Saadet Büşra ÇAM-22201857
25.03.2024
EE102-1 Lab 2 Report

# LAB 5: SEVEN SEGMENT DISPLAY

## Purpose

The purpose of this lab is to learn simultaneously how to function numerous seven segments on FPGA basys3 board.

## Interview Questions

1. Basys3 internal clock frequency is 100MHz.
2. 100MHz is too fast to display 4 bits in 7 segment display. Hence, we slow the clock frequency by dividing each digit with 4 to refresh each cycle. We get 25MHz frequency by flipping the clock accordingly the base clock's selected refreshing cycle value. On this case, we divided with 4, acquiring 25MHz clock frequency.
3. One can acquire different clock frequencies with integer division. The fractional division is also supported, however, duty cycle is not programmable for outputs used in fractional mode.

## Design

This design converts 16 bit led switch into 4 bit hexadeximal code. Inputs are input1[3:0], input2[3:0], input3[3:0], and input4[3:0], represents 16 bit binary number, and sevensegmentdisplay[6:0] and anode_select[3:0] are outputs. Additionally, multiplexer function "mux" submodule used for selection of 16 inputs and 2 select signals into 4 bit array output signal. Output of this represents hexadecimal of binary inputs. Output of this "mux" submodule sent to sev_seg module. Sev_seg module output is sevensegmentdisplay[6:0]. Furthermore, the submodule invert_decoder has d_in[1:0] 2 bit input taken from clock_divider submodule d_out[3:0] output.

| 4 Bit Binary Number | Hexadecimal Representation |
|---|---|
| 0000 | 0 |
| 0001 | 1 |
| 0010 | 2 |
| 0011 | 3 |
| 0100 | 4 |
| 0101 | 5 |
| 0110 | 6 |
| 0111 | 7 |
| 1000 | 8 |

| | |
|---|---|
| 1001 | 9 |
| 1010 | A |
| 1011 | B |
| 1100 | C |
| 1101 | D |
| 1110 | E |
| 1111 | F |

Table 1: Binary and Hexadecimal Relations

## Methodology

I designed four submodules; Mux, Sev_Seg, clock_divider, invert_decoder, performing specific tasks accordingly their inputs.  Some of their input are output from another. Mux submodule takes its input from 16 switch of basys3. Select signals are taken from output of clock divider submodule. Sev_Seg module obtains the 4 bit array output of multiplexer function and sends corresponding hexadecimal representation to 7 segment display on basys3 screen. Clock divider submodule adjusts frequency. The basys3 has internal 100MHz frequency which is way too fast. According to persistence of vision principle, we divide frequency with clock divider into 25MHz which is understandable. Invert_decoder submodule takes the clock divider submodule's output as 2 bit array and converts it into 4-bit signal determining the anode to be used. (To choose anode, voltage level should be LOW)

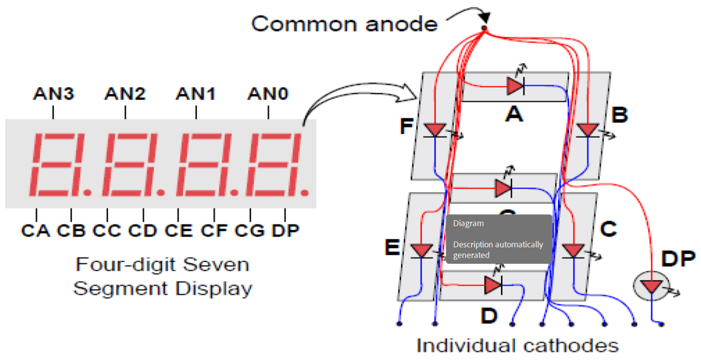

Figure-1:Seven Segment Representation

| Hex | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 2 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 4 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 5 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 6 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 9 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 10 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 11 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 12 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 13 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 14 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 15 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |

Table-2: Hexadecimal number representation on Seven Segment Display

Additionally, the rightmost switch is assigned as least significant number, and leftmost switch is most significant number. Group 1 includes least significant 4 digits and Group 4 includes most significant 4 digits.

## Results

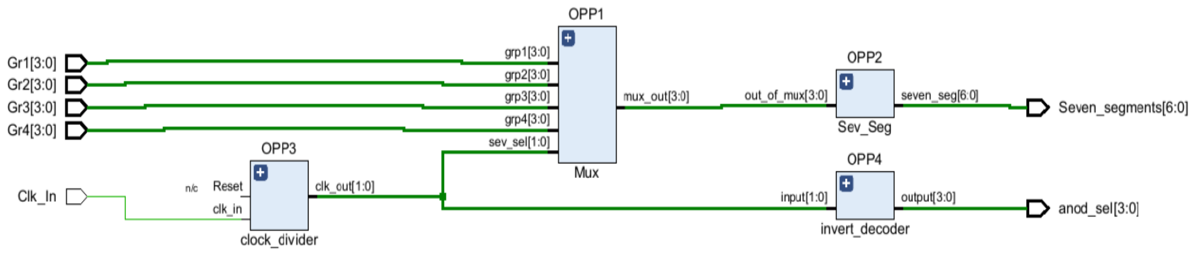

Figure-2: RTL schematic

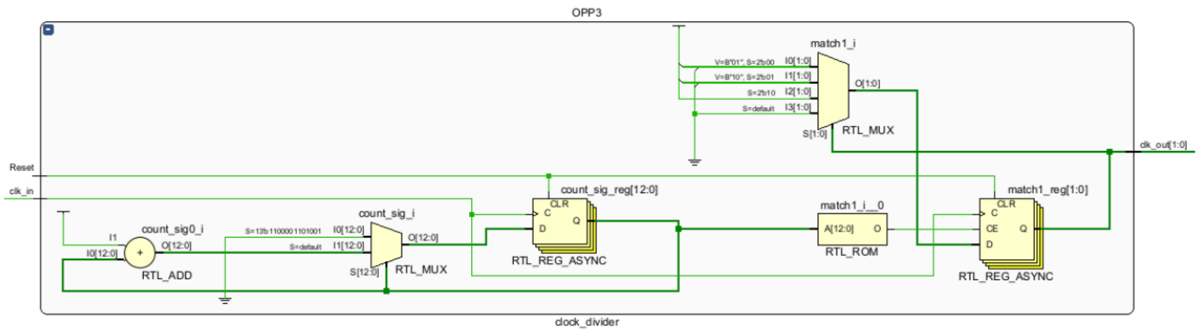

Figure-3: clock_divider RTL schematic

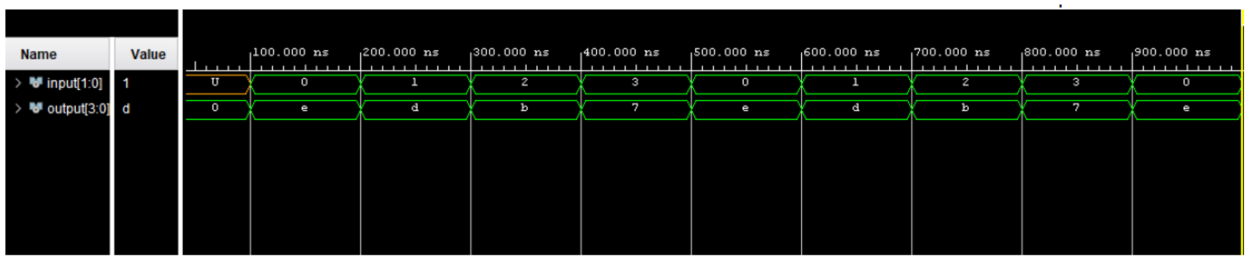

Figure-4: Anode Selection Simulation

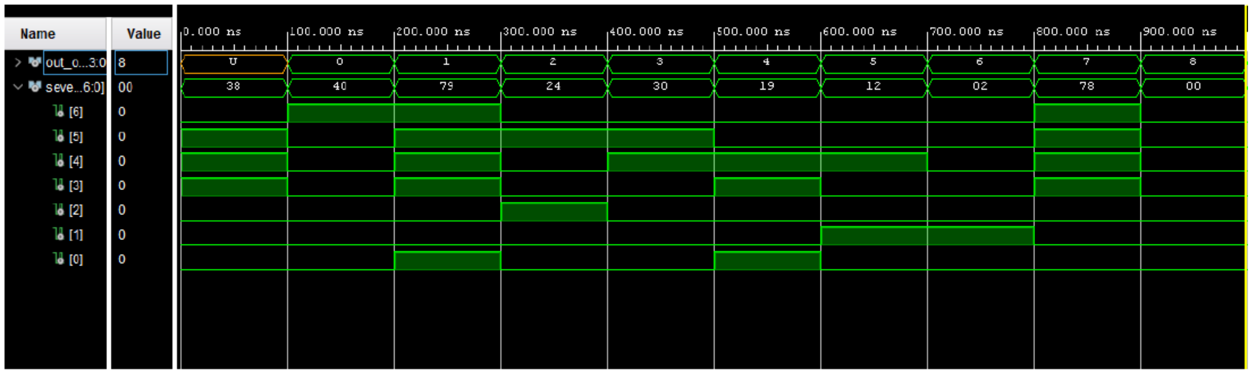

Figure-5: Seven Segment Test Bench Simulation

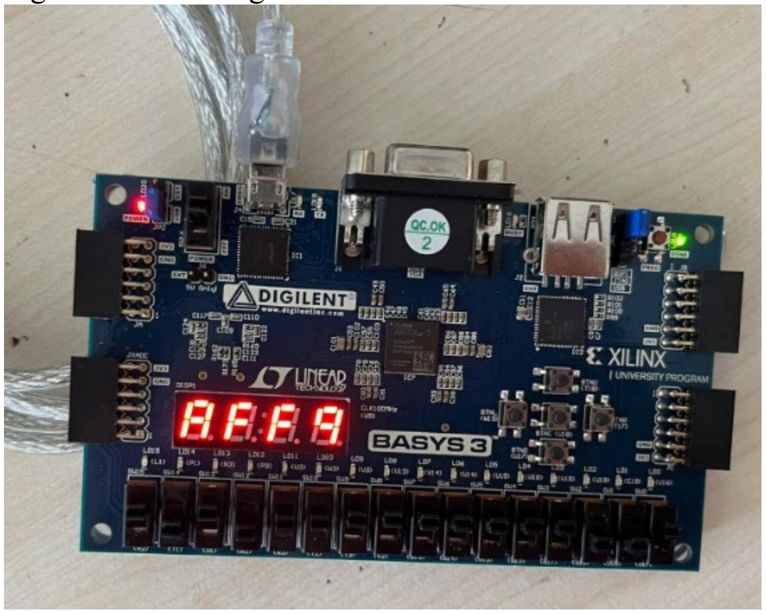
Figure-6: AFF9 displayed

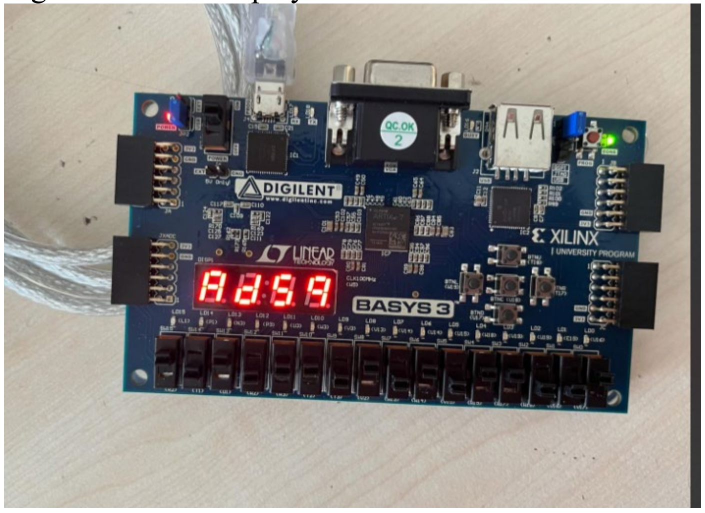
Figure-7: Ad59 displayed

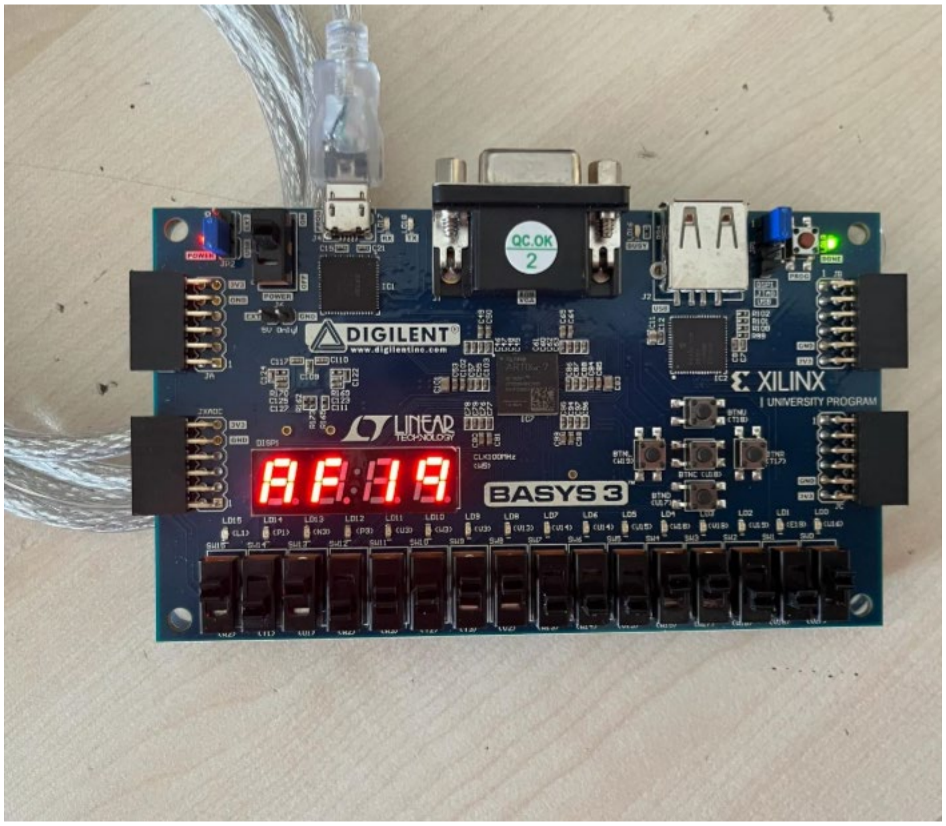
Figure-8: AF19 displayed

## Conclusion

In conclusion, on this lab, I gained knowledge about displaying 16 bit input in seven segment display screen. Additionally, I learned about the terms of clock frequency, clock, and internal frequency of basys3. The experiment was successful since my results matched with table 1 and table 2.

## References

Wikimedia Foundation. (2024, February 4). *Persistence of vision*. Wikipedia. https://en.wikipedia.org/wiki/Persistence_of_vision

*[FPGA tutorial] seven-segment LED display on Basys 3 FPGA*. FPGA Projects, Verilog Projects, VHDL Projects - FPGA4student.com. (n.d.). https://www.fpga4student.com/2017/09/seven-segment-led-display-controller-basys3-fpga.html

## Appendices

**Main_sevensegment.vhd**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity main_module is
Port ( input1 : in STD_LOGIC_VECTOR (3 downto 0);
Input2 : in STD_LOGIC_VECTOR (3 downto 0);
Input3 : in STD_LOGIC_VECTOR (3 downto 0);
Input4: in STD_LOGIC_VECTOR (3 downto 0);
Clk_In : in STD_LOGIC;
Seven_segments : out STD_LOGIC_VECTOR (6 downto 0);
anod_select : out STD_LOGIC_VECTOR (3 downto 0));
end main_module;
architecture Behavioral of main_module is
COMPONENT Mux
PORT ( group1 : in STD_LOGIC_VECTOR (3 downto 0);
group2 : in STD_LOGIC_VECTOR (3 downto 0);
group3 : in STD_LOGIC_VECTOR (3 downto 0);
group4 : in STD_LOGIC_VECTOR (3 downto 0);
seven_select : in STD_LOGIC_VECTOR (1 downto 0);
mux_output : out STD_LOGIC_VECTOR (3 downto 0));
END COMPONENT;
COMPONENT Sev_Seg
PORT ( muxoutput: in STD_LOGIC_VECTOR (3 downto 0);
segmentdisplay: out STD_LOGIC_VECTOR (6 downto 0));
END COMPONENT;
COMPONENT clock_divider
PORT ( clock_in : in STD_LOGIC;
Reset : in STD_LOGIC;
clock_out : out STD_LOGIC_VECTOR (1 downto 0));
END COMPONENT;
COMPONENT invert_decoder
PORT ( input : in STD_LOGIC_VECTOR (1 downto 0);
output : out STD_LOGIC_VECTOR (3 downto 0));
END COMPONENT;
SIGNAL temp_dig : STD_LOGIC_VECTOR (3 DOWNTO 0);
SIGNAL temp_reset : STD_LOGIC;
SIGNAL temp_clock : STD_LOGIC_VECTOR (1 DOWNTO 0);
begin
OPP1 : Mux
PORT MAP (group1 (3 DOWNTO 0) => input1 (3 DOWNTO 0),
group2 (3 DOWNTO 0) => input2 (3 DOWNTO 0),
group3 (3 DOWNTO 0) => input3 (3 DOWNTO 0),
group4 (3 DOWNTO 0) => input4 (3 DOWNTO 0),
sevenselect (1 DOWNTO 0) => temp_clock (1 DOWNTO 0),
muxout (3 DOWNTO 0) => temp_dig (3 DOWNTO 0));
OPP2 : Sev_Seg
PORT MAP (muxoutput (3 DOWNTO 0) => temp_dig (3 DOWNTO 0),
segmentdisplay (6 DOWNTO 0) => Seven_segments (6 DOWNTO 0));
OPP3 : clock_divider
PORT MAP (clock_in => Clock_in,
Reset => temp_reset,
clock_output (1 DOWNTO 0) => temp_clock (1 DOWNTO 0));
OPP4 : invert_decoder
PORT MAP (input (1 DOWNTO 0) => temp_clock (1 DOWNTO 0),
output (3 DOWNTO 0) => anod_select (3 DOWNTO 0));
end Behavioral;
```

**sev_seg.vhd**

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity Sev_Seg is
Port ( muxoutput : in STD_LOGIC_VECTOR (3 downto 0);
segmentdisplay: out STD_LOGIC_VECTOR (6 downto 0));
end Sev_Seg;
architecture Behavioral of Sev_Seg is
begin
PROCESS(muxoutput)
BEGIN
CASE out_of_mux IS
WHEN "0000" => segmentdisplay <= "1000000"; -- 0
WHEN "0001" => segmentdisplay <= "1111001"; -- 1
WHEN "0010" => segmentdisplay <= "0100100"; -- 2
WHEN "0011" => segmentdisplay <= "0110000"; -- 3
WHEN "0100" => segmentdisplay <= "0011001"; -- 4
WHEN "0101" => segmentdisplay <= "0010010"; -- 5
WHEN "0110" => segmentdisplay <= "0000010"; -- 6
WHEN "0111" => segmentdisplay <= "1111000"; -- 7
WHEN "1000" => segmentdisplay <= "0000000"; -- 8
WHEN "1001" => segmentdisplay <= "0011000"; -- 9
WHEN "1010" => segmentdisplay <= "0001000"; -- 10 -> A
WHEN "1011" => segmentdisplay <= "0000011"; -- 11 -> b
WHEN "1100" => segmentdisplay <= "1000110"; -- 12 -> C
WHEN "1101" => segmentdisplay <= "0100001"; -- 13 -> d
WHEN "1110" => segmentdisplay <= "0000110"; -- 14 -> E
WHEN others =>  segmentdisplay <= "0001110"; -- 15 -> F
END CASE;
END PROCESS;
```

```
end Behavioral;
```

**clockdivider.vhd**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity clock_divider is
Port ( clock_in : in STD_LOGIC;
Reset : in STD_LOGIC;
clock_out : out STD_LOGIC_VECTOR (1 downto 0));
end clock_divider;
architecture Behavioral of clock_divider is
SIGNAL eq1 : STD_LOGIC_VECTOR (1 DOWNTO 0);
SIGNAL c1 : INTEGER RANGE 0 TO 6249 :=0;
begin
PROCESS (Reset, clk_in)
BEGIN
IF( Reset = '1') THEN
Eq1<= "00";
C1 <= 0;
ELSIF RISING_EDGE(clk_in) THEN
IF(count_sig = 6249) THEN
CASE eq1 IS
WHEN "00" => match1 <= "01";
WHEN "01" => match1 <= "10";
WHEN "10" => match1 <= "11";
WHEN others => match1 <= "00";
END CASE;
C1 <= 0;
ELSE
C1 <= c1 + 1;
END IF;
END IF;
END PROCESS;
clock_out <= eq1;
end Behavioral;
```

**invert_decoder**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity invert_decoder is
Port ( d_in : in STD_LOGIC_VECTOR (1 downto 0);
d_out : out STD_LOGIC_VECTOR (3 downto 0));
end invert_decoder;
architecture Behavioral of invert_decoder is
begin
PROCESS(d_in)
BEGIN
CASE d_in IS
WHEN "00" => d_out <= "1110";
WHEN "01" => d_out <= "1101";
WHEN "10" => d_out <= "1011";
WHEN "11" => d_out <= "0111";
WHEN others => d_out <= "0000";
END CASE;
END PROCESS;
end Behavioral;
```

**mux.vhd**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity Mux is
Port ( group1 : in STD_LOGIC_VECTOR (3 downto 0);
group2 : in STD_LOGIC_VECTOR (3 downto 0);
group3 : in STD_LOGIC_VECTOR (3 downto 0);
group4 : in STD_LOGIC_VECTOR (3 downto 0);
sev_sel : in STD_LOGIC_VECTOR (1 downto 0);
mux_out : out STD_LOGIC_VECTOR (3 downto 0));
end Mux;
architecture Behavioral of Mux is
SIGNAL eq1 : STD_LOGIC_VECTOR (3 DOWNTO 0);
begin
PROCESS (group1, group2, group3, group4, sev_sel)
BEGIN
CASE sev_sel IS
WHEN "00" => eq1 <= grp1;
WHEN "01" => eq1 <= grp2;
WHEN "10" => eq1" <= grp3;
WHEN others => eq1 <= grp4;
END CASE;
mux_out <= eq1;
END PROCESS;
end Behavioral;
```

**TESTBENCHES**

**anodselect.vhd:**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity anodselect is
end anodselect;
architecture Behavioral of anodselect is
COMPONENT invert_decoder
PORT ( anodinput : in STD_LOGIC_VECTOR (1 downto 0);
anodoutput : out STD_LOGIC_VECTOR (3 downto 0));
END COMPONENT;
SIGNAL anodinput : STD_LOGIC_VECTOR (1 DOWNTO 0);
SIGNAL anodoutput: STD_LOGIC_VECTOR (3 DOWNTO 0);
begin
UUT: invert_decoder
PORT MAP (anodinput (1 DOWNTO 0) => anodinput (1 DOWNTO 0),
anodoutput (3 DOWNTO 0) => anodoutput (3 DOWNTO 0));
```

```
anodselect : PROCESS
BEGIN
wait for 100 ns;
input <= "00";
wait for 100 ns;
input <= "01";
wait for 100 ns;
input <= "10";
wait for 100 ns;
input <= "11";
END PROCESS;
end Behavioral;
```

**sevensegmenttest.vhd**

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity sevensegmenttest is
end sevensegmenttest;
architecture Behavioral of seven_test is
COMPONENT Sev_Seg
PORT ( muxoutput : in STD_LOGIC_VECTOR (3 downto 0);
seven_seg : out STD_LOGIC_VECTOR (6 downto 0));
END COMPONENT;
SIGNAL muxoutput : STD_LOGIC_VECTOR (3 downto 0);
SIGNAL seven_seg : STD_LOGIC_VECTOR (6 downto 0);
begin
UUT: Sev_Seg
PORT MAP (muxoutput (3 downto 0) => muxoutput (3 downto 0),
seven_seg (6 downto 0) => seven_seg(6 downto 0));
seven_test : PROCESS
BEGIN
wait for 100 ns;
muxoutput <= "0000";
wait for 100 ns;
muxoutput <= "0001";
wait for 100 ns;
muxoutput <= "0010";
wait for 100 ns;
muxoutput <= "0011";
wait for 100 ns;
muxoutput <= "0100";
wait for 100 ns;
muxoutput <= "0101";
wait for 100 ns;
muxoutput <= "0110";
wait for 100 ns;
muxoutput <= "0111";
wait for 100 ns;
muxoutput <= "1000";
```

```
wait for 100 ns;
muxoutput <= "1001";
wait for 100 ns;
muxoutput <= "1010";
wait for 100 ns;
muxoutput <= "1011";
wait for 100 ns;
muxoutput <= "1100";
wait for 100 ns;
muxoutput <= "1101";
wait for 100 ns;
muxoutput <= "1110";
wait for 100 ns;
muxoutput <= "1111";
END PROCESS;
end Behavioral;
```