VLSI Test and Validation

Final Test, 6/5/2007

1. Answer the following questions: (20%)
   (a) For a digital signature analyzer implemented using an n-stage LFSR, what is the aliasing probability?
   (b) Can a single ring boundary scan structure implement different test instructions for different chips at the same time? Can it implement different capture, shift, or update operations for different chips at the same time?
   (c) Given a march element ⇑(r0, r0, w1), is it the same as the combination of two march elements ⇑(r0) and ⇑(r0, w1)? Do they detect the same faults? Why?
   (d) Sometimes, a circuit might generate unknown value in the outputs during BIST. What problem might cause if this occurs? How to fix this problem? Show by an example.
   (e) A CBILBO register can work as test pattern generator and MISR when testing a circuit. Compared with implementing a single CBILBO register by two separate registers (one is used for TPG while the other for MISR), Can CBILBO save test application time for BIST? Can CBILBO save hardware overhead? Can CBILBO increase fault coverage?

2. Given a boundary scan structure with two chips as shown in Fig. 1, how many test cycles are needed to shift a 10-bit test instruction into the Instruction Register of the second chip? Assume you start from the Test-Logic-Reset state and after the instruction is shifted in, the TAP will be in Select-DR-Scan state. Fig. 2 gives all diagrams you need for boundary scan design for Questions 2 and 3. Use the following representation to give your answer:

   1 (to state A) +

   1 (to state B)*20+

   ..... (16%)

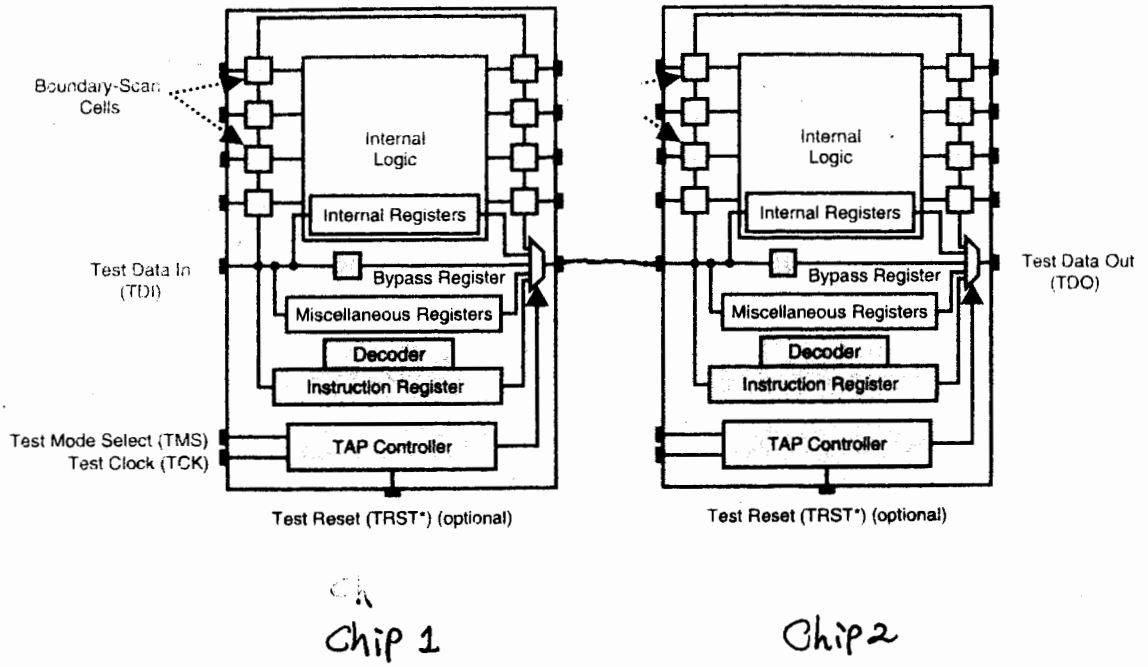


Fig. 1.

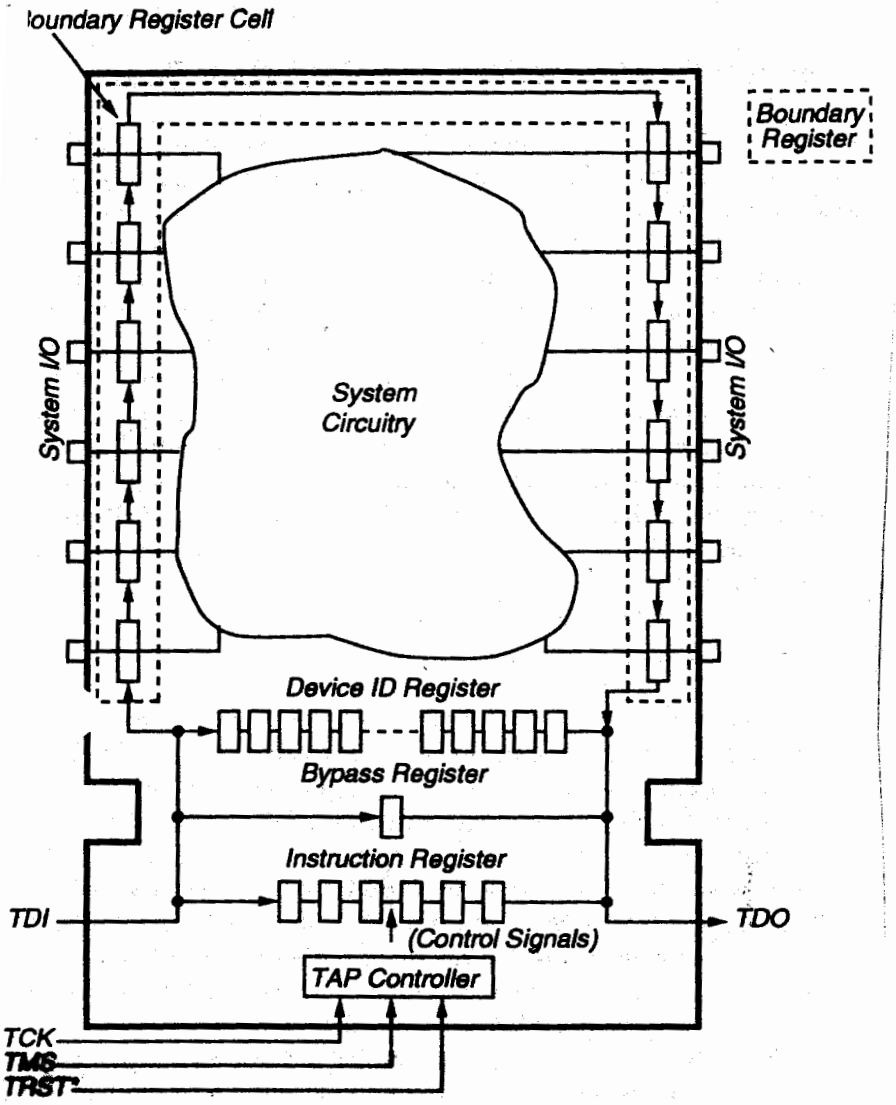
3. Figure 3 shows a boundary scan structure with scan testing and memory BIST structure. Assume the instruction register has 10 bits, and the compressor is an LFSR with 20 bits. Further we assume that 1000 test patterns will be applied to the memory array, and you start from the Test-Logic-Reset. How many test cycles, at least, will be required to perform the memory BIST process using the boundary scan structure? Your test process must includes: (a) scan in the instruction to the instruction register, (b) scan in the initial seed to the LFSR (compressor), (c) perform the memory BIST, (d) scan out the digital signature stored in the compressor (LFSR), and finally (e) drive TAP controller to the Test-Logic-Reset state. Use the following representation to give your answer:

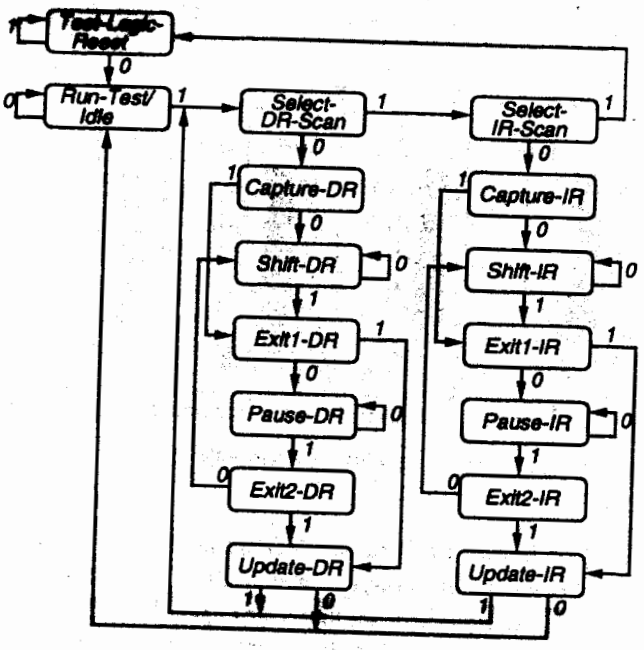
   1 (to state A) +

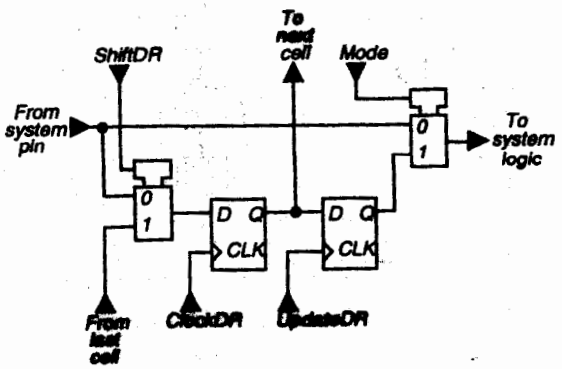
   1 (to state B)*20+

   ..... (16%)

Fig. 2.


oundary Register Cell
Boundary Register
System I/O
System Circuitry
System I/O
Device ID Register
Bypass Register
Instruction Register
TDI
TDO
(Control Signals)
TAP Controller
TCK
TMS
TRST*


Test-Logic-Reset
Run-Test/ Idle
Select-DR-Scan
Capture-DR
Shift-DR
Exit1-DR
Pause-DR
Exit2-DR
Update-DR
Select-IR-Scan
Capture-IR
Shift-IR
Exit1-IR
Pause-IR
Exit2-IR
Update-IR


ShiftDR
To next cell
Mode
From system pin
To system logic
D Q
CLK
From last cell
ClockDR
UpdateDR

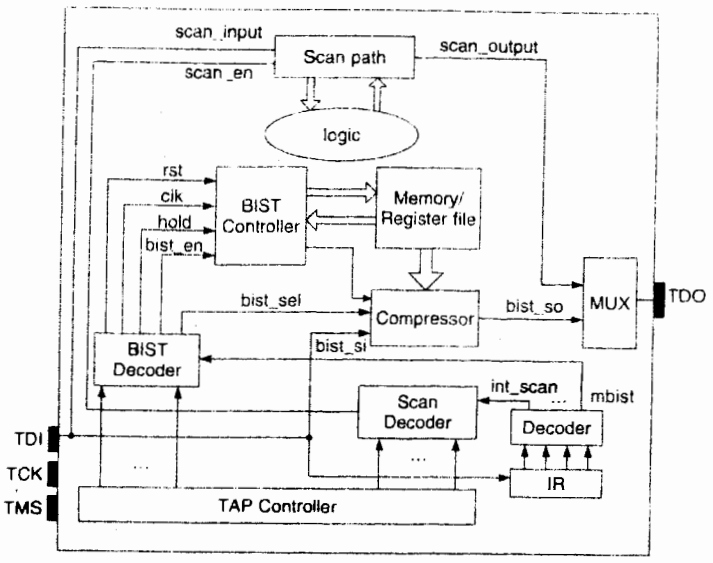


Fig. 3.

4. Compare the performance of a STUMPS design and a BILBO design. Assume that both designs operate at 200MHz and the circuit under test has 100 scan chains each having 1000 scan cells as shown in Fig. 4(a). (a) Compute the test time for each design when 100,000 test patterns are applied. For BILBO, we assume that MISR achieved by each scan chain in BILBO mode can work as test patterns simultaneously as shown in Fig. 4(b) (b) In general, the shift (scan) speed is much slower than a circuit's operating speed. Assume that the shift speed is 20MHz, and compute the test time for the STUMPS again. (16%)

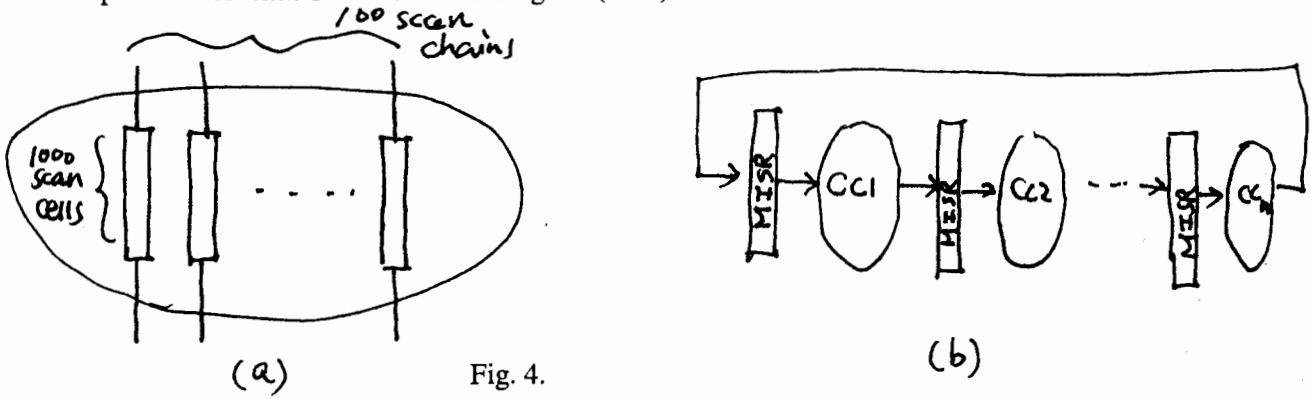


Fig. 4.

5. An idempotent coupling fault (CFid) is where a rising or falling transition on aggressor cell Ca sets the victim cell Cv to 0 or 1. This can be denoted by four different fault primitives: <↑;0>, <↑;1>, <↓;0>, and <↓;1>. Assume we have a memory array with 100 words and each word contains 1 bit only. For the March C- algorithm shown in Fig. 5(a), give the first march element(s) that will sensitize and observe each fault primitive. Use a table like the one shown in 5(b) to give your answer. No grade will be given if you do not summarize your results in the table. (16%)

M_0 M_1 M_2 M_3 M_4 M_5

$\Updownarrow(w0)$; $\Uparrow(r0,w1)$; $\Uparrow(r1,w0)$; $\Downarrow(r0,w1)$; $\Downarrow(r1,w0)$; $\Updownarrow(w0)$

Fig. 5(a)

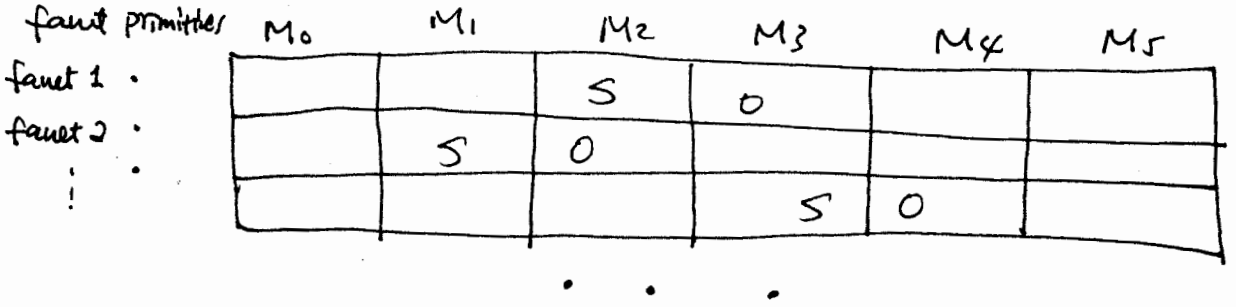


Fig. 5(b)

6. The state coupling fault (CFst) is where the aggressor cell Ca in state x will force the victim cell Vc to state y. There are four different fault primitives: <0;0>, <0:1>, <1,0>, and <1,1> which are all different possibilities of <x; y>. You are given March X shown in Fig. 6(a). Answer the following questions: (16%)

(a) Can March X detect all CFst between different words if each word is assume to contain only one bit? If not, which CFst primitive(s) cannot be detected if the address of Ca is larger than that of Cv.

(b) How many background data will be required if CFst can only occur on adjacent cells in the same word (but no CFst in different words), and each word contains 8 bits? Give these background data.

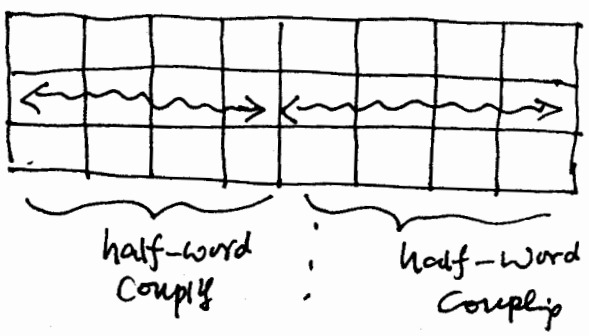
(c) How many background data will be required if CFst can only occur on any pair of cells in the same word (but no CFst in different words), and each word contains 8 bits? Give these background data.

(d) How many background data will be required if CFst can only occur on any pair of cells in the same half word as shown in Fig, 6(b) (but no CFst in different words), and each word contains 8 bits? Give these background data.

$\Updownarrow(w0)$; $\Uparrow(r0,w1)$; $\Downarrow(r1,w0)$; $\Updownarrow(r0)$

(a)

Fig. (6)



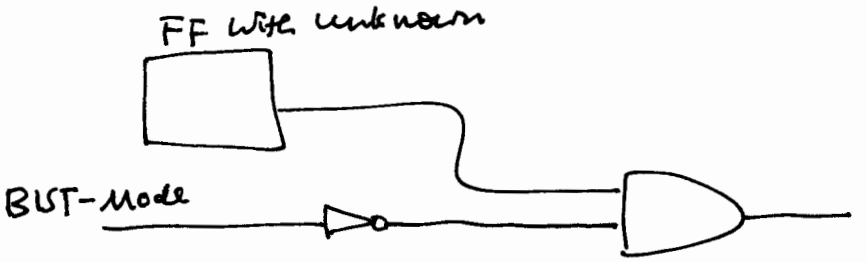
(1) (a) 2^{-n}

(b) yes, no

(c) NO, NO

(d) Unknown causes unpredictable digital signature. We can use a MUX or a gate to choose a specific logic during BIST.

For example:



(2) There are many solutions:

Solution 1: Assume chip1 and chip2 use diffent TMS and the necessary instruction to Select Bypass register has been initialized in chip1.

In this case, the bypass register of the first chip can be treated as part of the IR in the 2nd chip.

Also assume, the necessary TMS for chip1 is synchronized with chip2.

1 (to ~~Test-Logic-Reset~~ Run-Test/Idle) +

1 (to Select-DR-Scan) +

1 (to Select-IR-Scan) +

1 (to Capture-IR) +

1 (to Shift-IR) *11 + [2]

1 (to Exit1-IR) +

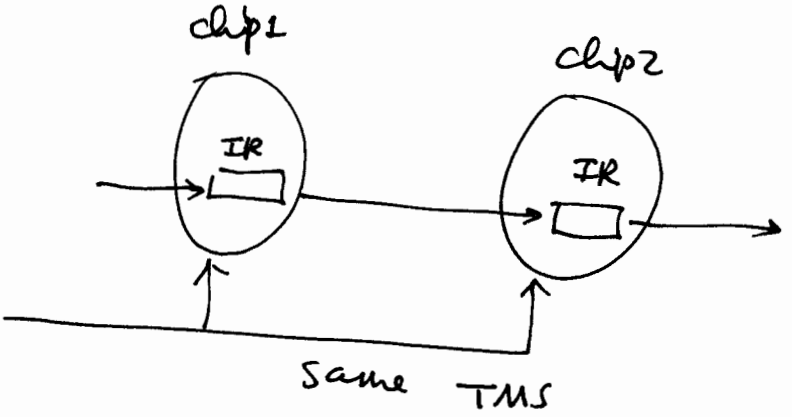
1 (to Update-IR) +

1 (to Select-DR-Scan)

= 18 test cycles

* This is not the best solution due to too many assumptions.

Solution II:



- Shift instruction to ~~the~~ IR of chip 1 and then IR of chip 2.

The solution is the same as above, but

α is replaced by

1 (to Shift-IR) * 20 ← This is the most natural Solution!!!

∴ the test cycle number is 27.

(3)

4 { 1 (to run-test/Idle) +
1 (to Select-DR-Scan) +
1 (to Select-IR-Scan) +
1 (to Capture-IR) +

10 { 1 (to Shift-IR) * 10 +

3 { 1 (to Exit-1-IR) +
1 (to update-IR) +
1 (to Select-DR-Scan) +

23 { 1 (to Capture-DR) +
1 (to Shift-DR) * 20 +
1 (to Exit-1-DR) +
1 (to update-DR) +

1000 { 1 (to Run-test/Idle) * 1000 +

2 { 1 (to Select-DR-Scan) +
1 (to Capture-DR) +

20 { 1 (to Shift-DR) * 20 +

5 { 1 (Exit-1-DR) +
1 (to update-DR) +
1 (to Select-DR-Scan) +
1 (to Select-IR-Scan) +
1 (to Test-logic-Reset)

Total # of test cycles:

4 + 10 + 3 + 23 + 1000 +
2 + 20 + 5
= 1067

(4)

(a)

STUMPS:

shift ↙ capture ↙

$$\text{test time} = \left(100{,}000 \times 1000 + 100{,}000\right) \times \frac{1}{200M}$$

$$= 0.5005\text{s}.$$

BILBO:

# of test patterns ↙ ; scan in (out) initial pattern (MISR value) ↙

$$\text{test time} = 100{,}000 \times \frac{1}{200M} + 2 \times 1000 \times 100 \times \frac{1}{200M}$$

$$= 1.5\ \text{ms}$$

(b)

STUMS:

~~Scan~~ shift time ↙

$$\text{test time} = 100{,}000 \times 1000 \times \frac{1}{20M} +$$

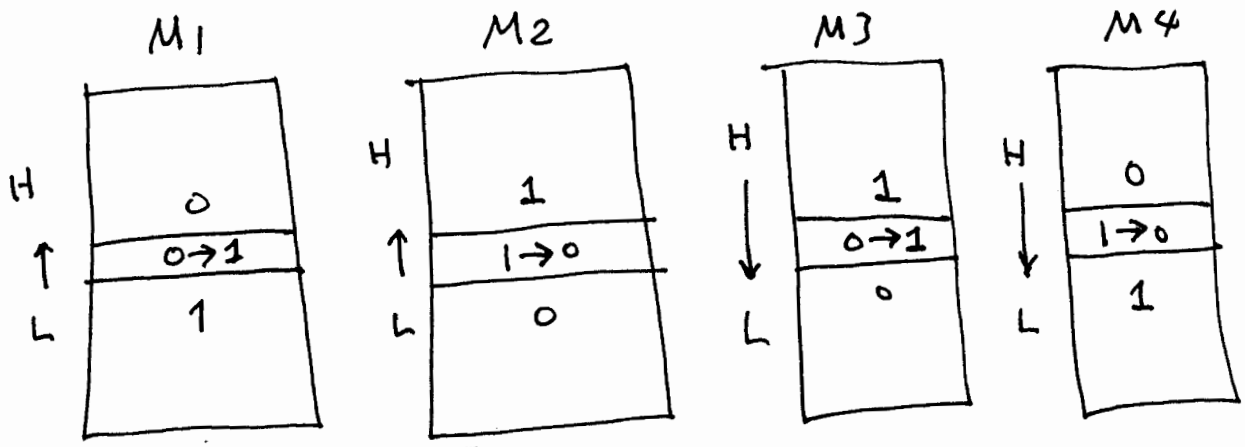
$$100{,}000 \times \frac{1}{200M}$$

↑ capture time

$$= 5.005\text{s}$$

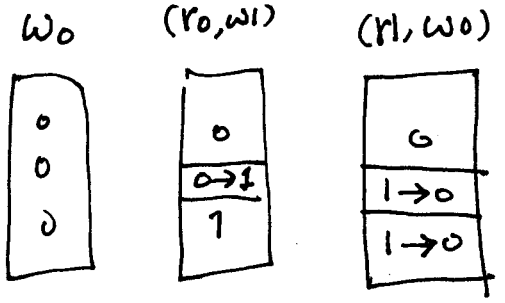
(P1)

(5)



	M0	M1	M2	M3	M4	M5	
<↑;0>				S	O		Add (aggressor) <
<↑;1>		S,O					Add (victim)
<↓;0>			S,O				
<↓;1>					S	O	
<↑;0>		S	O				Address (aggressor) >
<↑;1>				S,O			
<↓;0>					S,O		Address (victim)
<↓;1>		O	O	S	O		

(6) (a)



(0,1) (0,0) <1,0>

(1,0) <0,0>, <0,1>

∴ <1,1> is not detected

(b) 2 BG.

1st { 0 = 00..0, 1 = 11..1 }

2nd { 0 = 0101..01, 1 = 1010..10 }

(c) 4 BG

	0	1
1st	00000000	11111111
2nd	01010101	10101010
3rd	00110011	11001100
4th	00001111	11110000

(d) 3 BG

	0	1
1st	0000 0000	1111 1111
2nd	0101 0101	1010 1010
3rd	0011 0011	1100 1100