FORMAL HARDWARE VERIFICATION OF AES DECODER

Equivalence Checking by Cadence Conformal And Synopsys Formality

Under the supervision of Professor **SOFIENE TAHAR**

Presented By: Group B

Zahidul Amin 40031489

Arihant Jain 40067961

Melany Valder 40054184

Elnaz Ghafaradli 40021804

INTRODUCTION

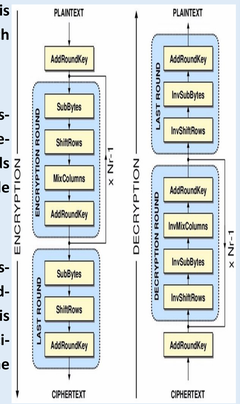
* In todays world data privacy and security is becoming very important factor in building trust between users. Advanced Encryption Standard, allows us to build that trust.
* As digital circuits become more larger and complex its is not easy to verify circuits using traditional simulation technique.
* Formal Hardware Verification comes in to action to make sure we create a bug free device.
* We focus on Tools Like Synopsys Formality and Cadence Conformal to apply Equivalence Checking technique on Reference and Implantation Design .

BEHAVORIAL 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 .

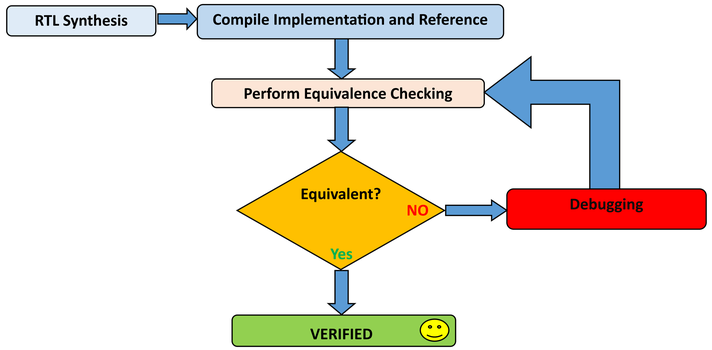
The Decoder has three units, the processor interface which oversees all the registers for communication between the blocks. The Key Expansion module is responsible for manipulating the Key data and finally the decryption module holds necessary data including AES states and converts the ciphered text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

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.



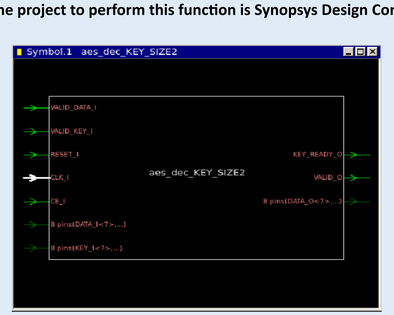
FORMALITY VS CADENCE

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



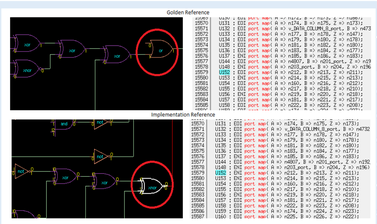
RTL to NETLIST

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.

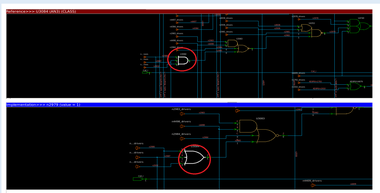



```
Inferred memory devices in process
        in routine aes_dec_KEY_SIZE2 line 422 in file
                '/nfs/home/e/e_gh/Documents/Desing_vision/design_vision_6/design/aes_dec.vhdl'.
===================================================================
|    Register Name    |    Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
===================================================================
|   CALCULATION_reg   | Flip-flop  |   1   |  N  |  N |  N |  N |  N |  N |  N |
===================================================================

Inferred memory devices in process
        in routine aes_dec_KEY_SIZE2 line 440 in file
                '/nfs/home/e/e_gh/Documents/Desing_vision/design_vision_6/design/aes_dec.vhdl'.
===================================================================
|     Register Name     |    Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
===================================================================
|     i_ROUND_reg       | Flip-flop  |   4   |  Y  |  N |  N |  N |  N |  N |  N |
| v_CALCULATION_CNTR_reg | Flip-flop |   8   |  Y  |  N |  N |  N |  N |  N |  N |
|     LAST_ROUND_reg    | Flip-flop  |   1   |  N  |  N |  N |  N |  N |  N |  N |
===================================================================

Inferred memory devices in process
        in routine aes_dec_KEY_SIZE2 line 477 in file
                '/nfs/home/e/e_gh/Documents/Desing_vision/design_vision_6/design/aes_dec.vhdl'.
===================================================================
|    Register Name    |    Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
===================================================================
|  STATE_TABLE1_reg   | Flip-flop  |  128  |  Y  |  N |  N |  N |  N |  N |  N |
|     DATA_O_reg      | Flip-flop  |   8   |  Y  |  N |  N |  N |  N |  N |  N |
|     VALID_O_reg     | Flip-flop  |   1   |  N  |  N |  N |  N |  N |  N |  N |
===================================================================
Presto compilation completed successfully.
Elaborated 1 design.
Current design is now 'aes_dec_KEY_SIZE2'.
design_vision> Current design is 'aes_dec_KEY_SIZE2'.
```

DEBUGGING PROCESS



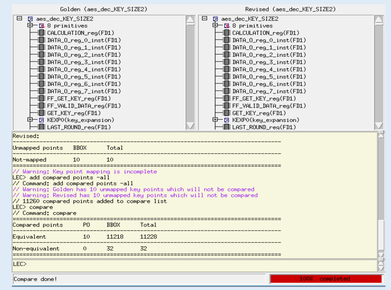
The Cadence Conformal debugging process starts by clicking on the mismatched gates marked in red on the table of conformal, The circuit is analyzed and backward tracing is done until difference in gates is found, as one bug in design can introduce many errors it is not possible to identify the exact 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.

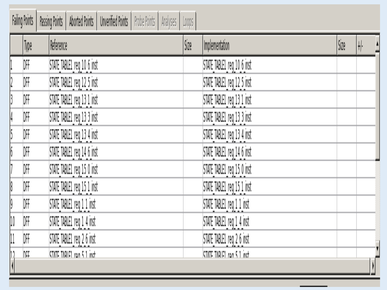


Similarly in formality the failing points tabulated shows which are the points where the output of the gates do not match. Then the failing point is selected and the path is traced to detect the buggy gate.

EQUIVALENCE CHECKING

Equivalence checking is the process of verifying that the two circuits we have designed are equivalent. 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. Following figures shows failing points in Conformal & Formality





RESULT ANALYSIS

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 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 AN2I(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)

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.