# Abstract

The goal of this lab was to create a circuit in VHDL that uses combinational and sequential logic circuits. This was done by making a VHDL program that would output 4 signals that are unique from each other (figure 1). The program would take a 50 MHz clock, derive it to 50 kHz, then using that derived clock signal to output a sync signal, and 3 data signals on the FPGA output pins along with the clock signals. The output signals were then read on using the analog discovery to see if they match the desired output signals. In the end the lab was successful. We were able to successfully output the desired signals to the output pins on the FPGA pins with minor issues (figure 4).

# The Process

To complete this lab, we first created a state diagram (figure 2) after realizing that the signals rely on a 3 bit counter. We drew the states and determined the output logic using k-maps. Although we could have just used conditional statements or case statements within the process to achieve the same function, I believe that this method used less lines to achieve the same goal. We used code made by Dr. Aamodt as a basis for the clock divider and the derived output logic to create the code. The signals were then sent out of Extout 0-4, Extout 7, Tek1 0-4, Tek1 7, and Tek5 (figure 3). The only errors we encountered were syntax errors that were easily corrected and an error in the output that was caused by the first and third bits of the ‘Q’ signal being swapped. We also found that the number of size flip flops used was 14 of 120800, the number of occupied slices was 8 of 15850, the number of bonded IOB pins was 19 of 300, the number of BUFGs was 2 of 32, and the average fanout of non-clock nets was 2.54.Diagram

Description automatically generated

Figure 1: The desired output signals

Text, letter

Description automatically generated

Figure 2: State diagram and output logic

Table

Description automatically generated with medium confidence

Figure 3: The signals and their connections

Graphical user interface, application, table, Excel

Description automatically generated

Figure 4: Data recorded on the Analog Discovery

# The Code

----------------------------------------------------------------------------------

-- Company:

-- Engineer: Eric Walsh & Nicholas Zimmerman

--

-- Create Date: 14:29:41 11/02/2021

-- Design Name:

-- Module Name: Lab6\_top\_sch - Behavioral

-- Project Name: Digital Design Lab 6

-- Target Devices: Aritix-7 (Xilinx ISE)

-- Tool versions:

-- Description: Generates multiple signals from a 50 MHz clock derived from a 50 GHz master clock

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

----------------------------------------------------------------------------------

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

library UNISIM;

use UNISIM.VComponents.all;

entity Lab6\_top\_sch is

port( mclk : in std\_logic;

sw0 : in std\_logic;

Extout : out std\_logic\_vector(7 downto 0); --using 6 bits out of 8 bits

TEK1 : out std\_logic\_vector(7 downto 0); --using 6 bits out of 8 bits

TEK5 : out std\_logic );

end Lab6\_top\_sch;

architecture Structural of Lab6\_top\_sch is

--slow\_clock generator

signal clk\_next, clk\_reg : unsigned(9 downto 0); --2^9=512

signal t\_next, t\_reg : std\_logic;

signal slow\_clk : std\_logic;

--FSM

signal q,d : unsigned(2 downto 0);

--Generated signals

signal sync, data1, data2, data3 : std\_logic;

begin

-----------------------------------------------------------------------------------

-- slow\_clock generator E.Welch & N. Zimmerman

--

-- Based on L.Aamodt's derived clock generator

-----------------------------------------------------------------------------------

process(mclk)

begin

if (mclk'event and mclk='1') then

clk\_reg <= clk\_next;

t\_reg <= t\_next; -- T-f/f register

end if;

end process;

clk\_next <= (others=>'0') when clk\_reg=499 else clk\_reg+1;

t\_next <= (not t\_reg) when clk\_reg = 499 else t\_reg; --Switch every 500, divide by 1000

Clk\_Buffer: BUFG -- Put t\_reg on a buffered clock line

port map ( I => t\_reg, O => slow\_clk);

-- slowclk is a square wave with 50 kHz frequency

-----------------------------------------------------------------------------------

-- Top-level structural architecture E.Welch & N.Zimmerman

-----------------------------------------------------------------------------------

--DFF

process(slow\_clk, sw0)

begin

if (slow\_clk'event and slow\_clk='1') then

q <= d;

end if;

end process;

--N.S. Logic

d <= (others=>'0') when q=7 else q+1;

--Output Logic (could have used a process with a case statement [type-casting?])

sync <= sw0 and not(q(0)) and not(q(1)) and not(q(2));

data1 <= sw0 and q(0) and (q(2) xor q(1));

data2 <= sw0 and (q(2) or q(1));

data3 <= sw0 and q(1) and (not(q(2)) or q(0));

--Output signals

Extout(0) <= sync;

TEK1(0) <= sync;

Extout(1) <= data1;

TEK1(1) <= data1;

Extout(2) <= data2;

TEK1(2) <= data2;

Extout(3) <= data3;

TEK1(3) <= data3;

Extout(4) <= slow\_clk;

TEK1(4) <= slow\_clk;

Extout(7) <= mclk;

TEK1(7) <= mclk;

TEK5 <= mclk;

--Unused outputs

Extout(5) <= '0';

Extout(6) <= '0';

TEK1(5) <= '0';

TEK1(6) <= '0';

end Structural;