

综合实例-音乐播放器设计

主讲教师： 邱德慧



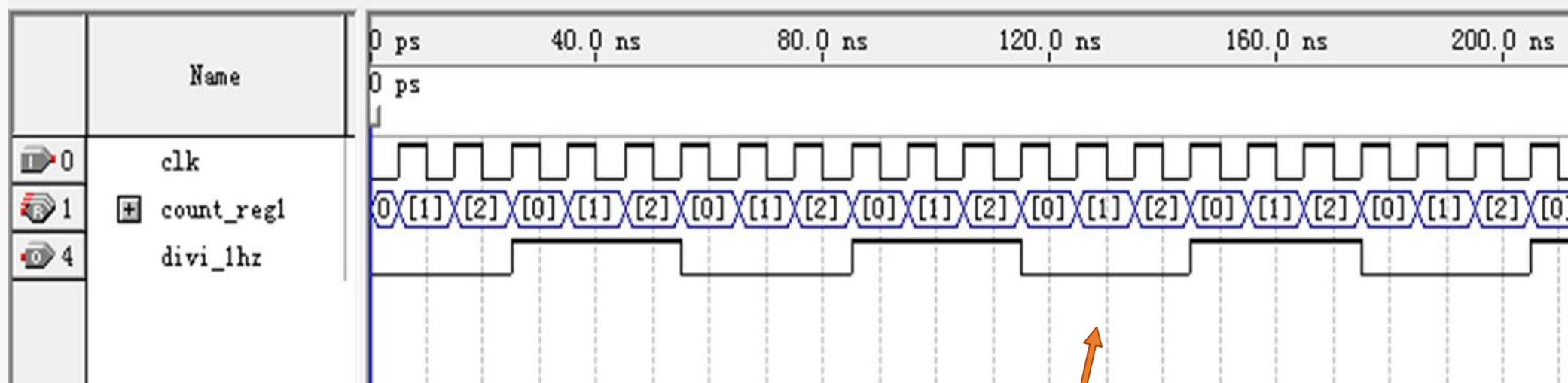
知识回顾

🎯 计数器的应用-分频器设计

```
1 module divi_1hz(clk,divi_1hz);  
2     input clk;  
3     output reg divi_1hz;  
4     reg [1:0] count_reg1;  
5  
6     //*****div the clk*****//  
7     always@(posedge clk)  
8         if(count_reg1==2'd2) // count_reg div clk to generate clk_reg  
9         = begin  
10            divi_1hz<=~divi_1hz;  
11            count_reg1<=2'd0;  
12        end  
13        else  
14        = begin  
15            count_reg1<=count_reg1+2'd1;  
16        end  
17  
18 endmodule
```

知识回顾

计数器的应用-分频器仿真波形



N分频，占空比50%的方波

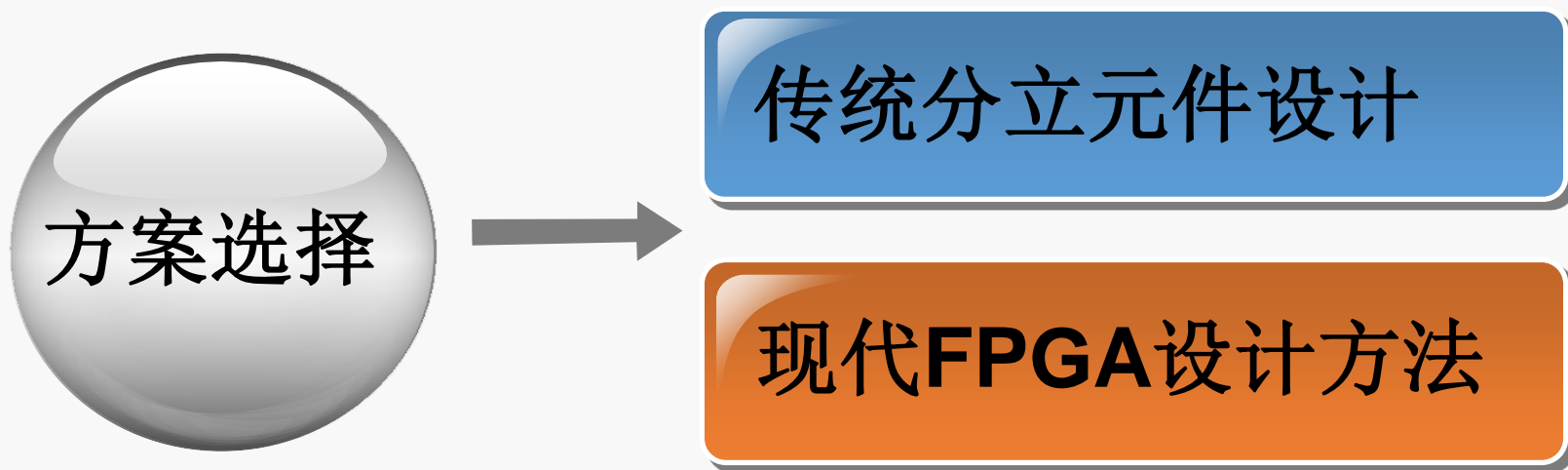
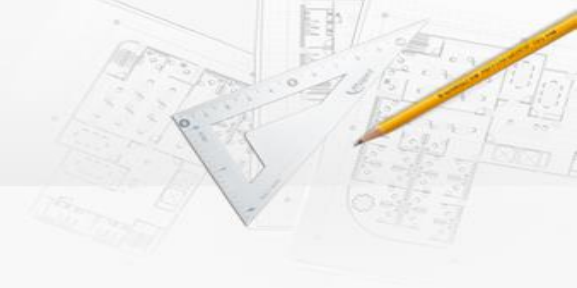
问题引入



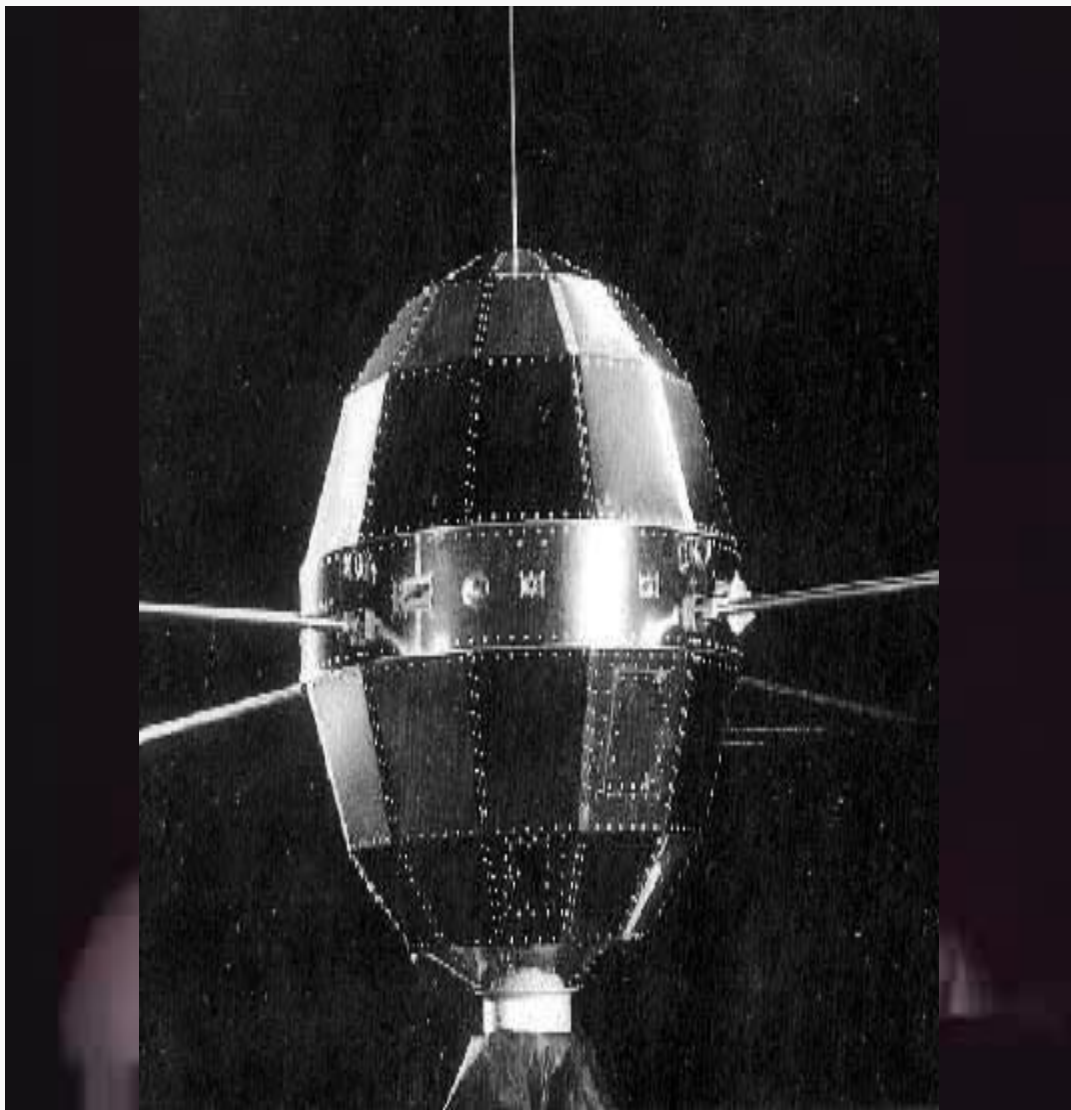
🎵 音乐播放器的设计

- **要求1**：实现多首歌曲自动播放，实现手动选择歌曲、暂停歌曲、快慢播放等功能，并在数码管上显示歌谱。
- **要求2**：实验箱的键1-键7作为电子琴的7个音符，按下键1能让实验箱的扬声器发出“哆”，一直到高音“西”。而且能够实现二个或三个音区；
- **要求3**：利用16*16点阵屏显示歌名；

问题引入-音乐播放器设计

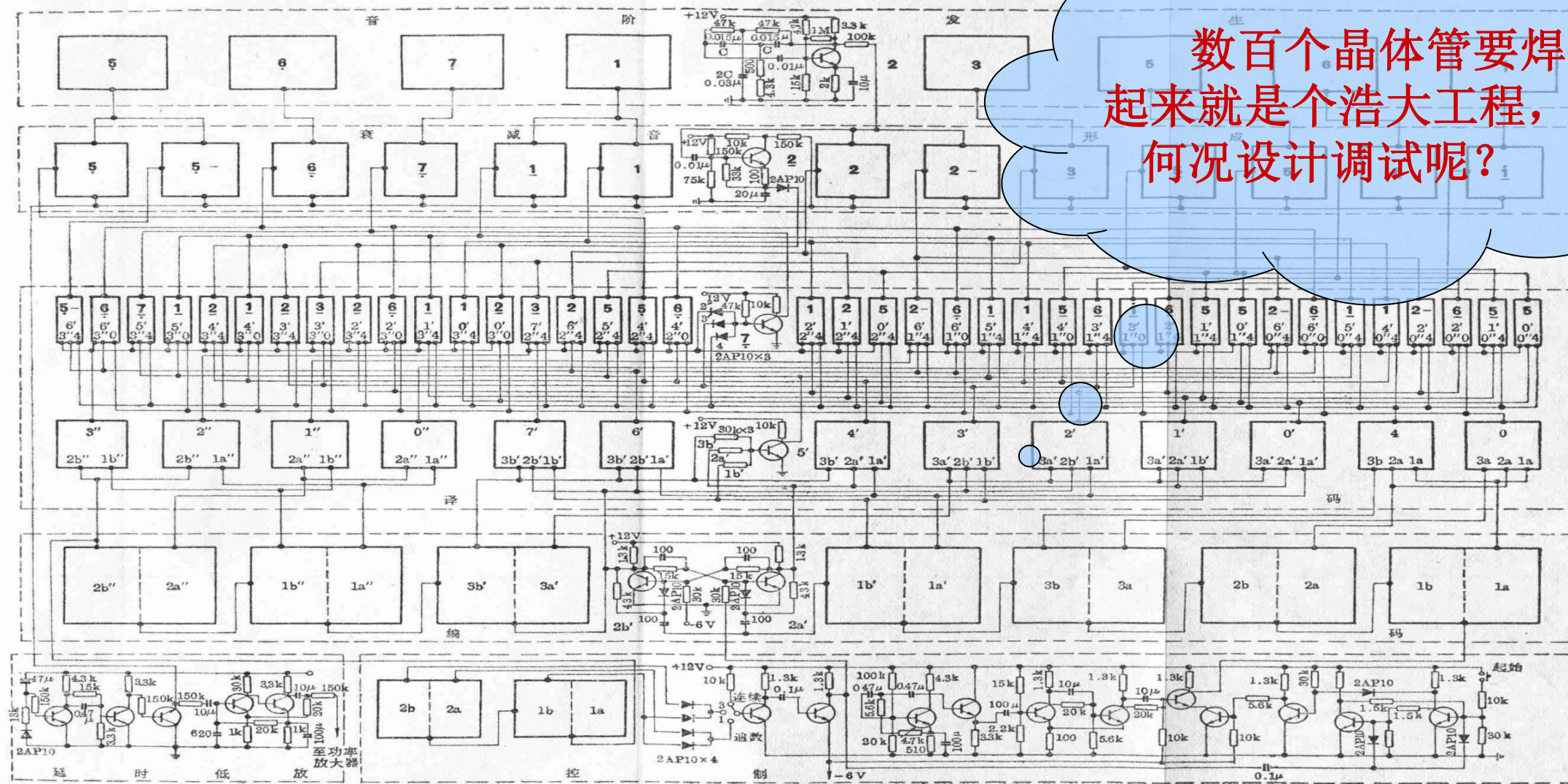


最有影响力的乐曲演奏电路



- ④ 1970年中国发射的第一颗人造卫星因安装模拟演奏《东方红》乐曲的音乐仪器而得名“**东方红一号**”。
- ④ 《东方红》音乐演奏电路全部用**晶体管分立元件**做成。据统计，整个音乐装置共用了**110**多个三极管、约**150**个二极管以及大量的电阻和电容等。

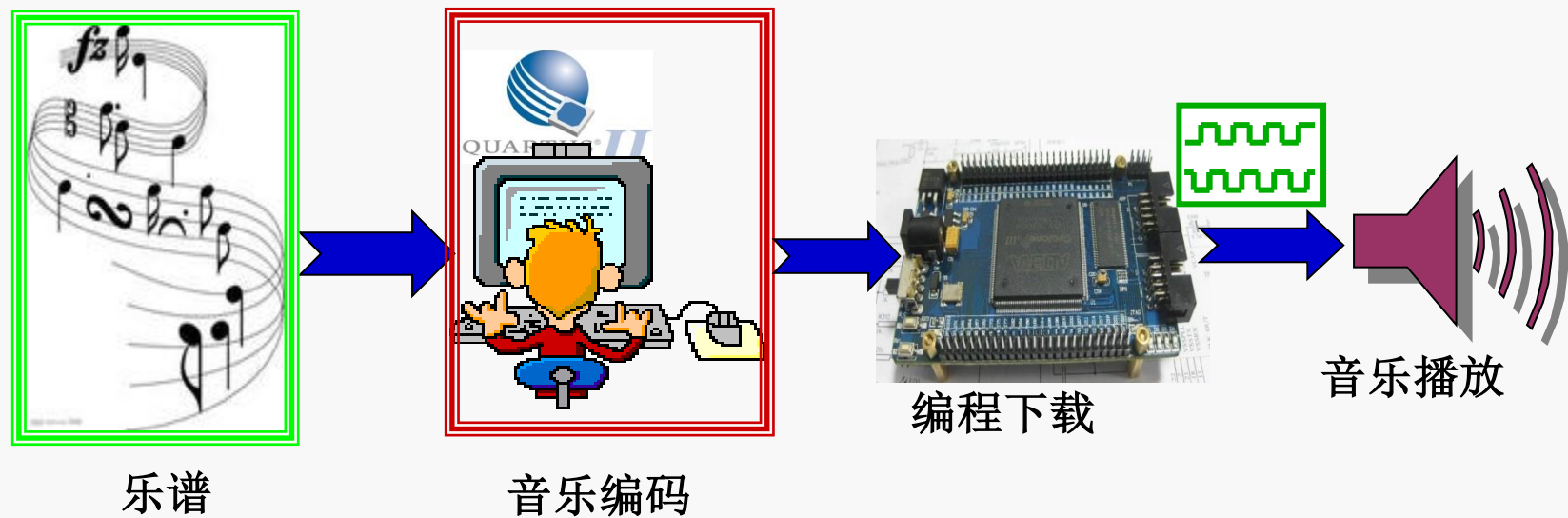
最有影响力的乐曲演奏电路



数百个晶体管要焊接起来就是个浩大工程，何况设计调试呢？

“东方红一号”卫星上的音乐演奏器电路图

现代基于FPGA的设计方法



基于FPGA的设计方法



I 乐曲演奏电路设计原理

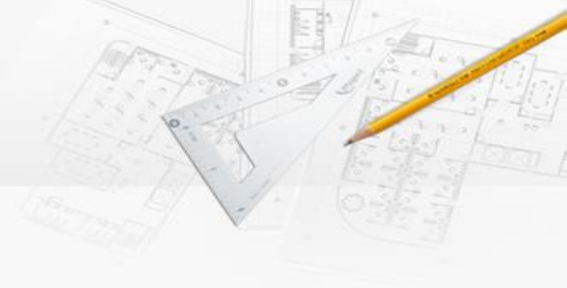
II 基于FPGA的设计思路

III 宏功能模块使用

IV 详细设计步骤



I 乐曲演奏电路工作原理



- ④ 1. 确定乐谱
- ④ 2. 音乐编码原理
- ④ 3. 音乐编码实现

1 确定乐谱

🎵 我和我的祖国

我和我的祖国

1 = D $\frac{6}{8}$ $\frac{9}{8}$

J = 150 庄重 热情地

李谷一 演唱

张黎词
秦咏诚曲

($\dot{1}$ $\dot{2}$ $\dot{3}$ $\dot{2}$ $\dot{1}$ $\dot{6}$ | $\dot{7}$ $\dot{6}$ $\dot{3}$ $\dot{5}$ $\dot{5}$ | $\dot{1}$ $\dot{2}$ $\dot{3}$ $\dot{2}$ $\dot{1}$ $\dot{6}$ | $\dot{7}$ $\dot{5}$ $\dot{3}$ $\dot{6}$ $\dot{6}$ |

$\dot{5}$ $\dot{4}$ $\dot{3}$ $\dot{2}$ | $\dot{7}$ $\dot{6}$ $\dot{5}$ $\dot{3}$ | $\dot{4}$ $\dot{3}$ $\dot{2}$ $\dot{1}$ | $\dot{1}$ $\dot{3}$ $\dot{3}$ $\dot{5}$ $\dot{3}$ $\dot{3}$ | $\dot{1}$ $\dot{3}$ $\dot{3}$ $\dot{5}$ $\dot{3}$ $\dot{3}$) |

$\dot{5}$ $\dot{6}$ $\dot{5}$ $\dot{4}$ $\dot{3}$ $\dot{2}$ | $\dot{1}$ $\dot{5}$ | $\dot{1}$ $\dot{3}$ $\dot{1}$ $\dot{7}$ $\dot{6}$ $\dot{3}$ | $\dot{5}$ $\dot{5}$ | $\dot{6}$ $\dot{7}$ $\dot{6}$ $\dot{5}$ $\dot{4}$ $\dot{3}$ | $\dot{2}$ $\dot{6}$ |

1. 我和我的祖国，一刻也不能分割。无论我走到哪里，
2. 我的祖国和我，像海和浪花一朵。浪是海的赤子，

$\dot{7}$ $\dot{6}$ $\dot{5}$ $\dot{5}$ $\dot{1}$ $\dot{2}$ | $\dot{3}$ $\dot{3}$ | $\dot{5}$ $\dot{6}$ $\dot{5}$ $\dot{4}$ $\dot{3}$ $\dot{2}$ | $\dot{1}$ $\dot{5}$ | $\dot{1}$ $\dot{3}$ $\dot{1}$ $\dot{7}$ $\dot{2}$ $\dot{1}$ | $\dot{6}$ $\dot{6}$ |

都流出一首赞歌。我歌唱每一座高山，我歌唱每一条河。
海是那浪的依托。每当大海在微笑，我就是笑的酒窝。

$\dot{1}$ $\dot{7}$ $\dot{6}$ $\dot{5}$ | $\dot{6}$ $\dot{5}$ $\dot{4}$ $\dot{3}$ | $\dot{7}$ $\dot{6}$ $\dot{5}$ $\dot{2}$ | $\dot{1}$ $\dot{1}$ | $\dot{1}$ $\dot{2}$ $\dot{3}$ $\dot{2}$ $\dot{1}$ $\dot{6}$ | $\dot{7}$ $\dot{6}$ $\dot{3}$ $\dot{5}$ $\dot{5}$ |

袅袅炊烟，小小村落，路上一道辙。啦啦啦 啦啦啦 啦啦啦啦
我分担着海的忧愁，分享海的欢乐。啦啦啦 啦啦啦 啦啦啦啦

$\dot{1}$ $\dot{2}$ $\dot{3}$ $\dot{2}$ $\dot{1}$ $\dot{6}$ | $\dot{7}$ $\dot{5}$ $\dot{3}$ $\dot{6}$ $\dot{6}$ | $\dot{5}$ $\dot{4}$ $\dot{3}$ $\dot{2}$ | $\dot{7}$ $\dot{6}$ $\dot{5}$ $\dot{3}$ | $\dot{4}$ $\dot{2}$ $\dot{1}$ | $\dot{1}$ $\dot{1}$ $\dot{0}$ ||

啦啦啦 啦啦啦 啦啦啦啦 你用你那母亲的温情和我诉说。
啦啦啦 啦啦啦 啦啦啦啦 永远给我碧浪清波，心中的

$\dot{1}$ $\dot{1}$ $\dot{0}$ || $\dot{1}$ $\dot{2}$ $\dot{3}$ $\dot{2}$ $\dot{1}$ $\dot{6}$ | $\dot{7}$ $\dot{6}$ $\dot{3}$ $\dot{5}$ | $\dot{1}$ $\dot{2}$ $\dot{3}$ $\dot{2}$ $\dot{1}$ $\dot{6}$ | $\dot{7}$ $\dot{5}$ $\dot{3}$ $\dot{6}$ |

歌。啦啦啦 啦啦啦 啦啦啦啦 啦啦啦啦 啦啦啦啦

$\dot{5}$ $\dot{4}$ $\dot{3}$ $\dot{2}$ | $\dot{7}$ $\dot{6}$ $\dot{5}$ $\dot{3}$ | $\dot{4}$ $\dot{2}$ $\dot{1}$ | $\dot{1}$ $\dot{1}$ $\dot{0}$ || $\dot{5}$ $\dot{2}$ $\dot{1}$ | $\dot{1}$ $\dot{1}$ ||

永远给我碧浪清波，心中的歌。心中的歌。

2 音乐编码原理



🎵 **问题1**：如何发出不同音调的声音？

高音



$\dot{1}$ $\dot{2}$ $\dot{3}$ $\dot{4}$ $\dot{5}$ $\dot{6}$ $\dot{7}$

中音



1 2 3 4 5 6 7

低音



$\underset{\cdot}{1}$ $\underset{\cdot}{2}$ $\underset{\cdot}{3}$ $\underset{\cdot}{4}$ $\underset{\cdot}{5}$ $\underset{\cdot}{6}$ $\underset{\cdot}{7}$

2 音乐编码原理



🎵 音调控制——音符的频率值

音名	频率 (Hz)	音名	频率 (Hz)	音名	频率 (Hz)
低音1	261.6	中音1	523.3	高音1	1046.5
低音2	293.7	中音2	587.3	高音2	1174.7
低音3	329.6	中音3	659.3	高音3	1318.5
低音4	349.2	中音4	698.5	高音4	1396.9
低音5	392	中音5	784	高音5	1568
低音6	440	中音6	880	高音6	1760
低音7	493.9	中音7	987.8	高音7	1975.5

2 音乐编码原理

🎵 **问题2**：如何控制音符的节拍？

名称	简谱记法	节拍
全音符	5---	4
二分音符	5-	2
四分音符	5	1
八分音符	<u>5</u>	1/2
十六分音符	<u><u>5</u></u>	1/4

备注：带附点的音符：5. = 5 + 5



分频器

永远给我 碧浪清波， 心中的歌 心中的歌

$\overset{\frown}{5\ 6\ 5}\ \overset{\frown}{4\ 3\ 2} \mid 1.\ 5. \mid \overset{\cdot}{1}\ 3\ \overset{\cdot}{1}\ \overset{\cdot}{7}\ 6\ 3 \mid \overset{\frown}{5.\ 5.}$
 1. 我 和 我 的 祖 国, 一 刻 也 不 能 分 割。
 2. 我 的 祖 国 和 我, 像 海 和 浪 花 一 朵。

基于FPGA的设计方法

I 乐曲演奏电路设计原理



II 基于FPGA的设计思路

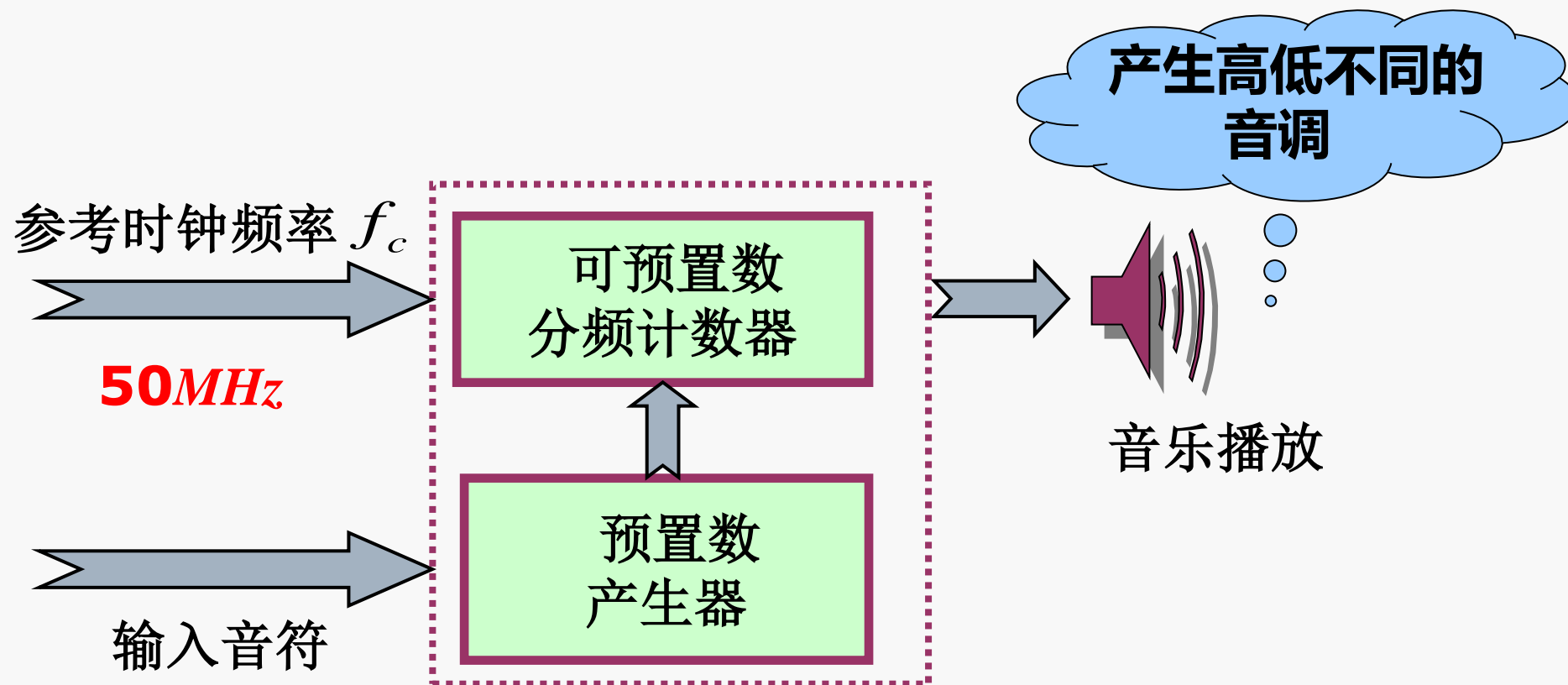
III 宏功能模块使用

IV 详细设计步骤



II 基于FPGA的设计思路

④ 音调发生器



基于FPGA的设计思路



5 6 5 | 4 3 2 | 1. 5. | 1 3 1̇ | 7 6 3 | 5. 5. |

1. 我 和 我 的 祖 国， 一 刻 也 不 能 分 割。
2. 我 的 祖 国 和 我， 像 海 和 浪 花 一 朵。

参数 \ 音名	中音5	中音3	中音2
音符频率 (Hz)	784	659.3	587.3
分频比	63776	75838	85136

分频比计算方法

以中音5为例：

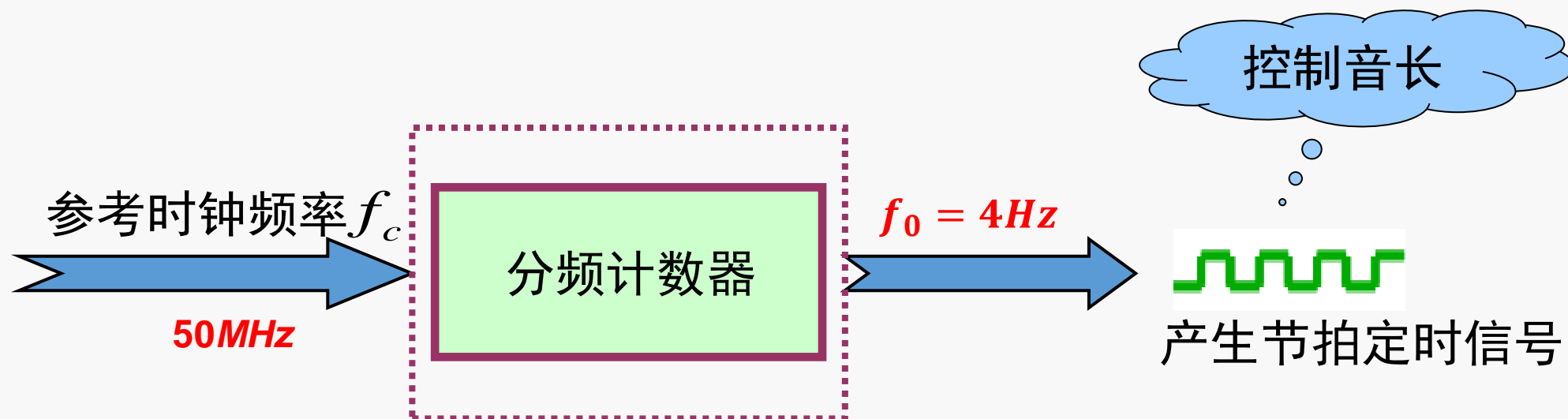
$$50\text{MHz}/784=63776$$

需要几位计数器呢？

20位+

基于FPGA的设计思路

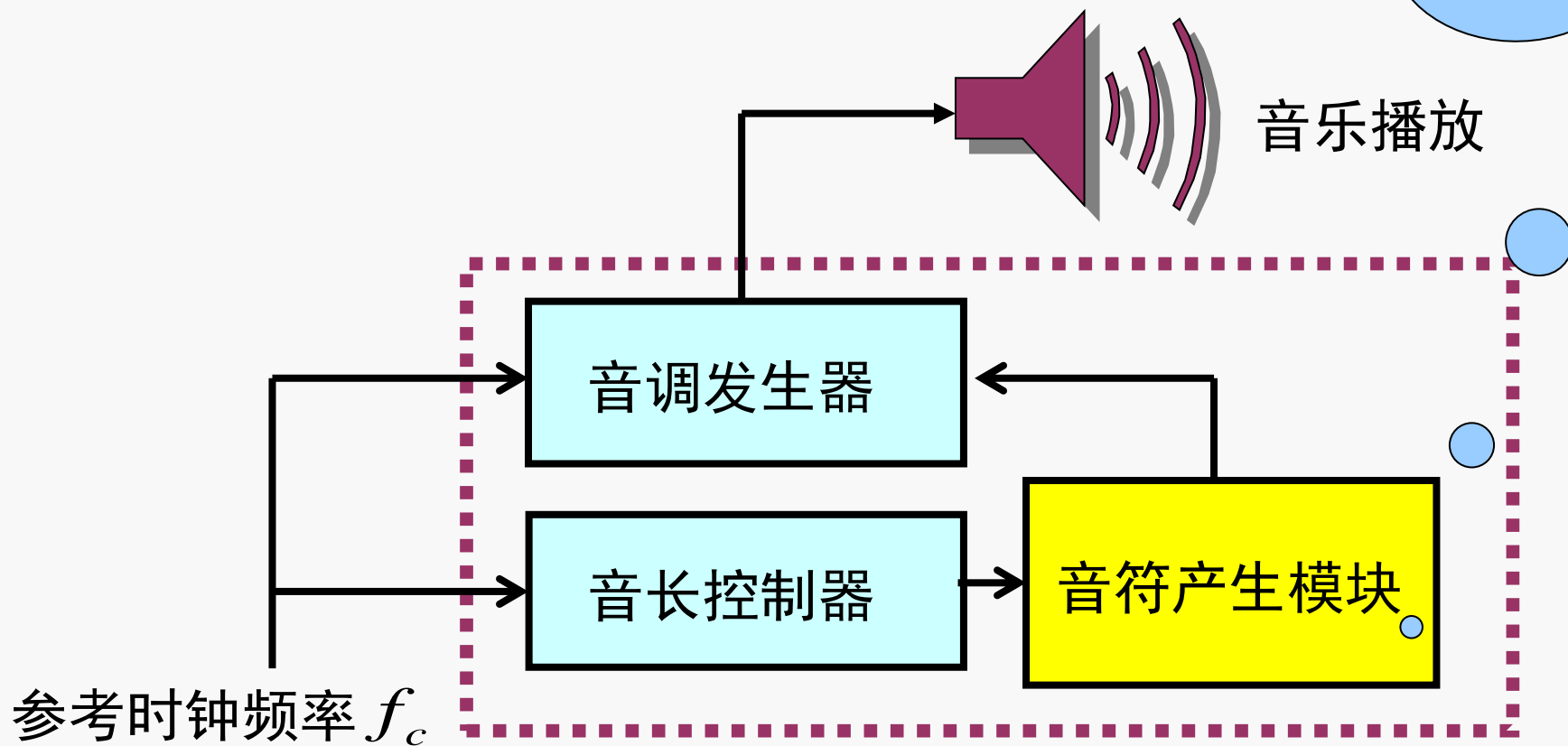
④ 音长控制器



$$\text{分频比} = \frac{f_c}{f_0} = \frac{50MHz}{4Hz} = \mathbf{12.5 \times 10^6}$$

基于FPGA的设计思路

④ 音乐播放器总体框图



音符产生模块按节拍要求产生乐曲演奏需要的音符。
如何实现？

基于FPGA的设计方法

I 乐曲演奏电路设计原理

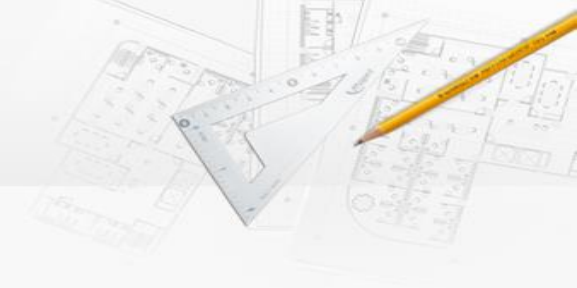
II 基于FPGA的设计思路

III 宏功能模块使用

IV 详细设计步骤



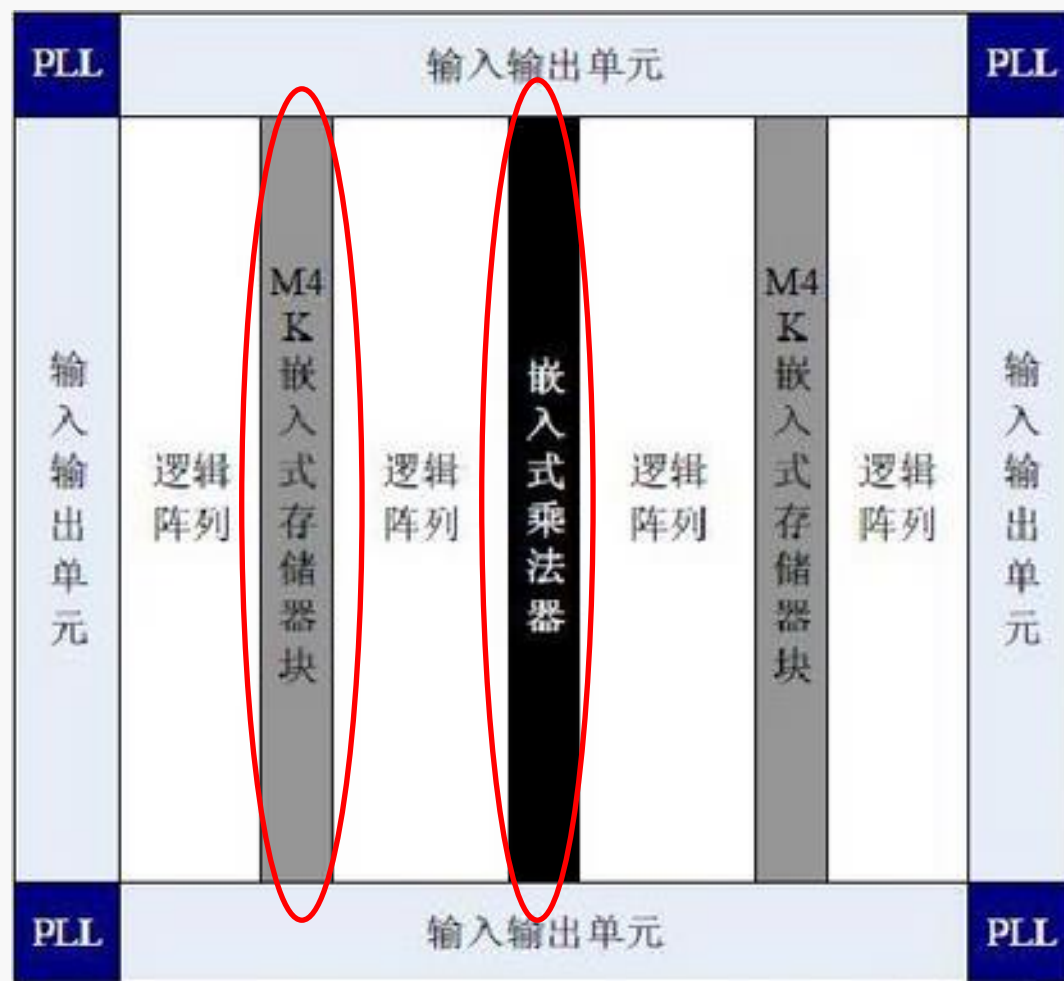
III 宏功能模块应用



提高电子设计的效率和可靠性

使用**Altera**器件的特定硬件功能

III 宏功能模块应用



Cyclone 系列结构

III 宏功能模块应用



宏功能模块介绍

LPM(Library of Parameterized Modules)参数可设置模块库

LPM提供的功能模块可以通过**修改参数**获得不同的逻辑功能，实现了**规模可变性**和**自适应性**，极大地**简化**了设计任务。

93年，LPM被EIA（Electronic Industries Association）确立为**过渡标准**。

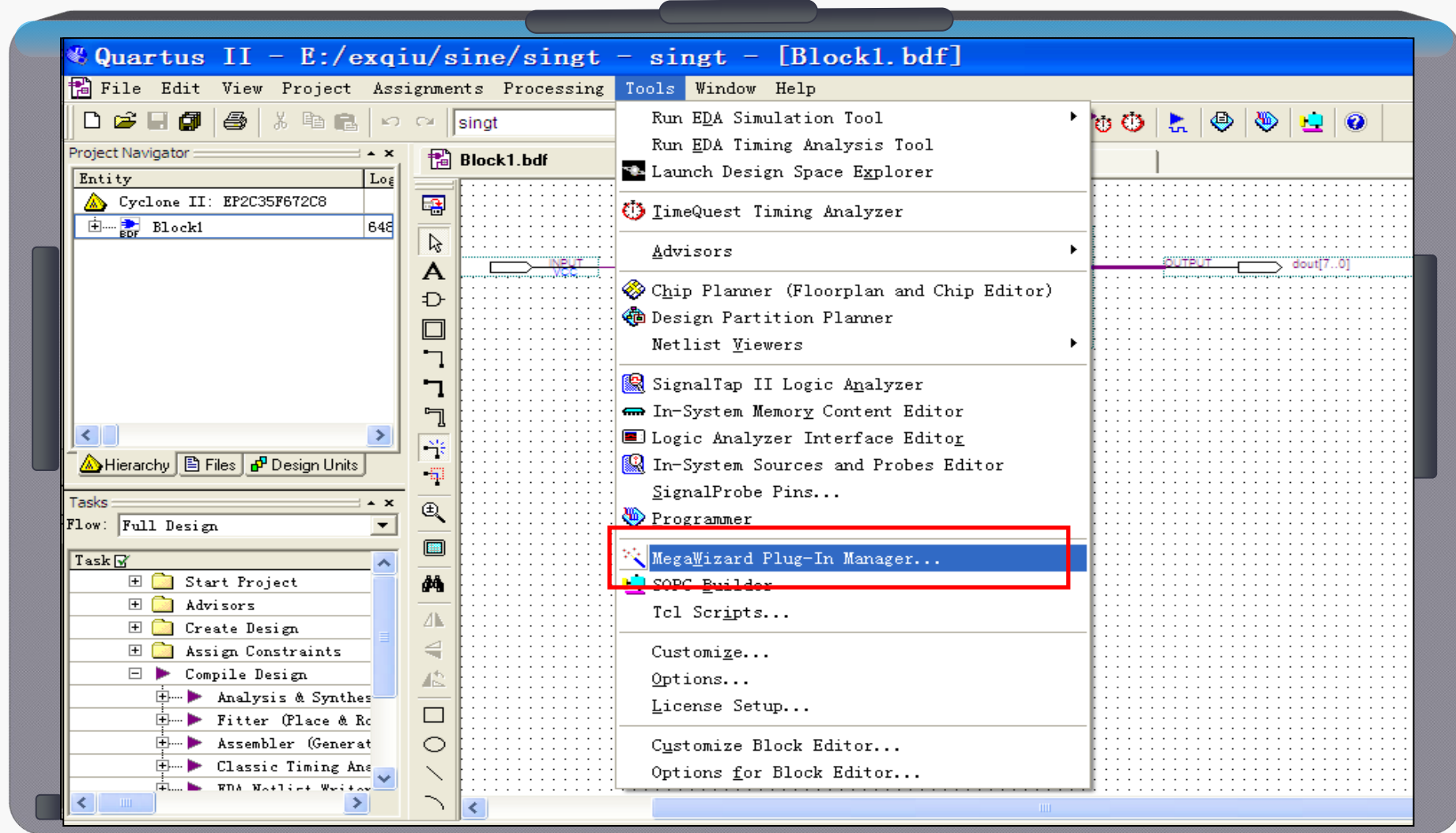
LPM被大多数的EDA开发商所支持，如Cadence, Mentor Graphics, Synopsys, Synplicity 和 Viewlogic等。

III 宏功能模块应用

- 免费Megafunction: 免费的模块也足够满足大多数设计的需要
(LPM库只有25个基本模块就号称可以完成所有的设计)。
- 评估付费Megafunction: 使用开放式内核(Open IPCore)技术

1. 宏功能模块概述

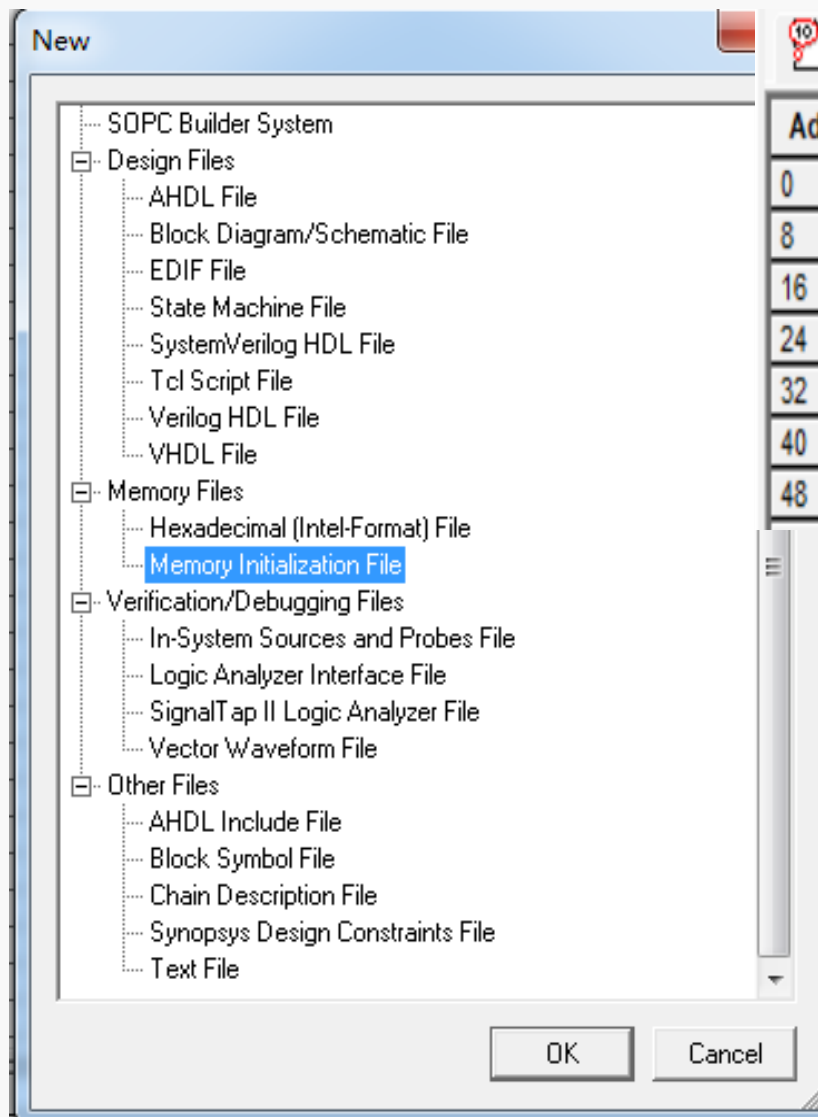
使用MegaWizard Plug-In Manager



2. LPM_ROM的应用

Step 1: 定制初始化数据文件

- 建立.mif文件或.hex文件
- 方法：填表或编程
- Width=? ; Depth=?



zuguo.mif

Addr	+0	+1	+2	+3	+4	+5	+6	+7
0	12	12	13	13	12	12	11	11
8	10	10	9	9	8	8	8	8
16	8	8	5	5	5	5	5	5
24	8	8	10	10	15	15	14	14
32	13	13	13	10	12	12	12	12
40	12	12	12	12	12	12	12	12
48	13	13	14	14	13	13	12	12

2. LPM_ROM的应用

Step 1: 定制LPM_ROM元件

MegaWizard Plug-In Manager - ROM: 1-PORT [page 7 of 7] -- Summary

ROM: 1-PORT

1 Parameter Settings 2 EDA 3 Summary

Block type: AUTO

Turn on the files you wish to generate. A gray checkmark indicates a file that is automatically generated, and a red checkmark indicates an optional file. Click Finish to generate the selected files. The state of each checkbox is maintained in subsequent MegaWizard Plug-In Manager sessions.

The MegaWizard Plug-In Manager creates the selected files in the following directory:
D:\BICE\EDA\5.2.2_Play_of_Organ\

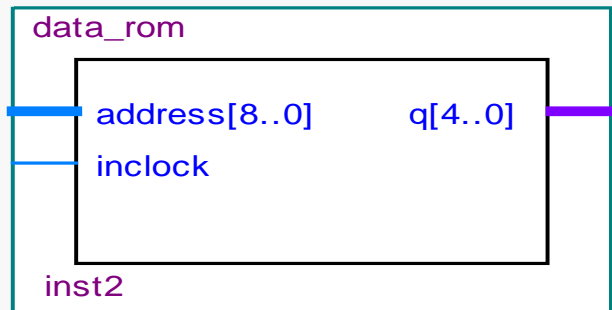
File	Description
<input checked="" type="checkbox"/> chuanq1.v	Variation file
<input type="checkbox"/> chuanq1.inc	AHDL Include file
<input type="checkbox"/> chuanq1.cmp	VHDL component declaration file
<input type="checkbox"/> chuanq1.bsf	Quartus II symbol file
<input type="checkbox"/> chuanq1_inst.v	Instantiation template file
<input checked="" type="checkbox"/> chuanq1_bb.v	Verilog HDL black-box file
<input checked="" type="checkbox"/> chuanq1_waveforms.html	Sample waveforms in summary
!... chuanq1_wave*.jpg	Sample waveform file(s)

Resource Usage
1 M9K + 1 sld_mod_ram_rom

Cancel < Back Next > Finish

2. LPM_ROM的应用

- 自动生成data_rom.v

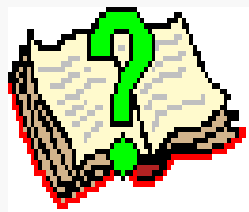


```
36 // synopsys translate_off
37 `timescale 1 ps / 1 ps
38 // synopsys translate_on
39 module data_rom (
40     address,
41     inclock,
42     q);
43
44     input    [8:0] address;
45     input    inclock;
46     output   [4:0] q;
47
48     wire [4:0] sub_wire0;
49     wire [4:0] q = sub_wire0[4:0];
```

```
75
76 defparam
77     altsyncram_component.address_aclr_a = "NONE",
78     altsyncram_component.clock_enable_input_a = "BYPASS",
79     altsyncram_component.clock_enable_output_a = "BYPASS",
80     altsyncram_component.init_file = "zuguo.mif",
81     altsyncram_component.intended_device_family = "Cyclone III",
82     altsyncram_component.lpm_hint = "ENABLE_RUNTIME_MOD=YES,INSTANCE_NAME=ROM1",
83     altsyncram_component.lpm_type = "altsyncram",
84     altsyncram_component.numwords_a = 512,
85     altsyncram_component.operation_mode = "ROM",
86     altsyncram_component.outdata_aclr_a = "NONE",
87     altsyncram_component.outdata_reg_a = "UNREGISTERED",
88     altsyncram_component.width_a = 9,
89     altsyncram_component.width_a = 5,
90     altsyncram_component.width_byteena_a = 1;
```

2. LPM_ROM的应用

🕒 课堂讨论



本设计中如果需要演奏多首曲目，如何实现？

基于FPGA的设计方法

I 乐曲演奏电路设计原理

II 基于FPGA的设计思路

III 宏功能模块使用

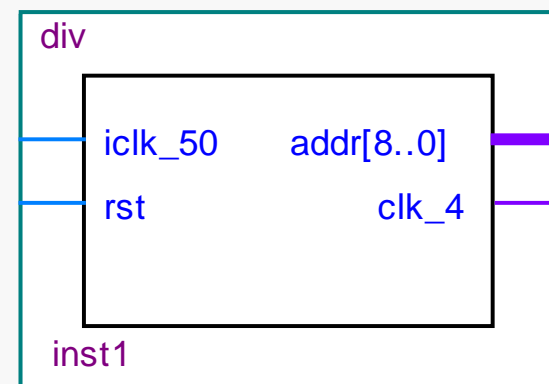
IV Verilog设计步骤



IV. Verilog 设计步骤

🕒 Step 2: 读LPM_ROM存储模块—div.v

```
6 module div(iclk_50,rst,addr,clk_4);
7     input iclk_50;
8     input rst;
9     output clk_4; // clk of 4Hz
10    output [8:0]addr; //ROM address
11
12    reg clk_4;
13    reg [8:0] addr;
14    reg [31:0] count_4;
15
16    //--generate 4Hz clock-----
17    always@(posedge iclk_50 or negedge rst)
18    begin
19        if(!rst)
20        begin
21            clk_4<=1'b0;
22            count_4<=32'd0;
23        end
24        else if (count_4==50000000/(4*2)-1)
25        begin
26            clk_4<=~clk_4;
27            count_4<=32'd0;
28        end
29        else count_4<=count_4+32'd1;
30    end
```

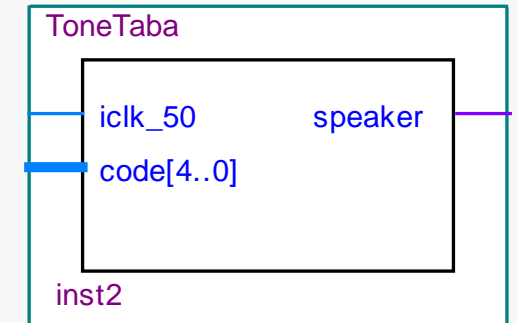


IV. Verilog设计步骤

Step 3: 音频产生模块-ToneTab.v

- 设计关键：产生占空比50%方波的分频器

```
7  module ToneTab(iclk_50,code,speaker);
8      input iclk_50;
9      input [4:0] code; //store music datas
10     output speaker; //output clock frequency
11
12
13     reg [17:0] Tone; //clk division index
14     reg [17:0] cnt;
15     reg clk_tmp;
16
17     always@(posedge iclk_50)
18     begin
19         case (code)
20             5'd0:    Tone<=134;
21             5'd1:    Tone<=191132;
22             5'd2:    Tone<=170242;
23             5'd3:    Tone<=151700;
```

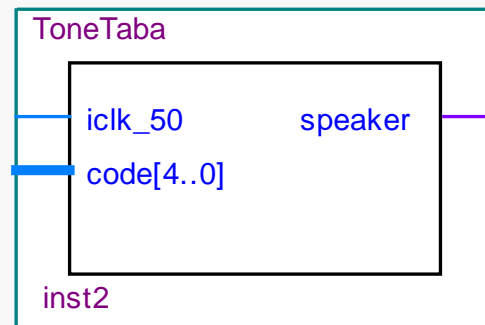


IV. VHDL 设计步骤

Step 3: 音频产生模块-ToneTab.v

- 设计关键：产生占空比50%方波的分频器

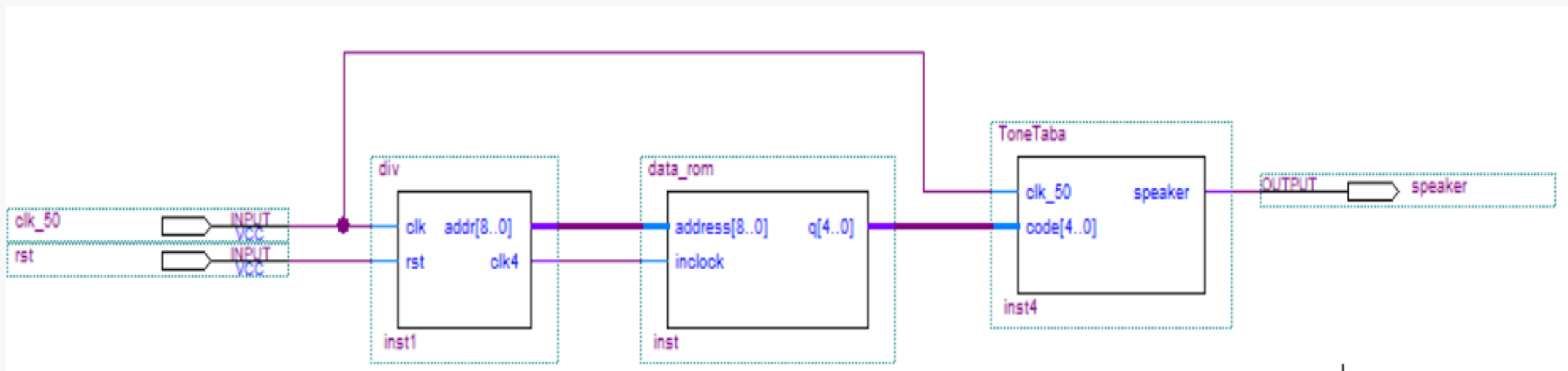
```
49      //always@(posedge iclk_50)
50      if(cnt==Tone/2-1)
51      begin
52          clk_tmp<=~clk_tmp;
53          cnt<=0;
54      end
55      else
56          cnt<=cnt+1;
57      end
58
59      assign speaker=clk_tmp;|
60
61      endmodule
```



IV. Verilog设计步骤

Step 4: 顶层电路设计-songer.v

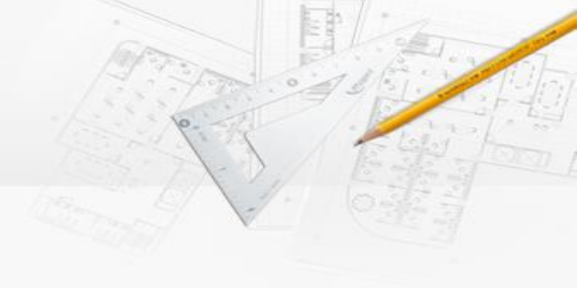
- 设计关键：什么方法设计顶层电路？ 元件例化语句或原理图法



思考：

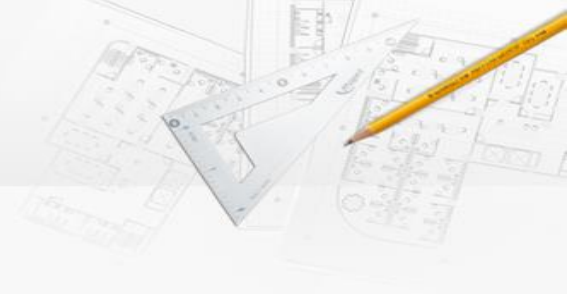
- 原理图作为顶层设计文件有什么优点和缺点？
- 如何使用“元件例化”语句完成模块调用和信号线连接，顶层程序怎样设计？

IV. Verilog 设计步骤--总结

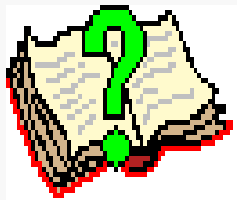


- ⌚ Step 1: 存歌曲LPM_ROM模块—data_rom.v
- ⌚ Step 2: 读歌曲LPM_ROM模块--div.v
- ⌚ Step 3: 音频产生模块--ToneTab.v
- ⌚ Step 4: 顶层电路设计—songer.v

IV. Verilog 设计步骤--总结



🎵 课堂讨论



如果要设计一个电子琴，具有**8**个按键，当按下某一个按键的时候，能够演奏**8**个音符之一，如何实现？

V. 16*16点阵屏的设计

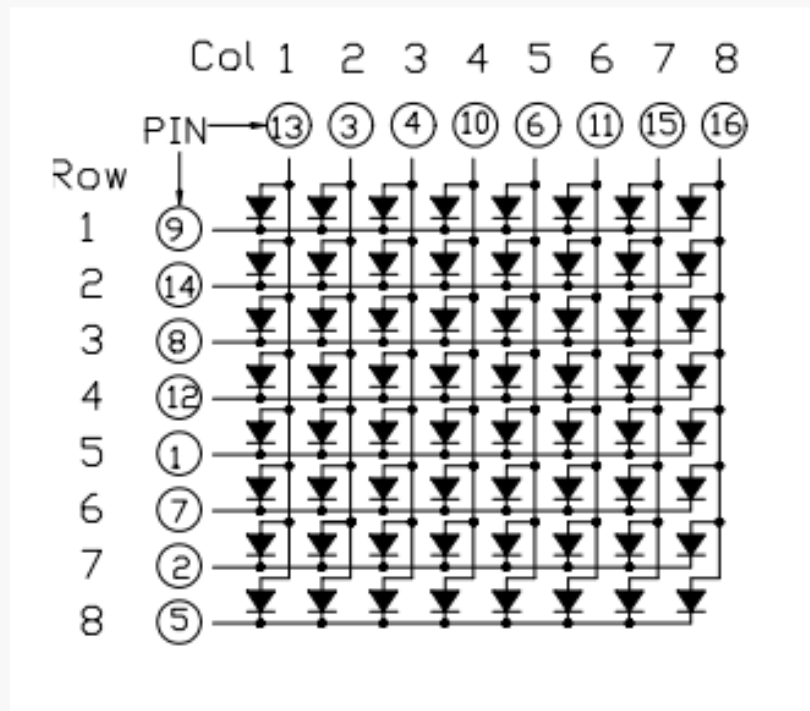
点阵屏的用处：

- 汽车报站器、广告屏以及公告牌等



V. 16*16点阵屏的设计

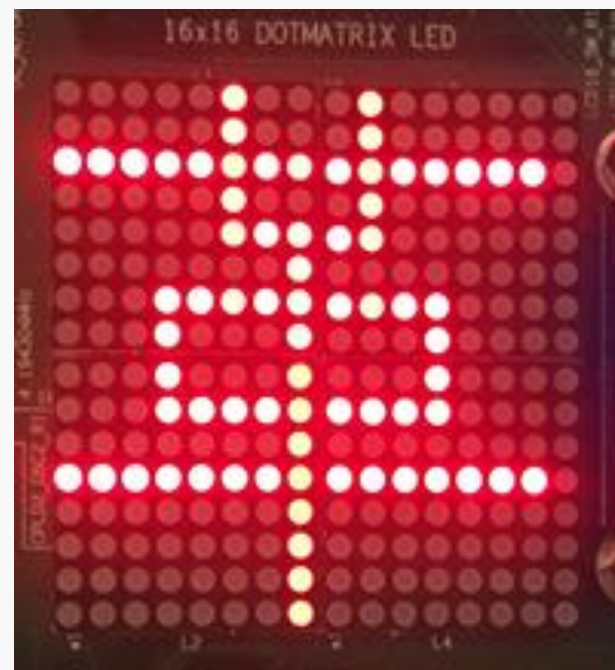
🎯 实验平台上16*16点阵屏：



1. 16*16点阵屏的工作原理

🕒 点阵显示屏的工作原理：

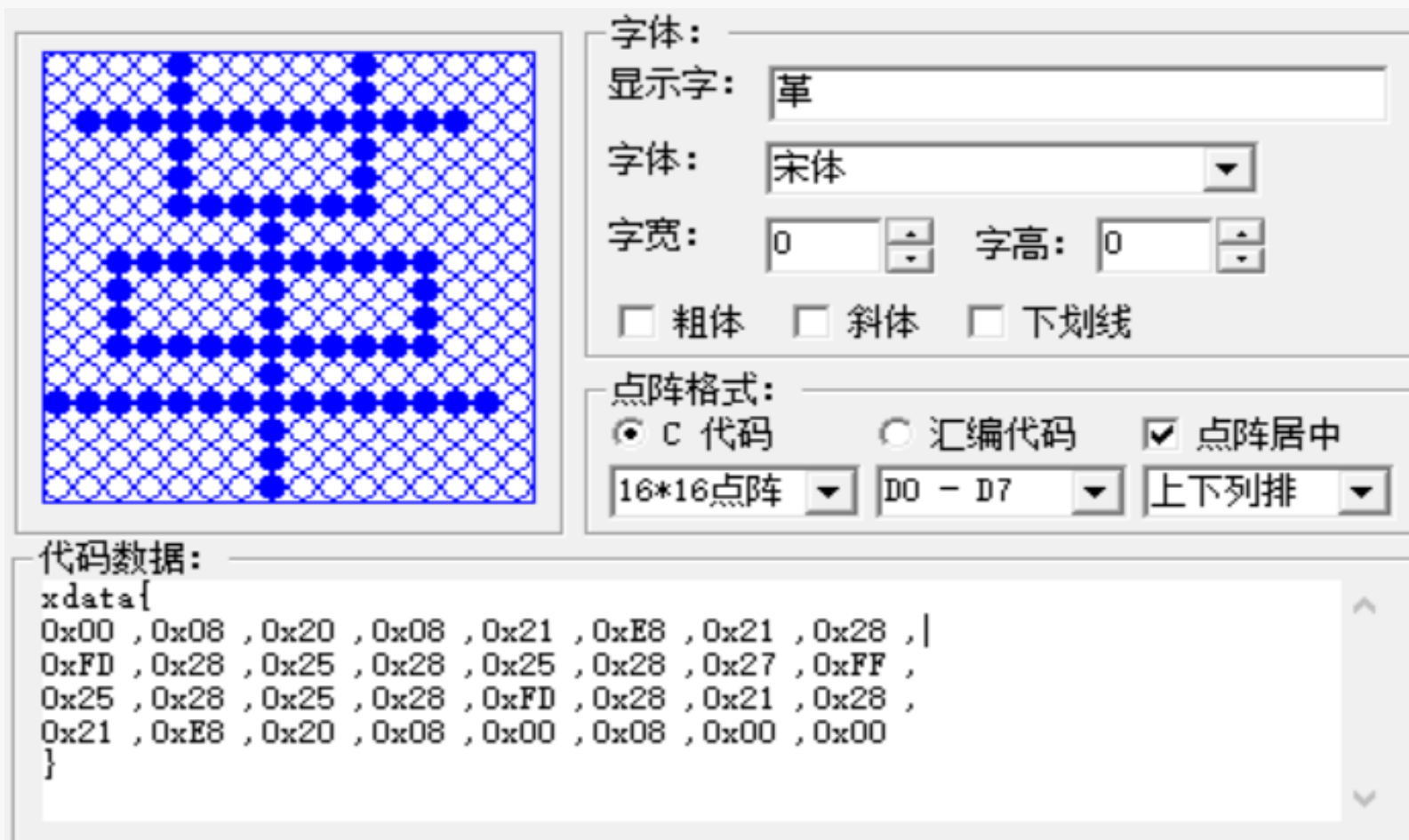
- 16X16点阵扫描控制电路的光点是从显示模块左上角像素点为起始点扫描，终止于右下角像素点。
- 本实验系统所有的点阵是行共阴、列共阳。
- 当列共阳为低电平，行共阴为高电平，则所接的发光像素点点亮，反之，列共阳为高电平，行共阴为低电平，则所接的发光像素处于截止状态不发光。
- 扫描频率为1K-1MHz.



2. 汉字字符点阵提取

🕒 汉字字符点阵提取软件：

- 给出每一列的汉字点阵编码：0000 0000 0000 1000 0008H



字体：显示字：革

字体：宋体

字宽：0 字高：0

☐ 粗体 ☐ 斜体 ☐ 下划线

点阵格式：
☒ C 代码 ☐ 汇编代码 ☒ 点阵居中

16*16点阵 DO - D7 上下列排

代码数据：

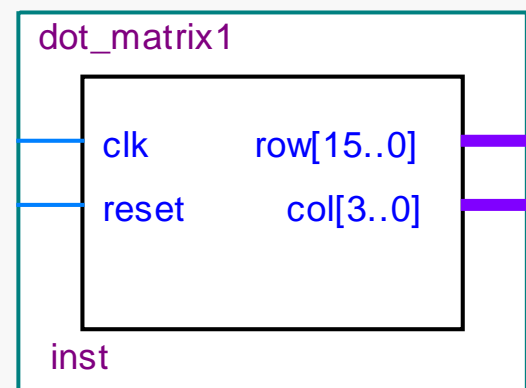
```
xdata{
0x00 , 0x08 , 0x20 , 0x08 , 0x21 , 0xE8 , 0x21 , 0x28 , |
0xFD , 0x28 , 0x25 , 0x28 , 0x25 , 0x28 , 0x27 , 0xFF ,
0x25 , 0x28 , 0x25 , 0x28 , 0xFD , 0x28 , 0x21 , 0x28 ,
0x21 , 0xE8 , 0x20 , 0x08 , 0x00 , 0x08 , 0x00 , 0x00
}
```

3. 点阵屏的Verilog设计

🕒 Verilog源程序：

```
6  module dot_matrix1
7      (input          clk,
8              reset,
9              output reg [15:0] row,    //行
10             output reg [3:0] col      //列
11         );
12     reg clk1;
13     integer i;
```

```
15     /*****字符显示*****/
16     initial col=4'b0;
17     always @(posedge clk1)
18     begin
19         if (!reset)    col<=4'b0;
20         else
21         begin
22             if (col==4'b1111)
23                 col<=4'b0;
24             else    col<=col+1;
25         end
26     end
```



4. 点阵屏的Verilog设计

🔍 Verilog源程序:

```
28  always @(reset or col)           //革新科技
29  begin
30      if (!reset) row<=16'b0;
31      else
32          begin
33              case (col)
34                  4'b0000: row<=16'b01000100000010100; //1
35                  4'b0001: row<=16'b0101110001011000; //2
36                  4'b0010: row<=16'b1111110011111111; //3
37                  4'b0011: row<=16'b0111111100011000; //4
38                  4'b0100: row<=16'b1111110001110100; //5
39                  4'b0101: row<=16'b0101110011111111; //6
40                  4'b0110: row<=16'b01000100000010000; //7
41                  4'b0111: row<=16'b00000000000000000; //8
42                  4'b1000: row<=16'b00000000000000000; //9
43                  4'b1001: row<=16'b0001000100100100; //10
44                  4'b1010: row<=16'b0111011011111111; //11
45                  4'b1011: row<=16'b1101111100101001; //12
46                  4'b1100: row<=16'b0111011001011010; //13
47                  4'b1101: row<=16'b00111111111110100; //14
48                  4'b1110: row<=16'b0101111101011010; //15
49                  4'b1111: row<=16'b10010000001010001; //16
50              default: row<=16'b00000000000000000;
51          endcase
52      end
53  end
```

总结



- ④ 可变分频器的设计
- ④ 宏功能模块的使用
- ④ 点阵屏的工作原理