# 设计概念

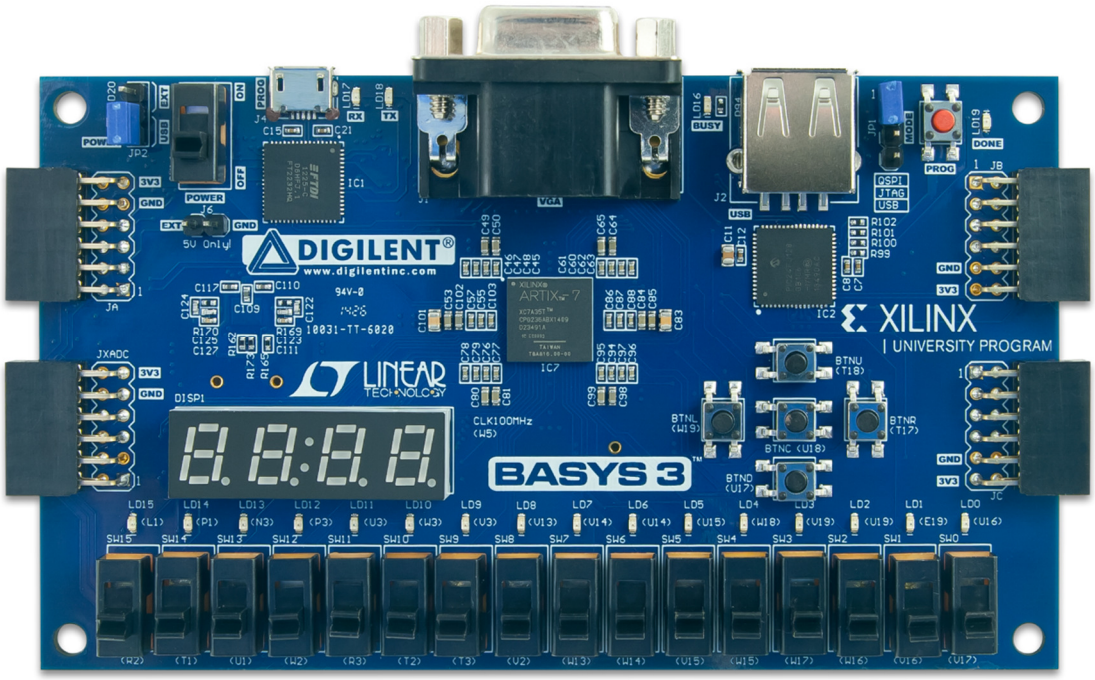
## 简介

Verilog HDL 设计语言支持3种设计风格： 门级，数据流级和行为级。 门级和数据流级设计风格通常用于设计组合逻辑电路，而行为级设计风格既可以用于设计组合逻辑电路又可以设计时序逻辑电路。本次实验通过使用Vivado 2015.1软件工具，以Basys3和Nexys4 DDR开发板为目标板，设计简单的组合逻辑电路来展示3种设计风格的用法。请参考Vivado手册了解如何使用Vivado工具创建工程并验证数字电路。

Basys3开发板特性如下: (译者注：开发板各批次参数不同，仅供参考)

- 1,800 Kbits 的快速随机存储器
- 33,280 逻辑单元，5200个切片 (每片包含4个6-input LUT和8个触发器)
- 5个时钟管理，每个包含一个锁相环
- 90 DSP 切片
- 内部时钟速率超过 450MHz
- 片内模数转换器(XADC)
- 16 个拨码开关
- 16 个用户LED
- 5 个用户按键
- 4位7段数码管
- 3个 Pmod 连接器
- XADC信号Pmod
- 12位 VGA 输出
- USB-UART 桥接
- 串行闪存
- Digilent USB-JTAG 端口，支持FPGA编程和通信
- 鼠标、键盘和记忆棒的USB HID 主机

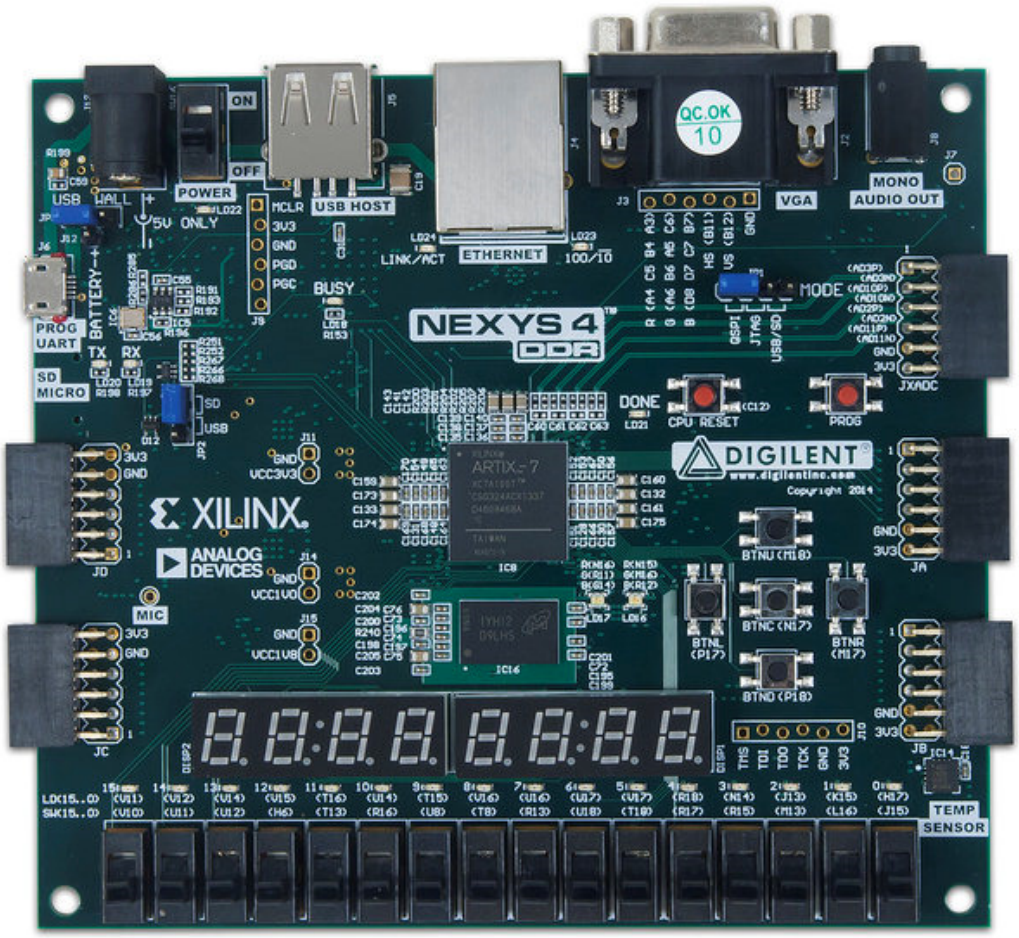
Basys 3开发板如下图所示。



Nexys4 DDR 特性如下: (译者注：开发板各批次参数不同，仅供参考)

- 128 MiB DDR 2 SDRAM
- 16Mbytes SPI (quad 模式) PCM 非易失型存储器
- 16Mbytes 并行 PCM 非易失型存储器
- 10/100 以太网PHY
- USB-UART 和 USB-HID 端口(用于鼠标和键盘)
- 8-bit VGA 端口
- 100MHz CMOS 振荡器
- 72个I/O连接到扩展连接器
- GPIO 包括 8个LED， 5个按键开关，8个拨码开关和2个4位7段数码管

Nexys4 DDR 开发板如下图所示




## 目标

完成本次实验后，你将有能力:

- 使用门级、数据流级和行为级3种风格设计一位和多位组合逻辑电路
- 设计模型读取拨码开关和按键开关并输出到LED和7段数码管
- 仿真并理解设计的输出
- 创建分层的设计
- 综合，实现并生成比特流文件
- 下载比特流文件到开发板，并验证功能

# 门级建模 Part 1

Verilog HDL 支持内建的原始的门级设计。门级支持包括多输入、多输出、三态和拉态。 多输入门支持包括： **and**, **nand**, **or**, **nor**, **xor**, 和**xnor**，它们的输入为2个及以上，输出只有1个。多输出门支持包括 **buf** 和 **not** ，它们的输出为2个及以上，输入只有1个。 Verilog HDL语言还支持三态门： **bufif0**, **bufif1**, **notif0**, 和**notif1**。这些三态门有一个输入，一个控制信号和一个输出。拉门支持包括 **pullup**和 **pulldown**，只有一个输出（没有输入）。

这些门的零延迟的基本语法如下：

**and** | **nand** | **or** | **nor** | **xor** | **xnor** [instance name] (out, in1, …, inN); // [] is optional and | is selection
**buf** | **not** [instance name] (out1, out2, …, out2, input);
**bufif0** | **bufif1** | **notif0** | **notif1** [instance name] (outputA, inputB, controlC);
**pullup** | **pulldown** [instance name] (output A);

你也可以在同一语句中，用逗号分隔，创建多个相同类型门的实例，比如：

**and** [inst1] (out11, in11, in12), [inst2] (out21, in21, in22, in23), [inst3] (out31, in31, in32, in33);

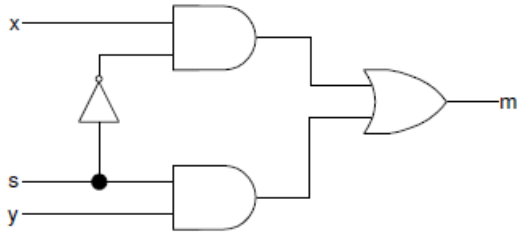
Verilog HDL语言也允许在实例化门电路时加入延迟。 加入的延迟来自输入或输出。这些延迟可以表达为上升、下降或关断延迟；在一个实例中可以使用1、2或3种延迟。关断延迟可以用于输出能被关掉的门(如 notif1).

比如，

```
and #5 A1(Out, in1, in2);       // the rise and fall delays are 5 units
and #(2,5) A2(out2, in1, in2);        // the rise delay is 2  unit and the fall delay is 5 units
notif1 #(2, 5, 4) A3(out3, in2, ctrl1); //the rise delay is 2, the fall delay is 5, and the turn-off delay is 4 unit
```

门级建模在设计简单的组合电路时非常有用，比如设计一个多路选择器。多路选择器是包含多个输入和一个输出的简单电路。在本章，你将设计一个简单的2-to-1多路选择器并将其扩展到多位。

## 1-1. 使用门级建模风格设计2-to-1多路选择器




**1-1-1.** 打开 Vivado 2015.1 并创建空白工程，取名为 *lab1_1_1* (参考 Vivado2015.1 手册 Step 1)。

**1-1-2.** 使用门级建模风格创建Verilog module包含3个输入(x,y,s)和1个输出(m) (参考Vivado2015.1手册 Step 1).

提示：单击在New Project窗口，Add Source上的**绿色加号**按钮。然后单击Create File。修改文件名为lab1_1_1，单击**OK**。确认目标语言和仿真语言都设置为Verilog。单击两次Next。

**1-1-3.** 将适合开发板的XDC文件添加到工程。

提示：单击在New Project窗口Add Constraints上的绿色加号按钮。单击AddFile.选择Basys3_Master.xdc (Basys3)或Nexys4DDR_Master.xdc (Nexys4 DDR)。点击Next。

**1-1-4.** 在New Project窗口选择**xc7a35tcpg236-1**(Basys3)或**xc7a100tcsg324-1**(Nexys4 DDR)。单击Next。单击Finish。

**1-1-5.** 一个定义*Module*的窗口会出现， 通过单击Port Name并输入变量名，创建3个输入 (x, y, s) 和1个输出(m) 。 通过单击下拉列表选择正确的方向修改 **Direction**。单击**OK.**

**1-1-6.** 打开**lab1_1_1.v** 文件编辑其中内容。 在分号 (;)后添加上文电路的结构逻辑。
选择**File > Save File**或 CRTL-S保存。

**1-1-7.** 单击RTL Analysis上的 *Elaborated Design*选项卡。

**1-1-8.** 单击 **Schematic** 查看门级建模的设计。

**1-1-9.** 编辑XDC文件。去注释并将 ***SW0*** 和 ***SW1***赋给*x* 和 *y*, ***SW7*** 给 *s*, ***LED0***给*m*。
保存XDC文件。

**1-1-10.** 综合你的设计 (参考Vivado 2015.1手册 Step 3)。

**1-1-11.** 实现你的设计 (参考Vivado 2015.1手册 Step 4)。

**1-1-12.** 生成比特流文件，将其下载到Basys3或Nexys4 DDR开发板，并验证功能 (参考Vivado 2015.1 手册 Step 6 有关创建和下载比特流部分)。

## 1-2. 使用门级建模风格设计一个2位宽的2-to-1多路选择器。

**1-2-1.** 打开Vivado 2015.1并创建空白工程取名为 *lab1_1_2*。

**1-2-2.** 使用门级建模风格，创建一个 Verilog module并增加2个2-bit 输入 (*x[1:0]*, *y[1:0]*)，1个1bit 选择信号输入(*s*)和一个2-bit输出(*m[1:0]*)。

**1-2-3.** 添加XDC文件到工程。 编辑XDC文件，将***SW0*** 和 ***SW1***赋给*x[1:0]*, ***SW2*** 和 ***SW3***赋给*y[1:0]*， ***SW7*** 赋给 *s*， ***LED0***和 ***LED1*** 赋给 *m[1:0]*.

**1-2-4.** 综合你的设计。

**1-2-5.** 实现你的设计。

**1-2-6.** 生成比特流文件，将其下载到Basys3或Nexys4 DDR开发板，并验证功能。

# 数据流级建模 Part 2

数据流级建模风格主要用于描述组合逻辑电路。一种基本的手法就是使用持续赋值(continuous assignment)。 在持续赋值中，一个值被指派到一种叫做线网(**net**)的数据类型。
持续赋值的语法为：

```
assign [delay] LHS_net = RHS_expression;
```

其中LHS_net是1bit或多bit的目标线网，而 RHS_expression是一个包含各种运算符(operator)的表达式(expression) 。该语句在任何时候都对源操作数值的任何更改进行运算，并将结果经过延迟单元后赋值给目标线网。在Part 1 中的门级建模风格的例子可以用数据流级建模风格的持续赋值表达。比如：

```
assign out1 = in1 & in2; // perform and function on in1 and in2 and assign the result to out1
assign out2 = not in1;
assign #2 z[0] = ~(ABAR & BBAR & EN); // perform the desired function and assign the result after 2 units
```

持续赋值语句中的目标可以是下面的一种:
1. 标量线网scalar net (比如上面第1和2个例子)
2. 向量线网Vector net
3. 向量线网的常数位选定Constant bit-select of a vector (比如上面第3个例子)
4. 向量线网的常数部分选定Constant part-select of a vector
5. 以上任意的拼接

我们再举一些例子，其中用到了标量和向量线网：

```
wire COUNT, CIN;   // scalar net declaration
wire [3:0] SUM, A, B;    // vector nets declaration
assign {COUT,SUM} = A + B + CIN; // A and B vectors are added with CIN and the result is
                                 // assigned to a concatenated vector of a scalar and vector nets
```

需要注意的是，多个持续赋值不能使用同一个目标线网。

## 2-1. 使用数据流级建模风格，将拨码开关的输入连接到LED上输出。

**2-1-1.** 打开Vivado 2015.1创建空白工程，命名为lab1_2_1。

**2-1-2.** 创建Verilog module，使用数据流级建模风格，添加8输入(x_in)和8输出(y_out)。不对拨码开关的输入做任何处理，直接使用向量赋值。

**2-1-3.** 添加XDC文件到工程。编辑添加的XDC文件，将***SW7***-***SW0***赋值给*x_in*，将***LED7***-***LED0***给*y_out*.

**2-1-4.** 综合你的设计。

**2-1-5.** 实现你的设计。

**2-1-6.** 生成比特流文件，将其下载到Basys3或Nexys4 DDR开发板，并验证功能。

## 2-2. 使用数据流建模风格设计一个2-bit宽的2-to-1多路选择器，包含3 ns线网的延迟。

**2-2-1.** 打开Vivado 2015.1创建一个空白工程，命名为lab1_2_2。

**2-2-2.** Create and add the Verilog module with two 2-bit inputs (*x[1:0], y[1:0]*), a one bit select input (*s*), and two-bit output (*m[1:0]*) using dataflow modeling. Each assignment statement should have 3 units delay.

As an example, a one-bit 2-to-1 multiplexer can be described as follows:

```
assign #3 m = (~s & x) | (s & y); // 3 units delay
```

**2-2-3.** 添加XDC文件到工程。编辑添加的XDC文件，将***SW0***和 ***SW1***赋给*x[1:0]*, ***SW2***和 ***SW3***给 *y[1:0]*，***SW7***给 *s*， ***LED0***和 ***LED1*** 给 *m[1:0]*。

**2-2-4.** 添加提供的测试文件(mux_2bit_2_to_1_dataflow_tb.v)到工程。

将文件名中的mux_2bit_2_to_1_dataflow改为lab1_2_2，并保存文件。

**2-2-5.** 对你的设计仿真(行为级仿真behavioral simulation) 100 ns，并分析输出。

**2-2-6.** 综合你的设计。

**2-2-7.** 实现你的设计。

**2-2-8.** 生成比特流文件，将其下载到Basys3或Nexys4 DDR开发板，并验证功能。

# 行为级建模 Part 3

行为级建模通常用于描述复杂的电路。行为级建模主要用于设计时序逻辑电路，但也可以用于设计纯组合逻辑电路。一个电路的行为级建模（语句）如下：
**initial** Statements
**always** Statements

一个模块可以包含任意数量的**initial**和**always**语句，并且可以在其中包含一个或多个过程语句。这些**initial**和**always**语句会同时执行（换句话说，它们用于描述并行的过程，即它们在模块中出现的顺序没有关系），而过程语句是按序执行的（换句话说，它们出现的顺序有影响）。

initial和always语句都在time=0时刻执行，在其余时间只有always语句执行。语法如下：

```
initial [timing_control] procedural_statements;
always [timing_control] procedural_statements;
```

其中的过程语句procedural_statement是下面之一：
过程赋值procedural assignment
条件语句conditional_statement
案例语句case_statement
循环语句loop_statement
等待语句wait_statement

**initial**语句是不可综合的（**non-synthesizable**）通常用在测试中。**always**语句是可综合的**(synthesizable**）并且最终产生的电路可以是组合的也可以是时序的。为了生成组合逻辑电路，**always**块：**(i)**不能是对边沿敏感的 **(ii)**条件语句的每一个分支都需要定义好输出 **(iii)case**语句中的每个案例（**case**）需要定义所有输出且必须有一个默认情况（**default case**）。有关这个话题的更详细讨论在**Lab 7**中涉及。语句的目标**(LHS)**须为寄存器（**reg**）类型**;** 可以是标量或向量。举个例子：

```
reg m; // scalar reg type
reg [7:0] switches; // vector reg type
```

下面是一个2-to-1多路选择器模型的例子。注意在这个例子中的begin和end是多余的，加入它们是为了更好的可读性。

```
always @ (x or y or s)
begin
      if(s==0)
            m=y;
      else
            m=x;
end
```

## 3-1. 用行为级建模的风格设计一个2-to-1多路选择器

**3-1-1.** 打开Vivado 2015.1，创建空白工程命名为 *lab1_3_1*。

**3-1-2.** 创建一个Verilog module，按行为级建模的风格在其中加入3个输入(*x*, *y*, *s*) 和1个输出(*m*) 。使用上面的示例代码。

要声明一个寄存器(reg)，需要在 input/output 后面，端口名前面放**reg**。

**3-1-3.** 添加XDC文件到工程。 编辑添加的XDC文件，将***SW0***和***SW1***赋给 *x*和*y*， ***SW7*** 给 *s*，***LED0*** 给*m*。

**3-1-4.** 综合你的设计。

**3-1-5.** 实现你的设计。

**3-1-6.** 生成比特流文件，将其下载到Basys3或Nexys4 DDR开发板，并验证功能。

## 3-2. 使用数据流建模风格设计一个2-bit宽的2-to-1多路选择器

**3-2-1.** 打开 Vivado 2015.1 ，创建一个空白工程命名为 *lab1_3_2*。

**3-2-2.** 创建一个Verilog module，按行为级建模风格，添加2-bit输入(*x[1:0], y[1:0]*)，1-bit选择信号输入(*s*)，和2-bit输出(*m[1:0]*) 。

**3-2-3.** 添加XDC文件到工程。编辑添加的XDC文件，将 ***SW0*** 和 ***SW1***赋给 *x[1:0]*, ***SW2*** 和 ***SW3*** 给 *y[1:0]*,，***SW7***给 *s*， ***LED0***和 ***LED1***给 *m[1:0]*。

**3-2-4.** 综合你的设计。

**3-2-5.** 实现你的设计。

**3-2-6.** 生成比特流文件，将其下载到Basys3或Nexys4 DDR开发板，并验证功能。

# 混合设计风格建模 **Part 4**

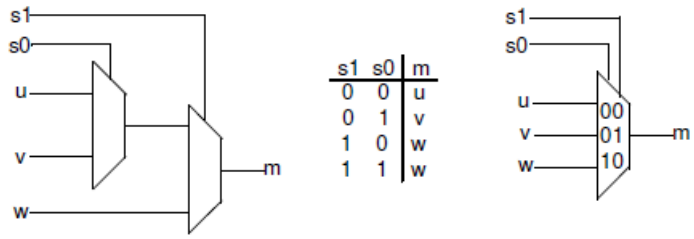
复杂的系统可以使用混合设计风格的Verilog HDL描述。这样的设计风格支持分层的描述。描述一个设计可以使用以下各项：
内建的门级描述 (见Part 1)
数据流级建模(见Part 2)
行为级建模(见Part 3)
模块实例化
以上各项的组合。

各种对像通过线网(类型为***wire***)完成互联。 线网可以是标量和向量。举个例子：

```
wire y; // scalar net
wire [3:0] sum; // vector net
```

当线网的大小缺省时，假设其为标量类型。

作为混合风格建模的例子，下面的电路图展示了如何用多个2-to-1多路选择器的实例来构建一个3-to-1多路选择器。下图还展示了符号和真值表。




在上图中 u, v, w 是输入数据，S0, S1 是选择信号，输出是m。它使用了两个2-to-1多路选择器。

## 4-1. 用多个2-to-1多路选择器来设计一个3-to-1多路选择器。

**4-1-1.** 打开Vivado 2015.1，创建一个空白工程命名为 *lab1_4_1*。

**4-1-2.** 创使用之前的2-to-1多路选择器，创建一个顶层（top-level）的Verilog module，包含3个输入 (*u, y, w*)，2个选择信号输入 (*s0, s1*)和一个1-bit输出 (*m*)。你可以使用任意的风格设计的 2-to-1 多路选择器 (1-1， 2-1，或 3-1)。按上面电路图将它们连接起来。 Verify确认带有 (u, y, w)的.v 文件为顶层（top）文件。要设置成顶层文件，右击 **Set as Top**。

**4-1-3.** 添加之前用过的2-to-1多路选择器的文件到工程。

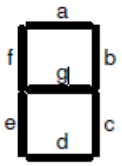
**4-1-4.** 添加XDC文件到工程。编辑添加的XDC文件，将 ***SW0, SW1***和***SW2*** 分别赋给 *u, y*和 *w*， ***SW3***给 *s0*， ***SW4***给 *s1*，***LED0***给 *m*。

**4-1-5.** 综合并实现你的设计。

**4-1-6.** 生成比特流文件，将其下载到Basys3或Nexys4 DDR开发板，并验证功能。

## 4-2. 设计一个BCD码到7段数码管的译码器。

7段数码管显示器包含7段，编号从a到g，可以用来显示一个字符。根据不同的输入类型，可能需要做类型转换。如果需要显示4bit的BCD码，则需要一个BCD码到7段数码管的译码器。下表展示了你要显示一个数字的bit图样。
(需要注意的是，你如果需要点亮某一段，则要输出逻辑0到这一段，并且这个显示器的阳极需要用板上的逻辑0驱动)。

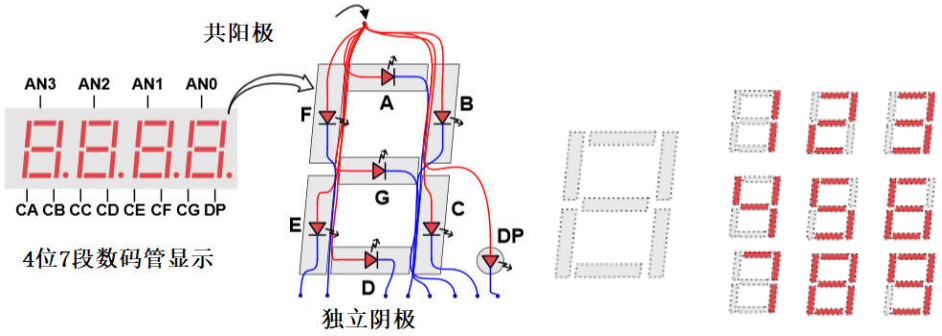



.

| 输入 | a | b | c | d | e | f | g |
|---|---|---|---|---|---|---|---|
| 0000 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0001 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0010 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0011 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0100 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0101 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0110 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0111 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1000 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1001 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1010 到 1111 | X | X | X | X | X | X | x |

x表示无所谓。

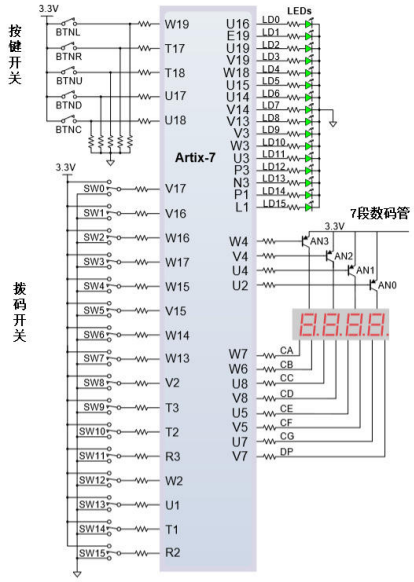
**Basys3** 包含1个4位共阳极**7**段数码管LED显示模块。这个模块中4位的每1位都由7段组成，每一段都嵌入了一颗LED，顺序如下图。 每段的LED可以被单独点亮。因而通过点亮特定的段，熄灭其他段，全部128种图样都可以在1位上实现。在这128种图样中，10种数字的图样最有用。




[参考 – Basys3 参考手册]

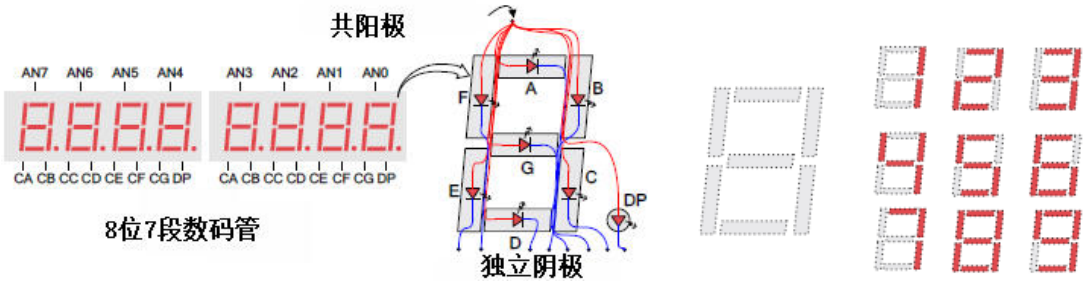
组成每一位的7颗LED的阳极被连接在一起，称为“共阳极”的电路节点，但LED的阴极是独立的。共阳极节点的信号可以被获取作为4位显示器的使能信号。

不同位，相同段的阴极信号分别接到7个电路节点，从CA 到 CG (比如上图中4位的 “D” 笔画的阴极被连接到一起，连接到 “CD”这个电路节点)。这7个阴极节点信号可以被获取作为4位显示器的输入信号。 这种信号连接的方案支持了多路复用的显示，虽然每个笔画的阴极信号是4位共用的，但它们只能点亮阳极信号生效的位上的笔画。



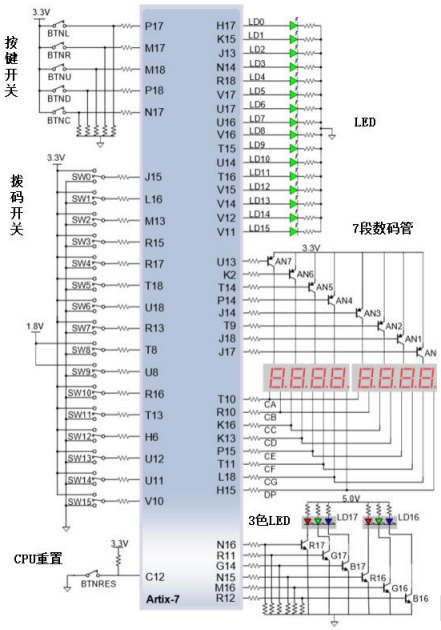

[参考 – Basys3 参考手册]

**Nexys4 DDR**开发板包含两个4位7段数码管LED显示器。每个模块中4位的每1位都由7段组成，每一段都嵌入了一颗LED，顺序如下图。每段的LED可以被单独点亮。因而通过点亮特定的段，熄灭其他段，全部128种图样都可以在1位上实现。在这128种图样中，10种数字的图样最有用。




[参考 – Nexys4 DDR 参考手册]

组成每一位的7颗LED的阳极被连接在一起，称为“共阳极”的电路节点，但LED的阴极是独立的。共阳极节点的信号可以被获取作为4位显示器的使能信号。不同位，相同段的阴极信号分别接到7个电路节点，从CA到CG（比如上图中4位的 “D” 笔画的阴极被连接到一起，连接到 “CD”这个电路节点)。这7个阴极节点信号可以被获取作为4位显示器的输入信号。这种信号连接的方案支持了多路复用的显示，虽然每个笔画的阴极信号是4位共用的，但它们只能点亮阳极信号生效的位上的笔画。




[参考 – Nexys4 DDR 参考手册]

扫描显示控制器电路可以用来控制这个显示器显示一个4位数。这个电路根据阴极图样，以高于人眼可分辨地频率，对每一位的阳极信号进行重复地、持续地、接连地驱动。如果它的更新或“刷新”率降到大约45 Hz时，大多数人能看出显示在闪烁。从Lab 8（体系结构向导和知识产权目录）开始你将设计和使用一个扫描电路。

**4-2-1.** 打开Vivado 2015.1，创建一个空白工程命名为 *lab1_4_2*。

**4-2-2.** 创建一个顶层Verilog module，命名为bcdto7segment_dataflow，按数据流建模风格添加4-bit data 输入(*x[3:0]*),阳极使能输出信号 (*an[3:0]*)， 7-bit 输出 (*seg[6:0]*) 。
(提示：你必须在纸上推导7个段的七个表达式。). 对 *an[3:0]* 输出合适的信号使得只有最右边的一位显示。

**4-2-3.** 添加对应开发板的XDC文件到工程。编辑XDC 文件，将 ***SW3-SW0***赋给*x[3:0]*，**CA, CB, CC, CD, CE, CF, CG**给 *seg[0]*到 *seg[6]*， ***J17, J18, T9,J14, P14, T14, K2, U13***给 an7, an6, an5, an4,*an3, an2, an1, an0* (Nexys4 DDR开发板) 或者***U2,U4,V4,W4***给an3, an2, an1, an0(Basys3开发板)。

**4-2-4.** 综合你的设计。

**4-2-5.** 实现你的设计。

**4-2-6.** 生成比特流文件，将其下载到Basys3或Nexys4 DDR开发板，并验证功能。

## 总结

本次实验中，你创建了多个Vivado 2015.1工程设计了各种电路模型。你实现了设计并在硬件和仿真环境中验证了功能正确性。 你学会了三种建模风格。门级和数据流级建模主要用于组合电路，而行为级建模支持组合和时序电路。本实验中你使用了行为级建模风格完成了组合电路的设计。在后面几个实验中，你会使用数据流级建模设计各种组合电路，从 Lab 7开始，你将使用行为级建模设计时序电路。