Ain Shams University

Fuculty of Engineering - CHEP
CESS Program

Fall 2019

Electronic Design Automation (CSE215)

PROJECT PART (2): SYNTHESIS

Name: Ahmed Abd Elnasser

ID: 17P8113

Email: aahmed.aabdelnasser@gmail.com

Program: CESS

INTRODUCTION:

This document reports part (2) of the project where we make the low level (logic) synthesis to the finite state machine from part (1), using alliance tools descripped below.

The ALLIANCE tools used are:

- **syf**: Finite State Machine synthesizer
- **boom:** BOOlean Minimization.
- **boog:** Library Binding.
- **loon:** Local optimizations of Nets.
- **xsch:** Graphical netlist viewer.
- **flatbeh:** Behavioral from Structural.
- **proof:** Formal Verification.
- **scapin**: Scan-path insertion (DFT).

In addition to **ModelSim** for simulation.

2. PART 2: LOGIC SYNTHESIS:

2.1. FSM Synthesis (syf):

Applying (syf) on the (.fsm) file, Several dataflow descriptions (.vbe) corresponding to different state encodings are obtained.

Different state encoding algorithms -a, -j, -m, -o, -r, using the options -CEV.

2.2. Boolean Network Optimization (boom):

Launching Boolean optimization with the tool **BOOM** by asking an appropriate optimization in **delay** and **area** for each of the (.vbe) obtained above.

2.3. Binding and Optimizing On Gates (boog):

Mapping into a standard cell library (Technology Dependent), Translates boolean functions associated to each node of a boolean network into a network of gates. Each node is treated independently.

Uses a paramfile.lax containing appropriate options for optimizing.

2.4. Local Optimizing Of Nets (loon):

Operates only on critical paths of all the structural views obtained from the (boom) step to reduce the propagation time. And also uses the paramfile.lax used in the previous step.

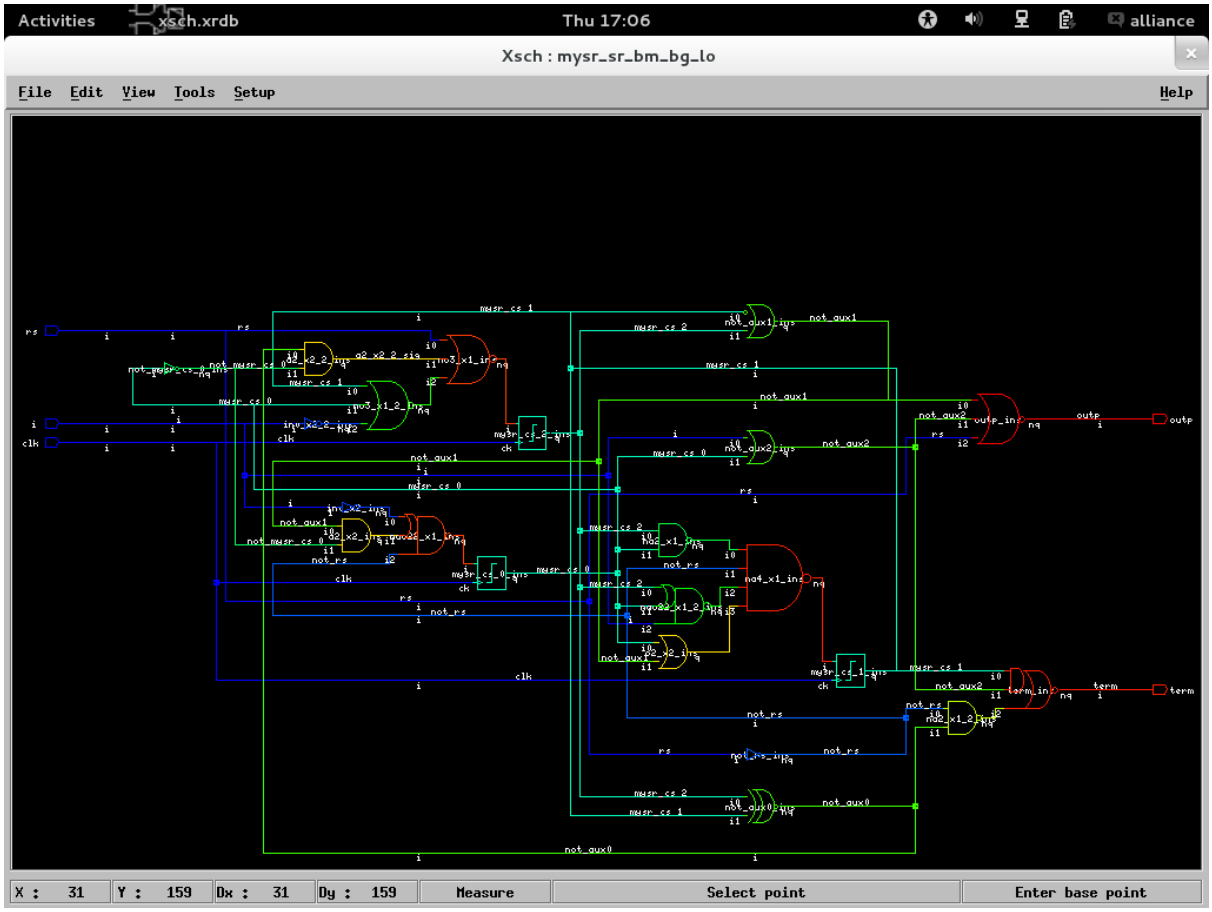
2.4.1. Comparing all realizations:

State Encoding	Area (lamda^2)	CP Delay (ps)
a	37750	1891
j	34750	2032
m	42750	2327
o	47750	1449
r	36750	1649

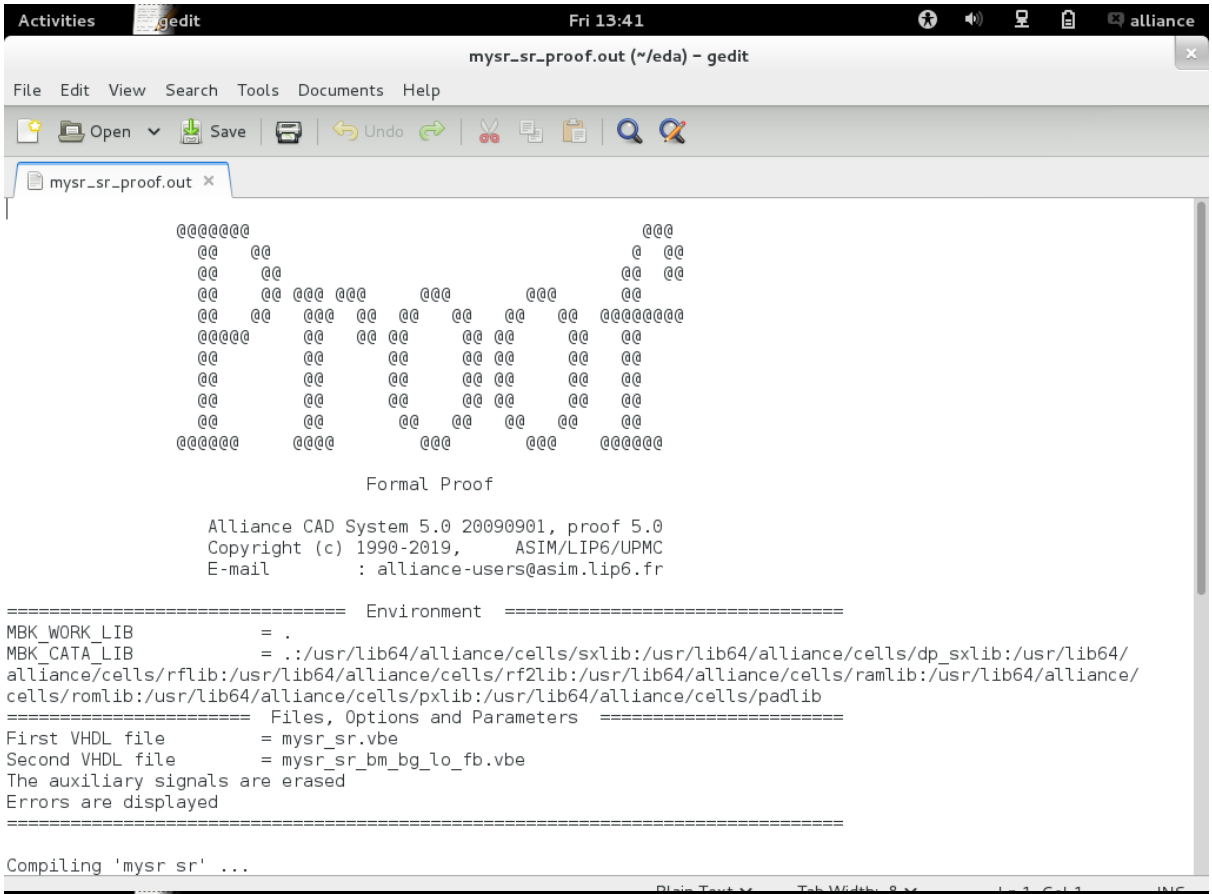
2.4.2. Choosing a netlist:

The choose of the netlist to resume with depended on harmonizing between Area and Delay. So, the **(r)** was the chosen one. Although it doesn't has the minimum of all, but has good combination between Area and Delay time.

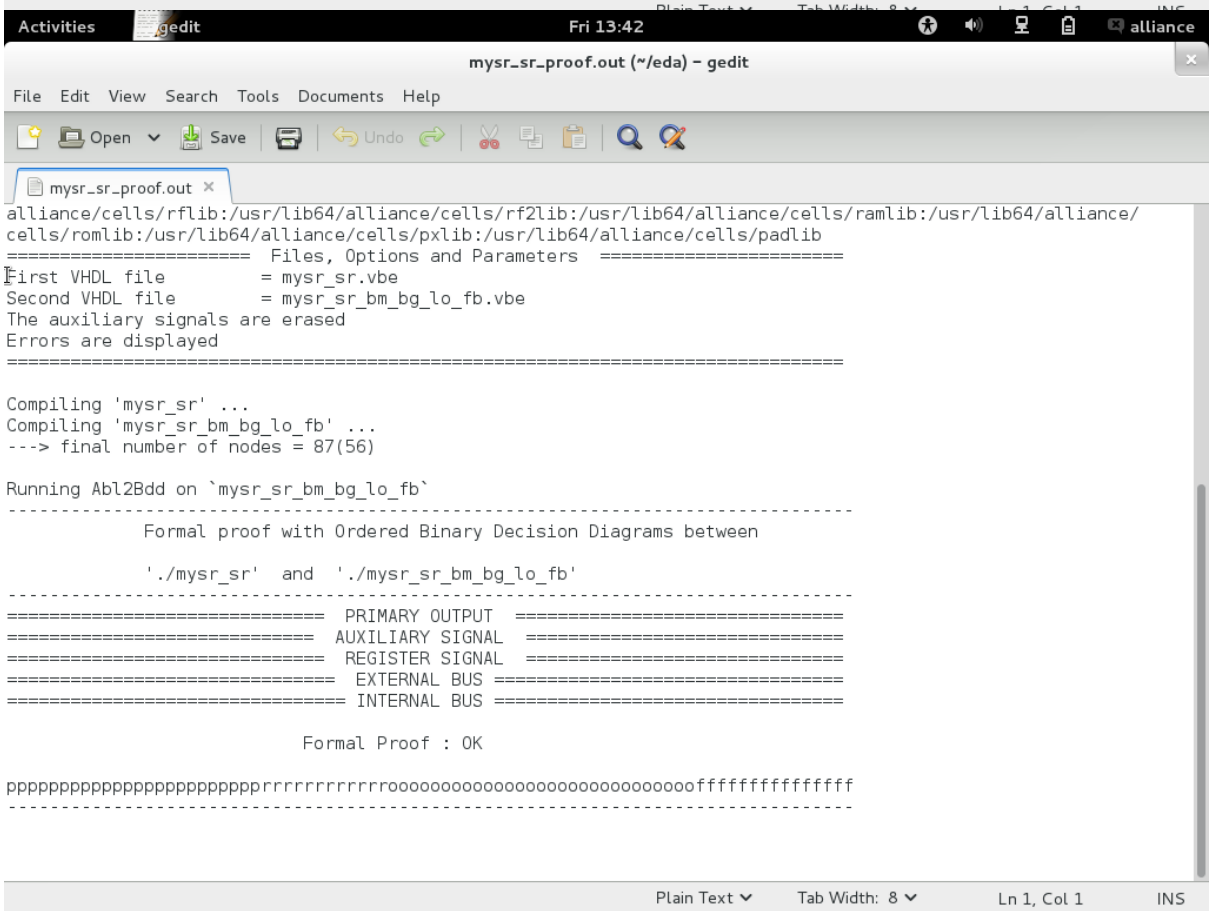
2.5. CP Netlist Visualization:



2.6. Netlist Checking (Formal proof):



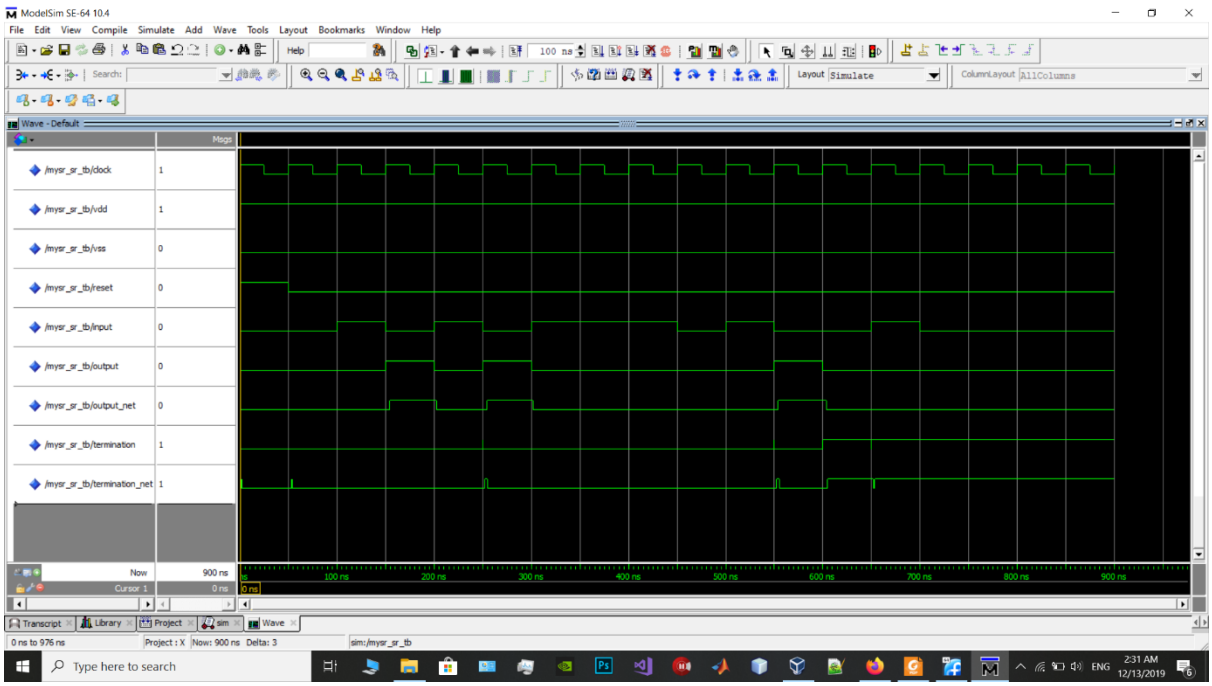
```
Activities   gedit   Fri 13:41   alliance
mysr_sr_proof.out (~/eda) – gedit
File  Edit  View  Search  Tools  Documents  Help
Open  Save  Undo
mysr_sr_proof.out

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

                              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/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         = mysr_sr.vbe
Second VHDL file        = mysr_sr_bm_bg_lo_fb.vbe
The auxiliary signals are erased
Errors are displayed
================================================================================

Compiling 'mysr sr' ...
```



```
Activities   gedit   Fri 13:42   alliance
mysr_sr_proof.out (~/eda) – gedit
File  Edit  View  Search  Tools  Documents  Help
Open  Save  Undo
mysr_sr_proof.out
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         = mysr_sr.vbe
Second VHDL file        = mysr_sr_bm_bg_lo_fb.vbe
The auxiliary signals are erased
Errors are displayed
================================================================================

Compiling 'mysr_sr' ...
Compiling 'mysr_sr_bm_bg_lo_fb' ...
---> final number of nodes = 87(56)

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

            './mysr_sr'  and  './mysr_sr_bm_bg_lo_fb'
--------------------------------------------------------------------------------
================================  PRIMARY OUTPUT  ================================
==============================  AUXILIARY SIGNAL  ================================
==============================  REGISTER SIGNAL  =================================
================================  EXTERNAL BUS ==================================
================================ INTERNAL BUS ==================================

                         Formal Proof : OK

ppppppppppppppppppppppppprrrrrrrrrrrrrrooooooooooooooooooooooooooooooffffffffffffffff
--------------------------------------------------------------------------------

Plain Text   Tab Width: 8   Ln 1, Col 1   INS
```

2.7. Delay Simulation:

The obtained gate-level netlist has been functionally verified. Standard-cell delay has been ignored during all the above steps. Note that both **BOOG** and **LOON** synthesis and optimization tools have estimated the critical path delay. It should be easy to verify that this delay is less than the required clock period. It is time now for a delay simulation to doublecheck the speed performance.



2.7.1 Simulation wave observations:

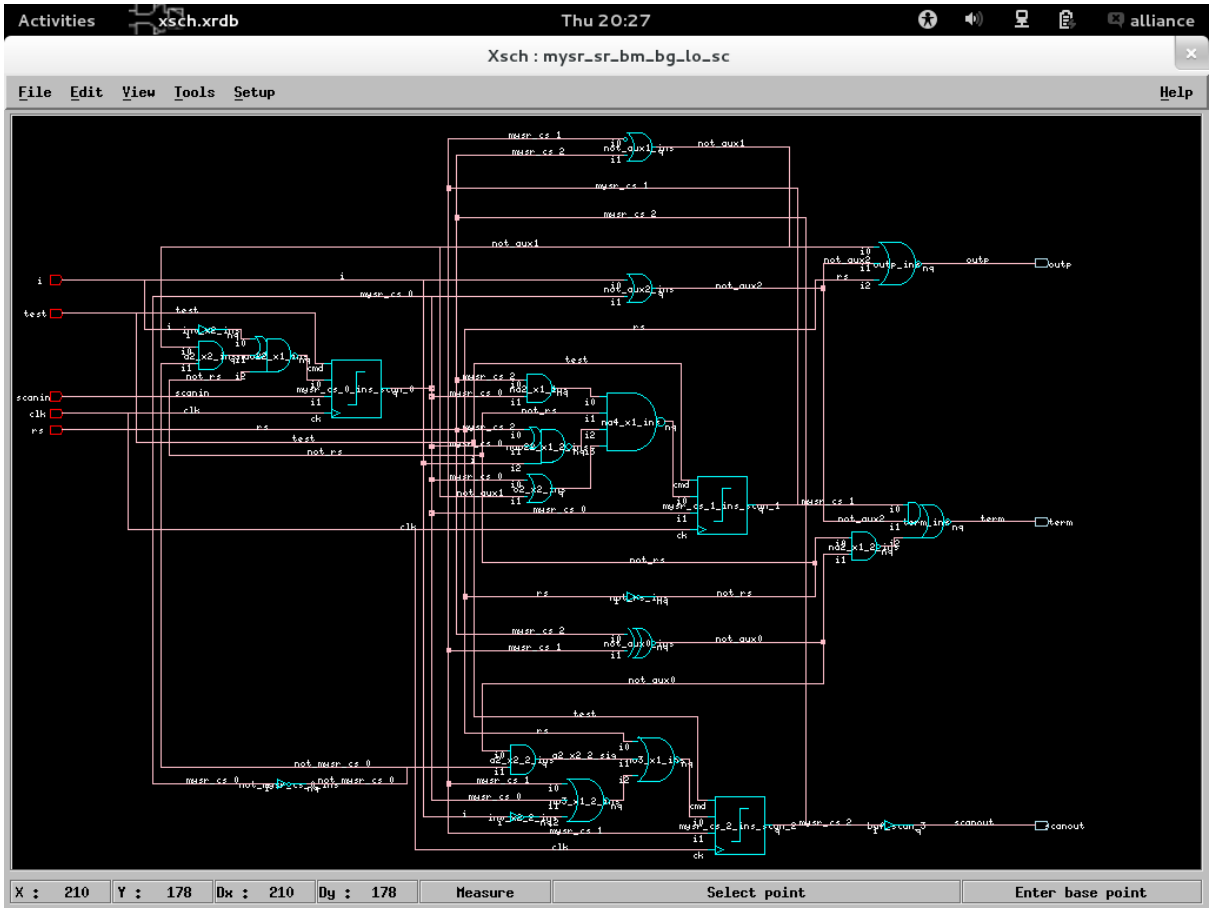
- The structural (output_net) wave matches the behavioural FSM (output) wave with a little delay (netlist gates delay).
- The structural (termination_net) wave matches the behavioural FSM (termination) wave with a little delay (netlist gates delay).

2.7.2 Conclusion:

The structural view of the synthesized FSM is **valid** compared to the initial behavioural FSM.

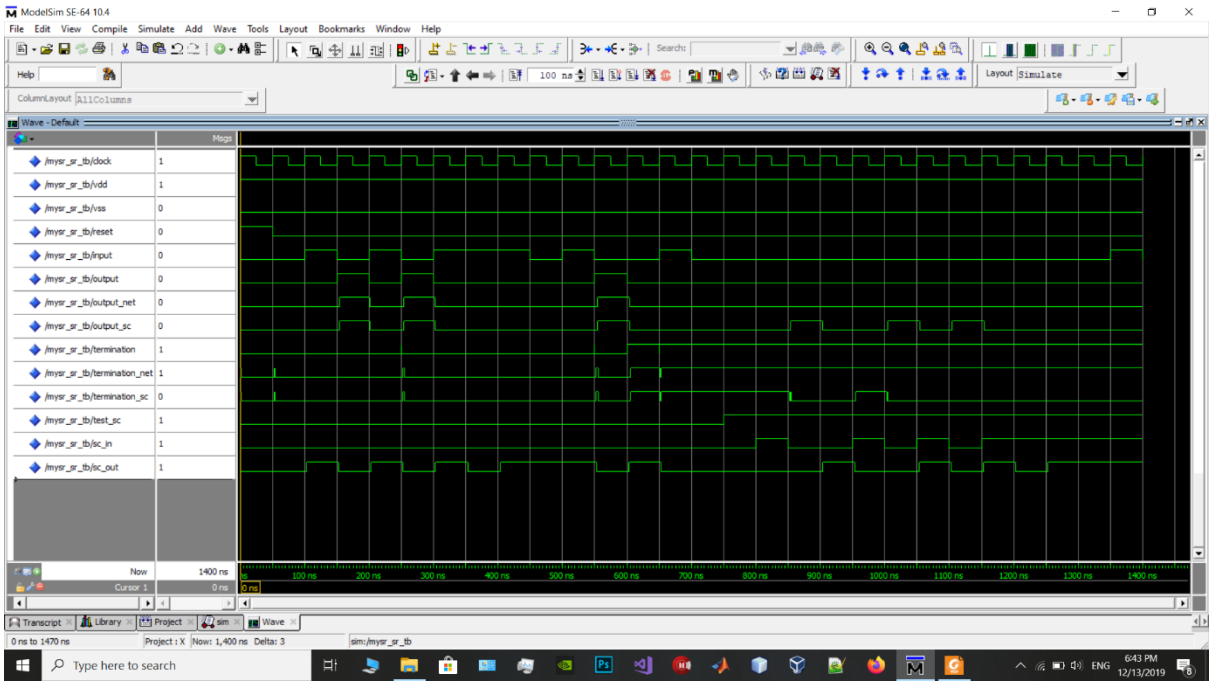
2.8. Scan Path Insertion (scapin) (DFT):

With the **SCAPIN** tool, we can insert a scan-path into the netlist. The scan-path allows the designer to observe in test mode the stored values of all registers of the circuit. The path is created by changing each register into a mux_register (i.e. by inserting a multiplexer in front of all registers) and connecting them in series.



2.9. Scan Path Simulation:

After extending the testbench to now test the three modules of the FSM, we obtained the wave shown below.



2.9.1. Simulation wave observation:

- While the (test) signal is set to zero, the three (output) and (termination) waves are almost identical with a little delay.
- When the (test) signal becomes 1, the circuit mode is swiched to the shifting mode. And we can observe that the scanout (sc_out) wave mathces the scanin (sc_in) with a delay of 3 clock cycles (at the rising edge of the third one).

2.9.2. Conclusion:

The scan path test is successful.

Appendix:

This Appendix contains all the vhdl source codes, in addition to the makefile and any other used codes.

VHDL Source Codes:

Behavioural FSM:

```
--string recognizer finite state machine (mysr) simulation



entity mysr is

port(
i: in bit;
outp: out bit;
term: out bit;
clk: in bit;
rs: in bit;
vdd,vss: in bit
);

end mysr;
```

```
architecture mealy_arch of mysr is

--enumeration type for states
type state is (s0,s1,s2,s3,s4,s5);

--declaring signals
signal cs: state;
signal ns: state;

--synthesis directives
--pragma current_state cs
--pragma next_state ns
--pragma clock clk


begin

--next_state transition process with synchronous reset
process(cs,i,rs)
begin

if(rs = '1') then ns <= s0; outp <= '0'; term <= '0';

else

case cs is
```

```
when s0 =>
if(i = '0') then ns <= s1; outp <= '0'; term <= '0';
else ns <= s4; outp <= '0'; term <= '0';
end if;

when s1 =>
if(i = '0') then ns <= s1; outp <= '0'; term <= '0';
else ns <= s2; outp <= '0'; term <= '0';
end if;

when s2 =>
if(i = '0') then ns <= s3; outp <= '1'; term <= '0';
else ns <= s4; outp <= '0'; term <= '0';
end if;

when s3 =>
if(i = '0') then ns <= s5; outp <= '0'; term <= '1';
else ns <= s2; outp <= '0'; term <= '0';
end if;

when s4 =>
if(i = '0') then ns <= s3; outp <= '0'; term <= '0';
else ns <= s4; outp <= '0'; term <= '0';
end if;

when s5 =>
ns <= s5; outp <= '0'; term <= '1';
```

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


end case;


end if;


end process;



--state update process
process(clk)
begin

if(clk = '1' and not clk' stable) then
cs <= ns;
end if;


end process;


end mealy_arch;
```

Structural Netlist (loon):

```
LIBRARY sx_lib;

entity mysr_sr_bm_bg_lo is
   port (
      i    : in      bit;
      outp : out     bit;
      term : out     bit;
      clk  : in      bit;
      rs   : in      bit;
      vdd  : in      bit;
      vss  : in      bit
 );
end mysr_sr_bm_bg_lo;

architecture structural of mysr_sr_bm_bg_lo is
Component nxr2_x1
   port (
      i0  : in      bit;
      i1  : 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 o2_x2
   port (
      i1  : in      bit;
      i0  : in      bit;
      q   : out     bit;
      vdd : in      bit;
      vss : in      bit
 );
end component;

Component nao22_x1
   port (
      i0  : in      bit;
      i1  : in      bit;
      i2  : in      bit;
      nq  : out     bit;
      vdd : in      bit;
```

```
      vss : in      bit
 );
end component;

Component na4_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 a2_x2
   port (
```

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

```
Component noa22_x1
   port (
      i0  : in      bit;
      i1  : in      bit;
      i2  : in      bit;
      nq  : out     bit;
      vdd : in      bit;
      vss : in      bit
 );
end component;

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

signal mysr_cs        : bit_vector( 2 downto 0);
signal not_mysr_cs    : bit_vector( 0 downto 0);
signal o2_x2_sig      : bit;
signal not_rs         : bit;
```

```
signal not_aux2       : bit;
signal not_aux1       : bit;
signal not_aux0       : bit;
signal no3_x1_sig     : bit;
signal no3_x1_2_sig   : bit;
signal nao22_x1_sig   : bit;
signal nao22_x1_2_sig : bit;
signal na4_x1_sig     : bit;
signal na2_x1_sig     : bit;
signal na2_x1_2_sig   : bit;
signal inv_x2_sig     : bit;
signal inv_x2_2_sig   : bit;
signal a2_x2_sig      : bit;
signal a2_x2_2_sig    : bit;

begin

not_aux2_ins : o2_x2
   port map (
      i0  => i,
      i1  => mysr_cs(0),
      q   => not_aux2,
      vdd => vdd,
      vss => vss
   );

not_aux0_ins : nxr2_x1
```

```
   port map (
      i0  => mysr_cs(2),
      i1  => mysr_cs(1),
      nq  => not_aux0,
      vdd => vdd,
      vss => vss
   );

not_mysr_cs_0_ins : inv_x2
   port map (
      i   => mysr_cs(0),
      nq  => not_mysr_cs(0),
      vdd => vdd,
      vss => vss
   );

not_aux1_ins : on12_x1
   port map (
      i0  => mysr_cs(1),
      i1  => mysr_cs(2),
      q   => not_aux1,
      vdd => vdd,
      vss => vss
   );

not_rs_ins : inv_x2
   port map (
```

```
      i   => rs,
      nq  => not_rs,
      vdd => vdd,
      vss => vss
   );

a2_x2_ins : a2_x2
   port map (
      i0  => not_aux1,
      i1  => not_mysr_cs(0),
      q   => a2_x2_sig,
      vdd => vdd,
      vss => vss
   );

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

nao22_x1_ins : nao22_x1
   port map (
      i0  => inv_x2_sig,
      i1  => a2_x2_sig,
```

```
      i2  => not_rs,
      nq  => nao22_x1_sig,
      vdd => vdd,
      vss => vss
   );

mysr_cs_0_ins : sff1_x4
   port map (
      ck  => clk,
      i   => nao22_x1_sig,
      q   => mysr_cs(0),
      vdd => vdd,
      vss => vss
   );

o2_x2_ins : o2_x2
   port map (
      i1  => not_aux1,
      i0  => mysr_cs(0),
      q   => o2_x2_sig,
      vdd => vdd,
      vss => vss
   );

nao22_x1_2_ins : nao22_x1
   port map (
      i0  => mysr_cs(2),
```

```
      i1  => mysr_cs(0),
      i2  => i,
      nq  => nao22_x1_2_sig,
      vdd => vdd,
      vss => vss
   );

na2_x1_ins : na2_x1
   port map (
      i0  => mysr_cs(2),
      i1  => mysr_cs(0),
      nq  => na2_x1_sig,
      vdd => vdd,
      vss => vss
   );

na4_x1_ins : na4_x1
   port map (
      i0  => na2_x1_sig,
      i1  => not_rs,
      i2  => nao22_x1_2_sig,
      i3  => o2_x2_sig,
      nq  => na4_x1_sig,
      vdd => vdd,
      vss => vss
   );
```

```
mysr_cs_1_ins : sff1_x4
   port map (
      ck  => clk,
      i   => na4_x1_sig,
      q   => mysr_cs(1),
      vdd => vdd,
      vss => vss
   );

inv_x2_2_ins : inv_x2
   port map (
      i   => i,
      nq  => inv_x2_2_sig,
      vdd => vdd,
      vss => vss
   );

no3_x1_2_ins : no3_x1
   port map (
      i0  => mysr_cs(1),
      i1  => mysr_cs(0),
      i2  => inv_x2_2_sig,
      nq  => no3_x1_2_sig,
      vdd => vdd,
      vss => vss
   );
```

```
a2_x2_2_ins : a2_x2
   port map (
      i0  => not_aux0,
      i1  => not_mysr_cs(0),
      q   => a2_x2_2_sig,
      vdd => vdd,
      vss => vss
   );

no3_x1_ins : no3_x1
   port map (
      i0  => rs,
      i1  => a2_x2_2_sig,
      i2  => no3_x1_2_sig,
      nq  => no3_x1_sig,
      vdd => vdd,
      vss => vss
   );

mysr_cs_2_ins : sff1_x4
   port map (
      ck  => clk,
      i   => no3_x1_sig,
      q   => mysr_cs(2),
      vdd => vdd,
      vss => vss
   );
```

```
na2_x1_2_ins : na2_x1
   port map (
      i1  => not_aux0,
      i0  => not_rs,
      nq  => na2_x1_2_sig,
      vdd => vdd,
      vss => vss
   );

term_ins : noa22_x1
   port map (
      i0  => mysr_cs(1),
      i1  => not_aux2,
      i2  => na2_x1_2_sig,
      nq  => term,
      vdd => vdd,
      vss => vss
   );

outp_ins : no3_x1
   port map (
      i1  => not_aux2,
      i0  => not_aux1,
      i2  => rs,
      nq  => outp,
      vdd => vdd,
```

```
        vss => vss
    );


end structural;
```

Structural Scan Path (scapin) (DFT):

```
LIBRARY sx_lib;

entity mysr_sr_bm_bg_lo_sc is
   port (
      i       : in      bit;
      outp    : out     bit;
      term    : out     bit;
      clk     : in      bit;
      rs      : in      bit;
      vdd     : in      bit;
      vss     : in      bit;
      scanin  : in      bit;
      test    : in      bit;
      scanout : out     bit
 );
end mysr_sr_bm_bg_lo_sc;

architecture structural of mysr_sr_bm_bg_lo_sc is
Component nxr2_x1
   port (
      i0  : in      bit;
      i1  : 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 o2_x2
   port (
      i1  : in      bit;
      i0  : in      bit;
      q   : out     bit;
      vdd : in      bit;
      vss : in      bit
 );
end component;

Component nao22_x1
   port (
      i0  : in      bit;
      i1  : in      bit;
```

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

Component na4_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 a2_x2
   port (
      i0  : in      bit;
      i1  : in      bit;
      q   : out     bit;
      vdd : in      bit;
      vss : in      bit
 );
end component;

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

Component noa22_x1
   port (
      i0  : in      bit;
      i1  : in      bit;
      i2  : in      bit;
      nq  : out     bit;
```

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

Component no3_x1
   port (
      i1  : in      bit;
      i0  : in      bit;
      i2  : 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 mysr_cs        : bit_vector( 2 downto 0);
signal not_mysr_cs    : bit_vector( 0 downto 0);
signal o2_x2_sig      : bit;
signal not_rs         : bit;
signal not_aux2       : bit;
signal not_aux1       : bit;
signal not_aux0       : bit;
signal no3_x1_sig     : bit;
signal no3_x1_2_sig   : bit;
signal nao22_x1_sig   : bit;
signal nao22_x1_2_sig : bit;
signal na4_x1_sig     : bit;
signal na2_x1_sig     : bit;
signal na2_x1_2_sig   : bit;
signal inv_x2_sig     : bit;
signal inv_x2_2_sig   : bit;
signal a2_x2_sig      : bit;
```

```
signal a2_x2_2_sig    : bit;

begin

not_aux2_ins : o2_x2
   port map (
      i1  => mysr_cs(0),
      i0  => i,
      q   => not_aux2,
      vdd => vdd,
      vss => vss
   );

not_aux0_ins : nxr2_x1
   port map (
      i0  => mysr_cs(2),
      i1  => mysr_cs(1),
      nq  => not_aux0,
      vdd => vdd,
      vss => vss
   );

not_mysr_cs_0_ins : inv_x2
   port map (
      i   => mysr_cs(0),
      nq  => not_mysr_cs(0),
      vdd => vdd,
```

```
      vss => vss
   );

not_aux1_ins : on12_x1
   port map (
      i0  => mysr_cs(1),
      i1  => mysr_cs(2),
      q   => not_aux1,
      vdd => vdd,
      vss => vss
   );

not_rs_ins : inv_x2
   port map (
      i   => rs,
      nq  => not_rs,
      vdd => vdd,
      vss => vss
   );

a2_x2_ins : a2_x2
   port map (
      i0  => not_aux1,
      i1  => not_mysr_cs(0),
      q   => a2_x2_sig,
      vdd => vdd,
      vss => vss
```

```
   );

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

nao22_x1_ins : nao22_x1
   port map (
      i0  => inv_x2_sig,
      i1  => a2_x2_sig,
      i2  => not_rs,
      nq  => nao22_x1_sig,
      vdd => vdd,
      vss => vss
   );

o2_x2_ins : o2_x2
   port map (
      i1  => not_aux1,
      i0  => mysr_cs(0),
      q   => o2_x2_sig,
      vdd => vdd,
      vss => vss
```

```
   );

nao22_x1_2_ins : nao22_x1
   port map (
      i0  => mysr_cs(2),
      i1  => mysr_cs(0),
      i2  => i,
      nq  => nao22_x1_2_sig,
      vdd => vdd,
      vss => vss
   );

na2_x1_ins : na2_x1
   port map (
      i1  => mysr_cs(0),
      i0  => mysr_cs(2),
      nq  => na2_x1_sig,
      vdd => vdd,
      vss => vss
   );

na4_x1_ins : na4_x1
   port map (
      i0  => na2_x1_sig,
      i1  => not_rs,
      i2  => nao22_x1_2_sig,
      i3  => o2_x2_sig,
```

```
      nq  => na4_x1_sig,
      vdd => vdd,
      vss => vss
   );

inv_x2_2_ins : inv_x2
   port map (
      i   => i,
      nq  => inv_x2_2_sig,
      vdd => vdd,
      vss => vss
   );

no3_x1_2_ins : no3_x1
   port map (
      i1  => mysr_cs(0),
      i0  => mysr_cs(1),
      i2  => inv_x2_2_sig,
      nq  => no3_x1_2_sig,
      vdd => vdd,
      vss => vss
   );

a2_x2_2_ins : a2_x2
   port map (
      i0  => not_aux0,
      i1  => not_mysr_cs(0),
```

```
      q   => a2_x2_2_sig,
      vdd => vdd,
      vss => vss
   );

no3_x1_ins : no3_x1
   port map (
      i1  => a2_x2_2_sig,
      i0  => rs,
      i2  => no3_x1_2_sig,
      nq  => no3_x1_sig,
      vdd => vdd,
      vss => vss
   );

na2_x1_2_ins : na2_x1
   port map (
      i1  => not_aux0,
      i0  => not_rs,
      nq  => na2_x1_2_sig,
      vdd => vdd,
      vss => vss
   );

term_ins : noa22_x1
   port map (
      i0  => mysr_cs(1),
```

```
      i1  => not_aux2,
      i2  => na2_x1_2_sig,
      nq  => term,
      vdd => vdd,
      vss => vss
   );

outp_ins : no3_x1
   port map (
      i1  => not_aux2,
      i0  => not_aux1,
      i2  => rs,
      nq  => outp,
      vdd => vdd,
      vss => vss
   );

mysr_cs_0_ins_scan_0 : sff2_x4
   port map (
      ck  => clk,
      cmd => test,
      i0  => nao22_x1_sig,
      i1  => scanin,
      q   => mysr_cs(0),
      vdd => vdd,
      vss => vss
   );
```

```
mysr_cs_1_ins_scan_1 : sff2_x4
   port map (
      ck  => clk,
      cmd => test,
      i0  => na4_x1_sig,
      i1  => mysr_cs(0),
      q   => mysr_cs(1),
      vdd => vdd,
      vss => vss
   );

mysr_cs_2_ins_scan_2 : sff2_x4
   port map (
      ck  => clk,
      cmd => test,
      i0  => no3_x1_sig,
      i1  => mysr_cs(1),
      q   => mysr_cs(2),
      vdd => vdd,
      vss => vss
   );

buf_scan_3 : buf_x2
   port map (
      i   => mysr_cs(2),
      q   => scanout,
```

```
        vdd => vdd,
        vss => vss
    );


end structural;
```

TestBench:

```
--testbench for the string recognizer fsm
--added test for the delay simulation
--added scan path simulation test

entity mysr_sr_tb is
end entity mysr_sr_tb;



architecture test of mysr_sr_tb is

--component declaration for the duts
component mysr is

port(
i: in bit;
outp: out bit;
term: out bit;
clk: in bit;
rs: in bit;
vdd,vss : in bit
);

end component mysr;

--delay simulation component
```

```
component mysr_sr_bm_bg_lo is

port(
i: in bit;
outp: out bit;
term: out bit;
clk: in bit;
rs: in bit;
vdd: in bit;
vss: in bit
);

end component mysr_sr_bm_bg_lo;

--scan path component
component mysr_sr_bm_bg_lo_sc is

port(
i: in bit;
outp: out bit;
term: out bit;
clk: in bit;
rs: in bit;
vdd: in bit;
vss: in bit;
scanin: in bit;
test: in bit;
```

```
scanout: out bit
);

end component mysr_sr_bm_bg_lo_sc;

--duts
for mysr_dut: mysr use entity work.mysr(mealy_arch);
--adding the new delay simulatoin dut
for mysr_dut_net: mysr_sr_bm_bg_lo use entity work.mysr_sr_bm_bg_lo(structural);
--adding the new scan path dut
for scan_patht:mysr_sr_bm_bg_lo_sc use entity work.mysr_sr_bm_bg_lo_sc(structural);



--declaring input signals and initialize them
signal input: bit := '0';
signal output: bit := '0';
signal termination: bit := '0';
signal clock: bit := '1';
signal reset: bit := '1';
signal vdd: bit := '1';
signal vss: bit := '0';

--new added signals for the delay simulation test
signal output_net: bit := '0';
signal termination_net: bit := '0';
```

```
--new added signals for the scan path test
signal output_sc: bit := '0';
signal termination_sc: bit := '0';
signal test_sc:bit := '0';
signal sc_in: bit := '0';
signal sc_out: bit := '0';



constant clock_period: time := 50 ns;
constant seq: bit_vector := "001010111010010";
constant seq_sc: bit_vector := "0100101011";



begin


--instantiating the duts
mysr_dut:                    mysr                    port
map(input,output,termination,clock,reset,vdd,vss);
mysr_dut_net:             mysr_sr_bm_bg_lo           port
map(input,output_net,termination_net,clock,reset,vdd,vss);
scan_patht:            mysr_sr_bm_bg_lo_sc           port
map(input,output_sc,termination_sc,clock,reset,vdd,vss,sc_
in,test_sc,sc_out);



--clock process with 50% duty cycle
process
begin
```

```
clock <= '1';
wait for clock_period/2;
clock <= '0';
wait for clock_period/2;

end process;



--stimulus process
process
begin

reset <= '0' after clock_period; --reseting the new signals

for x in 0 to (seq'length-1) loop
input <= seq(x);
wait for clock_period;

if(x >= 2) then

if(seq(x) = '0') then

--the next 2 if conditions check for the output
if(seq(x-1) = '1') then

if(seq(x-2) = '0') then
```

```
report "output = 1";
assert output = '1' report "output should be 1 here" severity
error;
end if;

--this else if condition checks for the termination
elsif(seq(x-1) = '0') then

if(seq(x-2) = '1') then
report "termination = 1";
assert termination = '1' report "termination should be 1
here" severity error;
end if;

end if;

end if;

end if;

end loop;

--testing the scan path
--testing the scanin & scan out waves
test_sc <= '1'; --mode switching

--new for loop to test the scan
for z in 0 to (seq_sc'length-1) loop
```

```
sc_in <= seq_sc(z);
wait for clock_period;

if z >= 3 then
Assert sc_out = seq_sc(z-3)
Report "scanout does not follow scan in"
Severity error;
end if;

end loop;

end process;

end test;
```

Makefile:

```
#-------------------- mysr --------------------#

# s for syf
# bm for boom
# bg for boog
# lo for loon
# fb for flatbeh
# sc for scan

syf_all:mysr_sa.vbe \
     mysr_sj.vbe \
     mysr_sm.vbe \
     mysr_so.vbe \
     mysr_sr.vbe
     @echo "--->>> all syf done successfully"

boom_all:mysr_sa_bm.vbe \
     mysr_sj_bm.vbe \
     mysr_sm_bm.vbe \
     mysr_so_bm.vbe \
     mysr_sr_bm.vbe
     @echo "--->>> all boom done successfully"

boog_all:mysr_sa_bm_bg.vst \
     mysr_sj_bm_bg.vst \
```

```
     mysr_sm_bm_bg.vst \
     mysr_so_bm_bg.vst \
     mysr_sr_bm_bg.vst
     @echo "--->>> all boog done successfully"

loon_all:mysr_sa_bm_bg_lo.vst \
     mysr_sj_bm_bg_lo.vst \
     mysr_sm_bm_bg_lo.vst \
     mysr_so_bm_bg_lo.vst \
     mysr_sr_bm_bg_lo.vst
     @echo "--->>> all loon done successfully"

flatbeh:mysr_sr_bm_bg_lo_fb.vbe
     @echo "--->>> flatbeh and proof done successfully"

scan:mysr_sr_bm_bg_lo_sc.vst
     @echo "--->>> scan done successfully"


#-------------------- mysr rules --------------------#

#---------- rename ----------#

vhd_to_fsm:
     rename .vhd .fsm *.vhd
     @echo "--->>> renamed"
```

```
#---------- syf ----------#

mysr_sa.vbe : mysr.fsm
     @echo "--->>> encoding -> a"
     syf -CEV -a mysr mysr_sa > mysr_sa.out

mysr_sj.vbe : mysr.fsm
     @echo "--->>> encoding -> j"
     syf -CEV -j mysr mysr_sj > mysr_sj.out

mysr_sm.vbe : mysr.fsm
     @echo "--->>> encoding -> m"
     syf -CEV -m mysr mysr_sm > mysr_sm.out

mysr_so.vbe : mysr.fsm
     @echo "--->>> encoding -> o"
     syf -CEV -o mysr mysr_so > mysr_so.out

mysr_sr.vbe : mysr.fsm
     @echo "--->>> encoding -> r"
     syf -CEV -r mysr mysr_sr > mysr_sr.out

#---------- boom ----------#

%_bm.vbe:%.vbe
     @echo "--->>> boolean minimization -> $*_bm"
     boom -V -d 50 $* $*_bm > $*_bm.out
```

```
#---------- boog ----------#

%_bg.vst:%.vbe paramfile.lax
     @echo "--->>> logical synthesis -> $*_bg"
     boog -x 1 -l paramfile $* $*_bg > $*_bg.out

#---------- loon ----------#

%_lo.vst:%.vst paramfile.lax
     @echo "--->>> netlist optimization -> $*_lo"
     loon -x 1 -l paramfile $* $*_lo > $*_lo.out

#---------- flatbeh ----------#

%_bm_bg_lo_fb.vbe:%_bm_bg_lo.vst %.vbe
     @echo "--->>> formal checking -> $*"
     flatbeh       $*_bm_bg_lo       $*_bm_bg_lo_fb       >
$*_bm_bg_lo_fb.out
     proof -d $* $*_bm_bg_lo_fb > $*_proof.out

#---------- ac_scapin_registers ----------#

ac_scapin_registers:
     cat mysr_sr_bm_bg_lo.vst | grep sff

#---------- scan ----------#
```

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

#-------------------- clear --------------------#

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

Paramfile.lax:

```
## Optimization Mode
#M{2}

## Optimization Level
#L{2}

## External Output Capacitance (in fF)
#C{
outp:100;
term:100;
}
```

Scan.path:

```
BEGIN_PATH_REG
mysr_cs_0_ins
mysr_cs_1_ins
mysr_cs_2_ins
END_PATH_REG

BEGIN_CONNECTOR
SCAN_IN scanin
SCAN_OUT scanout
SCAN_TEST test
END_CONNECTOR
```