Project Task C: Report

Keyboard Controller

EVANS, TANNER J.
ECE238L – Section 013 (Mondays 11:00 AM – 1:45 PM)
11/27/19

Contents

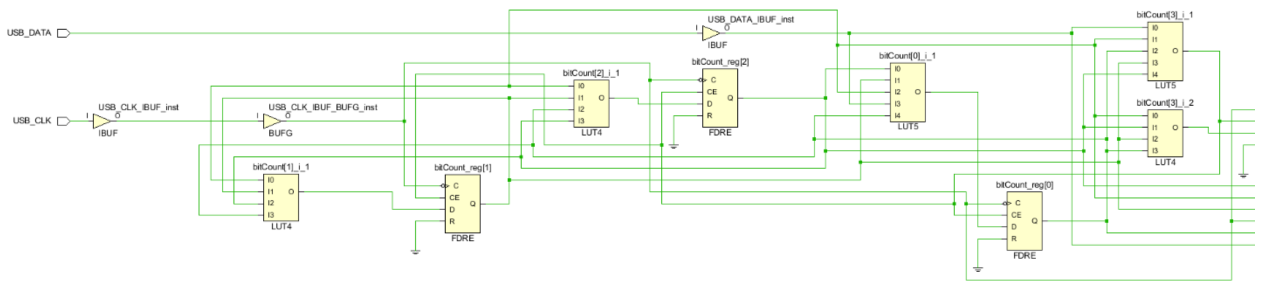
Project Task C: Keyboard Controller

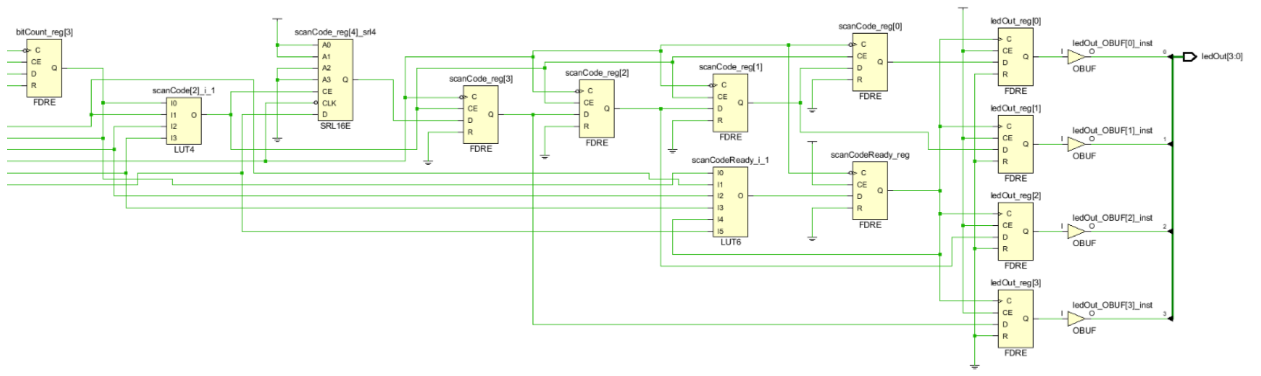
VHDL Source Code:

keyboardController.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity keyboardController is
    Port ( USB_CLK  : in  std_logic;
           USB_DATA : in  std_logic;
           ledOut   : out std_logic_vector (3 downto 0) := "0000");
end keyboardController;

architecture Behavioral of keyboardController is

    signal bitCount      : integer range 0 to 100 := 0;
    signal scanCodeReady : std_logic := '0';
    signal scanCode      : std_logic_vector (7 downto 0);
    signal breakReceived : std_logic := '0';

begin

    keyboardScanReadyEnable : process (USB_CLK) is
    begin
        if falling_edge(USB_CLK) then
            if bitCount = 0 AND USB_DATA = '0' then
                scanCodeReady <= '0';
                bitCount <= bitCount + 1;
            elsif bitCount > 0 AND bitCount < 9 then
                scanCode <= USB_DATA & scanCode (7 downto 1);
                bitCount <= bitCount + 1;
            elsif bitCount = 9 then
                bitCount <= bitCount + 1;
            elsif bitCount = 10 then
                scanCodeReady <= '1';
                bitCount <= 0;
            end if;
        end if;
    end process keyboardScanReadyEnable;

    scanKeyboard : process (scanCodeReady, scanCode) is
    begin
        if rising_edge(scanCodeReady) then
            if breakReceived = '1' then
                breakReceived <= '0';
            elsif breakReceived = '0' then
                if scanCode = "11110000" then
                    breakReceived <= '1';
                end if;
            end if;
            ledOut <= scanCode (3 downto 0);
        end if;
    end process scanKeyboard;

end Behavioral;
```

Design Schematic:





Truth Tables:

There are no Vivado-generated truth tables for this project.

Simulation Code:

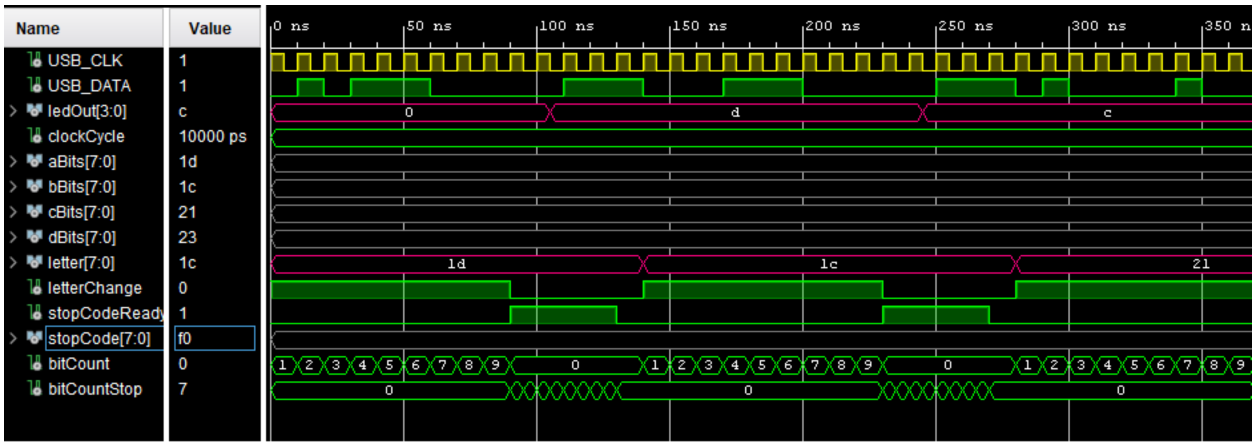
keyboardController_bench.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity keyboardController_bench is
--  Port ( );
end keyboardController_bench;

architecture bench of keyboardController_bench is

component keyboardController is
    Port ( USB_CLK           : in  std_logic;
           USB_DATA          : in  std_logic;
           ledOut            : out std_logic_vector (3 downto 0) );
end component;

    --port signals
    signal USB_CLK, USB_DATA : std_logic := '0';
    signal ledOut            : std_logic_vector (3 downto 0);
    signal clockCycle        : time                          := 10 ns;

    --test values
    signal aBits             : std_logic_vector (7 downto 0) := "00011101";
    signal bBits             : std_logic_vector (7 downto 0) := "00011100";
    signal cBits             : std_logic_vector (7 downto 0) := "00100001";
    signal dBits             : std_logic_vector (7 downto 0) := "00100011";

    --state trackers and counters
    signal letter            : std_logic_vector (7 downto 0) := "00100011";
    signal letterChange      : std_logic                     := '0';
    signal stopCodeReady     : std_logic                     := '0';
    signal stopCode          : std_logic_vector (7 downto 0) := "11110000";
    signal bitCount          : integer range 0 to 100        := 0;
    signal bitCountStop      : integer range 0 to 100        := 0;

begin

    keyboardController1: keyboardController
    port map ( USB_CLK  => USB_CLK,
               USB_DATA => USB_DATA,
               ledOut   => ledOut );

    clockProcess: process is
    begin
        USB_CLK <= '1';
        wait for clockCycle/2;
        USB_CLK <= '0';
        wait for clockCycle/2;
    end process clockProcess;

    testLetterSelector: process (letterChange) is
    begin
        if rising_edge(letterChange) then
            if letter = aBits then
                letter <= bBits;
            elsif letter = bBits then
                letter <= cBits;
            elsif letter = cBits then
                letter <= dBits;
            elsif letter = dBits then
                letter <= aBits;
            end if;
```

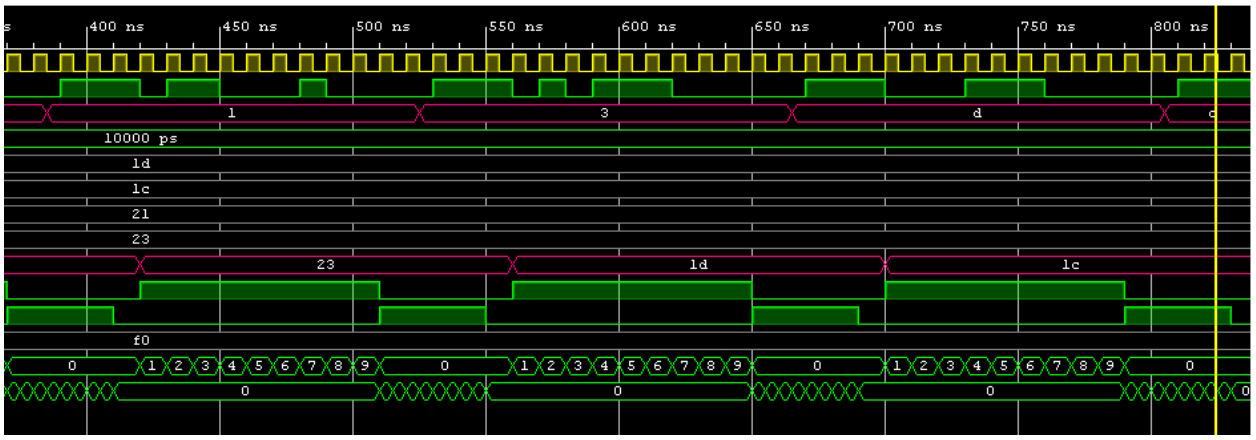
keyboardController_bench.vhd continued…

```
        end if;
    end process testLetterSelector;

    bitSenderProcess: process (USB_CLK, stopCodeReady, bitCount) is
    begin
        if stopCodeReady = '0' then
            if rising_edge(USB_CLK) then
                if bitCount = 0 then
                    USB_DATA <= '0';
                    bitCount <= (bitCount + 1);
                    letterChange <= '1';
                elsif bitcount > 0 AND bitCount < 9 then
                    USB_DATA <= letter (bitCount - 1);
                    bitCount <= (bitCount + 1);
                elsif bitCount = 9 then
                    USB_DATA <= '1';
                    bitCount <= 0;
                    letterChange <= '0';
                    stopCodeReady <= '1';
                end if;
            end if;
        elsif stopCodeReady = '1' then
            if bitCountStop > -1 AND bitCountStop < 8 then
                USB_DATA <= stopCode (bitCountStop);
                bitCountStop <= (bitCountStop + 1);
            elsif bitCountStop = 8 then
                bitCountStop <= 0;
                stopCodeReady <= '0';
            end if;
        end if;
    end process bitSenderProcess;

end bench;
```

Simulation Waveform:



Name	Value
USB_CLK	1
USB_DATA	1
ledOut[3:0]	c
clockCycle	10000 ps
aBits[7:0]	1d
bBits[7:0]	1c
cBits[7:0]	21
dBits[7:0]	23
letter[7:0]	1c
letterChange	0
stopCodeReady	1
stopCode[7:0]	f0
bitCount	0
bitCountStop	7



Constraints File:

```
set_property PACKAGE_PIN F4      [get_ports USB_CLK]
set_property PACKAGE_PIN B2      [get_ports USB_DATA]
set_property PACKAGE_PIN H17     [get_ports {ledOut[0]}]
set_property PACKAGE_PIN K15     [get_ports {ledOut[1]}]
set_property PACKAGE_PIN J13     [get_ports {ledOut[2]}]
set_property PACKAGE_PIN N14     [get_ports {ledOut[3]}]
set_property IOSTANDARD  LVCMOS33 [get_ports {ledOut[3]}]
set_property IOSTANDARD  LVCMOS33 [get_ports {ledOut[2]}]
set_property IOSTANDARD  LVCMOS33 [get_ports {ledOut[1]}]
set_property IOSTANDARD  LVCMOS33 [get_ports {ledOut[0]}]
set_property IOSTANDARD  LVCMOS33 [get_ports USB_CLK]
set_property IOSTANDARD  LVCMOS33 [get_ports USB_DATA]
```

Summary:

Work Completed:

For this lab, we created a keyboard interface. To accomplish this, we captured each bit as it was coming in, counting up. We tossed the first and fed the next eight into an eight-bit register from the bottom up to reverse their order. The last bit is used to signal the code is ready to be sent to the LED outputs. We then waiting for the stop code of F0 before accepting new inputs.

All parts of the lab were attempted and completed.

Problems Encountered:

The simulation code was the hardest part of this program. I spent some time trying to replicate exactly how the data would be read out serially, rather than hard-coding in sequences of 1s ad 0s. It took some extra time and a fair bit of messing around with new errors to me (multiply-driven values), but it paid off with a beautiful simulation.

Helpful Hints:

The fact that you can trigger processes on the rising or falling edge of internal signals is incredibly helpful for the simulation.

Suggested Improvements:

None at this time.