Clock Domain Crossing Fault Model and Coverage Metric for Validation of SoC Design

Yi Feng, Zheng Zhou, Dong Tong, Xu Cheng
Dept. of Computer Science, Peking University
Beijing, P.R.China
{fengyi, zhouzheng, tongdong, chengxu}@mprc.pku.edu.cn

Abstract

Multiple asynchronous clock domains have been increasingly employed in System-on-Chip (SoC) designs for different I/O interfaces. Functional validation is one of the most expensive tasks in the SoC design process. Simulation on register transfer level (RTL) is still the most widely used method. It is important to quantitatively measure the validation confidence and progress for clock domain crossing (CDC) designs. In this paper, we propose an efficient method for definition of CDC coverage, which can be used in RTL simulation for a multi-clock domain SoC design. First, we develop a CDC fault model to present the actual effect of metastability. Second, we use a temporal data flow graph (TDFG) to propagate the CDC faults to observable variables. Finally, CDC coverage is defined based on the CDC faults and their observability. Our experiments on a commercial IP demonstrate that this method is useful to find CDC errors early in the design cycles.

1. Introduction

As modern System-on-Chip (SoC) designs continue to face increasing size and complexity challenges, multiple asynchronous clock domains have been employed for different I/O interfaces. A CDC design is a design that has one clock asynchronous to, or has a variable phase relation with, another clock. A CDC signal is a signal latched by a flip-flop in one clock domain and sampled in another asynchronous clock domain. Transferring signals between asynchronous clock domains may lead to setup or hold timing violations of flip-flops. These violations may cause signals to be metastable [2]. Even if synchronizers could eliminate the metastability, incorrect use, such as convergence of synchronized signals or improper synchronization protocols, may also result in functional CDC errors [5].

Functional validation of such SoC designs is one of the most complex and expensive tasks. Simulation on register transfer level (RTL) is still the most widely used method. However, standard RTL simulation can not model the effect of metastability. As the CDC errors are not addressed and verified early in the design cycles, many designs exhibit functional errors only late in their design cycles or during post-silicon verification. Several coverage metrics are proposed to measure the validation's adequacy and progress [14], such as code based coverage, finite state machine coverage and functional coverage. Nevertheless, these coverage metrics do not have direct relations with CDC issues. Therefore, there is a need for coverage metric on CDC issues in standard RTL simulation flow.

In this paper, we propose an efficient method for definition of CDC coverage. First, we present a functional CDC fault model that actually corresponds to the effect of metastability. Second, a temporal data flow graph (TDFG) is derived from RTL description to connect the starting variables of CDC paths to observable variables of SoC design. Then we insert the CDC fault at each starting vertex in TDFG. As the CDC fault may propagate to observable variables according to the data dependence between vertices, we evaluate the state of CDC faults for every vertex in TDFG. Finally, the CDC coverage is defined based on the CDC faults and their observability. With this approach, we can calculate CDC coverage for a multi-clock domain SoC design in standard RTL simulation flow. For a given design, a set of test vectors with higher CDC coverage is more likely to detect CDC functional errors.

The rest of the paper is organized as follows. Section 2 presents related work addressing validation of SoC designs, verification of CDC issues and observability of coverage metrics. Section 3 describes the functional CDC fault model. The TDFG is proposed in Section 4. Section 5 defines the CDC coverage. A case study is presented in Section 6 and Section 7 concludes the paper.

2. Related Work

The state-of-art validation of a SoC design has used simulation techniques on a combination of constraint random

978-3-9810801-2-4/DATE07 © 2007 EDAA

and directed test vectors. Many coverage metrics have been proposed for complex designs [6, 8, 9]. Each coverage metric has an underlying fault model [11, 14]. These fault based metrics are inherited from software testing and hardware manufacturing testing. A simulation run would detect a fault if that fault causes the design to behave differently than it would without the fault.

Q.Zhang and I.G.Harris [15] have presented a timing fault model, the mis-timed event (MTE) fault, to model the functional errors introduced by timing violation. CDC signals may have timing violation. However, the MTE fault model has a loose connection with the effect of metastability in flip-flops.

T.Ly and N.Hand [10] have proposed a formal verification method to check CDC errors. Although the formal method is useful on small portions of a design [7], it is not suitable on large scale SoC designs. Furthermore, it does not consider the observability of CDC errors.

S.Devadas and K.Keutzer [1, 3, 4] have investigated the observability of fault models. They have proposed a positive and negative tag to represent assignment incorrect possibilities, and evaluated the observability of the tags by data flow graph (DFG). But the DFG they used do not distinguish combinational logic and sequential logic.

The functional validation in CDC designs is timing related. Metastability may happen only in certain clock cycles and all CDC protocols are dealing with temporal relations between CDC signals. To evaluate the observability of CDC faults, therefore, we need to consider the temporal relation in DFG. To the best of our knowledge, there are no previous approaches that describe functional CDC fault model, a DFG with temporal relation to evaluate the CDC faults observability and a coverage for CDC issues in RTL simulation flow.

3. Functional CDC Fault Model

A useful coverage metric contains two essential factors. One is fault model, the other is observability of the fault [4]. In this section, we first outline the background of metastability. Then we propose a functional CDC fault model which actually corresponds to the effect of metastability.

3.1. Metastability

Consider a one bit CDC signal in Figure 1. Signal *R1* is latched in FF1 by CLKA and propagate to FF2, which is sampled by CLKB. CLKA and CLKB are asynchronous. Since *R1* comes from a different clock domain, its value can change at any time with respect to CLKB. If the value of *R1* changes within FF2's setup or hold time, FF2 may assume a state between 0 and 1, which is called metastability. A metastable value will unpredictably settle to either 0 or 1.

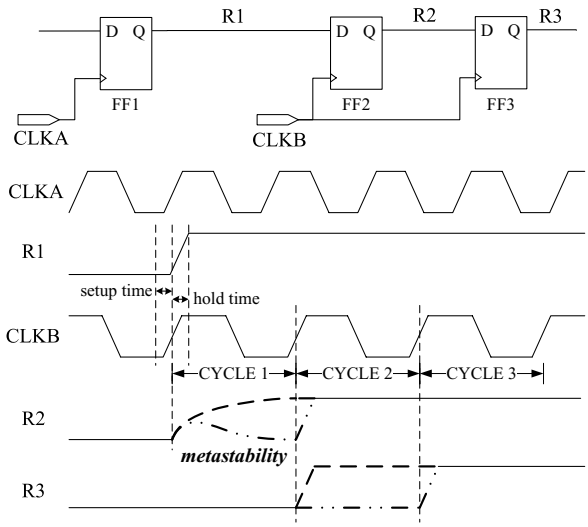

Figure 1. CDC Signal Example.

Synchronizer's reliability is expressed in terms of mean time between failures (MTBF). With a simple two flip-flops synchronizer, the MTBF is typically many eras [2].

3.2. CDC Fault Model

Even if metastability could be eliminated by synchronizer, however, the exact cycle when the flip-flop samples the value's changing is still unpredictable. A two flip-flops synchronizer is shown in Figure 1. *R1* is the input of FF2, and *R3* is the output of FF3. Although FF3 does not have metastability (according to the MTBF), *R3* may represent *R1*'s changing in either cycle2 or cycle3. That is the inherent reason for the functional errors in CDC designs.

Consider a CDC signal changes in a sampling clock cycle. It could happen during three periods of time: 1) the flip-flop's setup time; 2) the flip-flop's hold time; 3) the other time except for setup and hold time.

If a signal changes during setup or hold time, the flip-flop may or may not capture the value's changing. We can define five states for a flip-flop when samples a CDC signal:

Definition 1 (CDC State):

1. The signal changes during *setup time*; the flip-flop *samples* the changing.
2. The signal changes during *setup time*; the flip-flop does *not sample* the changing.
3. The signal changes during *hold time*; the flip-flop *samples* the changing.
4. The signal changes during *hold time*; the flip-flop does *not sample* the changing.
5. The signal changes during *other time* except for setup and hold time; the flip-flop *samples* the changing.

```
1  always @ (posedge HCLK)
2    begin
3      RdDMAH <= (RWCON & !RdDMAH);
4      WrDMAH <= (!RWCON & !WrDMAH);
5      if (WrCp & WrCPRun)
6        WrReqH <= !WrReqH;
7      if (RdCp & RdCPRun)
8        RdReqH <= !RdReqH;
9    end
10
11 always @ (posedge ICLK)
12   begin
13     WrReqI_Meta <= WrReqH;
14     RdReqI_Meta <= RdReqH;
15     WrReqI <= WrReqI_Meta;
16     RdReqI <= RdReqI_Meta;
17   end
18
19 assign XCS <= !WrDMAH & !RdDMAH & CS;
20
21 always @ (posedge ICLK)
22   NCS <= (WrReqI | RdReqI) & XCS;
```

Figure 2. A Verilog Example.

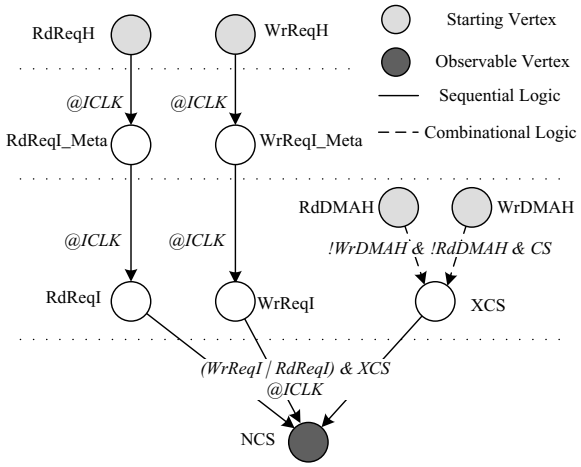

Figure 3. TDFG of Figure 2.

The *CDC state* can be defined for all CDC signals, not limited to the two flop-flops structure. In standard RTL simulation, every changing before clock edge of a flip-flop will be sampled and after clock edge will not be sampled. We define *CDC fault* for above CDC state 2 and 3. It is called *fault* because they behave differently than standard RTL simulation and designers' intention.

Definition 2 (CDC Fault):

CDC Setup Fault: A flip-flop is metastable due to *setup time* violation, it does *not sample* the value's changing.

CDC Hold Fault: A flip-flop is metastable due to *hold time* violation, it *samples* the value's changing.

In order to model the CDC setup and CDC hold faults, we insert *CDC_delay* and *CLK_jitter* modules into original RTL description. They add random delays for all CDC signals and jitters for asynchronous clocks. We also define *CDC_monitor* module to represent the state of CDC faults. Since metastability happens only in the clock cycle when a CDC signal changes, the state of CDC fault is active only in that cycle to indicate the CDC setup or hold fault.

4. Temporal Data Flow Graph

We add temporal relations between vertices into original DFG. In this section, a TDFG is first proposed to model CDC designs. Then the CDC fault is inserted to each CDC path. Finally we use the TDFG to propagate the CDC faults to observable variables.

4.1. Graph Structure

A TDFG is a directed graph $G(V,E,B,O)$, where:

- V is the set of vertices representing variables in RTL description;
- $E \subseteq V \times V$ is the set of edges connecting the vertices. Each edge $e(v,w) \in E$ is a directed edge from source vertex v to destination vertex w. In order to represent the temporal relation between variables, we distinguish two types of edges E_S and E_C, where $E_S \cup E_C = E$. If the logic between vertex v and w is *sequential logic*, then $e(v,w) \in E_S$. Otherwise, if the logic between v and w is *combinational logic*, then $e(v,w) \in E_C$. Each edge $e(v,w) \in E_C$ represents the combinational logic expression between v and w. Each edge $e(v,w) \in E_S$ represents both the combination and sequential logic condition between vertices. $Pre(v)$ is the set of predecessor vertices of v, that $\forall\, w \in Pre(v),\ \exists\, e(w,v) \in E$. Similarly, $Post(v)$ is the set of successor vertices of v, that $\forall\, w \in Post(v),\ \exists\, e(v,w) \in E$;
- $B \subset V$ stands for the set of starting vertices of G;
- $O \subset V$ stands for the set of ending vertices of G.

Let G_{CDC} denote TDFG for CDC design. $B \subset G_{CDC}$ is the set of starting variables of CDC paths, $O \subset G_{CDC}$ is the set of observable variables of CDC design. The observable variables are those that will be compared with a reference model described at a different abstraction level to check for certain behavior [14]. Variables on the paths between B and O make up of $V \subset G_{CDC}$.

Figure 3 shows the G_{CDC} of a Verilog example in Figure 2. Vertices *RdDMAH* and *WrDMAH* are starting variables of CDC paths from HCLK domain to ICLK domain. Edge *e(WrDMAH, XCS)* $\in E_C$ indicates the combinational logic from *WrDMAH* to *XCS*. Edge *e(RdReqI, NCS)* $\in E_S$ indicates the sequential logic from *RdReqI* to *NCS*. The edge *e(RdReqI, NCS)* $\in E_S$ contains two expressions: combinational logic expression *(WrReqI | RdReqI) & XCS* and sequential logic condition *@(posedge ICLK)*, which are derived from line 22 and 21 in Figure 2. Observable variable *NCS* $\in O$ is the design output.

4.2. CDC Fault Insertion

After creating the TDFG, we insert CDC faults for every starting vertex of G_{CDC} using *CDC_delay* and *CLK_jitter* modules. In order to model the unpredictable possibility of metastability, the random delay value for each starting vertex should be different. Delay insertion for vector type variable *reg[msb:lsb]* needs special consideration. For control vector, which may be used as command or status variable, different metastable possibility of each bit introduces different wrong value, and indicates different design status. In this case, each bit of control vector should have different delay value. For data vector, which may be used as FIFO pointer or transmission data in CDC design, we insert the same delay value for each bit to reduce complexity.

We monitor the state of CDC fault for all vertices during RTL simulation. For starting vertices in G_{CDC}, the CDC fault state is defined in definition 2. For other vertices, the CDC fault state is defined in Section 5.

4.3. CDC Fault Propagation

In simulation based validation environment, a discrepancy from desired behavior is detected only if an observable variable takes on a value that conflicts with the reference model. Therefore, we should evaluate whether the CDC faults would be propagated to the observable variables.

The propagation condition is evaluated on G_{CDC}. An active edge is an edge that can propagate the CDC fault state from source vertex to destination vertex. Consider an edge $e(v,w) \in E_C$. The propagation depends on the combinational logic expression between v and w when the CDC fault state of vertex v is active. If the edge $e(v,w) \in E_S$, besides the combination, the propagation also depends on the sequential logic condition. After propagation, the active cycle of vertex w's CDC fault state should be re-calculated. We discuss the CDC fault propagation under combinational logic expression and sequential logic condition separately.

1. Combinational logic expression between vertices:

The CDC fault propagation under combinational logic is similar to the tag simulation calculus described in [3].

- Logic gates:

Boolean expression is composed of three types of basic Boolean logic gate: one-input *INVERTOR*, two-input *AND* and two-input *OR* gate. For *INVERTOR* gate, the CDC fault state will propagate from input to output directly. For *AND* gate, the edge from one input to output is active when the other input is logic *1*. For *OR* gate, the edge from one input to output is active when the other input is logic *0*.

- Arithmetic operators:

Consider an expression $v(F) = v(A) <op> v(B)$. Two vectors A and B are operands for the arithmetic operator *op*, vector F is the result of computing. Let $v(A_{ORI})$ denote the original value and $v(A_{CDC})$ denote the value with CDC fault. Then we compute both $v(F_{ORI}) = v(A_{ORI}) <op> v(B)$ and $v(F_{CDC}) = v(A_{CDC}) <op> v(B)$, the edge from A to F is active when $v(F_{ORI}) \neq v(F_{CDC})$.

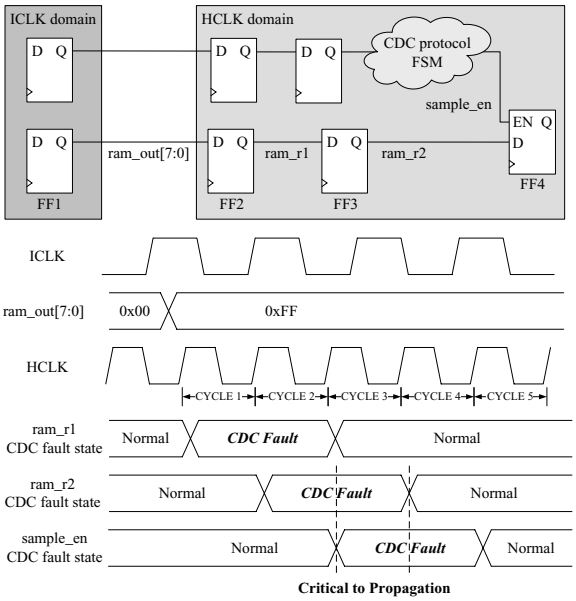

Figure 4. Scenario for Temporal Relation.

- Conditionals:

Consider the following conditional logic:

```
if (F)
  Out <= A;
else
  Out <= B;
```

If the conditional control variable F has CDC fault state, it may cause incorrect branch taken. Therefore the edge from F to *Out* is active when $A \neq B$. If the conditional input variable A has a CDC fault state, it will propagate to *Out* if F is true.

2. Sequential logic condition between vertices:

The CDC protocol is carefully designed for temporal relations between CDC signals. Take the scenario in Figure 4 for example. Vector *ram_out[7:0]* is latched by FF1 in ICLK domain and sampled by a two flip-flops synchronizer FF2 and FF3 in HCLK domain. The FF3's output *ram_r2* is sampled by FF4 if the enable signal *sample_en* is true. It is assumed that the CDC path delay between FF1 and FF2 is less than two HCLK cycles. If *ram_out[7:0]* has a new value, the FF2 may be metastable in the following cycle 1 and 2 in HCLK domain. And the CDC fault of FF2's output *ram_r1* may be active in cycle 1 and 2. FF3 samples the *ram_r1*, although it can not be metastable, it still may have wrong value in cycle 2 and 3. It is said that the CDC fault is *propagated* from FF2 to FF3 and *registered* for one HCLK cycle.

Multiple CDC faults could interact during their propagation [5]. In the above example, the active cycle of *sample_en* is critical for *ram_r1*'s CDC fault propagation to FF4. In most CDC designs, *sample_en* is an output of CDC protocol FSM and also driven by CDC signals. If *sample_en*'s CDC fault is also active in the cycle 2 or 3, *sample_en* and *ram_r1*'s CDC faults can interact to propagate or cancel each other. Therefore, we draw two conclusions on CDC fault propagation:

- CDC faults' propagation on G_{CDC} should calculate the temporal relations between them;
- Multiple CDC faults' interaction should be considered.

5. CDC Coverage Estimation

We define the CDC coverage based on the CDC fault model described in Section 3 and the CDC fault propagation described in Section 4.

First, we define *CDC fault coverage point* for each vertex $v \in V$ in G_{CDC}. It covers each CDC fault state of the vertex and represents the CDC fault's propagation. Second, a set of *monitor values* is defined for the observable vertex $o \in O$. Third, we define *CDC coverage points* for all observable vertices. It evaluates whether the CDC fault coverage points have an effect on every monitor value of those observable vertices. Finally, *CDC coverage* could be calculated as the ratio of the activated number to the total number of the CDC coverage points.

1. CDC fault coverage points:

We use $CP_f(v)$ to represent the CDC fault coverage points for vertex $v \in V$. It is defined recursively:

$CP_f(v) = \cup_{j=1}^{n} CP_f(w_j)$ where $w_j \in Pre(v)$, $n = |Pre(v)|$

For starting vertex $b \in B$, where the CDC fault is inserted and has no predecessor vertex, we define the $CP_f(b) = \{b_{setup}, b_{hold}\}$ to represent vertex b's CDC setup fault and CDC hold fault which are described in definition 2.

2. Monitor values:

The CDC fault is independent of the value of vertex. For observable vertex $o \in O$, which is compared with reference model to check for certain behavior, the CDC coverage should monitor the CDC fault for each meaningful value.

We use $Value(o)$ to present the set of monitor values of vertex $o \in O$. We divide the signals into two types: 1) Control signal: The monitor values should cover all possible values of control signal. For example we need to evaluate whether a control signal has CDC fault when it is set to both active and inactive. 2) Data signal: If the data signal is used as FIFO pointer, it should be covered when the pointer changes the FIFO full or empty signal. If the data signal is used only for transmission data and do not change the design status, we do not monitor the value of that signal, only evaluate the CDC fault coverage points.

3. CDC coverage points:

Let $Cov_{CDC}(O)$ denote the CDC coverage points for all observable vertices:

$$Cov_{CDC}(O) = \bigcup_{j=1}^{n} \left\{ Value(o_j) \times CP_f(o_j) \right\}$$
$$= \bigcup_{j=1}^{n} \left\{ Value(o_j) \times \cup_{k=1}^{m} CP_f(w_{j_k}) \right\}$$

where $o_j \in O$, $n = |O|$, $w_{j_k} \in Pre(o_j)$, $m = |Pre(o_j)|$

CDC coverage is calculated as the ratio of the activated number to the total number of the $Cov_{CDC}(O)$ during RTL simulation. The intuition behind CDC coverage is that, for all observable variables of CDC design, what is the proportion of the CDC faults would be observed and have an impact on the monitor values.

6. A Case Study

We applied our CDC coverage metric to a commercial IP which implements the ATA-5 IDE controller. It includes a vendor designed asynchronous FIFO to transmit data between AMBA AHB clock domain and IDE clock domain. While CDC coverage growing up, we found a bug on CDC protocol of this commercial IP.

Table 1. Basic of the ATA-5 IDE IP.

Module Name	Function Description	Num. of Lines	Num. of CDC Coverage Points
10fa	Asynchronous FIFO	426	9557
08fa	Async FIFO Control	917	3811
07fa	IDE PIO/DMA Control	353	1458
12fa	IDE UDMA Control	470	648
05fa	AHB DMA Control	471	207

6.1. Environment Setup

Our SoC verification environment was built up using Synopsys Vera and RVM [13]. We generated both IDE and AHB constraint random transactions. In order to guarantee the system behavior, we instanced AMBA VIP monitor and IDE device monitor to ensure the proper behavior of AHB and IDE interface. From bottom to up, the IP contains 5 hierarchy models related to G_{CDC} which implement asynchronous FIFO, asynchronous FIFO control, IDE PIO/DMA/UDMA timing control and AMBA AHB DMA control (shown in Table 1). The number of lines and the number of CDC coverage points for each module are also shown in that table. We inserted the CDC faults into original Verilog RTL description and calculated the propagation conditions while simulation proceeded. The CDC coverage was defined using Vera coverage group cross the monitor values and the CDC fault coverage points.

Table 2. Comparing Code Coverage with CDC Coverage.

Module Name	Line Coverage (%)			Path Coverage (%)			CDC Coverage (%)		
	100 Trans.	1500 Trans.	3000 Trans.	100 Trans.	1500 Trans.	3000 Trans.	100 Trans.	1500 Trans.	3000 Trans.
10fa	95.14	95.14	95.14	65.75	83.05	83.05	3.40	66.94	87.95
08fa	97.14	97.31	97.31	51.54	69.45	69.45	2.14	53.37	80.82
07fa	100	100	100	85.54	90.03	92.57	5.27	43.46	68.87
12fa	100	100	100	62.48	78.14	78.14	7.37	50.38	74.50
05fa	95.70	95.70	95.70	68.71	76.27	76.27	6.44	38.16	65.05

6.2. Results and Discussions

Table 2 compares the line coverage, path coverage and CDC coverage after 100, 1500 and 3000 IDE transactions, respectively. With 100 transactions, the line coverage already reaches at a steady and very high level. It has nearly no more increase after that. For path coverage, the same situation happens after 1500 transactions. But the CDC coverage is still increasing after 1500 transactions. It means that the transaction from 1500 to 3000 is still useful to find CDC function errors.

We found a bug related to an IDE interface output signal *NIOR* during the CDC coverage growing. In an UDMA read transaction, the CDC fault of a FIFO pointer would propagate to *NIOR*. The FIFO pointer with CDC fault would induce a FIFO full indicator to activate at a wrong time and abnormally suspend the read transaction. It is out of the IP designers' intention.

A CDC coverage which is not 100% means that some CDC fault of predecessor vertex was not observed at monitor value of successor vertex. It is important to find out whether it is unobservable or unobserved CDC fault. An unobservable CDC fault is the one that has been eliminated by CDC protocol and no test vector can detect. An unobserved CDC fault, however, could propagate to an observable variable during a simulation run. The fact that we have not observed the CDC fault merely indicates that we have not run the appropriate test vector. Static analysis on the TDFG and dynamic analysis during simulation could help to distinguish the unobservable and unobserved CDC fault.

7. Conclusions

In this paper, we presented CDC coverage for validation of multi-clock domain SoC designs. The method proposed made three major contributions. First, a CDC fault model was developed to present the actual effect of metastability. Second, we proposed a temporal data flow graph for the propagation of the CDC faults to observable variables. Finally, CDC coverage was defined based on the CDC faults and their observability.

This approach can be used in standard RTL simulation flow. For a multi-clock domain SoC design, a set of test vectors with higher CDC coverage is more likely to detect CDC function errors. Experiments on a commercial IP demonstrate that it is useful to find bugs related to CDC issues.

Our future work includes formal analysis the unobserved CDC fault from TDFG and a coverage feedback delay generator for CDC fault insertion. They could improve the accuracy and efficiency of CDC coverage.

References

[1] S. Devadas, A. Ghosh, and K. Keutzer. An observability-based code coverage metric for functional simulation. *ICCAD*, pages 418–425, 1996.

[2] C. Dike and E. Burton. Miller and noise effects in a synchronizing flip-flop. *IEEE Journal of Solid-State Circuits*, 34(6):849–855, Jun 1999.

[3] F. Fallah, S. Devadas, and K. Keutzer. OCCOM: efficient computation of observability-based code coverage metrics for functional verification. *DAC*, pages 152–157, 1998.

[4] F. Fallah, S. Devadas, and K. Keutzer. OCCOM-efficient computation of observability-based code coverage metrics for functional verification. *ICCAD*, 20(8):1003–1015, Aug 2001.

[5] R. Ginosar. Fourteen ways to fool your synchronizer. *Asynchronous Circuits and Systems (ASYNC)*, pages 389–96, May 2003.

[6] R. C. Ho and M. A. Horowitz. Validation coverage analysis for complex digital designs. *ICCAD*, pages 146–151, 1996.

[7] T. Kapschitz and R. Ginosar. Formal verification of synchronizers. *CHARME*, 2005.

[8] Y.-S. Kwon and C.-M. Kyung. Functional coverage metric generation from temporal event relation graph. *DATE*, pages 670–671, Feb 2004.

[9] C. Liu and J. Jou. An efficient functional coverage test for HDL descriptions at RTL. *ICCD*, pages 325–327, Oct 1999.

[10] T. Ly and N. Hand. Formally verifying clock domain crossing jitter using assertion-based verification. *Design & Verification Conference*, 2004.

[11] P. Mishra and N. Dutt. Functional coverage driven test generation for validation of pipelined processors. *DATE*, pages 678–683, 2005.

[12] A. Piziali. *Functional Verification Coverage Measurement and Analysis*. Kluwer Academic Publishers, 2004.

[13] Synopsys. *Vera User Guide. Reference Verification Methodology User Guide*, 2005.

[14] S. Tasiran and K. Keutzer. Coverage metrics for functional validation of hardware designs. *IEEE Design & Test of Computers*, 18(7):36–45, Jul/Aug 2001.

[15] Q. Zhang and I. G. Harris. A validation fault model for timing induced functional errors. *ITC*, pages 813–820, 2001.