

# **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： 丁文隆**

**学 号： U201714850**

**班 级： ACM1701**

**指 导 教 师： 赵贻竹**

**计算机科学与技术学院**

**2019 年 5 月 27 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1所示。

sqrX

a

b

c

d

a′ = a2+bc

b′ = ab+bd

c′ = ac+cd

d′ = bc+d2

图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3所示。

Fibo

start

clr

ni-1

clk

Fi = bi′ or bi″

图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所示。



图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5所示。



图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。

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？

如果n=48，则电路输出的结果为512559680，显然这个结果是不对的，因为它比n=47的时候还要小。我们猜测可能是有值溢出才会导致错误的结果，于是我们返回到Fibo模块中来看看n=48的时候，是哪个值会溢出。通过实验我们发现，在第五个脉冲的时候，d的值比c的值变小了，可见第五个值的时候d发生了溢出，给第六个脉冲的时候，c的值比之前的反而变小了，可见c的值发生了溢出。所以，是因为d和c都发生了溢出现象，导致输出结果不对（比n=47还要小）。

如果要使得n=64的时候结果也正确，我们不能只满足于a、b、c、d只有32位，我们要使得计算的时候没有溢出产生。经试验，我们发现，将位数扩展到64位时不会有溢出的情况产生。所以我们将矩阵的值a、b、c、d扩展成64位即可。（对应模块的位宽或者寄存器从32位改成64位即可）

**（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。

查阅资料发现斐波那契数列的递推公式的另外一种形式：

用这个公式给出的Fibonacci计算的时间复杂度也为。

下面给出这个算法复杂度的实现思路（硬件实现）：

首先，由上面两个公式我们能推出以下两个我们需要用的公式：

我们构造一个矩阵，计算矩阵，其中。初始状况n=1的时候，X=A。

（a）下面我们决定要计算几次和几次：

1.构造一个判断串m，以及一个计数器。串中为0的位数表示计算F2n，串中为1的位数表示计算F2n+1。（为了保险，串m构造32位的，初始值为全是0；计数器初始值为0）

2.判断输入的n，看是否为1，若是1，锁住计数器的值不再增加。若不是1，则判断是否为偶数，若是偶数，则n=n/2，给m当前位加载一个0，m当前位往下移动一位，计数器加一。若n不为1且判断n当前值为奇数，则n=n+1/2，给m当前位加载一个0，m当前位往下移动一位，计数器加一。

3.重复2中的步骤，直到n=1，锁住计数器值不再增加。

（b）下面开始计算Fibonacci

1.开始计算时给一个clear（装载）信号，将矩阵A的值赋给矩阵X且计数器、比较串m的值清零。

2.将m的值从高位向低位一位位地输出（即从先加载的位数开始）。若碰到0，则对矩阵X执行以下操作：

先执行：

再执行：

计时器减一。

若碰到1，则对矩阵X执行以下操作：

先执行：

再执行：

计时器减一，

(先执行再执行可以通过前后两个时钟脉冲来实现)

3.到计数器减到0时，矩阵的值稳定住，并输出b，即为要输出的Fibonacci计算值。

**（3）**请谈谈对用硬件和用软件实现同一算法的优势和劣势。

用软件来实现一个算法，最大的好处就是方便快捷，而且也具有用户友好性（易于被除设计者外的其他人读懂）。一个很复杂的电路，往往只对应着很简单的算法，通过软件语言，我们将复杂的电路抽象成公式或者数学逻辑，目的明确，更有利于展现思维，和其他人交流，让其他人看懂你的思路。而软件的劣势在于，一旦涉及到一些比较底层的东西，软件实现算法就有劣势了，比如如果出现了溢出问题（位数不够大的问题），用硬件我们只需要进行位扩展即可，而对于软件，我们可能要重新设计算法来实现同一个功能（大数算法还是很复杂的），同时，若用软件来编写算法，我们想了解其底层运作就更加复杂（和其编译器有关），我们需要了解其编译特点，看懂其汇编代码，然后再来看它是怎么操作的；若用硬件语言，其在机器层面上怎么运作的就一目了然了。