Finite State Machine in VHDL
Lab Report 3

Lab Section: 003
Group 29

Zhiyi Sun 260768094
Nariman Zendehrooh 260700556
Mcgill University, Montreal, Quebec, Canada

zhiyi.sun@mail.mcgill.ca
Nariman.zendehroohKermani@mail.McGill.ca

1.0 Table of Contents

2.0 Finite State Machine Diagram

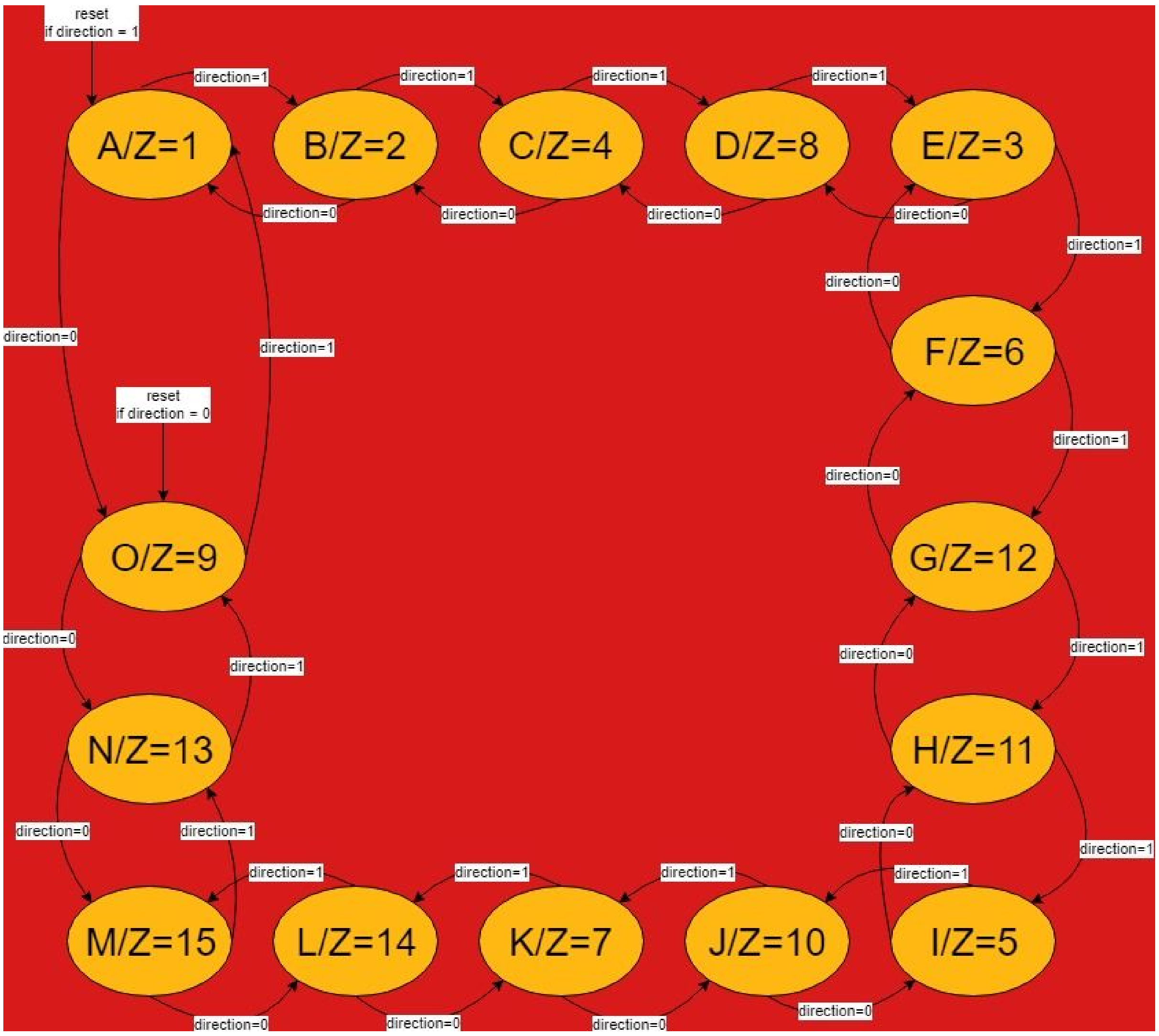


Figure 2.0-1: Finite Machine State Diagram

3.0 Finite State Machine and the Stimulation

The circuit is Moore's state machine means that the output of the state machine depends only on present state. Also, the output of the state machine is only updated at the positive clock edge. In the circuit, each state is represented by a letter from A to O. As in Moore's state machine, each of the states has value as the output which is chosen by the given sequence of numbers. The circuit takes 4 inputs to include enable, direction, reset and clk. Enable input starts the circuit. Direction input sets the transition between states. The asynchronous reset input is an active low element. When the reset is 0, it sets the state to the initial state. Otherwise, the circuit acts as usual. Clk input keeps track of time. The circuit has one output, count, which is the output that we have given to each state. The process block is started by setting reset to 0. As a result, the initial state of the circuit will be set. If the direction is 1, the initial state will be 1. Otherwise, it will be set to 9. After the circuit is enabled if the direction is 1, it will go clockwise as in the figure above. Otherwise, it will go counter-clockwise. When the state changes, it will produce an output which is the value that is given to each state. For example, after entering A, it will produce 1 as the output. Then, the circuit decides where to go as the next state by the direction that is given to it.
We simulated the state machine to check whether it starts at the correct state and moves in the right direction considering the given direction. The clock process is created to keep track of time in the circuit. The simulation process is used to assign different values to the direction to test the state machine's functionality. The simulation once will be reset and 1 will be assigned to it as the initial direction, so it starts at A and goes all the way to O, and then we change the direction to test if it will return back. Then, it will be reset once again. However, this time 0 will be assigned as the initial direction, so it starts at O and goes all the way to A and returns back by changing the direction. In each of the states, the simulation shows the assigned output to that state.

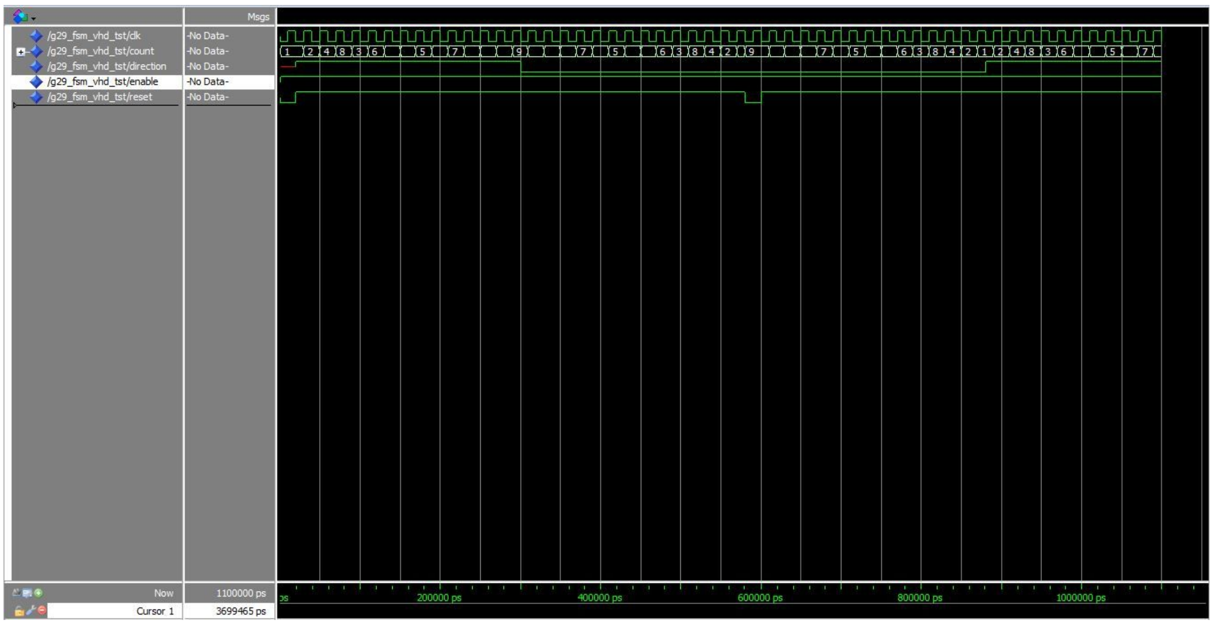


Figure 3.0-1: Testbench simulation of the finite state machine

4.0 A Description of the Multi-mode Counter

The aim of the multi-mode counter is to test the function of a FSM on a FPGA board (similar to the function of "stopwatch" code in the previous lab). The circuit consists of three components (multi_mode_ocunter, clockDivider and 7-segment decoder), and this circuit maps its inputs (start, stop, direction and reset) to the variables in each components. The process then overall describes what the circuit will do when it receives different input signals, such as setting the masterstart as "0" when reset or stop is "0" (the clock stops) and when strat is "0" the masterstart is "1" (the clock continues and the FSM continues to change its states). The outputs of the FSM is then converted to decimals by using the temp1 and temp2, and is then displayed on the FPGA board by using the 7-segments-decoder (one for each output).

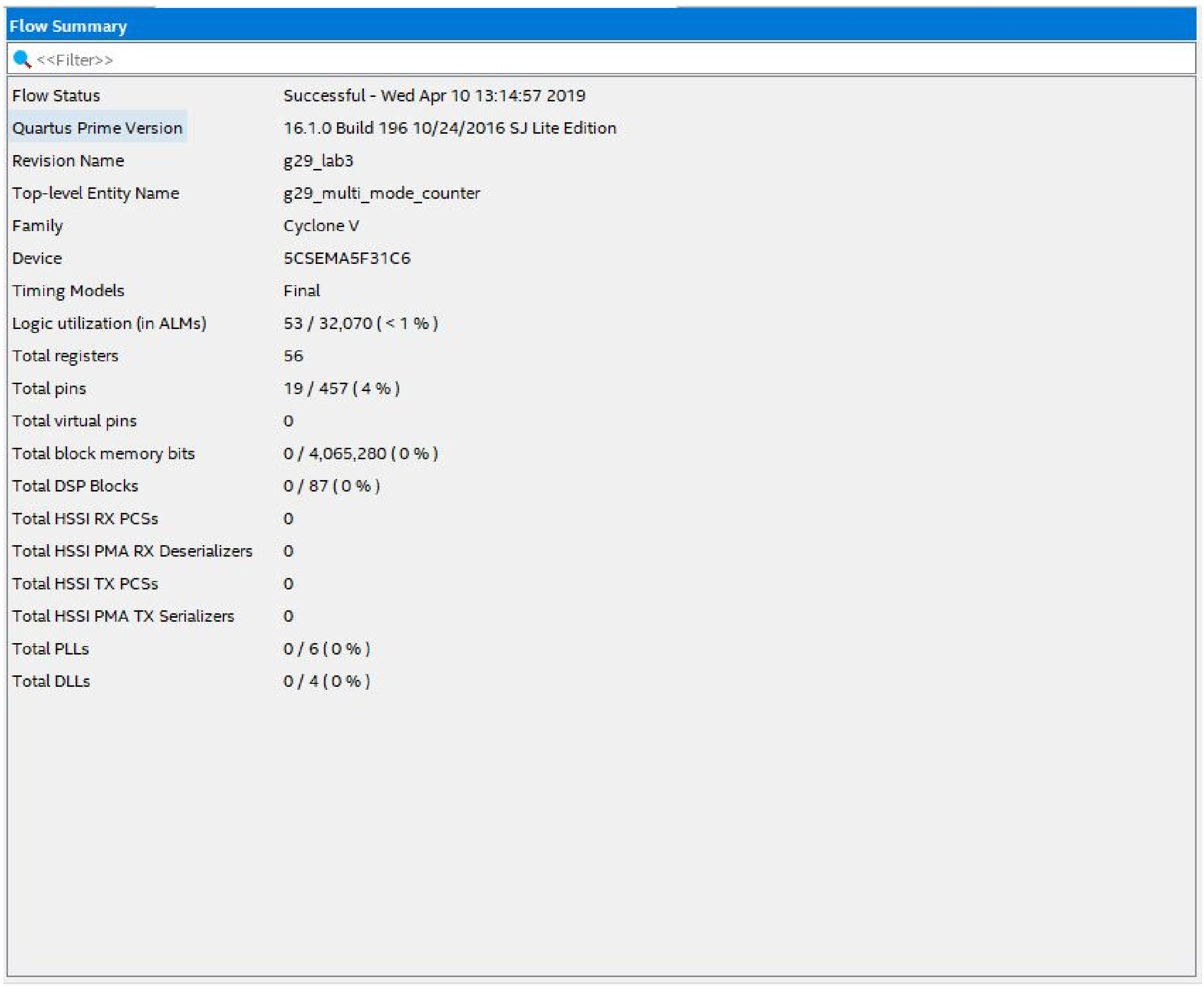
5.0 A Discussion of How the Multi-mode Counter Is Tested on the FPGA Board

The multi-mode counter code is tested using the Altera DE1-SoCboard (FPGA board). The code for multi-mode counter circuit is first compiled in the Quartus software; once it is done, we map the code to the Altera DE1-SoCboard by assigning the 7-segments declared in the code to their corresponding pin locations, the clock to the pin location for a 50 Mhz clock on the manual, and the start, stop and reset to the pin locations of the push buttons. In addition, the direction signal will be assigned to one of the switches on the board.

After setting up the FPGA board, we press the start button and the two rightmost LED displayers will start to follow the sequence given in the assignment. If we change the state of the switch (the switch assigned to the direction signal), the displayers will then show the sequence in a reversed order. If we press the stop button, the board will stop displaying the next output (stay at the current value); if the reset button is pressed, the values of the two rightmost LED displayers will be reset to 0. If the direction switch is taken off (0), the outputs will be displayed in a reversed order.

The multi-mode counter serves just as a tester for our FSM code. The correct sequence of the outputs indicates the FSM is working (changes its states) properly.

6.0 Summary of the FPGA Resource Utilization and the RTL Schematic Diagram



Flow Status	Successful - Wed Apr 10 13:14:57 2019
Quartus Prime Version	16.1.0 Build 196 10/24/2016 SJ Lite Edition
Revision Name	g29_lab3
Top-level Entity Name	g29_multi_mode_counter
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	53 / 32,070 (< 1 %)
Total registers	56
Total pins	19 / 457 (4 %)
Total virtual pins	0
Total block memory bits	0 / 4,065,280 (0 %)
Total DSP Blocks	0 / 87 (0 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 4 (0 %)

Fig.6.1 This figure gives the flow summary of the multi_mode_counter

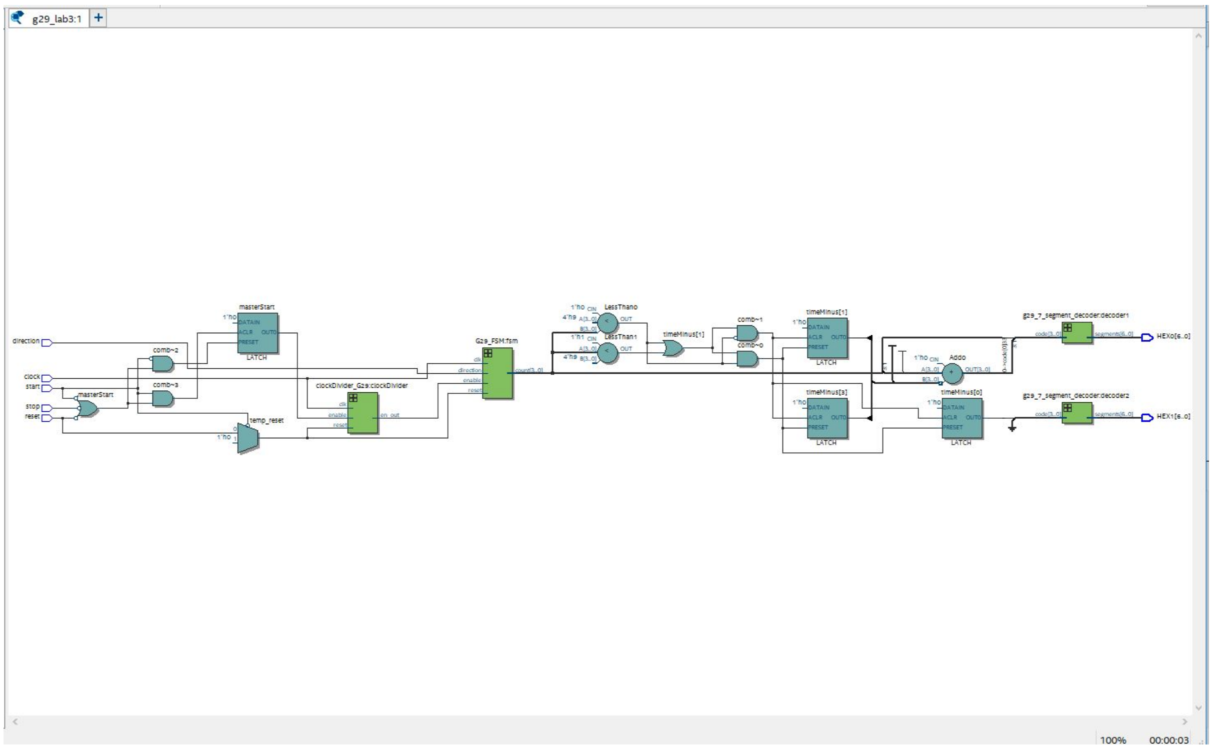


Fig.6.2 This figure shows the overview of the entire RTL diagram

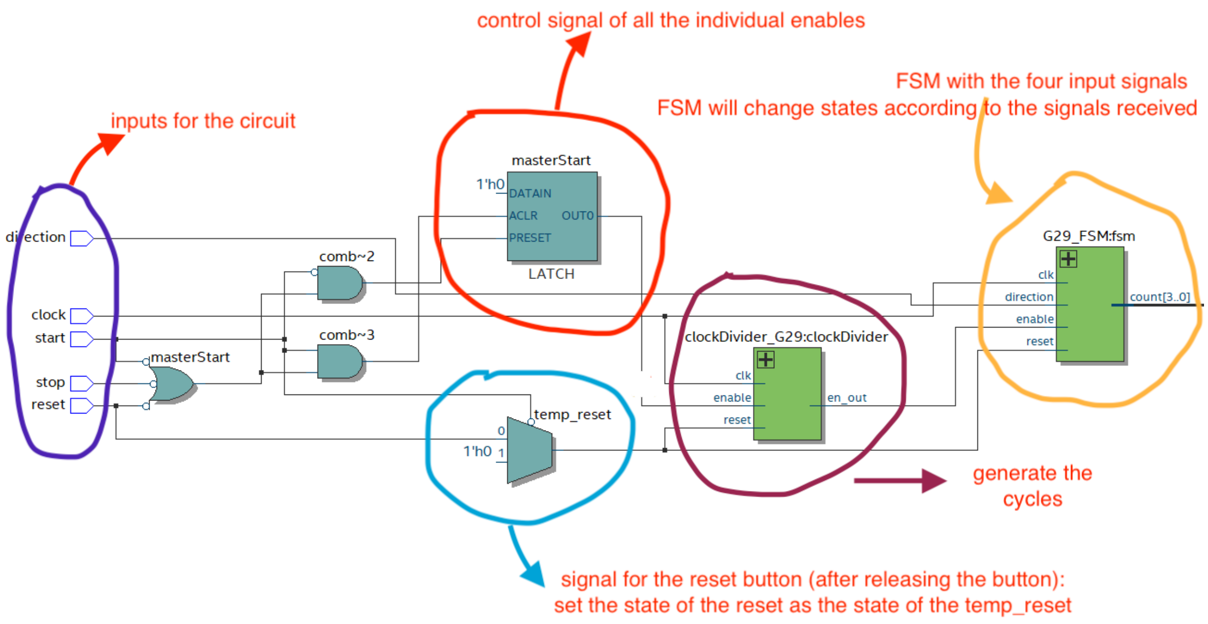

control signal of all the individual enables
FSM with the four input signals
FSM will change states according to the signals received
inputs for the circuit
masterStart
1'h0
DATAIN
ACLR
OUT0
PRESET
LATCH
direction
clock
start
stop
reset
comb~2
comb~3
masterStart
G29_FSM:fsm
clk
direction
enable
reset
count[3..0]
clockDivider_G29:clockDivider
clk
enable
reset
en_out
temp_reset
0
1'h0
1
generate the cycles
signal for the reset button (after releasing the button):
set the state of the reset as the state of the temp_reset

Fig.6.3 This figure shows part of the entire RTL diagram

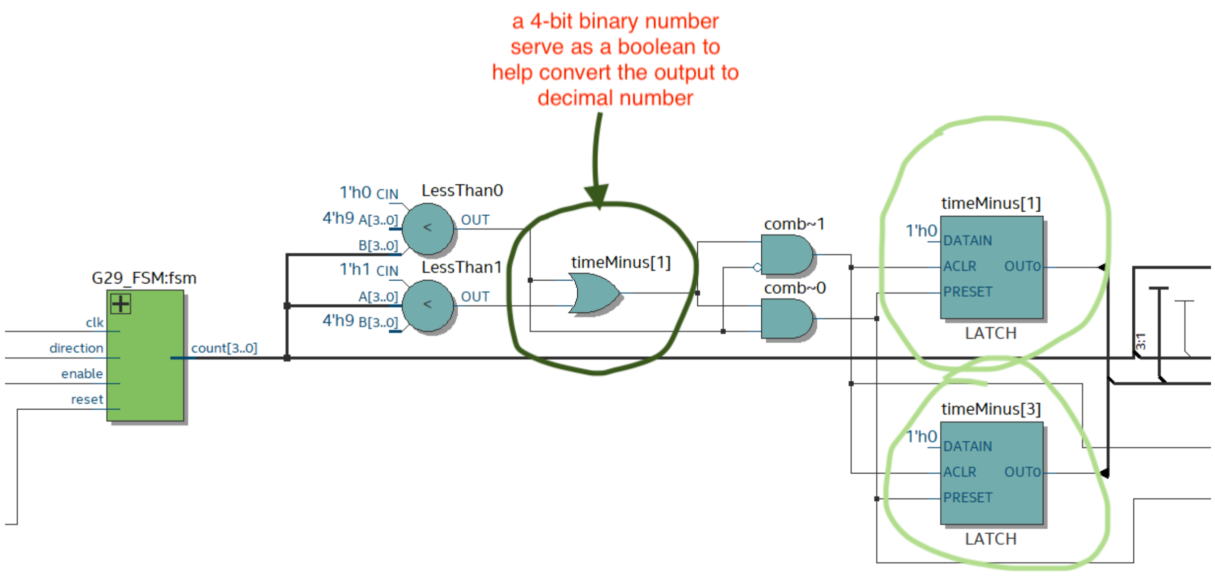


Fig.6.4 This figure shows part of the entire RTL diagram

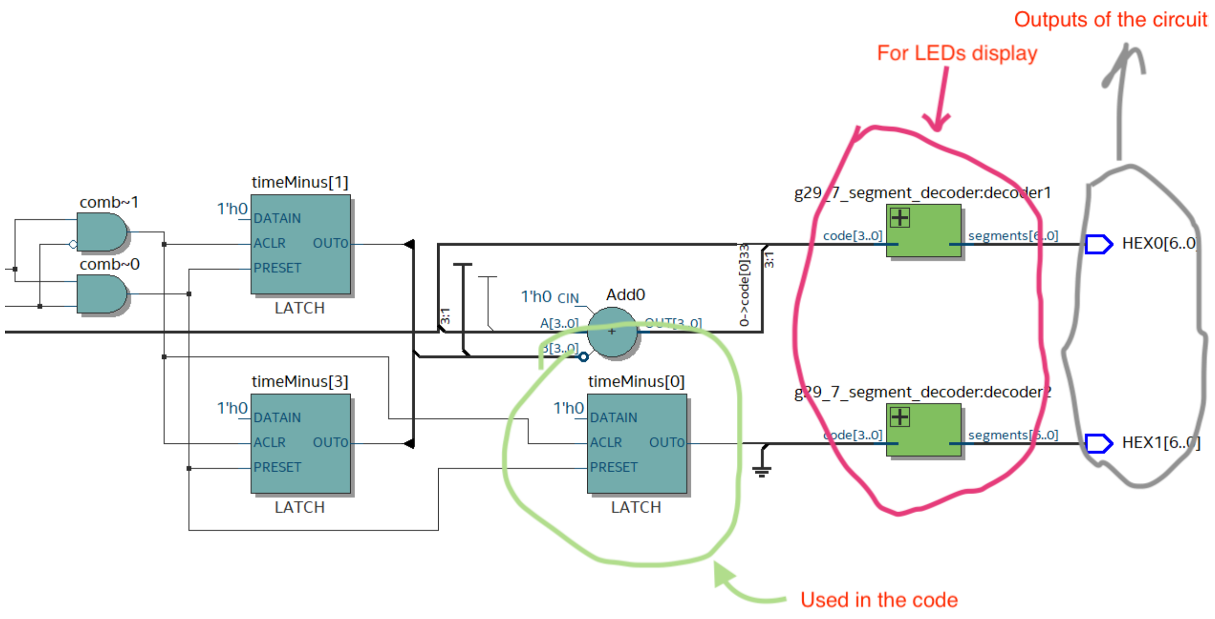
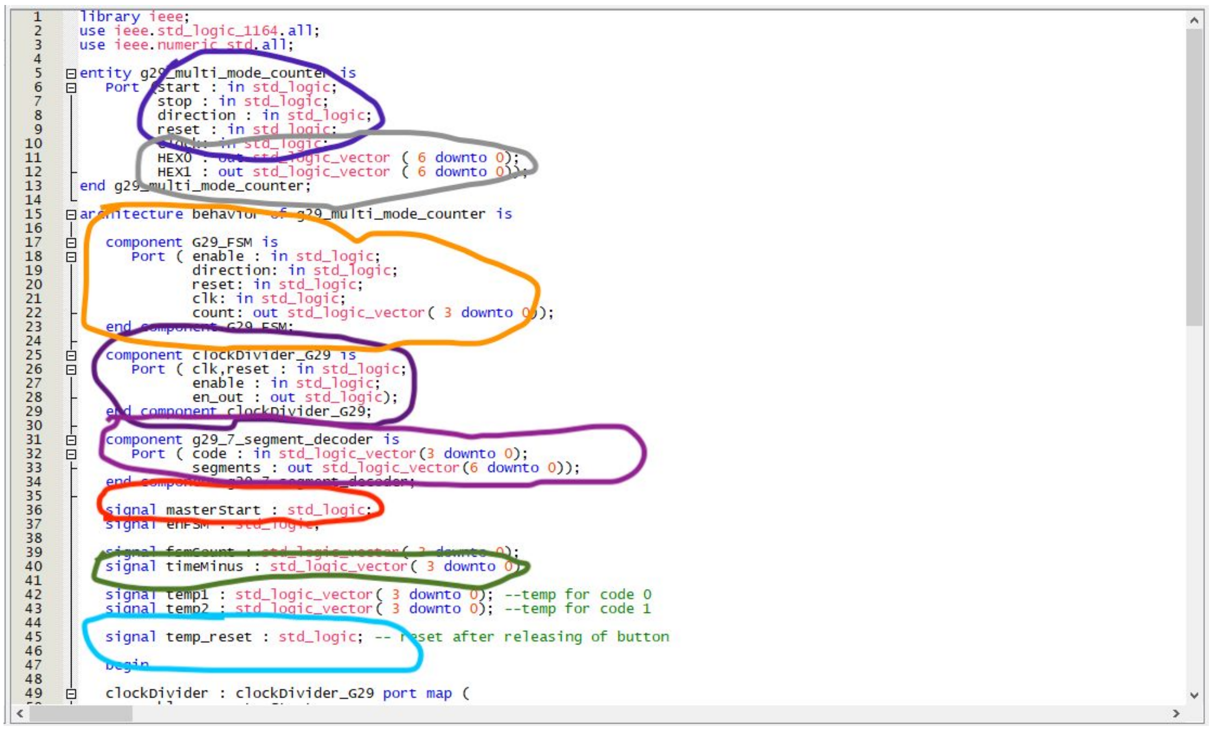
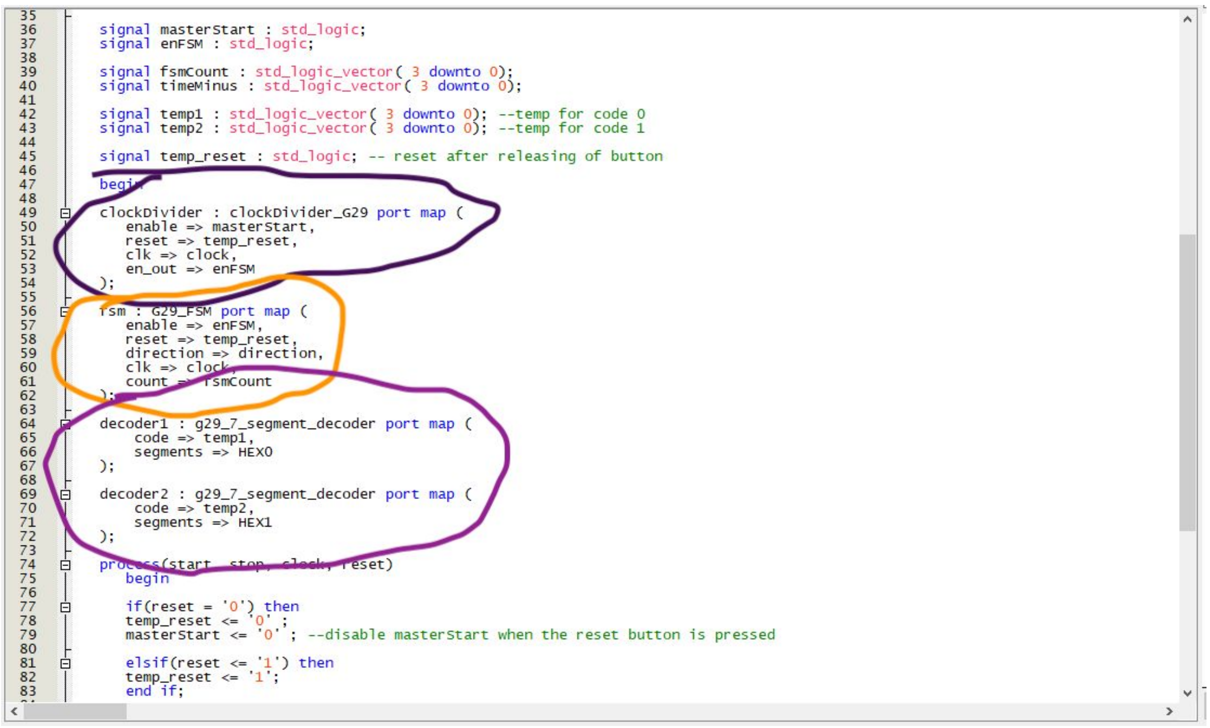


Fig.6.5 This figure shows part of the entire RTL diagram



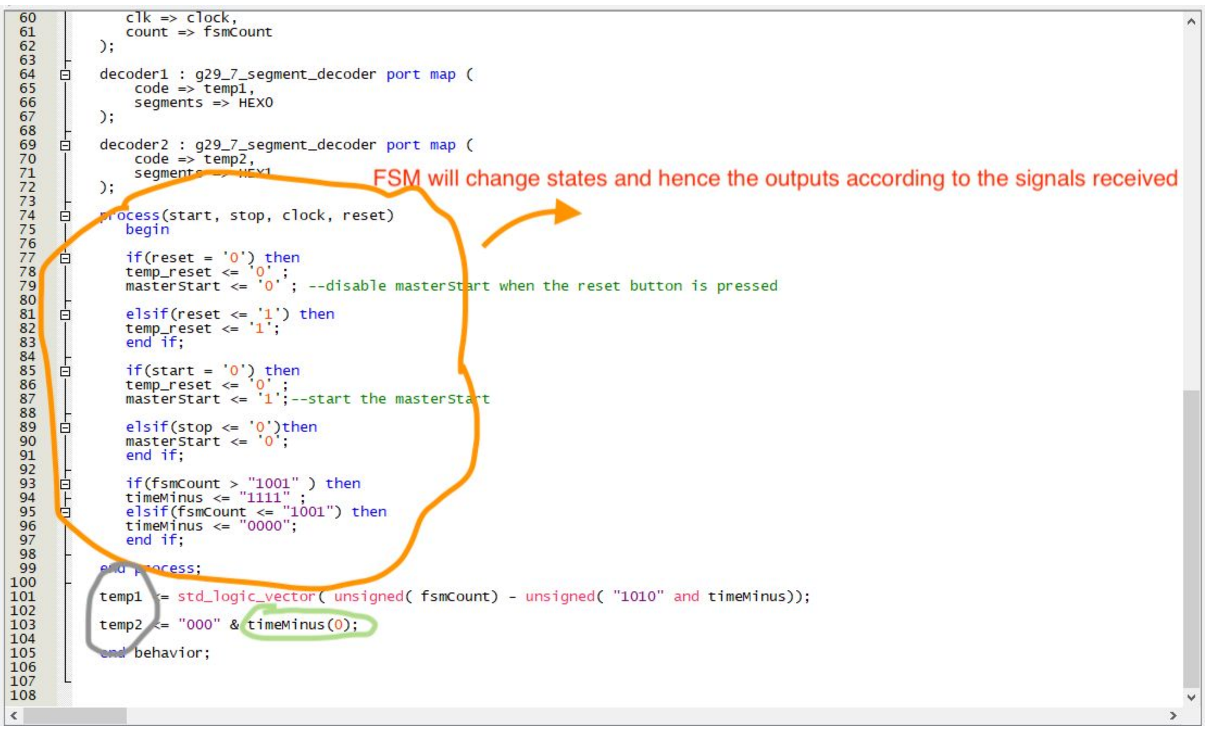
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity g29_multi_mode_counter is
   Port (start : in std_logic;
         stop : in std_logic;
         direction : in std_logic;
         reset : in std_logic;
         clock: in std_logic;
         HEX0 : out std_logic_vector ( 6 downto 0);
         HEX1 : out std_logic_vector ( 6 downto 0));
end g29_multi_mode_counter;

architecture behavior of g29_multi_mode_counter is

   component G29_FSM is
      Port ( enable : in std_logic;
             direction: in std_logic;
             reset: in std_logic;
             clk: in std_logic;
             count: out std_logic_vector( 3 downto 0));
   end component G29_FSM;

   component clockDivider_G29 is
      Port ( clk,reset : in std_logic;
             enable : in std_logic;
             en_out : out std_logic);
   end component clockDivider_G29;

   component g29_7_segment_decoder is
      Port ( code : in std_logic_vector(3 downto 0);
             segments : out std_logic_vector(6 downto 0));
   end component g29_7_segment_decoder;

   signal masterStart : std_logic;
   signal enFSM : std_logic;

   signal fsmCount : std_logic_vector( 3 downto 0);
   signal timeMinus : std_logic_vector( 3 downto 0);

   signal temp1 : std_logic_vector( 3 downto 0); --temp for code 0
   signal temp2 : std_logic_vector( 3 downto 0); --temp for code 1

   signal temp_reset : std_logic; -- reset after releasing of button

   begin

   clockDivider : clockDivider_G29 port map (
```

Fig.6.6 This figure shows part of the multi_mode_counter code



```
   signal masterStart : std_logic;
   signal enFSM : std_logic;

   signal fsmCount : std_logic_vector( 3 downto 0);
   signal timeMinus : std_logic_vector( 3 downto 0);

   signal temp1 : std_logic_vector( 3 downto 0); --temp for code 0
   signal temp2 : std_logic_vector( 3 downto 0); --temp for code 1

   signal temp_reset : std_logic; -- reset after releasing of button

   begin

   clockDivider : clockDivider_G29 port map (
      enable => masterStart,
      reset => temp_reset,
      clk => clock,
      en_out => enFSM
   );

   fsm : G29_FSM port map (
      enable => enFSM,
      reset => temp_reset,
      direction => direction,
      clk => clock,
      count => fsmCount
   );

   decoder1 : g29_7_segment_decoder port map (
       code => temp1,
       segments => HEX0
   );

   decoder2 : g29_7_segment_decoder port map (
       code => temp2,
       segments => HEX1
   );

   process(start, stop, clock, reset)
      begin

      if(reset = '0') then
      temp_reset <= '0' ;
      masterStart <= '0' ; --disable masterStart when the reset button is pressed

      elsif(reset <= '1') then
      temp_reset <= '1';
      end if;
```

Fig.6.7 This figure shows part of the multi_mode_counter code


FSM will change states and hence the outputs according to the signals received

```
        clk => clock,
        count => fsmCount
    );

    decoder1 : g29_7_segment_decoder port map (
         code => temp1,
         segments => HEX0
    );

    decoder2 : g29_7_segment_decoder port map (
         code => temp2,
         segments => HEX1
    );

    process(start, stop, clock, reset)
        begin

        if(reset = '0') then
        temp_reset <= '0' ;
        masterStart <= '0' ; --disable masterStart when the reset button is pressed

        elsif(reset <= '1') then
        temp_reset <= '1';
        end if;

        if(start = '0') then
        temp_reset <= '0' ;
        masterStart <= '1';--start the masterStart

        elsif(stop <= '0')then
        masterStart <= '0';
        end if;

        if(fsmCount > "1001" ) then
        timeMinus <= "1111" ;
        elsif(fsmCount <= "1001") then
        timeMinus <= "0000";
        end if;

    end process;

    temp1 <= std_logic_vector( unsigned( fsmCount) - unsigned( "1010" and timeMinus));

    temp2 <= "000" & timeMinus(0);

    end behavior;
```

Fig.6.8 This figure shows part of the multi_mode_counter code

Same colours indicate corresponding parts in the code and the schematic diagram.