STAR+ Codes: Triple-Fault-Tolerant Codes with Asymptotically Optimal Updates and Efficient Encoding/Decoding

Author : Hanxu Hou, Patrick P. C. Lee
Research unit : School of Electrical Engineering & Intelligentization, Dongguan University of Technology
Department of Computer Science and Engineering, The Chinese University of Hong Kong

STAR+ code 是对于 STAR code 的一种改良编码，它在具有二进制极大距离可分码的两种特性的同时又具有其自身的优化复杂度优势。它通过修改了更新所需的部分信息进而使得解码编码算法发生改变，并降低了复杂度。

INTRODUCTION

二进制极大距离可分码(MDS)是一种纠错码，它具有两种特性：1、当错误情况被限制于一个给定的范围时，所需的冗余消息量是最少的。2、在解码和译码操作中只需要异或操作。确切的说，一个(n,k,m)的二进制 MDS 是一个 m*n 的矩阵，它利用其中的 k*m 个信息比特来获得(n-k)*m 个校验比特。一个(n,k,m)的二进制 MDS 可以检测出 n 列中的任意 r=n-k 列错误。而为了能在例如数据库这样的更新频繁的工作环境下发挥作用，在解码译码过程中异或操作少，在更新过程中平均被影响的校验位少是至关重要的。
EVENODD 码是一种可以检出两列错误的二进制 MDS，而 STAR 码是基于 EVENODD 码进行拓展，有三行校验列且可以检出至多三列错误的一种 MDS，尽管 EVENODD 码和 STAR 码都具有 MDS 的特性，它们的更新复杂度仍不是最优情况。因此我们提出了 STAR+码，在 STAR 码的基础上作出改良，使得它的解码译码复杂度更低，更新复杂度更优。STAR 码在计算后两列校验位时会给每个校验位都加上一个 adjuster bit，而 STAR+码则只会给其中一部分校验位加上 adjuster bit，从而减少了更新复杂度。

CONSTRUCTION OF STAR+ CODES

STAR+码有 r=3 个校验列，而给定一个奇数 m>=k，同时使得对于所有的 l=1,2,...,k-1,gcd(m,l)=1，就可以定义出一个(m-1)*(k+3)的矩阵。其中前 k 列是信息列，后三列被称为校验列。在校验列的计算过程中，我们假设第 m-1 行的前 k 列比特都为 0。对于第 k 列来说，其计算方法为：

$$b_{i,k} = \sum_{j=0}^{k-1} b_{i,j} \text{ for } 0 \le i \le m-2. \qquad (1)$$

对于第 k+1 和 k+2 列来说，计算方法为：

$$b_{i,k+1} = \begin{cases} b_{m-1,k+1} + \sum_{j=0}^{k-1} b_{i-j,j} & \text{for } 0 \le i \le 2\lfloor \frac{k}{2} \rfloor - 1, \\ \sum_{j=0}^{k-1} b_{i-j,j} & \text{for } 2\lfloor \frac{k}{2} \rfloor \le i \le m-2, \end{cases} \qquad (2)$$

$$b_{i,k+2} = \begin{cases} b_{m-1,k+2} + \sum_{j=0}^{k-1} b_{i+j,j} & \text{for } m-1-2\lfloor \frac{k}{2} \rfloor \le i \le m-2, \\ \sum_{j=0}^{k-1} b_{i+j,j} & \text{for } 0 \le i \le m-2-2\lfloor \frac{k}{2} \rfloor, \end{cases} \tag{3}$$

其中出现的第 m-1 行的第 k+1 列和 k+2 列的比特即为 adjuster bit，其计算方法为：

$$b_{m-1,k+1} = \sum_{j=1}^{k-1} b_{m-1-j,j}, \text{ and } b_{m-1,k+2} = \sum_{j=1}^{k-1} b_{m-1+j,j}.$$

例如：

TABLE I: STAR+(9,3), with the adjuster bits $b_{8,4} = b_{7,1} + b_{6,2}$ and $b_{8,5} = b_{0,1} + b_{1,2}$.

0	1	2	3	4	5
$b_{0,0}$	$b_{0,1}$	$b_{0,2}$	$b_{0,0}+b_{0,1}+b_{0,2}$	$b_{0,0}+b_{7,2}+b_{8,4}$	$b_{0,0}+b_{1,1}+b_{2,2}$
$b_{1,0}$	$b_{1,1}$	$b_{1,2}$	$b_{1,0}+b_{1,1}+b_{1,2}$	$b_{1,0}+b_{0,1}+b_{8,4}$	$b_{1,0}+b_{2,1}+b_{3,2}$
$b_{2,0}$	$b_{2,1}$	$b_{2,2}$	$b_{2,0}+b_{2,1}+b_{2,2}$	$b_{2,0}+b_{1,1}+b_{0,2}$	$b_{2,0}+b_{3,1}+b_{4,2}$
$b_{3,0}$	$b_{3,1}$	$b_{3,2}$	$b_{3,0}+b_{3,1}+b_{3,2}$	$b_{3,0}+b_{2,1}+b_{1,2}$	$b_{3,0}+b_{4,1}+b_{5,2}$
$b_{4,0}$	$b_{4,1}$	$b_{4,2}$	$b_{4,0}+b_{4,1}+b_{4,2}$	$b_{4,0}+b_{3,1}+b_{2,2}$	$b_{4,0}+b_{5,1}+b_{6,2}$
$b_{5,0}$	$b_{5,1}$	$b_{5,2}$	$b_{5,0}+b_{5,1}+b_{5,2}$	$b_{5,0}+b_{4,1}+b_{3,2}$	$b_{5,0}+b_{6,1}+b_{7,2}$
$b_{6,0}$	$b_{6,1}$	$b_{6,2}$	$b_{6,0}+b_{6,1}+b_{6,2}$	$b_{6,0}+b_{5,1}+b_{4,2}$	$b_{6,0}+b_{7,1}+b_{8,5}$
$b_{7,0}$	$b_{7,1}$	$b_{7,2}$	$b_{7,0}+b_{7,1}+b_{7,2}$	$b_{7,0}+b_{6,1}+b_{5,2}$	$b_{7,0}+b_{0,2}+b_{8,5}$

在 STAR+ code 中，只需要将 adjuster bit 与$2\lfloor \frac{k}{2} \rfloor$个比特相加即可，而 STAR code 则需要与所有校验比特相加。

DECODING ALGORITHM OF THREE FAILURES

在解码算法中，有一条关键定理

Theorem 1. *For* $\ell = 1,2$, $b_{m-1,k+\ell} = \sum_{i=0}^{m-2}(b_{i,k} + b_{i,k+\ell})$.

根据公式 1 和公式 2，右边的算式可以表达为

$$\begin{aligned} \sum_{i=0}^{m-2}(b_{i,k}+b_{i,k+1}) &= \sum_{i=0}^{m-2}\sum_{j=0}^{k-1} b_{i,j} + \\ &\sum_{j=0}^{k-1}\left(\sum_{i=0}^{2\lfloor \frac{k}{2} \rfloor -1}(b_{i-j,j}+b_{m-1,k+1}) + \sum_{i=2\lfloor \frac{k}{2} \rfloor}^{m-2} b_{i-j,j}\right) \\ &= \sum_{i=0}^{m-2}\sum_{j=0}^{k-1} b_{i,j} + \sum_{j=0}^{k-1}\sum_{i=0}^{m-2} b_{i-j,j}. \end{aligned}$$

再考虑 adjuster bit 的计算公式，以及 m-1 行的前 k 列为 0，可得

$$\sum_{i=0}^{m-1}\sum_{j=0}^{k-1} b_{i,j} + \sum_{i=0}^{m-1}\sum_{j=0}^{k-1} b_{i-j,j} + \sum_{j=0}^{k-1} b_{m-1-j,j} = b_{m-1,k+1}.$$

同理可证 l=2 的情况。

因为 STAR+ code 可以检出并恢复至多三列数据，我们假设出现错误的三列的列号为 f,g,h 且 0<=f<g<h<=k+2，一共可以分为四种情况：

1、三个校验列出错，f = k,g = k+1,h = k+2；

2、两个校验列出错，0 ≤ f ≤ k−1 并且 k ≤ g < h ≤ k +2；

3、一个校验列出错，$0 \leqslant f < g \leqslant k-1$ and $k \leqslant h \leqslant k+2$;
4、无校验列出错，$0 \leqslant f < g < h \leqslant k-1$
对于第一种情况，同编码过程。对于第二种情况，考虑三个类型：(i)
g = k+1,h = k+2, (ii) g = k,h = k+2, and (iii) g = k,h = k+1。这三种类型的恢复方法与EVENODD+编码的相同。
对于第三种情况来说，也有三种情况：(i) h = k+2, (ii) h = k+1, and
(iii) h = k。前两种的恢复方法与 EVENODD+所用方法相同。而对于第三种情况，我们首先提出一条引理

Lemma 2. $b_{m-1,k+1}+b_{m-1,k+2}=\sum_{i=0}^{m-2}(b_{i,k+1}+b_{i,k+2})$.

从 bi,k+1 以及 bi,k+2，i=0,1,...,m-2 中减去所有未损坏的信息列，我们可以得到 2m-2 个 syndrome bits，如下

$$p_{i,1}=\begin{cases} b_{-f+i,f}+b_{-g+i,g}+(b_{-1-f,f}+b_{-1-g,g}) \\ \quad \text{for } i=0,1,\ldots,2\lfloor\frac{k}{2}\rfloor-1, \\ b_{-f+i,f}+b_{-g+i,g} \\ \quad \text{for } i=2\lfloor\frac{k}{2}\rfloor,2\lfloor\frac{k}{2}\rfloor+1,\ldots,m-2, \end{cases} \tag{4}$$

$$p_{i,2}=\begin{cases} b_{f+i,f}+b_{g+i,g} \\ \quad \text{for } i=0,1,\ldots,m-2-2\lfloor\frac{k}{2}\rfloor, \\ b_{f+i,f}+b_{g+i,g}+(b_{-1+f,f}+b_{-1+g,g}) \\ \quad \text{for } i=m-1-2\lfloor\frac{k}{2}\rfloor,\ldots,m-2. \end{cases} \tag{5}$$

而依据引理 2，我们可以将公式 4 和公式 5 相加，从而得到

$$(b_{-1-f,f}+b_{-1-g,g})+(b_{-1+f,f}+b_{-1+g,g})$$

首先考虑 f>0 的情况，对于 i=f-1,i=g-1,i=m-f-1 以及 i=m-g-1 的情况，我们可以得到四个 starting bits

$$p_{f-1,1}=b_{-g+f-1,g}+(b_{-1-f,f}+b_{-1-g,g}) \tag{6}$$

$$p_{g-1,1}=b_{-f+g-1,f}+(b_{-1-f,f}+b_{-1-g,g}) \tag{7}$$

$$p_{m-f-1,2}=b_{g-f-1,g}+(b_{-1+f,f}+b_{-1+g,g}) \tag{8}$$

$$p_{m-g-1,2}=b_{f-g-1,f}+(b_{-1+f,f}+b_{-1+g,g}) \tag{9}$$

从其中选取一个 starting bit，并递归地从公式 4 或公式 5 中抽取一个可以抵消已存在的式子中的一个比特的情况，最后即可得到一个方程，而同时对两个相应的 starting bit 作出上述操作后，即可得到两个损坏信息列的比特，四个 starting bits 可得到四个比特，最终可由这四个比特推出完整的信息列。
考虑 f=0 的情况，当 f=0 时，只有两个 staring bits，只需要按上述方式计算即可。
对于第四种情况，无校验列出错时，我们可用如下方式进行恢复：
1、根据定理 1 计算两个 adjuster bit。
2、计算如下 3p-1 个 syndromes

$$b_{i,f}+b_{i,g}+b_{i,h} \text{ for } 0 \leq i \leq p-2,$$
$$b_{i-f,f}+b_{i-g,g}+b_{i-h,h} \text{ for } 0 \leq i \leq p-1,$$
$$b_{i+f,f}+b_{i+g,g}+b_{i+h,h} \text{ for } 0 \leq i \leq p-1,$$

计算方式为从两个 adjuster bit 和 3(p-1)个校验位中减去 k-3 个未损坏信息列。
3、寻找 g 列的一个 starting point，并以此恢复第 g 列。
4、恢复第 f 和第 g 列的算法与 EVENODD+的相同。

COMPARISON

在文章中，作者也对比了 STAR code 与 STAR+ code 的更新复杂度，如下图

TABLE II: Update complexities of STAR+$(m,7)$ and STAR codes.

m	STAR	STAR+(m,k)	m	STAR	STAR+(m,k)
7	4.4286	4.4286	31	4.6571	3.2857
11	4.5429	3.8571	37	4.6667	3.2381
13	4.4571	3.7143	41	4.6714	3.2143
17	4.6071	3.5357	43	4.6735	3.2041
19	4.6190	3.4762	47	4.6770	3.1863
23	4.6364	3.3896	49	n/a	3.1786
29	4.6531	3.3061	53	4.6813	3.1648

可以看出在 m>7 时 STAR+拥有更低的更新复杂度，同时由于 STAR+ code 无需将 adjuster bit 加给所有校验位，因此 STAR+ code 也有更低的解码/编码复杂度。

CONCLUSION

该文章介绍了一种 STAR 的改良编码，它降低了 STAR 编码的解码编码复杂度，同时令其更新复杂度更接近最优。