Design Document -Single Clock FIFO

Tom Diederen

Revision History

|  |  |  |
| --- | --- | --- |
| **Date** | **Version** | **Comments** |
| Feb 2023 | 1.0 | Initial Release |
|  |  |  |
|  |  |  |
|  |  |  |

List of Figures

[Figure 1 RTL Code - Overview 3](#_Toc127879476)

[Figure 2 RTL Code - Detailed 4](#_Toc127879477)

[Figure 3 Test Bench Design 5](#_Toc127879478)

[Figure 4 Simulation - Start 6](#_Toc127879479)

[Figure 5 Simulation - End 6](#_Toc127879480)

# Summary

The goal of this mini project is to get familiar with the Quartus FPGA EDA tooling as well as the Questa simulator. A simple single clock FIFO from [1] is used as a reference design. Compared to the original design, I parameterized the FIFO length and consolidated some logic leading to fewer procedural blocks. I also added comments and updated port declarations to a newer Verilog style to enhance readability of the RTL.

# Specifications & Requirements

There aren’t any strict power, performance, or area requirements for this design. It’s simple, small, and will fit many FPGAs. Since standalone implementation of the module isn’t very useful, I choose not to implement it in an actual FPGA. The requirements are:

* The design must support a FIFO that reads, and writes based on a single clock.
* Input and output buffers can be written to or read from at different clock speeds.
* A concurrent read and write should not cause data corruption
* Flags should indicate whether the FIFO is empty or full
* FIFO will contain 1-byte elements although this could be parameterized in a future design

# Architecture

A block diagram of the module to be designed is shown below.

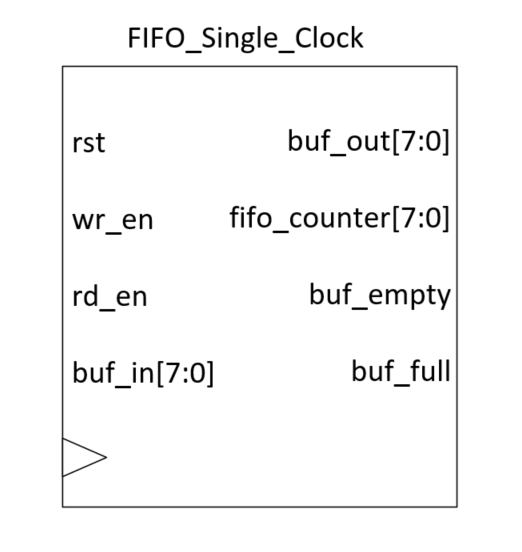


Figure Block diagram of the design

The design will feature the following inputs:

Table I/O Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Signal Name | Acronym | I/O | Width (bits) | Summary |
| Reset | rst | Input | 1 | Synchronous reset |
| Write enable | wr\_en | Input | 1 | Enables writing to FIFO. Active high. |
| Read enable | rd\_en | Input | 1 | Enables reading from FIFO. Active high. |
| Buffer in | buf\_in | Input | 1 | Input buffer for value to be written to FIFO |
| Buffer out | Buf\_out | Output | 1 | Output buffer for value to be written to FIFO |
| Buffer Empty | Buf\_empty | Output | 1 | Flag that indicates an empty buffer. Active high. |
| Buffer Full | Buf\_full | Output | 1 | Flag that indicates a full buffer. Active high. |
| FIFO Counter | Fifo\_counter | Output | 10 | Indicates how many elements are stored in FIFO. |

# RTL Design

The module has 3 internal registers:

1. A 4-bit read pointer
2. A 4-bit write pointer
3. The FIFO register of length FIFO\_DEPTH that holds 8 bit elements.

The module contains 4 procedural blocks:

1. Sets buf\_empty and buf\_full flags based on the fifo\_counter register.
2. Handles updates to the fifo\_counter
3. Handles writing to the FIFO
4. Handles reading from the FIFO

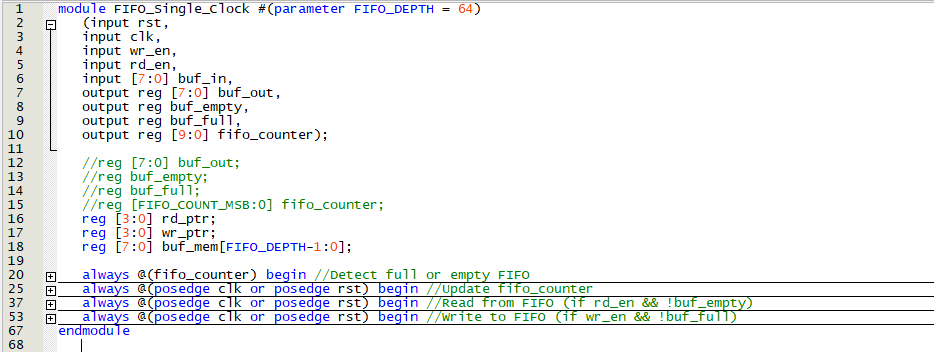
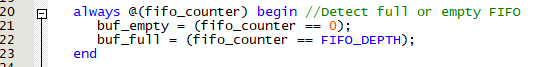
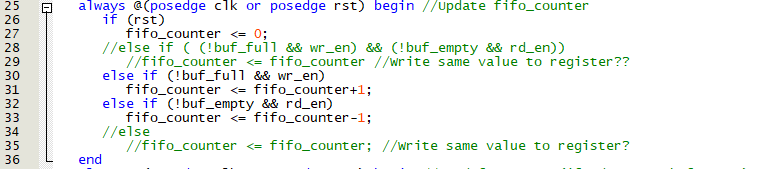
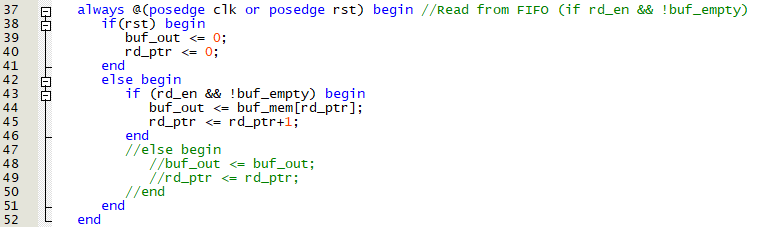


Figure RTL Code - Overview







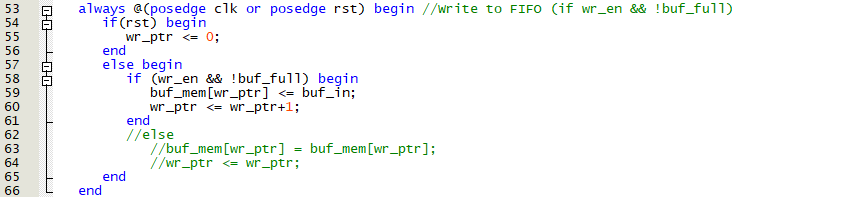


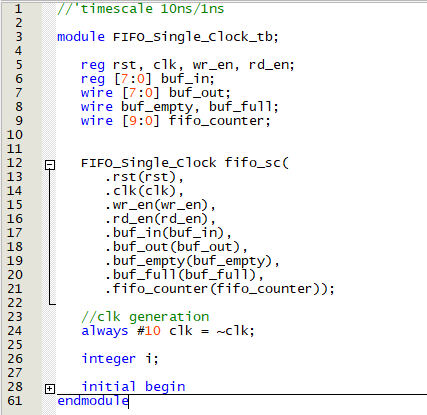
Figure RTL Code - Detailed

# Verification (Simulation)

## Testbench

I created a very simple test bench to manually check some key events. The test bench checks:

* Reset behavior
* A write followed by a read
* Multiple reads to fill up the FIFO to check buf\_full flag (buf\_empty is checked in the reset test case)



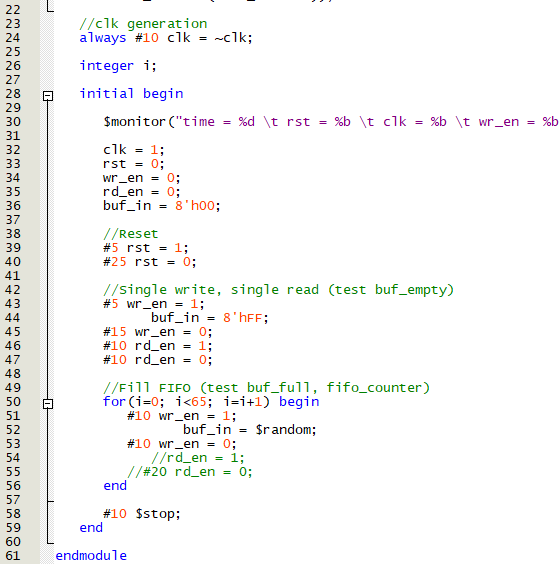


Figure Test Bench Design

## Simulation results



Figure Simulation - Start

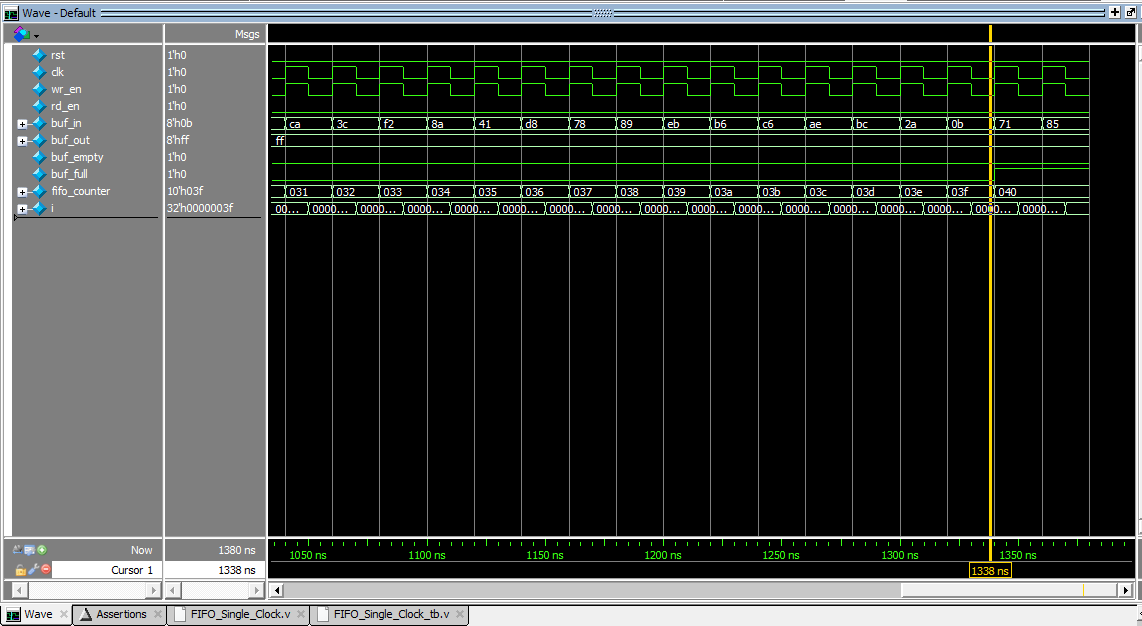


Figure Simulation - End

# Implementation

The design is intended to be used as a building block in a future, bigger, project. Implementing it as is on an FPGA is not very useful.

# Future improvements

The current design only has a few parameters and a limitation for the sake of simplicity. The fifo\_counter has a hard coded 10 bit value in this design. If a FIFO depth > 1024 were to be selected the counter would never reach FIFO\_DEPTH and buf\_full would never be set. A counter overflow could result in data being overwritten before being read. It would be better if fifo\_counter depended on FIFO\_DEPTH so this never occurs.

Also, the input and output buffer can only store a single byte each. If this FIFO were to be used in an actual design, bigger input and output buffers would probably be needed.

# References

|  |  |
| --- | --- |
| [1] | Shepherd Tutorials, "Section 4.105," February 2023. [Online]. Available: https://www.udemy.com/course/verilog-hdl-vlsi-hardware-design-comprehensive-masterclass/. |