VLSI Testing Homework 3

系所：資應所

姓名：林元泰 / Yuan-Tai Lin

學號：109065520

1. Derive the logic circuit of a 16-bit adder

Input : 2 16-bits data

Output : 1 17-bits result

Synthesis result of my adder with period = 40ns by

Design compiler :

```
Number of ports:                          105
Number of nets:                           152
Number of cells:                           54
Number of combinational cells:             35
Number of sequential cells:                17
Number of macros/black boxes:               0
Number of buf/inv:                          1
Number of references:                       4

Combinational area:               371.851194
Buf/Inv area:                       2.116800
Noncombinational area:            263.894401
Macro/Black Box area:               0.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:                  635.745595
Total area:                 undefined
```

```
Point                                   Incr       Path
------------------------------------------------------------
clock clk (rise edge)                   0.00       0.00
clock network delay (ideal)             0.00       0.00
result_reg[0]/CK (DFFRQX2)              0.00       0.00 r
result_reg[0]/Q (DFFRQX2)               0.21       0.21 r
U21/Y (AO22X2)                          0.07       0.28 r
result_reg[0]/D (DFFRQX2)               0.00       0.28 r
data arrival time                                  0.28

clock clk (rise edge)                  40.00      40.00
clock network delay (ideal)             0.00      40.00
result_reg[0]/CK (DFFRQX2)              0.00      40.00 r
library setup time                     -0.15      39.85
data required time                                39.85
------------------------------------------------------------
data required time                                39.85
data arrival time                                 -0.28
------------------------------------------------------------
slack (MET)                                       39.57
```

2. My BISTed circuit

A. top.v : Connect all modules.

B. adder.v : Add two 16-bits data into 17-bits result.

C. LFSR.v :

   Characteristic polynomial: g(x) = x^7 + x^1 + 1

   I use 6 LFSRs to generate two inputs of adder.v. Each LFSR has a unique random seed.

D. LFSR_manager.v :

   I use 2 LFSR_managers to generate 2 16-bits input for adder.

   The LFSR_manager combine 3 output of LFSRs into 16-bits data.

E. MISR.v

   The MISR compress the 17-bits result of adder into 10-bits signature.

My golden signature:

Signature	1 : 807	Signature	37 : 911
Signature	2 : 399	Signature	38 : 966
Signature	3 : 520	Signature	39 : 514
Signature	4 : 322	Signature	40 : 798
Signature	5 : 478	Signature	41 : 654
Signature	6 : 1010	Signature	42 : 843
Signature	7 : 283	Signature	43 : 421
Signature	8 : 680	Signature	44 : 755
Signature	9 : 883	Signature	45 : 412
Signature	10 : 731	Signature	46 : 267
Signature	11 : 657	Signature	47 : 676
Signature	12 : 361	Signature	48 : 626
Signature	13 : 977	Signature	49 : 597
Signature	14 : 1003	Signature	50 : 70
Signature	15 : 22	Signature	51 : 868
Signature	16 : 10	Signature	52 : 718
Signature	17 : 801	Signature	53 : 132
Signature	18 : 977	Signature	54 : 384
Signature	19 : 9	Signature	55 : 226
Signature	20 : 321	Signature	56 : 946
Signature	21 : 994	Signature	57 : 506
Signature	22 : 783	Signature	58 : 28
Signature	23 : 169	Signature	59 : 9
Signature	24 : 600	Signature	60 : 802
Signature	25 : 361	Signature	61 : 940
Signature	26 : 721	Signature	62 : 536
Signature	27 : 648	Signature	63 : 12
Signature	28 : 358	Signature	64 : 295
Signature	29 : 946	Signature	65 : 725
Signature	30 : 502	Signature	66 : 939
Signature	31 : 284	Signature	67 : 245
Signature	32 : 174	Signature	68 : 893
Signature	33 : 152	Signature	69 : 763
Signature	34 : 365	Signature	70 : 443
Signature	35 : 433	Signature	71 : 509
Signature	36 : 731	Signature	72 : 540

Signature 73 : 49
Signature 74 : 532
Signature 75 : 521
Signature 76 : 838
Signature 77 : 929
Signature 78 : 751
Signature 79 : 948
Signature 80 : 252
Signature 81 : 416
Signature 82 : 749
Signature 83 : 405
Signature 84 : 746
Signature 85 : 116
Signature 86 : 887
Signature 87 : 991
Signature 88 : 302
Signature 89 : 945
Signature 90 : 732
Signature 91 : 146
Signature 92 : 108
Signature 93 : 854
Signature 94 : 232
Signature 95 : 111
Signature 96 : 855
Signature 97 : 464
Signature 98 : 488
Signature 99 : 535
Signature 100 : 841
Signature 101 : 675
Signature 102 : 78
Signature 103 : 67
Signature 104 : 545
Signature 105 : 17
Signature 106 : 547
Signature 107 : 814
Signature 108 : 693
Signature 109 : 888
Signature 110 : 506
Signature 111 : 831
Signature 112 : 222
Signature 113 : 907
Signature 114 : 194
Signature 115 : 127
Signature 116 : 575
Signature 117 : 349
Signature 118 : 686
Signature 119 : 153
Signature 120 : 623
Signature 121 : 87
Signature 122 : 876
Signature 123 : 759
Signature 124 : 638
Signature 125 : 570
Signature 126 : 861
Signature 127 : 972

3. Test my adder whether it can operate at speed of 1 GHz.

Synthesis my adder with period = 1ns

```
# add your design here
read_file -format verilog {./adder.v }

create_clock -name "clk" -period 1 -waveform { 0 0.5} { clk }
set_fix_hold clk
set_dont_touch_network clk
compile -exact_map

write_file -format verilog -hierarchy -output adder_syn.v
write_sdf -version 1.0 adder_syn.sdf
uplevel #0 { report_timing -path full -delay max -nworst 1 -max_paths 1 -signifi
cant_digits 2 -sort_by group }
report_area
exit
```

<adder.dc>

```
Point                                   Incr       Path
-----------------------------------------------------------
clock clk (rise edge)                   0.00       0.00
clock network delay (ideal)             0.00       0.00
result_reg[0]/CK (DFFRQX2)              0.00       0.00 r
result_reg[0]/Q (DFFRQX2)               0.21       0.21 r
U21/Y (AO22X2)                          0.07       0.28 r
result_reg[0]/D (DFFRQX2)               0.00       0.28 r
data arrival time                                  0.28

clock clk (rise edge)                   1.00       1.00
clock network delay (ideal)             0.00       1.00
result_reg[0]/CK (DFFRQX2)              0.00       1.00 r
library setup time                     -0.15       0.85
data required time                                 0.85
-----------------------------------------------------------
data required time                                 0.85
data arrival time                                 -0.28
-----------------------------------------------------------
slack (MET)                                        0.57
```

< Timing report >

Simulation result:

The adder is a netlist. Rest of modules are RTLs.

Testbench timescale	100ns/1ps	10ns/1ps	1ns/1ps
Result	pass	pass	pass

4. The maximum operating speed of my BISTed adder design

我把所有的 module 放進同一個 file(BISTed_adder.v)進行 synthesis，從 1 GHz 開始測試 synthesis 的結果。

Synthesis with period = 1ns => fail

```
# add your design here
read_file -format verilog {./BISTed_adder.v }

create_clock -name "clk" -period 1 -waveform { 0 0.5 } { clk }
set_fix_hold clk
set_dont_touch_network clk
compile -exact_map

write_file -format verilog -hierarchy -output BISTed_adder_syn.v
write_sdf -version 1.0 BISTed_adder_syn.sdf
uplevel #0 { report_timing -path full -delay max -nworst 1 -max_paths 1 -signifi
cant_digits 2 -sort_by group }
report_area
exit
```

< BISTed_adder.dc>

```
--------------------------------------------------------------------------
data required time                                                 0.91
data arrival time                                                 -0.98
--------------------------------------------------------------------------
slack (VIOLATED)                                                  -0.07
```

< Timing report >

Synthesis with period = 1.1ns => fail

```
# add your design here
read_file -format verilog {./BISTed_adder.v }

create_clock -name "clk" -period 1.1 -waveform { 0 0.55 } { clk }
set_fix_hold clk
set_dont_touch_network clk
compile -exact_map

write_file -format verilog -hierarchy -output BISTed_adder_syn.v
write_sdf -version 1.0 BISTed_adder_syn.sdf
uplevel #0 { report_timing -path full -delay max -nworst 1 -max_paths 1 -signifi
cant_digits 2 -sort_by group }
report_area
exit
```

< BISTed_adder.dc>

```
--------------------------------------------------------------------------
data required time                                                  1.01
data arrival time                                                  -1.02
--------------------------------------------------------------------------
slack (VIOLATED)                                                   -0.02
```

< Timing report >

Synthesis with period = 1.2ns => success

```
# add your design here
read_file -format verilog {./BISTed_adder.v }

create_clock -name "clk" -period 1.2 -waveform { 0 0.6 } { clk }
set_fix_hold clk
set_dont_touch_network clk
compile -exact_map

write_file -format verilog -hierarchy -output BISTed_adder_syn.v
write_sdf -version 1.0 BISTed_adder_syn.sdf
uplevel #0 { report_timing -path full -delay max -nworst 1 -max_paths 1 -signifi
cant_digits 2 -sort_by group }
report_area
exit
```

<BISTed_adder.dc>

```
--------------------------------------------------------------------------
data required time                                                  1.06
data arrival time                                                  -1.06
--------------------------------------------------------------------------
slack (MET)                                                         0.00
```

< Timing report >

Simulate the netlist which period = 1.2ns is successful.

```
                Always blocks:           2       2
                Initial blocks:          3       3
                Cont. assignments:       6     151
                Pseudo assignments:      8       8
                Timing checks:         774      84
                Simulation timescale:  1ps
        Writing initial simulation snapshot: worklib.testbench:v
Loading snapshot worklib.testbench:v ..................... Done
*Novas* Loading libsscore_ius111.so
ncsim> source /usr/cad/cadence/INCISIV/INCISIVE_15.20.084/tools/inca/files/ncsimrc
ncsim> run
Novas FSDB Dumper for IUS, Release 2012.01, Linux x86_64/64bit, 01/12/2012
Copyright (C) 1996 - 2012 by SpringSoft, Inc.
*********************************************************************
*  WARNING -                                                        *
*  The simulator version is newer than the FSDB dumper version which  *
*  may cause abnormal behavior. Please contact SpringSoft support at  *
*  support@springsoft.com for assistance.                           *
*********************************************************************
*Novas* : Create FSDB file 'adder.fsdb'
*Novas* : Begin traversing the scopes, layer (0).
*Novas* : End of traversing.
Start
Signature           1 :  807
Signature           2 :  399
Signature           3 :  520
Signature           4 :  322
Signature           5 :  478
Signature           6 : 1010
```

The shortest period is 1.2ns => The max speed of my design is 0.83 GHz.