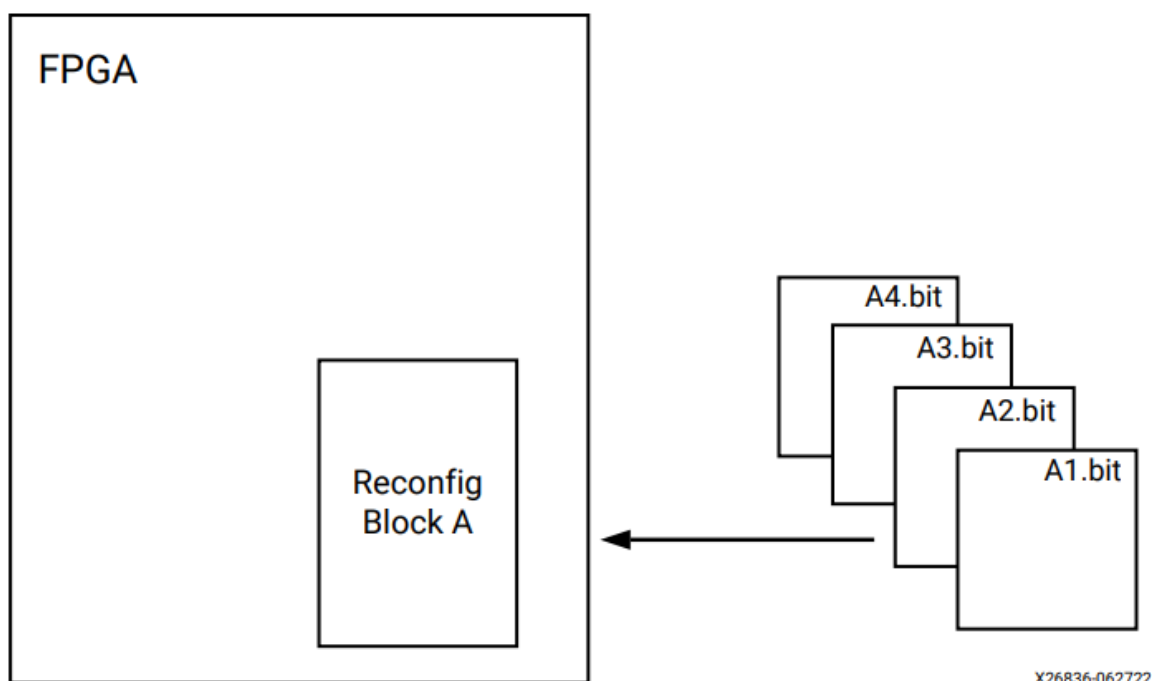


# Dynamic Function eXchange

从Vivado2019.1开始，Xilinx为之前的Partial Reconfiguration（部分重配置）功能取了个新名字--Dynamic Function eXchange（动态功能交换），在Vivado ML 2021.1, DFX是一项基本特性，不再和之前版本里的PR功能一样需要特定License。

Xilinx认为：DFX 技术允许设计者动态修改功能性，无需全面重配置和重建链路，极大地增强了 FPGA 提供的灵活性；它是平台化设计流程的关键功能，对于Alveo 加速卡而言，尤为明显。使用 DFX 有助于设计者转而采用更少或更小的器件，降低功耗并提高系统升级能力。随时按需加载功能，更有效利用芯片。现在DFX也是高层次设计特性的一部分，并列于HLS, IPI, Model composel。

Xilinx官方文档：ug909



非项目模式：

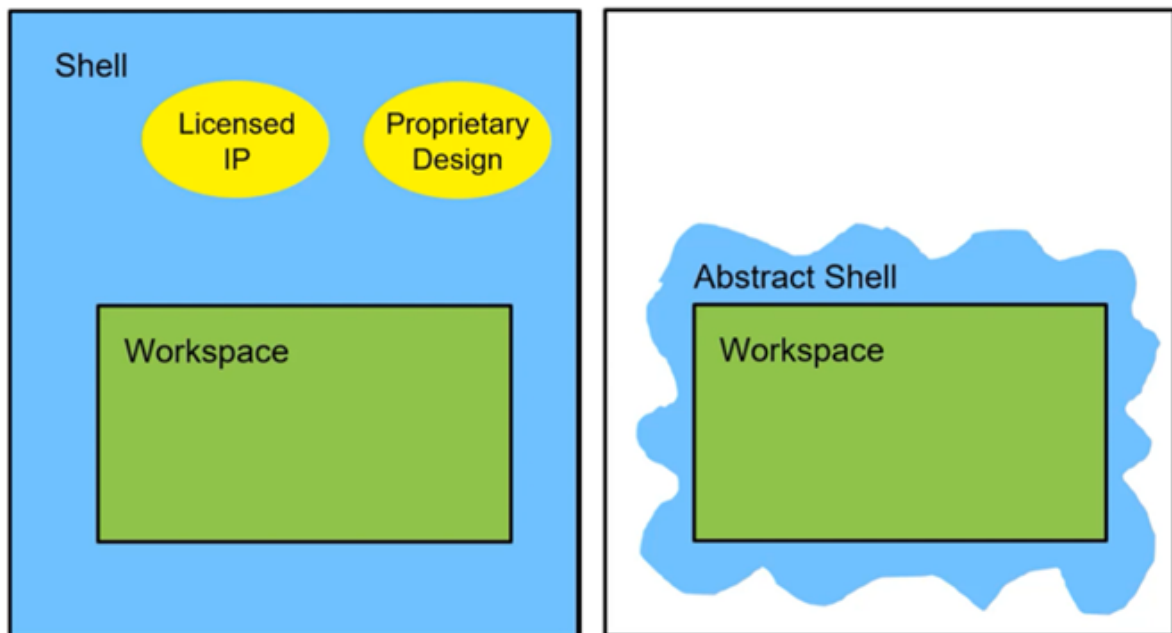
软件流程实现。使用TCL脚本命令完成整体工程搭建与重构设计。

Abstract Shell设计。

项目模式：

- 1、基于RTL的项目设计流程（与低版本PR设计流程一致）
- 2、bd容器设计流程。在Block design中划分模块作为重构部分，进行层次化设计，在子bd中设计重构。

### 3、Abstract Shell设计。



在标准DFX设计流程内，重构区RP的每一个重构模块RM编译时都需要提供完整的锁定的静态区image文件。这样的流程需要Vivado较长的编译时间。

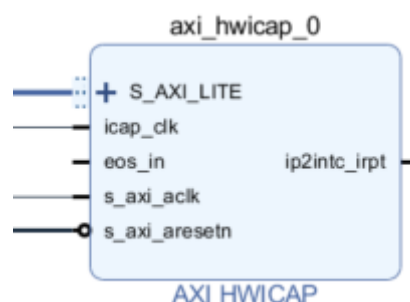
在Abstract Shell模式下的DFX设计流程中，编译不再需要提供完整的静态image文件，使用Abstract Shell文件（后续对新RM进行编译并生成相应部分bit文件所需的静态区image文件最小版本，包含RP的约束信息（Pblock）、布线信息、边界的时序信息）。

## 项目模式动态重构流程

参考网站：[https://blog.csdn.net/weixin\\_43189165/article/details/106647004](https://blog.csdn.net/weixin_43189165/article/details/106647004)

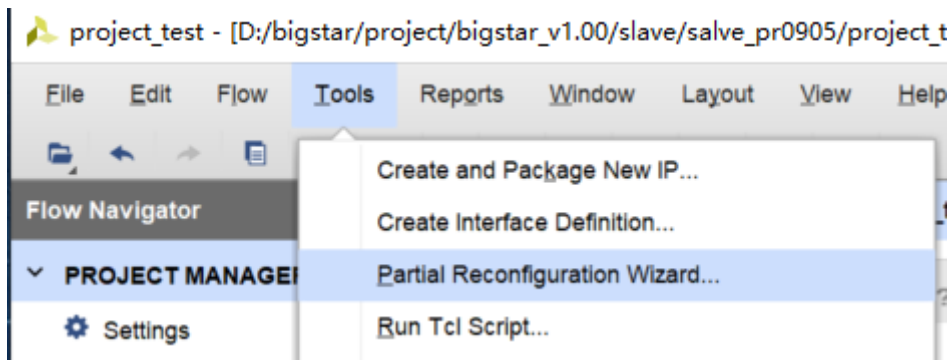
参考2：<https://zhuanlan.zhihu.com/p/147651921>

在功能电路上划分动态区，并在该动态区内设计不同功能的模块，综合布线实现后，会生成对应动态区的多个重构bit，加载对应bit即可实现FPGA局部的可重构。通过AXI HWICAP接口实现重构bit数据的传输。

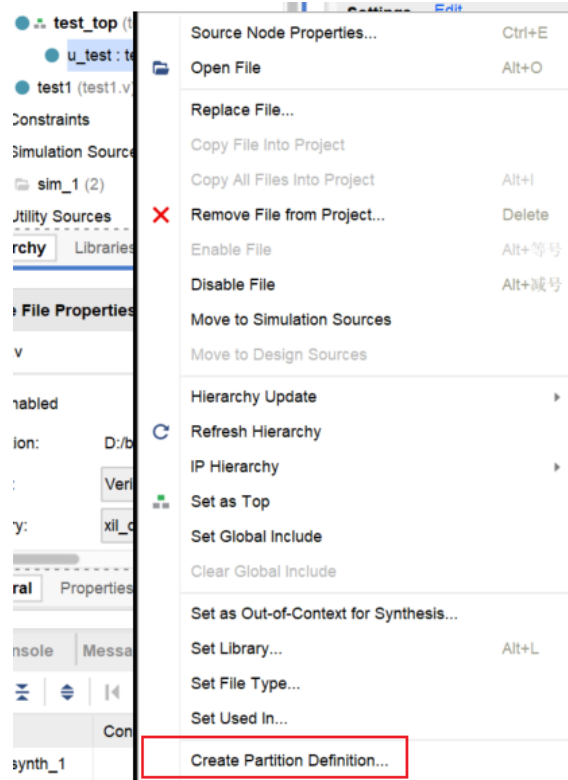


#### 动态重构硬件实现流程：

1、新建工程，点击Tools->Enable Partial Reconfiguration，使能后如下所示，增加了Partial Reconfiguration Wizard选项



2、导入代码文件，右键点击选择做重构的文件，选择Create Partition Definition。



3、点击Partial Reconfiguration Wizard选项，添加多个重构的代码文件，根据提示进行以下操作。

Add Reconfigurable Module
 ✕

Add sources to a Reconfigurable Module. Choose which Partition Definition the module should be associated with and specify a name for the module. A Top Module name is required if the module hierarchy is ambiguous within the set of source files.

Partition Definition Name: test ▾
 Top Module Name:

Reconfigurable Module Name: test1 ✕
☐ Sources are already synthesized

	Index	Name	Library	Location
	1	test1.v	xil_defaultlib	D:/bigstar/project/bigstar_v1.00/slave/salve_pr0905/project_test/project_te

☐ Scan and add RTL include files into project  
☒ Copy sources into project  
☒ Add sources from subdirectories

?

### Reconfigurable Module Associations with Partition Definitions

Reconfigurable Module	Partition Definition
test	test
test1	test

Partial Reconfiguration Wizard
 ✕

### Edit Configurations

Create and modify Configurations. Configurations specify the Reconfigurable Module that will be used for each partition in the design. Automatically create a minimum set of Configurations or manually create them individually.

**Configurations**

Configuration Name		test u_test
config_1		<greybox> ▾
config_2		test ▾
config_3		test1 ▾

Partial Reconfiguration Wizard
 ✕

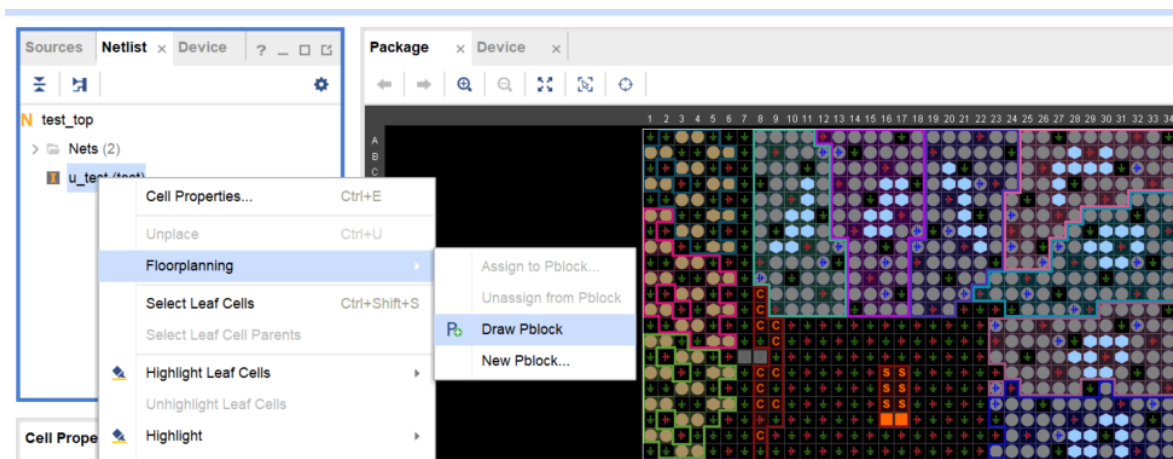
### Edit Configuration Runs

Create and modify Configuration Runs. The parent implementation run is completed first, then the static results from the parent run is inherited by the child implementation runs.

**Configuration Runs**

Name	Configuration	Parent	Constraints	Run Strategy	Report Strategy
impl_1	config_1 ▾	synth_1	constrs_1 (acti... ▾	Vivado Implementation Defaults (Vivado Implementation 20... ▾	Vivado Implementation Defau
child_0_impl_1	config_2 ▾	impl_1 ▾	constrs_1 (acti... ▾	Vivado Implementation Defaults (Vivado Implementation 20... ▾	Vivado Implementation Defau
child_1_impl_1	config_3 ▾	impl_1 ▾	constrs_1 (acti... ▾	Vivado Implementation Defaults (Vivado Implementation 20... ▾	Vivado Implementation Defau

4、进行综合，完成综合后打开综合设计，通过以下操作，进行动态区划分



画出一块作为pblock，保存，工具会将pblock约束添加到约束文件中，然后执行reports》report drc  
约束文件中生成了划分动态区的对应语句。

```
> bigstar > project > bigstar_v1.00 > slave > salve_pr0905 > project_test > project_test.srcs > constrs_1 > new >   
1 create_pblock pblock_u_test  
2 add_cells_to_pblock [get_pblocks pblock_u_test] [get_cells -quiet [list u_test]]  
3 resize_pblock [get_pblocks pblock_u_test] -add {SLICE_X110Y101:SLICE_X173Y149}  
4 resize_pblock [get_pblocks pblock_u_test] -add {DSP48_X9Y42:DSP48_X16Y59}  
5 resize_pblock [get_pblocks pblock_u_test] -add {RAMB18_X7Y42:RAMB18_X10Y59}  
6 resize_pblock [get_pblocks pblock_u_test] -add {RAMB36_X7Y21:RAMB36_X10Y29}  
7
```

5、综合布局布线生成bit文件即可。

## 非项目模式动态重构流程

以下步骤总结了非项目模式下DFX设计的处理过程：

- 1.分别对静态模块和可重构模块进行综合。
- 2.创建物理约束（Pblocks）以定义可重新配置的区域。
- 3.在每个可重新配置的分区（RP）上设置HD.RECONFIGURABLE属性。
- 4.在上下文中实现完整的设计（静态和每个RP一个RM）。
- 5.为完全路由的设计保存一个设计检查点。
- 6.从此设计中删除RM，并保存一个仅静态的设计检查点。
- 7.锁定静态放置和路由。
- 8.将新的RM添加到仅静态设计中，并实现此新配置，为完全路由设计保存一个检查点。
- 9.重复步骤8，直到执行完所有RM。
- 10.在所有配置上运行验证实用程序（pr\_verify）。
- 11.为每个配置创建比特流。

**设计流程指令：**

参考：<https://xilinx.eetrend.com/content/2022/100566069.html>

顶层模块综合命令：

```
synth_design -flatten_hierarchy rebuilt -top <top_module_name> -part <part>
```

RM模块综合命令：

```
synth_design -mode out_of_context -flatten_hierarchy rebuilt -top  
<reconfig_module_name> -part <part>
```

组合网表命令：

```
<1> Add and Link Files  
#在内存中直接创建工程  
create_project -part <part> -in_memory  
#添加文件  
add_files <top>.dcp  
add_files <rp1_rmA_top>.dcp  
add_files <rp1_rmA_lower>.dcp  
add_files <rp2_rmA_top>.dcp  
#用文件的SCOPED_TO_CELLS属性定义这些cell的层次关系  
set_property SCOPED_TO_CELLS {<RP1_module_instance>} [get_files  
<rp1_rmA_top>.dcp]  
set_property SCOPED_TO_CELLS {<RP1_lower_module_instance>} [get_files  
<rp1_rmA_lower>.dcp]  
set_property SCOPED_TO_CELLS {<RP2_module_instance>} [get_files  
<rp2_rmA_top>.dcp]  
#使用link_design命令组合这些网表文件，指定在DFX设计中所包含的RP  
link_design -top <top> -part <part> -reconfig_partitions {<RP1_module_instance>  
<RP2_module_instance>}  
  
<2>Read Netlist Design  
这种方式一般用于综合网表不是vivado生成的情况  
read_edif 可以接受多种格式的网表，包括 edf, edn和ngc(ISE XST生成的网表)  
read_edif <top>.edf/edn/ngc  
read_edif <rp1_a>.edf/edn/ngc  
read_edif <rp2_a>.edf/edn/ngc  
link_design -top <top> -part <part> -reconfig_partitions {<RP1_module_instance>  
<RP2_module_instance>}  
  
<3> Open/Read Checkpoint  
#打开顶层dcp网表，然后读入属于rp1的rmA的dcp网表  
open_checkpoint <top>.dcp  
read_checkpoint -cell <cellname> <rp1_rmA_top> [-strict]  
  
<4> Open Checkpoint/Update Design  
这种方式用于合入顶层的RM的网表格式是edf或者edn。  
open_checkpoint <top>.dcp  
lock_design -level routing  
update_design -cells <rp1> -from_file <rp1_b>.{edf/edn}  
update_design -cells <rp2> -from_file <rp2_b>.{edf/edn}
```

约束命令

```
<1> 设置HD.RECONFIGURABLE属性, 定义RP模块:
set_property HD.RECONFIGURABLE TRUE [get_cells xx]

<2> 物理约束
Floorplan约束
DFX设计中RP模块强制要求有Floorplan约束(Pblock), 动态区的逻辑只能放在RP的Pblock中.
管脚约束
Partition Pin约束
可以将Partition Pin放在特定位置或者区域内
set_property HD.PARTPIN_LOCS INT_R_X4Y153 [get_pins <hier/pin>]
set_property HD.PARTPIN_RANGE SLICE_X4Y153:SLICE_X5Y157 [get_pins <hier/pins>]

<3> 时序约束
```

## Software Flow Example

### 例程: prj\_test

前提: 有完整的工程文件和重构区文件, 工程文件中重构模块的例化后需要再创建一个黑盒模块, 无代码对应, 有包含管脚约束与pblock约束的xdc文件, 此例程中只有一个RP(重构区)和这个RP下的三个RM(重构模块)。

打开Vviado Tcl Shell (2018.3和2020.2都验证有效)

```
#修改路径, 指向当前提供设计源文件的目录
cd D:/bigstar/project/DFX_software/prj_test

#读取文件, top.xdc不包含pblock的相关约束
read_verilog test_top.v
read_verilog test1.v
read_verilog test2.v
read_xdc top.xdc

#综合顶层模块, 导出综合网表
synth_design -flatten_hierarchy rebuilt -top test_top -part xc7vx690tffg1761-2
write_checkpoint top_synth.dcp

#综合RM模块, 导出综合网表
synth_design -mode out_of_context -flatten_hierarchy rebuilt -top test1 -part
xc7vx690tffg1761-2
write_checkpoint rp1_a_synth.dcp
synth_design -mode out_of_context -flatten_hierarchy rebuilt -top test2 -part
xc7vx690tffg1761-2
write_checkpoint rp1_b_synth.dcp

#组合顶层网表、RM网表和约束文件, 定义重构区, top_impl.xdc中包含了重构区创建pblock的约束
open_checkpoint top_synth.dcp
read_xdc top_impl.xdc
set_property HD.RECONFIGURABLE true [get_cells Inst_test]
read_checkpoint -cell Inst_test rp1_a_synth.dcp

#生成Configuration1布局布线后完整设计(静态+动态逻辑)的dcp(Top routed dcp)
opt_design
place_design
```

```

route_design
write_checkpoint config1_routed.dcp

#生成动态逻辑的布局布线后dcp(RM routed dcp)
write_checkpoint -cell Inst_test rp1_a_route_design.dcp

#去掉动态逻辑(为了加入新的动态逻辑网表做准备)
update_design -cell Inst_test -black_box

#锁定所有静态部分的布局布线信息，保证静态部分在之后所有的Configuration中都不再有任何变化
lock_design -level routing

#生成只包含静态逻辑的dcp(此时动态模块为黑盒)
write_checkpoint static_routed.dcp

#生成Configuration2
open_checkpoint static_routed.dcp
read_checkpoint -cell Inst_test rp1_b_synth.dcp
opt_design
place_design
route_design
write_checkpoint config2_routed.dcp
write_checkpoint -cell Inst_test rp1_b_route_design.dcp

#验证Configuration1和Configuration2 布线后设计的命令
pr_verify -full_check config1_routed.dcp config2_routed.dcp -file
pr_verify_c1_c2.log
#验证多个布线后设计的命令：
pr_verify -full_check -initial config1.dcp -additional {config2.dcp config3.dcp}
-file three_config.log

#生成比特流文件
open_checkpoint config1_routed.dcp

#1.默认生成完整设计的bit文件与Partial bit文件
write_bitstream config1

#2.只生成Partial bit文件而不生成完整设计的bit文件
write_bitstream -cell Inst_test RM_count_down_partial.bit























例：
open_checkpoint config1_routed.dcp
write_bitstream -bin_file config1
open_checkpoint config2_routed.dcp
write_bitstream -bin_file config2

单独新增动态重构的bit：
read_verilog test3.v
synth_design -mode out_of_context -flatten_hierarchy rebuilt -top test3 -part
xc7vx690tffg1761-2
write_checkpoint rp1_c_synth.dcp
open_checkpoint static_routed.dcp
read_checkpoint -cell Inst_test rp1_c_synth.dcp
opt_design
place_design
route_design
write_checkpoint config3_routed.dcp
write_checkpoint -cell Inst_test rp1_c_route_design.dcp

```



```
open_checkpoint config3_routed.dcp
write_bitstream -bin_file config3
```

 config1.bin	2023/4/6 10:35	BIN 文件	28,062 KB
 config1.bit	2023/4/6 10:35	BIT 文件	28,062 KB
 config1_pblock_inst_test_partial.bin	2023/4/6 10:35	BIN 文件	2,764 KB
 config1_pblock_inst_test_partial.bit	2023/4/6 10:35	BIT 文件	2,764 KB
 config1_routed.dcp	2023/4/6 10:30	Vivado Checkpoint ...	403 KB
 config2.bin	2023/4/6 10:37	BIN 文件	28,062 KB
 config2.bit	2023/4/6 10:36	BIT 文件	28,062 KB
 config2_pblock_inst_test_partial.bin	2023/4/6 10:37	BIN 文件	2,764 KB
 config2_pblock_inst_test_partial.bit	2023/4/6 10:37	BIT 文件	2,764 KB
 config2_routed.dcp	2023/4/6 10:33	Vivado Checkpoint ...	403 KB
 pr_verify_c1_c2.log	2023/4/6 10:33	文本文档	1 KB
 rp1_a_route_design.dcp	2023/4/6 10:30	Vivado Checkpoint ...	385 KB
 rp1_a_synth.dcp	2023/4/6 10:28	Vivado Checkpoint ...	9 KB
 rp1_b_route_design.dcp	2023/4/6 10:33	Vivado Checkpoint ...	386 KB
 rp1_b_synth.dcp	2023/4/6 10:28	Vivado Checkpoint ...	9 KB
 static_routed.dcp	2023/4/6 10:31	Vivado Checkpoint ...	391 KB
 test_top.v	2023/3/31 16:41	V 文件	1 KB
 test1.v	2023/3/31 20:50	V 文件	3 KB
 test2.v	2023/3/31 20:47	V 文件	3 KB
 top.xdc	2023/4/7 16:35	XDC 文件	2 KB
 top_impl.xdc	2023/4/7 16:36	XDC 文件	1 KB
 top_synth.dcp	2023/4/6 10:27	Vivado Checkpoint ...	8 KB

以上命令可以集成为tcl脚本直接运行，运行tcl脚本的命令：

```
source ./**/*.tcl
```

对于由Xilinx官方IP的工程，我暂时的处理方法是把包含Xilinx IP的子模块封装成edf文件再进行添加调用，若是一个单独的模块则使用read\_ip命令进行添加调用。

添加edif命令：

```
read_edif ***.edf
read_ip [glob ./**/*.xci]
```

```
read_verilog [ glob ./sources_1/imports/new/*.v ]

read_ip      [ glob./sources_1/ip/PLL50MTo100M/*.xci]

read_ip      [ glob./sources_1/ip/Pll50MTo148M5Drg90/*.xci]

read_ip      [ glob ./sources_1/ip/vio_0/*.xci]

read_xdc     [ glob ./constrs_1/imports/new/*.xdc ]
```

生成edf命令：

```
write_verilog -mode synth_stub F:/FPGA/abc_stub.v
write_edif -security_mode all F:/FPGA/abc.edf
```

# 总结

---

项目模式下实现动态重构：

优点：界面操作简单，方便设计与修改工程

缺点：新增和修改重构模块后需要重新进行综合布线生成流程，耗时较长

非项目模式下实现动态重构：

优点：大大减少增加或修改重构模块功能所需的时间

缺点：实现搭建复杂工程较难

建议：可以项目模式先实现复杂的工程建立，对已完成工程进行部分封装后再进行非项目模式下的重构设计。