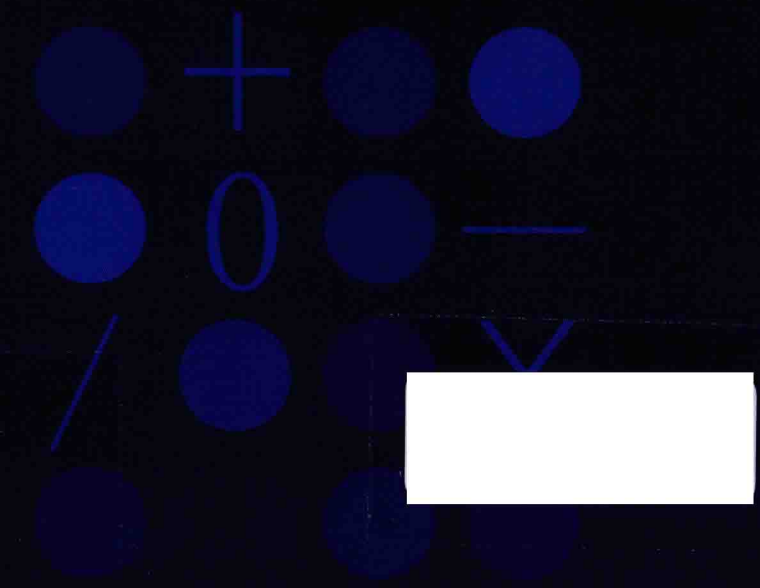
Verilog FPGA数字系统设计自学丛书

四则运算小计算器设计过程实录

——Verilog FPGA数字系统设计入门学习日记

赵　然　编著
夏宇闻　审定



北京航空航天大学出版社
BEIHANG UNIVERSITY PRESS

Verilog FPGA 数字系统设计自学丛书

四则运算小计算器设计过程实录

——Verilog FPGA 数字系统设计入门学习日记

赵 然 编著
夏宇闻 审定

下载程序请用
QQ浏览器扫码

北京航空航天大学出版社

内 容 简 介

本书以日记的形式记录了一个可实现四则运算计算器的设计过程，从而达到学习 FPGA 设计的目的。全书共 10 章，讲述了从设计开始到完成的全过程，其中包括数码管显示、键盘扫描、状态机等基础模块的设计，以及设计中需要注意的问题等，每一章的最后还有夏宇闻老师对本章内容的点评及给读者的学习建议。

希望读者按顺序阅读本书，同时进行实践操作，并与书中的进度保持一致，最终完成整个设计。读者也可以根据自己的想法来实现想要的功能，做到举一反三，以达到最好的学习效果。书中使用的硬件为至芯科技的四代开发板、Altera Cyclone IV 的芯片，软件为 Quartus II 13.0 sp1。

本书可作为电子工程类、自动控制类、计算机类的大学本科高年级学生及研究生设计实验参考用书，亦可供其他工程人员自学与参考。

图书在版编目(CIP)数据

四则运算小计算器设计过程实录：Verilog FPGA 数字系统设计入门学习日记 / 赵然编著. -- 北京 : 北京航空航天大学出版社，2015.9

ISBN 978-7-5124-1958-2

Ⅰ. ①四… Ⅱ. ①赵… Ⅲ. ①硬件描述语言-程序设计②现场可编程门阵列-系统设计 Ⅳ. ①TP312 ②TP332.1

中国版本图书馆 CIP 数据核字(2015)第 281303 号

四则运算小计算器设计过程实录——Verilog FPGA 数字系统设计入门学习日记

赵 然 编著

夏宇闻 审定

责任编辑 陈守平

*

北京航空航天大学出版社出版发行

北京市海淀区学院路 37 号(邮编 100191) http://www.buaapress.com.cn

发行部电话：(010)82317024 传真：(010)82328026

读者信箱：emsbook@gmail.com 邮购电话：(010)82316936

北京泽宇印刷有限公司印装 各地书店经销

*

开本：710×1000 1/16 印张：11.5 字数：245 千字

2016 年 1 月第 1 版 2016 年 1 月第 1 次印刷 印数：3000 册

ISBN 978-7-5124-1958-2 定价：29.00 元

目 录

第 1 章 第一天——数码管显示模块的设计 …… 1

1.1 设计需求讲解 …… 1
1.2 七段式数码管显示原理讲解 …… 2
1.3 设计工具使用讲解 …… 4
1.3.1 Quartus II 工具的配置 …… 4
1.3.2 数码管显示模块的可综合代码 …… 6
1.3.3 显示模块的测试 …… 8
1.3.4 转到 ModelSim 仿真工具进行测试 …… 11
1.3.5 下载程序到开发板进行调试 …… 13
1.4 今天工作总结 …… 19
1.5 夏老师评述 …… 19

第 2 章 第二天——键盘扫描模块的设计 …… 21

2.1 设计需求讲解 …… 21
2.2 七段式数码管显示原理讲解 …… 22
2.3 设计工具使用讲解 …… 23
2.3.1 矩阵键盘码扫描分析模块的可综合代码 …… 24
2.3.2 矩阵键盘码扫描分析模块代码解析 …… 29
2.3.3 矩阵键盘扫描分析模块的测试 …… 35
2.3.4 转到 ModelSim 仿真工具进行测试 …… 42
2.3.5 下载程序到开发板进行调试 …… 42
2.4 今天工作总结 …… 46
2.5 夏老师评述 …… 47

第 3 章 第三天——输入状态机模块的设计 …… 48

3.1 设计需求讲解 …… 48
3.2 我对状态机概念的理解 …… 48
3.3 设计工具使用讲解 …… 49

3.3.1 范例代码解析 …… 49
3.3.2 重写状态机代码 …… 61
3.3.3 转到 ModelSim 仿真工具进行测试 …… 67
3.3.4 下载程序到开发板进行调试 …… 67
3.4 今天工作总结 …… 71
3.5 夏老师评述 …… 72

第 4 章 第四天——BCD 码与二进制码转换模块的设计 …… 73

4.1 设计需求讲解 …… 73
4.2 BCD 码转二进制码 …… 74
4.2.1 BCD 码转二进制码的可综合代码 …… 74
4.2.2 BCD 码转二进制码模块的测试代码 …… 75
4.2.3 转到 ModelSim 仿真工具进行测试 …… 76
4.2.4 二进制码转 BCD 码的可综合代码 …… 76
4.2.5 二进制码转 BCD 码模块的测试 …… 80
4.2.6 转到 ModelSim 仿真工具进行测试 …… 82
4.3 今天工作总结 …… 83
4.4 夏老师评述 …… 83

第 5 章 第五天——计算模块的设计 …… 84

5.1 设计需求讲解 …… 84
5.2 设计工具使用讲解 …… 84
5.2.1 计算模块的可综合代码 …… 85
5.2.2 计算模块的测试 …… 85
5.2.3 转到 ModelSim 仿真工具进行仿真 …… 87
5.2.4 模块连接关系 …… 88
5.2.5 下载程序到开发板进行调试 …… 95
5.3 今天工作总结 …… 95
5.4 夏老师评述 …… 95

第 6 章 第六天——可进行连续运算的状态机改进 …… 97

6.1 设计需求讲解 …… 97
6.2 状态机设计讲解 …… 97
6.2.1 状态机的编码形式 …… 97
6.2.2 状态机的分类 …… 98
6.2.3 状态转移图(STD) …… 102

6.3 设计工具使用讲解 …… 103
6.3.1 状态机模块的可综合代码 …… 103
6.3.2 状态机模块的测试 …… 107
6.3.3 转到 ModelSim 仿真工具进行仿真 …… 111
6.3.4 下载程序到开发板进行调试 …… 111
6.4 今天工作总结 …… 113
6.5 夏老师评述 …… 114

第 7 章 第七天——面积优化 …… 115

7.1 设计需求讲解 …… 115
7.2 面积与速度 …… 116
7.3 模块改进 …… 118
7.3.1 计算模块的可综合代码 …… 118
7.3.2 转到 ModelSim 仿真工具进行测试 …… 121
7.3.3 下载程序到开发板进行调试 …… 123
7.4 今天工作总结 …… 125
7.5 夏老师评述 …… 126

第 8 章 第八天——二进制码转 BCD 码模块的优化 …… 127

8.1 设计需求讲解 …… 127
8.2 算法实现 …… 128
8.3 模块改进 …… 129
8.3.1 二进制码转 BCD 码模块的可综合代码 …… 129
8.3.2 转到 ModelSim 仿真工具进行测试 …… 132
8.3.3 下载程序到开发板进行调试 …… 134
8.4 今天工作总结 …… 138
8.5 夏老师评述 …… 140

第 9 章 第九天——去“0”模块的设计 …… 141

9.1 设计需求讲解 …… 141
9.2 算法实现 …… 141
9.3 模块改进 …… 142
9.3.1 去“0”模块的可综合代码 …… 142
9.3.2 转到 ModelSim 仿真工具进行测试 …… 144
9.3.3 下载程序到开发板进行调试 …… 146
9.4 今天工作总结 …… 150

9.5 夏老师评述 …… 151

第 10 章 第十天——负数计算 …… 152

10.1 设计需求讲解 …… 152
10.2 二进制数表示法 …… 152
10.3 补码原码转换模块 …… 153
10.3.1 补码转原码模块的可综合代码 …… 154
10.3.2 转到 ModelSim 仿真工具进行测试 …… 154
10.3.3 原码转补码模块的可综合代码 …… 155
10.3.4 转到 ModelSim 仿真工具进行测试 …… 155
10.4 其他模块的修改 …… 156
10.4.1 显示模块的修改 …… 156
10.4.2 消“0”模块的修改 …… 159
10.4.3 BCD 码和二进制码转换模块的修改 …… 160
10.4.4 计算模块的修改 …… 163
10.4.5 按键状态机模块的修改 …… 165
10.4.6 顶层模块的修改 …… 169
10.5 下载程序到开发板进行调试 …… 171
10.6 今天工作总结 …… 171
10.7 夏老师评述 …… 172

参考文献 …… 174

第1章

第一天——数码管显示模块的设计

1.1 设计需求讲解

今天是我参加 FPGA 设计培训班的第 2 周。

上周夏宇闻老师把 Verilog 语言的基本概念、FPGA 工作原理和 FPGA 开发工具做了全面的介绍。通过几天的听讲和上机练习，我已初步掌握了用 ModelSim 进行简单模块仿真和综合的步骤和要点，也初步理解了可综合模块和专门用于仿真的测试模块有什么不同，当然我的语法知识还十分贫乏，电路结构的概念也十分模糊。因此心里不免有些担心，我能开始独立设计吗？怀着忐忑不安的心情，我走进了教室。

今天在课堂上，夏老师要求我们用十天时间完成一个能进行三位数加、减、乘、除四则运算的计算器，最后结果可以显示六位数字。他对设计目标和要求做了明确的说明，并介绍了几个相关的小模块，作为大家开始设计时的参考。他说先把功能做出来，再逐步提高性能，减小电路资源的消耗，让大家边设计边学习。

这些小模块容易理解吗？怎样才能把这些小模块加以改造，重新组装，最后做成一个能进行三位数的加、减、乘、除四则运算功能基本完善的小计算器？我心里实在没有把握。

今天上午夏老师的讲课非常重要，我一定要认真听讲，积极思考，多提问题，多上机练习，争取对设计过程和方法有完整彻底的理解。夏老师今天讲课的主要内容包括五个部分：

1）LED 数码显示部分；

2）4×4 键盘码扫描分析电路；

3）算术逻辑运算单元部分；

4）数制转换部分：BCD 码到二进制，二进制到 BCD 码；

5）运算操作过程的状态机。

他描述了这几个部分之间是如何联系的，讲解了最低设计要求和最高设计要求。他让我们每位同学各显其能，尽量达到最高标准。最后一天每位同学都必须要把自己完成的计算器样机演示给大家看，并允许别人操作，大家民主评比，看哪位同学设

计的样机性价比最高。

今天老师在课堂上演示了几个程序段，但并没有详细讲解。详细讲解的只有LED数码显示部分，要求大家在今天完成的也只有数码显示部分。其余部分将在随后几天详细讲述。

1.2 七段式数码管显示原理讲解

下面是我对夏老师LED数码显示讲课要点的总结。

七段式数码管就是使用七段点亮的线段来拼成常见的数字和某些字母，这种显示方式在数字电路中非常容易见到。再加上右下角显示的小数点，实际上一个显示单元包括了8根信号线。根据电路设计的不同，这些信号线可能高电平有效也可能低电平有效。我们通过控制这些线段为0或者1，即高或低电平即可达到显示效果。

单个数码管的结构图如图1-1所示。图中共有8个显示段(7个数字显示段，1个小数点显示段)，每个显示段由一个独立的发光二极管(LED)组成，通过8位数据线(sig[7:0])控制8个LED的亮和灭，由此显示出设计者想要显示的字符和小数点。本实验中采用的是至芯开发板，它采用共阳极数码管，因此若显示某段LED的另外一端为高电平(1)，则该段LED熄灭；若为低电平(0)，则该段LED点亮。例如，若想要显示数字"1"，则由图中可以看出1是由B和C两发光段组成的，所以只要向显示B、C段的两个LED的另外一端输出0，向显示其余各段的LED的另外一端输出1，即11111001(由高至低)，便可显示数字1。由此可知，0～F的显示码依次是：

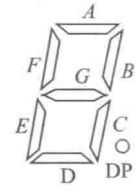


图1-1 数码管结构图

0：8'b11000000； 1：8'b11111001； 2：8'b10100100； 3：8'b10110000；
4：8'b10011001； 5：8'b10010010； 6：8'b10000010； 7：8'b11111000；
8：8'b10000000； 9：8'b10010000； A：8'b10001000； B：8'b10000011；
C：8'b11000110； D：8'b10100001； E：8'b10000110； F：8'b10001110；

上述显示码中的8表示8位数，'b表示二进制。对于有多个数码管的显示模块，将每一个模块都连接到FPGA的引脚会耗用大量的引脚资源。在实际电路中，每个数码管中的8个LED管的正极被连接在一起，通过一个通/断可控的三极管再连接到电源的阳极，控制该三极管的通/断并配合8位数据线(sig[7:0])的输入，就可依次点亮6个数码管中的某一个。这种方法就是广泛应用于数码显示的扫描点亮的方式。换言之，在点亮某一个数码管期间，在8位数据线(sig[7:0])上输入该管应该显示的电平，即依次点亮6个数码管中的某一个，显示该数码管应该显示的数字。当扫描的频率足够高时，由于数码管的余晖以及人眼的视觉延迟会觉得数码管是全部点亮的。通常建议使用60 Hz左右(即每个数码管导通约16 ms)的扫描频率即可达到

多位数码管同时点亮(只是人眼看上去是同时点亮,如果用照相机拍照即可发现并不是同时点亮)的效果。

由于是用作简易计算器显示,本计算器的功能不涉及浮点数据,因此忽略小数点显示段。本开发板使用的是共阳极数码管,其原理如图 1-2 所示,位选通过 PNP 三极管接电源,因此数码管为低电平选通,显示段为低电平点亮。

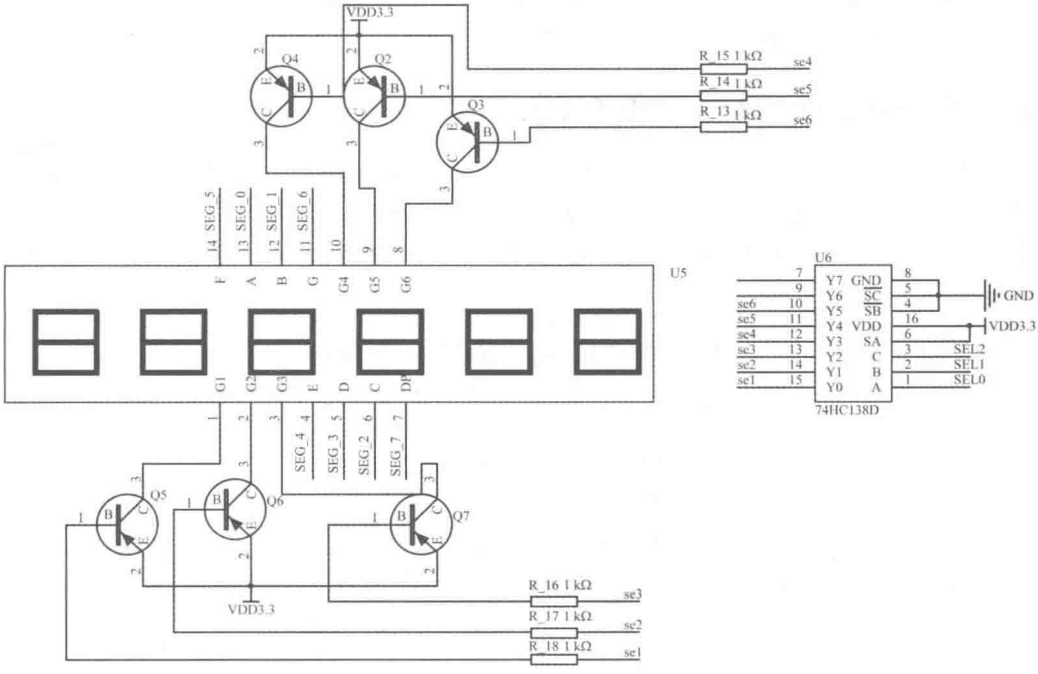


图 1-2　数码管显示模块原理图

由原理图可以看出,数码管的输入,也就是 FPGA 控制数码管的连线有段选 SEG-0～SEG-7,位选 SEL0～SEL2,共 11 根线。没有接触过数码管扫描显示的同学注意了,**要有段选和位选的概念。**“段”是指刚刚所说的 A、B、C、D、E、F、G、DP 这 8 根信号线,通过“段选”来控制数码管显示的是什么字;“位”是指这 6 个数码管其中的一个,注意这里是“个”而不是“段”,通过“位选”来控制这 6 个数码管哪一个会亮。也就是说,板子上有 6 个数码管,每个数码管有 8 段,段选控制数码管显示的是哪个数字,而位选则控制这个数字会出现在哪个位置,即个位、十位、百位、千位……本来位选也应该是由 6 根线来控制 6 个数码管的,但是考虑到这些数码管不会同时点亮,所以采用一个 3-8 译码器将 6 根线减少到 3 根,节约了 FPGA 的引脚资源。

那么数码管显示模块的输出就确定下来了,3 根选择线(SEL[2 : 0])和 8 根字段线(SEG[7:0])。接下来就可以开始写代码了。首先暂不考虑输入的问题,先让该模块输出一个固定的数字,比如输出 8 个 1,就是让 SEG 为 11111001,SEL 的取值从 000 到 101,正好对应上 6 个数码管,所以让 SEL 随时钟变化而增加,就会出现

6 个数码管上都显示 1。

1.3 设计工具使用讲解

想要完成设计，计算机上必须安装 Quartus II 和 ModelSim 两个工具。前者是 FPGA 综合分析工具；后者是验证设计的 Verilog 模块功能是否正确，是否与其他模块能正常交流数据的仿真工具。

1.3.1 Quartus II 工具的配置

单击计算机桌面上的 Quartus II 图标，启动 Quartus II 工具，稍等片刻，系统会弹出一个 Introduction 页面。单击页面下方的 next 按钮，系统弹出 Quartus II 的主菜单，单击 File→New Project Wizard（见图 1－3），随即弹出如图 1－4 所示的对话框。

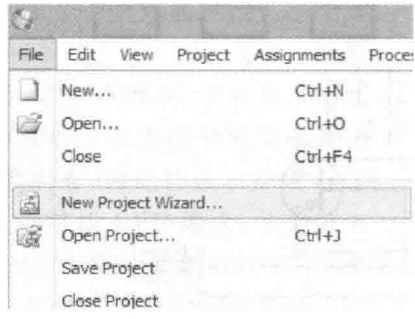


图 1－3　文件下拉菜单

在开始设计之前，Quartus II 工具自动弹出要求设计者回答：工程的名称、放置的目录路径、设计的模块名、所用器件型号、所用仿真工具类型、所用语言等重要设置。如图 1－4 所示的对话框中需要填写工程所在目录、工程名以及模块名。

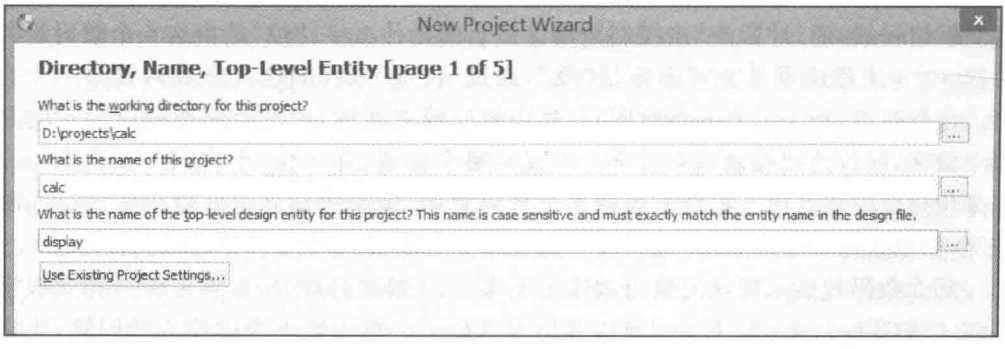


图 1－4　新工程向导 1

夏老师在讲解中特别提到：**所有的设计文件必须放置在自己觉得取用方便的硬**

盘目录中，任何设计目录都不能用中文命名。在这里我们把工程定义为计算器，取名 calc，填入第 2 行空格；模块为数码管显示模块，取名为 display，填入第 3 行空格（见图 1－4），写好之后单击对话框下面的 Next 按钮。

弹出的第 2 个对话框是要添加已经写好的代码文件。因为要从头开始写，所以这一页可以不予处理，直接按 Next 按钮。

弹出的第 3 个对话框是设置 FPGA 器件页。找到与开发板上对应的器件型号 EP4CE10F17C8（见图 1－5），选好之后单击对话框下面的 Next 按钮。

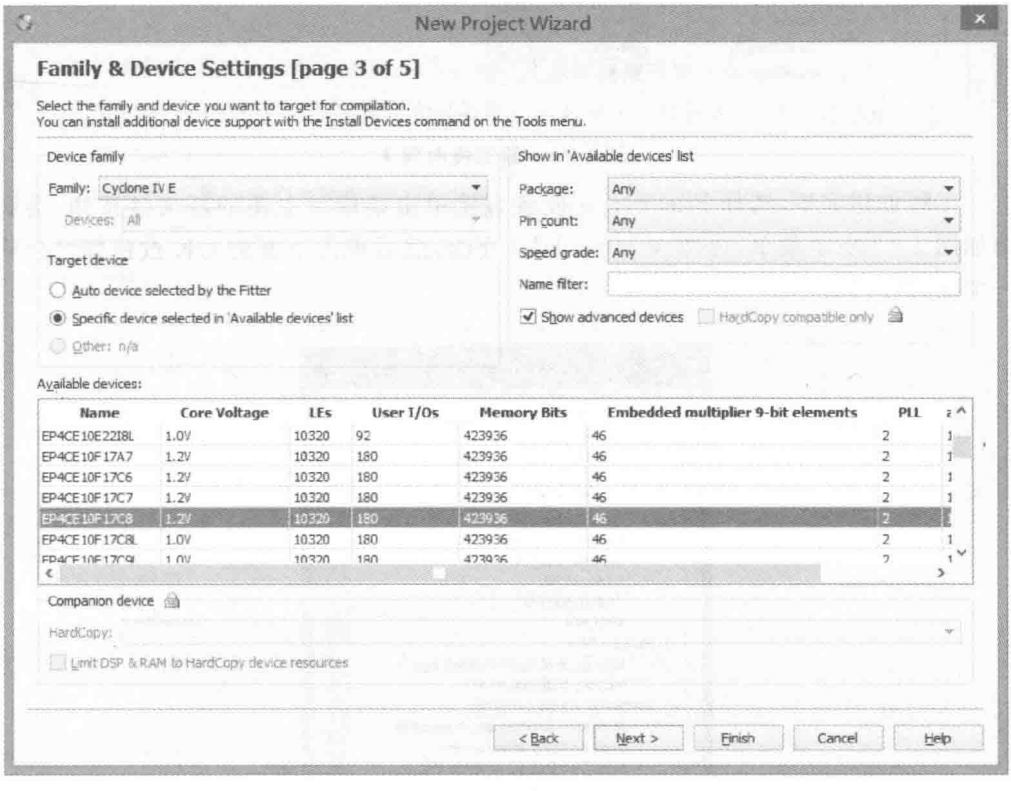


图 1－5　新工程向导 3

随即弹出的第 4 个对话框（见图 1－6）是配置 EDA 工具的界面。这里只需把对话框中 Simulation 工具栏的空格设置一下即可。选择 Tool Name（工具名称）为 ModelSim-Altera 或者 ModelSim，选择 Format(s)（语言格式）为 Verilog HDL，如图配置好后，单击对话框下面的 Next 按钮。

注意：选择哪种类型的仿真工具与用户所安装的 Quartus II 版本有密切的关系。该类型的仿真工具和语言必须得到 Quartus II 版本的支持。

随即弹出的第 5 个对话框是对之前所选的项做一个总结，请用户核对一下信息，看是否跟之前设置的一致，若正确无误，单击下面的 Finish 按钮即可。

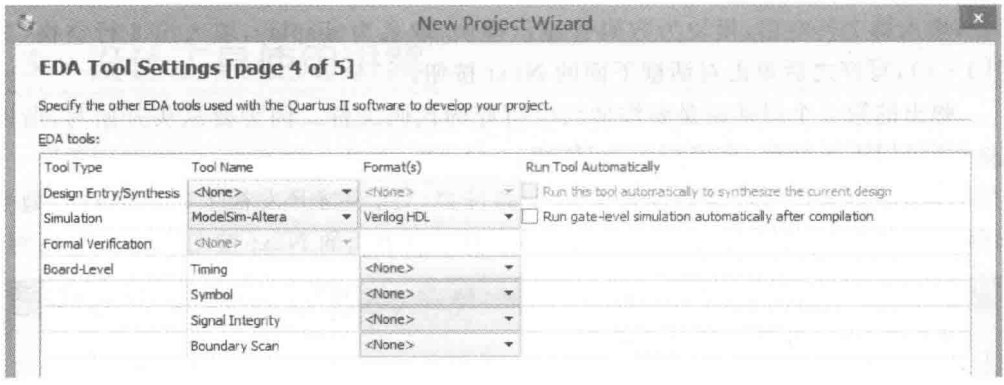


图 1-6　新工程向导 4

工程创建完毕，选择 File→New 或者直接单击菜单左上角的新文件按钮，会弹出如图 1-7 所示菜单，选择 Verilog HDL File，然后单击下面的 OK 按钮即可创建一个新的代码文件。

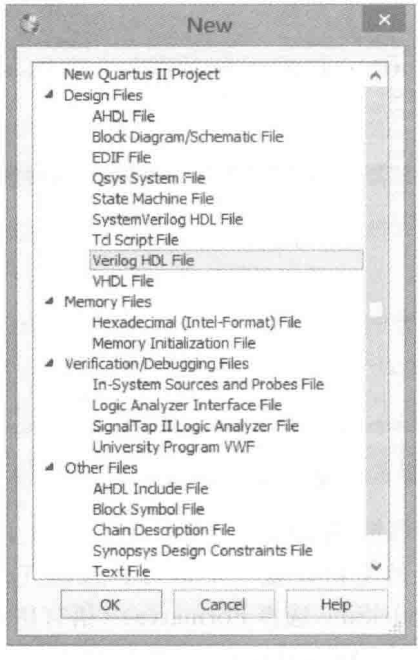


图 1-7　新建文件

1.3.2　数码管显示模块的可综合代码

至此，编写模块代码的准备工作已完成，可以开始编写代码了。以下是数码管显

示模块的可综合代码。

```
module display0(clk, rst_n , sel, seg);  //定义模块名为 display0,定义输入输出端口
//两个输入,一个时钟 clk,一个复位 rst_n(信号名后缀_n 表示该信号低电平有效)
    input clk;
    input rst_n;
//两个输出,位选 sel 和段选 seg
    output reg [2:0] sel;
    output reg [7:0] seg;
//数码管扫描需要一个慢时钟 clk_slow,而产生慢时钟则需要一个计数器 cnt
    reg [15:0] cnt;      //定义一个计数器 cnt,位宽大一些没关系,要保证够用
    reg clk_slow;
//这个 always 块用来产生慢时钟 clk_slow
    always @ (posedge clk)
    begin
        if(! rst_n)
        begin
            seg <= 8'b11111001;    //复位时输出数字 1 所对应的 seg
            cnt <= 0;
            clk_slow <= 1; //复位时 clk_slow 静止不动
        end
        else
        begin
            cnt <= cnt + 1; //复位结束后 cnt 开始计数
            clk_slow <= cnt[12]; //扫描没有必要非得是 60 Hz 整,大于 60 Hz 即可
        end
    end
//这个 always 块用于扫描数码管,也就是 sel 循环地变化
//时钟每一次上升沿 sel 变化一次,所以在括号里写上时钟上升沿作为触发条件
    always @ (posedge clk_slow or negedge rst_n)
    begin
        if(! rst_n)
        begin
            sel <= 0;     //复位时 sel 静止
        end
        else
```

```
        begin
            sel <= sel + 1;     //复位后 sel 开始扫描
            if(sel >= 5)
                sel <= 0;     //因为只有 6 个数码管,所以让 sel 在 0~5 之间循环
        end
    end
endmodule
```

写完之后将文件命名为 display0. v,按 ctrl+s 键保存。**注意,文件名必须与模块名相同,并将该模块设置为顶层模块(在文件名上单击鼠标右键,选择 Set as Top-Level Entity,如图 1-8所示)。**

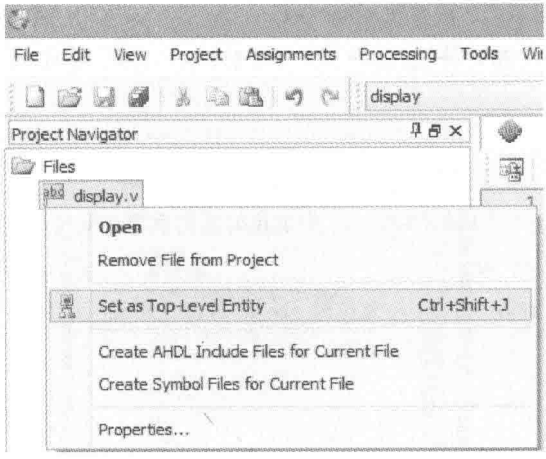


图 1-8　设置顶层模块

设置好后,按 Ctrl+K 键分析综合,让编译器检查代码的语法错误。没有了错误之后,还要为该代码写测试(testbench)代码。

1.3.3　显示模块的测试

下面简单介绍一下测试模块文件 testbench。当你编写完一个模块的代码并且成功将其综合成电路后,如何知道你编写的代码是否就综合成了你想要的电路了呢?硬件语言不像 C 语言那样按顺序执行代码,逐句调试成功即可得到最终结果,而是由编译器分析之后转变成一些由逻辑组成的电路。编译器很聪明,它通过我们写的代码就能得到相应的电路;但是写代码的我却很傻,傻到不能准确地写出我们想要的结果的代码,导致代码编译之后下载到开发板里,什么反应都没有,或者奇怪的错误

一大堆。当然，一些简单的代码可以通过一些输出（LED灯、数码管等）观察到逻辑的错误，但是当逻辑比较复杂、时序快、信号量多的时候，这种测试方法显然就不给力了。而通过调用ModelSim进行仿真，可以看出模块中每个信号的变化过程，这样就可以监测这些信号是否正确变化，达到查错的目的。有时候，因为一个信号的错误，就会导致结果大不一样，所以如果你不喜欢写testbench测试代码，你就有可能离正确只差那么一点点却无法企及。

夏老师在课堂上曾多次提醒我们编写测试代码的重要性，他说磨刀不误砍柴工，养成良好的测试习惯非常重要。在编程实践中，我深刻体会到夏老师的话千真万确。按照他的建议多写测试代码进行仿真，使我的代码成功率大大提高，减少了盲目修改的时间，大大加快了工作的进度。通过了RTL仿真和布局布线后仿真的代码块，下载到开发板里出错的概率非常小，而没有经过仿真的代码，下载到开发板几乎总是不能运行，即使能运行也会存在各种奇怪的问题。

由于刚才这段代码的输入只有时钟clk和复位信号rst_n，所以只需要定义这两个信号为reg型的变量，作为数码管显示代码的输入。和刚刚的方法一样，新建一个Verilog文件，并写测试代码testbench如下：

```
`timescale 1ns/1ns //设置延迟时间单位和时间精度，注意这句结束没有分号
module display_tb0; //测试模块没有输入输出端口，所以只需写模块名 display_tb0
//声明激励信号
    reg clk;
    reg rst_n;
//定义例化中的连线
    wire [2:0] sel;
    wire [7:0] seg;
//对被测试模块进行例化
    display0 u1(.clk(clk), .rst_n(rst_n), .sel(sel), .seg(seg));
    initial
    begin
        clk = 1; rst_n = 0; //给变量信号赋初始值，使复位信号有效
        #101 rst_n = 1; //经过一段时间(101ns)后将复位信号置为无效
        #10000000 $stop; //经过10ms之后停止仿真
    end
//生成50MHz的时钟(开发板上时钟为50MHz)
    always #10 clk = ~clk;
endmodule
```

代码写好之后，按 Ctrl＋S，Ctrl＋K 组合键，让编译器检查一下有没有语法错误。通过了之后就可以进行 simulation 的设置了。在 Quartus II 主菜单顶层工具栏上，单击 Assignments→Settings，弹出设置页，左侧选择 EDA Tool Settings 里的 Simulation 选项，右侧选择 Compile test bench 并单击 Test Benches（见图 1－9）。

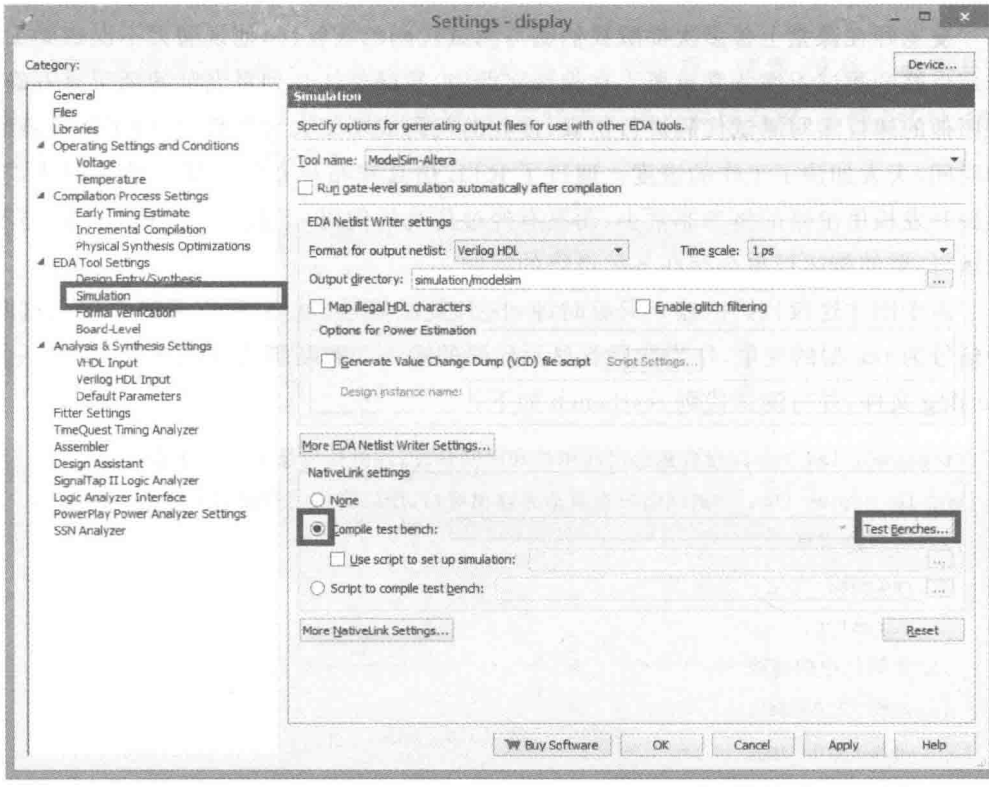


图 1－9　仿真设置

在弹出的页面中选择 new 来新建一个测试，前两行就写 testbench 的模块名 display_tb，然后在下面添加文件，找到刚刚写好的 testbench 和被测试文件，并添加进去，如图 1－10 所示。看到文件已经添加到下面的空白框里就可以单击 OK 按钮了。