Eliminating Data Invalidation in Debugging Multiple-Clock Chips

Jianliang Gao*†, Yinhe Han* and Xiaowei Li*
* Key Laboratory of Computer System and Architecture, Institute of Computing Technology, Chinese Academy of Sciences
† School of Information Science and Engineering, Central South University

***Abstract*—A critical concern for post-silicon debug is the need to control the chip at clock cycle level. In a single clock chip, run-stop control can be implemented by gating the clock signal using a stop signal. However, data invalidation might occur when it comes to multiple-clock chips. In this paper, we analyze the possible data invalidation, including data repetition and data loss, when stopping and resuming a multiple-clock chip. Furthermore, we propose an efficient solution to eliminate data repetition and data loss. Theoretical analysis and simulation experiments are both conducted for the proposed solution. We implement the proposed Design-for-Debug (DfD) circuit with SMIC 0.18μm technology and simulate the data transfer across clock domains using SPICE tool. The results show that both data repetition and data loss can be avoided with the proposed solution, even if metastability occurs.**

I. Introduction

Before an integrated circuit (IC) is manufactured, pre-silicon verification techniques are used to kill functional bugs in the circuit. For the increasing system complexity, existing pre-silicon techniques such as simulation and formal verification cannot guarantee that first silicon will be bug-free [1]. Moreover, electrical bugs are becoming more serious as the decrease of feature size [2]. Whereas, pre-silicon methods don't address many deep-submicron electrical bugs that occur in the actual devices [3]. Therefore, post-silicon debug is becoming more important and the most time-consuming part, 35% on average, of the development cycle of a new chip [4]. Considering the increasing cost, it is imperative to identify the bugs that remain in the chip as soon as the first silicon is available.

One of the biggest challenges in post-silicon is the observability of the internal signals. It is not practical to observe all the internal signals in realtime [8]. Run-stop control provides a convenient approach to freeze the chips for offline observation. For a stopped system, states can be unloaded for both scan-based debug and trace buffer-based debug. In scan-based debug, a structural method can be used to access the manufacturing-test scan chains from test-access-port (TAP) [14]. In trace buffer-based debug, the contents saved in trace buffer can be dumped out for offline analysis. Afterwards, the execution is resumed (continue to run). The resume operation is indispensable for debugging, especially under non-deterministic condition.

The work was supported in part by National Natural Science Foundation of China (NSFC) under grant No.(60806014, 60831160526, 60633060, 60921002, 60906018, 61076037), in part by National Basic Research Program of China (973) under grant No. 2011CB302503, and in part by Hi-Tech Research and Development Program of China (863) under grant No. 2009AA01Z126.

978-3-9810801-7-9/DATE11/©2011 EDAA

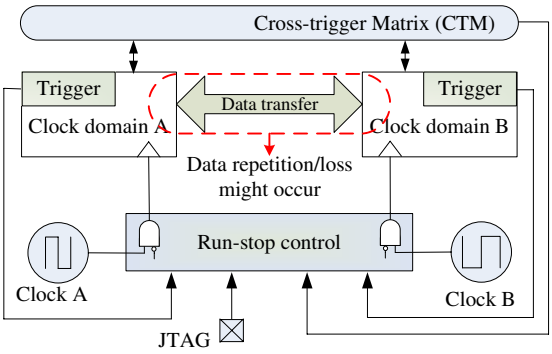
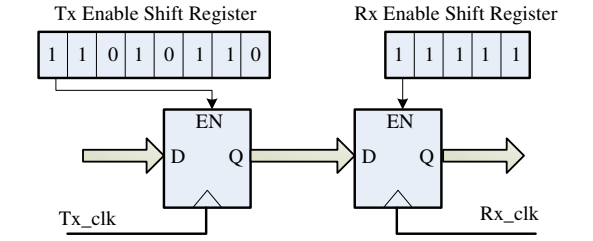


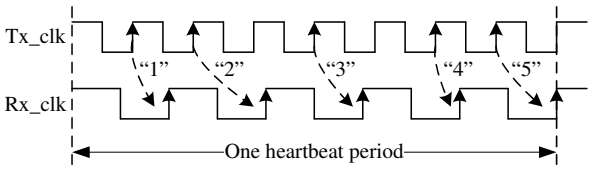
Fig. 1: Run-stop control might result in data repetition/loss

The stop and resume operations can be implemented by the run-stop control module on the chip. As shown in Fig. 1, run-stop control module receives commands from the IEEE 1149.1 (JTAG) port [15] or the on-chip trigger/cross-trigger modules (e.g., [9][10][11]). The stop signal is subsequently generated to control the clock signals. In a single clock chip, it is feasible to gate the clock signal using the stop signal directly. However, in multiple-clock chips, non-integral clock signal might be produced due to the asynchronous stop signal. Synchronizer is used to avoid the unexpected results. Whereas, data invalidation is likely to happen for the introduction of the synchronizer in debugging multiple-clock chips. The data invalidation includes data repetition (data is received more than once) and data loss (data is sent, but failed to be received). Goel and Vermeulen presented the problem of data repetition when stopping (called data invalidation in [6]) for the first time. To the best knowledge of the authors, data loss in debugging multiple-clock chips has not been discussed in public paper. More importantly, data repetition and data loss must be avoided to support the resume operation in debugging.

To address this problem, we first analyze the possibility and conditions for data repetition and data loss in multiple-clock chips debugging. Then, we propose a Design-for-Debug (DfD) circuit to avoid data repetition and data loss, when both stopping and resuming. The proposed DfD circuit includes some control logic and three main modules, i.e. the configurable synchronizer, the Relative Order Recovery (ROR) module, and



(a) data transfer logic



(b) Timing graph of sending clock and receiving clock

Fig. 2: Data transfer across clock domains [12]

the First-stop First-resume (FSFR) module. The configurable synchronizer ensures the correctness of stopping. The three modules together ensure the correctness of resuming. The control logic switches the authority of controlling the clock signal between them. The main contributions of this paper are:

- We present the problem of data loss for the first time, and extend the problem of data repetition on existing work.
- We draw the conditions for the occurring of data loss when stopping and present the theoretical analysis to avoid both data repetition and data loss.
- We propose a DfD circuit to avoid data repetition and data loss, for stopping and resuming in debugging multiple-clock chips.

II. Background

A. Data Transfer across Clock Domains

Multiple clock domain designs offer increased ease of functional-block reuse, simplified timing closure, and power advantages. Today's complex chips usually contain multiple clock domains. There are two main strategies to address the problem of reliable data transfer between independent clock domains, named one-way mode and burst mode [16]. In one-way mode, one bit of data is transferred at each handshaking (for the simplicity of explanation, assume the bandwidth of data transfer is one). For the handshaking signals are needed to be synchronized via special mechanism in multiple clock domains, it is time consuming for one-way mode [13]. A burst mode transfers multiple bits of data at one handshaking.

The interface between independent clock domains may implemented with first-in first-out buffer [5]. However, the latency introduced by buffer is a serious problem in latency-sensitive applications. To meet the latency requirement, the data transfer across clock domains can be implemented by communicating directly according the predefined scheme.

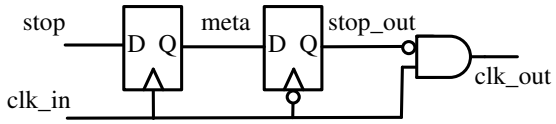
Fig. 2 shows the process of data transfer across clock domains directly. Tx_clk and Rx_clk represent the sending clock and the receiving clock respectively. The ratio of the frequency of Tx_clk to the frequency of Rx_clk is is $8:5$. In one heartbeat period, there are eight Tx_clk cycles and five Rx_clk cycles. The Tx and Rx enable shift registers determine whether the cycle can send or receive data or not. The shift registers circulates once during one heartbeat period. As shown in Fig. 2b, five bits of data can be transferred successfully in one heartbeat period. The direct data transfer improves communication efficiency, but data invalidation might happen during debugging for the lack of buffer.

B. Debugging Multiple-Clock Chips

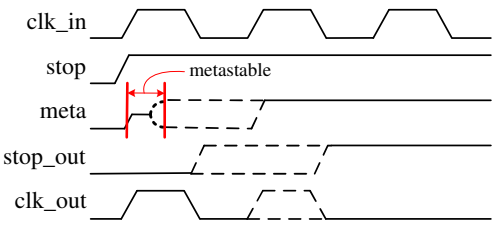
To address the problem of limited observability in post-silicon debug, a number of DfD solutions were proposed [7]. These solutions can be categorized as scan-based and trace buffer-based techniques [8]. Run-stop control is needed in both scan-based and trace buffer-based techniques. Scan-based technique utilizes internal scan chains to capture and off-load the internal states in stop mode [1]. Trace buffer-based technique acquires data in realtime, but the on-chip buffer resource is limited. When the buffer is full, it needs to stop the chip for shifting out the saved data. After the states are examined, the execution is resumed.

The stop signal is used to implement the stop and resume operations in run-stop control. Since the stop signal is asynchronous in multiple clock domains, non-integral clock signal might be produced if gating it with the stop signal directly. For example, if the stop signal comes when the clock signal is transforming from logic "0" to "1", metastability is likely to occur. To avoid unexpected results, synchronizer is commonly used in multiple clock domains. Fig. 3a shows a common used two-flop synchronizer. As shown in Fig. 3b, the clock signal can be gated correctly even when metastability occurs. However, the delay of the stop signal due to the synchronizer contributes to the data invalidation in debugging.

In the public domain, to the best of our knowledge, the only work discussed data invalidation of multiple-clock chip debugging is presented by Goel and Vermeulen [6]. They analyzed the phenomenon that the receiving clock domains

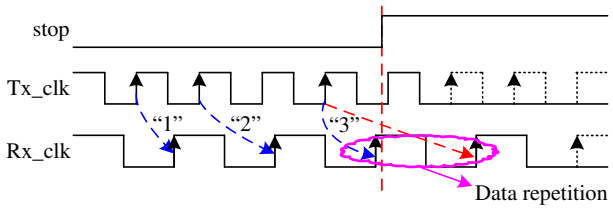


(a) Stop controller with a two-flop synchronizer

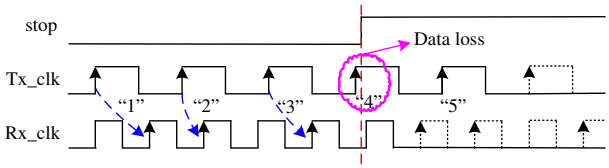


(b) Timing graph under metastability

Fig. 3: A common used synchronizer for the stop signal

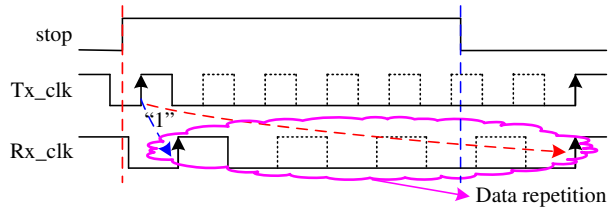


(a) Data repetition when stopping

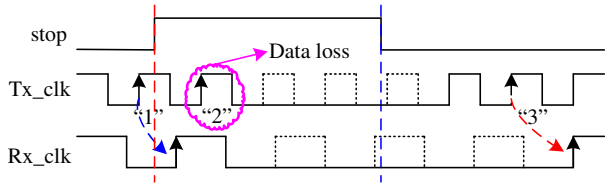


(b) Data loss when stopping

Fig. 4: Timing diagram of data invalidation when stopping



(a) Data repetition when resuming



(b) Data loss when resuming

Fig. 5: Timing diagram of data invalidation when resuming

use data from the clock domains that have already stopped. Furthermore, they proposed an efficient detector to find out the potential data invalidation. With the knowledge of the possible invalid states, the comparison between the invalid data and simulation data is avoidable. However, data invalidation when resuming has not been discussed in public paper. More serious, the execution can not be continued correctly once data invalidation occurs. Whereas, it is a basic operation to aid debugging, especially for non-deterministic bugs.

III. Data Repetition and Data Loss

For the chip with multiple clock domains, it is not practical to stop or resume at the same point in time. If the clock domains are not stopped or resumed at the same point in time, it is likely that the flip-flops in receiving clock domain capture the unexpected data. This phenomenon can be divided into data repetition and data loss. The phenomenon that the data is received repeatedly by the receiver is called ***data repetition***. The phenomenon of data repetition when stopping has been explored in [6]. We find that data repetition might occur not only when stopping, but also when resuming. In addition to this extension, we find another possible data invalidation, i.e., data loss. The phenomenon that the data is sent, but failed to be received is called ***data loss***. In the following, we explore these phenomenons and their occurring conditions.

A. Data Repetition and Data Loss When Stopping

When stopping a multiple-clock chip, the receiving clock domain might capture data from the stopped clock domain. Fig. 4a illustrates an example of data repetition when stopping. The upward arrowheads denote that the clock cycles are valid for sending or receiving data. Since the receiving clock (Rx_clk) has been stopped, the last valid cycle of the sending clock (Tx_clk) captures the data which has been captured by previous receiving cycle, i.e, data repetition occurs. In this example, the ratio of the frequencies of the sending and receiving clocks is 8:5. One necessary condition of data repetition when stopping has been discussed in [6]. Let f_s and f_r represent the frequencies of the sending and the receiving clock domain respectively. If data repetition occurs when stopping, then $f_s > f_r$.

Besides data repetition, the stop operation might cause data loss. Fig. 4b shows an example of data loss when stopping. In this example, f_s:f_r=5:8. The receiving clock is stopped before it captures Data "4". However, Data "5" is issued before the sending clock is stopped. So Data "5" overwrites Data "4" in the sending flip-flop. Thus, even if the receiving clock domain resumes firstly, it captures Data "5", instead of Data "4". Data "4" can not be received by the receiving clock domain. We draw one necessary condition of data loss when stopping as:

Theorem 1. *If data loss occurs when stopping, then* $f_s < f_r$.

Proof: If data loss occurs when stopping, the stop signal is asserted between the last two valid sending rising edge. It means that there is a valid rising edge of receiving clock between the last two valid sending rising edges. Since the last two issued data have not been captured, so the rising edge of receiving clock that captures the penultimate issued data is disabled. The rising edge of receiving clock that captures the penultimate issued data must be earlier than the last valid rising edge of sending clock to ensure the correctness. Therefore, there are two rising edges between the last two valid sending rising edges, so $f_s < f_r$. ■

B. Data Repetition and Data Loss When Resuming

Even if no data invalidation occurs when stopping, data repetition and data loss might both occur when resuming. Different from the stopping scenario, they might occur under three frequency relations, i.e. $f_s < f_r$, $f_s = f_r$ and $f_s > f_r$. Fig. 5 illustrates an example of data invalidation with

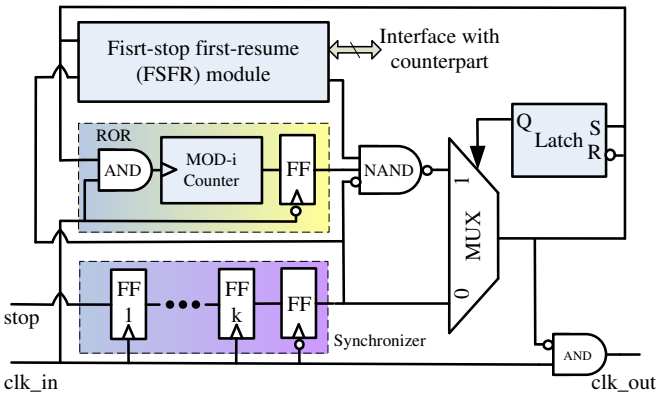


Fig. 6: The proposed design for debug

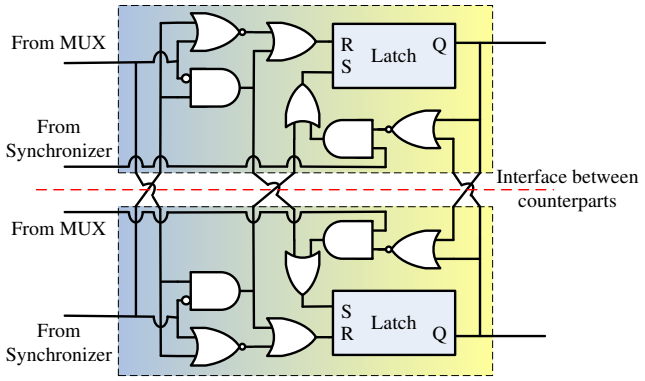


Fig. 7: The FSFR module

f_s:f_r=8:5. It can be seen from the example that not only data invalidation occurs, but also the relation of the sending and receiving clocks has been changed. For example, in Fig. 5a, the first stopped rising edge of Tx_clk is earlier than that of Rx_clk. However, the first resumed rising edge of Tx_clk is at the same time with that of Rx_clk. As discussed earlier, the predefined scheme is determined according to the requirement of timing. It is likely that the requirement of timing will not be met if the sending and receiving orders in a heartbeat period has been changed.

IV. Eliminating Data Repetition and Data Loss

A. Overview the Proposed Solution

To avoid data repetition and data loss, two requirements need to be met: (1) no data repetition and data loss when stopping; (2) the relation between sending clock and receiving clock is recovered completely. We propose a DfD circuit with little area cost to meet these requirements. As shown in Fig. 6, the proposed DfD circuit includes mainly three components, i.e. configurable synchronizer, Relative-Order-Recovery (ROR) module and First-Stop First-Resume (FSFR) module. The configurable synchronizer guarantees no data repetition and data loss when stopping. The three components together guarantee the recovery of the relation between clock domains. A multiplexer (MUX) is used to select which signal to control the clock signal. At the beginning, the latch is reset as logic "0". Once the synchronizer outputs the asserted stop signal, the control logic switches to the state of waiting for resuming. In the following, we detail the design and present the corresponding theoretical analysis.

B. Ensuring the Correctness of Stopping

To avoid data repetition and data loss when stopping, we design a configurable synchronizer to control the stop signal. Different from the existing synchronizers, the proposed synchronizer adopts a configurable number of flip-flops according to the ratio of f_s:f_r, instead of fixed number. The key problem is to determine the number of flip-flops. Theorem 2 shows the principle.

Theorem 2. *For any two communicating clock domains with f_s:f_r=m:n (m, n are relatively prime numbers), no data repetition/loss occurs when stopping, if the sending clock delays $\lceil \frac{m}{n} \rceil$ cycles and the receiving clock delays $\lceil \frac{n}{m} \rceil$ cycles.*

Proof: Let d_1, d_2 represent the time difference from the stop signal is asserted to the first rising edge of the sending and receiving clock domains respectively. T_1, T_2 represent the clock periods of the sending and receiving clock domains respectively. It can be drawn that $0 < d_1 \leq T_1$ and $0 < d_2 \leq T_2$.

First consider data repetition. From Section III-A, it can be obtained that $m > n$. Assuming that data repetition occurs, from that, it can be derived:

$$d_2 > d_1 + \lceil \frac{m}{n} \rceil * T_1 \tag{1}$$

For $d_2 \leq T_2$ and $T_2 = \frac{m}{n} * T1$, Formula 1 can be rewritten as $\frac{m}{n} * T1 > d_1 + \lceil \frac{m}{n} \rceil * T_1$. That is $d_1 < (\frac{m}{n} - \lceil \frac{m}{n} \rceil) * T_1 \leq 0$, which violates against $d_1 > 0$. So the hypothesis does not hold, i.e. no data repetition occurs.

Secondly, consider data loss. From Theorem 1, it can be deduced that $m < n$. So $d_1 \leq T_1 \Rightarrow d_1 < T_1 + d_2 < \lceil \frac{n}{m} \rceil * \frac{m}{n} * T_1 + d_2 \Rightarrow d_1 < \lceil \frac{n}{m} \rceil * T_2 + d_2$, which means the first invalid rising edge of the receiving clock domain is later than the last valid rising edge of the sending clock domain. So the last second issued data must has been captured by the receiving clock domain before stopped, i.e., no data loss occurs when stopping. ■

The proposed configurable synchronizer includes k positive-edge-triggered flip-flops and one negative-edge-triggered flip-flop. From Theorem 2, it can be known that k in Fig. 6 is determined by the the ratio of f_s:f_r. Assuming f_s:f_r=m:n, k is equal to $\lceil \frac{m}{n} \rceil$, $\lceil \frac{n}{m} \rceil$ for the sending clock domain and the receiving clock domain, respectively.

C. Ensuring the Correctness of Resuming

As analyzed in Section III-B, data repetition and data loss might occur when resuming the execution. Moreover, the sending and receiving relationship has been changed. Therefore, ensuring correctness of resuming the execution is to recover the relationship between the sending clock and the receiving

clock completely. It indicates two necessary requirements. One requirement is that the relative order of the first stopped cycle in a heartbeat period must be the same with that of the first resumed sending and receiving cycles. For example, in Fig. 5b, the relative order of the first stopped cycle of Tx_clk is the third, but the order of the first resumed cycle is the sixth. The other requirement is that the time difference between the first stopped sending and receiving cycles must be equal to that between the first resumed sending and receiving cycles. We use the ROR module and FSFR module to meet the requirements.

As shown in Fig. 6, the ROR module mainly contains an increasing *MOD-i* counter, which will immediately be reset to zero once it reaches *i*. Assuming f_s:f_r=m:n, *i* is equal to m and n in the sending and receiving clock domains respectively. The counter is driven by the input clock signal (clk_in) during the stop phase (logic AND between clk_in and the output of the configurable synchronizer). Once the counter becomes zero, the output of the counter becomes logic "1". The output of the counter is sampled by a negative-edge-triggered flip-flop. Note that non-integral clock period might be outputted if without using the negative-edge-triggered flip-flop.

The FSFR module guarantees that the relative order of the resumed sending and receiving clock domains is the same with that of the stopped clock domains. The FSFR modules of two communicating clock domains (called counterparts) are illustrated in Fig. 7. Each counterpart has a RS (Reset-Set) latch. The Q port of latch is connected with the three-input NAND gate, which means three conditions for resuming. Consider the FSFRs in two communicating clock domains, the function of first-stop first-run is implemented by controlling the states of the latches. The latches are reset as logic "0" when the chip initiates. When one clock domain stops firstly, it set its latch as logic "1". One clock domain can begin only if the value of its latch is logic "1". By this way, only the first stopped clock domain can resume first. When it resumes, it changes the counterpart's latch as "1". For both clock domains, their latches are set as "0" when they resume.

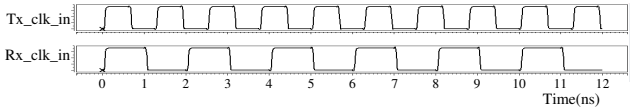
V. Simulation Results

A. Experiment Setup

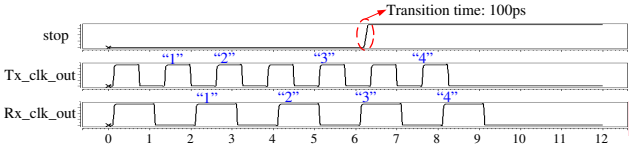
We construct a multiple-clock system including the proposed DfD using SMIC 0.18μm Standard Cell Library. The stop and resume operations are simulated using Synopsys HSPICE tool on the prototype with a supply voltage of 1.8v.

In the simulations, we set the sending frequency (f_s) and the receiving frequency (f_r) as 800MHz and 500MHz, respectively. Since f_s:f_r=8:5, we configure the number of the positive-edge-triggered flip-flops of the configurable synchronizer (k in Fig. 6) as two ($\lceil \frac{8}{5} \rceil$) in the sending clock domain and one ($\lceil \frac{5}{8} \rceil$) in the receiving clock domain. MOD-8 and MOD-5 counters are used in them respectively. The circular enable shift registers in the sending and receiving clock domain are set as "11010110" and "11111" respectively.

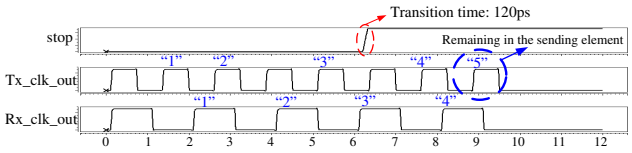
Due to space limitation, we only discuss the the scenario of $f_s > f_r$. Note that the same effectiveness of the proposed DfD circuit is aslo achieved for $f_s \leq f_r$.



(a) The input clock signals



(b) The output clock signals when metastability becomes high (logic "1")



(c) The output clock signals when metastability becomes low (logic "0")

Fig. 8: Waveforms for stopping with metastability occurring

B. Simulating the Stop

The objective of this experiment is to show the effectiveness of the proposed DfD on stopping operation. To validate the robustness, we tune carefully the stop signal and simulate the scenario of metastability in this experiment.

Fig. 8 shows the timing relationship between sending clock (Tx_clk) and receiving clock (Rx_clk) under the control of the stop signal. Fig. 8a shows the input clock signals. In this simulation, we let the stop signal begin to high at 6.2ns, which is so close to the rising edge of Tx_clk that metastability occurs. Fig. 8b shows the waveforms of the output of the clock signals when the transition time of the stop signal is set as 100ps. In this scenario, the metastability becomes logic "1" finally. Tx_clk is stopped before Rx_clk. It can be seen that four bits of data are sent and all the data are received. So no data repetition and data loss occur.

When the transition time of the stop signal is set as 120ns, the metastability becomes logic "0" finally. Fig. 8c shows the clock signals of this scenario. It can been seen that five bits of data are sent, four bits of data of which are received before Rx_clk is stopped. The last issued data, i.e. Data "5", remains in the sending flip-flop. The remaining data can be received rightly only if the stopped relationship of clocks is recovered when resuming. Therefore, no data invalidation and data loss occur.

C. Simulating the Resume

The preceding analysis and experiments show that data invalidation during stopping can be avoided by the proposed DfD circuit. So we need next simulate the resume to see whether the clock relationship can be recovered completely or not when resuming.

As show in Fig. 9, there are eight sending cycles and five receiving cycles in a heartbeat period (10ns). According to the transfer scheme introduced in Section V-A, five bits of data can

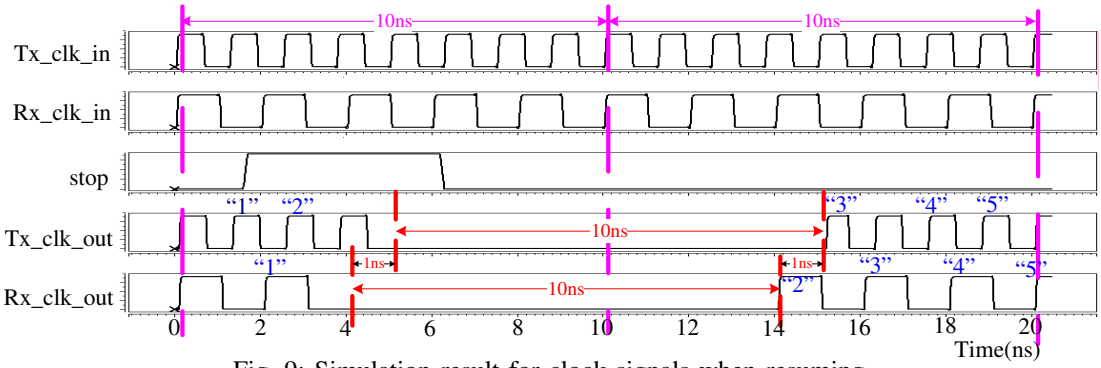


Fig. 9: Simulation result for clock signals when resuming

be transferred in a heartbeat period. The stop signal is asserted at 1.6ns, i.e. between the second sending cycle and the first receiving cycle. The first stopped sending and receiving cycles are the fourth and the second of the first heartbeat period. Two data bits are sent before the Tx_clk_out is stopped. The receiver samples one data-bit before being gated, i.e., Data "2" is left in the sending flip-flop.

The stop signal returns low (logic "0") at 6.1ns, i.e. the fifth sending cycle and the fourth receiving cycle. As it can be seen from Fig 9, the first resumed sending and receiving cycles are the fourth and the second of the second heartbeat period. Therefore, the relative order is recovered precisely. The first resumed receiving cycle is $1ns$ earlier than the first resumed sending cycle, which is the same with the first stopped cycles. Therefore, the phase relation is also recovered completely. According the transfer scheme, the receiving clock domain resumes to receive the Data "2" firstly, which is remained in the sending flip-flop. The proposed DfD ensures that both Tx_clk_out and Rx_clk_out keep inactive for integral multiples of the heartbeat period. After that, the clocks resume with the same relative relation. In this example, Tx_clk_out and Rx_clk_out keep inactive for $10ns$, i.e. one heartbeat period. Thus, the relationship between sending clock domain and receiving clock domain is maintained.

D. Required Silicon Area

To observe the area cost of the proposed DfD, we synthesize the DfD circuit with a $0.18\mu m$ SMIC CMOS technology. Considering the circuit in the sending clock domain of our experiment prototype, the total DfD area is $1127\mu m^2$ (about 113 equivalent two-input NAND gates). When compared to today's large design size, in general, the area cost to implement the proposed DfD is quite small.

Regarding timing, the DfD circuit is not in the critical path. The delay of clock signal is determined by one AND gate inserted for gating clock. Therefore, the DfD circuit does not introduce additional delay of clock signal.

VI. Conclusions

Debug the chip with multiple clock domains is a challenge task. In this paper, we analyze the conditions for the occurring of data repetition and data loss in depth. Then, we propose an efficient solution to eliminate data repetition and data loss in debugging multiple-clock chips. The theoretical analysis shows that proposed solution can avoid data invalidation in multiple-clock chip debug. Simulation results confirm the effectiveness of the proposed solution at low DfD cost.

References

[1] A. B. T. Hopkins and K. D. McDonald-Maier, "Debug support for complex systems on-chip: a review," IEE Proceedings Computers and Digital Techniques, vol. 153, pp. 197-207, 2006.

[2] Y. Chia-Chih, L. Ten, H. Lin, Y. Kai, L. Tayung, and H. Yu-Chin, "A general failure candidate ranking framework for silicon debug," in Proc. IEEE VLSI Test Symposium (VTS), 2008, pp. 352-358.

[3] P. Sung-Boem and S. Mitra, "IFRA: Instruction Footprint Recording and Analysis for post-silicon bug localization in processors," in Proc. ACM/IEEE Design Automation Conference (DAC), 2008, pp. 373-378.

[4] M. Abramovici, P. Bradley, K. Dwarakanath, P. Levin, G. Memmi, and D. Miller, "A reconfigurable design-for-debug infrastructure for SoCs," in Proc. ACM/IEEE Design Automation Conference, 2006, pp. 7-12.

[5] R. W. Apperson, Z. Yu, M. J. Meeuwsen, T. Mohsenin, and B. M. Baas, "A scalable dual-clock FIFO for data transfers between arbitrary and haltable clock domains," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 15, pp. 1125-1134, 2007.

[6] S. K. Goel and B. Vermeulen, "Hierarchical data invalidation analysis for scan-based debug on multiple-clock system chips," in Proc. International Test Conference (ITC), 2002, pp. 1103-1110.

[7] B. Vermeulen and S. K. Goel, "Design for debug: catching design errors in digital chips," IEEE Design & Test of Computers, vol.19, no.3, pp. 35-43, 2002.

[8] H. F. Ko and N. Nicolici, "Automated trace signals identification and state restoration for improving observability in post-silicon validation," in Proc. Design, Automation and Test in Europe Conference (DATE), 2008, pp. 1298-1303.

[9] ARM Ltd. CoreSight Architecture specification, 2004, ARM IHI 0029B.

[10] S. Tang and Q. Xu, "In-band cross-trigger event transmission for transaction-based debug," in Proc. Design, Automation and Test in Europe Conference (DATE), 2008, pp. 414-419.

[11] B. Vermeulen, M. Z. Urfianto, and S. K. Goel, "Automatic generation of breakpoint hardware for silicon debug," in Proc. IEEE/ACM Design Automation Conference (DAC), 2004, pp. 514-517.

[12] S. Balasubramanian, N. Natarajan, O. Franza and C. Gianos, "Deterministic low-latency data transfer across non-integral ratio clock domains," in Proc. the International Conference on VLSI Design (VLSID), 2006, pp.1063-1067.

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

[14] K. Holdbrook, S. Joshi, S. Mitra, J. Petolino, R. Raman, and M. Wong, "MicroSPARCTM: a case-study of scan based debug," in Proc. IEEE International Test Conference (ITC), 1994, pp. 70-75.

[15] IEEE JTAG 1149.1-2001 Std. IEEE standard test acess port and boundary-scan architecture, IEEE Computer Society, 2001.

[16] Open Core Protocol Specification. http://www.ocpip.org.