19302010011 王海伟

part1：3 - 8译码器 decoder 设计

实验内容：

根据教材4 - 8节，设计3 - 8译码器

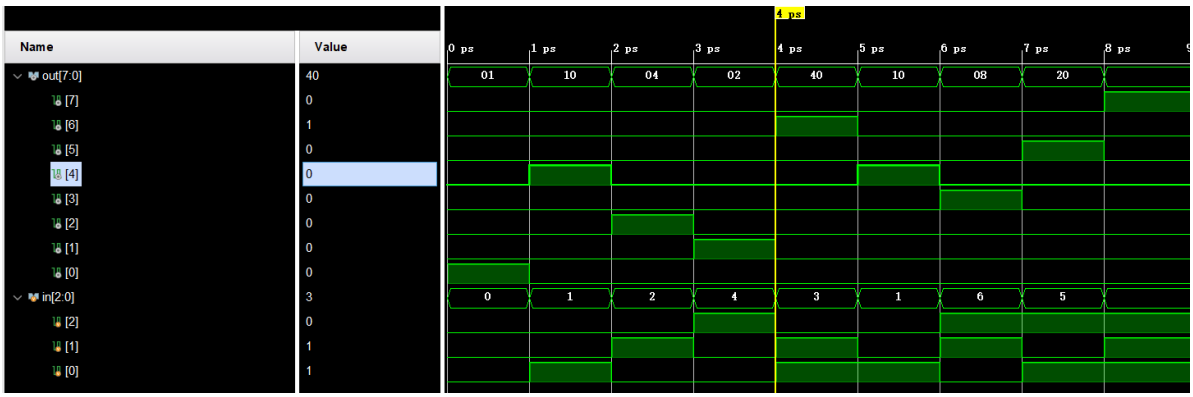
- 1．写出3 - 8译码器真值表，通过化简写出译码器布尔表达式
- 2．使用Verilog HDL实现

输入使用板上的switch拨段开关，输出使用板上的led灯。

输入				**输出**							
x	y	z		D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0		1	0	0	0	0	0	0	0
0	0	1		0	1	0	0	0	0	0	0
0	1	0		0	0	1	0	0	0	0	0
0	1	1		0	0	0	1	0	0	0	0
1	0	0		0	0	0	0	1	0	0	0
1	0	1		0	0	0	0	0	1	0	0
1	1	0		0	0	0	0	0	0	1	0
1	1	1		0	0	0	0	0	0	0	1

$D_0 = x^{'}y^{'}z^{'}\ D_1 = x^{'}y^{'}z\ D_2 = x^{'}yz^{'}\ D_3 = x^{'}yz\ D_4 = xy^{'}z^{'}\ D_5 = xy^{'}z\ D_6 = xyz^{'}$
$D_7 = xyz$

波形图如下：



part2：4 - 2编码器 encoder 设计

实验内容：

根据教材4 - 9节，设计**4 - 2**编码器

- 1．写出**4 - 2**编码器真值表，通过化简写出编码器布尔表达式
- 2．使用Verilog HDL实现

输入使用板上的switch拨段开关，输出使用板上的led灯。

输出			**输入**			
x	y		D_0	D_1	D_2	D_3
0	0		1	0	0	0
0	1		0	1	0	0
1	0		0	0	1	0
1	1		0	0	0	1

$x = D_2 + D_3 \quad y = D_3 + D_1 D_2^{'}$

波形图如下：

part3：mips control unit设计

实验内容：如果需要支持MIP指令集的如下13条指令，｛add，sub，and，or，slt，addi，andi，ori，slti，sw，lw，j，nop｝，MIPS处理器中的Control Unit设计。

ALU三位输入的作用

$F_{2:0}$	Function
000	A&B
001	A|B
010	A+B
011	not used
100	A&~B
101	A|~B
110	A-B
111	SLT

ALUop，meaning and ALUcontrol output

$ALUOP_{1:0}$	Meaning
00	ADD
01	subtract
10	funct
11	not used->sll

$ALUOP_{1:0}$	Funct	$ALUControl_{2:0}$
00	x	010(add)
x1	x	110(sub)
1x	100000(add)	010(add)
1x	100010(sub)	110(sub)
1x	100100(and)	000(and)
1x	100101(or)	001(or)
1x	101010(slt)	111(slt)

到这边alu decoder的逻辑已经好了

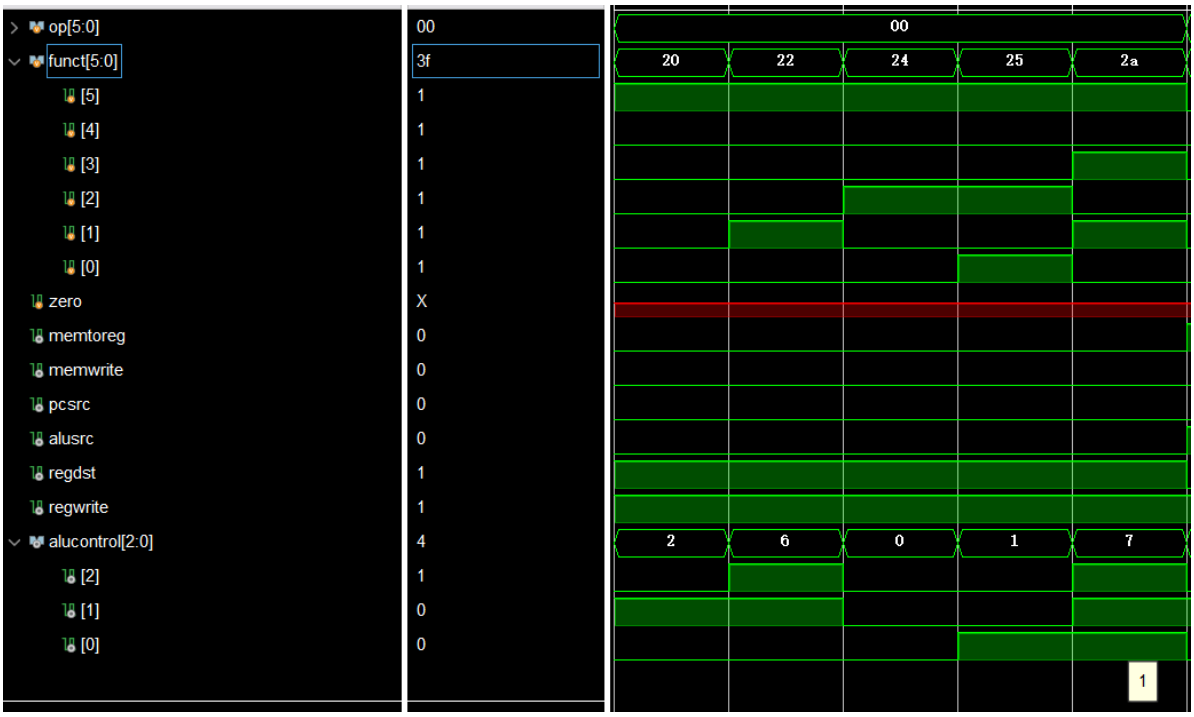
然后就是mainDecoder

Instruction	op	RegWrite	RegDst	AluSrc	Branch	MenWrite	MenToReg	ALUOp	Jump
R-type	000000	1	1	0	0	0	0	10	0
lw	100011	1	0	1	0	0	1	00	0
sw	101011	0	x	1	0	1	x	00	0
addi	001000	1	0	1	0	0	0	00	0
beq	000100	0	x	0	1	0	x	01	0
j	000010	0	x	x	x	0	x	xx	1
ori	001101	1	0	1	0	0	0	10	0
andi	000110	1	0	1	0	0	0	10	0
slti	001010	1	0	1	0	0	0	10	0
nop	000000	1	1	0	0	0	0	10	0

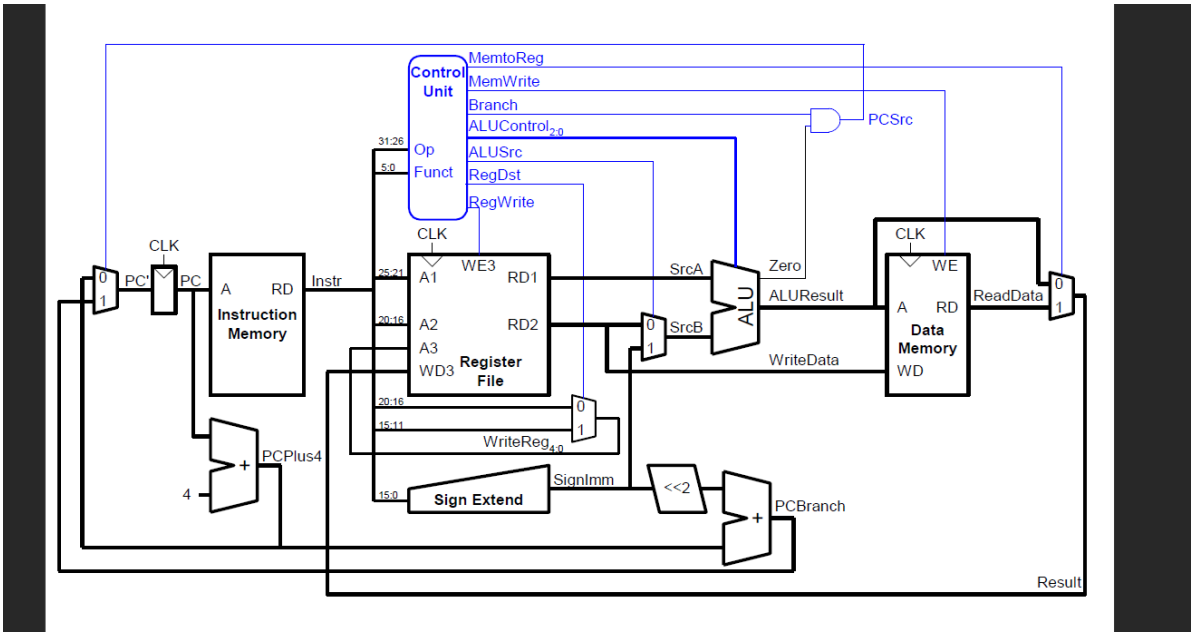
这边比较特殊的是nop，根据查到的资料，nop=sll $0,$0,0,但是我找不到当前结构下sll的信息，无奈之下我把sll指定了一个funct code来实现nop。

波形图：

rtype：



其他（包括nop）：



这边比较需要思考的就是control unit这么多输出的原因是什么，其实直接看架构图的话比较清晰，每一个输出值都代表了这个值所代表的wire是否连通，或者充当了一个选择器的作用。举个例子，对于add，首先它是r-type（所有的rtype除了alucontrol都一样），它需要寄存器写，RegWrite为1，RegDst需要选择15：11的bit位置，为1，alusrc为0，使数据来自寄存器，ALUcontrol由alu译码器产生，branch不需要，为0，MemWrite和MemtoReg也不需要要，都为0。实际上这个过程就是在分析这条指令执行是的datapath。