CSc 21100 (Spring 2021)
Optional Project
(Bonus points: 9)

IMPORTANT!

Please follow the **submission guidelines** below or your submission will be rejected.

1. You are expected to submit two files (a PDF lab report and a ZIP file containing the source files) to Blackboard. Note that both files must be upload in the same submission attempt.
2. Each task must have its own source file(s). For VHDL assignments, all source files of this assignment must be in a single Xilinx VHDL project.
3. For VHDL assignments, you must use the export function of the Xilinx ISE Design Suite to create the ZIP file properly. Please refer to the document "Exporting_VHDL_project_files" on Blockboard.
4. Naming convention:
   Report: "FirstName_LastName_Project_**XX**_CC**Y**.pdf"*
   Project: "FirstName_LastName_Project_**XX**_CC**Y**.zip"*
   *Replace "XX" and "Y" with the actual project number (two digits) and section number, respectively.
5. After the due day, all submissions are final. You cannot change it for any reasons. Double check before you make the submission.

In this project, students are expected to use the Xilinx ISE Design Suite (Webpack edition) 14.7 to complete the following tasks.

Please read the instructions carefully. Failing to follow the instructions would lead to significant point deductions.

Finite State Machine (FSM)

Write a VHDL program to implement a "combination lock" state machine that activates and "unlock" output when a certain binary input sequence is received.

Requirements:

- Similar to the example in the lecture, the FSM has one input X and two outputs UNLK and HINT;

- The UNLK output should be 1 if and only if X is 1 and the sequence of inputs received on X at the preceding three clock ticks was “101”, and the FSM returns to the INIT state;
- HINT output should be 1 if and only if the current value of X is the correct one to move the machine closer to being in the “unlocked” state (with UNLK=1);
- Whenever a wrong input is detected, the FSM returns to the INIT state and HINT=0;
- For the state memory, only two D flip-flops are allowed;

Please make sure you use the entity declaration provided below. No points would be given if failed to follow it.

```
entity combination_lock_4b is
   port ( CLK  : in    std_logic;
          X    : in    std_logic;
          HINT : out   std_logic;
          UNLK : out   std_logic);
end combination_lock_4b;
```

In this task, use the *FD* component (essentially a D flip-flop) defined in the *UNISIM* library. In order to use the *FD* component, please make sure you have the following lines in your VHDL program to include the *UNISIM* library.

```
library UNISIM;
use UNISIM.VComponents.all;
```

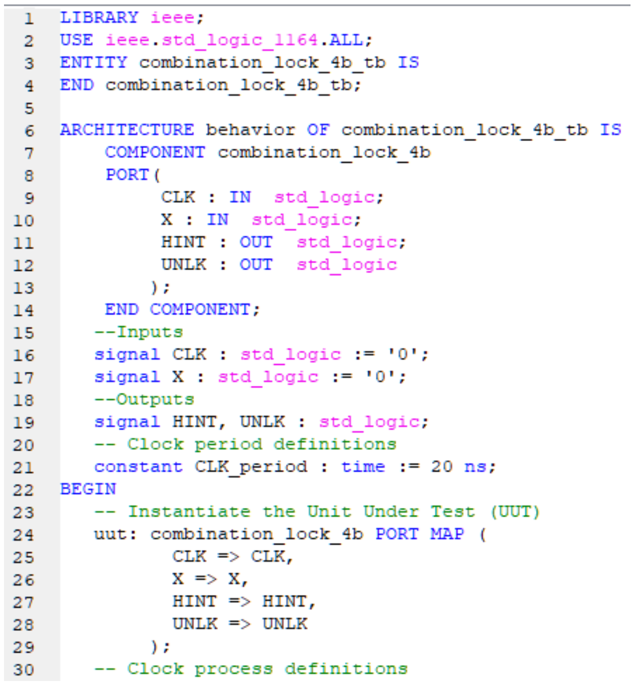
Then you will need to declare the component in the VDHL architecture definition. For example:

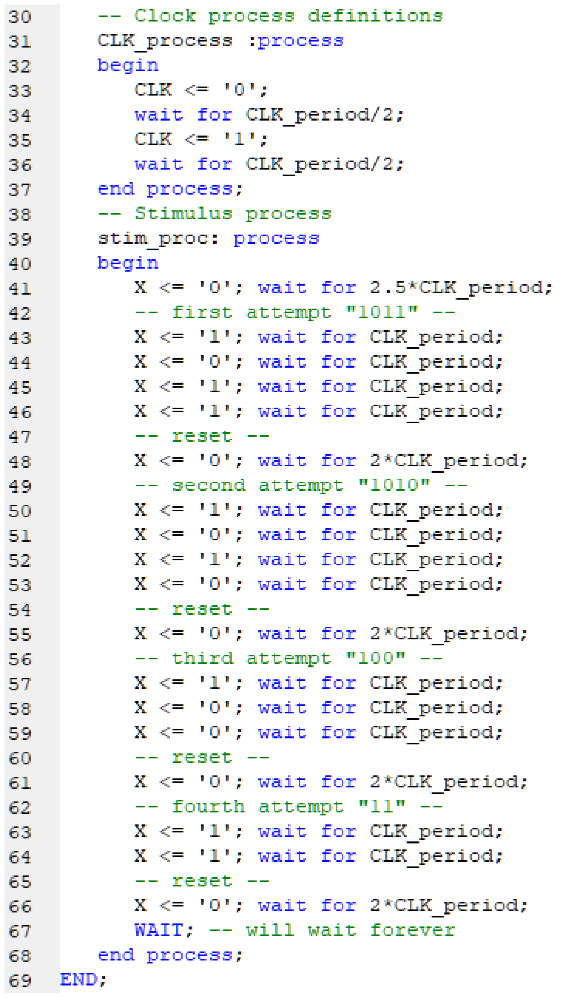
```
component FD
   generic( INIT : bit :=  '0');
   port ( C : in    std_logic;
          D : in    std_logic;
          Q : out   std_logic);
end component;
```

Then you can use it like this:

```
Your_Label : FD port map (
   C => Your_signal_1,
   D => Your_signal_1,
   Q => Your_signal_1);
```

Write a test-bench program and run simulations to validate your design. Use the given test cases in your test-bench program. Pay attention to the signal names, signal values, and the time.





```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY combination_lock_4b_tb IS
END combination_lock_4b_tb;

ARCHITECTURE behavior OF combination_lock_4b_tb IS
    COMPONENT combination_lock_4b
    PORT(
         CLK : IN  std_logic;
         X : IN  std_logic;
         HINT : OUT  std_logic;
         UNLK : OUT  std_logic
        );
    END COMPONENT;
   --Inputs
   signal CLK : std_logic := '0';
   signal X : std_logic := '0';
   --Outputs
   signal HINT, UNLK : std_logic;
   -- Clock period definitions
   constant CLK_period : time := 20 ns;
BEGIN
   -- Instantiate the Unit Under Test (UUT)
   uut: combination_lock_4b PORT MAP (
          CLK => CLK,
          X => X,
          HINT => HINT,
          UNLK => UNLK
        );
   -- Clock process definitions
   CLK_process :process
   begin
      CLK <= '0';
      wait for CLK_period/2;
      CLK <= '1';
      wait for CLK_period/2;
   end process;
   -- Stimulus process
   stim_proc: process
   begin
      X <= '0'; wait for 2.5*CLK_period;
      -- first attempt "1011" --
      X <= '1'; wait for CLK_period;
      X <= '0'; wait for CLK_period;
      X <= '1'; wait for CLK_period;
      X <= '1'; wait for CLK_period;
      -- reset --
      X <= '0'; wait for 2*CLK_period;
      -- second attempt "1010" --
      X <= '1'; wait for CLK_period;
      X <= '0'; wait for CLK_period;
      X <= '1'; wait for CLK_period;
      X <= '0'; wait for CLK_period;
      -- reset --
      X <= '0'; wait for 2*CLK_period;
      -- third attempt "100" --
      X <= '1'; wait for CLK_period;
      X <= '0'; wait for CLK_period;
      X <= '0'; wait for CLK_period;
      -- reset --
      X <= '0'; wait for 2*CLK_period;
      -- fourth attempt "11" --
      X <= '1'; wait for CLK_period;
      X <= '1'; wait for CLK_period;
      -- reset --
      X <= '0'; wait for 2*CLK_period;
      WAIT; -- will wait forever
   end process;
END;
```

Requirement(s):

(1) You must follow the structural design method.
(2) You must use the *FD* component.
(3) You must follow the submission guidelines.

Note: no points will be given if any of the requirements are not satisfied.

Deliverables: Report

1.1	Draw a circuit diagram of the module to show the design. (2 point)
1.2	Include your VHDL entity declaration(s), architecture definition(s) and the testbench program. (1 point)
1.3	Show simulation results (e.g. the waveforms). Describe the outcome of each testcase with screenshots. Explain why the simulation result is correct. (4 point)

Deliverables: Source Code

1.4	Can compile without any errors. (1 point)
1.5	Can run simulations without any errors. (1 point)