实验1：32位MIPS单周期处理器的设计

完成日期：2022/3/18 周训哲 20307110315

一、设计原理

1.体系结构状态和指令集

R类型算数/逻辑指令：add、sub、and、or、slt

存储器指令：lw、sw

分支指令：beq

拓展指令：addi、j、bne、ori、andi

2.MIPS结构设计

可将微体系结构分为两个互相作用的部分：数据路径（datapath）和控制（control）。

其中数字路径具体执行控制的指令，并对数据进行操作；而控制又接受到数字路径传来的指令，再控制数据路径执行。

数字路径包含了寄存器（regfile）、ALU、复用器（mux）、移位器（sl）、加法器（adder）、扩展器（符号扩展/零扩展）、计数器（flopr）等结构元件。

控制包含了主译码器（maindec）和alu译码器（aludec）两个部分。

3.单周期微体系结构设计

单周期微体系结构：在一个时钟周期内执行完一条完整指令(CPI=1)。 时钟周期以最长的指令(如Load指令)所花的时间为准。 控制简单，但速度慢、成本高。

单周期微体系结构在MIPS的基础上增加了外部存储接口，将指令存储器（imem）和数据存储器（dmem）与主处理器分离，通过地址和数据总线进行连接。

4.增加I/O接口

对于数据存储译码器模块进行更改，除了数据存储器以外，增加了I/O接口，用来处理NEXYS4 DDR开发板的外部输入，并且引入七段数码管模块对计算结果进行led显示的输出。

其中数据存储译码器（DataMemoryDecoder）模块对控制器传入的write信号进行选择，选择写入的数据传入数据存储器还是I/O接口，对处理器读入的数据也进行选择，是外部输入的数据还是数据存储器存储的数据，最终计算结果由七段数码管进行输出。

二、实验方案

1.存储器模块

（1）指令存储器（imem）：

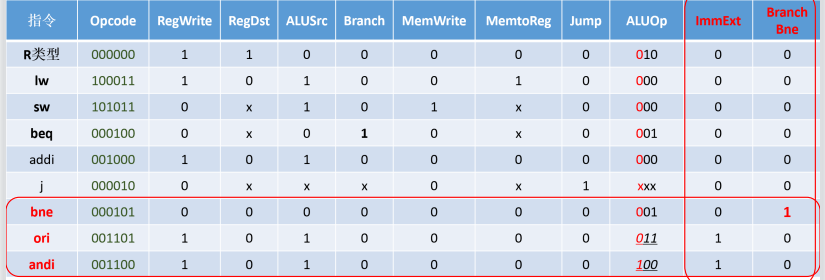
有一个读取端口，用来读取已有文件编写的汇编语言，并且将数据传输到rd上。

（2）数据存储器（dmem）：

有一个读/写端口，如果使能we为1，则在时钟的上升沿将数据wd写入地址a，如果使能为0，则将地址a读到rd。

2.指令的执行：

根据opcode，将控制指令进行拆解，得到主译码器真值表，分别对应各种控制指令，输出到aludec与datapath模块中进行计算和执行。



3.controller：

（1）maindec主译码器：

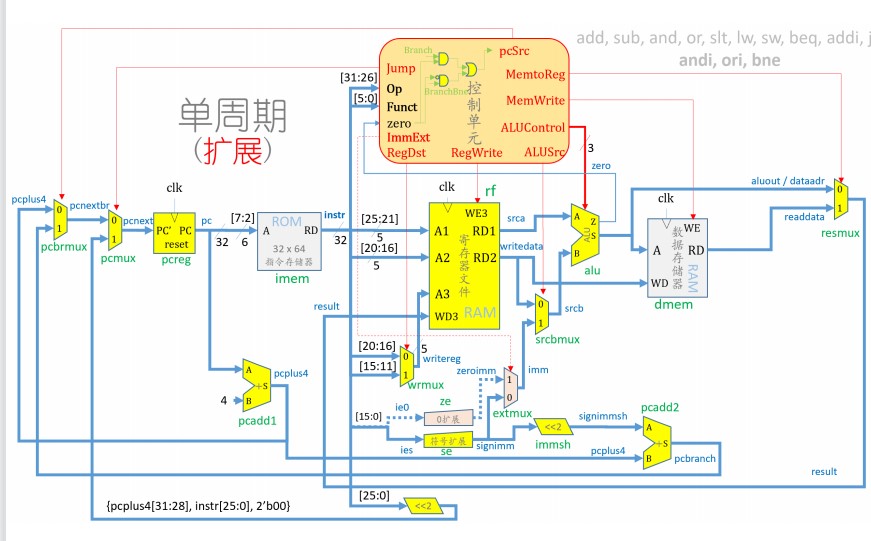
输入op地址，根据op地址的真值表，对regwrite,regdst,alusrc,branch,memwrite,memtoreg,jump,aluop,immext,branchbne进行赋值，并将数据输出。

（2）aludecALU译码器：

输入功能指令及aluop指令，得到alucontrol指令对alu模块的计算进行控制。

（3）增加bne指令后，还要对分支指令进行处理。

4.datapath：主要的数据处理部分



（1）首先，由指令存储器读入数据，将instr[25:21],instr[20:16]与instr[15:11],instr[20:16]的由regdst选择的结果传入到寄存器文件中：32个单元×32位寄存器文件有两个读端口和一个写端口。读端口具有5位地址输入 A1和A2，每个用于指定2^5=32个寄存器中的一个作为源操作数。它们可以读32 位寄存器的值并分别传送到 RD1和RD2 上。写端口具有5位地址输入 A3.32位数据输入 WD.写入使能 WE3和时钟信号 CLK。如果写入使能为1，则寄存器文件将在时钟的上升沿将数据写入特定寄存器。

（2）instr[15:0]进行位扩展,符号扩展及零扩展的由immext选择的结果,再与rd2由alusrc选择的结果选择得到srcb

将寄存器rd1作为scra与srcb在alucontrol指令下在alu模块进行运算输出,并且alu模块返回zero传入控制单元。

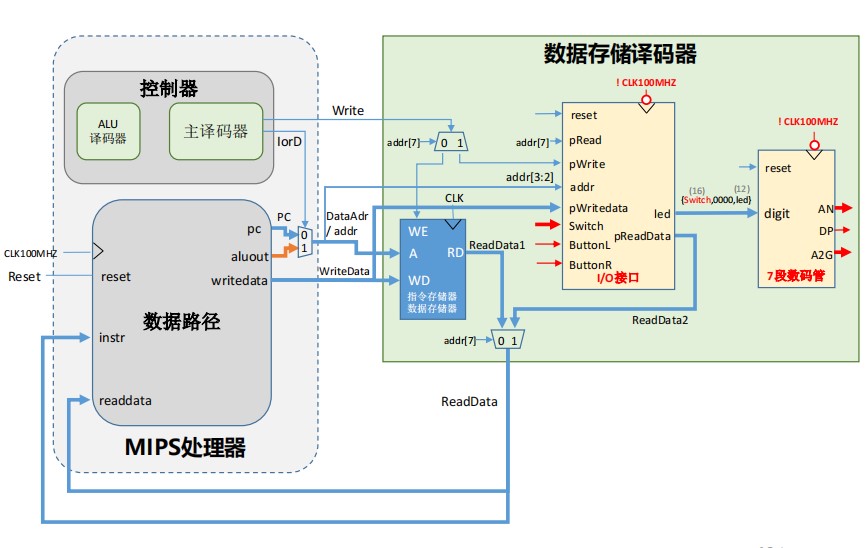
（3）alu模块输出计算结果s传输给数据存储器（dmem）。

同时，memwrite作为we，rd2作为wd，输入到dmem中，输出结果与s在memtoreg的信号控制下选择输出为最终结果result，然后result作为wd3返回输入到寄存器文件中。

（4）符号扩展结果经移位器之后与pcreg模块输出（pc+4）进行加法运算得到pcbranch与（pc+4）在pcsrc的控制下选择得到pcnextbr。

（5）instr[25:0]移位处理后{pcplus4[31:28],instr[25:0],2’b00}与pcnextbr在jump信号的选择下得到pcnext通过pcreg模块处理得到pc。传输到指令存储器中，作为数据结果。

5.数据存储译码器

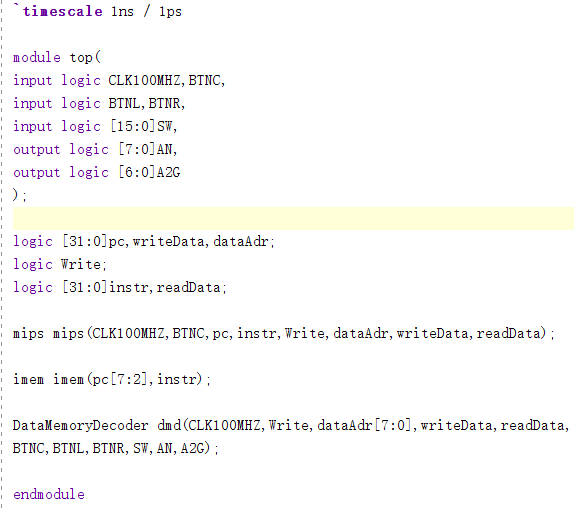


主要分为三个模块，数据存储器对传输的数据进行存储及运算输出，I/O接口接收外部接口（switch）的数据，将数据进行输出，同时led数据对七段数码管进行控制，使开发板显示出结果。

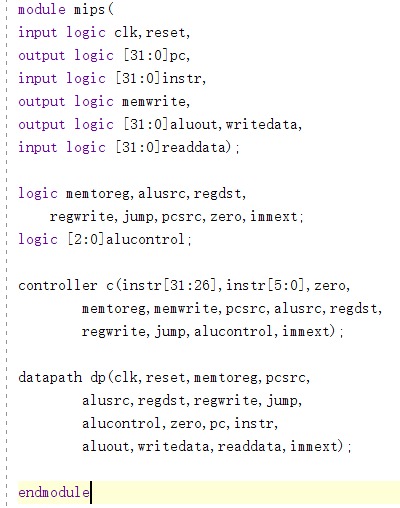
Datapath中alu模块输出aluout作为addr,addr[7]作为pread控制读入操作,控制器传出的write在addr[7]的控制下,选择输入到dmem与io控制读入的模块。Switch，btnl，btnr作为外部信号控制读入与写出操作。最终readdata在addr[7]的控制下选择输出结果作为instr信号传入数据路径中。

三、关键代码

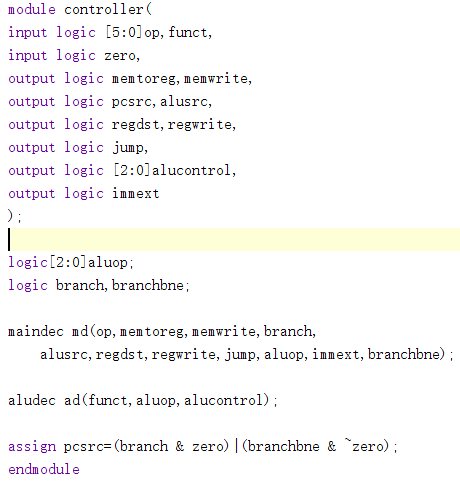
**Top：**



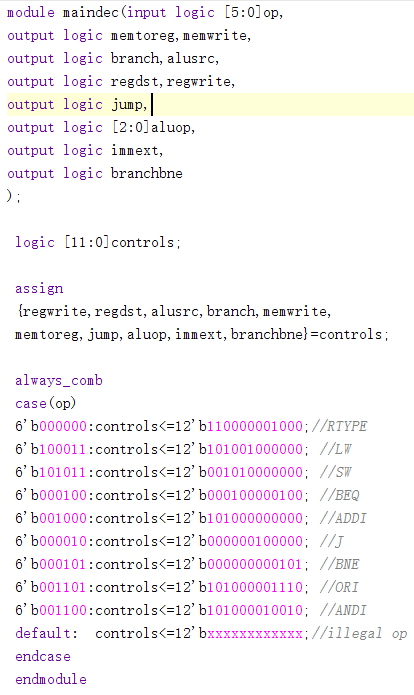
**Mips**



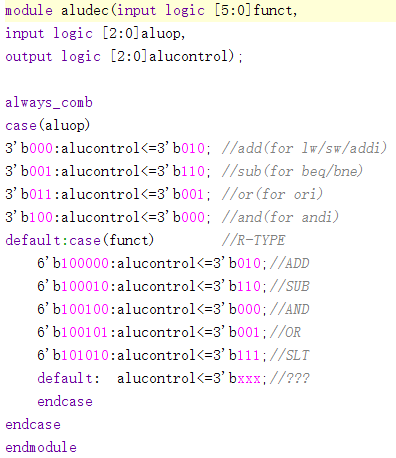
**Controller**



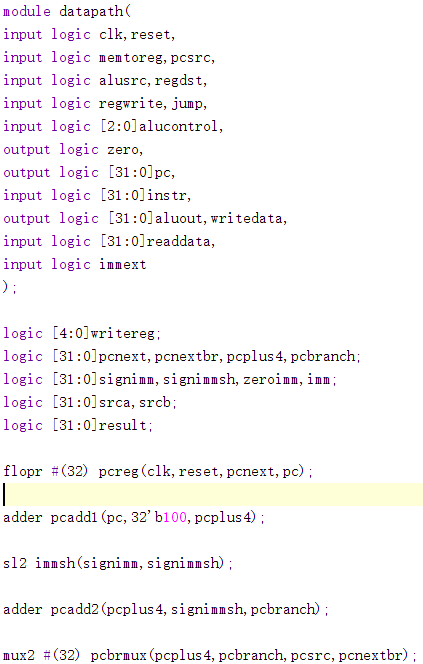
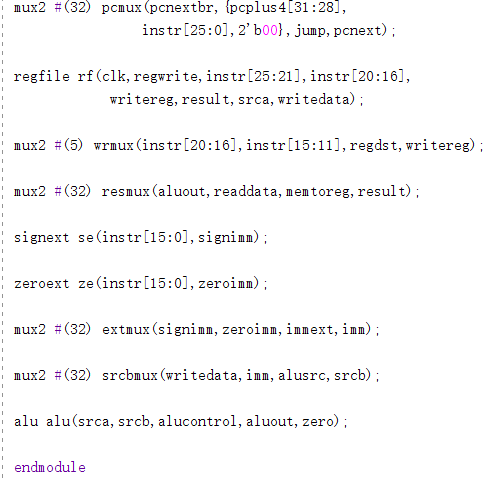
**Maindec**



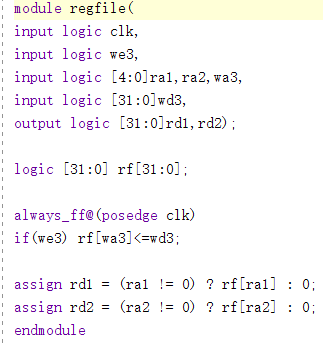
**Aludec**



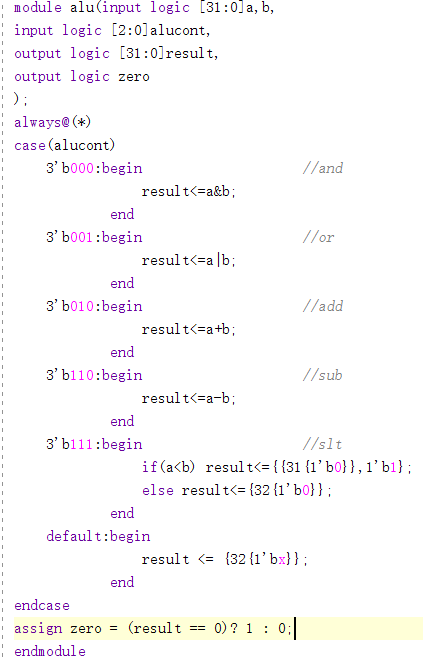
**Datapath**

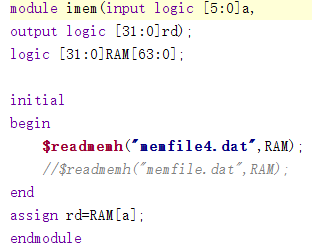
**Regfile**



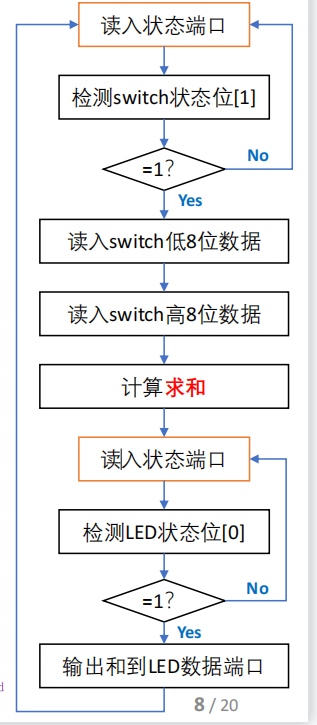
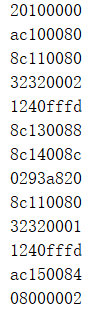
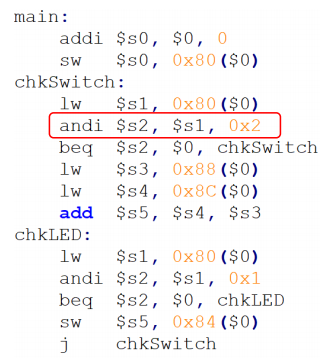
**Alu**



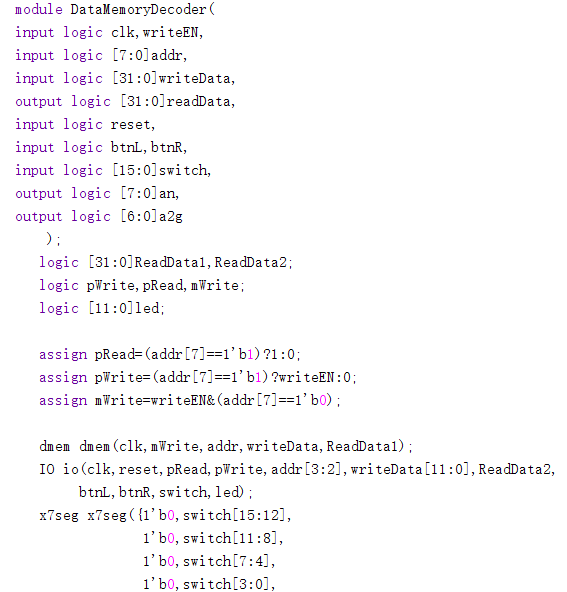
**Imem**

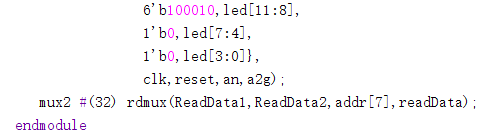


**Memfile4.dat**

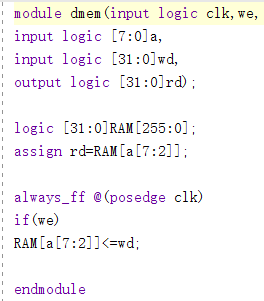


**Datamemorydecoder**

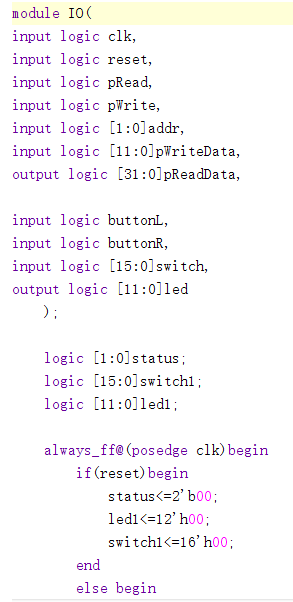


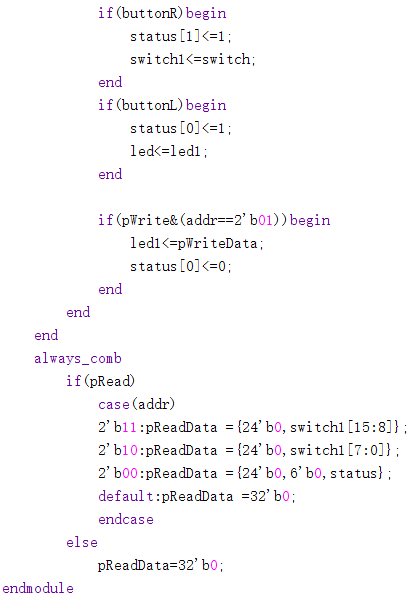


**Dmem**

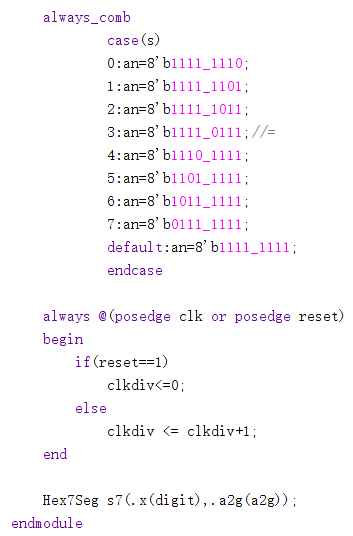
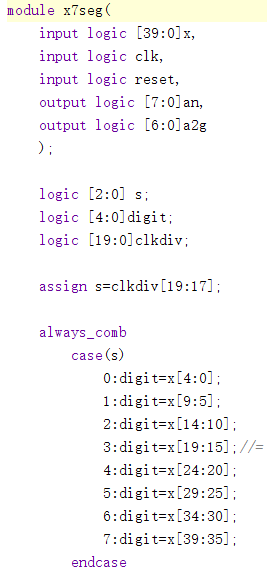


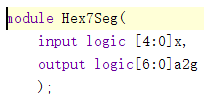
**IO**





**X7seg**

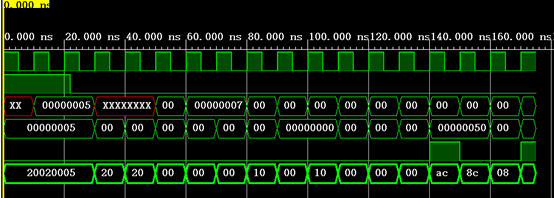




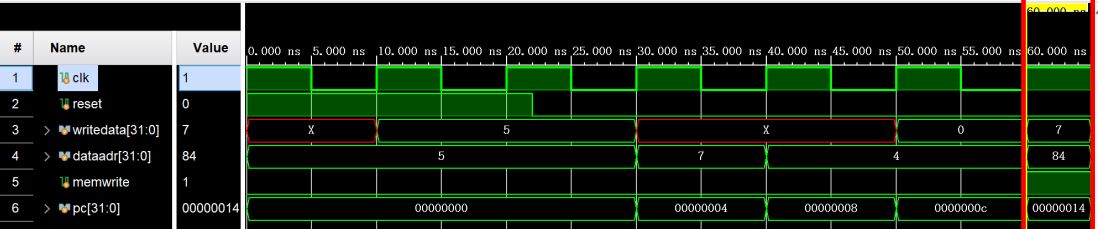


四、仿真截图

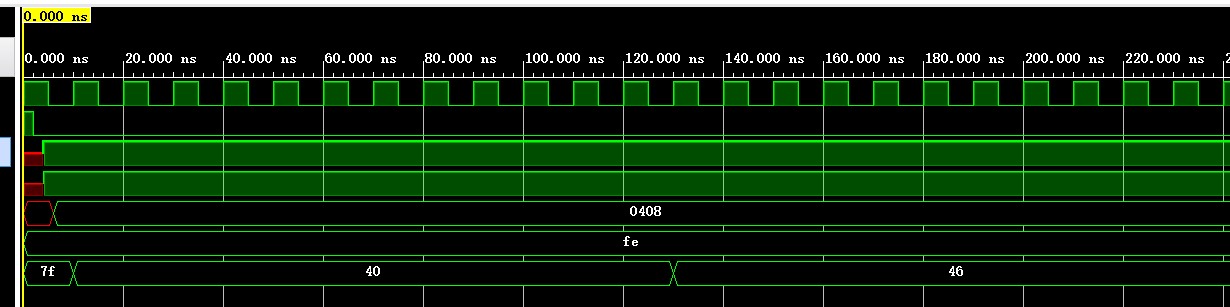
1.单周期处理器仿真测试



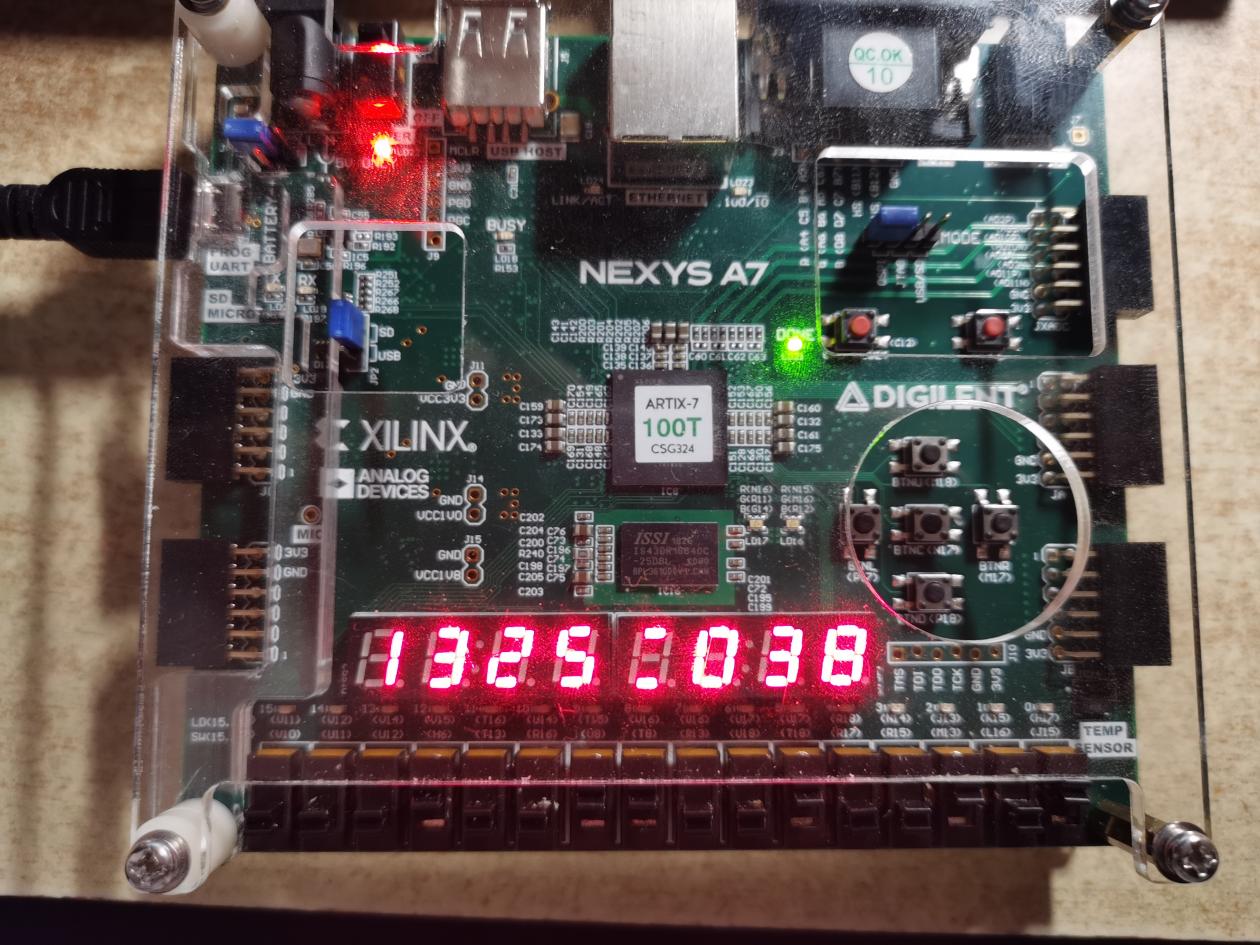
2.指令扩展后仿真测试(测试时忘记截图了,所以借用ppt的图片)

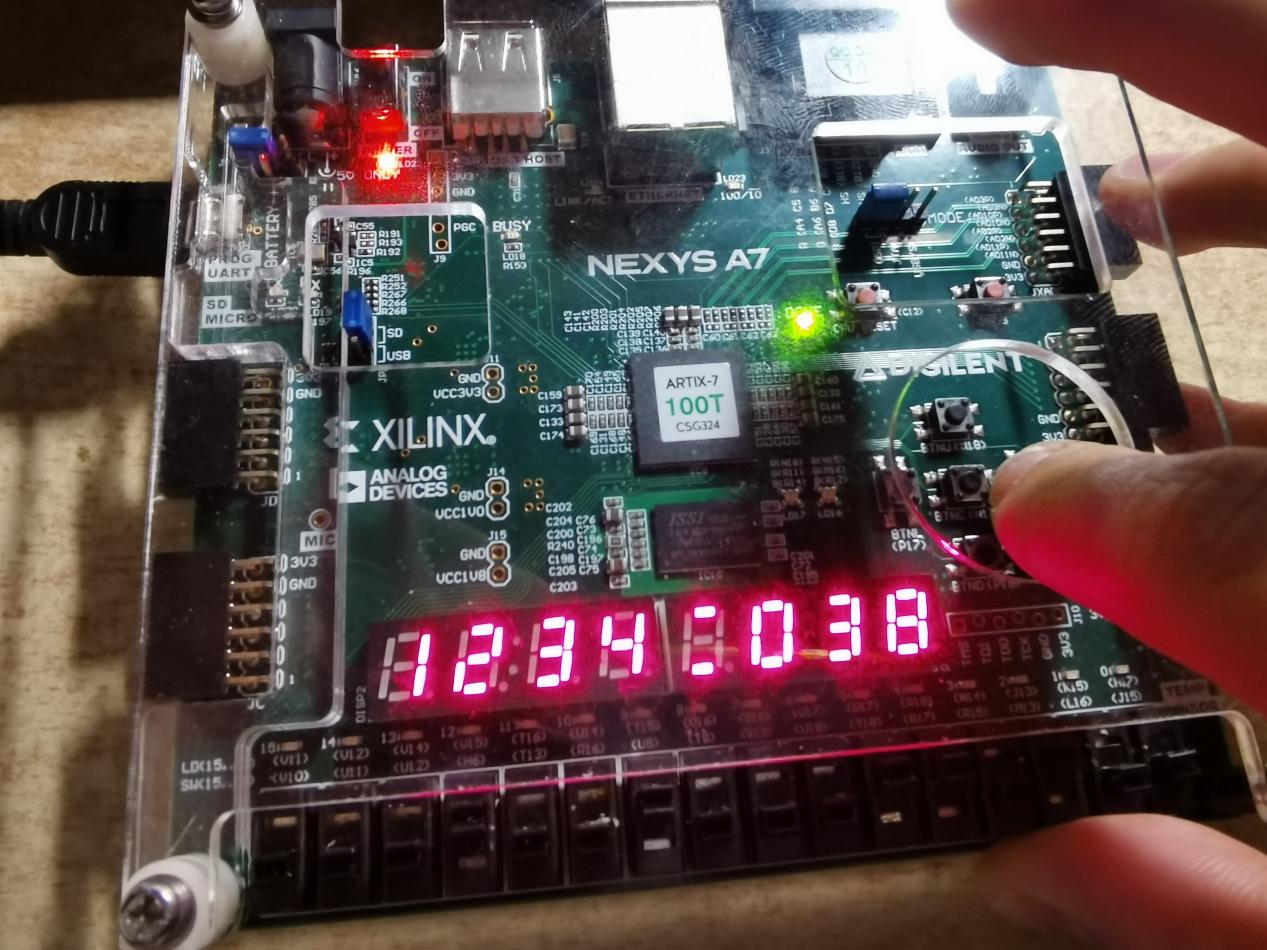


3.增加I/O接口仿真测试

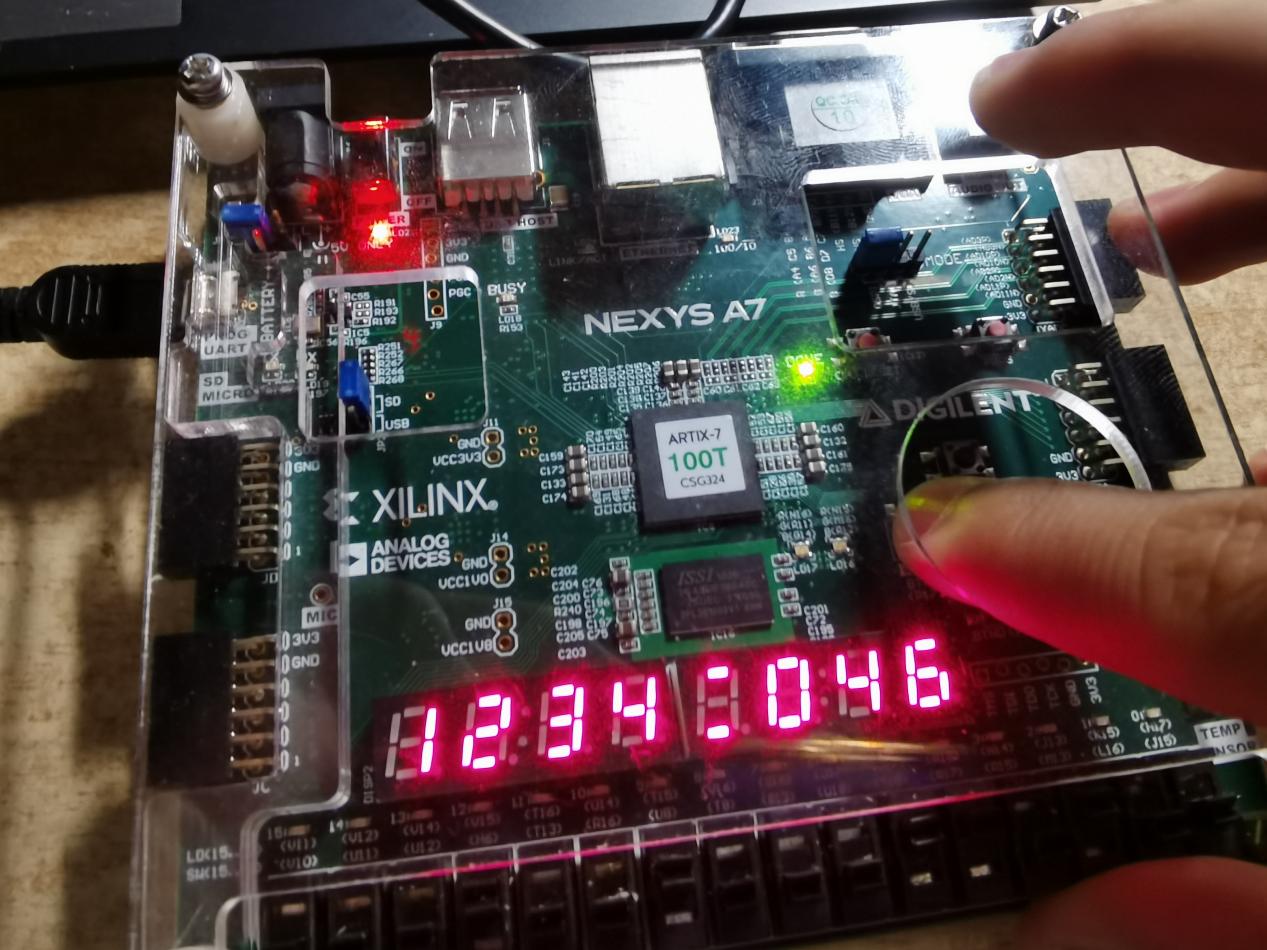


五、实验开发板照片

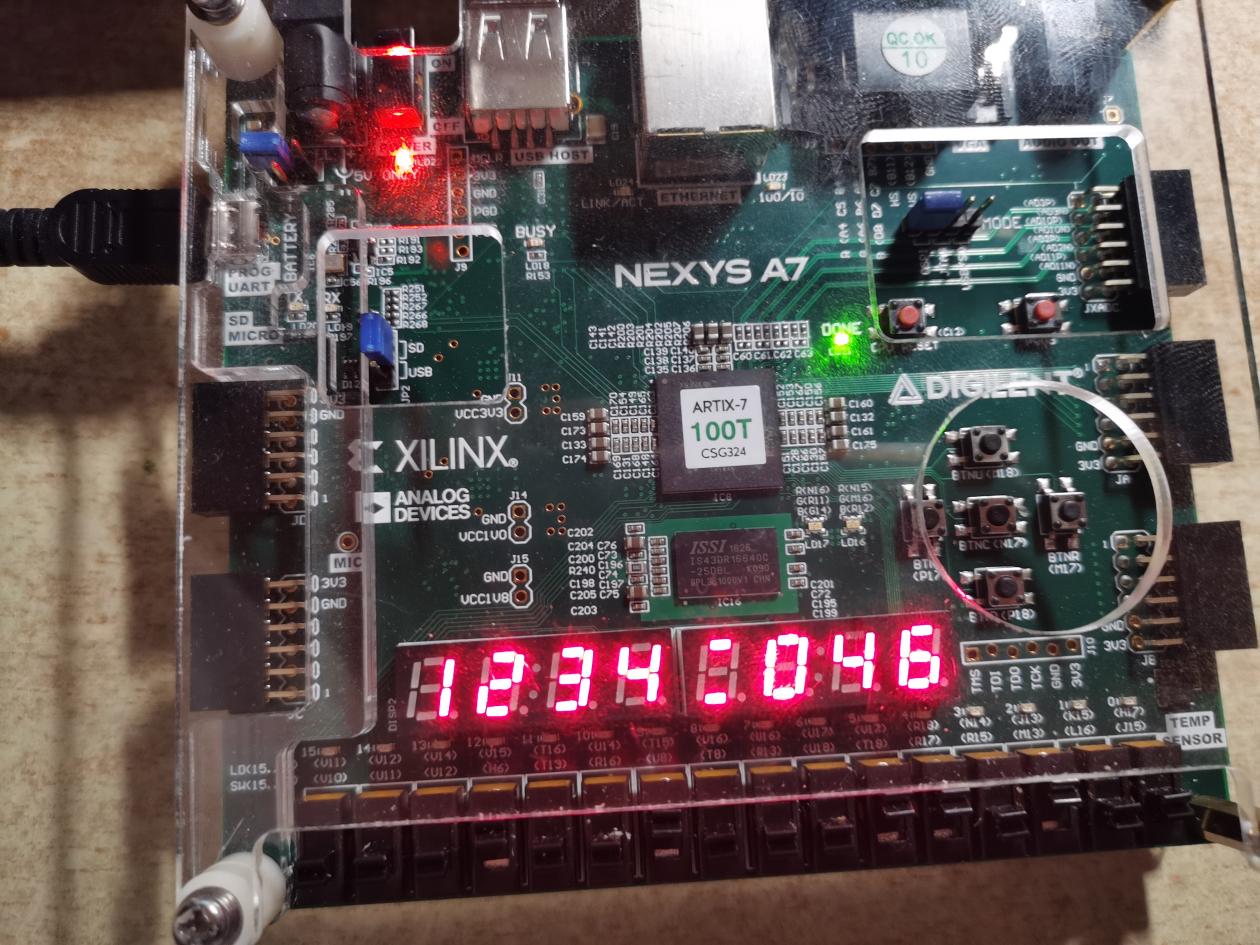




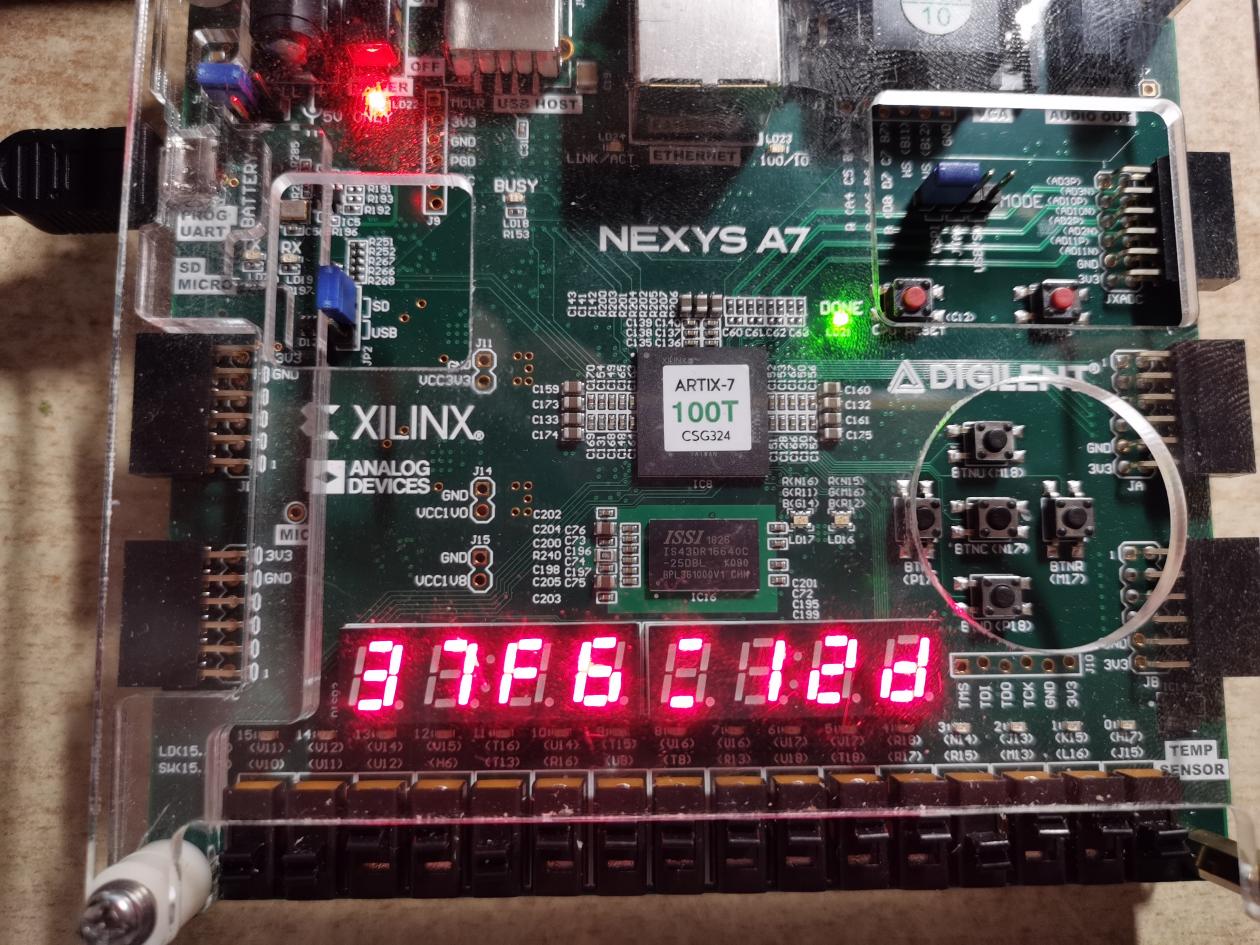
使用btnr读取数据

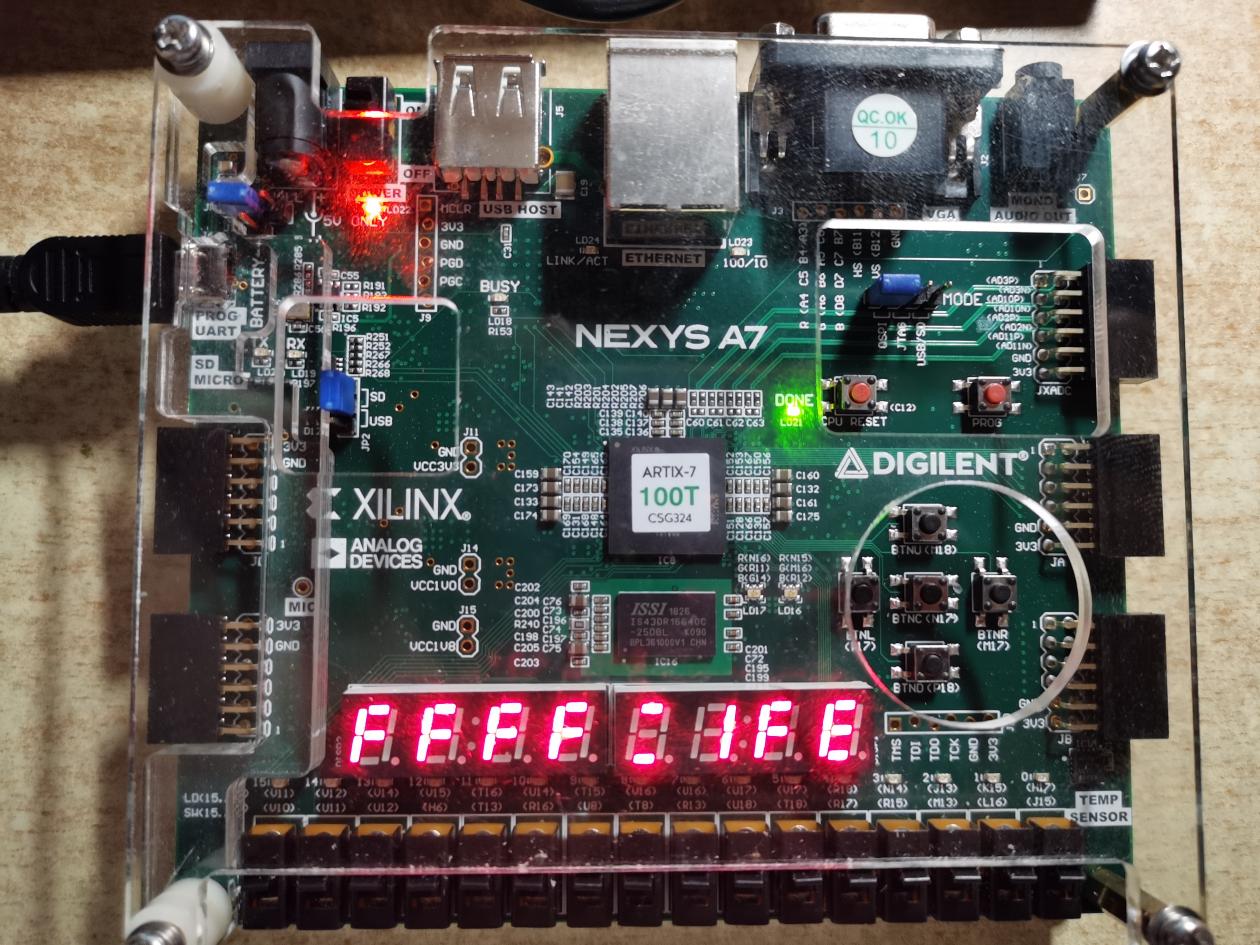


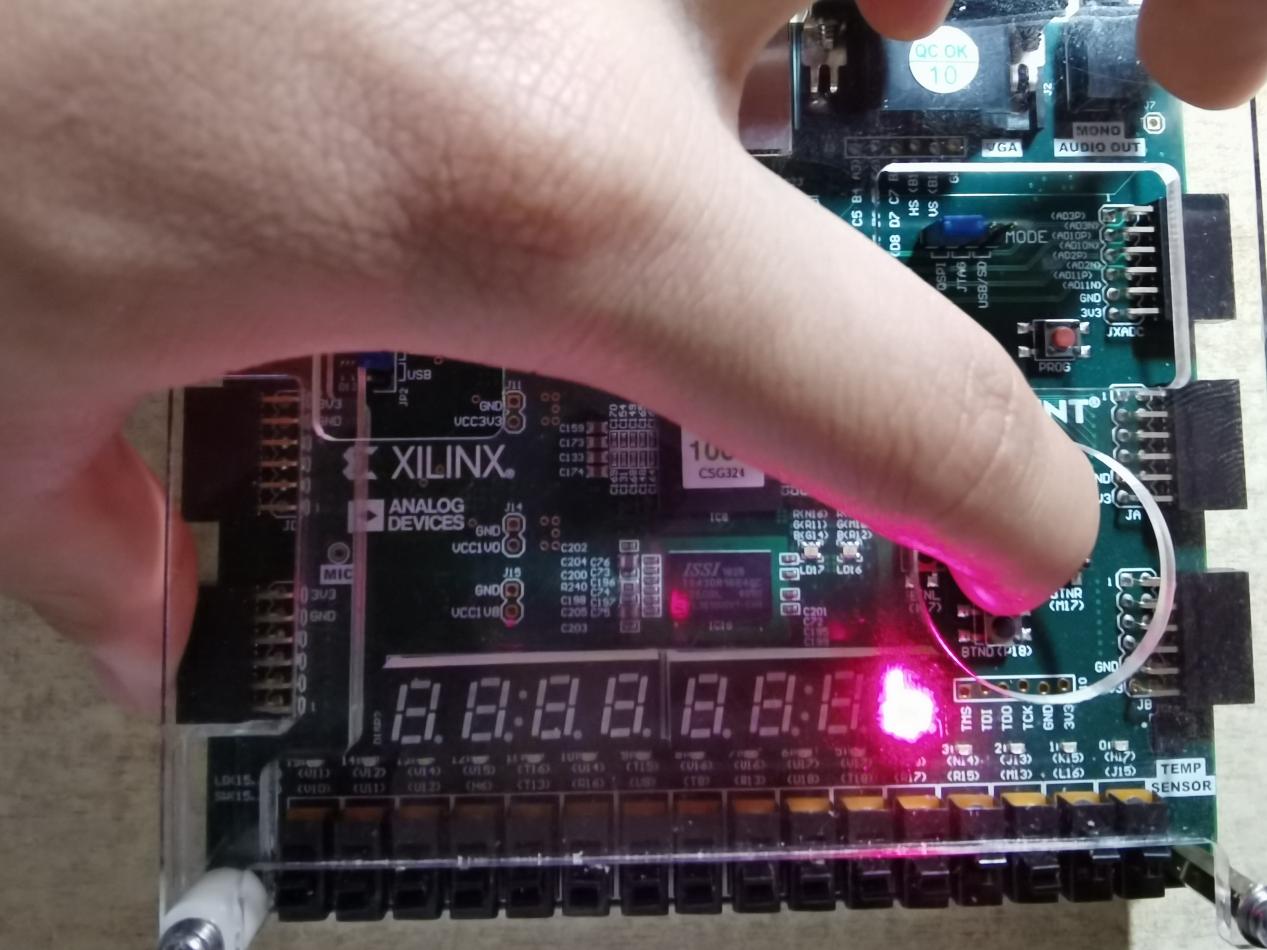
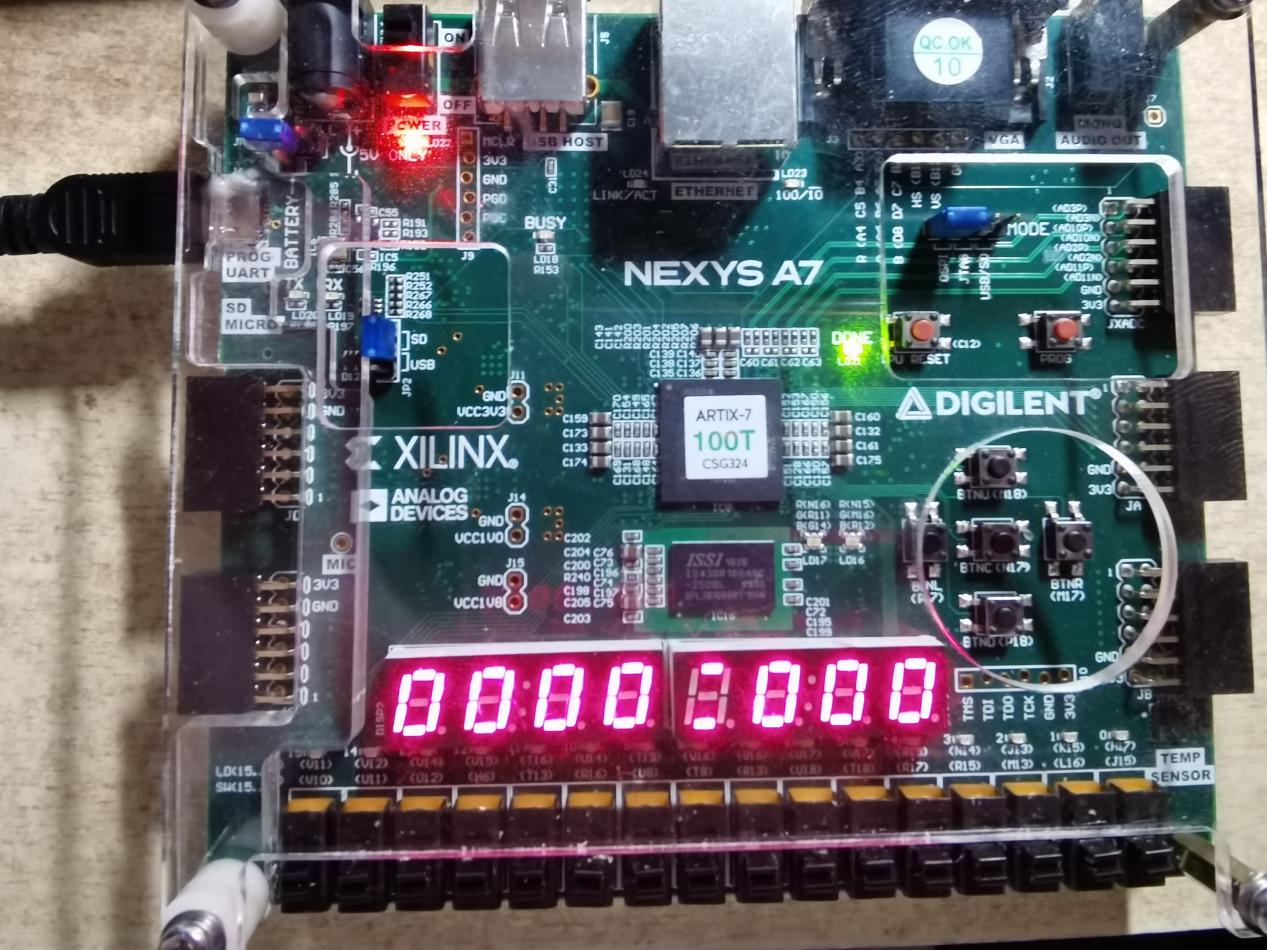
使用btnl计算并显示结果



最终结果展示







使用btnc实现重置操作

六、总结

1.调试发生的问题与对应解决方案

1.运行结果进入到default，仿真显示结果全为x

解决方案：利用仿真数据，根据电路图，逐层向上查找，最终寻找到发生错误的最初模块，进行检查与修改。

2.运行结果有浮空值z

解决方案：说明有结果并未读入，检查输入端是否有数据输入，若无，则检查上一层模块的数据是否正常，若正常，这检查浮空值模块代码是否有误，若异常，则检查上一层代码是否存在问题，检查方式为逐层检查。

3.io接口时，仿真文件无法得到结果，在开发板上运行时结果显示为0

解决方案：先检查是否有数据出错，最终发现是dmem给出的位数不够，更改后发现仍无法得出结果，根据warning逐层修改。最终发现在imem读入汇编代码时就无法读入，更改文件格式从.txt到.dat后结果运行正常。

2.收获与体会

由于本次实验为设计一个完整的mips单周期处理器，故模块数量及功能数，端口数均远超于之前所学的简单模块，所以报错后调试难度也急剧增加。这次实验除了让我了解了cpu的工作原理之外，最重要的还是让我掌握了调试的技巧，以及报错之后的修改方式。本次实验，不仅会加深我对cpu构造的认识，还会对我今后的硬件语言学习与调试有着极大的帮助！