A Project Report on

VERIFICATION OF AES DECODER USING SYNOPSYS FORMALITY AND CADENCE CONFORMAL

Submitted By,
GROUP B

Arihant Jain	40067961
Elnaz Ghafaradli	40021804
Melany Valder	40054184
Zahidul Amin	40031489

FORMAL HARDWARE VERIFICATION – COEN 6551

Course Given By,
Prof. Sofiene Tahar

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Table of Contents

List of Figures 2
List of Tables 2
ABSTRACT 3
INTRODUCTION 4
1. DESIGN DESCRIPTION 4
  1.1 Decoding Process 5
  1.2 Working Principle 5
2. RTL SYNTHESIS USING SYNOPSYS DESIGN COMPILER 7
3. EQUIVALENCE CHECKING 9
  3.1 RTL to Gate Level Equivalence Checking 9
  3.2 Gate Level to Gate Level Equivalence Checking 9
4. VERIFICATION PROCESS IN FORMALITY/CONFORMAL 14
5. ANALYSIS OF VERIFICATION RESULTS 16
  5.1 Error Reduction in Conformal/ Formality 17
CONCLUSION 18
CHALLENGES 19
REFERENCES 20

List of Figures

Figure 1 AES Encryption / Decryption flow………………………………… 3
Figure 2 Design module description………………………………………. 4
Figure 3 Setting clock constraints………………………………………… 6
Figure 4 Compilation of the design……………………………………….. 6
Figure 5 Symbolic and schematic view of the decoder…………………… 7
Figure 6 Number of passing and failing points between RTL and gate level files in Formality………………………………………………… 8
Figure 7 Number of passing and failing points between RTL and gate level files in Conformal……………………………………………….. 8
Figure 8 Number of passing and failing points between reference and buggy file in Formality………………………………………………… 9
Figure 9 List of failing points……………………………………………... 9
Figure 10 Number of mapped points, equivalent and nonequivalent points in Conformal…………………………………………………….. 10
Figure 11 List of equivalent and non-equivalent points ……………………… 10
Figure 12 Traced error candidate in Conformal……………………………. 11
Figure 13 Traced error candidate in Formality……………………………... 11
Figure 14 Successful Verification message in Formality……………………... 12
Figure 15 Design verification process flow overview……………………….. 13

List of Tables

Table 1 AES Decoder Block Diagram and Signal Information…………….. 5
Table 2 Error candidates and their correction………………………………. 16
Table 3 Error reduction comparison between the two tools………………... 16
Table 4 Comparison between Formality and Conformal…………………… 17

ABSTRACT

In recent years, formal methods have gained popularity in the field of hardware testing to guarantee the quality and accuracy of designs as traditional techniques such as simulation have some limitations. With growing technology, the designs of old and new devices, algorithms and circuits are becoming more and more complex. Each and every connection and component cannot be individually tested by simulation.

Formal verification techniques are usually used to evaluate whether two different circuits show the same behaviour. The different techniques include Equivalence checking, model checking and theorem proving. Each of these methods are preferred in different environments depending on the type of designs.

Equivalence checking using formal methods is done to prove whether the same behavior is exhibited by two different circuits. Equivalence checking tools include Formality by Synopsys and Conformal LEC by Cadence.

This project makes use of these two equivalence checking tools to check the equivalence between the given RTL code and gate level code of a AES decoder and debug the design file at gate level in order to obtain equivalence.

INTRODUCTION

In today's world data security is the only form of trust required to build good communication. A communication channel is said to be efficient when there is a guarantee that any communication between two systems is adamant enough to provide privacy and security. It is crucial for any form of communication, whether for business, banking or entertainment that no party can have any advantage over the other. With the extraordinary maturity of data exchange in network environments and increasing the attackers' capabilities, information security has become the most important process for data storage and communication [1]. Cryptography corresponds to usage of protocols in order to prevent third parties from reading private messages. AES is one method for encryption of data.

1. DESIGN DESCRIPTION

The Advanced Encryption Standard (AES) also known as Rijndael algorithm is used to encrypt/decrypt data, the design standard of this tool has been set forth by Federal Information Processing Standard FIPS. The process flow of a 128 bit block-size AES algorithm which accepts a cyphered text and decrypt using 128,192 or 256 bit AES key is shown below.

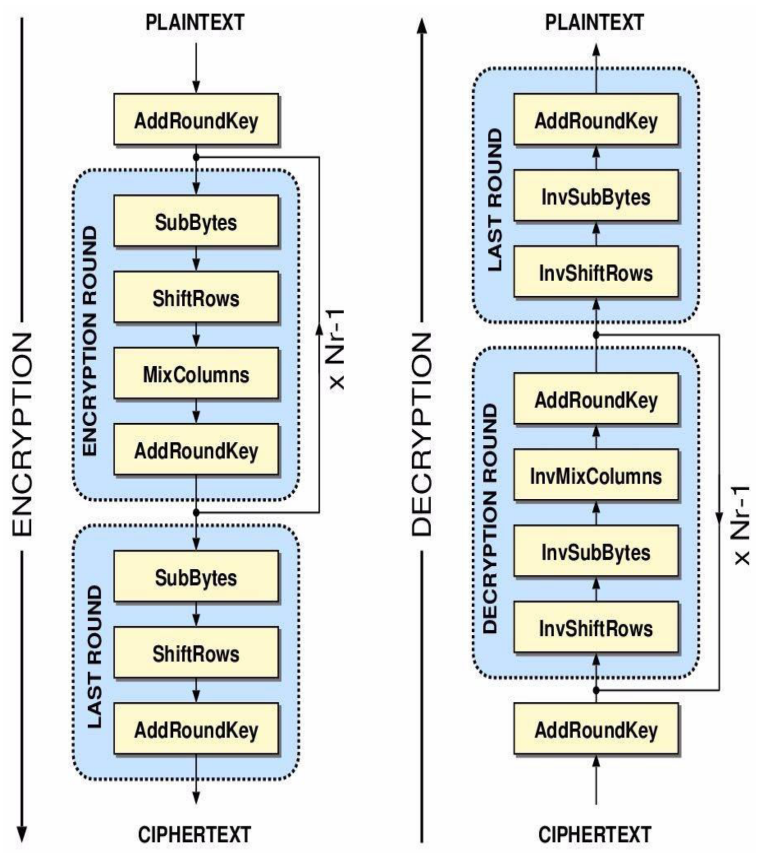


FIGURE 1 AES ENCRYPTION / DECRYPTION FLOW [2]

1.1 Decoding Process

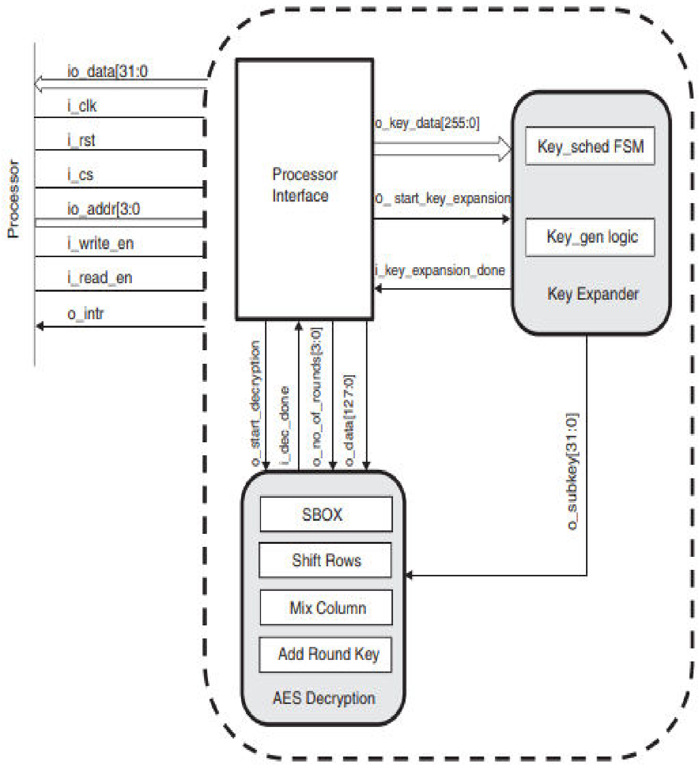
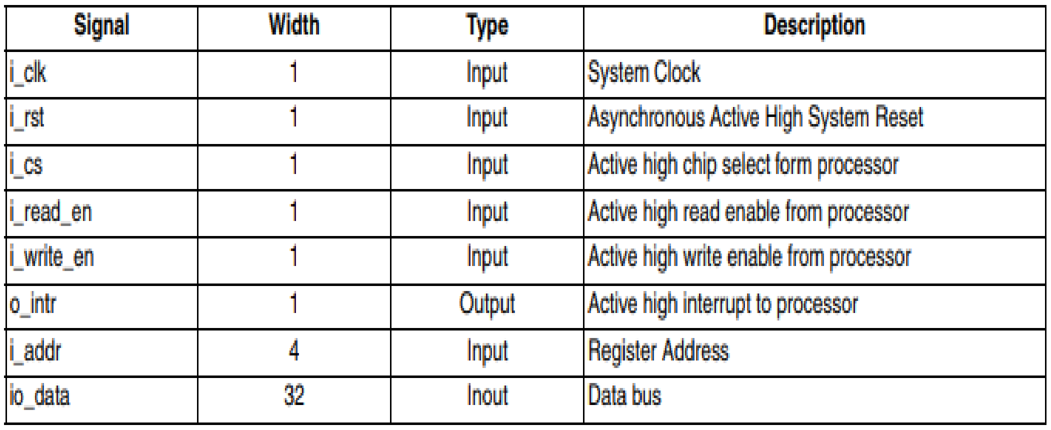


Figure 2. Design module description [3]

The main task of the decoder is to decipher encrypted data via Rijndael decoding algorithm to make understandable only to the person who is holding the Key. They Key allows ciphered text to be decoded, this is known as decipher text. The length of the Key can be 256/192/128 bit. Inside the decoder there is a Register map, which controls the data flow to the AES decoder. AES also has status registers which keeps track of the status of AES Decoder. Similarly, AES control register takes the responsibility of read and write operation. The KEY register is only used to write KEY data to the AES engine and Data registers are used to write or retrieve data from AES Engine.

1.2 Working Principle

The Decoder has three units, the processor interface which oversees all the registers for communication between the blocks. The Key Expansion module is responsible for manipulating the Key data and finally the decryption module holds necessary data including AES states and converts the ciphered text to readable text with the aid of the key.



Signal	Width	Type	Description
i_clk	1	Input	System Clock
i_rst	1	Input	Asynchronous Active High System Reset
i_cs	1	Input	Active high chip select form processor
i_read_en	1	Input	Active high read enable from processor
i_write_en	1	Input	Active high write enable from processor
o_intr	1	Output	Active high interrupt to processor
i_addr	4	Input	Register Address
io_data	32	Inout	Data bus

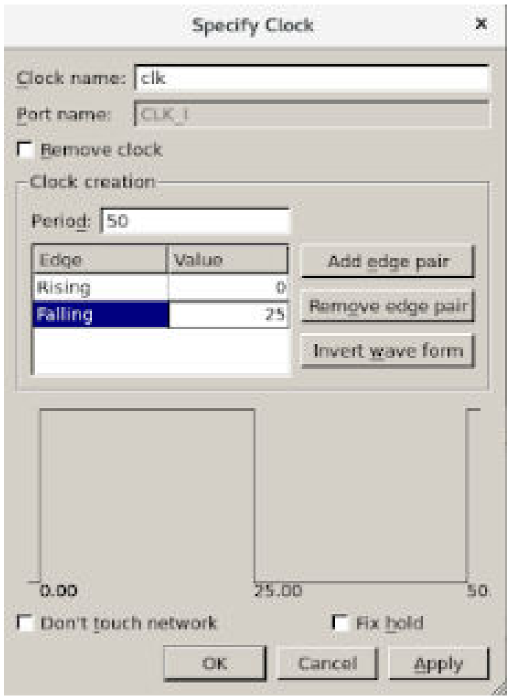
TABLE 1. AES DECODER BLOCK DIAGRAM AND SIGNAL INFORMATION [3]

When a Ciphered text containing passwords or credit card information is put into the decoder, it is only deciphered via Key. Decoder then follows a predefined step as shown in Figure 1 to decode the information known as decipher text. Decoder uses an iterated block decipher algorithm, it has a fixed block size of 128 with a changing key length. It utilizes 128 bits of data and gives out 128 bits of output. The Key length is used to decode the ciphered text. AES is designed to use as SPN (substitution-permutation Network). Decoding process uses decryption rounds of 10, 12 and 14 for key length of 128, 192 and 256.

The whole process starts where the processor writes the Key data into Key registers, then key setup command is initiated with the help of Key expansion module. Operation is requested to control registers; the completion of this step is monitored and noted inside status register. After that the processor stores the ciphered text to the data registers. After this decoding command is initiated to the control registers and completion of this task is updated in status registers.

2. RTL SYNTHESIS USING SYNOPSYS DESIGN COMPILER

In order to perform equivalence checking, we need to convert the RTL file to gate level netlist. For the given gate level file and reference file to be structurally same, the constraints such as area effort, power effort, timing, clock period, etc. has to be considered. The library file must be already loaded when the tool opens. This process of changing the Register Transfer Level file into the gate level netlist is called Logic synthesis. The tool that is used for the project to perform this function is Synopsys Design Compiler.



Specify Clock

Clock name: clk

Port name: CLK_I

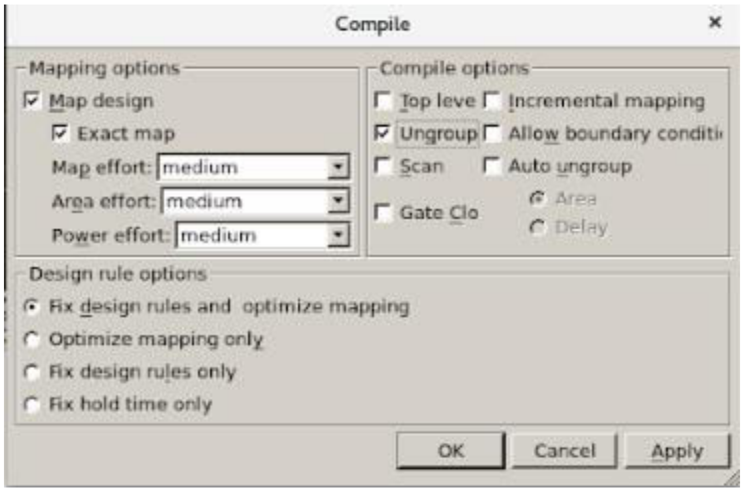
Remove clock

Clock creation

Period: 50

Edge	Value
Rising	0
Falling	25

Figure 3. Setting clock constraints



Compile

Mapping options

[✓ Map design]

[✓ Exact map]

Map effort: medium

Area effort: medium

Power effort: medium

Compile options

Top leve

Incremental mapping

[✓ Ungroup]

Allow boundary conditi

Scan

Auto ungroup

Gate Clo

[✓ Area]

Delay

Design rule options

[✓ Fix design rules and optimize mapping]

Optimize mapping only

Fix design rules only

Fix hold time only

Figure 4. Compilation of the design

The given design in VHDL is loaded into the tool and has to be analyzed. This step checks for non-synthesizability in the code. The design is then elaborated using the given Key-size. ”Check design” option is used to eliminate unconnected ports. Next, the clock period has to be specified to be set so that the design can operate within the clock rate. Other constraints are forced to the design through command box. After compilation, the design is converted from RTL to gate level netlist.

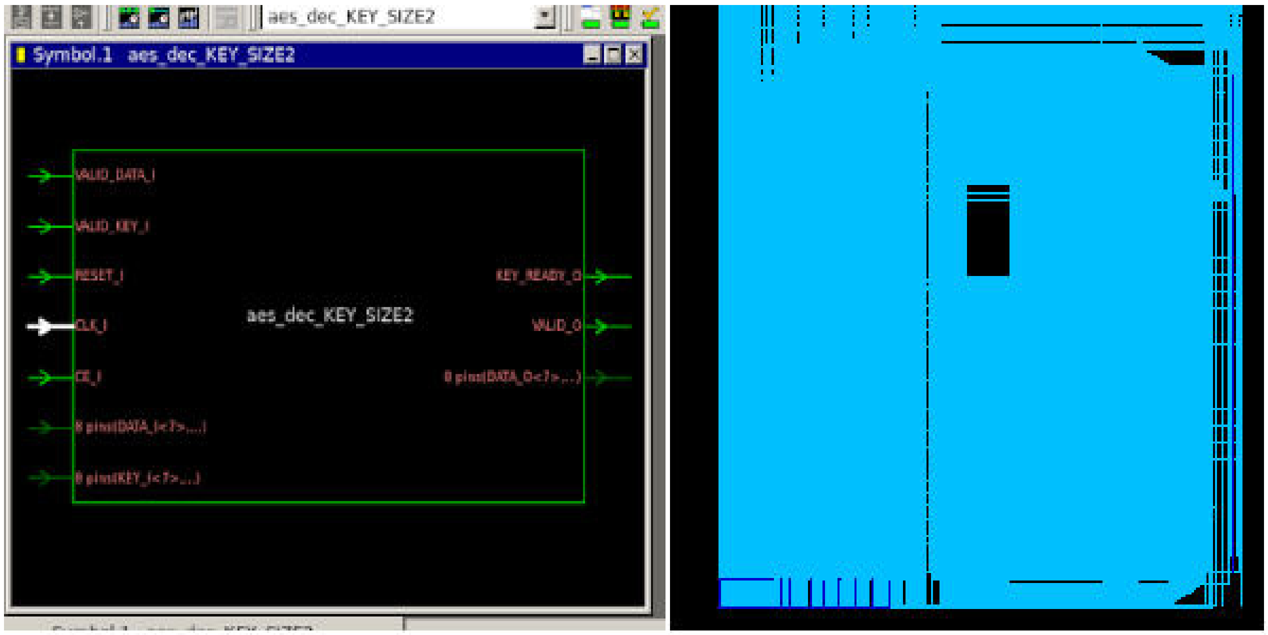


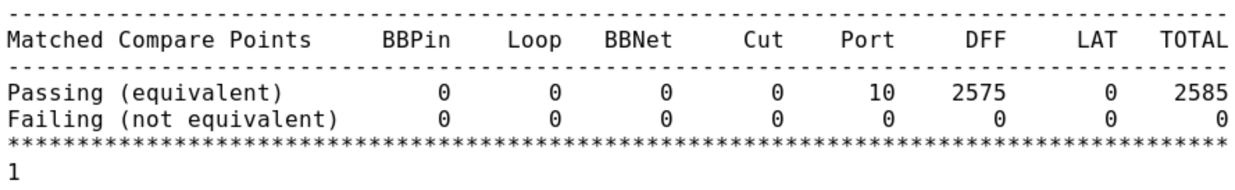
Figure 5. Symbolic and schematic view of the decoder

3. EQUIVALENCE CHECKING

There are two kind of equivalence checking to be performed. Firstly, Equivalence checking is performed between the given RTL file and the synthesized gate level netlist. Then, it is applied between the synthesized gate level netlist and the buggy gate level file.

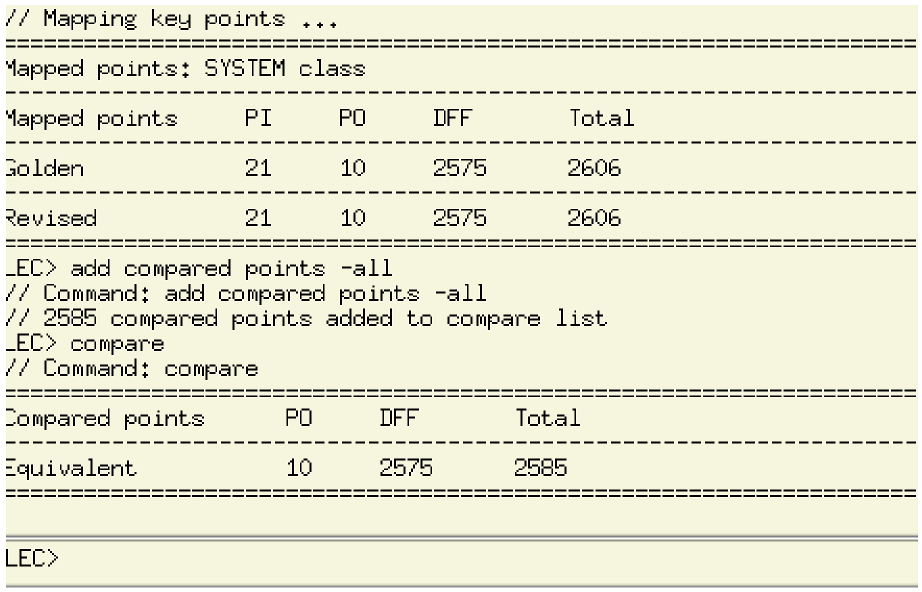
3.1 RTL to Gate Level Equivalence Checking

In both tools, Cadence Conformal and Synopsys Formality, equivalence between the two files are checked. The two files and the library are loaded in the two mentioned tools and "Match" option is operated in Formality and the "LEC" option is operated in Conformal. Clearly as it is shown in both tools, there is no non-equivalent point. At this point, RTL file is said to be equivalent to the synthesized gate level file.



```
-------------------------------------------------------------------------------------
Matched Compare Points     BBPin    Loop   BBNet     Cut    Port     DFF     LAT   TOTAL
-------------------------------------------------------------------------------------
Passing (equivalent)           0       0       0       0      10    2575       0    2585
Failing (not equivalent)       0       0       0       0       0       0       0       0
*************************************************************************************
1
```

Figure 6. Number of passing and failing points between RTL and gate level files in Formality



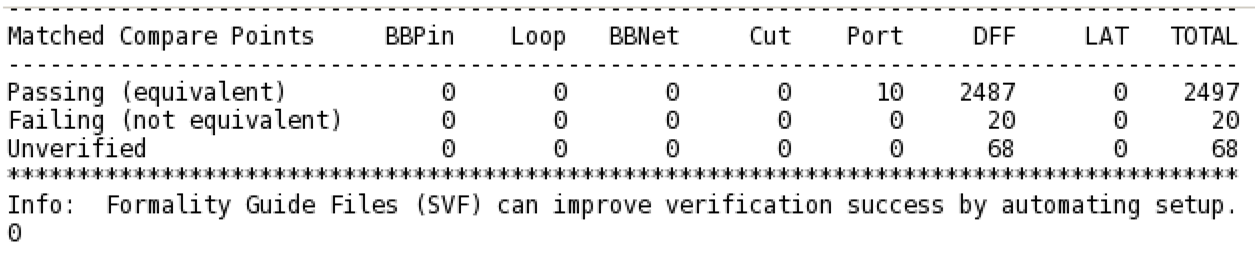
```
// Mapping key points ...
==================================================================
Mapped points: SYSTEM class
------------------------------------------------------------------
Mapped points     PI     PO     DFF       Total
------------------------------------------------------------------
Golden            21     10     2575      2606
------------------------------------------------------------------
Revised           21     10     2575      2606
==================================================================
LEC> add compared points -all
// Command: add compared points -all
// 2585 compared points added to compare list
LEC> compare
// Command: compare
==================================================================
Compared points      PO     DFF       Total
------------------------------------------------------------------
Equivalent           10     2575      2585
==================================================================

LEC>
```

Figure 7. Number of passing and failing points between RTL and gate level files in Conformal

3.2 Gate Level to Gate Level Equivalence Checking

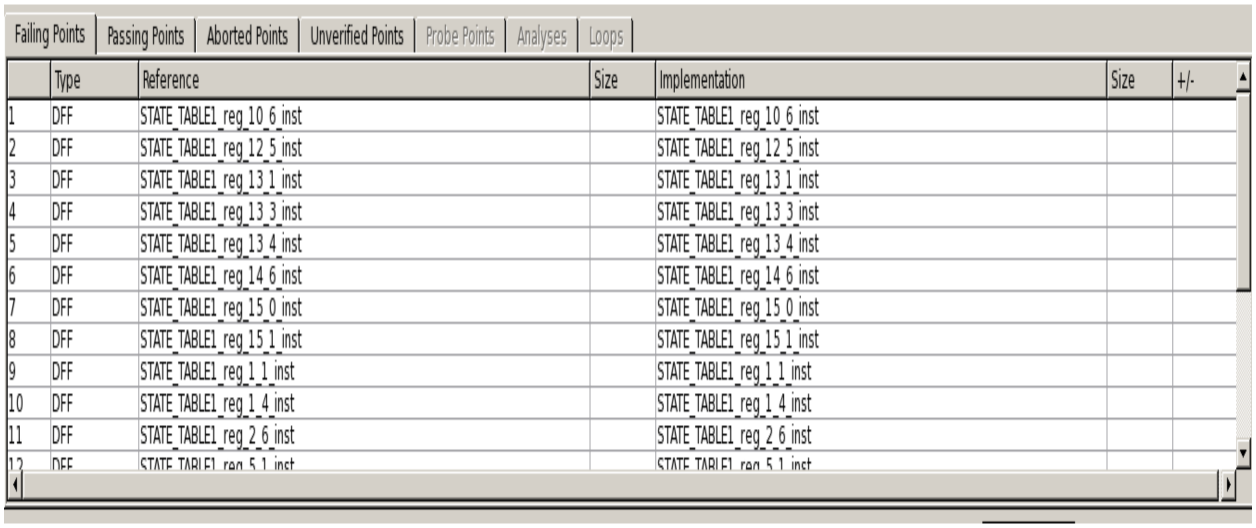
In this step, the synthesized design is considered a "Reference" and the buggy design is used as "Implementation". The previous steps are repeated as going through the equivalence checking process.

In Synopsys Formality tool, by selecting the bug, option called “Diagnose” can be used in order to trace back the error. At this point, the “Show Logic Cone” is checked which leads to the buggy gate. Once the buggy gate is detected, the “View Source” option is selected that shows the line that represents the bug in the code. Then, the buggy file will be corrected according to the Golden file and then the corrected buggy file will be loaded again. At this point, the number of bugs will be reduced. This process is repeated until the Verification is succeeded.



```
--------------------------------------------------------------------------------
Matched Compare Points     BBPin    Loop   BBNet     Cut    Port     DFF     LAT   TOTAL
--------------------------------------------------------------------------------
Passing (equivalent)           0       0       0       0      10    2487       0    2497
Failing (not equivalent)       0       0       0       0       0      20       0      20
Unverified                     0       0       0       0       0      68       0      68
********************************************************************************
Info:  Formality Guide Files (SVF) can improve verification success by automating setup.
0
```

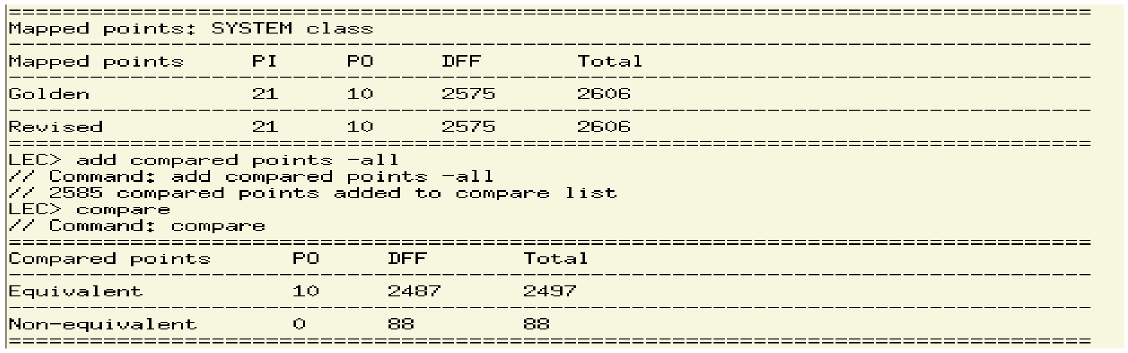
Figure 8. Number of passing and failing points between reference and buggy file in Formality



	Type	Reference	Size	Implementation	Size	+/-
1	DFF	STATE_TABLE1_reg_10_6_inst		STATE_TABLE1_reg_10_6_inst		
2	DFF	STATE_TABLE1_reg_12_5_inst		STATE_TABLE1_reg_12_5_inst		
3	DFF	STATE_TABLE1_reg_13_1_inst		STATE_TABLE1_reg_13_1_inst		
4	DFF	STATE_TABLE1_reg_13_3_inst		STATE_TABLE1_reg_13_3_inst		
5	DFF	STATE_TABLE1_reg_13_4_inst		STATE_TABLE1_reg_13_4_inst		
6	DFF	STATE_TABLE1_reg_14_6_inst		STATE_TABLE1_reg_14_6_inst		
7	DFF	STATE_TABLE1_reg_15_0_inst		STATE_TABLE1_reg_15_0_inst		
8	DFF	STATE_TABLE1_reg_15_1_inst		STATE_TABLE1_reg_15_1_inst		
9	DFF	STATE_TABLE1_reg_1_1_inst		STATE_TABLE1_reg_1_1_inst		
10	DFF	STATE_TABLE1_reg_1_4_inst		STATE_TABLE1_reg_1_4_inst		
11	DFF	STATE_TABLE1_reg_2_6_inst		STATE_TABLE1_reg_2_6_inst		
12	DFF	STATE_TABLE1_reg_5_1_inst		STATE_TABLE1_reg_5_1_inst		

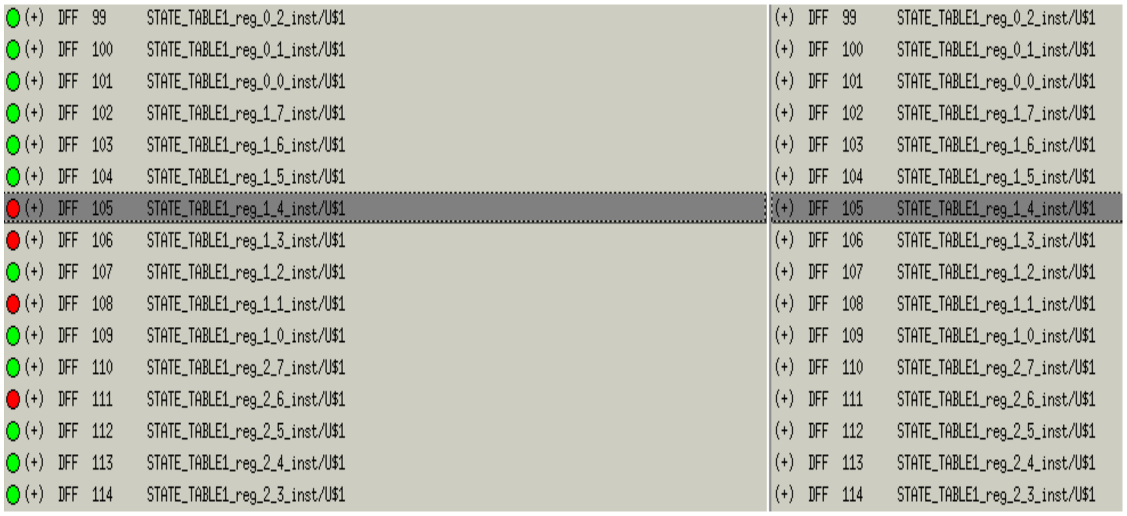
Figure 9. List of failing points

In Cadence conformal, the Matching part is done using “LEC” option. By selecting the “Compare” option in the “Run” menu, the equivalency is checked. Obtaining the report of the Compared points are by choosing “Compared points” in “Reports” menu. A number of failing points and passing points are provided. To detect a bug, a particular failing point is selected and diagnosed. By expanding the Schematic, the error can be traced. The correction is done in the correspondent line of the buggy code and the implementation design file is reloaded. This process is repeated until there are no failing points.



```
==================================================================================
Mapped points: SYSTEM class
----------------------------------------------------------------------------------
Mapped points      PI      PO      DFF        Total
----------------------------------------------------------------------------------
Golden             21      10      2575       2606
----------------------------------------------------------------------------------
Revised            21      10      2575       2606
==================================================================================
LEC> add compared points -all
// Command: add compared points -all
// 2585 compared points added to compare list
LEC> compare
// Command: compare
==================================================================================
Compared points       PO      DFF        Total
----------------------------------------------------------------------------------
Equivalent            10      2487       2497
----------------------------------------------------------------------------------
Non-equivalent        0       88         88
==================================================================================
```

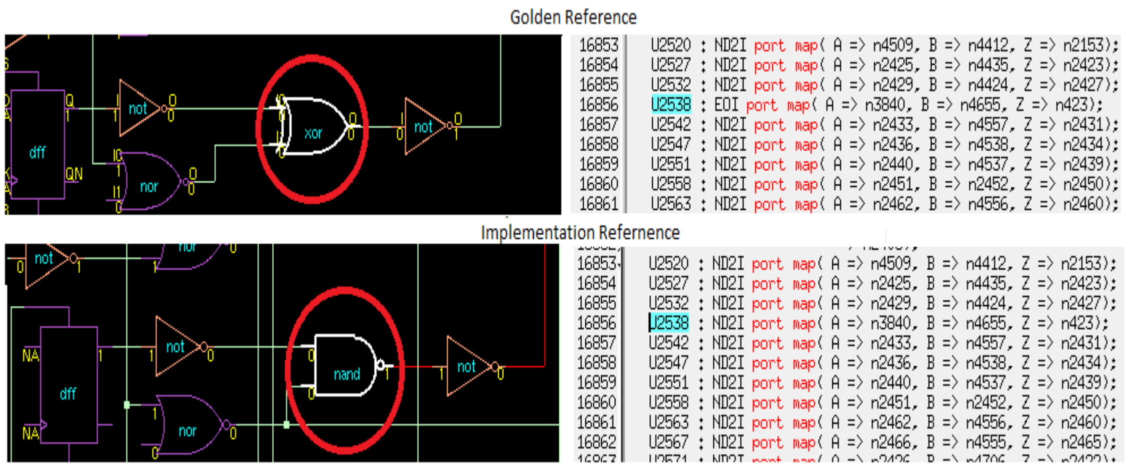
Figure 10. Number of mapped points, equivalent and nonequivalent points in Conformal



```
🟢 (+) DFF 99    STATE_TABLE1_reg_0_2_inst/U$1    (+) DFF 99    STATE_TABLE1_reg_0_2_inst/U$1
🟢 (+) DFF 100   STATE_TABLE1_reg_0_1_inst/U$1    (+) DFF 100   STATE_TABLE1_reg_0_1_inst/U$1
🟢 (+) DFF 101   STATE_TABLE1_reg_0_0_inst/U$1    (+) DFF 101   STATE_TABLE1_reg_0_0_inst/U$1
🟢 (+) DFF 102   STATE_TABLE1_reg_1_7_inst/U$1    (+) DFF 102   STATE_TABLE1_reg_1_7_inst/U$1
🟢 (+) DFF 103   STATE_TABLE1_reg_1_6_inst/U$1    (+) DFF 103   STATE_TABLE1_reg_1_6_inst/U$1
🟢 (+) DFF 104   STATE_TABLE1_reg_1_5_inst/U$1    (+) DFF 104   STATE_TABLE1_reg_1_5_inst/U$1
🔴 (+) DFF 105   STATE_TABLE1_reg_1_4_inst/U$1    (+) DFF 105   STATE_TABLE1_reg_1_4_inst/U$1
🔴 (+) DFF 106   STATE_TABLE1_reg_1_3_inst/U$1    (+) DFF 106   STATE_TABLE1_reg_1_3_inst/U$1
🟢 (+) DFF 107   STATE_TABLE1_reg_1_2_inst/U$1    (+) DFF 107   STATE_TABLE1_reg_1_2_inst/U$1
🔴 (+) DFF 108   STATE_TABLE1_reg_1_1_inst/U$1    (+) DFF 108   STATE_TABLE1_reg_1_1_inst/U$1
🟢 (+) DFF 109   STATE_TABLE1_reg_1_0_inst/U$1    (+) DFF 109   STATE_TABLE1_reg_1_0_inst/U$1
🟢 (+) DFF 110   STATE_TABLE1_reg_2_7_inst/U$1    (+) DFF 110   STATE_TABLE1_reg_2_7_inst/U$1
🔴 (+) DFF 111   STATE_TABLE1_reg_2_6_inst/U$1    (+) DFF 111   STATE_TABLE1_reg_2_6_inst/U$1
🟢 (+) DFF 112   STATE_TABLE1_reg_2_5_inst/U$1    (+) DFF 112   STATE_TABLE1_reg_2_5_inst/U$1
🟢 (+) DFF 113   STATE_TABLE1_reg_2_4_inst/U$1    (+) DFF 113   STATE_TABLE1_reg_2_4_inst/U$1
🟢 (+) DFF 114   STATE_TABLE1_reg_2_3_inst/U$1    (+) DFF 114   STATE_TABLE1_reg_2_3_inst/U$1
```

Figure 11. List of equivalent and non-equivalent points

The debugging process starts by clicking on the mismatched gates marked in red on the table of conformal. After that the schematics are selected. Then the primary task is to observe the difference between the two circuits. The circuit is analyzed and backward tracing is done until difference in gates are found. As one bug in design can introduce many errors, it is not possible to identify the exact bug, so the method used to trace back is noted down where the difference in output of bits occurs starting from a common point in both circuits.



```
U2520 : ND2I port map( A => n4509, B => n4412, Z => n2153);
U2527 : ND2I port map( A => n2425, B => n4435, Z => n2423);
U2532 : ND2I port map( A => n2429, B => n4424, Z => n2427);
U2538 : EOI port map( A => n3840, B => n4655, Z => n423);
U2542 : ND2I port map( A => n2433, B => n4557, Z => n2431);
U2547 : ND2I port map( A => n2436, B => n4538, Z => n2434);
U2551 : ND2I port map( A => n2440, B => n4537, Z => n2439);
U2558 : ND2I port map( A => n2451, B => n2452, Z => n2450);
U2563 : ND2I port map( A => n2462, B => n4556, Z => n2460);
```

```
U2520 : ND2I port map( A => n4509, B => n4412, Z => n2153);
U2527 : ND2I port map( A => n2425, B => n4435, Z => n2423);
U2532 : ND2I port map( A => n2429, B => n4424, Z => n2427);
U2538 : ND2I port map( A => n3840, B => n4655, Z => n423);
U2542 : ND2I port map( A => n2433, B => n4557, Z => n2431);
U2547 : ND2I port map( A => n2436, B => n4538, Z => n2434);
U2551 : ND2I port map( A => n2440, B => n4537, Z => n2439);
U2558 : ND2I port map( A => n2451, B => n2452, Z => n2450);
U2563 : ND2I port map( A => n2462, B => n4556, Z => n2460);
U2567 : ND2I port map( A => n2466, B => n4555, Z => n2465);
```

Figure 12. Traced error candidate in Conformal

Unlike Conformal, in Formality the list containing only the failing points is shown. By selecting a particular point and diagnosing it, the schematic of the combinational circuit pertaining that point can be viewed. Then the trace back method is used on the erroneous path to detect the bug.

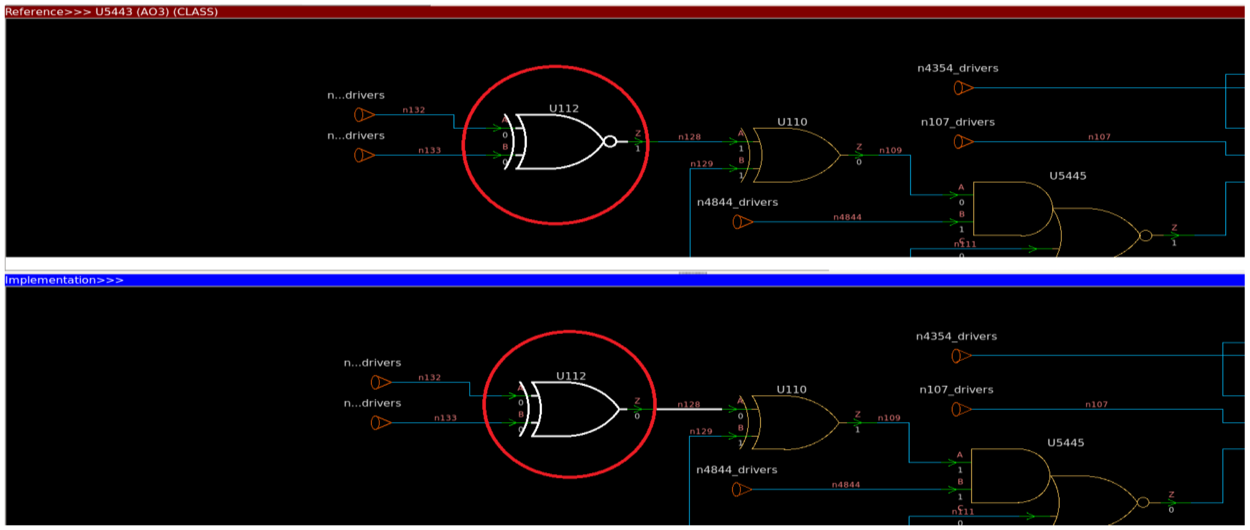
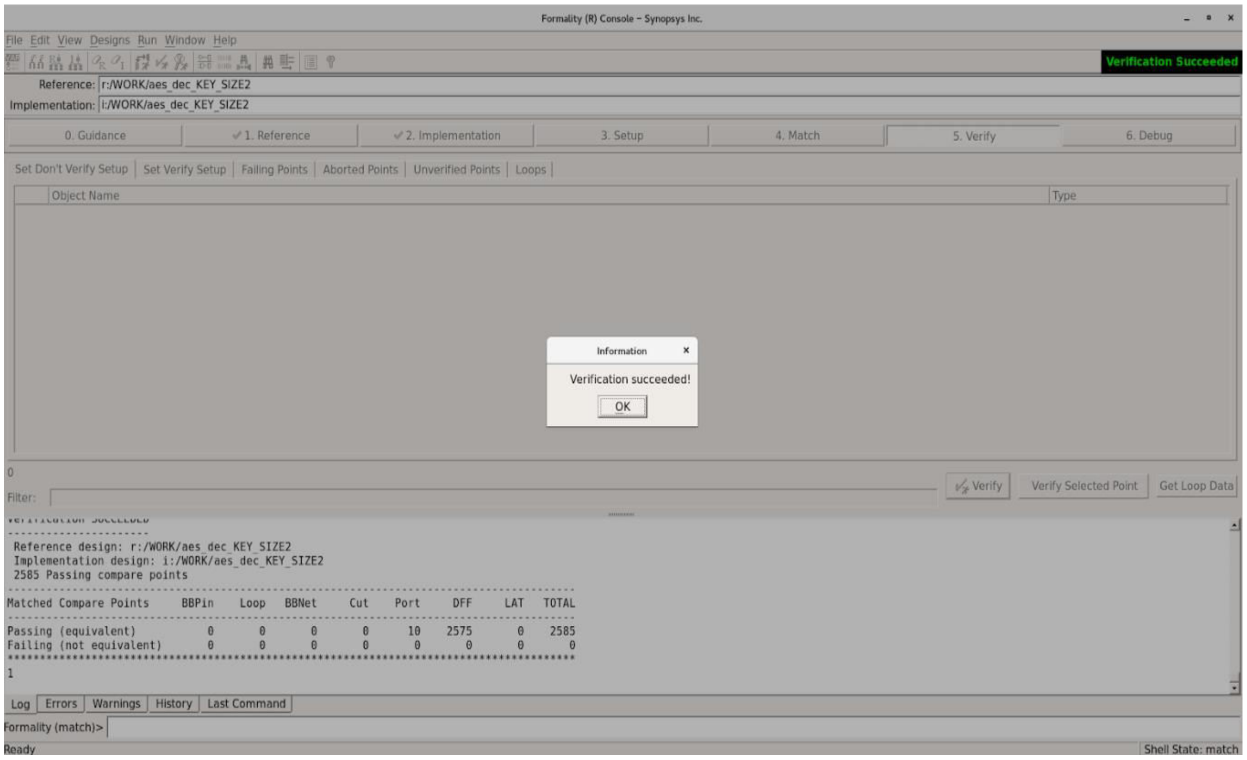


Figure 13. Traced error candidate in Formality

One of the merits of Formality is that, it explicitly displays whether the verification is successful or not. In Conformal, only comparison between key points, passing and failing points are mentioned in the scripts.



```
----------------------
 Reference design: r:/WORK/aes_dec_KEY_SIZE2
 Implementation design: i:/WORK/aes_dec_KEY_SIZE2
 2585 Passing compare points
--------------------------------------------------------------------------------------
Matched Compare Points     BBPin    Loop   BBNet     Cut    Port     DFF     LAT   TOTAL
--------------------------------------------------------------------------------------
Passing (equivalent)           0       0       0       0      10    2575       0    2585
Failing (not equivalent)       0       0       0       0       0       0       0       0
**************************************************************************************
1
```

Figure 14. Successful Verification message in Formality

4. VERIFICATION PROCESS IN FORMALITY/CONFORMAL

In order to approve two designs or technology libraries functional equivalence, Synopsys formality is among the very well-known tools. For design verification in formality, only 2 files (reference file known as golden and implementation file) is needed. Technology library verification is the same except for the cell to cell comparison between 2 files. After proving the equivalency of the implementation design, it can be used as a reference design for the next step. The overall verification time is reduced because formality can compare designs which are structurally similar in a better way than the ones that are not.

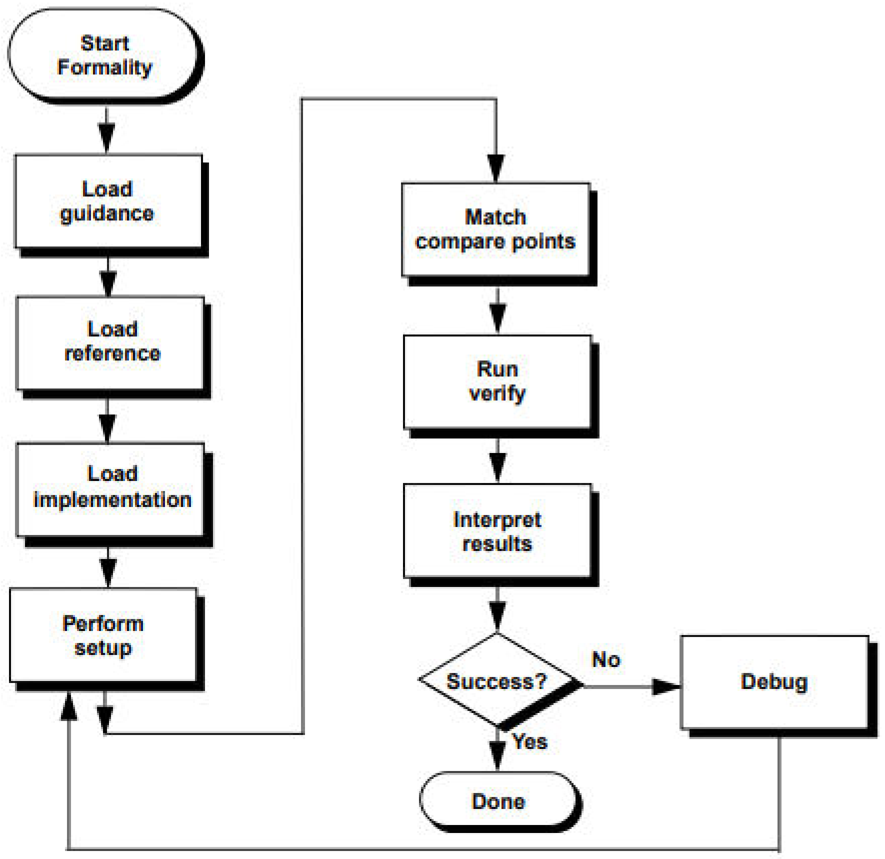


Figure 15. Design verification process flow overview [4]

In Formality, two working environments exist: the Formality Shell (a command-line-based user interface) and the Formality GUI (a Graphical windows-based interface). In this project, the latter is used for start. Before adding reference and implementation files to the project, loading an automated setup file (.svf) into formality can be done optionally. This file can be useful in terms of verification process and helping to align the compared points of the verifying design. In order to load the reference and implementation files, according to their types a proper option must be selected. After loading the files, the files are transferred between context panes. Next step is done generally for setting design-specific options to help with the Formality verification process and optimization of the design verification such as black boxes, external constraints, etc. One step before verification is Matching Compared points which is done automatically by specifying the “Verify” command. At this stage if any unmatched point are detected, it means the two designs are not equivalent and must be troubleshooted. Having all the steps matched, verification provides a report on non-equivalency. These unverified points are due to either an incorrect setup or a logical design difference between the two designs. Based on the provided information by Formality, it can be decided what the problem is and then the source of the issue can be traced back.

Similarly, Cadence Conformal does the same steps in order to get the 2 designs equivalent. By comparing the “key points” (primary inputs, primary outputs, latches, etc.) of the two designs the Conformal decides if they are equivalent or not.

5. ANALYSIS OF VERIFICATION RESULTS

The verification step in Synopsys Formality displayed 20 non-equivalent points. Whereas, Cadence Conformal showed 88 non-equivalent points. Diagnosing each error and correcting it, often led to the reduction of the number of error by a large margin. This implies that each bug can lead to non-equivalency at other points in the circuit. Both Formality and Conformal detected 11 bugs and corresponding changes were made. Table 2. Shows all the bugs and corresponding replacements.

Line Number	Failed points	Reference Design	Implementation Design	Replacement
5533	U146	U146 : ND2I port map(A => n154, B => n153, Z => n114);	U146 : NR2I port map(A => n154, B => n153, Z => n114);--changing NR2I with ND2I	Replace NR2I (2 input NOR) BY ND2I (2input NAND)
5564	U203	U203 : ND2I port map(A => n164, B => n2325, Z => n159);	U203 : AN2I port map(A => n164, B => n2325, Z => n159);-- changing AN2I with ND2I	Replace AN2I(2 input AND) BY ND2I(2 input NAND)
5638	U1846	U1846 : NR2I port map(A => n6798, B => n6799, Z => n1781);	U1846 : AN2I port map(A => n6798, B => n6799, Z => n1781)	Replace AN21(2 input AND) BY NR2I(2 input NOR)
15462	U2013	U2013 : OR3 port map(A => n4780, B => n4558, C => n4706, Z => n1961);	U2013 : AN3 port map(A => n4780, B => n4558, C => n4706, Z => n1961);	Replace AN3(3 input AND) BY OR3(3 input OR)
15463	U2018	U2018 : OR3 port map(A => n4571, B => n4450, C => n4706, Z => n1964)	U2018 : AN3 port map(A => n4571, B => n4450, C => n4706, Z => n1964);	Replace AN3(3 input AND) BY OR3(3 input OR)
15511	U3084	U3084 : AN3 port map(A => n2986, B => n2987, C => n2932, Z => n2985);	U3084 : OR3 port map(A => n2986, B => n2987, C => n2932, Z => n2985);	Replace OR3(3 input OR) BY AN3(3 input AND)
15552	U112	U112 : ENI port map(A => n132, B => n133, Z => n128);	U112 : EOI port map(A => n132, B => n133, Z => n128);	Replace EOI(2 input XOR) BY ENI(2 input XNOR)
15561	U121	U121 : EOI port map(A => n152, B => n153, Z => n151);	U121 : NR2I port map(A => n152, B => n153, Z => n151);	Replace NR2I(2 input NOR) BY EOI(2 input XOR)

15579	U152	U152 : EOI port map(A => n212, B => n213, Z => n211);	U152 : ENI port map(A => n212, B => n213, Z => n211);	Replace ENI(2 input XNOR) BY EOI(2 input XOR)
16777	U2384	U2384 : NR2I port map(A => n2075, B => n4361, Z => n2241);	U2384 : ND2I port map(A => n2075, B => n4361, Z => n2241);	Replace ND2I(2 input NAND) BY NR2I(2 input NOR)
16856	U2538	U2538 : EOI port map(A => n3840, B => n4655, Z => n423);	U2538 : ND2I port map(A => n3840, B => n4655, Z => n423);	Replace ND2I(2 input NAND) BY EOI(2 input XOR)

Table 2. Error candidates and their correction

5.1 Error Reduction in Conformal/ Formality

As observed, a small amount of error can introduce more errors in the circuit, Conformal detected 88 faulty points where as formality detected 20 nonequivalent points. Upon solving the bug by tracing the path of the gates backwards, number of faulty point reduced as follows:

Tools	Error Reduction											
Conformal	88	66	57	27	17	16	15	14	9	5	3	0
Formality	20	20	20	20	20	20	20	13	7	6	5	0

Table 3. Error reduction comparison between the two tools

CONCLUSION

The first step of the project is to provide a Reference file which is done by synthesizing the RTL code to gate level file. Once the synthesized file is obtained, Synopsys Formality and Cadence Conformal are used to perform equivalence checking between RTL to Gate level files and Gate level to Gate level files. Here some of each tool merits and demerits are mentioned:

Formality	Conformal
• Easier for beginner to understand the function of the tool • Library file has to be loaded every time a bug is corrected • Crashes several time due to memory consumption • Graphical representation of the schematics is clearer • Bug hunting is easier because of better graphical representation while tracing back the error • Implementation file has to be edited outside of the tool. • Shows LESS equivalent points • Less time consumption is finding the bug • Success or failure of the verification is explicitly mentioned	• More suitable for professionals • Library files are just loaded for both implementation and reference files only once • Operates smoothly and without crashes • Graphical representation of the schematic is presented in very big scales which make it hard to trace back the buggy gate • Implementation file can be edited in the tool itself • Shows all the non-equivalence points • Doesn’t provide a separate list for non-equivalence points and equivalence points • More time consumption in finding the bugs • No explicit message of the successful verification

Table 4. Comparison between Formality and Conformal

CHALLENGES

While working with Design-Vision, the gate level file couldn't be synthesized correctly from the RTL file. The reference file that was obtained from this tool, was compared with the RTL file in Formality and the Conformal. Even though, the matching process was successful, the schematic view did not show the identical gate names and numbers. Also, the inputs and outputs from the sequential circuits (D flip flops) were not similar. One major issue was that the order of the corresponding files of the project, was incorrect.

REFERENCES

[1] FPGA implementation of dual key based AES encryption with key Based S-Box generation, L.S. Abhiram, B.K. Sriroop, L. Gowrav, Kumar H.L. Punith, Manjunath C. Lakkanvar, 2nd INDIACom, 2015

[2] Implementation of Advanced Encryption Standard Algorithm with Key Length of 256 Bits for Preventing Data Loss in an Organization, Owusu Nyarko- Boateng, Michael Asante, Isaac Kofi Nti, International Journal of Science and Engineering Applications Volume 6 Issue 03, 2017

[3] AES description, Lattice semiconductor, April 2013

[4] Formality User Guide, Version Z-2007.06

Form ENCS-001(09/03)

Report

Confirmation of Originality

Faculty of Engineering and Computer Science

Course Name & Number/Term:________________**Section:**_____**Instructor:**___________________________
e.g., ENGR410/2 e.g. M

Having researched and prepared this report for submission to the Faculty of Engineering & Computer Science, the undersigned certify that the following statements are to the best of my/our knowledge true:

1. The undersigned have written this report myself/themselves.
2. This report consists entirely of ideas, observations, references, information and conclusions composed or paraphrased by the undersigned, as the case may be, except for statements contained within quotation marks and attributed to the best of my/our knowledge to their proper source in footnotes or otherwise referenced.
3. With the exception of material in appendices, the undersigned have endeavored to ensure that direct quotations make up a very small proportion of the attached report, not exceeding 5% of the word count.
4. Each paragraph of this report that contains material which the undersigned have paraphrased from a source (print sources, multimedia sources, web-based sources, course notes or personal interviews, etc), has been identified by numerical reference citation.
5. All of the sources that the undersigned consulted and/or included in the report have been listed in the Reference section of the report.
6. All drawings, diagrams, photos, maps or other visual items derived from sources have been identified by numerical reference citations in the caption.
7. Each of the undersigned has revised, edited and proofread this report individually.
8. In preparing this report the undersigned have read and followed the guidelines found in Form and Style, by Patrick MacDonagh and Jack Borden (Fourth Edition: May 2000), available at http://www.encs.concordia.ca/scs/Forms/Form&Style.pdf.

Name:_________________________ ID No:___________ Signature:______________________ Date:_________
(please print clearly)

Name:_________________________ ID No:___________ Signature:______________________ Date:_________
(please print clearly)

Name:_________________________ ID No:___________ Signature:______________________ Date:_________
(please print clearly)

Name:_________________________ ID No:___________ Signature:______________________ Date:_________
(please print clearly)

Name:_________________________ ID No:___________ Signature:______________________ Date:_________
(please print clearly)

Name:_________________________ ID No:___________ Signature:______________________ Date:_________
(please print clearly)

Do Not Write in this Space – Reserved for Instructor