PART 1:

VHDL code of the clock divider of 1 HZ:

```
----------------------------------------------------------------------------------
-- Company: Department of Electrical and Computer Engineering, University of Alberta
-- Engineer: Xinyue Chen
-- Create Date: 2020/11/09 19:57:43
-- Design Name: “01101” sequence detector
-- Module Name: clock_divider - Behavioral
-- Project Name: “01101” sequence detector
-- Target Devices: ZYBO Z7-10
-- Tool Versions:
-- Description:
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity clk_divider is
    Port ( clk_in : in STD_LOGIC;
           clk_out : out STD_LOGIC);
end clk_divider;

architecture Behavioral of clk_divider is
signal clock_out : std_logic := '0';
signal count : integer := 1;
begin
    process(clk_in)
    begin
        if clk_in='1' and clk_in'event then
            count <= count + 1;
            if(count =62500000) then
                clock_out <= not clock_out;
                count <= 1;
            end if;
        end if;
    clk_out <= clock_out;
    end process;
end Behavioral;
```

VHDL code of “01101” sequence detector:

```
----------------------------------------------------------------------------------
-- Company: Department of Electrical and Computer Engineering, University of Alberta
```

```
    -- Engineer: Xinyue Chen
    -- Create Date: 2020/11/08 11:34:32
    -- Design Name: “01101” sequence detector
    -- Module Name: sequence_detector - Behavioral
    -- Project Name: “01101” sequence detector
    -- Target Devices: ZYBO Z7-10
    -- Tool Versions:
    -- Description:
    ----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity sequence_detector is
    Port ( input_sequence : in STD_LOGIC;
           clk : in STD_LOGIC;
           reset : in STD_LOGIC;
           output_sequence : out STD_LOGIC);
end sequence_detector;

architecture finite_state_machine of sequence_detector is
component clk_divider is
  Port ( clk_in : in std_logic;
         clk_out : out std_logic);
end component clk_divider;

type state_type is (idle,A,B,C,D,E);
-- idle is the initial state
-- A: '0' detected, B: "01" detected, C: "011" detected, D: "0110" detected, E: "01101" detected.
signal clk_out_component : std_logic;
signal current_state, next_state : state_type;
begin
label_clock_divider: component clk_divider port map(clk_in => clk,
                                                    clk_out => clk_out_component);
sequ_logic: process (clk_out_component,reset) is
begin
if (reset ='1') then
current_state <= idle;
elsif (clk_out_component'event and clk_out_component='1') then
current_state <= next_state;
end if;
end process sequ_logic;

comb_logic: process (current_state,input_sequence) is
begin
```

```
output_sequence <= '0';
case current_state is
when idle =>
if (input_sequence = '0') then
next_state <= A;
output_sequence <= '0';
else
next_state <= idle;
output_sequence <= '0';
end if;

when A =>
if (input_sequence = '1') then
next_state <= B;
output_sequence <= '0';
else
next_state <= A;
output_sequence <= '0';
end if;

when B =>
if (input_sequence = '1') then
next_state <= C;
output_sequence <= '0';
else
next_state <= A;
output_sequence <= '0';
end if;

when C =>
if (input_sequence = '0') then
next_state <= D;
output_sequence <= '0';
else
next_state <= idle;
output_sequence <= '0';
end if;

when D =>
if (input_sequence = '1') then
next_state <= E;
output_sequence <= '1';
else
next_state <= A;
```

```
output_sequence <= '0';
end if;

when E =>
if (input_sequence = '1') then
next_state <= C;
output_sequence <= '0';
else
next_state <= A;
output_sequence <= '0';
end if;
end process comb_logic;
end finite_state_machine;
```

VHDL testbench of “01101” sequence detector:

```
----------------------------------------------------------------------------------
-- Company: Department of Electrical and Computer Engineering, University of Alberta
-- Engineer: Xinyue Chen
-- Create Date: 2020/11/09 20:18:59
-- Design Name: “01101” sequence detector
-- Module Name: sequence_detector_tb - Behavioral
-- Project Name: “01101” sequence detector
-- Target Devices: ZYBO Z7-10
-- Tool Versions:
-- Description:
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity sequence_detector_tb is
end sequence_detector_tb;

architecture Behavioral of sequence_detector_tb is
component sequence_detector is
      port ( input_sequence : in std_logic;
            clk : in std_logic;
            reset : in std_logic;
            output_sequence : out std_logic);
end component sequence_detector;
    signal clk_tb : std_logic := '0';
    signal reset_tb : std_logic := '0';
    signal input_sequence_tb : std_logic := '0';
    signal output_sequence_tb : std_logic;
    constant clock_period : time := 8 ns;
```

```
begin
component_sequ: component sequence_detector port map( input_sequence => input_sequence_tb,
                                                       clk => clk_tb,
                                                       reset => reset_tb,
                                                       output_sequence => output_sequence_tb);
----Clock process
clock: process
begin
        clk_tb <='0';
        wait for clock_period/2;
        clk_tb <='1';
        wait for clock_period/2;
end process clock;

stim_proc: process
   begin
   reset_tb <= '1';
   wait for 16 ns;
   reset_tb <= '0';
   wait for 16 ns;
   input_sequence_tb <= '1';
   wait for 16 ns;
   input_sequence_tb <= '1';
   wait for 16 ns;
   input_sequence_tb <= '1';
   wait for 16 ns;
   input_sequence_tb <= '0';
   wait for 16 ns;
   input_sequence_tb <= '0';
   wait for 16 ns;
   input_sequence_tb <= '1';
   wait for 16 ns;
   input_sequence_tb <= '0';
   wait for 16 ns;
   input_sequence_tb <= '1';
   wait for 16 ns;
   input_sequence_tb <= '1';
   wait for 16 ns;
   input_sequence_tb <= '0';
   wait for 16 ns;
   input_sequence_tb <= '1';
   wait for 16 ns;
   input_sequence_tb <= '1';
   wait for 16 ns;
```

```
    input_sequence_tb <= '0';
    wait for 16 ns;
    input_sequence_tb <= '1';
    wait for 16 ns;
    input_sequence_tb <= '0';
    wait for 16 ns;
    input_sequence_tb <= '1';
    wait for 16 ns;
    input_sequence_tb <= '1';
    wait for 16 ns;
    input_sequence_tb <= '0';
    wait for 16 ns;
    input_sequence_tb <= '0';
    wait for 16 ns;
    input_sequence_tb <= '1';
    wait for 16 ns;
    input_sequence_tb <= '1';
    wait for 16 ns;
    input_sequence_tb <= '1';
       wait;
     end process;
end Behavioral;
```

Constraint file of “01101” sequence detector:

In constraint file, each entity port is mapped to the Zybo Z7 board as shown below:

```
##Clock signal
set_property -dict {PACKAGE_PIN K17       IOSTANDARD LVCMOS33} [get_ports { clk }];
#create_clock -period 8.000 -name sys_clk_pin -waveform {0.000 4.000} -add [get_ports { }];
##Switches
set_property -dict {PACKAGE_PIN G15       IOSTANDARD LVCMOS33} [ get_ports { input_sequence }];
#Sch=sw[0]
##Buttons
set_property -dict { PACKAGE_PIN K18       IOSTANDARD LVCMOS33 } [get_ports { reset }];
#IO_L12N_T1_MRCC_35 Sch=btn[0]
##LEDs
set_property -dict {PACKAGE_PIN M14        IOSTANDARD LVCMOS33} [get_ports { output_sequence }];
```

PART 2:

VHDL code of Vending Machine (we use the same clock divider as demonstrated in PART 1 to view the result on FPGA board):

```
----------------------------------------------------------------------------------
```

```
-- Company: Department of Electrical and Computer Engineering, University of Alberta
-- Engineer: Xinyue Chen
-- Create Date: 2020/11/08 16:28:53
-- Design Name: vending machine for chocolate and chips
-- Module Name: vending_machine - Behavioral
-- Project Name: vending machine for chocolate and chips
-- Target Devices: ZYBO Z7-10
-- Tool Versions:
-- Description:
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity vending_machine is
    Port ( clk : in STD_LOGIC;
           reset : in STD_LOGIC;
           item_select : in STD_LOGIC;
           coins_in : in STD_LOGIC_VECTOR (1 downto 0);
           change_out: out STD_LOGIC_VECTOR (1 downto 0);
-- To display the output change ("01" is 1$,"10" is 2$ and "11" is 5$)
           display_sum : out STD_LOGIC_VECTOR (6 downto 0);
           select_segment : out STD_LOGIC;
           chips : out STD_LOGIC;
           chocolate : out STD_LOGIC);
end vending_machine;

architecture finite_state_machine of vending_machine is
component clock_divider is
  Port ( clk_in : in std_logic;
           clk_out : out std_logic);
end component clock_divider;

type state_type is
(idle,ready_chocolate,ready_chips,A_1,B_1,A_2,B_2,D,dispense_choc,dispense_chip,change_choc_1,change_c
hoc_2,change_choc_1_2,change_choc_2_2,change_chip_1,change_chip_2,change_chip_1_2,change_chip_2_2
);
signal current_state,next_state : state_type;
signal clock_out_component : std_logic;

begin
label_clock_divider: component clock_divider port map(clk_in => clk,
                                                      clk_out => clock_out_component);

sequ_logic: process (clock_out_component,reset) is
```

```
begin
if (reset='1') then
current_state <= idle;
elsif (clock_out_component'event and clock_out_component='1') then
current_state <= next_state;
end if;
end process sequ_logic;

comb_logic: process (current_state,item_select,coins_in) is
begin
case current_state is
when idle =>
                change_out <= "00";
                display_sum <= "0000000";
                select_segment <= '0';
                chips <= '0';
                chocolate <= '0';
if (item_select='0') then
next_state <= ready_chocolate;
else
next_state <= ready_chips;
end if;

when ready_chocolate =>
                change_out <= "00";
                display_sum <= "0000000";
                select_segment <= '0';
                chips <= '0';
                chocolate <= '0';
if (coins_in="01") then
next_state <= A_1;
elsif (coins_in="10") then
next_state <= B_1;
elsif (coins_in="11") then
next_state <= change_choc_2;
elsif (coins_in="00") then
next_state <= ready_chocolate;
end if;

when A_1 =>
                change_out <= "00";
                display_sum <= "0000110";
                select_segment <= '0';
                chips <= '0';
```

```
                chocolate <= '0';
if (coins_in="01") then
next_state <= B_1;
elsif (coins_in="10") then
next_state <= dispense_choc;
elsif (coins_in="11") then
next_state <= change_choc_1_2;
elsif (coins_in="00") then
next_state <= A_1;
end if;

when B_1 =>
                change_out <= "00";
                display_sum <= "1011011";
                select_segment <= '0';
                chips <= '0';
                chocolate <= '0';
if (coins_in="01") then
next_state <= dispense_choc;
elsif(coins_in="10") then
next_state <= change_choc_1;
elsif (coins_in="11") then
next_state <= change_choc_2_2;
elsif (coins_in="00") then
next_state <= B_1;
end if;

when ready_chips =>
                change_out <= "00";
                display_sum <= "0000000";
                select_segment <= '0';
                chips <= '0';
                chocolate <= '0';
if (coins_in="01") then
next_state <= A_2;
elsif (coins_in="10") then
next_state <= B_2;
elsif (coins_in="11") then
next_state <= change_chip_1;
elsif (coins_in="00") then
next_state <= ready_chips;
end if;

when A_2 =>
```

```
                change_out <= "00";
                display_sum <= "0000110";
                select_segment <= '0';
                chips <= '0';
                chocolate <= '0';
if (coins_in="01") then
next_state <= B_2;
elsif (coins_in="10") then
next_state <= D;
elsif (coins_in="11") then
next_state <= change_chip_2;
elsif (coins_in="00") then
next_state <= A_2;
end if;

when B_2 =>
                change_out <= "00";
                display_sum <= "1011011";
                select_segment <= '0';
                chips <= '0';
                chocolate <= '0';
if (coins_in="01") then
next_state <= D;
elsif (coins_in="10") then
next_state <= dispense_chip;
elsif (coins_in="11") then
next_state <= change_chip_1_2;
elsif (coins_in="00") then
next_state <= B_2;
end if;

when D =>
                change_out <= "00";
                display_sum <= "1001111";
                select_segment <= '0';
                chips <= '0';
                chocolate <= '0';
if (coins_in="01") then
next_state <= dispense_chip;
elsif (coins_in="10") then
next_state <= change_chip_1;
elsif (coins_in="11") then
next_state <= change_chip_2_2;
elsif (coins_in="00") then
```

```
next_state <= D;
end if;

when dispense_choc =>
                change_out <= "00";
                display_sum <= "1001111";
                select_segment <= '0';
                chips <= '0';
                chocolate <= '1';
next_state <= idle;

when dispense_chip =>
                change_out <= "00";
                display_sum <= "1100110";
                select_segment <= '0';
                chips <= '1';
                chocolate <= '0';
next_state <= idle;

when change_choc_1 =>
                change_out <= "01";
                display_sum <= "1100110";
                select_segment <= '0';
                chips <= '0';
                chocolate <= '1';
next_state <= idle;

when change_chip_1 =>
                change_out <= "01";
                display_sum <= "1101101";
                select_segment <= '0';
                chips <= '1';
                chocolate <= '0';
next_state <= idle;

when change_choc_2 =>
                change_out <= "10";
                display_sum <= "1101101";
                select_segment <= '0';
                chips <= '0';
                chocolate <= '1';
next_state <= idle;

when change_chip_2 =>
```

```
                change_out <= "10";
                display_sum <= "1111101";
                select_segment <= '0';
                chips <= '1';
                chocolate <= '0';
next_state <= idle;

when change_choc_1_2 =>
                change_out <= "01";
                display_sum <= "1111101";
                select_segment <= '0';
                chips <= '0';
                chocolate <= '0';
next_state <= change_choc_2;

when change_chip_1_2 =>
                change_out <= "01";
                display_sum <= "0000111";
                select_segment <= '0';
                chips <= '0';
                chocolate <= '0';
next_state <= change_chip_2;

when change_choc_2_2 =>
                change_out <= "10";
                display_sum <= "0000111";
                select_segment <= '0';
                chips <= '0';
                chocolate <= '0';
next_state <= change_choc_2;

when change_chip_2_2 =>
                change_out <= "10";
                display_sum <= "1111111";
                select_segment <= '0';
                chips <= '0';
                chocolate <= '0';
next_state <= change_chip_2;
end case;
end process comb_logic;
end finite_state_machine;
```

VHDL testbench of the Vending Machine:

```
--------------------------------------------------------------------------------
```

```
-- Company: Department of Electrical and Computer Engineering, University of Alberta
-- Engineer: Xinyue Chen
-- Create Date: 2020/11/12 11:05:03
-- Design Name: vending machine for chocolate and chips
-- Module Name: vending_machine_tb - Behavioral
-- Project Name: vending machine for chocolate and chips
-- Target Devices: ZYBO Z7-10
-- Tool Versions:
-- Description:
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity vending_machine_tb is
end vending_machine_tb;

architecture Behavioral of vending_machine_tb is
component vending_machine is
Port( clk : in std_logic;
        reset : in std_logic;
        item_select : in std_logic;
        coins_in : in std_logic_vector (1 downto 0);
        change_out : out std_logic_vector (1 downto 0);
        display_sum : out std_logic_vector (6 downto 0);
        select_segment : out std_logic;
        chips : out std_logic;
        chocolate : out std_logic);
end component vending_machine;
--input
    signal clk_tb : std_logic := '0';
    signal reset_tb : std_logic := '0';
    signal item_select_tb : std_logic := '0';
    signal coins_in_tb : std_logic_vector (1 downto 0) := "00";
--output
    signal change_out_tb : std_logic_vector (1 downto 0);
    signal display_sum_tb : std_logic_vector (6 downto 0);
    signal select_segment_tb : std_logic;
    signal chocolate_tb : std_logic;
    signal chips_tb : std_logic;
    constant clock_period : time := 8 ns;
begin
component_vend: component vending_machine port map( clk => clk_tb,
                                                    reset => reset_tb,
                                                    item_select =>    item_select_tb,
```

```
                                        coins_in => coins_in_tb,
                                        change_out => change_out_tb,
                                        display_sum => display_sum_tb,
                                        select_segment => select_segment_tb,
                                        chocolate => chocolate_tb,
                                        chips => chips_tb);

----Clock process
clk: process
        begin
            clk_tb <='0';
            wait for clock_period/2;
            clk_tb <='1';
            wait for clock_period/2;
end process clk;
stim_proc: process
    begin
   reset_tb <= '0';
   item_select_tb <= '0';
   coins_in_tb    <= "00";
   wait for 16 ns;
   coins_in_tb    <= "01";
   wait for 16 ns;
   coins_in_tb    <= "01";
   wait for 16 ns;
   reset_tb <= '1';
   coins_in_tb    <= "00";
   wait for 16 ns;
   reset_tb <= '0';
   item_select_tb <= '0';
   coins_in_tb    <= "00";
   wait for 16 ns;
   coins_in_tb    <= "01";
   wait for 16 ns;
   coins_in_tb    <= "10";
   wait for 32 ns;
   coins_in_tb    <= "00";
   wait for 16 ns;
   coins_in_tb    <= "10";
   wait for 16 ns;
   coins_in_tb    <= "10";
   wait for 32 ns;
   coins_in_tb    <= "00";
   wait for 16 ns;
   coins_in_tb    <= "11";
```

```
    wait for 32 ns;
    item_select_tb <= '1';
    coins_in_tb    <= "00";
    wait for 16 ns;
    coins_in_tb    <= "10";
    wait for 16 ns;
    coins_in_tb    <= "11";
    wait for 48 ns;
    coins_in_tb    <= "00";
    wait for 16 ns;
    coins_in_tb    <= "10";
    wait for 16 ns;
    coins_in_tb    <= "01";
    wait for 16 ns;
    coins_in_tb    <= "11";
    wait for 48 ns;
    wait;
    end process;
end Behavioral;
```

Constraint file of the Vending Machine:

In constraint file, each entity port is mapped to the Zybo Z7 board as shown below:

```
##Clock signal
set_property -dict {PACKAGE_PIN K17      IOSTANDARD LVCMOS33} [get_ports { clk }];
##create_clock -period 8.000 -name sys_clk_pin -waveform {0.000 4.000} -add [get_ports { }];
##Switches
set_property -dict {PACKAGE_PIN G15      IOSTANDARD LVCMOS33} [ get_ports { item_select }];
#Sch=sw[0]
set_property -dict {PACKAGE_PIN P15      IOSTANDARD LVCMOS33} [ get_ports { coins_in[0] }];
#Sch=sw[1]
set_property -dict {PACKAGE_PIN W13      IOSTANDARD LVCMOS33 } [get_ports { coins_in[1] }];
#IO_L4N_T0_34 Sch=sw[2]
##Buttons
set_property -dict { PACKAGE_PIN K18     IOSTANDARD LVCMOS33 } [get_ports { reset }];
#IO_L12N_T1_MRCC_35 Sch=btn[0]

##LEDs
set_property -dict {PACKAGE_PIN M14      IOSTANDARD LVCMOS33} [get_ports { chips }];
set_property -dict {PACKAGE_PIN M15      IOSTANDARD LVCMOS33} [get_ports { chocolate }];
set_property -dict {PACKAGE_PIN G14      IOSTANDARD LVCMOS33} [get_ports { change_out[0] }];
set_property -dict {PACKAGE_PIN D18      IOSTANDARD LVCMOS33} [get_ports { change_out[1] }];
##Pmod Header JC
set_property -dict { PACKAGE_PIN V15     IOSTANDARD LVCMOS33} [get_ports { display_sum[0] }];
#IO_L10P_T1_34 Sch=jc_p[1]
```

```
set_property -dict { PACKAGE_PIN W15     IOSTANDARD LVCMOS33} [get_ports { display_sum[1] }];
#IO_L10N_T1_34 Sch=jc_n[1]
set_property -dict { PACKAGE_PIN T11     IOSTANDARD LVCMOS33} [get_ports { display_sum[2] }];
#IO_L1P_T0_34 Sch=jc_p[2]
set_property -dict { PACKAGE_PIN T10     IOSTANDARD LVCMOS33} [get_ports { display_sum[3] }];
#IO_L1N_T0_34 Sch=jc_n[2]
##Pmod Header JD
set_property -dict { PACKAGE_PIN T14     IOSTANDARD LVCMOS33} [get_ports { display_sum[4] }];
#IO_L5P_T0_34 Sch=jd_p[1]
set_property -dict { PACKAGE_PIN T15     IOSTANDARD LVCMOS33} [get_ports { display_sum[5] }];
#IO_L5N_T0_34 Sch=jd_n[1]
set_property -dict { PACKAGE_PIN P14     IOSTANDARD LVCMOS33} [get_ports { display_sum[6] }];
#IO_L6P_T0_34 Sch=jd_p[2]
set_property -dict { PACKAGE_PIN R14     IOSTANDARD LVCMOS33} [get_ports { select_segment }];
#IO_L6N_T0_VREF_34 Sch=jd_n[2]
```