論文99-36C-10-3

ASIC 설계의 효과적인 검증을 위한 에뮬레이션 시스템

(An Emulation System for Efficient Verification of ASIC Design)

柳 光 基 *, 鄭 正 和 *

(Kwang-Ki Ryoo and Jong-Wha Chong)

요 약

본 논문에서는 ASIC 설계 회로를 빠른 시간 내에 구현 및 검증할 수 있는 에뮬레이션 시스템 ACE(ASIC Emulator)를 제안한다. ACE는 EDIF 번역기, 라이브러리 변환기, 기술 맵퍼, 회로 분할기, LDF 생성기를 포함하는 에뮬레이션 소프트웨어와 에뮬레이션 보드, 논리 분석기를 포함하는 에뮬레이션 하드웨어로 구성된다. 기술 맵퍼는 회로 분할과 논리 함수식 추출, 논리 함수의 최소화, 논리 함수식의 그룹핑의 세 과정으로 이루어지며, 같은 기본 논리 블록에 할당되는 출력의 적항과 변수들을 많이 공유하게 하여 기본 논리 블록 수와 최대 레벨 수를 최소화한다. 에뮬레이션 보드의 배선 구조와 FPGA 칩이 갖는 제한 조건들을 만족시키면서 서로 다른 칩 사이에 연결된 신호선뿐만 아니라 서로 다른 그룹 사이에 연결된 신호선 수의 최소화를 목적 함수로 하는 새로운 회로 분할 알고리듬을 제안한다. 여러 FPGA 칩으로 구성된 에뮬레이션 보드는 완전 그래프와 부분 그래프를 결합한 새로운 배선 구조로 회로의 크기에 관계없이 칩 사이의 지연 시간을 최소화하도록 설계하였다. 논리 분석기를 이용하여 구현된 회로에서 검증을 원하는 내부신호에 대한 파형을 PC의 모니터로부터 관측할 수 있다.

제안한 에뮬레이션 시스템의 성능을 평가하기 위하여 상용 회로중 하나인 화면4분할기 회로를 에뮬레이션 보드 상에 설계하여 동작 시간과 기능을 확인한 결과, 14.3MHz의 실시간 동작과 함께 기능이 완전함을 확인할 수 있었다.

Abstract

In this paper, an ASIC emulation system called ACE (ASIC Emulator) is proposed. It can produce the prototype of target ASIC in a short time and verify the function of ASIC circuit immediately. The ACE is consist of emulation software in which there are EDIF reader, library translator, technology mapper, circuit partitioner and LDF generator and emulation hardware including emulation board and logic analyzer.

Technology mapping is consist of three steps such as circuit partitioning and extraction of logic function, minimization of logic function and grouping of logic function. During those procedures, the number of basic logic blocks and maximum levels are minimized by making the output to be assigned in a same block sharing product-terms and input variables as much as possible. Circuit partitioner obtain chip-level netlists satisfying some constraints on routing structure of emulation board as well as the architecture of FPGA chip. A new partitioning algorithm whose objective function is the minimization of the number of interconnections among FPGA chips and among group of FPGA chips is proposed.

The routing structure of emulation board take the advantage of complete graph and partial crossbar structure in order to minimize the interconnection delay between FPGA chips regardless of circuit size. logic analyzer display the waveform of probing signal on PC monitor that is designated by user. In order to evaluate the performance of the proposed emulation system, video quad-splitter, one of the commercial ASIC, is implemented on the emulation board. Experimental results show that it is operated in the real time of 14.3MHz and functioned perfectly.

* 正會員, 漢陽大學校 電子工學科

(Dept. of Electronics Eng., Hanyang University)

接受日字:1999年6月10日, 수정완료일:1999年9月27日

Ⅰ. 서 론

최근 반도체 설계 기술이 발전함에 따라 FPGA(Field Programmable Gate Array)가 출현함으로서 기존의 주문형 반도체인 게이트 어레이나 스탠다드 셀에 의존해 왔던 ASIC(Application Specific Integrated Circuit) 시장에 큰 변화가 일어나고 있다. 특히 FPGA가 갖는 장점인 재프로그램 가능성과 end-user에 의한 신속한 회로 설계로 많은 각광을 받고 있어 ASIC시장에서 주목받는 소자로 부각되고 있다.[1] 근래에 들어서는 ASIC으로 구현되는 회로의 크기가 커지고 복잡해짐에 따라 그의 검증 또한 중요한 과제가 되었다. 보다 빠르고 정확한 검증은 일각을 다투는 전자 산업의 시장에서 보다 상대적으로 신속하게 제품을 생산할 수 있고, 제품 생산까지의 시행착오를 줄여줄 수 있기 때문에 상당히 중요한 의미를 갖는다.

ASIC을 검증하기 위해서 종래에는 주로 소프트웨어적인 시뮬레이션에 의존하여 왔으나, 시뮬레이션 소프트웨어와 컴퓨터의 성능이 최근 급속도로 증가하는 회로의 크기 및 복잡도를 따라가지 못하고 있다. 장시간을 필요로 하는 시뮬레이션 기법 및 방대한 테스트 벡터로 인하여 검증이 불가능한 설계를 단시간 내에 검증 가능케 하는 에뮬레이션 시스템이야말로 ASIC 설계의 필수 불가결한 요소이다. 결과적으로 에뮬레이션 시스템을 사용함으로서 목적회로를 빠르고 정확하게 제작, 검증할 수 있으므로 최종적으로 원하는 제품의 빠른 시장진출을 유도할 수 있으며 재설계로 인한 시간 및 노력을 최소화할 수 있다. 따라서 최근 이러한 시뮬레이션의 단점을 극복하고 보다 정확한 검증을 위하여 실제 상황으로 빠른 검증을 수행할 수 있는 하드웨어적 에뮬레이션 기법에 많은 관심이 집중되고 있다.[2-4] FPGA를 이용한 검증은 FPGA의 성능 및 구현 가능한 회로의 크기 등에 의해 하나의 FPGA를 이용하여 구현할 수 없는 대규모의 회로인 경우 필요할 때마다 여러 개의 FPGA를 연결하기 위해 래핑 또는 PCB를 다시 제작해야하는 문제가 생기므로 FPGA간의 결선이 미리 형성되어 있는 범용성 FPGA 즉 ASIC 에뮬레이터를 출현시켰다. ASIC 에뮬레이터를 이용하면 설계된 회로를 실제적으로 FPGA에 제작하여 검증할 수 있으며 또한 회로의 변경에 따른 새로운 추가 비용과 시간을 절감할 수 있다.

본 논문에서는 여러 개의 FPGA 칩을 어떤 특정한 배선 구조로 연결한 후, 재프로그램 가능한 에뮬레이션 보드에 설계된 회로를 자동으로 제작하여 회로의 기능을 신속하게 검증할 수 있는 에뮬레이션 시스템인 ACE(ASIC Emulator)를 개발하여 제안한다. ACE는 FPGA칩들과 이들 사이의 배선 구조, 논리 분석기 등의 에뮬레이션 하드웨어와 주어진 회로의 연결 정보로부터 자동으로 에뮬레이션 보드에 ASIC을 구현해 주는 에뮬레이션 소프트웨어들로 구성된다. 에뮬레이션 보드의 배선구조는 구현된 회로의 동작속도에 영향을 미치는 요인중의 하나이다. 제안하는 배선구조는 어떠한 신호선이라도 최대 지연 시간이 일정하며 지연 시간을 최소화하도록 설계하였다. 칩단위 회로 분할기는 제안하는 배선 구조에 적합하도록 설계되었으며 그룹 사이에 연결된 신호선의 개수를 최소화함으로써 지연 시간을 최소화하고 배선으로 인한 핀의 부족현상을 방지하였다. 본 논문의 구성은 다음과 같다. 2장에서는 전체 시스템의 구성에 대하여 설명한다. 에뮬레이션 시스템을 구성하는 하드웨어는 3장에서 기술하고, 소프트웨어는 4장에서 기술한다. 5장에서는 실험 및 결과에 대하여 설명하고 6장에서 결론을 맺는다.

Ⅱ. 전체 시스템 구성

ACE의 전체적인 시스템 구성은 그림 1과 같다. ACE는 ASIC 회로의 EDIF 기술을 입력으로 하여 최종적으로 에뮬레이션 보드 상에 회로를 제작한 후 검증하는 시스템이다. ACE를 이용한 에뮬레이션 과정은 그림 1과 같은 과정으로 진행된다.

설계자가 회로 편집기 상에서 작성하거나 VHDL로부터 자동 생성된 회로를 EDIF 형식의 파일로 출력한다. EDIF 변환기는 회로를 기술하는 EDIF 형식의 파일을 분석하여 전체 회로를 구성하는 회로 소자간의 연결 정보를 추출한다. EDIF 변환기로부터 출력된 연결 정보는 특정 회사에서 개발된 ASIC 라이브러리로 기술되어 있으므로, 에뮬레이션 보드상의 FPGA 칩에 회로를 제작하기 위해서는 FPGA 회사에서 제공하는 라이브러리들의 연결 정보로 변환시켜주어야 하는데 이와 같은 역할을 라이브러리 변환기가 수행한다. 기술 맵퍼가 회로 소자간의 연결 정보로부터 FPGA 칩을 구성하는 기본 논리 블록 단위로 주어진 회로를 맵핑하여 기본 논리 블록간의 연결 정보를 출력한다. 기본 논리 블록 단

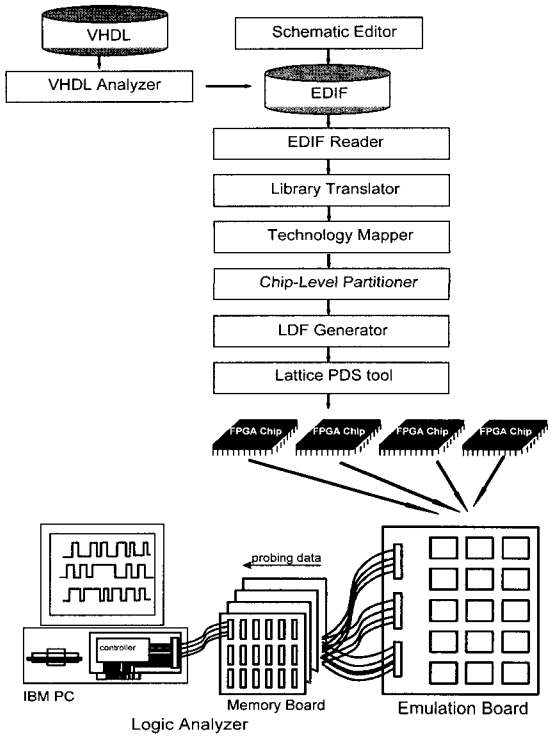

그림 1. ACE의 전체적인 구성도
Fig. 1. Overall configuration of ACE.

위로 맵핑된 회로의 크기가 하나의 FPGA 칩에 구현할 수 있는 용량을 초과할 경우에는 칩단위 회로 분할기가 기본 논리 블록간의 연결 관계, 검증 신호선, 에뮬레이션 보드의 제한 조건을 만족시키면서 FPGA 칩 사이에 연결 가능한 신호선의 개수가 최소화 되도록 기본 논리 블록들을 여러 개의 FPGA 칩에 나누어 할당하여 분할한다. LDF(Lattice Design Format) 생성기는 맵핑된 기본 논리 블록의 논리 정보, 핀 정보를 고려하여 Lattice사의 PDS(pLSI Development System) 소프트웨어가 읽어들일 수 있는 LDF 형식으로 기술된 화일을 생성한다. 출력된 LDF 화일은 여러 개의 FPGA 칩에 다운로드하여 에뮬레이션 보드에 ASIC 회로를 제작한 후, 논리 분석기를 이용하여 검증을 원하는 신호선의 파형을 관측한다.

Ⅲ. 에뮬레이션 하드웨어

1. 에뮬레이션 보드의 구조

에뮬레이션 보드는 설계한 ASIC 회로가 구현될 FPGA 칩과 연결 포트들로 구성된 PCB 보드를 말한다. 에뮬레이션 보드 설계시 고려해야 할 사항은 주어진 회로를 실시간에 가깝게 동작시키기 위한 효율적인 배선구조의 설계와 회로의 신속한 검증에 있다.[6] ASIC에 구현되는 회로가 점점 커지고 복잡해짐에 따라 자체 회로만의 검증보다는 전체 시스템을 구축한 상태에서의 검증이 중요하다. 따라서 에뮬레이션 보드는 충분한 외부 입출력 핀을 확보해야 하며 자체 회로의 검증을 위하여 검증용 핀의 개수가 많을수록 좋다. 기존의 제안된 연결 구조는 직접 연결[7], 완전 크로스바, 부분 크로스바[8] 구조로 나뉜다. 직접 연결 구조는 상, 하, 좌, 우의 인접한 칩들과 직접적으로 연결하는 구조이므로 멀리 떨어진 칩 사이의 배선을 위해서는 중간의 칩들을 거쳐야 하기 때문에 지연 시간이 커지고 핀의 소모가 많아진다. 크로스바는 로직이 구현된 칩의 연결을 위하여 사용되는 스위칭 칩을 말한다. 완전 크로스바 구조는 하나의 핀에 대하여 다른 칩의 모든 핀들과의 연결이 크로스바 칩을 경유하여 가능하도록 만든 구조이므로 모든 핀들을 제한 없이 연결할 수 있는 장점이 있으나, 스위칭 소자의 크기가 과도하게 커지는 문제점을 가지며 하나의 핀에 대한 연결 요구를 갖는 핀의 개수에 비하여 낭비가 많기 때문에 실제 회로의 응용에는 적합하지 않다. 부분 크로스바 구조는 크로스바 칩을 경유하여 인접한 칩의 일부 핀들만 부분적으로 연결하는 구조로 두 핀 사이의 연결을 위해서는 오직 하나의 크로스바 칩만 거치면 되므로 배선으로 인한 지연 시간이 일정한 장점이 있다. 그러나 인접한 칩 사이의 연결도 반드시 하나의 칩을 거쳐야 하므로 중요도가 높은 신호선을 직접 연결할 수 없는 단점을 가진다. 새롭게 개발한 에뮬레이션 보드는 완전 그래프 구조와 부분 크로스바 연결 구조의 장점을 취하여 개선한 구조이다.

먼저 4개의 칩을 한 그룹으로 묶어 완전 그래프 구조로 연결하고, 각 그룹 사이의 연결은 배선 전용 칩을 이용하여 실현한다. 완전 그래프 구조를 이용한 이유는 크로스바 칩을 경유하지 않고 직접 연결 가능하도록 하여 지연 시간을 줄일 수 있기 때문이다. 특히 지연 시간에 민감한 임계 신호선들은 그룹 내에서의 연결이 가능하다. 그룹간의 연결은 오직 하나의 배선 전용 칩을 경유하여 연결함으로써 그룹 사이의 지연 시간을 일정하게 유지 할 수 있다. 에뮬레이션 보드상의 모든 신호선의 연결은 직접 또는 최대 하나의 배선 전용 칩

만을 통과하여 모두 연결 가능하므로 작은 지연 시간을 갖는 장점이 있다.

에뮬레이션 보드의 FPGA 칩간 연결 구조를 그림 2에 나타낸다. 에뮬레이션 보드는 Lattice사의 ispLSI1048 FPGA 칩 15개와 40개의 핀을 갖는 입출력 포트 18개로 구성된다. 사용된 ispLSI1048칩은 최대 8K 게이트의 회로를 구현할 수 있고 전체 120개의 핀 중에서 96개를 일반적인 입출력 핀으로 이용할 수 있다. 에뮬레이션 보드에서 사용한 15개의 칩 중에서 12개를 논리 구현용으로 사용하고 3개를 배선 전용 칩으로 사용하였다. 검증 가능한 회로의 크기는 이론적으로 최대 96K 게이트이지만, 회로를 FPGA에 구현할 때 배치 및 배선의 어려움 때문에 통상적으로 게이트 이용률을 약 30%에서 40%정도로 계산하여 실제 검증 가능한 회로의 크기는 30K게이트에서 40K게이트이다. 외부 시스템과의 연결을 위해 총 288개의 입출력 핀을 가지고 있으며 내부 신호의 검증을 위해 최소 96개의 핀을 가진다. 논리 구현용 칩 12개를 세 개의 그룹으로 나누어 각 그룹당 4개씩의 칩을 완전 그래프 구조로 연결하고, 그룹간의 연결을 위하여 나머지 3개의 칩을 배선 전용 칩으로 사용한다. 배선 전용 칩은 모든 논리 구현용 칩의 핀들과 연결되어 있으므로 그룹간의 배선을 프로그램할 수 있으며 배선 전용 칩 사이에는 연결 관계가 없다. 각 그룹에 속한 FPGA 칩은 96개의 일반 입출력 핀 중 그룹 내의 다른 칩과 각각 20개씩 연결되어 모두 60개의 핀이 한 그룹 내에서 다른 세 개의 칩과의 연결을 위해 사용된다. 서로 다른 그룹간의 연결 관계는 그룹과 그룹간의 연결을 위해 3개의 배선 전용 칩을 사용한다. 배선 전용 칩은 각각의 논리 구현용 칩으로부터 6핀씩 총 24핀을 서로 다른 그룹에 속한 칩간의 배선을 위하여 하나의 배선 전용 칩에 각각 연결한다. 따라서 하나의 그룹에서 다른 그룹과의 연결을 위해 72개의 연결선이 존재한다. ASIC 에뮬레이터의 목적 중 하나는 설계된 회로의 신속한 검증에 있다. 따라서 에뮬레이션 보드는 충분한 외부 입출력 핀을 확보해야 하며 자체 회로의 검증을 위하여 검증용 핀의 개수가 많을수록 좋다. 이러한 요구에 만족시키기 위해 총 288개의 핀을 외부 입출력을 위해 사용할 수 있도록 설계하였다. 최소 96개의 핀을 회로 검증용으로 사용할 수 있다.

하나의 논리 구현용 칩에는 18개의 핀을 외부 입출력 핀으로 사용한다. 따라서 하나의 그룹에는 72개, 전

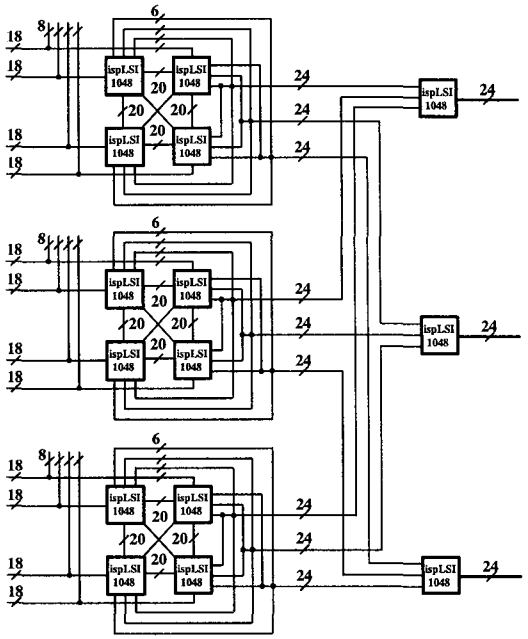

그림 2. FPGA 칩간 연결 구조
Fig. 2. Interconnection of FPGAs.

체 논리 구현용 칩으로부터는 216개의 외부 입출력 신호를 연결할 수 있다. 하나의 배선 전용 칩에서도 외부와의 연결을 위해 24개의 입출력 핀이 존재하므로 외부 입출력 핀의 개수는 총 288개가 된다. 내부 회로의 검증을 위한 핀은 기본적으로 외부 입출력 핀과 공유된다. 하나의 논리 구현용 칩에서 나오는 18개의 외부 입출력 핀 중 8개의 핀이 검증을 위한 핀으로 사용될 수 있다. 그러므로 한 그룹당 모두 32개의 핀을 내부 회로 검증용 핀으로 사용할 수 있다.

2. 논리 분석기

ASIC 에뮬레이터로 구현된 회로가 정상적으로 동작하는지 검증하기 위해서는 원하는 신호선의 파형을 관측하고 분석하는 논리 분석기가 필수적이다. 논리 분석기는 검증을 원하는 지정된 신호선에 대하여 회로 동작 중의 상태 변화를 일정한 샘플링 간격으로 저장한 후 지정된 신호선의 파형을 그래픽으로 출력한다. 논리 분석기의 하드웨어는 데이터를 저장할 SRAM 및 부가적인 논리 회로로 이루어져 있는 메모리 보드와 이러한 메모리 보드들을 제어하고 PC와의 인터페이스를 담당하는 인터페이스 보드로 구성된다. 메모리 보드는 하나의 보드에 여러 개의 SRAM을 탑재시킬 수 있는데 현재는 4개의 SRAM으로 구성되어 있다. 각 SRAM은

8개의 채널로 사용되므로 모두 32채널을 갖게 된다. 이러한 메모리 보드를 64개까지 연결할 수 있도록 설계되어 있다. 따라서 최대 211개의 채널을 갖도록 설계되어 있다. 신호 파형의 샘플링은 사용자가 지정한 일정한 샘플링 간격에 따라서 주기적으로 신호의 샘플을 취해서 SRAM에 저장한다. SRAM에 데이터를 모두 저장해서 메모리가 차게 되면 PC에 하드웨어 인터럽트 신호를 발생하여 데이터의 샘플링이 종료되었음을 알리고 SRAM에 저장된 데이터를 디스크에 화일의 형태로 저장한다. SRAM에 저장된 데이터를 화일로 저장할 때에는 메모리 보드의 SRAM을 PC가 직접 접근해서 데이터를 읽는다. 논리 분석기가 다루어야 하는 전체 데이터의 수는 매우 많으므로 어드레스 공간도 메가바이트 단위를 필요로 한다. 그러나, PC에서 사용할 수 있는 어드레스의 빈 공간은 제한되어 있으므로 SRAM의 전체 주소 공간을 여러 개의 뱅크로 나누어 사용한다. 각 SRAM의 뱅크로부터 샘플링 데이터를 읽어 들일 때마다 PC상에 어드레스 맵핑을 하여 각 뱅크로부터 차례로 읽어오게 된다. 데이터가 모두 디스크에 저장되면 사용자의 요구에 따라서 데이터를 모니터 상에 그래픽 파형으로 출력하도록 프로그램되어 있다.

IBM PC 상의 메모리 어드레스 공간의 E0000H 번지부터 64K바이트 정도는 대부분의 PC에서 비어 있으므로 이 번지 공간을 논리 분석기의 SRAM 주소로 할당한다. 논리 분석기의 메모리 공간은 64K 바이트를 초과하므로 SRAM 주소 공간을 여러 개의 뱅크로 나누어서 각 뱅크를 차례로 PC의 E0000H 번지에 맵핑한다. 뱅크의 선택은 논리 분석기의 내부에 있는 레지스터 값에 의해서 이루어지고 이 레지스터 값으로부터 해당 SRAM의 칩 선택 신호를 제어함으로써 각 SRAM의 PC로의 맵핑이 이루어진다. 각 뱅크의 데이터 비트는 8비트로 되어 있으므로 하나의 뱅크에는 8개의 신호 파형을 저장한다. SRAM을 제어하는 상태는 두 가지이다. 하나는 샘플링 데이터를 에뮬레이션 보드로부터 받아들이는 상태이고, 다른 하나는 받아들인 데이터를 PC의 디스크로 가져가는 상태이다. 두 가지 상태에 대해서 CS(chip select), /WE(write enable), /OE(output enable), 어드레스 신호를 멀티프렉싱하여 SRAM에 가해진다. 그림 3의 MUX는 이를 위한 멀티프렉서를 개념적으로 나타낸 것으로 여러 개의 SRAM으로 이루어져 있는 논리 분석기의 SRAM 부분을 하나의 개념적인 SRAM으로 표현한 것이다.

데이터를 일정 시간 간격으로 샘플링하므로 어드레스도 어드레스 카운터에 의해 샘플링 간격에 맞추어 증가시킨다. 카운터에 공급되는 클락의 주파수가 샘플링 간격을 결정하므로 카운터의 클락을 조정하므로써 샘플링 간격을 조절한다.

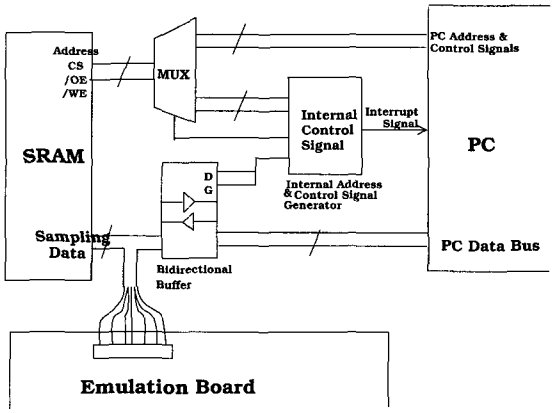

그림 3. 논리 분석기의 하드웨어 구성
Fig. 3. Hardware configuration of logic analyzer.

카운터는 계속 어드레스를 증가시키다가 출력 캐리를 발생하면 PC에 인터럽트를 발생시키고 이 인터럽트 처리 프로그램이 SRAM에 저장된 데이터를 하드디스크에 화일로 저장한다. PC가 SRAM으로부터 데이터를 읽어들이는 것은 PC가 자체 시스템 내의 DRAM으로부터 데이터를 읽어 들이는 방법과 동일하다. SRAM의 데이터는 에뮬레이션 보드로부터 샘플링을 위한 입출력 포트로부터 입력되어 SRAM에 저장되었다가 PC의 시스템 버스를 통해서 PC내로 읽혀져서 처리된다. PC의 시스템 버스와 에뮬레이션 보드의 입출력 포트와의 데이터 충돌을 막기 위해서 버퍼를 데이터 버스 상에 두어 PC 시스템이 데이터를 읽어 들일 때만 인에이블된다.

Ⅳ. 에뮬레이션 소프트웨어

1. EDIF 번역기

설계자가 회로 편집기상에서 작성하거나 VHDL로부터 자동 생성된 회로를 EDIF의 형태로 출력하면 EDIF 번역기는 회로를 기술하는 EDIF 형식의 화일에 대하여 LEX와 YACC를 이용하여 구문 분석(syntax analysis)과 의미 분석(semantic analysis)을 하여 전체 회로를 구성하는 회로 소자간의 연결 정보를 추출한다. 또한

개발한 EDIF 번역기는 계층적으로 기술된 EDIF 입력 화일도 처리가 가능하도록 설계되어 있다.

EDIF 번역기의 출력 파일은 기본 입력(primary input) 신호선의 리스트와 개수, 기본 출력(primary output) 신호선의 리스트와 개수, 인스턴스(instance)의 종류와 개수, 인스턴스가 참조하는 셀 레퍼런스(cell reference)의 종류와 개수 그리고 각각의 신호선에 연결된 인스턴스 이름, 셀 레퍼런스 이름, 셀 레퍼런스의 입출력 포트 이름, 입출력 형태 등에 관한 정보와 검증 신호선의 이름, 특정 신호선에 대한 타이밍 정보 등을 포함한다.

2. 라이브러리 변환기

EDIF 번역기로부터 출력된 연결 정보는 특정 회사에서 개발된 ASIC 라이브러리로 기술되어 있고, ACE의 에뮬레이션 보드는 Lattice사의 FPGA 칩으로 구성되어 있으므로 주어진 회로를 에뮬레이션 보드 상에 구현하기 위해서는 Lattice사에서 제공하는 라이브러리들간의 연결 정보로 변환시켜야 하는데 이와 같은 역할을 라이브러리 변환기가 수행한다. 회로를 기술하는 라이브러리의 종류와 형식은 매우 다양하므로 라이브러리 변환기의 입력 요구 조건에 맞는 타이밍을 고려한 입출력 포트 맵핑 정보를 입력 화일의 형태로 기술해 주어야 한다. ACE는 어떠한 종류의 라이브러리라도 라이브러리 변환기의 입력 기술 형식에 따라 입력해 주면 Lattice사의 FPGA에서 적용 가능한 형태의 연결 정보로 출력해 준다. 라이브러리 변환기는 세개의 화일, 즉 src.lib, lat.lib, trans.lib를 입력으로 한다. src.lib는 EDIF 입력 화일에서 사용한 라이브러리의 종류와 입출력 포트에 대한 정보를 기술하고, lat.lib는 Lattice사의 라이브러리의 종류와 입출력 포트에 대한 정보를 기술하고, trans.lib는 EDIF 기술에 사용한 라이브러리를 Lattice 라이브러리로 변환할 때 추가되는 새로운 인스턴스와 새로운 신호선에 대한 정보를 포함한 새로운 연결 정보를 기술한다. 라이브러리 변환기의 입력 화일의 형식은 표 1과 같다.

Lattice사에서 제공하는 라이브러리에는 마크로 라이브러리가 존재한다. 제공되는 각각의 마크로 라이브러리들은 Lattice칩에 맞게 최적화되어 있기 때문에 라이브러리 변환시 가능하면 이러한 마크로 라이브러리를 이용하여 변환하도록 설계한다. 예를 들어, 3×8 디코더는 Lattice사에서 제공하는 라이브러리에는 존재하지

표 1. 라이브러리 변환기의 입력 화일

Table 1. Input files of library translator.

src.lib
aoi3(z:O, a1:I, a2:I, b1:I, b2:I, c:I) dec38(z0:O, z1:O, z2:O, z3:O, z4:O, z5:O, z6:O, z7:O, a0:I, a1:I, a2:I)
lat.lib
inv(z0:O, a0:I) and2(z0:O, a0:I, a1:I) nor3(zn0:O, a0:I, a1:I) dec4e(z0:O, z1:O, z2:O, z3:O, s0:I, s1:I, s2:I)
trans.lib
aoi3(z, a1, a2, b1, b2, c) and2($newnet0, a1, a2) and2($newnet1, b1, b2) nor3(z, $newnet0, $newnet1, c) dec38(z0, z1, z2, z3, z4, z5, z6, z7, a0, a1, a2) inv($newnet0, a0) dec4e($newnet1, $newnet2, $newnet3, $newnet4, a2, a1, $newnet0) dec4e($newnet5, $newnet6, $newnet7, $newnet8, a2, a1, a0) inv(z0, $newnet1) inv(z1, $newnet2) inv(z2, $newnet3) inv(z3, $newnet4) inv(z4, $newnet5) inv(z5, $newnet6) inv(z6, $newnet7) inv(z7, $newnet8)

않으므로 Lattice사에서 제공되는 마크로 라이브러리인 2×4 디코더 2개로 구성한다. Lattice사에서 제공되는 마크로 라이브러리를 조합해도 원하는 논리 기능을 구현할 수 없을 경우에는 해당 논리 블록을 기본 게이트들의 조합으로 펼쳐놓은 후 입출력 포트 맵핑을 실시한다.

3. 기술 맵퍼

기술 맵퍼는 라이브러리 변환기로부터 출력된 회로의 연결 정보로부터 Lattice사의 FPGA 칩을 구성하는 기본 논리 블록인 GLB 단위로 주어진 회로를 분할하여 각각의 기본 논리 블록의 구조에 알맞게 할당하고 칩단위 회로 분할기에 기본 논리 블록간의 연결 정보를 출력한다. 기술 맵퍼의 목적 함수는 기본 논리 블록 수의 최소화와 기본 입력으로부터 기본 출력까지 걸쳐 있는 기본 논리 블록 수, 즉 최대 레벨 수의 최소화로 설정하였다. Lattice사에서 제공하는 ispLSI1048 칩의 기본 논리 블록은 기본적으로 AND-배열과 OR-배열로

구성되어 있고 AND-배열과 OR-배열을 자유롭게 프로그램해서 원하는 함수를 구현한다. 하나의 기본 논리 블록은 출력 수가 최대 4개, 입력 수가 최대 16개, 적항(product term) 수가 최대 20개까지 구현할 수 있다. 내부에는 플립플롭이 4개가 포함되어 있는데 이들은 모두 같은 클락 신호와 리셋 신호를 공유한다. 기술맵핑 과정은 크게 회로 분할과 논리 함수식 추출, 논리 함수의 최소화, 논리 함수식의 그룹핑의 세 과정으로 이루어진다.

1) 회로 분할과 논리 함수식 추출

회로 분할과 논리 함수식 추출 과정에서는 먼저 주어진 회로를 기본 출력에서 기본 입력으로 향하는 방향성 그래프로 모델링한 후, 기본 논리 블록의 출력 함수로 할당될 가능성이 있는 신호선을 잘라서 각 출력 콘의 부울 대수식을 생성한다. 연결 정보로부터 조합 논리 회로에는 피드백 루프가 없다는 가정 하에 기본 출력으로부터 신호선의 입력 방향으로 콘을 형성시키면서 함수식을 구한다. 콘의 루트 노드는 기본 출력, 플립플럽의 입력, 매크로 라이브러리의 입력, 삼상 버퍼의 입력, XOR 게이트의 입력, 기본 논리 블록의 용량을 초과하는 함수식의 출력 신호선이고, 리프 노드는 기본 입력, 플립플롭의 출력, 매크로 라이브러리의 출력, 기본 출력, XOR 게이트의 출력, 기본 논리 블록의 용량을 초과하는 함수식의 출력 신호선이다. 리프 노드에 기본 출력이 있는 것은 그 변수가 피드백 되어 다시 사용될 수 있기 때문이다. 이때 각각의 단일 출력 함수의 부울 대수식의 일부가 서로 중복되는 것을 허락한다. 이는 칩의 지연 시간이 주로 함수의 레벨에 영향을 받기 때문에 불필요하게 중간 변수를 삽입시킬 이유가 없기 때문이다.

단일 출력 함수를 구하기 위해서 리프 노드에서부터 루트 노드로 반복하여 함수식을 구한다. 단일 출력 함수를 구할 때 함수간 논리 중복을 허용하고 각각의 게이트에서 함수식이 로직 블록의 용량을 초과하는가를 확인하고 초과하는 경우는 함수식을 구하는 것을 중지하고 새로운 콘을 형성한다. 논리 함수식을 칩의 구조에 맞게 구하는 것이 회로를 분할하는 것과 같은 의미를 갖는다. 이는 함수식을 구하는 것이 연결 정보로부터 필요한 최종적인 정보이기 때문이다. 필요한 함수식은 루트 노드의 조건에 해당하는 것이고 루트 노드의 함수식을 구하는 과정에서 기본 논리 블록의 용량에 맞게 함수식을 추출한다. 이때 기본 논리 블록의 용량을 초과하는 부분은 새로운 루트 노드와 리프 노드가 된다.

2) 논리 함수의 최소화

논리 함수의 최소화의 목적은 이 과정에서 함수식에서 불필요한 변수가 제거되고 함수식들간의 정확한 적항 공유와 입력 변수의 공유 정도를 알 수 있기 때문이다. 보다 많은 변수와 적항의 공유는 논리 블록의 이용률을 높일 수 있다. Lattice사의 FPGA 칩의 구조가 기본적으로 AND-OR 배열의 이단 구조이기 때문에 이단 논리 최소화기인 Espresso를 이용하였다. 최소화된 함수식을 기본 논리 블록이 허용하는 용량에 맞게 여러 개를 모아서 각 블록에 할당하면 기본 논리 블록간의 연결 정보를 얻을 수 있다. 논리 최소화 과정의 출력은 각각의 단일 출력 함수에 대한 입력 변수의 개수, 적항 수, 공유하는 입력 변수의 개수, 공유하는 적항의 개수 등에 대한 정보로 구성된다. 구해진 단일 출력 함수식을 기본 논리 블록의 AND-OR 배열 구조에 맞게 Espresso를 이용하여 논리 최소화를 거친 후 적항과 변수의 공유가 많은 것을 기본 논리 블록의 용량에 맞게 모아 나간다. 이때 단일 출력의 함수식에서 서로 공유가 많은 식들은 하나의 로직 블록으로 할당되도록 한다.

3) 논리 함수식의 그룹핑

함수식의 그룹핑은 회로 분할 과정과 논리 최소화를 통해서 얻은 각각의 함수식을 기본 논리 블록의 제약 조건을 만족시키면서 기본 논리 블록에 단일 출력 함수들을 모아서 할당하는 과정이다. 논리 함수식의 그룹핑은 다음과 같은 전처리 과정을 수행한다. XOR 게이트는 기본 논리 블록내의 하드웨어적인 XOR를 이용하기 때문에 반드시 기본 논리 블록의 출력으로 할당한다. 같은 클락 신호 또는 같은 리셋 신호를 공유하는 플립플럽들은 가능하면 하나의 기본 논리 블록 내에 구현되어야 하므로 하나의 그룹으로 모은다. 입출력 중에서 삼상 버퍼는 기본 논리 블록에서 처리할 수 없고 기본 논리 블록의 사용에도 기여하지 않으므로 그룹핑에서 제외시킨다. Lattice사에서 제공하는 하드 타입(hard-type)의 마크로 셀은 이미 기본 논리 블록의 구조에 맞게 최적화되어 있으므로 그룹핑에서 제외한다.

그룹핑 알고리듬은 먼저 가장 많은 변수를 사용하는 씨드(seed)를 선정하고 선택된 씨드와 기본 논리 블록의 제한 조건에 따라서 적항을 가장 많이 공유하는 함

수식들을 최대 4개까지 계속 찾아가면서 기본 논리 블록에 함수식을 할당해 나가는 과정을 반복하는 것이다. 씨드가 되는 함수는 가장 많은 변수를 사용하는 것을 선택하는데 이는 논리 함수식을 구하는 과정에서 모든 함수식은 기본 논리 블록의 크기를 초과하지 않도록 생성되었기 때문이다. 함수식의 그룹핑이 끝나면 기본 논리 블록간의 연결 정보를 출력한다.

제안하는 알고리즘을 몇 가지 예제에 대하여 실험을 수행하였다. 화면4분할기 회로에 대하여 실험한 결과를 Lattice 프로그램을 이용한 결과와 비교하였다. Lattice 프로그램을 이용한 경우 69개의 기본 논리 블록을 차지하고 구현된 회로의 최대 레벨 수는 8로 나타났다. 제안한 알고리즘은 61개의 기본 논리 블록을 차지하고 최대 레벨 수는 6으로 나타나서 기본 논리 블록 수와 최대 레벨 수에 대하여 향상된 결과를 얻을 수 있었다.

4. 칩단위 회로 분할기

회로의 분할은 ASIC 에뮬레이터에 있어서 성능을 결정 짓는 중요한 과정이다. 분할의 결과에 따라 배치, 배선되어야 할 FPGA 칩의 개수 및 칩간의 연결을 위한 신호선의 수가 결정되기 때문이다. 칩단위 회로 분할기는 기술 맵퍼에서 출력된 기본 논리 블록간의 연결 정보, 검증 신호선 및 입출력 신호선 정보 등을 고려하여 사용하는 FPGA 칩의 용량에 맞게 분할하며, 특히 에뮬레이션 보드가 갖는 제한 조건을 만족시키면서 FPGA칩간의 연결된 신호선 수의 최소화 및 FPGA 칩으로 구성되는 그룹 간 신호선 수의 최소화를 목적 함수로 설정하였다.

분할의 입력 화일은 파라미터 화일과 기본 논리 블록 단위의 연결 정보 화일이 있다. 파라미터 화일은 크게 에뮬레이션 보드 정보, 사용되는 FPGA칩의 정보, 그리고 분할에 사용되는 각종 파라미터 정보로 구성된다. 에뮬레이션 보드 정보는 4개의 칩으로 구성되는 그룹 수, 그룹내 칩 수, 배선 전용 칩 수, 그룹간 연결 신호선 수, 논리 구현용 칩간의 신호선 수, 논리 구현용 칩과 배선 전용 칩간의 신호선 수, 논리 구현용 칩에 연결되는 입출력 신호선 수, 논리 구현용 칩에 연결되는 검증 신호선 수 등이 포함된다. 사용되는 FPGA칩에 대한 정보는 FPGA 칩의 종류, 칩 내부에 포함되는 기본 논리 블록 수, 전체 핀 수, 칩의 속도, 칩의 메가 블록 수 등으로 구성된다. 분할에 사용되는 파라미터 정보는 칩 내의 기본 논리 블록 사용율, 각 분할의 크기 편차, 분할 그룹 수, 초기 분할 방법, 검증 신호선과 출력 인에이블 신호의 웨이트 등으로 구성된다. 기본 논리 블록간의 연결 정보 화일에는 신호선 수, 기본 논리 블록 수, 기본 입출력 신호선 수, 검증 신호선 수와 출력 인에이블 신호선 수 등으로 구성되는 글로벌 정보와 기본 입출력 신호선 리스트, 검증 신호선 리스트, 출력 인에이블 신호선 리스트가 포함된다. 그리고 각각의 기본 논리 블록에 연결된 신호선 연결정보를 표현한다.

본 논문에서 제안하는 칩단위 회로 분할 알고리듬은 두 그룹의 분할뿐만 아니라 임의의 여러 그룹으로 분할이 가능한 다중 분할 방법으로 기본적으로 계층 구조적으로 분할이 가능하다. 개발된 분할 알고리즘은 그림 4와 같다. 먼저 기본 논리 블록간의 연결 정보를 가지고 있는 연결 정보 파일과 분할에 관한 제한 조건들을 기술한 파라미터 화일을 읽어 들인다. 입력 화일을 모두 파싱한 후 분할을 위해 사용되는 데이터 구조를 구성하는데 데이터 구조는 크게 셀 어레이, 신호선 어레이, 신호선 분포 테이블, 그리고 셀을 이동했을 경우의 이득을 표현하는 이득 테이블로 구성된다. 이득 테이블은 각 셀을 다른 분할 그룹으로 이동시켰을 때 얻을 수 있는 이득에 대한 정보를 갖고 있으며 분할 그룹이 k개일 때 이득 테이블의 개수는 k(k-1)개이다. 이득에 따라서 최대 이득으로부터 최소 이득 중의 한곳에 연결 리스트 구조로 매달아 놓음으로써 분할 개선시에 후보 셀들을 빠른 시간 내에 탐색할 수 있다. 또한 신호선의 웨이트 계산을 쉽게 하기 위해서 신호선 분포 테이블을 구성하는데 이는 각 신호선에 대하여 신호선에 연결된 셀의 개수에 대한 정보를 갖고 있어서 전체 신호선 웨이트를 쉽게 계산하고 셀의 이동시에는 셀의 이동에 따른 정보의 갱신을 하는데 이용한다. Level은 회로 분할의 계층을 의미한다. Level의 값은 DeterminePartLevel() 함수에서 결정된다. 물론 에뮬레이터의 연결 구조 자체가 계층 구조를 지원해야 한다. 각 FPGA의 GLB 수, 각 그룹의 FPGA 수와 GLB 이용률을 고려하여 몇 개의 계층으로 구현 가능한지를 계산하여 변수 Level에 할당한다.

분할 그룹 수를 결정하기 위해서 필요한 정보는 전체 회로를 구성하는 기본 논리 블록의 개수 (NoOfGLB), 하나의 FPGA칩을 구성하는 최대 기본 논리 블록의 개수(MaxNoOfGLB), 그리고 FPGA칩의 기본 논리 블록 이용률 (%GLB)이며, 분할 그룹수(PartWay)는 다음과

같은 식으로 구해진다.

$$PartWay = \frac{NoOfGLB}{MaxNoOfGLB * \%GLB}$$

분할 그룹수가 결정되면 초기 분할을 수행하는데 초기 분할은 각 분할 그룹의 균형 조건을 만족시키도록 균일한 분할 크기로 각 기본 논리 블록을 할당한다. 균형 조건은 다음과 같이 설정한다.

$$Average(P) - AreaMargin \le |P| \le Average(P) + AreaMargin$$

여기서 |P|는 분할 그룹의 크기, *Average(P)*는 전체 분할 그룹의 평균 크기, *AreaMargin*은 파라미터 화일에서 주어지는 값으로 분할 그룹의 크기 편차를 조절하는 요소로써 입력 화일에서 영으로 주어지는 경우 기본 논리 블록의 크기 중에서 최대값을 갖게 된다. 초기의 분할은 균형 조건을 만족하는 범위 내에서 연결도는 고려하지 않고 랜덤하게 기본 논리 블록을 각 분할 그룹에 할당한다. 신호선의 분포를 나타내는 신호선 분포표는 세로축에 각 신호선을, 가로축은 각 분할 그룹을 나타낸다. 따라서 임의의 신호선에 대한 컷 웨이트는 신호선에 연결된 기본 논리 블록이 속해있는 분할 그룹의 수에서 1을 뺀 값에 그 신호선의 고유 웨이트를 곱한 것과 같다. 즉 전체 신호선 웨이트(Total Cut Weight)는 다음과 같은 식으로 표현된다.

$$TotalCutWeight = \sum_{i=1}^{N} [\, NoOfPart(n_i) - 1 \,] * Weight(n_i)$$

여기서, *Weight(ni)*는 신호선 *ni*의 고유웨이트, *NoOfPart(ni)*는 신호선 *ni*에 연결된 기본 논리 블록을 포함하는 분할 그룹의 개수를 의미한다. 신호선의 고유 웨이트는 특정 신호선에 대하여 하나의 칩 또는 하나의 그룹 내에서만 연결이 가능하도록 제어하기 위해서 사용한다. 웨이트의 크기가 클수록 여러 칩에 나뉘어 배선될 가능성이 작아진다. 특히 임계 경로(critical path)상의 신호선들은 웨이트 값을 크게 설정하여 여러 칩에 걸쳐서 배선되는 현상을 줄이는데 이용할 수 있다.

모든 기본 논리 블록에 대하여 현재 속한 분할 그룹으로부터 다른 분할 그룹으로 이동할 경우의 신호선 분포표의 변화에 따른 이득을 계산하여 이득표를 구성한다. 이득의 최대치는 기본 논리 블록에 연결된 최대 핀의 개수가 되고, 이득의 최소치는 최대 핀의 개수에 -1을 곱한 값을 갖는다. 다음 과정은 기본 논리 블록을 이동시켜 가면서 초기 분할의 해를 향상시키는 분할 개선 과정으로써 *CellMovingSequence()*함수에서 수행한다. 먼저 최대 이득을 갖는 기본 논리 블록 셀을 선택하여 셀을 이동시킬 경우 균형 조건과 에뮬레이션 보드의 제한조건을 고려하여 이동시키고 셀의 이동에 대한 신호선 분포표와 이득표를 갱신한다. 이때 초기에는 에뮬레이션 보드의 제한조건을 위배하더라도 이동하기 위한 후보 셀로 선택하지만 점진적으로 제한조건을 강화시켜서 나가는 방법을 사용하였다. 이것은 초기에 보드의 제한 조건을 위배하더라도 최종적으로 최적에 더 가까운 해를 얻을 가능성을 이용하기 위함이다. 전체 신호선의 총 웨이트가 감소했으면 셀의 이동에 대한 정보를 갖고 있는 스택을 모두 비우고 위의 과정을 반복하게 된다. 만약 총 웨이트가 감소되지 않았을 때는 스택에 이동하는 셀과 이동 방향에 대한 정보를 기록하고 위의 과정을 반복하는데 그 이유는 현재는 분할 결과가 전보다 더 개선될 여지가 있기 때문이다.

```
ChipLevelPartitioningAlgorithm()
  {
  GLBNetListFileRead();
  ParameterFileRead();
  DeterminePartLevel();
  for(Level=0;Level<PartLevel;Level++){
      DeterminePartWay();
      InitialPartitioning();
      PartitionRateSetting();
      MakeAlphaTableAndGainTable();
      CutWeight= ComputeTotalWeight();
      NewWeight= CellMovingSequence();
      while(1){
          if(NewWeight<CutWeight){
              CutWeight= NewWeight;
              NewWeight= CellMoving
              Sequence();
              NotImproved= 0;
          }else{
              NotImproved++;
              If(NotImproved>3)
                  break;
              NewWeight= CellMoving
              Sequence();
          }/*if*/
      }/*while*/
  }/*for*/
}/*Algorithm*/
```

그림 4. 칩단위 회로 분할 알고리즘
Fig. 4. Chip-level partitioning algorithm.

위와 같은 과정을 이동시킬 셀 후보가 존재할 때까지 반복하여 수행한다. 분할의 출력은 각 신호선에 대하여 연결된 칩들의 리스트와 각 칩에 포함되는 기본 논리 블록들의 리스트 그리고 각 칩 내에서의 기본 논리 블록의 위치 정보로 구성된다.

제안하는 분할 알고리즘을 효율성을 보이기 위하여 MFM과 비교하였다. MFM은 FM 알고리즘을 다중 분할이 가능하도록 변형한 것이다. 화면4분할기 회로를 12개의 FPGA 칩에 나누어 분할하여 배치시켰다. 실험결과 MFM는 FPGA 칩의 외부로 연결되는 신호선의 개수가 94개, 미배선 신호선은 5개로 나타난 반면, 제안하는 분할 알고리즘은 외부 신호선은 84개, 미배선 신호선은 0개로 100% 배선 가능한 결과를 얻었다. 결과적으로 에뮬레이션을 위한 분할 알고리즘 설계시 외부 신호선의 개수보다도 보드의 구조가 갖는 제한 조건을 고려한 분할 알고리즘의 개발이 더 효과적이다.

5. LDF Generator

기본 논리 블록의 연결 정보로부터 에뮬레이션 보드의 구조에 맞게 칩단위의 분할, 칩 내의 기본 논리 블록의 배치, 칩 내의 핀 할당을 수행한 결과와 각각의 칩에 할당된 기본 논리 블록의 논리 함수식으로부터 칩을 프로그래밍하기 위한 입력 화일인 LDF(Lattice Design File) 화일을 생성한다. 생성된 LDF 화일을 이용하여 각각의 칩에 대하여 Lattice사의 PDS tool을 이용하여 칩내 의 배선을 수행한다. 칩 내에서의 완전한 배치, 배선이 끝나면 칩을 프로그래밍 할 수 있는 비트 스트림인 JEDEC 화일로 칩을 프로그래밍 한다. Lattice사의 FPGA 칩은 시스템 내에서 프로그램이 가능하기 때문에 특별한 프로그래밍 장치가 필요 없이 에뮬레이션 보드에 프로그래밍에 필요한 5개의 핀만 모든 칩에 직렬로 배선한 후 모든 칩을 데이지 체인방식으로 연결하여 에뮬레이션 보드 상에서 순차적으로 프로그램 할 수 있다.

V. 실험 및 고찰

ACE의 성능을 검증하기 위해 실제 사용되고 있는 몇 가지 회로에 대한 실험을 수행하였다. 실험에 사용한 회로는 16비트 카운터와 움직임 검출 회로로써, 카운터의 경우 비교적 높은 주파수에서 동작하는 것을 확인하였으며 움직임 검출 회로의 경우 실시간으로 정상적인 동작을 하였다. 에뮬레이션 보드의 성능을 검증하기 위해 먼저 16비트 카운터에 대하여 세 가지 경우로 나누어 실험하였다. 16비트 카운터는 회로의 크기가 작기 때문에 하나의 칩만으로도 구현할 수 있었으며 이 경우 60MHz이상에서도 정상적으로 동작하는 것을 확인하였다. 두 번째는 카운터를 한 그룹을 구성하는 네 개의 칩에 나누어 구현하였는데 이때는 20MHz에서 동작하였다. 마지막으로 카운터를 12개의 칩에 나누어 배선 전용 칩을 이용하여 연결하였을 경우에는 10MHz 이상에서 정상적으로 동작하였다. 그림 5는 12개의 논리구현용 칩과 3개의 배선 전용 칩으로 구성된 에뮬레이션 보드의 실제 사진을 보여준다.

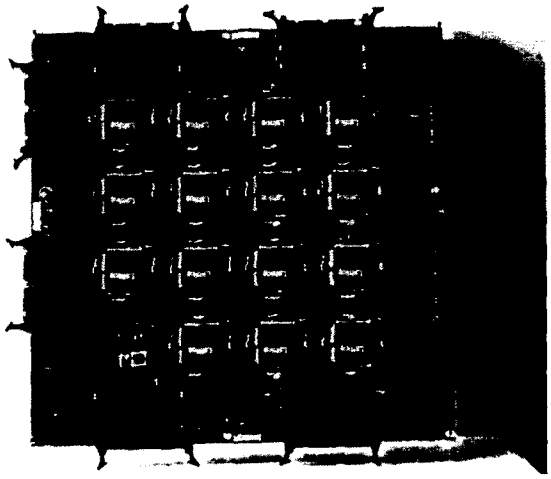

그림 5. 에뮬레이션 보드
Fig. 5. Picture of emulation board.

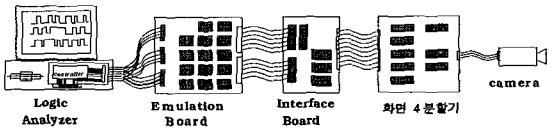

그림 6. 움직임 검출 회로의 에뮬레이션
Fig. 6. Emulation on motion-detection circuit.

다음은 본 연구실에서 직접 설계, 제작한 화면 4분할기의 움직임 검출 회로에 대해 실험하였다. 움직임 검출 회로는 어느 특정한 영상을 메모리에 저장하고 현재 카메라에 잡힌 영상과 저장되어 있는 두 영상을 비교하여 두 영상간에 차이가 발생하면, 즉 움직임이 검출되면 경보를 발생하는 기능을 가진 회로이다. 움직임 검출 회로는 3K개 이상의 기본 게이트들로 구성되어 있다. 움직임 검출 회로는 화면 4분할기 내에서 사용하

기 위해서 설계되었으며, 14.3MHz의 클락을 사용하고 외부의 다른 시스템으로부터 A/D 변환 신호 및 동기 신호, 클락 신호를 입력으로 받으며 외부에 있는 메모리로 억쎄스한다. 에뮬레이션을 수행하기 위해 메모리 소자 및 다른 아날로그 회로를 구현한 인터페이스 기판을 따로 제작하여 그림 6와 같이 화면 4분할기 및 에뮬레이션 보드와 연결하였다. 실제 실험 사진은 그림 7과 같다. 에뮬레이션을 수행한 결과 화면 4분할기 회로의 클락 주파수인 14.3MHz에서 성공적으로 동작함을 확인하였고 원하는 신호선을 논리분석기를 이용하여 관측하였다.

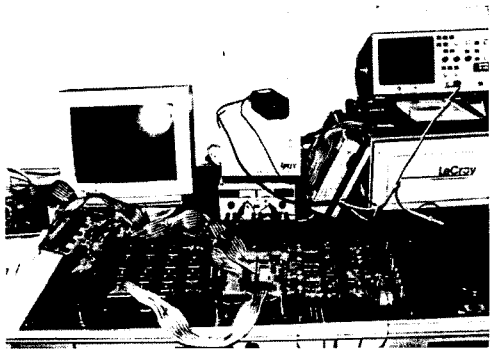

그림 7. 움직임 검출 회로의 에뮬레이션 사진
Fig. 7. Picture of Emulation on motion-detection circuit.

Ⅳ. 결 론

본 논문에서는 ASIC 설계회로를 빠른 시간 내에 구현 및 검증할 수 있는 에뮬레이션 시스템을 제안하였다. 에뮬레이션 보드를 구성하는 논리 구현용 FPGA 칩과 배선 전용 FPGA 칩 사이의 연결을 위해 완전그래프 구조와 부분 연결 구조를 개선한 새로운 배선 구조를 설계하여 회로의 크기에 관계없이 작은 지연 시간으로 FPGA 칩간 연결을 수행할 수 있었다. 메모리 뱅크 개념을 이용한 논리분석기는 최대 211개의 신호선을 검증할 수 있는 구조로 설계되었으며 PC와 인터페이스가 가능하므로 검증 신호선을 화면으로 직접 확인할 수 있다. 기본논리블록 수와 최대 레벨 수를 최소화를 목적으로 하여 주어진 회로에 대하여 기술 맵핑한 결과, 상용 소프트웨어보다 감소된 기본논리블록 수와 최대 레벨 수를 얻을 수 있었다. 칩단위 회로분할기를 이용하여 에뮬레이션 보드의 배선구조와 FPGA 칩이 갖는 제한 조건들에 위배됨이 없이, 여러 FPGA 칩 사이에 연결된 신호선뿐만 아니라 서로 다른 그룹사이에 연결된 신호선 수를 최소화하여 회로를 분할할 수 있었다.

ACE의 성능을 평가하기 위하여 상용회로 중의 하나인 화면4분할기의 움직임 검출회로를 에뮬레이션 보드에 제작한 후 검증하고자 하는 신호선의 파형을 PC 모니터로 관측해 본 결과, FPGA 칩에 제작된 회로는 실제 동작 시간으로 정확히 동작함을 확인할 수 있었다. 개발된 에뮬레이션 시스템을 ASIC 설계에 활용할 경우 설계 현장에서 회로의 동작을 신호선의 검증을 통하여 직접 확인 할 수 있으며, 회로가 변경되더라도 별도의 하드웨어가 필요 없이 빠른 시간 안에 FPGA를 이용한 시제품을 제작할 수 있어 개발 시간 및 비용을 대폭 절감할 수 있으리라 본다.

앞으로의 연구 과제로는 대규모 회로를 수용하기 위하여 확장성을 갖는 에뮬레이션 보드의 구조에 관한 연구가 지속되어야 하고, 또한 FPGA 칩 외에 DSP 칩과 CPU 칩이 탑재되어 상호간에 인터페이스가 가능한 에뮬레이션 보드의 개발이 뒤따라야 하겠다.

참 고 문 헌

[1] 김기현, 정정화, "FPGA 설계 기술 및 응용", 대한전자공학회지 제20권 11호, pp.61-68, 1993년

[2] Stephen Walters, "Computer-Aided Prototyping for ASIC-Based Systems",IEEE Design & Test of Computers, pp.4-10, 1991.

[3] Michael Butts, Jone Batcheller and Joseph Varghese, "An Efficient Logic Emulation System", Proc. Int'l Conf. Computer Design, pp.170-177, 1992.

[4] Thomas A. Horvath and Norman H. Kreitzer, " Hardware Emulation of VLSI Design", Proc. 3rd Annual IEEE ASIC Seminar and Exhibit, p13-6.1 - p13-6.4, 1990.

[5] Lattice Data Book, Lattice Semiconductor Corp., 1994.

[6] Wai-Kai Mak and D. F. Wong, "On Optimal Board-Level Routing for FPGA-based Logic Emulation", Proc. 32nd DAC, pp.552-556, 1995.

[7] Scott Hauck, Gaetano Borriello and Carl Ebeling, "Mesh Routing Topologies for

Multi-FPGA Systems", Proc. Int'l Conf. Computer Design, pp. 170-176, 1994.

[8] Michael Butts, Jon Batcheller and Joseph Varghese, "An Efficient Logic Emulation System", Proc. Int'l Conf. Computer Design, pp.170-177, 1992.

저 자 소 개

鄭 正 和(正會員)

1975년 한양대학교 전자공학과(학사). 1977년 한양대학교 전자공학과(석사). 1981년 일본 와세다대학교 전자통신공학과(박사). 1979년~1980년 일본 NEC 중앙연구소 위촉연구원. 1983년~1984년 KIET(Korea Institute of Electronics & Technology) 위촉연구원. 1986년~1987년 미국 Berkeley대학 초빙교수. 1993년~1994년 대한전자공학회 CAD 및 VLSI 분과 위원장. 1995년~1996년 대한전자공학회 교육이사. 1996년~1997년 영국 Newcastle대학 초빙교수. 1997년~1998년 대한전자공학회 편집이사. 1997년~1999년 한양대학교 정보통신원 원장. 1999년~현재 대한전자공학회 학술이사. 1981년~현재 한양대학교 전자전기공학부 교수. 관심분야는 VLSI의 CAD, ASIC 에뮬레이션 시스템 개발, MPEG 디코더/인코더 설계, 통신 회로 설계, 특히 무선 모뎀 칩 개발

柳 光 基(正會員)

1986년 한양대학교 전자공학과(학사). 1988년 한양대학교 전자공학과(석사). 1991년 한양대학교 전자공학과 박사과정 수료. 1991년~1994년 육군사관학교 교수부 전자공학과 교수. 1994년~현재 한양대학교 강사. 관심분야는 VLSI의 CAD, ASIC 에뮬레이션 시스템 개발, 초미세 배선 설계