**Lab 2: Seven-Segment Display Decoder**

**Authors:**

**Documentation:**

**Purpose:** The purpose of this lab is to write, test and implement a seven-segment display decoder on the Basys3 development board. The decoder will take a four bit input value using SW3, SW2, SW1, and SW0. The right most display of the seven-segment displays will be activated by pressing BTNC. When activated, the display will show the hexadecimal value represented by the four switches.

**Prelab:** The first step in our design was to determine the Boolean logic needed to appropriately activate each of the seven segments. This was done according to the following diagram.





Each of the possible letters was evaluated to determine how the individual segments would be affected. The results were compiled in Table 1 below. The hex column is to make the test bench validation easier.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Hexadecimal Digit | Inputs | | | | Outputs | | | | | | | (in hex) |
| **D3** | **D2** | **D1** | **D0** | **Sg** | **Sf** | **Se** | **Sd** | **Sc** | **Sb** | **Sa** |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0x40 |
| 1 | 0 | 0 | 0 | 1 |  |  |  |  |  |  |  |  |
| 2 | 0 | 0 | 1 | 0 |  |  |  |  |  |  |  |  |
| 3 | 0 | 0 | 1 | 1 |  |  |  |  |  |  |  |  |
| 4 | 0 | 1 | 0 | 0 |  |  |  |  |  |  |  |  |
| 5 | 0 | 1 | 0 | 1 |  |  |  |  |  |  |  |  |
| 6 | 0 | 1 | 1 | 0 |  |  |  |  |  |  |  |  |
| 7 | 0 | 1 | 1 | 1 |  |  |  |  |  |  |  |  |
| 8 | 1 | 0 | 0 | 0 |  |  |  |  |  |  |  |  |
| 9 | 1 | 0 | 0 | 1 |  |  |  |  |  |  |  |  |
| A | 1 | 0 | 1 | 0 |  |  |  |  |  |  |  |  |
| B | 1 | 0 | 1 | 1 |  |  |  |  |  |  |  |  |
| C | 1 | 1 | 0 | 0 |  |  |  |  |  |  |  |  |
| D | 1 | 1 | 0 | 1 |  |  |  |  |  |  |  |  |
| E | 1 | 1 | 1 | 0 |  |  |  |  |  |  |  |  |
| F | 1 | 1 | 1 | 1 |  |  |  |  |  |  |  |  |

Table 1 – Showing the mapping between possible inputs and desired outputs for each segment

Utilizing Table 1 above, we were able to derive the following boolean equations for segments Sa-Sg. In order to make the implementation easier, K-maps were used to reduce the equations and can be found with the equations below

**[Insert K-maps and Boolean equations here]**

**Design:**  In order to implement this lab our final design needed to match the design seen in Figure 1.

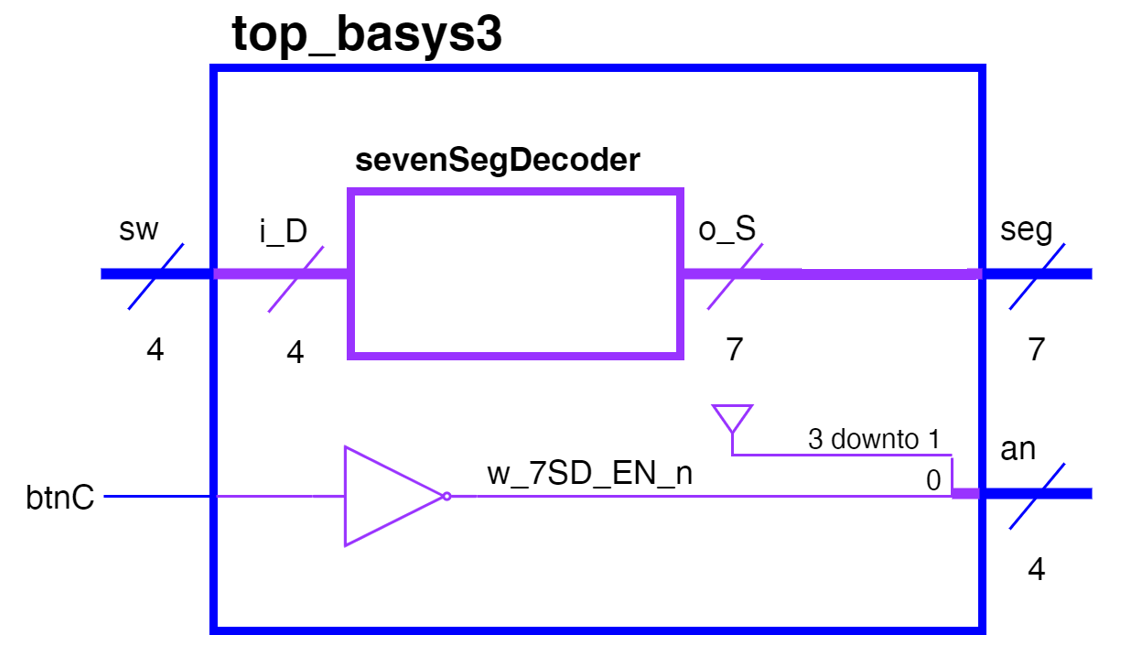


Figure 1. Top Level Schematic

Our first step in reaching this goal was to design and test the SevenSegDecoder component separately utilizing the equations Sa-Sg above and matching Figure 2.

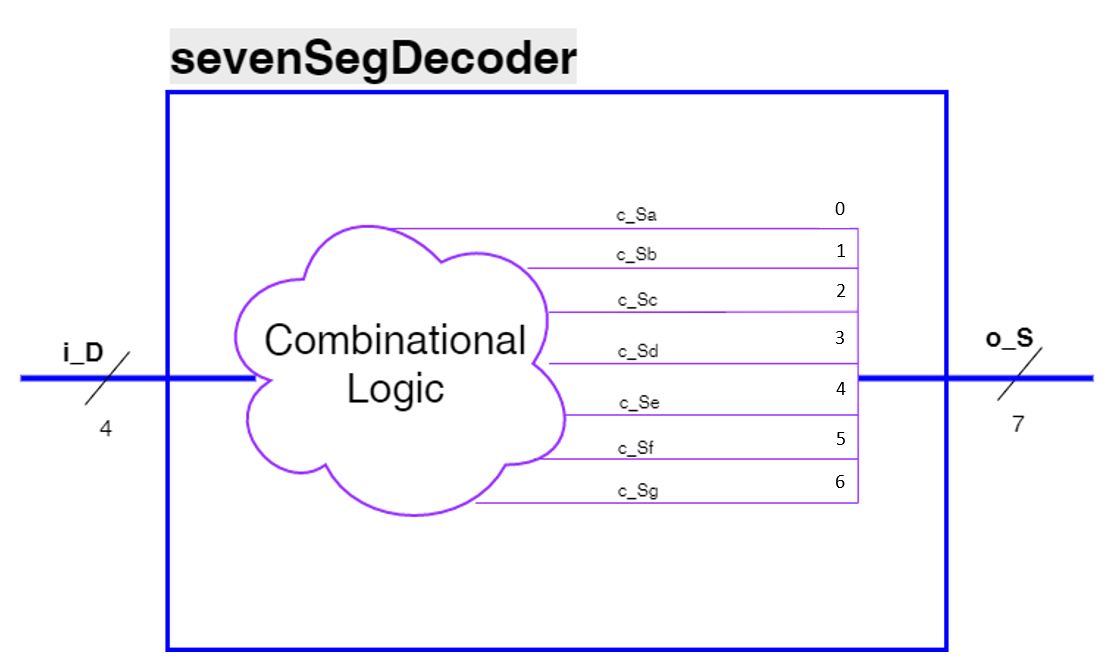
5

Figure 2. High Level Approach View of sevenSegDecoder

**RECOMMENDED APPROACH but modify to reflect how you did it** – We set up our component file utilizing the provided class template and renamed it SevenSegmentDecoder. After renaming the template we defined the entity to match Figure 1 with a four-bit i\_D input and a seven-bit o\_S output. We then defined signals for Sa-Sg to align with our prelab equations. After implemented the Sa equation, we stopped and checked for syntax errors. [Identify any issues you encountered/fixed]. We implemented the next segment utilizing a LUT instead of the equation. We found that we prefer [Discuss which method your team preferred) We repeated this process with each of the segments, checking syntax as we went.

Our next step was to design the test bench to ensure our component worked correctly. We modified the class testbench template and renamed it sevenSegmentDecoder\_tb.vhd. We declared our sevenSegmentDecoder and instantiated a version of it. We mapped the component to our artificially created input [put input name here] and output [put output name here]. We then created a test process to run through all possible inputs. Since the input is 4-bits we were able to use hex notation and ran through test cases from 0x0 to 0xF. [Describe any issues you had and how you fixed them]. Once the file was free of syntax errors we ran the simulation. The simulation waveform can be seen in Figure 3.

[Insert Waveform Here]

Figure 3. Simulation Waveform

We examined the waveform and confirmed that test cases 0x0 – 0xF were all run. For each test case, we checked the output vector’s hexadecimal values against the values we determined in Table 1. After verifying that they matched, we were confident that our component was operating normally.

The next design step was to create the top\_level design file to match Figure 1. We started with the provided top\_basys3.vhd file provided for the lab. The entity was already defined for us, so we just needed to complete the architecture. We declared our sevenSegmentDecoder and created a wire w\_7SD\_EN\_n for our button based enable. We then instantiated the component and connected i\_D to sw and o\_S to seg. [Identify any syntax errors that occurred or other debugging you accomplished]. We then connected our w\_7SD\_EN\_n to not btnC (since the an is 0 enabled) and connected an(0) to w\_7SD\_EN\_n. The remaining three bits of an were connected to ‘1’. [Identify any additional debugging accomplished]

**Final Results:** Once we finished creating our design in VHDL we looked at the RTL schematic for both the top\_basys3 design and the sevenSegmentDecoder design. Figure 4 shows top\_basys3.

[Insert Top Basys3 RTL Schematic]

Figure 4. Top\_Basys3 RTL Schematic

[Discuss the design here. This one will pretty much look as expected]

Figure 5 shows sevenSegmentDecoder.

[Insert SevenSegDecoder RTL Schematic]

Figure 5. SevenSegDecoder RTL Schematic

[Discuss the design here. This one may look different than expected]

Our final step was to generate the bitstream for our project. Because the top\_basys3 utilized standard notation for sw and seg, we only had to uncomment the respective lines in the Basys3\_Master.xdc file. [Discuss any erros/debugging here]. We pushed the bit stream to our board and successfully demonstrated all 16 possible inputs to [Insert Instructor Name here].

**Conclusions:** Discuss final thoughts here. What did you learn? Was there anything you found particularly interesting?

**Reflection:**

* **Number of hours spent on Lab2 (Combined):** \_\_\_\_\_\_\_\_
* What portion of the lab was the most difficult for you? How did you overcome it?
* What lessons, previous assignments, or activities did you find helpful is completing this lab?
* What suggestions do you have for improving Lab2 in future years? Be specific. Ex: “The instructions were confusing” does not help. What parts of the instructions were confusing