XDF XILINX DEVELOPER FORUM

Timing Closure Tips and Tricks

Presented By

Ron Plyler
Product Marketing, Vivado Implementation Tools
October 2, 2018

XILINX

Topics for Today

> Implementation Flow: Timing Closure Enhancements

> SSI Design: Tips for Maximizing Performance

> Timing Closure: Automating Solutions

Implementation Flow

Timing Closure
Enhancements

Implementation: Evolving for Timing Closure

2012-2013	2013-2014	2015-2016	2017-2018+
opt_design	**opt_design** +HFN global buffering	**opt_design** +Hierarchy-based replication driver merging	**opt_design**
power_opt_design	power_opt_design	power_opt_design	power_opt_design
place_design	**place_design**	**place_design**	**place_design** +PSIP (phys_opt_design replication, MAX_FANOUT support) +HFN global buffering
	+phys_opt_design +replication, retiming, and re-placement	phys_opt_design	phys_opt_design +SLR crossing optimization
route_design	**route_design**	**route_design**	**route_design** +PSIR (phys_opt_design replication and re-routing)
		+phys_opt_design (post-route) +critical path replication and re-routing	phys_opt_design +SLR crossing optimization

Bold indicates *required* flow step

Increasing Effectiveness

XDF XILINX DEVELOPER FORUM

XILINX.

Core Placement and Routing Improvements

> **Replication provides better default QoR**

- PSIP (Physical Synthesis In Placer) enabled by default starting 2018.2
- New MAX_FANOUT recommendations
  - Synthesis: use MAX_FANOUT only on local, low-fanout replication, not design-wide signals
  - PSIP: use MAX_FANOUT to suggest replication candidate nets during placement phase
- PSIR (Physical Synthesis In Router) introduced 2018.1 for UltraScale+

> **Router directives for higher design performance**

- New directives built on top of **Explore**
- More runtime tradeoffs

-directive AggressiveExplore (planned 2018.3)

-directive NoTimingRelaxation

-directive Explore

Post-route critical path optimization

US+ clock skew optimization

+Maintain original timing targets (don’t relax)

+More exhaustive thresholds and effort levels

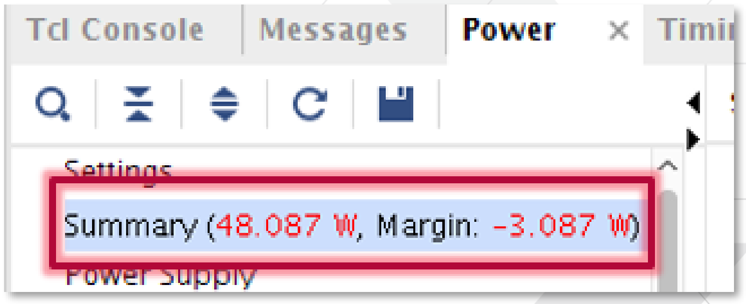
Tips on Timing Closure and Power Optimization

> **Optimize for power up front, don’t wait until after timing closure**
   >> Include power_opt_design in the flow
   >> Enable BRAM power opt in opt_design (disabled for Explore)
   >> Maximize usage and depths of cascaded BRAMs and URAMs
   >> Use set_power_opt to enable and disable power_opt at the cell level

> **Adding PowerOpt too late changes the netlist which may affect timing**

> **Set a power budget for reporting**

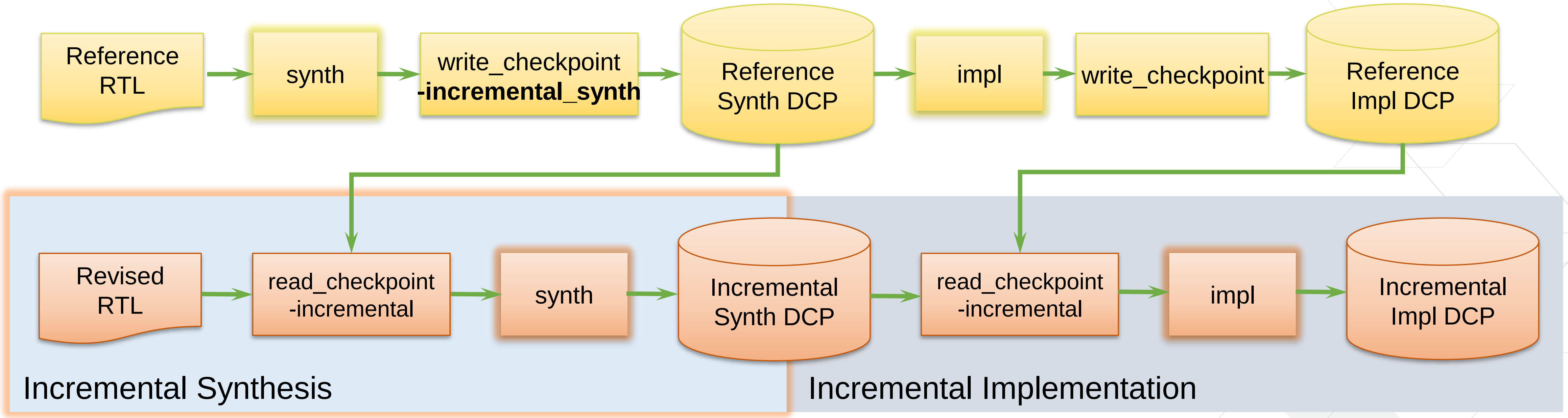
```
set_operating_conditions -design_power_budget 45
```



Summary (48.087 W, Margin: -3.087 W)

> **Consistently monitor total power for large swings**
   >> Alerts you to design changes, tool options, and strategies with negative power impact

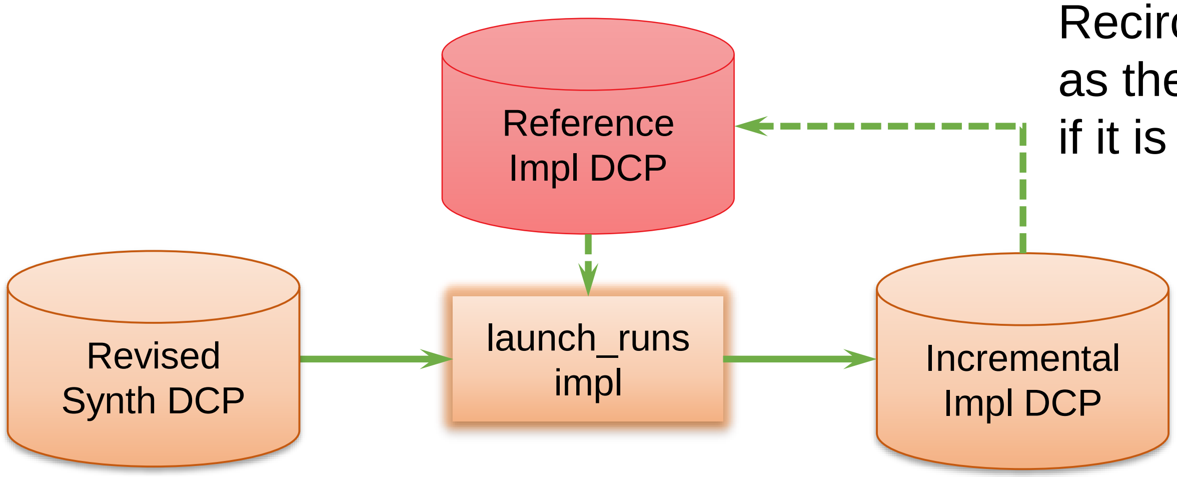
The New Incremental Compile Flow



- **Incremental Synthesis and Implementation bolt together to reduce compile time and preserve timing-closed results**
- **Incremental Synthesis minimizes netlist changes**
  - Requires write_checkpoint *-incremental_synth* option to save incr data

Incremental Implementation Enhancements

- New read_checkpoint -incremental options to reuse portions of a design
  - Streamlined, consistent, and scalable
  - Only two options: -reuse_objects <args> and -fix_objects <args>
  - Arguments are objects: <cells> <clock regions> <SLRs> [current_design]
- Automatic Incremental Implementation for Projects (EA in 2018.3)
  - Pushbutton mode where Vivado manages Incremental DCP for each run



Recirculates latest routed DCP as the reference DCP if it is a good fit

- Auto mode can coexist with manual mode
- Multiple runs supported

SSI Design

Tips for Maximizing Performance

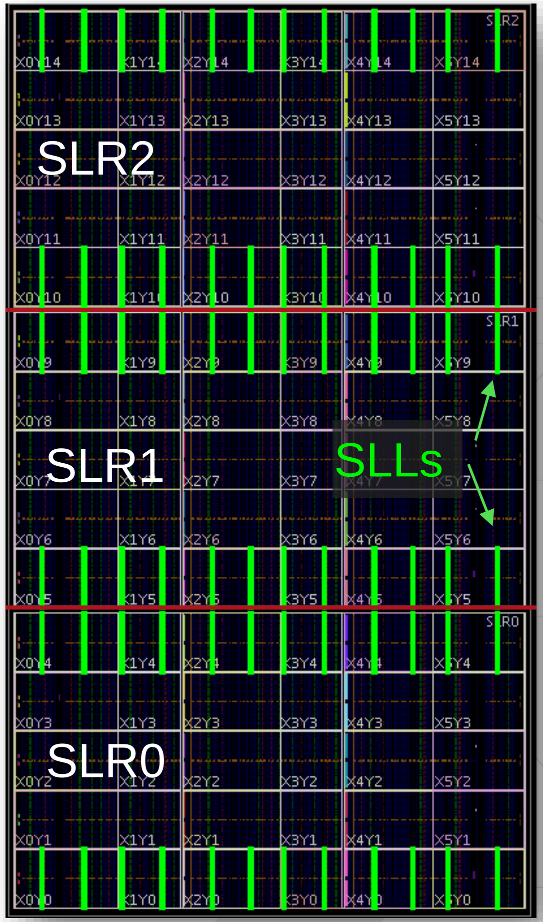
XILINX DEVELOPER FORUM

Proliferating SSI as the Platform of Choice

- > **Vivado placement and routing are continuously improving in basic key areas**
  - >> Delay Estimation - more accurate pre-route modeling of SLR crossings
  - >> Congestion - better spreading near SLR crossings
  - >> SLR Crossing Speed - more opportunistic use of SLL registers

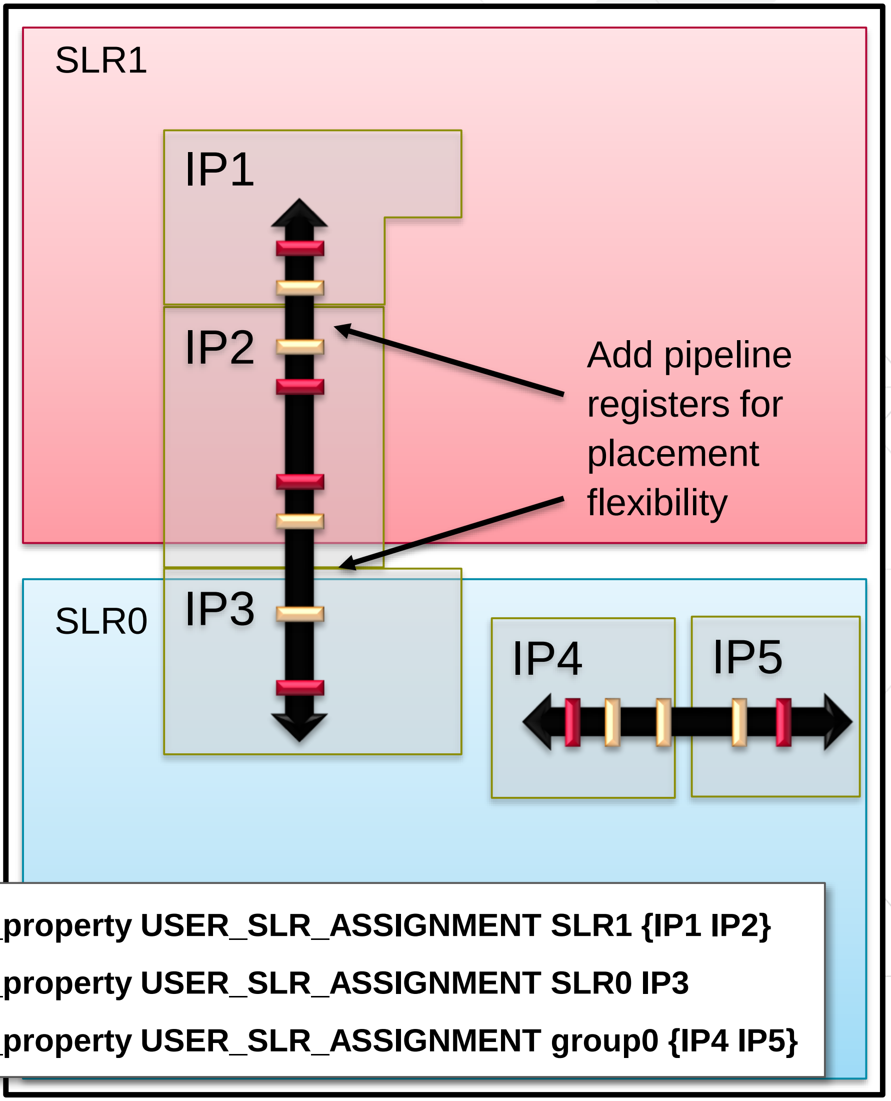
- > **New features improve quality of Partitioning and Placement**
  - >> USER_SLR_ASSIGNMENT: Control partitioning of cells
  - >> USER_CROSSING_SLR (EA): Control partitioning based on nets/pins
  - >> Laguna TX_REG -> RX_REG direct connection (UltraScale+ only)
  - >> USER_SLL_REG (EA): SLL (Laguna) register preference to improve speed, predictability

VU9P: 3 SLRs



Partitioning with USER_SLR_ASSIGNMENT

- **Hierarchical cell property *(not for leaf cells)***
  - Assigns cells to SLR when SLR name is used: SLR0, SLR1, SLR2, ...
  - Keeps cells in same SLR when value is a string
  - Tries to prevent cells from crossing SLR boundaries
- **More flexible than Pblocks**
  - Soft constraint, ignored if prevents successful partition
  - Placer, PhysOpt not restricted by Pblock bounds
- **Add pipeline registers on cell boundaries**
  - Helps maintain clock speed
  - Allows even greater placement flexibility

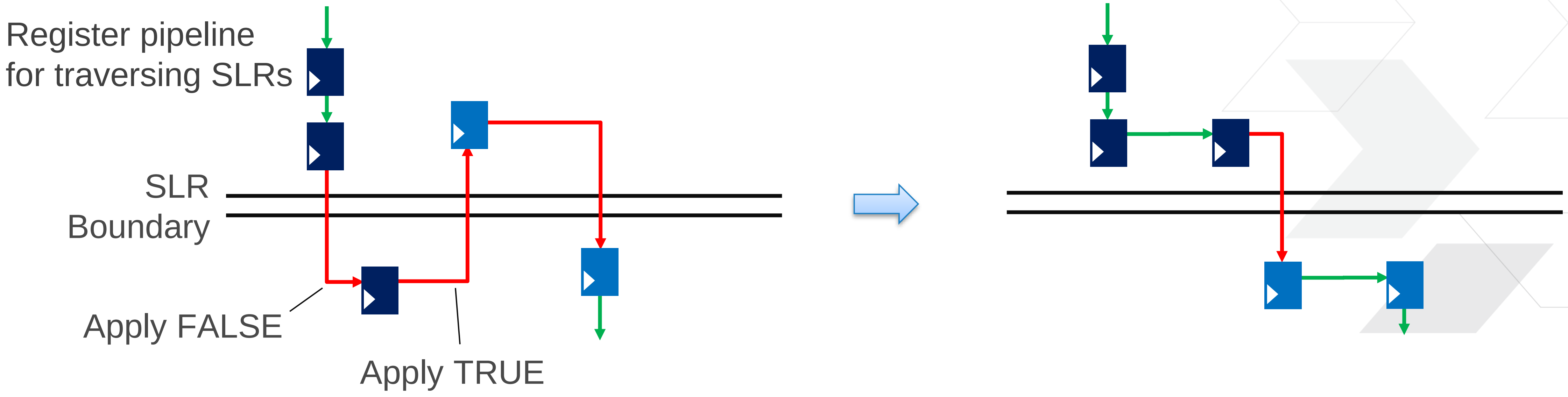


```
set_property USER_SLR_ASSIGNMENT SLR1 {IP1 IP2}
set_property USER_SLR_ASSIGNMENT SLR0 IP3
set_property USER_SLR_ASSIGNMENT group0 {IP4 IP5}
```

USER_CROSSING_SLR (Early Access Until 2018.3)

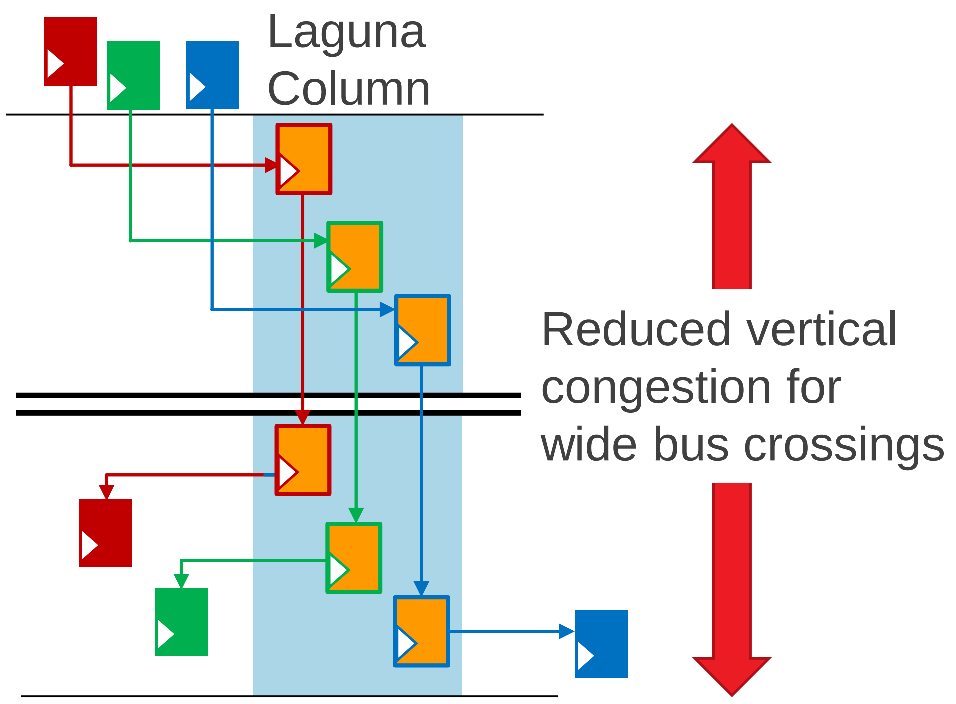
> **Soft constraint: pin/net property for fine-tuning SLR partitioning**

> **Specifies a preference that connections should cross an SLR boundary**
  >> True applies only to single-fanout pipeline register connections
  >> False applies to any net or input pin except internal library macros: PRIMITIVE_LEVEL == INTERNAL (restriction removed in 2018.3)



Using UltraScale+ SLL Registers

Register pipeline for traversing SLRs



Consistent crossing performance

- **TX_REG can drive RX_REG directly (2018.1)**
  - Router adjusts leaf clock skews to fix hold
  - Fits most intra-clock topologies
  - ***Not for use with Clock Domain Crossings***
- **Use BEL and LOC to constrain and lock SLR crossing interfaces**
- **Use USER_SLL_REG register property (EA until 2018.3)**
  - Easier method to move register from fabric to nearby Laguna register
  - Similar behavior as IOB property

SLR Crossing Optimization Throughout Implementation

- **Placer**
  - Improved automatic placement and spreading, new property-driven mapping
  - TX_REG to RX_REG direct connection (UltraScale+ only)
- **Post-Place PhysOpt**
  - -slr_crossing_opt now supported, considers small positive slack paths too
  - Add -tns_cleanup option to focus on SLR crossings more aggressively
- **Router**
  - Adjusts skew using programmable clock leafs to fix TX_REG -> RX_REG hold
- **Post-Route PhysOpt**
  - slr_crossing_opt + -tns_cleanup to focus on SLR crossings more aggressively

Note: phys_opt_design -slr_crossing_opt and -tns_cleanup are optional, not default

Feature Summary and Availability

🔴 Not yet supported
🟡 Early Access
🟢 Production

Feature	UltraScale?	UltraScale+?	2018.1	2018.2	2018.3
Manual Laguna TX-> RX	No	Yes	🟢	🟢	🟢
Auto Laguna TX-> RX	No	Yes	🔴	🟡	🟢
USER_SLR_ASSIGNMENT	Yes	Yes	🟢	🟢	🟢
USER_SLL_REG	Yes (no TX->RX)	Yes	🟡	🟡	🟢
USER_CROSSING_SLR	Yes	Yes	🟡	🟡	🟢
PhysOpt Laguna TX-> RX	No	Yes	🔴	🔴	🟢

Notes

– property name change: 2018.2: USER_SLL_REG, 2018.1: USER_LAGUNA
– USER_CROSSING_SLR only applicable to pins in 2018.1, nets planned for 2018.2

Timing Closure

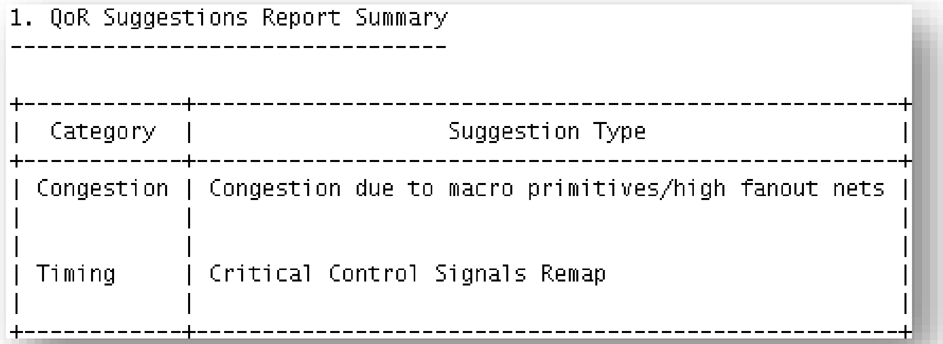
Automating Solutions

report_qor_suggestions (RQS)

> **Reduce timing closure time and effort (Introduced in 2017.1)**

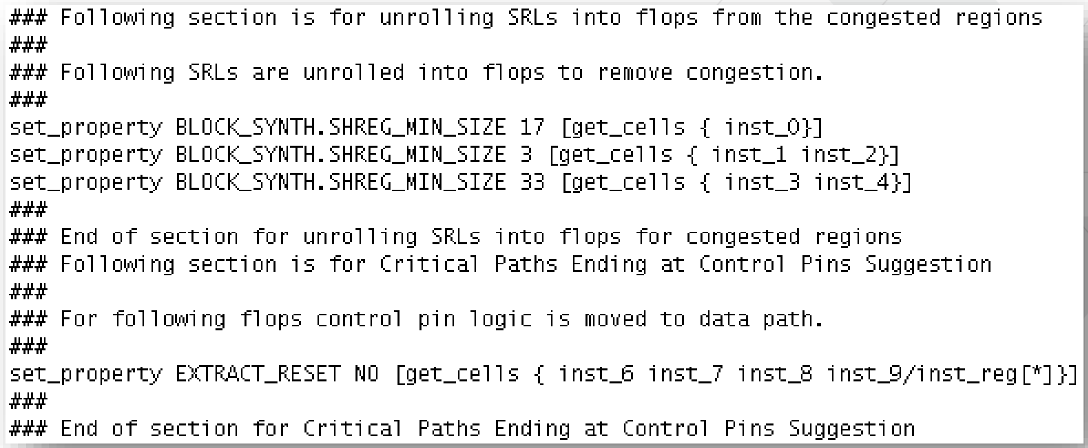
> **Run report and follow suggestions**

> **Example: RQS analysis generated suggestions to:**
>> Improve congestion
>> Improve critical paths ending at control pins



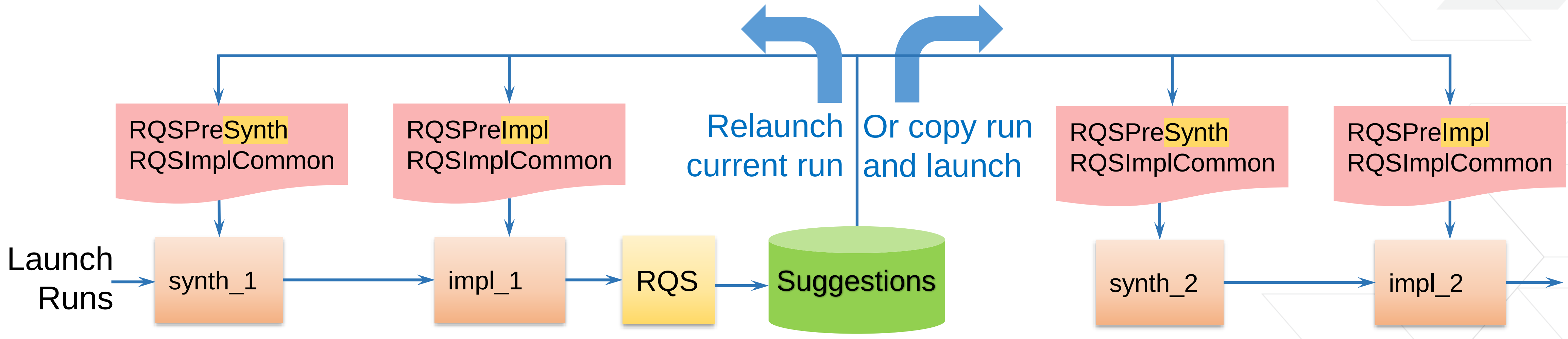
```
1. QoR Suggestions Report Summary
---------------------------------

+------------+-------------------------------------------------------+
|  Category  |                    Suggestion Type                    |
+------------+-------------------------------------------------------+
| Congestion | Congestion due to macro primitives/high fanout nets |
|            |                                                       |
|            |                                                       |
| Timing     | Critical Control Signals Remap                        |
|            |                                                       |
+------------+-------------------------------------------------------+
```

RQSPreSynth.xdc output file



```
### Following section is for unrolling SRLs into flops from the congested regions
###
### Following SRLs are unrolled into flops to remove congestion.
###
set_property BLOCK_SYNTH.SHREG_MIN_SIZE 17 [get_cells { inst_0}]
set_property BLOCK_SYNTH.SHREG_MIN_SIZE 3 [get_cells { inst_1 inst_2}]
set_property BLOCK_SYNTH.SHREG_MIN_SIZE 33 [get_cells { inst_3 inst_4}]
###
### End of section for unrolling SRLs into flops for congested regions
### Following section is for Critical Paths Ending at Control Pins Suggestion
###
### For following flops control pin logic is moved to data path.
###
set_property EXTRACT_RESET NO [get_cells { inst_6 inst_7 inst_8 inst_9/inst_reg[*]}]
###
### End of section for Critical Paths Ending at Control Pins Suggestion
```

Applying Suggestions



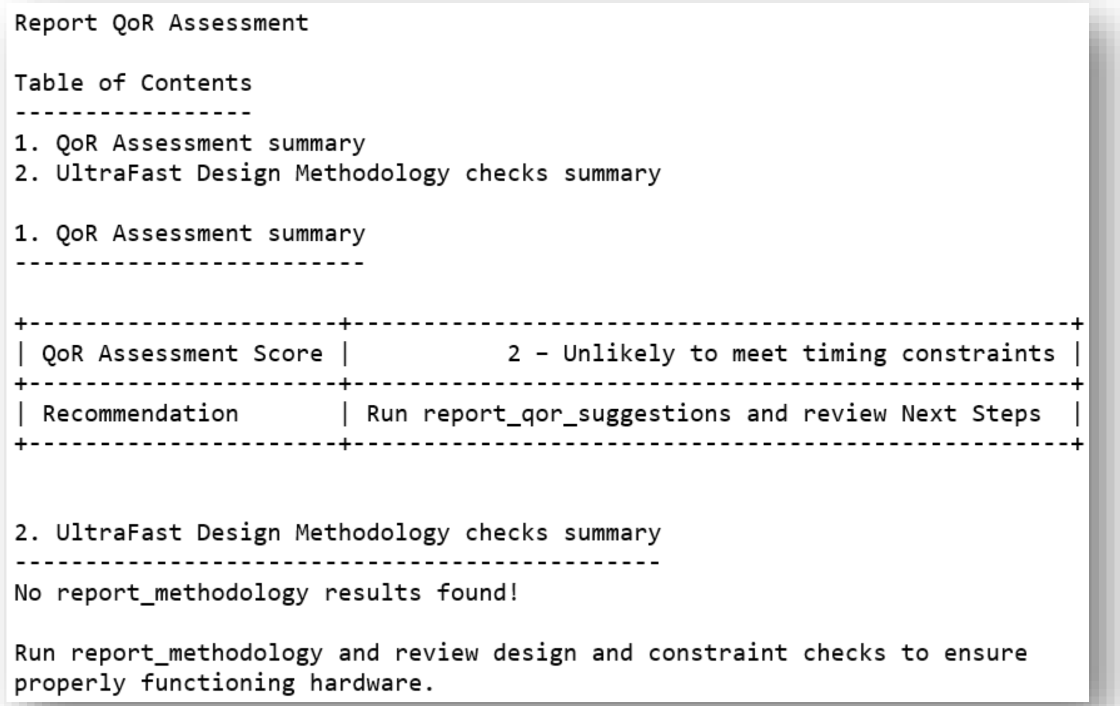
> Design sources frozen?
  → Start with Implementation
> Design sources in development?
  → Start with Synthesis
> Add RQS XDC and Tcl.pre files
  Note: XDC being phased out for Tcl

> **RQS Automation Roadmap**
  >> 2018.3: Interactively create & launch runs
  >> 2019.X: Integrate Incremental Compile
  >> 2019.X: Dynamically update suggestions throughout the flow

Introducing report_qor_assessment (RQA)

> How likely will design goals be met? Evaluates an entire design and generates a simple score

> Planned release: 2018.3



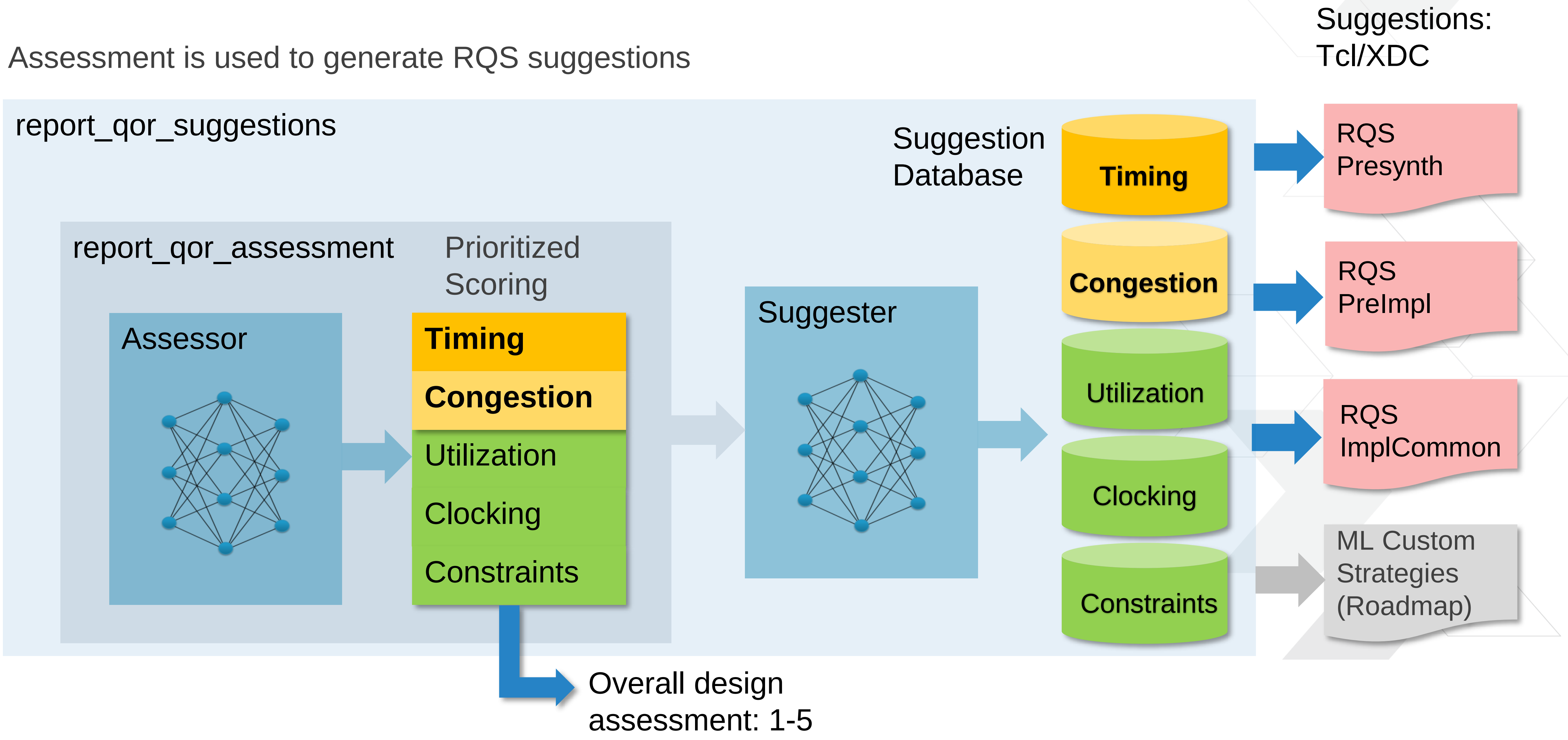
```
Report QoR Assessment

Table of Contents
-----------------
1. QoR Assessment summary
2. UltraFast Design Methodology checks summary

1. QoR Assessment summary
-------------------------

+--------------------+-----------------------------------------------------+
| QoR Assessment Score |           2 - Unlikely to meet timing constraints |
+--------------------+-----------------------------------------------------+
| Recommendation       | Run report_qor_suggestions and review Next Steps  |
+--------------------+-----------------------------------------------------+


2. UltraFast Design Methodology checks summary
----------------------------------------------
No report_methodology results found!

Run report_methodology and review design and constraint checks to ensure
properly functioning hardware.
```

Assessment Scores

1	Implementation will fail, stop flow
2	Timing will fail, review RQS
3	Timing difficult, add many strategies
4	Timing fair, add a few strategies
5	Timing easy to meet

Assessment and RQS Suggestion Integration

Assessment is used to generate RQS suggestions



Summary

> Use 2018.2 for the best UltraScale+ QoR

> Use Incremental Compile to reduce compile times and preserve timing closure

> Apply new SSI constraints to improve UltraScale and UltraScale+ performance

> Benefit from automated analysis and solutions: report_qor_assessment (2018.3) and report_qor_suggestions (Now)

>> Please share feedback on problems and improvements

XDF XILINX DEVELOPER FORUM