Ain shams university
Faculty of engineering
Computer engineering and software systems programs

Course: EDA (CSE215)

# Project 2

Submitted by: Shehab Ahmed Hassan Kotb

Id: 16p6014

Email : shehabktob@gmail.com

Submitted to: Dr. Mohamed Dessouky
Eng. Michael Hany

# Table of Contents

## 1.0 Introduction

This document covers the low-level synthesis of a Digital access control system, it’s high level design was already covered in the previous document.

The low-level synthesis is achieved by using alliance tool such as syf, boom, boog, loon; the output of these tool is detailed bellow.

## 2.0 boom comparison

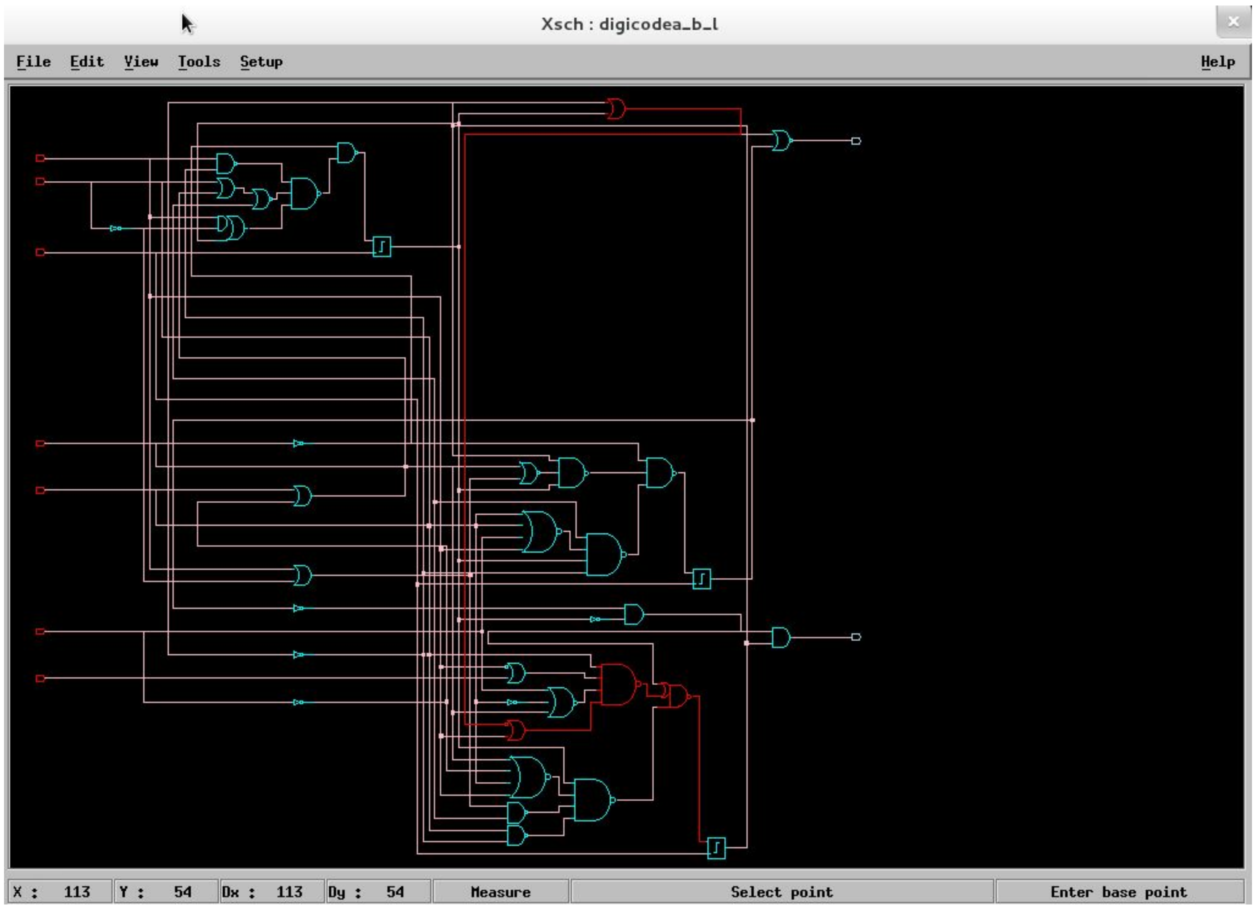
| encoding | before boom (literals) | after boom (literals) |
|---|---|---|
| a | 137 | 54 |
| j | 138 | 56 |
| m | 137 | 56 |
| o | 122 | 70 |
| r | 138 | 70 |

## 3.0 loon comparison

| encoding | before loon (delay ps) | after loon (delay ps) | before loon (area $\lambda^2$) | after loon (area $\lambda^2$) |
|---|---|---|---|---|
| a | 1808 | 1783 | 51250 | 51250 |
| j | 1858 | 1836 | 52500 | 52750 |
| m | 2229 | 2119 | 50750 | 53000 |
| o | 2919 | 2805 | 81000 | 85000 |
| r | 2651 | 2536 | 68000 | 70000 |

We will choose to continue with the a encoding as we can see it has the lowest area and the lowest delay making it the best among all the other encodings.

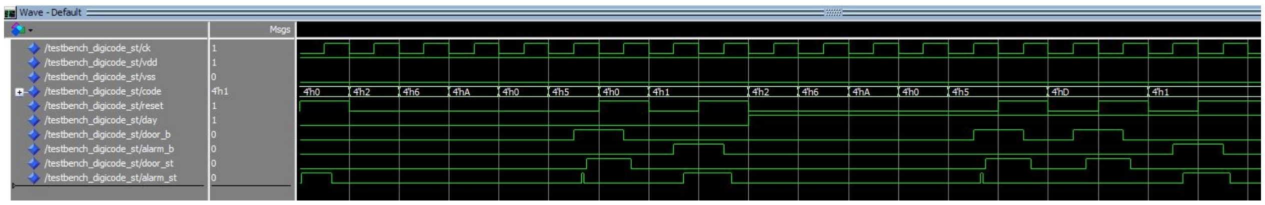
## 4.0 XSCH visualization



## 5.0 Netlist checking

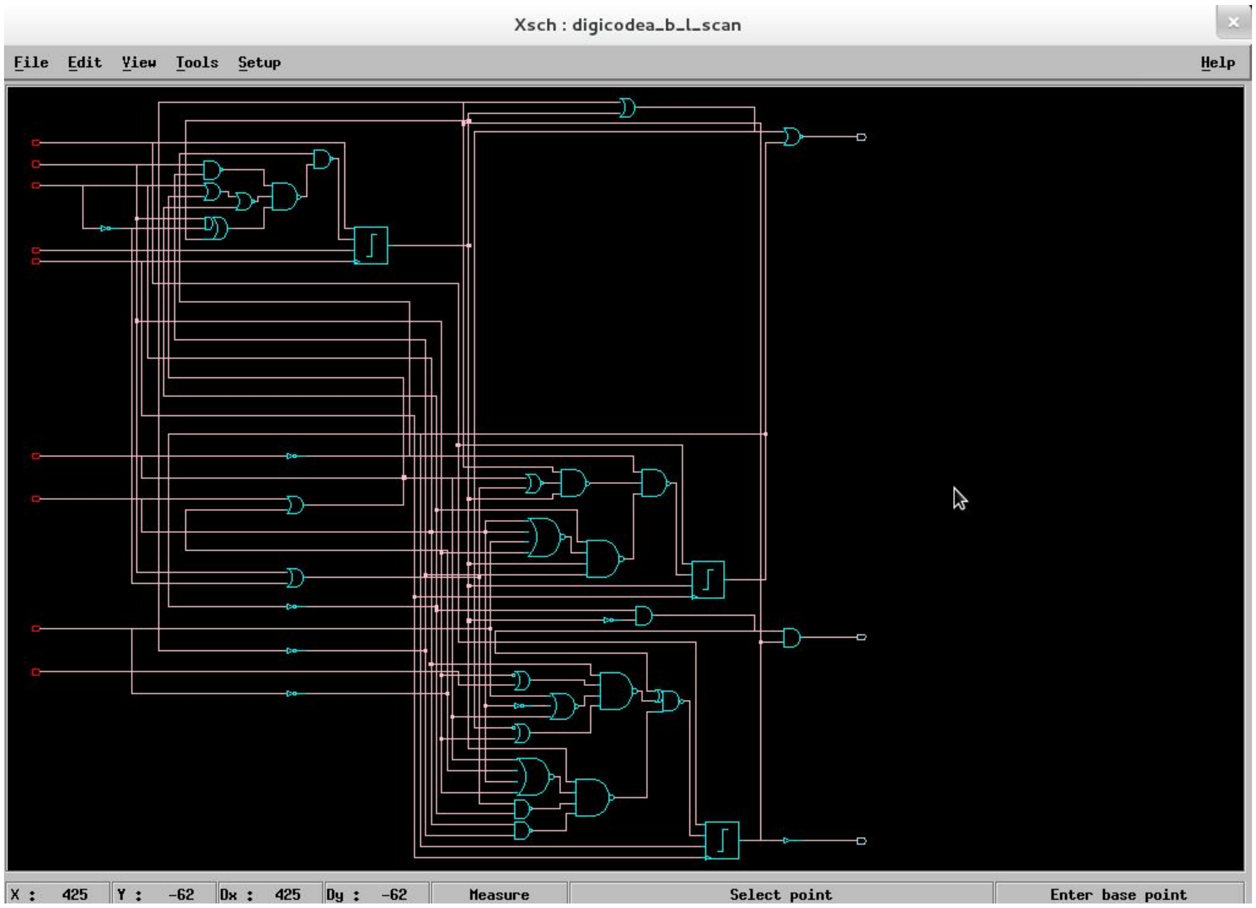
The log files of flatbeh & proof are in the appendix.

## 6.0 delay simulation (behavioral & structural)


Wave - Default
Msgs
/testbench_digicode_st/ck 1
/testbench_digicode_st/vdd 1
/testbench_digicode_st/vss 0
/testbench_digicode_st/code 4'h1
/testbench_digicode_st/reset 1
/testbench_digicode_st/day 1
/testbench_digicode_st/door_b 0
/testbench_digicode_st/alarm_b 0
/testbench_digicode_st/door_st 0
/testbench_digicode_st/alarm_st 0
4'h0 4'h2 4'h6 4'hA 4'h0 4'h5 4'h0 4'h1 4'h2 4'h6 4'hA 4'h0 4'h5 4'hD 4'h1

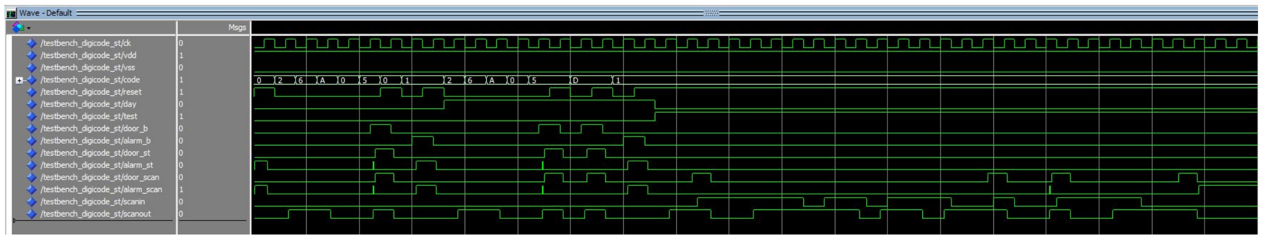
The test are the same as last project, except a reset case added at the begging to stabilize initial outputs; from the graph we can see that the structural output follow the behavioral outputs by a slight delay.

## 7.0 DFT


Xsch : digicodea_b_l_scan
File Edit View Tools Setup
Help
X : 425 Y : -62 Dx : 425 Dy : -62 Measure Select point Enter base point

The .path file is attached in appendix.

## 8.0 DFT simulation


Wave - Default
Msgs
/testbench_digicode_st/ck 0
/testbench_digicode_st/vdd 1
/testbench_digicode_st/vss 0
/testbench_digicode_st/code 1
/testbench_digicode_st/reset 1
/testbench_digicode_st/day 0
/testbench_digicode_st/test 1
/testbench_digicode_st/door_b 0
/testbench_digicode_st/alarm_b 0
/testbench_digicode_st/door_st 0
/testbench_digicode_st/alarm_st 0
/testbench_digicode_st/door_scan 0
/testbench_digicode_st/alarm_scan 1
/testbench_digicode_st/scanin 0
/testbench_digicode_st/scanout 0
0 2 6 A 0 5 0 1 2 6 A 0 5 D 1

The code is in hexadecimal, D in hexadecimal is (1101) in binary which is the encoding given for the O button; after the normal testing is done test becomes 1 and scanout follows scanin with delay of 3 clock cycles because we have three flipflops the result takes 3 cycles to propagate from scanin to scanout.

# Appendix

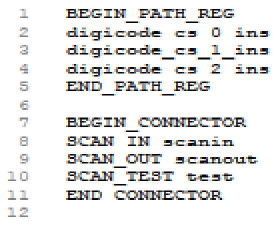
## Flatbeh logs

```

      @@@@@@@@@  @@@@                      @@@@@@@               @@@
        @@    @    @@                @       @@   @@              @@
        @@     @   @@               @@       @@    @@             @@
        @@         @@     @@@@      @@       @@    @@    @@@@@    @@ @@@
        @@   @     @@    @@   @   @@@@@@@@   @@   @@    @     @   @@@  @@
        @@@@@@     @@    @@   @@    @@       @@@@@@    @@     @@  @@    @@
        @@   @     @@      @@@@@    @@       @@    @@  @@@@@@@@@  @@    @@
        @@         @@    @@   @@    @@       @@     @@ @@         @@    @@
        @@         @@   @@    @@    @@       @@     @@ @@      @  @@    @@
        @@         @@   @@   @@@    @@   @   @@    @@   @@    @@  @@    @@
      @@@@@@     @@@@@@  @@@@  @@    @@@@  @@@@@@@@       @@@@   @@@@  @@@@

                              a netlist abstractor

           Alliance CAD System 5.0 20090901, flatbeh 5.0 [2000/11/01]
           Copyright (c) 1993-2019,                    ASIM/LIP6/UPMC
           Author(s):                 François DONNET, Huu Nghia VUONG
           E-mail        :                 alliance-users@asim.lip6.fr

     =========================  Environnement  =========================
     MBK WORK LIB        = .
     MBK_CATA_LIB        =
     .:/usr/lib64/alliance/cells/sxlib:/usr/lib64/alliance/cells/dp sxlib:/usr/lib64/alli
     ance/cells/rflib:/usr/lib64/alliance/cells/rf2lib:/usr/lib64/alliance/cells/ramlib:/
     usr/lib64/alliance/cells/romlib:/usr/lib64/alliance/cells/pxlib:/usr/lib64/alliance/
     cells/padlib
     MBK CATAL NAME      = CATAL
     ============================== Files ==============================
     Netlist file        = digicodea b 1.vst
     Output  file        = digicodea_b_1_net.vbe
     ===================================================================

Loading './digicodea b 1.vst'
flattening figure digicodea b 1
 loading oa22_x2
 loading o2 x2
 loading na3_x1
 loading na2 x1
 loading no4_x1
 loading inv x2
 loading no3_x1
 loading on12 x1
 loading na4_x1
 loading nao22 x1
 loading sff1_x4
 loading a2_x2
 loading no2 x1
Restoring array's orders
BEH : Saving 'digicodea b 1 net' in a vhdl file (vbe)

```

## Proof logs

```

                @@@@@@@                                     @@@
                  @@   @@                                  @  @@
                  @@    @@                                @@  @@
                  @@    @@ @@@ @@@     @@@       @@@      @@
                  @@   @@   @@@  @@  @@   @@   @@   @@  @@@@@@@@
                  @@@@@     @@   @@ @@     @@ @@     @@   @@
                  @@        @@      @@     @@ @@     @@   @@
                  @@        @@      @@     @@ @@     @@   @@
                  @@        @@      @@     @@ @@     @@   @@
                  @@        @@       @@   @@   @@   @@    @@
                @@@@@@     @@@@        @@@       @@@    @@@@@@

                                  Formal Proof

                   Alliance CAD System 5.0 20090901, proof 5.0
                   Copyright (c) 1990-2019,     ASIM/LIP6/UPMC
                   E-mail        : alliance-users@asim.lip6.fr

================================  Environment  ================================
MBK WORK LIB        = .
MBK CATA LIB        =
.:/usr/lib64/alliance/cells/sxlib:/usr/lib64/alliance/cells/dp_sxlib:/usr/lib64/alliance/
cells/rflib:/usr/lib64/alliance/cells/rf2lib:/usr/lib64/alliance/cells/ramlib:/usr/lib64/
alliance/cells/romlib:/usr/lib64/alliance/cells/pxlib:/usr/lib64/alliance/cells/padlib
=======================  Files, Options and Parameters  =======================
First VHDL file     = digicodea.vbe
Second VHDL file    = digicodea b l net.vbe
The auxiliary signals are erased
Errors are displayed
================================================================================

Compiling 'digicodea' ...
Compiling 'digicodea b l net' ...
---> final number of nodes = 287(139)

Running Abl2Bdd on `digicodea b l net`
---------------------------------------------------------------------------------
             Formal proof with Ordered Binary Decision Diagrams between

             './digicodea'  and  './digicodea b l net'
---------------------------------------------------------------------------------
==============================  PRIMARY OUTPUT  ================================
=============================  AUXILIARY SIGNAL  ===============================
==============================  REGISTER SIGNAL  ===============================
===============================  EXTERNAL BUS ==================================
================================ INTERNAL BUS ==================================

                            Formal Proof : OK

pppppppppppppppppppppppprrrrrrrrrrrrooooooooooooooooooooooooooooooofffffffffffffff
---------------------------------------------------------------------------------




```

## .path file



```
BEGIN_PATH_REG
digicode cs 0 ins
digicode_cs_1_ins
digicode cs 2 ins
END_PATH_REG

BEGIN_CONNECTOR
SCAN IN scanin
SCAN_OUT scanout
SCAN_TEST test
END CONNECTOR

```

```vhdl
---------------original behavioral fsm----------------------

library IEEE;
USE ieee.std_logic_1164.ALL;

entity DigiCode is
port  (
    ck    : in  bit;
    vdd   : in  bit;
    vss   : in  bit;
    code  : in  bit_vector (3 downto 0);
    reset : in  bit;
    day   : in  bit;
    alarm : out bit;
    door  : out bit
      );
end DigiCode;

architecture MOORE of DigiCode is
  type STATE_TYPE is (S0, S1, s2, s3, s4, s5, s6);
  signal NS, CS : STATE_TYPE;

begin
  process (CS, reset, code, day)
  begin

    if (reset='1') then
      NS<=S0;
    else
      case CS is
        when S0 =>
          if (day='1' and code = "1101") then --code = o
            NS <= S5;
          elsif (code = x"2") then
            NS <= s1;
          else
            Ns <= s6;
          end if;

        when S1 =>
          if (day='1' and code = "1101") then --code = o
            NS <= S5;
          elsif (code = x"6") then
            NS <= s2;
          else
            Ns <= s6;
          end if;


        when S2 =>
          if (day='1' and code = "1101") then --code = o
            NS <= S5;
          elsif (code = "1010") then
            NS <= s3;
          else
            Ns <= s6;
          end if;


        when S3 =>
          if (day='1' and code = "1101") then --code = o
            NS <= S5;
          elsif (code = x"0") then
            NS <= s4;
          else
            Ns <= s6;
          end if;


        when S4 =>
          if (day='1' and code = "1101") then --code = o
            NS <= S5;
          elsif (code = x"5") then
            NS <= s5;
          else
            Ns <= s6;
          end if;


        when S5 =>
          --door <= '1';
          null;
          --alarm <= '0';
          -- if (day='1' and code == '1100') then --code = o
            -- NS <= S5;
          -- elsif (code == '0')
            -- NS <= s4;
          -- else
            -- Ns <= s6;
          -- end if;


        when S6 =>
          --door <= '0';
          --alarm <= '1';
          null;
          -- if (day='1' and code == '1100') then --code = o
            -- NS <= S6;
          -- elsif (code == '0')
            -- NS <= s5;
          -- else
            -- Ns <= s6;
          -- end if;

        when others => assert(false)
        report "illegal state" severity error;


      end case;
end if;

case cs is
    when s0 =>
    door <= '0';
    alarm <= '0';
    when s1 =>
    door <= '0';
    alarm <= '0';
    when s2 =>
    door <= '0';
    alarm <= '0';
    when s3 =>
    door <= '0';
    alarm <= '0';
    when s4 =>
    door <= '0';
    alarm <= '0';
    when s5 =>
    door <= '1';
    alarm <= '0';
    when s6 =>
    door <= '0';
```

```vhdl
    alarm <= '1';
    end case;
  end process;

-- Process (2): State update (sequential)
  process(ck)
  begin
    if(ck = '1' and not ck'stable)then
      CS <= NS;
    end if;
  end process;

end MOORE;
```

```vhdl
---------------structural output from loon----------------------

LIBRARY sxlib_ModelSim;

entity digicodea_b_l is
   port (
      ck    : in      bit;
      vdd   : in      bit;
      vss   : in      bit;
      code  : in      bit_vector(3 downto 0);
      reset : in      bit;
      day   : in      bit;
      alarm : out     bit;
      door  : out     bit
 );
end digicodea_b_l;

architecture structural of digicodea_b_l is
Component oa22_x2
   port (
      i0  : in      bit;
      i1  : in      bit;
      i2  : in      bit;
      q   : out     bit;
      vdd : in      bit;
      vss : in      bit
 );
end component;

Component o2_x2
   port (
      i0  : in      bit;
      i1  : in      bit;
      q   : out     bit;
      vdd : in      bit;
      vss : in      bit
 );
end component;

Component na3_x1
   port (
      i0  : in      bit;
      i1  : in      bit;
      i2  : in      bit;
      nq  : out     bit;
      vdd : in      bit;
      vss : in      bit
 );
end component;

Component na2_x1
   port (
      i0  : in      bit;
      i1  : in      bit;
      nq  : out     bit;
      vdd : in      bit;
      vss : in      bit
 );
end component;

Component no4_x1
   port (
      i0  : in      bit;
      i1  : in      bit;
      i2  : in      bit;
      i3  : in      bit;
```

```
         nq  : out     bit;
         vdd : in      bit;
         vss : in      bit
   );
  end component;

  Component inv_x2
     port (
         i   : in      bit;
         nq  : out     bit;
         vdd : in      bit;
         vss : in      bit
   );
  end component;

  Component no3_x1
     port (
         i0  : in      bit;
         i1  : in      bit;
         i2  : in      bit;
         nq  : out     bit;
         vdd : in      bit;
         vss : in      bit
   );
  end component;

  Component on12_x1
     port (
         i0  : in      bit;
         i1  : in      bit;
         q   : out     bit;
         vdd : in      bit;
         vss : in      bit
   );
  end component;

  Component na4_x1
     port (
         i1  : in      bit;
         i0  : in      bit;
         i3  : in      bit;
         i2  : in      bit;
         nq  : out     bit;
         vdd : in      bit;
         vss : in      bit
   );
  end component;

  Component nao22_x1
     port (
         i1  : in      bit;
         i0  : in      bit;
         i2  : in      bit;
         nq  : out     bit;
         vdd : in      bit;
         vss : in      bit
   );
  end component;

  Component sff1_x4
     port (
         ck  : in      bit;
         i   : in      bit;
         q   : out     bit;
         vdd : in      bit;
         vss : in      bit
```

```
 );
end component;

Component a2_x2
   port (
      i0  : in      bit;
      i1  : in      bit;
      q   : out     bit;
      vdd : in      bit;
      vss : in      bit
 );
end component;

Component no2_x1
   port (
      i0  : in      bit;
      i1  : in      bit;
      nq  : out     bit;
      vdd : in      bit;
      vss : in      bit
 );
end component;

signal digicode_cs     : bit_vector( 2 downto 0);
signal not_code        : bit_vector( 2 downto 1);
signal not_digicode_cs : bit_vector( 2 downto 0);
signal on12_x1_sig     : bit;
signal on12_x1_2_sig   : bit;
signal oa22_x2_sig     : bit;
signal o2_x2_sig       : bit;
signal not_reset       : bit;
signal not_aux3        : bit;
signal not_aux2        : bit;
signal not_aux1        : bit;
signal not_aux0        : bit;
signal no4_x1_sig      : bit;
signal no4_x1_2_sig    : bit;
signal no3_x1_sig      : bit;
signal no2_x1_sig      : bit;
signal no2_x1_2_sig    : bit;
signal nao22_x1_sig    : bit;
signal na4_x1_sig      : bit;
signal na4_x1_3_sig    : bit;
signal na4_x1_2_sig    : bit;
signal na3_x1_sig      : bit;
signal na3_x1_3_sig    : bit;
signal na3_x1_2_sig    : bit;
signal na2_x1_sig      : bit;
signal na2_x1_4_sig    : bit;
signal na2_x1_3_sig    : bit;
signal na2_x1_2_sig    : bit;
signal inv_x2_sig      : bit;

begin

not_aux0_ins : o2_x2
   port map (
      i0  => digicode_cs(2),
      i1  => digicode_cs(0),
      q   => not_aux0,
      vdd => vdd,
      vss => vss
   );

not_aux2_ins : a2_x2
   port map (
```

```vhdl
         i0  => not_digicode_cs(1),
         i1  => not_digicode_cs(0),
         q   => not_aux2,
         vdd => vdd,
         vss => vss
      );

   not_aux1_ins : o2_x2
      port map (
         i0  => code(3),
         i1  => not_code(2),
         q   => not_aux1,
         vdd => vdd,
         vss => vss
      );

   not_aux3_ins : o2_x2
      port map (
         i0  => code(0),
         i1  => not_code(1),
         q   => not_aux3,
         vdd => vdd,
         vss => vss
      );

   not_digicode_cs_1_ins : inv_x2
      port map (
         i   => digicode_cs(1),
         nq  => not_digicode_cs(1),
         vdd => vdd,
         vss => vss
      );

   not_digicode_cs_2_ins : inv_x2
      port map (
         i   => digicode_cs(2),
         nq  => not_digicode_cs(2),
         vdd => vdd,
         vss => vss
      );

   not_digicode_cs_0_ins : inv_x2
      port map (
         i   => digicode_cs(0),
         nq  => not_digicode_cs(0),
         vdd => vdd,
         vss => vss
      );

   not_code_2_ins : inv_x2
      port map (
         i   => code(2),
         nq  => not_code(2),
         vdd => vdd,
         vss => vss
      );

   not_code_1_ins : inv_x2
      port map (
         i   => code(1),
         nq  => not_code(1),
         vdd => vdd,
         vss => vss
      );

   not_reset_ins : inv_x2
```

```
      port map (
         i   => reset,
         nq  => not_reset,
         vdd => vdd,
         vss => vss
      );

   oa22_x2_ins : oa22_x2
      port map (
         i0  => code(3),
         i1  => not_code(2),
         i2  => digicode_cs(0),
         q   => oa22_x2_sig,
         vdd => vdd,
         vss => vss
      );

   o2_x2_ins : o2_x2
      port map (
         i0  => code(2),
         i1  => not_aux3,
         q   => o2_x2_sig,
         vdd => vdd,
         vss => vss
      );

   no2_x1_ins : no2_x1
      port map (
         i0  => o2_x2_sig,
         i1  => not_digicode_cs(1),
         nq  => no2_x1_sig,
         vdd => vdd,
         vss => vss
      );

   na2_x1_2_ins : na2_x1
      port map (
         i0  => code(3),
         i1  => not_digicode_cs(2),
         nq  => na2_x1_2_sig,
         vdd => vdd,
         vss => vss
      );

   na3_x1_ins : na3_x1
      port map (
         i0  => na2_x1_2_sig,
         i1  => no2_x1_sig,
         i2  => oa22_x2_sig,
         nq  => na3_x1_sig,
         vdd => vdd,
         vss => vss
      );

   na2_x1_ins : na2_x1
      port map (
         i0  => not_reset,
         i1  => na3_x1_sig,
         nq  => na2_x1_sig,
         vdd => vdd,
         vss => vss
      );

   digicode_cs_0_ins : sff1_x4
      port map (
         ck  => ck,
```

```vhdl
         i   => na2_x1_sig,
         q   => digicode_cs(0),
         vdd => vdd,
         vss => vss
      );

   no4_x1_ins : no4_x1
      port map (
         i0  => code(0),
         i1  => code(2),
         i2  => code(1),
         i3  => code(3),
         nq  => no4_x1_sig,
         vdd => vdd,
         vss => vss
      );

   na4_x1_ins : na4_x1
      port map (
         i0  => not_digicode_cs(1),
         i1  => no4_x1_sig,
         i2  => digicode_cs(0),
         i3  => not_digicode_cs(2),
         nq  => na4_x1_sig,
         vdd => vdd,
         vss => vss
      );

   no2_x1_2_ins : no2_x1
      port map (
         i0  => not_aux3,
         i1  => not_aux1,
         nq  => no2_x1_2_sig,
         vdd => vdd,
         vss => vss
      );

   na3_x1_3_ins : na3_x1
      port map (
         i0  => digicode_cs(2),
         i1  => no2_x1_2_sig,
         i2  => digicode_cs(0),
         nq  => na3_x1_3_sig,
         vdd => vdd,
         vss => vss
      );

   na3_x1_2_ins : na3_x1
      port map (
         i0  => not_reset,
         i1  => na3_x1_3_sig,
         i2  => na4_x1_sig,
         nq  => na3_x1_2_sig,
         vdd => vdd,
         vss => vss
      );

   digicode_cs_1_ins : sff1_x4
      port map (
         ck  => ck,
         i   => na3_x1_2_sig,
         q   => digicode_cs(1),
         vdd => vdd,
         vss => vss
      );

```

```
na2_x1_3_ins : na2_x1
   port map (
      i0  => code(2),
      i1  => not_digicode_cs(2),
      nq  => na2_x1_3_sig,
      vdd => vdd,
      vss => vss
   );

na2_x1_4_ins : na2_x1
   port map (
      i0  => not_aux1,
      i1  => not_digicode_cs(1),
      nq  => na2_x1_4_sig,
      vdd => vdd,
      vss => vss
   );

no4_x1_2_ins : no4_x1
   port map (
      i0  => reset,
      i1  => not_code(1),
      i2  => code(0),
      i3  => code(3),
      nq  => no4_x1_2_sig,
      vdd => vdd,
      vss => vss
   );

na4_x1_2_ins : na4_x1
   port map (
      i0  => digicode_cs(0),
      i1  => no4_x1_2_sig,
      i2  => na2_x1_4_sig,
      i3  => na2_x1_3_sig,
      nq  => na4_x1_2_sig,
      vdd => vdd,
      vss => vss
   );

inv_x2_ins : inv_x2
   port map (
      i   => code(0),
      nq  => inv_x2_sig,
      vdd => vdd,
      vss => vss
   );

no3_x1_ins : no3_x1
   port map (
      i0  => code(1),
      i1  => inv_x2_sig,
      i2  => reset,
      nq  => no3_x1_sig,
      vdd => vdd,
      vss => vss
   );

on12_x1_ins : on12_x1
   port map (
      i0  => not_aux0,
      i1  => code(3),
      q   => on12_x1_sig,
      vdd => vdd,
      vss => vss
   );
```

```vhdl

on12_x1_2_ins : on12_x1
   port map (
      i0  => code(3),
      i1  => day,
      q   => on12_x1_2_sig,
      vdd => vdd,
      vss => vss
   );

na4_x1_3_ins : na4_x1
   port map (
      i1  => on12_x1_2_sig,
      i0  => code(2),
      i3  => on12_x1_sig,
      i2  => no3_x1_sig,
      nq  => na4_x1_3_sig,
      vdd => vdd,
      vss => vss
   );

nao22_x1_ins : nao22_x1
   port map (
      i1  => na4_x1_3_sig,
      i0  => not_aux2,
      i2  => na4_x1_2_sig,
      nq  => nao22_x1_sig,
      vdd => vdd,
      vss => vss
   );

digicode_cs_2_ins : sff1_x4
   port map (
      ck  => ck,
      i   => nao22_x1_sig,
      q   => digicode_cs(2),
      vdd => vdd,
      vss => vss
   );

door_ins : a2_x2
   port map (
      i0  => not_aux2,
      i1  => digicode_cs(2),
      q   => door,
      vdd => vdd,
      vss => vss
   );

alarm_ins : no2_x1
   port map (
      i0  => not_aux0,
      i1  => digicode_cs(1),
      nq  => alarm,
      vdd => vdd,
      vss => vss
   );


end structural;

```

```vhdl
---------------structural output from scapin----------------------

LIBRARY sxlib_ModelSim;

entity digicodea_b_l_scan is
   port (
      ck      : in      bit;
      vdd     : in      bit;
      vss     : in      bit;
      code    : in      bit_vector(3 downto 0);
      reset   : in      bit;
      day     : in      bit;
      alarm   : out     bit;
      door    : out     bit;
      scanin  : in      bit;
      test    : in      bit;
      scanout : out     bit
 );
end digicodea_b_l_scan;

architecture structural of digicodea_b_l_scan is
Component oa22_x2
   port (
      i0  : in      bit;
      i1  : in      bit;
      i2  : in      bit;
      q   : out     bit;
      vdd : in      bit;
      vss : in      bit
 );
end component;

Component o2_x2
   port (
      i0  : in      bit;
      i1  : in      bit;
      q   : out     bit;
      vdd : in      bit;
      vss : in      bit
 );
end component;

Component na3_x1
   port (
      i0  : in      bit;
      i1  : in      bit;
      i2  : in      bit;
      nq  : out     bit;
      vdd : in      bit;
      vss : in      bit
 );
end component;

Component na2_x1
   port (
      i0  : in      bit;
      i1  : in      bit;
      nq  : out     bit;
      vdd : in      bit;
      vss : in      bit
 );
end component;

Component no4_x1
   port (
      i0  : in      bit;
```

```vhdl
         i1  : in      bit;
         i2  : in      bit;
         i3  : in      bit;
         nq  : out     bit;
         vdd : in      bit;
         vss : in      bit
  );
 end component;

 Component inv_x2
    port (
         i   : in      bit;
         nq  : out     bit;
         vdd : in      bit;
         vss : in      bit
  );
 end component;

 Component no3_x1
    port (
         i0  : in      bit;
         i1  : in      bit;
         i2  : in      bit;
         nq  : out     bit;
         vdd : in      bit;
         vss : in      bit
  );
 end component;

 Component on12_x1
    port (
         i0  : in      bit;
         i1  : in      bit;
         q   : out     bit;
         vdd : in      bit;
         vss : in      bit
  );
 end component;

 Component na4_x1
    port (
         i1  : in      bit;
         i0  : in      bit;
         i3  : in      bit;
         i2  : in      bit;
         nq  : out     bit;
         vdd : in      bit;
         vss : in      bit
  );
 end component;

 Component nao22_x1
    port (
         i1  : in      bit;
         i0  : in      bit;
         i2  : in      bit;
         nq  : out     bit;
         vdd : in      bit;
         vss : in      bit
  );
 end component;

 Component a2_x2
    port (
         i0  : in      bit;
         i1  : in      bit;
```

```
      q   : out     bit;
      vdd : in      bit;
      vss : in      bit
 );
end component;

Component no2_x1
   port (
      i0  : in      bit;
      i1  : in      bit;
      nq  : out     bit;
      vdd : in      bit;
      vss : in      bit
 );
end component;

Component sff2_x4
   port (
      ck  : in      bit;
      cmd : in      bit;
      i0  : in      bit;
      i1  : in      bit;
      q   : out     bit;
      vdd : in      bit;
      vss : in      bit
 );
end component;

Component buf_x2
   port (
      i   : in      bit;
      q   : out     bit;
      vdd : in      bit;
      vss : in      bit
 );
end component;

signal digicode_cs     : bit_vector( 2 downto 0);
signal not_code        : bit_vector( 2 downto 1);
signal not_digicode_cs : bit_vector( 2 downto 0);
signal on12_x1_sig     : bit;
signal on12_x1_2_sig   : bit;
signal oa22_x2_sig     : bit;
signal o2_x2_sig       : bit;
signal not_reset       : bit;
signal not_aux3        : bit;
signal not_aux2        : bit;
signal not_aux1        : bit;
signal not_aux0        : bit;
signal no4_x1_sig      : bit;
signal no4_x1_2_sig    : bit;
signal no3_x1_sig      : bit;
signal no2_x1_sig      : bit;
signal no2_x1_2_sig    : bit;
signal nao22_x1_sig    : bit;
signal na4_x1_sig      : bit;
signal na4_x1_3_sig    : bit;
signal na4_x1_2_sig    : bit;
signal na3_x1_sig      : bit;
signal na3_x1_3_sig    : bit;
signal na3_x1_2_sig    : bit;
signal na2_x1_sig      : bit;
signal na2_x1_4_sig    : bit;
signal na2_x1_3_sig    : bit;
signal na2_x1_2_sig    : bit;
signal inv_x2_sig      : bit;
```

```vhdl

begin

not_aux0_ins : o2_x2
   port map (
      i0  => digicode_cs(2),
      i1  => digicode_cs(0),
      q   => not_aux0,
      vdd => vdd,
      vss => vss
   );

not_aux2_ins : a2_x2
   port map (
      i0  => not_digicode_cs(1),
      i1  => not_digicode_cs(0),
      q   => not_aux2,
      vdd => vdd,
      vss => vss
   );

not_aux1_ins : o2_x2
   port map (
      i0  => code(3),
      i1  => not_code(2),
      q   => not_aux1,
      vdd => vdd,
      vss => vss
   );

not_aux3_ins : o2_x2
   port map (
      i0  => code(0),
      i1  => not_code(1),
      q   => not_aux3,
      vdd => vdd,
      vss => vss
   );

not_digicode_cs_1_ins : inv_x2
   port map (
      i   => digicode_cs(1),
      nq  => not_digicode_cs(1),
      vdd => vdd,
      vss => vss
   );

not_digicode_cs_2_ins : inv_x2
   port map (
      i   => digicode_cs(2),
      nq  => not_digicode_cs(2),
      vdd => vdd,
      vss => vss
   );

not_digicode_cs_0_ins : inv_x2
   port map (
      i   => digicode_cs(0),
      nq  => not_digicode_cs(0),
      vdd => vdd,
      vss => vss
   );

not_code_2_ins : inv_x2
   port map (
      i   => code(2),
      nq  => not_code(2),
      vdd => vdd,
      vss => vss
   );

not_code_1_ins : inv_x2
   port map (
      i   => code(1),
      nq  => not_code(1),
      vdd => vdd,
      vss => vss
   );

not_reset_ins : inv_x2
   port map (
      i   => reset,
      nq  => not_reset,
      vdd => vdd,
      vss => vss
   );

oa22_x2_ins : oa22_x2
   port map (
      i0  => code(3),
      i1  => not_code(2),
      i2  => digicode_cs(0),
      q   => oa22_x2_sig,
      vdd => vdd,
      vss => vss
   );

o2_x2_ins : o2_x2
   port map (
      i0  => code(2),
      i1  => not_aux3,
      q   => o2_x2_sig,
      vdd => vdd,
      vss => vss
   );

no2_x1_ins : no2_x1
   port map (
      i0  => o2_x2_sig,
      i1  => not_digicode_cs(1),
      nq  => no2_x1_sig,
      vdd => vdd,
      vss => vss
   );

na2_x1_2_ins : na2_x1
   port map (
      i0  => code(3),
      i1  => not_digicode_cs(2),
      nq  => na2_x1_2_sig,
      vdd => vdd,
      vss => vss
   );

na3_x1_ins : na3_x1
   port map (
      i0  => na2_x1_2_sig,
      i1  => no2_x1_sig,
      i2  => oa22_x2_sig,
      nq  => na3_x1_sig,
      vdd => vdd,
      vss => vss
```

```vhdl
      );

   na2_x1_ins : na2_x1
      port map (
         i0  => not_reset,
         i1  => na3_x1_sig,
         nq  => na2_x1_sig,
         vdd => vdd,
         vss => vss
      );

   no4_x1_ins : no4_x1
      port map (
         i0  => code(0),
         i1  => code(2),
         i2  => code(1),
         i3  => code(3),
         nq  => no4_x1_sig,
         vdd => vdd,
         vss => vss
      );

   na4_x1_ins : na4_x1
      port map (
         i1  => no4_x1_sig,
         i0  => not_digicode_cs(1),
         i3  => not_digicode_cs(2),
         i2  => digicode_cs(0),
         nq  => na4_x1_sig,
         vdd => vdd,
         vss => vss
      );

   no2_x1_2_ins : no2_x1
      port map (
         i0  => not_aux3,
         i1  => not_aux1,
         nq  => no2_x1_2_sig,
         vdd => vdd,
         vss => vss
      );

   na3_x1_3_ins : na3_x1
      port map (
         i0  => digicode_cs(2),
         i1  => no2_x1_2_sig,
         i2  => digicode_cs(0),
         nq  => na3_x1_3_sig,
         vdd => vdd,
         vss => vss
      );

   na3_x1_2_ins : na3_x1
      port map (
         i0  => not_reset,
         i1  => na3_x1_3_sig,
         i2  => na4_x1_sig,
         nq  => na3_x1_2_sig,
         vdd => vdd,
         vss => vss
      );

   na2_x1_3_ins : na2_x1
      port map (
         i0  => code(2),
         i1  => not_digicode_cs(2),
         nq  => na2_x1_3_sig,
         vdd => vdd,
         vss => vss
      );

   na2_x1_4_ins : na2_x1
      port map (
         i0  => not_aux1,
         i1  => not_digicode_cs(1),
         nq  => na2_x1_4_sig,
         vdd => vdd,
         vss => vss
      );

   no4_x1_2_ins : no4_x1
      port map (
         i0  => reset,
         i1  => not_code(1),
         i2  => code(0),
         i3  => code(3),
         nq  => no4_x1_2_sig,
         vdd => vdd,
         vss => vss
      );

   na4_x1_2_ins : na4_x1
      port map (
         i1  => no4_x1_2_sig,
         i0  => digicode_cs(0),
         i3  => na2_x1_3_sig,
         i2  => na2_x1_4_sig,
         nq  => na4_x1_2_sig,
         vdd => vdd,
         vss => vss
      );

   inv_x2_ins : inv_x2
      port map (
         i   => code(0),
         nq  => inv_x2_sig,
         vdd => vdd,
         vss => vss
      );

   no3_x1_ins : no3_x1
      port map (
         i0  => code(1),
         i1  => inv_x2_sig,
         i2  => reset,
         nq  => no3_x1_sig,
         vdd => vdd,
         vss => vss
      );

   on12_x1_ins : on12_x1
      port map (
         i0  => not_aux0,
         i1  => code(3),
         q   => on12_x1_sig,
         vdd => vdd,
         vss => vss
      );

   on12_x1_2_ins : on12_x1
      port map (
         i0  => code(3),
         i1  => day,
         q   => on12_x1_2_sig,
         vdd => vdd,
         vss => vss
      );

   na4_x1_3_ins : na4_x1
      port map (
         i1  => on12_x1_2_sig,
         i0  => code(2),
         i3  => on12_x1_sig,
         i2  => no3_x1_sig,
         nq  => na4_x1_3_sig,
         vdd => vdd,
         vss => vss
      );

   nao22_x1_ins : nao22_x1
      port map (
         i1  => na4_x1_3_sig,
         i0  => not_aux2,
         i2  => na4_x1_2_sig,
         nq  => nao22_x1_sig,
         vdd => vdd,
         vss => vss
      );

   door_ins : a2_x2
      port map (
         i0  => not_aux2,
         i1  => digicode_cs(2),
         q   => door,
         vdd => vdd,
         vss => vss
      );

   alarm_ins : no2_x1
      port map (
         i0  => not_aux0,
         i1  => digicode_cs(1),
         nq  => alarm,
         vdd => vdd,
         vss => vss
      );

   digicode_cs_0_ins_scan_0 : sff2_x4
      port map (
         ck  => ck,
         cmd => test,
         i0  => na2_x1_sig,
         i1  => scanin,
         q   => digicode_cs(0),
         vdd => vdd,
         vss => vss
      );

   digicode_cs_1_ins_scan_1 : sff2_x4
      port map (
         ck  => ck,
         cmd => test,
         i0  => na3_x1_2_sig,
         i1  => digicode_cs(0),
         q   => digicode_cs(1),
         vdd => vdd,
         vss => vss
      );
```

```vhdl

digicode_cs_2_ins_scan_2 : sff2_x4
   port map (
      ck  => ck,
      cmd => test,
      i0  => nao22_x1_sig,
      i1  => digicode_cs(1),
      q   => digicode_cs(2),
      vdd => vdd,
      vss => vss
   );

buf_scan_3 : buf_x2
   port map (
      i   => digicode_cs(2),
      q   => scanout,
      vdd => vdd,
      vss => vss
   );


end structural;
```

```vhdl
---------------test bench of behavioral, loon output, scapin output-------------------

library IEEE;
USE ieee.std_logic_1164.ALL;

-- Entity declaration for your testbench. Don't declare any ports here
ENTITY testbench_DigiCode_st IS
END ENTITY testbench_DigiCode_st;

ARCHITECTURE testbench_DigiCode_st OF testbench_DigiCode_st IS

-- Component Declaration for the Device Under Test (DUT)
COMPONENT DigiCode IS
port  (
    ck    : in  bit;
    vdd   : in  bit;
    vss   : in  bit;
    code  : in  bit_vector (3 downto 0);
    reset : in  bit;
    day   : in  bit;
    alarm : out bit;
    door  : out bit
      );
END COMPONENT DigiCode;

FOR dut_beh: DigiCode USE ENTITY WORK.DigiCode (MOORE);

COMPONENT digicodea_b_l IS
port  (
    ck    : in  bit;
    vdd   : in  bit;
    vss   : in  bit;
    code  : in  bit_vector(3 downto 0);
    reset : in  bit;
    day   : in  bit;
    alarm : out bit;
    door  : out bit
      );
END COMPONENT digicodea_b_l;

FOR dut_st: digicodea_b_l USE ENTITY WORK.digicodea_b_l (structural);

COMPONENT digicodea_b_l_scan IS
port  (
  ck      : in      bit;
  vdd     : in      bit;
  vss     : in      bit;
  code    : in      bit_vector(3 downto 0);
  reset   : in      bit;
  day     : in      bit;
  alarm   : out     bit;
  door    : out     bit;
  scanin  : in      bit;
  test    : in      bit;
  scanout : out     bit
      );
END COMPONENT digicodea_b_l_scan;

FOR dut_scan: digicodea_b_l_scan USE ENTITY WORK.digicodea_b_l_scan (structural);


-- Declare input signals and initialize them
SIGNAL ck    : bit := '0';
SIGNAL vdd   : bit := '1';
SIGNAL vss   : bit := '0';
SIGNAL code  : bit_vector(3 downto 0) := x"0";
SIGNAL reset : bit := '0';
SIGNAL day   : bit := '0';

SIGNAL test    : bit := '0';

SIGNAL door_b  : bit := '0';
SIGNAL alarm_b : bit := '0';

SIGNAL door_st  : bit := '0';
SIGNAL alarm_st : bit := '0';

SIGNAL door_scan  : bit := '0';
SIGNAL alarm_scan : bit := '0';

SIGNAL scanin  : bit := '0';
SIGNAL scanout : bit := '0';

-- Constants and Clock period definitions
constant clk_period : time := 20 ns;
constant sequence : bit_vector := "0011111011011100100111100";
BEGIN

-- Instantiate the Device Under Test (DUT)
dut_beh: DigiCode PORT MAP (ck, vdd, vss, code, reset, day, alarm_b, door_b);
dut_st: digicodea_b_l PORT MAP (ck, vdd, vss, code, reset, day, alarm_st, door_st);
dut_scan: digicodea_b_l_scan PORT MAP (ck, vdd, vss, code, reset, day, alarm_scan,
door_scan, scanin, test, scanout);

-- Clock process definitions( clock with 50% duty cycle )
   clk_process :process
   begin
        ck <= '0';
        wait for clk_period/2;
        ck <= '1';
        wait for clk_period/2;
   end process;

-- Stimulus process, refer to clock signal
stim_proc: PROCESS IS
BEGIN
    --reset case added to stabllise output at the beginnning
    code <= x"0";
    reset <= '1';
    test <= '0';
    WAIT FOR clk_period;
    ASSERT alarm_b = '0' and door_b = '0' and alarm_st = '0' and door_st = '0' and
    alarm_scan = '0' and door_scan = '0'
    REPORT "output error"
    SEVERITY error;

    -- case 1 correct code entered at night "day = 0"
    code <= x"2";
    reset <= '0';
    WAIT FOR clk_period;
    ASSERT alarm_b = '0' and door_b = '0' and alarm_st = '0' and door_st = '0' and
    alarm_scan = '0' and door_scan = '0'
    REPORT "output error"
    SEVERITY error;

    code <= x"6";
    WAIT FOR clk_period;
    ASSERT alarm_b = '0' and door_b = '0' and alarm_st = '0' and door_st = '0' and
    alarm_scan = '0' and door_scan = '0'
    REPORT "output error"
    SEVERITY error;
```

```

        code <= x"a";
        WAIT FOR clk_period;
        ASSERT alarm_b = '0' and door_b = '0' and alarm_st = '0' and door_st = '0' and
        alarm_scan = '0' and door_scan = '0'
        REPORT "output error"
        SEVERITY error;

        code <= x"0";
        WAIT FOR clk_period;
        ASSERT alarm_b = '0' and door_b = '0' and alarm_st = '0' and door_st = '0' and
        alarm_scan = '0' and door_scan = '0'
        REPORT "output error"
        SEVERITY error;

        code <= x"5";
        WAIT FOR clk_period;
        ASSERT alarm_b = '0' and door_b = '1' and alarm_st = '0' and door_st = '1' and
        alarm_scan = '0' and door_scan = '1'
        REPORT "output error"
        SEVERITY error;

        --case 2 reset test
        code <= x"0";
        reset <= '1';
        WAIT FOR clk_period;
        ASSERT alarm_b = '0' and door_b = '0' and alarm_st = '0' and door_st = '0' and
        alarm_scan = '0' and door_scan = '0'
        REPORT "output error"
        SEVERITY error;

        --case 3 testing wrong code at night "day = 0"
        code <= x"1";
        reset <= '0';
        WAIT FOR clk_period;
        ASSERT alarm_b = '1' and door_b = '0' and alarm_st = '1' and door_st = '0' and
        alarm_scan = '1' and door_scan = '0'
        REPORT "output error"
        SEVERITY error;

        --returning to s0
        reset <= '1';
        WAIT FOR clk_period;
        ASSERT alarm_b = '0' and door_b = '0' and alarm_st = '0' and door_st = '0' and
        alarm_scan = '0' and door_scan = '0'
        REPORT "output error"
        SEVERITY error;

        --case 4 testing correct code with day = 1
        code <= x"2";
        day <= '1';
        reset <= '0';
        WAIT FOR clk_period;
        ASSERT alarm_b = '0' and door_b = '0' and alarm_st = '0' and door_st = '0' and
        alarm_scan = '0' and door_scan = '0'
        REPORT "output error"
        SEVERITY error;

        code <= x"6";
        day <= '1';
        reset <= '0';
        WAIT FOR clk_period;
        ASSERT alarm_b = '0' and door_b = '0' and alarm_st = '0' and door_st = '0' and
        alarm_scan = '0' and door_scan = '0'
        REPORT "output error"
        SEVERITY error;
```

```vhdl

        code <= x"a";
        day <= '1';
        reset <= '0';
        WAIT FOR clk_period;
        ASSERT alarm_b = '0' and door_b = '0' and alarm_st = '0' and door_st = '0' and
        alarm_scan = '0' and door_scan = '0'
        REPORT "output error"
        SEVERITY error;

        code <= x"0";
        day <= '1';
        reset <= '0';
        WAIT FOR clk_period;
        ASSERT alarm_b = '0' and door_b = '0' and alarm_st = '0' and door_st = '0' and
        alarm_scan = '0' and door_scan = '0'
        REPORT "output error"
        SEVERITY error;

        code <= x"5";
        day <= '1';
        reset <= '0';
        WAIT FOR clk_period;
        ASSERT alarm_b = '0' and door_b = '1' and alarm_st = '0' and door_st = '1' and
        alarm_scan = '0' and door_scan = '1'
        REPORT "output error"
        SEVERITY error;

        --case 5 testing rest when day = 1
        reset <= '1';
        WAIT FOR clk_period;
        ASSERT alarm_b = '0' and door_b = '0' and alarm_st = '0' and door_st = '0' and
        alarm_scan = '0' and door_scan = '0'
        REPORT "output error"
        SEVERITY error;

        --case 6 testing "o" button when day = 1
        code <= "1101";
        day <= '1';
        reset <= '0';
        WAIT FOR clk_period;
        ASSERT alarm_b = '0' and door_b = '1' and alarm_st = '0' and door_st = '1' and
        alarm_scan = '0' and door_scan = '1'
        REPORT "output error"
        SEVERITY error;

        --returning to s0
        reset <= '1';
        WAIT FOR clk_period;
        ASSERT alarm_b = '0' and door_b = '0' and alarm_st = '0' and door_st = '0' and
        alarm_scan = '0' and door_scan = '0'
        REPORT "output error"
        SEVERITY error;

        --case 6 wrong code when day = 1
        code <= x"1";
        day <= '1';
        reset <= '0';
        WAIT FOR clk_period;
        ASSERT alarm_b = '1' and door_b = '0' and alarm_st = '1' and door_st = '0' and
        alarm_scan = '1' and door_scan = '0'
        REPORT "output error"
        SEVERITY error;

        --returning to s0
        reset <= '1';
```

```vhdl
    WAIT FOR clk_period;
    ASSERT alarm_b = '0' and door_b = '0' and alarm_st = '0' and door_st = '0' and
    alarm_scan = '0' and door_scan = '0'
    REPORT "output error"
    SEVERITY error;

    --case 7 test = 1
    day <= '0';
    test <= '1';
    for i in 0 to sequence'length-1 loop
      scanin <= sequence(i);
      wait for clk_period;
      if i>=2 then
        ASSERT scanout=sequence(i-2)
        REPORT "scanout does not follow scan in"
        SEVERITY error;
      end if;
    end loop;



WAIT; -- stop process simulation run

END PROCESS stim_proc;
END ARCHITECTURE testbench_DigiCode_st;



```