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.

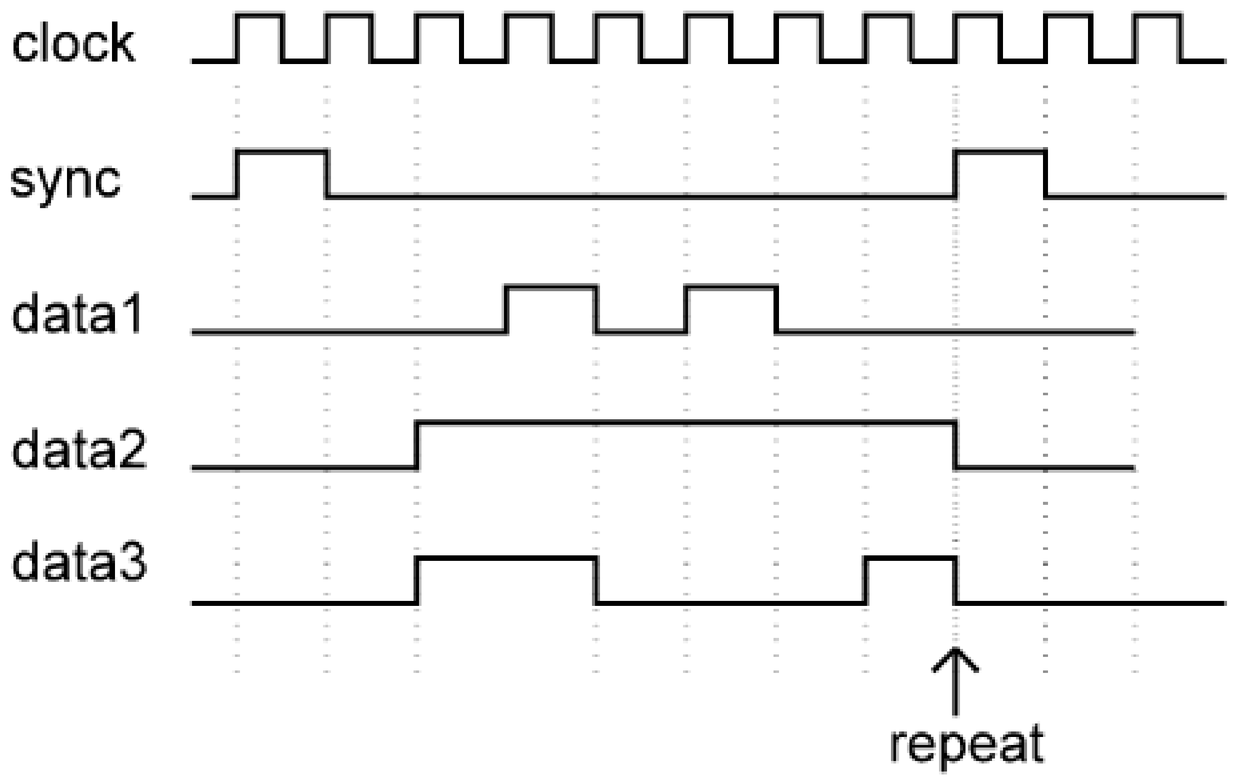
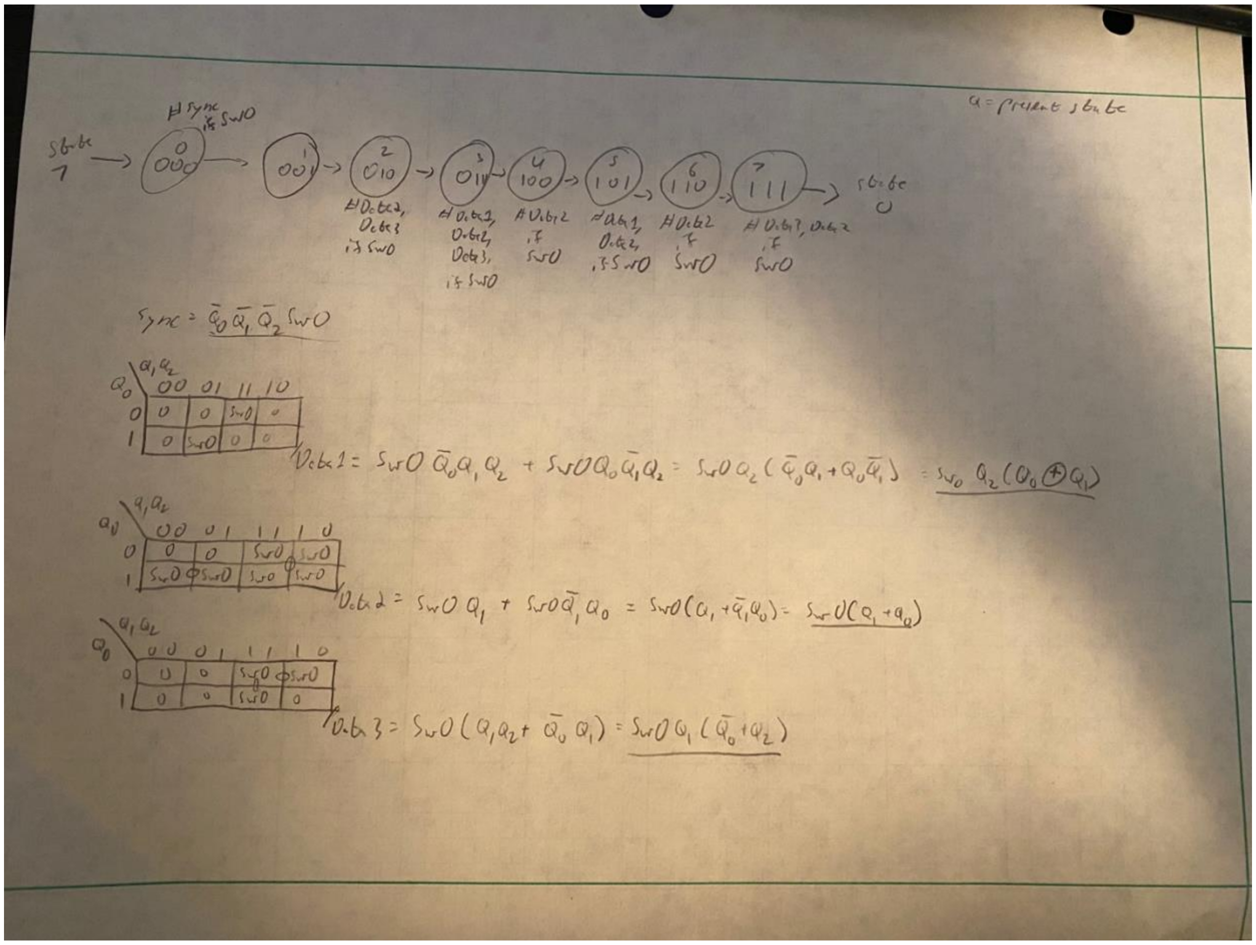


Figure 1: The desired output signals



$\text{Sync} = \underline{\bar{Q}_0 \bar{Q}_1 \bar{Q}_2 \text{Sw0}}$

Q_0 \ Q_1Q_2	00	01	11	10
0	0	0	Sw0	0
1	0	Sw0	0	0

$\text{Data1} = \text{Sw0}\,\bar{Q}_0 Q_1 Q_2 + \text{Sw0}\,Q_0 \bar{Q}_1 Q_2 = \text{Sw0}\,Q_2(\bar{Q}_0 Q_1 + Q_0 \bar{Q}_1) = \underline{\text{Sw0}\,Q_2(Q_0 \oplus Q_1)}$

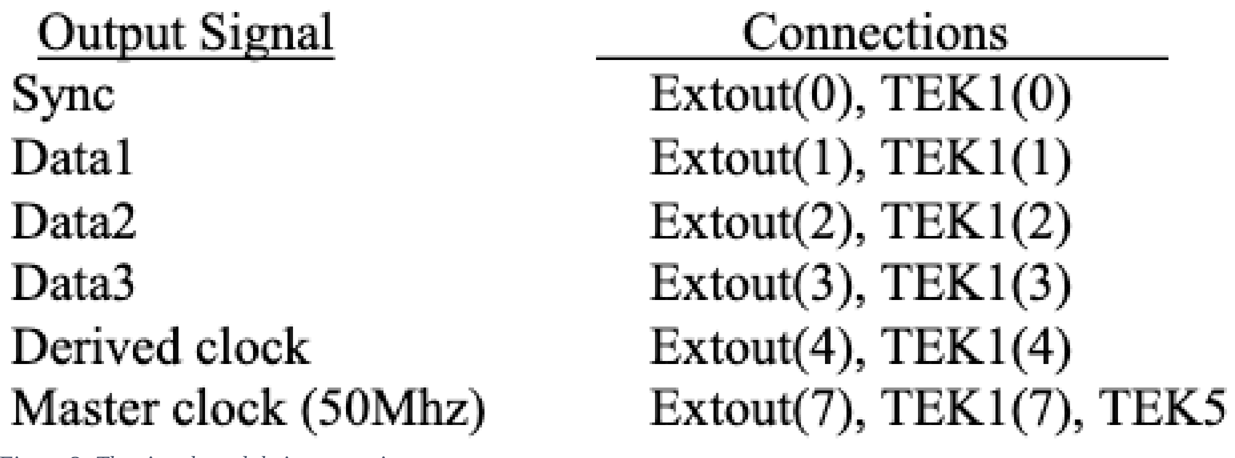
Q_0 \ Q_1Q_2	00	01	11	10
0	0	0	Sw0	Sw0
1	Sw0	Sw0	Sw0	Sw0

$\text{Data2} = \text{Sw0}\,Q_1 + \text{Sw0}\,\bar{Q}_1 Q_0 = \text{Sw0}(Q_1 + \bar{Q}_1 Q_0) = \underline{\text{Sw0}(Q_1 + Q_0)}$

Q_0 \ Q_1Q_2	00	01	11	10
0	0	0	Sw0	Sw0
1	0	0	Sw0	0

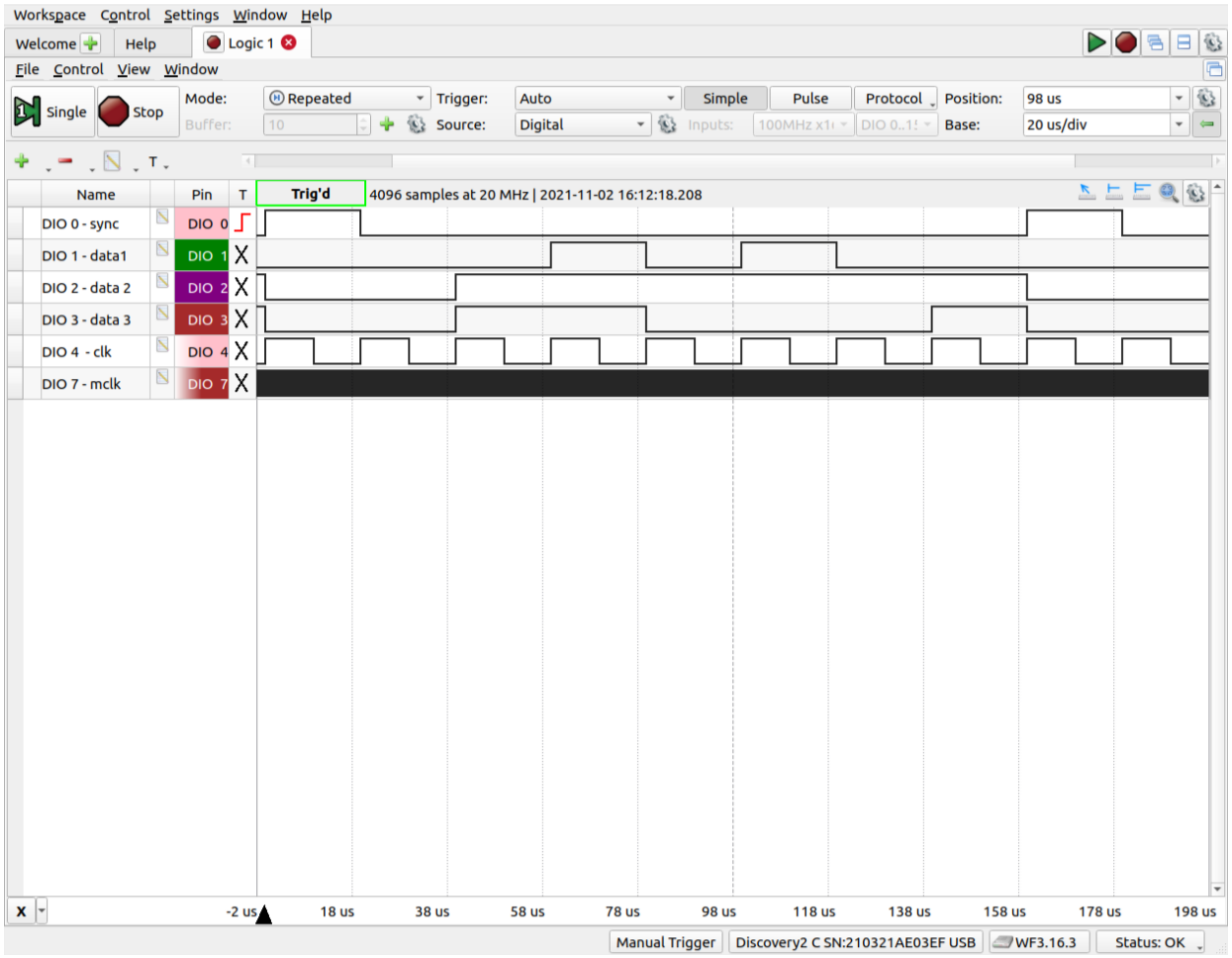
$\text{Data3} = \text{Sw0}(Q_1 Q_2 + \bar{Q}_0 Q_1) = \underline{\text{Sw0}\,Q_1(\bar{Q}_0 + Q_2)}$

Figure 2: State diagram and output logic



Output Signal	Connections
Sync	Extout(0), TEK1(0)
Data1	Extout(1), TEK1(1)
Data2	Extout(2), TEK1(2)
Data3	Extout(3), TEK1(3)
Derived clock	Extout(4), TEK1(4)
Master clock (50Mhz)	Extout(7), TEK1(7), TEK5

Figure 3: The signals and their connections



Mode: Repeated
Buffer: 10
Trigger: Auto
Source: Digital
Inputs: 100MHz x1
Position: 98 us
Base: 20 us/div

4096 samples at 20 MHz | 2021-11-02 16:12:18.208

Name	Pin
DIO 0 - sync	DIO 0
DIO 1 - data1	DIO 1
DIO 2 - data 2	DIO 2
DIO 3 - data 3	DIO 3
DIO 4 - clk	DIO 4
DIO 7 - mclk	DIO 7

Discovery2 C SN:210321AE03EF USB | WF3.16.3 | Status: OK

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;
```