

# **数字逻辑实验报告（4**）

|  |  |  |
| --- | --- | --- |
| **数字逻辑实验4** | | |
| **斐波那契(Fibonacci)数列计算器设计** | **成绩** |  |

评语：（包含：预习报告内容、实验过程、实验结果及分析）

教师签名

**姓 名： 王明明**

**学 号： U201714726**

**班 级： CS1705**

**指 导 教 师： 何云峰**

**计算机科学与技术学院**

**2019 年 6 月 5 日**



**数字逻辑实验报告**

斐波那契(Fibonacci)数列计算器设计

1、实验名称

斐波那契(Fibonacci)数列计算器设计。

2、实验目的

要求使用合适的逻辑电路的设计方法，通过工具软件logisim进行斐波那契(Fibonacci)数列计算器设计和验证，记录实验结果，验证设计是否达到要求。

通过斐波那契(Fibonacci)数列计算器的设计、仿真、验证3个训练过程，掌握数字逻辑电路的设计、仿真、调试的方法。

3、实验所用设备

Logisim2.7.1软件一套。

4、实验内容

斐波那契(Fibonacci)数列中每项数值都是其两个直接前项的和，其生成规则如下公式1所示：

（公式1）

**（1）求Fibonacci数的矩阵算法**

首先，对于数列的初始条件对应公式2的矩阵运算：

（公式2）

更一般化地，有公式3：

（公式3）

所以，根据递推关系可以得到公式4：

（公式4）

由公式4可推出，。

因此，对求斐波那契数列的第n项的问题，可以转化为对一个二维矩阵求n次幂。采用矩阵的快速幂算法，操作次数可优化为O(log2 n)。

由于F(47)=(2971215073)10<232，F(48)=(4807526976)10>232，电路中采用32位二进制数表示一个整数。为了避免整数溢出，取2≤n≤47，n用6位二进制数表示。

**（2）算法描述**

Fibonacci(){

初始化：, Start=0；

For (i=5 downto 0)

{

if (Start==0) then

{

if (n[i]==1) then Start=1;

}

Else

{

if (n[i]==1)

then X=X2•A;

else

X=X2; }

}

return(X);

}

例如：n = (101100)2 = (44)10

step1：i=5，Start=0，n[5]=1，此时Start置1；

step2：i=4，Start=1，n[4]=0，此时X = X2 = A2；

step3：i=3，Start=1，n[3]=1，此时X = X2 •A = (A2)2•A；

step4：i=2，Start=1，n[2]=1，此时X = X2 •A = ((A2)2•A)2 •A；

step5：i=1，Start=1，n[1]=0，此时X = X2 = (((A2)2•A)2 •A)2；

step6：i=0，Start=1，n[0]=0，此时X = X2 = ((((A2)2•A)2 •A)2)2；

循环执行完后，X = ((((A2)2•A)2 •A)2)2 = A44

**（3）矩阵计算模块**

计算X2模块sqrX

（公式5）

其相应的输入/输出如图4-1所示。

sqrX

a

b

c

d

a′ = a2+bc

b′ = ab+bd

c′ = ac+cd

d′ = bc+d2

图4-1 计算X2模块sqrX输入/输出示意图

这里，a, b, c, d, a′, b′, c′, d′都为32位无符号二进制整数。

计算X2·A模块sqrX\*A

（公式6）

其相应的输入/输出如图4-2所示。

sqrX\*A

a

b

c

d

a″ = ab+bd

b″ = a2+bc+ab+bd

c″ = bc+d2

d″ = ac+cd+bc+d2

图4-2 计算X2·A模块sqrX\*A输入/输出示意图

这里，a, b, c, d, a″, b″, c″, d″都为32位无符号二进制整数。

**（4）矩阵快速幂算法迭代模块**

该模块Fibo输入/输出端如图4-3所示。

Fibo

start

clr

ni-1

clk

Fi = bi′ or bi″

图4-3 Fibo输入/输出示意图

这里，start为Fibonacci()算法中的6位二进制数n左移出的第一个（最高位的）1的标志信号；ni-1是start=1之后左移出的下一位；clr为初始化（清零）信号，此时X = A；clk为时钟脉冲信号。Fi为Fibonacci()算法迭代的中间结果，根据ni-1取0或1来决定Fi是取sqrX或者sqrX\*A运算后的矩阵元素bi，在第6个时钟脉冲时，Fi即为输入n的Fibonacci数Fn。

其内部逻辑结构图如图4-4所示。



图4-4 Fibo内部逻辑结构图

**（5）Fibonacci数显示模块**

将二进制数转换成十进制数在数码显示管上显示出来。

输入为32位二进制的Fibonacci数F(n)。

由于32位二进制Fibonacci数表示的最大十进制数的位数是10位，该模块的输出为10组8421BCD码D9、D8、D7、D6、D5、D4、D3、D2、D1、D0，每组8421BCD码表示1位10进制数。

**（6）主模块main**

主模块main的逻辑结构图4-5所示。



图4-5 主模块main的逻辑结构图

控制器Controller中包括三个功能块：6位二进制数n的左移控制电路、6个时钟脉冲控制电路、start信号产生电路。

6位二进制数n的左移控制电路，使用一个移位寄存器，在时钟脉冲作用下产生ni-1。用clear信号装入n，进行移位寄存器的初始化。

使用1个8位计数器、1个比较器和适当的门电路，可以控制Fibo只接收6个clock时钟脉冲（产生clk）。直至下一个clear信号初始化后，才准备产生下一组6个时钟脉冲。

使用1个D触发器加适当的门电路构成一个锁存器Latch，在接收到n的最高位1时start=1，直至下一个clear信号使start=0。

在6个clock时钟脉冲信号后，电路就产生了第n个Fibonacci数F(n)，并经过Display电路转换成十进制数在数码管上显示出来。

5、实验方案设计

**（1）斐波那契(Fibonacci)数列计算**

**要求：**

给出Fibonacci数列通项公式；



给出Fibonacci数列的递归算法（指数时间复杂度）形式化描述；

利用递归思想，每次计算当前的值时候，就要引用之前的两个值，一步一步的递归，一直到最起始处，才能用到F(1)和F(2)。递归算法时间复杂度为：O(2^(N/2))<=T(N)<=O(2^N)。

给出Fibonacci数列的多项式时间复杂度算法形式化描述。

Fibonacci(n)

a=0;

b=1;

if(n==0) return a;

for(i=0;i<n;i++)

{

a=b;

b=a+b;

}

return b;

**（2）计算矩阵X2模块**

**要求：**

给出矩阵X2计算模块的设计思路；

输入为a,b,c,d输出为a’,b’,c’,d’。其中

a′ = a2+bc

b′ = ab+bd

c′ = ac+cd

d′ = bc+d2

根据输出与输入的关系，用乘法器和加法器连接即可。

给出logisim软件绘制的电路图（经过仿真验证基本正确）；

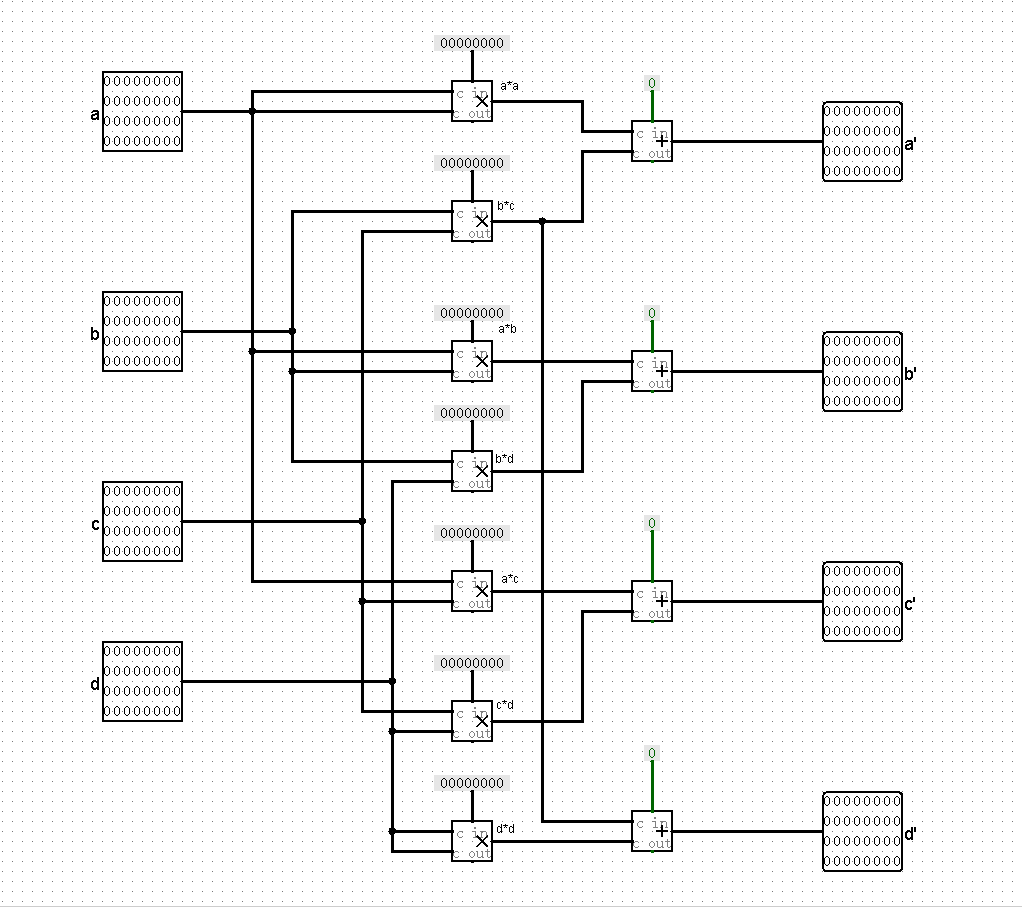


图4-6 矩阵X2计算模块电路图

对矩阵X2模块进行封装，给出封装后的模块图。

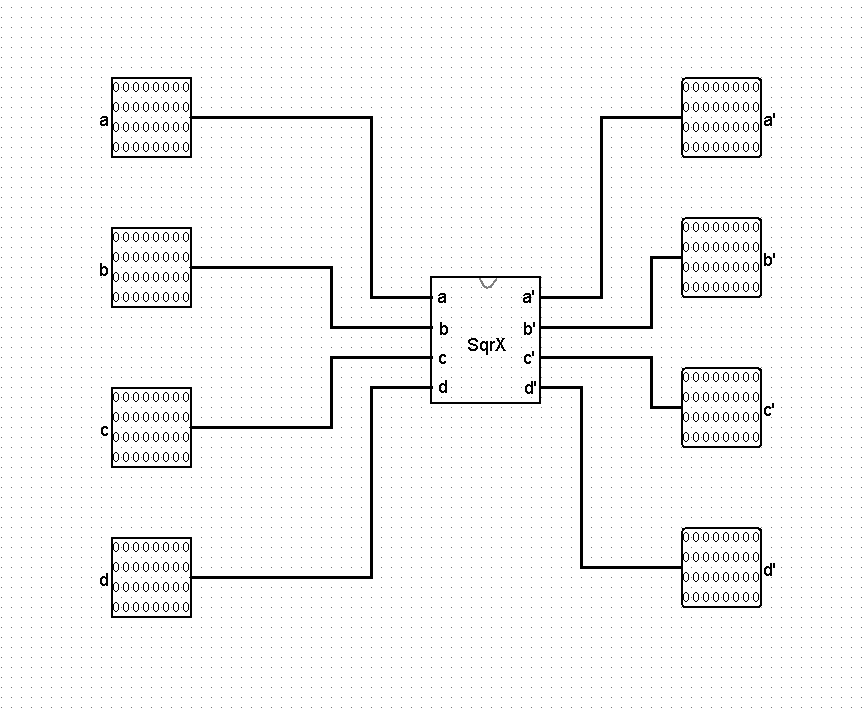


图4-7 矩阵X2模块封装截图

**（3）计算矩阵X2·A模块**

**要求：**

给出矩阵X2·A计算模块的设计思路；

输入为a,b,c,d，输出为a’’,b’’,c’’,d’’，其中

a″ = ab+bd

b″ = a2+bc+ab+bd

c″ = bc+d2

d″ = ac+cd+bc+d2

根据输出与输入的关系，用乘法器和加法器连接即可。

给出logisim软件绘制的电路图（经过仿真验证基本正确）；

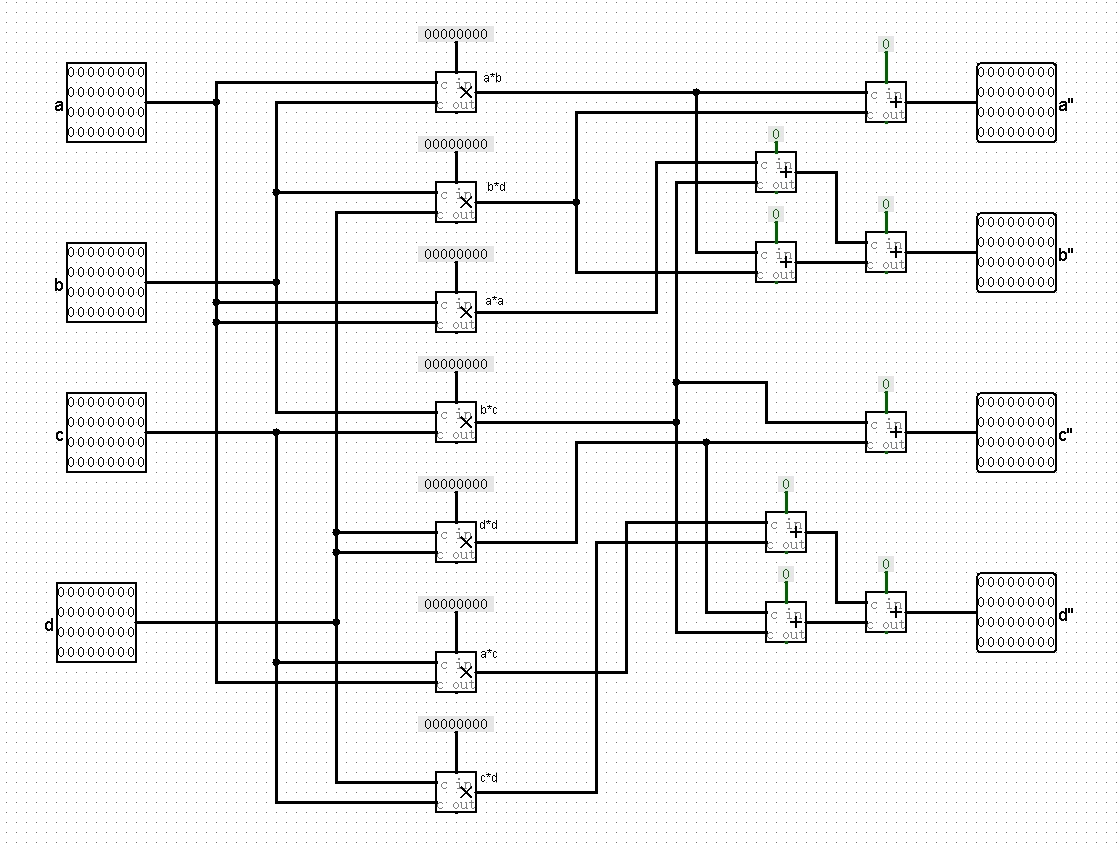


图4-8 矩阵X2·A计算模块电路图

对矩阵X2·A模块进行封装，给出封装后的模块图。

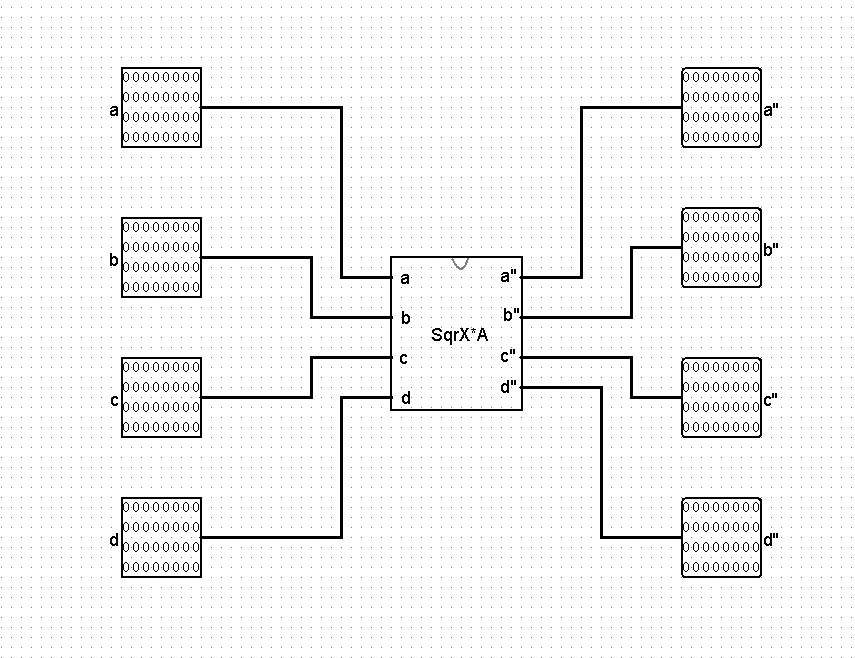


图4-9 矩阵X2·A模块封装截图

**（4）矩阵快速幂算法迭代模块设计**

**要求：**

给出矩阵快速幂算法迭代模块设计思路；

利用四个多路选择器，当start为0，输入初始矩阵{0，1，1，1}，为1时输入上一步的结果，接着用四个寄存器存储多路选择器的值，当时钟来时将值传给下一步，再用四个多路选择器根据ni-1的值选择矩阵计算模块，其中，start信号后加两个缓冲器确保清零功能正常。

给出logisim软件绘制的电路图（经过仿真验证基本正确）；

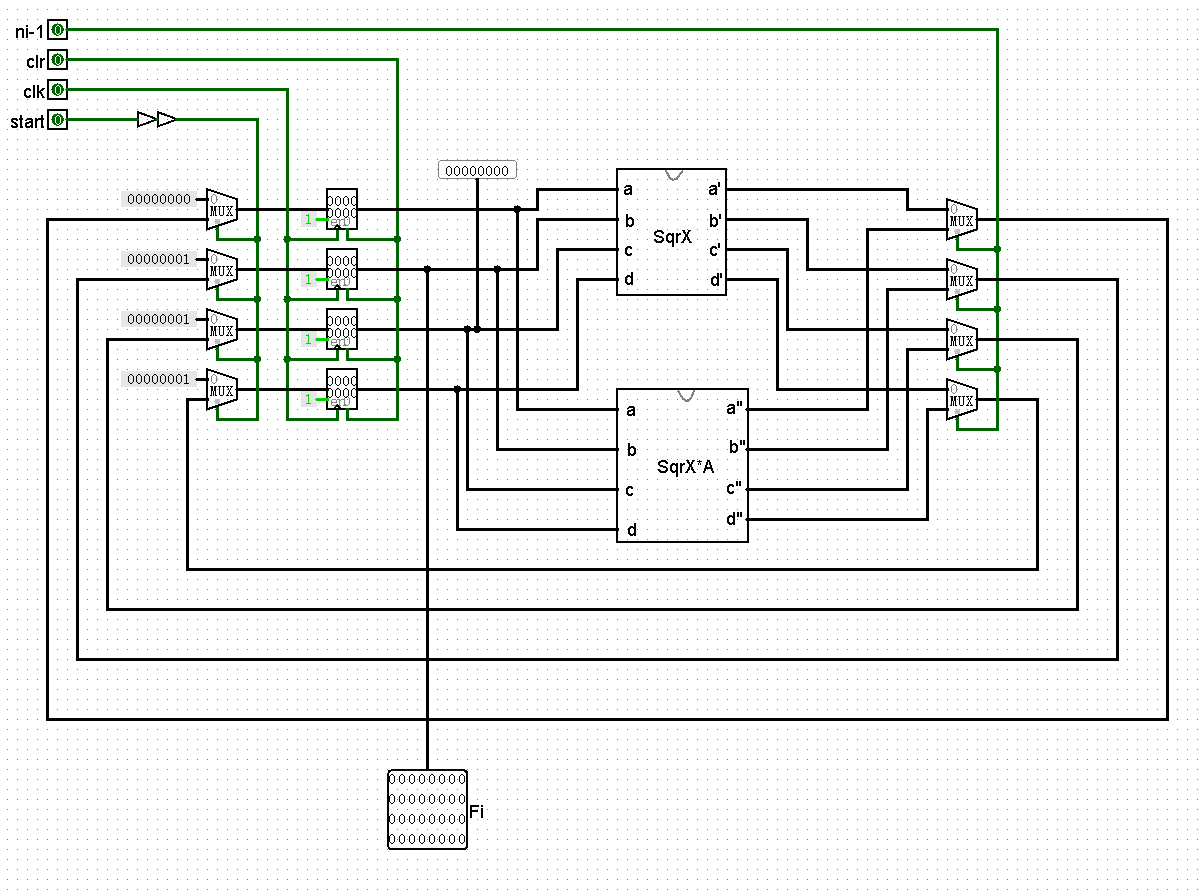


图4-10 矩阵快速幂算法迭代模块电路图

对矩阵X2·A模块进行封装，给出封装后的模块图。

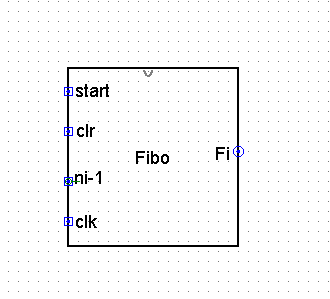


图4-11 矩阵快速幂算法迭代模块封装

**（5）主模块main设计**

**要求：**

说明主模块main中控制和显示部分的设计思路；

控制：将n的每一位扩展成6位送给移位器的输入，并在clear信号给移位器的加载端和移位端的分支加两个缓冲器确保移位器存储值正常更新。利用计数器和比较器，当接受6个脉冲后，小于端输出为0，跟时钟端通过与门连接，这样时钟无法通过。利用D触发器和或门，当start值为1时，保证D一直为1，直到clear信号。

显示：采用除10取余法将结果转化为10进制数，输入为Fi，输出为10组8421BCD码，串联10个除法器，每个除法器的被除数为上一个的商，余数即为该组8421BCD码。

给出主模块的logisim软件绘制的电路图（经过仿真验证基本正确）。

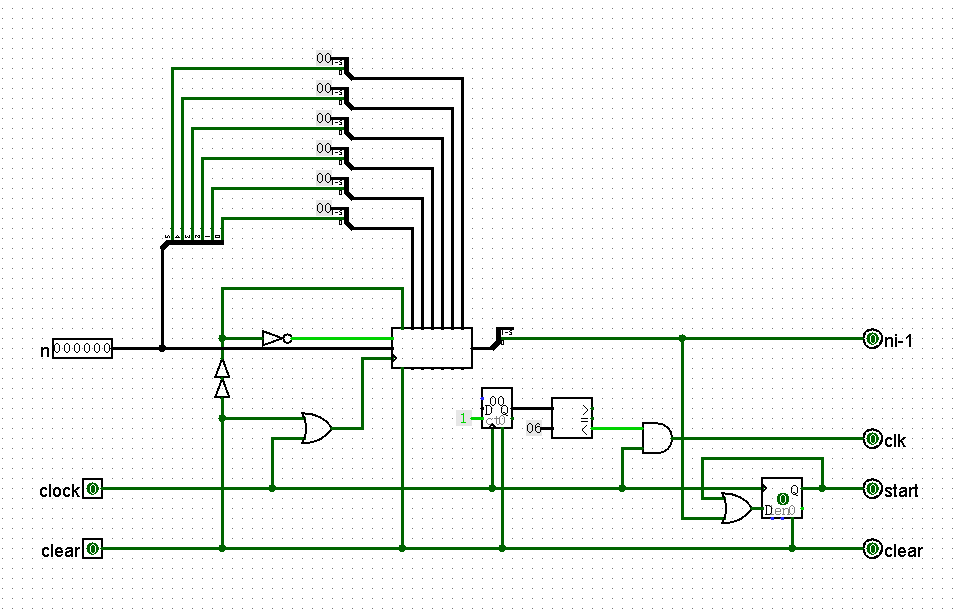


图4-12 控制电路

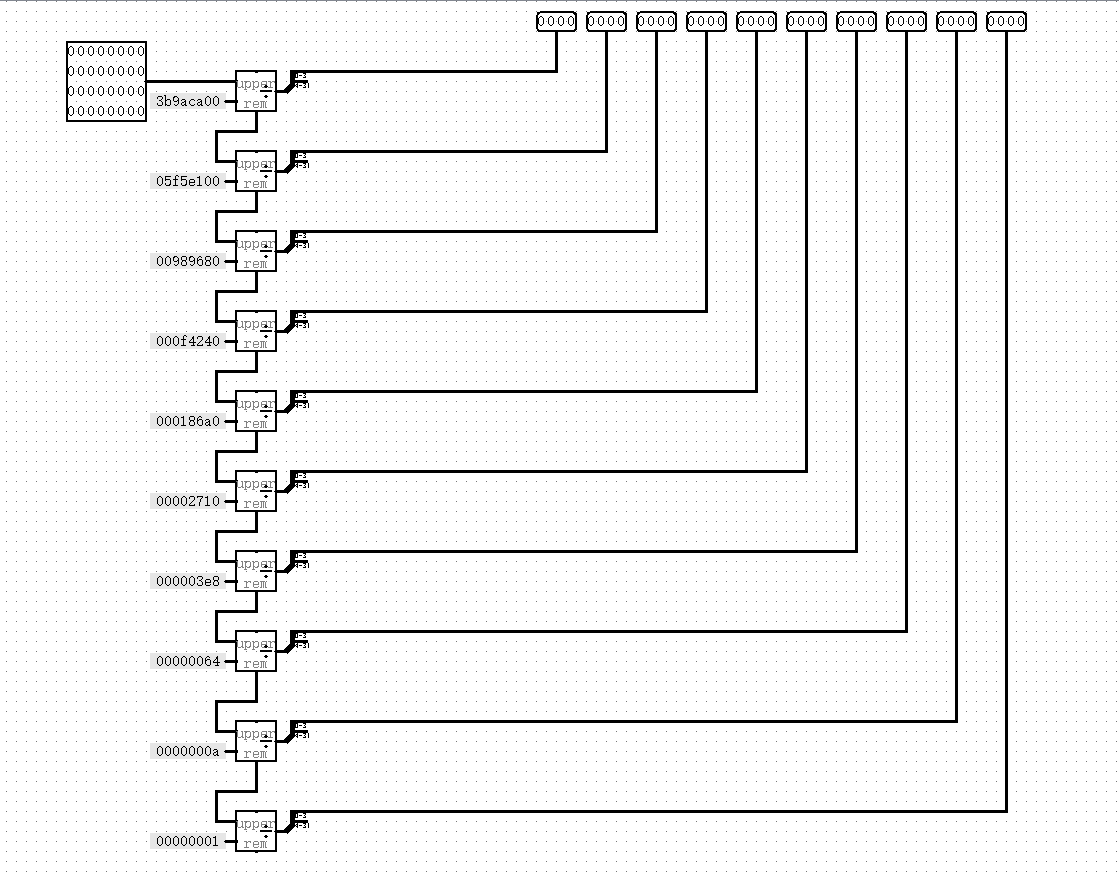


图4-13 显示电路

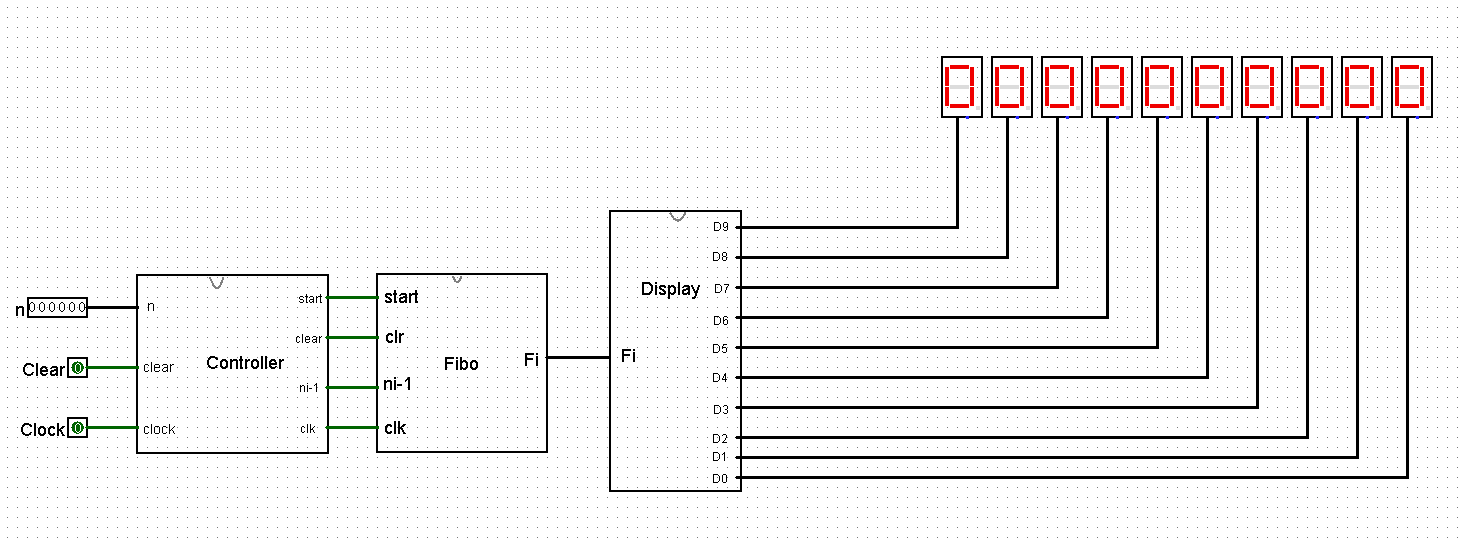


图4-14 主模块电路

6、实验结果记录

根据下表中所列内容，记录相应信号作用后输出数码管显示数据，并填入表1中（注：要求clear、clock使用按钮输入）。

表1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Input n | clear | 1st clock | 2nd clock | 3rd clock | 4th clock | 5th clock | 6th clock | After  6th clock |
| 2 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 5 | 0 | 1 | 1 | 1 | 1 | 1 | 5 | 5 |
| 10 | 0 | 1 | 1 | 1 | 1 | 5 | 55 | 55 |
| 17 | 0 | 1 | 1 | 1 | 3 | 21 | 1597 | 1597 |
| 25 | 0 | 1 | 1 | 2 | 8 | 144 | 75025 | 75025 |
| 32 | 0 | 1 | 1 | 3 | 21 | 987 | 2178309 | 2178309 |
| 44 | 0 | 1 | 1 | 5 | 89 | 17711 | 701408733 | 701408733 |
| 45 | 0 | 1 | 1 | 5 | 89 | 17711 | 1134903170 | 1134903170 |
| 46 | 0 | 1 | 1 | 5 | 89 | 28657 | 1836311903 | 1836311903 |
| 47 | 0 | 1 | 1 | 5 | 89 | 28657 | 2971215073 | 2971215073 |

7、实验中遇到的问题及解决方法

（1）故障1

问题描述：清零出现错误，不能输出0

问题分析：start和clear信号同时到，导致寄存器输出的值并不是0，二是前面的数据选择器组的值

解决方法：如图4-15

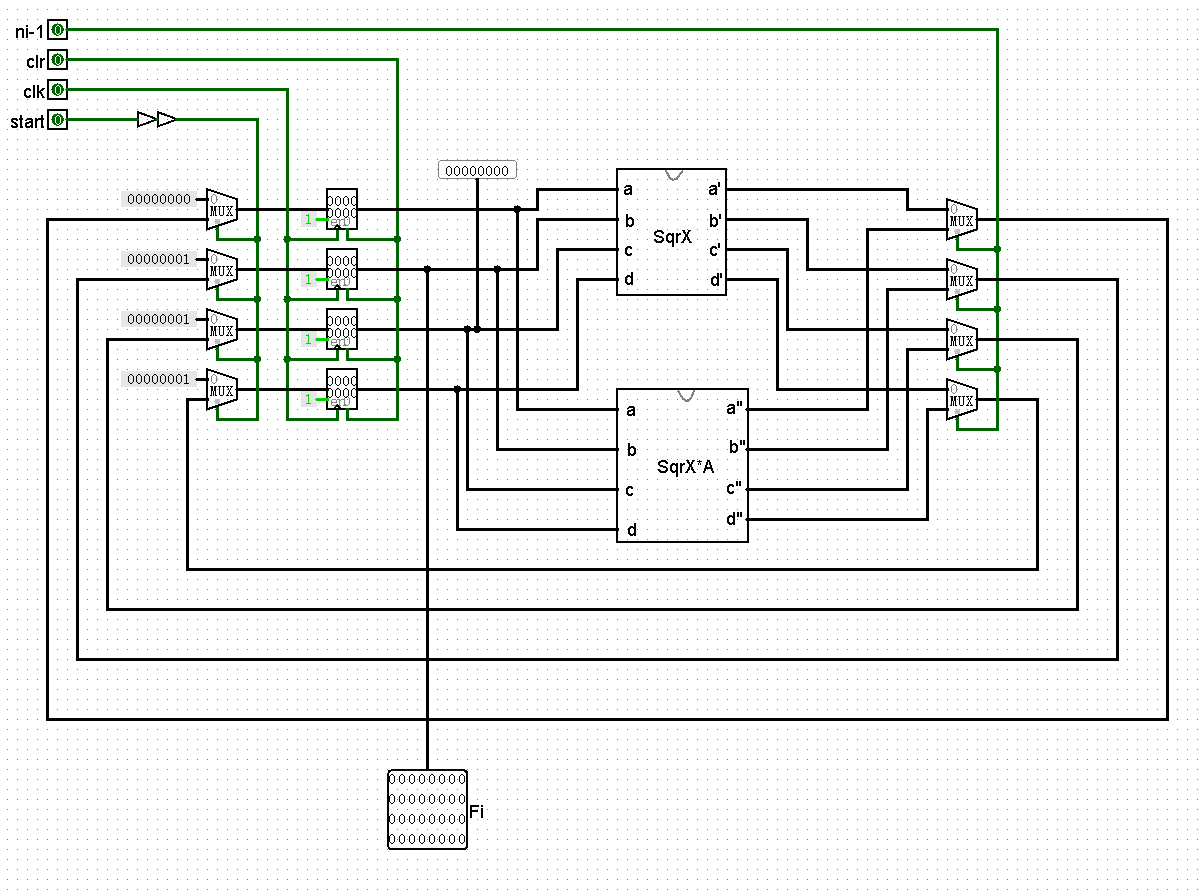


图4-15 更改后的实例

8、思考题

**（1）**如果n=48,电路的输出结果是多少？为什么？如果要求n=64时电路的输出结果也需正确，对你设计的电路应作哪些改进？

输出512559680，因为n的值超过了该电路能计算的斐波那契数列的值。

改进：计算模块每个输入和输出都改成两个32位引脚，矩阵快速幂算法迭代模块里面的寄存器和数据选择器数量也要翻倍，主模块n为8位，十六进制数字显示器数量也做相应增加。

**（2）**还有没有其它时间复杂度为O(log2 n)的算法？如有，请描述该算法，并简要说明该算法的硬件实现思路。

算法描述如下：



递推公式为：

Fibonacci(){

a=0,b=1,Start=0；

For (i=5 downto 0)

{

if (Start==0) then

{

if (n[i]==1) then Start=1;

a=1;

b=1;

}

Else

{

t1=(2\*b-a)\*a;

t2=a2+b2

if (n[i]==1) then

{

a=t1;

b=t2;

}

else

{

a=t2;

b=t1 + t2; +

}

}

}

return(a);

}

由控制模块产生start和ni-1，控制模块基本不变，修改计算模块，输入a和b，计算t1和t2，由ni-1的值选择a=t1,b=t2。

**（3）**请谈谈对用硬件和用软件实现同一算法的优势和劣势。

硬件实现一般效率更高，但难以更改，成本较高；

软件实现效率比硬件实现要低，但易修改，成本低。

9、心得体会、意见与建议

本次实验相比上次的电子钟实验，实验内容有所减少，因此相比上一次轻松一些。实验中最大的难题，矩阵算法的实现已经在实验内容里很详细的给出了，相比较递归算法，这种算法不仅高效，且更易用硬件实现。而且实验报告中已经给出了电路设计的思路或者模块图，矩阵计算模块、矩阵快速幂算法迭代模块也都只需要按照表达式或者逻辑构图，使用相应器件实现。另外在设计的时候针对不同问题，一定要注意数据位宽的问题，不要让整数溢出。