特輯

VHDL을 이용한 마이크로프로세서의 FPGA 구현

李 昇 禑, 趙 仲 彙
仁川大學校 工科大學 電子工學科

Ⅰ. 서 론

1987년 IEEE 표준안 1076으로 VHDL (VHSIC Hardware Description Language)이 발표된 이후 오늘날 이를 이용한 디지탈 시스템 설계 및 ASIC의 설계는 점점 확대되고 있다.[1] 국내에서도 한국 VHDL 사용자 모임 (KVUG : Korea VHDL User's Group)을 중심으로 VHDL의 기초와 응용에 대한 교육과 VHDL 툴의 실습을 통한 VHDL의 사용이 증가하고 있다.[2-4]

그런데 VHDL을 이용한 시뮬레이션의 단순한 사용에서 더 나아가 VHDL을 이용한 논리 회로의 합성과 칩 제작을 행하여야 VHDL의 사용이 더욱 증대할 것이며 설계 기술의 질적 향상을 꾀함으로써 ASIC 설계에 대한 국내 산업의 발전에 도움이 될 것이다. 이와같은 VHDL 응용에 대한 교육은 다양한 형태로 요구되고 있는데 이를 구현할 수 있는 방법으로 VHDL을 이용한 모델링, 시뮬레이션, 논리합성과 FPGA 툴에 의한 칩의 제작이 제시되고 있다. FPGA 칩의 특성으로 인한 크기와 동작 속도에의 한계성은 있으나 설계 즉시 구현을 할 수 있는 장점으로 인하여 이 방법의 이용은 교육에서는 보다 적극적으로 도입되어야 할 것이다.

본 연구실에서는 여러가지 VHDL 툴과 FPGA 툴을 이용하여 다양한 디지탈 시스템 설계에 대한 설계 및 구현의 교육을 행하고 있는데 설계 과제 중의 하나가 마이크로 프로세서 구현 및 마이크로 컴퓨터의 제작이다.

상용화된 마이크로 프로세서의 설계 및 구현을 위하여는 여러가지 문제점이 있어[10-12] 이를 위한 제한된 과제가 필요하게 되어 8-비트이면서 다양한 하드웨어, 소프트웨어 응용 및 확장이 가능하도록 구조를 갖는 마이크로 프로세서로 CR8000 칩과 이를 이용하여 구성한 마이크로 컴퓨터 SD8000을 정의하였다.

본 稿의 2장에서는 설계하고자 하는 마이크로 프로세서의 명령어 집합과 이를 위한 하드웨어, 소프트웨어의 특성을 정의하고 3장에서는 논리합성이 가능한 레지스터 전송 레벨의 VHDL 모델링에 대하여 기술하며 4장에서는 Xilinx FPGA에의 구현에 대하여 기술한다.

Ⅱ. SD8000의 하드웨어 특성

디지탈 시스템 설계, VHDL 모델링, FPGA 제작 및 ASIC 설계 교육을 위하여 설계한 8-비트 마이크로 컴퓨터는 다음과 같은 9개의 모듈로 구성된다.

(1) CR8000 마이크로프로세서
(2) 데이타 메모리
(3) 입,출력 디바이스
(4) 키보드
(5) 모드 선택 스위치
(6) 클럭 생성기
(7) 디스플레이 모듈
(8) 리셋 키
(9) 인터페이스 모듈

각 모듈에 대한 기능과 입, 출력 핀에 대한 기본적인 기능은 다음과 같다.

CR8000의 핀은 데이타 메모리 또는 입,출력 디바이스와의 데이타 전송을 위하여 양방향 8-비트의 데

이타 전용 버스 (data_bus)를 지니고 있으며 메모리 내의 256개 번지 지정을 위한 8-비트의 어드레스 버스 (addr_bus)를 지니고 있다. 또한, 메모리로부터의 읽기 동작을 요구하는 rd_n과 쓰기 동작을 요구하는 wr_n의 2개 출력 신호와 읽기와 쓰기를 할 것이 데이타 메모리인지 입,출력인지를 결정하기 위해 io_m_n신호가 있다. 한편, 클럭 생성기로부터 입력되는 시스템 클럭신호가 있으며, 시스템의 초기화 동작을 요구하는 reset_n 입력 신호가 있다.

데이타 메모리와 입,출력 디바이스 메모리는 각각 칩 선택(cs_n)신호와 읽기 또는 쓰기 동작을 나타내는 rd_n과 wr_n이 있다. 또한, 8-비트의 어드레스 버스와 8-비트의 데이타 버스가 있다.

키보드는 '0' 부터 'F' 까지의 16진 데이타를 입력하기 위한 16개의 키와 1개의 enter key로 구성된다.

동작 상태를 구분하는 2개의 선택 모드 스위치는 그림 1의 동작을 3가지로 구분하는 선택 스위치인데 Load 스위치와 Read 스위치이다. 스위치의 구성이 (Load, Read)=(1, 0)일때는 Load 모드로 CR8000의 기계어 코드를 키보드를 사용하여 메모리에 입력하는 상태를 나타낸다. (Load, Read) = (0, 1)의 경우는 메모리에 저장된 데이타를 어드레스와 함께 디스플레이 모듈에 표시하는 디버그 상태이며 그 외의 상태일 때는 메모리에 입력된 기계어 코드를 수행하는 상태를 나타낸다.

클럭 생성기는 SD8000에서 필요로 하는 시스템 클럭을 생성 출력하는 모듈이며, 디스플레이 모듈은 4개의 7-세그먼트로 구성되는데 상위 2개는 8-비트의 어드레스 값을 표시하며, 하위 2개는 지정된 어드레스의 8-비트 데이타 값을 나타내도록 정의한다. 한편, 리셋 키는 SD8000의 초기화를 행하기 위하여 설정한 키이다.

한편, 위와같은 8개의 모듈 사이의 정보 교환을 위한 기능을 하나로 모듈화 한 것이 인터페이스 모듈이다.

다음에서는 SD8000의 모듈중 8-비트 마이크로 프로세서인 CR8000의 하드웨어 및 소프트웨어의 특성을 설명한다.

1. CR8000의 하드웨어 특성

CR8000은 표 1과 같은 총 23개의 1-바이트의 명령어 길이를 갖는 명령어를 수행하는데 Instruction Pointer (IP) 레지스터를 addr_bus로 출력하여 메모리로부터 페치한다. Opcode는 명령어 수행 동안에 필요로 하는 여러가지의 제어 신호를 생성하기 위하여 수행하는 동안 저장되어야 하는데 이를 위하여 8-비트의 레지스터로 Instruction Register (IR)을 설정한다.

표 1. CR8000 명령어 집합 일람표

mnemonic	opcode	cycle	operation
ADD rs	00000 sss	1	r0 <= r0 + rs
INC rs	00001 sss	1	rs <= rs + 1
SUB rs	00010 sss	1	r0 <= r0 - rs
DEC rs	00011 sss	1	rs <= rs - 1
NOT rs	00100 sss	1	rs <= NOT rs
AND rs	00101 sss	1	r0 <= r0 AND rs
OR rs	00110 sss	1	r0 <= r0 OR rs
XOR rs	00111 sss	1	r0 <= r0 XOR rs
CMP rs	01000 sss	1	compare(r0,rs)
SHL rs	01001 sss	1	rs <= rs(6:0) & '0'
SHR rs	01010 sss	1	rs <= '0' & rs(7:1)
SCL rs	01011 sss	1	rs <= rs(6:0) & cf
SCR rs	01100 sss	1	rs <= cf & rs(7:1)
SAR rs	01101 sss	1	rs <= rs(7) & rs(7:1)
JMP cond,data8	01110 ccc	2	pc <= data8
MOV rd,rs	11 ddd sss	1	rd <= rs
LDI rd,data8	10000 ddd	2	rd <= data8
LOAD rd,[data8]	10001 ddd	2	rd <= [data8]
STORE rs,[data8]	10010 sss	2	[data8] <= rs
IN rd,[data8]	10011 ddd	2	rd <= [data8]
OUT rs,[data8]	10100 sss	2	[data8] <= rs
NOP	10101	1	No operation
HALT	10111	1	halt

ccc	jump condition
000	Unconditional Jump
001	If sign_flag = '1' then Jump
010	If zero_flag = '1' then Jump
101	If sign_flag = '0' then Jump
110	If zero_flag = '0' then Jump

ddd or sss	register	property
000	r0	accumulator
001	r1	general register
010	r2	
011	r3	
100	r4	
101	r5	
110	r6	
111	r7	indir register

그림 1은 마이크로 프로세서 구조의 일반적 개념에 따라 CR8000을 4개의 블럭으로 구성한 것이며 각 블럭에 대한 개략적인 기능은 다음과 같다.

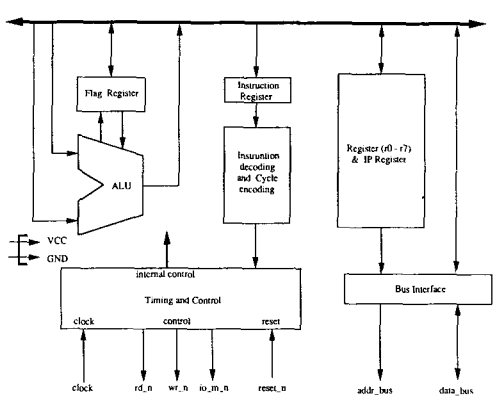


그림 1. CR8000의 블럭 구성도

1) 레지스터 블럭

IP 레지스터와 8개의 범용 레지스터 (r0 - r7)로 구성되며, 명령어에서의 범용 레지스터 지정을 위하여는 3-비트의 레지스터 지정 코드가 요구된다.

IP 레지스터는 메모리에 기억된 프로그램 명령어의 절대번지 위치를 기억하는 레지스터로 명령어의 페치에 따라 1 씩 증가되고, Jump 계열 명령어의 수행에 따라 지정된 값으로 변화한다. 한편, CR8000의 초기화 상태일때는 X"00"로 초기화 된다.

2) 산술 및 논리 연산 블럭

산술 및 논리 연산 (ALU) 블럭은 산술 연산 명령어 (ADD, INC, SUB, DEC), 논리 연산 명령어 (NOT, AND, OR, XOR, CMP), 데이타 이동 명령어 (MOVE) 와 쉬프트 연산 명령어 (SHL, SHR, SCL, SCR, SAR)의 처리를 수행한다. 또한, 레지스터 블럭의 IP 레지스터에 대한 1 증가 또는 지정된 값으로의 변화를 위한 동작을 수행한다.

ALU 블럭의 수행에 필요로 하는 데이타는 1개 또는 2개인데 수행하는 명령어에 표시된 레지스터의 값과 어큐뮬레이터 (accumulator)인 r0 또는 IP의 값을 레지스터 블럭으로 부터 전달 받는다. ALU 블럭 수행의 결과는 지정된 레지스터에 저장되거나 어큐뮬레이터 인 r0 또는 IP 레지스터에 저장된다.

한편, 플래그 블럭은 산술 연산 명령어 (ADD, INC, SUB, DEC)의 수행 결과에 대한 부호 비트를 기억하는 부호 플래그 비트와 앞의 산술 명령어 연산 결과의 제로 여부를 나타내는 제로 플래그 비트의 각 정보를 생성하는 기능과 이를 저장하는 2-비트의 레지스터로 구성된다.

3) 해독 및 제어 블럭

CR8000의 설계를 위하여 1개의 시스템 클럭을 1개의 머신 스테이트로 정의하며 4개의 머신 스테이트를 1개의 머신 사이클로 정의한다. CR8000의 모든 명령어는 머신 사이클 단위로 수행되는데 명령어의 머신 스테이트와 머신 사이클에 따른 동작은 정확하게 정의되어야 한다.

Jump 계열의 명령어와 메모리 또는 I/O 장치와의 데이타 전송을 행하는 명령어 (LDI, LOAD, STORE, IN, OUT)는 2 개의 머신 사이클을 필요로 하며, 기타의 명령어는 1 개의 머신 사이클을 필요로 한다.

따라서, 2개의 머신 사이클을 표현하기 위한 2진 카운터 블럭이 필요하며 머신 스테이트를 표현하기 위하여는 one-hot 코드를 이용한 4진 카운터 블럭이 필요하다.

또한 각 머신 사이클과 머신 스테이트에 따른 지정된 동작을 수행하기 위하여는 ALU, 레지스터, 멀티플렉서와 같은 데이타 블럭들은 여러가지 제어 신호를 필요로 하는데 제어신호는 IR의 값, 머신 사이클, 머신 스테이트 및 플래그 레지스터로부터의 정보등을 조합하여 생성된다. 이 기능을 수행하기 위하여 제어 신호 생성 블럭이 필요하다.

한편, HALT 명령어의 수행에 따라 CPU의 동작의 중단과 CPU의 연속적인 동작 상태를 표시하기 위한 플립플럽이 필요하다.

4) 버스 인터페이스 블럭

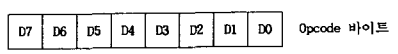
8-비트의 어드레스 버스는 레지스터 블럭의 IP 레지스터 또는 r7 레지스터의 값을 전달받아 메모리 또는 입,출력 장치에 전달하는 기능을 갖는다. 8-비트 데이타 버스는 STORE 와 OUT 명령어의 수행에 따른 일부의 머신 스테이트에 대하여는 출력 모드이며, 기타의 모든 머신 스테이트에 따른 동작에 대하여는 입력 모드이다. 따라서, 데이타 버스는 양버향의 신호로 설정하여야 하므로 데이타 버스에의 출력은 3-상태 버퍼를 사용한다.

2. CR8000의 소프트웨어 특성

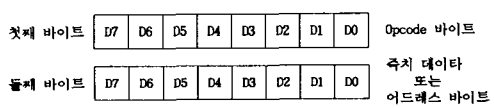
1) 데이타 및 명령어 형식

CR8000의 데이타는 8-비트 2진수로 2의 보수 형태로 표현된다.

CR8000의 어셈블러 언어는 1-바이트 또는 2-바이트의 기계어 코드로 어셈블 되어 메모리에 저장되어 있으며 일반적인 명령어 형식은 그림 2와 같으며 명령어에 따른 opcode 바이트와 명령어의 의미는 표 1과 같다.



(a) 1-바이트 길이의 명령어



(b) 2-바이트 길이의 명령어

그림 2. CR8000 명령어의 형식

2) 머신 스테이트와 머신 사이클

CR8000의 명령어의 수행을 위한 시스템 클럭, 머신 스테이트 및 머신 사이클의 관계는 그림 3과 같다.

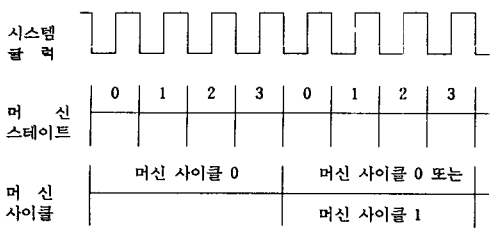


그림 3. 시스템 클럭, 머신 스테이트 및 머신 사이클의 정의

1개의 머신 사이클을 필요로 하는 명령어에 대한 각 머신 스테이트에 대한 동작을 표현하면 표 2와 같다.

표 2. 1-머신 사이클 명령어의 각 머신 스테이트의 동작

머신 스테이트	동 작
스테이트-0	addr_bus <= IP; rd_n <= '0'; wr_n <= '1'; io_m_n <= '0';
스테이트-1	IR <= data_bus;
스테이트-2	명령어의 해독; IP <= IP + 1; rd_n <= '1'; wr_n <= '0';
스테이트-3	명령어의 수행;

또한 2개의 머신 사이클을 필요로 하는 명령어의 머신 시이클-1의 동작은 스테이트-0 부터 스테이트-2 까지의 동작은 동일하며, 스테이트-3에서의 동작은 정의되지 않는다.

한편 머신 사이클-2에서의 동작을 각 명령어에 따라 표현하면 표 3과 같다.

표 3. 2-머신 사이클 명령어의 각 머신 스테이트의 동작

머신 스테이트	명 령 어	동 작
스테이트-0	공 통	addr_bus <= IP; rd_n <= '0'; wr_n <= '1'; io_m_n <= '0';
스테이트-1	JUMP LOAD STORE IN OUT	r7 <= data_bus;
	LDI	rd <= data_bus;

스테이트-2	JUMP	if 조건=true then
		IP <= r7;
		else
		IP <= IP + 1;
		end if;

	LDI	IP <= IP + 1;
	LOAD	IP <= IP + 1; addr_bus <= r7;

	STORE	IP <= IP + 1;
		addr_bus <= r7;
		data_bus <= rs;
		rd_n <= '1';
		wr_n <= '0';

	IN	IP <= IP + 1; addr_bus <= r7; io_m_n <= '1';

	OUT	IP <= IP + 1;
		addr_bus <= r7;
		data_bus <= rs;
		rd_n <= '1';
		wr_n <= '0';
		io_m_n <= '1';

스테이트-3	JUMP, LDI	no-operation;
	LOAD, IN	rd <= data_bus;
	STORE, OUT	스테이트-2와 동일함

Ⅲ. CR8000의 합성이 가능한 VHDL 모델링

2장에서 분석한 CR8000의 동작 특성에 대하여 알고리즘 레벨에서의 VHDL 모델링을 행하여 시뮬레이션을 행함으로써 설계 사양과 동작 특성에 대한 정확한 분석을 확인한다. 그런데 현재 상용화 되어 있는 모든 VHDL 합성기들은 레지스터 전송 (RT : Register-Transfer) 레벨의 VHDL 소스 코드들에 대하여만 논리 합성을 행하는 한계성을 지니고 있다. [5,7]

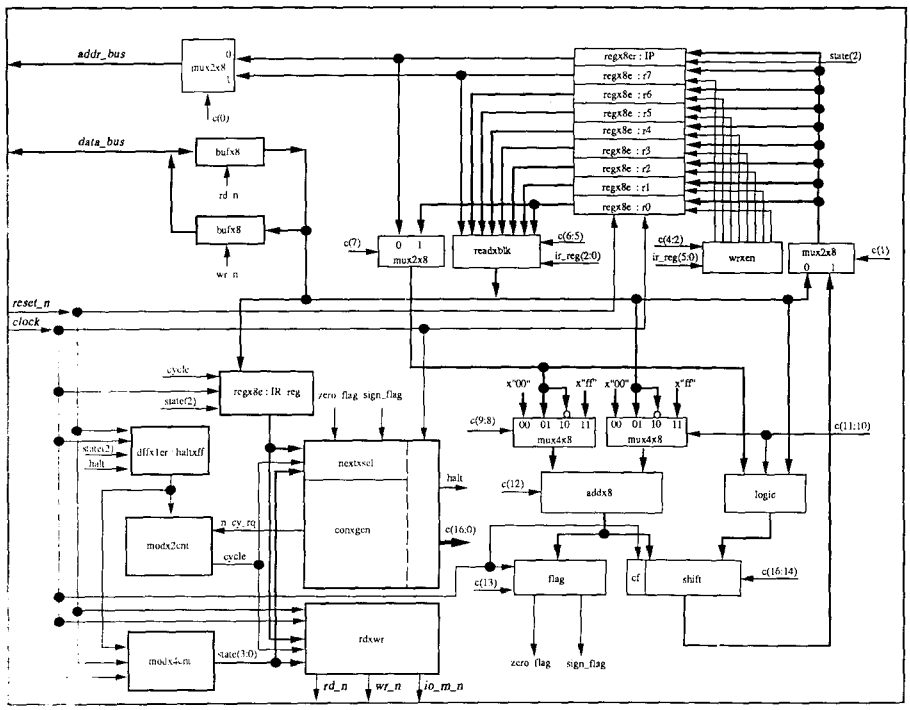


그림 4. CR8000의 RT 레벨 블럭 구성도

따라서 CR8000의 동작 특성을 RT 레벨로 분석 하는데 그림 4는 CR8000의 RT 레벨 분석에 대한 하나의 예로 30개의 설계 블럭으로 구성한 구성도이다.

그림 4에 대하여 design tree [3,11 13]를 구성하고 tree의 leaf-node는 state 스케쥴링이 필요없는 PROCESS 문의 IF, CASE 및 LOOP 문 또는 concurrent signal 할당 문을 이용한다. 또한 non-leaf-node에 대하여는 component 참조 문을 이용하여 VHDL 모델링을 행하면 VHDL 합성기들은 이에 대한 논리회로도를 생성한다.

Leaf-node의 설계 블럭은 조합논리회로, 순서논리회로로 구분하며 순서논리회로의 경우는 비동기신호 및 동기신호에 대한 정확한 모델링을 사용하는 VHDL 툴에 따라 행하여야 한다. [3,11 13]

한편 알고리즘 레벨의 설계 사양에서 RT 레벨 블럭도의 추출 결과에 따라 논리 합성의 성능은 차이가 있게 된다. 이와같은 문제점은 상위 레벨 합성기와 분할 툴에 대한 상용화의 시점까지는 설계자의 경험에 의존 할 수 밖에 없다.

또한 technology 라이브러리의 hard macro들을 고려한 모델링을 하면 논리 합성의 성능이 보다 우수하게 되는데 이를 "Performance Driven VHDL Modeling"이라 한다. 이를 위하여는 VHDL 툴과 사용할 라이브러리에 대한 보다 철저한 해석과 이용이 요구된다.

CR8000의 VHDL 모델링과 논리 합성을 위하여 본 稿에서는 (1) Viewlogic VHDL 툴 [6], (2) MyVHDL [8] 툴을 이용한다. Viewlogic VHDL 툴과 LSI 10k design library를 사용한 결과 총 게이트 수는 1997개이고, 여기에서 320개의 CLB (Configurable Logic Block)를 가진 Xilinx 3090PC84를 이용하여 총 205개의 CLB가 사용되었다.

Ⅳ. CR8000의 FPGA로의 구현

널리 사용되고 있는 FPGA 칩은 크게 static RAM, EEPROM, EPROM 방식과 anti- fuse 방식의 4가지가 있는데 여기서는 static RAM 방식을 택하기로 하였다. [9]

CR8000의 FPGA 구현을 위하여 static RAM방식의 Xilinx 3000 시리즈의 FPGA를 선택하고 생성된 논리 회로도에 대하여 XACT 툴에 의한 배치 와 배선을 행한다. [10]

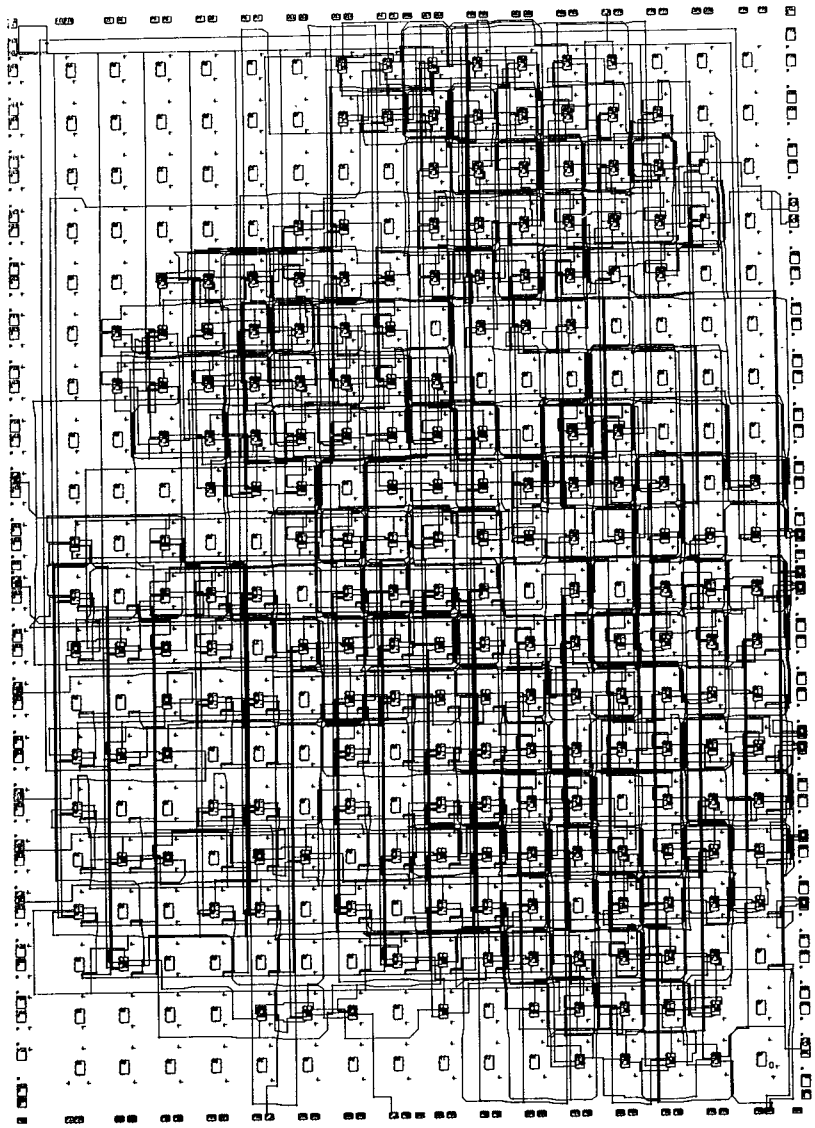


그림 5. CR8000의 배치 및 배선 결과

그림 5는 Viewlogic 툴의 결과에 대한 Xilinx 3090PC84에의 CR8000의 배치 및 배선의 결과이다.

한편 이와같은 VHDL 모델링과 FPGA에의 구현 방법을 이용하면 인터페이스 모듈도 FPGA로의 구현이 가능하므로 완성한 SD8000 8-비트 마이크로 컴퓨터는 그림 6과 같다.

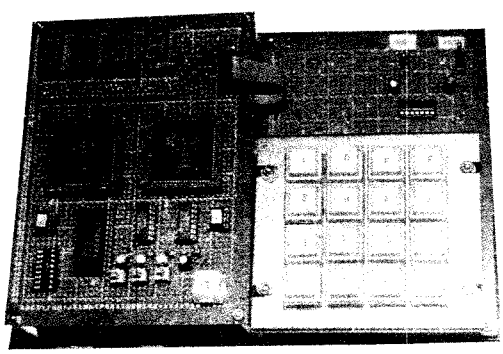


그림 6. SD8000 마이크로 컴퓨터의 완성도

V. 결 론

본 稿에서는 교육용 8-비트 마이크로 컴퓨터의 구

현을 위하여 VHDL 툴을 이용한 논리합성을 행하고 FPGA 칩에의 구현을 행하는 과정을 기술하였다.

VHDL 모델링에 대한 논리합성을 위하여 마이크로프로세서에 대한 동작특성을 RT 레벨에서 분석하고 "Performacne Driven VHDL Modeling"을 위한 design tree를 생성하였다. Design tree의 leaf-node는 PROCESS 문을 이용한 behavior 모델링을 행하고 non-leaf node는 structure 모델링을 행하였으며 2가지 상용 VHDL tool과 1가지 개발중인 툴을 이용하여 Xilinx FPGA를 위한 XNF 데이타를 생성하였다. 생성한 XNF 데이타에 대하여 XACT 툴에 의한 FPGA 칩을 구현하고 마이크로컴퓨터에서의 동작의 정확성을 확인하였다.

이와같이 구현한 CR8000의 1-칩 대신에 다양하게 FPGA, PLD 칩으로 구성된 보드의 설계, 구현에 대한 교육등을 다양하게 진행할 수 있으며 physical design tool을 이용하면 ASIC 실습과 동작의 확인을 행할수도 있어 교육의 질적 향상을 기할 수 있을 것으로 기대된다.

參 考 文 獻

[1] M. Carroll, VHDL - Panacea or Hype?, IEEE Spectrum, PP. 34-37.

[2] 최기영, VHDL 이론, 한국 VHDL 사용자 모임, 1993.

[3] 조중휘, VHDL 응용, 한국 VHDL 사용자 모임, 1993.

[4] KVUG VHDL Workshop, 한국 VHDL 사용자 모임, 1993.

[5] S. Carlson, Introduction to HDL-Based Design Using VHDL, Synopsys, Inc.

[6] Compass ASIC Synthesizer for VHDL Design, 아남 반도체 설계 주식회사.

[7] VHDL Logic Synthesis Training Manual, 한국 엠제엘 주식회사.

[8] MyVHDL Manual, 서두로직 주식회사.

[9] FPGA Design Workshop, ASIC Conference '92 Workshop and Tutorial.

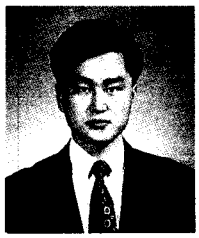
[10] R. L. Ukeiley, Field Programmable Gate Arrays (FPGAs) - The 3000 Series -, Prentice-Hall, 1993.

[11] 박무영, 조중휘, "8051 MCU의 CPU Block에 대한 VHDL 모델링", 1993년도 하계종합학술대회 논문집, PP. 516- 519, 대한전자공학회, 1993.

[12] 박무영, 조중휘, "8051 MCU의 Peripheral Block에 대한 VHDL 모델링", 1993년도 하계종합학술대회 논문집, PP. 512- 515, 대한전자공학회, 1993.

[13] 임재영, 조중휘, "Z80 마이크로프로세서의 VHDL 모델링", 1993년도 하계종합학술대회 논문집, PP. 520- 523, 대한전자공학회, 1993.

筆 者 紹 介



李 昇 禧
1969年 10月 1日生
1993年 2月 인천대학교 전자공학과 졸업

1993年 3月 ~ 현재 인천대학교 전자공학과 석사과정

주관심 분야 : 마이크로프로세서 및 마이크로콘트롤러 설계, ASIC 설계, VHDL 모델링

趙 仲 彙
1957年 5月 16日生
1981年 2月 한양대학교 전자공학과 졸업
1983年 2月 한양대학교 대학원 전자공학과(석사)
1986年 8月 한양대학교 대학원 전자공학과(박사)

1986年 9月 ~ 현재 인천대학교 전자공학과 부교수
1989年 8月 ~ 1990年 7月 UC, Irvine, Post Doc.
1991年 5月 ~ 현재 한국 VHDL 사용자 모임 운영위원

주관심 분야 : 마이크로프로세서 및 마이크로콘트롤러 설계, ASIC 설계, VHDL 모델링