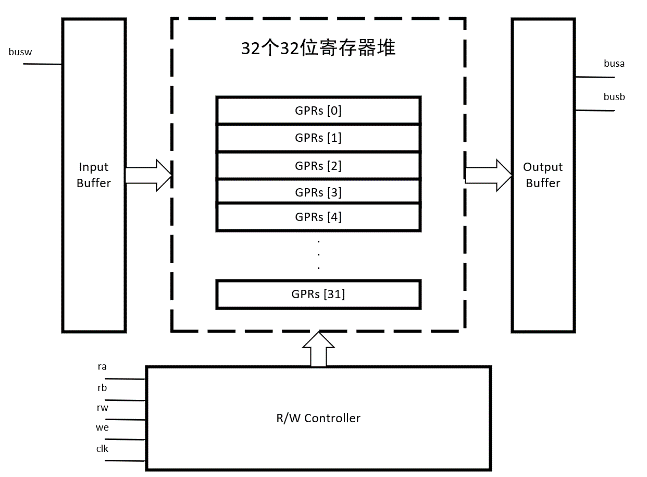
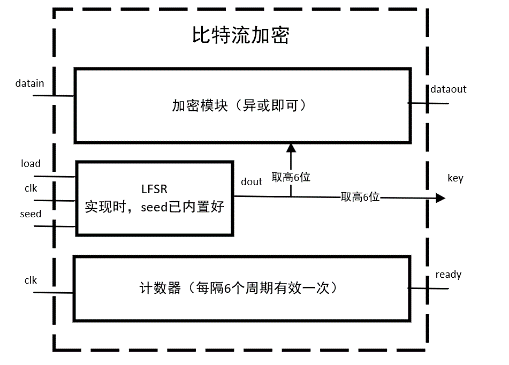
实验二 时序逻辑电路

## 一、系统设计

本次实验实现了三个时序逻辑电路：寄存器堆，比特流加密，数字时钟。其中寄存器堆是简单的时序逻辑，只使用了时钟驱动寄存器赋值，具体框图见图1.1。比特流加密的关键模块是LFSR线性移位寄存器生成的密钥串，具体框图见图1.2。数字时钟的关键模块是多个计数器的级联形成的输出信号，同时由于数字时钟集成了众多的功能，对于每个状态的控制信号需要考虑状态机来控制，具体框图见图1.3。

图1.1 寄存器堆框图

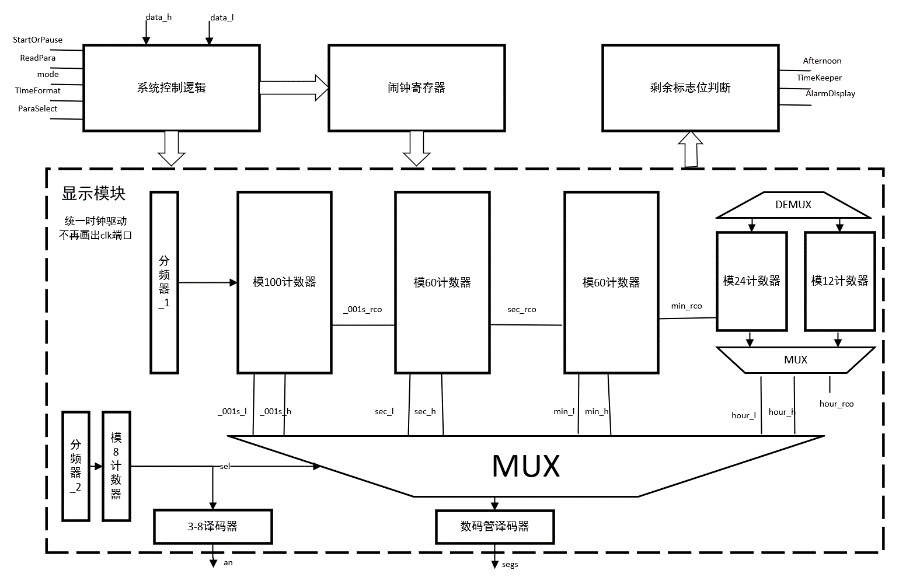
图1.2 比特流加密框图

图1.3 数字时钟框图

实验根据lab2指导中的步骤完成，寄存器堆的实现较为简单，不赘述。比特流加密先完成移位寄存器模块，数字时钟首先调用lab1工程中的数码管驱动模块，然后完成多个计数器模块，最后进行系统集成设计，主要工程量在于控制信号的状态机。

## 二、实现细节

由于端口定义在实验指导中已经给出，此处不再赘述

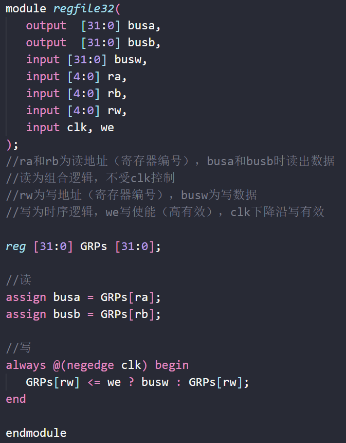
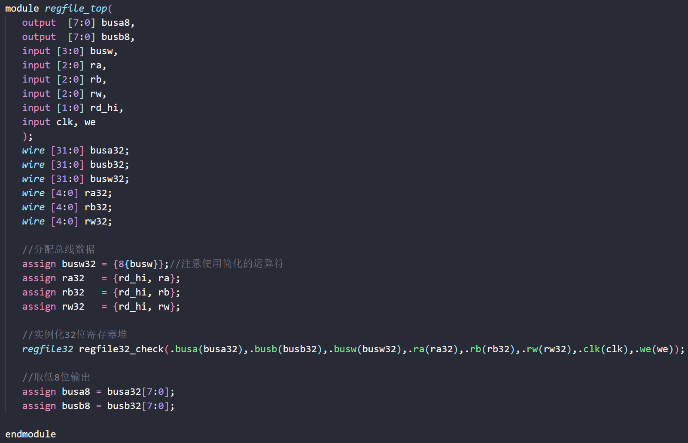
寄存器堆实现并不困难，创建存储单元后直接根据输入信号访问和读写即可，注意在分配总线数据时的“重复八次”可以用运算符轻松实现，见图2.1

图2.1 寄存器堆实现细节

加密模块也相对简单，此处不贴图展示。重点讲讲集成了多功能的数字时钟模块的一些设计。

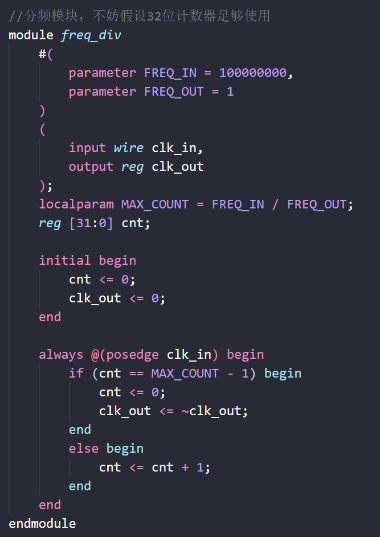
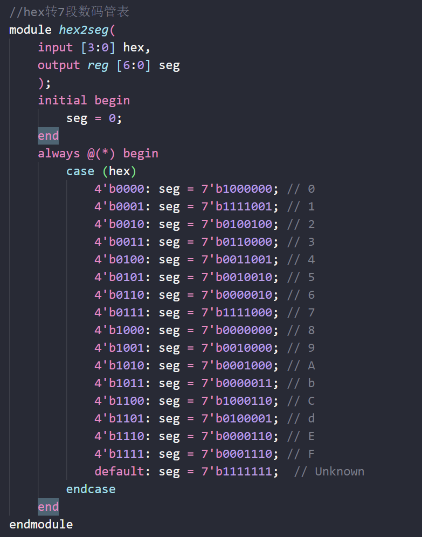
在设计数字时钟模块时，对数码管译码器和分频器模块做了通用的设计，见图2.2，方便未来的更多工程调用。

图2.2 通用模块设计

此外原本想设计通用的计数器模块，但是由于数字时钟的计数器要求与通用的计数器相比更苛刻，例如希望输出的是BCD码而非二进制数，因此单独设计了模100、模60、模24计数器，附带了根据mode不同进行倒计时，正计时的功能。以功能最多的模24计数器为例，见图2.3，截取模24计数器的主要代码。

由mode和TimeFormat控制计数状态，其中load和en信号由控制系统给出。

图2.3 模24计数器设计片段

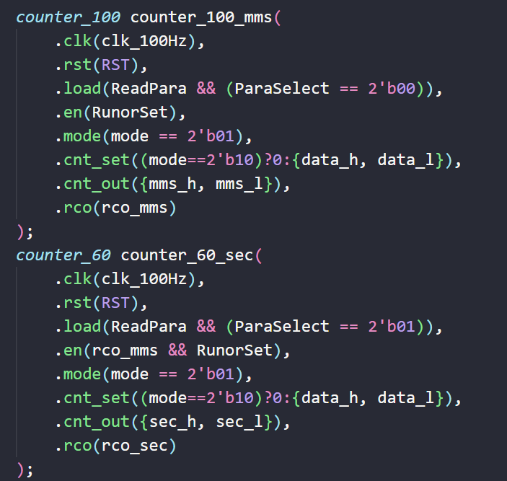
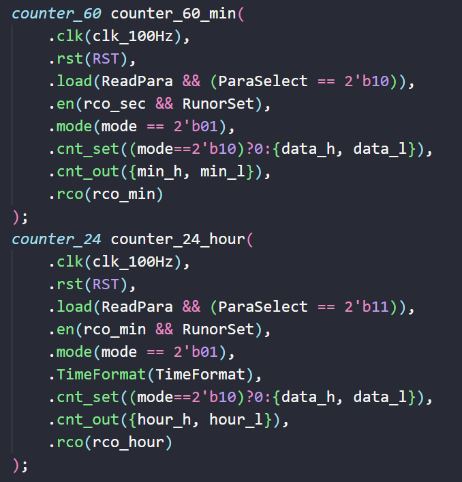
在进行实例化时，传入的端口列表如图2.4所示。

图2.4计数器实例化的端口列表

其余的标志位判断直接按照功能描述判断即可，注意考虑每个模式下的标志位判断，此处不再提供细节代码，参考附件。

## 三、实验结果

3.1 波形仿真

3.1.1 寄存器堆仿真波形

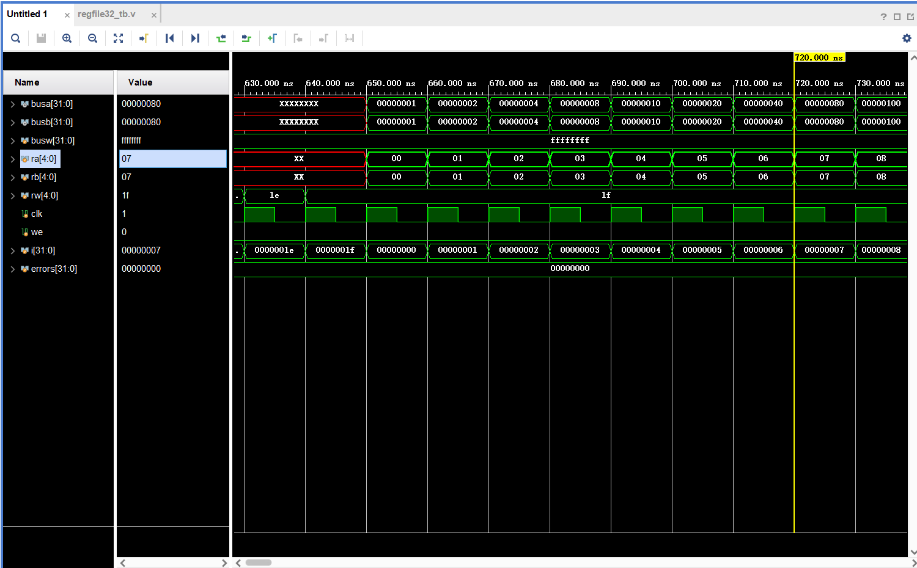
波形仿真结果见图3.1.1，可以发现在650ns前，一直在向reg [rw]中写数据，写入的数据位testbench中的i，在650ns后，开始一次读取ra和rb地址中的数据。由于i是循环的，因此此时的i正好应该是对应地址里读出的数据，根据busa，busb和i的结果对比，发现成功读出了存在对应地址中更多数据。

图3.1.1 寄存器堆仿真波形

3.1.2 比特流加密波形

波形仿真结果见图3.1.2，当计数到第一个6时，即第六个周期时，ready输出有效，对应的密钥为000010，这与lab2指导的结果时相同的，因此LFSR的输出是正确的。在加密部分，首先dataout只在ready为高时输出，其他时刻为X符合要求。另外dataout应等于datain和key异或，即01000000和00000010异或，结果为01000010，对应16进制的42，因此输出的结果也成功加密。

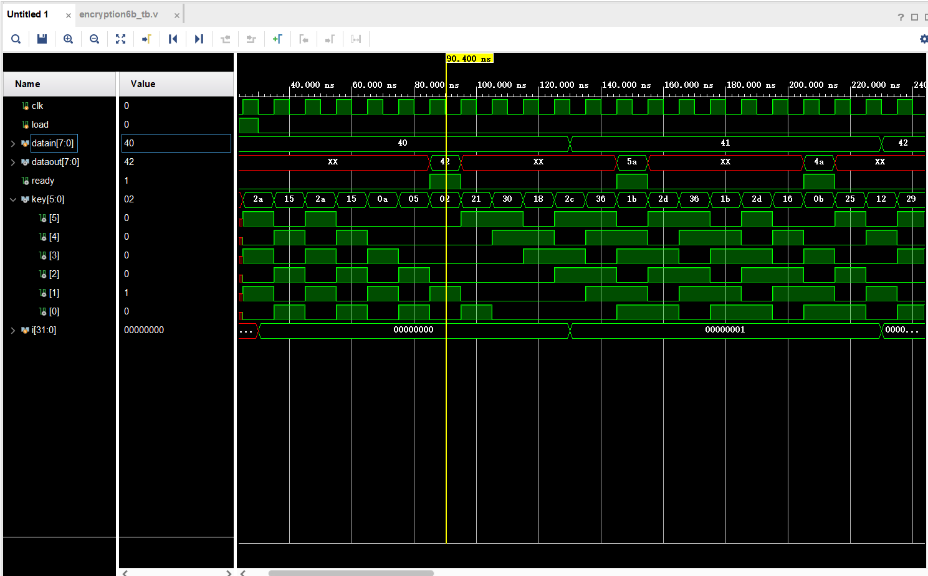
可以在后面的第12、18周期看到加密结果，也同样符合要求。

图3.1.2 比特流加密仿真波形

3.1.3 数字时钟波形

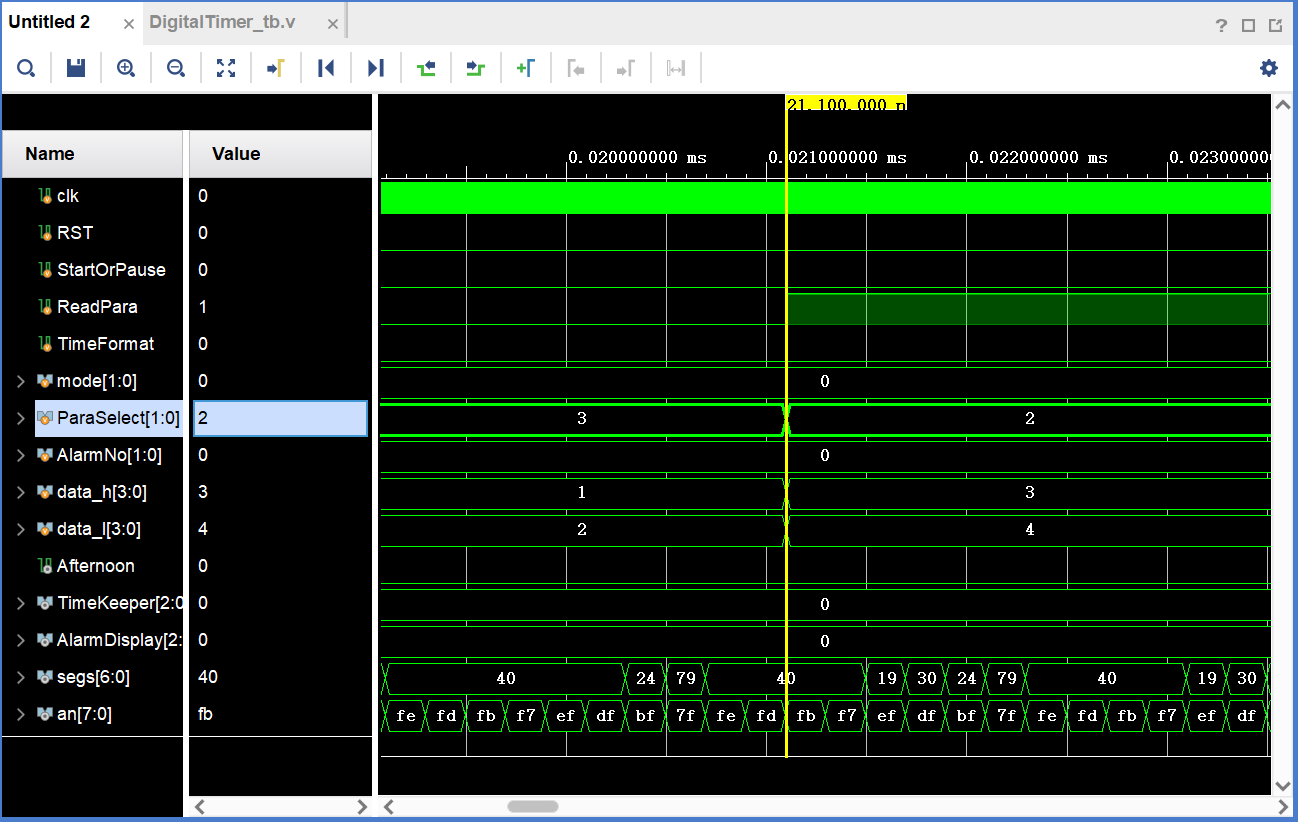
由于整个系统功能复杂，这里仅测试时钟置数功能，其余的现象均可在上板测试中看到。如图3.1.3， 仿真结果如下：在ReadPara为高时，跟据不同的ParaSelect可以正确的设置和显示，如图3.1.3所示，ParaSelect从3切换为2时，设置了秒。

图3.1.3 数字时钟时间设置仿真波形

3.2 上板测试

3.2.1 寄存器堆上板结果

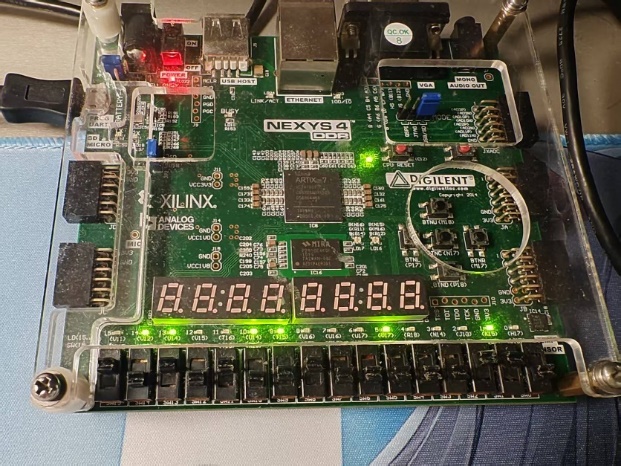
见图3.2.1，向寄存器堆的01010号和01000号中分别写入了4’b0110和4’b0010。如图所示，目前的rb为010，ra为000，高位相同都是01，读出的8位数据分别为01100110，00100010，符合预期结果。

图3.2.1 寄存器堆上板测试结果

3.2.2 比特流加密上板结果

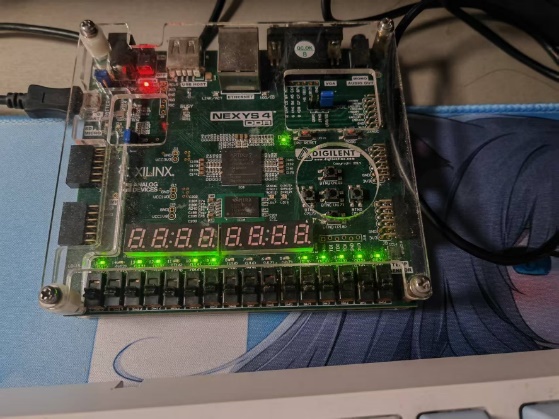
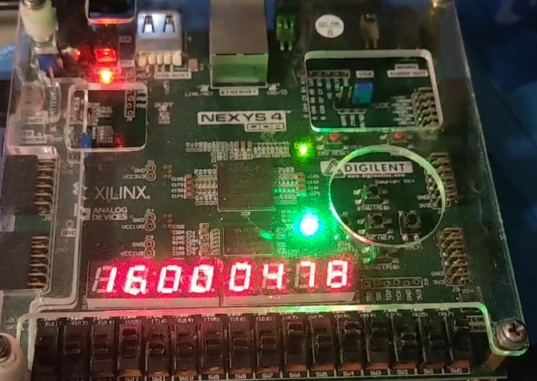
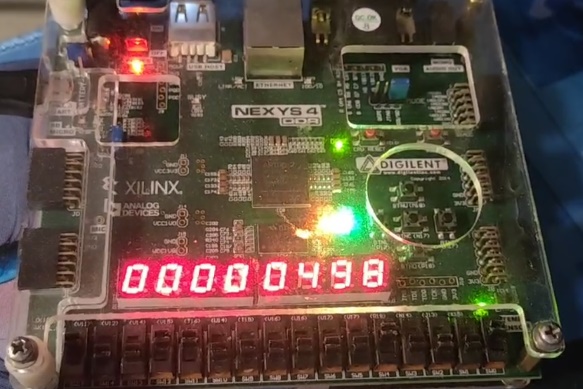
见图3.2.2，LED15量，代表密钥完成，为011011，与第12周期的密钥结果相符合。输入数据为拨码结果00000100，异或结果为00011111，和LED输出结果相符合。

图3.2.2 比特流加密测试结果

3.2.3 数字时钟上板结果

见图3.2.3，图3.2.4，图3.2.5分别代表了时钟模式下的整点报时；时钟模式12小时制的下午与闹钟报警；倒计时；

图3.2.3 时钟报时测试结果

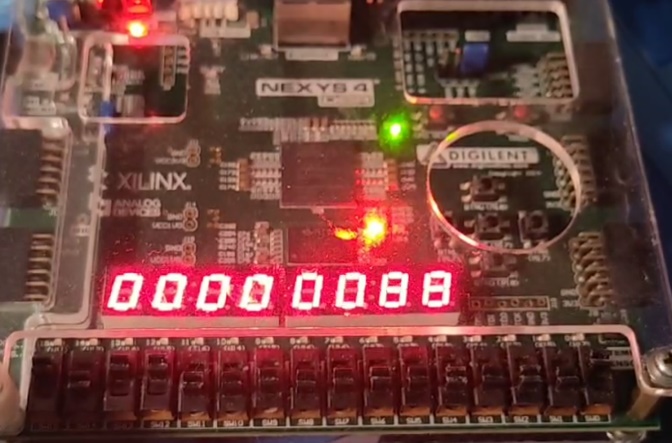
图3.2.4 12小时制测试结果

图3.2.5 倒计时测试结果

四、反思与总结

4.1 挫折与进步本部分包含遇到的问题、解决方法、实验感想

时序逻辑电路主要由寄存器变量进行运算，因此一定要注意寄存器变量只能在always块中进行赋值，并且在时序逻辑的always块需要进行非阻塞赋值。最后一个时钟模块，花了三天才测试完，在这里给出一些经验和教训：

1. verilog的代码是没有顺序的，且每个寄存器只能在一个always块中赋值，因此在写代码之前一定要构思好某个寄存器在所有情况下的取值情况，再上手写代码。

2.一定要初步检查一下所有的变量是否在正确的地方被赋值，阻塞赋值和非阻塞赋值有没有写在一个块内，等基本的语法错误。一次综合的时间很久，不要因为低级的错误浪费时间。

4.2 思考题

1. 分析32个32位的寄存器堆占用的逻辑片资源

通过时序逻辑驱动的寄存器堆应该只需要32\*32 = 1024个D触发器。寄存器也可以用MUX的组合逻辑实现，那么五级的二选一选择器构成32选1选择器，然后将32个相同的选择器构成32位数据，这样需要大量的LUT资源。

2. 分析64位移位寄存器的时序性能和资源占用情况；并通过资料查找到其他的生成LFSR的反馈公式。

只需要64个触发器即可实现，时序性能取决于LFSR的计算公式，多个异或的延迟，以及触发器本身的延迟，时钟频率由这二者一起决定。

LFSR反馈公式属于密码学的内容，其反馈公式可以写*Snew*​(0)=⨁*i*∈*P*​*S*(*i*)

其中P的选取取决于具体的需求，例如本次实验中为0，1，3，4.

3. 数字时钟中是如何实现倒计时和毫秒计时器功能

倒计时配置计时器进行倒数即可，毫秒需要将时钟分频为1kHz。

4. 如何实现寄存器堆中0号寄存器的值始终为零

将0号寄存器的编号作为写使能的判断条件之一，例如将写使能we和写地址waddr做与运算，因此waddr == 0会禁止写操作。只需要在最初将0号寄存器的值赋为零即可。

通用的来说，可以将传入端口写使能改为.we(we & (waddr != addr))，其中addr为希望不被修改的寄存器的地址。