AIN SHAMS UNIVERSITY

FACULTY OF ENGINEERING

Project 2 Report

Name: Hazem Hamada Abdellatif Mohamed.

ID: 16p3100.

Group1, Section1.

Email: 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

```
> syf -CEV -a <fsm_source>
```

```
# Encoding figure "dacs"
-dacs a 3
se  0
sf  7
s5  3
s0  4
sa  2
s6  6
s2  1
ss  5
```

```
# Encoding figure "dacs"
-dacs j 3
se  0
sf  7
s5  1
s0  2
sa  4
s6  6
s2  3
ss  5
```

```
# Encoding figure "dacs"
-dacs m 3
se  0
sf  7
s5  2
s0  4
sa  5
s6  6
s2  1
ss  3
```

```
# Encoding figure "dacs"
-dacs o 8
se  0   1
sf  1   2
s5  2   4
s0  3   8
sa  4   10
s6  5   20
s2  6   40
ss  7   80
```

```
# Encoding figure "dacs"
-dacs r 3
se  6
sf  0
s5  2
s0  1
sa  3
s6  7
s2  5
ss  4
```

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

```
>boom -V -d <optimization_%>  <vbe_source> <vbe_destination>
```

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).

```
> boog -l paramfile <vbe_source>
```

Param file:

```
## set the Optimization Mode (0..4)
## 0 : full area optimization
## 2 : 50% area, 50% delay
## 4 : full delay optimization
#M{2}

## set the Optimization Level (1..5)
## 1 : poor optimization - small computation time
## 5 : best optimization - long computation time
#L{3}

## External Output Capacitance (in fF)
#C{
door:100;
alarm:100;
}

```

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:

```
>loon <vst_source> <vst_destination>  paramfile
```

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

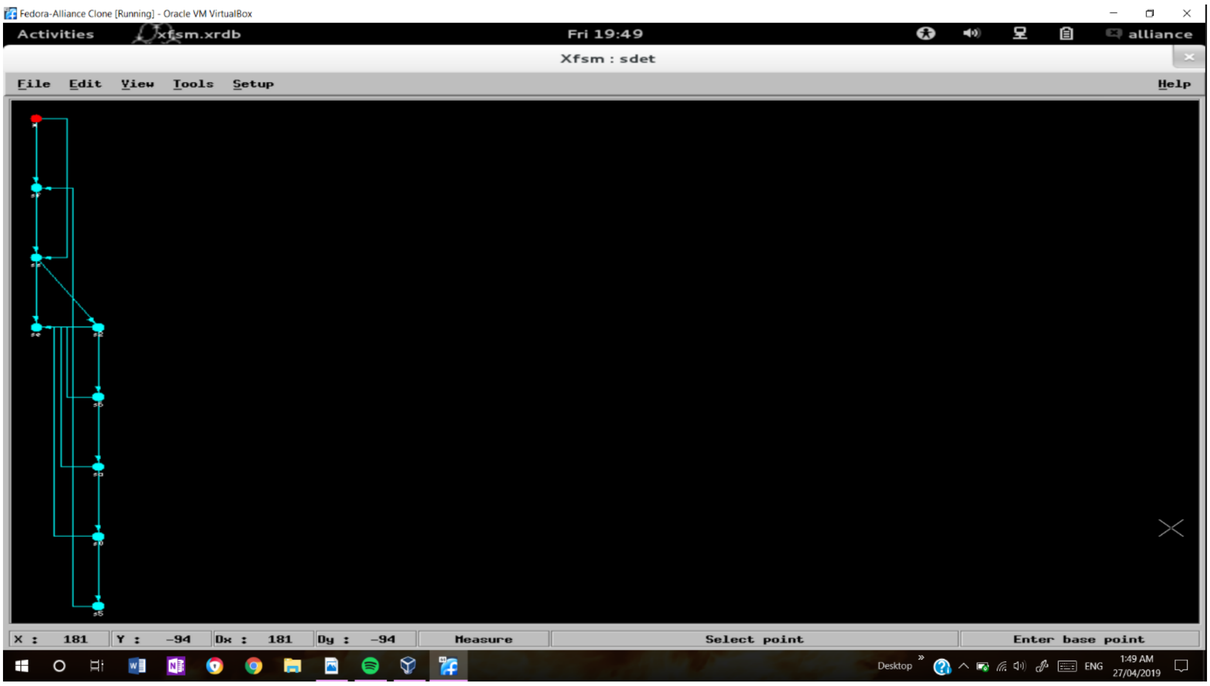
For the sdetr encoding case:

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 ***<u>sdetj</u>*** implementation as is has the minimum number of components, so it has the minimum number of components, area, and delay.

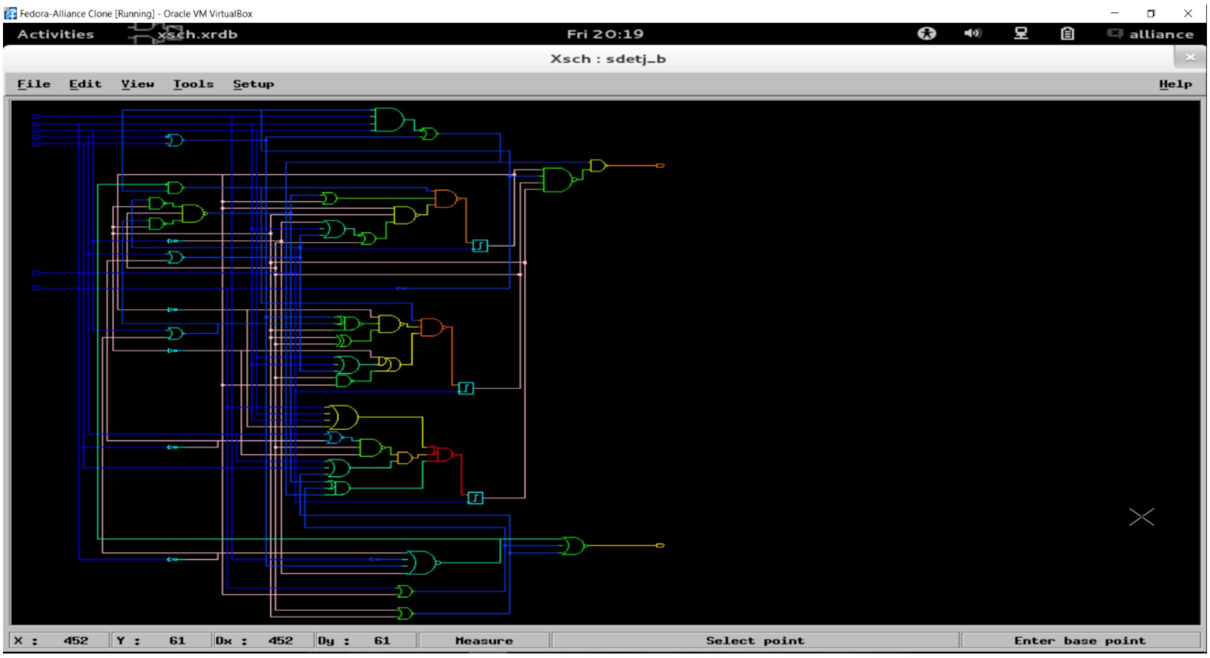
We will use the xfsm tool to visualize the state diagram of the circuit:

```
>xsch -I vst -l <vst_source>
```

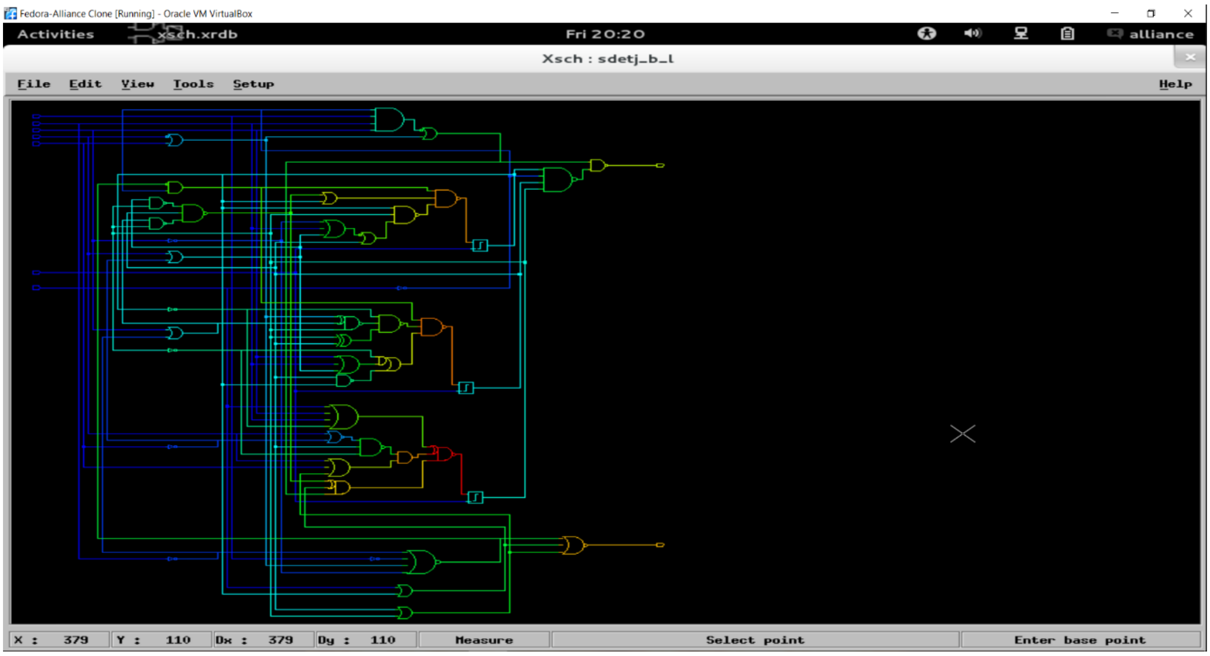


And then use the xsch tool to visualize the circuit it self:

Before the loon:



After the loon:

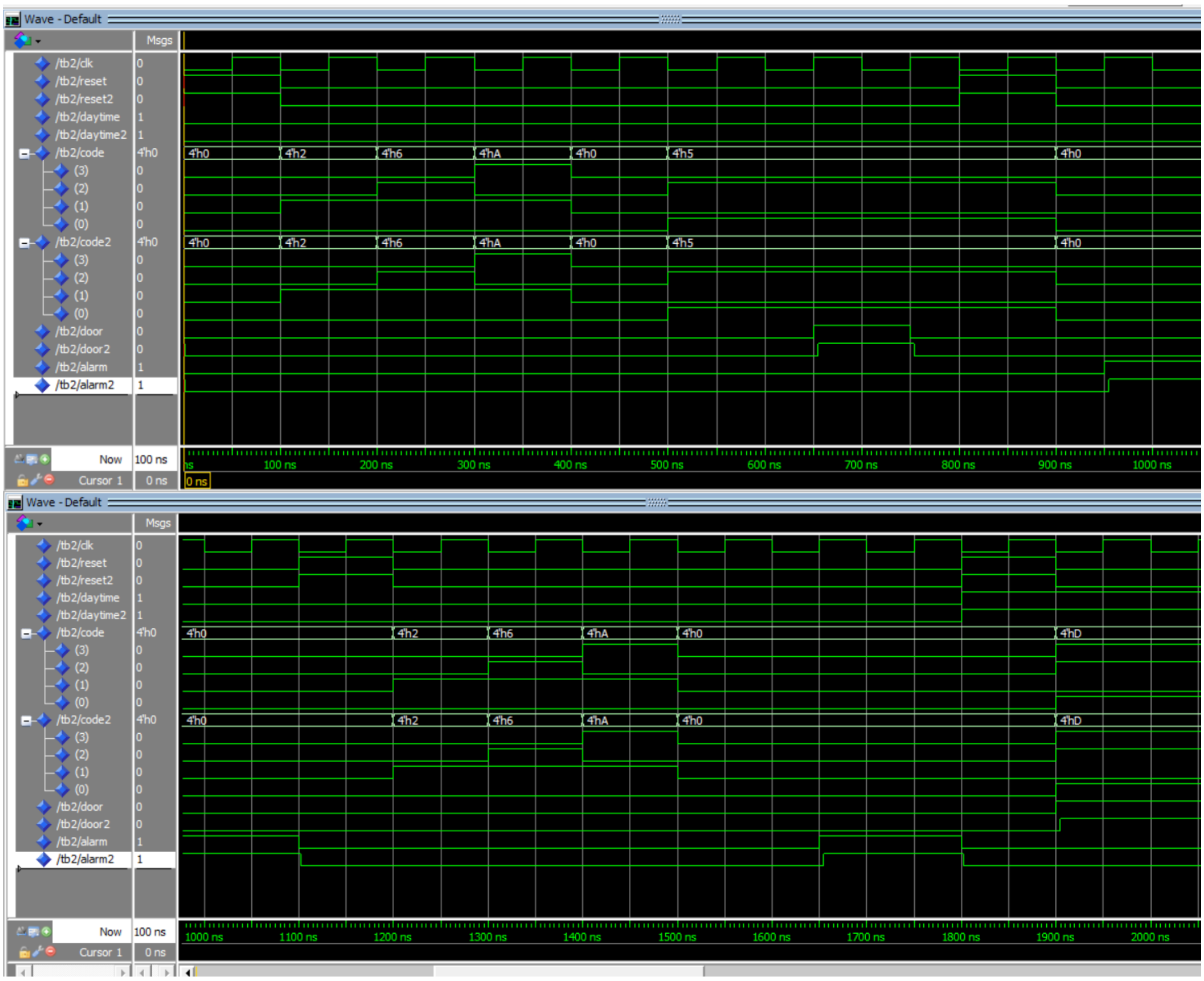


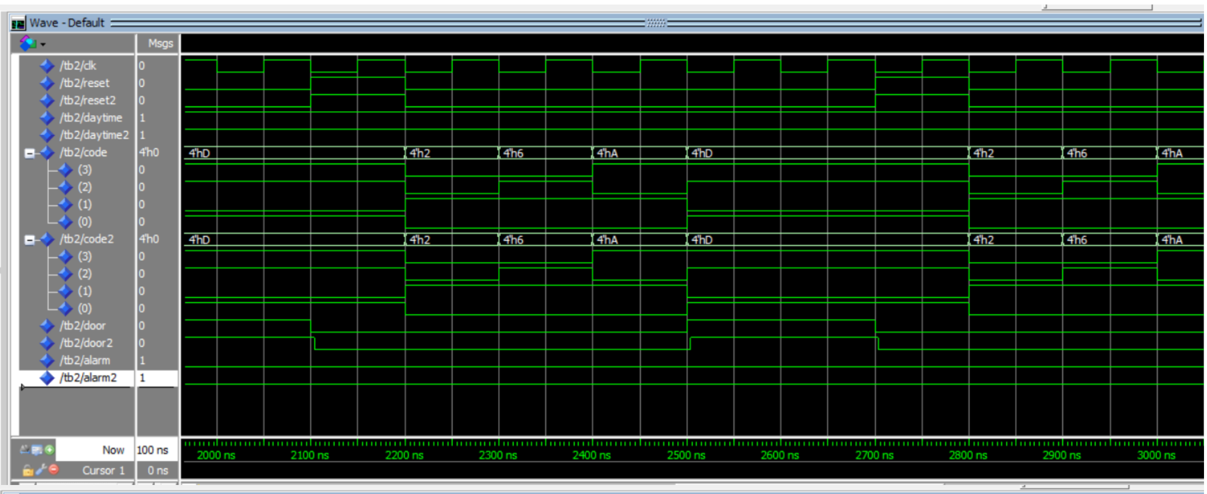
And then Make a formal comparison of our netlist with the original behavioural file resulting from **SYF** using the proof and flatbeh tools.

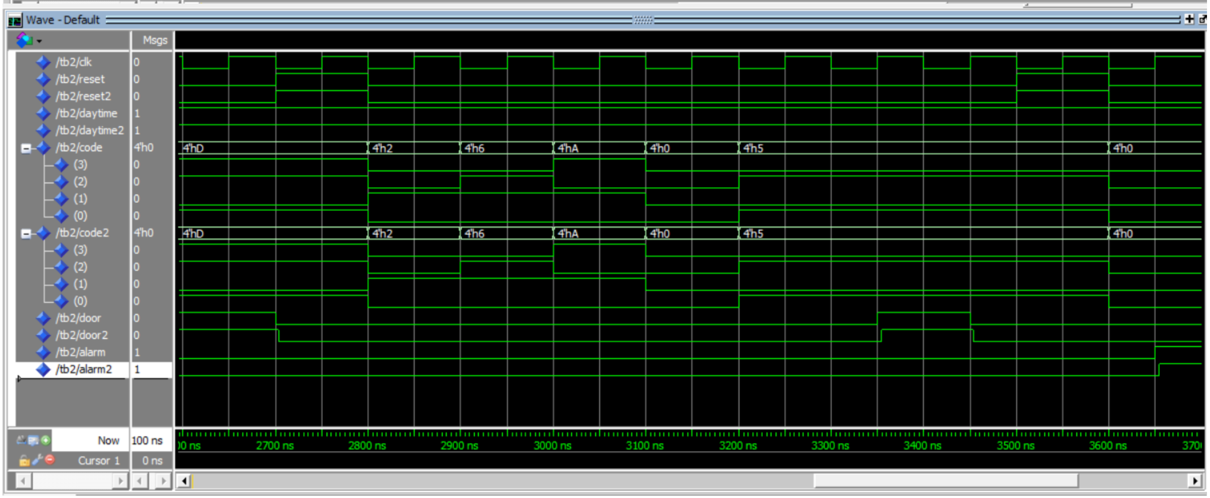
```
>flatbeh <vst_source> <vbe_dest>
```

```
>proof -d <vbe_origin> <vbe_dest>
```

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 :

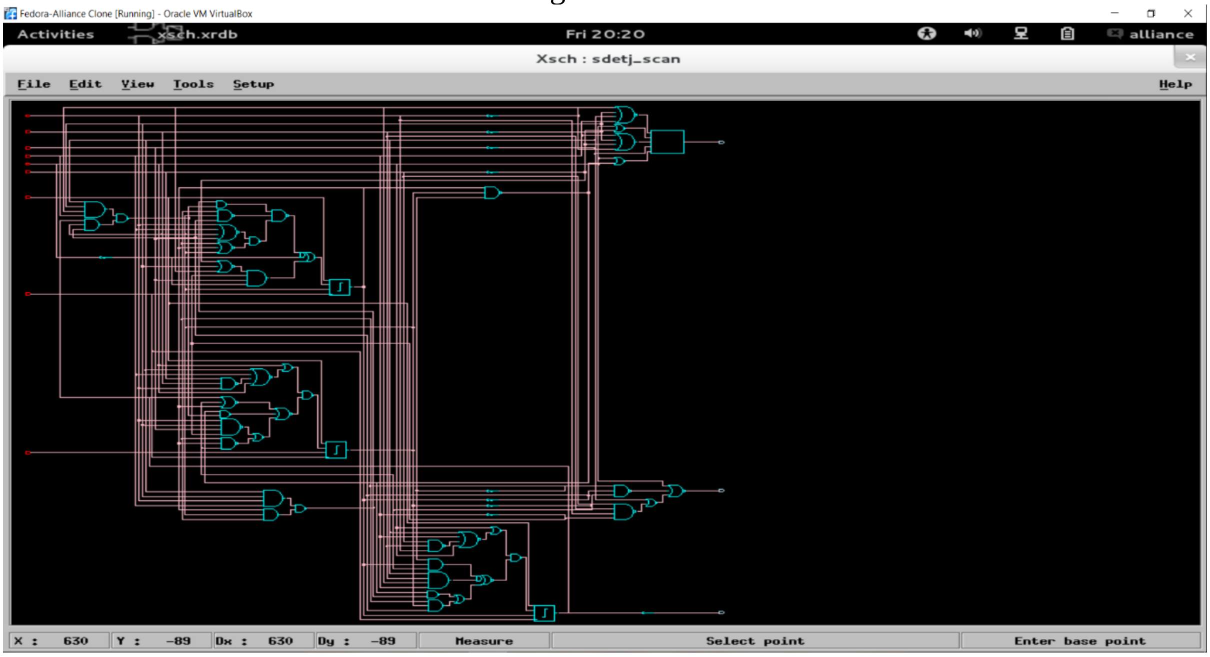
```
BEGIN_PATH_REG
dacs_cs_0_ins
dacs_cs_1_ins
dacs_cs_2_ins
END_PATH_REG


BEGIN_CONNECTOR
SCAN_IN scanin
SCAN_OUT scanout
SCAN_TEST test
END_CONNECTOR
```

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.

Insert a scan-path connecting all the design registers with the following command:

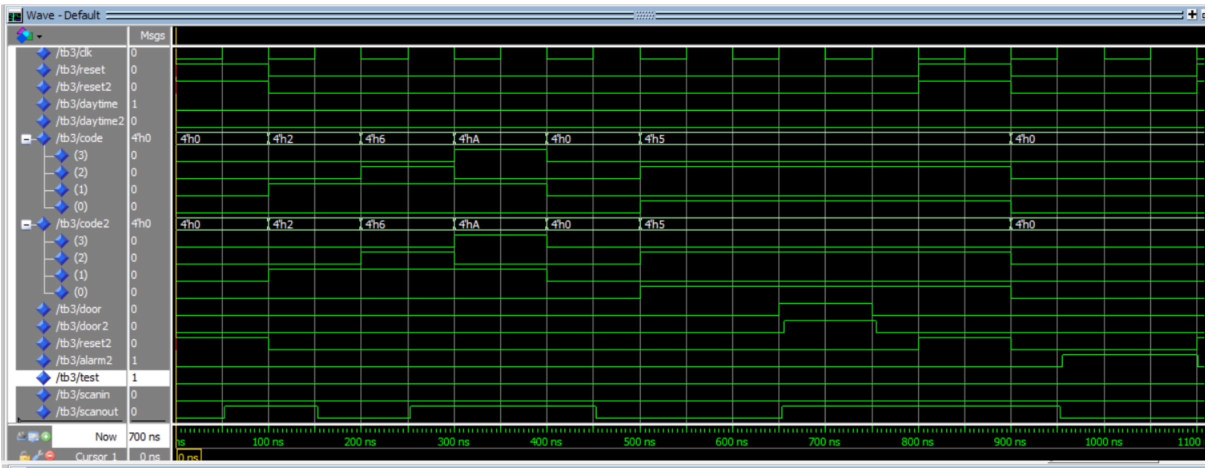
```
>scapin -VRB <vst_source> <path_file> <vst_dest>
```

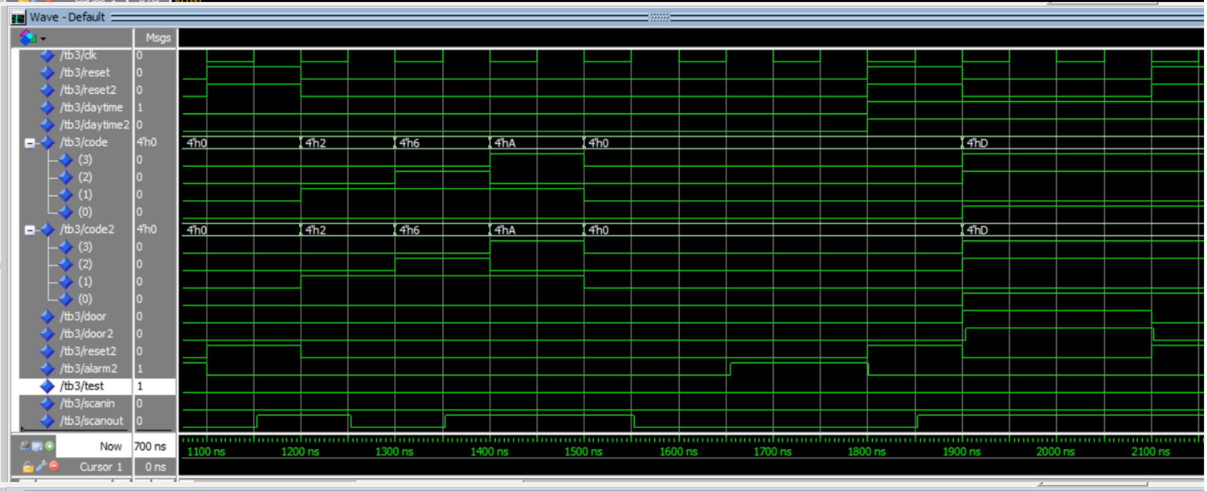
Then use xsch tool to visualize the resulting netlist:

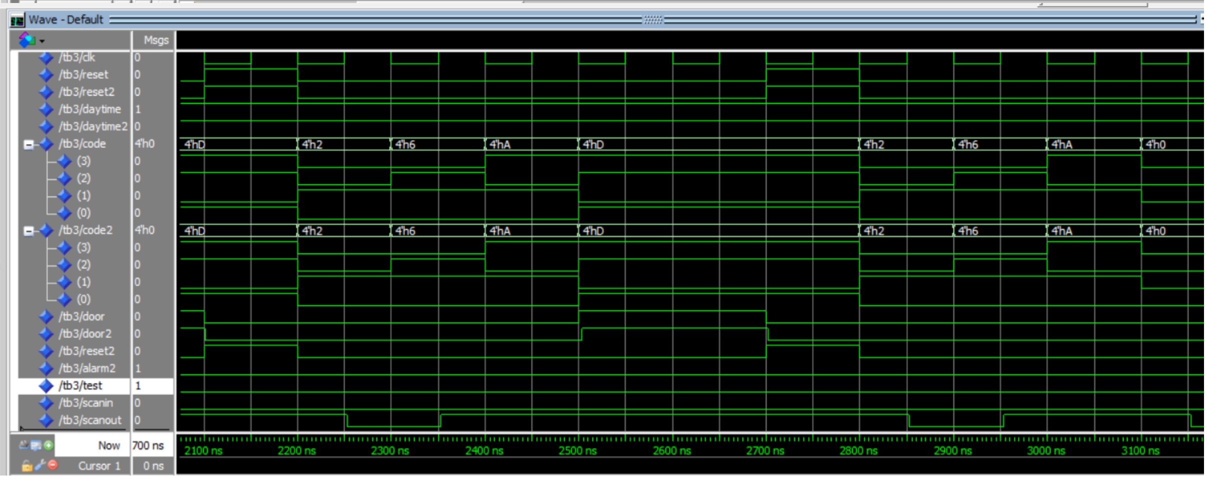


Extend the original testbench to test the scan-path and simulate with **ModelSim**.

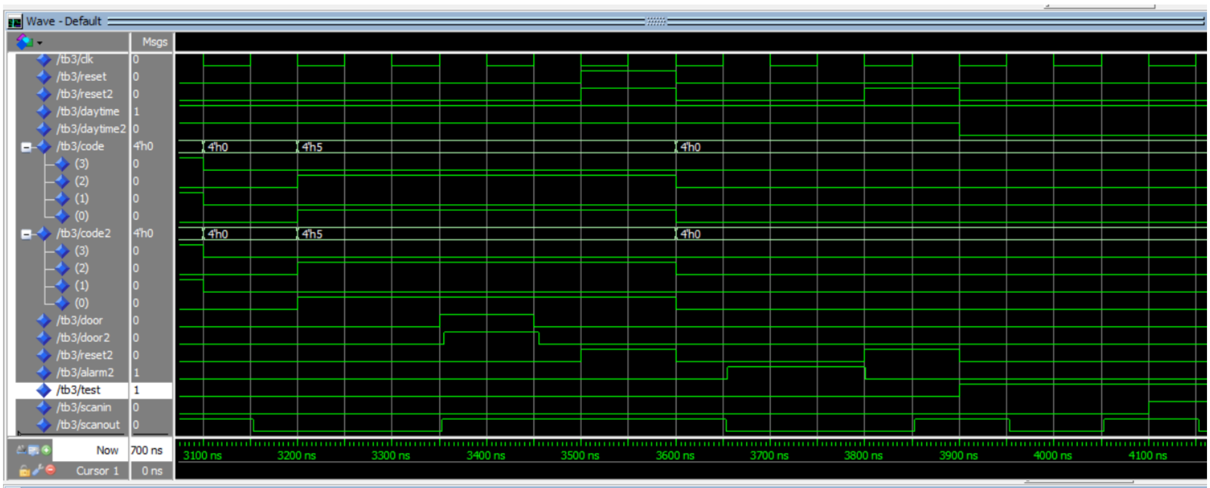
Test=0:

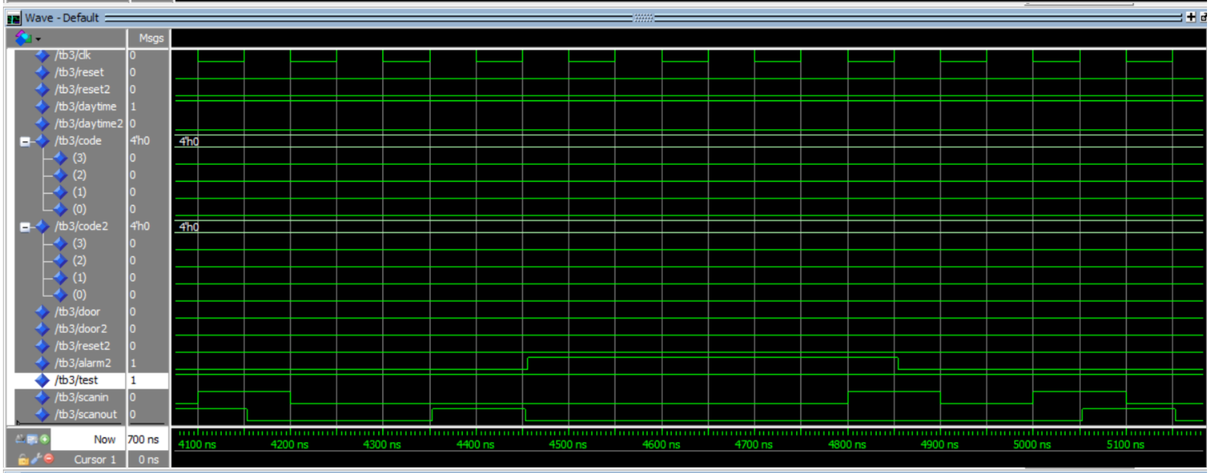

Wave - Default
Msgs
/tb3/clk
/tb3/reset
/tb3/reset2
/tb3/daytime
/tb3/daytime2
/tb3/code
(3)
(2)
(1)
(0)
/tb3/code2
(3)
(2)
(1)
(0)
/tb3/door
/tb3/door2
/tb3/reset2
/tb3/alarm2
/tb3/test
/tb3/scanin
/tb3/scanout
Now 700 ns
Cursor 1 0 ns
4'h0 4'h2 4'h6 4'hA 4'h0 4'h5 4'h0
100 ns 200 ns 300 ns 400 ns 500 ns 600 ns 700 ns 800 ns 900 ns 1000 ns 1100

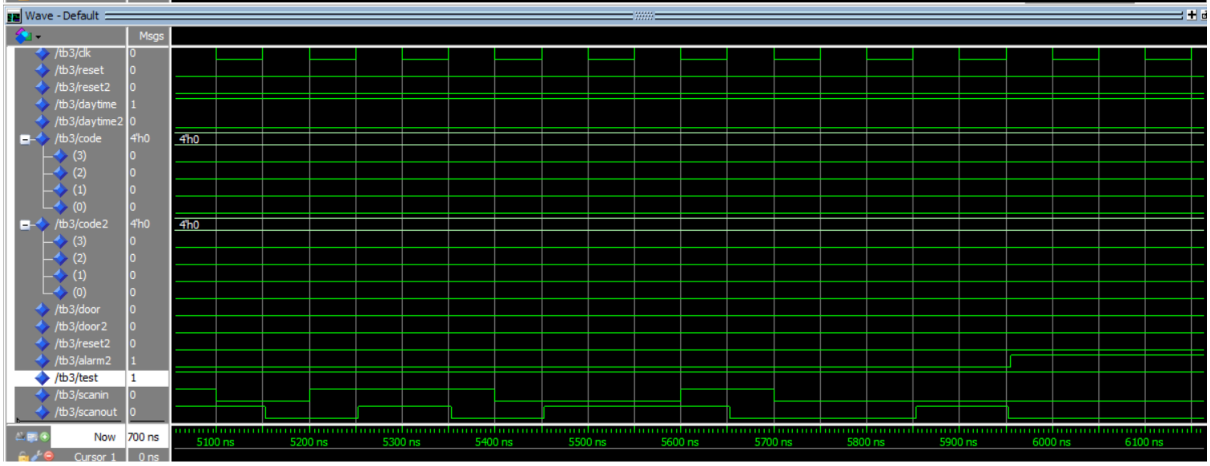

Wave - Default
Msgs
/tb3/clk
/tb3/reset
/tb3/reset2
/tb3/daytime
/tb3/daytime2
/tb3/code
(3)
(2)
(1)
(0)
/tb3/code2
(3)
(2)
(1)
(0)
/tb3/door
/tb3/door2
/tb3/reset2
/tb3/alarm2
/tb3/test
/tb3/scanin
/tb3/scanout
Now 700 ns
Cursor 1 0 ns
4'h0 4'h2 4'h6 4'hA 4'h0 4'hD
1100 ns 1200 ns 1300 ns 1400 ns 1500 ns 1600 ns 1700 ns 1800 ns 1900 ns 2000 ns 2100 ns


Wave - Default
Msgs
/tb3/clk
/tb3/reset
/tb3/reset2
/tb3/daytime
/tb3/daytime2
/tb3/code
(3)
(2)
(1)
(0)
/tb3/code2
(3)
(2)
(1)
(0)
/tb3/door
/tb3/door2
/tb3/reset2
/tb3/alarm2
/tb3/test
/tb3/scanin
/tb3/scanout
Now 700 ns
Cursor 1 0 ns
4'hD 4'h2 4'h6 4'hA 4'hD 4'h2 4'h6 4'hA 4'h0
2100 ns 2200 ns 2300 ns 2400 ns 2500 ns 2600 ns 2700 ns 2800 ns 2900 ns 3000 ns 3100 ns

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<="0010011010100000101";
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:

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

                             SCAn Path INsertion

                Alliance CAD System 5.0 20090901, scapin 5.0
                Copyright (c) 2000-2019,      ASIM/LIP6/UPMC
                Author(s):                   Ludovic Jacomme
                Contributor(s):              Ilhem Kazi Tani
                E-mail        :  alliance-users@asim.lip6.fr

--> Parse parameter file /usr/lib64/alliance/etc/sxlib.scapin
--> Parse path file scan
--> Compile Structural file sdetr
--> Insert Scan Path
    > Replace register dacs_cs_0_ins (sff1_x4) by a reg-mux (sff2_x4)
    > Replace register dacs_cs_1_ins (sff1_x4) by a reg-mux (sff2_x4)
    > Replace register dacs_cs_2_ins (sff1_x4) by a reg-mux (sff2_x4)
    Insert a buffer (buf_x2) before the port scanout
--> Save Structural file sdetr_scan
```

Proof output:

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

                                  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   = sdetj.vbe
Second VHDL file  = sdetj_b_net.vbe
The auxiliary signals are erased
Errors are displayed
===============================================================================

Compiling 'sdetj' ...
Compiling 'sdetj_b_net' ...
---> final number of nodes = 295(138)

Running Abl2Bdd on `sdetj_b_net`
```

```
-------------------------------------------------------------------------------
             Formal proof with Ordered Binary Decision Diagrams between

             './sdetj'  and  './sdetj_b_net'
-------------------------------------------------------------------------------
==============================  PRIMARY OUTPUT  ==============================
=============================  AUXILIARY SIGNAL  =============================
==============================  REGISTER SIGNAL  =============================
================================  EXTERNAL BUS ===============================
================================ INTERNAL BUS ================================

                            Formal Proof : OK

pppppppppppppppppppppppppprrrrrrrrrrooooooooooooooooooooooooooooooffffffffffffff
-------------------------------------------------------------------------------
```

Flatbeh output:

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

                        a netlist abstractor

     Alliance CAD System 5.0 20090901, flatbeh 5.0 [2000/11/01]
     Copyright (c) 1993-2019,                       ASIM/LIP6/UPMC
     Author(s):                 François 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:/usr/lib64/alliance/cells/pxlib:/usr/
lib64/alliance/cells/padlib
MBK_CATAL_NAME      = CATAL
============================== Files ==============================
Netlist file        = sdetj_b.vst
Output  file        = sdetj_b_net.vbe
===================================================================
```

```
Loading './sdetj_b.vst'
flattening figure sdetj_b
 loading a4_x2
 loading an12_x1
 loading inv_x2
 loading no4_x1
 loading oa22_x2
 loading xr2_x1
 loading ao22_x2
 loading no2_x1
 loading a2_x2
 loading o4_x2
 loading nao22_x1
 loading o3_x2
 loading on12_x1
 loading o2_x2
 loading na3_x1
 loading sff1_x4
 loading no3_x1
 loading na4_x1
 loading na2_x1
Restoring array's orders
BEH : Saving 'sdetj_b_net' in a vhdl file (vbe)
```

.path file:

```
BEGIN_PATH_REG
dacs_cs_0_ins
dacs_cs_1_ins
dacs_cs_2_ins
END_PATH_REG


BEGIN_CONNECTOR
SCAN_IN scanin
SCAN_OUT scanout
SCAN_TEST test
END_CONNECTOR
```

Paramfile:

```
## set the Optimization Mode (0..4)
## 0 : full area optimization
## 2 : 50% area, 50% delay
## 4 : full delay optimization
#M{2}

## set the Optimization Level (1..5)
## 1 : poor optimization - small computation time
## 5 : best optimization - long computation time
#L{3}

## External Output Capacitance (in fF)
#C{
door:100;
alarm:100;
}
```

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 *~
```