iJRASET
International Journal For Research in Applied Science and Engineering Technology

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

**Volume:** 6 **Issue:** VI **Month of publication:** June 2018

**DOI:** http://doi.org/10.22214/ijraset.2018.6121

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887*
*Volume 6 Issue VI, June 2018- Available at www.ijraset.com*

# An Approach for Generating the Self-Checking Test-Bench

Dr. Manju Nanda[1], P.Rajshekhar Rao[2]
[1]*(Aerospace Electronics & Systems Division, CSIR- NAL, Bangalore, India)*
[2]*(Department of Avionics, Inst. Of Science & Technology, JNTU, Kakinada, India)*

***Abstract:*** ***To verify the complex functionalities for an IP core or testing of critical IP core under module level testing giving more complexity at the time of simulation to analysis the output at accurate level due this fact, the engineer takes more time to finalize the outputs at different level. Due to that, the complexity of IP core at market level gets down. Complex critical functionalities and to verify the IP core as per Do-254, in any case, confirmation of necessities by test during board testing is challenging and time-consuming in some cases like normal, boundary and robustness test benches. This paper clarifies the critical functionalities with one of technique i.e. self checking test bench, and gives proposals how to beat them. The efficiency of the self-checking test bench is demonstrated with FIFO as case study.***

***Keywords: DO-254, FPGA, Assertion, FIFO, Coverage***

## I. INTRODUCTION

Do-254/ED-80 is a method for consistence to flight directions for all aeronautical electronic equipment named custom small scale coded devices for example FPGA, ASICs and PLDs, these devices are frequently as unpredictable as programming controlled microchip based frameworks along these lines they require a stringent improvement way to deal with fulfil airworthiness necessities. The principle motivation behind the direction is to guarantee that the device assembled meets the necessities and securely plays out all proposed works under typical and unusual working conditions [1]. Keeping in mind the end goal to get consistence the candidate must actualize the stringent advancement and check procedure of do-254 and fulfill the fundamental goals at the target level. The structure of hardware design life cycle and the information created in the process are for and particular to the device itself.

To verify the design under test (DUT) as per DO-254 consistently is basic in demonstrating that the device meets the necessities. In any mode of testing for device at target level (normal, boundary, and robustness test cases) is very crucial role to confirming the particular pre-requisites for a prohibitive action take place. This document encloses the purposes for these difficulties are examined and proposals how to defeat them are given. The methodology for testing the FPGA objectives at target level to be accurate, it should be done with one of the technique like self checking test bench [2].

## II. THE CHALLENGE

The standard way to deal with check electronic segments and guarantee legitimate practical conduct is to utilize simulation tool. The check/verify is then in view of timing models of the constitutive components of a FPGA. The device may be demonstrated to meet the do-254 level A desires. Implementation in FPGA using self checking test bench for accuracy needs to satisfy the customer needs as well to increase the percentage of FPGA safety critical design [6].

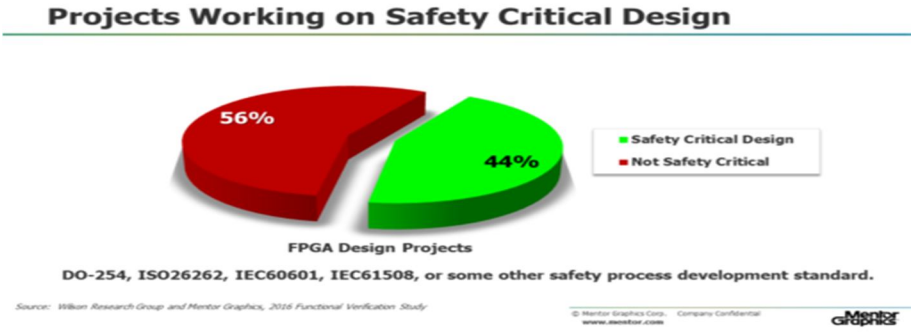



Fig. 1. Percentage degradation for Avionics critical system as per DO-254[7]

©IJRASET: All Rights are Reserved

*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887*
*Volume 6 Issue VI, June 2018- Available at www.ijraset.com*

## III. CHECKING/VERIFICATION PROCESS

The confirmation steps are indispensable and a key component to demonstrating that the device fabricated meets the requirement functionality. The confirmation procedure gives a specialized evaluation of rightness of the plan against the necessities. The key factor in planning an effective confirmation design/verification design is to comprehend the reason destinations and exercises of the confirmation steps as characterized in the direction

The logic for checking the procedure for confirming that device should map the pre-requisites. The design under target level of testing should convince the applicant with that must fulfil the necessities. The confirmation/verification exercise with the combination of audits, examinations and tests as prescribed by the applicant in the confirmation of design and it should be follow during all periods of life cycle of hardware.

Validation and Verification Process ensures that test cases, methods, procedures, pass/fail criteria and results/coverage are reviewed to determine whether the developed test cases, methods, procedures, pass/fail criteria satisfy the objectives of RTCA/DO254[8].

Verification of the implementation is the verification (e.g. post layout simulations) of the Detailed Design after place and route and of the device itself.

### *A. Review and analysis*

The entry criteria to perform the review of test cases, methods, procedures, pass/fail criteria and results/coverage are release of initial version of those data items and other supporting hardware verification data will be under configuration control as appropriate. Designers will perform the review of these test cases, methods, procedures & pass/fail criteria and provide the feedback to the testing team [10].

The verification on device level, to assess unacceptable robustness defects through RBT (requirement based testing) to cover the normal and abnormal input conditions and operating conditions. This document will be prepared to justify for level of implementation (RTL, post layout, physical device, board level) and the type of the planned verification activities (test, simulation, analysis, inspection etc.,)using self checking test bench as to improve the performance of the design.

At the post-design level of confirmation for the HDL level verification which is done by means of HDL coding standard (HCS) which potentially records the dynamic timing simulation (DTS) and static timing simulation (STS) and essential examination which can be investigate with code coverage under design level test or the verification of design under test inside the element is done by means of device testing. Remember that all design or elements are required to confirm all FPGA necessities by test which is the genuine trail for the applicant [11].

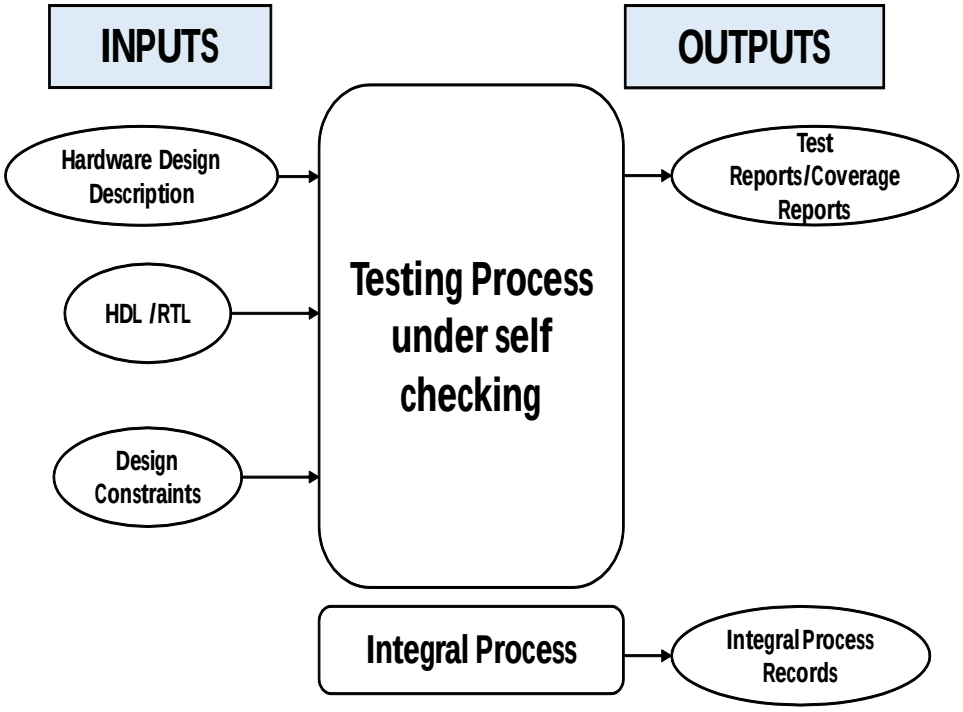



Fig. 2. Flow of testing process under self checking [12]

©IJRASET: All Rights are Reserved

As per DO-254, it guides the particular idea to complete the entire design which encloses the requirement to DUT level of testing, considering all modes of scenario test cases. To achieve pass scenario at module and integration level of testing should verify with the expected outputs via code coverage analysis.

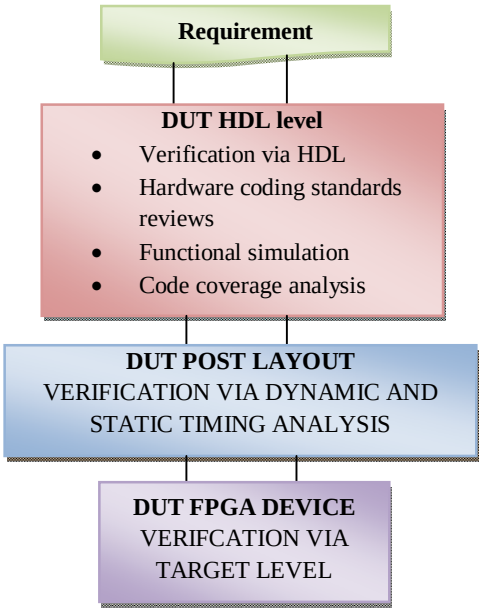



Fig. 3. Flow of Verification of Design And Implementation [13]

*B. Developing Acceptance Coverage by Test*

The following is an overall of outline comments/suggestion that can be executed to expand check by test. The outline of suggestions proposes an approach that can use a similar experiments and test contributions for recreation and target testing considering a substantially more grounded contention of check integrity and legitimacy to the affirmation expert.

*1)* Compose the prerequisites with the end goal that they are obvious at the FPGA stick level. This guarantees the inputs are controllable and the outputs are detectable with the goal that every necessity can be checked by test at the stick level.

*2)* Make test cases should follow the prerequisites, not the design. Experiments portray how to confirm the necessities.

*3)* Make the HDL test bench to actualize the experiments for recreation. The contributions from the test case are connected to the HDL plan and outputs are gathered by means of waveforms. Self-checking test benches can likewise be utilized to computerize checking for PASS/FAIL comes about.

*4)* The test bench can be utilized for utilitarian recreation and code scope examination. A similar test vectors are utilized for useful reproduction, utilitarian re-enactment with scope measurements, post-design timing reproduction and for device testing [14]

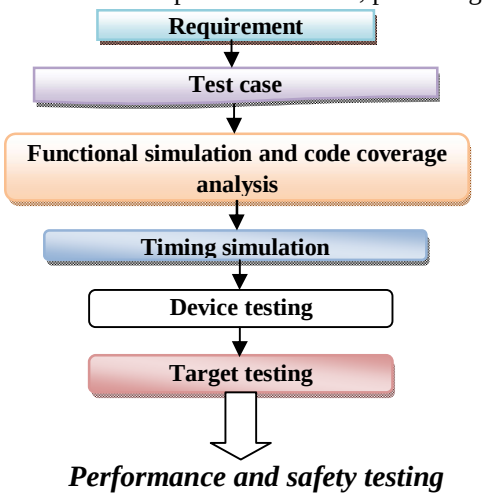



Fig. 3. Flow of FPGA design safety critical [15]

©IJRASET: All Rights are Reserved

*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887*
*Volume 6 Issue VI, June 2018- Available at www.ijraset.com*

As a FPGA developer, your life is fundamentally one huge investigates cycle. From the minute you get almost priority from marketing, until creation silicon is prepared, you are searching for issues. When you perceive an issue, at that point remedial activity must be recognized.

*a)* Create an empty test bench

```
entity tb_adder_comb is
.....
end tb_adder_comb;
```

*b)* Add an architecture

```
[header files]
entity tb_adder_comb is
.....
end tb_adder_comb;
architecture test of tb_adder_comb is
begin
end architecture test;
```

*c)* Instantiate the design/unit under test (DUT/UUT)

```
Entity tb_adder_comb is
generic(
N_BITS: positive range 2 to positive'right
);
port (op1:.....N-bit input;
op2:.....N-bit input;
sum:.....(N+1)-bit output
);
```

*d)* The test bench is updated as

```
-create a constant for every generic parameter in the DUT/Uut and assign a value.
-create a signal for every port that is input abd output port
--generics
constant N-bits:positive range 2 to positive'right:=4
--ports
--inputs
signal op1:.....;
signal op2:.....;
--outputs
signal sum:.....;
-- instantiate the DUT
DUT: entity work.adder_comb
generic map(N_BITS=>N_BITS)
port map (OP1=>OP1,
OP2=>OP2,
SUM=>SUM);
```

*e)* Input feeding

```
--inputs from vector1(op1,op2)
--outputs from vector1(sum)
--inputs from vector2(op1,op2)
--outputs from vectors2(sum)
constant time_delta time:=100ns;
--generic
--ports
-- instantiate the DUT
```

©IJRASET: All Rights are Reserved

```
--test_adder_comb
simulation:process
begin
--current values
op1<=”000”;--1
op2<=”0101”;--5
wait for time delta;
end process simulation;
end architecture;
```

*f)* To prevent duplication of the code we add a procedure

```
procedure check_add
(constant in 1:in natural;
constant in 2: in natural)is
begin
--current values
op1<= std_logic_vector(to_unsigned(in1,op1’length));
op2<=std_logic_vector(to_unsigned(in1,op2’length));
wait for time delta;
end process;
```

*g)* The self checking test bench has two main parts;

-a report statement and string describing the error using and assert statement.

-a severity statement describing the level of the issue that is note,warning, error or failure.

```
Procedure check_add
(constant in 1: in natural;
constant in 2: in natural;
constant res_expected:in natural:in natural)is
variable res:natural;
begin
--current values
op1<=std_logic_vector(to_unsigned(in1,op1’length));
op2<=std_logic_vector(to_unsigned(in1,op2’length));
wait for time delta;
res:=to_integer(unsigned(sum));
assert res=res_expected;
report”unexpected results.’&”op1=”&integer’_image(in1)&”,”&
.
.
.(repeat for all the other ports)
severity error
end procedure check_add
begin
check_add(12,8,20)--will show a sucess full result
check_add(12,8,21)—will show a failure
.
.
wait
--end process;
--end architecture
```

©IJRASET: All Rights are Reserved

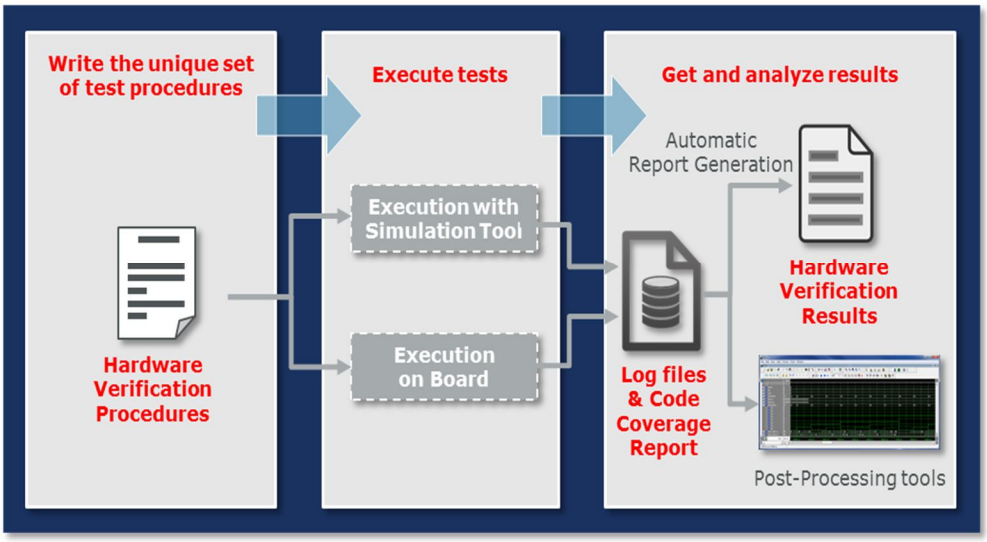



Fig. 4. Procedure of Self-Checking Test-bench [17]

Self-checking test bench (SCTB) has three primary records: the first HDL demonstrates the last gate level demonstrates and a reciprocity block. The test bench (TB) works by contrasting the outcomes from the two FPGA models which both get a similar input i/p stimulus and after that errors any disparities that are found. The genuine power in this strategy is that when a mistake is discovered you can test into the two models and follow signs to perceive what is causing the issue [17]

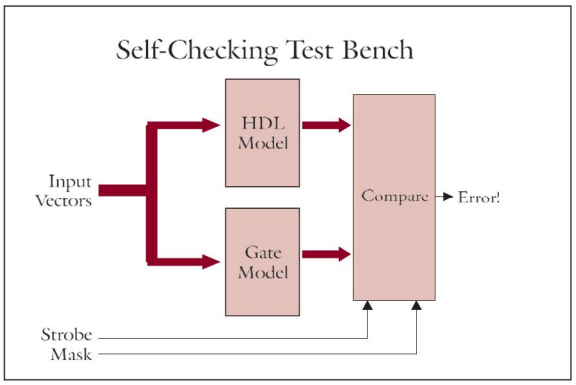



Fig. 5. Structure of Self Checking Test bench [17]

*C. Trace Data Activities in Detailed Design & Implementation Process*

*1)* Link the specific lines of the HDL design source (single line or multiple lines of code) to the related FPGA requirement it covers.

*2)* Generate a downstream traceability to ensure that each FPGA requirement is fully implemented by an HDL function. FPGA designers must create additional functions as needed to fully implement each FPGA requirement.

*3)* Generate an upstream traceability to expose unused HDL design functions. Unused functions of the HDL code may lead to unexpected behavior of the device, and must be removed or updated.

*4)* Ensure that the post-synthesis and post-layout design meet the specified constraints.

*5)* Baseline or record the trace data between FPGA requirements and HDL design data.

*D. Trace Data Activities in Verification process*

*1)* Link each verification test scenario to the related FPGA requirement it covers. Test scenarios are created based on the FPGA requirements, and they are reviewed for suitability and completeness to cover the related FPGA requirement. Test scenarios define how each FPGA requirement will be verified, and includes the appropriate input conditions, test sequence and expected results.

©IJRASET: All Rights are Reserved

*2)* Generate a downstream traceability to ensure each FPGA requirement is covered by a test scenario.
*3)* Generate an upstream traceability to expose unnecessary test scenarios.
*4)* Baseline or record the trace data between FPGA requirements and test scenarios.
*5)* Link the specific lines of the test bench (single line or multiple lines of code) to the related test scenario it covers.
*6)* Generate a downstream traceability to ensure that each test scenario is fully implemented by the test bench. Verification engineers must create the test bench with the appropriate test inputs, sequence and expected results as defined in the test scenarios.
*7)* Generate an upstream traceability to expose unused functions or code of the test bench that needs to be removed or updated.
*8)* Baseline or record the trace data between test scenarios and test bench.
*9)* Link the specific simulation results to the related test scenario it covers. Simulation results are a combination of simulation logs, waveforms and coverage data. These results are analyzed and reviewed for correctness.

High level of verification like self-checking test bench is an awesome method for playing out that last check organizes previously making silicon. It gives you the additional certainty that the design will be right and easily roll into generation. Coverage is observed through various methodologies. Code scope and functional scope are utilized as correlative strategies together. also assertion/comparison based scope is a reasonable technique to implement and verify the critical design behaviour.

## IV. CASE STUDY- FIFO

For safety and performance criteria of self checking test bench, hereby to implement and verify a module like FIFO using self checking approach, for better time to market and customer supports. The process needs to proceed the module level testing like FIFO which is need to test the functionality of that particular module which tells input and output behavioural, according to that need to verify the self checking methodology.
The self checking methodology can be placed to normal, boundary and robustness test cases which include:

*A.* Performance
*B.* Safety analysis
*C.* Low latency
*D.* Analysis of Clock Skew, drift factor

*E. Functionality Description*

*1)* Data FIFO block shall be used for storing the user defined data of particular input source
*2)* There shall be a 32 bit data bus to write the user defined data
*3)* Data FIFO block shall an i/p AXI-clock for its internal operation
*4)* Data FIFO shall have a width of 32-bit
*5)* FIFO depth in this block using 16000 words
*6)* There shall be a tvalid input signal to data FIFO, to indicate the valid user defined data
*7)* There shall be a tlast input signal to data FIFO, to indicate the last word (32 bit) of user defined data
*8)* There shall be a tready o/p signal from the data FIFO.

Note 1: t ready set to high shall indicate, FIFO is ready for receiving AXI stream data of 32 bit.
Note 2: t ready set to low shall indicate, FIFO is not ready for receiving AXI stream data.
Below table indicates the following requirement analysis for FIFO as per DO-254

TABLE 1 REQUIREMENT ANALYSIS

| Requirement Identification | Interface Name | Width (in bits) |
|---|---|---|
| XXXXXXXX | FIFO module shall be interfaced with following port for reading from buffer: | |
| | RdClk_i | 1 |
| | rd_en_i | 1 |
| | rd_data_o | 32 |

©IJRASET: All Rights are Reserved

| | |
|---|---|
| XXXXXXXX | FIFO module shall be interfaced with following port for writing from buffer: |

| | |
|---|---|
| WrClk_i | 1 |
| Wr_en_i | 1 |
| Wr_data_o | 32 |

## V. RESULTS

Step1: Firstly generation of the RTL for the function of the module

Step2: Secondly, feed the input values in HEX values in stimulus as FILE i/o assertion based testing

file vector_file:text open read_mode is "XXXXXXXXXX";--reading the expected results from read text file

file result_file:text open write_mode is " XXXXXXXXXX ";--writing the compared results in the write text file

--variables Declaration

variable file_line: line;

variable str_stim_in :string(32 downto 1);

variable stim_in :std_logic_vector(31 downto 0);

variable x : std_logic_vector(31 downto 0);

variable y :integer;

variable result_line:line;

variable i :integer :=0;

Step3: Thirdly, as per feeding the input values check the expected values as per feed values which gives the accuracy as per design and engineer will satisfy the requirement given by the organisation. This step tells about the simulation results is PASS/FAIL condition as per analysis.

Step4: Finally as per simulation based testing or assertion included testing which shows PASS/FAIL condition, the engineer or designer come to know the timing analysis of the design for accuracy which gives performance and safety analysis.

### *A. Expected inputs/ feed values*

```
00000000000000000000000000001001
00000000000000000000000000001010
00000000000000000000000000001011
00000000000000000000000000001100
00000000000000000000000000001101
00000000000000000000000000001110
00000000000000000000000000001111
00000000000000000000000000010000
00000000000000000000000000010001
00000000000000000000000000010010
00000000000000000000000000010011
00000000000000000000000000010100
00000000000000000000000000010101
00000000000000000000000000010110
00000000000000000000000000010111
00000000000000000000000000011000
00000000000000000000000000011001
00000000000000000000000000011010
00000000000000000000000000011011
00000000000000000000000000011100
00000000000000000000000000011101
00000000000000000000000000011110
00000000000000000000000000011111
00000000000000000000000000100000
00000000000000000000000000100001
00000000000000000000000000100010
00000000000000000000000000100011
00000000000000000000000000100100
00000000000000000000000000100101
00000000000000000000000000100110
00000000000000000000000000100111
00000000000000000000000000101000
00000000000000000000000000101001
00000000000000000000000000101010
```

©IJRASET: All Rights are Reserved

*B. Expected outputs in Simulation Based Testing*

```
test bench
67275000 ps time is:67281250 ps, the output case2 is test case failed, not correct match:= 1
 expected value is 00000000000000000000000000000001 but its current simulation value is xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 *************** &&&&&&& ********
67281250 ps time is:67293750 ps, the output case2 is test case failed, not correct match:= 2
 expected value is 00000000000000000000000000000001 but its current simulation value is xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 *************** &&&&&&& ********
67293750 ps time is:67306250 ps, the output case2 is test case failed, not correct match:= 3
 expected value is 00000000000000000000000000000010 but its current simulation value is xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 *************** &&&&&&& ********
67306250 ps time is:67318750 ps, the output case2 is test case failed, not correct match:= 4
 expected value is 00000000000000000000000000000011 but its current simulation value is xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 *************** &&&&&&& ********
67318750 ps time is:67331250 ps, the output case2 is test case failed, not correct match:= 5
expected value is 00000000000000000000000000000100 but its current simulation value is xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 *************** &&&&&&& ********
67331250 ps time is:67343750 ps, the output case2 is test case failed, not correct match:= 6
expected value is 00000000000000000000000000000101 but its current simulation value is xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 *************** &&&&&&& ********
67343750 ps time is:67356250 ps, the output case2 is test case failed, not correct match:= 7
 expected value is 00000000000000000000000000000110 but its current simulation value is xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 *************** &&&&&&& ********
67356250 ps time is:67368750 ps, the output case2 is test case failed, not correct match:= 8
 expected value is 00000000000000000000000000000111 but its current simulation value is xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 *************** &&&&&&& ********
67368750 ps time is:67381250 ps, the output case2 is test case failed, not correct match:= 9
 expected value is 00000000000000000000000000001000 but its current simulation value is xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 *************** &&&&&&& ********
67381250 ps time is:67393750 ps, the output case2 is test case failed, not correct match:= 10
expected value is 00000000000000000000000000001001 but its current simulation value is xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 *************** &&&&&&& ********
67393750 ps time is:67406250 ps, the output case2 is test case failed, not correct match:= 11
expected value is 00000000000000000000000000001010 but its current simulation value is xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
 *************** &&&&&&& ********
```

The below results or cases shows the expected i/o values with cases when feed in to stimulus

Case1: When assertion based testing feed in to stimulus input using FILE i/p

# ** Note: Port mapped signal Areset_i is matched
    #  Time: 5 ns  Iteration: 1  Instance:
# ** Note: Port mapped signal wr_en_i is matched
#  Time: 5 ns  Iteration: 1  Instance:
# ** Note: Port mapped signal full_o is matched
#  Time: 5 ns  Iteration: 1  Instance:
# ** Note: Port mapped signal RdClk_i is matched
#  Time: 5 ns  Iteration: 1  Instance:
# ** Note: Port mapped signal rd_en_i is matched
#  Time: 5 ns  Iteration: 1  Instance:
# ** Note: Port mapped signal wr_data_i is matched  and the value is 00000000000000000000000000001010
#  Time: 5 ns  Iteration: 1  Instance:
# ** Warning: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
#  Time: 5 ns  Iteration: 1  Instance: /vdb_n
# ** Warning: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
#  Time: 5 ns  Iteration: 1  Instance:
# ** Note: Port mapped signal rd_data_o is matched  and the value is xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
#  Time: 5 ns  Iteration: 1  Instance:
# ** Note: Port mapped signal WrClk_i is matched
#  Time: 5 ns  Iteration: 1  Instance:
# ** Note: Port mapped signal Areset_i is matched
#  Time: 15 ns  Iteration: 1  Instance:
# ** Note: Port mapped signal wr_en_i is matched
#  Time: 15 ns  Iteration: 1  Instance:
# ** Note: Port mapped signal full_o is matched
#  Time: 15 ns  Iteration: 1  Instance:
# ** Note: Port mapped signal RdClk_i is matched
#  Time: 15 ns  Iteration: 1  Instance:
# ** Note: Port mapped signal rd_en_i is matched
#  Time: 15 ns  Iteration: 1  Instance:

©IJRASET: All Rights are Reserved

# ** Note: Port mapped signal wr_data_i is matched and the value is 00000000000000000000000000001010
# Time: 15 ns Iteration: 1 Instance:
# ** Warning: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
# Time: 15 ns Iteration: 1 Instance:
# ** Warning: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic operand, the result will be 'X'(es).
# Time: 15 ns Iteration: 1 Instance:
# ** Note: Port mapped signal rd_data_o is matched and the value is xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
# Time: 15 ns Iteration: 1 Instance:
# ** Note: Port mapped signal WrClk_i is matched
# Time: 15 ns Iteration: 1 Instance:
# ** Note: Port mapped signal Areset_i is matched
# Time: 25 ns Iteration: 1 Instance:
# ** Note: Port mapped signal wr_en_i is matched
# Time: 25 ns Iteration: 1 Instance: # ** Note: Port mapped signal full_o is matched
# Time: 25 ns Iteration: 1 Instance:
# ** Note: Port mapped signal RdClk_i is matched
# Time: 25 ns Iteration: 1 Instance:
# ** Note: Port mapped signal rd_en_i is matched
# Time: 25 ns Iteration: 1 Instance:
# ** Note: Port mapped signal wr_data_i is matched and the value is 00000000000000000000000000001010
# Time: 25 ns Iteration: 1 Instance:
# ** Note: Port mapped signal rd_data_o is matched and the value is 00000000000000000000000000000000
# Time: 25 ns Iteration: 1 Instance:
# ** Note: Port mapped signal WrClk_i is matched
# Time: 25 ns Iteration: 1 Instance:
# ** Note: Port mapped signal Areset_i is matched
# Time: 35 ns Iteration: 1 Instance:
# ** Note: Port mapped signal wr_en_i is matched
# Time: 35 ns Iteration: 1 Instance:
# ** Note: Port mapped signal full_o is matched
# Time: 35 ns Iteration: 1 Instance:
# ** Note: Port mapped signal RdClk_i is matched
# Time: 35 ns Iteration: 1 Instance:
# ** Note: Port mapped signal rd_en_i is matched
# Time: 35 ns Iteration: 1 Instance:
# ** Note: Port mapped signal wr_data_i is matched and the value is 00000000000000000000000000001010
# Time: 35 ns Iteration: 1 Instance:
Case2 : Port mapped signal rd_data_o is matched and the value is 00000000000000000000000000000000
# Time: 35 ns Iteration: 1 Instance:
# ** Note: Port mapped signal WrClk_i is matched
# Time: 35 ns Iteration: 1 Instance:
# ** Note: Port mapped signal Areset_i is matched
# Time: 45 ns Iteration: 1 Instance:
# ** Note: Port mapped signal wr_en_i is matched
# Time: 45 ns Iteration: 1 Instance:
# ** Note: Port mapped signal full_o is matched
# Time: 45 ns Iteration: 1 Instance:
# ** Note: Port mapped signal RdClk_i is matched
# Time: 45 ns Iteration: 1 Instance:
# ** Note: Port mapped signal rd_en_i is matched

©IJRASET: All Rights are Reserved

```
#    Time: 45 ns  Iteration: 1  Instance:
# ** Note: Port mapped signal wr_data_i is matched   and the value is 00000000000000000000000000001010
Case 3: Port mapped signal rd_data_o is matched   and the value is 00000000000000000000000000000000
#    Time: 195 ns  Iteration: 1  Instance:
# ** Note: Port mapped signal WrClk_i is matched
#    Time: 195 ns  Iteration: 1  Instance:
# ** Note: Port mapped signal Areset_i is matched
#    Time: 205 ns  Iteration: 1  Instance:
# ** Note: Port mapped signal wr_en_i is matched
#    Time: 205 ns  Iteration: 1  Instance:
# ** Note: Port mapped signal full_o is matched
#    Time: 205 ns  Iteration: 1  Instance:
# ** Note: Port mapped signal RdClk_i is matched
#    Time: 205 ns  Iteration: 1  Instance:
# ** Note: Port mapped signal rd_en_i is matched
#    Time: 205 ns  Iteration: 1  Instance:
Case4: Port mapped signal rd_data_o is matched   and the value is 00000000000000000000000000000000
#    Time: 205 ns  Iteration: 1  Instance:
# ** Note: Port mapped signal WrClk_i is matched
#    Time: 205 ns  Iteration: 1  Instance:
# ** Note: Port mapped signal Areset_i is matched
#    Time: 215 ns  Iteration: 1  Instance:
# ** Note: Port mapped signal wr_en_i is matched
#    Time: 215 ns  Iteration: 1  Instance:
# ** Note: Port mapped signal full_o is matched
#    Time: 215 ns  Iteration: 1  Instance:
# ** Note: Port mapped signal RdClk_i is matched
#    Time: 215 ns  Iteration: 1  Instance:
# ** Note: Port mapped signal rd_en_i is matched
#    Time: 215 ns  Iteration: 1  Instance:
# ** Note: Port mapped signal wr_data_i is matched
```

## VI. CONCLUSION

Before for testing the functionality of any individual module as per DO-254 gives a bad review from a customer side because of the engineer fails to verify the all modes of scenario's which tells the new method which is self checking test bench for performing the final target level testing or verification stage. It gives an accurate measurement to the design for production.

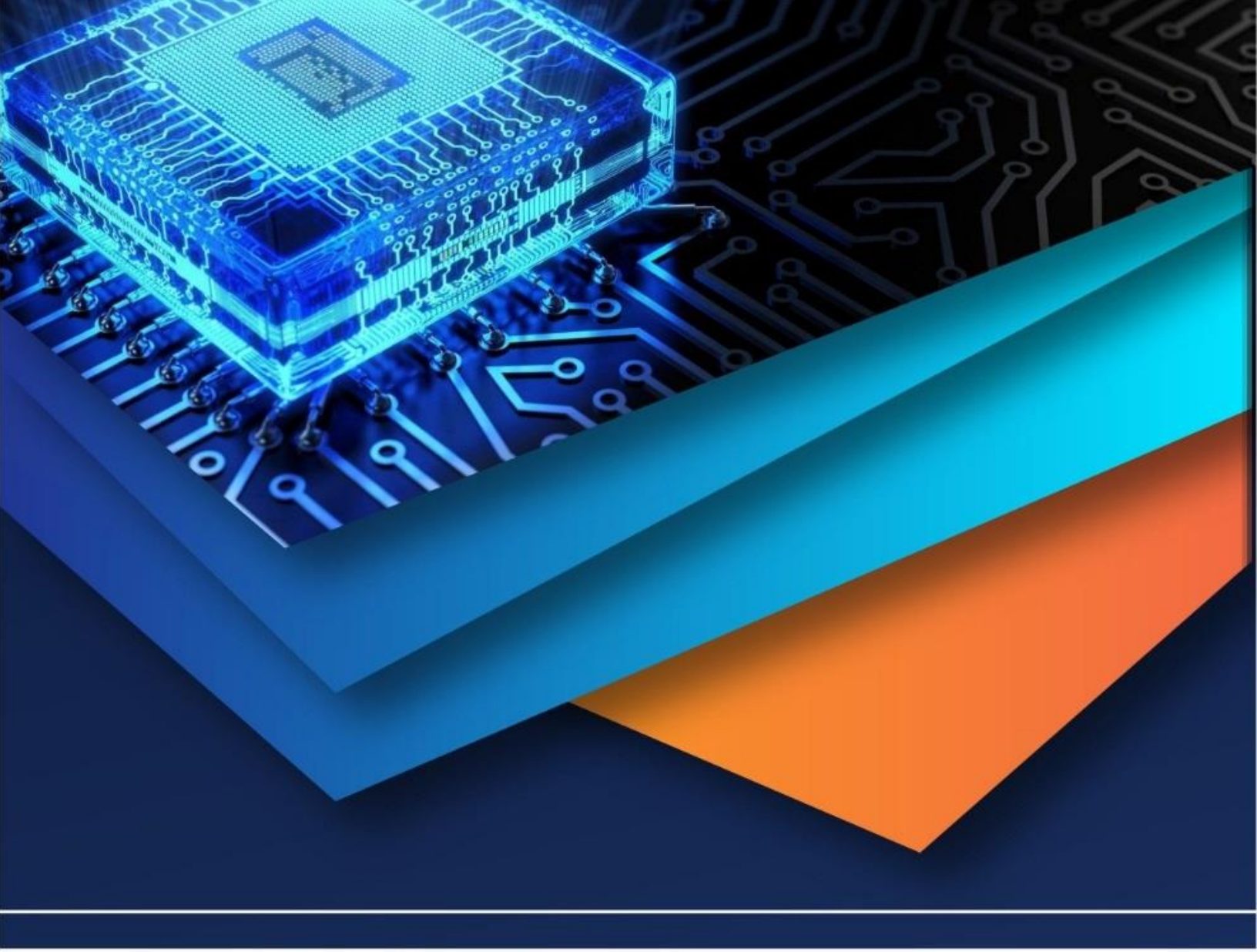
## REFERENCES

[1]. Molina and O. Cadenas, \Functional veri_cation: approaches and challenges,"Latin American applied research, vol. 37, no. 1, pp. 65-69, 2007.
[2]. Rath, V. Esen, and W. Ecker, \Analog transaction level modeling," in High Level Design Validation and Test Workshop (HLDVT), 2011 IEEE International.IEEE, 2011, pp. 82-82.
[3]. Cadence, \Using coverage, guiding veri_cation to e_cient completion," 2003.
[4]. C. Spear, SystemVerilog for veri_cation: a guide to learning the testbench language features. Springer Verlag, 2008.
[5]. J. Sordoillet and S. Davey, \Integrated, comprehensive assertion-based coverage," in EDA Tech Forum, vol. 3, no. 1, 2006, pp. 22-25.
[6]. B. Bhattacharya, J. Decker, G. Hall, N. Heaton, Y. Kashai, N. Khan, Z. Kirshenbaum, and E. Shneydor, Advanced Veri_cation Topics. lulu.com, 2012.
[7]. K. Karnane, G. Curtis, and R. Goering, \Solutions for mixed-signal soc veri_cation, as old methods fall short, new techniques make advanced soc veri_cation possible,"2012.
[8]. Bauer, M. and W. Ecker: "A VHDL-Based Hierarchical, Highly Flexible and Extendable Testbench Approach". International High Level Design Validation and Test Workshop, 1996.
[9]. Swavely, G.W., Beaton, J., and W. Debany: "A Generic VHDL Testbench to aid in Development of Board-Level Test Programm", AUTOTESTCON, 1994.
[10]. Heinkel, U. and W.H. Glauert: An Approach for a Dynamic Generation / Validation System for the Functional Simulation Considering Timing Constraints. European Design and Test Conference, 1996.
[11]. Zinn, A., Ecker, W., Bauer, M. and B. Bernard:"Comparison of Sequential VHDL and C Revised". International HDL Conference, 2001.

©IJRASET: All Rights are Reserved

[12]. Sztipanovits, J.: "Engineering of Computer-Based Systems: An Emerging Discipline", Conference and Workshop on Engineering of Computer-Based Systems, 1998.
[13]. Drager, S.L., Hanna, J.P., and R.G. Hillmann: "VHDL Model Verification and System Life Cycle Support", VHDL International User Forum, Spring 1996.
[14]. Goldbach M., Grams H., Glauert W., Hartl W. and Voit G.: "Simulation-Based Test Programm Verifiaction Using the SZ Testsystem Environment", IMSTW, 1998.
[15]. Garbe, H., Jentschel, H.J., and R. Kaminski: "Multilevel Simulation in the Design of Communication Systems", NE Science, 1994.
[16]. Delvai M., Huber W., Rahbaran B., and A. Steininger: "An FPGA-Based Development Platform for the Virtual RealTime Processor Component SPEAR". IEEE International Workshop on Design and Diagnostics of Electronic Circuits and Systems, 2002.
[17]. Lentz, K.P., Heller, J., and P.L. Montessoro: "System Verification using Multilevel Concurrent Simulation", IEEE Micro, 1999, p.60-67.
[18]. M. H. Schulz and E. Auth, "Improved deterministic test pattern generation with applications to redundancy identification," IEEE Trans. on CAD, Vol. 8, No. 7, July 1989, pp. 811-816.
[19]. Neil H.E. Weste and David Harris, CMOS VLSI Design: A Circuits and Systems Perspective (3rd Edition), Addison Wesley; 3rd edition (May 11, 2004), ISBN: 0321149017.
[20]. Y. Kim, M.-H. Yang, Y. Lee, and S. Kang, ―A new low power test pattern generator using a transition monitoring window based on BIST architecture,‖ in Proc. Asian Test Symp. (ATS), Dec. 2005, pp. 230–235.

©IJRASET: All Rights are Reserved



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)