实验1 单周期CPU仿真

姓名： 江书洋
学号： 519030910043

实验目的

1. 采用Verilog硬件描述语言在 EDA 设计平台 Quartus Prime 上，基于 Intel cyclone IV 系列 FPGA 完成具有执行 20 条 MIPS 基本指令的单周期 CPU 模块的设计。根据提供的单周 期 CPU 示例程序的 Verilog 代码文件，将设计代码补充完整，实现该模块的电路设计。
2. 利用实验提供的标准测试程序代码，完成单周期 CPU 模块的功能仿真测试，验证 CPU 执行所设计的 20 条 RISC 指令功能的正确性。
3. 理解计算机五大组成部分的协调工作原理，理解存储程序自动执行的原理和掌握运 算器、存储器、控制器的设计和实现原理。

实验流程

1. 根据所学知识填补`Real_Value_Table_to student.xls`，从而使用该文件完成`sc_cu.v`文件
2. 根据逻辑填充`sc_cu.v`和`alu.v`文件。
3. 使用给出的波形仿真文件进行仿真得到波形图

excel文件填写思路

R型指令

1. 由于下一条指令的执行都是`PC + 4`，因此`pcsource`一列全部为`0 0`。
2. 查找教科书，得到`aluc`各项值。
3. 根据逻辑，只有`sll, srl, sra`三条指令有移位操作，因此只有这三条指令的`shift`为1。
4. R型指令都没有进行立即数计算，因此`aluimm`列都为0。
5. R型指令的符号扩展并没有影响，所有`sext`列为x。
6. 根据R型指令的特性，都有写寄存器操作，都没有写内存操作，因此`wmem`为0，`wreg`为1。
7. R型指令都没有从内存写到寄存器、写到`rt`寄存器、调用`jal`的操作，因此`m2reg, regrt, call jal`三列都为0。

I型指令

1. 当`beq`的比较结果为1时，零标志位置为1，此时下一条指令从`PC + 4 + imm * 4`位置处执行。`bne`操作刚好相反。通过以上的逻辑填充`z`字段以及`pcsource`字段，其余指令的`pcsource`和R型指令类似，都为`0 0`。
2. 查找教科书，得到`aluc`各项值。其中`beq`和`bne`命令我是使用异或的aluc进行填写，实际上使用`sub`指令的aluc也是可以的。
3. 没有移位操作，`shift`均为0。
4. 除了`beq, bne`操作是用立即数进行PC运算，其余都是进行alu运算，因此除了这两条指令的`aluimm`为0以外，其他指令的该字段都为1。
5. 根据各项指令的内部逻辑，填写`sext`字段。
6. 只有`sw`操作有写内存操作，它的`wmem`为1，其余的指令都为0。alu型的I型指令都有写寄存器操作，`lw, lui`也写寄存器，因此除这几条指令的`m2reg`为1以外其他指令的该字段为0。

7. 根据各条指令的特性，填写`regrt`字段。由于只有`jal`一条指令，因此只有它的`call jal`字段为1，其余的都为0或者x。

填写完成的excel表展示

输入

指令	指令格式	op	rs	rt	rd	sa	func	z	pcsource [1..0]	aluc [3..0]	shift	aluimm	sext	wmem	wreg	m2reg	regrt	call jal
add	add rd, rs, rt	000000	rs	rt	rd	00000	100000	x	0 0	0 0 0 0	0	0	x	0	1	0	0	0
sub	sub rd, rs, rt	000000	rs	rt	rd	00000	100010	x	0 0	x 1 0 0	0	0	x	0	1	0	0	0
and	and rd, rs, rt	000000	rs	rt	rd	00000	100100	x	0 0	x 0 0 1	0	0	x	0	1	0	0	0
or	or rd, rs, rt	000000	rs	rt	rd	00000	100101	x	0 0	x 1 0 1	0	0	x	0	1	0	0	0
xor	xor rd, rs, rt	000000	rs	rt	rd	00000	100110	x	0 0	x 0 1 0	0	0	x	0	1	0	0	0
sll	sll rd, rt, sa	000000	00000	rt	rd	sa	000000	x	0 0	0 0 1 1	1	0	x	0	1	0	0	0
srl	srl rd, rt, sa	000000	00000	rt	rd	sa	000010	x	0 0	0 1 1 1	1	0	x	0	1	0	0	0
sra	sra rd, rt, sa	000000	00000	rt	rd	sa	000011	x	0 0	1 1 1 1	1	0	x	0	1	0	0	0
jr	jr rs	000000	rs	00000	00000	00000	001000	x	1 0	x x x x	x	x	x	0	0	x	x	x

指令	指令格式	op	rs	rt	rd	sa	func		pcsource [1..0]	aluc [3..0]	shift	aluimm	sext	wmem	wreg	m2reg	regrt	call jal
addi	addi rt, rs, imm	001000	rs	rt	imm				0 0	0 0 0 0	0	1	1	0	1	0	1	0
andi	andi rt, rs, imm	001100	rs	rt	imm				0 0	x 0 0 1	0	1	0	0	1	0	1	0
ori	ori rt, rs, imm	001101	rs	rt	imm				0 0	x 1 0 1	0	1	0	0	1	0	1	0
xori	xori rt, rs, imm	001110	rs	rt	imm				0 0	x 0 1 0	0	1	0	0	1	0	1	0
lw	lw rt, imm(rs)	100011	rs	rt	imm				0 0	x x x x	0	1	1	0	1	1	1	0
sw	sw rt, imm(rs)	101011	rs	rt	imm				0 0	x x x x	0	1	1	1	0	0	0	0
beq	beq rs, rt, imm	000100	rs	rt	imm			0	0 0	x 0 1 0	0	0	1	0	0	0	0	0
								1	0 1									
bne	bne rs, rt, imm	000101	rs	rt	imm			0	0 1	x 0 1 0	0	0	1	0	0	0	0	0
								1	0 0									
lui	lui rt, imm	001111	00000	rt	imm				0 0	x 1 1 0	0	1	x	0	1	0	1	0
j	j addr	000010	addr						1 1	x x x x	x	x	x	0	0	x	x	x
jal	jal addr	000011	addr						1 1	x x x x	x	x	x	0	1	x	x	1

填写完成的alu.v代码

```
module alu (a,b,aluc,s,z);
   input [31:0] a,b;
   input [3:0] aluc;
   output [31:0] s;
   output        z;
   reg [31:0] s;  // alu的output
   reg        z;  // zero flag 如果输出的s为0那么flag置为1,反之置为0
   always @ (a or b or aluc)
      begin                                    // event
         casex (aluc)
             4'bx000: s = a + b;               //x000 ADD
             4'bx100: s = a - b;               //x100 SUB
             4'bx001: s = a & b;               //x001 AND
             4'bx101: s = a | b;               //x101 OR
             4'bx010: s = a ^ b;               //x010 XOR
             4'bx110: s = b << 16;             //x110 LUI: imm << 16bit
             4'b0011: s = b << a;             //0011 SLL: rd <- (rt << sa)
             4'b0111: s = b >> a;             //0111 SRL: rd <- (rt >> sa)
(logical)
             4'b1111: s = $signed(b) >>> a;    //1111 SRA: rd <- (rt >> sa)
(arithmetic)
             default: s = 0;
         endcase
```

```
          if (s == 0 )  z = 1;
             else z = 0;
       end
endmodule
```

这里的逻辑右移左移都不需要加$unsigned()进行转换，这是由于逻辑右移的定义决定的，如果一个有符号的数进行逻辑右移，那么会在开始处补0， 如下所示：
10111001 -> 01011100（右移一位） 因此逻辑右移有可能改变原值的符号，不需要加入$unsigned()对b进行类型转换。

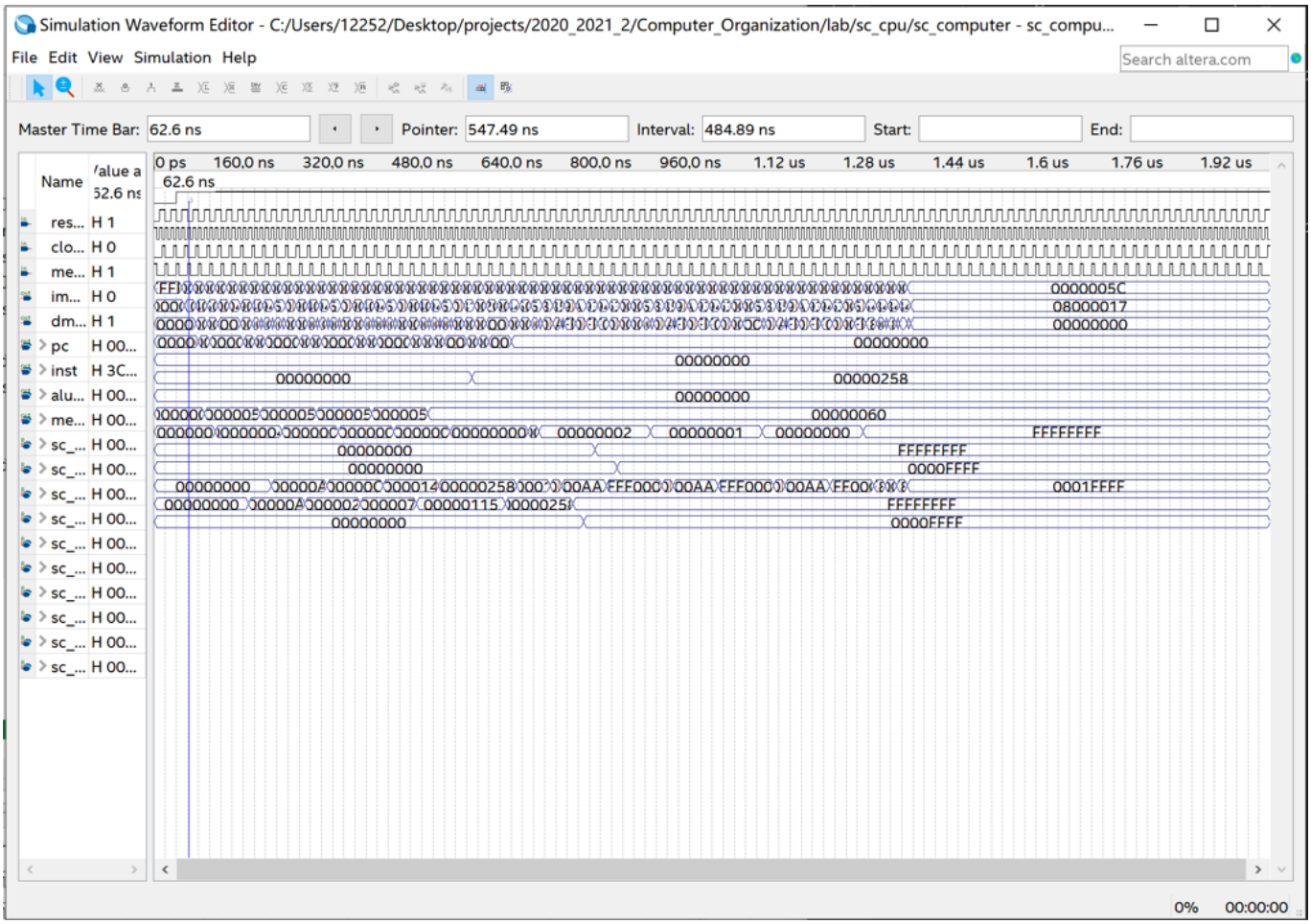
填写完成的sc_cu.v代码

```
module sc_cu (op, func, z, wmem, wreg, regrt, m2reg, aluc, shift,
              aluimm, pcsource, jal, sext);
   input  [5:0] op,func;
   input        z;
   output       wreg,regrt,jal,m2reg,shift,aluimm,sext,wmem;
   output [3:0] aluc;
   output [1:0] pcsource;
   wire r_type = ~|op;
   wire i_add = r_type & func[5] & ~func[4] & ~func[3] &
                ~func[2] & ~func[1] & ~func[0];          //100000
   wire i_sub = r_type & func[5] & ~func[4] & ~func[3] &
                ~func[2] &  func[1] & ~func[0];          //100010

   //  please complete the deleted code.

   wire i_and = r_type & func[5] & ~func[4] & ~func[3] &
                         func[2] & ~func[1] & ~func[0]; // 100100

   wire i_or  = r_type & func[5] & ~func[4] & ~func[3] &
                         func[2] & ~func[1] &  func[0]; // 100101

   wire i_xor = r_type & func[5] & ~func[4] & ~func[3] &
                         func[2] & func[1] & ~func[0];  // 100110

   wire i_sll = r_type & ~func[5] & ~func[4] & ~func[3] &
                         ~func[2] & ~func[1] & ~func[0]; // 000000

   wire i_srl = r_type & ~func[5] & ~func[4] & ~func[3] &
                         ~func[2] &  func[1] & ~func[0]; // 000010

   wire i_sra = r_type & ~func[5] & ~func[4] & ~func[3] &
                         ~func[2] &  func[1] &  func[0]; // 000011

   wire i_jr  = r_type & ~func[5] & ~func[4] &  func[3] &
                         ~func[2] & ~func[1] & ~func[0]; // 001000

   wire i_addi = ~op[5] & ~op[4] &  op[3] & ~op[2] & ~op[1] & ~op[0]; //001000
   wire i_andi = ~op[5] & ~op[4] &  op[3] &  op[2] & ~op[1] & ~op[0]; //001100
```

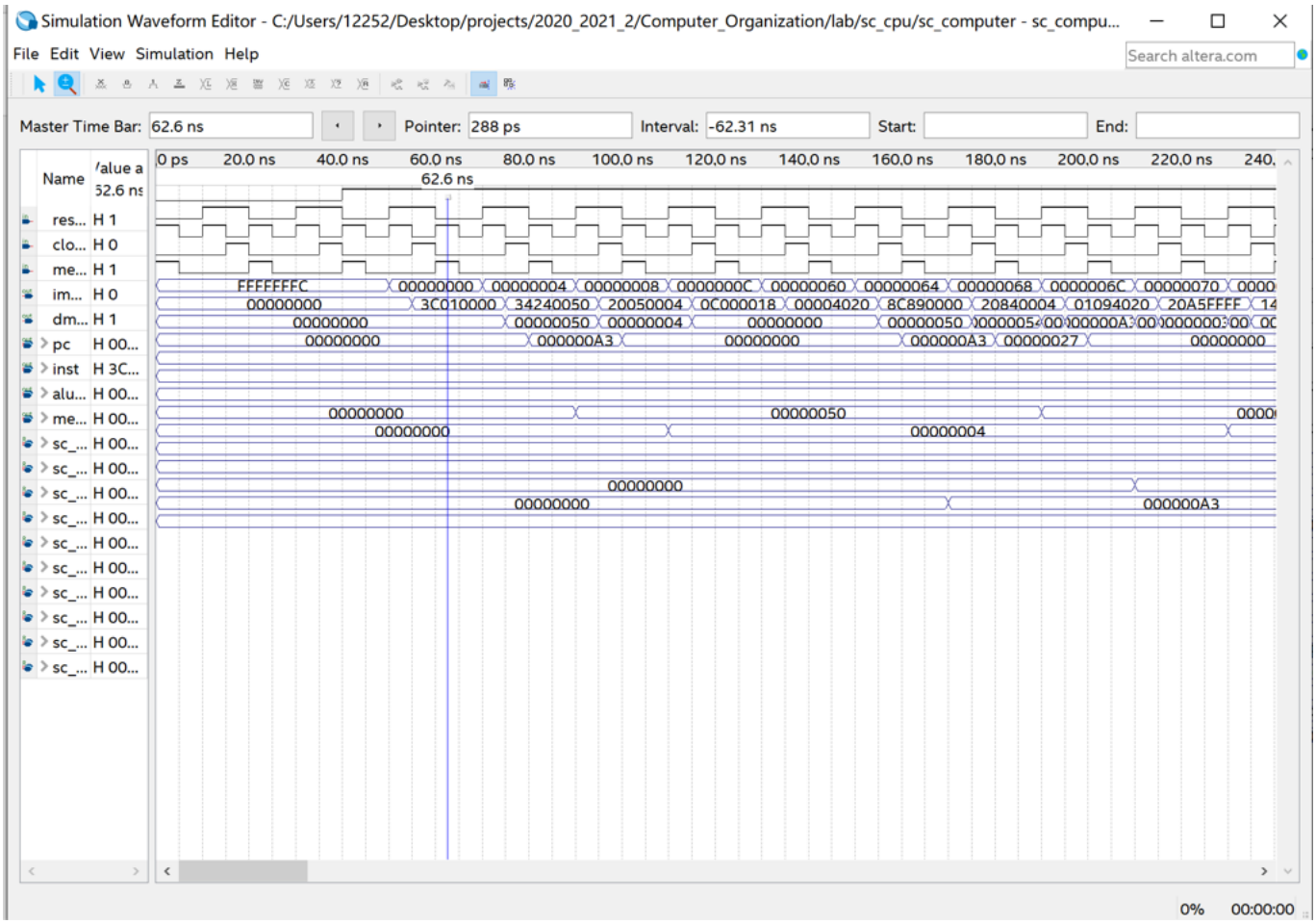
```
    wire i_ori  = ~op[5] & ~op[4] &  op[3] &  op[2] & ~op[1] &  op[0]; // 001101
    wire i_xori = ~op[5] & ~op[4] &  op[3] &  op[2] &  op[1] & ~op[0]; // 001110
    wire i_lw   =  op[5] & ~op[4] & ~op[3] & ~op[2] &  op[1] &  op[0]; // 100011
    wire i_sw   =  op[5] & ~op[4] &  op[3] & ~op[2] &  op[1] &  op[0]; // 101011
    wire i_beq  = ~op[5] & ~op[4] & ~op[3] &  op[2] & ~op[1] & ~op[0]; // 000100
    wire i_bne  = ~op[5] & ~op[4] & ~op[3] &  op[2] & ~op[1] &  op[0]; // 000101
    wire i_lui  = ~op[5] & ~op[4] &  op[3] &  op[2] &  op[1] &  op[0]; // 001111
    wire i_j    = ~op[5] & ~op[4] & ~op[3] & ~op[2] &  op[1] & ~op[0]; // 000010
    wire i_jal  = ~op[5] & ~op[4] & ~op[3] & ~op[2] &  op[1] &  op[0]; // 000011


    assign pcsource[1] = i_jr | i_j | i_jal;
    assign pcsource[0] = ( i_beq & z ) | (i_bne & ~z) | i_j | i_jal ;

    assign wreg = i_add | i_sub | i_and | i_or   | i_xor  |
                  i_sll | i_srl | i_sra | i_addi | i_andi |
                  i_ori | i_xori | i_lw | i_lui  | i_jal;

    assign aluc[3] = i_sra;
    assign aluc[2] = i_sub | i_or | i_srl | i_sra | i_ori | i_lui;
    assign aluc[1] = i_xor | i_sll | i_srl | i_sra | i_xori | i_beq | i_bne |
i_lui;
    assign aluc[0] = i_and | i_or | i_sll | i_srl | i_sra | i_andi | i_ori;
    assign shift   = i_sll | i_srl | i_sra ;

    assign aluimm  = i_addi | i_andi | i_ori | i_xori | i_lw | i_sw | i_lui;
    assign sext    = i_addi | i_lw | i_sw | i_beq | i_bne;
    assign wmem    = i_sw;
    assign m2reg   = i_lw;
    assign regrt   = i_addi | i_andi | i_ori | i_xori | i_lw | i_lui;
    assign jal     = i_jal;

endmodule
```

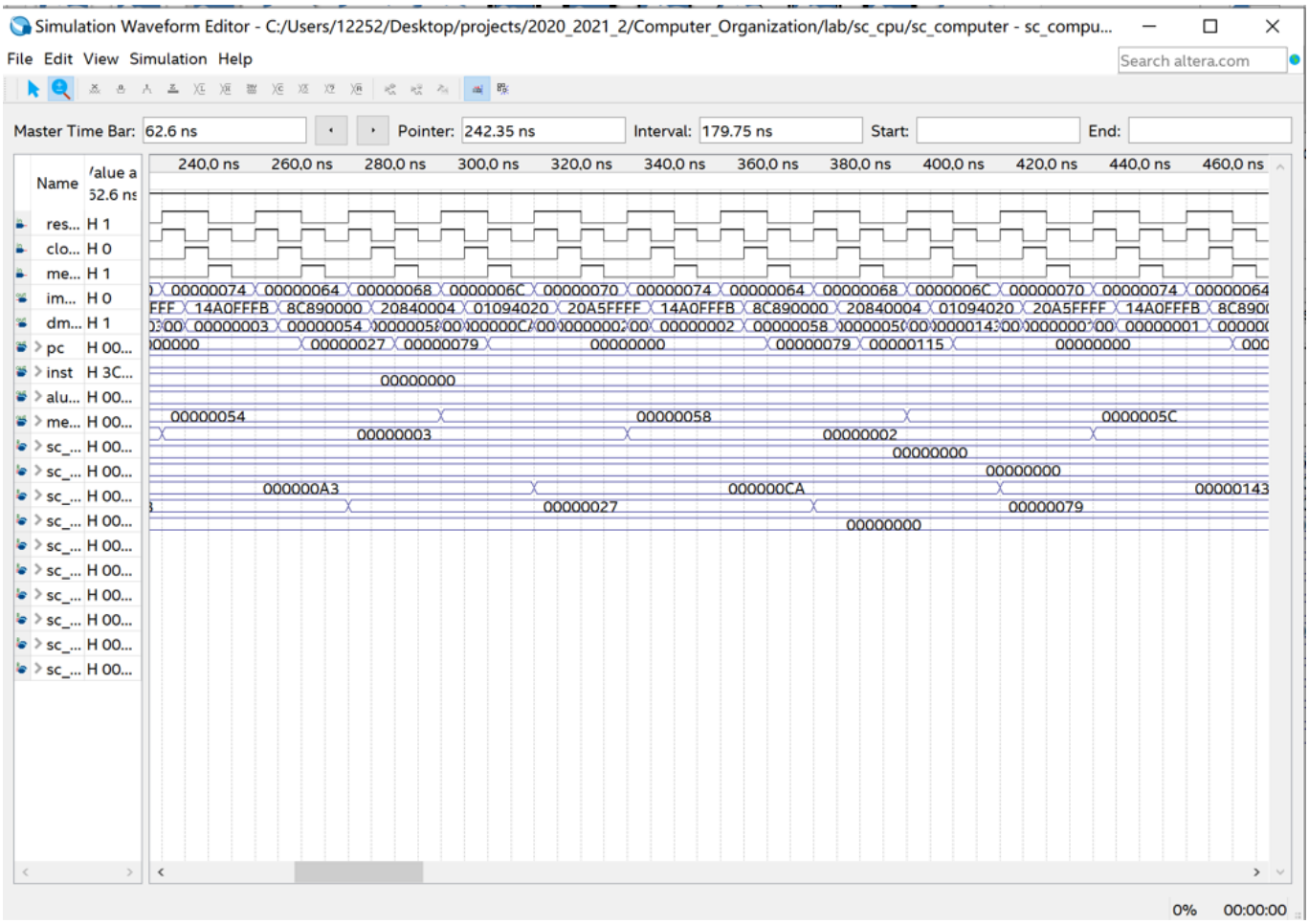
实验结果

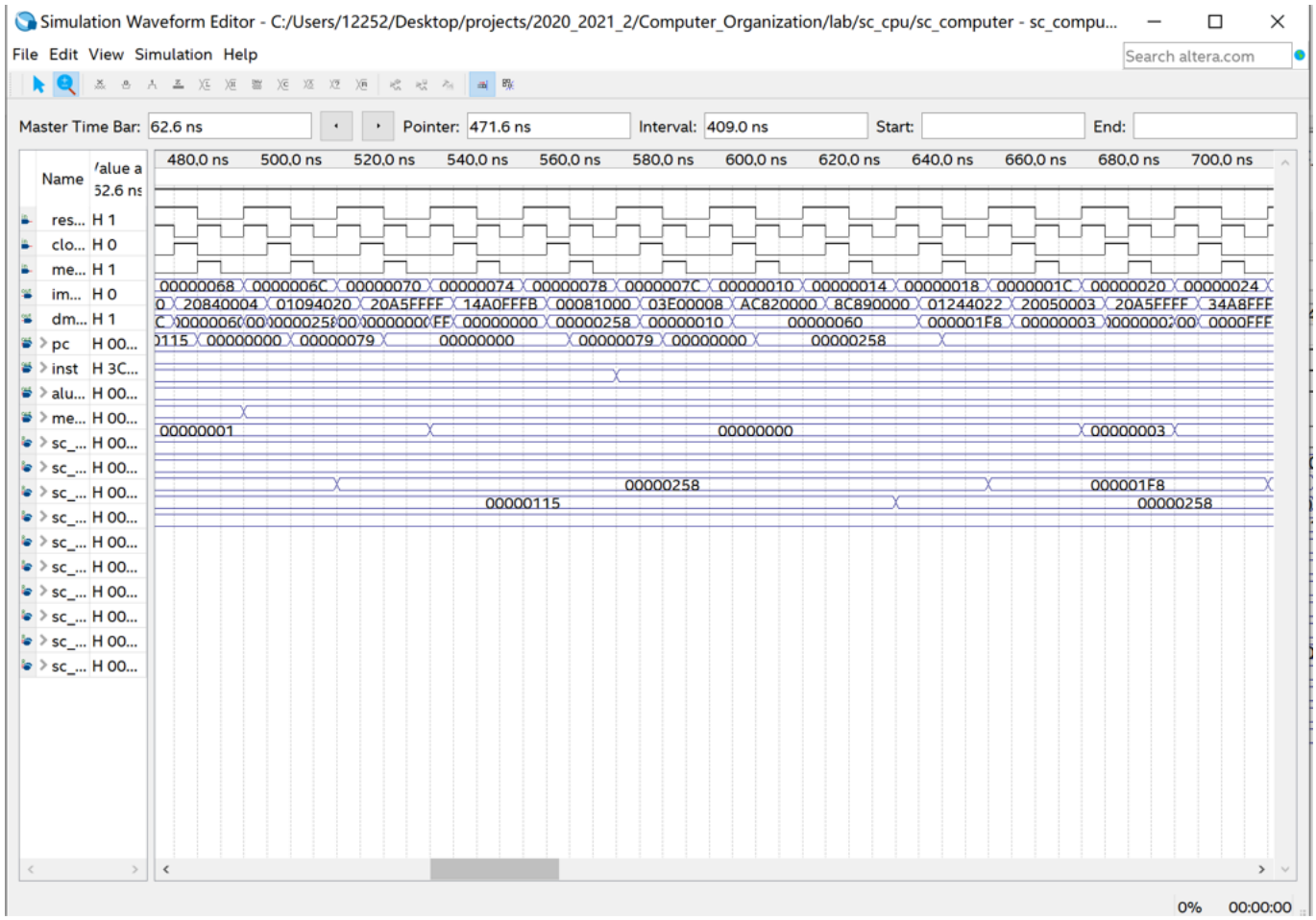
整体的波形图一览

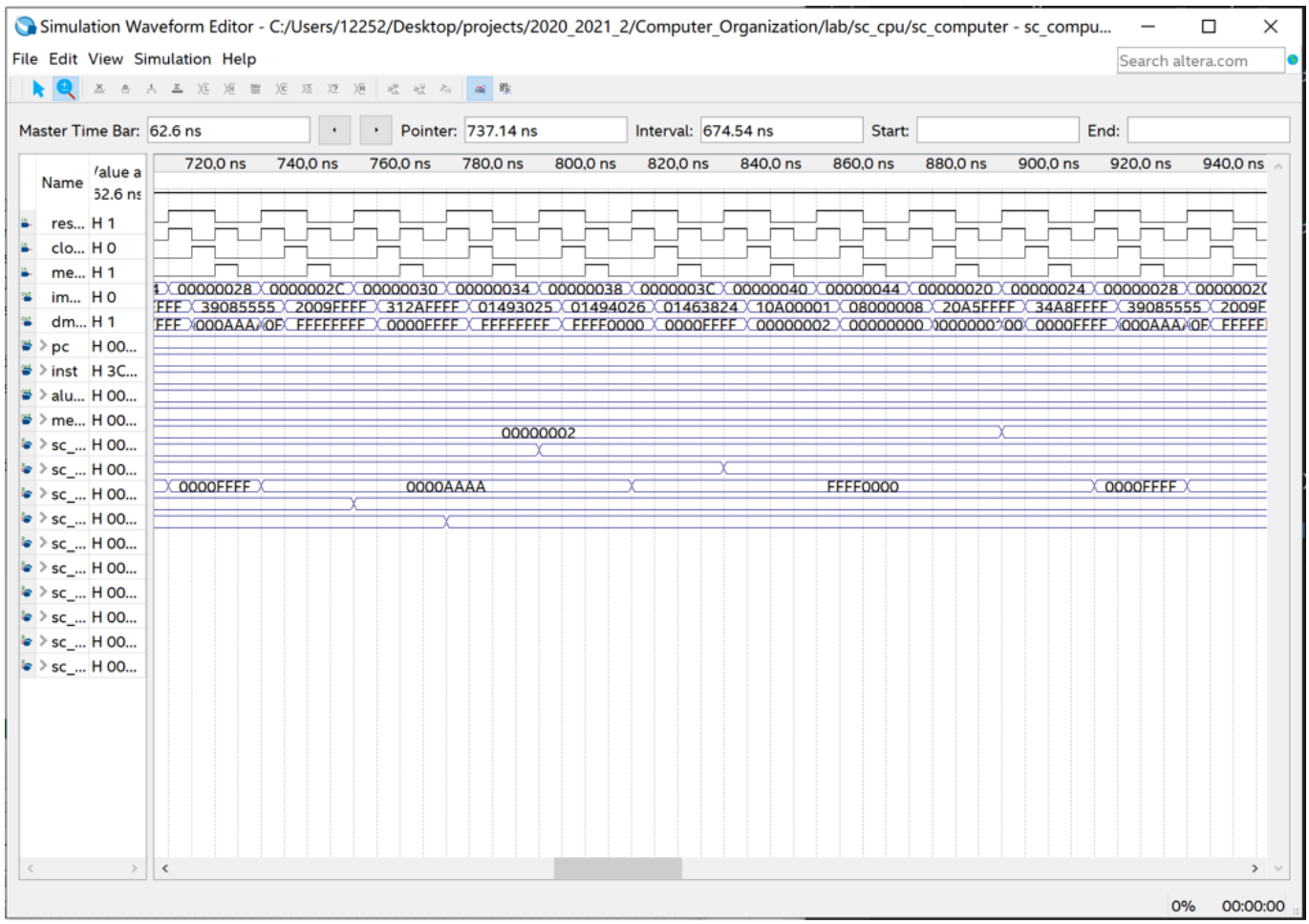


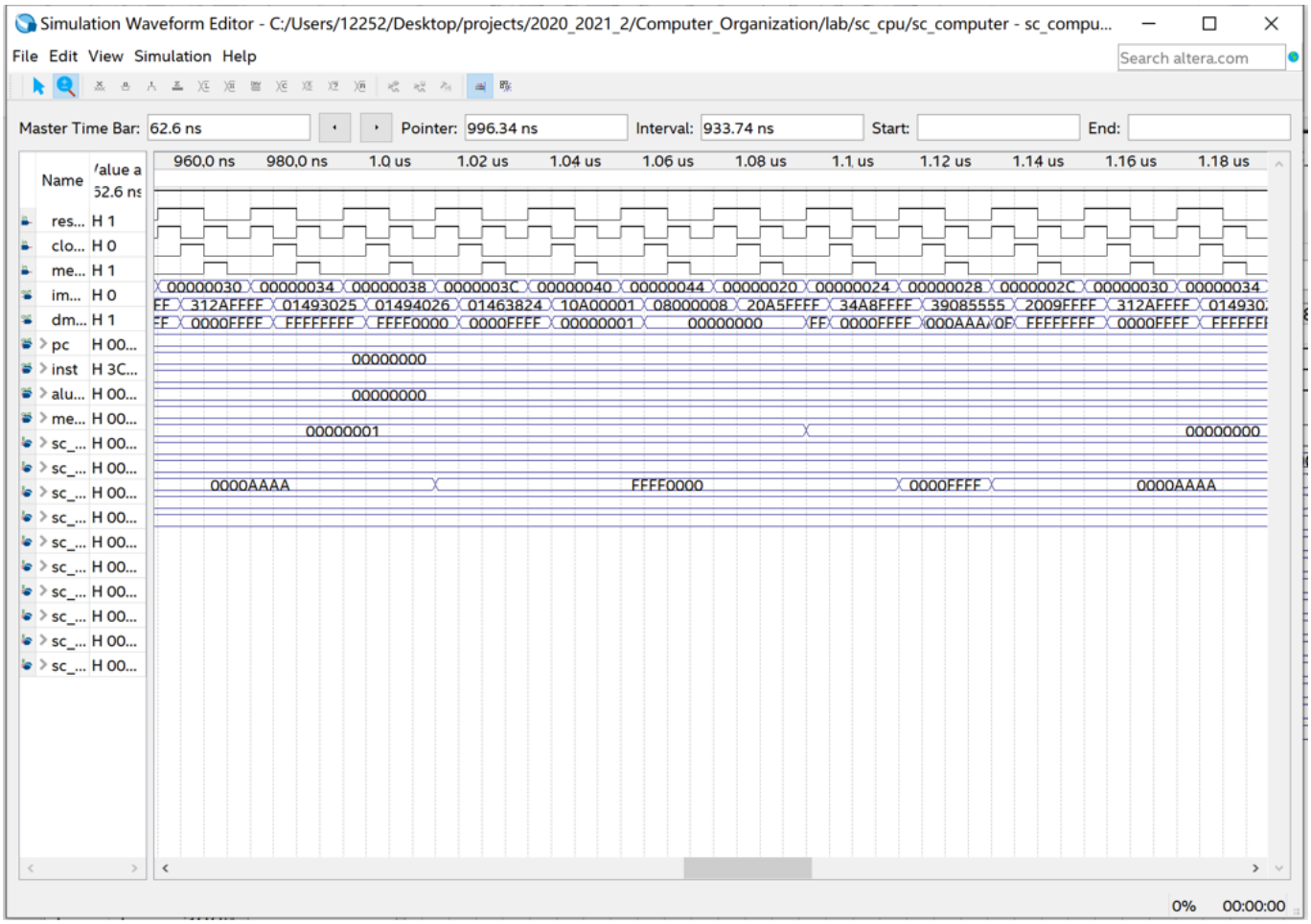
波形图细节（从上到下按照时钟周期排序）

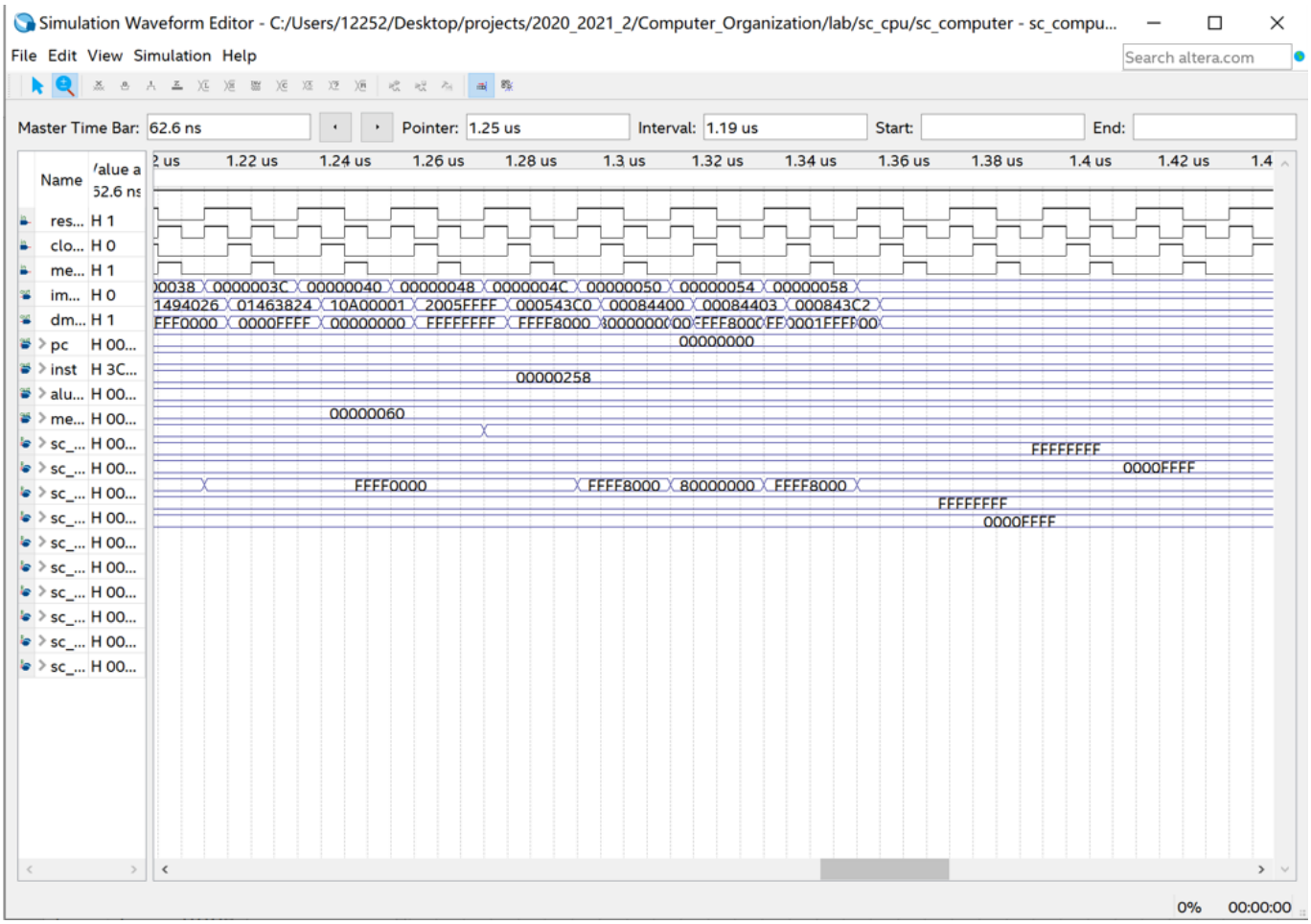

Simulation Waveform Editor - C:/Users/12252/Desktop/projects/2020_2021_2/Computer_Organization/lab/sc_cpu/sc_computer - sc_compu...
File Edit View Simulation Help
Search altera.com
Master Time Bar: 62.6 ns
Pointer: 288 ps
Interval: -62.31 ns
Start:
End:
Name
res...
clo...
me...
im...
dm...
pc
inst
alu...
me...
sc_...
62.6 ns
0% 00:00:00

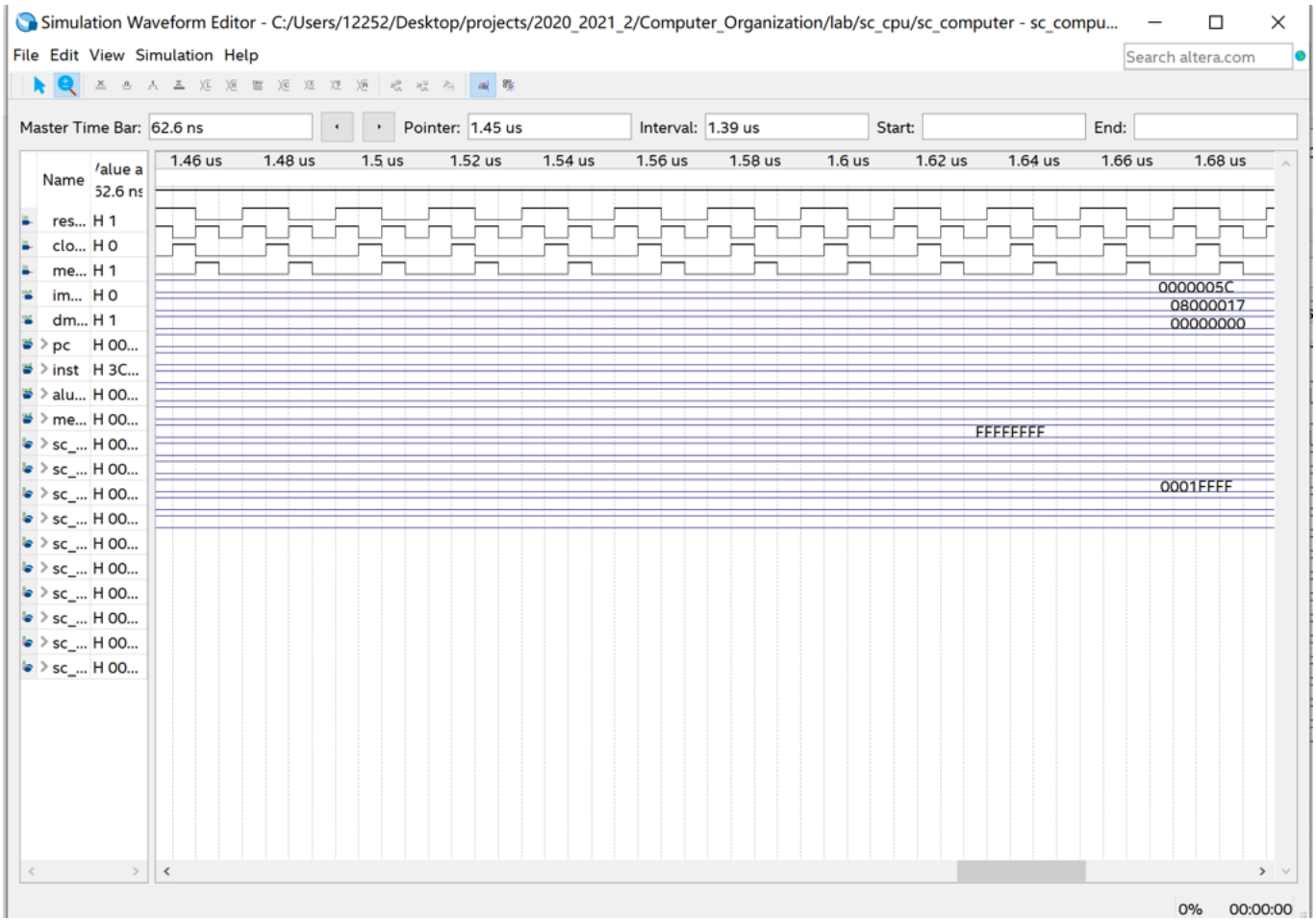

Simulation Waveform Editor - C:/Users/12252/Desktop/projects/2020_2021_2/Computer_Organization/lab/sc_cpu/sc_computer - sc_compu...
File Edit View Simulation Help
Search altera.com
Master Time Bar: 62.6 ns
Pointer: 242.35 ns
Interval: 179.75 ns
Start:
End:
Name
res...
clo...
me...
im...
dm...
pc
inst
alu...
me...
sc_...
0% 00:00:00


Simulation Waveform Editor - C:/Users/12252/Desktop/projects/2020_2021_2/Computer_Organization/lab/sc_cpu/sc_computer - sc_compu...
File Edit View Simulation Help
Search altera.com
Master Time Bar: 62.6 ns
Pointer: 471.6 ns
Interval: 409.0 ns
Start:
End:
Name
480,0 ns 500,0 ns 520,0 ns 540,0 ns 560,0 ns 580,0 ns 600,0 ns 620,0 ns 640,0 ns 660,0 ns 680,0 ns 700,0 ns
00000001
00000000
00000003
00000258
000001F8
00000115
00000258
0% 00:00:00


Simulation Waveform Editor - C:/Users/12252/Desktop/projects/2020_2021_2/Computer_Organization/lab/sc_cpu/sc_computer - sc_compu...
File Edit View Simulation Help
Search altera.com
Master Time Bar: 62.6 ns
Pointer: 737.14 ns
Interval: 674.54 ns
Start:
End:
Name
720,0 ns 740,0 ns 760,0 ns 780,0 ns 800,0 ns 820,0 ns 840,0 ns 860,0 ns 880,0 ns 900,0 ns 920,0 ns 940,0 ns
00000002
0000FFFF
0000AAAA
FFFF0000
0000FFFF
0% 00:00:00


Simulation Waveform Editor - C:/Users/12252/Desktop/projects/2020_2021_2/Computer_Organization/lab/sc_cpu/sc_computer - sc_compu...
File Edit View Simulation Help
Search altera.com
Master Time Bar: 62.6 ns
Pointer: 996.34 ns
Interval: 933.74 ns
Start:
End:
Name
960.0 ns 980.0 ns 1.0 us 1.02 us 1.04 us 1.06 us 1.08 us 1.1 us 1.12 us 1.14 us 1.16 us 1.18 us
res... H 1
clo... H 0
me... H 1
im... H 0
dm... H 1
pc H 00...
inst H 3C...
alu... H 00...
me... H 00...
sc_... H 00...
00000030 00000034 00000038 0000003C 00000040 00000044 00000020 00000024 00000028 0000002C 00000030 00000034
312AFFFF 01493025 01494026 01463824 10A00001 08000008 20A5FFFF 34A8FFFF 39085555 2009FFFF 312AFFFF
0000FFFF FFFFFFFF FFFF0000 0000FFFF 00000001 00000000 0000FFFF 0000AAAA FFFFFFFF 0000FFFF
00000000
00000000
00000001
00000000
0000AAAA
FFFF0000
0000FFFF
0000AAAA
0%
00:00:00


Simulation Waveform Editor - C:/Users/12252/Desktop/projects/2020_2021_2/Computer_Organization/lab/sc_cpu/sc_computer - sc_compu...
File Edit View Simulation Help
Search altera.com
Master Time Bar: 62.6 ns
Pointer: 1.25 us
Interval: 1.19 us
Start:
End:
Name
1.22 us 1.24 us 1.26 us 1.28 us 1.3 us 1.32 us 1.34 us 1.36 us 1.38 us 1.4 us 1.42 us
res... H 1
clo... H 0
me... H 1
im... H 0
dm... H 1
pc H 00...
inst H 3C...
alu... H 00...
me... H 00...
sc_... H 00...
0003C 00000040 00000048 0000004C 00000050 00000054 00000058
01463824 10A00001 2005FFFF 000543C0 00084400 00084403 000843C2
00000000 FFFFFFFF FFFF8000 FFF8000 0001FFFF
00000000
00000258
00000060
FFFFFFFF
0000FFFF
FFFF0000 FFFF8000 80000000 FFFF8000
FFFFFFFF
0000FFFF
0%
00:00:00



可以看到和老师给出的参考波形图一样，可以认为仿真正确。

总结

通过本次实验，对教科书中有关单周期CPU的部分有了更加深刻的了解，同时也更加熟悉了使用FPGA进行仿真的流程。