The Design of an i8080A Instruction Compatible Processor with Extended Memory Address

Chiaki Kon

Graduated School of Engineering,
Tokai University
Kitakaname 1117, Hiratsuka,
Kanagawa 259-1292
e-mail: 1aepm024@keyaki.cc.u-tokai.ac.jp

Naohiko Shimizu

Dept. Communications Engineering,
Tokai University
Kitakaname 1117, Hiratsuka,
Kanagawa 259-1292
e-mail: nshimizu@keyaki.cc.u-tokai.ac.jp

Abstract— Computer systems with 8080 or compatible processor and CP/M were most popular from the end 70's to early in 80's. We designed an 8080 compatible processor with MMU and implement it on an FPGA as a CP/M system.
Keyword: SFL, 8080, CP/M

I. Introduction

Intel 8080 was a famous processor. It was used as CPU in old computer systems and the successors of 8080 are still used to control embedded systems in present. We have designed an Intel 8080A instruction compatible processor core "my80" and designed a CP/M machine with it. My80 can access to the extended memory area up to 16MB with newly added instructions, and it can execute all 8080 instructions faster than the original. In this paper, we present the design of my80, and the implementation.

II. Hardware Design

My80 is designed with a description language SFL[3]. The Fig.1 is the block diagram of my80. My80 is a non-pipelined CISC processor. My80 executes all 8080's instruction faster than the original(TABLE I). Fig.2 is the state transition of My80. In if state, my80 fetches an instruction and set it in the OP0 register and transit to f1 state. Depending on the instruction length, f2 and f3 states are activated, respectively. If my80 needs more than 4 clock cycles for the execution of instruction(for example, the instruction needs multiple memory access), the state of my80 is transferred to instruction specific states.

My80 has programmable register set as same as the original i8080A. OP0 - OP2 register is used as internal registers. An ALU unit and an INC/DEC unit ware designed as ripple carry adders to reduce the gate size.

As a disk-less CP/M workstation, we added the following changes to the original i8080A.

A. Address Extension

We added additional 8bit address lines. Hence, My80 has 16MB address space. The extended address space is used as

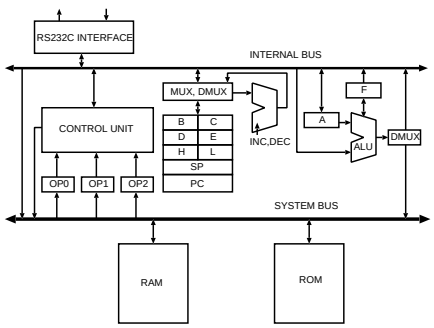

Fig. 1. The block diagram of My80.

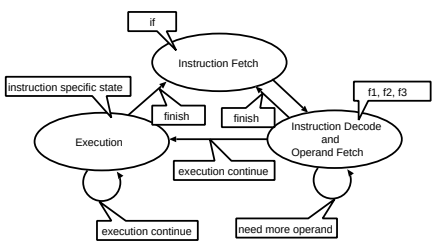

Fig. 2. State Transition

ROM/RAM disks in the CP/M system. My80 has two original instructions to access the extended address space. BIOS parameters configure the geometry of the disks. For example, We decided to use the extended address space as a 128kB disk(Fig.3,4). Boot time bank switching is also a part of the logic. And there is no need for off-chip logic.

B. Additional Instructions

My80 has two memory access mode. One is normal memory access mode, the other can access to extended address space.

In the normal memory access, the extended address lines are intended to be zero(Fig.3). MSB is a RAM/ROM select bit from a programmable status register for the boot up bank switching. If bank switching register value was zero, my80 access to RAM, if it was one, my80 access to ROM. We can alter

TABLE I
MACHINE CYCLES COMPARISON BETWEEN ORIGINAL I8080A AND MY80

instruction	original[2]	my80	instruction	original	my80	instruction	original	my80	instruction	original	my80
MOV(rr)	5	2	XRI	7	3	CP	$11/17^a$	$4/6^a$	XCHG	4	2
MOV(mr, rm)	7	3	ORI	7	3	CM	$11/17^a$	$4/6^a$	XTHL	18	5
HLT	7	2	CPI	7	3	CPE	$11/17^a$	$4/6^a$	SPHL	5	2
MVI(r, m)	7, 10	3	RLC	4	2	CPO	$11/17^a$	$4/6^a$	PCHL	5	3
INR(r)	5	2	RRC	4	2	RET	10	3	DAD	10	3
DCR(r)	5	2	RAL	4	2	RC	$5/11^a$	$2/3^a$	STAX	7	2
INR(m)	10	3	RAR	4	2	RNC	$5/11^a$	$2/3^a$	LDAX	7	2
DCR(m)	10	3	JMP	10	4	RZ	$5/11^a$	$2/3^a$	INX	5	3
ADD(r, m)	4, 7	3	JC	10	4	RNZ	$5/11^a$	$2/3^a$	DCX	5	3
ADC(r, m)	4, 7	3	JNC	10	4	RP	$5/11^a$	$2/3^a$	CMA	4	2
SUB(r, m)	4, 7	3	JZ	10	4	RM	$5/11^a$	$2/3^a$	STC	4	2
SBB(r, m)	4, 7	3	JNZ	10	4	RPE	$5/11^a$	$2/3^a$	CMC	4	2
ANA(r, m)	4, 7	3	JP	10	4	RPO	$5/11^a$	$2/3^a$	DAA	4	3
XRA(r, m)	4, 7	3	JM	10	4	RST	11	4	SHLD	16	5
ORA(r, m)	4, 7	3	JPE	10	4	IN	10	3	LHLD	16	5
CMP(r, m)	4, 7	3	JPO	10	4	OUT	10	3	EI	4	2
ADI	7	3	CALL	17	6	LXI	10	4	DI	4	2
ACI	7	3	CC	$11/17^a$	$4/6^a$	PUSH	11	4	NOP	4	2
SUI	7	3	CNC	$11/17^a$	$4/6^a$	POP	10	3	STAB		3
SBI	7	3	CZ	$11/17^a$	$4/6^a$	STA	13	4	LDAB		3
ANI	7	3	CNZ	$11/17^a$	$4/6^a$	LDA	13	4			

^{a} False/Success

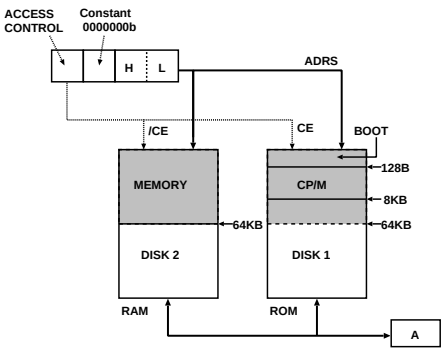

Fig. 3. Normal Memory Access Mode

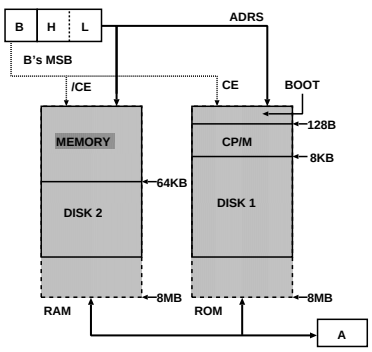

Fig. 4. Extended Address Memory Access Mode

TABLE II
SYNTHESIZE RESULTS

Device	Maximum Frequency	Used LC or Gates
EPF10K30AQC240-3^a	8.90 MHz	1260 LCs
EPF1S80F1508C6^a	57.21 MHz	1240 LCs
NEC CMOS9^b	34.36 MHz	7826 Gates

^{a} Synthesized by Max+plus2
^{b} Synthesized by PARTHENON

bank switching register with out instruction.

We added two instructions, STAB and LDAB. These Instructions are used to access for the extended address space. STAB and LDAB works as a store instruction and a load instruction, respectively. The address is designated with the combination of the B, H and L registers, as shown in Fig.4.

The op-codes are selected from the unused codes of the Z80(DD02, DD03). Hence, it is easy to port these instruction for Z80 compatible processors.

III. IMPLEMENTATION

We implemented my80 as a CP/M machine. We used CSP-011-30A[5] for MPU and an original PCB board for RAM/ROM.

The results of logic synthesis by Parthenon[3], Max+plus2 and Quartus2[6] are shown in the TABLE II. We used an EPF10K30AQC240-3 device for our demo board. Target module contains My80 core, SIO interface and required address decoders. Then we do not need any glue logic. We used 2.304 MHz clock due to the EPROM access speed.

IV. CONCLUSION

We designed an 8080A compatible system with extended address space running as a disk-less CP/M machine. Because it uses only a small amount of gates, we think that it is possible to use my80 as an embedded controller in SOC design, and/or educations.

REFERENCES

[1] Homepage: http://shimizu-lab.et.u-tokai.ac.jp/
[2] MCS-86 User's Manual (1979) Intel
[3] Parthenon: http://www.kecl.ntt.co.jp/parthenon/
[4] Retro Archive: http://www.retroarchive.org
[5] Humandata: http://www.hdl.co.jp
[6] Altera: http://www.altera.com