

# NAND Flash ECC Algorithm (Error Checking & Correction)

June 2004

Product Planning & Application Engineering Team

MEMORY DIVISION  
SAMSUNG ELECTRONICS Co., LTD

# ECC Code Generation

- ◆ **ECC code consists with 3byte per 256bytes**

- Actually 22bit ECC code per 2048bits
- 22bit ECC code = 16bit line parity + 6bit column parity

- ◆ **Data bit assignment table with ECC code**

I/O 7	I/O 6		I/O 1	I/O 0	
D(00000000,111)	D(00000000,110)		D(00000000,001)	D(00000000,000)	1st Byte
D(00000001,111)	D(00000001,110)		D(00000001,001)	D(00000001,000)	2nd Byte
D(00000010,111)	D(00000010,110)	• • • •	D(00000010,001)	D(00000010,000)	3rd Byte
D(11111110,111)	D(11111110,110)		D(11111110,001)	D(11111110,000)	255th Byte
D(11111111,111)	D(11111111,110)		D(11111111,001)	D(11111111,000)	256th Byte

- ◆ **ECC code assignment table**

I/O 7	I/O 6	I/O 5	I/O 4	I/O 3	I/O 2	I/O 1	I/O 0
<b>P64</b>	P64`	<b>P32</b>	P32`	<b>P16</b>	P16`	<b>P8</b>	P8`
<b>P1024</b>	P1024`	<b>P512</b>	P512`	<b>P256</b>	P256`	<b>P128</b>	P128`
<b>P4</b>	P4`	<b>P2</b>	P2`	<b>P1</b>	P1`	1	1

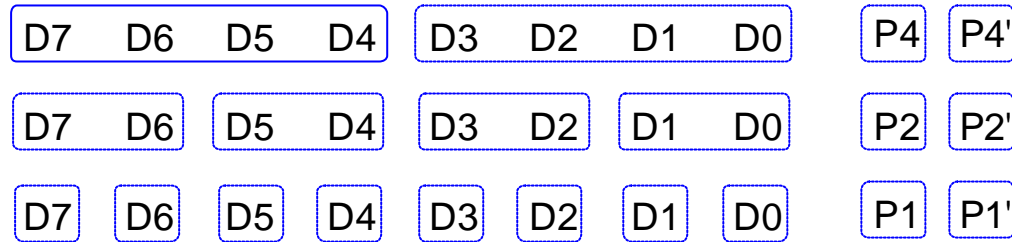
P8 ~ P1024 : Line parity  
P1 ~ P4 : Column parity

**Fail bit address offset**

=> (P1024,P512,P256,P128,P64,P32,P16,P8,P4,P2,P1)

# ECC Code Generation Example

## ◆ Parity Generation ( In case of 8bit input )



$$\begin{aligned}
 P4 &= D7(+ )D6(+ )D5(+ )D4 & P4' &= D3(+ )D2(+ )D1(+ )D0 & * (+ ) & \text{means XOR (Even parity)} \\
 P2 &= D7(+ )D6(+ )D3(+ )D2 & P2' &= D5(+ )D4(+ )D1(+ )D0 \\
 P1 &= D7(+ )D5(+ )D3(+ )D1 & P1' &= D6(+ )D4(+ )D2(+ )D0
 \end{aligned}$$

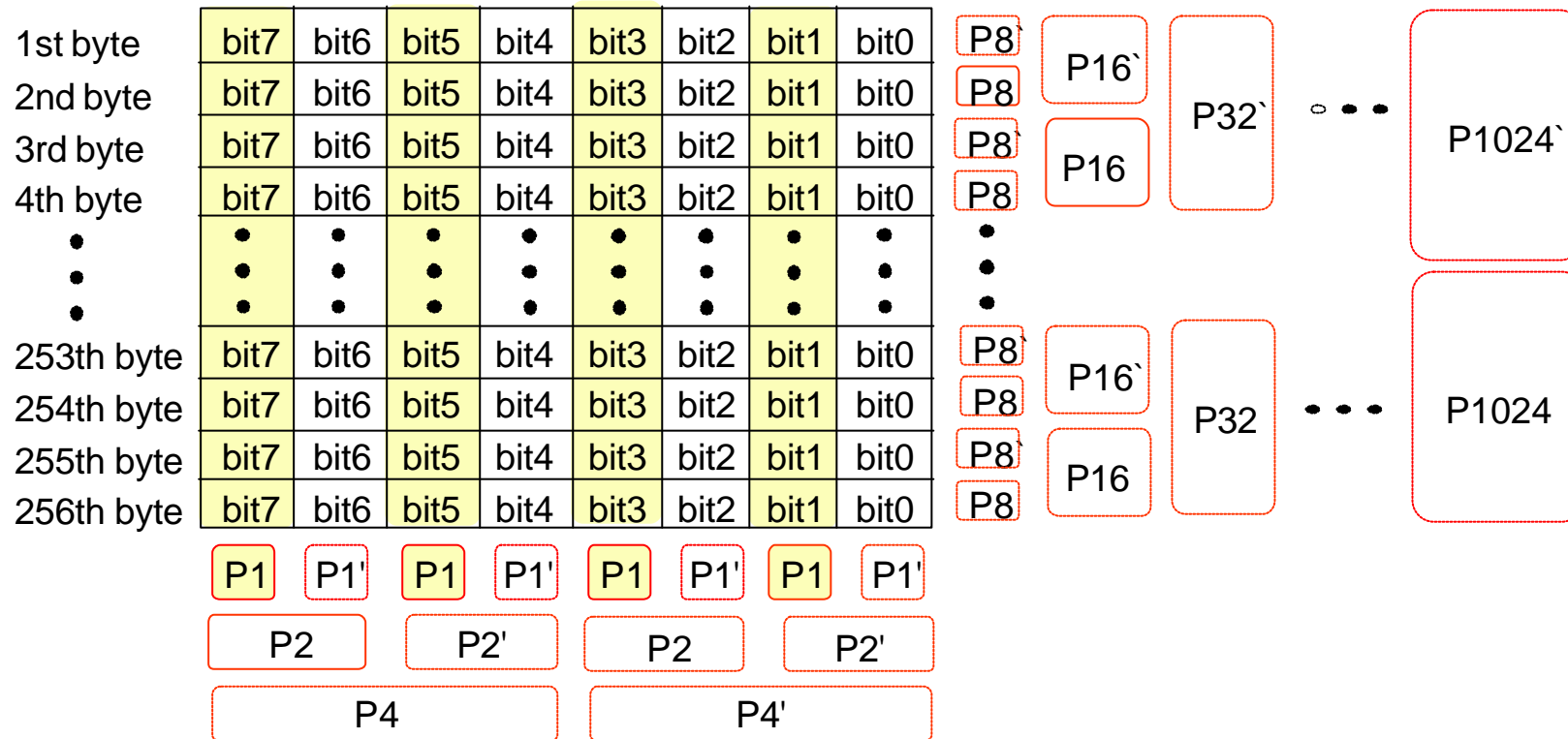
## ◆ For example ( In case of 1 bit fail )

Original data : 1 0 1 0 1 0 1 0	$P4 = 1(+ )0(+ )1(+ )0 = 0$ , $P4' = 1(+ )0(+ )1(+ )0 = 0$ $P2 = 1(+ )0(+ )1(+ )0 = 0$ , $P2' = 1(+ )0(+ )1(+ )0 = 0$ $P1 = 1(+ )1(+ )1(+ )1 = 0$ , $P1' = 0(+ )0(+ )0(+ )0 = 0$
↓	
Changed data : 1 0 1 <b>1</b> 1 0 1 0	$P4 = 1(+ )0(+ )1(+ )\mathbf{1} = \mathbf{1}$ , $P4' = 1(+ )0(+ )1(+ )0 = 0$ $P2 = 1(+ )0(+ )1(+ )0 = 0$ , $P2' = 1(+ )\mathbf{1}(+ )1(+ )0 = \mathbf{1}$ $P1 = 1(+ )1(+ )1(+ )1 = 0$ , $P1' = 0(+ )\mathbf{1}(+ )0(+ )0 = \mathbf{1}$

In here, fail bit location is column address offset ( $P4, P2, P1 = 100 = I/O4$ )

# ECC Code Generation Method (1)

## ◆ Parity Generation ( In case of 256 byte input )



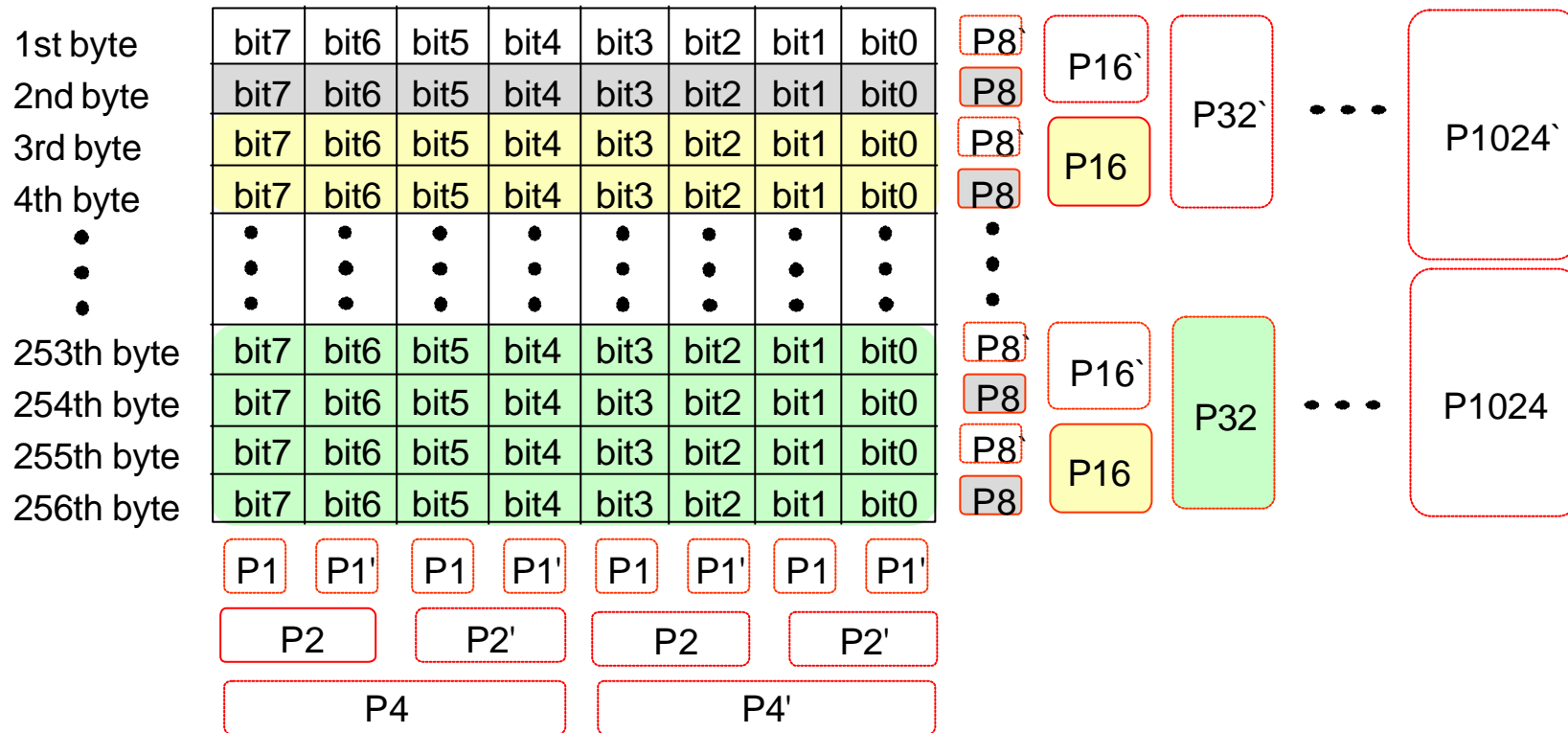
$$P1 = \text{bit7}(+) \text{bit5}(+) \text{bit3}(+) \text{bit1}(+) P1$$

$$P2 = \text{bit7}(+) \text{bit6}(+) \text{bit3}(+) \text{bit2}(+) P2$$

$$P4 = \text{bit7}(+) \text{bit6}(+) \text{bit5}(+) \text{bit4}(+) P4$$

# ECC Code Generation Method (2)

## ◆ Parity Generation ( In case of 256 byte input )

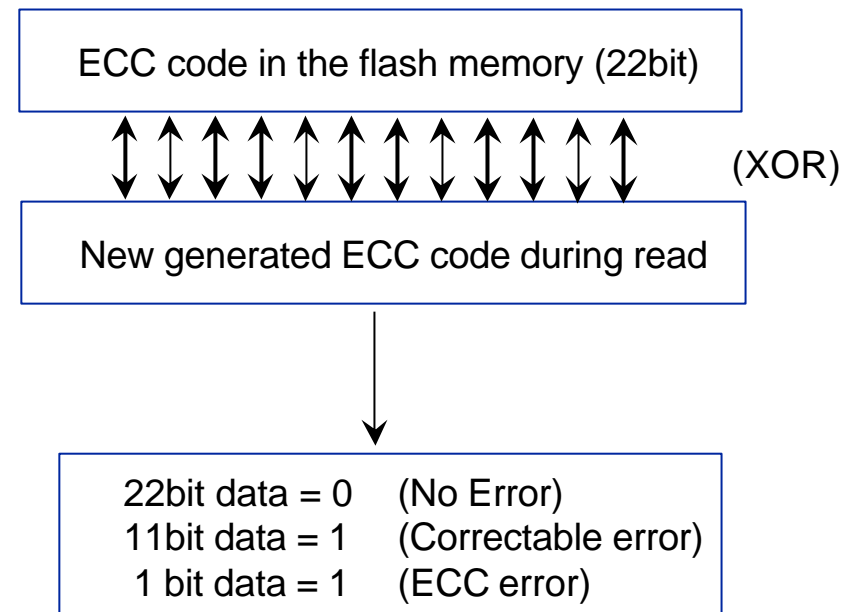
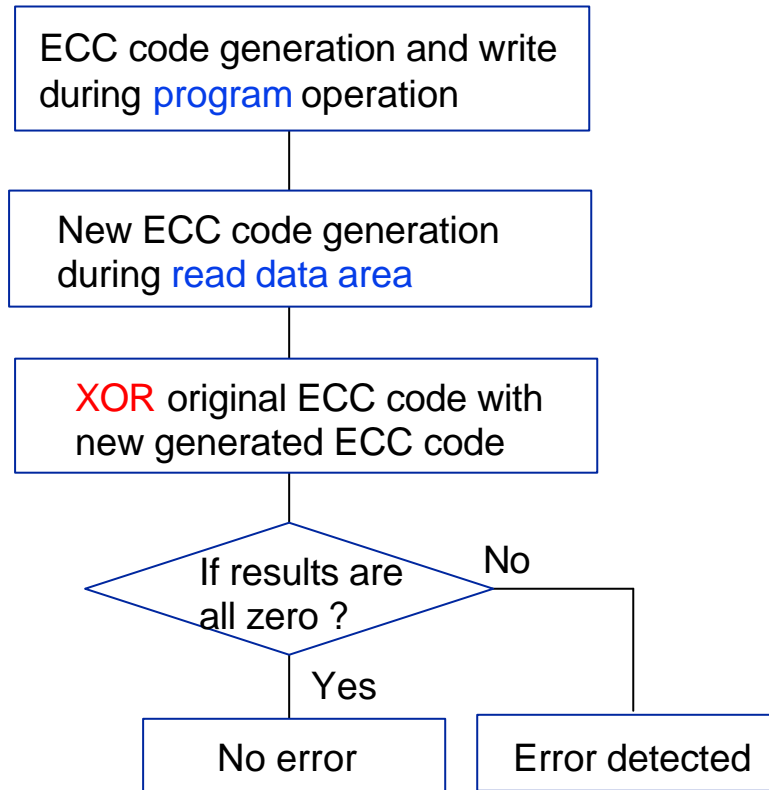


$$P8 = \text{bit7}(+)\text{bit6}(+)\text{bit5}(+)\text{bit4}(+)\text{bit3}(+)\text{bit2}(+)\text{bit1}(+)\text{bit0}(+)P8$$

⋮

# Error Detection Method

## ◆ Error Detection Sequence



# Error Detect Result

## ◆ No Error

- The result of XOR : all ECC code is `0`

## ◆ Correctable Error

- The result of XOR : total 11bits are `1`
- When main area has 1 bit error, each parity pair (ex. P8 & P8`) has 1 & 0 or 0 & 1

## ◆ ECC Error

- The result of XOR : only 1bit is `1`
- When ECC area has an error, call it ECC error

## ◆ Uncorrectable Error

- The result of XOR : random data
- When the flash memory has more than 2 bits error, data couldn't be corrected

# ECC Algorithm Example

Original Data : 01 02 03 04 05 06 07 08 09 . . . .

ECC code : 01000011 001

Changed Data : 01 02 03 04 05 07 07 08 09 . . . .

ECC code : 01000110 001

ECC code : 00000101 000 (5h 0h)

⇒ Fail bit is 6th byte 1st bit

	P1024	P1024`	P512	P512`	P256	P256`	P128	P128`	P64	P64`	P32	P32`	P16	P16`	P8	P8`	P4	P4`	P2	P2`	P1	P1`
Original ECC	0	0	1	1	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1
New ECC	0	1	1	0	0	1	0	1	0	1	1	0	1	0	0	1	0	1	0	1	1	0
Result of XOR	0	1	0	1	0	1	0	1	0	1	1	0	0	1	1	0	0	1	0	1	0	1

- The result data of XOR means **correctable error**

- Because we know fail byte & bit address, can make it with correct data.