Verification to Static Sign-off: Automotive Approach

Krishna B Pandit
PG – VLSI Design and Embedded Systems
Dept of ECE RVCE
Bangalore -59

Sudha R Karbari
Assistant Professor
Dept of ECE RVCE
Bangalore - 59

***Abstract*— As the technology nodes are getting smaller and smaller the design of a SoC is becoming more complex day by day. Modern day SoCs are a collection of individual intellectual properties [IP]. Each IP has its own clocking regime and each clock belongs to the separate clock domains. This leads to the challenge in verification of SoCs. The functional verification of SoCs is becoming more and more difficult due to the data transferred between individual IP modules. When data transfers between two IPs the data crosses clock domains leading to clock domain crossing [CDC]. For a SoC to be free of all CDC errors, verification is paramount along with meeting the timing requirements. This work focuses on the conversion of SVAs to timing constraints for timing sign-off. Timing sign-off requires the need to detect all the untimed paths present in the design. Validating each data path is meeting the timing requirements. Automation of the detection and conversion of assertion to timing constraints is done here.**

Keywords—Clock Domain Crossing, Verification, System Verilog Assertions, Verification, Sign-off

I. INTRODUCTION

Modern day SoCs are extremely complex in their composition. Each SoC is composed of smaller individual modules called as intellectual properties. Each IP block operates on its own clock and clock domains. When data transfers in-between the IPs, the data crosses clock domains giving raise to clock domain crossing [CDC]. When transfers from one clock domain to another, it is called as clock domain crossing. CDC usually happens when the sending and the receiving clocks are asynchronous w.r.t to each other.

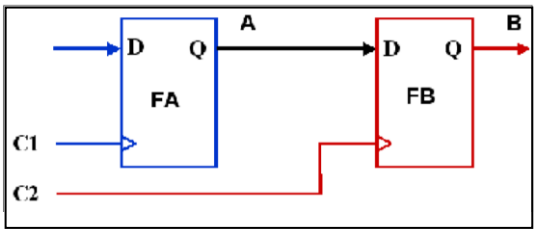

Figure 1: Clock Domain Crossing

In figure 1, flip-flop F_A and flip-flop F_B are clocked by two clocks C_1 and C_2. C_1 and C_2 are asynchronous w.r.t each other. Data is said to cross from C_1 domain to C_2 domain. The major issues of CDC are metastability, data loss and data incoherency. Metastability is the major error of CDC. Metastability is the state of the flip-flop when the sampled data is in the unknown state of either 1 or 0.

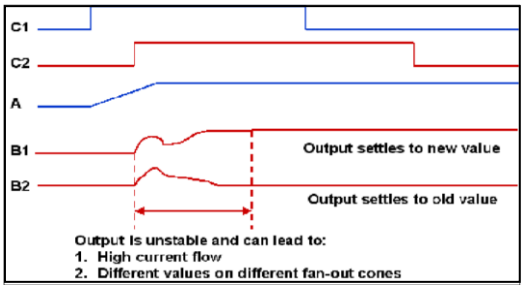

Figure 2: Metastability

When the transition on signal A happens very close to the rising edge of C_2, violates the setup time or the hold time of the flip-flop clocked by C_2 i.e F_B goes metastable. The value sampled by the flip-flop may settle to either B_1 or B_2 and is completely unknown. Metastability leads to data loss and data incoherency. Two of the most common problems of CDC. Data loss whenever the destination flip-flop captures source data, if each transition is captured, then data is not lost. Data loss happens in a serial transmission of data bits where each transmitted bit may settle at either 0 or 1 due to metastability owing to data loss. To prevent data loss, the sending end data is to be held stable for a minimum period of 1.5x clock period of the receiving clock. This includes the

setup-hold margin and the single clock cycle latency in the receiving domain.

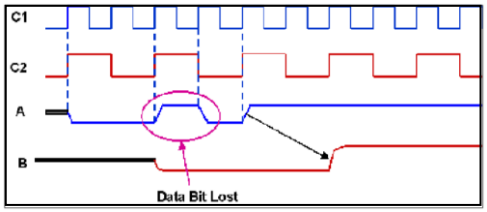

Figure 3: Data Loss

Figure 3 represents data loss. Here instead of capturing 1011 the flip-flop captures 0011 losing the first bit. Data incoherency is the third most common error of CDC. When multi-bit data cross clock domains, due to metastability each bit transferred may settle at a different logic state compared to the input. The data sent and the data received will not match and hence are incoherent. To solve this problem handshaking and asynchronous FIFO based synchronizers are employed.

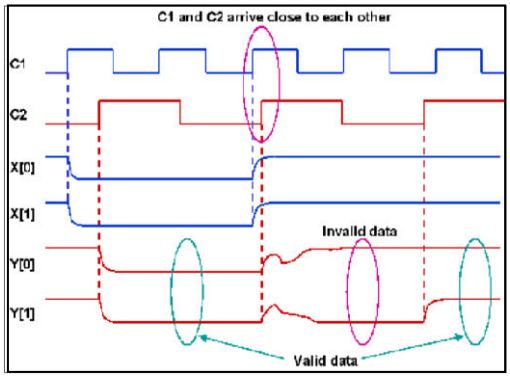

Figure 4: Data Incoherency

Figure 4 represents data incoherency. The data being sent is X[0] and X[1] and the data received is Y[0] and Y[1]. Due to metastability, Y[1] does not match with X[1] leading to data incoherency.

Verification and Static Sign-off

As known from the previous section, the problems of CDC are enormous when gone unnoticed. It may even lead to functional and electrical failure of the chip. To combat this problem, synchronizers are employed across clock domains to synchronize data with the receiving clock. There are various types of synchronizers present and based on the data being sent, one can employ different types of synchronizers. Each synchronizing scheme comes with a set of built in SystemVerilog Assertions [SVAs] to verify their functionality and their usage. The verification of Practices from the past have made use of this part to verify synchronizers and declare a chip as CDC bug free.

Clock Domain Crossing Synchronizers

There are various types of synchronizers used for CDC synchronization. Here are the most common schemes along with their SVAs.

2 Flip-flop synchronizer

The basic of all the type of synchronizer is the 2 flip flop synchronizer. It consists of 2 back-to-back flip-flops. The 2 flip-flop synchronizer resolves metastability issues.

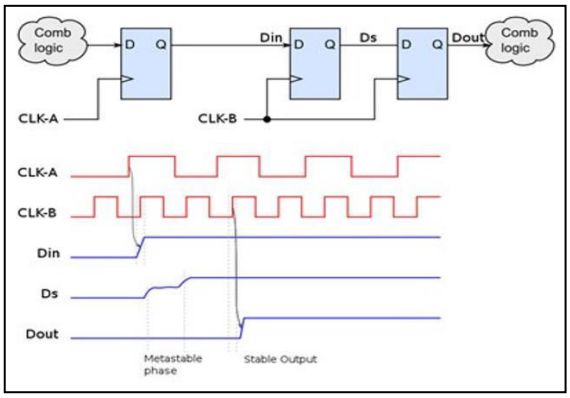

Figure 5: 2 flip-flop Synchronizer

The major checks for 2FF synchronizer are the metastability check and data stability check when it comes to verification. Metastability check checks whether there are glitches in the data path and the data stability check checks whether the data is stable for a minimum of 1.5x the receiving clock cycles. The SVAs for the two checks are given below.

```
property p_stability;
@(posedge clk-B)
!$stable(Din) |=> $stable(Din) [*2];
endproperty : p_stability

property p_no_glitch;
logic data;
@(D_in)
(1, data = !Din) |=> @(posedge clk-B)
(Din == data);
endproperty : p_no_glitch

assert property(p_stability);
assert property(p_no_glitch);
```

The above properties describe the SVAs for a 2 flip-flop synchronizer. SVAs must pass in the simulation for the design to be functionally correct.

Handshaking Synchronizer

Handshaking synchronizer works on the protocol of request and acknowledge. There are two types of handshaking synchronizers- 2phase and 4phase. The major functional criteria for a handshaking synchronizer are – each request must get an acknowledge, each acknowledge must be

preceded by a request and when the request is asserted, the data sent must be stable.

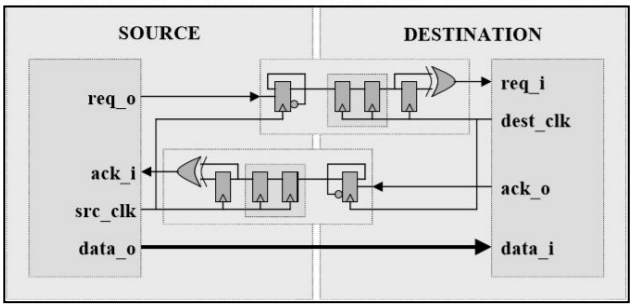

Figure 6: Handshaking Synchronizer

The sample SVAs for a handshake synchronizer are:

```
sequence Data_tx;
@(posedge clk)
req ##1 !req [*1:max] ##0 ack;
endsequence

property Req_G_Ack;
@(posedge clk)
req |-> Data_tx;
endproperty

property Ack_H_Req;
@(posedge clk)
ack |->Data_tx.ended;
endproperty

property Data_Stability;
@(posedge clk)
req |=> $stable(data) [*1:max] ##0 ack;
endproperty

assert property(Req_G_Ack);
assert property(Ack_H_Req);
assert property(Data_Stability);
```

Asynchronous FIFO Based Synchronizer

The most commonly used synchronizer is dual clock asynchronous FIFO based synchronizer. It is composed of a memory element a block RAM or a dual port SRAM, a 2 flip-flop synchronizer, binary to gray and gray to binary counters write and read control logics separated by asynchronous read and write clocks. This synchronizer is used to synchronize multibit data crossing clock domains such as address bus.

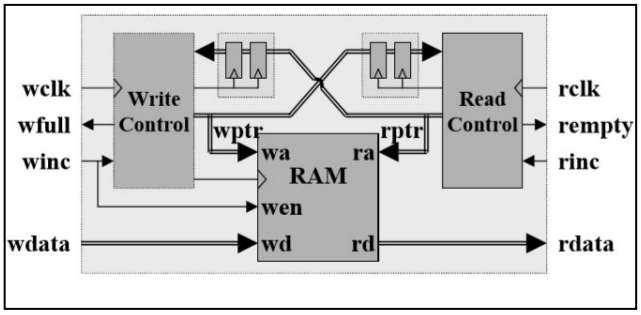

Figure 7: Asynchronous FIFO Synchronizer

Figure 7 shows the basic block diagram of asynchronous FIFO synchronizer. The basic checks for a FIFO synchronizer are – never write a full FIFO, never read to an empty FIFO, binary to gray checks and 2 flip-flop synchronizer checks. The sample SVAs of a FIFO based synchronizer is given below.

```
property Access;
@(posedge clk)
inc |->!flag;
endproperty

property Gray_Code
@(posedge clk) disable iff (!rst_n)
!$stable(data) |-> $onehot(data ^ $past(data));
endproperty
```

The property Access is used for both the write and read domains. The property Gray_Code checks whether the consecutive pointers have a single bit change between them.

Sign-Off

For the proper functioning of SoC, not only should it be functionally correct but also it must meet the timing requirements. The timing requirements are checked at the synthesis stage. There are a variety of timing checks that are performed on the SoC for timing sign-off. They are:

Setup Check [max-delay check]

Setup checks are done on the receiving flip-flop to check whether the incoming data meets the set-up time of the flip-flop. For positive edge triggered launch and capture flops, setup checks are called as single cycle checks. Since the data launched in the present posedge is checked for setup violation in the next posedge. Setup checks consider the maximum delays of the data and clock paths. Hence called as max-delay checks.

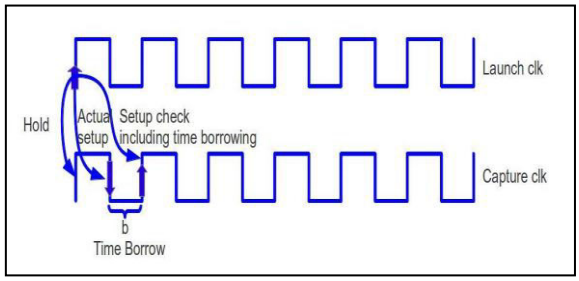

Figure 8: Setup Check hold check

Hold Checks [min-delay checks]
Hold checks check if the hold time of the flip-flop is met by the incoming data. For flip-flops triggered on the same clock edge, the hold checks are termed as zero cycle checks. Hence hold checks are not dependent on the frequency of the clock triggering the flip-flop.

Verification to Sign-off Methodology
Previous works in this domain have treated verification and sign-off as two separate entities in the design of an SoC. The verification checks verify the SoC based on functionality whereas the timing sign-off checks verifies the SoC based entirely on timing. The functional pre-synthesis verification does not consider the net and the cell delays and the timing sing-off checks post synthesis, do not take into account the functionality of the chip. For a proper and reliable functioning of the chip both in terms of logical functionality and meeting the timing requirements, there must be an intersection between verification and sign-off.

Methodology
The verification to sign-off methodology bridges the gap between verification and static sign-off first manually and then the process is automated. In the manual process the methodology is as follows:

Step 1:
For the closing the gap between verification and sign-off, it is paramount to find out the number of asynchronous paths present in the given design. This is done by running CDC on RTL on the tool spyglass. This tool verifies the each and every asynchronous crossing and the presence of synchronization scheme for that particular crossing. This step requires the inputs to the tool about the design in the form of tcl constraints and based on the design, waivers.

Step 2:
After a successful CDC run, with resolution of all the CDC violations by writing proper constraints and waivers, the tool generates the SVAs for each waiver present in the waiver file and each constraint in the constraint file. These SVAs are verified along with the SVAs written for each type of synchronization cell in the simulation of the design.

Step 3:
After verifying the SVAs for constraints, waivers and synchronization schemes, we need to detect the presence and the number of untimed paths present in the design. Untimed paths are basically paths where we cannot put a timing check. All asynchronous paths present in the design are untimed paths along with multi-cycle paths and false paths. Here to perform the timing checks on these paths, the constraints are written by the conversion of SVAs for the synchronization cell to timing check. This timing check verifies the timing of that given data path along the synchronization cell. Hence, we're verifying the functionality through SVAs and the timing by converting those SVAs to timing checks for the timing sign-off.

Step 4:
As this is a manual and time-consuming job, scripts were written to find out the number of asynchronous paths and to find the type of synchronization scheme used and the number of a synchronization scheme present in the design.

Step 5:
The manual mapping of SVAs to the timing checks were put in a file for each of the synchronization scheme present in the design. This gives the necessary timing check for a given SVA for a given synchronizer present in the design.

Discussion and Future Scope
This project combines two of the most important part of VLSI design flow i.e verification and synthesis. The transition from verifying the design functionally into the sign-off of the design w.r.t timing has a huge impact on how future SoCs are designed. The verification to sign-off methodology removes the human dependency in the type and manner of checks that are supposed to be done on a multi clock complex SoC. This project does not take into account the resets present in the design. The major factor for resets is the de-assertion of resets if it is close w.r.t the clock, it requires a reset synchronizer and gives raise to extra complexities. This can be used as a reference for the future works on resets.

References

[1] Pranav Ashar, Vinod Viswanath "Closing the Verification Gap with Static Sign-off" ISQED 2019
[2] Pranav Ashar, Vikas Sachdeva, Vinod Viswanath, "Failures and verification solutions related to untimed paths in SOCs," ISQED 2017: 460-465
[3] Sourabh Verma, Ashima S Dabare "Understanding clock domain Crossing Issues" EE Times India December 2007

[4] Matt Litterick Verilab "Pragmatic Simulation-Based Verification of Clock Domain Crossing Signals and Jitter using SystemVerilog Assertions" DVCon 2006
[5] Sachin Hatture, Sudhir Dighe, "Open loop and closed loop solutions for clock domain crossing faults" GCCT - 2015
[6] Ghaith Tarawneh, Andrey Mokhov, Alex Yakovlev, "Formal Verification of Clock Domain Crossing using Gate-level Models of Metastable Flip-Flops" DATE – 2016 : 1060 – 1065
[7] Salomon Beer, Ran Ginosar "Eleven Ways to boost synchronizer" IEEE Transactions on VLSI Systems 2014
[8] Sachin Hatture, Sudhir Dighe "Multi Clock Domain Synchronizers" 2015 INTERNATlONAL CONFERENCE ON COMPUTATlON OF POWER, ENERGY, INFORMATlON AND COMMUNICA TlON: 403-408
[9] Matej Bartik "Clock Domain Crossing- An Advanced course for Future digital engineers" Mediterranean Conference on Embedded Computing 2018
[10] Clifford E Cummings "Clock Domain Crossing Design and Verification Using System Verilog" SNUG 2008