JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

FUNCTIONALLY VERIFYING 16-BIT ALU USING SYSTEM VERILOG.

Priyadarshi Priyesh[1], Prajkta Dharmik[2], Shobhit Tiwari[3] ,Dr. P.G. Chilveri[4], Mrs. T. A. Mate[5]

[1]Student, Department of Electronics and Telecommunication SKNCOE,SPPU Pune, India, [2]Student, Department of Electronics and Telecommunication SKNCOE,SPPU Pune, India, [3]Student, Department of Electronics and Telecommunication SKNCOE,SPPU Pune, India, [4]Associate Professor, Department of Electronics and Telecommunication SKNCOE,SPPU Pune, India,[5]Asst. Professor, Department of Electronics and Telecommunication SKNCOE,SPPU Pune, India,

Abstract

Arithmetic and logical units are an integral part of any SOC (System on chip) that performs Arithmetic and logical operations. It performs operations like addition, subtraction, multiplication, division, AND operations, OR operations, etc. System Verilog is a hardware verification language (HVL) that is an advanced version of Verilog and System Verilog is based on previous Verilog standards and some additional functions which make the verification easy. The environment presented in this paper will provide a set of functional blocks through which the Design Under Test (DUT) will be passed and checked for various parameters. In this paper the verification of a sixteen-bit ALU in which the design under test will take an input of Sixteen bits and will be subjected to several operations is being discussed. The goal of the project is to cover the maximum number of test cases and attain the maximum coverage.

Keywords: **System Verilog, DUT, ALU, SOC, Questasim.**

I. Introduction

System Verilog is a hardware description and hardware verification language which is an advanced version of Verilog. It is used for modeling circuits, designing the circuits, simulating them, performing tests on these circuits to know whether the circuit will be given the required outputs, and finally, these circuits can be implemented on hardware. We used the system Verilog inside of Verilog because in the 1990s Verilog was the only language that could be used for the description of any circuit, but as the complexity of circuits increases the functionality that was present in Verilog was not enough. So they developed an advanced version of Verilog i.e. system Verilog. System Verilog consisted of all the functionality of Verilog and added to them were some features like OOP which helps in the construction of testbench, functional coverage, assertions, and many other things. In this project, we use the system Verilog for verification of the design under test (DUT), i.e. 16-bit ALU. Verification is a process to ensure whether the DUT is working as expected or not. As chip designing is a very long and expensive process, directly manufacturing the chip and finding out the problem is a very tedious and expensive process so we first design and then verify the design with the help of system Verilog. It helps in cost and time-saving.

II. Survey

In our study we firstly focused on what is System Verilog. It is an HVL i.e., Hardware Verification Language that is used for designing and verification of the required IP/Core. It is a standardized IEEE1800 that supports features for verification like OOPs, Assertions, Coverage, etc. which makes this language suitable for both designing and verification. It is compatible with a wide range of tools. Then we came to the ways in which we can verify different RTL codes i.e., the verification methodologies. There are many methodologies but the widely used ones are testbench file, functional simulation, UVM, FPGA Prototyping, etc. we then went on selecting Functional verification specifically Functional Simulation methodologies as the RTL code we would be testing is of a 16-bit ALU which is less complicated and Functional verification is generally used and gives best results for smaller RTL designs. Finally, we did some research using research papers where we studied the RTL designs and which methodology was used to verify these designs. In addition, we learned the need for a standard test bench[1], studied a case study on IEEE-754[1], the role of verification in the design cycle[2], verification and debugging of LC-3 Microcontroller[4], inter-integrated service protocol[6], coverage metrics[6], etc.

III. ALU

In this paper we have discussed how the verification of a 16- bit ALU can be done using the system Verilog. But before that, we need to know what a 16- bit ALU is. ALU is an abbreviation of arithmetic and logical unit i.e., it performs the task related to arithmetic and logical operations in a SOC. 16-bit means that there are inputs where each input is of 16-bits which can be given to the arithmetic and logical unit which will perform arithmetic and logical operations on this 16-bit data.

In our verification environment, an ALU is a DUT i.e., design under test. All the randomly generated inputs will be given to the design under test and will be verified whether the output is expected or not. An ALU performs arithmetic and bitwise operation on integer binary numbers. The symbol given in Fig.1 is the symbol for an ALU.

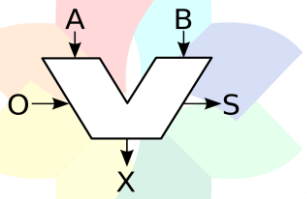

Fig1: Pinout diagram of an ALU

An ALU generally has two inputs A and B also called operands. Then X represents the output given by the ALU, O represents the select line i.e. O will tell us which operation is to be performed, and S is the output status. There can be many more inputs and outputs if we require which can make the ALU design more complex. In this paper, we are limiting the ALU design to two inputs, one output, and one select line. The operations that we will be performing here will be ADD i.e. addition without carry, SUB i.e. Subtraction, MUL i.e Multiplication, Division, and some bitwise operations like AND, OR, NOT, XOR.

IV. Methodology

There are many ways to do the verification of hardware but in this specific project, we are doing functional verification of our hardware i.e. a 16-bit ALU. In Functional Verification also there are different ways of doing the verification like Static verification, Functional simulation, FPGA Prototyping, Emulation, and UVM. So for our project, we went with the functional Simulation technique under Functional Verification.

i. Functional Simulation

Functional simulation is a technique of verifying the functional behavior of the desired Hardware by simulation in software. The functional simulation doesn't consider time delays like internal logic or interconnects, and Function Simulation is not helpful when it comes to the verification of software.

The main purpose of using this methodology is because we can simulate and verify the individual IP's or individual blocks of and particular IC. System-level verification is not possible with functional simulation. There are many advantages of function simulation methodology like it is very quick to set up, there is high visibility of the design, bugs can be identified very early on in the designing process, all the cases of the design are verified, it is not expensive. In the given Fig.2, we have the simulation setup which has blocks named test bench, measure coverage, generate simulation, design, and check results. The basic flow of work as shown in fig. 3. We create a test bench ii. Then instantiation of design in the test bench is done iii. Then stimuli i.e., the inputs are provided iv. The compilation of the test bench with design in the simulator takes place v. then we run the simulator vi. The outputs are observed vii. Verification of the logic of the design takes place i.e., whether the required output is coming or not.

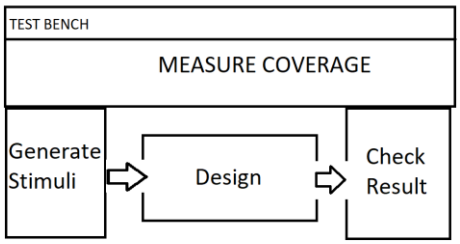

Fig 2: Simulation setup

V. Verification Methodology

1. Testbench components

The major elements of the verification environment that we created with there working in this project are-

1.1.1 Tx The inputs and outputs that are required for the verification environment are declared here in this block. Some examples of the input and outputs required are two input values, the select line, the output after the operation, etc.

1.1.2 Generator For testing any environment we need to generate some input values and give them to the design under test. The values generated for the input are generated randomly by the generator.

1.1.3 Driver The randomly generated inputs that are given out by the generator are in packet level information and the values to be processed further on should be in pin level information. Hence, the driver converts the packet level information to pin level of information i.e. it maps Tx class data to the Interface class.

1.1.4 Monitor It does the exact opposite of driver i.e. it converts pin-level information to packet-level information and it maps interface to the scoreboard.

1.1.5 Scoreboard When the output comes from the DUT we need to check whether the output is as expected or not. So the scoreboard compares the output given by the DUT and compares it with the expected output and then it issues an information message. It is also useful in coverage calculation.

1.1.6 Coverage The main function of coverage is to check how many possible cases are being executed and calculate the functional coverage of our design which will consist of cover bins and cover points of the same.

1.1.7 Mail Box These are the means of transport for the data from one block to other. These can be unidirectional as well as Bi directional.

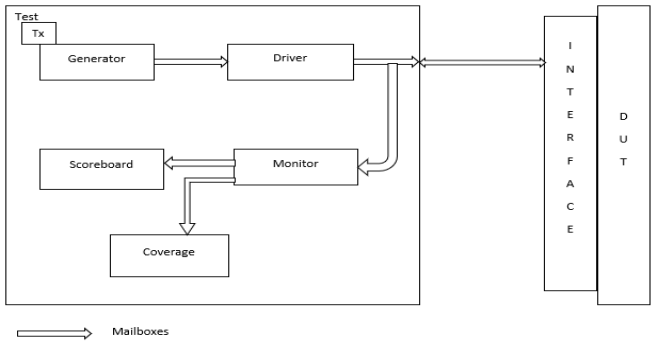

Fig.3: Block Diagram

VI. Results and Discussions

The design code written in verilog was converted in the simulated hardware using Xilinx ISE design suite. This design was verified under Questasim advanced Simulator.

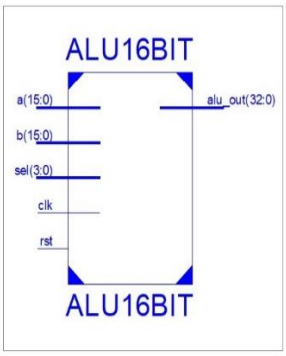

Fig 4: RTL pinout diagram

Fig 4 shows the RTL pin out diagran which was made on Xilinx ISE design suit. It has two inputs of 16 bits, one select line of 4 bits, clock, reset, and one output of 33 bits.

S.No	Bits	Operations
1	0000	Addition
2	0001	Subtraction
3	0010	Multiplication
4	0011	Division
5	0100	AND
6	0101	OR
7	0110	NAND
8	0111	NOR
9	1000	Logical shift left
10	1001	Logical shift right
11	1010	Rotate left
12	1011	Rotate right
13	1100	Logical xor
14	1101	Logical xnor
15	1110	Greater comparison
16	1111	Equal comparison

Table 1: Operations of ALU

Table 1: shows the bits used for the operation in ALU, a total of 16 operations are there.

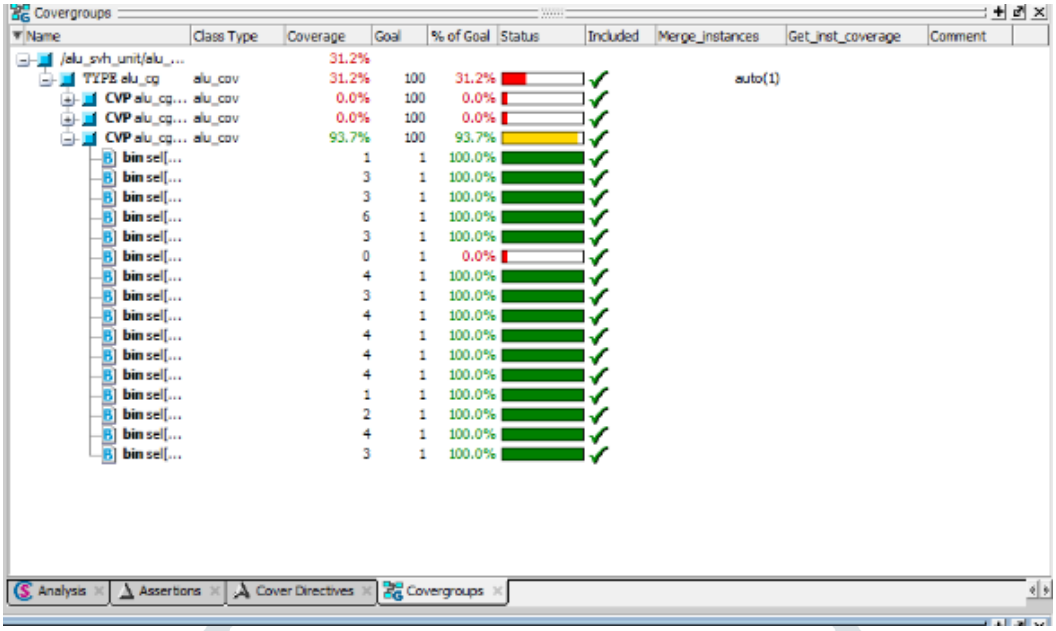

Fig6: Transcript with 31.2% functional coverage

Fig 6 shows the coverage report after running the codes for the sixth time. Attained 31.2%.

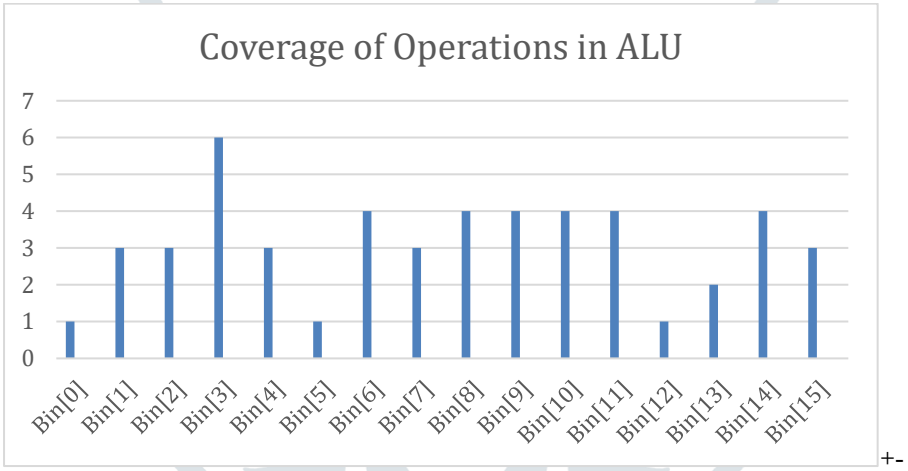

Fig 7 : Bar graph representing the coverage of each operation

Fig 7 is a bar graph representing the coverage of each operation i.e. on how many random value is being tested by a particular operation.

Conclusion

The environment will be very useful in reducing the time for the hardware to hit the market and provide efficient hardware with maximum coverage to test cases. This environment is fast and reliable in terms of testing and functional coverage for the design. The design has a few drawbacks which can be overcome using some advanced techniques and using industry grade tools, like UVM (Universal Verification Methodology) and tools like Synopsys VCS. The project led us to attain maximum functional coverage which in our case is 33.3%. All the operation of the design which it was intended to do were covered by us in this project. A 100% code coverage was attained in the project leaving no part of the code left out to be checked or verified by the EDA tool.

References

[01] Martin Keaveney, Anthony Mcmahon, Niall O'Keeffe, Kevin Keane, James O'Reilly " The development of advanced verification environments using System Verilog" ISSC 2008, Galway, June 18-19.

[02] Rakhi Nangia, Niraj Kumar Shukla, "Functional verification of I2C core using system Verilog"International general of engineering science and technologyJune 2014.

[03] Varun Kumar J, Anusha R, Shrinidhi S, Vishwas S, " Advanced verification of Single precision floating point ALU " 978-1-7281-0418-8/19/ IEEE 2019.

[04] Joyjit Chatterjee, Ayush Saxena, Anu Mehra, Garima Vyas, Vajja Mukesh, "Verification and Debugging of LC-3 Test Bench Environment using SystemVerilog" Proceedings of the 2nd international conference on Electronics, Communication and Aerospace Technology (ICECA 2018) IEEE Xplore ISBN:978-1-5386-0965-1

[05] Gaurav Sharma, Lava Bhargava"MDAB: Module Design Automation Block for Verification using SystemVerilog Testbench" 3rd International Conference and Workshops on Recent Advances and Innovations in Engineering, 22-25 November 2018

[06] Rakhi Nangia, Niraj Kumar Shukla, "Functional Verification Environment for I2C Master Controller using System Verilog" 2017 4th International Conference on Signal Processing, Communications and Networking (ICSCN -2017), March 16 – 18, 2017, Chennai, INDIA

[07] Ondrej Cekan, Richard Panek, Zdenek Kotasek "Input and Output Generation for the Verification of ALU: a Use Case" 978-1-5386-5710-2/18 IEEE 2018

[08] Roopa R. Kulkarni, S. Y.Kulkarni, "Energy Efficient Implementation, Power Aware Simulation and Verification of 16-bit ALU using Unified Power Format Standards"2014 International Conference on Advances in Electronics, Computers and Communications (ICAECC)

[09] Dharmavaram Asha Devi, Sai Sugun L, "Design, implementation and Verification of 32 bit ALU with VIO" Proceedings of the Second International Conference on Inventive Systems and Control (ICISC 2018) ISBN:978-1-5386-0807-4; ISBN:978-1-5386-0806-7

[10] Ankita Yadav,Varsha Bendre, "Design and Verification of 16 bit RISC Processor using Vedic Mathematics" 2021 International Conference on Emerging Smart Computing and Informatics (ESCI) AISSMS Institute of Information Technology, Pune, India. Mar 5-7, 2021

[11] [11] Edit code - EDA Playground

[12] [12] What are the ABCs of functional verification techniques? (analogictips.com)