Practical Lab Exam

Problem Set 1

Problem 1

1) Write a **VHDL code** that implements the following functionality:

1.1) I/Os
- a 50 MHz system clock (**CLK**)
- an active-low asynchronous reset (**RESET_N**)
- an active-low push button (**BTN**)
- a pulse output (**PULSE**)
- a 3-digit 7-segment display output (**HEX0**, **HEX1**, **HEX2**)

1.2) Counter Operation
- It has a binary down-counter with range **999** to **0**.
- The 3-digit 7-segment display output shows the current decimal value of the counter.
- The counter decrements by 1 **every 100 ms** while reset is inactive (**RESET_N** = '1').
- If reset is active (**RESET_N** = '0'), the counter is initialized to **999**.
- In the initial state, the counter is paused. The push button can be used to start, pause, or resume the counter operation.
- If the counter is running and the push button (**BTN**) is pressed, the counter pauses. Pressing the button again resumes the counter operation.
- When the counter reaches 0, it stops and does not roll over.
- In addition, a single-cycle pulse signal (**PULSE**) is generated **every 1 ms**.

2) Create a Quartus Prime project, implement your design, and test it on the DE10-Lite FPGA board. Use the digital oscilloscope to measure the **PULSE** output.

3) Demonstrate the correct operation of your design to the lab instructor.

VHDL entity

```
entity top is
    port (
        CLK     : in  std_logic; -- 50 MHz system clock
        RESET_N : in  std_logic; -- active-low asynchronous reset
        BTN     : in  std_logic; -- active-low push button
        HEX0    : out std_logic_vector(6 downto 0); -- 7-seg digit (ones)
        HEX1    : out std_logic_vector(6 downto 0); -- 7-seg digit (tens)
        HEX2    : out std_logic_vector(6 downto 0); -- 7-seg digit (hundreds)
        PULSE   : out std_logic   -- 1 ms pulse output
    );
end top;
```

Problem 2

1) Write a **VHDL code** that implements the following functionality:

1.1) I/Os

- a 50 MHz system clock (**CLK**).
- an active-low asynchronous reset (**RESET_N**).
- an active-low push button (**BTN**).
- 10-bit LED output (**LEDS**)

1.2) LED pattern generation

- Initially, **LEDS(0)** is ON while all others are OFF.
  There is no change until the first button click (**BTN**).
- After the first button click, the active LED begins shifting to the next position at each update.
- When it reaches **LEDS(9)**, the ON position moves back down toward **LEDS(0)**.
  This creates a "running LED light pattern" with one active LED moving back and forth.
- The LED update rate should be **10 Hz**.

2) Write a **VHDL testbench** and simulate your design.

3) Create a Quartus Prime project, implement your design, and test it on the DE10-Lite FPGA board.

4) Demonstrate the correct operation of your design to the lab instructor.

LED patterns

```
0000000001   -- LEDS(0) ON
0000000010   -- LEDS(1) ON
0000000100   -- LEDS(2) ON
...
1000000000   -- LEDS(9) ON
0100000000   -- LEDS(8) ON
0010000000   -- LEDS(7) ON
...
0000000001   -- LEDS(0) ON
```

VHDL entity

```
entity top is
    port (
        CLK     : in  std_logic; -- 50 MHz system clock
        RESET_N : in  std_logic; -- active-low asynchronous reset
        BTN     : in  std_logic; -- active-low push button
        LEDS    : out std_logic_vector(9 downto 0); -- 10-bit LEDs
    );
end top;
```

Problem 3

1) Write a **VHDL code** that implements the following functionality:

1.1) I/Os
- a 50 MHz system clock (**CLK**).
- an active-low asynchronous reset (**RESET_N**).
- an active-low push button (**BTN**).
- 4-bit slide switch input (**SW**)
- a pulse output (**PULSE**)

1.2) Pulse Generation
- Initially, the **PULSE** output is LOW, and when the push button is clicked, it generates a burst of pulses.
- The number of pulses generated depends on the current value of the 4-bit slide switch input. For example, "0000" => no pulse, "1111" => 15 pulses.
- Each pulse starts with a **HIGH** interval of **1msec**, followed by a LOW interval of **2 msec**.

2) Create a Quartus Prime project, implement your design, and test it on the DE10-Lite FPGA board. Use the digital oscilloscope to measure the **PULSE** output.

3) Demonstrate the correct operation of your design to the lab instructor.

VHDL entity

```
entity top is
    port (
        CLK     : in  std_logic; -- 50 MHz system clock
        RESET_N : in  std_logic; -- active-low asynchronous reset
        BTN     : in  std_logic; -- active-low push button
        SW      : in  std_logic_vector(3 downto 0); -- 4-bit slide switches
        PULSE   : out std_logic  -- 1 ms pulse output
    );
end top;
```

Problem 4

1) Write a **VHDL code** that implements the following functionality:

1.1) I/Os
- a 50 MHz system clock (**CLK**)
- an active-low asynchronous reset (**RESET_N**)
- an active-low push button (**BTN**)
- 6-digit 7-segment display (**HEX0**,...,**HEX5**)

1.2) Moving Active Segment
- Segments of a digit are labeled 'a'–'g', starting with segment 'a' as the first one to turn ON.
- At startup, all LEDs of the six-digit 7-segment display are OFF, except for the first segment of **HEX0** (the first digit).
- No change occurs until the first push-button click.
- After the first click, the active-segment moving process begins:
- The active (ON) segment shifts to the next segment within the current digit.
- When it reaches the last segment of the current digit, it continues from the first segment of the next digit.
- When it reaches the last segment of **HEX5**, it wraps around to the first segment of **HEX0**.
- The active segment updates at a rate of **10 Hz** (i.e., 10 movements per second).

2) Create a Quartus Prime project, implement your design, and test it on the DE10-Lite FPGA board.

3) Demonstrate the correct operation of your design to the lab instructor.

```
entity top is
    port (
        CLK     : in  std_logic; -- 50 MHz system clock
        RESET_N : in  std_logic; -- active-low asynchronous reset
        BTN     : in  std_logic; -- active-low push button
        HEX0    : out std_logic_vector(6 downto 0);
        HEX1    : out std_logic_vector(6 downto 0);
        HEX2    : out std_logic_vector(6 downto 0);
        HEX3    : out std_logic_vector(6 downto 0);
        HEX4    : out std_logic_vector(6 downto 0);
        HEX5    : out std_logic_vector(6 downto 0)
    );
end top;
```