第5章 远程实验

本章的内容是为了远程实验方便，单独设计的实验内容和实验方法，为了方便读者，对各实验中所涉及的主要实验原理进行了概述。

5.1 基本实验

实验1 点亮数字人生

一、实验目的

- 了解实验系统原理和使用方法。
- 掌握 VHDL 程序基本结构，学习编写简单的 VHDL 程序。
- 了解 Quartus Ⅱ的设计流程，掌握 Quartus Ⅱ的使用方法。
- 点亮数码管。

二、实验设备

- PC 一台
- Quartus 软件 一套
- 实验系统 一套

三、实验要求

从这个实验开始，将开启数字世界大门，带你进入数字世界。

数码管是认识数字电路的最直观的一扇窗，请根据数码管的译码要求，用硬件描述语言设计出通过输入端控制数码管输出，显示有规律的数字，设计时请参考本实验最后的设计提示。

四、实验原理

译码器是一个多输入、多输出的组合逻辑电路。它的作用是把给定的代码进行“翻译”，变成相应的状态，使输出中有相应的状态。译码器在数字系统中有广泛的用途，不仅用于代码的转换、终端的数字显示，还用于数据分配，存储器寻址和组合控制信号等。不同的功能可选用不同种类的译码器。

译码器可以分为通用译码器和显示译码器两大类，本次实验需要完

成一组数码显示译码器。

七段发光二极管(LED)数码管是目前最常用的数字显示器。一个 LED 数码管可以用来显示 0～9、A～F 十六进制数和一个小数点。LED 数码管要显示 BCD 码所表示的十进制数字就需要有一个专门的译码器。

实验平台上的引脚配置如下：

- 输入端——K1 K2 K3 K4 pin_90、pin_97、pin_96、pin_100(4 个按键作为输入，实验者也可以选择其他可控开关绑定引脚)。
- 输出端——a～g pin_178、pin_177、pin_175、pin_174、pin_173、pin_171、pin_170(以其中一个数码管为例，数码管引脚分配请参照附录内容)，数码管的功能表如表 5.1 所示，其中各段的对应关系如图 5.1 所示。

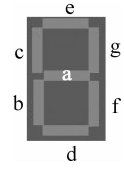

图 5.1 数码管各段标识图

当输入相应的数据时，需要完成相应的译码使之有相应的输出，从而使 LED 数码管相应位置发亮，显示出译码的结果。

表 5.1 数码管译码编码

输入					输出						
数字	A	B	C	D	a(6)	b(5)	c(4)	d(3)	e(2)	f(1)	g(0)
0	0	0	0	0	0	1	1	1	1	1	1
1	0	0	0	1	0	0	0	0	0	1	1
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	0	0	1	1	1	1
4	0	1	0	0	1	0	1	0	0	1	1
5	0	1	0	1	1	0	1	1	1	1	0
6	0	1	1	0	1	1	1	1	1	1	0
7	0	1	1	1	0	0	0	0	1	1	1
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	0	1	1	1	1	1
a	1	0	1	0	1	1	1	0	1	1	1
b	1	0	1	1	1	1	1	1	0	1	0
c	1	1	0	0	0	1	1	1	1	0	0
d	1	1	0	1	1	1	0	1	0	1	1
e	1	1	1	0	1	1	1	1	1	0	0
f	1	1	1	1	1	1	1	0	1	0	0

五、实验设计

这个实验是最基础的实验，主要是为了让读者熟悉 VHDL 语言，了解掌握硬件程序的编写规范，同时为以后的实验打下基础。建议若有条件的话，到实验室完成这个实验，了解实验箱的操作。

若在软件平台上完成实验，则可以将输出引脚绑定在一段有数码管的连续的地址总线上，实验结果将在主程序界面后方显示出来(如图 5.2 所示)。

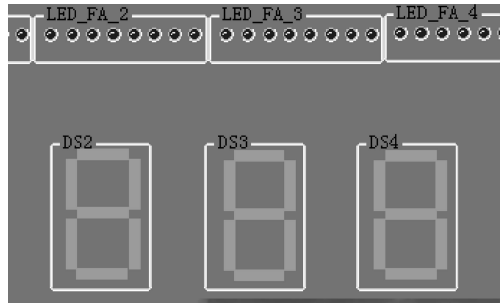

图 5.2 LED 数码管

1. 设计提示

本实验是进入数字世界的第一个实验，要求有规律的数字显示，其规律是可以根据各人的理解设计的，可以完全参考原理中的描述，也可以设计一组有规律的显示方式，例如奇偶数、顺序显示十进制数或者十六进制数等，可以仅仅点亮一个数码管，也可以依据引脚绑定表，尝试同时点亮多个数码管。

2. 本地实验操作

首先，按照前面所讲到的 Quartus 软件的操作步骤建立工程文件，然后在 File 菜单中选择 New 命令，在弹出的对话框中选择 VHDL File 选项，然后单击 OK 按钮，如图 5.3 所示。

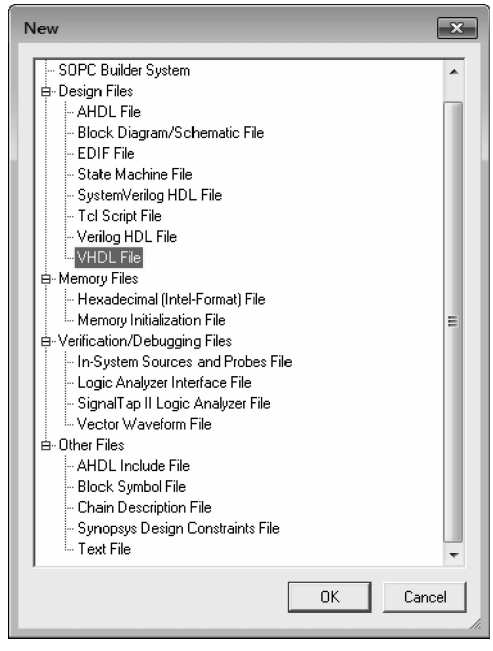

图 5.3 选择 VHDL File 选项

这样就建立好了一个空白文件，在空白处编写代码，如图 5.4 所示。

请注意：图 5.5 中标出的几处的名称要保持一致，否则编译会报错。

代码编写完成后进行编译，选择工具栏中的编译按钮（图 5.6 中用黑色圆圈标出）。

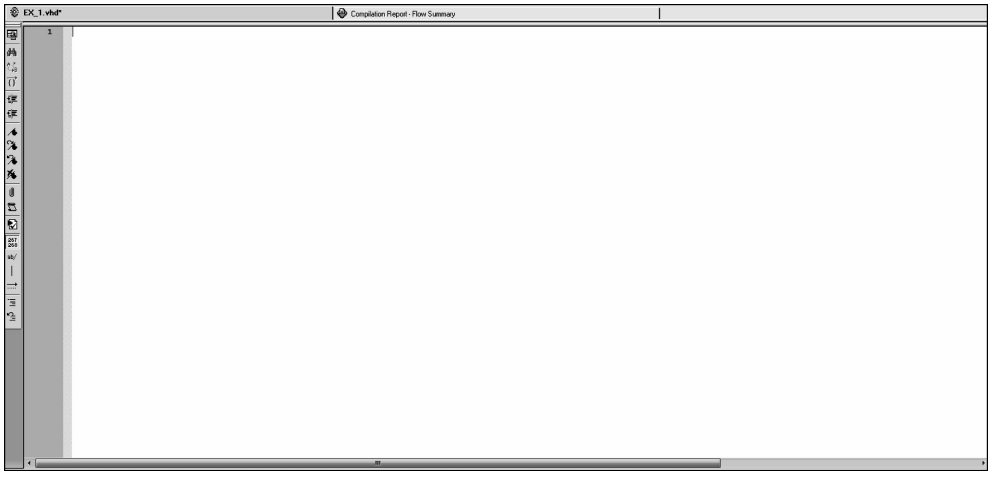

图 5.4 空白文件写入窗口

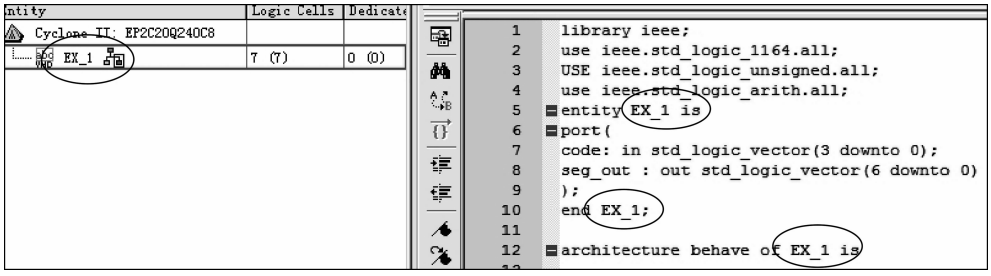

图 5.5 实体名称一致性提示

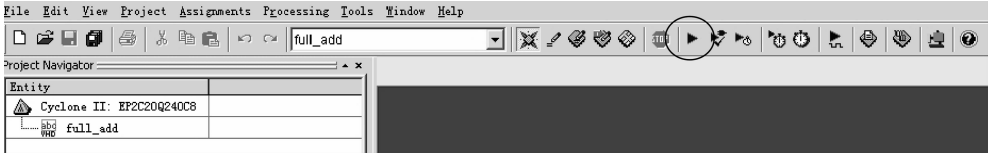

图 5.6 利用编译按钮进行编译

编译成功后会出现如图 5.7 所示的对话框表示编译成功,其中括号内有 3 处警告信息(warings),在此可以忽略。

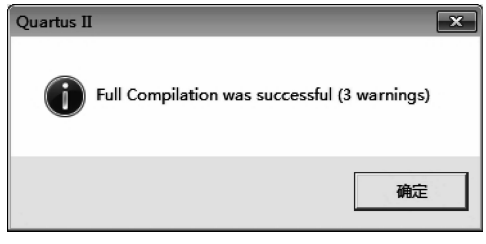

图 5.7 编译成功提示

编译成功只能说明代码没有语法上的错误，不能说明实验的正确性，需要指定输入输出引脚，通过设置输入引脚和输出引脚，观察实验平台的变化验证是否正确显示实验结果。下面就要进行引脚的绑定，在工具栏中选择 Asigments→Pins 命令，会弹出引脚绑定的界面，如图 5.8 所示。

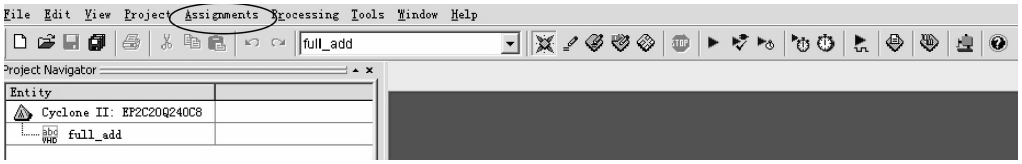

图 5.8　引脚设置菜单

图 5.9 为引脚绑定的工作界面(Quartus 会根据你的代码自动识别工程中所需要的引脚数，并且根据代码中引脚功能的设定自动对引脚进行归类)。此实验中使用了 4 个输入和 7 个输出，Quartus 已经做好归类，其中的 Location 一栏中，需要对输入和输出进行引脚的绑定，通过双击来 Location 列中的某一行选择想要绑定的引脚。

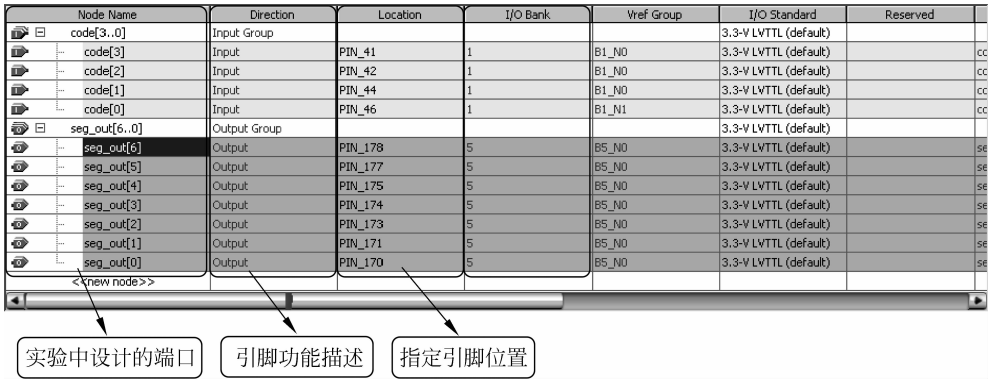

图 5.9　引脚绑定界面

在把代码下载到实验平台做实验之前，可以先进行软件仿真，这样就可以观察输入输出的波形来检查实验结果是否正确(具体的波形仿真可以参照本书 3.3 节的介绍)。

当仿真完成之后，就可以将写好的代码下载到实验平台的 FPGA 中，如果是进行本地实验，可以通过选择工具栏中的 program 按钮进行下载(如图 5.10 所示)。

图 5.10　利用 program 按钮下载

单击 program 按钮后会出现如图 5.11 所示的界面，在其中要选中 Program/Configure，下载方式选择 JTAG，然后单击 Start 按钮，Quartus 就会把代码下载到 FPGA 中，当进度条中显示 100%后，就可以操作和观察实验箱，验证实验是否正确。

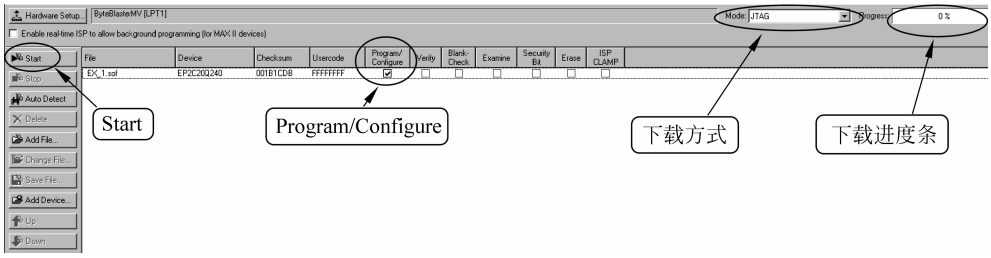

图 5.11 下载设置

3. 远程实验操作

此实验也可以通过远程实验平台来操作，同样是 4 个输入引脚、7 个输出引脚。只不过输入控制变为通过鼠标单击，控制信号可以绑定在 pin178、pin177、pin175、pin174，由于这几个输入引脚是复用引脚，所以要避开同时使用这几个引脚的数码管，这里使用 pin161、pin162、pin164、pin165、pin166、pin167、pin168 控制显示数码管的结果，操作者通过观察实验平台上的数码管变化来验证实验结果是否正确。

实验 2 多路选择器

一、实验要求

- 深入了解远程实验系统及使用方法。
- 掌握多路选择器的原理，用硬件描述语言实现多路选择器的门级设计。
- 学会利用软件仿真和远程实验系统实现对数字电路的逻辑功能进行验证和分析。

二、实验原理

数据选择器是常用的组合逻辑部件之一。它由组合逻辑电路对数字信号进行控制来完成比较复杂的逻辑功能。它有若干个数据输入端，若干个控制输入端和一个输出端。数据选择是指经过选择，把多个通道的数据传送到一个公共数据通道上去。数据选择器的逻辑功能是在控制输入端上加上适当的信号，可以从多个输入数据源中将所需的数据信号选择出来，送到输出端。

图 5.12 是一个 4 选 1 多路选择器原理图，即 abcd 4 个输入通过 $s0$、$s1$ 选择输出其中一路。2 选 1 或者其他多路选择器的设计原理都如此。

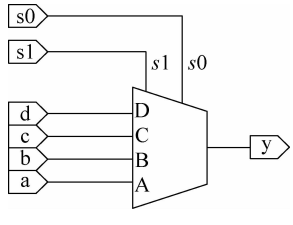

图 5.12 4 选 1 多路选择器

大部分多路选择器会配有一个使能端 s：

当 $s=1$ 时，不论输入如何，均无输出；

当 $s=0$ 时，多路选择器正常工作。

若没有配置以上的使能端，则使能端是直接选择输入信号作为控制信号。

三、实验设计及实现

用 VHDL 设计好的选择器程序经过编译后，在工程文件夹内，可以找到一个以.rbf 后缀结尾的文件，具体操作如下：在实验平台界面中，单击“选择实验”菜单（如图 5.13 所示），会出现实验种类的下拉菜单。

图 5.13 “选择实验”菜单

在“数字逻辑实验”中，选择“多路选择器设计”，会出现如图 5.14 所示的对话框，在工程文件夹中选中.rbf 后缀文件。

图 5.14 “打开”对话框

单击“打开”按钮之后，在实验平台界面的左下端的“设备控制”进度就会变亮，然后选择“设备控制”→“连接设备”命令，如图 5.15 所示，实验视窗左下方有实验操作进度表，当每一个进度前面显示为绿色则该操作成功，否则请重新操作。

图 5.15 “连接设备”命令

连接好设备后,在“设备控制”菜单中选择“FPGA 下载”命令,如图 5.16 所示,系统会把写好的程序自动下载到实验平台的 FPGA 中。

图 5.16 “FPGA 下载”命令

然后在“运行程序”菜单中选择“运行”命令,如图 5.17 所示,系统就会弹出相应的实验界面供实验者操作。

图 5.17 “运行”命令

实验操作进度可以通过观察实验平台左下方的进度列表查看相关操作是否成功。每成功一步，相应的操作项会变为绿色，如图 5.18 所示。

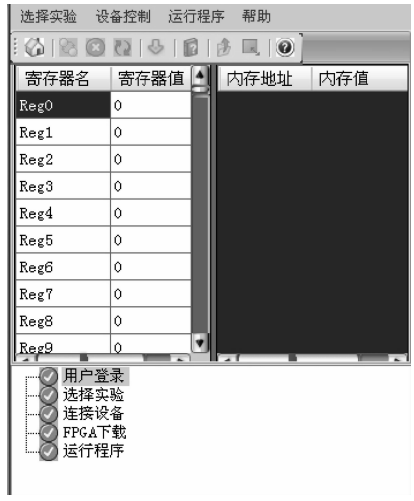

图 5.18　查看实验操作进度

通过软件平台下载后运行，在如图 5.19 所示的实验界面中，左边的数字框有事件监听功能，当单击鼠标时，会使数字在 0、1 间变换，并会将最后选择的结果自动刷新显示。当有鼠标单击操作时，系统首先会从界面左侧重新读入新的数据，并发往实验平台硬件 SRAM 数据总线，经过短暂的延迟之后，系统从硬件 FPGA 地址总线读入选择运算之后的结果，解析后将结果显示在实验界面上。

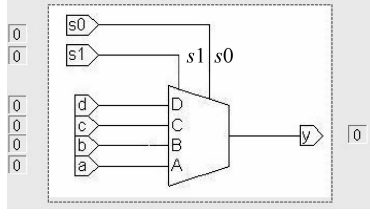

图 5.19　多路选择器实验界面

实验结果解析之后，在右边的输出结果中会根据解析后的数据而显示出结果。输出的结果是由两个使能端确定是哪个输入数据被选择。

在实验中，需要注意的是引脚绑定。与直接在硬件平台上运行程序不同，在软件平台运行程序时，输入的参数会由程序发往数据总线。所以，在引脚配置时，发送的数据配置是以下的配置：

- s0、s1、d、c、b、a 分别对应数据总线的前 6 位。
- 输出结果对应地址总线的第一位。

所以，引脚配置时选择的引脚是由 PIN_178、PIN_177、PIN_175、PIN_174、PIN_173 和 PIN_171 这 6 个数据总线的引脚。具体分配是 s0、s1、d、c、b、a 分别对应 PIN_178、PIN_177、PIN_175、PIN_174、PIN_173、PIN_171。

实验 3　4 位加法器

一、实验要求

- 掌握组合逻辑电路的基本分析和设计方法。
- 理解半加器和全加器的工作原理，用硬件描述语言实现半加器和全加器的门级设计，并使用自己设计的半加器组件构建全加器。
- 学会利用软件仿真和远程实验系统实现对数字电路的逻辑功能进行验证和分析。

二、实验原理

这个实验的目的是为了让学生们掌握组合逻辑电路的基本分析和设计方法，理解半加器和全加器的工作原理，用硬件描述语言实现半加器和全加器的门级设计并掌握利用全加器构成不同字长加法器的各种方法，用硬件描述语言实现指定字长加法器的设计。

半加器(Half Adder)是不考虑来自低位的进位信号，其输入为 1 位的被加数和加数，输出为 2 位：本位的和以及向高一位的进位。在二进制加法的运算只采用半加器是不够的，必须有低一位的进位才能完成正确地运算。考虑低位进位的 1 位二进制加法器称为全加器(Full Adder)，其输入为被加数、加数以及低一位来的进位，输出为本位的和及向高一位的进位。

一位全加器可以由两个半加器及一个或门连接而成，半加器和全加器逻辑结构如图 5.20 和图 5.21 所示。

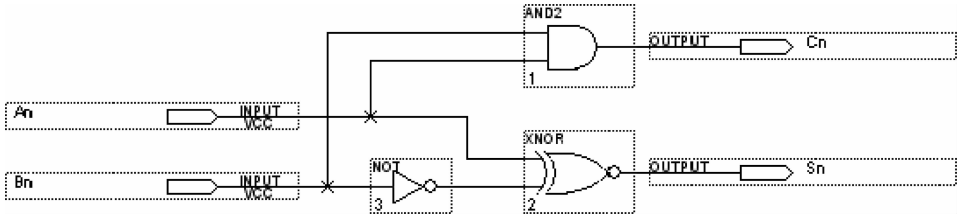

图 5.20　半加器原理图

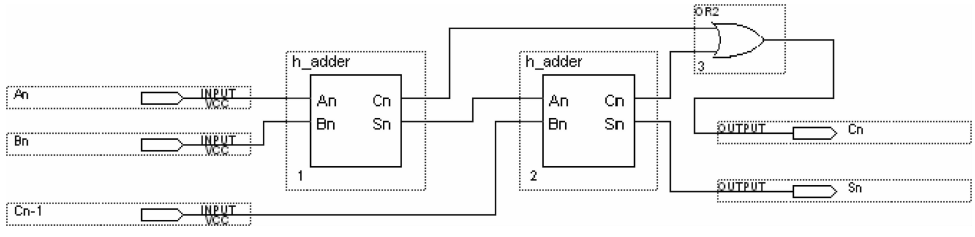

图 5.21　全加器原理图

利用全加器级联可以构成多位二进制加法器，如图 5.22 所示为 4 位二进制加法电路，低一位的进位输出作为高一位的进位输入。这种结构称为逐次进位加法器(Ripple Adder)。

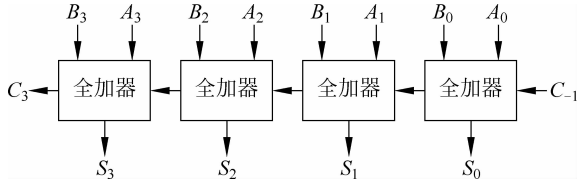

图 5.22 逐次进位加法器

由于逐次进位加法器的进位信号是在各级间逐级传递的，所以高位的输出必须等低位的进位输入稳定后才有效，这就使得逐次进位加法器的延时比较大，速度比较慢。为了提高加法器的运算速度，需要对加法器的结构进行改进。

引入进位传递信号和进位产生信号的概念，有

$$P_n = A_n \oplus B_n$$

$$G_n = A_n B_n$$

利用这两个信号，可以把和信号与进位输出信号表示为

$$S_n = P_n \oplus C_{n-1}$$

$$C_n = P_n C_{n-1} + G_n$$

根据上面给出的进位输出表达式，可得

$$C_0 = G_0 + P_0 C_{-1}$$

$$C_1 = G_1 + P_1 C_0 = G_1 + P_1 G_0 + P_1 P_0 C_{-1}$$

$$C_2 = G_2 + P_2 C_1 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{-1}$$

$$C_3 = G_3 + P_3 C_2 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_{-1}$$

从上面的表达式可以看出，各级的进位输出信号仅取决于其后各级的进位传递信号和进位产生信号，以及最低位的进位输入信号。由于各级的进位传递信号和进位产生信号是同时生成的，所以各级的进位输出信号不再需要等待低一位的进位输入信号，从而大大减小了整个电路的延时，提高了加法器的运算速度。这种方法称为超前进位，超前进位加法器就是利用超前进位形成电路实现快速进位的。图 5.23 给出了利用全加器和超前进位电路组成的 4 位二进制超前进位加法器的结构图。

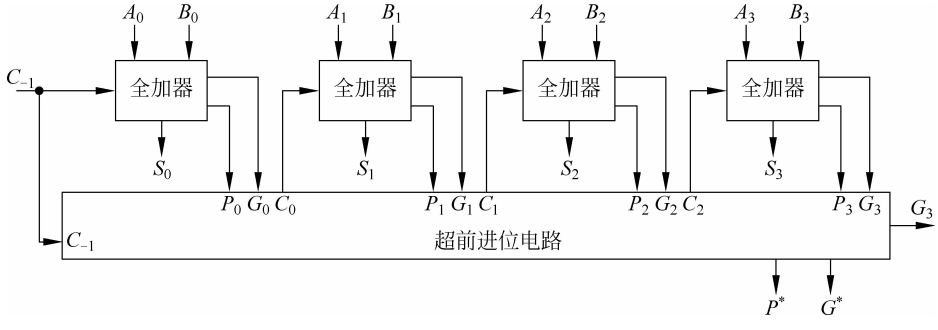

图 5.23 超前进位加法器结构图

超前进位加法器通过超前进位形成电路实现进位信号的产生，其优点是速度快，但是超前进位形成电路需要比较复杂的电路实现，特别是当加法位数较多时，进位形成电路将会十

分复杂。为了实现多位加法电路的快速进位，而又不使电路过于复杂，可以在上述 4 位二进制超前进位加法器的基础上，利用整个加法器的进位传递信号和进位产生信号，再一次采用超前进位形成电路，形成更多位数的超前进位加法电路。

实验中，需要完成逐次进位以及超前进位加法器。这两种 VHDL 程序在编写时会有所不同，不过在运行时由于软件显示的时延和人的感受时延，其差别很难感受到。

三、实验设计与实现

图 5.24 是加法器的实验界面。左边的 A3～A0，B3～B0 分别是两个输入数据的二进制编码，右边 F3～F0 是输出结果，上方的 Cout 是进位的显示。

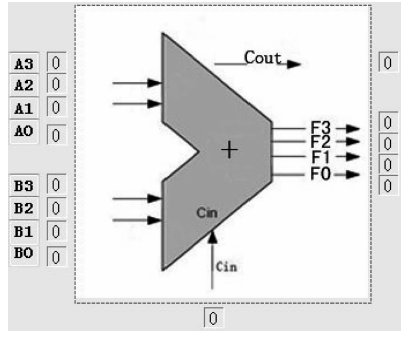

图 5.24　加法器实验界面

在选择实验、下载并运行所设计加法器程序之后，会弹出如图 5.24 所示的实验界面。左边的数字框具有事件监听功能，单击左边数字框，框内的数字会在 0、1 之间变化，并刷新结果。

在引脚配置时，向数据总线传输的数据由低位到高位分别是 Cin、A[3]～A[0]、B[3]～B[0]，故引脚分配时所对应的分别是数据总线的第 1 位到第 8 位，具体对应是：Cin、A[3]～A[0]、B[3]～B[0]分别对应 PIN_178、PIN_177、PIN_175、PIN_174、PIN_173、PIN_171、PIN_170、PIN_168、PIN_167 这几个引脚。

VHDL 程序输出结果时，不需要将其编码后在数码管上显示数字，而只需要将进位 Cout、F[0]～F[3]对应 FPGA 地址总线的第 1 位到第 4 位。

实验 4　一位十进制加法器

一、实验要求

- 掌握组合逻辑电路的基本分析和设计方法。
- 理解一位十进制加法器的工作原理，理解利用 BCD 码实现十进制加法器的原理，用硬件描述语言实现一位十进制加法器的门级设计。
- 学会利用软件仿真和远程实验系统实现对数字电路的逻辑功能进行验证和分析。

二、实验原理

本实验的目的是让大家掌握组合逻辑电路的基本分析和设计方法，理解利用 BCD 码实现十进制加法器的原理，用硬件描述语言实现指定字长十进制加法器的设计。

在实际应用中，经常需要进行十进制的加法运算，可以利用 8421-BCD 码来实现。在对 BCD 码进行加法运算时，可先按二进制进行计算，然后对所得结果进行修正，得到正确的结果。两个一位十进制数 BCD 码相加，考虑可能存在的低位进位，最大和为 19。对于和小于 9 的数，结果无须修正；对于和大于 9 的数，通过先减 10，再加 16 的修正，即加 6 的修正，可以实现十进制的逢 10 进位，具体操作如表 5.2 所示。

表 5.2　十进制的 BCD 编码

十进制数	原始的 BCD 和					修正的 BCD 和				
	C_n	S_3	S_2	S_1	S_0	C_n	S_3	S_2	S_1	S_0
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1
2	0	0	0	1	0	0	0	0	1	0
3	0	0	0	1	1	0	0	0	1	1
4	0	0	1	0	0	0	0	1	0	0
5	0	0	1	0	1	0	0	1	0	1
6	0	0	1	1	0	0	0	1	1	0
7	0	0	1	1	1	0	0	1	1	1
8	0	1	0	0	0	0	1	0	0	0
9	0	1	0	0	1	0	1	0	0	1
10	0	1	0	1	0	1	0	0	0	0
11	0	1	0	1	1	1	0	0	0	1
12	0	1	1	0	0	1	0	0	1	0
13	0	1	1	0	1	1	0	0	1	1
14	0	1	1	1	0	1	0	1	0	0
15	0	1	1	1	1	1	0	1	0	1
16	1	0	0	0	0	1	0	1	1	0
17	1	0	0	0	1	1	0	1	1	1
18	1	0	0	1	0	1	1	0	0	0
19	1	0	0	1	1	1	1	0	0	1

从表 5.2 中可以看出，可以根据 C_n 的值判断结果是否需要进行加 0110 修正。基于上述考虑，得到一位十进制加法电路结构如图 5.25 所示。

三、实验设计及实现

图 5.26 是加法器的实验界面。与 4 位加法器的实验界面类似，左边的 A3～A0、B3～B0 分别是两个输入数据的二进制编码，右边 F3～F0 是输出结果，右边上方的 Cout 是进位的显示。

选择实验并运行编译下载后的程序之后，会弹出如图 5.26 的实验界面。左边的数字框具有鼠标事件监听功能，鼠标单击左边的数字框，框内的数字会在 0、1 之间变化。输入稍加

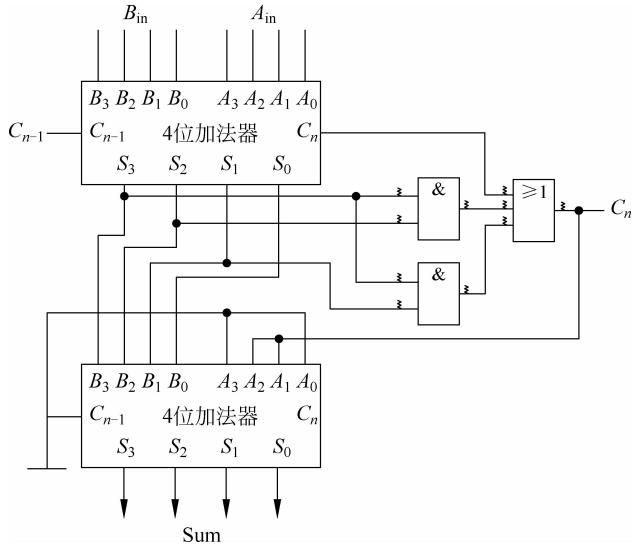

图 5.25 一位十进制加法结构

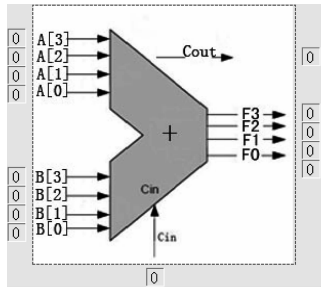

图 5.26 一位十进制加法器实验界面

延迟之后，实验系统会从地址总线读出数据，解析后更新实验界面右边的实验结果。

引脚配置方面也与 4 位加法器类似，A[0]～A[3]、B[0]～B[3]、Cin 分别对应 PIN_178、PIN_177、PIN_175、PIN_174、PIN_173、PIN_171、PIN_170、PIN_168、PIN_167 这几个数据总线的引脚。输出 F[0]～F[3]、Cout 是地址总线的前 5 位：PIN_118、PIN_117、PIN_116、PIN_114、PIN_113。

在实验箱上完成实验时，输出结果是需要进行译码的，将其以数字的形式显示在数码管上。在软件平台上完成实验时，也可以将译码后的结果绑定在如表 5.3 所示的引脚上。

表 5.3 远程实验引脚绑定表

DS2[6]	LED1[0]	FPGA 地址总线第 17 位	PIN_78
DS2[5]	LED2[7]	FPGA 地址总线第 16 位	PIN_79
DS2[4]	LED2[6]	FPGA 地址总线第 15 位	PIN_80
DS2[3]	LED2[5]	FPGA 地址总线第 14 位	PIN_84
DS2[2]	LED2[4]	FPGA 地址总线第 13 位	PIN_86
DS2[1]	LED2[3]	FPGA 地址总线第 12 位	PIN_87
DS2[0]	LED2[2]	FPGA 地址总线第 11 位	PIN_88

将编码后的结果绑定在以上的引脚时，是不会对弹出的实验界面有影响的，只是会在软件平台主界面的 LED 数码管那里的第二组数码管（界面如图 5.27 所示）显示出与硬件平台一样的显示结果，使得实验结果更加直观。

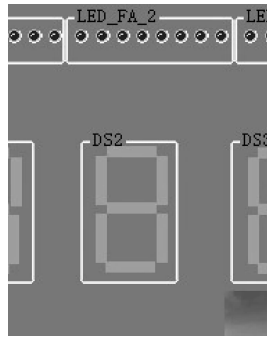

图 5.27　第二组数码管界面

5.2　综合实验

实验 1　计数器的设计

一、实验要求

- 掌握时序逻辑电路的基本分析和设计方法。
- 理解同步时序电路和异步时序电路的区别。
- 掌握计数器电路设计原理，用硬件描述语言实现指定功能的计数器设计。
- 学会利用软件仿真和远程实验系统实现对数字电路的逻辑功能进行验证和分析。

二、实验原理

本次实验的实验目的是：掌握时序逻辑电路的基本分析和设计方法，理解触发器的工作原理，用硬件描述语言实现触发器的门级设计，理解寄存器的工作原理，用硬件描述语言实现不同字长寄存器的设计，掌握利用触发器构成各种计数器的方法，用硬件描述语言实现指定功能计数器设计。

计数器是一种常用的时序电路，它按照规定的方式改变内部各触发器的状态，以记录输入的时钟脉冲的个数。按照规定的计数顺序的不同，计数器可以分为加法计数器、减法计数器和可逆计数器；按照工作方式的不同，又可以分为异步计数器和同步计数器。

加法计数器在计数脉冲依次输入时，相应的二进制数据是依次增加的。用 D 触发器构成的 4 位加法计数器的结构图如图 5.28 所示，每来一个计数脉冲（clk），最低位 Q_D 的状态变化一次，其后各位则在低一位触发器的状态由 1 变为 0 时发生状态变化。这样，利用低一位的反相输出作为高一位的时钟输入，便可以构成加法计数器。

减法计数器的计数规律与加法计数器相反，每来一个计数脉冲，计数器的值减一。利用 D 触发器也可以方便地构成减法计数器，与加法计数器不同的是，减法计数器利用低一位的

输出作为高一位的时钟输入，而加法计数器则是利用低一位的反相输出作为高一位的时钟输入。

上面介绍的加法计数器和减法计数器属于异步计数器，由于计数控制信号是在各级间逐级传递的，这种计数器从时钟脉冲上升沿到达到最后一个触发器翻转到规定的状态，需要较长的延时；计数器位数越多，翻转到稳定状态的时间就越长。为了提高计数器的工作速度，可以采用同步计数器。在同步计数器中，各个触发器使用同一个计数控制时钟，触发器的翻转是在时钟上升沿同步进行的，其翻转稳定时间仅取决于单级触发器的翻转时间，而与计数器的位数无关。

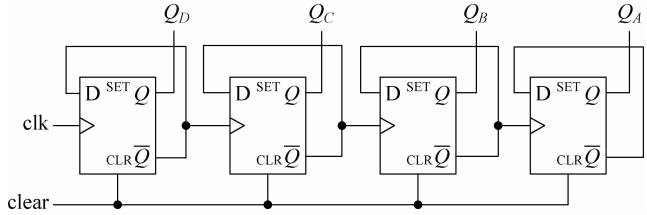

图 5.28　4 位加法计数器结构图

三、实验设计及实现

图 5.29 是计数器的实验界面。在选择实验、编译下载并运行程序之后，会弹出这个实验界面。

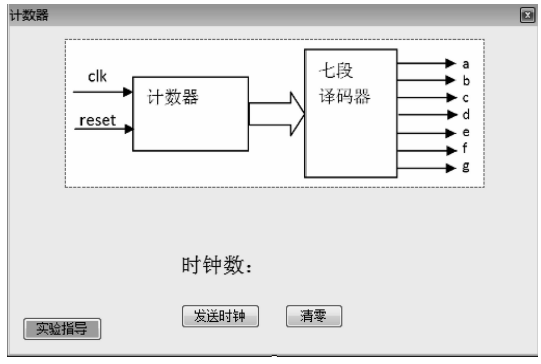

图 5.29　计数器实验界面

这个实验的 VHDL 源程序并没有需要向数据总线发送输入数据，只有两个控制接口：可控的时钟信号以及重置信号。单击“发送时钟”按钮时，系统会运行一个时钟，并将实验运行结果送往已绑定的引脚。本次实验时，需要将输出结果绑定在地址总线的前几位。

经过一个短的延迟之后，系统会从地址总线读出当前计数器的值，解析后并重新编码，显示在时钟数后面，显示部分任意指定数码管即可。

单击“清零”按钮则会重置程序端的计数值，并重置数据总线以及地址总线。

图 5.30 是计数器的引脚绑定示例：其中用到了时钟信号(clk)，使能信号(enable：当使能信号为 1 时计时器开始工作)，复位信号(reset 用于清零)，以及两个数码管用于显示计数结果。

	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard
1	clk	Input	PIN_30	2	B2_N1	3.3-V LVTTL (default)
2	disp_h[6]	Output	PIN_78	8	B8_N0	3.3-V LVTTL (default)
3	disp_h[5]	Output	PIN_79	8	B8_N0	3.3-V LVTTL (default)
4	disp_h[4]	Output	PIN_80	8	B8_N0	3.3-V LVTTL (default)
5	disp_h[3]	Output	PIN_84	8	B8_N0	3.3-V LVTTL (default)
6	disp_h[2]	Output	PIN_86	8	B8_N0	3.3-V LVTTL (default)
7	disp_h[1]	Output	PIN_87	8	B8_N0	3.3-V LVTTL (default)
8	disp_h[0]	Output	PIN_88	8	B8_N0	3.3-V LVTTL (default)
9	disp_l[6]	Output	PIN_105	7	B7_N1	3.3-V LVTTL (default)
10	disp_l[5]	Output	PIN_106	7	B7_N1	3.3-V LVTTL (default)
11	disp_l[4]	Output	PIN_109	7	B7_N0	3.3-V LVTTL (default)
12	disp_l[3]	Output	PIN_110	7	B7_N0	3.3-V LVTTL (default)
13	disp_l[2]	Output	PIN_111	7	B7_N0	3.3-V LVTTL (default)
14	disp_l[1]	Output	PIN_113	7	B7_N0	3.3-V LVTTL (default)
15	disp_l[0]	Output	PIN_114	7	B7_N0	3.3-V LVTTL (default)
16	enable	Input	PIN_178	5	B5_N0	3.3-V LVTTL (default)
17	reset	Input	PIN_127	6	B6_N1	3.3-V LVTTL (default)

图 5.30　计数器引脚绑定示例

实验 2　数字钟

一、实验要求

- 掌握时序逻辑电路的基本分析和设计方法。
- 充分理解同步时序电路和异步时序电路的区别。
- 理解利用计数器实现时钟信号分频的原理，用硬件描述语言实现时钟分频电路设计，设计时间秒表的功能。
- 学会利用软件仿真和远程实验系统实现对数字电路的逻辑功能进行验证和分析。

二、实验原理

本次实验目的是掌握计时电路、分频电路设计方法，掌握有限状态机设计方法，学习如何构造一个复杂的数字系统。

这次实验是一个综合实验，工作量比较大，需要完成的设计比较多，实际的难度不是很高。

数字钟实际上可以看成是一个对标准频率(1Hz)进行计数的计数电路。可以根据上一个实验(实验 1)的实验原理，在计数器实验的基础上，完成本次实验。除了需要设计一个计时电路之外，还需要设计存储器，以及带进位的存储器。

在实验设计中，需要先设计一个分频电路，产生 1Hz 的标准脉冲信号，之后设计一个计时电路，完成时、分、秒的计时，用数码管显示当前时间。

在实验测试时，由于 1Hz 的脉冲信号测试时所需时间太长，故测试时可以将时钟的引脚绑在 24MHz 时钟上，也就是 PIN_212 引脚，这样可以快速验证设计的正确性，通过验证后，可以连接到分频所得的时钟上演示最后实验结果。

三、实验设计及实现

图 5.31 是数字钟实验界面。由于实验需要较多数据位，所以实验中暂时只显示秒的实验。

图 5.31 数字钟实验界面

另外，由于实验平台与服务器端数据交换的延迟，且软件平台也不支持高频率采样，故在数据采样时可能无法做到同步采样。即当以 24 小时制时钟作为检验时钟时，数字钟的显示感觉是跳跃的而不是连续的。因此发送时钟信号时可以先不使用固定频率的时钟信号，而是选用按键时钟，这样每次给出一个时钟信号，数字钟会增加一秒，这样可以方便测试，测试通过后再连接分频出来的 1Hz 时钟验证数字钟的效果。在时钟信号的引脚绑定时，对于时钟源的选择需要注意。

同样需要注意的是数码管的引脚绑定。在硬件平台上直接实验时，可以直接绑定相应的数码管，程序成功运行之后会在实验箱上相应的数码管上显示结果。但若是要在软件平台上完成实验，则需要将输出引脚绑定到地址总线相应的引脚，由程序端读出地址总线的相应数据，并显示在界面上。

实验 3 用状态机实现序列检测

一、实验要求

- 掌握有限状态机设计方法并实现串行序列检测器。
- 学会利用软件仿真和远程实验系统实现对数字电路的逻辑功能进行验证和分析。

二、实验原理

有限状态机及其设计技术是使用数字系统设计中的重要组成部分，是实现高效率、高可靠逻辑控制的重要途径。由于状态机结构模式相对简单，设计方案相对固定，特别是可以定义符号化枚举类型的状态，这一切都为 VHDL 综合器尽可能发挥其强大的优化功能提供了有利条件。本实验的任务是实现一个具有特定功能的状态机，为了方便描述选择简单计算器的控制状态机加以介绍。

状态机用一个 3bit 的寄存器表示，各状态的转换关系如图 5.32 所示，具体描述如下：

reset 后，状态机处于状态 0。

在状态机处于状态 0 时，在时钟上升沿采集输入数据和数据使能，如果数据使能有效且输入数据在 0～9 之间，则状态机跳转到状态 1，同时将输入的数据保存在运算数 A 的寄存器中，否则状态机保持在状态 0。

在状态机处于状态 1 时，在时钟上升沿采集输入数据和数据使能，如果数据使能有效且输入数据为运算符 A 或 B(A 表示加号，B 表示减号)，则状态机跳转到状态 3(状态是可以不连续的数字)，同时将输入的数据保存在运算符的寄存器中，否则状态机保持在状态 1。

在此状态，如果数据使能有效且输入的数据是 0～9 之间的数，则更新运算数 A 的内容。

在状态机处于状态 3 时，在时钟上升沿采集输入数据和数据使能，如果数据使能有效且输入数据在 0～9 之间，则状态机跳转到状态 4 同时将输入的数据保存在运算数 B 的寄存器中，否则状态机保持在状态 3。在此状态，如果数据使能有效且输入的数据是 A 或 B，则更新运算符寄存器的内容。

在状态机处于状态 4 时，在时钟上升沿采集输入数据和数据使能，如果数据使能有效且输入数据是 C(表示＝)，则状态机跳转到状态 5，否则状态机保持在状态 4。在此状态，如果数据使能有效且输入的数据是 0～9 的数，则更新运算数 B 的内容。

在状态机处于状态 5 时，在时钟上升沿采集输入数据和数据使能，如果数据使能有效且输入数据是 D(代替回车，表示重新开始)，则状态机跳转到状态 0，否则状态机保持在状态 5。

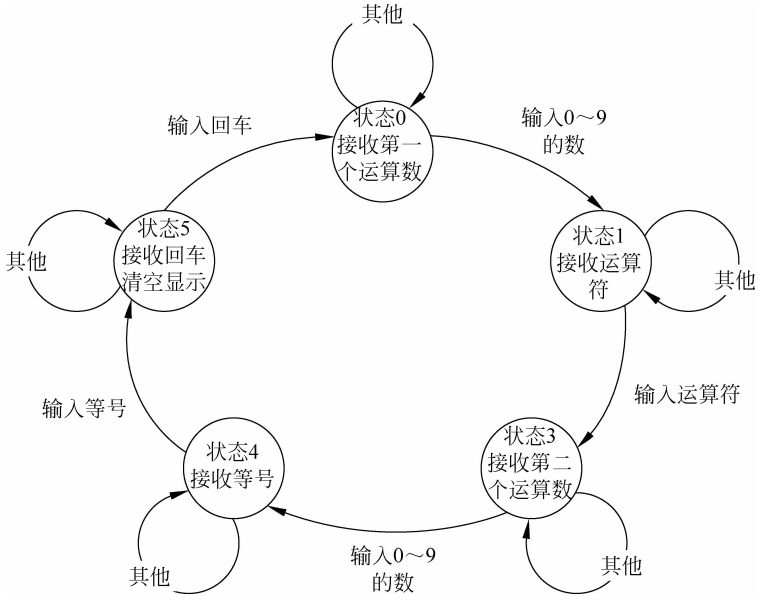

图 5.32　简单计算器状态机

三、实验设计及实现

为了简化设计，本次实验是要求画出 1001 序列检测器状态图，采用状态机的设计方法设计串行序列检测器。利用实验装置串行输入一个 8 位的二进制信号序列。对输入的序列进行检测，判断其是否含有“1001”序列。将检测的结果输出到数码管，若检测到“1001”，则输出 A，否则输出 B。

图 5.33 是这个实验的实验界面。在右侧有一个输入序列，单击输入序列中的数字会改变输入序列的值，以预置一个 8 位二进制数，将这个数作为待检测的数值。左侧是显示检测后当前的状态。单击“运行”按钮可以让程序运行一个时钟，此时预置数将被左移1 位，高位作为状态机输入，最低位移入 0。单击“复位”按钮会重置实验界面以及实验系统的输入数据。

图 5.33 状态机实验界面

程序运行并显示实验界面之后，右侧的输入序列中的数字框具有监听功能，鼠标单击之后会使数字框中的数字在 0、1 之间变化。单击“运行”按钮会发送一个时钟信号到服务器端，使得 VHDL 程序运行一个时钟周期，左侧的状态栏会读出当前地址总线中的数据，解析后会显示实验运行的结果：当前状态为状态 A 或者是状态 B。