 전반적인 환경 및 시뮬레이터 구성도

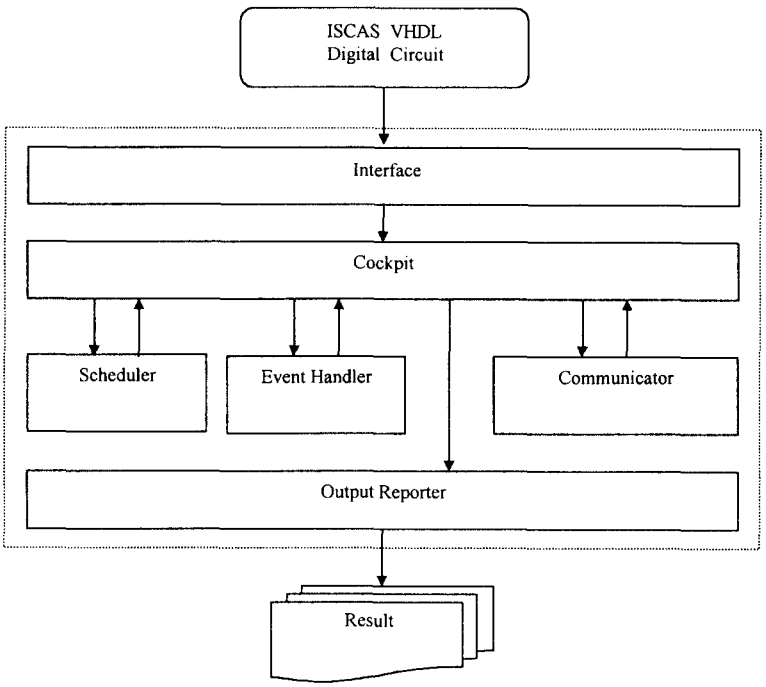
본 장에서는 디지털 회로를 병렬로 시뮬레이션하기 위한 전반적인 환경을 소개한다. 먼저, 본 연구에서 고려하는 디지털 회로는 ISCAS '89 s38584 로서 이는 총 회로의 개수가 20717개이며, AND, OR, NAND, XOR, invert, D 플립플롭(flip-flop) 등으로 구성되어 있다. 이 디지털 회로를 입력 데이터로 병렬 시뮬레이션이 수행되기 위한 전반적인 환경에 대한 것을 도식적으로 표현하면 다음 <그림 1>과 같다. <그림 1>에서 보는 바와 같이 ISCAS '89 s38584 회로를 시뮬레이션하기 위해서는 우선 각 게이트에 대한 정보와 게이트들간의 상호 연결 정보가 필요하다. 또한, 병렬 시뮬레이션을 위해서는 ISCAS 회로를 프로세서 갯수만큼 분할(partition)하여 각 프로세서는 자신에게 주어진 회로 부분에 대해서만 계산을 수행해야 한다. 즉, 프로세서의 개수를 n이라 하고 게이트의 수 m이라 하면 분할은 $m/n$ 수만큼의 게이트들이 각 프로세서에 할당된다. 이를 위해 perl 언어를 사용하여 디지털 회로로부터 필요한 정보를 추출하는 프로그램을 작성하였다.

ISCAS '89 s38584 회로를 입력으로 하여 이 perl 프로그램을 수행시키면 각 프로세서가 가진 게이트들에 대해 *<input numbers, output numbers, delay, interconnection information among data>*의 정보를 저장한 MAP 화일과 디지털 회로에는 38개로 구성된 입력단자에 입력되는 각 외부 입력에 대한 입력 stimulus 화일이 생성된다. 본 연구에서의 동기화 시뮬레이션 프로토콜을 이용한 병렬 시뮬레이션은 이 두 화일을 입력으로 하여 수행되며 수행 결과에 대한 통계를 출력한다.

VHDL의 병렬 시뮬레이션을 위한 시스템 구성도는 다음 <그림 2>와 같다. 병렬 시뮬레이터는 기본적으로 여섯 개의 모듈로 구성되어 있으며, 인터페이스는 VHDL을 입력으로 하여 MAP 화일과 랜덤 확률 분포함수를 이용하여 기본 입력 백터들의 값들을 생성한다. 메인 모듈을 나타내는 cockpit은 입력 데이터를 읽고 모듈들을 초기화시키고 시뮬레이션을 시작하게 하며, 다른 모듈을 종합적으로 관리한다. 스케쥴러는 병렬 시뮬레이션 프로토콜에 따라 사건들을 스케쥴링한다. 사건 스케쥴러는 큐내의




<그림 1> 병렬 VHDL 시뮬레이션을 위한 전반적인 환경




<그림 2> 병렬 시뮬레이터 구성도

사건들을 관리하고 큐(queue)내 사건들은 최소 힙(min-heap)구조로 유지된다. 통신자는 MPI 라이브러리를 이용하여 메시지를 다른 프로세서에게 송수신하는 기능을 수행한다. 결과 출력자는 병렬 시뮬레이션된 결과들을 출력하는 기능을 수행한다.

## 3. 대규모 병렬 컴퓨터 시스템 구조

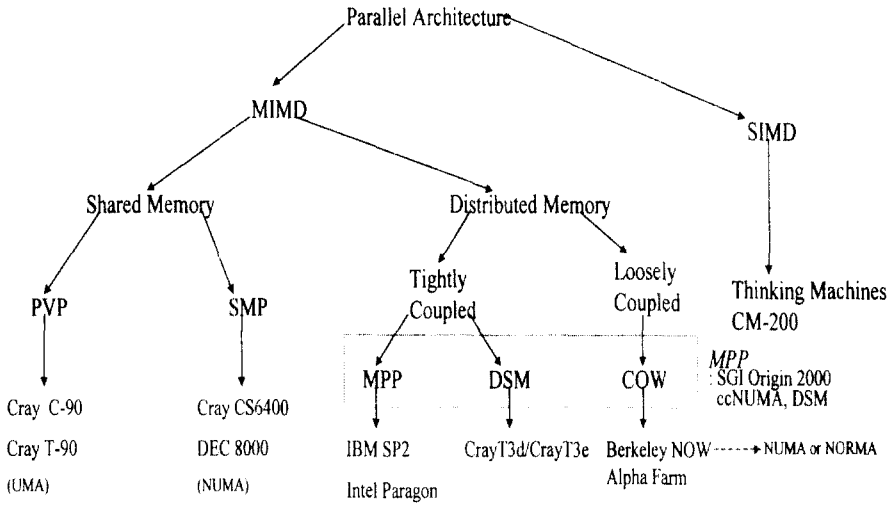
본 장에서는 대규모 병렬 컴퓨터 시스템의 구조들을 분류(taxonomy)하고 본 연구에서 이용하고자 하는 IBM SP2와 SGI Origin 2000 시스템들의 하드웨어 구성에 대해 살펴보기로 한다. 먼저, 대규모 병렬 컴퓨터 시스템 구조에 대한 분류는 관점에 따라 조금씩 상이하게 분류되는데 일반적으로 SIMD(Single Instruction Multiple Data)와 MIMD(Multiple Instruction Multiple Data)로 크게 나누어지고 그 아래 개념적인 형태들로 세분화된다.

본 연구에서는 <그림 3>과 같이 분류하고[8,9] 본 연구의 병렬 시뮬레이션을 수행할 대규모 병렬 컴퓨터 시스템들인 IBM SP2와 SGI Origin 2000이 어느 곳에 속하는지를 알 수 있다. IBM SP2는 하와이대에 있는 MHPCC(Maui High Performance Computing Center)의 시스템을 이용하고 SGI Origin 2000은 Wright-Patterson 공군기지 내의 ASC (Aeronautical Systems Center)의 시스템을 이용하였다. 병렬 시뮬레이션에 사용된 IBM SP2와 SGI Origin 2000의 하드웨어 구성과 특성은 다음 표 1과 같다[6,8,9,10,11].

표 1에서 제시된 IBM SP2의 경우는 실제 MPI를 사용하여 메시지(message) 전송을 했을 때의 값이고, SGI Origin 2000의 경우는 원격 메모리(remote memory)를 접근(access)했을 때의 값이다.

① IBM SP2

MHPCC에 설치되어 있는 IBM SP2 시스템은 480 노드(node)를 가지고 있다. 각 노드는 66.7 MHz POWER2 프로세서(processor) 하나와 64~256 MB의 로컬 메모리(local memory)를 가지고 있고 각 프로세서는 266 MFLOPS/s의 성능을 가진다. 이러한



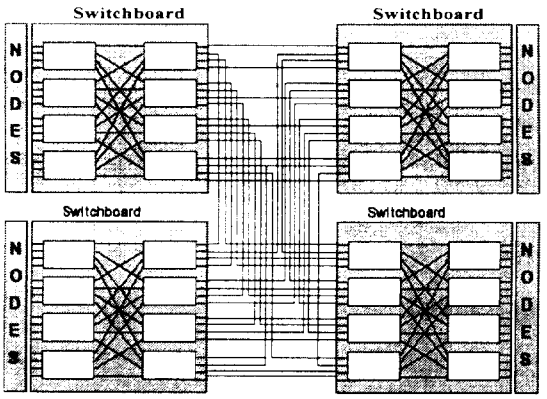

<그림 3> 대규모 병렬 컴퓨터 시스템 구조의 분류

<표 1> IBM SP2와 SGI Origin 2000의 구성과 특성

| | IBM SP2 | SGI Origin 2000 |
|---|---|---|
| · 설치 장소 | Maui High Performance Computing Center | Aeronautical Systerms Center |
| · 시스템구성 | 480 노드/100 GFLOPS | 112 노드 |
| · 노드 구성 | 1 프로세서 | 2 프로세서 |
| | 64MB~256MB 로컬메모리 | 1GB 로컬메모리 |
| | 1~4.5 GB 로컬디스크 | 6GB 로컬디스크 |
| · CPU | 66.7 MHz POWER2 | 195MHz R10000 |
| · 연결망구조 | 다단계 상호연결망(MIN) | 계층 팻 하이퍼큐브(hierarchical fat hypercube) |
| · 메모리 모델 | NUMA/NORMA | ccNUMA |
| · 노드간 지연시간 깊 대역폭 | 40us/35MB/s | 945 ns (128 프로세서에서)<br>780MB/s (메모리 대역폭) |
| · 프로그래밍모델 | 메시지 전송 | 공유메모리 + 메시지 전송 |
| · MPI 지원 | 지원 | 지원 |

로컬 메모리는 공유되지 않고 메시지 전송에 의해 필요한 데이터를 상호교환한다. 이를 위해 480개의 노드들은 HPS(High Performance Switch) 로 구성된 다단계 상호연결망 (Multistage Interconnection Network : MIN)에 의해 서로 연결된다.

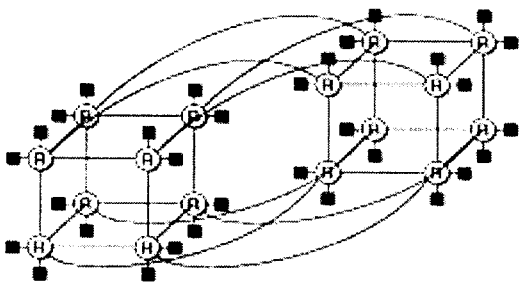
HPS는 버퍼링(buffering) 기능이 추가된 웜홀 라우팅(wormhole routing)을 하며 각 메시지는 256 바이트의 패킷(packet)으로 나뉘어져 전송된다. 네트워크(network)에 경쟁이 없는 경우 하나의 HPS 칩(chip)을 지나는 하드웨어 지연시간은 125 ns이며 512 노드의 경우 875 ns의 지연시간을 가진다[8]. 하지만, 실제 MPI를 사용하여 메시지 전송을 할 경우 시스템 소프트웨어 오버헤드 때문에 40 us라는 높은 노드간 지연시간을 가진다. <그림 4>의 (a)는 64 개의 노드를 가진 IBM SP2 하드웨어 구성도이다.




(a) IBM SP2

② SGI Origin 2000

ASC에 설치되어 있는 Origin 2000은 112 노드 (224 프로세서)를 가지고 있다. Origin 2000은 IBM SP2와는 달리 각 노드의 메모리는 하드웨어 캐쉬(cache) 일관성 프로토콜에 의해 공유되어 사용되며 추가적으로 메시지 전송 모델도 지원한다. 각 노드는 하나 혹은 두개의 R10000 프로세서와 1GB 의 로컬 메모리를 가진다. 각 노드는 32 노드 (64프로세서)까지는 하이퍼큐브에 의해, 그 이상은 계층 팻 하이퍼큐브(hierarchical fat hypercube)에 의해 연결되며 최대 512 노드 (1024 프로세서) 까지 구성할 수 있다. 계층 팻 하이퍼큐브는 SPIDER 라는 라우터(router)에 의해 구성된다[11]. SPIDER 라우터는 낮은 지연시간을 위해 웜홀 라우팅을 사용하며 각 물리적 채널(physical channel) 이 4개의 가상채널(virtual channel) 에 의해 공유되도록 하여 실제 채널의 대역폭이 효율적으로 사용될 수 있도록 하였다. 네트워크에 경쟁이 없는 경우 SPIDER 라우터 칩을 지나는 하드웨어 지연시간은 40 ns이며 512 노드의 경우 371 ns의 평균 지연시간을 갖는다. 128 프로세서를 가진 시스템에서 실제 원격(remote) 메모리를 접근할 경우의 평균 지연시간도 945 ns로 상당히 낮은 지연시간을 보여준다[10]. <그림 4>의 (b)는 SGI Origin 2000 시스템의 하드웨어 구성도이다.




R : Router

(b) SGI Origin 2000

<그림 4> 64개의 노드로 구성된 IBM SP2와 SGI Oriin 2000

## 4. 논리적 프로세스 구성 및 GVT 계산

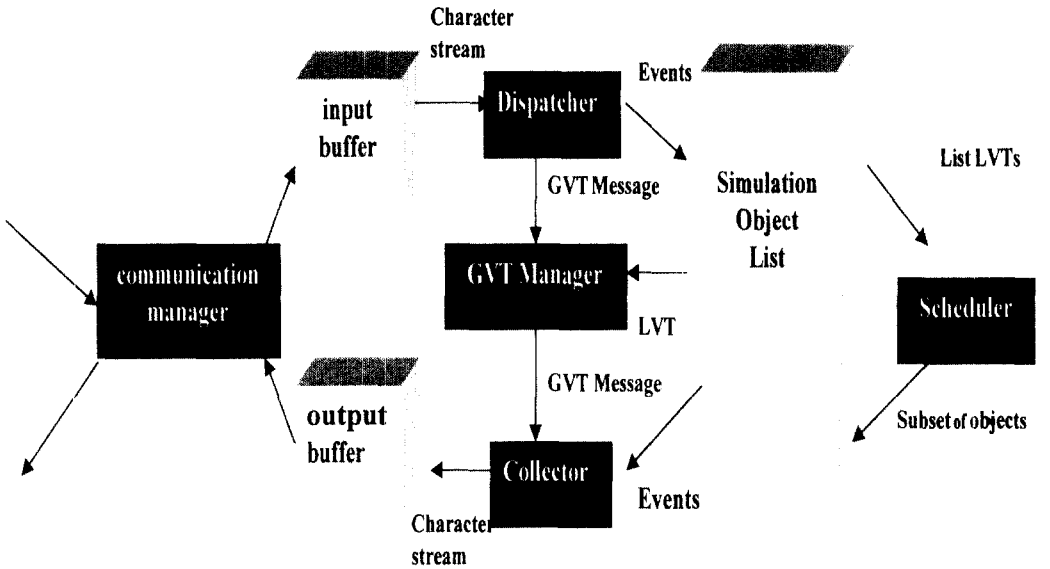
### 4.1 논리적 프로세스 구성

본 연구의 병렬 시뮬레이션을 위해 각 프로세서는 하나의 프로세스를 생성하고 그것에 의해 시뮬레이션이 진행된다. 또한, 본 연구에서 사용하는 병렬 시뮬레이션 프로토콜은 병렬 시뮬레이션에서 가장 보편적인 스킴(scheme)으로 알려져 있는 동기화(synchronous) 프로토콜을 사용하며, 이는 매 사건 처리마다 GVT 계산을 수행한다. 각 프로세스는

MAP 파일로 부터 자신에게 할당된 게이트들에 대한 정보를 읽어 들이고 입력 stimulus 화일에 있는 외부 입력을 사용하여 시뮬레이션을 수행한다. 논리적 프로세스의 구성은 다음 <그림 5>와 같다.

### 4.2 GVT 계산

병렬 시뮬레이션은 통신을 위해 서로 주고 받는 여러 프로세스(process)들의 집합으로 간주되는데 프로세스간 주고 받는 사건에는 시간표(timestamp)가 붙어 있으며 그 시간표에는 그 사건이 처리되어




<그림 5> 논리적 프로세스의 구성

<표 2> MPI_Bcast( )와 Piggybacking에 의한 GVT 계산

| MPI_Bcast() 함수 이용 | Piggybacking |
|---|---|
| · compute GVT using MPI_REDUCE function ;<br>· get the minimal GVT from all processors using MPI_REDUCE function ;<br>· broadcasting minimal GVT to all processors using MPI_Bcast function ;<br>· execute events with minimal GVT at the procesors ;<br>· send events to processors using *isend* function ;<br>· probe and receive events ;<br>· insert into event queue ; | · execute events each processor ;<br>· send events to processors using *isend* function ;<br>· probe and receive events ;<br>· compute GVT using piggybacked LVT ;<br>· insert into event queue ;<br>· execute waitall function for isend |

야 하는 시간이 기록된다. 일반적으로 각 프로세스에는 여러 개의 수행해야할 사건들이 큐(queue)에 대기하게 된다. 이 경우 시뮬레이션이 올바르게 진행되기 위해서는 각 프로세스에 도착하는 모든 사건들이 그 사건의 시간표의 시간 순으로 배열되어 그 순서에 맞추어 시뮬레이션하여야 한다. 각 프로세스들은 자신에 도착하는 사건들이 가진 시간표에 따라 처리되는데, 이때 현재 처리되는 사건은 자신의 프로세스 내에서 대기중인 사건들이 가진 시간표를 검사하여 가장 작은 시간표를 가진 사건을 의미한다. 또한, 각 프로세스의 사건들에 대한 시간표는 LVT(Local Virtual Time)에 의해 유지된다. 모든 LVT의 시간표들 중에서 최소 값으로 정의되는 GVT(Global Virtual Time)는 일반적으로 N를 논리적 프로세스들의 집합, $LVT_i$를 논리적 프로세스 $i(i \in N)$의 LVT로 가정하면,

$GVT = \min_{i \in N} \{LVT_i\}$ 이다. 최소의 GVT 값을 갖는 사건들이 각 프로세스에서 수행되고 다음 단계로 진행된다. 동기화 병렬 시뮬레이션에서 가장 중요한 부분중의 하나가 GVT를 계산하는 것이다. GVT 계산을 위한 가장 간단한 방법은 MPI 라이브러리내 MPI_Bcast() 함수 즉, 글로벌 브로드캐스팅 함수(global broadcasting function)를 이용하여 구현하는 것이다. 각 프로세스는 글로벌 브로드캐스팅을 하는 시점에서 동기화가 이루어진다. 이 메카니즘은 구현이 간단하다는 장점이 있지만, 모든 프로세스들 중에서 가장 작은 LVT 를 찾는 과정이 필요하며 글로벌 브로드캐스팅을 위한 오버헤드가 크다는 단점이 있다. 두 번째 GVT 계산을 위한 메카니즘은 각 프로세스가 동기화를 위한 함수를 수행하지 않고, 메시지를 보낼 때 자신의 LVT를 메시지에 넣어 함께 보내는 피기백킹(piggybacking) 방법이 있다. 각 프로세스는 다른 모든 프로세스들로부터 메시지를 받은 후 GVT를 결정할 수 있는데, 만약 LP를 프로세스들의 집합, e를 사건, t(e)를 사건 e에 대한 시간표, $m_{i,j}(i, j \in LP)$를 프로세스 $i$에서 프로세스 $j$로 전송되는 메시지라 가정하면,

$$GVT = \min\left\{ \min_{i \in N}\{ LVT_i \},\ \min_{e \in m_{i,j}} \{t(e)\} \right\}$$

로 결정된다. 즉, GVT는 현재 자신의 LVT와 다른 프로세스들로부터 도착한 메시지가 가지고 있는 LVT들 중 최소값으로 결정할 수 있다. 표 2는 이들 방법의 단계를 나타내고 있다.

### 4.3 병렬 프로그래밍

병렬 프로그래밍을 위한 모델은 묵시적 모델(implicit model), 데이터 병렬 모델(data parallel model),공유변수 모델(shared variable model), 메시지 패싱 모델(message passing model : MPM)로 분류[8,9]되는데, 일반적으로 디지털 신호처리(digital signal processing)에서는 효율성(efficiency)과 이식성(portability)에 있어서 장점을 갖는 메시지 패싱 모델을 이용한다. MPM을 구현하는 방법도 MPI (message passing interface)에 의한 방법과 PVM (parall -el virtual machine)에 의한 방법이 있다.

PVM은 이질적 네트워크(heterogeneous net -work) 에서 수행하는 방법이어서 본 연구에서는 이보다도 효율성, 병렬성 및 이식성에 있어서 좋은 평가를 받고있는[8,9] MPI 라이브러리[7]를 이용하여 병렬 프로그래밍화 한다. MPI 초기화와 본 연구에서 병렬 프로그래밍을 위해 사용한 MPI 주요 함수들은 다음과 같다.

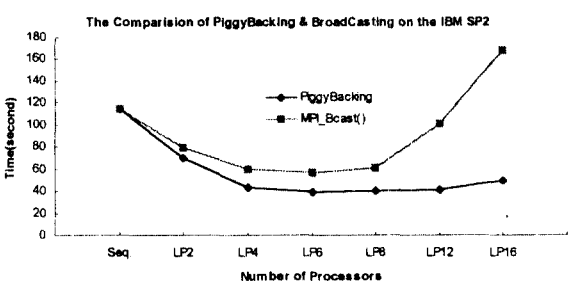
```
MPI_Init(&argc, &argv);   //initialize MPI
MPI_Comm_rank(MPI_COMM_WORLD, &taskid);
    //find out which process I am
MPI_Comm_size(MPI_COMM_WORLD,&numtasks);
    //find out how many processes there are
int MPI_Isend(void* buf, int count, MPI_Datatype
    datatype, int dest, int tag, MPI_Comm comm,
    MPI_Request *request);
int MPI_Probe(int source, int tag, MPI_Comm
    comm, MPI_Status status);
int MPI_Recv(void* buf, int count, MPI_Datatype
    datatype, int source, int tag, MPI_Comm
    comm, MPI_Status *status);
int MPI_Waitall(int count, MPI_Request *array
    _of_requests, MPI_Status *array_of _statuses);
```

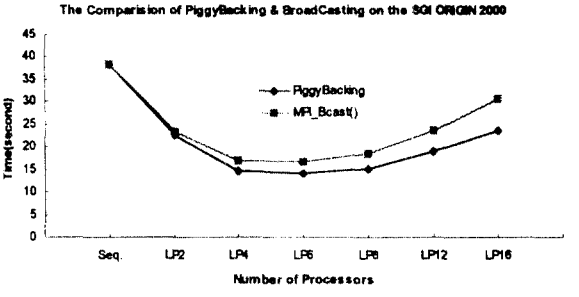
## 5. 병렬 시뮬레이션 결과

### 5.1 GVT 계산 방법의 성능비교

본 연구에서 이미 언급을 했듯이 ISCAS '89 회로 중 최대의 게이트 수(20679)를 갖는 s38584를 이용하였으며, 기본 입력 벡터 수가 38개이며, 각 벡터의 입력 데이터를 균일 확률 함수에 의해 1,000개 생성하여 시뮬레이션 하였다. 이 디지털 회로에 대해 순차적 시뮬레이션 전체 수행시간($T_{seq}$)은 피기백킹인 경우 IBM SP2가 114.0331초, SGI Origin 2000이 38.2468초의 결과를 얻었고 전체 시뮬레이션 사이클(cycle)은 32831, 처리된 전체 사건 수(6134579) 등의 기본 척도(basic measures) 결과들을 얻었다.

제 4장에서 제안된 GVT 계산을 위한 두 메카니즘을 적용한 벤치마크 결과를 IBM SP2와 SGI Origin 2000으로 나누어서 제시하면 다음 <그림 6>과 같다.




<그림 6> SP2/SGI Origin 2000에서의 피기백킹 과 MPI_Bcast() 함수이용의 성능비교

<그림 6>에서 보는 바와 같이 전체적으로 MPI_Bcast() 함수를 이용하는 것 보다 피기백킹 방법이 IBM SP2에서 약 89.1113%(프로세스 12개, 16개 기준), SGI Origin 2000에서 약 18.7924% 정도 좋은 성능을 보였다. 특히, IBM SP2에서 MPI_Bcast() 함수를 이용하는 경우 프로세스가 증가함에 따라 통신시간(communication time)이 계산시간(computation time)보다 훨씬 많이 소요되어 결과적으로 전체적인 수행시간(total execution time)이 증가함을 의미한다. 그러므로 다음 절 부터는 피기백킹 GVT 계산방법을 바탕으로 여러 가지를 시뮬레이션하고 그 결과들을 제시한다.

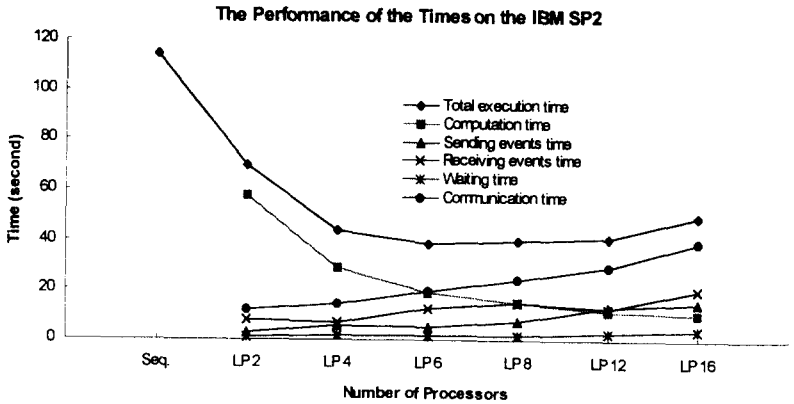
### 5.2 병렬 시뮬레이션의 기본 성능평가 결과

기본 성능평가를 위한 척도(measures)들은 총 시뮬레이션 수행시간, 메시지 통신시간, 계산수행시간 및 메시지 송수신 시간을 중심으로 결과들을 제시한다. 이들 요소들에 대한 해석적 모델은 T를 실행시간(execution time), N은 문제 크기, P를 프로세서의 개수일 때, $T = T^{j}_{comp} + T^{j}_{comm} + T^{j}_{idle}$ $(j \in P)$ 혹은

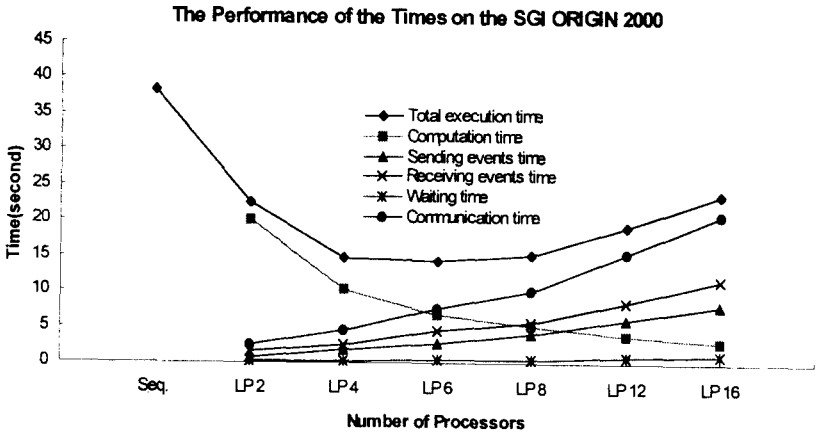
$$T = \frac{1}{p}(T_{comp} + T_{comm} + T_{idle})$$

$$= \frac{1}{p}\left(\sum_{i=0}^{p-1} T_{comp} + \sum_{i=0}^{p-1} T_{comm} + \sum_{i=0}^{p-1} T_{idle}\right)$$

이고 여기서, $T_{comp}$ 는 작업을 수행하는데 걸리는 계산시간, $T_{comm}$은 각 프로세스들간의 통신시간, $T_{idle}$은 자원의 휴지시간으로 정의할 수 있고 실제 메시지 전송에 따른 오버헤드는 $T_{msg} = t_s + t_w L$이 된다( $t_s$ : 메시지 전송 시작을 위해 소요되는 비용, $t_w$ 전송하고자 하는 워드당 소요되는 비용, L은 메시지 크기). <그림 7>과 <그림 8>에서 보는 바와 같이 ISCAS s38584를 순차적 시뮬레이션하여 얻은 결과는

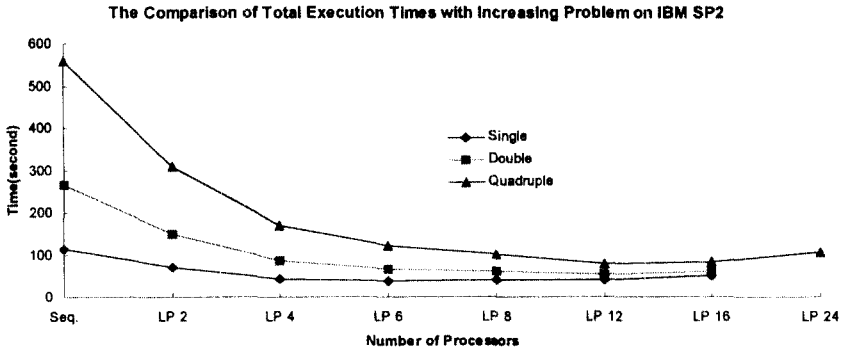



<그림 7> IBM SP2에서 병렬 시뮬레이션 기본결과

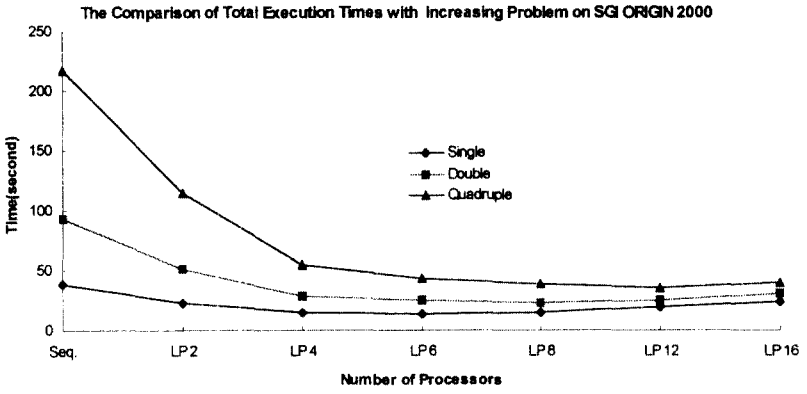



<그림 8> SGI Origin 2000에서 병렬 시뮬레이션 기본결과

IBM SP2에서는 114.0331초, SGI Origin 2000에서는 38.2468초의 시간이 소요되었다. 점차적으로 프로세스 수를 2, 4, 6, 8, 12, 16까지 증가시켜 병렬로 수행한 결과들이 <그림 7>과 <그림 8>에 제시되었다. 프로세스를 6개 사용하는 경우가 가장 좋은 성능을 보이고 있고 프로세스 수가 8개 이상되는 경우는 통신시간이 계산시간 보다 많이 소요되어 결과적으로 전체 수행시간이 증가하게 된다. 즉, 전체 수행시간은 통신시간에 계산시간을 합한 것이 되고 통신시간은 모든 프로세스들이 사건을 보내는 시간(sending events time)과 받는 시간(receiving events time)을 합한 것이 된다. 본 연구에서 사건을 보낼 때, MPI_Isen d() 함수를 이용하였는데, 이 경우 사건을 다른 프로세스들에게 보내기 위해 "send" 명령이 시작하면, 입력 큐에 있는 모든 사건들이 시스템 버퍼에 완전히 복사되기를 기다리게 하는 MPI_Waitall() 함수가 요구됨에 따라 이에 소요되는 대기시간(waiting time)이 소요된다. 그러므로 통신시간에는 이 대기시간을 포함하게 된다.




<그림 9> IBM SP2에서 확장성 분석결과

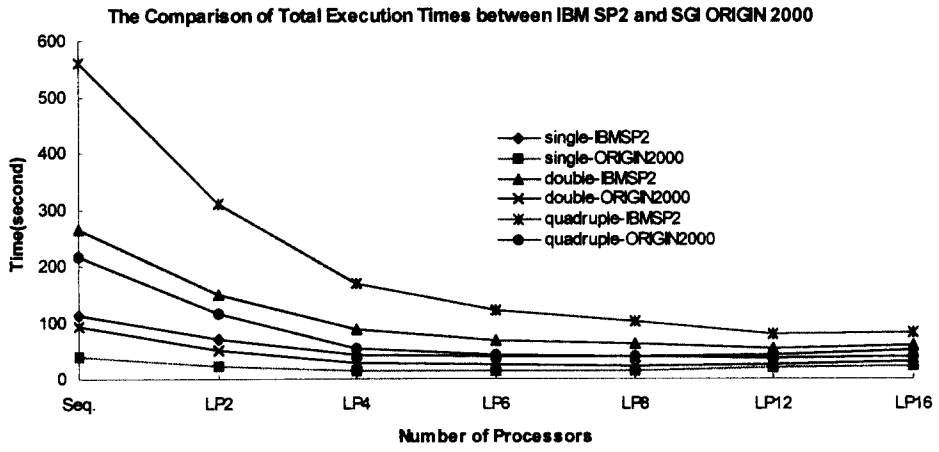



<그림 10> SGI Origin 2000에서 확장성 분석결과

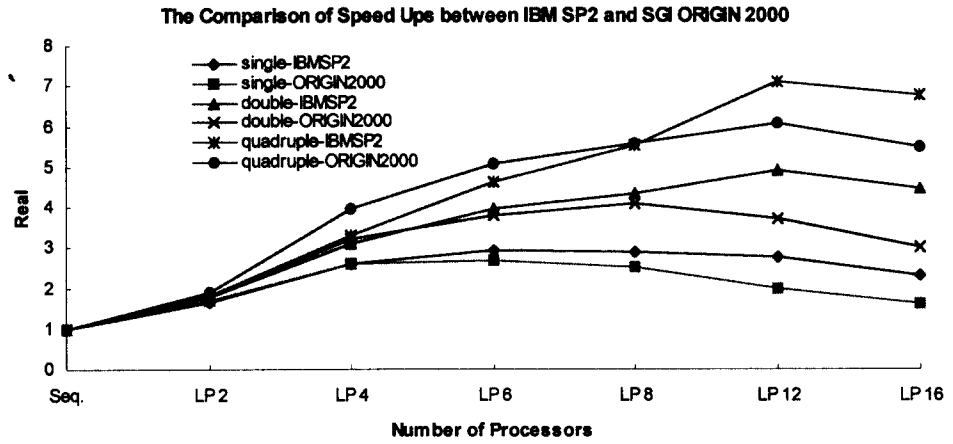
### 5.3 확장성 분석결과

순차적 시뮬레이션의 결과와 프로세스의 수를 계속적으로 증가시킴으로써 시뮬레이션의 병렬성에 대한 결과들을 제시한다. <그림 9>와 <그림 10>은 디지털 회로의 전체 크기를 두 배, 네 배로 확장하여 병렬 시뮬레이션 한 결과들이다. 즉, ISCAS s38584의 크기를 두 배, 네 배로 증가(게이트 수를 41,358개, 82,716개로 증가시키고, 기본 입력 벡터의 수를 74개, 148로 증가)시켜 수행시킨 것으로 전체 수행시간에 대해 IBM SP2와 SGI Origin 2000에서 얻은 결과들을 제시하였다.

<그림 9>에서 보는 바와 같이 IBM SP2는 디지털 회로를 두 배로 증가시키는 경우, 프로세스 8개를 사용하는 것이 가장 좋은 성능을 보이고 있음을 알 수 있었고 네 배로 증가시키는 경우는 프로세스 12개를 사용하는 것이 가장 좋은 성능을 보이고 있다. <그림 10>에서 보는바와 같이 SGI Origin 2000에서도 동일한 결과를 보이고 있음을 알 수 있다. 즉, 병렬 시뮬레이션을 적용하고자 하는 응용분야의 문제 크기가 크면 클수록 병렬성(parallelism)은 증가함을 알 수 있다.




<그림 11> 전체 수행시간 척도를 통한 IBM SP2와 SGI Origin 2000의 간접적인 성능비교




<그림 12> 상대적 속도 척도를 통한 IBM SP2와 SGI Origin 2000의 간접적인 성능비교

### 5.4 IBM SP2와 SGI Origin 2000의 성능비교 결과

본 연구에서는 동일한 문제를 IBM SP2와 SGI Origin 2000에서 병렬 시뮬레이션을 수행함을 이미 언급하였다. ISCAS 디지털 회로의 원래 크기와 두 배, 네 배 크기를 병렬 시뮬레이션 한 전체 수행시간에 대해 IBM SP2와 SGI Origin 2000에서의 결과들을 비교하면, 전체적으로 SGI Origin 2000이 IBM SP2보다 약 3 배정도 이상의 속도가 빠름을 알 수 있었다. 또한, IBM SP2에서 네 배의 크기를 가지고 병렬 시뮬레이션 하는 경우 순차적 시뮬레이션보다

약 8배 정도의 속도 향상이 되었음을 알았다. 물론 SGI Origin 2000에서도 약 7배 정도의 속도향상이 있었다. 그러므로 대규모의 응용문제에 대해 병렬 시뮬레이션 하는 경우 상당한 속도 향상이 예상됨을 알 수 있다. 두 대규모 병렬 컴퓨터 시스템사이의 속도향상 차이가 있는 것은 각 시스템이 가진 고유특성에 따른 것이다. 이에 대한 자세한 결과는 <그림 11>에 나타나 있으며, 두 시스템간 성능비교에 간접적인 척도로 이용할 수 있다. 여기서는, 4.2절의 기본 척도를 위한 해석적 모델과 효율성(efficiency)

$$E_r = \frac{T_1}{PT_p} (T_1: \text{execution time on one processor}, T_p: \text{the time one p processors})$$

척도를 이용하여 상대적(relative) 속도(speedup) $S_r = PE$ r (여기서 r은 relative임) 에 대한 성능평가 결과를 제시한다. 속도에 대해서 IBM SP2와 SGI Origin 2000에서 수행된 결과들이 <그림 12>에 잘 나타나 있다.

## 6. 결 론

본 연구에서는 기존의 연구들과는 달리 실제 큰 디지털 회로에 대해 동기화 시뮬레이션 프로토콜을 이용하여 병렬 시뮬레이션을 대규모 병렬 컴퓨터 시스템에서 수행하여 얻은 결과들을 제시하였다. 또한, 병렬 프로그래밍을 위해 표준 MPI를 이용함으로써 어떠한 병렬 프로그래밍 환경에서도 본 연구에서 개발된 것이 수행할 수 있도록 하기 위한 환경을 마련하였다. 병렬 시뮬레이션 프로그램은 현재 대표적인 대규모 병렬 컴퓨터 시스템인 IBM SP2와 SGI Origin 2000에서 수행하였다. 병렬 시뮬레이션을 위해 국제표준인 ISCAS '89 회로 중 최대의 게이트 수를 갖는 s38584를 기본적으로 사용하였고 또한, 문제 크기에 따른 속도 향상의 정도를 살펴보기 위해 s38584의 두 배, 네 배의 크기를 가진 회로들을 병렬 시뮬레이션을 수행시켰다. 약 8만개 이상의 게이트를 가진 디지털 회로에 대해 병렬 시뮬레이션을 수행한 경우, 순차적 시뮬레이션에 비해 약 8배 가까운 성능 향상을 얻을 수 있었다. 그러므로 대규모의 응용문제에 대해 병렬 시뮬레이션하는 경우 상당한 속도 향상이 예상됨을 알 수 있었다. 또한, 본 연구를 통해 IBM SP2와 SGI Origin 2000 시스템을 간접적으로 비교할 수 있는 부수적인 결과도 얻었다. 본 연구의 주 목적인 실제 큰 디지털 회로 시뮬레이션을 병렬 컴퓨터에서 실행한 결과들을 제시하였고 VHDL전용언어를 사용하지 않고, 병렬 프로그래밍에 의한 시뮬레이션을 할 수 있는 기반을 마련하였다.

본 연구에서는 동기화 시뮬레이션 프로토콜 방식에 의해 GVT를 계산하여 병렬 시뮬레이션이 진행되는 것을 연구하였다. 추후 연구과제로는 GVT 계산과 시뮬레이션 수행 에 대한 보다 나은 알고리즘 개발을 위해 비동기화 방식의 적용 즉, Time Warp 알고리즘에 의한 병렬 시뮬레이션을 적용하여 동기화 방식과의 성능을 비교하고 이때, 적용하는 GVT 계산을 위해 기존에 개발된 모든 것들을 조사 분석하고 새롭고 효율적인 GVT 계산 알고리즘을 연구할 예정이다. 또한, 병렬 시뮬레이션 스킴(scheme)을 객체지향(object-oriented) 모델을 접목시켜 보다 나은 성능을 얻도록 하는 것이다.

## 참고 문헌

[1] V.D. Agrawal and S.T. Chakradhar, "Performance analysis of synchronized iterative algorithms on multiprocessors systems," IEEE Trans. on Parallel and Distributed Systems, vol. 3, pp.739-746, Nov. 1992.

[2] M.L. Bailey, "How circuit size affects parallelism," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Vol. 11, pp.205-215, Feb. 1992.

[3] K. Chandy and J. Misra, "Distributed simulation : A Case study in design and verification of distributed programs, IEEE Trans. on S/W, Vol. 5, pp.440-452, Sep. 1979.

[4] J. Misra, "Distributed discrete-event simulation," Computing Surveys, vol. 18, pp.39-65, Mar. 1986.

[5] R.M. Fujimoto, "Parallel discrete event simulation," Communications of ACM, Vol. 33, pp.30-53, Oct. 1990.

[6] MHPCC, "Maui High Performance Computing Center(MHPCC)," http://www.mhpcc.edu/mhpcc.html, 1997.

[7] University of Tennessee, *MPI : A Message-Passing Interface Standard*, Jun. 1995.

[8] K. Hwang, Z. Xu, and M. Arakawa, "Benchmark Evaluation of the IBM SP2 for Parallel Signal Processing," IEEE Trans. on Parallel and Distributed Systems, vol. 7, No. 5, pp.522-536, May 1996.

[9] K. Hwang, Z. Xu, "Scalable Parallel Computers for Real-Time Signal Processing," IEEE signal processing magazine pp.50-66, July 1996.

[10] J. Laudon and D. Lenoski, "The SGI Origin: A ccNUMA Highly Scalable Server," In Proceedings of the 24th Annual International Symposium on Computer Architecture, 1997.

[11] M. Galles, "Scalable Pipelined Interconnect for Distributed Endpoint Routing: The SGI SPIDER Chip," In Hot Interconnects 1996.

[12] E. Choi, "Parallel Asynchronous Protocols on Parallel and Distributed Systems," Ph. D. Dissertation in Michigan State University, 1997.

[13] 성영락, 김탁곤, 박규호, "병렬 DEVS 시뮬레이션 환경(P-DEVSIM++)의 성능평가," 한국 시뮬레이션 학회 논문지 제 2권 제 1호, pp.31-44, 1993년 12월.

[14] 성영락, 정성훈, 김탁곤, 박규호, "병렬 분산환경에서의 DEVS 형식론의 구현," 한국 시뮬레이션 학회 논문지 제 1권 제 1호, pp.64-75, 1992년 12월.

[15] 박종범, 박윤보, 박찬익, "Petri Net을 이용한 병렬 프로그래밍 개발 환경의 설계 및 구현," 한국정보과학회 논문지 Vol. 18, No. 6, pp.619-628, 1991년 12월.

## 저자소개

**정영식**

1987년 고려대학교 수학과 학사
1989년 고려대학교 전산학 석사
1993년 고려대학교 전산학 박사
1997~98 미시간 주립대학교 전산학과 Post Doc.
1993~현재 원광대학교 컴퓨터 및 정보통신공학부 조교수
관심분야 : 모델링과 시뮬레이션, 병렬분산 프로그래밍, 멀티미디어 CAI 등

**정문정**

1973년 서울대학교 응용수학과 학사
1975년 한국과학기술원 전산학 석사
1981년 Northwest University 전산학 박사
1981~87 RensselaerPolytechnic Institute 전산학과 조교수
1987~현재 미시간 주립대학교 전산학과 부교수
관심분야 : Process Management, Engineering Infomation System, Design Automation, Parallel Algorithm, Parallel Simulation