Tcl让Vivado更完美

叶咏辰
资深FPGA工程师，Polycom

Club Vivado, 2014/10

Polycom

概要

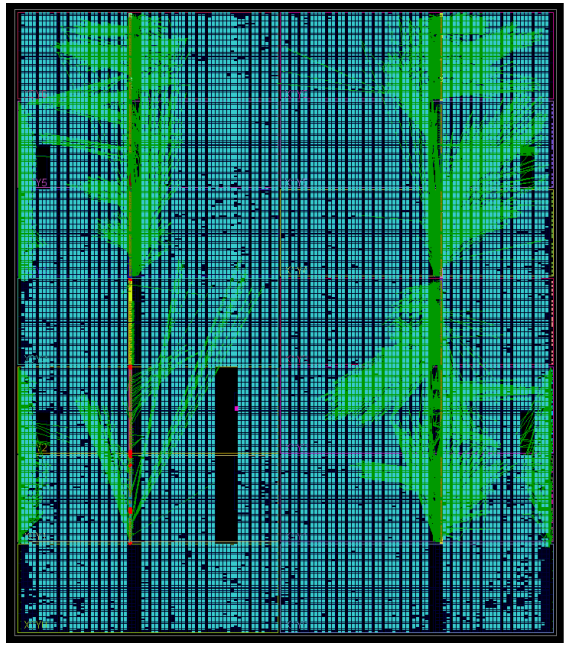
- 项目背景
- 经验分享
  - Vivado中的Tcl基本知识
  - Vivado下利用Tcl编辑综合后的网表文件
  - Vivado下利用Tcl定制丰富的报告
  - Tcl和Vivado图形界面的交互使用

概要

- **项目背景**
- 经验分享
  - Vivado中的Tcl基本知识
  - Vivado下利用Tcl编辑综合后的网表文件
  - Vivado下利用Tcl定制丰富的报告
  - Tcl和Vivado图形界面的交互使用

项目背景

- Polycom 下一代MCU产品
- FPGA主要实现视频切换和图像缩放等功能
- 芯片型号: XC7VX485T

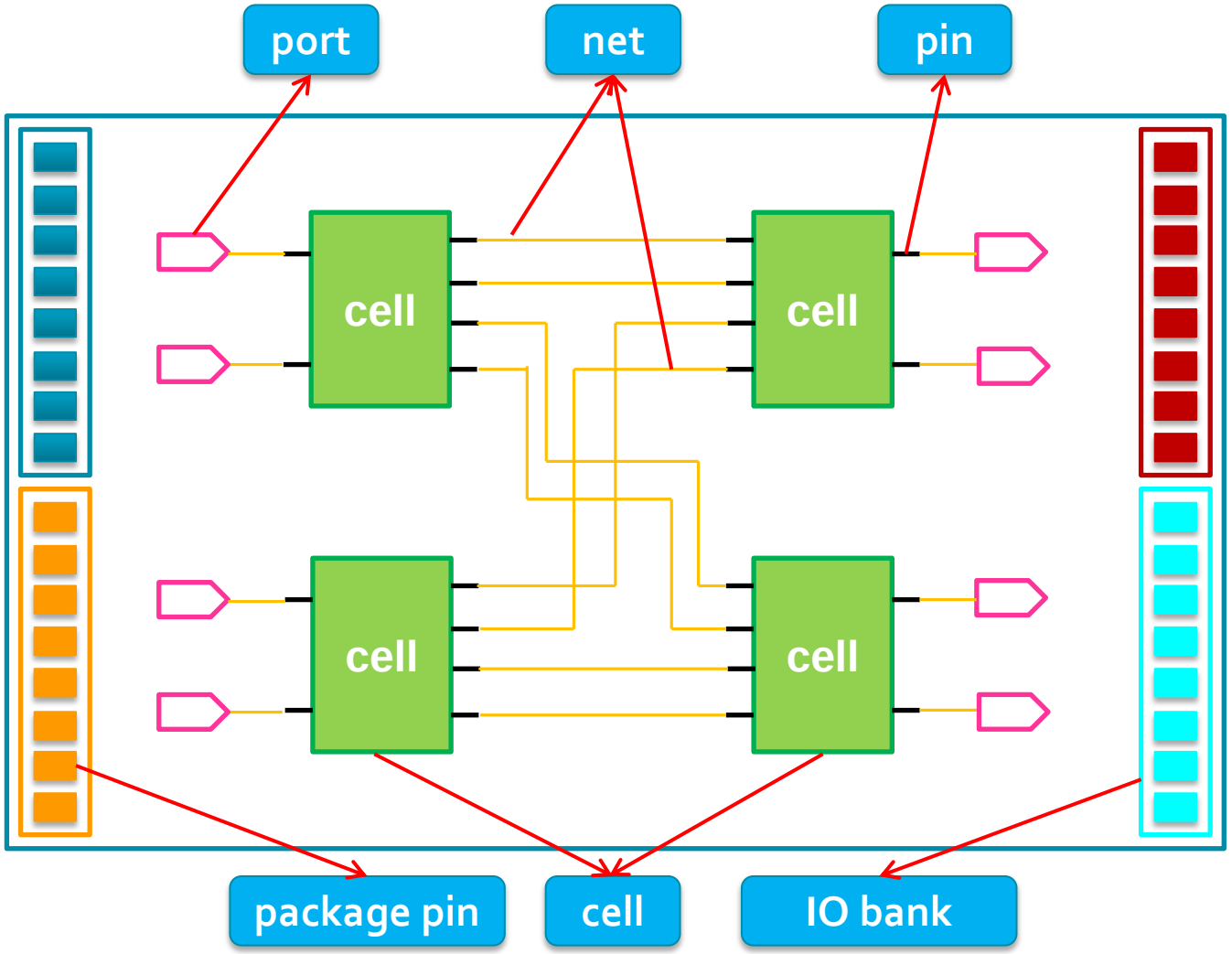


- **Vivado提供了很多策略用于实现时序收敛和资源优化**
- **通过Tcl，Vivado具备了强大的设计分析能力**
  - 快速定位设计中的问题
  - 减少设计迭代周期

概要

- 项目背景
- 经验分享
  - **Vivado中的Tcl基本知识**
  - Vivado下利用Tcl编辑综合后的网表文件
  - Vivado下利用Tcl定制丰富的报告
  - Tcl和Vivado图形界面的交互使用

Vivado网表中的基本对象



- 每个对象都有自己的属性
- 有些属性是只读的
- 有些属性是可编辑的
- 通过属性过滤可查找对象

Vivado中的五个常用Tcl命令

Command	-hierarchical	-regexp	-nocase	-filter	-of_objects
get_cells	√	√	√	√	√
get_nets	√	√	√	√	√
get_pins	√	√	√	√	√
get_ports	X	√	√	√	√
get_clocks	X	√	√	√	√

- -hierarchical ←→ -hier
- -of_objects ←→ -of
- -filter: 使用属性过滤

A.

字符串比较	
equal	==
not equal	!=
match	=~
not match	!~

B. *可以利用多个属性进行过滤* C. *返回值为二进制类型的属性可进行逻辑操作*

① `get_ports -filter {DIRECTION == IN && NAME !~ "*RESET*"}`

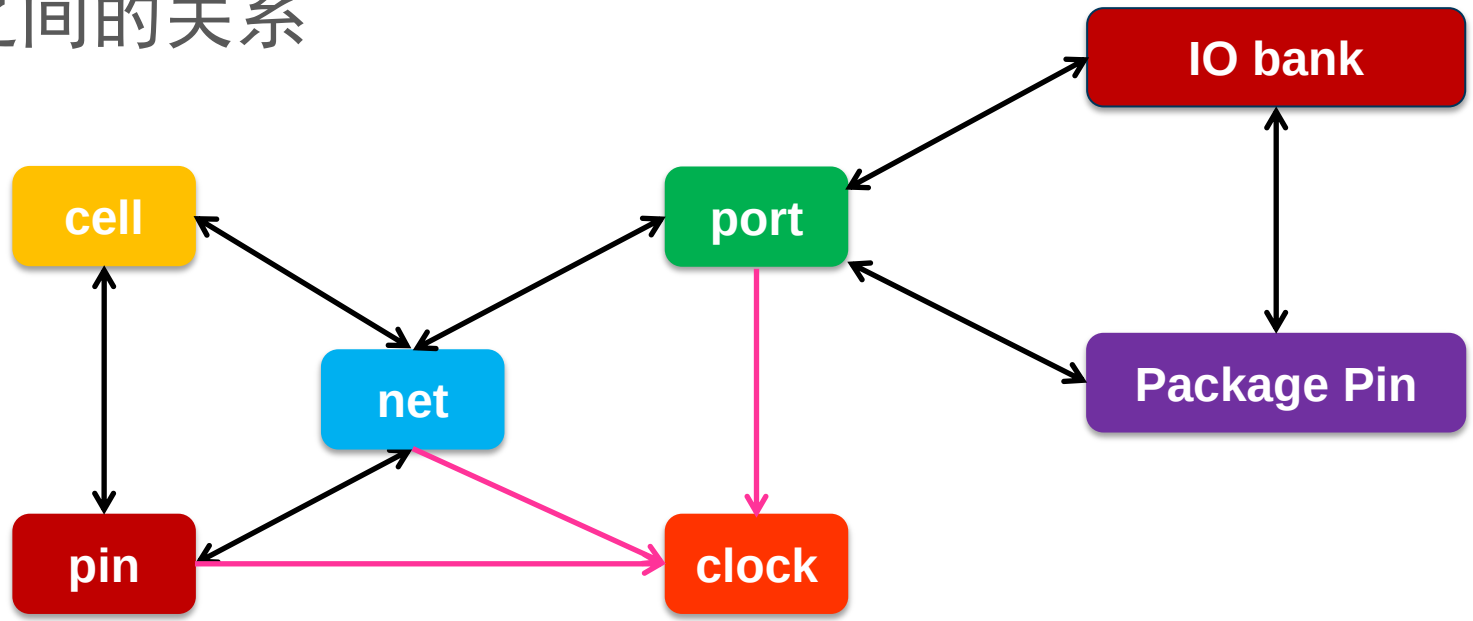
② `get_cells -filter {IS_PRIMITIVE && !IS_SEQUENTIAL}`

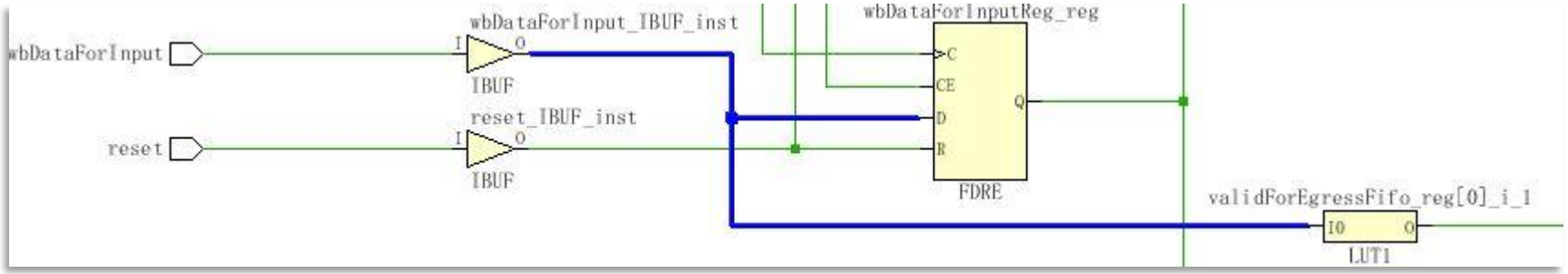
③ `get_cells -hier {*State*  *reg*}`

④ `get_cells ←→ get_cells *`

Polycom

基本对象之间的关系





示例：

输入

```
get_cells -of [get_nets -of [get_pins -of [get_cells wbDataForInput_IBUF_inst] -filter {DIRECTION==OUT}]]
```

输出

```
wbDataForInputReg_reg validForEgressFifo_reg[0]_i_1 wbDataForInput_IBUF_inst
```


Polycom

概要

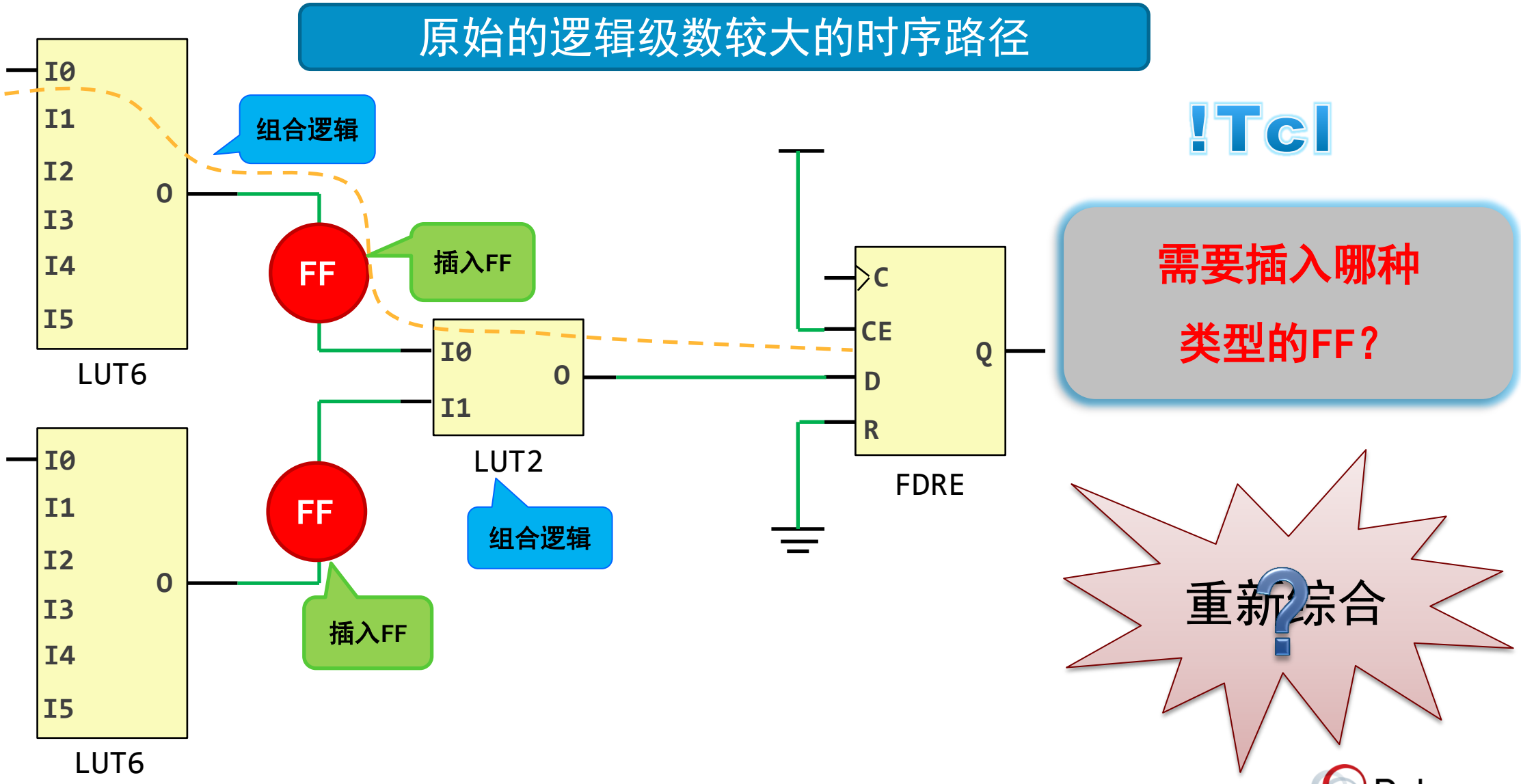
- 项目背景
- 经验分享
  - Vivado中的Tcl基本知识
  - **Vivado下利用Tcl编辑综合后的网表文件**
  - Vivado下利用Tcl定制丰富的报告
  - Tcl和Vivado图形界面的交互使用

利用Tcl编辑综合后的网表的主要应用

- 在网表中插入触发器（FF）
  - 在逻辑级数较大的时序路径上插入FF
  - 在DSP48E1之前或之后插入FF
  - 在RAMB36E1之前或之后插入FF
- 降低大扇出信号的扇出
  - 对大扇出网线做寄存器复制
  - 在大扇出网线上插入BUFG
- 修改测试信号
  - 将FPGA内部信号连接到管脚上用于测试
- 删除网表中不需要的对象
  - 删除指定模块或网线

- **节省运行时间**
  - **无需重新综合**
- **快速定位问题**
  - **避免重新综合结果的不一致使问题无法复现**

案例1：在逻辑级数较大的时序路径中插入FF



Polycom

确定FF的类型

- 新插入FF需要注意的三个问题
  - 实例化名
  - 实体名
  - 初始值
- 原始网表中的FF
  - 实例化名: local_if/data_buffer/raddr_reg
    - file dirname [get_property NAME [get_cells local_if/data_buffer/raddr_reg]]
  - 实体名: FDRE
    - get_property REF_NAME [get_cells local_if/data_buffer/raddr_reg]
  - 初始值: 1'b0
    - get_property INIT [get_cells local_if/data_buffer/raddr_reg]]

FDCE	异步复位	D, CE, C	CLR
FDPE	异步置位	D, CE, C	PRE
FDRE	同步复位	D, CE, C	R
FDSE	同步置位	D, CE, C	S

设置新插入FF的属性

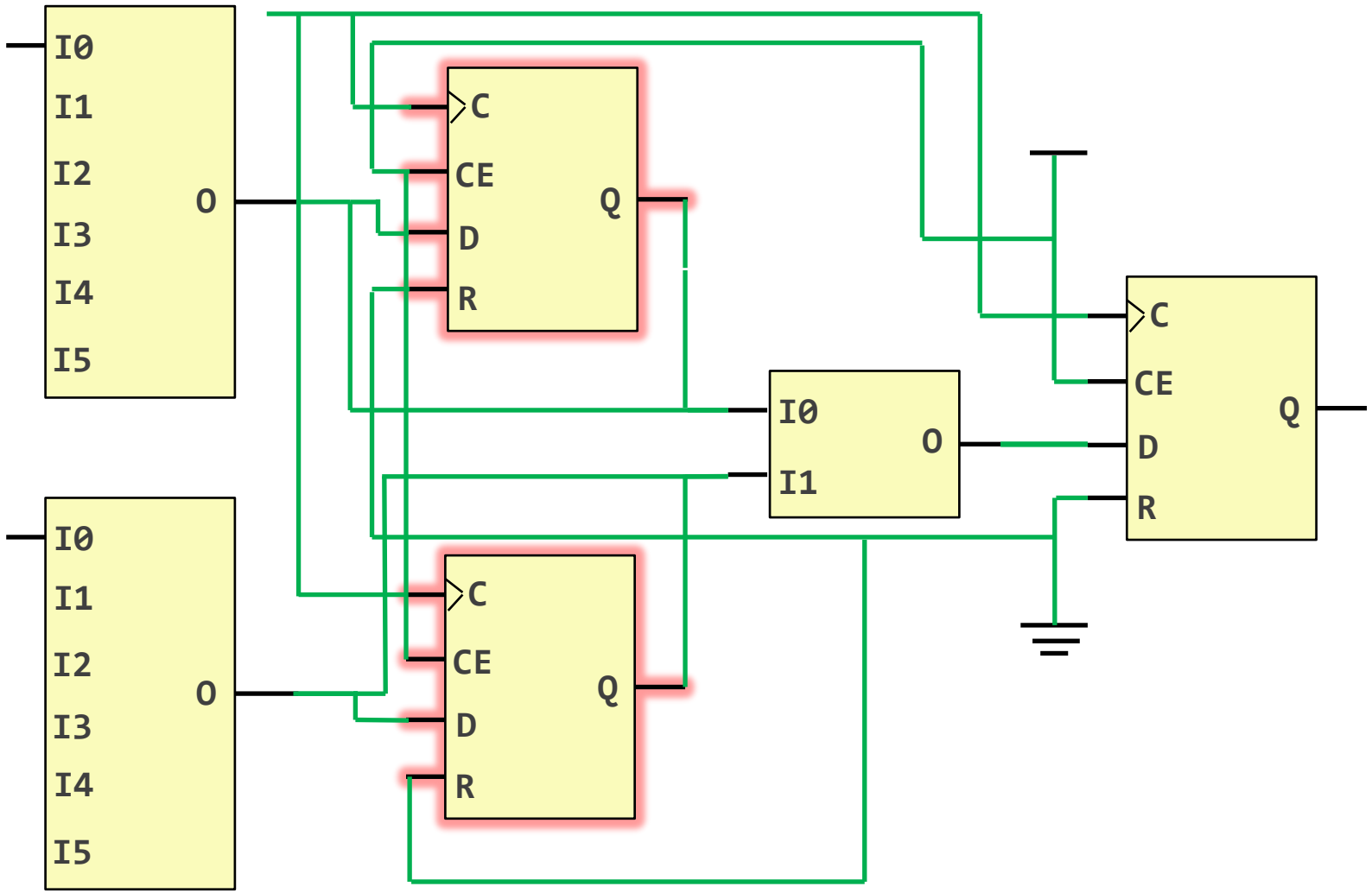
- 新插入FF的三个属性
  - 实例化名和实体名
    - `create_cell -ref FDRE $new_FF_name`
  - 初始值
    - `set_property INIT $INIT_value [get_cells $new_FF_name]`
- 三个很有用的和属性相关的脚本
  - **report_property, get_properety** and **set_property**

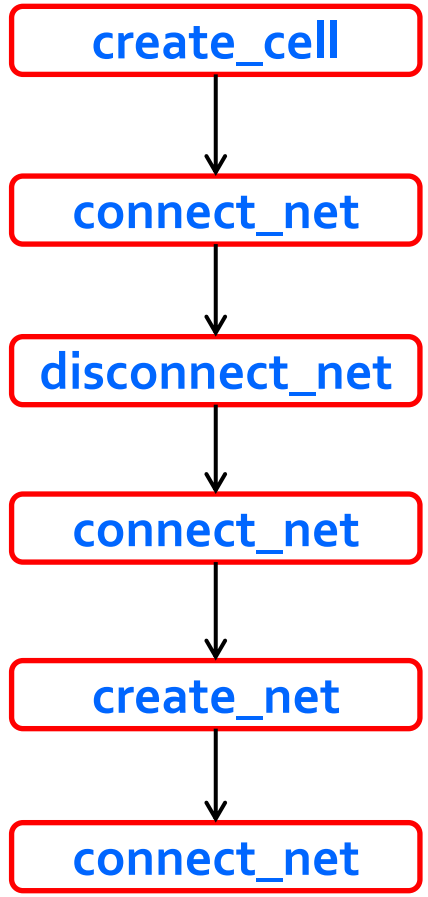
Property	Type	Read-only	Value
CLASS	string	true	cell
FILE_NAME	string	true	F:/Vivado/CPU/cpu_netlist.srcs/
INIT	binary	false	1'b0
IS_BLACKBOX	bool	true	0
IS_C_INVERTED	binary	false	1'b0
IS_D_INVERTED	binary	false	1'b0
IS_PRIMITIVE	bool	true	1
IS_R_INVERTED	binary	false	1'b0
IS_SEQUENTIAL	bool	true	1

LINE_NUMBER	int	true	948114
NAME	string	true	usbEngine1/wb_data_o_reg[9]
PARENT	cell	true	usbEngine1
PRIMITIVE_COUNT	int	true	1
PRIMITIVE_GROUP	string	true	FLOP_LATCH
PRIMITIVE_LEVEL	enum	true	LEAF
PRIMITIVE_SUBGROUP	string	true	flop
PRIMITIVE_TYPE	enum	true	FLOP_LATCH.flop.FDRE
REF_NAME	string	true	FDRE
STATUS	enum	true	UNPLACED

Polycom

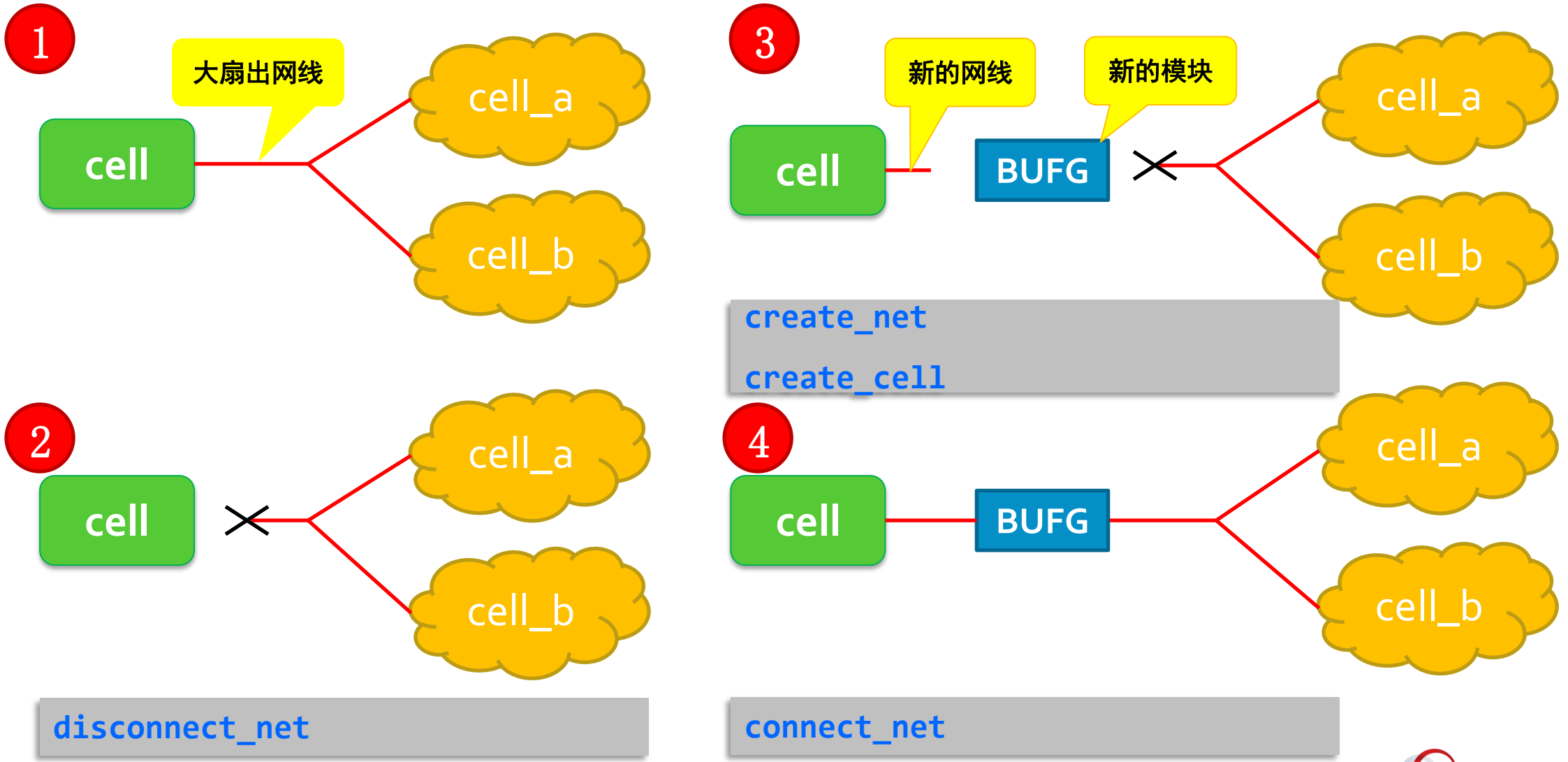
插入FF之后的网表





Polycom

案例2：在大扇出网线上插入BUFG



Polycom

插入BUFG的Tcl脚本

```
01 proc insert_BUFG {net_name {buf_name ""}} {
02   set old_net [get_nets $net_name]
03   if {[llength $old_net]!=1} {
04     puts "Error - invalid net argument - $net_name"
05     return 1
06   }
07   set opin [get_pins -leaf -of $old_net -filter {DIRECTION==OUT}]
08   if {[llength $opin]!=1} {
09     puts "Error - could not find valid driver - $net_name"
10     return 1
11   }
12   puts "Net name - $net_name - valid!"
13   # create valid bufg name
14   set net_hier [file dirname $old_net]
15   set net_parent [get_property PARENT_CELL $old_net]
16   if {$buf_name==""} {
17     if {[llength $net_parent]==0} {
18       puts "$net_name is in the top level"
19       set buf_name "my_BUFG"
20     } else {
21       puts "$net_name is not in the top level"
22       set buf_name $net_hier/my_BUFG
23     }
24   }
```

```
27 if {[llength [get_cells -quiet $buf_name]]!=0} {
28     puts "Warning - cell name $buf_name already exists."
29     set ind 0
30     while {[llength [get_cells -quiet $buf_name\_$ind]]!=0} {incr ind}
31     set buf_name $buf_name\_$ind
32   }
33   puts "Creating cell $buf_name (BUFG)"
34   create_cell -ref BUFG $buf_name
35   set new_net_name $buf_name\_inet
36   puts "Creating new $new_net_name"
37   create_net $new_net_name
38   disconnect_net -net $old_net -objects $opin
39   connect_net -net $new_net_name -objects $opin
40   connect_net -net $new_net_name -objects [get_pins $buf_name/I]
41   connect_net -net $old_net -objects [get_pins $buf_name/O]
42   puts "Insert BUFG \"$buf_name\" Successfully!"
43 }
```


Polycom

注意事项

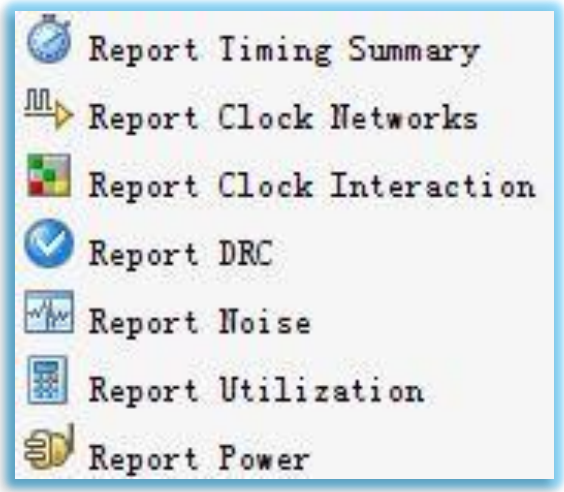
- 如何找到大扇出网线
- 如何获取经过大扇出网线的时序报告
- 如何确定BUFG资源是否可用

概要

- 项目背景
- 经验分享
  - Vivado中的Tcl基本知识
  - Vivado下利用Tcl编辑综合后的网表文件
  - **Vivado下利用Tcl定制丰富的报告**
  - Tcl和Vivado图形界面的交互使用

Vivado中的各种报告

- 采用图形界面方式可生成的各种报告
  - 时序报告： **report_timing_summary**
  - 时钟关系报告： **report_clock_interaction**
  - 资源利用率报告：**report_utilization**
  - 功耗报告： **report_power**
- 采用Tcl命令可生成的各种报告
  - 时钟属性报告： **report_clocks**
  - 时钟资源报告： **report_clock_utilization**
  - 指定路径的时序报告： **report_timing**
  - 扇出报告： **report_high_fanout_nets**
  - 控制集报告： **report_control_sets**
  - IP状况报告： **report_ip_status**
  - 功耗优化报告：**report_power_opt**



对设计分析

非常有利！

高扇出网线报告和控制集报告

```
report_high_fanout_nets -min_fanout 500 -timing -load_types
```

Net Name	Fanout	Driver Type	Worst Slack(ns)	Worst Delay(ns)	Clock Enable	Set/Reset	Data & Other	Clock
rectify_reset	10287	FDRE	8.665	0.466	0	10287	0	0
cpuEngine/or1200_cpu/or1200_ctrl/017	1017	LUT2	3.649	0.407	0	0	1017	0
usbEngine0/usb_dma_wb_in/buffer_fifo/05	912	LUT2	5.428	0.742	0	0	912	0
usbEngine1/usb_dma_wb_in/buffer_fifo/05	912	LUT2	5.428	0.742	0	0	912	0
usbEngine0/u1/u3/03	560	FDRE	8.589	0.267	0	0	560	0
usbEngine1/u1/u3/03	560	FDRE	8.589	0.267	0	0	560	0
usbEngine0/n_0_buf0_orig_reg[31]_i_2	528	LUT2	5.367	0.742	0	0	528	0
usbEngine1/n_0_buf0_orig_reg[31]_i_2	528	LUT2	5.367	0.742	0	0	528	0
n_0_reset_reg_reg_rep	525	FDRE	7.174	0.527	0	0	525	0
usbEngine0/n_0_csr0_reg[12]_i_2__11	512	LUT2	5.346	0.527	0	0	512	0

```
report_control_sets -verbose -sort_by {clk set}
```

Clock Signal	Enable Signal	Set/Reset Signal
clkgen/cpuClk	cpuEngine/or1200_ic_top/or1200_ic_fsm/WEA[0]	
clkgen/cpuClk	cpuEngine/or1200_dc_top/or1200_dc_fsm/WEA[0]	
clkgen/cpuClk	cpuEngine/or1200_cpu/or1200_ctrl/0204[0]	
clkgen/cpuClk		
clkgen/cpuClk		cpuEngine/or1200_cpu/or1200_ctrl/0112
clkgen/cpuClk		cpuEngine/or1200_dmmu_top/or1200_dmmu_tlb/dtlb_tr_ram/SR[0]
clkgen/cpuClk		cpuEngine/or1200_immu_top/iaddr_qmem_hit

Polycom

可定制的资源利用率报告

```
report_utilization -hierarchical -cells [get_cells usbEngine0/u1]
```

Instance	Module	Total LUTs	Logic LUTs	LUTRAMs	SRLs	FFs	RAMB36	RAMB18	DSP48 Blocks
u1	usbf_pl_29	1771	1763	0	8	516	0	0	0
u1	usbf_pl_29	1771	1763	0	8	516	0	0	0
(u1)	usbf_pl_29	89	89	0	0	63	0	0	0
u0	usbf_pd_32	768	760	0	8	58	0	0	0
u2	usbf_idma_33	218	218	0	0	218	0	0	0
u3	usbf_pe_34	696	696	0	0	177	0	0	0

Switch Name	Property Name	Value
-x	Ignore User Timing Constraints	☐
-ntd	Timing Mode	Performance Evaluation
-u	Trim Unconnected Signals	☑
-detail	Generate Detailed MAP Report	☐
-pr	Pack I/O Registers/Latches into IOBs	For Outputs Only
-power	Power Reduction	Off
-activityfile	Power Activity File	

```
report_clock_utilization
```

Type	Used	Available	Num Fixed
BUFG	12	32	0
BUFH	0	96	0
BUFIO	0	24	0
MMCM	1	6	0
PLL	0	6	0
BUFR	0	24	0
BUFMR	0	12	0

- 在ISE中，只有选定Map属性中的'Generate Detailed Map Report'才可以看到某个模块的资源利用率
- 在ISE中，没有单独的时钟资源报告，只能在系统资源利用率中看到
- 在Vivado中，通过Tcl可以产生各种可定制的报告

Polycom

可定制的时序报告

```
01 # Description: -through: net pin or cell
02 # Use -through to get timing path and report timing
03 proc thr_timing_rpt {ListOfEmt} {
04   puts [format {%-40s %-40s %-20s %-20s %7s} "Start Point" "End point"
"Launch Clock" "Capture Clock" "Slack"]
05   puts [string repeat "-" 140]
06   set path [list]
07   set class_type [list net cell pin]
08   foreach thr_opt $ListOfEmt {
09     set class [get_property CLASS $thr_opt]
10     if {[lsearch $class_type $class]==-1} {
11       puts "Error: -through opt must be net, cell or pin!"
12       return 1
13     }
14     set path_i [get_timing_paths -through $thr_opt -nworst 100 -
unique_pins]
15     lappend path $path_i
16   }
17   foreach mypath $path {
18     set startpoint [get_property STARTPOINT_PIN $mypath]
19     set startclock [get_property STARTPOINT_CLOCK $mypath]
20     set endpoint [get_property ENDPOINT_PIN $mypath]
21     set endclock [get_property ENDPOINT_CLOCK $mypath]
22     set slack [get_property SLACK $mypath]
23     puts [format {%-40s %-40s %-20s %-20s %7s} $startpoint $endpoint
$startclock $endclock $slack]
24   }
25 }
```

- report_timing 和 get_timing_path中的更多选项
  - -from
  - -to
  - -through
  - -delay_type
  - -max_paths
  - -nworst
  - -unique_pins
  - -sort_by
  - -slack_lesser_than

概要

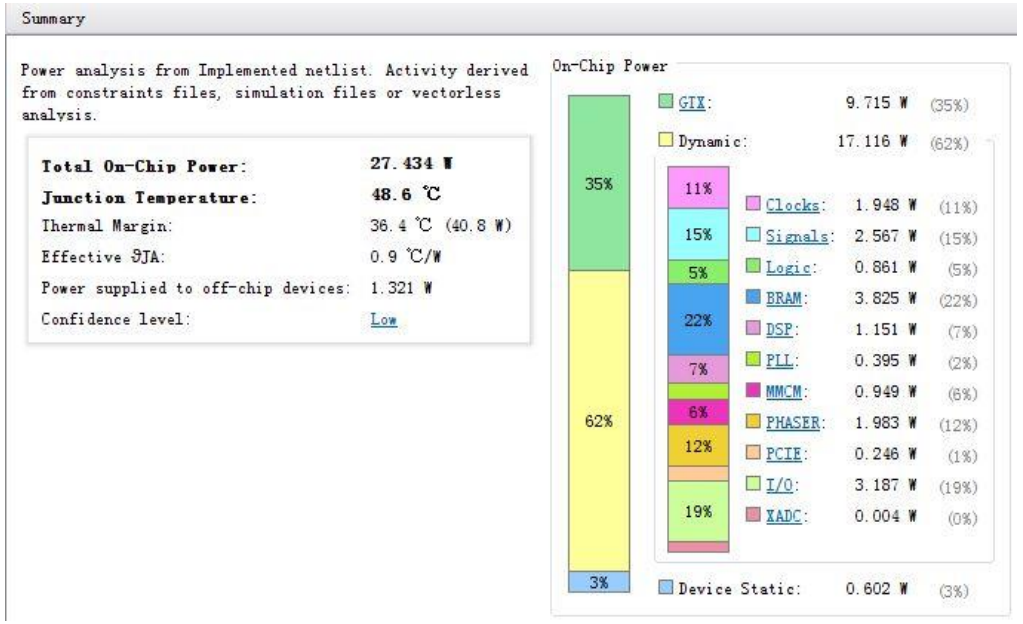
- 项目背景
- 经验分享
  - Vivado中的Tcl基本知识
  - Vivado下利用Tcl编辑综合后的网表文件
  - Vivado下利用Tcl定制丰富的报告
  - **Tcl和Vivado图形界面的交互使用**

将I/O寄存器放入IOB中

- 在ISE中，Map属性里有‘Pack I/O registers into IOBs’，该属性有4个值：
  - For Inputs Only, For Outputs Only, For Inputs and Outputs, Off
- 在Vivado中，采用Tcl实现同样功能将更为灵活
  - 将指定的输入管脚对应的寄存器放入IOB中
    - `set_property IOB true [all_fanout -flat -endpoints_only -only_cells [get_ports lb_sel_pin]]`
    - `set_property IOB true [get_ports lb_sel_pin]`
  - 将所有输入管脚对应的寄存器放入IOB中
    - `set_property IOB true [all_fanout -flat -endpoints_only -only_cells [all_inputs]]`
    - `set_property IOB true [all_inputs]`
  - 将指定的输出管脚对应的寄存器放入IOB中
    - `set_property IOB true [all_fanin -only_cells -startpoints_only -flat [get_ports led_pins[0]]]`
    - `set_property IOB true [get_ports led_pins[0]]`
  - 将所有输出管脚对应的寄存器放入IOB中
    - `set_property IOB true [all_fanin -flat -startpoints_only -only_cells [all_outputs]]`
    - `set_property IOB true [all_outputs]`

设置功耗优化

- 在Vivado里，综合后的任一阶段都可进行功耗评估
- 功耗评估的两种模式：向量模式和非向量模式
- 功耗优化的目的
  - 最大限度地降低功耗
  - 最小限度地影响时序
- 关键路径上除BRAM之外都进行功耗优化
  - `set_power_opt -exclude_cells [get_cells alu/store_ram]`
- 只对指定时钟域进行功耗优化
  - `set_power_opt -clocks [get_clocks rx_clk]`
- 只对指定类型的模块进行功耗优化
  - `set_power_opt -cell_types {bram reg}`

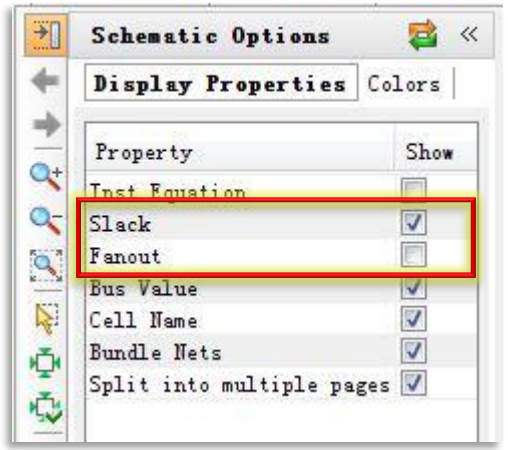


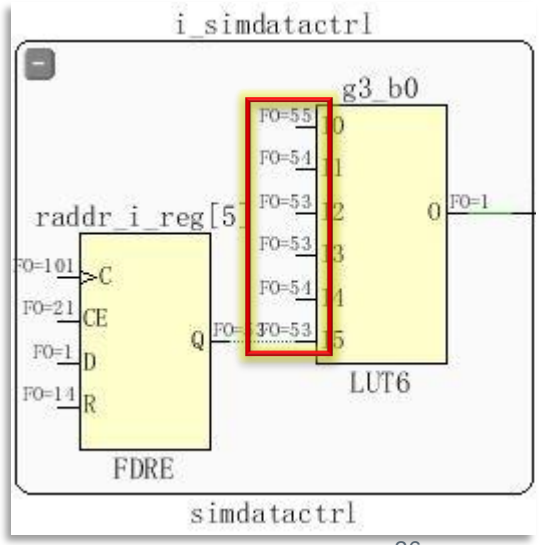
Summary

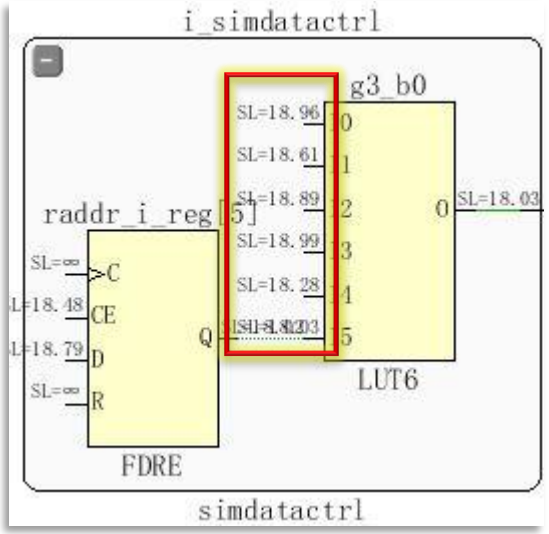
Elements	TOTAL	GATED	% GATED
Number of BRAMs	1168	1168	100.000
Number of SRLs	3	0	0.000
Number of Slice Registers	331788	143699	43.310
BRAM write mode changes	2348	16	0.681

Vivado里的原理图方式

- 获取当前选择对象的Tcl命令
  - `get_selected_objects`
- 选中指定对象并在图形界面中查看
  - `select_objects`
- 释放之前选择的对象
  - `unselect_objects`
- 快捷键
  - **F4：** 生成原理图； **F6：** 显示层次关系； **F7：** 返回代码； **F12：** 释放所有选择对象







总结

- Tcl 让Vivado更强大
  - 可以完成ISE无法胜任的工作
- Tcl让Vivado更灵活
  - 类似功能，可以比ISE做得更好
- Tcl让Vivado具有更强的交互性
  - 可以在图形界面和Tcl之间流畅切换