CSE215: EDA
Project
Phase (2)

Prepared by: Ayman Hesham Mohamed
Email: ayman19982016@outlookcom

Department: Computer Engineering
CESS
ID: 16P3037

Group:2

1-Introduction:

This project is about implementing logic synthesis of the RTL design of phase 1 with getting experienced with the complete digital logic synthesis flow based on the previous phase results staring from the verified FSM already implemented using ModelSim and now using the Alliance virtual machine

2-Table of BOOM output

	-a	-j	-m	-o	-r
Number of literals before optimization	156	159	158	149	160
Number of literals after optimization	65	73	72	86	84

3-Table of LOON output:

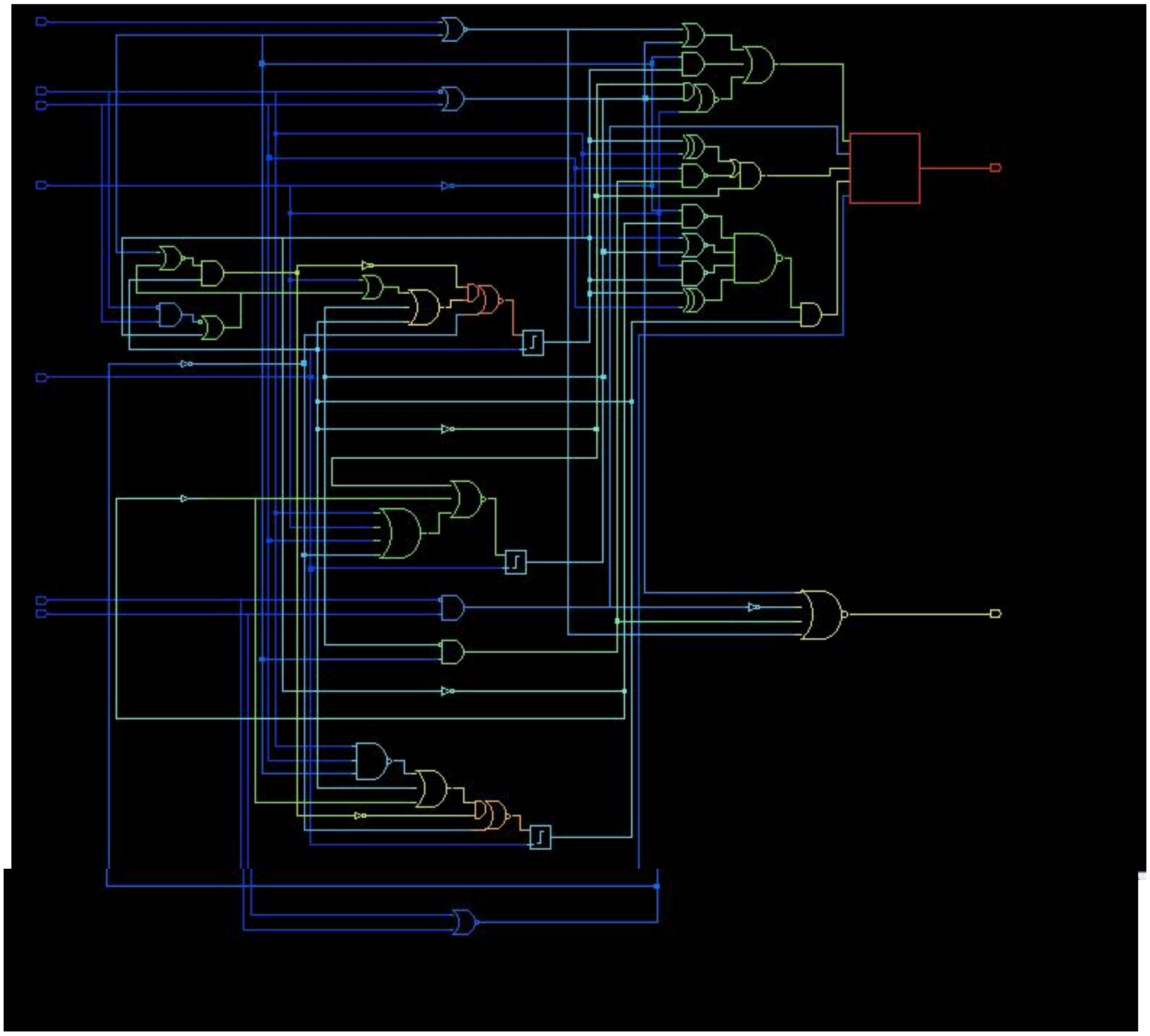
	-a	-j	-m	-o	-r
Delay (ps)	2388	3298	2583	2910	2822
Area (lamda^2)	63000	65750	69000	84500	73000

3-Table of BOOG output:

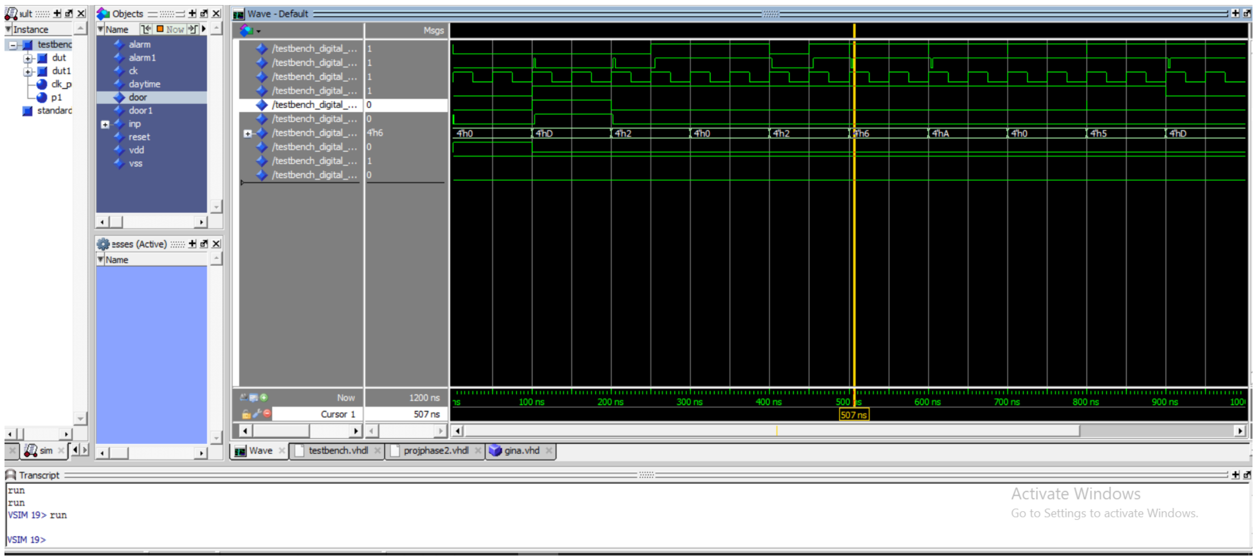
	-a	-j	-m	-o	-r
Delay (ps)	2341 ps	2246	2135	1778	2324
Area (lamda^2)	63000 lambda^2	65750	69000	84500	73000

I will choose the –a encoding because it has the best delay and area.

4- XSCH of –a encoding style from LOON

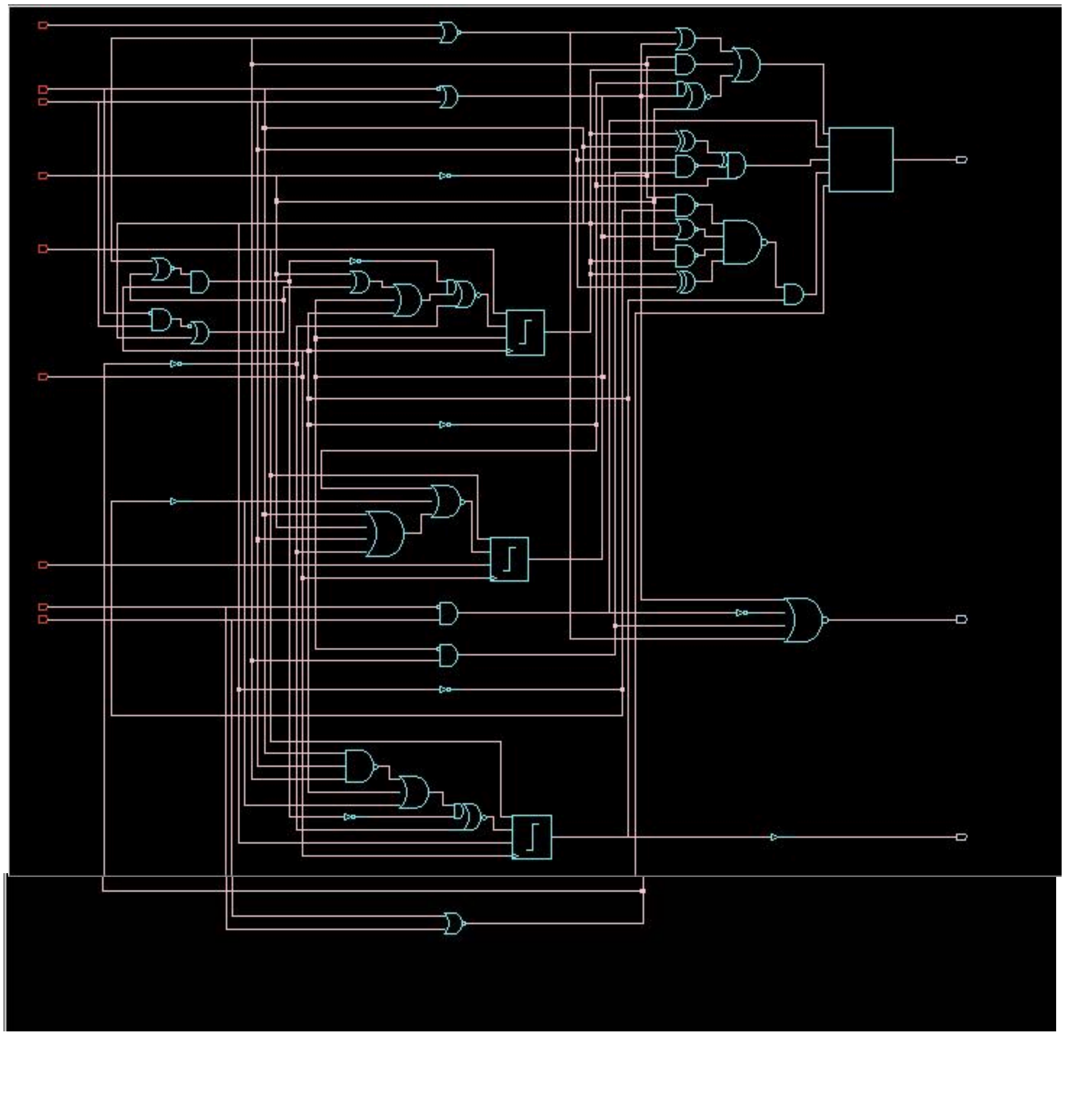


5- Snapchot of output daca_b_l.vhd compared with dac.vhd

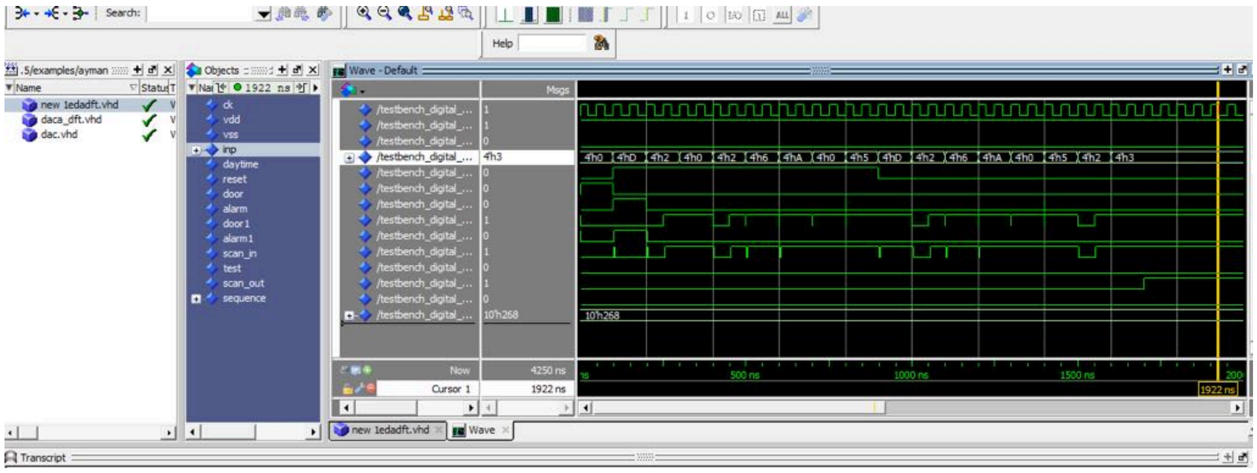


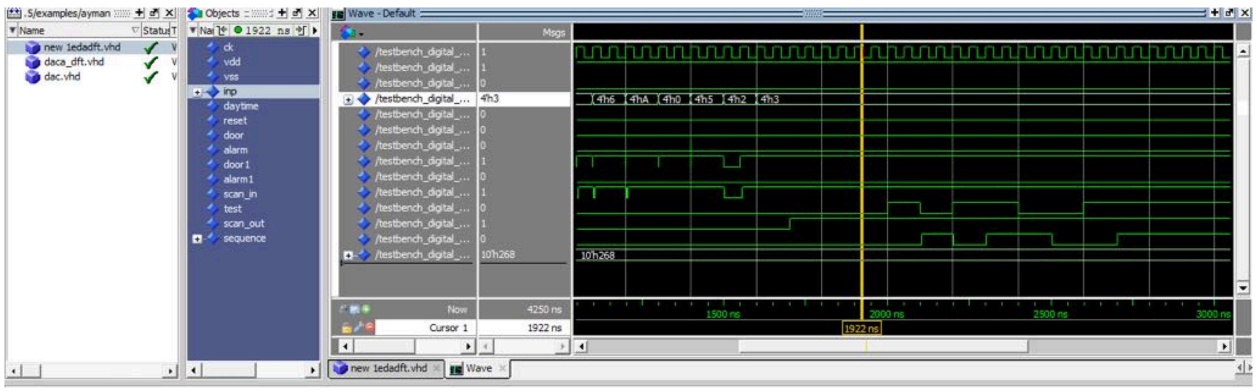
```
run
run
VSIM 19> run

VSIM 19>
```

6- XSCH of –a encoding style from DFT



7- Snapshot of output of daca_dft.vhd compared with dac.vhd





8 – Makefile:

```
#-------Project1------------------------------------------#
Syf: dacj.vbe \
        dacm.vbe \
        daco.vbe \
        dacr.vbe \
        daca.vbe
              @echo "<-- Generated SYF"

Boom: dacj_b.vbe \
         dacm_b.vbe \
         daco_b.vbe \
         dacr_b.vbe \
         daca_b.vbe
              @echo "<-- Generated BOOM"

Boog: dacj_b.vst \
         dacm_b.vst \
         daco_b.vst \
         dacr_b.vst \
        daca_b.vst
              @echo "<-- Generated BOOG"

Loon: dacj_b_l.vst \
        dacm_b_l.vst \
        daco_b_l.vst \
        dacr_b_l.vst \
        daca_b_l.vst
              @echo "<-- Generated LOON"

flatbeh: daca_b_l_net.vbe
              @echo "<-- Generated FLATBEH and PROOF"

dft: daca_dft.vst
              @echo "<-- Generated DFT"



#-------Finite State Machine Synthesis-----------------#

vhd_to_fsm:
	rename .vhd .fsm *.vhd

daca.vbe: dac.fsm
	@echo "    Encoding Synthesis -> daca.vbe"
	syf -CEV -a dac

dacj.vbe: dac.fsm
	@echo "    Encoding Synthesis  -> dacj.vbe"
	syf -CEV -j dac

dacm.vbe: dac.fsm
	@echo "    Encoding Synthesis  -> dacm.vbe"
	syf -CEV -m dac

daco.vbe: dac.fsm
	@echo "    Encoding Synthesis  -> daco.vbe"
	syf -CEV -o dac

dacr.vbe: dac.fsm
	@echo "    Encoding Synthesis  -> dacr.vbe"
	syf -CEV -r dac

%_b.vbe: %.vbe
	@echo "    Boolean Optimization  -> $@"
	boom -V -d 50 $* $*_b>$*_boom.out

%.vst: %.vbe paramfile.lax
	@echo "    Logical Synthesis  -> $@"
	boog -x 1 -l paramfile $*>$*_boog.out

%_l.vst: %.vbe paramfile.lax
	@echo "    Netlist Optimization  -> $@"
	loon -x 1 -l paramfile $* $*_l > $*_loon.out

%_b_l_net.vbe: %_b_l.vst %.vbe
	@echo "    Formal checking  -> $@"
	flatbeh $*_b_l $*_b_l_net > $*_flatbeh.out
	proof -d $* $*_b_l_net > $*_proof.out

daca_dft.vst : daca_b_l.vst  daca_dft.vst
```

```
	@echo "    DFT  -> $@"
	scapin -VRB daca_b_l path daca_dft> daca_DFT.out


#-------Clean Up--------------------------------------#
clean :
	rm -f  *.vbe *.enc *~
	@echo "Erase all the files generated by the makefile"
```

9- paramfile.lax:

```
#M{2}
#L{5}
#C{
	door:100;
	alarm:100;
}
```

10- path.path:

```
BEGIN_PATH_REG
dac_cs_0_ins
dac_cs_1_ins
dac_cs_2_ins
END_PATH_REG
```

```
BEGIN_CONNECTOR

SCAN_IN scan_in

SCAN_OUT scan_out

SCAN_TEST test

END_CONNECTOR
```

11- daca_proof.out:

```
      @@@@@@@                @@@
       @@  @@              @  @@
       @@   @@            @@  @@
       @@   @@ @@@ @@@   @@@    @@@   @@
       @@  @@  @@@ @@ @@  @@  @@  @@
@@@@@@@@
       @@@@@   @@  @@ @@   @@ @@   @@
@@
       @@    @@   @@   @@ @@   @@  @@
       @@    @@   @@   @@ @@   @@  @@
       @@    @@   @@   @@ @@   @@  @@
       @@    @@    @@  @@  @@  @@  @@
```

```
          @@@@@@    @@@@      @@@     @@@
@@@@@@

                  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/lib/alliance/cells/sxlib:/usr/lib/alliance/cells/dp_sxlib:/usr
/lib/alliance/cells/rflib:/usr/lib/alliance/cells/rf2lib:/usr/lib/allia
nce/cells/ramlib:/usr/lib/alliance/cells/romlib:/usr/lib/alliance/
cells/pxlib:/usr/lib/alliance/cells/padlib
====================== Files, Options and Parameters
======================
First VHDL file         = daca.vbe
Second VHDL file     = daca_b_l_net.vbe
The auxiliary signals are erased
```

```
Errors are displayed
====================================================
===========================


Compiling 'daca' ...
Compiling 'daca_b_l_net' ...
---> final number of nodes = 276(106)


Running Abl2Bdd on `daca_b_l_net`
------------------------------------------------------------------------------
      Formal proof with Ordered Binary Decision Diagrams
between


      './daca'  and  './daca_b_l_net'
------------------------------------------------------------------------------
===============================  PRIMARY OUTPUT
===============================
==============================  AUXILIARY SIGNAL
==============================
===============================  REGISTER SIGNAL
==============================
================================  EXTERNAL BUS
==================================
================================= INTERNAL BUS
==================================

                    Formal Proof : OK

pppppppppppppppppppppppppprrrrrrrrrroooooooooooooooooo
oooooooooooofffffffffffff
```

<u>**12- Testbench code for daca_b_l :**</u>

```
ENTITY testbench_Digital_Access_Control IS
END ENTITY testbench_Digital_Access_Control;


ARCHITECTURE test_Digital_Access_Control OF
testbench_Digital_Access_Control IS
Component dac is
port  (
        ck    : in    bit;
        vdd   : in   bit;
        vss   : in    bit;
        inp     : in  bit_vector (3 downto 0);
        daytime    : in   bit;
        reset : in  bit;
        alarm     : out   bit;
        door    : out     bit
      );
end Component dac;

Component daca_b_l is
  port (
      ck      : in      bit;
      vdd     : in      bit;
      vss     : in      bit;
      inp     : in      bit_vector(3 downto 0);
      daytime : in      bit;
      reset   : in      bit;
      alarm   : out     bit;
      door    : out     bit
 );
```

```
end Component daca_b_l;

FOR dut: dac USE ENTITY WORK.dac (fsm);

FOR dut1: daca_b_l USE ENTITY WORK.daca_b_l (structural);

SIGNAL ck    : bit := '0';
SIGNAL vdd   : bit := '1';
SIGNAL vss   : bit := '0';
SIGNAL inp     : bit_vector (3 downto 0);
SIGNAL daytime    : bit;
SIGNAL reset : bit ;
SIGNAL door   : bit := '0';
SIGNAL alarm   : bit := '0';
SIGNAL door1   : bit := '0';
SIGNAL alarm1   : bit := '0';

--constant clk_period : time := 50 ns;

BEGIN
```

```
dut: dac PORT MAP (ck, vdd, vss, inp, daytime, reset, alarm,
door);

dut1: daca_b_l PORT MAP (ck, vdd, vss, inp, daytime, reset,
alarm1, door1);


 clk_process :process

   begin

       ck <= '1';

       wait for 25 ns;

       ck <= '0';

       wait for 25 ns;

   end process;


p1: process


begin



-- test 0 ( i use the reset button )

reset<='1';
```

```
wait for 100 ns ;
Assert (door=door1) and (alarm=alarm1)
report "ERROR in test 0 at state 0"
severity error;

--test 1 (we use 'O' at the begin )
reset<='0';
daytime<='1';
inp<="1101";
wait for 100 ns ;
Assert (door=door1) and (alarm=alarm1)
report "ERROR in test 1 at state 0"
severity error;

--test 2 (a wrong input )

--correct input '2':

inp<="0010";
wait for 100 ns ;
```

```
Assert (door=door1) and (alarm=alarm1)
report "ERROR in test 2 at state 0"
severity error;



inp<="0000";
wait for 100 ns ;
Assert (door=door1) and (alarm=alarm1)
report "ERROR in test 2 at state 1"
severity error;

--test 3 (the daytime at morining and the normal sequence
"26A05" )

--correct input '2' :
reset<='0';
daytime<='1';
inp<="0010";
wait for 100 ns ;
Assert (door=door1) and (alarm=alarm1)
```

```
report "ERROR in test 3 at state 0" severity error;
--correct input '6':

inp<="0110";
wait for 100 ns ;
Assert (door=door1) and (alarm=alarm1)
report "ERROR in test 3 at state 1" severity error;



--correct input'A':

inp<="1010";
wait for 100 ns ;
Assert (door=door1) and (alarm=alarm1)
report "ERROR in test 3 at state 2" severity error;



--correct input'0':

inp<="0000";
```

```
wait for 100 ns ;
Assert (door=door1) and (alarm=alarm1)
report "ERROR in test 3 at state 3 " severity error;



--correct input '5':


inp<="0101";
wait for 100 ns ;
Assert (door=door1) and (alarm=alarm1)
report "ERROR in test 3 at state 4 " severity error;



--test 4 (we use button 'O' at night)


daytime<='0';
inp<="1101";
wait for 100 ns ;
Assert (door=door1) and (alarm=alarm1)
report "ERROR in test 4 at state 1" severity error;
```

```
--test 5 (the daytime at night and the normal sequence "26A05"
)

--correct input '2' :

daytime<='0';
inp<="0010";
wait for 100 ns ;
Assert (door=door1) and (alarm=alarm1)
report "ERROR in test 5 at state 0" severity error;


--correct input '6':
inp<="0110";
wait for 100 ns ;
Assert (door=door1) and (alarm=alarm1)
report "ERROR in test 5 at state 1" severity error;
```

```
--correct input'A':

inp<="1010";
wait for 100 ns ;
Assert (door=door1) and (alarm=alarm1)
report "ERROR in test 5 at state 2" severity error;



--correct input'0':

inp<="0000";
wait for 100 ns ;
Assert (door=door1) and (alarm=alarm1)
report "ERROR in test 5 at state 3 " severity error;



--correct input '5':
```

```
inp<="0101";
wait for 100 ns ;
Assert (door=door1) and (alarm=alarm1)
report "ERROR in test 5 at state 4 " severity error;



-- test 6 (a wrong input at night )


--correct input '2':
daytime<='0';
inp<="0010";
wait for 100 ns ;
Assert (door=door1) and (alarm=alarm1)
report "ERROR in test 6 at state 0" severity error;



--wrong input '3'


inp<="0011";
wait for 100 ns ;
```

```
Assert (door=door1) and (alarm=alarm1)
report "ERROR in test 6 at state 0" severity error;



WAIT;
END PROCESS p1;
END ARCHITECTURE test_Digital_Access_Control;
```

13- Testbench code for daca_dft :

```
ENTITY testbench_Digital_Access_Control IS
END ENTITY testbench_Digital_Access_Control;


ARCHITECTURE test_Digital_Access_Control OF
testbench_Digital_Access_Control IS
Component dac is
port  (
        ck    : in    bit;
        vdd   : in   bit;
        vss   : in    bit;
        inp     : in  bit_vector (3 downto 0);
        daytime    : in   bit;
        reset : in  bit;
        alarm     : out   bit;
        door    : out      bit
     );
end Component dac;


Component daca_dft is
```

```
  port (
    ck       : in     bit;
    vdd      : in     bit;
    vss      : in     bit;
    inp      : in     bit_vector(3 downto 0);
    daytime  : in     bit;
    reset    : in     bit;
    alarm    : out    bit;
    door     : out    bit;
    scan_in  : in     bit;
    test     : in     bit;
    scan_out : out    bit
 );
end Component daca_dft;

FOR dut: dac USE ENTITY WORK.dac (fsm);

FOR dut1: daca_dft USE ENTITY WORK.daca_dft (structural);

SIGNAL ck    : bit := '0';
SIGNAL vdd   : bit := '1';

SIGNAL vss   : bit := '0';

SIGNAL inp     : bit_vector (3 downto 0);

SIGNAL daytime    : bit;

SIGNAL reset : bit ;

SIGNAL door   : bit := '0';

SIGNAL alarm   : bit := '0';

SIGNAL door1   : bit := '0';

SIGNAL alarm1   : bit := '0';

SIGNAL    scan_in : bit     := '0';

SIGNAL    test  : bit  := '0';

SIGNAL    scan_out: bit   := '0';

SIGNAL    sequence: bit_vector(9 downto 0) := "1001101000";


--constant clk_period : time := 50 ns;


BEGIN


dut: dac PORT MAP (ck, vdd, vss, inp, daytime, reset, alarm,
door);
```

```
dut1: daca_dft PORT MAP (ck, vdd, vss, inp, daytime, reset,
alarm1, door1,scan_in,test,scan_out);

 clk_process :process
   begin
        ck <= '1';
        wait for 25 ns;
        ck <= '0';
        wait for 25 ns;
   end process;

p1: process

begin



-- test 0 ( i use the reset button )
reset<='1';
wait for 100 ns ;
Assert (door=door1) and (alarm=alarm1)
```

```
report "ERROR in test 0 at state 0"
severity error;

--test 1 (we use 'O' at the begin )
reset<='0';
daytime<='1';
inp<="1101";
wait for 100 ns ;
Assert (door=door1) and (alarm=alarm1)
report "ERROR in test 1 at state 0"
severity error;

--test 2 (a wrong input )

--correct input '2':

inp<="0010";
wait for 100 ns ;
Assert (door=door1) and (alarm=alarm1)
report "ERROR in test 2 at state 0"
```

```
severity error;



inp<="0000";
wait for 100 ns ;
Assert (door=door1) and (alarm=alarm1)
report "ERROR in test 2 at state 1"
severity error;

--test 3 (the daytime at morining and the normal sequence
"26A05" )

--correct input '2' :
reset<='0';
daytime<='1';
inp<="0010";
wait for 100 ns ;
Assert (door=door1) and (alarm=alarm1)
report "ERROR in test 3 at state 0" severity error;
--correct input '6':
```

```
inp<="0110";
wait for 100 ns ;
Assert (door=door1) and (alarm=alarm1)
report "ERROR in test 3 at state 1" severity error;


--correct input'A':

inp<="1010";
wait for 100 ns ;
Assert (door=door1) and (alarm=alarm1)
report "ERROR in test 3 at state 2" severity error;


--correct input'0':

inp<="0000";
wait for 100 ns ;
Assert (door=door1) and (alarm=alarm1)
```

```
report "ERROR in test 3 at state 3 " severity error;



--correct input '5':


inp<="0101";
wait for 100 ns ;
Assert (door=door1) and (alarm=alarm1)
report "ERROR in test 3 at state 4 " severity error;



--test 4 (we use button 'O' at night)


daytime<='0';
inp<="1101";
wait for 100 ns ;
Assert (door=door1) and (alarm=alarm1)
report "ERROR in test 4 at state 1" severity error;
```

```
--test 5 (the daytime at night and the normal sequence "26A05"
)

--correct input '2' :

daytime<='0';
inp<="0010";
wait for 100 ns ;
Assert (door=door1) and (alarm=alarm1)
report "ERROR in test 5 at state 0" severity error;



--correct input '6':
inp<="0110";
wait for 100 ns ;
Assert (door=door1) and (alarm=alarm1)
report "ERROR in test 5 at state 1" severity error;
```

```
--correct input'A':

inp<="1010";
wait for 100 ns ;
Assert (door=door1) and (alarm=alarm1)
report "ERROR in test 5 at state 2" severity error;



--correct input'0':

inp<="0000";
wait for 100 ns ;
Assert (door=door1) and (alarm=alarm1)
report "ERROR in test 5 at state 3 " severity error;



--correct input '5':

inp<="0101";
wait for 100 ns ;
```

```
Assert (door=door1) and (alarm=alarm1)
report "ERROR in test 5 at state 4 " severity error;



-- test 6 (a wrong input at night )

--correct input '2':
daytime<='0';
inp<="0010";
wait for 100 ns ;
Assert (door=door1) and (alarm=alarm1)
report "ERROR in test 6 at state 0" severity error;



--wrong input '3'

inp<="0011";
wait for 100 ns ;
Assert (door=door1) and (alarm=alarm1)
report "ERROR in test 6 at state 0" severity error;
```

```
test <= '1';
        for  i In 0 to  sequence'length -1 loop
            scan_in <= sequence(i);  -- Assign values to
circuit inputs
             -- Wait to "propagate" values
            -- Check output against expected result.
            if  i >= 3 then -- 3 registers in the scan chain
                Assert  scan_out = sequence(i-3)
                Report " scanout does not follow scan in"
                Severity error;
            end if;
            wait for  100 ns ;
        end loop;

WAIT;
END PROCESS p1;
END ARCHITECTURE test_Digital_Access_Control;
```