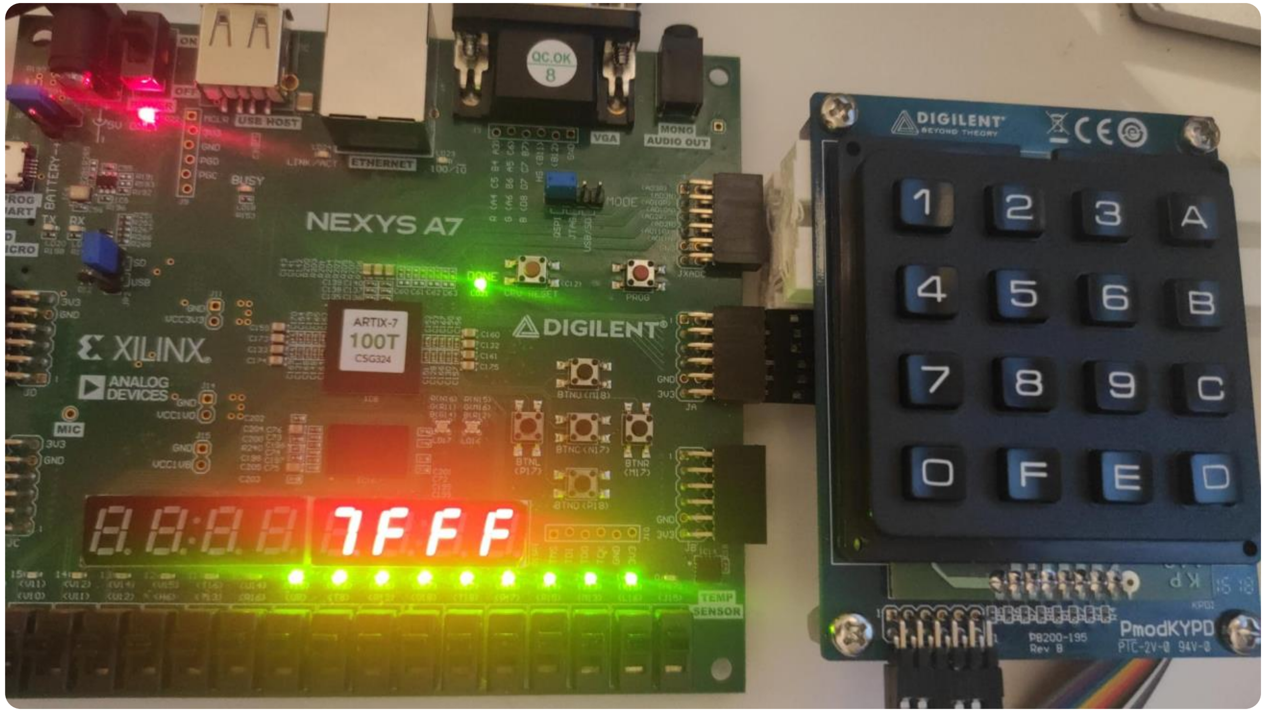
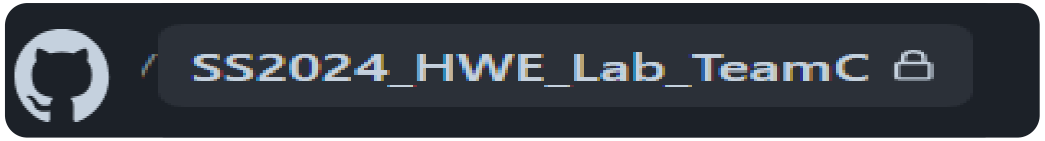
Group C

HOCHSCHULE
HAMM-LIPPSTADT



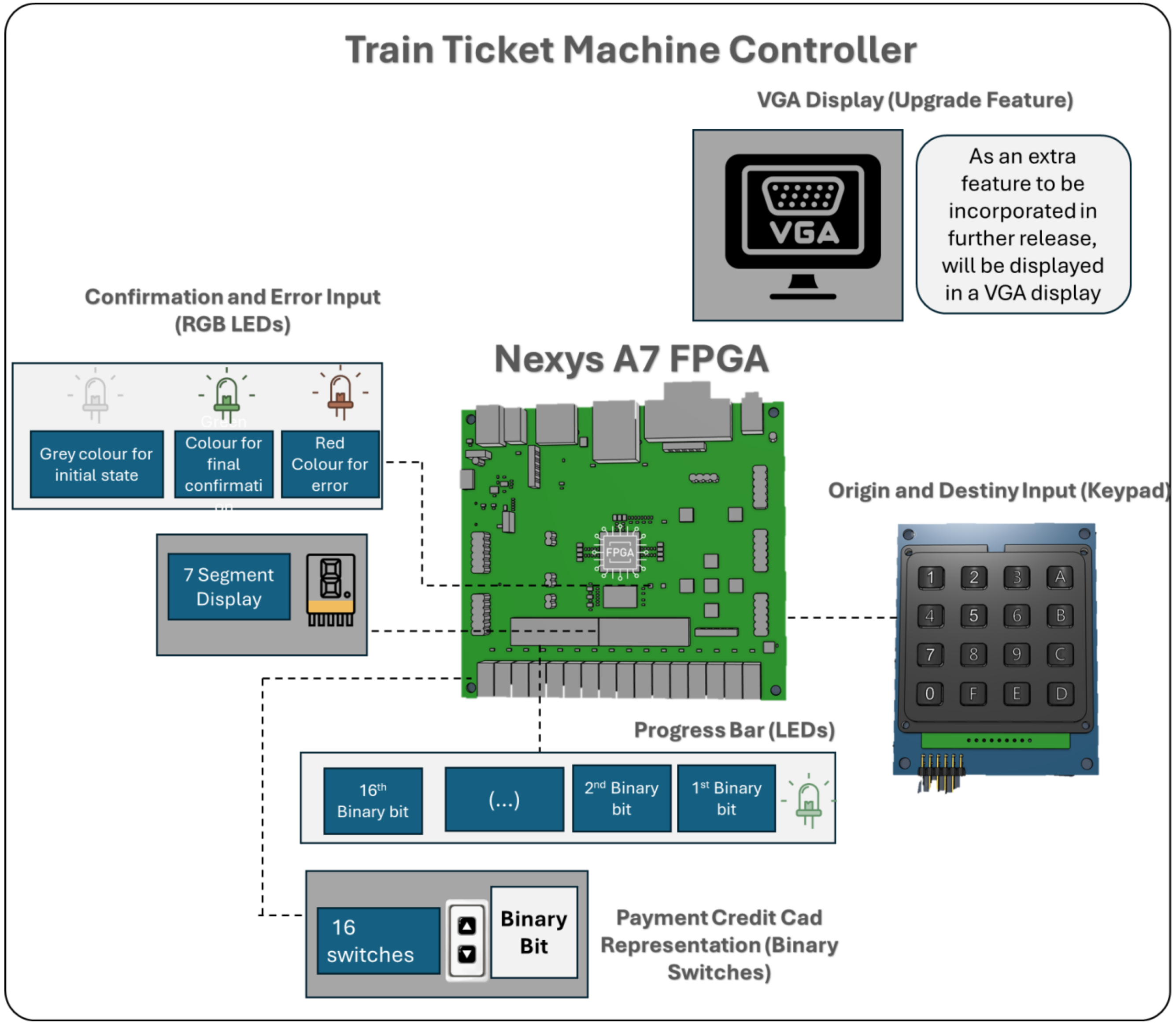


/ SS2024_HWE_Lab_TeamC

Teammates (alphab.):
- Akram Md ,
- Choi Younsuk
- Richard Jimenez

Module: Hardware Engineering– SuSe24
Examiners: Prof. Dr. Hayek
ELE – Hochschule Hamm-Lippstadt
July 2nd 2024

Controller’s Graphical Block Diagram



Train Ticket Machine Controller

VGA Display (Upgrade Feature)

As an extra feature to be incorporated in further release, will be displayed in a VGA display

Confirmation and Error Input (RGB LEDs)

Grey colour for initial state

Colour for final confirmati

Red Colour for error

7 Segment Display

Nexys A7 FPGA

Origin and Destiny Input (Keypad)

Progress Bar (LEDs)

16th Binary bit

(...)

2nd Binary bit

1st Binary bit

16 switches

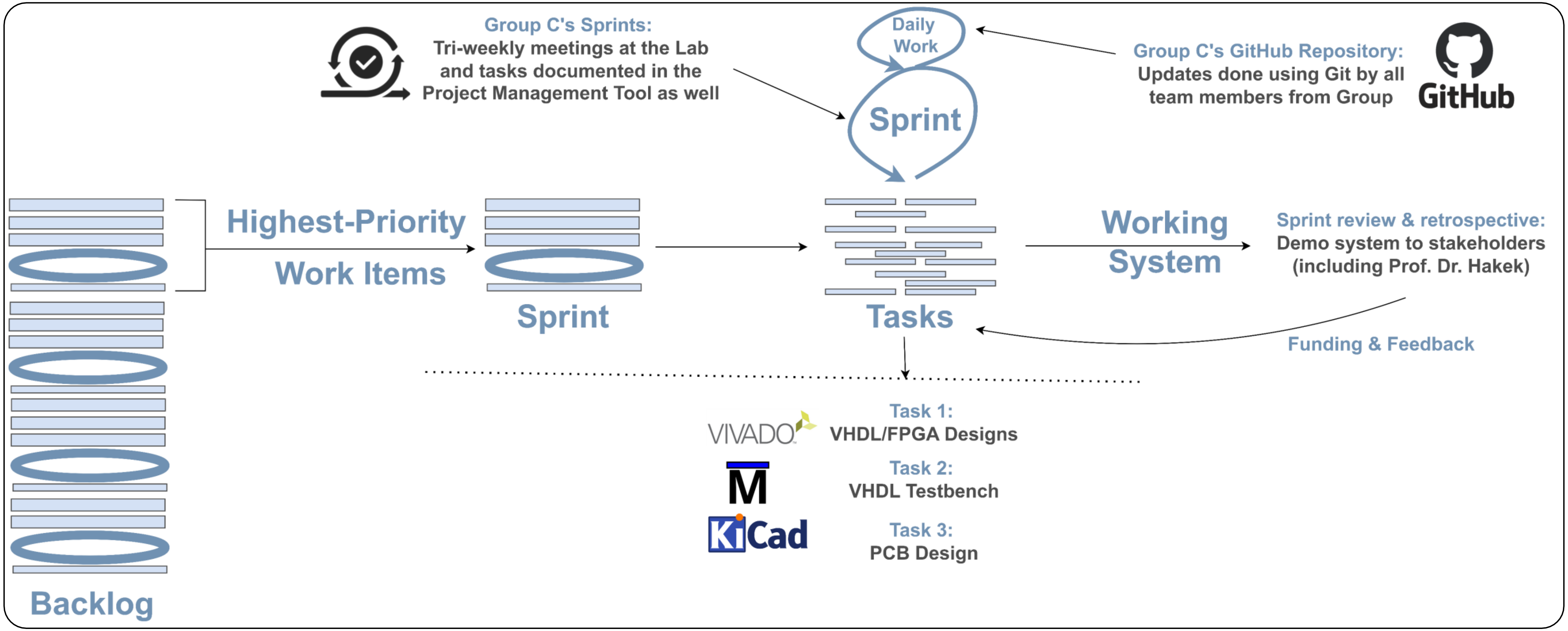
Binary Bit

Payment Credit Cad Representation (Binary Switches)

Source: Own Creation.

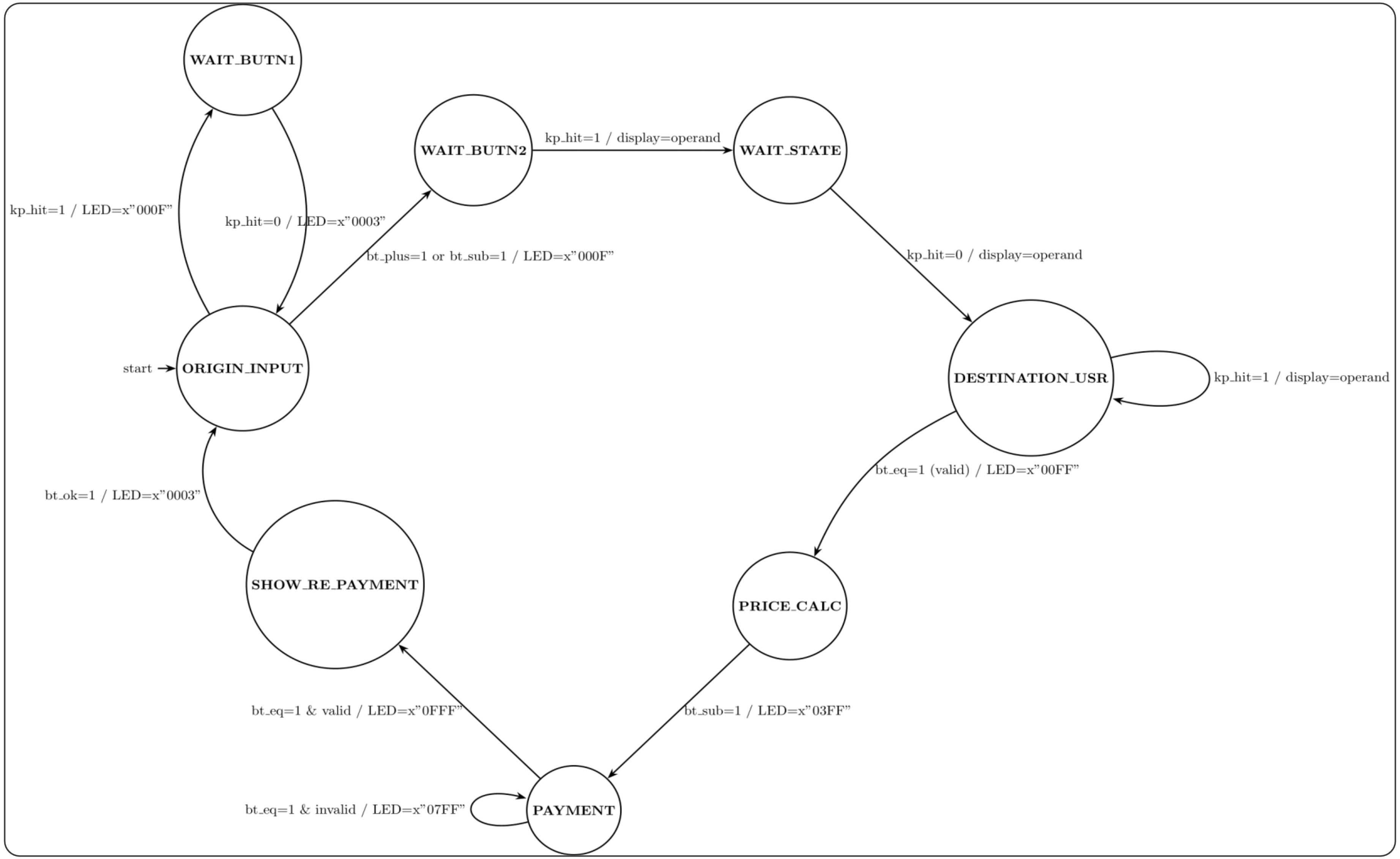
Group C

Controller’s Graphical Block Diagram


Group C's Sprints:
Tri-weekly meetings at the Lab and tasks documented in the Project Management Tool as well
Daily Work
Sprint
Group C's GitHub Repository:
Updates done using Git by all team members from Group
GitHub
Highest-Priority Work Items
Sprint
Tasks
Working System
Sprint review & retrospective:
Demo system to stakeholders (including Prof. Dr. Hakek)
Funding & Feedback
VIVADO
Task 1:
VHDL/FPGA Designs
M
Task 2:
VHDL Testbench
KiCad
Task 3:
PCB Design
Backlog

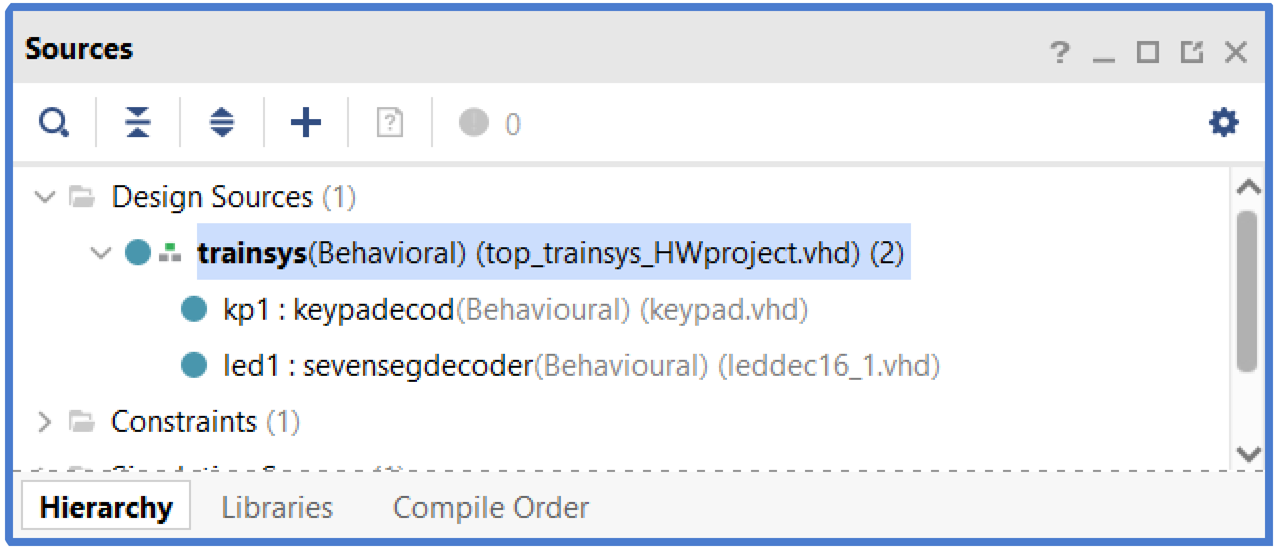
Agile Model-Based Methodology Development. Source: Own Creation.

Controller’s Mealy FSM


WAIT_BUTN1
WAIT_BUTN2
kp_hit=1 / display=operand
WAIT_STATE
kp_hit=1 / LED=x”000F”
kp_hit=0 / LED=x”0003”
bt_plus=1 or bt_sub=1 / LED=x”000F”
kp_hit=0 / display=operand
start
ORIGIN_INPUT
DESTINATION_USR
kp_hit=1 / display=operand
bt_eq=1 (valid) / LED=x”00FF”
bt_ok=1 / LED=x”0003”
SHOW_RE_PAYMENT
PRICE_CALC
bt_eq=1 & valid / LED=x”0FFF”
bt_sub=1 / LED=x”03FF”
bt_eq=1 & invalid / LED=x”07FF”
PAYMENT

Source: Own Creation.

VHDL/FPGA - Tool



Sources

Design Sources (1)
- trainsys(Behavioral) (top_trainsys_HWproject.vhd) (2)
  - kp1 : keypadecod(Behavioural) (keypad.vhd)
  - led1 : sevensegdecoder(Behavioural) (leddec16_1.vhd)

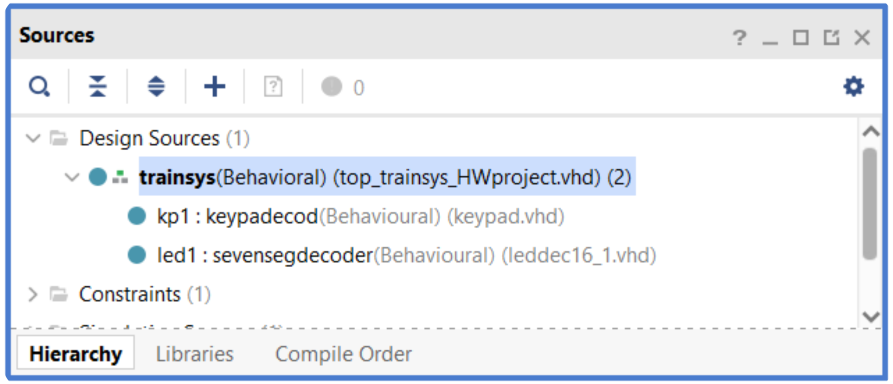
Constraints (1)

VHDL Sources three in Vivado. Source: Own Creation.

Controller's Logical Train Ticket:

- Top file: trainsys (FSM)
- Peripherals: PMOD keypad, 7-Segment Display
- Other peripherals (in top file): LEDs, switches, RGB LEDs, push buttons

VHDL/FPGA – Top File Structure

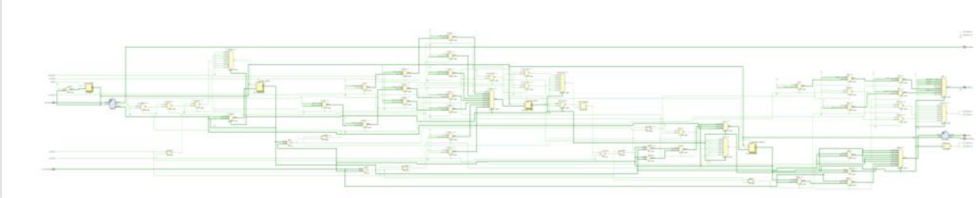


Design Sources (1)

- trainsys(Behavioral) (top_trainsys_HWproject.vhd) (2)
  - kp1 : keypadecod(Behavioural) (keypad.vhd)
  - led1 : sevensegdecoder(Behavioural) (leddec16_1.vhd)

Constraints (1)

VHDL Sources three in Vivado. Source: Own Creation.



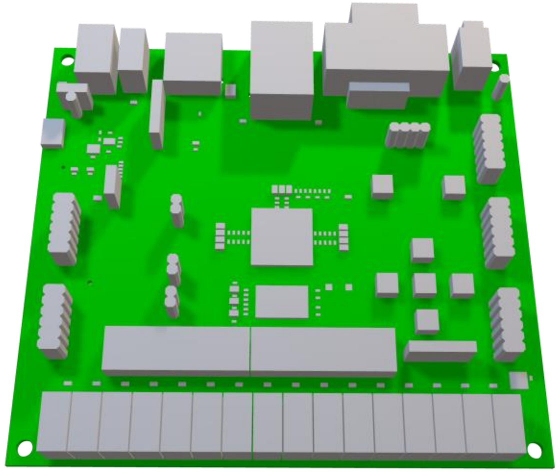
Generated schematics in Vivado. Source: Own Creation.

Controller's Logical Train Ticket:

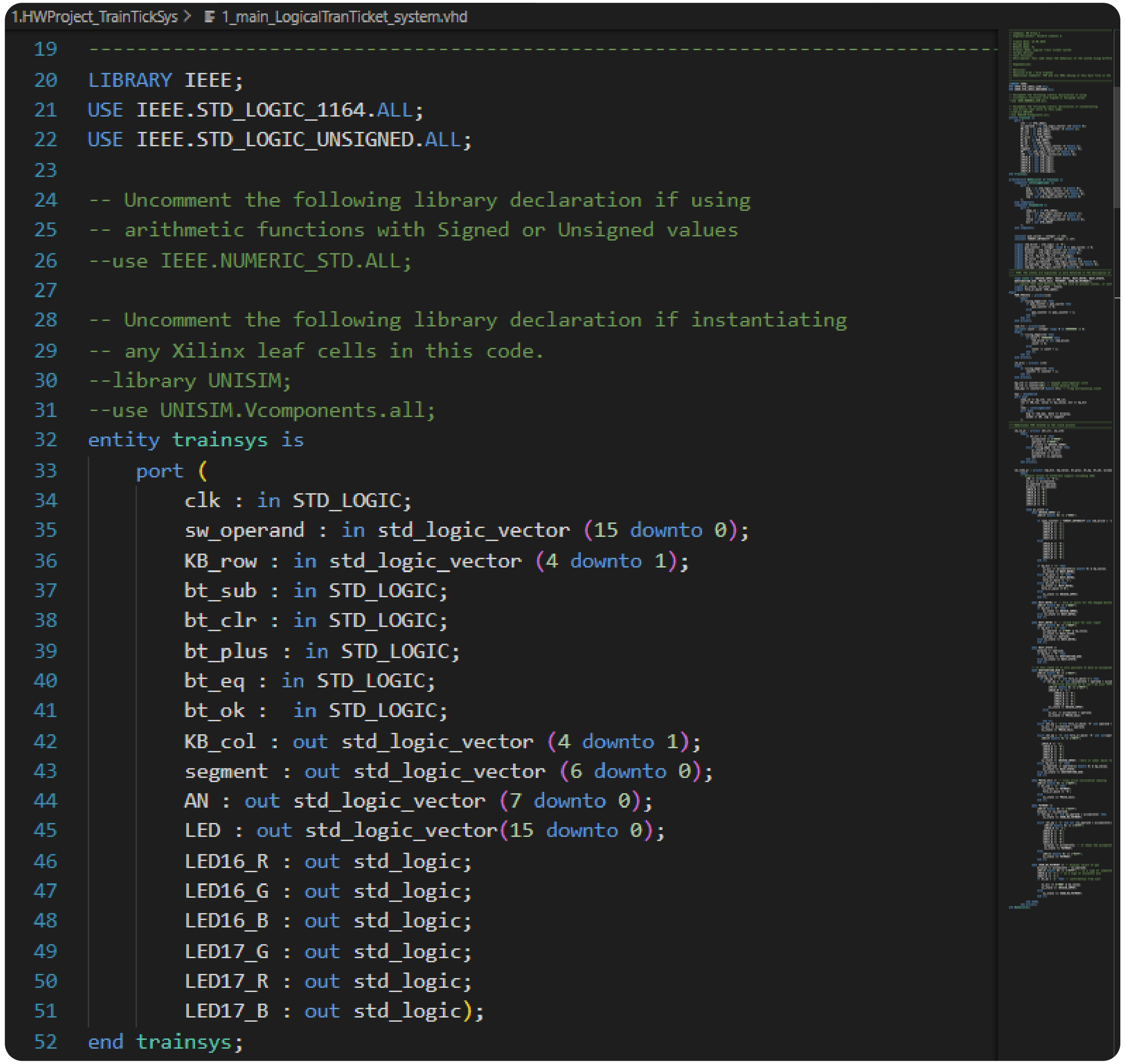
- Components:
  - sevensegdecoder: Decodes data for seven-segment display
  - keypadecoder: Handles keypad input
- Processes:
  - PWM_PROCESS: Manages PWM counter for LED intensity
  - led_fls: Controls LED blinking
  - ck_proc: Increments the main counter
  - sm_ck_pr: Handles state machine clock and clear button
  - sm_comb_pr: Main state machine combinational process
- Clocks:
  - Main clock (clk): Input to the entitykp_clk: Keypad interrogation clock (counter(15))
  - sm_clk: State machine clock (counter(20))
  - led_mpx: 7-segment multiplexing clock (counter(19 downto 17))
- Finite State Machine (FSM): 8 Mealy states

Group C

VHDL/FPGA – Top File Structure



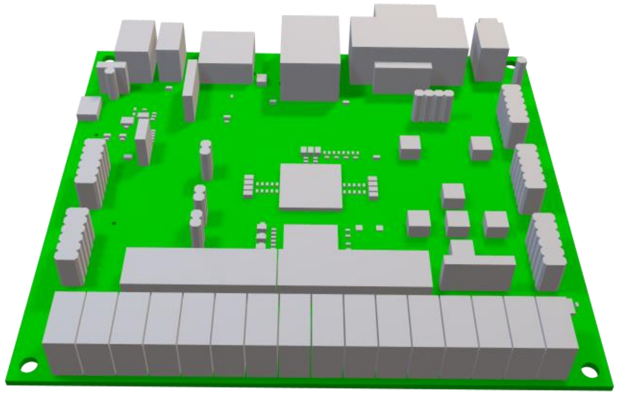
Nexys A7. Source: Diligent.



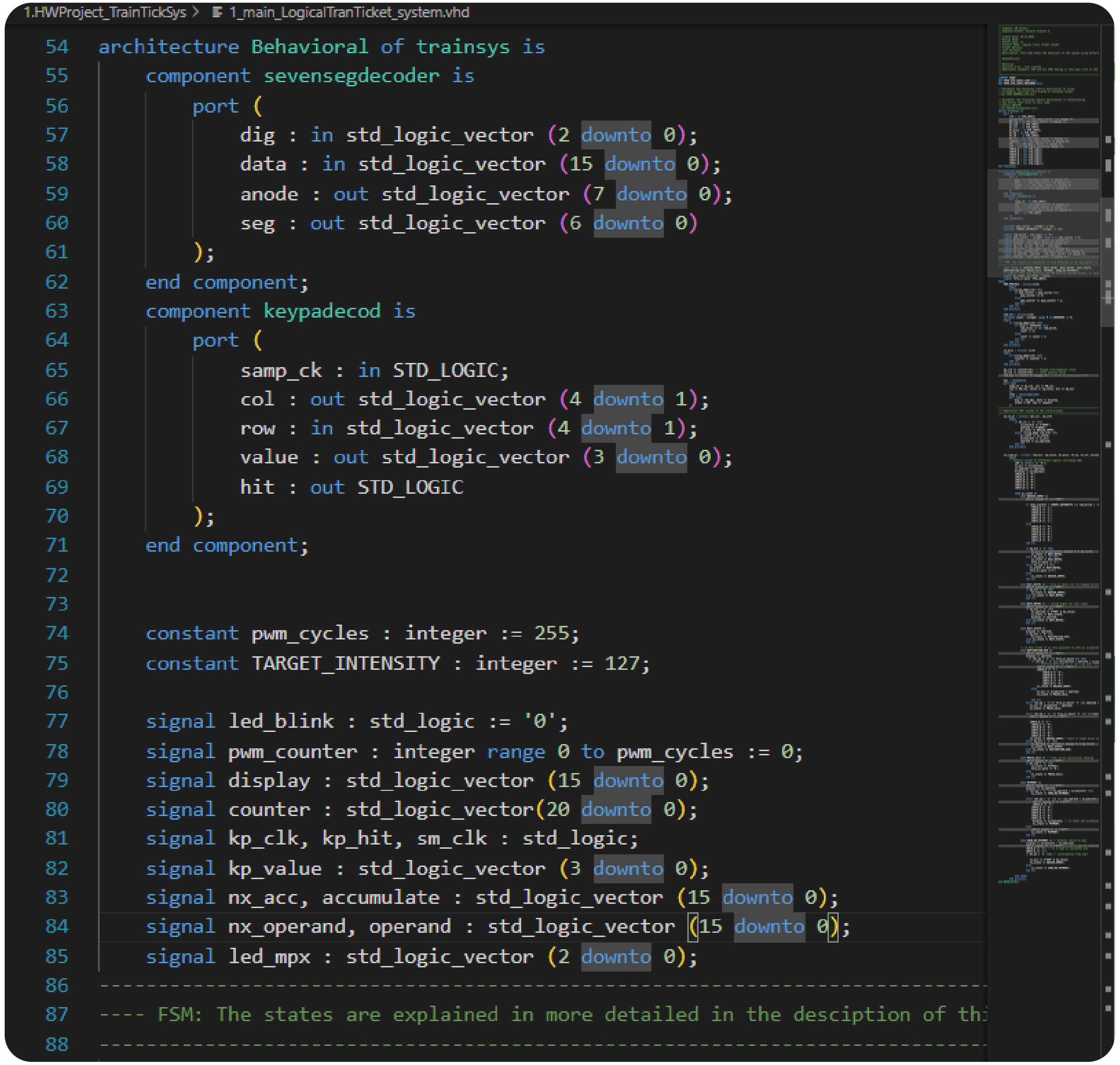
1.HWProject_TrainTickSys > 1_main_LogicalTranTicket_system.vhd

```vhdl
--------------------------------------------------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.Vcomponents.all;
entity trainsys is
    port (
        clk : in STD_LOGIC;
        sw_operand : in std_logic_vector (15 downto 0);
        KB_row : in std_logic_vector (4 downto 1);
        bt_sub : in STD_LOGIC;
        bt_clr : in STD_LOGIC;
        bt_plus : in STD_LOGIC;
        bt_eq : in STD_LOGIC;
        bt_ok :  in STD_LOGIC;
        KB_col : out std_logic_vector (4 downto 1);
        segment : out std_logic_vector (6 downto 0);
        AN : out std_logic_vector (7 downto 0);
        LED : out std_logic_vector(15 downto 0);
        LED16_R : out std_logic;
        LED16_G : out std_logic;
        LED16_B : out std_logic;
        LED17_G : out std_logic;
        LED17_R : out std_logic;
        LED17_B : out std_logic);
end trainsys;
```

VHDL/FPGA – Top File Structure



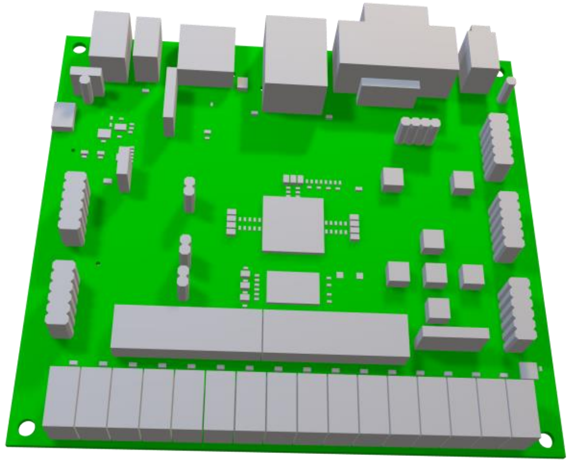
Nexys A7. Source: Diligent.



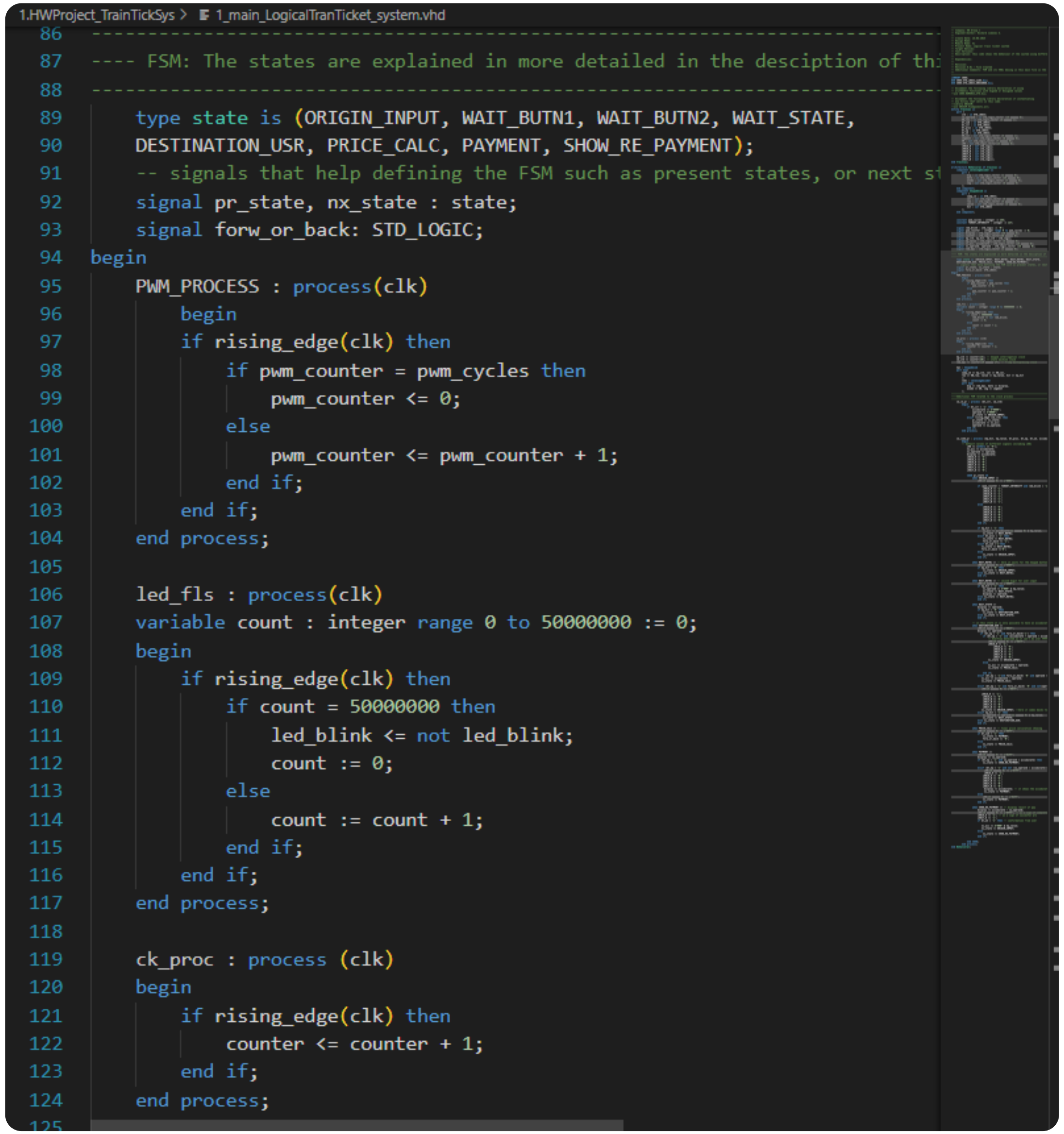
1.HWProject_TrainTickSys > 1_main_LogicalTranTicket_system.vhd

```vhdl
architecture Behavioral of trainsys is
    component sevensegdecoder is
        port (
            dig : in std_logic_vector (2 downto 0);
            data : in std_logic_vector (15 downto 0);
            anode : out std_logic_vector (7 downto 0);
            seg : out std_logic_vector (6 downto 0)
        );
    end component;
    component keypadecod is
        port (
            samp_ck : in STD_LOGIC;
            col : out std_logic_vector (4 downto 1);
            row : in std_logic_vector (4 downto 1);
            value : out std_logic_vector (3 downto 0);
            hit : out STD_LOGIC
        );
    end component;


    constant pwm_cycles : integer := 255;
    constant TARGET_INTENSITY : integer := 127;

    signal led_blink : std_logic := '0';
    signal pwm_counter : integer range 0 to pwm_cycles := 0;
    signal display : std_logic_vector (15 downto 0);
    signal counter : std_logic_vector(20 downto 0);
    signal kp_clk, kp_hit, sm_clk : std_logic;
    signal kp_value : std_logic_vector (3 downto 0);
    signal nx_acc, accumulate : std_logic_vector (15 downto 0);
    signal nx_operand, operand : std_logic_vector (15 downto 0);
    signal led_mpx : std_logic_vector (2 downto 0);
---------------------------------------------------------------------------
---- FSM: The states are explained in more detailed in the desciption of thi
---------------------------------------------------------------------------
```

VHDL/FPGA – Top File Structure



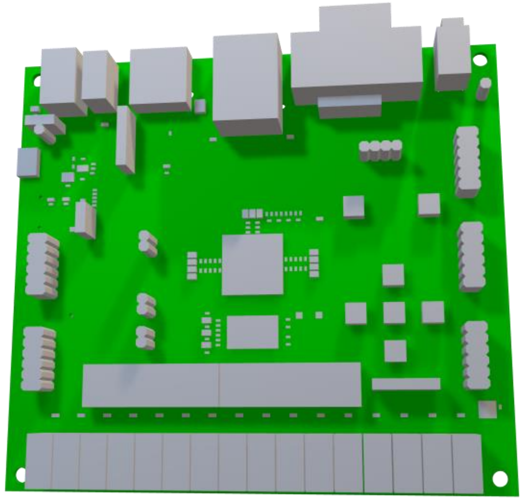
Nexys A7. Source: Diligent.



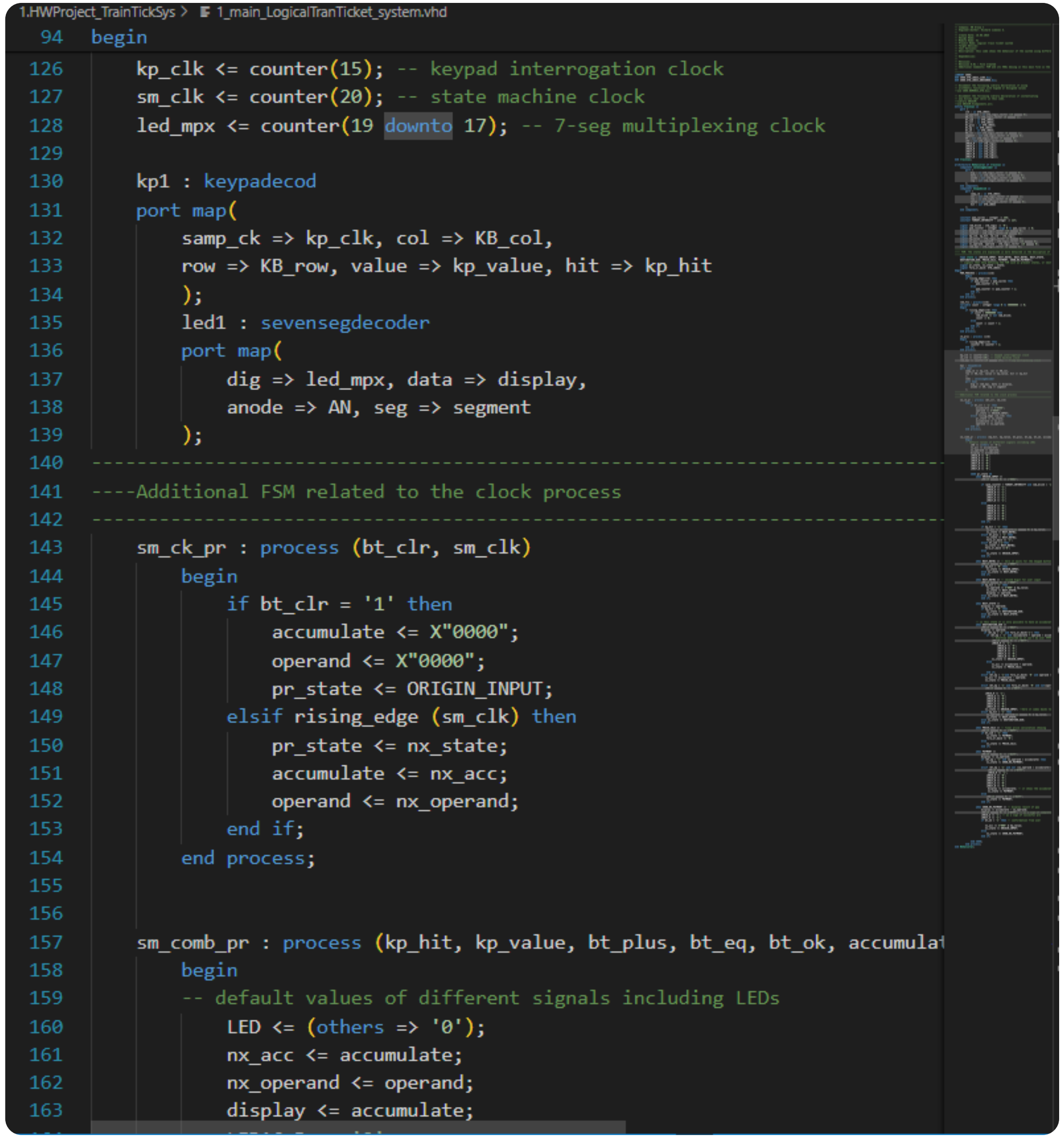
1.HWProject_TrainTickSys > 1_main_LogicalTranTicket_system.vhd

```vhdl
---------------------------------------------------------------------------
---- FSM: The states are explained in more detailed in the desciption of thi
---------------------------------------------------------------------------
    type state is (ORIGIN_INPUT, WAIT_BUTN1, WAIT_BUTN2, WAIT_STATE,
    DESTINATION_USR, PRICE_CALC, PAYMENT, SHOW_RE_PAYMENT);
    -- signals that help defining the FSM such as present states, or next st
    signal pr_state, nx_state : state;
    signal forw_or_back: STD_LOGIC;
begin
    PWM_PROCESS : process(clk)
        begin
        if rising_edge(clk) then
            if pwm_counter = pwm_cycles then
                pwm_counter <= 0;
            else
                pwm_counter <= pwm_counter + 1;
            end if;
        end if;
    end process;

    led_fls : process(clk)
    variable count : integer range 0 to 50000000 := 0;
    begin
        if rising_edge(clk) then
            if count = 50000000 then
                led_blink <= not led_blink;
                count := 0;
            else
                count := count + 1;
            end if;
        end if;
    end process;

    ck_proc : process (clk)
    begin
        if rising_edge(clk) then
            counter <= counter + 1;
        end if;
    end process;
```

VHDL/FPGA – Top File Structure



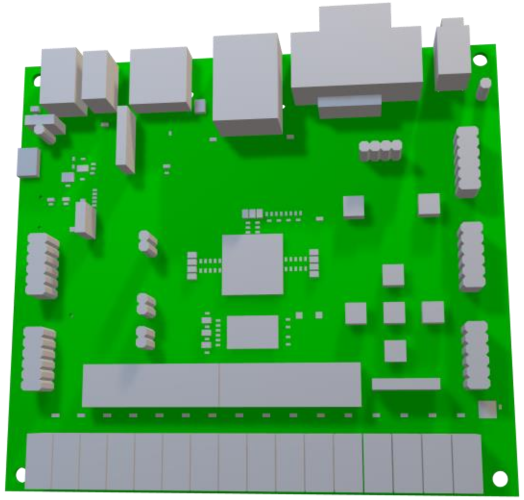
Nexys A7. Source: Diligent.



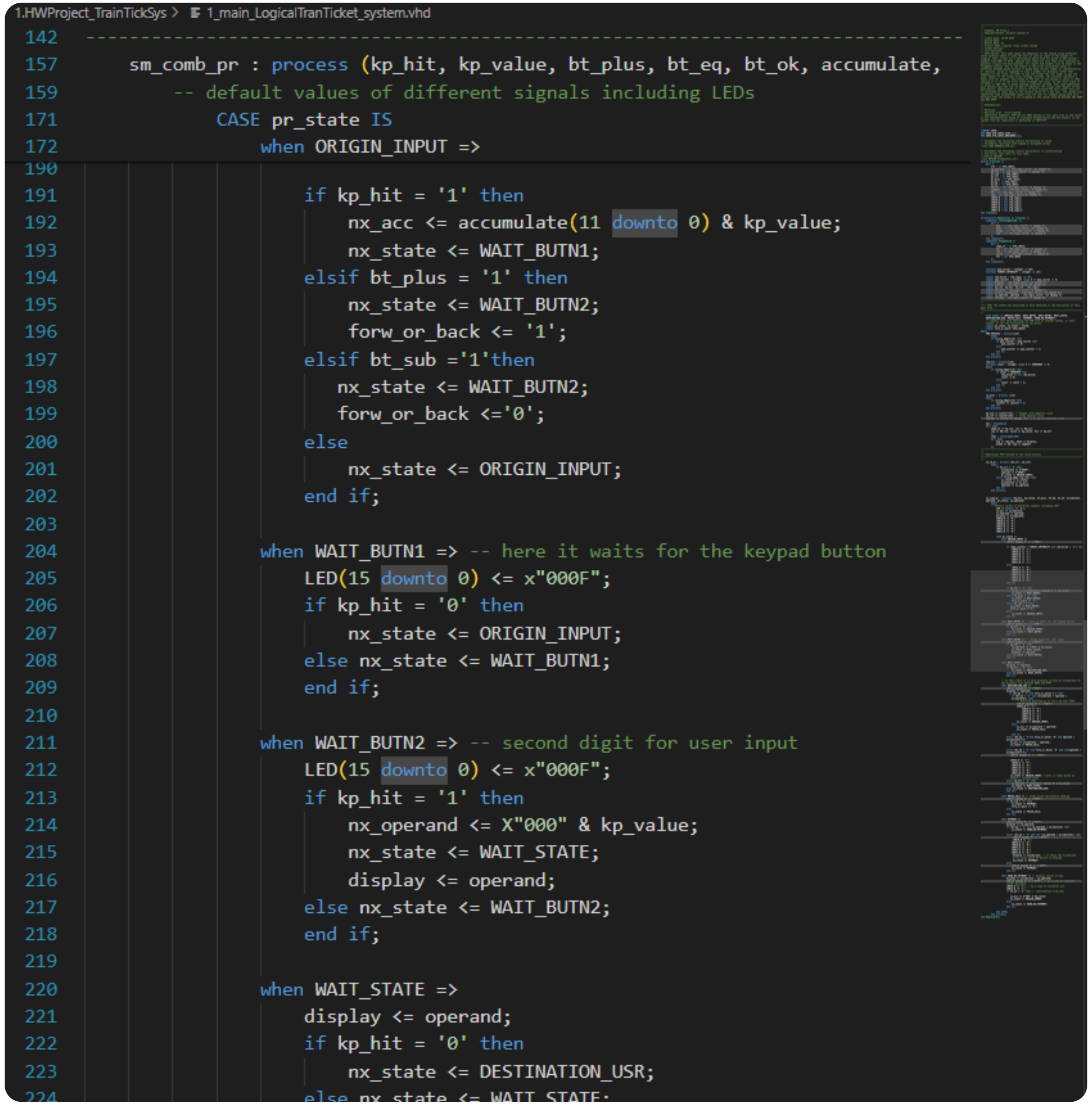
1.HWProject_TrainTickSys > 1_main_LogicalTranTicket_system.vhd

```vhdl
begin
    kp_clk <= counter(15); -- keypad interrogation clock
    sm_clk <= counter(20); -- state machine clock
    led_mpx <= counter(19 downto 17); -- 7-seg multiplexing clock

    kp1 : keypadecod
    port map(
        samp_ck => kp_clk, col => KB_col,
        row => KB_row, value => kp_value, hit => kp_hit
        );
        led1 : sevensegdecoder
        port map(
            dig => led_mpx, data => display,
            anode => AN, seg => segment
        );
---------------------------------------------------------------------------
----Additional FSM related to the clock process
---------------------------------------------------------------------------
    sm_ck_pr : process (bt_clr, sm_clk)
        begin
            if bt_clr = '1' then
                accumulate <= X"0000";
                operand <= X"0000";
                pr_state <= ORIGIN_INPUT;
            elsif rising_edge (sm_clk) then
                pr_state <= nx_state;
                accumulate <= nx_acc;
                operand <= nx_operand;
            end if;
        end process;


    sm_comb_pr : process (kp_hit, kp_value, bt_plus, bt_eq, bt_ok, accumulat
        begin
        -- default values of different signals including LEDs
            LED <= (others => '0');
            nx_acc <= accumulate;
            nx_operand <= operand;
            display <= accumulate;
```

VHDL/FPGA – Top File Structure



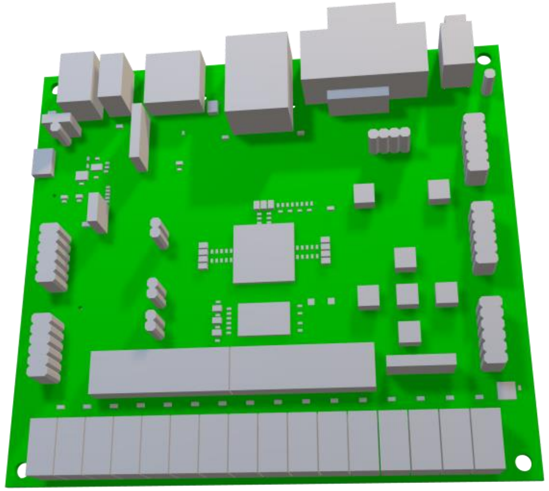
Nexys A7. Source: Diligent.



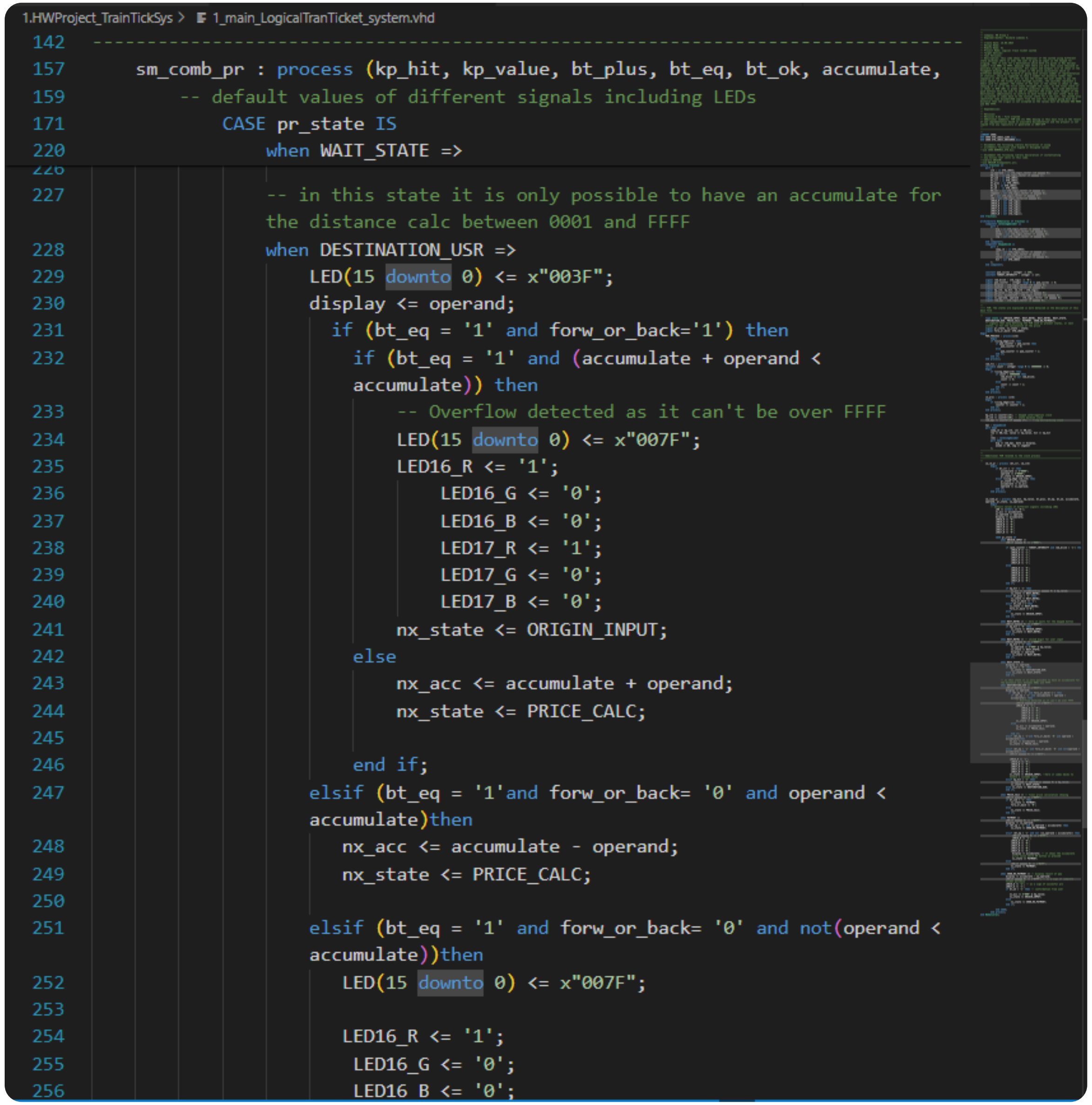
1.HWProject_TrainTickSys > 1_main_LogicalTranTicket_system.vhd

```vhdl
-------------------------------------------------------------------------------
    sm_comb_pr : process (kp_hit, kp_value, bt_plus, bt_eq, bt_ok, accumulate,
        -- default values of different signals including LEDs
            CASE pr_state IS
                when ORIGIN_INPUT =>

                    if kp_hit = '1' then
                        nx_acc <= accumulate(11 downto 0) & kp_value;
                        nx_state <= WAIT_BUTN1;
                    elsif bt_plus = '1' then
                        nx_state <= WAIT_BUTN2;
                        forw_or_back <= '1';
                    elsif bt_sub ='1'then
                       nx_state <= WAIT_BUTN2;
                       forw_or_back <='0';
                    else
                        nx_state <= ORIGIN_INPUT;
                    end if;

                when WAIT_BUTN1 => -- here it waits for the keypad button
                    LED(15 downto 0) <= x"000F";
                    if kp_hit = '0' then
                        nx_state <= ORIGIN_INPUT;
                    else nx_state <= WAIT_BUTN1;
                    end if;

                when WAIT_BUTN2 => -- second digit for user input
                    LED(15 downto 0) <= x"000F";
                    if kp_hit = '1' then
                        nx_operand <= X"000" & kp_value;
                        nx_state <= WAIT_STATE;
                        display <= operand;
                    else nx_state <= WAIT_BUTN2;
                    end if;

                when WAIT_STATE =>
                    display <= operand;
                    if kp_hit = '0' then
                        nx_state <= DESTINATION_USR;
                    else nx_state <= WAIT_STATE;
```

VHDL/FPGA – Top File Structure



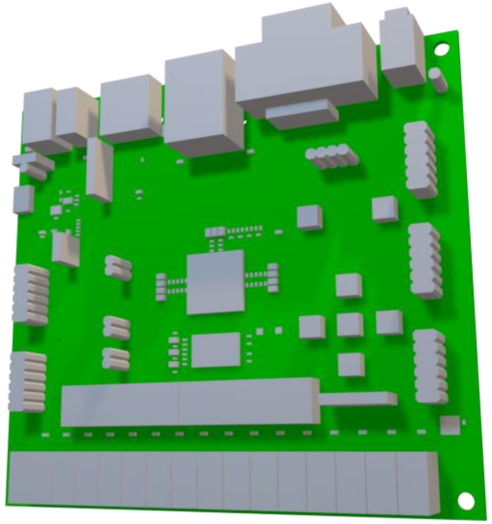
Nexys A7. Source: Diligent.


1.HWProject_TrainTickSys > 1_main_LogicalTranTicket_system.vhd

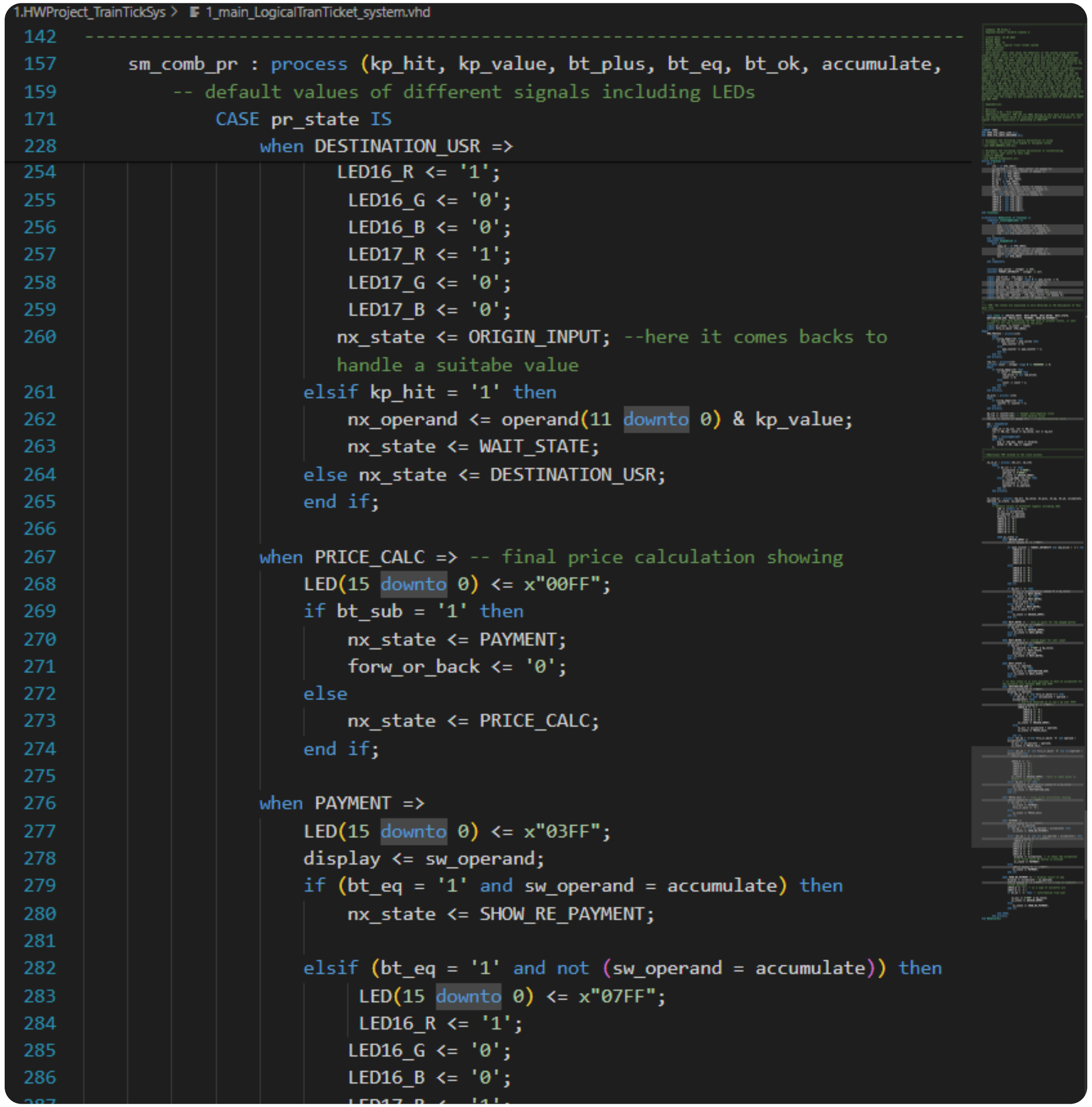
```
--------------------------------------------------------------------------------
    sm_comb_pr : process (kp_hit, kp_value, bt_plus, bt_eq, bt_ok, accumulate,
        -- default values of different signals including LEDs
            CASE pr_state IS
                when WAIT_STATE =>

                -- in this state it is only possible to have an accumulate for
                the distance calc between 0001 and FFFF
                when DESTINATION_USR =>
                    LED(15 downto 0) <= x"003F";
                    display <= operand;
                      if (bt_eq = '1' and forw_or_back='1') then
                        if (bt_eq = '1' and (accumulate + operand <
                        accumulate)) then
                            -- Overflow detected as it can't be over FFFF
                            LED(15 downto 0) <= x"007F";
                            LED16_R <= '1';
                                LED16_G <= '0';
                                LED16_B <= '0';
                                LED17_R <= '1';
                                LED17_G <= '0';
                                LED17_B <= '0';
                            nx_state <= ORIGIN_INPUT;
                        else
                            nx_acc <= accumulate + operand;
                            nx_state <= PRICE_CALC;

                        end if;
                    elsif (bt_eq = '1'and forw_or_back= '0' and operand <
                    accumulate)then
                       nx_acc <= accumulate - operand;
                       nx_state <= PRICE_CALC;

                    elsif (bt_eq = '1' and forw_or_back= '0' and not(operand <
                    accumulate))then
                       LED(15 downto 0) <= x"007F";

                       LED16_R <= '1';
                        LED16_G <= '0';
                        LED16_B <= '0';
```

Group C

VHDL/FPGA – Top File Structure



Nexys A7. Source: Diligent.

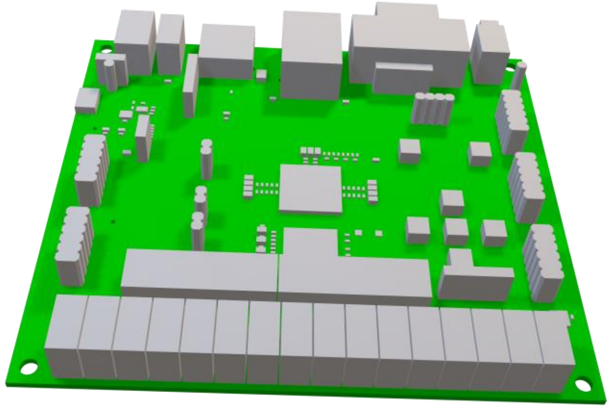


1.HWProject_TrainTickSys > 1_main_LogicalTranTicket_system.vhd

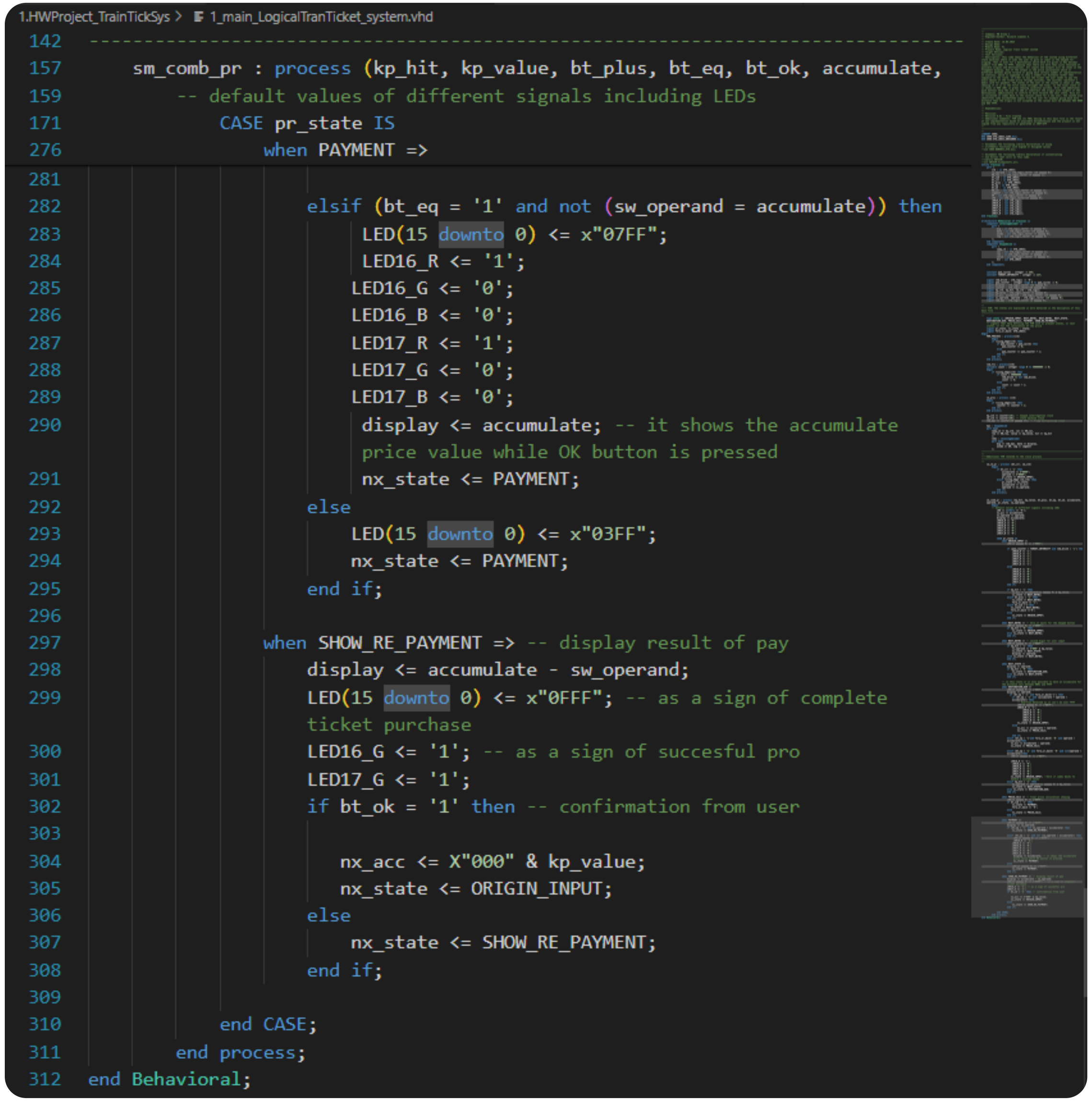
```vhdl
-------------------------------------------------------------------------------
    sm_comb_pr : process (kp_hit, kp_value, bt_plus, bt_eq, bt_ok, accumulate,
        -- default values of different signals including LEDs
            CASE pr_state IS
                when DESTINATION_USR =>
                       LED16_R <= '1';
                        LED16_G <= '0';
                        LED16_B <= '0';
                        LED17_R <= '1';
                        LED17_G <= '0';
                        LED17_B <= '0';
                       nx_state <= ORIGIN_INPUT; --here it comes backs to handle a suitabe value
                    elsif kp_hit = '1' then
                        nx_operand <= operand(11 downto 0) & kp_value;
                        nx_state <= WAIT_STATE;
                    else nx_state <= DESTINATION_USR;
                    end if;

                when PRICE_CALC => -- final price calculation showing
                    LED(15 downto 0) <= x"00FF";
                    if bt_sub = '1' then
                        nx_state <= PAYMENT;
                        forw_or_back <= '0';
                    else
                        nx_state <= PRICE_CALC;
                    end if;

                when PAYMENT =>
                    LED(15 downto 0) <= x"03FF";
                    display <= sw_operand;
                    if (bt_eq = '1' and sw_operand = accumulate) then
                        nx_state <= SHOW_RE_PAYMENT;

                    elsif (bt_eq = '1' and not (sw_operand = accumulate)) then
                         LED(15 downto 0) <= x"07FF";
                         LED16_R <= '1';
                        LED16_G <= '0';
                        LED16_B <= '0';
```

Group C

VHDL/FPGA – Top File Structure



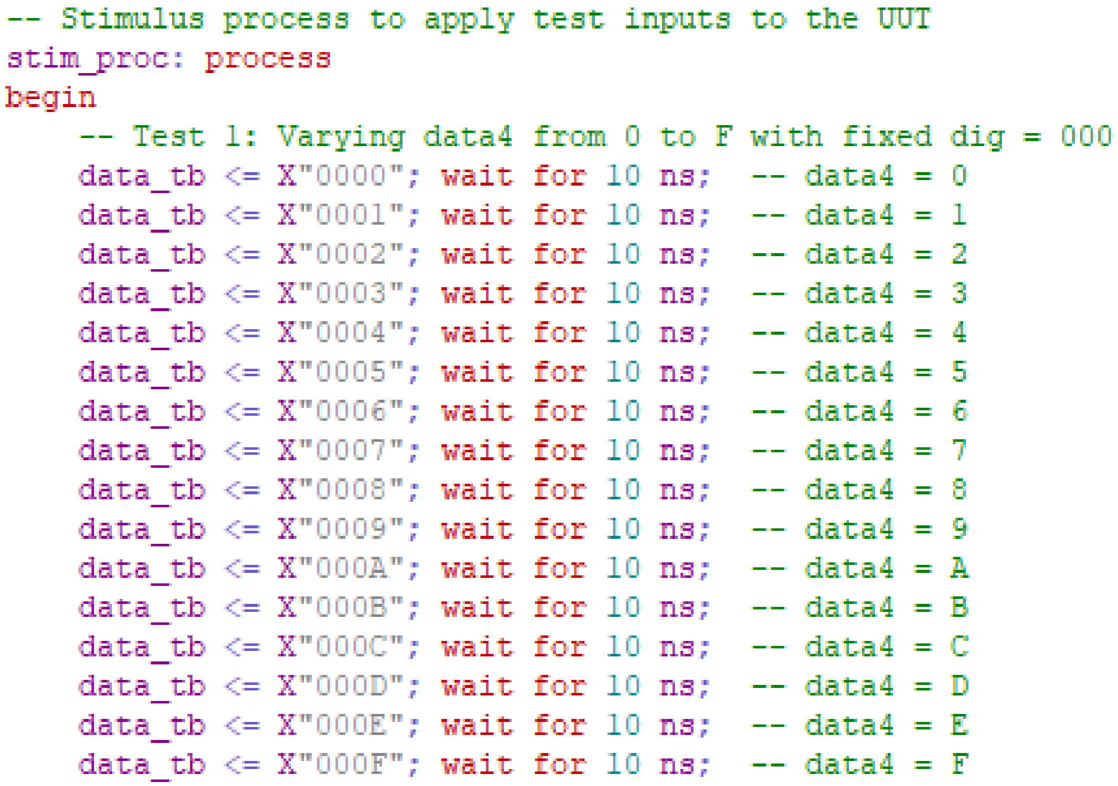
Nexys A7. Source: Diligent.



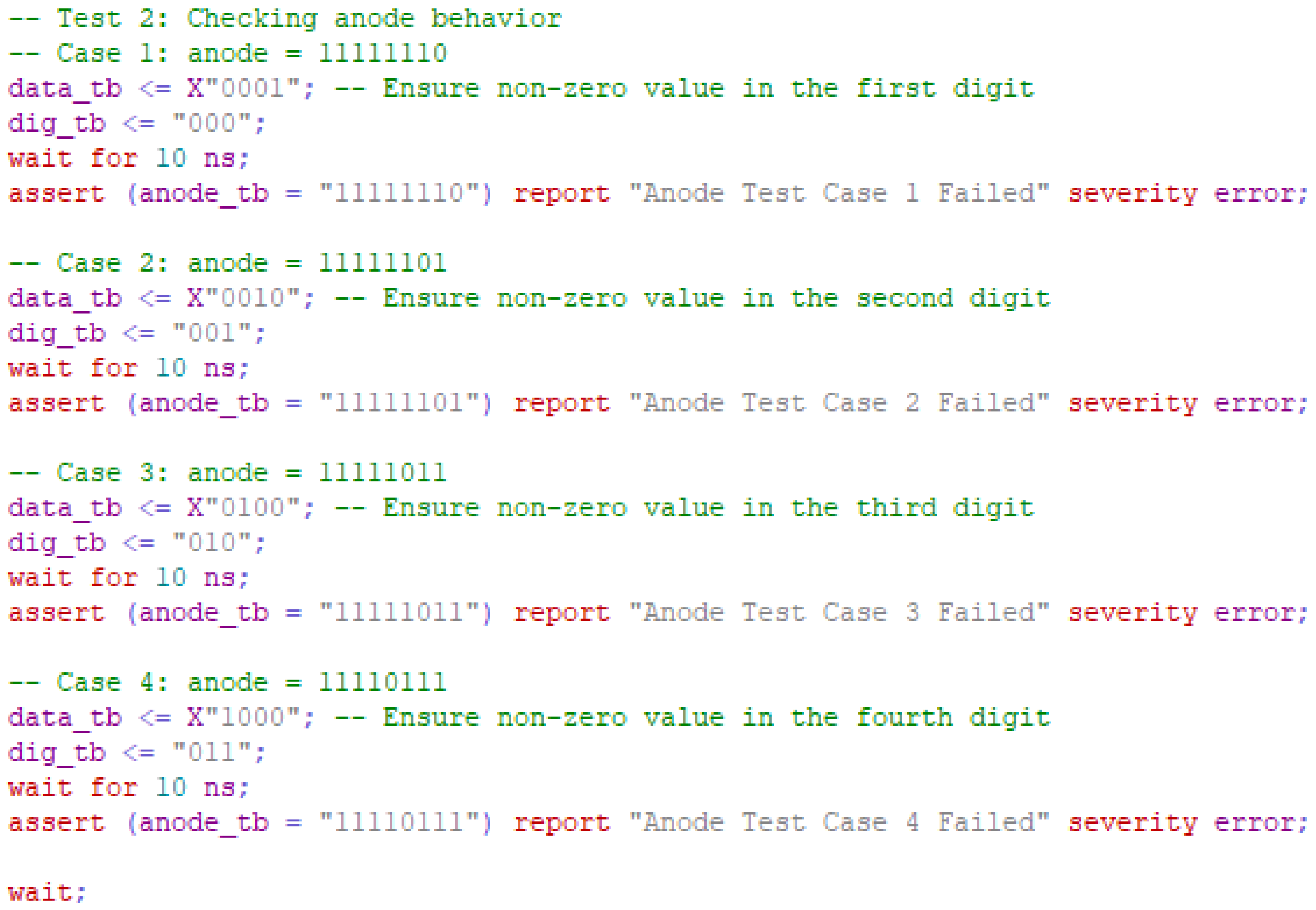
1.HWProject_TrainTickSys > 1_main_LogicalTranTicket_system.vhd

```
----------------------------------------
  sm_comb_pr : process (kp_hit, kp_value, bt_plus, bt_eq, bt_ok, accumulate,
    -- default values of different signals including LEDs
      CASE pr_state IS
        when PAYMENT =>

          elsif (bt_eq = '1' and not (sw_operand = accumulate)) then
            LED(15 downto 0) <= x"07FF";
            LED16_R <= '1';
            LED16_G <= '0';
            LED16_B <= '0';
            LED17_R <= '1';
            LED17_G <= '0';
            LED17_B <= '0';
            display <= accumulate; -- it shows the accumulate price value while OK button is pressed
            nx_state <= PAYMENT;
          else
            LED(15 downto 0) <= x"03FF";
            nx_state <= PAYMENT;
          end if;

        when SHOW_RE_PAYMENT => -- display result of pay
          display <= accumulate - sw_operand;
          LED(15 downto 0) <= x"0FFF"; -- as a sign of complete ticket purchase
          LED16_G <= '1'; -- as a sign of succesful pro
          LED17_G <= '1';
          if bt_ok = '1' then -- confirmation from user

           nx_acc <= X"000" & kp_value;
           nx_state <= ORIGIN_INPUT;
          else
            nx_state <= SHOW_RE_PAYMENT;
          end if;

      end CASE;
    end process;
end Behavioral;
```

TestBench_sevensegdecoder

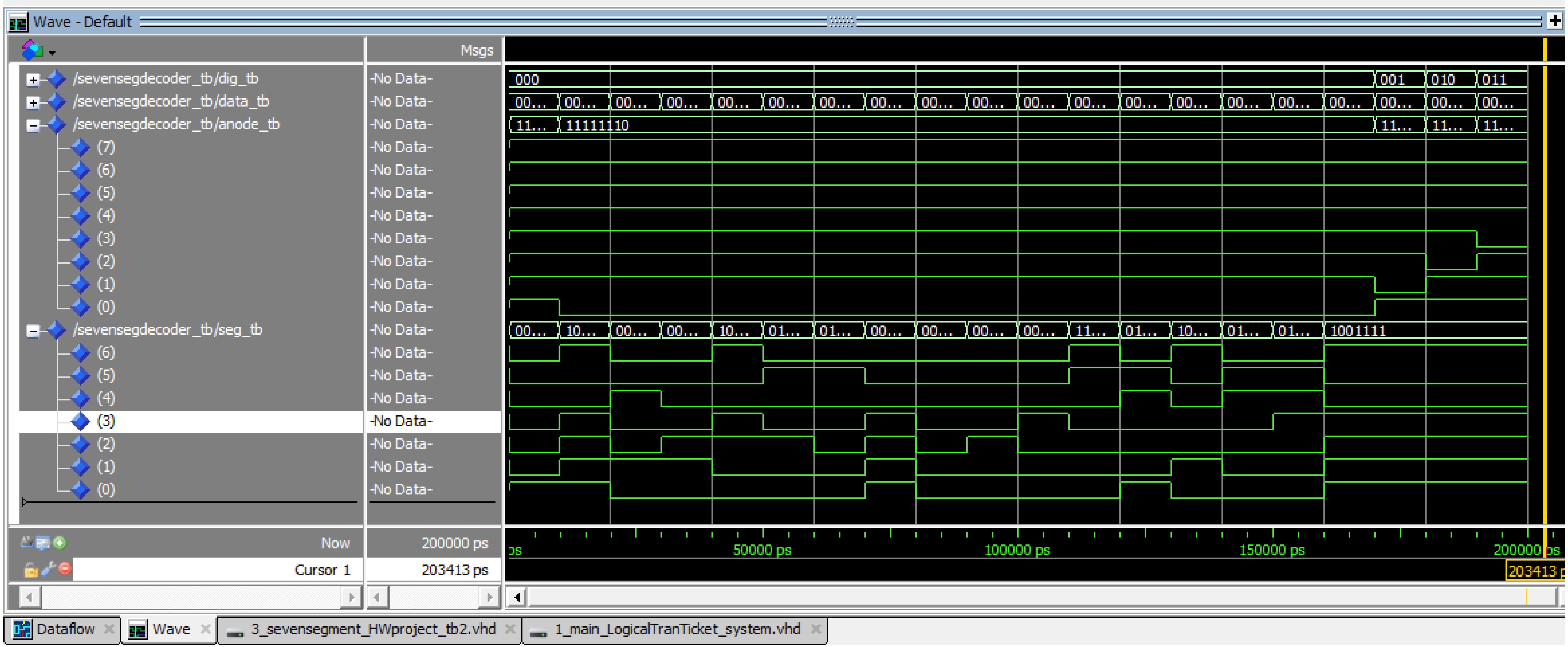


```
-- Stimulus process to apply test inputs to the UUT
stim_proc: process
begin
    -- Test 1: Varying data4 from 0 to F with fixed dig = 000
    data_tb <= X"0000"; wait for 10 ns;  -- data4 = 0
    data_tb <= X"0001"; wait for 10 ns;  -- data4 = 1
    data_tb <= X"0002"; wait for 10 ns;  -- data4 = 2
    data_tb <= X"0003"; wait for 10 ns;  -- data4 = 3
    data_tb <= X"0004"; wait for 10 ns;  -- data4 = 4
    data_tb <= X"0005"; wait for 10 ns;  -- data4 = 5
    data_tb <= X"0006"; wait for 10 ns;  -- data4 = 6
    data_tb <= X"0007"; wait for 10 ns;  -- data4 = 7
    data_tb <= X"0008"; wait for 10 ns;  -- data4 = 8
    data_tb <= X"0009"; wait for 10 ns;  -- data4 = 9
    data_tb <= X"000A"; wait for 10 ns;  -- data4 = A
    data_tb <= X"000B"; wait for 10 ns;  -- data4 = B
    data_tb <= X"000C"; wait for 10 ns;  -- data4 = C
    data_tb <= X"000D"; wait for 10 ns;  -- data4 = D
    data_tb <= X"000E"; wait for 10 ns;  -- data4 = E
    data_tb <= X"000F"; wait for 10 ns;  -- data4 = F
```



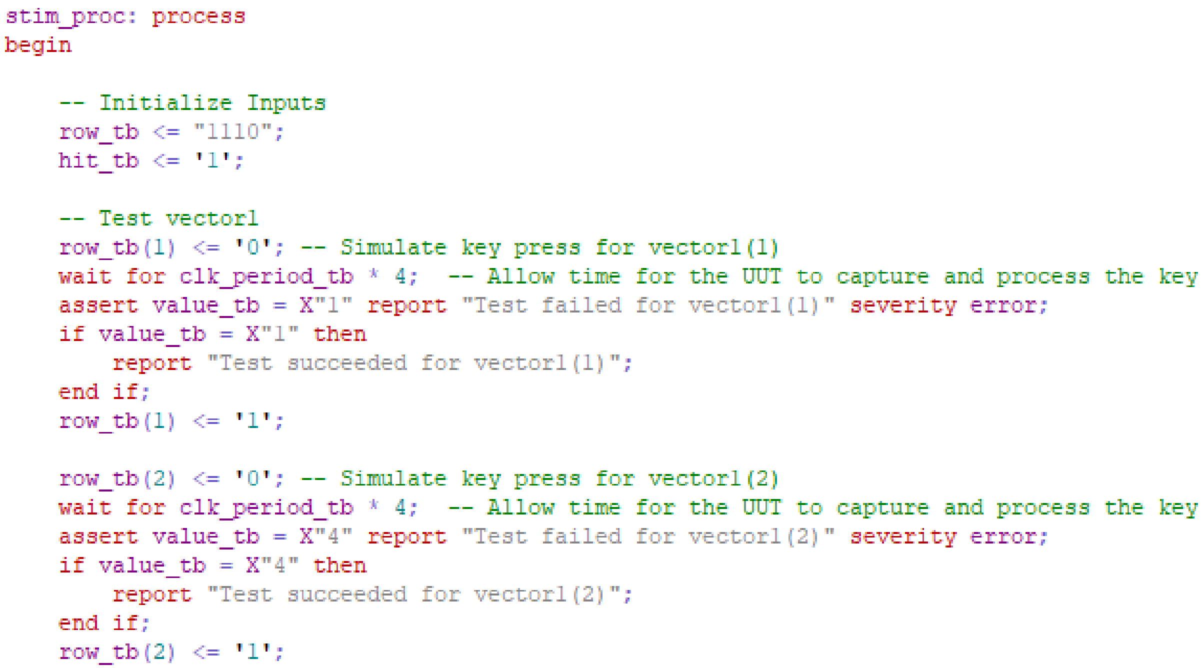
```
-- Test 2: Checking anode behavior
-- Case 1: anode = 11111110
data_tb <= X"0001"; -- Ensure non-zero value in the first digit
dig_tb <= "000";
wait for 10 ns;
assert (anode_tb = "11111110") report "Anode Test Case 1 Failed" severity error;

-- Case 2: anode = 11111101
data_tb <= X"0010"; -- Ensure non-zero value in the second digit
dig_tb <= "001";
wait for 10 ns;
assert (anode_tb = "11111101") report "Anode Test Case 2 Failed" severity error;

-- Case 3: anode = 11111011
data_tb <= X"0100"; -- Ensure non-zero value in the third digit
dig_tb <= "010";
wait for 10 ns;
assert (anode_tb = "11111011") report "Anode Test Case 3 Failed" severity error;

-- Case 4: anode = 11110111
data_tb <= X"1000"; -- Ensure non-zero value in the fourth digit
dig_tb <= "011";
wait for 10 ns;
assert (anode_tb = "11110111") report "Anode Test Case 4 Failed" severity error;

wait;
```

TestBench_sevensegdecoder

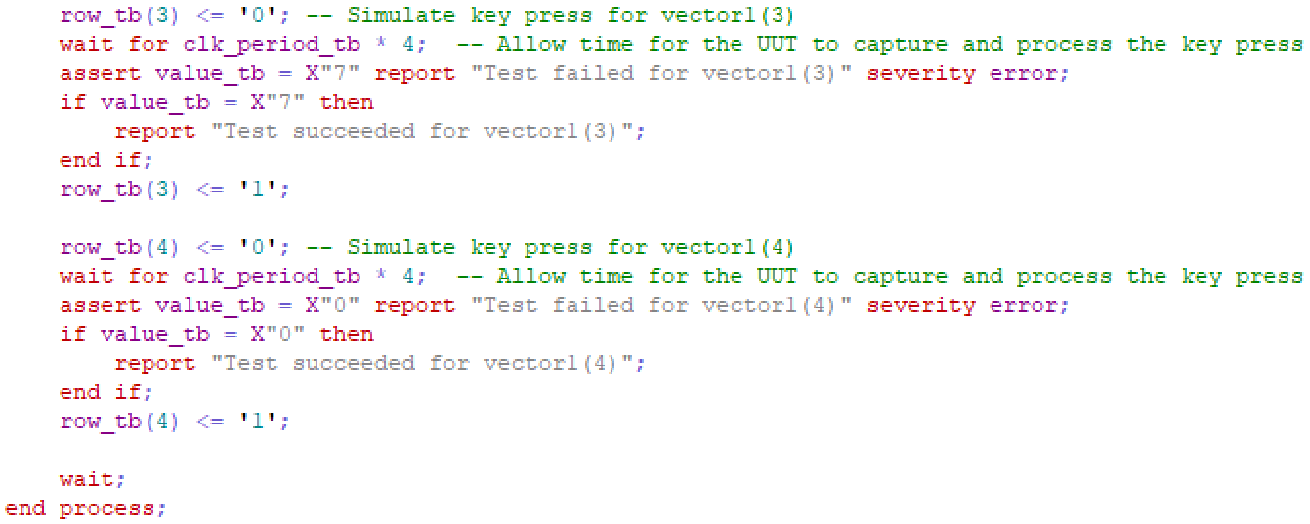


Group C

TestBench_keypadcod

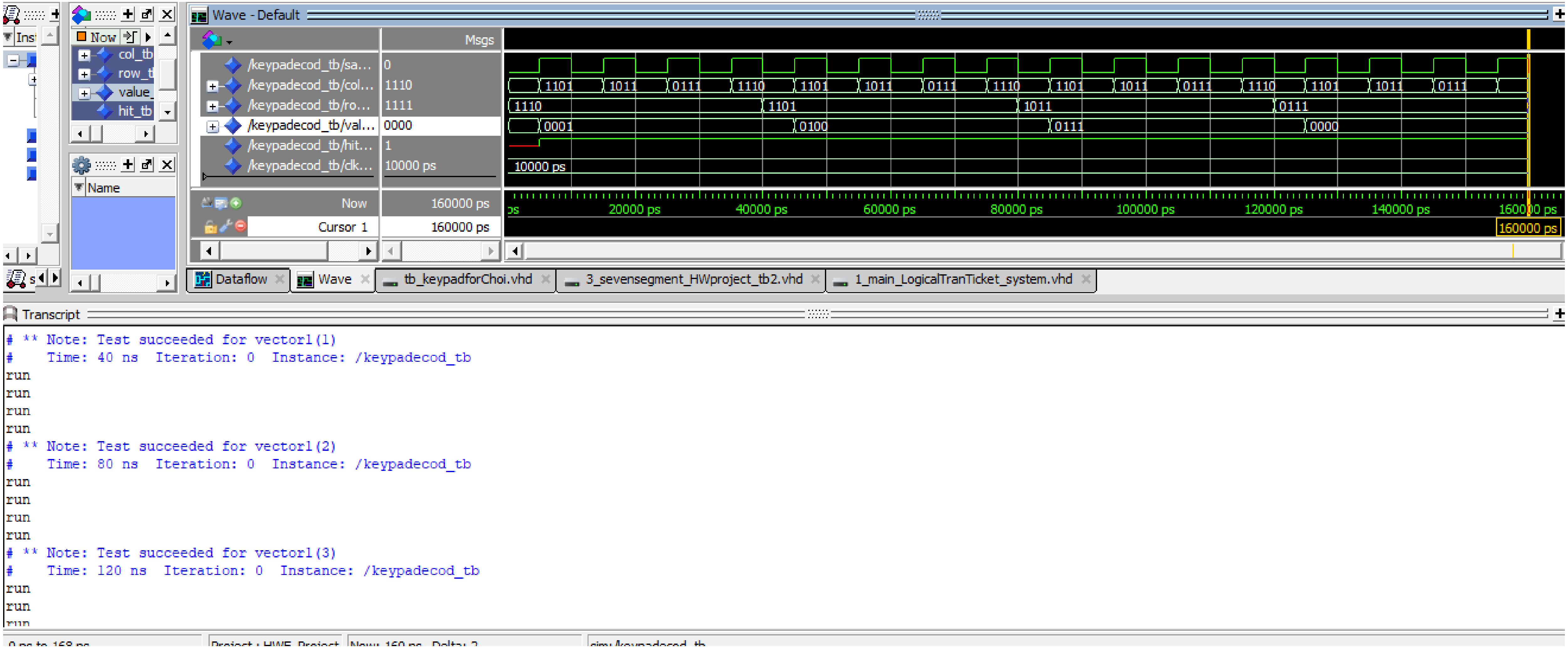


```
stim_proc: process
begin

    -- Initialize Inputs
    row_tb <= "1110";
    hit_tb <= '1';

    -- Test vector1
    row_tb(1) <= '0'; -- Simulate key press for vector1(1)
    wait for clk_period_tb * 4;  -- Allow time for the UUT to capture and process the key
    assert value_tb = X"1" report "Test failed for vector1(1)" severity error;
    if value_tb = X"1" then
        report "Test succeeded for vector1(1)";
    end if;
    row_tb(1) <= '1';

    row_tb(2) <= '0'; -- Simulate key press for vector1(2)
    wait for clk_period_tb * 4;  -- Allow time for the UUT to capture and process the key
    assert value_tb = X"4" report "Test failed for vector1(2)" severity error;
    if value_tb = X"4" then
        report "Test succeeded for vector1(2)";
    end if;
    row_tb(2) <= '1';
```



```
    row_tb(3) <= '0'; -- Simulate key press for vector1(3)
    wait for clk_period_tb * 4;  -- Allow time for the UUT to capture and process the key press
    assert value_tb = X"7" report "Test failed for vector1(3)" severity error;
    if value_tb = X"7" then
        report "Test succeeded for vector1(3)";
    end if;
    row_tb(3) <= '1';

    row_tb(4) <= '0'; -- Simulate key press for vector1(4)
    wait for clk_period_tb * 4;  -- Allow time for the UUT to capture and process the key press
    assert value_tb = X"0" report "Test failed for vector1(4)" severity error;
    if value_tb = X"0" then
        report "Test succeeded for vector1(4)";
    end if;
    row_tb(4) <= '1';

    wait;
end process;
```

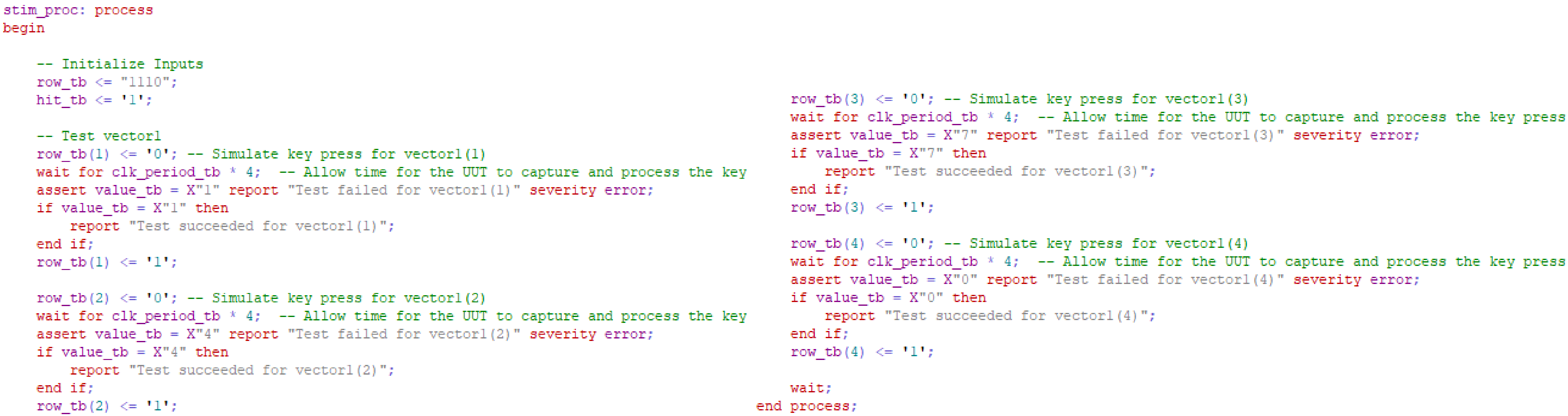
TestBench_keypadcod



```
# ** Note: Test succeeded for vector1(1)
#    Time: 40 ns  Iteration: 0  Instance: /keypadecod_tb
run
run
run
run
# ** Note: Test succeeded for vector1(2)
#    Time: 80 ns  Iteration: 0  Instance: /keypadecod_tb
run
run
run
run
# ** Note: Test succeeded for vector1(3)
#    Time: 120 ns  Iteration: 0  Instance: /keypadecod_tb
run
run
run
```

Group C

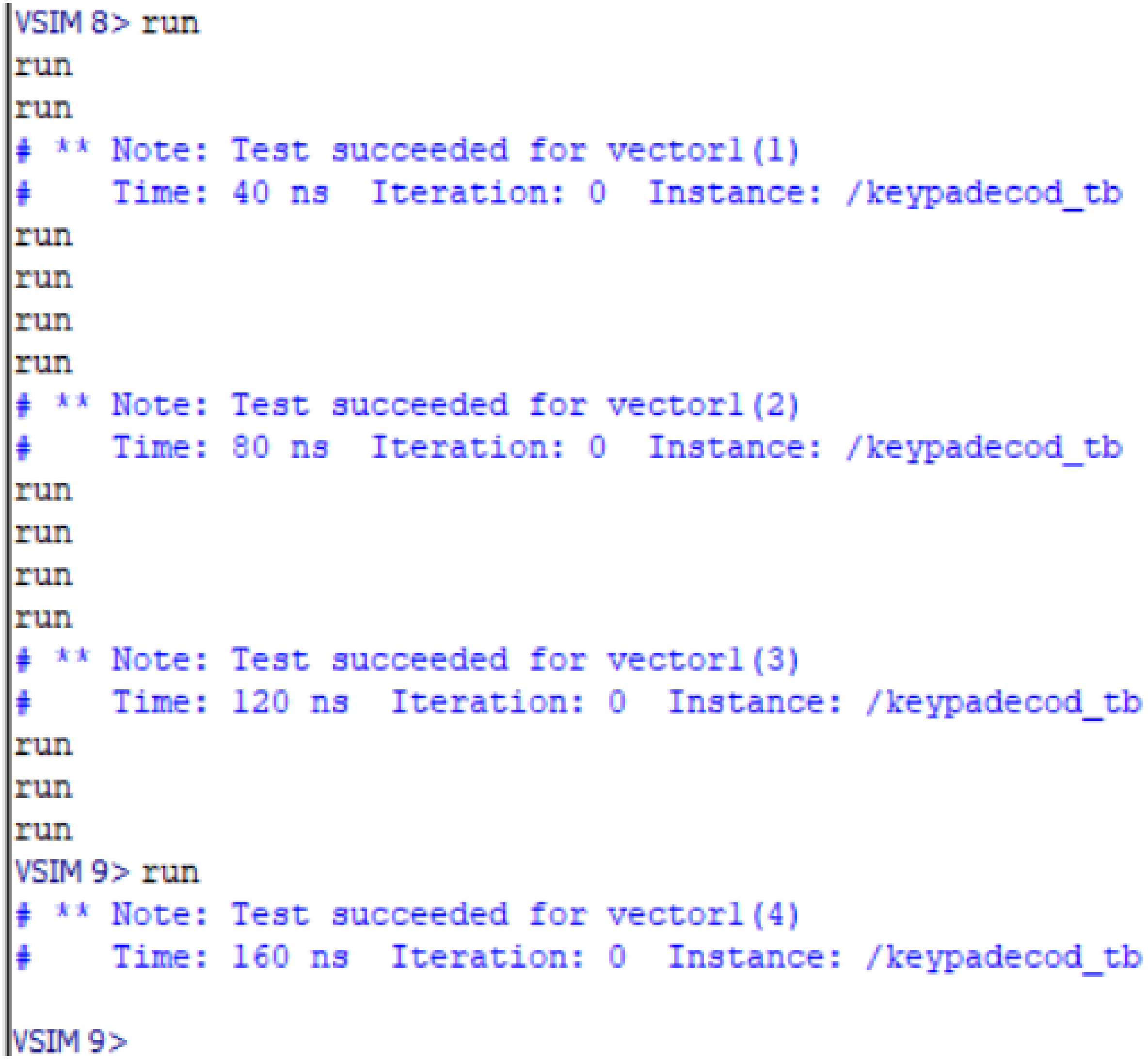
TestBench_main



```
stim_proc: process
begin

    -- Initialize Inputs
    row_tb <= "1110";
    hit_tb <= '1';

    -- Test vector1
    row_tb(1) <= '0'; -- Simulate key press for vector1(1)
    wait for clk_period_tb * 4;  -- Allow time for the UUT to capture and process the key
    assert value_tb = X"1" report "Test failed for vector1(1)" severity error;
    if value_tb = X"1" then
        report "Test succeeded for vector1(1)";
    end if;
    row_tb(1) <= '1';

    row_tb(2) <= '0'; -- Simulate key press for vector1(2)
    wait for clk_period_tb * 4;  -- Allow time for the UUT to capture and process the key
    assert value_tb = X"4" report "Test failed for vector1(2)" severity error;
    if value_tb = X"4" then
        report "Test succeeded for vector1(2)";
    end if;
    row_tb(2) <= '1';
    row_tb(3) <= '0'; -- Simulate key press for vector1(3)
    wait for clk_period_tb * 4;  -- Allow time for the UUT to capture and process the key press
    assert value_tb = X"7" report "Test failed for vector1(3)" severity error;
    if value_tb = X"7" then
        report "Test succeeded for vector1(3)";
    end if;
    row_tb(3) <= '1';

    row_tb(4) <= '0'; -- Simulate key press for vector1(4)
    wait for clk_period_tb * 4;  -- Allow time for the UUT to capture and process the key press
    assert value_tb = X"0" report "Test failed for vector1(4)" severity error;
    if value_tb = X"0" then
        report "Test succeeded for vector1(4)";
    end if;
    row_tb(4) <= '1';

    wait;
end process;
```

Group C

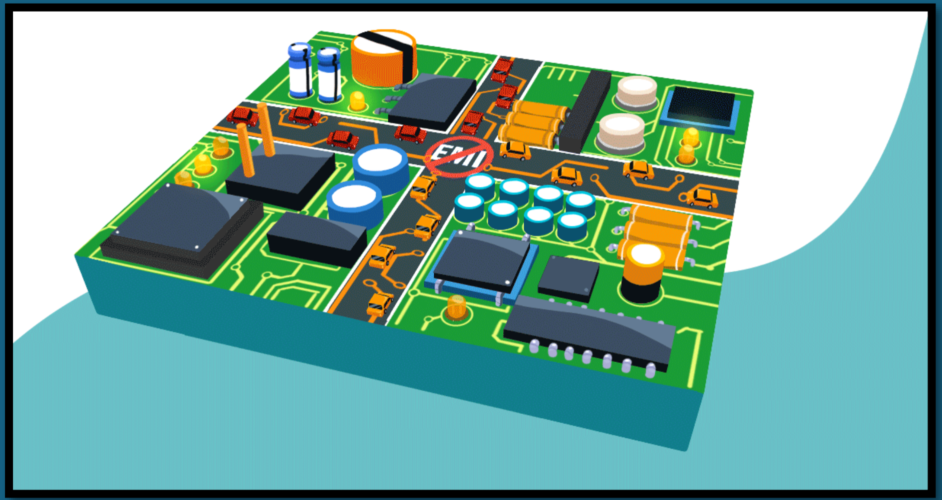
TestBench_main



```
VSIM 8> run
run
run
# ** Note: Test succeeded for vectorl(1)
#    Time: 40 ns  Iteration: 0  Instance: /keypadecod_tb
run
run
run
run
# ** Note: Test succeeded for vectorl(2)
#    Time: 80 ns  Iteration: 0  Instance: /keypadecod_tb
run
run
run
run
# ** Note: Test succeeded for vectorl(3)
#    Time: 120 ns  Iteration: 0  Instance: /keypadecod_tb
run
run
run
VSIM 9> run
# ** Note: Test succeeded for vectorl(4)
#    Time: 160 ns  Iteration: 0  Instance: /keypadecod_tb

VSIM 9>
```

PCB Design

PCB design refers to the process of designing printed circuit boards.

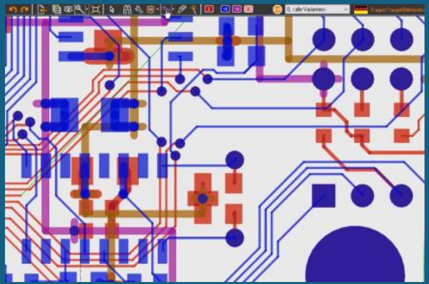


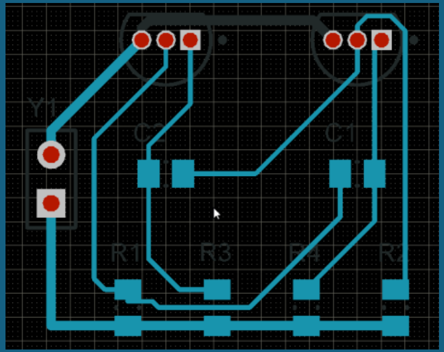
- Electrically connect electronic components
- Reduce the risk of accident
- Minimize the size

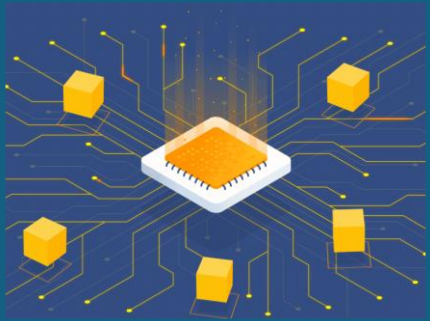
Process of PCB Design

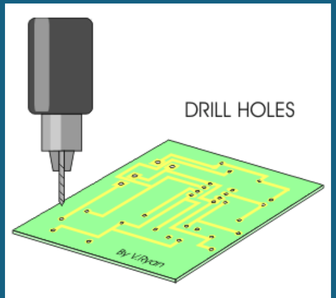
There are may process of PCB Design

- Find out the requirement according the target group or client
- Select the design tools
- Schematics drawing
- Components placements
- Routing
- Design Rules Check
- Garber file generation
- Simulation and testing
- Prototyping
- Final production





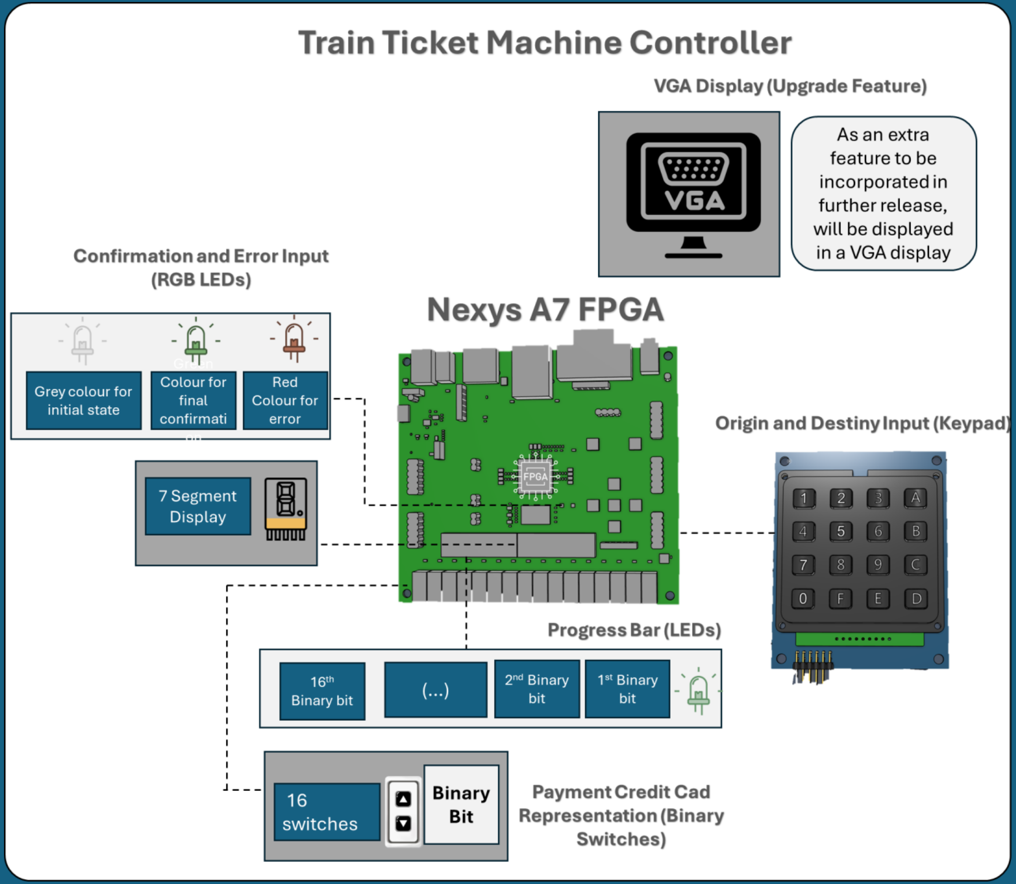




Requirement for our custom PCB Board

According our use case diagram and discussing with team members

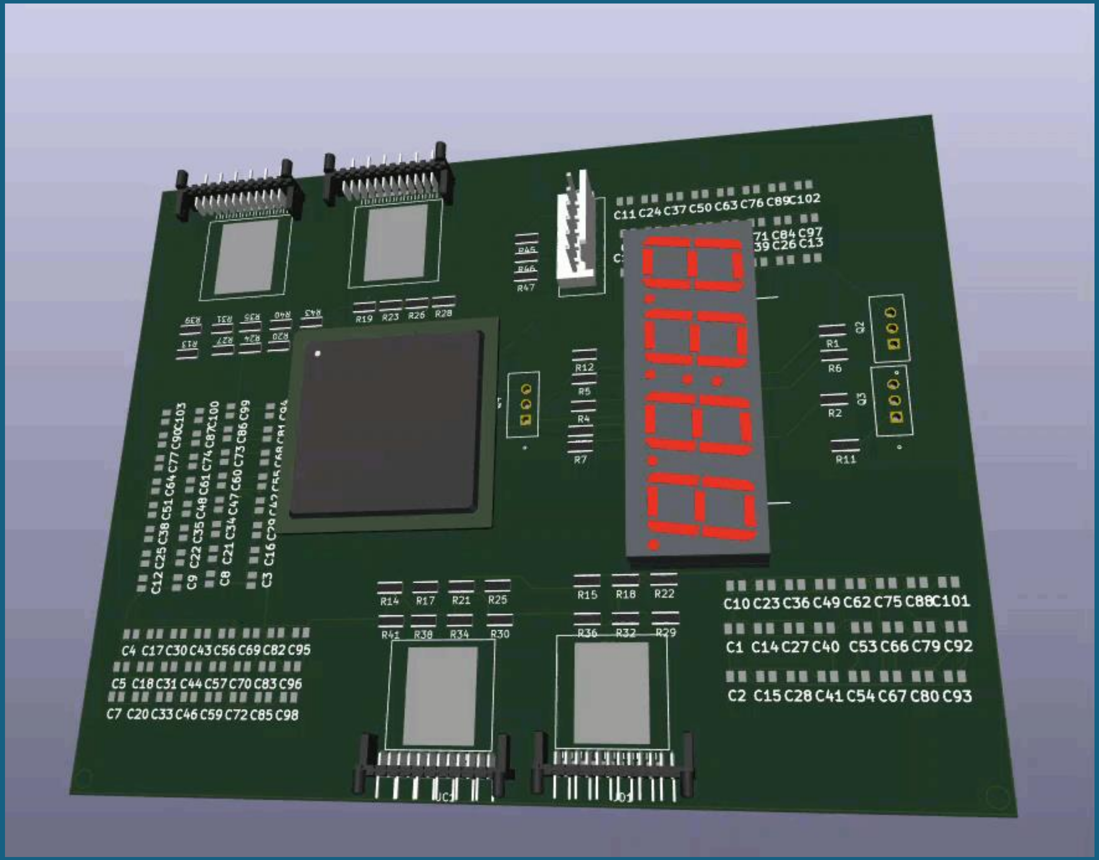
- should have enough I/O pin
- Must have a 7 segment display
- Must have a programming interface
- Must have power supply system
- Should have indicating LED
- Must have switch for giving input


Train Ticket Machine Controller
VGA Display (Upgrade Feature)
VGA
As an extra feature to be incorporated in further release, will be displayed in a VGA display
Confirmation and Error Input (RGB LEDs)
Grey colour for initial state
Colour for final confirmati
Red Colour for error
Nexys A7 FPGA
FPGA
7 Segment Display
Origin and Destiny Input (Keypad)
1
2
3
A
4
5
6
B
7
8
9
C
0
F
E
D
Progress Bar (LEDs)
16th Binary bit
(...)
2nd Binary bit
1st Binary bit
16 switches
Binary Bit
Payment Credit Cad Representation (Binary Switches)

Our custom PCB Board Components

After discussion and deep analysis we select those elements for our PCB

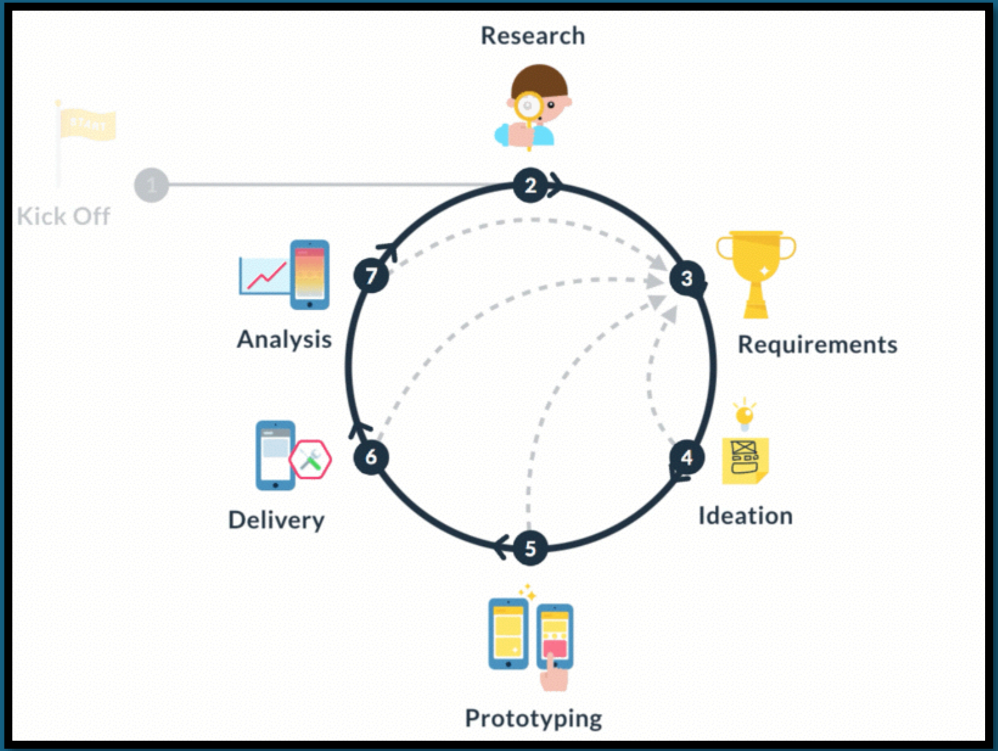
- One 7 segment display
- 28 I/O pins
- A Jtag for programming interface
- power supply
- Artix A7 FPGA



Our custom PCB Board Design Process

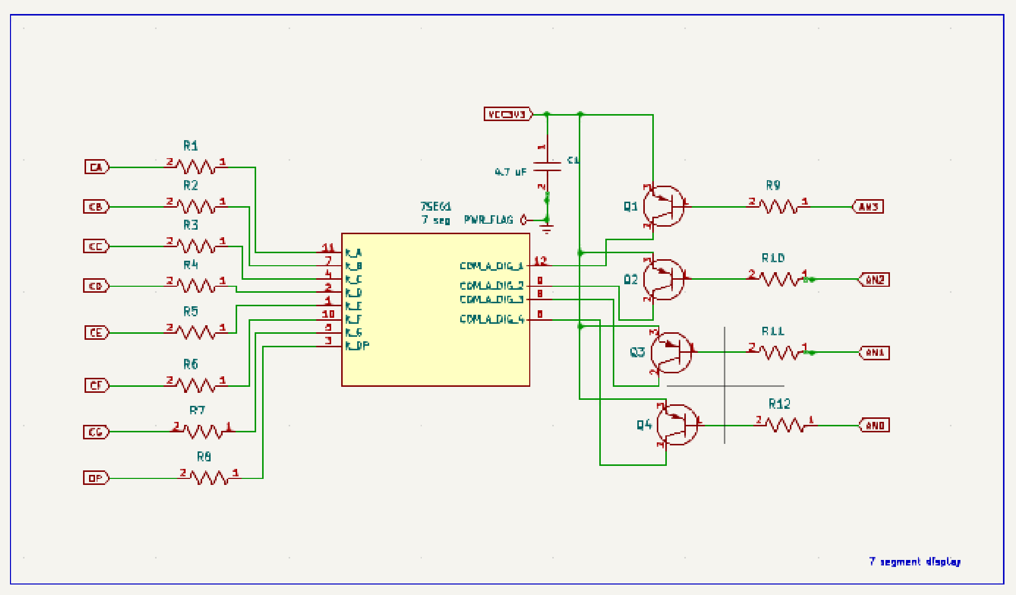
I did follow some steps to creating our custom PCB

- Initially I did install Kicad 8.0 for designing
- I did create a folder for save our custom board.
- I start add components from library to schematics
- I downloaded some components from ultra librarian
- After complete schematics then I start routing
- I added external plugin for auto routing
- I added some connection manually
- I am able to see the nice 3D view of custom PCB
- Now I am able to generate Garver file

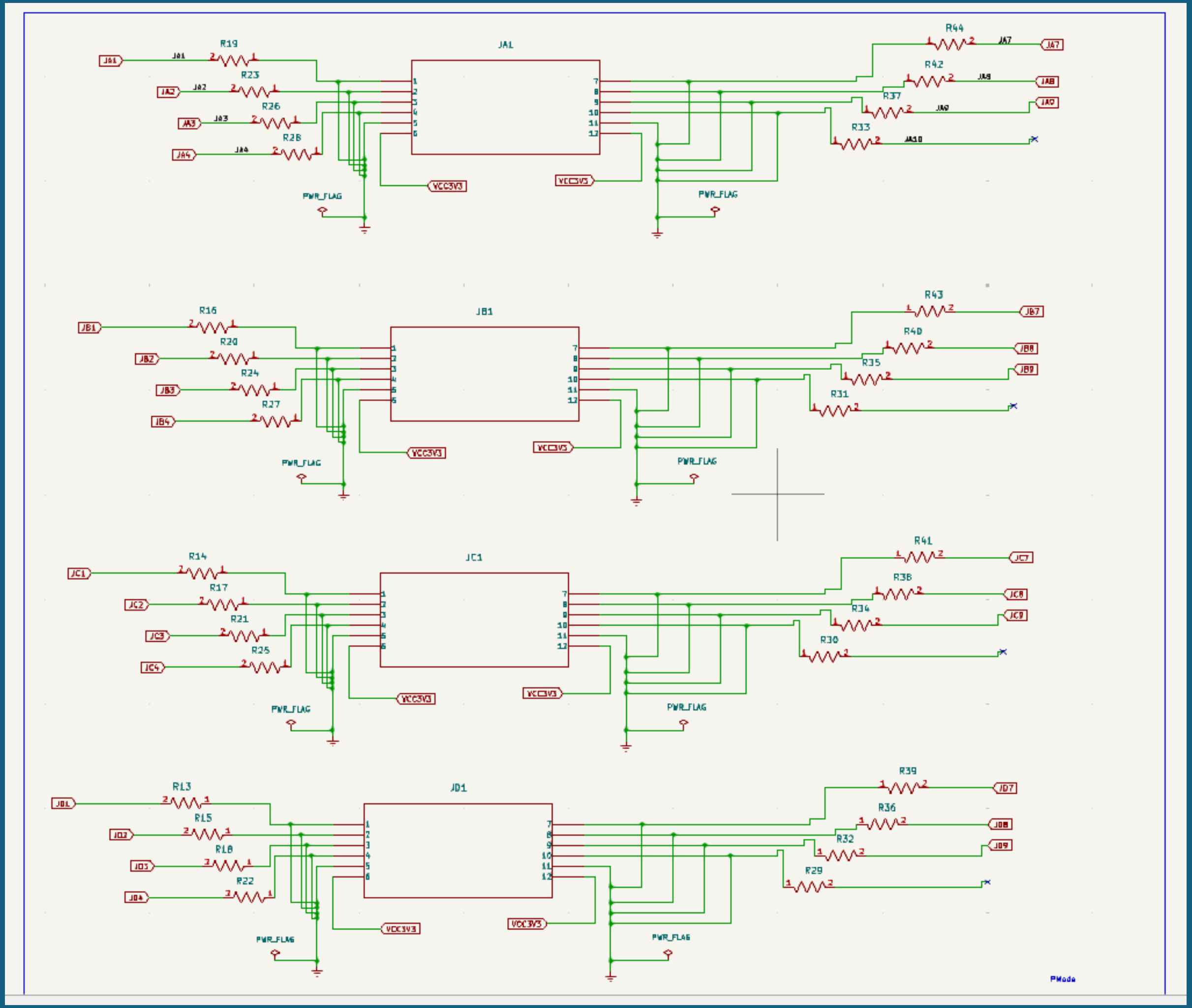


Our custom PCB Board Design Process

Here is the schematics for our PCB



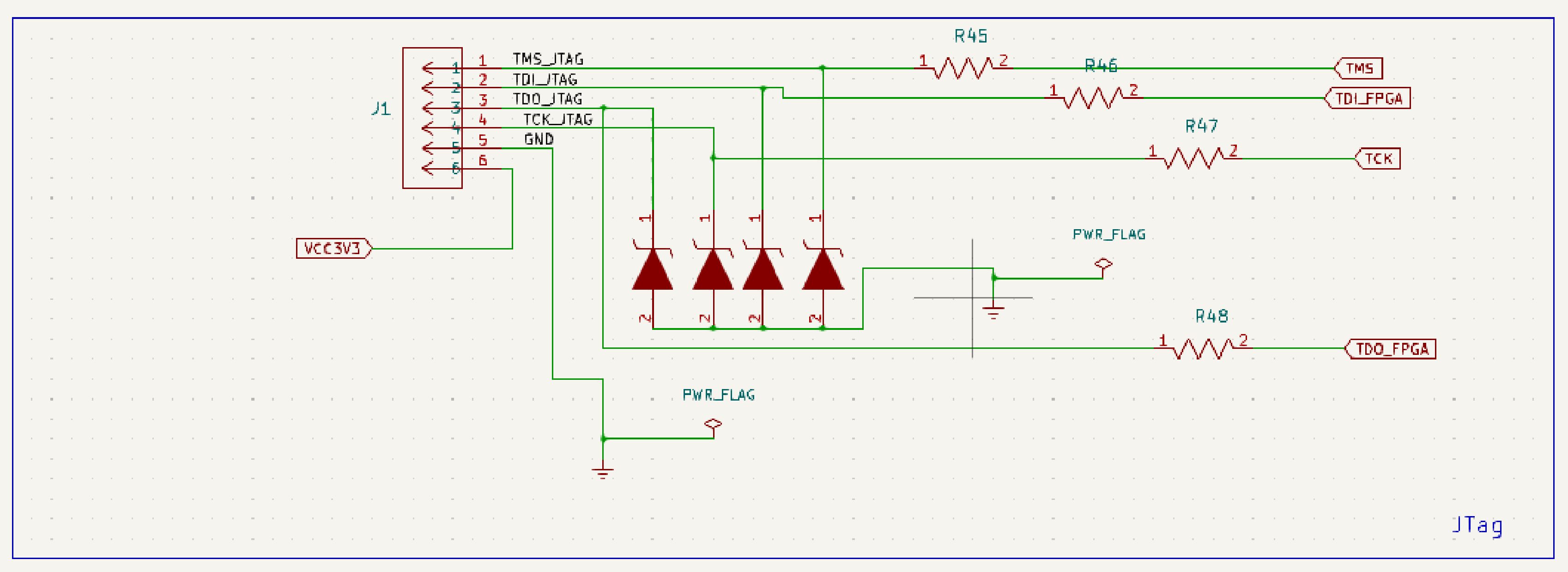
7 Segment display



Pmods

Our custom PCB Board Design Process

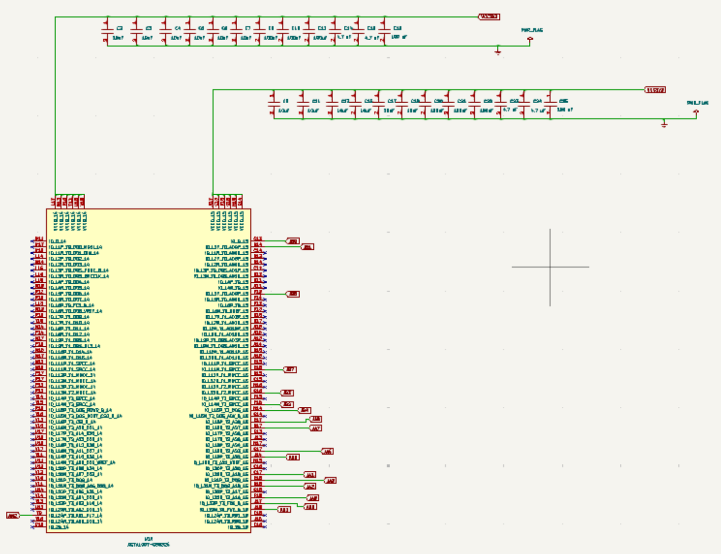
Here is the schematics for our PCB

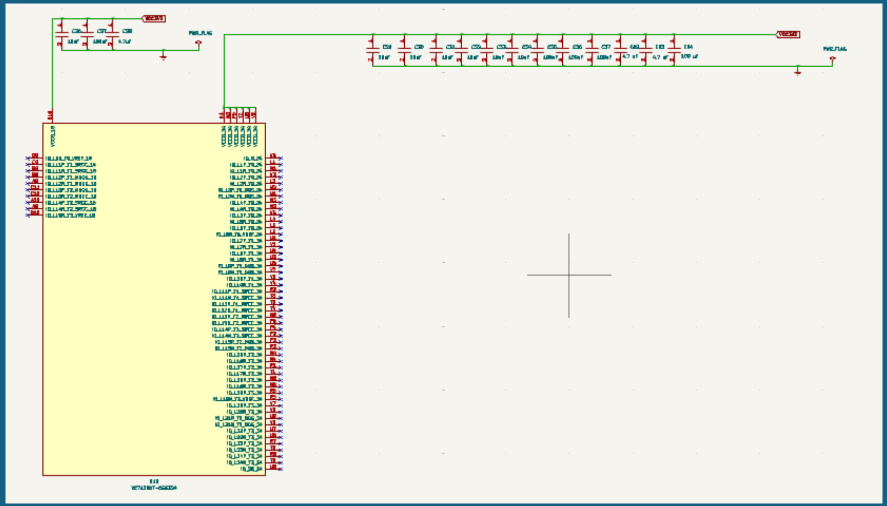


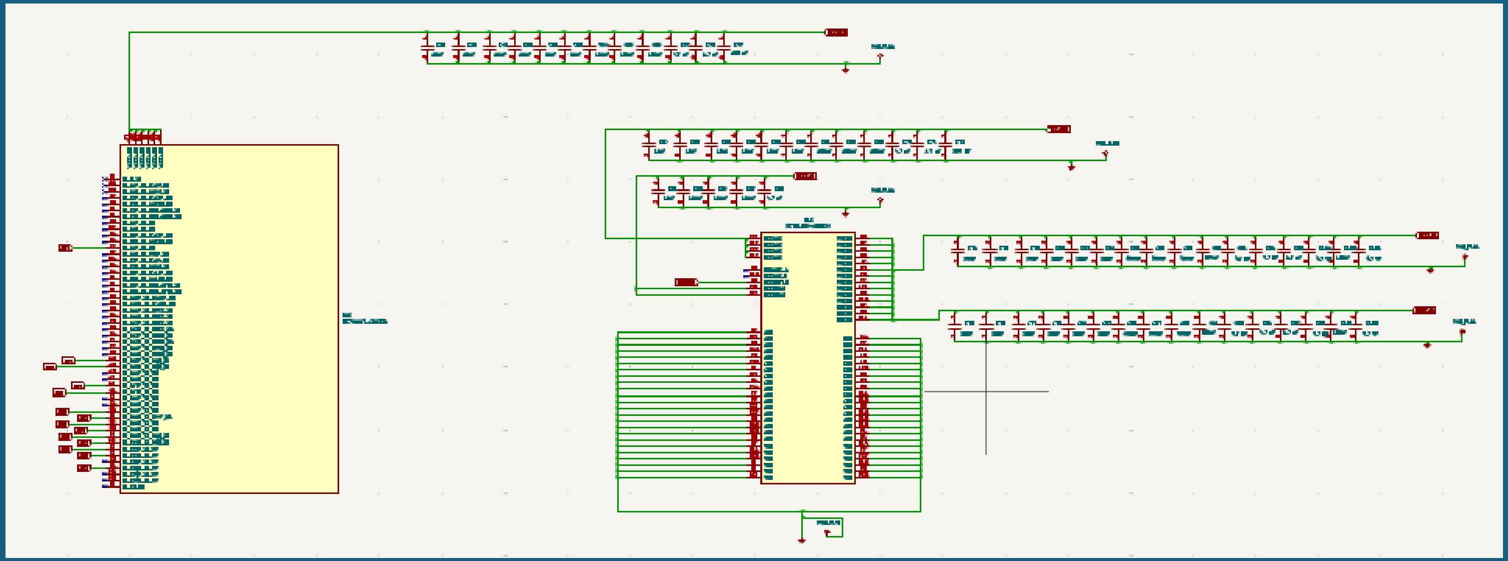
Jtag interface for programming and power supply

Our custom PCB Board Design Process

Here is the schematics for our PCB

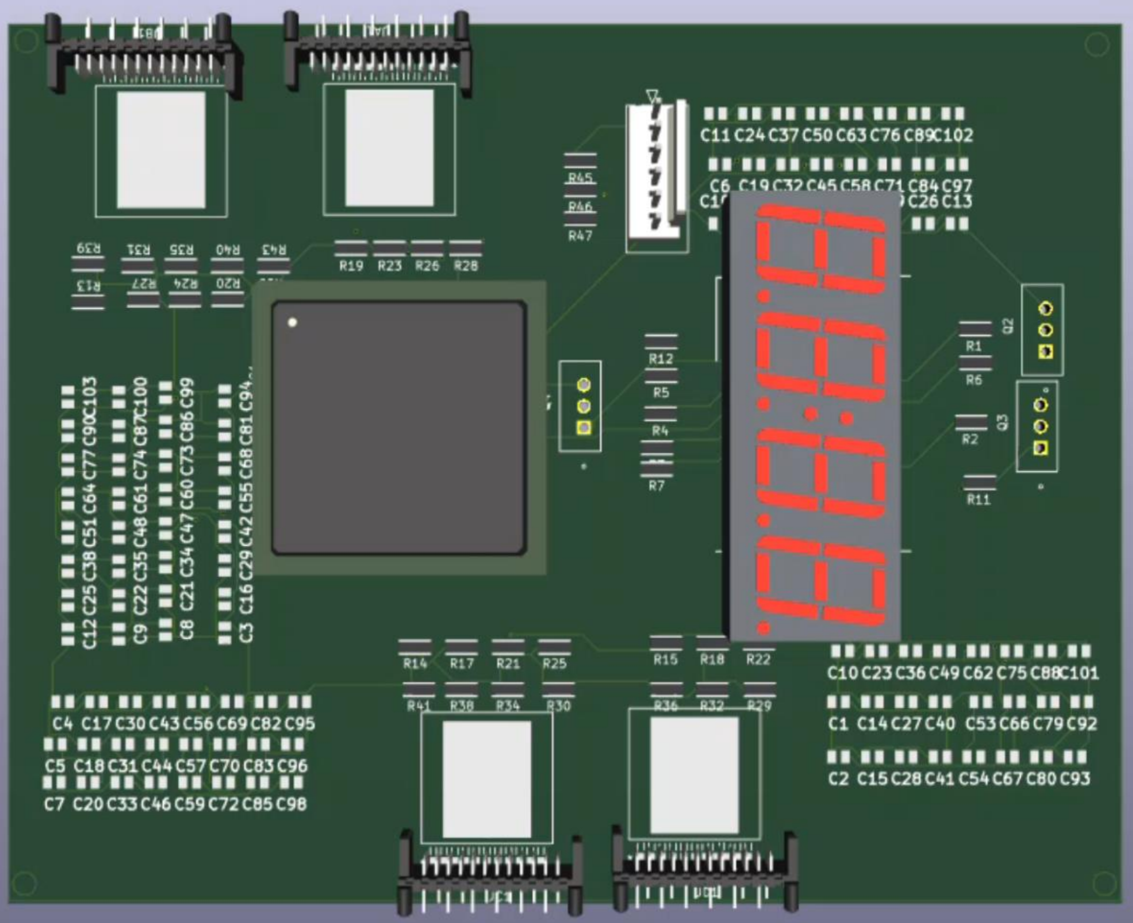






FPGA with power and pins

Our custom PCB Board final design


C11 C24 C37 C50 C63 C76 C89 C102
C6 C19 C32 C45 C58 C71 C84 C97
C26 C13
R45
R46
R47
R39 R31 R35 R40 R43
R13 R27 R24 R20
R19 R23 R26 R28
R12
R5
R4
R7
Q2
R1
R6
Q3
R2
R11
C12 C25 C38 C51 C64 C77 C90 C103
C9 C22 C35 C48 C61 C74 C87 C100
C8 C21 C34 C47 C60 C73 C86 C99
C3 C16 C29 C42 C55 C68 C81 C94
R14 R17 R21 R25
R41 R38 R34 R30
R15 R18 R22
R36 R32 R29
C10 C23 C36 C49 C62 C75 C88 C101
C1 C14 C27 C40 C53 C66 C79 C92
C2 C15 C28 C41 C54 C67 C80 C93
C4 C17 C30 C43 C56 C69 C82 C95
C5 C18 C31 C44 C57 C70 C83 C96
C7 C20 C33 C46 C59 C72 C85 C98