Timing Closure Tips and Tricks

Presented By

Ron Plyler
Product Marketing, Vivado Implementation Tools
December 10, 2018
Topics for Today

- Implementation Flow: Built for Timing Closure
- SSI Design: Tips for Maximizing Performance
- Timing Closure: Automating Solutions
Implementation Flow
Built for Timing Closure
## Start New Projects with Latest Vivado Versions

<table>
<thead>
<tr>
<th></th>
<th>2012-2013</th>
<th>2013-2014</th>
<th>2015-2016</th>
<th>2017-2018+</th>
</tr>
</thead>
<tbody>
<tr>
<td>opt_design</td>
<td>opt_design</td>
<td>opt_design</td>
<td>opt_design</td>
<td>opt_design</td>
</tr>
<tr>
<td></td>
<td>+HFN global buffering</td>
<td>+Hierarchy-based replication driver merging</td>
<td></td>
<td></td>
</tr>
<tr>
<td>power_opt_design</td>
<td>power_opt_design</td>
<td>power_opt_design</td>
<td>power_opt_design</td>
<td>power_opt_design</td>
</tr>
<tr>
<td>place_design</td>
<td>place_design</td>
<td>place_design</td>
<td>place_design</td>
<td>place_design</td>
</tr>
<tr>
<td></td>
<td>+phys_opt_design, replication, retiming, and re-placement</td>
<td>phys_opt_design</td>
<td>phys_opt_design</td>
<td>phys_opt_design</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+SLR crossing optimization</td>
</tr>
<tr>
<td>route_design</td>
<td>route_design</td>
<td>route_design</td>
<td>route_design</td>
<td>route_design</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+PSIR (phys_opt_design replication and re-routing)</td>
</tr>
<tr>
<td>Bold</td>
<td>required</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bold** indicates *required* flow step

+phys_opt_design (post-route) +critical path replication and re-routing

More and more Timing Closure features built into Implementation
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 (2018.3)

-directive NoTimingRelaxation
  - More exhaustive thresholds and effort levels
The New Incremental Compile Flow

The New Incremental Compile Flow

Reference RTL → synth → write_checkpoint -incremental_synth → Reference Synth DCP → impl → write_checkpoint → Reference Impl DCP

Revised RTL → read_checkpoint -incremental → synth → Incremental Synth DCP → read_checkpoint -incremental → impl → Incremental Impl DCP

 Incremental Synthesis and Implementation bolt together to reduce compile time and preserve timing-closed results

> 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

© Copyright 2018 Xilinx
Automatic Incremental Implementation for Projects

> Pushbutton mode where Vivado manages Incremental DCP for each run

New in 2018.3

- Recirculates latest routed DCP as the reference DCP
  - if it is a good fit
- Reference DCP added to sources in utility_1 fileset
  - Captured in project archives
SSI Design

Tips for Maximizing Performance
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
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}
```
> 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

> 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 USER_SLL_REG register property (2018.3)
  > Easier method to move register from fabric to nearby Laguna register
  > Similar behavior as IOB property

> Use BEL and LOC to constrain and lock SLR crossing interfaces
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

<table>
<thead>
<tr>
<th>Category</th>
<th>Suggestion Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Congestion</td>
<td>Congestion due to macro primitives/high fanout nets</td>
</tr>
<tr>
<td>Timing</td>
<td>Critical Control Signals Remap</td>
</tr>
</tbody>
</table>

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 3 [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_5 inst_7 inst_8 inst_9 inst_reg[*])]

### End of section for Critical Paths Ending at Control Pins Suggestion
Applying Suggestions

Launch Runs

- Design sources frozen?
  → Start with Implementation
- Design sources in development?
  → Start with Synthesis
- Add RQS XDC and Tcl.pre files

> 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) in 2018.3

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

<table>
<thead>
<tr>
<th>QoR Assessment Score</th>
<th>Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Run report_qor_suggestions and review Next Steps</td>
</tr>
<tr>
<td>1</td>
<td>Implementation will fail, stop flow</td>
</tr>
<tr>
<td>3</td>
<td>Timing difficult, add many strategies</td>
</tr>
<tr>
<td>4</td>
<td>Timing fair, add a few strategies</td>
</tr>
<tr>
<td>5</td>
<td>Timing easy to meet</td>
</tr>
</tbody>
</table>

Report QoR Assessment

Table of Contents

1. QoR Assessment summary
2. UltraFast Design Methodology checks summary

1. QoR Assessment summary

No report_methodology results found!

Run report_methodology and review design and constraint checks to ensure properly functioning hardware.
Assessment and RQS Suggestion Integration

Assessment is used to generate RQS suggestions

Overall design assessment: 1-5
Summary

> Begin new projects with the latest Vivado version
  >> 2018.3 planned for mid-December

> 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