HDL을 이용한 간략형 8-Bit 프로세서의 설계

송호정, 송기용
충북대학교 컴퓨터공학과

Design of a Simple 8-Bit Processor Using HDL

Ho-Jeong Song, Gi-Yong Song
Dept. of Computer Engineering, Chungbuk National University
hjsong@dce3.chungbuk.ac.kr, gysong@chungbuk.ac.kr

요 약

본 논문에서는 HDL을 이용하여 간략형 8-bit 프로세서를 설계하였다. 본 논문에서 설계한 8-bit 프로세서는 3가지의 주소 지정 방법으로 19개의 명령어를 수행하며, 256Kbyte의 메모리와 IR, PC, SP, Y, MA, MD, AC, IN, OUT의 레지스터를 가지고 있다. 설계된 간략형 8-bit 프로세서를 시뮬레이션을 통하여 작동 검증하였고 FPGA 칩상에 합성하였다.

Abstract

In this paper we designed a simple 8-bit processor using HDL. The simple 8-bit processor has 19 instructions with three different addressing modes. The processor includes registers - IR, PC, SP, Y, MA, MD, AC, IN, OUT - and 256Kbyte memory. We examined the operation of the processor through simulation and then synthesized it on FPGA.

I. 서론

전자공학의 급격한 발달로 인해 집적회로에서 컴퓨터분야에 이르는 눈부신 성장을 이룩하게 되었다. 특히 반도체의 발달로 수백만 개의 트랜지스터를 집적할 수 있는 IC, 주문형 반도체(ASIC) 등이 개발되어 그 수요는 날로 증가되고 있는 실정이다.

이렇게 하드웨어가 반도체 기술의 발달로 고집적화, 고속화, 소형화 되어가면서 회로 설계는 기존의 CAD 설계툴에 의한 방식에서 고집적화된 하드웨어를 설계할 수 있는 HDL(Hardware Description Language)을 이용하는 방식으로 전환되고 있다.

본 논문에서는 immediate, direct, register indirect 3가지의 주소 지정 방법으로 19개의 명령어를 수행하는 간략형 8-bit 프로세서를 VHDL로 설계하고 시뮬레이션을 통하여 검증한 후 FPGA 상에 합성하였다.

II. 프로세서의 기본 구성

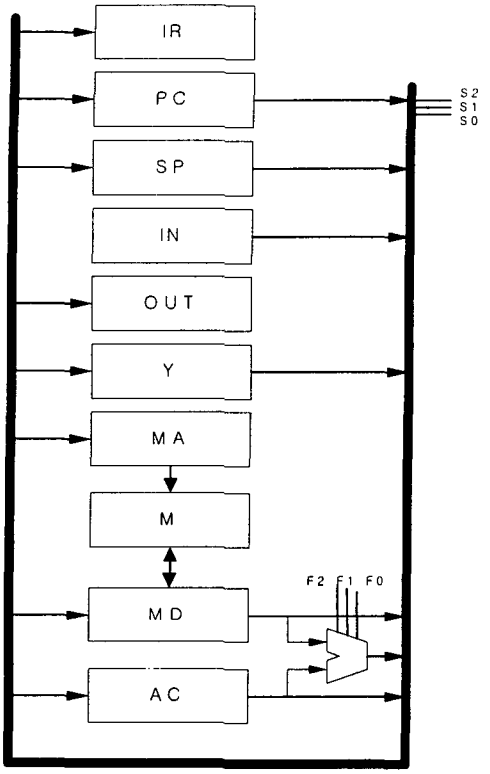
1. 프로세서의 기본 구성

본 논문에서 설계한 간략형 8-bit 프로세서는 256Kbyte의 프로그램과 데이터 메모리 공간과 IR, PC, SP, Y, MA, MD, AC, IN, OUT의 내부 레지스터로 구성된다.

<그림 1>은 본 논문에서 설계한 8-bit 프로세서의 레지스터 연결 구성도이다.

명령어들은 메모리의 프로그램 영역에 연속적으로 저장되어 있다. 명령어를 메모리에서 읽기 위하여 다음과 같은 과정을 밟는다. 먼저 PC는 다음 수행될 명령어가 보관된 메모리 번지를 가지고 있으므로 PC를 MA로 전달한다. 그리고 PC를 MA로 전달한 뒤에 PC는 다음 명령어의 주소 지정을 위하여 1 증가된다. MA에 전달된 번지는 메모리의 주소를 지정하고 해당 주소의 메모리의 데이터가 MD로 전달된다. 만일 MD의

내용이 명령어일 경우 이를 IR 레지스터에 다시 전달하여 IR에 명령어를 보관하게 한다. IR에 보관된 명령어는 제어회로의 명령어 해독기와 타이밍 해독기를 통하여 각 명령어에 따르는 일련의 마이크로 오퍼레이션들로 수행된다.



<그림 1> 간략형 8-bit 프로세서의 구성도

<표 1> 레지스터의 기능

레지스터	기능
IR	현재의 명령어 코드를 저장한다.
PC	다음 수행 될 명령어의 번지를 지정한다.
SP	메모리 스택의 주소를 지정한다.
Y	메모리 데이터의 번지 또는 임시데이터를 저장한다
MA	메모리 주소를 지정한다.
MD	메모리 데이터를 저장한다.
AC	연산 대상이나 결과 또는 입출력 데이터를 저장한다.
IN	외부로부터 데이터를 입력받는 포트
OUT	외부로 데이터를 출력하는 포트

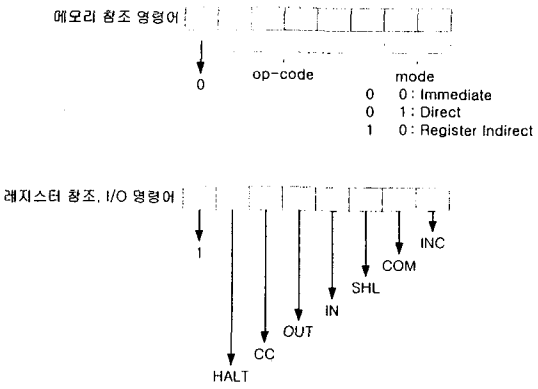
설계한 프로세서에서 사용되는 레지스터의 기능은 <표 1>과 같다.

2. 명령어의 구성

간략형 8-bit 프로세서의 명령어 op-code는 <그림 2>와 같은 구조로 구성되어 있다.

명령어는 크게 메모리 참조 명령과 레지스터 참조, I/O 명령으로 구분되며 이들은 명령어 코드의 최상위 비트로서 구분 된다. 즉 최상위 비트 IR(7)이 '0'이면 메모리 참조 명령어, '1'이면 레지스터 참조, I/O 명령어를 나타낸다.

메모리 참조 명령어는 IR(6-3) 비트로서 각개 명령어를 나타내기 때문에 디코딩 과정이 필요하다. 또한 IR(1-0) 비트로서 immediate, direct, register indirect의 3가지 주소지정 방식을 구분한다.



<그림 2> 명령어의 구성

3. 명령어의 종류

간략형 8-bit 프로세서는 19개의 명령어를 가진다.

메모리 참조 명령어는 4-bit의 코드로 명령어들을 나타내기 때문에 디코딩 과정이 필요하지만, 레지스터 참조, I/O 명령어는 각 비트와 각 명령어의 대응방식을 사용하기 때문에 디코딩 과정이 필요하지 않다.

<표 2>와 <표 3>은 메모리 참조 명령어와 레지스터 참조, I/O 명령어의 규정, 즉 수행 내용과 영향받는 플래그들을 나타낸다.

<표 2> 메모리 참조 명령어

명령어	code	Description
ADD	0000	AC←AC+[M]/imm; CF,ZF
AND	0001	AC←AC · [M]/imm; ZF
OR	0010	AC←AC∨[M]/imm; ZF
CMP	0011	CF←AC-[M]/imm; CF,ZF
LDA	0100	AC←[M]/imm
LDSP	0101	SP←[M]/imm
LDY	0110	Y←[M]/imm
ST	0111	[M]←AC
BR	1000	PC←imm
BC	1001	if CF='1' then PC←imm
BZ	1010	if ZF='1' then PC←imm
BY	1011	if YZF='1' then PC←imm
CALL	1100	[SP]←PC, PC←imm, SP←SP+1
RET	1101	PC←[SP-1]

<표 3> 레지스터 참조, I/O 명령어

명령어	code	Description
INC	0000001	Y←Y+1
COM	0000010	AC←AC'
SHL	0000100	AC←SHL(AC)
IN	0001000	AC←IN
OUT	0010000	OUT←AC
CC	0100000	CF←0
HALT	1000000	Halt

Ⅲ. 프로세서의 설계

1. 레지스터의 종류

각 레지스터는 inc, dec, load, clr의 4가지 기능 중에서 선택적으로 보유하는 기능에 의해 <표 4>와 같이 4개 그룹으로 구분된다.

<표 4> 레지스터 그룹

그룹	레지스터	inc	dec	load	clr
0	IR, AC MA, MD IN, OUT			○	
1	Y	○		○	
2	SP	○	○	○	
3	PC	○		○	○

IR, AC, MA, MD, IN, OUT 레지스터는 데이터를 저장하는 load 기능만을 지니고 있으며, Y 레지스터는 inc, load 기능, SP 레지스터는 inc, dec, load 기능, 그리고 PC 레지스터는 inc, load, clr 기능을 지니고 있다.

2. ALU의 기능

ALU(산술논리 연산 장치)란 데이터프로세서 내에서 산술 연산을 하거나 논리 연산을 하는 장치를 말한다.

중앙처리장치내의 ALU가 산술논리 연산을 수행하기 위해서는 연산이 수행될 데이터가 중앙처리장치내의 레지스터에 존재하여야 한다. ALU의 입력은 AC와 MD로부터 공급되며 연산선택입력 F(2-0)의 설정에 따라서 연산이 선택적으로 이루어지고 연산결과가 발생한다.

<표 5>는 연산선택입력 F에 따른 ALU의 기능을 보여주고 있다.

<표 5> ALU의 기능

F2	F1	F0	연산
0	0	0	a + b
0	0	1	a - b
0	1	0	a ∧ b
0	1	1	a ∨ b
1	0	0	not a
1	0	1	shift left a

3. 내부 버스로의 레지스터 전달

<표 6> 버스입력 S에 따른 레지스터 선택

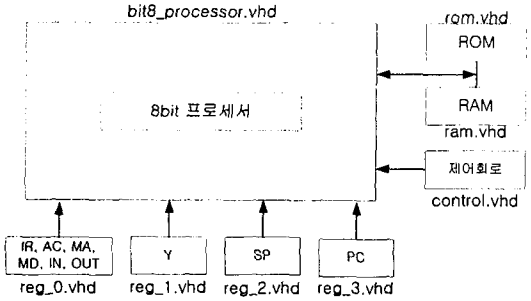
S2	S1	S0	레지스터
0	0	0	PC
0	0	1	SP
0	1	0	IN
0	1	1	Y
1	0	0	MD
1	0	1	AC
1	1	0	ALU

각 레지스터의 데이터들을 필요에 따라 내부 버스로 전달해야 하며, 이러한 기능을 하는 것이 버스입력 S이다. 이 S에 따라서 각각의 레지스터 값들이 내부 버스로 전달이 가능해진다<표 6>.

4. 8-bit 프로세서의 파일 구성

설계된 프로세서는 4개 레지스터 그룹, RAM, ROM, 제어회로 그리고 top level의 8-bit 프로세서 모듈로 구성된다.

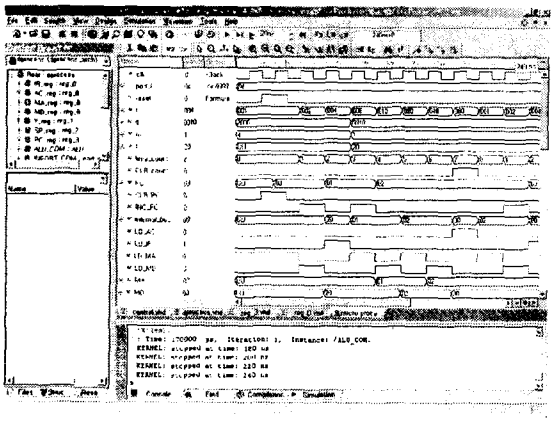
Top level의 8-bit 프로세서에서는 각각의 모듈을 component로 연결하였다 <그림 3>.



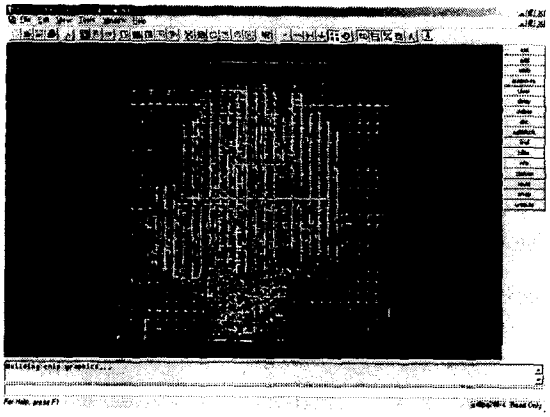
<그림 3> 8-bit 프로세서의 파일 구성도

Ⅳ. 시뮬레이션 및 합성

설계된 프로세서에 Active-HDL로 시뮬레이션을 수행하여 작동을 확인하였으며, <그림 4>, Xilinx Foundation 툴을 사용하여 Xilinx FPGA 상에 설계된 프로세서를 합성하였다<그림 5>.



<그림 4> 설계된 프로세서의 시뮬레이션



<그림 5> 설계된 프로세서의 합성

Ⅴ. 결론

본 논문에서는 immediate, direct, register indirect의 3가지 주소 지정 방식을 가지고 19개의 명령어를 수행하는 간략형 8-bit 프로세서를 VHDL로 설계하고 Active-HDL을 사용하여 시뮬레이션 한 다음 Xilinx FPGA VL-XCS40FP 상에 합성하였다.

참고문헌

[1] Charles H. Roth, Jr., *Digital Systems Design using VHDL*, PWS Publications.

[2] Smith, D.J., *HDL Chip Design*, Doone Publications.

[3] Hohn P.Hayes, *Computer Architecture and Organization*, McGraw-Hill.

[4] M.Morris Mano, *Computer System Architecture*, Prentice-Hall.

[5] Xilinx, 1999 *Xilinx Data Book*.

[6] 박세현, *디지털 컴퓨터 설계와 구현*, 그린