Reset Domain Crossing for designs with set-reset flops

Abdul Moyeen[1], Inayat Ali[2]
[1]Mentor, A Siemens Business, abdul_moyeen@mentor.com;
[2]NXP Semiconductors, *inayat.ali@nxp.com*

Abstract-Reset domain Crossing has emerged into a major and un-avoidable design step in modern ASIC design flows. In any Digital design, Reset Domain Crossing (RDC) is essentially a structure where a signal crosses over from one reset domain to another reset domain.

There are cases where the Reset Domain definition is not that simple and straight forward. One such case is the handling of "Set-Reset" flops. We face design structures where there are more than one asynchronous set/reset controlling a flop. Then there can be scenarios involving data transfer between two such flops. Another matter of concern is if the output of such flops is used as reset further down the design. In this case, the question arises if we should treat that as a new reset domain or should there be some strategy to reuse the priority information of the contributing resets or the should the set/reset values of the flop be considered or the usage of the generated reset.

In this paper we will try to look at the problem structurally and propose a strategy to reach a conclusion where we face such design structures.

I. INTRODUCTION

Reset domain Crossing:

Reset domain crossing (RDC) is a structure in digital design where a signal generated by a flip-flop using an asynchronous reset (or set) is captured by another flop using another asynchronous reset (or set).

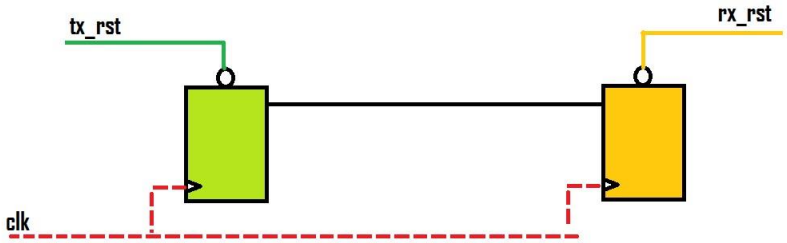

Figure 1. Basic RDC (Reset domain crossing)

Every reset signal in a digital design, which can assert asynchronously and independently of any clock, is called as *"Asynchronous reset".* All flops using same asynchronous reset in a digital design corresponds to a *"Reset domain".* So, if signal generated by a flop in one reset domain is captured by a flop in another reset domain, such structures are called *"Reset Domain Crossings".* The transmitting flop is called Tx flop and the receiving flop is referred to as Rx flop. Further, *reset (or set)* driving the Tx flop is called *tx_rst (or tx_set)* and *reset (or set)* driving the Rx flop is called *rx_rst (or rx_set).*

This can be problematic because an asynchronous reset assertion on the Rx flop can occur asynchronous to the Rx clock, resulting in metastability or data loss. Suppose both flops are using the same clock and the Tx flop launches at the positive edge of clock, an asynchronous rx_rst at same time can force the Rx flop to a conflicting value at same time, forcing the flop into metastability or incorrect data capture.

The problem becomes more complex when set-reset flop comes into picture. A set-reset flop, in the context of this paper, is a flip-flop with two asynchronous set/reset (Figure 2). With more than one reset on a single flop, the problem become more complex. Although designers are taught to avoid such structures in designs, various design scenarios can

still end-up with RTL code which can get synthesized as a set/reset flop. Often in large design teams, engineers or teams responsible for RDC cleanup of the design are different from the RTL coders. And with tight design schedules and RDC happening late in the RTL closure stage, it is becoming very difficult to avoid such design structures. This structure is hence expected to be caught by the RDC EDA tools and have to be analyzed and fixed by the RDC engineer.

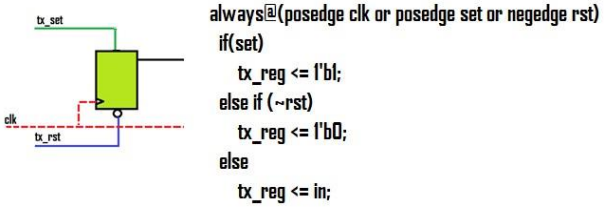

Figure 2. Basic SR flop with active_high priority set and active_low secondary reset.

As per our experience, we have seen that most common usage of set-reset flop is as shown in example above in Figure 2. As possible in RTL code, this set-reset flop can have many variations (Table 1) based on edge sensitivity of both resets involved and the value forced by these resets. For simplicity, if the value forced is binary 1, we will call it a set, and if the value forced to binary '0', we would call it a reset. Also, you can also see that in the RTL code in Figure 2 above suggest that one reset has priority over another based on the if-else conditional hierarchy.

In this Paper, we will assume all sets and resets as asynchronous. Cases covering one or both of resets being synchronous are kept out of the scope of paper and can be considered for future work. We will also assume that one set/reset is always having priority over another as shown in the example above. We have also kept non-resettable flops out of the purview of this paper.

We will use following nomenclature moving forward.

- In general, a flop with two resets will be called a *set-reset flop* or *SR flop* in short.
- A reset signal forcing the flop to binary '1' will be called as 'set'
- A reset signal forcing the flop to binary '0' will be called as 'reset' or 'rst' in short.
- The reset having priority over another in if-else tree will be considered 'priority set' or 'priority reset' depending upon the output.
- The reset having lower priority another in if-else tree will be considered 'secondary set' or 'secondary reset' depending upon the output.
- In general, a wire will be called 'reset' as a generic term if used in a flop to force a specific constant value.

As discussed above, theoretically we can have several variations of set-reset flop with difference in edge sensitivity of resets, value forced at output of flop and also priority.

Table 1. Variations of set/reset for one flop

S.no	Priority set/reset	Secondary set/reset	Sample Code for case F7

F1	Active high Set	Active high Set	always@(posedge clk or posedge set or negedge rst)
			if(set)
			sr_reg <= 1'b1;
			else if (~rst)
			sr_reg <= 0; else
			sr_reg <= in;

F2	Active low Set	Active high Set	
F3	Active high Set	Active low Set	
F4	Active low Set	Active low Set	
F5	Active high Set	Active high Reset	
F6	Active low Set	Active high Reset	
F7	Active high Set	Active low Reset	
F8	Active low Set	Active low Reset	
F9	Active high Reset	Active high Set	
F10	Active low Reset	Active high Set	
F11	Active high Reset	Active low Set	
F12	Active low Reset	Active low Set	
F13	Active high Reset	Active high Reset	
F14	Active low Reset	Active high Reset	
F15	Active high Reset	Active low Reset	
F16	Active low Reset	Active low Reset	

It should be noted that the most common usage is case "F7" which is priority active_high set and secondary active_low reset, for which the example Verilog code is given in Figure 2. We will be using this example more often moving forward.

Table 1 is an exhaustive list and real design scenarios might not see all these cases. But theoretically all cases are possible depending upon the design requirements. So, it is important for the RDC engineers to understand the issues related to those structures.

II. PROBLEMATIC AND SAFE RDC

A. Problem Statement

Whenever there are multiple resets involved in a crossing, there is a risk of potentially problematic RDC. But if we classify all RDCs as problematic and try to insert synchronizers everywhere, will be an overkill and will surely be against our design constraints and is not a good solution. EDA RDC tools on the other hand are also noisy in their analysis. Considering tight deadlines and already squeezed schedules, it is dangerous to get into analyzing false crossings, wasting precious time.

There can be many structures or design conditions in RDC involving set reset flops which can affect the data handling over reset domain crossings and hence result in a safe design crossing not requiring any design change. We will discuss this later in the paper.

This paper will try to structurally analyze the problem and come up with a basic strategy which can help designers state conditions involving set-reset flops, in order to help classify RDC crossing as safe or dangerous based on purely structural basis.

B. Usage Definition

When is the RDC problematic and when it is safe? As a matter of fact, an RDC is considered problematic if Rx flop can go into metastability or possibility of data loss. So, we need to figure out all cases where, due to some assumptions and design conditions, Rx can never go into metastability or incorrect data capture and mark them safe.

SRDC Baseline definition:

We will use following baseline definition to decide an RDC to be safe or not:

- Identify structures which makes sure Rx is not asserted.
    - Either Always, or/and simultaneously when Tx reset is asserted.
    - This is because if Rx flop is no asserted, there can never be any problematic RDC.

- Structures which make sure Rx reset is already asserted when Tx reset asserts.
    - This is because if Rx flop is having an ordered relationship with Tx reset and is sure to be asserted when Tx reset asserts, there is no problem in RDC.

- Following condition is true whenever Rx reset asserts:
    - There is no conflict in value forced on Rx flop either through data path or Reset path.
        - Data path value means Rx is fed by a 0 or 1 from Tx flop output, while clock edge comes
        - Reset path value means Rx reset is being asserted and a 0 if rx_rst is used as a reset and 1 if Rx reset is used as a set.
        - Data path value will depend on Tx assertion
        - Reset Path value will depend on Rx assertion

We will also have to define "*Relationship*" between two resets.

Relationship: we will be talking about binary relationship of any two resets. We will define relationship between two resets as Direct, Inverted or No-control.

Direct Relationship: if one reset is driven by another reset in its combo cone and is exactly following another reset keeping all other variables as constants, it will be a direct relationship. i.e., if one reset goes 0, another also goes 0. And if one reset goes 1, this also goes 1.

Inverted Relationship: if one reset is driven by another reset in its combo cone and is inversely following another reset keeping all other variables as constants, it will be a direct relationship. i.e. if one reset goes 0, another also goes 1. And if one reset goes 1, this also goes 0.

No Control: means both resets are either structurally unconnected or there is no Direct or inverted relationship between two.

Based on these relationships and the SRDC baseline definitions above, we created the following table for any simple RDC structure with variation in there set/reset value and assertion logic (active high or active low) to decide if an RDC is safe or not.

Table 2. Variations of simple RDC Crossing considering one reset at a time.

TX Set/Reset	Tx Used as	Rx Set/Reset	Rx Used as	Relationship	Safe RDC?
Active high	Set	Active high	Set	Direct	YES
Active low	Set	Active low	Set	Direct	YES
Active high	Set	Active high	Reset	Direct	NO
Active low	Set	Active low	Reset	Direct	NO
Active high	Reset	Active high	Set	Direct	NO
Active low	Reset	Active low	Set	Direct	NO
Active high	Reset	Active high	Reset	Direct	YES
Active low	Reset	Active low	Reset	Direct	YES
Active high	Set	Active high	Set	Inverted	NO
Active low	Set	Active low	Set	Inverted	NO
Active high	Set	Active high	Reset	Inverted	YES
Active low	Set	Active low	Reset	Inverted	YES
Active high	Reset	Active high	Set	Inverted	YES
Active low	Reset	Active low	Set	Inverted	YES
Active high	Reset	Active high	Reset	Inverted	NO
Active low	Reset	Active low	Reset	Inverted	NO
Active high	Set	Active high	Set	No-Control	NO
Active low	Set	Active low	Set	No-Control	NO
Active high	Set	Active high	Reset	No-Control	NO
Active low	Set	Active low	Reset	No-Control	NO
Active high	Reset	Active high	Set	No-Control	NO
Active low	Reset	Active low	Set	No-Control	NO
Active high	Reset	Active high	Reset	No-Control	NO
Active low	Reset	Active low	Reset	No-Control	NO

This table (Table 2) is important as we will now use this to define if any RDC is safe or not in our further analysis in this paper.

Classification of set-reset flops based on usage: Set-reset flops or SR flops can also be classified on usage in design. To analyze the problem structurally we can define the usage for SR flop as Tx flop, Rx flop or both.

Usage variations of set-reset flop considered in this paper:

1. Set-reset flop used as only as Tx
    - 1.1 Output of set-reset flop used as data.
    - 1.2 Output of set-reset flop used as reset.
2. Set--reset flop as Only Rx.
3. Set-reset as both Rx and Tx

We will take these 3 cases above and will try to put down our observation. We will try to point out cases which will not end up in RDC and can be waived or removed from analysis automatically or by tool if possible.

III. ANALYSIS

A. *Case1 : set-reset flop only in Tx and Output used as Data*

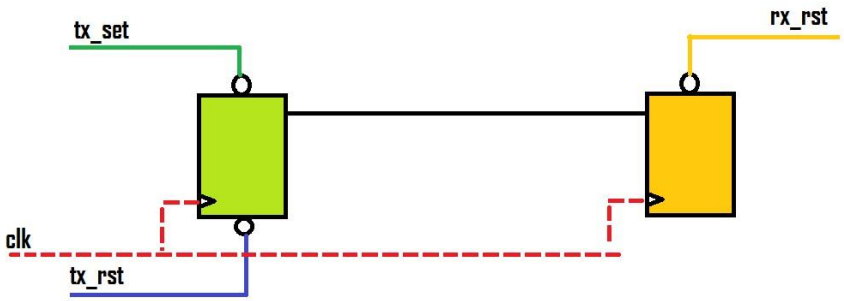

Figure 3. SR flop only in Tx and Output used as Data.

This behavior of this structure will depend on the relationship between tx_set /tx_rst and rx_rst. For example a simple case can be where rx_rst is in "direct relationship" with tx_rst , so automatically which means Tx flop will be reset whenever Rx flop is reset. Hence there will never be a case that can result in metastability. If we see the problem structurally, we see following variables which define the problem.

X1 = Priority reset is set or reset. (0 for reset, 1 for set)
X2 = Priority reset is active_high or active_low. (0 for active_low, 1 for active_high)
X3 = Secondary reset is set or reset. (0 for reset, 1 for set)
X4 = Secondary reset is active_high or active_low. (0 for active_low, 1 for active_high)
X5 = Rx reset is set or reset. (0 for reset, 1 for set)
X6 = Rx reset is active_high or active_low. (0 for active_low, 1 for active_high)
X7,X8 = Two bit variable for Rx relationship with priority set/reset, 00 for Direct, 01,10 for No relationship , 11 for Inverted.
X9,X10 = Two bit variable for Rx relationship with secondary set/reset, 00 for Direct, 01,10 for No relationship , 11 for Inverted.

SRDC = 0 if RDC is not safe and = 1 of RDC is safe.

We can define SRDC as a function of all other variables as:

SRDC = f(X1,X2,X3,X4,X5,X6,X7,X8,X9,X10)

Using all 10 variables we can create a large 10-variable truth table for which we can fill up SRDC value as per table 2 above. Which can be solved into a binary equation and be used as a benchmark to define an RDC is safe for this scenario or not. This truth table can either be used by any RDC tool or fed in any script to get a list of safe RDC crossings.

For a quick analysis, we can use same truth table for a specific scenario.

Let's assume:

- SR flop to be of F7 type as per table 1. Which is active_high priority set and active low secondary set
- Rx flop is active low reset.

Now only variation left is the relationship between variables X7, X8, X9, X10. Value of Safe RDC can be decided based on *table 2*.

For example, for first line entry in truth table shown in figure 4 below:

If X7=0, X8=0, X9=0, X10 = 0, this condition means:

- {X7,X8} = Two bit variable for Rx relationship with priority set/reset, 00 for Direct, 01 and 10 for No relationship, 11 for Inverted.
  - {X7,X8} = {0,0} means Rx reset is in direct relationship with priority Tx reset.
- {X9,X10} = Two bit variable for Rx relationship with secondary set/reset, 00 for Direct, 01 and 10 for No relationship , 11 for Inverted.
  - {X9,X10} = {0,0} means Rx reset is in direct relationship with secondary Tx reset.

So this case can never be a problem as per table 2. Hence SRDC = 1.
Similarly we can fill all other values for SRDC and get following truth table.

X7	X8	X9	X10	Safe RDC?
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Karnaugh map for same

X7,X8 \ X9,X10	OO	OI	II	IO
OO	1	1	1	1
OI	1	0	0	0
II	0	0	0	0
OO	1	0	0	0

Figure 4. Truth Table for special case F7 and Karnaugh map.

Solving Karnaugh map in *figure 4* we will get following equation:

$$SRDC = X7'.X8' + X9'.X10'.(X7' + X8') \quad \text{-------} \quad eq\ 1.$$

Equation 1 above actually means that this RDC is safe only if there is:

- A direct relationship between priority set/reset and Rx. (first row of Karnaugh map)
- A direct relationship between secondary set/reset and Rx given there is no inverted relationship between Primary and Rx. (first column of Karnaugh map except third element)
  - But this exception cannot happen as then it will mean:
    - Rx having direct relationship with Priority.
    - Rx having inverted relationship with Secondary
    - Both above cannot be true simultaneously.
    - Hence Primary and secondary will have an inverted relationship. Which is not a practical scenario and an example of bad design. Pointing out such cases can be considered for future work on this topic.

Now this is an interesting piece of information which can be reached directly by observing the crossing. But the method employed above will generalize the solution. Similarly, equation can be derived for all 16 * 4 cases. Or a large truth table can be created for all 10 variables and solved to get a SRDC equation valid for all such scenarios.

B. Case2 : set-reset flop only in Rx and Output used as Data

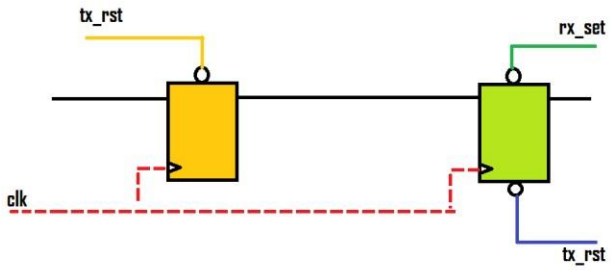

Figure 5. SR flop only in Rx.

In this case we will always be using output of Tx flop only as data because if it's used as reset, it will be normal RDC structure.

We can do a similar analysis as done for case1.
Structurally, we see following variables which define the problem.

X1 = Priority reset is set or reset. (0 for reset, 1 for set)
X2 = Priority reset is active_high or active_low. (0 for active_low, 1 for active_high)
X3 = Secondary reset is set or reset. (0 for reset, 1 for set)
X4 = Secondary reset is active_high or active_low. (0 for active_low, 1 for active_high)
X5 = Tx reset is set or reset. (0 for reset, 1 for set)
X6 = Tx reset is active_high or active_low. (0 for active_low, 1 for active_high)
X7,X8 = Two bit variable for Tx relationship with priority set/reset, 00 for Direct, 01,10 for No relationship , 11 for Inverted.
X9,X10 = Two bit variable for Tx relationship with secondary set/reset, 00 for Direct, 01,10 for No relationship , 11 for Inverted.
SRDC = 0 if RDC is not safe and = 1 of RDC is safe

SRDC = f(X1,X2,X3,X4,X5,X6,X7,X8,X9,X10)

Similar to Case1, Let's analyze for the most common scenario, which is:

- Rx flop is active low reset
- Tx flop is priority active high set and secondary active low reset

Filling in the truth table again for these values and as per table 2, we will again get a truth table and Karnaugh map as follows:

X7	X8	X9	X10	Safe RDC?
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Karnaugh map for same

X7,X8 \ X9,X10	OO	OI	II	IO
OO	1	1	1	1
OI	1	0	0	0
II	0	0	0	0
OO	1	0	0	0

Figure 6. Truth Table for special case F7 and Karnaugh map

Solving Karnaugh map we will get following equation:

$$SRDC = X7'.X8' + X9'.X10'.(X7' + X8') \qquad \text{-------} \quad eq\ 2.$$

Which means that this RDC is safe only if there is:

- A direct relationship between primary and Tx. (first row of Karnaugh map)
  - A direct relationship between secondary and Tx given there is no inverted relationship between Primary and Rx. (first column of Karnaugh map except third element)
    - Similar to case1,this exception actually cannot happen as because then it will mean :
      - Rx having direct relationship with Priority
      - Rx having inverted relationship with Secondary

Equation can be derived for all 16 * 4 cases. Or a large truth table can be created for all 10 variables and solved to get a SRDC equation valid for all such scenarios.

C. Case3 : set-reset flop used in Both Tx and Rx and Data generated from Tx flop feed into Tx input pin.

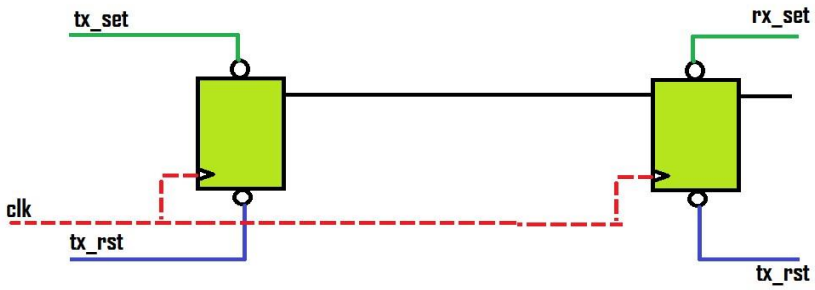

Figure 7. SR flop used in both Rx and Tx.

As shown in Figure 7 above, in this scenario, we can see that there are 2 pairs of set/reset. We can easily say that if all of these 4 are different and independent of each others, the crossing is not safe and can result is an RDC. So, for this scenario also we will try to list down cases which are safe.

Again, structurally, we see following variables which define the problem. Here four variables are increase in form of secondary set/reset and its priority.

X1 = Priority Tx reset is set or reset. (0 for reset, 1 for set)
X2 = Priority Tx reset is active_high or active_low. (0 for active_low, 1 for active_high)
X3 = Secondary Tx reset is set or reset. (0 for reset, 1 for set)
X4 = Secondary Tx reset is active_high or active_low. (0 for active_low, 1 for active_high)
X5 = Priority Rx reset is set or reset. (0 for reset, 1 for set)
X6 = Priority Rx reset is active_high or active_low. (0 for active_low, 1 for active_high)
X7 = Secondary Rx reset is set or reset. (0 for reset, 1 for set)
X8 = Secondary Rx reset is active_high or active_low. (0 for active_low, 1 for active_high)
X9,X10 = Two bit variable for Primary Rx relationship with Primary Tx set/reset, 00 for Direct, 01,10 for No relationship , 11 for Inverted.
X11,X12 = Two bit variable for Primary Rx relationship with Secondary Tx set/reset, 00 for Direct, 01,10 for No relationship , 11 for Inverted.
X13,X14 = Two bit variable for Secondary Rx relationship with Primary Tx set/reset, 00 for Direct, 01,10 for No relationship , 11 for Inverted.
X15,X16 = Two bit variable for Secondary Rx relationship with Secondary Tx set/reset, 00 for Direct, 01,10 for No relationship , 11 for Inverted.
SRDC = 0 if RDC is not safe and = 1 of RDC is safe

SRDC = f(X1,X2,X3,X4,X5,X6,X7,X8,X9,X10,X11,X12,X13,X14,X15,X16)

We will reduce this equation by solving for the most-common SR flop. Also further solving only for case where:

- Secondary Rx don't have Primary Tx in input cone.
- Secondary Rx don't have Secondary Tx in input cone.

Which means Secondary Rx is async with no relationship with any other contributor. Hence a pretty pessimist case. Filling in the truth table again for these values and as per table 2, we will get a truth table and Karnaugh map:

X9	X10	X11	X12	X13	X14	X15	X16	Safe RDC?
0	0	0	0	X	X`	X	X`	1
0	0	0	1	X	X`	X	X`	1
0	0	1	0	X	X`	X	X`	1
0	0	1	1	X	X`	X	X`	1
0	1	0	0	X	X`	X	X`	0
0	1	0	1	X	X`	X	X`	0
0	1	1	0	X	X`	X	X`	0
0	1	1	1	X	X`	X	X`	1
1	0	0	0	X	X`	X	X`	1
1	0	0	1	X	X`	X	X`	0
1	0	1	0	X	X`	X	X`	0
1	0	1	1	X	X`	X	X`	0
1	1	0	0	X	X`	X	X`	1
1	1	0	1	X	X`	X	X`	1
1	1	1	0	X	X`	X	X`	1
1	1	1	1	X	X`	X	X`	1

X9,X10 \ X11,X12	OO	OI	II	IO
OO	1	1	1	1
OI	0	0	1	0
II	1	1	1	1
IO	1	0	0	0

Figure 8. Truth Table for special case F7 and Karnaugh map

Solving this we get following equation:

$$SRDC = (X9.X10 + X9'.X10' + X9(X11'X12') + X9'(X11.X12)) \quad \text{-------} \quad eq\ 3.$$

Similarly, this equation can be solved for a generic case of all 16 variables.

D. Case4 : set-reset flop used in Tx and Output used as Reset But Tx flop is simple flop

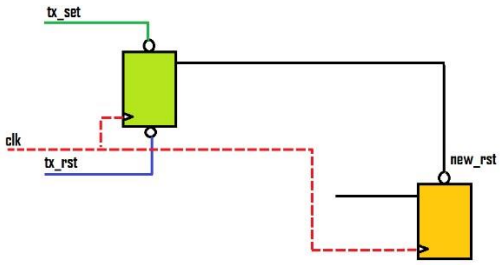

Figure 9. SR flop output used as clock

This problem in this scenario is to define what reset domain should Tx flop be called in and if that makes a crossing with another level for Tx2 (as in ckt shown below), In what scenario this will be a safe crossing? Or if the data coming in will result in a crossing with Tx flop or not. So if we consider the output of the set/reset flop as *new_rst*, what are the cases which will not cause problematic RDC? With a third flop in picture, apart from Tx and Rx flop, we divide this in two sub-scenarios.

a.) Flop using set-reset generated reset as Rx and a third flop as Tx.
b.) Flop using set-reset generated reset as Tx and a third flop as Rx.

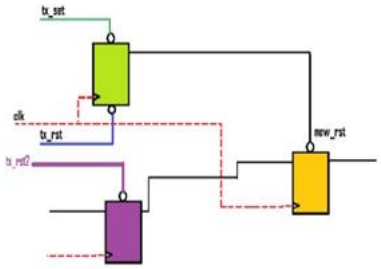

Figure 10. SR flop output as Clock and third flop as Tx

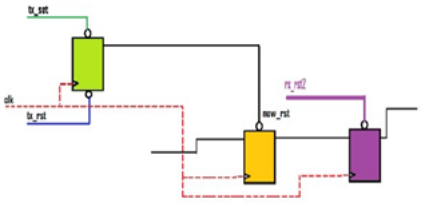

Figure 11. SR flop output as Clock and third flop as Rx

In both scenarios, apart from the third set/reset, one extra thing which comes into play is data stored in set-reset flop. And we also do not have any control over that. By control over data, we mean we don't know the value stored in set-reset flop in de-asserted mode and that is effecting the generated reset.

Another way to look at this scenario is Imagining a basic RDC crossing with a tx_rst and an rx_rst, with either or both resets are generated by SR flops.

Let's try to look the problem from purely structural point of view (and leave out the effect of the value stored in set-reset flop for future work). We can assume generated reset to be used as only active_low reset just to reduce variables in our equation. Also the third flop as TX for specific case analysis.

So similar to case1, we can try forming an equation here also. Variables will be defined as follows:

X1 = Priority reset is set or reset. (0 for reset, 1 for set)
X2 = Priority reset is active_high or active_low. (0 for active_low, 1 for active_high)
X3 = Secondary reset is set or reset. (0 for reset, 1 for set)
X4 = Secondary reset is active_high or active_low. (0 for active_low, 1 for active_high)
X5 = Third flop is Rx or Tx
X6 = Third reset is set or reset. (0 for reset, 1 for set)
X7 = Third reset is active_high or active_low. (0 for active_low, 1 for active_high)
X8,X9 = Relationship with priority, 00 for Direct, 01,10 for No relationship, 11 for Inverted.
X10,X11 = Relationship with priority, 00 for Direct, 01,10 for No relationship, 11 for Inverted.
SRDC = 0 if RDC is not safe and = 1 of RDC is safe

Accordingly we can have an SRDC function as:
SRDC = f(X1,X2,X3,X4,X5,X6,X7,X8,X9,X10,X11,X12)
We will analyze this also for the specific case where SR flop is active high priority set and active low, generated reset is used as active_low reset and third flop is also active low reset.

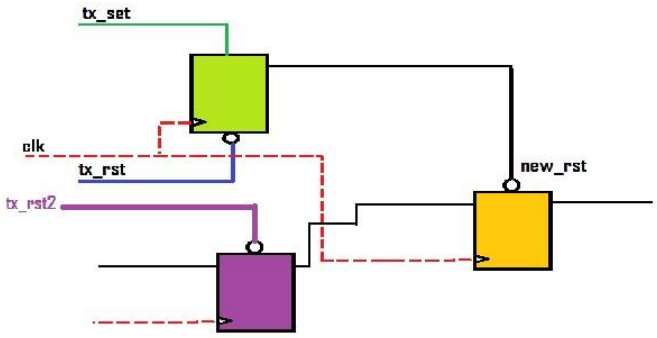

Figure 12. SR flop output as Clock and third flop as Tx for specific case

Following truth table and Karnaugh Map will be drawn for this case:

X8	X9	X10	X11	Safe RDC?
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

X8, X9 \ X10, X11	OO	OI	II	IO
OO	1	1	1	1
OI	1	0	1	0
II	1	1	1	1
IO	1	0	1	0

Figure 13. Truth Table for special case F7 and Karnaugh map

Binary equation from above Karnaugh map from be derived as:

SRDC = (X8'X9' + X8X9 + X10'X11' + X10X11) ------- *eq 4.*

Similarly a full equation can be drawn for this case involving all variables.

IV. RESULTS

Four equations for 4 specific cases are presented in this paper. These cover the most common cases involving set-reset flops where output is used as data or reset. For generic cases we can derive the equations honoring all variables.

Case1 : Set-reset flop only in Tx and Output used as Data
SRDC = X7'.X8' + X9'.X10'.(X7' + X8') ------- *eq 1.*

Case2 : Set-reset flop only in Rx and Output used as Data
SRDC = X7'.X8' + X9'.X10'.(X7' + X8') ------- *eq 2.*

Case3 : Set-reset flop used in Both Tx and Rx and Data generated from Tx flop feed into Tx input pin.
SRDC = (X9.X10 + X9'.X10' + X9(X11'X12') + X9'(X11.X12)) ------- *eq 3.*

Case4 : Set-reset flop used in Tx and Output used as Reset but Tx flop is simple flop
SRDC = (X8'X9' + X8X9 + X10'X11' + X10X11) ------- *eq 4.*

V. Conclusion and Future Work

The four equations derived above form the basis for mathematically deciding if an RDC involving an SR flop is safe or not. These mathematical equations and underlying logic can now be used either by EDA tools or scripts to decide if an RDC is safe or not.

There are many cases which are marked as safe due to various reasons, but many of them come under a case where the data will be blocked in that path. For example, in Case 1, if the SR flop is priority set with active high and Rx flop is active low set: this case will always keep the output of Rx flop to a high. This may seem like a safe crossing, but this is surely a stuck-at-one resulting in extra hardware. We can point out such scenarios easily from the equations mentioned above.

The super set of four equations can be driven using all the effective variables providing a complete solution.

Among the safe RDCs there are also few scenarios resulting in a SR flop to be set and reset simultaneously. These can be pointed out for correction as these should never exist in real design.

Acknowledgment

We would acknowledge all our friends and colleagues who worked with us on numerous project cleaning up digital designs for Reset domain crossing and learning a lot in the process. This paper is a tribute to numerous discussion we had over violations from different EDA tools reporting a problem in our design when we thought there are none.

References

[1] Noise Reduction in Reset Domain Crossings Verification Using Formal Verification , Mohamed Fawzy; Ahmed Elgohary; Hala Ibrahim , 10.1109/EWDTS50664.2020.9224763

[2] Asynchronous & Synchronous Reset Design Techniques, Clifford E. Cummings, Don Mills, Steve Golson .

[3] K. Nouh Ahmed and A. Abbas, "Multiple reset domains verification using assertion based verification", 2017 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC), pp. 1-6, 2017.

[4] https://www.mentor.com/products/fv/resources/overview/faster-reset-verification-closure-with-intelligentreset-domain-crossings-detection-56e29187-ff6e-4f38-a811-795b3ec5e5ca

[5] Multi-Domain Verification of Power, Clock and Reset Domains, Ping Yeung , Mentor Graphics , DOI: 10.1007/978-3-319-26287-1_15

[6] https://www.mentor.com/products/fv/resources/overview/a-specification-driven-methodology-for-thedesign-and-verification-of-reset-domain-crossing-logic-0c10dfb0-7079-4e0d-90184730c4095f7a?cmpid=10168