Experiment 6

Procedure 1:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **BCD** | **Input** |  | **Digit** |  |  | **7-** | **Segment** | **Display** | **Signals** |  |
| **B3** | **B2** | **B1** | **B0** |  | **A** | **B** | **C** | **D** | **E** | **F** | **G** |
| 0 | 0 | 0 | 0 | **0** | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | **1** | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | **2** | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | **3** | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | **4** | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | **5** | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | **6** | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | **7** | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | **8** | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | **9** | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| - | - | - | - | **-** | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

In procedure 1 we ran into some problems with our code. First, the process statement would not work, so we decided to go with WHEN ELSE statements instead. We also made the input and display output STD\_LOGIC\_VECTORs instead of STD\_INPUT in our entity description to make the compiler happy.

Procedure 2:

8:3 Priority Encoder:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| I7 | I6 | I5 | I4 | I3 | I2 | I1 | I0 | Y2 | Y1 | Y0 | STROBE |
| 1 | - | - | - | - | - | - | - | 1 | 1 | 1 | 1 |
| 0 | 1 | - | - | - | - | - | - | 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | - | - | - | - | - | 1 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | - | - | - | - | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | - | - | - | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | - | - | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | - | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Methodology for testing: Start with all inputs at 0, then switch them to 1 in order from lowest priority to highest priority.

Questions:

1) The program takes the inputs from the switches, uses the procedure 2 implementation, then uses the output of procedure 2 as the inputs for procedure 1’s implementation.

2) By doing Procedure 1 and 2 first, we could more easily debug the programs and use a step by step process. If we just did procedure 3 then it would be a lot harder to debug all of the code at once.

3) Pros of our lab: Up to date software and faster implementation

Pros of their lab: Helps with K-maps and off-computer implementation

Cons of our lab: Requires a computer and the required software

Cons of their lab: Takes much longer to implement

4) The 0 turns off the cathode, which creates a voltage difference which turns on the LED. The 1 doesn’t create the same voltage difference so the LED does not turn on.

5) The strobe will let you know when someone breaks in, because when the sensors are tripped on the priority encoder, the strobe will turn on and the BREAK\_IN signal will become true in the Alarm FSM.

6) The advantage of the one in experiment 6 is that it is easier to visualize and debug/edit. The code in experiment 5 was longer on the computer which made it harder to edit correctly. The advantage of the experiment 5 code was that the VHDL would create a solution with less gates and signals, while the experiment 6 code would have a larger implementation.

Conclusions:

Tyler:

In this lab I learned how to use VHDL and if statements within it to create a priority encoder and a 7-segment display program. At first VHDL was hard to use because all of the statements are different from other programming languages I have used. The final product though was worth the trouble. It was interesting seeing how the VHDL code can translate into a real program that actually does something. It was also interesting combining the two in procedure 3.

Suzie:

Today in lab I used a 7- segment display program and priority encoder made using VHDL to have the board display base ten numbers. The most trouble was with the VHDL code and just figuring out the right syntax. Having the lab split up into three parts made testing a lot easier. We were able to fix a simple problem with part of the code that would have been almost impossible to find the source of if the whole thing was done at once. I enjoyed how this lab combined different things learned in this course to perform a task.