实验三 芯片的封装及应用

一、实验目的

1. 掌握Verilog语言框架，编程及调试的方法；

2. 熟悉Verilog的基本语法；

3. 掌握Vivado开发平台及FPGA开发板的使用。

二、实验内容

1. 完成74LS161计数器芯片的实现、测试及6进制计数器的实现，将程序下载到FPGA开发板进行验证；

2. 分析电路中的竞争与冒险，给出解决方案并验证；

3. 将芯片及相关模块封装为IP核，通过原理图设计实现10进制计数器，观察、分析仿真波形和电路图；

三、实验要求

1. 画出模块的电路图；

2. 分析电路的仿真波形和板卡验证结果；

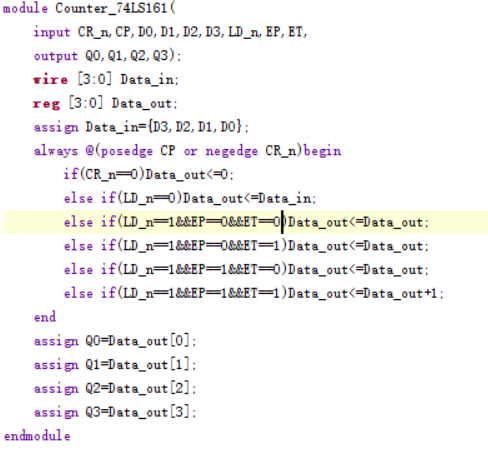
3. 记录设计和调试过程。

四、实验过程及结果分析

**对实验代码、仿真结果等进行分析说明，不能只贴截图。**

**1、6 进制计数器设计**

**（1）74LS161 计数器芯片的实现**



这一模块模拟了 74LS161 这种 4 位二进制计数器的功能。以下是该模块的各部分详细

解释：

 input CR\_n, CP, D0, D1, D2, D3, LD\_n, EP, ET, output reg Q0, Q1, Q2, Q3：这是模块的接口声明，定义了模块的输入和输出。CR\_n, CP, D0-D3, LD\_n, EP, 和 ET 是输入信号，而 Q0-Q3 是输出信号。

 wire [3:0] Data\_in 和 reg [3:0] Data\_out：Data\_in 是一个 4 位的输入向量，Data\_out 是一个 4 位的输出寄存器。

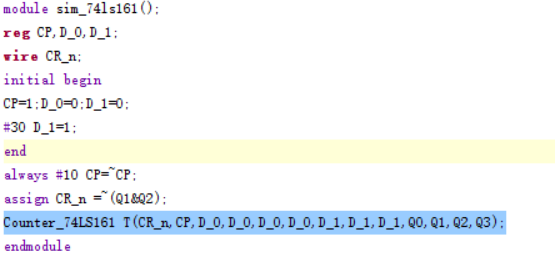
 assign Data\_in = {D3, D2, D1, D0}：此行将 4 个单独的输入位 D0-D3 合并到一个 4 位的向量 Data\_in 中。

 always @(posedge CP or negedge CR\_n) begin...end：这个是一个敏感于 CP 上升沿或CR\_n 下降沿的过程。也就是说，只有当 CP 信号从 0 变为 1，或者 CR\_n 信号从 1 变

为 0 的时候，这个过程中的逻辑才会被执行。在这个过程中，有一个条件语句根据 CR\_n, LD\_n, EP,和 ET 的值来改变 Data\_out的值。如果 CR\_n 为 0，Data\_out 被重置为 0；如果 LD\_n 为 0，Data\_out 赋值为Data\_in；如果 LD\_n, EP, ET全部为 1，Data\_out增加 1；其他情况，Data\_out 保持不变。

 assign Q0 = Data\_out[0]; assign Q1 = Data\_out[1]; assign Q2 = Data\_out[2]; assign Q3 = Data\_out[3]：这些语句将 Data\_out 中的每一位分别赋给输出变量 Q0-Q3。

所以，这个模块的主要功能是，根据输入的四个数据位 D0-D3 和控制信号 CR\_n, LD\_n, EP, ET，控制 4 位计数值的输出 Q0-Q3



这是 Counter\_74LS161 模块的测试模块，以下是该模块的各部分详细解释：

 reg CP, D0, D1, CR\_n ；：定义了 4 个寄存器变量 CP, D0, D1, 和 CR\_n。

 wire Q0, Q1, Q2, Q3； ：定义了 4 个线变量 Q0, Q1, Q2, 和 Q3。

 initial begin...end ：这是一个初始块，它在模拟开始时运行一次。在这个块中，首先设置 CP 为 1， D0 和 D1 为 0，然后在模拟时间经过 30 单位后，将 D1 设置为 1。

 always #10 CP = ~CP; ：这是一个始终块，它在模拟时间每过 10 单位时，对CP取反

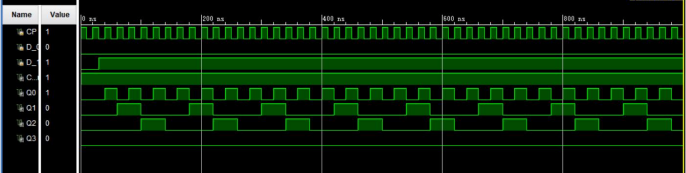
 assign CR\_n = ~(Q1 & Q2); ：这是一个连续赋值语句，将 Q1 和 Q2 的与值取反后赋

给 CR\_n。

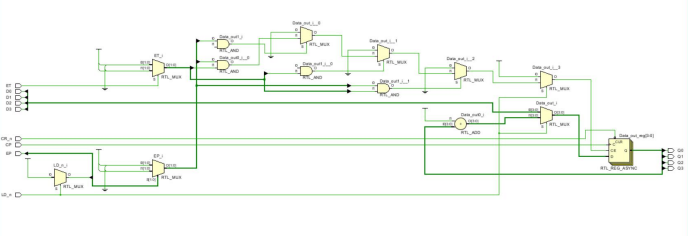
 Counter\_74LS161 T(...); ：这是一个模块实例化语句，创建了一个 Counter\_74LS161 的实例 T，并将定义的寄存器变量和线变量连到模块的输入和输出上。

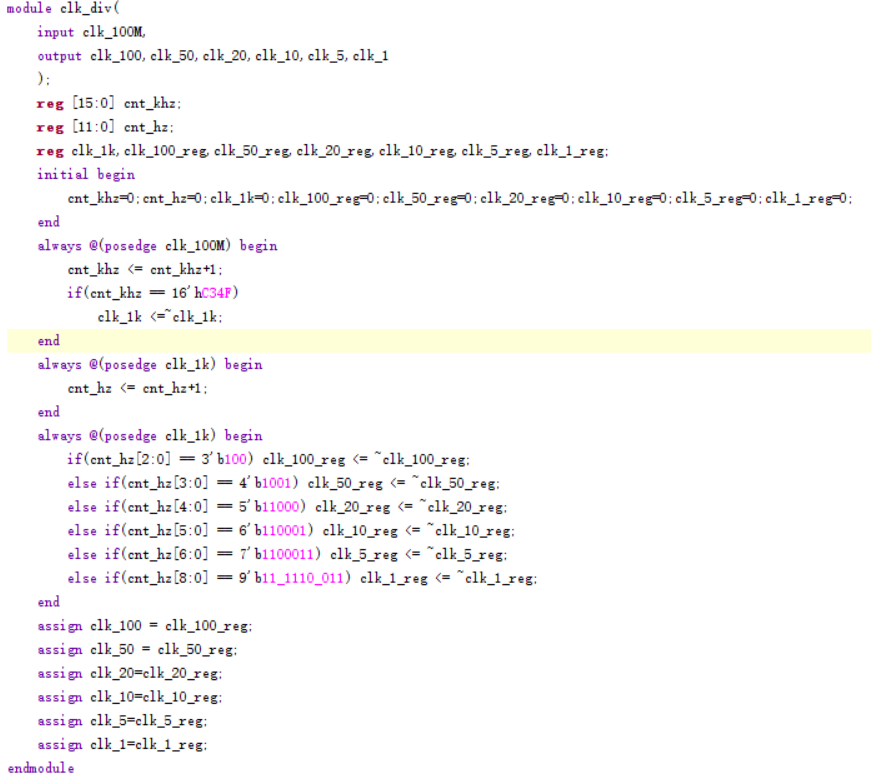
所以， sim\_74ls161 模块的主要功能是生成输入信号，并观察 Counter\_74LS161 模块

的输出结果，以此来测试和模拟 Counter\_74LS161 模块的行为。测试结果如下：



对应的RTL电路分析图如下：

**（2）六进制计数器的实现**



这是一个时钟分频模块，它以 100 MHz 的时钟输入（clk\_100M），并输出一系列分频

后的时钟信号，包括 100 kHz、50 kHz、20 kHz、10 kHz、5 kHz 和 1 kHz 的时钟。以下

是该模块的各部分详细解释：

 **模块输入/输出接口：**

1. input clk\_100M：这是输入的时钟信号，频率为 100 MHz。

2. output clk\_100, clk\_50, clk\_20, clk\_10, clk\_5, clk\_1：这些是模块的输出信号，表示不同的时钟频率。

 **寄存器声明：**

1. reg [15:0] cnt\_khz 和 reg [11:0] cnt\_hz：这两个寄存器用于在不同频率的时钟下进行计数。

2. reg clk\_1k, clk\_100\_reg, clk\_50\_reg, clk\_20\_reg, clk\_10\_reg, clk\_5\_reg, clk\_1\_reg：这些寄存器用于存储生成的不同频率的时钟信号。

 **初始化过程**：在 initial begin...end 块中，所有的寄存器都被初始化为 0。

 **计数器及时钟生成**：

1. always @(posedge clk\_100M) 块：这个块定义了在 100 MHz 时钟上升沿发生时的

动作。在这个块中，cnt\_khz 寄存器每次增加 1。当它达到 16'hC34F（约等于 50000，

代表 1 毫秒的 100MHz 时钟周期数）时，1kHz 的时钟信号 clk\_1k 翻转，且计数器

cnt\_khz 清零。

2. always @(posedge clk\_1k) 块：这个块定义了在 1 kHz 时钟上升沿发生时的动作。

在这个块中，cnt\_hz 寄存器每次增加 1。

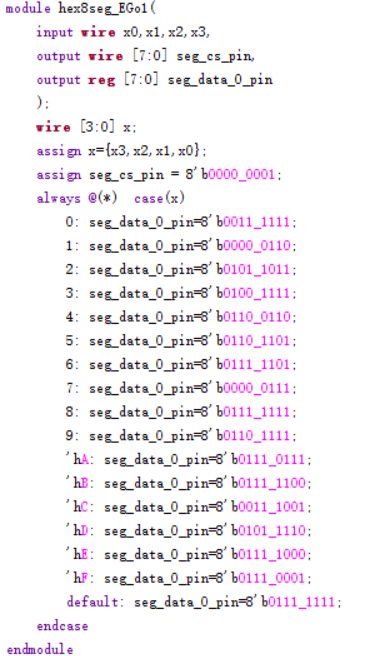
3. 另一个 always @(posedge clk\_1k) 块：这个块在 1 kHz 时钟上升沿发生时，根据

cnt\_hz 的值来翻转不同频率的时钟信号。每个条件语句都是检查 cnt\_hz 的某些位是否

等于一个特定的二进制值，如果等于就翻转相应频率的时钟信号。

 **信号赋值**：assign 语句用于将寄存器的值赋给模块的输出端口，也就是将内部生成的

各种频率的时钟信号赋给输出端口



这是一个用于驱动 EGo1 开发板的模块，以下是该模块的各部分详细解释：

 **模块输入/输出接口**：

1. input wire x0，x1，x2，x3：模块的输入信号，这是四位二进制数的输入，表示从 0

到 15 的十进制数或从 0 到 F 的十六进制数。

2. output wire [7:0] seg\_cs\_pin：模块的输出信号，这是 8 位位选信号，用于驱动数码管的每一位。

3. output reg [7:0] seg\_data\_o\_pin：模块的输出信号，这是 8 位段选信号，用于驱动数码管的每一段。

 **信号定义和赋值**：

1. wire [3:0] x：定义了一个 4 位宽的线网（wire）变量 x，将输入信号组合为一个 4 位的信号。

2. assign x = {x3, x2, x1, x0}：将 4 位输入信号合并为一个 4 位的信号 x。

3. assign seg\_cs\_pin = 8'b0000\_0001：位选信号固定为第一位有效，说明这是一个单位

数码管或者在多位数码管中只驱动第一位。

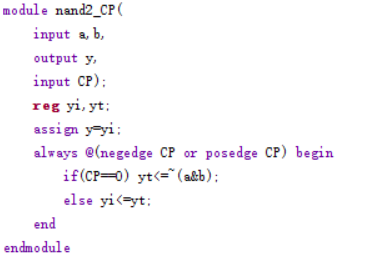
 **逻辑处理**：

通过 always @(\*) case(x) 块对输入的信号 x 进行解码，根据 x 的值（0 到 15）生

成对应的段选信号。这个模块将 4 位的输入解码为可以驱动数码管显示相应数字或字母

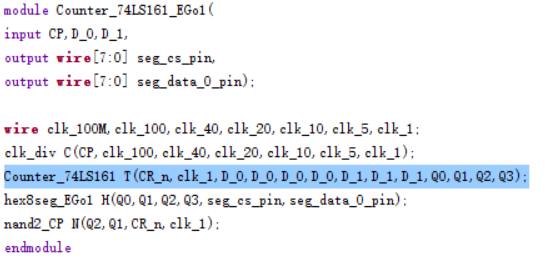
的 8 位段选信号。例如当 x 为 2时，seg\_data\_o\_pin 被设置为 8'b101\_1011，这将在数码管上显示数字 2。当 x 为'A'（十六进制）时，seg\_data\_o\_pin 被设置为8'b111\_0111，这将在数码管上显示字母 A。这里的每一位都对应数码管的一段，通过改变每一位的状态（0 或 1），可以在数码管上显示不同的数字和字母。

nand2\_cp模块：



该模块使用时钟同步电路，利用触发器进行拍打延迟消除险象，在CP下降沿或者上升沿到来之时，将两个输入信号进行逻辑与运算，然后取反，得到输出信号。

Counter\_74LS161\_EGo1模块：



该模块以 74LS161 芯片为基础，实现了一个计数器，并将计数结果通过 EGo1 开发板的

数码管显示出来。以下是对这个模块各部分的详细解释：

 **模块输入/输出接口**：

1. input CP, D\_0, D\_1：模块的输入信号，包括时钟输入（CP）和两个控制信号 D\_0、D\_1。

2. output wire [7:0] seg\_cs\_pin：模块的输出信号，是一个 8 位的数码管位选信号，用

于控制数码管的每一位。

3. output wire [7:0] seg\_data\_0\_pin：模块的输出信号，是一个8位的数码管段选信号，

用于控制数码管的每一段。

 **信号定义和分频模块**：

1. 定义了一系列的时钟信号：wire clk\_100M, clk\_100, clk\_40, clk\_20, clk\_10, clk\_5, clk\_1。

2. clk\_div c(CP, clk\_100, clk\_40, clk\_20, clk\_10, clk\_5, clk\_1)：这是一个时钟分频模块，根据输入的时钟信号 CP，生成一系列的分频时钟信号。

 **计数器模块**：

Counter\_74LS161 T(CR\_n, clk\_1, D\_0, D\_0, D\_0, D\_0, D\_1, D\_1, D\_1, Q0,Q1,Q2,Q3)：

这是一个基于 74LS161 芯片的计数器模块，根据输入的时钟信号和控制信号，生成计数器的输出结果。

 **数码管显示模块**：

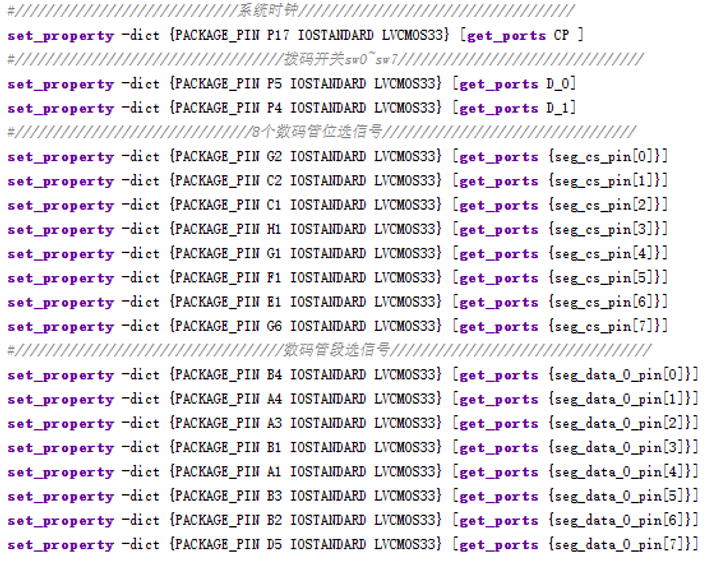
hex8seg\_EGo1 H(Q0, Q1, Q2, Q3, seg\_cs\_pin, seg\_data\_o\_pin)：这是一个 EGo1 开发板的数码管显示模块，根据计数器的输出结果，生成对应的数码管显示信号。

 **逻辑门模块**：

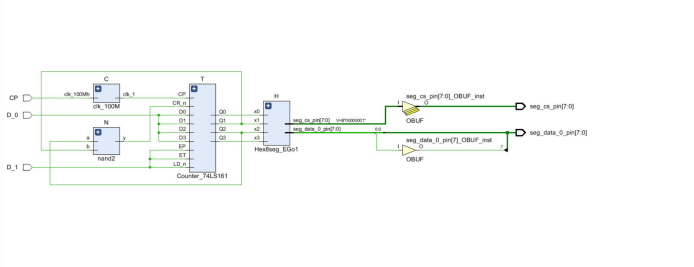
1. nand2 u(Q2, Q1, CR\_n)：这是一个与非门模块，根据计数器的两个输出信号 Q2、Q1，

生成与非门的输出信号 CR\_n

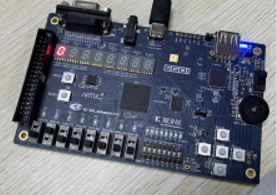
**约束文件：**

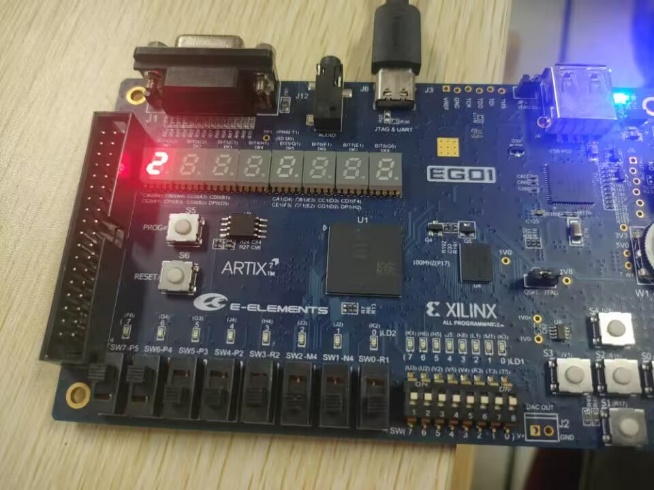
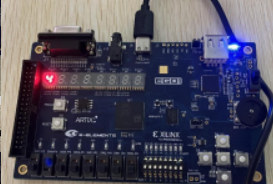
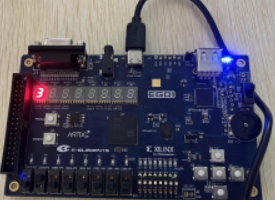


这是六进制计数器约束文件部分，用来设置设计中引脚的物理特性和位置。约束文件在 实现设计的过程中起到非常重要的作用，它决定了逻辑设计与硬件如何互相映射。



这是经过RTL分析后得到的最终电路图。烧录至开发板后，运行结果如下：

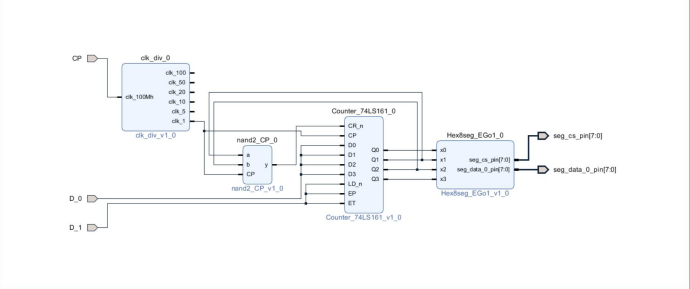
 

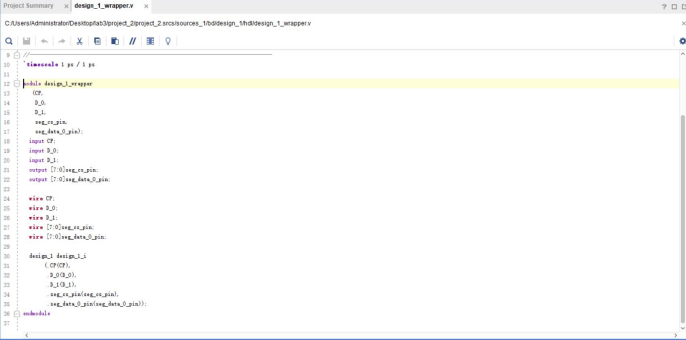
 



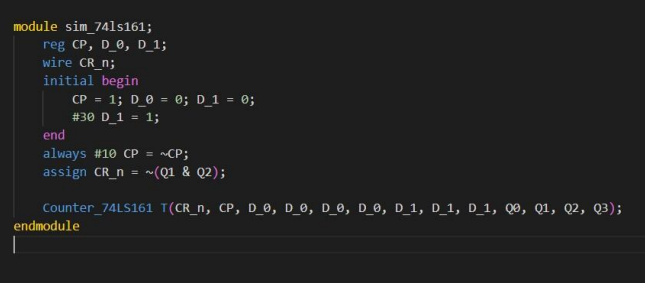
实验结果与预期一致

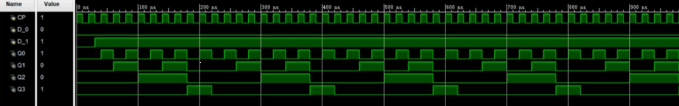
**2、10 进制计数器设计**





依照实验指导书，我选择使用 IP 封装的方法构建 10 进制计数器电路。我首先将 Counter\_74LS161、nand2\_CP、clk\_div、Hex8seg\_EGo1 模块分别单独封装成 IP 核，然后我 新建了一个项目文件并添加了这些 IP，最后我通过鼠标拖动连接各模块接口连线完成了 10 进制计数器电路设计。设计完成后，我编写了相应的测试程序，程序及测试结果如下





由图可得，该电路满足 10 进制计数的功能

五、调试和心得体会

**记录实验过程中遇到的问题和调试解决过程，心得体会等。**

在完成第三次数电实验之后，我对硬件设计和电路理论有了更深入的理解。通过设计各种模块，如计数器、时钟分频器、8 位数码管以及与非门等，我不仅理解了这些模块的工作原理，还尝试了使用 Verilog 硬件描述语言进行模块设计。在这次实验中我理解到了芯片模块化设计的重要性。每个电路设计都可以看作是一个独立的模块，这些模块可以独立设计、测试和验证，然后在最后的设计中组装起来形成一个完整的芯片。这种模块化的设计思想大大简化了设计过程，同时也提高了代码的复用性。此外，我还尝试了模块的 IP 封装和使用，在将相应的 IP 核加载进入 IP 目录后，我使用鼠标连接相应的输入输出即可设计出功能更强的电路。这对于我而言是数字电路设计的全新体验。

总的来说，这一系列的实验让我收获颇丰。通过动手实践，我对数字电路设计有了更深的理解。我相信，这些实验所学的知识和技能，将对我未来的学习和工作产生深远的影响。