

# Vivado 下 LED 流水灯实验

黑金动力社区 2019-07-25

# 1 实验简介

通过 LED 流水灯实验,介绍使用 vivado 软件开发 FPGA 的基本流程,器件选择、设置、代码编写、编译、分配管脚、下载、程序 FLASH 固化、擦除等;同时也检验板上 LED 灯是否正常。

# 2 实验环境

- Windows 10 64 位
- vivado 2019.1
- 黑金 FPGA 开发板(AX7050 开发板、AX7035 开发板)

# 3 实验原理

## 3.1 LED 硬件电路



#### AX7050 开发板 LED 部分原理图

从上面的 LED 部分原理图可以看出, AX7035 开发板和 AX7050 开发板都是将 IO 经过一个电 阻和 LED 串联接电源端, FPGA 的 IO 输出低电平点亮 LED。IO 输出高电平 LED 灯熄灭,其中的串 联电阻都是为了限制电流。

## 3.2 程序设计

FPGA 的设计中通常使用计数器来计时,对于 50Mhz 的系统时钟,一个时钟周期是 20ns,那 么表示一秒需要 50000000 个时钟周期,如果一个时钟周期计数器累加一次,那么计数器从 0 到 49999999 正好是 50000000 个周期,就是 1 秒的时钟。

程序中定义了一个 32 位的计数器:

```
//Define the time counter
reg [31:0] timer;
```

最大可以表示 4294967295,十六进制就是 FFFFFFFF,如果计数器到最大值,可以表示 85.89934592秒。程序设计中是每隔1秒 LED 变化一次,一共消耗4秒做一个循环。

```
always@(posedge sys_clk or negedge rst_n)
begin
    if (~rst_n)
        timer <= 32'd0;
    else if (timer == 32'd199_999_999)
        timer <= 32'd0;
    else
        timer <= timer + 1'b1;
end</pre>
```

在第一秒、第二秒、第三秒、第四秒到来的时候分别改变 LED 的状态,其他时候都保持原来的值不变。

```
// LED control
always@(posedge sys_clk or negedge rst_n)
begin
    if (~rst_n)
        led <= 4'b0000;
    else if (timer == 32'd49_999_999)
        led <= 4'b0001;
    else if (timer == 32'd99_999_999)
        led <= 4'b0010;
    else if (timer == 32'd149_999_999)
        led <= 4'b0100;
    else if (timer == 32'd199_999_999)
        led <= 4'b1000;
    else if (timer == 32'd199_999_999)
        led <= 4'b1000;
    end</pre>
```

# 4 Vivado 工程

## 4.1 创建工程

1. 启动 Vivado 2019.1 开发环境(在开始菜单中选择 Xilinx Design Tools->Vivado 2019.1->Vivado 2019.1。 或者双击桌面的 Vivado 2019.1 的图标直接打开软件。



2. 在 Vivado 2019.1 开发环境里双击 Create Project,如下图:





3. 弹出一个 Vivado 的工程向导 , 点击 Next 按钮。

🚴 Hew Project		×
	Create a New Vivado Project	
HLx Editions	This wizard will guide you through the creation of a new project.	
	To create a Vivado project you will need to provide a name and a location for your project files. Next, you will specify the type of flow you'll be working with. Finally, you will specify your project sources and choose a default part.	
EXILINX ALL PROGRAMMABLE.		
?	<上一步(B) Lext > Einish Cancel	

4. 在弹出的对话框中输入工程名和工程存放的目录,这里取一个 led\_test 的工程名,点击 Next。

New Project						×
Project Name Enter a name for yo	our project and specify a dire	ctory where the p	roject data files will	be stored.		
<u>P</u> roject name:	led_test					$\bigotimes$
Project location:	D:/demo					
Create projec	ct subdirectory					
Project will be cr	reated at: D:/Demo/led_test					
(?)			< <u>B</u> ack	<u>N</u> ext >	<u>F</u> inish	Cancel

5. 在下面的对话框中默认选择 RTL Project, 因为我们这里使用 verilog 行为描述语言来编程。下面的 Do not specify source at this time 的勾也可以打上。如果不打上,下一步会进入添加 source file 界面,



lev P	roject 🗙
r <b>oje</b> pecify	ct Type (the type of project to create.
۲	RTL Project You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.
0	Do not specify sources at this time Post-synthesis Project: You will be able to add sources, view device resources, run design analysis, planning and implementation.
D	Do not specify sources at this time J/O Planning Project Do not specify design sources. You will be able to view part/package resources.
)	Imported Project Create a Vivado project from a Synplify, XST or ISE Project File.
)	Example Project Create a new Vivado project from a predefined template.
)	<上一步(B) <u>Next</u> > <u>Finish</u> Cancel

6. 进入添加 source file 界面,这里先不添加任何设计文件。点击 Next

Ker Project	×
Add Sources Specify HDL, netlist, Block Design, and IP files, or directories containing those files, to add to your project. Create a new source file on disk and add it to your project. You can also add and create sources later.	4
+,   -   +   +	
Use Add Files, Add Directories or Create File buttons below	
Add Files Add Directories Create File	
Target language: Verilog V Simulator language: Mixed V	
	cel

7. 提示是否添加已有的约束文件,这里约束文件我们也没有设计好,也不添加。

A New Project	×
Add Constraints (optional)	I
Specify or create constraint files for physical and timing constraints.	- 🔶
	_
Use Add Files or Create File buttons below	
Add Files Create File	
<     L     上     Einish     C	Cancel

8. 在接下来的对话框选择所用的 FPGA 器件,以及进行一些配置。FPGA 芯片型号一定要跟开发板 上的型号一致,AX7035 开发板首先在 Family 栏里选择 Artix-7, Speed grade 栏选择-2,在 Package 栏 选择 fgg484,然后在下面的列表中选择 xc7a35tfgg484-2;AX7050 开发板首先在 Family 栏里选择 Spartan-7, Speed grade 栏选择-1,在 Package 栏选择 fgga484,然后在下面的列表中选择 xc7s50fgga484-1;单击 NEXT 进入下一界面:

Parts   Boards									
Reset All Filters									
Category: All			v Pack	age: fgg48	4	~ Ter	nperature:	All Remaining	`
amily: Artix-7			~ Spee	ed: -2		✓ Sta	tic power:	All Remaining	`
xc7a15tfgg484-2	484	250	10400	20800	25	0	45	4	4
c7a35tfgg484-2	484	250	20800	41600	50	0	90	4	4
xc7a50tfgg484-2	484	250	32600	65200	75	0	120	4	4
xc7a75tfgg484-2	484	285	47200	94400	105	0	180	4	4
xc7a100tfqq484	2 484	285	63400	126800	135	0	240	4	4

AX7035 开发板 FPGA



Reset Al	Filters									
Category:	All			✓ Pace	kage: fgga4	84 ·	✓ Tem	perature:	All Remaining	`
Family:	Spartan-7			✓ Spe	ed: -1		✓ Stati	c power:	All Remaining	
Part		I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	Gb Transceivers	GT
				22600	65200	75	0	120	0	0
xc7s50f	gga484-1	484	250	32000	03200					
xc7s50f	gga484-1 gga484-1	484	250 338	48000	96000	90	0	140	0	0

AX7050 开发板 FPGA

9. 再次确认一下板子型号有没有选对,没有问题再点击"Finish"完成工程创建。

VIVADO.	New Project Summary
HLx Editions	A new RTL project named 'led_test' will be created.
	O No source files or directories will be added. Use Add Sources to add them later.
	• No constraints files will be added. Use Add Sources to add them later.
	The default part and product family for the new project. Default Part: xc7a35tfgg484-2 Product: Artix-7 Family: Artix-7 Package: fgg484 Speed Grade: -2
	To create the project, click Finish
(?)	< <u>B</u> ack <u>N</u> ext> <u>Einish</u> Cancel

#### AX7035 开发板 FPGA





AX7050 开发板 FPGA

10. 工程创建后如下图所示(AX7050 开发板):



# 4.2 编写流水灯的 verilog 代码

1. 点击 Project Manager 下的 Add Sources 图标 (或者使用快捷键 Alt+A)。

<u>Eile Edit Flow Tools Windo</u>	w Layout View Help Q- Quick Acc	ess	
, ∎j k i≁ in inixi≯	<b>Φ Σ % Ø %</b>		
Flow Navigator 🗧 🌩 ? 🔔	PROJECT MANAGER - led_test		
Y PROJECT MANAGER	Sources	2 – D B X	Project Summary
Settings			,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
Add Sources		*	Settings Edit
Language Templates	<ul> <li>Design Sources</li> <li>Constraints</li> </ul>		Project name:
👎 IP Catalog	~ 🗁 Simulation Sources		Project location:
	🗁 sim_1		Product family:
IP INTEGRATOR			Project part:
Create Block Design			Top module name:
Open Block Design			Target language:
Generate Block Design			Simulator language
✓ SIMULATION	Hierarchy Libraries Compile Order		Synthesis
Run Simulation	Properties	? _ 🗆 🖒 X	Status: No
			Messages: No
<ul> <li>RTL ANALYSIS</li> </ul>		+ + + + ₩	Part xc7
> Open Elaborated Design	[]		Strategy: Viv

2. 选择 Add or create design sources 选项,点击 Next。

À Add Sources	×
HLx Editions	Add Sources This guides you through the process of adding and creating sources for your project
	Add or create design sources     Add or create <u>s</u> imulation sources
?	<上一步(B) <u>Next</u> > <u>Finish</u> Cancel



3. 点击 Add Files 可以添加源文件,点击 AddDirectories 可以按目录添加源文件。因为现在我们还没有设计程序,这里要点击 Create File 按钮。

ecify HDL, netlist, Block Design, ar k and add it to your project.	IP files, or directories containing those file types to add to your project. Create a new source file or	n 🄰
+,  -   +   +		
	Use Add Files, Add Directories or Create File buttons below	
Scan and add RTL include files	Add Files Add Directories Create File	
Copy <u>s</u> ources into project Add so <u>u</u> rces from subdirectorie		

在弹出的对话框里选择 File type 是 verilog, File name 为 led\_test, 点击 OK 按钮。



4.点击"Finish"完成。



Add Source	95					
Add or Crea Specify HDL, n project. Create	<b>ate Design</b> netlist, Block D e a new source	Sources Design, and IP fi e file on disk an	iles, or directorie Id add it to your p	s containing those file project.	e types to add to you	r 🗼
+,   -	<b>†</b>   <b>‡</b>					
	Index	Name	Library	Location		
•	1	led_test.v	xil_defaultlib	<local project="" to=""></local>		
		<u>A</u> dd Files	A <u>d</u> d Dire	ectories <u>C</u> rea	te File	
Scan an	d add RTL <u>i</u> nc	lude files into p	roject			
Copy <u>s</u> o	urces into pro	ject				
🖉 Add so <u>u</u>	rces from sub	directories				
?			< <u>B</u> ack	Next >	<u>F</u> inish	Cancel

向导会提示您定义 I/O 的端口,这里我们可以不定义,后面自己在程序中编写就可以,单击 OK 完

### 成。

Define ∎odule						
Define a module For each port spe MSB and LSB Ports with blan	and specify I/ ecified: values will be k names will	O Ports ignore not be v	to add t d unless written.	to your s s its Bus	cource file. column is checked.	4
Module Definitio	n					
<u>M</u> odule nam	e: led_test					0
I/O Port Defi	nitions					
+ -	+ +					
Port Name	Direction	Bus	MSB	LSB		
	input 🗸		0	0		
						_
?					ОК Са	ncel



这时在 Project Manager 界面下的 Design Sources 里已经有了一个 led\_test.v 文件, 并且自动成为项目的顶层 (Top ) 模块了。



5. 接下去我们来编写 led\_test.v 的程序,这里我们定义了一个 32 位的寄存器 timer,用于循环计数 0~199\_999\_999(4 秒钟),当计数到 49\_999\_999(1 秒)的时候,熄灭第一个 LED 灯;当计数到 99\_999\_999(2 秒)的时候,熄灭第二个 LED 灯;当计数到 149\_999\_999(3 秒)的时候,熄灭第三 个 LED 灯;当计数到 199\_999\_999(4 秒)的时候,熄灭第四个 LED 灯,计数器再重新计数。具体的操作直接看代码吧。



```
//-----
// Module name: led test,v
`timescale 1ns / 1ps
module led test
(
           // system clock 50Mhz on board
 sys_clk,
               // reset ,low active
 rst_n,
               // LED,use for control the LED signal on board
 led
);
// PORT declarations
//-----
input
           sys clk;
input
           rst n;
output [3:0] led;
//define the time counter
reg [31:0] timer;
reg [3:0]
           led;
// cycle counter: from 0 to 4 sec
always @ (posedge sys clk or negedge rst n)
   begin
     if (~rst n)
        timer <= 32'd0;
                                        // when the reset signal valid, time counter clearing
     else if (timer == 32'd199_999_999) //4 seconds count(50M*4-1=1999999999)
        timer <= 32'd0;
                                          //count done, clearing the time counter
     else
          timer <= timer + 1'b1;</pre>
                                        //timer counter = timer counter + 1
   end
//------
// LED control
always @(posedge sys clk or negedge rst n)
   begin
     if (~rst n)
        led <= 4'b0000;</pre>
                                     //when the reset signal active
     else if (timer == 32'd49 999 999)
                                    //time counter count to 1st sec,LED1 lighten
        led <= 4'b0001;
     else if (timer == 32'd99 999 999) //time counter count to 2nd sec,LED2 lighten
     begin
        led <= 4'b0010;</pre>
       end
     else if (timer == 32'd149 999 999) //time counter count to 3nd sec_LED3 lighten
        led <= 4'b0100;</pre>
     else if (timer == 32'd199 999 999) //time counter count to 4nd sec,LED4 lighten
        led <= 4'b1000;</pre>
   end
endmodule
```

6. 编写好代码后保存,点击菜单 File -Save All Files。

# 4.3 添加 XDC 管脚约束文件

和 ISE 软件不同 , Vivado 使用的约束文件格式为 xdc 文件。xdc 文件里主要是完成管脚的 约束,时钟的约束,以及组的约束。这里我们需要对 led\_test.v 程序中的输入输出端口分配到 FPGA 的真实管脚上,这需要准备一个 FPGA 的引脚绑定文件.xdc 并添加到工程中。

1. 点击 Project Manager 下的 Add Sources 图标。



2. 选择 Add or create constraints 选项,点击 Next。

À Add Sources		×
HLx Editions	Add Sources This guides you through the process of adding and creating sources for your project Add or greate constraints Add or create design sources Add or create gimulation sources	
E XILINX ALL PROGRAMMABLE.		
?	<上一步(B) Lext > Einish Cancel	

3. 点击 Create File 按钮。

Add Sources
Add or Create Constraints Specify or create constraint files for physical and timing constraint to add to your project.
Specify constraint set: 📾 constrs_1 (active)
$ \mathbf{+}_{\mathbf{i}}  =  \mathbf{+}  \mathbf{+} $
Use Add Files or Create File buttons below
Add Files Create File Copy constraints files into project
?       < Back

在弹出的对话框里选择 File type 是 XDC, File name 为 led, 点击 OK 按钮。

ALINX

Create Constraints File							
Create a new constraints file and add it to your project							
<u>F</u> ile type:	File type:						
F <u>i</u> le name:	led	$\otimes$					
Fil <u>e</u> location:	🛜 <local project="" to=""></local>	~					
?	OK Cano	cel					

4.点击"Finish"完成。

À Add Sources	
Add or Create O	Constraints
Specify or create co	nstraint files for physical and timing constraint to add to your project.
Specify constrain	nt set: 📄 constrs_1 (active) 🗸
$ +_{\lambda}  =  + $	÷
Constraint File	Location
led.xdc	<local project="" to=""></local>
	Add Files Create File
Co <u>p</u> y constrai	nts files into project
(?)	< Back Next > Finish Cancel
$\odot$	

这时在 Project Manager 界面下的 Constraints 目录的 constrs\_1 目录下已经有了一个 led.xdc 文件。

#### www.heijin.org



5. 双击打开这个 led.xdc 文件,在这个文件里添加以下的引脚定义(以 AX7050 开发板为例)。

ALINX



set\_property CFGEVS VCC0 [current\_design] set\_property CONFIG\_VOLTAGE 3.3 [current\_design] set\_property BITSTREAM.CONFIG.SPI\_BUSWIDTH 4 [current\_design] set\_property CONFIG\_MODE SPIx4 [current\_design] set\_property BITSTREAM. CONFIG. CONFIGRATE 50 [current\_design] create\_clock -period 20 [get\_ports sys\_clk] set\_property IOSTANDARD LVCMOS33 [get\_ports {sys\_clk}] set\_property PACKAGE\_PIN P15 [get\_ports {sys\_clk}] set\_property IOSTANDARD LVCMOS33 [get\_ports {rst\_n}] set\_property PACKAGE\_PIN AE2 [get\_ports {rst\_n}] set\_property PACKAGE\_PIN H16 [get\_ports {led[0]}] set\_property IOSTANDARD LVCMOS33 [get\_ports {led[0]}] set\_property PACKAGE\_PIN G16 [get\_ports {led[1]}] set\_property IOSTANDARD LVCMOS33 [get\_ports {led[1]}] set\_property PACKAGE\_PIN K15 [get\_ports {led[2]}] set\_property IOSTANDARD LVCMOS33 [get\_ports {led[2]}] set property PACKAGE PIN J15 [get ports {led[3]}] set\_property IOSTANDARD LVCMOS33 [get\_ports {led[3]}]

XDC 文件前面两句是配置 CFGBVS 管脚的电压和配置电路的电压,因为在开发板上 CFGBVS 管脚是上拉到 3.3V 的,也就是 BANKO 的 VCCIO。另外配置电路的电压是 3.3V。所以这里分别配置成 VCCIO 和 3.3V。

下面来介绍一下最基本的 XDC 编写的语法, 普通 IO 口只需约束引脚号和电压, 管脚约束如下:

set\_property PACKAGE\_PIN "引脚编号" [get\_ports "端口名称" ]

电平信号的约束如下:

set\_property IOSTANDARD "电压" [get\_ports "端口名称"]

这里需要注意文字的大小写,端口名称是数组的话用{}刮起来,端口名称必须和源代码中的 名字一致,且端口名字不能和关键字一样。

时钟端口还可以定义时钟周期约束,比如我们在 XDC 里面定义了输入的差分时钟的时钟周期为 20ns。时钟周期的约束方法如下:

create\_clock -period "周期" [get\_ports "端口名称" ]

完成后选择菜单 File->Save all files 保存所有文件。

# 4.4 编译

点击 Run Synthesis,即可开始综合并生成网表文件:

<u>F</u> ile <u>E</u> dit F <u>l</u> ow <u>T</u> ools Rep <u>o</u> rt	s <u>W</u> indow Layout <u>V</u> iew <u>H</u> elp <u>Q- Quick Access</u>
🖕 🔸 🗇 🗉 🗈 🗙 🕨	III ↓ ∑ ↓ ∞ ↓ ∞ ↓ ∞ ↓ ∞ ↓
Flow Navigator	PROJECT MANAGER - led_test
✓ PROJECT MANAGER	Sources 2 _ D L X
Settings Add Sources	$\mathbf{Q}_{\mathbf{x}} \stackrel{\mathbf{x}}{=} \mathbf{\varphi} $
Language Templates	<ul> <li>■ Ied_test (led_test.v)</li> <li>✓ ■ Constraints (1)</li> </ul>
<ul> <li>IP INTEGRATOR</li> <li>Create Block Design</li> </ul>	<ul> <li>Consuls_1(1)</li> <li>Ied.xdc</li> <li>Simulation Sources (1)</li> </ul>
Open Block Design Generate Block Design	> 🚍 sim_1 (1) > 🚍 Utility Sources
	Hierarchy Libraries Compile Order
<ul> <li>RTL ANALYSIS</li> <li>&gt; Open Elaborated Design</li> </ul>	Source File Properties     ? _ □ □ ×       I led.xdc     ←   →   ☆       Encrypted:     No       Core Container:     No
<ul> <li>SYNTHESIS</li> <li>Run Synthesis</li> <li>Open Synthesized Design</li> </ul>	Used In Synthesis Implementation
✓ IMPLEMENTATION	General Properties

在 Tcl Console 窗口或者 Messages 窗口可以看到一些状态信息。



<ul> <li>IP INTEGRATOR</li> <li>Create Block Design</li> <li>Open Block Design</li> <li>Generate Block Design</li> </ul>	<pre>led.xdc &lt; li&gt; Simulation Sources (1)</pre>
<ul> <li>SIMULATION</li> <li>Run Simulation</li> </ul>	Hierarchy       Libraries       Compile Order         Source File Properties       ? _ □ Ľ ×
<ul> <li>RTL ANALYSIS</li> <li>&gt; Open Elaborated Design</li> </ul>	Ied.xdc     ←     ⇒     ☆       Encrypted:     No     ∧       Core Container:     No
<ul> <li>SYNTHESIS</li> <li>Run Synthesis</li> <li>Open Synthesized Design</li> <li>IMPLEMENTATION</li> <li>Run Implementation</li> </ul>	Used In  Synthesis  Implementation  General Properties
<ul> <li>&gt; Open Implemented Design</li> <li>&gt; PROGRAM AND DEBUG</li> <li>III Generate Bitstream</li> <li>&gt; Open Hardware Manager</li> </ul>	Tcl Console×MessagesLogReportsDesign RunsQXImage: Construct and the state

综合完成后,在下图界面的右上角会显示 Synthesis Complete。可以点这里的 Run Implementation 来开始布局布线:





布线完成后会在下图界面的右上角显示 Implementation Complete。然后点击 Generate Bitstream 即可生成 bit 文件。





bit 文件生成完成后,我们可以打开 Project Summary 页面的 Table 来查看板子上实际资源的使用情况,因为我们这里的 led\_test 程序比较简单,只用到了四个资源: LUT(查找表),FF(Flip Flop 寄存器),IO(管脚)和 BUFG(时钟 Buffer)。



黑金动力社区

# 4.5 vivado 仿真验证

接下来我们不妨小试牛刀,让仿真工具 Vivado 来输出波形验证流水灯程序设计结果和我们的预想是否一致。具体步骤如下:

1. 设置 Vivado 的仿真配置,右击 SIMULATION 中 Simulation Settings。



2. 在 Simulation Settings 窗口中进行如下图来配置,这里设置成 50ms(根据需要自行设定),其它按默认设置,单击 OK 完成。

ý.	Simulation	- N
Project Settings General	Specify various settings associated to Simulation	<u> </u>
Simulation	Target simulator: Vivado Simulator	~
Elaboration Synthesis	Simulator language: Mixed	~
Implementation	Simulation set: 🕞 sim_1	~
Bitstream	Simulation top module name: led_test	
> IP		
ool Settings	Compilation Elaboration Simulation Netlist Advanced	
Project	xsim simulate tcl post	^
IP Defaults	xsim.simulate.runtime 50ms	
Board Repository	xsim.simulate.log_all_signals	
Source File	xsim.simulate.no_quit	
WebTalk	voim aimulata austam, tal	~
Help	xsim.simulate.runtime	
Text Editor	Specify simulation run time	
3rd Party Simulators		

3. 添加激励测试文件,点击 Project Manager 下的 Add Sources 图标,按下图设置后单击 Next。



ALINX

4. 点击 Create File 生成仿真激励文件。

Add Sources dd or Create Sir pecify simulation sp	nulation Sources ecific HDL files, or direct	tories containing HDI	L files, to add to your project.	
reate a new source	ile on disk and add it to	your project.		
Specify simulation	set: 📄 sim_1	~		
$ +_j  =  + $	+			
	Use Add Files, Add D Add Files	irectories or Create I A <u>d</u> d Directories	File buttons below	
Scan and add R	TL <u>i</u> nclude files into pro	ject		
Copy <u>s</u> ources ir	to project			
Add so <u>u</u> rces fro	m subdirectories			
🕑 Include all desig	n s <u>o</u> urces for simulatio	n		
?	< <u>B</u>	ack <u>N</u> ext	> <u>F</u> inish	Canco

在弹出的对话框中输入激励文件的名字,这里我们输入名为 vtf\_led\_test。

À Create Source File						
Create a new source file and add it to your project.						
<u>F</u> ile type:	Verilog	~				
F <u>i</u> le name:	vtf_led_test	$\otimes$				
Fil <u>e</u> location:	😜 <local project="" to=""></local>	~				
?	ОКСа	ancel				

6. 点击 Finish 按钮返回。



Add Sources	;				×
Add or Creat Specify simulati Create a new se	te Simulatic ion specific HI ource file on d	on Sources DL files, or direct isk and add it to p	ories containing your project.	HDL files, to add to y	our project.
Specify simu	lation set:	sim_1	~	]	
+  =	+ I +				
	Index	Name	Library	Location	
•	1	vtf_led_test.v	xil_defaultlib	<local project="" to=""></local>	
	A	1d Files	A <u>d</u> d Directories	<u>C</u> reate File	
Scan and	add RTL <u>i</u> nclu	ide files into proj	ect		
Copy <u>s</u> ou	rces into proje	ct			
Add so <u>u</u> ro	ces from subd	irectories			
Include al	l design s <u>o</u> uro	ces for simulation	n		
?		< <u>B</u> ;	ack	lext >	sh Cancel

# 这里我们先不添加 IO Ports , 点击 OK。

À Def	Eine ∎odule							×
Det For M P	ine a module a each port species ISB and LSB va Ports with blank	and specify I/ cified: alues will be names will	ignore not be v	d unless written.	to your s s its Bus	column is chec	ked.	A
Мо	dule Definition	I						
	<u>M</u> odule name	vtf_led_te	st					0
	I/O Port Defini	tions						
	+  -	<b>*</b>   <b>*</b>						
	Port Name	Direction	Bus	MSB	LSB			
		input 🗸 🗸		0	0			
?	)						ОК	Cancel



在 Simulation Sources 目录下多了一个刚才添加的 vtf\_led\_test 文件。双击打开这个文件,可以看到里面只有 module 名的定义,其它都没有。

#### www.heijin.org



7. 接下去我们需要编写这个 vtf\_led\_test.v 文件的内容。首先定义输入和输出信号,然后需要实例 化 led\_test 模块,让 led\_test 程序作为本测试程序的一部分。再添加复位和时钟的激励。完成后 的 vtf\_led\_test.v 文件如下:

```
timescale 1ns / 1ps
// Module Name: vtf led test
module vtf led test;
   // Inputs
   reg sys_clk;
   reg rst n;
   // Outputs
   wire [3:0] led;
   // Instantiate the Unit Under Test (UUT)
   led test uut (
      .sys clk(sys clk),
      .rst_n(rst n),
      .led(led)
   );
```

ALINX



# initial begin // Initialize Inputs sys\_clk = 0; rst\_n = 0; // Wait 100 ns for global reset to finish #1000; rst\_n = 1; // Add stimulus here #20000; // \$stop; end always #10 sys\_clk = ~ sys\_clk; endmodule

8. 编写好后保存, vtf\_led\_test.v 自动成了这个仿真 Hierarchy 的顶层了, 它下面是设计文件

led\_test.v。



9. 点击 Run Simulation 按钮,再选择 Run Behavioral Simulation。这里我们做一下行为级的仿真就可以了。



如果没有错误, Vivado 中的仿真软件开始工作了。

10. 在弹出仿真界面后如下图, 界面是仿真软件自动运行到仿真设置的 50ms 的波形。

•	th 🔅 Σ 🕺 🖉	× • • •	10 ms •	× ≝ ∥ C		
-	SIMULATION - Behavioral Sime	ulation - Functional - sim_1 - vt	f_led_test			
î	Sco× Sour _ 🗆 🖾	× <sup>p</sup> r ? _ 🗆 🗹	vtf_led_test.v ;	K Untitled 2 X		
I.	Q   素   ≑   🔅	Q 🌼	Q, 💾 🖯 🕄	ର   ⊠   ୶   ।∢   ▶।	h± h±r h+F h≣e hat her h	
	Name ^	Name ^				<mark>49, 999, 999. 996 ns</mark>
	v stf_led_test	16 sys_clk	Name	Value	49, 999, 800 ns 49, 999, 900 ns	50,000,000 ns 50,00
	📒 uut	Ъ rst_n	18 sys_clk	1		<b>-</b>
	📒 gibi	> 18 led[3:0]	<mark>1</mark> ₄ rst_n	1		<b>↓</b> 50ms
11			> led[3:0]	0	0	
ч						

由于 LED[3:0]在程序中设计的状态变化时间长,而仿真又比较耗时,在这里观测 timer[31:0]计数器变化。把它放到 Wave 中观察(点击 Scope 界面下的 uut , 再右键选择 Objects 界面下的 timer , 在弹出的下拉菜单里选择 Add Wave Window)。

#### www.heijin.org



SIMULATION - Behavioral Simulation	on - Functional - si	m_1 - vtf_led_test								
Scope × Sources _ C	0bject	s ?_ 🗆	Ľ ×	vtf_led_test.v × Un	titled 2 $\times$					
Q   素   ♦	<b>\$</b> Q		ø	ର୍ 💾 ବ୍ ର୍	20 <b>•</b>	I II I	e er H	Te ⊨F ⊡		
Name De	sign L Name		Valı^							
✓ ■ vtf_led_test vtf_	led_t. 📸 s	ys_clk	0	Name	Value		İ	49, 999, 800 ns		49.9
🥘 uut 🛛 🗐 Ied	Ltest 🛅 n	st_n	1	🕼 sys_clk 0						
🛢 gibi gib	l 🗦 🐳 ti	mer[31:0]	002	Add to Wave Window	1					┍
	> 🚺 I	ed[3:0]	0	Log to Wave Database					0	
				Show in Wave Window						
				Go to Source Code						
				Padix						
				Show as Enumeration						
				Show as chumeration						
				Report Drivers						
				Force Constant						
				Force Clock						
				Remove Force						
				Default Radix	<b>•</b>					
			-							

添加后 timer 显示在 Wave 的波形界面上,如下图所示。

	% Ø X		500 us	✓ II	C							
SIMULATION - Behavioral S	IMULATION - Behavioral Simulation - Functional - sim_1 - vtf_led_test											
Scope × Sources	_ 🗆 🖻	Objects ? _	. 🗆 🖒 🗙	vtf_led_test.v ;	VINTITIEN 2*	×						
Q ≚ ≑	٥	Q	٥	Q, 📕 🗑 🛛	Q 🔀 📲	I II	t≛   ±r   +Γ					
Name	Design L	Name	Valı ^									
✓	vtf_led_t.	http://www.sys_clk	0	Name	Value			49 999 800 55		49 999 900 55		
🔒 uut	led_test	🛅 rst_n	1	🐚 sys_clk	0			10,000,000 11		10,000,000 1.1		
🧧 gibi	gibi	> 🔞 timer[31:0]	002	13 rst_n	1							
		> 📸 led[3:0]	0	> 🔣 led[3:0]	0				0			
				> 🦋 timer[3	0026256e							

11. 点击如下标注的 Restart 按钮复位一下,再点击 Run All 按钮。(需要耐心!!!),可以看到 仿真波形与设计相符。



<u>W</u> indow La <u>v</u> out	<u>V</u> iew <u>R</u> un	<u>H</u> elp	Q- Quick Ac	cess		
×▶∣₩∣¢∣	Σ % Ø	×	4 🖡 h	n		
SIMULATION - Behav	ioral Simulation $\mathbf{R}\epsilon$	Functiona	ll - sim_1 - vtf_	_led_tes		
Scope × Sour	ces _ 🗆	Ci Ob	jects	? _		
Q <b>≍</b> ♦	4	🕈 Rur	A11			
Name	Desig	yn L Na	Name			
✓	vtf_le	d_t.				
🥃 uut	led_t	est				
🥃 gibi	gibi	>				
		>	📸 led[3:0]			

L	vtf led test.v x	Untitled 2*	×																		
	Q, 💾 🔍 🤇	a   🔀   📲	ŀ	• •	12	tr   +	[+	•F   I	→												
																	1,000,00	0 <mark>, 990. 00</mark>	0 ns		
	Name	Value	<u>+</u>	<b> </b>		ļ		1, 000,	000, 900 ns			<b></b>	1,000,00	0,950 ns				1,000,00	1,000 ns		
	1 sys_clk	1																			<u> </u>
	1≦ rst_n	1																			
	> 📲 led[3:0]	1								0										1	
	> 👋 timer[31:0]	50000000	49	999993	4999	994	X 4999	995	X 4999	996	X 49991	997	X 49999	998	49999	999	5000	000	50000	001	5000

我们可以看到 led 的信号会逐一变 1 , 说明 LED1~LED4 灯逐个熄灭。

# 4.6 下载和调试

经过前面的编译和仿真,我们可以把 bit 文件下载到 FPGA 芯片中,看一下 LED 实际运行的 效果。下载和调试之前先连接硬件,把 JTAG 下载器和开发板连接,然后开发板上电(下图为 AX7035 开发板的硬件连接图)。



1. 下载之前还需进行设置:右击 PROGRAM AND DEBUG 按下图进行设置。

ہ Generate Block Design		🐼 uut:led_			
> SIMULATION	Hierarchy	Libraries			
Y RTL ANALYSIS					
> Open Elaborated Design	Properties				
	vtf_led_test.	v			
> SYNTHESIS	C Enabled				
✓ IMPLEMENTATION	Location:	D:/demo			
Run Implementation	Type:	Verilog			
> Open Implemented Design	Library:	xil_defa			
PROGRAM AND DEBUG	General Pr	operties			
Generate Bitstre Bitstrea	m Settings				
∽ Open Hardware 🗙 Close	e	Messages			
Open Target	4 = =	7			
Program Device	Hierarchy Summary	Hierarchy Summary			

选择生成 bin 文件,生成的 bin 文件在后面固化程序到 FLASH 里会用到,这里一起生成一下。



Q	Bitstream	>
Project Settings	Specify various settings related to writing bitstream	P.
General Simulation	Configure additional bitstream settings.	
Synthesis	✓Write Bitstream (write_bitstream)	
Implementation	tcl.pre	
Bitstream	tcl.post	
> IP	-raw_bitfile	
Teal Cattings	-mask_file	
Project	-no_binary_bitfile	
IP Defaults	-bin_file	
Source File	-readback_file	
Display	-logic_location_file	
WebTalk	-verbose	
Help	More Options	
> Text Editor 3rd Party Simulators		
<ul> <li>Colors</li> <li>Selection Rules</li> <li>Shortcuts</li> </ul>	<b>-bin_file</b> Write a binary bit file without header (.bin).	
Strategies ~		

设置完成后单击 Generate Bitstream 生成 bit 和 bin 文件。



2. 点击 Open target 按钮->Auto Connect,在 hardware 界面下会显示 xc7a35t\_0(AX7035 开发板) 或 xc7s50\_0(AX7050 开发板)的图标,说明 JTAG 连接已经建立。

~	IMPLEMENTATION <ul> <li>Run Implementation</li> <li>Open Implemented Design</li> </ul>	Select an object to see properties
~	PROGRAM AND DEBUG	
	👫 Generate Bitstream	
	<ul> <li>Open Hardware Manager</li> <li>Open Target</li> </ul>	Tcl Console × Messages Serial I/O Links Serial I/O
	Open rarget	
	Program Dev Recent	Targets arty STEPS. WRITE_BITSTREAM. ARGS. BIN_FILE true [

#### www.heijin.org

Navigator 🛣 🖛 🐔 🗕	HARDWARE MANAGER - IOCAINOSUXIIIIX_ICI/DI	ngnenv210512180081
ROJECT MANAGER ^	There are no debug cores. Program devic	ce Refresh device
▶ Settings		
Add Sourcos	Hardware ?	? _ □ Ľ × Vtf_led_test.v × led_test.v ×
Add Sources		D:/domo_AV7050/2_lod_test/lod_test_cros/cim_1/or
Language Templates		
F IP Catalog	Name	Status Q 🖬 ← → 🐰 🗉 🛍 //
-	<ul> <li>Iocalhost (1)</li> </ul>	Connected 16 . svs clk(svs clk).
	x arr tef/Digilent/2105121800	Open 17 .rst_n(rst_n),
INTEGRATOR	✓ ⊕ xc7s50_0 (2)	Programmed 18 .led(led)
Create Block Design	XADC (System Monitor)	19 );
Open Block Design	🌾 mt25gl128-spi-x1 x2 x4	20
Conorate Block Design		21 initial begin
Generate Block Design		23 sys clk = 0;
	Properties ?	? _ □ Ľ × 24 rst_n = 0;
MULATION		25
Run Simulation	<b>`</b>	26 // Wait 100 ns for global reset to f
		27 #1000;
TL ANALYSIS		28 rst_n = 1;
		30 <b>#20000</b> :
Open Elaborated Design	Select an object to see properti	ies 31 // \$stop;
		32 🔶 end
YNTHESIS		33
Run Synthesis		34 always #25 sys_clk = ~ sys_clk; //5ns-

右键选择 xc7a35t\_0 (AX7035 开发板 ) 或 xc7s50\_0 (AX7050 开发板 ),在弹出的选项里选择

Program Device 项。

PROJECT MANAGER	There are no debug cores.	Program device Refresh device	
Settings			
Add Sources	Hardware	? _ 🗆 🖆 ×	vtf_led_test.v × led
Language Templates	Q   素   ⊜   ∅   ▶	» 🔹 🗘	D:/demo_AX7050/2_led_
	Name	Status	
Y IP Catalog	🗸 🚪 localhost (1)	Connected	
	✓ ✓ xilinx_tcf/Digilent/2	105121800 Open	17 .rst_n(rst_n
Create Block Design	✓ (i) xc7s50_0 (i)	Hardware Device Properties	Ctrl+E <sup>1)</sup>
Open Block Design	🦉 XADC (S	Program Device	
Generate Block Design	· · · · · · · · · · · · · · · · · · ·	Verify Device	isliz
SIMULATION	Hardware Device Prop	Add Configuration Memory Devi-	= 0; ce
Run Simulation	@ xc7s50_0	Boot from Configuration Memor	y Device 100
	Name: xc	Program BBR Key	= 1;
RTL ANALYSIS	Part: xc	Clear BBR Key	stimu
> Open Elaborated Design	ID code: 03	Program eFUSE Registers	
SYNTHESIS		Export to Spreadsheet	
Run Synthesis	General Properties		34 always #25 sys_c

在弹出的 Program Device 对话框中,选择 led\_test 项目生成的 bit 文件,点击 Program 按钮 烧写 FPGA。



🍌 Program Device	10 10 10 10 10 10 10 10 10 10 10 10 10 1								
Select a bitstream programming file and download it to your hardware device. You can optionally select a debug probes file that corresponds to the debug cores contained in the bitstream programming file.									
		_							
Bitstre <u>a</u> m file:	D:/demo_AX7050/2_led_test/led_test.runs/impl_1/led_test.bit								
Debu <u>a</u> probes file:									
✓ Enable end of st	tartup check								
?	Program Cance	el							

烧写完成后 xc7a35t\_0(AX7035 开发板)或 xc7s50\_0(AX7050 开发板)的状态会变成 Programmed, 这时我们可以看到开发板上的四个 LED 灯已经在做流水灯动作了。



你也可以试着别的花样来点亮 LED,比如,让灯跑得更快一些,或几个灯同时亮同时灭等等, 就看你的想象力了,通过自己写程序更能有成就感,而且还能把书本的知识用到实际中,何乐而 不为呢!是吧?

#### 4.7 FLASH 程序固化

可能已经有朋友发现下载 Bit 文件到 FPGA 后,开发板重新上电后配置程序已经丢失,还需要 JTAG 下载。这岂不麻烦!好吧,这一节我们来介绍如何把配置程序固化到开发板上的 FLASH 中, 这样不用担心掉电后程序丢失了。 在我们的开发板上有一个 8Pin 的 128Mbit 的 FLASH, 用于存储配置程序。我们不能直接把 Bit 文件下载到这个 FLASH 中,只能下载 Bin 文件或者 MCS 文件,在前面我们生成 bit 文件的同时也 生成了 Bin 文件。下面以下载 Bin 文件为例为大家介绍 FLASH 程序的固化。

1. 在如下图中右键选择 xc7a35t\_0(AX7035 开发板)或 xc7s50\_0(AX7050 开发板)芯片,在弹出 的列表中选择 Add Configruation Memory Device...

Flow Navigator — — — — — — — — — — — — — — — — — — —	HARDWARE MANAGER - loc	alhost/xilinx_tcf/Dig	gilent/21051218	0081	
✓ PROJECT MANAGER ^	There are no debug core	s. Program device	Refresh devic	e	
🌣 Settings	-			r	
Add Sources	Hardware	?	_ 🗆 🖒 ×	vtf_led_test.v	× led
Language Templates	Q   ≚   ♦   ∅		\$	D:/demo_AX705	50/2_led
T IP Catalog	Name		Status	0 🖬 🛧	-
	🗸 🚦 localhost (1)		Connected	16	e elk (sm
	✓ I or xilinx_tcf/Digiler	nt/2105121800	Open	17 .rst	i_n(rst_i
• IP INTEGRATOR	✓ ⊕ xc7s50_0(1)		Programmed	18 .lea	1(1ed)
Create Block Design	🔯 XADC	Hardware Device	Properties	Ctrl+E	
Open Block Design		Program Device			begin
Generate Block Design	<	Verify Device			ritiali.
	~	Defreck Device			:1k = 0
	Hardware Device Pr	Refresh Device			ι = 0;
Pup Simulation	@ xc7s50_0	Add Configuratio	e	ait 100	
Kun Sinulation		Boot from Config	Boot from Configuration Memory Device		
	Name:	Program BBR Ke	ev		n = 1
· RTEANALYSIS	Part:	Clear BBB Key			id stim: 10.
> Open Elaborated Design	ID and a	Clear DDR Rey			-lop;
	ID code:	Program eFUSE	Registers		
✓ SYNTHESIS	<	Export to Spread	sheet		L
Run Synthesis	General Properties	1			(5 sys_

注意:如果发现此项变为灰色不能选,是因为我们提供的工程中已经添加了 FLASH 配置,不能再添加 FLASH,如下:



Hardware	?	_ 🗆 🗆 ×	
Q   ¥   ♦   ∅   ▶   >	>   🔳	•	
lame		Status	
<ul> <li>Iocalhost (1)</li> </ul>		Connected	
✓ ✓ × ilinx_tcf/Digilent/21051	21800	Open	
~ 🛈 xc	1	Not programm	
🦉 XADC (System	Hardw	vare Device Properties	Ctrl+E
👒 mt25ql128-spi-	Progra	am Device	
	Verify	Device	
flash	Refres	sh Device	
	Add C	onfiguration Memory Device	
lardware Device Properties	Boot fr	rom Configuration Memory Devic	e
t xc7a200t 0	Progra	am BBR Key	
	Clear	BBR Key	
Name: xc7a200t	Progra	am eFUSE Registers	
Part: xc7a200t	Export	to Spreadsheet	

当然自己如果想在已有 flash 的工程中再次添加一下进行实验话,可按如下图移除 flash,然后 按上面添加 flash 的步骤进行即可:

Name		Status		
<ul> <li>Iocalhost (1)</li> </ul>		Connected		
✓ ✓ xilinx_tcf/Digilent/210	5121800	Open		
(2)		Not programm		
o XADC (System Monitor)				
🌼 mt25ql128-sni-x1_x2_x4				
Configuratio		on Memory Devic	e Properties	Ctrl+E
	Remove Co		ory Device	Delete
<	Program Configuration Memory Device			
Configuration Memory D	Readback Configuration Memory Device			
computation memory bi	Export to Spreadsheet			
mt25ql128-spi-x1_x2_x4	-	<del>-</del> - <del>-</del>		

2. 在 Add configruation Memory Device 的配置界面里选择正确的 FLASH 型号,如下图所示:



Add Configuration Memory Dev	<b>rice</b>						
Choose a configuration memory part. This can be changed later.							
Davica: @ xc7a100t_0							
tei							
Manufacturer Micron	~		Туре	All		~	
Density ( <u>M</u> b) 128	*		Width	x1_x2_x4		~	
		<u>R</u> eset A	II Filters				
elect Configuration Memory Part							
Search: Q-							
Name	Part	Manufact	Alias	Family	Туре	Density (	
🕸 mt25ql128-spi-x1_x2_x4	mt25ql128	Micron	n25q128-3.3v-spi-x1_x2_x4	mt25ql	spi	128	
🌼 mt25qu128-spi-x1_x2_x4	mt25qu128	Micron	n25q128-1.8v-spi-x1_x2_x4	mt25qu	spi	128	
<				,	-	>	
(?)					ОК	Cancel	
<u> </u>							

3. 提示是否对 SPI FLASH 进行编程,点击 OK。

Ad	d Configuration Memory Device Completed	×
	Po you want to program the configuration memory device now?	
:	Don't show this dialog again	
:	OK	

在弹出的 Program Configuration Memory Device 窗口中, Configration file 项选择 Vivado 生成的 led\_test.bin 文件(此文件默认在 imp1\_1 目录下)。PRM File 项不用选。另外在这个窗口用户还可 以配置 I/O 为上拉,下拉或者无上下拉。配置操作选项保留默认就可以。



AProgram Configuration Hemory Device						
Select a configuration file and set programming options.						
Memory Device:		@mt25qI128-spi-x1_x2_x4				
Configuration file:		st.runs/impl_1/led_test.bin 💿				
PR <u>M</u> file:						
State of non-config mem	I/O pins:	Pull-none 🗸				
Program Operations						
Address Ra <u>n</u> ge:	Configura	ation File Only 🗸 🗸				
✓ Erase						
Blank Check						
✓ Program						
✓ <u>V</u> erify						
(?)	ок	Cancel Apply				

点击 OK 开始编程 FLASH。

Program Configuration Memory Device		×
Performing program operation - Step 2 of 2		
		18%
	<u>B</u> ackground	<u>C</u> ancel

FLASH 编程完毕后,会弹出如下成功的界面。



至此, SPI FLASH 烧写完毕, led\_test 程序已经固化到 SPI FLASH 中了。我们来验证一下,关电重新启动开发板,等待一会儿你就可以看到开发板上的 LED 灯已经在做跑马运动了。

可能您也发现了,关电后重新上电需要等好一会儿,开发板上的 LED 灯才会开始启动跑马动 作。这对有些上电马上就要工作的项目肯定是不满足了,那有没有办法解决的呢!当然有的,我 们可以提高 SPI FLASH 的读写时钟,另外使用 x4 的方式读写,QSPI 是支持 4 根数据线读写的哦! 方法很简单,我们只要在 xdc 文件里加入以下 3 条语句:

其中前面两条设置 QSPI FLASH 的数据宽度和配置模式,后面一条是配置速度,这个值越大, 速度越快。修改 xdc 文件后需要重新编译,再重新生成 bit 和 bin 文件,然后按前面的方法再烧写 一遍 SPI FLASH 芯片哈。完成后开发板上电,这下是不是一上电,LED 灯就开始做运动了吧?

这里为止,我们的第一个项目就圆满完成了,相信您也掌握了 Vivado 的 FPGA 开发的整个流程,再也不是那个 FPGA 的门外汉了吧!师傅领进门,修行还需要靠本身!vivado 软件的一些技巧的使用和掌握就需要靠大家在长期实践和探索中慢慢熟悉了。

# 5 附录

led\_test.v(verilog 代码)

```
timescale 1ns /
module led_test
(
                      sys_clk,// system clock 50Mhz on boardrst_n,// reset ,low activeled// LED,use for control the LED signal on board
      input
      input
      output reg[3:0] led
);
//define the time counter
reg [31:0]
                timer;
// cvcle counter:from 0 to 4 sec
always@(posedge sys_clk or negedge rst_n)
begin
      if (~rst_n)
                timer <= 32'd0;
                                                //when the reset signal valid, time counter clearing
                                              //4 seconds count (50M*4-1=199999999)
      else if (timer == 32'd199_999_999)
               timer <= 32'd0;
                                               //count done, clearing the time counter
      else
               timer <= timer + 1'b1;</pre>
                                              //timer counter = timer counter + 1
end
// LED control
always@(posedge sys_clk or negedge rst_n)
begin
      if (~rst_n)
               led <= 4'b0000;
                                                //when the reset signal active
      else if (timer == 32'd49 999 999)
                                               //time counter count to 1st sec, LED1 lighten
                led <= 4'b0001;
      //time counter count to 2nd sec,LED2 lighten
      else if (timer == 32'd149 999 999)
                                              //time counter count to 3rd sec.LED3 lighten
               led <= 4'b0100;</pre>
      else if (timer == 32'd199 999 999)
                                               //time counter count to 4th sec.LED4 lighten
               led <= 4'b1000;</pre>
end
endmodule
```

黑金动力社区

注意:在定义寄存器时,如果寄存器在 always 块里使用必须定义为 reg 类型,如果仅是用于 连线或是直接赋值需定义为 wire 类型,输入信号的类型不能定义为 reg 型,不管是 reg 类型信号 还是 wire 类型的信号,定义的寄存器宽度必须满足使用时的需要,但必须稍大于或等于需要使用 的位宽。若定义寄存器位宽远远大于使用需求则会浪费资源,如果定义的位宽小于使用需求,则 会造成数据位截断,导致程序错误。还有其他信号的类型及用法请大家参考 Verilog 语法教程。