Students: Kevin (Kaiwen) Xue - 20814292, Anni Wang - 20849760

BME 393L: Prelab 4

Question-1

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;


entity one_hertz_clock is
   port(
         CLOCK_50_B5B:  in  std_logic ;     -- 50MHz clock on the board
         LEDG:       out std_logic_vector(9 downto 0)
      );
end entity one_hertz_clock;


architecture one_hertz_clock_architecture of one_hertz_clock is

signal counter: unsigned(24 downto 0);
signal output: std_logic := '0';

begin

counting: process (CLOCK_50_B5B)
begin
 if rising_edge(CLOCK_50_B5B) then
  if (counter = to_unsigned(24999999, 25)) then
   counter <= to_unsigned(0, 25);
   output <= not output;
  else
   counter <= counter + 1;
  end if;
 end if;
end process;

LEDG(0) <= output;

end architecture one_hertz_clock_architecture;
```

Figure 1: VHDL code with the entity and architecture declaration for the 1 Hz clock. Here, the output is sent to LEDG(0).

Question-2

```
-- ^^^ seven_segment entity and architecture declaration above ^^^
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity random_number_generator is port(
         CLOCK_50_B5B:  in  std_logic ;    -- 50MHz clock on the board
         KEY:     in  std_logic_vector(3 downto 0);
         HEX3:    out std_logic_vector(6 downto 0);
         HEX2:    out std_logic_vector(6 downto 0);
         HEX1:    out std_logic_vector(6 downto 0);
         HEX0:    out std_logic_vector(6 downto 0)
         );
end entity random_number_generator;

architecture random_number_generator_architecture of random_number_generator is

component seven_segment is port (
      data_in: in std_logic_vector(3 downto 0);
      blank: in std_logic;
      data_out: out std_logic_vector(6 downto 0)
);
end component;

signal rndm: unsigned(7 downto 0);
signal prev_rndm: unsigned(7 downto 0);
signal hex0_input, hex1_input, hex2_input, hex3_input: std_logic_vector(3 downto 0);
signal blank_prev_rndm: std_logic := '1';
signal blank_rndm: std_logic := '1';

begin

hex0_inst: entity work.seven_segment(seven_segment_architecture) port map (hex0_input, blank_rndm, hex0);
hex1_inst: entity work.seven_segment(seven_segment_architecture) port map (hex1_input, blank_rndm, hex1);
hex2_inst: entity work.seven_segment(seven_segment_architecture) port map (hex2_input, blank_prev_rndm, hex2);
hex3_inst: entity work.seven_segment(seven_segment_architecture) port map (hex3_input, blank_prev_rndm, hex3);

rndm <= rndm + 1 when rising_edge(CLOCK_50_B5B);

rng_process: process (KEY(0))
begin

if rising_edge(KEY(0)) then
   if blank_rndm = '0' then
      blank_prev_rndm <= '0';
      hex3_input <= std_logic_vector(prev_rndm)(7 downto 4);
      hex2_input <= std_logic_vector(prev_rndm)(3 downto 0);
   else
      blank_rndm <= '0';
   end if;
   hex1_input <= std_logic_vector(rndm)(7 downto 4);
   hex0_input <= std_logic_vector(rndm)(3 downto 0);
   prev_rndm <= rndm;

end if;


end process;

end architecture random_number_generator_architecture;
```

Figure 2: VHDL code with the entity and architecture declaration for the random number generator. Here, part of the architecture of the random number generator is implemented using a process.