Advanced Techniques for Enabling Gate-level CDC Verification Closure

Gunjan Jain, Mentor, a Siemens Business, India
Kurt Takara, Mentor, a Siemens Business, USA
Yuxin You, Mentor, a Siemens Business, China
Agenda

• Why CDC Verification on Gate-Level Designs
• Traditional Methodology and Challenges
• Proposed Gate-CDC Verification Methodology
• Experiments & Results
Clock Domain Crossing (CDC)

- What is CDC
  - Signal originating in one clock domain sampled in another asynchronous clock domain
Need for Gate-Level CDC Verification

• Traditionally CDC verification done on RTL

Possible CDC issues introduced during synthesis
- Glitch-prone combinational logic in CDC path
- New CDC paths due to insertion of Power/DFT logic
- CDC synchronizer template may break due to logic optimizations
Gate-CDC Example

- CDC path introduced due to insertion of logic

```vhdl
upf_version 2.0
set_design_top top
create_power_domain TOP
create_power_domain PD_TX -elements {Tx}
create_power_domain PD_RX -elements {Rx}
create_power_domain PD_RX2 -elements {Rx2}
create_supply_port VDD_HIGH
create_supply_net VDD_HIGH -domain TOP
connect_supply_net VDD_HIGH -ports VDD_HIGH
set_isolation PD_TX_ISO_OUT -domain PD_TX -clamp_value 1 -applies_to outputs -isolation_power_net VDD_HIGH
  -isolation_ground_net VSS -isolation_signal {x} -isolation_sense low -location parent
set_isolation PD_TX_ISO_OUT -domain PD_RX -clamp_value 1 -applies_to outputs -isolation_power_net VDD_HIGH
  -isolation_ground_net VSS -isolation_signal {x} -isolation_sense low -location parent
```
Gate-CDC Glitch Structure Example

- Combination logic that can reduce to:
  - $X \cdot \neg X$ or $X + \neg X$
Gate-CDC Mux Glitch Example

RTL Logic: Mux based synchronizer

Mux implementation after synthesis

Glitch-prone: sel $|$ $\sim$sel when $A = 1$ and $B = 1$

Glitch blocked: Output = 1 when $A = 1$ and $B = 1$
For given constants, logic reduces to \((tx0|\neg tx0)\) which causes glitch.
Gate-CDC Mux Synchronizer Glitch Example

RTL Logic: Mux based synchronizer

Combo-logic implementation after synthesis
CDC Verification Flow

- CDC verification is necessary on gate-level netlist
Traditional Methodology

- Setup
  - Design
  - Setup Constraints: clocks, resets, constants, ...
  - Run CDC analysis
  - Refine clocks
- CDC Analysis
  - Analyze CDC results
  - yes
    - Add constraints: stable, constant
  - no
    - Fix violations
- Debug
  - Add waivers
  - CDC sign-off
Traditional Methodology

- **Setup**
  - Design
  - Run CDC analysis
  - Analyze CDC results
  - Setup Constraints: clocks, resets, constants, ...
  - Refine clocks
  - Add constraints: stable, constant
  - Add waivers
  - Fix violations
  - CDC sign-off

- **CDC Analysis**
  - Noise
    - • bit-blasted CDC paths
    - • Scan inserted CDC path
    - • New hierarchies introduced
    - • Too many iterations needed
    - • Crossings multiplied by number of bits of vector signals in RTL
    - • False crossings due to inserted scan logic

- **Debug**
  - Huge setup effort
    - • Manually creating constraints setup not feasible
  - Noise
    - • RTL waivers do not work
    - • No correlation with RTL
  - Difficult to debug
    - • RTL waivers do not work
    - • No correlation with RTL
  - What about glitch?
Proposed Methodology

**Setup**
- Gate Design
- Constraints at RTL

**CDC Analysis**
- Run CDC analysis
- Waivers at RTL
- CDC results
- Glitch results

**Debug**
- Fix CDC & glitch issues

Transformation engine

- Automated Setup
- Reduced Noise
- Debug Eased
- Glitches Detected
- Functional combinational logic for glitch
- Focused glitch debug
• Constraints not applied due to name and topology changes post synthesis

• No black-boxing
  • Redundant processing inside the module

• Custom synchronizer not detected
  • False missing synchronizer - Noise
RTL Constraints Reuse

Auto-generated gate constraints

```
netlist blackbox cm_cdc_master
netlist blackbox cm_cdc_master_0_2
netlist blackbox cm_cdc_master_0_1

cdc custom sync
qctlib_edge_detect_async_rs_ctrl
```
RTL Constraints

Auto-generated gate constraints

Reduces noise and verification effort

cdc report crossing -from
{"io_cal_top.pcnt_reg[0]} -to {*loc.pcnt_set[0]} -severity waived

cdc report crossing -from
{"io_cal_top.pcnt_reg_reg_0_.iq} -to {*loc.pcnt_set_0_.iq} -severity waived
Auto Infer Test Mode Settings

- Scan mux inserted CDC path

- Traditional methodology
  - Synchronizer not detected

False Alarm
Auto Infer Test Mode Settings

• Scan mux inserted CDC path

• Proposed methodology
  • Structural analysis detects scan enable settings
  • Two dff synchronizer detected
Regroup Bit-blasted Crossings

- Traditional methodology
  - 16 separate 1-bit paths reported
  - `r_reg_0.iq -> q_reg_0.iq, r_reg_1.iq -> q_reg_1.iq, ...`

Crossing count multiplied by number of vector signal bits
Proposed methodology
- 16 separate paths regrouped to 1 CDC path
- `r_reg[15:0].iq -> q_reg[15:0].iq`

Crossing count same as RTL for vector signals
Proposed Methodology

Setup

CDC Analysis

Debug

Gate Design
Constraints at RTL

Run CDC analysis
Waivers at RTL

Transformation engine

CDC results
Glitch results

Transformation engine

Fix CDC & glitch issues

CDC sign-off

Automated Setup

Reduced Noise

Debug Eased

Glitches Detected
Experiment Results

- Average 1.6x improvement in runtime
- Average 18% improvement in memory consumption
- No iterations required for setup
Experiment Results

- Upto 80% reduction in noise
  - Bit-merging
  - Constraints & waivers reuse
  - Test logic removed
Summary

• Gate-CDC verification closure is necessary and now possible

• Proposed methodology addresses gate-level CDC verification challenges
  • Automatic RTL constraints transformation engine
  • Auto detection of test logic
  • Detect glitches introduced in synthesis
  • Regroup bit-blasted CDC paths
  • Correlates gate-CDC results with RTL
  • Waiver reuse

• Benefits
  • Seamless setup and reduced noise
  • Accelerates verification closure
Thank You