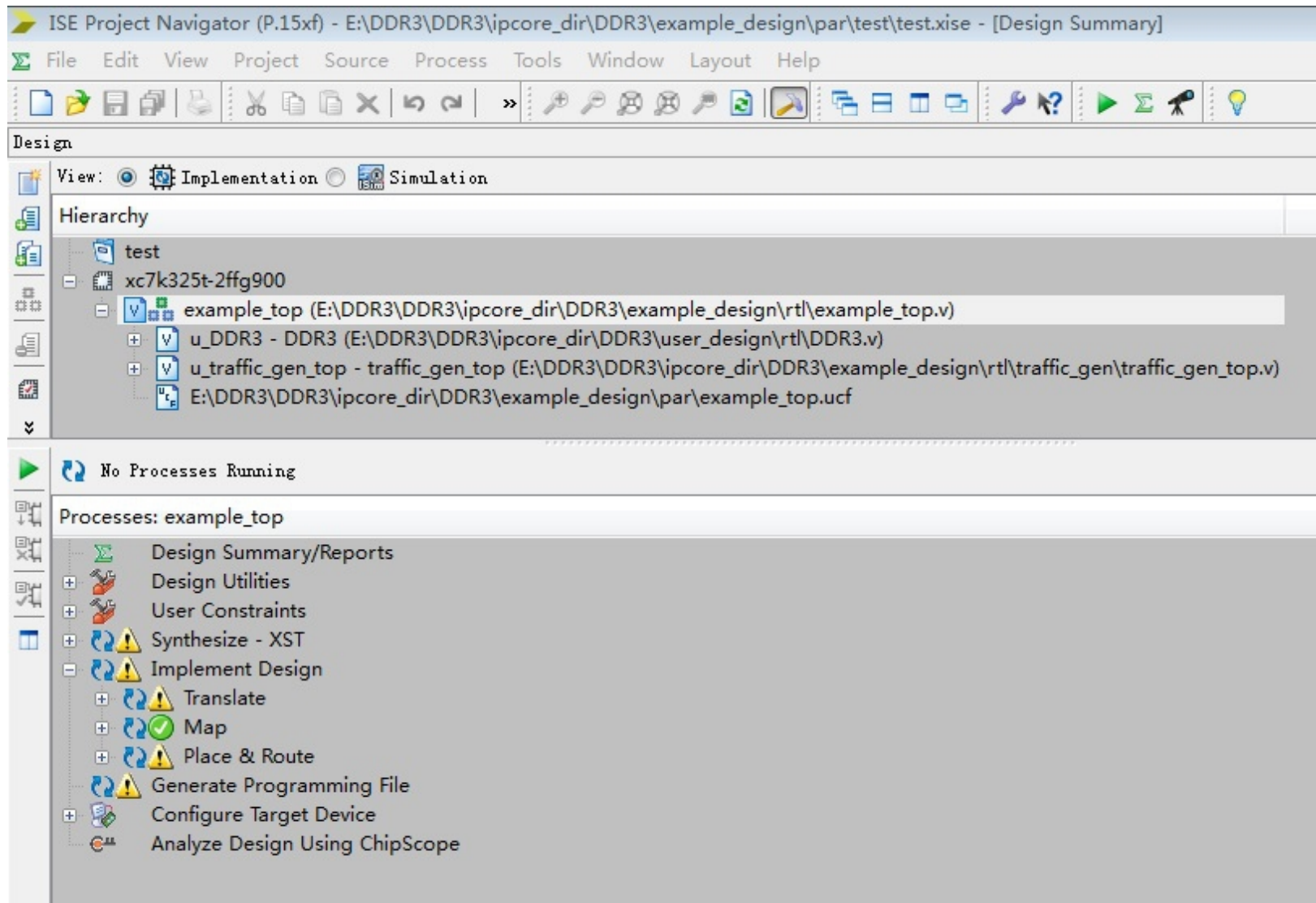


你已经看过我写的仿真教程，并且按照步骤都做成功了吧？  
现在要做综合？来个一图流就行了——看明白了吗？编译已经成功，bit文件生成了。  
什么？没看明白？下面分开讲吧~





之前仿真教程里面讲过过`traffic_gen`，现在派用场了

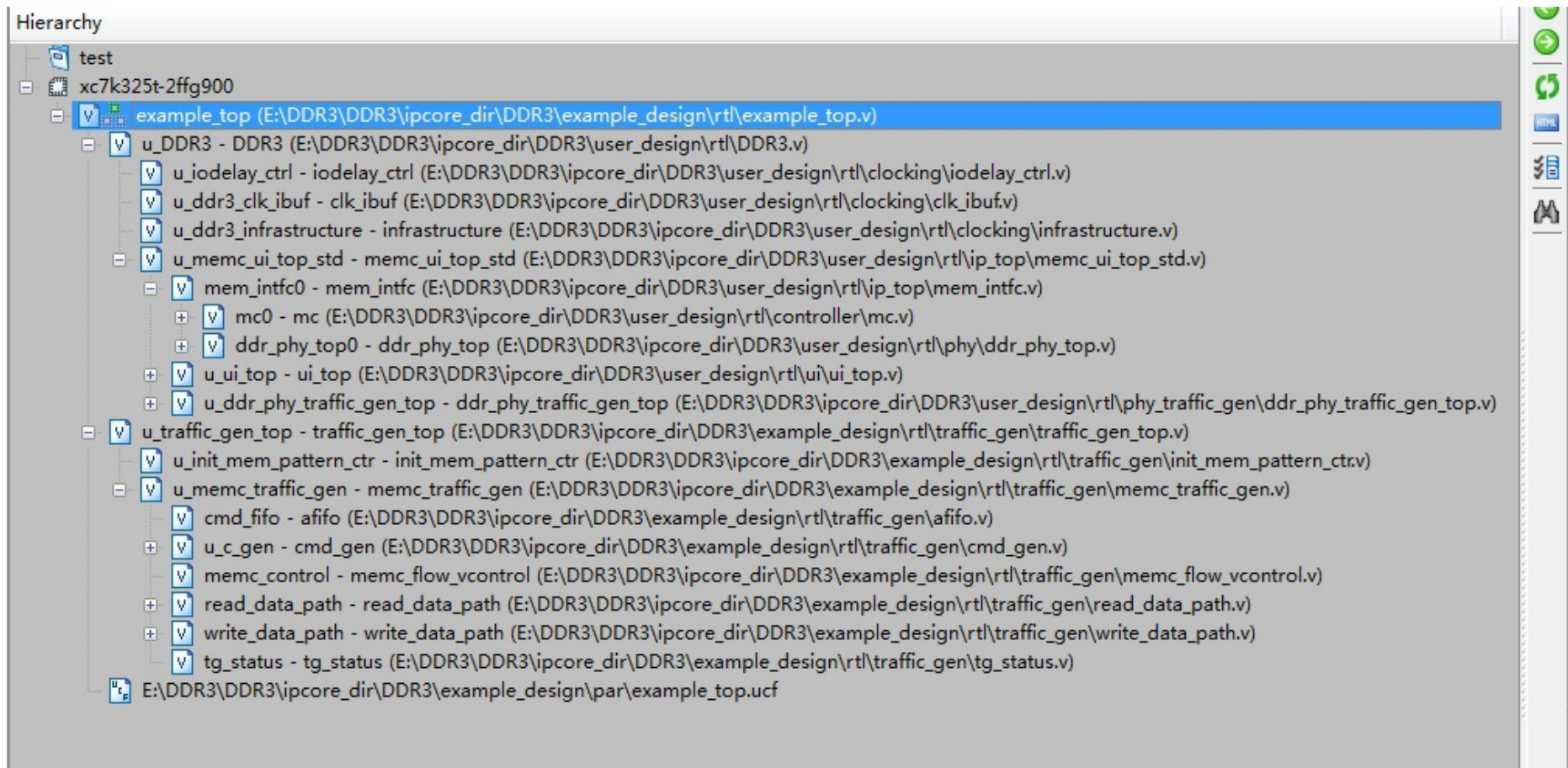
看这个工程里面的顶层文件，不是我写的  
其实这工程里面所有的文件都不是我写的

只要你生成好了IP core，管脚分配好（你也可以让core gen自动分配，用默认值就行，还是参见仿真教程）

你要做的事，就是这么区区几步：

- 1，建立一个ISE工程，FPGA型号和封装选对就可以了
- 2，加入`example_design/rtl`下面的 `example_top.v` 以及其他所有.v文件  
（什么，你写VHDL的？乖乖，不知道现在写VHDL的很难找工作？）
- 3，加入`user_design/rtl`下面的rtl下面的所有.v文件
- 4，加入`example_design/par` 下面的`example_top.ucf`文件

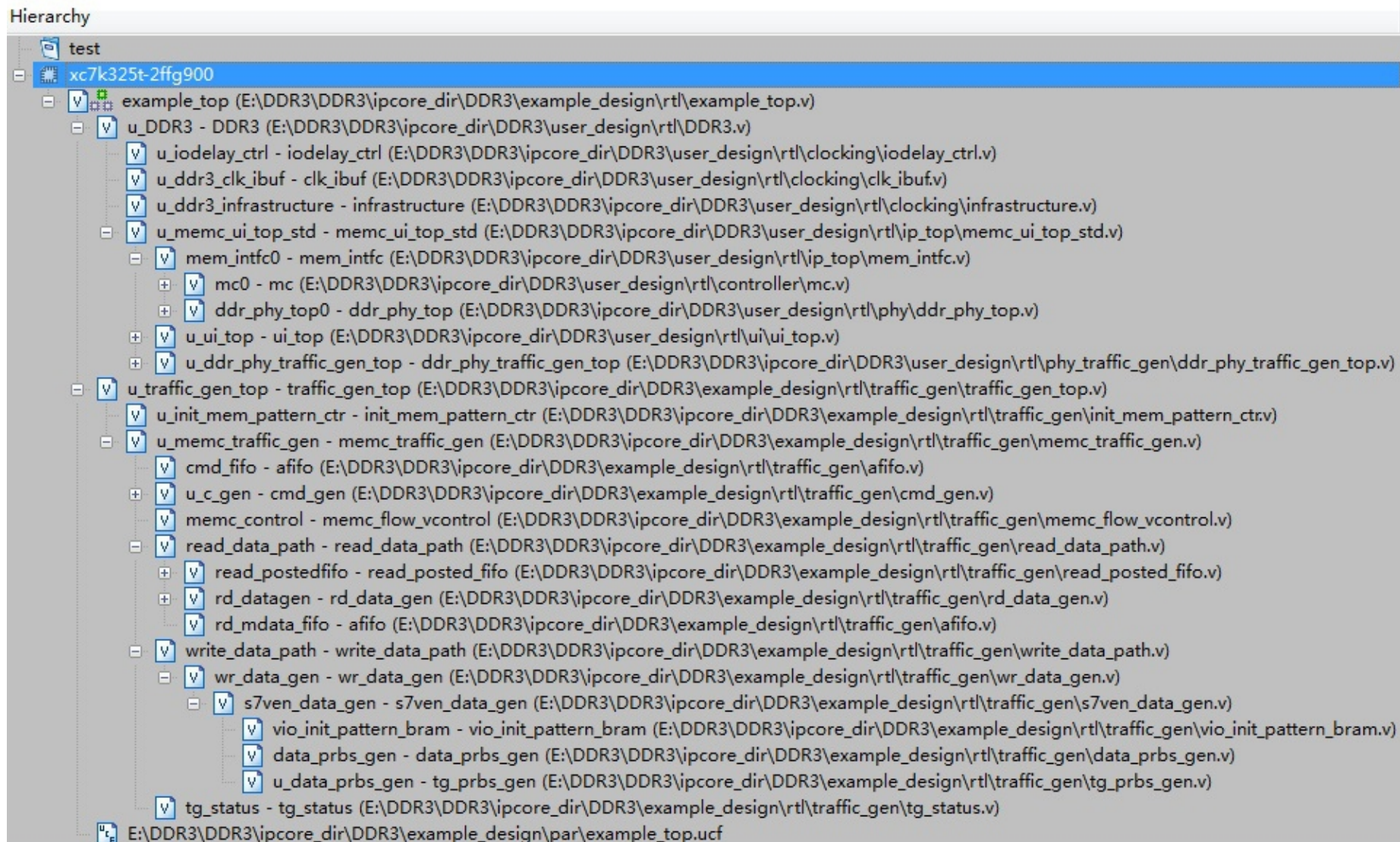
然后你就开始跑编译吧~



文件全部加完了之后就是这样的，都是绝对路径显示



多展开几级，来个壮观点儿的



E:\DDR3\DDR3\ipcore_dir\DDR3\example_design\rtl					
(E) 查看(V) 工具(T) 帮助(H)					
包含到库中 ▾ 共享 ▾ 新建文件夹					
	名称	修改日期	类型	大小	
	traffic_gen	2013/10/30 22:17	文件夹		
	example_top.v	2013/10/30 22:17	V 文件	45 KB	

E:\DDR3\DDR3\ipcore_dir\DDR3\user_design\rtl					
(E) 查看(V) 工具(T) 帮助(H)					
包含到库中 ▾ 共享 ▾ 新建文件夹					
	名称	修改日期	类型	大小	
	clocking	2013/10/30 22:17	文件夹		
	controller	2013/10/30 22:19	文件夹		
	ecc	2013/10/30 22:17	文件夹		
	ip_top	2013/10/30 22:17	文件夹		
	phy	2013/10/30 22:17	文件夹		
	phy_traffic_gen	2013/10/30 22:19	文件夹		
	ui	2013/10/30 22:17	文件夹		
	DDR3.v	2013/10/30 22:17	V 文件	48 KB	

其实就是把这两个目录下的\*.v都加进去，最后别忘了加上  
example\_design/par 下面的example\_top.ucf文件，就可以了

只要你在core gen的时候分配好了管脚，就不可能编译不成功。

这张图还记得吗？

万一做板子的人把按键和LED灯的管脚给你分配到了这里不能选的地方，你就得留意下一页的内容了。

**REFERENCE DESIGN**

**System Signals Selection**

Select the system pins below appropriately for the interface. Customization of these pins can also be made in the UCF after the design is generated. For more information see [UG586 Bank and Pin rules](#).

	Signal Name	Bank Number	Pin Number
1	sys_clk_p/n	33	AE10/AF10 (CC_F/N)

**Reference Clock Pin Selection**

The **clk\_ref** input is used as the reference clock for the IODELAY. Refer the "7 Series FPGA SelectIO Resources User Guide" for more information. This input can be generated internally or can be connected to an external clock source on a clock capable differential (P/N) pair.

	Signal Name	Bank Number	Pin Number
1	clk_ref_p/n	Select Bank	No connect

**Status Signals**

These signals may be connected internally to other logic or brought out to a pin.

- **sys\_rst**: This input signal is used to reset the interface.
- **init\_calib\_complete**: This signal indicates that the interface has completed calibration and memory initialization and is ready for commands. LOC constraint will be generated in UCF for Example design only based on "Pin Number" selection below.
- **error**: This output signal indicates that the traffic generator in the Example Design has detected a data mismatch. This signal does not exist in the User Design.

	Signal Name	Bank Number	Pin Number
1	sys_rst	Select Bank	No connect
2	init_calib_complete	Select Bank	No connect
3	tg_compare_error	Select Bank	No connect

按键复位，初始化成功和错误指示灯  
这里并不是所有空闲管脚都可以选的

All pins must be constrained to specific locations in order to generate a bit file in the implementation phase (this is not required for simulation).

**XILINX**

User Guide Version Info < Back Next > Cancel



在上面这个文档，也就是xilinx MIG的用户手册里面的132页，有这么一段：

## I/O Standards

These rules apply to the I/O standard selection for DDR3 SDRAMs:

- Designs generated by the MIG tool use the SSTL15\_T\_DCI and DIFF\_SSTL15\_T\_DCI standards for all bidirectional I/O (DQ, DQS) in the High-Performance banks. In the High-Range banks, the tool uses the SSTL15 and DIFF\_SSTL15 standard with the internal termination (IN\_TERM) attribute chosen in the GUI.
- The SSTL15 and DIFF\_SSTL15 standards are used for unidirectional outputs, such as control/address, and forward memory clocks.
- LVCMOS15 is used for the RESET\_N signal driven to the DDR3 memory.

The MIG tool creates the UCF using the appropriate standard based on input from the GUI.

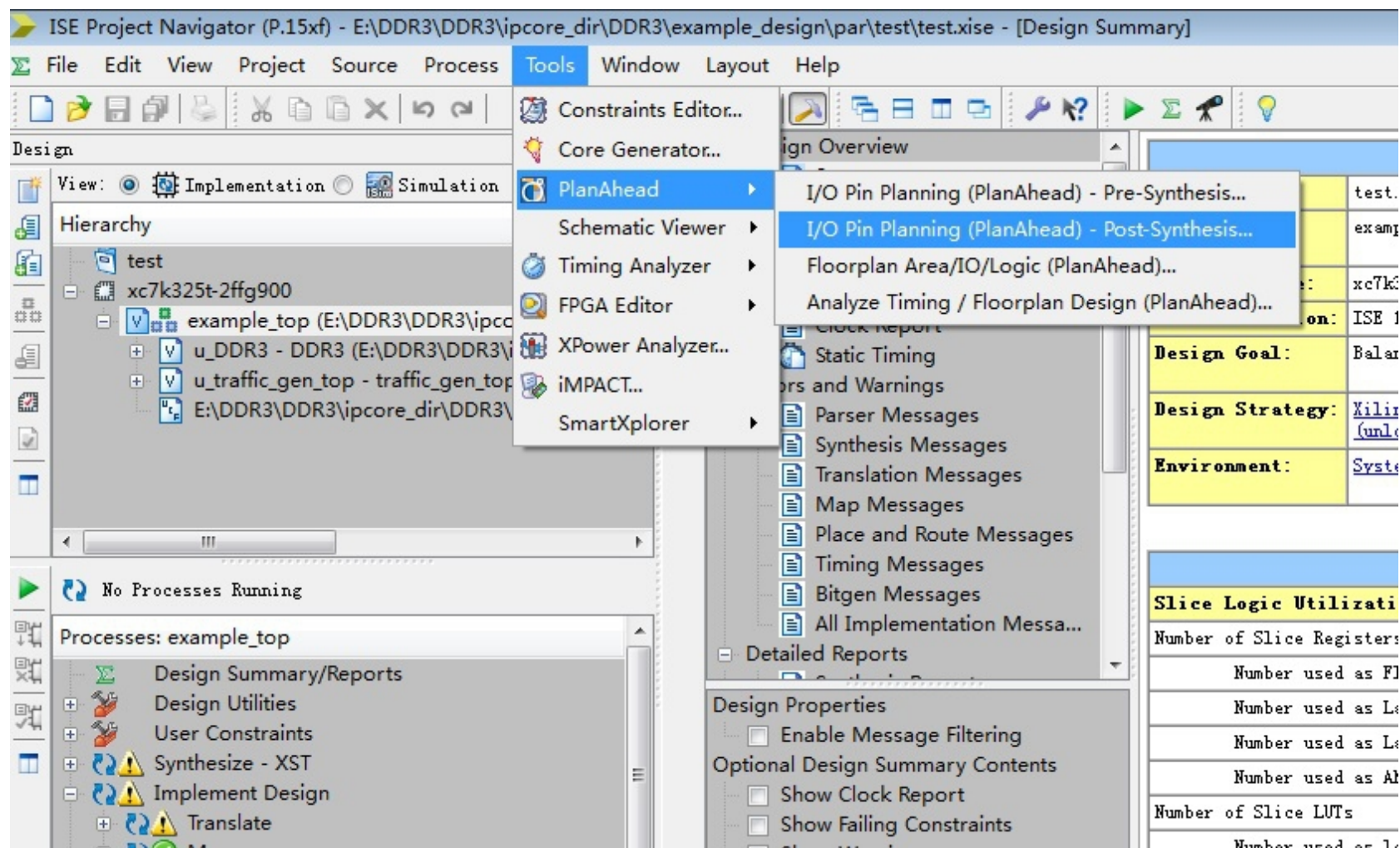
简单说一下要点：

数据管脚应该是SSTL15系列的电平，而复位管脚应该是LVCMOS15系列的电平

其中15表示1.5V

管脚电平可以用planAhead来设置。

推荐你综合完了之后用post-synthesis（综合后）的选项来重设管脚属性，  
这个比pre-synthesis（综合前）有说服力，行不行还是得看你综合完了之后不是？





Direction: Output  
Site: AF11 ☒ Fixed  
Package Pin: AF11

General | Configure | Attributes | Power

Properties | Clock Regions

### I/O Ports

Name	Direction	Neg Diff Pair	Site	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type	Pull Type	Off-Chip Termination	IN_TERM
ddr3_dqs_p (16)	In/Out	ddr3_dqs_n				DIFF_SSIL135...	1.35			FAST*	NONE		
ddr3_odt (1)	Output				33	SSIL135*	1.35	0.675		FAST*	NONE		
Scalar ports (11)													
clk_ref_p	Input	clk_ref_n	AB27	<input checked="" type="checkbox"/>	13	LVDS_25*					NONE		
ddr3_cas_n	Output		AF11	<input checked="" type="checkbox"/>	33	SSIL135*	1.35	0.675		FAST*	NONE		
ddr3_ras_n	Output		AE11	<input checked="" type="checkbox"/>	33	SSIL135*	1.35	0.675		FAST*	NONE		
ddr3_reset_n	Output		AK3	<input checked="" type="checkbox"/>	34	SSIL135*	1.35	0.675		FAST*	NONE		
ddr3_we_n	Output		AD12	<input checked="" type="checkbox"/>	33	SSIL135*	1.35	0.675		FAST*	NONE		
init_calib...	Output		C19	<input checked="" type="checkbox"/>	17	LVC MOS25*	2.5		12	SLOW	NONE	FP_VII_50	
sys_clk_p	Input	sys_clk_n	AE10	<input checked="" type="checkbox"/>	33	DIFF_SSIL135*				SLOW	NONE		NONE
sys_rst	Input		C17	<input checked="" type="checkbox"/>	17	LVC MOS25*	2.5		12	SLOW	NONE	NONE	
tg_compare...	Output		C20	<input checked="" type="checkbox"/>	17	LVC MOS25*	2.5		12	SLOW	NONE	FP_VII_50	

Icl Console | Package Pins | I/O Ports

但是实际上你打开了之后会发现不是那么回事儿  
这为啥LVCMOS都是2.5V，SSTL都是1.35V，编译也通过了？  
可能待会儿上电了下载还正常跑。

那我告诉你，这个例子里SSTL都是1.35V，跟core gen的时候选的sodimm条子款式有关，有些条子就是1.35V的，比如我这里选的这根儿就是。

电压或许可以和1.5V有所区别，但是SSTL和LVCMOS的区别不能搞混，否则可能map不过去。

SSTL和LVCMOS有啥区别的呢

SSTL是针对DDR之类的高速管脚的电平

LVCMOS就是常用的默认电平

只要知道，DDR的高速打数据，尤其是双沿动作的管脚，都得是SSTL的电平；而按键指示灯之类的慢速控制管脚，应该是lvcmos电平，就可以了。

如果你在planAhead里面选择好管脚属性保存之后没有立即提示错误，也不代表你后面map和par就能通过。

DDR设计的原则之一就是，别去改哪些你想当然认为肯定没事的东西。

先全部用默认值生成一个DDR工程，编译通过，保存好，然后再在这个工程里改动。在编译出问题的时候，仔细核对你改动的部分和原先的初始工程有什么区别，项目才能比较顺利的做下去。

最后大致讲一下traffic gen

这个貌似DDR2里面没有  
只有DDR3有

功能之前讲过了，就是写数据进DDR，读出来看和写进去的是不是一致，不一致的话就报错，拉高tg\_compare\_error

很不幸的是，这东西虽然用了DDR core用户接口，但是代码写得基本没有通用性，你不可能在这个基础上修改修改就拿来自己用。

所以要做DDR设计，你得自己重新写一个控制DDR用户接口的代码。

最典型的例子就是从一个fifo里面读数据，然后写进DDR

这个例子我之后再讲了。

现在，先动手建一个工程吧。

在ISE14.2和以上版本，可以直接进example\_design\par文件夹，运行create\_ise.bat，然后工程会自动生成。

其实这个过程和你手动建立工程是一样一样的，聊胜于无吧，嘿嘿~