

《红外线遥控系统》设计

至芯科技论坛—www.fpgaw.com

至芯科技教研部

李昭

2017-6-20



至芯科技
ZHI XIN TECHNOLOGY

FPGA 培训专家

www.zxopen.com

至芯科技官网: www.zxopen.com

至芯科技技术论坛: www.fpgaw.com

至芯科技淘宝网址:

<https://shop101836044.taobao.com/?spm=a230r.7195193.1997079397.2.9gJ436>

至芯科技腾讯课堂:

<https://ke.qq.com/course/list/%E8%87%B3%E8%8A%AF%E7%A7%91%E6%8A%80>

至芯科技-fpag 交流群(QQ): 282124839

至芯科技 fpga 就业班火爆招生中, 全国统一咨询热线: 400-6810708

至芯科技论坛 www.fpgaw.com

《红外线遥控系统》设计

设计背景:

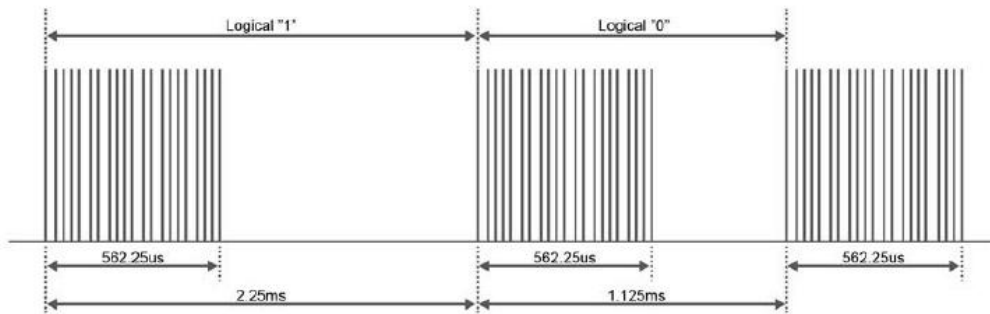
红外线(Infrared)是波长介乎微波与可见光之间的电磁波,波长在 760 纳米(nm)至 1 毫米(mm)之间,比红光长的非可见光。红外线遥控是目前使用最广泛的一种通信和遥控手段。由于红外线遥控装置具有体积小、功耗低、功能强、成本低等特点,因而,继彩电、录像机之后,在录音机、音响设备、空调机以及玩具等其它小型电器装置上也纷纷采用红外线遥控。现在工业设备中,也已经广泛在使用。。。。。

设计原理:

红外遥控系统主要由红外的发送装置和接收装置组成,发送装置可由按键,编码模块,发射电路等组成,接收装置由红外接收电路,遥控,解码模块等组成,此次设计我们用到的硬件平台是 Altera 的 DE1_SOC,晶振为 50MHZ。

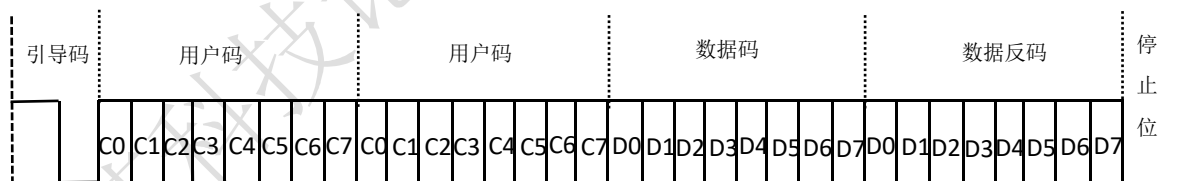
在红外的编码中,我们对 1 和 0 的编码是通过 38KHZ 的脉冲来

定义的, 在红外的编码中每个脉冲的为 256. 25us 长的 38KHZ 载波频率 (26. 3us), 对 0, 1 的脉冲的定义的时间如下图



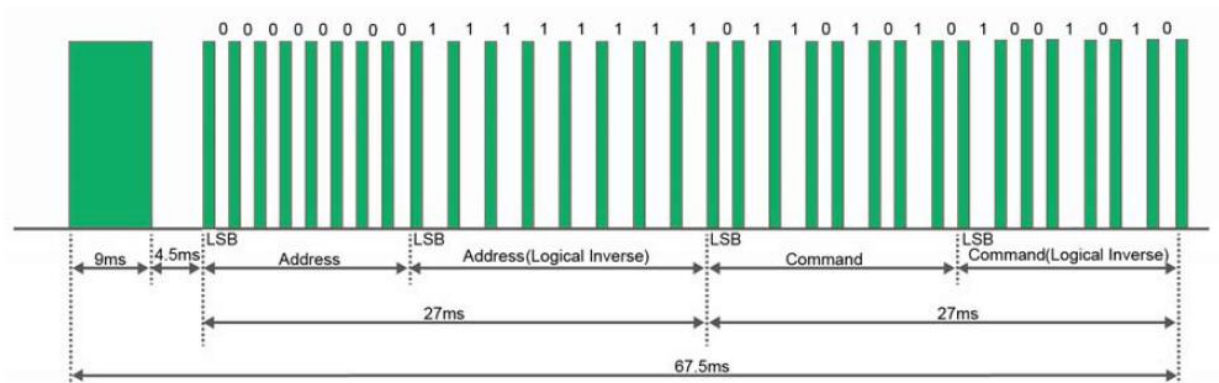
红外的数据格式为包括引导码, 用户码, 数据码和数据纠错码, 停止位编码总为 32 位。数据反码是数据码反相后的编码, 可用于对数据的纠错。此外第二段的用户码可以在遥控应用电路中设置为第一段用户码的反码。

数据格式如下图:



























一帧数据在发送时先发送 9MS 的高电平, 然后发送 4. 5MS 的低电平的起始位, 然后发送用户码, 数据码, 数据反码。然后再发送一位的停止位。不发送数据时数据线一直为低。

发送的时序图如下:



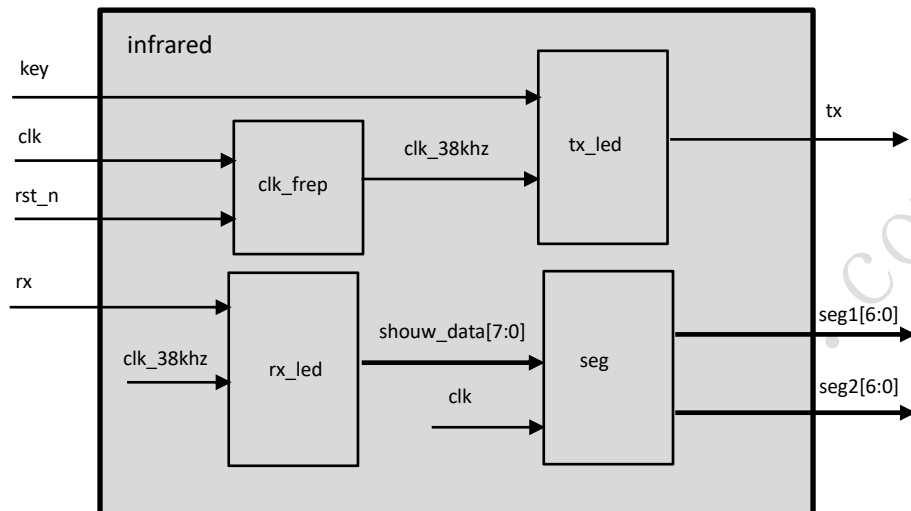
接受的时, 接收到的时序和发送的时序恰恰相反, 如发送时先发送 9ms 的高, 4.5ms 的低, 接收为接收 9ms 的低电平, 4.5ms 低电平。

接收的控制器我们用的时红外遥控装置, 按键发送的数据如下图所示

Key	Key Code	Key	Key Code	Key	Key Code	Key	Key Code
	0x0F		0x13		0x10		0x12
	0x01		0x02		0x03		0x1A
	0x04		0x05		0x06		0x1E
	0x07		0x08		0x09		0x1B
	0x11		0x00		0x17		0x1F
	0x16		0x14		0x18		0x0C

设计架构图:

设计的总框架如下图：



在我们的设计中分频模块提供所需要的 38KHZ 的时钟，当按键按下时发送我们的发送模块发送一个给定的数值，我的设计中用户码为 8' b0 第二段用户码为 8' hff, 然后发送给定的数据码，和数据反码。上电后我们的设计会发一次我们给定的数据码，然后在接受模块会接受到其发送的数据并在数码管上显示出来，之后我们可以用我们的遥控键盘来发送数据，接收模块接收显示出来，通过验证我们接收和发送的正确。

设计代码：

顶层代码：



```
00 module infrared(clk, rst_n, key, tx, seg1, seg2, rx);
01
02 input clk, rst_n;
03 input key;
04 output tx;
05 input rx;
06 wire [7:0] show_data;
07 output [7:0] seg1, seg2;
08 wire [31:0] data_n;
09 wire clk_38khz;
10
11
12 clk_frep clk_frep_dut( //分频模块
13     .clk(clk),
14     .rst_n(rst_n),
15     .clk_38khz(clk_38khz)
16 );
17
18
19 tttxxx tx_dut( //发送模块
20     .clk(clk_38khz),
21     .rst_n(rst_n),
22     .data_n(data_n),
23     .tx(tx),
24     .key(key)
25 );
26
27 seg seg01( //数码管模块
28     .clk(clk),
29     .rst_n(rst_n),
30     .seg7(seg1),
31     .data_in(show_data[3:0])
32 );
33
34 seg seg02(
35     .clk(clk),
36     .rst_n(rst_n),
37     .seg7(seg2),
38     .data_in(show_data[7:4])
39 );
```



```
40
41 rx_led led_dut( //接收模块
42     .clk(clk_38khz),
43     .rst_n(rst_n),
44     .rx(rx),
45     .show_data(show_data)
46 );
47
48
49 endmodule
```

发送模块:

```
000 module tttxxx(clk, rst_n, data_n, tx, key);
001
002 input clk, rst_n;
003 input key;
004 input [31:0] data_n;
005 output reg tx;
006
007 parameter T9ms = 342; //9000/26.3
008 parameter T4500us = 171; //4.5ms 4500/26.3
009 parameter T0 = 21; //((1125-562.25)/26.3
010 parameter T1 = 63; //((2250-562.25)/26.3
011 parameter T562us = 21; // 562.25/26.3;
012 parameter T = 2666; // 一帧数据
013 reg T9_flag;
014 reg T45_flag;
015 reg T0_flag;
016 reg T1_flag;
017 reg T9_down;
018 reg T45_down;
019 reg T0_down;
020 reg T1_down;
021 reg [9:0] cnt9;
022 reg [9:0] cnt45;
023 reg [9:0] cnt0;
024 reg [9:0] cnt1;
025 reg [9:0] cnt562;
026 reg t0_clk, t1_clk;
```




```
027 reg [8:0] count;
028 reg [2:0] state;
029 reg data;
030 reg [31:0] d_data;
031
032 always @ (posedge clk)
033     if(!rst_n)
034         begin
035             count <= 0;
036             state <= 0;
037             tx <= 0;
038             d_data <= {8'b0,8'hff, 8'b10100010, 8'b01011101}; //默认发送数据
039         end
040     else
041         case (state)
042             0 : if(count < 10)
043                 begin
044                     tx <= 0;
045                     count <= count + 1;
046                 end
047             else if(!key)
048                 begin
049                     count <= 0;
050                     state <= 1;
051                     T9_flag <= 1;
052                     tx <= 1;
053                 end
054             1 : if(T9_down) //起始位 高电平 9 ms
055                 begin
056                     state <= 2;
057                     T45_flag <= 1;
058                     tx <= 0;
059                     T9_flag <= 0;
060                 end
061             else
062                 begin
063                     tx <= 1;
064                     state <= 1;
065                 end
```



```
066         end
067
068     2 :   if(T45_down)           // 低电平 4.5ms
069         begin
070             state <= 3;
071             tx <= 0;
072             T45_flag <= 0;
073         end
074     else
075         tx <= 0;
076
077     3 :   if(count < 32)         //32 位的数据编码, 如果那一位
078                                     为 1(0)跳转 4(5) 状态通过发送标志结束来发送出位 1 的时序
079         begin
080             count <= count + 1;
081             if(!d_data[31 - count])
082                 begin
083                     T0_flag <= 1;
084                     state <= 4;
085                     T1_flag <= 0;
086                 end
087             else
088                 begin
089                     T1_flag <= 1;
090                     state <= 5;
091                     T0_flag <= 0;
092                 end
093         end
094     else
095         begin
096             count <= 0;
097             state <= 6;
098             T0_flag <= 0;
099             T1_flag <= 0;
100         end
101     4 :   if(T0_down)           //位 0 的设置
102         begin
103             state <= 3;
104             tx <= 0;
```



```
105         end
106     else
107         begin
108             tx <= t0_clk;
109         end
110
111     5 :   if(T1_down)           //位 1 的设置
112         begin
113             state <= 3;
114             tx <= 0;
115         end
116     else
117         tx <= t1_clk;
118
119     6 :   if(count < T562us - 1) //停止位
120         begin
121             count <= count + 1;
122             tx <= 1;
123         end
124     else
125         begin
126
127             tx <= 0;
128         end
129     default: state <= 0;
130 endcase
131
132 always @ (posedge clk)           //计数一个 9ms
133     if(!rst_n)
134         begin
135             T9_down <= 0;
136             cnt9 <= 0;
137         end
138     else if (T9_flag)
139         begin
140             if(cnt9 < T9ms - 1)
141                 begin
142                     T9_down <= 0;
143                     cnt9 <= cnt9 + 1;
144                 end
```



```
145         else
146             begin
147                 T9_down <= 1;
148                 cnt9 <= 0;
149             end
150         end
151
152 always @ (posedge clk) //计数一个 4.5ms
153     if(!rst_n)
154         begin
155             T45_down <= 0;
156             cnt45 <= 0;
157         end
158     else if (T45_flag)
159         begin
160             if(cnt45 < T4500us - 1)
161                 begin
162                     T45_down <= 0;
163                     cnt45 <= cnt45 + 1;
164                 end
165             else
166                 begin
167                     T45_down <= 1;
168                     cnt45 <= 0;
169                 end
170         end
171
172 reg [9:0] cnt00;
173 always @ (posedge clk) //产生位 0 的时序
174     if(!rst_n)
175         begin
176             t0_clk <= 0;
177             T0_down <= 0;
178             cnt0 <= 0;
179             cnt00 <= 0;
180         end
181     else if (T0_flag)
182         begin
183             if(cnt0 < T562us - 1)
184                 begin
```



```
185         t0_clk <= 1;
186         cnt0 <= cnt0 + 1;
187         T0_down <= 0;
188     end
189 else
190     begin
191         if(cnt00 < T0 - 1)
192             begin
193                 cnt00 <= cnt00 + 1;
194                 t0_clk <= 0;
195                 T0_down <= 0;
196             end
197         else
198             begin
199                 T0_down <= 1;
200                 cnt0 <= 0;
201                 cnt00 <= 0;
202             end
203         end
204     end
205
206 reg [9:0] cnt11;
207 always @ (posedge clk) //产生位1的时序
208     if(!rst_n)
209         begin
210             t1_clk <= 0;
211             T1_down <= 0;
212             cnt1 <= 0;
213             cnt11 <= 0;
214         end
215     else if (T1_flag)
216         begin
217             if(cnt1 < T562us - 1)
218                 begin
219                     t1_clk <= 1;
220                     cnt1 <= cnt1 + 1;
221                     T1_down <= 0;
222                 end
223             else
224                 begin
```



```
225         if(cnt11 < T1 - 1)
226             begin
227                 cnt11 <= cnt11 + 1;
228                 t1_clk <= 0;
229                 T1_down <= 0;
230             end
231         else
232             begin
233                 T1_down <= 1;
234                 cnt1 <= 0;
235                 cnt11 <= 0;
236             end
237         end
238     end
239
240 endmodule
```

接收模块:

```
0  module rx_led(clk, rst_n, rx, show_data);
1
2  input clk, rst_n;
3  input rx;
4  output reg [7:0] show_data;
5
6  reg [1:0] state;
7  reg [7:0] cnt;
8  reg temp;
9  reg [9:0] num;
10 reg flag;
11 reg [31:0] data;
12 reg [1:0] state_s;
13 reg flag_x;
14 reg [12:0] count;
15
16 parameter T = 2566; // 一帧数据的时间
17
18 //这个模块是中因为接受的 32 位编码数据中, 不管是位 0 还是位 1, 接受的低电平都是相
19 //我们可以通过来判断高电平的时间来确定为位 1 还是位 0, 位' 1 ' 1.68MS, 位 0
```



```
562.25us
20 always @ (posedge clk)
21     if(!rst_n)
22         begin
23             num <= 0;
24             data <= 0;
25             state_s <= 0;
26             flag_x <= 0;
27             count <= 0;
28         end
29     else
30         begin
31             case (state_s)
32                 0 : if(!rx) //判断起始位, 是否接
受=收数据
33                     begin
34                         state_s <= 1;
35                         flag_x <= 0;
36                         count <= count + 1;
37                     end
38                 else
39                     begin
40                         flag_x <= 0;
41                         state_s <= 0;
42                         count <= count + 1;
43                     end
44
45                 1 : if(num < (342 + 171 - 1)) //延迟 9ms + 4.5ms
的起始时间
46                     begin
47                         num <= num + 1;
48                         state_s <= 1;
49                         count <= count + 1;
50                     end
51                 else
52                     begin
53                         num <= 0;
54                         state_s <= 2;
55                         count <= count + 1;
56                     end
end
```



```
57
58      2 : if(flag && num < 32) //flag 来的时候表示接到
        了位 1 , 或者位 0,
59                                     //通过移位寄存器来获取
        32 位数据
60         begin
61             data <= {data[30:0],temp};
62             state_s <= 2;
63             num <= num + 1;
64             count <= count + 1;
65         end
66     else if(num == 32)
67         begin
68             state_s <= 3;
69             num <= 0;
70             count <= count + 1;
71         end
72     else
73         state_s <= 2;
74
75     3 : if(num < 21 - 1) //延迟结束位的时间
76         begin
77             num <= num + 1;
78             count <= count + 1;
79         end
80     else
81         begin
82             if(count == T - 1) //延迟一帧数据
            的时间后, 发送一个标志位
83                 begin
84                     num <= 0;
85                     state_s <= 0;
86                     flag_x <= 1;
87                     count <= 0;
88                     count <= count + 1;
89                 end
90             else
91                 count <= count + 1;
92         end
93     default: state_s <= 0;
```




```
94         endcase
95     end
96
97 always @ (posedge clk)
98     if(!rst_n)
99         begin
100             cnt <= 0;
101             state <= 0;
102             temp <= 0;
103             flag <= 0;
104         end
105     else
106         if(state_s > 1 && state_s < 3)
107             case (state)
108                 0 : if(rx)
109                     begin
110                         cnt <= cnt + 1;
111                         state <= 1;
112                         flag <= 0;
113                     end
114                 else
115                     begin
116                         state <= 0;
117                         flag <= 0;
118                     end
119
120                 1 : if(!rx)
121                     begin
122                         cnt <= cnt;
123                         state <= 2;
124                     end
125                 else
126                     cnt <= cnt + 1;
127
128                 2 : if(400 < cnt * 26 && cnt * 26 < 600) //判断
高电平的时间
129                     begin
130                         temp <= 0;
131                         flag <= 1;
132                         state <= 0;
```



```
133         cnt <= 0;
134     end
135     else if (1400 < cnt * 26 && cnt * 26 < 1700) //
判断高电平的时间
136         begin
137             temp <= 1;
138             flag <= 1;
139             state <= 0;
140             cnt <= 0;
141         end
142     else
143         begin
144             state <= 0;
145             cnt <= 0;
146         end
147     default: state <= 0;
148 endcase
149
150 always @ (*) //接收完一帧数据后,当标志位来的时候通过对数据的纠错来
捕获数据
151 //我们接收的数据用的是左移,而发送的时候先发的是低位
152     if(!rst_n)
153         show_data <= 0;
154     else if((data[7:0] == ~data[15:8]) && (data[31:24] == ~data[23:16])
&& flag_x)
155         begin
156             show_data[0] <= data[15];
157             show_data[1] <= data[14];
158             show_data[2] <= data[13];
159             show_data[3] <= data[12];
160             show_data[4] <= data[11];
161             show_data[5] <= data[10];
162             show_data[6] <= data[9];
163             show_data[7] <= data[8];
164
165         end
166     else
167         show_data <= show_data;
168
169 endmodule
```



数码管模块:

```
0  module seg(clk, rst_n, seg7, data_in);
1
2  input clk;
3  input rst_n;
4  input [3:0] data_in;
5
6  output reg [7:0] seg7;
7
8
9  `define T1ms 50_000 //分频出 1k 的时钟
10 //`define T1ms 5
11 reg [15:0] count;
12 reg flag;
13 always @ (posedge clk or negedge rst_n)
14     if(!rst_n)
15         begin
16             count <= 15'b0;
17             flag <= 1;
18         end
19     else
20         if(count == `T1ms /2 - 1)
21             begin
22                 count <= 15'b0;
23                 flag <= ~flag;
24             end
25         else
26             begin
27                 count <= count + 1'b1;
28             end
29
30 always @ (posedge flag)
31     if(!rst_n)
32         seg7 <= 8'b1010_0100;
33     else
34         begin
35             case (data_in)
36                 0:seg7 <= 8'b1100_0000;
```



```
37         1:seg7 <= 8'b1111_1001;
38         2:seg7 <= 8'b1010_0100;
39         3:seg7 <= 8'b1011_0000;
40         4:seg7 <= 8'b1001_1001;
41         5:seg7 <= 8'b1001_0010;
42         6:seg7 <= 8'b1000_0010;
43         7:seg7 <= 8'b1111_1000;
44         8:seg7 <= 8'b1000_0000;
45         9:seg7 <= 8'b1001_0000;
46         10:seg7 <= 8'b1000_1000;
47         11:seg7 <= 8'b1000_0011;
48         12:seg7 <= 8'b1100_0110;
49         13:seg7 <= 8'b1010_0001;
50         14:seg7 <= 8'b1000_0110;
51         15:seg7 <= 8'b1000_1110;
52         default;;
53     endcase
54 end
55 endmodule
```

分频模块:

```
0  module clk_frep(clk, rst_n, clk_38khz);
1
2  input clk, rst_n;
3  output reg clk_38khz;
4
5  reg [9:0] count;
6
7  //分频出红外模块所用的 38Khz 的时钟
8  //也可以用占空比为 1:3 的 38khz 的时钟
9
10 always @ (posedge clk or negedge rst_n)
11     if(!rst_n)
12         begin
13             count <= 0;
14             clk_38khz <= 1;
15         end
16     else if(count == (50_000_000 / 38000 / 2 - 1))
17         begin
```



```
18         clk_38khz <= ~clk_38khz;
19         count <= 0;
20     end
21 else
22     count <= count + 1'd1;
23
24 endmodule 8
```

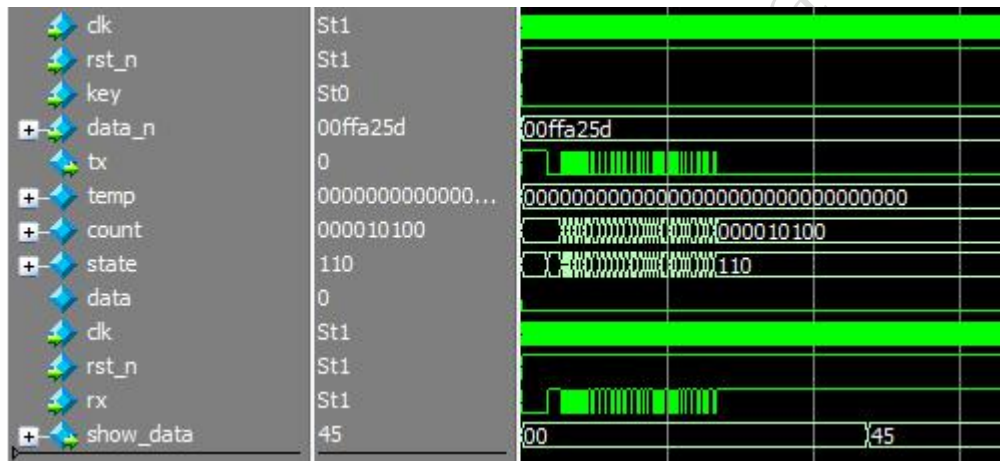
测试模块:

```
0  `timescale 1ns/1ps
1
2  module infrared_tb();
3
4  reg clk, rst_n;
5  reg key;
6  wire tx;
7  wire [7:0] show_data;
8
9  //因为我们代码中只发送一次数据，所以可以把 key 一直拉低
10
11  initial begin
12      clk = 1;
13      rst_n = 0;
14      key = 1;
15
16      #100.1 rst_n = 1;
17
18      #200 key = 0;
19
20  end
21
22  always # 10 clk = ~clk;
23
24  infrared dut(
25      .clk(clk),
26      .rst_n(rst_n),
27      .key(key),
28      .tx(tx),
29      .rx(rx),
30      .seg1(seg1),
```

```
31     .seg2(seg2)
32 );
33
34 endmodule
```

仿真图:

仿真中我们可以把数码管模块的计数器的值改小一点，便于仿真



如图中所示的我们发的是 32' h00ffa25d，那么数据为是 8' b1010_0010, 那么先发送时就时就按下面的序列开始 0100_0101 接收到的为 45，所以工程正确。