Completing this assignment takes three steps:

1. Brainstorm about what the best solution would be for the problem in the assignments and *sketch the chosen solution* (top level and sublevel functional blocks that you think you will need). Also define the **state diagram** for the (Moore) finite state machine.
2. Write the code, *verify* it with a *testbench* using ModelSim, and debug it.
3. Program the DE1-SoC board with your solution and *test* its functionality. Check the **State Machine Viewer** to see if your finite state machine is implemented as you intended.

After completing *every* step, call your instructor to verify that you are done with that step.

Assigment 5: Implementing a simple Finite State Machine

In this assignment you will implement a system consisting of the following components:

- 7-segment display driver (from the previous assignment);
- clock divider (from the previous assignment);
- nibble-counter (based on the one from the previous assignment);
- LED driver;
- finite state machine;

The LED driver is a simple component that flashes a LED with a certain frequency and duty cycle. The LED driver has three inputs: `clk`, `enable`, and `speed_select`. The `enable` input is used to enable or disable the LED driver. When the `enable` input is high and the `speed_select` input is low the LED will start flashing every second with a duty cycle of 25%. When the `enable` input is high and the speed select input is high, the led will start flashing twice as fast (every half second) with a duty cycle of 75%.

The nibble-counter should have three inputs (`clk`, `reset`, and `enable`) and two outputs (`count_done` and `count`). You can use the nibble-counter from the previous assignment as a starting point. The `enable` input is used to enable or disable the nibble-counter. When the `enable` input is low the nibble-counter should not count and hold its current value. When the `enable` input is high the nibble-counter should count. The `count_done` output should be high when the nibble-counter falls over from 9 to 0. The `count` output should be the current value of the nibble-counter. The `reset` input is used to reset the `count` and the `count_done` outputs to 0 when it is high. The nibble-counter should be connected to the 7-segment display driver and count up from 0 to 9.

The LED driver should drive LEDR0 and the 7-segment display driver should drive HEX0. The system will use three input keys (KEY0, KEY1, and KEY2).

KEY0 is used as the reset button for the system. KEY1 and KEY2 are used as inputs for the finite state machine.

Program the finite state machine to function like this:

- In the initial state of the state machine (when reset is released), the nibble-counter should be zero and disabled and the LED should be flashing slowly.
- When you press KEY1, the HEX display will start counting from 0 to 9 (every second it will count up) and the LED must be turned off.
- When you press KEY2, the counter must pause counting and the LED must start flashing slowly again until KEY1 is pressed.
- The HEX display should turn dark, and the LED should be flashing quickly when the nibble-counter is done counting.
- When you press reset everything must start over again immediately.