M. Tech. 2nd Semester Mini-Project Report

On

Simulation of Combinational Circuits on FPGA

Chinnapareddy Krishna Vamsi

(222CS012)

Guide

Dr. Basavaraj Talawar

Department of Computer science and Engineering,

NITK, Surathkal

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA,

SURATHKAL, MANGALORE - 575025

April, 2023

DECLARATION

I hereby declare that the M. Tech. 2nd Semester **Mini-Project** Report entitled **Simulation of Combinational Circuits on FPGA** which is being submitted to the National Institute of Technology Karnataka Surathkal, in partial fulfilment of the requirements for the award of the Degree of **Master of Technology in Computer Science and Engineering** in the department of **Computer Science and Engineering**, is a bonafide report of the work carried out by me. The material contained in this Report has not been submitted to any University or Institution for the award of any degree.

Chinnapareddy Krishna Vamsi
222CS012
Department of Computer Science and Engineering
NITK, Surathkal

Place: NITK, Surathkal.
Date: 12-04-2023

CERTIFICATE

This is to certify that the M. Tech. 2nd Semester **Mini-Project** Report entitled **SIMULATION OF COMBINATIONAL CIRCUITS ON FPGA** submitted by **Chinnapareddy Krishna Vamsi**, (Roll Number: 222CS012) as the record of the work carried out by him/her, is accepted as the M. Tech. 2nd Semester Mini-Project Report submission in partial fulfilment of the requirements for the award of degree of Master of **Master of Technology in Computer Science and Engineering** in the Department of **Computer Science and Engineering**.

Guide

<Dr. Basavaraj Talawar>

Department of Computer Science and Engineering

NITK, Surathkal

Chairman - DPGC

<Dr. Manu Basavaraj>

Department of Computer Science and Engineering

NITK, Surathkal

Abstract

This project focuses on deploying various combinational circuits on an FPGA. The circuits deployed in this project include a full adder, multiplexer, demultiplexer, encoder, and decoder. The aim of this project is to demonstrate the practical application of combinational circuits and to investigate their performance when deployed on an FPGA.

The project methodology involves designing the circuits using Verilog and implementing them on an FPGA board. The circuits are then tested for functionality and performance, including timing and power consumption. The results of this project demonstrate the successful deployment of combinational circuits on an FPGA and highlight the benefits and drawbacks of using FPGA technology for implementing digital circuits.

The findings of this project have implications for the design and implementation of digital circuits, particularly in the context of FPGA technology. This project provides insight into the capabilities and limitations of FPGAs for deploying combinational circuits, and can inform future research and development in this area. Overall, this project demonstrates the feasibility and potential of using FPGAs for implementing digital circuits.

Keywords: combinational circuits, FPGA, Verilog, full adder, multiplexer, demultiplexer, encoder, decoder

Contents

List of Figures

List of Tables

1 Introduction

Digital circuits are ubiquitous in modern electronics and are used in a wide range of applications, from simple calculators to complex computing systems. Combinational circuits are an essential component of digital circuits, and are used to perform logic operations on input signals. Combinational circuits include a range of logic gates, such as AND, OR, and NOT gates, as well as more complex circuits such as adders, multiplexers, and encoders.

Field Programmable Gate Arrays (FPGAs) have emerged as a popular technology for implementing digital circuits. FPGAs are highly flexible and can be programmed to implement a wide range of circuits, including combinational circuits. They offer several advantages over traditional application-specific integrated circuits (ASICs), including flexibility, lower development costs, and faster time-to-market.

In this project, we focus on the deployment of combinational circuits on an FPGA. We deploy a range of circuits, including a full adder, multiplexer, demultiplexer, encoder, and decoder, and investigate their performance when implemented on an FPGA. The project methodology involves designing the circuits using Verilog HDL and implementing them on an FPGA board. We then test the circuits for functionality and performance, including timing and power consumption.

This project's outcomes showcase the effective utilization of combinational circuits on an FPGA and outline the advantages and disadvantages of incorporating FPGA technology in the implementation of digital circuits. The results of this study have significant implications for digital circuit design and implementation, especially in the realm of FPGA technology. Ultimately, this project sheds light on the possibilities and constraints of utilizing FPGAs to deploy combinational circuits and highlights their feasibility and potential for implementing digital circuits.

2 Introduction to FPGA and its Architecture

Not so long ago, most software was permanently shipped along with their respective hardware, and there was no way to change it. But as technology matured, manufacturers found ways to update the software with additional features on existing hardware.

Now, imagine a future where hardware updates also become a possibility — wouldn't that be fascinating?

Well, such programmable hardware whose sub-system configurations can be modified even after fabrication, falls under the category of Reconfigurable System. And the most predominant integrated circuit that supports reconfigurable computing is FPGA, an acronym for Field Programmable Gate Array.

FPGA enables you to program product features, adapt to new standards, and reconfigure hardware for specific applications even after the product has been installed in the field — hence the term 'field-programmable'. Whereas 'gate arrays' refer to two-dimensional array of logic gates present in its architecture.

All modern personal computers including desktops, notebooks, smartphones, and tablets, are examples of general-purpose computers. General-purpose computing incorporates 'Von Neumann' approach, which states that an instruction fetch and a data operation cannot occur simultaneously. Therefore, being sequential machines, their performance is also limited.

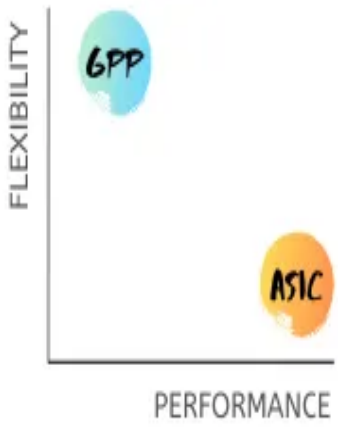

Figure 1: trading off performance of ASICs vs general-purpose processors

On the other hand, we have the Application Specific Integrated Circuits (ASICs) which are customized for a particular task like a digital voice recorder or a high-efficiency Bitcoin miner. An ASIC uses a spatial approach to implement only one application and provides maximum performance. However, it can't be used for the tasks other than those for which it has been originally designed.

Having said that, FPGAs are less energy efficient when compared to ASICs and also not suitable for large volume productions. However, they are reprogrammable and have low NRE costs when compared to an ASIC.

You see, ASICs and FPGAs have different value propositions. Most device manufacturers typically prefer FPGAs for prototyping and ASICs for very large production volumes.

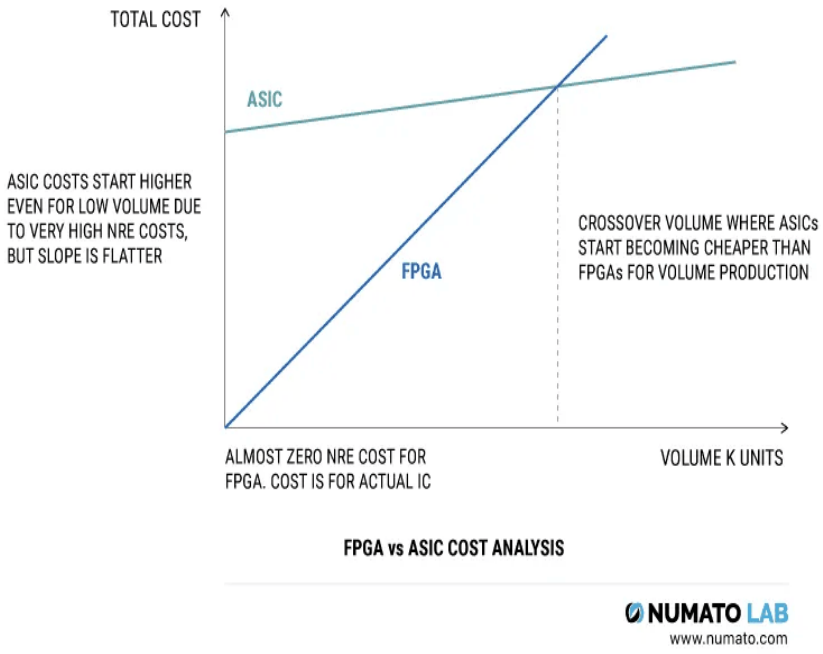

Figure 2: FPGA Vs ASIC cost analysis

FPGAs used to be chosen for lower speed and complex designs in the past, but nowadays FPGAs can easily surpass the 500 MHz performance benchmark.

3 Internal Architecture of FPGA

In 1985, a semiconductor manufacturing company named Xilinx invented the first commercially viable FPGA — XC2064. Another company Altera, which was acquired by Intel in 2015, also pushed the boundaries and drove this market forward along with Xilinx.

FPGA emerged from relatively simpler technologies such as programmable read-only memory (PROM) and programmable logic devices (PLDs) like PAL, PLA, or Complex PLD (CPLD).

It consists of three main parts:

- Configurable Logic Blocks — which implement logic functions.
- Programmable Interconnects — which implement routing.
- Programmable I/O Blocks — which connect with external components.

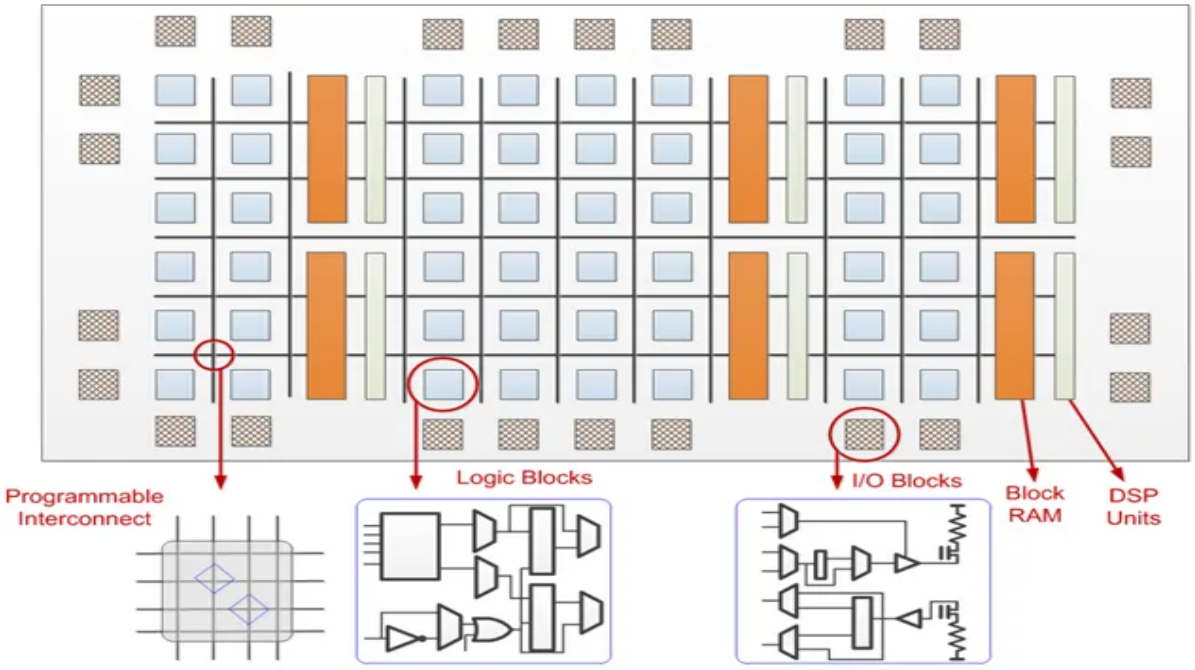

Figure 3: The basic architecture of an FPGA

Logic blocks implement the logical functions required by the design and consist of various components such as transistor pairs, look-up tables (LUTs), flip flops, and multiplexers.

You can think of logic blocks as separate modules like lego blocks which can operate in parallel. Unlike a lego block, a logic block is configurable i.e. its internal state can be controlled and you can hook these together by programming the interconnects in order to build something meaningful. This hierarchy of programmable interconnection is used for allocating resources among configurable logic blocks (CLBs); where routing paths contain wire segments of varying lengths that can be connected via anti-fuse or memory-based techniques.

Each CLB is tied to a switch matrix to access the general routing structure. The switch matrix provides programmable multiplexers, which are used to select the signals in a given routing channel and thereby connect vertical and horizontal lines. Lastly, the I/O blocks (IOBs) are used to interface the CLBs and routing architecture to the external components.

In earlier FPGAs, there was no processor to run any software; hence implementing an application implied designing the circuit from scratch. So, we could have configured an FPGA to be as simple as an OR gate or as complex as the multi-core processor. But we have come a long way since XC2064 and the basic FPGA architecture has developed through the addition of more specialized programmable function blocks like ALUs, block RAM, multiplexers, DSP-48, and microprocessors.

4 Nexys A7 Board

The Nexys A7 board is a complete, ready-to-use digital circuit development platform based on the latest Artix-7™ Field Programmable Gate Array (FPGA) from Xilinx®. With its large, high-capacity FPGA, generous external memories, and collection of USB, Ethernet, and other ports, the Nexys A7 can host designs ranging from introductory combinational circuits to powerful embedded processors. Several built-in peripherals, including an accelerometer, temperature sensor, MEMs digital microphone, a speaker amplifier, and several I/O devices allow the Nexys A7 to be used for a wide range of designs without needing any other components.

Features:

- 4,860 Kbits of fast block RAM (*2,700 Kbits)
- 128MiB DDR2
- microSD card slot
- 16 Switches
- 16 LEDs
- Two RGB LEDs
- Two 4-digit 7-segment displays

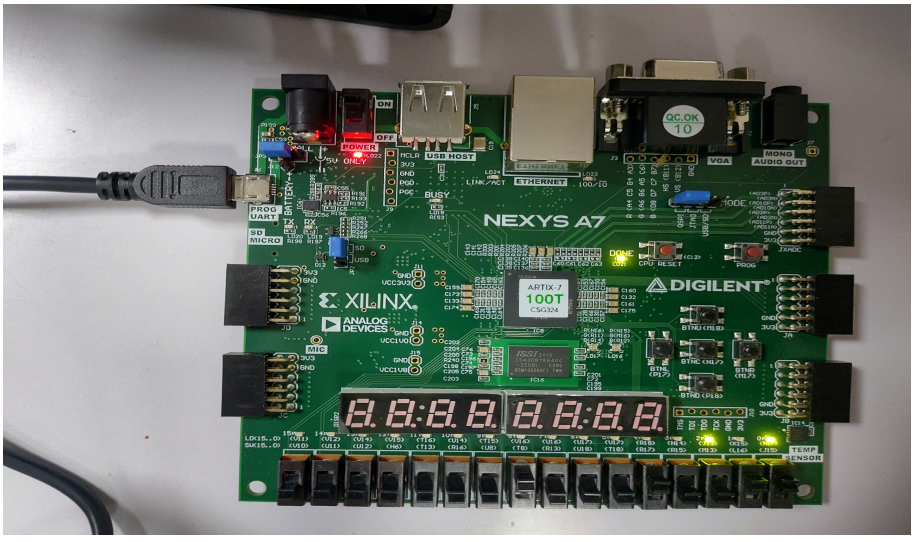

Figure 4: Nexys A7 Board

5 Implementation

5.1 Implementation of Full Adder

Full Adder is the adder that adds three inputs and produces two outputs. The first two inputs are A and B and the third input is an input carry as C-IN. The output carry is designated as C-OUT and the normal output is designated as S which is SUM. A full adder logic is designed in such a manner that can take eight inputs together to create a byte-wide adder and cascade the carry bit from one adder to another. we use a full adder because when a carry-in bit is available, another 1-bit adder must be used since a 1-bit half-adder does not take a carry-in bit. A 1-bit full adder adds three operands and generates 2-bit results.

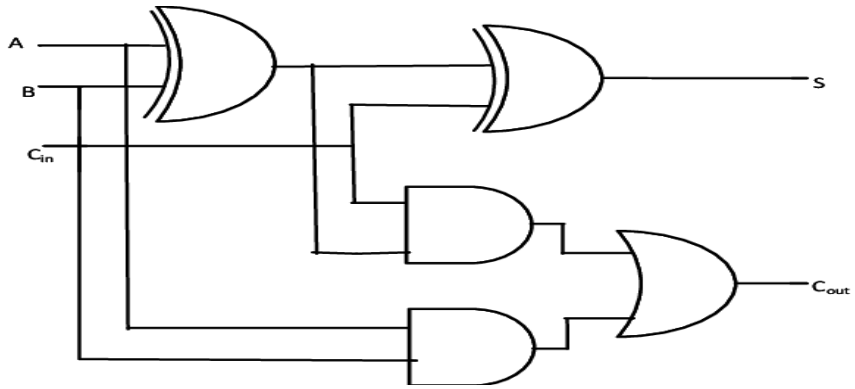

Figure 5: Block diagram of Full Adder

A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Table 1: Truth Table for Full Adder

5.1.1 Verilog Implementation of Full Adder

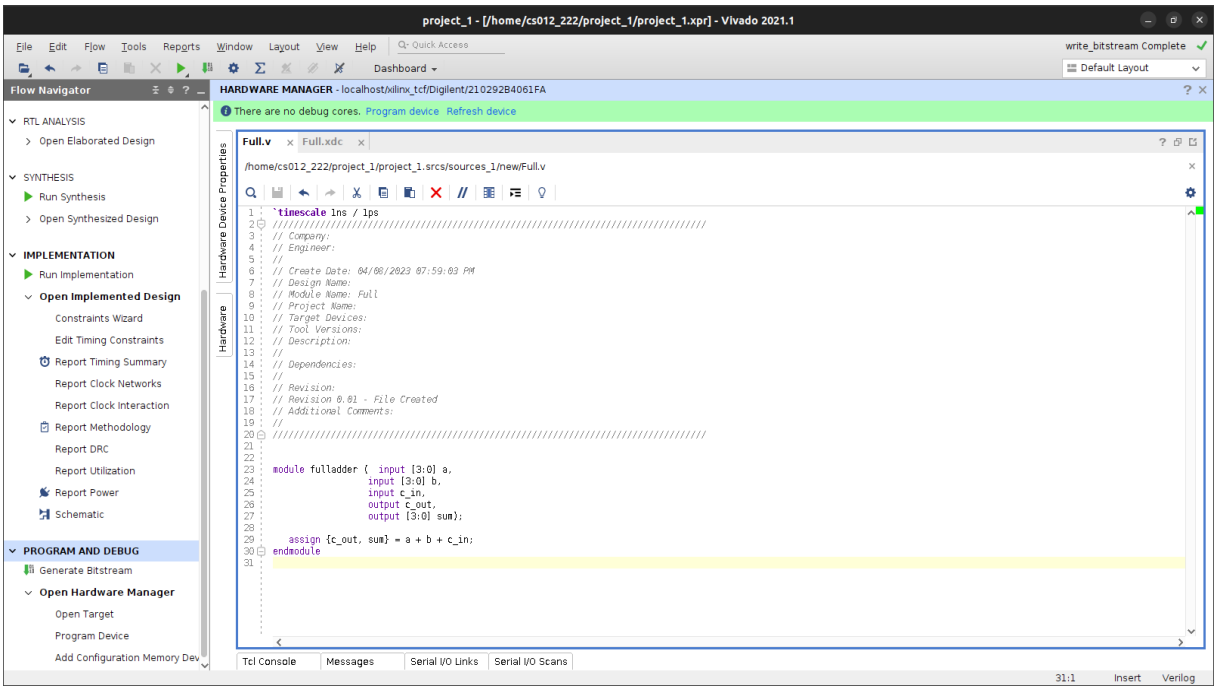

Figure 6: Verilog Implementation of Full Adder

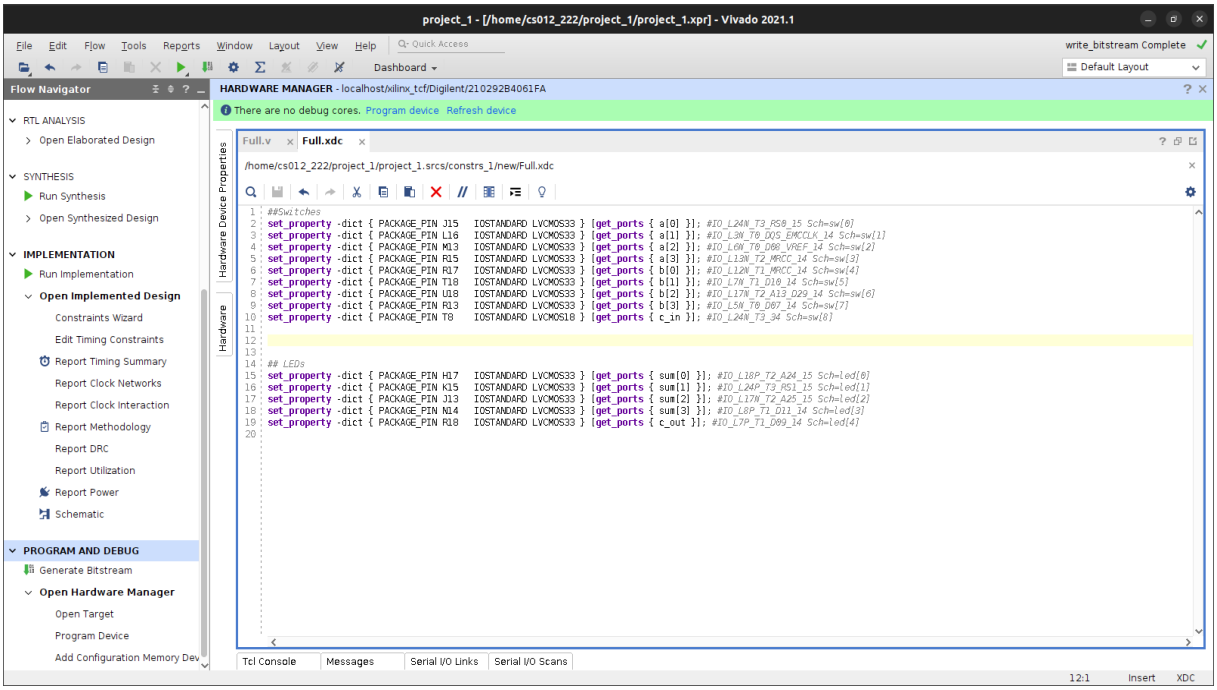

Figure 7: XDC Implementation of Full Adder

5.2 Implementation of Multiplexer

It quite often happens, in the design of large-scale digital systems, that a single line is required to carry two or more different digital signals. Of course, only one signal at a time can be placed on the one line. What is required is a device that will allow us to select, at different instants, the signal we wish to place on this common line. Such a circuit is referred to as a Multiplexer. A multiplexer performs the function of selecting the input on any one of 'n' input lines and feeding this input to one output line.

Multiplexers are used as one method of reducing the number of integrated circuit packages required by a particular circuit design. This in turn reduces the cost of the system.

Assume that we have four lines, C0, C1, C2 and C3, which are to be multiplexed on a single line, Output (f). The four input lines are also known as the Data Inputs. Since there are four inputs, we will need two additional inputs to the multiplexer, known as the Select Inputs, to select which of the C inputs is to appear at the output. Call these select lines A and B. The gate implementation of a 4-line to 1-line multiplexer is shown below:

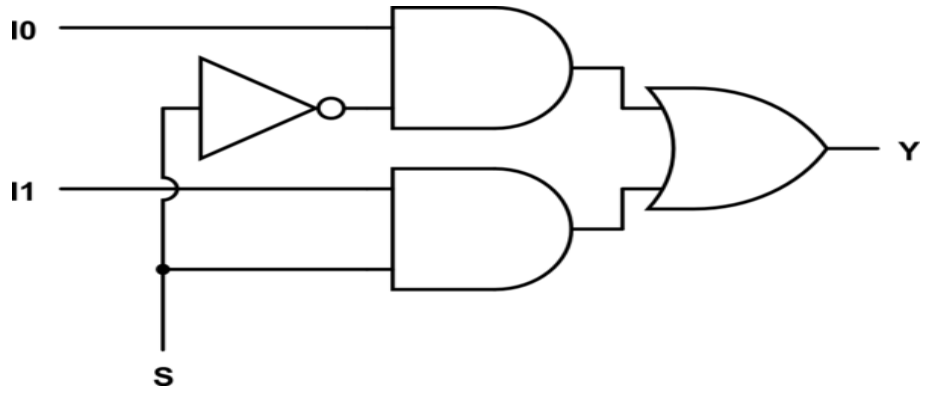

Figure 8: Block diagram of Multiplexer

S	D_0	Y
0	I_0	D_0
1	I_1	D_1

Table 2: Truth Table for Multiplexer

5.2.1 Verilog Implementation of Multiplexer

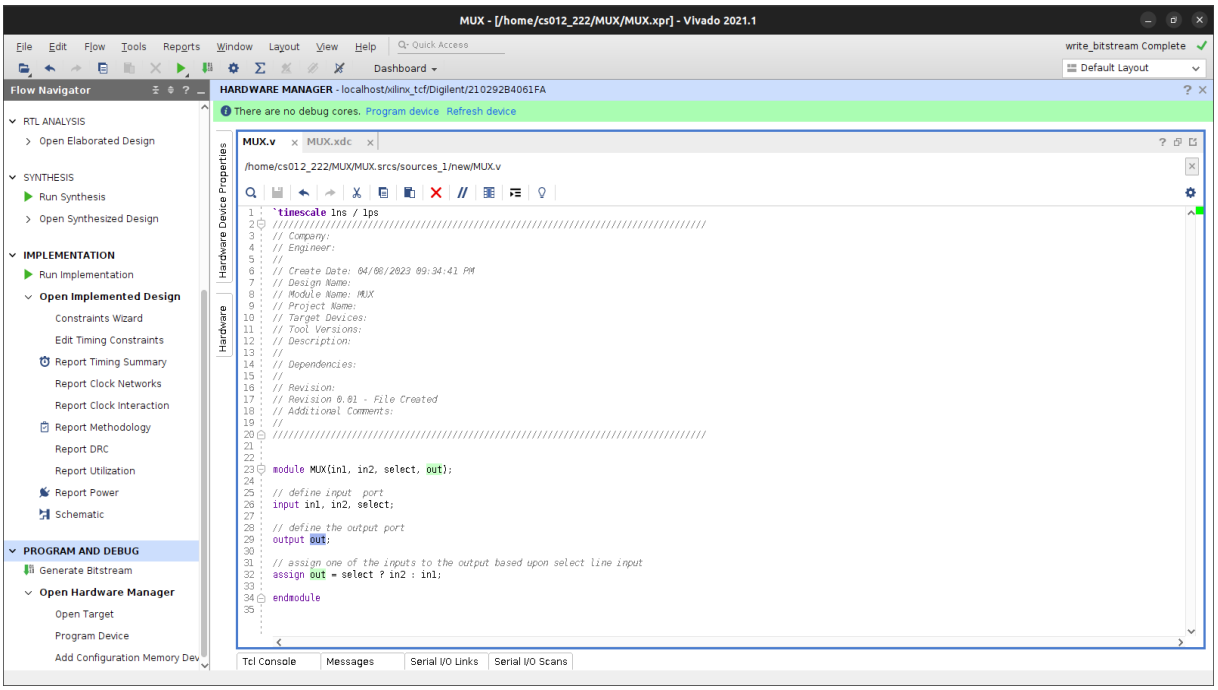

Figure 9: Verilog Implementation of Multiplexer

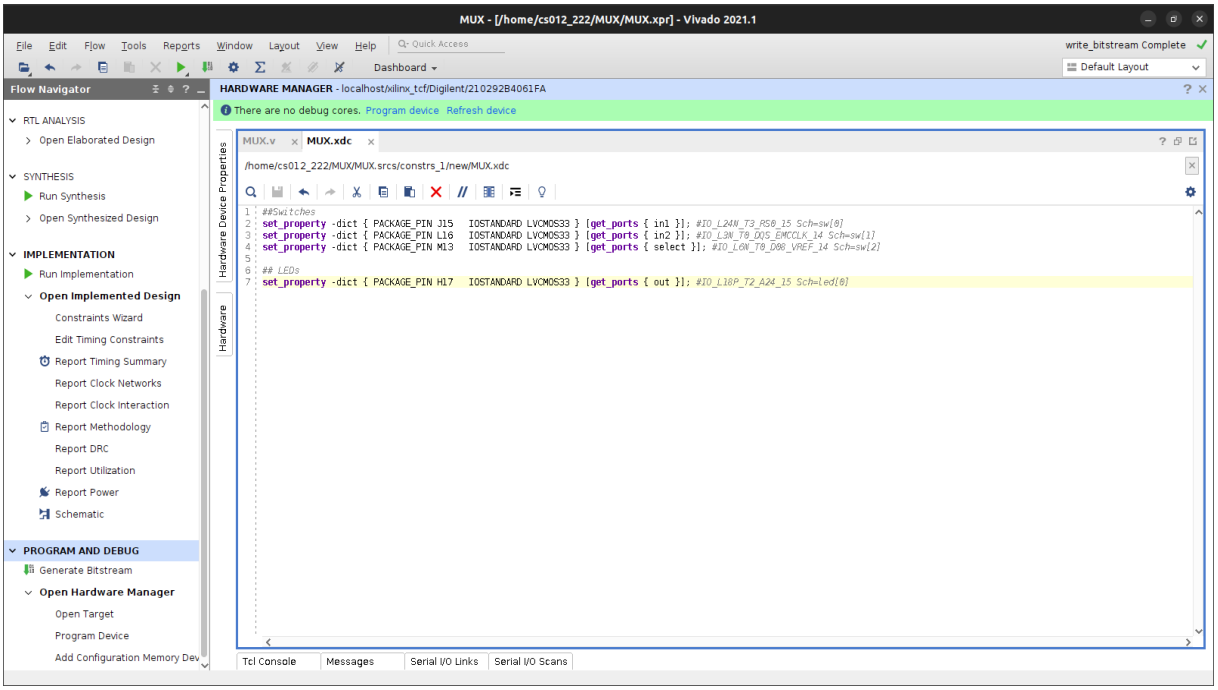

Figure 10: XDC Implementation of Multiplexer

5.3 Implementation of De-Multiplexer

Demultiplexer is a data distributor which takes a single input and gives several outputs.In demultiplexer we have 1 input and 2n output lines where n is the selection line. It works on one to many operational principle. In time division Multiplexing, demultiplexer is used at the receiver end

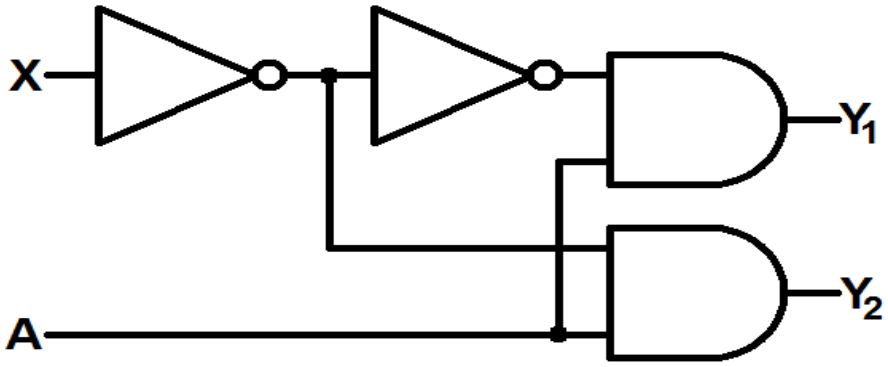

Figure 11: Block diagram of DeMultiplexer

D	$S=0$	$S=1$
0	0	0
1	1	0

Table 3: Truth Table for DeMultiplexer

5.3.1 Verilog Implementation of DeMultiplexer

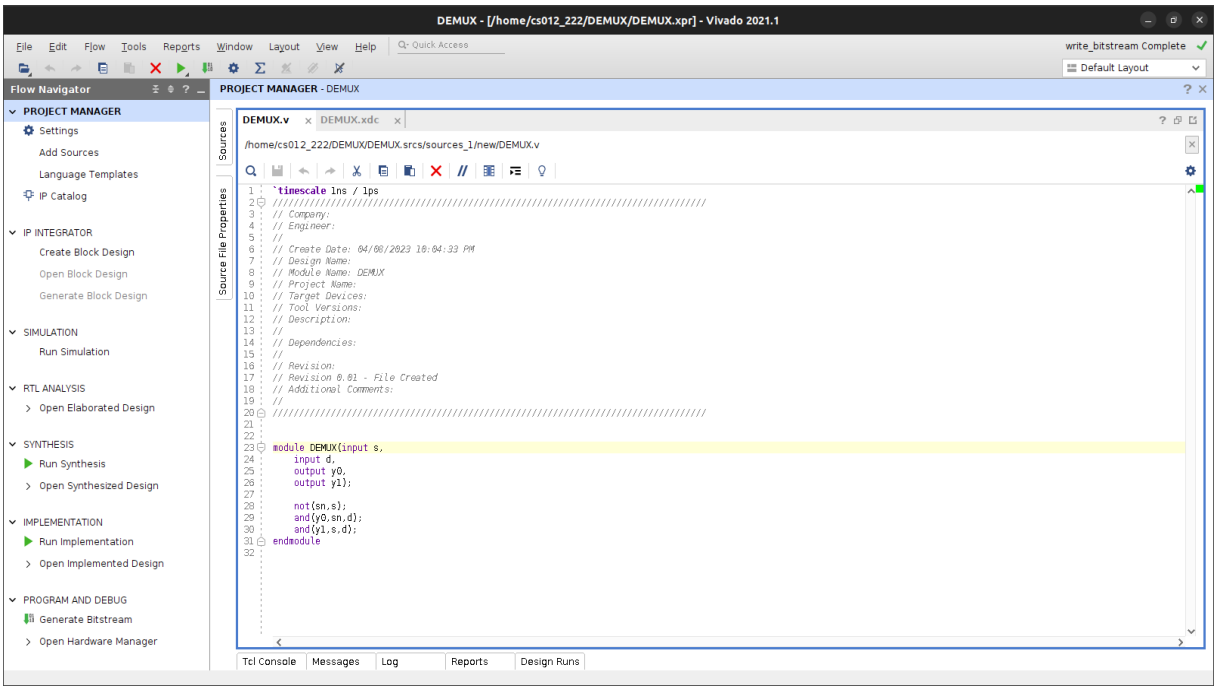

Figure 12: Verilog Implementation of DeMultiplexer

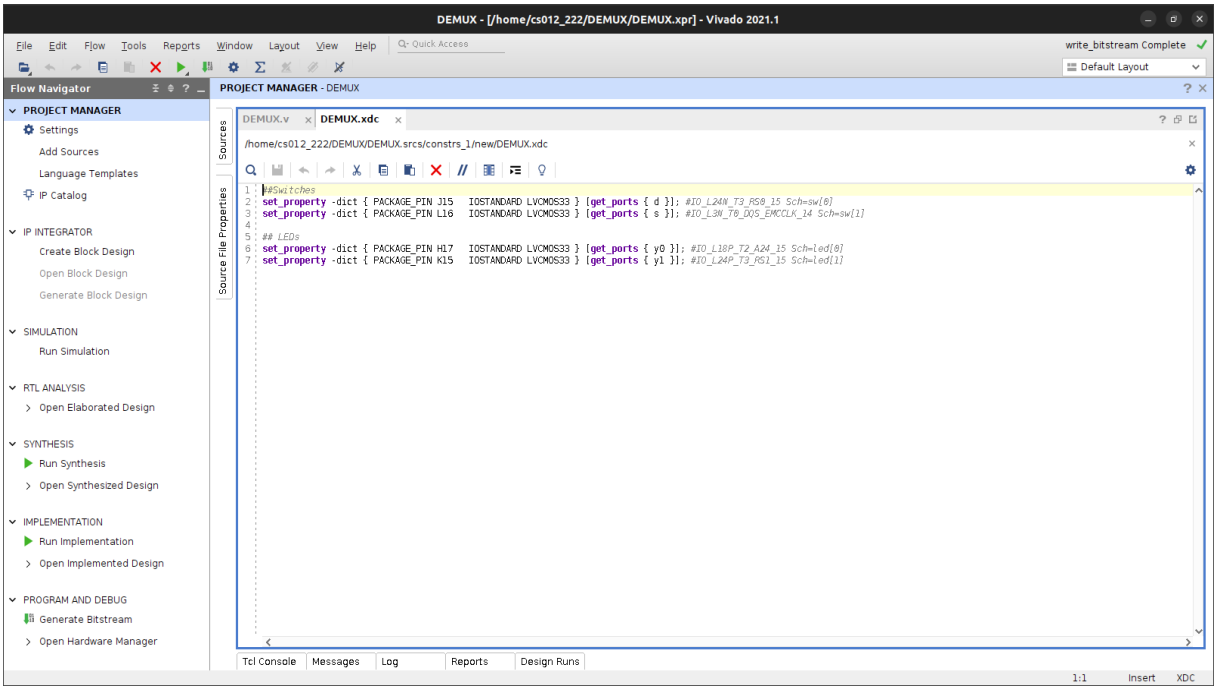

Figure 13: XDC Implementation of DeMultiplexer

5.4 Implementation of Encoder

An encoder is a digital circuit that converts a set of binary inputs into a unique binary code. The binary code represents the position of the input and is used to identify the specific input that is active. Encoders are commonly used in digital systems to convert a parallel set of inputs into a serial code.

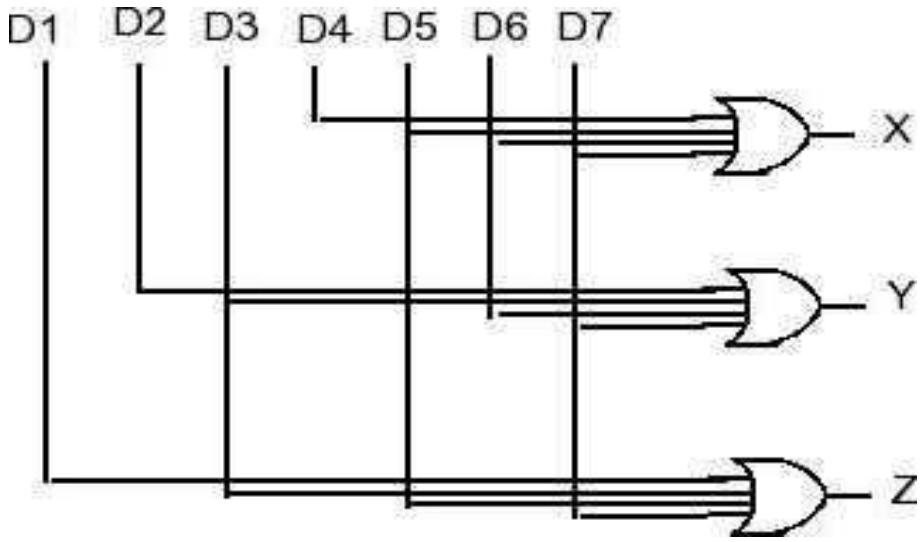

Figure 14: Block diagram of Encoder

A	B	C	D_0	D_1	D_2
0	0	0	1	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	0	0	0

Table 4: Truth Table for Encoder

5.4.1 Verilog Implementation of Encoder

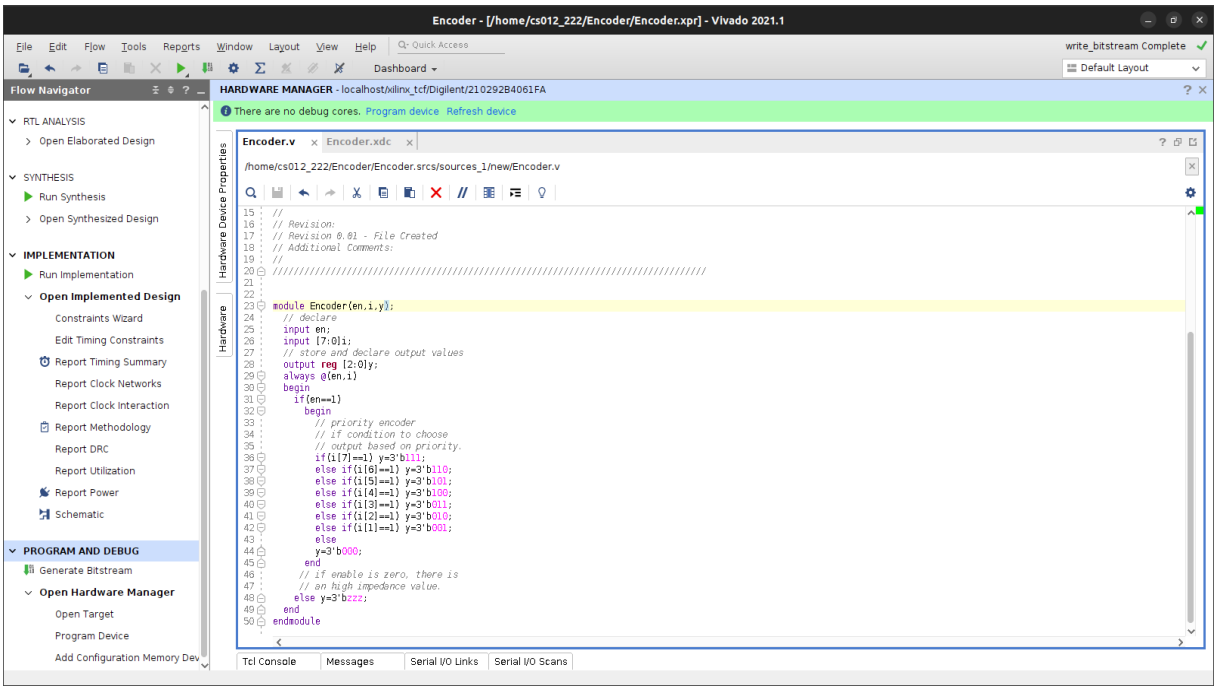

Figure 15: Verilog Implementation of Encoder

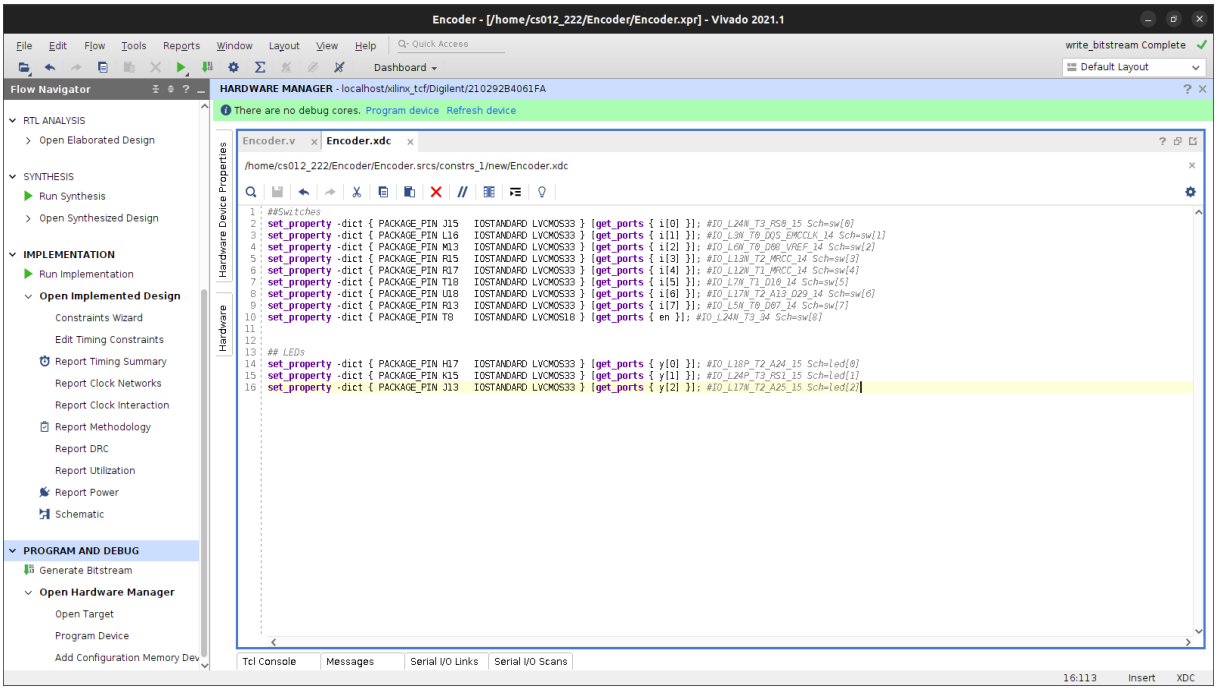

Figure 16: XDC Implementation of Encoder

5.5 Implementation of Decoder

A binary decoder is a digital circuit that converts a binary code into a set of outputs. The binary code represents the position of the desired output and is used to select the specific output that is active. Binary decoders are the inverse of encoders and are commonly used in digital systems to convert a serial code into a parallel set of outputs.

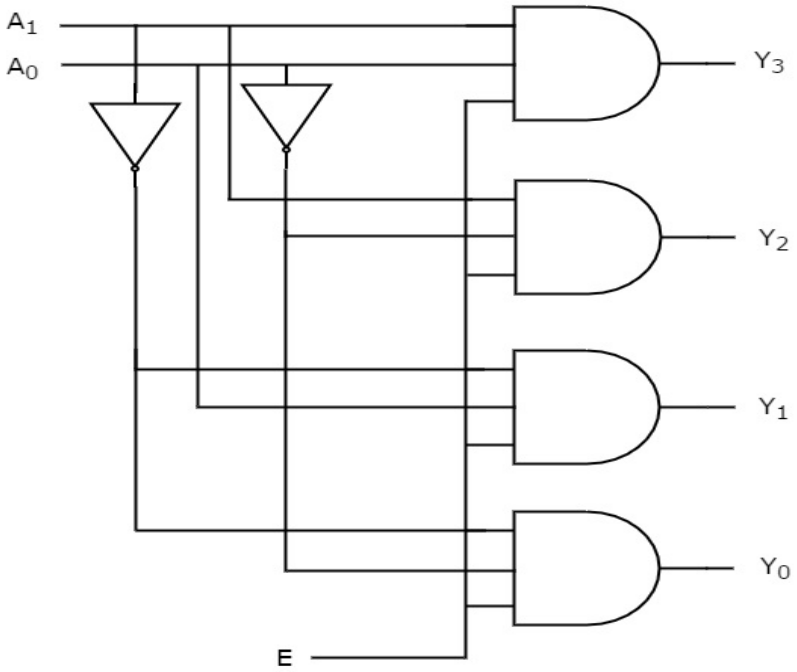

Figure 17: Block diagram of Decoder

A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Table 5: Truth Table for Decoder

5.5.1 Verilog Implementation of Decoder

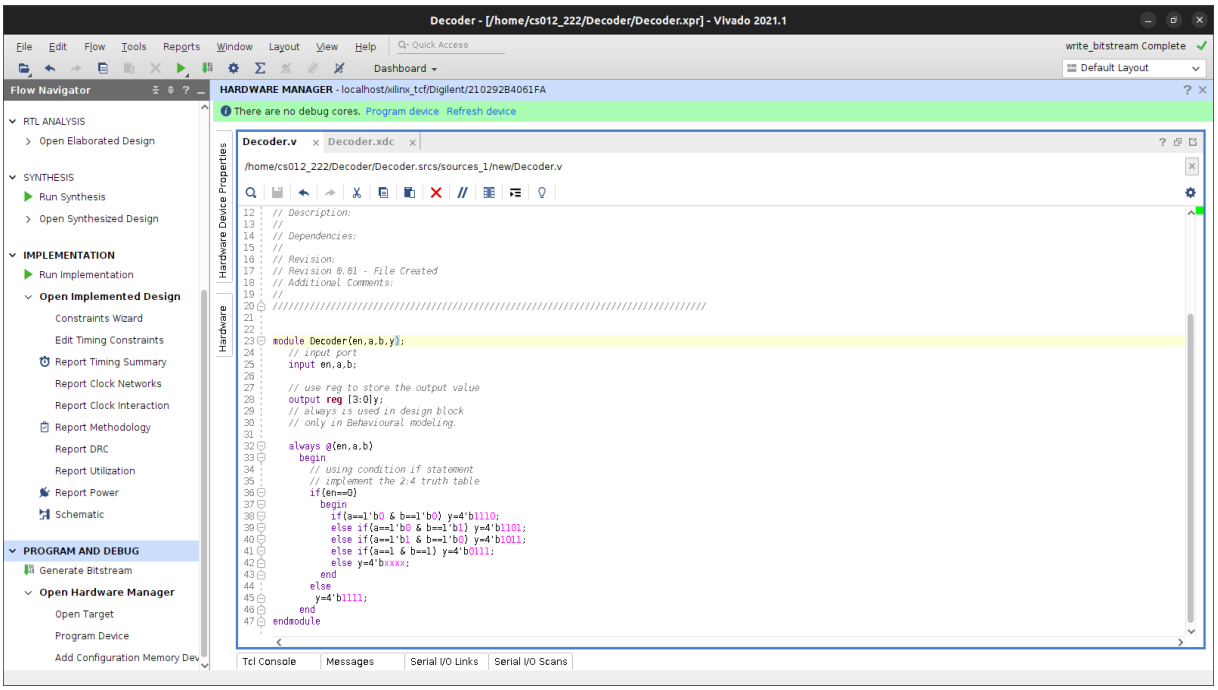

Figure 18: Verilog Implementation of Decoder

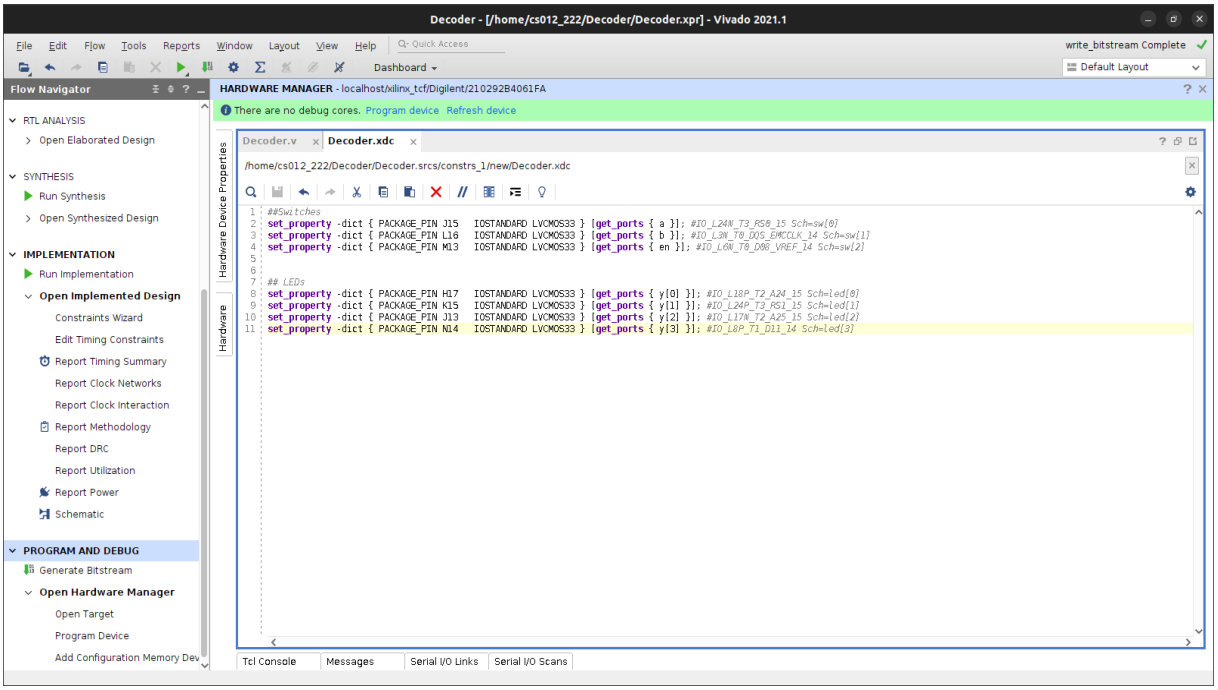

Figure 19: XDC Implementation of Decoder

6 Conclusion and Future Work

In this project, we implemented several combinational circuits including a full adder, multiplexer, demultiplexer, encoder, and decoder on an FPGA board. These circuits are fundamental building blocks in digital systems, and are widely used in various applications.

In our implementation, we used the Verilog hardware description language to describe the behavior of each circuit, and simulated their functionality using a software simulation tool vivado. We then synthesized the design and generated a bitstream file that could be loaded onto the FPGA board. We tested the functionality of each circuit on the FPGA board using input test vectors, and verified the correctness of the output signals.

Overall, this project provided us with hands-on experience in designing and implementing combinational circuits on an FPGA board. We gained a better understanding of the underlying digital logic and hardware design concepts, and learned how to use Verilog and FPGA tools to develop and test digital circuits.

Industry trends are driving FPGAs toward playing a big part in the heterogeneous computing paradigm. Here, heterogeneous computing refers to systems that use more than one type of processor to perform specialized processing capabilities. And all these different processors, including FPGAs, can be programmed via OpenCL — which is an industry-standard development platform.

The thing is FPGA provides cost-efficient parallel computing power which makes it suitable for rapid prototyping. There are cases where FPGA also outperforms GPU when testing a neural network. While GPU might be good for training but when it comes to real-time applications, FPGA is more adaptable.