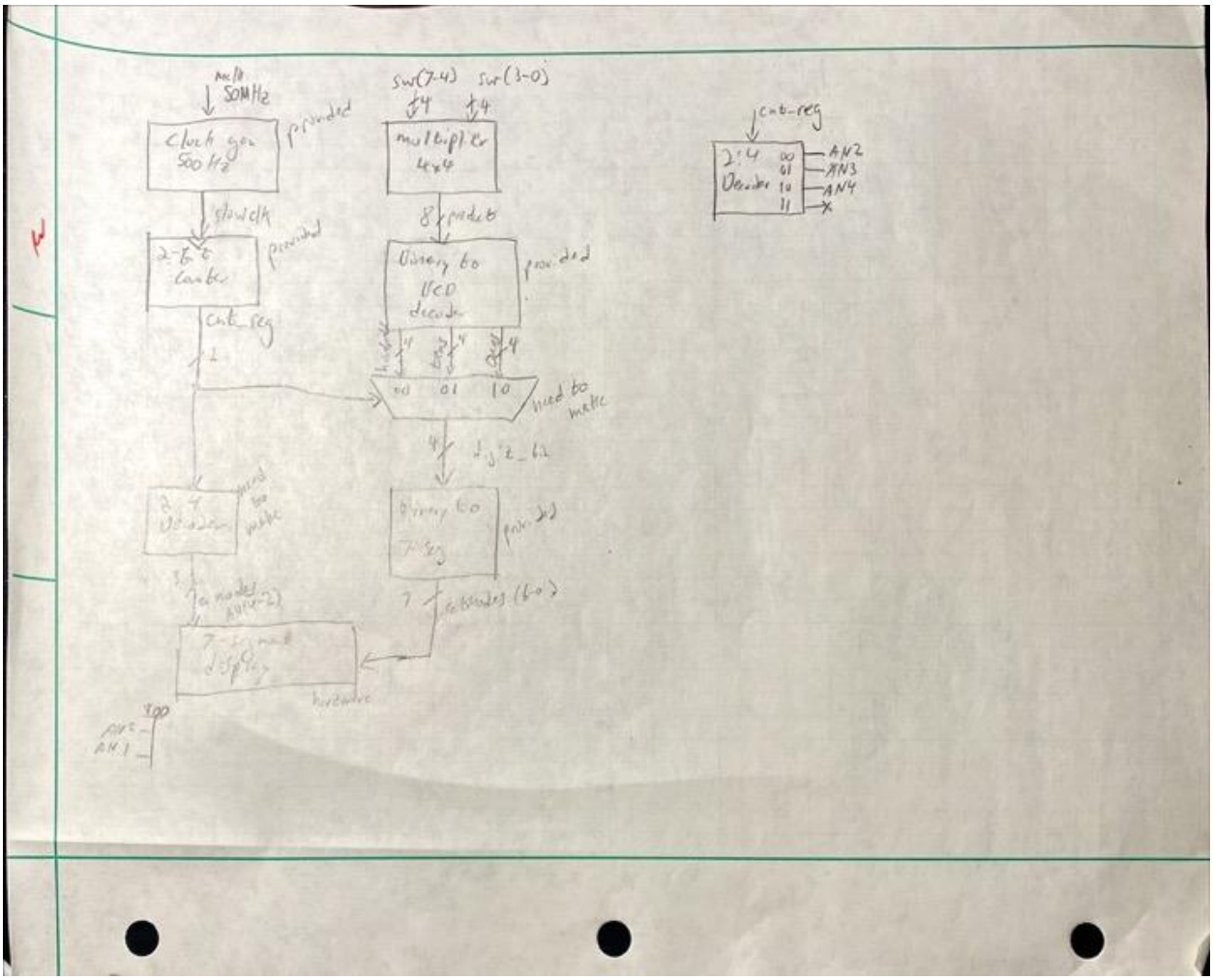
Abstract

The object of this lab was to continue the work done on the previous lab. The previous lab set out to create a 4-bit multiplier that would output an 8-bit result onto 8 LEDS. This lab expanded on that by outputting the result to the 7-segment display in decimal. This required minimal changes to the original VHDL code. However, it required the use of additional code to properly output it to the 7-segment display. The result was achieved with minimal errors in the code and the lab was successful.

The Process

The process of the lab was simple, we took our code from the previous lab and combined it with code made by Dr. Aamodt. The code from Dr. Aamodt was a clock divider that took a 5 MHz clock signal and output a 500 Hz clock signal, a binary to BCD decoder, a 2-bit counter, and a binary to 7-segment decoder. The task we were presented with was to create a multiplexer and a 2:4 decoder. The multiplexer took in signals from the binary to BCD converter and the 2-bit counter and output the signals to the binary to 7 – segment decoder. The 2:4 decoder took in signals from the 2-bit counter and output signals to the anodes of the 7 – segment display. The errors we ran into during this process were minor errors. This includes syntax errors and small mistakes made when attaching the old code to the new code, like issues with signals and ports. We also ran into an issue where the ones' place and the hundreds' place values on the 7 – segment display were swapped. This was fixed by simply going back to the data sheet and correcting the 2:4 decoder.

The Block Diagram



The Code

```
----------------------------------------------------------------------------------
-- Company: Walla Walla University
-- Engineer: Eric Walsh & Nicholas Zimmmeran
--
-- Create Date:    16:37:18 10/26/2021
-- Design Name:
-- Module Name:    Multiplier - Behavioral
-- Project Name:   Digital Design Lab 5
-- Target Devices: Artix-7 (Xilinx ISE)
-- Tool versions:
-- Description: Multiplies two 4-bit inputs and produces an 8-bit output. Switches are inputs and outputs are signals.
--
-- Dependencies:
--
```

```
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity adder_type1 is
	port(j1,j2,k1,k2,Cin : in std_logic;
				Cout, sum: out std_logic);
end adder_type1;

architecture Behavioral of adder_type1 is
	signal u, v, uXORv : std_logic;
begin
	u <= j1 and j2;
	v <= k1 and k2;

	--Full-adder
	uXORv <= u xor v;
	sum <= uXORv xor Cin;
	Cout <= (u and v) or (uXORv and Cin);

end Behavioral;
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity adder_type2 is
	port(j1,j2,k,Cin : in std_logic;
				Cout, sum: out std_logic);
end adder_type2;

architecture Behavioral of adder_type2 is
	signal u, uXORk : std_logic;
begin
	u <= j1 and j2;

	--Full-adder
	uXORk <= u xor k;
	sum <= uXORk xor Cin;
	Cout <= (u and k) or (uXORk and Cin);

end Behavioral;
----------------------------------------------------------------------------------
```

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

entity lab4_solution_top is
	port( A : in std_logic_vector (3 downto 0);
			B : in std_logic_vector (3 downto 0);
			r : out std_logic_vector (7 downto 0));
end lab4_solution_top;

architecture Behavioral of lab4_solution_top is

	component adder_type1
		port(j1,j2,k1,k2,Cin : in std_logic;
			Cout, sum: out std_logic);
	end component;

	component adder_type2
		port(j1,j2,k, Cin : in std_logic;
			Cout, sum: out std_logic);
	end component;

	signal C: std_logic_vector (12 downto 1);
	signal S2, S3, S4, S6, S7, S8: std_logic;
begin
	r(0) <= A(0) and B(0);
	add1: adder_type1 port map(A(0),B(1),A(1),B(0),'0',C(1),r(1));
	add2: adder_type1 port map(A(1),B(1),A(2),B(0),C(1),C(2),S2);
	add3: adder_type1 port map(A(2),B(1),A(3),B(0),C(2),C(3),S3);
	add4: adder_type2 port map(A(3),B(1),'0',C(3),C(4),S4);
	add5: adder_type2 port map(A(0),B(2),S2,'0',C(5),r(2));
	add6: adder_type2 port map(A(1),B(2),S3,C(5),C(6),S6);
	add7: adder_type2 port map(A(2),B(2),S4,C(6),C(7),S7);
	add8: adder_type2 port map(A(3),B(2),C(4),C(7),C(8),S8);
	add9: adder_type2 port map(A(0),B(3),S6,'0',C(9),r(3));
	add10: adder_type2 port map(A(1),B(3),S7,C(9),C(10),r(4));
	add11: adder_type2 port map(A(2),B(3),S8,C(10),C(11),r(5));
	add12: adder_type2 port map(A(3),B(3),C(8),C(11),r(7),r(6));
end Behavioral;

----------------------------------------------------------------------------
-- Company: Walla Walla University
-- Engineer:
--
-- Create Date:    14:02:46 10/26/2021
-- Design Name:
-- Module Name:    Lab5_top_sch - Structural
```

```
-- Project Name:
-- Target Devices: Artix-7 Xilinx ISE
-- Tool versions:
-- Description: Multiplies two 4-digit values entered with switches and outputs
--                  the result on a 7-segment display.
-- 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 Lab5_top_sch is
   port(
      sw: in std_logic_vector (7 downto 0);
      mclk: in std_logic;

      anode: out std_logic_vector (5 downto 1);       --rightmost 3 digits are anodes 4 downto 2
      cath: out std_logic_vector (7 downto 0)     -- a-g
   );
end Lab5_top_sch;

architecture Structural of Lab5_top_sch is
   --Data line
   signal product: std_logic_vector (7 downto 0);
   signal hundreds, tens, ones: std_logic_vector (3 downto 0);
   signal bcd: std_logic_vector (3 downto 0);

   --COMPONENTS--
   --clock_gen
   signal clk_next, clk_reg : unsigned(16 downto 0);
   signal t_next, t_reg : std_logic;
   signal slow_clk: std_logic;

   --counter
   signal cnt_next, cnt_reg : unsigned(1 downto 0);

   component lab4_solution_top
      port( A : in std_logic_vector (3 downto 0);
            B : in std_logic_vector (3 downto 0);
```

```
            r : out std_logic_vector (7 downto 0));
    end component;

    component bin2bcd_top
        port (b0,b1,b2,b3,b4,b5,b6,b7 : in  STD_LOGIC;
              grp0 : out  STD_LOGIC_VECTOR (3 downto 0); --ones
              grp1 : out  STD_LOGIC_VECTOR (3 downto 0); --tens
              grp2 : out  STD_LOGIC_VECTOR (3 downto 0)); --hundreds
    end component;

    component bcd2_7seg
       port (data : in  STD_LOGIC_VECTOR (3 downto 0);
             cath_out : out  STD_LOGIC_VECTOR (7 downto 0));
    end component;

begin

    ----------------------------------------------------------------------------
    --
    --      Derived Clock generator.  Generates square waves
    --      L.Aamodt
    --      As shown, if mclk is 50 Mhz, t_reg and slowclk are 500 Hz
    --
    ---------------- 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=49999 else clk_reg+1;
    t_next <= (not t_reg) when clk_reg = 49999 else t_reg;

    Clk_Buffer: BUFG                    -- Put t_reg on a buffered clock line
        port map ( I => t_reg, O => slow_clk);
        -- use slowclk to run flip/flops and counters;
        -- slowclk is a square wave with 500 Hz frequency

    ----------------------------------------------------------------------------
    --      Counter - counts 0,1,2,3 and then starts over at 0      L.Aamodt
    ----------------------------------------------------------------------------

    process(slow_clk)
```

```
    begin
        if (slow_clk'event and slow_clk='1') then
            cnt_reg <= cnt_next;
        end if;
    end process;

    cnt_next <= cnt_reg+1;

    --  Note: cnt_reg is the output you need to drive the mux and 2:4 decoder
    --  but you may need to type cast it back to std_logic_vector

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

    multiplier: lab4_solution_top port map( A => sw(7 downto 4),
                                            B => sw(3 downto 0),
                                            r => product );

    dec_tobcd: bin2bcd_top port map (   b0 => product(0),
                                        b1 => product(1),
                                        b2 => product(2),
                                        b3 => product(3),
                                        b4 => product(4),
                                        b5 => product(5),
                                        b6 => product(6),
                                        b7 => product(7),
                                        grp0 => ones,
                                        grp1 => tens,
                                        grp2 => hundreds );
    bcd <=  hundreds when (cnt_reg="00") else
              tens when (cnt_reg="01") else
              ones;   --2 clk cycles
    dec_to7seg: bcd2_7seg port map( data => bcd,
                                    cath_out => cath ); --cathode 7 always set to '1' in the block
    with cnt_reg select
       anode(4 downto 2) <= "110" when "00", --anode 2
                            "101" when "01", --anode 3
                            "011" when "10", --anode 4
                            "111" when others;
    anode(1) <= '1'; --Leftmost digit unused
    anode(5) <= '1'; --Colon unused
end Structural;
```