## 1 FPGA 在线调试说明

在 FPGA 开发过程中,经常碰到仿真上板行为不一致,更多时候是仿真正常,上板异常。由于上板调试手段 薄弱,导致很难定位错误。这时候可以借助 Xilinx 的下载线进行在线调试,在线调试是在 FPGA 上运行过程中探 测预定好的信号,然后通过 USB 编程线缆显示到上位机上。

本文档给出简单使用在线调试的方法: RTL 里设定需探测的信号,综合并建立 Debug core,实现并生产 bit 流 文件,下载 bit 流和 debug 文件,上板观察。

#### 1.1 抓取需探测的信号

```
在 RTL 源码中,给想要抓取的探测信号声明前增加(*mark_debug = "true"*)。
```

比如,我们想要在 FPGA 板上观察 debug 信号、PC 寄存器和数码管寄存器,需要在代码里这样设置。

```
//debug interface
  (*mark_debug = "true"*)output [31:0]
                                       debug_wb_pc;
                                                                        inst_ex_dib;
                                                            wire
  (*mark_debug = "true"*)output [3 :0]
                                        debug_wb_rf_wen;
                                                            (*mark_debug = "true"*)reg [31:0] inst_pc_r;
  (*mark_debug = "true"*)output [4 :0]
                                       debug_wb_rf_wnum;
                                                            reg [31:0] inst_code_r;
  (*mark_debug = "true"*)output [31:0]
                                        debug_wb_rf_wdata;
                                                                         (2) 抓取 PC 寄存器
                   (1) 抓取写回信息
                                    reg [JI.0] red rgo data
                                    reg [31:0] led_rg1_data;
                            5
                                 (*mark_debug = "true"*) reg [31:0] num_data;
                            6
                                    wire [31:0] switch_data;
                            7
                                    wire [31:0] htn key data-
                                           (3) 抓取数码管寄存器
    设定完成后,就可以运行综合了。
1.2 综合并建立 debug
    在综合完成后, 需建立 debug。
    点击工程左侧 synthesis->Open Synthesized Desgin->Set Up Debug。
                                        SYNTHESIS
                                           Run Synthesis
                                           Open Synthesized Design
                                              Constraints Wizard
                                              Edit Timing Constraints
                                           🕷 <u>Set Up Debug</u>
                                           🖸 Report Timing Summary
                                              Report Clock Networks
                                              Report Clock Interaction
                                           Report Methodology
```

随后会出现如下界面,点击 Next:



选择抓取的深度和触发控制类型,点击 Next(更高级的调试可以勾选"Advanced trigger"):

A Set Up Debug	23	
ILA Core Options Choose features for the ILA debug cores.	4	
Sample of data depth: 1024 <mark>选择抓取的深度</mark> Input pipe stages: 0 v		
Trigger and Storage Settings	-	
✓ Capture control 远律肥友突空, 習可勾远另一个即可 Advanced triager		
?      Next >     Einish     Cancel	el	

最后,点击 Finish:



# 1.3 实现并生产 bit 流文件

在 1.2 节完成后会出现类似下图界面,直接点击 Generate Bitstream:

Flow Navigator	SYNTHE SIZED DE SIGN * - synth_1   xc7a200tfbg676-2 (ar	ctive)						
	Sources Netlist × ?		Schemat	tic x co	onfreg.v	× soc axi lite to	p.v × Is132r top.v	×
Run Simulation	<u>х</u> н	ð	<b>.</b>	<b>•</b> •	- 	S A La		ells 55 I/O Por
	> JF data_sram_addr (32)	^	•   ,					
RIL ANALYSIS     Open Eleberated Design	> J∱ data_sram_wdata (32)							
✓ Open Elaborated Design	> J dcr_reqbus (66)							
Report Methodology	> Jr m debug_wb_pc (32) > Jr m debug wb f wdata (32)					<u>0</u>		
Report DRC		~						
Report Noise	Bus Net Properties ? _ □	$\square \times$						
Schematic	师** debug_wb_rf_wdata           🗧	•						
✓ SYNTHESIS	General Scalar Nets	Q.						
Run Synthesis				_				
> Open Synthesized Design	Tcl Console Messages Log Reports Des	ign Runs	Debug	×				
	Q   ≚   ≑   兼   <b>+</b>   ➡							
✓ IMPLEMENTATION	Name	Driver	Driver	Probe Typ	e			
Run Implementation	✓ ¥i abg_nub (labtools_xsabm_v3) ✓ ₩i uia 0 (labtools_ila_v6)							
> Open Implemented Design	> > clk (1)							
	> 📄 probe0 (32)			Data and	Trigger 🚿	v		
	> 🝺 probe1 (32)			Data and	Trigger 🚿	¥		
	Dehun Cores Dehun Nots			Data and I	Triooers	e		
<b>弹出如下界面</b> , 占击	f Save:		//					
		$\mathbf{A}^{\prime}$				~		
	🝌 Save Project					×		
	Save project before gene Data to Save	erating b	itstrear	n?				
						_		
	✓ Synthesized Design - co	onstrs_1	- SOC_	lite.xdc				
17	S <u>a</u> ve	Do	on't Sav	e	Cano	cel		
加果有后续弹虫界面	1. 继续占击 OK 戓 Vee 即可	2时前;	井λ⋿	遠生さ	ב ב hit ל	 大件的流程 <sup>-</sup>	了。此时 Viva	do界面甲
如不有但实理山尔田 的 armthasis 1 甲五		그만에 가지지 그는 미	山日田	「次工」 	いしく	「日田坦女	」, JULIET VIV& タウナレニ 12	40 75 囲王
市 syntnesis design 齐国, 录深度即可:	<u>帆町以大内</u> 丁。 如未 <b>反</b> 现以下锥	厌, 贝	」定囚	八哈住	:人休,	,「用起米	<b>石</b> 子 从 て , 庠	瓜工作日
	Reports Design Runs							2 5 5
		0 (44E)		tuo (60)	01			
└└└ ≍ │ ╤ │ ╹╷ ╠ │ Ш │ ╚	🗸 🖶 Entro (19) 🔄 🚽 Warning (328) 🗌 🚺 Inf	0 (413)	U Sta	ius (08)	Show A	<b>MI</b>		¢

E:/loongson/arcn_ucas/1/-18/labb/ucas_CDE_axi_v0.1/ucas_CDE_axi_v0.3_ret/mycpu_axi_venty/run_vivado/mycpu_prj1/mycpu.sim/sim_1/benav/elaborate.log file for more information.
Vivado 12-4473] Detected error while running simulation. Please correct the issue and retry this operation.
🗡 Թ Implementation (2 errors)
Y Թ Opt Design (2 errors)
<ul> <li>[Chipscope 16-302] Could not generate core for dbg_hub. Aborting IP Generation operation. The current Vivado temporary directory path,</li> <li>[E:/loongson/arch_ucas/17-18/lab6/ucas_CDE_axi_v0.1/ucas_CDE_axi_v0.3_ref/mycpu_axi_verify/run_vivado/mycpu_prj1/mycpu.runs/impl_1/.Xii/Vivado-10160-PC-2016022800</li> <li>44', is 163 characters. Errors on the host OS will occur when trying to insert logic for debug core 'dbg_hub' when temporary directory paths exceed 146 characters. Please move this Vivado project or the Vivado working directory to a shorter path; alternately consider using the OS subst command to map part of the path to a drive letter.</li> <li>[Chipscope 16-307] Implementing debug core dbg_hub failed due to earlier errors.</li> </ul>
• • • • • • • • • • • • • • • • • • •

## **1.4** 下载 bit 流和 debug 文件

在完成 1.3 节后,会生成 bit 流文件和调试使用 ltx 文件。这里,打开 Open Hardware Manager,连接好 FPGA 开发板后,选择 Program device,如下图。自动加载了 bit 流文件和调试的 ltx 文件。选择 Program,等待下载完成。

НА	RDWARE MANAGER - loca	lhost/xilinx_tcf/Xilinx/Port_#0003.Hub_#0003
0	There are no debug cores	. <u>Program device</u> Refresh device
e Device Pronertigs	Hardware Program Device Select a bitstream prog select a debug probes t programming file.	? _ ा स x ramming file and download it to your hardware device. You can optionally file that corresponds to the debug cores contained in the bitstream
Hardwar	Bitstre <u>a</u> m file: Debu <u>q</u> probes file: ✓ <u>E</u> nable end of st	_verify/run_vivado/mycpu_prj1/mycpu.runs/impl_1/soc_axi_lite_top.bit verify/run_vivado/mycpu_prj1/mycpu.runs/impl_1/soc_axi_lite_top.ltx  artup check
	?	Program Cancel

## 1.5 上板观察

在下载完成后, vivado 界面如下, 在线调试就是在 hw\_ila\_1 界面里进行的。

HARDWARE MANAGER - localhost/xilinx_tcf/Xilinx/Port_#0003	Hub_#0003	? >
Hardware ? _ D Ľ ×	hw_ila_1	? 🗆 🖒 X
Q   ≚   ≑   ∅   ▶   ≫   ■   ♦	Waveform - hw_ila_1	? _ 🗆 X
Name Status		+F   F=   **
✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓		
<ul> <li>v (1) xc7a200t_0 (2)</li> <li>Programme</li> </ul>	ILA Status: Idle	i i
XADC (System Monitor)	ا Name Value 0 ا	3
the isolation of the is		
Hardware Device Properties ? _ D 🗅 X		>
(t) xc7a200t_0 ←   →   ♦	Settings - Status × ? _ D Trigger Setup - hw_ila_1 × Capture Setup - hw_ila_1	? _ □
Name: xc7a200t_0	♂     >     >	
Part: xc7a200t		
ID code: 13636093	Core status Press the + button to add probes.	
General Properties	VVattr ~	
Tcl Console × Messages Serial I/O Links Seria	I/O Scans	? _ 🗆 🖸
Q ≚ ≑ II ₪ 面		
INFO: [Labtools 27-2302] Device xc7a200t (JTAG devi	ce index = $0$ ) is programmed with a design that has 1 ILA core(s).	^
☐ refresh hw device: Time (s): cnn = 00:00:08 · elans	ad = 00:00:09 Memory (MB): nesk = 1805 602 : gain = 0.000	> ×
1 - 1 -		

hw\_ila\_1 界面主要分为3个界面,分别如下:

hw	_ila_1		? 凸 岱 >
tions	Waveform - hw_ila_1 Q   +   −   &   ▶   ≫   ■   B   @   Q	波形观察区域 ☆ ●「 【◀ ▶】 セ� ●「 F# ●「 ▶□	? _ □ × ¢
Dashboard Opt	ILA Status:Idle       Name     Value       > M u_confreg/num_data[31:0]     0       > M u_cpu/debug_wb_pc_1[0:0]     0       > M u_la_0_debug_wb_pc_1[1:1]     0       > M u lia_0_debug_wb_pc[31:3]     0	1  2  3  4  5  6 	7 8 9
	Settings - hw_ila_1 Status - hw_ila_1 × ひ ▶ ≫ ■ 9a Idle Waiting for Trigger Post-Trigger ure status 状态显示区域,当前状态是Idle indow 1 of 1 Window sample 0 of 1024 Total sample Idle Idle Idle	? _ □ Trigger Setup - hw_ila_1 × Capture Setup - hw_ila_1 Q   +   -   ⊅,   Press the + button to add probes.	? _ 🗆

首先,我们需要在右下角区域设定触发条件。所谓触发条件,就是设定该条件满足时获取波形,比如我先设定触发条件是数码管寄存器到达 0x0500\_0005。在下图中,先点击"+",在双击 num\_data。



6

可以设定多个触发条件,比如,再加一个除法条件是写回使能是 0xf,可以设定多个触发条件直接的关系,比 如是任意一个满足、两个都是满足等等,如下图:

Trig	Trigger Setup - hw_ila_1 × Capture Setup - hw_ila_1														
Q															
Nar	~	Set Trigger Condition to Global AND'		Radix		Value		Port							
u_0		Set Trigger Condition to 'Global OR'	-	[H]	~	0500_0005		probe0[31:0]							
u_c		Set Trigger Condition to Global NAND'		[H]	Ŧ	F	Ŧ	probe4[3:0]							
		Set Trigger Condition to 'Global NOR'													
			,												

在左下角窗口,选择 settings,可以设定 Capture 选项,可能经常用到的是 Trigger position in window,用来设 定触发条件满足的时刻在波形窗口的位置。比如,下图设定为 500,当触发条件满足时,波形窗口的第 500 个 clk 的位置是该条件,言下之意,将触发条件满足前的 500 个 clk 的信号值也抓出来了,这样可以看到触发条件之前的 电路行为。Refresh rate 设定了波形窗口的刷新频率。

Settings - hw_ila_1 × Status -	hw_ila_1 ? _		Trigger Setup - hw_ila_1 × Captu
Capture Mode Settings		^	Q + - D
Capture mode:	ALWAYS 🗸		Name
Number of windows:	1 [1 - 1024]		u_confreg/num_data[31:0]
Window data depth:	<b>1024</b> • [1 - 1024]		u_cpu/debug_wb_rf_wen[3:0]
Trigger position in window:	500 [0-1023] 这里设定捕获时	刻	
General Settings Refresh rate: 500	ms	Ļ	

触发条件建立后,就可以启动波形抓取了,最关键的有三个触发按键,即下图圈出的3个按键:

- 左起第一个,设定触发模式,有两个选项:单触发;循环触发。当该按键按下时,表示循环检测触发,那么只要触发条件满足,波形窗口就会更新。当设置为单触发时,就是触发一次完成后,就不会再检测 触发条件了。比如,如果我们设定触发条件是 PC=0xbfc00690,那么如果该 PC 被多次执行到。如果设定 为单触发,那按下 FPGA 板上的复位键,波形窗口只会展示第一次触发时的情况。如果设定循环触发,那么波形窗口会以 Refresh rate 不停刷新新捕获的触发条件。
- 左起第二个,等待触发条件被满足。点击该按键,就是等待除法条件被满足,展示出波形。

● 左起第三个,立即触发。点击该按键,表示不管触发条件,立即抓取一段波形展示到窗口中。

下图就是点击第三个按键得到的波形,因为是立即触发,所以 num\_data 不是 0x0500\_0005,且有一条标注为 "T"的红色,就是触发的时刻。由于触发时刻位于波形窗口的 500 clk 位置,所以红色的位置正好是 500 clk 处。

Waveform - hw_ila_1															
Q   <b>+</b>   <b>−</b> 😻 🕨 ≫ 🔳	🛃 🛛 🛛	Q,	20 +		H	12   2	r   +F	<b>F</b> ⊭	•ľ   ⊡						
ILA Status:Idle						529									
Name	Value		200		400		600		800	1	L, 000	1,200	1,400	1,600	<b></b> 1,
> Value = 10 - 10 - 10 - 10 - 10 - 10 - 10 - 10	5e00005					5e0000	)\$e								
> 📢 u_cpu/debug_wb_pc_1[0:0]	0					0									
> 🔣 u_ila_0_debug_wb_pc_1[1:1]	0					0									
> 🍣 u_cpu/debug_wb_pc[2:2]	1	XXX						XXXIXX							
> 🔣 u_ila_0_debug_wb_pc[31:3]	17f80020			0.010		XO									
> 🔣 u_ila_0_debug_wb_rf_wnum[4:0]	00	Ж	0.00	0		0 0									
> 🔣 u_cpu/debug_wb_rf_wdata[31:0]	0000000	ЮК	0.00												
> 📢 u_cpu/debug_wb_rf_wen[3:0]	0			000		X	00								
		Upda	ted at: 201	17-Dec-0	6 16:06:	20				·					

从上图也能看到, num\_data 是 0x5c00\_005c, 表示一次测试已经完成了。所以这时候, 点击第二个触发按键等 待触发, 会发现波形窗口没有反应。这是因为触发条件没有被满足, 这时按下 FPGA 板上的复位键即可。结果如 下图。红色圈出的就是触发条件: num\_data==32'h5c00\_005c && rf\_wen==4'hf。

Waveform - hw_ila_1																		? _	×
Q   +   <b>-</b>   ♂   ▶   ≫   ■	🕒 🔍	Q,	2018	F I I	н	12	±r   +Γ	<b>Fe</b>   ⇒Γ	<b> </b>										ø
ILA Status:Waiting For Trigger (500 out of 1	024 sample															51 9			í
Name	Value			485		490		495		500		505	510	515			520	!	525
> 😼 u_confreg/num_data[31:0]	0500000		04	000004								05000005	5						
> 🔣 u_cpu/debug_wb_pc_1[0:0]	0										0								
> Mainto u_ila_0_debug_wb_pc_1[1:1]	0										0								
> 🔣 u_cpu/debug_wb_pc[2:2]	0	0		1			0										0		
> 📢 u_ila_0_debug_wb_pc[31:3]	17f800e1		17f84	e 315					1 f84	e Bo			17f800e0	$\square$	17	f800	e1	17f80	19d
> 🦋 u_ila_0_debug_wb_rf_wnum[4:0]	00	08						00					1f	$\square$		00		X 05	
> nu_cpu/debug_wb_rf_wdata[31:0]	0000000	050	X					00000000					bfc0070c	$\square$	00	0000	00	X fffff	ff4
> 🔏 u_cpu/debug_wb_rf_wen[3:0]	0					0				X£X,		0		Х	_X			XIX	0
		Updated	1 at: 2(	)17-Dec-06	6 16:10:	: 30													

剩下的 debug 过程,就和仿真 debug 类似了,去观察波形。但在线调试时,你无法添加在 1.1 节未被添加 debug mark 的信号。在线调试过程中,可能需要不停的更换触发条件,不停的按复位键。

### 1.6 注意事项

在 1.1 节中添加要抓取的信号时,不要给太多信号标注 debug mark 了。在线调试时抓取波形是需要消耗电路 资源和存储单元的,因而能抓取的波形大小是受限的。应当只给必要的 debug 信号添加 debug mark。

在 1.2 节中抓取波形的深度,不宜太深。如果设定得太深,那么会存在存储资源不够,导致最后生成 bit 流和 debug 文件失败。

抓取的信号数量和抓取的深度是一对矛盾的变量。如果抓取深度相对较低,抓取的信号数量就可以相对多些。

在 1.5 节中,相对仿真调试,在线调试对调试思想和技巧有更高的要求,请好好整理思路,多多总结技巧。 特别强调以下几点:

- 触发条件的设定有很多组合,请根据需求认真考虑,好好设计。
- 通常只需要使用单触发模式,但循环触发有时候也很有用,必要时好好利用。

● 在线调试界面里很多按键,请自行学习,可以网上搜索资料,xilinx 官网上搜索,查找官方文档等。

最后再提醒一点,仿真通过但上板失败时,**请先重点排查其他问题,最后再使用在线调试的方法**。也就是仿 真通过,上板异常时,应按以下流程排查:

- 1) 复核生成、下载的 bit 文件是否正确。
- 2) 复核仿真结果是否正确。
- 3) 检查实现时的时序报告(Vivado 界面左侧 "IMPLEMENTATION" "Open Implemented Design" "Report Timing Summary")。
- 4) 认真排查综合和实现时的 Warning
- 5) 排查 RTL 代码规范, 避免多驱动、阻塞赋值乱用。
- 6) 使用 Vivado 的逻辑分析仪进行板上在线调试