Lab1
Test Pattern Generation

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 pattern generation ............................................................ 3  
3. Assignment 2: Automatic test pattern generation ....................................................... 3  
4. Examination and submission ...................................................................................... 4  
   4.1. Oral examination..................................................................................................... 4  
   4.2. Written examination.............................................................................................. 4  
Appendix A C17 VHDL description ................................................................................. 5  
Appendix B S27 VHDL Description ................................................................................ 9
1. **Introduction**

The goal of the lab is to get experience and knowledge about test pattern generation for combinatorial and sequential circuits.

The following techniques will be used:
- Manual test pattern generation
- Automatic Test Pattern Generation (ATPG).
- Fault simulation

The following tools will be used:
- FastScan

1.1. **Input**

Two designs, c17 and s27, from the ISCAS 85 benchmark are given. The files needed to solve the lab can be downloaded from the following link:

www.ida.liu.se/~TDDC33/labs/download/lab1.tar.gz

The combinational design c17 has five inputs and two outputs as illustrated in Figure 1. The c17 design is originally described in VHDL format and the complete description is given in Appendix A.

![Figure 1. The c17 design.](image)

The sequential design s27 has four inputs and one output as illustrated in Figure 2. The s27 design is originally described in VHDL format and the complete description is given in Appendix B.
2. **Assignment 1: Manual test pattern generation**

- Analyze the c17 and the s27 designs and write (manually) test patterns that detects as many stuck-at faults as possible.
- You are allowed to use any approach when writing the test patterns. You can select random patterns and then try step by step either to increase or decrease the number of patterns such that maximal fault coverage is reached. You can use some of the existing algorithms (D-algorithm), in which case you need to make sure that you perform the analysis on the "right" design.
- Use FastScan for fault simulation to obtain the fault coverage.

Report the following for each design:

- **Number of test patterns used**
- **Achieved fault coverage**
- **Test patterns**

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

3. **Assignment 2: Automatic test pattern generation**

- Use the ATPG capability of FastScan to generate test patterns for the c17 and the s27 designs.

Report the following for each design:

- **Number of test patterns**
- **Achieved fault coverage**
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 C17 VHDL description

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

ENTITY c17_i89 IS
PORT (
    INP: in std_ulogic_vector(0 to 4);
    OUTP : out std_ulogic_vector(0 to 1)
);
END c17_i89 ;

ARCHITECTURE structural OF c17_i89 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;

component invg

TDDC33 Design for Test of Digital Systems

component buffg
  generic (tpd_hl : time; tpd_lh : time);
  port (in1 : std_logic; 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)}
end component;
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 3):=(others=>'0') ;
signal OUTPI : std_ulogic_vector(OUTP'range):=(others=>'0') ;
BEGIN
  NAND0 : NANDG_N generic map (2,1 ns,1 ns)
  port map (
    inp(0) => INP(0),
    inp(1) => INP(2),
    out1 => INTERP(0));

  NAND1 : NANDG_N generic map (2,1 ns,1 ns)
  port map (
    inp(0) => INP(2),
    inp(1) => INP(3),
    out1 => INTERP(1));

  NAND2 : NANDG_N generic map (2,1 ns,1 ns)
  port map (
    inp(0) => INP(1),
    inp(1) => INTERP(1),
    out1 => INTERP(2));

  NAND3 : NANDG_N generic map (2,1 ns,1 ns)
  port map (
    inp(0) => INTERP(1),
    inp(1) => INP(4),
    out1 => INTERP(3));

  NAND4 : NANDG_N generic map (2,1 ns,1 ns)
  port map (
    inp(0) => INTERP(0),
    inp(1) => INTERP(2),
    out1 => OUTPI(0));

  NAND5 : NANDG_N generic map (2,1 ns,1 ns)
  port map (
    inp(0) => INTERP(2),

TDDC33 Design for Test of Digital Systems
inp(1) => INTERP(3),
out1 => OUTPI(1));

BUFFER_OUT : OUTP <= OUTPI;

END structural;

ARCHITECTURE rtl OF c17_i89 IS
    signal INTERP : std_ulogic_vector(0 to 3):=(others=>'0') ;
    signal OUTPI : std_ulogic_vector(OUTP'range):=(others=>'0') ;
    BEGIN
        NAND6 : INTERP(0) <= NOT(INP(0) AND INP(2)) after 1 ns;
        NAND7 : INTERP(1) <= NOT(INP(2) AND INP(3)) after 1 ns;
        NAND8 : INTERP(2) <= NOT(INP(1) AND INTERP(1)) after 1 ns;
        NAND9 : INTERP(3) <= NOT(INP(0) AND INP(4)) after 1 ns;
        NAND10 : OUTPI(0) <= NOT(INP(0) AND INTERP(2)) after 1 ns;
        NAND11 : OUTPI(1) <= NOT(INP(2) AND INTERP(3)) after 1 ns;
        BUFFER_OUT : OUTP <= OUTPI;
    END rtl ;
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 nor 
    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 ;
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 ;

TDDC33 Design for Test of Digital Systems 12
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(10) AND INTERP(8)) 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 ;