First In First Out (FIFO)

VERSION 1.0

Revision History

Revision	Description	Date	Modified By
1.0	Initial Document	30/05/2023	Neel Pambhar
1.1	--	--	--

Table of Contents

Chapter 1: FIFO Overview

FIFO is a type of computer memory that temporarily stores data and provides at output in same sequence as it is data in sequence and instructions for the CPU. It is fast, volatile, and upgradeable, and plays a critical role in system performance.

Why we use FIFO?

- **Purpose**: FIFO is used to store data and provide the data in same sequence as it is inputted.
- **Capacity**: FIFO comes in various sizes. The size of FIFO determines how much data can be stored and accessed by the CPU at any given time. More size of FIFO allows for more data to be stored.
- **Speed**: FIFO operates at a much faster speed .

FIFO Block Diagram

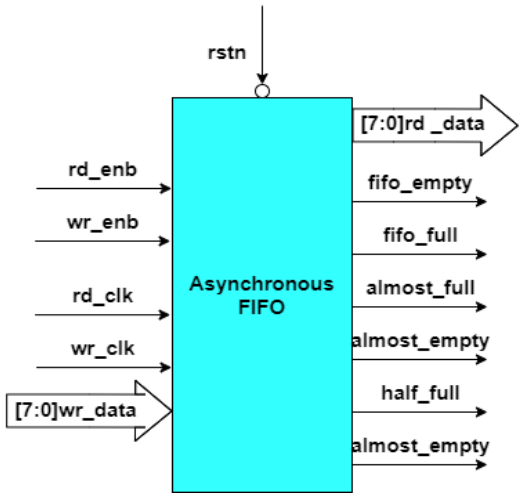

Figure 1.1: FIFO Block Diagram

Chapter 2: FIFO features

The key features of the Asynchronous FIFO are:

- Empty flag
- Full flag
- Half full flag
- almost_full flag
- Almost_empty flag
- Back to back operation
- Simultaneous read and write
- Reset

Chapter 3: FIFO Verification Plan

Verification involves studying the relevant specifications, extracting features from it that are to be tested, devising a strategy as to how these features are to be tested, developing a verification environment based on the strategy, writing testcases to cover all the scenarios and achieving 100% functional coverage figures.

3.1 Feature Extraction

In this phase we read first specification and indentify the specs. based on that plans ahead the above spreadsheet.

FIFO_feature_List.xlsx

Lists out features to be tested in the corresponding specification, assigns a feature id to them and tells how the feature is to be tested (testcase name or checker task name).

3.2 Coverage Plan

A functional coverage plan needs to be made based on the feature extraction document. This plan lists out the various combinations of stimuli that need to be generated for the proper verification. This has been included in the feature extraction spreadsheet itself.

FIFO_feature_List.xlsx

3.3 Checker Plan

A checker plan needs to be made based on the feature extraction document. Implementations for the features marked as "checker" are elaborated here. We have included it in the feature extraction spreadsheet itself.

3.4 Verification Environment Development

Our environment is based on SystemVerilog.

3.5 Test suite development

3.5.1 Directed Testcases

Testcases written to test specific areas or to generate a specific kind or sequence of transactions is known as a directed testcase. These testcases are helpful in the initial and final stages of verification In the final stages, they are used to hit specific functional or code coverage areas.

3.5.2 Random Testcases

Random scenarios are generated based on constraints provided in the testcase. These testcases are run several times with different seed numbers to generate different scenarios to achieve more functional coverage figure.

Chapter 4: FIFO Verification Environment Development

4.1 FIFO Block Diagram

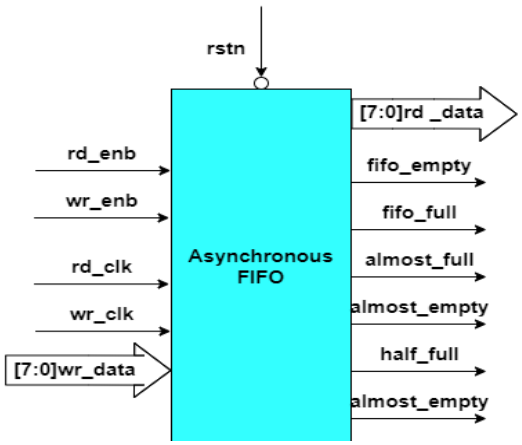

Figure 4.1: FIFO Block Diagram

4.2 Verification Architecture

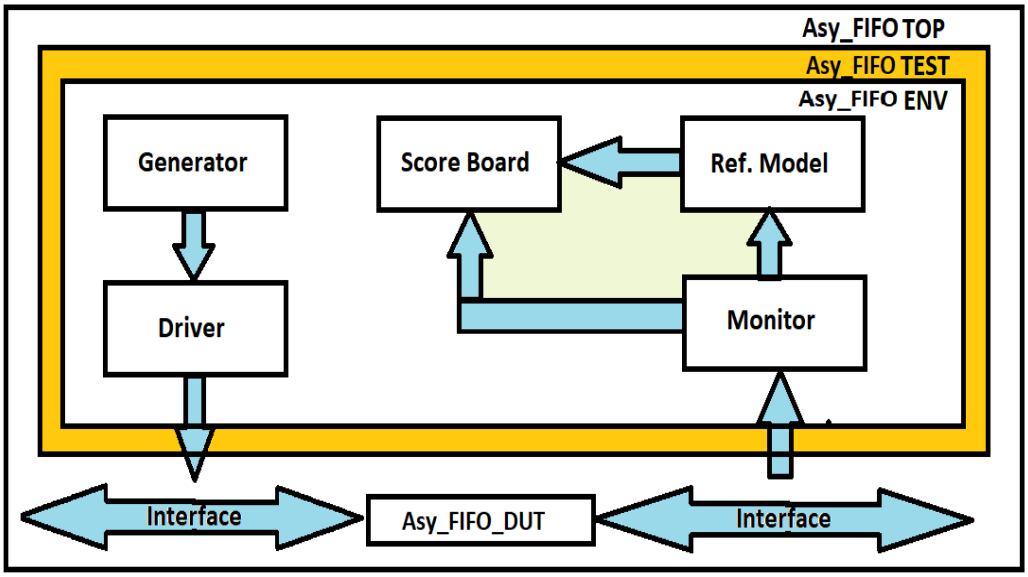

Figure 4.2: FIFO Verification Architecture in System Verilog

4.3 FIFO Component

4.3.1 Transaction Class

This component is responsible for creating different meaningful data for the DUT. These data are basically the inputs given to the DUT.

It' s contains data members that represent the various signals or values that are part of the transaction.

Data members: -

- type trans_kind_e : trans_kind_e is a instance of 2-bit wide enum that represents the mode of operation(randomize)
- A random enumeration variable of type trans_kind_e that represents the signal type (IDEAL, READ, WRITE, SIM_RW).
- type wr_data is randomize data that will be written into the FIFO. rd_data is data read from the FIFO (not randomized).
- empty, full, almost_full, almost_empty and half_full are 1-bit variable that represents flags of FIFO. Which is use for sampling data
- Also, there is one display method ,which is specially for scoreboard

In transaction class of FIFO, enum is define for selecting the mode of operation. In this 00 mode is define as IDLE state, 01 mode is defined as WRITE state, 10 mode is defined as READ state and 11 mode is defined as SIMULTANIOUS READ WRITE state. In the same write data is defined as random data. So, in the simulation write data will be randomized. For operation in fifo full and empty flags are also included.

4.3.2 Generator Class

Generator class is responsible for stimulus/traffic (transaction items) generation and keep the same in mailbox which is further processed by driver.

1. A mailbox (gen_drv) of transaction class type is used to take data from generator and provide it to driver.
2. This class(generator) is declared as virtual so that the testcase writer can extend it but can not create object to modify the data fields.

3. A run method is declared as pure virtual to tell the testcase writer to compulsory overwrite.
4. In run method testcase writer will randomize the transaction class as per the testcase.
5. put_gen_drv() task is used to put the generated data in mailbox.
6. put_gen_drv() task is declared as protected but still child can access this method to put the data in mailbox.
7. new constructor with argument is used to connect the sub mailbox to main mailbox;
8. In generator class, gen_drv is mailbox for communicate with the environment. In this, virtual task run is taken for running the different test cases. In this one protected task is created name as trans_h. this protected task will give permission to the child only for the extend of anything

4.3.3 Driver Class

- The Driver is responsible to takes transaction or sequence level activity(stimulus) coming from generator and convert it into the pin or system level activity.

1. Driver is connected to interface to provide the data to DUT so the virtual instance of interface of interface is taken(vif) from driver modport.
2. Write and read task is used to drive the write and read data as per the write and read clock.
3. As per the enum 'mode' the mode is selected and asper the mode the respective write, read and sim_rw task is called
4. After drving data enable signal will low.

4.3.4 Monitor Class

- The monitor is responsible to take pin or system level activity coming from bus (interface) and convert it into the transaction level activity.

1. In monitor class has two mailboxes. One mailbox is for sending the data from monitor to reference model (here mailbox name is mon_ref) and one mailbox is for sending the data from monitor to score-board (here mailbox name is mon_sb). One virtual interface is taken for sampling the output data of FIFO the data from interface to monitor.
2. The `forever` loop is indicating that monitor will continuously ready to sample anytime when data is available.
3. Write task is called at the write clock edge and data(write or read data) is sampled from interface and transferred to the reference model and scoreboard same goes for the read task the task is called at read clock edge.

4.3.5 Reference Model

- Reference model is a component where we wrote a logic to generate expected output (checker logics). (Predicting to output).

1. Two mailbox are taken one is points to mon_rf and another one points to rf_sb .

2. Get data form mon_rf and according to enum type operation perform in referance model que.
3. According to size of que array try to high flags.
4. Now store data in trans_h and put in ref_sb mailbox.

4.3.6 Scoreboard

Scoreboard is responsible to check whether your design output is correct or not. It's collects expected value from reference model, actual value from monitor and compare those values and log the status.

1) Taken two mailbox handle which is point to mon_rf and rf_sb

2) Mon_rf is consider as act_trans and rf_sb is consider as exp_trans .

3) If data is not matched than $error will raised

4) Special display for score board call.

5) Cover group sample method call .

Chapter 5: Running Simulation

MAKE FILE:-

```
RTL = ..\RTL\Asy_fifo.v

PKG = ..\TEST\pkg.sv

TOP = ..\TOP\top.sv

ASSERTION = ..\Assertion\assertion_fifo.sv

INCR = +incdir+..\ENV +incdir+..\TEST

TOP_NAME = tb

COVERAGE = -c -do "coverage save -onexit -directive -cvg -codeall




vlog:

        vlog $(RTL) $(PKG) $(ASSERTION) $(TOP) $(INCR)


qverilog :

        vsim -novopt $(TOP) -c -do "run -all ; exit ; " +NORMAL_WRITE_READ


vsim: vlog

        vsim -novopt tb +NORMAL_WRITE_READ



write_read: vlog

        vsim -novopt $(TOP_NAME) +NORMAL_WRITE_READ
```

regression:

vlog -coveropt 3 +cover +acc $(RTL) $(PKG) $(TOP) +incdir+..\ENV +incdir+..\TEST

vsim -coverage -vopt $(TOP_NAME) $(COVERAGE) NORMAL_WRITE_READ.ucdb; run -all; exit" +NORMAL_WRITE_READ

vsim -coverage -vopt $(TOP_NAME) $(COVERAGE) RSTN_DRN_OPN_TC.ucdb; run -all; exit" +RSTN_DRN_OPN_TC

vsim -coverage -vopt $(TOP_NAME) $(COVERAGE) SIMULTANEOUS_TC.ucdb; run -all; exit" +SIMULTANEOUS_TC

vcover merge fifo.ucdb NORMAL_WRITE_READ.ucdb FULL_FLAG_TC.ucdb EMPTY_FLAG_TC.ucdb ALMOST_FLAG_TC.ucdb ALMOST_FULL_FLAG_TC.ucdb RSTN_DRN_OPN_TC.ucdb SIMULTANEOUS_TC.ucdb

vcover report -html -htmldir FIFO_MERGE_CVG fifo.ucdb

DO FILE :-

vlib work

make vlog

vsim -novopt tb

run 0ns

do wave.do

run -all

Chapter 6: Closure Report