DIGITAL ACCESS CONTROL

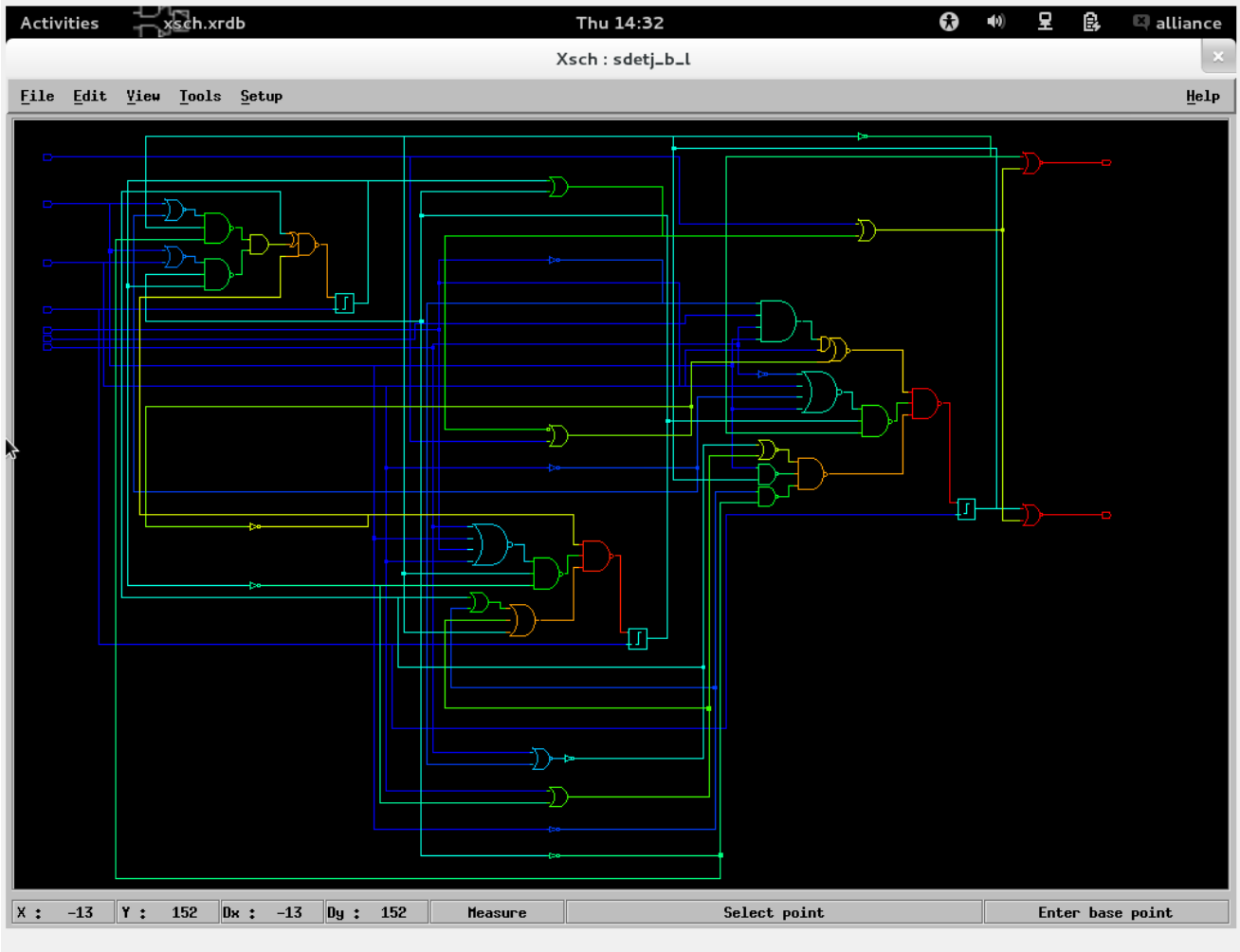
COURSE: CSE 215

NAME: KARIM WALID ELHAMMADY
ID: 16P3090
PROGRAM: CESS
EMAIL: KARIM.ELHAMMADY629@GMAIL.COM

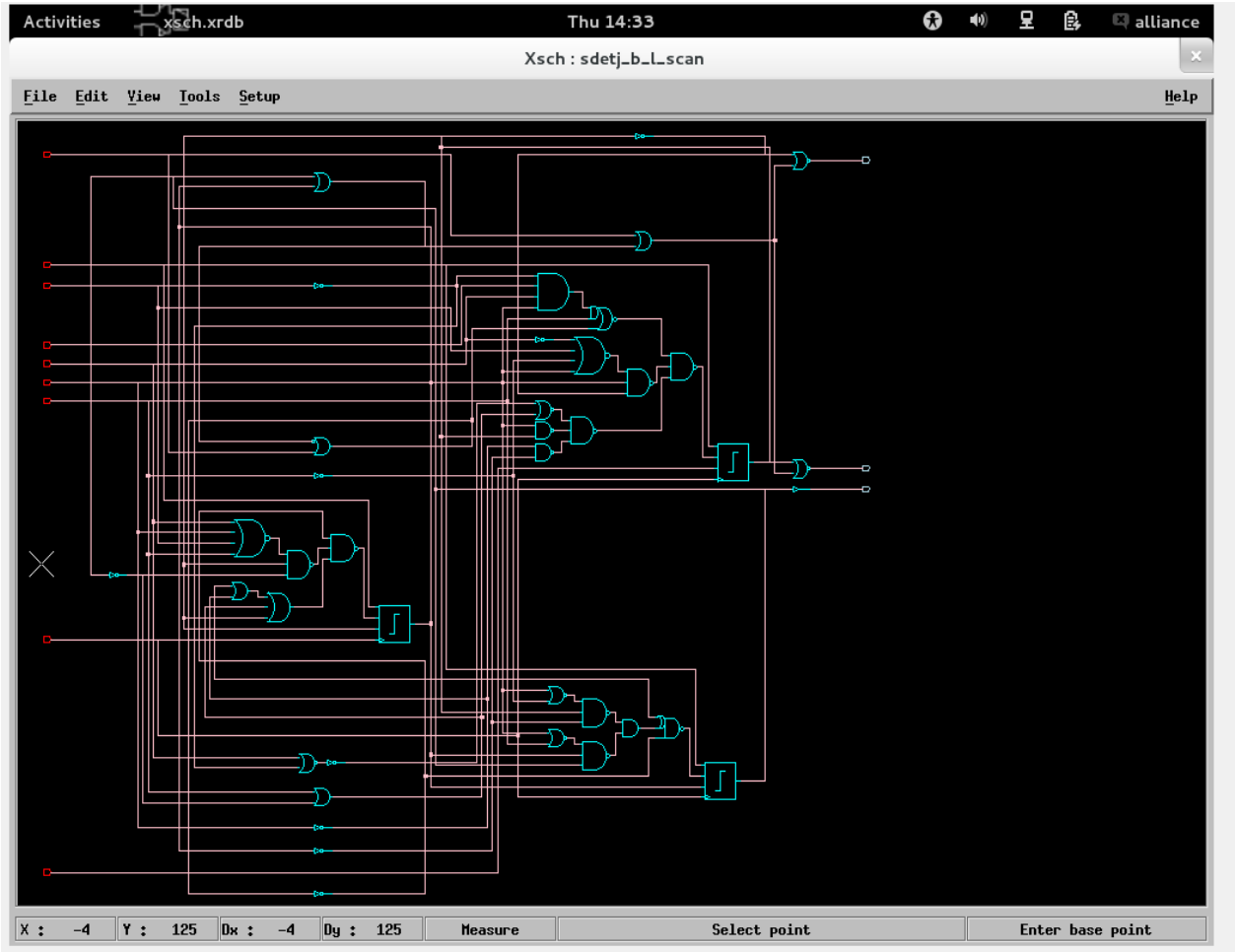
1) Introduction

The purpose of this document is to explain the steps followed to perform logic synthesis of the RTL design implemented in phase 1; it was made using alliance tools ; syf, boom, boog, loon, xsch, flatbeh, proof and scapin, and Symphony for Simulation.

2) XSCH_LOON:


Activities
xsch.xrdb
Thu 14:32
alliance
Xsch : sdetj_b_l
File Edit View Tools Setup
Help
X : -13
Y : 152
Dx : -13
Dy : 152
Measure
Select point
Enter base point

3) XSCH_SCAPIN:



4) Delay and Area (boog):

enc	Area (lamda²)	delay (ps)
a	1396	57750
j	1597	53750
m	1729	56750
o	2171	88250
r	1932	64750

5) Delay and Area (loon):

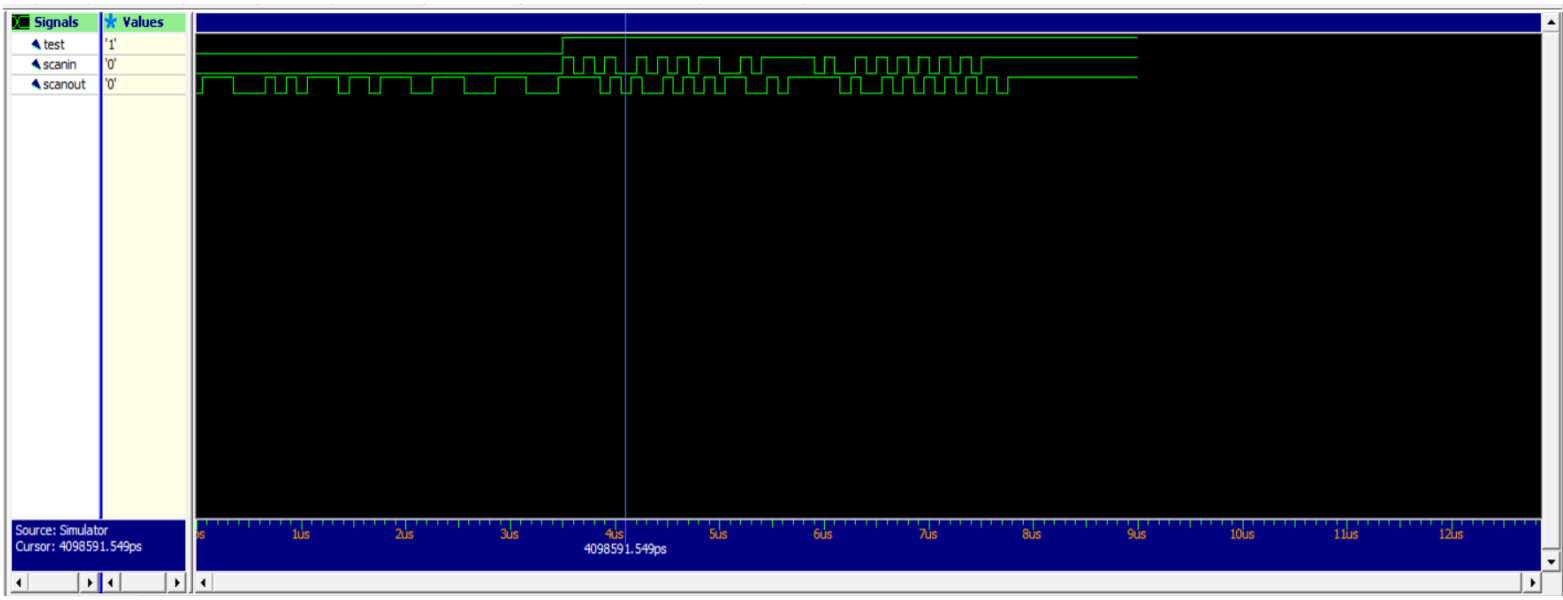
enc	area (lamda²)	delay (ps)
a	2148	58750
j	1832	53750
m	2485	59000
o	2253	91250
r	2335	64750

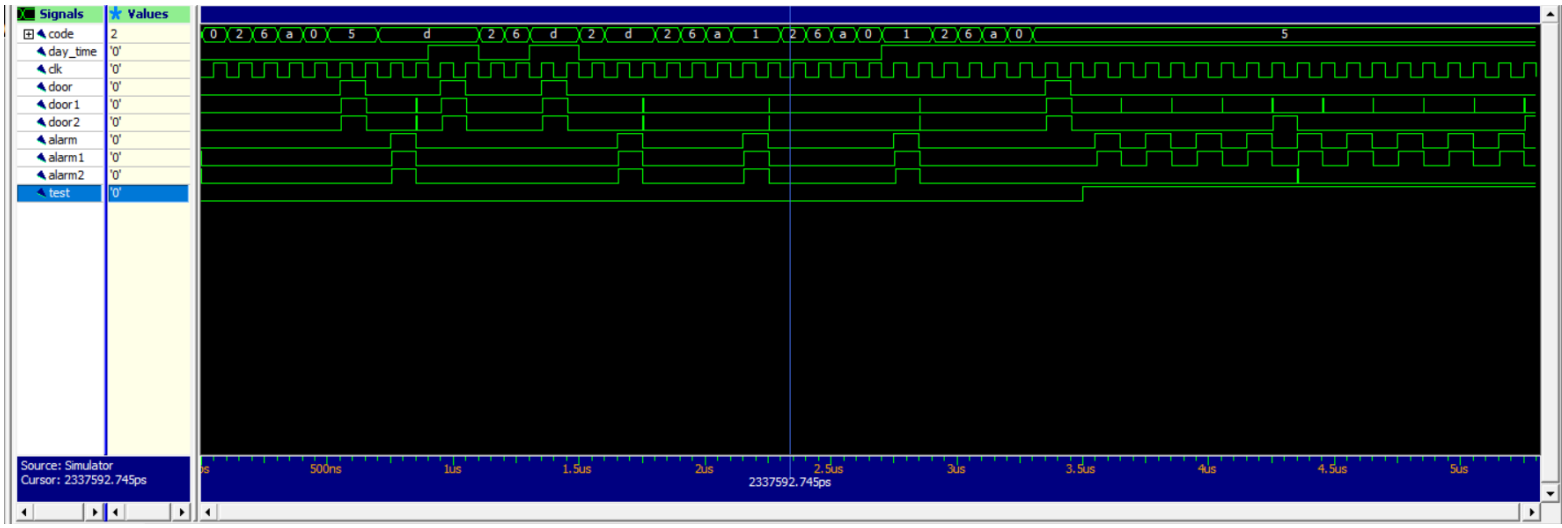
6) Literal:

enc	literal no
a	63
j	56
m	59
o	82
r	68

J encoding was chosen as it has the smallest area and delay time of all encodings.

7) Simulation





8) Make file:

```
#-------Sdet-----------------------------------------------#

all: sdeta.vbe \
     sdetj.vbe \
     sdetm.vbe \
     sdeto.vbe \
     sdetr.vbe
         @echo "<-- Generated"

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

vhd_to_fsm:
    rename .vhd .fsm *.vhd

sdeta.vbe: sdet.fsm
    @echo "    Encoding Synthesis -> sdeta.vbe"
    syf -CEV -a sdet

sdetj.vbe: sdet.fsm
    @echo "    Encoding Synthesis  -> sdetj.vbe"
    syf -CEV -j sdet

sdetm.vbe: sdet.fsm
    @echo "    Encoding Synthesis  -> sdetm.vbe"
    syf -CEV -m sdet

sdeto.vbe: sdet.fsm
    @echo "    Encoding Synthesis  -> sdeto.vbe"
    syf -CEV -o sdet

sdetr.vbe: sdet.fsm
    @echo "    Encoding Synthesis  -> sdetr.vbe"
    syf -CEV -r sdet

#-------Clean Up---------------------------------------#

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

#-------BOOM-------------------------------------------#
```

```
sdet_boom: sdeta_b.vbe sdetj_b.vbe sdetm_b.vbe sdeto_b.vbe sdetr_b.vbe

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

#-------BOOG-------------------------------------------#

sdet_boog: sdeta_b.vst sdetj_b.vst sdetm_b.vst sdeto_b.vst sdetr_b.vst

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

#-------LOON-------------------------------------------#

sdet_loon: sdeta_b_l.vst sdetj_b_l.vst sdetm_b_l.vst sdeto_b_l.vst sdetr_b_l.vst

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

#-------Flatbeh&Proof----------------------------------#

%_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

#-------scapin-----------------------------------------#

ac_scapin_registers:
    cat sdetj_b_l.vst | grep sff

%_scan.vst: %.vst scan.path
    @echo "    scan-path insertion  -> $@"
    scapin -VRB $* scan $*_scan > scapin.out
```

9) Parmfile:

```
#M{2}

#L{2}

#C{
door:100;
alarm:100;
}
```

10) Pathfile:

```
BEGIN_PATH_REG

fsm_alarm_cs_0_ins
fsm_alarm_cs_1_ins
fsm_alarm_cs_2_ins

END_PATH_REG

BEGIN_CONNECTOR

SCAN_IN scanin
SCAN_OUT scanout
SCAN_TEST test

END_CONNECTOR
```

11) Testbench:

```
ENTITY testbench IS
END ENTITY testbench;

ARCHITECTURE tb OF testbench IS
COMPONENT FSM_alarm IS port
(code: in bit_vector(3 downto 0);
reset: in bit;
day_time: in bit;
vdd: in bit;
vss: in bit;
clk: in bit;
door: out bit;
alarm: out bit);
end component;
COMPONENT FSM_alarm1 IS port
(code: in bit_vector(3 downto 0);
reset: in bit;
day_time: in bit;
vdd: in bit;
vss: in bit;
clk: in bit;
door: out bit;
alarm: out bit);
end component;

COMPONENT FSM_alarm2 IS port
(  code     : in      bit_vector(3 downto 0);
reset    : in      bit;
day_time : in      bit;
vdd      : in      bit;
vss      : in      bit;
clk      : in      bit;
door     : out     bit;
alarm    : out     bit;
scanin   : in      bit;
test     : in      bit;
scanout  : out     bit);
end component;


FOR test0: FSM_alarm USE ENTITY work.FSM_alarm (behavioral_description);
FOR test1: FSM_alarm1 USE ENTITY work.sdetj_b_1 (structural);
```

```
FOR test2: FSM_alarm2 USE ENTITY work.sdetj_b_l_scan (structural);

SIGNAL code : bit_vector (3 downto 0) := "0000";
SIGNAL reset : bit := '0';
SIGNAL day_time : bit := '0';
SIGNAL vdd : bit := '1';
SIGNAL vss : bit := '0';
SIGNAL clk : bit := '0';
SIGNAL scanin : bit ;
SIGNAL test : bit := '0';
SIGNAL scanout : bit ;
SIGNAL door : bit ;
SIGNAL alarm : bit ;
SIGNAL door1 : bit ;
SIGNAL alarm1 : bit ;
SIGNAL door2 : bit ;
SIGNAL alarm2 : bit ;
constant sequence : bit_vector := "10101001010101100101111101001010101010101";

begin
test0: FSM_alarm PORT MAP (code,reset,day_time,vdd,vss,clk,door,alarm);
test1: FSM_alarm1 PORT MAP (code,reset,day_time,vdd,vss,clk,door1,alarm1);
test2: FSM_alarm2 PORT MAP
(code,reset,day_time,vdd,vss,clk,door2,alarm2,scanin,test,scanout);


clk_gen: process is begin
wait for 50 ns;
if clk = '0' then
clk <= '1';
else
clk <= '0';
end if;
end process;
testing: process is begin
reset <= '1';
wait for 100 ns;
reset <= '0';
code <="0010";
day_time<='0';
wait for 100 ns;
assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2
report "wrong answer"
severity error;
```

```
code <="0110";
day_time<='0';
wait for 100 ns;
assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2
report "wrong answer"
severity error;

code <="1010";
day_time<='0';
wait for 100 ns;
assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2
report "wrong answer"
severity error;

code <="0000";
day_time<='0';
wait for 100 ns;
assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2
report "wrong answer"
severity error;

code <="0101";
day_time<='0';
wait for 100 ns;
assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2
report "wrong answer"
severity error;

wait for 100 ns;
--test set 1

code <="1101";
day_time<='0';
wait for 100 ns;
assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2
report "wrong answer"
severity error ;
wait for 100 ns;

code <="1101";
day_time<='1';
wait for 100 ns;
assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2
report "wrong answer"
severity error;
```

```
wait for 100 ns ;
--test set 2

code <="0010";
day_time<='0';
wait for 100 ns;
code <="0110";
day_time<='0';
wait for 100 ns;
code <="1101";
day_time<='1';
wait for 100 ns;
assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2
report "wrong answer"
severity error;
wait for 100 ns;
--test set 3

code <="0010";
day_time<='0';
wait for 100 ns;
code <="1101";
day_time<='0';
wait for 100 ns;
assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2
report "wrong answer"
severity error;
wait for 100 ns;
--test set 4

code <="0010";
day_time<='0';
wait for 100 ns;
code <="0110";
day_time<='0';
wait for 100 ns;
code <="1010";
day_time<='0';
wait for 100 ns;
code <="0001";
day_time<='0';
wait for 100 ns;
assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2
report "wrong answer"
```

```
severity error;
wait for 100 ns;
--test set 5
code <="0010";
day_time<='0';
wait for 100 ns;
code <="0110";
day_time<='0';
wait for 100 ns;
code <="1010";
day_time<='0';
wait for 100 ns;
code <="0000";
day_time<='0';
wait for 100 ns;
code <="0001";
day_time<='1';
wait for 100 ns;
assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2
report "wrong answer"
severity error;
wait for 100 ns;
--test set 6
code <="0010";
day_time<='1';
wait for 100 ns;
assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2
report "wrong answer"
severity error;

code <="0110";
day_time<='1';
wait for 100 ns;
assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2
report "wrong answer"
severity error;

code <="1010";
day_time<='1';
wait for 100 ns;
assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2
report "wrong answer"
severity error;

code <="0000";
```

```
day_time<='1';
wait for 100 ns;
assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2
report "wrong answer"
severity error;

code <="0101";
day_time<='1';
wait for 100 ns;
assert alarm=alarm1 and door=door1 and alarm =alarm2 and door=door2
report "wrong answer"
severity error;

wait for 100 ns;
--test case 7

test <= '1' ;

for i in 0 to sequence'length-1 loop
    scanin <= sequence(i);
    wait for 100 ns;
    if i>=2 then
        assert scanout = sequence (i-2)
        report "scanout does not follow scan in"
        severity error ;
    end if;
    end loop;
    wait;

end process;
end architecture tb;
```

12) Flatbeh:

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

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

a netlist abstractor

Alliance CAD System 5.0 20090901, flatbeh 5.0 [2000/11/01]
Copyright (c) 1993-2019, ASIM/LIP6/UPMC
Author(s): Fran&#65533;ois 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/alliance/cells/rflib:/u
sr/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       = sdetj_b_l.vst
   Output  file       = sdetj_b_l_net.vbe
   ===================================================================

Loading './sdetj_b_l.vst'
flattening figure sdetj_b_l
 loading inv_x4
 loading inv_x2
 loading an12_x1
 loading a3_x2
 loading na2_x1
 loading o2_x2
 loading o3_x2
 loading no4_x1
 loading na3_x1
 loading a2_x2
 loading nao22_x1
 loading sff1_x4
 loading no2_x1
Restoring array's orders
BEH : Saving 'sdetj_b_l_net' in a vhdl file (vbe)
```

13) Proof:

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

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:/u
sr/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             = sdetj.vbe
Second VHDL file            = sdetj_b_l_net.vbe
The auxiliary signals are erased
Errors are displayed
================================================================================
=

Compiling 'sdetj' ...
Compiling 'sdetj_b_l_net' ...
---> final number of nodes = 295(151)

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

        './sdetj'  and  './sdetj_b_l_net'
--------------------------------------------------------------------------------
```

============================== PRIMARY OUTPUT ==============================
============================== AUXILIARY SIGNAL ==============================
============================== REGISTER SIGNAL ==============================
============================== EXTERNAL BUS ==============================
============================== INTERNAL BUS ==============================

Formal Proof : OK

pppppppppppppppppppppppprrrrrrrrrrrroooooooooooooooooooooooooooooffffffffffffff
---