ISSN: 2581-4419 Volume I Issue I

IMPLEMENTATION OF 8:1 MULTIPLEXER ON FPGA

BAISHALI CHAUDHURY[1] TANISHKA BHIKOT[2] GAURI LAKESHRI[3] RONNIT SINGH[4]
PROF.YOGESH KARUNAKAR[5]

[1-5]Department of Electronics and Telecommunication
K.C. College of Engineering & Management Studies & Research, Kopri, Thane (E)-400 603, India.

Abstract-- In order to transmit multiple signals over the same channel, it is essential to keep the signals apart to avoid the interference between them. In this case, the cost of implementing separate channels for each data source is high. Therefore, we use a multiplexing/demultiplexing function which transmits a composite signal over a medium by combining many signals into one and then separating into original signals at the receiver. In this paper we have implemented a digital 8:1 Multiplexer on FPGA to reduce network costs by minimizing the number of communication links needed between two points. All the results of this paper are simulated on ModelSim tool.

I. INTRODUCTION

Multiplexing is a method that allows the synchronous transmission of various signals across a medium. The individual network signals are input into a multiplexer (mux) that combines them into a compound signal, which is then transmitted through a particular medium. When the compound signal reaches its destination, a demultiplexer (demux) splits the signal back into the original signals and converts them into separate lines for use by other operations. Multiplexer transfers all input signals to a microprocessor, which receives and processes the data, transmits it to the output devices, and controls the system as a whole. It is also known as a data selector as it selects one of the n data inputs and forwards it to the output.

The need for multiplexing is driven by the fact that in utmost applications it's much further provident to transmit data at advanced rates over a single fibre than it is to transmit at lower rates over multiple fibres, in most applications. Multiplexer substantially use to increase the number of data that can be transferred over the network within certain quantity of time and bandwidth.

In this paper, Multiplexer is implemented in FPGA using combinational or sequential logic to ensure a response time that is always the same and under milliseconds. FPGA based designs are easier to program and can be reconfigured anytime without changing the whole machine design.

II. IMPLEMENTATION

II.1 BLOCK DIAGRAM

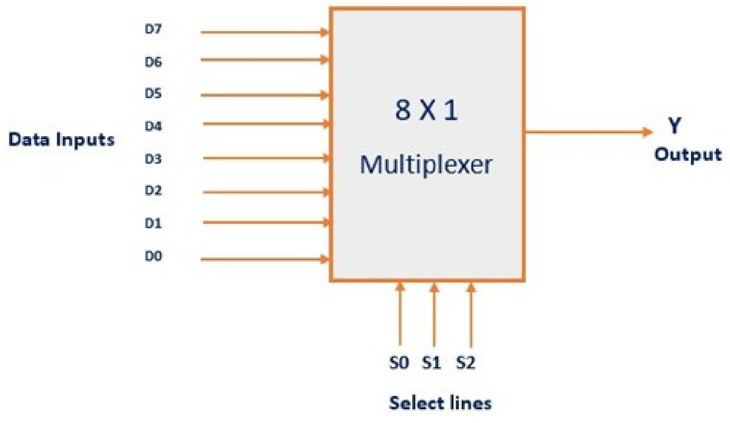

Fig 1: Block diagram of 8:1 Multiplexer

Multiplexer is a combinational circuit that has most of 2^n data inputs, 'n' selection lines with a single output. One of these data inputs will be connected to the output Y based on the values of selection lines. 8 X 1 Multiplexer has 8 data inputs D0, D1, D2, D3, D4, D5, D6 & D7, 3 select lines S0, S1, & S2 and one output Y.

II.2.LOGICAL CIRCUIT DIAGRAM

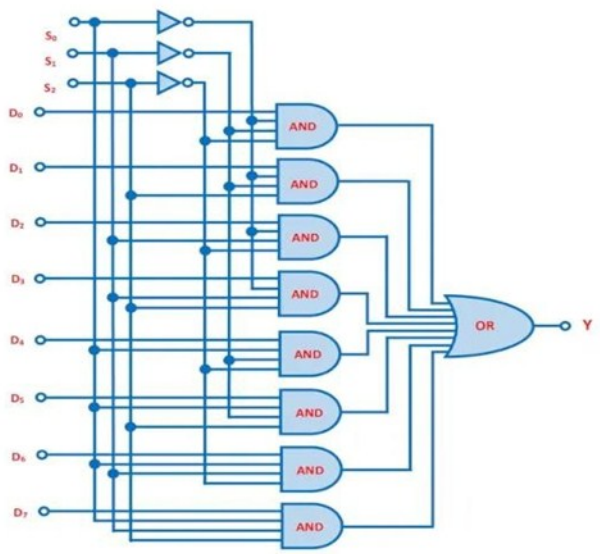

Fig 2. Logical Circuit Diagram of 8X1 MUX

II.3. WORKING

In the logic circuit diagram, 8-to-1 multiplexer can be implemented by using 8 AND gates, 1 OR gate and 3 NOT gates. The output gets connected to only one of the n data inputs at a given instant of time. The output is selected based on the truth table of this integrated circuit.

For e.g., if all the three select inputs S0=0, S1=0, & S2=0 then the topmost AND gate is enabled and all other AND gate is disabled. So, the data input D0 is selected and transmitted as output. Hence, we get the output Y = D0.

INPUTS			**OUTPUTS**
S0	***S1***	***S2***	***Y***
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3
1	0	0	D4
1	0	1	D5
1	1	0	D6
1	1	1	D7

Table 1: Truth Table of 8:1 MUX

II.4 MPLEMENTING 8:1 MUX using LOW ORDER MUX

Larger Multiplexers can be constructed by using lower multiplexers by chaining them together. For illustration, an 8:1 multiplexer can be formed with two 4:1 and one 2:1 multiplexer.

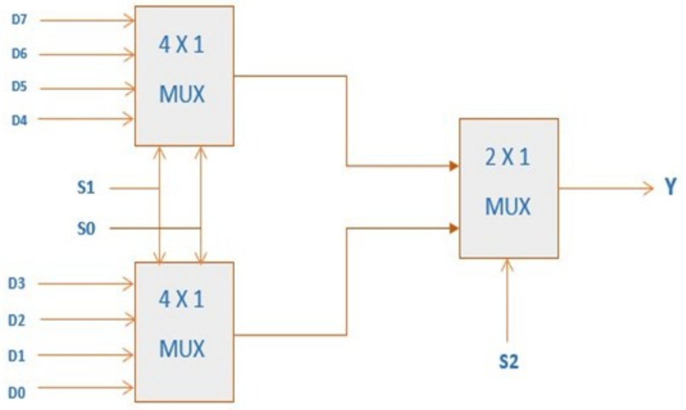

Fig 3: Block diagram of 8:1 MUX using 4:1 MUX &

2:1 MUX

So, we need two 4x1 Multiplexers in first stage in order to get the 8 data inputs. And the yield of each multiplexer will feed into a 2x1 Multiplexer in successive stage by considering the yields of first stage as inputs and to produce the final result Y.

III. SOFTWARE USED

1. MODELSIM: To program FPGA in VERILOG language, we have used the Modelsim Software. It is a multi-language environment by Mentor Graphics, for simulation of hardware description languages such as VHDL, Verilog and SystemC, and includes a built-in C debugger.
2. Alternative Softwares like EDA PLAYGROUND can be used for simulation. It is an open source software, a free web application that allows users to edit, simulate (and view waveforms), synthesize, and share their HDL code. Its goal is to accelerate the learning of design and testbench development with easier code sharing and with simpler access to simulators and libraries It supports several HDLs such as System Verilog, VHDL, MyHDL, and Migen.

IV. RESULTS

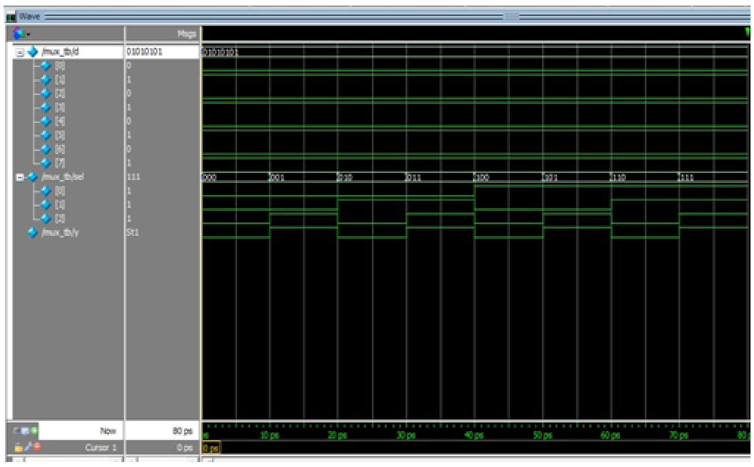

Figure 1. Waveform of 8:1 MUX

V. CONCLUSION

The objective of applying a multiplexer is to produce full usage of the capacity of the communication channel and greatly reduce the cost of the system. The multiplexer improves the effectiveness of the communication system. It is possible to execute number of combinational circuits by using a multiplexer. Multiplexers are fundamental constructing blocks of FPGA boards. The circuit executed in FPGA is significant faster and productive than general CPU's.

REFERENCES

[1] S.M.Turkane, N.V.Ambedkar, S.V.Kaname, T.V.Kharde, "Performance Analysis of different MUX for FPGA", IJARIIE-ISSN(O)-2395-4396, Vol-2 Issue-3 2016
http://ijariie.com/AdminUploadPdf/Performance_A nalysis_of_different_MUX_for_FPGA_ijariie2070.pdf

[2] Multiplexing in Computer Networks
https://www.tutorialspoint.com/data_communicatio n_computer_network/physical_layer_multiplexing.htm#:~:text=Multiplexing%20is%20a%20techniqu e%20by,then%20shared%20by%20different%20streams.

[3] Verilog Design Tutorial

ISSN: 2581-4419 Volume I Issue I

https://www.tutorialspoint.com/vlsi_design/vlsi_de sign_verilog_introduction.html

[4] Introduction to Verilog Programming
https://www.nandland.com/verilog/tutorials/tutorial -introduction-to-verilog-for-beginners.html