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

Timing Closure
Part 2

Lauren Gao

Agenda

- Vivado Baseline Timing Constraint
- Timing Closure Tips

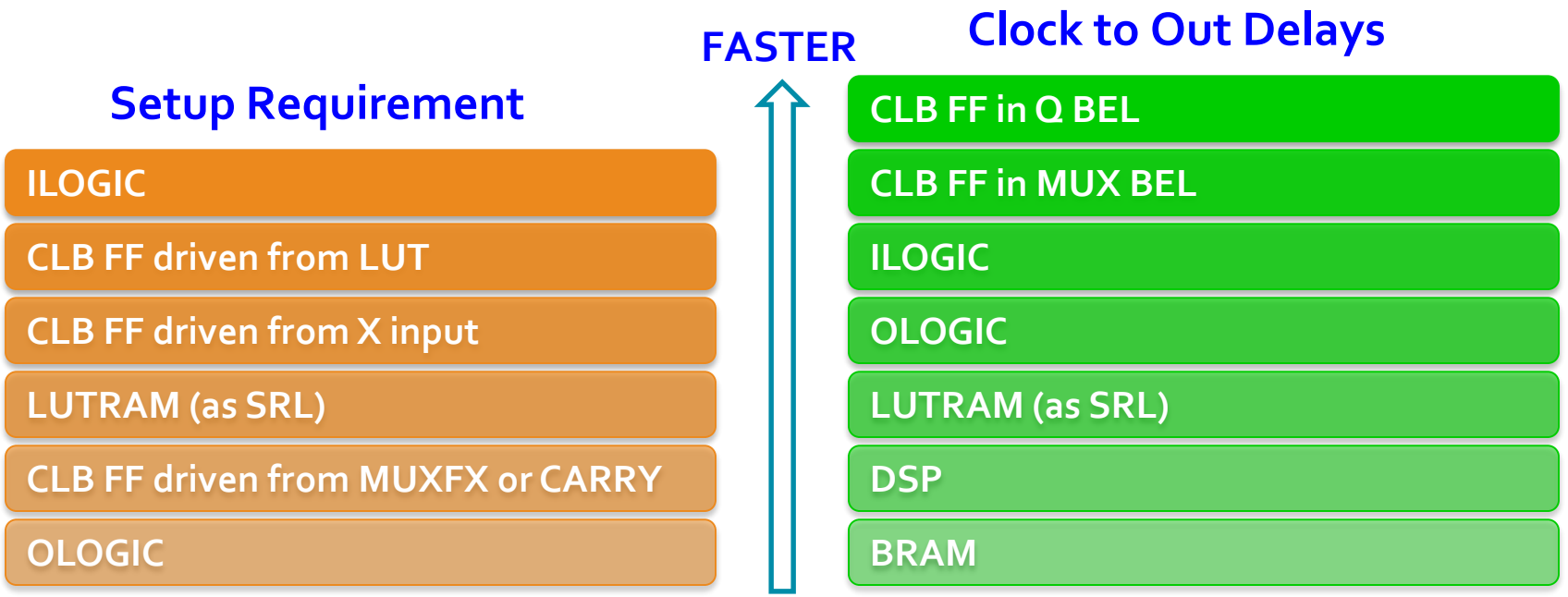
Tip 1: Good HDL Coding Style

- Coding style is not independent,oppositely, it's highly related to device architeture
- Avoid asynchronous reset
- Add pipeline register in the hierarchy boundary
- Pipeline is very helpful to improve clock frequency
  - RAM and DSP
- Read and analyze timing report **after synthesis**

Sequential Logic: Registers

Ug949 > Ch5 > Timing Closure > Reviewing Technology Choices

- **A register can be mapped to one of several types of resources in the device**
  - CLB register, CLB LUTRAM as an SRL
  - ILOGIC, OLOGIC
  - DSP and block RAMs
    - If the register is adjacent to arithmetic or memory functionality



Block RAM or Distributed RAM

Ug949 > Ch5 > Timing Closure > Reviewing Technology Choices

- **Block RAM**
  - Dedicated hardware resources, higher capacity
  - Smaller power consumption compared to distributed RAM of similar capacity
  - Higher delay getting to and from the block RAM columns
- **Distributed RAM**
  - Implemented using CLB logic
  - More suited to smaller capacity

Virtex-7 Speed Grade: -2

8kx32

	block RAM	Distributed RAM
Fmax	Over 500 MHz	250 MHz
Area	8 RAMB36, 18 CLBs	2043 CLBs
Power	370 mW	440 mW

128x4

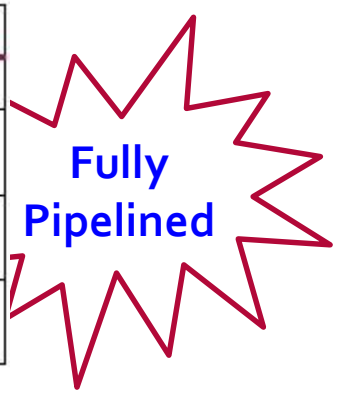
	block RAM	Distributed RAM
Fmax	About 400 MHz	Over 500 MHz
Area	1 RAMB18, 3 slices	4 slices
Power	260 mW	260 mW

DSP48E1

Ug949 > Ch5 > Timing Closure > Reviewing Technology Choices

- **DSP48E1**
  - It is more suited to wide, high-speed multiplication
  - Multiplier, MAC, Adder, Subtraction, Counters, Wide parallel logic gates
- **CLB carry logic**
  - It is usually more appropriate for KCM and small-width multipliers
- **CLB-logic based functions can be moved to DSP48E1 when CLBs are over utilized**
  - The latter is useful for addressing areas of congestion

Latency	AREG/BREG	MREG	PREG	setup path	clock to output path	Fmax
0	No	No	No	n/a	n/a	250 MHz
1	No	No	Yes	2.65 setup + 400 ps routing	350 ps clk->out + 770 ps routing	300 MHz
2	Yes	No	Yes	260 setup + 760 ps routing	350 ps clk->out + 700 ps routing	360 MHz
3	Yes	Yes	Yes	260 setup + 760 ps routing	350 ps clk->out + 700 ps routing	over 500 MHz



Tip 2: Make Your Constraints Precise and Proper

- **Over constrained**
  - Use more memory and runtime
  - To overconstrain, uncertainty gets added directly to slack equation
    - `set_clock_uncertainty -setup 0.3 [get_clocks my_CLK]`
    - After phys_opt or after route:

      `set_clock_uncertainty -setup 0  [get_clocks my_CLK]`
- **Under constrained**
  - Your design may close timing but exhibit hardware failures due to missing paths
- **All the clocks are contrained**
  - `check_timing`
  - Analyze CDC path: `report_clock_interaction`

Tip 2: Make Your Constraints Precise and Proper

- **It's better to analyze timing without IO constraints**
  - Address internal timing issues firstly
  - Vivado will not analyze IO timing if IO constraints are not avaliable
  - If internal timing is closure, IO constraints can be set
  - Use source/system synchronous timing constraints template
- **Timing exception: Less is more**
  - `set_multicycle_path -from [get_cells regB] -to [get_cells regC] N -setup`
  - `set_multicycle_path -from [get_cells regB] -to [get_cells regC] N-1 -hold`

Tip 3: Manage High Fanout Nets

- Recommended to drive high fanout nets from a synchronous start point
- Identify high fanout nets driven by LUTs
  - `report_high_fanout_nets -load_types -max_nets 100`

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

Tip 4: Bottom to Up Design Flow

- **Verify bottom module by OOC, which can accerlate design iteration**
  - Design based on IP is buttom-up
  - Each IP has its own DCP which can be resued in each iteration
- **Verison control: Git, Subversion, RCS**
- **Hierarchy design methodology**

Tip 5: Clock Domain Cross Design and Constraint

- In SDC, ALL clocks are considered related by default
- Check clock interaction report
  - `report_clock_interatcion`
- Create asynchronous clock group
  - `set_clock_groups`
- Check clock network report
  - `report_clock_networks`
- Check CDC report (2014.3 or later)
  - `report_cdc`

Tip 5: Clock Domain Cross Design and Constraint

- **Synchronous clock domain cross path**
  - Both source and destination clocks are from the same MMCM/PLL
  - The phase between source clock and destination clock is specific
  - Constraint: `set_multicycle_path`
- **Asynchronous clock domain cross path**
  - Source clock and destination clock are from different MMCM/PLL
  - The phase between them is not specific
  - Design: dual-register; FIFO; Hand-shaking
  - Add "ASYNC_REG" attribute to synchronizer
  - Constraint: `set_clock_groups; set_max_delay -datapath_only; set_false_path`

Tip 6: Physical Constraints

- **Before PCB design, run DRC**
  - IO planning
  - Clock planning
- **LOC for Macro**
  - Block RAM and DSP
- **Before floorplan, improving HDL and constraints are done firstly**
  - Floorplan: less is more
    - Only for critical part
    - Don't creat the pblocks with high resource utilization
    - Avoid overlap pblocks

Tip 7: Choose the Proper Strategy

- **The same project can include multiple design runs with different strategies**
  - Implementation strategy can cover different requirements: performance, power, area, flow
  - You can create your own strategy
- **You can add Hook Script in each design step**
  - Multiple iterations of `phys_opt_design`

Tip 8: Share Control Set Signals

- **Reduce control sets**
  - Check control sets report: `report_control_sets`
- **Combine the clocks with the same frequency**
- **Integrate clock enable signals**
- **Follow the principle of reset**
  - No reset is best
  - Synchronous reset is better if reset is needed
  - Avoid asynchronous reset
    - BRAM and DSP don't support asynchronous reset

Tip 9: Understand Log And Report Files

- **Address critical warnings and errors**
- **Check DRC violations: After synthesis and Implementation**
  - Run methodology_checks and timing_checks
- **Understand timing report and violations**
  - Both data path and clock path should be analyzed
- **Check design analysis report**
  - Provides data on critical path characteristics and complexity of the design to help identify and analyze problem areas that are subject to timing closure issues and routing congestion
  - `report_design_analysis (2014.3 or later)`

Tip 10: Bring Tcl into Full Play

- Customize your own reports
- Edit netlist
- Insert Hook Script in each design stage

Summary

- Tip 1: Good HDL Coding Style
- Tip 2: Make Your Constraints Precise and Proper
- Tip 3: Manage High Fanout Nets
- Tip 4: Bottom to Up Design Flow
- Tip 5: Clock Domain Cross Design and Constraint
- Tip 6: Physical Constraints
- Tip 7: Choose the Proper Strategy
- Tip 8: Share Control Set Signals
- Tip 9: Understand Log And Report Files
- Tip 10: Bring Tcl into Full Play