# Intelligent Approach to Shift-Left the Identification of CDC Convergence Issues During RTL Simulation

Charles Lou, Tenafe, California, USA (clou@tenafe.com)

Susan Xie, Synopsys, Beijing, China (xiedh@synopsys.com)

***Abstract*—Chip design involves complex processes and requires careful consideration of many factors. The designer must address clock domain crossing (CDC) Convergence. CDC Convergence issue occurs when a signal cross from one clock domain to another; it happens when multiple correlated signals are separately synchronized, or a single signal is synchronized more than once. The root cause is that these separated synchronization paths have different data capture timing; it may lead to different clock cycle delays when transferred through the clock domain, which can have significant consequences for the chip's functioning. Unlike Logical bugs, CDC Convergence issues are almost impossible to find in RTL simulation, and their discovery in Gate Level simulation is often just inadvertent. Moreover, CDC convergence issues discovered after the RTL freeze will be time and resource-consuming and, in some cases, impossible to fix in netlist. In this paper, we will present the Dynamic CDC Jitter flow, some shift-left RTL simulation techniques, demonstrate specific CDC convergence scenarios, and how this flow help to catch the real CDC convergence issue during RTL Simulation. Chip designers must carefully design RTL from structural and functional perspectives to address the CDC convergence issues. This requires static signoff tools and techniques to mitigate the effects of convergence. Additionally, designers must ensure that their designs meet industry standards and are thoroughly tested to verify their performance and reliability.***

***Keywords*—CDC Convergence; Dynamic CDC Jitter methodology**

## I. INTRODUCTION

Clock Domain Crossing (CDC) refers to the scenario in chip design where signals are transferred from one clock domain to another. A clock domain is a chip region that operates based on a specific clock signal. Each clock domain has its clock source and timing characteristics.

When signals cross from one clock domain to another, it introduces potential timing issues due to differences in clock frequencies, phase misalignment, and metastability. These timing issues can result in data loss, data corruption, or even functional failure of the chip.

The primary cause of CDC issues is that different clock domains operate independently, meaning they have different clock frequencies, clock phases, and timing constraints. As a result, when a signal transitions from one clock domain to another, there is a possibility that the receiving domain's clock edge may not align with the transitioning signal, leading to improper sampling.

Several techniques and methodologies are employed during chip design to address CDC issues. Here are some common approaches:

1. Synchronization: Synchronizers ensure proper data transfer between clock domains. A synchronizer consists of a pair of flip-flops, one in the source domain and another in the destination domain. It helps capture the data from the source domain and synchronize it with the destination domain's clock. Synchronization introduces additional delay but minimizes the risk of metastability.

2. Asynchronous FIFOs: First-In-First-Out (FIFO) buffers with asynchronous clock domains are employed to manage data transfer between different clock domains. These FIFOs help maintain data integrity by providing synchronization and buffering capabilities.

3. Gray coding: Gray coding encodes control or status signals that cross clock domains. Gray coding ensures that only one-bit changes at a time during transitions, reducing the likelihood of multiple bits changing simultaneously, which can lead to metastability.

4. Multi-Register Synchronizers: In cases where a signal needs to cross multiple clock domains, a chain of synchronizers can be used to capture and synchronize the data at each stage. This helps propagate the data through multiple clock domains while ensuring proper synchronization.

5. Clock Domain Crossing Analysis: Specialized tools and methodologies are employed to analyze the design and identify potential CDC issues. These tools help identify critical paths, potential metastability issues, and setup and hold time violations.

It's important to note that CDC design is an iterative process involving careful analysis, simulation, and verification. Designers need to ensure that the necessary constraints and techniques are applied correctly to mitigate CDC issues and guarantee the reliable operation of the chip across different clock domains.

## II. TENAFE CHALLENGES

CDC (clock domain crossing) check is a significant flow to ensure the silicon can work properly. The clocks from different clock domains may reach different flip-flops during the design run at different times in each cycle. This timing uncertainty may cause random setup or hold timing violations. It will make the silicon fail, and it's tough to debug on the silicon due to some limitations of RTL simulations, FPGA simulations, and even gate simulations that can't detect all CDC issues. We need some tools to analyze the design at the RTL level or netlist level.

Tenafe encountered a CDC issue in our first-generation SSD controller chip, and we noticed it in the later stage of the development cycle (during GLS); Merak silicon issue (data mismatch, no wanted data got latched), design issue and debug for two months. => ECO (diverged two signals into one signal). Details are shown below in Figure 1:

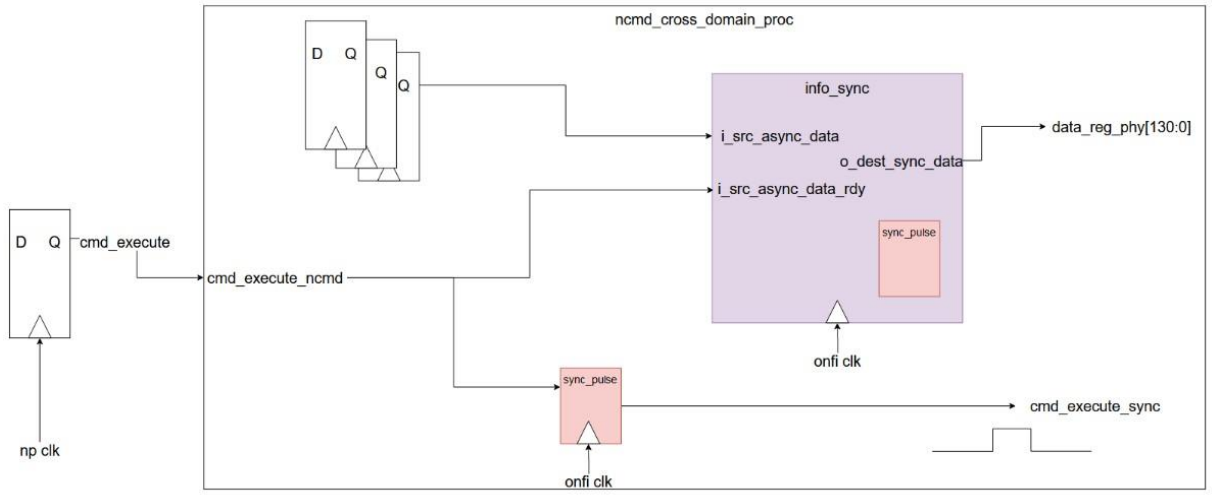


Figure 1. CDC convergence issue on ‘cmd_execute_ncmd’

Use ‘cmd_execute_ncmd’ CDC path as an example:

1. Failure caught on silicon when cmd_execute_sync arrived one cycle earlier or later than data_reg_phy; silicon failed.
2. Root cause was determined to be as ‘cmd_execute_ncmd’ being synchronized separately on the control path ‘cmd_execute_sync’ and data path ‘data_reg_phy’, which resulted in a typical CDC coherency issue and not captured by normal RTL simulation and Gate level simulation.

## III. How Synopsys CDC Jitter Flow Helped Us?

There was no good solution before we adopted Synopsys CDC Jitter Flow. We were purely relying on design review, as stated earlier, it is hard to debug on the silicon due to limitations of RTL simulations, FPGA simulations, and even Gate level simulations that cannot detect all CDC issues.

### *A. CDC Jitter Verification Approach Adopted by Tenafe*

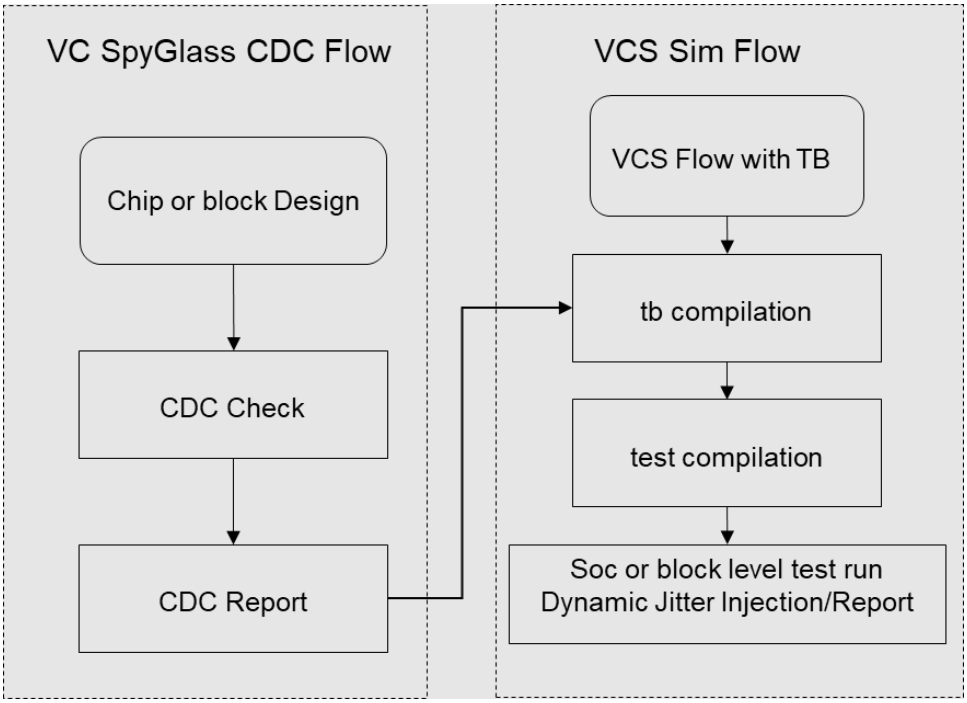


Figure 2. CDC Jitter Flow Adopted in Tenafe

As described in Figure 2, we run VC SpyGlass CDC Jitter flow in two stages:

At first, we run VC SpyGlass CDC static check flow; it will do the structural check based on RTL design and corresponding SDC constraint; do synchronization and convergence analysis; after that, it will write out CDC Paths corresponding Database, which contains detailed Clock Domain Crossing information and can be used as input for the 2nd Stage Flow.

Then, we feed the CDC database into Synopsys VCS. During functional simulation using VCS, the impact of jitter injection over corresponding critical CDC Paths on various testcases is analyzed. If there are convergence issues in design, then the random jitter injection on the CDC path may lead to simulation failure or data corruption occurring during the RTL stage simulation.

With the 1st Stage VC SpyGlass structural analysis flow, the RTL designer can review the result to make sure all the essential synchronization has been structurally done in their design and then further examine the complex violations such as CDC convergence or glitch issues; some simple and obvious CDC convergence issue can be found during the CDC result review directly; but if the CDC path related design structure is complex to be analyzed through schematic review, then locating the actual CDC convergence violation is “needle in a haystack” kind of challenge. Progressing to the second stage, the Verification engineer validates the RTL design with CDC jitter modeling mimicking the actual silicon conditions using VCS Dynamic Jitter simulation flow.

### *B. How does CDC Jitter Flow identify the specified CDC issues*

Combined with the circuit structure shown in Figure 1, let's see how CDC Jitter Flow can help us identify actual bugs. Figure 3 is the RTL simulation waveform with CDC Jitter Injection; as you can see for asynchronous signal ‘cmd_execute_ncmd’, the

simulation behavior of its two different synchronizers ‘sync_pulse_async_data_ready’ and ‘u_cmd_execute_sync’ is exactly same. Based on this, RTL simulation results are as expected, data sampling is correct, and the test case finishes smoothly.

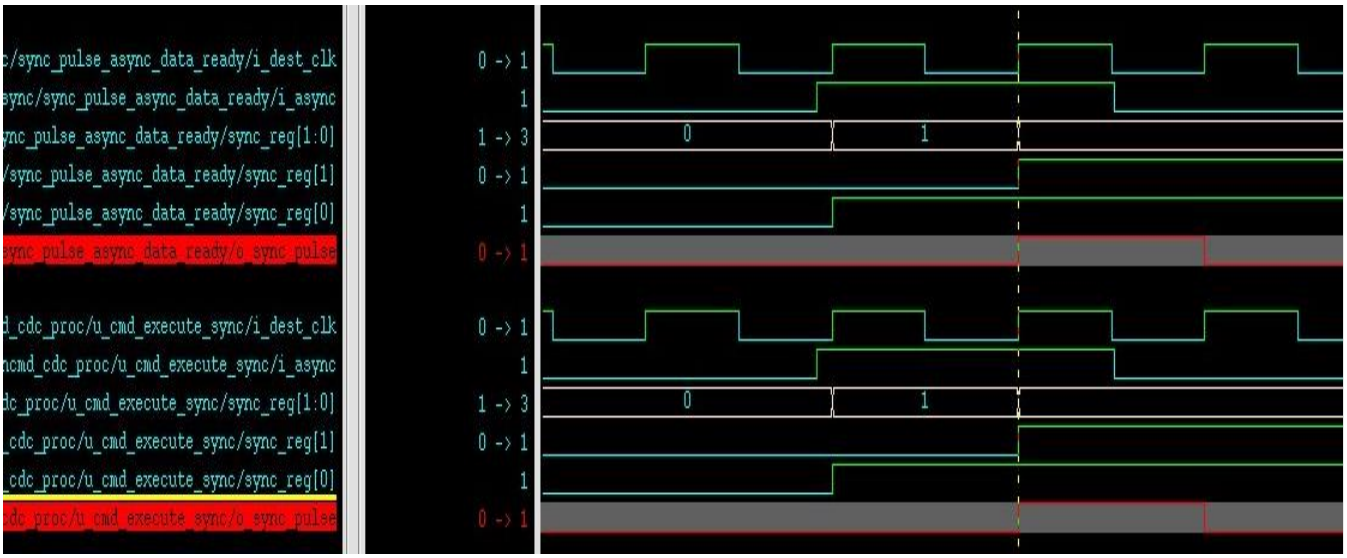


Figure 3. RTL Simulation waveform without CDC Jitter Flow Injection

Figure 4 and Figure 5 illustrate the RTL simulation waveform with CDC Jitter Injection. As shown in these figures for asynchronous signal ‘cmd_execute_ncmd’, the simulation behavior of its two different synchronizers ‘sync_pulse_async_data_ready’ and ‘u_cmd_execute_sync’ is different due to the random CDC Jitter injection. The delayed jitter injected on synchronizer ‘u_cmd_execute_sync’ causes its output to be misaligned with ‘sync_pulse_async_data_ready/o_sync_pulse’. The corresponding RTL simulation results failed since the data sampling was wrong, and the whole test case hung. The CDC Jitter flow helped to re-create the silicon bug scenario in the RTL phase using the existing testbench and testcases, dramatically improving the efficiency of finding real silicon issues.

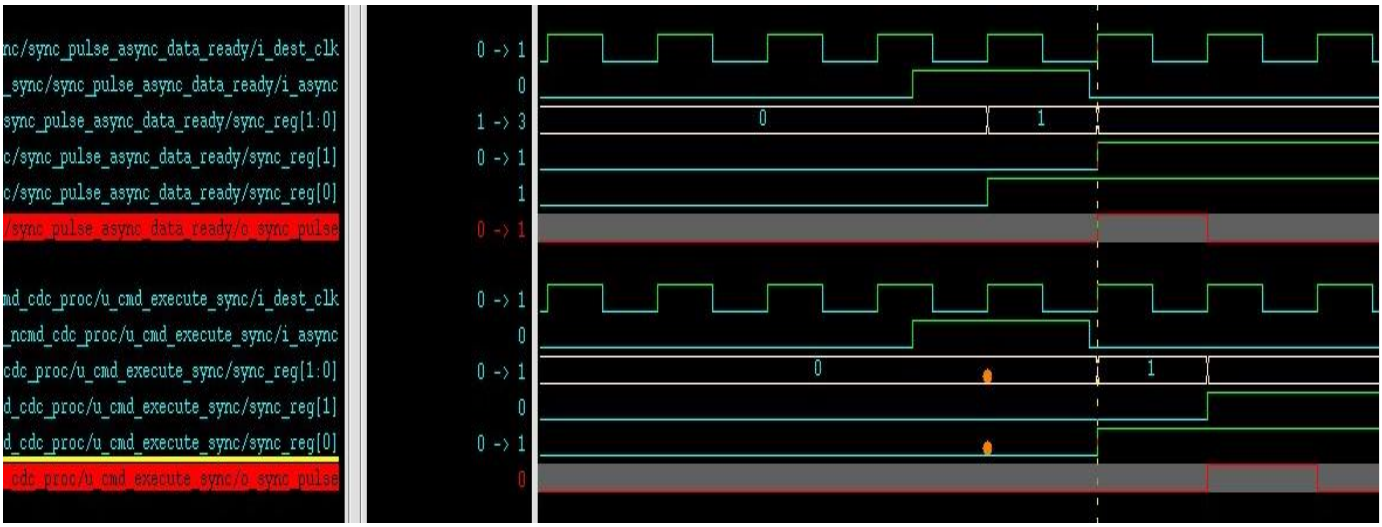


Figure 4. RTL Simulation waveform (1) with CDC Jitter Flow Injection

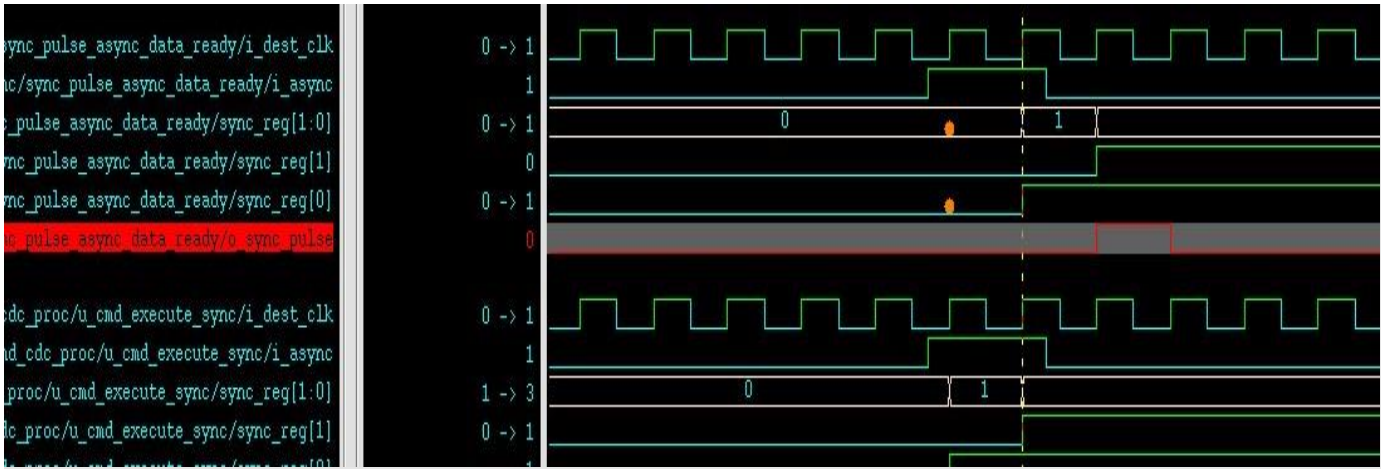


Figure 5. RTL Simulation waveform (2) with CDC Jitter Flow Injection

As the above waveform demonstrates, CDC Jitter Simulation can randomly insert jitter based on the Jitter Database generated by VC SpyGlass CDC; meanwhile, VCS can print out the CDC Jitter Injection related debug information as Figure 6 demonstrated. Designers could visualize the jitter injection timestamp for specific CDC paths. This visualization aligns with the simulation waveform debug aid—the orange dot. All these debug tips make the CDC Jitter injection result review easy and effective.

```
dyncdcdebug> TIME=30,180989254 jitter_inject, cdc=4939 synch_changed=0 jitter_type=no_jitter
dyncdcdebug> TIME=30,180989254 jitter_inject, cdc=4946 synch_changed=0 jitter_type=no_jitter
dyncdcdebug> TIME=30,180989254 jitter_inject, cdc=4948 synch_changed=0 jitter_type=delayed
dyncdcdebug> TIME=30,180989254 jitter_inject, cdc=4955 synch_changed=0 jitter_type=delayed
dyncdcdebug> TIME=30,180989254 jitter_inject, cdc=4957 synch_changed=0 jitter_type=delayed
dyncdcdebug> TIME=30,180989254 jitter_inject, cdc=4963 synch_changed=0 jitter_type=no_jitter
dyncdcdebug> TIME=30,180989254 jitter_inject, cdc=4966 synch_changed=0 jitter_type=delayed
dyncdcdebug> TIME=30,183716593 jitter_inject, cdc=3747 synch_changed=0 jitter_type=delayed
-:---  cdc_jitter.log   Top of 21M  (1,0)      (Fundamental) Fri Apr 16 8:02AM 1.24
```

Figure 6. CDC Jitter Flow Injection Debug Information

## IV. CDC JITTER FLOW METHODOLOGY

The design issue mentioned in Section II can be extracted as below Figure 10 — CDC convergence issue with different depths; VC SpyGlass CDC structural checks can also flag these types of issues, but the schematics of CDC convergence could become very complicated if it is deep convergence; which means the synchronizers go through several sequential logic before final convergence. This raises the need for dynamic simulation with metastability injection on the CDC path, which can help to find real issues by introducing meta-stability effects on CDC paths in RTL simulation; If the CDC convergences are not effectively managed in design, verification regression tests should fail. In this way, CDC Jitter flow helps to eliminate the CDC convergence issue on real silicon.

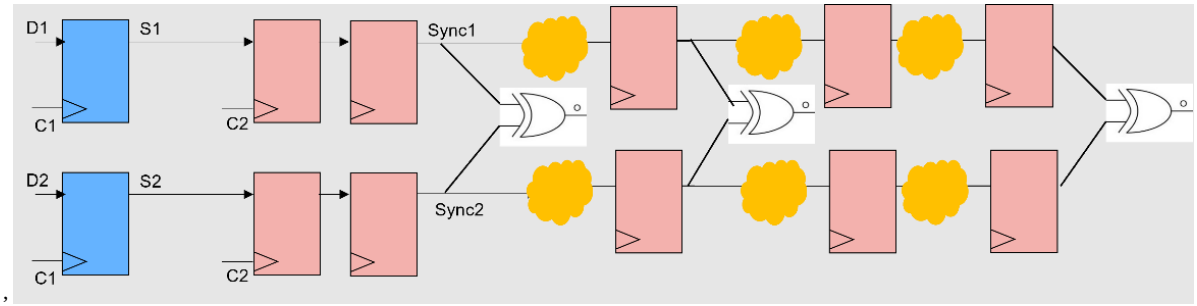


Figure 10. CDC Convergence with different depth

*A. General flow adopted by Tenafe*

The CDC Jitter flow adopted is simple and can be divided into two parts, as shown in Figure 11.

1. VC SpyGlass CDC structural analysis
    a. Generate database for CDC jitter modeling.
    b. Generate run-time configuration file.
2. Use VC SpyGlass generated Jitter DB with VCS
    a. Injects CDC Jitter on specific CDC paths in runtime without noise.
    b. Provide coverage report for each CDC path and related convergence group.
    c. Minimal overhead of jitter analysis on performance.

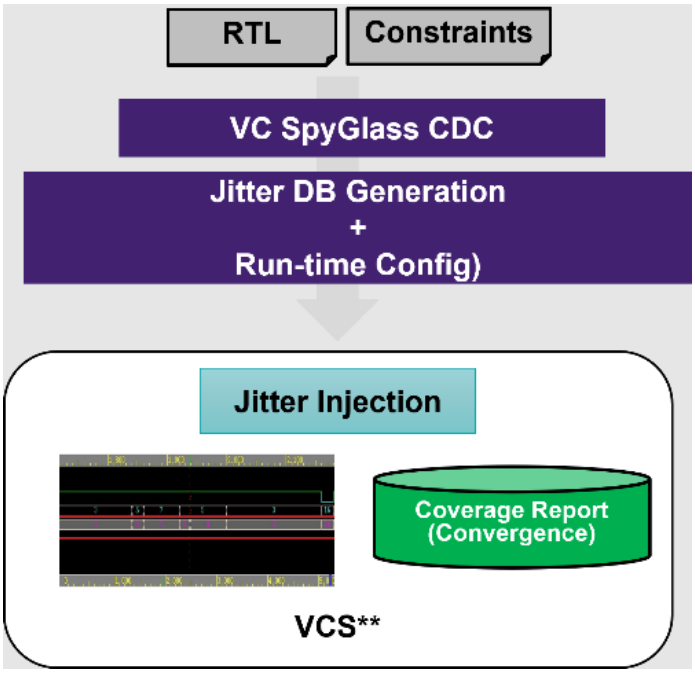


Figure 11. Dynamic CDC Jitter Flow

*B. Scripts for CDC Jitter Flow*

vc_cdc.tcl

```
set_app_var enable_cdc true
...
set search_path "<space separated list of directories >"
set link_library  "<space separated list of libs>"

analyze -format  vhdl -work lib_name file.vhd
analyze -format  verilog -work lib_name file.v
...
elaborate  -work libname DUT

read_sdc top.sdc
source plus.tcl

check_cdc -type jitter
report_cdc
```

vcs_cmp.sh

```
vcs -o simv -debug_access+all -kdb
+vcs+fsdbon <sourcefiles>
-dynamic_cdc
-dynamic_cdc_db=<path-tomonitor-database-file>
-dynamic_cdc_convergence
-sverilog -full64 -Xdyncdc=0x40
```

vcs_sim.sh

```
setenv FSDB_JITTER 1
setenv DYNAMIC_CDC_DEBUG 1

simv -dynamic_cdc -verdi &
```

Figure 12. Dynamic CDC Jitter Script of VC SpyGlass and VCS

*C. Coverage of CDC Jitter Flow*

Figure 13 shows how VCS collects, organizes, and presents coverage-related results during the VCS Dynamic Jitter simulation:

1. Firstly, the tool will collect the toggle statistics of CDC paths and then further calculate whether there are multiple signals toggle simultaneously for each structural CDC convergence group.
2. If multiple signals toggle simultaneously, the probability of inserting a jitter in them is further calculated.
3. If no multiple signals toggle simultaneously, the designer must confirm whether multiple signals toggle simultaneously will not be accurate due to function. If not, the designer may need to update the testbench to fix the lack of coverage.

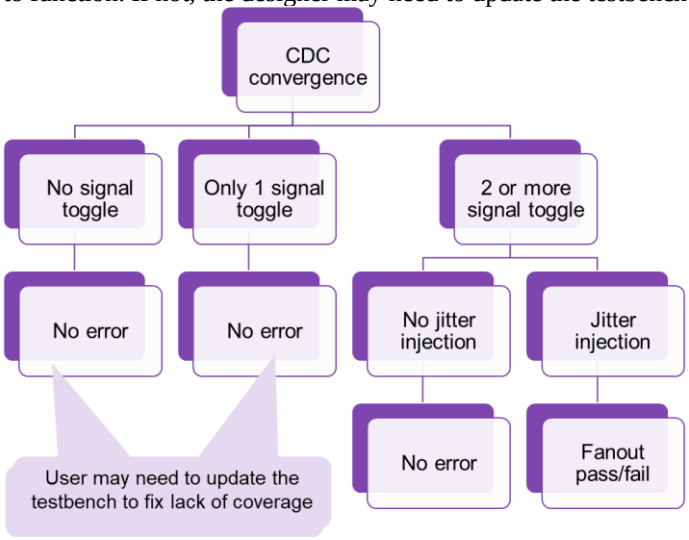


Figure 13. Dynamic CDC Jitter Script of VC SpyGlass and VCS

## V. SUMMARY

The CDC Jitter Flow leverages production-proven VC SpyGlass CDC technology and native jitter injection with VCS to deliver complementary verification methods for CDC coherency/convergence paths during the design and verification cycle. Combining these two technologies provides lower noise and highly accurate jitter injection results than conventional randomization of simulation models for CDC synchronizers. Designers can determine if there is a need to develop new testcases by analyzing the jitter injection coverage achieved during CDC convergence verification. Tenafe successfully reproduced the real silicon problems through this approach, shifted the verification of related CDC convergence issues from the Silicon debug to the RTL simulation stage, and advanced the corresponding design versions from Metal ECO to RTL Coding. It provides an intelligent solution to avoid the complex CDC convergence issues found initially in silicon.

## VI. ACKNOWLEDGEMENT

The author would like to thank Anshu Malani, Sudeep Mondal, Rimpy Chugh, Amit Goldie, Navneet Kumar Chaurasia who are VC SpyGlass CDC and Dynamic Jitter experts in Synopsys, and Haibo Zhu, Hanjun Zhang from Tenafe, for valuable feedbacks during flow deployment.

## VII. REFERENCE

[1] Synopsys. (2023) VC SpyGlass Clock Domain Crossing LCA Features Guide, version U-2023.03.

[2] Synopsys. (2023) VC Spyglass CDC Jitter Flow