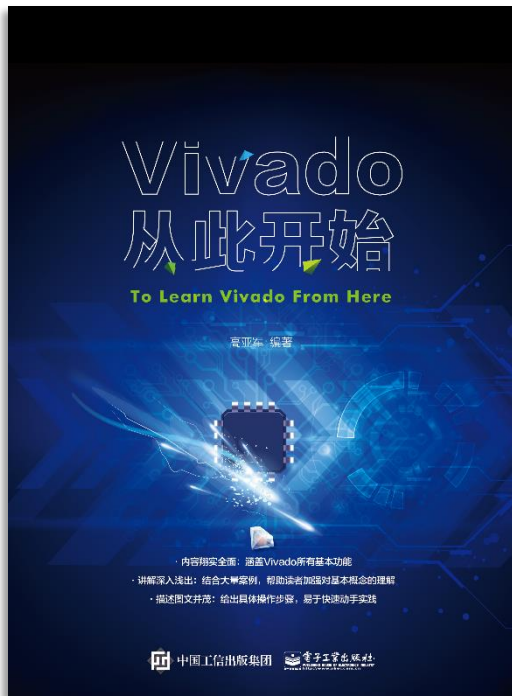


# Vivado从此开始 ( To Learn Vivado From Here )



## 本书围绕Vivado四大主题

- 设计流程
- 时序约束
- 时序分析
- Tcl脚本的使用



作者：高亚军（Xilinx战略应用高级工程师）

- 2012年2月，出版《基于FPGA的数字信号处理（第1版）》
- 2012年9月，发布网络视频课程《Vivado入门与提高》
- 2015年7月，出版《基于FPGA的数字信号处理（第2版）》
- 2016年7月，发布网络视频课程《跟Xilinx SAE学HLS》

- ◆ 内容翔实全面：涵盖Vivado所有基本功能
- ◆ 讲解深入浅出：结合大量案例，帮助读者加强对基本概念的理解
- ◆ 描述图文并茂：给出具体操作步骤，易于快速动手实践



**XILINX**

ALL PROGRAMMABLE™

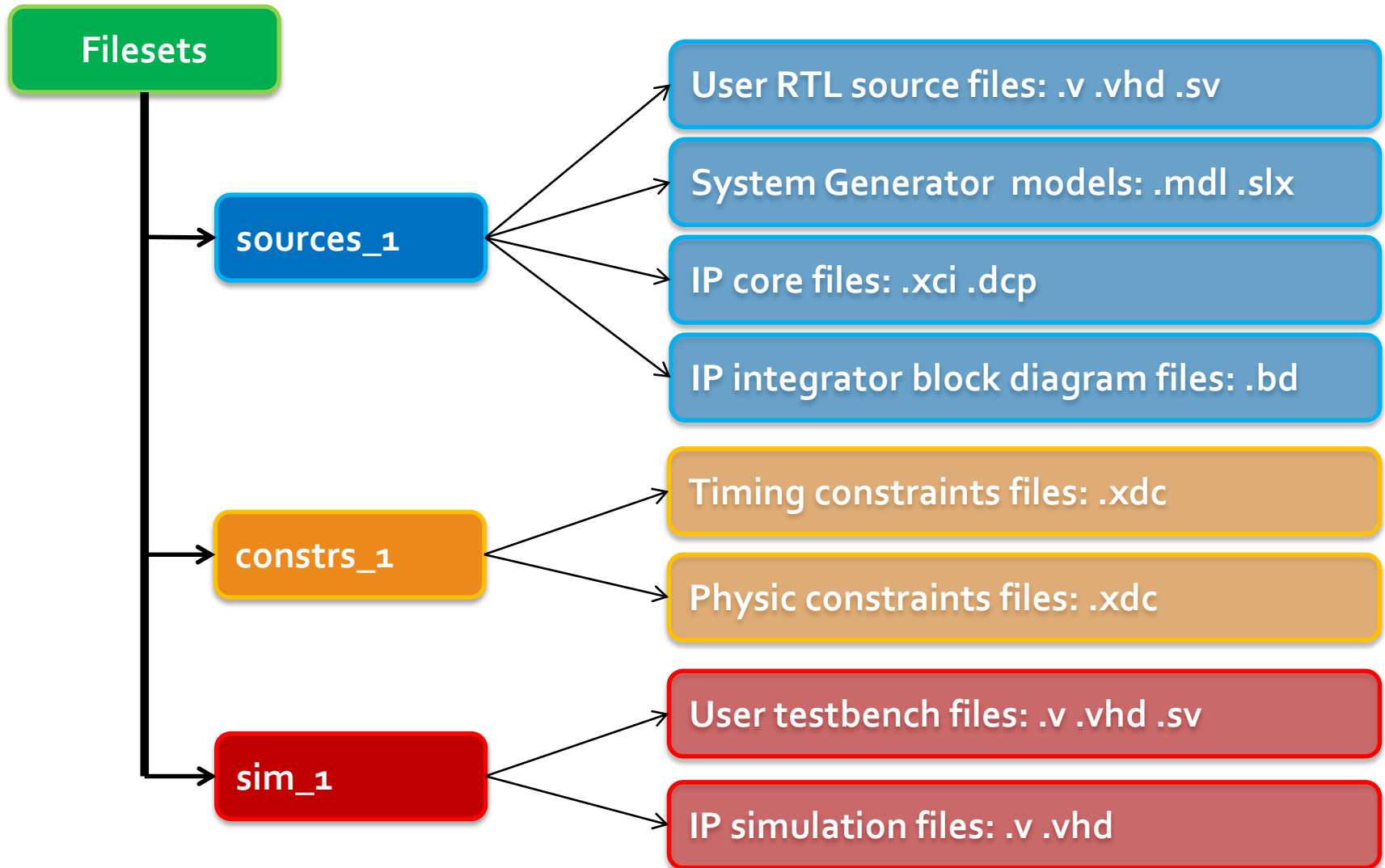
## **Some Tips About Vivado Design Flow**

**Lauren Gao**

# Agenda

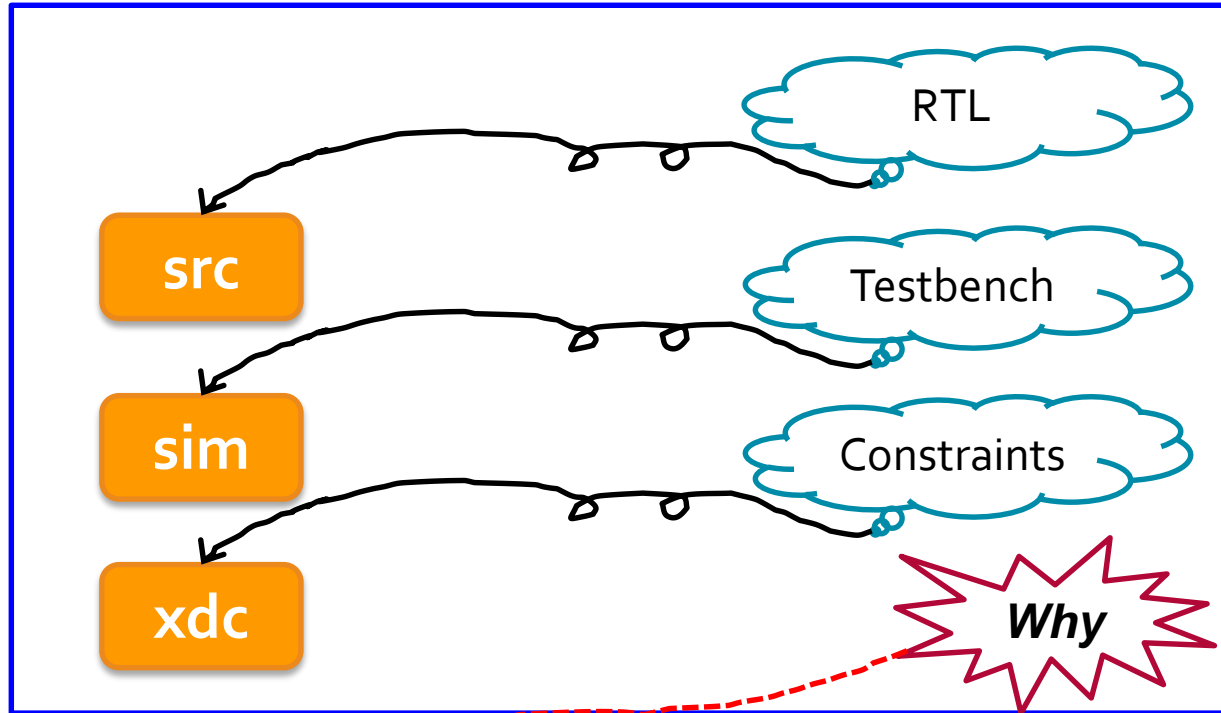
- Tips of user design source files management
- Tips of IP management
- Tips hardware management

# Design Files Architecture in Vivado



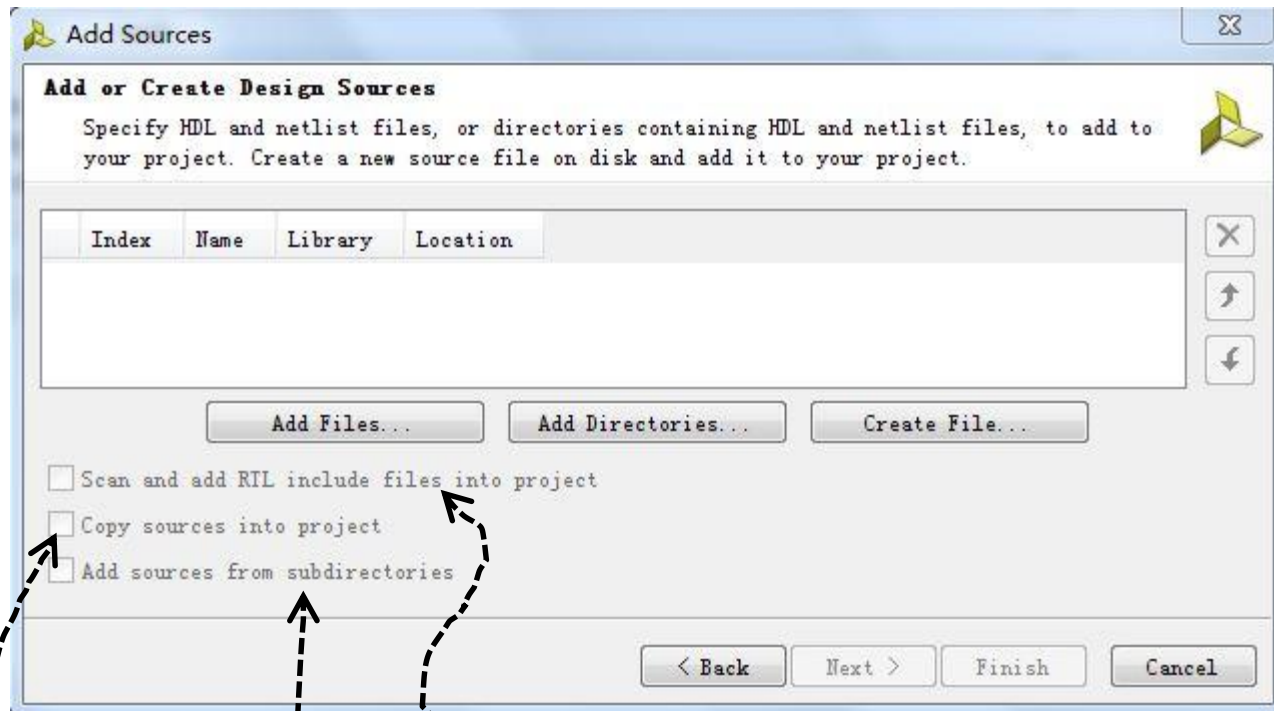
# User Design Source Files Management

- Put different type of files into different folders



- Add design files to the project easily and quickly
- Add the sources to the project as referenced sources instead of importing them into the project

# Add User Design Files to Project Efficiently



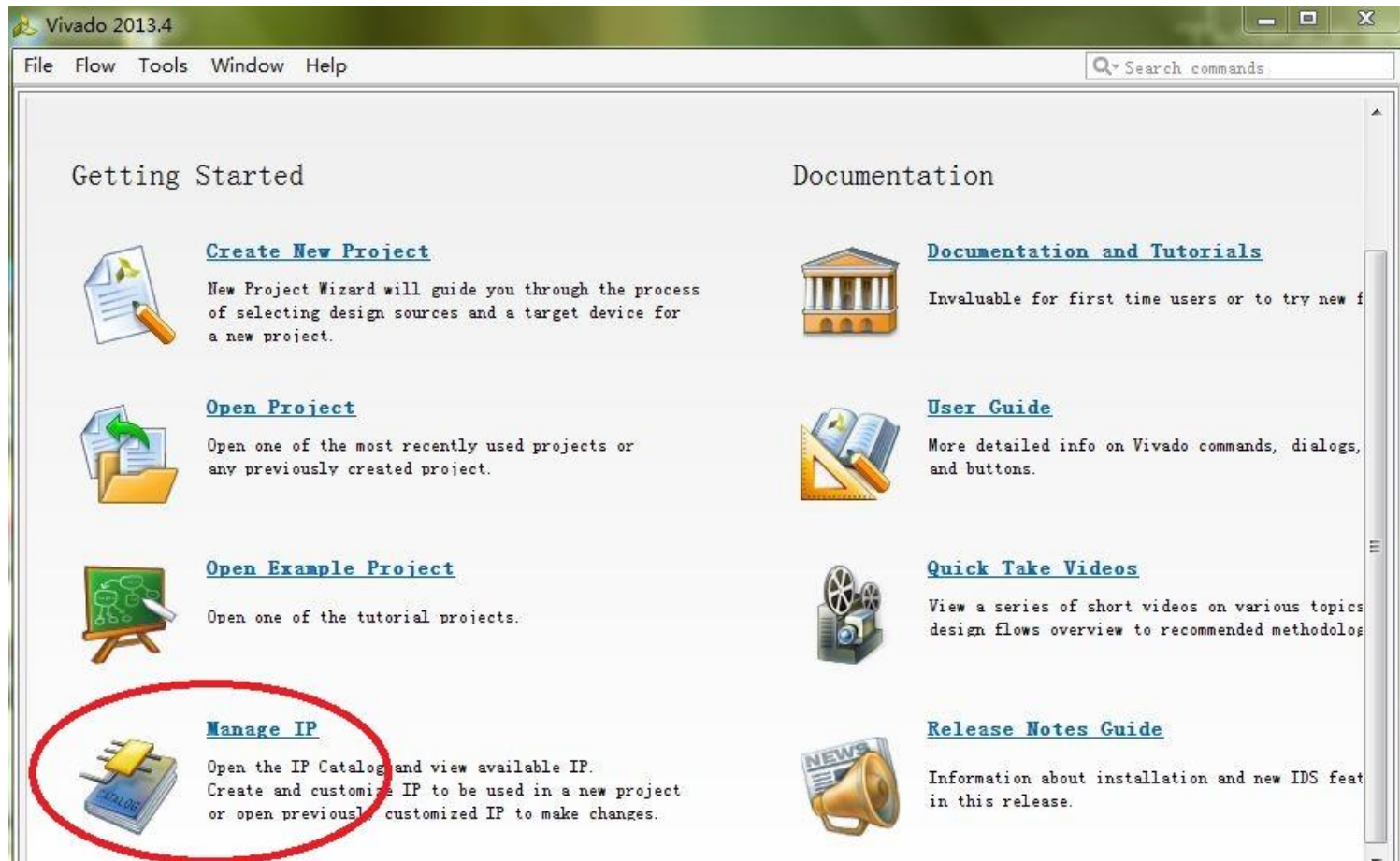
```
add_files -scan_for_includes F:/Vivado/ug937/sources
```

```
add_files -scan_for_includes F:/Vivado/ug937/sources
```

```
import_files F:/Vivado/ug937/sources
```

```
add_files -norecurse F:/Vivado/ug937/sources
```

# IP Management



- *It makes revision control more straightforward*
- *It allows for ease of sharing customized IP with others*

# Manage IP in managed\_ip\_project

## ➤ Check IP status

```
report_ip_status -name ip_status_1
```

– Upgrade?

- IP version has changed

```
upgrade_ip [get_ips sine_high]
```

– Is locked?

- Design part has changed

## ➤ Get PART & TARGET\_LANGUAGE of current project

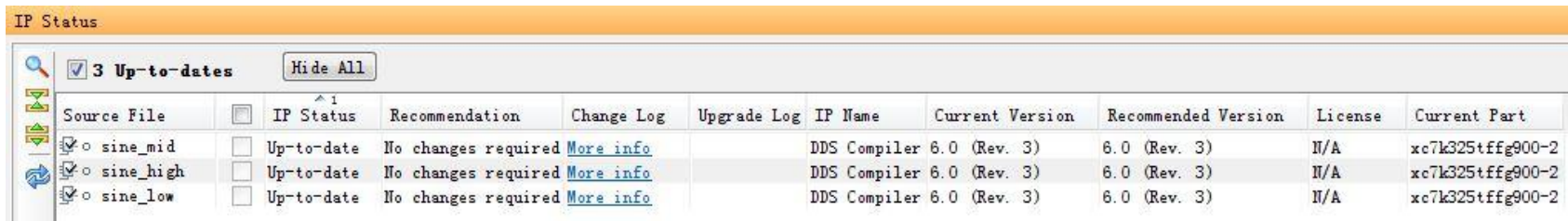
```
get_property PART [current_project]  
get_property TARGET_LANGUAGE [current_project]
```

## ➤ Change PART & TARGET\_LANGUAGE of current project

```
set_property PART xc7k325tffg900-2 [current_project]  
set_property TARGET_LANGUAGE VHDL [current_project]
```



# If PART Is Changed...

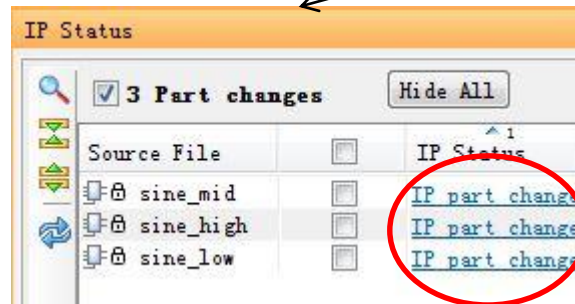


IP Status

3 Up-to-dates Hide All

Source File	IP Status	Recommendation	Change Log	Upgrade Log	IP Name	Current Version	Recommended Version	License	Current Part
sine_mid	Up-to-date	No changes required	<a href="#">More info</a>		DDS Compiler 6.0 (Rev. 3)	6.0 (Rev. 3)	6.0 (Rev. 3)	II/A	xc7k325tffg900-2
sine_high	Up-to-date	No changes required	<a href="#">More info</a>		DDS Compiler 6.0 (Rev. 3)	6.0 (Rev. 3)	6.0 (Rev. 3)	II/A	xc7k325tffg900-2
sine_low	Up-to-date	No changes required	<a href="#">More info</a>		DDS Compiler 6.0 (Rev. 3)	6.0 (Rev. 3)	6.0 (Rev. 3)	II/A	xc7k325tffg900-2

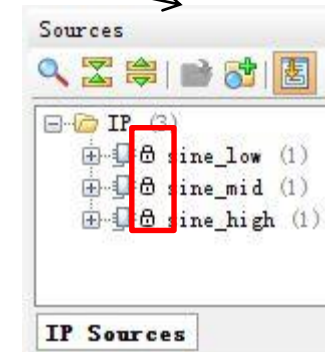
After PART changed



IP Status

3 Part changes Hide All

Source File	IP Status
sine_mid	IP part change
sine_high	IP part change
sine_low	IP part change



Sources

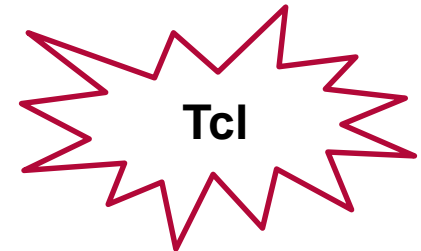
IP Sources

sine_low (1)
sine_mid (1)
sine_high (1)

Reset all IPs → Upgrade all IPs → Regenerate all Ips

One by one? far too inefficient !

Do it with Tcl script automatically!



# Reset and Regenerate all Ips with Tcl

```
01 # Author: Lauren Gao
02 # Date: 2014-2-24
03 # delete target ip
04 # Example
05 # set myips [get_ips]
06 # reset_ips $myips
07 proc reset_ips {myips} {
08     foreach ip $myips {
09         reset_target all $ip
10         delete_ip_run $ip
11     }
12 }
```

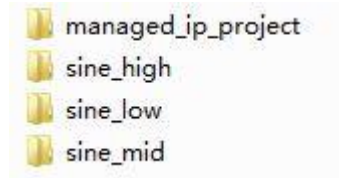
```
01 #Author: Lauren Gao
02 #Date: 2014-2-24
03 #Note: regenerate target IPs
04 #Exampe
05 #set myips [get_ips]
06 #regen_ips $myips
07 proc regen_ips {myips} {
08     foreach ip $myips {
09         upgrade_ip [get_ips $ip]
10         set ipname [get_property NAME [get_ips $ip]]
11         generate_target all $ip
12         create_ip_run $ip
13         launch_run ${ipname}_synth_1
14         wait_on_run ${ipname}_synth_1
15     }
16 }
```

*What's the function of **upgrade\_ip** here?*

# Add IPs to the Project

## ➤ Manage IP project structure

- Each IP has its own directory



## ➤ Which type of IP generated files can be added to the project

- xci (Xilinx Core Instance)
- dcp (Design Checkpoint)

## ➤ How to add xci or dcp files to current project efficiently

- Find xci one by one?
- **Tcl**

## ➤ What's the difference between adding xci and adding dcp to current project

## ➤ Add the sources to the project as referenced sources instead of importing them into the project

# Project Management

## ➤ Design Files

- User design source files: Verilog, VHDL, SystemVerilog, XDC
- IP files: xci, dcp

## ➤ Project properties

Property	Type	Read-only	Value
BOARD	string	false	xilinx.com:kintex7:kc705:1.1
CLASS	string	true	project
COMPXLIB.COMPILED_LIBRARY_DIR	string	false	F:/Vivado/manage_ip/sine_demo.cache/compile_simlib
COMPXLIB.EDK.EXCLUDE_SUB_LIBS	bool	false	0
COMPXLIB.EDK.EXCLUDE_SUPERSEDED_CORES	bool	false	1
COMPXLIB.EDK.PREVIOUS_LIB_PATH	string	false	
COMPXLIB.EDK.SOURCE_LIB	string	false	
COMPXLIB.EDKLIB	bool	false	0
COMPXLIB.FUNCSIM	bool	false	1
COMPXLIB.OVERWRITE_LIBS	bool	false	0
COMPXLIB.TIMESIM	bool	false	1
COMPXLIB.XILINXCORELIB	bool	false	1
DIRECTORY	string	false	F:/Vivado/manage_ip
IS_HD	bool	false	0
IS_READONLY	bool	false	0
MANAGED_IP	bool	false	0
NAME	string	true	sine_demo
PART	part	false	xc7k325tffg900-2
SIMULATOR_LANGUAGE	enum	false	Mixed
SOURCE_MGMT_MODE	enum	false	All
TARGET_LANGUAGE	enum	false	Verilog
TARGET_SIMULATOR	enum	false	XSim

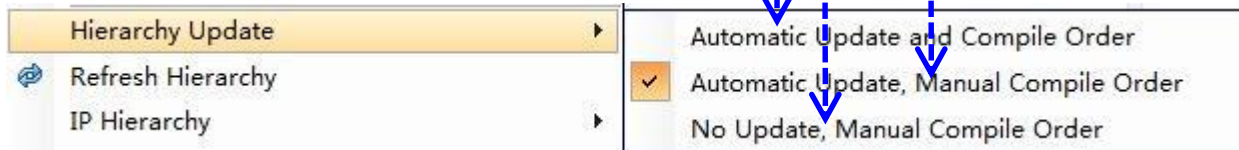
# Change Project Properties

## ➤ PART & TARGET\_LANGUAGE

```
set_property PART xc7k325tffg900-2 [current_project]
set_property TARGET_LANGUAGE VHDL [current_project]
```

## ➤ SOURCE\_MGMT\_MODE

```
set_property source_mgmt_mode All [current_project]
set_property source_mgmt_mode DisplayOnly [current_project]
set_property source_mgmt_mode None [current_project]
```



# Popular Tcl Command: `get_files`

- `get_files [-regexp] [-nocase] [-filter arg] [-of_objects args]`  
`[-compile_order arg] [-used_in arg] [-all] [-quiet] [-verbose] [patterns]`
- `-of_objects`
  - A single fileset
  - A sub-design: IP core, Block design or DSP design
- `-compile_order`
  - sources | constraints
- `-used_in`
  - synthesis, simulation, implementation
  - This option must be used with the `-compile_order` option

# Examples of get\_files

- Get all RTL files in sources\_1 fileset

```
set rtl_files [get_files -of_objects [get_filesets sources_1]]
llength $rtl_files
```

- Get all VHDL files in sources\_1 fileset

```
set vhd_files [get_files -of_objects [get_filesets sources_1]\
-filter {NAME =~ *.vhd}]
llength $vhd_files
```

- Get all simulation files of an IP core

```
set sim_files [get_files -of_objects [get_ips sine_high]\
-compile_order sources -used_in simulation]
llength $sim_files
```

- Get all constraint files of an IP core?

# Check Compile Order

- `report_compile_order [-of_objects args] [-fileset arg]`  
`[-missing_instances] [-constraints] [-sources] [-used_in arg]`  
`[-file arg] [-append] [-quiet] [-verbose]`

➤ Example :

- Check constraints compile order

```
report_compile_order -constraints
```

- Check simulation compile order

```
report_compile_order -used_in simulation
```

- Check synthesis compile order?



# Three Types of Design in Vivado (Project Mode)

Elaborated Design

```
synth_design -rtl -name rtl_1
```

Synthesized Design

```
open_run synth_1 -name netlist_1
```

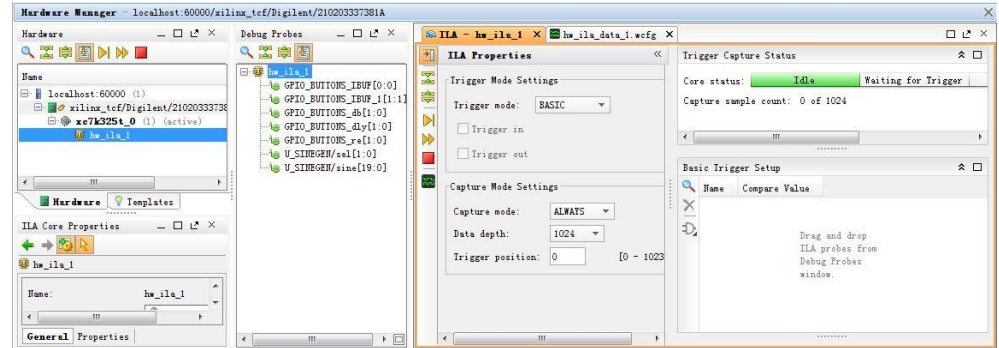
Implemented Design

```
open_run impl_1
```

# Hardware Manager: .bit & .ltx

- Two files are necessary
  - ✓ .bit
  - ✓ debug\_nets.ltx: probes information files
  - ✓ .ltx can be generated by write\_debug\_probes Tcl command

```
set_property PROGRAM.FILE {C:/design.bit} [lindex [get_hw_devices] 0]  
set_property PROBES.FILE {C:/design.ltx} [lindex [get_hw_devices] 0]
```



# Save and Restore ILA Data

## ➤ Saving Captured ILA Data to a File

```
write_hw_ila_data my_hw_ila_data_file.zip [upload_hw_ila_data hw_ila_1]
```

## ➤ Restoring Captured ILA Data from a File

```
display_hw_ila_data [read_hw_ila_data my_hw_ila_data_file.zip]
```