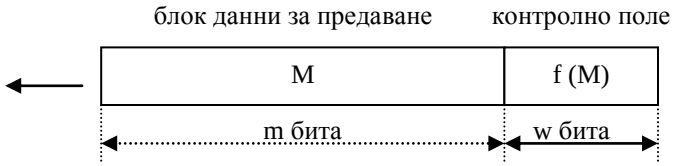
ОТКРИВАНЕ И КОРЕКЦИЯ НА ГРЕШКИ

Основна идея

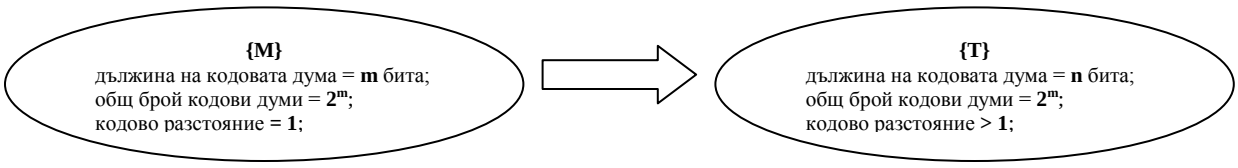
Към всеки обменян блок данни M се добавя информационен излишек, наречен контролно поле, позволяващ на приемника да определи коректността на приетата информация.



В зависимост от предоставените възможности различаваме два типа кодове:

- **коригиращи кодове**- приемникът има възможност да открие точно каква комуникационна грешка е възникнала. Това му позволява да коригира получените грешни данни и да възстанови първоначалния вид на информацията;
- **откриващи кодове**- приемникът може единствено да открие възникването на някаква комуникационна грешка. Коригирането на данните се извършва чрез заявка за повторно предаване на сгрешения блок.

Добавянето на контролното поле води до кодова трансформация от множеството на възможните комбинации от данни **{M}** с дължина на кодовата дума **m** бита, към множеството на предаваните блокове **{T}** с дължина на кодовата дума **n=m+r** бита, при следните характеристики:



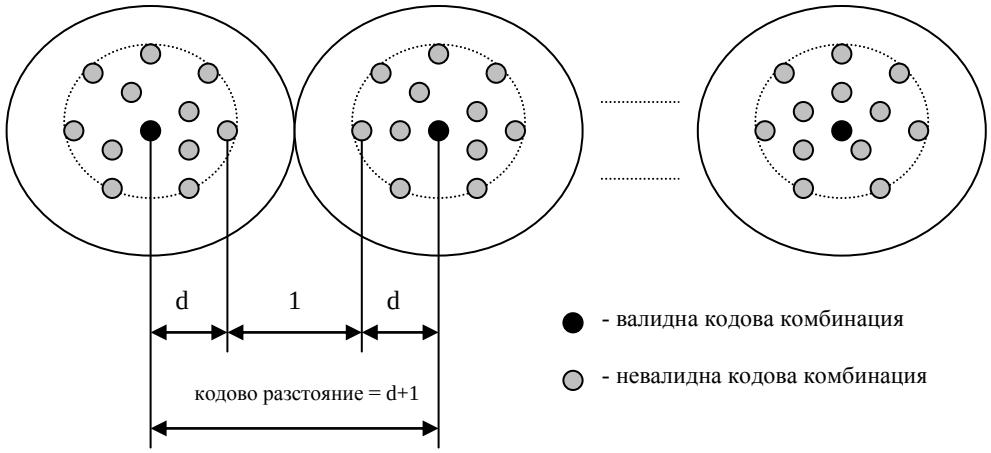
Откриващите и коригиращи свойства на изброените кодове зависят от кодовото разстояние между елементите на множеството {T}:

- кодовото разстояние по Hamming между двойка елементи от произволно множество кодови думи е равно на броя позиции, в които те имат разлики в стойностите на битовете;
- то се изчислява като между двойката елементи се извършва логическа операция **сума по модул 2** (Exclusive OR), след което се преброяват битовете единици от резултата;
- кодовото разстояние за цялото множество е равно на минималното кодово разстояние по Hamming между двойките елементи на множеството.

1. Коригиращи кодове

Допускаме, че в дадена валидна кодова дума са сгрешени стойностите на d на брой бита. Всяка комбинация от d на брой единични грешки ще води до невалидна кодова дума отдалечена на разстояние d. Тогава:

- откриването на d- кратна грешка изисква кодово разстояние d+1, тъй като всички сгрешени валидни кодови думи ще водят като резултат до невалидни кодови комбинации;
- коригирането на d- кратна грешка изисква кодово разстояние 2d+1, тъй като всички сгрешени валидни кодови думи ще водят като резултат до принадлежащи невалидни кодови комбинации;



- корекцията се осъществява като всяка получена невалидна кодова комбинация се причислява към принадлежащата и валидна кодова комбинация, отстояща от нея на растояние $\leq$ d.

Пример за коригиращ двойна грешка код:

{T} = {0000000000, 0000011111, 1111100000, 1111111111}

Корекция на единична грешка

Нека обменяният блок данни има дължина **n = m + r** бита, където **m** е дължината на информационното поле, а **r** е дължината на контролното поле. Общият брой валидни комбинации е $\mathbf{2^m}$, а всяка валидна комбинация има **n** на брой принадлежащи невалидни комбинации (възможна е единична грешка в **n** на брой позиции). Тогава необходимото условие за реализуемост на кода е:

$$\mathbf{(n + 1) . 2^m \leq 2^n}$$

но тъй като **n = m + r,** тогава:

$$\mathbf{(m + r + 1) \leq 2^r}$$

Коригиращ код на Hamming (създаден през 1950 г.)

Позволява корекция на единични грешки в произволно дълги блокове данни. Основни характеристики:

- битовете се номерират последователно от ляво на дясно, започвайки от начална стойност 1;
- битовете с номера равни числа, цели на степени на 2 (1, 2, 4, 8, …) са контролни;
- битовете с останалите номера (3, 5, 6, 7, 9, …) съдържат данните;
- номерът на всеки информационен бит може да бъде разложен на сума от събираеми, цели степени на 2:

  3 = 1 + 2 5 = 1 + 4 6 = 2 + 4 7 = 1 + 2 + 4
  9 = 1 + 8 10 = 2 + 8 11 = 1 + 2 + 8 12 = 4 + 8;
- всеки контролен бит формира допълва до четен или нечетен брой единиците на множество информационни битове с определени номера;
- един информационен бит принадлежи към множеството на даден контролен бит ако съдържа номера му в сумата на която е разложен:

  1 → 3, 5, 7, 9, 11, … 2 → 3, 6, 7, 10, 11 …
  4 → 5, 6, 7, 12 … 8 → 9, 10, 11, 12 … ;

Пример:

Информационно поле	**Информационно + контролно поле**
10110010	10**1**0**011**1**0010**
01100001	11**0**1**110**1**0001**
11100110	10**1**0**110**0**0110**
01110010	11**0**1**111**1**0010**
01011010	00**0**1**10**1**1010**
11100101	01**1**1**110**0**0101**
10111100	10**1**0**011**0**1100**
……	……
00010101	10**0**0**001**0**0101**

Проверка на коректността в приемната страна:

- номера на сгрешената позиция от блока данни се формира в брояч;
- при приемане на блок данни стойността на брояча се установява в нула;
- извършва се последователна проверка на корекността на контролните битове;
- при констатирана некоректност номера на сгрешения бит се добавя към стойността на брояча;
- информацията се счита за коректна, ако след края на проверката стойността на брояча е нула;
- ако брояча не е равен на нула, стойността му сочи сгрешения бит и той може да бъде коригиран.

Корекция на пакети от грешки чрез код на Hamming:

- последователност от **k** на брой кодови думи може да се разглежда като матрица с **k** на брой колони;
- данните се предават последователно по колони;
- матрицата се възстановява в приемната страна;
- извършва се проверка и корекция на единични грешки по редове;
- така е възможна корекция на единична пакетна грешка с дължина **k**, в рамките на цялото съобщение.

2. Откриващи кодове

Най- често използувани са полиномните кодове, наричани също **CRC** кодове (**Cyclic Redundancy Code**). Контролното поле се изчислява чрез полиномна аритметика.

Основни характеристики на полиномната аритметика:

- всяко двоично число се представя чрез полином със степени на **x** равни на позицията на всеки отделен бит и коефициенти равни на стойността на съответния бит;
- например:

$$\mathbf{101101 \Rightarrow 1.x^5 + 0.x^4 + 1.x^3 + 1.x^2 + 0.x^1 + 1.x^0 \Rightarrow x^5 + x^3 + x^2 + 1}$$

- действията събиране и изваждане се извършват без отчитане на преноса:

0 + 0 = 0	**0 - 0 = 0**
0 + 1 = 1	**0 - 1 = 1**
1 + 0 = 1	**1 - 0 = 1**
1 + 1 = 0	**1 - 1 = 0**

- следователно събирането и изваждането се свеждат до едно и също действие - сума по модул 2;
- тогава:

$$\mathbf{x^n + x^n = (1+1).x^n = 0} \qquad \mathbf{x^n - x^n = (1-1).x^n = 0}$$

- понятието естествен ред на числата губи смисъла си, тъй като за някои двойки числа с еднаква разрядност- например 1010 и 1001, не може да се определи по- голямото число:

$$\mathbf{1001 = 1010 + 0011} \quad \text{и} \quad \mathbf{1001 = 1010 - 0011} \;;$$

- тогава, при сравнение на две числа **x** и **y** може да се каже, че:
  - **x** е по- голамо от **y**, ако старшият бит на **x** е по- голям от старшият бит на **y;**
  - **x** се вмества в **y** или **y** се вмества в **x**, ако старшияте битове на двете числа са равни;

- действието деление се извършва, както за определяне на резултата се прави проверка дали делителя се вмества в делимото и след това не се извършва изваждане, а сума по модул 2 :

```
  1 1 0 0 1 1 0 1 : 1 0 0 1 1 = 1 1 0 1
−
  1 0 0 1 1
  ---------
  0 1 0 1 0 1
−
    1 0 0 1 1
    ---------
    0 0 1 1 0 0
−     0 0 0 0 0
      ---------
      0 1 1 0 0 1
−
        1 0 0 1 1
        ---------
          1 0 1 0      ← Остатък
```

Обща постановка на задачата:

- изчислението на стойността на контролното поле се извършва чрез деление в полиномна аритметика;
- съобщението **M**, тълкувано като двоично число, е делимото и се представя чрез полинома **M(x)**;
- делителят е предварително дефиниран, нарича се пораждащ полином и се представя чрез **G(x)**;
- целта е към данните **M(x)** да се добави такава стойност на контролното поле, така че целият предаван блок **T(x)** - поле данни + контролно поле, да се дели без остатък на **G(x)**;
- тоест извършва се кодова трансформация от множеството на кодовите думи **{M(x)}** към множеството на кодовите думи**{T(x)}**, където всяка разрешена кодова дума от **{T(x)}**се дели без остатък на **G(x)**;
- проверката за грешка се извършва като приетите блокове се делят на **G(x)** и получаването на ненулев остатък означава откриване на комуникационна грешка;

Методика за пресмятане на стойността на контролното поле:

- нека **M(x)** е полиномът съотвуващ на предаваното съобщение M, с дължина **m** бита;
- нека **w** е най- високата степен на пораждащият полином **G(x)**;
- към предаваното съобщение M, се прибавят **w** на брой нули (след най-младшия му бит);
- така се получава двоично число с дължима **m + w** бита, представено от полинома $\mathbf{M'(x) = x^w.M(x)}$;

- **M'(x)** се дели в полиномна аритметика на **G(x)**;
- полученият остатъкът **R(x)** представлява стойността на контролното поле;
- той се прибавя към полинома **M'(x)** и резултатът от събирането е полиномът **T(x)**, съответстващ на предавания блок данни:

$$\mathbf{T(x) = M'(x) + R(x)}$$

- полиномът **T(x)** се дели без остатък на **G(x)**, тъй като:

$$\mathbf{T(x) = M'(x) + R(x) = x^w.M(x) + R(x)}$$

но $\mathbf{x^w.M(x) = Q(x).G(x) + R(x)}$
следователно $\mathbf{T(x) = Q(x).G(x) + R(x) + R(x)}$
или $\mathbf{T(x) = Q(x).G(x)}$

Пример: **M = 11001101** **G(x) = $x^4 + x + 1$** **(w = 4)**

делимо: $M' = 2^4 . M = 110011010000$
делител: $G(x) = x^4 + x + 1 = 10011$
остатък: $R(x) = 1110$

$$\begin{array}{r} 110011010000 \\ + \\ \hline 1101 \end{array}$$

Предаван блок: $T(x) = M' + R(x) = \underbrace{110011011101}_{\text{данни}\quad\text{контрол}}$

Деление: **$x^w . M(x) / G(x)$**

```
1 1 0 0 1 1 0 1 0 0 0 0 : 1 0 0 1 1 = 1 1 0 1 1 0 1 1
- 1 0 0 1 1
  0 1 0 1 0 1
-   1 0 0 1 1
    0 0 1 1 0 0
-     0 0 0 0 0
      0 1 1 0 0 1
-       1 0 0 1 1
        1 0 1 0 0
-       1 0 0 1 1
        0 0 1 1 1 0
-         0 0 0 0 0
          1 1 1 0 0
-         1 0 0 1 1
          1 1 1 1 0
-         1 0 0 1 1
          0 1 1 0 1   Остатък
```

Какви грешки могат да бъдат открити?

- нека в резултат на комуникационна грешка предаваният блок данни **T(x)** бъде приет като **T*(x)**;
- тогава:

$$T^*(x) = T(x) + E(x)$$

където E(x) е шумовия вектор на грешката, представен като полином;

- в приемната страна ще бъдат открити всички комуникационни грешки за които **E(x)** не се дели без остатък на **G(x)**, тъй като:

$$T^*(x)/G(x) = T(x)/G(x) + E(x)/G(x) = E(x)/G(x)$$

Следователно пораждащият полином **G(x)** трябва да се избира според характерните за дадена съобщителна среда шумови вектори.

Практически използувани полиноми са:

CRC32 –> 04C11DB7 (Ethernet, FDDI, pkzip, …)
CRC16 –> 1021 (SDLC, X.25, …)

Последователно пресмятане на стойността на контролното поле

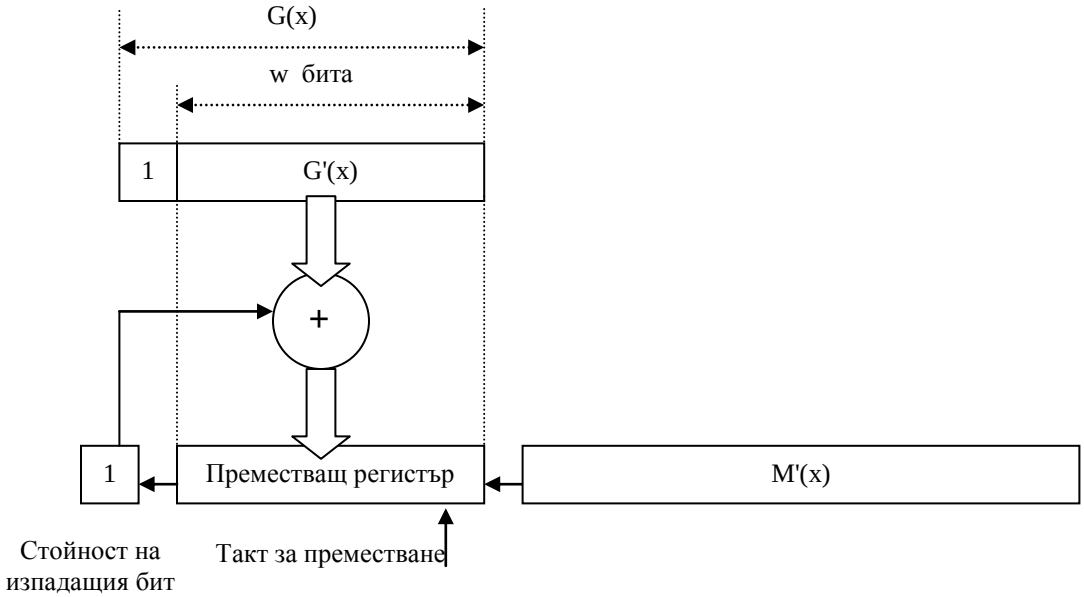


Схема за последователно пресмятане на CRC

Алгоритъм на функциониране:

1. Добавят се **w** на брой нули след младшия бит на съобщението за предаване: **M => M'**.

2. Съобщението **M'** се подава със старшия си бит напред на входа на преместващ регистър с дължина **w** бита ($\mathbf{SR^w}$).

3. $\mathbf{SR^w = 0}$.

4. Преместване в ляво на **1** позиция.

5. Ако изпадналия старши бит от преместващия регистър е равен на 1:

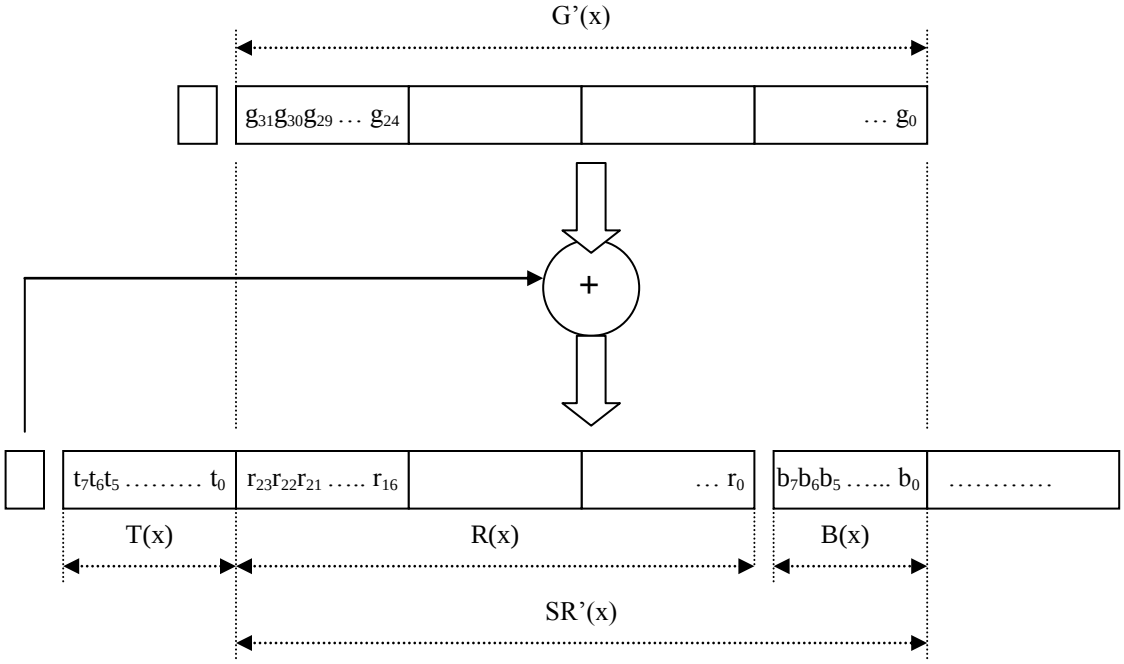
$$\mathbf{SR^w = SR^w \oplus G'(x)}$$

6. Ако M' не е изчерпано, премини към т. 4.
7. Ако M' е изчерпано, $\mathbf{SR^w}$ = стойност на контролното поле

Паралелно пресмятане на стойността на контролното поле

Изчислението на контролното поле се извършва чрез едновременна обработка на няколко бита: тетрада, байт, дума, двойна дума.

Методиката ще бъде илюстрирана за изчисление на контролно поле с дължина 32 бита при паралелна обработка на 1 байт:



Паралелната обработка на 1 байт е реализуема, тъй като е възможно директното изчисление на новата стойност $\mathbf{SR_n}$ на преместващия регистър:

$$\mathbf{SR_n(x) = f(T(x), G'(x), R(x), B(x))}$$

Например стойностите на старшия бит в преместващия регистър след първите две премествания ще са съответно:

$$\mathbf{t_6 \oplus t_7 \,.\, g_{31}}$$

и

$$\mathbf{(t_5 \oplus t_7 \,.\, g_{30}) \oplus (t_6 \oplus t_7 \,.\, g_{31}) \,.\, g_{31}}$$

На базата на този подход е възможно изчислението на всички битове на преместващия регистър **SR**, след всичките 8 последователни премествания. Междинните стойности може да бъдат изразени обобщено по следния начин:

$$\mathbf{SR_1(x) = f_1(T(x), G'(x)) \oplus SR'(x)}$$

$$\mathbf{SR_2(x) = f_2(T(x), G'(x)) \oplus SR_1(x)}$$

$$= f_2(T(x), G'(x)) \oplus f_1(T(x), G'(x)) \oplus SR'(x)$$

$$SR_3(x) = f_3(T(x), G'(x)) \oplus SR_2(x)$$
$$= f_3(T(x), G'(x)) \oplus f_2(T(x), G'(x)) \oplus f_1(T(x), G'(x)) \oplus SR'(x)$$

……..

$$SR_8(x) = f_8(T(x), G'(x)) \oplus SR_7(x)$$
$$= f_8(T(x), G'(x)) \oplus f_7(T(x), G'(x)) \oplus ....\oplus f_1(T(x), G'(x)) \oplus SR'(x)$$

но тъй като:

$$f_8(T(x), G'(x)) \oplus f_7(T(x), G'(x)) \oplus ....\oplus f_1(T(x), G'(x)) = f(T(x), G'(x))$$

тогава:

$$SR_8(x) = f(T(x), G'(x)) \oplus SR'(x)$$

Следователно може да се твърди, че при паралелна обработка на няколко бита, новата стойност в преместващият регистър ще зависи само от старата стойност, новопостъпващата информация и от пораждащият полином G(x). Всички те, както и функционалната зависимост са предварително извесни. Тогава може да се твърди, че съществува константа **k**, за която изчислението се свежда до следната формула:

$$SR_8(x) = k \oplus SR'(x)$$

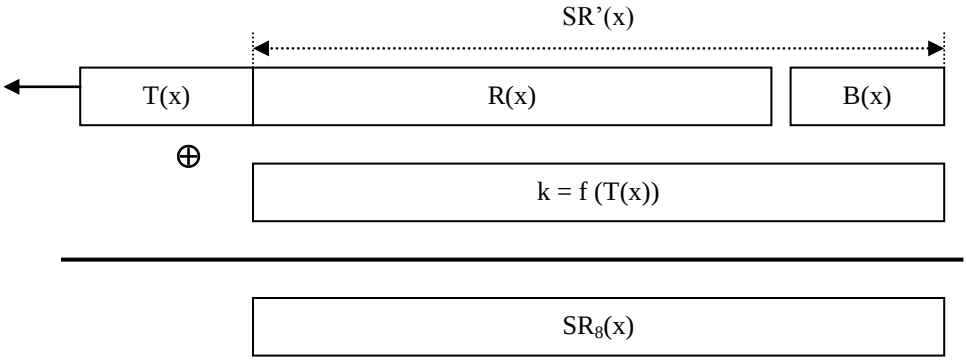


Схема за паралелно пресмятане

Възможните стойности на полинома T(x) при паралелна обработка на 1 байт отговарят на двоичните числа в диапазона от 0 до 255.

Следователно множеството **{K}** на всички константи $k_i=f(T_i(x),G'(x))$ съдържа 256 елемента. Те могат да бъдат предварително изчислени, тъй като $k_i=f'(i)$, където i=0÷255 и аналитичният израз на функцията **f'** е известен.

След като множеството константи е вече известно, най- бързият начин за достигане до стойността на произволна константа k_i на база на неиния номер **i** е

използуването на **hash-** таблица. В нея константите се подреждат така, че всеки ред с номер **i** от таблицата да съдържа стойността на константата $\mathbf{k_I}$.

Изчислението на стойността на контролното поле се извършва с помощта на следната схема:

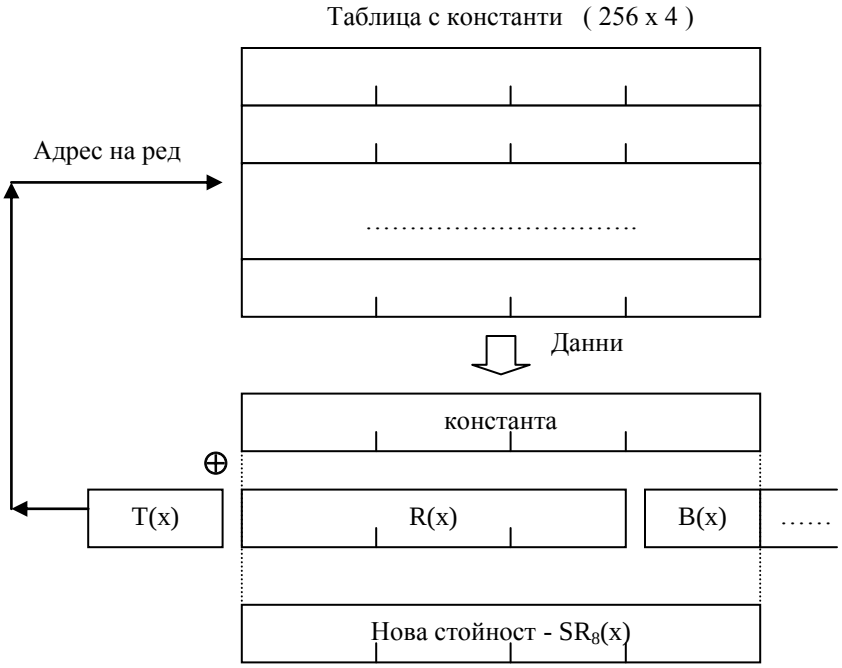


Схема за паралелно пресмятане на CRC

Алгоритъм на функциониране:

1. Добавят се **32** на брой нули след младшия бит на съобщението за предаване **M => M'**.

2. Съобщението **M'** се подава със старшия си бит напред на входа на преместващ регистър с дължина **32** бита (**SR^{32}**).

3. **$SR^{32} = 0$**.

4. Преместване в ляво на **8** позиции.

5. Стойността на изпадналия старши байт служи за адресиране на таблицата с константите и получаване на стойността **k_I**.

6. Извършва се действието:

$$SR^{32} = SR^{32} \oplus k_I$$

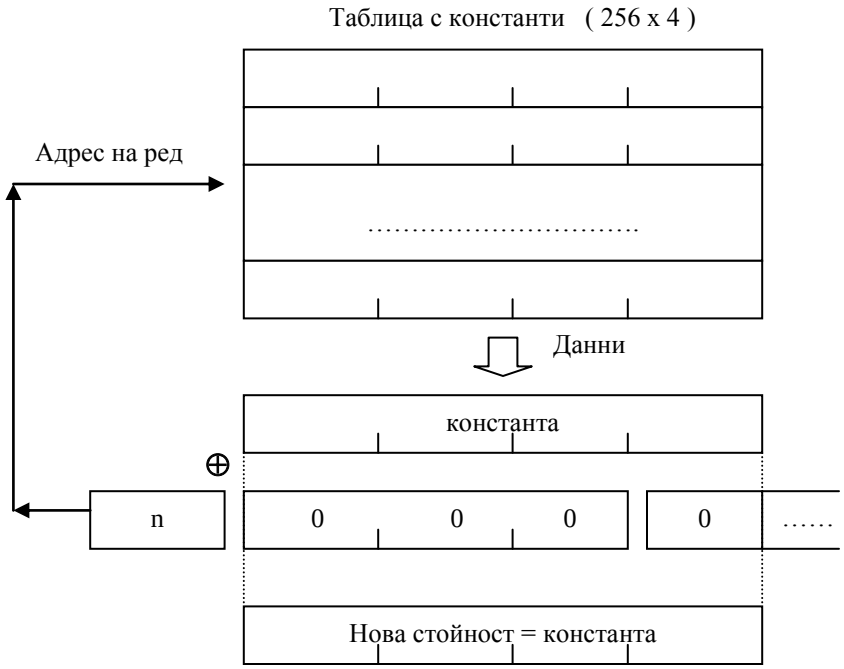
6. Ако M' не е изчерпано, премини към т. 4.

7. Ако M' е изчерпано, **SR^{32}** = стойност на контролното поле

Недостатък на описаната методика от гледна точка на практическа реализация е затрудняващото определяне на стойностите на константите по формулата **k_i=f'(i)**. Аналитичното описание на функцията **f'(i)** е обемистото и сложно, а изчислението трябва да бъде извършено по отделно за всеки бит от преместващия регистър.

Този недостатък може да бъде преодолян, като определянето на стойностите на константите се извършва чрез изчисление по последователния алгоритъм.

Нека допуснем, че при обработка по паралелния алгоритъм, след преместване на 1 байт в ляво, стойността на преместващия регистър е става нулева:



Тогава, след изпълнение на поредния цикъл от изчислението, новополучената стойност в преместващия регистър ще бъде равна на константата, съответстваща на изпадащата стойност.

Следователно ако пораждащият полином **G(x)** е с максимална степен **w** и $\mathbf{T_i(x)}$ е изпадащата стойност при паралелна обработка, стойността на константата $\mathbf{k_i}$ може да бъде получена чрез изчисление на стойността на контролното поле за съобщение $\mathbf{M'(x) = T_i(x).x^w}$ по последователния алгоритъм.