Ain Shams University
Faculty of Engineering
CHEP- CESS

Mohammed Ehab Elsaeed
16P8160
m.elsaeed1998@gmail.com

Electronic Design Automation
CSE 215
Digital Access Control Finite State Machine
Project 1

Introduction

The purpose of this document is to illustrate the usage of Alliance tools to synthesise the digital access control finite state machine created in project one.

Table of Contents

1- Code Modification for Alliance
2- BOOM Outputs Comparison
3- LOON Outputs Comparison
4- LOON Netlist of Chosen State Encoding
5- Delay Simulation After FLATBEH and PROOF.
6- SCAPIN Netlist
7- Simulation of FSM Using Behavioural, LOON's and SCAPIN's structural Components.
8- Test of scan-path
9- Appendix
    1- Makefile
    2- Paramfile
    3- .path file
    4- FLATBEH log
    5- PROOF log
    6- LOON's and SCAPIN's Structural Components and FSM's Behavioural Component Comparison-Based-Testbench

1- Code Modification for Alliance

For Alliance tools to work, I had to make a slight modification to my FSM's VHDL Code.

For Alliance:

```
                when others =>
                    assert ('1')
                    report "Invalid state";
```

For Symphony EDA (ModelSim Equivalent):

```
                when others =>
                    assert false
                    report "Invalid state"
                        severity failure;
```

SYF never accepted the "assert false" statement.

2- BOOM Outputs Comparison

Encoding	A	J	M	O	R
Final Literals	92	95	78	100	90

3- LOON Outputs Comparison

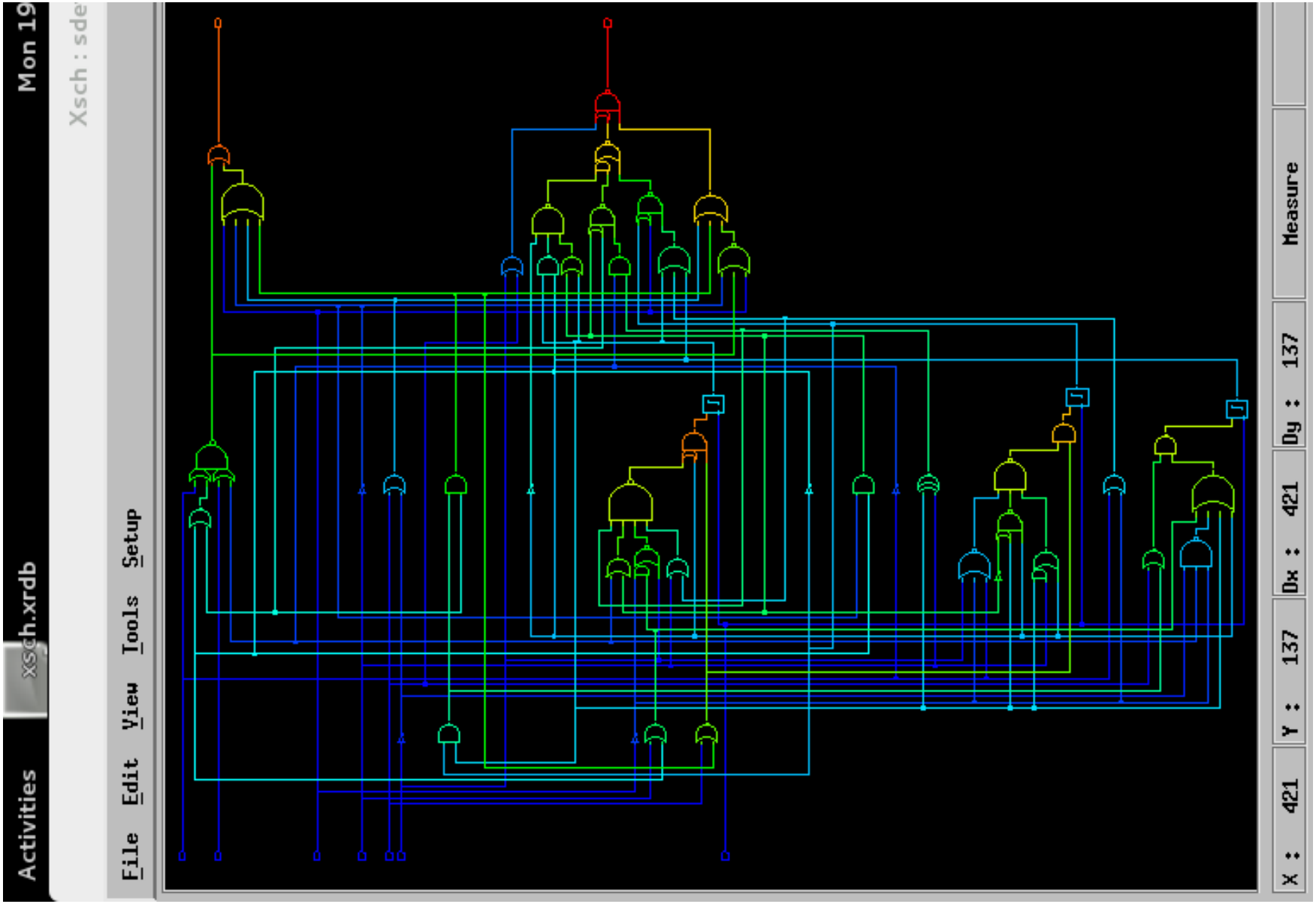
Encoding	A	J	M	O	R
Critical Path Delays	2363	3156	2584	3022	3015
Areas	80750	86250	70250	103000	82000

Decision: Encoding **M** was chosen.

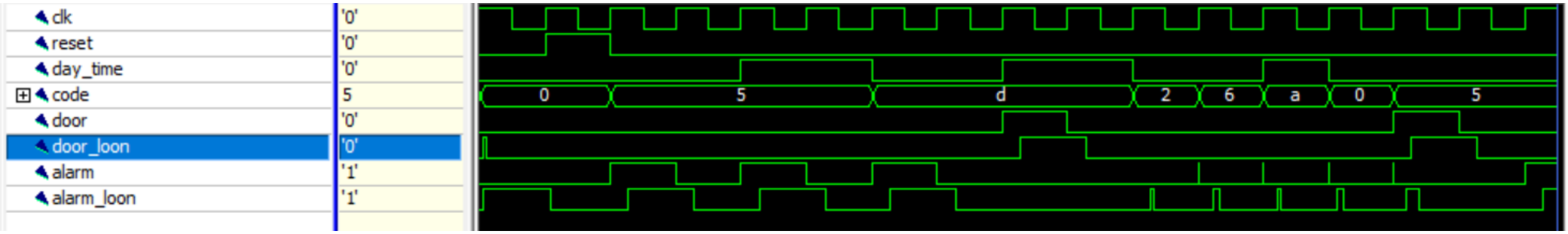
Justification: It has the lowest area and the 2nd lowest delay among other encodings.

Paramfile is in the appendix.

4- LOON Netlist of Chosen Encoding



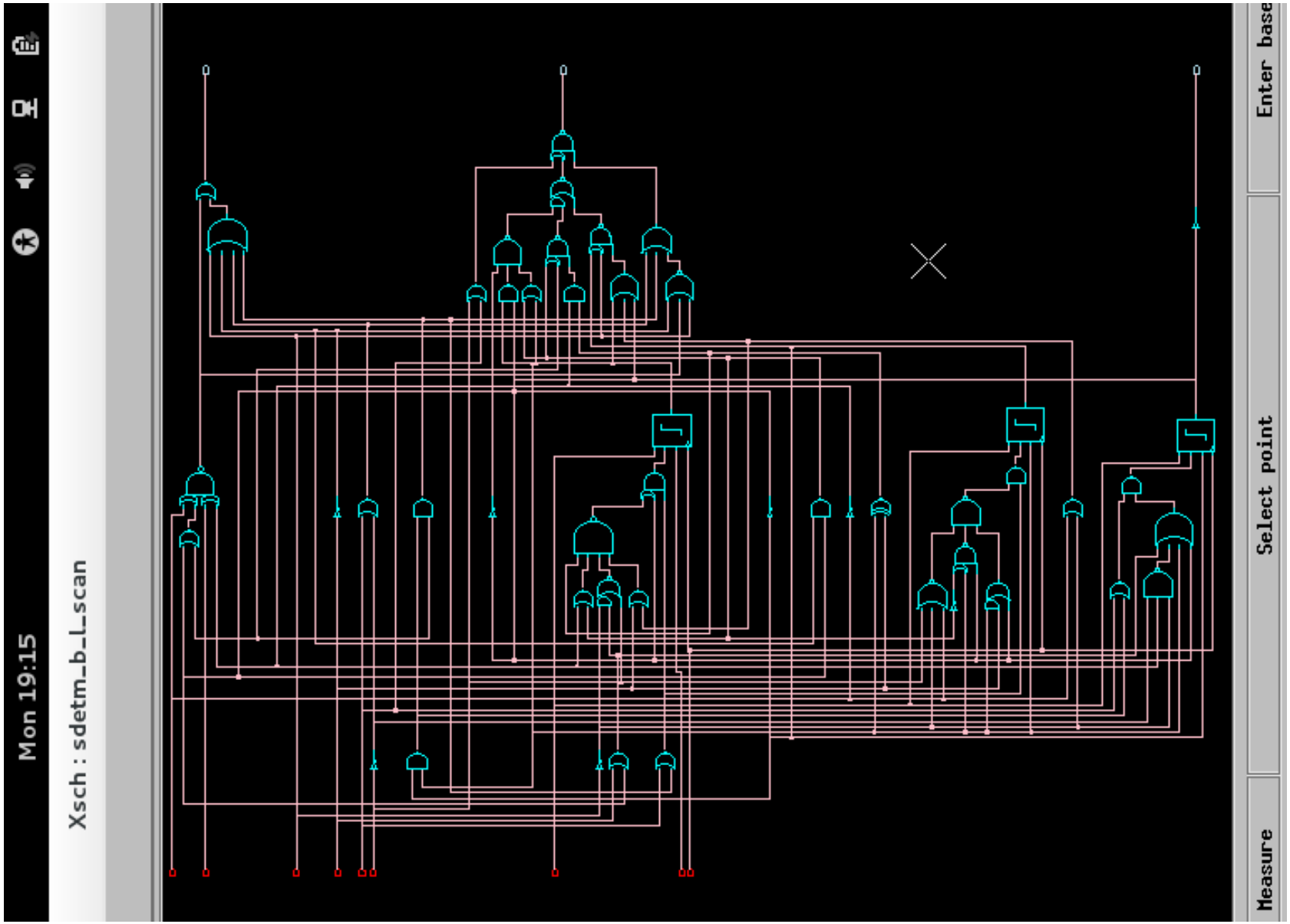
5- Delay Simulation After FLATBEH and PROOF.



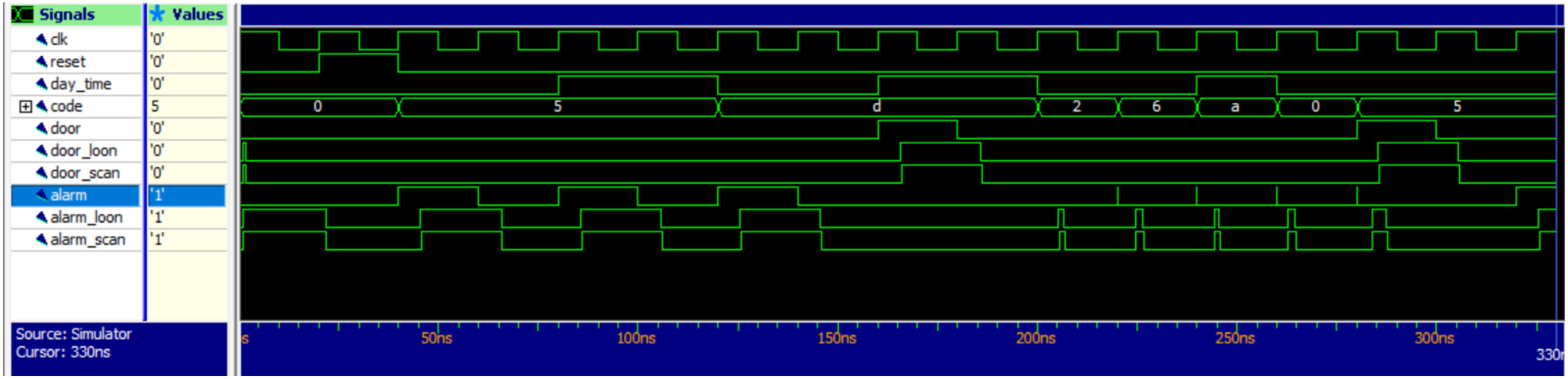
Terminal Output:

```
ASSERT: ERROR at 320 ns+0: FSM PASSED ALL TEST SUCCESSFULLY.
    At Phase 2/Post Alliance Files/all_tb.vhd: (line 274)
        Instance = :all_tb(all_tb_behav):
Simulation stopped at: 330 ns
Elapsed Time: 00h:00m:00s:176ms
%
```

6- SCAPIN Netlist



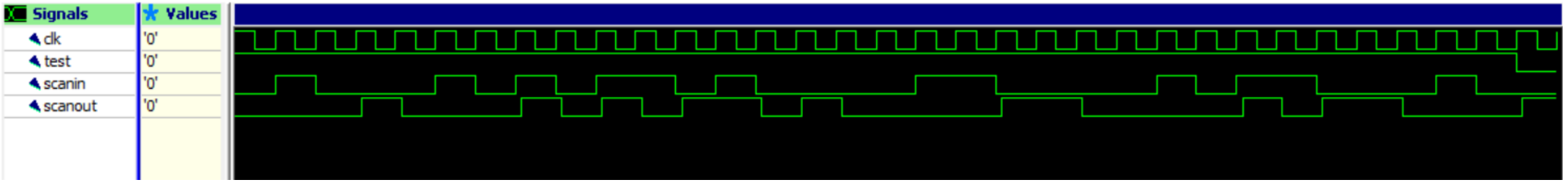
7- Simulation of FSM Using Behavioural, LOON's and SCAPIN's structural Components.




```
ASSERT: ERROR at 320 ns+0: FSM PASSED ALL TEST SUCCESSFULLY.
    At Phase 2/Post Alliance Files/all_tb.vhd: (line 274)
        Instance = :all_tb(all_tb_behav):
Simulation stopped at: 330 ns
Elapsed Time: 00h:00m:00s:270ms
%
```

Comments: The Structural Components produced by SCAPIN and LOON have delays unlike the initial behavioural component. These delays, however, cause no problems as they are within half a clock period.

8- Test of scan-path




```
ASSERT: ERROR at 640 ns+0: SCANPATH PASSED ALL TEST SUCCESSFULLY.
    At Phase 2/Post Alliance Files/all_tb.vhd: (line 90)
        Instance = :all_tb(all_tb_behav):
Simulation stopped at: 660 ns
Elapsed Time: 00h:00m:00s:203ms
%
```

9- Appendix

1- Makefile

```
all_syf: sdeta.vbe \
    sdetj.vbe \
    sdetm.vbe \
    sdeto.vbe \
    sdetr.vbe
        @echo "<-- SYF DONE"

all_boom: sdeta_b.vbe \
    sdetj_b.vbe \
    sdetm_b.vbe \
    sdeto_b.vbe \
    sdetr_b.vbe
        @echo "<-- BOOM DONE"

sdet_boog : sdeta_b.vst \
    sdetj_b.vst \
    sdetm_b.vst \
    sdeto_b.vst \
    sdetr_b.vst
        @echo "<-- BOOG DONE"

sdet_loon : sdeta_b_l.vst \
    sdetj_b_l.vst \
```

```
    sdetm_b_l.vst \
    sdeto_b_l.vst \
    sdetr_b_l.vst
        @echo "<-- LOON DONE"

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

vhd_to_fsm:
    rename .vhd .fsm *.vhd

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

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

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

%_b.vbe: %.vbe
    @echo "     Boolean Optimization -> $@"
```

```
    boom -V -d 50 $* $*_b > $*_boom.out

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 *.out *.vbe *.enc *~
    @echo "Erase all the files generated by the makefile"
```

2- Paramfile

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

3- .Path File

```
BEGIN_PATH_REG
dac_current_state_0_ins
dac_current_state_1_ins
dac_current_state_2_ins
END_PATH_REG

BEGIN_CONNECTOR
SCAN_IN scanin
SCAN_OUT scanout
SCAN_TEST test
END_CONNECTOR
```

4- FLATBEH Log

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

                          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?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:
/usr/lib64/alliance/cells/rf2lib:/usr/lib64/alliance/cells/ramlib:/usr/lib64/alliance/cells/romlib:/u
sr/lib64/alliance/cells/pxlib:/usr/lib64/alliance/cells/padlib
```

```
     MBK_CATAL_NAME      = CATAL
     ============================== Files ==============================
     Netlist file        = sdetm_b_l.vst
     Output  file        = sdetm_b_l_net.vbe
     ===================================================================

Loading './sdetm_b_l.vst'
flattening figure sdetm_b_l
 loading nao2o22_x1
 loading nxr2_x1
 loading inv_x4
 loading na4_x1
 loading ao22_x2
 loading oa22_x2
 loading inv_x2
 loading sff1_x4
 loading no3_x1
 loading o3_x2
 loading o2_x2
 loading na2_x1
 loading na3_x1
 loading a2_x2
 loading noa22_x1
 loading nao22_x1
 loading o4_x2
 loading no2_x1
Restoring array's orders
BEH : Saving 'sdetm_b_l_net' in a vhdl file (vbe)
```

5- PROOF LOG

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

                              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/l
ib64/alliance/cells/pxlib:/usr/lib64/alliance/cells/padlib
======================  Files, Options and Parameters  ======================
```

```
First VHDL file   = sdetm.vbe
Second VHDL file  = sdetm_b_l_net.vbe
The auxiliary signals are erased
Errors are displayed
================================================================================

Compiling 'sdetm' ...
Compiling 'sdetm_b_l_net' ...
---> final number of nodes = 343(127)

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

             './sdetm'  and  './sdetm_b_l_net'
--------------------------------------------------------------------------------
==============================  PRIMARY OUTPUT  ==============================
============================  AUXILIARY SIGNAL  ==============================
============================  REGISTER SIGNAL  ===============================
============================  EXTERNAL BUS ===================================
============================ INTERNAL BUS ====================================

                           Formal Proof : OK

pppppppppppppppppppppppprrrrrrrrrrrroooooooooooooooooooooooooooffffffffffffff
--------------------------------------------------------------------------------
```

6- LOON's and SCAPIN's Structural Components and FSM's Behavioural Component Comparison-Based-Testbench

```
entity all_tb is
    end all_tb;

    architecture all_tb_behav of all_tb is
        component dac is
            port (
                reset    : in      bit;
                day_time : in      bit;
                code     : in      bit_vector(3 downto 0);
                door     : out     bit;
                alarm    : out     bit;
                clk      : in      bit;
                vdd      : in      bit;
                vss      : in      bit
            );
        end component dac;
        component dac_loon is
            port (
                reset    : in      bit;
                day_time : in      bit;
                code     : in      bit_vector(3 downto 0);
                door     : out     bit;
                alarm    : out     bit;
                clk      : in      bit;
                vdd      : in      bit;
                vss      : in      bit
```

```
    );
end component dac_loon;
component dac_scan is
    port (
        reset    : in      bit;
        day_time : in      bit;
        code     : in      bit_vector(3 downto 0);
        door     : out     bit;
        alarm    : out     bit;
        clk      : in      bit;
        vdd      : in      bit;
        vss      : in      bit;
        scanin   : in      bit;
        test     : in      bit;
        scanout  : out     bit
    );
end component dac_scan;


signal reset        : bit;
signal day_time     : bit;
signal code         : bit_vector(3 downto 0);
signal door         : bit;
signal alarm        : bit;
signal door_loon    : bit;
signal alarm_loon   : bit;
signal door_scan    : bit;
signal alarm_scan   : bit;
signal clk          : bit;
```

```
        signal vdd          : bit := '1';
        signal vss          : bit := '0';
        signal scanin       : bit := '0';
        signal test         : bit := '0';
        signal scanout      : bit := '0';

        for all             : dac      use entity work.dac(dac_behav);
        for all             : dac_loon use entity work.sdetm_b_l(structural);
        for all             : dac_scan use entity work.sdetm_b_l_scan(structural);

        constant clk_period : time                   :=  20 ns;
        constant a          : bit_vector(3 downto 0) := "1010";
        constant b          : bit_vector(3 downto 0) := "1011";
        constant o          : bit_vector(3 downto 0) := "1101";
        constant scantest   : bit_vector             := "01000101011010000110000101100010";
    begin

        dut      : dac      port map(reset, day_time, code, door, alarm, clk, vdd, vss);
        dut_loon : dac_loon port map(reset, day_time, code, door_loon, alarm_loon, clk, vdd, vss);
        dut_scan : dac_scan port map(reset, day_time, code, door_scan, alarm_scan, clk, vdd, vss,
scanin, test, scanout);

        process begin

            test <= '1';
            for i In 0 to scantest'length-1 loop
                scanin <= scantest(i);
                wait for clk_period;
                if i>=2 then
```

```
                Assert scanout=scantest(i-2)
                Report "scanout does not follow scan in"
                Severity error;
            end if;
        end loop;

        assert false
        report "SCANPATH PASSED ALL TEST SUCCESSFULLY."
            severity error;

        test<= '0';
        wait for clk_period;

        -- 1
        reset <= '1';
        -- day_time<='0';
        -- code <= x"";
        wait for clk_period;
        assert door = door_loon and alarm = alarm_loon
        report "door_loon != door OR alarm_loon != alarm. Setting reset = 1 should reset the
circuit to its initial state"
            severity failure;

        assert door = door_scan and alarm = alarm_scan
        report "door_scan != door OR alarm_scan != alarm. Setting reset = 1 should reset the
circuit to its initial state"
            severity failure;

        -- 2
```

```
            reset    <= '0';
            day_time <= '0';
            code     <= x"5";
            wait for clk_period;
            assert door = door_loon and alarm = alarm_loon
            report "door_loon != door OR alarm_loon != alarm. Entering a wrong code, should trigger the
alarm"
                severity failure;

            assert door = door_scan and alarm = alarm_scan
            report "door_scan != door OR alarm_scan != alarm. Entering a wrong code, should trigger the
alarm"
                severity failure;

            -- 3
            wait for clk_period;
            assert door = door_loon and alarm = alarm_loon
            report "door_loon != door OR alarm_loon != alarm. Waiting for a clock cycle, should reset
both door and alarm, to zero"
                severity failure;

            assert door = door_scan and alarm = alarm_scan
            report "door_scan != door OR alarm_scan != alarm. Waiting for a clock cycle, should reset
both door and alarm, to zero"
                severity failure;

            -- 4
            reset    <= '0';
            day_time <= '1';
```

```
            code     <= x"5";
            wait for clk_period;
            assert door = door_loon and alarm = alarm_loon
            report "door_loon != door OR alarm_loon != alarm. Entering a wrong code, even during
daytime should trigger the alarm"
                severity failure;

            assert door = door_scan and alarm = alarm_scan
            report "door_scan != door OR alarm_scan != alarm. Entering a wrong code, even during
daytime should trigger the alarm"
                severity failure;

            -- 5
            wait for clk_period;
            assert door = door_loon and alarm = alarm_loon
            report "door_loon != door OR alarm_loon != alarm. Waiting for a clock cycle, should reset
both door and alarm, to zero"
                severity failure;

            assert door = door_scan and alarm = alarm_scan
            report "door_scan != door OR alarm_scan != alarm. Waiting for a clock cycle, should reset
both door and alarm, to zero"
                severity failure;

            -- 6
            reset    <= '0';
            day_time <= '0';
            code     <= o;
            wait for clk_period;
```

```
            assert door = door_loon and alarm = alarm_loon
            report "door_loon != door OR alarm_loon != alarm. Entering 'O' during night time should
trigger the alarm"
                severity failure;

            assert door = door_scan and alarm = alarm_scan
            report "door_scan != door OR alarm_scan != alarm. Entering 'O' during night time should
trigger the alarm"
                severity failure;

            -- 7
            wait for clk_period;
            assert door = door_loon and alarm = alarm_loon
            report "door_loon != door OR alarm_loon != alarm. Waiting for a clock cycle, should reset
both door and alarm, to zero"
                severity failure;

            assert door = door_scan and alarm = alarm_scan
            report "door_scan != door OR alarm_scan != alarm. Waiting for a clock cycle, should reset
both door and alarm, to zero"
                severity failure;

            -- 8
            reset    <= '0';
            day_time <= '1';
            code     <= o;
            wait for clk_period;
            assert door = door_loon and alarm = alarm_loon
```

```
            report "door_loon != door OR alarm_loon != alarm. Entering 'O' during daytime should open
the door"
                severity failure;

            assert door = door_scan and alarm = alarm_scan
            report "door_scan != door OR alarm_scan != alarm. Entering 'O' during daytime should open
the door"
                severity failure;

            -- 9
            wait for clk_period;
            assert door = door_loon and alarm = alarm_loon
            report "door_loon != door OR alarm_loon != alarm. Waiting for a clock cycle, should reset
both door and alarm, to zero"
                severity failure;

            assert door = door_scan and alarm = alarm_scan
            report "door_scan != door OR alarm_scan != alarm. Waiting for a clock cycle, should reset
both door and alarm, to zero"
                severity failure;

            -- 10
            reset    <= '0';
            day_time <= '0';
            code     <= x"2";
            wait for clk_period;
            assert door = door_loon and alarm = alarm_loon
            report "door_loon != door OR alarm_loon != alarm. Entering 2 neither opens the door, nor
triggers the alarm"
```

```
                severity failure;

            assert door = door_scan and alarm = alarm_scan
            report "door_scan != door OR alarm_scan != alarm. Entering 2 neither opens the door, nor
triggers the alarm"
                severity failure;

            -- 11
            reset    <= '0';
            day_time <= '0';
            code     <= x"6";
            wait for clk_period;
            assert door = door_loon and alarm = alarm_loon
            report "door_loon != door OR alarm_loon != alarm. Entering 2, 6 neither opens the door, nor
triggers the alarm"
                severity failure;

            assert door = door_scan and alarm = alarm_scan
            report "door_scan != door OR alarm_scan != alarm. Entering 2, 6 neither opens the door, nor
triggers the alarm"
                severity failure;

            -- 12
            reset    <= '0';
            day_time <= '1';
            code     <= a;
            wait for clk_period;
            assert door = door_loon and alarm = alarm_loon
```

```
            report "door_loon != door OR alarm_loon != alarm. Entering 2, 6, A neither opens the door,
nor triggers the alarm, switching daytime to 1 and not pressing 'O' , will not open the door nor
trigger the alarm"
                severity failure;

            assert door = door_scan and alarm = alarm_scan
            report "door_scan != door OR alarm_scan != alarm. Entering 2, 6, A neither opens the door,
nor triggers the alarm, switching daytime to 1 and not pressing 'O' , will not open the door nor
trigger the alarm"
                severity failure;

            -- 13
            reset    <= '0';
            day_time <= '0';
            code     <= x"0";
            wait for clk_period;
            assert door = door_loon and alarm = alarm_loon
            report "door_loon != door OR alarm_loon != alarm. Entering 2, 6, A, 0 neither opens the
door, nor triggers the alarm"
                severity failure;

            assert door = door_scan and alarm = alarm_scan
            report "door_scan != door OR alarm_scan != alarm. Entering 2, 6, A, 0 neither opens the
door, nor triggers the alarm"
                severity failure;

            -- 14
            reset    <= '0';
            day_time <= '0';
```

```
            code     <= x"5";
            wait for clk_period;
            assert door = door_loon and alarm = alarm_loon
            report "door_loon != door OR alarm_loon != alarm. Entering 2, 6, A, 0, 5 opens the door,
but doesn't trigger the alarm"
                severity failure;

            assert door = door_scan and alarm = alarm_scan
            report "door_scan != door OR alarm_scan != alarm. Entering 2, 6, A, 0, 5 opens the door,
but doesn't trigger the alarm"
                severity failure;

            -- 15
            wait for clk_period;
            assert door = door_loon and alarm = alarm_loon
            report "door_loon != door OR alarm_loon != alarm. Waiting for a clock cycle, should reset
both door and alarm, to zero"
                severity failure;

            assert door = door_scan and alarm = alarm_scan
            report "door_scan != door OR alarm_scan != alarm. Waiting for a clock cycle, should reset
both door and alarm, to zero"
                severity failure;


            assert false
            report "FSM PASSED ALL TEST SUCCESSFULLY."
                severity error;
            wait;
```

```
    end process;

    process begin
        clk <= '1', '0' after (clk_period/2);
        wait for clk_period;
    end process;

end architecture all_tb_behav;
```