

AIN SHAMS UNIVERSITY FACULTY OF ENGINEERING

Project 2 Report

Name: Hazem Hamada Abdellatif Mohamed.

ID: 16p3100.

Group1, Section1.

Email: [zomahamada.hh@gmail.com](mailto:zomahamada.hh@gmail.com).

Program: Computer Engineering and Software Systems.

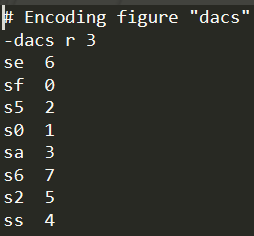
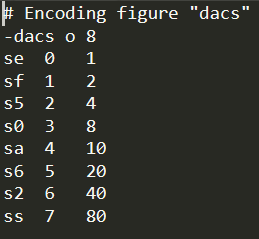
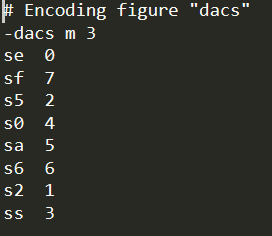
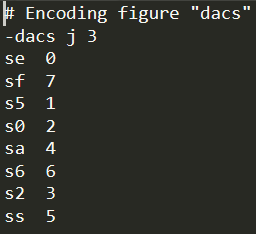
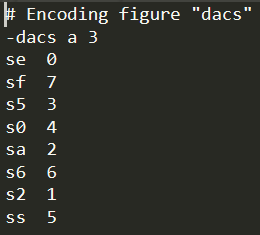
In the previous part of the project, we obtained a golden FSM for digital access control system description (High-Level Design) and developed a test bench for verification. In this part of the project we will complete the Low-Level Synthesis and Design for test (DFT) using the alliance tool.

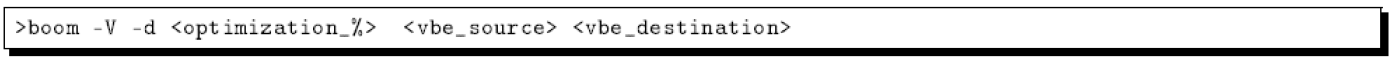
The **ALLIANCE** tools used are:

* **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.

First of all we use the syf tool for state encoding to our behavioural model 



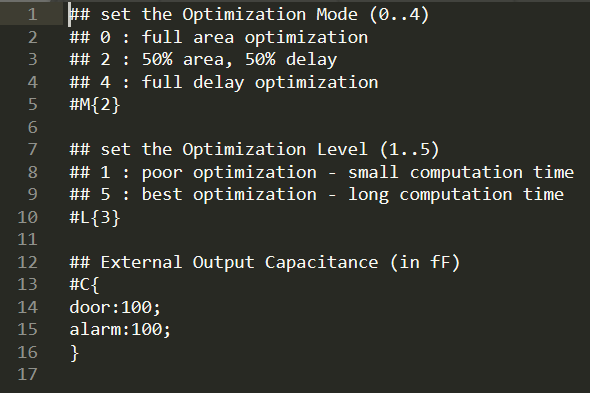
After the syf tool we used the boom tool for Boolean optimization for each encoding case which gives different structural models. 

Number of literals:

|  |  |  |
| --- | --- | --- |
| State encoding | Number of literals in initial cost | Number of literals in final cost |
| Sdeta | 133 | 74 |
| Sdetj | 133 | 73 |
| Sdetm | 133 | 76 |
| Sdeto | 142 | 90 |
| sdetr | 133 | 92 |

After that we lunch the boog tool for logical synthesizing for the .vst files and choosing the optimization parameters in the paramfile, we launch **BOOG** on different *netlist*s to observe **SYF** options influence (different state encoding techniques). 

Param file:



After the boog tool we lunch the loon tool for optimizing the gate level design for all the state encoding cases and then choosing only one state encoding to complete the rest of the project with it: 

For the sdeta encoding case:

|  |  |  |  |
| --- | --- | --- | --- |
| Sdeta\_b (before loon) | | Sdeta\_b\_l (after loon) | |
| inv\_x2: 8 (9%) | | inv\_x2: 7 (7%) | |
| na2\_x1: 6 (9%) | | na2\_x1: 6 (8%) | |
| o3\_x2: 4 (9%) | | o3\_x2: 4 (8%) | |
| no3\_x1: 4 (7%) | | no3\_x1: 4 (7%) | |
| na3\_x1: 3 (5%) | | na3\_x1: 3 (5%) | |
| sff1\_x4: 3 (21%) | | sff1\_x4: 3 (19%) | |
| nao22\_x1: 3 (7%) | | nao22\_x1: 3 (6%) | |
| no2\_x1: 3 (4%) | | no2\_x1: 3 (4%) | |
| a2\_x2: 3 (5%) | | a2\_x2: 3 (5%) | |
| on12\_x1: 2 (3%) | | buf\_x2: 2 (2%) | |
| no4\_x1: 2 (4%) | | on12\_x1: 2 (3%) | |
| oa22\_x2: 1 (2%) | | buf\_x4: 1 (1%) | |
| nmx2\_x1: 1 (2%) | | no4\_x4: 1 (3%) | |
| ao22\_x2: 1 (2%) | | oa22\_x2: 1 (2%) | |
| an12\_x1: 1 (1%) | | nmx2\_x1: 1 (2%) | |
| o2\_x2: 1 (1%) | | ao22\_x2: 1 (2%) | |
| total | 46 | total | 49 |
| Delay=2387ps | Area=64250 | Delay=2237ps | Area=68750 |

For the sdetj encoding case:

|  |  |  |  |
| --- | --- | --- | --- |
| Sdetj\_b (before loon) | | Sdetj\_b\_l (after loon) | |
| inv\_x2: 7 (8%) | | inv\_x2: 7 (8%) | |
| na3\_x1: 6 (11%) | | na3\_x1: 6 (11%) | |
| o2\_x2: 5 (9%) | | o2\_x2: 5 (9%) | |
| na2\_x1: 4 (6%) | | na2\_x1: 4 (6%) | |
| sff1\_x4: 3 (21%) | | sff1\_x4: 3 (21%) | |
| on12\_x1: 3 (5%) | | on12\_x1: 3 (5%) | |
| o3\_x2: 3 (7%) | | o3\_x2: 3 (7%) | |
| nao22\_x1: 2 (4%) | | nao22\_x1: 2 (4%) | |
| na4\_x1: 1 (2%) | | na4\_x1: 1 (2%) | |
| no3\_x1: 1 (1%) | | no3\_x1: 1 (1%) | |
| o4\_x2: 1 (2%) | | o4\_x2: 1 (2%) | |
| a2\_x2: 1 (1%) | | a2\_x2: 1 (1%) | |
| no2\_x1: 1 (1%) | | no2\_x1: 1 (1%) | |
| ao22\_x2: 1 (2%) | | ao22\_x2: 1 (2%) | |
| xr2\_x1: 1 (3%) | | xr2\_x1: 1 (3%) | |
| oa22\_x2: 1 (2%) | | oa22\_x2: 1 (2%) | |
| total | 44 | total | 44 |
| Delay=2042ps | Area=64250 | Delay=2042ps | Area=64250 |

For the sdetm encoding case:

|  |  |  |  |
| --- | --- | --- | --- |
| Sdetm\_b (before loon) | | Sdetm\_b\_l (after loon) | |
| inv\_x2: 12 (13%) | | inv\_x2: 11 (12%) | |
| na3\_x1: 7 (13%) | | na3\_x1: 7 (13%) | |
| o2\_x2: 5 (9%) | | o2\_x2: 5 (9%) | |
| na4\_x1: 3 (6%) | | na4\_x1: 3 (6%) | |
| no3\_x1: 3 (5%) | | no3\_x1: 3 (5%) | |
| sff1\_x4: 3 (20%) | | sff1\_x4: 3 (20%) | |
| on12\_x1: 2 (3%) | | on12\_x1: 2 (3%) | |
| nao22\_x1: 2 (4%) | | nao22\_x1: 2 (4%) | |
| ao22\_x2: 2 (4%) | | ao22\_x2: 2 (4%) | |
| no4\_x1: 2 (4%) | | no4\_x1: 2 (4%) | |
| na2\_x1: 1 (1%) | | buf\_x2: 1 (1%) | |
| nxr2\_x1: 1 (3%) | | inv\_x1: 1 (1%) | |
| no2\_x1: 1 (1%) | | na2\_x1: 1 (1%) | |
| o3\_x2: 1 (2%) | | nxr2\_x1: 1 (3%) | |
| an12\_x1: 1 (1%) | | no2\_x1: 1 (1%) | |
| oa22\_x2: 1 (2%) | | o3\_x2: 1 (2%) | |
| total | 47 | total | 48 |
| Delay=1953ps | Area=65750 | Delay=1936ps | Area=66750 |

For the sdeto encoding case:

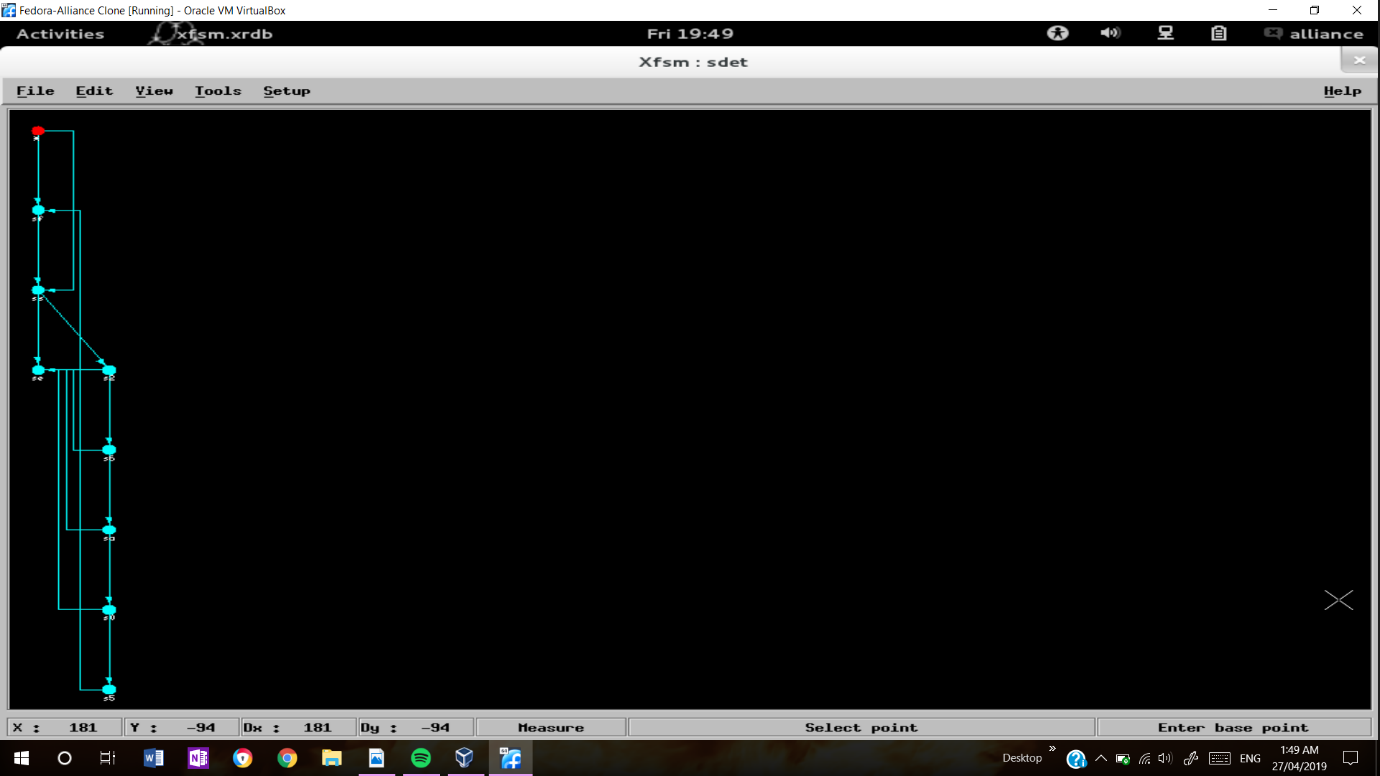
|  |  |  |  |
| --- | --- | --- | --- |
| Sdeto\_b (before loon) | | Sdeto\_b\_l (after loon) | |
| inv\_x2: 10 (8%) | | inv\_x2: 10 (8%) | |
| sff1\_x4: 8 (39%) | | sff1\_x4: 8 (39%) | |
| no3\_x1: 6 (8%) | | no3\_x1: 6 (8%) | |
| no2\_x1: 6 (6%) | | no2\_x1: 6 (6%) | |
| a2\_x2: 5 (6%) | | a2\_x2: 5 (6%) | |
| na4\_x1: 5 (8%) | | na4\_x1: 5 (8%) | |
| na2\_x1: 5 (5%) | | na2\_x1: 5 (5%) | |
| ao22\_x2: 2 (3%) | | ao22\_x2: 2 (3%) | |
| oa22\_x2: 1 (1%) | | buf\_x2: 1 (1%) | |
| oa2ao222\_x2: 1 (2%) | | oa22\_x2: 1 (1%) | |
| nao2o22\_x1: 1 (1%) | | oa2ao222\_x2: 1 (2%) | |
| o4\_x2: 1 (1%) | | nao2o22\_x1: 1 (1%) | |
| noa22\_x1: 1 (1%) | | o4\_x2: 1 (1%) | |
| a4\_x2: 1 (1%) | | noa22\_x1: 1 (1%) | |
| on12\_x1: 1 (1%) | | a4\_x2: 1 (1%) | |
| inv\_x2: 10 (8%) | | on12\_x1: 1 (1%) | |
| total | 54 | total | 55 |
| Delay=2661ps | Area=90750 | Delay=2603ps | Area=91750 |

For the sdetr encoding case:

|  |  |  |  |
| --- | --- | --- | --- |
| Sdetr\_b (before loon) | | Sdetr\_b\_l (after loon) | |
| inv\_x2: 8 (7%) | | inv\_x2: 7 (6%) | |
| na2\_x1: 6 (7%) | | na2\_x1: 6 (7%) | |
| o2\_x2: 5 (8%) | | o2\_x2: 5 (7%) | |
| no4\_x1: 4 (7%) | | no4\_x1: 4 (7%) | |
| no2\_x1: 4 (5%) | | no2\_x1: 4 (5%) | |
| nao22\_x1: 4 (7%) | | nao22\_x1: 4 (7%) | |
| na3\_x1: 3 (4%) | | na3\_x1: 3 (4%) | |
| no3\_x1: 3 (4%) | | no3\_x1: 3 (4%) | |
| sff1\_x4: 3 (17%) | | sff1\_x4: 3 (17%) | |
| a2\_x2: 3 (4%) | | a2\_x2: 3 (4%) | |
| oa22\_x2: 3 (5%) | | oa22\_x2: 3 (5%) | |
| noa22\_x1: 2 (3%) | | noa22\_x1: 2 (3%) | |
| on12\_x1: 2 (3%) | | on12\_x1: 2 (3%) | |
| a4\_x2: 2 (4%) | | a4\_x2: 2 (4%) | |
| noa2ao222\_x1: 1 (2%) | | buf\_x2: 1 (1%) | |
| xr2\_x1: 1 (2%) | | noa2ao222\_x1: 1 (2%) | |
| total | 55 | total | 56 |
| Delay=2545ps | Area=77750 | Delay=2497ps | Area=79000 |

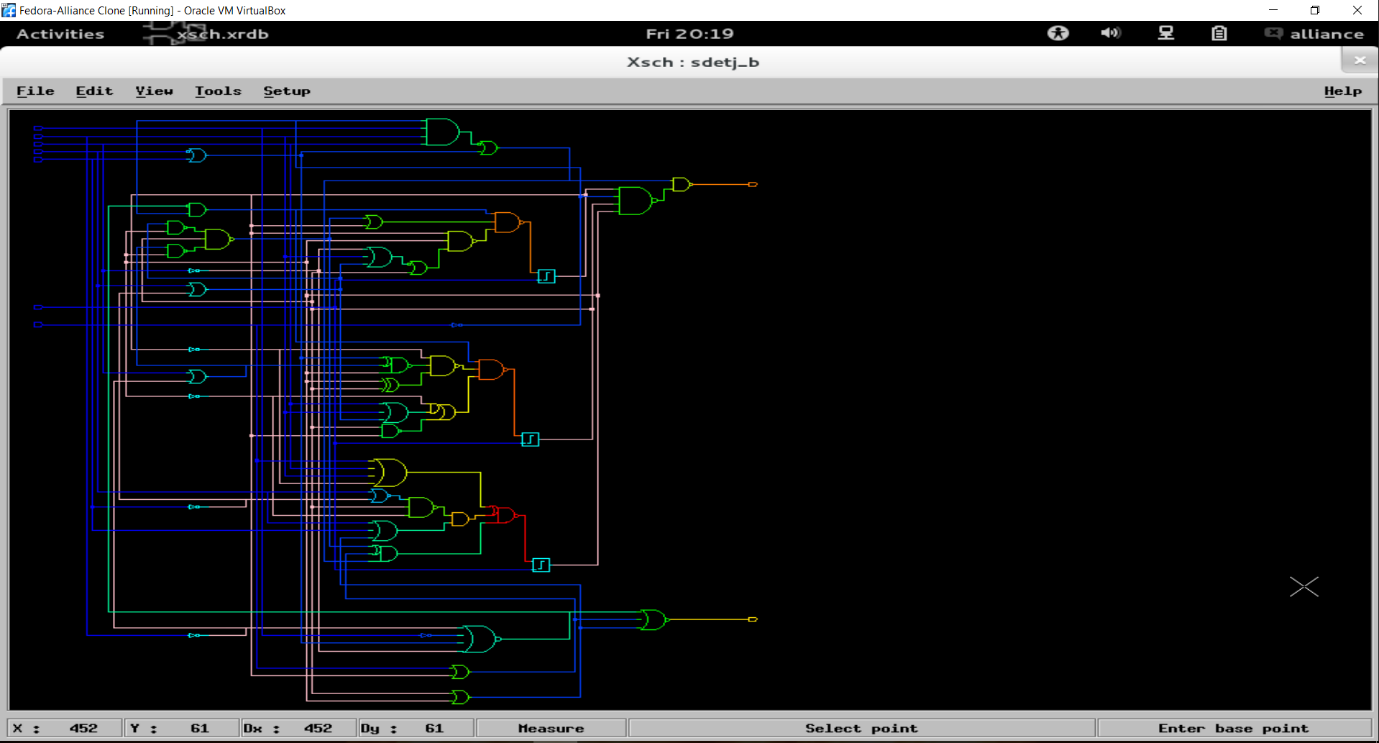
We would choose the ***sdetj*** implementation as is has the minimum number of components, so it has the minimum number of components, area, and delay.

We will use the xfsm tool to visualize the state diagram of the circuit: 

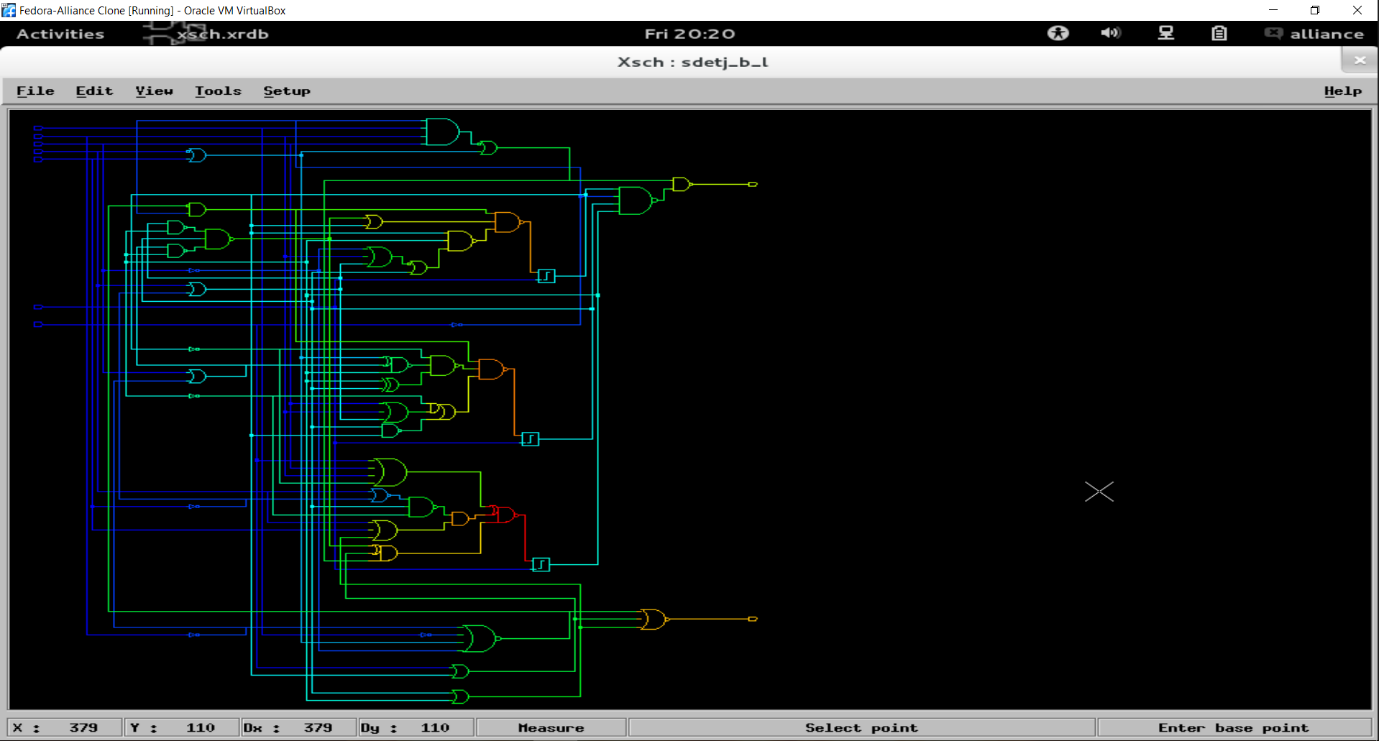


And then use the xsch tool to visualize the circuit it self:

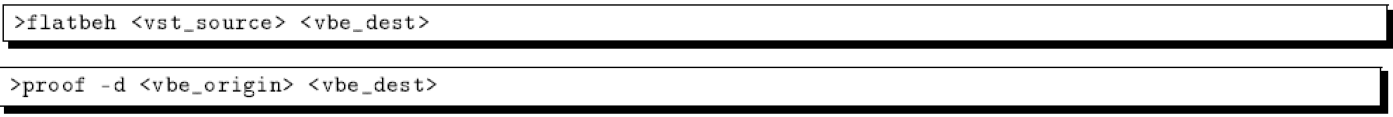
Before the loon:



After the loon:

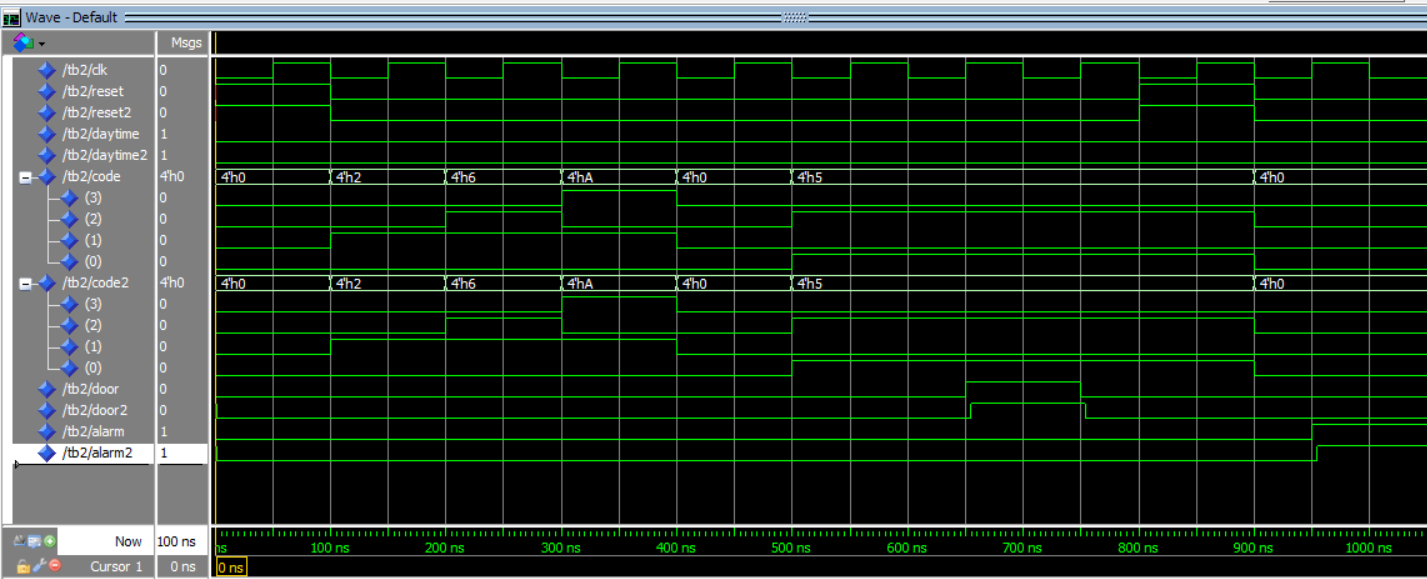
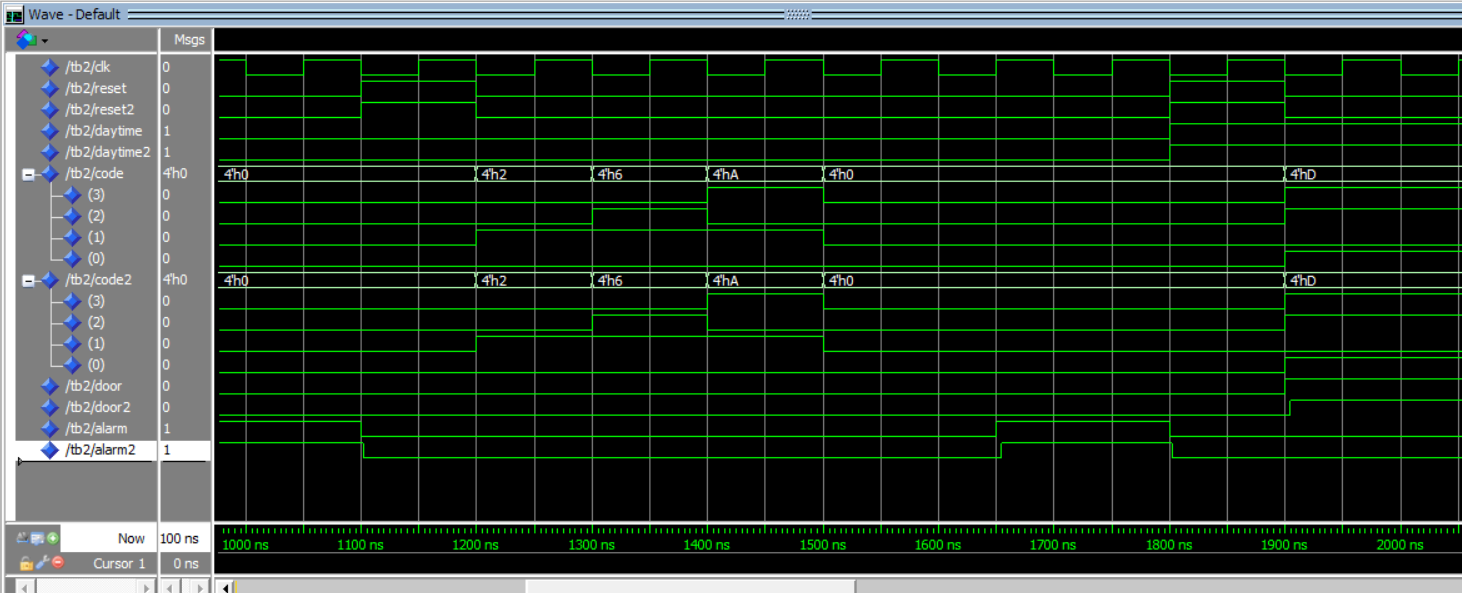
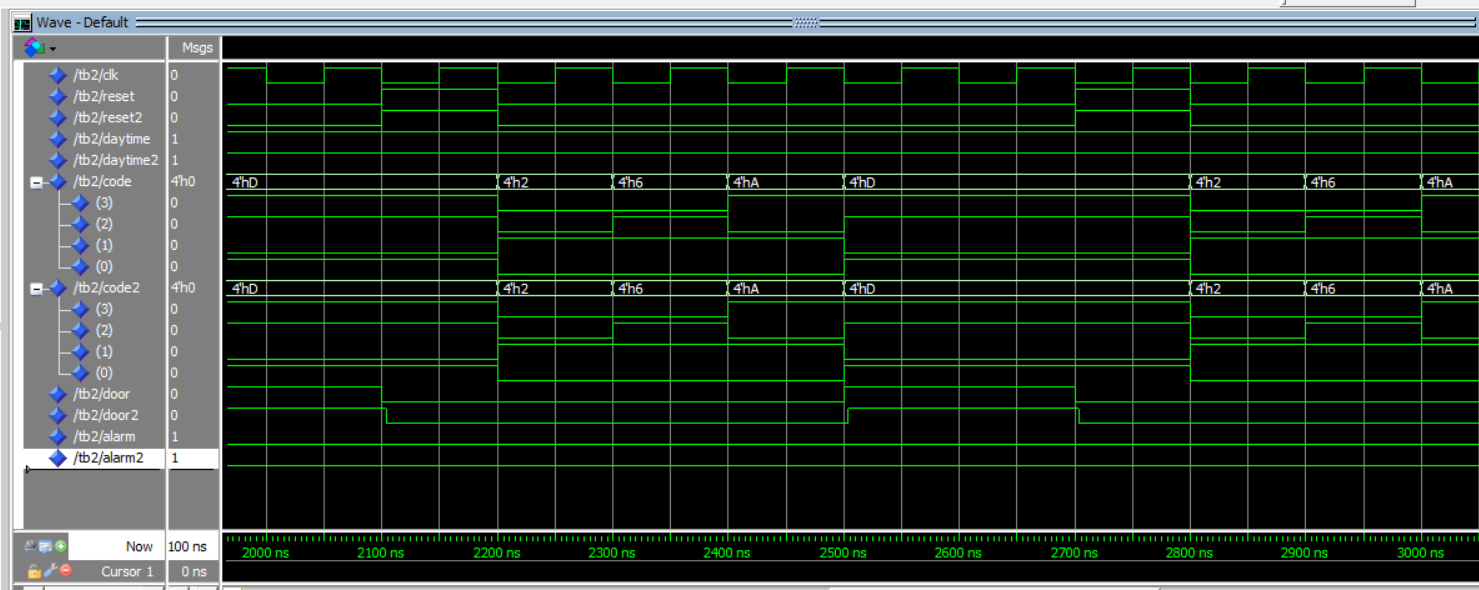
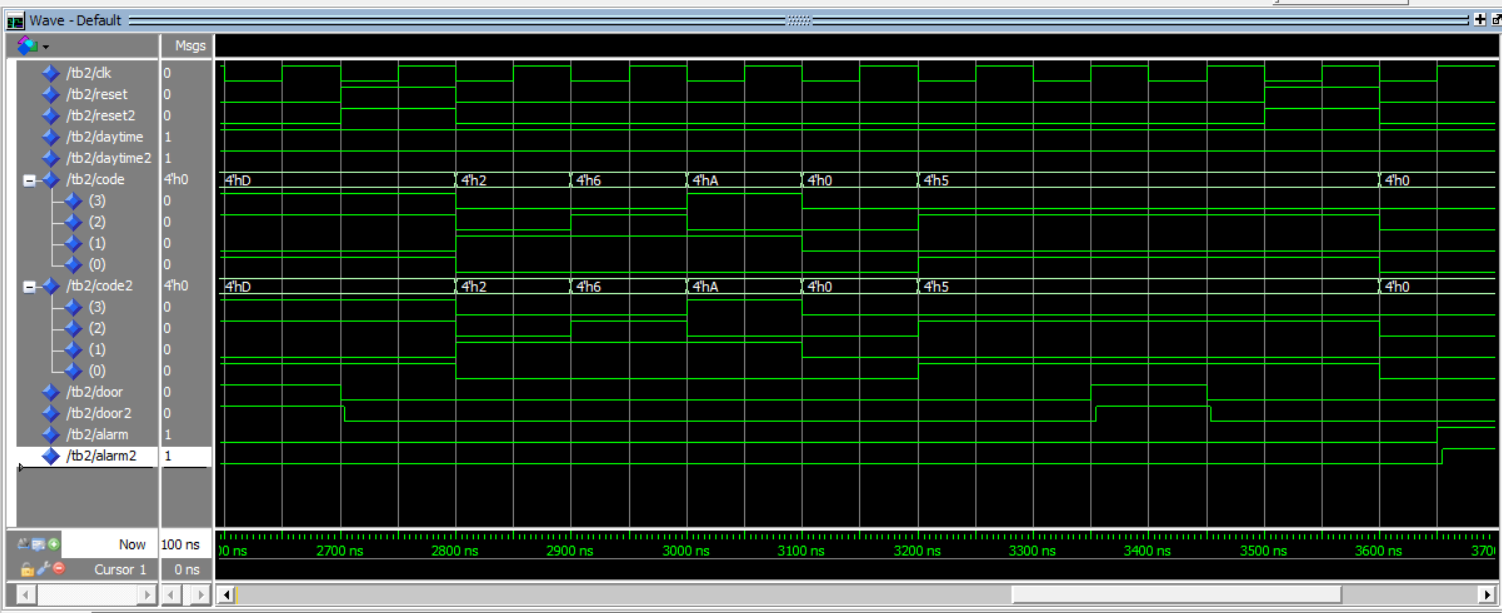


And then Make a formal comparison of our netlist with the original behavioural file resulting from **SYF** using the proof and flatbeh tools.



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 double-check the speed performance.

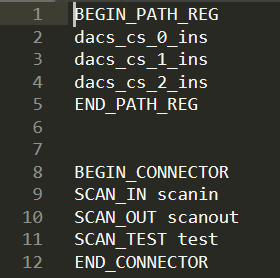
Validate the synthesis results using **ModelSim** using the same test bench and assertions which was used to validate the initial behavioural FSM

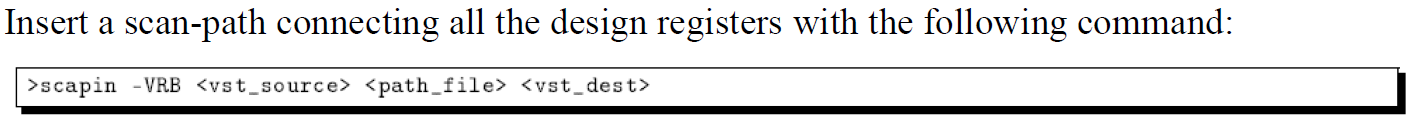
As shown in the wave form snap shot that there is some delay between the behavioural output (door ,alarm) and the actual structural output(door2, alarm2) this is the delay of the components in the circuit.

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.

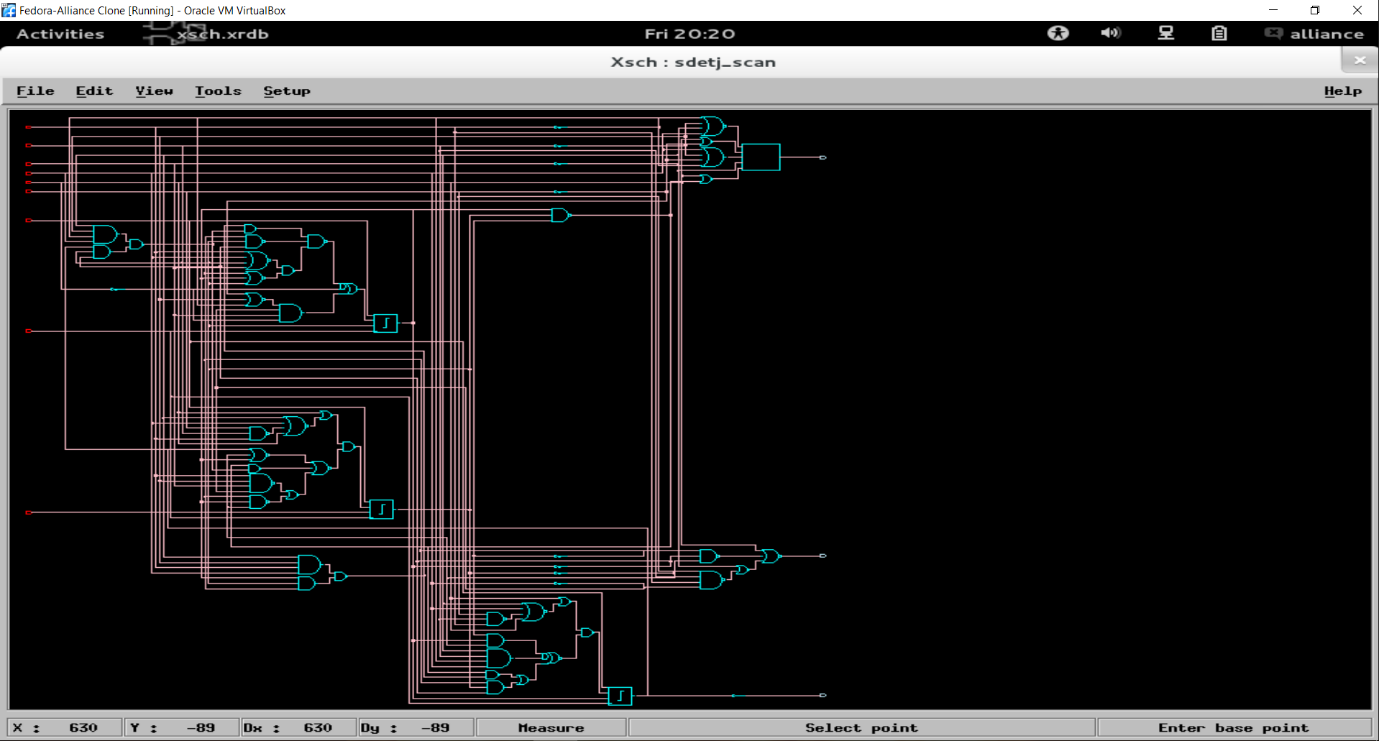
Prepare a “.path” file. Open the gate-level netlist file obtained from synthesis “.vst”, search for registers. Register component name in Alliance standard cell library is “sff1”. Then put them in the “.path” file as follows :



where dasc\_cs\_0\_ins, dacs\_cs\_1\_ins and dacs\_cs\_2\_ins are the registers in the input netlist (.vst) file that will be placed in the scan-path.

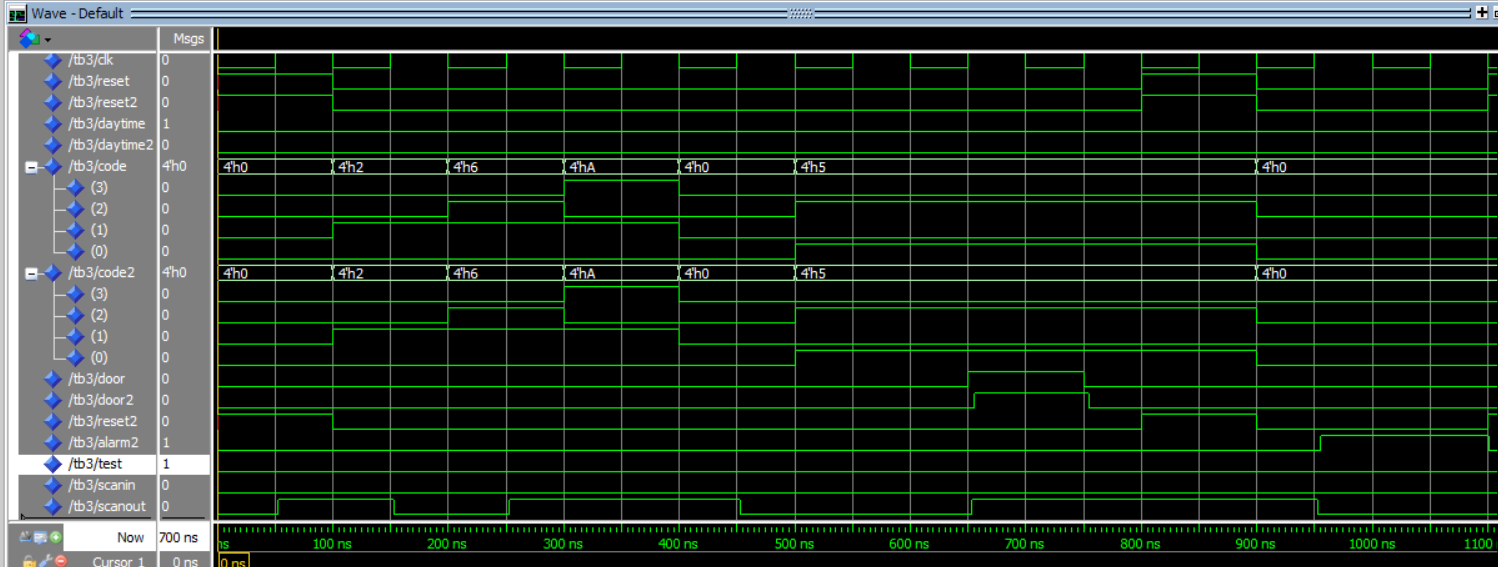
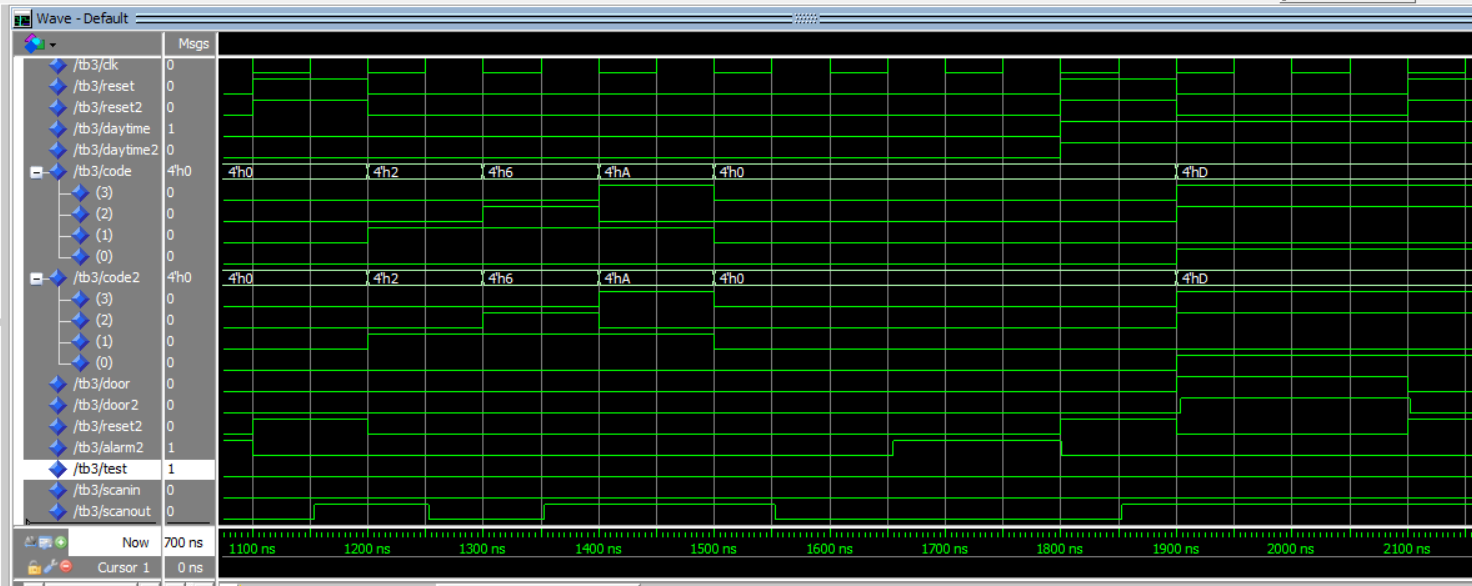
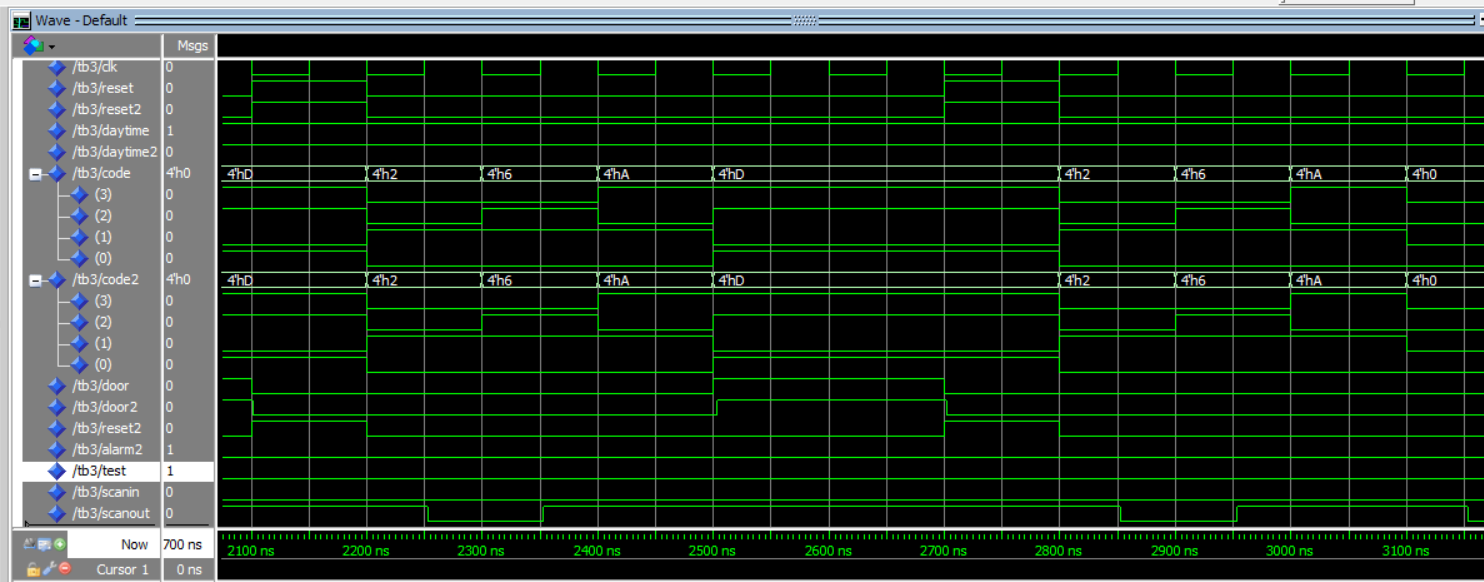


Then use xsch tool to visualize the resulting netlist:

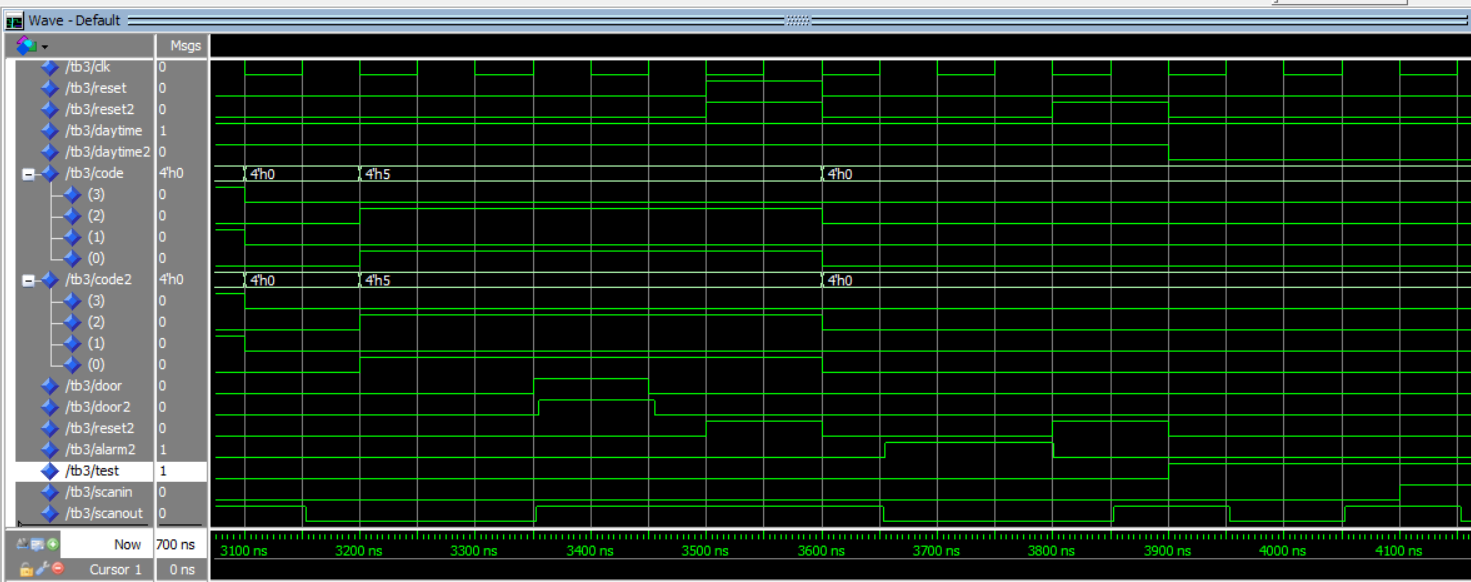
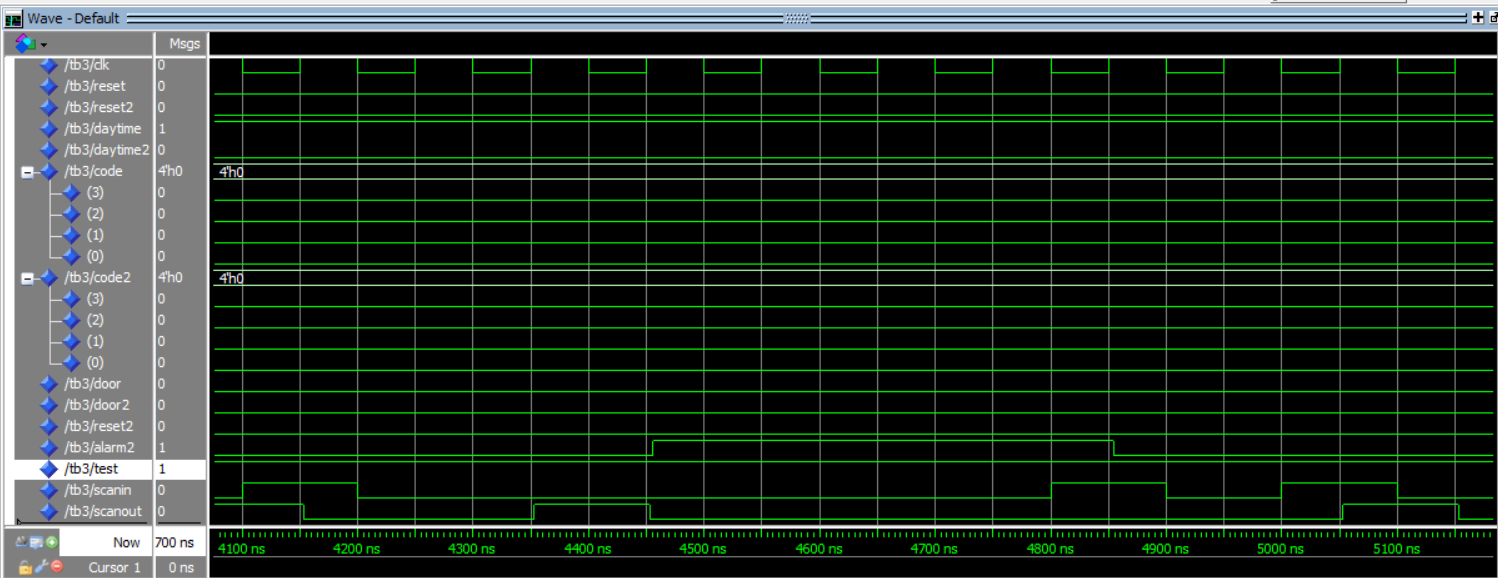
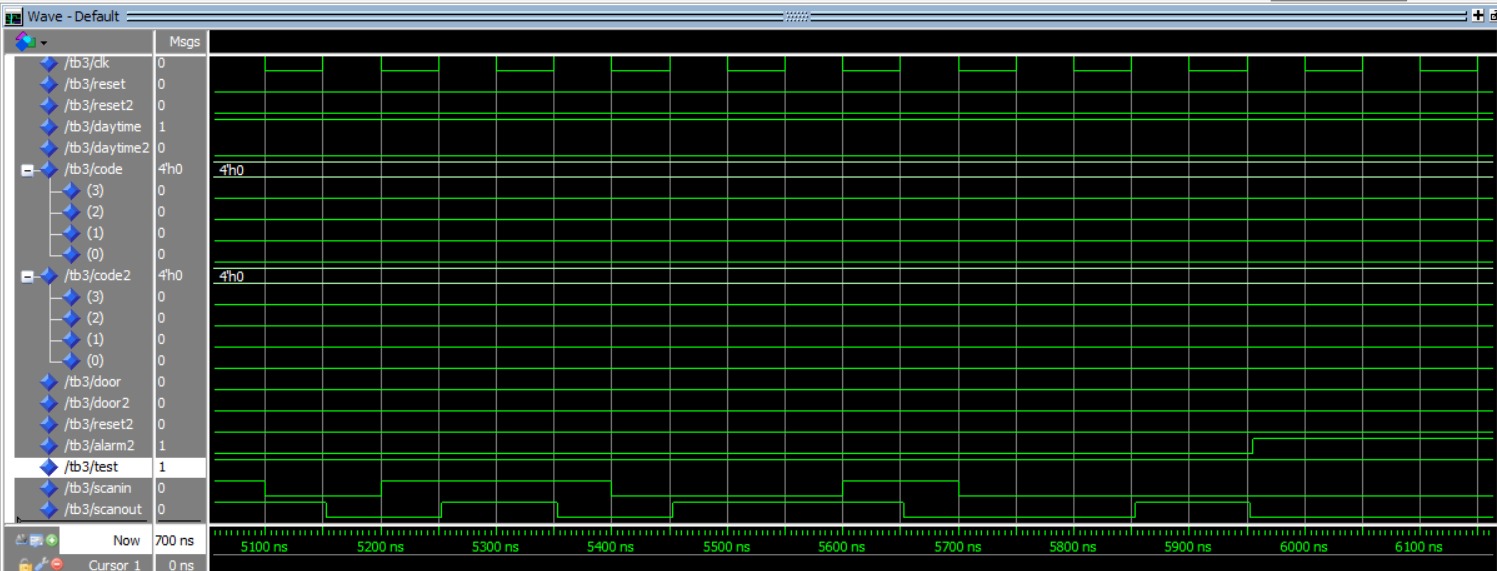


Extend the original testbench to test the scan-path and simulate with **ModelSim**.

Test=0:

Test=1:

***Appendix:***

***Delay test bench code:***

LIBRARY sxlib\_ModelSim;

entity tb2 is

end entity tb2;

architecture test of tb2 is

component dacs is

port( vdd,clk,vss,reset,daytime: in bit;

code: in bit\_vector(3 downto 0);

door,alarm: out bit );

end component dacs;

component sdetj\_b\_l is

port (

vdd : in bit;

clk : in bit;

vss : in bit;

reset : in bit;

daytime : in bit;

code : in bit\_vector(3 downto 0);

door : out bit;

alarm : out bit

);

end component sdetj\_b\_l;

signal vdd,vss,clk,reset,daytime,door,alarm:bit;

signal code:bit\_vector(3 downto 0);

signal reset2,daytime2,door2,alarm2:bit;

signal code2:bit\_vector(3 downto 0);

constant clk\_period : time := 100ns;

for dut:dacs use entity work.dacs(behav);

for dut2:sdetj\_b\_l use entity work.sdetj\_b\_l(structural);

begin

dut:dacs port map(vdd,clk,vss,reset,daytime,code,door,alarm);

dut2:sdetj\_b\_l port map(vdd,clk,vss,reset2,daytime2,code2,door2,alarm2);

clk\_process :process

begin

clk <= '0';

wait for clk\_period/2;

clk <= '1';

wait for clk\_period/2;

end process;

p:process

begin

--happy scenario test.

reset<='1';

reset2<='1';

wait for clk\_period;

reset<='0';

daytime<='0';

code<="0010";

reset2<='0';

daytime2<='0';

code2<="0010";

wait for clk\_period;

code<="0110";

code2<="0110";

wait for clk\_period;

code<="1010";

code2<="1010";

wait for clk\_period;

code<="0000";

code2<="0000";

wait for clk\_period;

code<="0101";

code2<="0101";

wait for 2\*clk\_period;

assert door=door2 and alarm=alarm2

report "happy scenario error"

severity error;

wait for clk\_period;

--alarm scenario 1 test.

reset<='1';

reset2<='1';

wait for clk\_period;

reset<='0';

code<="0000";

reset2<='0';

code2<="0000";

wait for clk\_period;

assert door=door2 and alarm=alarm2

report "alarm scenario 1 error"

severity error;

wait for clk\_period;

--alarm scenario 2 test.

reset<='1';

reset2<='1';

wait for clk\_period;

reset<='0';

code<="0010";

reset2<='0';

code2<="0010";

wait for clk\_period;

code<="0110";

code2<="0110";

wait for clk\_period;

code<="1010";

code2<="1010";

wait for clk\_period;

code<="0000";

code2<="0000";

wait for clk\_period;

code<="0000";

code2<="0000";

wait for clk\_period;

assert door=door2 and alarm=alarm2

report "alarm scenario 2 error"

severity error;

wait for clk\_period;

--daytime happy scenario 1 test.

reset<='1';

daytime<='1';

reset2<='1';

daytime2<='1';

wait for clk\_period;

reset<='0';

code<="1101";

reset2<='0';

code2<="1101";

wait for clk\_period;

assert door=door2 and alarm=alarm2

report "daytime happy scenario 1 error"

severity error;

wait for clk\_period;

--daytime happy scenario 2 test.

reset<='1';

reset2<='1';

wait for clk\_period;

reset<='0';

code<="0010";

reset2<='0';

code2<="0010";

wait for clk\_period;

code<="0110";

code2<="0110";

wait for clk\_period;

code<="1010";

code2<="1010";

wait for clk\_period;

code<="1101";

code2<="1101";

wait for clk\_period;

assert door=door2 and alarm=alarm2

report "daytime happy scenario 2 error"

severity error;

wait for clk\_period;

--daytime happy scenario 3 test.

reset<='1';

reset2<='1';

wait for clk\_period;

reset<='0';

code<="0010";

reset2<='0';

code2<="0010";

wait for clk\_period;

code<="0110";

code2<="0110";

wait for clk\_period;

code<="1010";

code2<="1010";

wait for clk\_period;

code<="0000";

code2<="0000";

wait for clk\_period;

code<="0101";

code2<="0101";

wait for 2\*clk\_period;

assert door=door2 and alarm=alarm2

report "daytime happy scenario 3 error"

severity error;

wait for clk\_period;

--alarm scenario 3 test.

reset<='1';

daytime<='1';

reset2<='1';

daytime2<='1';

wait for clk\_period;

reset<='0';

code<="0000";

reset2<='0';

code2<="0000";

wait for clk\_period;

assert door=door2 and alarm=alarm2

report "alarm scenario 3 error"

severity error;

wait for clk\_period;

wait;

end process p;

end architecture test;

***Scan test bench:***

LIBRARY sxlib\_ModelSim;

entity tb3 is

end entity tb3;

architecture test of tb3 is

component dacs is

port( vdd : in bit;

clk : in bit;

vss : in bit;

reset : in bit;

daytime : in bit;

code : in bit\_vector(3 downto 0);

door : out bit;

alarm : out bit

);

end component dacs;

component sdetj\_scan is

port (

vdd : in bit;

clk : in bit;

vss : in bit;

reset : in bit;

daytime : in bit;

code : in bit\_vector(3 downto 0);

door : out bit;

alarm : out bit;

scanin : in bit;

test : in bit;

scanout : out bit

);

end component sdetj\_scan;

signal sequence : bit\_vector(19 downto 0);

signal vdd,vss,clk,reset,daytime,door,alarm:bit;

signal code:bit\_vector(3 downto 0);

signal reset2,daytime2,door2,alarm2,scanin,test,scanout:bit;

signal code2:bit\_vector(3 downto 0);

constant clk\_period : time := 100ns;

for dut:dacs use entity work.dacs(behav);

for dut2:sdetj\_scan use entity work.sdetj\_scan(structural);

begin

dut:dacs port map(vdd,clk,vss,reset,daytime,code,door,alarm);

dut2:sdetj\_scan port map(vdd,clk,vss,reset2,daytime2,code2,door2,alarm2,scanin,test,scanout);

clk\_process :process

begin

clk <= '0';

wait for clk\_period/2;

clk <= '1';

wait for clk\_period/2;

end process;

p:process

begin

-------------test off

test<='0';

--happy scenario test.

reset<='1';

reset2<='1';

wait for clk\_period;

reset<='0';

daytime<='0';

code<="0010";

reset2<='0';

daytime2<='0';

code2<="0010";

wait for clk\_period;

code<="0110";

code2<="0110";

wait for clk\_period;

code<="1010";

code2<="1010";

wait for clk\_period;

code<="0000";

code2<="0000";

wait for clk\_period;

code<="0101";

code2<="0101";

wait for 2\*clk\_period;

assert door=door2 and alarm=alarm2

report "happy scenario error"

severity error;

wait for clk\_period;

--alarm scenario 1 test.

reset<='1';

reset2<='1';

wait for clk\_period;

reset<='0';

code<="0000";

reset2<='0';

code2<="0000";

wait for clk\_period;

assert door=door2 and alarm=alarm2

report "alarm scenario 1 error"

severity error;

wait for clk\_period;

--alarm scenario 2 test.

reset<='1';

reset2<='1';

wait for clk\_period;

reset<='0';

code<="0010";

reset2<='0';

code2<="0010";

wait for clk\_period;

code<="0110";

code2<="0110";

wait for clk\_period;

code<="1010";

code2<="1010";

wait for clk\_period;

code<="0000";

code2<="0000";

wait for clk\_period;

code<="0000";

code2<="0000";

wait for clk\_period;

assert door=door2 and alarm=alarm2

report "alarm scenario 2 error"

severity error;

wait for clk\_period;

--daytime happy scenario 1 test.

reset<='1';

daytime<='1';

reset2<='1';

daytime2<='1';

wait for clk\_period;

reset<='0';

code<="1101";

reset2<='0';

code2<="1101";

wait for clk\_period;

assert door=door2 and alarm=alarm2

report "daytime happy scenario 1 error"

severity error;

wait for clk\_period;

--daytime happy scenario 2 test.

reset<='1';

reset2<='1';

wait for clk\_period;

reset<='0';

code<="0010";

reset2<='0';

code2<="0010";

wait for clk\_period;

code<="0110";

code2<="0110";

wait for clk\_period;

code<="1010";

code2<="1010";

wait for clk\_period;

code<="1101";

code2<="1101";

wait for clk\_period;

assert door=door2 and alarm=alarm2

report "daytime happy scenario 2 error"

severity error;

wait for clk\_period;

--daytime happy scenario 3 test.

reset<='1';

reset2<='1';

wait for clk\_period;

reset<='0';

code<="0010";

reset2<='0';

code2<="0010";

wait for clk\_period;

code<="0110";

code2<="0110";

wait for clk\_period;

code<="1010";

code2<="1010";

wait for clk\_period;

code<="0000";

code2<="0000";

wait for clk\_period;

code<="0101";

code2<="0101";

wait for 2\*clk\_period;

assert door=door2 and alarm=alarm2

report "daytime happy scenario 3 error"

severity error;

wait for clk\_period;

--alarm scenario 3 test.

reset<='1';

daytime<='1';

reset2<='1';

daytime2<='1';

wait for clk\_period;

reset<='0';

code<="0000";

reset2<='0';

code2<="0000";

wait for clk\_period;

assert door=door2 and alarm=alarm2

report "alarm scenario 3 error"

severity error;

wait for clk\_period;

-------------test on

reset2<='1';

wait for clk\_period;

reset2<='0';

daytime2<='0';

sequence<="00100110101000000101";

test<='1';

for i in 0 to sequence'length-1 loop

scanin<=sequence(i);

wait for clk\_period;

if i>=3 then

assert scanout=sequence(i-2)

report "scanout doesn't follow scanin"

severity error;

end if;

end loop;

wait;

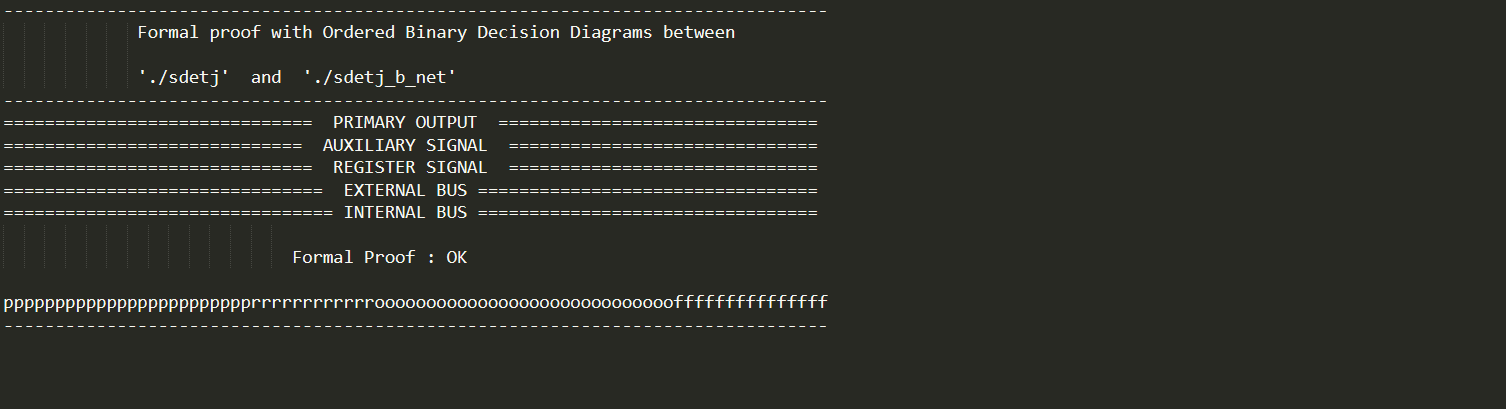
end process p;

end architecture test;

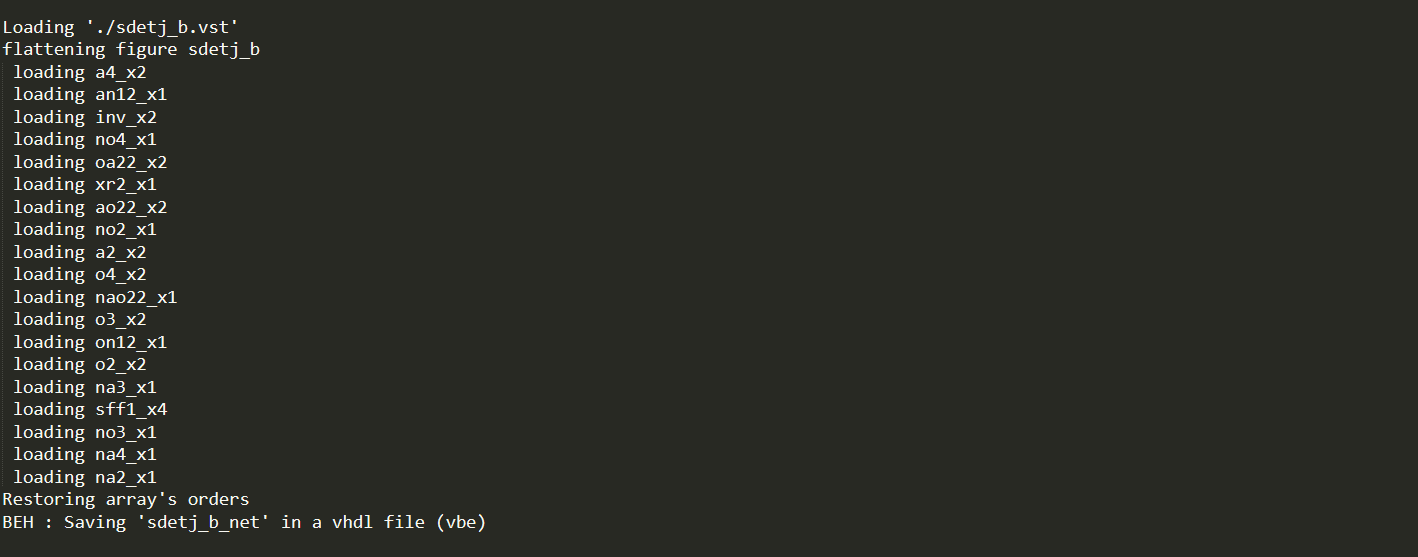
***Scan output:***

******

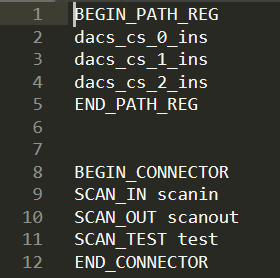
***Proof output:***

****** ******

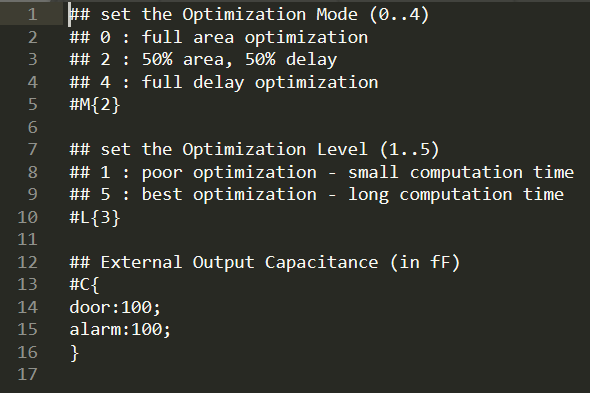
***Flatbeh output:***

****** ******

***.path file:***



***Paramfile:***



***Makefile:***

vhd\_to\_fsm:

rename .vhd .fsm \*.vhd

sdeta.vbe: sdet.fsm

@echo "Encoding -a -> $@"

syf -CEV -a sdet

sdeto.vbe: sdet.fsm

@echo "Encoding -o -> $@"

syf -CEV -o sdet

sdetj.vbe: sdet.fsm

@echo "Encoding -j -> $@"

syf -CEV -j sdet

sdetr.vbe: sdet.fsm

@echo "Encoding -r -> $@"

syf -CEV -r sdet

sdetm.vbe: sdet.fsm

@echo "Encoding -m -> $@"

syf -CEV -m sdet

all: sdeta.vbe\

sdeto.vbe\

sdetr.vbe\

sdetj.vbe\

sdetm.vbe

sdet\_boom: sdeta\_b.vbe\

sdeto\_b.vbe\

sdetr\_b.vbe\

sdetj\_b.vbe\

sdetm\_b.vbe

%\_b.vbe: %.vbe

@echo "Boolean Optimization -> $@"

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

sdet\_boog: sdeta\_b.vst\

sdeto\_b.vst\

sdetr\_b.vst\

sdetj\_b.vst\

sdetm\_b.vst

%.vst: %.vbe paramfile.lax

@echo " Logical Synthesis -> $@"

boog -x 1 -l paramfile $\* > $\*\_boog.out

sdet\_loon: sdeta\_b\_l.vst\

sdeto\_b\_l.vst\

sdetr\_b\_l.vst\

sdetj\_b\_l.vst\

sdetm\_b\_l.vst

%\_l.vst: %.vst paramfile.lax

@echo " Netlist Optimization -> $@"

loon -x 1 $\* $\*\_l paramfile > $\*\_loon.out

sdet\_proof : sdeta\_b\_net.vbe sdetj\_b\_net.vbe sdetm\_b\_net.vbe \

sdeto\_b\_net.vbe sdetr\_b\_net.vbe

%\_b\_net.vbe : %\_b.vst %.vbe

@echo " Formal checking -> $@ "

flatbeh $\*\_b $\*\_b\_net > $\*\_flatbeh.out

proof -d $\* $\*\_b\_net > $\*\_proof.out

%\_scan.vst : %.vst scan.path

@echo " scan-path insertion -> $@ "

scapin -VRB $\* scan $\*\_scan > scapin.out

clean:

rm -f \*.vbe \*.enc \*~