Lab1 运算器与寄存器

1. 设计逻辑
2. ALU：为了实现ALU的加、减、与、或、非、异或功能，用自己拟定的4位操作码来区分功能：

0001——加 0010——减 0011——与

0100——或 0101——非 0110——异或

用行为级建模风格电路。减法运算为减数取反加一后做加法运算

对于进位的判别：最高位有进位（此时当作无符号数）

对于借位的判别：被减数小于减数，结果符号位为1

对于溢出的判别：作为有符号数时，加法：最高位进位和次高位不同

减法：取反加一作为加法

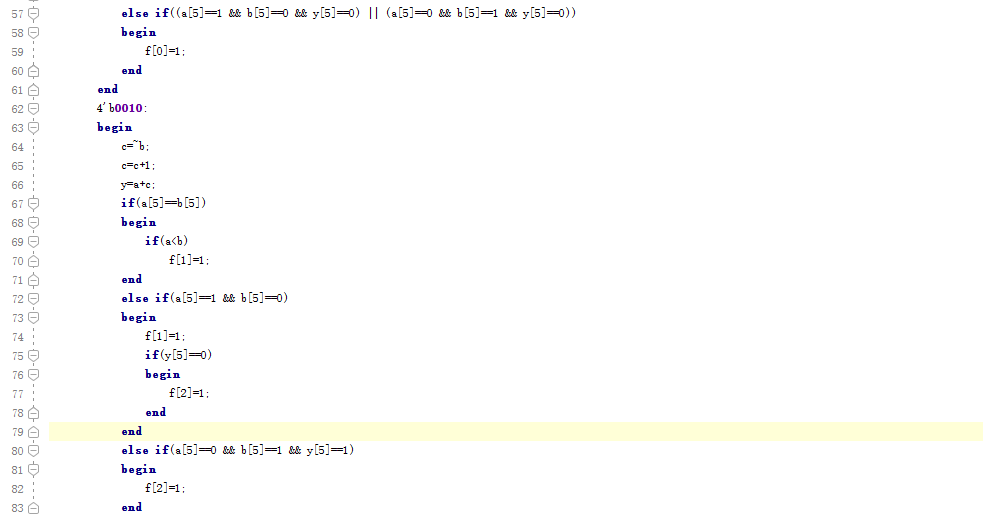
对于零标志的判别：结果为6’b000000则零标志为1

4位标志位 f[3]=1零标志 f[2]=1溢出 f[1]=1借位 f[0]=1进位

1. Fibonacci数列：利用ALU的加法功能和存储器的存值功能实现Fibonacci数列的输出。将两个6位初始值分别置于两个存储器清零后的输入端，存储器为时钟上沿触发，存储器输出接入ALU的两个输入端，限定ALU的操作数为0001执行加法运算，并将输出结果接入第二个存储器。在下一个时钟上沿，第一个存储器的值更新为第二个存储器的值，第二个存储器的值更新为ALU的输出结果。
2. 核心代码



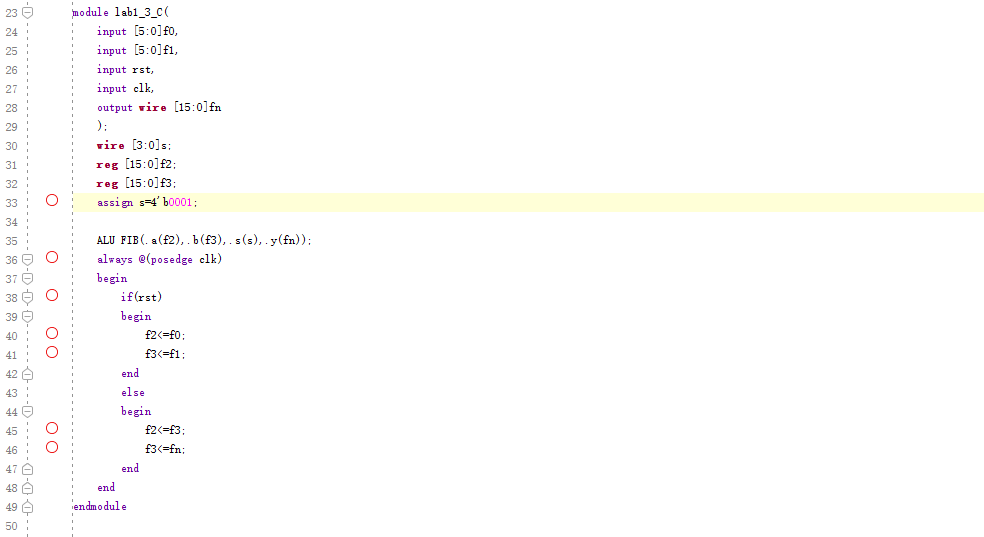
ALU\_code(1)



ALU\_code(2)

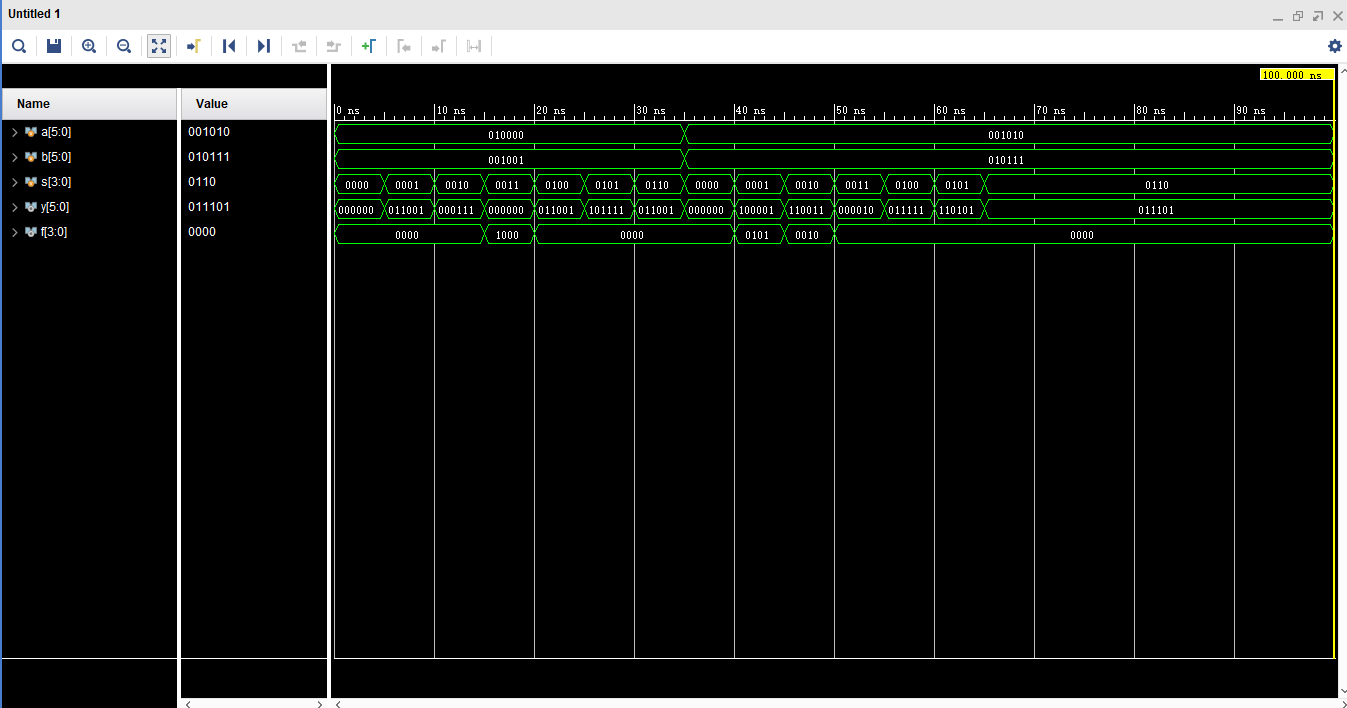
这里只放ALU加法与减法的代码，其他运算过于简单，略过

Fibonacci代码：



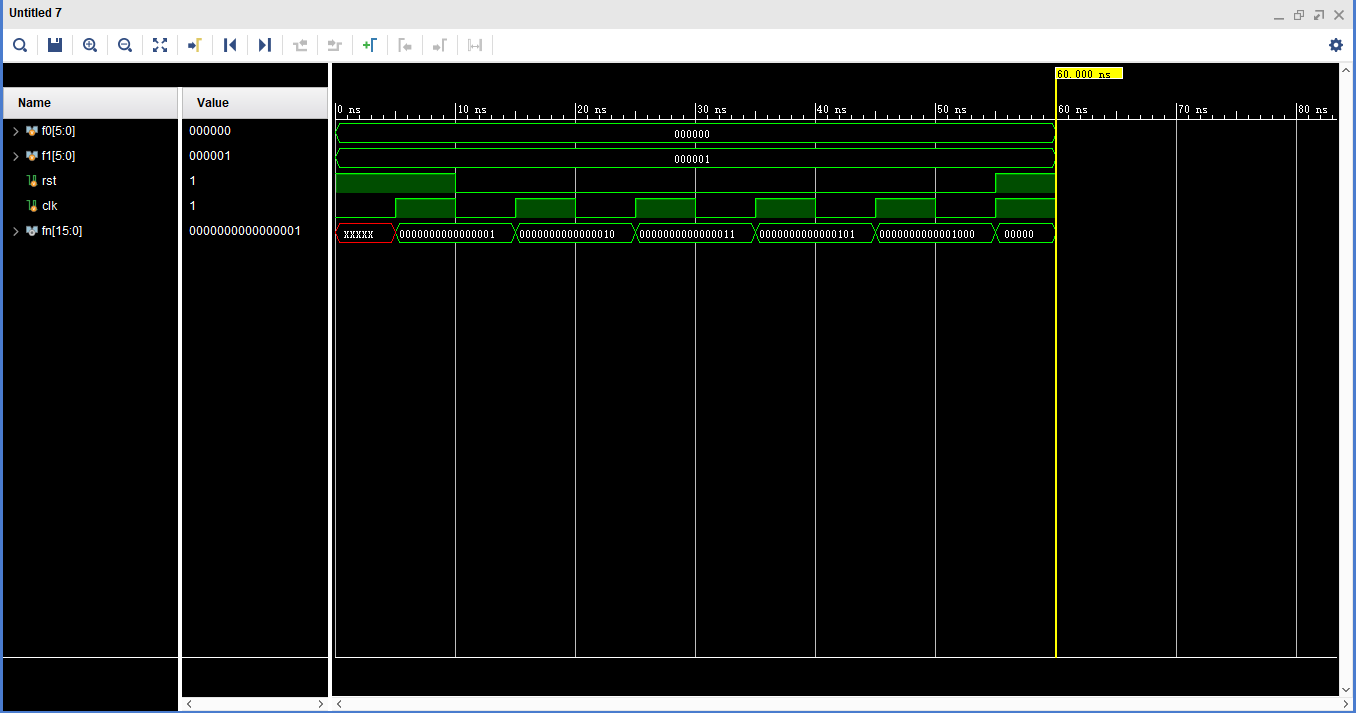
Fibonacci\_code

1. 仿真结果与下载结果：

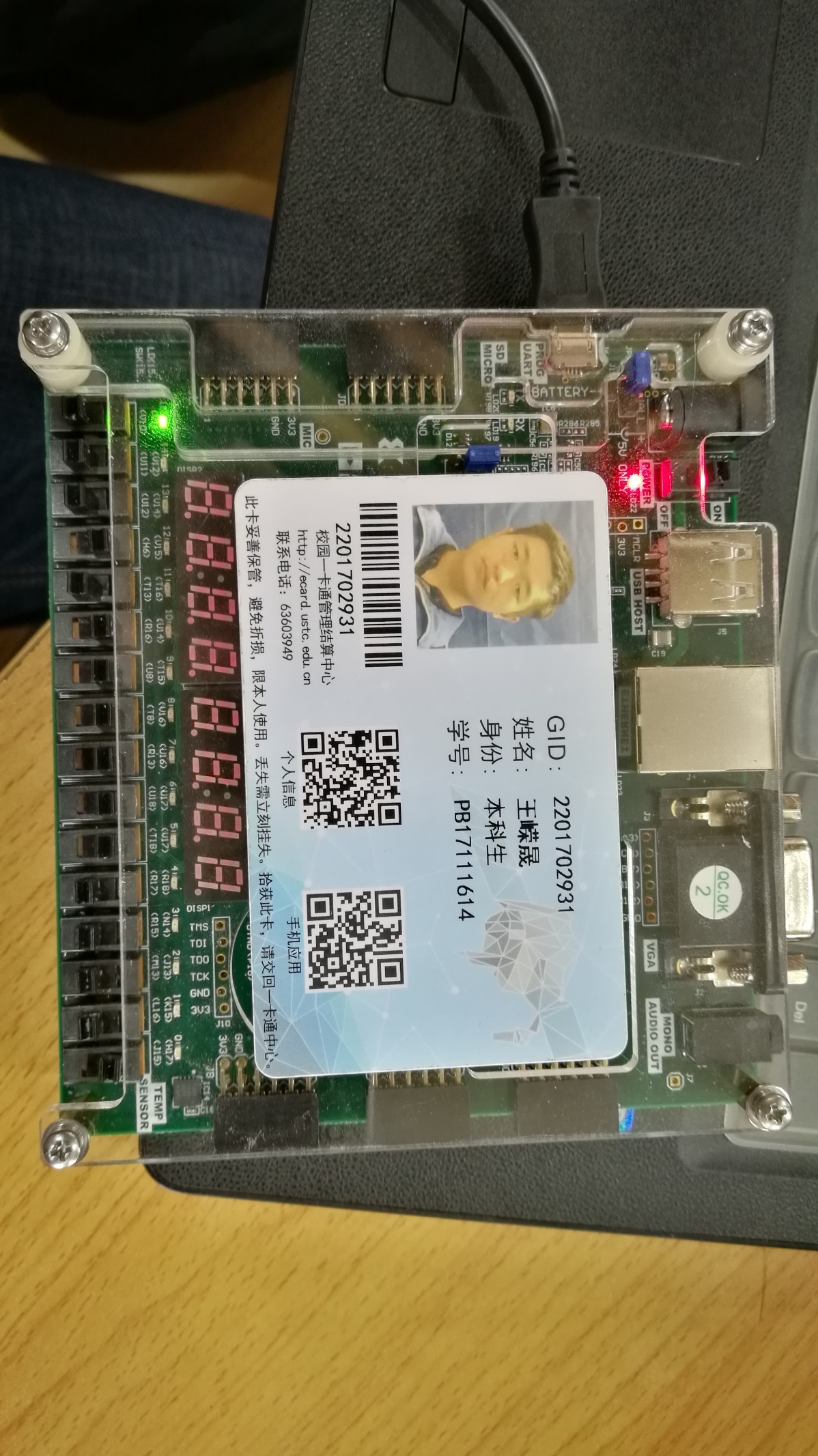


ALU\_simulation

做ALU的仿真时，输入了两组变量：16 9和8 23，分别进行六种运算得到结果



Fib\_simulation

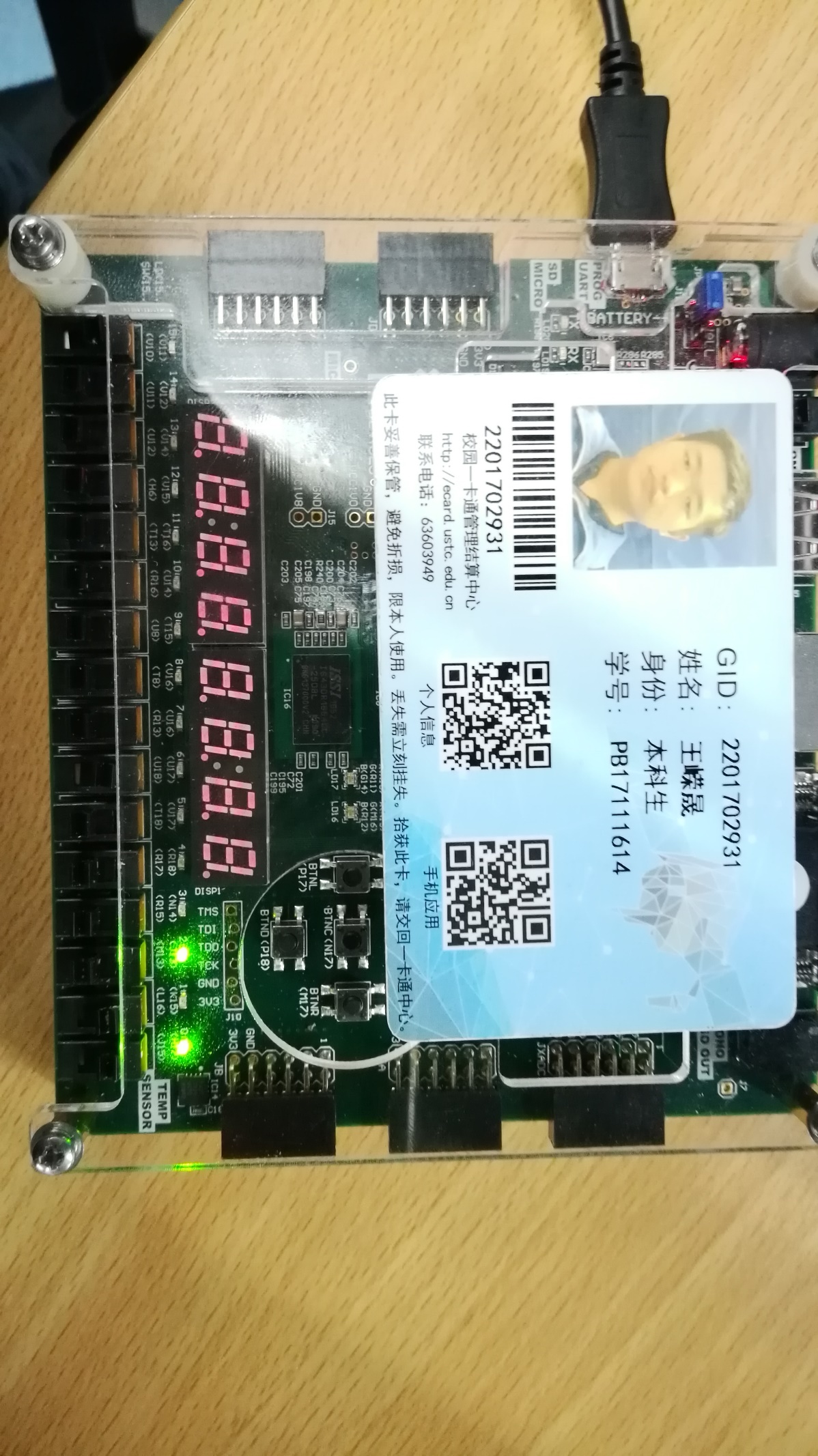
做Fib的仿真时，设定初始输入的值为0和1，在reset后启动

下载照片：

ALU下载照片（1）



ALU下载照片（2）







上图为输出Fibonacci数列的抓拍，初始值为2 3

1. 结果分析

ALU运算结果准确，Fibonacci数列输出准确

1. 实验总结

在近3个月没有写verilog代码后，对语法有些生疏。还好在编写代码时慢慢找回了感觉。设计ALU以及用ALU和存储器的组合输出Fibonacci数列以实现ALU的应用，算不上难度很大的实验，大概主要为了适应。在Fib仿真时发现代码逻辑的诸多问题，但已成功修改。

1. 意见/建议：

希望在今后的实验中实验要求可以给的更明确，防止引起歧义而带来没有必要的时间浪费。