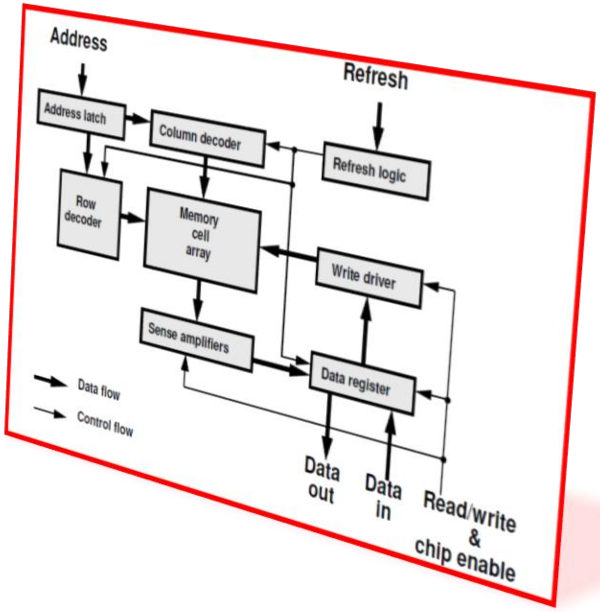
Memory Testing

- Introduction
- Memory Fault Model
- **Memory Test Algorithms**
  - Classical algorithms
  - **March algorithms**
    - **MATS (1979), MATS+(1983), MATS++(1991)**
    - **March X**
    - **March C (1982), March C-(1991)**
- Memory Fault Simulation (*not in exam)
- Memory Test Generation (*not in exam)
- Memory BIST (*not in exam)



March Test Algorithms

- ***March test algorithm*** **has a sequence of *march elements***
- **Each *march element* is specified by**
  1. ***Operations*** **and data :**
     - **Reading an expected 0 (r0); reading an expected 1 (r1)**
     - **Writing 0 to a cell (w0); writing 1 to a cell (w1)**
  2. ***Address sequence*** **:**
     - **⇑: address changes in ascending order**
     - **⇓ : address changes in descending order**
     - **⇕: address sequence either ⇑ or ⇓**
- **Example: {⇕ (*w*0); ⇑(*r*0,*w*1); ⇓ (*r*1)}**
  - **has 3 march elements, separated by ;**
  - **complexity = 4*N***

March Tests are Linear Time

QUIZ

Q: {⇕ w0); ⇑(r0, w1);⇑(r1, w0); ⇕ (r0); ⇓(r0, w1); ⇓(r1, w0); ⇕ (r0)}

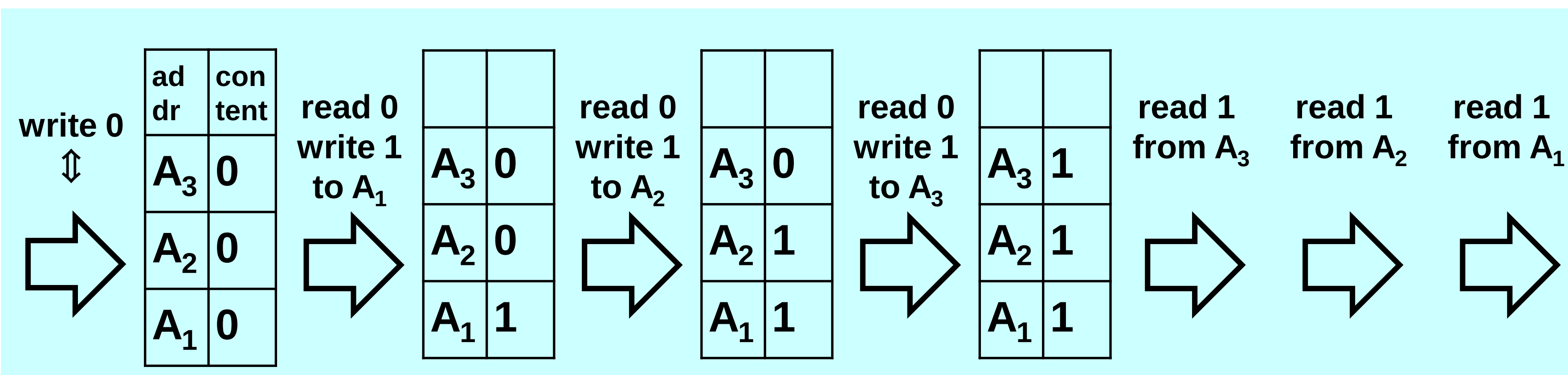
How many march elements? Time complexity=?

ANS:

MATS [Nair 1979]

- *Modified Algorithmic Test Sequence* (MATS)
- 3 march elements
  - *{⇕ (w0); ⇑(r0,w1); ⇓ (r1)}*
- Detects all SAF, half TF
- Complexity *4N*

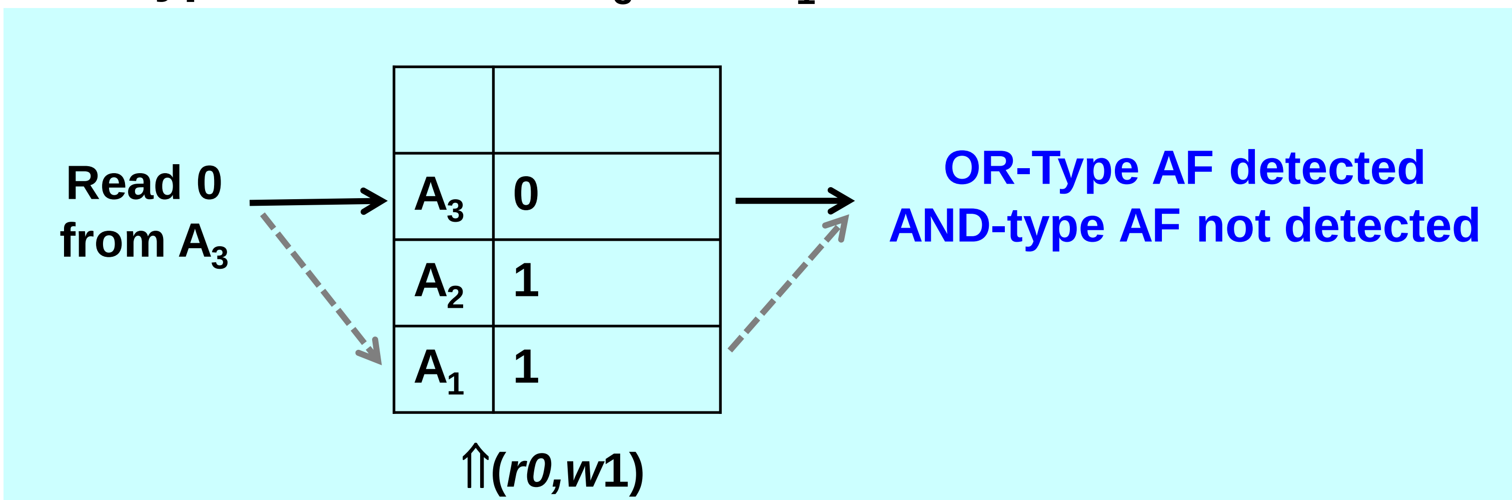
MATS
1. Write zero to all cells
   in ascending or descending address order
2. Read zero and then write one
   in ascending address order
3. Read one in descending address order



How about AF?

MATS (2)

- **Can MATS algorithm detect AF?**
  - ***{⇕ (w0); ⇕ (r0,w1); ⇕(r1)}***
- **Example: OR-type AF between A_3 and A_1**



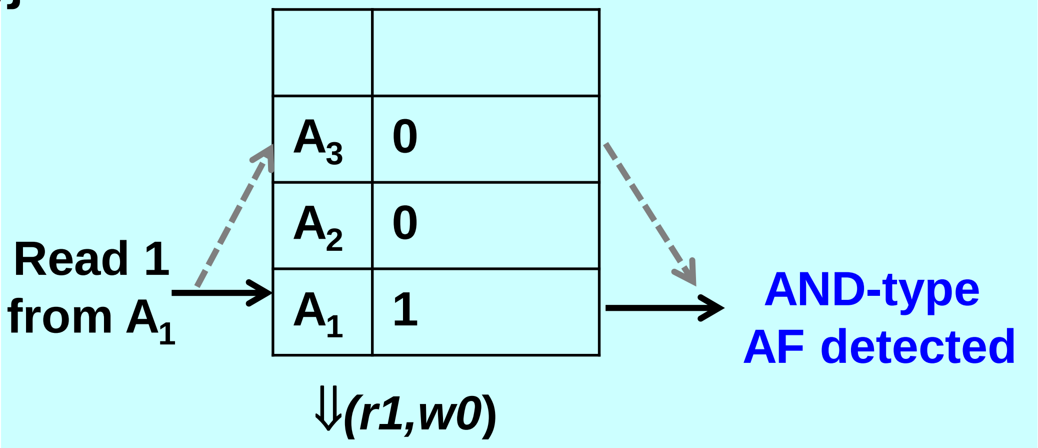
- **How to fix it? Reverse MATS. Detects AND-type but not OR-type**
  - ***{⇕ (w1); ⇕ (r1,w0); ⇕ (r0)}***

	SAF	AF	TF	CF	Complexity
MATS	D	1/2	1/2	-	4*N*

D=all detected
½=half detected
- = not detected

MATS+ [Abdir 1983]

- OR-type MATS
  - $\{\Updownarrow(w0); \Updownarrow(r0,w1); \Updownarrow(r1)\}$
- AND-type MATS
  - $\{\Updownarrow(w1); \Updownarrow(r1,w0); \Updownarrow(r0)\}$
- MATS+ combines both AND-type OR-type MATS
  - $\{\Updownarrow(w0); \Uparrow(r0,w1); \Downarrow(r1,w0)\}$
- Detects all SAF and AF
- Detect half TF
- Complexity $5N$



	SAF	AF	TF	CF	Complexity
MATS+	D	D	1/2	-	$5N$

MATS++ [Goor 1991]

- **Original MATS+** $\{\Updownarrow(w0); \Uparrow(r0,w1); \Downarrow(r1,w0)\}$
- **MATS++ Algorithm** $\{\Updownarrow(w0); \Uparrow(r0,w1); \Downarrow(r1,w0,r0)\}$
- **Detects all SAF, AF and TF**
  - **Similar to MATS+, but detects all TF**
- **Complexity 6*N***

	SAF	AF	TF	CF	Complexity
MATS++	D	D	D	-	6*N*

How about CF?

QUIZ

Q: Can MATS++ {⇕ (*w*0); ⇑(*r*0,*w*1); ⇓(*r*1,*w*0,*r*0)} detect CF_{in} <↓; ∀/ ↕ > ?

A1 is aggressor, A3 is victim

ANS:

A_3 (V)	1
A_2	1
A_1 (A)	1

⇕ (*w*0); ⇑(*r*0,*w*1);

A_3 (V)	0
A_2	0
A_1 (A)	1

⇓(r1,w0, r0)

A_3 (V)	?
A_2	?
A_1 (A)	?

NOT detected

March X

*Called *March X* because it's not published

- **MATS++ algorithm {⇕ (w0); ⇑(r0,w1); ⇓(r1,w0, r0)}**
- **March X algorithm {⇕ (w0); ⇑(r0,w1); ⇓(r1,w0); ⇕(r0)}**
- **Detects AF, SAF, TF, CF_{in}**
- **Example:**
  - **CF_{in} <↓; ∀/ ↕ > between A_1(A) and A_3 (V)**

A_3 (V)	1
A_2	1
A_1 (A)	1

⇕ (w0); ⇑(r0,w1);

A_3 (V)	0
A_2	0
A_1 (A)	1→0

⇓(r1,w0)

green = activation

A_3 (V)	0/1
A_2	0
A_1 (A)	0

⇕(r0)

blue = detection

March X (cont'd)

- **March X algorithm {⇕ (w0); ⇑(r0,w1); ⇓(r1,w0); ⇕ (r0)}**
  - **All four cases CF_{in} are detected, still 6*N* but better than MATS++**

<↑ ; ∀/ ↕ >

V	0/1
A_2	0
A	0→1

⇑(r0,w1)

<↑ ; ∀/ ↕ >

A	0→1
A_2	1
V	1/0

⇑(r0,w1); ⇓(r1,w0)

< ↓; ∀/ ↕ >

A	1→0
A_2	1
V	1/0

⇓(r1,w0)

<↓; ∀/ ↕ >

V	0/1
A_2	0
A	1→0

⇓(r1,w0);⇕ (r0)

green = activation
blue = detection

	SAF	AF	TF	CF	Complexity
March X	D	D	D	CF_{in}	6*N*

March C [Marinescu 1982]

- March C = two March X combined in opposite address order
  - {⇕ w0); ⇑(r0, w1); ⇑(r1, w0); ⇕ (r0); ⇓(r0, w1); ⇓(r1, w0); ⇕ (r0)}
- Detects AF, SAF, TF, & all CF
- March C detects all eight cases of CF_{id}
  - A_1 is aggressor and A_3 is victim
  - The other four cases (A_1 is V, A_3 is A) are symmetric

<↑ ; 0/1 >

V	0/1
A_2	0
A	0→1

⇑(r0,w1)

<↑ ; 1/0 >

V	1/0
A_2	1
A	0→1

⇓(r0,w1) ; ⇓(r1, w0)

< ↓; 1/0 >

V	1/0
A_2	1
A	1→0

⇑(r1,w0)

<↓; 0/1 >

V	0/1
A_2	0
A	1→0

⇓(r1,w0) ; ⇕ (r0)

QUIZ

- **Q: Can March C detects CF_{st} ?**
  - **{$\Updownarrow$ w0); $\Uparrow$(r0, w1); $\Uparrow$(r1, w0); $\Updownarrow$ (r0); $\Downarrow$(r0, w1); $\Downarrow$(r1, w0); $\Updownarrow$ (r0)}**
  - **consider A_1 = Aggressor and A_3 = Victim**

ANS:

<1 ; 0/1 >

V	?
A_2	?
A	?

$\Uparrow$(r0,w1)

<1 ; 1/0 >

V	?
A_2	?
A	?

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

< 0; 1/0 >

V	?
A_2	?
A	?

$\Uparrow$(r1,w0)

<0; 0/1 >

V	?
A_2	?
A	?

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

March C Detects All CF

March C- [Goor 1991]

- **March C**
  - **{⇕ w0); ⇑(r0, w1);⇑(r1, w0); ⇕ (r0); ⇓(r0, w1); ⇓(r1, w0); ⇕ (r0)}**
- **March C- remove redundancy in March C**
  - **{⇕ (w0); ⇑(r0,w1); ⇑(r1,w0); ⇓(r0,w1); ⇓(r1,w0); ⇕ (r0)}**

- **March C- detects AF, SAF, TF, & all CF**
- **Complexity 10*N***
  - **Shortest test that detect all four faults**

	SAF	AF	TF	CF	Complexity
March C	D	D	D	D	11 *N*
March C-	D	D	D	D	10 *N*

Summary

- March Tests contains
  - march elements
    - Operations (R/W)
    - Address order
- March tests
  - Linear time
  - Good FC
- March C-
  - 10 *N*
  - Detects 4 faults

	SAF	AF	TF	CF	Complexity
MSCAN	D	-	-	-	4*N*
checkerboard	D	-	-	-	4*N*
GALPAT	D	D	D	D	$4N^2$
BUTTERFLY	D	-	D	-	5*N* log *N*
MATS	D	-	-	-	4*N*
MATS+	D	D	-	-	5*N*
MATS++	D	D	D	-	6*N*
March X	D	D	D	-	6*N*
March C	D	D	D	D	11*N*
March C-	D	D	D	D	10*N*

March Tests Very Useful in Practice

FFT

- Q: Even with linear algorithm, 28 minutes is still too long.
  - how to reduce it?

Size	N	$10N$	$N \lg N$	$N^{1.5}$	N^2
1M	0.01s	0.1s	0.2s	11s	3h
16M	0.16s	1.6s	3.9s	11m	33d
64M	0.66s	6.6s	17s	1.5h	1.43y
256M	2.62s	26s	1.23m	12h	23y
1G	10.5s	1.8m	5.3m	4d	366y
4G	42s	7m	22.4m	32d	59c
16G	2.8m	28m	1.6h	261d	936c