Clock Domain Crossing Verification in Transistor-level Design

Hyungjung Seo, KwangSun Kim, YoungRok Choi, Jihwan Kim, Jong-Bae Lee
Samsung Electronics Inc, Hwaseong-si, Gyeonggi-do, 18448, Koera
hjung.seo, kwsun, yrok.choi, join.kim@samsung.com

***Abstract*- A synchronous design has a clock source to synchronize all sequential elements. Most of modern VLSI designs contain multiple clocks which have asynchronous relationship among them to model complex behavior. Asynchronous clocks generate lots of timing problems which cannot be verified by traditional methodologies. Clock Domain Crossing (CDC) is one of the challenging verification works. Even though verifying CDC problem has been investigated by many researchers and EDA vendors, its main target is limited in Register-Transfer Level (RTL) design. In this paper, CDC problem in transistor-level is covered and we first propose a CDC analyzer based on commercial Transistor-level Static Timing Analysis (Tr-STA) tool. However, the CDC analysis produces poor results in terms of performance and coverage, tool for only Transistor-level CDC (Tr-CDC) problem is necessarily. As a result, Tr-CDC analyzer is proposed and experimental results show the proposed Tr-CDC analyzer extracts CDC points that cover the entire area of target design within less than 5 hours.**

I. Introduction

Clock Domain Crossing (CDC) is one of the most challenging issues in IC design [1]. Many EDA companies support CDC analysis in Register Transfer Level (RTL) or gate-level design and most of System-On-Chip (SOC) engineers can verify CDC problem in their design. However, engineers who design their circuits in transistor-level suffer from lack of CDC analysis tool. Dynamic Random Access Memory (DRAM) is the representative circuit that is designed in transistor-level. CDC problem of DRAM is continuously increased as specification of device becomes complex because of function and power requirements. In this paper, we propose a systematic CDC analysis approach to verify CDC problem in transistor-level based on commercial Transistor-level Static Timing Analysis (Tr-STA) tool, called NanoTime, developed by Synopsys. Furthermore, limitations of commercial tool based approach are addressed. With the lessons of the proposed commercial tool based approach, we finally propose Transistor-level CDC tool (Tr-CDC) which can analyze CDC problems in the transistor-level circuit.

II. Backgrounds

A. Metastability

Metastability is the signal that is not stable 0 or 1. The signal is generated when there's not enough setup/hold time for signal transition. Figure 1 shows latch setup time violation and metastable output signal. Time gap between L2/D and CLK2 produces setup time violation, thus output of L2 cannot generate signal transition from 0 to 1. Metastability can be avoided by Static Timing Analysis, but asynchronous clock in design forces designer cannot guarantee fault-free timing analysis because of its own nature. Communications among asynchronous clocks should be checked by Clock Domain Crossing check.

B. Clock Domain Crossing

Clock Domain Crossing (CDC) point is a latch or flip-flop point where its timing path is constructed by asynchronous clocks. Figure 1 is simple CDC point when CLK1 and CLK2 have asynchronous relation. The source of CLK1 and CLK2 are different and they are not governed by global clock source. Since their timing characteristics are not synchronized, circuit behaviors or even variation can vary their timing relations and disturb timing signoff. Commercial tools like Spyglass CDC of Synopsys [2] or Meridian CDC of Real Intent [3] give static CDC analyzing solution to check all CDC points in design and designers should include special logic to remove CDC problems, permanently. The logic called "Synchronizer" is generally composed of serially connecting two flip-flops and there are lots of variations for removing CDC problems for different structures. Figure 2 is two flip-flop synchronizer and its timing behavior. Serially connecting flip-flop rectifies metastable signal.

C. Synchronizer for Multi bits Signal

The synchronizer in Figure 2 cannot be used when CDC issues are generated on bus connection, thus different types of synchronizers must be designed. Reference [4] describes Gray encoding for multi bits signal shown inside Figure 3. Synchronizing multi bit signal is challenging work. Gray encoding transform multi bit signal transition problem to single bit signal transition problem. Single bit signal transition problem is a simple CDC problem and

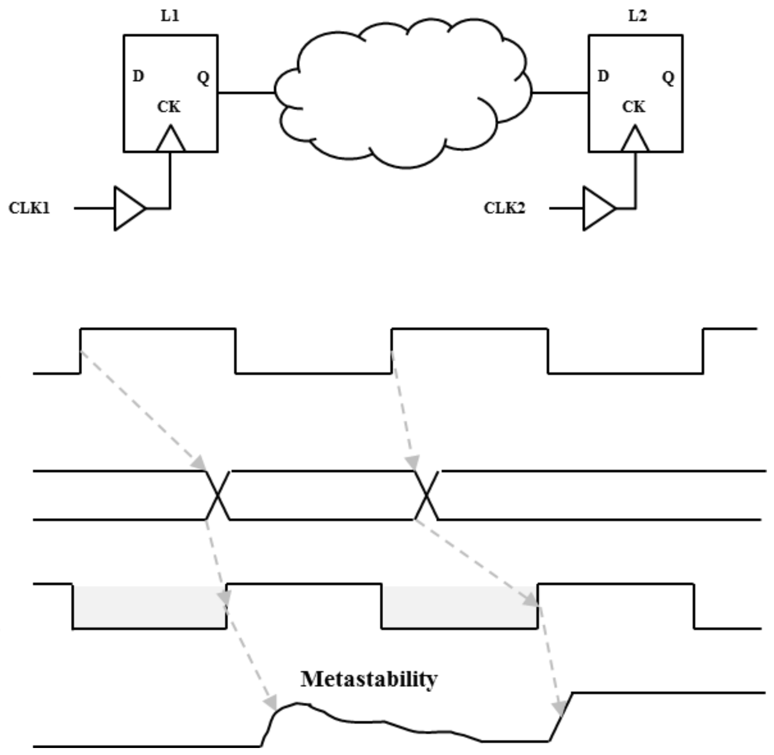


Figure 1 Setup failure and metastability.

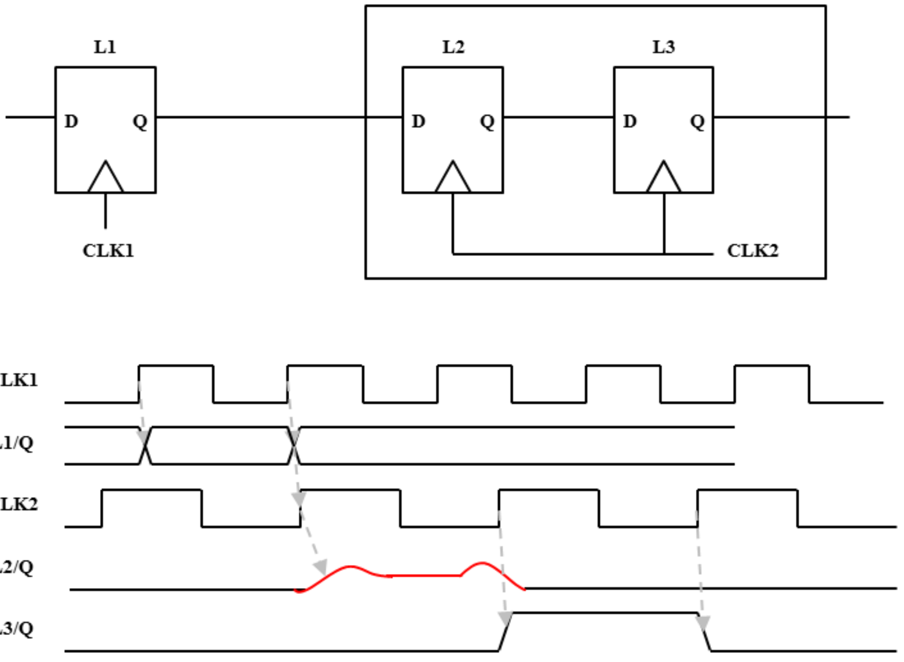


Figure 2 Two flip-flop synchronizer and its behavior.

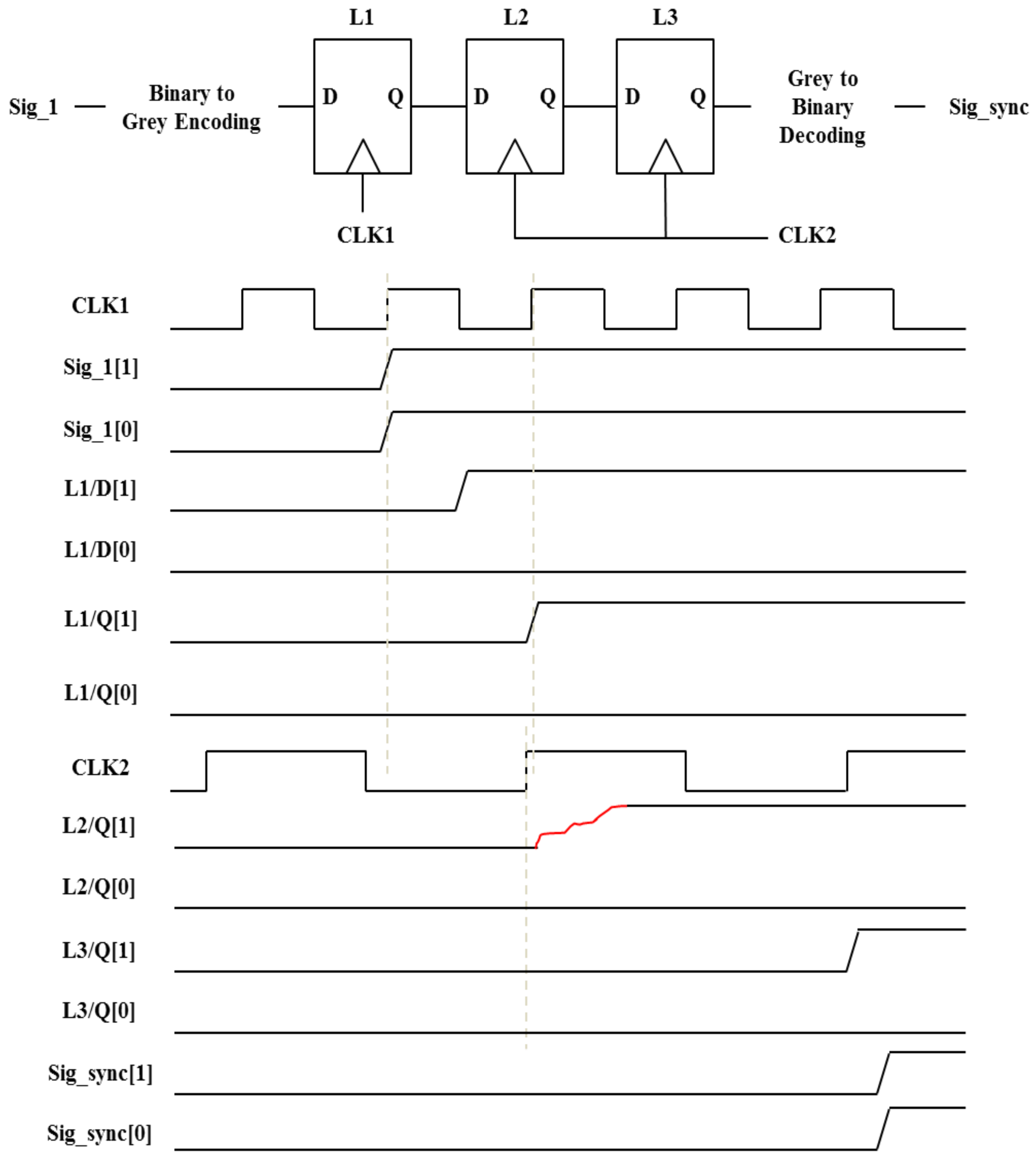


Figure 3 Gray encoding for multi bit signal and its timing diagram.

solvable by placing synchronizer on CDC point.

III. NanoTime-based Clock Domain Crossing Analyzer

A. NanoTime: Transistor-level Static Timing Analysis Tool

One of the most challenging issues in transistor-level circuit exploration is pattern recognition problem. Synopsys NanoTime is Tr-STA tool which can resolve the pattern recognition problem in transistor-level circuits [5]. To model the functional behavior and timing characteristic of the given circuit, the tool recognizes not only patterns those are included in its topology library including simple inverters, nand/nor gates and sequential elements, but also user-defined special topologies including multi-input latches and flip-flops within less than 2 hours for circuits which include millions of transistors. Even though the tool can recognize patterns and trace circuits, it is able to explore only a small amount of CDC points, by itself. The goal of this section is addressing the limitations of NanoTime as CDC analyzer and proposing a workaround solution, called NanoTime-CDC.

B. *Clock Domain Exploration using NanoTime*

NanoTime traces the whole circuit when the clock sources are given and false path exceptions among clocks in the same clock domain make NanoTime generate the CDC points. But coverage is extremely low. In this section, we address the region that cannot be explored by NanoTime and propose the workaround solution.

The clock network is composed of two different parts: (1) clock network propagated from a clock source (The path from CLK1 to The Latch0 in Figure 4), (2) clock network propagated from latch outputs (The path from The Latch0 to Latch1 in Figure 4). We call both clock networks as a clock domain of the clock source. NanoTime can find (1) type of clock network which is propagated from clock source. On the other hand, (2) type of clock network is recognized as data path by NanoTime tool which cannot generate the paths from/to the following latches. Since the number of latches which should have above commands is extremely large, finding the latches and setting commands should be done automatically. Flowchart of fully automated algorithm is shown in Figure 5. The flowchart includes basic settings of NanoTime tool which is represented voltage settings, reading technology and circuit descriptions, discarding analog circuits and clock definitions for clock domain. Match_topology command of NanoTime tool propagates all defined clock signals and recognizes all circuit topologies. After match_topology command is performed, procedure that finds newly clocked latches those are not identified in the previous NanoTime iteration. If nets of clock pins of a latch have clock flag, the latch is marked as “clocked latch” and the latch is new clocked latch when it’s not clocked in the previous NanoTime run. If the number of new clocked latches

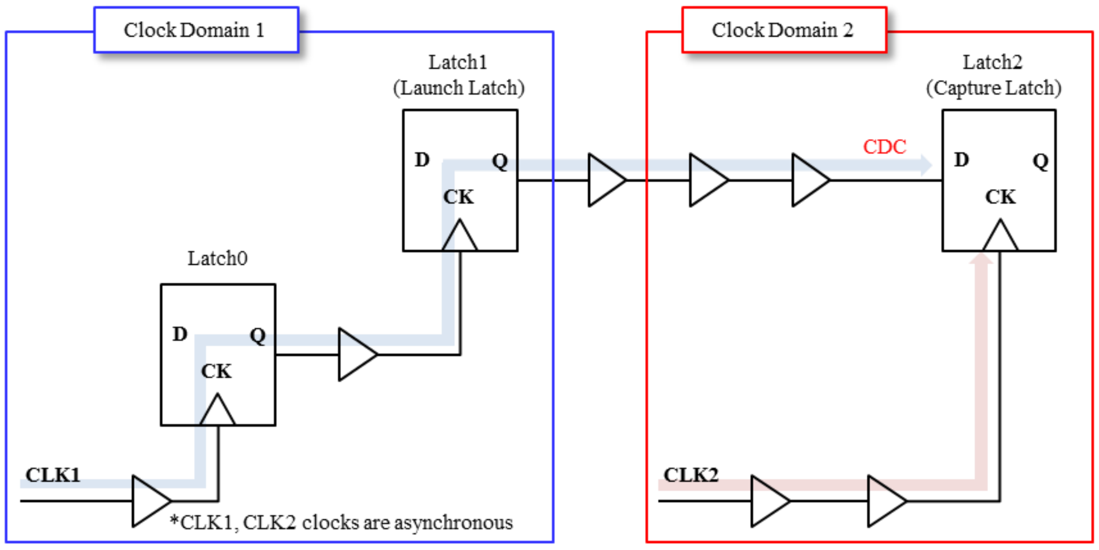
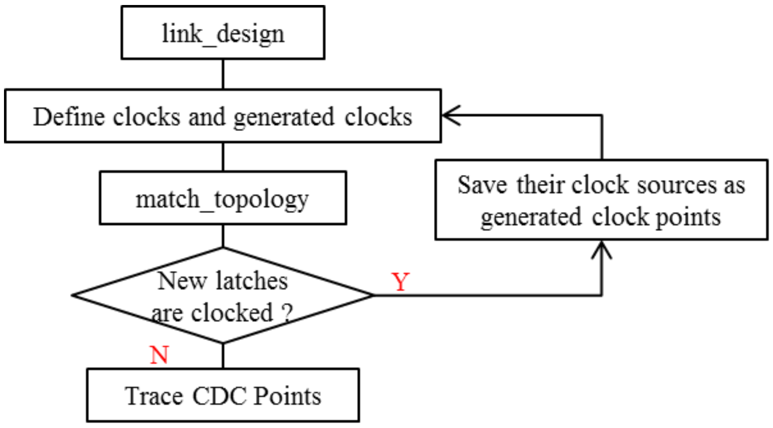


Figure 4 CDC point generated between two different clock domains..



link_design
Define clocks and generated clocks
match_topology
Save their clock sources as generated clock points
New latches are clocked ?
Y
N
Trace CDC Points

Figure 5 GClk Finder Algorithm.

is not zero, new clocked latches should be marked as clock network (mark_clock_network –force) to propagate (2) type of clock network. NanoTime iteratively finds new clocked latches and propagates clock signals. If there's no new clocked latch, algorithm is terminated and finding all clock domains can be achieved. The algorithm increases coverage of the NanoTime-CDC, significantly.

C. CDC Point Exploration using NanoTime

After exploring all clock domains, NanoTime can analyze CDC points. All propagated clock domains are crossed on CDC points. Tool is able to generate the timing paths between asynchronous clock domains except the paths on same clock domain.

D. Limitations

Finding all CDC points using NanoTime tool has several limitations. First, runtime of NanoTime-CDC is not acceptable. One of the most timing consuming parts of NanoTime-CDC is timing calculation behavior of NanoTime tool. Since NanoTime is designed for static timing analysis, the tool must calculate all path delay of target design. The tool simulates all timing blocks in the design and uses results when it performs timing analysis. However, timing calculation is not necessarily in CDC analysis. If there's another tool that doesn't perform timing calculation for CDC analysis, its runtime would be much faster than NanoTime-CDC. Second, due to large runtime of NanoTime itself, tool coverage is limited for trade-off between runtime and coverage. NanoTime recommends saving paths with maximum and minimum delay and it's the only reasonable option when target design has more than 10 million transistors. Figure 6 shows the path that is saved by NanoTime when it's default mode. Even though NanoTime support full path enumeration feature to obtain all timing paths in design, enabling the feature is unrealistic because of exploding runtime. Thus, the NanoTime-CDC is not acceptable because of its runtime and coverage limitations.

IV. Transistor-level Clock Domain Crossing Analysis

In the previous section, NanoTime-CDC is proposed and its limitations are described. Even NanoTime-CDC is the only tool that analyzes CDC points in transistor-level design, its performance is not acceptable. In this section, Transistor-level CDC (Tr-CDC) is introduced which is designed for CDC analysis in transistor-level design. Tr-CDC includes pattern matching and path tracing algorithms and excludes timing calculation behavior that degrades the performance of NanoTime-CDC. Coverage of Tr-CDC is also enhanced by virtue of reduced runtime. Following sections show simplified pattern matching and path tracing algorithms of Tr-CDC.

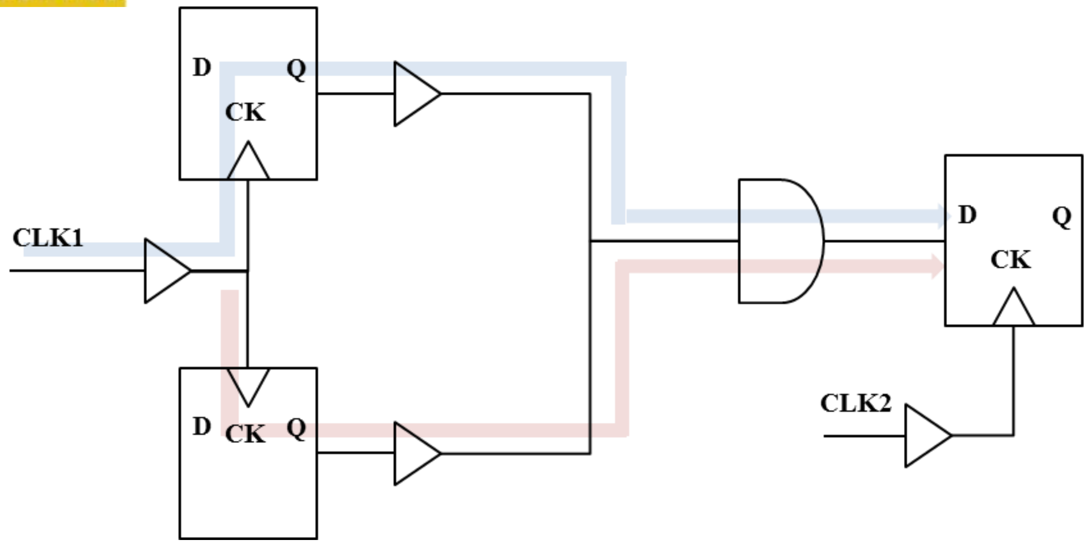


Figure 6 Two different CDC Paths in target circuit. NanoTime traces only paths with blue arrow.

A. Topology Recognition

Designing circuit exploration tool for transistor-level design is a challenging work due to complexity of pattern recognition algorithm. However, through the lessons of NanoTime-CDC, we have confirmed that recognizing channel connected blocks (CCBs) [5], transmission gates and latch structures is only needed and required pattern matching algorithm has lower complexity. Figure 7 shows two CCBs. The first one is inverter structures. Recognizing the structure guides timing paths from net A to net Z. Recognizing transmission gates generate three different timing paths. When there's no pre-defined transistor direction, signal can propagate from SEL1/SEL2 to A/Z. In addition, when signal comes to net A, Z is the proper output and vice versa. Likewise, recognizing CCBs and transmission gates directs timing paths in circuit. Latch structure should be recognized to find the termination point. Propagated clock signals should be stopped at the latch input net which is the CDC point candidate. The proposed methodology focuses on recognizing only few types of latches.

B. Clock Domain and CDC Point Exploration

Basically, exploring circuit from pre-defined clock sources is an application of the simple graph search algorithm. Breadth-First Search (BFS) algorithm can handle any big circuit with $O(|V|+|E|)$ time complexity. The algorithm traces circuits from clock sources through the clock pins of the latches and generates BFS tree until latch input, circuit output ports, and dangled ports are encountered. With the path tracing algorithm, GClk Finder in the Section II.B is not necessary. The algorithm can search all clock domains and also can find CDC points by checking the clock and input pins of latch inputs which are saved as the leaf nodes of BFS tree.

C. Analyzing Clock Domain Crossing

All CDC points should have metastability-free logic. Not all points should have their own synchronizer. Some points can be verified as CDC-free logic since the specification blocks the CDC operation. Thus, verifying the CDC points should include path analysis like commercial CDC verification tool that the proposed work supports.

V. Experimental Result

The proposed Tr-CDC is implemented in C++ and both algorithms are tested on Linux Machine. The target benchmark is DRAM circuits. Latest DRAM circuits have lots of transistors for their peripheral logics to control DRAM functions and multiple numbers of clock domains to decrease power consumption of DRAM circuits. The number of clock sources is less than 100, and their clock groups are 28.

Figure 8 summarizes the results of NanoTime-CDC and Tr-CDC. To enable fair comparison, pattern matching engine of Tr-CDC is replaced by that of NanoTime-CDC. The total numbers of CDC points of NanoTime-CDC and Tr-CDC are 43,702 and 435,394, respectively. Since the results have extremely large number of CDC points, some workaround analysis is performed. First, both results have all known CDC points which are intended by the circuit designer. Second, for all clock domains, Tr-CDC always finds more CDC points than NanoTime-CDC. Finally, for clock domain 26, it is verified that NanoTime-CDC needs additional setups to explore the missing points.

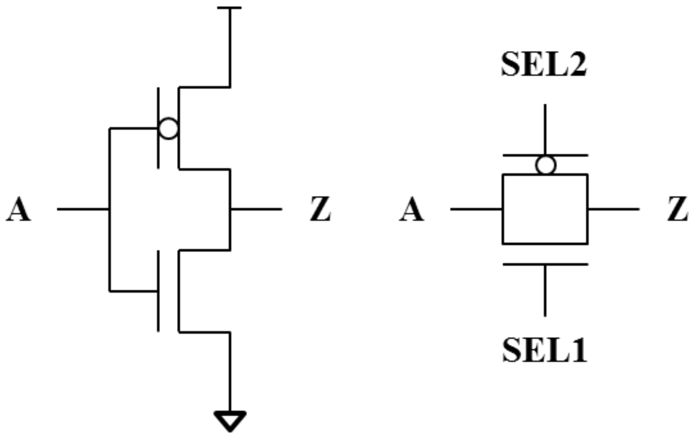


Figure 7 Channel Connected Blocks (CCBs). The left is inverter structure and the right is transmission gate.

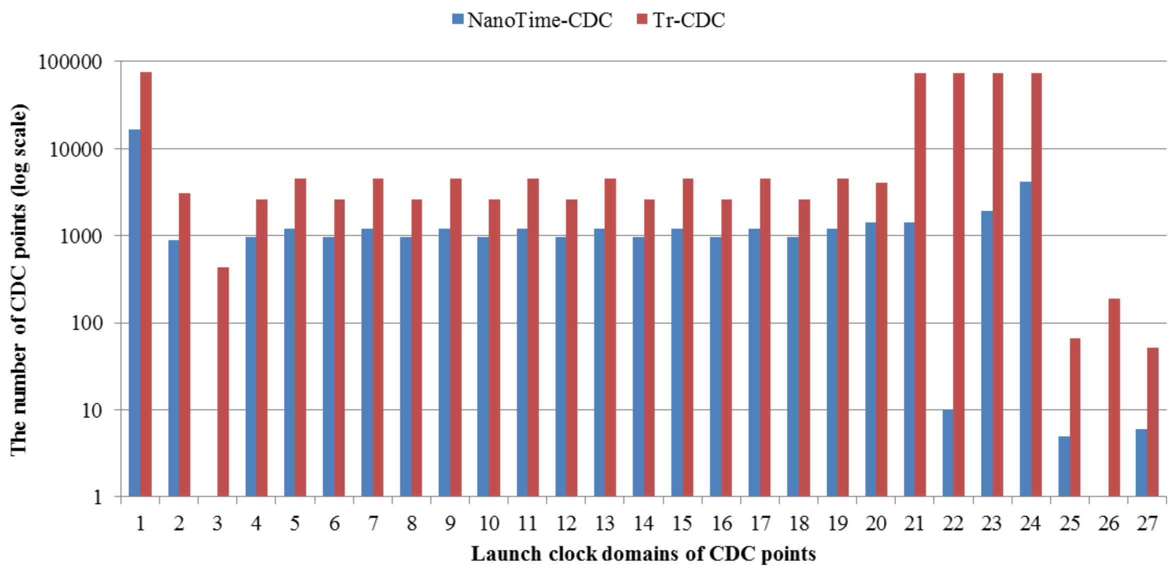


Figure 8 Comparison of the number of identified CDC points between NanoTime-CDC and Tr-CDC.

Therefore the result of Tr-CDC covers that of NanoTime-CDC which misses valid CDC points. Since the result takes all CDC point including points already have synchronizer, further verification flow is needed. With deeper verification, the number of CDC points can be reduced and compressed below 100 points. Figure 9 shows runtime of the proposed algorithm and it takes less than an hour while the NanoTime-CDC consumes more than a day because of the runtime of GClk Finder algorithm and its unnecessary timing calculation.

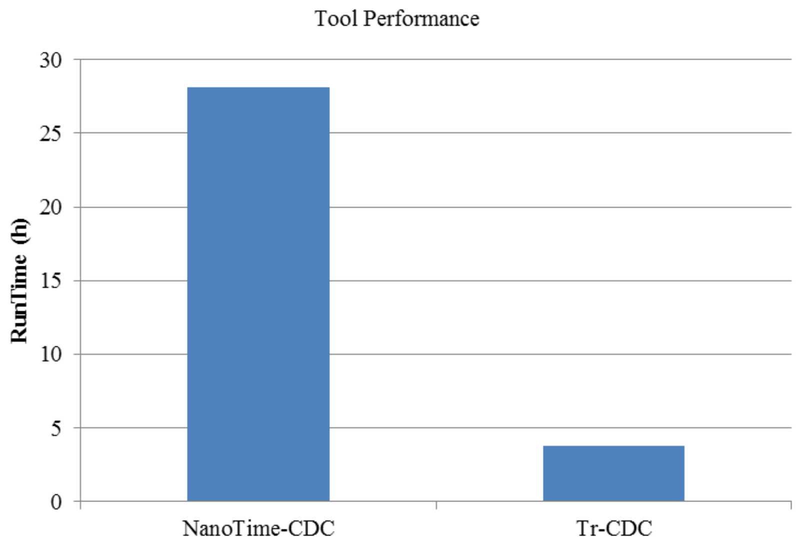


Figure 9 Comparison of tool performance between NanoTime-CDC and Tr-CDC.

VI. Conclusion and Future Works

This work addressed the problem of verifying CDC problem, which has never been automated in the transistor-level design. The proposed technique implements the Tr-CDC tool with a small effort with a simple pattern matching and graph traversal algorithms. Experimental results show the superiority of the addressed Tr-CDC compared to the workaround solution using commercial NanoTime tool. Many commercial CDC tools supporting RTL/Gate level design have graphical user interface to facilitate convenient CDC analysis which includes defining false path exceptions and it's one of the confirmed next steps for the addressed technique.

References

[1] Saurabh Verma, Ashima S. Dabare, Atrenta, "Understanding Clock Domain Crossing Issues," EETimes 2007, https://www.eetimes.com/document.asp?doc_id=1276114

[2] SpyGlass CDC Submethodology Version O-2018.09, Synopsys, Inc., 2018.

[3] Meridian CDC User Guide Version 5.0, Real Intent, Inc., 2018.

[4] T. D. Jain and D. Jain, "Synchronizer techniques for multi-clock domain SoC & FPGAs," EDN Network, Sep. 2014.

[5] NanoTime User Guide Version O-2018.06, Synopsys, Inc., 2018.