嵌入式系统结构与设计课程实验

实验报告 4

Lab 12 及 Lab 5 两次实验的实验报告

邱迪聪 / 11331262

2013/10/30

目录

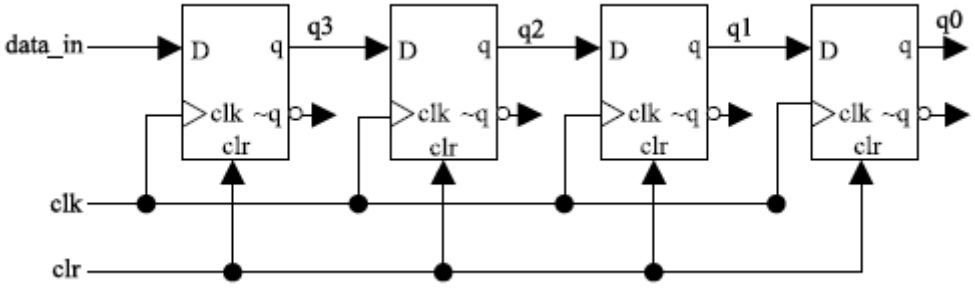
一、实验目的

- 熟悉使用 ISE 软件设计并仿真；
- 学会程序下载；
- 更深入地学习与使用 Verilog 语言来设计硬件电路。

二、Lab12 (4 位移位寄存器的设计与实现)

1、实现 4 位移位寄存器逻辑

实验题目已经给出了该移位寄存器的逻辑图，如下：



为了方便程序编写，首先需要实现一个 D 触发器的逻辑。在我编写的程序中，其模块名称为 DDF（D Flip-Flop）。虽然在该逻辑图中没有给出使能端 EN 输入，但为了完整起见，DDF 模块有 CLK, CLR, EN, D 四个输入，分别代表时钟、异步清零、使能端以及数据端。而输出只有一个 Q，表示当前的寄存数据输出。

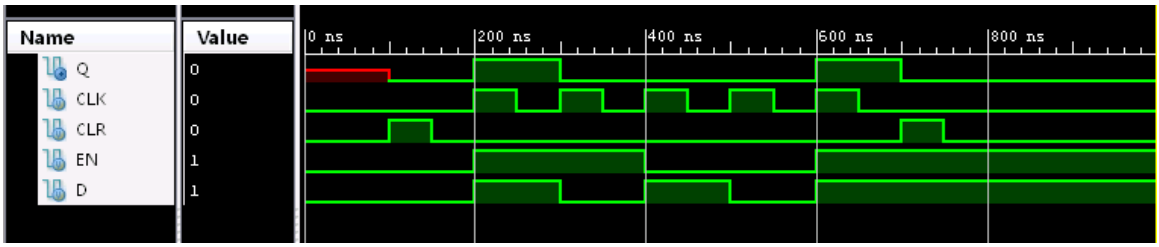
实现好该逻辑后，就可以按照如上逻辑图来组合成 4 位以为寄存器了。按照上图，4 位移位寄存器有 CLK, CLR, D 三个输入；输出以一个 wire[3:0] Q 数组表示。每个 D 寄存器首尾相接即可连接成需要的逻辑。

2、模拟运行

该模拟分为两个部分，一个是 D 触发器的模拟运行，另外一个是 4 位移位寄存器的模拟运行。

为了模拟 D 触发器的运行情况，模拟情况中选择了几个比较有代表性的情况来进行测试，分别模拟了 D 触发器的启动时刻、使能端控制情况、异步清零性能、触发效果是否如预期等多种情况。

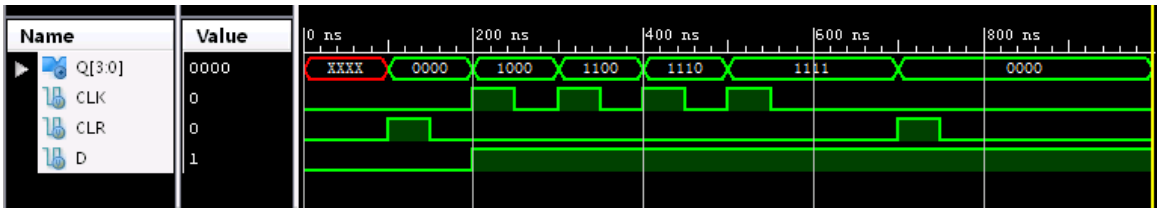
D 触发器的模拟运行如下：



模拟结果与 D 触发器的逻辑相符合。

对于 4 位移位寄存器，需要进行测试的情况与 D 触发器类似。模拟情况测试了启动情况、异步清零端口、移位寄存性能等。

模拟结果如下：



模拟结果与移位寄存器的逻辑相符。

3、综合、实现及程序生成

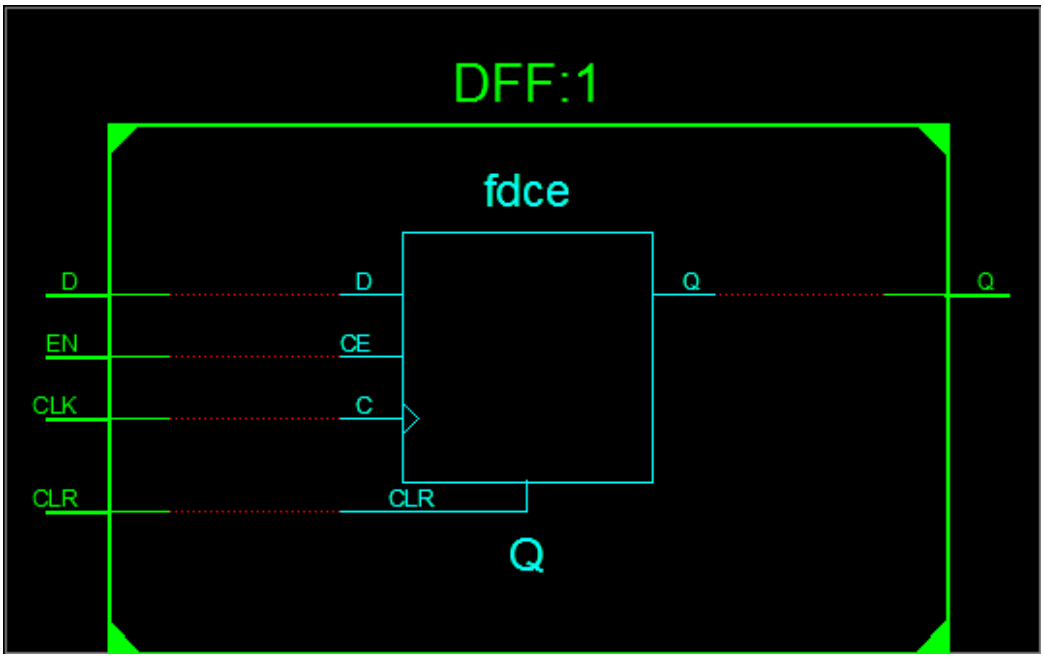
编写管脚约束，如下：

```
// Input: Data, CLK, CLR
NET "D" LOC = "T5"; // left most button
NET "CLK" CLOCK_DEDICATED_ROUTE = FALSE;
NET "CLK" LOC = "B8"; // central button
NET "CLR" LOC = "C9"; // down (BTND)

// Output: Q's
NET "Q[3]" LOC = "T11"; // left most LED
NET "Q[2]" LOC = "R11";
NET "Q[1]" LOC = "N11";
NET "Q[0]" LOC = "M11";
```

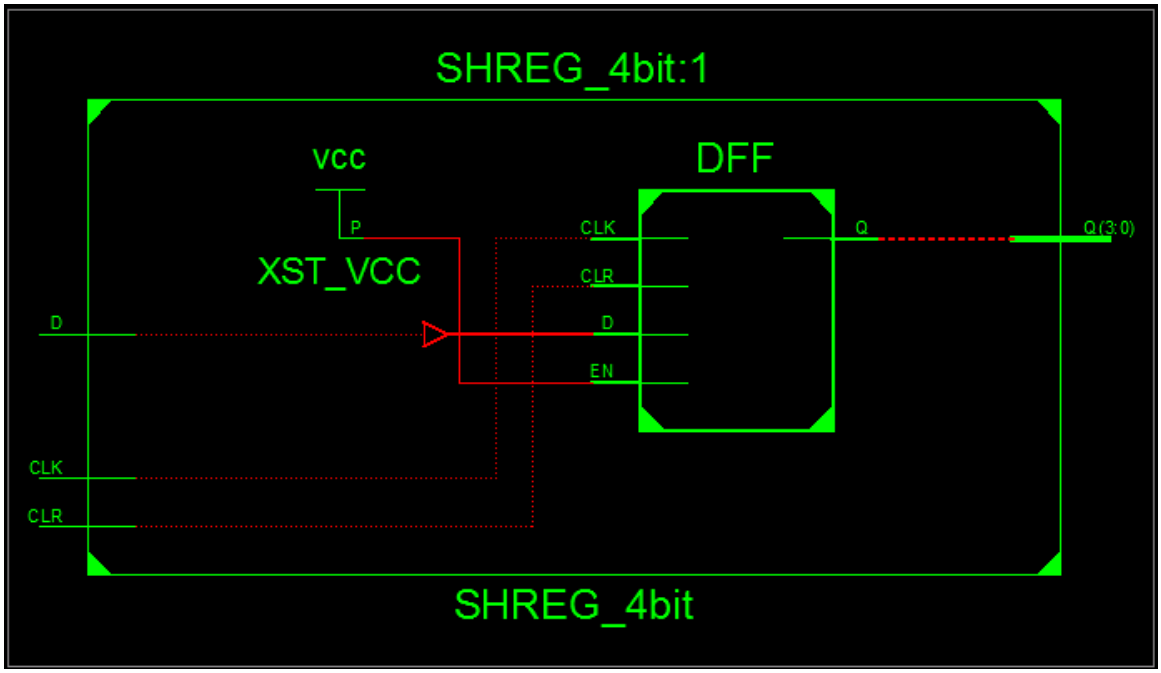
其中 `NET "CLK" CLOCK_DEDICATED_ROUTE = FALSE;` 这一句是用来把默认的时钟引脚接到按钮上面所需要特别声明的。把时钟引脚接到按钮上是为了更好地进行测试，并观察测试结果。

在经过综合后得到如下 RTL 图。

以下为 D 触发器的 RTL 图：



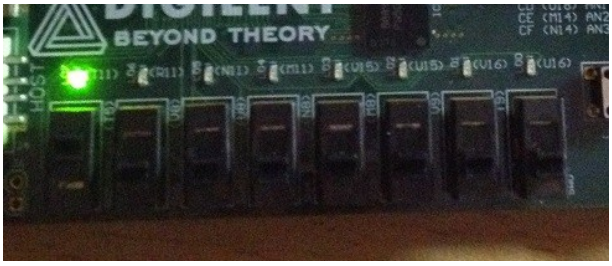
以下为 4 位移位寄存器的 RTL 图：

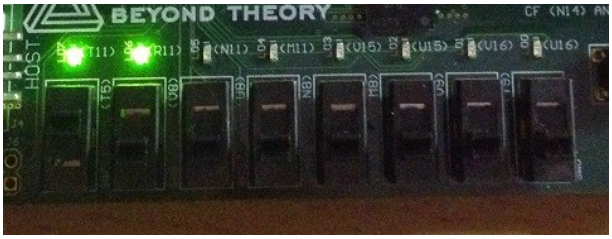


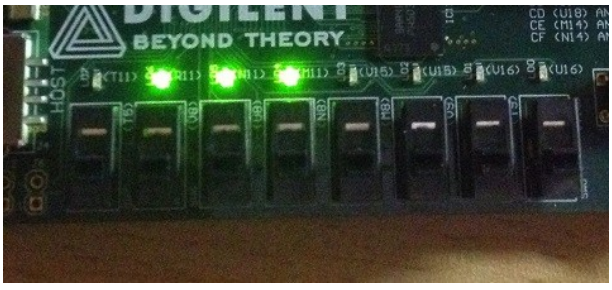
之后进行实现与生成程序步骤均通过。

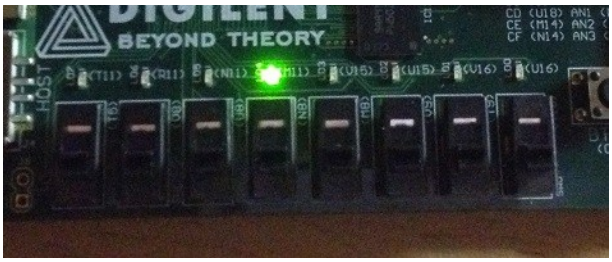
4、真机测试

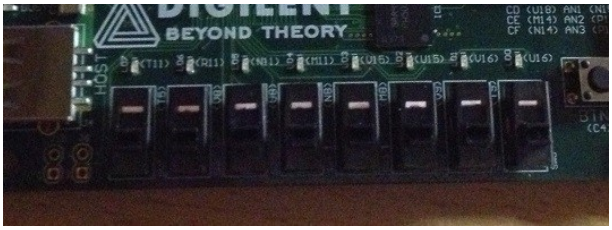
测试移位串：00001110







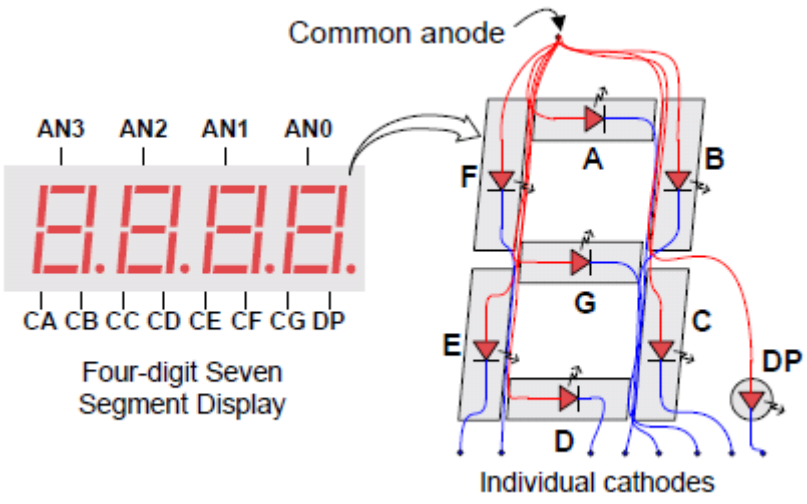




三、Lab5 (7 段显示管的设计与实现)

1、实现 7 段显示管的逻辑

7 段显示管即数码管，其原理图如下：

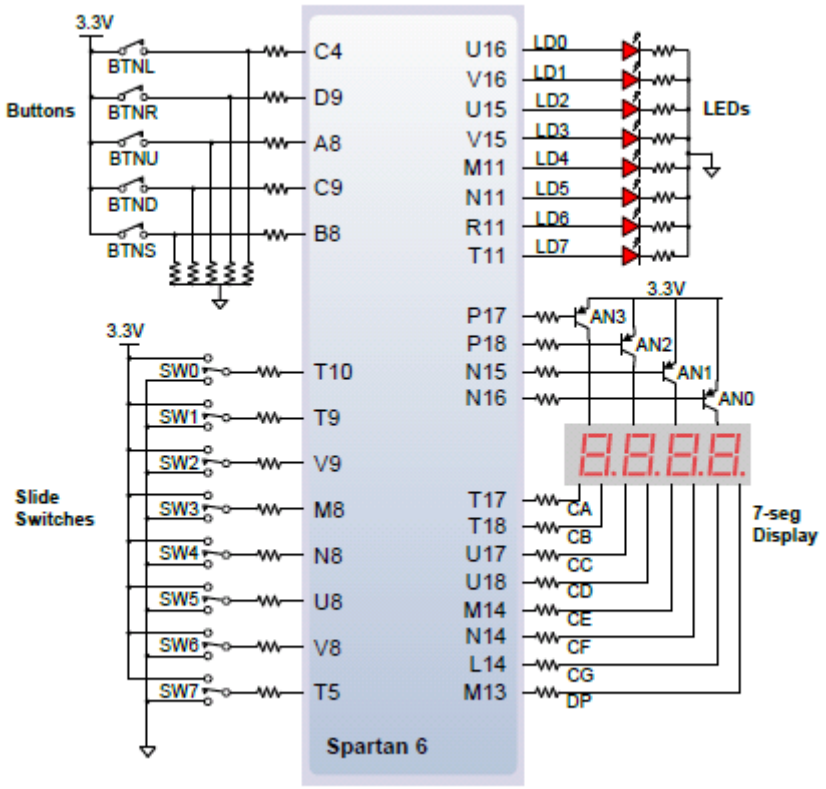


从参考文档中的这个图可以知道，该数码管是共阳极的。故在设置某段亮的时候，应该设为低电平，而非高电平。

由于第 2 次实验的时候已经做过 BCD 到 7 段数码管表示的转换了，且为了更好的表示所有输入情况，实际上也已经处理了 16 进制的所有输入情况。实际上是一个 HEX 到 7 段数码管的转换模块。直接使用该模块即可实现对一个数码管的输出了。

可是要实现两个数码管同时显示，且显示内容不同。则需要不断交换被激活的数码管，以达到肉眼看起来好像是同时显示的样子。实际上是在不断地扫描，当扫描到第一个数码管的时候输出第一个数码管对应的内容到数码管阵列；当扫描到第二个数码管的时候输出第二个数码管对应的内容到数码管阵列。这样就可以实现同时显示了。

其中根据参考文档提供的内容，数码管阵列的激活方式可以从以下逻辑图中看出：

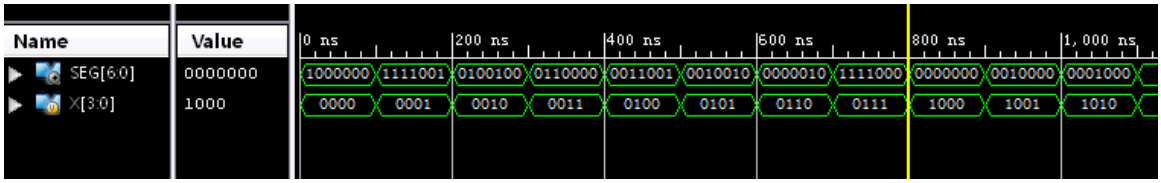


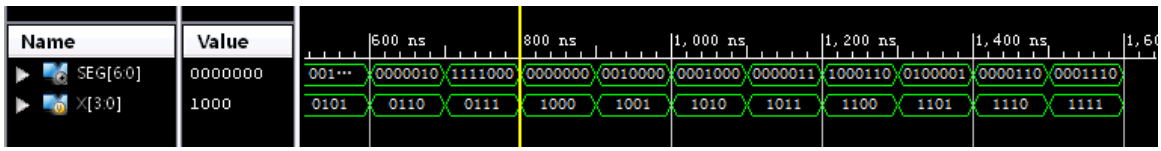
故我实现了一个模块，里面通过三元操作符以及时钟来不断选择需要显示的内容以及激活的数码管。使用 `(CLK)?(HexA):(HexB)` 可以避免使用必须在时序逻辑里面出现的 if 语句。

2、模拟运行

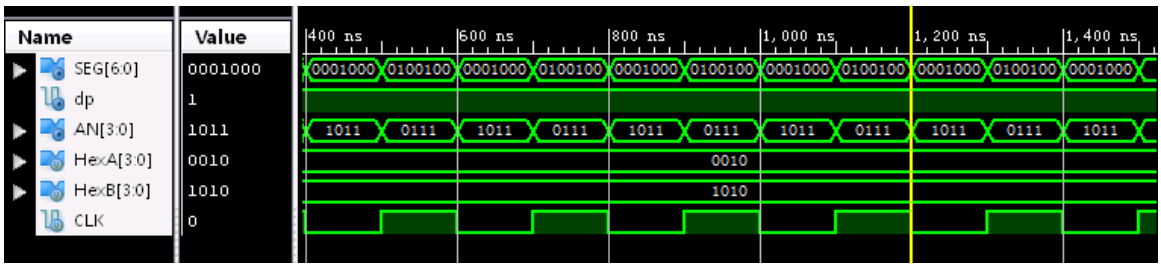
模拟运行分为两个部分，第一部分是 HEX 转数码管的模块的模拟运行，另外一部分是 7 段数码管阵列的整体模拟运行。

对 HEX 转数码管模块，我做了完整性测试，测试结果如下：





对于 7 段数码管阵列的测试，主要是测试数码管是否能被交替激活，然后显示出好像是同时出现的样子。测试结果如下：



其中测试用的两个十六进制数分别为 2 和 A。

3、综合、实现及程序生成

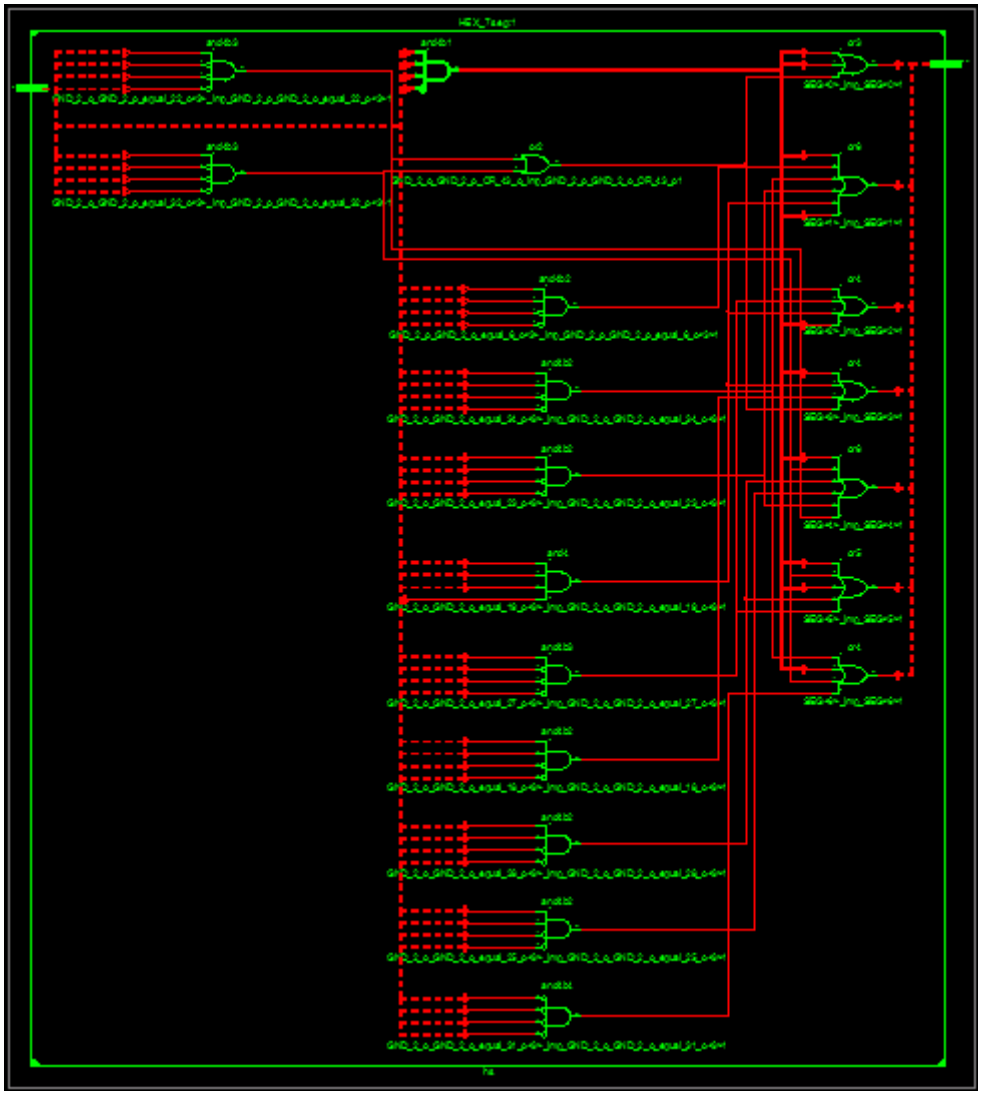
管脚约束如下：

```
// Input: HexA
NET "HexA[3]" LOC = "T5"; // left most button
NET "HexA[2]" LOC = "V8";
NET "HexA[1]" LOC = "U8";
NET "HexA[0]" LOC = "N8";

// Input: HexB
NET "HexB[3]" LOC = "M8";
NET "HexB[2]" LOC = "V9";
NET "HexB[1]" LOC = "T9";
NET "HexB[0]" LOC = "T10"; // right most button

// Clock -> CLK
NET "CLK" LOC = "B8";

// Output: 7-Segment Display
NET "SEG[0]" LOC = "T17"; // CA
```

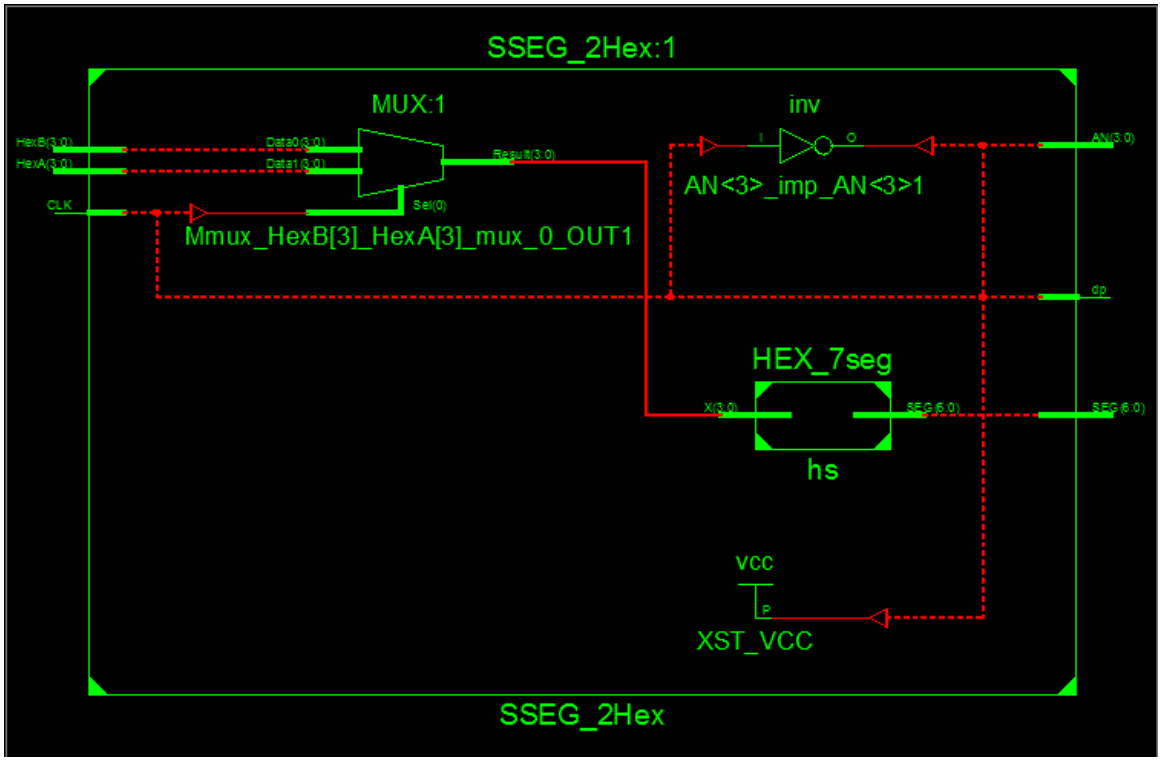
```
NET "SEG[1]" LOC = "T18"; // CB
NET "SEG[2]" LOC = "U17"; // CC
NET "SEG[3]" LOC = "U18"; // CD
NET "SEG[4]" LOC = "M14"; // CE
NET "SEG[5]" LOC = "N14"; // CF
NET "SEG[6]" LOC = "L14"; // CG
NET "dp" LOC = "M13"; // DP
NET "AN[0]" LOC = "N16"; // AN0
NET "AN[1]" LOC = "N15"; // AN1
NET "AN[2]" LOC = "P18"; // AN2
NET "AN[3]" LOC = "P17"; // AN3
```

有一个比较麻烦的地方是，我经过多翻查询，还是弄不懂 Spartan-6 的时钟是要如何使用的，所以这里只好先使用一个按钮代替时钟。

以下为综合后生成的 HEX 转数码管模块的 RTL 图：



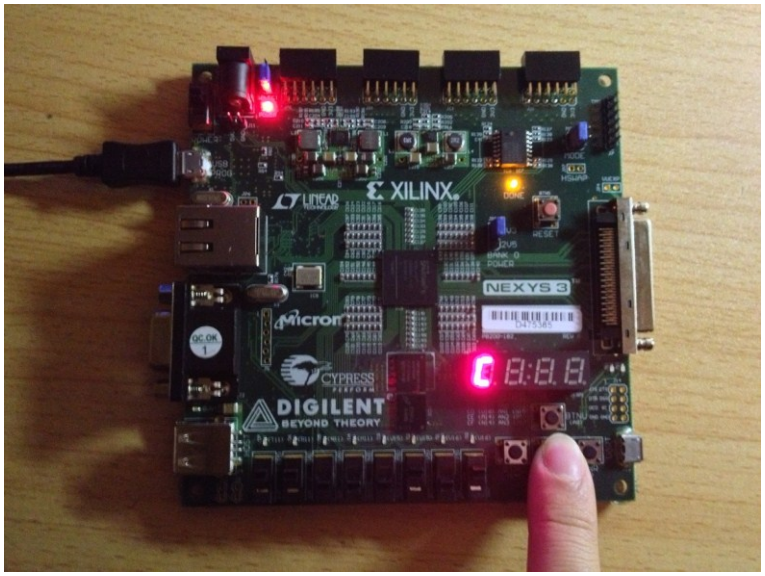
以下为 2 位十六进制数码管的 RTL 图：

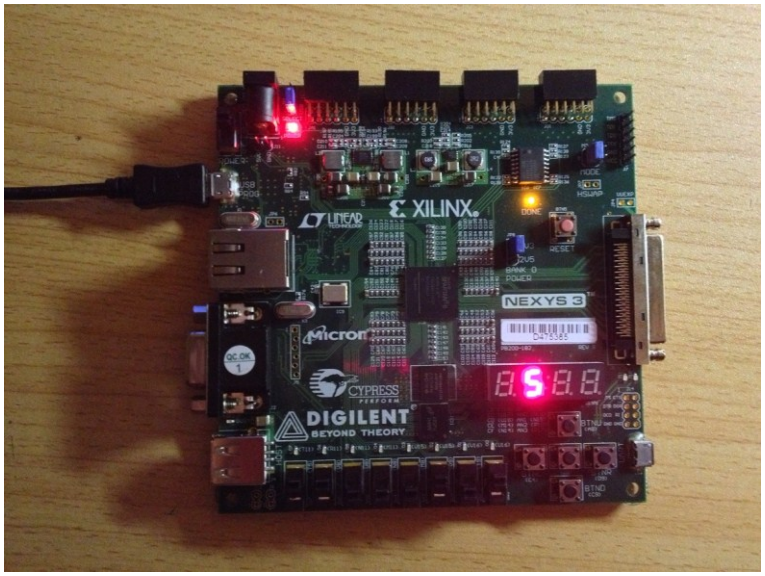


之后进行实现与程序生成均成功。

4、真机测试

使用 0xC5 进行测试，结果如下：





四、实验感想

本次实验虽然只有两个内容，可是却也非常需要时间和精力去做，本次做到凌晨 5 点半了。不过话说回来，如果能够得到一些额外的知识还是非常好的。

这次实验第一项是 4 位移位寄存器。这个实验其实难度不大，只要逐步地去实现，就可以很顺利地做出来了。可是这个实验却有着比较大的意义，因为它是第一个开始真正进入时序电路的实验。之前的电路虽然我有用到时序电路部分去简化代码的编写，可是都可以通过其他的组合逻辑电路实现。而这次这个寄存器，是非常明显的时序电路的内容。而进入这个部分就意味着，之后会需要开始考虑关于时间以及冲突的问题了。

第二项的实验是之前做过其中一部分的 7 段数码管实验。之前没有要求用到开发板上面的数码显像管，可是我之前的实验为了显示美观，其实也用过了一下。不过上次用的时候弄来弄去还是不太清楚要如何指定激活某一个数码管，而是直接就全部都显示了。这次有些改进的是我知道了该如何激活某个具体的数码管，而且让其他的都变黑了，这个是有一定进步的。

这次需要同时显示出两个十六进制的数字，其实我们很早之前就知道了在数码管上要干这个事情，实际上是不断地去扫描每个数码管，然后再将对应显示的数据传给数码管阵列，然后交替显示。看起来就好像是真的同时显示的样子了。然后这个功能模块也都做出来了。

我想不到是，居然这次卡在了时钟上面。我上网找了许多资料，然后也知道在这个板子上要用时钟的话，实际上是用一个叫 DCM，即时钟管理器的东西。可惜无奈怎么找也找不到一份适合初学者入门的时钟使用教材。全部都是非常高深的内容，看得眼花缭乱。所以最终还是只能把时钟引脚接到一个按钮上，先将就着实现它，然后待知道时钟如何调用的时候再进一步完善吧。

邱迪聪

2013 年 10 月 16 日

附录 1：附件列表

Lab12 SHREG_4bit

\ SHREG_4bit.v

\ SHREG_4bit _ctr.ucf

\ SHREG_4bit _test.v

\ DFF.v

\ DFF_test.v

Lab5 SSEG_2Hex

\ SSEG_2Hex.v

\ SSEG_2Hex_ctr.ucf

\ SSEG_2Hex_test.v

\ HEX_7seg.v

\ HEX_7seg_test.v

其中 X.v 文件为程序模块文件，X_ctr.ucf 为管脚约束文件，X_test.v 为模拟测试代码文件。