

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Special Issue 9, May 2016

Design of Flash Controller for Single Level Cell NAND Flash Memory

Ashwin Bijoor¹, Sudharshana²

P.G Student, Department of Electronics and Communication, NMAMIT, Nitte, Karnataka, India¹

Assistant Professor, Department of Electronics and Communication, NMAMIT, Nitte, Karnataka, India²

ABSTRACT: NAND flash is used in many modern memory applications. The paper presents a novel approach for design of NAND flash controller. An 8 bit error correcting Reed Solomon (RS) is designed for detecting and correcting errors of flash memory. RS encoder is designed using cyclic encoder design. RS decoder used in controller design is complex. Decoder design uses Berlekamp Massey, Chien Search and Foney algorithms for detecting and correcting errors. Both RS encoder and decoder designs are efficient and utilizes Field Programmable Gate Array (FPGA) hardware effectively. Controller design also proposes an effective method for generating control signals for NAND flash memory.

KEYWORDS: Berlekamp Massey, Chien Search and Foney algorithms

I. INTRODUCTION

Flash memory is a non-volatile memory that is used in applications where high memory designs are required. NAND flash memory must be accessed sequentially. Therefore a controller is designed for sequential access of data. The block diagram of NAND flash controller is shown Figure 1.



Figure 1: NAND Flash Controller

NAND flash controller block controls flow of data from memory to host or from host to memory using main module. Module also generates control signals and set of commands required for exchange of data. Timing module monitors time delay to data and control signals. It ensures that data and control signals are applied to NAND flash when it is in ready state. Timing module produces required amount of delay to corresponding control signals so that required operation is successfully performed. Error Correction code (ECC) generator and detector module generates Cyclic Redundancy Codes (CRC) for data to be written into flash, detects and corrects error in data block that is read from flash memory.

II. LITERATURE SURVEY

Most of NAND flash application use hamming code for detecting correcting errors. But hamming code can only correct single error in a block of data. In case of applications where each bit of data is very critical, this could lead to failure of the device. So RS encoder design is used to provide a reliable and efficient way of producing correction bits. The



(An ISO 3297: 2007 Certified Organization)

Vol. 5, Special Issue 9, May 2016

correction bits produced at output provides capability of correcting multiple errors. The encoder design uses cyclic encoder design to produce correction bits [1], [7]. These correction bits are produced using two arithmetic operations that is galios multiplication and galios addition [2]. The RS decoder design proposed detects and corrects error. The decoder design uses various blocks. The block used in decoder design are syndrome computation, Berlekamp Massey algorithm, Chien Search algorithm [5], foney algorithm [6] and error correction. The controller designed uses an asynchronous interface for NAND flash to speed up communication.

III.RS ENCODER

RS encoder is represented as RS(m, k) with m bit symbols and t symbol error correction capability [1]. RS encoder output is m symbol block for k symbol data. Figure 3 shows RS encoder block diagram. A function of the transmitted message m(x), the generator polynomial g(x) and the number of parity symbols 2t is defined and systematically encoded as transmitted code word as shown in (1) [7], [1].



Figure 3: RS Encoder Block Diagram

A. Galois Multiplier

The Galois multiplication of two 8 bit data is done using (2) to (19).

$$\beta = \sum_{\substack{i=0\\i\neq j}}^{r} a_i \cdot \alpha^i \tag{2}$$

$$\gamma = \sum_{i=0}^{n} b_i \cdot \alpha^i \tag{3}$$

$$\beta \bullet \gamma = \sum_{i=1}^{14} p_i \bullet \alpha^i \tag{4}$$

Intermediate product p can be obtained multiplying β and γ and equating equal power terms.

$p_0 = a_0 \bullet b_0$	(5)
$\mathbf{p}_1 = \mathbf{a}_0 \bullet \mathbf{b}_1 + \mathbf{a}_1 \bullet \mathbf{b}_0$	(6)
$p_2 = a_0 \bullet b_2 + a_1 \bullet b_1 + a_2 \bullet b_0$	(7)
$p_3 = a_0 \bullet b_3 + a_1 \bullet b_2 + a_2 \bullet b_1 + a_3 \bullet b_0$	(8)
$p_4 = a_0 \bullet b_4 + a_1 \bullet b_3 + a_2 \bullet b_2 + a_3 \bullet b_1 + a_4 \bullet b_0$	(9)
$p_5 = a_0 \bullet b_5 + a_1 \bullet b_4 + a_2 \bullet b_3 + a_3 \bullet b_2 + a_4 \bullet b_1 + a_5 \bullet b_0$	(10)
$p_6 = a_0 \bullet b_6 + a_1 \bullet b_5 + a_2 \bullet b_4 + a_3 \bullet b_3 + a_4 \bullet b_2 + a_5 \bullet b_1 + a_6 \bullet b_0$	(11)
$p_7 = a_0 \bullet b_7 + a_1 \bullet b_6 + a_2 \bullet b_5 + a_3 \bullet b_4 + a_4 \bullet b_3 + a_5 \bullet b_2 + a_6 \bullet b_1 + a_7 \bullet b_0$	(12)
$p_8 = a_1 \bullet b_7 + a_2 \bullet b_6 + a_3 \bullet b_5 + a_4 \bullet b_4 + a_5 \bullet b_3 + a_6 \bullet b_2 + a_7 \bullet b_1$	(13)
$p_9 = a_2 \cdot b_7 + a_3 \cdot b_6 + a_4 \cdot b_5 + a_5 \cdot b_4 + a_6 \cdot b_3 + a_7 \cdot b_2$	(14)



(An ISO 3297: 2007 Certified Organization)

Vol. 5, Special Issue 9, May 2016

$p_{10} = a_3 \cdot b_7 + a_4 \cdot b_6 + a_5 \cdot b_5 + a_6 \cdot b_4 + a_7 \cdot b_3$	(15)
$p_{11} = a_4 \bullet b_7 + a_5 \bullet b_6 + a_6 \bullet b_5 + a_7 \bullet b_4$	(16)
$p_{12} = a_5 \bullet b_7 + a_6 \bullet b_6 + a_7 \bullet b_5$	(17)
$p_{13} = a_6 \bullet b_7 + a_7 \bullet b_6$	(18)
$p_{14} = a_7 \bullet b_7$	(19)
Output bit p_0 to p_{14} is intermediate product bits.	These bits are used to calculate Galois product.
$d_0 = p_0 + p_8 + p_{12} + p_{13}$	(20)
$d_1 = p_1 + p_8 + p_9 + p_{12} + p_{14}$	(21)
$d_2 = p_2 + p_9 + p_{10} + p_{13}$	(22)
$d_3 = p_3 + p_8 + p_{10} + p_{11} + p_{12} + p_{13} + p_{14}$	(23)
$d_4 = p_4 + p_8 + p_9 + p_{11} + p_{14}$	(24)
$d_5 = p_5 + p_9 + p_{10} + p_{12}$	(25)
$d_6 = p_6 + p_{10} + p_{11} + p_{13}$	(26)
$d_7 = p_7 + p_{11} + p_{12} + p_{14}$	(27)

The outputs d₀ to d₇ represents Galois product output for 8 bit Galois input [2].

Β. **Generator Polynomial**

The generator polynomial is generated using primitive polynomial and primitive element α . p(x) = x8+x4+x3+x2+1 is used as primitive polynomial. The value of primitive element used is 2 (i.e. $\alpha = 2$).

The generator polynomial is derived by applying Galois multiplication and Galois addition for a set of variables. Generation of generator polynomial for RS coding with 8 bit errors is shown in (28) and (29) [1].

$$g(x) = \prod_{i=0}^{15} (x - \alpha^{i})$$
(28)
$$g(x) = x^{16} + 110 x^{15} + 252 x^{14} + x^{13} + 196 x^{12} + 127 x^{11} + 98 x^{10} + 241 x^{9} + 231 x^{8} + 48 x^{7} + 61 x^{6} + 126 x^{5} + 60 x^{4} + 210 x^{3} + 193 x^{2} + 142x + 29$$
(29)

IV.RS DECODER

Decoding process is difficult to understand and to implement in hardware than encoding process [3]. Figure 4 shows RS decoder implementation diagram.



Figure 4: RS Decoder Implementation Diagram [4]

А. **Syndrome Computation**

At the receiver, syndrome is calculated to check presence of errors. The syndrome polynomial contains point and proportion of upto t errors in an invalid code word [4]. The computation of syndrome coefficients is given by (30).

$$S_i = R(x)|_{x=\alpha^i} = R(\alpha^i)$$
(30)



(An ISO 3297: 2007 Certified Organization)

Vol. 5, Special Issue 9, May 2016

Thus syndrome polynomial is represented as

$$S = \sum_{i=0}^{i=x^{2t-1}} S_i \cdot x^{(i-1)}$$
(31)

where R(x) is received symbols and it is represented as

$$R(x) = \sum_{n=0}^{n=2t-1} R_{n-1} \cdot x^{(n-1)}$$
(32)

B. Berlekamp Massey Algorithm

Key equation solver is the main component of RS decoder. This equation involves 2t linearly dependent equations. It generates the key equations [5].

 $\Phi(x) \bullet S(x) = \Omega(x) \bullet mod(x2t)$ (34) where $\Phi(x)$ is error locator polynomial and $\Omega(x)$ is error evaluator polynomial [6].

C. Chien Search Algorithm

Chien search performs function of finding error locations, when the Chien sum is zero [6]. Chien search utilizes all possible input variables and then checks whether outputs are zero. The summation of odd variables are computed at one end and the summation of even variables is computed at other end [5]. Then two summation outputs are added. If summation variable is zero at any clock cycle, then error point is determined from position of clock cycle [6]. The magnitude of error value is calculated using Forney Algorithm. The coefficients of $\Omega(x)$ is input for algorithm [6].

D. Error correction

Once error points and proportions are computed, error correction block accepts received code word. At particular error location, XOR operation is performed with received code word and error magnitude at the location to obtain corrected massage symbols. To interlock order of bytes in both variables, a FIFO register is utilized either at received code word or at error variable because error variables are produced in reverse order of received code word [6].

V. NAND FLASH CONTROLLER

Figure 3.1 shows block level representation of NAND Flash Controller. The control signal output is generated using a 50MHz clock at input of Controller. On application of reset pulse at the input, all variables are initialized to default value. After a reset pulse, an activate signal is made logic high. At the occurrence of activate signal, controller gets activated and waits for a command input that specifies controller about operation to be performed on flash. Depending on operation, control signals are generated and are applied to flash. The controller also has two 8 bit data I/O that interfaces both flash as well as PC.



Figure 5: NAND Flash Controller



(An ISO 3297: 2007 Certified Organization)

Vol. 5, Special Issue 9, May 2016

VI.RESULTS AND DISCUSSION

Figure 6 shows an example control signal generation for reset operation. Initially nreset goes logic low state to initialize all variables. Once nreset goes logic high, an activate signal is given for a clock pulse. On applying activate pulse, command FFh is latched onto data bus for one clock cycle.



Figure 6: Flash Control Signal Generation

Figure 7 shows RS encoder simulation results. During first two clock cycles, all variables are initialized and a counter k is initialized to 0. After third pulse, input is applied to encoder through datain port and dataout port is switched to input. Once counter reaches a value of 188, the dataout is switched to output of encoder.

Name	Value	0 ns	50 ns	1.1	100 ns	I	150 ns	a s. P	200 /16	P	50 ns	, p	00 ns	. 3	50.05	a n P	100 ns	5 A.	150 ns	1.1	500 ns			13,600 m		3,650 m	7.7	p, 700 m		13,750 m		3,800 m		p,850 m		13,900 ne		3,950 n	2.21	4,000 ms	. 19	250 ne
▶ 🌌 op[7:0]	x	X 159 43	112		193 (557	(83)	7000	119 (2	1 37		103 (13	20 205	23	- 00 X	256 (2	++)(2)	4 (210	135	61		22 143	X	168	38 10) 86	1 99 X	196)	25 (23	0 (109	*						25	3					
 M crc(7:0) 	x										×															x				X	92 2	3 17	175	(42 X	32 1	94 140	(125	222	169 22	1 (42)	33	137 61
 ip(7)0) 	109	(43)(11	2 131	(158)(1	167 (83	X 70 X	119 (211 39	(86)	203	120 20	35 (235	(80)	236	244)(2	34 (25	0 135	61	15	222)	18 90 11	X	238	00 00	X 99	X 146 X	25 (2	30 20	9 203													
1 clk	0																					1.1																				
12 1	x							_				_				_																										
 b[31:0] 	20000000000000								300			000000	2000			_															3000000			kannaar	ĸ							
 M N31/0 	0	(0X 1 X 2	13	(1)	5 X 6	X 7 X	8	9 X 20	X 11	12 1	13 X 1	4 X 15	2 26	17 1	18 . ;	0 X 20	X 21	X 22	23)	24)	3 X 26 X	X	179	50 X 15	102	X 353 X	104 1	85 13	6 1 187	1 105 X	359 X 1	0 19	192	193 X	194 X 1	15 X 196	197	190	199 23	0 (201)	202	203 204
V M (150.70)	(X. X. X. X. X. X.	(1238.)	18. 2 1122	5. (142	11122. 18	3.2 123	7. 1203	X[81	168. 15	1. 1125	(173.)	1235. 1	154. 012	2. 140		1170.0	1205.	217. 19	5. 110	1. (5.2	(Iss.,)/15	1753.	17188	TISE Y	165. 172	00 1000	VIS0.	1146	TELL VIS	1. YN2.	1213	1171.01	175. 174	L YES.	17104	1140.01	128. 1100		(1221)	142. (151	TER.	X161
III 115.7:00	x	X (0) (238)	O X E	25 (49	1 159	6 (23	7 203	X 01 1	258 X 3	123	0(173)	228	134 1 1	2 1 40	X 59	170 1	205	217 1	95 11	1 5	66 1 169	53	158	159	165 Y 2	22 28	X 50	1 10	131	6 Y 92	× 213	171	125 1 4	6 X 39	104	142	128 11	A VI	221	42 8	1 117	X 61 X 0
In 114,7.00	x	X 0 42	1 11	9 50	248	251 1 15	7 127	X 87 1	56 1	200	0.211	47	148 2	111	249	21	62	72 1 1	19 2	0 241	48 189	1 344	¥ 215	200	125 1	186 21	327	137	98 V 1	21 / 217	171	175	42 7 3	A 1 32	1 1 1 1 1 1	128	109 X 10	2 7	1 42	AS 12	7 1 10	X 0
M 11370	x	X OX 9 Y	212 X 11	18 X 41	241	187 2 2	26	X 21	113 X 2	H 225	000	193	49 X 2	2 1 195	148	147	18 V	24 X 2	S1 X 11	1 1 80	130 132	1	1.130	A2 1	M7 X 1	120 111	V 180	1 51	74	0 V 171	175	and	72 V 1	A4 14	128	102	M9 V 7		1 53	177 6		
► M 0.2.7.00	x	X (0) 215	175 2	32 66	208	101 29	2 × 178	X 244	74 X 1	51 244	232	23	111 7	X 210	75	356 X	21	37 (20 2	2 247	350 38	20	114	251	76 1 1	179 90	219	150	215	0 17	40	20	104	12 12	109	169	221 4		137	61		-
11,7:0)	x	X (0) 84)	183 / 7	0 X 51	X 157 X	100 22	1 79	X 109 X	1 (2	53 (135	(7)	69	155 (25	1 1 175) 135	215 1	107	141 (1	97 1.1	3 / 10	35 22	27	169	1 52 1	44 1 1	198 12-	× 168	183	251 1 1	1 1 40	X 39	224	140 X 1	28 × 22	169	1 221 X	42 5		7 (61)		0	
10,7:0]	x	X (0) 171 (20 1 1	43 2	1 124)	243 (29	1 60	X 91 X	201 2	11 (12	(131)	49	175 3	9 243	(123	46 1	57)	248 (1	43 (4	1 227	204 (47	71	123	43 X	296 . 1	61 157	205	8 82	235 2	25 39	104	140	128 1	57 205	221	1 42 X	53 X 1	37 1 0	X		0	
 M 9,761 	×	X (0) 75 X	24 X 2	15 X 40	X 214 X	49 X 60	226	X 74 X	244 X 1	3 28	(192)	85	22 X 1	9 X 85	X 95	139 1	28 X	114 X I	12 X 3	× 91	92 169	225	196	203	138 1	95 133	X 95	251	183	D 104	140	125	109 1	\$2 × 22	42	53 X	137 6			0		
N 8.7.01	x	X (0) 204 X	154 X Z	55 4	1 81 1	22 1	5 X 238	X 135	85 X 5	5 240	96	136	192 2	140	X 57	234	145	105 2	32 21	5 113	74 214	81	247	1 1 50	87 1	152 158	X a	10	121	0 14	128	259	169 X 2	21 42	Y 53	117 1	61			0		
III 17.7:01	x	X (0) (231)	225 1 9	3 225	1 17 1	96 X 25	6 1 29	142 1	67 Y 1	233	247	45	117 9	172	2 93	156	193	162	15 3	4 114	130 157	129	87	131	165 V 2	227 6	× 42	178	58 (2	1 128	109	364 V	221 X 4	5 V 51	137	61 V	_			0		
M 167.01	x	X (0) (192)	228 1	12 15	285	170 21	217	1 57	26 1	6 4	22	250	199 1	109	23	222	96	217	8 21	0 78	122 202	213	¥ 134	102	383 / 1	158 30	1 31	198	-	105	102	221	42 1 1	5 V 18	1 61	<u></u>						
M 1570	x	X (0) (161)	211 1 5	3 X 44	157 1	31 20	2 Y 241	X 117	128 X 1	145	18	31	208 1 1	5 110	XA	212	7.4	94 V 1	9 Y 9	1 1 39	106 / 115	1 55	¥ 57	102	M7 X 3	20 11	V 20	1 25	141 / 1	17 V 160	¥ 221	42	53 V I	17 1 61	V.	1			0			
► M H.7:0	x	X 0 232	75 1 1	12 / 71	03	45 23	9 128	177	17 1	7 179	201	185	38 2	3 X 143	1 85	37	60 X	150		4 132	15 112	236	191	239	222	St 163	XS	1	156 1	59 221	42	197	137 6						0		_	
III 0.7:01	x	X (Q) (190)	174 1 5	0 X 147	220	149 20	1 02	X 120 Y	254 X I	2 201		24	41 1	150	2.30	200	175	225 (2	86 5	\$ 245	236 35	23	¥ 68	111	32 V 1	112 183	158	18	215 (2	29 1 42	¥ 53	187	61	_								
12.7.61	x	X (0) (249)	121 7 2	5 (85	162	84 2	2 248	230	188 1 8	6 214	129	116	195 2	3 21	40	199	209	242 1	2 9	20	214 123	200	¥ 213	115	74 X 1	170 21	¥ 157	98	45. (2	20 / 53	117	61		_				0				
B BM 0 7.01	· ·	X (0) 229 V	219 X 4	5 V 33	67 1	241 11	0 187	X 11	243	1 1 12	(145)	155	158 X 1	2 X 151	71	20	135	15 X 1	10 X 11	9 1 130	165 X 211	21	V #2	102	60 V 3	2 2 1 12	V 225	<u> </u>	195 7	26 V 137	V AL			-			0	· ·				
N 10 7.00	÷	X 0 57	167 V 10	11 1 130	1 131	241 / 12	5 X 45	V 90	18 X 1	20 125	245	62	34 1 6	× 243	7 717	114	30	W.V.	1	1	201 / 279	200	V 64	20	440 V 4	144 21	V 140	1.00		A 1 61	~	<u> </u>		_								
P P P P P P P P P P P P P P P P P P P	-		10.00		1			<u></u>						-					1-1-1		1	- 100 A	~~~	1	me	dive.	140	1.00	car	grad	~			-							_	
		X1: 0.000 ns																																								

Figure 7: RS Encoder Results

Figure 8 shows RS decoder simulation. During first few clock cycles, inputs are accepted, syndrome values are calculated and stored into a buffer. The syndrome values calculated are applied to other blocks in consecutive clock cycles to deduce error position and magnitude of error. In simulation, Address_error shows that there is an error at location number 131 and correction_value shows magnitude of error as 180.

Name	Value		400 ns	600 ns	800 ns	1,000 ns	1,200	ns	Å	118,200 ns	118,400 ns		165,400 ns	165,600 ns	165,800 ns	166,000 ns	166,200 ns	1	166,400 ns
Out_byte[7:0]	0				0							88	240	X	37	X 5-	4		29
Un clk	1						Л					JU					IIII	J	
Ve reset	0						- 3		_										
Input_byte[7:0]	187	0 X	167	X 185	X	26		187							X				
Itrue_out[7:0]	х				x							88	186	X	37	X	54		29
Address_error_	0				0				\square	131					15			1 8	
Image: Section_value	0				0					0	(180)				177				

Figure 8: RS Decoder Results



(An ISO 3297: 2007 Certified Organization)

Vol. 5, Special Issue 9, May 2016

Table 1 shows synthesis report of different blocks in NAND flash controller.

Table 1: Synthesis Report of NAND flash controller

Functional	Number	Number	Number	Number of	Minimum	Power	Maximum	Memory
DIOCKS	or Slices	Flops	OILUIS	IUDS	Period (ns)	(mvv)	(MHz)	Usage
RS Encoder	27	0	52	24	10.244	60	97.618	240960 kB
RS Decoder	3076	2826	5675	21	12.557	60	79.637	406336 kB
Control Signal Generation	2446	2526	2788	52	11.179	60	89.456	336192 kB

VII. CONCLUSION

A novel design of NAND flash controller was presented to control operations on flash memory. An 8 bit error correcting RS encoder and decoder was designed and simulated using Verilog on Xilinx ise 13.2. The analysis of result shows that encoder produces CRC bits within 4.2 μ s, decoder detects and corrects error within 150 μ s. Another important part of flash controller is control signal generation. The synthesis report shows that FPGA hardware is efficiently utilized and flash controller can be effectively implemented.

REFERENCES

[1] Azad, Aqib Al, and Md Imam Shahed. "A Compact And Fast FPGA Based Implementation Of Encoding And Decoding Algorithm Using Reed Solomon Codes". IJFCC, pp. 31-35, 2014.

[2] Lin, Shu, and Daniel J Costello. Error Control Coding. Englewood Cliffs, N.J.: Prentice-Hall, 1983.

[3] Tiwari, Bhawna, and Rajesh Mehra. "Design And Implementation Of Reed Solomon Decoder For 802.16 Network Using FPGA", IEEE International Conference on Signal Processing, Computing and Control, 2012.

[4] Dayal, Priyanka, and Rajeev Kumar Patial. "FPGA Implementation Of Reed-Solomon Encoder And Decoder For Wireless Network 802. 16". International Journal of Computer Applications, pp. 42-45, 2013.

[5] Sarwate, D.V., and N.R. Shanbhag. "High-Speed Architectures For Reed-Solomon Decoders". IEEE Transactions on Very Large Scale Integration (VLSI) Systems, pp. 641-655, 2001.

[6] Das, Anindya Sundar, Satyajit Das, and Jaydeb Bhaumik. "Design Of RS (255, 251) Encoder And Decoder In FPGA". International Journal of Soft Computing and Engineering, vol. 2, pp. 391-394, 2013.

[7] Ziliang Lin, Changyin Liu, and Lei Liu. "An Efficient RS Encoder For CCSDS". 2013 IEEE 4th International Conference on Software Engineering and Service Science, pp.791-794, 2013.