International Journal of Computing and Digital Systems

Automated Verification and Clock Frequency Characteristics in CDC Solution

Akitoshi Matsuda[1] and Shinichi Baba[2]

[1]*Dept. of Automotive Science, Kyushu University, Fukuoka 814-0001, Japan*
[2]*Kyushu Embedded Forum, Fukuoka 814-0001, Japan*

e-mail: matsuda_aki@slrc.kyushu-u.ac.jp, shinichi.baba@nifty.com

Received 17 May. 2012, Revised 31 Oct. 2012, Accepted 8 Dec. 2012

Abstract: Recently, the design complexity of System-on-a-chip (SoC) has increased and additional functionalities have been added to SoC. Moreover, their operation clocks are frequently transferred from single clock domain to multiple clock domains. Because of the volume of the crossing signals and the variety methods of implementing crossed clock, the verification of clock domain crossings (CDCs) has become a very important and challenging task in deep submicron designs. Therefore, specialized CDC verification solutions can accurately detect CDC issues and efficiently debug the root causes of these problems that we require to perform design analyses. This paper describes certain case studies involving CDC verification and the relationship between clock and operating frequencies. Variable combination of clock frequencies "rclk" and "wclk" was evaluated for the asynchronous first-in first-out (FIFO) memory design. When a synchronizer was inserted, the power consumption decreased with the frequency; however, the performance achieved a "dead point" when "wclk" was about 1.3 times the value of "rclk." The results confirmed that different combinations of clock frequency affected the circuit characteristics.

Keywords: System-on-a-chip; clock domain crossings; clock frequency; CDC solution; asynchronous memory.

I. INTRODUCTION

Advanced System-on-a-chip (SoC) architectures support many asynchronous clock domains. The implementation of a faulty clock domain crossing (CDC) interface can cause metastability propagation downstream, resulting in functional errors. These errors are intermittent, and are often hard to detect and diagnose using traditional verification techniques such as simulations. Undetected CDC issues can cause chip failures in the field and result in expensive chip respins. The most precise, comprehensive, and innovative CDC solution performs structural and functional analyses to ensure that signals crossing asynchronous clock domains on SoC designs are reliably received [1].

CDC issues typically require considerable time and effort for debugging. Therefore, automatic verification and debugging capabilities are required in CDC solutions to reduce the turn-around-time (TAT) required for FPGA designs [2].

Our paper is structured as follows: Section 2 describes the background to multi-clock design. Section 3 gives an overview of CDC verification solution in three essential elements. Section 4 presents an outline of the CDC verification used in the asynchronous FIFO memory design. Section 5 describes a set of experiments that we conducted on diverse real-world applications on the asynchronous design, and furthermore presents the design methodology for the simulation and verification of FIFO memory design. Lastly, Section 6 discusses our conclusions and future work.

II. BACKGROUND

The number of advanced multiclocking architectures has increased dramatically because of the demands for high performance and low power consumption in SoC designs, recently. Interactions between these clock domains need to be verified to ensure correct functionality [3]. For multi-clock domain designs, Register Transfer Level (RTL) or gate-level simulation can not accurately capture the timing of data transfers between different clock domains. Consequently, even if the simulation is run, the device behavior can not be accurately predicted [4], and serious bugs might be missed in the verification flows [5]. Therefore, a specialized CDC verification tool is needed to deal with the difficult verification issue, which can not be handled by simulation-based verified methodologies [6].

III. CDC VERIFICATION FLOW

The CDC verification solution sets the industry benchmark by providing the following three essential elements for a complete CDC verification solution: automatic clock intent analysis (structural CDC analysis), integrated formal analysis, and dynamic CDC verification [7]. It is the only solution that enables accurate prediction of device behavior for an asynchronous clock interface [8]. It gives users the confidence that all CDC bugs will be found before tapeout, and expensive respins will be avoided [9].

A. Automatic clock intent analysis

Automatic clock intent analysis infers CDC intent from the design stage, and offers a comprehensive analysis that identifies clock or reset issues, incorrect or missing synchronization, glitch potentials, reconvergence, structurally unsafe crossings, and potential data or control crossings that require functional verification. This analysis is the fastest and most accurate because of the CDC solution's advanced correlation algorithms, and offers the most complete support for crossing styles in the industry.

For example, with structural analysis, the CDC solution can automatically identify the following issues:

1) Reset issues

In clock/reset issues, the CDC solution can identify issues related to missing synchronizers on the asynchronous reset crossings. Fig. 1 indicates two asynchronous clock domains "clka" and "clkb" and an asynchronous reset "async_reset," generated from the "clka" domain used to reset flops in the "clkb" domain. The problem with this circuit is that there is a missing synchronizer for the "clkb" domain reset.

The CDC solutions can automatically identify such issues with asynchronous reset signals. For more details on the reset related issues, CDC solution can identify the point of reset signals without the synchronizer. Therefore, we easy to be able to find out a point to be input by some synchronizers on circuit point.

2) Structural CDC issues

The CDC solution performs CDC analysis and identifies unsafe CDC structures in the design. Fig. 2 shows signals from different asynchronous clock domains combining and crossing into another clock domain. The crossing signal may have glitches that can cause problems in downstream logic. The CDC solution can automatically identify such cases using glitch potential. Fig. 3 shows two signals crossing clock domains, are synchronizing in the clk2 domain, and reconverging. The CDC solution can automatically identify such reconvergence issues. In Fig. 4, a crossing signal is diverges into two different synchronizer paths. Because of this, the signals can lose correlation, and problems can therefore occur in the downstream logic. The CDC solution can automatically identify such issues with divergence.

In Fig. 5, the signal crossing clock domains has a missing synchronizer. This can lead to loss of data because of metastability problems [10]. This system can automatically identify such problems.

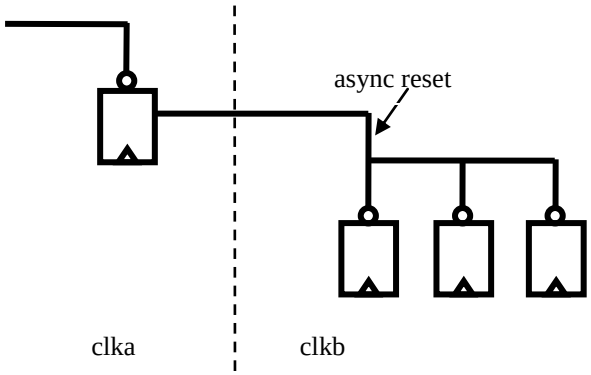

Figure 1. Missing reset synchronizer.

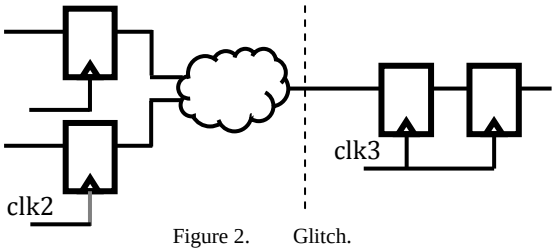

Figure 2. Glitch.

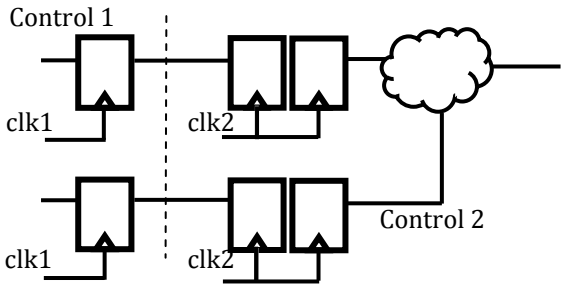

Figure 3. Reconvergence.

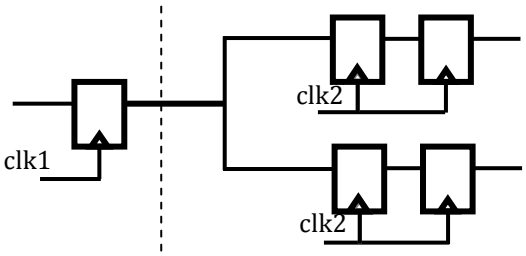

Figure 4. Signal losing correlation.

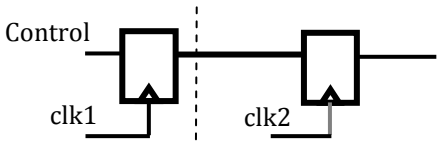

Figure 5. Loss of data.

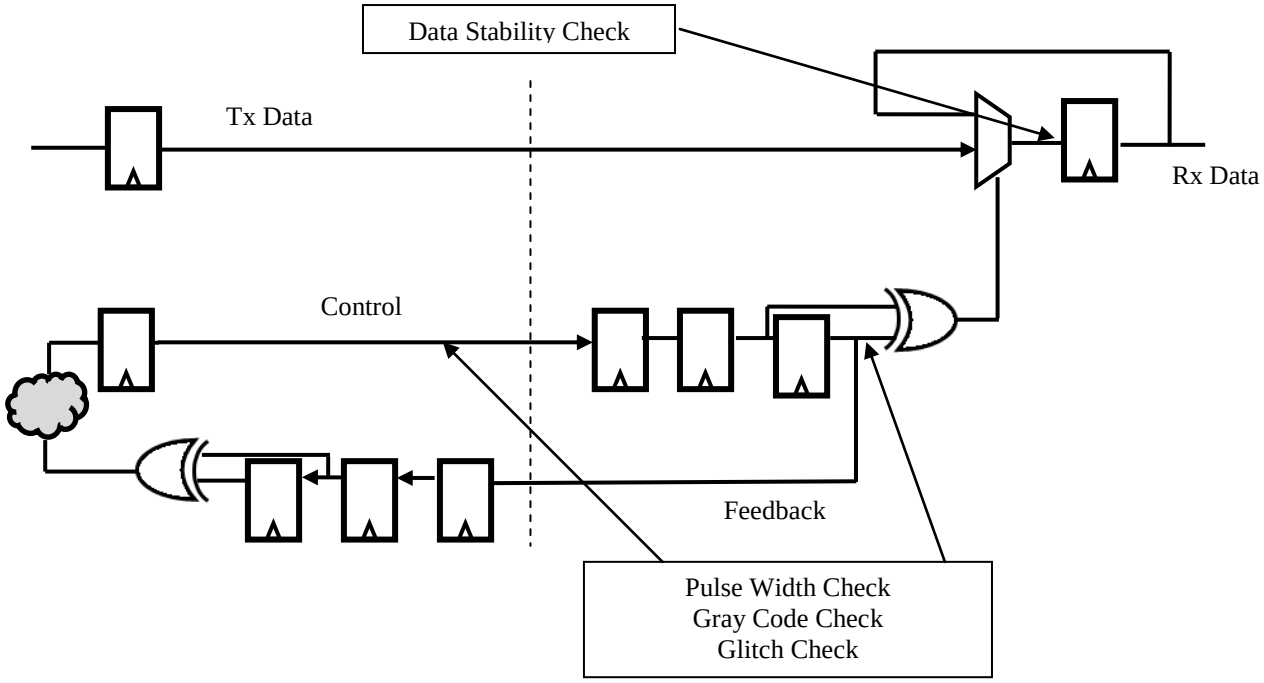

Figure 6. CDC handshake and formal analysis.

3) CDC interface/handshake identification

In Fig. 6, a simple CDC handshake interface is illustrated. For correct operation of the interface, the control signal needs to be synchronized. This synchronized signal needs to be associated with a data transfer from Tx Data to Rx Data. Moreover, a feedback signal needs to be generated in the Tx domain to indicate that the data transfer was performed. This CDC system will automatically identify all handshake structures. Thus, all types of synchronizer structures will be understood and identified.

B. Formal integration analysis

The integrated formal analysis exhaustively verifies the safety of crossings using an innovative metastability-aware formal engine, which ensures the safety of data transfer by verifying the underlying design principle, according to which the CDC data path must be a multi-cycle path. This addresses the root case of metastability problems, and can be applied to any form of data transfer protocol. The integrated formal solution leverages the automatic clock intent analysis result to produce a composite report, saving users time and effort [11]. Thus, this automatic and formal CDC solution simplifies the designer's sign-off task.

Fig. 6 illustrates a simple CDC interface, which consists of a data crossing that is controlled by a control and feedback signal. This system CDC will automatically identify these data and control crossings. For the control crossings, this system will perform formal analysis to verify the following:

1) Pulse width check

It verifies that the control pulses generated in the transmitter domain are of a sufficient duration to be captured in the receiver domain. If the pulse is missed, the pulse width check fails, and a value change dump (VCD) trace will be generated. Failing pulse width checks will identify problems with the pulse generation logic from fast-to-slow clock domains.

2) Gray code check

It verifies that the first-in first-out (FIFO) control signals are correctly gray coded. If gray encoding is incorrect, the gray code check will fail, and a VCD trace will be generated. Failing gray code checks identify problems with gray encoding.

3) Formal glitch check

It verifies that the glitch generated in the transmitter domain can be captured in the receiver domain. If the glitch is captured, the formal glitch check fails, and a VCD trace will be generated. Failing glitch checks identify problems with the combinatorial logic on asynchronous crossings.

4) Data stability check

It verifies that it is possible to launch Tx data and capture Rx data on the next receiving clock edge. If possible, the data stability check fails, and a VCD trace will be generated. Failing data stability checks identify problems with control and feedback logic.

C. *Dynamic CDC verification*

The dynamic CDC verification capability in this CDC system leverages existing simulation test-benches for CDC verification. Dynamic CDC verification injects the effects of metastability during the simulation to catch functional errors. It also includes monitors to check for gray code, pulse width, data stability, and glitch violations. This automates CDC sign-off and supports popular simulators.

As shown in Fig. 6, the control and feedback signals functionally control the data transfer from Tx data to Rx data. In regular simulation-based verification, the effects of metastability across asynchronous clock domains are not considered. However, during real chip operations, it is possible that a control signal transition can be missed in the receiver clock domain because of metastability, and the data transfer can therefore be corrupted. This problem cannot be identified with regular simulation-based verification.

Dynamic CDC verification will automatically inject metastability effects on the crossing signals. Users can include this metastability injection file in the top-level test-bench, and perform simulation-based verification with metastability models. In this way, problems with improper control logic can be found and debugged in the test-benches. Dynamic CDC verification will also include monitors for catching gray code, pulse width, data stability, and glitch violations in top-level test benches.

IV. CDC Analysis

CDC verification is conducted in the asynchronous FIFO memory design file "sync.v" by Verilog HDL. The specification of this design can be easily described. This design file has two clock domains "wclk" and "rclk." When a read enable signal "rd_ok" is asserted for more than one cycle, a read pointer signal "rd_ptr" is generated. Simultaneously, when a data read enable signal "rd_enb" is asserted, the read pointer signal "rd_ptr" will still be generated. The write pointer signal "wr_ptr" is used in the "rd_ptr" generation logic to detect the FIFO empty situation.

CDC verification was first carried out in this design, and the following situations were identified:

1. Two clock domains "wclk" and "rclk" were recognized.
2. All FIFO data were recognized by the category as W_DATA (data crossing with potential error)
3. The control signal controlling the transmission between the clock domains was not recognized.

The W_DATA indicates that it is a DATA crossing with warnings. The data association status was "none," which means there are no CTRL signals to handle the data transmission. As a result, an error in the clock name was determined in the always block, in where "rd_ptr" is generated. Therefore, it was corrected from "wclk" to "rclk," and we then reran this system.

The 2nd analysis result indicates that the "wr_ptr" information showed up in the W_DATA category. However, there were still no CTRL signals. Note that CTRL signals are usually required to control data transmission in the design. The "wr_ptr" information is used in the "rd_ptr" generation logic, which is generated in the "rd_clk" domain. The "wr_ptr" is generated in the "wr_clk" domain, whereas the "rd_ptr" is generated in the "rd_clk" domain. Therefore, "wr_ptr" should be considered as the CDC signals. However, no synchronizers were used in this "wr_ptr"; therefore "wr_ptr" was seen as W_DATA. We then added the double synchronizer for the "wr_ptr," as shown in Fig. 7, and reran this system.

Type	Signal	Receiving Flop	Clock Domains(From::To)	Location(File:Line)	Association
W_DATA:	fifo_4[0]	dout[0]	wclk_RI_W::rclk_RI_W	sync.v:160	None
W_DATA:	fifo_3[0]	dout[0]	wclk_RI_W::rclk_RI_W	sync.v:159	None
W_DATA:	fifo_5[0]	dout[0]	wclk_RI_W::rclk_RI_W	sync.v:161	None
W_DATA:	fifo_2[0]	dout[0]	wclk_RI_W::rclk_RI_W	sync.v:158	None
W_DATA:	fifo_0[0]	dout[0]	wclk_RI_W::rclk_RI_W	sync.v:162	None
W_DATA:	fifo_1[0]	dout[0]	wclk_RI_W::rclk_RI_W	sync.v:157	None
W_DATA:	fifo_0[0]	dout[0]	wclk_RI_W::rclk_RI_W	sync.v:156	None

Figure 7. Analysis result from the 1st run.

Type	Signal	Receiving Flop	Clock Domains(From::To)	Location(File:Line)
CNTL:	wr_ptr[0]	wr_ptr_s1[0]	wclk_RI_W::rclk_RI_W	sync.v:120
CNTL:	wr_ptr[1]	wr_ptr_s1[1]	wclk_RI_W::rclk_RI_W	sync.v:120
CNTL:	wr_ptr[2]	wr_ptr_s1[2]	wclk_RI_W::rclk_RI_W	sync.v:120

Figure 8. Analysis result from the 2nd run.

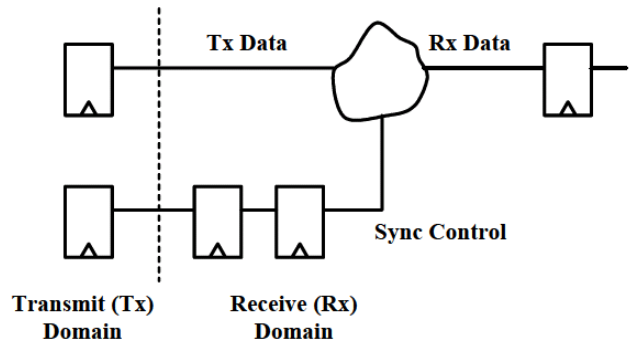

Figure 9. Structure of the control signal is used for data transfer in the received (RX) domain.

W_ASYNC_RST_FLOPS - Flops with resets from asynchronous clock domain (Total: 3)

Type	Reset	Flop	Location(File:Line)	Clock Domains(From::To)
W_ASYNC_RST_FLOPS:	rstn	rd_ptr[2:0]	sync.v:133	wclk_RI_W::rclk_RI_W
W_ASYNC_RST_FLOPS:	rstn	rd_enb	sync.v:134	wclk_RI_W::rclk_RI_W
W_ASYNC_RST_FLOPS:	rstn	dout[3:0]	sync.v:160	wclk_RI_W::rclk_RI_W

Figure 10. W_ASYNC_RST_FLOPS warning.

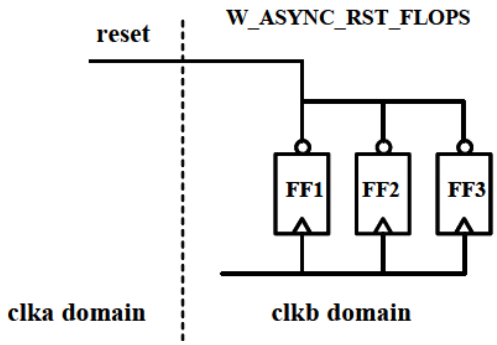

Figure 11. Flops with reset from asynchronous domains.

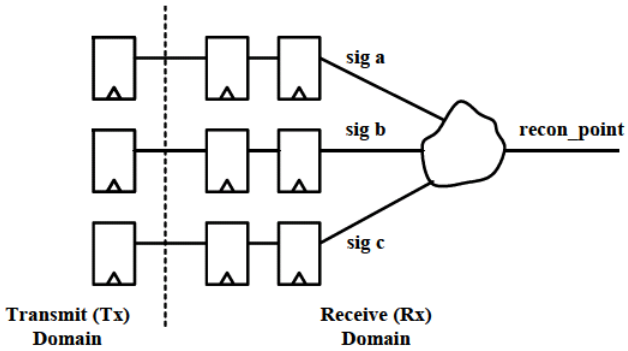

Figure 12. Example of reconvergence issue.

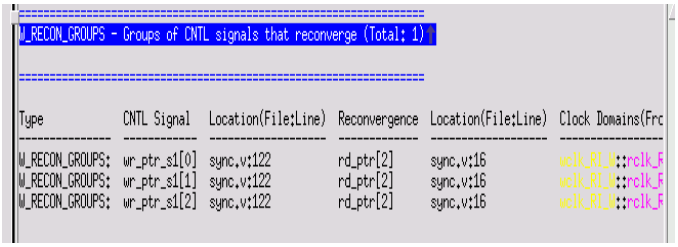

Figure 13. Reconvergence issue in this design.

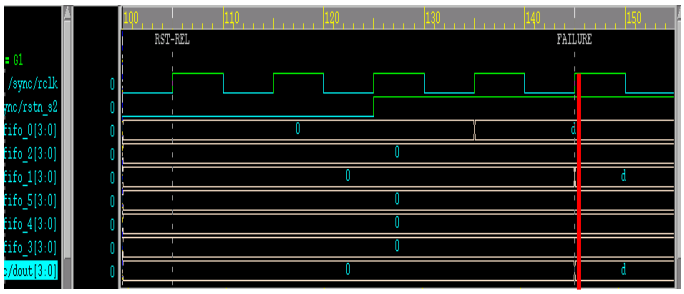

Figure 14. Example of data stability issue.

The signal "wr_ptr" is identified as a CTRL signal as shown in Fig. 8. Also, the data association status is now Load-Control, which means that the "wr_ptr" is a type of CTRL signals that handles data transmission, as shown in Fig. 9.

This system has the capability to detect the issues related to reset signals. In this design, this system showed a W_ASYNC_RST_FLOPS warning, which means that the flops are reset by a signal from the asynchronous clock domain, as shown in Figs. 10 and Fig. 11.

Then, we added the reset synchronizer in the design as shown in Fig. 12, and reran this system. The W_ASYNC_RST_FLOPS did not show up in this new analysis result. This system also detected the reconvergence issue, as shown in Fig. 13.

This system identifies the issue as a W_RECON_GROUPS warning. In this design, the write pointers "wr_ptr" have this warning since it is a multibit control signal, but in this case, the warning can safely be ignored as long as the write pointers is correctly gray coded.

In addition to these structural analyses, we also ran this formal analysis system and found the data stability issue, as shown in Fig. 14. The red line in Fig. 14 accurately indicates the point of "FAILURE."

This system performs data stability checks to ensure that the transmitted data are not captured at the next clock edge in the receiving clock domain. In this design, this failure indicates that the data were read out too early.

In summary, we identified the following issues during this CDC system analysis:

1) Incorrect clk specification in the "rd_ptr" logic.

2) Missing "wr_ptr" synchronizer clock for domain crossing.

3) Missing reset synchronizer.

4) Data stability issue on FIFO data.

We now describe the benefit of using the CDC solution. First, we obtained the result of comparing the design effort with a traditional design methodology and a design methodology that uses the CDC solution tool.

The analysis results revealed the mistake where the clock was always used in the block in which "rd_ptr" is generated. It should have been used in the block "rclk," instead of "wclk." Next, the signal "wr_ptr" was generated as the control signal in the "wclk" domain, but in this case, it is recognized as the data signal. Therefore, the signal "wr_ptr" was corrected so that it may be recognized as a control signal using a synchronizer. The control signals show the structure used for data transfer by the receiver (read side) in Fig. 9. Also, this design uses a reset signal "as is" from a different clock domain. Therefore, the reset was set to always use a synchronized reset that generates "rd_ptr," "rd_enb" and "dout" signals.

V. Experimental Results

CDC verification was carried out in the asynchronous FIFO memory design whose specification can be easily described. The FIFO memory designs consist of two clock domains, namely "wclk" and "rclk." When the "rd_ok" signal is asserted for more than one cycle, a read pointer "rd_ptr" signal is generated. At the same time, when the data read enable signal "rd_enb" is asserted, the read pointer "rd_ptr" continues to be generated. The write pointer "wr_ptr" is used in the "rd_ptr" generation logic to detect the FIFO empty situation [12].

```
…
always@(posedge rclk or negedge r_rstn2)
        begin
                if (~r_rstn2)
                  begin
                  rd_ptr <= 3'b000;
            rd_enb <= 1'b0;
            end
           else if (wr_ptr == rd_ptr)
```

Figure 15. Description with no synchronizer.

```
always@(posedge rclk or negedge r_rstn2)
 begin
      if ( ~r_rstn2 ) begin
            wr_ptr1 <= 3'b000;
            wr_ptr2 <= 3'b000;
          end
      else
          begin
            wr_ptr1 <= wr_ptr;
            wr_ptr2 <= wr_ptr1;
          end
 end

…

always@(posedge rclk or negedge r_rstn2
        begin
                if (~r_rstn2)
                  begin
                    rd_ptr <= 3'b000;
                    rd_enb <= 1'b0;
                  end
  else if (wr_ptr2 == rd_ptr)
```

Figure 16. The double flip-flop synchronizer description.

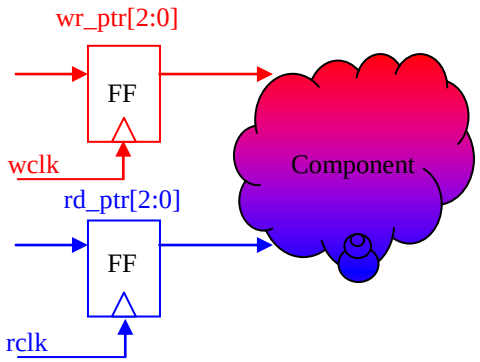

Figure 17. Circuit diagram with no synchronizer.

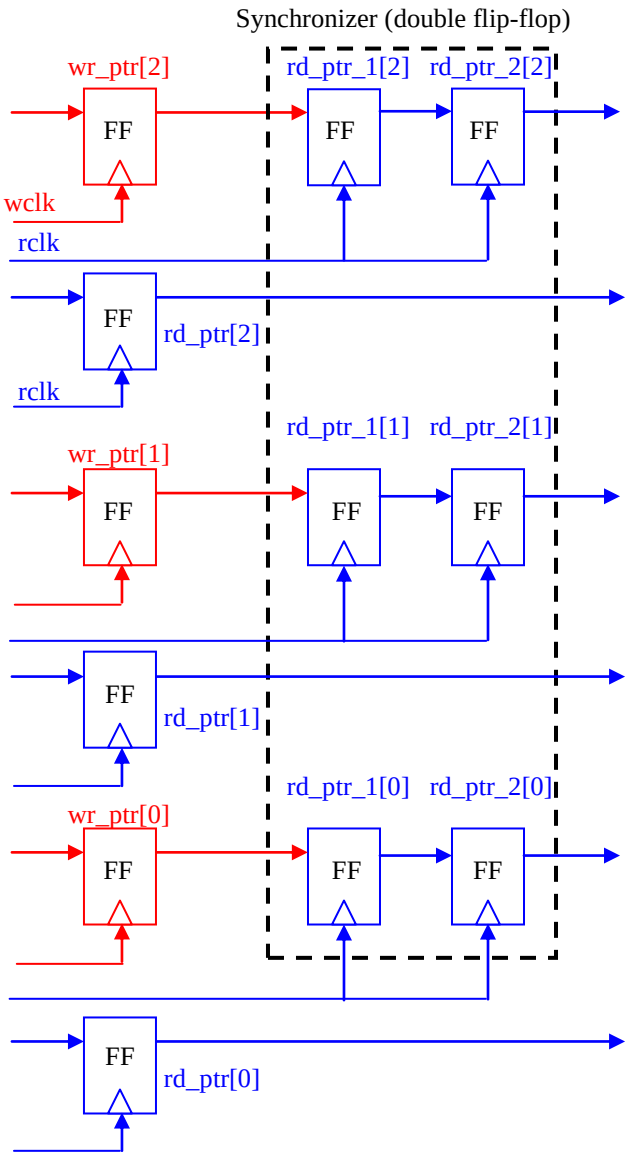

Figure 18. Circuit diagram of double flip-flop synchronizer circuit.

TABLE I. CIRCUIT EVALUATION "WCLK" = 60MH

rclk (MHz)	wclk (MHz)	Max Freq (MHz)	Power (mW)
60	60	113.3	67.3
70	60	114.3	68.2
80	60	102.7	68.6
90	60	105.2	69.2
100	60	108.0	71.3
110	60	114.0	72.2
120	60	118.2	72.8
130	60	120.5	73.0
140	60	128.1	74.0

TABLE II. CIRCUIT EVALUATION "WCLK" = 75MHZ

rclk (MHz)	wclk (MHz)	Max Freq (MHz)	Power (mW)
75	75	110.2	58.2
80	75	112.2	58.1
90	75	115.0	57.9
100	75	105.3	58.0
110	75	112.2	57.9
120	75	121.4	57.8

In this design, the CDC verification was first carried out and the two clock domains were identified as "wclk" and "rclk." No synchronizer existed between these domains, as shown in Fig. 15. Therefore, the synchronizers were inserted into these circuits to counter metastability, as shown in Fig. 16. For simplicity, these are described using a concrete circuit diagram. The circuit diagram of the asynchronous transfer was not properly synchronized, as shown in Fig. 17. For these measures, the synchronizers were inserted into the "wr_prt" signal with "rclk," as shown in Fig. 18. The three double flip-flops synchronizers were eventually inserted into these circuits, shown by a dashed line surrounding the three synchronizers in Fig. 18.

The clock frequency combination of "rclk" and "wclk" plays a significant role in the CDC solution. Therefore, a variable clock frequency was evaluated for the design in which the synchronizer was inserted for CDC issue measure.

Table 1 shows the combination of "rclk" and "wclk." The "wclk" frequency was fixed to 60MHz and the "rclk" frequency was changed from 60 MHz to 140 MHz. In the design in which the synchronizer was inserted, the power consumption decreased with the frequency, but the performance achieved a "dead point" when "wclk" was about 1.3 times the value of "rclk." In particular, the point was a combination of "rclk" = 80 MHz and "wclk" = 60 MHz. These results show that different combinations of clock frequency are effective in resolving the CDC issues [13].

Next, the "wclk" signals were fixed to 75 MHz, and the "rclk" signals were changed from 75 MHz to 120 MHz, as shown in Table 2. This is also the same as the results in Table 1, and the dead point was observed when "wclk" was about 1.3 times "rclk." In particular, the point was a combination of "rclk" = 100 MHz and "wclk" = 75 MHz.

The results of this experiment are described in the combination of "rclk" and "wclk," which is shown in the following equation [14].

$$f_{rclk} = (1 + \frac{1}{N}) * f_{wclk} \quad (1)$$

Experimental results when N = 3 are shown in Fig. 18.

$$f_{rclk} = \frac{4}{3} * f_{wclk} \ . \quad (2)$$

When (2) is satisfied, the dead point affects the performance. This can be verified from Tables 1 and 2.

VI. CONCLUSION AND FUTURE WORK

There are complex issues regarding multiple clock domains and operating clock frequency, which complicate the CDC solution. Clock frequency is considered when dealing with CDC issues and it is inevitable for automation to be considered in CDC solutions. This case study shows that automated verification and debugging using the CDC solution are very effective. Moreover, it clarifies the analysis methodology in the design of multiple clock domains. Debugging was advanced while changing design descriptions based on an analysis report of the CDC solution beginning at the original source cord in this case study.

In addition, CDC issues require attention to be paid to different combinations of clock frequency. It is thought that it can lead to improvements in the performance (clock cycle) of the synchronous circuit design for the CDC issue measure. In the future, we intend to analyze many more clock domains in order to confirm the commonly used methodologies.

ACKNOWLEDGMENT

This work was supported in part by Kakihara Science and Technology Foundation. The authors would like to thank Dr. Jin Zhang at EVE-USA Inc. who has shared her valuable comments.

REFERENCES

[1] Y. Feng, Z. Zhou, D. Tong, and X. Cheng, "Clock domain crossing fault model and coverage metric for validation of SoC design," in Proc. Design, Automation and Test in Europe (DATE07), pp. 1-6, 2007.

[2] M. Su, Y. Chen and X. Gao, "A general method to make multi-clock system deterministic", in Proc. Design, Automation and Test in Europe (DATE10), pp. 1480-1485, 2010.

[3] R. Ginosar, "Fourteen ways to fool your synchronizer," in Proc. International Synposium Asynchronous Circuits and Systems, pp. 89-96, 2003.

[4] D. Kim, M. Ciesielski, K. Shim and S. Yang, "Temporal parallel simulation: A fast gate-level HDL simulation using higher level models" in Proc. Design, Automation and Test in Europe (DATE11), pp. 1584-1589, 2011.

[5] S. Sarwary, and S. Verma, "Critical clock-domain-crossing bugs," Electronics Design, Strategy, News, pp. 55-60, 2008.

[6] A. Chakraborty and M. R. Greenstreet, "Efficient self-timed interfaces for crossing clock domains," International Symposium on Advanced Research in Asynchrnous Circuits and Systems, pp.78-88, 2003.

[7] J. Kessels, A. Peeters, and S. Kim, "Bridging clock domains by synchronizing the mice in the mousetrap," in Proc. International Workshop on Power and Timing Modeling, Optimization, and Simulation, pp.141-150, 2003.

[8] C. Kwok, V. Gupta, and T. Ly, "Using assertion-based verification to verify clock domain crossing signals," in Proc. Design and Verification Conference, pp. 654-659, 2003.

[9] T. Kapschitz and R. Ginosar, "Formal verification of syncronizers," in Correct Hardware Design and Verification Methods, Vol. 3725, pp. 359-362, 2005.

[10] R. Ginosar, "Metastability and synchronizers: A tutorial," IEEE Trans. on Design & Test of Computers, Vol. 28, No. 5, pp. 23-35, 2011.

[11] B. Keng, S. Safarpour, and A. Veneris, "Automated debugging of SystemVerilog assertions" in Proc. Design, Automation and Test in Europe (DATE11), pp. 323-328, 2011.

[12] A. Matsuda and J. Zhang, "Debugging methodology and timing analysis in CDC solution," in Proc. the IEEE International Conference on ASIC, Vol.1, pp. 393-396, 2011.

[13] A. Matsuda and S. Baba, "A debug solution with synchronizer for CDC," in Proc. the Workshop on Synthesis And System Integration of Mixed Information Technologies (SASIMI2012), pp. 390-393, 2012.

[14] A. Matsuda and S. Baba, "A clock frequency characteristics in across clock domains," in Proc. International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC2012), C-T1-01, 2012.