ABSTRACT

The world is gradually being influenced by technological systems. Therefore, safety should be put in a high regard in safety-critical applications where lives of humans are at stake. If the software is defective, it may be fixed with the use of a patch program. The costs associated with this procedure will be minimal. If the hardware is defective, however, it may result in significant life and financial loss. As a result, the industries devote roughly 70% of the project's budget on hardware verification and validation. Hence, the verification teams assure that there are no errors when the product is ready by performing several techniques. Formal verification is the use of mathematical formula to verify the behavior of a system. Equivalence Checking, Model Checking and Theorem Proving are three methods of formal verification.

In this project, we used equivalence checking to verify the equivalence between RTL and gate level netlist and fix the bugs between gate and buggy gate level description of the same design using Synposys Formality and Cadence Conformal tools. Additionally, the comparison between Synposys Formality and Cadence Conformal tools are provided.

This project report provides brief description about,

i. System behavior and function of RS232
ii. RTL synthesis using Synopsys design compiler.
iii. Equivalence checking between RTL-gate and gate-buggy gate level design (bug hunting), using Synposys Formality and Cadence Conformal tools.
iv. Description of validation process in Synopsys Formality and Cadence Conformal.
v. Analysis of verification result.

1. INTRODUCTION:

Formal verification aims to construct a computer-based mathematical model of the system and verify if the model meets the intended behavior. In this project, we are going to verify RS232 model using formal verification tool such as Synopsys Formality and Cadence Conformal tools based on equivalence checking methodology. RS232 is an asynchronous communication protocol that lets you transfer data between electronic devices. Since data is only transmitted in one way per line, bi-directional communication (two directions) requires two wires. Therefore, the two wires along with a third ground reference wire make the configuration of RS 232.

1.1. RS 232 – SYSTEM BEHAVIOUR AND FUNCTION

RS232 is a two-way communication system that sends and receives data. There are two devices connected to each other: (DTE) Data Transmission Equipment and (DCE) Data Communication Equipment, both of which have pins such as TXD (Transmitter), RXD (Receiver), RTS (Request To Send), and CTS (Clear To Send). The RTS now creates a request to transfer data from the DTE source. Then, on the other side, DCE, the CTS, clears the way for the data to be received. It will deliver a signal to RTS of the DTE source to send the signal after clearing a path. Afterwards, the bits are sent from DTE to DCE. RTS and CTS of DTE sources can now create the request from DCE sources, clearing the path for receiving data and giving a signal to transmit data. This is the entire data transmission process. Two-wire RS232 communication module with 5, 6, 7, and 8-bit word communication, as well as parity, and 1 and 2 stop bits. A Wishbone interface is used to regulate the transmit and receive buffers. Auxiliary signals for Buffer status and interrupt driven routines are supplied in addition to the Wishbone interface.

Settings:

- Baud rate is 9600 bps
- The output voltage specification,
    - +5V to +25V (transmitting a logical zero) and
    - -5V to -25V (transmitting a logical one).
- The receiver voltage specification,
    - Greater than 3V (receiving logic zero) and
    - Smaller than -3V (receiving logic one).

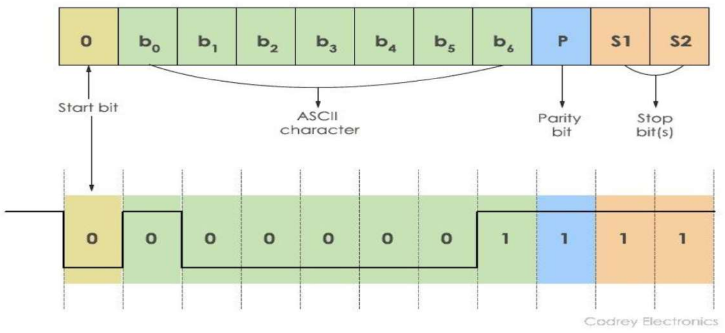
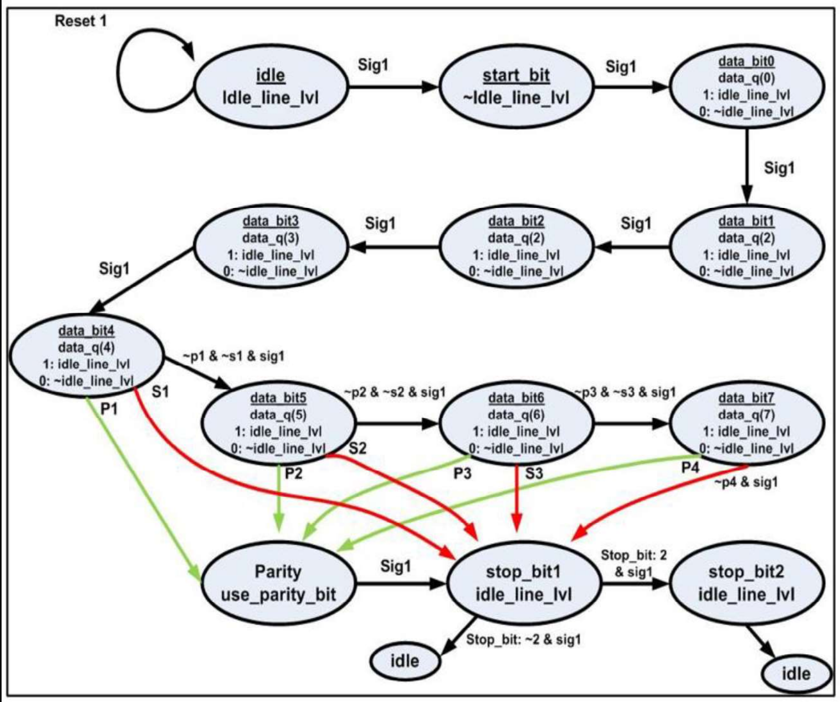


Figure 1: RS232-Transmission

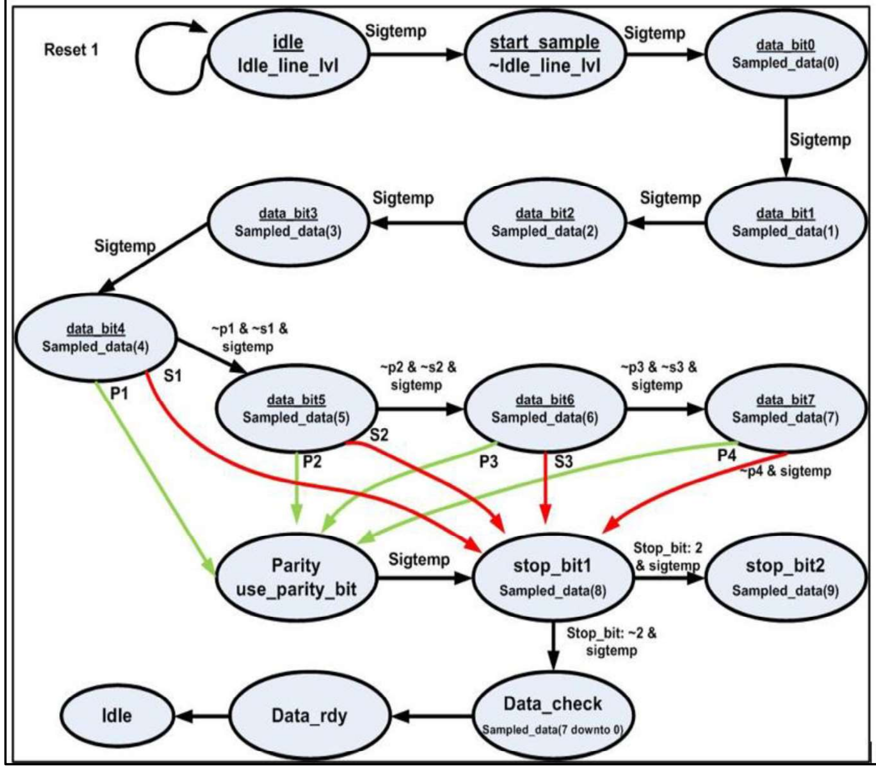
1.2. State Diagram:

The below state diagrams of transmitter and receiver explains the behavior of the system.



Sig1 – Register_enable & ~ Reset
P1 – Word_width = 5 & use_parity_bit & sig1
S1- Word_width = 5 & ~ use_parity_bit & sig1
P2- Word_width = 6 & use_parity_bit & sig1
S2- Word_width = 6 &~ use_parity_bit & sig1
P3- Word_width = 7 & use_parity_bit & sig1
S3- Word_width = 7 & ~ use_parity_bit & sig1
P4- sig1 & use_parity_bit

Figure 2 : Transmitter State Diagram



Sigtemp – baud_tick = '1' & ~ reset
P1 – word_width = 5 & use_parity_bit
S1 – word_width = 5 & ~use_parity_bit
P2 – word_width = 6 & use_parity_bit
S2- word_width = 6 & ~use_parity_bit
P3- word_width = 7 & use_parity_bit
S3- word_width = 7 & ~ use_parity_bit

Figure 3 : Receiver State Diagram

1.3. Circuit Diagram using precision tool:

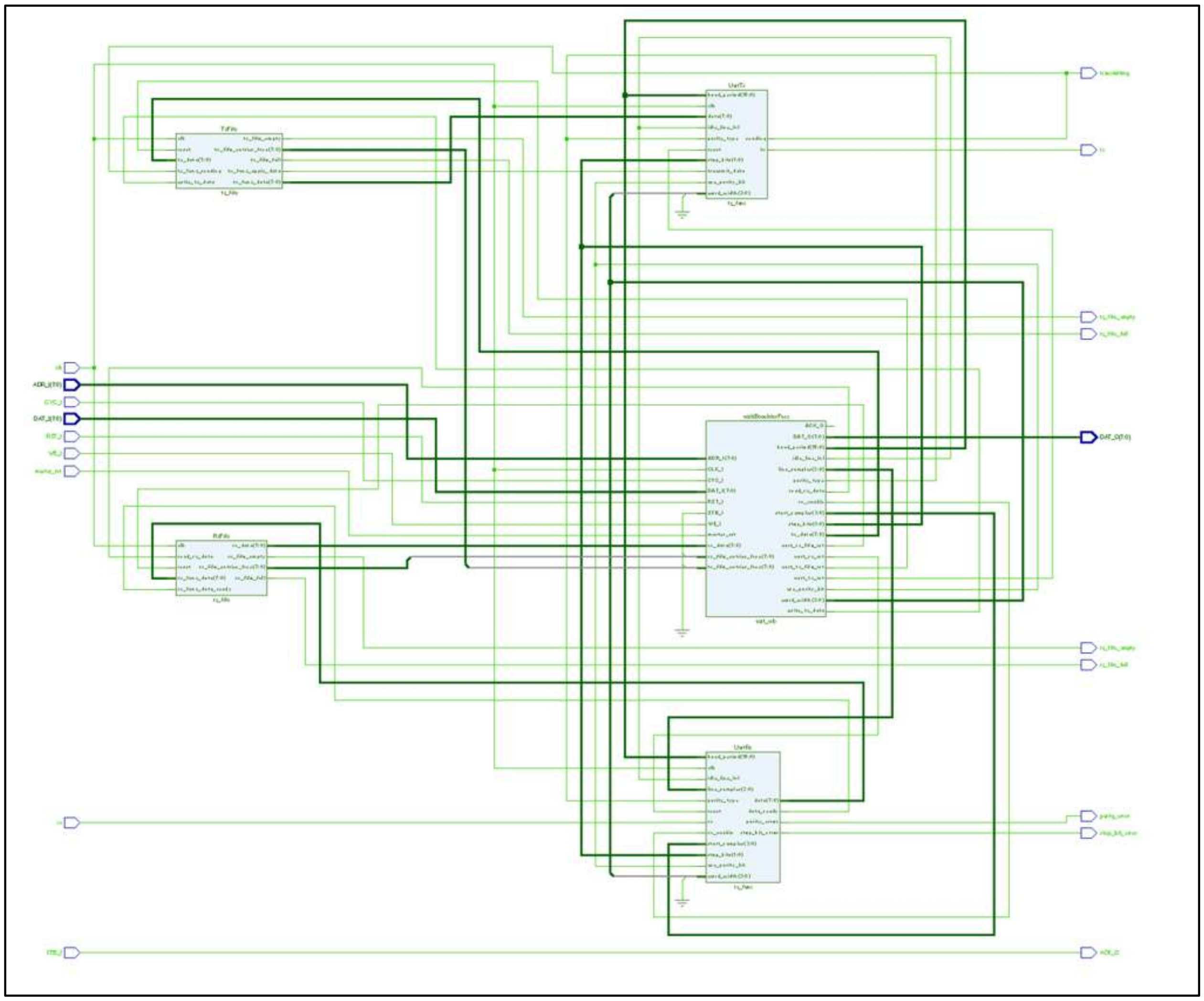


Figure 4 : Circuit Diagram

2. RTL SYNTHESIS USING SYNOPSYS DESIGN COMPILER:

The Design Compiler RTL synthesis solution allows users to determine today's design difficulties by optimizing timing, area, power, and test at the same time. Design Compiler incorporates topographical technology that ensures a consistent flow and faster time to results. Logic synthesis tool involves the process of converting an RTL circuit description into an optimized gate-level representation.

There are three steps involved in logical synthesis. The RTL description is first converted to a boolean description that is not optimized. HDL constructs like IF, CASE, LOOPs, and conditional assignments are transformed to boolean counterparts that include elementary gates like NAND and NOR gates, flops, and latches etc.,

The logical optimization step comes after this. To provide an optimal equivalent description, boolean optimization methods are utilized. To optimize the logic, these methods use logic flattening and logic factoring operations, as well as fan out and loading limitations. Flattening operations eliminate structure, whereas factoring procedures provide new structure. When these operations are used in combination, they serve to optimize the logic.

Finally, the optimized boolean equivalent description is translated to actual logic gates using the target process's technology library. To create a netlist, this phase takes logical and temporal information from the technology library. The created netlist satisfies the user's requirements for area and speed.

Logical synthesis eliminates the issue of distinct designer styles for different design blocks and suboptimal designs. Technology-independent design is possible with logic synthesis tools. To synthesis tools, design engineers submit HDL descriptions as well as different restrictions and limitations on the design. These limitations represent the requirements that the design must fulfil. It supports VHDL and Verilog languages. It can also synthesize to other design libraries such as the vtvt_tsmc libraries or OSU standard cell libraries.

In this project, there are six vhdl files as follows, uart_tx.vhd, uart_tx-fifo.vhd, uart_rx.vhd, uart_rx_fifo.vhd, uart_wb.vhd, uart_top.vhd. The design vision tool compiles and synthesizes all the six files and are saved as syn_2.vhdl. This synthesized file will be used as the reference (gate level design) for comparing with buggy file during bug hunting in Synopsys Formality and Cadence Conformal tools.

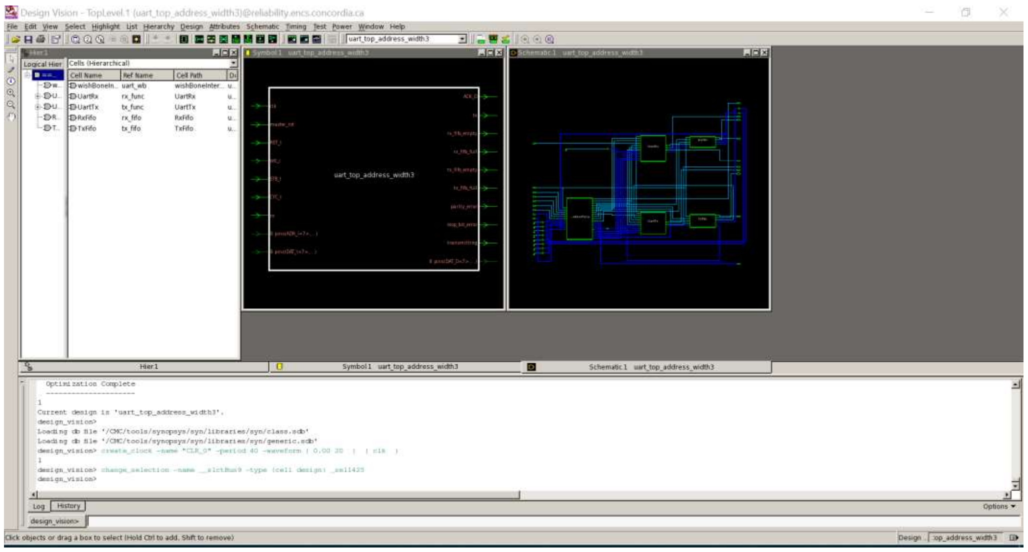
Steps in RTL Synthesis:
The following steps are used in Synopsys design vision tool,

a. First, read five sub files and then read the top file.
b. In Analyze, add the top file under "file names in analysis order" section.
c. Elaborate the design (give the required address_width). Here, address_width is given as 3. Then check design by selecting Design -> Check Design. Value 1 is returns which means the design check is executed successfully.
d. Select the top file and view symbol in logical hierarchy work plane, then select the clock pin and give the following clock specifications

Attribute Name	**Value**
Clock name	clk
Clock period	40
Raising	0.00
Falling	20

e. Finally, select Design -> Compile design.

The synthesis and optimization of design is completed successfully and value 1 is returned.


```
 Optimization Complete
 ---------------------
1
Current design is 'uart_top_address_width3'.
design_vision>
Loading db file '/CMC/tools/synopsys/syn/libraries/syn/class.sdb'
Loading db file '/CMC/tools/synopsys/syn/libraries/syn/generic.sdb'
design_vision> create_clock -name "CLK_0" -period 40 -waveform { 0.00 20 } { clk }
1
design_vision> change_selection -name __slctBus9 -type {cell design} _sel1425
design_vision>
```
Design Vision - TopLevel.1 (uart_top_address_width3)@reliability.encs.concordia.ca
File Edit View Select Highlight List Hierarchy Design Attributes Schematic Timing Test Power Window Help
Hier.1
Symbol.1 uart_top_address_width3
Schematic.1 uart_top_address_width3
Log History
Options
design_vision>
Click objects or drag a box to select (Hold Ctrl to add, Shift to remove)

Cell Name	Ref Name	Cell Path
wishBoneIn...	uart_wb	wishBoneInter...
UartRx	rx_func	UartRx
UartTx	tx_func	UartTx
RxFifo	rx_fifo	RxFifo
TxFifo	tx_fifo	TxFifo

Figure 5 : Symbol view and schematic diagram in Synopsys design vision

After performing synthesis, the file is saved as syn_2.vhdl (it is the gate level description of given RTL file). This synthesized file is used for equivalence checking and bud hunting using Synopsys and Conformal tool.

3. EQUIVALENCE CHECKING USING FORMALITY:

3.1. SYNOPSYS FORMALITY:

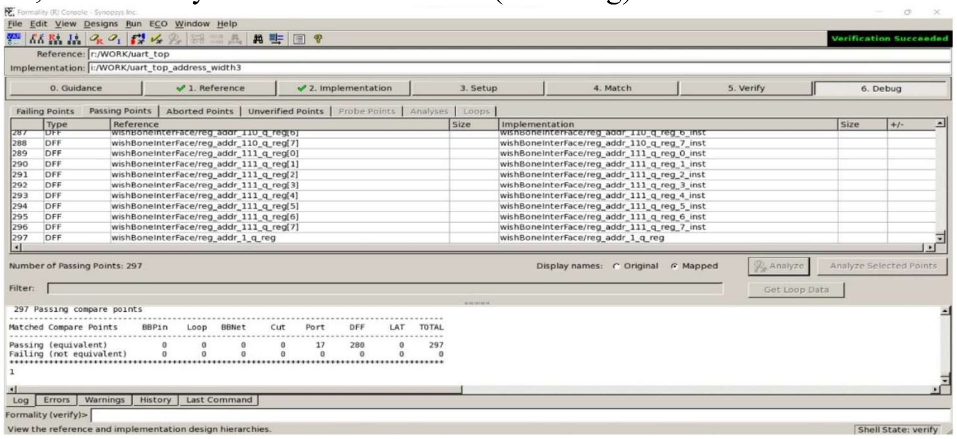
Formality is an equivalence-checking (EC) system that determines if two versions of a design are functionally similar using formal, static methodologies. The scale and complexity of today's designs, together with the problems of achieving time, area, power, and schedule, require highly verified synthesis optimizations. Formality supports all the Design Compiler and Fusion Compiler optimizations out of the box, resulting in the highest quality, completely verifiable outputs. Power-up and power-down states, multi-voltage, multi-supply, and clock-gated designs are all supported by formality. The flow-based graphical user interface and auto-setup mode of Formality make it simple for even inexperienced users to finish verification in the minimum time.
The auto-setup option in Formality makes verification easier by avoiding false failures caused by improper or missing setup parameters. Auto-setup uses Formality setup information to meet Design Compiler or Fusion Compiler assumptions, such as naming conventions, unused pins, test inputs, and clock gating. RTL, netlists, and libraries, among other important files, are automatically located.

3.2. RTL-GATE LEVEL EQUIVALENCE CHECKING:

It is critical to validate the legitimacy of the synthesized file by comparing it to the original RTL file before verifying the buggy file. So, the RTL file is loaded in the reference area, and the synthesis file (syn_2.vhdl) is loaded in the implementation area with the appropriate libraries and support files, followed by verification. The Synopsys Formality GUI will provide the result.

Steps involved in equivalence checking are as follows,

a. Firstly, setup the environment, the formality window will pop up.
b. Under "Reference" tab, read the RTL file (given design) and load [It is any design that need to be referred for doing comparison].
c. In the Read DB Libraries, select **class.db** and load it.
d. In the Set Top Design choose uart_top and select set top design and then set reference.
e. Check whether a green tick mark appears in the Reference tab.
f. Under "Implementation" tab, read the syn_2.vhdl file (synthesized file) and load [It is any file that must be compared with reference file for doing needed correction].
g. In the Read DB Libraries, select class.db and load it.
h. In the Set Top Design choose main and select set top design and then set reference.
i. Check whether a green tick mark appears near the Implementation tab.
j. Under "match" tab, select run matching.
k. Under "Verify" tab, select verify to see the verification(matching) result.



Reference: r:/WORK/uart_top
Implementation: i:/WORK/uart_top_address_width3

	Type	Reference	Size	Implementation	Size	+/-
287	DFF	wishBoneInterFace/reg_addr_110_q_reg[6]		wishBoneInterFace/reg_addr_110_q_reg_6_inst		
288	DFF	wishBoneInterFace/reg_addr_110_q_reg[7]		wishBoneInterFace/reg_addr_110_q_reg_7_inst		
289	DFF	wishBoneInterFace/reg_addr_111_q_reg[0]		wishBoneInterFace/reg_addr_111_q_reg_0_inst		
290	DFF	wishBoneInterFace/reg_addr_111_q_reg[1]		wishBoneInterFace/reg_addr_111_q_reg_1_inst		
291	DFF	wishBoneInterFace/reg_addr_111_q_reg[2]		wishBoneInterFace/reg_addr_111_q_reg_2_inst		
292	DFF	wishBoneInterFace/reg_addr_111_q_reg[3]		wishBoneInterFace/reg_addr_111_q_reg_3_inst		
293	DFF	wishBoneInterFace/reg_addr_111_q_reg[4]		wishBoneInterFace/reg_addr_111_q_reg_4_inst		
294	DFF	wishBoneInterFace/reg_addr_111_q_reg[5]		wishBoneInterFace/reg_addr_111_q_reg_5_inst		
295	DFF	wishBoneInterFace/reg_addr_111_q_reg[6]		wishBoneInterFace/reg_addr_111_q_reg_6_inst		
296	DFF	wishBoneInterFace/reg_addr_111_q_reg[7]		wishBoneInterFace/reg_addr_111_q_reg_7_inst		
297	DFF	wishBoneInterFace/reg_addr_1_q_reg		wishBoneInterFace/reg_addr_1_q_reg		

Number of Passing Points: 297
Display names: Original [✓ Mapped]
Filter:

```
 297 Passing compare points
-------------------------------------------------------------------------------
Matched Compare Points     BBPin    Loop   BBNet     Cut    Port     DFF     LAT   TOTAL
-------------------------------------------------------------------------------
Passing (equivalent)           0       0       0       0      17     280       0     297
Failing (not equivalent)       0       0       0       0       0       0       0       0
*******************************************************************************
1
```

Formality (verify)>
View the reference and implementation design hierarchies.
Shell State: verify

Figure 6 : RTL-gate level checking – Formality

As per the above design, there are no failing points between RTL and synthesized gate level design(syn_2.vhdl) and the verification is successful, indicating that our synthesized file accurate.

3.3. GATE-GATE LEVELEQUIVALENCE CHECKING:

In the reference tab, loaded the synthesized file (syn_2.vhdl). In the implementation tab, loaded the buggy file (uart_syn_buggy.vhdl) and repeated the same steps as above. If there exist any mismatches between two gate level design, the tool will show the failing point in GUI.

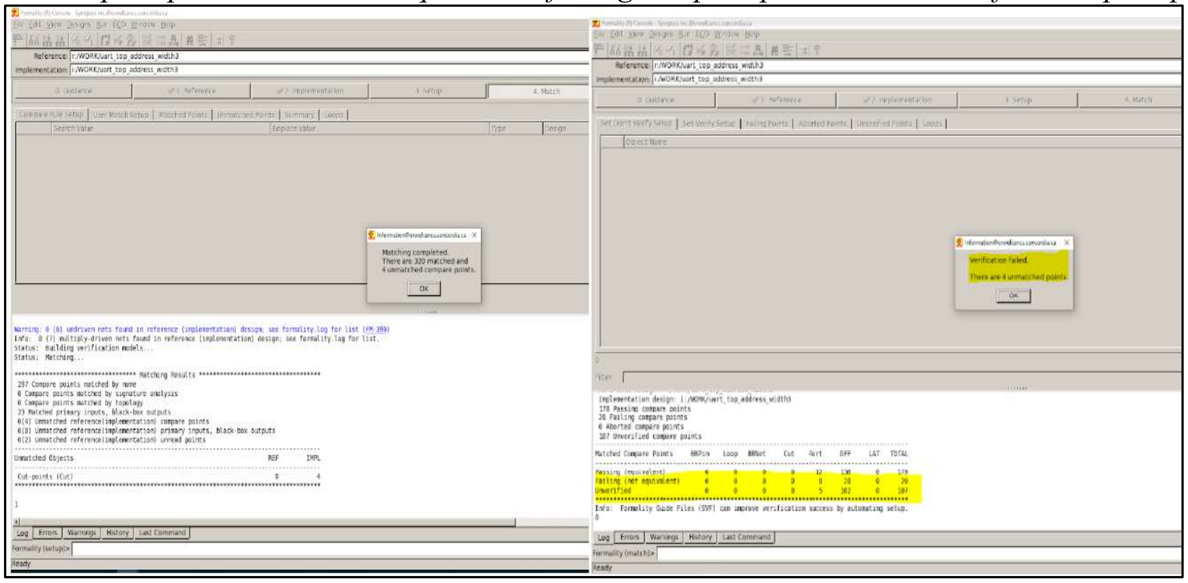
I. **Match report:**

320 matched points= 297 compare points by name **AND** 23 Matched by primary inputs

There were 4 unmatched (cutnet) points.

II. Verification report:

Equivalent compare points: 170, Non-Equivalent/ failing compare points: 20, Unverified compare points: 107



Reference: r:/WORK/uart_top_address_width3
Implementation: i:/WORK/uart_top_address_width3

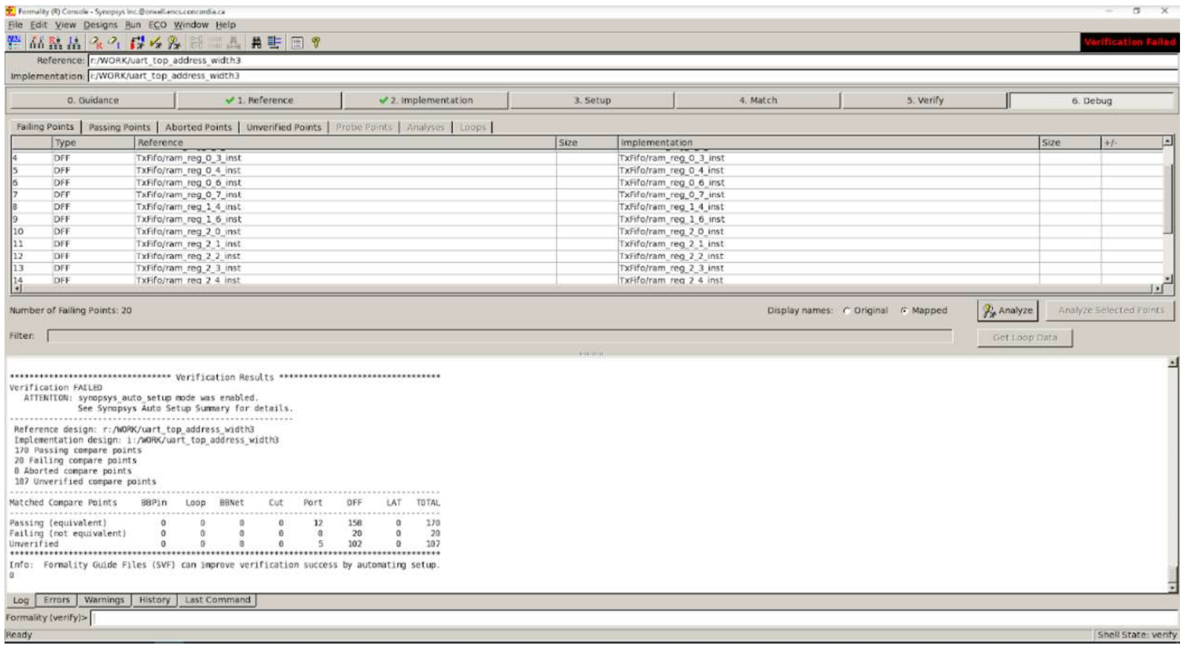
Matching completed.
There are 320 matched and
4 unmatched compare points.

```
Warning: 0 (6) undriven nets found in reference (implementation) design; see formality.log for list (FM-399)
Info:  0 (7) multiply-driven nets found in reference (implementation) design; see formality.log for list.
Status:  Building verification models...
Status:  Matching...

********************************* Matching Results *********************************
 297 Compare points matched by name
 0 Compare points matched by signature analysis
 0 Compare points matched by topology
 23 Matched primary inputs, black-box outputs
 0(4) Unmatched reference(implementation) compare points
 0(0) Unmatched reference(implementation) primary inputs, black-box outputs
 0(2) Unmatched reference(implementation) unread points
----------------------------------------------------------------------------------
Unmatched Objects                                                    REF       IMPL
----------------------------------------------------------------------------------
 Cut-points (Cut)                                                      0          4
**********************************************************************************

1
```

Verification failed
There are 4 unmatched points

```
 Implementation design: i:/WORK/uart_top_address_width3
 170 Passing compare points
 20 Failing compare points
 0 Aborted compare points
 107 Unverified compare points
----------------------------------------------------------------------------------
Matched Compare Points     BBPin    Loop   BBNet     Cut    Port     DFF     LAT   TOTAL
----------------------------------------------------------------------------------
Passing (equivalent)           0       0       0       0      12     158       0     170
Failing (not equivalent)       0       0       0       0       0      20       0      20
Unverified                     0       0       0       0       5     102       0     107
**********************************************************************************
Info:  Formality Guide Files (SVF) can improve verification success by automating setup.
0
```

Figure 7 : Gate-gate matching ang verification



Reference: r:/WORK/uart_top_address_width3
Implementation: i:/WORK/uart_top_address_width3

	Type	Reference	Size	Implementation	Size	+/-
4	DFF	TxFifo/ram_reg_0_3_inst		TxFifo/ram_reg_0_3_inst		
5	DFF	TxFifo/ram_reg_0_4_inst		TxFifo/ram_reg_0_4_inst		
6	DFF	TxFifo/ram_reg_0_6_inst		TxFifo/ram_reg_0_6_inst		
7	DFF	TxFifo/ram_reg_0_7_inst		TxFifo/ram_reg_0_7_inst		
8	DFF	TxFifo/ram_reg_1_4_inst		TxFifo/ram_reg_1_4_inst		
9	DFF	TxFifo/ram_reg_1_6_inst		TxFifo/ram_reg_1_6_inst		
10	DFF	TxFifo/ram_reg_2_0_inst		TxFifo/ram_reg_2_0_inst		
11	DFF	TxFifo/ram_reg_2_1_inst		TxFifo/ram_reg_2_1_inst		
12	DFF	TxFifo/ram_reg_2_2_inst		TxFifo/ram_reg_2_2_inst		
13	DFF	TxFifo/ram_reg_2_3_inst		TxFifo/ram_reg_2_3_inst		
14	DFF	TxFifo/ram_reg_2_4_inst		TxFifo/ram_reg_2_4_inst		

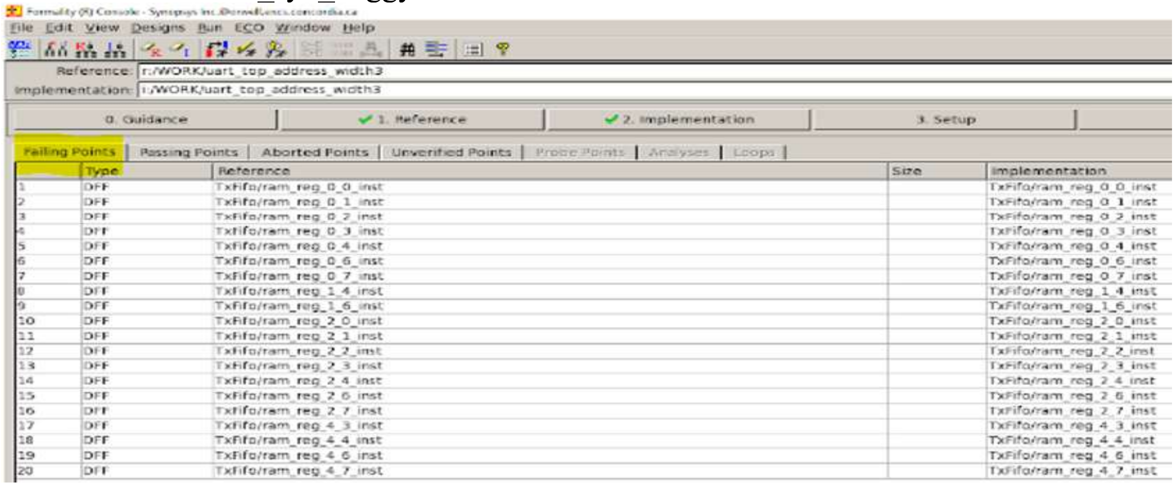
Number of Failing Points: 20

```
********************************* Verification Results *********************************
Verification FAILED
   ATTENTION: synopsys_auto_setup mode was enabled.
              See Synopsys Auto Setup Summary for details.
-----------------------------------------------------------------
 Reference design: r:/WORK/uart_top_address_width3
 Implementation design: i:/WORK/uart_top_address_width3
 170 Passing compare points
 20 Failing compare points
 0 Aborted compare points
 107 Unverified compare points
----------------------------------------------------------------------------------
Matched Compare Points     BBPin    Loop   BBNet     Cut    Port     DFF     LAT   TOTAL
----------------------------------------------------------------------------------
Passing (equivalent)           0       0       0       0      12     158       0     170
Failing (not equivalent)       0       0       0       0       0      20       0      20
Unverified                     0       0       0       0       5     102       0     107
**********************************************************************************
Info:  Formality Guide Files (SVF) can improve verification success by automating setup.
0
```

Figure 8 : Gate- gate level synthesis

3.4. DESCRIPTION OF VERIFICATION PROCESS IN SYNOPSYS FORMALITY:

Failing point tab shows the list verification failed points which need to be diagnosed. By analyzing the failing point, the bugs can be fixed in the uart_syn_buggy.vhdl .



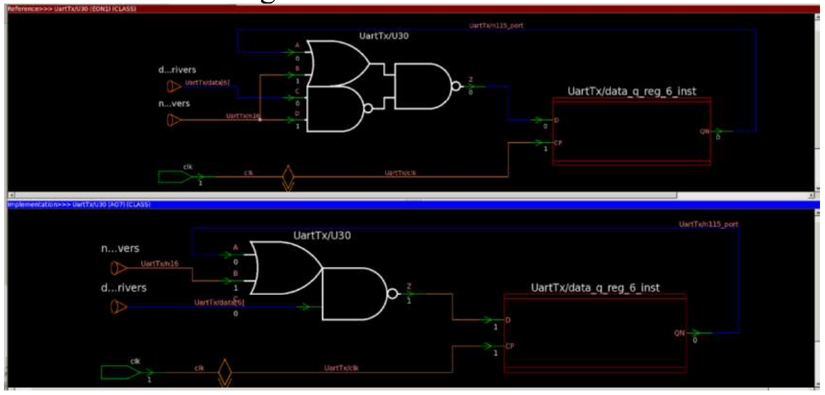
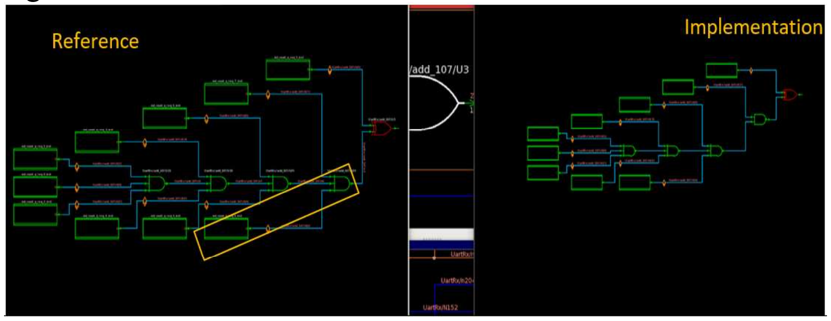
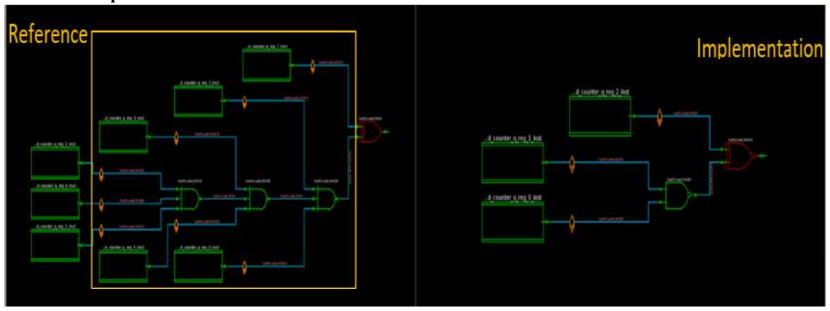
Reference: r:/WORK/uart_top_address_width3
Implementation: i:/WORK/uart_top_address_width3

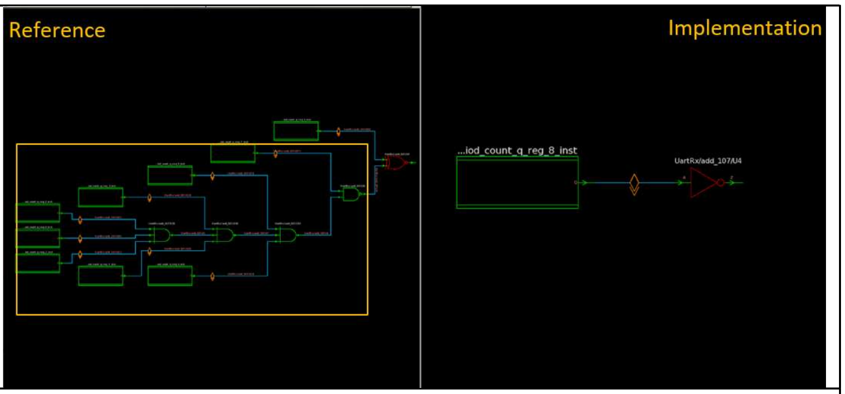
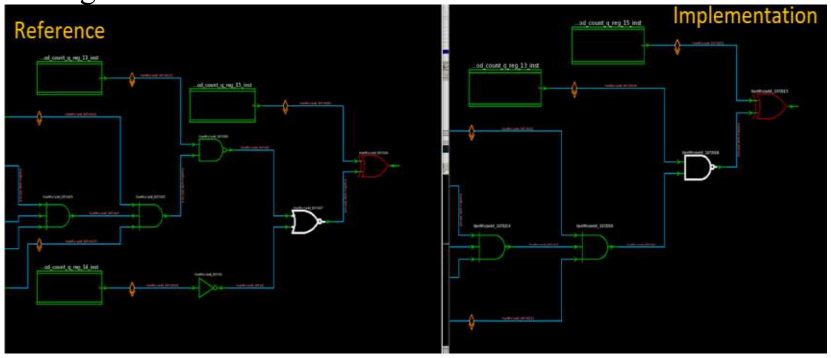
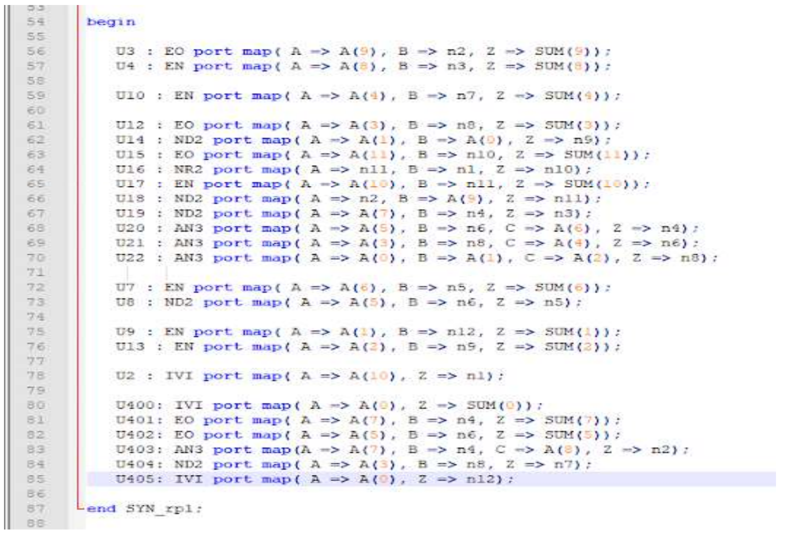
	Type	Reference	Size	Implementation
1	DFF	TxFifo/ram_reg_0_0_inst		TxFifo/ram_reg_0_0_inst
2	DFF	TxFifo/ram_reg_0_1_inst		TxFifo/ram_reg_0_1_inst
3	DFF	TxFifo/ram_reg_0_2_inst		TxFifo/ram_reg_0_2_inst
4	DFF	TxFifo/ram_reg_0_3_inst		TxFifo/ram_reg_0_3_inst
5	DFF	TxFifo/ram_reg_0_4_inst		TxFifo/ram_reg_0_4_inst
6	DFF	TxFifo/ram_reg_0_6_inst		TxFifo/ram_reg_0_6_inst
7	DFF	TxFifo/ram_reg_0_7_inst		TxFifo/ram_reg_0_7_inst
8	DFF	TxFifo/ram_reg_1_4_inst		TxFifo/ram_reg_1_4_inst
9	DFF	TxFifo/ram_reg_1_6_inst		TxFifo/ram_reg_1_6_inst
10	DFF	TxFifo/ram_reg_2_0_inst		TxFifo/ram_reg_2_0_inst
11	DFF	TxFifo/ram_reg_2_1_inst		TxFifo/ram_reg_2_1_inst
12	DFF	TxFifo/ram_reg_2_2_inst		TxFifo/ram_reg_2_2_inst
13	DFF	TxFifo/ram_reg_2_3_inst		TxFifo/ram_reg_2_3_inst
14	DFF	TxFifo/ram_reg_2_4_inst		TxFifo/ram_reg_2_4_inst
15	DFF	TxFifo/ram_reg_2_6_inst		TxFifo/ram_reg_2_6_inst
16	DFF	TxFifo/ram_reg_2_7_inst		TxFifo/ram_reg_2_7_inst
17	DFF	TxFifo/ram_reg_4_3_inst		TxFifo/ram_reg_4_3_inst
18	DFF	TxFifo/ram_reg_4_4_inst		TxFifo/ram_reg_4_4_inst
19	DFF	TxFifo/ram_reg_4_6_inst		TxFifo/ram_reg_4_6_inst
20	DFF	TxFifo/ram_reg_4_7_inst		TxFifo/ram_reg_4_7_inst

Figure 9 : Failing points

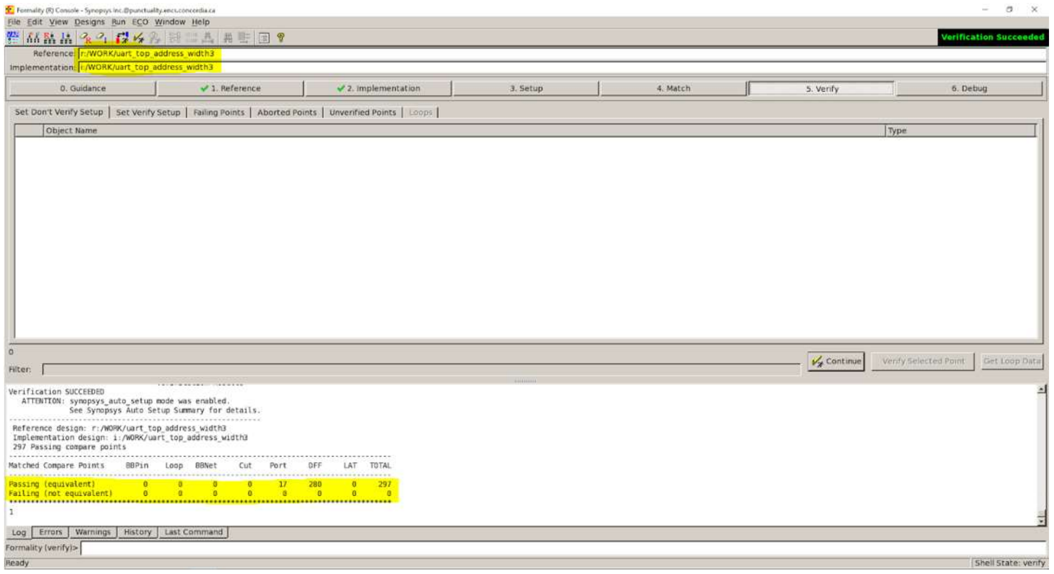
We started by fixing the failing points, which resulted in reducing the unverified compare points. We tabulated few in the table. By back-tracing the schematic diagram inputs and outputs between reference and implementation logic design, the changes are made in the code, which reduced the failing points.

BUG HUNTING		
S. No	**Bug location**	**Fix**
1	TxFifo/n2 **cutnet**	tx_entries_back_q_reg_3_inst: changed QN => n245 **to** QN => n2
2	TxFifo/tx_fifo_entries_free[0] **cutnet**	tx_entries_back_q_reg_0_inst: changed Q => tx_fifo_entries_free_1_port **to** Q => tx_fifo_entries_free_0_port
3	UartTx/baud_counter_q_reg_15_inst	Changed U81 **to** ND2
4	UartTx/baud_counter_q_reg_15_inst	U86: changed A04 **to** ND4
5	UartTx/baud_counter_q_reg_15_inst	U77: changed OR2P **to** EN
6	UartTx/period_count_q_reg_3_inst	U12: changed ND2 **to** XOR
7	UartTx/ period_count_q_reg_15_inst	No need for U16 NOT gate
8	UartTx/ period_count_q_reg_15_inst	U18: change from ND2 **to** NOR gate
9	DAT_O(2)	U128: changed AO3 **to** AO2
10	Tx_fifo/ tx_out_addr_q_reg_0_inst	Changed the output **to** Q => tx_out_addr_q_0_port

11	UartTx/ data_q_reg_6_inst	UartTX/ U30: Changed **to** EON1  UartTx/U30 d...rivers n...vers UartTx/data_q_reg_6_inst UartTx/n115_port clk UartTx/U30 n...vers d...rivers UartTx/data_q_reg_6_inst UartTx/n115_port clk
12	UartTX/ baud_counter_q_reg_7_inst	UartTx/add_84/U13: changed **to** XOR gate
13	UartTX/ baud_counter_q_reg_5_inst	UartTx/U51: changed from AO4 **to** EON1
14	UartRx/ period_count_q_reg_14_inst	UartRx/add_107/U18: changed **to** ND2
15	UartRx/ period16_count_q_reg_8_inst	UartRx/add_96/U4: changed **to** EN
16	UartRx/ period_count_q_reg_9_inst	period_count_q_reg_7_inst and period_count_q_reg_8_inst was not connected. So added 3 input NAND to add the missing signal. 
17	UartRx/ baud_counter_q_reg_9_inst	Added baud_counter_q_reg_8_inst **to** the baud_counter_q_reg_9_inst
18	UartRx/ baud_counter_q_reg_7_inst	Pulled the wire from the previous baud_counter ie from baud_counter_q_reg_6_inst and gave it to 3 i/p AND rather than 2 i/p NAND. 
19	UartRx/ period_count_q_reg_8_inst	Pulled the signal from period_count_q_reg_7_inst. The signal was pulled and reg_8_inst must fed to XNOR gate

		 Reference Implementation ...iod_count_q_reg_8_inst UartRx/add_107/U4
20	UartRx/ period_count_q_reg_15_inst	The output from UartRx/ad_107/U18 and inverter, fed it to XOR gate. 
21	TxFifo/tx_entries_back_q_req_2_inst TxFifo/tx_out_addr_q_req_2_inst	Port mismatch
22	UartRx/ period_count_q_reg_12_inst	UartRX/U21 change from AN3 to XNOR by getting the output only from reg_12 and ND2
23-35	UartRx/ period16_count_q_reg_1_inst UartRx/ period16_count_q_reg_2_inst UartRx/ period16_count_q_reg_3_inst UartRx/ period16_count_q_reg_4_inst UartRx/ period16_count_q_reg_5_inst UartRx/ period16_count_q_reg_6_inst UartRx/ period16_count_q_reg_7_inst UartRx/ period16_count_q_reg_8_inst UartRx/ period16_count_q_reg_9_inst UartRx/ period16_count_q_reg_10_inst UartRx/ period16_count_q_reg_11_inst	For all errors related to UartRx/period16_count_q_reg_X_inst and cutnet n6 and n12, we did some direct changes in the buggy file after understanding the behavior of the system. 

```
begin

    U3  : EO port map( A => A(9), B => n2, Z => SUM(9));
    U4  : EN port map( A => A(8), B => n3, Z => SUM(8));

    U10  : EN port map( A => A(4), B => n7, Z => SUM(4));

    U12  : EO port map( A => A(3), B => n8, Z => SUM(3));
    U14  : ND2 port map( A => A(1), B => A(0), Z => n9);
    U15  : EO port map( A => A(11), B => n10, Z => SUM(11));
    U16  : NR2 port map( A => n11, B => n1, Z => n10);
    U17  : EN port map( A => A(10), B => n11, Z => SUM(10));
    U18  : ND2 port map( A => n2, B => A(9), Z => n11);
    U19  : ND2 port map( A => A(7), B => n4, Z => n3);
    U20  : AN3 port map( A => A(5), B => n6, C => A(6), Z => n4);
    U21  : AN3 port map( A => A(3), B => n8, C => A(4), Z => n6);
    U22  : AN3 port map( A => A(0), B => A(1), C => A(2), Z => n8);

    U7  : EN port map( A => A(6), B => n5, Z => SUM(6));
    U8  : ND2 port map( A => A(5), B => n6, Z => n5);

    U9  : EN port map( A => A(1), B => n12, Z => SUM(1));
    U13  : EN port map( A => A(2), B => n9, Z => SUM(2));

    U2  : IVI port map( A => A(10), Z => n1);

    U400: IVI port map( A => A(0), Z => SUM(0));
    U401: EO port map( A => A(7), B => n4, Z => SUM(7));
    U402: EO port map( A => A(5), B => n6, Z => SUM(5));
    U403: AN3 port map(A => A(7), B => n4, C => A(8), Z => n2);
    U404: ND2 port map( A => A(3), B => n8, Z => n7);
    U405: IVI port map( A => A(0), Z => n12);

end SYN_rpl;
```



```
Reference:      r:/WORK/uart_top_address_width3
Implementation: i:/WORK/uart_top_address_width3
```

```
Verification SUCCEEDED
   ATTENTION: synopsys_auto_setup mode was enabled.
              See Synopsys Auto Setup Summary for details.
--------------------------------------------------------
 Reference design: r:/WORK/uart_top_address_width3
 Implementation design: i:/WORK/uart_top_address_width3
 297 Passing compare points
------------------------------------------------------------------------------------------
Matched Compare Points     BBPin    Loop   BBNet     Cut    Port     DFF     LAT   TOTAL
------------------------------------------------------------------------------------------
Passing (equivalent)           0       0       0       0      17     280       0     297
Failing (not equivalent)       0       0       0       0       0       0       0       0
******************************************************************************************
1
Formality (verify)>
```

Figure 10:Gate -gate level synthesis after bug hunting

By adopting the above tabulated bug hunting techniques, all the errors are resolved, and the verification is successful with 0 failing points.

4. EQUIVALENCE CHECKING USING CONFORMAL

The Cadence Conformal Equivalence Checker (EC) can validate and debug multi-million–gate designs without using test vectors. Conformal is a tool developed by Cadence, a well-known corporation that is a pioneer in the field of electrical design. Designers may use Cadence Conformal EC to test a wide range of circuits, including complicated arithmetic logic, data routes, memory, and custom logic. Conformal EC is the industry's most extensively supported independent equivalency checking solution, having been validated in thousands of tape outs. It's important to remember that RTL is also used at the gate level, in addition to RTL.

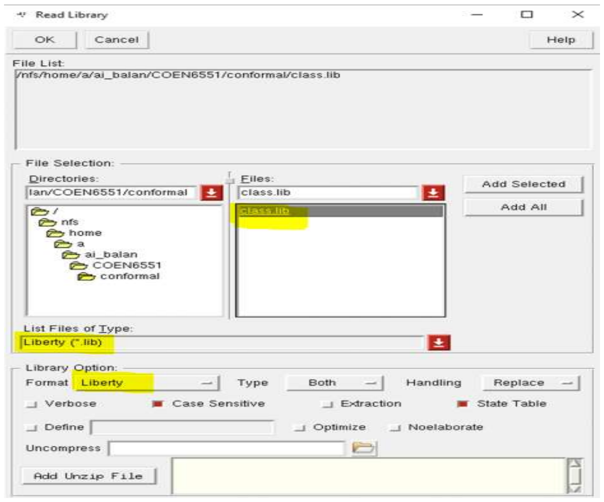
Cadence supports the following verification languages and standards:

- Formats VerilogR, VHDL, NDL, EDIF and SPICE (traditional, LVS).
- Standard Library Formats “Verilog simulation libraries and the SynopsysR LibertyTM Format Libraries.

The advantages listed below are the reasons why this tool was chosen and used. Even though the test vectors can be removed, millions of gate designs can be verified and debugged. It is less probable that the primary problems will go unnoticed when verification technology is used. It also ensures that errors are detected and corrected in a reliable and timely manner throughout the design process. The RTL model, in addition to the capabilities listed above, performs, and fulfils the same objective as the transistor circuit. Conformal LEC is an extremely effective equivalency checking method. It can offer formal verification that the Synthesized output matches the source RTL code. It can achieve all of this without running a single simulation.

4.1. RTL-GATE LEVEL EQUIVALENCE CHECKING

There are two different designs in Conformal, namely, ‘Golden’ and ‘Revised’ design. The first phase of the Conformal LEC is the setup phase followed by the LEC mode. The setup phase consists of reading the Library and the VHDL files (golden and revised design). Golden design is loaded by selecting Read design and select VHDL as the file format, then choose the reference design file (RTL file). Also, it is important to include the library file at the beginning with proper file format. Here, Liberty must be chosen since the format of designs might be Verilog, VHDL, SPICE, NDL and EDIF. In the Revised design the format is the same as Golden which is VHDL but the synthesized file (syn_2.vhdl) has been opted. The setup mode is completed.

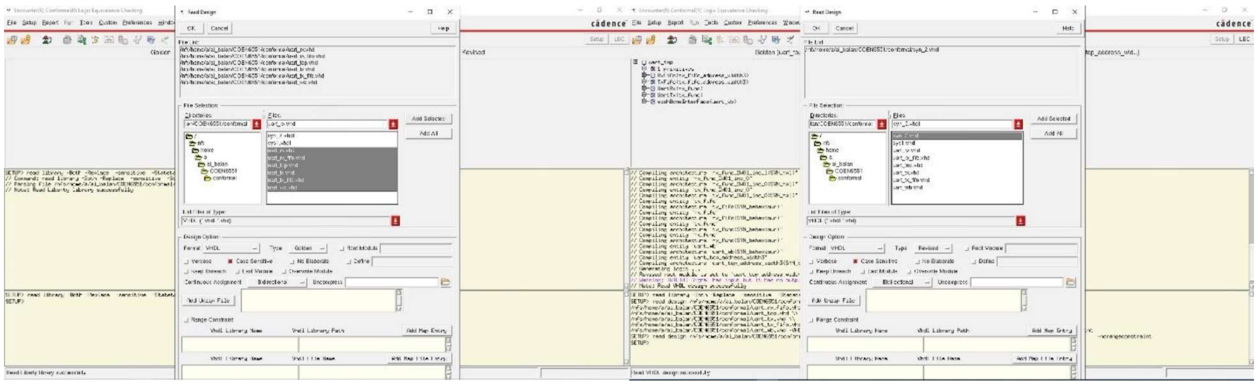


File List:
/nfs/home/a/ai_balan/COEN6551/conformal/class.lib

File Selection:
Directories: lan/COEN6551/conformal
Files: class.lib
List Files of Type: Liberty (*.lib)

Library Option:
Format Liberty   Type Both   Handling Replace
Verbose   Case Sensitive [✓]   Extraction   State Table [✓]
Define   Optimize   Noelaborate
Uncompress

Figure 11: Library file setup

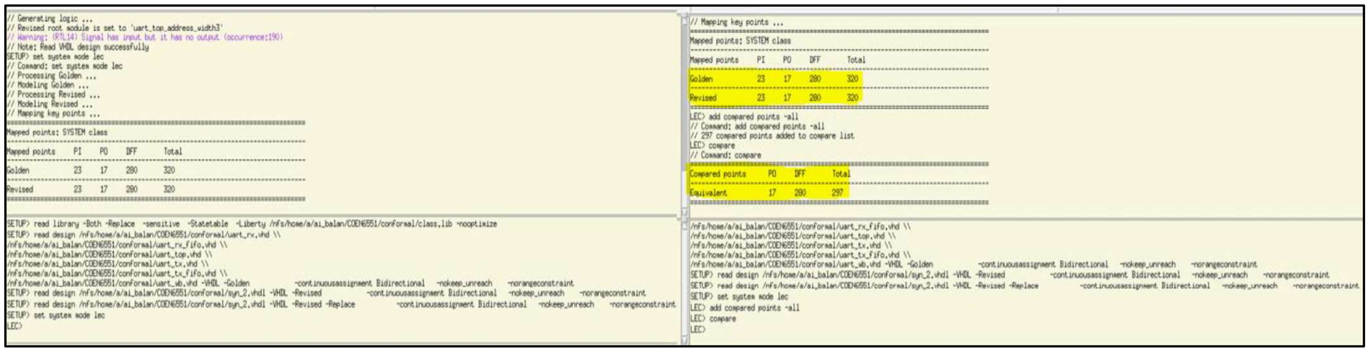


```
SETUP> read library -Both -Replace -sensitive -Statet
// Command: read library -Both -Replace -sensitive -S
// Parsing file /nfs/home/a/ai_balan/COEN6551/conformal
// Note: Read Liberty library successfully
```

Figure 12: RTL-GATE level Golden and revised setup

To map critical locations and compare them, the tool must be switched to LEC mode. After that, we can see result by clicking on Run on the toolbar and selecting Compare. There are three sections: Mapped points, Golden points, and Revised points, which include Primary Outputs, Primary Inputs, and D Flip-Flops. After comparing (compare all), Equivalent and Non-equivalent points will appear.

Critical locations such as primary inputs (PI), primary outputs (PO), DFFs, D-latches, blackboxes, Z-gates, and cut gates are instantly **mapped.** The tool begins with name-based mapping and then moves on to function-based mapping. Name-based mapping requires less time to execute than function-based mapping, the tool first attempts to map critical points using name-based mapping before moving to function-based mapping**.** Only the mapped key points will be compared. The tool will attempt to show equivalence by feeding all possible input combinations into the logic cone and observing the output behavior. The tool will then categorize the compared points as equivalent, inverted equivalent, non-equivalent, and abort points after the comparison procedure. Except equivalent points everything must be debugged. If all the comparison points are equal, we conclude that the golden and revised design are compatible and bug free. To perform comparison, select **Run → Compare**. Then, add all compare points.



```
// Generating logic ...
// Revised root module is set to 'uart_top_address_width3'
// Warning: (RTL14) Signal has input but it has no output (occurrence:190)
// Note: Read VHDL design successfully
SETUP> set system mode lec
// Command: set system mode lec
// Processing Golden ...
// Modeling Golden ...
// Processing Revised ...
// Modeling Revised ...
// Mapping key points ...
================================================================================
Mapped points: SYSTEM class
--------------------------------------------------------------------------------
Mapped points     PI     PO     DFF       Total
--------------------------------------------------------------------------------
Golden            23     17     280       320
--------------------------------------------------------------------------------
Revised           23     17     280       320
================================================================================

SETUP> read library -Both -Replace  -sensitive  -Statetable  -Liberty /nfs/home/a/ai_balan/COEN6551/conformal/class.lib -nooptimize
SETUP> read design /nfs/home/a/ai_balan/COEN6551/conformal/uart_rx.vhd \\
/nfs/home/a/ai_balan/COEN6551/conformal/uart_rx_fifo.vhd \\
/nfs/home/a/ai_balan/COEN6551/conformal/uart_top.vhd \\
/nfs/home/a/ai_balan/COEN6551/conformal/uart_tx.vhd \\
/nfs/home/a/ai_balan/COEN6551/conformal/uart_tx_fifo.vhd \\
/nfs/home/a/ai_balan/COEN6551/conformal/uart_wb.vhd -VHDL -Golden          -continuousassignment Bidirectional   -nokeep_unreach    -norangeconstraint
SETUP> read design /nfs/home/a/ai_balan/COEN6551/conformal/syn_2.vhdl -VHDL -Revised          -continuousassignment Bidirectional   -nokeep_unreach    -norangeconstraint
SETUP> read design /nfs/home/a/ai_balan/COEN6551/conformal/syn_2.vhdl -VHDL -Revised -Replace          -continuousassignment Bidirectional   -nokeep_unreach    -norangeconstraint
SETUP> set system mode lec
LEC>
```

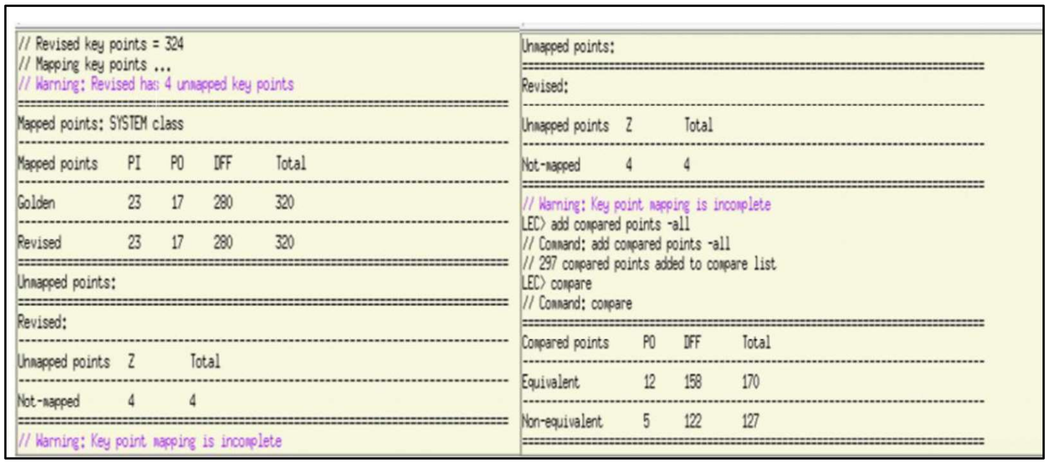
```
// Mapping key points ...
================================================================================
Mapped points: SYSTEM class
--------------------------------------------------------------------------------
Mapped points     PI     PO     DFF       Total
--------------------------------------------------------------------------------
Golden            23     17     280       320
--------------------------------------------------------------------------------
Revised           23     17     280       320
================================================================================
LEC> add compared points -all
// Command: add compared points -all
// 297 compared points added to compare list
LEC> compare
// Command: compare
================================================================================
Compared points      PO     DFF       Total
--------------------------------------------------------------------------------
Equivalent           17     280       297
================================================================================

/nfs/home/a/ai_balan/COEN6551/conformal/uart_rx_fifo.vhd \\
/nfs/home/a/ai_balan/COEN6551/conformal/uart_top.vhd \\
/nfs/home/a/ai_balan/COEN6551/conformal/uart_tx.vhd \\
/nfs/home/a/ai_balan/COEN6551/conformal/uart_tx_fifo.vhd \\
/nfs/home/a/ai_balan/COEN6551/conformal/uart_wb.vhd -VHDL -Golden          -continuousassignment Bidirectional   -nokeep_unreach    -norangeconstraint
SETUP> read design /nfs/home/a/ai_balan/COEN6551/conformal/syn_2.vhdl -VHDL -Revised          -continuousassignment Bidirectional   -nokeep_unreach    -norangeconstraint
SETUP> read design /nfs/home/a/ai_balan/COEN6551/conformal/syn_2.vhdl -VHDL -Revised -Replace          -continuousassignment Bidirectional   -nokeep_unreach    -norangeconstraint
SETUP> set system mode lec
LEC> add compared points -all
LEC> compare
LEC>
```

Figure 13: RTL-Gate level equivalence checking

Figure 13 shows the RTL-Gate level equivalence checking. There are no failing point or unmatched point. So, the synthesized file is accurate to use it for comparing with buggy gate level file for doing error correction.

4.2. GATE-GATE LEVEL EQUIVALENCE CHECKING

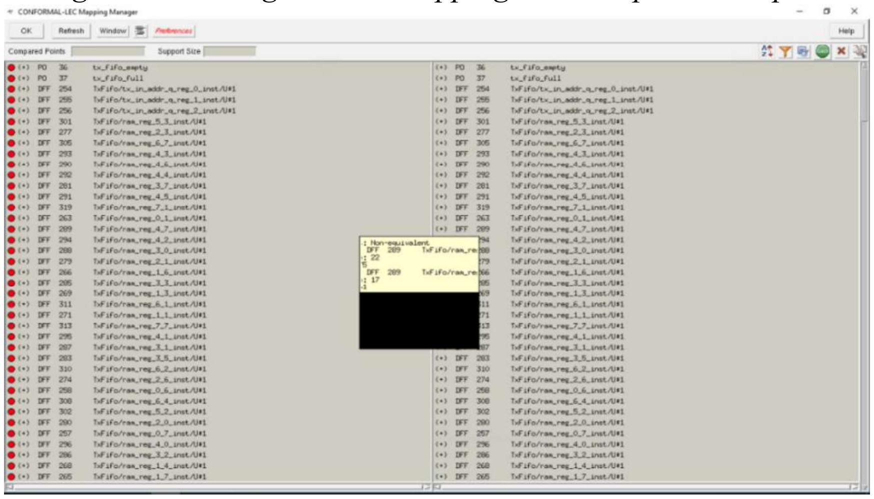
The synthesized file (syn_2.vhdl) is loaded in Golden design and buggy gate level file (uart_syn_buggy.vhdl) is loaded in revised design. The same steps are followed as in RTL-Gate level equivalence checking. There are 4 unmapped points and 127 non-equivalent points after comparing as shown in the figure 14.



```
// Revised key points = 324
// Mapping key points ...
// Warning: Revised has 4 unmapped key points
================================================================================
Mapped points: SYSTEM class
--------------------------------------------------------------------------------
Mapped points     PI     PO     DFF       Total
--------------------------------------------------------------------------------
Golden            23     17     280       320
--------------------------------------------------------------------------------
Revised           23     17     280       320
================================================================================
Unmapped points:
================================================================================
Revised:
--------------------------------------------------------------------------------
Unmapped points   Z          Total
--------------------------------------------------------------------------------
Not-mapped        4          4
================================================================================
// Warning: Key point mapping is incomplete
```

```
Unmapped points:
================================================================================
Revised:
--------------------------------------------------------------------------------
Unmapped points   Z          Total
--------------------------------------------------------------------------------
Not-mapped        4          4
================================================================================
// Warning: Key point mapping is incomplete
LEC> add compared points -all
// Command: add compared points -all
// 297 compared points added to compare list
LEC> compare
// Command: compare
================================================================================
Compared points      PO     DFF       Total
--------------------------------------------------------------------------------
Equivalent           12     158       170
--------------------------------------------------------------------------------
Non-equivalent       5      122       127
================================================================================
```

Figure 14: Gate-gate level Mapping and Comparison Report



```
● (+)  PO   36    tx_fifo_empty
● (+)  PO   37    tx_fifo_full
● (+)  DFF  254   TxFifo/tx_in_addr_q_reg_0_inst/U#1
● (+)  DFF  255   TxFifo/tx_in_addr_q_reg_1_inst/U#1
● (+)  DFF  256   TxFifo/tx_in_addr_q_reg_2_inst/U#1
● (+)  DFF  301   TxFifo/ram_reg_5_3_inst/U#1
● (+)  DFF  277   TxFifo/ram_reg_2_3_inst/U#1
● (+)  DFF  305   TxFifo/ram_reg_6_7_inst/U#1
● (+)  DFF  293   TxFifo/ram_reg_4_3_inst/U#1
● (+)  DFF  290   TxFifo/ram_reg_4_6_inst/U#1
● (+)  DFF  292   TxFifo/ram_reg_4_4_inst/U#1
● (+)  DFF  281   TxFifo/ram_reg_3_7_inst/U#1
● (+)  DFF  291   TxFifo/ram_reg_4_5_inst/U#1
● (+)  DFF  319   TxFifo/ram_reg_7_1_inst/U#1
● (+)  DFF  263   TxFifo/ram_reg_0_1_inst/U#1
● (+)  DFF  289   TxFifo/ram_reg_4_7_inst/U#1
● (+)  DFF  294   TxFifo/ram_reg_4_2_inst/U#1
● (+)  DFF  288   TxFifo/ram_reg_3_0_inst/U#1
● (+)  DFF  279   TxFifo/ram_reg_2_1_inst/U#1
● (+)  DFF  266   TxFifo/ram_reg_1_6_inst/U#1
● (+)  DFF  285   TxFifo/ram_reg_3_3_inst/U#1
● (+)  DFF  269   TxFifo/ram_reg_1_3_inst/U#1
● (+)  DFF  311   TxFifo/ram_reg_6_1_inst/U#1
● (+)  DFF  271   TxFifo/ram_reg_1_1_inst/U#1
● (+)  DFF  313   TxFifo/ram_reg_7_7_inst/U#1
● (+)  DFF  295   TxFifo/ram_reg_4_1_inst/U#1
● (+)  DFF  287   TxFifo/ram_reg_3_1_inst/U#1
● (+)  DFF  283   TxFifo/ram_reg_3_5_inst/U#1
● (+)  DFF  310   TxFifo/ram_reg_6_2_inst/U#1
● (+)  DFF  274   TxFifo/ram_reg_2_6_inst/U#1
● (+)  DFF  258   TxFifo/ram_reg_0_6_inst/U#1
● (+)  DFF  308   TxFifo/ram_reg_6_4_inst/U#1
● (+)  DFF  302   TxFifo/ram_reg_5_2_inst/U#1
● (+)  DFF  280   TxFifo/ram_reg_2_0_inst/U#1
● (+)  DFF  257   TxFifo/ram_reg_0_7_inst/U#1
● (+)  DFF  296   TxFifo/ram_reg_4_0_inst/U#1
● (+)  DFF  286   TxFifo/ram_reg_3_2_inst/U#1
● (+)  DFF  268   TxFifo/ram_reg_1_4_inst/U#1
● (+)  DFF  265   TxFifo/ram_reg_1_7_inst/U#1
```

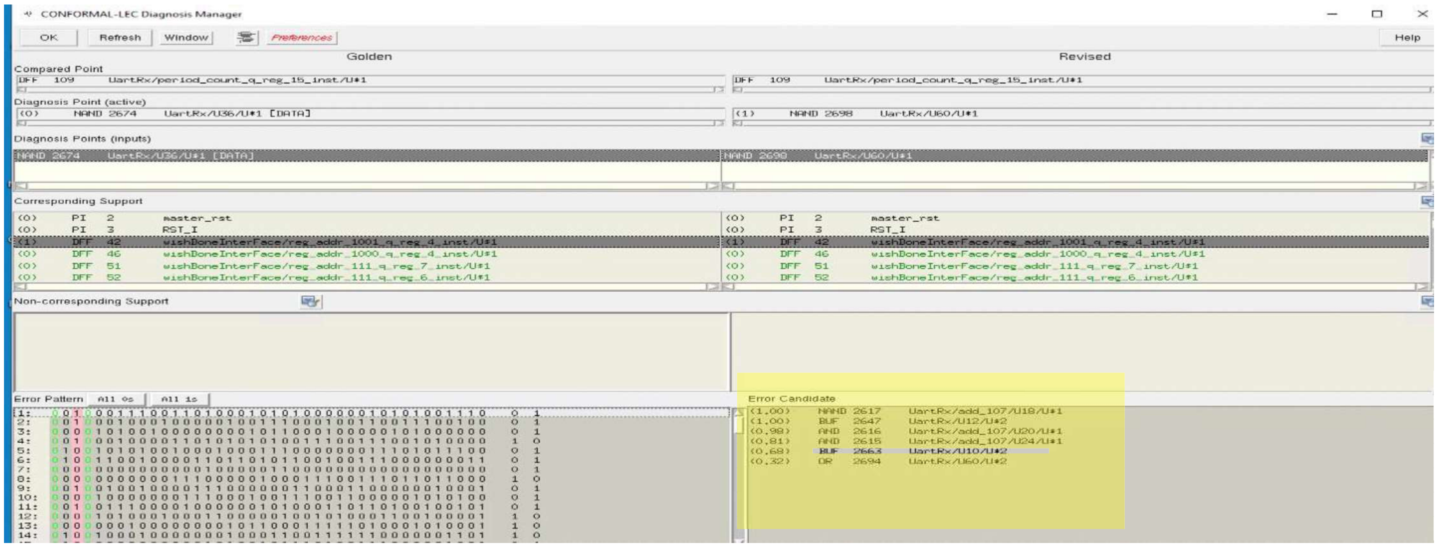
```
: Non-equivalent
 DFF  289     TxFifo/ram_re
: 22
5
 DFF  289     TxFifo/ram_re
: 17
-1
```

```
(+)  PO   36    tx_fifo_empty
(+)  PO   37    tx_fifo_full
(+)  DFF  254   TxFifo/tx_in_addr_q_reg_0_inst/U#1
(+)  DFF  255   TxFifo/tx_in_addr_q_reg_1_inst/U#1
(+)  DFF  256   TxFifo/tx_in_addr_q_reg_2_inst/U#1
(+)  DFF  301   TxFifo/ram_reg_5_3_inst/U#1
(+)  DFF  277   TxFifo/ram_reg_2_3_inst/U#1
(+)  DFF  305   TxFifo/ram_reg_6_7_inst/U#1
(+)  DFF  293   TxFifo/ram_reg_4_3_inst/U#1
(+)  DFF  290   TxFifo/ram_reg_4_6_inst/U#1
(+)  DFF  292   TxFifo/ram_reg_4_4_inst/U#1
(+)  DFF  281   TxFifo/ram_reg_3_7_inst/U#1
(+)  DFF  291   TxFifo/ram_reg_4_5_inst/U#1
(+)  DFF  319   TxFifo/ram_reg_7_1_inst/U#1
(+)  DFF  263   TxFifo/ram_reg_0_1_inst/U#1
(+)  DFF  289   TxFifo/ram_reg_4_7_inst/U#1
            94   TxFifo/ram_reg_4_2_inst/U#1
            88   TxFifo/ram_reg_3_0_inst/U#1
            79   TxFifo/ram_reg_2_1_inst/U#1
            66   TxFifo/ram_reg_1_6_inst/U#1
            85   TxFifo/ram_reg_3_3_inst/U#1
            69   TxFifo/ram_reg_1_3_inst/U#1
            11   TxFifo/ram_reg_6_1_inst/U#1
            71   TxFifo/ram_reg_1_1_inst/U#1
            13   TxFifo/ram_reg_7_7_inst/U#1
            95   TxFifo/ram_reg_4_1_inst/U#1
            87   TxFifo/ram_reg_3_1_inst/U#1
(+)  DFF  283   TxFifo/ram_reg_3_5_inst/U#1
(+)  DFF  310   TxFifo/ram_reg_6_2_inst/U#1
(+)  DFF  274   TxFifo/ram_reg_2_6_inst/U#1
(+)  DFF  258   TxFifo/ram_reg_0_6_inst/U#1
(+)  DFF  308   TxFifo/ram_reg_6_4_inst/U#1
(+)  DFF  302   TxFifo/ram_reg_5_2_inst/U#1
(+)  DFF  280   TxFifo/ram_reg_2_0_inst/U#1
(+)  DFF  257   TxFifo/ram_reg_0_7_inst/U#1
(+)  DFF  296   TxFifo/ram_reg_4_0_inst/U#1
(+)  DFF  286   TxFifo/ram_reg_3_2_inst/U#1
(+)  DFF  268   TxFifo/ram_reg_1_4_inst/U#1
(+)  DFF  265   TxFifo/ram_reg_1_7_inst/U#1
```

Figure 15: List of Non-equivalent points

4.3. DESCRIPTION OF VERIFICATION PROCESS IN CADENCE CONFORMAL:

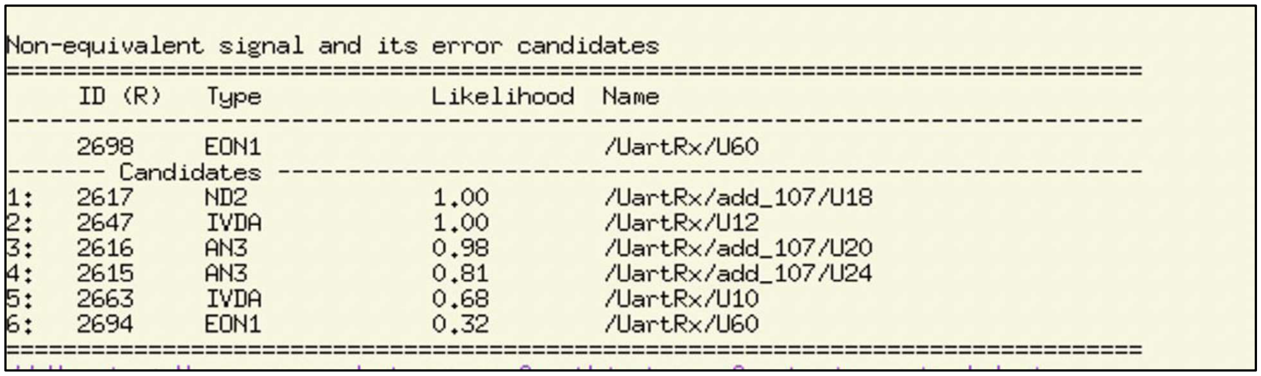
To resolve the bugs, select any one non-equivalent point in the mapping manager icon. Right click on the bug and click on diagnose option, the diagnose manager dialog box will open indicating all the error candidates. The error must be corrected in such a way that the candidate that has higher level of percentage error must be given priority. From the figure 16 below, the gates which have percentage as 1 must be debugged first. Once this is done all other errors which lower percentage will be corrected.



```
CONFORMAL-LEC Diagnosis Manager
OK   Refresh   Window   Preferences                                   Help
                 Golden                                              Revised
Compared Point
DFF   109     UartRx/period_count_q_reg_15_inst/U#1          DFF   109     UartRx/period_count_q_reg_15_inst/U#1
Diagnosis Point (active)
(0)      NAND 2674    UartRx/U36/U#1 [DATA]                  (1)      NAND 2698    UartRx/U60/U#1
Diagnosis Points (inputs)
NAND 2674    UartRx/U36/U#1 [DATA]                           NAND 2698    UartRx/U60/U#1
Corresponding Support
(0)   PI   2    master_rst                                   (0)   PI   2    master_rst
(0)   PI   3    RST_I                                        (0)   PI   3    RST_I
(1)   DFF  42   wishBoneInterFace/reg_addr_1001_q_reg_4_inst/U#1   (1)   DFF  42   wishBoneInterFace/reg_addr_1001_q_reg_4_inst/U#1
(0)   DFF  46   wishBoneInterFace/reg_addr_1000_q_reg_4_inst/U#1   (0)   DFF  46   wishBoneInterFace/reg_addr_1000_q_reg_4_inst/U#1
(0)   DFF  51   wishBoneInterFace/reg_addr_111_q_reg_7_inst/U#1    (0)   DFF  51   wishBoneInterFace/reg_addr_111_q_reg_7_inst/U#1
(0)   DFF  52   wishBoneInterFace/reg_addr_111_q_reg_6_inst/U#1    (0)   DFF  52   wishBoneInterFace/reg_addr_111_q_reg_6_inst/U#1
Non-corresponding Support

Error Pattern   All 0s   All 1s                              Error Candidate
1:   0 0 1 0 0 0 1 1 1 0 0 1 1 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 1 0 1 0 1 0 0 1 1 1 0   0 1    (1,00)   NAND 2617   UartRx/add_107/U18/U#1
2:   0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 1 1 1 0 0 0 1 0 0 1 1 0 0 1 1 1 0 0 1 0 0     0 1    (1,00)   BUF  2647   UartRx/U12/U#2
3:   0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0   0 1    (0,98)   AND  2616   UartRx/add_107/U20/U#1
4:   0 0 1 0 0 0 1 0 0 0 0 1 1 0 1 0 1 0 1 0 1 0 0 1 1 1 0 0 1 1 1 0 0 1 0 1 0 0 0 0     1 0    (0,81)   AND  2615   UartRx/add_107/U24/U#1
5:   0 1 0 0 1 0 1 0 1 0 0 1 0 0 0 1 0 0 0 1 1 1 0 0 0 0 0 0 0 1 1 1 0 1 0 1 1 1 0 0     0 1    (0,68)   BUF  2663   UartRx/U10/U#2
6:   0 1 0 0 1 1 0 0 1 0 0 0 0 1 1 0 1 1 0 1 0 1 1 0 0 1 0 0 1 1 1 0 0 0 0 0 0 0 1 1     0 1    (0,32)   OR   2694   UartRx/U60/U#2
7:   0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0     0 1
8:   0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 0 0 0 1 1 1 0 0 1 1 1 0 1 1 0 1 1 0 0 0     1 0
9:   0 0 1 0 0 1 0 0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 1     0 1
10:  0 0 0 0 1 0 0 0 0 0 0 0 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 1 0 0 0 0 0 1 0 1 0 1 0 0     0 1
11:  0 0 1 0 0 1 1 1 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 1 1 0 1 1 0 1 0 0 1 0 0 1 0 1     0 1
12:  0 0 0 0 1 0 1 0 0 0 1 0 0 0 1 1 0 0 0 0 0 1 0 0 1 0 1 0 0 0 1 1 0 0 1 0 0 0 0 1     1 0
13:  0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 1 1 1 1 1 0 1 0 0 0 1 0 1 0 0 0 1   1 0
14:  0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 1 1 1 1 1 1 0 0 0 0 0 0 1 1 0 1   1 0
```

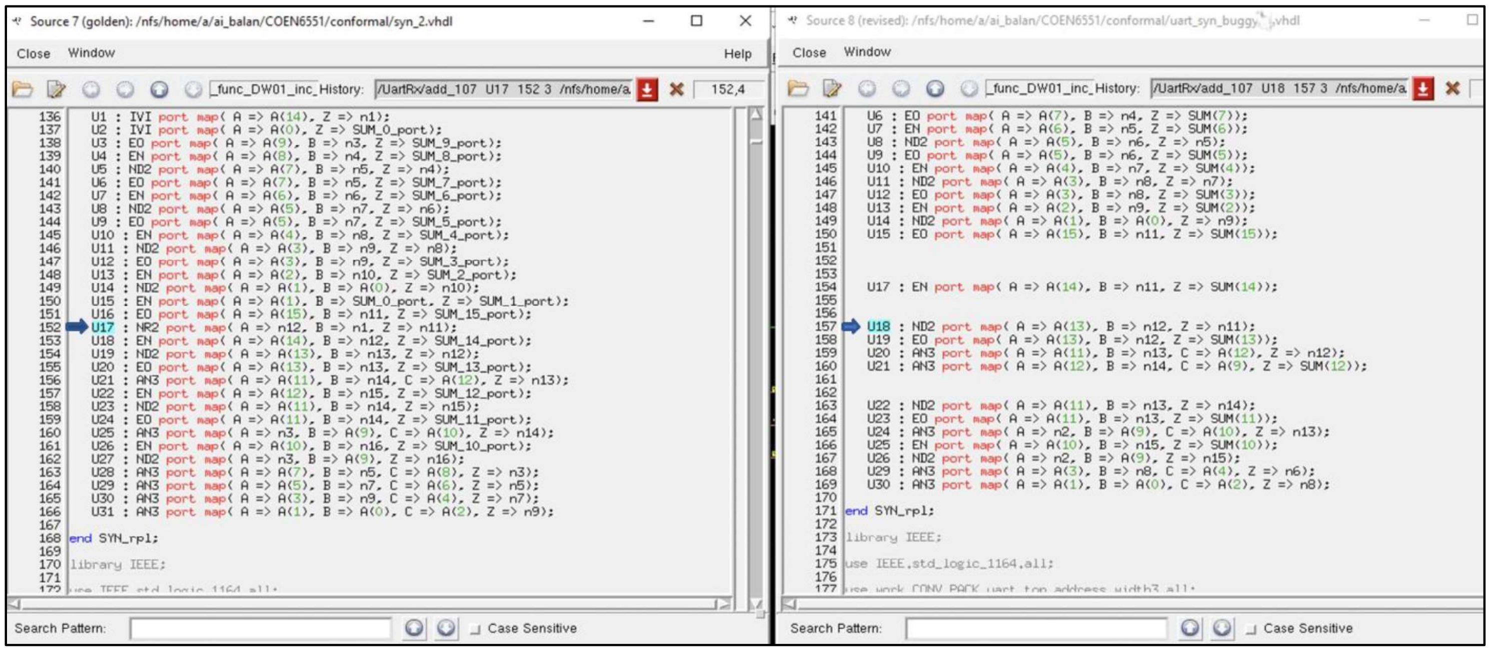
Figure 16: Diagnose Manager

For example, the compared chosen is “UartRx/period_count_q_reg_15_inst”. From the diagnose manager the diagnose point is shown as UartRx/U60 and its corresponding error candidate are shown below. The error which higher percentage must be resolved first, from Figure 17 it is found out to be U18: NAND and U12: NOT gate. Right click on the error candidate to get the source code. Now this can be compared with the source code of the golden file and the point which is found to be non-equivalent can be corrected. The identical technique must be followed, but because the buggy file has been modified, it is necessary to utilize the amended buggy file. Overall, it will be possible to determine whether the non-equivalent points are lowered.



```
Non-equivalent signal and its error candidates
================================================================================
     ID (R)    Type            Likelihood  Name
--------------------------------------------------------------------------------
     2698      EON1                        /UartRx/U60
------- Candidates ------------------------------------------------------------
1:   2617      ND2             1.00        /UartRx/add_107/U18
2:   2647      IVDA            1.00        /UartRx/U12
3:   2616      AN3             0.98        /UartRx/add_107/U20
4:   2615      AN3             0.81        /UartRx/add_107/U24
5:   2663      IVDA            0.68        /UartRx/U10
6:   2694      EON1            0.32        /UartRx/U60
================================================================================
```

Figure 17: List of error candidates



Source 7 (golden): /nfs/home/a/ai_balan/COEN6551/conformal/syn_2.vhdl

History: /UartRx/add_107 U17 152 3 /nfs/home/a

```
U1 : IVI port map( A => A(14), Z => n1);
U2 : IVI port map( A => A(0), Z => SUM_0_port);
U3 : EO port map( A => A(9), B => n3, Z => SUM_9_port);
U4 : EN port map( A => A(8), B => n4, Z => SUM_8_port);
U5 : ND2 port map( A => A(7), B => n5, Z => n4);
U6 : EO port map( A => A(7), B => n5, Z => SUM_7_port);
U7 : EN port map( A => A(6), B => n6, Z => SUM_6_port);
U8 : ND2 port map( A => A(5), B => n7, Z => n6);
U9 : EO port map( A => A(5), B => n7, Z => SUM_5_port);
U10 : EN port map( A => A(4), B => n8, Z => SUM_4_port);
U11 : ND2 port map( A => A(3), B => n9, Z => n8);
U12 : EO port map( A => A(3), B => n9, Z => SUM_3_port);
U13 : EN port map( A => A(2), B => n10, Z => SUM_2_port);
U14 : ND2 port map( A => A(1), B => A(0), Z => n10);
U15 : EN port map( A => A(1), B => SUM_0_port, Z => SUM_1_port);
U16 : EO port map( A => A(15), B => n11, Z => SUM_15_port);
U17 : NR2 port map( A => n12, B => n1, Z => n11);
U18 : EN port map( A => A(14), B => n12, Z => SUM_14_port);
U19 : ND2 port map( A => A(13), B => n13, Z => n12);
U20 : EO port map( A => A(13), B => n13, Z => SUM_13_port);
U21 : AN3 port map( A => A(11), B => n14, C => A(12), Z => n13);
U22 : EN port map( A => A(12), B => n15, Z => SUM_12_port);
U23 : ND2 port map( A => A(11), B => n14, Z => n15);
U24 : EO port map( A => A(11), B => n14, Z => SUM_11_port);
U25 : AN3 port map( A => n3, B => A(9), C => A(10), Z => n14);
U26 : EN port map( A => A(10), B => n16, Z => SUM_10_port);
U27 : ND2 port map( A => n3, B => A(9), Z => n16);
U28 : AN3 port map( A => A(7), B => n5, C => A(8), Z => n3);
U29 : AN3 port map( A => A(5), B => n7, C => A(6), Z => n5);
U30 : AN3 port map( A => A(3), B => n9, C => A(4), Z => n7);
U31 : AN3 port map( A => A(1), B => A(0), C => A(2), Z => n9);

end SYN_rpl;

library IEEE;
```

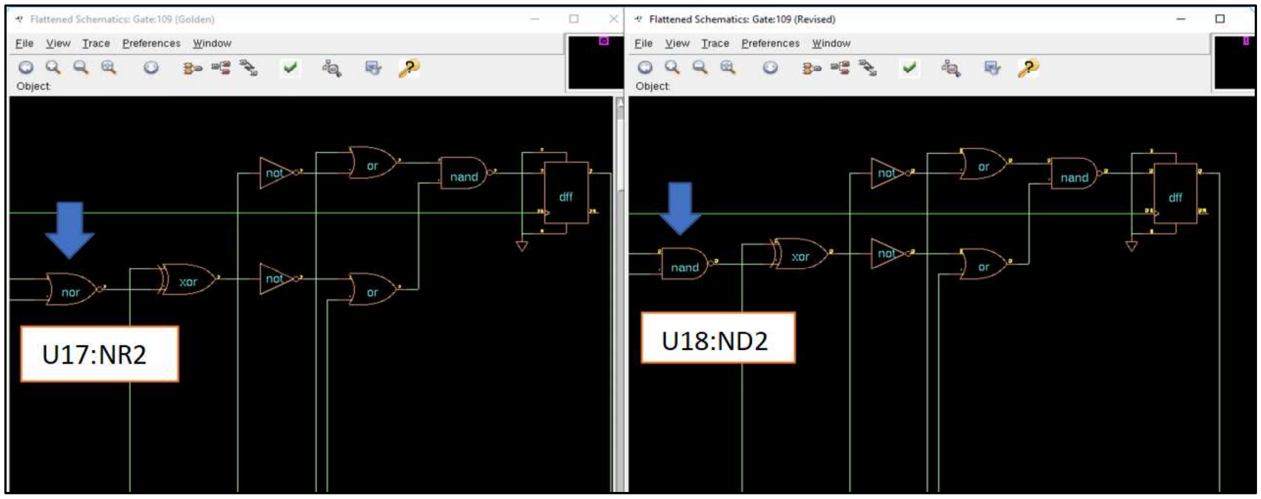
Source 8 (revised): /nfs/home/a/ai_balan/COEN6551/conformal/uart_syn_buggy.vhdl

History: /UartRx/add_107 U18 157 3 /nfs/home/a

```
U6 : EO port map( A => A(7), B => n4, Z => SUM(7));
U7 : EN port map( A => A(6), B => n5, Z => SUM(6));
U8 : ND2 port map( A => A(5), B => n6, Z => n5);
U9 : EO port map( A => A(5), B => n6, Z => SUM(5));
U10 : EN port map( A => A(4), B => n7, Z => SUM(4));
U11 : ND2 port map( A => A(3), B => n8, Z => n7);
U12 : EO port map( A => A(3), B => n8, Z => SUM(3));
U13 : EN port map( A => A(2), B => n9, Z => SUM(2));
U14 : ND2 port map( A => A(1), B => A(0), Z => n9);
U15 : EO port map( A => A(15), B => n11, Z => SUM(15));



U17 : EN port map( A => A(14), B => n11, Z => SUM(14));


U18 : ND2 port map( A => A(13), B => n12, Z => n11);
U19 : EO port map( A => A(13), B => n12, Z => SUM(13));
U20 : AN3 port map( A => A(11), B => n13, C => A(12), Z => n12);
U21 : AN3 port map( A => A(12), B => n14, C => A(9), Z => SUM(12));


U22 : ND2 port map( A => A(11), B => n13, Z => n14);
U23 : EO port map( A => A(11), B => n13, Z => SUM(11));
U24 : AN3 port map( A => n2, B => A(9), C => A(10), Z => n13);
U25 : EN port map( A => A(10), B => n15, Z => SUM(10));
U26 : ND2 port map( A => n2, B => A(9), Z => n15);
U29 : AN3 port map( A => A(3), B => n8, C => A(4), Z => n6);
U30 : AN3 port map( A => A(1), B => A(0), C => A(2), Z => n8);

end SYN_rpl;

library IEEE;

use IEEE.std_logic_1164.all;
```

Figure 18: Source code comparison

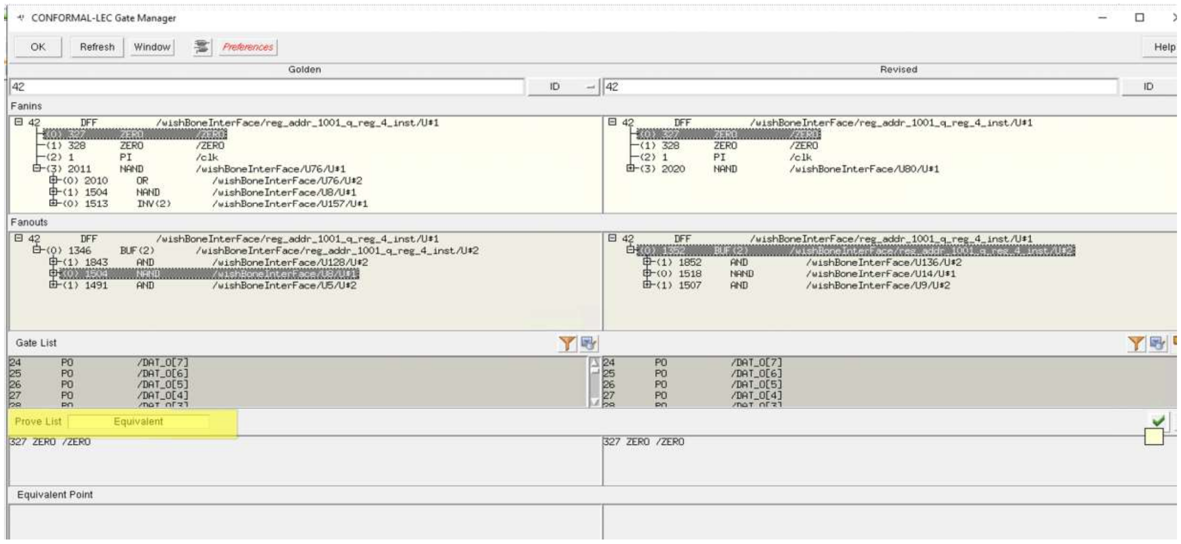
Another technique to diagnose errors is to use a schematic viewer to analyze these given candidates, and if errors have been identified, Source Code Manager may be used to edit them, as illustrated in Figure 19. To see the schematic view right click on the error candidate and select schematics. Below it can be seen that NAND gate in the revised needs to be changed to NOR, which is indicated in the golden.



Flattened Schematics: Gate:109 (Golden)
Flattened Schematics: Gate:109 (Revised)
File View Trace Preferences Window
Object:
not
or
nand
dff
nor
xor
U17:NR2
U18:ND2

Figure 19: Schematics view

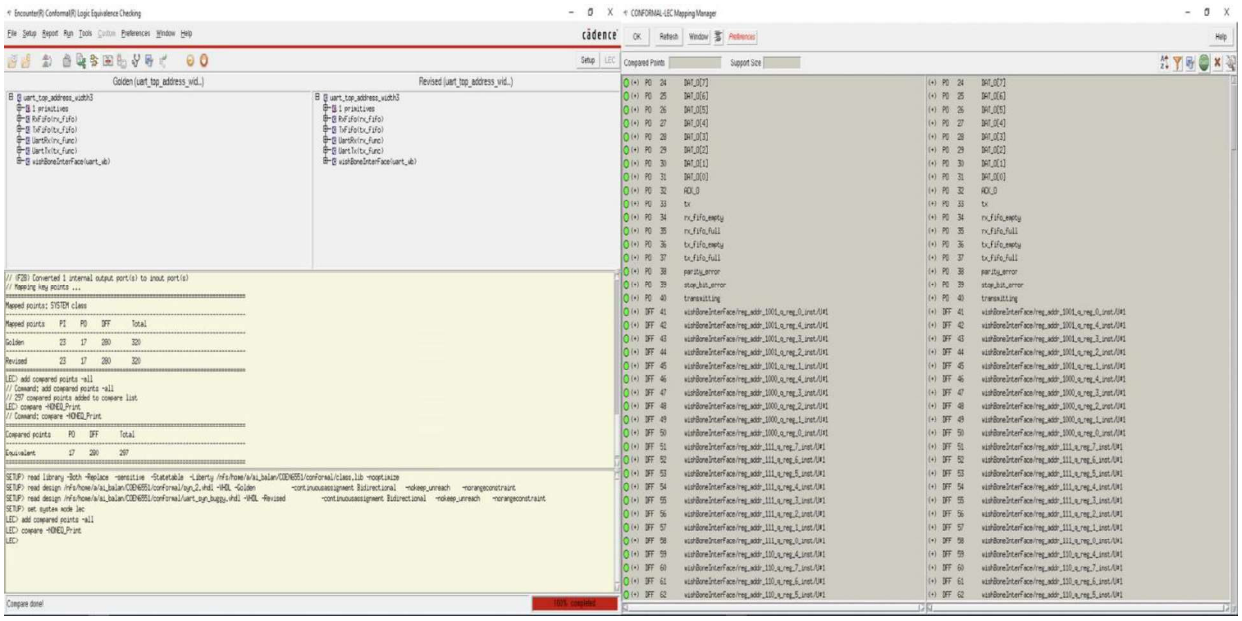
Another way to fix errors with Conformal is to double-check each gate. However, after selecting the relevant error candidate from the list, Report gate option is chosen. All the errors may be debugged by examining which gate is coupled to which gate, input, and output, and validating their equivalence. Figure 20 illustrates how to verify for equivalence. At this point, by adding the selected line to the proof list, one may deduce that if the word equivalent occurs on the blank rectangle, they are both comparable; otherwise, additional gates must be checked to see if they are equal.



```
Golden: 42
Fanins
⊟ 42       DFF       /wishBoneInterFace/reg_addr_1001_q_reg_4_inst/U#1
  ├(0) 327     ZERO      /ZERO
  ├(1) 328     ZERO      /ZERO
  ├(2) 1       PI        /clk
  ⊟(3) 2011    NAND      /wishBoneInterFace/U76/U#1
     ⊞(0) 2010    OR        /wishBoneInterFace/U76/U#2
     ⊞(1) 1504    NAND      /wishBoneInterFace/U8/U#1
     ⊞(0) 1513    INV(2)    /wishBoneInterFace/U157/U#1
Fanouts
⊟ 42       DFF       /wishBoneInterFace/reg_addr_1001_q_reg_4_inst/U#1
  ⊟(0) 1346    BUF(2)    /wishBoneInterFace/reg_addr_1001_q_reg_4_inst/U#2
     ⊞(1) 1843    AND       /wishBoneInterFace/U128/U#2
     ⊞(0) 1504    NAND      /wishBoneInterFace/U8/U#1
     ⊞(1) 1491    AND       /wishBoneInterFace/U5/U#2
Gate List
24     PO     /DAT_O[7]
25     PO     /DAT_O[6]
26     PO     /DAT_O[5]
27     PO     /DAT_O[4]
28     PO     /DAT_O[3]
327 ZERO /ZERO

Revised: 42
Fanins
⊟ 42       DFF       /wishBoneInterFace/reg_addr_1001_q_reg_4_inst/U#1
  ├(0) 327     ZERO      /ZERO
  ├(1) 328     ZERO      /ZERO
  ├(2) 1       PI        /clk
  ⊞(3) 2020    NAND      /wishBoneInterFace/U80/U#1
Fanouts
⊟ 42       DFF       /wishBoneInterFace/reg_addr_1001_q_reg_4_inst/U#1
  ⊟(0) 1352    BUF(2)    /wishBoneInterFace/reg_addr_1001_q_reg_4_inst/U#2
     ⊞(1) 1852    AND       /wishBoneInterFace/U136/U#2
     ⊞(0) 1518    NAND      /wishBoneInterFace/U14/U#1
     ⊞(1) 1507    AND       /wishBoneInterFace/U9/U#2
Gate List
24     PO     /DAT_O[7]
25     PO     /DAT_O[6]
26     PO     /DAT_O[5]
27     PO     /DAT_O[4]
28     PO     /DAT_O[3]
327 ZERO /ZERO
```

Figure 20: Report gate error tracking

Furthermore, if they are not comparable, the procedure must be repeated by comparing each Golden and updated section one by one. Overall, this strategy aids us in locating the malfunctioning gate. The malfunctioning gate's corresponding golden gate are be used to troubleshoot it. After following the above-mentioned methods of error correction, all the bugs are resolved, and the verification is successful.


Encounter(R) Conformal(R) Logic Equivalence Checking
File
Setup
Report
Run
Tools
Custom
Preferences
Window
Help
cadence
Setup
LEC
Golden (uart_top_address_wid..)
Revised (uart_top_address_wid..)
CONFORMAL-LEC Mapping Manager
OK
Refresh
Window
Preferences
Help
Compared Points
Support Size

```
// (F28) Converted 1 internal output port(s) to inout port(s)
// Mapping key points ...
================================================================
Mapped points: SYSTEM class
----------------------------------------------------------------
Mapped points    PI    PO    DFF      Total
----------------------------------------------------------------
Golden           23    17    280      320
----------------------------------------------------------------
Revised          23    17    280      320
================================================================
LEC> add compared points -all
// Command: add compared points -all
// 297 compared points added to compare list
LEC> compare -NONEQ_Print
// Command: compare -NONEQ_Print
================================================================
Compared points      PO    DFF      Total
----------------------------------------------------------------
Equivalent           17    280      297
================================================================

SETUP> read library -Both -Replace  -sensitive  -Statetable  -Liberty /nfs/home/a/ai_balan/COEN6551/conformal/class.lib -nooptimize
SETUP> read design /nfs/home/a/ai_balan/COEN6551/conformal/syn_2.vhdl -VHDL -Golden          -continuousassignment Bidirectional   -nokeep_unreach    -norangeconstraint
SETUP> read design /nfs/home/a/ai_balan/COEN6551/conformal/uart_syn_buggy.vhdl -VHDL -Revised          -continuousassignment Bidirectional   -nokeep_unreach    -norangeconstraint
SETUP> set system mode lec
LEC> add compared points -all
LEC> compare -NONEQ_Print
LEC>

Compare done!                                   100% completed
```

```
(+) PO   24   DAT_O[7]
(+) PO   25   DAT_O[6]
(+) PO   26   DAT_O[5]
(+) PO   27   DAT_O[4]
(+) PO   28   DAT_O[3]
(+) PO   29   DAT_O[2]
(+) PO   30   DAT_O[1]
(+) PO   31   DAT_O[0]
(+) PO   32   ACK_O
(+) PO   33   tx
(+) PO   34   rx_fifo_empty
(+) PO   35   rx_fifo_full
(+) PO   36   tx_fifo_empty
(+) PO   37   tx_fifo_full
(+) PO   38   parity_error
(+) PO   39   stop_bit_error
(+) PO   40   transmitting
(+) DFF  41   wishBoneInterface/reg_addr_1001_q_reg_0_inst/U#1
(+) DFF  42   wishBoneInterface/reg_addr_1001_q_reg_4_inst/U#1
(+) DFF  43   wishBoneInterface/reg_addr_1001_q_reg_3_inst/U#1
(+) DFF  44   wishBoneInterface/reg_addr_1001_q_reg_2_inst/U#1
(+) DFF  45   wishBoneInterface/reg_addr_1001_q_reg_1_inst/U#1
(+) DFF  46   wishBoneInterface/reg_addr_1000_q_reg_4_inst/U#1
(+) DFF  47   wishBoneInterface/reg_addr_1000_q_reg_3_inst/U#1
(+) DFF  48   wishBoneInterface/reg_addr_1000_q_reg_2_inst/U#1
(+) DFF  49   wishBoneInterface/reg_addr_1000_q_reg_1_inst/U#1
(+) DFF  50   wishBoneInterface/reg_addr_1000_q_reg_0_inst/U#1
(+) DFF  51   wishBoneInterface/reg_addr_111_q_reg_7_inst/U#1
(+) DFF  52   wishBoneInterface/reg_addr_111_q_reg_6_inst/U#1
(+) DFF  53   wishBoneInterface/reg_addr_111_q_reg_5_inst/U#1
(+) DFF  54   wishBoneInterface/reg_addr_111_q_reg_4_inst/U#1
(+) DFF  55   wishBoneInterface/reg_addr_111_q_reg_3_inst/U#1
(+) DFF  56   wishBoneInterface/reg_addr_111_q_reg_2_inst/U#1
(+) DFF  57   wishBoneInterface/reg_addr_111_q_reg_1_inst/U#1
(+) DFF  58   wishBoneInterface/reg_addr_111_q_reg_0_inst/U#1
(+) DFF  59   wishBoneInterface/reg_addr_110_q_reg_4_inst/U#1
(+) DFF  60   wishBoneInterface/reg_addr_110_q_reg_7_inst/U#1
(+) DFF  61   wishBoneInterface/reg_addr_110_q_reg_6_inst/U#1
(+) DFF  62   wishBoneInterface/reg_addr_110_q_reg_5_inst/U#1
```

Figure 21:Corrected buggy file with zero unmapped points

5. DISCUSSION AND ANALYSIS OF VERIFICATION RESULT

The verification method using formality is carried out using schematic view, which aided us in locating problematic gates on comparing the schematic view of both reference and implementation. The errors are rectified by replacing appropriate gates, corresponding port mapping, created necessary signals to match the required connection. Also, we checked the behavior of the block to do the corrections. Moreover, by back-tracing the inputs and outputs in comparison with reference schematic view, we were able to find the faulty blocks. The verification process using conformal is done by selecting the non-equivalent point and diagnosing the errors with various options like error tacking, schematic view, source code comparison. In both formality and conformal, schematic view comparison was convenient in bug hunting.

6. CHALLENGES

- **Synopsys Design compiler**

  After synthesizing, when we loaded the synthesized file in Synopsys formality, we got a few errors and warnings, like FMR 089, FMR VHDL-1002, VHDL- 290, FMR VHDL-011, FMR VHDL-048.

  VHDL-290 stated that a dummy netlist was created, which was not present in class.db file. FMR VHDL-048 was raised because the packages in the library file were corrupted. FMR-089 states that verification results may disagree with the logic simulator. So, we tried to change the FMR-089 error into a warning by executing -> set hdlin_error_on_mismatch_message "FMR_VHDL-1002". Finally, we figured out that there were some err with the Design Compiler, which resulted in errors while loading the file in Synopsys Formality.

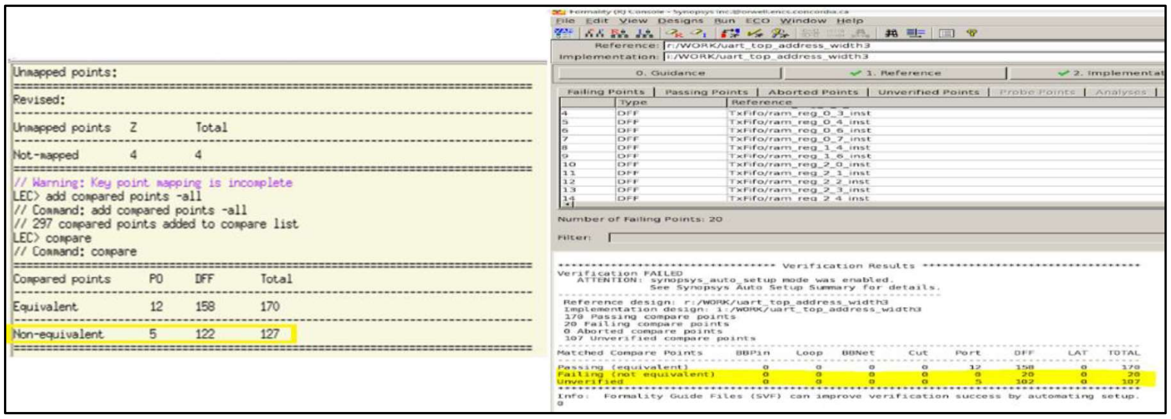
- Behavioral code understanding: Knowing how the designer interlinked/ called each component, nomenclature, and functionality took us more time.
- Back tracing versus error candidate: We found fixing fail compare points using back tracing easier than looking up the error candidates.
- **Conformal**

  There were two formats for reading the library. One is Liberty, and the other is SVA. SVA supports .V, **.lib** and .vpx file. When we tried using the SVA format, there were too many failing points with Blackbox errors compared to the liberty format. So we proceeded with the Liberty format.

The order of reading the library file created some challenges. Reading the library file between the golden and revised file caused more non-equivalent points. So, we read the library before reading the design files.

7. SYNOPSYS FORMALITY vs CADENCE CONFORMAL

For equivalence checking, both formality and conformal tool are helpful, and they compare the functionality of the design. In formality, the total failing (not equivalent) points are 20 and unverified points are 107. So, totally, there are 127 bugs. In Conformal, there are 127 non-equivalent points as highlighted in the figure below. Both formality and conformal tool shows same number of errors.



```
Unmapped points:
================================================================================
Revised:
--------------------------------------------------------------------------------
Unmapped points   Z          Total
--------------------------------------------------------------------------------
Not-mapped        4          4
================================================================================
// Warning: Key point mapping is incomplete
LEC> add compared points -all
// Command: add compared points -all
// 297 compared points added to compare list
LEC> compare
// Command: compare
================================================================================
Compared points      PO      DFF        Total
--------------------------------------------------------------------------------
Equivalent           12      158        170
--------------------------------------------------------------------------------
Non-equivalent       5       122        127
================================================================================
```

Formality (R) Console - Synopsys Inc.@orwell.encs.concordia.ca
File Edit View Designs Run ECO Window Help
0. Guidance
1. Reference
2. Implementat
Failing Points | Passing Points | Aborted Points | Unverified Points | Probe Points | Analyses

```
Reference:       r:/WORK/uart_top_address_width3
Implementation:  i:/WORK/uart_top_address_width3

       Type     Reference
4      DFF      TxFifo/ram_reg_0_3_inst
5      DFF      TxFifo/ram_reg_0_4_inst
6      DFF      TxFifo/ram_reg_0_6_inst
7      DFF      TxFifo/ram_reg_0_7_inst
8      DFF      TxFifo/ram_reg_1_4_inst
9      DFF      TxFifo/ram_reg_1_6_inst
10     DFF      TxFifo/ram_reg_2_0_inst
11     DFF      TxFifo/ram_reg_2_1_inst
12     DFF      TxFifo/ram_reg_2_2_inst
13     DFF      TxFifo/ram_reg_2_3_inst
14     DFF      TxFifo/ram_reg_2_4_inst

Number of Failing Points: 20

Filter:

********************************* Verification Results *********************************
Verification FAILED
   ATTENTION: synopsys_auto_setup mode was enabled.
              See Synopsys Auto Setup Summary for details.
------------------------------------------------------------
 Reference design: r:/WORK/uart_top_address_width3
 Implementation design: i:/WORK/uart_top_address_width3
 170 Passing compare points
 20 Failing compare points
 0 Aborted compare points
 107 Unverified compare points
------------------------------------------------------------------------------------------
Matched Compare Points     BBPin    Loop    BBNet    Cut    Port    DFF    LAT    TOTAL
------------------------------------------------------------------------------------------
Passing (equivalent)           0       0        0      0      12    158      0      170
Failing (not equivalent)       0       0        0      0       0     20      0       20
Unverified                     0       0        0      0       5    102      0      107
******************************************************************************************
Info:  Formality Guide Files (SVF) can improve verification success by automating setup.
0
```

Figure 22: Formality Vs Conformal error report

The risk of missing critical bugs are reduced in cadence tool with independent verification technology. They are helpful in corner case verification.

8. CONCLUSION

The design provided to us is a two-way RS232 communication module capable of transmitting and receiving 5,6,7 or 8-bit words. In addition, it also consists of a Parity bit, two stop bits to establish an appropriate communication protocol. The design file consists of uart_tx.vhd, uart_tx-fifo.vhd, uart_rx.vhd, uart_rx_fifo.vhd, uart_wb.vhd and uart_top.vhd. We synthesize the RTL design for Logical Equivalence Checking (LEC). Synopsys' Design Compiler is used for the synthesis process. The synthesized file will be in the.vhd/.vhdl format, and it will be created once the

RTL design has been constrained into design constraints like as power, area, and time. Buggy file is provided with bugs which need to be detected and corrected using equivalence checking tool. RTL- gate level equivalence checking is done in both tools to check the accuracy of synthesis.

In formality, synthesized file (syn_2. vhdl)/Reference is compared with the given buggy file (uart_syn_buggy.vhdl)/Implementation. There were 20 not-equivalent points and 107 unverified points detected. Using schematic view, the errors such as gate mismatch, wrong port mapping and missing signals are detected, and corrections are made. Finally, the verification is successful with 0 not-equivalence points. In Conformal, by comparing synthesized file (syn_2.vhdl)/Golden with buggy design(uart_syn_buggy.vhdl)/Revised, there were 127 non-equivalent points. They are diagnosed by source code comparison, schematic view comparison and the error candidates were traced and corrected. All the bugs are fixed, and verification is successful with 0 non-equivalent points. Both tools have different approaches for tracing and correcting the errors in the design.

9. REFERENCE:

[1] https://users.encs.concordia.ca/~tahar/coen6551/notes/ec-manual16.pdf
[2] https://users.encs.concordia.ca/~tahar/coen6551/notes/dc-slides16.pdf
[3] https://users.encs.concordia.ca/~tahar/coen6551/notes/formality-slides16.pdf
[4] https://users.encs.concordia.ca/~tahar/coen6551/notes/lec-slides16.pdf
[5] https://opencores.org/projects/rs232_with_buffer_and_wb
[6] "Krishnegowda, Deekshith. (2021). A primer on Logical Equivalence Checking (LEC) using Conformal." https://www.edn.com/a-primer-on-logical-equivalence-checking-lec-using-conformal/
[7] https://s2.smu.edu/~manikas/CAD_Tools/SDC/lab2/lab2_synopsys_dc.pdf
[8] https://pdfcoffee.com/formality-debugging-failing-verifications-presentation-pdf-free.html
[9] https://www.synopsys.com/content/dam/synopsys/implementation&signoff/datasheets/formality-and-formality-ultra-ds.pdf