Verifying clock-domain crossing at RTL IP level using coverage-driven methodology

Jean-François Vizier
STEricsson
12, rue Jules Horowitz
38000 Grenoble, FRANCE
+33-47658-6068
jean-francois.vizier@stericsson.com

Dennis Ramaekers
STEricsson
12, rue Jules Horowitz
38000 Grenoble, FRANCE
+33-47658-7123
dennis.ramaekers@stericsson.com

Zheng Hai Zhou
STEricsson
88, Zihai Rd,
200241 Shanghai, CHINA
+86-21-2418-8970
zhenghai.zhou@stericsson.com

ABSTRACT
Usage of a GALS approach for a SoC implies the creation of several asynchronous paths. These paths can be critical for the system as some of them are part of the system bus. They require special attention during verification.

RTL simulations are not able to model either CDC issues or their effects. Gate level simulations are delay aware and could fill the gap. But they can only be performed late in the projects life, which makes this solution inappropriate.

This paper presents the verification methodology developed to address this topic early, during IP RTL simulations. This methodology responds to the following constraints: no design instrumentation, coverage-driven verification compliancy and easy reuse. The final section presents some interesting results achieved by applying the methodology to real multi-clock IPs.

Categories and Subject Descriptors
B.6.3 **[Logic Design]**: Design Aids, *Verification*

General Terms
Verification.

Keywords
GALS, metastability, CDC, eVC, methodology, coverage-driven verification.

1. INTRODUCTION
Digital baseband SoCs developed for 3G multimedia phones integrate more and more features. As the target clock frequency is drastically increasing, it's getting unviable to keep a fully balanced clock tree for all the SoC. Consequently a GALS approach is used. The outcome is that many CDC paths are present at the boundary of each synchronous domain. These CDC paths are used intensively as they are supporting huge data transfers inside the SoC. Hence, special care must be paid in the verification of these paths.

A flip-flop which is the end point of a clock domain crossing (CDC) path is subject to timing violation due to asynchronism of the clocks. The flip-flop can enter in a metastable state which is dangerous for the rest of the chip. We present in Section 2 this phenomenon and the way to minimize the probability of its propagation.

Structures used to reduce propagation of metastability can't avoid the effects of this phenomenon. Data can be corrupted and the design must be resistant. Simulations can't properly model these effects. We will present the limitations of the simulators in Section 3 and the model we propose to fill the gap in Section 4.

A methodology has been defined to benefit from this model. This methodology will be detailed in Section 5. In Section 6, we will present the results we could obtain with it.

2. ASYNCHRONOUS DESIGN ISSUES
In order to observe a deterministic behavior at the output of a flip-flop, its data input must remain stable during the setup and hold time.

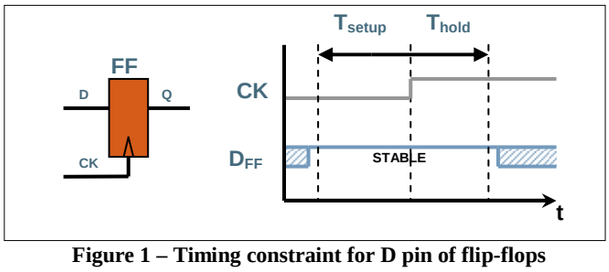

Figure 1 – Timing constraint for D pin of flip-flops

When these timings are not respected (i.e. a transition on D occurs too close to CK edge), the behavior of the flip-flop output is unpredictable: the output can stabilize either to 0 or 1. In some cases, the output can also become metastable, which means oscillating at a level between 0 and 1. This oscillation should diverge and finally settle to 0 or 1 randomly after a resolution time.

In asynchronous designs, there is no way to guarantee setup and hold time for first flip-flops in the receiving clock domain. So metastability and data corruptions will occur for that reason. The design must be resistant to those issues.

Some structures exist to reduce metastability propagation probability, in particular the n flip-flop synchronizer, presented in figure 2.

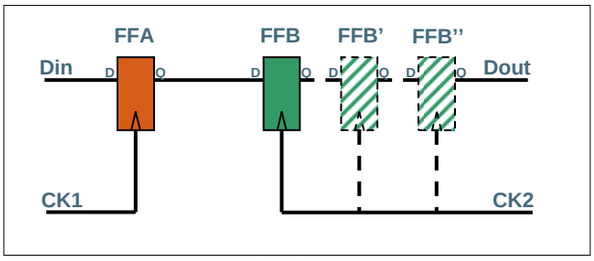

Figure 2 – n flip-flop synchronizer

Design constraints determine the number n of FFB flip-flops to instantiate.

This structure reduces the probability for metastability to be propagated on Dout but doesn't ensure data correctness. This becomes an issue when several resynchronized signals reconverge in the receiving clock domain.

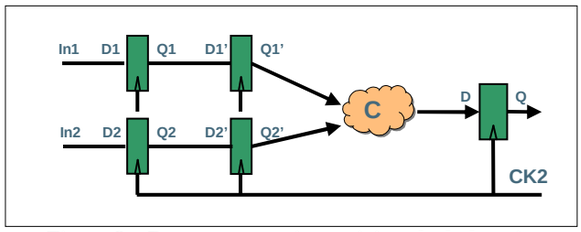

Figure 3 – Reconvergence in receiving clock domain

If In1 and In2 both violate setup/hold window on D1 and D2, data may become incoherent between Q1 and Q2 during the following receiving clock cycle, as shown in figure 4.

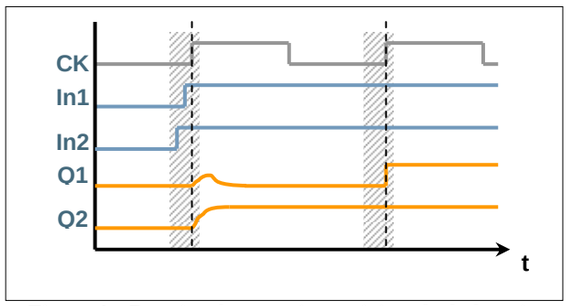

Figure 4 – Data incoherency in receiving clock domain

If those resynchronized signals reconverge in the receiving clock domain, special care must be taken in the design phase to deal with this incoherent data. This has to be verified.

3. SIMULATION MODELS LIMITATIONS

3.1 RTL Simulations

When performing RTL simulations, flip-flops are modeled as sequential elements with no setup/hold time information. On the clock rising edge, the input is copied on the output with no consideration of timing violation. So, there is a class of bugs related to metastability which cannot be detected with a traditional RTL model.

3.2 Gate level Simulations

Gate level simulation provides a solution to model timing violation issues. It uses a back-annotated netlist, provided by the physical implementation team.

This netlist is available when back-end tasks are already very advanced, which means late in the project life. What is more, this netlist is not available independently for each IP.

Another issue with gate level simulations is that the flip-flop model is the behavioral model which is described in the libraries. This model can detect a setup/hold violation but doesn't model metastability effect and data corruption in a usable way. The model introduces an X on the output of the flip-flop for one clock cycle when a setup/hold violation is detected. So, to avoid X propagation, the timing checks are disabled.

4. PROPOSED MODEL

To address the lack of accurate simulation models, a new model of metastability effects and data corruption is needed. This model has to represent a "realistic simplification" of what occurs on silicon. Some requirements have to be respected.

4.1 Model requirements

The model must correspond to the following requirements:

- Usable in RTL simulations
- Doesn't need design instrumentation
- Doesn't have significant impact on run-time
- Easily reusable from one IP to another
- Provide coverage on exercised data corruptions.

4.2 Injector cell

To cope with those requirements we have defined an injector cell, described in Specman's e language. By using Specman, we'll benefit from the advanced constraints solver and coverage features. We'll also use the tool to monitor and force signals during the simulation, so there is no need to modify the DUT.

An injector cell is connected around each first flip-flop of the receiving clock domain and will add, during RTL simulations, a model of timing violation for the input of the flip-flop.

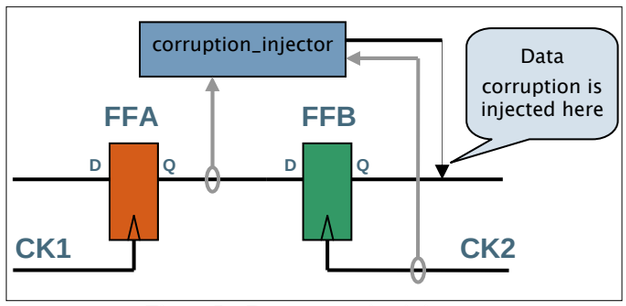

Figure 5 – Data corruption injector

The injector cell monitors two pins of the flip-flop: data and clock. When a transition occurs on the input, close to the clock edge, the injector generates an event to indicate a timing violation. Length of setup and hold windows are dynamically configurable.

When a timing violation event is generated, a random generator will decide to corrupt or not the data on the output of the flip-flop. This random generator can be constrained on several parameters (simulation time, current Q value, kind of violation).

Each injector can report its operations in the simulator log. It can report detected timing violations and injected data corruptions. Verbosity of injectors is configurable so to choose the appropriate level of information depending on the maturity of the verification environment.

Injectors are instantiated in an existing verification environment, using macros. It's necessary to provide the path to the pins of the flip-flops on which we want to model effects of timing violations.

Once instantiated, injectors will then behave in a random way.

Having injectors instantiated in the verification environment allows to model effect of timing violations on CDC paths. As we want to use these injectors in coverage driven verification environments, we need to collect coverage data.

4.3 Coverage Model

4.3.1 Signal level coverage

A first level of coverage is provided by each injector cell. It is a signal level coverage. It shows which kind of violations occurred for the related flip-flop and reports the data corruptions which were inserted. It also crosses data.

Signal level coverage gives a good overview of global corruption injection. If all corruption injectors are enabled, there will be a list of coverage groups, each one related to an instrumented flip-flop.

This can highlight some holes, related to flip-flops on which timing violations never occur. It can mean that testcases or relative clock phases are not able to activate the corruption injectors.

Once signal level coverage reaches the targeted value, it is important to focus on a higher level coverage, especially the reconvergence coverage.

4.3.2 Reconvergence coverage

As highlighted in section 2, when several signals are resynchronized into the receiving clock domain, there can be a loss of data coherency between these signals. This can give an issue if those signals are reconverging in the receiving clock domain.

The design has to be robust enough to handle these data incoherencies and its own behavior has to be verified. The purpose of the reconvergence coverage is to ensure that all possible data incoherencies combinations have been generated by the injectors.

A bus level monitor allows to collect events and data. It collects events and data coming from several corruption injectors. Here is its instantiation:

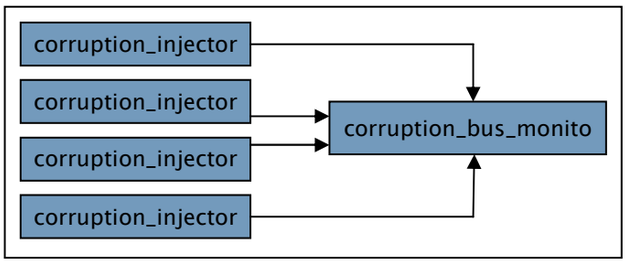

Figure 6 – Bus monitor cell instantiation

Each time a corruption is performed by an injector, an event is emitted. This event is received by the corruption bus monitor. The bus monitor then collects values from the injectors and defines which was the simulated value and which value has been forced.

These values are used to perform a cross coverage between simulated and forced values.

This monitor provides several crossed items, ensuring all possible data incoherencies are simulated. It is then the task of the scoreboard to perform data checking.

5. VERIFICATION FLOW

Usage of injector cells allow to model CDC issues in RTL simulations. These injectors have to be added in an existing verification environment. And it is mandatory to instantiate them on all first flip-flops of the receiving clock domains.
To ensure efficient usage of these injectors, a methodology has been defined.

As injectors must be plugged in an existing verification environment, the methodology can be composed of two main phases:

- Phase 1: Building of a standard coverage driven verification environment for a multi-clock IP.
- Phase 2: Usage of the injectors in the verification environment built in phase 1.

Here is a detailed description of those phases.

5.1 Phase 1

Typical functional verification applies to multi-clock IPs. This first phase consists in the building of the verification environment. It must respect the coverage driven verification methodology and pay special care to the clock generation.

Clocks must be generated according to clock scenarios, defining clock frequencies at which the IP will have to work. These scenarios must appear in verification plan of phase 1.

The coverage of phase 1 needs the full functional coverage of the IP, which has to be crossed with the clock scenarios coverage. Here is a summary of phase 1 steps.

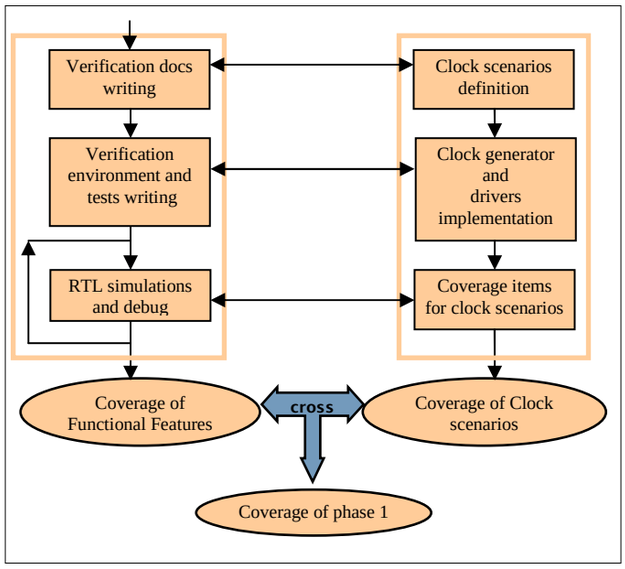

Figure 7 – Phase 1 steps

This figure shows typical verification steps on the left. Steps are added on the right to deal with multi-clock aspects.

This first phase is qualified with a coverage which is a cross of the IP functional coverage and the clock scenarios coverage.

Once the coverage has reached the targeted value, it is time to switch to phase 2, to address CDC issues.

5.2 Phase 2

In this phase, we can investigate specific CDC issues, thanks to the injectors. There must be one injector instantiated for each flip-flop subject to metastability, i.e. each first flip-flop of a receiving clock domain.

Phase 2 is divided in 3 steps: recognition, instrumentation and corruption. Here is a schematic view of phase 2:

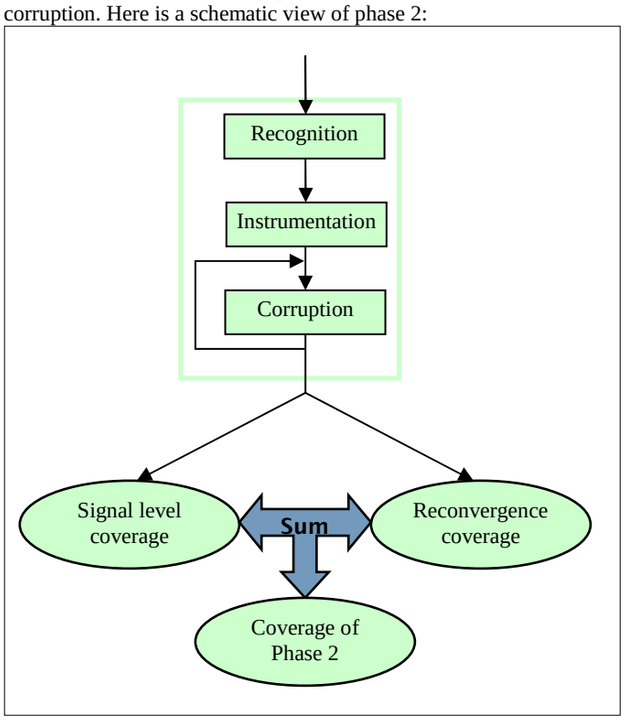

Figure 8 – Phase 2 steps

5.2.1 Recognition

Injectors have to be connected on each flip-flop subject to metastability. Recognition step consists in identifying all synchronization elements of the DUT.

This recognition must be as automated as possible. For that, we rely on structural checking tool, in our case Spyglass from Atrenta. Spyglass contains some CDC check rules which can provide the list of CDC path endpoints.

- clock_sync02 which reports the recognized synchronization structures.
- clock_sync01 which reports unrecognized synchronization structures. As we want to model effect of timing violations on all flip-flops which are subject to it, we need to consider this list.

From these reports, it's possible to build the list of pins on which the injectors must be connected.

The methodology directly relies on the correctness of this list. So, special care must be given to the setup of Spyglass and to the filtering of its reports.

Reconvergence coverage requires the lists of resynchronized signals which reconverge in the receiving clock domain. For this task, we also use Spyglass. Several CDC check rules are necessary to build the list of reconverging signals:

- Clock_sync03b provides reconvergence of signals coming from different clock domains.
- Ac_conv01 and Ac_conv02 provide reconvergence of signals coming from the same clock domain.

5.2.2 Instrumentation

This step consists in the instantiation of the injectors and of the bus monitors.

Injectors are instantiated by loading an e-configuration file. This file contains the configuration of each injector. This configuration is done thanks to macros and is directly given by the list of synchronization elements provided by the recognition step.

Bus monitors will collect information from several injectors, in order to build reconvergence coverage. The instantiation of these monitors is also done in an e-configuration file. It consists in defining the list of injectors a bus monitor is connected to. This list is provided by the recognition step.

The writing of the e-configuration files is done thanks to a script, using the outputs of the recognition step.

5.2.3 Corruption

This step consists in running the regression built in phase 1, with the injectors activated.

Running the regression with the injectors might trigger some new bugs. To help debug, it is then possible to constrain the injectors so to deactivate some of them or activate them in restricted conditions.

Phase 2 ends when the targeted coverage is reached. Some loops might be required with design teams when targeted reconvergence coverage is not reachable. It can occur when some reconverging signals have mutually exclusive states or transitions. Those transitions must be removed from the reconvergence coverage.

5.3 Verification qualification

The quality of the verification is measured thanks to the full coverage. This coverage is given by the sum of coverage of phase 1 and the coverage of phase 2.

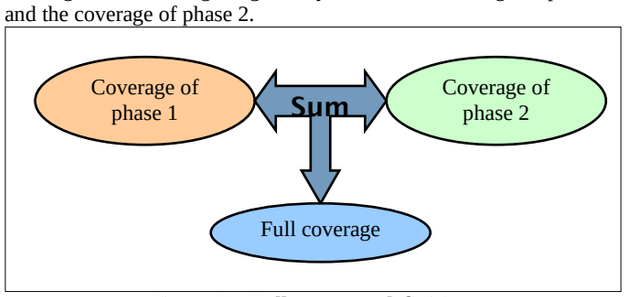

Figure 9 – Full coverage definition

6. RESULTS

The methodology is being deployed on several asynchronous IPs. First application was done on asynchronous protocol bridges, as these IPs are inserted on critical data paths of the SOC. It is now used for clock generator and power management IPs.

The methodology allowed finding several bugs due to CDC issues. In particular, it was applied to an IP, in which one bug was found during gate level simulations. Thanks to the injectors the bug could be reproduced during the first RTL simulation.

Some coverage is provided automatically by the package and can be used for the qualification of the verification environment. As transitions on the inputs can be mutually exclusive, in some case manual refining is needed for the reconvergence coverage goals.

The simulation run-time overhead has been computed on an asynchronous AXI-AXI bridge verification environment. The comparison is done, running the same test with the same seed on the environment of phase 1 and the environment of phase 2. The overhead is 37%.

The flexibility of the solution allows deploying it on larger IPs, such as system buses or in the future on NoCs. It is also possible to benefit of the corruption injectors during gate level simulations. This way, the behavioral model of the flip-flops could be overridden to model metastability effects.

7. CONCLUSIONS

GALS approach for new 3G baseband SoCs adds a lot of asynchronous paths. Some of those paths are extensively used as they are part of the system buses.

Special care must be given to the verification of those asynchronous paths. But sequential elements receiving data from an asynchronous clock domain are subject to timing violations on their input. As RTL simulations are not able to model the effect of those timing violation, some bugs due to CDC issues are not detectable.

The methodology presented here provides a solution to model CDC issues in RTL simulations. This way, it allows discovering bugs related to metastability effects, early in the project life.

The solution is compatible with a coverage driven verification methodology and can be added to an existing verification environment. It doesn’t require DUT instrumentation nor recompilation.

8. ACKNOWLEDGMENTS

Many persons contributed to develop the methodology and to deploy/debug it.
I want to thank STEricsson design and verification team members for their support, useful advices and technical knowledge sharing.
I also want to thank Patrick Oury, from Cadence, for is efficient support and his proposals for the development of the injectors.

9. REFERENCES

[1] Mark Litterick, DVCon 2006 Pragmatic Simulation-Based Verification of Clock Domain Crossing Signals and Jitter using systemVerilog Assertions.
[2] Roger Sabbagh, Hiroaki Iwashita, DVCon 2008, Does Your Simulation Model Metastability Effects Completely and Accurately?
[3] Clifford E. Cummings, Sunburst Design, Inc., SNUG-2001, Synthesis and Scripting Techniques for Designing Multi-Asynchronous Clock Designs
[4] Ran Ginosar, VLSI Systems Research Center, Technion—Israel Institute of Technology, Proceedings of the Ninth International Symposium on Asynchronous Circuits and Systems (ASYNC’03), Fourteen Ways to Fool Your Synchronizer
[5] Spyglass 4.2 Application Notes, Metastability Analysis in Dynamic Verification Flow.