实验八

1. 实验内容

实现一个简易的音乐播放器

1. 实验原理

将不同音符对应的频率与长度信息写入mif,生成rom

1. Mif

21 21 21 21 21 21 21 21

0 17 17 17 17 17 17 17

17 0 15 15 15 15 15 15

15 15 0 15 15 15 15 15

15 15 15 0 16 16 16 16

0 16 16 16 16 0 16 16

16 16 16 16 16 16 0 15

15 15 15 15 15 15 15 0

15 15 15 15 15 15 15 15

0 16 16 16 16 0 16 16

16 16 0 16 16 16 16 16

16 16 16 0 15 15 15 15

15 15 15 15 0 15 15 15

15 15 15 15 15 0 16 16

16 16 0 16 16 16 16 0

16 16 16 16 16 16 16 16

0 16 16 16 16 16 16 16

16 0 16 16 16 16 16 16

16 16 0 17 17 17 17 17

17 17 17 0 21 21 21 21

21 21 21 21 0 22 22 22

22 22 22 22 22 0 17 17

17 17 17 17 17 17 17 17

17 17 17 17 17 17 0 15

15 15 15 15 15 15 15 15

15 15 15 15 15 15 15 0

24 24 24 24 24 24 24 24

24 24 24 24 24 24 24 24

0 22 22 22 22 22 22 22

22 22 22 22 22 22 22 22

22 0 22 22 22 22 22 22

22 22 22 22 22 22 0 22

22 22 22 0 22 22 22 22

22 22 22 22 0 23 23 23

23 23 23 23 23 0 0 0

0 0 0 0 0 0 0 21

21 21 21 21 21 21 21 0

17 17 17 17 17 17 17 17

0 15 15 15 15 15 15 15

15 0 15 15 15 15 15 15

15 15 0 16 16 16 16 0

16 16 16 16 0 16 16 16

16 16 16 16 16 0 15 15

15 15 15 15 15 15 0 15

15 15 15 15 15 15 15 0

16 16 16 16 0 16 16 16

16 0 16 16 16 16 16 16

16 16 0 16 16 16 16 16

16 16 16 0 16 16 16 16

16 16 16 16 0 17 17 17

17 17 17 17 17 0 21 21

21 21 21 21 21 21 0 22

22 22 22 22 22 22 22 0

16 16 16 16 16 16 16 16

0 16 16 16 16 16 16 16

0 0 0 0 0 0 0 0

16 16 16 16 16 16 16 16

0 15 15 15 15 15 15 15

15 0 16 16 16 16 16 16

16 16 0 17 17 17 17 17

17 17 17 0 21 21 21 21

21 21 21 21 0 22 22 22

22 22 22 22 22 0 17 17

17 17 17 17 17 17 17 17

17 17 17 17 17 17 0 15

15 15 15 15 15 15 15 15

15 15 15 15 15 15 15 0

13 13 13 13 13 13 13 13

13 13 13 13 13 13 13 13

0 15 15 15 15 15 15 15

15 15 15 15 15 15 15 15

15 0 15 15 15 15 15 15

15 15 0 16 16 16 16 0

16 16 16 16 0 16 16 16

16 16 16 16 16 0 21 21

21 21 21 21 21 21 0 17

17 17 17 17 17 17 17 0

15 15 15 15 0 13 13 13

13 0 13 13 13 13 13 13

13 13 0 15 15 15 15 15

15 15 15 0 15 15 15 15

15 15 15 15 0 16 16 16

16 16 16 16 16 16 16 16

16 16 16 16 16 16 16 16

16 16 16 16 16 16 16 16

16 16 16 16 16 16 16 16

16 16 16 16 16 16 16 16

16 16 16 16 16 16 16 16

16 16 16 16 16 16 16 16

16 16 16 16 16 16 16 16

由于有很多同音的音符，为了体现出节奏，在音符之间插入了0来进行停顿。为了使停顿不过于明显，选择将原来音符的长度按比例加长，此时要求读入速度更快。

1. 代码

分频

module div(in\_clk,rst,clk\_4,addr);

input in\_clk,rst;

output reg [9:0]addr;

output reg clk\_4;

reg [31:0] count\_4;

always@(posedge in\_clk or negedge rst)

begin

if(!rst)

begin

clk\_4<=1'b0;

count\_4<=32'b0;

end

else if(count\_4==50000000/(8\*4\*2)-1)

begin

clk\_4<=~clk\_4;

count\_4<=32'd0;

end

else count\_4<=count\_4+32'd1;

end

always@(posedge clk\_4 or negedge rst)

begin

if(!rst) addr<=9'd0;

else if(addr==9'd420) addr<=9'd0;

else addr<=addr+9'd1;

end

endmodule音符读入

module player(iclk,code,speaker);

input iclk;

input [7:0] code;

output speaker;

reg [18:0]Tone;

reg [18:0]cnt;

reg clk\_temp;

assign speaker=clk\_temp;

always@(posedge iclk)

begin

case(code)

5'd0:Tone<=134;

5'd01:Tone<=494 ;

5'd02:Tone<=554 ;

5'd03:Tone<=622 ;

5'd04:Tone<=659 ;

5'd05:Tone<=740 ;

5'd06:Tone<=831 ;

5'd07:Tone<=932 ;

5'd11:Tone<=988 ;

5'd12:Tone<=1109 ;

5'd13:Tone<=1245 ;

5'd14:Tone<=1318 ;

5'd15:Tone<=1480 ;

5'd16:Tone<=1661 ;

5'd17:Tone<=1865 ;

5'd21:Tone<=1976 ;

5'd22:Tone<=2218 ;

5'd23:Tone<=2490 ;

5'd24:Tone<=2636 ;

5'd25:Tone<=2960 ;

5'd26:Tone<=2322 ;

5'd27:Tone<=3730 ;

default: Tone<=134;

endcase

if(cnt==(50000000/Tone)/2-1)

begin

clk\_temp=~clk\_temp;

cnt<=0;

end

else cnt<=cnt+1;

end

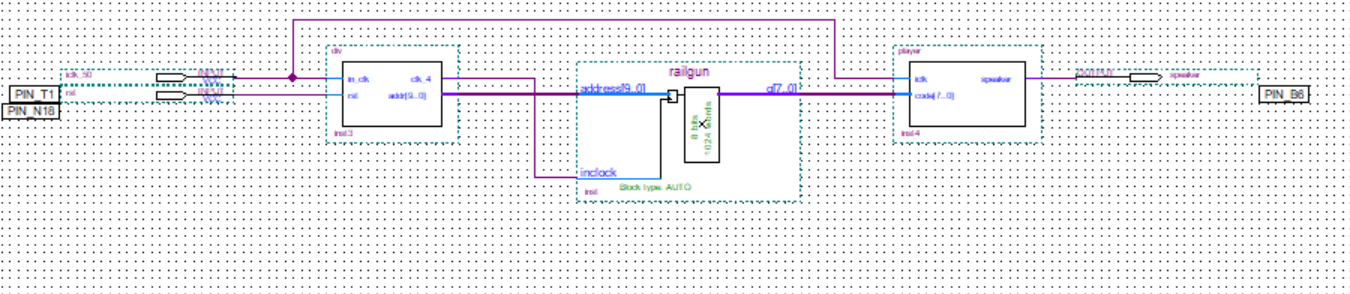
endmodule

endmodule

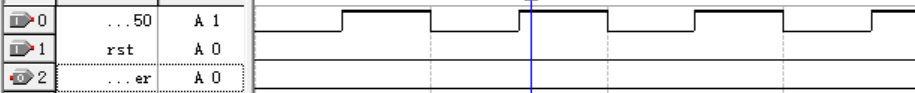
由于歌曲的音调不同（本首歌为B大调），重新计算出各个音符的频率

每八度音符的频率乘2

顶层



1. 仿真



1. RTL



