BCD to Binary Converter

EGR 2700 Final Project

List of Authors (Christopher MacKenzie, Haojia Sun, Joel Fazecas, Jimmy Yousif)
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
e-mails: cmackenzie2@oakland.edu, haojiasun@oakland.edu, joelfazecas@oakland.edu, jimmyyousif@oakland.edu

***Abstract*- This project uses Digilent Nexys 50T equipment to realize the conversion from BCD code to binary with the use of a keyboard. The BCD input is realized by the input of the keyboard, and displayed on the 8 7-segment displays. Finally the conversion result will be shown on the display. This project familiarizes those designing it with the method and process of using FPGA boards for digital electronic system design and development. This project also provides experience for further research in the future. This project can be expanded to output the result to an LCD interface and could allow the use of external keyboards as BCD code input in the future.**

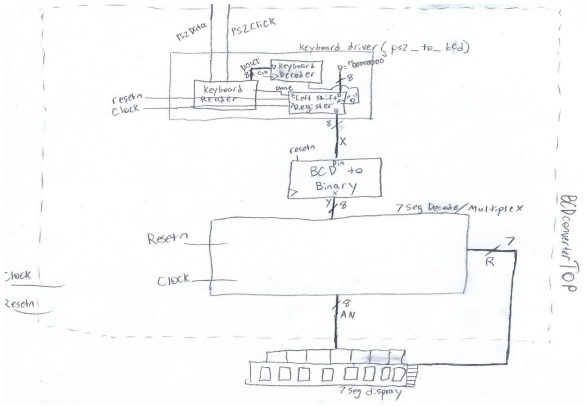
I. Introduction

This report will cover the final status of this project. The methodology section will cover a highly detailed Block diagram along with the internal components of which. The methodology section will also cover the current state of the development of VHDL code for the system.

The project is a BCD to Binary converter. The motivation was to create a project that will convert a user inputted BCD from an external device into a Binary output. The project utilizes a keyboard for the input method to generate a BCD input.

The motivation for this project is to create a simple to use physical BCD to Binary calculator. This could be useful for anyone that often does these conversions or for someone who is learning about these concepts. This project covers multiple topics covered in this course such as; the calculations and concepts used to convert these number systems between each other, and structural implementation of a system in VHDL. The topics not learned during this course that is pursued in this project is the implementation of an external usb keyboard as well as multiplexing of data to be displayed on all sections of a seven segment display.

II. METHODOLOGY

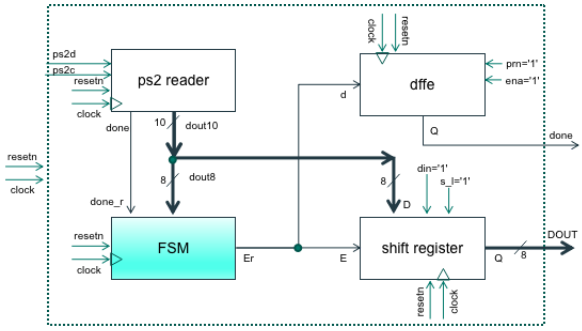


[Figure 1]

A. Keyboard Implementation

The project is represented in the Block Diagram displayed above. A user will generate an 8-bit BCD number by inputting 1's and 0's into a usb keyboard. The keyboard is connected directly to the usb port on the Nexys board. A PS/2 keyboard will output data each time a key is pressed and released. This data is represented on the diagram as PS2C and PS2D. This data consists of 11 bits. Two bits are start and stop bits, and one bit is a parity bit. The remaining 8 bits are the data. As seen in the diagram, the Keyboard driver component takes this data and converts it into BCD. This keyboard driver is composed of three major components. These components can be seen in figure 1 above. These components are the keyboard reader, the keyboard decoder, and a left shift register. The keyboard reader component is composed of 4 components itself and will be described below. The keyboard decoder takes the output from the keyboard reader (DOUT) and converts the data to a one or a zero. This is because the user inputs either a one or a zero from the keyboard. The DOUT from the keyboard reader is the stripped down 8 bits of data from the keyboard itself. This data is read by the decoder and depending on what the data is, a 1 or a 0 will be produced. This output from the decoder is inputted into the Din input of a left shift register. The enable pin of the shift register is tied to the "done" output of the keyboard reader. This allows

the register to only shift in values from the decoder when the keyboard reader is done grabbing a key press from the keyboard. The shift register outputs the current value held as "x", this x value is the current BCD input.



[Figure 2]

The last component in the keyboard driver is the keyboard reader. The keyboard driver in this project is sourced from Professor Daniel Llamocca's free VHDL coding resource website. [1] The keyboard reader, as seen above in figure 2, is composed of 4 different parts. The PS/2 reader reads the data in from the keyboard and outputs 10 bits (dout10). This output is fed into the finite state machine (FSM) as well as the shift register. The finite state machine waits for conditions to be met and outputs a signal Er, this signal acts as an enable for the shift register as well as the data input for the D-type flip flop. Other connections can be seen in the figure above and workings of these components can be explored in the VHDL code for each component. Finally, the shift register outputs its current value as DOUT. DOUT then passes through the previously explained components in the keyboard driver module, and an 8 bit BCD value is outputted (x).

B. BCD to Binary Computation

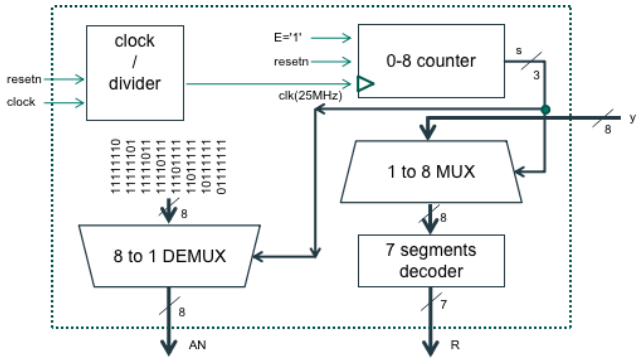
The BCD to Binary component in this project is done completely through manipulation of data through VHDL coding. This manipulation is based on a property of the conversion to binary from BCD. This property is that if the inputted BCD is broken down into two nibbles (4 bit groups), then a specific binary arithmetic operation will convert to binary. This is best visualized by observing the VHDL code itself.

```
entity bcdTObin_2 is
    Port ( Din : in STD_LOGIC_vector(7 downto 0);
           X : out std_logic_vector (7 downto 0));
end bcdTObin_2;

architecture Behavioral of bcdTObin_2 is
signal bcd1,bcd0 : std_logic_vector(7 downto 0);

signal temp5: std_logic_vector(7 downto 0);
begin
temp5<= bcd1+bcd1+bcd1+bcd1+bcd1+bcd1+bcd1+bcd1+bcd1+bcd1;
bcd1<= '0'&'0'&'0'&'0'&Din(7)&Din(6)&Din(5)&Din(4);
bcd0<= '0'&'0'&'0'&'0'&Din(3)&Din(2)&Din(1)&Din(0);
x<=temp5+bcd0;

end Behavioral;
```

[Figure 3]

Figure 3 above shows the implemented code for the BCD to binary converter. Firstly, the inputted BCD value is split into two 8 bit bytes (bcd1 and bcd0). The first 4 bits of each byte are set to 0. bcd1, which contains the first 4 bits of the original BCD input, is multiplied by 10. This new byte is stored as temp5. This new byte is then added to the second byte (bcd0). The calculated value of this is the final binary output (y). This stems off a property of the conversion from BCD to binary as explored by the website Electrical Engineering 123. [2]

C. Seven Segment Decoder/Multiplex



[Figure 4]

Figure 4 shown above shows the expanded view of the Seven Segment Decoder and Multiplexer. This module works by switching through the different anodes of the 7 segment display as well as switching through the values of each bit to be displayed. These are both done simultaneously to allow the display to show each value in its corresponding position fast enough that the human eye can perceive each display as being on concurrently.

The first component in this module is the clock divider. The clock divider is needed to slow the system clock to a usable level for this circuit. The clock divider takes in the system clock and reduces the frequency from 100 Mhz to 25 Mhz. This is done through signal manipulation with VHDL coding. The clock divider is necessary in this circuit because switching these values at

100 Mhz is too fast and reduces the on-time for each position to such a small interval, that they are not bright enough to be seen. This adjusted clock is then fed into a zero to eight counter. The counter outputs a three bit signal which has 8 possible values. The counter increases the current output by 1 for each rising clock edge. The output from this counter is then fed into the select input of the eight to one Demux and the select input of the one to eight Mux. This forces these two Muxs to switch their outputs as the counter counts. The input for the eight to one Demux is the anode values for each position on the display. The input for the one to eight Mux is the individual bits of the binary input. The most significant bit corresponds to the input anode signal of the left most display position. This switching of display positions and input values is known as serialization of a display and is explored in the website FPGA 4 student. [3]

```
begin

    with s select
        AN <= "11111110"   when "000",
              "11111101" when "001",
              "11111011" when "010",
              "11110111" when "011",
              "11101111" when "100",
              "11011111" when "101",
              "10111111" when "110",
              "01111111"    when others;
end struct;
```

[Figure 5]

```
begin

    with s select
        y2 <= y(0)    when "000",
              y(1) when "001",
              y(2) when "010",
              y(3) when "011",
              y(4) when "100",
              y(5) when "101",
              y(6) when "110",
              y(7)    when others;
end struct;
```

[Figure 6]

Figure 5 and 6 show the VHDL implementation of the 8 to 1 demultiplexer and the 1 to 8 Multiplexer respectively.

```
begin
with y2 select
R<="1001111" when '1',
   "0000001" when others;
   end behavioral;
```

[Figure 7]

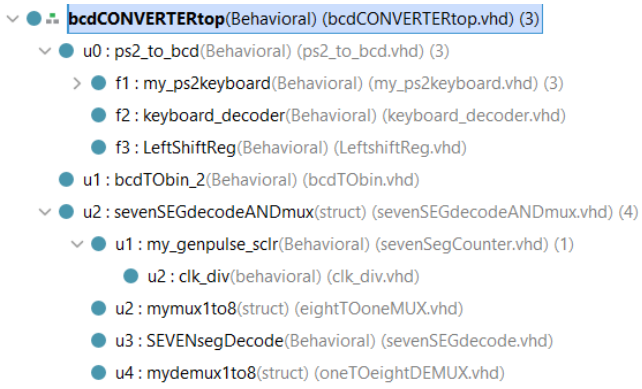
The final component within the 7 segment decoder and multiplexer is the seven segment decoder. The input for the decoder is the selected value from the 1 to 8 Mux. The decoder takes this value and sets the value of R to the corresponding positions on a seven segment display position to display either a one or a zero. The VHDL code to implement this is displayed above in figure 7.

III. Experimental Setup

The experimental setup of this system was utilized through the program Vivado 2019.2. The components needed to complete this project are a Digilent's Nexys 50T development board with USB A to B-micro cables, and a PC or laptop equipped with Vivado.

Vivado simulation was used to confirm the functionality of components that do not rely on the input from the keyboard. The BCD to binary component was tested by setting up a test bench file to input a predetermined BCD value and the binary output was observed. The seven segment decoder/multiplexer was the majority of the simulated experiments. Testing for the necessary clock speed was implemented. Testing for the switching of the multiplexers was also performed. This allowed for these components to be verified as functional.

The testing of the whole project was done with implementation on the hardware itself. This was done so the input from the keyboard could be tested. The first test that was conducted was the test to verify the correct decoding of the input from the keyboard. The values within the left shift register (x) were bound to the LEDs present on the Nexys A7 board. This gave a visual representation of how the driver was decoding inputs. The next major hardware testing setup was the simple switching of the seven segment display. After designing the seven segment decoder and multiplexer, sample values were placed into the inputs of the multiplexers. This allowed a visual representation of how the circuit would switch through anodes and if the timing of the switching was correct.



[Figure 8]

Figure 8 shows the complete list of components implemented in this project. This shows the structural design of the project as it was simulated and as it was implemented onto the board.

IV. Results

Overall this project functioned as it was initially designed to. The user of the interface inputs 8 bits of BCD through the keyboard and the circuit correctly decodes this input into binary. This binary output is displayed onto the seven segment display. Many issues were encountered and changes to the design were made to come to this result. The first major change was the addition of the keyboard interface entirely. Originally it was planned to use the switch inputs on the board as the BCD input. This had to be changed as it did not fit the requirements of the project. This caused many issues and created much more work to be done. At this point the project was reimagined and the keyboard driver was added. As this was implemented, things started to come together for the project. At this point the simulation of the computation was finalized. The first major achievement or result was the correct use of the keyboard reader program. When the correct BCD was displayed on the corresponding LEDs, major progress had been made and the project would continue smoothly from this point. The next major result in this project was the correct implementation of the seven segment multiplexer. This was a new concept for our group to overcome. The rapid switching of anodes to display more than one digit on the seven segment display was something that we had never done. Research was conducted to solve this issue. With some experimentation, the issue was solved. The final major result in this project was the combination of all the working parts into one top file. There were issues that arose when combining all the components. These issues were due to different variable names when creating each component. This was solved by mapping out the connections on a rough schematic drawing. This allowed the port mapping in the final top file to be checked. When the connections were finalized and verified to be correct, the system worked as predicted. Finally the results showed the achievements of this project.

Conclusions

After doing lots of research and running into many issues we had finally drawn out our conclusion. One of the issues that are rarely brought up is the compatible interface. The implementation of a keyboard may seem like a simple issue, but many components and setup is necessary to get this functional. Our first issue we had was using an RGB keyboard and not using a standard usb keyboard on the Nexys board. After extensive research we concluded that most keyboards that use multiple usb inputs seem to have issues. In the end converting to binary from BCD was a success when inputting from a standard usb keyboard to the Nexys board. This project was a fantastic learning experience for those involved. This project started from a base idea and progressed through all steps of developing a system. This is the first time we have done all of these activities learned in class in the correct order. Overall this project was a great success and was enjoyable to complete.

References

[1] Llamocca, Daniel. *Reconfigurable Computing Research Laboratory*, Oakland University. Accessed 16 Apr. 2021.

[2] *Electrical Engineering 123*, electricalengineering123.com/bcd-decimal-binary-binary-decimal-conversion-methods/. Accessed 16 Apr. 2021.

[3] *FPGA 4 Student*, www.fpga4student.com/2017/09/vhdl-code-for-seven-segment-display.html. Accessed 16 Apr. 2021.