Key Gochas in implementing CDC for various Bus Protocols

Nikita Gulliya nikita@agnisys.com | Mukesh Kumar Singh Mukesh@agnisys.com | Abhishek Bora abhishek@agnisys.com
Agnisys Inc

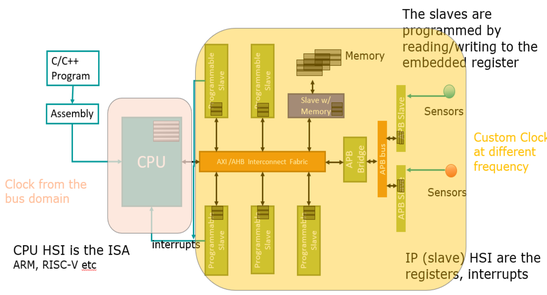
Abstract

In the industry where the designs are getting complex with huge hierarchy, there is often a requirement of multiple clock domains. But while, multiple clock domains exist various complications occurs, such as setup and hold time differences which further leads to a metastability state. Some of these errors can be caught at an early stage in the design while few occur at a later verification stage. Other issues include violation of the protocols. The 2-D flip flop technique only solves the CDC issues for 1-bit transaction. Automation has been done for the generation of proper code of design and verification from the specification itself for multi-bit transactions eliminating all the CDC related issues. SoC level specification can be used for the generation of correct design code for clock domain crossing, covering all the scenarios and various bus protocols. CDC has been implemented between various bus protocols used in the industry and the custom bus which can be user defined. The paper will talk about the simulation results obtained for the implementation of a low power RTL design and techniques used for interconnection with various bus protocols.

Introduction

Often the design demands dealing with various clock domains. Specially when the design is implemented with a user defined decode logic working on a custom bus. The interfacing can be between custom clock and any other bus protocol, the frequencies of the two clocks may vary in this case leading to various issues such as metastability, loss of data, data incoherence etc. The figure hereby shows the CPU and IP Hardware software interface(HSI). The CPU contains bunch of register in it, C/C++ program with the help of compiler is converted in Assembly. The assembly code is burnt on to the CPU with the help of interconnect fabric, which can be based on any bus protocol such as AMBA- AXI, APB, AHB.

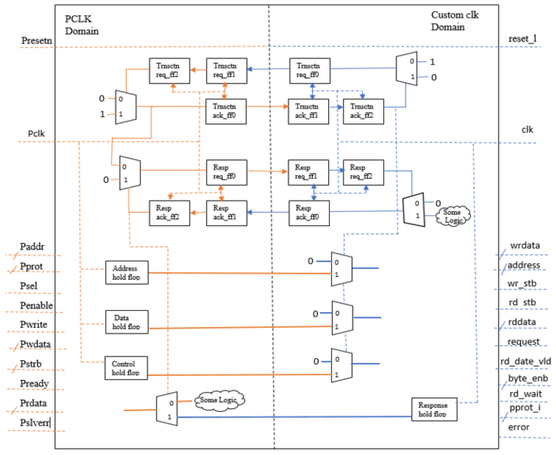
CPU HSI can be divided into two parts. The software side or the bus interface is working on a different clock frequency than the Hardware side or the custom clock domain. The programmable slave side is termed as the custom clock domain. The clock domain crossing has been implemented with respect to two clocks. The first clock is the custom clock that the user must be working on with its own application logic from the slave side, and the other clock is from the bus domain termed as the master clock. Both these clocks must be working on different clock frequencies, automation has been done for various bus protocols, which includes, AMBA-AHB, AXI, APB, Tilelink etc.



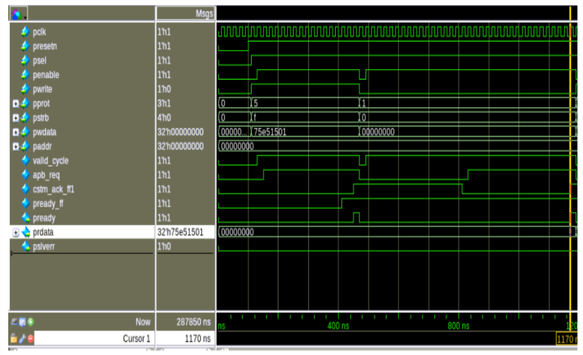
CDC from Software Side

A. CDC for AMBA-APB Bus Protocol

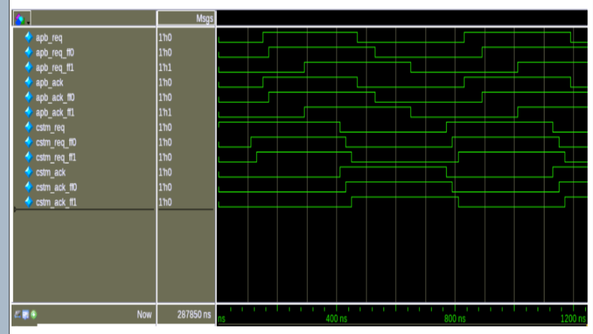
- According to the APB Bus protocol, when the *PSEL* signal from bus is high then it indicate that the bus is in setup phase. In the next clock cycle when the *PENABLE* signal goes low to high then the bus is in the access phase and expects custom to derive *PSLVERR* and *PREADY*.
- In access phase if *PREADY* is high and *PSLVERR* is low it indicates transaction is complete. If *PREADY* is low, it indicates slave is still working on current transaction and not ready for another one. If *PREADY* is high and *PSLVERR* is also high, then it indicates that there is some error in the transaction.
- To perform cdc between APB *PCLK* domain and custom clock handshake signals are used, request and acknowledge signal for transaction from APB to CUSTOM and request and acknowledge for response from custom to APB.
- By default transaction request from custom will stay high and it will pass to *PCLK* domain using 2-ff synchronizer. When APB goes in access phase it will check if request for transaction is there or not if request is present then it will derive *PREADY* low, put the address, data and control in a flop then send transaction acknowledge to custom clock domain using 2-dff and also generate request for response and send it to custom using 2-ff synchronizer. It will keep *PREADY* low until it gets response acknowledge from custom domain.
- In custom clock, when transaction acknowledge arrive through 2-ff synchronizer, it will deassert transaction request, take the address, data and control to generate valid response, then store that response in a flop and assert response acknowledge. After this it will wait for transaction acknowledge to go low so that it can generate transaction request again.
- In APB *PCLK* domain, when it gets response acknowledge then it take the valid response from response flop and assert the *PREADY* and deassert transaction acknowledge and response request. So, that's how handshake takes place between APB and custom.



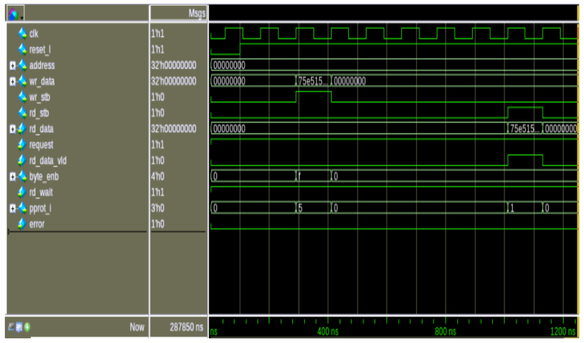
The simulation waveforms shown in figure below shows the simple read-write transaction in case of CDC for APB bus domain.



The figure below shows the handshaking taking place taking place between the two clock domains.



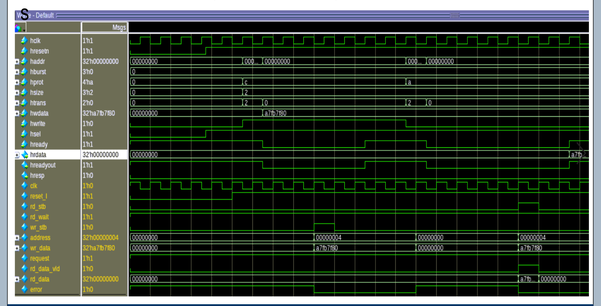
The figure below shows the CDC at the custom bus domain side, where when APB acknowledge is received, for the write transaction, the data write strobe and address are obtained.



B. CDC for AMBA-AHB Bus Protocol

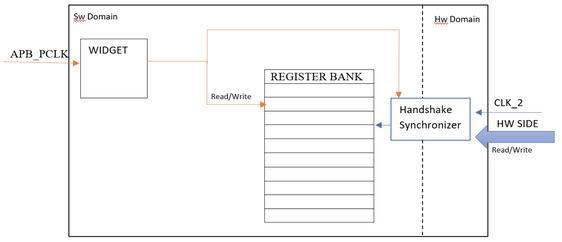
•For cdc between AMBA-AHB bus domain and custom clock domain, a similar approach as mentioned in AMBA-APB is followed.
•For normal operation: In address phase, address is stored in flop and then during data phase *HREADY* is derived low and wait for transaction request from custom.
•When custom request come through 2-ff synchronizer transaction acknowledge signal is asserted to indicate that address, data and control signals are available in flop and response request signal is asserted for valid response.

- When custom domain receive transaction acknowledge, it will take data, address and control from the flop and deassert transaction request low then store valid response in flop and assert response acknowledge signal high.
- When AHB domain receive response acknowledge, it will derive low transaction acknowledge and response request. Also it will assert *HREADY* high and provide valid response to the AHB. Hence transaction gets completed through handshaking.
- For burst/pipeline operation: similar approach is used except the usage of two flops to store addresses. During address phase address of transaction1 is stored in one flop and then in data phase transaction2's address is stored in another flop, then *HREADY* is de-asserted and waits for response acknowledge. When a transaction gets completed, then that address is moved from next address flop to address flop.



CDC from Hardware side

In the diagram shown below, two clocks one coming from bus (say APB) on which read write are will take place on register bank from software side and the other one is user define clock(CLK_2) on which hardware side will do the read/write operation on the register bank. The handshake synchronizer is used to translate the transaction coming in from CLK_2 domain to APB_PCLK domain.



Results

Therefore, various techniques such as handshake synchronizer have been used according to the bus protocol for the implementation of clock domain crossing. The implementation and simulation results inferred low power RTL generation. The clock domain crossing was seen from software bus side and hardware end. The disadvantage from hardware side with the current solution is that, there can be multiple requests from the hardware side at the same time to different registers, in this case multiple handshake synchronizers will be created. We are in pursuit to find out the best way to implement the clock domain crossing from hardware side.

TEMPLATE DESIGN © 2008
www.PosterPresentations.com