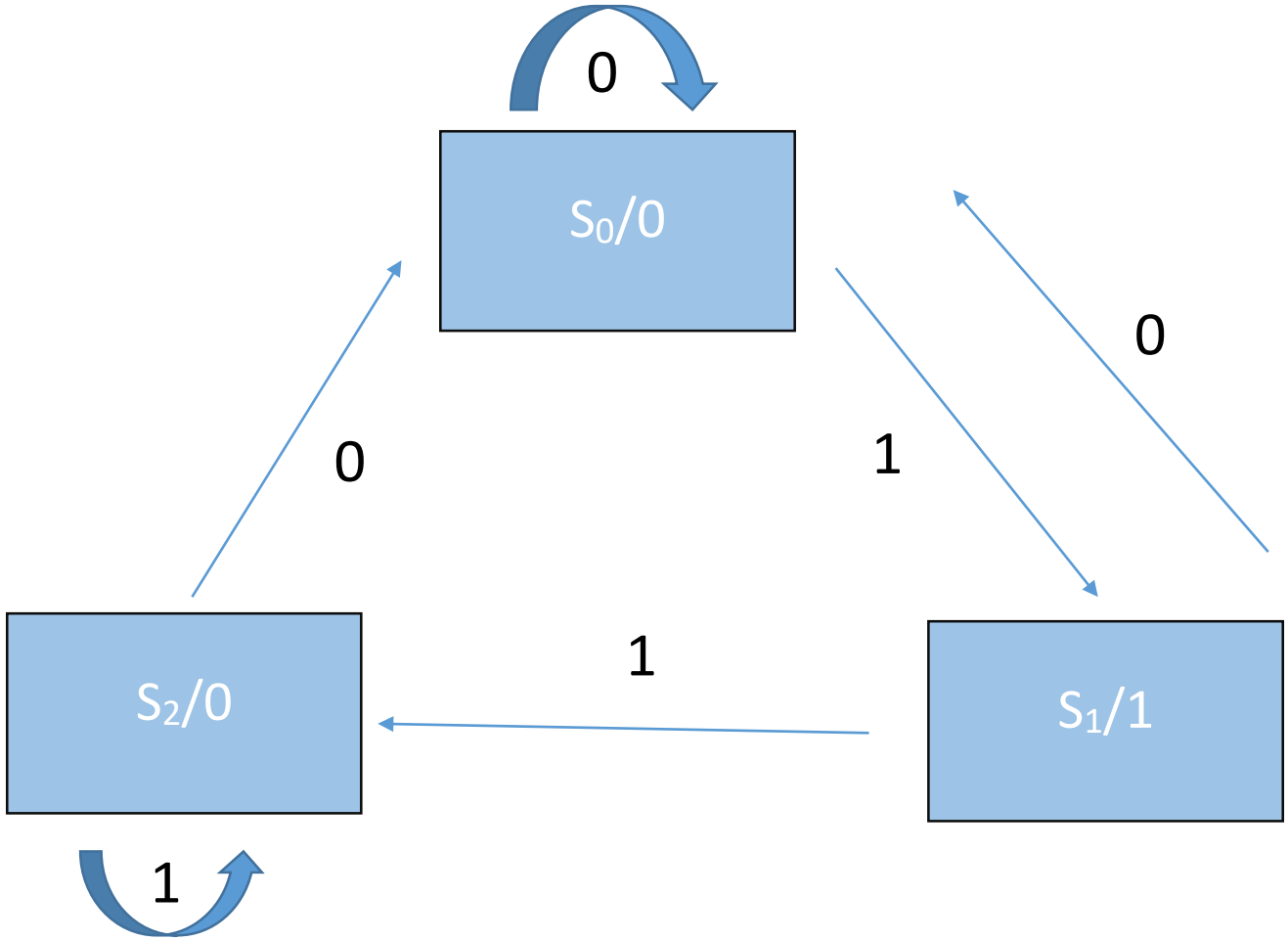
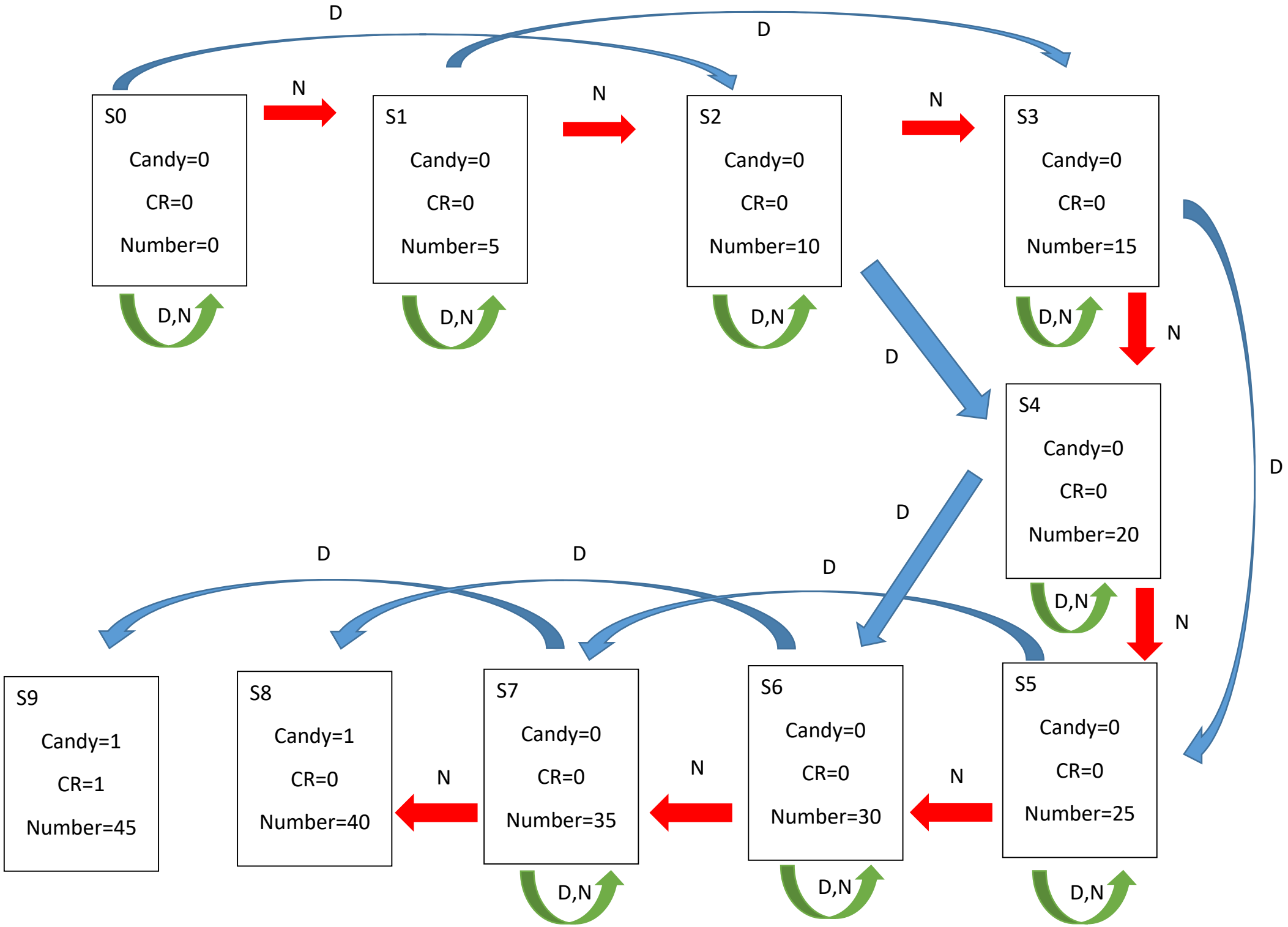

0
$S_0/0$
0
1
0
$S_2/0$
1
$S_1/1$
1


S0
Candy=0
CR=0
Number=0
S1
Candy=0
CR=0
Number=5
S2
Candy=0
CR=0
Number=10
S3
Candy=0
CR=0
Number=15
S4
Candy=0
CR=0
Number=20
S5
Candy=0
CR=0
Number=25
S6
Candy=0
CR=0
Number=30
S7
Candy=0
CR=0
Number=35
S8
Candy=1
CR=0
Number=40
S9
Candy=1
CR=1
Number=45
D
N
D,N

## TOP MODEL VHDL CODES

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


entity top_model is
Port (
top_model_clk_i: in STD_LOGIC;
top_model_reset_i: in STD_LOGIC;
top_model_nickle_i: in STD_LOGIC;
top_model_dime_i: in STD_LOGIC;

top_model_candy_o: out STD_LOGIC:='0';
top_model_cr_o: out STD_LOGIC:='0';
top_model_anode_terminals_o: out STD_LOGIC_VECTOR (3 downto 0);
top_model_cathode_terminals_o: out STD_LOGIC_VECTOR (6 downto 0)
```

```
--top_model_clk_out_o: out STD_LOGIC;
--top_model_number_o: out integer range 0 to 99; --
--top_model_nickle_o: out STD_LOGIC;
--top_model_dime_o: out STD_LOGIC

);
end top_model;

architecture Behavioral of top_model is

COMPONENT clk_divider
Port (
      clk_in : in  STD_LOGIC;
      reset  : in  STD_LOGIC;
      clk_out: out STD_LOGIC
   );
END COMPONENT;
```

```
COMPONENT btn_syc
Port (
       btn_clk_i : in  STD_LOGIC;
       btn_reset_i  : in  STD_LOGIC;
       btn_input_i : in STD_LOGIC;
       btn_output_o: out STD_LOGIC:= '0'
   );
END COMPONENT;



COMPONENT vending_machine
 Port (
    machine_clk_i: in STD_LOGIC;
    machine_reset_i: in STD_LOGIC;
    machine_nickle_i: in STD_LOGIC;
    machine_dime_i: in STD_LOGIC;

    machine_candy_o: out STD_LOGIC:='0';
    machine_cr_o: out STD_LOGIC:='0';
```

```
    machine_number_o: inout integer range 0 to 99
  );
END COMPONENT;



COMPONENT display_two_digit
Port (
display_clk_i: in STD_LOGIC;
display_decimal_number_i: in integer range 0 to 99;

display_anode_terminals_o: out STD_LOGIC_VECTOR (3 downto 0);
display_cathode_terminals_o: out STD_LOGIC_VECTOR (6 downto 0)

);
END COMPONENT;

constant top_model_clk_reset_c: STD_LOGIC:='0';
signal top_model_clk_out_s:  STD_LOGIC;
signal top_model_nickle_s:  STD_LOGIC;
```

```
signal top_model_dime_s:  STD_LOGIC;
signal top_model_number_s:  integer range 0 to 99; --




begin


ClockDivider: clk_divider
        port map (
            clk_in  => top_model_clk_i,
            reset  => top_model_clk_reset_c,

            clk_out  => top_model_clk_out_s
            );
```

```
    --top_model_clk_out_o <= top_model_clk_out_s;

ButtonNickle: btn_syc
        port map (
            btn_clk_i  => top_model_clk_out_s,
            btn_reset_i  => top_model_reset_i,
            btn_input_i  => top_model_nickle_i,

            btn_output_o => top_model_nickle_s
            );

ButtonDime: btn_syc
        port map (
            btn_clk_i  => top_model_clk_out_s,
            btn_reset_i  => top_model_reset_i,
            btn_input_i  => top_model_dime_i,

            btn_output_o => top_model_dime_s
            );
```

```
VendingMachine: vending_machine
        port map (
            machine_clk_i  => top_model_clk_out_s,
            machine_reset_i  => top_model_reset_i,
            machine_nickle_i  => top_model_nickle_s,
            machine_dime_i => top_model_dime_s,

            machine_candy_o  => top_model_candy_o,
            machine_cr_o  => top_model_cr_o,
            machine_number_o => top_model_number_s
            );

            --top_model_number_o <= top_model_number_s;
            --top_model_nickle_o <= top_model_nickle_s;
            --top_model_dime_o <= top_model_dime_s;


SevntSegment: display_two_digit
```

```
        port map (
            display_clk_i  => top_model_clk_i,
            display_decimal_number_i  => top_model_number_s,

            display_anode_terminals_o  => top_model_anode_terminals_o,
            display_cathode_terminals_o  => top_model_cathode_terminals_o
            );

end Behavioral;
```

## TOP MODEL TESTBANCH VHDL CODES

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

entity top_model_tb is
end top_model_tb;

architecture Behavioral of top_model_tb is

COMPONENT top_model
```

```
Port (
top_model_clk_i: in STD_LOGIC;
top_model_reset_i: in STD_LOGIC;
top_model_nickle_i: in STD_LOGIC;
top_model_dime_i: in STD_LOGIC;

top_model_candy_o: out STD_LOGIC:='0';
top_model_cr_o: out STD_LOGIC:='0';
top_model_anode_terminals_o: out STD_LOGIC_VECTOR (3 downto 0);
top_model_cathode_terminals_o: out STD_LOGIC_VECTOR (6 downto 0)

--top_model_clk_out_o: out STD_LOGIC;
--top_model_number_o: out integer range 0 to 99;
--top_model_nickle_o: out STD_LOGIC;
--top_model_dime_o: out STD_LOGIC
);
END COMPONENT;
```

```
-- Inputs
        signal top_model_clk_i_tb  : std_logic := '0';
        signal top_model_reset_i_tb  : std_logic := '0';
        signal top_model_nickle_i_tb  : std_logic := '0';
        signal top_model_dime_i_tb  : std_logic := '0';

-- Outputs
        signal top_model_candy_o_tb  : std_logic := '0';
        signal top_model_cr_o_tb  : std_logic := '0';
        signal top_model_anode_terminals_o_tb : STD_LOGIC_VECTOR (3 downto 0):= "1110";
        signal top_model_cathode_terminals_o_tb : STD_LOGIC_VECTOR (6 downto 0):= "1111111";

--          signal top_model_clk_out_o_tb:  STD_LOGIC;
--    signal top_model_number_o_tb:  integer range 0 to 99; --
--    signal top_model_nickle_o_tb: STD_LOGIC;
--    signal top_model_dime_o_tb: STD_LOGIC;

--Clock Time
        constant clk_time : time := 10 ns;
```

```
begin



-- Instance of unit under test.
Uut: top_model
            port map (
                  top_model_clk_i  => top_model_clk_i_tb,
                  top_model_reset_i  => top_model_reset_i_tb,
      top_model_nickle_i  => top_model_nickle_i_tb,
                  top_model_dime_i  => top_model_dime_i_tb,


                  top_model_candy_o  => top_model_candy_o_tb,
                  top_model_cr_o  => top_model_cr_o_tb,
                  top_model_anode_terminals_o  => top_model_anode_terminals_o_tb,
                  top_model_cathode_terminals_o  => top_model_cathode_terminals_o_tb

          --        top_model_clk_out_o  => top_model_clk_out_o_tb,
          --        top_model_number_o  => top_model_number_o_tb, --
```

```
        --      top_model_nickle_o => top_model_nickle_o_tb,
        --      top_model_dime_o => top_model_dime_o_tb
                );

-- Clock definition.
entrada_process :process
                begin
                top_model_clk_i_tb <= '0';
                wait for clk_time / 2;
                top_model_clk_i_tb <= '1';
                wait for clk_time / 2;
        end process;


-- Processing.
stimuli: process
        begin

--2 dime 4 nickle 2dime 1 reset
```

```
top_model_nickle_i_tb <= '0';
        top_model_dime_i_tb <= '0';
        wait for 10 ms;
        top_model_nickle_i_tb <= '0';
        top_model_dime_i_tb <= '1';
        wait for 20 ms;
        top_model_nickle_i_tb <= '0';
        top_model_dime_i_tb <= '0';
        wait for 10 ms;
        top_model_nickle_i_tb <= '0';
        top_model_dime_i_tb <= '1';
        wait for 20 ms;
        top_model_nickle_i_tb <= '0';
        top_model_dime_i_tb <= '0';
        wait for 10 ms;
        top_model_nickle_i_tb <= '1';
        top_model_dime_i_tb <= '0';
        wait for 20 ms;
```

```
top_model_nickle_i_tb <= '0';
top_model_dime_i_tb <= '0';
wait for 10 ms;
top_model_nickle_i_tb <= '1';
top_model_dime_i_tb <= '0';
wait for 20 ms;
top_model_nickle_i_tb <= '0';
top_model_dime_i_tb <= '0';
wait for 10 ms;
top_model_nickle_i_tb <= '1';
top_model_dime_i_tb <= '0';
wait for 20 ms;
top_model_nickle_i_tb <= '0';
top_model_dime_i_tb <= '0';
wait for 10 ms;
top_model_nickle_i_tb <= '1';
top_model_dime_i_tb <= '0';
wait for 20 ms;
top_model_nickle_i_tb <= '0';
```

```
top_model_dime_i_tb <= '0';
wait for 20 ms;
top_model_nickle_i_tb <= '0';
top_model_dime_i_tb <= '1';
wait for 20 ms;
top_model_nickle_i_tb <= '0';
top_model_dime_i_tb <= '0';
wait for 10 ms;
top_model_nickle_i_tb <= '0';
top_model_dime_i_tb <= '1';
wait for 20 ms;
top_model_reset_i_tb <= '1';
wait for 10 ms;
top_model_reset_i_tb <= '0';
wait for 10 ms;
```

--3nickle 2dime 1dimeve1nickle aynı anda 1 dime 1reset

```
top_model_nickle_i_tb <= '0';
        top_model_dime_i_tb <= '0';
        wait for 10 ms;
        top_model_nickle_i_tb <= '1';
        top_model_dime_i_tb <= '0';
        wait for 20 ms;
        top_model_nickle_i_tb <= '0';
        top_model_dime_i_tb <= '0';
        wait for 10 ms;
        top_model_nickle_i_tb <= '1';
        top_model_dime_i_tb <= '0';
        wait for 20 ms;
        top_model_nickle_i_tb <= '0';
        top_model_dime_i_tb <= '0';
        wait for 10 ms;
        top_model_nickle_i_tb <= '1';
        top_model_dime_i_tb <= '0';
        wait for 20 ms;
        top_model_nickle_i_tb <= '0';
```

```
top_model_dime_i_tb <= '0';
wait for 10 ms;
top_model_nickle_i_tb <= '0';
top_model_dime_i_tb <= '1';
wait for 20 ms;
top_model_nickle_i_tb <= '0';
top_model_dime_i_tb <= '0';
wait for 10 ms;
top_model_nickle_i_tb <= '0';
top_model_dime_i_tb <= '1';
wait for 20 ms;
top_model_nickle_i_tb <= '0';
top_model_dime_i_tb <= '0';
wait for 10 ms;
top_model_nickle_i_tb <= '1';
top_model_dime_i_tb <= '1';
wait for 20 ms;
top_model_nickle_i_tb <= '0';
top_model_dime_i_tb <= '0';
```

```
                wait for 10 ms;
                top_model_nickle_i_tb <= '0';
                top_model_dime_i_tb <= '1';
                wait for 20 ms;
                top_model_nickle_i_tb <= '0';
                top_model_dime_i_tb <= '0';
                wait for 20 ms;
-- 0.5s



        end process;
end Behavioral;
```

## TOP MODEL CONSTRAINTS (XDC) CODES

```
## Clock signal
set_property PACKAGE_PIN W5 [get_ports top_model_clk_i]
        set_property IOSTANDARD LVCMOS33 [get_ports top_model_clk_i]
        #create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]


## LEDs
```

```
set_property PACKAGE_PIN U16 [get_ports top_model_candy_o]
        set_property IOSTANDARD LVCMOS33 [get_ports top_model_candy_o]
set_property PACKAGE_PIN E19 [get_ports top_model_cr_o]
        set_property IOSTANDARD LVCMOS33 [get_ports top_model_cr_o]
#set_property PACKAGE_PIN U19 [get_ports top_model_clk_out_o]
        #set_property IOSTANDARD LVCMOS33 [get_ports top_model_clk_out_o]
#set_property PACKAGE_PIN V19 [get_ports top_model_nickle_o]
        #set_property IOSTANDARD LVCMOS33 [get_ports top_model_nickle_o]
#set_property PACKAGE_PIN W18 [get_ports top_model_dime_o]
        #set_property IOSTANDARD LVCMOS33 [get_ports top_model_dime_o]
#set_property PACKAGE_PIN U15 [get_ports {top_model_number_o}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {top_model_number_o}]



##7 segment display
set_property PACKAGE_PIN W7 [get_ports {top_model_cathode_terminals_o[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {top_model_cathode_terminals_o[0]}]
set_property PACKAGE_PIN W6 [get_ports {top_model_cathode_terminals_o[1]}]
```

```
        set_property IOSTANDARD LVCMOS33 [get_ports {top_model_cathode_terminals_o[1]}]
set_property PACKAGE_PIN U8 [get_ports {top_model_cathode_terminals_o[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {top_model_cathode_terminals_o[2]}]
set_property PACKAGE_PIN V8 [get_ports {top_model_cathode_terminals_o[3]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {top_model_cathode_terminals_o[3]}]
set_property PACKAGE_PIN U5 [get_ports {top_model_cathode_terminals_o[4]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {top_model_cathode_terminals_o[4]}]
set_property PACKAGE_PIN V5 [get_ports {top_model_cathode_terminals_o[5]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {top_model_cathode_terminals_o[5]}]
set_property PACKAGE_PIN U7 [get_ports {top_model_cathode_terminals_o[6]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {top_model_cathode_terminals_o[6]}]

#set_property PACKAGE_PIN V7 [get_ports dp]
        #set_property IOSTANDARD LVCMOS33 [get_ports dp]

set_property PACKAGE_PIN U2 [get_ports {top_model_anode_terminals_o[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {top_model_anode_terminals_o[0]}]
set_property PACKAGE_PIN U4 [get_ports {top_model_anode_terminals_o[1]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {top_model_anode_terminals_o[1]}]
```

```
set_property PACKAGE_PIN V4 [get_ports {top_model_anode_terminals_o[2]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {top_model_anode_terminals_o[2]}]
set_property PACKAGE_PIN W4 [get_ports {top_model_anode_terminals_o[3]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {top_model_anode_terminals_o[3]}]


##Buttons
set_property PACKAGE_PIN U18 [get_ports top_model_reset_i]
        set_property IOSTANDARD LVCMOS33 [get_ports top_model_reset_i]
set_property PACKAGE_PIN T18 [get_ports top_model_nickle_i]
        set_property IOSTANDARD LVCMOS33 [get_ports top_model_nickle_i]
set_property PACKAGE_PIN W19 [get_ports top_model_dime_i]
        set_property IOSTANDARD LVCMOS33 [get_ports top_model_dime_i]


## Configuration options, can be used for all designs
set_property CONFIG_VOLTAGE 3.3 [current_design]
set_property CFGBVS VCCO [current_design]
```

## **COMPONENT- Clock Divider vhdl codes**

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

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

architecture Behavioral of clk_divider is
   signal temporal: STD_LOGIC:='0';
   signal counter : integer range 0 to 499999 := 0;
begin

frequency_divider: process (reset, clk_in) begin
      if (reset = '1') then
         temporal <= '0';
```

```
            counter <= 0;
        elsif falling_edge(clk_in) then
            if (counter = 499999) then
                temporal <= NOT(temporal);
                counter <= 0;
            else
                counter <= counter + 1;
            end if;
        end if;
    end process;

clk_out <= temporal;
end Behavioral;
```

**COMPONENT- Button Synchronous vhdl codes**

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

entity btn_syc is
Port (
```

```
        btn_clk_i : in  STD_LOGIC;
        btn_reset_i  : in  STD_LOGIC;
        btn_input_i : in STD_LOGIC;
        btn_output_o: out STD_LOGIC:= '0'
    );

end btn_syc;

architecture Behavioral of btn_syc is

type t_state is (BTN_INITIAL, BTN_SECOND, BTN_THIRD);
signal state : t_state := BTN_INITIAL;

begin

process (btn_reset_i,btn_clk_i) begin

if (btn_reset_i = '1') then
    btn_output_o <= '0';
```

```
elsif falling_edge(btn_clk_i) then
        case state is

            when BTN_INITIAL =>

                btn_output_o <= '0';

                if (btn_input_i = '0') then
                    state    <= BTN_INITIAL;
                    btn_output_o <= '0';
                else
                    state    <= BTN_SECOND;
                    btn_output_o <= '1';
                end if;

            when BTN_SECOND =>
```

```
            if (btn_input_i = '0') then
                    state    <= BTN_INITIAL;
                    btn_output_o <= '0';
            else
                    state    <= BTN_THIRD;
                    btn_output_o <= '0';
            end if;

when BTN_THIRD =>


            if (btn_input_i = '0') then
                    state    <= BTN_INITIAL;
                    btn_output_o <= '0';
            else
                    state    <= BTN_THIRD;
                    btn_output_o <= '0';
            end if;
```

```
            end case;

end if;
end process;




end Behavioral;
```

## COMPONENT- Vending Machine vhdl codes

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;
use ieee.numeric_std.all;

entity vending_machine is
 Port (
    machine_clk_i: in STD_LOGIC;
    machine_reset_i: in STD_LOGIC;
```

```
    machine_nickle_i: in STD_LOGIC;
    machine_dime_i: in STD_LOGIC;

    machine_candy_o: out STD_LOGIC:='0';
    machine_cr_o: out STD_LOGIC:='0';
    machine_number_o: inout integer range 0 to 99
  );
end vending_machine;

architecture Behavioral of vending_machine is

 type M_state is (M_INITIAL, M_SECOND, M_THIRD,M_FOURTH,
     M_FIFTH, M_SIXTH, M_SEVENTH, M_EIGHTH, M_NINTH, M_TENTH);
 signal state : M_state := M_INITIAL;

 signal ex_machine_nickle_i: STD_LOGIC:='0';
 signal ex_machine_dime_i: STD_LOGIC:='0';
```

```
begin

ex_machine_nickle_i <= machine_nickle_i;
ex_machine_dime_i <= machine_dime_i;



process (machine_reset_i, machine_clk_i) begin

if (machine_reset_i = '1') then
   ex_machine_nickle_i <= '0';
   ex_machine_dime_i <= '0';
   machine_cr_o <= '1';
   state<= M_INITIAL;

elsif rising_edge(machine_clk_i) then
          case state is
```

```
            when M_INITIAL =>


                    machine_cr_o <= '0';
                    machine_number_o <= 0;
        machine_candy_o <= '0';


if (machine_nickle_i='1' and machine_dime_i='1') then
 state    <= M_INITIAL;
                    elsif (machine_nickle_i = '1') then
                            state    <= M_SECOND;
                            machine_number_o <= 5;
                    elsif (machine_dime_i ='1') then
                            state    <= M_THIRD;
                            machine_number_o <= 10;
                    else
                        state          <= M_INITIAL;
                    end if;
```

```
            when M_SECOND =>

                    machine_number_o <= 5;

if (machine_nickle_i='1' and machine_dime_i='1') then
state    <= M_SECOND;
                    elsif (machine_nickle_i = '1') then
                            state    <= M_THIRD;
                            machine_number_o <= 10;
                    elsif (machine_dime_i ='1') then
                            state    <= M_FOURTH;
                            machine_number_o <= 15;
                    else
                        state          <= M_SECOND;
                    end if;

            when M_THIRD =>

                    machine_number_o <= 10;
```

```
    if (machine_nickle_i='1' and machine_dime_i='1') then
    state     <= M_THIRD;
                        elsif (machine_nickle_i = '1') then
                                state    <= M_FOURTH;
                                machine_number_o <= 15;
                        elsif (machine_dime_i ='1') then
                                state    <= M_FIFTH;
                                machine_number_o <= 20;
                        else
                            state          <= M_THIRD;
                        end if;



                when M_FOURTH =>
           machine_number_o <= 15;

if (machine_nickle_i='1' and machine_dime_i='1') then
state         <= M_FOURTH;
```

```
        elsif (machine_nickle_i = '1') then
                state <= M_FIFTH;
                machine_number_o <= 20;
        elsif (machine_dime_i = '1') then
                state <= M_SIXTH;
                machine_number_o <= 25;
        else
                state <= M_FOURTH;
        end if;



 when M_FIFTH =>
         machine_number_o <= 20;

if (machine_nickle_i='1' and machine_dime_i='1') then
state         <= M_FIFTH;
        elsif (machine_nickle_i = '1') then
                state <= M_SIXTH;
                machine_number_o <= 25;
```

```
        elsif (machine_dime_i = '1') then
                state <= M_SEVENTH;
                machine_number_o <= 30;
        else
                state <= M_FIFTH;
        end if;


when M_SIXTH =>
        machine_number_o <= 25;

if (machine_nickle_i='1' and machine_dime_i='1') then
state         <= M_SIXTH;
        elsif (machine_nickle_i = '1') then
                state <= M_SEVENTH;
                machine_number_o <= 30;
        elsif (machine_dime_i = '1') then
                state <= M_EIGHTH;
                machine_number_o <= 35;
```

```
        else
                state <= M_SIXTH;
        end if;



 when M_SEVENTH =>
        machine_number_o <= 30;

        if (machine_nickle_i='1' and machine_dime_i='1') then
state           <= M_SEVENTH;
        elsif (machine_nickle_i = '1') then
                state <= M_EIGHTH;
                machine_number_o <= 35;
        elsif (machine_dime_i = '1') then
                state <= M_NINTH;
                machine_number_o <= 40;
        else
                state <= M_SEVENTH;
        end if;
```

```
 when M_EIGHTH =>
        machine_number_o <= 35;



        if (machine_nickle_i='1' and machine_dime_i='1') then
state         <= M_EIGHTH;
        elsif (machine_nickle_i = '1') then
                state <= M_NINTH;
                machine_number_o <= 40;

                elsif (machine_dime_i = '1') then
                state <= M_TENTH;
                machine_number_o <= 45;
        else
                state <= M_EIGHTH;
        end if;
```

```
        when M_NINTH =>
                machine_number_o <= 40;
                machine_candy_o <= '1';
                state <= M_INITIAL;

                when M_TENTH =>
                machine_number_o <= 45;
                machine_candy_o <= '1';
                machine_cr_o <= '1';
                state <= M_INITIAL;


            end case;

end if;
end process;
```

```
end Behavioral;
```

## COMPONENT- Two Digit Display vhdl codes

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



entity display_two_digit is
generic (
c_clkfreq        : integer := 100_000_000
);
Port (
display_clk_i: in STD_LOGIC;
display_decimal_number_i: in integer range 0 to 99;

display_anode_terminals_o: out STD_LOGIC_VECTOR (3 downto 0);
display_cathode_terminals_o: out STD_LOGIC_VECTOR (6 downto 0)

);
end display_two_digit;
```

```
architecture Behavioral of display_two_digit is

signal display_decimal_left_digit : integer range 0 to 9;
signal display_decimal_right_digit : integer range 0 to 9;

signal display_cathode_terminals_left: STD_LOGIC_VECTOR (6 downto 0);
signal display_cathode_terminals_right: STD_LOGIC_VECTOR (6 downto 0);

--constant c_timer1mslim: integer := c_clkfreq/1000;
--signal timer1ms: integer range 0 to c_timer1mslim:= 0;
signal anodes : std_logic_vector (3 downto 0) := "1110";


begin

display_decimal_left_digit <= display_decimal_number_i/10;
display_decimal_right_digit <= display_decimal_number_i mod 10;
```

```
DecimalToSeventSegmentLEFT: process (display_decimal_left_digit) begin

        case display_decimal_left_digit is

                when 0 =>
                        display_cathode_terminals_left      <= "0000001"; -- CACBCCCDCECFCG

                when 1 =>
                        display_cathode_terminals_left      <= "1001111"; -- CACBCCCDCECFCG

                when 2 =>
                        display_cathode_terminals_left      <= "0010010"; -- CACBCCCDCECFCG

                when 3 =>
                        display_cathode_terminals_left      <= "0000110"; -- CACBCCCDCECFCG

                when 4 =>
                        display_cathode_terminals_left      <= "1001100"; -- CACBCCCDCECFCG
```

```
        when 5 =>
                display_cathode_terminals_left      <= "0100100"; -- CACBCCCDCECFCG

        when 6 =>
                display_cathode_terminals_left      <= "0100000"; -- CACBCCCDCECFCG

        when 7 =>
                display_cathode_terminals_left      <= "0001111"; -- CACBCCCDCECFCG

        when 8 =>
                display_cathode_terminals_left      <= "0000000"; -- CACBCCCDCECFCG

        when 9 =>
                display_cathode_terminals_left      <= "0000100"; -- CACBCCCDCECFCG

        when others =>
                display_cathode_terminals_left      <= "1111111"; -- CACBCCCDCECFCG

end case;
```

```
end process;

DecimalToSeventSegmentRIGHT: process (display_decimal_right_digit) begin

        case display_decimal_right_digit is

                when 0 =>
                        display_cathode_terminals_right     <= "0000001"; -- CACBCCCDCECFCG

                when 1 =>
                        display_cathode_terminals_right     <= "1001111"; -- CACBCCCDCECFCG

                when 2 =>
                        display_cathode_terminals_right     <= "0010010"; -- CACBCCCDCECFCG

                when 3 =>
                        display_cathode_terminals_right     <= "0000110"; -- CACBCCCDCECFCG
```

```
when 4 =>
        display_cathode_terminals_right     <= "1001100"; -- CACBCCCDCECFCG

when 5 =>
        display_cathode_terminals_right     <= "0100100"; -- CACBCCCDCECFCG

when 6 =>
        display_cathode_terminals_right     <= "0100000"; -- CACBCCCDCECFCG

when 7 =>
        display_cathode_terminals_right     <= "0001111"; -- CACBCCCDCECFCG

when 8 =>
        display_cathode_terminals_right     <= "0000000"; -- CACBCCCDCECFCG

when 9 =>
        display_cathode_terminals_right     <= "0000100"; -- CACBCCCDCECFCG

when others =>
```

```
                        display_cathode_terminals_right     <= "1111111"; -- CACBCCCDCECFCG

        end case;

end process;



P_ANODES : process (display_clk_i) begin
if (falling_edge(display_clk_i)) then

        anodes(3 downto 2)  <= "11";
        anodes(1)           <= anodes(0);
        anodes(0)           <= anodes(1);

--      if (timer1ms = c_timer1mslim-1) then
--        timer1ms                  <= 0;
--              anodes(1)           <= anodes(0);
--              anodes(0)           <= anodes(1);
--      else
```

```
--                  timer1ms                        <= timer1ms + 1;
--          end if;

end if;
end process;



P_CATHODES  : process (display_clk_i) begin
if (falling_edge(display_clk_i)) then

          if (anodes(0) = '0') then
                    display_cathode_terminals_o          <= display_cathode_terminals_left;
          elsif (anodes(1) = '0') then
                    display_cathode_terminals_o          <= display_cathode_terminals_right;
          else
                    display_cathode_terminals_o          <= (others => '1');
          end if;

end if;
```

```
end process;

display_anode_terminals_o  <= anodes;

end Behavioral;
```

## Testbench Button

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

entity btn_syc_tb is
end btn_syc_tb;

architecture Behavioral of btn_syc_tb is

COMPONENT btn_syc
        Port (
```

```
    btn_clk_i : in  STD_LOGIC;
    btn_reset_i  : in  STD_LOGIC;
    btn_input_i : in STD_LOGIC;
    btn_output_o: out STD_LOGIC:= '0'
  );
        END COMPONENT;

  -- Inputs
        signal btn_clk_i_tb  : std_logic := '0';
        signal btn_reset_i_tb   : std_logic := '0';
        signal btn_input_i_tb   : std_logic := '0';
        -- Outputs
        signal btn_output_o_tb : std_logic := '0';
        --Clock Time
        constant clk_time : time := 10 ns;
        -- State monitor
  signal state_tb       : std_logic_vector(1 downto 0) := "00";

begin
```

```
-- Instance of unit under test.
        uut: btn_syc
        port map (
                btn_clk_i  => btn_clk_i_tb,
                btn_reset_i  => btn_reset_i_tb,
                btn_input_i => btn_input_i_tb,
                btn_output_o => btn_output_o_tb

        );

-- Clock definition.
        entrada_process :process
                begin
                btn_clk_i_tb <= '1';
                wait for clk_time / 2;
                btn_clk_i_tb <= '0';
                wait for clk_time / 2;
        end process;

-- Reset.
```

```
        resetting: process
        begin
                btn_reset_i_tb <= '1'; -- Initial conditions.
                wait for 50 ns;
                btn_reset_i_tb <= '0'; -- Down to work!
    wait;
        end process;

-- Process
        simuti: process
        begin
                btn_input_i_tb <= '0' ;
    wait for 10 ns;

    btn_input_i_tb <= '1' ;
    wait for 30ns;

    btn_input_i_tb <= '0' ;
    wait for 10 ns;
```

```
        btn_input_i_tb <= '1' ;
        wait for 40 ns;

        btn_input_i_tb <= '0' ;
        wait for 30 ns;

        btn_input_i_tb <= '1' ;
        wait for 10 ns;

        btn_input_i_tb <= '0' ;
        wait for 10ns;
            end process;



end Behavioral;
```

## Testbench Vending Machine

```
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
        Port (
   machine_clk_i: in STD_LOGIC;
   machine_reset_i: in STD_LOGIC;
   machine_nickle_i: in STD_LOGIC;
   machine_dime_i: in STD_LOGIC;

   machine_candy_o: out STD_LOGIC:='0';
   machine_cr_o: out STD_LOGIC:='0';
   machine_number_o: inout integer range 0 to 255
   );
        END COMPONENT;

        -- Inputs
```

```
        signal machine_clk_i_tb  : std_logic := '0';
        signal machine_reset_i_tb   : std_logic := '0';
        signal machine_nickle_i_tb   : std_logic := '0';
        signal machine_dime_i_tb   : std_logic := '0';

        -- Outputs
        signal machine_candy_o_tb : std_logic := '0';
        signal machine_cr_o_tb : std_logic := '0';
        signal machine_number_o_tb : integer := 0;

        --Clock Time
        constant clk_time : time := 10 ns;


begin

-- Instance of unit under test.
            uut: vending_machine
            port map (
                    machine_clk_i  => machine_clk_i_tb,
```

```
                machine_reset_i  => machine_reset_i_tb,
                machine_nickle_i => machine_nickle_i_tb,
                machine_dime_i => machine_dime_i_tb,
                machine_candy_o => machine_candy_o_tb,
                machine_cr_o => machine_cr_o_tb,
                machine_number_o => machine_number_o_tb
        );

-- Clock definition.
        entrada_process :process
                begin
                machine_clk_i_tb <= '1';
                wait for clk_time / 2;
                machine_clk_i_tb <= '0';
                wait for clk_time / 2;
        end process;

-- Reset.
        resetting: process
        begin
```

```
                machine_reset_i_tb <= '1'; -- Initial conditions.
                wait for 50 ns;
                machine_reset_i_tb <= '0'; -- Down to work!
        wait;
            end process;

-- Process
            simutin: process
            begin
                machine_nickle_i_tb <= '0' ;
        machine_dime_i_tb <= '0' ;
        wait for 10 ns;

        machine_nickle_i_tb <= '1' ;
        machine_dime_i_tb <= '0' ;
        wait for 10 ns;

        machine_nickle_i_tb <= '0' ;
        machine_dime_i_tb <= '0' ;
        wait for 10 ns;
```

```
machine_nickle_i_tb <= '0' ;
machine_dime_i_tb <= '1' ;
wait for 10 ns;

machine_nickle_i_tb <= '0' ;
machine_dime_i_tb <= '0' ;
wait for 10 ns;

machine_nickle_i_tb <= '1' ;
machine_dime_i_tb <= '1' ;
wait for 10 ns;

machine_nickle_i_tb <= '0' ;
machine_dime_i_tb <= '0' ;
wait for 10 ns;

    end process;
```

```
end Behavioral;
```