Atalay Üstündağ

22203554

EE102-03

01.11.2023

Lab 5: Seven Segment Display

1.Purpose:

This experiment’s purpose is to create a 7-segment display with using decoders and encoders. Additionaly, this experiment helps us understand clock division.

2.Methodology:

The seven-segment display has a simple operating mechanism. Each section has seven LEDs. The anode and the cathode are the two inputs that control how they function. The segments are operated by the anodes, while the LEDs are operated by the cathodes. Multiple segments do not turn on or off at the same time. As in the "persistence of vision," the segments must rapidly cycle between being turned on and off.

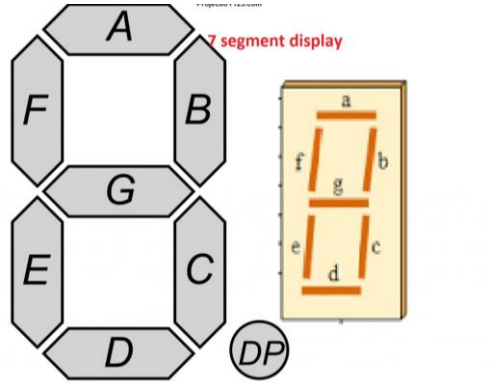

Figure 1 : LEDs on 7-segment display

Selecting the appropriate clock frequency was the first task in the experiment. The inbuilt clock of the Basys3 is 100MHz. The design's base clock was selected at this value. Then testbench and constraint file are developed after the design code process.

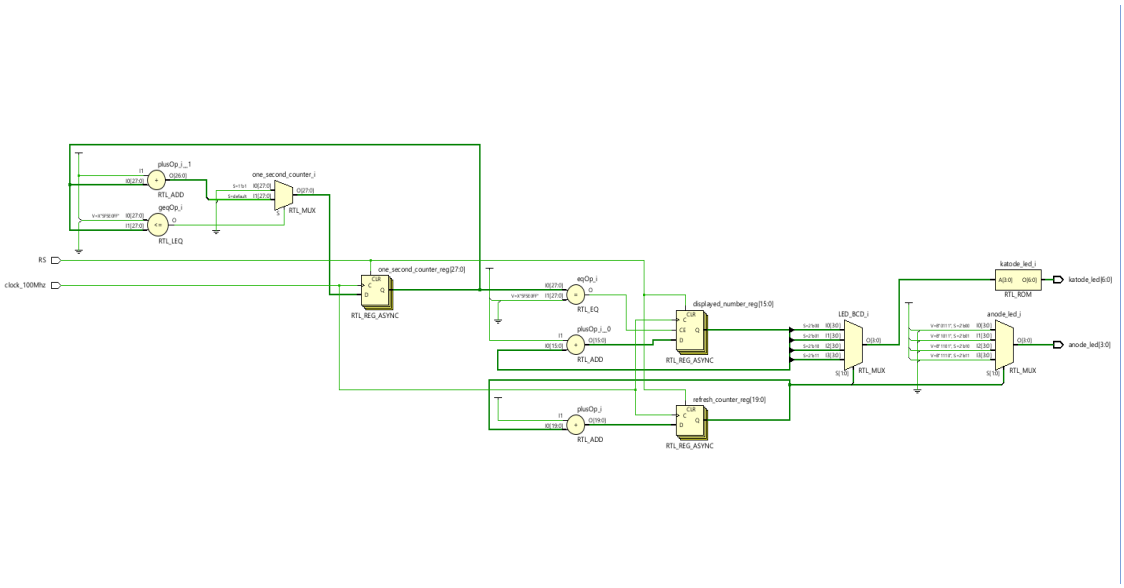

Figure 2: RTL schematic

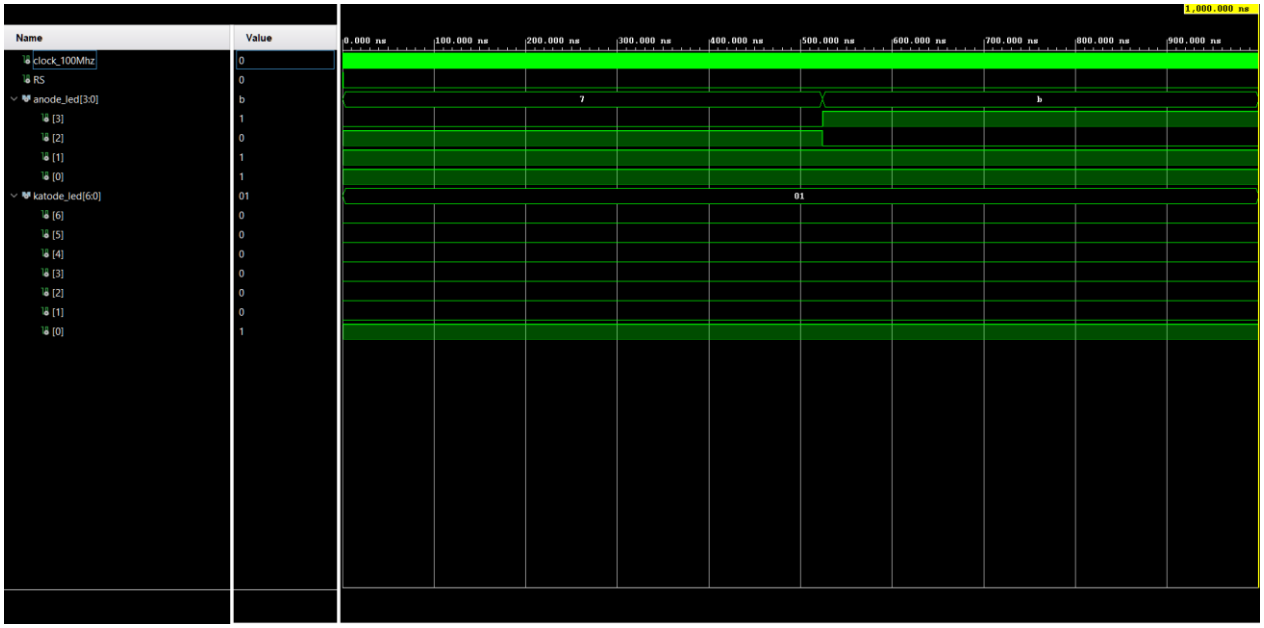

Figure 3: Simulation of the testbench

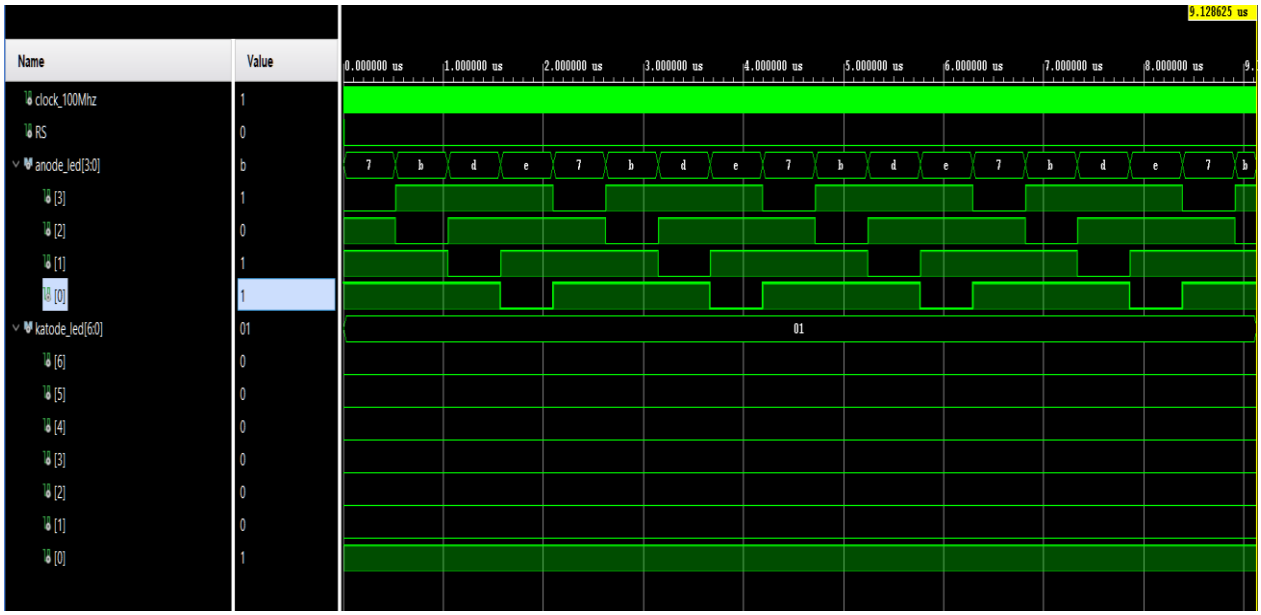

Figure 4: Simulation of the same testbench (but time is running)

Using the test bench code to execute the simulation makes it clear that the code is operating as intended. The bitstream was constructed in accordance with the specified constraints, and the proper anodes and cathodes were assigned to the assigned 7-segment display. Then BASYS 3 is programmed and the tests below are executed in the results

3.Results:

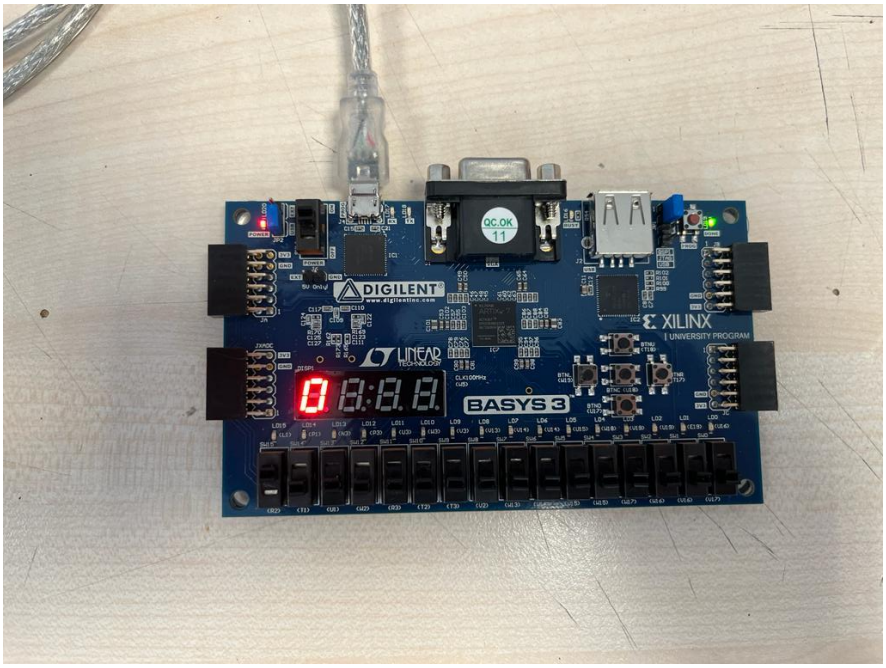

Figure 5: 0 is displayed when resetted

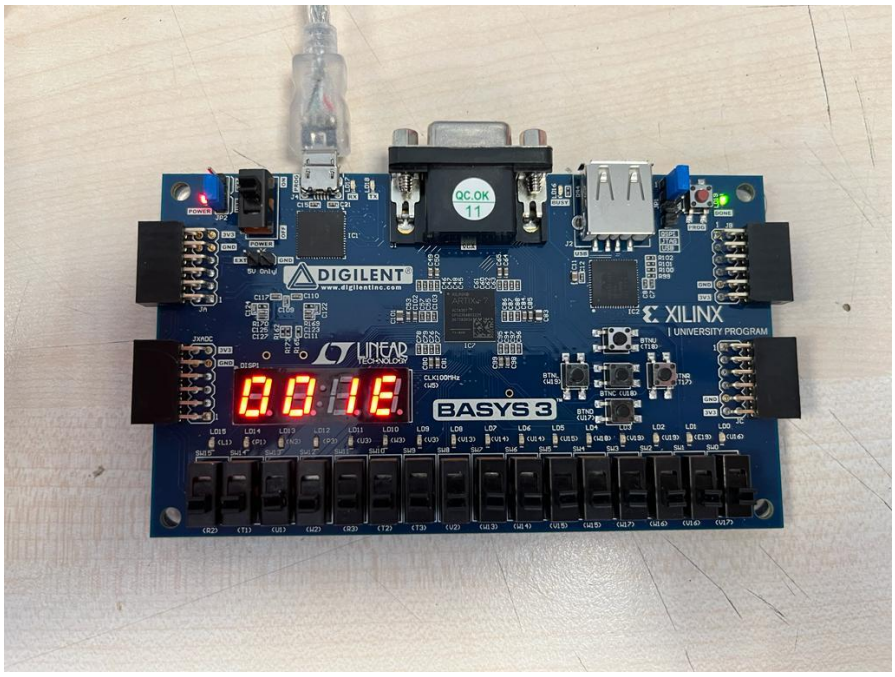

Figure 6: 30 seconds later, 435 in hexadecimal bases

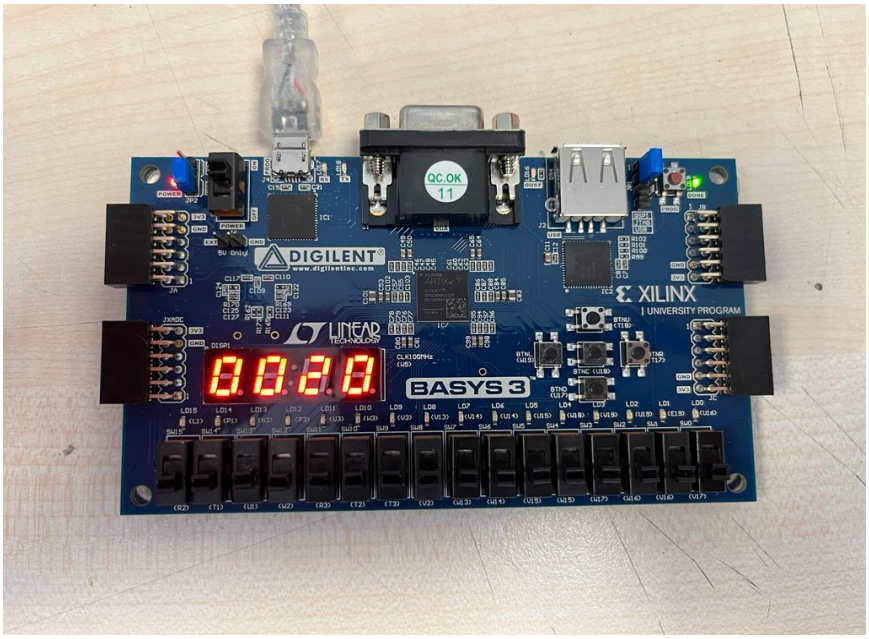

Figure 7: 32 seconds later, 435 in hexadecimal bases

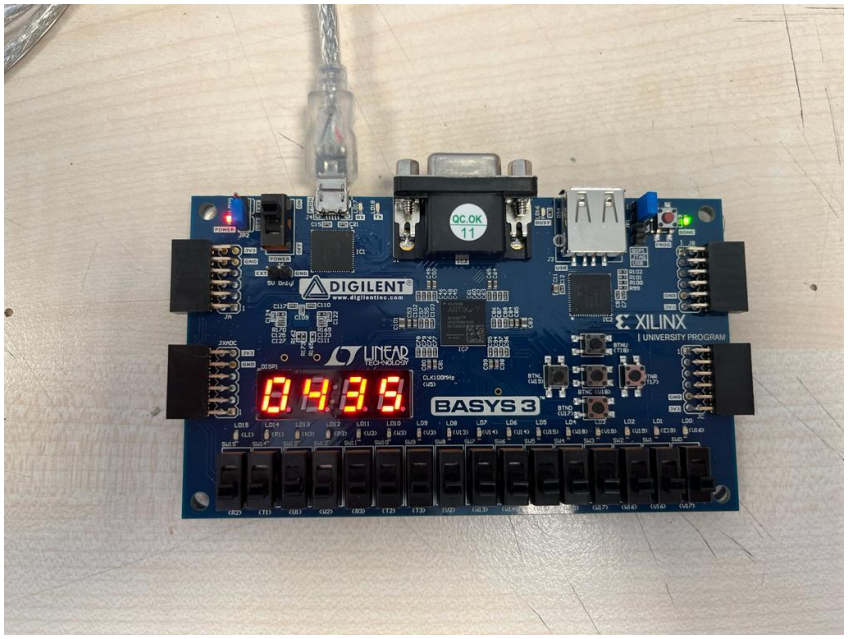

Figure 8: 1077 seconds later, 435 in hexadecimal bases

4. Conclusion and Answers to the Quesitons:

The goal of this lab was to use VHDL to create a 7 segment display with using clock and clock divider. The outcomes of the simulation and the execution matched those that were anticipated. The base clock used in the design, 100MHz, is the internal clock of the Basys 3. Still, a slower clock might have been employed. A clock divider might be utilized in this way. For example, by toggling the 50MHz clock every two ticks of the 100MHz clock, we may get a 50MHz clock from a 100MHz clock. Nevertheless, it is not feasible to generate a clock with a random value less than the Basys3 base clock. The split clock is toggled in accordance with the predetermined tick count of the base clock by the clock divider. For example, if we decide to toggle two ticks every toggle, we generate a clock frequency of 100/2 = 50 Mhz. We are unable to choose fractional ticks for each toggle number, thus the divisor needs to be an integer. As a result, we can split the 100 MHz clock into 1-100 MHz per toggle (100 MHz-1 Hz).

5.Appendix:

Design Code For Seven Segment Display

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_unsigned.all;
entity seven_segment_display_VHDL is
    Port ( clock_100Mhz : in STD_LOGIC;-- 100Mhz clock on Basys 3 FPGA board
           RS : in STD_LOGIC; -- I use this to reset.
           anode_led : out STD_LOGIC_VECTOR (3 downto 0);-- stands for the anode signals
           katode_led : out STD_LOGIC_VECTOR (6 downto 0


architecture Behavioral of seven_segment_display_VHDL is
signal first_counter : STD_LOGIC_VECTOR (27 downto 0);
signal o_s_e: std_logic;
signal show_num: STD_LOGIC_VECTOR (15 downto 0);
signal show_led: STD_LOGIC_VECTOR (3 downto 0);
signal counter_resetting : STD_LOGIC_VECTOR (19 downto 0);
signal myLEDcounter: std_logic_vector(1 downto 0);
begin



process(show_led)
begin
    case show_led is
```

```
    when "0000" => katode_led <= "0000001"; -- "stands for the number 0"
    when "0001" => katode_led <= "1001111"; -- " stands for the number 1"
    when "0010" => katode_led <= "0010010"; -- " stands for the number 2"
    when "0011" => katode_led <= "0000110"; -- " stands for the number 3"
    when "0100" => katode_led <= "1001100"; -- " stands for the number 4"
    when "0101" => katode_led <= "0100100"; -- " stands for the number 5"
    when "0110" => katode_led <= "0100000"; -- " stands for the number 6"
    when "0111" => katode_led <= "0001111"; -- " stands for the number 7"
    when "1000" => katode_led <= "0000000"; -- " stands for the number 8"
    when "1001" => katode_led <= "0000100"; -- " stands for the number 9"
    when "1010" => katode_led <= "0000010"; -- stands for the number a
    when "1011" => katode_led <= "1100000"; -- stands for the number b
    when "1100" => katode_led <= "0110001"; -- stands for the number C
    when "1101" => katode_led <= "1000010"; -- stands for the number d
    when "1110" => katode_led <= "0110000"; -- stands for the number E
    when "1111" => katode_led <= "0111000"; -- stands for the number F
    when others => katode_led<= "1111111"; --no number
    end case;
end process;
process(clock_100Mhz,RS)
begin
    if(RS='1') then
        counter_resetting <= (others => '0');
    elsif(rising_edge(clock_100Mhz)) then
        counter_resetting <= counter_resetting + 1;
```

```
    end if;
end process;
 myLEDcounter <= counter_resetting (19 downto 18);
process(myLEDcounter)
begin
    case myLEDcounter is
    when "00" =>
        anode_led <= "0111";

        show_led <= show_num(15 downto 12);

    when "01" =>
        anode_led <= "1011";
              show_led <= show_num(11 downto 8);

    when "10" =>
        anode_led <= "1101";
              show_led <= show_num(7 downto 4);

    when "11" =>
        anode_led <= "1110";

        show_led <= show_num(3 downto 0);

    when others =>
```

```
        anode_led <= "1111";
    end case;
end process;
process(clock_100Mhz, RS)
begin
        if(RS='1') then
            first_counter <= (others => '0');
        elsif(rising_edge(clock_100Mhz)) then
            if(first_counter>=x"5F5E0FF") then
                first_counter <= (others => '0');
            else
                first_counter <= first_counter + "0000001";
            end if;
        end if;
end process;
o_s_e <= '1' when first_counter =x"5F5E0FF" else '0';
process(clock_100Mhz, RS)
begin
        if(RS='1') then
            show_num <= (others => '0');
        elsif(rising_edge(clock_100Mhz)) then
             if(o_s_e='1') then
                show_num <= show_num + x"0001";
             end if;
        end if;
```

```
end process;

end Behavioral;
```

Constraint Codes:

```
# Pins for signals from the clock

set_property PACKAGE_PIN W5 [get_ports clock_100Mhz]

 set_property IOSTANDARD LVCMOS33 [get_ports clock_100Mhz]

set_property PACKAGE_PIN R2 [get_ports RS]

      set_property IOSTANDARD LVCMOS33 [get_ports RS]

#Pins for the seven segment display

      set_property PACKAGE_PIN W7 [get_ports {katode_led[6]}]

         set_property IOSTANDARD LVCMOS33 [get_ports {katode_led[6]}]

      set_property PACKAGE_PIN W6 [get_ports {katode_led[5]}]

         set_property IOSTANDARD LVCMOS33 [get_ports {katode_led[5]}]

      set_property PACKAGE_PIN U8 [get_ports {katode_led[4]}]

         set_property IOSTANDARD LVCMOS33 [get_ports {katode_led[4]}]

      set_property PACKAGE_PIN V8 [get_ports {katode_led[3]}]

         set_property IOSTANDARD LVCMOS33 [get_ports {katode_led[3]}]

      set_property PACKAGE_PIN U5 [get_ports {katode_led[2]}]

         set_property IOSTANDARD LVCMOS33 [get_ports {katode_led[2]}]

      set_property PACKAGE_PIN V5 [get_ports {katode_led[1]}]

         set_property IOSTANDARD LVCMOS33 [get_ports {katode_led[1]}]

      set_property PACKAGE_PIN U7 [get_ports {katode_led[0]}]

         set_property IOSTANDARD LVCMOS33 [get_ports {katode_led[0]}]

      set_property PACKAGE_PIN U2 [get_ports {anode_led[0]}]
```

```
    set_property IOSTANDARD LVCMOS33 [get_ports {anode_led[0]}]
  set_property PACKAGE_PIN U4 [get_ports {anode_led[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {anode_led[1]}]
  set_property PACKAGE_PIN V4 [get_ports {anode_led[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {anode_led[2]}]
  set_property PACKAGE_PIN W4 [get_ports {anode_led[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {anode_led[3]}]
```

Test Bench Codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity tb is
end tb;

architecture Behavioral of tb is

COMPONENT seven_segment_display_VHDL Port ( clock_100Mhz : in STD_LOGIC;--
100Mhz clock on Basys 3 FPGA board
    RS : in STD_LOGIC; -- this is resetting operation
    anode_led : out STD_LOGIC_VECTOR (3 downto 0);-- 4 Anode signals
    katode_led : out STD_LOGIC_VECTOR (6 downto 0));
END COMPONENT;
```

```
    signal clock_100Mhz, RS : std_logic := '0';
    signal anode_led: std_logic_vector(3 downto 0);
    signal katode_led: std_logic_vector(6 downto 0);



begin
    UUT : entity work.seven_segment_display_VHDL
        port map (
            clock_100Mhz => clock_100Mhz,
            RS => RS,
            anode_led => anode_led,
            katode_led => katode_led
        );

    -- Clock generation process
    process
    begin
            clock_100Mhz <= '0';
            wait for 1ps ;
            clock_100Mhz <= '1';
            wait for 1ps ;
    end process;
```

```
    -- Reset generation process
    process
    begin
        RS <= '1'; -- Initial reset
        wait for 10ps;
        RS <= '0'; -- Release reset
        wait;
    end process;

end Behavioral;
```