

Problem 8: Pattern Generator for Built-In Self-Test

Source: SynTest Technologies, Inc.

December 28, 2000

I. Introduction

BIST (Built-In Self-Test) is usually applied on million gate designs to achieve at-speed testing. Many BIST methodologies have been proposed [1]. Nevertheless, they all share the similar structure, e.g., a pseudo-random pattern generator (*PRPG*), a circuit under test, and an output-response analyzer. In particular, the quality of a random pattern generator plays a key role in getting higher fault coverage. Therefore, it is desirable to develop a PRPG with the highest fault coverage.

A popular BIST approach is called *STUMPS* (*Self-Testing Using MISR [Multiple Input Signature Register] and Parallel SRSG [Shift Register Sequence Generator]*). Figure 1 shows such a STUMPS BIST architecture. As shown in the figure, a full-scan circuit is fed with random patterns from the Primary/Signal Inputs (PIs/SIs), and the corresponding test responses are shifted out and analyzed through the Primary/Signal Outputs (POs/SOs).

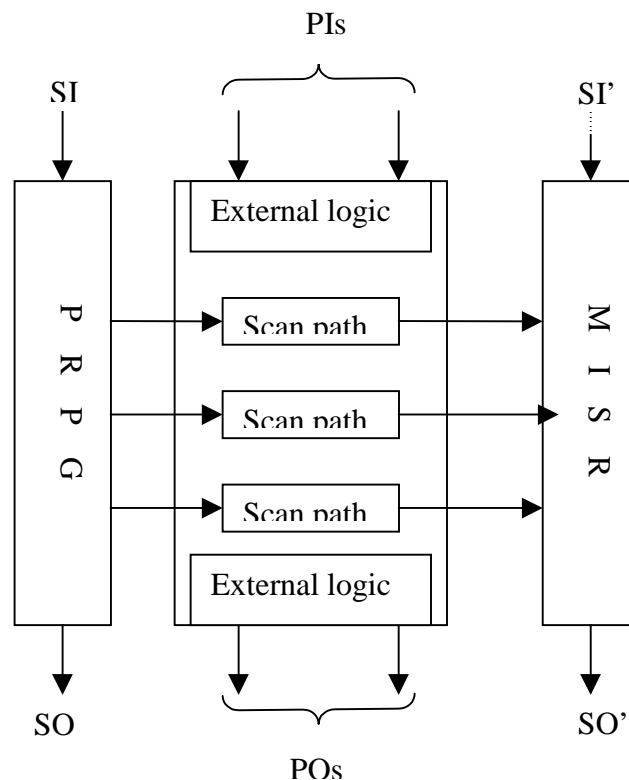


Figure 1. The STUMPS architecture

II. Problem Description

Given a full-scan gate-level circuit described in Verilog, you are asked to design a Pseudo-Random Pattern Generator (PRPG) on the STUMPS BIST structure to achieve the highest fault coverage.

III. Input

A full-scan gate-level circuit in Verilog format. An example of a full-scan gate-level circuit is given below.

```

module s27_s0 (G17, SCAN_OUT, G0, G1, G2, G3, SCAN_IN, CLK,
              STI_TM1);
output  G17, SCAN_OUT;
input   G0, G1, G2, G3, SCAN_IN, CLK, STI_TM1;
wire    G10, G11, G12, G5, G13, G6, G14, G7, G15, G8, G9, G16;
FD1S I5 (.CP (CLK), .D(G10), .Q(G5), .TE (I5_TE ) , .TI
( SCAN_IN ) );
FD1S I6 (.CP(CLK), .D(G11), .Q(G6), .TE(I5_TE), .TI(G5));
FD1S I7 (.CP(CLK), .D(G13), .Q(G7), .TE(I5_TE), .TI(G6));
IV I14 (.A(G0), .Z(G14));
IV I17 (.A(G11), .Z(G17));
AN2 I8 (.A(G14), .B(G6), .Z(G8));
OR2 I15 (.A(G12), .B(G8), .Z(G15));
OR2 I16 (.A(G3), .B(G8), .Z(G16));
ND2 I9 (.A(G16), .B(G15), .Z(G9));
NR2 I10 (.A(G14), .B(G11), .Z(G10));
NR2 I11 (.A(G5), .B(G9), .Z(G11));
NR2 I12 (.A(G1), .B(G7), .Z(G12));
NR2 I13 (.A(G2), .B(G12), .Z(G13));
SCAN_DECODER_CLAS SCAN_DECODER (.CTRL1(STI_TM1) ,
                               .SCANSYN_SCAN(I5_TE));
BU SCANSYN_BUF_0 (.A(G7), .Z(SCAN_OUT));
endmodule

module SCAN_DECODER_CLASS (SCANSYN_SCAN, CTRL1);
output SCANSYN_SCAN;
input  CTRL1;
BU C1 (.A (CTRL1), .Z (SCANSYN_SCAN));
Endmodule

```

IV. Output

- A pseudo-random pattern generator (PRPG) in Verilog RTL format.

- Output patterns of PRPG in VCD format.

V. Language/Platform

- Language: C
- Platform: Sun/Solaris

VI. Evaluation

Fault coverage (most important) based on SynTest tools, CPU time, and memory requirement. The contestants may also use Verifault (from Cadence) or other fault simulator, if available, to obtain their fault coverage reports. ***The evaluation, however, will be based on the results reported by the SynTest tools.*** (It should be noted that different fault simulators might report different results on fault coverage.)

VII. Technical Support

Tools provided by SynTest:

- A Verilog parser that can parse a Verilog source code into an AST. A library with programming language interface (PI) is available to convert the AST into your data structure. ***(In fact, you really don't need the parser to work on this problem; all you need is to identify related keywords. Hopefully, this remark can make you feel more comfortable with this problem set.)***
- A fault-simulation tool to read in the output of PRPG (in VCD format) and to report fault coverage.

Additional tools will be provided by SynTest, if necessary. Please email Guli at kifli@syntest.com.tw for further information.

VIII. Questions

Please report any question regarding this problem to cad@cis.nctu.edu.tw with the email subject "CAD Contest: Problem 8." Your question(s) will be answered in two weeks, and the Q&A's will be posted at the contest web site.

References:

- [1] M. Abramovici, M. Breuer, and A. Friedman, *Digital Systems Testing and Testable Designs*, Chapters 10 and 11, Publisher: Computer Science Press.
- [2] P. Bardell, "Analysis of cellular automata used as pseudo random pattern generator," *Proceedings of 1990 International Test Conference (ITC'90)*, page 762-768.

Appendix:

- **Hint:**

There are two ways to enhance your PRPG design. The basic idea is to do fault simulation on a given core circuit with the patterns generated by PRPG and then modify the PRPG to get higher fault coverage.

1. Using ASICGEN & Verilog simulator

SynTest will provide a fault simulation tool called ASICGEN. Here is the process:

- a. Use Verilog to simulate your PRPG and dump the simulation result in VCD format.
- b. Use VCDIN provided by SynTest to translate VCD format to SynTest test pattern format, .TP.
- c. Using front end tools provided by SynTest to translate a given core circuit into the SynTest database.
- d. ASICGEN reads in the TP file and database, and reports the fault coverage.

2. Using Verifault

You can also use Verifault to do fault simulation on PRPG and the given full-scan circuit. Please note that the fault model in Verifault and SynTest are somewhat different, thus the fault coverage are not the same all the time. You should only compare the results generated by one methodology.

ATTENTION: The evaluation will be based on the result reported by ASICGEN.

- **Bonus:**

The advanced approach is to insert test points. You can design a tool to report the locations with very poor controllability and observability. Insert test points as an extra scan chain either automatically or manually. Do fault simulation to verify your progress.

● A VCD example:

\$date	\$var wire 1 /	\$var wire 1 G	\$var wire	\$var wire 1 s	\$var wire 1 0"
Oct 09,	wr_en_n \$end	reg_latched_din_5.qb	l a g24.y	reg_write.q \$end	write_cntr.gl6.y \$end
2000 18:37:45	\$var wire 1 0	\$end	\$end	\$var wire 1 t	\$var wire 1 1"
\$end	rd_en_n \$end	\$var wire 1 H	\$var wire	reg_write.qb \$end	write_cntr.gl5.y \$end
\$version	\$var wire 1 1 din	reg_latched_din_6.q	l b	\$var wire 1 u g9.y	\$var wire 1 2"
SynTest	[8] \$end	\$end	g23.sum	\$end	write_cntr.gl4.y \$end
TurboFault	\$var wire 1 2 din	\$var wire 1 I	\$end	\$var wire 1 v g8.y	\$var wire 1 3"
V1.6	[7] \$end	reg_latched_din_6.qb	\$var wire	\$end	write_cntr.gl3.y \$end
\$end	\$var wire 1 3 din	\$end	l c	\$var wire 1 w g7.y	\$var wire 1 4"
\$timescale	[6] \$end	\$var wire 1 J	g23.cout	\$end	write_cntr.gl2.y \$end
1ns	\$var wire 1 4 din	reg_latched_din_7.q	\$end	\$var wire 1 x g6.y	\$var wire 1 5"
\$end	[5] \$end	\$end	\$var wire	\$end	write_cntr.gl1.y \$end
\$scope module	\$var wire 1 5 din	\$var wire 1 K	l d g22.y	\$var wire 1 y g5.y	\$var wire 1 6"
Top \$end	[4] \$end	reg_latched_din_7.qb	\$end	\$end	write_cntr.reg_address_3.q
\$var wire	\$var wire 1 6 din	\$end	\$var wire	\$var wire 1 z	\$end
l !	[3] \$end	\$var wire 1 L	l e g21.y	reg_ef_n.qb \$end	\$var wire 1 7"
X_default_zero	\$var wire 1 7 din	reg_latched_din_8.q	\$end	\$var wire 1 { g4.y	write_cntr.reg_address_3.qb
\$end	[2] \$end	\$end	\$var wire	\$end	\$end
\$var wire 1	\$var wire 1 8 din	\$var wire 1 M	l f g20.y	\$var wire 1 g3.sum	\$var wire 1 8"
" ff_n \$end	[1] \$end	reg_latched_din_8.qb	\$end	\$end	write_cntr.gl0.y \$end
\$var wire 1	\$var wire 1 9 din	\$end	\$var wire	\$var wire 1 } g3.cout	\$var wire 1 9"
# ef_n \$end	[0] \$end	\$var wire 1 N	l g	\$end	write_cntr.g9.y \$end
\$var wire 1	\$var wire 1 :	g35.y \$end	reg_read.q	\$var wire 1 ~ g2.y	\$var wire 1 :
\$	g37.y \$end	\$var wire 1 O	\$end	\$end	write_cntr.g8.y \$end
\$	\$var wire 1 ;	g34.y \$end	\$var wire	\$var wire 1 !" gl.y	\$var wire 1 ;"
\$	g36.y \$end	\$var wire 1 P	l h	\$end	write_cntr.g7.y \$end
\$	\$var wire 1 <	g33.y \$end	reg_read.qb	\$var wire 1 ""	\$var wire 1 <"
\$	reg_latched_din_0.q	\$var wire 1 Q	\$end	reg_ff_n.qb \$end	write_cntr.g6.y \$end
\$	\$end	g32.y \$end	\$var wire	\$var wire 1 #"	\$var wire 1 ="
\$	\$var wire 1 =	\$var wire 1 R	l i gl9.y	readl.g3.y \$end	write_cntr.reg_address_2.q
%	reg_latched_din_0.qb	ram.Q0 \$end	\$end	\$var wire 1 \$"	\$end
\$	\$end	\$var wire 1 S	\$var wire	readl.g2.y \$end	\$var wire 1 >"
\$	\$var wire 1 >	ram.Q1 \$end	l j gl8.y	\$var wire 1 %"	write_cntr.reg_address_2.qb
&	reg_latched_din_1.q	\$var wire 1 T	\$end	readl.gl.y \$end	\$end
\$	\$end	ram.Q2 \$end	\$var wire	\$var wire 1 &"	\$var wire 1 ?"
\$	\$var wire 1 ?	\$var wire 1 U	l k gl7.y	readl.g0.y \$end	write_cntr.g5.y \$end
'	reg_latched_din_1.qb	ram.Q3 \$end	\$end	\$var wire 1 '"	\$var wire 1 @"
\$	\$end	\$var wire 1 V	\$var wire	readl.reg_read_pulse.q	write_cntr.g4.y \$end
\$	\$var wire 1 @	ram.Q4 \$end	l l gl6.y	\$end	\$var wire 1 A"
(reg_latched_din_2.q	\$var wire 1 W	\$end	\$var wire 1 ("	write_cntr.g3.y \$end
\$	\$end	ram.Q5 \$end	\$var wire	readl.reg_read_pulse.qb	\$var wire 1 B"
\$	\$var wire 1 A	\$var wire 1 X	l m gl5.y	\$end	write_cntr.reg_address_1.q
l	reg_latched_din_2.qb	ram.Q6 \$end	\$end	\$var wire 1)" "	\$end
\$	\$end	\$var wire 1 Y	\$var wire	writel.g4.y \$end	\$var wire 1 C"
\$	\$var wire 1 B	ram.Q7 \$end	l n gl4.y	\$var wire 1 *"	write_cntr.reg_address_1.qb
*	reg_latched_din_3.q	\$var wire 1 Z	\$end	writel.g3.y \$end	\$end
\$	\$end	ram.Q8 \$end	\$var wire	\$var wire 1 +"	\$var wire 1 D"
\$	\$var wire 1 C	\$var wire 1	l o gl3.y	writel.g2.y \$end	write_cntr.g2.y \$end
+	reg_latched_din_3.qb	[g30.y \$end	\$end	\$var wire 1 ,"	\$var wire 1 E"
\$	\$end	\$var wire 1 \	\$var wire	writel.gl.y \$end	write_cntr.gl.y \$end
\$	\$var wire 1 D	g29.y \$end	l p gl2.y	\$var wire 1 -"	\$var wire 1 F"
l	reg_latched_din_4.q	\$var wire 1]	\$end	writel.g0.y \$end	write_cntr.g0.y \$end
\$	\$end	g28.y \$end	\$var wire	\$var wire 1 ."	\$var wire 1 G"
\$	\$var wire 1 E	\$var wire 1 ^	l q gl1.y	writel.reg_write_pulse.q	write_cntr.reg_address_0.q
-	reg_latched_din_4.qb	g27.y \$end	\$end	\$end	\$end
\$	\$end	\$var wire 1 _	\$var wire	\$var wire 1 /"	\$var wire 1 H"
\$	\$var wire 1 F	g26.y \$end	l r gl0.y	writel.reg_write_pulse.qb	write_cntr.reg_address_0.qb
l	reg_latched_din_5.q	\$var wire 1 `	\$end	\$end	\$end
\$	\$end	g25.y \$end			

\$var wire 1 I"	\$scope \$end	xR	1)"	x^"	1.	1c	#1000
read_cntr.g16.y \$end	\$enddefinitions	xS	0*"	0_"	1a	1e	0.
\$var wire 1 J"	\$end	xT	x+"	x`"	1d	1i	1e
read_cntr.g15.y \$end	#0	xU	1,"	xa"	0e	0k	1+"
\$var wire 1 K"	\$dumpvars	xV	0-"	\$end	0g	1m	#1100
read_cntr.g14.y \$end	x!	xW	0."	#100	1h	1n	1.
\$var wire 1 L"	x"	xX	x/"	0-	1j	0o	0e
read_cntr.g13.y \$end	x#	xY	10"	0.	0l	1q	0+"
\$var wire 1 M"	x\$	xZ	x1"	1/	1p	0w	#1200
read_cntr.g12.y \$end	x%	1[x2"	10	0s	1x	0/
\$var wire 1 N"	x&	x\	x3"	1:	1u	0l	11
read_cntr.g11.y \$end	x'	0]	x4"	0;	0v	1}	02
\$var wire 1 O"	x(x^	05"	1[1z	1~	03
read_cntr.reg_address_3.q	x)	x_	x6"	0]	0{	1+"	14
\$end	x*	x`	x7"	1e	1\$"	1M"	05
\$var wire 1 P"	x+	xa	x8"	0f	0+"	00"	06
read_cntr.reg_address_3.qb	x,	xb	x9"	0r	01"	1T"	07
\$end	0-	xc	x:"	0y	02"	0V"	18
\$var wire 1 Q"	x.	xd	x;"	1!"	13"	1Y"	19
read_cntr.g10.y \$end	1/	xe	0<"	1#"	14"	0["	1r
\$var wire 1 R"	10	Of	x="	0%"	06"	1^"	#1300
read_cntr.g9.y \$end	x1	xg	x>"	0&"	08"	0`"	0.
\$var wire 1 S"	x2	xh	x?"	0'"	09"	#500	1e
read_cntr.g8.y \$end	x3	xi	x@"	1)"	1:"	1-	1+"
\$var wire 1 T"	x4	xj	0A"	0*"	1;"	1.	#1400
read_cntr.g7.y \$end	x5	xk	xB"	1+"	0="	0[1.
\$var wire 1 U"	x6	xl	xC"	1,"	0?"	1]	1<
read_cntr.reg_address_2.q	x7	xm	xD"	0-"	1@"	0e	1>
\$end	x8	xn	xE"	0."	0B"	0#"	0@
\$var wire 1 W"	x9	xo	0F"	10"	1D"	1&"	0B
read_cntr.reg_address_2.qb	1:	xp	xG"	05"	1E"	0+"	0D
\$end	0;	xq	xH"	0<"	0G"	0,"	1F
\$var wire 1 X"	x<	0r	1I"	0A"	0J"	1-"	0H
read_cntr.g5.y \$end	x=	xs	xJ"	0F"	0K"	00"	0J
\$var wire 1 Y"	x>	xt	xK"	1I"	1L"	0I"	1L
read_cntr.g4.y \$end	x?	xu	xL"	0N"	0Q"	#600	0e
\$var wire 1 Z"	x@	xv	xM"	0U"	0R"	0.	1s
read_cntr.g3.y \$end	xA	xw	0N"	0Z"	1S"	1e	0u
\$var wire 1 ["	xB	xx	xO"	0_"	0X"	1+"	1v
read_cntr.reg_address_1.q	xC	0y	xP"	#300	1]"	#700	0)"
\$end	xD	xz	xQ"	1"	#400	1.	1*"
\$var wire 1 \"	xE	x{	xR"	0#	0.	0e	0+"
read_cntr.g2.y \$end	xF	xl	xS"	z\$	0N	0+"	0E"
\$var wire 1 ^"	xG	x}	xT"	z%	0O	#800	1F"
read_cntr.g1.y \$end	xH	x~	0U"	z&	0P	0.	#1500
\$var wire 1 _"	xI	1!"	xV"	z'	0Q	1e	1/
read_cntr.g0.y \$end	xJ	x""	xW"	z(0\	1+"	0r
\$var wire 1 `"	xK	1#"	xX"	z)	1^	#900	#1600
read_cntr.reg_address_0.q	xL	x\$"	xY"	z*	0_	1.	0.
\$end	xM	0%"	0Z"	z+	0`	0e	1:
\$var wire 1 a"	xN	0&"	x["	z,	0b	0+"	0;
read_cntr.reg_address_0.qb	xO	0'"	x\"				
\$end	xP	x("	x]"				
	xQ						

1e	#2200	1e	1}	0+"
1+"	1.	1+"	1~	#4200
1-"	0N	#3300	1+"	0.
0."	0e	1\$	1R"	1e
#1800	0+"	0%	1X"	1+"
1.	#2300	0&	0Y"	#4300
0^	0.	1'	1Z"	1.
1_	1N	0(0J"	0e
0`	1e	0)	1^"	0+"
1b	1+"	0*	0_"	#4400
0c	#2400	1+	1`"	0.
0e	1.	1,	#3700	1e
0j	0N	1.	0#	1+"
1k	0e	1:	z\$	#4500
0s	0+"	0;	z%	1.
1u	#2500	0N	z&	0e
0v	0.	1R	z'	0+"
0x	1N	1S	z(#4600
1y	1e	0T	z)	0.
1l	1+"	0U	z*	1e
0}	#2600	0V	z+	1+"
0~	1.	1W	z,	#4700
1!"	0N	0X	1.	1.
1)"	0e	0Y	0e	0e
0*"	0+"	1Z	0g	0+"
0+"	#2700	0e	1h	#4800
09"	0.	1g	1z	0.
0?"	1N	0h	0{	1e
1@"	1e	1{	1\$"	1+"
0A"	1+"	1&"	0+"	
1D"	#2800	0'"	0R"	
0E"	1.	0+"	0X"	
1F"	0N	0^"	1Y"	
1G"	0e	1_"	0Z"	
#1900	0+"	#3400	1J"	
0.	#2900	10	0^"	
1N	0.	0f	1_"	
1e	1N	0%"	#3800	
1+"	1e	#3500	0.	
#2000	1+"	0.	1e	
1#	#3000	1N	1+"	
1.	1.	1^	#3900	
0N	0N	0_	1.	
0e	0e	0b	0e	
0z	0+"	1c	0+"	
0\$"	#3100	1e	#4000	
0+"	00	0i	0.	
#2100	1f	1j	1e	
0.	1%"	0k	1+"	
1N	#3200	1x	#4100	
1e	0.	0y	1.	
1+"	1N	0l	0e	