Princess Sumaya University for Technology

King Abdullah II Faculty of Engineering

Electrical Engineering Department



|  |
| --- |
| **digital logic lab**  **final design project**  **transmitter and receiver function** |

|  |  |  |  |
| --- | --- | --- | --- |
| *Authors:* |  |  | *Supervisor:* |
| Nedal Abu haltam  Malik Al Satari | 20210125  20210021. | Computer Engineering  Network information security  Engineering | Eng. R.Harasis |

*January 11, 2024*

***Abstract***

*Transmitter and receiver technology is a fundamental block in our complicated, sensitive communication systems. It helps in sending data from one place to another using different techniques, technologies, and methods. It is an essential part of any modern system. There are many ways to implement it for different needs and requirement. On of the ways to build such a system is through a hardware description language (HDL). So, in this report we will discuss the process we went through to build this system using system Verilog. And we will test to make sure that our design is correct and meet the needs.*

Table of Contents

[1 Introduction 2](#_Toc500411309)

[1.1 Theory 2](#_Toc500411311)

[2 Design process 2](#_Toc500411313)

[3 Tests and Results 8](#_Toc500411314)

[4 Conclusions 12](#_Toc500411320)

# Introduction

Transmitter and receiver functionality is very important to implement in every communication system. And there are many ways to apply and implement such a system. In this report we choose to use Verilog language (i.e. a hardware description language). We will go through the steps we took to design and write the code for the system and test whether it works correctly or not.

## Theory

In theory begin the transmission is simple. We want a device (i.e. the transmitter) to read data from a source memory that it can access from, then it prepares a req signal when it wants to send the data to the other device (i.e. the receiver), and a parity bit that can be used to detect a corruption in the transmitted data. After that, the receiver receives the data and checks it using the parity bit and store it in the destination memory. And finally, the receiver sends back the acknowledge signal (ack) express that the signal is received and stored and the full signal that indicates if the destination memory is full or not.

# design process

in this section we will go through the process step by step mentioning the details of the design.

Of course, the transmitter and the receiver take the clk signal as an input to be synchronized with each other. And the error module to update the state.

1. Building the source memory to read the data from is essential and it has to take as a module input the source address that is from the transmitter and give an output the source data to the transmitter to be able to transmit it. For testing purposes, we shall initialize the memory with initial data.

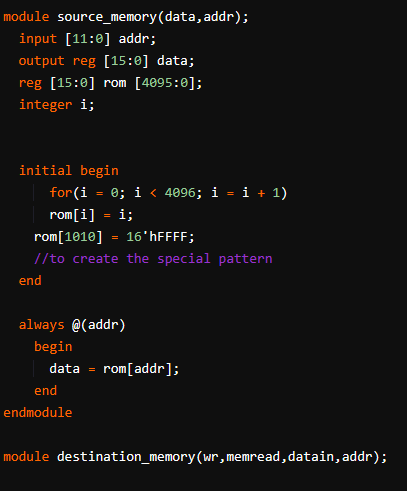


Figure 1: source memory module.

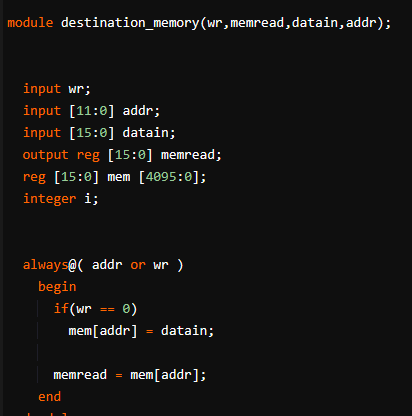
1. Building the destination memory is not complicated it has to take two inputs from the receiver the destination address and the source data the was transmitted we shall call it the destination data. The memory has to store the given data in the given address if and only if the receiver allowed, we used a write signal to indicate when to write on the memory. And for testing purposes we took an output from the destination memory to make sure that the data written is correct and on the right time.

Figure 2: destination memory module.

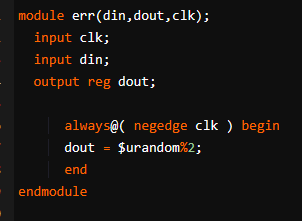
1. Implementing the err (i.e. the error module). The reason to build this module is to introduce an error in the data transmitted or to simulate real life scenarios like noise to the data. It is very simple it takes the MSB of the data and flip it randomly and then pass it to the receiver.

Figure 3: error module.

1. Building the transmitter. It takes as an input the ack and the full from the receiver and the source data from the source memory and the source register (i.e. the source address) to start copying from and outputs the data, the parity bit, and the req. The functionality of the transmitter is very critical because it generates the request signal (i.e. req) and the parity bit signal. So, any error in these will cause a false write on the destination memory, thus, data will be corrupted. So, we check the conditions carefully so we can safely request to transmit and write correctly. We added the prevaddr and prevdata se we can save the previous address and data to get them back when we need to resend the old data if it was invalid. The den (i.e. the data enable) is an input from us to decide when the system should start the transmission sequence and when to stop.

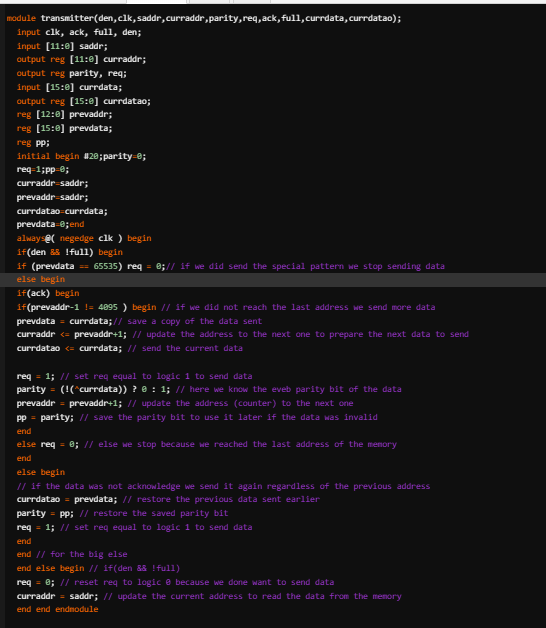


Figure 4: transmitter module.

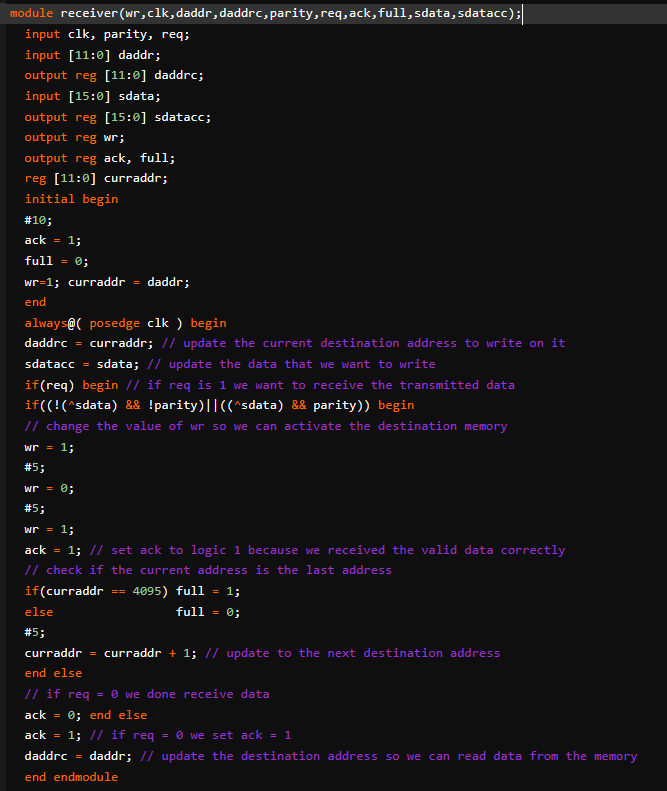
1. Building the receiver is not less important than the transmitter because they have to work together in harmony so no catastrophe would happen to the system. It takes as an input destination address to give it to the destination memory the as well as the source data and outputs the ack and the full signals. What it does is first checks if the req is valid from the transmitter and then the parity bit to see if there and errors in the data after that if there are no errors there is a write signal from the receiver to the destination memory so it can control when to write and when not to write finally it acknowledges by setting ack and see if the memory is full or not.

Figure 5: Receiver module.

1. Finally, the system module which gather all the blocks and connect the input and the outputs for every module to make it work. And using the system module we can test and see if our system works.

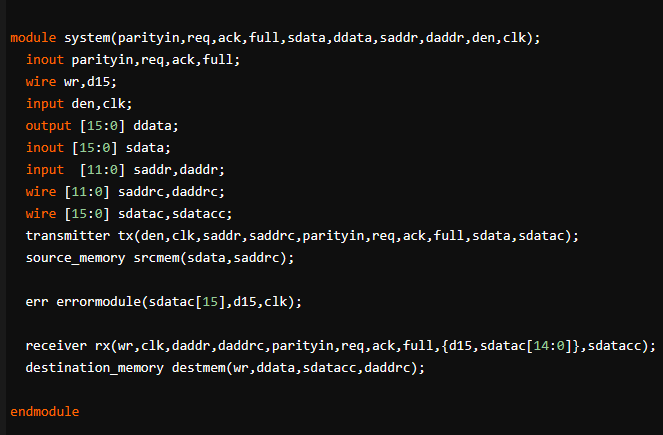


Figure 6: the System module.

# tests and Results

In this section we shall build the test bench module and try different inputs to meet the proposed requirements of the transmission protocol.

1. The first test bench we choose the normal transmission operation.

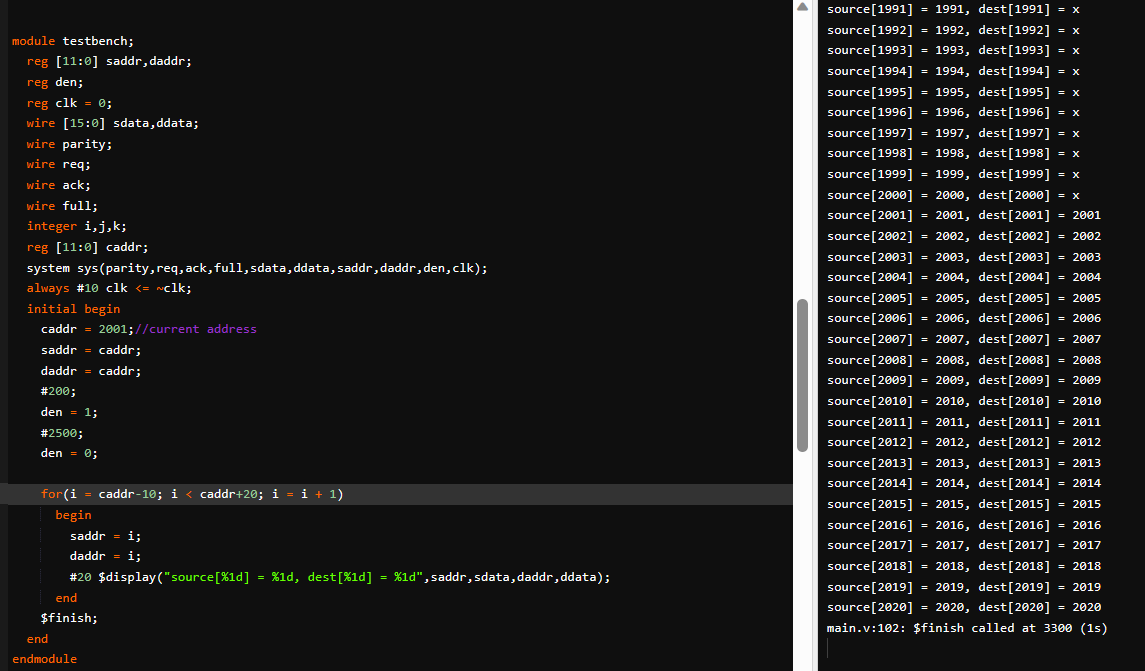


Figure 7: testbench.

1. The second test is to see if the transmitter will stop if it countered the data pattern 16’hFFFF

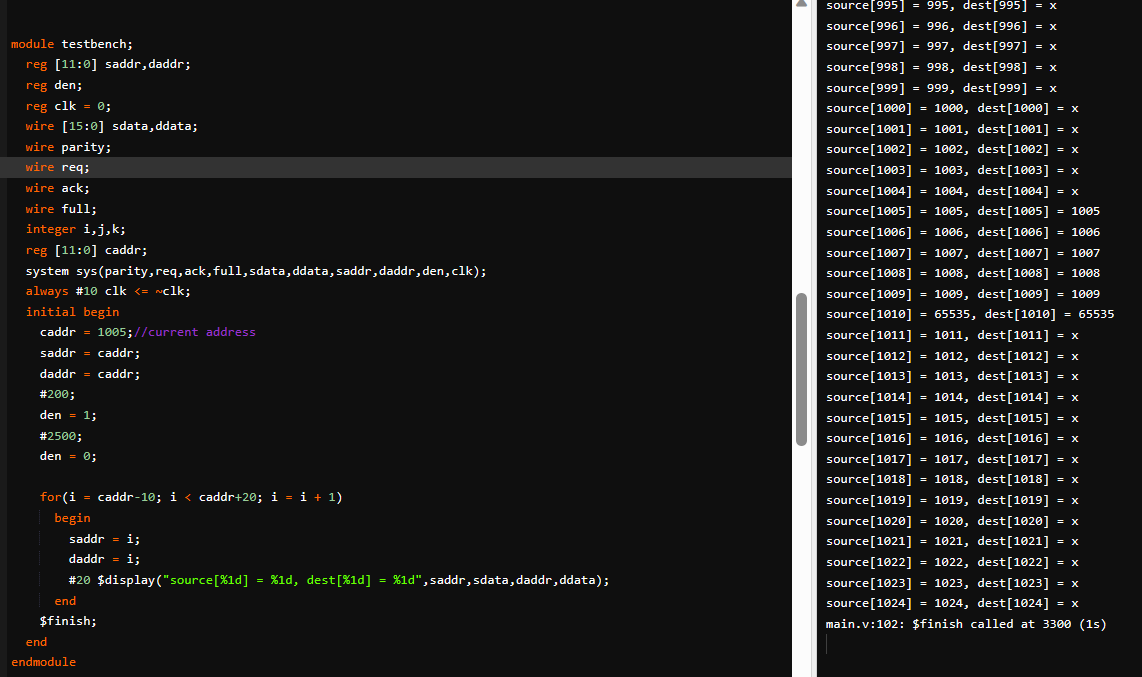


Figure 8

1. The third test is to see whether the full flag will indicate if we reached the last address of the destination memory.

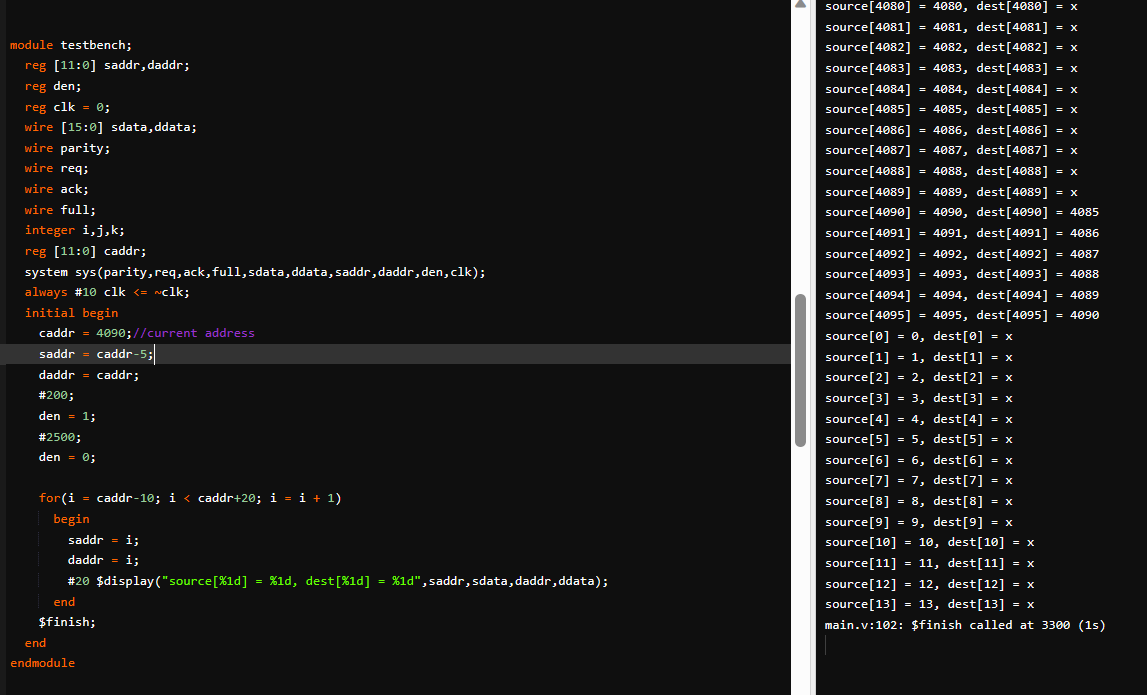


Figure 9

1. The fourth test is to see if the transmitter will stop if it also reaches the last address of the source memory, in our implementation we decided that it should copy the last address of the source memory then stop.

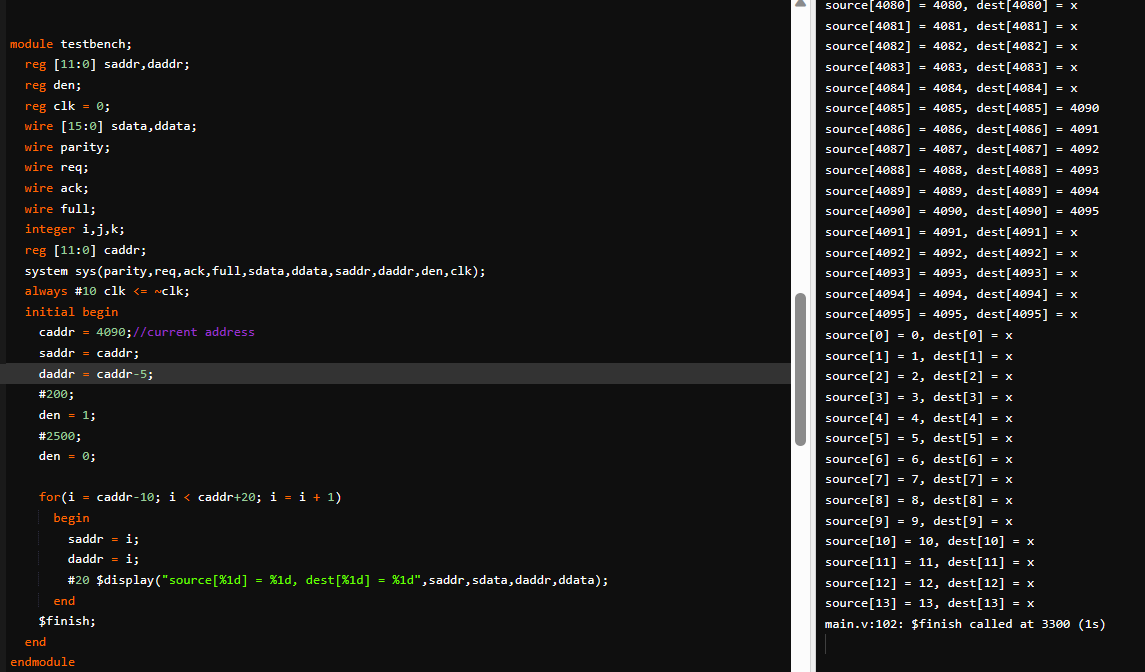


Figure 10

1. Note if you want to transmit from the address zero for example you will need to change the for loop attributes accordingly so you can see them correctly.

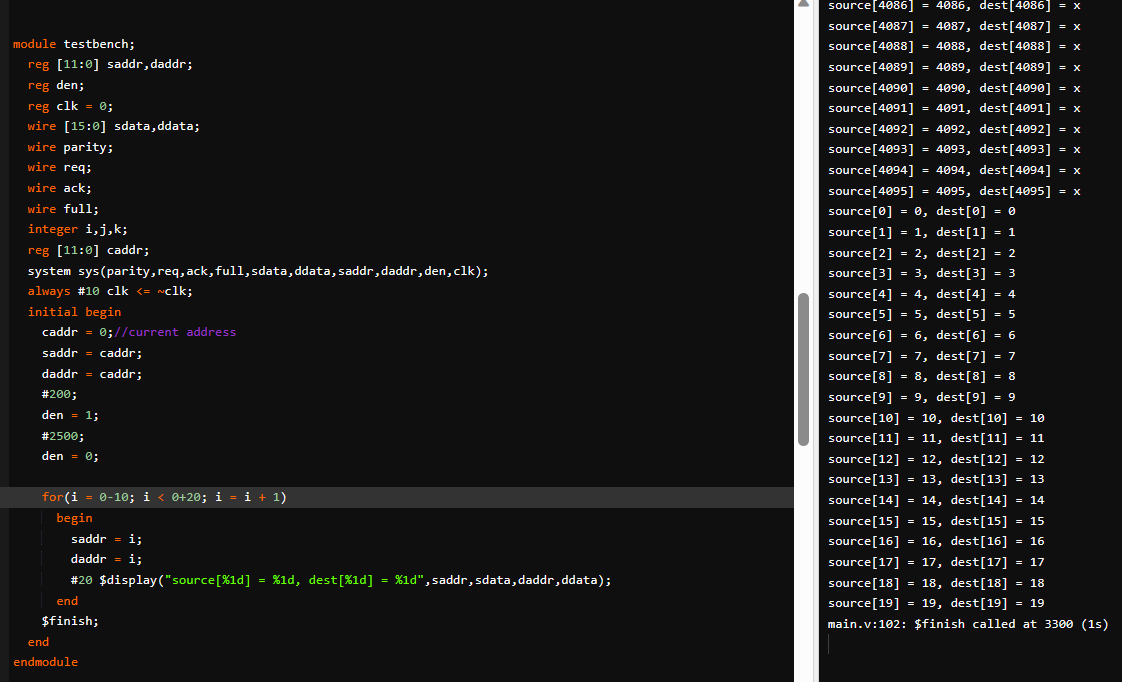


Figure 11.

# Conclusions

In conclusion, we saw that there are many different ways to implement the same functionality but there are ways that is more efficient. we designed the transmitter and the receiver using Verilog to see if our design is true or not, and this is very useful and practical because we reduced the cost by testing it in a cheap environment and after that we can build the hardware later on. Many improvements can be applied to enhance the performance of the system. So we usually may use software tools because it faster.