2021 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION CHINA
SHANGHAI | MAY 26, 2021

accellera SYSTEMS INITIATIVE

SIEMENS

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

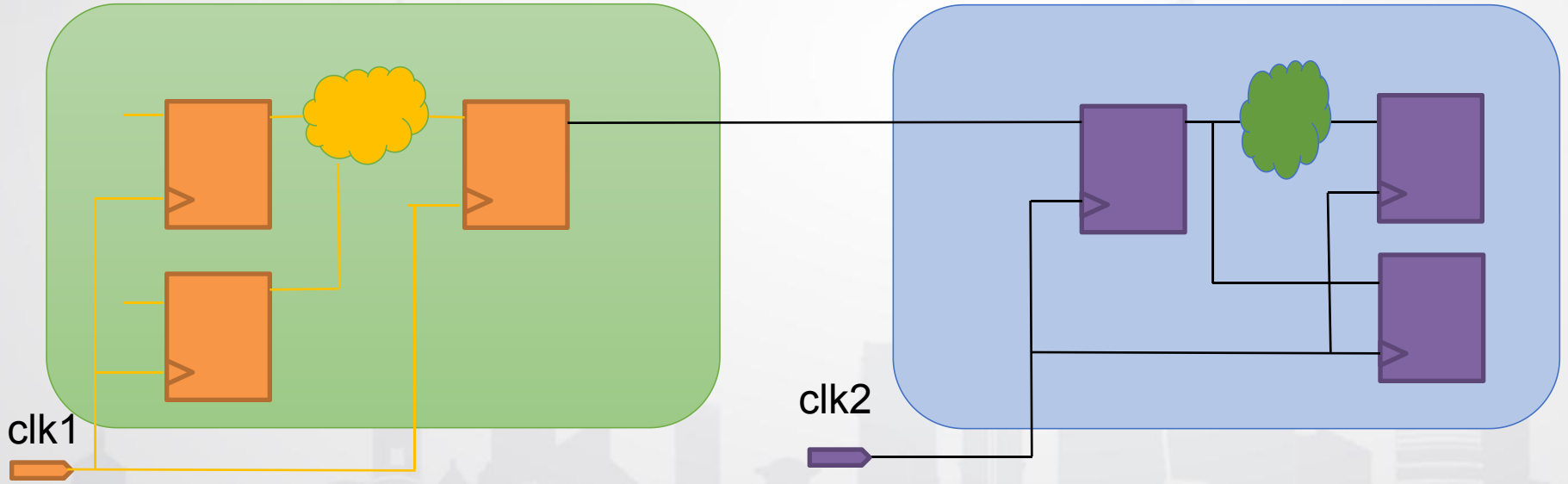
2021 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION CHINA
SHANGHAI | MAY 26, 2021

accellera SYSTEMS INITIATIVE

- 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



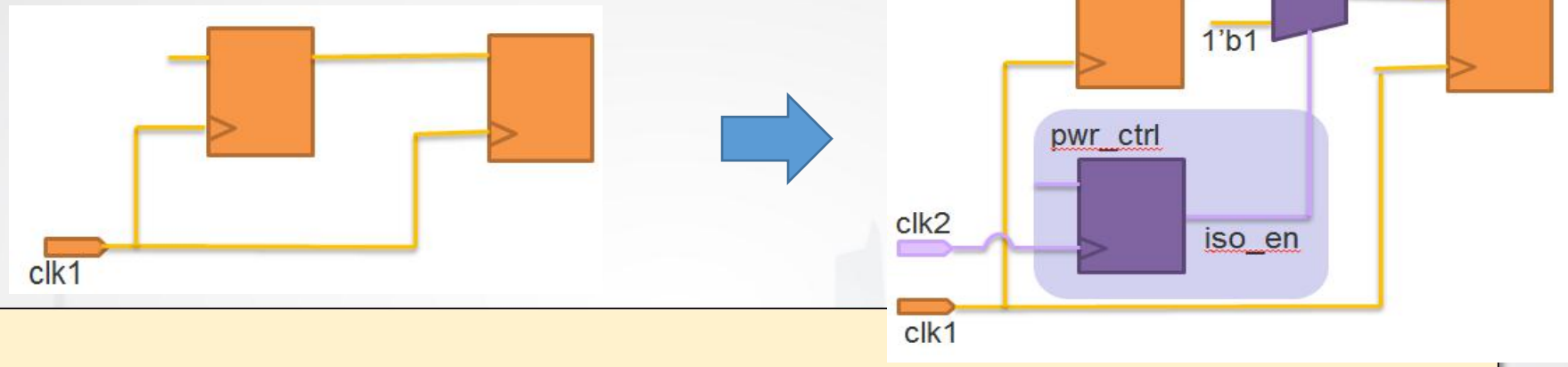
Need for Gate-Level CDC Verification

- 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

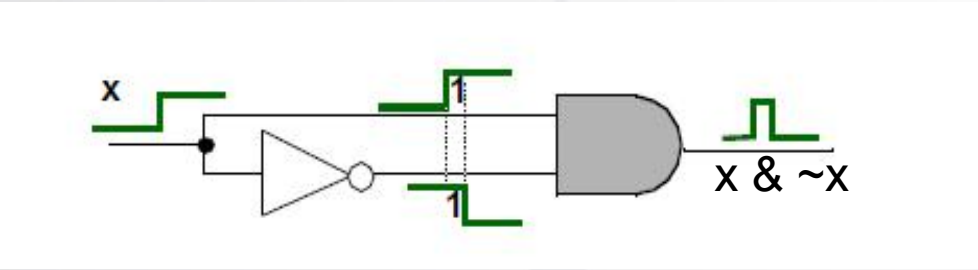
- CDC path introduced due to insertion of log

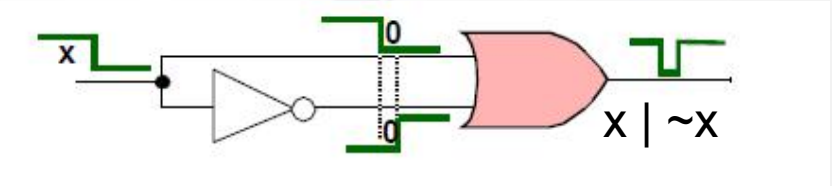



```
upf_version 2.0
set_design_top top
create_power_domain TOP
create_power_domain PD_TX -elements {Tx}
create_power_domain PD_RX -elements {Rx}
create_power_domain PD_RX2 -elements {Rx2}
create_supply_port VDD_HIGH
create_supply_net VDD_HIGH -domain TOP
connect_supply_net VDD_HIGH -ports VDD_HIGH
set_isolation 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 \
```

Gate-CDC Glitch Structure Example

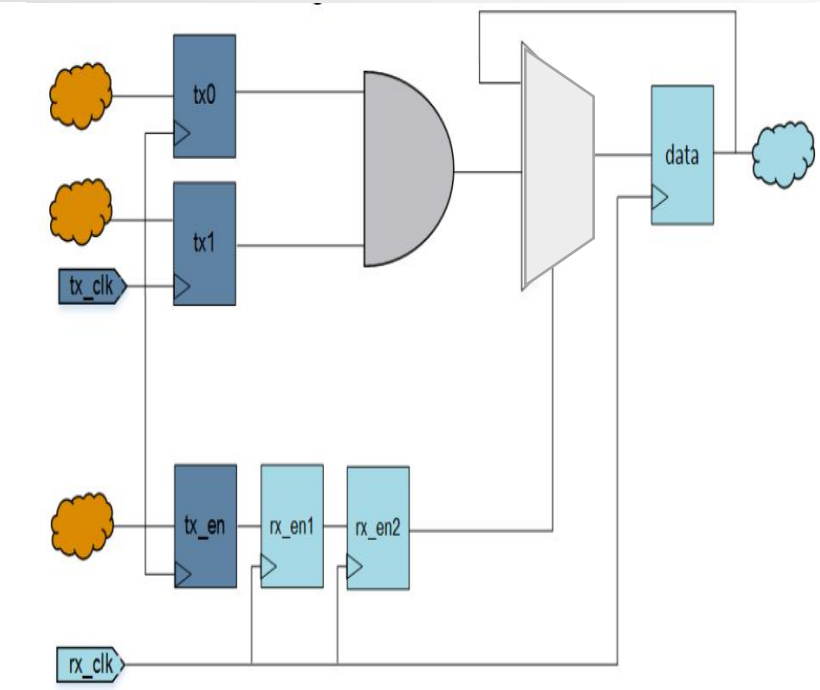
- Combination logic that can reduce to:
  - X.~X or X+~X



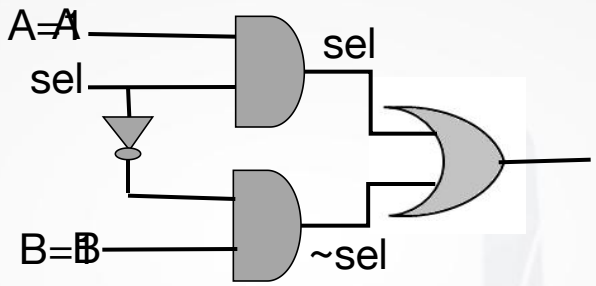


Gate-CDC Mux Glitch Example

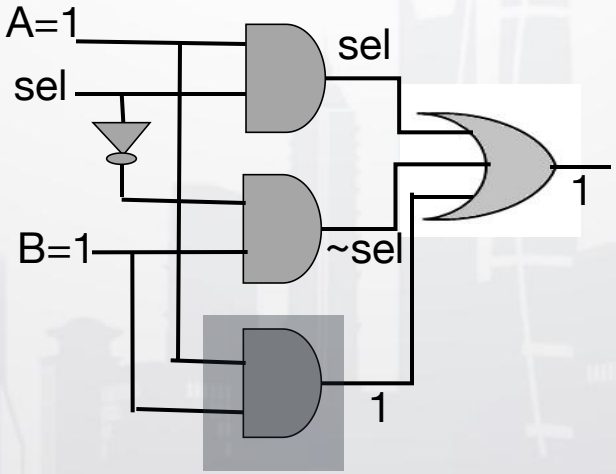
RTL Logic : Mux based synchronizer



Mux implementation after synthesis



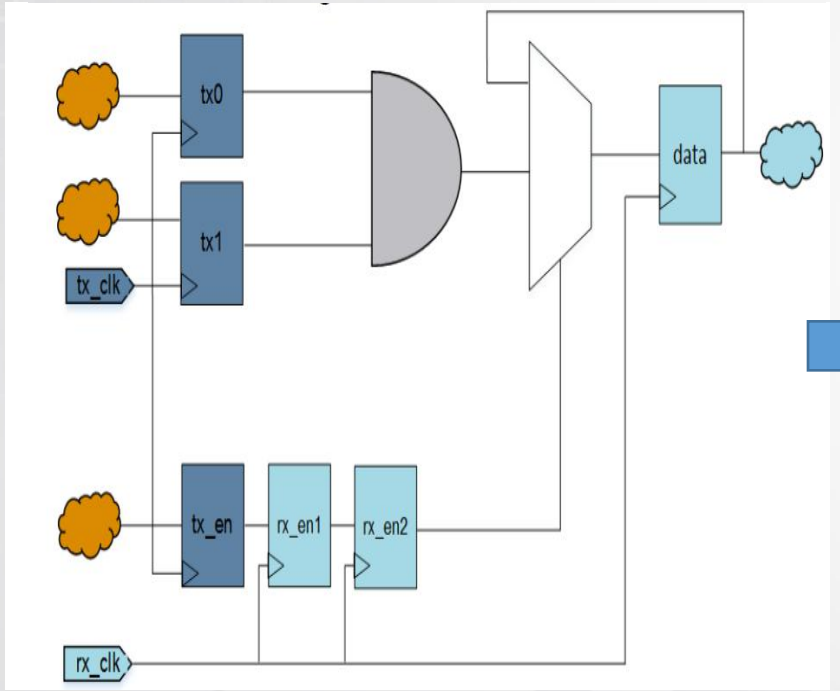
Glitch-prone :
sel | ~sel when
A = 1 and B =1



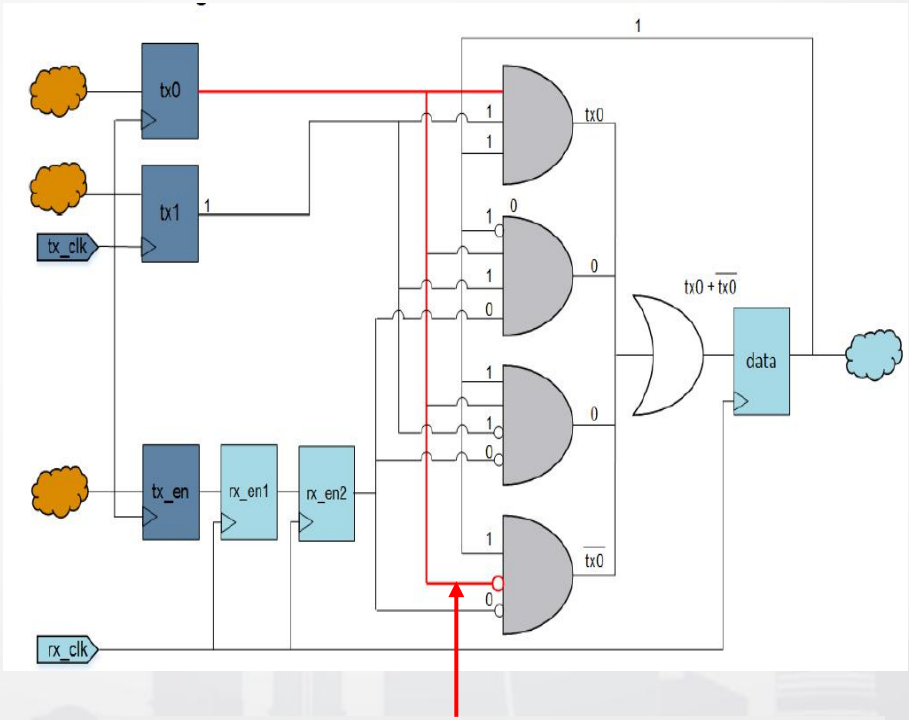
Glitch blocked :
Output = 1 when
A = 1 and B =1

Gate-CDC Mux Synchronizer Glitch Example

RTL Logic : Mux based synchronizer



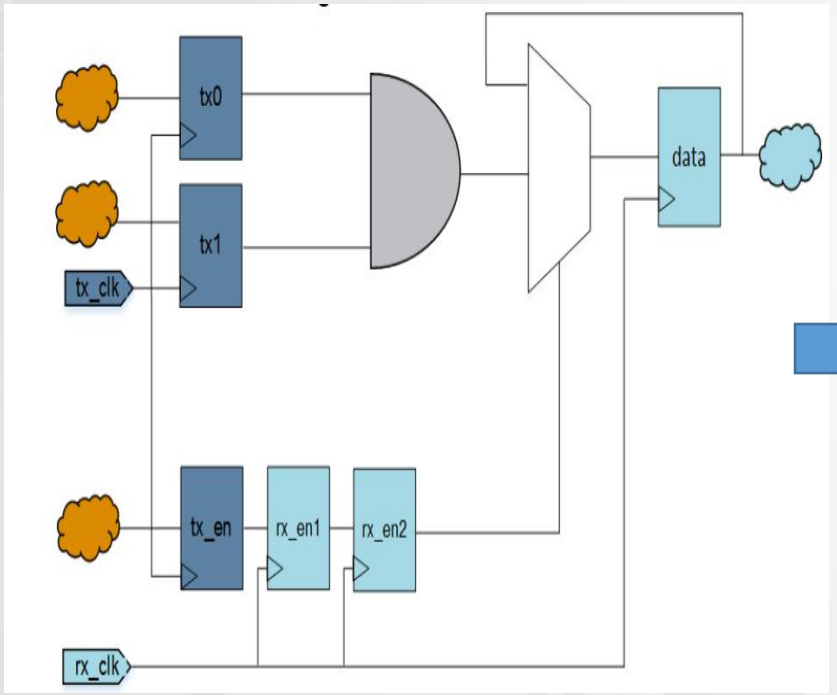
Combo-logic implementation after synthesis



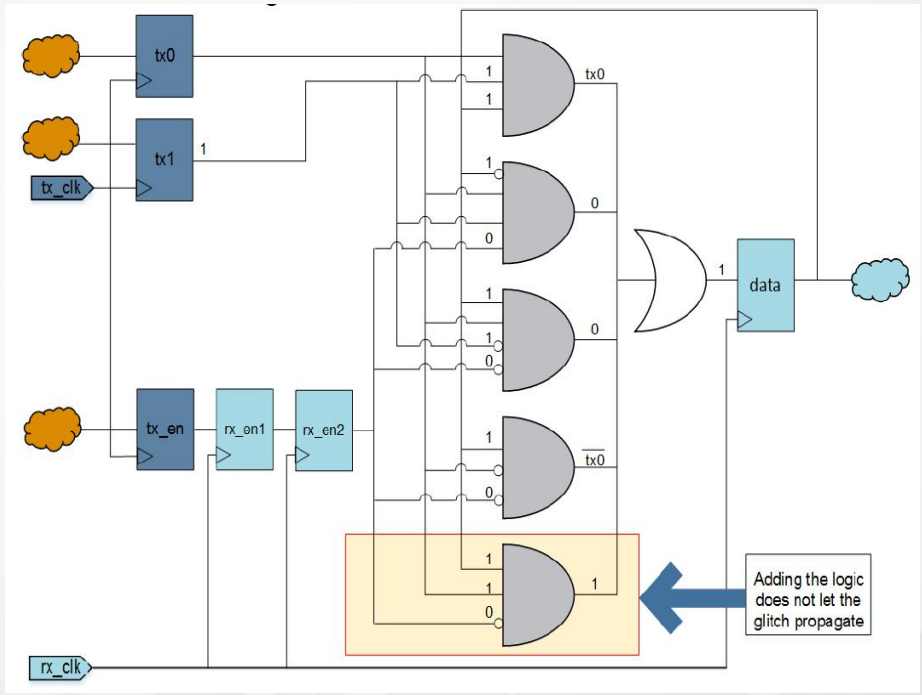
For given constants, logic reduces to (tx0|~tx0) which causes glitch

Gate-CDC Mux Synchronizer Glitch Example

RTL Logic : Mux based synchronizer

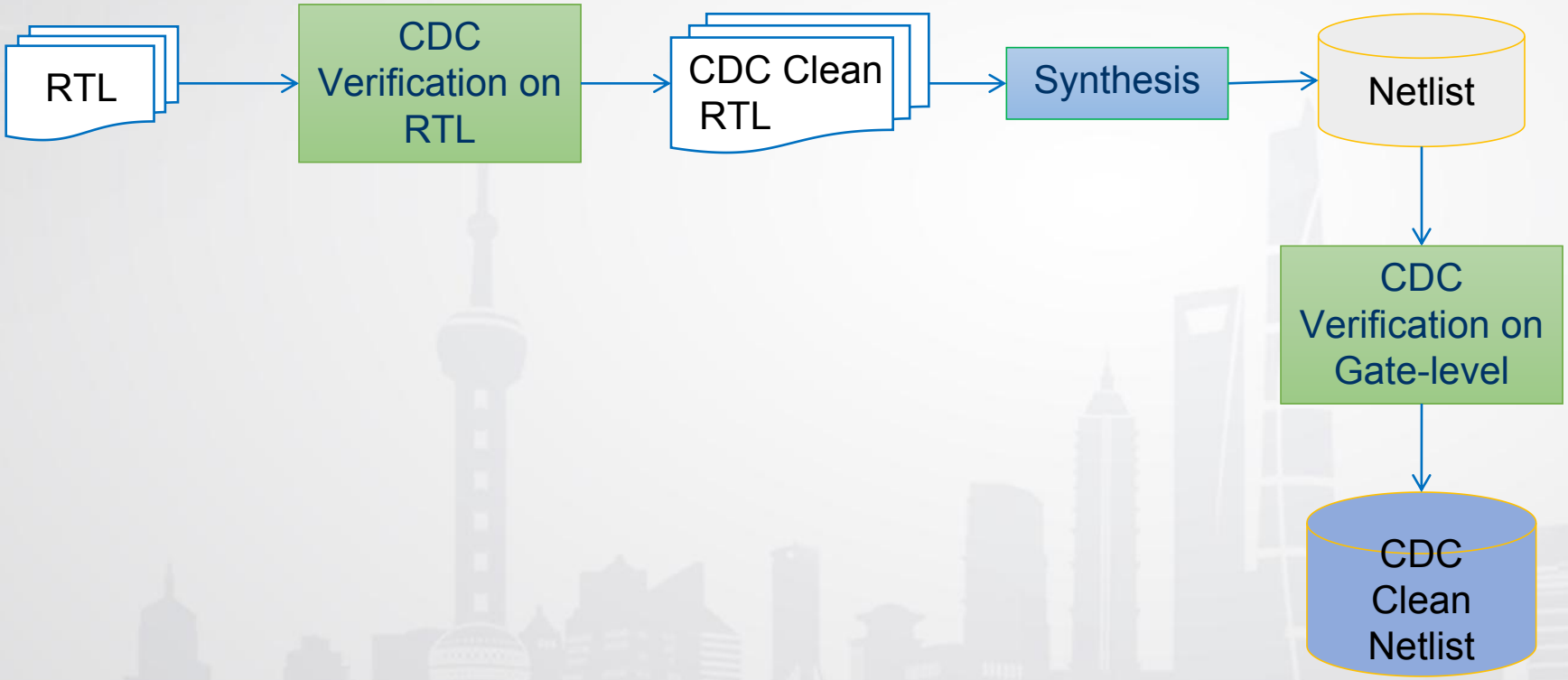


Combo-logic implementation after synthesis

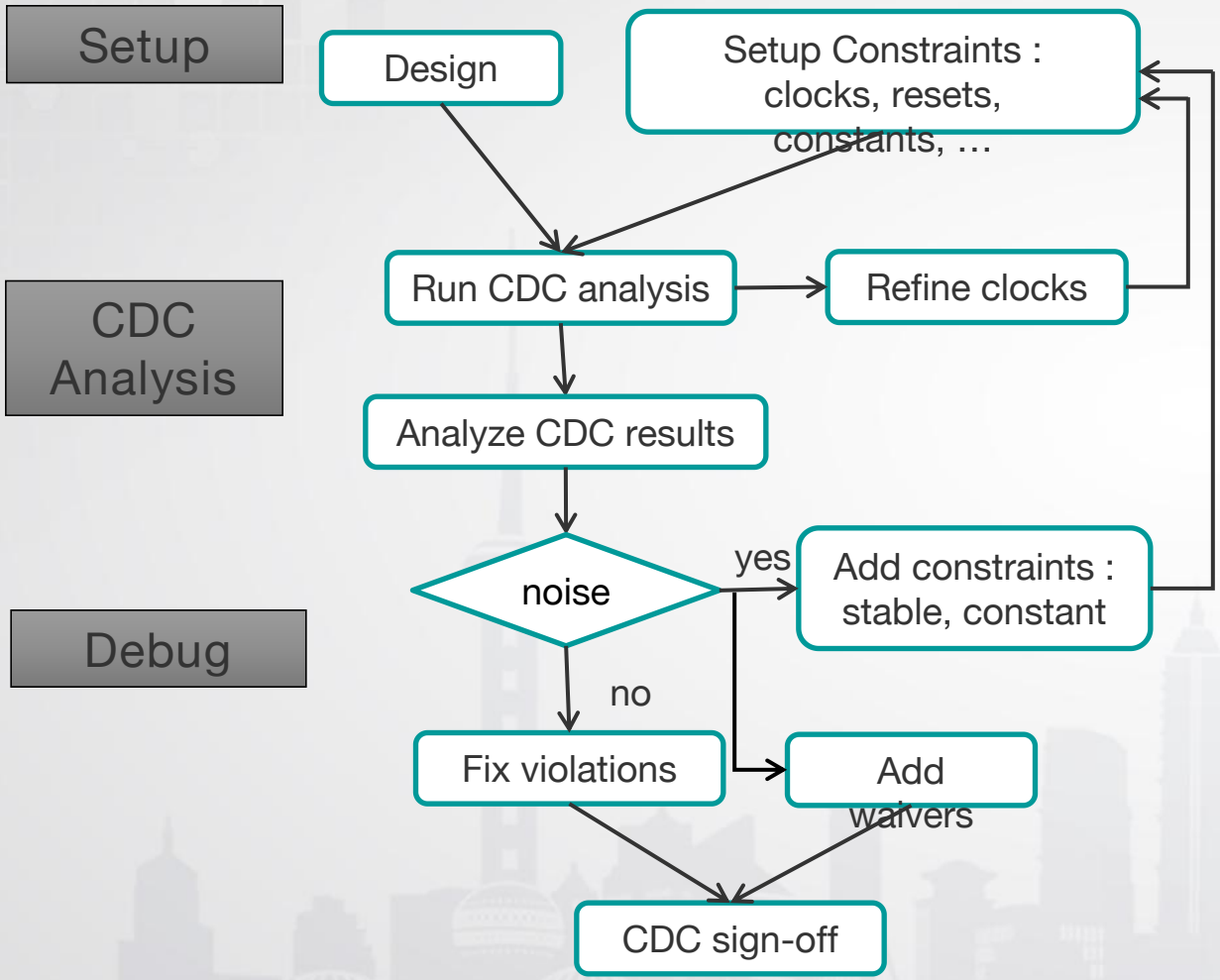


CDC Verification Flow

- CDC verification is necessary on gate-level netlist



Traditional Methodology

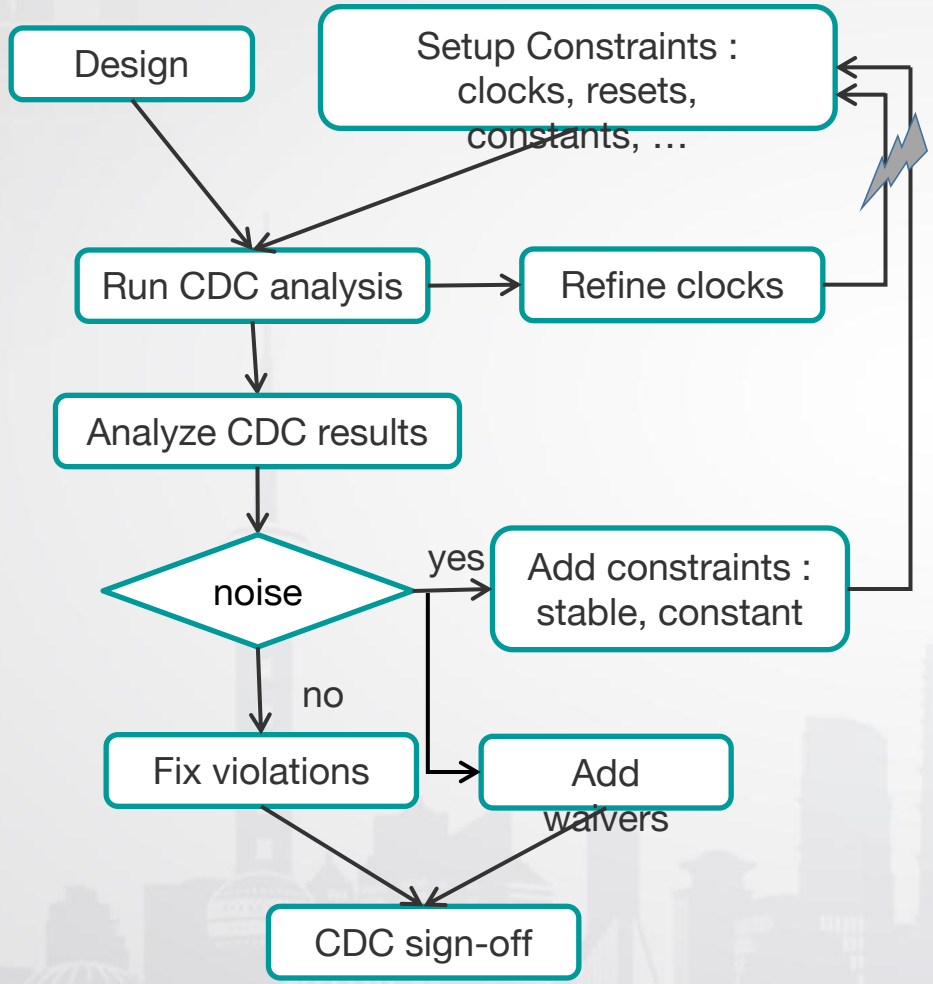


Traditional Methodology

Setup

CDC Analysis

Debug



Huge setup effort

- Manually creating constraints setup not feasible

Noise

- bit-blasted CDC paths
- Scan inserted CDC path

esis

- New hierarchies introduced

- Too many iterations needed

ber

TL

Difficult to debug

- RTL waivers do not work
- No correlation with RTL

ted

scan logic

What about glitch ?

Proposed Methodology

Setup

CDC Analysis

Debug

Gate Design

Constraints at RTL

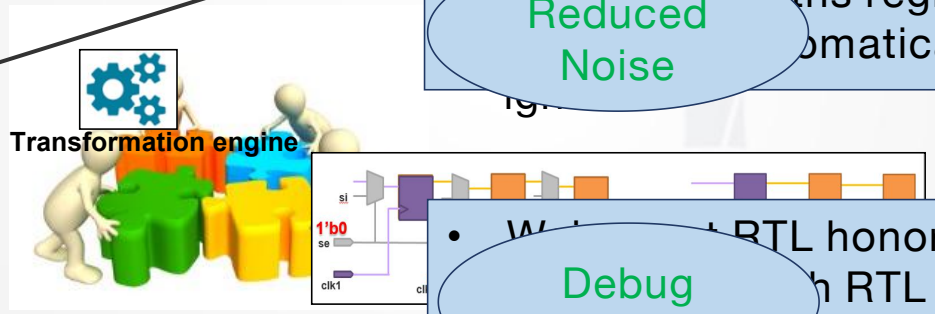
Transformation engine

Run CDC analysis

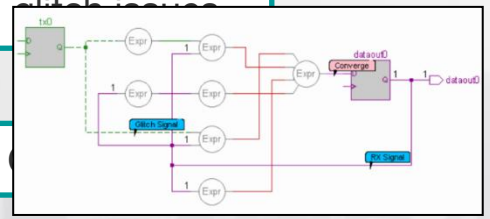
Waivers at RTL

CDC results
Glitch results

Transformation engine



Fix CDC &



Automated Setup

- RTL reused

Reduced Noise

- ths regrouped omatically

Debug Eased

- RTL honored
- h RTL signals

Glitches Detected

- nctional
- binational logic for Glitch
- Focused glitch debug

RTL Constraints Reuse

RTL Constraints

netlist blackbox

```
netlist constant cm_cdc_master
netlist constant ccc_sync0_async1_sel 1'b1
netlist constant shadow_clock 1'b0
netlist constant tap_atpg_shift 1'b0
netlist constant tap_test_mode_tdr 1'b0
netlist constant tcr_async_reset_atpg_ctrl 1'b0
netlist constant tcr_async_set_atpg_ctrl 1'b0
netlist constant tcr_cgc_atpg_ctrl 1'b0
netlist constant msm_addr_range 1'b1
netlist clock axi_clk -group AXICLK
netlist clock ddr_cc.u_core_2x_clk_mux.genblk1[1].inst0.z -group DDR2XCLK
netlist clock ddr_cc.u_core_clk_div2.inst0.clk_out -group DDR2XCLK
netlist clock ddr_cc.u_core_clk_mux.genblk1[1].inst0.z -group DDR2XCLK
netlist clock ddr2xclk -group DDR2XCLK
netlist clock memintclk -group DDR2XCLK
netlist clock cdcslogic_clk -group DDR2XCLK
netlist clock ddr_cc.runAlwaysClock_gate.genblk1[1].inst0.clk -group DDR2XCLK
netlist clock ddr_cc.intClk_gate.genblk1[1].inst0.clk -group DDR2XCLK
netlist clock ddr_cc.ddr2xClk_gate.genblk1[1].inst0.clk -group DDR2XCLK
netlist clock ddr_cc.u_clk_src.genblk1[1].inst0.z -group DDR2XCLK
```

cdc custom sync

```
qctlib_edge_detect_async_rs_ctrl
cdc custom sync qctlib_edge_detect_dftc_async_rs_ctrl -type DFF
cdc custom sync data -from edge_in -to async_edge -module qctlib_edge_detect_dftc_async_rs_ctrl
hier assume port edge_in -no_combo -module  qctlib_edge_detect_dftc_async_rs_ctrl
hier port domain reset edge_stb -clock clk -module qctlib_edge_detect_dftc_async_rs_ctrl
hier port domain async edge edge_in  -module qctlib_edge_detect_dftc_async_rs_ctrl -sync
cdc report crossing -through { smiEbi.memController.dataReadLogic.devRegData[*] } -severity waived
cdc report crossing -from *io_cal_top.pcnt_qual* -to *io_cal_top.pcnt_reg* -severity waived
cdc report crossing -from *io_cal_top.ncnt_qual* -to *io_cal_top.ncnt_reg* -severity waived
cdc report crossing -from {*io_cal_top.pcnt_reg[0]} -to {*ioc_pcnt_set[0]} -severity waived
cdc report crossing -from *io_cal_top.ncnt_reg* -to *ioc_ncnt_set* -severity waived
cdc report crossing -through msm_addr -severity waived
```

Traditional Methodology :

- 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

RTL Constraints

netlist blackbox cm_cdc_master

```
netlist constant ccc_sync0_async1_sel 1'b1
netlist constant shadow_clock 1'b0
netlist constant tap_atpg_shift 1'b0
netlist constant tap_test_mode_tdr 1'b0
netlist constant tcr_async_reset_atpg_ctrl 1'b0
netlist constant tcr_async_set_atpg_ctrl 1'b0
netlist constant tcr_cgc_atpg_ctrl 1'b0
netlist constant msm_addr_range 1'b1
netlist clock axi_clk -group AXICLK
netlist clock ddr_cc.u_core_2x_clk_mux.genblk1[1].inst0.z -group DDR2XCLK
netlist clock ddr_cc.u_core_clk_div2.inst0.clk_out -group DDR2XCLK
netlist clock ddr_cc.u_core_clk_mux.genblk1[1].inst0.z -group DDR2XCLK
netlist clock ddr2xclk -group DDR2XCLK
netlist clock memintclk -group DDR2XCLK
netlist clock cdcslogic_clk -group DDR2XCLK
netlist clock ddr_cc.runAlwaysClock_gate.genblk1[1].inst0.clk -group DDR2XCLK
netlist clock ddr_cc.intClk_gate.genblk1[1].inst0.clk -group DDR2XCLK
netlist clock ddr_cc.ddr2xClk_gate.genblk1[1].inst0.clk -group DDR2XCLK
netlist clock ddr_cc.u_clk_src.genblk1[1].inst0.z -group DDR2XCLK
netlist clock {ddr_read_dqs[3]} -group group3
netlist clock {ddr_read_dqs[2]} -group group2
netlist clock {ddr_read_dqs[1]} -group group1
```

cdc custom sync

qctlib_edge_detect_async_rs_ctrl

```
hier port domain reset edge_stb -clock clk -module qctlib_edge_detect_dftc_async_rs_ctrl
hier port domain async_edge edge_in  -module qctlib_edge_detect_dftc_async_rs_ctrl -sync
cdc report crossing -through { smiEbi.memController.dataReadLogic.devRegData[*] } -severity waived
cdc report crossing -from *io_cal_top.pcnt_qual* -to *io_cal_top.pcnt_reg* -severity waived
cdc report crossing -from *io_cal_top.ncnt_qual* -to *io_cal_top.ncnt_reg* -severity waived
cdc report crossing -from {*io_cal_top.pcnt_reg[0]} -to {*ioc_pcnt_set[0]} -severity waived
cdc report crossing -from *io_cal_top.ncnt_reg* -to *ioc_ncnt_set* -severity waived
cdc report crossing -through msm_addr -severity waived
```

Auto-generated gate constraints

netlist blackbox cm_cdc_master_0_2
netlist blackbox cm_cdc_master_0_1

```
netlist constant shadow_clock 1'b0
netlist constant tap_atpg_shift 1'b0
netlist constant tap_test_mode_tdr 1'b0
netlist constant tcr_async_reset_atpg_ctrl 1'b0
netlist constant tcr_async_set_atpg_ctrl 1'b0
netlist constant tcr_cgc_atpg_ctrl 1'b0
netlist clock axi_clk -group AXICLK
netlist clock ddr_cc.u_core_2x_clk_mux.genblk1_1_.inst0.z -group DDR2XCLK
netlist clock ddr_cc.u_core_clk_div2.inst0.clk_out -group DDR2XCLK
netlist clock ddr_cc.u_core_clk_mux.genblk1_1_.inst0.z -group DDR2XCLK
netlist clock ddr2xclk -group DDR2XCLK
netlist clock memintclk -group DDR2XCLK
netlist clock cdcslogic_clk -group DDR2XCLK
netlist clock ddr_cc.runAlwaysClock_gate.genblk1_1_.inst0.clk -group DDR2XCLK
netlist clock ddr_cc.intClk_gate.genblk1_1_.inst0.clk -group DDR2XCLK
netlist clock ddr_cc.ddr2xClk_gate.genblk1_1_.inst0.clk -group DDR2XCLK
netlist clock ddr_cc.u_clk_src.genblk1_1_.inst0.z -group DDR2XCLK
netlist clock {ddr_read_dqs[3]} -group group3
netlist clock {ddr_read_dqs[2]} -group group2
netlist clock {ddr_read_dqs[1]} -group group1
```

cdc custom sync

qctlib_edge_detect_async_rs_ctrl_16

```
hier port domain reset edge_stb -clock clk -module qctlib_edge_detect_dftc_async_rs_ctrl_16
hier port domain async_edge edge_in  -module qctlib_edge_detect_dftc_async_rs_ctrl_16 -sync
cdc report crossing -through { smiEbi.memController.dataReadLogic.devRegData_*_ } -severity waived
cdc report crossing -from *io_cal_top.pcnt_qual* -to *io_cal_top.pcnt_reg* -severity waived
cdc report crossing -from *io_cal_top.ncnt_qual* -to *io_cal_top.ncnt_reg* -severity waived
cdc report crossing -from {*io_cal_top.pcnt_reg_reg_0_.iq} -to {*ioc_pcnt_set_reg_0_.iq} -severity waived
cdc report crossing -from *io_cal_top.ncnt_reg* -to *ioc_ncnt_set* -severity waived
cdc report crossing -through msm_addr -severity waived
```

RTL Waivers Reuse

RTL Constraints

```
netlist blackbox cm_cdc_slave
netlist blackbox cm_cdc_master
netlist constant propagation
netlist constant ccc_sync0_async1_sel 1'b1
netlist constant shadow_clock 1'b0
netlist constant tap_atpg_shift 1'b0
netlist constant tap_test_mode_tdr 1'b0
netlist constant tcr_async_reset_atpg_ctrl 1'b0
netlist constant tcr_async_set_atpg_ctrl 1'b0
netlist constant tcr_cgc_atpg_ctrl 1'b0
netlist constant msm_addr_range 1'b1
netlist clock axi_clk -group AXICLK
netlist clock ddr_cc.u_core_2x_clk_mux.genblk1[1].inst0.z -group DDR2XCLK
netlist clock ddr_cc.u_core_clk_div2.inst0.clk_out -group DDR2XCLK
netlist clock ddr_cc.u_core_clk_mux.genblk1[1].inst0.z -group DDR2XCLK
netlist clock ddr2xclk -group DDR2XCLK
netlist clock memintclk -group DDR2XCLK
netlist clock cdcslogic_clk -group DDR2XCLK
netlist clock ddr_cc.runAlwaysClock_gate.genblk1[1].inst0.clk -group DDR2XCLK
netlist clock ddr_cc.intClk_gate.genblk1[1].inst0.clk -group DDR2XCLK
netlist clock ddr_cc.ddr2xClk_gate.genblk1[1].in
netlist clock ddr_cc.u_clk_src.genblk1[1].inst0.
netlist clock {ddr_read_dqs[3]} -group group3
netlist clock {ddr_read_dqs[2]} -group group2
netlist clock {ddr_read_dqs[1]} -group group1
netlist clock {ddr_read_dqs[0]} -group group0
cdc custom sync qctlib_edge_detect_dftc_async_rs_ctrl -type DFF
cdc custom sync data -from edge_in -to async_edge -module qctlib_edge_detect_dftc_async_rs_ctrl
hier assume port edge_in -no_combo -module  qctlib_edge_detect_dftc_async_rs_ctrl
hier port domain reset edge_stb -clock clk -module qctlib_edge_detect_dftc_async_rs_ctrl
hier port domain async_edge edge_in  -module qctlib_edge_detect_dftc_async_rs_ctrl -sync
cdc report crossing -through { smiEbi.memController.dataReadLogic.devRegData[*] } -severity waived
cdc report crossing -from *io_cal_top.pcnt_qual* -to *io_cal_top.pcnt_reg* -severity waived
```

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

Reduces noise and verification effort

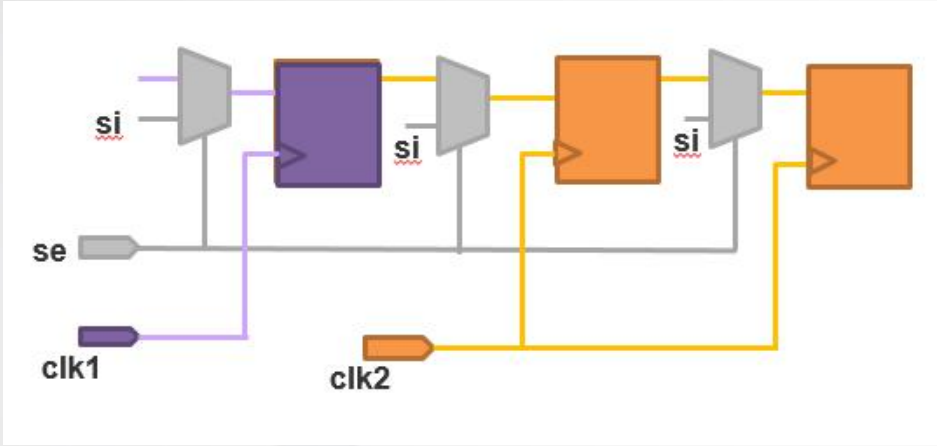
Auto-generated gate constraints

```
netlist blackbox cm_cdc_slave_1
netlist blackbox cm_cdc_master_0_1
netlist blackbox cm_cdc_master_0_2
netlist constant propagation
netlist constant ccc_sync0_async1_sel 1'b1
netlist constant shadow_clock 1'b0
netlist constant tap_atpg_shift 1'b0
netlist constant tap_test_mode_tdr 1'b0
netlist constant tcr_async_reset_atpg_ctrl 1'b0
netlist constant tcr_async_set_atpg_ctrl 1'b0
netlist constant tcr_cgc_atpg_ctrl 1'b0
netlist clock axi_clk -group AXICLK
netlist clock ddr_cc.u_core_2x_clk_mux.genblk1_1_.inst0.z -group DDR2XCLK
netlist clock ddr_cc.u_core_clk_div2.inst0.clk_out -group DDR2XCLK
netlist clock ddr_cc.u_core_clk_mux.genblk1_1_.inst0.z -group DDR2XCLK
netlist clock ddr2xclk -group DDR2XCLK
netlist clock memintclk -group DDR2XCLK
netlist clock cdcslogic_clk -group DDR2XCLK
netlist clock ddr_cc.runAlwaysClock_gate.genblk1_1_.inst0.clk -group DDR2XCLK
                                                              clk -group DDR2XCLK
                                                              0.clk -group DDR2XCLK
                                                              -group DDR2XCLK
netlist clock {ddr_read_dqs[3]} -group group3
netlist clock {ddr_read_dqs[2]} -group group2
netlist clock {ddr_read_dqs[1]} -group group1
netlist clock {ddr_read_dqs[0]} -group group0
cdc custom sync qctlib_edge_detect_dftc_async_rs_ctrl -type DFF
cdc custom sync data -from edge_in -to async_edge -module qctlib_edge_detect_dftc_async_rs_ctrl_16
hier assume port edge_in -no_combo -module  qctlib_edge_detect_dftc_async_rs_ctrl_16
hier port domain reset edge_stb -clock clk -module qctlib_edge_detect_dftc_async_rs_ctrl _16
hier port domain async_edge edge_in  -module qctlib_edge_detect_dftc_async_rs_ctrl_16 -sync
cdc report crossing -through { smiEbi.memController.dataReadLogic.devRegData_*_ } -severity waived
```

cdc report crossing -from
{*io_cal_top.pcnt_reg_reg_0_.iq}
-to {*ioc_pcnt_set_0_.iq} -severity waived

Auto Infer Test Mode Settings

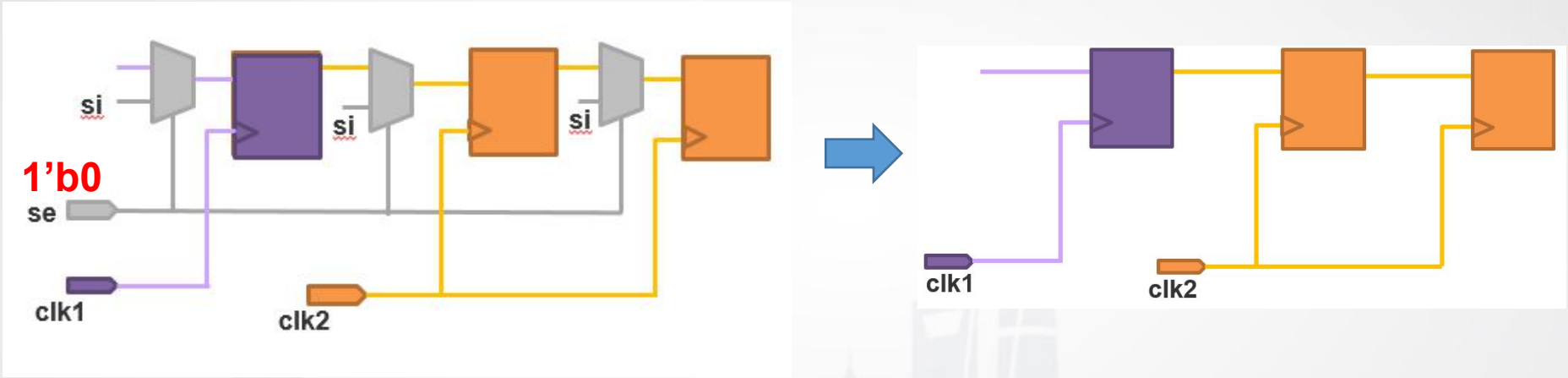
- Scan mux inserted CDC path



- Traditional methodology
  - Synchronizer not detected

Auto Infer Test Mode Settings

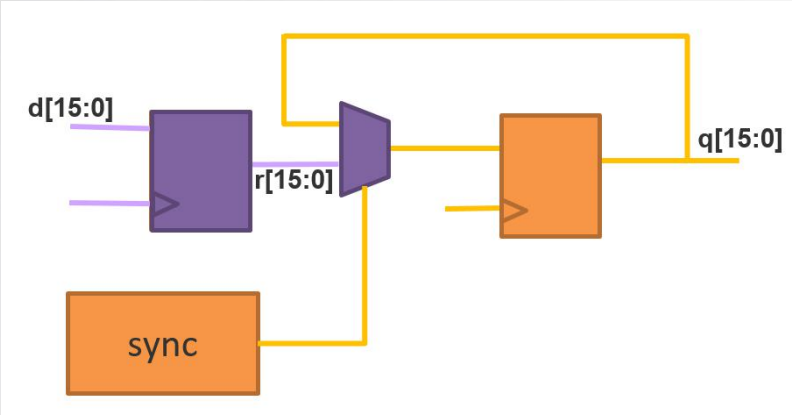
- Scan mux inserted CDC path



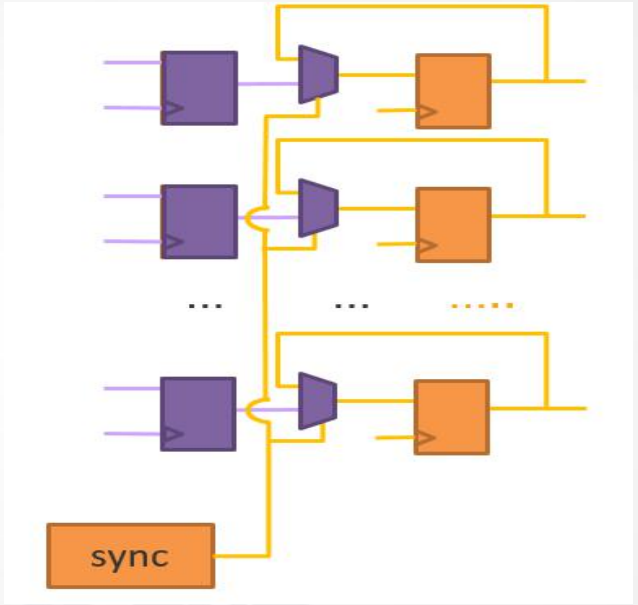
- Proposed methodology
  - Structural analysis detects scan enable settings
  - Two dff synchronizer detected

Regroup Bit-blasted Crossings

RTL : 16-bit data path



Gate-level : 16 separate 1-bit paths

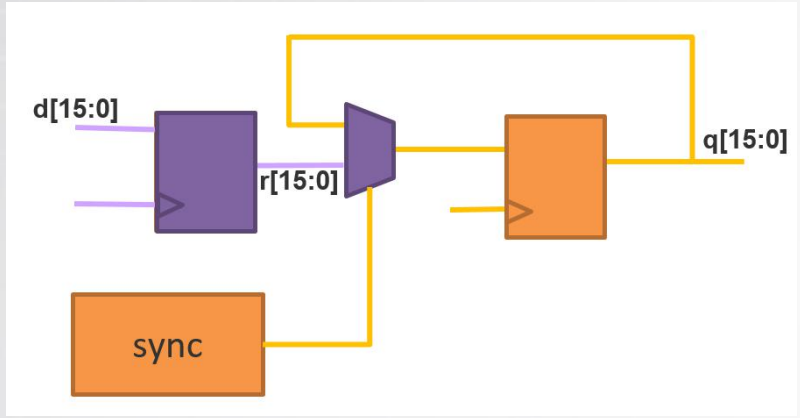


- Traditional methodology
  - 16 separate 1-bit paths reported
  - r_reg_0.iq -> q_reg_0.iq, r_reg_1.iq -> q_reg_1.iq, …

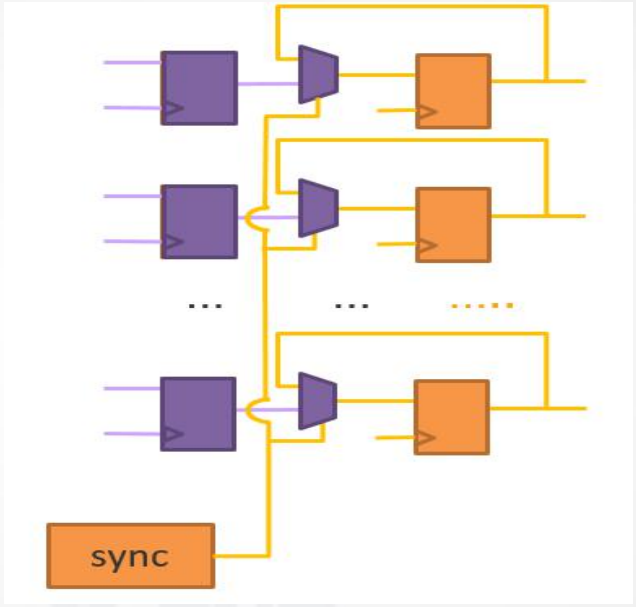
Crossing count multiplied by number of vector signal bits

Regroup Bit-blasted Crossings

RTL : 16-bit data path



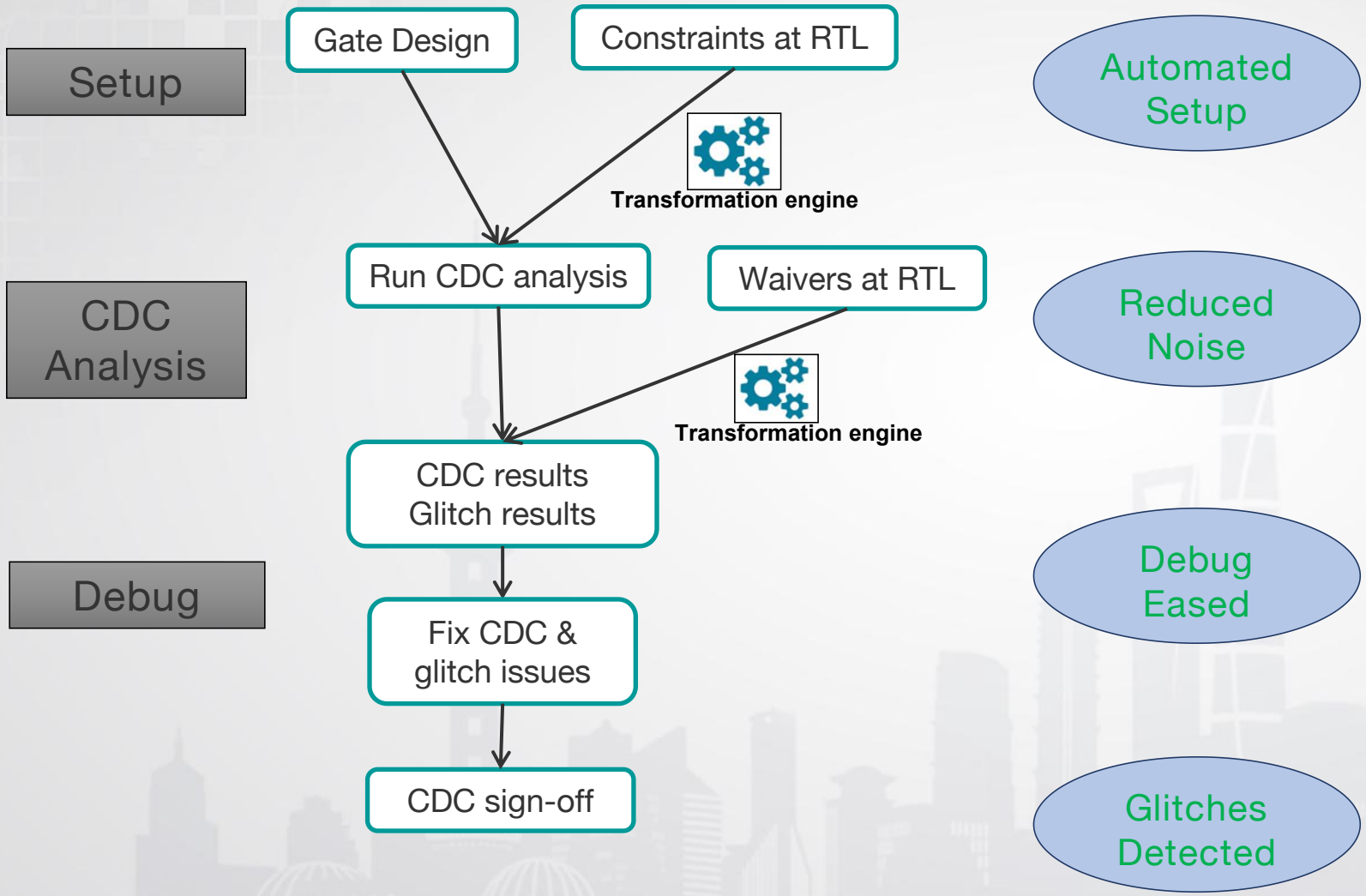
Gate-level : 16 separate 1-bit paths



- 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



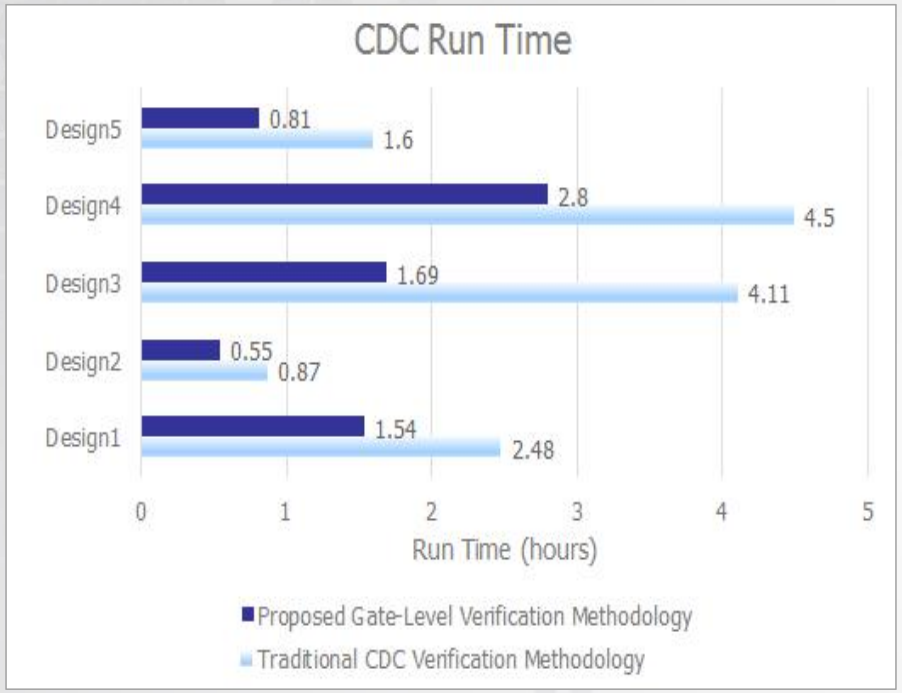
Automated Setup

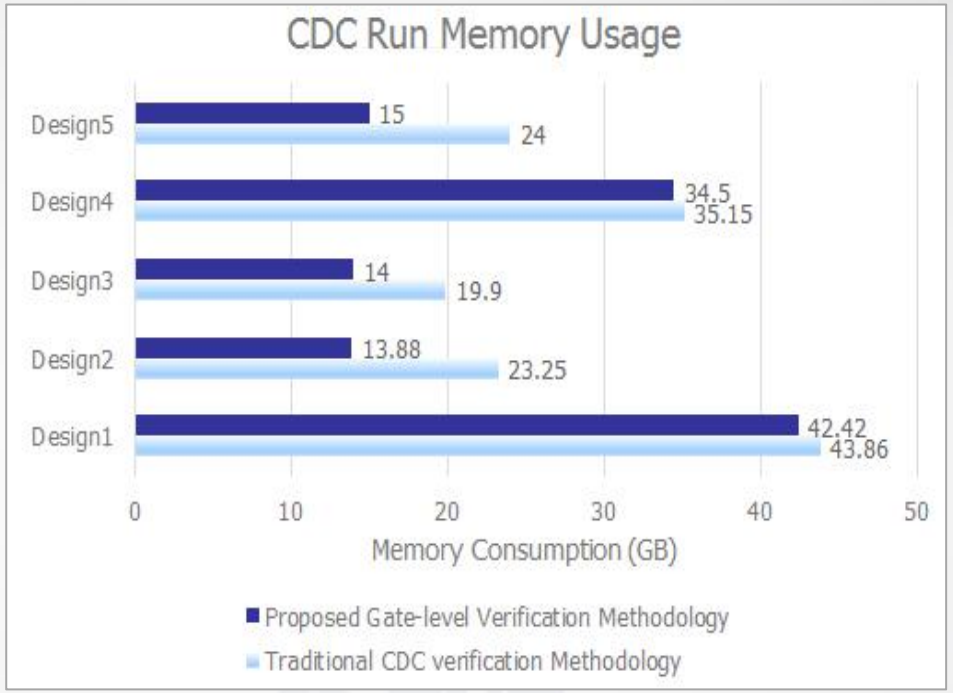
Reduced Noise

Debug Eased

Glitches Detected

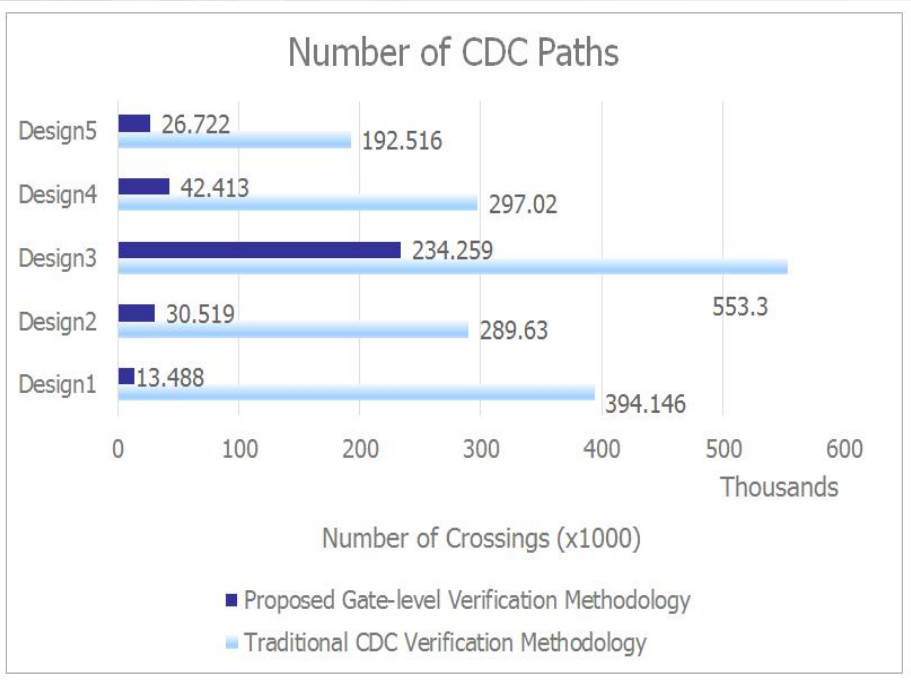
Experiment Results





- 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

2021 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION CHINA
SHANGHAI | MAY 26, 2021

accellera SYSTEMS INITIATIVE

Thank You