2.2. Надеждност на компютърната обработка

2.2.1. Понятие за код и кодиране

Код е система от символи и правила за еднозначно представяне на данните в КС, а кодирането е функционално преобразуване на данните при това представяне. Ако са зададени две азбуки $A=\{a_1, \ldots, a_n\}$ и $B=\{b_1, \ldots, b_m\}$, то *кодиране* се нарича функцията $F:S\to B^*$, където A^* и B^* са множествата на всички думи над съответните азбуки, а $S\subset A^*$ е подмножество на A^*, включващо *съобщенията* за кодиране. Елементите $\beta=F(s)$, където $s\in S$ и $\beta\in B^*$, се наричат *кодови думи* или накратко кодове на съответните съобщения. Функцията F^{-1} (ако съществува) се нарича *декодиране*. Ако мощността на множеството B е $|B|=m$, функцията F се нарича m-ично кодиране. В КС се използва *двоично кодиране*, където $|B|=2$ и $B=\{0,1\}$.

Типична задача пред теорията на кодирането е при зададени азбуки A, B и множество на съобщенията S да се намери такова кодиране F, което да притежава определени свойства и да е оптимално съгласно зададен критерий. Критерият за оптималност обикновено е минимална дължина на кодовите думи, а свойствата могат да се свържат с изискванията:

- да съществува обратна функция F^{-1} (декодиране);
- да е устойчив на грешки или да има възможност за корекция на грешки;
- да притежава определена простота или достатъчна сложност за функциите F и F^{-1}.

Най-често прилаганото кодиране е *побуквеното*. Съобщенията $s\in S$ се състоят от части (букви) $a\in A$, като всяко съобщение $s=a_1a_2\ldots a_k\in A^*$ има фиксирана дължина $|s|=k$. За всяка буква a_i се дефинира еднозначен уникален код $\beta_i\in B^*$. Побуквеното кодиране се задава чрез кодова таблица:

$$\sigma := <a_1\to\beta_1, a_2\to\beta_2, \ldots a_k\to\beta_k>;\ a_i\in A;\ \beta\in B^*$$

където β_i, се наричат елементарни кодове. Изисква се схемата да осигурява еднозначно кодиране.

Пример: A:={0,1,2,3,4,5,6,7,8,9}; B:={0,1};

σ':=<0→0,1→1,2→10,3→11,4→100,5→101,6→110,7→111,8→1000,9→1001>

Схемата е еднозначна, но не осигурява еднозначно кодиране, защото $F_{\sigma'}(333)=111111=F_{\sigma'}(77)$. За преодоляване на този недостатък се прилага схемата за двоично-десетично кодиране:

σ":=<0→0000, 1→0001, 2→0010, 3→0011, ... , 8→1000,9→1001>

Схемата σ на побуквеното кодиране е *разделима*, ако всяка дума, съставена от елементарни кодове, може да се разложи на елементарните кодове. Побуквено кодиране с разделима схема допуска декодиране.

2.2.2. Отказоустойчиво кодиране, видове грешки и кодово разстояние

При работа на електронните устройства е възможно да възникнат грешки (систематични или случайни). Причина за това могат да бъдат различни външни въздействия, изкривявания на сигнала по линиите за връзка, дефекти в магнитната повърхност на диска и пр. Такива грешки водят до промяна или загуба на предаваните или съхраняваните данни. За намаляване на външните въздействия, както и за откриване на грешки в информацията, се прилагат ***отказоустойчиви кодове***.

Нека C е канал за пренасяне на съобщения, съдържащ източник на грешки. Това може да се запише чрез $s\xrightarrow{C}s'$, където $s\in A^*$ е предаденото съобщение, а $s'\in B^*$ е приетото съобщение. Ако с S и S' се отбележат множествата на предадените и приетите съобщения, то кодиране F, което притежава свойството:

$$S\xrightarrow{F}K\xrightarrow{C}K'\xrightarrow{F^{-1}}S;\forall s\in S;F^{-1}(C(F(s)))=S,$$

се нарича *отказоустойчиво кодиране* или *кодиране с откриване на грешки*.

В КС се използва двоичната азбука, т.е. A=B={0,1} и множествата A^* и B^* съдържат думи над {0,1}. Ако се приеме, че при самото кодиране не се въвежда грешка, то при предаване са възможни следните ***видове грешки*** (с Λ е означена празна дума):

(1) 0→1; 1→0 (промяна на разряд);

(2) 0→Λ; 1→Λ (пропадане на разряд);

(3) Λ→0; Λ→1 (добавяне на разряд).

Канал C се характеризира с горна граница δ_i за броя на грешките от вид (i), които са възможни при предаване на съобщения с определена дължина. Общата оценка се означава с $\delta = <\delta_1, \delta_2, \delta_3>$.

Пример (отказоустойчиво кодиране):
Нека канал C има характеристика $\delta=<1,0,0>$, т.е. възможна е една грешка от първи вид (промяна). Предаваните съобщения са $s=a\in\{0,1\}$.

Кодиране: F(a)=aaa

Декодиране: $F^{-1}(abc)=a+b+c>1$ (мажоритарно възстановяване)

Възможни ситуации:

$$a=1\Rightarrow a\xrightarrow{F}K=aaa=111\xrightarrow{C}K'=110\xrightarrow{F^{-1}}a=[1+1+0>1]\Rightarrow a=1$$

$$a=0\Rightarrow a\xrightarrow{F}K=aaa=000\xrightarrow{C}K'=010\xrightarrow{F^{-1}}a=[0+1+0\le 1]\Rightarrow a=0$$

При двоичното кодиране в КС, използващо ограничен брой разряди на машинната дума, дължината на кодовата дума определя броя на възможните комбинации, които дефинират т.нар. *кодово пространство*. За n-разрядна кодова дума мощността на кодовото пространство A^* над множеството {0,1} е $N=2^n$. Две различни кодови думи $\alpha=\{\alpha_1\alpha_2\ldots\alpha_n\}$ и

$\beta=\{\beta_1\beta_2\ldots\beta_n\}$ от едно кодово пространство се намират на определено ***кодово разстояние*** $d(\alpha,\beta)$ една от друга. Под кодово разстояние (метрика) се разбира броя на различаващите се по стойност разряди за две думи, т.е.

$$d(\alpha,\beta) = \alpha_1\oplus\beta_1 + \alpha_2\oplus\beta_2 + \alpha_3\oplus\beta_3 + \ldots + \alpha_n\oplus\beta_n .$$

Пример: $A=\{0,1\}$; $n=4 \Rightarrow |A^*|=2^4=16$; $\alpha,\beta\in A^*$
α:= <1001>; β:= <0010> $\Rightarrow$ $d(\alpha,\beta) = 3$

Най-малкото кодово разстояние $d_{min} = MIN\{d_j/j=1\div N\}$, осигурено от даден код, се нарича *минимално кодово разстояние* и определя степента на защитеност на кода. За да бъде един код защитен, в неговото кодово пространство трябва да е осигурено $d_{min} \geq 2$. В такъв код е въведен излишък от кодови комбинации, които не се използват като правилни кодови думи. Така множеството A^* на кодовите комбинации се разделя на две непресичащи се подмножества: A^*_1 – на правилните кодови думи; A^*_2 – на неизползваните кодови комбинации.

Ако α е изходното кодирано съобщение, а β е полученото след пренасяне по канал съобщение, то възможни са следните ситуации:

а) $\beta=\alpha$; $\alpha,\beta\in A^*_1$ – пренасяне без грешка;

б) $\beta\neq\alpha$; $\alpha\in A^*_1$ и $\beta\in A^*_2$ – възникване на откриваема грешка;

в) $\beta\neq\alpha$; $\alpha,\beta\in A^*_1$ – възникване на неоткриваема грешка.

Въвеждането на излишък става чрез допълване на информационната дума (от n разряда) с контролно поле (k на брой контролни разряди), с което се увеличава d_{min}. Така се формира кодова дума с дължина n+k разряда. Отказоустойчивите възможности на такъв код се определят от условията:

- $d_{min} \geq r + 1$ - за откриване на r-кратна грешка;
- $d_{min} \geq 2r + 1$ - за откриване и коригиране на r-кратна грешка.

2.2.3. Приложение на отказоустойчиви кодове в КС

Контролиращи кодове

Предназначени са за откриване на грешки при пренасяне и обработка на информацията. Най-често използваните кодове са лесни за формиране и имат малък излишък. Обикновено се прилагат за контролиране на байтовете в паметта и при пренасяне по канали в компютъра.

- ***Двоичен код по четност/нечетност.*** Използва един контролен разряд, добавян към информационното поле X от 8 бита. Това е разделим код, т.е. допуска разпознаване и отделяне на контролното поле (виж т.2.2.1). Кодовата дума може да се запише <$x_7x_6x_5x_4x_3x_2x_1x_0k$>, където:

$$k = \begin{cases} x_0 \oplus x_1 \oplus x_2 \oplus x_3 .... \oplus x_7; & \text{код по четност;} \\ x_0 \oplus x_1 \oplus x_2 \oplus x_3 .... \oplus x_7 \oplus 1; & \text{код по нечетност.} \end{cases}$$

Поддържа d_{min} = 2, което определя откриваемост на единична или нечетно-кратна грешка.

Пример: Информационна дума: X=<10011001>
Контролен бит по четност: k=0
Изпратена кодова дума: α=X;k = <10011001;0>
Получена кодова дума: β=<10011011;0> (с еднократна грешка)
Проверка: k' = 1⊕1⊕0⊕0⊕1⊕1⊕0⊕0⊕1 = 1 ≠ k=0

- ***Код с остатък.*** Двоичен код по mod m (m>2), който е разделим код с дължина на контролното поле $\log_2 m$. Ако информационната част е <$x_0x_1\ldots x_{n-1}$>, то контролното поле се определя като $K=REST[(\Sigma x_j.2^j) \bmod m]$. За двоичен код (p=2) е в сила m=p+1=3, като се осигурява d_{min} = 2.

Пример: от информационна дума X=<00010011> се изчислява $(\Sigma x_j.2^j)$=19 и се определя контролно поле K=REST[(19)mod 3] = 01. Тогава изходната кодова дума ще бъде: α=X;K = <00010011;01>.

Коригиращи кодове

Тези кодове позволяват откриване и коригиране на възникнали при пренасяне грешки, за което се поддържа по-голямо минимално кодово разстояние. Типичен представител е ***код на Хеминг***, който е разделим код с d_{min} = 3 (може да открива и коригира еднократна грешка от тип промяна). При *формиране на кодовата дума* контролните разряди k_i се разполагат на позиции с индекс $i=2^j$ (j=0,1,2,....), т.е. на позиции i = 1, 2, 4, 8 и т.н.:

$$\alpha = <\alpha_m \ldots \alpha_3\alpha_2\alpha_1> = <\alpha_m \ldots \alpha_9k_8\alpha_7\alpha_6\alpha_5k_4\alpha_3k_2k_1> ,$$

където броят k на контролните разряди се определя чрез $2^k \geq m+1$ (при m=n+k), а номерирането е от 1 до m.

Кодирането на информацията се основава на формиране на групи V_i от индекси, които имат стойност '1' за i-тия разряд на двоичните им кодове:

V_1 = 1, 3, 5, 7, 9, 11, ... (с '1' в най-младшия разряд на двоичния им код)
V_2 = 2, 3, 6, 7, 10, 11, ...
V_3 = 4, 5, 6, 7, 12, ...
V_4 = 8, 9, 10, 11, 12, ...

Разрядите от кодовата дума α, които попадат в една и съща група, се сумират по модул 2 за формиране на условията P_j = 0 (j=0, 1, 2, ...), чрез които се определят стойностите за контролните разряди:

$P_0 = k_1\oplus\alpha_3\oplus\alpha_5\oplus\alpha_7\oplus\alpha_9\oplus\alpha_{11}\ldots = 0 \Rightarrow k_1 = \alpha_3\oplus\alpha_5\oplus\alpha_7\oplus\alpha_9\oplus\alpha_{11}\ldots$
$P_1 = k_2\oplus\alpha_3\oplus\alpha_6\oplus\alpha_7\oplus\alpha_{10}\oplus\alpha_{11}\ldots = 0 \Rightarrow k_2 = \alpha_3\oplus\alpha_6\oplus\alpha_7\oplus\alpha_{10}\oplus\alpha_{11}\ldots$
$P_2 = k_4\oplus\alpha_5\oplus\alpha_6\oplus\alpha_7\oplus\alpha_{12}\ldots = 0 \Rightarrow k_4 = \alpha_5\oplus\alpha_6\oplus\alpha_7\oplus\alpha_{12}\ldots$
$P_3 = k_8\oplus\alpha_9\oplus\alpha_{10}\oplus\alpha_{11}\oplus\alpha_{12}\ldots = 0 \Rightarrow k_8 = \alpha_9\oplus\alpha_{10}\oplus\alpha_{11}\oplus\alpha_{12}\ldots$

При *декодиране* на получената кодова дума β се проверява изпълнението на условията P_j = 0 (j=0,1,2,...). При неизпълнение, числото

$$N = \ldots P_3.2^3 + P_2.2^2 + P_1.2^1 + P_0.2^0$$

определя номера на сгрешения разряд в β, който се коригира чрез инвертиране.

Пример: $X=x_8x_7x_6x_5x_4x_3x_2x_1 = 10101001$ (n=8) ; от $2^k \geq n+k+1 \Rightarrow 2^k \geq 9+1 \Rightarrow k=4$.

Кодова дума: $\alpha = \alpha_{12}\ \alpha_{11}\ \alpha_{10}\ \alpha_9\ \alpha_8\ \alpha_7\ \alpha_6\ \alpha_5\ \alpha_4\ \alpha_3\ \alpha_2\ \alpha_1$

$\alpha = x_8\ \ x_7\ \ x_6\ \ x_5\ k_8\ x_4\ \ x_3\ \ x_2\ \ k_4\ x_1\ \ k_2\ \ k_1 = 1010k_8100k_41k_2k_1$

Кодиране: $k_1 = 1\oplus0\oplus1\oplus0\oplus0 = 0$ $\quad k_4 = 0\oplus0\oplus1\oplus1 = 0$

$k_2 = 1\oplus0\oplus1\oplus1\oplus0 = 1$ $\quad k_8 = 0\oplus1\oplus0\oplus1 = 0$

Предадена дума: $\alpha = \alpha_{12}\alpha_{11}\alpha_{10}\alpha_9k_8\alpha_7\alpha_6\alpha_5k_4\alpha_3k_2k_1 = 101001000110$

Приета дума (с грешка в трети разряд): $\beta = 101001000\underline{0}10$

Декодиране: $P_0 = \beta_1\oplus\beta_3\oplus\beta_5\oplus\beta_7\oplus\beta_9\oplus\beta_{11} = 0\oplus0\oplus0\oplus1\oplus0\oplus0 = 1$

$P_1 = \beta_2\oplus\beta_3\oplus\beta_6\oplus\beta_7\oplus\beta_{10}\oplus\beta_{11} = 1\oplus0\oplus0\oplus1\oplus1\oplus0 = 1$

$P_2 = \beta_4\oplus\beta_5\oplus\beta_6\oplus\beta_7\oplus\beta_{12} = 0\oplus0\oplus1\oplus0\oplus1 = 0$

$P_3 = \beta_8\oplus\beta_9\oplus\beta_{10}\oplus\beta_{11}\oplus\beta_{12} = 0\oplus0\oplus1\oplus0\oplus1 = 0$

Корекция: $N = P_3.2^3 + P_2.2^2 + P_1.2^1 + P_0.2^0 = 0.2^3 + 0.2^2 + 1.2^1 + 1.2^0 = 3$

2.2.4. Методи за повишаване на надеждността

В КС се използват различни методи за повишаване достоверността на информацията, като изборът зависи от степента на отговорност на изчисленията. Тези методи използват излишък (допълнителни средства), което повишава стойността на компютърната обработка. Основни методи са следните.

- ***Информационен излишък*** – използване на контролиращи и коригиращи кодове с определена отказоустойчивост, която се дефинира чрез въведеното контролно поле К, допълващо информацията Х до кодова дума W=XK Прилагат се при предаване на данни, като на базата на приетата дума W'=X'K' се изчислява отново стойността K"=f(X') на контролното поле и се сравнява с приетото K'. При съвпадение K"=K' се приема, че грешка не е настъпила.
- ***Апаратен излишък*** – прилагат се при обработващи устройства за изчисляване на аритметични функции C=F(A,B) над защитени данни ($A=XK_X$; $B=YK_Y$; $C=ZK_Z$). Използват се допълнителни апаратни средства за изчисляване на стойността на контролното поле на резултата $K'_Z=f(K_X,K_Y)$ паралелно с извършване на основната операция, след което изчислената контролна стойност K'_Z се сравнява с тази от изчисления резултат $C=ZK_Z$.
- ***Резервиране*** – изпълнение на една и съща операция в две (дублиране) или повече (мажориране) устройства едновременно и следващо сравняване на получените резултати. Ако се открие несъвпадение при дублирането се съобщава за грешка. При мажорирането верният отговор се определя на принципа на гласуването. Тази група методи изисква значителен апаратен излишък, което повишава стойността на компютърната обработка и затова се прилагат при изчисления, които изискват висока надеждност и сигурност.

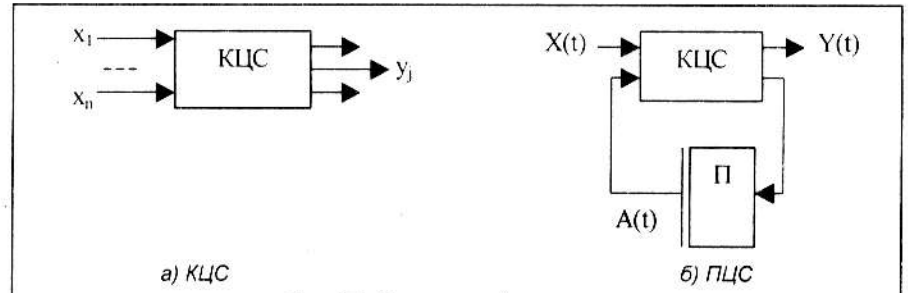
2.3. Организация на изчисленията в КС

2.3.1. Функционално описание на структурни елементи

За изграждане на компютърните устройства и основните блокове се използват цифрови схеми, които могат да се разделят на две групи:

1. *Комбинационни цифрови схеми (КЦС)* – фиг. 2.6.(а), които реализират една или няколко логически функции на n аргумента: $y_j = F_j(x_1, x_2, \ldots, x_n)$; j=1,2,...,k. Изграждат се от логически елементи, които реализират функционален базис (функционално пълна система). Такива базиси са {И, ИЛИ, НЕ}, {НЕ-И}, {НЕ-ИЛИ} и др. Действието на КЦС се състои във формиране на двоични сигнали на изходите y_j в зависимост от постъпващия на входовете ($x_1, x_2, \ldots, x_n$) набор от двоични стойности.
2. *Последователностни цифрови схеми (ПЦС)* – фиг. 2.6(б), чието поведение във времето зависи от постъпващата входна информация X(t), и се описва с последователност от състояния, в които може да се попадне. Включва КЦС и памет (П) за съхраняване на текущото състояние A(t), като функционалното описание се дава от две функции:
   - функция на изходите $Y(t+1) = F_1[X(t), A(t)]$;
   - функция на преходите $A(t+1) = F_2[X(t), A(t)]$.

Такава схема представлява краен автомат, чието поведение може да се опише чрез насочен граф (възлите задават състоянията, а ребрата – преходите между тях).



Фиг. 2.6. Основни цифрови схеми

Структурните елементи, използвани за изграждане на компютърните устройства, попадат в една от посочените групи. По-долу е представено функционалното описание на някои от тях.

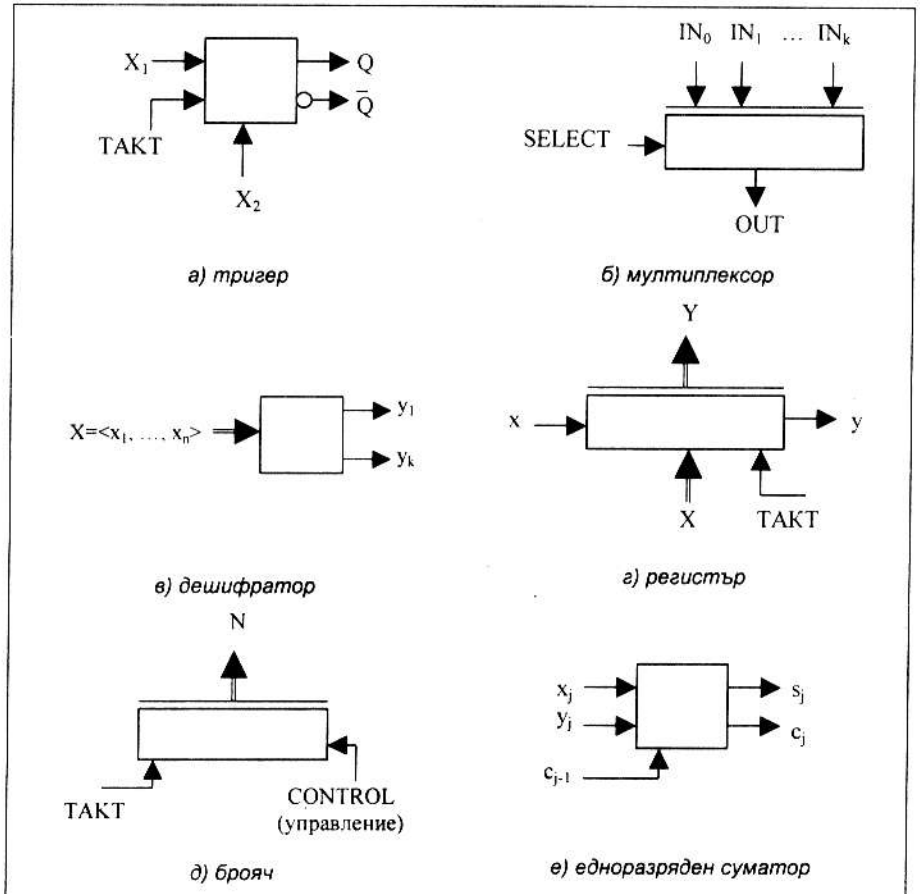
Тригер. Това е елементарна ПЦС, използвана за представяне на един бит информация, чрез установяване на изхода Q в нула или единица в зависимост от текущото състояние на входните въздействия – фиг.2.7а. Функционалното описание е $Q(t+1) = F_1[X_1(t), X_2(t), Q(t), ТАКТ]$, където: $X_1(t)$ е информацията, постъпваща на един или няколко синхронни входа; X_2 са

входове за асинхронно установяване на изхода в 0 или в 1, независимо от синхронните входове X_1; TAKT е синхронизиращ тактов сигнал за управление.

Мултиплексор. КЦС за разпределяне на постъпващата на входовете IN_j информация към изхода OUT в зависимост от управляващия код SELECT (фиг. 2.7б). Функционирането може да се опише чрез:

$$OUT(SELECT=j) = IN_j\ ;\ j \in \{1, 2, \ldots, k\}.$$

Схемата се използва за управляемо разпределяне на информацията от източник към приемник, наречено мултиплексиране на данни.



а) тригер *б) мултиплексор*

в) дешифратор *г) регистър*

д) брояч *е) едноразряден суматор*

Фиг. 2.7. Основни структурни елементи

Дешифратор. КЦС за преобразуване на двоичен код σ:<x_n, ..., x_2, x_1>, постъпващ на входовете, в т.нар. унитарен (единичен) код на изхода. При този код само един изход y_j може да бъде активиран, като j е съответния десетичен еквивалент на двоичния код σ_j. Останалите изходи са дезактивирани. Например, при постъпване на двоичния код 011 ще се активира y_3 от изходите {y_0, y_1, ..., y_7}. Функционалното описание на дешифратора е: $y_j = 1$, ако <x_n, ..., x_2, x_1> = σ_j ; $0 \le j \le 2^n-1$.

Схема, която извършва обратното преобразуване, се нарича ***шифратор***.

Регистър. ПЦС, използвана за временно съхраняване на многоразряден двоичен код (фиг. 2.7г). Изгражда се от тригери и се характеризира с определена дължина n на съхранявания код. Съществуват два основни типа:

- паралелен регистър, при който в рамките на един такт входната информация се предава към изходите: $Y(t+1) = F_1[X(t), TAKT]$;
- последователен (преместващ) регистър, при който постъпващият на входа x двоичен разряд при всеки такт последователно преминава през позициите, преди да се появи на изхода y: $y(t+n) = F_1[x(t), TAKT]$.

Брояч. ПЦС, използвана за преброяване на постъпващите на входа тактови сигнали TAKT (фиг. 2.7д). При всеки тактов сигнал на изхода се формира двоичен код N, който отразява броя на поредицата тактове в зависимост от управлението: $N(t+1) = F_1[TAKT, N(t), CONTROL]$. Чрез управляващите входове се установява посоката на броене и началното установяване ($N \neq 0$) или нулиране ($N=0$) на брояча.

Суматор. Общото предназначение е за извършване на двоично сумиране на две числа (два двоични кода), постъпващи на входовете. Базовата структурна единица е едноразряден суматор (фиг. 2.7е), реализиращ двоично сумиране на два разряда (виж табл. 2.4), при което се формира поразрядна сума $s_j = f_s(x_j, y_j, c_{j-1})$ и пренос $c_j = f_c(x_j, y_j, c_{j-1})$ към следващ разряд.

При многоразрядното сумиране двоичните кодове $X=<x_n \ldots x_2x_1>$ и $Y=<y_n \ldots y_2y_1>$ обикновено постъпват паралелно на входовете, при което се формират поразрядни суми s_j и се разпространяват преносите. За ускоряване на последното се използват специални подходи, с което се намалява времето за установяване на крайната стойност $S=<s_{n+1}s_n \ldots s_2s_1>$ на сумата.

2.3.2. Изпълнение на операциите

Както беше посочено в т. 1.3.3, компютърната обработка има две нива: на машинното изчисление (микро-ниво) и на реализацията на програмните единици (макро-ниво). В този смисъл, всяка операция по компютърната обработка на информацията се представя като фиксирана