Table of content

1. Introduction........................................................................................................................... 2
   1.1. Input .......................................................................................................................... 2
2. Assignment 1: Manual test point insertion ........................................................................ 3
3. Assignment 2: Automatic scan chain insertion................................................................. 3
4. Examination and submission .......................................................................................... 3
   4.1. Oral examination..................................................................................................... 3
   4.2. Written examination............................................................................................... 3
Appendix A S27 VHDL description....................................................................................... 4
1. Introduction
The goal of the lab is to get experience and knowledge about different design for test (DFT) techniques.

The following techniques will be used:
- Manual test point insertion (VHDL)
- Automatic scan chain insertion
- Automatic test pattern generation

The following tools will be used:
- Leonardo Spectrum
- DFTAdvisor
- FlexTest

1.1. Input
One designs names s27, from the ISCAS 85 benchmark is given. The files needed to solve the lab are downloaded from using the following link:

\texttt{www.ida.liu.se/~TDDC33/labs/download/lab2.tar.gz}

The sequential design s27 has four inputs and one output as illustrated in Figure 1. The s27 design is originally described in VHDL format and the complete description is given in Appendix A.

![Figure 1. The s27 design.](image-url)
2. Assignment 1: Manual test point insertion
   - Analyze the s27 design and insert test points, both for enhanced controllability and observability.
   - Use FlexTest to generate test patterns for the s27 design.

   Report the following for each design:
   - Number of test points added
   - Achieved fault coverage before adding test points
   - Achieved fault coverage after adding test points

   The obtained fault coverage for each student will be published on the course web-page.

3. Assignment 2: Automatic scan chain insertion
   - Use DFTAdvisor to insert scan chains into the s27 design
   - Use FlexTest to generate test patterns for the scanable s27 design.

   Report the following for the design:
   - Achieved fault coverage
   - Number of random patterns (cycles)
   - Number of deterministic patterns (cycles)

4. Examination and submission

4.1. Oral examination
   - Prepare to show your results and to answer questions about the lab assignments during the lab sessions. Notify the lab assistant!

4.2. Written examination
   - Write a report containing the results from Assignment 1 and Assignment 2.
   - Write a short summary of your personal experience of the lab (will not be graded). This summary can for instance contain comments on the level of difficulty, the instructions, the tools, and possible improvements.

   Place the report inside a Laboration report cover, which should be signed and submitted to the lab assistant. (The Laboration report cover can be found by the printers.)

   The deadline for submitting the lab is presented on the course web-page.

Good Luck!
Appendix A S27 VHDL description

library IEEE;
use IEEE.std_logic_1164.all;
use work.all;

ENTITY s27_bench IS
    PORT (INP: in std_ulogic_vector(0 to 3);
          OUTP : out std_ulogic_vector(0 to 0);
          H : in std_ulogic);
END s27_bench;

ARCHITECTURE structural OF s27_bench IS
    component andg
        generic (tpd_hl : time;
                 tpd_lh : time);
        port (in1, in2 : std_logic;
              out1 : out std_logic);
    end component;

    component org
        generic (tpd_hl : time;
                 tpd_lh : time);
        port (in1, in2 : std_logic;
              out1 : out std_logic);
    end component;

    component xorg
        generic (tpd_hl : time;
                 tpd_lh : time);
        port (in1, in2 : std_logic;
              out1 : out std_logic);
    end component;

    component xnorg
        generic (tpd_hl : time;
                 tpd_lh : time);
        port (in1, in2 : std_logic;
              out1 : out std_logic);
    end component;

    component nandg
        generic (tpd_hl : time;
                 tpd_lh : time);
        port (in1, in2 : std_logic;
              out1 : out std_logic);
    end component;

    component norg
        generic (tpd_hl : time;
                 tpd_lh : time);
        port (in1, in2 : std_logic;
              out1 : out std_logic);
    end component;
end component;

component invg
  generic (tpd_hl : time;
           tpd_lh : time);
  port (in1 : std_logic;
        out1 : out std_logic);
end component;

component buffg
  generic (tpd_hl : time;
           tpd_lh : time);
  port (in1 : std_logic;
        out1 : out std_logic);
end component;

-- ******* Portes generiques sur le nombre d'entr,e

component andg_n
  generic (n      : integer ;
           tpd_hl : time ;
           tpd_lh : time);
  port (inp : std_logic_vector(0 to n-1);
        out1 : out std_logic) ;
end component;

component nandg_n
  generic (n      : integer ;
           tpd_hl : time ;
           tpd_lh : time);
  port (inp : std_logic_vector(0 to n-1);
        out1 : out std_logic) ;
end component;

component org_n
  generic (n      : integer ;
           tpd_hl : time ;
           tpd_lh : time);
  port (inp : std_logic_vector(0 to n-1);
        out1 : out std_logic) ;
end component;

component norg_n
  generic (n      : integer ;
           tpd_hl : time ;
           tpd_lh : time);
  port (inp : std_logic_vector(0 to n-1);
        out1 : out std_logic) ;
end component;

component xorg_n
  generic (n      : integer ;
           tpd_hl : time ;
           tpd_lh : time);
  port (inp : std_logic_vector(0 to n-1);
        out1 : out std_logic) ;
end component;

component xnorg_n
  generic (n      : integer ;
           tpd_hl : time ;
           tpd_lh : time);
  port (inp : std_logic_vector(0 to n-1);
        out1 : out std_logic) ;
end component;
component DFF
  generic (tpd_hl : time;
    tpd_lh : time);
  port (D,H : std_logic;
    Q : out std_logic);
end component;

component DFFC
  generic (tpd_hl : time;
    tpd_lh : time);
  port (DFFC,H,C : std_logic;
    Q : out std_logic);
end component;

component TFFC
  generic (tpd_hl : time;
    tpd_lh : time);
  port (T,H,C : std_logic;
    Q : out std_logic);
end component;

signal INTERP : std_ulogic_vector(0 to 11):=(others=>'0') ;
signal OUTPI : std_ulogic_vector(OUTP'range):=(others=>'0') ;
BEGIN
  DFF0 : DFF generic map (1 ns,1 ns)
  port map (D => INTERP(1),
             H => H,
             Q => INTERP(0));

  DFF1 : DFF generic map (1 ns,1 ns)
  port map (D => INTERP(3),
             H => H,
             Q => INTERP(2));

  DFF2 : DFF generic map (1 ns,1 ns)
  port map (D => INTERP(5),
             H => H,
             Q => INTERP(4));

  INV0 : INVG generic map (1 ns,1 ns)
  port map (in1 => INP(0),
             out1 => INTERP(6));

  INV1 : INVG generic map (1 ns,1 ns)
  port map (in1 => INTERP(3),
             out1 => OUTPI(0));

  AND0 : ANDG_N generic map (2,1 ns,1 ns)
  port map (}
inp(0) => INTERP(6),
inp(1) => INTERP(2),
out1 => INTERP(7));

OR0 : ORG_N generic map (2,1 ns,1 ns)
port map (  
ip(0) => INTERP(9),
inp(1) => INTERP(7),
out1 => INTERP(8));

OR1 : ORG_N generic map (2,1 ns,1 ns)
port map (  
ip(0) => INP(3),
inp(1) => INTERP(7),
out1 => INTERP(10));

NAND0 : NANDG_N generic map (2,1 ns,1 ns)
port map (  
ip(0) => INTERP(10),
inp(1) => INTERP(8),
out1 => INTERP(11));

NOR0 : NORG_N generic map (2,1 ns,1 ns)
port map (  
ip(0) => INTERP(6),
inp(1) => INTERP(3),
out1 => INTERP(1));

NOR1 : NORG_N generic map (2,1 ns,1 ns)
port map (  
ip(0) => INTERP(0),
inp(1) => INTERP(11),
out1 => INTERP(3));

NOR2 : NORG_N generic map (2,1 ns,1 ns)
port map (  
inp(0) => INP(1),
inp(1) => INTERP(4),
out1 => INTERP(9));

NOR3 : NORG_N generic map (2,1 ns,1 ns)
port map (  
inp(0) => INP(2),
inp(1) => INTERP(9),
out1 => INTERP(5));

BUFFER_OUT : OUTP <= OUTPI;

END structural ;

ARCHITECTURE rtl OF s27_bench IS
signal INTERP : std_ulogic_vector(0 to 11):=(others=>'0') ;
signal OUTPI : std_ulogic_vector(OUTP'range):=(others=>'0') ;
BEGIN
REGVECT : BLOCK (H='1' AND NOT H'STABLE)
BEGIN
DFF3 : INTERP(0) <= GUARDED INTERP(1) after 1 ns;
DFF4 : INTERP(2) <= GUARDED INTERP(3) after 1 ns;
DFF5 : INTERP(4) <= GUARDED INTERP(5) after 1 ns;
END BLOCK ;
INV2 : INTERP(6) <= NOT(INP(0)) after 1 ns;
INV3 : OUTPI(0) <= NOT(INTERP(3)) after 1 ns;
AND1 : INTERP(7) <= INTERP(6) AND INTERP(2) after 1 ns;
OR2 : INTERP(8) <= INTERP(9) OR INTERP(7) after 1 ns;
OR3 : INTERP(10) <= INP(3) OR INTERP(7) after 1 ns;
NAND1 : INTERP(11) <= NOT(INTERP(10) AND INTERP(8)) after 1 ns;
NOR4 : INTERP(1) <= NOT(INTERP(6) OR INTERP(3)) after 1 ns;
NOR5 : INTERP(3) <= NOT(INTERP(0) OR INTERP(11)) after 1 ns;
NOR6 : INTERP(9) <= NOT(INP(1) OR INTERP(4)) after 1 ns;
NOR7 : INTERP(5) <= NOT(INP(2) OR INTERP(9)) after 1 ns;
BUFFER_OUT : OUTP <= OUTPI;

END rtl ;