International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 04 Issue: 05 | June -2017 www.irjet.net p-ISSN: 2395-0072

# Assertion based verification strategy for a generic first in first out (FIFO)

**Naveen T V[1], Dr. M V Latte[2], Mr. Sathish Shet[3], Mr. Shashidhara H R[4]**

[1] *Mtech in VLSI Design and Embedded Systems, Dept of ECE, JSSATE Bengaluru*
[2]*The Principal, JSSATE Bengaluru*
[3]*Asst Professor, Dept of ECE, JSSATE Bengaluru*
[4]*Asst Professor, Dept of ECE, JSSATE Bengaluru*

---

***Abstract:*** A generic FIFO is designed using a system Verilog coding technique. The pointers will indicate the status of the FIFO, the flag information's like full, empty, almost full, almost empty will be indicated and the FIFO will have a synchronous RESET capability. The functional verification will be done using assertion technique. The verification plan affords a definition of the test bench, verification properties, test surroundings, coverage sequences, application of test cases, and verification procedures for the FIFO design. The desires of this plan isn't always only to offer an define on how the aspect will be examined, but additionally to offer a straw man file that can be scrutinized by other design and system engineers to refine the verification technique.

**Keywords:** ***Generic FIFO, Synchronous, Assertion, Test bench, Straw man.***

## 1. INTRODUCTION.

The FIFO element shall represent a layout written in System Verilog with System Verilog assertions. The FIFO shall be synchronous with a unmarried clock that governs each reads and writes. The FIFO usually interfaces to a controller for the synchronous pushing and popping of facts. The FIFO shall encompass the following capabilities consisting of Parameterized garage area for data buffers, Parameterized facts widths for the information, Flag statistics for complete, EMPTY, nearly complete at the ¾ stage, nearly EMPTY on the ¼ level, A synchronous RESET capability. System Verilog with assertions in conjunction with simulation might be used because the verification language because it's miles an open language that provides exact constructs and verification capabilities. This plan consists of characteristic extraction and test approach, take a look at software approach for the FIFO, check verification method. System Verilog will be used for this design because it is a fashionable language, and is portable throughout gear. A reusable design fashion may be implemented.

## 2. ARCHITECTURAL VIEW OF FIFO AND ITS INTERFACES.

The FIFO consists of a clock input, reset, data in, push and pop inputs whereas the output portion consists of the status lines to show whether the FIFO is almost full, full, almost empty, empty and a provision for data out along with the error indicator. The figure below represents one such architecture.

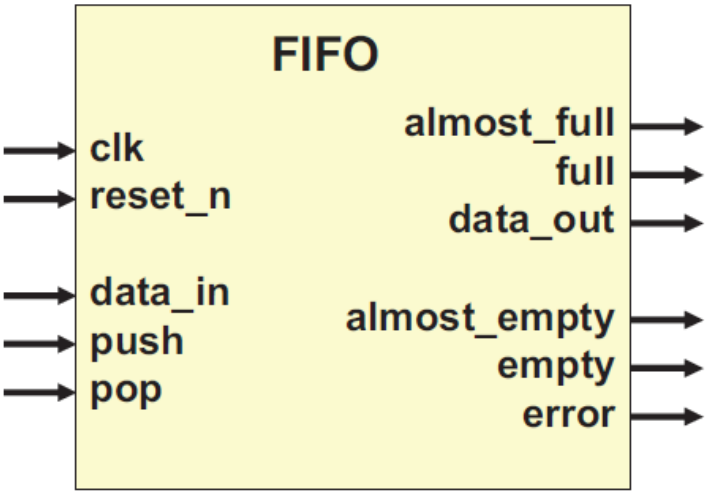


Figure 1: Interfaces of the FIFO

Now the high level architecture of the FIFO interfaces has to be built to show the controller and its input outputs.The FIFO should have the below mentioned characters:

- Parameterized space for data buffers.
- Parameterized data size for the information.
- Status flags for FULL, EMPTY, ALMOST FULL at the ¾ level, ALMOST EMPTY at the ¼ level.
- A synchronous RESET provision.

The architecture of one such system is shown in the figure below.

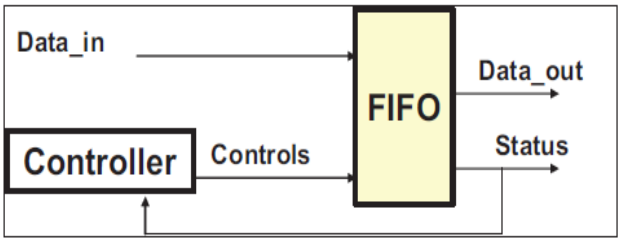


Figure 2: High level architecture of interfaces in FIFO.

## 3.SYSTEM APPLICATIONS.

The FIFO can be carried out in a ramification of gadget configurations. Determine underneath demonstrates one such configuration wherein the FIFO interfaces on one facet to a bus controller, and on the other facet to a distinctive controller. All buses use the same system clock. it is the responsibility of the enqueue/dequeue controller to manipulate the integrity of quantity of data transferred into FIFO and extracted from it. The hardwired

Architectural view of FIFO application is shown in the figure.

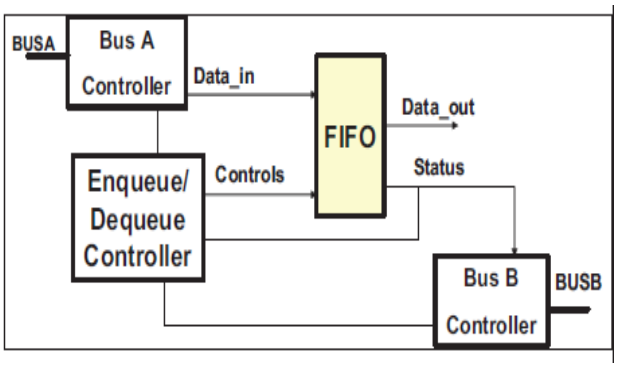


Figure 3: Hardwired FIFO Application Architecture.

The specific port factors within the interface in the figure 3 are defined in this section with necessities on them captured as assertions. since number of the ports describe records in depth part of the system (consisting of the records being popped from the FIFO), some of the System Verilog test bench capabilities together with queues and duties are used to capture their necessities. Because these tasks and queues are supposed completely for the cause of specification and verification, and do now not have an instantaneous correlation to the hardware implementation of the FIFO, they're declared within the interface itself. The timing diagram is shown below

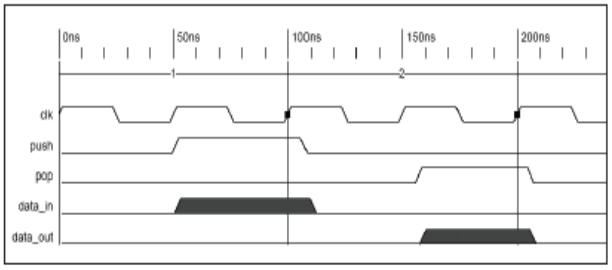


Figure 4: Timing waveforms for Interface of FIFO

## 4. FEATURE EXTRACTION AND TESTING.

The layout features are absorbed from the requirements characters. For every characteristic of the layout, a test methodology is identified. The methodology comprises of directed and pseudo-random checking. A verification criteria for every of the design characteristic is documented .This selection definition, test method, check series, and verification standards forms the premise of the purposeful verification plan. Table underneath summarizes the function extraction and verification standards for the functional necessities. For nook checking out, pseudo-random push and pa transactions might be simulated to mimic a FIFO in a gadget surroundings. The surroundings will carry out the following transactions at pseudo-random intervals:

- Generate push commands.
- Generate pop commands.
- Force resets

| Tst | FEATURE AND DIRECTED TEST STRATEGY | Priority | TEST SEQUENCE/STARTEGY | VERIFICATION CRITERIA Interface verification SVA Properties(fifo_props_sv)-the following |
|---|---|---|---|---|
| 1 | Fixed Parameterization<br>-Bit_depth (fifo buffer size)<br>-data width | 1 | Configuration setup<br>Buffer depth=2**4,2**8<br>Width=16,32<br>Pseudo-random push and pop transaction<br>Unique data patterns using random values | Design compiles & elaborates correctly. Simulation + monitoring of properties & coverage.<br>Verification of output sequence(i.e, first in first out) |
| 2 | RESET<br>Reset applied when full state of FIFO is at different levels | 2 | Property p_t1_full; @(posedge_clk)<br>full=>!reset_n;<br>endproperty;p_t1_full<br><br>Property p_t2_afull; @(posedge_clk)<br>Almost_full=>!reset_n;<br>endproperty;p_t2_afull<br><br>Property p_t3_empty; @(posedge_clk)<br>empty=>!reset_n;<br>endproperty;p_t3_empty<br><br>Property p_t4_a empty; @(posedge_clk)<br>almost_empty=>!reset_n;<br>Endproperty;p_t4_a empty | Simulation +monitoring of properties and coverage |
| 3 | COVERAGE<br>Using the properties and sequence defined in 4:1 | 3 | Cover property<br>(p_push_pop_sequensing);<br>cover property (qFull);<br>cover property (qEmpty);<br>cover property (qAlmost_empty);<br>cover property (qAlmost_full);<br>cover property (qOffFull);<br>cover property (qOffEmpty);<br>cover property (qOFFAlmost_empty);<br>cover property (qOFFAlmost_full); | Sequences must all have a coverage count of at least one. |

## 5. TEST BENCH ARCHITECTURE.

The numerous architectural factors need to be taken into consideration inside the definition of the test bench environment, together with the following:

- Reusability/portability/Verification language.
- Range of BFMs to emulate the separate bus.
- Synchronization strategies between BFMs.
- Transaction directives and sequencing techniques.
- Transaction driving strategies.
- Verification methods for designing its sub blocks.

Figure below shows the test bench structure. The test bench uses the FIFO interface definition, FIFO bundle, and the FIFO belonging module. The test bench consists a transactor module to create transactions along with reset, push, pop, and idle cycles. A couple of server obligations offers the low level protocols to complete the transactions. The test bench architecture is given below.

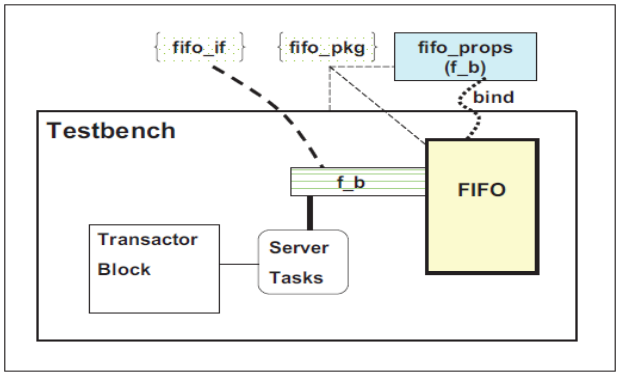


Figure 5: The Testbench Overview of a System

System Verilog can be used for this design because it's far a popular language and a reusable design fashion may be implemented. A System Verilog package deal captures the common parameters for this design and is proven in figure. A property module file is described in figure for binding to the FIFO from within the Test bench. An define of the FIFO test bench that demonstrates the module instantiations and binding is shown in figure 5.

## 6. RESULTS AND DISCUSSIONS.

Amid the recreation, the declaration continues following the exercises which are in the outline and gives the data showing how profoundly the test condition incorporates the plan's usefulness. By getting and sparing the FSM's properties utilizing attestations, we can identify them effectively, handle them completely, and gather the cross item covering information amid the reproduction. Notwithstanding that, the execution of the Finite State Machine likewise directly affects the adequacy of the confirmation.

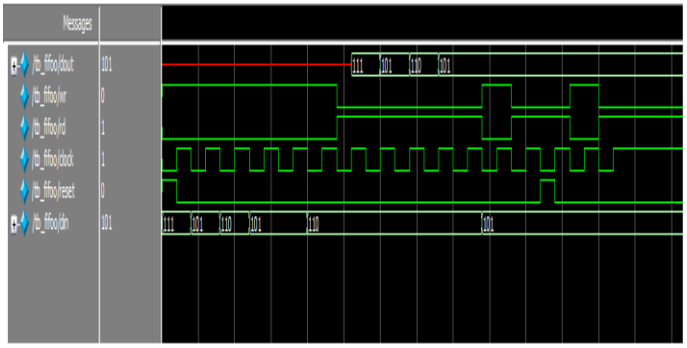

Figure 6: Simulation waveform of a generic FIFO

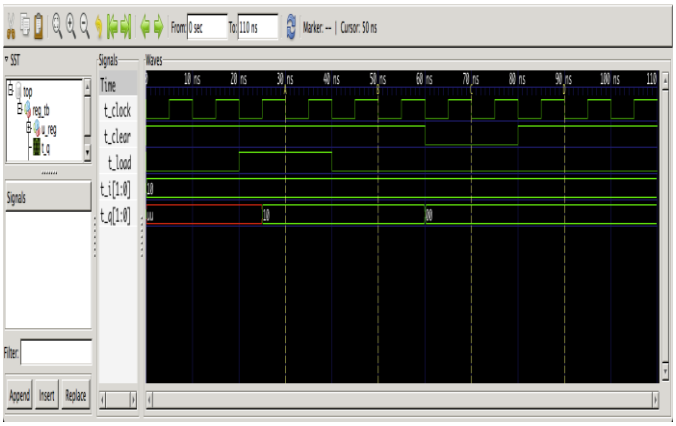

Figure 7: Assertion verification report

## 7. CONCLUSION.

The project taught us the behavior of FIFO during the information read and write operations and By the use of assertion technique, several unique behaviors of the FIFO system were clarified and the system verilog coding knowledge got enumerated through this research. A synchronous clock characters are also clarified.

## 8. ACKNOWLEDGEMENT

We authors would like to thank the JSS Academy Of Technical Education, Bengaluru for providing us an opportunity to carry our project and also for guiding us throughout the process in completion of this project.

## 9. REFERENCES.

[1] Sayantan Das, RiziMohanty, PallabDasgupta, P.P. Chakrabarti “Synthesis of System Verilog Assertions”, Design, Automation and Test in Europe, München, Germany, 2006

[2] Ivan Kastelan, ZoranKrajacevic, “Synthesizable System Verilog Assertions as a Methodology for SoC Verification”,

First IEEE Eastern European Conference on the Engineering of Computer Based Systems, 2009.

[3] T. Yunfeng, "An introduction to assertion-based verification", IEEE 8th International Conference on ASIC, Hunan, China, 20–23 October 2009, pp. 1318–1323.

[4] Friedman E G. Clock distribution networks in synchronous digital integrated circuits. Proceedings of the IEEE, 2001, **89**(5): 665-692.

[5] Martin A J, Nystrom M. Asynchronous techniques for system- on-chip design. Proceedings of the IEEE, 2006, **94**(6): 1089-1120.