Lab 2 - Stopwatch

Nate Herbst
A02307138
Nathan Walker
A02364124

Introduction

In this assignment, we implemented a stopwatch on the DE-10 Lite development board. The stopwatch was designed to accurately display time down to hundredths of a second in the format MM.SS.HH (minutes, seconds, hundredths of a second) using six seven-segment displays. Two push buttons controlled the stopwatch, with one acting as the reset and the other as the start/stop control.

Procedure

Initial Setup and Mistakes

We started by setting up the development environment using Questa for simulation. Initially, it took a while to get Questa working properly because we mistakenly included a `.v` file instead of the required `.vhd` file. This caused simulation errors and delayed our progress until we corrected the file linking.

Stopwatch Design Approach

Our design utilized six different counters, one for each digit in the time display. Each counter was responsible for a particular segment (hundredths, seconds, or minutes). The challenge here was calculating the clock cycles accurately for each digit without using a gated or register-driven clock as per the assignment's requirements.

1. **Clock Divisions**: We used the 50 MHz clock available on the DE-10 Lite board and calculated the appropriate division values for each segment:
    a. For hundredths of a second, we used `div = 500,000`.

    b. For seconds, we calculated divisions like `div = 50,000,000` for the lower seconds place. For minutes, the upper tens place needed to count up to 6 (i.e., div = 30,000,000,000).
2. **Decimal Point Placement**: Another key challenge was ensuring that the decimal points in the display were correctly placed between the seconds and hundredths places. To resolve this, we used masking logic in the VHDL code:

```
HEX4 <= "01111111" and HEX4_temp;

HEX2 <= "01111111" and HEX2_temp;
```

Mistakes and Troubleshooting

Throughout the process, we encountered several roadblocks:

- Miscounting the limits of the tens place for minutes, which led to incorrect time representation. Initially, we forgot that the tens place for minutes should only count up to 6 (representing 60 minutes), but corrected this after running tests.
- During simulation, the counters didn't seem to increment correctly at first. Upon reviewing the clock divisions, we realized the misconfiguration and updated the generic values in the counter modules.
- We initially overlooked including some important signals in our entity declaration, which caused confusion and errors during compilation.

Results

- Once the design was properly implemented, the stopwatch worked as intended:
- The six-digit display accurately tracked time from 00.00.00 up to 59.59.99.
- The decimal points appeared in the correct positions, dividing the seconds and hundredths.
- The stopwatch reset functionality worked correctly, resetting the display to zero upon pressing the reset button.
- No hardware issues like fried wires or burned boards occurred, and we were able to test the functionality using the simulation environment without any physical damage to the DE-10 Lite.

The final implementation was able to meet all the assignment's requirements for timing accuracy and display format.

Figures

Figure 1: Block Diagram

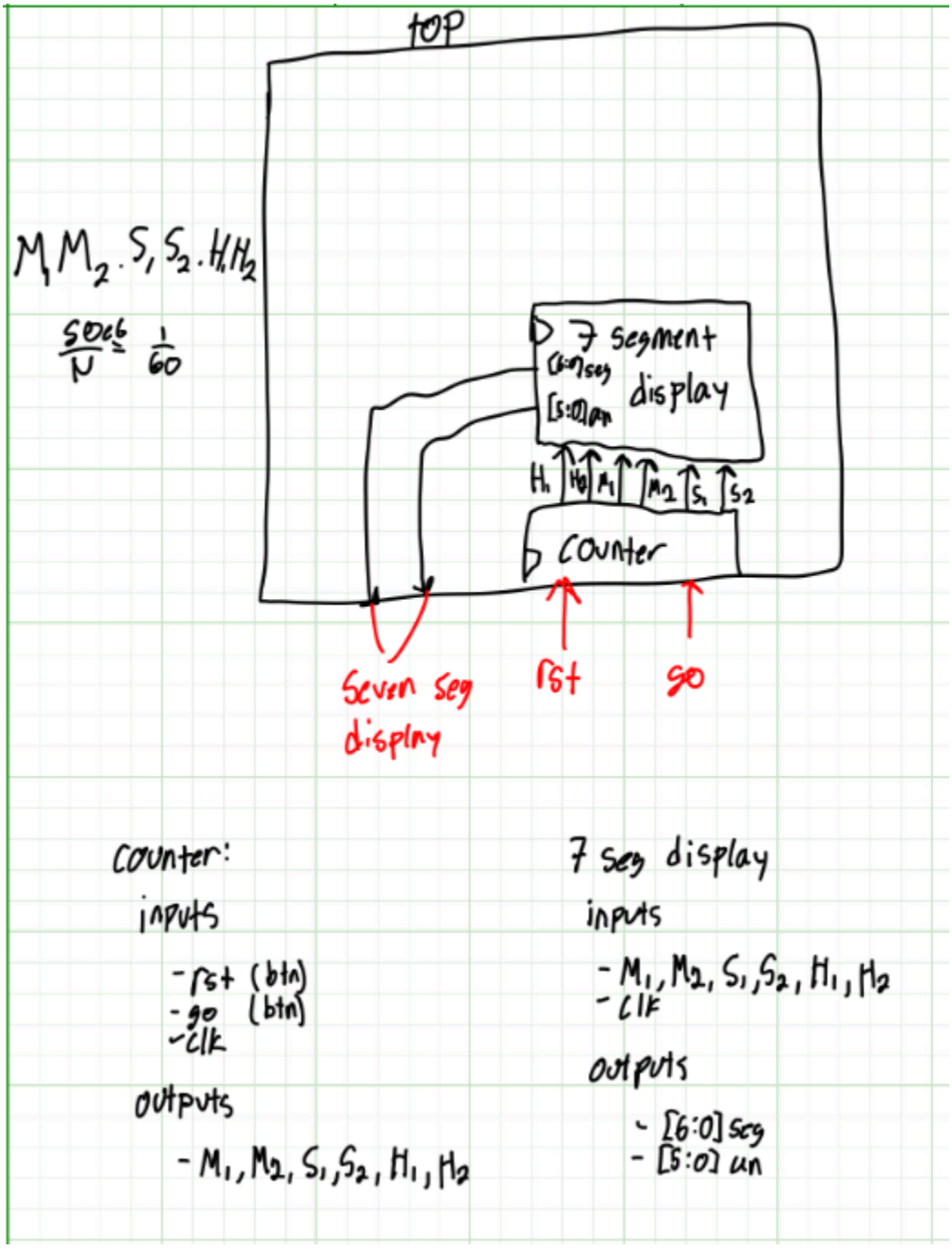


Figure 2: VHDL Code for Counter

```
counter_S1 : counter

        generic map (

            n => 4,

            div => 500000000,

            div_min => 5

        )

        port map (

            clk => MAX10_CLK1_50,

            rst_l => key(0),

            go => key(1),

            output => IN3

        );
```

This code snippet shows the counter setup for seconds counting, with the clock division set to 50,000,000.

Conclusion

In conclusion, the stopwatch project provided valuable experience in managing timing in VHDL without gated clocks and required careful consideration of clock divisions and digit limits. Despite early challenges, including simulation issues and incorrect file linking, we were able to overcome these and successfully implement a stopwatch that met the project requirements. The final design was tested through simulation and was able to track time down to hundredths of a second, with properly placed decimal points, and functional reset and go buttons.

Final Code Implementation - Top entity:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity Lab2_stopwatch is
   port (
       HEX0 : out std_logic_vector(7 downto 0);
       HEX1 : out std_logic_vector(7 downto 0);
       HEX2 : out std_logic_vector(7 downto 0);
       HEX3 : out std_logic_vector(7 downto 0);
       HEX4 : out std_logic_vector(7 downto 0);
       HEX5 : out std_logic_vector(7 downto 0);
       ADC_CLK_10 : IN STD_LOGIC;
       MAX10_CLK1_50 : IN STD_LOGIC;
       MAX10_CLK2_50 : IN STD_LOGIC;
       KEY : IN STD_LOGIC_VECTOR(1 DOWNTO 0)
   );
end Lab2_stopwatch;


architecture componentlist of Lab2_stopwatch is
   component HEX_seven_seg_disp_6
       port (
           clk : in std_logic;
           IN0 : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
           IN1 : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
           IN2 : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
           IN3 : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
           IN4 : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
           IN5 : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
           HEX0 : out std_logic_vector(7 downto 0);
           HEX1 : out std_logic_vector(7 downto 0);
           HEX2 : out std_logic_vector(7 downto 0);
```

```
            HEX3 : out std_logic_vector(7 downto 0);
            HEX4 : out std_logic_vector(7 downto 0);
            HEX5 : out std_logic_vector(7 downto 0)
        );
    end component HEX_seven_seg_disp_6;

    component counter
        generic (
            n : integer := 4;
            div : integer := 4;
            div_min : integer := 9
        );
        port (
            clk : in std_logic;
            rst_l : in  std_logic;
            go : in std_logic;
            output : out std_logic_vector((n-1) downto 0)
        );
    end component counter;
    signal IN0 : std_logic_vector(3 downto 0);
    signal IN1 : std_logic_vector(3 downto 0);
    signal IN2 : std_logic_vector(3 downto 0);
    signal IN3 : std_logic_vector(3 downto 0);
    signal IN4 : std_logic_vector(3 downto 0);
    signal IN5 : std_logic_vector(3 downto 0);
    signal hex4_temp : std_logic_vector(7 downto 0);
    signal hex2_temp : std_logic_vector(7 downto 0);
begin
    display : HEX_seven_seg_disp_6
        port map (
            clk => MAX10_CLK1_50,
            IN0 => IN0,
            IN1 => IN1,
            IN2 => IN2,
            IN3 => IN3,
            IN4 => IN4,
            IN5 => IN5,
            HEX0 => HEX0,
            HEX1 => HEX1,
            HEX2 => HEX2_temp,
            HEX3 => HEX3,
            HEX4 => HEX4_temp,
            HEX5 => HEX5
        );
    counter_H2 : counter
        generic map (
            n => 4,
            div => 500000,
            div_min => 9
        )
        port map (
            clk => MAX10_CLK1_50,
            rst_l => key(0),
            go => key(1),
            output => IN0
        );
```

```
counter_H1 : counter
    generic map (
        n => 4,
        div =>5000000,
        div_min => 9
    )
    port map (
        clk => MAX10_CLK1_50,
        rst_l => key(0),
        go => key(1),
        output => IN1
    );
counter_S2 : counter
    generic map (
        n => 4,
        div => 50000000,
        div_min => 9
    )
    port map (
        clk => MAX10_CLK1_50,
        rst_l => key(0),
        go => key(1),
        output => IN2
    );
counter_S1 : counter
    generic map (
        n => 4,
        div => 500000000,
        div_min => 5
    )
    port map (
        clk => MAX10_CLK1_50,
        rst_l => key(0),
        go => key(1),
        output => IN3
    );
counter_M2 : counter
    generic map (
        n => 4,
        div => 3000000000,
        div_min => 9
    )
    port map (
        clk => MAX10_CLK1_50,
        rst_l => key(0),
        go => key(1),
        output => IN4
    );
counter_M1 : counter
    generic map (
        n => 4,
        div => 30000000000,
        div_min => 5
    )
    port map (
        clk => MAX10_CLK1_50,
```

```
          rst_l => key(0),
          go => key(1),
          output => IN5
      );
   HEX4 <= "01111111" and HEX4_temp;
   HEX2 <= "01111111" and HEX2_temp;

end componentlist;
```

7 Seg single digit code:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity HEX_seven_seg_disp is
   port (
       hex : in std_logic_vector(3 downto 0);
       clk : in std_logic;
       oseg : out std_logic_vector(7 downto 0)
   );
end HEX_seven_seg_disp;

architecture behavior of HEX_seven_seg_disp is
   type HEX_TABLE is array (0 to 15) of std_logic_vector(7 downto 0);
   constant table : HEX_TABLE := ("11000000", "11111001", "10100100",
"10110000", "10011001", "10010010", "10000010",
       "11111000", "10000000", "10011000", "10001000", "10000011",
"11000110", "10100001", "10000110", "10001110");
begin
   process(clk)
   begin
   if rising_edge(clk) then
       case to_integer(unsigned(hex)) is
       when 0 =>
           oseg <= table(0);
       when 1 =>
           oseg <= table(1);
       when 2 =>
           oseg <= table(2);
       when 3 =>
```

```
            oseg <= table(3);
        when 4 =>
            oseg <= table(4);
        when 5 =>
            oseg <= table(5);
        when 6 =>
            oseg <= table(6);
        when 7 =>
            oseg <= table(7);
        when 8 =>
            oseg <= table(8);
        when 9 =>
            oseg <= table(9);
        when 10 =>
            oseg <= table(10);
        when 11 =>
            oseg <= table(11);
        when 12 =>
            oseg <= table(12);
        when 13 =>
            oseg <= table(13);
        when 14 =>
            oseg <= table(14);
        when 15 =>
            oseg <= table(15);
        when others =>
            oseg <= "11111111";
        end case;
    end if;
    end process;
end behavior;
```

7 Seg 6 digit code:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity HEX_seven_seg_disp_6 is
```

```
    port (
        IN0 : in std_logic_vector(3 downto 0);
        IN1 : in std_logic_vector(3 downto 0);
        IN2 : in std_logic_vector(3 downto 0);
        IN3 : in std_logic_vector(3 downto 0);
        IN4 : in std_logic_vector(3 downto 0);
        IN5 : in std_logic_vector(3 downto 0);
        clk : in std_logic;
        HEX0 : out std_logic_vector(7 downto 0);
        HEX1 : out std_logic_vector(7 downto 0);
        HEX2 : out std_logic_vector(7 downto 0);
        HEX3 : out std_logic_vector(7 downto 0);
        HEX4 : out std_logic_vector(7 downto 0);
        HEX5 : out std_logic_vector(7 downto 0)
    );
end HEX_seven_seg_disp_6;

architecture component_list of HEX_seven_seg_disp_6 is
    component HEX_seven_seg_disp
        port (
            clk : in std_logic;
            hex : in std_logic_vector(3 downto 0);
            oseg : out std_logic_vector(7 downto 0)
        );
    end component HEX_seven_seg_disp;
begin

    disp1 : HEX_seven_seg_disp
        port map (
            clk => clk,
            hex => IN0,
            oseg => HEX0
        );
    disp2 : HEX_seven_seg_disp
        port map (
            clk => clk,
            hex => IN1,
            oseg => HEX1
        );
    disp3 : HEX_seven_seg_disp
```

```
        port map (
            clk => clk,
            hex => IN2,
            oseg => HEX2
        );
    disp4 : HEX_seven_seg_disp
        port map (
            clk => clk,
            hex => IN3,
            oseg => HEX3
        );
    disp5 : HEX_seven_seg_disp
        port map (
            clk => clk,
            hex => IN4,
            oseg => HEX4
        );
    disp6 : HEX_seven_seg_disp
        port map (
            clk => clk,
            hex => IN5,
            oseg => HEX5
        );


end component_list;
```

Counter Code:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity counter is
    generic (
        n : integer := 4;
        div : integer := 4;
        div_min : integer := 9
    );
```

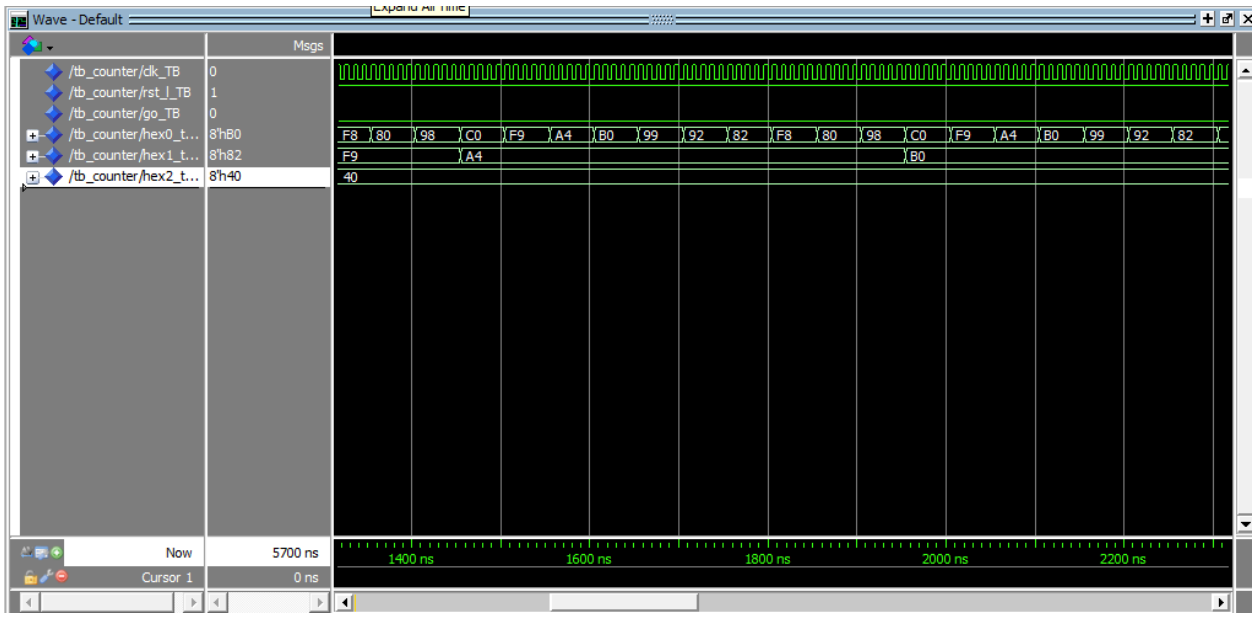
```
    port (
        clk : in std_logic;
        rst_l : in std_logic;
        go : in std_logic;
        output : out std_logic_vector((n-1) downto 0)
    );
end;
architecture simple of counter is

begin
    process(clk, rst_l)
        variable count : unsigned ((n-1) downto 0);
        variable dly : integer;
    begin
        if rst_l = '0' then
            count := (others => '0');
            dly := 0;
        elsif rising_edge(clk) then
            if go = '0' then
                if dly = (div-1) then
                    dly := 0;
                    count := count + 1;
                    if count > div_min then
                        count := (others => '0');
                    end if;
                else
                    dly := dly + 1;
                end if;
            end if;
        end if;

    output <= std_logic_vector(count);
    end process;


end simple;
```

Figures 1 - 3: Simulations



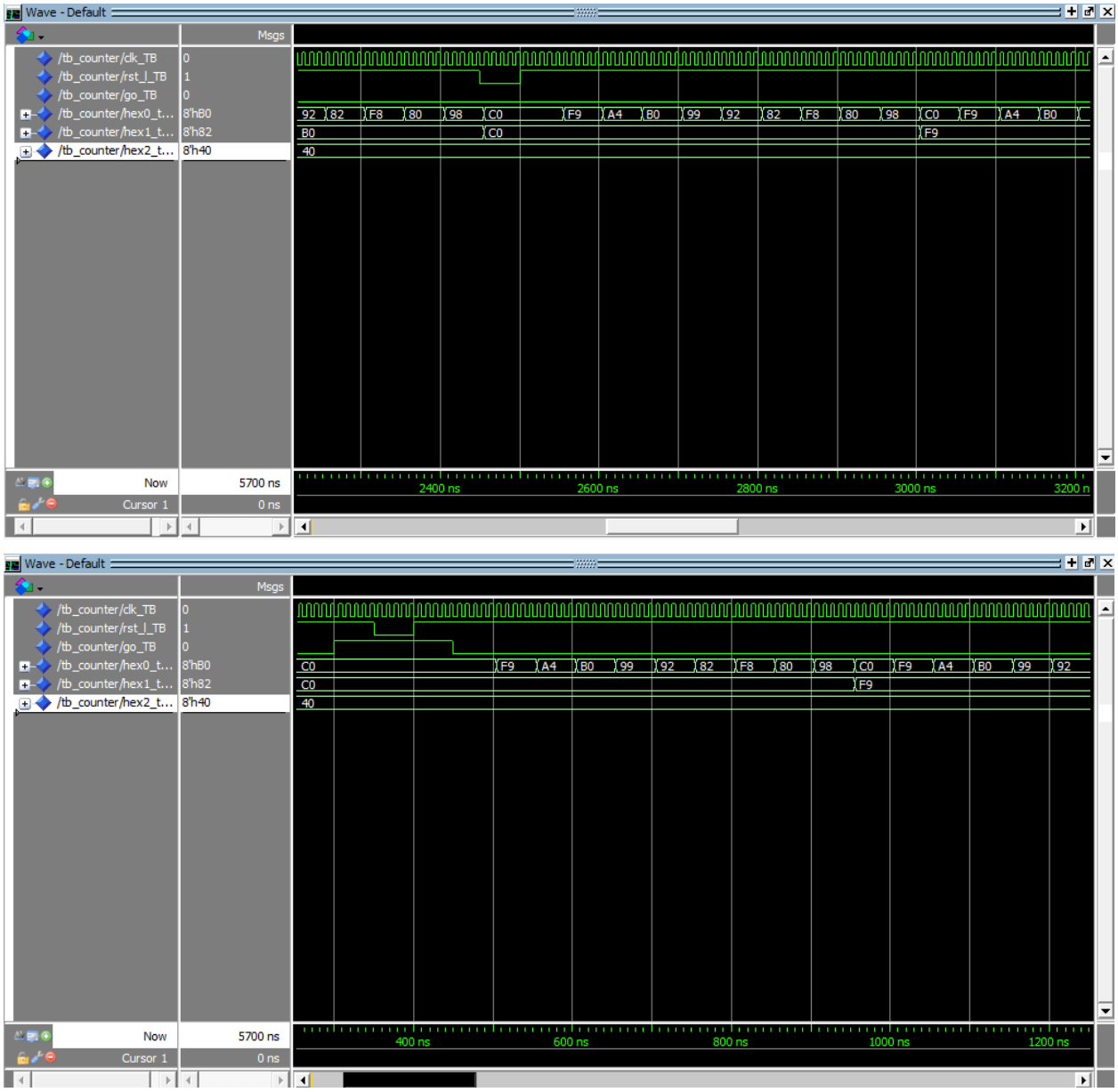


Figure 1 - 3: Screenshots of simulation of top level 7 seg and counters.