实验6控制器的设计和验证

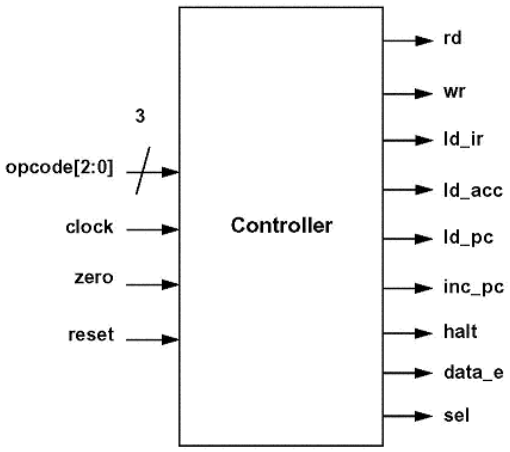

图 1 控制器端口信息

实验说明：

1． 控制器的设计规则。

该控制器与时钟的上升沿同步产生8个节拍，这8个节拍构成一个指令周期。每条指令的取指和执行在8个节拍内完成，前半个指令周期进行取指操作，后半个指令周期指令执行操作。

2.操作码与指令的对应关系如下：

CPU Instruction Set

INSTRUCTION		OPCODE
HLT	(halt)	000
SKZ	(skip if zero bit set)	001
ADD	(data + accumulator)	010
AND	(data & accumulator)	011
XOR	(data ^ accumulator)	100
LDA	(load accumulator)	101
STO	(store accumulator)	110
JMP	(jump to new address)	111

图2 指令与操作码对应关系

3.在每一个节拍，控制器都会输出9个控制信号，这些控制信号是由指令操作码和ALU的零标志位（zero）产生的。控制器采用低电平的异步复位，当reset信号置0时，所有的输出被清零，节拍也重新开始计数。

输出信号的部分时序信息如下图所示。

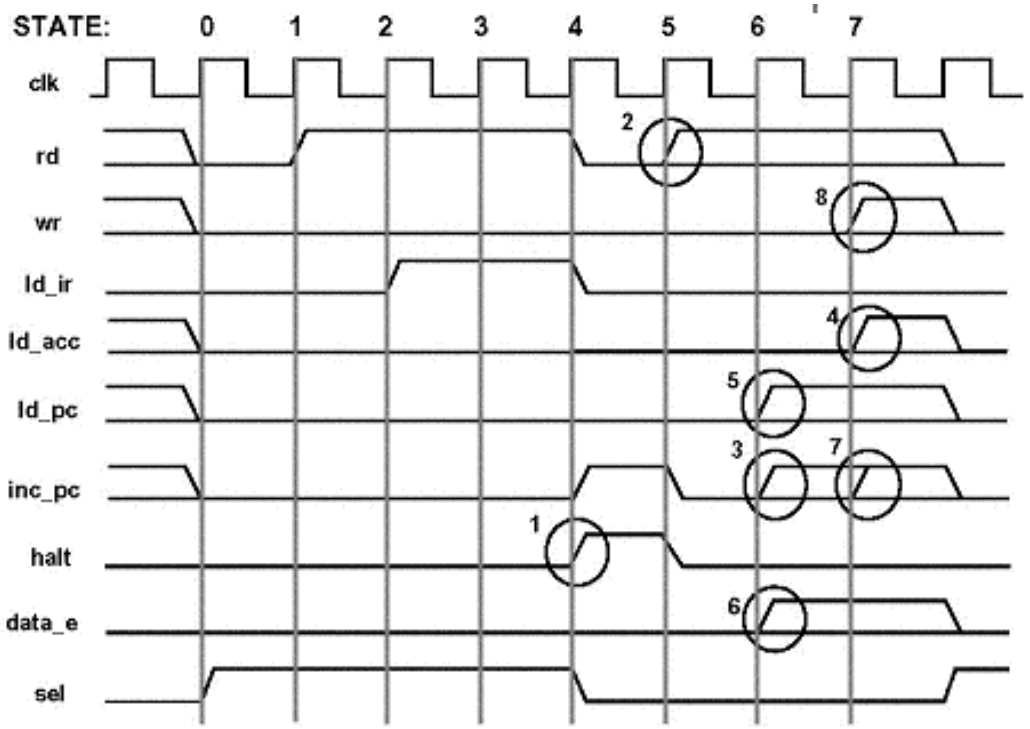

图3 输出时序信息

注意事项：

1）执行HLT 指令时，halt 信号被置1；

2）执行ADD、AND、XOR、LDA 操作时，rd 信号置1；

3）执行SKZ 指令且零标志位被置1 时，inc_pc 置1；

4）执行ADD、AND、XOR、LDA 指令时，ld_acc 置1；

5）指令是JMP 时，ld_pc 置1；

6）如果操作码不是ADD、AND、XOR、LDA，data_e 置1；

7）指令为JMP，inc_pc 置1；

8）指令为STO,wr 信号置1。

下面的图可以更直观的看出控制信号是如何产生的。

rst	state	rd	wr	ld_ir	ld_acc	ld_pc	inc_pc	halt	data_e	sel
0	?: Reset	0	0	0	0	0	0	0	0	0
1	0: Address Setup 1	0	0	0	0	0	0	0	0	1
1	1: Instruction Fetch	1	0	0	0	0	0	0	0	1
1	2: Instruction Load	1	0	1	0	0	0	0	0	1
1	3: Idle	1	0	1	0	0	0	0	0	1
1	4: Address Setup 2									
	if HLT opcode	0	0	0	0	0	1	1	0	0
	all other opcodes	0	0	0	0	0	1	0	0	0
1	5: Operand Fetch									
	ADD, AND, XOR, LDA	1	0	0	0	0	0	0	0	0
	all other opcodes	0	0	0	0	0	0	0	0	0
1	6: ALU Operation									
	SKZ **and** zero set	0	0	0	0	0	1	0	1	0
	ADD, AND, XOR, LDA	1	0	0	0	0	0	0	0	0
	JMP	0	0	0	0	1	0	0	1	0
	all other opcodes	0	0	0	0	0	0	0	1	0
1	7: Store Result									
	SKZ **and** zero set	0	0	0	0	0	1	0	1	0
	ADD, AND, XOR, LDA	1	0	0	1	0	0	0	0	0
	STO	0	1	0	0	0	0	0	1	0
	JMP	0	0	0	0	1	1	0	1	0
	all other opcodes	0	0	0	0	0	0	0	1	0

图4 控制信号生成

控制器可以使用状态机来实现。