计算机组成原理第八次实验报告

计科一班 劳嘉辉 13349051

实验目的

在 FPGA 实验板上实现 lab6(2)的内容。即利用 ISE 软件在 FPGA 实验板上实现 pipeline 的 datapath 以及 data-forward 和 load-hazard-detect 等实验六要求的机制。

实验难点

1、 优化设计，防止出现 run out of memory 的情况。

2、 重新修改代码，使 ISE 可以正常合成正确的信号。

3、 合理地将接口引脚与相应的信号连接起来。

实验测试

1、 这里提供了一个 ram 的 test 文件，测试的时候只要将像实验六一样将文件放在 ISE 里面编译就可以了。

2、 这里预先有一个 example 的执行文件 pipeline.bit, 也可以直接将这个文件放进 FPGA 板子进行运行。

3、 Example 文件里面的是汇编码，machine 文件里面的是机器码。

实验过程

1、 解决信号合成的问题

```
if(clk'event and clk='1') then
 if(enable = '1') then
   O<=data;
   temp:=data;
   else
    O<=temp;
   end if;
end if;
```

在 ISE 信号合成的时候，不能在时延判断语句中加入其他的变量的判断语句。如果加入的话会造成信号判断失败而合成失败，所以只能在时延被判断后才判断 enable。

2、 解决 run out of memory 的方法

由于 Spartan 3E 这个 FPGA 实验板的资源有限，所以这里的 MEMORY 的大小改成了 16BYTES 的大小。相应的寻址的变量的范围也要改变。特别要注意的是，所有的 memory 都必须初始化。

```
architecture bhv of data_mem is
type memory is array (0 to 15) of std_logic_vector(31 downto 0);
signal mem: memory;

begin
process(rst,data_in,enable,inenable,inaddr,clk)
variable temp:std_logic_vector (31 downto 0):="00000000000000000000000000000000";
  begin
    if (rst ='1') then
mem(0) <= conv_std_logic_vector(8519688,32);
mem(1) <= conv_std_logic_vector(8585225,32);
mem(2) <= conv_std_logic_vector(1245185,32);
mem(3) <= conv_std_logic_vector(589825,32);
mem(4) <= conv_std_logic_vector(8650763,32);
mem(5) <= conv_std_logic_vector(5505029,32);
mem(6) <= conv_std_logic_vector(8781836,32);
mem(7) <= conv_std_logic_vector(25165824,32);
mem(8) <= conv_std_logic_vector(11,32);
mem(9) <= conv_std_logic_vector(7,32);
mem(10) <= conv_std_logic_vector(0,32);
mem(11) <= conv_std_logic_vector(12,32);
mem(12) <= conv_std_logic_vector(33,32);
mem(13) <= conv_std_logic_vector(0,32);
mem(14) <= conv_std_logic_vector(0,32);
```

```
mem(15) <= conv_std_logic_vector(0,32);
data_out <=conv_std_logic_vector(0,32);
indata_out <=conv_std_logic_vector(0,32);
    else
           if(clk'event and clk ='0') then
            if(R_W='0' and enable ='1' ) then
               data_out<=mem(conv_integer(addr));
          end if;
        end if;
           if (clk'event and clk ='1') then
            if(R_W ='1' and enable  ='1') then
               mem(conv_integer(addr))<=data_in;
             end if;
        end if;
          if(clk'event and clk ='1') then
            if(inenable='1' ) then
               indata_out<=mem(conv_integer(inaddr));
               temp:=mem(conv_integer(inaddr));
          else
          indata_out<=temp;
               end if;
       end if;
   end if;
end process;
end bhv;
```

如果不初始化所有 memory，由于可能会获取到没有初始化的 memory，造成错误。

3、 引脚问题

NET "clock" LOC="G12";

NET "reset" LOC="C11";

NET "clock" CLOCK_DEDICATED_ROUTE = FALSE;

NET "reset" CLOCK_DEDICATED_ROUTE = FALSE;

Clock 和 reset 信号都是连接到 FPGA 板上的按钮上的。

NET "regnum0" LOC="N3";

NET "regnum1" LOC="E2";

NET "regnum2" LOC="F3";

NET "regnum3" LOC="G3";

NET "regnum4" LOC="B4";

NET "regnum5" LOC="K3";

NET "regnum6" LOC="L3";

NET "regnum7" LOC="P11";

Regnum 信号对应 **FPGA** 板上的 **switch** 开关。向上为“**1**”，向下为“**0**”。从左到右分别是“**0-7**”，对应零号到七号寄存器。某个开关向上，就会在 **LED** 灯看到相应寄存器的后八位值了。注意，多个开关向上时，以最低位为准。

NET "regvalue(0)" LOC="M5";

NET "regvalue(1)" LOC="M11";

NET "regvalue(2)" LOC="P7";

NET "regvalue(3)" LOC="P6";

NET "regvalue(4)" LOC="N5";

NET "regvalue(5)" LOC="N4";

NET "regvalue(6)" LOC="P4";

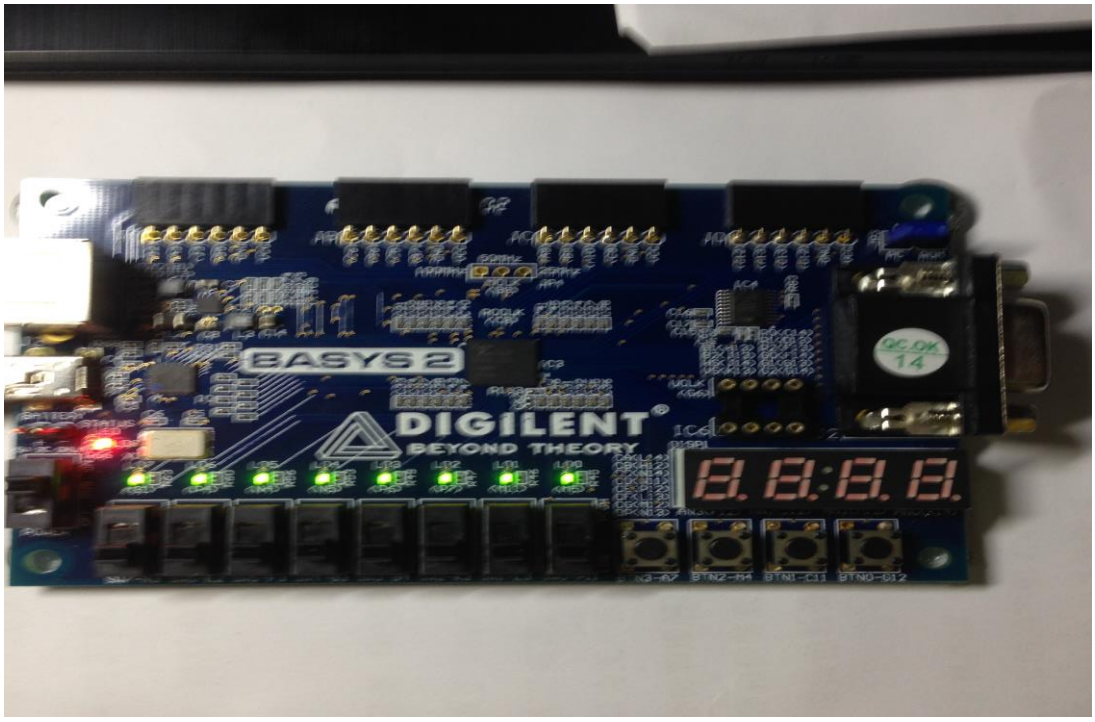
NET "regvalue(7)" LOC="G1";

Regvalue 代表相应寄存器的值。后八位在 **FPGA** 板上的 **LED** 灯上显示。灯亮的时候表示相应的位置为“**1**”。灯灭的时候表示相应的位置为“**0**”。

实验展示

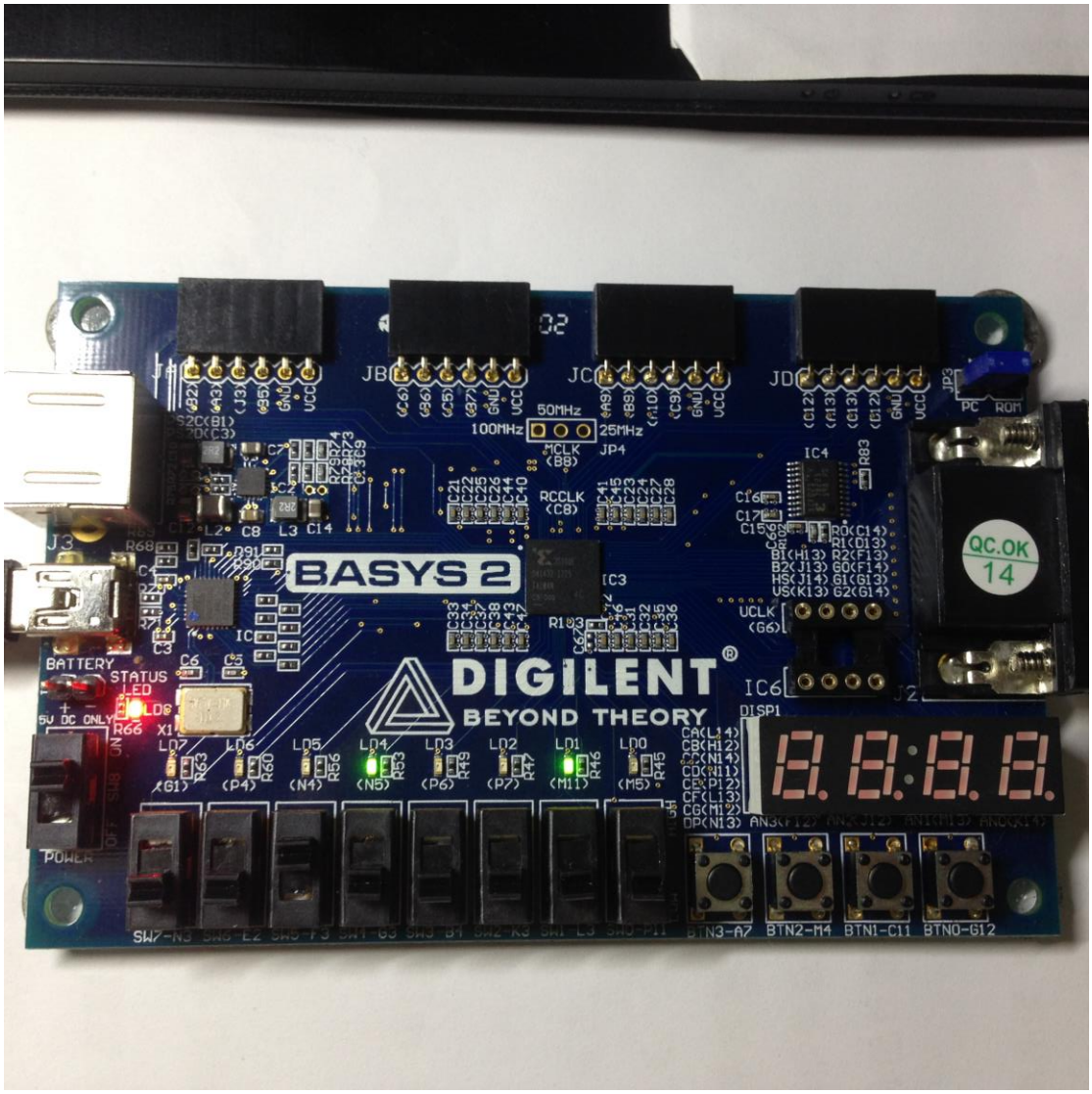
```
lw          0          2          mcand    //register2 = 18
lw          0          3          mplier   //register3 = 7
add         2          3          1        //register1 = 25
add         1          1          1        //register1 = 50
nand        1          1          4
done        halt                           //end of program
mcand       .fill      18
roy         .fill      0
you         .fill      0
```

reset



所有 switch 为 0 的时候让所有的指示灯亮起。

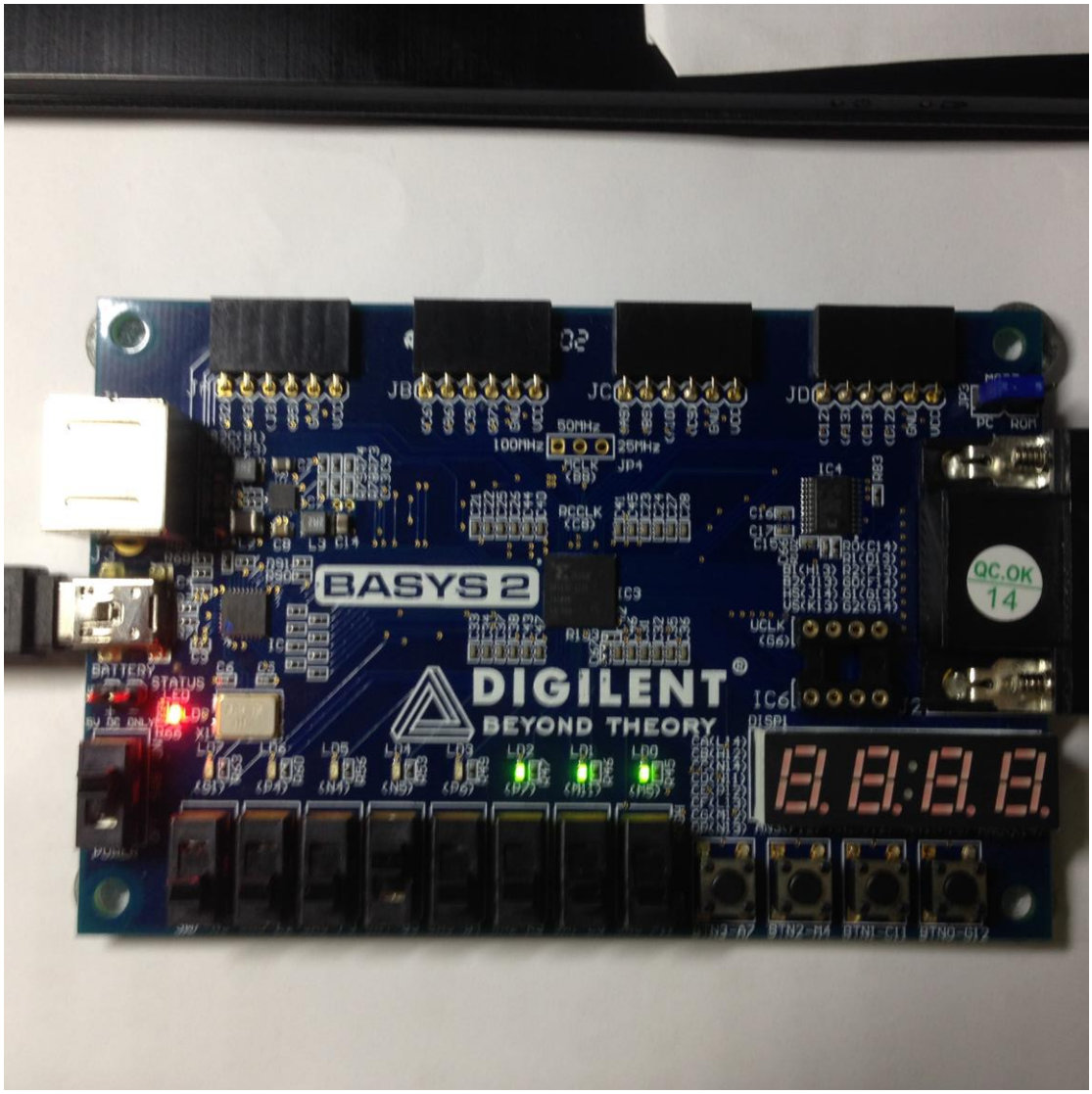
lw 0 2 mcand //register2 = 18



在按下 clock 按钮五次后，将 regnum2 置为“1”，可以看到相应的值为“10010” 即 18。

Regiset2=18 答案正确

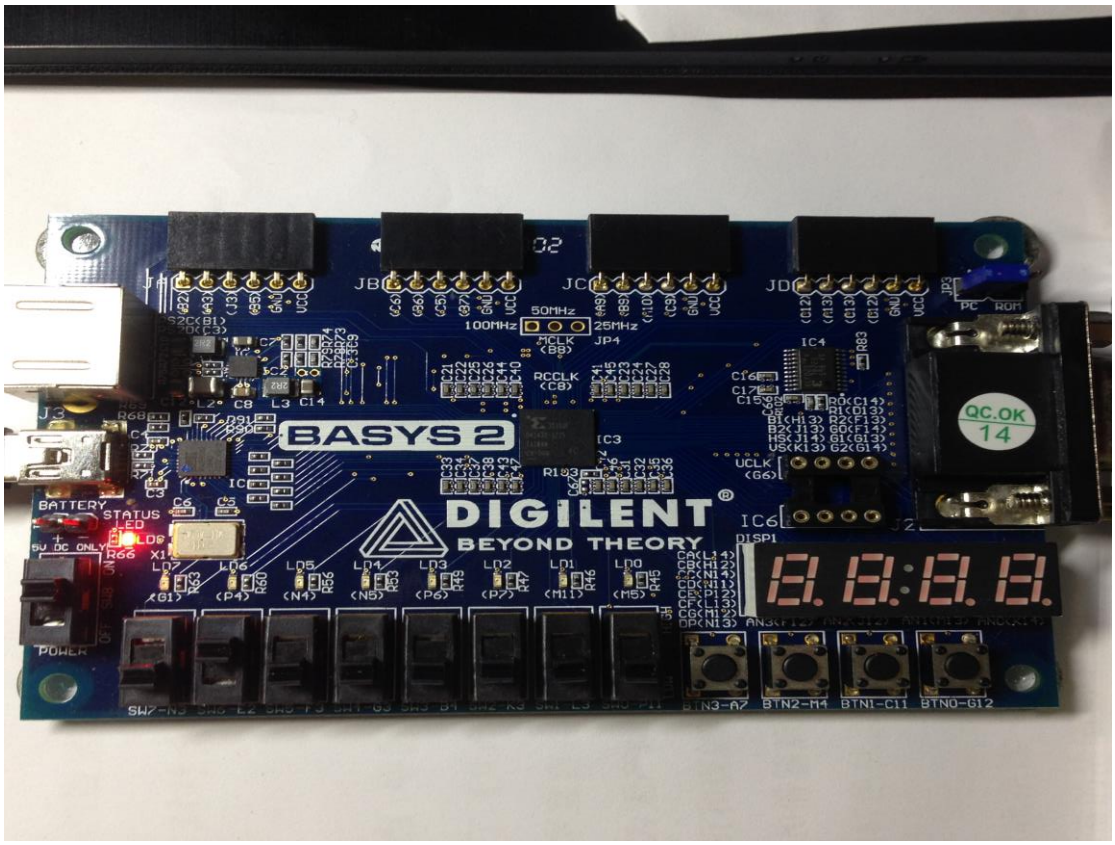
lw 0 3 mplier //register3 = 7



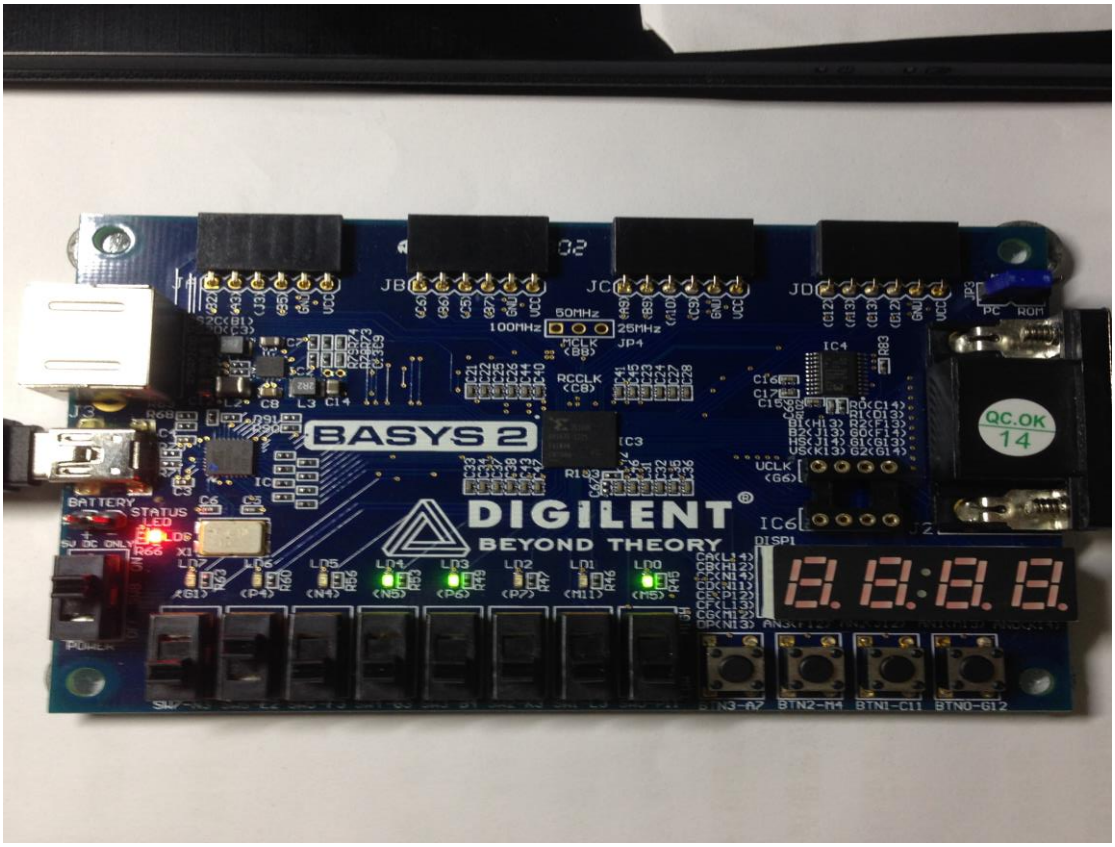
在按下 clock 按钮第六次，将 regnum3 置为“1”，可以看到相应的值为“111” 即 7 。

Regiset3=7 答案正确

add 2 3 1 //register1 = 25



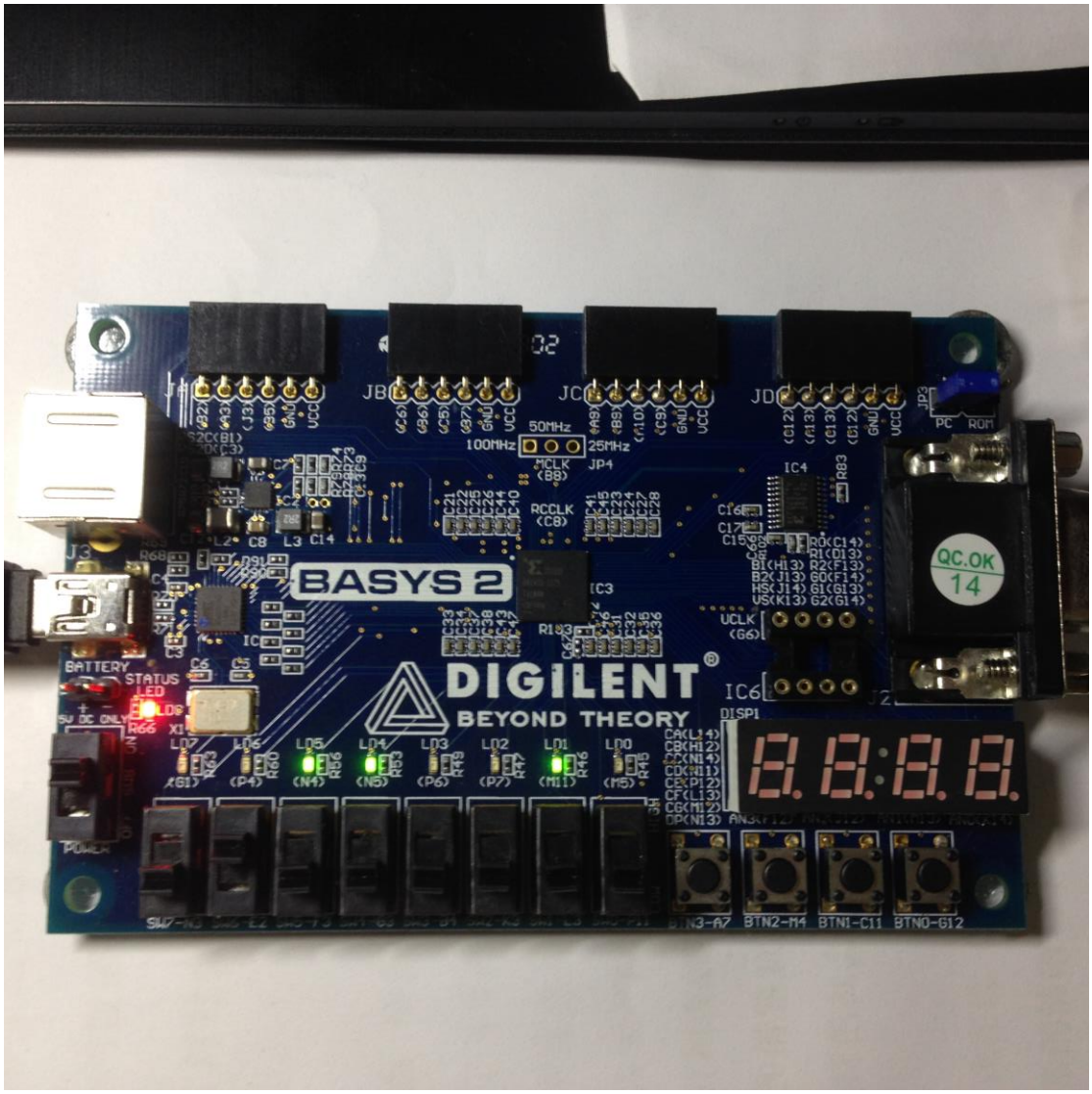
上图是第七次按 clock 按钮后的结果，将 regnum1 置为“1”，由于 load-use，因此这里必须拖延一个周期才能够正确读入结果。可以看出 register1=0 答案正确



因此，在第八次按 clock 按钮后，将 regnum1 置为“1”，可以看到相应的值为“11001” 即 25 。

Regiset1=25 答案正确

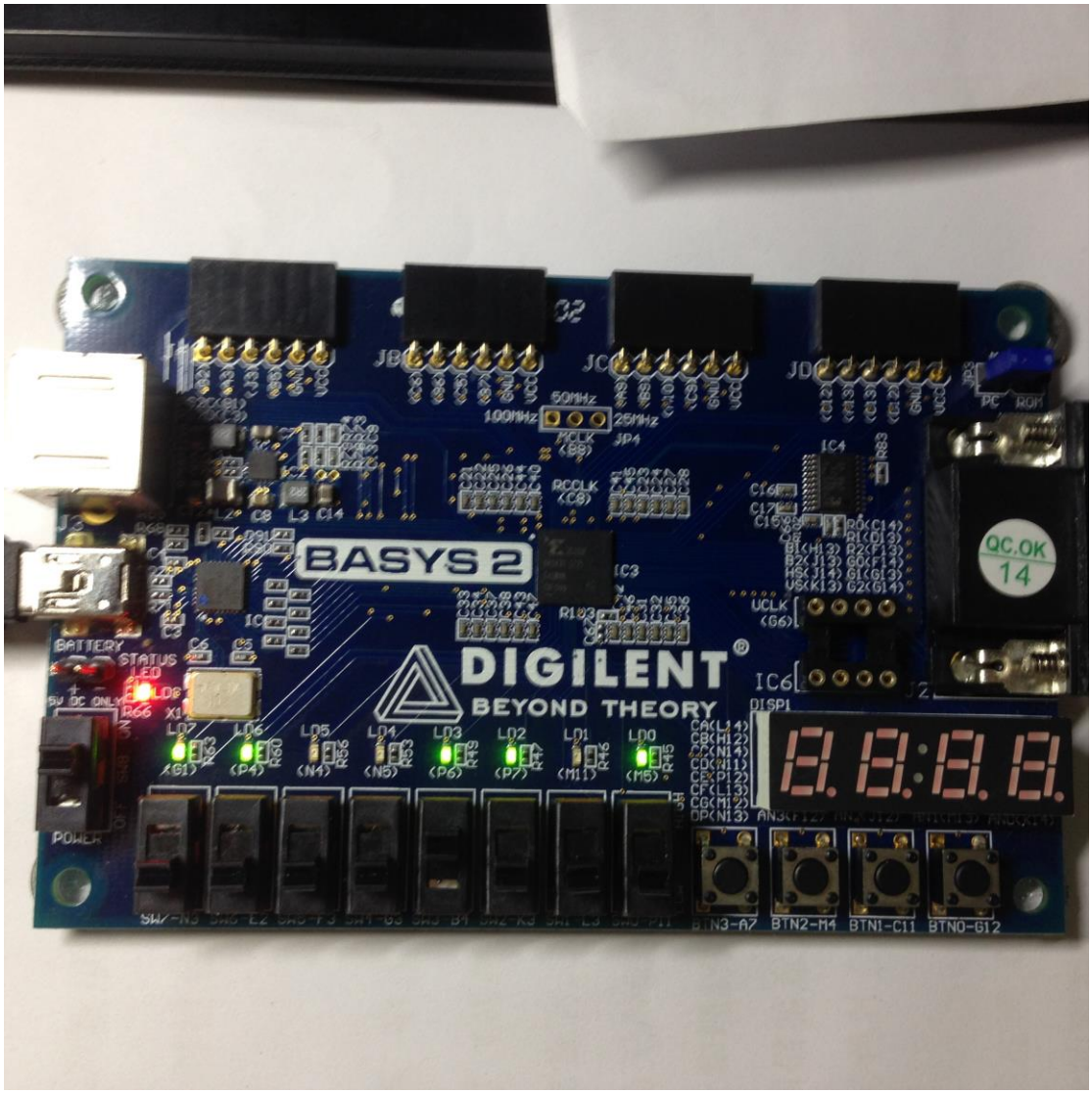
add 1 1 1 //register1 = 50



在第九次按 clock 按钮后，将 regnum1 置为“1”，可以看到相应的值为“110010” 即 50 。

Regiset1=50 答案正确

nand 1 1 4



在第十次按 clock 按钮后，将 regnum4 置为“1”，可以看到相应的值为“11001101”。

Regiset4 = ~（00110010&00110010）=11001101

答案正确

Example 正确

实验感想

实验比想象中要难很多。

首先是信号合成的问题，由于在 Modelsim 编译的时候呀一直没有遇到这个问题，而这个问题是 ISE 在合成的时候才出现的，所以第一次碰到这种问题也不清楚怎么做，不过后来通过调整 if-else 语句就解决了这个问题。

第二个问题是 run out of memory 的问题。 由于这个 pipeline 设计需要很多 component，因此在放入板子的时候必须要删除一些多余部分才能够在 map 那一步的时候正确将信号值映射到 FPGA 板上去。

解决了这两个问题就能成功做完这个实验了。这七个实验，从 assembler 到 simulator ， 到使用 VHDL 语言模拟 processor，在到放在 FPGA 板子上实现功能，感觉这学期的计组实验都学到了很多东西，从无知到逐步熟悉。有空应该多研究 VHDL，优化设计。