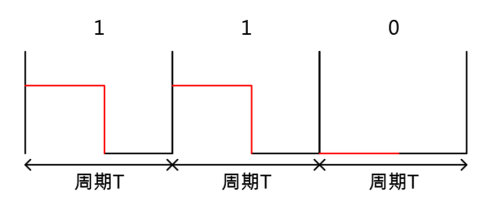
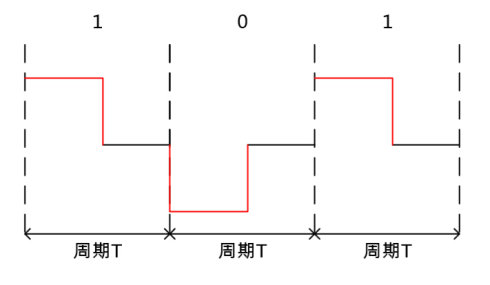
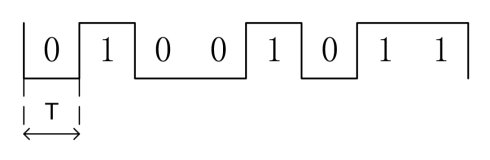
1. 编码类型

1.RZ(Return Zero Code)编码

[](http://photo.blog.sina.com.cn/showpic.html#blogid=78e87ba10102wj9g&url=http://album.sina.com.cn/pic/002dho0Vzy754gdZFkdf1)

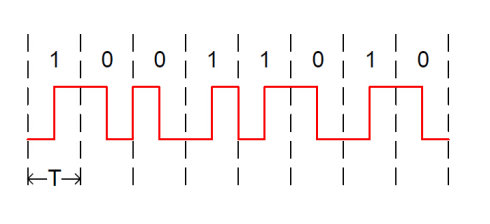
[](http://photo.blog.sina.com.cn/showpic.html#blogid=78e87ba10102wj9g&url=http://album.sina.com.cn/pic/002dho0Vzy754gI8T9P3d)

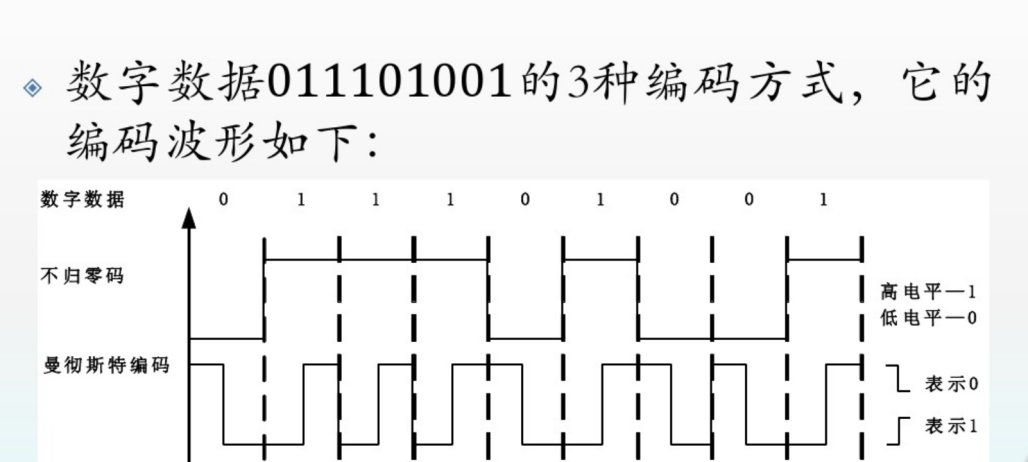
2.NRZ(Non Return Zero Code)编码



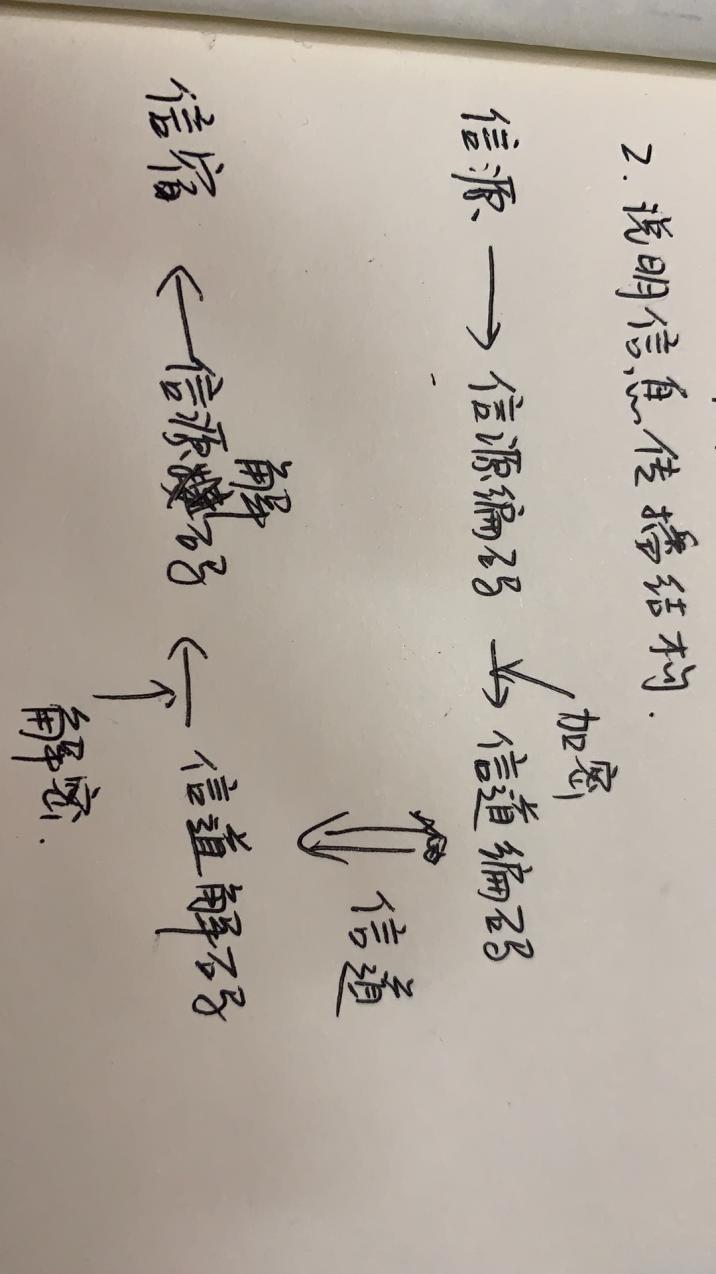
3.曼彻斯特编码

曼彻斯特编码方式和NRZI编码十分相似，只不过它是利用信号的跳变方向来决定数据的。在位中间，信号由高向低跳变表示数据0，信号由低向高跳变表示数据1。举个图例吧，若要表示数据1001 1010(B)，则信号波形图如下图所示：





二、信息传输过程



信源编码：目的在于减少信息的冗余度，压缩消息以便占用最少的资源，提高传输效率，提高有效性。

信道编码：目的在于增加一部分冗余度，通过牺牲有效性来换取信息传递的可靠性。

曼彻斯特编码是一种信源编码

三、检测上升沿代码

module posedge\_detection(clk,rst\_n,i\_data\_in,o\_rising\_edge);

input clk;

input rst\_n;

input i\_data\_in;

output o\_rising\_edge;

reg r\_data\_in0;

reg r\_data\_in1;

assign o\_rising\_edge=r\_data\_in0&~r\_data\_in1;

always@(posedge clk or negedge rst\_n)

begin

if(rst\_n==1'b0)

begin

r\_data\_in0<=0;

r\_data\_in1<=0;

end

else

begin

r\_data\_in1<=r\_data\_in0;

r\_data\_in0<=i\_data\_in;

end

end

endmodule

四、memory数组赋值讲解

存储器是一个寄存器数组。存储器使用如下方式说明：

reg [ msb: lsb] memory1 [ upper1: lower1],  
memory2 [upper2: lower2],. . . ；

例如：  
reg [0:3 ] MyMem [0:63]  
//MyMem为64个4位寄存器的数组。  
reg Bog [1:5]  
//Bog为5个1位寄存器的数组。

MyMem和Bog都是存储器。数组的维数不能大于2。

注意存储器属于寄存器数组类型。线网数据类型没有相应的存储器类型。

parameter ADDR\_SIZE = 16 , WORD\_SIZE = 8;  
reg [1: WORD\_SIZE] RamPar [ ADDR\_SIZE－1 : 0], DataReg;

RamPar是存储器，是16个8位寄存器数组，而DataReg是8位寄存器。  
　　在赋值语句中需要注意如下区别：存储器赋值不能在一条赋值语句中完成，但是寄存器可以。因此在存储器被赋值时，需要定义一个索引。下例说明它们之间的不同。

reg [1:5] Dig; //Dig为5位寄存器。  
Dig = 5'b11011;

上述赋值都是正确的, 但下述赋值不正确：

reg Bog[1:5]; //Bog为5个1位寄存器的存储器。  
Bog = 5'b11011;

有一种存储器赋值的方法是分别对存储器中的每个字赋值。例如：

reg [0:3] Xrom [1:4]  
. . .  
Xrom[1] = 4'hA;  
Xrom[2] = 4'h8;  
Xrom[3] = 4'hF;  
Xrom[4] = 4'h2;

为存储器赋值的另一种方法是使用系统任务：  
1) $readmemb （加载二进制值）  
2) $readmemb （加载十六进制值）  
这些系统任务从指定的文本文件中读取数据并加载到存储器。文本文件必须包含相应的二进制或者十六进制数。例如：

reg [1:4] RomB [7:1] ;  
$ readmemb ("ram.patt", RomB);

Romb是存储器。文件“ram.patt”必须包含二进制值。文件也可以包含空白空间和注释。下面是文件中可能内容的实例。

1101  
1110  
1000  
0111  
0000  
1001  
0011

　　系统任务$readmemb促使从索引7即Romb最左边的字索引，开始读取值。如果只加载存储器的一部分，值域可以在$readmemb方法中显式定义。例如：

$readmemb ("ram.patt", RomB, 5, 3);

在这种情况下只有Romb[5],Romb[4]和Romb[3]这些字从文件头开始被读取。被读取的值为1101、1100和1000。  
文件可以包含显式的地址形式。

@hex\_address value  
如下实例：  
@5 11001  
@2 11010

在这种情况下，值被读入存储器指定的地址。  
　　当只定义开始值时，连续读取直至到达存储器右端索引边界。例如：

$readmemb ("rom.patt", RomB, 6);  
//从地址6开始，并且持续到1。  
$readmemb ( "rom.patt", RomB, 6, 4);  
//从地址6读到地址4。