实验 07 FPGA 实验平台及 IP 核使用

简介



前面的实验中,我们完成了一个简单的电路设计,并顺利的烧写 到 FPGA 实验平台上。但可能会有部分读者对于其中的一些步骤并不 了解,本次实验中,我们将对所使用的 FPGA 实验平台进行介绍,以 帮助读者加深对 FPGA 开发流程各环节的理解。此外,我们还会介绍 到如何使用 IP 核进行电路设计。

实验目的

熟悉 FPGAOL 在线实验平台结构及使用

掌握 FPGA 开发各关键环节

学会使用 IP 核(知识产权核)

实验环境

VLAB 平台: vlab.ustc.edu.cn

FPGAOL 平台: fpgaol.ustc.edu.cn

Vivado

Logisim

实验步骤

Step1. FPGAOL 实验平台介绍

FPGA online(简称 FPGAOL)是中国科学技术大学计算机学院实验中心自主开发的一套在线实验平台,该平台通过分时复用的方式为用户提供 FPGA 设备节点的在线服务,目前 FPGAOL 一代系统已正式对外提供服务,用户可通过网址 fpgaol.ustc.edu.cn 访问平台,并使用统一身份认证登陆。



FPGAOL 平台的每个设备节点都包含了一个树莓派和一个 FPGA 板 卡,树莓派与 FPGA 芯片的 26 个 10 管脚直接相连,并最终映射到网 页端,其中包括 8 个对应 LED (同时与七段数码管复用),8 个对应 开关,7 个对应8 位 16 进制数码管,1 个对应按键,2 个对应串口 UART。 用户在网页端对虚拟化外设进行的操作都会通过树莓派映射到对应 的 FPGA 管脚上。同理,FPGA 管脚的电平变化也会通过树莓派反应到 网页端的虚拟化外设上。因此,用户是通过网页端的虚拟化外设实际 地控制物理设备节点。

除了网页端的虚拟化外设,FPGA 板卡上还带有板载 100MHz 时钟, 128MB DDR 内存芯片,以及各类传感器及接口外设,如温度传感器、



麦克风等,具体管脚对应关系可参考下图或网站提供的 XDC 文件。

Step2. 外设工作原理介绍

FPGAOL 实验平台的开发板上有一个 100MHz 的时钟晶振,该晶振 与 FPGA 芯片的 E3 管脚直接相连,在开发板上电后,会持续的为 FPGA 芯片提供一个 100MHz 频率的时钟信号,不需要用户进行操作,如下 图所示。



FPGAOL 实验平台中 LED 的工作方式为: FPGA 管脚为高电平时对应 LED 点亮,为低电平时则熄灭,其原理图如下所示:



拨动开关作为输入设备,当开关推到上方后,对应 FPGA 管脚输入 高电平,当开关拉下来之后,对应 FPGA 管脚输入低电平,其工作原 理如下图所示:



按键的工作原理与拨动开关类似,区别在于:按键按下时向 FPGA 芯片输入高电平信号,松开时输入低电平信号,按键默认处于松开状态。FPGAOL 1.0 实验平台中只有一个按键,且与 soft clock 功能复用(实际上 soft clock 功能暂未对用户开放)。

因平台可用的 FPGA 管脚数量有限,我们对七段数码管与 LED 的管脚进行了复用。数码管本质上是由 8 个 LED (发光二极管)构成,发光二极管的阴极共同连接到一端,并接到地,8 个阳极分别由 FPGA 的 8 个输出管脚控制,当输出管脚为高电平时,对应的 LED 亮起。通过控制 8 个 LED 的亮灭,便能显示出不同的字符,例如,当 A[~]F 6 个 LED 亮,G、DP 两个 LED 不亮时,显示的便是字符"0"。



平台包含一个8位的十六进制数码管,该数码管共用4bit位宽的数据位,同时由3bit的选择位经译码后生成8个选择信号,控制8

个数码管的使能,数据位个选择位信号都是高电平有效。其工作原理如下图所示。



此外,FPGAOL平台还集成了一个串口终端,FPGA芯片上的两个 IO 直接与树莓派上的串口相连,在 FPGA端编写程序,将 RXD 信号直 接与 TXD 信号直连,然后在网页端的串口终端发送数据,便能够实时 接收到从 FPGA 侧环回的数据了,感兴趣的同学可以实际测试一下。



Step3. 使用时钟管理单元 IP 核

在 FPGA 开发中,有很多常用功能的模块是不需要自己开发的,用 户可以复用第三方开发好的模块,这种模块被称为 IP 核。此处我们 先学习时钟管理单元 IP 核的使用。

前面已经讲到, FPGA 开发板的 FPGA 芯片 E3 管脚连接了一个 100MHz 频率的时钟晶振,可用作时序逻辑电路的时钟信号。如果我 们需要一个其它频率的时钟信号,例如 10MHz,应该怎么办呢?一般 的做法是通过计数器产生一个低频的脉冲信号,然后再将该脉冲信号 控制其他逻辑的控制信号,如下代码所示,通过 pulse_10mhz 信号控 制 1ed 信号。

```
module ttt(
input clk, rst,
output reg led);
reg [3:0] cnt;
wire
            pulse 10mhz;
always@(posedge clk)
begin
    if(rst)
        cnt \leq 4'h0:
    else if(cnt>=9)
        cnt <= 4'h0;
    else
        cnt \leq cnt + 4'h1;
end
assign pulse 10mhz = (cnt == 4'h1);
always@(posedge clk)
begin
    if(rst)
        led \langle = 1'b0;
    else if(pulse_10mhz)
        led \leq ~led;
end
```

endmodule

其仿真波形如下图所示

ame	Value		1550 ns		isoo ,			l650 -			1700	ns		17	750 n	e			800 1	.	
li clk	0	ПГ								П				ľ							
1 rst	0																				
🐸 ent[3:0]	0) () ()	<u>)</u> (6)(1	89		2 3	4	X	7 8	<u>)(e)(</u>	Xī)(2)	3 (4	Хİ)6)		e X 9	XC	Xī	2 3	
use_10mhz	0					L															
led 🛛	1																				

有些读者喜欢将分频信号直接作为时钟使用,虽然这种设计也能 工作,但我们强烈反对这种实现方式,因为这样做会产生许多无谓的 警告,也容易引起电路工作的不稳定。希望读者时刻牢记,在实际的 数字电路中(仿真代码除外),时钟信号是非常特殊的一种信号,它 不应该出现在过程语句和连续赋值语句内部,如下所示的代码都是不 推荐的(clk为时钟信号 signal_x 为其它普通信号)

```
错误示例: assign signal_1 = func(clk, xxx);
错误示例: always@(posedge signal_2) //signal_2 由 clk 生成
错误示例: always@(posedge clk or negedge clk) //两边沿不能同时使用
```

时钟信号只应该出现在 always 语句的时序控制部分

正确示例: always@(posedge clk) //时钟上升沿触发,同步复位或无复位 正确示例: always@(negedge clk) //时钟下降沿触发,同步复位或无复位 正确示例: always@(posedge clk or posedge rst) //异步复位,高有效 正确示例: always@(posedge clk or negedge rst_n) //异步复位,低有效

如果设计中确实需要不同频率的时钟信号应该通过时钟管理单元 IP核生成。首先,点击"IP catalog",在对应窗口中选中"Clocking Wizard"并双击。



在弹出的窗口中,对其进行设置,输入时钟频率设置为100MHz,输出时钟有两个,分别为10MHz和200MHz。

		Customite P		O Controller P							
		Clocking Wizard (5.3)	<u> </u>	Clecking Wixard (5.3)							
				😰 Bormonitation 🚞 IP Location 📪 Switch to Beford to							
	All of the second of the s	a based in some	Amount Barriel da six 6	IP Symbol Incomerce	Corporat Base clauses)						
		Within Simplify parts	Instance over the second se	Willow decabled ports	Inst Clubbag Spring	Balgat Clocks. MCH Se	things Port Beaming St	umary .			
			The state of the s		Do phase in calculated rel	stive to the active input o	ink.				
		A CONTRACTOR OF	12 faults Code Mexicania	E Constantino de la constant	Datgas Check Balant	free (Min)	Phase Generat	Actual	Buty Cyale (%)	Actual	De
		410,00,0		Contraction of Contraction	E clk_esti 10.000	15.000	0.800	0 0.000	90.000	1 90.0	22
		•••••••••••	Trainities	Contrast and a	E 434, eas2 200.000	291.900	0.000	0 0.000	50.006	10.0	22
	•••••••••••••••••••••••••••••	Call Areasta	8 mon () FL	A CALL AND THE	C 434, eet3 #80.000	2.1	6 900		90.000		30
	Image: Section of the section of t			and see all a second se	C eth. met. 400 000	2.2	0.000		50.000		1
Image: Control of the stand		th out o	Coding Ferrares Jitter Sptinization.	- 100/will.by	Calk.ext5 390.000	10	0.900	1.1	50.000	10.	
	Image: Section of the section of t	13,0013,019	🛙 brepany lystanis 🗇 Kainiss Pare 🔹 Bilanol	- Charles and an and an	T1 414 4415 180 000		0.900		20.000		
Image: marked of ma	Base And Base Base	the set of a main the	Enser Alignent. Spring Spring Officiality States	and the second second second second	T100.001 100.000		0.000		100 000	1.5	
Image: Second	Image:	-th_out(_cotht)(t)(t)=	Denneir Recording Denneir Dane Shift O Rations Spot Jitter filtering		and the second second						
Note: Note: Note: Note: Note: Note: Note: Note: Note: Note:	And Control A	th and m all and the	Fibb Geb State	- increasing and and and	CAR CAR SHERCEN		Locking Peobleck				
North Roll Mark Machine State Mark Machine State Mark Machine State Mark Machine State Mark Mark Machine State Mark Mark Mark Mark Mark Mark Mark Mark	Note: Note: Note: Note: Note: Note: Note: Note: Note: Note: Note: Note: Note: Note: Note: Note: Note: Note: Note: Note: Note: Note: Note: Note: Note: Note: Note: Note: Note: Note: Note: Note: Note: Note: Note: Note:	cik_outh_cir cik_outh		- according to a statement	and the second sec		10000				
Image: Section 1 Image: Section 2 Image: Section 2 <td>Specify Comparing /td> <td>allowed, as an allowed with a second</td> <td>Ignoria, Secondia, Saturface, Sptima,</td> <td></td> <td>Bulget Clash</td> <td>restore Parter</td> <td>🖉 Automatic Co</td> <td>extral Oprikip</td> <td>10 Single on</td> <td>÷.</td> <td></td>	Specify Comparing	allowed, as an allowed with a second	Ignoria, Secondia, Saturface, Sptima,		Bulget Clash	restore Parter	🖉 Automatic Co	extral Oprikip	10 Single on	÷.	
Image: Section of the section of t	Image: Constraint of the	th, out, or	@ 4254.05 O 202 O faint but field field g _ brits 102 registers	- industry and	43,440		🔿 Antonatio Co	sales of the	C Editorial	÷.	
And Charles Fact Charles State State State State Sta	And the set of the first set of the set of t	dik_out 7_six		and the second second	194,000		C Four Castrol	Lini de-Chip			
	Image: Section of the section of t	Seconda .	Super Clock Information	and the second			© Install	Lief Off-Day			
None None <th< td=""><td>A series and a ser</td><td>- ort, 113</td><td>Tapat Goda Tapat Propinsy 900) Stitur Optima Tapat Stitur Tearsa</td><td></td><td>dame.</td><td></td><td></td><td></td><td></td><td></td><td></td></th<>	A series and a ser	- ort, 113	Tapat Goda Tapat Propinsy 900) Stitur Optima Tapat Stitur Tearsa		dame.						
Angle	incluit B Sender INV C 0 * 10 * 10 Note that the sender.* incluit Incluit Incluit Incluit Incluit incluit Incluit Incluit Incluit Incluit	and the	Primary 200 000 0 10 000 - 800 000 WI - 0.000 Single ended tirely republic pix -		- 10 mm						
- ALM - AND	Atta	- 1142 (112	Seesdary 100.000 12.10 - 114.200 0.000 finds and their spalls								
				-rik, sai	Radia Optional Supate / Or	epsta	Brist Type				

设置完成后点击确认按钮,生成 IP 核。生成的 IP 文件可在"工程目录/工程名.srcs/sources_1/ip/IP 核名称/IP 核名称.v"找到。 用户可在设计文件中像调用其它模块一样使用该 IP 核,使用时只需 要了解 IP 核的功能及端口信号的含义及时序,而不用关心模块内部 的具体实现。如下代码所示,分别同 10M 和 200M 的时钟作为计数器 的计数时钟,通过仿真和烧写 FPGA,可以发现两个时钟频率的明显 差异。

```
module test(
input
                     clk,
input
                     rst,
output
            [7:0]
                    led);
wire
            clk 10m, clk 200m, locked;
reg [31:0] cnt_1, cnt_2;
always@(posedge clk 200m)
begin
    if (~locked)
        cnt 1 <= 32' hAAAA AAAA;
    else
        cnt 1 <= cnt 1+1'b1;;
end
always@(posedge clk_10m)
begin
    if(~locked)
        cnt 2 <= 32'hAAAA AAAA;
    else
        cnt 2 <= cnt 2+1'b1;;
end
assign led = {cnt 1[27:24], cnt 2[27:24]};
clk_wiz_0 clk_wiz_0_inst(
.clk inl
            (clk),
            (clk 10m),
.clk out1
.clk_out2
            (c1k_200m),
.reset
            (rst),
.locked
            (locked));
endmodule
```

强烈建议读者在综合烧写之前先进行功能仿真,以确保电路功能

的正确性,下面是一段最简单的仿真测试文件。

```
`timescale lns / lps
module tb( );
reg clk,rst;
initial
begin
    clk = 0;
    forever
    #5 clk = ~clk;
end
initial
```

```
begin
    rst = 1;
    #100 rst = 0;
end
test test(
. clk (clk),
. rst (rst),
. led ());
endmodule
```

其仿真波形如下所示:



对于 FPGA 开发板,我们将 LED 信号分配到 8 个 LED 灯上,其 XDC

```
文件如下:
```

<pre>set_property -dict { PACKAGE_PIN E3</pre>	<pre>IOSTANDARD LVCMOS33 } [get_ports { clk }];</pre>
<pre>set_property -dict { PACKAGE_PIN B18</pre>	<pre>IOSTANDARD LVCMOS33 } [get_ports { rst }];</pre>
## leds	
<pre>set_property -dict { PACKAGE_PIN G18</pre>	<pre>IOSTANDARD LVCMOS33 } [get_ports { led[7] }];</pre>
<pre>set_property -dict { PACKAGE_PIN F18</pre>	<pre>IOSTANDARD LVCMOS33 } [get_ports { led[6] }];</pre>
<pre>set_property -dict { PACKAGE_PIN E17</pre>	<pre>IOSTANDARD LVCMOS33 } [get_ports { led[5] }];</pre>
<pre>set_property -dict { PACKAGE_PIN D17</pre>	<pre>IOSTANDARD LVCMOS33 } [get_ports { led[4] }];</pre>
<pre>set_property -dict { PACKAGE_PIN G17</pre>	<pre>IOSTANDARD LVCMOS33 } [get_ports { led[3] }];</pre>
<pre>set_property -dict { PACKAGE_PIN E18</pre>	<pre>IOSTANDARD LVCMOS33 } [get_ports { led[2] }];</pre>
<pre>set_property -dict { PACKAGE_PIN D18</pre>	<pre>IOSTANDARD LVCMOS33 } [get_ports { led[1] }];</pre>
<pre>set_property -dict { PACKAGE_PIN C17</pre>	<pre>IOSTANDARD LVCMOS33 } [get_ports { led[0] }];</pre>

需要特别指出的是,时钟管理单元并不能产生任意频率的时钟信号,以前面为例,输入时钟为100MHz时,输出时钟只能是频率范围在4.687MHz[~]800MHz之间的某些频率,如4MHz超出范围无法产生, 799MHz 虽然在频率范围内,但因精度问题,只能产生以及频率相近的时钟信号,有兴趣的读者可自行验证。

如果电路中需要一个较低频率的时序,例如我们需要一个每秒钟加一的计数器,那我们就只能通过前面介绍的方法,用周期为1秒的

脉冲信号来控制了。

Step4. 使用片内存储单元

在前面的实验中,我们已经对存储器的行为特性进行了简单的介绍,例如 ROM 为只读存储器,包含了地址和数据两组端口,ROM 内部 包含 2^{地址位宽}个存储单元,每个单元里存储了与数据端口相同位宽的数据,存储器电路会将与地址相对应单元的数据呈现在数据端口上,下 图是一个包含 2²=4 个存储单元的 ROM,数据位宽为 16 位。



我们可以从 ROM 读取数据,但无法通过端口修改其内容,与此相对应 的是 RAM (随机存储器),其内容可读可写,下图为 Logisim 中的 RAM 模型,包含地址(A)、数据(D,输入输入复用)、片选(sel)、 时钟(clk,数据可在时钟上升沿写入)、输出使能(ld,为1 时数 据端口为输出,否则为输入)、清空(clr)等端口信号。这种 RAM 包含一套读写端口,因此成为单端口 RAM。读者可在 Logisim 中实际 体验各端口信号的功能。



Vivado 中也提供了存储器相关的 IP 核,种类和接口类型比 Logisim 中更加丰富,我们通过实际的例子来学习一下。IP 核目录中 提供了两种存储器实现方式:"Distributed Memory"和"Block Memory" 两种,我们以"Distributed Memory"为例进行介绍。



在存储器参数设置页面,用户可以对 IP 核名称、存储单元深度和 位宽、存储器类型等进行设置。在页面的左侧是模块的端口图。在上 图中,存储器深度设为 16=2⁴,数据位宽为 3 位,因此地址信号为 4 位,数据端口位宽为 3。存储器类型选为简单双端口,因此包含了两 套端口,其中 dpra、dpo 构成了读端口,d、a、clk、we 构成了写端 口。在"RST&Initialization"页面,我们还可以通过后缀为 coe 的 文件对存储器进行初始化,如下图右侧所示,coe 文件格式非常简单, 可以以文本方式打开和编辑。其中"memory_initialization_radix =16 " 表示 coe 文件采用 的是 16 进制方式显示, "memory_initialization_vector = ..."表示的是用 16 进制显示 的初始化向量,下图右侧的示例表示,该存储器地址 0 内的数据为 "23f4",地址 1 内的数据为"0721",以此类推。本例中的初始化 文件内容为:

memory_initialization_radix=16; memory_initialization_vector=7 5 3 6 4 1 2 0 3 1 6;

nitialization
et by using a COE file. This will be passed to the core F).
07. srcs/sources_1/ip. coe
Value
16
75364120316

An example COE file:

memory_initialization_radix = 16; memory_initialization_vector = 23f4 0721 11ff ABe1 0001 1 0A 0 23f4 0721 11ff ABe1 0001 1 0A 0 23f4 721 11ff ABe1 0001 1 A 0 23f4 721 11ff ABe1 0001 1 A 0;

例化完成后,可以在"工程路径\工程名.ip_user_files\ip\

dist_mem_gen_0\"目录下找到例化的文件,我们可以像调用其它模块一样调用该模块,编写仿真文件,对该 RAM 进行仿真,仿真文件代

码为:

```
`timescale Ins / Ips
module tb( );
reg clk;
reg [3:0] a, dpra;
reg [2:0] d;
reg we;
wire [2:0] dpo;
initial
begin
    c1k = 0:
    forever
    #5 \ clk = clk:
end
initial
begin
    a = 0;dpra=0;d=0;we=0;
    #20
    repeat (5)
    begin
        @(posedge clk); #1;
        dpra = dpra +1;
    end
    repeat (10)
    begin
        @(posedge c1k); #1;
        a = $random%16;
        dpra = $random%16;
        d = \$random\%8;
        we = $random%2;
    end
        @(posedge c1k); #1;
        a = 0;
        dpra = 0;
        d = 0;
        we = 0;
        #20 $stop;
end
dist_mem_gen_0 dist_mem_gen_0(
```

. a	(a),
. d	(d),
. dpra	(dpra),
.clk	(c1k),
.We	(we),
. dpo	(dpo));
endmodule	

其仿真波形如下图所示,可以看到,5 号地址的初始值为1,与 coe 文件相一致,在225ns 时钟的上升沿处,通过写端口将该地址改 写成了5,因此在236ns 处读端口从该地读取到的是改写后的数值。

Name	Value		40 ns		I	60 n	s,		80 n:	s		1	100 1	۱S ,		1	20 ns			140 1	ns ,		160	ns			180 ns
🖩 📲 dpra[3:0]	0	<u>1</u>	2 X	3	×	4	5		1	ż	d		<u> </u>		ΞX	a	Ξx	f	Ċ		5	ХÈ	0	ż	3	ΧŻ	0
⊞-₩ dpo[2:0]	2	5	з X	6	X	4	X 1		5	X			0			6	_X	0	X		5	ХE	7		6	X	2
u clk	1															1											
We we	0													L													
🖩 📲 a[3:0]	0			0				X	4	X	d)	\square	1	d		5	X	2	Х	8)	<u> </u>	X	0	X	6	X	0
⊡ - %] d[2:0]	0			0				X	1	X	5	\square	6	1		5		2	X	4	3	X	2	X	5	Х—	0

除了时钟管理单元和存储单元外,Vivado 中还提供了许多功能的 IP 核,在 IP 核参数设置界面提供了说明文档的连接,读者可选择自 己感兴趣的功能模块自行学习。

实验练习

题目1. 例化一个16*8bit 的 ROM,并对其进行初始化,输入端口由4个开关控制,输出端口连接到七段数码管上(七段数码管与 LED 复用相同的一组管脚),控制数码管显示与开关相对应的十六进制数字,例如四个开关输入全为零时,数码管显示"0",输入全为1时,数码管显示"F"。

题目2. 采用8个开关作为输入,两个十六进制数码管作为输出,采 用时分复用的方式将开关的十六进制数值在两个数码管上显示出来, 例如高四位全为1,低四位全为0时,数码管显示"F0"。

题目3. 利用本实验中的时钟管理单元或周期脉冲技术,设计一个精

度为0.1秒的计时器,用4位数码管显示出来,数码管从高到低,分 别表示分钟、秒钟十位、秒钟个位、十分之一秒,该计时器具有复位 功能(可采用按键或开关作为复位信号),复位时计数值为1234, 即1分23.4秒

总结与思考

1. 请总结本次实验的收获

- 2. 请评价本次实验的难易程度
- 3. 请评价本次实验的任务量
- 4. 请为本次实验提供改进建议