BCD to Binary Converter

Androw Attya, Josh Biriiac, Rachelle Galan, Rami Sulaiman
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
e-mails: androwattya@oakland.edu, bethuelbiriiac@oakland.edu, rachellegalan@oakland.edu, ramisulaiman@oakland.edu

***Abstract*—A binary-coded decimal (BCD) to binary and hexadecimal converter designed, implemented and written in VHDL. With the use of an FPGA (Field Programmable Gate Array) switches and a keyboard to input the BCD number to obtain an 8-bit binary and hexadecimal number on to the seven-segment display.**

I. Introduction

The analysis of the report will cover a thorough process in designing a conversion between binary-coded decimal (BCD) to an 8-bit binary formatted and hexadecimal number operating with VHDL. The inputs will be controlled by a keyboard and three switches on the FPGA (Nexys 4) board. The outputs of the conversion will be displayed on the built-in 7 segment display, with binary operated by one switch and hexadecimal operated by another.

The binary-coded decimal (BCD) is a proficient way of encoding binary numbers. Each digit of a decimal number is represented by its binary form in 4 bits that holds the place of the decimal number it represents. As for binary code, the whole decimal number is converted into its binary form. The hexadecimal system is generally operated to portray locations in memory to signify every byte as two sequential hexadecimal digits instead of the eight digits that would be required by binary numbers. With this development, the conversion of the BCD to binary and hexadecimal would make it additionally simpler.

The report will indicate and provide the design of the project in the methodology segment, the software and hardware used to program the FPGA board will be in the experimental setup along with the digital logic design and results that were expected or were obtained in the project advancement.

II. Methodology

A. Design Overview

For this design, the user will input a binary coded decimal number into a keyboard attached to the USB port of the Nexys board. The BCD number will be displayed on the seven-segment display. The display and all necessary variables will be updated whenever the user inputs a keyboard click. Once the final 8-bit BCD value is in, two different switches may be flipped to display either the binary converted number or the hex converted number. A third switch will allow the user to reset the process. The project code may be divided into three major portions; the keyboard receiver, top module, and the seven-segment display. The overall data-path schematic is illustrated in Figure 1 below.

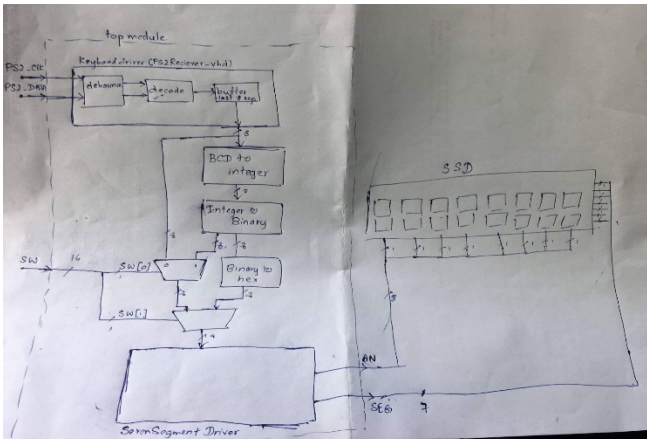

Figure 1: Data-Path

B. Keyboard Receiver

The keyboard receiver is mainly responsible for reading the digits inputted by the user and ensuring all variables are updated. During a key press, we receive 8-bits in 10 keyboard clock cycles; different key presses will send different 8-bits. The set keyboard scan codes are depicted in Figure 1. The two important codes used in this project are '1' and '0'. A key press of '1' will send a pattern of hex'16' which will be BCD'00010110'. A key press of '0' will send a pattern of hex'45' which will be BCD '01000101'.

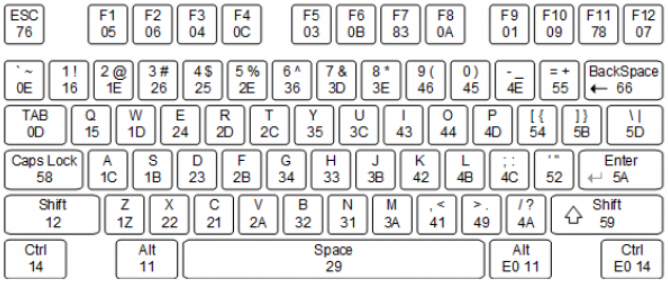

Figure 2: Keyboard Scan Codes

The keyboard driver will read the 8-bits received and check whether an appropriate key was pressed. All other keys will be disabled since the user is only inputting a BCD value. Each digit that is entered, is buffered into an 8-bit memory. Eight keys are buffered at a time because there are eight seven-

segment displays. The buffer is first in, first out nature, so older values leave first.

C. Top Module

The top module is responsible for checking the switches, converting between BCD to binary to hex, and displaying the correct number on the seven-segment display based on the switches.

Within this module, there is a combinational circuit which converts the BCD value outputted from the keyboard receiver into an integer. There is then another combinational circuit which converts this integer value to binary. With every key press, all the above variables get updated instantly. It was clear that with this implementation, there was no need for a complex state machine.

To convert from BCD to Binary, the input BCD number needs to be split by place value and converted separately. The first conversion will be of the one's place BCD number, or, the last four bits. Logically, to convert this value to binary, the number needs to be multiplied by one in BCD (0001). The second conversion is of the ten's place BCD number, or, the first four bits. This binary coded decimal number will be multiplied by ten (1010). The final step of the BCD to binary conversion is to add the two results together. The final value will be the correct binary conversion of the input.

Using the switches, the top module will select the relevant output, either BCD, Binary, or hex, and send it to the seven-segment display. The logic of this project, specifically the top module, may be further simplified in Figure 2. This flow diagram illustrates the path the BCD input will take.

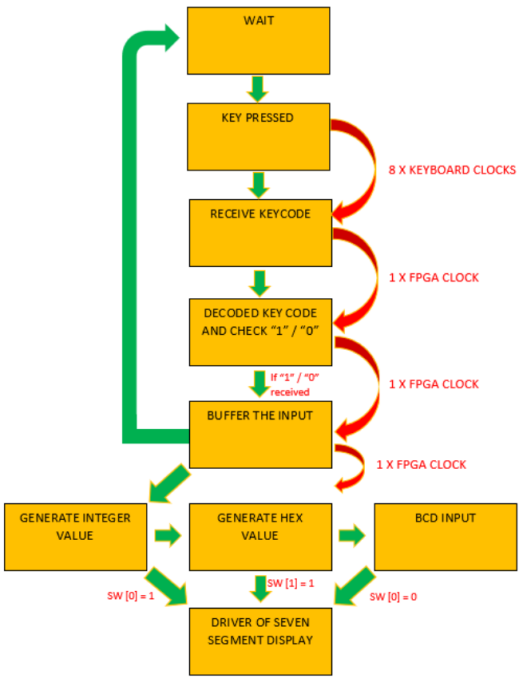

Figure 3: Logic Diagram of Project

D. Seven-Segment Display

The seven-segment driver is composed of eight seven-segment displays. In these seven-segment displays there is a common cathode bus with different anodes for each individual display. Since there is a common bus, the same pattern will be displayed on all the seven segment displays that are turned on. This is an issue considering the user will input a combination of ones and zeros and not all one value. To avoid this issue, each seven-segment display is turned on for about $1/100^{th}$ of a second, one after another. The anodes of each display are used to show the pattern relevant to that particular seven segment display based on a change in the refresh counter. To sum, only a single seven-segment display is on at a time, but since it flashes so quickly, the human eye can not pick up on the flash and will see all seven segments on at once.

III. Experimental Setup

Random numbers were picked to simulate into the testbench to analyze the outputs and the behavior of the design. A few changes were made to see the expected outputs, and the changes were some naming conventions needed to be addressed. The results came out to be as expected after a few tries. The group used NEXYS 4 DDR board to implement the design with a keyboard connected via USB. Three switches were used on the board, SW [0], SW [1], and SW [2]. The first switch is used to display the result after converting from BCD to binary. The second switch is used to display HEX, and finally the third switch is used to reset the values back to zero and back to the original state. All eight seven segment displays were used to show the input and output of the result. As the BCD number is entered from the keyboard, the '0' or '1'is shifted over to the next seven segment display to the left. This is shown in Figure 4.

When the desired BCD number is entered, one can use switches zero to two as one can wish what they want to display. The random decimal numbers were picked, is converted to BCD, then is used to enter and test on the implemented NEXYS board. Then the group tried the different switches to display the desired results. The conversions were fully functional as expected and using the experimental setup helped us verify our results. The following table shows some of the BCD numbers the group tried.

TRIAL #	DECIMAL #	INPUT (BCD)	OUTPUT (BINARY)	OUTPUT (HEX)
1	19	0001-1001	00010011	13
2	7	0000-0111	00000111	7
3	93	1001-0011	01011101	5D
4	9	0000-1001	00001001	9
5	48	0100-1000	00110000	30

Figure 4: Simulation Trials

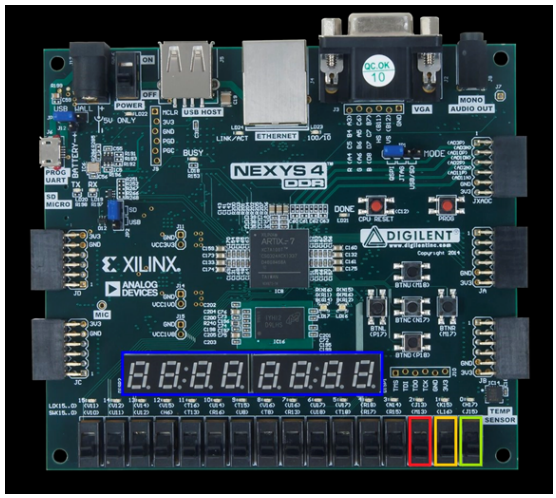

Figure 5: Nexys Board Layout

IV. RESULTS

The team was able to successfully achieve the main goal of the project which is converting BCD to binary but not only that, we decided to go the extra mile and add conversion in system that able to convert BCD to hex too. The team was able to achieve all the goals of the project by developing a code that is written by VHDL language.

Our model is an input 8 digits between [1,0] that will be inserted via keyboard, then go through the top level diagram design that we designed then the results will show in 7 seg-display, getting into depth of the project the keyboard will always provide an 8 bit binary output which indicates each seven segment display value whether it is a 1 or a 0. Followed by combinational circuit which converts BCD value into an integer, then there is another combinational circuit which converts the integer value back to binary.

The architecture of our project is a bit complex, even though we didn't use FSM which wasn't necessary for our project success but on the other side, we used several combinational circuits. We had an issue with representing the input of the 7 seg display while we pressing the numbers on keyboard from left to right, first it was flashy but then we found out that the clock needed to be synchronized in a way that it can be faster than the human eye.

Results of our project is divided into five cases that prove the success of our project, inserting an input of BCD eight digits and shows the right answer for the binary conversion, not only that we inserted all the five cases in test bench so it can appear on simulation. Looking at the simulation section, it represents the five cases that we presented in class in front of everyone that shows BCD to binary to hex results. Simulation shows several sections, the SSD binary, keypad input numbers and hex output that will be simulated on seven segment display by using switch [2].


Figure 6: Simulation Results

CONCLUSIONS

Finally the team successfully achieved all the goals of the project, first attaching a keyboard to the FPGA board that able to insert an input between 0,1. Secondly designed an accurate system that will convert from BCD to binary and then to hex and the output of eight bits of the conversion will be simulated on 7 seg-display. Our architecture will allow us to easily switch between BCD, binary and hex at any second but we are limited from 0 to 99 ,after that the design won't be able to give accurate conversion results. Completing the project gave us confident in ourselves that we are familiar with VHDL language, the use of NEXYS-4 board and become proficient in using Vivado.

REFERENCES

[1] Reference.digilentinc.com.(2019).reference:programmable-logic:nexys-4-ddr [Reference.Digilentinc]. [online] Available at: https://reference.digilentinc.com/reference/programmable-logic/nexys-4-ddr [Accessed 21 Nov. 2019].

[2] D. Llamocca, VHDL Coding for FPGAs. [Online]. Available: http://www.secs.oakland.edu/~llamocca/VHDLforFPGAs.html [Accessed 21 Nov. 2019].