

**数字电子技术基础**

**实验报告**

**题目：实验一 TTL集成门电路逻辑变换**

小组成员： 姓名 学号

小组成员： 姓名 学号

组 号： 组号

## 实验一 TTL集成门电路逻辑变换

### 一、实验目的

掌握HDL语言用法

### 二、实验要求

要求 1：学习并掌握硬件描述语言 VHDL；熟悉门电路的逻辑功能，并用硬件描述语言实现门电路的设计。参考“参考内容 1”中给出的与门源程序，编写一个异或门逻辑电路。1）用 QuartusII波形仿真验证；2）下载到 DE0 开发板验证。

要求 2：熟悉中规模器件译码器的逻辑功能，用硬件描述语言实现其设计。参考“参考内容 2”中给出的将 8421BCD 码转换成 0-9 的七段码译码器源程序，编写一个将二进制码转换成 0-E 的七段码译码器。1）用 QuartusII 波形仿真验证；2）下载到 DE0 开发板，利用开发板上的数码管验证。

要求 3：熟悉时序电路计数器的逻辑功能，用硬件描述语言实现其设计。参考“参考内容 3”中给

出的四位二进制计数器的源程序，编写一个计数器实现 0-E 计数。用 QuartusII 波形仿真验证；

要求 4：熟悉分频电路的逻辑功能，并用硬件描述语言实现其设计。参考“参考内容 4”中给出的50M 分频器的源程序，编写一个能实现占空比 50%的 5M 和 50M 分频器即两个输出，输出信号频率分别为 10Hz 和 1Hz。下载到 DE0 开发板验证。（提示：利用 DE0 板上已有的 50M 晶振作为输入信号，通过开发板上两个的 LED 灯观察输出信号）。电路框图如下：



要求 5：利用已经实现的 VHDL 模块文件，顶层文件采用原理图设计方法，实现 a,b,c,d,E,学号 5位，E，d，c，b，a 计数自动循环显示，频率 1Hz 和 10Hz 可以切换。（提示：如何将 VHDL 模块文件在顶层原理图文件中引用，参考参考内容 5）

### 三、实验设备

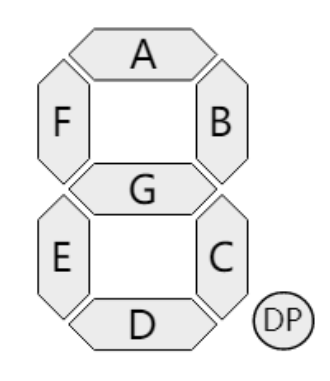
开发板DE0

### 四、实验原理

**七段数码管原理:**

七段数码管由七个LED数码管以共阳或共阴的形式连接组成。对于共阳型数码管，每个LED数码管的阳极都通过导线连接，并与正极Vcc相连，七根阴极则连接到控制芯片的输出端口。由控制芯片控制不同的输出端口接通或者断开，就可以控制七段数码管内部的LED是否亮起。共阴型七段数码管则是将上述原理反过来，即七个LED数码管阴极通过导线连接并与负极GND相连，而阳极则连接到控制芯片的输出端口。

七段分别标记为A、B、C、D、E、F、G，还有一个小数点DP



### 五、实验内容

**由于要求5集成前4项任务的所有工作，这里我们直接展示要求5实验内容**

1. 电路搭建

电路搭建整体由Verilog编写，下面具体介绍对应代码以及实现相关功能

**（1）1Hz时钟与10Hz时钟分频计数器以及切换模块**

Code:

module my\_clk(

clk,

sel,

outclk

);

input clk;

input sel;

output outclk;

//20ns

parameter cnt1 = 50\_000\_000;

reg [30:0] cnt\_1;

reg clk1,clk2;

always@(posedge clk)

    if(!sel)

        if (cnt\_1 == cnt1)

            cnt\_1 <= 0;

        else

            cnt\_1 <= 1 + cnt\_1;

    else

        cnt\_1 <= 0;

always@(posedge clk)

    if(cnt\_1 == cnt1)

        clk1 <= 1;

    else

        clk1 <= 0;

parameter cnt2 = 5\_000\_000;

reg [30:0] cnt\_2;

always@(posedge clk)

    if(sel)

        if (cnt\_2 == cnt2)

            cnt\_2 <= 0;

        else

            cnt\_2 <= 1 + cnt\_2;

    else

        cnt\_2 <= 0;

always@(posedge clk)

    if(cnt\_2 == cnt2)

        clk2 <= 1;

    else

        clk2 <= 0;

assign outclk = sel ? clk2 : clk1;

endmodule

**clk:时钟信号,sel:选择位,outclk:输出分频时钟信号**

开发板时钟频率为50MHz,想要获得1Hz和10Hz只需要添加50\_000\_000计数器和5\_000\_000计数器即可，再通过sel位三元运算选择输出时钟即可

**(2)7位数码管逻辑编写和时钟模块实例化**

module why(

clk,

sel,

led

);

input clk;

input sel;

output reg [6:0] led;

wire outclk;

reg [3:0] cnt;

my\_clk clk\_inst(

.clk(clk),

.sel(sel),

.outclk(outclk)

);

always@(posedge clk)

    if (outclk)

        if(cnt == 4'b1110)

            cnt <= 0;

        else

            cnt <= cnt + 1;

    else

        cnt <= cnt;

reg [3:0] logic;

always@(posedge clk)

    case(cnt)

        4'b0000:logic<=0;

        4'b0001:logic<=2;

        4'b0010:logic<=0;

        4'b0011:logic<=9;

        4'b0100:logic<=8;

        4'b0101:logic<=0;

        4'b0110:logic<=2;

        4'b0111:logic<=1;

        4'b1000:logic<=1;

        4'b1001:logic<=5;

        4'b1010:logic<=14;

        4'b1011:logic<=13;

        4'b1100:logic<=12;

        4'b1101:logic<=11;

        4'b1110:logic<=10;

        default:logic<=0;

        endcase

always@(posedge clk)

    case(logic)

        4'b0000:led<=7'b1000000;

        4'b0001:led<=7'b1111001;

        4'b0010:led<=7'b0100100;

        4'b0011:led<=7'b0110000;

        4'b0100:led<=7'b0011001;

        4'b0101:led<=7'b0010010;

        4'b0110:led<=7'b0000010;

        4'b0111:led<=7'b1111000;

        4'b1000:led<=7'b0000000;

        4'b1001:led<=7'b0010000;

        4'b1010:led<=7'b0001000;

        4'b1011:led<=7'b0000011;

        4'b1100:led<=7'b1000110;

        4'b1101:led<=7'b0100001;

        4'b1110:led<=7'b0000110;

        default:led<=7'b1111111;

        endcase

endmodule

本模块通过实例化前面编写的时钟模块实现分频时钟信号获得

再编写数码管逻辑具体看下图:

always@(posedge clk)

    case(logic)

        4'b0000:led<=7'b1000000;

        4'b0001:led<=7'b1111001;

        4'b0010:led<=7'b0100100;

        4'b0011:led<=7'b0110000;

        4'b0100:led<=7'b0011001;

        4'b0101:led<=7'b0010010;

        4'b0110:led<=7'b0000010;

        4'b0111:led<=7'b1111000;

        4'b1000:led<=7'b0000000;

        4'b1001:led<=7'b0010000;

        4'b1010:led<=7'b0001000;

        4'b1011:led<=7'b0000011;

        4'b1100:led<=7'b1000110;

        4'b1101:led<=7'b0100001;

        4'b1110:led<=7'b0000110;

        default:led<=7'b1111111;

        endcase

endmodule

通过判断wire logic的值来输出对应符号哪些led的高低电平来实现数字示

控制显示的逻辑如下图:

always@(posedge clk)

    case(cnt)

        4'b0000:logic<=0;

        4'b0001:logic<=2;

        4'b0010:logic<=0;

        4'b0011:logic<=9;

        4'b0100:logic<=8;

        4'b0101:logic<=0;

        4'b0110:logic<=2;

        4'b0111:logic<=1;

        4'b1000:logic<=1;

        4'b1001:logic<=5;

        4'b1010:logic<=14;

        4'b1011:logic<=13;

        4'b1100:logic<=12;

        4'b1101:logic<=11;

        4'b1110:logic<=10;

        default:logic<=0;

        endcase

每来一个时钟上升沿信号进行cnt自增然后logic值被更新位要显示的数字

组员一学号为:2022302098 取后五位 02098

组员二学号为:2022302115 取后五位 02115

接着显示从E到A

故顺序为 **0 2 0 9 8 0 2 1 1 5 14 13 12 11 10**

**板集验证可由当堂成绩可以考察是否做出故不展示**

### 六、实验过程中的问题

1. 第一次再QuartusII中编写Verilog，编译的过程中在文件命名出了问题，

主模块命名要与文件名称一致，不然会编译报错.

### 七、心得体会

1.QartusII的编译速度很快，很酷！学会在QuartusII上编写Verilog