Summer Internship Report
on
Mixed Signal/Digital Simulation in eSim

Submitted by
Nameera Jabi
B.Tech
Electronics and Communication Engineering
Jamia Millia Islamia,New Delhi.

Under the guidance of
Prof. Kannan M. Moudgalaya
Chemical Engineering Department
IIT Bombay

June 7, 2024

Acknowledgement

I am thankful to the FOSSEE team for providing me with an incredible and educational opportunity. Their support and guidance have made a significant impact on my learning journey. I am grateful for the valuable experience they have given me. I am deeply grateful to the FOSSEE Team at IIT Bombay for entrusting me with this project and believing in my capabilities. This opportunity has been invaluable to my growth and development, and I sincerely appreciate their support.

I would like to express my gratitude to Prof. Kannan M. Moudgalaya for his valuable and constructive guidance throughout this FOSSEE fellowship program.

I would like to express my heartfelt appreciation to my mentors, Mr. Sumanto Kar, Mr. Rahul Paknikar, Mrs. Madhuri Kadam, Prof. Inderjit Singh Dhanjal, Mrs. Usha Vishwanathan, and the entire team. Their unwavering support and invaluable guidance throughout my internship have been instrumental in my growth. I am truly grateful for their wealth of knowledge and constructive suggestions that have greatly enhanced my learning experiences.

I am committed to utilizing everything I have gained here for my personal and professional growth, as well as for the advancement of our society.

Contents

1 Introduction

1.1 FOSSEE

FOSSEE (Free/Libre and Open Source Software for Education) is a project part of the National Mission on Education through Information and Communication Tech- nology (ICT), Ministry of Human Resource Development (MHRD), Government of India. FOSSEE has developed various open source tools and promotes the use of these tools in improving the quality of education and helping every individual avail these sources free of cost. The softwares is being developed in such a way that it can stay relevant with respect to the commercial softwares.

1.2 eSim

eSim is a free/libre and open source Electronic Design Automation (EDA) tool developed by FOSSEE (Free and Open Source Software for Education) at IIT Bombay. It provides a comprehensive platform for circuit design, simulation, analysis, and PCB (Printed Circuit Board) design. eSim is built using various free/libre and open source software components, including:
1. KiCad: A popular EDA suite that offers schematic capture and PCB layout tools.
2. Ngspice: A mixed-level/mixed-signal circuit simulator that can perform analog, digital, and mixed-signal simulations.
3. NGHDL: An open source VHDL simulator that enables simulation and analysis of digital circuits.
4. GHDL: Another open source VHDL simulator that supports the IEEE 1076 VHDL standard.

1.3 NGHDL

NGHDL is a mixed mode circuit simulator developed by FOSSEE, using NgSpice and GHDL. The NGHDL feature makes it easier to create models for eSims simulation of mixed-signal circuits defined by users. In NGHDL, the analog and digital components communicate through sockets and NgSpice is used to simulate the analog components and GHDL to simulate the digital components. This feature was added to eSim so that a user who is familiar with designing circuits in Verilog can do so with eSim. In order to write

Verilog code for a digital model and install it as a model in Ngspice, NGHDL users an interface.

1.4 Makerchip

Makerchip is a browser-based IDE (Integrated Development Environment) that allows users to simulate Verilog, System Verilog, and TL-Verilog. It is developed using Verilator, which converts Verilog into C++ objects. Before using NgVeri in eSim, the design can be simulated in Makerchip with random inputs to ensure that it produces the desired and consistent results. Once the design is successfully simulated, it can be used in mixed-signal designs. These models can be used in digital/mixed signal simulations. Here are some key features of Makerchip:

1. Browser-Based Environment: Makerchip allows users to perform Verilog design tasks directly from their web browser. This eliminates the need for local installations and provides flexibility in accessing the tools from any device with internet access.
2. Code, Compile, Simulate, and Debug: Users can write Verilog code, compile it into a circuit representation, simulate the behavior of the design, and debug any issues directly within the Makerchip environment. This integrated workflow streamlines the design process, reducing the need for switching between different tools.
3. Seamless Integration: Makerchip offers tight integration between code, block diagrams, waveforms, and novel visualization capabilities. This integration enhances the design experience by providing a cohesive and intuitive interface for designing, visualizing, and analyzing circuits.
4. Advanced Verilog Design Capabilities: Makerchip introduces innovative features and capabilities for advanced Verilog design. It incorporates groundbreaking functionalities that facilitate complex digital circuit design tasks. These capabilities empower users to tackle sophisticated design challenges effectively.
5. User-Friendly Design Experience: Makerchip aims to make circuit design easy and enjoyable for users of all skill levels. The platform provides a user-friendly interface, intuitive workflows, and a range of helpful features that simplify the design process and enhance the overall user experience.

1.5 Ngspice

ngspice is an open-source simulation program for electric and electronic circuits based on the SPICE (Simulation Program with Integrated Circuit Emphasis) simulation engine. It provides a powerful platform for simulating a wide range of circuits, including analog, digital, and mixedsignal circuits.
Some key features and capabilities of ngspice:
1. Component Support: ngspice supports a variety of components and devices, including JFETs, bipolar and MOS transistors, passive elements such as resistors (R), inductors (L), and capacitors (C), diodes, transmission lines, and more. These elements can be interconnected in a circuit using a netlist.
2. Mixed-Signal Simulation: ngspice allows you to simulate mixed-signal circuits, which combine both analog and digital components. This enables the analysis of complex systems that involve both continuous and discrete signals.
3. Comprehensive Device Models: ngspice provides a wide range of device models for active and passive components, covering both analog and digital elements. These models are sourced from various collections, semiconductor manufacturers, and semiconductor foundries. They include accurate descriptions of device behavior and characteristics.
4. Graphical Outputs and Data Logging: The simulation results in ngspice can be visualized through graphs showing currents, voltages, and other electrical quantities. Additionally, the simulation data can be saved in a data file for further analysis and processing.
5. Event-Driven Simulation: ngspice utilizes an event-driven simulation approach, which ensures efficient and fast simulation of digital circuits. It can handle circuits ranging from simple gates to complex digital systems.

2 Features of eSim

1. Circuit Design and Schematic Capture: eSim provides a user-friendly interface for designing electronic circuits. It offers a schematic capture tool where users can create circuit diagrams by placing and connecting components.
2. Component Libraries: eSim includes extensive libraries of electronic components, such as resistors, capacitors, inductors, transistors, diodes, and integrated circuits (ICs). These libraries help users easily access and integrate components into their designs.
3. Symbol and Footprint Creation: Users can create custom symbols and footprints for components that are not available in the existing libraries. This allows for the inclusion of specialized or unique components in circuit designs.
4. Circuit Simulation: eSim integrates powerful simulation engines, such as Ngspice, to simulate the behavior of electronic circuits. It supports analog, digital, and mixed-signal simulations, enabling comprehensive analysis of circuit performance.
5. Waveform Viewer: The tool provides a waveform viewer that allows users to visualize simulation results in the form of waveforms. This helps in analyzing and understanding circuit behavior, including voltage levels, currents, and signal timing.
6. PCB Design and Layout: eSim seamlessly integrates with KiCad, a popular open-source PCB design tool. Users can transfer their circuit designs from eSim to KiCad for further PCB layout and routing.
7. Interactive Simulation and Analysis: This feature facilitates real-time monitoring and evaluation of circuit performance.
8. Open Source and Customization: eSim is built using free/libre and open-source software, providing users with the freedom to modify and customize the tool according to their specific requirements. It encourages collaboration and community-driven development.
9. Educational Resources: eSim is developed with a focus on education. It provides educational resources like tutorials, documentation, and example circuits to help users learn and understand various aspects of circuit design and simulation.
10. Cross-Platform Support: eSim is designed to work on multiple operating systems, including Windows, Linux, and macOS, ensuring accessibility for users across different platforms.

3 Problem Statement

Implementing open source microcontrollers/processors using NGHDL present in eSim so that the user can simulate these processors in eSim by changing the in- struction sets which can be changed from

le or by providing instruction through the input pins in KiCad.

3.1 Approach

The general approach which was implemented to the problem statement was first searching for an open source processor on GitHub Project which has open source licensing such as MIT license. Then the implementation process is as follows:
The verilog code,each was tested in Vivado and then fed to Ngveri to see if it converts correctly without throwing any error.
Then that individual block is simulated and the Ngspice waveform is generated to check the desired waveform.
After all the components are simulated the blocks are then simulated in makerchip and a module is created to link the input instruction with that of the processor top. Then the simulations and conversion takes place by adding the other files as dependencies in ngveri.
The file is simulated in Ngspice to check the result of the processor instruction set.

4 Adder

4.1 About

The 4-bit adder circuit will incorporate a series of full adder subcircuits to perform the addition operation. Each full adder will take in two bits from A and B, as well as the carry generated from the previous stage. The full adders will produce the corresponding sum bit and the carry bit for that stage. The carry generated from the most significant stage (bit 3) will be the final carry output of the 4-bit adder. The sum bits from each stage will be combined to form the 4-bit sum output.

4.2 Verilog Code

```
module adder(PCINPUT,RESULT);
input [31:0] PCINPUT;
output [31:0] RESULT;
reg RESULT;
always@(PCINPUT)
begin
RESULT = PCINPUT+ 4;
end
endmodule
```

4.3 Schematic Diagram

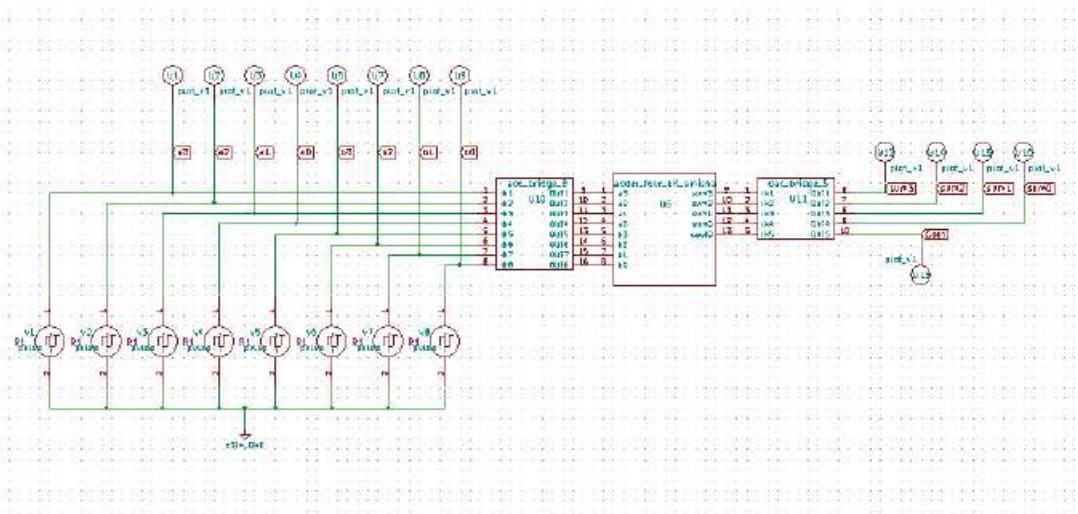

Figure 1: 4-bit Adder

5 4-bit Counter

5.1 About

A 4-bit counter is a digital device that counts from 0 to 15 (in binary: 0000 to 1111) and then wraps around back to 0. It consists of four flip-flops, with each flip-flop representing one bit. Counters can be implemented in various ways, such as using synchronous or asynchronous designs.

5.2 Verilog Code

```
module counter(clk,reset,updown,load,data,count);
input clk,reset,load,updown;
input [3:0] data;
output reg [3:0] count;
always@(posedge clk)
begin
if(reset) //Set Counter to Zero
count <= 0;
else if(load)
count <= data;
else if(updown)
count <= count + 1;
else
count <= count - 1;
end
endmodule
```

5.3 Schematic Diagram

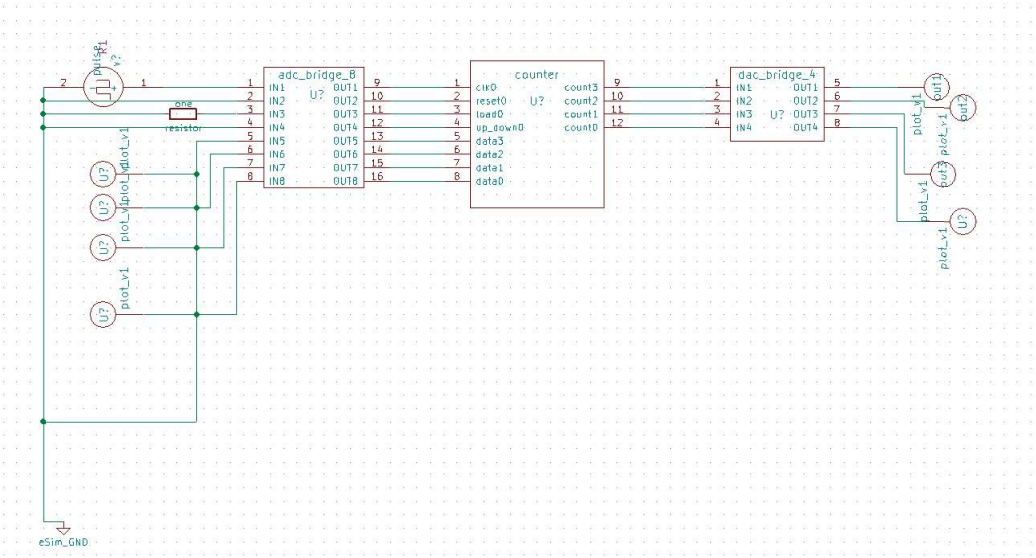

Figure 2: 4-bit Counter

6 Barrel Shifter

6.1 About

The barrel shifter is a digital circuit implemented using pure combinational logic, which enables shifting a data word by a specified number of bits. It eliminates the need for sequential logic elements, making it a highly efficient and fast solution for shifting operations. The circuit design employs a parallel structure, allowing simultaneous shifting of multiple bits in parallel. By utilizing combinational logic elements, the barrel shifter achieves a high-speed data shifting capability while maintaining simplicity in its architecture. This circuit serves as a crucial component in digital systems and processors where efficient data manipulation and rearrangement are required.

6.2 Verilog Code

```
module barrel_shifter_8bit (in, ctrl, out);
input [7:0] in;
input [2:0] ctrl;
output [7:0] out;
wire [7:0] x,y;
//4 bit shift right
mux2X1 ins17 (.in0(in[7]),.in1(1'b0),.sel(ctrl[2]),.out(x[7]));
mux2X1 ins16 (.in0(in[6]),.in1(1'b0),.sel(ctrl[2]),.out(x[6]));
mux2X1 ins15 (.in0(in[5]),.in1(1'b0),.sel(ctrl[2]),.out(x[5]));
mux2X1 ins14 (.in0(in[4]),.in1(1'b0),.sel(ctrl[2]),.out(x[4]));
mux2X1 ins13 (.in0(in[3]),.in1(in[7]),.sel(ctrl[2]),.out(x[3]));
mux2X1 ins12 (.in0(in[2]),.in1(in[6]),.sel(ctrl[2]),.out(x[2]));
mux2X1 ins11 (.in0(in[1]),.in1(in[5]),.sel(ctrl[2]),.out(x[1]));
mux2X1 ins10 (.in0(in[0]),.in1(in[4]),.sel(ctrl[2]),.out(x[0]));
//2 bit shift right
mux2X1 ins27 (.in0(x[7]),.in1(1'b0),.sel(ctrl[1]),.out(y[7]));
mux2X1 ins26 (.in0(x[6]),.in1(1'b0),.sel(ctrl[1]),.out(y[6]));
mux2X1 ins25 (.in0(x[5]),.in1(x[7]),.sel(ctrl[1]),.out(y[5]));
mux2X1 ins24 (.in0(x[4]),.in1(x[6]),.sel(ctrl[1]),.out(y[4]));
mux2X1 ins23 (.in0(x[3]),.in1(x[5]),.sel(ctrl[1]),.out(y[3]));
mux2X1 ins22 (.in0(x[2]),.in1(x[4]),.sel(ctrl[1]),.out(y[2]));
mux2X1 ins21 (.in0(x[1]),.in1(x[3]),.sel(ctrl[1]),.out(y[1]));
```

mux2X1 ins20 (.in0(x[0]),.in1(x[2]),.sel(ctrl[1]),.out(y[0]));
//1 bit shift right
mux2X1 ins07 (.in0(y[7]),.in1(1'b0),.sel(ctrl[0]),.out(out[7]));
mux2X1 ins06 (.in0(y[6]),.in1(y[7]),.sel(ctrl[0]),.out(out[6]));
mux2X1 ins05 (.in0(y[5]),.in1(y[6]),.sel(ctrl[0]),.out(out[5]));
mux2X1 ins04 (.in0(y[4]),.in1(y[5]),.sel(ctrl[0]),.out(out[4]));
mux2X1 ins03 (.in0(y[3]),.in1(y[4]),.sel(ctrl[0]),.out(out[3]));
mux2X1 ins02 (.in0(y[2]),.in1(y[3]),.sel(ctrl[0]),.out(out[2]));
mux2X1 ins01 (.in0(y[1]),.in1(y[2]),.sel(ctrl[0]),.out(out[1]));
mux2X1 ins00 (.in0(y[0]),.in1(y[1]),.sel(ctrl[0]),.out(out[0]));
endmodule

6.3 Schematic Diagram

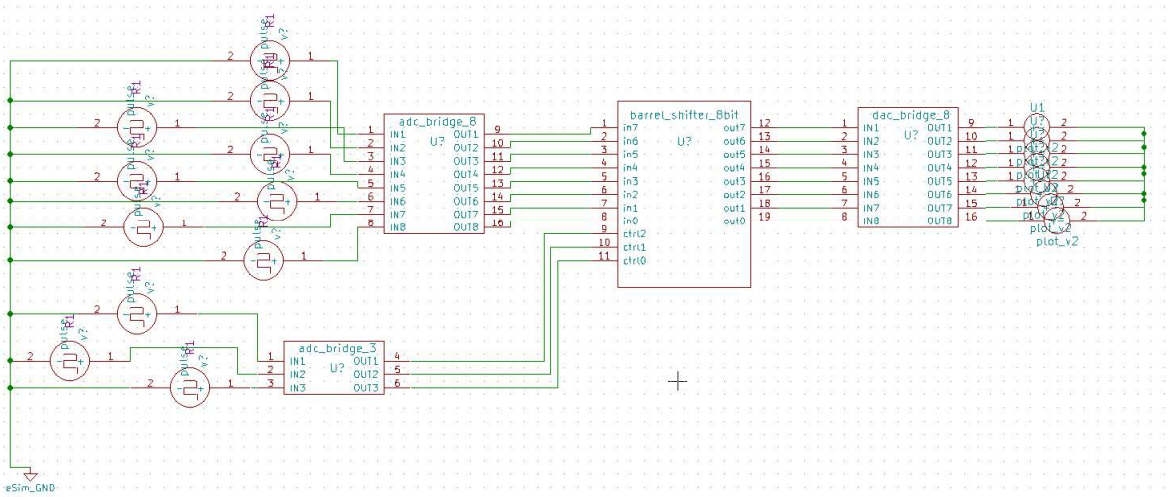

Figure 3: 8-bit Barrel Shifter

7 D Flip Flop

7.1 About

D flip-flop is the most important flip-flop in digitial circuit.D flip-flop is also known as delay type flip-flop because output of d flip-flop is 1 clock pulse delay of the input appled to the D flip-flop. When there is negative edge trigger clock, it stores the previous input applied to the flip-flop. In positive edge trigger of clock, input of the D flip-flop is stored.

7.2 Verilog Code

```
module dff(clk,reset,d,q);
input clk,reset,d;
output reg q;
always @ (posedge clk)begin
if(reset)
q <= 0;
else
q <= d;
end
endmodule
```

7.3 Schematic Diagram

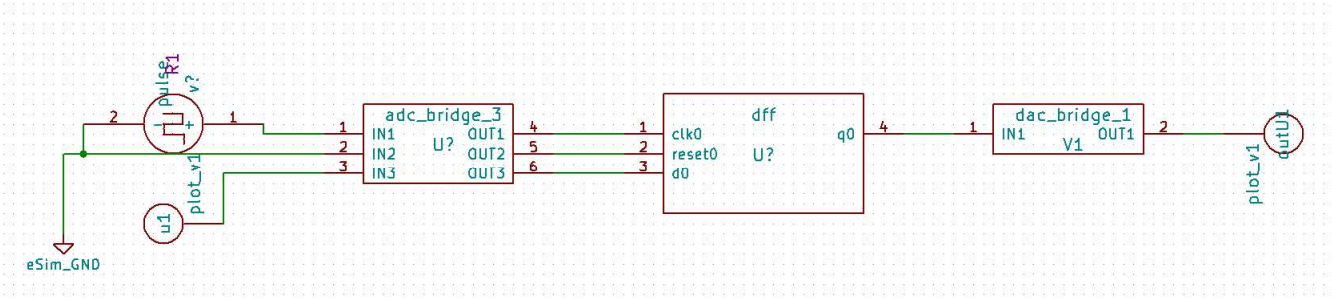

Figure 4: D Flip Flop

8 Multiplexer 2x1

8.1 About

The 2×1 is a fundamental circuit which is also known 2-to-1 multiplexer that are used to choose one signal from two inputs and transmits it to the output. The 2×1 mux has two input lines, one output line, and a single selection line. It has various applications in digital systems such as in microprocessor it is used to select between two different data sources or between two different instructions.

8.2 Verilog Code

```
module mux2_1(in0,in1,se1,out);
input se1;
input [7:0] in0;
input [7:0] in1;
output [7:0] out;
reg out;
always @(in0,in1,se1)
begin
if(se1==1'b1) begin
out =in0;
end
else
begin
out =in1;
end
endmodule
```

8.3 Schematic Diagram

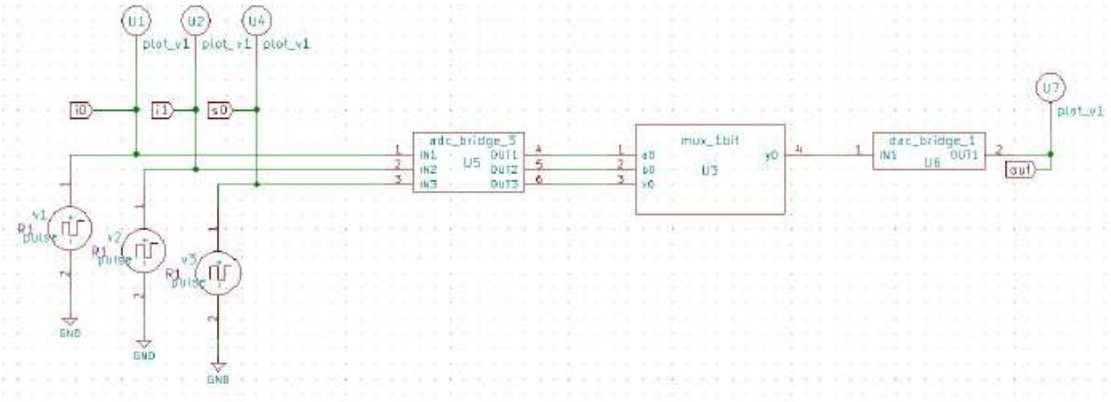

Figure 5: 2x1 Multiplexer

9 BCD to seven segment converter

9.1 About

BCD (Binary Coded Decimal) is an encoding scheme which represents each of the decimal numbers by its equivalent 4-bit binary pattern. Seven segment displays comprise of seven individual segments formed by either Light Emitting Diodes (LEDs) or Liquid Crystal Displays (LCDs) arranged in a definite pattern.For the display to work, these segments are to be driven by the certain logic level at their input.

9.2 Verilog Code

‘timescale 1ns / 1ps
module bcd2sevenseg(input [3:0] bcd, output reg [6:0] seg);
always @(bcd)
begin
case (bcd)

$$0 : seg <= 7'b1111110;$$

$$1 : seg <= 7'b0110000;$$

$$2 : seg <= 7'b1101101;$$

$$3 : seg <= 7'b1111001;$$

$$4 : seg <= 7'b0110011;$$

$$5 : seg <= 7'b1011011;$$

$$6 : seg <= 7'b1011111;$$

$$7 : seg <= 7'b1110000;$$

$$8 : seg <= 7'b1111111;$$

$$9 : seg <= 7'b1111011;$$

$$10 : seg <= 7'b1110111;$$

$$11 : seg <= 7'b0011111;$$

$$12 : seg <= 7'b1001110;$$

$$13 : seg <= 7'b0111101;$$

$$14 : seg <= 7'b1001111;$$

$$15 : seg <= 7'b1000111;$$

$$default : seg <= 7'b0000000;$$

endcase
end
endmodule

9.3 Schematic Diagram

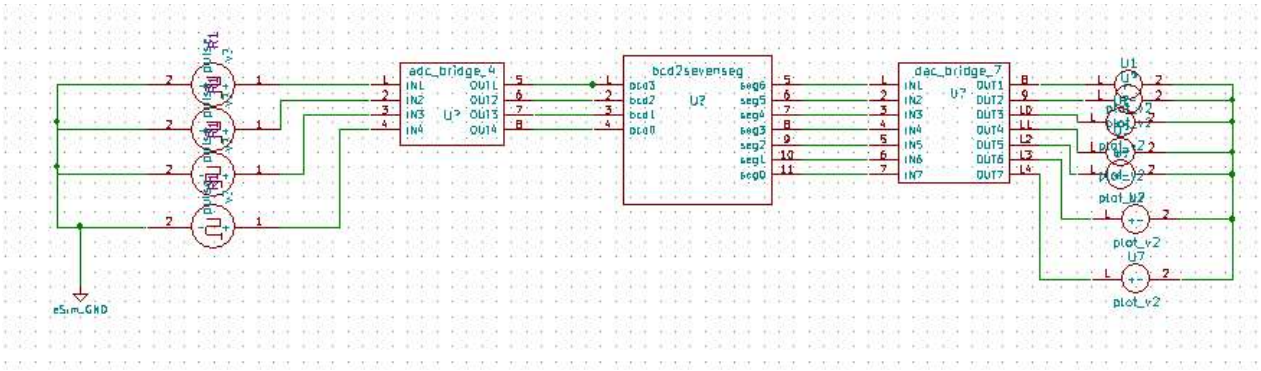

Figure 6: BCD to Seven Segment Converter

10 References

$https://www.chipverify.com/verilog/verilog-4-bit-counter$

$https://circuitfever.com/d-flip-flop-in-verilog$

$https://www.electrical4u.com/bcd-to-seven-segment-decoder/$

$https://github.com/TheSUPERCD/8bit\_MicroComputer\_Verilog/blob/master/VerilogModul$