|  |
| --- |
|  |
| 汉明码编解码系统 |
| 设计报告 |
|  |
| 组号：D106 |
| 2018年5月12日 |

|  |
| --- |
|  |

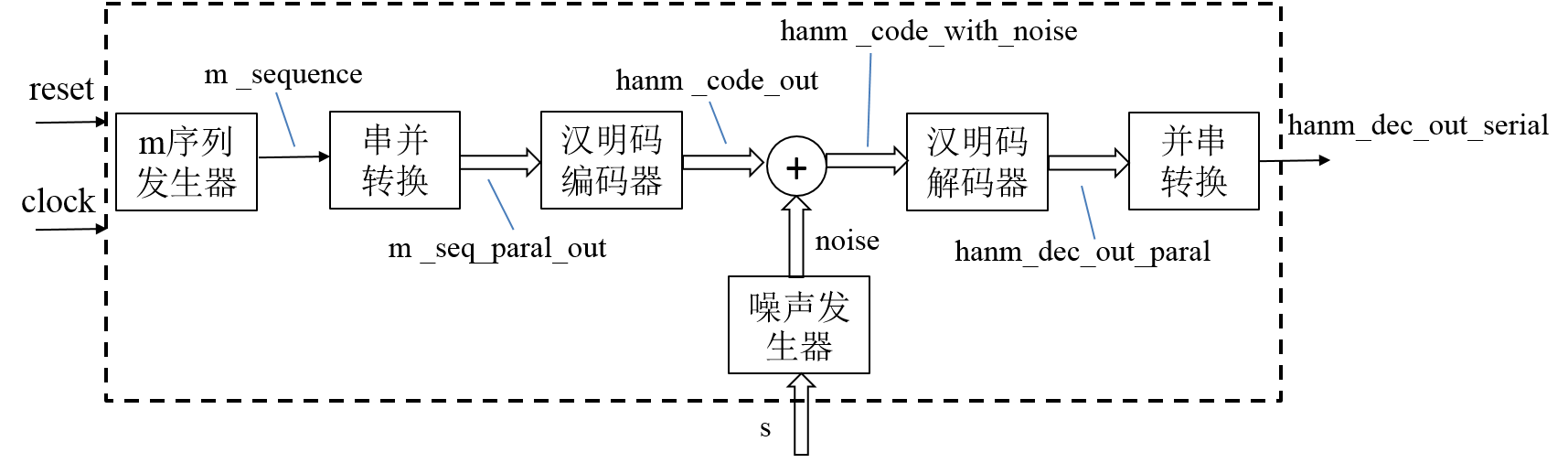
一、电路与程序设计

（一）设计思路

该作品设计了一个完整的汉明码编解码系统，引入了一定的噪声，完成了汉明码检错和纠错的功能。我们设计的系统包括包括m序列生成电路（M\_seq）、串并转换电路（paral\_out），汉明码编码电路（hanm\_code）、加噪电路（noise\_generate）、汉明码解码电路（hanm\_decode）、并串转换电路（out\_serial）共六个模块，并在hanming\_system文件中进行引用并且生成内部时钟模块方便各个模块调用（top文件），最终利用已有的testbench进行测试，得到了测试结果。

（二）设计框图

总体设计框图如下图所示：（题目中给出的系统框图）



**图1 总体设计框图**

根据题目的要求，我们得到了系统RTL级电路图与总体设计框图相一致，但是由于篇幅所限，附在了文件夹中名为“RTL级电路hanming\_system”的图片中。

（三）时序说明

testbench当中提供的系统时钟周期为20ns，频率为50MHz，题目当中要求的序列信号m\_sequence码速率为10kbps，因此首先需要在系统内部生成10kHz的时钟1（clock\_1）。

然后将生成的10kHz时钟通入M序列生成模块（M\_seq），每1个上升沿输出一个序列信号m\_sequence，送入到串并转换模块当中。

串并转换模块仍然使用10kHz时钟，每4个上升沿输出一个4位的码组m\_seq\_paral\_out，先输入的先输入的串行数据位于并行输出数据的高位。

4位的码组输入到汉明码编码器当中进行编码，此时可以采用内部生成的2.5kHz时钟2（clock\_2）进行处理，每1个上升沿处理一次。

噪声产生模块与加噪模块不需要在时钟的控制，直接将噪声与编码后的结果进行相加，送入汉明码解码器。

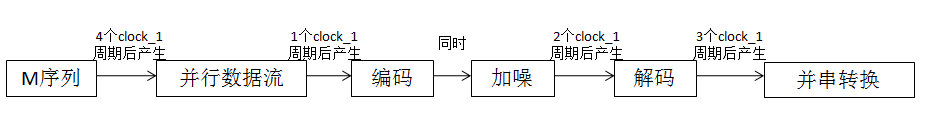
汉明码解码器使用2.5kHz的内部clock\_2，每1个时钟下降沿进行解码，将解码之后的结果进行并串转换。

并串转换使用10kHz的内部时钟clock\_1，每一个时钟上升沿将并行数据依次输出码元，至此该系统完成了全部的任务。

**表1 各模块使用的时钟频率**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 模块名称 | hanming\_system | M\_seq | paral\_out | hanm\_code |
| 时钟频率 | 50MHz | 10kHz | 10kHz | 2.5kHz |
| 模块名称 | noise\_generate | hanm\_decode | out\_serial |  |
| 时钟频率 | 不需要 | 2.5kHz | 10kHz |  |

经过仿真测试，reset信号变成低电平后，得到的时序如下图所示：



**图2 时序说明**

（四）模块设计说明

1、top模块（hanming\_system）

该模块的功能是作为top模块，生成10kHz及2.5kHz系统内部时钟，调用各个功能模块。系统内部时钟由cnt1及cnt2计数分频所得，clock\_1是clock时钟5000分频，clock\_2是clock时钟20000分频。

**表2 top模块端口说明**

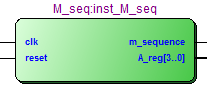
|  |  |  |  |
| --- | --- | --- | --- |
| 端口说明 | 输入/输出 | 位宽 | 端口作用 |
| clock | input | 1 | 外部时钟信号50MHz |
| reset | input | 1 | 复位信号 |
| m\_sequence | output | 1 | 输出m序列数字信号 |
| A\_reg | output | 4 | 移位寄存器状态信号 |
| m\_seq\_paral\_out | output | 4 | 并行输出数据流 |
| hanm\_code\_out | output | 7 | 输出汉明编码结果 |
| s | input | 3 | 噪声控制信号 |
| hanm\_code\_with\_noise | output | 7 | 输出汉明编码加噪信号 |
| noise | output | 7 | 噪声信号 |
| hanm\_dec\_out\_paral | output | 4 | 汉明解码信号 |
| error\_pointer | output | 3 | 误码位置指示信号 |
| ne | Output | 1 | 误码存在指示信号 |
| hanm\_dec\_out\_serial | output | 1 | 串行数字序列 |

2、m序列生成电路（M\_seq）

该模块的功能是根据题目的要求生成相应的m序列。m序列的特征方程为，采用线性移位寄存器来产生，因此递归关系式为a4=a0a1。利用always语句在reset为高电平时复位，A\_seq=1000，reset为低电平时，每一个时钟上升沿，此处采用位拼接的方法对寄存器进行移位，输出寄存器的最低位A\_seq[0]作为m\_sequence。

**表3 M\_seq模块端口说明**

|  |  |  |  |
| --- | --- | --- | --- |
| 端口说明 | 输入/输出 | 位宽 | 端口作用 |
| clk | input | 1 | 时钟信号 |
| reset | input | 1 | 复位信号 |
| m\_sequence | output | 1 | 输出m序列数字信号 |
| A\_reg | output | 4 | 移位寄存器状态信号 |



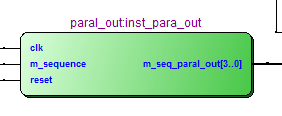
**图3 M\_seq模块端口说明**

3、串并转换（paral\_out）

该模块的功能是对生成的m序列进行串并转换，每四个序列信号生成一个4位的数据流送入下一个模块当中进行编码。利用always语句在reset为高电平时复位，数据流m\_seq\_paral\_out异步置零，reset为低电平时，每一个时钟上升沿，采用位拼接对寄存器进行移位，刚输入的m序列位于寄存器的最低位，其余各位依次向高位移动，直至4个时钟上升沿后输出4位的数据流，这样就可以达到每4位m序列输出一个数据流的目的了。

**表4 paral\_out模块端口说明**

|  |  |  |  |
| --- | --- | --- | --- |
| 端口说明 | 输入/输出 | 位宽 | 端口作用 |
| clk | input | 1 | 时钟信号 |
| reset | input | 1 | 复位信号 |
| m\_sequence | Input | 1 | 输入m序列数字信号 |
| m\_seq\_paral\_out | output | 4 | 并行输出数据流 |



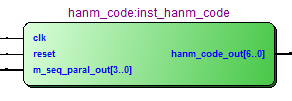
**图4 paral\_out模块端口说明**

4、汉明码编码（hanm\_code）

该模块对串行输入的4位一组m序列进行汉明码编码，最终生成七位一组的包含四位信息位和三位监督元的码组送入下一个模块进行加噪。利用always语句在reset在高电平时进行异步置零，reset为低电平时，每时钟上升沿将m\_seq\_paral\_out赋给hanm\_code信息位即高四位，监督位即低三位通过对m\_seq\_paral\_out中信号进行相应的模二加得到。具体编码规则即监督位a2a1a0和信息位a6a5a4a3的关系为：

**表5 hanm\_code模块端口说明**

|  |  |  |  |
| --- | --- | --- | --- |
| 端口说明 | 输入/输出 | 位宽 | 端口作用 |
| clk | input | 1 | 时钟信号 |
| reset | input | 1 | 复位信号 |
| m\_seq\_paral\_out | Input | 4 | 输入m序列并行数据流 |
| hanm\_code\_out | output | 7 | 输出汉明编码结果 |



**图5 hanm\_code模块端口说明**

5、加噪模块（noise\_generate）

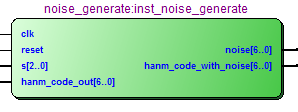
噪声生成电路输入信号与输出信号关系如下表所示，利用always语句及case语句，当s（2..0）发生变化后，生成相应的噪声，然后直接对噪声与上一个模块生成的数据流m\_seq\_paral\_out逐位模2加，就可以得到加噪的数据流，以此验证汉明码的检错纠错功能。

**表6 噪声生成电路输入信号与输出信号关系**

|  |  |  |
| --- | --- | --- |
| reset | s(2..0) | noise(6..0) |
| 1 | - | 0000000 |
| 0 | 000 | 0000001 |
| 001 | 0000010 |
| 010 | 0000100 |
| 011 | 0001000 |
| 100 | 0010000 |
| 101 | 0100000 |
| 110 | 1000000 |
| 其它 | 1100000 |

**表7 noise\_generate模块端口说明**

|  |  |  |  |
| --- | --- | --- | --- |
| 端口说明 | 输入/输出 | 位宽 | 端口作用 |
| clk | input | 1 | 时钟信号 |
| reset | input | 1 | 复位信号 |
| s | input | 3 | 噪声控制信号 |
| hanm\_code\_out | input | 7 | 输入汉明编码结果 |
| hanm\_code\_with\_noise | output | 7 | 输出汉明编码加噪信号 |
| noise | output | 7 | 噪声信号 |



**图6 noise\_generate模块端口说明**

6、汉明码解码电路（hanm\_decode）

该模块的功能是将加噪后的汉明编码信号进行处理，得出是否存在误码以及误码位置等信息，并最终得到正确的汉明码编码和原来的m序列信号。首先对通过以下运算得到校正子S:

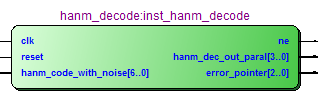
再根据校正子和误码位置的关系得到误码位置，定义error\_pointer为误码位置指示信号对误码位置进行定位。最终通过将e即传输中的误码和加噪汉明码进行模二加运算后得到纠正的汉明码编码，位截取得到最终解码出来的四位一组m序列。

**表8 （7,4）汉明码校正子和误码位置的关系**

|  |  |  |  |
| --- | --- | --- | --- |
| S2S1S0 | 误码位置 | S2S1S0 | 误码位置 |
| 001 | a0 | 101 | a4 |
| 010 | a1 | 110 | a5 |
| 100 | a2 | 111 | a6 |
| 011 | a3 | 000 | 无错码 |

**表9 hanm\_decode模块端口说明**

|  |  |  |  |
| --- | --- | --- | --- |
| 端口说明 | 输入/输出 | 位宽 | 端口作用 |
| clk | input | 1 | 时钟信号 |
| reset | input | 1 | 复位信号 |
| hanm\_code\_with\_noise | Input | 7 | 汉明编码加噪信号 |
| hanm\_dec\_out\_paral | output | 4 | 汉明解码信号 |
| error\_pointer | output | 3 | 误码位置指示信号 |
| ne | Output | 1 | 误码存在指示信号 |



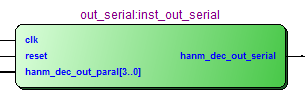
**图7 hanm\_decode模块端口说明**

7、并串转换（out\_serial）

该模块的功能是将并行的数据流转换成串行数据进行输出，利用always语句及位拼接对temp寄存器进行循环移位，每一个上升沿输出一个码元，每四个上升沿之后，对temp寄存器进行刷新，从而输出正确m序列。

**表10 out\_serial模块端口说明**

|  |  |  |  |
| --- | --- | --- | --- |
| 端口说明 | 输入/输出 | 位宽 | 端口作用 |
| clk | input | 1 | 时钟信号 |
| reset | input | 1 | 复位信号 |
| hanm\_dec\_out\_paral | Input | 4 | 汉明码解码信号 |
| hanm\_dec\_out\_serial | output | 1 | 串行数字序列 |



**图8 out\_seriale模块端口说明**

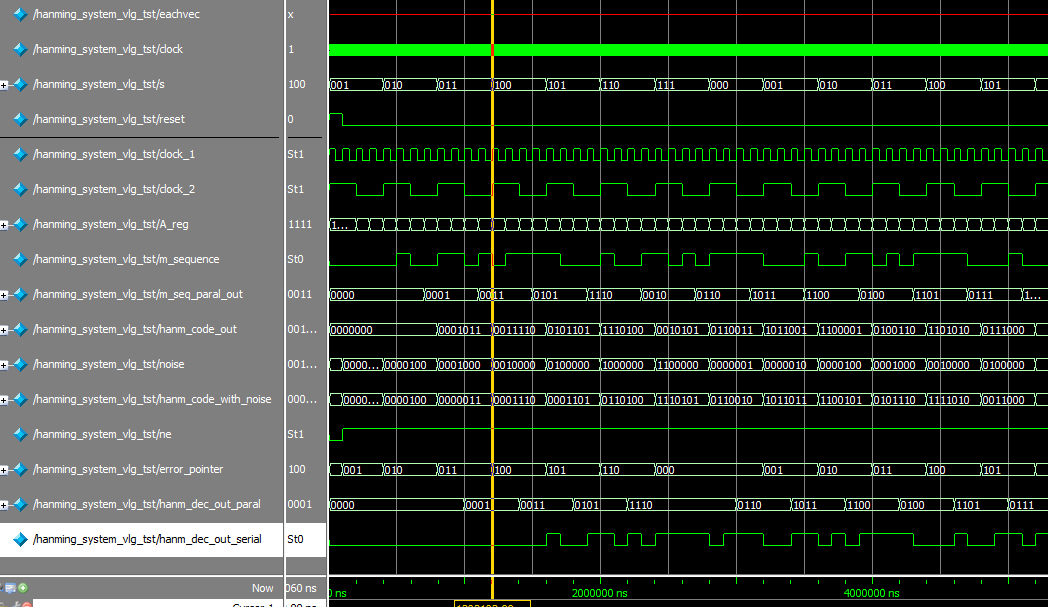
**（五）程序及注释**

具体程序及和注释见**附录一**。

二、测试结果及结论

（一）测试结果

在Modelsim仿真得到的波形图如下图所示：



**图9 Modelsim 仿真测试图（更大的图详见附录二）**

由仿真图中得到的结果如下：

1. clock\_1，clock\_2成功对clock信号进行分频，经过读图可以得到其频率分别为10kHz及2.5kHz。



**图10 clock\_1，clock\_2波形图**

1. A\_reg可以看到移位寄存器移位的整个过程，m序列为0001\_0011\_0101\_111这15个数的周期循环。

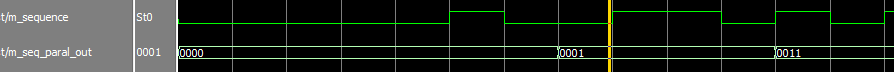


**图11 m\_sequence波形图**



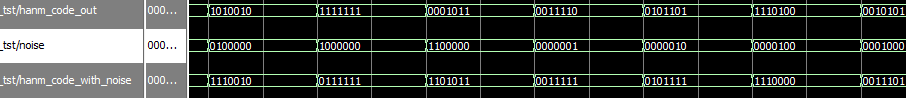
**图12 A\_reg波形图**

1. 经过串并转换后，成功得到m\_seq\_paral\_out数据流，且顺序完全正确。



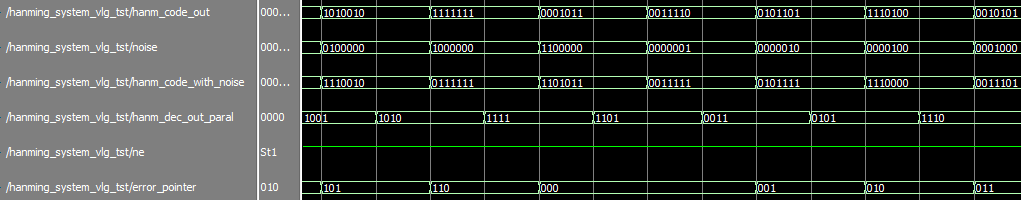
**图13 m\_seq\_paral\_out波形图**

（4）经过汉明码编码模块后4为数据流加上了三位监督位，在引入了噪声之后，输出的hanm\_code\_with\_noise与原先的汉明码编码hanm\_code\_out有1-2位的不同。



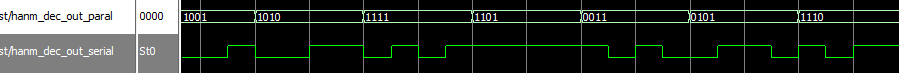
**图14 引入噪声后编码波形图**

（5）经过汉明码解码模块之后，进行了纠错，可以看到只是错误的数据ne在检测到错误之后一直保持高电平，并且大部分指针能指出错误的位置，当噪声为1100\_000（即s=111时），无法指出正确的位置，且解码错误。



**图15 解码后波形图**

（6）经过并串转换之后，解码得到的4位结果转换成了码元。



**图16 并串转换后波形图**

（7）让reset信号为高电平时，所有模块都进行了异步置零，当reset信号变成低电平时，系统开始正常工作。

另外，由于在reset下降沿时正好是clock\_1的上升沿，因此存在竞争与冒险的情况，我们可以看到A\_seq进行推移的时间向后顺延，由此带来的后果是串并转换、编码等后续步骤都向后顺延，我们不断调整程序，使得最终保证输出的4位编码4位数据流顺序与理论中相同，即首先输出的是0001（见下文分析）。

（二）结论

在实际数字基带传输中存在连“0”码或者连“1”码，为解决这个问题，将二进制数字信息先进行“随机化”处理，变为伪随机序列。m序列就是最常用的一种伪随机序列，是由带线性反馈的移位寄存器产生的序列。从本设计的m序列发生电路仿真结果中可以看出，移位寄存器序列是一种周期序列，在相同级数状态下，采用不同的线性反馈逻辑得到的周期长度不同。在本题目的设计中m序列的周期是15，得到的线性反馈移位寄存器序列是0001\_0011\_0101\_111，跟仿真图中得到的结果一致。

在实际信道传输中，由于信道传输特性不理想及加性噪声的影响，收到的数字信号会发生错误，从而需要采用信道编码。本题设计中即通过人为地加噪模拟了这一过程。汉明码就是一种典型的信道编码，通过对信息位进行操作产生监督位加在信息序列后面，达到监督误码及纠错的目的，提高了信号传输的准确性。

但汉明码也具有它的局限性即无法监督并改正多于一位的误码，所以在本题中当输入噪声为1100000时，无法指出误码位置并进行纠错。

本题将伪随机序列和汉明编解码结合在一起，让我们对这两个概念有了更深的理解，达到了这次集成电路设计比赛的初衷。

附录一：

**1、top模块（hanming\_system）**

module hanming\_system

(clock,

reset,

s,

m\_sequence,

A\_reg,

m\_seq\_paral\_out,

clock\_1,

clock\_2,

hanm\_code\_out,

noise,

hanm\_code\_with\_noise,

hanm\_dec\_out\_paral,

error\_pointer,

ne,

hanm\_dec\_out\_serial

);

input clock;

input reset;

input [2:0] s;

output clock\_1;

output clock\_2;

output m\_sequence;

output [3:0] A\_reg;

output [3:0] m\_seq\_paral\_out;

output [6:0] hanm\_code\_out;

output [6:0] noise;

output [6:0] hanm\_code\_with\_noise;

output [3:0] hanm\_dec\_out\_paral;

output [2:0] error\_pointer;

output ne;

output hanm\_dec\_out\_serial;

integer cnt1=0;

integer cnt2=0;

reg clock\_1=1'b1;

reg clock\_2=1'b1;

//分频时钟1，频率10kHz（码速率）

always@(posedge clock)

begin

if(cnt1==2499)

begin

cnt1=0;

clock\_1=~clock\_1;

end

else

cnt1=cnt1+1;

end

//分频时钟2，频率2.5kHz（输出四位一组的数据流的速率）

always@(posedge clock)

begin

if(cnt2==9999)

begin

cnt2=0;

clock\_2=~clock\_2;

end

else

cnt2=cnt2+1;

end

//m序列发生器

M\_seq inst\_M\_seq

(

.clk(clock\_1),

.reset(reset),

.m\_sequence(m\_sequence),

.A\_reg(A\_reg)

);

//串并转换

paral\_out inst\_para\_out

(

.clk(clock\_1),

.reset(reset),

.m\_sequence(m\_sequence),

.m\_seq\_paral\_out(m\_seq\_paral\_out)

);

//汉明码编码

hanm\_code inst\_hanm\_code

(

.clk(clock\_2),

.reset(reset),

.m\_seq\_paral\_out(m\_seq\_paral\_out),

.hanm\_code\_out(hanm\_code\_out)

);

//噪声生成及加噪

noise\_generate inst\_noise\_generate

(

.clk(clock\_1),

.reset(reset),

.s(s),

.hanm\_code\_out(hanm\_code\_out),

.noise(noise),

.hanm\_code\_with\_noise(hanm\_code\_with\_noise)

);

//汉明码解码

hanm\_decode inst\_hanm\_decode

(

.clk(clock\_2),

.reset(reset),

.hanm\_code\_with\_noise(hanm\_code\_with\_noise),

.hanm\_dec\_out\_paral(hanm\_dec\_out\_paral),

.error\_pointer(error\_pointer),

.ne(ne)

);

//并串转换

out\_serial inst\_out\_serial

(

.clk(clock\_1),

.reset(reset),

.hanm\_dec\_out\_paral(hanm\_dec\_out\_paral),

.hanm\_dec\_out\_serial(hanm\_dec\_out\_serial)

);

endmodule

**2、m序列生成电路（M\_seq）**

module M\_seq

(

clk,

reset,

m\_sequence,

A\_reg);

input clk;

input reset;

output m\_sequence;

output [3:0] A\_reg;

reg m\_sequence=1'bz;

reg [3:0] A\_reg=4'b1000;

reg temp=1'b0;

always@ (posedge clk or posedge reset)

begin

if(reset)

begin

//复位状态为最高位为1，其余位为0

A\_reg=4'b1000;

m\_sequence=1'b0;

temp=A\_reg[1]+A\_reg[0];

end

else

begin

//移位寄存器最低位输出为m序列信号

m\_sequence=A\_reg[0];

//通过位拼接更新移位寄存器状态

A\_reg={temp,A\_reg[3:1]};

temp = A\_reg[1]+A\_reg[0];

end

end

endmodule

**3、串并转换（paral\_out）**

module paral\_out

(

clk,

reset,

m\_sequence,

m\_seq\_paral\_out

);

input clk;

input reset;

input m\_sequence;

output [3:0] m\_seq\_paral\_out;

reg [3:0] m\_seq\_paral\_out;

integer cnt=0;

reg [3:0] temp;

always@(posedge clk or posedge reset)

begin

if(reset)

//异步清零

begin

cnt=0;

m\_seq\_paral\_out=4'b0000;

temp=4'b0000;

end

else

if(cnt==1)

begin

cnt=cnt+1;

m\_seq\_paral\_out=temp;

//位拼接进行串并转换

temp={temp[2:0],m\_sequence};

end

else if(cnt==3)

begin

cnt=0;

temp={temp[2:0],m\_sequence};

end

else

begin

cnt=cnt+1;

temp={temp[2:0],m\_sequence};

end

end

endmodule

**4、汉明码编码（hanm\_code）**

module hanm\_code

(

clk,

reset,

m\_seq\_paral\_out,

hanm\_code\_out

);

input clk;

input reset;

input [3:0] m\_seq\_paral\_out;

output [6:0] hanm\_code\_out;

reg [6:0] code\_out;

always@(posedge clk or posedge reset)

begin

if(reset)

//异步清零

begin

code\_out=7'b0000000;

end

else

//汉明吗编码

begin

code\_out[6:3]<=m\_seq\_paral\_out[3:0];

code\_out[0]<=(m\_seq\_paral\_out[3]^m\_seq\_paral\_out[1]^m\_seq\_paral\_out[0]);

code\_out[1]<=(m\_seq\_paral\_out[3]^m\_seq\_paral\_out[2]^m\_seq\_paral\_out[0]);

code\_out[2]<=(m\_seq\_paral\_out[3]^m\_seq\_paral\_out[2]^m\_seq\_paral\_out[1]);

end

end

assign hanm\_code\_out=code\_out;

endmodule

**5、加噪模块（noise\_generate）**

module noise\_generate

(

clk,

reset,

s,

hanm\_code\_out,

noise,

hanm\_code\_with\_noise

);

input clk;

input reset;

input [2:0] s;

input [6:0] hanm\_code\_out;

output [6:0] hanm\_code\_with\_noise;

output [6:0] noise;

reg [6:0] noise;

always@ (s[2:0] or reset)

begin

if(reset)

//异步清零

noise=7'd0;

else

//根据噪声控制信号产生噪声信号

case(s[2:0])

3'b000:noise=7'b0000\_001;

3'b001:noise=7'b0000\_010;

3'b010:noise=7'b0000\_100;

3'b011:noise=7'b0001\_000;

3'b100:noise=7'b0010\_000;

3'b101:noise=7'b0100\_000;

3'b110:noise=7'b1000\_000;

default:noise=7'b1100\_000;

endcase

end

//加噪

assign hanm\_code\_with\_noise = hanm\_code\_out ^ noise;

endmodule

**6、汉明码解码电路（hanm\_decode）**

module hanm\_decode

(

clk,

reset,

hanm\_code\_with\_noise,

hanm\_dec\_out\_paral,

error\_pointer,

ne

);

input clk;

input reset;

input [6:0] hanm\_code\_with\_noise;

output [3:0] hanm\_dec\_out\_paral;

output [2:0] error\_pointer;

output ne;

wire [2:0] s;

reg [6:0] e;

reg [6:0] c;

reg [2:0] error\_pointer;

reg ne=0;

reg [3:0] hanm\_dec\_out\_paral;

//汉明吗校正子的产生

assign s[0]=hanm\_code\_with\_noise[6]^hanm\_code\_with\_noise[4]^hanm\_code\_with\_noise[3]^hanm\_code\_with\_noise[0];

assign s[1]=hanm\_code\_with\_noise[1]^hanm\_code\_with\_noise[3]^hanm\_code\_with\_noise[5]^hanm\_code\_with\_noise[6];

assign s[2]=hanm\_code\_with\_noise[2]^hanm\_code\_with\_noise[4]^hanm\_code\_with\_noise[5]^hanm\_code\_with\_noise[6];

always@(s[2:0] or reset)

begin

if(reset)

//异步清零

begin

e=7'd0;

error\_pointer=3'b111;

end

else

//根据校正子判断有无误码和误码位置

case(s[2:0])

3'b000:

begin

e=7'b000\_0000;

error\_pointer=3'b111;

ne=0;

end

3'b001:

begin

e=7'b000\_0001;

error\_pointer=3'b000;

ne=1;

end

3'b010:

begin

e=7'b000\_0010;

error\_pointer=3'b001;

ne=1;

end

3'b100:

begin

e=7'b000\_0100;

error\_pointer=3'b010;

ne=1;

end

3'b011:

begin

e=7'b000\_1000;

error\_pointer=3'b011;

ne=1;

end

3'b101:

begin

e=7'b001\_0000;

error\_pointer=3'b100;

ne=1;

end

3'b110:

begin

e=7'b010\_0000;

error\_pointer=3'b101;

ne=1;

end

3'b111:

begin

e=7'b100\_0000;

error\_pointer=3'b110;

ne=1;

end

default:

begin

e=7'b000\_0000;

error\_pointer=3'b111;

ne=0;

end

endcase

end

//输出汉明吗解码结果

always@ (negedge clk or posedge reset)

begin

if(reset)

begin

hanm\_dec\_out\_paral=3'b000;

end

else

begin

c=hanm\_code\_with\_noise[6:0]^e;

hanm\_dec\_out\_paral=c[6:3];

end

end

endmodule

**7、并串转换（out\_serial）**

module out\_serial

(

clk,

reset,

hanm\_dec\_out\_paral,

hanm\_dec\_out\_serial

);

input clk;

input reset;

input [3:0] hanm\_dec\_out\_paral;

output hanm\_dec\_out\_serial;

reg hanm\_dec\_out\_serial;

reg [3:0] temp=4'b0000;

integer cnt=0;

always@ (posedge clk or posedge reset)

begin

if(reset)

//异步清零

begin

hanm\_dec\_out\_serial=1'b0;

cnt=0;

end

else

//并串转换

if(cnt==3)

begin

temp=hanm\_dec\_out\_paral;

hanm\_dec\_out\_serial=temp[3];

cnt=0;

end

else

begin

temp={temp[2:0],temp[3]};

hanm\_dec\_out\_serial=temp[3];

cnt=cnt+1;

end

end

endmodule

附录二：

