Design for Test of Digital Systems

TDDC33

Lab2
Design for Test

Date of last revision 24/08/2012

2011
Dimitar Nikolov, IDA/SaS ESLAB
# 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 ......................................................................................5  
   4.1. Oral examination.................................................................................................5  
   4.2. Written examination............................................................................................5  
Appendix A S27 VHDL description..................................................................................6
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
- FastScan

1.1. **Input**

The design named s27, from the ISCAS 85 benchmark is given. The files needed to solve the lab can be downloaded from the following link:

[www.ida.liu.se/~TDDC33/labs/download/lab2.tar.gz](http://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.**
2. **Assignment 1: Manual test point insertion**
   - Analyze the s27 design and insert test points, both for enhanced controllability and observability.
   - Use FastScan 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

3. **Assignment 2: Full scan chain insertion**
   - Use DFTAdvisor to insert scan chains of type full scan into the s27 design
   - Use FastScan 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. **Assignment 3: Partial scan chain insertion**
   - Use DFTAdvisor to insert scan chains of type partial scan into the s27 design
   - Use FastScan to generate test patterns for the scanable s27 design.

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

**Note!** Since this process is done manually, i.e. the user identifies which flip-flops to be converted into scan cells and connected in a scan chain, the fault coverage may differ. Therefore there is a challenging task, and that is find the best solution (a design with the highest fault coverage) when you insert scan chains with two scan cells, and scan chains with only one scan cell. You are allowed to use exhaustive search, i.e. you try all possible combinations for implementing scan chains with length of 1 and 2, or you might want to do some analysis on the circuit to identify which flip-flops should be included in the scan chains such that the fault coverage for the new design is maximized.
4. Assignment 3: Test point insertion combined with Partial scan

- Analyze the s27 design and insert test points, both for enhanced controllability and observability.
- Use DFTAdvisor to insert scan chains of type partial scan into the s27 design.
- Use FastScan to generate test patterns for the scanable s27 design.

Report the following for the design:

- Number of test points inserted
- Report the test points, location and type (controllable or observable)
- Number of scan cells used
- 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.

 The Lab report should include the fault coverage for the following designs:
  
  o s27 with no modifications
  
  o at least 3 different modifications of s27, with test points inserted
     for each modification, report the type and the location of the test point
  
  o Full scan insertion for s27

  o Partial scan insertion, where the length of the scan chain is 2. If you have tried the exhaustive search to find the best solution (a design with the highest fault coverage) report the results for all combinations. If you have analyzed the circuit, report your analysis and motivate why you selected those two particular flip-flops to be converted into scan cells and connected in a scan chain.
     Report which two flip-flops are the best choice to be selected

  o Partial scan insertion, where the length of the scan chain is 1. If you have tried the exhaustive search to find the best solution (a design with the highest fault coverage) report the results for all combinations. If you have analyzed the circuit, report your analysis and motivate why you selected those one particular flip-flop to be converted into a scan cell.
     Report which flip-flop is the best choice to be selected

  o At least 2 variations of s27, that include both test points and partial scan
     Report the location and type of the inserted test points
     Report which flip-flops have been used for partial scan

 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!
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 DFFC
  generic (tpd_hl : time;
            tpd_lh : time);
  port (DFFC,H,C : std_logic;
        Q     : 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 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 (  
inp(0) => INTERP(9),
inp(1) => INTERP(7),
out1 => INTERP(8));

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

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

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

NOR1 : NORG_N generic map (2,1 ns,1 ns)
port map (  
inp(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(9)) 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 ;