

FPGA实验报告

|  |  |
| --- | --- |
| 实验题目 | 硬件电子琴电路设计 |
| 专业班级 | 电子卓越1401 |
| 学生姓名 | 王春城 |
| 学 号 | 5120140364 |
| 教 师 | 龙惠民 |

西南科技大学信息工程学院

2017年4月

硬件电子琴电路设计

**一、实验目的**

（1）熟悉ISE Foundation设计流程和软件操作，使用FPGA完成复杂的数字系统设计；

（2）掌握基于Verilog的组合和时序逻辑电路的设计方法；

（3）学习利用数控分频器设计硬件电子琴实验。

**二、实验要求**

设计的电子琴系统需满足：

（1）正确播放“梁祝”乐曲；

（2）播放其他音乐；

（3）实验下载和硬件调试。

**三、实验原理**

（一）播放原理

要实现系统设计，分如下4步操作：

1、将待播放的《梁祝》音乐音符存入ROM；

2、以4HZ频率读取ROM ；

3、根据1MHZ的内部时钟要求，将读出的音符换算成应计数的数值；

4、以1MHZ为内部时钟，实现符合要求的方波信号。

由于系统时钟是50Mhz，所以我们为了实现1Mhz的时钟信号，故将50M分成12.5Mhz，后续在得到近似1MHZ时钟。故按照“自上而下”设计原则，系统框图1如示。

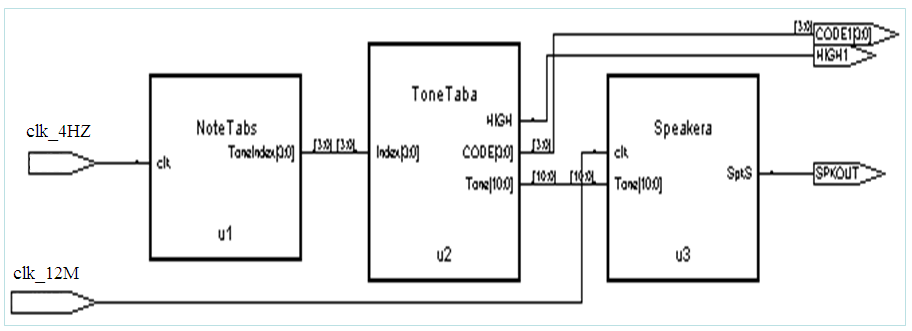


图1 硬件电子琴电路设计方案

（二）任务分析

**1．输入端口：**

（1）复位信号RESET，当RESET=1时，输出全部置零；当RESET=1，系统正常工作；

（2）时钟信号CLK\_50M，CLK\_50M用于产生4Hz和12.5MHz分频时钟信号。

**2．输出端口：**

（1）Code[3]~Code[0]是琴音简谱码；

（2）High是高8度端口，表示音阶的高低；

（3）Spkout表示音符的频率，外接蜂鸣器端口。

按照自顶向下设计，应该分为以下模块：

**3．分频模块**

将下载板上50MHz时钟分频为频率是4Hz和12.5Mhz的时钟信号，其中4Hz用于读取Rom，12.5MHz在后续模块中在分频得到近似1MHz时钟。

**4．音阶发生器模块**

模块ToneTaba是音阶发生器，当8位发声控制输入Index中某一位为高电平时，则对应某一音阶的数值将从端口Tone输出，作为获得该音阶的分频预置值；同时由Code输出对应该音阶简谱的显示数码，如‘5’，并由High输出指示音阶高8度显示。

**5．数控分频器模块**

模块Speakera中的主要电路是一个数控分频器，它由一个初值可预置的加法计数器构成，当模块Speakera由端口Tone获得一个2进制数后，将以此值为计数器的预置数，对端口Clk12MHZ输入的频率进行分频，之后由Spkout向扬声器输出发声。

**6．乐曲自动演奏电路模块**

在原设计的基础上，增加一个Notetabs模块用于产生节拍控制（Index数据存留时间）和音阶选择信号，即在Notetabs模块放置一个乐曲曲谱真值表，乐曲曲谱真值表放置于名为Music的ROM模块中。

由一个计数器来生成读取ROM所需的地址数据，对ROM以4HZ的频率进行读取，实现控制此真值表的输出，而由此计数器的计数时钟信号作为乐曲节拍控制信号，从而可以设计出一个纯硬件的乐曲自动演奏电路。

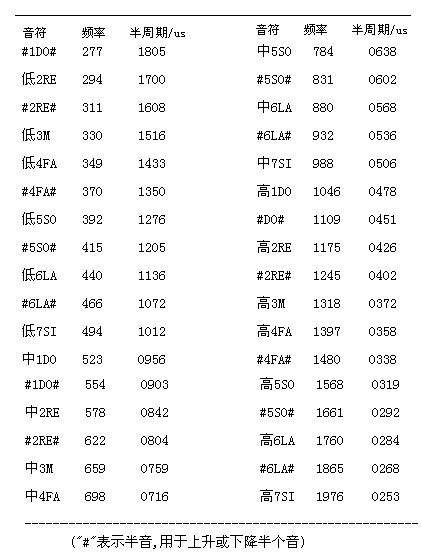


图2 音符半周期及频率对应图

**四、实验步骤**

**（一）**模块设计

利用Verilog HDL语言，对各个模块分别进行设计、仿真和综合。

**1．分频代码（12.5MHz）**

`timescale 1ns / 1ps

module CLK\_12M(CLK\_50,CLR,CLK\_12M);

input CLK\_50,CLR;

output CLK\_12M;

reg CLK\_12M;

reg [3:0] div\_count;

always @(posedge CLK\_50 or posedge CLR)

if(CLR) div\_count <= 4'h0;

else

if(div\_count == 4'd3)//

div\_count <= 4'h0;

else div\_count <= div\_count+1'b1;

always @ (posedge CLK\_50 or posedge CLR)

if(CLR) CLK\_12M <= 1'b0;

else

if(div\_count== 0)

CLK\_12M <= 1'b1;

else

CLK\_12M <= 1'b0;

endmodule

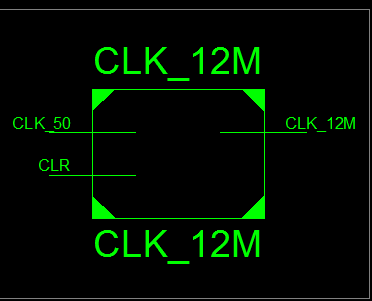


图3 分频计12.5MHz综合后RTL模块符号

测试文件

module text\_125M;

reg CLK\_50;

reg CLR;

wire CLK\_12M;

CLK\_12M uut (

.CLK\_50(CLK\_50),

.CLR(CLR),

.CLK\_12M(CLK\_12M)

);

initial begin

CLK\_50 = 0;

forever #10 CLK\_50 = ~CLK\_50 ;

end

initial begin

#20 CLR = 1'b1 ;

#20 CLR = 1'b0 ;

end

endmodule

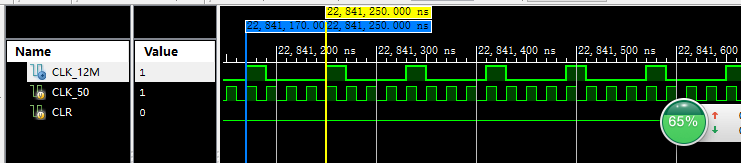


图4 分频1Hz功能仿真图

经分析，CLK\_50我给的T=20ns（即50MHz），从图可以清晰看见分频后时钟周期为4T，即12.5MHz，故仿真正确。

**2．分频4Hz频率**

`timescale 1ns / 1ps

module CLK\_4(CLK\_50,CLR,CLK\_4);

input CLK\_50,CLR;

output CLK\_4;

reg CLK\_4;

reg [29:0] div\_count;

always @(posedge CLK\_50 or posedge CLR)

if(CLR) div\_count <= 30'h0;

else

if(div\_count == 30'd12500000)//和5000万个

div\_count <= 31'h0;

else div\_count <= div\_count+1'b1;

always @ (posedge CLK\_50 or posedge CLR)

if(CLR) CLK\_4 <= 1'b0;

else

if(div\_count== 0)

CLK\_4 <= 1'b1;

else

CLK\_4 <= 1'b0;

endmodule

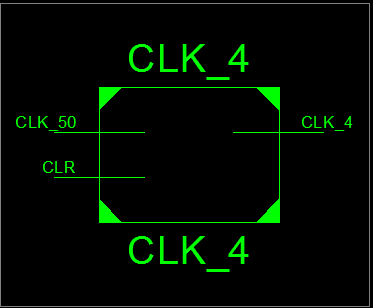


图5 分频4Hz综合后RTL模块符号

测试文件和分频12.5MHz类似，故不再赘述。

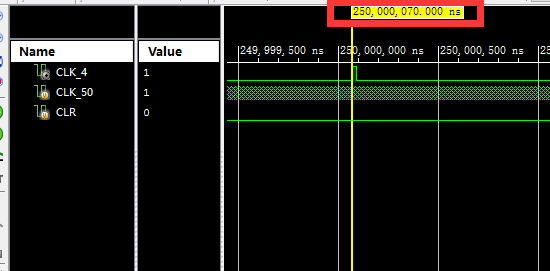


图5 分频4Hz功能仿真图

CLK\_50我给的T=20ns（即50MHz），从图中，CLK\_4在t=250ms处电平翻转，故T=0.25s仿真正确

**3．音阶发生器**

`timescale 1ns / 1ps

module ToneTaba (Index,Code,High,Tone);

input[3:0] Index;

output[3:0] Code;

output High;

output[10:0] Tone;

reg[3:0] Code;

reg High;

reg[10:0] Tone;

always @ (Index)

begin

case(Index)

4'b0000 :begin Tone<=11'b11111111111;Code<=4'b0000;High<=1'b0;end//2047 0

4'b0001 :begin Tone<=11'b01100000101;Code<=4'b0001;High<=1'b0;end//773 1

4'b0010 :begin Tone<=11'b01110010000;Code<=4'b0010;High<=1'b0;end//912 2

4'b0011 :begin Tone<=11'b10000001100;Code<=4'b0011;High<=1'b0;end//1036 3

4'b0101 :begin Tone<=11'b10010101101;Code<=4'b0101;High<=1'b0;end//1197 4

4'b0110 :begin Tone<=11'b10100001010;Code<=4'b0110;High<=1'b0;end//1290 5

4'b0111 :begin Tone<=11'b10101011100;Code<=4'b0111;High<=1'b0;end//1372; 7

4'b1000 :begin Tone<=11'b10110000010;Code<=4'b0001;High<=1'b1;end//1410; 8

4'b1001 :begin Tone<=11'b10111001000;Code<=4'b0010;High<=1'b1;end//1480; 9

4'b1010 :begin Tone<=11'b11000000110;Code<=4'b0011;High<=1'b1;end//1542; 10

4'b1100 :begin Tone<=11'b11001010110;Code<=4'b0101;High<=1'b1;end//1622; 12

4'b1101 :begin Tone<=11'b11010000100;Code<=4'b0110;High<=1'b1;end//1668; 13

4'b1111 :begin Tone<=11'b11011000000;Code<=4'b0001;High<=1'b1;end//1728; 15

default :begin Tone<=11'b11111111111;Code<=4'b0000;High<=1'b0;end//2047

endcase

end

endmodule

从ROM中读取音符值，在此模块中，完成Tone、琴音频谱和高8度赋值。

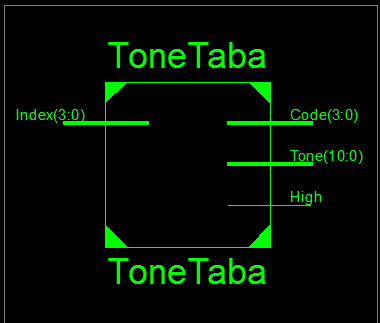


图7 ToneTaba综合后RTL模块符号

测试文件：

module test\_toteTaba;

reg [3:0] Index;

wire [3:0] Code;

wire High;

wire [10:0] Tone;

ToneTaba uut (

.Index(Index),

.Code(Code),

.High(High),

.Tone(Tone)

);

initial begin

Index = 0;

#20 Index = 4'b0000 ;

#20 Index = 4'b0001 ;

#20 Index = 4'b0010 ;

#20 Index = 4'b0011 ;

#20 Index = 4'b0100 ;

#20 Index = 4'b0101 ;

#20 Index = 4'b0110 ;

#20 Index = 4'b0111 ;

#20 Index = 4'b1000 ;

#20 Index = 4'b1001 ;

#20 Index = 4'b1010 ;

#20 Index = 4'b1011 ;

#20 Index = 4'b1100 ;

#20 Index = 4'b1101 ;

#20 Index = 4'b1110 ;

#20 Index = 4'b1111 ;

#100 $stop;

end

endmodule

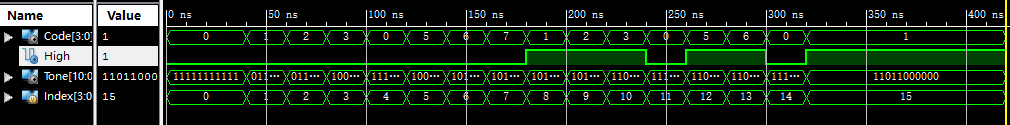


图5音阶发生器功能仿真图1

从仿真图中，我拟输入音阶0~15，其中0~7时，High=0；8、9、10、12、14、15时候，正好相反，High=1；code显示对应数值。

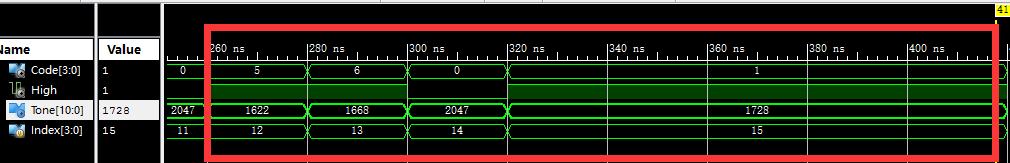


图6音阶发生器功能仿真图2

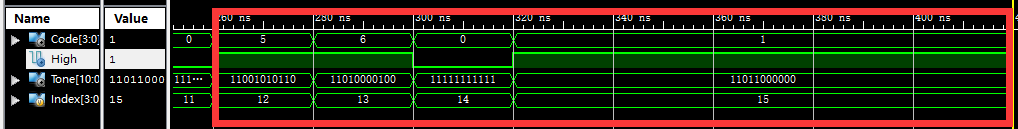


图7音阶发生器功能仿真图3

输入音阶12时，toe=11'b11001010110，即1622；输入音阶13时，toe=11'b11010000100，即1668；观察仿真图，与之符合，故音阶发生器功能模块满足设计要求。

**4．乐曲自动演奏电路**

`timescale 1ns / 1ps

module NoteTabs(Clk,ToneIndex);

input Clk;

output[3:0] ToneIndex;

reg[7:0] Counter=8'b0;

always@(posedge Clk ) begin

if(Counter>=138) Counter<=8'b00000000;

else Counter<=Counter+1'b1;

end

music1 u5( .addra(Counter),

.clka(Clk),

.douta(ToneIndex) );

endmodule

其中music1是IPCORE，用于调用ROM实现乐曲曲谱的存储，存储器位宽，width=4，depth=256。通过在NoteTabs.v文件中例化ROM文件，实现按照4HZ频率循环读取ROM的音乐节拍、频率信号。

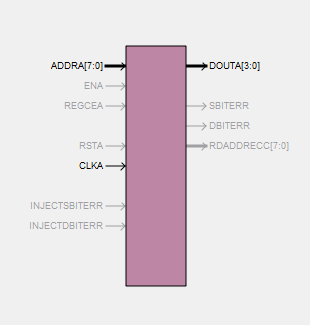


图8 IPCORE存储器

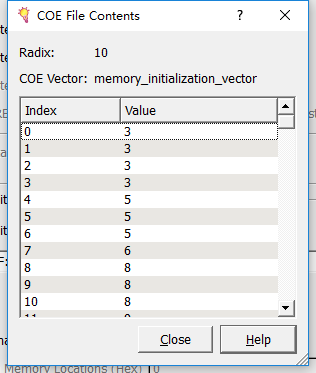


图9 存储“梁祝”乐谱

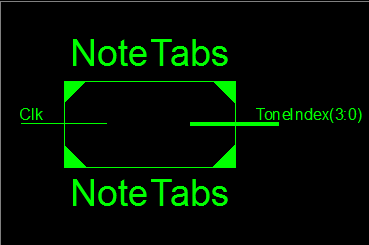


图10 NoteTabs综合后RTL模块符号

测试文件

module xxxxnote;

reg Clk;

wire [3:0] ToneIndex;

NoteTabs uut (

.Clk(Clk),

.ToneIndex(ToneIndex)

);

initial

begin

Clk = 0;

forever #10 Clk=~Clk;

end

endmodule

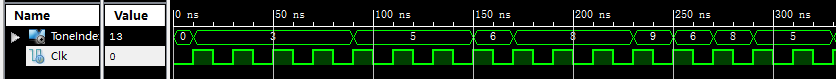


图11 NoteTabs综合仿真图

给定一个时钟信号，每一个周期读取一次ROM值，故从仿真中显示值3、3、3、3、3、5、5、5、6、8、8、8等，和梁祝乐谱（图9对比）一致，故设计符合要求。

**5．数控分频器模块**

`timescale 1ns / 1ps

module Speakera(Clk,Tone,SpkS);

input Clk;

input[10:0] Tone;

output SpkS;

reg PreClk = 1'b0;

reg FullSpkS = 1'b0;

reg[3:0] Count4 = 4'b0000;

reg[10:0]Count11 = 11'b0;

reg Count2=1'b0;

reg SpkS;

//分频1MHz

always@(posedge Clk)

begin

if(Count4>11)begin

PreClk<=1'b1;

Count4<=1;

end

else begin

PreClk<=1'b0;

Count4<=Count4+1'b1;

end

end

//音阶半周期

always@(posedge PreClk)begin

if(Count11>=11'h7FF) begin

Count11<=Tone;

FullSpkS<=1'b1;

end

else begin

Count11<=Count11+1'b1;

FullSpkS<=0;

end

end

//音阶全周期

always@(posedge FullSpkS)begin

Count2<=~Count2;

if(Count2==1'b1) SpkS<=1'b1;

else SpkS<=1'b0;

end

endmodule

按照代码，Clk是一个12M的时钟信号，进过第一个always语句时分频PreClk =1MHz（1us）；第二个always语句触发条件是1MHz时钟上升沿，所以音符半周期；第三个always语句触发条件是Fullspks时钟上升沿，所以最后。

比如“5”，Tone=1290，则音符半周期是758us，最后的输出时钟信号周期即为1516us，查表表示“中3M”音符。

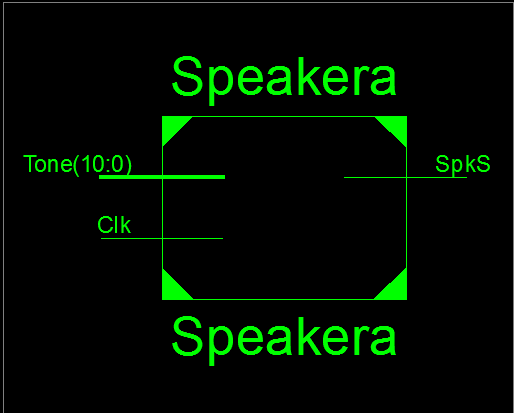


图12 数控分频器综合后RTL模块符号

激励文件

`timescale 1ns / 1ps

module test\_Speakera;

reg Clk;

reg [10:0] Tone;

wire SpkS;

Speakera uut (

.Clk(Clk),

.Tone(Tone),

.SpkS(SpkS)

);

initial begin

Clk = 0;

forever #41.6 Clk = ~Clk ;

end

initial begin

#2000 Tone = 11'b10100001010;//“5”--“中3M”音符，1290

end

endmodule

激励文件，我编写的输入时钟信号近似12MHz,直接测“5”对应的tone=1290，如果最后结果与计算相符，则设计满足要求。

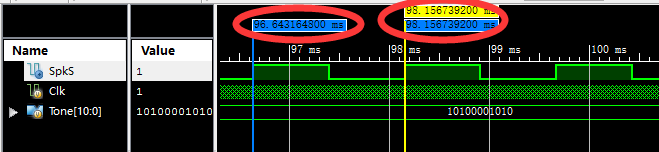


图13十进制计数器功能仿真图

经分析：

**；**

**。**

即“5”表示的半周期音符周期756us，忽略误差因素，则可推出表示“中3M”音符，故数控分频器满足设计要求。

（二）系统顶层设计

按照自顶向下的设计原则，设计代码如下

`timescale 1ns / 1ps

module songer(clk\_50M,reset,Code1,High1,Spkout);

input clk\_50M;//CLK=50MHZ

input reset;

output[3:0] Code1;

output High1,Spkout;

wire[10:0] Tone;

wire[3:0] ToneIndex;

wire clk\_4HZ,clk\_12M;

NoteTabs u0( .Clk(clk\_4HZ),

.ToneIndex(ToneIndex) );

ToneTaba u1( .Index(ToneIndex),

.Code(Code1),

.High(High1),

.Tone(Tone) );

Speakera u2( .Clk(clk\_12M),

.Tone(Tone),

.SpkS(Spkout) );

CLK\_12M u3(clk\_50M,reset,clk\_12M);

CLK\_4 u4(clk\_50M,reset,clk\_4HZ);

endmodule

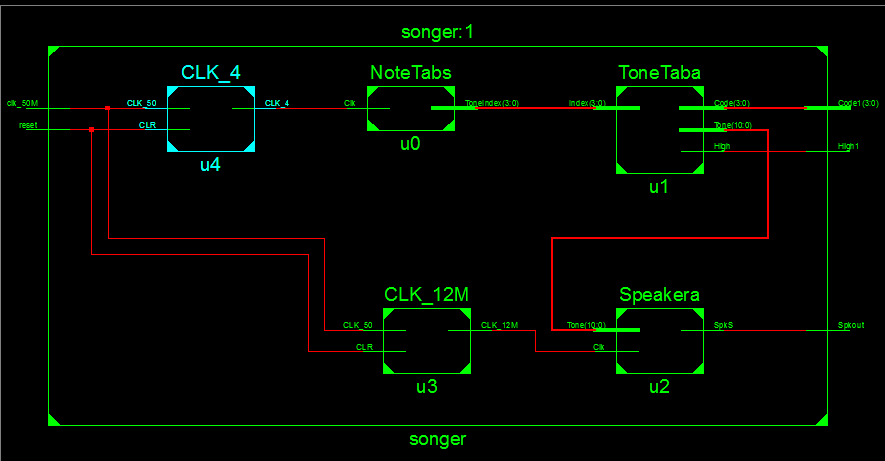


图14电子琴综合后RTL内部逻辑模块

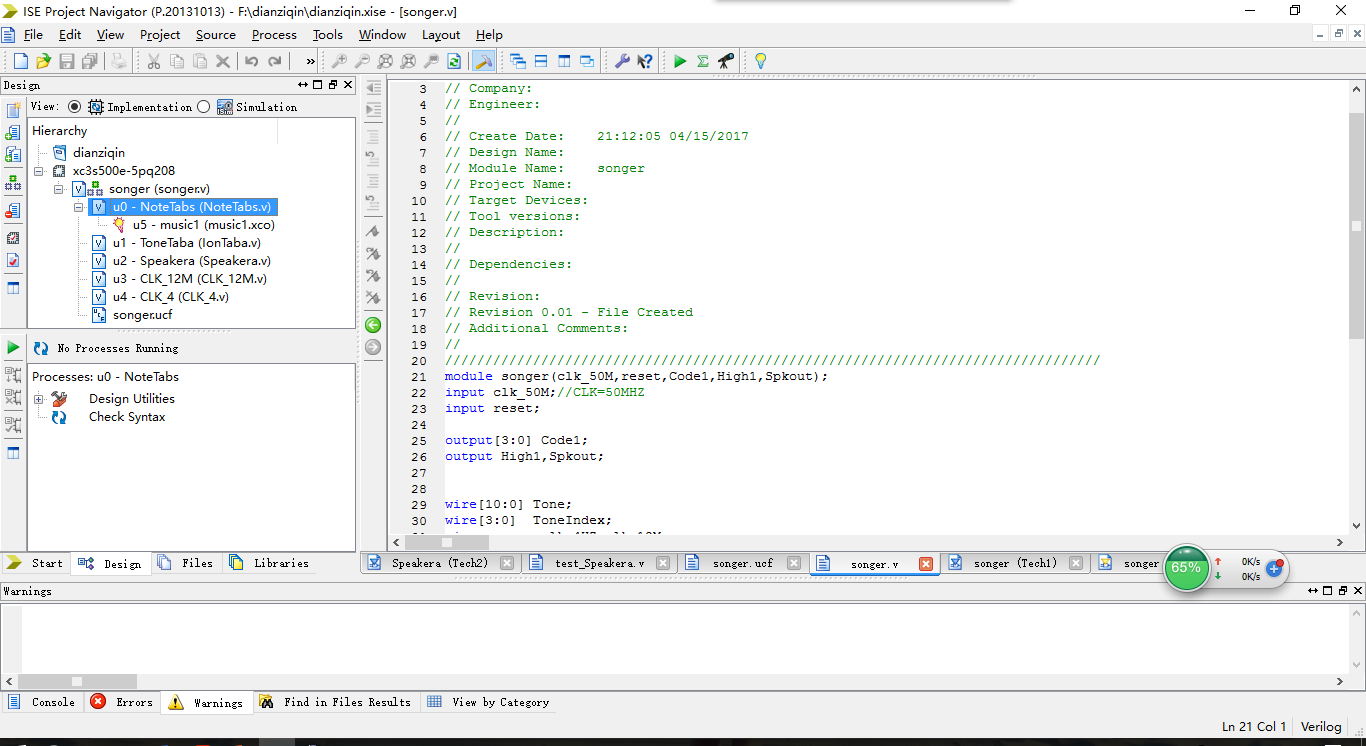


图15 电子琴系统设计综合界面

由于Speakera.v、div\_50\_4HZ.v模块的仿真时间过长，所以顶层模块不需要仿真。



图16 锁引脚操作

锁引脚，添加约束文件后，需要下载到实验板和硬件调试

**四、思考题**

1.电路上应该满足哪些条件，才能用数字器件直接输出的方波驱动扬声器发声？

答：需满足如下条件：①输出频率在扬声器的工作范围内；②驱动电流能够驱动扬声器。

2.如果演奏其他乐曲，程序应做哪些方面的改动？

答：首先.coe文件内改成对应乐曲的谱，同时在ToneTaba和NoteTaba做一些小改动。

**五、实验体会**

本实验是预先将“梁祝”乐谱以.coe文件格式存储，在调用ROM实现乐曲曲谱的存储，结合程序将乐谱的值依次读出来，转化对应的音阶的频率，同时在实验板上通过扬声器发出音乐，LED等显示琴音谱，因此这是一个纯硬件乐曲演奏电路。对我而言，理解其原理难度很大。带着问题，一步步探索。首先，乐谱的音调、音阶和节拍的含义；其次，乐谱的存储和如何读出值；再者，读出的值怎样才能转换为对应的频率；最后，怎样仿真验证功能。

在仿真中，验证数控分频器模块时，我不能准确把握每一个音符转化成对应的半周期的时间间隔，所以就只验证了一个“5”。并且，我遇到很多模块仿真波形图中，输出端出现“xxx”。反复查阅资料，才发现原来，定义的某些中间变量都需要赋初值，否则最后仿真数据结果就会出现“xxx”不定态，对模块的功能仿真验证造成干扰。

整个设计流程走下来，能力得到了一定的提升，收获很多，特别是对接下来的实验调试充满了期待。