**计数器, 计时器 和 实时时钟**

**介绍**

在上一个实验中，您了解了如何使用体系结构向导生成所需的时钟频率，以及如何使用IP目录生成包括计数器在内的各种内核。 这两个功能电路是用于创建定时器（timer）和实时时钟（real-time clock）的基本电路。 在本实验中，您将生成多种计数器，计时器和实时时钟。 请参阅Vivado教程，了解如何使用Vivado工具创建项目和验证数字电路。

**目标**

完成本实验后，您将能够：

•定义参数化模型

•使用行为建模计数器

•使用IP目录设计计数器

•比较和对比使用行为建模和IP目录开发的计数器

•使用内核和用HDL建模的附加电路设计定时器电路

•创建实时时钟

**参数和defparam语句（定义参数声明语句）**

Verilog HDL语言支持模型参数化，即以HDL编写模型，并通过传递不同的参数值（通常是宽度和延迟）重复使用相同的模型。 使用参数构造来定义参数化模型。 有两种方法可以更改值：（i）在模块实例化期间（如下例所示），以及（ii）使用defparam语句。 以下是将宽度定义为参数的示例，默认值为4。

**parameter** WIDTH = 4;

必须先定义参数（WIDTH），然后才能在模型中使用它。 以下是Verilog-2001标准中的还原“或”功能数据流模型的示例。

module reduction\_or #(**parameter** WIDTH = 4)(input ain[WIDTH-1:0], output result);

assign result = |ain;

endmodule

module test\_reduction\_or\_tb;

reg [3:0] ain1;

reg [6:0] ain2;

reg [9:0] ain3;

wire result1, result2, result3;

reduction\_or U1 (ain1, result1); // use default value of 4

reduction\_or #(7) U2 (ain2, result2); // use value of 7

reduction\_or #(10) U3 (in3, result3); // use value of 10

endmodule

请注意，参数WIDTH在使用之前已定义。 在多个参数定义的情况下，实例化期间列出的值映射到定义参数的顺序。 在Verilog-2001标准中，在多个参数的情况下，必须定义所有参数值，即使某些值是默认值。

例如

module FA;

parameter AND\_DELAY=2, XOR\_DELAY=3;

… //gate-level model which uses AND\_DELAY and XOR\_DEALY in the gate

//instantiations

endmodule

module FA2BIT( ports listing);

FA #(4,5) f1(ports connections); // AND\_DELAY will be 4 and XOR\_DELAY will be 5

FA #(1,2) f2(ports conenctions); // AND\_DELAY will be 1 and XOR\_DELAY will be 2

endmodule

defparam语句还可用于通过引用模块实例化来更改更高级别模块的参数值。 例如，

module TOP;

defparam fa2.f1.AND\_DELAY=4, fa2.f1.XOR\_DELAY=5, fa2.f2.AND\_DELAY=1,

fa2.f2.XOR\_DELAY=2;

FA2BIT fa2(ports connections);

endmodule

参数定义可以在使用它的同一文件中，也可以在单独的文件中（如C编程语言中的头文件），头文件可以使用`include伪指令包含在模型中。 例如，

让我们在名为header.v的文件中定义以下参数

parameter WIDTH = 4;

然后，要在定义了reduction\_or和test\_reduction\_or\_tb模块的文件中使用它，请在文件开头使用以下行。

`include header.v // you can add a full path to the file if it is located in some other directory path

1-1。 设计一个类似于实验2第4-1部分设计但使用门级建模的进位超前加法器。 使用参数语句为全加器电路中使用的每种门定义2个单位延迟。 在创建分层模型时，对逆变器使用1个单位延迟，对和和或门使用3个单位延迟，对于xor门使用4个单位延迟。 使用模块实例参数值赋值语句方法。 开发测试平台以验证功能并查看通过层次结构传播的延迟。 不要实现设计。

1-2。 使用defparam语句修改1-1的进位超前加法器，更改测试平台的延迟值。 执行行为建模仿真并观察通过层次结构传播的延迟。 不要实现设计。

**计数器和使用的资源**

计数器是用于测量事件的基本电路。它们可以被分类为简单的自由运动二进制计数，向上/向下计数，模k计数，约翰逊计数，灰码计数和特殊序列计数。在给定的设计或一组设计中，设计者多次使用不同宽度的计数器。第1部分中介绍的参数和defparam语句用于获取不同的宽度。

计数器通常使用加法器和减法器电路，可以使用LUT和/或FF或DSP48片在Xilinx FPGA中实现。您可以在Verilog代码中使用名为USE\_DSP48且值为“yes”或“no”的Synthesis属性来控制要使用的资源类型。 USE\_DSP48指示综合工具如何处理合成算术结构。默认情况下，mults，mult-add，mult-sub，mult-accumulate类型结构进入DSP48模块。加法器，减法器和累加器也可以进入这些块，但默认情况下，使用结构而不是DSP48模块实现。 USE\_DSP48属性会覆盖默认行为并将这些结构强制转换为DSP48单元。该属性可以放在信号和模块的RTL中。

例如，

(\* use\_dsp48 = "yes" \*) // will force the use of DSP48 even on adders

module updown\_ctr\_behavior(

input clk,

...

Endmodule

(\* use\_dsp48 = "no" \*) // will force not to use the DSP48 even on mults

module matrixmul\_behavior(

input clk,

...

endmodule

2-1。 使用行为建模设计一个8位向上/向下计数器。 您的模型应将COUNT\_SIZE定义为参数并在模型中使用它。 计数器将使用板载100 MHz时钟源。 使用时钟向导生成5 MHz时钟，通过时钟分频器进一步分频，以生成周期为1秒的信号。 定义合成属性以不使用DSP48单元。 使用BTNU按钮作为电路复位，SW0作为使能，SW1作为Up / Dn（1 = Up，0 = Dn），LED7~LED0作为计数器输出。 完成设计流程，生成比特流，并将其下载到Basys3或Nexys4 DDR板。 验证功能。 查看“项目摘要”选项卡后，请填写以下信息。

1. Number of BUFG/BUFGCTRL \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Number of Slice LUTs used: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Number of FF used: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Number of DSP48E1 slices used: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Number of IOs used: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2-2。 使用2-1的8位向上/向下计数器设计。 设置合成属性以强制使用DSP48单元。 使用BTNU按钮作为电路复位，SW0作为使能，SW1作为Up / Dn（1 = Up，0 = Dn），LED7作为LED0输出计数器输出。 完成设计流程，生成比特流，并将其下载到Basys3或Nexys4 DDR板。 验证功能。 查看“项目摘要”选项卡后，请填写以下信息。

2. Number of BUFG/BUFGCTRL \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Number of Slice LUTs used: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Number of FF used: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Number of DSP48E1 slices used: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Number of IOs used: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2-3。 用使用IP目录生成的8位内核设计一个8位向上/向下计数器。 生成核心时，将其设置为使用结构资源。 使用时钟向导从板载100 MHz时钟源生成5 MHz时钟，再用时钟分频器进行分频，以生成以一秒为周期的信号。 使用BTNU按钮作为电路复位，SW0作为使能，SW1作为Up / Dn（1 = Up，0 = Dn），LED7~LED0作为计数器输出。 完成设计流程，生成比特流，并将其下载到Basys3或Nexys4 DDR板。 验证功能。 查看“项目摘要”选项卡后，请填写以下信息。

3. Number of BUFG/BUFGCTRL \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Number of Slice LUTs used: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Number of registers used: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Number of DSP48E1 slices used: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Number of IOs used: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2-4。 使用2-3的8位向上/向下计数器设计，但计数器使用DSP48切片重新生成。 使用BTNU按钮作为电路复位，SW0作为使能，SW1作为Up / Dn（1 = Up，0 = Dn），LED7作为LED0输出计数器输出。 完成设计流程，生成比特流，并将其下载到Basys3或Nexys4 DDR板。 验证功能。 查看“项目摘要”选项卡后，请填写以下信息。

4. Number of BUFG/BUFGCTRL \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Number of Slice LUTs used: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Number of registers used: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Number of DSP48E1 slices used: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Number of IOs used: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**定时器和实时时钟**

定时器和实时时钟是计数器的自然应用。 定时器包括秒表计时器和失效计时器。 实时时钟用于日常生活中，以跟踪一天中的时间。

3-1。 使用IP目录设计秒表计时器，以生成具有所需输入控制信号的适当大小（精确度）计数器核心，以十分之一秒的分辨率测量长达五分钟的时间。 在多个所需时间实例化核心，并添加所需的附加电路，以在四个7段显示器上以M.SS.f格式显示时间。 设计输入为100 MHz时钟源，使用BTNU按钮的复位信号和使用SW0的使能信号。 当使能信号有效（ON）时，时钟计数，当它被置低（OFF）时，时钟暂停。 在任何时候如果按下BTNU，时钟将重置为0.00.0值。 使用Basys3或Nexys4 DDR板验证硬件中的设计功能。

3-2。 使用行为建模设计倒数计时器，以使用期望的输入控制信号对参数化的计数器向下计数器建模，以秒为分辨率显示从板子的两个滑动开关设置的期望初始倒计时间开始的倒计时。 在三个7段显示器上以MM.SS格式显示时间。 设计输入为100 MHz时钟源，使用BTNU按钮的重载信号，使用SW0的使能信号和SW7-SW6作为整分钟数的起始值。 当使能信号有效（ON）时，时钟计数，当它被置低（OFF）时，时钟暂停。 当按下BTNU时，定时器加载到MM.00，其中MM的值由SW7-SW6设置确定（MM = 00将被忽略）。 使用Basys3或Nexys4 DDR板验证硬件中的设计功能。

3-3。 使用IP Catalog设计实时时钟，以生成具有所需输入控制信号的适当大小（精度）计数器核心。 实例化两次并添加所需的电路以在四个7段显示器上以MM.SS格式显示时间。 设计输入为100 MHz时钟源，使用BTNU按钮提供复位信号。 在按下BTNU的任何时候，时钟重置为00.00。 使用Basys3或Nexys4 DDR板验证硬件中的设计功能。

**结语**

在本实验中，您学习了如何参数化模型，以便可以在后续设计中使用它们。 您还设计并比较了使用IP目录建模的计数器的资源使用情况。 您还使用计数器设计了时钟应用程序。