Practical No.06: Keypad Interfacing

Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
entity tb_keypad is
 Port ( clk : in STD_LOGIC; -- 100 MHz clock
 row : in STD_LOGIC_VECTOR (3 downto 0); -- Rows of keypad (inputs)
 col : out STD_LOGIC_VECTOR (3 downto 0); -- Columns of keypad (outputs)
 seg : out STD_LOGIC_VECTOR (6 downto 0); -- Seven-segment cathodes
 an : out STD_LOGIC_VECTOR (3 downto 0)); -- Seven-segment anodes
end tb_keypad;
architecture Behavioral of tb_keypad is
 signal col_sel : std_logic_vector(1 downto 0) := "00";
 signal display_digit : std_logic_vector(3 downto 0) := "1111"; -- Latched display
 signal counter: unsigned(28 downto 0) := (others => '0');
 signal clk_div : std_logic := '0';
begin
 an <= "1110"; -- Enable rightmost digit
 -- Clock divider process
 process(clk)
 begin
 if rising_edge(clk) then
```

```
counter <= counter + 1;
if counter = 19 then -- This value determines the scanning speed.
clk_div <= not clk_div;
counter <= (others => '0');
end if;
end if;
end process;
-- Drive columns based on col_sel
with col_sel select
col <= "0111" when "00",
"1011" when "01",
"1101" when "10",
"1110" when "11",
"1111" when others;
-- Keypad scanning and display latching process
process(clk_div)
begin
if rising_edge(clk_div) then
case col_sel is
when "00" =>
if row(3) = '0' then display_digit <= "0001"; -- 1
elsif row(2) = '0' then display_digit <= "0100"; -- 4
elsif row(1) = '0' then display_digit <= "0111"; -- 7
elsif row(0) = '0' then display_digit <= "0000"; -- 0
```

```
 end if;
 when "01" =>
 if row(3) = '0' then display_digit <= "0010"; -- 2
 elsif row(2) = '0' then display_digit <= "0101"; -- 5
 elsif row(1) = '0' then display_digit <= "1000"; -- 8
 elsif row(0) = '0' then display_digit <= "1111"; -- F
 end if;
 when "10" =>
 if row(3) = '0' then display_digit <= "0011"; -- 3
 elsif row(2) = '0' then display_digit <= "0110"; -- 6
 elsif row(1) = '0' then display_digit <= "1001"; -- 9
 elsif row(0) = '0' then display_digit <= "1110"; -- E
 end if;
 when "11" =>
 if row(3) = '0' then display_digit <= "1010"; -- A
 elsif row(2) = '0' then display_digit <= "1011"; -- B
 elsif row(1) = '0' then display_digit <= "1100"; -- C
 elsif row(0) = '0' then display_digit <= "1101"; -- D
 end if;
 when others =>
 null;
 end case;
 col_sel <= std_logic_vector(unsigned(col_sel) + 1); -- Cycle col_sel to scan all
columns
 end if;
 end process;
-- Seven-segment decoder
 process(display_digit)
 begin
 case display_digit is
 when "0000" => seg <= "1000000"; -- 0
 when "0001" => seg <= "1111001"; -- 1
 when "0010" => seg <= "0100100"; -- 2
 when "0011" => seg <= "0110000"; -- 3
 when "0100" => seg <= "0011001"; -- 4
 when "0101" => seg <= "0010010"; -- 5
 when "0110" => seg <= "0000010"; -- 6
 when "0111" => seg <= "1111000"; -- 7
 when "1000" => seg <= "0000000"; -- 8
 when "1001" => seg <= "0010000"; -- 9
 when "1010" => seg <= "0001000"; -- A
 when "1011" => seg <= "0000011"; -- B
 when "1100" => seg <= "1000110"; -- C
 when "1101" => seg <= "0100001"; -- D
 when "1110" => seg <= "0000110"; -- E
 when "1111" => seg <= "0001110"; -- F
 when others => seg <= "1111111"; -- Blank
 end case;
 end process;
 end Behavioral;
```

.xdc

```
set_property PACKAGE_PIN W4 [get_ports {an[3]}]
set_property PACKAGE_PIN V4 [get_ports {an[2]}]
set_property PACKAGE_PIN U4 [get_ports {an[1]}]
set_property PACKAGE_PIN U2 [get_ports {an[0]}]
set_property PACKAGE_PIN G3 [get_ports {row[3]}]
set_property PACKAGE_PIN H2 [get_ports {row[2]}]
set_property PACKAGE_PIN K2 [get_ports {row[1]}]
set_property PACKAGE_PIN H1 [get_ports {row[0]}]
set_property PACKAGE_PIN G2 [get_ports {col[3]}]
set_property PACKAGE_PIN J2 [get_ports {col[2]}]
set_property PACKAGE_PIN L2 [get_ports {col[1]}]
set_property PACKAGE_PIN J1 [get_ports {col[0]}]
set_property PACKAGE_PIN U7 [get_ports {seg[6]}]
set_property PACKAGE_PIN V5 [get_ports {seg[5]}]
set_property PACKAGE_PIN U5 [get_ports {seg[4]}]
set_property PACKAGE_PIN V8 [get_ports {seg[3]}]
set_property PACKAGE_PIN U8 [get_ports {seg[2]}]
set_property PACKAGE_PIN W6 [get_ports {seg[1]}]
set_property PACKAGE_PIN W7 [get_ports {seg[0]}]
set_property PACKAGE_PIN W5 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {seg[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {col[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {col[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {col[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {col[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {row[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {row[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {row[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {row[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {an[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {an[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {an[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {an[0]}]
```