Using Modal Analysis to Increase Clock Domain Crossing (CDC) Analysis Efficiency and Accuracy

Kurt Takara, Mentor Graphics, A Siemens Business, Fremont, CA, USA
Tel: +1-510-354-7400 Kurt_Takara@mentor.com,
Gargi Sharma, Mentor Graphics, A Siemens Business, Fremont, CA, USA
Tel: +1-510-354-7400 Gargi_Sharma@mentor.com,
Ajay Thadhlani, Barefoot Networks, Santa Clara, CA, USA
Tel: +1 (650) 924-9363 athadhlani@barefootnetworks.com,
Bhrugurajsinh Chudasama, Barefoot Networks, Santa Clara, CA, USA
Tel: +1 (650) 924-9363 bchudasama@barefootnetworks.com

***Abstract*- In this paper we will show how an automated modal CDC analysis can be used to exhaustively verify CDC issues in all test and operational modes of a large SoC with multiple IPs. Modal CDC analysis configures the design as a set of operational modes and runs CDC analysis on each modal version of the design. Before the modal analysis flow was adopted, users would have to manually setup CDC analysis runs for each of the parameter sets corresponding to each DUT mode. The main benefit of this new approach has come from the automatic consolidation of all the results from each mode, making issues very easy to interpret and debug. Now it only takes us a few hours to review results and begin to take corrective action, vs. days and weeks with the prior, manual approach.**

I. Introduction

Today's SoC designs employ advanced multi-clocking architectures to meet the high-performance and low-power requirements. Since metastability from the intermixing of multiple clock signals is not modeled by digital simulation, we must perform exhaustive, automated Clock Domain Crossing (CDC) analyses to identify and correct problem areas to avoid unpredictable behavior when the chip samples come back from the fab. Furthermore, given the breadth of end customers' requirements, SoC's must support numerous "modes" for system start-up and configuration, BIST, end-customer use cases and interface combinations. Being able to satisfy the need to perform extensive CDC analysis, and the high operational flexibility of the end product, poses a significant design and verification challenge.

In addition, while many SoC operational modes share a baseline clocking and data path configuration, commonly there are a significant number of modes that are very different from the baseline and/or from each other. This means signals and registers that are "CDC safe" in a given mode can be a violation when the SoC is running in other legal modes. From EDA methodology perspective, there is another layer of complexity to address: because the SoC's are so large (nearly 1 billion gates), we must perform the CDC analysis and aggregate the results in a hierarchical manner.

II. Prior Solutions

Before the modal analysis flow described above was adopted, users would have to manually setup CDC analysis runs for each of the parameter sets corresponding to each DUT mode. This was a tedious and error prone process to set-up. Additionally, because each run was independent of the other, there was a high degree of overlap in the analysis between runs, meaning a lot of compute cycles were essentially wasted. Finally, trying to manually merge together the massive amount of results from such repetitive, individual runs was also a tedious, error prone process.

III. THE NEW APPROACH

The essence of the new, automated approach is to programmatically configure the design into each legal operational mode, then run CDC analysis on each modal version of the design. This accomplishes several things:

- *Captures the modal nature of the design*

An artifact of modal CDC analysis is that a design's various legal operational modes become well-defined and explicitly specified. Understanding these operational modes helps designers grasp the true nature of the design's functionality, which helps avoid misinterpretation of the specification.

- *Reduces complexity and "pessimism"*

Using a divide-and-conquer approach, CDC analysis is split into smaller pieces. Rather than have a huge number of violations and cautions for the general design, CDC analysis returns results in groups relevant to the legal operational modes of the design. Running regular CDC analysis on such a design returns pessimistic results because the design has more clock domains – modal CDC analysis eliminates these pessimistic results.

- *Improves multi-mode analysis accuracy*

Modal CDC analysis enables designers to analyze the correlation between design modes. Designers can easily see the commonalities and differences between the design modes and quickly identify design issues and CDC bugs. Without the aggregation of results, designers would review the N mode results separately which would increase the review time by up to N times.

IV. METHODOLOGY

The work flow for modal CDC analysis is largely the same as regular CDC verification, except with some additional steps:

• Phase 1: Design setup

The design configuration setup is the same for both regular and modal analysis flows. In addition, the CDC analysis will automatically detect the clock configuration registers and determine the permutations and combinations of design configurations. The design team determines the legal and illegal design modes and configurations. Then a follow-up mode and configuration review allows engineers to detect modes that were missed or are "accidental" and need to be corrected.

• Phase 2: CDC analysis

Once the clock domains and operational modes are properly specified, an automatically-generated run script configures and executes a full CDC analysis for each mode. The CDC analysis detects all CDC paths and verifies the synchronization structures on every CDC path. The mode-specific CDC results are aggregated for review by the designer.

• Phase 3: CDC Debug

The GUI has several modal indicators that are not relevant to non-modal designs, and thus are not visible when viewing regular CDC analysis results – i.e. the GUI presents a clear separation of results based on the various design operational modes that were analyzed. However, the results of all the modal CDC runs are conveniently aggregated, so designers can compare and contrast the good and bad synchronization structures for CDC paths across different operational modes.

V. MODAL ANALYSIS EXPLAINED

During design initialization, the clock configuration registers are loaded and this selects the clock sources and clock speeds. Here is an example showing the clock configuration register *ctrl* selecting the appropriate clock paths by controlling the select pin of the clock mux (Figure 1).

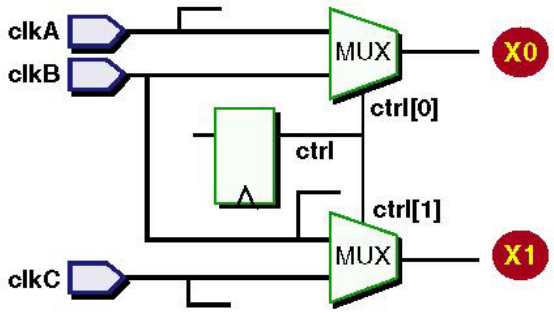

Figure 1. Clock configuration registers select the clock sources.

Assume that *clkA*, *clkB* and *clkC* are asynchronous and *X0/X1* are clock signals. Unless *X0* or *X1* are specified to be synchronous with one of the input clocks, the regular CDC analysis infers that this circuit has five clock groups: *clkA*, *clkB, clkC, X0, X1*. CDC clock analysis will pessimistically assume that the mux *ctrl* signal will dynamically switch during normal operation, so the mux outputs are considered new asynchronous clock groups and cannot be considered synchronous to either of its input clocks. However, when the clock configuration does not change dynamically during normal operation, assuming new asynchronous clocks at the output of the muxes are an overly pessimistic assumption. For static clock mode scenarios, the mux outputs will be associated with only one of the input clocks at any given configuration. Re-running with modal analysis yields the correct number of 4 modes:

cdc_mode_0: X0 = clkA and X1 = clkB
ctrl = 2'b00
cdc_mode_1: X0 = clkA and X1 = clkC
ctrl = 2'b01
cdc_mode_2: X0 = clkB and X1 = clkB
ctrl = 2'b10
cdc_mode_3: X0 = clkB and X1 = clkC
ctrl = 2'b11

Each of these modal configurations is less complex than the non-modal configuration. In particular, each mode has only 3 input clock domains rather than 5 asynchronous clock domains that is specified for the non-modal case.

Of course this is a very basic example. While we generally trust the ability of the analysis to infer the right modes on large scale IPs, designers often manually specify the mode configurations as well. Any mode configuration discrepancies are flagged as errors, providing a form of "specification verification" for free.

VI. RESULTS

We ran the modal analysis on a small example design to characterize the value of the modal CDC analysis. This design used an extensive number of clock muxes driven by clock configuration logic. The normal CDC analysis, detected 338 asynchronous clock domains. Although there were 22 primary clocks in the design, the analysis also inferred 316 new asynchronous clocks that were driven by clock muxes. Since the clock configuration happened statically, all clock muxes would resolve to one of the two input clock domains. Resolving the 316 muxed clocks would significantly reduce the number of asynchronous clock domains and avoid reporting of pessimistic CDC paths to/from the clock muxes.

A. Phase1: Design Setup

The modal CDC analysis resolved all 318 muxed clocks, so the analysis was run on only the 22 primary asynchronous clocks (See Table 1). We recognized that the modal analysis reduction in total asynchronous clocks reduces the noise and redundancy in modal CDC results vs. the normal CDC analysis.

Table 1: Asynchronous Clock Comparison

	Primary Clocks	Muxed Clocks	Total Clocks
Normal CDC Analysis	22	316	338
Modal CDC Analysis	22		22

During the design setup, the CDC analysis automatically detected the clock muxes and the clock configuration registers. The analysis also determined the available mode configurations by calculating the unique permutations and combinations of clock configuration register values. In our design, the modal analysis detected 19 unique configurations. For each configuration, the analysis specified the appropriate clock configuration register values (See Figure 2). Designers reviewed the design configurations and the clock configuration values to confirm that all scenarios were legal for our design use.

```
Inferred Modes
==============

Constants in cdc_mode_0
--------------------------------------------------------------------------------
Signal                                  Value
--------------------------------------------------------------------------------
inst_LogicCore.inst_RegistersTop.inst_Registers1_1.SWCLOCK_EN 1'b1
inst_LogicCore.inst_RegistersTop.inst_Registers1_0.BIST_CLK 1'b1
inst_MemorySystem.inst_MemorySystemIo.inst_CpuBusIo.SelCpuClk_i 1'b0
inst_LogicCore.inst_RegistersTop.inst_Registers1_1.HW_GEN_DATA_MODE_PODS 122'b10
--------------------------------------------------------------------------------

Constants in cdc_mode_1
--------------------------------------------------------------------------------
Signal                                  Value
--------------------------------------------------------------------------------
inst_LogicCore.inst_RegistersTop.inst_Registers1_1.SWCLOCK_EN 1'b0
inst_LogicCore.inst_RegistersTop.inst_Registers1_0.BIST_CLK 1'b1
inst_MemorySystem.inst_MemorySystemIo.inst_CpuBusIo.SelCpuClk_i 1'b0
inst_LogicCore.inst_RegistersTop.inst_Registers1_1.HW_GEN_DATA_MODE_PODS 122'b10
--------------------------------------------------------------------------------
```

Figure 2: Inferred modes subset

B. Phase 2: Running CDC analysis

The CDC modal analysis automatically generated a run script that would run the unique design configurations through CDC analysis. As we are able to parallelize the modal runs, the total CDC analysis run time of the modal analysis was similar to the normal CDC analysis run and sometimes faster (See Table 2). The normal CDC run took longer than any single modal run, because there were a greater number of CDC paths in the normal CDC run (See Table 3). A distinct set of CDC results is generated for each configuration.

Table 2: CDC Run Time Comparison between Normal and Modal Runs

	Setup	Mode-based runs X 20	Total
Normal CDC Analysis			471s
Serial Modal Analysis	110s	109s-122s	2322s
Parallel Modal Analysis	110s	109s-122s	132s

C. Phase 3: CDC debug

Although the CDC modal analysis generated a distinct set of results for each mode, the debug GUI automatically aggregated the results. The aggregated results were a more efficient way for designers to compare and contrast the CDC path results between the modal runs.

The comparison between the normal and modal runs emphasized the difference in CDC path accuracy and pessimism. The pessimism of the normal runs generated 10,973 CDC paths which includes paths to/from clock mux domains, but the more realistic modal analysis generated between 402 and 928 CDC paths in any one design configuration (See Table 3). The total number of CDC paths analyzed in the modal analysis is 12,526 paths.

Table 3: CDC Path Comparison between Normal and Modal Results

	Incorrect CDC Paths	Correct CDC Paths	Total CDC Paths
Normal CDC Analysis	10973	105	11078
Modal Analysis Accumulated	12427	99	12526
Modal Analysis per Configuration	401-921	1-7	402-928

The modal analysis results emphasized the accuracy of the results vs. the normal results (See Table 4). For any one configuration, a designer would look at a lower number of CDC paths and violations in the modal results vs. the normal CDC run. For example, a designer can avoid reviewing 12,526 individual paths by aggregating the common paths on a modal basis.

Table 4: Modal CDC Path Results

	Incorrect CDC Paths	Correct CDC Paths	Total CDC Paths
cdc_mode_0	847	7	854
cdc_mode_1	849	7	856
cdc_mode_2	843	7	850
cdc_mode_3	422	7	429
cdc_mode_4	401	1	402
cdc_mode_5	423	8	431
cdc_mode_6	630	1	631
cdc_mode_7	847	7	854
cdc_mode_8	849	7	856
cdc_mode_9	421	7	428
cdc_mode_10	473	1	474
cdc_mode_11	494	7	501
cdc_mode_12	919	7	926
cdc_mode_13	493	7	500
cdc_mode_14	699	1	700
cdc_mode_15	702	1	703
cdc_mode_16	495	8	503
cdc_mode_17	921	7	928
cdc_mode_18	699	1	700
TOTAL	**12427**	**99**	**12526**

A common occurrence was a CDC path that was a violation in multiple configurations (See Figure 3). We were able to determine that this violation is shared across multiple modes and avoided reviewing this violation 19 separate times when the violation was not aggregated. The aggregation of the CDC modal results, allows designers to review each CDC path in the context of multiple configurations. In this case, there was an incorrect synchronization structure resulting in a combinational logic violation, but the designer determined that the violation could be waived. The designer was able to add 1 waiver to resolve 19 violations. We did not find examples of any CDC paths that were a violation in one configuration, but a good synchronization in another configuration.

Figure 3: Incorrect CDC path shared by multiple modes.

Since it is very easy to compare and contrast each CDC violation between mode configurations, constraints and waivers can be more efficiently identified and then added to the CDC setup. Finally, we recommend that designers are careful about waiving or constraining violations that may be shared across multiple modes. It is safest to review each CDC path or violation in every mode to be sure that there is no possibility of CDC errors.

VII. Conclusion

The main benefits of this new approach have come from a more complete analysis of all possible design configurations, the overall reduction in CDC noise, and the automatic consolidation of all the results from each mode; making issues very easy to identify and debug. Now, it typically takes only a few hours to review results and begin to take corrective action, vs. days and weeks with the prior, manual approach. However, because the automatically aggregated results are so much more efficient to use, the engineers feel that they are getting much higher throughput for the designers' time investment than before.

Acknowledgments

We would like to recognize Joseph Hupcey III for his invaluable help with editing this paper and the corresponding slide set.

Memoriam

We dedicate this paper to our co-author Cheng-I Huang. We will miss his creativity, diligence, and spirit – he will remain in our hearts and memories.

References

[1] C.E.Cummings, "Clock Domain Crossing (CDC) Design and Verification Techniques using System Verilog," SNUG 2008.

[2] A. Chen, et. al., "Power Aware Clock Domain Crossing Verification", System-On-A-Chip Reuse Platforms Can Dramatically Shorten Design Cycles. (2000, September 16). Electronic Design. Retrieved September 10, 2015, from http://electronicdesign.com/dsps/system-chip-reuse-platforms-can-dramatically-shorten-design-cyclesDAC 2014.