FPGA-based Clock Domain Crossing Validation for Safety-Critical Designs

Alexander Gnusin
Aldec Inc.
2260 Corporate Circle Henderson, NV, 89074
alexg@aldec.com
613-270-9454

Abstract-

The verification of complex FPGA designs is becoming increasingly challenging for safety-critical and mission-critical applications. The design complexity grows, and the number of tests to verify design functionality grows even further. Functional simulation cannot provide designers with the verification completeness criteria, as it is able to verify design using only limited tests vectors subset. Therefore, such industry standards as DO-254 (design Assurance Guidance for Airborne Electronic Hardware) require that safety- and mission-critical designs be verified on the real hardware for certification purposes.

Clock Domain Crossing verification in one of the most challenging topics of complex FPGA verification process. The Clock Domain Crossings generate asynchronous events in synchronous designs, causing the metastability effect and introducing functional non-determinism. Static CDC code analysis with LINT tools completely eliminates the metastability sources. Also, CDC code analysis generates assertions (functional checkers) for functional validation of clock domain crossing logic. The task of functional simulation is to ensure that CDC-related assertions pass for any possible valid design stimuli. The above requirement is not achievable in practice, as functional verification tests do not fully cover input space and do not target specifically the CDC-related logic. Formal model checking may completely prove some of CDC-related properties, while being unable to prove all of them. Also, formal model checking heavily depends on functional constraints (assumptions), which may contain bugs themselves.

Therefore, existing CDC verification methods - Static CDC analysis, augmented with functional verification using either dynamic or formal methods still may not guarantee the verification completeness. The proposed CDC verification method, while being combined with the previously listed ones, significantly increases the confidence that clock domain crossing logic performs flawlessly in the mission and safety-critical designs.

Introduction

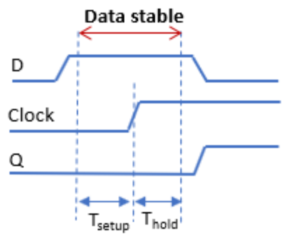
Modern FPGA devices capabilities compete with ASIC devices in terms of size and complexity. However, there is still no well-defined CDC verification methodology to guarantee that mission-critical FPGA devices are completely safe from the clock domain crossing perspective. Currently, CDC verification is performed primarily at RTL level using static code analysis, dynamic simulation and formal model checking. Due to the overall complexity, the CDC verification process may be still error-prone. Therefore, it is important to verify CDC issues at implementation as well. Paper [1] describes how to verify CDC issues in ASIC design using emulation. It introduces the CDC jitter model controlled using SCE-MI 2.0 standard interface. However, the described method cannot be applied for the FPGA devices as well. Current paper presents an efficient way to verify clock domain crossing issues at FPGA implementation stage.

The Clock Domain Crossing verification in one of the most challenging topics of complex FPGA verification process. Various methods and tools have to be used to guarantee flawless multi-clock design operation. These tools include static design verification, functional simulation and formal property checking. Each method verifies clock domain crossings from different angles, leaving the probability of non-verified CDC-related corner cases to remain in mission-critical designs.

The Clock Domain Crossing issues

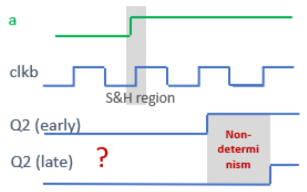
The clock domain crossing issues are the result of asynchronous signals activity on the clock domain boundaries. This activity leads to the metastability effect and to functional non-determinism. The metastability effect leads to excessive current flow, causing chip burnout in critical cases. Also, the metastability effect leads to functional non-determinism as it is impossible to ensure deterministic delay for the clock-domain crossing signals. So, functional verification for the designs with multiple clock domains should include thorough testing that CDC-injected non-determinism does not influence on the overall deterministic behavior and functions.

A. *The metastability effect*

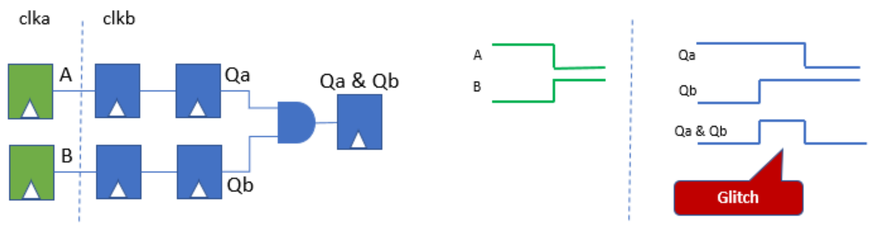


In sequential design, the data input of all sequential elements should be stable for the certain time around sequential element data capturing event:
In a case of above rule violation, sequential element output may bounce for the indefinite amount of time around the intermediate (metastable) state. During this time, sequential element as well as all connected to sequential element combinational elements experience higher than usual leakage current, leading to chip overheat and even burnout.
The two or more Flip-Flop synchronizer is the best way to reduce the impact of metastability effect. Since there is no logic between the flip-flops, the intermediate logic value appears only between the Flip-Flops. Assuming that receiving clock period is greater than signal meta-stability time, the second Flip-Flop latches known logic value, propagating it to the next logic stages. If the above assumption is considered to be wrong, there is a need to use three or more flip-flops in the synchronization chain.
Static Code analysis tools are able to ensure the that design has proper design structures (such as 2-FF synchronizer) at all clock domain crossings. To do so, design constraints has to be feed for static code analysis as well. It is important to mention that static code analysis relies on the correctness of clocks and input/output clock phases constraints.

B. *Functional non-determinism*

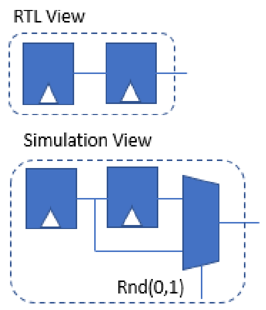


From the functional perspective, clock domain crossings introduce non-determinism into the hardware designs. The delay through the synchronizer is generally not predictable, as it is show in the following picture:
In this picture, signal *a* change may propagate to the second flip-flop output in either one or two clock cycles.
In most cases, designs may tolerate non-deterministic signal delay though the synchronizer. Usually, there are no problems transferring software configuration settings to another clock domains since these signals could be treated as "pseudo-static". Also, there are no problems transferring single-bit control signals with non-deterministic delay. However, non-deterministic delay may cause functional problems during reconvergence, when related control signals converge in the receiving clock domain. For example, two related control signals may generate harmful glitch as it is shown in the following picture:



CDC Functional Verification Methodology

Static Verification methods with linting tools are able to verify that proper design structures are present at clock domain crossings. Also, they are able to generate design assertions verifying such properties as pulse propagation though the clock domain boundary, data bus stability during it's capture in the receiving clock domain and so on. Other verification methods have to prove these properties for all possible design stimulus.
One of the most commonly-used property verification methods is dynamic simulation. CDC-related assertions could be added to functional regressions with minimal efforts. In addition to assertions, there is a need to properly model the CDC-injected non-determinism, or CDC jitter. This could be modeled varying clock frequencies and phases as well as using special pseudo-random synchronizer model, as shown in the following picture:



Dynamic simulation with CDC jitter and assertions is an efficient way to reveal most of the CDC-related "shallow" bugs. However, the capabilities of dynamic simulation are still limited because of the following reasons:

- Usually, CDC-related dynamic verification reuses functional tests and regressions, targeted to verify design functions according to the verification plan. In the absence of CDC-targeted tests and regressions the CDC verification may be incomplete.
- It is difficult to setup functional coverage on for CDC verification, as CDC jitter randomization coverage has to be crossed with specific CDC-related signals coverage.
- It may be complicated to collect desired coverage using functional verification tests and testbenches, as they target other design functions

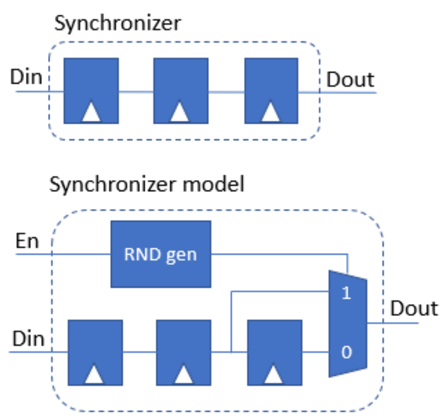
Formal model checking could be used to verify some of CDC-related properties as well. However, formal technology has limitations when applied to large designs. As the number of state variables in the system increases, the size of the system state space, representing current design, grows exponentially. This is called the "state explosion problem". Assumptions constrain system design space, but it is difficult to develop them in correct and functionally complete way. So, there is still a risk that formal proof of selected CDC properties, relied on the human-written assumption, is not complete and may fail in real design.

Static verification of CDC structures with LINT tools relies on the design constraints correctness, such as clocks and clock phases definitions. Design with complex clock multiplexing schemes also require static functional constrains and case analysis for each combination of constraint values. These designs may be also error-prone due to human mistakes of design constrains and case analysis incompleteness.

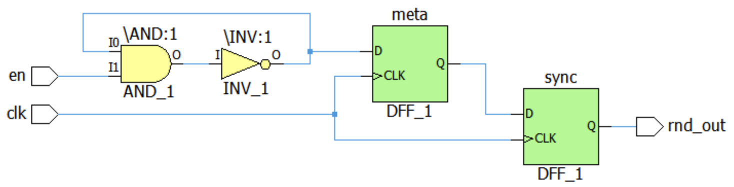
In the FPGA world, there is an opinion that Lab testing is sufficient to test multi-clock design correctness. While Lab test is able to generate huge amount of design stimulus and provide more realistic (comparing to functional simulation) clock sources, it is still limited as it does not execute all possible process, temperature and power variations that may be occur later.

So, the overall CDC verification process is quite complex, as it spans through the number of processes and verification methods. The processes require either functional of timing constraints, suffering from human-related mistakes and possible incompleteness. For the multi-clock mission-critical designs, there is a need to add another standalone CDC verification method to confirm design functional correctness during data and control transfers between asynchronous clocks domains. Current article presents an idea of synthesizable synchronizers to model and enhance the CDC jitter effect. Being implemented in design for the CDC-specific Lab testing, these synchronizer models efficiently verify design robustness from the clock domain crossing perspective.

Clock Domain Crossing Jitter verification in FPGA



Paper [1] presents an idea on how to verify the Clock-Domain Crossing Jitter in ASIC designs using Emulation. In this paper, we describe how to verify the Clock-Domain Crossing Jitter in FPAG devices during Lab Verification. The method does not require additional equipment and could be used widely increasing the quality of the clock domain verification process.
The key idea of proposed verification technology is the usage of synthesizable CDC jitter models. Regular N-DFF synchronizers are replaced by synthesizable synchronizer models. These models implement the CDC jitter effect, providing random throughput delay for signal changes, as it is presented in the following picture:
The synchronizer model implies the minimum amount of three stages. There are no problems to implement synchronizer model with more that three stages as well, if there is a need to further increase the MTBF (Mean Time Before Failures).
Then we assume that synchronizer is implemented in a separate design unit, and LINT tools verify that these units reside on all required clock domain crossings. For the purpose of FPGA-based CDC verification, there is a need to replace all synchronizer instances with synthesizable synchronizer model instances. For each receiving clock cycle, the model randomly selects the output of either second of third synchronizer stage, connecting it to the output of synchronizer. This behavior models CDC-related setup violation, achieving random one-clock shift of changed input signal.
Paper [1] describes a model for the setup/hold violation, with the worst-case signal change shift of two cycles. However, it is unlikely that related synchronizers will experience both setup and hold problems, with modeled 2-clock cycle delay between related signals. For the FPGA-specific CDC verification, the one-clock cycle delay through the synchronizer should be sufficient.
There exist different ways to implement random component in FPGA. Paper [1] suggests using LFSR-based pseudo-random generator. In order to produce different results, LFSR-based random generators have to be initialized with different initial values. These initial values could be either supplied through the software interface or hard-coded in RTL. In the first case, design size and complexity increases, leading to more complex routing and timing closure. In the second case, the number of different random delay combinations is limited, requiring design re-compilation with different initial values sets.
Current paper proposes ring oscillator (RO) usage for generating random sequences. The presence of process variation inside logic gates and wires causes an uneven delay across the chip. As a result, different ROs produces slightly different frequencies, and accumulated phase shift between ring oscillators leads to unique randomization results. The following picture presents ring oscillator schematics:



The ring oscillator contains single inverter only. This inverter is mapped into the single LUT in Xilinx Vivado. The single-inverter ring oscillator output is connected to the enabled-based synchronizer, built from two “meta” and “sync” flip-flops. The metastability of “meta” flip-flop is a main source of entropy for the random bit generator inside the synchronizer model cell. In order to increase the randomization quality, there is a need to:

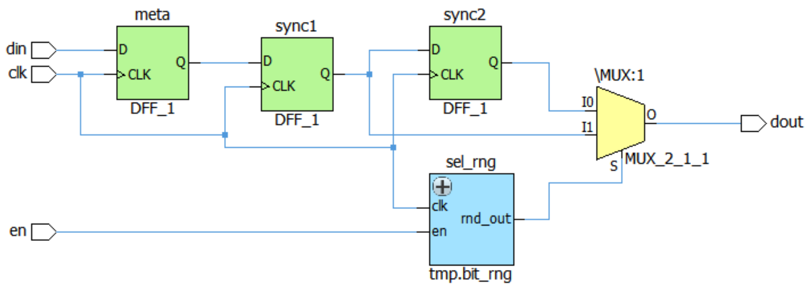
- Increase the metastability effect frequency. In our case, the LUT delay implementing single inverter varies from 0.097 (FAST_MAX) to 0.116 (SLOW_MIN), so oscillator frequency is orders of magnitude higher than capture clock frequency.

- Increase the slew rate of asynchronous signal feeding to “meta” flip-flop data pin. This widens the metastability window, increasing the randomization quality of random generator.

The excessive amount of metastability effects leads to leakage current increase and chip overheat. Therefore, there is a need to use the “enable” control signals to outline the periods when randomization, and therefore the metastability effect should happen. In our design, we enable random data capture in synchronizer only upon the change of incoming to main synchronizer model “din” signal.

The other aspect of randomization is the uniqueness of different randomizers functions. Different randomizers should produce un-correlated results. According to our experiments, each 1-bit randomizer provides unique results. Also, we have examined some researches in this area and found that even under the same power and voltage conditions, the jitter of a LUT propagation delay is approximately 2 percent. (Paper [4]). This jitter leads to the floating phases shifts between different oscillators, causing different oscillator to behave as an independent clock source.

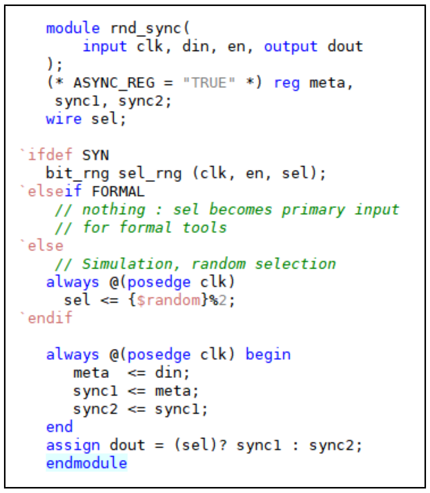
The following schematics picture illustrates the design of the 3-stage synchronizer with random jitter model implementation:



Capturing random signal in random generator occurs only when input data is changing. After each input signal change, new random value propagated to the multiplexer select bit, randomly choosing the outputs of sync1 or sync2 flip-flops to be connected to the synchronizer cell output. The random selection between two or three synchronizer stages pessimistically models the CDC jitter effect in synchronizer cell.
Since the sync2 synchronizer cell input is connected to multiplexor input, the input capacitance of second synchronizer cell increases, reducing the slew rate at synchronizer input. This, in turn, slightly reduces the MTBF (mean time between failures), increasing the rate of possible metastability states. It is desirable to group synchronizer cell contents for placement, thus reducing interconnect wire capacitance between synchronizer data input and the LUT implementing multiplexer function.

The Verilog code snippets demonstrate synchronizer and random bit generator designs for Xilinx technology (The Intel FPGA code snippets re similar, except from the IntelFPGA-specific attributes). It is possible to configure the synchronizer for three usage cases:

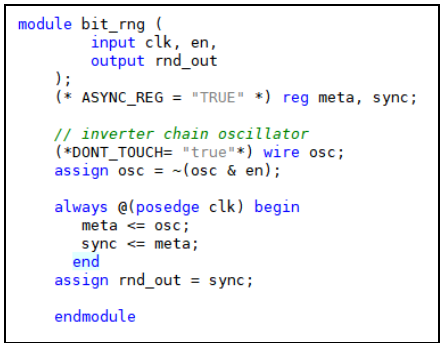
1. FPGA implementation: In this case, mux select signal receives “random” signal from the 1-bit random generator instance
2. Dynamic Simulation: The single-bit randomizer instance is replaced by cycle-base randomizer code based on Verilog $random function. It may be desirable to change random seeds to achieve different synchronizer delays cominpations in random simulations.
3. Formal Analysis: The single-bit randomizer instance is omitted for the Formal Analysis. The MUX select input is treated as an unconstrained input for Formal tools. This allows Formal tools to find scenarios to fail CDC-related assertions.



```
   module rnd_sync(
       input clk, din, en, output dout
   );
   (* ASYNC_REG = "TRUE" *) reg meta,
    sync1, sync2;
   wire sel;

`ifdef SYN
   bit_rng sel_rng (clk, en, sel);
`elseif FORMAL
    // nothing : sel becomes primary input
    // for formal tools
`else
    // Simulation, random selection
   always @(posedge clk)
     sel <= {$random}%2;
`endif

   always @(posedge clk) begin
      meta  <= din;
      sync1 <= meta;
      sync2 <= sync1;
   end
   assign dout = (sel)? sync1 : sync2;
   endmodule
```



```
module bit_rng (
        input clk, en,
        output rnd_out
    );
    (* ASYNC_REG = "TRUE" *) reg meta, sync;

    // inverter chain oscillator
    (*DONT_TOUCH= "true"*) wire osc;
    assign osc = ~(osc & en);

    always @(posedge clk) begin
       meta <= osc;
       sync <= meta;
      end
    assign rnd_out = sync;

    endmodule
```

FPGA-specific CDC verification process

Proposed synchronizer models enhance the influence of CDC jitter on a multiple-clock design. As FPGA devices delays vary depending on power, temperature and process variations, the Lab-based multiple-clock design validation may not reveal all possible clock-domain crossing scenarios. The proposed solution introduces into design synchronizers with random CDC jitter effect. These synchronizers act as a "CDC effect amplifiers", significantly increasing the number of possible delay combinations at clock domain crossings. In a case Lab verification passes with implemented synchronizer models, then design is robust from the CDC perspective.
We propose the following process to verify that FPGA is free from the clock domain crossing issues:

I. Use the RTL linting tool to identify CDC crossing and verify that RTL contains proper design patterns at the clock domain boundaries. Also, identify all divergence, convergence and reconvergence issues and related code.
II. Try to change design to eliminate all divergence, convergence and reconvergence issues. In a case it is not possible, mark the groups of 3-stage synchronizers, related to every one of above issues.
III. Replace all 3-stage synchronizers in design with synchronizer models, and start running RTL simulation. For the simulation purposes, replace random generator instance in synchronizer model with random function (such as $random in Verilog/SystemVerilog).
IV. Implement FPGA using synchronizer model with synthesizable random generator. Run Lab tests to verify that design is clean from the CDC-related issues.

Usually, it is difficult to debug and detect the root cause of the CDC issues. The following debugging process may help designers to nail down CDC-related bugs during Lab testing:

I. Make sure that functional failure is probabilistic. When functional failure detected, re-run design again few times to make sure failure is probabilistic and occurs at different time moments. In a case functional failure is probabilistic, it most likely corresponds to clock domain crossing issues. Also, run design without the synchronizer models to confirm above assumption.
II. Re-implement the design, connecting synchronizer model enable signals into the groups and controlling them from the software interface. These groups may be defined according to design hierarchy, or using the information about synchronizers, related to the same divergence / convergence / reconvergence group.

Conclusion

The above paper presents the FPGA-centric Clock Domain Crossing verification methodology for the mission-critical designs with multiple clock domains. This methodology spans from static verification with LINT tools through random simulation and finally Lab testing with synchronizer jitter models. These models amplify CDC effect during Lab testing, effectively enhancing the CDC effect influence on design functional behavior.

The CDC synchronizer models were designed and tested for both Xilinx and Intel FPGA technologies. Although randomizers are not ideal and don't implement the true random generation function, they provide significant benefits for the CDC verification process. Taking into account that the CDC-specific design overhead is minimal, we believe that described method provides significant benefits to efficiently verify CDC issues in mission-critical FPGA designs.

References

[1] A.Hari, S. Krishnamurthy, A. Jain. Y. Badaya. "Verification of Clock Domain Crossing Jitter and Metastability Tolerance using Emulation", DVCON US, 2012

[2] A.Hari, S. Krishnamurthy, Y. Badaya, "Systematically Achieving CDC Verification Closure based on Coverage Models and Coverage Metrics", DVCON US, 2012

[3] Tai Ly, Neil Hand, Chris Ka-kei Kwok, "Formally verifying clock domain crocking jitter using assertion-based verification", 2004

[4] Boyan Valtchanov, Alain Aubert, Florent Bernard, Viktor Fischer Laboratoire Hubert Curien, "Modeling and observing the jitter in ring oscillators implemented in FPGAs"