CSE215: EDA

Project

Phase (2)

Prepared by: Yara Hossam Mohamed

Email: youra_97@hotmail.com

Department: Computer Engineering

(CESS)

Group:2

ID: 16P3002

Introduction

In this project I am performing logic synthesis of the RTL design of phase 1. Based on phase 1 results starting from the verified FSM, I synthesize the corresponding gate using Alliance Fedora virtual machine and modelSim for simulation of the wave and testbenches.

Boom Output:

	-a	-j	-m	-o	-r
Number of litterals before optimization	161	163	163	154	168
Number of litterals after optimization	69	73	64	80	75

Loon Output:

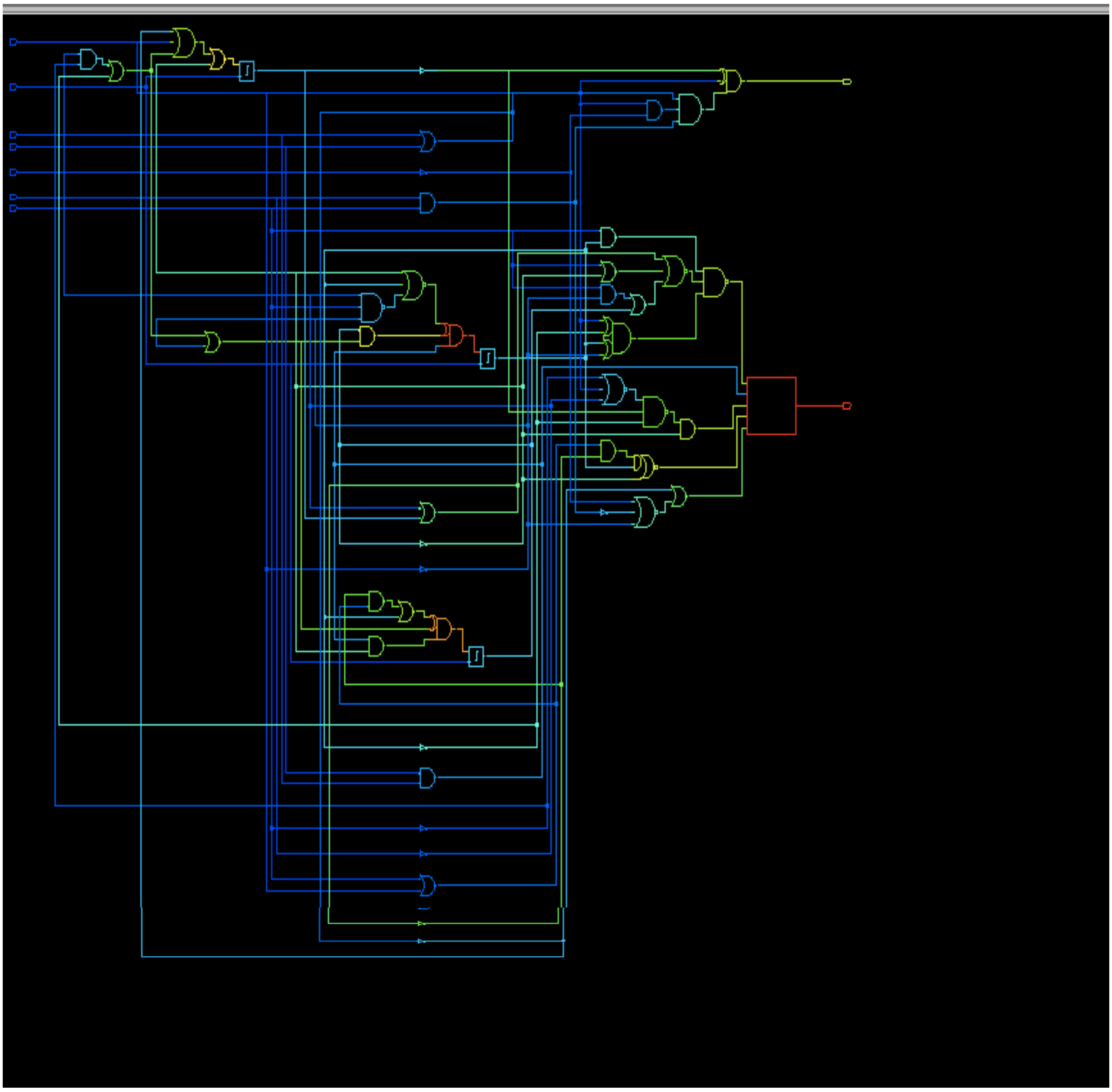
	-a	-j	-m	-o	-r
Delay in ps	2433 ps	2884 ps	2853 ps	2786 ps	2360 ps
Area in lamda power of 2	66000 lamda^2	65000 lamda^2	60000 lamda^2	79000 lamda^2	71000 lamda^2

Boog Output:

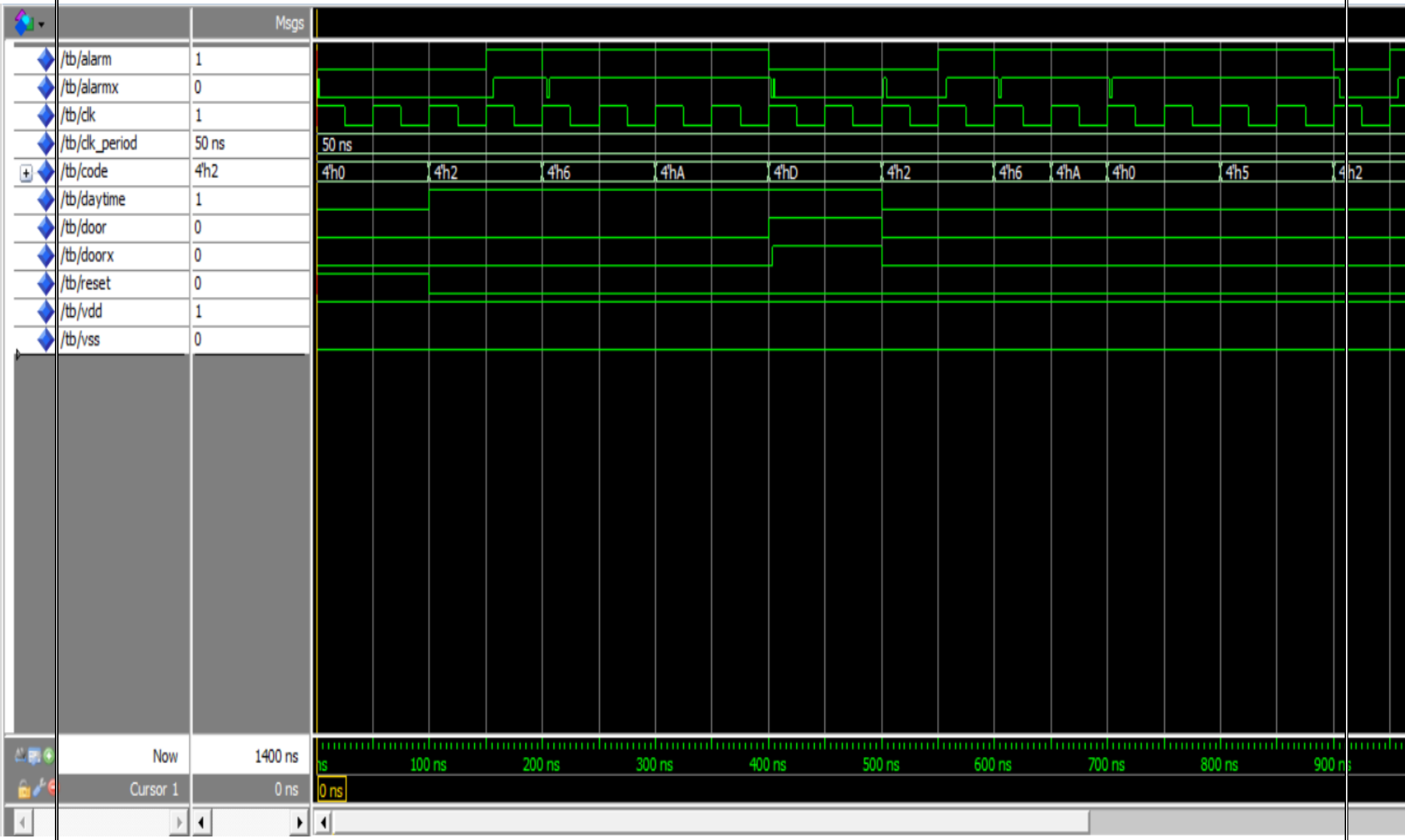
	-a	-j	-m	-o	-r
Delay in ps	2230 ps	2112 ps	2213 ps	1681 ps	2289 ps
Area in lamda square	66000 lambda²	65000 lambda²	60000 lambda²	79000 lambda²	71000 lambda²

I will choose the -a encoding because it has an average delay and area.

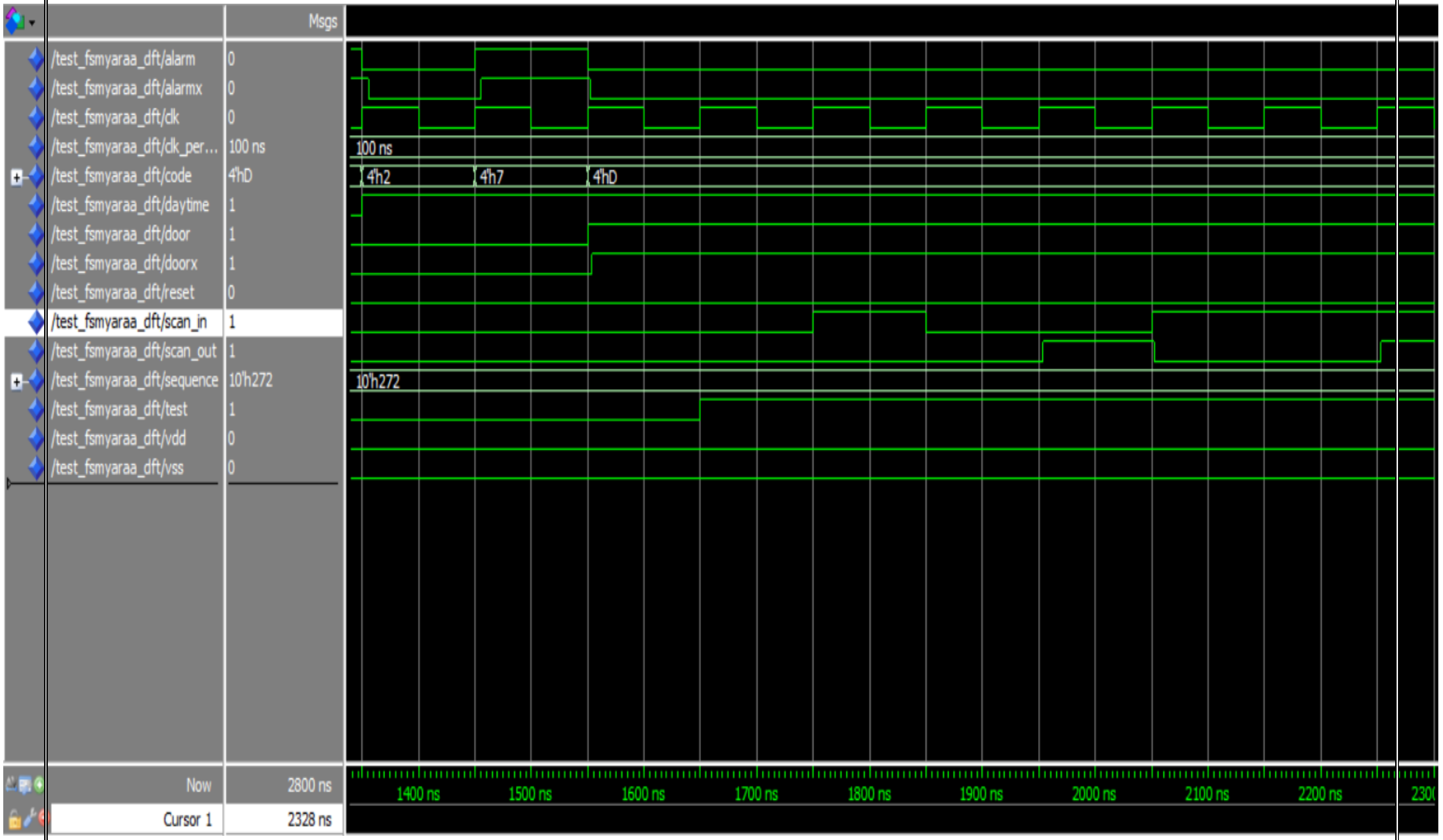
XSCH of -a encoding style from loon



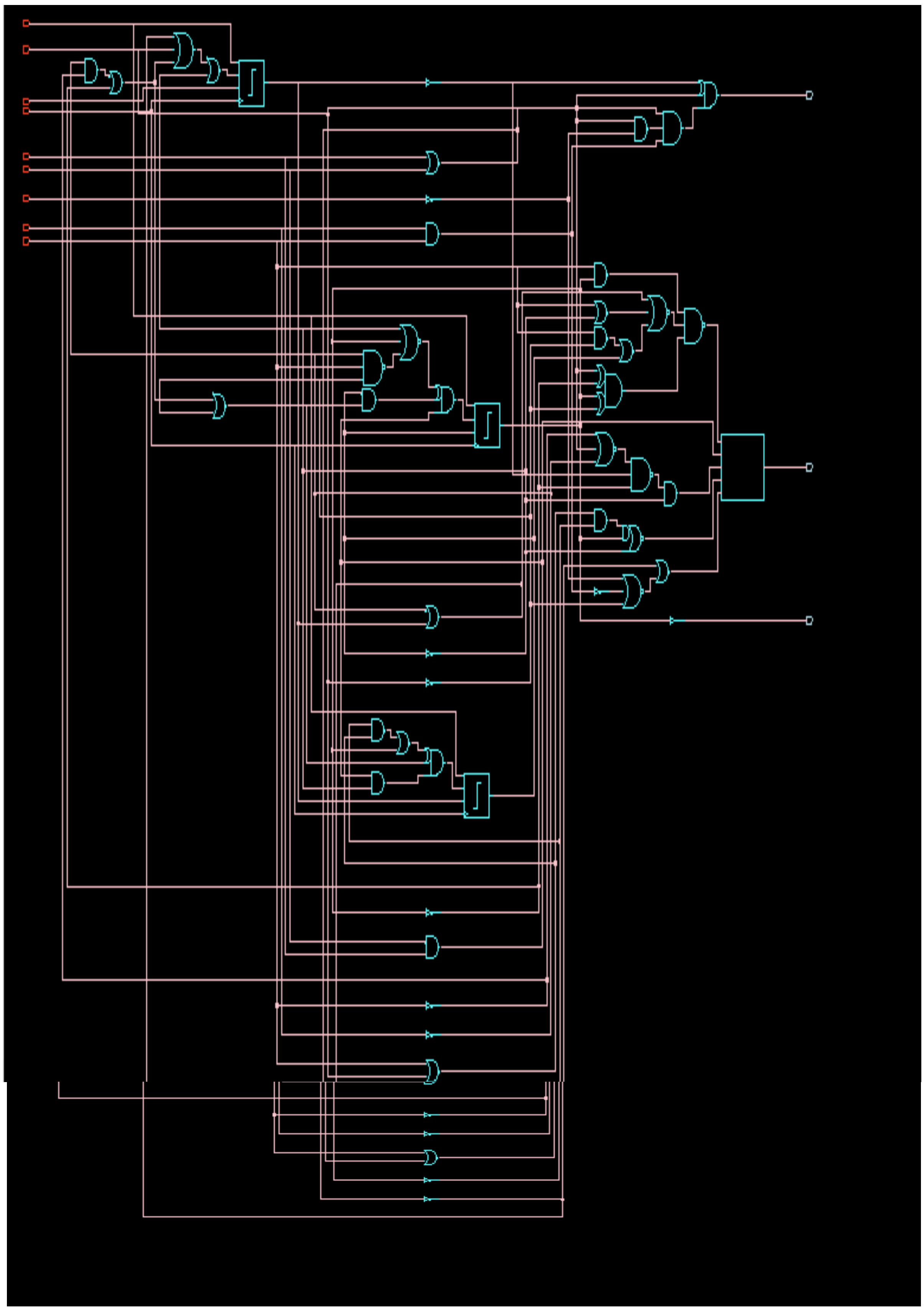
Comparaison between fsmyara.vhd and fsmyaraa_b_l.vhd of outputs:



Comparaison between fsmyara.vhd and fsmyaraa_dft.vhd of outputs:


Msgs
/test_fsmyaraa_dft/alarm 0
/test_fsmyaraa_dft/alarmx 0
/test_fsmyaraa_dft/clk 0
/test_fsmyaraa_dft/clk_per... 100 ns
/test_fsmyaraa_dft/code 4'hD
/test_fsmyaraa_dft/daytime 1
/test_fsmyaraa_dft/door 1
/test_fsmyaraa_dft/doorx 1
/test_fsmyaraa_dft/reset 0
/test_fsmyaraa_dft/scan_in 1
/test_fsmyaraa_dft/scan_out 1
/test_fsmyaraa_dft/sequence 10'h272
/test_fsmyaraa_dft/test 1
/test_fsmyaraa_dft/vdd 0
/test_fsmyaraa_dft/vss 0
100 ns
4'h2
4'h7
4'hD
10'h272
Now 2800 ns
Cursor 1 2328 ns
1400 ns
1500 ns
1600 ns
1700 ns
1800 ns
1900 ns
2000 ns
2100 ns
2200 ns

XSCH of -a encoding style from DFT



Makefile:

```
#-------fsmyara--------------------------------------------
syf: fsmyaraa.vbe \
     fsmyaraj.vbe \
     fsmyaram.vbe \
     fsmyarao.vbe \
     fsmyarar.vb
boom : fsmyaraa_b.vbe\
              fsmyaraj_b.vbe\
              fsmyarar_b.vbe\
              fsmyaram_b.vbe\
              fsmyarao_b.vbe
boog : fsmyaraa_b.vst\
              fsmyaraj_b.vst\
              fsmyaram_b.vst\
              fsmyarao_b.vst\
              fsmyarar_b.vst

loon : fsmyaraa_b_l.vst\
              fsmyaraj_b_l.vst\
              fsmyaram_b_l.vst\
              fsmyarao_b_l.vst\
            fsmyarar_b_l.vst

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


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


#-------Finite State Machine Synthesis----------------#
vhd_to_fsm:
      rename .vhd .fsm *.vhd

fsmyaraa.vbe: fsmyara.fsm
      @echo "    Encoding Synthesis -> fsmyaraa.vbe"
      syf -CEV -a fsmyara

fsmyaraj.vbe: fsmyara.fsm
      @echo "    Encoding Synthesis  -> fsmyaraj.vbe"
      syf -CEV -j fsmyara
```

```
fsmyaram.vbe: fsmyara.fsm
	@echo "    Encoding Synthesis  ->fsmyaram.vbe"
	syf -CEV -m fsmyara

fsmyarao.vbe: fsmyara.fsm
	@echo "    Encoding Synthesis  -> fsmyarao.vbe"
	syf -CEV -o fsmyara

fsmyarar.vbe: fsmyara.fsm
	@echo "    Encoding Synthesis  -> fsmyarar.vbe"
	syf -CEV -r fsmyara

%_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

fsmyaraa_dft.vst : fsmyaraa_b_l.vst  fsmyaraa_dft.vst
	@echo "    DFT  -> $@"
	scapin -VRB fsmyaraa_b_l path fsmyaraa_dft>
fsmyaraa_DFT.out


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

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

Paramfile.lax:

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

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
```

Fsmyaraa_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/alliance/cells/ramlib:/usr/lib/alliance/cells/romlib:/usr/lib/alliance/cells/pxlib:/usr/lib/alliance/cells/padlib
========================== Files, Options and Parameters ==========================
First VHDL file          = fsmyaraa.vbe
Second VHDL file= fsmyaraa_b_l_net.vbe
The auxiliary signals are erased
Errors are displayed
```

```
==========================================
=======================================

Compiling 'fsmyaraa' ...
Compiling 'fsmyaraa_b_l_net' ...
---> final number of nodes = 284(106)

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

            './fsmyaraa'  and  './fsmyaraa_b_l_net'
----------------------------------------------------------------------
-----
==============================  PRIMARY
OUTPUT  ===============================
=============================  AUXILIARY
SIGNAL  ===============================
==============================  REGISTER
SIGNAL  ===============================
```

============================= EXTERNAL BUS ==============================

============================== INTERNAL BUS ==============================

Formal Proof : OK

pppppppppppppppppppppprrrrrrrrrrrroooooooooooooooooooooooooofffffffffffffff
----

Testbench for fsmyaraa_b_l :

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

ARCHITECTURE tba OF tb IS

-- Component Declaration for the Device Under Test
(DUT)
component dac is
port  (
        clk    : in bit;
        vdd    : in bit;
        vss    : in bit;
        code       : in bit_vector (3 downto 0);
        daytime     : in      bit;
         reset : in bit;
        door     : out  bit;
        alarm     : out bit
       );
END component dac;
```

```
component fsmyaraa_b_l is
   port (
      clk     : in      bit;
      vdd     : in      bit;
      vss     : in      bit;
      code    : in      bit_vector(3 downto 0);
      daytime : in      bit;
      reset   : in      bit;
      door    : out     bit;
      alarm   : out     bit
 );
end component fsmyaraa_b_l;

FOR dut: dac USE ENTITY WORK.dac (fsm);
FOR dut1: fsmyaraa_b_l USE ENTITY
WORK.fsmyaraa_b_l (structural);


-- Declare input signals and initialize them
SIGNAL clk    : bit := '0';
SIGNAL vdd    : bit := '1';
SIGNAL vss    : bit := '0';
SIGNAL code     : bit_vector (3 downto 0);
SIGNAL daytime    : bit := '0';
SIGNAL reset    : bit := '0';
SIGNAL door : bit := '0';
SIGNAL alarm   : bit := '0';
SIGNAL doorx : bit := '0';
SIGNAL alarmx   : bit := '0';


-- Constants and Clock period definitions
constant clk_period : time := 50 ns;
BEGIN

-- Instantiate the Device Under Test (DUT)
  dut: dac PORT MAP (clk, vdd, vss, code, daytime,
reset, door, alarm);
  dut1: fsmyaraa_b_l PORT MAP (clk, vdd, vss, code,
daytime, reset, doorx, alarmx);

-- Clock process definitions
   clk_process :process
   begin
```

```
         clk <= '1';
         wait for clk_period/2;
         clk <= '0';
         wait for clk_period/2;
   end process;

-- Stimulus process, refer to clock signal
proc: PROCESS IS
BEGIN
        --test0
        reset<='1';
        WAIT FOR 100 ns;
        ASSERT door = doorx and alarm = alarmx
        REPORT "Error not 2"
        SEVERITY error;

        --test1
        reset <= '0'; code <= "0010" ; daytime <= '1';
        WAIT FOR 100 ns;
        ASSERT door = doorx and alarm = alarmx
        REPORT "Error not 2"
        SEVERITY error;
```

```
 code <= "0110" ;
WAIT FOR 100 ns;
ASSERT door = doorx and alarm = alarmx
REPORT "Error not 6"
SEVERITY error;

 code <= "1010" ;
WAIT FOR 100 ns;
ASSERT door = doorx and alarm = alarmx
REPORT "Error not A"
SEVERITY error;

 code <= "1101" ;
WAIT FOR 100 ns;
ASSERT door = doorx and alarm = alarmx
REPORT "Error , can't open!"
SEVERITY error;


 --test2
```

```
reset <= '0'; code <= "0010" ; daytime <= '0';
WAIT FOR 100 ns;
ASSERT door = doorx and alarm = alarmx
REPORT "Error not 2"
SEVERITY error;


 code <= "0110" ;
WAIT FOR 50 ns;
ASSERT door = doorx and alarm = alarmx
REPORT "Error not 6"
SEVERITY error;


 code <= "1010" ;
WAIT FOR 50 ns;
ASSERT door = doorx and alarm = alarmx
REPORT "Error not A"
SEVERITY error;


 code <= "0000" ;
```

```
WAIT FOR 100 ns;
ASSERT door = doorx and alarm = alarmx
REPORT "Error not zero"
SEVERITY error;

 code <= "0101" ;
WAIT FOR 100 ns;
ASSERT door = doorx and alarm = alarmx
REPORT "Error not 5"
SEVERITY error;

--test3



reset <= '0'; code <= "0010" ; daytime <= '0';
WAIT FOR 100 ns;
ASSERT door = doorx and alarm = alarmx
REPORT "Error not 2"
SEVERITY error;



 code <= "0110" ;
```

```
WAIT FOR 100 ns;
ASSERT door = doorx and alarm = alarmx
REPORT "Error not 6"
SEVERITY error;


 code <= "1010" ;
WAIT FOR 100 ns;
ASSERT door = doorx and alarm = alarmx
REPORT "Error not A"
SEVERITY error;


 code <= "0000" ;
WAIT FOR 50 ns;
ASSERT door = doorx and alarm = alarmx
REPORT "Error not Zero"
SEVERITY error;


 code <= "0111" ;
WAIT FOR 100 ns;
```

```
ASSERT door = doorx and alarm = alarmx
REPORT "Error can't read not 5"
SEVERITY error;

 --test4
reset <= '0'; code <= "0010" ; daytime <= '1';
WAIT FOR 100 ns;
ASSERT door = doorx and alarm = alarmx
REPORT "Error not 2"
SEVERITY error;



 code <= "0111" ;
WAIT FOR 100 ns;
ASSERT door = doorx and alarm = alarmx
REPORT "Error can't read not 6"
SEVERITY error;

 --test5
reset <= '0'; code <= "1101" ; daytime <= '1';
WAIT FOR 100 ns;
ASSERT door = doorx and alarm = alarmx
```

```
        REPORT "Error can't open"
        SEVERITY error;



    WAIT; -- stop process simulation run

    END PROCESS proc;
    END ARCHITECTURE tba;
```

Testbench for fsmyaraa_dft :

```
entity test_fsmyaraa_dft is
end test_fsmyaraa_dft;

architecture test of test_fsmyaraa_dft is
component dac is
port  (
      clk    : in bit;
      vdd    : in bit;
      vss    : in bit;
      code       : in bit_vector (3 downto 0);
      daytime     : in      bit;
       reset : in bit;
      door      : out  bit;
      alarm      : out bit


     );
end component dac;
for dut: dac use entity work.dac (fsm);
component fsmyaraa_dft is
   port (
     clk      : in      bit;
     vdd      : in      bit;
     vss      : in      bit;
     code     : in      bit_vector(3 downto 0);
     daytime  : in      bit;
     reset    : in      bit;
     door     : out     bit;
     alarm    : out     bit;
     scan_in  : in      bit;
     test     : in      bit;
     scan_out : out     bit
 );
end component fsmyaraa_dft;
for dut1: fsmyaraa_dft use entity work.fsmyaraa_dft
(structural);

      signal    code      : bit_vector(3 downto 0) :=
"0000";
      signal    daytime        : bit := '0';
      signal    reset     : bit := '0';
      signal    clk       : bit := '0';
      signal    vss  : bit := '0';
      signal    vdd       : bit := '0';
```

```
        signal    door       : bit := '0';
        signal    alarm            : bit := '0';
        signal    doorx      : bit := '0';
        signal    alarmx           : bit := '0';
        signal    scan_in : bit  := '0';
        signal    test : bit := '0';
        signal    scan_out: bit := '0';
        signal    sequence: bit_vector(9 downto 0) :=
"1001110010";
        constant clk_period : time := 100 ns;
begin
        dut: dac port map
(clk,vdd,vss,code,daytime,reset,door,alarm);
        dut1: fsmyaraa_dft port map
(clk,vdd,vss,code,daytime,reset,doorx,alarmx,scan_in,te
st,scan_out);
        clk_process :process
        begin
         clk <= '0';
         wait for clk_period/2;
         clk <= '1';
         wait for clk_period/2;
```

```
end process clk_process;

stim_process :process
begin

--test0
reset<='1';
WAIT FOR 100 ns;
ASSERT door = doorx and alarm = alarmx
REPORT "Error not 2"
SEVERITY error;

--test1
reset <= '0'; code <= "0010" ; daytime <= '1';
WAIT FOR 100 ns;
ASSERT door = doorx and alarm = alarmx
REPORT "Error not 2"
SEVERITY error;

 code <= "0110" ;
WAIT FOR 100 ns;
ASSERT door = doorx and alarm = alarmx
```

```
REPORT "Error not 6"
SEVERITY error;

 code <= "1010" ;
WAIT FOR 100 ns;
ASSERT door = doorx and alarm = alarmx
REPORT "Error not A"
SEVERITY error;

 code <= "1101" ;
WAIT FOR 100 ns;
ASSERT door = doorx and alarm = alarmx
REPORT "Error , can't open!"
SEVERITY error;


 --test2

reset <= '0'; code <= "0010" ; daytime <= '0';
WAIT FOR 100 ns;
ASSERT door = doorx and alarm = alarmx
REPORT "Error not 2"
```

```
SEVERITY error;


 code <= "0110" ;
WAIT FOR 50 ns;
ASSERT door = doorx and alarm = alarmx
REPORT "Error not 6"
SEVERITY error;


 code <= "1010" ;
WAIT FOR 50 ns;
ASSERT door = doorx and alarm = alarmx
REPORT "Error not A"
SEVERITY error;


 code <= "0000" ;
WAIT FOR 100 ns;
ASSERT door = doorx and alarm = alarmx
REPORT "Error not zero"
SEVERITY error;
```

```
 code <= "0101" ;
WAIT FOR 100 ns;
ASSERT door = doorx and alarm = alarmx
REPORT "Error not 5"
SEVERITY error;

--test3


reset <= '0'; code <= "0010" ; daytime <= '0';
WAIT FOR 100 ns;
ASSERT door = doorx and alarm = alarmx
REPORT "Error not 2"
SEVERITY error;


 code <= "0110" ;
WAIT FOR 100 ns;
ASSERT door = doorx and alarm = alarmx
REPORT "Error not 6"
SEVERITY error;
```

```
 code <= "1010" ;
WAIT FOR 100 ns;
ASSERT door = doorx and alarm = alarmx
REPORT "Error not A"
SEVERITY error;


 code <= "0000" ;
WAIT FOR 50 ns;
ASSERT door = doorx and alarm = alarmx
REPORT "Error not Zero"
SEVERITY error;


 code <= "0111" ;
WAIT FOR 100 ns;
ASSERT door = doorx and alarm = alarmx
REPORT "Error can't read not 5"
SEVERITY error;
```

```
 --test4
reset <= '0'; code <= "0010" ; daytime <= '1';
WAIT FOR 100 ns;
ASSERT door = doorx and alarm = alarmx
REPORT "Error not 2"
SEVERITY error;



 code <= "0111" ;
WAIT FOR 100 ns;
ASSERT door = doorx and alarm = alarmx
REPORT "Error can't read not 6"
SEVERITY error;

 --test5
reset <= '0'; code <= "1101" ; daytime <= '1';
WAIT FOR 100 ns;
ASSERT door = doorx and alarm = alarmx
REPORT "Error can't open"
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  clk_period  ;
        end loop;

        wait;
    end process stim_process;
end test;
```