Chapter 5
Asymmetric Solid Burst Correcting Integer Codes

The contents of this chapter are based on the paper mentioned below:

- Das, P. K. and Pokhrel, N. K. Asymmetric solid burst correcting integer codes. *Submitted for publication.*

Chapter 5

Asymmetric Solid Burst Correcting Integer Codes

5.1 Overview

With the development of technology, the storage channels are getting smaller day by day, so a noise factor may strike two consecutive cells leading to a multiple bit upset (MBU) [76]. Some examples in this regard are observed in the case of static random-access memory (SRAM) [8], dynamic random access-memory (DRAM) [52]. So, it becomes necessary to construct an error-correcting mechanism to deal with these errors occurring between two adjacent bytes.

We have already discussed the occurrence of asymmetric patterns and solid bursts in Chapters 2 and 4 respectively. To deal with the asymmetric solid bursts, in this chapter, we have developed a class of integer codes that can correct asymmetric solid bursts occurring within a b-bit byte as well as between two adjoining b-bit bytes. By doing so, the code becomes capable of correcting asymmetric solid bursts occurring anywhere in the codeword and the codes can be implemented without interleaving. We name these codes as integer $(A_lSB)_b$ codes. In Section 5.2, we present construction, encoding and decoding of $(A_lSB)_b$ codes along with an example. Further, the probability of erroneous decoding and conditions for undetected errors are derived in Section 5.3. It is followed by the comparison of this class with other similar existing classes in different aspects.

5.2 Construction of codes

For low-density asymmetric CT-bursts with $l = w$ (burst length = weight) introduced in Chapter 3, where bursts are considered anywhere within a b-bit byte, we have $e_{b,1} = \{2^{i-1} | 1 \leq i \leq b\}$ as the set of all asymmetric solid bursts of length 1. Similarly $e_{b,2} = \{2^{i-1} + 2^i | 1 \leq i \leq b-1\}, e_{b,3} = \{2^{i-1} + 2^i + 2^{i+1} | 1 \leq i \leq b-2\}, \ldots, e_{b,l} = \{2^{i-1} + 2^i + 2^{i+1} + \ldots + 2^{i+l-2} | 1 \leq i \leq b-l+1\}$ are the collection of all asymmetric solid bursts of length $2, 3, \ldots, l$, in general, $e_{b,t} = 2^i(2^t - 1)$ for $0 \leq i \leq b-t$. Thus by defining $E_{b,l} = \bigcup\limits_{t=1}^{l} e_{b,t}$, we get the desired collection of all asymmetric solid bursts up to length l occurring within a b-bit byte.

5.2.1 Encoding procedure

The encoding procedure is similar to the preceding chapters for errors occurring within a b-bit byte. But, for the errors occurring across two adjoining b-bit bytes, the procedure of finding coefficients changes. We now introduce the set of syndromes to be used in the error-correcting procedure for the asymmetric solid bursts.

Definition 5.1. *The set of all syndromes in integer $(A_lSB)_b$ codes where errors occur within a b-bit byte will be*

$$S_1 = \bigcup_{i=1}^{k+1} \Big[-C_i E_{b,l} \Big] \pmod{2^b - 1}, \tag{5.1}$$

and for errors occurring between two adjoining b-bit bytes, the set will be

$$S_2 = \bigcup_{i=1}^{k} \Big[C_i U_r + C_{i+1} V_s \Big] \pmod{2^b - 1}, \tag{5.2}$$

where $U_r = \{-2^{b-1} - 2^{b-2} - \ldots - 2^{b-r}\}$, $V_s = \{-2^0 - 2^1 - \ldots - 2^{s-1}\}$ such that $1 \leq r, s \leq l-1$, $max\{r+s\} = l$ and the coefficient $C_1, C_2, \ldots, C_k$ are chosen from $\mathbb{Z}_{2^b-1} \setminus \{0, 1\}$ such that the sets $-C_1E_{b,l}, -C_2E_{b,l}, \ldots, -C_kE_{b,l}, E_{b,l}$ and $C_iU_r + C_{i+1}V_s$ are all mutually disjoint.

The process of choosing coefficients can be done with the help of a computer (python code provided in Appendix E). Upon the availability of coefficients for the construction, we prepare look up table, LUT_2 consisting of all the syndromes, error locations and the corresponding errors within the prescribed limit and LUT_1 contains the coefficient C_i's.

Lemma 5.2. *The collection of asymmetric solid bursts occurring within a b-bit byte in $\mathbb{Z}_{2^b-1}$ is distinct.*

Proof. We have $E_{b,l} = e_{b,1} \cup e_{b,2} \cup \ldots \cup e_{b,l}$ with $e_{b,t} = \{2^i(2^t - 1) \,|\, 0 \leq i \leq b - t$ and $1 \leq t \leq l \leq b - 1\}$.

- **Case I:** If possible, let $2^i(2^t - 1) = 2^j(2^t - 1)$, where $0 \leq i < j \leq b - t$ and $1 \leq t \leq b-1$. Then $2^i = 2^j$, which is possible only if $i = j$ in the given interval leading us to the distinctness within the collection $e_{b,t}$.

- **Case II:** Suppose e_{b,t_1} & e_{b,t_2} have a common entry, where $1 \leq t_1, t_2 \leq b - 1$. Without the loss of generality, suppose $t_1 < t_2$, then $e_{b,t_1} = 2^0(2^{t_1} - 1), 2^1(2^{t_1}-1), \ldots, 2^{b-t_2}(2^{t_1}-1), \ldots, 2^{b-t_1}(2^{t_1}-1)$ and $e_{b,t_2} = 2^0(2^{t_2}-1), 2^1(2^{t_2} - 1), \ldots, 2^{b-t_2}(2^{t_2} - 1)$. If possible, let us suppose that $2^i(2^{t_1} - 1) = 2^j(2^{t_2} - 1)$, where $0 \leq i \leq b - t_1$, $0 \leq j \leq b - t_2$.

  Subcase I: $i = j$, then $2^{t_1} - 1 = 2^{t_2} - 1$, which is a contradiction.

  Subcase II: $i < j$, then $(2^{t_1} - 1) = 2^{j-i}(2^{t_2} - 1) \implies 1 + 2 + \ldots + 2^{t_1-1} = 2^{j-i}(2^{t_2} - 1)$, which is again a contradiction as both L.H.S. and R.H.S. lie within 1 and $2^b - 1$ but L.H.S. is odd and R.H.S. is even.

  Subcase III: $j < i$, similar to *Subcase II* we get contradiction here also. Hence we have distinctness within the collection $E_{b,l}$.

□

Remark 5.3. *The approach discussed above may not be applicable in the case of symmetric errors as both $\pm$ symbols are involved simultaneously and this method of odd/even may not work in the ring $\mathbb{Z}_{2^b-1}$. For instance in $\mathbb{Z}_{255}$, $-2^0(2^3 - 1) = 2^3(2^5 - 1) = 248$.*

Note: For determining distinct syndromes pertaining to the asymmetric solid bursts spread across two adjoining bytes, selection of coefficients play an important role. Based on the discussions done so far for the construction of error and syndrome sets along with Lemma 5.2, theorems below illustrate the number of elements/conditions required for the construction.

Theorem 5.4. *Integer $(A_lSB)_b$ code can correct asymmetric solid bursts up to length l occurring within a b-bit byte as well as spread across two adjacent b-bit bytes if there exist coefficients $C_1, C_2, \ldots, C_k$ in $\mathbb{Z}_{2^b-1}$ such that*

1. $|S_1| = (k+1)[\frac{l}{2}(2b-l+1)]$.
2. $|S_2| = k\frac{l(l-1)}{2}$.
3. $S_1 \cap S_2 = \phi$.

Proof.

1. It is sufficient to show that $|E_{b,l}| = \frac{l}{2}(2b-l+1)$. Clearly $e_{b,1} = 2^i$, $0 \leq i \leq b-1$ has b patterns, similarly $e_{b,2} = 2^i(2^2-1)$, $0 \leq i \leq b-2$ has $b-1$ patterns, by continuing this pattern for any $l \leq b-1$, $e_{b,l}$ has $b-l+1$ patterns. Thus $|E_{b,l}| = b + b - 1 + \ldots + b - (l-1) = bl - \frac{l(l-1)}{2} = \frac{l}{2}(2b-l+1)$.
2. For $l = 2$, $C_iU_1 + C_{i+1}V_1$ gives the number of asymmetric solid bursts falling under this category, which is 1. Similarly for $l = 3$, $C_iU_1 + C_{i+1}V_2$ and $C_iU_2 + C_{i+1}V_1$ are the 2 choices, continuing this, for any random $l \leq b-1$, we have $l-1$ different choices, viz., $C_iU_1 + C_{i+1}V_{l-1}, \ldots, C_iU_{l-1} + C_{i+1}V_1$. Since a codeword has $(k+1)$ b-bit bytes, so asymmetric solid bursts spread across two adjoining bytes have k possibilities. Thus the number of asymmetric solid bursts up to length l in this category will be $k\frac{l(l-1)}{2}$.
3. This condition is obvious as the criteria for constructing the code depends on this specificity.

□

Theorem 5.5. *By defining $\varepsilon_{b,l} = S_1 \cup S_2$, cardinality of the set of syndromes for an integer $(A_lSB)_b$ code will be $kbl + \frac{l}{2}(2b-l+1)$.*

Proof. From Theorem 5.4, we have $|\varepsilon_{b,l}| = k\frac{l}{2}(2b-l+1) + \frac{l}{2}(2b-l+1) + k\frac{l}{2}(l-1) = kbl + \frac{l}{2}(2b-l+1)$. □

Figure 5.1: **Asymmetric solid burst within a *b*-bit byte**

Syndrome element (S_1)	Error location (i)	Corresponding error e
$\longleftarrow b$ bits $\longrightarrow$	$\longleftarrow \lceil log_2(k+1)\rceil$ bits $\longrightarrow$	$\longleftarrow b$ bits $\longrightarrow$

Figure 5.2: **Asymmetric solid burst spread across two adjoining *b*-bit bytes**

Syndrome element (S_2)	Error location (i)	Error location ($i+1$)	Error vector e and e'
$\longleftarrow b$ bits $\longrightarrow$	$\longleftarrow \lceil log_2 k\rceil$ bits $\longrightarrow$	$\longleftarrow \lceil log_2 k\rceil$ bits $\longrightarrow$	$\longleftarrow 2(l-1)$ bits $\longrightarrow$

This class of code is also written as $((k+1)b, kb)$ integer $(A_lSB)_b$ code.

Corollary 5.6. *For any $((k+1)b, kb)$ integer $(A_lSB)_b$ code, $k \leq \frac{2^{b+1}-4-2bl+l^2-l}{2bl}$.*

Proof. Since all syndromes non-zero elements from the ring $\mathbb{Z}_{2^b-1}$, thus from Theorem 5.5, we have $kbl + bl - \frac{l^2}{2} + \frac{l}{2} \leq 2^b - 2 \implies k \leq \frac{2^{b+1}-4-2bl+l^2-l}{2bl}$. $\square$

5.2.2 Decoding procedure

For decoding LUT_2 is constructed with the help of (5.1) and (5.2), here the table consists of Sl. No., syndrome value, error location(s) and the corresponding error(s). Diagrammatic representation of a syndrome table entry in terms of bits is given in Figure 5.1-5.2.

After receiving a message $\bar{B}_1\bar{B}_2\ldots\bar{B}_k\bar{C}_B$, decoder obtains the syndrome value as $S = [C_{\bar{B}} - \bar{C}_B] \pmod{2^b-1}$, then tries to match this value with the available syndrome values in column 2 of LUT_2. Since the elements in column 2 of the look up table can be arranged in ascending order, so η_{TL} binary searches required for this matching is given by $1 \leq \eta_{TL} \leq \lfloor log_2|\varepsilon_{b,l}|\rfloor + 2$ (refer [63]). If no such value is available in the table, then it is assumed to be beyond the decoder's capability. However, this is only possible if the error occurred is either beyond the specified length or of some other nature. Also the size of LUT_2 will be $|S_1| \times [2b + \lceil log_2(k+1)\rceil] + |S_2| \times [b + 2\lceil log_2(k)\rceil + 2(l-1)]$ bits. Following steps are followed for decoding:

- For asymmetric solid bursts up to length l within an information byte i, $1 \leq$

$i \leq k$,

$$B_i = [\bar{B}_i + e] \pmod{2^b - 1},\ e \in E_{b,l},$$

where syndrome $S = -C_i \times e \pmod{2^b - 1}$.

- For asymmetric solid burst up to length l within the check byte, C_B,

$$C_B = [\bar{C}_B + e] \pmod{2^b - 1},\ e \in E_{b,l},$$

where syndrome $S = e$.

- For asymmetric solid burst of length up to l occurring between i^{th} and $(i+1)^{th}$ data byte ($1 \leq i \leq k-1$):

$$B_i = [\bar{B}_i + e] \pmod{2^b - 1},\ e \in E_{b,l},\ -e \in U_r;$$

$$B_{i+1} = [\bar{B}_{i+1} + e'] \pmod{2^b - 1},\ e' \in E_{b,l},\ -e' \in V_s;$$

where syndrome $S = [C_i(-e) + C_{i+1}(-e')] \pmod{2^b - 1}$.

- For asymmetric burst of length up to l occurring between the last data byte (k^{th} byte) and the check byte:

$$B_k = [\bar{B}_k + e] \pmod{2^b - 1},\ e \in E_{b,l},\ -e \in U_r;$$

$$C_B = [\bar{C}_B + e'] \pmod{2^b - 1},\ e' \in E_{b,l},\ -e' \in V_s;$$

where syndrome $S = [C_k(-e) + e'] \pmod{2^b - 1}$.

Example 5.7. *Let $b = 9$, $l = 3$, then $C_1 = 11, C_2 = 19, C_3 = 45$ and $C_4 = -1$ can be considered for the transmission, syndrome entries generated using (5.1) and (5.2) is given in Table 5.1. The message* 111000011 100100011 010111011 *is encoded as* 111000011 100100011 010111011 111001111*, we may have the following error possibilities:*

Case I*(Asymmetric solid burst within an information byte):*
Suppose the message is received as 111000011 100100011 010$\underline{000}$011 111001111*, then syndrome* $S = [11 - 487] \pmod{511} = 35 = -45 \times 56$. *Thus error* $56 = 2^3+2^4+2^5$ *has occurred at* $4^{th}, 5^{th}$ *and* 6^{th} *components of* 3^{rd} *information byte. Hence the corrected information byte will be* $B_3 = [386+56] \pmod{511} = 442 = 010111011$.

Case II*(Asymmetric solid burst within the check byte):*

Suppose message $111000011\ \ 100100011\ \ 010111011\ \ \underline{00}1001111$ *is received, then syndrome* $S = [487 - 484] \pmod{511} = 3$. *Thus error* $3 = 2^1 + 2^2$ *has occurred in the check byte* C_B *at* 1^{st} *and* 2^{nd} *components. Hence the corrected check byte will be* $C_B = [484 + 3] \pmod{511} = 487 = 111001111$.

Case III*(Asymmetric solid burst occurring between two adjoining information bytes):*
Suppose message $1110000\underline{00\ \ 0}00100011\ \ 010111011\ \ 111001111$ *is received, then syndrome* $S = [332-487] \pmod{511} = 356$. *Since* $356 = [11(127)+19(510)] \pmod{511} = [11(-384)+19(-1)] \pmod{511}$, *thus error* $e = 384 = 2^7+2^8$ *has occurred at* 8^{th} *and* 9^{th} *components of* 1^{st} *information byte and error* $e' = 1 = 2^0$ *has occurred at* 1^{st} *component of* 2^{nd} *information byte. So the corrected information bytes are* $B_1 = [7+384] \pmod{511} = 391 = 111000011$ *and* $B_2 = [392 + 1] \pmod{511} = 393 = 100100011$.

Case IV*(Asymmetric solid burst occurring between last information byte and check byte):*
Suppose message $111000011\ \ 100100011\ \ 01011101\underline{0\ \ 0}11001111$ *is received, then syndrome* $S = [209 - 486] \pmod{511} = 234 = [45(255) + (-1)(510)] \pmod{511} = [45(-256) + (-1)(-1)] \pmod{511}$. *Thus error* $e = 256 = 2^8$ *has occurred at* 9^{th} *component of* 3^{rd} *information byte and error* $e' = 1 = 2^0$ *has occurred at* 1^{st} *component of the check byte. So the corrected information byte is* $B_3 = [186 + 256] \pmod{511} = 442 = 010111011$ *and check byte is* $C_B = [486+1] \pmod{511} = 487 = 111001111$.

Case V*(Error type not as per specification):*
Suppose $111000011\ \ 100\underline{000000}\ \ 010111011\ \ 111001111$ *is received, then syndrome* $S = [193 - 487] \pmod{511} = 217$. *Since syndrome value* 217 *is not available in* LUT_2, *so it is beyond the scope of the decoder.*

Table 5.1: $\boldsymbol{LUT_2}$ **for** $\mathbf{(36, 27)}$ **integer** $\boldsymbol{(A_3SB)_9}$ **code**

Sl. No.	**Syndrome** $(\boldsymbol{\varepsilon_{9,3}})$	**Error Loc.** $(\boldsymbol{i})$	**Error** $(\boldsymbol{e})$	**Error Loc.** $(\boldsymbol{i+1})$	**Error** $(\boldsymbol{e'})$
1	1	4	1	0	0
2	2	4	2	0	0
3	3	4	3	0	0
4	4	4	4	0	0
5	6	4	6	0	0
6	7	4	7	0	0
7	8	4	8	0	0
8	12	4	12	0	0
9	14	4	14	0	0
10	16	4	16	0	0
11	24	4	24	0	0
12	28	4	28	0	0
13	32	4	32	0	0
14	35	3	56	0	0
15	47	3	192	0	0
16	48	4	48	0	0
17	55	2	24	0	0
18	56	4	56	0	0
19	64	4	64	0	0
20	70	3	112	0	0
21	91	1	224	0	0
22	93	3	32	0	0
23	94	3	384	0	0
24	95	3	384	4	1
25	96	4	96	0	0
26	110	2	48	0	0

Contd...

Sl. No.	Syndrome ($\varepsilon_{9,3}$)	Error Loc. (i)	Error (e)	Error Loc. ($i+1$)	Error (e')
27	111	2	256	3	3
28	112	4	112	0	0
29	123	2	128	0	0
30	125	1	128	0	0
31	128	4	128	0	0
32	140	3	224	0	0
33	151	3	8	0	0
34	159	1	32	0	0
35	175	2	448	0	0
36	182	1	448	0	0
37	186	3	64	0	0
38	192	4	192	0	0
39	193	1	256	2	3
40	196	3	7	0	0
41	201	2	256	3	1
42	203	1	28	0	0
43	207	2	16	0	0
44	220	2	96	0	0
45	224	4	224	0	0
46	231	1	256	2	1
47	233	3	256	0	0
48	234	3	256	4	1
49	236	3	256	4	3
50	241	3	6	0	0
51	245	2	14	0	0
52	246	2	256	0	0
53	247	1	24	0	0
54	250	1	256	0	0

Contd...

Sl. No.	**Syndrome ($\varepsilon_{9,3}$)**	**Error Loc. (i)**	**Error (e)**	**Error Loc. ($i+1$)**	**Error (e')**
55	256	4	256	0	0
56	273	3	28	0	0
57	279	3	96	0	0
58	280	3	448	0	0
59	283	2	12	0	0
60	301	1	112	0	0
61	302	3	16	0	0
62	317	2	64	0	0
63	318	1	64	0	0
64	324	2	384	3	1
65	331	3	4	0	0
66	335	1	16	0	0
67	343	2	224	0	0
68	356	1	384	2	1
69	357	1	14	0	0
70	359	2	8	0	0
71	369	2	384	0	0
72	372	3	128	0	0
73	375	1	384	0	0
74	376	3	3	0	0
75	378	2	7	0	0
76	379	1	12	0	0
77	384	4	384	0	0
78	392	3	14	0	0
79	395	3	48	0	0
80	397	2	6	0	0
81	406	1	56	0	0
82	414	2	32	0	0

Contd...

Sl. No.	Syndrome ($\varepsilon_{9,3}$)	Error Loc. (i)	Error (e)	Error Loc. ($i+1$)	Error (e')
83	421	3	2	0	0
84	423	1	8	0	0
85	427	2	112	0	0
86	434	1	7	0	0
87	435	2	4	0	0
88	440	2	192	0	0
89	443	1	192	0	0
90	445	1	6	0	0
91	448	4	448	0	0
92	453	3	24	0	0
93	454	2	3	0	0
94	466	3	1	0	0
95	467	1	4	0	0
96	469	2	56	0	0
97	473	2	2	0	0
98	477	1	96	0	0
99	478	1	3	0	0
100	482	3	12	0	0
101	489	1	2	0	0
102	490	2	28	0	0
103	492	2	1	0	0
104	494	1	48	0	0
105	500	1	1	0	0

For implementation of the codes on multi-core processors, we refer to the implementation discussed in Chapter 2. The memory required for storing the look up tables is given in Table 5.3. The codes considered here are constructed with the help of coefficients listed in Table 5.2.

Table 5.2: **Possible coefficients for the construction of integer $(A_lSB)_b$ codes up to $k = 32$**

b	l	**Coefficients**
7	2	7, 13, 19
7	3	Not possible
8	2	5, 7, 9, 29, 37
8	3	2, 29
9	3	11, 19, 45
10	2	7, 13, 25, 29, 31, 35, 49, 53, 71, 73, 79, 89, 115, 125, 127, 149, 205
10	3	17, 23, 47, 71, 107, 125, 191, 205, 239, 251
10	4	29, 35, 71, 167
12	3	9, 11, 19, 29, 45, 53, 69, 81, 85, 97, 99, 121, 127, 143, 155, 199, 209, 213, 249, 281, 303, 489, 957
12	6	11
14	4	17, 47, 71, 73, 81, 107, 109, 117, 121, 131, 143, 151, 167, 179, 187, 191, 199, 207, 209, 211, 221, 223, 229, 233, 241, 247, 253, 263, 271, 281, 289, 307
14	7	51, 61, 111
16	2	5, 7, 9, 11, 23, 35, 37, 41, 43, 45, 47, 49, 53, 55, 59, 61, 63, 65, 67, 71, 73, 77, 79, 81, 83, 85, 89, 91, 95, 97, 99, 103
16	4	31, 37, 49, 61, 67, 71, 73, 79, 81, 83, 89, 97, 99, 101, 103, 107, 109, 113, 117, 121, 127, 131, 137, 143, 149, 151, 153, 157, 163, 169, 179, 181
16	7	45, 67, 97, 131, 197, 219, 223, 241, 289, 317, 347, 349, 353, 361, 401, 437, 443, 449, 451, 459, 475, 481, 509, 563, 569, 573, 595, 599, 601, 619, 625, 643
20	2	5, 7, 9, 11, 23, 35, 37, 41, 43, 45, 47, 49, 53, 55, 59, 61, 63, 65, 67, 71, 73, 77, 79, 81, 83, 85, 89, 91, 95, 97, 99, 103
25	3	5, 7, 9, 11, 23, 35, 37, 41, 43, 45, 47, 49, 53, 55, 59, 61, 63, 65, 67, 71, 73, 77, 79, 81, 83, 85, 89, 91, 95, 97, 99, 103

Contd...

b	l	**Coefficients**
32	2	5, 7, 9, 11, 23, 35, 37, 41, 43, 45, 47, 49, 53, 55, 59, 61, 63, 65, 67, 71, 73, 77, 79, 81, 83, 85, 89, 91, 95, 97, 99, 103
32	3	17, 37, 41, 47, 49, 53, 59, 61, 65, 67, 71, 73, 79, 81, 83, 85, 89, 95, 97, 99, 101, 103, 107, 109, 113, 115, 117, 121, 125, 127, 131, 139
32	10	45, 71, 83, 97, 101, 103, 109, 113, 121, 137, 139, 143, 149, 151, 157, 163, 167, 169, 179, 181, 193, 197, 199, 209, 211, 223, 229, 233, 239, 247, 257

Table 5.3: **Bits required for some integer $(A_lSB)_b$ codes**

Codes	b	l	LUT_1 **size**	LUT_2 **size**	**Number of table look ups**
(528, 512)	16	2	64B	4.98 KB	$1 \leq \eta_{TL} \leq 12$
(528, 512)	16	7	64B	17.46 KB	$1 \leq \eta_{TL} \leq 13$
(660, 640)	20	2	80B	7.53 KB	$1 \leq \eta_{TL} \leq 12$
(1056, 1024)	32	3	128B	27.41KB	$1 \leq \eta_{TL} \leq 13$
(1056, 1024)	32	10	128B	90.21 KB	$1 \leq \eta_{TL} \leq 15$

5.3 Undetected errors, erroneous decoding probability and comparison

In this section, we derive two conditions for deciding undetected errors, followed by the probability of erroneous decoding and BER for the proposed codes and a few graphs are constructed. Also, we are going to compare the proposed codes with some linear and integer codes having similar error-correcting capability.

5.3.1 Undetected asymmetric solid bursts

An error is undetected if the resulting syndrome obtained after the occurrence of that error is equal to 0, in such situation, one may wrongly conclude the message to be error free. For any asymmetric solid burst to go undetected by an integer $(A_lSB)_b$ code, the length must be longer than l. In the following two results, LCM means Least Common Multiple.

Theorem 5.8. *An asymmetric solid burst with length r $(> l)$ within a b-bit byte will go undetected by a $(kb + b, kb)$ integer $(A_lSB)_b$ code with parity check matrix $H = (C_1\, C_2\, \ldots\, C_k\, -1)$ if and only if $2^b - 1$ divides LCM $(C_i, 2^r - 1)$.*

Proof. Consider an asymmetric solid burst of length r within i^{th} byte. An asymmetric solid burst (E_r) of length r will go undetected by the integer $(A_lSB)_b$ code if and only if

$$C_iE_r = 0 \pmod{2^b - 1}. \tag{5.3}$$

The binary representation of E_r is $2^p + 2^{p+1} + \ldots + 2^{p+r-1}$ (for $0 \leq p \leq b - r$) which is equal to $2^p(2^0 + 2^1 + \ldots + 2^{r-1}) = 2^p(2^r - 1)$. So, from (5.3), we have

$$C_i2^p(2^r - 1) = 0 \pmod{2^b - 1}.$$

As 2^p and $2^b - 1$ are relatively prime, so

$$C_i(2^r - 1) = 0 \pmod{2^b - 1}.$$

This implies $2^b - 1$ divides LCM of C_i and $2^r - 1$.

□

Theorem 5.9. *An asymmetric solid bursts with length $r = s_1 + s_2$ $(> l)$ affecting s_1 and s_2 consecutive components in two adjoining b-bit bytes will go undetected by a $(kb+b, kb)$ integer $(A_lSB)_b$ code with parity check matrix $H = (C_1\, C_2\, \ldots\, C_k\, -1)$ if $2^b - 1$ divides both $LCM(C_i, 2^{s_1} - 1)$ and $LCM(C_{i+1}, 2^{s_2} - 1)$.*

Proof. Consider an asymmetric solid burst (E_r) of length r occurring in adjoining two b-bit bytes where the first part U_{s_1} of length s_1 of E_r is in i^{th} byte and second part

V_{s_2} of length s_2 in $(i+1)^{th}$ byte. The asymmetric solid burst E_r will go undetected by the integer $(A_lSB)_b$ code if

$$C_iU_{s_1} + C_{i+1}V_{s_2} = 0 \pmod{2^b - 1}. \tag{5.4}$$

As U_{s_1} can be written as $2^{b-s_1}(2^{s_1}-1)$ and V_{s_2} as $2^{s_2}-1$. So, from (5.4), we have

$$C_i2^{b-s_1}(2^{s_1}-1) + C_{i+1}(2^{s_2}-1) = 0 \pmod{2^b - 1}. \tag{5.5}$$

As 2^{b-s_1} and 2^b-1 are relatively prime, and 2^b-1 divides both $LCM(C_i, 2^{s_1}-1)$ and $LCM(C_{i+1}, 2^{s_2}-1)$, so (5.5) is true. Hence such asymmetric solid burst will go undetected. □

5.3.2 Erroneous decoding probability

Since the codes are studied over the Z-channel, we consider the probability of $1 \to 0$ as ϵ and $0 \to 1$ as 0. Theorem below determines the probability of erroneous decoding ($P_d(ASB)$) for integer $(A_lSB)_b$ codes.

Theorem 5.10. *The probability of erroneous decoding $P_d(ASB)$ of a $((k+1)b, kb)$ integer $(A_lSB)_b$ code is* $((k+1)b-l+1)\sum_{i=1}^{l}\epsilon^i(1-\epsilon)^{(k+1)b-i} + \sum_{i=1}^{l-1}(l-i)\epsilon^i(1-\epsilon)^{(k+1)b-i}$.

Proof. To prove this result, we shall use the beginning positions of the asymmetric solid bursts. For an asymmetric solid burst of length 1 beginning from the 1^{st} position of the 1^{st} data byte B_1, the number of non-erroneous bits will be $(k+1)b-1$, so the corresponding probability will be $\epsilon^1(1-\epsilon)^{(k+1)b-1}$. Similarly for an asymmetric solid burst of length 2 beginning from the 1^{st} position of B_1, the probability will be $\epsilon^2(1-\epsilon)^{(k+1)b-2}$, by continuing this, the probability for asymmetric solid burst of length l beginning from the 1^{st} position of B_1 will be $\epsilon^l(1-\epsilon)^{(k+1)b-l}$. For asymmetric solid bursts up to length l occurring anywhere (i.e. within a b-bit byte or spread across two adjoining b-bit bytes), there are $kb+b-l+1$ beginning positions. Thus the probability of erroneous decoding for asymmetric solid bursts beginning from these positions will be $((k+1)b-l+1)\sum_{i=1}^{l}\epsilon^i(1-\epsilon)^{(k+1)b-i}$. Now we are left with a few asymmetric solid bursts in the last b-bit byte C_B (check byte) having length shorter than l occurring after $(b-l+1)^{th}$ position. The probability of asymmetric solid bursts having length at most $l-1$ beginning from $(b-l+2)^{th}$ position in C_B will be $\sum\limits_{i=1}^{l-1}\epsilon^i(1-\epsilon)^{(k+1)b-i}$, similarly for asymmetric solid bursts having length at most $l-2$ beginning from $(b-l+3)^{th}$ position in C_B, the probability will be $\sum\limits_{i=1}^{l-2}\epsilon^i(1-\epsilon)^{(k+1)b-i}$. Continuing this, we end up with asymmetric solid burst of length 1 beginning from the b^{th} position in C_B. Thus the total probability of erroneous decoding

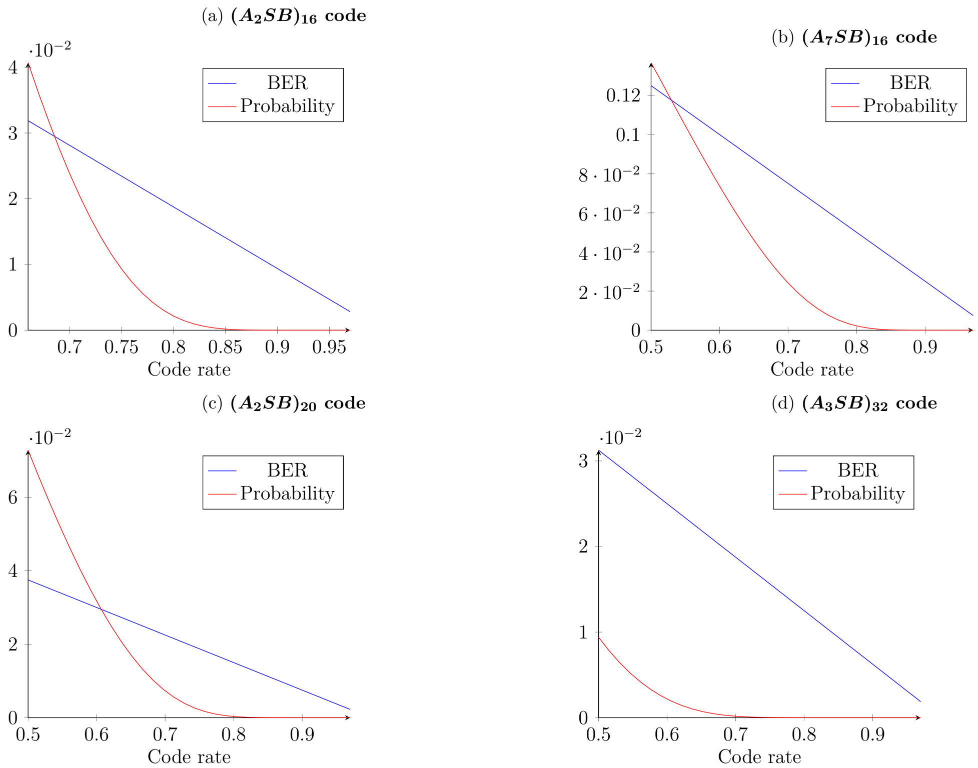
$$
\begin{aligned}
P_d(ASB) = {} & ((k+1)b-l+1)\sum_{i=1}^{l}\epsilon^i(1-\epsilon)^{(k+1)b-i} + \sum_{i=1}^{l-1}\epsilon^i(1-\epsilon)^{(k+1)b-i} \\
& + \sum_{i=1}^{l-2}\epsilon^i(1-\epsilon)^{(k+1)b-i} + \ldots + \sum_{i=1}^{2}\epsilon^i(1-\epsilon)^{(k+1)b-i} + \sum_{i=1}^{1}\epsilon^i(1-\epsilon)^{(k+1)b-i} \\
= {} & ((k+1)b-l+1)\sum_{i=1}^{l}\epsilon^i(1-\epsilon)^{(k+1)b-i} + \{\epsilon^1(1-\epsilon)^{(k+1)b-1} + \epsilon^2(1-\epsilon)^{(k+1)b-2} \\
& + \ldots + \epsilon^{l-1}(1-\epsilon)^{(k+1)b-(l-1)}\} + \{\epsilon^1(1-\epsilon)^{(k+1)b-1} + \epsilon^2(1-\epsilon)^{(k+1)b-2} + \ldots \\
& + \epsilon^{l-2}(1-\epsilon)^{(k+1)b-(l-2)}\} + \ldots + \{\epsilon^1(1-\epsilon)^{(k+1)b-1} + \epsilon^2(1-\epsilon)^{(k+1)b-2}\} \\
& + \{\epsilon^1(1-\epsilon)^{(k+1)b-1}\} \\
= {} & ((k+1)b-l+1)\sum_{i=1}^{l}\epsilon^i(1-\epsilon)^{(k+1)b-i} + (l-1)\epsilon^1(1-\epsilon)^{(k+1)b-1} \\
& + (l-2)\epsilon^2(1-\epsilon)^{(k+1)b-2} + \ldots + (l-(l-1))\epsilon^{l-1}(1-\epsilon)^{(k+1)b-(l-1)} \\
= {} & ((k+1)b-l+1)\sum_{i=1}^{l}\epsilon^i(1-\epsilon)^{(k+1)b-i} + \sum_{i=1}^{l-1}(l-i)\epsilon^i(1-\epsilon)^{(k+1)b-i}.
\end{aligned}
$$

□

Remark 5.11. *If we determine the probability of erroneous decoding for asymmetric solid bursts separately based on the occurrence, i.e., within a b-bit byte and between adjoining b-bit bytes, and add up the probabilities, we will obtain the same result as discussed above.*

Similar to the preceding chapters, the BER here will be $\frac{\frac{1+2+\ldots+l}{l}}{(k+1)b} = \frac{l(l+1)}{2bl(k+1)} = \frac{l+1}{2b(k+1)}$. A few graphs are plotted in Figure 5.3 to analyse the change in probability of erroneous decoding and BER with respect to different code rates, $\epsilon = 0.1$ is considered. In all of the cases it can be observed that both probability $P_d(ASB)$ and BER decrease with the increase in the code rate.

Figure 5.3: **BER and probability vs code rate**



5.3.3 Comparison

To the best of our knowledge, no error-correcting codes have been developed for the discussed type of error over any type of ring. In fact, over the ring $\mathbb{Z}_{2^b-1}$, with byte-oriented codes, no codes have been developed capable of correcting burst errors occurring anywhere in the codeword. Linear codes of dimension $(2k'+1, k'-\frac{t+1}{2})$ are discussed in Result 1.51, which are capable of correcting solid bursts up to length $t+2$. As solid bursts are mainly studied for double and triple adjacent, so we substitute $t = 1$ in this comparison, therefore the dimension will be $(2k'+1, k'-1)$. Drawing the parameters on same lines with equal number of bits to be transmitted before encoding, the proposed codes can transmit the message maintaining significantly lower redundancy. This results to a much higher code rate for the proposed codes. This can be justified by the following argument:

Let $(2k'+1, k'-1)$ be the dimension of the code discussed in Result 1.51 and $((k+1)b, kb)$ be the dimension of the proposed codes, then by equating the number

Table 5.4: **Comparison of some burst and adjacent error-correcting integer codes**

Codes	**Error-correction type**	b	l	LUT_2 **Size**	**No of table look ups**
(544, 512) (Proposed)	Asymmetric solid bursts	32	3	13.9 KB	$1 \leq \eta_{TL} \leq 12$
(544, 512) Result 1.31	Double and triple adjacent	32	3	7.81 MB	$1 \leq \eta_{TL} \leq 21$
(1056, 1024) (Proposed)	Asymmetric solid bursts	32	6	0.05 MB	$1 \leq \eta_{TL} \leq 13$
(1056, 1024) Theorem 2.2	Asymmetric CT-bursts	32	6	0.26 MB	$1 \leq \eta_{TL} \leq 16$
(1056, 1024) Theorem 4.3	Unidirectional solid bursts	32	6	0.1 MB	$1 \leq \eta_{TL} \leq 15$
(1056, 1024) Result 1.19	Symmetric bursts	32	6	0.52 MB	$1 \leq \eta_{TL} \leq 17$
(1056, 1024) Result 1.39	Only asymmetric bursts	32	6	0.26 MB	$1 \leq \eta_{TL} \leq 16$

of components to be sent, we have $k' - 1 = kb \implies k' = kb + 1$ and $k = \frac{k'-1}{b}$, clearly $k' > b$ and $k + 1 = \frac{k'-1+b}{b}$. So $(k + 1)b = k' + b - 1$, since $k' > b$, so $2k' + 1 > k' + b - 1$. Hence the proposed codes can transmit same number of components with less redundancy. Table 5.5 exhibits a few cases for this comparison.

Table 5.5: **Comparison of code rates**

Codes in Result 1.51		**Proposed codes**	
Dimension	**Rate**	**Dimension**	**Rate**
(35, 16)	0.46	(24, 16)	0.66
(57, 27)	0.47	(36, 27)	0.75
(203, 100)	0.5	(110, 100)	0.91
(555, 276)	0.5	(288, 276)	0.96

In Table 5.4, bit requirement and the number of table look ups in a few integer codes capable of correcting bursts and adjacent errors are given; all of the existing codes considered are capable of correcting errors only within a b-bit byte.

5.4 Conclusion

In this chapter, we have constructed a class of integer codes capable of correcting asymmetric solid bursts occurring within a b-bit byte as well as between two adjoining b-bit bytes. Extending the error-correcting capability of integer codes to adjoining b-bit bytes makes this class suitable for implementation in communication channels having multiple bit units. Since the existence of this class depends on computer search results, so to determine a necessary and sufficient condition mathematically for the existence can be considered as a further course of action.