Term Project Report: Word Clock
EE 102
Section 02
Bilkent University

Doğa Zeynep Tarman
22303157

May 18, 2024

Purpose

The main purpose of my project was to design a Word Clock using the Basys3 board. The idea that I tried to implement was having a counter and UART transmitter on the Basys3 side, and a UART receiver part which lit up the LEDs on the Arduino.

Methodology

To start my project, I first had to decide on where I was going to implement my project on. This could be a breadboard or anything with similar properties. I drew a small schematic for myself to allow myself some flexibility in this part, and in this schematic I separated the board into a matrix of 9x3 in order to place the LEDs. I then proceeded to write the necessary VHDL code, which was a counter that increased by one every second. I also made it so that this code displays its output on the display of the Basys3 in order to understand what time it is better. In addition to this, I added two buttons to control the time. Following the part on the Basys3, I started to design the LED-lighting-up part on the Arduino. The reason why I started this before implementing the UART code on either side was because I hadn't decided on how to receive the data on the Arduino side. I wanted to first finish the program on one side, and design the UART accordingly on the other. Since I had to make sure that all the LEDs were working correctly, I also wrote a small matrix-test code on the Arduino side, which lights up LEDs in a row with half a second delay. After completing the Arduino's LED part, I went back to my Basys3 to write the UART code. In this part, the Basys3 sends a "beginning" signal at the clock signal's rising edge, sends 16 bits, then sends a "stop" bit. On the Arduino side, I wrote a small serial receiver code to catch the data that was being sent. I made sure to match their baud rates and set it to 9600 for it is a standard. It is important to keep in mind that the reason why there are 16 bits transmitted. I made it so that the data was in a "0000" format. In other words, each digit was represented with 4 bits, hence the 16 bits of data.

After making sure the coding part was going smoothly by checking the outputs with several methods such as connecting the board's output pin to the oscilloscope or reading the data from the serial monitor on the Arduino side, I started to build the circuit. Looking at my previously drawn schematic, I implemented the matrix on a 09cm x 15cm vero-board and connected the Arduino to the board as well. I then soldered two cables to the Arduino to establish the connection between it and the Basys3.

Design Specifications

My design mainly focuses on the proper data transmission between the two boards. Hence, the part on the Basys3 is quite heavier than the one on the Arduino. The input and output signals on the Basys3 as follows:

```
inputs:
clk: clock
reset: reset
add1min: adds 60 to the counter
add1hour: adds 3600 to the counter

outputs:
anod_a: 4 anode signals
led_o: cathode patterns for the display
tx_o: UART transmit output
tx_tick_o: UART done tick
```

To make my design operate properly, I made a hierarchical order in my code, which can be seen in Figure 1. There is a hexadecimal counter which is the top module, an UART transmitter sub-module, and a constraints file. All of the code that I wrote can be found in the Appendix.

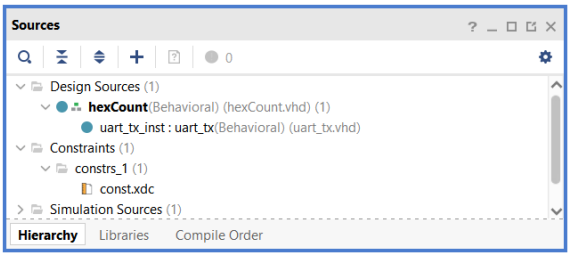

Figure 1: The Hierarchical Order

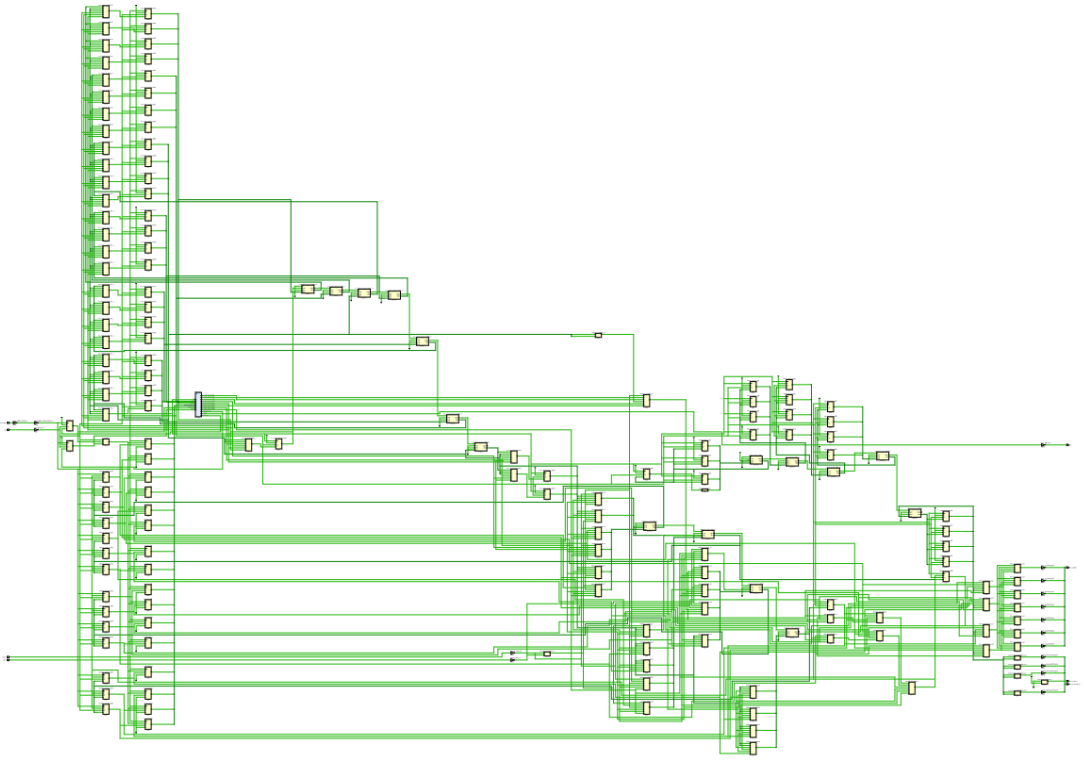

Figure 2: Implemented Design Schematic of the Top Module

As it can be seen from Figure 2, the design got more and more complicated over time as the UART code was introduced. This was because I only separated the UART part in my code, and wrote everything else inside of the main file. Hence, the code got way too complicated and started to get confusing. To handle this issue, I wrote comments all over the code that I have written for even the simplest of lines and named the variables in way that would remind me of their functions. I also left spaces between the lines for ease of readability. This helped me somewhat overcome the confusion that was caused by the complexity of my design.

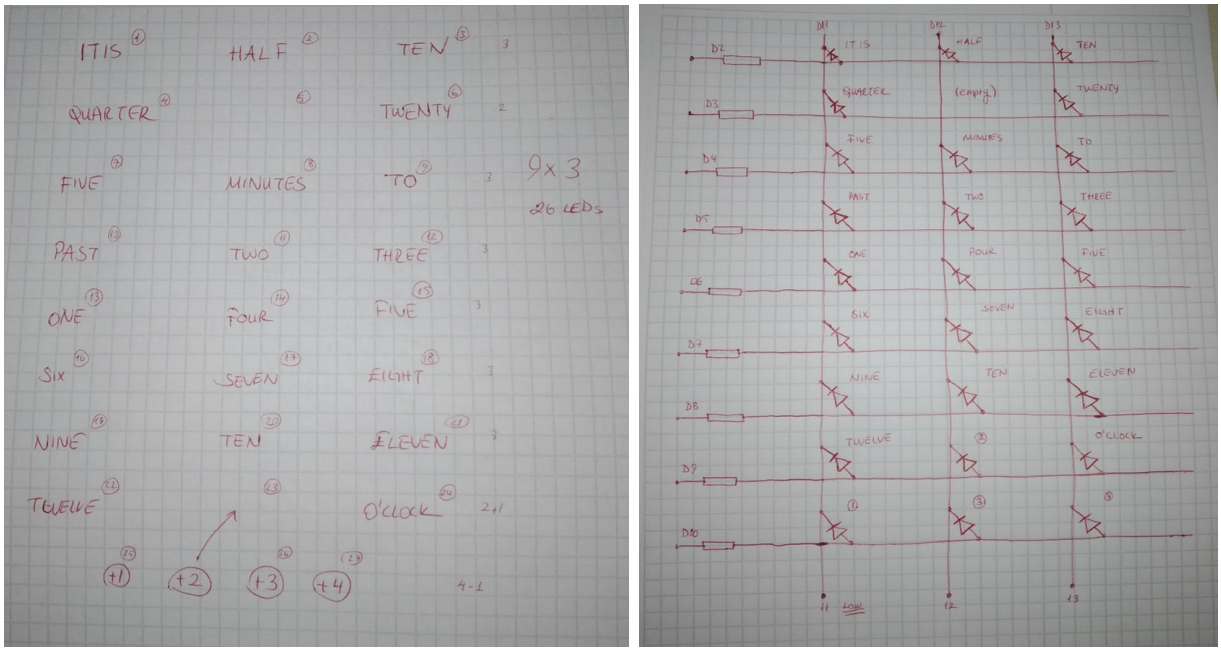

Figure 3 and 4: The Schematic I Drew

Additionally, I drew a small schematic for myself in order to better imagine what I would be implementing and how to write its code. The figures above, Figure 3 and 4, represent my planning of the circuit design. These drawings guided me through the code-writing process and allowed me to decide on what board to use.

Having completed the circuit design, it was now time to check the Basys3 code and move on to the Arduino. The following figures represent the RTL Schematic of my project.

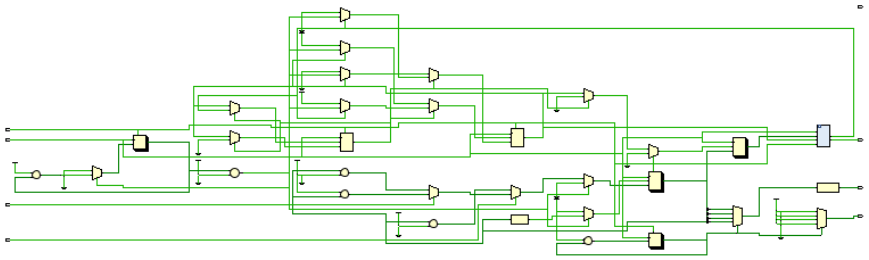

Figure 5: The RTL Schematic of the Top Module

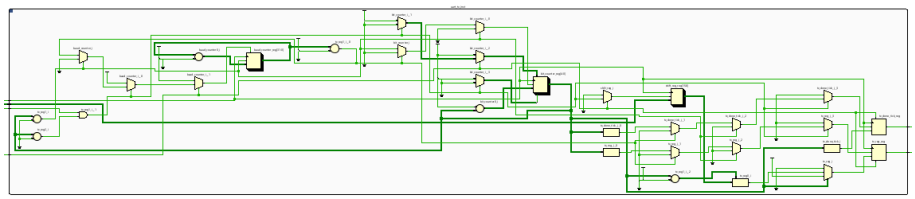

Figure 6: The RTL Schematic of the UART Module

On the Arduino side, I wrote three separate codes. Two of these were just for testing purposes, and only one of them was actually for the UART receiving purpose.

The first piece of code that I wrote on the Arduino was a matrix test. This code was for me to test whether the LEDs were working or not by lighting up every line one by one.

The second code that I wrote was for the receiving process from the FPGA board. This code basically had a serial printer function in it, which printed the data that it got from the Basys3 on the serial monitor. This helped me to determine whether I was sending the data correctly on the Basys3 side.

The last code, which was the actual Word Clock code, was the one that lit up the LEDs according to the data that it got from the Basys3.

Results

The video for my project can be found here.

As I have specified in the parts before, I designed my circuit on a piece of paper and then wrote the VHDL code accordingly. I had to use 26 LEDs for the design. I connected every LED to its respective place and printed out an A4 to write the words. I cut the paper into strips and glued them onto the board. I then made the necessary connections between the LEDs and also connected the Arduino Nano behind it. The circuit then reached its final state and it is shown below.

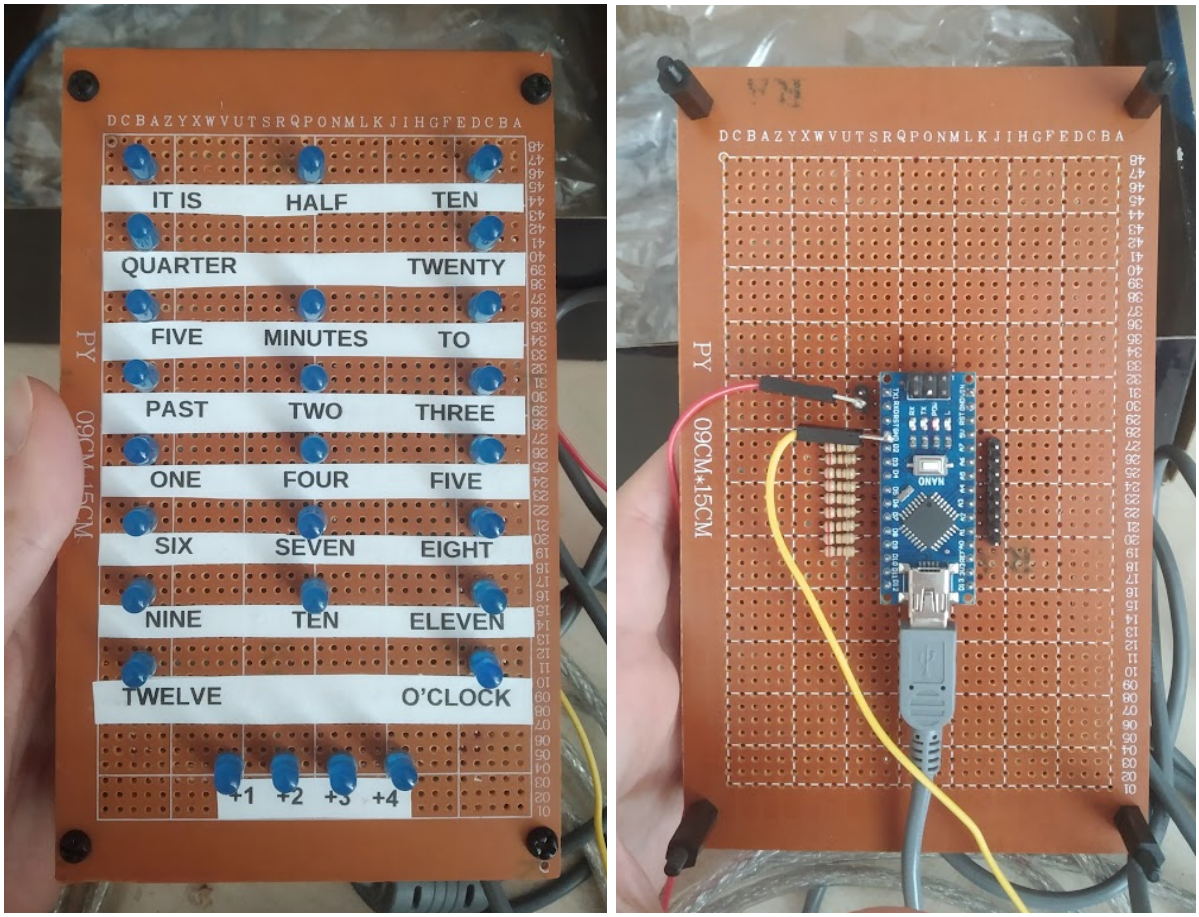

Figure 7 and 8: The Front and Back View of the Circuit

After the circuit was complete, it was now time to establish its connection with the Basys3 board. I did by connecting their GNDs to each other. I then connected the transmitter port (A17) of the Basys3 to the receiver (RX) of the Arduino. I did not do anything to receive anything from the Arduino though, because I did not think it was necessary. Normally, the Arduino should have been sending the Basys3 a ”done” signal, indicating that it has received the previously sent data. However, I did not need the UART transmission to be perfect anyway, since the circuit was not going to display seconds. Even if there was an error during the transmission at one point, it would be fixed with the next data sent. That is the reason why there are only two cables. The connection between the Basys3 and the Arduino is shown below.

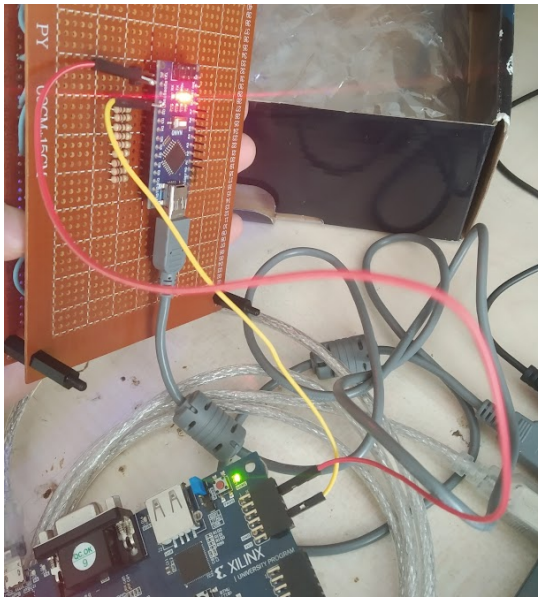

Figure 9: The Connection Made Between Basys3 and Arduino

After the connection was made, I started to test whether the UART code was working properly or not. For this, I wrote a simple code which would print the values that the Arduino received to the serial monitor. By looking at the serial monitor, I was able to determine whether the transmission was done correctly or not. The code for this test can be found in the Appendix part.
Since the transmission was working properly, all there was left to do was to test whether the buttons on the Basys3 were working. To test this, I have generated the bitstream and programmed my Basys3.

On my Basys3, I first tried to reset the output.

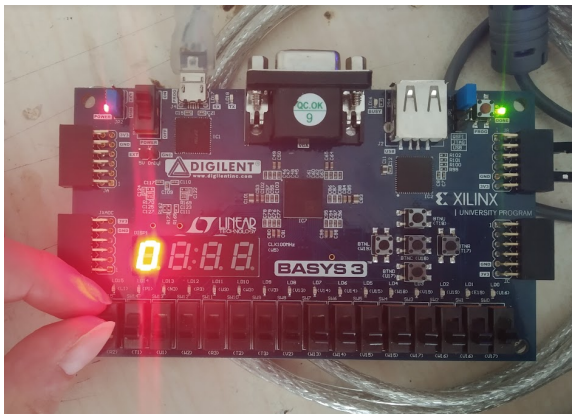

Figure 10: Resetting the Basys3

After the resetting process was successful, I then tried the other two buttons, which were supposed to add 60 and 3600 to the number that was being counted. These represent adding one minute and one hour respectively. I tried to take pictures of the adding process to include in this report. However, it should be considered that I was not fast enough while pressing the buttons and taking the pictures thus there is a difference of around 5 seconds in the pictures, either a little early or a little late.

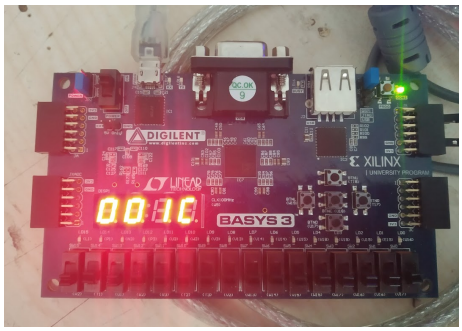

Figure 11: 28 in Hexadecimal

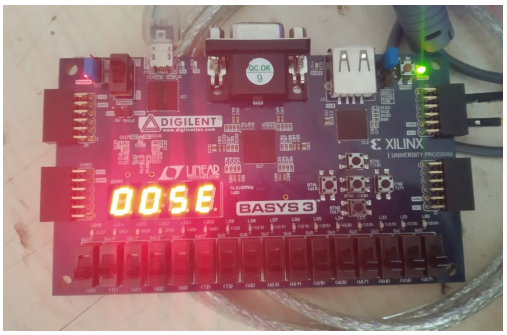

Figure 12: 93 in Hexadecimal

The above figures, Figure 11 and 12, represent adding a minute. The figures below are to demonstrate adding an hour.

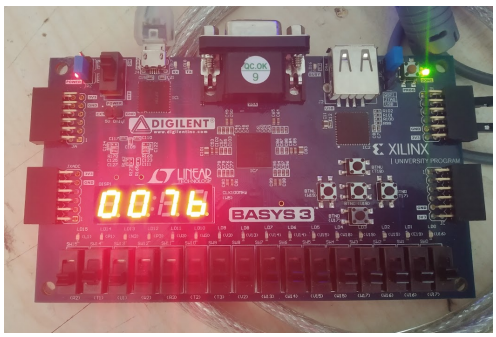

Figure 13: 123 in Hexadecimal

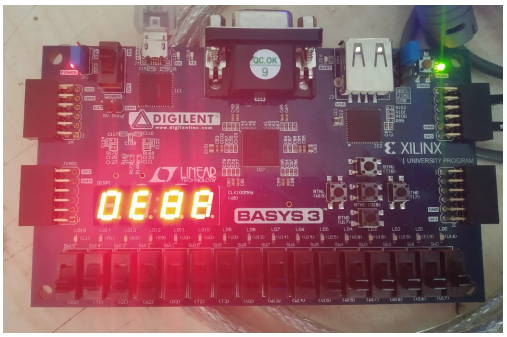

Figure 14: 3720 in Hexadecimal

After these tests, I was sure both the Basys3 and the Arduino were working correctly. Hence, I made connections between them and observed the working word clock by setting the time to a random value.

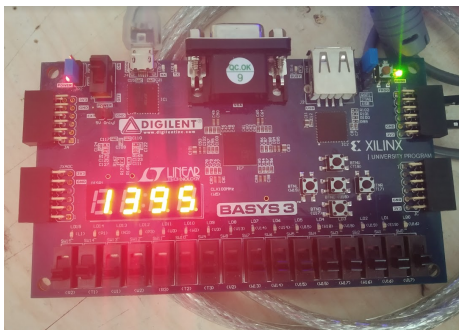

Figure 15: Time on Basys3

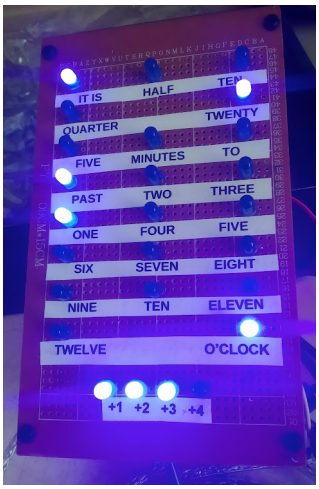

Figure 16: Time Observed in the Circuit

Conclusions

For this project, I have tried to design a Word Clock which had a UART transmission functionality. This clock also had buttons on it which helps set the time, since it does not operate with a real-time clock. The code writing process on the Basys3 went smoothly until I had to implement the UART code. It took me around two weeks to write an actual working code by myself and made me stress out a lot since I thought it was going to fail in the end. The main problem was due to the ”start” and ”end” signals. Due to some reason that I still am having difficulty understanding, these signals were not transmitted correctly. Hence, the Arduino either only received 0’s or 1’s only, resulting in unexpected values on my circuit. That was when I decided to write a separate code for testing the UART transmission. By using that code’s serial monitor and with lots of trial and error, I was able to overcome this issue. Another problem that I faced which was not as significant as the aforementioned one was about the LEDs. I am guessing that since the LEDs were relatively cheaper than what one would expect, they were also low in quality. Because of this, a few of them exploded and/or burned. Besides the explosions, I was not able to determine the burnt ones. This resulted in me thinking that there was something wrong with the code that I wrote, whereas the only problem was with the LEDs. When I realized I was going to keep facing a similar issue, I wrote a small matrix test code which lit up the LEDs line by line. By doing this, I was able to determine whether it was the code that was malfunctioning or the LEDs. Overall, this project was a fun one. I believe it could be improved in two different way. One is my separating the main module into separate smaller modules. The other is by adding a real-time clock to the design, but that is an idea for another time.

Appendix

hexCount.vhd

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

entity hexCount is
    Port (
        clk : in std_logic; -- 100Mhz clock
        reset : in std_logic; -- reset
        add1min: in std_logic; -- add 60
        add1hr: in std_logic; -- add 3600
        anod_a : out std_logic_vector (3 downto 0); -- 4 anodes
        led_o : out std_logic_vector (6 downto 0); -- cathode patterns
        tx_o : out std_logic; -- UART tx output
        tx_tick_o : out std_logic -- UART done tick
    );
end hexCount;

architecture Behavioral of hexCount is
    signal one_sec_count: UNSIGNED (27 downto 0) := (others => '0');
    signal one_sec_enable: std_logic := '0';
    signal disp_num: UNSIGNED (15 downto 0) := (others => '0');
    signal led_dis: std_logic_vector (3 downto 0);
    signal ffive_count: UNSIGNED (19 downto 0) := (others => '0');
    signal led_act_count: std_logic_vector(1 downto 0);
```

```
    -- UART signals
    signal tx_start_i : std_logic := '0';
    signal tx_data : std_logic_vector(15 downto 0) := (others => '0');
    signal tx_done_tick : std_logic;

    -- state machine for UART tx
    type uart_state_type is (IDLE, SEND_DIGIT1234);
    signal uart_state : uart_state_type := IDLE;
    signal digit_counter : integer range 0 to 1 := 0;

begin

-- uart_tx def
uart_tx_inst : entity work.uart_tx
    generic map (
        c_clkfreq => 100_000_000,
        c_baudrate => 9_600,
        c_stopbit => 1
    )
    port map (
        clk => clk,
        din_i => tx_data,
        tx_start_i => tx_start_i,
        tx_stop => reset,
        tx_o => tx_o,
```

```
        tx_tick_o => tx_done_tick
    );


-- patterns for 7-segment display
process(led_dis)
begin
    case led_dis is
    when "0000" => led_o <= "0000001"; -- "0"
    when "0001" => led_o <= "1001111"; -- "1"
    when "0010" => led_o <= "0010010"; -- "2"
    when "0011" => led_o <= "0000110"; -- "3"
    when "0100" => led_o <= "1001100"; -- "4"
    when "0101" => led_o <= "0100100"; -- "5"
    when "0110" => led_o <= "0100000"; -- "6"
    when "0111" => led_o <= "0001111"; -- "7"
    when "1000" => led_o <= "0000000"; -- "8"
    when "1001" => led_o <= "0000100"; -- "9"
    when "1010" => led_o <= "0000010"; -- "a"
    when "1011" => led_o <= "1100000"; -- "b"
    when "1100" => led_o <= "0110001"; -- "C"
    when "1101" => led_o <= "1000010"; -- "d"
    when "1110" => led_o <= "0110000"; -- "E"
    when "1111" => led_o <= "0111000"; -- "F"
    when others => led_o <= "0000000";
    end case;
```

```
end process;


-- 7-segment display controller with refresh period of 10.5ms
process(clk,reset)
begin
    if(reset='1') then
        ffive_count <= (others => '0');
    elsif(rising_edge(clk)) then
        ffive_count <= ffive_count + 1;
    end if;
end process;


led_act_count <= std_logic_vector(ffive_count(19 downto 18));


-- 4-to-1 mux for anode activation
process(led_act_count, disp_num)
begin
    case led_act_count is
    when "00" =>
        anod_a <= "0111";
        led_dis <= std_logic_vector(disp_num(15 downto 12)); -- digit 1
    when "01" =>
        anod_a <= "1011";
        led_dis <= std_logic_vector(disp_num(11 downto 8)); -- digit 2
    when "10" =>
```

```
        anod_a <= "1101";
        led_dis <= std_logic_vector(disp_num(7 downto 4)); -- digit 3
    when "11" =>
        anod_a <= "1110";
        led_dis <= std_logic_vector(disp_num(3 downto 0)); -- digit 4
    when others =>
        anod_a <= "0000";
    end case;
end process;

-- counting to be display on 7-segment display
process(clk, reset)
begin
    if reset = '1' then
        disp_num <= (others => '0');
        one_sec_count <= (others => '0');
        tx_start_i <= '0';
        uart_state <= IDLE;
        digit_counter <= 0;
    elsif rising_edge(clk) then
        if disp_num = x"A8C0" then
            disp_num <= (others => '0');
        end if;
        if one_sec_count >= x"5F5E0FF" then
            one_sec_count <= (others => '0');
```

```
    if add1min = '1' then
        disp_num <= disp_num + to_unsigned(60, 16);
    elsif add1hr = '1' then
        disp_num <= disp_num + to_unsigned(3600, 16);
    else
        disp_num <= disp_num + 1;
    end if;

    -- Start UART transmission
    tx_start_i <= '1';
else
    one_sec_count <= one_sec_count + 1;
end if;

-- UART state machine handling
case uart_state is
    when IDLE =>
        if tx_start_i = '1' then
            tx_data <= std_logic_vector(disp_num(15 downto 0));
            uart_state <= SEND_DIGIT1234;
            tx_start_i <= '0';
        end if;

    when SEND_DIGIT1234 =>
        if tx_done_tick = '1' then
```

```
                    uart_state <= IDLE;
                    tx_start_i <= '0';
                end if;

            when others =>
                uart_state <= IDLE;

        end case;
    end if;
end process;
end Behavioral;
```

uart_tx.vhd

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

entity uart_tx is
    generic (
        c_clkfreq : integer := 100_000_000; -- clock frequency in Hz
        c_baudrate : integer := 9_600;    -- baud rate
        c_stopbit : integer := 1            -- number of stop bits
    );
    port (
        clk : in std_logic;        -- clock input
        din_i : in std_logic_vector(15 downto 0); -- data input
        tx_start_i : in std_logic;          -- transmit start signal
```

```
        tx_stop : in std_logic;             -- reset signal
        tx_o : out std_logic;               -- UART transmit output
        tx_tick_o : out std_logic      -- transmit done tick
    );
end uart_tx;


architecture Behavioral of uart_tx is
    signal baud_counter : integer := 0;
    signal bit_counter : integer range 0 to 18 := 0;
    signal tx_reg : std_logic := '1';
    signal tx_done_tick : std_logic := '0';
    signal shift_reg : std_logic_vector(15 downto 0);
begin
    process(clk)
    begin
        if rising_edge(clk) then
            if tx_stop = '1' then
                tx_reg <= '1';
                bit_counter <= 0;
                baud_counter <= 0;
                tx_done_tick <= '0';
            elsif tx_start_i = '1' and bit_counter = 0 then
                shift_reg <= din_i;
                bit_counter <= 1;
                baud_counter <= 0;
```

```
elsif bit_counter > 0 then
    if baud_counter >= (c_clkfreq / c_baudrate) - 1 then
        baud_counter <= 0;
        case bit_counter is
            when 1 =>
                tx_reg <= '0'; -- start bit
            when 2 to 17 =>
                tx_reg <= shift_reg(bit_counter - 2);
            when 18 =>
                tx_reg <= '1'; -- stop bit
                if c_stopbit = 2 then
                    bit_counter <= bit_counter + 1;
                else
                    bit_counter <= 0;
                    tx_done_tick <= '1';
                end if;
            when others =>
                bit_counter <= 0;
        end case;
        if bit_counter > 0 then
            bit_counter <= bit_counter + 1;
        end if;
    else
        baud_counter <= baud_counter + 1;
    end if;
```

```
            end if;
        end if;
    end process;
    tx_o <= tx_reg;
    tx_tick_o <= tx_done_tick;
end Behavioral;
```

const.xdc

```
# Clock signal
set_property PACKAGE_PIN W5 [get_ports clk]
 set_property IOSTANDARD LVCMOS33 [get_ports clk]
set_property PACKAGE_PIN R2 [get_ports reset]
 set_property IOSTANDARD LVCMOS33 [get_ports reset]

#seven-segment LED display
set_property PACKAGE_PIN W7 [get_ports {led_o[6]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {led_o[6]}]
set_property PACKAGE_PIN W6 [get_ports {led_o[5]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {led_o[5]}]
set_property PACKAGE_PIN U8 [get_ports {led_o[4]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {led_o[4]}]
set_property PACKAGE_PIN V8 [get_ports {led_o[3]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {led_o[3]}]
set_property PACKAGE_PIN U5 [get_ports {led_o[2]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {led_o[2]}]
set_property PACKAGE_PIN V5 [get_ports {led_o[1]}]
```

```
 set_property IOSTANDARD LVCMOS33 [get_ports {led_o[1]}]
set_property PACKAGE_PIN U7 [get_ports {led_o[0]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {led_o[0]}]
set_property PACKAGE_PIN U2 [get_ports {anod_a[0]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {anod_a[0]}]
set_property PACKAGE_PIN U4 [get_ports {anod_a[1]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {anod_a[1]}]
set_property PACKAGE_PIN V4 [get_ports {anod_a[2]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {anod_a[2]}]
set_property PACKAGE_PIN W4 [get_ports {anod_a[3]}]
 set_property IOSTANDARD LVCMOS33 [get_ports {anod_a[3]}]

#buttons
set_property PACKAGE_PIN W19 [get_ports {add1min}]
 set_property IOSTANDARD LVCMOS33 [get_ports {add1min}]
set_property PACKAGE_PIN T17 [get_ports {add1hr}]
 set_property IOSTANDARD LVCMOS33 [get_ports {add1hr}]

#uart stuff
set_property PACKAGE_PIN A17 [get_ports {tx_o}]
 set_property IOSTANDARD LVCMOS33 [get_ports {tx_o}]
set_property PACKAGE_PIN A14 [get_ports {tx_tick_o}]
 set_property IOSTANDARD LVCMOS33 [get_ports {tx_tick_o}]
```

Matris_Test.ino

```
const int NoOfLEDOutputs = 12;

// first 9 are rows; 11,12 and 13 are columns
int LEDOutputs[NoOfLEDOutputs] = {2,3,4,5,6,7,8,9,10,11,12,13};

const int MyDelay = 1000;

void setup()
{
pinMode (2,OUTPUT);

pinMode (3,OUTPUT);

pinMode (4,OUTPUT);

pinMode (5,OUTPUT);

pinMode (6,OUTPUT);

pinMode (7,OUTPUT);

pinMode (8,OUTPUT);

pinMode (9,OUTPUT);

pinMode (10,OUTPUT);

pinMode (11,OUTPUT);

pinMode (12,OUTPUT);

pinMode (13,OUTPUT);
}


void loop() {

digitalWrite(11,LOW);

digitalWrite(12,LOW);

digitalWrite(13,LOW);



digitalWrite (2, HIGH);

delay (MyDelay);
```

```
digitalWrite (2,LOW);

delay(MyDelay);

digitalWrite (3, HIGH);

delay (MyDelay);

digitalWrite (3,LOW);

delay(MyDelay);

digitalWrite (4, HIGH);

delay (MyDelay);

digitalWrite (4,LOW);

delay(MyDelay);

digitalWrite (5, HIGH);

delay (MyDelay);

digitalWrite (5,LOW);

delay(MyDelay);

digitalWrite (6, HIGH);

delay (MyDelay);

digitalWrite (6,LOW);

delay(MyDelay);

digitalWrite (7, HIGH);

delay (MyDelay);

digitalWrite (7,LOW);

delay(MyDelay);

digitalWrite (8, HIGH);

delay (MyDelay);

digitalWrite (8,LOW);
```

```
delay(MyDelay);

digitalWrite (9, HIGH);

delay (MyDelay);

digitalWrite (9,LOW);

delay(MyDelay);

digitalWrite (10, HIGH);

delay (MyDelay);

digitalWrite (10,LOW);

delay(MyDelay);



digitalWrite(11,HIGH);

digitalWrite(12,HIGH);

digitalWrite(13,HIGH);
}
```

ReceiveFromFPGA.ino

```
// pin to the TX pin of FPGA
#define RX_PIN 2

// the baud rate in FPGA
#define BAUD_RATE 9600


void setup() {

  Serial.begin(BAUD_RATE);

  pinMode(RX_PIN, INPUT);
}


void loop() {

  if (Serial.available() >= 2) {

    int byte1 = Serial.read();
```

```
    int byte2 = Serial.read();

    Serial.print("Received bytes: ");

    Serial.print(byte1);

    Serial.print(", ");

    Serial.println(byte2);
  }
}
```

WordClock.ino

```
int receivedValue;

const int NoOfLEDOutputs = 12;
// 2,3,4,5,6,7,8,9,10,11,12,13 : First 9 are rows; 11,12, and 13 are columns

//             (COL 1)      (COL 2)          (COL 3)
// (ROW 1) 1   IT IS        2 HALF           3 TEN
// (ROW 2) 4   QUARTER      5 (EMPTY)        6 TWENTY
// (ROW 3) 7   FIVE         8 MINUTES        9 TO
// (ROW 4) 10  PAST        11 TWO           12 THREE
// (ROW 5) 13  ONE         14 FOUR          15 FIVE
// (ROW 6) 16  SIX         17 SEVEN         18 EIGHT
// (ROW 7) 19  NINE        20 TEN           21 ELEVEN
// (ROW 8) 22  TWELVE      23 (+2 MIN)      24 O'CLOCK
// (ROW 9) 25  (+1 MIN)    26 (+3 MIN)      27 (+4 MIN)

int Hour = 0;
int Minute = 0;
long int MyCounter = 0;

void setup()
{

  // baud rate to match FPGA
  Serial.begin(9600);

  // output ports
  for (int i=2; i<14;i++) pinMode(i, OUTPUT);
}

void TurnOffAllLEDs()
{

    // turn off rows
    for (int i=0; i<11;i++) digitalWrite (i,LOW);
```

```
    // turn off columns
    for (int i=11; i<14;i++) digitalWrite (i,HIGH);
}


void ReadFPGAData()
{
  if (Serial.available() >= 2) {
      long receivedValue = Serial.read();}
    delay(5);
    Hour = receivedValue / 3600;
    Minute = (receivedValue - (Hour * 3600)) / 60;
}

// function to turn on the first column
void LEDColumn1()
{
  // LEDs off before next cycle
  TurnOffAllLEDs();

  // activate column 1
  digitalWrite (11,LOW);

  // turn on "IT IS"
  digitalWrite (2,HIGH);


  // turn on "QUARTER"
  if (Minute >= 15 && Minute < 20) digitalWrite (3,HIGH);

  if (Minute >= 45 && Minute < 50) digitalWrite (3,HIGH);

  // turn on "FIVE"
  if (Minute >= 5 && Minute < 10)  digitalWrite (4,HIGH);

  if (Minute >= 55 && Minute <= 59)  digitalWrite (4,HIGH);

  if (Minute >= 25 && Minute < 30)  digitalWrite (4,HIGH);

  if (Minute > 34 && Minute < 40)  digitalWrite (4,HIGH);


  if (Minute <= 34)
  {
       // turn on "PAST"
       if (Minute > 4) digitalWrite (5,HIGH);

       // turn on "ONE"
       if (Hour == 1) digitalWrite (6,HIGH);
```

```
     // turn on "SIX"
     if (Hour == 6) digitalWrite (7,HIGH);

     // turn on "NINE"
     if (Hour == 9) digitalWrite (8,HIGH);

     // turn on "TWELVE"
     if (Hour == 12 || Hour == 0) digitalWrite (9,HIGH);
  };
}

  if (Minute > 34)

{
     // turn on "ONE"
     if (Hour == 0 || Hour== 12) digitalWrite (6,HIGH);

     // turn on "SIX"
     if (Hour == 5) digitalWrite (7,HIGH);

     // turn on "NINE"
     if (Hour == 8) digitalWrite (8,HIGH);

     // turn on "TWELVE"
     if (Hour == 11) digitalWrite (9,HIGH);
     };
}

// turn on (+1)
if (Minute == 1 || Minute == 11 || Minute == 21 || Minute == 31
|| Minute == 41 || Minute == 51) digitalWrite (10,HIGH);

if (Minute == 6 || Minute == 16 || Minute == 26 || Minute == 36
|| Minute == 46 || Minute == 56) digitalWrite (10,HIGH);

// turn on (+1) (+2)
if (Minute == 2 || Minute == 12 || Minute == 22 || Minute == 32
|| Minute == 42 || Minute == 52) digitalWrite (10,HIGH);

if (Minute == 7 || Minute == 17 || Minute == 27 || Minute == 37
|| Minute == 47 || Minute == 57) digitalWrite (10,HIGH);

// turn on (+1) (+2) (+3)
if (Minute == 3 || Minute == 13 || Minute == 23 || Minute == 33
|| Minute == 43 || Minute == 53) digitalWrite (10,HIGH);

if (Minute == 8 || Minute == 18 || Minute == 28 || Minute == 38
|| Minute == 48 || Minute == 58) digitalWrite (10,HIGH);
```

```
  // turn on (+1) (+2) (+3) (+4)
  if (Minute == 4 || Minute == 14 || Minute == 24 || Minute == 34
  || Minute == 44 || Minute == 54) digitalWrite (10,HIGH);

  if (Minute == 9 || Minute == 19 || Minute == 29 || Minute == 39
  || Minute == 49 || Minute == 59) digitalWrite (10,HIGH);

}

// function to turn on the second column
void LEDColumn2()
{
    // LEDs off before next cycle
    TurnOffAllLEDs();

    // Activate LED Column 2
    digitalWrite (12,LOW);

  // turn on "HALF"
  if (Minute >= 30 && Minute < 35) digitalWrite (2,HIGH);

  if (Minute < 30)
  {
       // turn on "TWO"
       if (Hour == 2) digitalWrite (5,HIGH);

       // turn on "FOUR"
       if (Hour == 4) digitalWrite (6,HIGH);

       // turn on "SEVEN"
       if (Hour == 7) digitalWrite (7,HIGH);

       // turn on "TEN"
       if (Hour == 10) digitalWrite (8,HIGH);
    }
  }
  if (Minute > 34)
  {
       // turn on "TWO"
       if (Hour == 1) digitalWrite (5,HIGH);

       // turn on "FOUR"
       if (Hour == 3) digitalWrite (6,HIGH);

       // turn on "SEVEN"
       if (Hour == 6) digitalWrite (7,HIGH);

       // turn on "TEN"
       if (Hour == 9) digitalWrite (8,HIGH);
       }
```

```
     // turn on "MINUTES" except when "QUARTER" is on
     if (Minute < 45 || Minute > 49) digitalWrite (4,HIGH);

}

// turn on (+2)
if (Minute == 2 || Minute == 12 || Minute == 22 || Minute == 32
|| Minute == 42 || Minute == 52) digitalWrite (9,HIGH);

if (Minute == 7 || Minute == 17 || Minute == 27 || Minute == 37
|| Minute == 47 || Minute == 57) digitalWrite (9,HIGH);


if (Minute == 3 || Minute == 13 || Minute == 23 || Minute == 33
|| Minute == 43 || Minute == 53)

{
    // turn on (+2) for (+3)
    digitalWrite (9,HIGH);

    // turn on (+3) for (+3)
    digitalWrite (10,HIGH);}


if (Minute == 8 || Minute == 18 || Minute == 28 || Minute == 38
|| Minute == 48 || Minute == 58)

{
    // turn on (+2) for (+3)
    digitalWrite (9,HIGH);

    // turn on (+3) for (+3)
    digitalWrite (10,HIGH);}


if (Minute == 4 || Minute == 14 || Minute == 24 || Minute == 34
|| Minute == 44 || Minute == 54)

  {
    // turn on (+2) for (+4)
    digitalWrite (9,HIGH);

    // turn on (+3) for (+4)
    digitalWrite (10,HIGH);}

if (Minute == 9 || Minute == 19 || Minute == 29 || Minute == 39
|| Minute == 49 || Minute == 59)
  {
    // turn on (+2) for (+4)
```

```
      digitalWrite (9,HIGH);

      // turn on (+3) for (+4)
      digitalWrite (10,HIGH);}
}


// function to turn on the third column
void LEDColumn3()
{

  // LEDs off before next cycle
  TurnOffAllLEDs();

  // activate column 3
  digitalWrite (13,LOW);;

  // turn on "O'CLOCK"
  digitalWrite (9,HIGH);;


  // turn on "TEN"
  if (Minute >= 10 && Minute < 15) digitalWrite (2,HIGH);

  // turn on "TEN"
  if (Minute >= 50 && Minute < 55) digitalWrite (2,HIGH);

  // turn on "TWENTY"
  if (Minute > 19 && Minute < 30) digitalWrite (3,HIGH);

  // turn on "TWENTY-SOMETHING"
  if (Minute > 34 && Minute < 45) digitalWrite (3,HIGH);

  if (Minute > 34)
  {
       // turn on "TO"
       digitalWrite (4,HIGH);

       // turn on "THREE"
       if (Hour == 2) digitalWrite (5,HIGH);;

       // turn on "FIVE"
       if (Hour == 4) digitalWrite (6,HIGH);;

       // turn on "EIGHT"
       if (Hour == 7) digitalWrite (7,HIGH);;

       // turn on "ELEVEN"};
       if (Hour == 10) digitalWrite (8,HIGH);;
  }
```

```
  if (Minute < 30)
  {
       // turn on "THREE"
       if (Hour == 3) digitalWrite (5,HIGH);;

       // turn on "FIVE"
       if (Hour == 5) digitalWrite (6,HIGH);;

       // turn on "EIGHT"
       if (Hour == 8) digitalWrite (7,HIGH);;

       // turn on "ELEVEN"
       if (Hour == 11) digitalWrite (8,HIGH);;   };
  }

  // turn on (+4)
  if (Minute == 4 || Minute == 14 || Minute == 24 || Minute == 34
  || Minute == 44 || Minute == 54) digitalWrite (10,HIGH);

  if (Minute == 9 || Minute == 19 || Minute == 29 || Minute == 39
  || Minute == 49 || Minute == 59) digitalWrite (10,HIGH);

} // END Function LEDColumn3

void loop ()
{ ReadFPGAData();
  LEDColumn1();
  LEDColumn2();
  LEDColumn3();
}
```